# FORTRAN II Y IV DEL PROGRAMADOR: Una Referencia Completa

Por
**CHARLES PHILIP LECHT**
Director, Advanced Computer Techniques Corporation

Con un prólogo por
**ROBERT BEMER**
de General Electric Company

COMPAÑIA EDITORIAL CONTINENTAL, S. A.
MEXICO – ESPAÑA – ARGENTINA

SUCURSALES, DEPOSITOS Y REPRESENTACIONES EN:

Bolivia - Brasil - Colombia - Costa Rica - CHILE - Dominicana - Ecuador - El Salvador
Estados Unidos - Guatemala - Honduras - Nicaragua - Panamá - Paraguay - Perú
Puerto Rico - Uruguay - Venezuela

Título original en inglés:
THE PROGRAMMER'S FORTRAN II AND IV - A Complete Reference

Traductor: ROSENDO JOSE SANCHEZ PALMA
Ingeniero Químico
Coordinador de Sistemas de Control de la IBM de México

Edición autorizada por:
McGRAW-HILL BOOK COMPANY

Primera edición en español: mayo de 1968

COMPAÑIA EDITORIAL CONTINENTAL, S. A.
CALZADA DE TLALPAN NÚM. 4620, MÉXICO 22, D. F.

MIEMBRO DE LA CAMARA NACIONAL DE LA INDUSTRIA EDITORIAL
Registro Núm. 43

AV. REPÚBLICA ARGENTINA NÚM. 168, BARCELONA 6, ESPAÑA
ENTRE RÍOS NÚM. 1256, BUENOS AIRES, ARGENTINA

# PROLOGO

Mi esperanza es que la mayoría de los lectores de este libro ya estén familiarizados con FORTRAN, es decir, más familiarizados que con el simple conocimiento de que es un acrónimo de la frase inglesa "FORmula TRANslation". Esto es por una razón opuesta a la que uno podría pensar inicialmente. No es que el libro sea tan difícil que sólo alguno entrenado en su uso pueda comprenderlo con facilidad. No, mi razón es que FORTRAN es usado extensamente pero escasamente entendido.

Recuerdo esa preocupación molesta con la que inspeccionaba a ciertos estudiantes compañeros en mis cursos de matemáticas en la universidad. Parecían asimilar cada concepto nuevo sin esfuerzo, mientras yo batallaba para relacionarlos a ejemplos numéricos y espaciales. Sin embargo, como matemático profesional descubrí que subconscientemente había estado sintetizando e integrando, mientras ellos meramente añadían capa sobre capa.

Vemos esta superficialidad cuando el usuario de computadoras principiante es expuesto por primera vez al poder tanto del FORTRAN como de la máquina. El lenguaje es relativamente fácil de aprender, puesto que mucho de él es una simple transformación de la terminología y el uso matemático común. El principiante se intoxica con el poder que tiene en la punta de sus dedos, pero que no olvide que es un poder costoso, fácilmente dilapidado sin saberlo. Si Parkinson necesitara pruebas adicionales de sus teorías, existe la comunidad FORTRAN.

Una de las mejores cosas de este libro es que fue escrito por un experto en documentación de computadoras. Esta es una declaración significativa, puesto que menos del 1% de los programadores del mundo documentan cuidadosa y correctamente. Aquellos que lo hacemos podemos estar motivados por los alcances a obtenerse de lo que concebimos; el programador que documenta pobremente pronto es descartado con aversión, lo mismo que sus logros. Un principio cardinal de la documentación en programación es lo que llamamos "la negación positiva". La computadora nos fuerza a esto. No es suficiente decir lo que cierta acción hará; uno debe declarar exacta y categóricamente lo que no hará (i.e., indicar posibles abusos relacionados al contenido de la información, cuyo acaecimiento puede ser causado por inferencia, consideraciones de admisibilidad o aun deducción lógica). Esto bien podría ser la esencia de este libro. No es un texto para enseñarle a programar en FORTRAN—quizá lo aprenda como un subproducto. El propósito es mostrar el alcance completo, el significado y las limitaciones de cada tipo de instrucción en el lenguaje FORTRAN, para que pueda sintetizar en vez de añadir capa sobre capa.

Hay muchos procesadores programados para las diferentes computadoras, cada uno de los cuales traduce una cierta variante de FORTRAN al lenguaje de la máquina con eficiencia variable en el programa objeto creado. Pero en cada variante hay muchas maneras de expresar la solución al problema, de la eficiente a la ineficiente, del mismo modo que uno puede llegar a un punto que esté a dos cuadras de distancia caminando doce, o balbucir y ser prolijo en vez de conciso. Es la responsabilidad de los fabricantes de estos procesadores, explicar tales idiosincrasias al usuario en textos que acompañen a este volumen. Aquí el propósito es más bien preparar al usuario en la forma más general para que entienda el efecto de esas variaciones, según se aplique a computadoras específicas.

No existe más que una breve historia de FORTRAN, dada por W. P. Heisling en el número de marzo de 1963 de "The Communications of ACM", Págs. 85 y 86. Siendo yo una persona conocedora de la historia íntima, me gustaría ponerla aquí por escrito en una forma menos formal.

La computadora impresionó gradualmente sobre sus primeros usuarios algo que debieron comprender desde un principio—que es un invento de aplicabilidad ilimitada, y que una aplicación muy importante podría ser operar sobre la expresión de un algoritmo (o solución de un problema) en un lenguaje conveniente a los humanos y transformarlo en un lenguaje conveniente a la máquina. Recuérdese que el lenguaje natural de los humanos es impreciso; gana comprensibilidad de muchos otros expedientes: inflexiones, movimiento de manos, redundancia, relación a condiciones previas, expresiones alternas u otras semejantes. Un lenguaje intermedio entre los humanos y la máquina, de los que FORTRAN es uno, debe tener ciertas características artificiales que lo haga preciso sin recurrir a tales expedientes.

Es típico de la época de las computadoras que pocas innovaciones son el producto de un solo individuo. Más bien es como si la misma naturaleza de las computadoras nos condujera a todos por un sendero inevitable de entendimiento. Hay pocos desarrollos en lenguajes o procesadores de computadoras que no se encuentren en forma embriónica en los programas anteriores de una docena de personas. Algunos vislumbres iniciales vinieron cuando el Dr. Grace Murray Hopper y sus asociados produjeron el compilador AO (mayo de 1952) para el UNIVAC® I, más tarde ampliándolo al A2 (agosto de 1953) y después al AT3 (o Math-Matic, junio de 1956), el cual tenía una forma limitada de instrucción algebraica. Quizá un trabajador anterior lo fue el Dr. Heinz Rutishauser, de Suiza, quien, ignoto a los trabajadores de los E. U., desarrolló un compilador semejante al FORTRAN para la computadora Zuse 4 en 1951, aunque no recibió un uso extenso apreciable. El Autocode de R. A. Brooker para el Manchester (Ferranti) Mercury, manejaba instrucciones de un tipo aritmético limitado. Laning y Zierler desarrollaron un sistema algebraico para el Whirlwind del M.I.T. alrededor de 1953 o 1954, aunque no he podido fijarle una fecha de operación. También preparando el camino había muchos sistemas matemáticos interpretativos para la IBM 650, la IBM 701 y la Datatron 205.

El Dr. Charles DeCarlo, en aquel tiempo Director de Ciencia Aplicada de IBM, estuvo lo suficientemente impresionado para establecer un grupo de desarrollo bajo John Backus, quien había escrito el sistema Speedcoding para la 701. Este grupo se organizó en el verano de 1954 y llevó su trabajo a una condición utilizable en enero de 1957. El Dr. David Sayre, Robert Beeber, Sheldon Best, Dr. Richard Goldberg, Lois Haibt, Harlan Herrick, R. A. Nelson, Peter Sheridan, Harold Stern e Irving Ziller eran otros miembros, junto con Roy Nutt, de United Aircraft y Robert Hughes, del Livermore Radiation Laboratories.

Este grupo estaba encargado no tan sólo de construir un compilador algebraico, sino también de probar que un compilador podía producir codificación objeto optimizada (programas de trabajo) comparable en eficiencia a aquella de los mejores codificadores manuales. Como resultado, el esfuerzo original tomó 25 años-hombre de esfuerzo sobre un periodo de dos años y medio, con un costo inicial de más de medio millón de dólares. Hoy podemos hacerlo mejor a un octavo del costo, pero el 707 también vuela más rápido que el aeroplano de los Wright. No puede uno decir bastante de la visión del Dr. DeCarlo, puesto que aisló y protegió al grupo FORTRAN por este largo periodo cuando el uso de las computadoras se estaba extendiendo tan rápidamente que cualquier programador bueno era desesperadamente necesitado. También fue una buena decisión, pues hoy IBM tiene un ingreso anual derivado tan sólo del FORTRAN en exceso de 300 millones de dólares, que fue la casi totalidad de sus ingresos en 1957 cuando se introdujo FORTRAN sin ruido o demasiada confianza.

Yo no tuve la oportunidad de participar en el desarrollo del FORTRAN de la 704, ya que el proyecto estaba a medio camino cuando me uní a IBM en diciembre de 1955. Sin embargo, presencié el desarrollo, ya que estaba en otro proyecto en el mismo cuarto. En ese tiempo John Backus fue nombrado Gerente de Investigación de Programación de IBM, bajo el gran repositorio de sabiduría en computadoras—John McPherson.

Unicamente un mes después, más o menos, de la introducción de FORTRAN, el Dr. Alan Perlis terminó un compilador algebraico para la 650, llamado IT (Internal Translator). IT fue originalmente concebido para el Datatron 205, pero ese procesador sufrió un retraso cuando el Dr. Perlis dejó Purdue para ir a Carnegie Tech., entrando en operación hasta el verano de 1957. Aunque los nombres de las variables eran de una forma muy limitada, el método de traducción para tales máquinas inferiores era mucho más ingenioso, y en ese tiempo yo estaba lo suficientemente entusiasmado acerca de la posibilidad de un lenguaje independiente de la máquina para pedirle permiso al Dr. Perlis para usar su sistema encajado en un sistema FORTRAN. El aceptó y se construyó un preprocesador para traducir de instrucciones FORTRAN ligeramente limitadas a instrucciones IT (las cuales producían instrucciones SOAP, las que entonces eran compiladas). Este proyecto estuvo dirigido por Dave Hemmes, quien califica como un verdadero documentador, con Florence Pessin, Otto Alexander y Leroy May. La Sra. Pessin, una aficionada del acróstico doble, bautizó el sistema como FORTRANSIT, teniendo un triple significado: 1) FOR TRANSITion (para transición), 2) FORTRAN, Soap, e IT y 3) it FORTRANS IT (lo "fortranea"). Aunque más tarde lo reemplazó un verdadero procesador FORTRAN, este artificio temporal añadió visiblemente a la prueba de independencia de máquina, particularmente en que una máquina era binaria y la otra decimal.

IBM formó el Departamento de Programación Aplicada bajo Jack Ahlin, cuando el FORTRAN de la 704 iba a ponerse en servicio. Como gerente de los Sistemas de Programación, la operación real en clientes del FORTRAN se convirtió en mi problema, mientras que el grupo de Backus continuaba en investigación para desarrollar lo que se convirtió en el FORTRAN II, que es una buena parte de lo que este libro describe. Y vaya que tuve problemas, porque con 25 000 instrucciones éste era un programa muy complejo para aquel entonces. La mejor indicación de la complejidad es que cuando Sheldon Best dejó el proyecto para ocupar un puesto en el M.I.T. antes de que el procesador estuviera totalmente terminado, les tomó tres meses a los Dres. Sayre y Goldberg, trabajando día y noche, para entender nada más lo que había hecho en su sección 5.

Las unidades de cinta de la 704 presentaron dificultades mayores. Parecía imposible correr FORTRAN en algo diferente a la máquina de pruebas. Finalmente se envió un grupo a la Costa Oeste para que trabajara con los ingenieros de servicio. Cuando finalmente FORTRAN corrió en una máquina diferente, los ingenieros de servicio anotaron cuidadosamente todo lo que habían hecho y prepararon un reporte de cómo ajustar la computadora para que corriera el FORTRAN. Esto marcó el fin de una era para los programas de diagnóstico, puesto que éstos indicaban que la máquina estaba correcta, mientras que el FORTRAN decía que no lo estaba. Además, ¿a quién le interesaba si un componente estaba defectuoso, si el sistema de programación no lo usaba? Armados con estos argumentos, Hemmes y yo fuimos al departamento de Prueba de Productos de Poughkeepsie, donde el Sr. G. A. Hemmer estuvo bastante dispuesto a usar FORTRAN como un componente principal en el programa de prueba y aceptación de la fábrica.

Entonces llegó el tiempo de producir un nuevo FORTRAN para la nueva 709. Entre tanto habíamos aprendido muchas cosas acerca de las necesidades funcionales para un procesador tal. Por ejemplo, en una instalación calcularon que ellos traducían y probaban un programa FORTRAN un promedio de 50 veces antes de que estuviera correcto, funcionando y completo. El tiempo de compilación estaba dejando atrás al tiempo de producción, y un 80% de aquél era para optimizar la utilización de registros índice para programas incorrectos. Obviamente era necesario poder eliminar la optimización.

Puesto que la nueva versión del FORTRAN (FORTRAN II) para la 704 estuvo disponible en junio de 1958, ésta fue la versión que se construyó para la 709, quedando funcionando en junio de 1959. La diferencia principal con el lenguaje original estaba en la habilidad de compilar independientemente subrutinas escritas en FORTRAN o en lenguaje ensamblador, y tenerlas disponibles para los programas FORTRAN principales compilados en otro momento. Esto no sólo ahorraría tiempo de máquina, sino que lógicamente era más sólido. Siempre lo he creído un desarrollo de importancia equivalente al FORTRAN original.

La continuación del FORTRAN II iba a ser el XTRAN, principalmente para quitar ciertas restricciones del lenguaje más que para introducir conceptos radicales. Sin embargo, XTRAN fue más o menos absorbido por el ALGOL 58 en el primer esfuerzo internacional cooperativo sobre lenguajes de programación. Mis frustraciones al perder la iniciativa sobre XTRAN fueron compensadas al serme posible editar el ALGOL 58 hasta tenerlo en una forma relativamente limpia.

El primer procesador FORTRAN no-IBM fue hecho para la Philco 2000, quedando funcionando en abril de 1960; sin embargo, lo llamaron ALTAC. Los primeros procesadores que realmente usaron el nombre FORTRAN fueron los de la UNIVAC Solid-State 80 y de la CDC 1604 en 1961. Para 1965, entre 60 y 100 procesadores FORTRAN habían sido implantados para varias máquinas. El Comité de FORTRAN de SHARE ha sido una influencia estabilizadora, al menos en toda la línea IBM. El lenguaje ensamblador ahora comúnmente asociado con FORTRAN (llamado FAP) se introdujo a través de este comité en septiembre de 1960 y fue el producto del Centro Occidental de Procesamiento de Datos en UCLA.

La más reciente etapa de producción en FORTRAN es el lenguaje FORTRAN IV. Este es un relajamiento adicional de las restricciones y una adición de nuevas características siguiendo las líneas de XTRAN y ALGOL. El comité de FORTRAN de SHARE lo aceptó para la 7094, dándose cuenta de que en varias partes era incompatible con FORTRAN II. Sin embargo, las diferencias eran convertibles mecánicamente. Esta vez no hubo arrastramiento de pies, UNIVAC empezó a construir el FORTRAN IV para la 1107 y en realidad consiguió tenerlo en operación antes que el de la 7094. Actualmente hay unos 10 o 12 cambios propuestos por el comité de SHARE para un FORTRAN IV mejorado. Sin embargo, en vista del NPL (new programming language) para la IBM 360, puede ser que éstos no sean implantados.

Los efectos y la influencia mundial de FORTRAN son asombrosos. Heising dice en su breve historia que se han distribuido más de 228 000 manuales. Espero que el libro del señor Lecht sea reconocido como una de las contribuciones más importantes. Ciertamente llena un vacío del que yo he estado altamente consciente.

ROBERT BEMER

STANFORD UNIVERSITY STANFORD, CALIFORNIA 94305-2085

DONALD E. KNUTH
Fletcher Jones Professor
Department of Computer Science
Telephone [415] 497-4367

March 20, 1984

Dr. Robert Bemer
2 Moon Mountain Trail
Phœnix, Arizona 85203

Dear Bob,

I enjoyed reading your piece about FORTRAN in the recent *Annals*, and I noted especially your story about Stan Poley and the SOAP embroidery.

You're the only person I've ever met who has claimed to have known Stan Poley. Since Poley taught me most of what I know about programming, by publishing the code for SOAP II, I've always considered him a legendary character. Now I can almost believe that he existed. He might even be alive today! It boggles my mind.

I sure hope you will be able to contribute to the forthcoming special issue about the 650. As a person right in the center of the action, your reminiscences are especially crucial.

One way to jog your memory might be for you to reread the paper you gave at Armour Institute (now IIT) in 1957, entitled "The status of automatic programming for scientific problems." It's the only paper written at the time that tried to present a thorough summary of what was happening, as far as I know.

Cordially,

Donald E. Knuth/pw
Donald E. Knuth
Professor

DEK/pw
cc: Cuthbert Hurd

For Share Files P. Lamb,

The first meeting of SHARE - August 22-24, 1955 - Los Angeles.

August 22, 1955

The meeting was opened by a welcome by Paul Armer, our host at Rand Corporation, and was gotten under way by the adoption of an agenda and the election of officers. The officers are: Chairman, Jack Strong; Vice-Chairman, Don Shell; Secretary, Fletcher Jones.

It appeared from the beginning that the bulk of the people present were going into this business of sharing 704 preparation in earnest. There was a first go round at the items on the agenda covering everything briefly. This was all taken care of by three-thirty or so in the afternoon. Some things were completely settled on the first go round but most of the items had to be left for more detailed discussion on the second time through. It was felt that the establishment of the detailed decisions in connection with all items on the agenda would be left for the most part in the hands of the working committee.

It was intended that the working committee be set up using at least one and not more than two, or at most three representatives from any one 704 installation. These people would be left with the responsibility of carrying out the overall decisions of the policy committee which was essentially the committee of the whole. This working committee would decide on specific detailed outlines of things which would be done by each of the installations involved.

I am not going to outline a complete discussion of what went on today because the secretary will provide us with copies of the minutes of the proceedings. However, it seems wise to get down on our own record the things that have already been definitely committed. These are as follows:

[illegible]

1. The "battle of the print wheels" is not yet over as John Backus will undoubtedly learn very soon. It was decided by practically a unamimous vote of the group that the plus sign and the equal sign should be interchanged. It was felt by the group present that it would be possible to accommodate those people who desire echo checking with this system. This could be done by echo checking the sign positions. If a minus sign was received in the echo it would be interpreted as negative, otherwise as positive. This, it seems to me, is not a complete check but ought to be sufficiently good to satisfy every one who wants to echo check. On the other hand we do have the advantage with this arrangement of keeping our plus sign in the twelve row, hence, not completely upsetting the key punching department.

2. The standard binary card format has been changed again. We are now back to the point where column 9, row 9 has no significance on the IBM binary card form. Hence, we will be able to use that column in the manner in which we have used it heretofore. In order to be able to accommodate this change the IBM and United Aircraft people have agreed to move the contents of columns 9 and 10 over to 10 and 11. This permits the same card form to be used by all of the presently existing assembly programs.

3. It was agreed by myself along with all others present that the very least we would do in the way of adopting a standard mnemonic code would be to communicate all items back and forth in the standard code. It appears at this juncture that the IBM code, or something very close to it as it now exists, will ultimately become the standard code. It may be possible to get our friends at IBM to abandon a lot of their other ideas if they are permitted to keep this particular one. The whole thing was left up in the air to a certain extent in today's meeting until the IBM people can find out what they are permitted to give in the way of modifications of their code. If it turns out that they can modify nothing at all it may be that some other code will be adopted. But at any rate when we communicate items to other members of the SHARE group we will do so through the standard code whatever it might be.

These are all of the items on which decisions were made today. It remains to be seen what the decision will be with regard to assembly programs, subroutines and so on. That will be discussed in another report.

August 23, 1955

Two or three definite conclusions were reached by the meeting of the policy committee of SHARE today. First of all it was definitely decided to adopt the IBM mnemonic code as the standard code to be used by the participants in SHARE. As mentioned in yesterday's report I agreed that we would communicate any of our own work in this particular code. However, I do not see where it is feasible for us to abandon our own mnemonic code for our own internal work. An exception which I made to this had to do with the distribution of material which is immediately concerned with CAGE. I proposed that any modifications in CAGE itself which are distributed to other people will be done in our own code. I am certain that this will not inconvenience anyone for reasons which will be apparent in the discussion below.

After a very lengthly discussion, which lasted practically the entire day, it was finally decided by the group to adopt the IBM assembly system as the interim standard system. It is proposed that the group should set up criteria for modifying this program and that it will be modified to incorporate some of the features turned up by the other systems which are not now present in the IBM program.

The following is a summary of the discussion which led up to this final decision. All of the assembly programs which have been written up until now were discussed in fairly complete detail by their originators. These systems included the IBM system, CAGE, a Los Alamos system, and the United Aircraft system. The program written at Los Alamos was the only one which differed in a really fundamental respect from the other three. I am now going to outline the various features of the other programs which do not now appear in CAGE.

1. All three of the other systems included a fairly simple method of providing complements in an operation. It was felt by all of these people that in many cases one really desires to have the complement of locations rather than the location itself. Machine coding requires three or four steps to obtain a complement, therefore, it was felt by these people that the production of complements by the assembly program was a necessary item. Me too AME

Good

2. The only major difference between CAGE and the United Aircraft system is the ability of the United Aircraft program to handle compound addresses and decrements. That is, an address may be made up of the algebraic combination of several symbolic locations. This result may be obtained by adding two or more symbols, subtracting, multiplying, dividing or performing any group of these operations so long as no parentheses are needed to specify the operations.

Good

3. The major difference between the New York IBM assembly program and CAGE is that their program makes up the symbol tables from those symbols occurring in address or decrement fields. Any symbols appearing in the location field which do not also occur in either an address or decrement field do not appear in the table. This makes it possible to put symbols into a location field for mnemonic reference only.

4. A further difference which the New York program permitted is the use of pure numerical symbols. This permits them to accommodate a larger class of symbols and of course requires a fixed field or something fairly near to that in order to use it. This also permits the sequencing of locations for those people who feel that that is a necessary item. A number of people expressed the feeling that a sequence check of some sort was necessary on the cards, at least external to the assembly program.

5. The Los Alamos system used a fixed format. It permitted numerical symbols only. Their system required that the symbols themselves be in sequence within "blocks".

6. Los Alamos also built into its system the ability to very simply do partial reassemblies. In fact this particular feature dictated a great deal of the method used throughout their system. They managed the partial reassembly feature by causing the program to automatically punch out in binary the information which is stored in the table of symbols or what they use as the equivalent of this. This then would make it possible at a later date to read in this information along with a small piece of the program which then would be completely re-assembled according to the original assignment of locations on the original assembly. This they felt was a necessary item for their applications. It seems that they have many production problems which are changing as time goes along. They feel that they must accommodate these changes in the simplest possible way.

After these four systems had been completely batted about for most of the day and it was finally realized that there was not a great deal to choose among them, the group began to talk in terms of accepting one of the systems as a temporary measure and setting up specifications for modifying the system to include what was felt to be essential items in the other systems. It was decided that whatever system should be adopted that the composite assembly program which would result after the modifications that I have mentioned should do everything that was done by the program that was adopted as a temporary measure. At this point a vote was taken for preference among the various systems. On this vote CAGE and the New York system came out exactly even. The Los Alamos system was completely squashed. The United Aircraft system ran a fairly close third.

It became apparent to me at this point that what with all of the changing that was going to be dictated it was fairly obvious that there was practically nothing to be gained by us having CAGE adopted. It would only mean that we must immediately translate the entire coding of CAGE into the IBM mnemonic code. This would be a real job in itself. In addition it would be necessary that we supervise the modification of CAGE in order to accommodate the suggestions that were going to be forthcoming. It appeared to me that adoption of CAGE by the group would cost us more in time and trouble than it was going to be worth, Especially in view of the fact that the IBM mnemonic code had been practically forced upon the group. So I suggested that it would probably be best for the organization to adopt the New York system in view of the fact that it already contains a greater number of bells and whistles than CAGE does. In addition the fact that they set up their table of symbols in a manner different from the way that we do would make it easier for them to accommodate the Los Alamos sequencing idea.

I did not wish to introduce a negative note into the proceedings. Therefore, I did not say that we would be most unhappy if our system were adopted under these circumstances. But it all turns out well apparently because the New York system was adopted by the group. This will now leave us free to modify CAGE in any way that we see fit.

August 24, 1955

Today's meeting was attended by the members of the "working committee" only. A few things were definitely decided.

The first of these was the final conclusions concerning the mnemonic code to be adopted. The IBM three-letter code is going to be the standard code for communication, except that the read, write and sense instructions are expanded as follows:

Read Instructions

| | |
|---|---|
| RCD | Read Card Reader |
| RDR | Read Drum |
| RPR | Read Printer |
| RTB | Read Tape in Binary |
| RTD | Read Tape in Decimal |

Write Instructions

| | |
|---|---|
| WDR | Write Drum |
| WPR | Write Printer |
| WPU | Write Punch |
| WTB | Write Tape in Binary |
| WTD | Write Tape in Decimal |
| WTS | Write Tape Simultaneously |
| WTV | Write Cathode Ray Tube |

Sense Instructions

| | |
|---|---|
| SLN | Sense Light On |
| SLS | Sense Lights Off |
| SLT | Sense Light Test and Off |
| SPR | Sense Printer |
| SPT | Sense Printer Test |
| SPU | Sense Punch |
| SWT | Sense Switch Test |
| IOD | Input Output Delay |
| CFF | Change Film Frame |

It was also definitely decided that the group should collectively and the various installations separately express their desire for IBM to place additional operations in the 704. The operations which were to be requested are the following:

1. Copy and add logical word
2. Exclusive or
3. Store index in address
4. Place index in address
5. Logical right shift
6. Store tag

In addition the following was going to be requested:

Backspace file, or if this was not available

Sense beginning of file condition on the backspace instruction.

It was felt in connection with this last request that a read backward instruction would be the most desirable one. However, it was realized by practically everyone present that IBM had tried to put a read backward instruction on the 727 tape unit and that this was extremely difficult to do. Hence, it was felt that we would ask for a backspace file instruction which should be a reasonable thing to put on the machine. Along with this it would probably be very desirable to incorporate an automatic skip on a beginning of tape condition for the backspace file instruction. This would correspond to a beginning of file condition on the backspace record instruction which would be the same as the backspace tape is now.

It was also decided that certain conventions would be adopted in connection with subroutines. First of all Index C was always to be used as the linkage index. The subroutine is responsible for retaining the condition of index registers A and B and restoring their contents after the subroutine calculation is finished. The condition of the various triggers in the machine were not to be the concern of the subroutine. It was felt by the majority that the overflow triggers might be used to convey a yes or no result back to the main program but in such instances the subroutines would first reset the triggers themselves. In other words the condition of the various triggers in the machine are in a random state when the subroutine is entered, unless of course the trigger condition is one of the input parameters to a subroutine.

In general it was felt that for one word input to a subroutine the accumulator should be used. If a second word of input is needed the MQ should be used. If more information is needed this should be a part of the calling sequence. Conversely, when only one result comes from the subroutine it should appear in the accumulator; two results the accumulator and MQ. If there is more the calling sequence should specify the locations of the results.

It was decided that when large blocks of data are to be carried along as a group, such as a matrix, that this block should have data determining its demensions as part of the block. For example, a method of doing this in a matrix would be to let the address and decrement of the first word contain the number of rows and columns respectively of the matrix. A suggestion along these lines was to put the number of elements in each row into an identification word for that row. The identification would also include the number of the row for that particular matrix. It was thought that this was more general and could be used to identify large matrices which might be in high speed memory only one row at a time. Another example of this sort of thing would be the identification of the degree of a polynomial. This identification could be contained in the first word of the portion of memory containing the coefficients of the polynomial.

In addition to all of this there was time enough for the various people who had done anything in the way of decimal input and output routines to describe briefly the specifications of the routines that they had written. Nothing was decided concerning these. It seems probable that the west coasters in particular will have to set up specifications for their own routines and perhaps prepare a set for themselves. The general tenor of the discussion indicated that most people were going to want to have something in an input output program that was not contained in any of those discussed.

There was a fair amount of discussion concerning the modification that should be applied to the New York assembly program in order to have it incorporate the various advantages of the other assembly systems. There were three things that it was felt should be added if possible. These are (1) variable field, (2) compound addressing, and (3) easy partial reassembly features.

Methods were discussed concerning the means of accomplishing item three especially. Several suggestions were made which seemed to be feasible and which would make it quite possible to perform partial reassemblies in a simple fashion. I feel quite certain that the suggestions were made in such a way that it will be easy to incorporate the change in the New York system. It looks to me as though this added feature will come without any undue effort.

Items one and two on the other hand are considerably more difficult to accommodate in the New York system. This is especially true of item two, compound addressing. In the first place the concept of variable field and pure numerical symbol are for all practical purposes mutually exclusive. The reason is that in a variable field it is necessary to have some sort of a character in a symbol which can be used to distinguish the symbol from a pure numerical constant. It was, therefore, decided that in order to obtain the feature of compound addressing especially and variable field if possible it should be agreed to put at least one alphabetic character in all symbols.

At this point the discussion pretty well went to seed. There didn't seem to be any very bright ideas for getting the feature of compound addressing into the system as it now stands without in some measure at least destroying the ability to retain the present format. It was hoped all along that any modifications which might be made would leave the resulting program in such a condition that it would be able to assemble cards which are prepared now for assembly on the present system. It seems to me that if the feature of compound addressing and variable fields are added to the system it will almost certainly destroy the program's ability to handle programs written in the present format. The entire matter was left more or less at this point. It is possible that the people at IBM and perhaps others can conceive of methods for circumventing this difficulty. If not this feature will probably not get into the system.

2⁵ TRAINING/EDUCATION

~~My early backgeound was in~~ mathematical calculation, bot bussness, in the service of the aircraft industry. I therefore saw early programming from that view, and indeed that is the primary Fortran side. Later, upon moving to IBM, I interested myself in all forms for programming -- starting Commercial Translator (originally COMTRAN), which was one of the three inputs to COBOL.

My assignment has been to give an impression of the programming scene prior to the advent and spread of Fortran. I shall start with machines that were not in themselves programmable -- the desk calculators. During World War II, the mass of calculations required were done in part by analog duvices. The digital part was accomplisbed via desk calculator -- Friden, Marchant, Monroe. (Figure ?)

The programming involved was of our own minds. The algorithm for square root was firmly burned into my head as well as any set of core storage. So was the process for rotation of coordinates in three dimensions.

TABLES

After the war I left that field for a while, to be an architactural draftsman and set designer for the movies. When I cam back in early 1949, I was introduced to IBm 601s at the RAND Corporation. (Figure ?). These wrought-iron-based monsters could do 5 additions per second. RAND had six -- two of them could also multiply. They were programmed by pbenolic strips, into which the programmer cut notches and slopes to indicate where the concerned fields of data started and ended in the card. (Manual or picture?). The last notch in the strip is where the card was filpped over and stacked, to keep the same sequence. I recall that the manual was even more oonfusing ~~of~~ than those of thoday.

1950 APR.

SEAC was going then. The JOHNNIAC was built at RAND. These were then programmed very closely to mabhine instructions. The volume side of the world was still IBM, and that came on 19?? when RaND got the first 604, with 20 program steps available. L learned the binary system on the graveyard shift by punching single digits in a card, feeding it, an noticing what lights came on. It was so clever I wished I had thought of it.

602
604A

Goodyear? Bosak, ~~[illegible]~~ Goody, Armer, Madden, Brown, Hastings

The 405 tabulator (printer) had taught us how to program with plugboards before that. It did in fact do arithmetic for totals, etc.(IBM Tech, N eWsletters 1-10). The indispenible item was a timing charts to synchronize the program steps, according to when grears and cams gave the impetus. (407 timing chart?) But the 604 had tubes, electronic ones with no air inside (assembly?). Here the successive program steps were the only timing you needed to use. 20 steps were soon found wanting, and 60 became the norm.

Each stage in quite complicated computations became represented by a wired 60-step plugboard. They were seldom generalized and thus reusable. I reckon that I wired somewhere between 700 and 800 such boards. The trick was to do as much computation as possible per boadd. The intermediate storage was all punched cards. (Il lost my only set of glasses one day in an overturn in the ocean. I ran thru a set of cards already punched becuase the porr vision said they were new stock. I had the devil of a time recovering the process).

One of the things I noticed in preparing this paper was the scarcity of published papers on programming methods prior to 1956. There were a few specialized conferences on programming, with proceddings, but people seemed to be either reluctant to tell about their work at the national events like the Joint COmputer Conferences, or else the hardware and application people didn't think it worth a hearing.

used
I ~~have XXXXX~~ Youden's ~~bibliogrpahy of computing (1943963%%?) (ref)~~ to extract XXXxproceedings through 195~~8~~7 (any later would be meaningless in a XXXXXXX context of ~~"Before Fortran"~~), and classified their content:

| ~~Year~~ | ~~Hardware~~ | ~~Applications~~ | ~~Software~~ | ~~Conference~~ |
|---|---|---|---|---|
| 1952 | ~~XXXX 16~~ | 8 | 1 | EJCC |

etc,

~~Of course you recognize~~ that one of the headings ~~didn't exist XX~~ in that ~~period - "software"~~.

In 19?? I felt like being a collector of programming systems. I wrote to everyome I could find to ask for five copies of manuals and associated materials (promotion, etc.). I kept one for myself. Two went to Richard Goodman at APIC, Brighton (later to the BCS library), and two were for ACM, thus starting that Repository.

I also uncovered some mönor statistics in this way, and published them in CACM (Ref)s). Later, when we started standardization of programming languages at the internatioal level, I ran a survey on behalf of ISO/TC97/SC5. The results were published in CACM ? (ref)

Headings

An interetsing side effect occurred. I wrote directly to the authors of these systems wherevere possible. One of the questions was when the system was really operational in a general sense, not just demonstrable. In effect, "lay off the sales pitches and publicity". As an example, the recnet book by Lavington (ref) about the Manhcester computers contained this statement:

However, you will notice that Brooker himself said ????? was the real date.

slide/figure

~~CPC PLUGBOARDS~~
~~BOSAK & ME ACM~~
~~SLIDE? WIRE DIAGRAMS~~

PROGRAMMIN WAS
"GLAMOROUS'

Cys: J. A. N. Lee
IBM Corporation
Dept. M43/D27
555 Bailey Avenue
✓ San Jose, CA 95150

CRMO w/o enc. (2), MS 150
OS-4 File

Distribution:

John Backus
✓ IBM Corporation
91 St. Germain Avenue
San Francisco, CA 94114

Jeanne Adams
NCAR
✓ P. O. Box 3000
Boulder, CO 80303

Bruce Rosenblatt
✓ Standard Oil of California
P. O. Box 3069
San Francisco, CA 94119

Hal Stern
697 West End Ave.
✓ Apt. 10F
New York, NY 10025

Richard Goldberg
✓ IBM Research Division
P. O. Box 218
Yorktown Heights, NY 10598

Robert A. Hughes
✓ Lawrence Livermore National Laboratory
P. O. Box 808
Livermore, CA 94450

John McPherson
P. O. Box 333
Short Hills, NJ 07078

James Sakoda
Director, Social Science Data Center
Brown University
Providence, RI 02912

Herbert S. Bright
✓ 7840 Aberdeen Road
Bethesda, MD 20014

William Heising
IBM Corporation
✓ Dept. E57, Bldg. 702-1
Poughkeepsie, NY 12602

Daniel McCracken
7 Sherwood Avenue
Ossining, NY 10562

Martin Greenfield
Honeywell Information Systems
✓ 300 Concord Road
Billerica, MA 01821

Robert Bemer
3 Moon Mountain Trail
Phoenix, AZ 85023

Roy Nutt
✓ Computer Sciences Corporation
650 N. Sepulveda Blvd.
El Segundo, 90245

Fran Allen
✓ IBM Research Center
Yorktown Heights, NY 10598

Charles Davidson
Dept. of Electrical Engineering
✓ B-544 Engineering Bldg.
University of Wisconsin
Madison, WI 53706

1982
National
Computer
Conference
Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

**CONFERENCE CHAIRMAN**
Russell K. Brown, CDP
3420 Yoakum Boulevard
Houston, Texas 77006
713-524-3420

**GRAM CHAIRMAN**
ard L. Morgan
epartment of Decision Sciences
The Wharton School/CC
Philadelphia, PA 19104
215-243-7731

**PROGRAM VICE-CHAIRMAN**
Eric K. Clemons
Associate Professor of Decision Sciences
The Wharton School/CC
University of Pennsylvania
Philadelphia, PA 19104
215-243-7747

**PROFESSIONAL DEVELOPMENT SEMINARS**
Joseph Campisi
Aetna & Casualty
151 Farmington Avenue
Hartford, CT 06156
203-273-3611

**NCCC LIAISON**
Harvey Garner
University of Pennsylvania
Moore School D-2
Philadelphia, PA 19104
215-243-4787

**PIONEER DAY**
J.A.N. Lee
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
703-961-6931

**PLENARY SESSIONS**
Susan Rosenbaum
& T
C278
O. Box 3509
New Brunswick, NJ 08903
201-457-2664

**VICE-CHAIRMAN (PROGRAM ACTIVITIES)**
Robert Stirling
IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604
914-696-4251

**PROGRAM COMMITTEE**
Gene P. Altshuler
O. Peter Buneman
James E. Emery
Dennis Frailey
Robert Frankston
Randall Jensen
Beverly K. Kahn
Alan N. Smith
Amy D. Wohl

**AFIPS PROJECT MANAGER**
Sam Lippman
AFIPS
1815 N. Lynn Street
Suite 800
Arlington, VA 22209
703-558-3614

Reply to: John A. N. Lee
Pioneer Day Chairman

1982 April 26

Charles Davidson
University of Wisconsin
B-554 Engineering Bldg
MADISON WI 53706

Dear Charlie,

As part of the preparation for PIONEER DAY and as part of the general collection of material for the archives which we are creating for FORTRAN, I have been interviewing some of the people involved in the early years. Mainly this is predicated on my being in their home-town on some other mission and having the time, rather than any planned series of visits. During February I visited Phoenix AZ and took the opportunity to conduct an hour's taped interview with Bob Bemer on the topics of FOR TRANSIT and ALGOL. During that interview Bemer pointed out that he was the author of the PRINT I system which he claimed to be the first load-and-go processor. Knowing that you are preparing a short talk for NCC on the topic of load-and-go systems for FORTRAN, I thought that you might be interested in this comment/claim and would be willing to follow it up with Bob. You both are on the program and thus could complement each other quite nicely on this matter.

Looking forward to seeing you in Houston,
Best wishes,

Jan

John A. N. Lee
Professor

xc: Bob Bemer
2 Moon Mtn. Trail
Phoenix AZ 85023

2 Moon Mountain Trail, Phoenix, AZ 85023
1982 April 28

Prof. Charles Davidson
Dept. of Computer Sciences
University of Wisconsin
Madison, WI 53706

Dear Charlie:

Here is some background information for your NCC talk on the origins of load-and-go systems, as promised via telephone. Attached are four items, marked up for comment:

1. Sample output of the PRINT 1 system for the IBM 705.

2. Pages from the first PRINT 1 manual.

3. Pages from the first production issue of the PRINT 1 manual.

4. Appropriate pages from the Franklin Institute Monograph.

The Output

PRINT stood for "PRe-edited INTerpretive". I have marked the source and object code, the latter being the pre-edited output (the "source" and "object" terms came later with FORTRAN). The fact that the object code is not machine instructions is of no matter -- observe the P-code of Western Digital's PASCAL machine. If I had had today's chips available then, I could have put the 6000 some characters of the executive interpreter into one.

The First PRINT Manual

I went to IBM in 1955 December, specifically to do some sort of scientific coding system for the IBM 705. Even though it was primarily a commercial machine, IBM understood that its customers had both types of needs.

I designed PRINT 1, and presented it on 1956 February 8 at the Western Joint Computer Conference. The single paragraph on system entry did not elaborate on the features. This preliminary manual, quoting the summer of 1956 as the future, was probably put out in May. The system was operational for some customers by 1956 August, for all by September.

This is probably the first mention of the load-and-go feature in PRINT 1. Item 3 shows that execution could be deferred. A backward way of stating it, but my thought was that execution should usually take place right after the object code was obtained, and this is reflected in Item 4 by the word "immediately".

The Intermediate Manual

Note how the cover of the first manual (core planes) corresponds to the early FORTRAN manual, and how this cover corresponds to the production FORTRAN manual. I'm not sure which was issued first. The ISO TC97/5 survey gives a date of 1956 October for the PRINT 1 manual.

A little more detail is given here about the human process. On page 7 it was asserted that this is really a compilation process. On page 45, note the concept of "alter" cards against the master source program on tape. On page 46, executing object code created sometime previously is shown as the exception.

The Franklin Institute Presentation

A memorable, and perhaps historic, occasion. I believe I met Grace Hopper there for the first time. One can note that I reported formally to John Backus. At that time I was concentrating on FORTRANSIT.

The essence of compilation is shown on page 35. I emphasize this because of its relation to most load-and-go systems. Other underlined words will assist an understanding of the load-and-go characteristics. I guess I must have been one of the earliest users of the term "timesharing".

I don't recall any more when I first knew Jack Laffan. Someone might ask him if he was familiar with the PRINT 1 system. I did circulate around Endicott and Poughkeepsie, and many seminars on automatic coding were held within IBM.

Cordially,

Bob

R. W. Bemer

cc: J. A. N. Lee

# Computing prior to FORTRAN

*by* R. W. BEMER
*Honeywell Information Systems*
Phoenix, Arizona

ABSTRACT

The life of the programmer in pre-FORTRAN days is characterized in modern terminology, indicating how strongly FORTRAN has changed the programmer's condition and working habits.

The 25 years since the introduction of FORTRAN covers most of programming as we know it, certainly in volume of usage. To minimize any possible communications gap, I have chosen to describe how it was before that watershed event by means of some of the terminology and buzzwords of today:

1. Conferences and published papers
2. Computer science education
3. Stored programming
4. Structured programming
5. Program portability
6. Performance measurement
7. Communications and timesharing
8. Compilers
9. Data independence
10. Software piece parts
11. Software packages

The technical history of early programming languages has been covered by many authors (it became a popular subject), so I'll confine my contribution to more general areas.

## CONFERENCES AND PUBLISHED PAPERS

Publication of software papers in pre-FORTRAN days was far less prolific than now. And it wasn't yet "software." Papers on software techniques prior to FORTRAN are given,[2-42] as found (mostly) in Youden's "Computer Literature Bibliography 1946 to 1963."[1] They're given in best chronological order. To avoid duplication, sources with multiple papers are referenced separately, and the individual papers are given decimal notation.

Doing an analysis of the paper content of the early Joint Computer Conferences (the only continuing national meetings of that era) yields the counts shown in Table I. The last entry is the meeting at which FORTRAN was presented.

The summary pre-FORTRAN count is that of Table II.

TABLE I—Paper distribution of early JCCs

| Year | JCC | Hard-ware | Appli-cations | Soft-ware |
|---|---|---|---|---|
| 1951 | Eastern | 16 | 2 | 0 |
| 1952 | Eastern | 26 | 0 | 0 |
| 1953 | Western | 8 | 11 | 0 |
| 1953 | Eastern | 18 | 4 | 1 |
| 1954 | Western | 8 | 14 | 0 |
| 1954 | Eastern | 9 | 7 | 2 |
| 1955 | Western | 6 | 16 | 1 |
| 1955 | Eastern | 6 | 9 | 1 |
| 1956 | Western | 18 | 10 | 6 |
| 1956 | Eastern | 29 | 0 | 0 |
| 1957 | Western | 28 | 4 | 3 |

TABLE II—Paper distribution by conference location

| JCC | Hard-ware | Appli-cations | Soft-ware | H/A | H/S |
|---|---|---|---|---|---|
| Eastern | 104 | 22 | 4 | 4.7 | 26.0 |
| Western | 68 | 55 | 10 | 1.2 | 5.5 |
| Total | 172 | 77 | 14 | 2.2 | 12.3 |
| % | 65 | 29 | 5 | | |

## COMPUTER SCIENCE EDUCATION

This was just starting, and in just a few schools. When you hired a programmer then, you didn't ask about a degree in computer science; there weren't any. IBM used its Programmer's Aptitude Test as one screening method, and it worked somewhat, but people had a tendency to read more into it than was warranted.

A lot of us had our own pet questions, for we were taking them off the street. Magazine writers were curious about how one became a programmer. Dave Sayre had been a crystallographer, and Sid Noble and Art Bisguier were hired when I, an ex-movie set designer, advertised for chess players.

Although there may not have been enough collected theories to support specific degrees, the university people were all busy creating courses. The summer sessions at MIT and Michigan brought many practioners together. Language processors were being built there and at Purdue, Pennsylvania, Carnegie Tech, Case, UCLA, and many others.

## STORED PROGRAMMING

Programs have always been "stored programs." The only difference is in where they were stored. In desk calculator days—in our heads. To program the IBM 601, one had to file notches in a phenolic strip, and they were stored in a box or hung on the machine. The IBM 604 was programmed by wires placed in plugboards, and often we stored them for reuse, if they were general enough. More often they were unwired for a new program (I wired about 700–800 60-step boards for the 604).

For the CPC the program was obviously in the cards. Bob Bosak and I devised a card system with 4 different tracks of 3-operand instructions, and so could feed a deck of cards continuously in a loop.

## STRUCTURED PROGRAMMING

Structure in programs is generally ascribed to Wilkes, Wheeler, and Gill,[5] in their book on programming for the EDSAC. The subroutine was the first element of structure, and was generally accepted by programmers, particularly those writing interpretive systems.

We had no DO UNTILs or semaphores at our disposal, but many programs had a structure that's all but forgotten now. It was called "optimum programming," a method of placing sequential instructions just right on a magnetic drum, so they would be ready to read just after the previous instruction was completed.

## PROGRAM PORTABILITY

The first way used to reconcile the differences between two types of computer was to recode the problem. The second way was to write a programmed interpretive emulator for one machine in the code of the other. When this resulted in performance degradation of 100:1 up to 1000:1 it lost a certain amount of favor.[43,44]

The third way was to use the source language of the interpreter and write another interpreter for the second machine. This had some success, because the degradation was often not very high (except for extremely dissimilar machines), and it could even run faster! Several of these were made.[44] If machines of today's speeds had suddenly been introduced then, this may have become commonplace; compilers might have a different role. Even now, after thousands of compilers, interpreters still enjoy a considerable vogue. The fourth way, with different compilers, did not to my knowledge receive substantial usage until FORTRANSIT, and even there the portability path from a 704 to a 650 was difficult because the 650 supported fewer index registers.

## PERFORMANCE MEASUREMENT

Although no hardware instrumentation was available for probes, much performance measurement did occur. It was vital because the computers were too slow for the amount of calculation waiting to be performed. While working at Marquardt, I was chastised one day by my boss, for not shaving. It was caused by being up since the previous morning running a trajectory simulation on the CPC. Under such circumstances, everyone wanted programs to run as fast as they could. That was why the program optimizers for drum machines (like SOAP) were so heavily used.

When the 701 superseded the CPC, the balance between user and machine changed. One man at the RAND Corporation took two years to program a problem that ran in two minutes. He experienced considerable culture shock.

There was competition everywhere to have the fastest program for a given task, quite often a mathematical subroutine. When published, those subroutines always had timing associated so the user could plan wisely. The situation was much the same as in the early days of microcomputers. Jewel work was needed, and the domain was small enough to see and measure something. There was even competition between software and hardware people. The 705 engineers were shocked when a programmed divide ran faster than the hardware instruction —without firmware, they could not program a Newtonian iteration.

I suspect that FORTRAN itself had much to do with the temporary hibernation of performance evaluation. After programming in the other languages, it gave so much power because of the ease of use (and the efficiencies were incorporated for you in the compiler), that the number of user of computers could expand much more rapidly. It wasn't until operating systems came into heavy use that we rediscovered the need to prevent waste.

## COMMUNICATIONS AND TIMESHARING

It wasn't Ethernet, but George Stibitz had tied into a relay computer by way of a Teletype—in 1940. SAGE was one of the first major projects to use direct inputs from communications lines. FORTRAN wasn't available when it began, and couldn't have been used for much of the job if it had, for it wasn't just a scientific problem.

Timesharing was just talk. The first time I find the word appearing is in a J. W. Forgie paper on the input-output system for the Lincoln TX-2 computer, concurrent with the 1957 FORTRAN paper. I proposed such usage in an article the next month; it was suggested that IBM should fire me, because that wasn't in line with their policy.

## COMPILERS

Compilers existed before FORTRAN, but they were all rudimentary in comparison. Grace Hopper, chief pioneer of the concept, might have gone faster further if she had had the type of support given to Backus and his group. IT, A2 and A3 were true compilers, but they avoided interactions and optimization.

## DATA INDEPENDENCE

This concept arose with the commercial compiler languages. Grace Hopper and company wrought the Data Division concept. Scientific languages all stuck to floating point, with integers for loop control.

Data structure was usually built into the program, and it didn't seem important, because hardly any interchange of programs took place between different computers. Even if that were possible one could not necessarily get the same answers due to different hardware characteristics.

## SOFTWARE PIECE PARTS

Piece parts for software first came to attention at the first Software Engineering conference in 1968, proposed by Doug McIlroy. However, Bob Glass makes a convincing case[45] that they were in existence before FORTRAN, certainly via the SHARE organization. Indeed they were necessary to counteract the inefficiencies of working without such compilers.

## SOFTWARE PACKAGES

In the modern sense the software package did not exist, for today they cost money. Before FORTRAN it was unthinkable to sell software, although the packages did exist. They were traded or given away. Examples are several general CPC boards, plus the many 650 packages published in the IBM Technical Newsletter No. 10.[27]

There is no doubt that packages existed. They were source programs for interpretation, not compiled source as today. A buzzword of the times was "abstraction." Douglas Aircraft had a "matrix abstraction," for example.[23] It manipulated matrices and performed combinatory functions. Ergo, if your problem could be expressed in matrix form, it could be solved. So it was urged that all problems be expressed this way, a not altogether natural way of use. But many of today's software packages have similar contortional requirements upon the user.

Codes for nuclear computation also fell in the category of software packages, even if they were exchanged in machine language form. Hundreds of these codes were disseminated.

## SUMMARY

I'm enjoying the developments of today, but my pleasure is a bit spoiled by the terrible waste in software development, and so much poor software. It's tempting to recall Miniver Cheevy, who loved "the medieval grace of iron clothing." Software before FORTRAN could be considered quite medieval, even primitive, but there were certain graces.

From my starting in the computer field in early 1949, until FORTRAN arrived, I was either working too hard to see the Peter Principle in effect, or else it didn't exist in such a virulent form. It was exciting to build software then. We had management support and trust for whatever we thought was possible. The number of levels of management was low, and the control tenuous. I reported to John Backus in FORTRAN days, but never felt the slightest pressure. I looked upon him as a friend, not a menace. So today we have better tools and knowledge, and theories of program correctness and such. I don't think that they have added to the fun and excitement of Computing Prior To FORTRAN!

## REFERENCES

1. Youden, W. W. "Computer Literature Bibliography 1946 to 1963." US Natl. Bur. Standards Misc. Publ. 266, 1965 Mar 31.
2. Wilkinson, J. H. "Coding on automatic digital computing machines." Report Conf. on High Speed Automatic Calculating-Machines, Univ. Math. Lab., Cambridge, England, 1949 Jun 22–25, 28–35.
3. Huskey, H. D. "Semiautomatic instruction on the Zephyr." Proc. 2nd Symp. on Large-scale Digital Calculating Machinery, Cambridge, MA, 1949 Sep 13–16, 83–90, Harvard U. Press, 1951, Annals Vol. 26.
4. Stowe, L. "Programming." Office of Naval Research Seminar on Data Handling and Automatic Computing, Washington, DC, 1951 Feb 26–Mar 6, 79–84.
5. Wilkes, M. V., D. J. Wheeler, S. Gill. "The Preparation of Programs for a Digital Computer." Addison-Wesley Press, Cambridge, MA, 1951.
6. Proc. ACM Conf., 1952 May, Pittsburgh, PA.

6.1 Adams, C. W. "Small problems on large computers." 99–102.

6.2 Lipkis, R. "The use of subroutines on SWAC." 231–234.

6.3 Wheeler, D. J. "The use of subroutines in programmes," 235–236.

6.4 Carr, J. W. III. "Progress of the Whirlwind computer towards an automatic programming procedure." 237–242.

6.5 Hopper, G. M. "The education of a computer." 243–250.

7. Proc. ACM Conf., 1952, Toronto.

7.1 Ridgway, R. K. "Compiling routines." 1–5.

7.2 Isaac, E. J. "Machine aids to coding." 17–28.

7.3 Strachey, C. S. "Logical or non-mathematical programmes." 46–49.

7.4 Bennett, J. M., D. G. Prinz, M. L. Woods. "Interpretative subroutines." 81–87.

8. Rutishauser, H. "Automatische Rechenplanfertigung bei programmgesteuterten Rechenmaschinen." Mitteilung aus dem Institut fur angewandte Mathematik, Basel, 1952, 1–45.
9. Rochester, N. "Symbolic programming." PGEC (IRE Transactions on Electronic Computers), Vol. EC-2, No. 1, New York, 1953 Mar, 10–15. LC Card 57-39723.
10. Hopper, G. M. "Compiling routines." Computers and Automation 2, No. 4, 1953 May, 1–5.
11. Hopper, G. M., J. W. Mauchly. "Influence of programming techniques on the design of computers." Proc. IRE 41, No. 10, 1953 Oct, 1250–54.
12. Bouricius, W. G. "Operating experience with the Los Alamos 701." Proc. Eastern Joint Comput. Conf., 1953 Dec 8–10, 45–47.
13. Bennett, J. M., A. E. Glennie. "Programming for high-speed digital calculating machines." In "Faster than Thought." B. V. Bowden, London, Pitman, 1953, 101–116. LC Card 54-15305.
14. Wilkes, M. V. "The use of a 'Floating Address' system for orders in an automatic digital computer." Proc. Camb. Phil. Soc., 49, Part I, 1953, 84.
15. Laning, J. H., N. Zierler. "A program for translation of mathematical equations for Whirlwind I." Engg. Memo. E-364, M.I.T. Instr. Lab., 1954 Jan.
16. Backus, J. W. "The IBM 701 speed-coding system." J. ACM 1, No. 1, 1954 Jan, 4–6.
17. Symposium on Automatic Programming for Digital Computers, Office of Naval Research, Washington, DC, 1954 May 13–14. LC Card 56-60789 rev.

17.1 G. Hopper. "G. M. Automatic programming definitions." 1–5.

17.2 Moser, N. B. "Compiler method of automatic programming." 15–21.

17.3 Waite, J. "Editing generators." 22–29.

17.4 Goldfinger, R. "New York University compiler system." 30–33.

17.5 Holberton, F. E. "Application of automatic coding to logical processes." 34–39.

17.6 Adams, C. W., J. H. Laning, Jr. "The M.I.T. systems of automatic coding: Comprehensive, Summer Session and Algebraic." 40–68.

17.7 Muller, D. E. "Interpretive routines in the Illiac library." 69–73.

17.8 Gorn, S. "Planning universal semiautomatic coding." 74–83.

17.9 Brown, J. H., J. W. Carr III. "Automatic programming and its development on the MIDAC." 84–98.

17.10 Livingston, H. M. "Automatic programming on the Burroughs Laboratory computer." 99–105.

17.11 Backus, J. W., H. Herrick. "IBM 701 speedcoding and other automatic-programming systems." 106–113.

17.12 Elmore, M. "The LMO edit compiler," 114–116.

17.13 Keller, A., R. A. Butterworth. "Programming for the IBM 701 electronic data processing machine with repetitively used functions." 117–149.

18. Jones, J. L. "A survey of automatic coding techniques for digital computers." M.S. Thesis, M.I.T., 1954 May.
19. Proc. Eastern Joint Comput. Conf. 1954 Dec 8–10.

19.1 Rice, R. Jr. "Why not try a plugboard?" 4–10.

19.2 Krider, L. D. "Applications of automatic coding to small calculators." 64–67.

20. Rutishauser, H. "Some programming techniques for the ERMETH." J. ACM 2, No. 1, 1955 Jan, 1–4.
21. Herbst, E., N. Metropolis, M. B. Wells. "Analysis of problem codes on the MANIAC." M.T.A.C. 9, No. 49, 1955 Jan, 14–20.
22. Hopper, G. M. "Automatic programming of digital computers." Proc. High Speed Comput. Conf., Baton Rouge, LA, 1955 Feb 16, 113–118. LC Card 57-63206.
23. Denke, P. H., I. V. Boldt. "A general digital computer program for static stress analysis." Proc. Western Joint Comput. Conf., 1955 Mar 1–3, 72–78.
24. Bradshaw, T. F. "Automatic data processing methods." Proc. Auto.

Data Proc. Conf., Cambridge, MA, 1955 Sep 8–9, 3–27, Harvard U. Press, 1956.

25. "Automatic programming the A-2 compiler system." Parts 1 and 2, Computers and Automation, 4, Nos. 9 and 10, 1955 Sep and Oct.

26. Electronic Digital Computers and Information Processing, Darmstadt, Germany, 1955 Oct 25–27, F Vieweg, Braunschweig, 1956. LC Card 59-18764.

26.1 Rutishauser, H. "Methods to simplify programming, 5 years work with the Z4 computer" (German), 26–30.

26.2 Samelson, K. "Problems of programming techniques" (German), 141–142.

26.3 Lehmann, M. J. "Automatic computer programming" (German), 143.

26.4 Loopstra, B. J. "Processing of formulas by machines." 146–147.

26.5 Thuring, B. "The automatic programming of Univac by the A-2 compiler system." (German), 154–156.

27. Technical Newsletter No. 10, IBM Applied Science Division, New York 1955 Oct.

27.1 Ruthrauff, R. E. "Symbolic coding and assembly for the IBM Type 650." 5–14.

27.2 Horner, J. T. "Relative programming for the IBM Type 650." 15–27.

27.3 Bosak, R. "Development of a floating decimal abstract coding system (FACS)." 28–30.

27.4 Bemer, R. W. et al. (Lockheed MSD). "A general utility system for the IBM Type 650." 31–48.

27.5 Mandelin, A. R., K. D. Weaver. "A selective automonitoring tracing routine called SAM." 49–62.

27.6 Battin, R. H., R. J. O'Keefe, M. E. Petrick. "The MIT Instrumentation Laboratory automatic coding 650 program." 63–79.

27.7 Titus, C. K. "An integrated computation system for the IBM 650." 80–89.

28. Hume, J. N. P., B. H. Worsley. "TRANSCODE, a system of automatic coding for FERUT." J. ACM 2, No. 4, 1955 Oct, 243–252.

29. T. Gorman, T. P., G. Kelly, R. B. Reddy. "Automatic coding for the IBM 701." J. ACM 2, No. 4, 1955 Oct, 253–261.

30. C. Adams, C. W. "Developments in programming research." Proc. Eastern Joint Computer Conf., 1955 Nov 7–9, 75–79.

31. Gordon, B. "An optimizing program for the IBM 650." J. ACM 3, No. 1, 1956 Jan, 3–5.

32. Proc. High Speed Computer Conference, Baton Rouge, LA, 1956 Feb. LC Card 57-63206.

32.1 Hopper, G. M. "Automatic coding techniques 1955." 6–12.

32.2 Heller, J. "Mathematical service routines." 151–153.

32.3 Perry, D. P. "Specifications for an automatic matrix program." 210–215.

33. Proc. Western Joint Computer Conf. 1956 Feb 7–9.

33.1 Ross, D. T. "Gestalt programming, a new concept in automatic programming." 5–9.

33.2 Grems, M., R. E. Porter. "A truly automatic programming system." 10–20.

33.3 Moncrieff, B. "An automatic supervisor for the IBM 702." 21–25.

33.4 Bemer, R. W. "PRINT I, a proposed coding system for the IBM Type 705." 45–48.

33.5 Goldfinger, R. "The IBM Type 705 Autocoder." 49–51.

33.6 Meek, H. V. "An experimental monitoring routine for the IBM 705." 68–69.

34. Brooker, R. A. "The programming strategy used with the Manchester University Mark I computer." IEEE Conf. on Digital Computer Techniques, Suppl. Part B, Vol. 103, London, 1956 Apr 9–13.

35. Perkins, R. "EASIAC, a pseudo-computer." J. ACM 3, No. 2, 1956 Apr, 65–72.

36. Bauer, W. F. "An integrated computation system for the ERA-1103." J. ACM 3, No. 3, 1956 Jul, 181–185.

37. Blum, E. K. "Automatic digital encoding system II (ADES II)." Proc. ACM Conf. 1956, 29.

38. Chipps, J., M. Koschmann, S. Orgel, A. J. Perlis, J. Smith. "A mathematical language compiler." Proc. ACM Conf. 1956, 31.

39. Symp. Advanced Programming Methods for Digital Computers. 1956 June 28–29, ONR Report ACR-15, 1956 Oct.

39.1 Hopper, G. M. "The interlude 1954 to 1956." 1–2.

39.2 Wegstein, J. H. "Automatic coding principles." 3–6.

39.3 Thompson, C. E. "Development of common language automatic programming systems." 7–14.

39.4 Benington, H. D. "Production of large computer programs." 15–28.

39.5 Jones, F. "SHARE, a study in the reduction of redundant programming efforts through the promotion of inter-installation communication." 29–34.

39.6 Carr, J. W. III, B. Arden, "Advanced programming techniques with smaller computers." 35–38.

39.7 Goldstein, M. "Computing at Los Alamos, Group T-1." 39–44.

39.8 Wells, M. "Coding for the MANIAC." 45–48.

39.9 Holberton, F. E. "Proposed advanced coding system for the UNIVAC-LARC." 49–56.

39.10 Waite, J. H. Jr. "RCA approach to automatic coding for commercial problems." 57–66.

39.11 Selfridge, R. G. "The PACT compiler for the 701." 67–70.

39.12 Blum, E. K. "Automatic digital encoding system II." 71–76.

40. J. ACM 4, No. 4, 1956 Oct.

40.1 Melahn, W. S. "A description of a cooperative venture in the production of an automatic coding system." 266–271.

40.2 Baker, C. L. "The PACT I coding system for the IBM Type 701." 272–278.

40.3 Mock, O. R. "The logical organization of the PACT I compiler." 279–287.

40.4 Miller, R. C. Jr., B. G. Oldfield. "Producing computer instructions for the PACT I compiler." 288–291.

40.5 Hempstead, G., J. I. Schwartz. "PACT loop expansion." 292–298.

40.6 Derr, J. I., R. C. Luke. "Semiautomatic allocation of data storage for PACT I." 299–308.

40.7 Greenwald, I. D., H. G. Martin. "Conclusions after using the PACT I advanced coding technique." 309–313.

41. Symp. on Automatic Coding, Franklin Institute, Philadelphia, PA, 1957 Jan 24–25, Monograph No. 3.

41.1 Petersen, R. M. "Automatic coding at G.E." 3–16.

41.2 Katz, C. "Systems of debugging automatic coding." 17–28.

41.3 Bemer, R. W. "PRINT I, an automatic coding system for the IBM 705." 29–38.

41.4 Kinzler, H. M., P. M. Moskowitz. "The procedure translator, a system of automatic coding." 39–56.

41.5 McGee, R. C. "Omnicode, a common language programming system." 57–70.

41.6 McGinn, L. C. "A matrix compiler for Univac." 71–86.

41.7 Perlis, A. J., J. W. Smith. "A mathematical language compiler." 87–102.

41.8 Yowell, E. C. "A mechanized approach to automatic coding." 103 ff.

42. Proc. Western Joint Computer Conf. 1957 Feb 26–28.

42.1 Grems, M. D., R. K. Smith, W. Stadler. "Diagnostic techniques improve reliability." 172–178.

42.2 Baskus, J. W., R. J. Beeber, S. Best, R. Goldberg, L. M. Haibt, H. L. Herrick, R. A. Nelson, D. Sayre, P. B. Sheridan, H. Stern, I. Ziller, R. A. Hughes, R. Nutt. "The Fortran automatic coding system." 188–197.

42.3 Newell, A., J. C. Shaw. "Programming the Logic theory machine." 230–240.

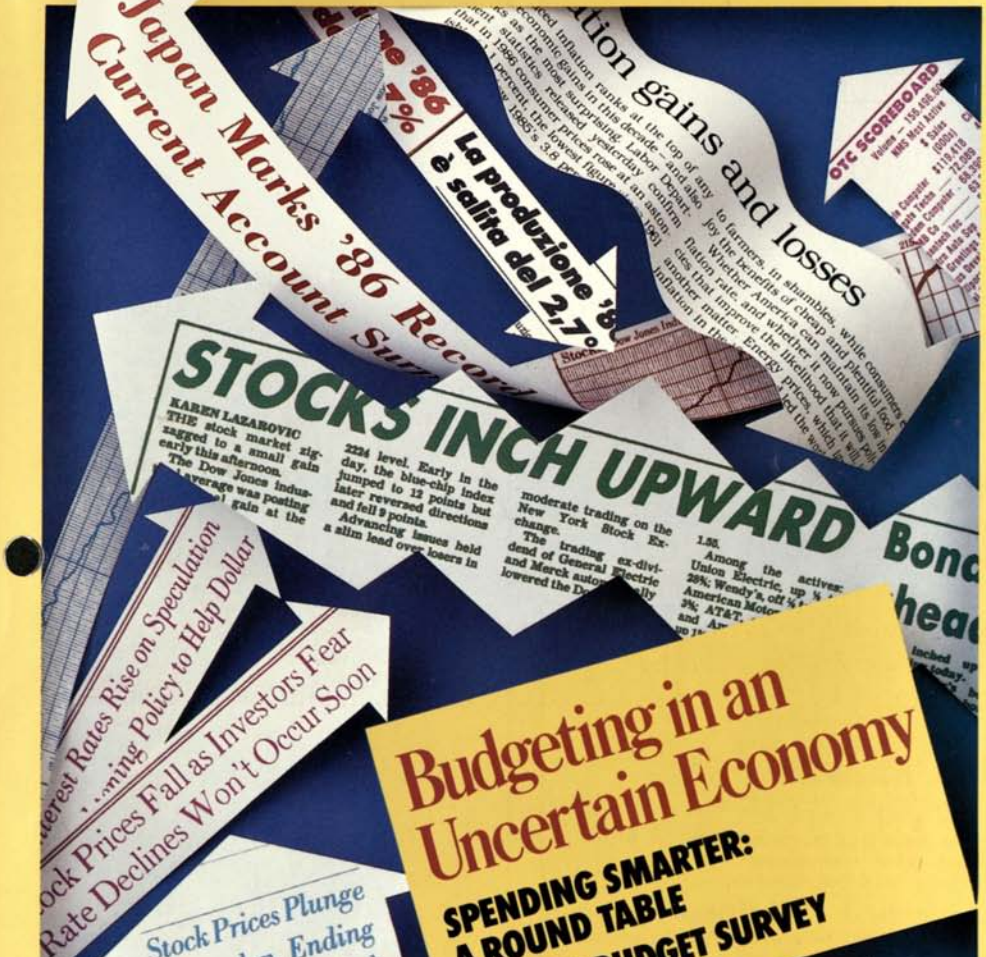
43. Table, Comm. ACM 1, No. 2, 4.

44. Tables, Comm. ACM 1, No. 11, 5–6.

45. Glass, R. L. "Software parts nostalgia." in Reader's Forum, Datamation 27, No. 12, 1981 Nov, 245–247.

# DATAMATION®

APRIL 1, 1987
A CAHNERS PUBLICATION


Japan Marks '86 Record Current Account Surplus
La produzione è salita del 2,7%
...tion gains and losses
STOCKS INCH UPWARD
OTC SCOREBOARD
Bond...
...erest Rates Rise on Speculation ...ning Policy to Help Dollar
...ock Prices Fall as Investors Fear Rate Declines Won't Occur Soon
Stock Prices Plunge In London, Ending Four-Session Climb
...ebbraio 1987
DOLLARO
MARCO

## Budgeting in an Uncertain Economy

**SPENDING SMARTER:**
**A ROUND TABLE**
**1987 DP BUDGET SURVEY**

**PLUS: THE INTEGRATED PROJECT SUPPORT ENVIRONMENT**
**FORTRAN AT 30 MODELING WITH MICROS**

# BUY A RACEHORSE... GET A WORKHORSE!

## NEW!

***Fast—850 cps!***
***And It Runs . . .***
***And Runs . . . And Runs . . .***

The fastest serial dot matrix printer on the market today! The all new ***850XL*** offers a world of benefits!

- Lightning fast at 850 cps (240 lpm throughput)
- Continuous printing capabilities with no overheating or unnecessary downtime!
- Over 120 local service centers nationwide to keep your jobs running day and night!*

The waiting game is over, as the ***850XL*** takes on mountains of data, round the clock, with no duty cycle restrictions! Any printing application you need is handled with rapid-fire reliability:

- Data Processing
- Financials
- Bar Codes
- Spreadsheets
- Near-letter Quality
- Labels
- Graphics

Standard features are better than ever!

- 5 to 18.2 Pitch Printing
- Front Panel Menu Programming (No DIP Switches)
- Quietized Enclosure
- EPSON & DEC Emulations
- 8K Data Buffers
- Serial & Parallel Ports
- Convenient Front & Bottom Paper Feed
- Full International Character Set

***OTC . . . An American Winner!***
Call today for more details.

***1-800-422-4850*** *(8 am - 5 pm PST)*

*Call for availability in your area.

OTC OUTPUT TECHNOLOGY CORPORATION

E. 9922 Montgomery, Suite #6, Spokane, WA 99206
(509) 926-3855. 1-800-422-4850
Telex #15-2269 OUTPUTSPOK FAX #922-4742

CIRCLE 1 ON READER CARD

FORTRAN GRAFFITI: Among the 1953-1957 IBM FORTRAN developers were Roy Nutt, Sheldon Best, Lois Haibt, and David Sayre.

Photograph by Bri[illegible]off

# FORTRAN at 30: Formula for Success

Although it is beginning to give way to other languages in some areas, FORTRAN is still healthy, 30 years after its birth.

BY STEPHEN G. DAVIS

In 1961, Robert W. Bemer, who was then a manager in programming research at IBM, saw just what the computing world needed in the high-level language defined by the international ALGOL 60 committee. "I have enough faith in the eventual future of ALGOL," Bemer wrote in the British Computer Society's *Computer Bulletin*, "to have caused a program to be constructed which converts from FORTRAN source language into a rather stupid ALGOL." As for FORTRAN, an IBM innovation that had begun spreading to other manufacturers' machines, Bemer insisted, "Its purpose has been served."

Thirty years ago this month, IBM delivered its first FORTRAN (formula translation) card deck for the model 704 computer, a binary scientific machine that featured miniaturized vacuum tubes. While slightly past its peak, FORTRAN enters its fourth decade a healthy, vital language, hardly lacking for purposes to serve. Today, FORTRAN is the dominant high-level language in supercomputing and remains the practical standard throughout the scientific and engineering realm. Vendors with significant new FORTRAN releases in the past year range from Cray Research, with its CFT 77, to Microsoft, with MS FORTRAN 4.0.

Meanwhile, the first widely used machine-independent language continues to be modernized. The next, so-called FORTRAN 8X language definition that's due from the American National Standards Institute and the International Standards Organization may enter its public review phase this year. Already, some nonstandard FORTRAN compilers include statements for programming bit-mapped displays and parallel processors—hardware that was barely fathomed in the vacuum tube era.

The reasons FORTRAN first became a de facto standard, according to dp industry veterans and FORTRAN pioneers, were simple and compelling: the language was relatively easy to learn and was available on a variety of machines almost from the start. Above all, FORTRAN compilers typically produced fast code. To this formula for success, today's users add such factors as the wealth of existing programs, the broad base of users who know the language, and—less favorably—inertia.

FORTRAN's broad user base does not come from the business dp side. Only 4% of the IBM mainframe sites polled by

Computer Intelligence, a La Jolla, Calif., research firm, use FORTRAN as a primary language. FORTRAN placed a respectable fourth among all languages in the survey, but far behind COBOL, the choice of 80% of Computer Intelligence's roughly 11,000 IBM mainframe respondents.

Among scientific and engineering users, on the other hand, FORTRAN reigns. Computer Intelligence's latest survey of 9,000 DEC VAX sites, for example, divides fairly evenly between science/engineering and business dp applications; 44% cited FORTRAN as their primary language (COBOL, together with variants like DIBOL, came in second with 16%). Today's public and private sector research centers, which are typically mixed equipment and mixed vendor shops, use FORTRAN in everything from small, ad hoc calculating programs to 100,000-line application systems. Knowledgeable observers have estimated that as much as 25% of the world's available machine cycles run with code generated by some form of FORTRAN.

### Alive and Well at Chevron

"FORTRAN is alive and well in the technical area at Chevron," reports Bruce Rosenblatt, manager of information and systems planning at the San Francisco-based oil company. Use of FORTRAN is certainly below 1960s and 1970s levels at Chevron, but still accounts for "probably two thirds" of the firm's engineering-oriented programming, Rosenblatt estimates.

Rosenblatt, a 36-year-veteran in engineering at Chevron who vividly remembers the impact of the first FORTRAN compilers, suggests that FORTRAN remains perfectly suitable for research applications like seismic processing and testing refinery units. The oil firm runs such applications on a variety of IBM mainframes, Crays, and DEC VAXs. "Most of our use of FORTRAN is on one-shot projects of a research nature, not amenable to higher-level languages," he says. The language is ideal for "compute-intensive" projects, Rosenblatt asserts, because it "let's you get down to machine speed if you need to."

The high quality of the machine code generated is precisely what established FORTRAN compilers in the first place. Indeed, the early FORTRANs didn't compete against other languages, but against other programmers. Their success on this score proved that compilers were feasible—a point that makes IBMer

## Still Crazy After All These Years

FORTRAN, which introduced the GO TO statement into the computer lexicon, has been called an "infantile disorder" by structured programming advocate Edsger Dijkstra. Despite continuing attempts to overlay FORTRAN with constructs borrowed from ALGOL and its more stylistically elegant descendants, the GO TO statement endures. As disorders go, FORTRAN is at least a mature one, as the following chronology shows.

**1953:** John Backus, project manager in programming research at IBM, proposes the FORTRAN idea for the 704 computer in a memo to Cuthbert Hurd, director of applied research.

**1954:** IBM 704 with built-in floating point and indexing capabilities is introduced. Internal version of FORTRAN compiler is produced.

**1956:** First FORTRAN programmers' reference manual is published by IBM.

**1957:** FORTRAN I is released to 704 customers.

First customer-written FORTRAN program is run at Westinghouse-Bettis Atomic Power Laboratory in Pittsburgh.

FORTRAN package for IBM 650 (FORTRANSIT) is released.

**1958:** FORTRAN II and FORTRAN III are released for 704. FORTRAN II, which enables independent compilations of program modules, subroutines, and COMMON blocks for shared variables, soon becomes the industry's de facto standard.

**1960:** Various non-IBM FORTRANs become available, including Seymour Cray's implementation for the CDC 1604, ALTAC for the Philco 2000, Honeywell's Algebraic Compiler, and Automath for the H-800.

**1961:** *A Guide to FORTRAN Programming* by Daniel D. McCracken is published (remains in print until 1986).

IBM releases FORTRAN IV for 7090/4 series.

Other manufacturers begin working on their own FORTRAN IV implementations.

**1962:** The American Standards Association—forerunner of the American National Standards Institute—forms a committee to develop a standard for FORTRAN.

U.S. space probe *Mariner I,* targeted for Venus, explodes after launch at Cape Canaveral; the mishap is later blamed on a misplaced comma in a FORTRAN DO statement.

**1963:** The second commercially published book on FORTRAN appears: *A FORTRAN Primer*, by Elliott I. Organick.

**1964:** IBM announces System/360.

DATAMATION article notes the existence of 43 different FORTRAN compilers for various systems.

**1966:** Standards for FORTRAN and Basic FORTRAN are released.

IBM FORTRAN H compiler, an optimizing FORTRAN IV for System/360, is released (70% of the compiler itself is written in FORTRAN).

**1967:** WATFOR, a load-and-go FORTRAN IV implementation, is announced by the University of Waterloo in Ontario.

**1978:** ANSI publishes revised FORTRAN standard, widely known as FORTRAN 77. It includes free format option that obviates the need for FORMAT statement.

First release of VAX FORTRAN by Digital Equipment Corp.

**1982:** Twenty-fifth anniversary of FORTRAN celebrated at National Computer Conference Pioneer Day in Houston with Backus and others in attendance.

Other galas and exhibits held at IBM Programming Center at Santa Teresa, Calif., and at SHARE meeting in New Orleans.

**1986:** IBM announces that support for FORTRAN H compiler will eventually be dropped, prompting many users to begin massive conversion to VS FORTRAN.

Cray releases CFT 77, first full FORTRAN 77 implementation for its supercomputers.

**1987:** Microsoft releases MS FORTRAN 4.0, its first full FORTRAN 77 for IBM PC-compatible micros.

On-line *Books in Print* database lists over 340 works on FORTRAN—and over 400 on Pascal.

Bemer's 1961 assessment of FORTRAN absolutely right.

"FORTRAN's primary purpose and achievement was not in being a computer language," Bemer says today from his home in Phoenix, where he runs his own software firm. "The aim was to make an efficient compiler." Compilers existed before an IBM programming researcher named John Backus proposed building one for the IBM 704 in a 1953 memo, but none could compare with what experienced programmers could produce by hand coding.

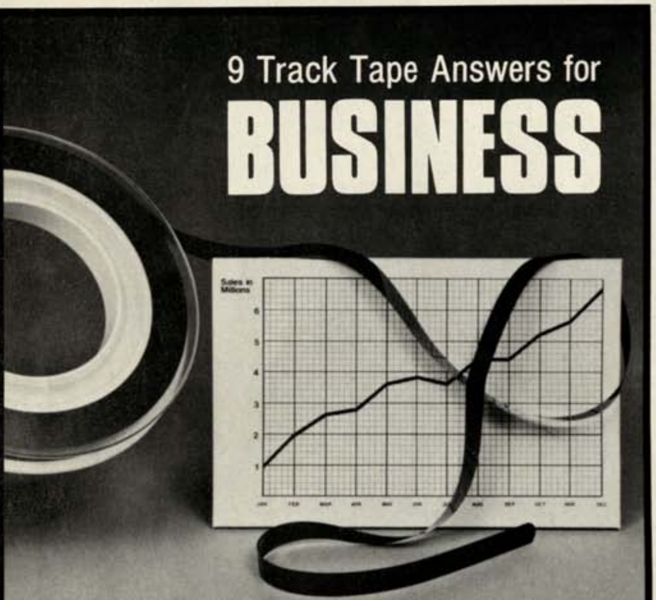
Dp consultant and DATAMATION advisor Robert Patrick recalls his reaction as a research engineer at General Motors in the late '50s, when IBM sent over an employee to describe a new software package being developed for the 704. "It was John Backus himself, and the package turned out to be FORTRAN," says Patrick. "I was lukewarm. At that time, I wasn't having any trouble getting work done in assembly language."

One reason Patrick and many other users were quickly won over by FORTRAN was because of the kind of code the compiler could produce. In fact, for some source program segments, the original FORTRAN compiler is said to have produced perfect code. Not for all segments, however—like its successors, the original FORTRAN compiler required several hundred fixes after its first release.

Yet, almost a decade later, IBM researchers proved that one of the internal compile algorithms developed by Backus's team was an optimal solution. Designers of IBM's H-level FORTRAN for the System/360, which was first released in 1966, used techniques developed in the original FORTRAN I. The H compiler is only now being displaced by VS FORTRAN as the state-of-the-art compiler of large-system IBM FORTRAN shops.

But the most obvious plus of early FORTRAN was that it saved programming time. In a paper delivered at FORTRAN's 25th anniversary celebration at the National Computer Conference in 1982, the late Herbert S. Bright described the first known commercial release of IBM's 704 FORTRAN compiler. On the very first day that it arrived at Bright's workplace, the Westinghouse-Bettis Atomic Power Laboratory in Pittsburgh, he and his colleagues were able to run a test program that had been written in a single afternoon. This was at a time when comparable programs took weeks to code in assembly language.

"FORTRAN shortened the time it took people to solve problems on a computer dramatically—in some instances, by a factor of 10," says City College of New York professor Daniel D. McCracken. McCracken's 1961 book, *A Guide to FORTRAN Programming,* probably introduced more people to the language than any other single book. The 88-page classic sold more than 300,000 copies before finally going out of print in its 25th year, 1986. McCracken sums up his book and FORTRAN's success this way: "Beginners could read my book over a weekend,



# What can you expect from the new LaserJet Series II Printer?

we never stop asking What if...

Everything.

Because the LaserJet Series II Printer from Hewlett-Packard is the product of experience. It's a second generation printer from the company with the world's largest installed base of laser printers.

Whatever your company's needs, the LaserJet Series II will deliver the performance you expect, at up to 8 pages/minute.

Take a simple memo like the Soup letter we created with Microsoft Word. As you can see, you can print in a variety of formats and type styles with our wide selection of LaserJet fonts.

Or you can create a sophisticated combination of text and graphics. With additional plug-in memory, you can also produce full-page 300 dpi graphics, like our Nuts form shown below. To do this, we used HP's new ScanJet desktop scanner, Microsoft Windows and Pagemaker® from Aldus.

With support by more than 500 of the most popular software packages, the LaserJet Series II Printer can produce whatever type of business document you need. And LaserJet Series II works with all popular PCs so it can easily be integrated into your existing system.

In fact, only the price is unexpected—starting as low as $2495.*

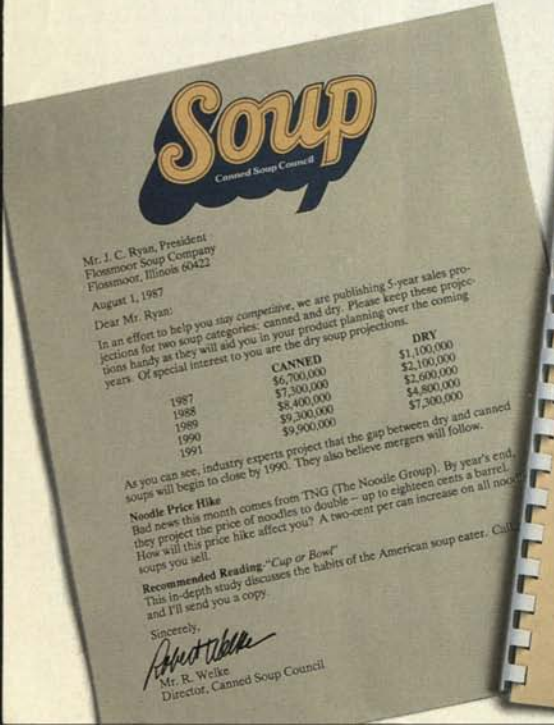
For the authorized dealer nearest you, call us at 1 800 367-4772, Ext. 275P.

HEWLETT PACKARD
*Business Computing Systems*

MicroSoft is a registered trademark of MicroSoft Corp. Pagemaker is a U.S. registered trademark of Aldus Corporation.

*Suggested U.S. list price. © 1987 Hewlett-Packard Co. PE12702DM4

CIRCLE 23 ON READER CARD


Soup
Canned Soup Council

Mr. J. C. Ryan, President
Flossmoor Soup Company
Flossmoor, Illinois 60422

August 1, 1987

Dear Mr. Ryan:

In an effort to help you stay competitive, we are publishing 5-year sales projections for two soup categories: canned and dry. Please keep these projections handy as they will aid you in your product planning over the coming years. Of special interest to you are the dry soup projections.

| | CANNED | DRY |
|---|---|---|
| 1987 | $6,700,000 | $1,100,000 |
| 1988 | $7,300,000 | $2,100,000 |
| 1989 | $8,400,000 | $2,600,000 |
| 1990 | $9,300,000 | $4,800,000 |
| 1991 | $9,900,000 | $7,300,000 |

As you can see, industry experts project that the gap between dry and canned soups will begin to close by 1990. They also believe mergers will follow.

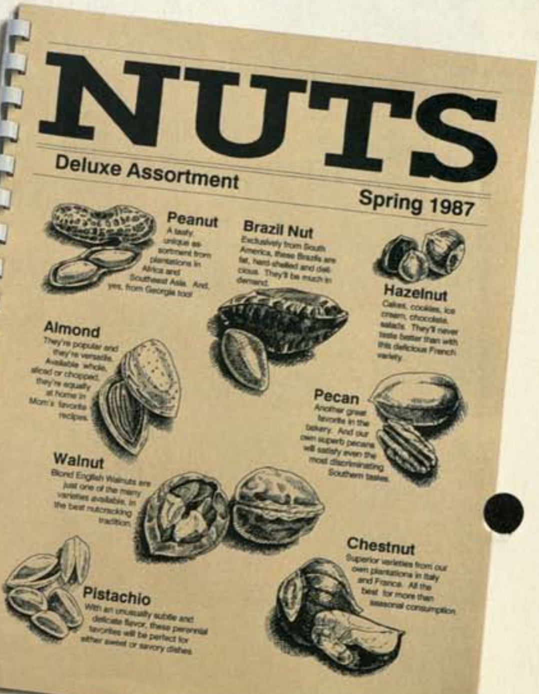
**Noodle Price Hike**
Bad news this month comes from TNG (The Noodle Group). By year's end, they project the price of noodles to double – up to eighteen cents a barrel. How will this price hike affect you? A two-cent per can increase on all noo soups you sell.

**Recommended Reading**: *"Cup or Bowl"*
This in-depth study discusses the habits of the American soup eater. Cal and I'll send you a copy.

Sincerely,

Robert Welke

Mr. R. Welke
Director, Canned Soup Council



# NUTS

**Deluxe Assortment**

**Spring 1987**

**Peanut**
A tasty, unique assortment from plantations in Africa and Southeast Asia. And, yes, from Georgia too!

**Brazil Nut**
Exclusively from South America, these Brazils are fat, hard-shelled and delicious. They'll be much in demand.

**Hazelnut**
Cakes, cookies, ice cream, chocolate, salads. They'll never taste better than with this delicious French variety.

**Almond**
They're popular and they're versatile. Available whole, sliced or chopped, they're equally at home in Mom's favorite recipes.

**Pecan**
Another great favorite in the bakery. And our own superb pecans will satisfy even the most discriminating Southern tastes.

**Walnut**
Blond English Walnuts are just one of the many varieties available, in the best nutcracking tradition.

**Chestnut**
Superior varieties from our own plantations in Italy and France. All the best for more than seasonal consumption.

**Pistachio**
With an unusually subtle and delicate flavor, these perennial favorites will be perfect for either sweet or savory dishes.

come in and try to program, and find, usually, that the computer hadn't blown up."

Ease of use remains an important FORTRAN feature today. While computer science majors and engineers usually have been exposed to some FORTRAN as students, the majority of programming courses today use Pascal, C, BASIC, and other languages (McCracken's latest book is on Modula-2). Originally designed with engineering problems in mind, FORTRAN remains easy for technical programmers to learn. "Recent graduates tend to be multilingual," notes Chevron's Rosenblatt, "and with our existing user base, bringing people up to speed in FORTRAN just isn't an issue."

A good thing, too, because today's graduates are bound to find FORTRAN on any machine they use in the technical world. The transportability of FORTRAN began early. Two months after Backus and his team of programmers delivered the first FORTRAN compiler to 704 users, another IBM programming group (led, incidentally, by Bemer) released a version for the IBM 650, an inexpensive commercial machine with a decimal-based architecture. By 1964, a DATAMATION article on "The Various FORTRANs" (August 1964, p. 25) noted the existence of 43 different FORTRAN compilers.

"We use FORTRAN because it's available on most all machines," says Al Williams, manager of computer resources and analysis at the Aerospace Division of GE/RCA, Princeton, N.J. GE/RCA Aerospace, which builds unmanned satellites, boasts a wide range of hardware from IBM, DEC, Prime, Data General, and Hewlett-Packard. "Ninety percent of our ground systems programming [e.g., design and testing of components] is in FORTRAN," Williams explains.

"FORTRAN'S STRENGTH REFLECTS A COMPUTER COMMUNITY WEAKNESS."

One person who doesn't use FORTRAN is John Backus. "I last used FORTRAN 20 years ago on something that it turned out to be unsuitable for," recalls Backus, now an IBM fellow working in the San Francisco Bay area. "I didn't like it then, and I don't like it now."

Backus's objections are not limited to FORTRAN. "Give or take 20%, it's like most other languages," he says, "and they're all lousy. ALGOL, PL/1, C—these are all a terrible way to think about programs." Commenting on FORTRAN's astonishing endurance, Backus says, "While this may be a strength of FOR-

mainframe graphics workstations.

And now with IRMALAN APA Graphics,™ PCs on your IBM® Token-Ring or other NETBIOS-compatible LAN can all be doing great work in pictures too.

All our graphics products are compatible with the latest GDDM host software product on the mainframe side and IBM PCs, XTs, ATs and AT&T 6300s on the personal computer side.

For more information on DCA's graphics solutions, call us today at 1-800-241-IRMA, ext.507

dca®

CIRCLE 26 ON READER CARD

TRAN, it really reflects one of the weaknesses of the computer community that we haven't come up with a better way."

The motivation behind Backus's current work, in an area he refers to as "functional programming," is that software should let programmers concentrate solely on the logical purpose of a program—and not worry about computing addresses, storage schemes, and the like. "Current languages force you to think at much too low a level," he contends. "What we need is a new proposition."

Insofar as they apply to FORTRAN, Backus's complaints are hardly unique or new. ALGOL adherents noted deficiencies in FORTRAN as early as the '60s. More recent critics, like Cornell University's Kenneth Wilson and Dutch computer scientist Edsger Dijkstra, have likened the constraints of FORTRAN programming to doing higher math with Roman numerals and controlling jumbo jets by whip and spur. Backus himself mounted an influential attack on conventional programming in a 1978 paper entitled, "Can Programming Be Liberated from the von Neumann Style?" published in *Communications of the ACM* (August 1978, vol. 21, no. 8).

Such criticism underlines an irony: the language that has long overshadowed so many others has really had little impact on language design and development. In this sense, ALGOL, which influenced the design of C, Pascal, Modula-2, and Ada, appears to be having the last laugh. While Ada's spread has been slow so far, the government's four-year-old mandate that Ada be used on so-called "mission critical" systems developed after 1984 is beginning to have some effect. For example, while most of the software used on NASA's shuttle project was written in FORTRAN and a customized FORTRAN-like language called GOAL, the space station project is using Ada. As for recent compiler development, Cray's CFT 77 was written in Pascal, while MS FORTRAN 4.0 was written in C.

FORTRAN's true legacy, beyond the latest versions of the language itself, is found in the off-the-shelf FORTRAN application systems that are widely used in engineering and scientific computing. Programs like NASTRAN from MacNeal-Schwendler Corp. (MSC), Los Angeles, and ANSYS from Swanson Analysis Systems, Houston, Pa.—the two leading structural analysis systems for mechanical engineering—are in a sense the logical successors to a language that was originally designed to help scientists and engineers solve problems on a computer. Structural engineering software packages are used to help build mathematical prototypes of large, complex devices. These packages are used by engineers wherever a model can be used to save time or money in testing or design. NASTRAN's heaviest users, according to Don McLean, MSC's vp of advanced projects, are in the automotive and aerospace fields—including the very same industrial companies that in the 1950s owned IBM 704s.

MSC's NASTRAN is over 500,000 lines of code, and—like many such engineering packages—over 95% in FORTRAN. "We use a subset of FORTRAN because of the variety of machines targeted," he says. The program has been customized for 21 different machines, including supers, mainframes, minis, and micros from all major manufacturers.

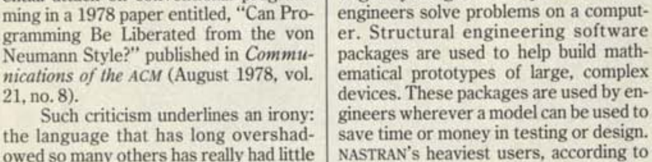
### Should Incorporate Modern Features

MSC is not about to start converting NASTRAN to another source language, but like many users McLean recognizes the appeal of other languages like C and Pascal and is anxious that FORTRAN incorporate modern features. "What FORTRAN needs," he says, "are pointer variables, new data structures, and better graphics; it'd be useful to replace a coded subroutine with a statement like BASIC's DRAW."

Keeping FORTRAN current is the work of the International Standards Organization's FORTRAN working group 5 and, in the U.S., ANSI's X3J3 subcommittee on FORTRAN. The two groups, which represent users, vendors, and computer scientists, try to coordinate their work on FORTRAN 8X in an effort to maintain a single worldwide standard. "We think FORTRAN's a good language, and we want to keep it modern," says Jeanne Adams, who chairs ANSI's FORTRAN committee.

The 8X draft adds to the standard FORTRAN language specification statements for array operations, permits programmer-defined data types (like those allowed in Pascal), and enhances procedure calls. Unlike the FORTRAN 77 standard, which removed Hollerith data types from the language spec, the current 8X draft proposes no outright deletions. Last December, a letter ballot vote recommended passing the draft on to the next higher parent committee at ANSI, but also elicited some negative comments that must be sorted out. Ultimate acceptance would be "no sooner than 1988 and possibly later," says Adams, who is well aware of how hard it is to satisfy FORTRAN's diverse and ancient constituency. "It's like changing the language you speak," she says.

Until a brave new way of speaking to computers arrives, Adams's subcommittee and their successors will have important work to do. As British computer scientist Tony Hoare remarked several years ago, "I don't know what the language of the year 2000 will look like, but I know it will be called FORTRAN." ■



"But when you said, 'The times they are a-changin',' could you perhaps be a little more specific?"

Cartoon by H.L. Schwadron

# Survey of Programming Languages and Processors

To Standards Editor
COMMUNICATIONS OF THE ACM

Under the International Standards Organization's Technical Committee 97, Subcommittee 5 has prepared a survey of common programming languages.

At the 9 October 1962 meeting, ISO/TC97/SC5 (formerly ISO/TC97/WG E) passed the following resolutions:

(1) To publish the survey in its present format (Survey-1)78, with additions and corrections received prior to 15 November 1962 to be incorporated;

(2) To continuously maintain and update the survey, with periodic publication. Format changes would require approval of WG E;

(3) WG E selects three languages for additional more detailed survey (ALGOL, COBOL and FORTRAN) without prejudice with regard to subsequent ISO standardization.

(4) The purposes of a more detailed survey of ALGOL, COBOL and FORTRAN lead to establishing as far as possible, common practices for each language, their extent and frequency of use. The specific format will be prepared by the survey committee and submitted for approval to the members of working group E.

(5) WG E establishes a permanent working group on survey, with the scope and program of work specified in document (USA-17)76 (page 4) (as modified in paragraph (d)).

On 18 October 1962, these resolutions were adopted at the Plenary Session of ISO/TC97.

In accordance with resolution (1) above, I am forwarding the survey to you for publication.

(signed) HOWARD BROMBERG
U.S. REPRESENTATIVES ISO/TC97/SC5

**The Survey** [ISO/TC97/SC5/(WGA)1,[*] 20 December 1962]

## Foreword

With a view toward international standards in the field of programing languages, a survey of common programming languages in current and imminent use was undertaken by the International Standards Organization. Technical Committee 97, Working Group E (now Subcommittee 5).

The survey project began May 1961. It is believed that the collected data, although highly perishable, are of value to the data processing community. Therefore, the survey is being distributed to member countries in form suitable for publication. ISO/TC97 has authorized publication of this work as well as continuous maintenance of the data and periodic publication. Members participating in this standardization effort are, currently, France, Germany, Italy, Netherlands, Sweden, United Kingdom and United States.

The survey data have already provided basis for the selection of three languages, ALGOL, COBOL and FORTRAN, for further work. These three languages will be considered for international standardization and a depth survey is being undertaken to establish common practice.

Please forward additions and corrections to: W. F. McClelland, Chairman, WGA, IBM Corporation, 150 Grand Street, White Plains, New York, U.S.A.

*(Please turn the page)*

[*] International Organization for Standardization: Technical Committee 97, Computers and Information Processing; Subcommittee 5, Programing Languages; Working Group A, Survey.

## SOURCES OF INFORMATION ON CAREER OPPORTUNITIES IN MATHEMATICS, PROGRAMMING AND ELECTRONIC DATA PROCESSING OCCUPATIONS

***Compilied by the Education Committee of the ACM***

*Careers in Science, Mathematics, and Engineering.* A selected bibliography. Superintendent of Documents, U. S Government Printing Office, Washington 25, D. C. 1961 DE-2-6007. 25 cents.

*Encouraging Future Scientists.* Keys to Careers. A selected bibliography. 1958–59. National Science Teachers Association. 1201 16th Street, N.W., Washington 6, D. C.

*Mathematics and Your Career.* 1960. U. S. Department of Labor, Bureau of Labor Statistics, Washington 25, D. C. 0-542720.

*Careers in Mathematics.* 1961. National Council of Teachers of Mathematics. 1201 16th Street, N.W., Washington, D. C. 25 cents.

*Professional Opportunities in Mathematics.* A Report for Undergraduate Student of Mathematics. 1961, The Mathematical Association of America, University of Buffalo, Buffalo 14, N. Y. 25 cents.

*Employment Outlook For Mathematicians, Statisticians, and Actuaries.* Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. Bulletin 1300-58. 10 cents.

*Employment Outlook For Electronic Computer Operating Personnel, Programmers.* Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. Bulletin No. 1300-34. 10 cents.

*Occupations in Electronic Data-Processing Systems.* 1959. Job Descriptions. Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. 25 cents.

*Computer Occupations.* Occupational Guide #26. 1960. Michigan Employment Security Commission, 7310 Woodward Avenue, Detroit 2, Michigan. 25 cents.

*Summer Employment in Federal Agencies.* 1962. U. S. Civil Service Commission Pamphlet 45. Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. 15 cents.

*Employment Opportunities For Women Mathematicians and Statisticians.* 1956. Women's Bureau Bulletin No. 262, Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. 25 cents.

NOTES

1. ABBREVIATIONS

| | |
|---|---|
| ALPHA | ALPHABETIC (DEVICE) |
| CHAN | CHANNEL(S) |
| CRT | CATHODE RAY TUBE OUTPUT |
| FLP | FLOATING POINT (DEVICE) |
| FLX | FLEXOWRITER |
| FXP | FIXED POINT |
| I/O | GENERAL (UNSPECIFIED) INPUT/OUTPUT DEVICE |
| N TAPES | (MAGNETIC) TAPE UNITS |
| OPT | OPTIONAL |
| PCH | PUNCH |
| PP | NUMBER OF PAGES |
| PRNTR | PRINTER |
| P-TAPE | PAPER TAPE |
| RDR | READER |
| XREG | INDEX REGISTER (B-REGUSTER( MODIFIER REGISTER) |
| / | OR |

2. NUMBERED COLUMN HEADINGS

1 TRANSLATOR PROGRAM SIZE (UNITS OF 1000 INSTRUCTIONS)
2 TRANSLATION IS FROM SOURCE LANGUAGE TO
  A A SEPARATE AND NAMED ASSEMBLY LANGUAGE
  B AN INTERMEDIATE MACHINE-INDEPENDENT LANGUAGE
  G MACHINE LANGUAGE
  I AN INTERMEDIATE LANGUAGE FOR ONLINE INTERPRETATION
3 TRANSLATION AND RUN
  A MUST BE SEPARATE
  B MUST BE IN SEQUENCE (MANDATORY LOAD-AND-GO)
  C MAY OCCUR ANT TIME (OPTIONAL LOAD-AND-GO)
  I ARE IDENTICAL (INTERPRETIVE)
4 FIELDS OF APPLICATION ARE
  A ALGEBRAIC, SCIENTIFIC
  B BUSINESS DATA PROCESSING
  C CONTROL (PROCESS, REALTIME, TOOL, ETC.)
  I INFORMATION RETRIEVAL
  L LIST HANDLING
  M SYMBOL MANIPULATION
  N NATURAL LANGUAGE TRANSLATION AND PROCESSING
  S SIMULATION

3. PUBLISHER

* MEANS PUBLISHER IS THE SAME AS THE CONSTRUCTOR

4. ASSEMBLY (MACHINE-DEPENDENT) LANGUAGES EXCLUDED FROM THE SURVEY

| | |
|---|---|
| ABC I | HONEYWELL 1000 |
| ACUTE | UNIVAC III |
| ALMOST | UNIVAC III |
| ARGUS | HONEYWELL 800 |
| AS-1 | UNIVAC 1206 |
| AUTOCODE | UNIVAC 1103 |
| ASAP | RECOMP II |
| AUTOCODE | NATIONAL'ELLIOTT 802 |
| AUTOCODER | IBM 702, 705, 7070, 1401, 1410, ETC. |
| A0,A1,A2,A3 | UNIVAC I, II |
| BLESSED | BURROUGHS 220 |
| CAGE | IBM 704 |
| CAP | UNIVAC 1103A |
| CASE SOAP | IBM 650 |
| CODAP | CDC 1604 |
| CS-1 | UNIVAC M-460 |
| CUT-AS | UNIVAC 1218 |
| DAS | BURROUGHS 204 |
| DATACODE I | BURROUGHS 205 |
| EASY | HONEYWELL 400 |
| ESCAPE | IBM 650 |
| FAP,BEFAP | IBM 704, 709, 4090 |
| FAST | IBM 650 |
| FLIP | UNIVAC 1103, 1103A |
| FLOP | IBM 701 |
| FORMOST | UNIVAC 1107 |
| GEPURS | IBM 701 |
| GP | UNIVAC I, II |
| GPX | UNIVAC II |
| JCS'13 | IBM 701 |
| K5 | UNIVAC LARC |
| LAS | UNIVAC LARC |
| MAC | MERCURY |
| MISHAP | UNIVAC 1103, 1103A |
| NEAT | NCR 304 |
| NYAP | IBM 704 |

4. ASSEMBLY (MACHINE-DEPENDENT) LANGUAGES EXCLUDED FROM THE SURVEY

| | |
|---|---|
| (S)PAR | BENDIX G'20 |
| POGO | BENDIX G'15 |
| RAWOOP'SNAP | UNIVAC 1103, 1103A |
| RELCODE | UNIVAC I, II |
| ROAR | RPC'4000 |
| RIII | UNIVAC M-460 |
| SAC | BURROUGHS 201, 205 |
| SAIL | UNIVAC LARC |
| SAL | UNIVAC LARC |
| SALT | UNIVAC III |
| SAP | IBM 704 |
| SCAT | IBM 709, 7090 |
| SCOPAC | RECOMP II |
| SCRAP | RECOMP II |
| SLAP | UNIVAC 1103 |
| SLEUTH | UNIVAC 1107 |
| SPURT | UNIVAC 490 |
| SNAP | BENDIX G'20 |
| SNAP | RECOMP II |
| SOAP I, II | IBM 650 |
| SO2 | IBM 701 |
| SPACE | IBM 650 |
| SPAR | BURROUGHS 205 |
| (S)PAR | BENDIX G'20 |
| STAR | BURROUGHS 205, 220 |
| STRAP I, II | IBM 7030'709 |
| STRIP | ALWAC III'E |
| TAC | .PHILCO 2000 |
| TASS | IBM 650 |
| TRANSUSE | UNIVAC 1103A |
| TRIM | UNIVAC 1218 |
| UNISAP | IBM 650, UNIVAC I |
| USE | UNIVAC 1103 |
| UTMOST | UNIVAC III |
| XIL | UNIVAC I |
| X1, X2 | UNIVAC I, II |
| X6 | UNIVAC 5580/90 |
| Z | UNIVAC 5580 |

5. LANGUAGES EXCLUDED FROM SURVEY DUE TO OBSOLESCENCE

| | |
|---|---|
| ACOM | IBM 701, 705 |
| ACT | UNIVAC I |
| ANCP | BURROUGHS 201, 205 |
| BACAIC | IBM 650, 701 |
| BALITAC | IBM 650 |
| BELL L2,L3 | IBM 650 |
| BIOR | UNIVAC I, II |
| BLIS | IBM 650 |
| BOEING COM | UNIVAC 1103A |
| B 15 | IBM 650 |
| CHIP | UNIVAC 1103 |
| COMPILER I | UNIVAC 1103A |
| COMM. TRANS. | IBM 7070/80/90 |
| COMPREHENSIVE | WHIRLWIND |
| DOW | BURROUGHS 201, 204, 205 |
| DRUCO I | IBM 650 |
| DUAL | IBM 701 |
| EASE II | IBM 650 |
| EASIAC | MIDAC |
| EASY FOX | JOHNNIAC |
| ELI | IBM 650, 705 |
| ERFPI | LGP'30 |
| FACS | IBM 650 |
| FAIR | IBM 705 |
| FLAIR | IBM 650, 704 |
| FLINT | IAS MACHINE |
| FLIP'SPUR | UNIVAC 1103 |
| FLIPPER | BENDIX G'15A |
| FLOP | ;BM 701 |
| GIF | IBM 650 |
| INTERCOM 101 | BENDIX G'15 |
| KOMPILER 2 | IBM 701 |
| LT'2 | IBM 701 |
| MAGIC | MIDAC |
| MATH-MATIC | UNIVAC I,II |
| MITILAC | IBM 650 |
| MJS | UNIVAC I, II |
| MORTRAN | JOHNNIAC |
| OMNICODE | IBM 650, 702 |
| OMNIFAX | UNIVAC I, II |
| PACT I | IBM 701 |
| PENNCODE | PENNSTAC |
| QUEASY | IBM 701 |
| QUICK | IBM 701 |
| SCRIPT | IBM 702 |
| SEESAW | IBM 701 |
| SHACO | IBM 701 |
| SHORTCODE | BINAC, UNIVAC I, II |
| SNAP | UNIVAC 1103A |
| SOHIO | IBM 705 |
| SPEEDCODE | IBM 650, 704 |
| SPEEDCODING | IBM 701 |
| SPEEDEX | IBM 701 |
| SPUR | IBM 650 |
| SWAP | IBM 650 |
| TRANSCODE | FERUT |

| LANGUAGE | MANUAL | | | | TRANSLATOR | | | | | | | | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDENTIFICATION | DATE | PP | PUBLISHER | CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFIGURATION | | |
| COBOL NARRATOR | 95-05-000 | DEC 60 | 161 | * | RCA | RCA 501 | 65 | SEP 60 | G | A | B | 16K CHAR, 6 TAPES, RDR, OFFLINE PRNTR | BROMBERG, H | BASED ON COBOL 60, PRELIM. MANUAL MAY 60, SEE ALSO 95-05-002(96PP), 95-05-003(32PP) |
| COBOL NARRATOR | 93-05-002 | FEB 62 | 208 | * | RCA | RCA 301 | 45 | OCT 62 | G | A | B | 20K CHAR, 6 TAPES, RDR, ONLINE PRNTR | BROMBERG, H | ALL OF REQ. COBOL 61 + SOME ELECTIVE. |
| COBOL NARRATOR | | APR 62 | 250 | ENGLISH ELECT | R.C.A. | ENGLISH ELECT. KDP10 | | OCT 61 | G | A | B | 66K, 6 TAPES, P TAPE, PRNTR | DUNCAN, FG | BASED ON COBOL 60 |
| COBOL | | | | | SPERRY-RAND | UNIVAC II | 480 | OCT 60 | G | A | B | 2K, 12 TAPES | | |
| COBOL 60 | | | | | SPERRY-RAND | USS 80 | 15 | APR 61 | G | A | B | 2 TAPES, DRUM, RDR, PCH, PRNTR | | |
| COBOL | 3166/1 | OCT 60 | 143 | * | I.C.T. | ICT 1301 | | | | | B | MARK I, 2K, 12K DRUM | ELLIS, PV | BASED ON COBOL 60 (RAPIDWRITE VERSION IN MANUAL P155, MAY 61) |
| | | | | | I.C.T. | ICT 1301 | | | | | B | MARK II, 2K, 4 TAPES | | |
| COBOL | | | | PHILCO | COMP. SCIENCES | PHILCO 2000 | | | | | B | 8K, 6 TAPES | GUERNACCINI, J | BASED ON COBOL 61 |
| COBOL | 5000-21002-P | SEP 61 | 45 | * | BURROUGHS | B-5000 | | | | | B | | SPEIERMAN, KH | BASED ON COBOL 61, DUE DEC 62 |
| COBOL | | | | | ARMOUR RESEARCH | UNIVAC 1103A | 3.5 | | A | A | B | 8K, 16K DRUM, 8 TAPES | MITMAN, B | BASED ON COBOL 61, PLUS ELECTIVES, DUE DEC 62 |
| COBOL 61 | F-7411 | JUN 61 | 122 | NCR | NCR/GE | NCR(304)GE | 50 | DEC 61 | A | A | AB | 4.8 K, 6 TAPES, RDR, PRNTR | KEATING, WP | |
| COBOL 61 | | | | | NCR | NCR 315 | 150 | AUG 62 | A | A | AB | 10K, 5 TAPES, RDR, PRNTR | KEATING, WP | |
| COBOL 61 | U3389 | APR 62 | 266 | SPERRY-R | SPERRY-RAND+CSC | UNIVAC III | | | | | | 16K, 8 TAPES | HOPPER, GM | DUE DEC 62 |
| COBOL 61 | | | | | SPERRY-RAND | UNIVAC 490 | | SEP 62 | A | | B | 16K, 800K DRUM, 6 TAPES, PRNTR | DESSILETS, PH | |
| COBOL 61 | | | | | SPERRY'RAND+CSC | UNIVAC 1107 | | | | | B | 16K, 800K DRUM, 6 TAPES, PRNTR PCH, RDR, P-TAPE/PCH/RDR | HOPPER, GM | DUE 4TH QTR 62 |
| COBOL 61 | AFLCM-500-11 | NOV 61 | | * | HQ. AIR FORCE LOGIS. COMMAND | UNIVAC 1105 | 80 | SEP 61 | B | A | B | 12K, 32K DRUM, 20 TAPES | JONES, J | COMPILES COBOL VERBS AT 4 SECONDS RATL. CORRECTOR-CLEANUP PHASE PRIOR TO RUN |
| COBOL 61 | | | | MINN-HONEY | HONEYWELL+CSC | MH 400 | | | | | B | 2K, 4 TAPES, RDR, PRNTR | COLEN, P | DUE SEP 62 |
| COBOL 61 | | | | MINN-HONEY | HONEYWELL+CSC | MH 800 | | | | | B | 4K, 4 TAPES, RDR, PRNTR | COLEN, P | COBOL 61 OR 62+SORT+REPORT GENERATOR, DUE FEB 63 |
| COBOL 61 | F28-8053-1 | JUN 61 | 112 | * | IBM | IBM 705 II | 50 | FEB 62 | A | A | B | 40K CHAR, 8 TAPES | | |
| COBOL 61 | 220-8045-1 | 61 | | CODASYL | IBM | IBM 705 III | 50 | FEB 62 | A | A | B | 40K CHAR, 8 TAPES | | |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COBOL 61 | J28-6177,91 | | | * | IBM | IBM 7080 | 50 | FEB 62 | A | A | B | 40K CHAR, 8 TAPES | | |
| COBOL | J28-6260, 6283, 6194,5,6, 6024-1 | | | * | IBM | IBM 7090/7094 | 55 | DEC 62 | A | C | B | PRNTR, CARD RDR, 3 TAPES PLUS 5 UNITS- TAPES OR DISK | | COMPILER AS COMPONENT OF IBJOB PROCESSOR |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COBOL 61 | J28-6177,91 |  |  |  | • | IBM | IBM 7080 | 50 | FEB 62 | A | A | B | 40K CHAR, 8 TAPES |  |  |
| COBOL | J28-6260, 6283, 6194,5,6, 6024-1 |  |  |  | • | IBM | IBM 7090/7094 | 55 | DEC 62 | A | C | B | PRNTR, CARD RDR, 3 TAPES PLUS 5 UNITS- TAPES OR DISK |  | COMPILER AS COMPONENT OF IBJOB PROCESSOR |
| COBOL 61 |  |  |  |  |  | IBM | IBM 7070,74 | 38 | FEB 62 | A | A | B | 10K, 7 TAPES |  |  |
| COBOL | J28-6213, 6203, 6243, 6236, 6210 |  |  |  | • | IBM | IBM 7040/7044 |  |  | A | C | B | 16K, CARD RDR-PCH, PRNTR, 5 TAPES |  | COMPILER AS COMPONENT OF IBJOB PROCESSOR |
| COBOL 61 | J28-6182-0(PROG) | MAR 62 |  |  | • | IBM | IBM 1401 |  |  |  |  | B | 12-16K, 4 TAPES |  |  |
| COBOL 61 | J28-6189(OPER) | MAR 62 |  |  | • | IBM | IBM 1401 |  |  |  |  | B | 4-8K, 6 TAPES |  |  |
| COBOL 61 | +J28-0232 | NOV 61 |  |  | • | IBM | IBM 1410 | 13 | JAN 62 | A | A | B | 20K, 4 TAPES, RDR/PCH, PRNTR | MORAN, H | MANY ELECTIVES. OBJECT PROGRAM USES IOCS |
| COBOL 61 |  | JAN 62 | 200 |  | • | SYLVANIA ELECT. | MOBIDIC | 50 |  |  |  | B | 8K, 5 TAPES, 1 I/O | SAMMET, J | DUE JUL 62. ALL OF REQ. COBOL+ SOME ELECTIVE |
| COBOL 61 |  |  |  |  | • | SYLVANIA ELECT. | SYLVANIA 9400 | 50 |  |  |  | B | 16K, 5 TAPES, 1 I/O | SAMMET, J | DUE JAN 62. ALL OF REQ. COBOL+ SOME ELECTIVE |
| COBOL 61 |  |  |  |  | • | BENDIX | G-20 |  |  |  |  | B | 8K, 4 TAPES, RDR, PCH, PRNTR | BAUER, F | DUE 62. ALL OF REQ. COBOL + SOME ELECTIVE |
| COBOL 61 |  |  |  |  |  |  | CDC 1604 |  |  |  |  | B |  |  | DUE 62. |
| COBOL 61 |  |  |  |  |  |  | CDC 924 |  |  |  |  | B |  |  |  |
| COBOL 61 |  |  |  |  |  | FERRANTI | ATLAS |  |  |  |  | B |  |  |  |
| COBOL GAMMA 30 | DCF-2-8 | 62 | 225 | BULL |  | RCA | GAMMA 30 | 50 | JUL 62 | G | A | B | 20K CHAR,6 TAPES, RDR, PRNTR | BULL | FRENCH VERSION OF COBOL 301 COMPRISING ALL REQUIRED COBOL 61 LANGUAGE |
| RAPIDWRITE |  |  |  |  | • | ICT LTD | ICT 1500 II, IIA |  | 62 | G | A | B | TAPE | ELLIS, PV | VERSION OF COBOL 61 |
| COBOL ICT |  |  |  |  | • | ICT LTD. | ICT 1301 |  | 62 | G | A | B | ANY | ELLIS, PV |  |
| KDF9 COBOL |  |  |  |  |  |  | ENGLISH ELECT. KDF9 |  |  | G | C | B | 8K, TAPES, PRINTER | DUNCAN, FG | COBOL 61 AS FAR AS POSSIBLE. DUE DEC 63 |
| COBOL 61 |  |  |  |  |  | OLIVETTI | OLIVETTI 9003 |  |  |  |  |  |  | PICCIAFUCCO, U |  |
| COBOL 61 |  |  |  |  |  | AEI | AEI 1010 |  |  |  |  |  |  | EVANS, KC |  |
| COBOL |  |  |  |  |  | INST. MATH.APP. | BULL GAMMA 60 |  |  | G | A | B |  | INST. DE MATH. GRENOBLE |  |
| AIMACO | AMCH 171-2 | JUN 59 | 87 |  | • | AIR MATERIEL CO | UNIVAC 1105 | 30 | JAN 60 | A |  | B | 8K, 5 TAPES + 10-TAPE UNIVAC I | JONES, J | UNIVAC I TRANSLATES TO USE(5K INSTR.), MODIFIED B-0 |
| CODEL |  | AUG 59 | 34 |  | • | CDL | ICT 1301 |  |  |  |  | B |  | WENSLEY, J | WITHDRAWN IN FAVOR OF COBOL |
| GECOM | CPB-123 | OCT 60 | 50 |  | • | GE | GE 225 | 80 | SEP 61 |  |  | AB | 8K, 6 TAPES, RDR, TAPE/PRNTR | KATZ, C | ACCEPTS TABSOL, COBOL 61, SOME ALGOL FEATURES AND FRINGE(9PAC-LIKE REPORT/FILE MAINT/SORT GENERATOR-DUE LATE 61), 12 RUNS OR LESS. 1600 ML INSTR/MINUTE |
| TABSOL | CPB-147 | MAR 61 | 30 |  | • | GE | GE 225 |  |  |  |  | B | 8K, 6 TAPES, RDR, TAPE/PRNTR | KATZ, C | LOGIC TABLE LANGUAGE. DUE NOV 61 |
| FACT | 160-2M DSI-2TE | JAN 61 | 175 | CSC/MINN-H |  | COMP. SCIENCES/ MINN'HONEYWELL | MH 800 | 230 | DEC 61 | A | A | B | 4K, 4 TAPES, RDR, PRNTR | CLIPPINGER, RF | 8-PASS. TRANSLATE 40 STATEMENTS/MIN. ALSO SORT/REPORT/FILE MAINT GENERATORS |
| FLOWMATIC | U1518 REV.1 | 58 | 103 |  | • | SPERRY-RAND | UNIVAC I |  | 56 |  |  | B | 1K, 10 TAPES | HOPPER, GM | (B-0) |
| FLOWMATIC |  |  |  |  |  | SPERRY-RAND | UNIVAC II |  | 56 |  |  | B | 2K, 16 TAPES | HOPPER, GM | (B-0) |
| ADAPT |  | MAY 61 | 34 |  |  | COMP. SCIENCES | IBM 1401 | 2 | OCT 61 | A | C | B | 4K, 2 TAPES, HI-LO. | BYHAM, C | 9-PASS FOR 4K, LESS FOR 12K. COBOL-LIKE + SEQ. VERB |
| FARGO | C24-1464 | DEC 61 | 48 |  | • | IBM | IBM 1401 |  |  | G | B | AB | 4K |  | REPORT GENERATOR. ALSO J24-1467-0 FOR RAMAC SYSTEMS |
| SURGE |  |  |  |  |  |  | IBM 704 |  |  |  |  | B |  |  |  |
| 9PAC | INTRO. TO 9 PAC<br>J28-6166,7,8 | JUN 61<br>MAY 60 | 78<br>200 | SHELL OIL<br>IBM |  | SHARE/IBM | IBM 709/7090 |  |  |  |  | B |  | FARNWORTH, G | REPORT, SORT + FILE MAINTENANCE GENERATOR |
| VIPP |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| APT III |  |  |  |  |  | SPERRY-RAND | UNIVAC 1107 |  |  |  |  |  |  | CHINGARI, G |  |
| APT |  |  |  |  | • | M.I.T. | IBM 704/709/9C | 40 | FEB 59 |  |  | C | 32K, 8 TAPES | ROSS, DT | WILL RUN IN PARALLEL WITH AUTOPROMT. AIA(SAN DIEGO) WRITING APT II IN FORTRAN. PRODUCES ML. |
| APT |  | 61 | 18 | PERGAMON PRESS |  | JOHN HOPKINS APL | 1103 A | 4 | MAR 58 | G | B | A | 8K CORE, 16K DRUM, 1 TAPE, CARD RDR | RICH, R.P. | ONE PASS COMPILER, USES POLISH PREFIX NOTATION |
| AUTOPROMT | M+A 12<br>E20-6104 | JUN 61 | 226<br>34 |  | • | IBM | IBM 704/709/9C | 32 | DEC 60 |  |  | C | 8K, 10 TAPES, NO DRUM | MATSA, SM | CONSTRUCTS MACHINE TOOL CONTROL PROGRAMS. SHARE PROGRAM IB-4PRM. |
| ROHR NUMERICAL TOOL CONTROL | UT 2495 | SEP 61 | 153 | SPERRY-RAND |  | ROHR + SPERRY-RAND | USSI 90 | 6 | SEP 61 |  |  | C | 5K, DRUM, SIR, RDR, PCH, PRNTR | HERTEL, P.H. | CONSTRUCTS MACHINE TOOL PROGRAMS |
| NUCOM |  | NOV 61 | 20 |  | • | AUTONETICS | RECOMP III |  |  |  |  | C | FXP | HALPRIN, L | DUE NOV 61 |
| DYNAMO |  | NOV 61 | 64 |  | • | M.I.T. | IBM 704/709/90 | 10 | APR 59 | G | B | S | 32K, 3 TAPES | PUGH, AL III | MODELS BUSINESS AND ECONOMIC SITUATIONS |
| IPL-V | P-1918 | MAR 60 | 94 | RAND CORP. |  | RAND CORP./ CARNEGIE TECH | IBM 650 RAMAC |  | JUN 60 |  | I | LM | 2K DRUM, DISK FILE | NEWELL, A | COMBINED MANUAL PUBLISHED BY PRENTICE-HALL, NOV 61 |
| IPL-V | P-1897 | MAY 60 | 138 | RAND CORP. |  | RAND CORP./ CARNEGIE TECH | IBM 704/7090 | 8 | AUG 60 |  | I | LM | 32K, 2 TAPES | NEWELL, A |  |
| IPL-V | P-1929 |  | 25 | RAND CORP. |  | LINCOLN LABS | IBM 709 |  |  |  | I | LM |  | GREEN |  |
| IPL-V |  |  |  |  |  | RAND CORP. | IBM 7090 |  |  |  |  | LM |  | TONGE |  |
| IPL-V |  |  |  |  |  | SDC | PHILCO 2000 |  |  |  |  | LM |  | FELDMAN |  |
| IPL-V |  |  |  |  |  | U. LONDON | MERCURY |  |  |  | I | LM |  | BUCKINGHAM |  |
| IPL-V |  |  |  |  |  | U. TEXAS | CDC 1604 |  |  |  | I | LM |  | LINDSAY |  |
| IPL-V |  |  |  |  |  | U. NO. CAROLINA | UNIVAC 1103 | 8 | MAR 62 |  | I | LM | 8K, 32K DRUM, 2 TAPES | CAVINESS | PILOT VERSION RUNNING. COMPLETE JAN 63 |
| IPL-V |  |  |  |  |  | CARNEGIE TECH | BENDIX G-20 |  |  |  | I | LM |  | NEWELL, A |  |
| IPL V |  |  |  |  |  | KISS, GR | FERRANTI PEGASUS 1 | 2 | MAR 62 | G | I | LMS | 4K, DRUM, 1 TAPE | KISS, GR | FULL FLEXIBILITY IS NOT AVAILABLE. INPUT, OUTPUT AND DEBUGGING FACILITIES MUCH SIMPLIFIED |
| LISP |  |  |  |  |  | BROWN, N<br>PEARSON, DW<br>STRACHEY, C | EMIDEC 2400 | 1 | OCT 61 | G<br>I | C | LM | 16K, 3 TAPES, P-TAPE RDR, FLX, PRNTR | BROWN, N | LISP TO COMMAND SEQUENCE FOR INTERPRETATION OR COMPILATION. |
| LISP 1 |  | MAR 60 | 156 |  | • | M.I.T. | IBM 704/709/90 | 11 | JAN 60 |  |  | LM | 32K, 4 TAPES, CARD RDR | MCCARTHY, J | LIST PROCESSING, HEURISTIC PROGRAMMING |
| LISP 1.5 |  | MAY 61 |  |  | • | M.I.T. | IBM 704/709/90 | 13 | MAY 61 |  |  | LM | 32K, 4 TAPES, CARD RDR | MCCARTHY, J |  |
| SHADOW |  |  |  |  | • | M.I.T. | IBM 704/709/90 | 5 | JUL 59 | B |  | M | 32K | BARNETT, MP | OUTPUT TO FORTRAN II. SCANNING SYSTEM. INPUT IS LANGUAGE STRUCTURE DEFINITION. |
| COMIT | INTRO TO PROG.<br>REFERENCE MANUAL | NOV 61 | 60<br>61 |  | •<br>• | M.I.T. | IBM 709/7090 | 16 | JUL 61 | I | C | ILM<br>NS | 32K, 5 TAPES, I/O | YNGVE, V | PROGRAM AVAILABLE FROM SHARE. PRINTED MANUAL AND COURSE ALSO AVAILABLE. 2 PASS. |
|  |  |  |  |  |  | U. MANCHESTER |  |  |  |  |  | LMN |  | BROOKER, RA | PHRASE STRUCTURE COMPILER. |
| MIMIC |  |  |  |  | • | RAND CORP. | IBM 704 |  |  | I |  | N |  | KELLY, H | FOR LINGUISTIC AND GRAMMAR STRUCTURE WORK |
| GPS |  |  |  |  | • | RAND CORP. |  | 8 |  | I |  | L |  | MEALY, G | GENERAL PROBLEM SOLVER. WRITTEN IN (AND OUTPUTS) IPL-V |
| GENIE |  | DEC 61 |  |  | • | RICE UNIV. | RICE COMPUTER | 7 | AUG 61 | G |  | AL | 8K | JODEIT, JG |  |
| DYANA |  |  |  |  | • | GENERAL MOTORS |  |  |  | B |  |  | 8K, 4 TAPES, OPT CRT | OLSZTYN, JT | FOR DYNAMICS PROBLEMS. OUTPUT IS A FORTRAN PROGRAM |
| MAC |  | JUN 58 |  |  | • | M.I.T. | IBM 650 | 1 | JUN 58 | G | C | A | DISK, FLP, ALPHA, XREG | WERNER, CP | FOR VECTORS,MATRICES,ORDINARY DIFFERENTIAL EQUATIONS. COMPILE AND RUN ON 650. |
| MAC |  | JUN 58 |  |  | • | M.I.T. | IBM 704/709/90 |  |  | G | A | A | 32K, 4 TAPES | WERNER, CP | FOR VECTORS,MATRICES,ORDINARY DIFFERENTIAL EQUATIONS. RUN ONLY, 650 COMPILES. |
| MAC-H |  |  |  |  | • | M.I.T. | MH 800 | 12 |  | G | C | A | 16K, 6 TAPES | WERNER, CP | FOR VECTORS,MATRICES,ORDINARY DIFFERENTIAL EQUATIONS. DUE MAR 62. |
| CL-1 | TO'B 61'3 | JAN 61 | 521 |  | • | TECH.OPERATIONS | IBM 704/709/90 |  |  |  |  |  | 32K, 8 TAPES, 2 I/O CHAN | CHEATHAM, T | ENVIRONMENT AND OPERATING SYSTEM. |

| LANGUAGE | MANUAL | | | | TRANSLATOR | | | | | | | | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDENTIFICATION | DATE | PP | PUBLISHER | CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFIGURATION | | |
| FORTRAN I | C28-6106(PROG) | MAR 61 | 104 | • | IBM | IBM 704 | 30 | JUN 57 | | | A | 4K, 8K DRUM, 4-7 TAPES, I/O | RIDGWAY, R | ALSO F28-6019(72PP,OCT 59)-PRIMER |
| FORTRAN I | C28-6097(OPER) | AUG 60 | 64 | • | IBM | IBM 704 | 30 | OCT 57 | | | A | 32K, 4-7 TAPES, I/O | RIDGWAY, R | ALSO ORIGINAL 32-7026,C28-6003(56PP,OCT56) |
| FORTRAN I | J28-6122 | JUL 61 | 58 | IBM | IBM/GUIDE | IBM 705 I,II | 15 | FEB 60 | A | A | A | 40K CHAR, 8 TAPES, I/O | RIDGWAY, R | |
| FORTRAN I | C28-6170 | FEB 62 | | • | IBM | IBM 7070 | | JUN 60 | | | A | 5K, I/O, NO TAPES | RIDGWAY, R | BASIC FORTRAN |
| | J28-6171(OPER) | | | • | | | | | | | | | | |
| FORTRAN I | C29-4047 | JUL 60 | 64 | • | IBM | IBM 650 | | JUN 59 | A | | A | 2K DRUM, XREG, ALPHA, RDR, PCH | | 2 PASS(FI TO SOAP TO ML) |
| FORTRAN I | J28-1455 | AUG 61 | 32 | • | IBM | IBM 1401 | 10 | DEC 61 | G | C | A | 8K, CARD SYSTEM | MACKLIN, D | AUGMENTED FI. TRANSLATOR ON TAPE, SOURCE PROGRAM IN CORE. UP TO 20 DIGITS OF FUNCTION ACCURACY |
| FORTRAN I | | | | | IBM | IBM 1401 | 10 | DEC 61 | G | C | A | 8K, 1 TAPE | MACKLIN, D | |
| FORTRAN I | J26-5598-0 | SEP 61 | 12 | • | IBM | IBM 1620 | 2 | JAN 61 | | | A | TAPE SYSTEM | JEANS, H | 1-PASS. ALSO J26-5597(16 PP) FOR PRE-COMPILE CHECKING PROGRAM |
| FORTRAN I | | | | | IBM | IBM 1620 | 2 | APR 61 | | | A | CARD I/O | JEANS, H | |
| FORTRAN I | | | | | IBM | IBM 7080 | | | | | A | | | |
| FORTRAN I | | | | | MEUDON OBSERVATORY | IBM 650 | 1.0 | YES | I | C | A | STANDARD | ARSAC, M | FORTRAN I LESS COMPUTED GO TO AND FORMATS. MATH ROUTINES INCLUDED IN 800 INSTR. INTERPRETING PROGRAM |
| FORTRANSIT | 32-7842-1 I | 57 | 67 | • | IBM | IBM 650 | 2 | AUG 57 | A | | A | 2K DRUM, RDR, PCH- | | IT AND SOAP II AS INTERMEDIATE LANGUAGES. |
| FORTRANSIT | C28-4028 II | OCT 59 | 80 | • | IBM | IBM 650 | 2 | JUN 58 | A | | A | 2K DRUM, RDR, PCH, XREG, FLP | | |
| FORTRANSIT | III | | | | IBM | IBM 650 | | | A | | A | TAPE SYSTEM | | |
| GOTRAN | C26-5594 | JAN 62 | 34 | • | IBM | IBM 1620 | 1 | APR 61 | | I | A | CARD I/O SYSTEM | JEANS, H | |
| GOTRAN | | | | | IBM | IBM 1620 | 1 | SEP 60 | | I | A | P-TAPE I/O SYSTEM | JEANS, H | |
| FORTRAN I | | | | | SPERRY-RAND | USS 80 | 6 | JAN 61 | G | A | A | 8K, CARD SYSTEM | DOBBS, CW | WILL RUN FORTRANSIT |
| FORTOCOM | 205-21001-P | MAR 61 | | • | BURROUGHS | BURROUGHS 205 | | SEP 60 | | | A | CARDATRON, FLP | SPEIERMAN, KH | 1-PASS, VERSION OF FORTRAN I |
| FORTRAN II | C28-6000 | OCT 59 | 72 | • | IBM | IBM 704 | 30 | MAY 58 | | | A | 4K, 8K DRUM, 4-7 TAPES, I/O | | |
| FORTRAN II | C28-6097(OPER) | AUG 60 | 64 | • | IBM | IBM 704 | 30 | MAY 58 | | | A | 8K, 8K DRUM, 4-7 TAPES, I/O | | |
| FORTRAN II | C28'6054'2(PROG) | JAN 61 | 112 | • | IBM | IBM 709/7090 | 30 | JAN 59 | | | A | 8K,5-8 TAPES, I/O | | |
| FORTRAN II | C28'6066'3(OPER) | JUN 61 | 72 | • | IBM | IBM 709/7090 | 30 | SEP 59 | | | A | 8K,5-8 TAPES, I/O | RIDGWAY, R | MONITOR VERSION |
| FORTRAN II | J28'6114'1(D-C) | APR 61 | 8 | • | IBM | IBM 709/7090 | 30 | MAY 60 | | | A | 32K, 5-8 TAPES, I/O | RIDGWAY, R | + DOUBLE PRECISION + COMPLEX. ALSO J28-6133,6135. |
| | C28-6100-1(IOCS) | FEB 61 | 96 | • | | | | | | | | | | |
| FORTRAN II | C28-6170 | FEB 62 | | • | IBM | IBM 7070/7074 | 20 | FEB 61 | A | | A | 5K, 6-9 TAPES, I/O | RIDGWAY, R | IN COMBINATION WITH AUTOCODER, REPORT GENERATOR |
| FORTRAN II | J24-1468 | FEB 62 | 20 | • | IBM | IBM 1410 | 16 | APR 62 | A | A | A | 20K, 4 TAPES, RDR/PCH, PRNTR | MORAN, MM | 5TH TAPE REQ. FOR UNINTERRUPTED TRANSLATION |
| FORTRAN II | J26'5602'0 | MAR 62 | 20 | • | IBM | IBM 1620 | | | | | A | 40K CHAR, AUTODIV, INDIR ADDRESS, RDR, PCH | JEANS, H | 2-PASS. DUE JUL 62 |
| FORTRAN II | | | | • | IBM | IBM STRETCH | 60 | | A | | A | 32K, DISK, 3TAPES | CAMPBELL, SG | AUGMENTED FII. NO. OF INSTR. INCLUDE 16K STRAP II, 20K SMAC, TABLES. DUE JUN 62 |
| FORTRAN II | | | | | AERE, HARWELL | IBM STRETCH | | | | | A | | CURTIS, AR | WRITTEN IN FORTRAN + SYMBOL MANIPULATION FACILITIES. |
| FORTRAN II | | | | | IBM(OAKLAND) | IBM STRETCH | | | | | A | | MANSFIELD, WV | AUGMENTED FORTRAN II. FAST 1-PASS FOR LIVERMORE AEC |
| FORTRAN II | J28-6132 | JAN 62 | 20 | • | IBM | IBM 7090 | | | | | A | | | AUGMENTED FORTRAN II |
| FORTRAN II | | | | | COMP. SCIENCES | LARC | 22 | AUG 61 | A | C | A | 20K, 4 LARC DRUMS. | NUTT, R | AUGMENTED FORTRAN II. CALLED LARC SCIENTIFIC COMPILER |
| FORTRAN II | | | | | | ATLAS | | | | | A | | BROOKER, RA | 3 MAN-MONTHS BY PHRASE-STRUCT. COMPILER. NO I/O. |
| FORTRAN II | | | | | COMP. SCIENCES | RCA 601 | | | | | A | | | |
| FORTRAN II | | | | | BENDIX COMPUTER | BENDIX G-20 | | | | | A | | | DUE SPRING 62 |
| FORTRAN II | 087A | AUG 61 | 113 | • | CONTROL DATA | CDC 1604 | | 115F | A | C | A | | | SEPARATE COMPILERS TO MACHINE OR ASSEMBLY LANGUAGES |
| FORTRAN II | | | | | COMPUTER USAGE | ASI 420 | | | A | A | A | 4K, 4 TAPES | OPLER, A | DUE LATE 62. VIRTUALLY IDENTICAL TO 709/90 FORTRAN. |
| FORTRAN II | | | | | MINN-HONEYWELL | MH 400 | | | | | A | | COLEN, P | DUE END OF 62 |
| FORTRAN II | | | | | SPERRY-RAND | USSI 80 | 5 | | G | C | A | 5K, DRUM, CARD | PASTER, A. | ONE PASS TO RELOCATABLE MACHINE CODE |
| FORTRAN II | | | | | SPERRY-RAND | USSI 90 | 5 | | G | C | A | 5K, DRUM, CARD | PASTER, A. | ONE PASS TO RELOCATABLE MACHINE CODE |
| FORTRAN II | | | | | SPERRY-RAND | USSII 80 | 3.8 | | G | C | A | 5K, DRUM, CARD | PASTER, A. | ONE PASS TO RELOCATABLE MACHINE CODE |
| FORTRAN II | | | | | SPERRY-RAND | USSII 90 | 3.8 | | G | C | A | 5K, DRUM, CARD | PASTER, A. | ONE PASS TO RELOCATABLE MACHINE CODE |
| FORTRAN II | | | | | GENERAL KINETICS, INC. | UNIVAC 1107 | | | | | | 32K, DRUM, 5 TAPES | MCCARTY, C.L. | DUE DECEMBER, 1962 |
| FORTRAN II | | | | | NCR | NCR 315 | | | B | A | A | 10K, 5 TAPES, RDR, PRNTR | KEATING, WP | |
| FORTRAN II | | JAN 63 | 60 | • | RCA | RCA 301 | | | G | C | A | 20K, 4 TAPES, RDR/PCH, PRNTR | HUX, R | AUGMENTED FORTRAN II. COMPILES ON 304 OR 305, 354 OR 355. OBJECT PROGRAM RUNS ON 354 OR 355 |
| S1 | | | | | | IBM 7030 | 7 | APR 62 | A | A | A | | GLENNIE, AE | VERSION OF A FORTRAN II. OUTPUT IS STRAP. |
| FORTRAN IV | | | | | CSC | UNIVAC 1107 | | | | | | 16K, DRUM, 3 TAPES | MCCARTY, C.L. | DUE JANUARY, 1963 |
| FORTRAN IV | | | | | CSC | UNIVAC 111 | | | | | | 16K, 4 TAPES | MCCARTY, C.L. | DUE JANUARY, 1963 |
| FORTRAN IV | J28-6212, 6203, 6243, 6236, 6209 | | | • | IBM | IBM 7040/7044 | | | A | C | A | 16K, CARD RDR, PRNTR, 4 TAPES | | COMPILER AS COMPONENT OF IBJOB PROCESSOR |
| FORTRAN IV | J28-6194,5,6,7 6283 | | | • | IBM | IBM 7090/7094 | 30 | DEC 62 | A | C | A | PRNTR, CARD RDR, 3 TAPES PLUS 5 UNITS- TAPES OR DISK | | COMPILER AS COMPONENT OF IBJOB PROCESSOR |
| FORTRAN | | | | | ENGLISH ELECT. | ENGLISH ELECT. KDF9 | | DEC 62 | A | C | A | 8K, 2-4 TAPES | DUNCAN, FG | ENABLES FORTRAN PROGRAMS TO BE PREPARED FOR TRANSLATION BY THE ALGOL COMPILER. |
| FORTRAN | | | | | AERE | FERRANTI ORION | | NOV 62 | G | C | A | 8K, 7 TAPES | TAYLOR, R | OFFSHOOT OF THE HARWELL/FORTRAN PROJECT. SOME PARTS OF WHICH ARE INDEPENDENT OF COMPILING MACHINES AND EXECUTION M/C. WRITTEN IN FORTRAN |
| ATLAS FORTRAN | | DUE 63 | 100 | FERRANTI | FERRANTI, AERE | ATLAS | 30 | DEC 62 | A | C | A | 8K, 3 TAPES, DRUM | PYLE, IC | TARGET LANGUAGE IS ASP OR BAS. INITIALLY FOR 7090 ALSO WRITTEN IN FORTRAN |
| ALGEBRAIC COMPILER | | | | | MINN'HONEYWELL/ COMPUTER USAGE | MH 800 | 60 | JUL 61 | | | A | 4K, 4 TAPES, RDR, PRNTR | COLEN, P | ACCEPTS FORTRAN II, 4 RUNS |
| ALTAC | TM-58 | FEB 61 | 175 | • | PHILCO | PHILCO 2000 | 8 | APR 60 | A | | A | 8K, 6 TAPES, RDR, PRNTR | GUERNACCINI, J | AUGMENTED FORTRAN II, SELF-ADAPTING TO CONFIGURATION. 2-PASS INTO TAC. |
| COMPACT | | | | | | RPC 4000 | 6 | OCT 61 | A | | A | 8K, 8K DRUM, P-TAPE RDR, PCH, TYPER | NATHER, RE | +2K TABLES. ACCEPTS AUGMENTED FORTRAN II. 1-PASS TO ROAR, 1-PASS TO ML AT 500 INSTR./MN |
| AUTOCODE | | | | | | PEGASUS | | | | | A | | GILL, S | |
| AUTOCODE | CS242A | JUN 61 | 73 | FERRANTI | MANCHESTER UNIV | MERCURY | 5 | APR 58 | G | B | A | 1K, 16K DRUM | BROOKER, RA | |
| UNICODE | U-1451-1 | OCT 58 | 45 | • | SPERRY-RAND | UNIVAC 1103A | 4 | NOV 59 | G | A | A | 6K, 16K DRUM, 7 TAPES | | AUTOMATIC SEGMENTATION |
| ALTRAN | | | 13 | • | EL-TRONICS | ALWAC III-E | | | A | | A | | MANDERFIELD, EL | ALGEBRAIC TRANSLATOR. |
| MADCAP I | ' | JUN 58 | 13 | • | LOS ALAMOS SCIENT. LABS | MANIAC II | 5 | APR 58 | | | A | 16K, FLP, 3 XREG, 3 TAPES, I/O | WELLS, MB | FORMULA TRANSLATION AND SIMPLE CONTROL STATEMENTS |
| MADCAP II | ' | JUN 59 | 59 | • | LOS ALAMOS SCIENT. LABS | MANIAC II | 8 | MAY 59 | | | A | 16K, FLP, 3 XREG, 3 TAPES, I/O | WELLS, MB | EXPANDED MADCAP I + LOOP STATEMENTS |
| MADCAP III | LAMS-2601 | AUG 61 | 82 | • | LOS ALAMOS SCIENT. LABS | MANIAC II | 11 | JUL 61 | | | A | 16K, FLP, 3 XREG, 3 TAPES, I/O | WELLS, MB | IMPROVED MADCAP II + SCRIPTING AND DISPLAYED DIVISION |
| COLASL | | | | | LOS ALAMOS | IBM STRETCH | | | | | A | | BALKE, K | |
| PROCOM | | | | | RAMO-WOOLDRIDGE CAE | RW 300 | | | G<br>I | A | AC | | CAE | FORTRAN NOTATION, DUE 1963 |

| LANGUAGE | MANUAL | | | | TRANSLATOR | | | | | | | | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDENTIFICATION | DATE | PP | PUBLISHER | CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFIGURATION | | |
| ALGOL | | | | • | BULL | GAMMA 60 | 3.0 | JUL 62 | I | C | A | 8K, CARD OR P-TAPE I/O OR TAPE | BULL | |
| ALGOL 60 | | SEP 62 | | • | ELLIOTT BROS. | NAT.ELLIOTT 803/503 | 8 | SEP 62 | G | C | A | ANY | HOARE, CAR | A FEW RESTRICTIONS ON ALGOL 60. TRANSLATED PROGRAM ONLY RUNS ON 803 WITH FLP OR 503 |
| ALGOL 60 | | | | • | EMI | EMIDEC 2400 | 2 | JUL 62 | G | B | A | 16K | BURGE, WH | TRANSLATION RESTRICTED AT PRESENT |
| DEUCE ALGOL | W/AT 844 | DEC 62 | 10 | • | ENGLISH ELECT. A.P. DIVISION | DEUCE MK IIA | 2 | NOV 61 | I | B | A | 8K, CARD RDR, P-TAPE I/O | RANDELL, B | SUBSET OF ALGOL 60 WITH SINGLE LETTER IDENTIFIERS, LESS SWITCHES, BOOLEANS, RECURSIVE PROCEDURES. |
| ALGOL KDF9 | | | | | ENGLISH ELECT. | ENGLISH ELECT. KDF9 | | SEP 62 | I | B | A | 8K, 2 TAPES, P-TAPE I/O | RANDELL, B | FULL ALGOL LESS DYNAMIC OWN ARRAYS, INTEGER LABELS, OPTIONAL SPECIFICATIONS. |
| KDF9 ALGOL | | DEC 61 | 100 | • | ENGLISH ELECT. | ENGLISH ELECT. KDF9 | 20-30 | DEC 62 | G | C | A | 8K, 2-4 TAPES, P-TAPE I/O | DUNCAN, FG | FULL ALGOL LESS DYNAMIC OWN ARRAYS, INTEGER LABELS, OPTIONAL SPECIFICATIONS. CONTAIN OPTIMISING AND OPERATING SYSTEM |
| ALGOL 60 | | | | • | U.OF N.CAROLINA | UNIVAC 1105 | 20 | | G | B | A | 8K, 32K DRUM, 4 TAPES | HANSON, J.W. | COMPLETE ALGOL 60. JAN. 63 |
| ALGOL 60 | | | | | SPERRY-RAND INT | UNIVAC 111 | | | | | | | PASTER, A. | DUE DEC. 1963 |
| ALGOL 60 | | | | | GKI | UNIVAC 1107 | | | | | | 16K, DRUM, 3 TAPES | PASTER, A. | DUE DEC. 1962 |
| ALGOL 60 | | | | | | RREAC | 4 | | G | C | A | 2 P-TAPE RDRS, 3 P-TAPE PCHS, TAPE | FOSTER, JM | DUE APR 63. |
| ALGOL 60 | ALGOL 60 REPORT | | | | IBM | IBM 709,7090 | 18 | DEC 60 | A | | A | 32K,8-10 TAPES,RDR,PRNTR | FRANCIOTTI, RG | ALGOL 60 LESS RECURSIVE PROCEDURES, OWN ARRAYS |
| ALGOL 60 | ALGOL 60 REPORT | | | | PRINCETON U. | CDC 1604 | 3 | JUN 61 | | | A | 32K,4 TAPES | IRONS, ET | ALGOL 60 + I/O. NO OPER. SYSTEM. +7K WORDS OF TABLES |
| ALGOL 60 | MR33 | OCT 60 | 67 | • | MATHEMATISCH CENTRUM | X-1 | 2.5 | AUG 60 | G | C | A | 8K, RDR, PRNTR | VAN WIJNGAARDEN,A | ALGOL 60 LESS DYNAMIC OWN ARRAYS. 2 ALTERNATE OPERATING SYSTEMS AT 2K WORDS EACH. |
| ALGOL 60 | | | | | DR. NEHER LABORATORIUM | STANTEC ZEBRA | 4 | MAY 62 | I | B | ALM | 8K | VAN DER POEL, WL | ALGOL 60 LESS DYNAMIC OWN ARRAYS. |
| INPUT LANGUAGE | (PRELIMINARY) | MAY 61 | 175 | • | INST. OF MATH. NOVOSIBERSK | M-20 | 12 | | G | C | A | 4K,16K DRUM, 4 TAPES, PCH, PRNTR | ERSHOV, AP | ALGOL 60 + VECTOR, MATRIX, ETC. AS SIMPLE VARIABLES, COMPLEX, CHAINS OF INEQUALITIES, INITIAL VALUES, FUNCTIONS YIELDED BY EXPRESSIONS. DUE LATE 62. |
| INPUT LANGUAGE | (REVISED) | MAR 62 | 100 | • | INST. OF MATH. NOVOSIBERSK | M-20 | 2 | | G | I | A | 4K,16K DRUM, 4 TAPES, PCH, PRNTR | ERSHOV, AP | |
| DASK ALGOL | (PRELIM) | NOV 60 | 29 | • | DANSK REGNECENTRALEN | DASK | 10 | AUG 61 | | | A | 2K,16K DRUM, 3 TAPES, FLX | NAUR, P | ALGOL 60 LESS OWN ARRAYS, RECURSIVE PROCEDURES |
| ALGOL 60 | FACIT-ALGOL 1 | DEC 61 | 49 | • | FACIT ELECT. AB | FACIT EDB | 11 | OCT 61 | G | C | A | 2K, 8K DRUM, 5-CHANNEL P-TAPE | DAHLSTRAND, I | ALGOL 60 LESS OWN ARRAYS,RECURSIVE PROCEDURES. RESTRICTIONS ON VARIABLE INDEX BOUNDS, EXPRESSIONS CALLED BY NAME, IDENTIFIER LENGTH. |
| ALGOL 60 | | | | | CARL ZEISS JENA | ZRA 1 | | | I | | A | 4K DRUM, CARD RDR, LINE PRNTR | KERNER, I | FULL ALGOL 60. HAND TRANSLATION TO AUTOCODE. AUTOMATIC TRANSLATION TO ML. DUE JUL 62 |
| ALGOL 60 | 5000-21001-P | JUL 61 | 30 | • | BURROUGHS | B-5000 | | | | | A | | | DUE SEP 62 |
| ALGOL 60 | | | | | ARMOUR RESEARCH | UNIVAC 1105 | 8 | | A | C | A | 8K, FLP, P-TAPE, 5 TAPES | MITMAN, B | ALGOL 60 + I/O. NO PROCEEDURES EXCEPT ELEMENTARY FUNCTIONS. RESTRICTED ARRAYS, SWITCHES. |
| ALGOL 60 | | APR 61 | 24 | • | OAK RIDGE NATL LABORATORIES | ORACLE | 3.2 | NOV 60 | A | C | A | 2K, 4 TAPES, P-TAPE RDR | GRAU, AA | ALGOL 60 LESS PROCEDURES, BLOCKS, SWITCHES. ADDED FEATURES FOR I/O, TAPE FILES, SEGMENTATION |
| ALGOL 60 | ALCOR, PART 1 | MAR 61 | 75 | OLDENBOURG MUNCHEN | U. MAINZ | ZUSE Z22, Z22R | 2 | MAY 61 | G | C | A | 8K DRUM | BAUER, FL | ALCOR + ALGOL 60 LESS OWN, RECURSIVE PROCEDURES. REPLACED PILOT TRANSLATOR USED FEB 59 TO APR 61 |
| ALGOL 60 | PART 2 | DEC 61 | | | U. MAINZ | SIEMENS 2002 | 5 | DEC 61 | G | C | A | 2K, 10K DRUM | SAMELSON, K | ALCOR. WILL REPLACE SIEMENS PILOT TRANSLATOR. |
| ALGOL 60 | | | | | T.H. MUNCHEN | PERM | 8.2 | AUG 61 | G | C | A | 2K, 8K DRUM, P-TAPE RDR, PCH | SEEGMULLER, H | ALCOR |
| ALGOL 60 | | | | | E.T.H. ZURICH | ERMETH | 8 | JUL 60 | G | C | A | 10K DRUM | SCHWARZ, HR | ALCOR WITH LIMITED PROC. HEADING, STATEMENTS, 1 BLOCK |
| ALGOL 60 | | | | | STANDARD ELECTRIC LORENZ | ER 56 | 4 | JAN 62 | G | C | A | 3.2K, 12K DRUM | | SOURCE + STD. ELECTRIC LORENZ INFORMATIKWERK, STUTTGART ABT. ISP PRO |
| ALGOL 60 | | | | | T.H. WIEN | MAILUFTERL | 1.5 | MAR 61 | G | C | A | 10K DRUM | | ALCOR LESS PROCEDURES. SEE KUDIELKA ET AL, LOGALGOL, EUR. RES. OFFICE REPORT DA-91-591-EUC-1430 |
| ALGOL 60 | | | | | DUKE UNIVERSITY | IBM 7070 | 5 | | G | B | A | 10K, RDR, PCH, 0 TAPES | GALLIE, T JR. | ALGOL 60 LESS OWN, RECURSIVE PROCEDURES. GRAU-TYPE. REST DOCUMENTED IN ALGOL. TRANSLATES 200-300 LINES/MINUTE. DUE JUN 62 |
| ALGOL 60 | | | | | VUMS PRAGUE | EPOS 1 | 6 | | G | A | A | .3K, 5K DRUM, 3 TAPES, I/O | KINDLER, E | LIMITED ALGOL 60. PROCESSOR WORKS IN CORE. EPOS PERMITS 5 PROGRAMS TO TIME-SHARE. |
| ALGOL 60 | | | | | VUMS PRAGUE | MSP | 2 | | G | A | AC | P-TAPE I/O | FABIAN, V | LIMITED ARRAYS AND DESIGNATIONAL EXPRESSIONS. DUE 62 |
| ALGOL | | | | | SYLVANIA ELECT. | SYLVANIA 9400 | | | | | A | 16K, 5 TAPES, 1 I/O | HATCH, A | NOT COMPLETELY SPECIFIED. DUE SEP 62 |
| ALGOL | | | | | GENL. KINETICS | UNIVAC 1107 | | | | | A | | | WILL ACCEPT MIXED ALGOL(58/60) AND FORTRAN II |
| ALGEBRAIC COMPILER | 220-21011-P | JAN 61 | 108 | • | BURROUGHS | BURROUGHS 220 | 3.5 | MAY 60 | G | | A | 5K, CARDATRON, 2 TAPES, I/O | SPEIERMAN, KH | MODIFIED MIX ALGOL 58/60. 2000 WORDS OF TABLES. COMPILES 500 ML INSTR./MINUTE. |
| ALGEBRAIC COMPILER | 205-21003-P | FEB 61 | 50 | • | BURROUGHS | BURROUGHS 205 | | NOV 60 | | | A | FLP, FLX, P-TAPE | SPEIERMAN, KH | ALGOL 58 |
| ALCOM | | | | | BENDIX | G-20 | | | | | A | 8K, 2 TAPES, RDR, PCH, PRNTR | BAUER, F | ACCEPTS MIXED ALGOL(58/60) + FORTRAN II. DUE FEB 62 |
| ALGOL | PROG. ALGOL | | | | INST. MATH.APP. | IBM 7090<br>SEPSEA CAB 500 | | DEC 62 | G | A | | | INST. DE MATH. GRENOBLE | DUE JUN 63 |
| ALGO | T21 | AUG 60 | 28 | • | BENDIX | G-15 | 9 | JUL 60 | I | A | A | P-TAPE, ALPHA TYPER, OPT 0-2 TAPES | BAUER, F | LIMITED SUBSET OF ALGOL. 3 PASSES TO COMPILE. |
| JOVIAL | TM555 | JUN 61 | 190 | • | SDC | IBM 7090 | 50 | | | | AM | 32K, 6 TAPES, RDR, PRNTR | SCHWARTZ, J | MODIFIED AND EXTENDED ALGOL 58 (JULES OWN VERSION OF IAL). ONE BASIC GENERATOR. SEPARATE TRANSLATORS FOR EACH MACHINE. DUE JAN 62 |
| JOVIAL | | | | | SDC | MIL. COMPUTER | 50 | | | | AM | | SCHWARTZ, J | DUE JEC 61 |
| JOVIAL | | | | | SDC | PHILCO 2000 | | | A | | AM | 16K, 32K DRUM, 6 TAPES, PRNTR | CLARK, E | DUE DEC 61 |
| JOVIAL | | | | | SDC | CDC 1604 | 50 | | | | AM | 32K, 8 TAPES, PRNTR | BRATMAN, H | DUE DEC 61 |
| JOVIAL | | | | | SDC | Q-7 | 80 | | | | AM | 65K, 4 TAPES, DRUMS | JACKSON, C | DUE JUN 61 |
| CLIP | TM543 | OCT 60 | 18 | • | SDC | IBM 7090 | 18 | 59 | A | | AM | 32K, 10 TAPES, I/O, SOS SYSTEM | BRATMAN, H | ALGOL-LIKE, CHARACTER MANIPULATION. USED TO WRITE OTHER TRANSLATORS. |
| MAD | | FEB 61 | 149 | • | U. MICHIGAN | IBM 704 | 9 | FEB 60 | | | AM | 8K, 4 LOGICAL DRUMS, 4 TAPES, PRNTR | ARDEN, B | VARIATION OF ALGOL 60. NO. OF INSTRUCTIONS DOES NOT INCLUDE TABLES. |
| MAD | | | | | U. MICHIGAN | IBM 709/7090 | 8.5 | MAR 61 | | | AM | 32K, 4 TAPES, I/O | ARDEN, B | HIGH SPEED COMPILATION. |
| NELIAC | | | | | NAVAL ELECT LAB | M-460 COUNTESS | 12 | MAR 59 | | | AM | | HALSTEAD, M | DIALECT OF ALGOL 58. ALSO NAVAL POSTGRADUATE SCHOOL, MONTEREY. ALL PROCESSORS ARE WRITTEN IN NELIAC LANGUAGE FOR BOOTSTRAPPING. |
| NELIAC | | | | | NAVAL ELECT LAB | CDC 1604 | | | | | AM | | HALSTEAD, M | |
| NELIAC | | | | | | M-490 | | JUN 61 | | | AM | | | |
| NELIAC | | JUL 61 | 25 | • | U.CAL.-BERKELEY | IBM 704 | 12 | | | | AM | 32K, TAPES | HUSKEY, HD | |
| NELIAC | | JAN 61 | 25 | • | RAMO-WOOLDRIDGE | IBM 709 | 12 | | | | AM | 32K, TAPES | HUSKEY, HD | |
| NELIAC | (PRELIMINARY) | AUG 61 | 30 | • | LOCKHEED MSC | IBM 7090 | 14 | | | | AM | 32K, TAPES | HUSKEY, HD | COMPILES 5300 INSTR/MIN. INCLUDES FORTRAN I/O |
| NELIAC | | | | | | BURROUGHS 220 | | | | | AM | | | VIA M-460 |
| NELIAC | | | | | | PB-250 | | | | | AM | | | VIA M-460 |
| NELIAC | | | | | | PHILCO CXPQ | | | | | AM | | | VIA M-460 |
| DIALGOL | | | | | U.CAL.-BERKELEY | IBM 7090 | | | G | C | AMS | 32K, TAPES | HUSKEY, HD | SUBSET OF ALGOL. EMPHASIS ON EFFICIENCY. DUE JUL 62 |
| SLANG | | AUG 60 | 38 | • | IBM FSD | IBM 709 | | | | | AM | 32K, 4 TAPES, I/O | SIBLEY, R | DUE AUG 62. PATTERNED ON ALGOL 58 + METALINGUISTICS. TO PRODUCE MULTIPLE MACHINE LANGUAGES. |
| WIZOR | | | | • | GENL. ELECTRIC | GE 225 | 6 | OCT 61 | G | A | AM | 8K(20-BIT), RDR, PCH, OPT PRNTR, TYPER | SCOTT, DW | 1-PASS. 1K TABLES. TRANSLATE 1K INSTR/MINUTE. SELF-COMPILING. FIXED PT. ARITH. ONLY. FORMAT SIMILAR TO ALGOL+TABSOL. BOOLEAN AND LITERAL CHAR. VALUE ARITH. MANUAL DUE DEC 61 |
| WIZ | | NOV 61 | 50 | • | GENL. ELECTRIC | GE 225 | 7 | | G | A | A | (SAME AS WIZOR) | SCOTT, DW | DUE JAN 62. WIZOR WITH MIXED FIXED AND FLT. PT. ARITH. |

| LANGUAGE | MANUAL |  |  |  | TRANSLATOR |  |  |  |  |  |  |  | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | IDENTIFICATION | DATE | PP | PUBLISHER | CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFISURATION |  |  |
| ILLIAD | (PRELIMINARY) | JUN 61 | 32 | * | SO. ILLINOIS U. | IBM 650 | 2 |  | G | A | A | 2K DRUM, RDR-PCH | BLOSE, W | SIMILAR TO MAD |
| SALT | 1034 | MAR 60 | 29 | * | AUTONETICS | RECOMP II |  | JAN 60 | A |  | A | FLP | JELINSKI, Z | FORTRAN-LIKE ALGEBRAIC COMPILER TO SCRAP |
| AUTOCOM | ' | DEC 61 | 13 | * | AUTONETICS | RECOMP III |  |  |  |  | A | FLP | JELINSKI, Z | FORTRAN-LIKE, 1-PASS, OPTIMIZING. DUE DEC 61 |
| SCOPAC | ' | OCT 61 |  | * | AUTONETICS | RECOMP II |  | OCT 61 |  |  | A | FLP | JELINSKI, Z | OPTIMIZING FORTRAN-LIKE COMPILER |
| IT | ' | SEP 59 | 64 | * | PURDUE UNIV. | BURROUGHS 205 |  | SEP 59 |  |  | A | 2 TAPES, FLP, CARD UNIT | SPEIERMAN, KH | PURDUE COMPILER II, IT WAS PC I |
| IT |  |  | 47 | * | CARNEGIE TECH | IBM 650 |  |  | A |  | A |  | PERLIS, AJ | 650 LIBRARY NO. 2.1.001 |
| IT |  |  | 60 | * | CASE INSTITUTE | IBM 650 | 2 |  | A |  | A | 2K, ALPHA | CONWAY, M | CALLED RUNCIBLE(IMPROVED IT). FORTRUNCIBLE HAS NAMED VARIABLES. |
| IT |  |  | 50 | * | CASE INSTITUTE | BURROUGHS 220 | 3 |  |  |  | A | 5K, 2 TAPES, CARDATRON | HAYNAM, GE | CALLED RUNCIBLE 220 |
| IT |  | JUN 58 | 30 | * | RAMO-WOOLDRIDGE | UNIVAC 1103A | 16 | MAR 58 |  |  | A | 4K, 2 TAPES, FLP, RDR, PRNTR | VAN ZOEREN, H |  |
| IT |  | APR 62 |  | * | U. NO. CAROLINA | UNIVAC 1107 | 16 | SEP 59 | G |  | A | 8K, 16K DRUM, FLP, 3 TAPES | HANSON, JW |  |
| IT |  | FEB 62 | 100 | * | ARMOUR RESEARCH | UNIVAC 1105 | 8 | AUG 61 | G | A | A | 8K, CORE, 10K DRUM, 4TAPES, P-TAPE | WRENN, M. | IT COMPILER WITH MODIFIED I/O |
| CALCULUS |  |  | 50 | * | CASE INSTITUTE | UNIVAC I |  |  |  |  | A |  | CONWAY, M | EXPANDED FORTRUNCIBLE |
| ORBIT | 59-9-20 | SEP 59 | 85 | * | OAK RIDGE NATL LABORATORIES | ORACLE | 5 | JAN 59 | A | C | A | 2K, 4 TAPES, P-TAPE | GRAU, AA | EXTENDED IT |
| GAT |  | APR 59 | 97 | * | U. MICHIGAN | IBM 650 | 2 | FEB 59 |  |  | A | 2K, XREG, FLP | ARDEN, B | IMPROVED IT |
| GAT |  | JUL 62 | 100 |  | U. NO. CAROLINA | UNIVAC 1105 | 12 | JUN 61 | G | C | A | 8K, 16K DRUM, FLP, 3 TAPES | HANSON, JW | EXTENDED 650 GAT, WRITTEN IN IT, THEN IN GATE, |
| GATE |  |  | 200 | * | CARNEGIE TECH | IBM 650 RAMAC | 2.4 | 58 |  |  | A | 2K, XREG, FLP | PERLIS, AJ | GAT EXTENDED. |
| CORREGATE |  | JAN 59 | 220 | * | CARNEGIE TECH | IBM 650 RAMAC | 3.5 | JAN 59 |  |  | A | 2K, XREG, FLP | PERLIS, AJ | GATE+SOURCE LANG. CORRECTING. U. OKLA. REPRINT MAR 61 |
| GATE-20 |  | JUL 61 | 50 | * | CARNEGIE TECH | BENDIX G-20 | 3 | JUN 61 | G | B | A | 8K, RDR, PRNTR | PERLIS, AJ | EXTENDED CORREGATE. 1-PASS |
| MYSTIC | ASTIA-134274 | FEB 58 | 61 | * | JOHNS HOPKINS U | IBM 650 | .8 | SEP 57 |  |  | A | 2K DRUM, RDR-PCH, OPT PRNTR | GORMAN, TP | SIMPLE ALGEBRAIC COMPILER |
| MYSTIC |  |  |  |  | JOHNS HOPKINS U | UNIVAC 1103A | 3 | SEP 57 |  |  | A | 16K, 2 TAPES, RDR, PCH, PRNTR | GORMAN, TP |  |
| MYSTIC |  |  |  |  | JOHNS HOPKINS U | IBM 704 | 3 | SEP 57 |  |  | A | 8K, NO TAPES, RDR, PRNTR | GORMAN, TP |  |
| MYSTIC 2 |  |  |  |  | JOHNS HOPKINS U | IBM 7090 | 5 | FEB 60 |  |  | A | 32K, 1 TAPE | GORMAN, TP |  |
| FORAST |  | AUG 60 | 41 | * | BRL, ABERDEEN | ORDVAC | 7 | JUN 60 | G | B | A | 4K, 10K DRUM, CARD RDR-PCH | CAMPBELL, L | FORMULA TRANSLATOR + ASSEMBLY |
| FORAST |  |  |  |  | PROVING GROUND | BRLESC | 4.5 |  |  |  | A | 4K, 6 TAPES, CARD RDR-PCH | CAMPBELL, L | DUE OCT 61 |
| SALE II |  | NOV 61 | 36 | * | A.O.SMITH | IBM 705 II | 7.2 | JUN 61 | G | B | A | 40K CHAR, 2 TAPES, RDR, PRNTR | BRITTENHAM, R | 10-DIGIT FLT. PT., PROCESS 250 STATEMENTS/MINUTE |
| ALPHACODE |  | 58 | 50 | * | ENGLISH ELECT. | ENGLISH ELECT. DEUCE | 22<br>3 | 59<br>57 | C<br>I | C<br>I | A | DEUCE MK I | DUNCAN, FG | SEE ANNUAL REVIEW IN AUTOMATIC PROGRAMMING VOL.1 TRANSLATOR IS OPTIMISING. |
| AUTOCODE |  |  |  |  | ENGLISH ELECT. | ENGLISH ELECT. KDF9 | 2 | OCT 62 | A | C | A | 8K, TAPES, P-TAPE I/O | RANDELL, B | CONVERTS AUTOCODE INTO ALGOL 60 |
| ACE AUTOCODE |  | FEB 61 | 27 | * | NATIONAL PHYS. LABORATORY | A.C.E. | 1.8 | DEC 60 | I | C | A | 2 8K DRUMS | WOODGER, M | IDENTICAL WITH THE MERCURY AUTOCODE, LESS LIBRARY FACILITIES OR MATRIX OPERATIONS. |
| EXTENDED AUTOCODE |  | MAR 62 |  | * | LEEDS U. COMP. LAB. | FERRANTI PEGASUS | 3 | JAN 62 | I | B | A | 7K.TAPE | MITCHELL, AJ |  |
| MANCHESTER AUTOCODE-MAC |  |  |  | MANCHESTER UNIVERSITY | ICT LTD. | ICT 1301 |  | 62 | G | I | A | ANY | ELLIS, PV | LANGUAGE RESTRICTIONS FOR LESS THAN 1.2K MEM. |
| MERCURY AUTOCODE-CHLF |  | NOV 61 | 51 | * | G.E.C. COMPUTER | FERRANTI MERCURY | 8 | SEP 61 | G | C | A | TWO LEVEL | PAYNE, JCF | BASED ON MANCHESTER MERCURY AUTOCODE - SEE FERRANTI PUBLICATION MERCURY AUTOCODE MANUAL LIST CS242A- MODIFIED TO MEET SPECIAL REQUIREMENTS |
| EXTENDED MERCURY AUTOCODE |  | END 62 |  | * | FERRANTI | ORION | 15 |  | A | C | A |  | FOSTER, DM | TARGET LANGUAGE IS ORION BASIC INPUT. DUE JAN 63 ALSO BEING WRITTEN FOR ATLAS. |
| AUTOCODE |  | SEP 61 | 34 | * | CAMBRIDGE UNIV. | EDSAC 2 | 3.5 | OCT 61 | G | C | A | 17K, 4 TAPES | HARTLEY, DF | SOME OPTIMIZATION ON COMMON SUB-EXPRESSIONS AND ON B-REGISTER LOADING. |
| BELL | C28-4024 | OCT 59 | 92 | IBM | BELL TEL. LABS | IBM 650 | 1 | 55 |  | I | A | 2K, RDR-PCH |  | WOLONTIS VERSION |
| BELL |  | FEB 60 | 16 | BURROUGHS | SHELL OIL CO | BURROUGHS 205 |  | FEB 60 |  | I | A | FLP, P-TAPE | SPEIERMAN, KH | WOLONTIS VERSION |
| BELL (SIS) |  |  | 90 | * | CASE INSTITUTE | BURROUGHS 220 |  |  |  | I | A |  | HAYNAM, GE | WOLONTIS VERSION + STATISTICAL FUNCTIONS |
| BELL (SIS) |  |  |  |  | CASE INSTITUTE | IBM 650 | 1 |  |  | I | A |  | HAYNAM, GE | WOLONTIS VERSION + STATISTICAL FUNCTIONS |
| BELL |  |  |  |  | IBM (ENDICOTT) | IBM 704 |  |  |  |  | A |  |  |  |
| BELL |  |  |  |  | SHELL DEV. CO. | USSI 80 | 4 | JAN 61 |  | I | A | 5K, DRUM, CARD | HERTEL, P.H. | WOLONTIS VERSION |
| BELL |  |  |  |  | COOKE ELECTRIC | USSI 90 | 4 | JAN 61 |  | I | A | 5K, DRUM, CARD | HERTEL, P.H. | WOLONTIS VERSION, INCLUES TRACE |
| BAMABELL |  |  |  |  | U. OF ALABAMA | USSI 80 | 2 | JAN 62 |  | I | A | 2.4K DRUM, CARD | HERTEL, P.H. | ABBREVIATED WOLONTIS VERSION |
| APT |  |  |  |  |  | IBM 650 |  |  |  |  |  |  |  |  |
| APT |  |  |  |  |  | UNIVAC 1103A |  |  |  |  |  |  |  |  |
| APT |  |  |  |  |  | ORDVAC |  |  |  |  |  |  |  |  |
| ADES II | REPORT 4209 | FEB 56 | 50 | * | NAVAL ORDNANCE | IBM 650 | 3 |  |  |  |  |  | BLUM, E | POLISH NOTATION, ALGEBRAIC COMPILERS. DO |
| ADES II | REPORT 4411 | FEB 56 | 94 | * | LABORATORY | IBM 704 |  |  |  |  |  |  | BLUM, E | SOME STORAGE ALLOCATION AND FLOWCHARTING. |
| KISS |  |  |  |  | CHRYSLER | IBM 650 |  |  |  | I | A |  |  |  |
| KISS |  |  |  |  | COMPUTER USAGE | IBM 704 |  |  |  |  | A |  |  |  |
| AFAC |  |  |  |  | ALLISON G.M. | IBM 704 |  |  |  |  | A |  |  |  |
| AFAC |  |  |  |  | ALLISON G.M. | IBM 705 I,II |  |  |  |  | A |  |  |  |
| SIR |  |  |  |  | IBM | IBM 650 |  |  |  |  | A |  |  |  |
| COMPILER I |  |  |  |  |  | UNIVAC 1103A |  |  |  |  | A |  |  |  |
| BOEING COMPILER |  |  |  |  | BOEING AIRPLANE | UNIVAC 1103A |  |  |  |  | A |  |  |  |
| FAP |  |  |  |  | LOCKHEED | UNIVAC 1103A |  |  |  |  | A |  |  |  |
| RAFT IV | 1058 | FEB 61 | 11 | * | AUTONETICS | RECOMP II | 1 | JAN 60 |  | I | A | FLP | JELINSKI, Z | SINGLE-CHARACTER VARIABLES |
| RIP-3000 |  | OCT 61 | 5 | * | AUTONETICS | RECOMP III |  | OCT 61 |  | I | A | FLP | JELINSKI, Z |  |
| ACT I,III |  |  |  |  | ROYAL MCBEE | LGP-30 |  |  |  |  | A |  |  | ACT III BY NATIONAL CARBON |
| JAZ |  |  |  |  | PHILCO WDL | LGP-30 |  |  |  | I | A |  |  |  |
| SPEED |  |  |  |  | ROYAL MCBEE | LGP-30 |  |  |  |  | A |  |  |  |
| FLIP |  |  |  |  | HUMBLE OIL | BENDIX G-15 |  |  | - | I | A |  |  |  |
| INTERCOM | T11-3 | JUN 59 | 27 | * | BENDIX | BENDIX G-15 | .9 | JUN 58 | I |  | A |  | BAUER, F | LIKE PRINT I (705). |
| DAISY | 465-15627-474 | OCT 57 | 22 | * | BENDIX RADIO | BENDIX G-15 | 1.4 | JAN 58 | - | I | A |  | BAUER, F | 201,202 VERSIONS |
| PINT |  |  |  |  | ROYAL MCBEE | RPC-4000 |  |  |  | I | A |  |  |  |
| ACE | CRT-916,AECL-996 | MAR 60 | 32 | * | AEC OF CANADA | BURROUGHS 205 |  | MAR 60 |  |  | A | FLP, 1 TAPE, FLX, PCH CARD I/O | SPEIERMAN, KH |  |
| APS III |  | FEB 58 | 56 | * | WESTINGHOUSE | BURROUGHS 204 | 2 | JAN 58 |  |  | A | FLP, 1 TAPE, P-TAPE | SPEIERMAN, KH | POLISH NOTATION, ORIGINAL APS HAD 1K INSTRUCTIONS |
| DUMBO | (PRIMER)<br>(COMMAND LIST) | APR 59<br>APR 57 | 15<br>10 | *<br>* | BABCOCK AND WILCOX | BURROUGHS 205 | 3 | APR 57 | G | I | A | FLP, 1 TAPE, PCH CARD I/O | SPEIERMAN, KH | OPERATES INTERPRETIVELY FOR DIAGNOSIS, COMPILES FOR PRODUCTION |
| SIMPLE | TECH. REPORT 112 | JAN 59 | 25 | * | CAL TECH | BURROUGHS 205 |  |  |  |  | A |  |  |  |
| SIMPLE CODE | IN ENGLISH IN HETPTT BEDRIJF | 60 | 60 | * | DUTCH PTT | STANTEC ZEBRA | 1 | APR 58 | G<br>I | B | A | 4K | VAN DER POEL, WL | PARTLY TRANSLATION, PARTLY INTERPRETATION |
| SIMPLE CODE |  |  |  |  | ELECTROLOGICA | ELECT. X1 | 1 | APR 62 | G | B | A | 4K | VAN DER POEL, WL |  |
| SIMPLE CODE |  |  |  |  | TNO | ELLIOT 803 | 1 | JUN 62 | G |  | A | 4K | VAN DER POEL, WL |  |

| LANGUAGE | MANUAL | | | | TRANSLATOR | | | | | | | | SOURCE OF INFORMATION AND VERIFICATION | NOTES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDENTIFICATION | DATE | PP | PUBLISHER | CONSTRUCTOR | MACHINE | 1 | RUN | 2 | 3 | 4 | MINIMUM CONFIGURATION | | |
| UGLIAC | | | | | UNITED GAS CORP | BURROUGHS 205 | | | | | A | 4K, FLP, 1 TAPE, FLX, P-TAPE | SPEIERMAN, KH | |
| ABC | | | | | AWRE,ENGLAND. | IBM 704 | | | | | A | | GLENNIE, AE | |
| FORC2 | | | | | GE HUNTSVILLE | IBM 704 | | | | | A | | | |
| KOMPILER3 | | | | • | LIVERMORE AEC | IBM 704 | | | | | A | | | |
| L-1 | | | | | TECH.OPERATIONS | IBM 704 | | | | | A | | | |
| PACT-IA | (PRELIMINARY) | JUL 57 | 150 | RAND CORP. | (PACT GROUP) | IBM 704 | 30 | JUL 57 | | | A | 4K, 1 DRUM, 4 TAPES, RDR, PCH, PRNTR | STEEL, TB | |
| PRINT I | 32-7334-1 | OCT 56 | 58 | • | IBM | IBM 705 I,II | 7 | AUG 56 | I | C | A | 20K, 4 TAPES, NO DRUM | BEMER, RW | ALSO PRIMER, 32-7855(74PP),SEP 57 |
| PRINT 08 | | FEB 58 | 45 | • | A.O. SMITH | IBM 705 I,II | 7 | | I | C | A | 20K. | BRITTENHAM, R | |
| L-2 | | | | | | IBM 709/7090 | | | | | A | | | |
| MAGNET | | | | | | IBM 709/7090 | | | | | | | | |
| CALINT | | | | | CONTROL DATA | CDC 160 | | | - | I | | | | |
| MAGE | | JAN 62 | 70 | • | SETI | PB 250 | 2 | YES | G | C | A | | SETI | |
| JOB | | OCT 61 | 10 | • | SETI | PB 250 | .5 | YES | G | B | A | | SETI | |
| PAF-CAB 500 | SEPSEA 661-500 | 60 | | SEPSEA | SEA | CAB 500 | 1.8 | YES | G | C | A | | SEPSEA | TRANSLATED MACHINE LANGUAGE IMPLEMENTS CODES AND MICRO-PROGRAMS ON MAGNETIC DRUM. |
| AUTOPROG-CAB 500 | SEPSEA 261-500 | 60 | | SEPSEA | SEA | CAB 500 | .4 | YES | G | I | A | | SEPSEA | |
| AP2 | CMB 0012 | 57 | 91 | • | BULL | GAMMA ET/AET | | YES | G | C | A | | BULL | SYMBOLIC THREE ADDRESS LANGUAGE- CT FOR GAMMA 60 ACCEPTS AP2 |
| C2 | CMBE 0012<br>CNCE D.40/62 | 62 | 40 | • | BULL | GAMMA ET/AET | | YES | G | C | A | 8K | BULL | SYMBOLIC LANGUAGE, 3 OR MORE INDEXED ADDRESSES, FOR SINGLE OR DOUBLE PRECISION, 9 OR 18 DIGIT, COMPLEX NUMBERS, MATRICES. GAMMA 60 CT ACCEPTS C2 |
| PAM | CMBE 2001/E2002 | 57 | 53 | • | BULL | GAMMA ET/AET | | YES | I | B | A | 4K | BULL | MATRIX CALCULATION ASSEMBLY PROGRAM |
| CT | CNCE D.29/61 | 61 | 27 | • | BULL | GAMMA 60 | 3.5 | YES | G | C | AB | | BULL | SYMBOLIC ASSEMBLY CODE FOR MACHINE CODE, SUB-PROGRAMS, STANDARD FUNCTIONS, PROGRAMS IN VARIOUS SYMBOLIC LANGUAGES AP2, C2, ALGOL IN DEVELOPMENT |
| CAM | | | | IBM 650 USERS | MEUDON OBSERVATORY | IBM 650 | 1.5 | YES | A | A | AB | STANDARD | ARSAC, M | FIXED AND FLOATING POINT ARITHMETIC |
| PIAS | | | 60 | • | SNECMA | GAMMA AET | | YES | G | C | A | 8K | SNECMA | ASSEMBLY-INTERPRETIVE PROGRAM |
| PIAS 2 | | JUN 61 | | • | SNECMA | IBM 650 | 2.5 | MAY 61 | G | A<br>C | A | FLP, XREG | SNECMA | RELATED TO PIAS 1 BY VARIANCE |
| ROYAL AIR-CRAFT EST. | T.N. MS 39 | SEP 57 | 8 | • | ROYAL AIRCRAFT ESTABLISHMENT | DEUCE | 1 | SEP 57 | G | C | A | | DODD, KN | CONVERTS INSTRUCTIONS OF A SIMPLE FUNCTION-AND-ADDRESS TYPE INTO MACHINE CODE. |
| BEVERLEY | W/AT 786 | OCT 61 | 28 | • | ENGLISH ELECT. | DEUCE MK I, II, IIA | 5 | APR 59 | I | C | A | 8K, CARD I/O | BROYDEN, CG | A SPECIAL-PURPOSE INTERPRETIVE SCHEME FOR BESSEL FUNCTION CALCULATIONS. |
| EASICODE | W/AT 585 | REV OCT 60 | 20 | • | ENGLISH ELECT. | DEUCE MK I, II, IIA | 5 | NOV 59 | G | C | A | 8K, CARD I/O | RANDELL, B | THREE ADDRESS INSTRUCTIONS, WITH EXTENSIVE ORDER MODIFICATION FACILITIES. FXP. |
| ELLIOTT 803 A2 803 A102 | | JUL 60 | 56 | A.P. DIV. • | ELLIOTT BROS. | NAT. ELLIOTT 803/503 | 4 | SEP 60 | G | C | A | ANY | HOARE, CAR | 803 A2 IS FOR 803 WITHOUT FLP. 803 A102 REQUIRES A FLP UNIT. BOTH LANGUAGES ARE IDENTICAL. SEE ANNUAL REVIEW OF AUTOMATIC PROGRAMMING, VOL II. |
| GENERAL SIMULAT. PROGRAM | VOL I<br>VOL II | DEC 60<br>OCT 61 | 79<br>67 | • | UNITED STEEL COMPANIES O.R. DEPARTMENTS | PEGASUS | 3 | MAR 60 | G | C | S | | TOCHER, KD | |
| GIPSY | CP 5323 | SEP 61 | | • | E.M.I. | EMIDEC 2400 | 1 | MAR 61 | A<br>B | C<br>I | ILM | ANY | BURGE, WH | USED MAINLY IN PROGRAM STORAGE, AMENDMENT AND RETRIEVAL FROM TAPE AND IN PRODUCING COMPILERS. |
| LANGUAGE H | | APR 61 | 46 | • | NATIONAL CASH REGISTER, UK | 405 M | 20 | DEC 61 | G | C | B | 1K, 4 TAPES, DISC, 2 P-TAPE RDR,PCH, TELEPRNTR | CORMACK, AS | VERSION TO PRODUCE CODE FOR 803 SCHEDULED FEB 62. |
| MATRIX | | | | • | ROE, AV | FERRANTI MK1 | 3 | JUN 59 | I | B<br>I | A | BASIC MACHINE | MORTON, JP | MATRIX OPERATIONS - INTERPRETIVE. NEEDS FOR FLP, ETC DISGUISED |
| NEBULA | | NOV 60 | 136 | • | FERRANTI | ORION + ATLAS | 30 | | A | A | B | | FRASER, AG | DUE LATE 62 |
| PIGEON | MATH 78 TECH. NOTE | DEC 61 | 21 | • | ROYAL AIRCRAFT ESTABLISHMENT | MERCURY | 4 | JUL 61 | I | I | C | | FOWELL, EJC | FOR E.M.I. NUMERICALLY CONTROLLED 2-D GRINDING M/C. OUTPUT TAPE IS TOOL INPUT TAPE.FOR 2-D PLOTTER-BOARD. DIAGNOSTIC CHECKS. |
| QUIGOL | W/AT 869 | APR 62 | | • | ENGLISH ELECT. | DEUCE MK I,II | .5 | MAR 62 | G | B | A | 8K DRUM, CARD RDR, PCH | PARKYN, DM | FORMULA INPUT FXP |
| INTERCOM | | | | • | SPERRY RAND | USSI 90 | 5 | MAY 62 | G | B | A | 5K, DRUM, CARD | LANNI, N. | LGP.90 PSEUDO CODE INPUT TO MACHINE CODE WHILE LOADING |
| E . 1 | | | | | SPERRY-RAND | USS 80 | 47 | OCT 61 | A | B | AB | 5K, DRUM, CARD | DANDO, W. | EDIT GENERATOR |
| PROGENY | | | | | SPERRY-RAND | USS 90 | 10 | OCT 61 | A | A | B | 5K, DRUM, CARD | DANDO, W. | REPORT GENERATOR |
| MATRIX MATH.COMP | OPM. 2499 | NOV 57 | 64 | • | FRANKLIN INST. | UNIVAC 1 | 100 | JUL 57 | G | P | A | 1K, 8 TAPES | MCGINN, L.C. | WILL INVERT UP TO 300 X 300 |
| STEVE | W/AT 786 | OCT 61 | 28 | • | ENGLISH ELECT. | DEUCE MK I, II, IIA | 5 | DEC 58 | I | C | A | 8K DRUM, CARD RDR, PCH | BROYDEN, CG | SPECIAL-PURPOSE INTERPRETIVE SCHEME FOR THERMODYNAMIC CALCULATIONS. |
| TABULAR INTERPRE-TIVE SCHEME | | | | • | ROE COMPUTER GROUP | FERRANTI MK I, II, IIA | 8 | JUL 58 | I | B<br>I | A | BASIC MACHINE | MORTON, JP | SIMPLE INSTRUCTIONS OPERATE ON SINGLE NUMBERS OR COLUMNS OF NUMBERS. |
| PRODIGE | 10.1322 | JAN 62 | 20 | • | IBM FRANCE | IBM 1620 | .8 | AUG 61 | G | C<br>I | A | 20K | IBM FRANCE | ASSEMBLES AND EXECUTES IMMEDIATELY OR NOT, OPTIONALLY, OR PUNCH AND RUN. |
| COSIMA | SCEA 124 | APR 62 | 15 | CEA | | IBM 1620 | 1.3 | YES | I | I | A | MIN.+ INDIRECT ADDRESSING | CEA | |
| DPVF INT. | 02 1 004FT | 59 | 6 | FACULTE SC TOULOUSE | OULES, H | IBM 650 | 1 | 59 | I | I | A | STANDARD | MINISTERE DE LA MARINE | DIRECTLY INTERPRETED, SAME LANGUAGE AS DPVF COMPILER, WITH ADDITIONAL FUNCTIONS |
| DPVF COMP. | DPVF | 61 | 10 | FACULTE SC TOULOUSE | OULES, H | IBM 650 | 1.5 | 61 | G | A | A | | MINISTERE DE LA MARINE | COMPILER FOR DPVF LANGUAGE |
| IVF | LANGUAGE 2 SIVF | 62 | 70 | FONDERIE DE RUELLE | OULES, H | SEA | .7 | 62 | | I | A | STANDARD | MINISTERE DE LA MARINE | OBTAINED BY TRANSLATION OF SIVF WITH TIVF |
| SIVF | LANGUAGE 2 SIVF | MAR 62 | 42 | FONDERIE DE RUELLE | OULES, H | SEA | 1.4 | 62 | I | A | A | STANDARD | MINISTERE DE LA MARINE | TRANSFORMS SIVF TO INTERMEDIATE LANG. IVF WHICH IS THEN INTERPRETED, ORIENTED TO IBM 650 |
| GRAPH PROCESSOR | | MAR 62 | | IBM FRANCE | TABORY, R | IBM 704/709/7090 | 1 | NO | A<br>G | C | ILM<br>NX | | IBM FRANCE | RELATED BY EXTENSION FORTRAN LIST PROCESSING |
| CSI | | MAY 62 | 230 | FONDERIE DE RUELLE | ICGM BOUCHER,H | SEA | 5.1 | JUN 62 | G | A | AS | 4 BUFFERS, A.L.5 MEMORY | MINISTERE DE LA MARINE | FIXED PT. TECHNICAL COMPUTATION |
| CAMO | CAMO | 61 | | FACULTY SC | FACULTY SC | IBM 650 | 2 | NOV 61 | A | C | A | FLP, XREG | FACULTE DES SC. ORSAY | RELATED TO CAM BY EXTENSION. |
| APB | | MAY 62 | 3 | FACULTY SC | BACCHUS | BULL GAMMA ET | 3.8 | JUN 62 | G | B | A | STD SCIENTIFIC, 64 TRACKS | FACULTE DES SC STRASBOURG | RELATED TO AP2, EQUIVALENT TO SYMBOLIC LANGUAGE OF SEPSEA CAB 500 AND IBM 1620 |
| ONERA | | | | • | ONERA | BULL GAMMA | 6 | MAY 61 | G | B | ASX | 8K | ONERA | SIMULATION OF 610 ON AET BULL. ALPH. TITLES |
| SYNTOL | LE SYNTOL | JUN 62 | 120 | EURATON | CNRS | IBM 7090 | 12 | MAR 62 | I | C | | 32K, 4 TAPES | CNRS | CREATED ESPECIALLY FOR DOCUMENTATION |

# TWENTY FIVE YEARS OF FORTRAN
A National ACM Lectureship
Series Presentation

by
J.A.N. Lee
Pioneer Day Chairman
National Computer Conference 1982

Department of Computer Sceince
Virginia Tech
Blacksburg VA 24061

82/11/12

CS82010

## Abstract

In 1982 FORTRAN will have existed in the environment of computers, computing and computation for 25 years, making it one of the most successful of programming languages even if it is not the actual oldest still surviving language. The honor of being the oldest still belongs to APT (Automatic Programmed Tool.) This report is the script of talk given at several institutions during the Spring of 1982 and serves as a skeleton on which a broader history is to be developed.

## HOW IT GOT STARTED

IBM was a comparative latecomer in the electronic computer market, T.J. Watson Sr. having little confidence in the reliability of machines which one could not actually see working, and perhaps influenced in some degree by the prognosis that a small number of such machines would satisfy the world's demands for computation. While it is true that IBM had supported Howard Aiken in the development of the (Harvard) MARK I, that too was a relay machine because IBM was marketing mechanical devices; if Munroe [1] had been the sponsor then perhaps the MARK I would have been electronic. Of course under those circumstances IBM would not have been injected into the electronic computing field any earlier. At the same time, when IBM did eventually enter this field with the SSEC, the emphasis was on providing hardware and supplying the customer with programming support, at a fee, through the service bureaus. Thus to provide "software" (as we know it today) would have counter to the profit motives of those bureaus and the day had not yet dawned when programs were for sale! User cooperatives were just beginning to emerge as a means by which customers could freely exchange their in-house programs.

## Speedcoding - the floating point simulator

After the advent of the IBM 701, some of the drudgery of keeping track of the radix point in fixed numeric fields was relieved by the introduction by John Backus of the Speedcoding system. This

--------------------

[1] See foreword by Bernard Cohen to: "History of Mechanical Computing Machinery", by George C. Chase, Ann. Hist. Comp., Vol.2, No.3, July 1980, p.198.

system made the 704 appear to be a three-address floating point calculator with the added advantages of input-output conveniences. While it was an interpretive system, the reduction in costs of coding, testing and operation proved it to be a more economical means of problem solving than direct machine language programming [2].

## The IBM 704 -- built-in floating point and indexing

The design of the next machine for the IBM family of electronic calculators originally did not include any provision for hardware floating point arithmetic or indexing operations. Backus championed this cause against strong opposition which was based on the profit motives of not giving away too much of the computational pie; after all, such niceties would only decrease the time needed to complete a computation in the service bureau thus decreasing the profit to be gained. The battle was won on the side of improvements in hardware services and as a result such systems as Speedcoding were no longer viable. Backus [3] stated that "... early systems ... had hidden a lot of gross inefficiencies ... in floating point routines ... (and) clumsy treatment of looping and indexing ...", so that when the 704 came along with its hardware floating point and indexing "... there was just nowhere to hide inefficiencies."

## WHY FORTRAN?

--------------------

[2] Backus, J.W., The IBM Speedcoding System, Jour. ACM, Vol.1, No.1, January 1954, pp.4-6.
[3] Backus, J.W., FORTRAN Session, History of Programming Languages, Academic Press, New York, 1981, p.50.

Thus IBM needed to look towards means of producing programs automatically which were at least as efficient as those that would be written by hand and preferably using a language similar to that that had been proposed by Rutishauser [4] and hinted at by such authors as Glennie [5].

## Automatic Programming

At this time (1953) automatic programming was regarded as the wave of the future though in general the attempts at design and implementation were somewhat narrow in their scope or foresight. The concept of machine independence was lacking and the point of using a language which was non-machine-like seemed to have been missed. There were many sceptics who believed either that the task was impossible (see [6]) or that it was beyond the current state of the art.

## THE PAPER LANGUAGE

### December 1953 Memorandum Backus to Hurd

To propose the development of an automatic programming system which was not only more advanced in its language concepts as well as its ability to produce efficient code was the daring step which Backus proposed to his manager, Cuthbert Hurd, in December 1953. Without either a pre-authorized budget and a supporting staff or a detailed proposal, Backus was given the go-ahead for

--------------------

[4] and [5] see Knuth, D.E. and Pardo, L.T., Early developments in programming languages, in Encyclopedia of Computer Science and Technology, Dekker, New York, Vol.7, pp.419-493.

[6] Hopper, G.M., The Early Days, in The History of Programming Languages, Academic Press, New York, 1981, p.13.

this research project. Perhaps Hurd didn't really expect the project to last too long, since in those days it was unheard of to spend "man-years" on writing programs. Actually he expected the task to be completed within six months.

## The Initial Proposal 1954

A mid-1954 initial proposal suggested that the programmer "... would like to write (the mathematical formula) instead of the ... instructions (for) this expression ...".

## Backus, Ziller and Herrick

Who were these visionaries who were to succeed where no-one else had ventured? The 1957 biographies state: "John (Backus) ... joined IBM in 1950 as a programmer in the Pure Science Department working with the Selective Sequence Electronic Calculator (SSEC). He transferred to the Scientific Computing Service in 1952 (and) ... in 1954 he was appointed Manager of the Programming Research Group in Applied Science and is presently the Department Manager of the Programming Research Department. John was awarded BS and MS degrees in mathematics by Columbia University ... (his) hobbies are hi-fidelity and chess."

"Harlan (Herrick) ... was raised in Iowa ... (and) came to IBM eight years ago (1949) from Yale where he taught mathematics. He received his masters at the State University of Iowa ... is a member of Phi Beta Kappa, Sigma Xi and the Masthematics Association of America ... (his) pet peeves are materialism, dishonesty and hypocracy ..."

Regrettably no such biography exists for Irving Ziller though he joined IBM in February 1952 and was the first to join Backus on the FORTRAN project.

PRELIMINARY FORTRAN (Nov. 1954)

The preliminary proposal prepared in May 1954 does not actually mention the name FORTRAN, that being left for a Preliminary Report in November 1954. The significant elements of that report include such items as:

2 character variable names<br>
function names 3 or more characters<br>
multiply -- x , involution -- xx<br>
relative constants

where relative constants where attributes which could be ascribed to identifiers whose associated values were to be "relatively constant".

Statements

Two statements in this preliminary report are significant for both their ingenuity and foresightedness, even though neither appeared in the resulting compiler to be delivered in 1957:

```
DO 10,14,50 I = 4,20,2
```

The significance of this statement is the three statement identifiers in the prefix to the loop control information; the meaning is that the block bounded by the statements labelled 10 and 14 should be repeated until the loop control conditions are exceeded, following which the next statement to be executed is that labelled 50! Since there was no requirement that the DO statement be contiguous to the block of statements to be repeated, then this might be termed a remote loop specification. Obviously this complexity was muted by the time the first implementation was completed but it is interesting that similar statements did exist later in JOSS and COBOL (PERFORM...VARYING), and is similar in its use of a remote block of code to the BASIC GOSUB statement. It is important to remember that subprograms were not "invented" [7] in this form until two years after this preliminary design, though Mauchly had mentioned the subroutine concept in 1948 [8].

### IF (X>Y) 12,55

Most people relate the three-way arithmetic IF statement with the original FORTRAN, but surprisingly enough that statement was a replacement for the much more modern statement shown above which did not re-appear in the language until FORTRAN IV in 1961. Similarly, the use of mathematical symbols such as > and thus the implication of logical expressions was left to a later version of the language. Quite distinctly, these symbols did not exist on the standard IBM key punches of the 1950's era.

### Relabel

The Relabel statement was the beginning of a concept which has not yet been reintroduced into FORTRAN, that is array processing.

--------------------

[7] Wheeler, D.J, Wilkes, M.V., and Gill, S., The Preparation of Programs for an Electronic Digital Computer, Addison-Wesley, Reading MA, 1957.

[8] Mauchly, John W., Preparation of problems for EDVAC-type machines, Proc. Symp. on Large Scale Digital Calculating Machinery, 1947 January 7-10, reprinted in Randell, B., (Ed.), The Origins of Digital Computers, Springer-Verlag, New York NY, 1982 (Third Edition), pp.393-397.

The intention of this statement was to permit the programmer to relabel the rows of a matrix so as to rearrange the row ordering and thus apply a single algorithm to a particular slice of that array. The complexity of this operation was not realized at the time of the preliminary report and it is clear that indexing through arrays explicitly is a much better programming technique than an implicit set of possibly unreadable instructions.

### Frequency

When considering that one of the major objectives of the research project was to prove that  machine could generate code on a par with a human programmer, the inclusion of information on the expected frequency of execution of statements is a logical necessity. However, it was to be found that such statements were unnecessary since logical flow analysis could provide the same (if not better) information.

## FUTURE FORTRAN (from a 1954 perspective)

The preliminary report was not bashful in suggesting that the language might one day be extended to include new facilities.

### begin-end

While the bracketing of a block of code by the reserved words begin and end is usually associated historically with ALGOL, this preliminary report included these terms as scoping delimiters for different types of arithmetic to be performed.

### complex, double and matrix

It was proposed that in subsequent versions of FORTRAN it would be possible to prescribe the type of arithmetic to be performed in certain segments of the program. It is not stated whether these operations would be associated with new data types or whether the first character in a name would signify different types than previously used.

## CHANGING ENVIRONMENTS

From the beginning of the development period, the Programming Research group were shuffled through various locations in the area of 590 Madison Avenue, there perhaps being a correlation between the quality of the facilities and the recognition given to this project by the IBM administration.

### Administrative attitudes

Chess in the afternoon or punch the time-clock?

Hal Stern [9] remembers clearly a steadily changing atmosphere through the period, beginning with a very "researchy" environment when people worked as hard as any other time but where the actual time of day was irrelevent. To him it was not unusual to work hard for a period, to play a game of chess and then to return, refreshed, to the task at hand. Thus when, apparently under pressure from upper level management, Backus broke up a chess game between Stern and Peter Sheridan, the former was somewhat incensed; as he remembers it was the first time he was winning!

### Changing locations and the outlook

Backus believed that the changing locations distinctly affected

--------------------

[9] Personal Correspondence

the productivity of the group, and there seemed to be a correlation with the view from the windows [10]:

> "... We shifted around from one small building (to another) at fairly regular intervals and this seemed to affect our work habits in a strange way. ... The first one we moved to overlooked the dressing rooms of the J. Thorpe department store and then we moved to another overlooking the dressing rooms of Bonwit Teller. ... (This was) a period in which our productivity seemed to decline considerably. From there we moved to a building on 56th street ... I noticed that when I came in everybody was there and apparently had been there for some time. (Eventually) someone confided to me that across the street ... was a young lady ... who slept without any clothes on ... and (who) danced very exuberantly ... before going to work. This was a period of great productivity because everybody came in early and after the show was over settled down to work long before (the official) starting time."

## THE VON NEUMANN CONSTANT

Cuthbert Hurd had originally expected that the FORTRAN project would have been completed within six months of its starting date. But he did not realize that the von Neumann constant applied to the project. This constant is defined as being the time to completion of a project from the instant the enquiry is made

--------------------

[10] From the Transcript of the Anecdotes told by numerous programming language pioneers at the History of Programming Languages Conference, Los Angeles CA, 1978, unpublished.

regarding the completion of the project and is a constant (usually about six months).

Missed delivery dates and changing machine requirements

An examination of successive SHARE meeting minutes starting in 1955 reveals that FORTRAN was to be delivered by the time of the next SHARE meeting (six months away) and that to meet this objective a little more machine was needed each time. When finally delivered the minimal machine configuration was 4k (IBM 704) words of core memory, 1 drum unit and 4 magnetic tape drives.

VON NEUMANN and FORTRAN

During the early years of the 1950's, von Neumann was hired by Cuthbert Hurd as a 30-days per year consultant mainly to assist with the design of the 704 and later to consult on mathematical problems. Apparently the technique was to establish von Neumann in an office in the World Headquarters and then to have those with problems bring them to him for consideration. Typical of the quickness of mind of von Neumann is the report of one of these consultations which involved John Greenstadt [11].

> "I did some hand calculations ... and it converged in the few cases I tried ... (so) I tried to prove convergence ... but with no hint of success ... Finally in the latter part of 1953, we decided to ask von Neumann (for help) ... I explained it to him in two minutes ... He spent the next

--------------------

[11] Goldstine, H.H., Footnote to a recent paper, Jour. ACM, Vol.7, No.1, January 1960.

> fifteen minutes thinking up all the approaches we had thought of in three or four months, plus a few new ones ... at this point he decided it was a non-trivial problem and perhaps not worth it anyway ... and immediately suggested ... the truly natural generalization of (the) method."

1954 Meeting -- Von Neumann, Backus, Hurd and Beckman

Thus it was natural for Hurd to suggest that von Neumann's opinion of FORTRAN might be worthwhile -- after all it would cost no more than a day's presentation! Besides Backus and Hurd, they were joined by Frank Beckman, then manager of "Pure Programming". Beckman reports the conversation in his book [12] and in personal correspondence:

> "I do not know if von Neumann expressed any opinion about FORTRAN outside of this meeting, but I would certainly not describe his reaction at the time as being unduly negative -- somewhat apathetic perhaps, but not strongly negative. I remember very vividly his allusion to Turing's "short code" ... In general von Neumann was not an enthusiastic of automated programming aids ... I have always felt that since he, himself, did not require such aids in writing programs, he could not empathize with the typical production programmer."

In effect, von Neumann's response to the presentation on FORTRAN was "Why would you want another language?"

--------------------

[12] Beckman, F.S., Mathematical Foundations of Programming, Addison-Wesley, Reading MA, 1980, pp.177-178.

Perhaps it is ironic that some twenty years later Backus chose for the title of his Turing Lecture "Can Programming be Liberated from the von Neumann Style?" [13].

1957

Eventually, the year arrived when FORTRAN was once more due to be delivered, but now the working system refused to be duplicated. The problem was that it was the practice to distribute programs through SHARE as card decks but this was probably the largest program ever intended to be distributed up to that time, and the card punches refused to remain stable enough to complete the punching of a single deck. Thus it was decided to distribute the compiler on magnetic tape instead.

Actual delivery

No record exists as to when the first compiler was actually shipped intentionally; perhaps part of the problem was that the US Post Office closed down a part of its operations about the time that it should have been shipped and was waiting for more money from Congress before they adjourned for Easter!

The way it was that week

As we can show later, it would appear that the first deck was shipped during the week of April 15-20, 1957. That was a week during which many things were reported in the New York Times:

-- Britain and Egypt were at war over the Suez Canal

--------------------

[13] Backus, J., Can Programming Be Liberated from the von Neumann Style?, CACM, Vol.21, No.8, August 1978, pp.613-641.

-- Nasser of Egypt instigated a coup against King Houssein of Jordan but failed

-- Ike was President and Dick was his VP

-- the Yankees beat the Dodgers 5-1 in the Sunday game -- the Brooklyn Dodgers that is

-- Studebaker-Packard announced that they intended to offer a new small car to sell at less than $2000 and for the first time it would offer as standard features items which had only been optional before -- a heater, a defroster and directional signals!

--The Chrysler Corporation announced first quarter earnings of $1,100,000,000

-- the Canadiens beat the Bruins for the Stanley Cup again

-- the Dodgers announced plans for a new stadium in Flushing Meadows

-- Dean Martin's guest on his first TV show was Bing Crosby

-- Desilu Productions were confident that "I Love Lucy" would be back next year

-- Poland was warned against Western aid by Krushchev

-- NBC was planning to put TV shows on tape instead of running them "live"

but the delivery of the first FORTRAN compiler and the running of the first program escaped unnoticed. Perhaps the week itself is not crowned with glory -- it was the same time in 1912 (45 years previously on April 15th) that the Titanic sank!

## INNOVATIONS

It is difficult to pick out any single item which makes FORTRAN unique by itself; almost everything that was delivered in that first package was innovative either because it was the first time that the feature had been placed at the disposal of a user of a high level language, or because in conjunction with other features of the language or the compiler it was an outstanding contribution to the science of computation.

FORMAT

The concept of being able to specify the format of an input or output item was not new; in fact the FORTRAN implementer of FORMAT, Roy Nutt, had previously included a similar system in an internal system for United Aircraft of Hartford CT.

Optimization techniques

In some respects, the development of language was incidental to the research to prove that a machine could produce good every bit as good as that produced by a human programmer. Backus [2] states that "... the degree of optimization they achieved was not really equalled again in subsequent compilers until the mid-1960's when the work of Fran Allen and John Cocke began to be used ..." This optimization was so good in fact that the "proprietors" of the optimization section (Irv Ziller and Bob Nelson) often thought that the results were wrong the generated code being unrecognizable as having originated in the code they input! Regrettably, it was not the practice to document as one went along in this age of compiler development and thus it was not until some years later that any of their techniques were published [14].

### User's manuals

While not significiant to either the design of the language or the implementation of the system, the introduction of a readable user's manual and a programmer's primer was clearly significant to the ultimate success of the language. The fact that the original user's manual described the whole of FORTRAN, with examples in less than 50 pages while at the same time providing wide margins for the keeping of notes by the reader, is both remarkable and a feat which has not been repeated again. Interestingly enough, the von Neumann constant apparently also relates the publication of the Programmer's Reference Manual and the actual delivery of the compiler -- the manual is dated October 15, 1956 and the compiler was released on April 15, 1957.

## THE FIRST PROGRAM

### The first error message

The running of the first program, though well documented by Herb Bright [15], was not a planned activity. There is one error in Bright's report which needs correction -- 1957 April 20 was a Saturday not a Friday as reported. It was on the afternoon of that Friday when an unmarked deck of cards was delivered to Westinghouse-Bettis and which was assumed by Lew Ondis to be the

--------------------

[14] Lowry, E., and Medlock, C.W., Object Code Optimization, CACM, Vol.12, No.1, January 1969, pp.13-22.

[15] Bright, H., FORTRAN comes to Westinghouse-Bettis, 1957, Computers and Automation, November 1971, pp.17-18.

right size to be a FORTRAN compiler. Jim Callaghan quickly wrote a small program based on a recent technical report by Ollie Swift, and using the "common" technique for running programs on the IBM 704, the unmarked deck was loaded into the syetm followed by the program. Surprisingly it worked and in a short time the first FORTRAN error message was output:

```
              FORTRAN DIAGNOSTIC PROGRAM RESULTS
                             ---
      05065 SOURCE PROGRAM ERROR. THIS IS A TYPE-GO TO ( ),I
       BUT THE RIGHT PARENTHESIS IS NOT FOLLOWED BY A COMMA
                             ---
               END OF DIAGNOSTIC PROGRAM RESULTS
```

The error was quickly fixed and the program (apparently) recompiled and executed to produce "... a whiff of computing followed by 28 pages of output ..."

### IMPROVING THE CODE

One of the flavors of computing in those days was the belief that almost anyone could produce code better than IBM could. Backus [2] quotes Perlis as wondering "... why those clods working on FORTRAN had taken 25 man years to produce a compiler, since one of his graduate students had written an IT compiler in a single summer ...!" Perhaps Perlis did not understand the complexities of code optimization.

### Frank Engel -- Westinghouse-Pittsburgh

Thus when Engel noticed that during the compilation process there was never any instant when two tapes were in use on the 704, he asked his account representative (Ken Powell) for a copy of the source code. Powell relayed the request to his manager, Frank Beckman, who responded "IBM does not supply source code." Not to be outdone, Engel dumped the compiler code (in octal), spotted the section which was responsible for this tape management, rewrote it (in octal) and produced a system which had an improved throughput of about 2 to 3 times the speed. When Powell saw this he asked Engel for a copy which he could send back to the FORTRAN group in New York -- Engel responded "Westinghouse does not supply object code".

## THE PEOPLE OF FORTRAN

A complete dossier on all the members of the FORTRAN team (John Backus, Harlan Herrick, Irving Ziller, Robert Nelson, Roy Nutt, Peter Sheridan, Lois B. Mitchell Haibt, Sheldon Best, Richard Goldberg, David Sayre and Grace (Libby) Mitchell) is not possible here. Their individual contributions to the implementation have been documented by Backus [2]. But what of the individuals?

### Backus, Nutt, Haibt

Backus has maintained his leadership in the field of computing through the years and is somewhat frustrated that the success of FORTRAN both overshadows and perhaps even thwarts his efforts to improve programming languages [13]. He has been recognized by the IEEE in 1967 (on the tenth anniversary of FORTRAN) with the W. Wallace McDowell Award, by the United States of America in 1976 (the bi-centennial year) with the National Medal of Science, and finally by ACM in 1977 (on the twentieth anniversary of FORTRAN) with the Turing Award. Having completed the FORTRAN implementation, Backus was appointed by John Carr, ACM President, to be a member of the joint ACM-GAMM group which developed ALGOL. A close examination of the ALGOL proposal and the FORTRAN preliminary report reveals that perhaps it was Backus who introduced certain salient language features, such as begin-end, which were not included in FORTRAN. Backus, with the editorship of Peter Naur, also invented the syntactic definition schema now universally known as BNF, and variously as Backus-Normal-Form or Backus-Naur-Form.

Roy Nutt was not a member of the IBM staff which was assembled to develop FORTRAN, but instead was a highly knowledgable user whose concepts on input-output, and especially FORMAT, could not be duplicated elsewhere. Thus with the support of Walter Ramshaw, his manager at United Aircraft in Connecticut, Nutt spent a few days each week in New York assisting the IBM team. Like Backus, Nutt has remained agile in the field of programming languages and was recently embroiled deeply in the Ada Programming Language controversy [16].

Lois Haibt, while a contributing member of the team, is probably notable for being one of the few computer scientists ever to be featured in the Madamoiselle Magazine [17]:

"This twenty-two-year-old girl started at IBM with a salary

--------------------

[16] See ACM Forum, CACM, Vol.24, No.11, November 1981, p.784 and succeeding issues.

[17] Kirkbride, K., and Garland, K., Machine, What do you think?, Madamoiselle, October 1958, pp.92-157.

of five-thousand dollars a year and increased her income to six thousand in eight months ... Lois spends a good part of her day at a large bare desk writing up instructions for the computer to follow ... (she) waits her turn at the machine in a glass enclosed, red-walled balcony above it ... The girl who keeps the computer's social calendar tells Lois to stand by, ready, so that when the person before her is finished Lois can step up for her "date" immediately."

## THE LANGUAGE MANUAL AND THE PRIMER

As stated previously one of the primary innovations which accompanied the FORTRAN implementation was the user's manuals which the group provided. These manuals are significant both from their conciseness and clarity, but also from their form. Apart from the PRINT 1 manuals which were published the same year, and the FOR TRANSIT manual of 1957, they are unique in the cover design (the team voted democratically on it) and for the fact that all these manuals contain the names of the authors. Like FORTRAN itself, these manuals were so good that they inhibited other publications on the topic and it was not until four years later that any form of competitive publication was available.

## FOR TRANSIT

### If we can do it once, we can do it again

In 1982 it is common sense that once one has produced the first version of a product the second version has got to be better. In fact, one of the proverbs of programming [18] states:

Don't Be Afraid to Start Over

So when IBM introduced the 650 computer in the midst of the fever of user's anticipations regarding FORTRAN, it was obvious that another implementation was needed.

## Novice crew

The 704 team led by Backus was still very busy in early 1957 completing the final stages of debugging and trying to get the system punched ready for distribution. Thus no-one could be spared from that group to start a new project, but there were people who had been close enough to the activity who could parallel their work. One of those was Bob Bemer. He assembled a 650 team consisting of Otto Alexander and David Hemmes at the Langdon Hotel on 56th Street, neither of whom had any previous experience with the kind of work to be undertaken; later they were joined by Flo Pessin who was equally unprepared for the task.

## Cascading Implementation

Bemer noted that there was another significant activity in progress which was to implemented on the 650, though it was not originally intended for that machine; it was the IT (Interpretive Translator) system being developed by Alan Perlis at Carnegie Institute. IT compiled a much simpler language into the assembly language of the IBM 650 (SOAP - Symbolic Optimized Assembly

[18] Ledgard, H.F., Programming Proverbs for FORTRAN Programmers, Hayden Book Company, Rochelle Park NJ, 1975

Program) which in turn was assembled into the object code for the machine. Thus the concept was developed to "cascade" the implementation from FORTRAN to IT to SOAP and hence to object code in four passes.

FORTRAN > IT > SOAP > OBJECT

Quick implementation

In spite of the inexperience of the crew, a version of FORTRAN was available for the 650 only a few months after the delivery of the 704 version and the expenditure of only 4-5 man-years of work. To accomplish this the language was a subset of that implemented for the 704, but this was consistent with the fact that the 650 was an even smaller machine than the 704. Part of the motivation for this effort was the fact that IBM expected many more 650's were expected to be installed in Universities than was the case for 704's and especially since IBM was now willing to offer a 60% educational discount to those institutions which used the systems for administration, scientific computation and business data processing classes. FORTRAN was to be the "hooker" of this new generation of students.

To accelerate the implementation, and partially in recognition of the success of the language for its own right rather than simply as a test bed for optimization research, optimization was omitted from this new implementation. A form of optimization existed in the SOAP system, but this was not language dependent and was merely concerned with the location of instructions on the drum (the main store of the 650) in relation to the pertinent data and/or the next instruction.

Bemer and Pessin

Bemer joined IBM Programming Research in 1955 after a career in the aircraft industry and was appointed manager of Programming Systems in 1960. In the "FORTRAN years" Bemer was active in many other ways which furthered the development of programming languages. He created the first load-and-go compiler for PRINT 1, developed the language known as COMTRAN (Commercial Translator*) and actively supported the development of ALGOL. In fact, in 1960 he was quoted as stating to the British Computer Society that "... we wish to obsolete FORTRAN and scrap it, not perpetuate it. Its purpose has been served."

Flo Pessin was given the task, by Bemer, of writing the arithmetic scanning routines for this new version of FORTRAN, but first she invented the name of the system -- FOR TRANSIT. Based on the cascading approach that Bemer had suggested, recognizing the contribution of IT and being a double-crostic addict, she coined the name as a three-way pun. One of the difficulties facing Pessin at this time was both her lack of experience in preparation for this task, the fact that the 704 team had created no documentation (though they were no different than most other implementers) and there was no help offered by the 704 group. Thus she was forced to invent new techniques of compilation, and

--------------------

* It is interesting that the name is reminiscent of the source of FORTRAN -- Formula Translator.

like the others omitted to document them because she really did not know that what she was doing was so innovative.

INNOVATIONS

The problem of how to analyze and then generate code from an arithmetic expression was solved in a highly ingeneous manner. Pessin recognized (probably from high school) that the order in which operations were to be executed in an arithmetic expression was determined by the hierarchial order of the operators. Further the order of execution of fully parenthesized expressions is determined by the depth of parenthesizing. Thus the technique developed was to introduce into the expressions additional parentheses surrounding the operators, but facing outwards away from the operators, such that the number of parentheses added was in inverse proportion to the hierarchy of the operator. Sufficient additional parentheses were made available at the ends of the expression to satisfy the parenthesis balancing requirements. Thus given the expression:

a + b * c

the first stage of parenthesizing would produce:

(...( a )))+((( b ))*(( c )...)

which after cancellation of parentheses surrounding the operands develops the expression:

( a + ( b * c ))

which is correct. Obviously this technique does not take into account left (or right) associativity, but that is unimportant once this stage has been reached. The next innovation which was introduced was the use of a tabular method of expression analysis which was later rediscovered by others [19].

FIRST FILM

The first film, known to us, was produced by the New York Education Center (of IBM?) and is interesting for several reasons.

Not only is it interesting from a content point of view but also visually and editorially. The commentator is architypical of the "IBM salesman" of the era -- grey suit, white shirt, dark tie and the neatest of haircuts. The presentation is similarly in a style which we associate with both the corporation and the era; looking straight out, unsmiling and sincere. The first sentence is perhaps a commentary on where it was thought the concept of programming languages was heading:

> "... FORTRAN represents the most advanced coding system available today and is a forerunner of a universal coding language toward which we are working ..."

John Backus would be very interested in that statement since he is still working towards that goal. A later statement regarding the effort and cost of the development process is the first which we have recorded:

--------------------

[19] See for example: Samelson, K., and Bauer, F.L., Sequential Formula Translation, in Programming Systems and Languages, Rosen, S. (Ed), McGraw-Hill, New York, 1967, pp.206-220, originally published in German in 1959.

> "... was developed ... at a cost of $475,000 and ... 29 man years ..."

Since the film refers both to FORTRAN for the 704 and FOR TRANSIT for the 650, perhaps this breaks down to 25 man-years for the Backus project and 4 man-years for Bemer's activity.

Another interesting aspect of this film is that it was obviously "shot" at one sitting (or standing), and no attempts were made to edit any abberations from the presentation. Thus a false start at presenting a problem for solution is somewhat amusing:

> "... The Indians bought Manhattan island at a cost of $24. [Pregant Pause] Pardon me, the Indians sold Manhattan Island at a cost of $24 ..."

The development of a program, its punching onto cards and the compilation process are much as one would expect today in a batch environment. The film shows some shots of the IBM 704 flashing its lights during compilation and the output coming out of the printer at 100 lines per minute!

## FIRST TEXTBOOKS

While there were a few textbooks that contained a chapter on programming languages [20] the production of a single topic textbook on FORTRAN was perhaps inhibited by the excellence of the user's manuals produced by the Backus group. In early 1961, Elliott Organick, then at the University of Houston produced an "internal" booklet on FORTRAN which was marketed through the

--------------------

[20] See for example: Andree, R.V., Programming the IBM 650 ..., Henry Holt and Co., New York, 1958, Ch.8, "Compilers".

university bookstore for local use. This was accompanied later in the same year by another volume of drill exercises. However by this time Daniel McCracken had overcome the opposition from a commercial publisher and produced the first textbook solely devoted to FORTRAN [21]. Surprisingly enough this volume was not much larger than the original user's manual and maintained the wide margins and clear text of that earlier IBM manual. The review in Computing Reviews was not encouraging:

> (Computing Reviews, Vol.3, No.1, Rev. 1421, 1962 January, p. 22) states: "There are versions of FORTRAN for the IBM 650, 1620, 704, 709, 7090, and for the Honeywell 800, the Philco ALTAC, and the Control Data 1604. Since each version has its own description this latest work might seem redundant but it does have some definite advantages."

It is interesting to note that five years later Computing Reviews refrained from soliciting formal reviews of FORTRAN texts due to their "proliferation" and resorted instead to merely publishing an extract from the author's introduction!

## LOAD AND GO SYSTEMS

Although Bemer had invented load-and-go systems for the PRINT 1 language, their emergence as a "standard" implementation of a programming language did not occur until the early 1960's.

## IBM 1620

The IBM 1620 was the first (IBM) machine which provided the user

--------------------

[21] McCracken, D.D., A Guide to FORTRAN Programming, John Wiley and Sons, New York NY, 1961, 88pp.

with a truly interactive capability and a machine language which was much better human engineered than its competitors such as the (Royal McBee) LGP-30 or the (Bendix) G-15. Thus users could benefit from the concepts of compiling programs and immediately executing those programs from memory without resorting to the production of an object deck or the bother of reloading the compiled program.

## FORGO -- University of Wisconsin

Part of the impetus for this movement was again the idea that "anything IBM can do, a user can do better." Thus to save time in an open shop environment with a multitude of engineering students desiring to compile and run programs, Charles Davidson at the University of Wisconsin implemented a FORTRAN II in this load-and-go environment named FORGO. It was the first of many other similar systems for the 1620 which included systems from the Air Force Institute of Technology (AFIT FORTRAN by Richard Pratt), UT FORTRAN (from the University of Toronto) and KINGSTRAN (developed by a joint team from the University of Toronto, Dupont of Canada, Ltd., and Queen's University at Kingston). This lineage eventually led to the development of WATFOR for the IBM 360 and the plethora of similar systems developed by universities for FORTRAN and other languages.

## WHAT IS WRONG WITH FORTRAN?

FORTRAN is regularly criticized, along with COBOL and other languages of the same era, as being a dinosaur that will not die. While it is true that much of the world's scientific programming is still being accomplished in FORTRAN, that may be due to the use of the language by other than computer scientists and the lack of high level programming language education in engineering and science schools.

LACK OF STRONG TYPING

One of the criticisms of FORTRAN is the lack of strong typing, which is interpreted as the lack of a requirement that every variable be included in a type-declaration statement.

FORTRAN introduced the concept of name-type relationships

In fact, FORTRAN in its original form was the first to introduce typing based on the syntactic characteristics of the name of an identifier; the problem was (is) that the programmer may not always be as aware of this associativity as is required by the language. If there be an error, the blame should not be heaped on the shoulders of the language originators; it was only in later versions under pressure from users that TYPE statements were added and thus the name-type relationship became a "default" association which is now so much decried.

LACK OF STRUCTURED PROGRAMMING SUPPORT

To attempt to introduce into a language, which reflected the programming habits and practices of the 1950's, the desires and demands of the 1970's is a project which is almost doomed to failure before it is started. The design of the 1977 FORTRAN merely took the style of the language and used that as clothing for another concept while at the same time including all the old features so as to maintain upward compatability.

## COMMON/EQUIVALENCE

If there be any "Sins of Programming" the foremost two must be:

Thou shalt not use global variables, and

Thou shalt not use false names.

FORTRAN (IV admittedly) introduced both of these concepts in response to the demands of the time. Programmers wanted to use COMMON and EQUIVALENCE and implementers who wanted their systems to be used introduced them into their language. Following the first stage of FORTRAN development (say up to 1960) it was natural for a language implementer to provide any additional language features he could provided that the cost was minimal. Thus very early in this process of evolution, the subscript restrictions introduced by Backus et al in order to minimize the problems associated with optimization and to force the development of FORTRAN programs which would be optimizable to a degree which was not expected of a programmer, were relaxed and any meaningful expression became acceptable. Even to the point where expressions were meaningless (such as where the result would be a real number) a default conversion procedure was introduced. Thus FORTRAN became:

"... a collection of warts held together by bits of Syntax."

## Extensibility -- has reached its limit

FORTRAN has now been extended to a point where it is doubtful whether the originators can recognize it. Of course one can say the same thing about children and their growth process, it is the environment which influences both mental and physical growth;

FORTRAN has now been extended to the point where it is now more like some other language than its original self.

### Modifiability -- must still be FORTRAN

To modify FORTRAN any more will mean that the result is no longer FORTRAN; perhaps that is what the 3x3 committee of IBM and SHARE recognized in 1964 when they decided to design a new language rather than introduce FORTRAN VI. And they called it NPL, MPL, MPPL, ..., PL/I.

### EASY TO HAVE MEANINGFUL ERRORS

Perhaps one of the difficulties with FORTRAN, and one which could not have been anticipated by its designers, and which is not well understood today even though we try to do something about it, is the problem of being able to develop programs which look correct but which due to some very small abberation are semantically wrong while being syntactically correct.

### Venus Probe Problem

Perhaps the most famous example of this in FORTRAN is that pertaining to the first American probe sent to Venus. The probe was lost due to a program fault caused [22] by the inadvertent substitution of a statement of the form

```
DO 3 I = 1.3
```

for one of the form

```
DO 3 I = 1,3
```

--------------------

[22] Horning, J., A Note on Program Reliability, ACM SIGSOFT, Software Engineering Notes, Vol.4, No.4, 1979 Oct., p.6.

which went undetected throughout the "career" of the probe.

## POPULARITY

One of the major factors in keeping FORTRAN alive must be its immense popularity outside the computer community; that is, amongst users whose primary vocation is not computing.

### 1976-77 Hamblin Survey

A survey of institutions of Higher Education showed that 71% were using FORTRAN (i.e. had it on the system) as contrasted with 59% with COBOL, 55% with BASIC and a lowly 9% with Pascal. Since the survey was taken in 1977 then this latter figure may have changed significantly.

### 1980 GUIDE Questionnaire

A slightly more recent survey, though amongst a group who one would expect to have less interest in a scientific language, provides the following data points:

> Amongst programmers (in GUIDE User's Group installations) who are full time involved in programming, 81% were using COBOL primarily with only 6% being devoted to FORTRAN.
>
> Conversely, amongst casual programmers, COBOL only commanded 23% of their usage while FORTRAN had climbed to 18%.

## HOW LONG WILL IT LAST?

Attempting to judge the longevity of any programming language is likely to be a slightly fruitless occupation, except in the case of a well established language such as FORTRAN. The ANSI committee responsible for FORTRAN has decided that the next version of the standard (the last was published in 1978 and refers to the language which is generally known as FORTRAN 77) will contain list of items to be deleted. The purpose of this list will be to warn programmers against using certain language features which are now considered to be obsolete or unappropriate in today's programming environment. If the standard is published in the same period as the previous two, then this list should be available in 1988 and the succeeding standard (which will not contain those items) will be ready in 1999. Thus FORTRAN will exist in basically its current form until the end of the twentieth century. Perhaps Tony Hoare [23] expressed this longevity best:

> "I don't know what the language of the year 2000 will look like, but I know it will be called FORTRAN."

--------------------

[23] Personal Conversation following the Turing Lecture at the Annual ACM Conference in Nshville TN, 1980 Nov.

Interview of Florence Pessin IBM Santa Teresa, June 24, 1981
by J.A.N. Lee

with annotations by Otto Alexander, April 16, 1982

Lee: Let me start off by asking what was your earliest involvement in FORTRAN.

Pessin: I joined IBM in February 1957.

Lee: Just before FORTRAN was released?

Pessin: Yes. I reported to Bob Bemer who reported to Backus. Backus had two things going: Fortran was in the last stages of development and then he had Bemer's activity, which he didn't take much interest in.

Lee: Was that the Applied Programming Group?

Pessin: At that time it was a group that reported into a corporate organization. Backus reported to [John] McPherson. Applied Programming was organized later that year.

Lee: The reason I asked is that some of the files I have seen talked about Bob Bemer as not being responsible for language design but only for language control.

Pessin: He did not have anything to do with FORTRAN design. However, when I came on board there were two people, Otto Alexander and Dave Hemmes. They were working on the thing that became FORTRANSIT, which was Bemer's idea.

Lee: That was pre-FORTRANSIT?

Pessin: Right, it didn't have the FORTRANSIT name but that was the activity.

Lee: Was it aimed at the 650?

Pessin: Yes, that was Bemer's thought. Since Perlis and company were working on IT, and Backus and company had developed the FORTRAN language and there seemed to be some similarities between the two.

Lee: The aim was to provide similar facilities on the 650?

Pessin: Right; the idea was that one could save a certain amount of processor building time if one simply translated from language to language. I will tell you in a few minutes that we demonstrated that that was false theory; However, that was the thought at the time; that was the start of the concept of cascading which turns out to be quite wasteful.

Alexander and Hemmes started working on the translator from FORTRAN to IT in late 1956.[*] When I came on board, the whole arithmetic area had not been touched. So I got the section, and, in my ignorance, I didn't know enough to know that it was nigh impossible to do. One of the big problems we had was that both the FORTRAN language and the IT language were in a state of flux. FORTRAN was really not fully defined and was still being tinkered with. And IT was changing on what seemed like a daily basis as they found things hard to implement. It was the very pragmatic approach they took.

Pessin: Well, it was indeed completed by summer 1957 but from week to week, as we were trying to implement we found that although in its broad outline it had been fine, but where the commas go, and where the parentheses go, those kind of things were changing. From a conceptual point of view, the important thing about FORTRANSIT is that it was the first implementation that moved in the direction of making FORTRAN machine independent.

Lee: When you worked from John's original FORTRAN, what we now call FORTRAN 0, what were the restrictions put on you for the 650?

Pessin: The original FORTRAN had 6 character names, that is up to 6 characters in a name that was a reflection of the 704 because the 704 had 36 bit words. The 650 had 10-digit or 5-character words, so we were restricted to up to 5 character names. Because of the size of the machine (2000 words), we were forced to restrict subscripting to two subscripts and then, because IT accepted only one subscript which could be any expression, what we had to do was to take the two subscripts and linearize them. The result was to come up with a kind of arithmetic expression that FORTRAN itself would not have accepted.

The hierarchy of operations was kept intact but the problem was that IT scanned a line from right to left and did operations in the order it found them. So the arithmetic section in FORTRANSIT had to go through and analyze the

--------------------

[* Alexander]: The savings planned inlcuded I/O routines, function subroutines and miscellaneous FORTRAN language subroutines which were available from IT. Their value had to be considered since they were available.

[1] Backus, J., "FORTRAN I, II and III", in Wexelblat, R., History of Programming Languages, Academic Press, 1981.

hierarchy of operations and insert parentheses to force IT to give the same result that FORTRAN would have given.[*]

Lee: Wasn't it FORTRANSIT that had that very neat little algorithm for putting in parentheses in [inverse] proportion to the hierarchy of the operator?[*]

Pessin: Yes.

Lee: John mentioned that technique to me many years ago but I never knew where it came from.

Pessin: Yes, I did that. If I had understood the problem, it would have scared me off; but since I didn't understand the problem, I went ahead and did it.

Lee: Had Samelson and Bauer [2] published their paper on tabular scanning techniques by then?

Pessin: I have no idea. I never read that paper. You have to understand that when I came to IBM I had only programmed the 650 in machine language. I didn't know anything about SOAP, I didn't know anything about compilers. I didn't know anything! I just tackled this as a logical task rather than learning anything about the state of the art.

Lee: When you talk to John about FORTRAN, he emphasizes over and over again that he didn't set out to invent a language, he set out to prove optimization techniques could produce programs just as good as by hand. Because you were going from FORTRAN to IT, to SOAP, did you take care of optimization or did you leave that to SOAP?

Pessin: The only optimization we did was simply the machine optimization in SOAP; we did not attempt to optimize the program in terms of common subexpressions or constant expressions. That was a level of sophistication we couldn't begin to achieve.

Lee: SOAP, on the other hand, optimized the placement of instructions and data on the drum, which optimized the access

--------------------

[* Alexander]: Input/Output was in a fixed format in IT.
[* Alexander]: Flo first implemented the arithmetic section with the responsibility (of parenthesizing) on the FORTRAN Programmer. Bemer raised (...) hell and she proceeded to make the processor insert the parentheses; it was quite and achievement.
[2] Samelson, K., and Bauer, F.L., Sequential Formula Translation, CACM, Vol.3, No.2, 1960 Feb., pp.76-83. Obviously I was wrong but there surely were other techniques known in 1957.

time. So FORTRANSIT used none of the techniques which were used in the original FORTRAN for the 704?

Pessin: No, there was no way to do that using the [cascading] system we used.

Lee: You didn't even have a magnetic tape on that machine, did you?

Pessin: No, no magnetic tape; it was a card machine at that time. When tapes were hung on the 650 a couple of years later, FORTRANSIT was adapted to tape [*] to avoid all the card handling. The output was stashed on tape and retrieved on the next pass, so you weren't aware of the three passes that you went through on a card machine.

Lee: Now one of the things that seems to be missing with regard to FORTRANSIT, and I asked Bob [Bemer] about this the other day, there were no published documents on FORTRANSIT, no technical reports, [no] papers published in CACM.

Pessin: No, the only thing we put out was the user's manual.

Lee: Wasn't that just a "dittoed" or purple?

Pessin: No, it was printed.

Lee: But there were no technical documents?

Pessin: No. I don't think we understood - the three of us - until very, very much later that we were ploughing new ground. We were set a logical task and we did it. And we really didn't understand about compilers. I had no idea that compilers were so new, that what we were doing was something quite new and radical, that we had discovered stuff so literally nothing got written down.

There were other things we were responsible for. The 650 was a numeric machine with a device you purchased to make it alphanumeric. We wired up a board so that you didn't have to have that device in order to use FORTRANSIT.[*] It was a very complicated piece of wiring. We were later taken to task by the sales team who reminded us that IBM was selling those devices. But here again, it was a logical problem which we were solving. How could FORTRANSIT possibly succeed if it required a special feature to be purchased? We got into a little bit of trouble when the MOD 2 machine was put

--------------------

[* Alexander]: Mike Starr did this work using the 650 tape system at Glendale Lab, Endicott.
[* Alexander]: The wiring diagrams are in the ("dittoed") manual. Dave Hemmes did this work.

out because the timing was different and we had to issue a new wiring diagram. But eventually we did it--we were young and eager!

Lee: And you didn't know any better!

Pessin: That's right. We just did it.

Lee: Were there any proposals later to transfer it [FORTRANSIT] to any other machine?

Pessin: Well, two things happened. One was that FORTRAN became very popular.

Lee: The 650 became popular.

Pessin: Yes, after we released it [FORTRANSIT] a programmer in what was at that time called Math and Applications named Lin Wu, (I have no idea what has become of him) [*] and I became somewhat disturbed by all the card handling. We looked at the problem and decided that a translator was a translator and it didn't really matter what your source language was or what your target language was: the principle was the same. Therefore, we thought we could cut out the FORTRANSIT to IT pass and go directly from FORTRAN to SOAP. We didn't want to get involved in rewriting the assembler, so we decided to go from FORTRAN to SOAP. At that time, I was already in Applied Programming still working for Bemer, and had the opportunity to pursue that sort of thing. Meanwhile FORTRANSIT itself had transferred to the 7070 development group [*]; they were the ones who put it on the tape machine.

Lee: Have you any idea of the date of release of the original FORTRANSIT?

Pessin: It was probably sometime in '57.

Lee: So not long after the original FORTRAN?

Pessin: Probably late summer or early fall. AUGUST

Lee: The original FORTRAN was delivered in April [1957]; this was just a few months later. RAN JUNE

Pessin: Sure, we didn't have to invent anything.

--------------------

[* Alexander]: I believe it is Lynn Woo and he is at Yorktown.

[* Alexander]: at 425 Park Avenue. Flo and Dave went with Bemer. I went with Liggett (7070 Programming Manager) as Project Leader for 7070 FORTRAN and maintenance of the 650 FOR TRANSIT.

Lee: The point is, you weren't far behind writing something similar for the 650. So the timing for the compiler going in one pass from FORTRAN to SOAP was maybe a year later? 59 JUNE

Pessin: It was probably sometime in '58 and was called 650 FORTRAN. I don't think it ever became as popular as FORTRANSIT simply because FORTRANSIT was there, it was [already] on the scene.[*]

Lee: It was established. Even though there was no change in the language. [For 650 FORTRAN].

Pessin: Yes. But at least we proved our case and from there on I think we were able to demonstrate that cascading from one processor to another was false economy.

Lee: Of course, by that time you had some experience in writing compilers, so the [second] experiment was slanted somewhat.

Pessin: Well not really, we still only had 2000 words of memory in the 650, and, yes, we knew the source language better, that's true, but we had a different process in a very, very constrained environment. So I don't think it was a slanted experiment.

Lee: Do you have any idea how long it took you to turn out the second version?

Pessin: Well I think it took less time because there were only two of us and we both knew the SOAP language quite well by that time. But we were still inventing compilation for the 650.

Lee: By '58, Irv [Ziller] had his plans for FORTRAN II well in hand, whereas when you did the original FORTRANSIT, that [FORTRAN II] was still in the future. When you did the 650 FORTRAN did you go back to the original FORTRAN?

Pessin: No, as I said, we used the same language level.[*] We were being pushed to get it done because the 7070 was then on its way and because of my experience I went over to the 7070 group. Not in FORTRAN as it turns out, but I was able to offer some help. Otto Alexander went over to 7070 FORTRAN.

Lee: Is Otto Alexander still with IBM?[*]

-------------------

[* Alexander]: Our efforts were primarily oriented to the 7070 at this time. Irv Liggett as 7070 Programming Manager was not interested in another 650 compiler.
[* Alexander]: ... as FORTRANSIT. 7070 FORTRAN was FORTRAN II. 7070 Basic FORTRAN was 650 FORTRAN.

Pessin: I have a vague idea that the answer is yes and maybe you can track him down. There is another name. Later in the project, we were joined by Leroy May. At that time he was the go-fer for the project, but later he contributed to FORTRAN.[*]

Lee: He also moved over to the FORTRAN II project I believe.

Pessin: Yes, that's true.

Lee: After you go moved over from the 650, did you move away from FORTRAN?[*]

Pessin: Yes, I moved away from FORTRAN but I kept my eye on what was going on with 7070 FORTRAN. Somewhere along the line came the 705 implementation and in the interim I was working on some languages which have not survived. After the 7070 I went over to Commercial Translator and COBOL.

Lee: You didn't participate in Commercial Translator did you?

Pessin: Yes, I was one of the designers of the language.

Lee: But that was still the late 50's or early 60's wasn't it?

Pessin: Yes, that was in '61 or thereabouts and I was also the project coordinator for the 7070 implementation. There were three parallel implementations, a 7090 one in Los Angeles, a 705/7080 in New York and the 7070.

Lee: Were you at 57th Streeet and at 590 Madison Avenue?

Pessin: Yes, originally. The original FORTRANSIT work was done at 56th Street in New York, the Langdon Hotel.[*]

Lee: That has since been torn down.

Pessin: Yes, we were all on one floor. Backus and his people were at one end of the corridor and Bemer and his people at

---

[* Alexander]: Retired in 1979.
[* Alexander]: Leroy's prime objective in life at that time wa to remove enough symbols from the code to permit the assembly of the processor in one pass through SOAP.
[* Alexander]: I think that Sam Kaufman (WHQ) and Ward Klein (Ed. Center NY) are the only ones still with IBM.
[* Alexander]: The programmers were in NYC while the 7070 was at the Glendale Lab. in Endicott with only trains (for transportation) from NYC to Endicott.
[* Alexander]: (The machines) 704 and 650 were on the ground floor of 590 Madison Avenue.

the other end. They just moved out the beds and things and moved in IBM grey desks. That was a very pleasant work place and very unusual. Very non-IBM.

Lee: What was the administrative discipline [in those surroundings]? I get the impression that it was a very free and open atmosphere in the Landgon Hotel.

Pessin: Yes, it was very "researchy" in nature and we were all very highly motivated and we cared a lot about what we were doing. It seemed that no matter what time of the day or night it was, there was somebody there working.

Lee: Did you move into 590 Madison Avenue after that?

Pessin: No. The Applied Programming group was organized. We had space at 425 Park Avenue. Some of the original FORTRAN group also went to 425, some went to research [Yorktown].

Lee: This was after '57?

Pessin: Late '57.

Lee: I think some of the main group went to 590 Madison by the time of the delivery of the original system.

Pessin: The Langdon was overflow for 590, since it was around the corner from 590.

Lee: Is that what they called "The Annex"?

Pessin: No, the Annex was around the corner on 56th Street but was contiguous with 590. It was one of those brownstones and they broke a wall through. We were across the street in rented non-IBM space.

Lee: The reason I asked about this, is that Hal Stern in one of his letters to me, commented on how the attitudes towards work changed over the period '57, '58 where it used to be very much "research" early on, people did some work, would relax [*] for a while, and there was nobody saying "punch a clock".

Pessin: Yes, during the time we were there never punched a clock. But, except for times when I was involved in a project trying to meet a deadline, I worked harder at that time than at any other because we were doing it for the love of it.

-------------------

[* Alexander]: Chess and Go (were the relaxations) to pass the time waiting for machine time at 590 on the 704 or 650.

Lee: After the Commercial Translator did you go back to FORTRAN?

Pessin: Yes, after Commercial Translator I was involved in a language design group and that's when we designed some FORTRAN extensions known internally as FORTRAN V. Then we got involved with a thing called Apollo, which was supposed to be a combination of FORTRAN and COBOL to replace Commercial Translator. It was our attempt to say that we were viewing the dichotomy of commerical and scientific languages with a little bit of a jaundiced eye. We wanted to bring then back together in one language.

Lee: George Radin [3] talks about the period prior to PL/I when there was a tremendous concern about the dichotomy between the two types of systems. He was trying to run two types of operations on different machines, and I think George's words were "somebody realized the worlds had to come together". So Apollo would fit into that era.

Pessin: Apollo was in that era, it was in the 1962-63 time frame. I managed a little design group.

Lee: By the 1962-63 period! - this then ties into the 360 architecture.

Pessin: Originally no, but later yes. We knew about the 360, but it was very early. Apollo never got off the ground, and I left the company. In a brief attempt not to leave the company, I went to work for David Sayre in Yorktown Hts. That lasted 4 months and then I left the company.

Lee: David had not returned to crystallography by that time?

Pessin: No he was working for Herman Goldstine in Research. He had an advanced programming group and I got involved. I read Iverson's book and reoriented my thoughts about language design.

Lee: Did you know about APL?

Pessin: No I didn't know about APL until I went to the research division.

Lee: In 62-63 Iverson published his book and had begun his work in the implementation. So you were outside the company at that time?

Pessin: Yes. I went to work for a software house and came back

--------------------

[3] Radin, G., "The Early History and Characteristics of PL/I" in History of Programming Languages, Wexelblat, R., (Ed), Academic Press, 1981. Poorly paraphrased!

in September '66. At that time, Time/Life had been given the FORTRAN mission and tragically, there was pitiful little FORTRAN expertise left. There was a small group of five people who knew nothing about FORTRAN but whose job it was to maintain the FORTRAN G compiler after it was built by DIGITEK. They would get new DIGITEK listings weekly and try to read the FORTRAN routines. One of those people was Phil Shaw [IBM, Santa Teresa in 1981].

It turns out that depending on how you count, there were something like thirteen to fourteen FORTRAN processors of one kind or another for the 360. On DOS and OS, a BPS card, a BPS tape, and all sorts of variants of this kind. So one task in rebuilding FORTRAN expertise at that location involved gathering together all those miscellaneous FORTRAN's and trying to bring about some order. Compatability was the last thing anyone had thought about when they did another one of those fourteen implementations.

Lee: By '66 the FORTRAN standard was available.

Pessin: Yes, there was a FORTRAN standard but it didn't have any retroactive effect. As it turned out, some of those processors had only two or three dozen users; they were all internal users rather than external customers and it was absurd to be spending a lot of money maintaining these things for no reason at all.

Lee: Is the reason for the diversity that they all came out of different sections or divisions of IBM?

Pessin: They came from different locations, with different purposes. Many of them were supposed to be stopgap measures but you know what happens - nothing dies unless you kill it. BPS [4] was one of those things used internally, a lot by SRI and various product test groups to train junior programmers in various maintenance activities but there were no customers. We tried to make some sense out of that chaos, bring them together, establish certain compatabilities, even change them. What emerged, of course, eventually was that [FORTRAN] G and H were the two main contenders. And those were incompatible in a number of different ways. We established an interlocation council, brought together the FORTRAN experts from various locations from time to time, on a regular basis, to hash out how we wanted to go, which one of two or three alternatives we would use.

Lee: Were you fighting PL/I for support at that time?

Pessin: Not in '66. By '68, I guess it was, the language

---

[4] BPS - Basis Programming Support

strategy was promulgated and that was when PL/I was annointed.

Lee: That was the single language strategy was it not? Who was really behind that? Fred Brooks?

Pessin: Not to my knowledge. It is possible, but I think it was a very high level decision that, I guess, appeared on the surface to be a very palatable, a very rational approach to take.[*] I think what was wrong with it was the same thing that hit us with unveiling the 360 on an unsuspecting world. That is to say, we were naive about the cost of the investment in program libraries, in programmer training, etc. I think we understand that now. But then we didn't provide facilities for them to convert.

Lee: But IBM had just gone through, a few years before, a very successful hardware strategy, saying "lets have a single family of computers." Do you think that decision influenced the idea of a single language strategy?

Pessin: I don't know for sure; it certainly was possible. Conceptually it is a very nice idea. In terms of cost to support, from an IBM point of view, it would have been much more attractive to have one language instead of many.

Lee: But the rest of the world wasn't willing to go along with that concept.

Pessin: I think that's the point. I think that given well-entrenched users of COBOL and of FORTRAN, it was an uphill battle. We were trying to introduce a third language and say "this is it - the wave of the future, you've got to move." It may also, and this is hindsight of course, have been untimely in the sense that we had caused our customers a very traumatic conversion when they went from the 7000 series to the 360 and they made that investment but were unwilling to do it again.

Lee: There were a lot of promises that came with the 360.

Pessin: I think the memory was still very very strong in their minds of what it took to convert and I don't think there was a lot of willingness to do it again - from their entrenched language to this great new pie in the sky. I think thats what it boils down to.

Lee: Let me go back to the time you were working on FORTRAN. What do you think was your greatest contribution to FORTRAN? And what is your favorite anecdote about that era?

-------------------

[* Alexander]: I thing it was Ted Climis and Carl Reynolds.

Pessin: I think we've touched on some of the technical contributions. I think the problem then was that we didn't realize we were ploughing new ground and therefore didn't document out inventions. I think now that is a loss, but we didn't even realize it then.

Lee: You talk about your tables of encountered operators yet I can remember Danny Leeson [4] had a chapter on those tables. I wonder if those came from FORTRANSIT. They didn't come from the original FORTRAN. They were included in the 1620 version and that seemed, to me, to follow on from the 650.[*]

Pessin: I don't know, I really don't know because I don't know what approach those [1620] people took. In FORTRANSIT the big things were the linearization of the subscripts of arrays, the preservation of the hierarchy of operations from FORTRAN to IT, and in the 650 FORTRAN the notion of Polish notation in the way the tables were organized.

Lee: Did Al Perlis have any other contact with FORTRAN other than providing IT?

Pessin: We used to see him fairly regularly to get his latest version of IT. But did he influence the processor as such? No. His involvement was limited to his language.[*]

Lee: IBM didn't buy that language [IT], did they?

Pessin: No.

Lee: I guess he was at Case then, or was it Carnegie?

Pessin: He was at Carnegie Tech, and he had a couple of graduate students who published papers with him. (I don't recall their names.) We never published papers, which is regrettable now.

In later years, my contribution was more of an administrative one, from a managerial point of view, bringing the FORTRANs together, establishing the principle of consistency and compatability.

--------------------

[4] Leeson, D., and Dimitry, D.L., Basic Programming Concepts and the IBM 1620 Computer, Holt, Reinhart and Winston, Inc., New York, 1962, 368pp.

[* Alexander]: FORTRAN for the 1410 operating system is based on this. Al Duke and Larry Brown were technical giants on th 1410 -- check with Scott Locken.

[* Alexander]: He was a great disbeliever in language to language translations.

Lee: One final question. You had a first hand view of what was happening in John Backus' group. How do you view their contributions to the industry?

Pessin: They did more than invent a language, they invented a process that moved computing forward lightyears - the concept of compilation of programming languages.[*]

Lee: Do you think John's latest proposals for programming languages [Functional notations] will take off?

Pessin: Well I have enormous respect for John. I've heard his presentation and I must say a great deal of it confused me, and it seemed to be completely upside down, which is one of the things he predicted at the start of the talk. I'm not knowledgeable enough in depth to know what will come of it, but based on John's form, it's got to win.

END OF TAPE

--------------------

[* Alexander]: At a time when there was much discussion over whether writing in symbolic language could possibly provide code comparable to that written in machine language! (The question was) "Will the cost of assembly kill us?"

seemed to be completely upside down, which is one of the things he predicted at the start of the talk. I'm not knowledgeable enough in depth to know what will come of it, but based on John's form, its got to win. ~~Given 20 years [since FORTRAN] we've become very emotional about our~~ current languages.

I think the process will require a great deal of re-education amongst people currently in computing and maybe we have start earlier in the schools.

I have one anecdote about John himself. When I was being interviewed by Bob Bemer, before coming to work for IBM, he and I were in an office at the Langdon Hotel and this eighteen-year-old college kid walked in and sat down at another desk in the room. I thought to myself "how odd, here I am being interviewed and this kid is sitting there." He got up and walked out and came back again. I though "what a strange place". Two weeks after I came to work for IBM, I realized that it was John Backus, who was the boss and he wasn't an eighteen-year-old college student any more than he was a football player! John and Bob shared an office together.

END OF TAPE * Backus, J., "FORTRAN I, II and III", in Wexelblat, R., History of Programming Languages, Academic Press, 1981.

* Samelson and Bauer

* Radin, G., "The Early History and Characteristics of PL/I" in History of

At this meeting, the LITs were so impressed that they decided a large SACK was needed to produce more and better GOODSTUFF and thus enhance and further their various activities and those of the new King. They appointed a committee to draw up some general requirements.

—George L. Whalley
Alexandria, Virginia

# SOFTWARE PARTS NOSTALGIA

The idea that software should be built up from off-the-shelf parts is receiving a great deal of attention in software's contemporary literature.

It is an extremely attractive idea for two reasons. First, the software builder can reduce both cost and schedule considerations because prewritten software is immediately available. Second, he can increase software quality, since pretested software is generally of higher reliability than freshly written software. Since cost/schedule and quality are often competitors in a difficult trade-off game, it is especially nice to find a methodology that enhances both. The notion of software parts, then, has a near-magic allure, especially in an era where "productivity" is the number one buzzword.

There are two ironies here. One is that the software parts approach is a bottom-up one and thus conflicts with the top-down approaches of the '70s. An even greater irony is that our field has not improved, but instead has suffered a major regression in the software parts area during the past 25 years.

Probably 95% of today's software developers were not in the field in the 1950s, and precisely for that reason, it is worth spending some time discussing that era.

When we open the door of the 1950s' "Computing Laboratory," several things leap to our attention: crew cuts on the programming men, bouffants on the programming women; the clatter of keypunch machines; the immensity of the computer room—all that square footage for a computer that, by today's standards, is truly tiny. Let's look a little closer. There, on the desk of every programmer . . . what's that manual? Noting that it says SHARE on the binding, we open it, study it, and a light slowly dawns. This is a software parts catalog, and every single programmer either has a copy or has access to one.

"Where did this come from?" we ask a nearby young programmer. (Interesting—every one of them is young, as you might have noticed.)

"Oh, that's the SHARE manual," he answers, offhandedly. "SHARE is our user group. We all contribute software routines to SHARE, and we all use what has been contributed."

"What about this page? It describes a uniform distribution random number generator. Where'd that come from?"

"Oh, that's from United Technologies. Fred Masner wrote it. In fact, he's written a lot of SHARE stuff."

"And what about this character string read routine?"

"Northwest Industries. Bill Clinger did it. His stuff is excellent, and it always works right."

Let's pause for a minute. It's important to realize a couple of things. First, in the '50s there was no academic computer science world worth speaking about. That development was still nearly a decade away. Programmers emerged from training in mathematics, business administration, or even English. And that, in turn, meant there was almost no computer science literature: a little bit of *Communications of the ACM*, but not much; a more universally available DATAMATION; a doomed fledgling called *Software Age*

## READERS' FORUM

As a route to software prestige, publishing was a limited outlet.

Second, Vendor software hadn't been unbundled yet. In fact, it hadn't even been bundled. Computer hardware often came with no software at all. And that's where the user groups, like SHARE, came in. It was a group for the sharing of software which was not available anywhere else.

Back to the '50s. The SHARE manual, a collection of software parts descriptions, begins to make a little more sense. Up front, here is a table of contents. Scanning down quickly, we can see a functional breakdown of software parts. Here's a section on "Input/Output," another on "Character String Manipulation," another on "Mathematical Services," and many more. Let's flip back to the math section to see how an individual section is organized.

Again, we see a functional breakdown. There's a section on trigonometric routines, another on matrix manipulation, another on integration routines, still another on random number generation.

Well, let's look even more closely. What's at the bottom of this whole parts taxonomy?

This page looks typical. Here's a first paragraph describing the functions performed by the part. Then we have the author's name and corporate affiliation. Now there's a description of the input requirements and the output produced, and finally, a d.scussion of restrictions and some miscellaneous notes. Usually there's one page per part. Sometimes, for the complex ones like I/O, there are two or three. Occasionally, when it matters, the underlying algorithm is discussed.

But always, near the top of the page, is the author's name and affiliation. And always, near the bottom of the page, is a disclaimer—"This software has been tested, but it is not guaranteed to be free from error"—or words to that effect.

"Is this stuff any good?" we ask the nearby programmer, wondering about that disclaimer.

"Yes, nearly always," he says. "In fact, if you read the code, you'll find it's usually—and I really hate to admit this—better than the best I can do. Most people don't contribute crumby stuff to SHARE—there's too much at stake. And we quickly spot the ones who do."

"Too much at stake." "Spot the ones who do." Another light is dawning. The building of software parts, in an era where there is little drive to "publish or perish," is the route to software renown and prestige in the '50s. It's a highly individualized effort, the success route of the single contributor. And there's an automatic screening out of the inept.

Let's browse through the SHARE manual a little more. Sure enough, some names and affiliations recur about every fifth or tenth page. That's why our programmer friend immediately remembered the names of Fred Masner and Bill Clinger, and United Technologies and Northwest Industries.

Here's what we were doing right in the '50s:

First, there was a thriving software parts technology. Everybody expected to have prebuilt parts available to them.

Second, there was an effective parts taxonomy and an effective delivery document. If you wanted to find out what parts were available, you could easily do so.

Third, there was pride in software authorship. Parts appeared in the shared domain because there was strong motivation to do it.

Fourth, there was no stifling counterinfluence. Software was not available "free" or at low cost from the vendor; it was either shared or developed by individual users.

Looked at in this light, the '50s are a wonderful model for the present. What an irony that where we are going is where we've already been.

What went wrong? I saw it happen, and it's a sad and frustrating story. As the '50s blurred into the '60s, it was apparent that software was increasingly more difficult to produce. I/O packages might be SHAREd, but could an operating system? More and more, SHARE members (and other user groups) pressed the vendors

RE

to c
soft
mor
soft
hoa

its p
the
noti
ies–
ven
ven
writ

com
can
acro
eme
part
Lea
aro
inve
ing
fun
mus
betw

hav
ofte
a sp

the
soft
tean
tean
was
that
with
man
arti
tack
are
the

SHA
take
pre
con

'80
● C
● I
● E
● D
● D
bew
cer
ope

con
tha
peo
cat
co
ne
to

to deliver the software, and eventually they did. The SHARing of software atrophied. After all, couldn't Big Brother do it better and more reliably? SHARE meetings changed from a community of software users presenting and sharing solutions to a clamoring hoard of users shouting "Gimme" at the vendors.

The SHARE manual fell into disuse, and finally vanished. In its place came a mile-long shelf of vendor literature. It emphasized the system, and often the use of tools within the system, but the notion of software parts—except for a few things like math libraries—simply disappeared. After all, a plethora of parts leaves a vendor open to a lot more user interaction and complaint. And can a vendor stamp that all-important disclaimer at the bottom of the writeup and legally get away with it?

A couple of other things happened, too, although their combined effect on the software parts community was less significant. Computer science departments sprang up in universities across the land, and a theory of computer science gradually emerged. The energies that had gone into producing better software parts now went into producing better software theories. Belady and Leavenworth said it best: ". . . software engineering is polarized around two subcultures—the speculators and the doers. The former invent but do not go beyond publishing novelty, hence never learning about the idea's usefulness—or the lack of it. The latter, not funded for experimentation but for efficient product development, must use proven, however antiquated, methods. Communication between them is sparse."

We all appreciate the rise of software theory, but what we have forgotten is acknowledgment of the software doer. All too often the doer is the butt of negative published comments written by a speculator.

The final strike against software parts was the emergence of the "egoless programmer" concept. Because ever-more-complex software required ever-more bodies to produce it, the notion of a team approach to software construction surfaced. And in those teams, human ego seemed to get in the way of team progress. That was true, of course. What was missed in this concept, however, is that human ego is an essential drive which cannot be suppressed without bad side effects. Can you imagine, for example, an egoless manager? Or can you imagine an egoless theoretician, publishing articles in professional journals with no name and affiliation attached and with no feedback to academic heads of department? We are all powerfully motivated by our egos, and when they are denied the result is lethargic irresponsibility.

This is precisely what went wrong with the old, true-sharing SHARE. A strong authority (the vendor) emerged and said, "We'll take over all this software tools and parts stuff; don't you worry your pretty little user programmer heads about it." With no ego pull to contribute parts to a SHAREd library, the parts stopped coming.

So what can be done to hasten the software parts era of the '80s? Learn from the '50s, of course. At your computing shop:

- Create a parts taxonomy and the shell of a parts document.
- Invite programmers to contribute generalized parts to the shell.
- Establish some sort of reward system for parts contributors.
- Distribute parts catalogs to all programmers.
- Decide either to allow disclaimers on parts, with a low-cost "user beware" mode of operation, or to establish a centralized parts certifying organization, with a high cost but high reliability mode of operation.

Gradually, within your computing shop—if not between computing shops—a thriving parts subculture will develop. Out of that subculture will come a collection of parts provided by the people most likely to understand what parts are needed—the applications programmers. And out of the reward system will come a collection of top programmers, their egos intact, who will have a new reason to feel proud of what they are doing, and visible rewards to show for it.

We know it can happen, because it all happened before.

**—Robert L. Glass**
**Seattle, Washington**

# SHARE

PAUL ARMER

*Mr. Armer is the Head, Numerical Analysis Department, mathematics division, The RAND Corporation, Santa Monica. He received his Bachelor's Degree from the University of California at Los Angeles in 1946 and served with the United States Air Force during World War II. As head of RAND's Numerical Analysis Department, Mr. Armer directs the activities of a group with programs, codes and operates an IBM type 704 computer, a digital computer of the Princeton type built at RAND called the JOHNNIAC, an analog installation, and a battery of punched-card equipment. The primary concern of the department is scientific computing, but some studies of data processing for the United States Air Force is carried on. Mr. Armer is a charter member of the Digital Computers Association and the first treasurer of the Los Angeles Chapter of the Association for Computing Machinery. He is a Charter member of SHARE and was recently elected to the Executive Board.*

Whenever someone asks about SHARE, the first question is usually "What do the initials mean?" The answer is that SHARE is a name and not a set of initials. The second question is usually "Just what is SHARE?" SHARE has been frequently described as a "users cooperative". It is made up of most of the organizations who have, or plan on getting, an IBM Type 704 EDPM. Like any cooperative, SHARE was formed to be of service to its members. Its aim is to eliminate, as much as possible, redundant effort expended in using the 704. It seeks to accomplish this aim by promoting inter-installation cooperation and communication.

### HISTORICAL SKETCH OF SHARE

As I attempt to paint a historical background for SHARE, it is important for you to remember two things about me, for what anyone has to say about the past is always greatly influenced by his vantage position. The two points are that my primary field is scientific computing and that all my experience has been with the equipment of one manufacturer, IBM. Although the latter point may affect what I have to say about the past, it has no bearing on my discussions of the future.

Before taking up SHARE itself, let's turn our attention to the history of cooperative effort in the field of machine accounting and computing. Since almost all early computing efforts got under way in an accounting machine installation, any discussion of early cooperative effort in computing is necessarily concerned with the machine accounting field. To begin with, we "shared" machine wiring diagrams, usually by submitting such diagrams to the machine manufacturer, who reproduced them and distributed copies to the field. As an example of this, many of you are familiar with IBM's "Pointers". Another important vehicle for the interchange of information of this sort is Fred Gruenberger's "Computing News", published in Richland, Washington. This newsletter frequently publishes wiring diagrams and other "ideas" submitted by its readers. This kind of cooperative activity continues today, although not at the level some would like.

And while discussing cooperation and the interchange of information and ideas, the various professional organizations, in particular the NMAA, should be given much credit for their efforts.

But the important point about these early efforts at cooperation is that seldom, if ever, did individuals from more than one organization sit down together to develop something through cooperative endeavor which each could take back to his own installation and use. Actually, this wouldn't have made much sense in the early days when machine work was divided into many separate and distinct steps. In fact, I doubt if cooperation of this sort made any sense at all prior to 1950 when the Model I Card Programmed Calculator (CPC) was introduced by IBM. Here, for the first time in punched card work, the concept of processing data in a serial fashion ("in-line"), rather than in parallel, was introduced. Now the CPC was really a computer kit rather than a finished calculator, for after it rolled in the door, one had to do a great deal of work designing, wiring and debugging a set of plug boards which connected the various pieces of the kit and made it into a calculator. Here then, was an opportunity for a cooperative effort in putting that kit together. This opportunity was completely overlooked, despite the fact that IBM brought together representatives from each of the organizations getting early model CPC's. Further, this meeting was held in advance of the delivery of the machines. The idea of a cooperative effort just didn't occur to anyone, for we were all too naive about the machine and about handling our work in this "in-line" fashion. But, most important to a moral I'd like to draw in this paper, we were all so naive that each of us believed that we could put the kit together better than anyone else. Consequently, we all went our separate ways and each of us ended up with a unique calculator.

I've somewhat overstated the "lost opportunity" aspects of this situation for it is probably true that, considering how little each of us knew about the machine and about "in-line" processing, it was necessary that we go back to our own installations and learn from our own mistakes. Nevertheless, some sort of sharing of information during the next few years might have reduced the duplication of mistakes that resulted from the spirit of splendid isolationism which prevailed.

Lest anyone get the impression that I entirely disapprove of the versatility inherent in the CPC, let me hasten to add that I don't believe IBM should have delivered the CPC with a set of plug boards, designed by IBM, soldered into the machine, for they didn't know very much about the potentialities of the CPC at this time either. But versatility can be carried to an extreme—what could be more versatile than a kit made up of tubes, relays, resistors, condensers, etc., with each customer left to his own desires?

We also missed our second chance at a cooperative effort when the Model II CPC was introduced, although some of the later organizations to accept machines did copy and use set-ups designed by others. The fact that this opportunity was overlooked can not be laid on the doorstep of inexperience with this type of equipment. The blame must be placed on the "I can do it better" attitude.

When the 701 came along, we still weren't very wise and once again almost everyone went his own way. But this time the amount of redundant effort was horrendous—the cost of developing a system for using the machine, and a set of routines to go with that system, was usually in excess of a year's rental for the equipment. But strangely enough, it wasn't these factors which resulted in what I consider to be the first successful cooperative effort in the field. I am referring to PACT, which is a set of initials and stands for the Project for the Advancement of Coding Techniques. But before discussing PACT further, let me return to the pressures which resulted in its birth.

In the fall of 1954, the several organizations who had been operating 701's in the Los Angeles area were going through a period of self-examination. The one thing plaguing all the organizations was the mismatch between the machine and its language and the human and his language. The elapsed time from problem origination to solution was frequently intolerable, problem check-out was difficult and expensive. People who had estimated that it would take a one-shift operation to handle their production load found themselves operating two shifts, not because they had missed their production estimate, but because they had overlooked a shift devoted to code-checking. Estimates of the cost of writing and checking a program ran as high as $10.00 per instruction. Training was difficult, took a long time and was expensive.

In response to these pressures, a number of interpretive systems were devised. These made problems easier to code and therefore reduce elapsed time and debugging difficulties. They reduced the training problem. But they introduced a new problem, one which frequently outweighed the advantages gained. The new problem was due to the fact that these interpretive routines slowed down the effective speed of the machine by a factor between 10 and 100. There were no longer enough hours in the day to get the machine's work done.

At this point, Jack Strong and Frank Wagner of North American Aviation, suggested that a cooperative effort, aimed at developing an automatic coding system, be undertaken by the computer users in the Los Angeles area. The enthusiasm of Strong and Wagner prevailed and PACT was born. The idea was to find a way to remove some of the coding burden from the human and place it on the machine without materially reducing machinery efficiency. I do not intend to go into PACT here; it did produce a successful compiler for the 701 which is referred to as PACT-1. A series of papers describing PACT-1 appears in the October 1956 issue of the Journal of the Association for Computing Machinery. The PACT group is presently working on PACT-1A, a compiler for the 704.

The important thing about PACT to my discussions today is that it is representative of the kind of cooperation where individuals from different organizations did sit down together to develop a system that each could take back to his own installation and use. In doing this, PACT rediscovered an age old truth that man has been forgetting and rediscovering over and over again since the Stone Age; i.e., cooperation is the greatest invention since the wheel. Actually, this was not an immediate discovery. The members of the working committee of PACT spent several weeks in mutual education, for at first they had to overcome the "our way is best" attitude and also a serious language problem. That this mutual education led to mutual admiration and respect for the other fellows' abilities is testified to by the final report of the PACT-I working committee to the PACT policy committee. I quote from their *Primary* recommendation.

> "The Spirit of cooperation between member organizations and their representatives during the formulating of PACT-I has been one of the most valuable resources to come from the project. It is essential that this spirit of cooperation continue with future project plans."

One might believe that in such a climate, an organization like SHARE would have developed almost spontaneously when the task of preparing for the advent of the 704 appeared. Strangely enough, it was not spontaneous, but rather somewhat of an accident, for even this opportunity for a major cooperative effort almost escaped us.

Three 701 installations in the Los Angeles area began to dig into the problem of preparing for the 704 in the summer of 1955. Because of the climate resulting from PACT-I, these three organizations started to discuss their individual plans with each other and to explore the possibilities of a joint effort in connection with program development for the 704. Accordingly The RAND Corporation, Lockheed Aircraft Corporation and North American Aviation, Inc. seriously began to consider standardization. This much of SHARE genesis was no accident—it flowed naturaly from the PACT experiences of the three groups. The fortunate accident was a seminar held by IBM in Los Angeles early in August for all Western installations considering the 704. The cooperative venture being launched by the three local groups was discussed with others at the seminar and although SHARE may not have started spontaneously, the fire soon burned furiously and spread rapidly across the country. Two weeks after the IBM seminar, the first meeting of SHARE was held at RAND during the week of August 22, 1955. Despite short notice, almost all (18 in number) the installations then contemplating the 704 were represented at the meeting.

I mentioned a minute ago that this opportunity almost escaped us. The problem was a matter of timing, for several organizations were expecting their equipment within three months after the initial meeting and had their systems for using the machine nearly complete. Of the four organizations well along in their plans, one was able to go along with SHARE when their system was adopted, with modifications, by the SHARE body. A second elected to junk what work had been done to date in order to go along. Two others were much too far along with their own systems to turn back; for them, SHARE did come too late.

I think it is important here to understand that SHARE was not organized just to facilitate the interchange of programs for the 704. This was a higher order of cooperation. The organizations who had interchanged 701 programs had found the routines of others almost useless, for each installation had its own system and a routine designed for one system just wouldn't fit into another system without modifications. Hence, it was usually easier to write a routine for your own system, starting from scratch, than to modify someone else's routine. And so, almost everyone wrote his own.

Actually, I personally believe that some of this reluctance to modify and use somebody else's routines can be traced to that naivety mentioned earlier in discussing the CPC; i.e., the belief that the other guy didn't really know what he was doing and that "I can do it better". In any event, the interchange of programs for the 701 had not, in general, been very successful.

At the first meeting of SHARE, disdain for the other fellow's abilities was gone—there was general "agreement to agree"—and almost all professed themselves as quite willing to accept the ideas of others, even to the extent of obsoleting things already done within their own installations. This spirit, however, was not carried to an extreme, for one of SHARE's principles is "unity in essentials and freedom in accidentals". Standardization is undertaken only where necessary. Let me quote from a statement of the "Obligations of a SHARE Member":

> "The principle obligation of a member is *to have a cooperative spirit.* It is expected that each member approach each discussion with an open mind, and, having respect for the competence of other members, be willing to accept the opinions of others more frequently than he insists on his own. On the other hand, majorities of members are not expected to be overbearing in their dealings with minorities. To win over dissenters to unanimity and not to vote them down is the foremost objective in every discusion. When it comes to standards, SHARE insists on adherence to them for communication purposes through SHARE channels to the extent that it refuses to distribute material not in SHARE language. Of course, decisions of SHARE can in no way be *binding* on any member installation so far as its internal operation is concerned. However, the great majority of SHARE members deviate internally only very slightly or not at all from the standards adopted by SHARE. New members are urged to scrutinize carefully any such deviation before deciding tnat it is imperative that they do so. Please note that the foregoing discussion refers to basic contradictions or radically different ways of doing things, and does not refer to minor improvements and additions which will not in the least interfere with normal communications."

As evidence that the SHARE membership paid more than lip service to these principles, let me point to the solid accomplishments of the first meeting of SHARE. After deciding on a loosely knit organizational structure and electing officers, attention was turned to those areas where standardization was essential to inter-installation communication. SHARE standards were adopted for a mnemonic operation code, assembly program, card format and print wheel configuration. A distribution system, the lifeline of the organization, was established. Without this distribution system, SHARE could not exist in the fashion that it does. Among the other decisions made were a definition of what constituted a minimum 704, the location of the binary point and the conventions to be used in writing subroutines. Along the latter lines, the work required to prepare various utility and mathematical routines for the machine was divided among the member installations on a purely voluntary basis. Another item of business of that first meeting was the appointment of a committee to prepare a glossary of terms to supplement the existing computing dictionaries. This came about when we soon realized that we were faced with the language problem which had plagued PACT in its early days.

I don't want to leave you with the impression that all SHARE decisions came easily. There was frequently much wrangling and discussion. But in each case, a spirit of cooperation prevailed and a compromise was reached.

The second meeting of SHARE was held some three weeks later in Philadelphia. This meeting was primarily devoted to a re-evaluation of the assembly program and to reporting on the programming commitments made at the first meeting. Of the thirty-seven programming assignments made at the first meeting, all but two were completed on schedule and more than twenty additional programs were submitted.

Subsequent meetings of SHARE were held in Boston, San Francisco, Chicago, and Denver. A meeting is to be held next month in New York and, in the coming year, meetings have been scheduled for Dallas and San Diego.

Some other topics which have been covered at these meetings include: the use of peripheral equipment, suggested changes to the 704 and to the peripheral equipment, the use of the cathode ray tube display device (the type 740), changes to the assembly program, discussion of forms, standard printer boards, computer layouts, development of a SHARE reference manual, the cataloging of SHARE programs, machine reliability (in particular, and a favorite topic of mine, tape reliability), diagnostic routines, education (both internal and external), machine statistics, programming in general, gadgets built to facilitate use of the computer, debugging techniques, data reduction, data transmission systems and, of course, as the membership grew, we found it necessary to devote some time to our organizational structure.

As of this writing, the SHARE membership has grown from 18 to 62. Included are installations in Canada, France and England. These 62 organizations have some 76 machines on order. Including associated peripheral and punched card equipment, the combined annual machine rentals for the present SHARE membership will one day easily exceed $50,000,000.00.

In addition to the 62 member installations, there are 88 additional organizations on the non-member distribution list for program write-ups.

### ADVANTAGES OF SHARE

Some three hundred programs have been distributed to the membership. There is surprisingly little duplication in this library. In the early days of SHARE, it was a standard joke that everyone was submitting square root routines, since they made convenient assignments for trainees. Nevertheless, there are only five square root routines in the literature. But more important, there is only one for such things as matrix abstraction. There are only three general printing routines. Needless to say, without a cooperative effort like SHARE, there would soon be at least fifty versions of most of the more important routines in the SHARE library.

Using the rough rule of thumb that the cost of setting up a system and its associated routines for a computer is approximately equal to the first year's rental for the equipment, we arrive at the conclusion that the savings to the membership, as a result of the reduction of redundant programming effort, is in the neighborhood of $50,000,000.00.

This seems quite reasonable—consider only the assembly program, which was originally developed by United Aircraft Corporation and subsequently modified by them to conform with suggestions from the SHARE body. By any standards, it's an elegant and complicated assembler. Consequently, it seems appropriate to assume that the cost per instruction in it is at the high end of the $2.00 to $10.00 scale usually quoted as the cost per instruction. Applying the $10.00 rate, we conclude that to develop a similar assembler would cost an "isolationist" some $25,000.00. Although not all the members of SHARE are using this assembly program, most are and therefore we may conclude that the resulting savings are of the order of $1,500,000.00.

Even so, there is a more important point here. Many of the later 704 customers are taking the giant step from slide rules, desk calculators, and/or CPC's to the 704 without the benefit of very much intervening experience with stored program equipment. On the other hand, the SHARE assembler and most of the other routines were developed and written by personnel with considerable 701 experience. Many of the newer 704 users have expressed the opinion that without SHARE they would have been unable to go so far up the computing capability ladder in a single step. In effect, SHARE has multiplied the efforts of the limited number of experienced computer personnel. Not only has it made available programs the newcomers might not have been able to produce for themselves, but in those organizations having a number of experienced personnel, the reduction of redundant effort has released many such people for work on more sophisticated utility and mathematical routines and on applied problems.

Another important advantage of SHARE flows from the personal acquaintanceships developed at its meetings. Subsets of the membership discover common problems—there is much cooperation at the two and three installation level. Information and ideas are continually being interchanged between members, both inside and outside the meetings. Because of the meetings and the distribution system, the transmission of information and ideas is made much easier.

Yet another advantage lies in an area which I haven't mentioned so far. In these days of automation, one of the much used "okeh" words is "feedback". SHARE provides collective "feedback" from the customers to the manufacturer. To me, this is extremely important. Both the customer and the manufacturer are vitally interested in improving the present equipment, in filling needs presently unfulfilled, and in seeing that the next generation of machines properly reflect the customers' needs. As an example of this, consider peripheral equipment. Designed for use with the 702 and 705, the peripheral equipment originally dealt only with cards using the Hollerith code. But SHARE felt a need for reading and punching binary cards. At SHARE's request and with suggestions from SHARE, a method was worked out to do so. SHARE has also provided IBM with collectively considered requests for changes to the 704 itself. And although SHARE has explicitly decided to limit its area of activity to the 704, the discussions between customers and manufacturer at SHARE meetings cannot help but have considerable effect on the computers of the future.

### DISADVANTAGES OF SHARE

I came here to praise SHARE and not to bury it, despite the connotation of "eulogy" in my sub-title. Actually, there is little to say on the disadvantages of SHARE. I think they're all rather obvious. Most important, but still of trivial import on an absolute scale, is that standardization obviously implies some loss of flexibility. And of course, SHARE provides 3 or 4 more meetings per year to be attended. These days, it is almost literally true that one can find enough meetings, in the EDP field, to enable one to avoid ever having to go to the office.

### OTHER COOPERATIVE EFFORTS IN THE COMPUTING WORLD

Anyone who will look at IBM Technical Newsletter No. 10 can conclude, by observing the number of "Systems" for the IBM Type 650 reported on therein, that a great deal of redundant effort went into these systems. And it still is. However, this situation was probably, to some extent, unavoidable. It's like things were with respect to the CPC; each user had to learn about the stored program concept, by his own missteps, before he could be ready for a cooperative effort. Nevertheless, I'm convinced that the 650 area could benefit greatly from some sort of cooperative effort.

From the Remington-Rand Univac Scientific Model 1103A, there is positive information to report. The users and prospective users of this equipment have banded together in a group called USE (Univac Scientific Exchange) with much the same aims as SHARE. This talk could just as well have been given by a member of USE.

Also, the organization of a cooperative group for the IBM Type 705 is underway with the first meeting scheduled for New York during the first week in December.

### THE FUTURE

I'm sure that the cooperative effort for the next model computer will come early and not be almost too late like SHARE. There are undoubtedly other things which will be different this time. Remember that SHARE came into being long after several prospective 704 users had their own systems under development. Because of this, when SHARE considered the question of a standard assembly program, several were essentially finished. SHARE picked one of these (that of the United Aircraft Corporation), with modifications, as its standard. This meant that almost all the burden for the assembler fell on UAC. This time we hope to apportion the load, while combining the ideas of many, by making the assembler the joint effort of a number of installations. This may not be easy because of geography. Few SHARE activities in the past have required that the personnel concerned work together in the same physical location for an extended period of time. However, if we are to have a joint assembly program, a way must be found to lick this problem.

### COOPERATIVE EFFORTS IN THE DATA PROCESSING AREA

As a preface to this topic, it is important to consider the ways in which scientific computing differs from business data processing. In the former field, we are faced with a large number of problems for the computer, most of them fairly small and non-repetitive in the sense that they may be in the production phase for less than a month. In such circumstances, any "good" way to solve the problem is preferred to spending time in search of the "best" way. One tries to develop a "general purpose" system through which almost all the problems can be pushed with a minimum of over-all effort. Tools in the form of utility and mathematical routines are developed to aid in attacking problems with some common attribute. Since these systems are "general purpose" in nature, they are as useful in one computing installation as another.

I needn't tell you that things are much different in the area of business data processing. Here there are a few very large applications which will be used over and over again. In these circumstances, it pays to search for the "best" way of doing a problem and to polish the final program in the interest of machine efficiency. Consequently, the "special purpose" approach is normally preferred to the "general purpose" method.

Another way in which problems of the two fields differ is important when considered in the light of cooperative effort. This difference is that computing deals with an exact science in Mathematics while business data processing deals with the vagaries of the world. The logarithm of a given number is identically the same in every installation and consequently a routine for calculating a logarithm can be gainfully passed among computing installations and used "as is". But could any of you make use, "as is", of the payroll routine of another company?

However, I don't mean to belittle the advantages of a cooperative effort in the business data processing field. On the contrary, I feel such an effort would pay tremendous dividends, even if limited just to getting the people with common problems together. And this reminds me of another important point about SHARE, where the idea is to get top-quality working-level personnel—not just the chiefs—together to discuss common problems. The resulting "mutual education" has been invaluable. I believe that this is an important concept and one which accounts for much of SHARE's success. I feel quite strongly about this point—much is to be gained from the cross-fertilization of top-quality working-level personnel. The section of the SHARE Reference Manual pertaining to "Obligations of SHARE Membership" contains this statement, ". . . it is desirable that each SHARE member be represented at every meeting by at least two men, one empowered to make basic policy decisions and another thoroughly familiar with techniques, programming and detailed operating matters."

But a cooperative effort in the business data processing field need not be limited to a series of seminars. Much could be done to facilitate inter-installation communication and joint endeavor could be brought to bear on common problems. Even the fact that a number of installations have been in actual operation for some time should not hinder the success of a cooperative effort organized for a specific machine. It is not necessarily too late. Much standardization may already exist due to the common practice of adopting the mnemonic code and assembly program supplied by the manufacturer. Inter-installation communication may come fairly easily—further standardization may not be difficult. Rather than being too late, this may be the first time that a cooperative effort is possible for a group having a common machine. For example, it is not reasonable to expect that users of the Univac (or the 705) could have gotten together in the past to adopt standards and to share the burden of preparing for the machine. For almost all the organizations concerned, this was their first encounter with a stored program machine. As with the CPC, a great deal had to be learned by each company about the equipment and about this new way of doing things before a cooperative effort could be undertaken.

As it was with SHARE, I feel that the success of any such venture will depend on the degree to which an attitude of "agreement to agree" pervades the membership. This attitude must go hand in hand with mutual respect for the ideas and opinions of others.

It's redundant for me to say, in summary, that I am enthusiastic about SHARE and about cooperative effort in general—I hope it's contagious.

List of FORTRAN Implementations
(Second Version 82-02-12)

| Date | Name | Machine | Author(s) | Location |
|---|---|---|---|---|
| 1954-57 | FORTRAN (O) | IBM 704 | Backus, et al# | IBM |
| 1957 | FORTRANSIT | IBM 650 | Bemer, Alexander, Pessin, Hemmes | IBM |
| 1958 | 650 FORTRAN | IBM 650 | Pessin, Wu | IBM |
| 1958 | FORTRAN | IBM 709 | * | IBM |
| 1958 | FORTRAN II | IBM 704 | Backus et al, Mitchell, Sheridan, Brady & May | IBM |
| 1958 | FORTRAN III | IBM 704 | Ziller, Nelson | IBM |
| 1960 | GOTRAN | IBM 1620 | Laffan | IBM |
| 1960 | ALTAC | Philco 2000 | Rosen, Goldberg | Philco |
| 1960 | FORTRAN | IBM 1620 | Laffan, Resta | IBM |
| 1960 | FORTRAN | IBM 7070 | Alexander | IBM |
| 1960 | FORTRAN | IBM 705 | Seldon e | IBM/GUIDE |
| 1960 | FORTRAN | CDC 1604 | ? | ? |
| 1961 | FORGO | IBM 1620 | Davidson | Wisconsin |
| 1961 | AUTOMATH-800 | H-800 | Opler, King, O'Conner, Beeber, Hopkins, Brestwick | CUC |
| 1961 | FORTRAN | H-290 | ? | Hankins Inc. |
| 1961 | FORTRAN I | UNIVAC | ? | Rem-Rand |
| 1961 | FORTRAN II | LARC | Computer Sc. Corp. | Rem-Rand |
| 1961 | UT FORTRAN | IBM 1620 | Lee, Field | Toronto |
| 1961 | AFIT FORTRAN | IBM 1620 | Pratt | Wright Patt. AFB |
| 1961 | 1401 FORTRAN | IBM 1401 | Haines, et al | IBM |
| 1961 | FORTRAN | B5000 | ? | Burroughs |
| 1961 | FORTRAN IV | IBM 7090/4 | Larner | IBM |
| 1961 | FORTRAN II | RCA 301 | Hux, et al | RCA |
| 1962 | FORTRAN IV | IBM 7040/4 | Medlock | IBM |
| 1962 | AUTOMATH-400 | H-400 | Greenfield, et al | HIS |
| 1962 | S1 | IBM Stretch | Glennie | UKAEA |
| 1962 | FORTRAN II | RCA 301 | Hux, et al | RCA |
| 1962 | FORTRAN IV | Univac 1107 | Gatt | CSC |
| 1963 | AUTOMATH-1800 | H-800/1800 | Greenfield et al | HIS |
| 1963 | AUTOMATH-1400 | H-1400 | Greenfield et al | HIS |
| 1963 | FORTRAN IV | IBM 7030 | ? | IBM |
| 1963 | KINGSTRAN | IBM 1620 | Field, Jardine, Lee, Lee & Robinson | Kingston, Ont |
| 1964 | FORTRAN | 7090/ATLAS | Pyle | UKAEA |
| 1964 | FORTRAN | SDS910 | Dunlap, Ryan | Digitek |
| 1964 | FORTRAN | ORION | Taylor, Harrigan | Rutherford |
| 1964 | FORTRAN | CDC 3600 | ? | CDC |
| 1964 | Fast FORTRAN | CDC 3600 | ? | Mich. St. |
| 1964 | FORTRAN II | RCA 3301 | Hux, et al | RCA |
| 1964 | FORTRAN IV | RCA 601 | Best | Decision Systems |
| 1965 | PUFFT | IBM 7094 | Rosen, et al | Purdue |
| 1965 | FORTRAN D | H-200 | Greenfield at al | HIS |
| 1965 | GE FORTRAN | GE 235 | ? | GE |
| 1966 | FORTRAN IV | SDS9300 | Owens, Hartman | SDS |
| 1966 | FORTRAN H | H-1200/2200 | Greenfield et al | HIS |

| | | | | |
|---|---|---|---|---|
| 1966 | FORTRAN J | H-4200 | Greenfield et al | HIS |
| 1966 | FORTRAN 66 | CDC 6600 | ? | CDC |
| 1966 | FORTRAN IV | RCA Spectra | Moshos | RCA |
| 1967 | DITRAN | IBM 1620 | Moulton, Muller | Wisconsin |
| 1967 | MOD 8 FORTRAN | H-8200 | Jackson et al | HIS |
| 1967 | WATFOR | S/360 | Shantz, et al | Waterloo |
| 1968 | FORTRAN V | UNIVAC 1108 | ? | CSC |
| 1968 | FORTRAN IV | SIGMA 7 | Owens,Hartman | SDS |

(Cut off 10 years after the release of the first compiler)

(Where do the System 360 F,G,H fit?)

Sources: Sammet, J.E., Programming Languages, P-H, 1969, memory, private correspondence, extensive research by Daniel Leeson (IBM), Martin Greenfield (HIS), and Richard Ragan (CDC).

# et al meaning that group who are identified in the 1957 WJCC paper's author list.

* Officially (from IBM files) this work was a straight conversion from the 704 implementation done by the Backus group. Medlock (private correspondence) attributes the management of this work to Harry Beckish.

? indicates that either the information is unknown or not yet researched fully. Please amplify as necessary.

PROGRAM LISTING OF FORTRAN PIONEERS
(PRELIMINARY)
82/04/02

Members of the original Backus group within IBM which developed FORTRAN

John Backus
Project Manager

Harlan Herrick
Section 1, FORTRAN 0

Irving Ziller
Section 2, FORTRAN 0
FORTRAN II

Robert Nelson
Optimization Section 2
FORTRAN 0

Roy Nutt
I/O FORMAT FORTRAN 0
United Aircraft Corp.

Peter B. Sheridan
Arithmetic FORTRAN 0
Functions FORTRAN II

Lois B. Mitchell Haibt
Section 4, FORTRAN 0

Sheldon F. Best
Section 5, FORTRAN 0
Index Reg. assignments
UNIVAC 1100 FTN
MIT

Richard Goldberg
Section 3, FORTRAN 0
Section 5, debugging

David Sayre
Editor of Ref. Manual
Asst. Project Manager
FORTRAN 0

Grace (Libby) E. Mitchell
Primer, FORTRAN 0
FORTRAN II

*Where not indicated otherwise, the original affiliation of the person is IBM Corp.

## Managers and Assistants to the Original FORTRAN Development Group

Cuthbert Hurd
Level 2 Manager
FORTRAN 0

Charles DeCarlo
Level 2 Manager
FORTRAN 0

John McPherson
Level 2 Manager
FORTRAN 0

Sidney Fernbach
Supervisor, Robert Hughes
Livermore Nat'l Laboratory

Walter Ramshaw
Manager of Roy Nutt
United Aircraft Corp.

Charles Adams
Manager of Sheldon Best
M.I.T.

R.J. Beeber
Assistant, Section 1, FORTRAN 0

Hal Stern
Documenter, Customer Liaison
FORTRAN 0

Robert Hughes
Documentation, FORTRAN 0
Lawrence Livermore Laboratory

Members of development teams for commercially sponsored processors for FORTRAN during the period 1957-60 and their immediate managers.

Robert Bemer
Manager, FORTRANSIT devel.

Harry Be~~c~~kish
IBM 709 FTN II

Florence Pessin
Member, FORTRANSIT group

Bernyce Brady Larner
FORTRAN II

I.C. Pyle
ATLAS S1
UK AEA

Saul Rosen
Philco ALTAC
Philco

Leroy M. May
FORTRANSIT and FORTRAN II

B. C. Chapman
ATLAS S1
UK AEA

Ray Larner
IBM 7090/94 FTN IV

Larry Michaels
IBM 7090/94 FTN IV

Seymour Cray
CDC FORTRAN
Control Data Corp.

Lin Wu
IBM 650 FORTRAN

Otto Alexander
Member, FORTRANSIT group

David A Hemmes
Member, FORTRANSIT group

David Mordy
IBM 7070 FTN

A.L. Harmon
IBM Staff

Wm. Andrus
(deceased)
IBM staff

Herbert Meltzer
FTN system maintenance

C.W. Medlock
IBM 7040/44 FTN IV

a?
Jack L~~e~~ffan
IBM 1620
GOTRAN developer

Frank Beckman
Manager, Appl. Prog., IBM

Ken F. Powell
IBM Appl. Sc. Rep., Pittsburgh

Members of development teams for experimental and non-vendor processors for FORTRAN which were available through user group libraries prior to Dec. 31, 1962.

P.G. Moulton
DITRAN developer
Univ. Wisconsin

M.E. Muller
DITRAN developer
Univ. Wisconsin

Charles Davidson
FORGO developer
Univ. Wisconsin

Users of processors identified in 1, 2, and 3 above with significant applications.

Frank Engel
Modifier of first compiler
Chairman, X3 FTN Committee
Westinghouse-Pittsburgh

George Ryckman
User - GM

Harry Cantrell
User - GE

Herbert S. Bright
Author first program
Westinghouse-Bettis

Thomas W. Martin

User - Westinghouse - Piitsburgh

Doris Clark
User of first FTN
GE Schenectady, 1957

4704

Ruth Callaghan Sheehy
Manager, FTN Test Site
GE Schenectady, 1957

Members of FORTRAN committees of users groups, standards committees and supporting vendor staff before Dec. 31, 1962.

John Greenstadt
SHARE Dist. Secr.

Jim Porter
SHARE FORTRAN Committee

Donn Parker
SHARE FORTRAN Committee
Stanford

William P. Heising
ANSI Committee Chairman

Martin Greenfield
Member, ANSI Committee
Honeywell

Don Furth
SHARE liaison

Stan Closman
SHARE liaison

Norman Sanders
Early ASA FTN Comm. Mbr.
Boeing Aircraft Corp.

Authors of text-books, articles in technical journals and printed manuals before Dec. 31, 1962.

Daniel McCracken
Author, first textbook
Consultant

Elliott Organick
Author, early textbook
Univ. Houston

Teachers and faculty that taught FORTRAN as at least part of a formal course in an accredited institution prior to Dec. 31, 1960.

Jean Sammet
1957 Course at Adelphi
Silvania

Herbert Leeds
IBM Course D.C. 1957

Albert Newhouse
Teacher at U. Houston

1982
National
Computer
Conference

Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

Reply to: John A. N. Lee
Pioneer Day Chairman

1982 April 19

Otto Alexander
615 Green Drive
Kissimmee FL 32741

Dear Otto,

Many thanks for your annotation of Flo's interview; I have decided to keep it in exactly that form provided that Flo does not object. Of course I will go through some editing with you and her to get a final form which will go into our archives and which will be the basis of the material that we publish on FOR TRANSIT in the proposed book on FORTRAN which we are working on.

One of the plans at Houston is for Bob Bemer to introduce the members of his "team" to the audience. As I have pointed out to him, to have persons stand up in a room of several hundred people is not very satisfying -- all one sees is the back of their head or a dark shape. Thus we are planning on simultaneously showing their picture as they are introduced. We now have slides for all of Backus' group and I have pictures from Bemer, Hemmes and Pessin. Could you send me a photograph (preferably in color) which we could make into a slide for this purpose? Obviously this would also become part of the archive.

Regarding your being in Houston; I have contacted several persons in Boca Raton and Tampa to attempt to get someone to take responsibility for this expenditure. I am also asking Flo (as the only member of your group still in IBM) for her suggestions and support. I will let you know of my success.

Hoping to see you re-united with your colleagues in Houston,

Yours sincerely,

John A. N. Lee
Professor

xc: Pessin, Hemmes, Bemer

7071

9PAC<br>SOS

## Automatic Programming Systems*

| Computer | In ACM Library | Do Not Have |
|---|---|---|
| 704 ✓ | AFAC<br>CAGE<br>CORBIE<br>FORTRAN ✓<br>KOMPILER 3<br>MYSTIC<br>NYAP<br>PACT IA<br>REG-SYMBOLIC<br>SAP | ADES<br>FORC<br>SURGE |
| 701 ✓ | BACAIC<br>DUAL-607<br>FLOP<br>JCS-13<br>KOMPILER 2<br>PACT I<br>QUICK<br>SEESAW<br>SHACO<br>SPEEDCODING 3 | BAP<br>DOUGLAS<br>GEPURS<br>LT-2<br>QUEASY<br>SO 2<br>SPEEDEX |
| ? II. 705 ✓ | ACOM<br>AUTOCODER III<br>ELI<br>PRINT I FORTRAN<br>SOHIO<br>SYMBOLIC ASSEMBLY | FAIR |
| 702 ✓ | AUTOCODER<br>ASSEMBLY<br>SCRIPT | |
| 650 ✓ | ADES II<br>APT<br>BACAIC<br>BELL<br>BELL L2, L3<br>CASE SOAP III<br>DRUCO I<br>EASE II<br>ELI<br>FAST<br>FOR TRANSIT<br>FORTRUNCIBLE<br>IT<br>IT 3<br>MYSTIC<br>RELATIVE TASS<br>RUNCIBLE<br>SIR<br>SOAP I<br>SOAP II | BALITAC<br>ESCAPE<br>FLAIR ✓<br>KISS<br>MITILAC<br>OMNICODE<br>SPEEDCODING<br>SPUR |
| 650 RAMAC | GAT-2 | |
| NORC ✓ | NORC COMPILER | |
| 7070 | BASIC AUTOCODER FORTRAN<br>AUTOCODER<br>COML TRAN | |

| Computer | In ACM Library | Do Not Have |
|---|---|---|
| 1103-<br>1103A | APT<br>BOEING<br>CHIP<br>FAP<br>FLIP-SPUR<br>MISHAP<br>MYSTIC<br>RAWOOP-SNAP<br>UNICODE<br>USE | TRANS-USE<br>SCAP (MSD) |
| UNIVAC I, II<br>UCT<br>X-2 Jun 60 | A0, A1, A2<br>ARITHMATIC (A-3)<br>BIOR<br>FLOWMATIC (B-0)<br>GP<br>GPX (II ONLY)<br>MATHMATIC (AT-3)<br>MATRIX MATH<br>NYU, OMNIFAX<br>SHORTCODE<br>UNISAP<br>X-1 | MJS<br>RELCODE |
| DATATRON<br>201<br>204 ✓<br>205 | APX III<br>DUMBO<br>PURDUE COMPILER<br>SAC<br>SIMPLE<br>UGLIAC | ANCP<br>BELL<br>DATACODE I<br>DOW COMPILER<br>SHELL<br>SPAR<br>STAR 0 |
| G-15 | DAISY 201<br>FLIP<br>INTERCOM 101<br>INTERCOM 1000 | POGO |
| WHIRLWIND ✓ | COMPREHENSIVE<br>SUMMER SESSION | ALGEBRAIC |
| FERUT ✓ | TRANSCODE | |
| JOHNNIAC | EASY FOX | |
| ILLIAC ✓ | ILLIAC | |
| LGP-30 ✓ | ERFPI<br>JAZ<br>SPEED | |
| MIDAC ✓ | EASIAC<br>MAGIC | |
| LARC ✓ | | K5<br>SAIL |
| FERRANTI<br>MERCURY ✓ | AUTOCODING<br>MAC (NORWAY) | |
| FERRANTI<br>PEGASUS ✓ | AUTOCODE | |

[*] See *ACM Communications 1* (Apr. 1958), 7.

59 MAY (58 NOV<br>58 AUG<br>58 APR)

H 800 – ARGUS<br>TRANSAC = TAC

which is to replace the $j$th column of $A$, where $x^j$ is the trial solution dropped in this iteration. The new inverse is calculated from the old by pivoting:

Calculate $q = A_*^{-1}p$. Then

$$(A_*^{-1})_{jk} = (A^{-1})_{jk}/q_j \qquad \text{for all } k,$$

and

$$(A_*^{-1})_{ik} = (A^{-1})_{ik} - (A^{-1})_{jk}\cdot(q_i/q_j) \quad \begin{cases}\text{for } \neq j,\\ \text{all } k.\end{cases}$$

## 4. Computational Experience

A FORTRAN II program has been written for trying out this procedure. Its input consists of $n$, a set of trial solutions or a signal that such a set should be generated, and programs which calculate the $f_i$. A variety of problems with $n = 2$ have been solved. The process has converged for these in a manner like that which Jeeves [1] has shown for the case $n = 1$, namely that the error at a given step is proportional to the product of the errors at the two previous steps—convergence of order $\frac{1}{2}(\sqrt{5} + 1)$.

SAMPLE: $n = 2$, $f_1(x, y) = x^2 + x - y^2 + 1$, $f_2(x, y) = y(1 - 2x)$ (the real and imaginary parts of $z^2 + z + 1$).

| | | *Points* | | *Norm* |
|---|---|---|---|---|
| | | $x$ | $y$ | |
| Initial | 1 | −0.600000 | 1.100000 | 0.370000 |
| | 2 | −0.300000 | 1.100000 | 1.518400 |
| | 3 | −0.600000 | 1.400000 | 0.250900 |
| | 4 | −0.516058 | 0.923358 | 0.011351 |
| | 5 | −0.503347 | 0.870741 | 0.000101 |
| | 6 | −0.500884 | 0.866819 | $0.423 \times 10^{-5}$ |
| | 7 | −0.499988 | 0.865996 | $0.306 \times 10^{-8}$ |
| | 8 | −0.500000 | 0.866025 | $0.106 \times 10^{-12}$ |

### REFERENCE

[1] T. A. JEEVES, Secant modification of Newton's method. *Comm. Assoc. Comp. Mach. 1*, No. 8 (1958), 9–10.

~

# Automatic Programming Systems

The following are additions to the ACM Library. For the previous status, please refer to the May 1959 issue of *Communications*, page 16.

| *Computer* | *In ACM Library* | *Do Not Have* |
|---|---|---|
| 709 | Commercial Translator<br>FORC 2<br>FORTRAN | S.O.S.<br>9 PAC |
| [redacted] | [redacted] | [redacted] |
| 705 III | Commercial Translator | |
| 705 | Autocoder III<br>FORTRAN | |
| [redacted] | [redacted] | |
| [redacted] | [redacted] | |
| 1103 | | SLAP |
| 1105 | AIMACO | |
| DATATRON 205, 220 | | FORTRAN |
| G-15 | POGO | |
| TRANSAC | | TAC<br>ALTAC<br>(FORTRAN) |
| H-800 | ARGUS<br>HBC | FORTRAN |
| NCR 304 | NEAT<br>STEP | |
| RCA 501 | Automatic Assembly | |
| ICT 1400 | CODEL | |

59 DEC

Canadian Conference for Computing and
Data Processing, June 9, 10, 1958
University of Toronto
pp. 349-359

# "FORTRANSIT
# A UNIVERSAL AUTOMATIC CODING SYSTEM FOR THE
# IBM 650"

B.C. Borden

International Business Machines

Bob — This copy came from Don Knuth —
any idea who Borden was, or is?
Jar

## INTRODUCTION

With the announcement of the FORTRANSIT Automatic Coding system, two very significant advances have been made in using computers intelligently.

One, we have a universal computer language. A standard method now exists for dictating to machines numerical procedures needed in solving mathematical and engineering problems.

Two, we have an automatic system. A long step forward has been taken in having machines program themselves. We have dispensed with a large portion of the time previously required for detail block diagramming, coding, key punching, key verifying and debugging.

In this paper I shall deal first with automatic programming in general, and define some terms which frequently appear. Following this, a review of the FORTRAN story will introduce FORTRANSIT, the Automatic Coding System for the IBM 650. This will be covered under headings of (a) the Language, (b) the Processor. Finally, a brief look at experiences gained from using the system, where it can be best applied, and what potential savings can be realized.

## AUTOMATIC PROGRAMMING

Computers are wonderful devices. An infinite variety of sequence can be undertaken by a general purpose computer. However, unless we tell it precisely what we want it to do, all this activity is to no avail. Once we have conceived a task worthy of data handling equipment, we must be able to communicate our desires to it. As in every phase of communication, a language problem arises. Efficient delegation of work to machines necessarily requires a thorough knowledge of their native language and often a long series of detailed instructions. We naturally seek to relieve ourselves of tedious repetitive work where possible, and the first step is to teach the computer our language.. The more thoroughly it understands us, the freer we can be in expressing our wishes and in describing our tasks to it. If, then, we can teach it to undertake several operations in a correct sequence merely with a word or two from us, a second much larger step has been taken.

Basically, these are the objectives of automatic programming - to ease our burden. Let us see how this affects a typical computer project.

Putting a computer to work for us usually follows these steps. Once the decision is reached that a certain goal can probably be attained with the help of computing equipment, a thorough analysis is made to determine scope, logical and mathematical methods, significance of figures, accuracies of computations and results.

Then follows programming - block diagramming the flow of the problem solution. Next the coding, a large time consuming part of programming, prepares the program in a form ready for the machine.

To be of any use, the program must be tested and proven accurate - it must be debugged. Errors stem from many sources, the most consistent being clerical. The more letters or numbers that are written, the more errors committed by both coder and key puncher. This error source is further compounded where the language one is using is remote from a familiar one - say English or mathematical notation.

All of these phases - analysis, programming, coding and debugging must be passed before the job can be started. A large part of this time was consumed between analysis and final testing.

Any means to shorten or eliminate this delay will pay off handsomely in both cost of programmers and in receiving results more quickly. Often, knowing an answer today instead of next Tuesday can make it possible to just meet a deadline, to quote sooner on a contract, or to prevent an ill-advised project from being started.

Automatic programs have been developed to take much of the coder's load, produce final programs more efficiently, and already in a form or language suitable for a machine. FORTRANSIT is the most advanced system yet devised to automatically prepare large complex problems for a computer from a very few hand written lines of coding.

## DEFINITION OF TERMS

In order to best appreciate the components of the FORTRAN and FORTRANSIT programs, a few moments considering some fairly general terminology will be well spent. The terms I refer to are words familiar to us all, computer people or not, but they take on special meaning to those in data processing or computing.

Unfortunately, many of these special words or phrases have not been thoroughly defined and accepted by writers in this field. Such words include source and object program, assembler, converter, generator, compiler, interpretive routine and so on. It is not my purpose or intention to give a standard meaning for such terms but merely to identify some of them in the light of a description of FORTRANSIT.

After the analysis stage, the programmer-coder, or whatever we choose to call him, prepares lists of instructions for the machine in some sort of code. This coding is referred to as the source program. In the days of automatic programming we are considering using a computer as an aid to preparing the final set of instructions. This final list of steps to be performed is in the native machine language and is in a character representation medium that can be accepted by the system - this is called the object program.

Man writes the source, the machine writes the object. The idea is to allow for simple source coding followed by an automatic process of preparing the object. We refer to the program that performs this function as the processor. Processors can be divided into two families; executive and translator (Figure 1).

Executive processors receive as input the source coding, interpret its meaning and directly perform the arithmetic or the logic required in arriving at the result. This type of processor, often referred to as an interpretive system, goes past the object program stage - the actual object is never available for inspection or repetitive use. The BELL floating point system on the 650 or PRINT on the 705 are excellent examples.

Opposite the executive set we have the very large family of translators. Many groups can be found which function more or less as follows: source programs are taken as input by the processor and object programs in a different language are prepared in punched card or magnetic tape medium and can be used immediately to run the final job. Translators themselves divide into three sub-sets, converters, assemblers and compilers.

I call a converter a processor which accepts one language and prepares another on about the same level. One might use a converter to change coding from 704 to 709 use. More simply, 650 program steps in one-per-card form are changed to seven-per-card form by a converter.

Most of you are acquainted with assemblers - particularly symbolic ones. A symbolic assembly program is one that receives source codes in as brief a form as possible written in non-machine language (usually alphabetic). The assembler makes absolute memory and code assignments for each step or unit of reference data. Generally, an assembler produces one output line for each input line and in this sense is not generative. Popular examples are SCAT for 709, SAP for 704, Autocoder on 705 and of course SOAP for the 650.

The most powerful translator is the compiler. Source programs are written in a very abbreviated form and it is this processors job to prepare complete sets of instructions from each single input line. A compiler's input is always symbolic but the output may be symbolic or absolute. If symbolic, the compiler is not a complete processor as it has not translated all the way to object form.

One should realize that these are broad classifications and that many processors invade territories of several of these definitions. For instance, the 705 Autocoder is fundamentally an assembler, but has many elements of compiler in it.

## THE FORTRAN STORY

Late in 1954 work started on FORTRAN. A working committee of fourteen was set up mostly from the Applied Programming group of IBM, but with representatives from the University of California Radiation Laboratory and the United Air Craft Corporation. The Name FORTRAN is an acronym formed from the words FORmula and TRANSlato

The project was to develop an automatic coding system which would enable the programmer to specify a numerical procedure, that is to write the source program in a concise language like that of mathematics. Two and one half years of development, exhaustive testing, revisions and more testing led to the completion of the system for the IBM 704. Since the beginning of 1957, it has been in increasing use and has more than met the hoped for goal of reducing coding and debugging times to one fifth that required by earlier methods.

FORTRAN consists of two components. First, the language - that is the proper symbols and rules for using them. Second, the processor - the translation from FORTRAN source program steps or statements, as they are called, to machine language.

FORTRAN, then, is a general coding system originally intended to prepare efficient 704 programs. The experience gained of this machine system led workers in other programming areas to consider its usefulness. It was immediately apparent that the language as it stood was in no way tied to the 704. All that was necessary was to write new processors for other machines and we would have a compatible language. Work started directly on both 650 and 705 processors. FORTRANSIT is the name given to the system on the IBM 650. We will first examine the type of coding techniques employed and then consider the processor system of programs.

## FORTRAN LANGUAGE

To use FORTRANSIT, one must know and be able to use the FORTRAN language. For any coding system, we conventionally define limits of addresses and scope of operation codes. Here we describe different types of statements. Not all the original FORTRAN statement types can be used in FORTRANSIT due to smaller machine configurations on the 650 compared to the 704. This is mainly an input-output consideration.

For the basic 650 there are ten types of statements:

1 Arithmetic  2 Input/Output
6 Control  1 Specification

An arithmetic statement looks like an equation. Polynomials or general mathematical relationships are written exactly as one prepares a formula. Operations include addition, subtraction, multiplication and division. Common functions such as square root or those found in trigonometry can be linked together with standard arithmetic in one FORTRAN statement. As an exact but simple example of such a statement, we could write:

Y = SUM + (0.4A X (B - 0.5 C))/37

Already existing special characters in the IBM card code are used for brackets and function signs.

Symbols used are alphabetic (one to five letters) and can represent floating point or fixed decimal quantities. Where constant numerical amounts are needed, they are written in absolutely without reference to storage cells. Variables may carry one or two subscripts to keep notation as close as possible to the familiar notation. One such statement can be up to 125 characters in length.

The first in the group of control statements is known as GO TO. Statements that are at key points through the program have their own identifying numbers. Transfers are effected simply by writing GO TO a particular statement number. These transfers can be made more powerful by making them conditional. GO TO one of several numbers depending on the condition of some index at conclusion of previous statement.

A very frequently used statement is called IF. This is basically an algebraic sign or zero test, but far more powerful than just "transfer on zero". Each IF statement has an argument which is usually a mathematical expression. After the argument has been evaluated, it is examined for less than, equal to, or greater than zero conditions and corresponding control transfers are then made. We could write for example:

IF (ROOT - 73 X A X B - ( -0.001)) 10, 20, 30

The argument is computed and on minus, zero, or plus status transfer would be made to statements 10, 20 or 30 respectively.

The most powerful feature is the DO statement. Such a statement details a group of arithmetic and logical statements that are to be executed and specifies how often the group is to be performed. For instance as a very simple example we could say;

```
DO 10 I = 1, 22, 3
```

This would mean perform statements up to 10 for values of I varying from 1 to 22 stepping I by 3 each time. All looping or iterative programming blocks are set up and controlled by DO's.

Other control statements are PAUSE, STOP and CONTINUE which are fairly self-explanatory.

Input/output instructions or statements labelled simply READ and PUNCH handle data being entered or punched. A single punch statement will punch a complete array with elements suitably identified.

I referred to a fourth type of statement - a specification statement. This most useful device is called the DIMENSION statement where an announcement is made of the size of all subscripted variables.

Gentlemen, the whole of FORTRANSIT is there in this cursory survey. Arithmetic statements, controls such as GO TO, IF, DO, the usual Input - Output commands and the DIMENSION comprise the basis for coding with the FORTRAN language.

Excellent provision has been allowed for inclusion of subroutines. A standard deck is used for all the floating and fixing routines as well as to perform regular arithmetic in both modes of representation. If needed, extra packages are incorporated to take care of all other desired functions such as logs, transcendental functions, etc. Rules are laid down for supplying one's own routines to the processor. Just as in the past the library of programs for the 650 has been built up to a very large size, we expect that all commonly encountered functions will be available shortly for FORTRANSIT.

## FORTRANSIT PROCESSOR

This language just described is used to write and prepare the source program. IBM's applied programming people worked closely with the "Computation Center" at Carnegie Institute of Technology with a view to developing a "processor" for these source statements. Priority was placed on time. We urgently wanted to have the 650 in a position to be able to accept FORTRAN statements and write object programs. As there already existed a compiler and an associated assembler for the 650, the decision was made to adapt and extend this system of existing programs and get into production.

This, then, was the result. Processing was divided into three phases - conversion, compilation and assembly (Figure 2). The first phase uses the FORTRANSIT deck proper. FORTRAN statements are merely converted to statements acceptable to the existing compiler. Incidentally, this is the IT compiler, prepared by Perlis, Smith and VanZoren at Carnegie Tech, with some modification.

In phase two, the compiler takes over and explodes these statements into groups or blocks of program steps. IT is responsible for setting up sub-routine linkage for entry and exit to the various float, fix and arithmetic packages. A major function is to generate the initializing, testing and stepping steps required by the DP statements for iterative computing. A most important consideration in planning this processor is that an efficient object program must be written as final output. This means in the case of the 650 a tighly optimized sequencing of control. Such a requirement is admirably met in the FORTRANSIT system as he compiler's output is in symbolic form ready for optimization.

Phase three follows, merely assembling these symbolic steps into actual machine language. Even at this point the package decks for the sub-routines are not needed as the assembler sets aside sufficient memory areas. This assembler is a slightly revised edition of SOAP the standard such program for 650. After this phase, an efficient optimized program in ready-to-load cards is now ready. Because the output from the FORTRANSIT system is in condensed form (several steps per card), the largest problem can be loaded for actual execution in about two minutes.

The FORTRANSIT system described so far has been designed for what we call the basic 650 - the 2000 word drum, modified one-address computer with buffered punched card input and output. Even though a majority of 650 users have a basic installation, many systems have added features such as floating point hardware and three index registers. Others have complete magnetic tape systems. There would undoubtedly be a limitation if FORTRANSIT could not take advantage of these additional features, consequently, several systems are being developed to accept the same FORTRAN statements as a source but to prepare the most efficient object program for the existing complement of features. In addition, the processor itself can be made more rapid by utilizing these expanded machines. For a progress report we can say the following:

| FORTRANSIT | Machine | Status |
|---|---|---|
| I | Basic 650 | In use |
| II | Index Reg. & Float Pt. | In use |
| III | I.R., F.P. & Tapes (3) | Ready July 1958 |

This third edition will permit completely automatic processing sequence with all intermediate storage of statements on magnetic tape instead of punched card form.

All these developments are in line with the universal approach of making one accepted FORTRAN language. Work is nearing completion for a processor to be used on the various 705 models.

## EXPERIENCE

### 1. Timing

During our discussion of automatic programming, we outlined the various stages encountered from start to conclusion of a computer job. For an example of time and cost distribution, consider the breakdown in table 1, showing both personnel and machine loads. These figures are of necessity, very general and approximate, but they represent the order of savings that one can realize by employing powerful automatic techniques. Note that a large part of programming and coding costs disappears. The ratio of object steps to source steps averages fifteen to one; this virtually eliminates key punching and verifying as a significant cost. Not only is the number of program steps manually prepared greatly reduced, but the number of characters per line is often less.

Debugging or straightening out "kinks" is generally a puzzling problem to those not familiar with such an automatic procedure. One hears the question, "How do I make sense of the final program in light of what I originally wrote?" Surprising as it may seem, very few errors are made once the rules of FORTRAN are absorbed. To help, the FORTRANSIT Processor checks for violations of these laws, and will detect the majority that are not of general logical nature. Errors that persist can be further tracked down by incorporating selective tracing in the final execution of the object program. Incidentally, such tracing can be introduced at the FORTRAN coding stage or at final testing sessions. Experience to date has shown that "bugs" are almost always easily located, and then simply corrected in the original statements.

## 2. Application

All automatic programs seem to have a preferred area for best use. About the only comment that can be made about FORTRAN's place in the sun is that engineering, mathematical and scientific problems are best suited. FORTRANSIT is quite efficient in its object program and it need not be reserved for "one-shot" programs. Organizations with 650's and also one or more of the 700 series can try jobs with FORTRANSIT and then run them at higher speed with expanded scope on their larger system. Coding, once completed for one system is done for all. Table 3 shows the relation between the cost of solving a problem by conventional coding methods versus using FORTRANSIT.

I do not have any statistics on the types of projects that have been undertaken in the United States to date on FORTRANSIT, but in Canada, at the IBM 650 Data Centre, we have worked in the fields of statistics, civil engineering, probability in scattering, air force research, etc.

## CONCLUSION

Let us review the various points which strongly recommend using FORTRANSIT where possible. Investigations into the suitability of complex mathematical models for physical situations are much more feasible, the language of FORTRAN being very close to that of mathematical notation. Problems that would have involved complicated linkage and logical connection can be tackled with little regard to these more technical aspects - FORTRANSIT will compile the longest or the shortest according to its rules.

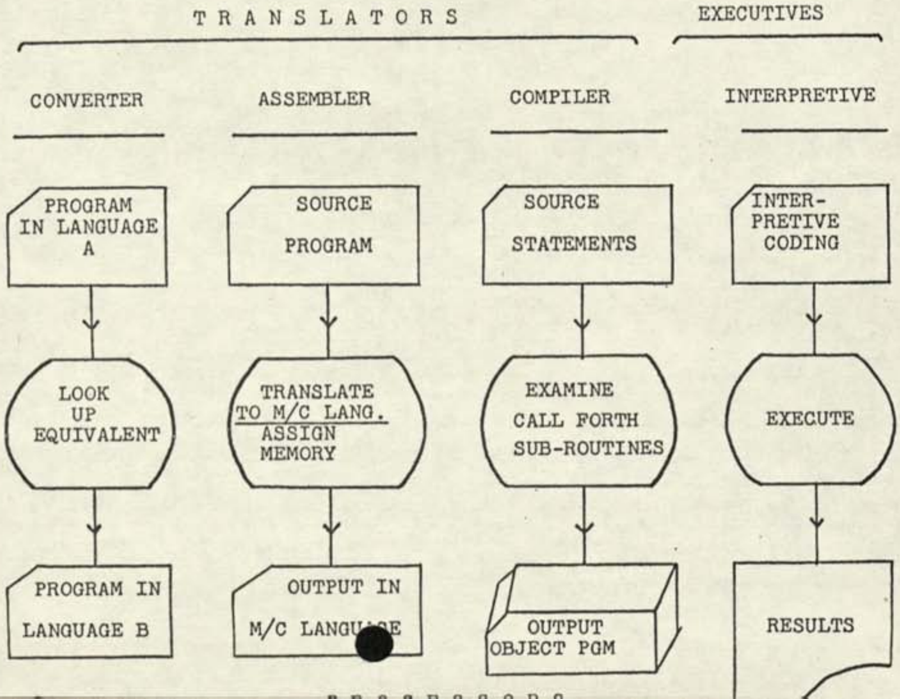
The more obvious cost savings are again; much less clerical
d key punch work, fewer errors, greatly reduced programmer's
ing and debugging duties. Computers as tools are now available
a whole new cross-section of workers in engineering and research.
e have a universal code and when a FORTRANSIT type system is
mpleted for the 705, virtually the same program cards can be processed
approximately 75% of all computers installed or on order. Teaching
RTRANSIT is very straight forward. We were surprised to find,
a recent course here, the man who knew least about the 650
cceeded first in completing his FORTRANSIT problem and getting
e correct result. Another example, two day's private instruction
re given to a customer who proceeded to write a program that
mpiled into over 1900 steps with only one or two small errors.

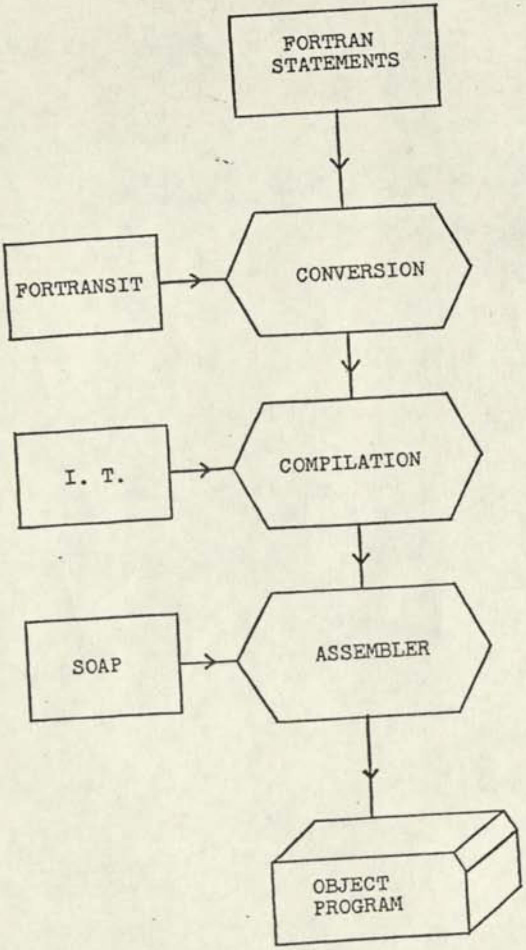
There is nothing inherent in the FORTRAN language which limits
s use on computers of the future of far greater capacity and speeds.
t before that time and until those machines arrive, 650 users
n learn and profit with FORTRANSIT.

| ACTIVITY | | CONVENTIONAL CODING METHODS | USING FORTRANSIT |
|---|---|---|---|
| PERSONNEL | PROGRAMMING | 45 | 15 |
| | CODING | 30 | |
| | KP & KV | 5 | - |
| | DEBUGGING | 10 | 2 |
| | COMPILING | - | 3 |
| MACHINE | COMPILING | - | 10 |
| | DEBUGGING | 10 | 5 |
| TOTAL | | 100 | 35 |

COST BREAKDOWN

Table 1


TRANSLATORS
EXECUTIVES
CONVERTER
ASSEMBLER
COMPILER
INTERPRETIVE
PROGRAM IN LANGUAGE A
SOURCE PROGRAM
SOURCE STATEMENTS
INTER-PRETIVE CODING
LOOK UP EQUIVALENT
TRANSLATE TO M/C LANG.
ASSIGN MEMORY
EXAMINE
CALL FORTH SUB-ROUTINES
EXECUTE
PROGRAM IN LANGUAGE B
OUTPUT IN M/C LANGUAGE
OUTPUT OBJECT PGM
RESULTS



FORTRANSIT

FIG. 2

# AN ANNOTATED BIBLIOGRAPHY OF FORTRAN

Compiled by
J.A.N. Lee

Second Version
82/02/17

## INTRODUCTION

The papers which are listed in this bibliography have been selected on the basis of their applicability to the language FORTRAN, its implementations and its definition. Only in exceptional cases have references been included to the use of FORTRAN in specialized applications. However where the application is in the development of a secondary language, or the modification of FORTRAN to be capable of supporting some application other than "scientific programming", these papers have been included. For historical completeness, a few references have been included to pre-FORTRAN systems which have been referenced in some of the significant other papers on FORTRAN.

Some consideration was given to the problem of an "official" type-face for the name of the language, and in particular whether it should be completely in uppercase. This problem was resolved by using the form which was used by the individual authors. It is interesting to note however that the Preliminary Report by the Programming Research Group of the Applied Science Division of IBM used "FORTRAN".

BIBLIOGRAPHY

(1954) Laning, J.H., and Zierler, N., A Program for Translation of Mathematical Equations for Whirlwind I, Eng. Memo. E-364, MIT Instrumentation Lab., Cambridge MA, January 1954.

(1954)* Backus, J.W., The IBM 701 Speedcoding System, JACM, Vol.1, No.1, 1954 January, pp.4-6.

(1954) Backus, J.W., and Herrick H., IBM 701 Speedcoding and other automatic programming systems, In Proc. Symp. on Automatic Programming for Digital Computer, Washington DC, The Office of Naval Research, May 1954, pp.106-113.

(1954)* [Backus, J.W., Herrick, H., and Ziller, I.]#, Preliminary Report: Specifications for the IBM Mathematical FORmula TRANslating Systems, FORTRAN, IBM Corp. Programming Research Group, Applied Science Division, 1954 November 10, 29 pp., (mimeograph).

(1956) Backus, J.W., Beeber, R.J., Best, S., Goldberg, R., Haibt, L.M., Herrick, H.L., Nelson, R.A., Sayre, D., Sheridan, P.B., Stern, H., Ziller, I., Hughes, R.A., and Nutt, R., Programmer's Reference Manual, The FORTRAN Automatic Coding system for the IBM 704 EDPM, IBM Corp., New York, 1956 October 15.

(1957) [Mitchell, Grace E.]#, Programmer's Primer for FORTRAN Automatic Coding System for the IBM 704, IBM Corp., New York, 1957, Form No.32-0306.

(1957)* anon., Preliminary Operator's Manual for the FORTRAN Automatic Coding System for the IBM 704 EDPM, IBM Corp. Programming Research Dept., New York, 1957 April 8, pp.37.
[from title page]: This manual describes the use of FORTRAN 4-1-4-1.

(1957)* Backus, J.W., Beeber, R.J., Best, S., Goldberg, R., Haibt, L.M., Herrick, J.L., Nelson, R.A., Sayre, D., Sheridan, P.B., Stern, H., Ziller, I., Hughes, R.A., and Nutt, R., The FORTRAN automatic coding system, In Proc. Western JCC, Los Angeles CA, AFIPS(?), 1957.

(1957) anon., Proposed Specifications for FORTRAN II for the 704, Unpublished memorandum, IBM Corp. Programming Research Dept., 1957 September 25.

(1958)* Borden, B.C., FORTRANSIT A Universal Automatic Coding

--------------------

* An asterisk after the date indicates that a copy of this paper is on file.
# The authors are not listed on the document itself, but are remembered to be as listed here.

System for the IBM 650, Canadian Conference for Computing and Data Processing, 1958 June 9-10, Univ. of Toronto, pp.349-359.

Apparently this is the only formal paper on the FORTRANSIT language which was published. Regrettably it is not by one of the developers of the language and thus is a view of the cascading processor from a user's point of view rather than being a technical exposition of the "internals" of the FORTRANSIT system. Reported in this paper are three versions of the processor, two of which were for the basic machine and a system with index registers and floating point respectively, while a third version for a system supporting magnetic tapes was expected in 1958 July.

(1958) Bemer, R.W., and Hemmes, D.A., Computer Language Compatability thru (sic) Multi-Level Processors, 13th Nat'l Mtg. ACM, Univ. of Illinois, 1958 June 11-13, paper no. 31.

The Preprints for this conference did not include an extended abstract for this talk and thus the information below is taken from the meeting announcement.

This paper discusses the philosophy of maintaining language compatability through the use of cascading from one language to another using source language to source language translators. As examples, the authors presented FORTRANSIT and XTRAN as typifying this procedure. One of the advantages of this technique is the "... movement of error detection toward the earliest level for each type, thus allowing earlier decisions to stop processing."

(1958)* Backus, J.W., Automatic programming: properties and performance of FORTRAN systems I and II, Proc. Symp. on the Mechanisation of Thought Processes, Teddington, Middx, England, The National Physical Laboratory, Nov. 1958.

Written at a distance of one year after the delivery of the first FORTRAN processor for the 704, this paper is significant in its presentation of FORTRAN as an "automatic programming system" in the environment of a symposium of the mechanisation of thought processes. Other attendees at the meeting included Jan Garwick (Norway), John McCarthy (USA), Grace Murray Hopper (USA) and Christopher Strachey (GB), each of whom commented on the presentation by Backus. Obviously Garwick was much more interested in telling the audience of developments by Ole-Johan Dahl while McCarthy (the author of LISP) praised FORTRAN for its ability to express "... quite lengthy algebraic expressions ..." and the implementation of separate compilation of subroutines (presumably in FORTRAN II). Hopper states that "... there is a lack of understanding of the systemisation (sic) of Fortran " and asks Backus to emphasize that Fortran does more than just the "housekeeping" for the programmer.

(1958?)* Ziller, I., Description of Source Language Additions to the FORTRAN II System, Unpublished memorandum, Programming Research, IBM Corp., undated, 12pp.

This is the document which proposes the extensions to

FORTRAN II to create FORTRAN III which was a very short-lived system. The fundamental addition was the allowance of symbolic statements intermixed with FORTRAN statements.

(1959)* Sheridan, P.B., The arithmetic translator-compiler of the IBM FORTRAN automatic coding system, CACM, Vol.2, No.2, 1959 February, pp.9-21.

[From the introduction]: The present paper describes, in formal terms, the steps in translation employed by the FORTRAN arithmetic translator in converting FORTRAN formulas into 704 assembly code. The steps are described in about the order in which they are actually taken during translation.

(1959)* Mitchell, Grace E., The 704 FORTRAN II Automatic Coding System, Research Report RC-136, IBM Research Center, Yorktown Heights NY, 1959 Sept. 4, pp.13.

[Abstract]: This paper discusses the addition made in the FORTRAN I translator to produce the FORTRAN II translator. The new source language statements, debugging facilities and loader are described.

(1959)* Rosen,S., Goldberg,I.B., ALTAC, the TRANSAC Algebraic Translator, Preprints, ACM 14th. Natl. Mtg., MIT Cambridge MA, 1959 Sept. 1-3, 3pp.

ALTAC was possibly the first implementation of a FORTRAN-like language on a non-IBM machine. Like FORTRANSIT, ALTAC was targetted to an already existing language named TAC and like FORTRAN III (see Ziller 1958?) was capable of including TAC language embedded in the source program. ALTAC extended FORTRAN in several ways including compound statements (what in 1982 would be termed multi-statement lines) and IF statements that mirrored the ALGOL-like conditional statement.

(1960)* Ferguson, D.E., Input-Output Buffering and Fortran, JACM, Vol.7, No.1, 1960 January, pp.1-9.

This paper deals with a method which wa used successfully at the University of California, Los Angeles for the reduction of up to 40 percent in the running time for FORTRAN routines, and is an example of the typical "improvement" made by several installations on the original FORTRAN implementation.

(1960)* Gelernter, H., Hansen, J.R., and Gerberick, C.L., A Fortran-Compiled List Processing Language, JACM, Vol.7, No.2, 1960 April, pp.87-101.

This paper, apart from its own intrinsic merit, also has the distinction of being the first FORTRAN related article which was reviewed in Computing Reviews (No. 0142 in Vol.1, No.4, 1960).

[From the Abstract]: A compiled computer language for the manipulation of symbolic expressions organized in storage as Newell-Shaw-Simon lists has been developed as a tool to make more convenient the task of programming the simulation of a geometry theorem-proving machine on the IBM 704 high-speed electronic digital computer. Statements in the language are written in the usual Fortran notation but with a large set of special list-processing functions appended to the standard Fortran library.

(1960)* Blatt, J.M. Comments from a FORTRAN user, CACM, Vol.3, No.9, 1960 September, pp.501-505.

The thesis that compilers should be tailored to two classes of users, the occasional user with short problems, and the experienced user with large problems, is developed at some length with particular reference to FORTRAN. The author feels that compilers suitable for the experienced user are not generally available and points out some of the requirements of such a compiler. Specifically, he outlines the requirements of the manual associated with such a compiling routine [sic], the need for substantially instantaneous compilation, and the desirability of including more "machine-like" commands. There are discussions of ways to allow the advanced programmer latitude in the allocation of storage space as well as the conflicting requirements of instantaneous compilation and sensible code checking features. An appendix containing minor complaints against FORTRAN is added. The article is followed by some editorial comments in amplification of the paper.

John R. Pasta, Kensington MD
Reprinted by permission, Computing Reviews No.632

(1961)* McCracken, D.D., A guide to FORTRAN programming, John Wiley and Sons, New York NY, 1961, 88 pp.

This is the first non-IBM book on FORTRAN which was published by a commercial publishing house (c.f. Organick (1963) below). Gotlieb in his review in Computing Reviews (Vol.3, No.1, Rev. 1421, 1962 January, p. 22) states: "The are versions of FORTRAN for the IBM 650, 1620, 704, 709, 7090, and for the Honeywell 800, the Philco ALTAC, and the Control Data 1604. Since each version has its own description this latest work might seem redundant but it does have some definite advantages."

It is interesting to note that five years later Computing Reviews refrained from soliciting formal reviews of FORTRAN texts due to their "proliferation" and resorted instead to merely publishing an extract from the author's introduction!

(1961) Melkanoff, M.A., Nodvik, J.S., Saxon, D.S., and Cantor, D.G., A FORTRAN program for elastic scattering analyses with the nuclear optical model, Univ. Calif. Press, Berkeley and Los Angeles, CA, 1961, 116 pp.

This is the first FORTRAN program which appears in Computing Reviews (No. 1188 in Vol.2, No.6, 1961) other than the FORTRAN List Processing Language paper by Gelernter, et al (1960). The program was available on the IBM 704 and 709 and was the "culmination of many years of research."

(1961)* Rosen, S., ALTAC, FORTRAN and Compatability, Preprints, ACM 16th Nat'l. Conf., 1961, pp.2B.2(1)-(4).

This is not truly a full paper but instead is a summary of the paper presented at the 1961 ACM National Conference. It discusses the concepts of developing universal or common languages "... to permit the user to make the transition from one computer to another without the necessity of a complete reprogramming job."

(1961)* Bemer, R.W., Survey of Modern Programming Techniques, The Computer Bulletin, Vol.4, No.4, British Comp. Soc., 1961 March.

Like so many other papers with a title which includes the term "survey" this one covers a wide diversity of topics, many of which are surprisingly modern even 2[illegible] years later. The significant element of this paper is the statement by the author (then a Manager of a group with language responsibilities) which reads (page 130):

> "I have enough faith in the eventual future of ALGOL to have caused a program to be constructed which converts FORTRAN ... into ... ALGOL. I have been asked ... why [?] ... The answer [is] ... that we wish to obsolete FORTRAN and scrap it, not perpetuate it. Its purpose has been served."

(1961)* Knuth, D.E., A History of Writing Compilers, Computers and Automation, Vol.11, 1962 December, pp.8-18.

While this transcript of a presentation given at the 1962 Annual ACM National Meeting deals with the general problem of compiler writing, it includes two references to FORTRAN which are interesting. Firstly, the author introduces the method of parenthesizing expressions by surrounding operators with back-to-back parentheses in quantities inversely proportional to the hierarchical power of the operator and ascribes this technique to "the first FORTRAN compiler". Presumably, since the auhtor (at that time) was more familiar with the IBM 650 than other machines, he intended to imply the "first IBM 650 FORTRAN compiler" as is shown below in Lee (1981).

Referring to optimization of the object code produced from an arithmetic analyzer, the author states parenthetically:

> "The first FORTRAN compiler ... took fairly great care to produce efficient code, although the methods were quite painful."

Again the reference to the "first FORTRAN compiler" is confusing since the IBM 650 compiler (FORTRANSIT) did not include any optimization!

(1962)* Leeson, D.N., and Dimitry, D.L., Basic Programming Concepts and the IBM 1620 Computer, Holt, Reinhart and Winston, Inc., New York, 1962, 368pp.

Although there is only a superficial treatment of FORTRAN for the IBM 1620 with the main body of this text, Appendix

VIII is an in-depth explanation of the "internals" of the IBM 1620 FORTRAN compiler. The major section of the appendix deals with the tabular method of analyzing arithmetic expressions which is an implementation of the Samelson and Bauer algorithm. As an early description of a compiler in practical terms, this appendix is significant and surprisingly lucid.

(1962) anon., General Panel Discussion: Is the Unification ALGOL-COBOL, ALGOL-FORTRAN Possible? In Symbolic Languages in Data Processing, Gordon and Beach, Pub., New York, 1962, pp.833-49.

(1962) McMahon, J.T., ALGOL vs FORTRAN: a defense of the former, Datamation, Vol.8, No.4, 1962 April, pp.88-89.

The author contends that "... there is not one FORTRAN ... [which] is not machine independent ... There is no FORTRAN that is not defined for use on a particular digital computer ..." while "... one may write ... ALGOL without regard to the machine ..." He blaims this variance in FORTRAN on the lack of a rigid syntactic definition "... from the beginning ..." and further claims that "The crux of the matter lies in ALGOL being a problem statement language. Every FORTRAN is a machine oriented macro-language." He quotes Bemer (1961) as stating "... we wish to obsolete FORTRAN, not perpetuate it. Its purpose has been served."

(1962)* Rabinowitz, I.N., Report on the Algorithmic Language FORTRAN II, CACM, Vol.5, No.6, 1962 June, pp.327-37.

This paper is a "take-off" on the ALGOL 60 report by Backus et al. giving a syntactic description for FORTRAN II in terms of an extended BNF. [From the introduction]: The immediate impetus for the work was the existance of PSYCO, a compiler for ALGOL 60 on the CDC 1604 which requires a complete "syntax table" of the source language in order to do the translation. If such a table could be constructed for FORTRAN, then the same compiler could be used for both languages.

(1963)* [Pessin, F., et al]#, Proposed FORTRAN Extensions, Internal Report, Language Development Group, DS Programming Systems Planning, IBM Corp., 1963 January 15, 35pp., mimeograph.

This document is a proposal to add statements to FORTRAN IV in order to develop FORTRAN V. The additions include STRING statements, STRUCTURES, PICTURES, literals, set operations, NAME LIST and multiple entry subroutines. Revisions were proposed for EQUIVALENCE and DATA statements. This proposal was superceded by the development of The New Programming Language (NPL) which eventually became PL/I.

(1963)* Allen, J.J., Moore, D.P., and Rogoway, H.P., SHARE Internal FORTRAN Translator, [SIFT], Datamation, Vol.9, No.3, 1963 March, pp.43-46.

The SHARE Internal FORTRAN Translator (SIFT) is a FORTRAN program to translate FORTRAN II source programs into the FORTRAN IV language.

(1963) Entwisle, Doris R., Auto-primer in computer programming for the IBM 1620 in FORTRAN, Blaisdell Pub. Co., New York, 1963, 335 pp.

The book is a general text on FORTRAN, although it is oriented to the 1620 computer, referring to the characteristics of the 1620 version of the language. The format is that of a workbook, designed to be scribbled in and worked in; no loose sheets of scrap paper are needed. Problem answers are to be written in the margins and when the flaps are raised, correct answers appear next to the student's answers.

P. M. Sherman, Murray Hill, NJ
Reprinted by permission, Computing Reviews No.5669.

(1963)* Heising, W.P., FORTRAN, part of Yngve, V.H., and Sammet, J.E., Toward Better Documentation of Programming Languages, CACM, Vol.6, No.3, 1963 March, pp.85-86.

This short paper contains a brief history of the FORTRAN development effort within IBM and the steps that were taken to provide adequate documentation for the users. Recognizing that FORTRAN is available in several foreign languages, including at least one (French) in which the keywords have been changed into the native language, the author points out that the problems of updating manuals due to the location of an error are enormous but that the emergence of user groups such as SHARE are enabling such changes to be brought to the user's attention much more rapidly. "... users want a programming system, not merely a compiler. Although a compiler may be the largest single component of a programming system, it has probably received more than its proper share of attention in the literature relative to system components which perform more mundane but equally vital functions."

(1963)* McClelland, W.F., Survey of Programming Languages and Processors, CACM, Vol.6, No.3, 1963 March, pp.93-99.

This is the report of the ISO Technical Committee 97, Subcommittee 5 (Programming Languages) survey of programming languages conducted in May 1961. With respect to FORTRAN, it lists 55 languages which are related to FORTRAN together with the date of publication, the machine used, the size of the translator, minimum configuration as well as the date of the first run.

(1963) Organick, E.I., A FORTRAN Primer, Addison-Wesley Pub. Co., Reading MA, 1963.

The main feature of this FORTRAN teaching manual is that it does not require any mathematical background whatsoever. It can be sucessfully used by freshmen in business administration or even by high school students. All recent advances in FORTRAN, up to FORTRAN IV, are covered.

FORTRAN is not a perfect mechanical language, but it is adequate for most mathematical and data reduction processes. It is quite entrenched, to the point of being considered "the dominant language of our civilization". Other languages have been proposed, all more polished, but none having any margin of superiority which would justify abandoning FORTRAN; so it is predictable that FORTRAN will continue being "the" mechanical language for a good many years.

This excellent manual, that both instructors and students like, helps make this indispensible tool easily available to large masses of occasional computer users other than professional programmers.

L. A. Lombardi, Cambridge, MA
Reprinted by permission, Computing Reviews No.5042

In correspondence with Organick, he has pointed out that this book was originally published by the University of Houston in 1961, thus making that edition the first non-supplier manual. The University of Houston edition was also accompanied by a booklet of drill exercises and examples also dated 1961 (November).

(1963) Pyle, I.C., Dialects of FORTRAN, CACM, Vol.6, No.8, 1963 August, pp.462-467.

Three dialects of FORTRAN II are compared with that language. The dialects are FORTRAN IV, the language of the S1 compiler for STRETCH, and the Atlas FORTRAN. the comparison is made with respect to 9 features of FORTRAN II that are dropped, and 21 different features that are added. In each case the feature is detailed.

T. E. Kurtz, Hanover, NH
Reprinted with permission, Computing Reviews No.5044.

(1963) Ayers, J.A., Recursive Programming in FORTRAN II, CACM, Vol.6, No.11, 1963 November, pp.667-68.

(1963)* Larner, R., Design of an Integrated Programming and Operating System, Part IV: The System's FORTRAN Compiler, IBM Syst. J., Vol.2, 1963, pp.311-321.

[from the introductory paragraph]: This paper is devoted to the [IBM] 7090/94 version of the system's FORTRAN compiler ... [in which the] design of the compiler was substantially simplified. This ... permitted more attention to other design problems, in particular:

-- generation of optimal object program code...
-- preservation of modularity in the compiler so that subsequent ... improvements could be readily accomodated.
-- attainment of higher translation speeds

(1964)* Rosen, S., Programming Systems and Languages, Proc. SJCC, AFIPS, 1964, pp.1-15.

The author reviews the history of programming languages and their associated processors up to 1964, and ascribes a number of firsts to the pioneers of the 1950's. With respect to FORTRAN, Rosen says:

> "Fortran is in many ways the most important and most impressive development in the early history of automatic programming."

The caveat on this statement is the phrase "early history"; the author later in this paper gives opinions of why Fortran should not be considered a universal language (in comparison to ALGOL) and why the academic community (ACM) ignored its existence.

(1964) Pyle, I.C., Implementation of FORTRAN on ATLAS, In Wegner, P., Introduction to System Programming, Proc. Symp., London School of Economics, Academic Press, New York, 1964, pp.86-100.

This is a general description of the ATLAS FORTRAN compiler [which was then] in the process of construction. The compiler itself is written in FORTRAN to make it, as much as possible, machine-independent. There is a detailed discussion of the "bootstrapping" involved in writing a compiler in the source language itself. The prototype copy is to be made available to the ATLAS via the IBM 7090.

Certain extensions are made to FORTRAN which make it close to FORTRAN IV. Some details are given with regards to the optimization in the translating of expressions. A special feature is the distinction (to be implemented at a later time) between "fast" and "slow" subscripts.

After an outline of the procedure for translating a source routine and its breakdown into sections, there follows a sample of the text of the compiler, which is a routine for collecting an unsigned integer.

M. Shirmat, Calgary, Alberta, Canada

Reprinted by permission, Computing Reviews No.6301.

(1964) Wegner, P., FORTRAN, ALGOL and COBOL, In Wegner, P., Introduction to System Programming, Proc. Symp., London School of Economics, Academic Press, New York, 1964, pp.20-37.

This article attempts to give something of the flavor of the three languages, chiefly by discussing examples from each. Some readers may find an occasional remark slightly irritating. For example, the use of := instead of = is cited as an example that "COBOL is a more precise and consistent language than FORTRAN."

H. G. Rice, Santa Monica CA

Reprinted by permission, Computing Reviews No.6674.

(1964) Pyle, I.C., An Outline of FORTRAN, In Wegner, P., Introduction to System Programming, Proc. Symp., London School of Economics, Academic Press, New York, 1964, pp.20-37.

Most of the components of FORTRAN are described in this article, with an emphasis on use and intuitive understanding rather than syntactic correctness and precision. The main subject is FORTRAN II, although features of the "new dialects" received some attention.

H. G. Rice, Santa Monica, CA
Reprinted by permission, Computing Reviews No.6675.

(1964) McCracken, D.D. and Dorn, W.S., Numerical Methods and FORTRAN Programming, John Wiley and Sons, New York, 1964, 457pp.

[from the flysheet]: ... this book offers an integrated treatment of both numerical techniques and computer programming. While the fundamentals of FORTRAN are introduced in separate chapters, the two topics are developed together throughout the book.

It is thought that this was the first book which combined the teaching of these two topics, though several others followed closely behind it. Even so, the technique of intertwining the two topics was unique. Later texts were criticized for containing the topics in two separate parts essentially duplicating the FORTRAN Manufacturer's manuals in one part. At a time when such manuals were provide freely to users, this was thought of as a disadvantage.

(1964) Fowler, M.E., and MacMasters, J.A., A FORTRAN Program for Polynomial Manipulation, IBM Corp., Data Processing Division, Tech. Rep. TR-24.012, Kingston NY, 1964 March.

(1964) Taylor, R., and Harragan, D.A., The FORTRAN system for ORION, Comput. J., Vol.7, No.2, 1964 July, pp.114-116.

(1964)* Backus, J.W., and Heising, W.P., FORTRAN, IEEE Trans. on Electronic Computers, EC-13, No.4, August 1964, pp.382-385.

[from the summary]: The fundamental concepts of FORTRAN, the most widely used high-level, scientific programming language, are set forth and the significant characteristics are described in historical order from inception ... in 1954 to [1964] ... The basic problem of how to get high quality programming from an-easy-to-write high-level language is emphasized.

Looking back after 10 years, Backus recalls the objectives of the FORTRAN effort and many of the frustrations which accompanied the development of the first processor. Of particular note is the commentary on the time taken during compilation to ensure the production of optimum code, time which is often fruitlessly wasted on simple programs. Mention is made of the technique of flow analysis used in the first compiler which was based on a Monte Carlo analysis of the frequency of execution of sections of the program. Regrettably (then and ever since) no documentation of this technique is provided.

(1964)* Heising, W.P., FORTRAN, Compatability and Standardization, Datamation, Vol.10, No.8, 1964 August, pp.24-25.

[from the preface]: "... the article is intended not as a progress report on the work of [the ASA] committee, but rather as some conclusions -- based on two years work involving most of the principal American computer manufacturers and users groups -- concerning FORTRAN compatability ... and what standards can and cannot be accomplished in the FORTRAN area."

(1964)* Oswald, H., The Various FORTRANs, Datamation, Vol.10, No.8, 1964 August, pp.25-29.

The actual purpose of this article is to introduce a device known as the Fortran infograph which provided the programmer with a means of looking up the variations in the language with respect to various machines. The infograph was similar to a telephone directory desk device in which the cover was imprinted with the statement types and a sliding pointer enabled the user to select the information he desired. On depressing a latch, the infograph then opened to reveal the required information.

(1964)* Fimple, M.D., FORTRAN vs COBOL, Datamation, Vol.10, No.8, 1964 August, pp.34, 39-40.

Whereas the majority of comparisons between FORTRAN and other languages (see McMahon (1962) above) tend to be castigations of FORTRAN, this article shows how the language is superior for "business DP" to other languages such as Commercial Translator or COBOL. Documented here is a subjective experiment using a typical program (constructed for the purpose of the experiment) as a result of which it is concluded that FORTRAN is superior. This is based on comparisons of ease of learning, size of program, documentation, diagnostics, size of object code and running times.

(1964)* Heising, W.P., History and Summary of FORTRAN Standardization Development for the ASA, CACM, Vol.7, No.10, 1964 October, p. 590.

This report is an introduction to the draft proposed American National Standard for FORTRAN (see next listing).

(1964) anon., FORTRAN vs Basic FORTRAN - A Programming Language for Information Processing on Automatic Data Processing Systems, CACM, Vol.7, No.10, 1964 October, pp.591-625.

This paper is actually the draft of the proposed FORTRAN standards which was eventually produced as the 1966 American National Standards X3.9-1966 and X3.10-1966.

(1965)* Haines, L.H., Serial Compilation and the 1401 FORTRAN Compiler, IBM Systems Journal, Vol.4, No.1, 1965, pp.73-80.

To our knowledge, the 1401 compiler for FORTRAN was the only one which passed the compiler through the program in memory. In essence the compiler consisted of 63 phases each of which modified the source program in situ to generate the object text.

(1965) McCracken, D.D., A Guide to FORTRAN IV Programming, John Wiley and Sons, New York, 1965, 151 pp.

(1965) Pollack, S.V., A Guide to FORTRAN IV, Columbia Univ. Press, New York, 1965, 260 pp.

(1965) Junker, J.P., and Boward, G.R., COBOL vs FORTRAN: A Sequel, Datamation, Vol.11, No.4, 1965 April, pp.65-67.

(1965) McCracken, D.D., How to Tell If It's FORTRAN IV, Datamation, Vol.11, No.10, 1965 October, pp.38-41.

(1965) Rosen, S., Spurgeon, R.A., and Donnelly, J.K., PUFFT - The Purdue University Fast FORTRAN Translator, CACM, Vol.8, No.11, 1965 November, pp.661-66.

> This is one of the most elegant and successful of the current generation of compile-and-go systems to batch-process large numbers of small-to-medium FORTRAN IV jobs. Written for the IBM 7094, it achieves, for such jobs, a 7- to 10-fold gain in speed over IBJOB with version 13 IBFTC processor. Programs compiled by PUFFT result in less carefully optimized code, however, so that the longer-running jobs are still run in the IBM system. Nevertheless, source programs for the two systems are highly compatible. Thus, PUFFT serves also as a training system and for debugging of larger programs. A special feature of PUFFT is the diagnostic error message routine. By means of an elaborate encoding scheme, several hundred different error messages at both compile and execute times, are available through slightly more than 500 words of core store. All such features recommend PUFFT in student as well as in a research environment. This is an important and well-written paper. All aspects of the self-contained system are clearly outlined, and many ideas of value to authors of compile-and-go systems are presented.

(1965) Sakoda, J.M., DYSTAL Manual - Dynamic Storage Allocation Language in FORTRAN, Dept. of Sociology and Anthropology, Brown U., 1965. Also in "Symbol Manipulation Languages and Techniques", Proc. IFIP Working Conf., Pisa Italy, North Holland Pub. Co., Amsterdam, 1968.

(1966) Wright, D.L., A Comparison of the FORTRAN Language Implementation for Several Computers, CACM, Vol.9, No.2, 1966 February, pp.77-79.

(1966)* Editor's Note to Review No. 10,461, Computing Reviews, Vol.7, No.5, Sept.-Oct. 1966, p.413.

> "In view of the extensive proliferation of textbooks on FORTRAN programming it has been decided that in lieu of full reviews, such books will be cited with a brief indication of intended audience and special features as seen by the author."
>
> Reprinted by permission, Computing Reviews.

(1966)* anon., The man behind FORTRAN, Computing Report, IBM

Corp., Vol.II, No.4, 1966 November, pp.7-10, 19.

This is not a technical paper, but rather is a question-answer report of an interview with John Backus containing some personal reminiscences about the period of the development of FORTRAN, 1954-57.

(1967)* Shantz, P.W., et al., WATFOR - The University of Waterloo FORTRAN IV Compiler, CACM, Vol.10, No.1, 1967 January, pp.41-44.

(1967)* Moulton, P.G., and Muller, M.E., DITRAN - A Compiler Emphasizing Diagnostics, CACM, Vol.10., No.1, 1967 January, pp.45-52.

(1967) Lee, John A.N., "The Anatomy of a Compiler", Reinhold Pub. Co., New York, 1967, 275 pp.

This text covers the development of a compiler for a variant of FORTRAN and contains a description of the techniques of parenthesizing technique used in the FORTRANSIT processor.

(1969)* anon., Clarification of Fortran Standards - Initial Progress, Comm. ACM, Vol.12, No.5, 1969 May, pp.289-194.

This is an initial report on the interpretation of 49 sections of the 1966 American National Standard Programming language FORTRAN (and Basic FORTRAN) and the correction of seven errors. In fact, this report only contains seven interpretations but sets the stage for a continuing process of publications of such interpretations.

(1969) Lowry, E.S., and Medlock, C.W., Object Code Optimization, CACM, Vol.12, No.1, 1969 January, pp.13-22.

(1969)* Sammet, J.E., Programming Languages: History and Fundamentals, Prentice-Hall Pub. Co., Englewood Cliffs NJ, 1969, pp.143-172, 302-304.

The section of this text which deals with FORTRAN is probably the first complete review of the language, its history and its contents which was published in a comprehensive volume dealing with the whole set of viable languages in the industry in 1969. The bibliography was used as the starting point of this bibliography although some of the references which dealt with applications rather than language were omitted.

(1970)* Cocke, J., and Schwartz, J.T., Programming Languages and their Compilers, Preliminary Notes, 2nd revised edition, Courant Inst., New York, April 1970, pp.510-515.

These five pages in the otherwise unpublished manuscript contain a review of the techniques of optimization that were used in the original FORTRAN compiler, and based on assistance from Sheldon Best, are a more detailed account of the processes used than were published previously.

(1970) Cress, P., Dirksen, P., and Graham, J.W., FORTRAN IV with WATFOR and WATFIV, Prentice Hall Pub. Co., Englewood Cliffs, NJ,

1970.

(1971)* Bright, H.S., FORTRAN comes to Westinghouse-Bettis, Computers and Automation, Vol.20, No.11, 1971, pp.17-18.

This is an anecdote regarding a strange and wonderful package that arrived unannounced at Westinghouse-Bettis and which turned out to be a binary deck of the original FORTRAN processor for the IBM 704. Included is a copy of the first program run and the output (including the first error message.)

(1971) Engel, F., et al, Clarification of FORTRAN Standards - Second Report, CACM, Vol.14, No.10, 1971 October, pp.628-42.

c.f. Initial Progress report of 1969.

(1971) Knuth, D.E., An Empiricial Study of FORTRAN Programs, Software -- Practice and Experience, Vol.1, No.2, 1971, pp.105-133.

[from the summary]: A sample of programs written in FORTRAN by a wide variety of people for a wide variety of applications, was chosen 'at random' in an attempt to discover quantitatively 'what programmers really do'. Statistical results of this survey are presented ...

The major finding of this work is that between program written in an industrial environment and those in an academic setting are little different in their statistical use of particular statement types. Assignment statements, IF statements and Go-to statements led each list (in that order) with 68% of the assignment statements being strict replacement and 17% involving only one operator!

(1972) Engel, F., Future FORTRAN Development, SIGPLAN Notices, Vol.8, No.3, 1972 March.
Also in Honeywell Comput. J., Vol.6, No.4, 1972, pp.298-99.

(1972)* Tropp, H., (Ed), Transcript of a discussion held at the Hilton Hotel, San Francisco, during the March 1972 SHARE meeting.

The complete transcript of this discussion covers much more than FORTRAN, but there are several pages of very frank and open comments about the development of FORTRAN and some of the vexations of getting the system into the hands of users. Participants in the discussion include John Backus, Tom Steel, Jr., Frank Engel, Jr., Betty and George Ryckman, Frank Wagner, ?? Gautney, John Greenstadt, Harry Cantrell, Ted Dollata, Arnold Smith and Mort Bernstein.

(1974) Engel, F., Revise FORTRAN Standard?, Datamation, Vol.20, No.5, 1974 April, pp.164-69.

(1974)* Greenfield, M.N., FORTRAN - A History of a Pragmatic Language, unpublished report of talk given to ISO/TC97/SC5 FORTRAN/BASIC subcommittee, 1974 June 11, 16pp.

(1974)* Ryder, B.G., The PFORT Verifier, Software -- Pract. & Exp., Vol.4, 1974, pp.359-377.

[from the summary]: The PFORT Verifier is a program which checks a FORTRAN program ... for adherence to a large, carefully defined, portable subset of ANS FORTRAN ... The Verifier is itself written in PFORT ...

(1975)* Kernighan, B.W., RATFOR - A Preprocessor for a Rational FORTRAN, Software -- Pract. & Exp., Vol.5, 1975, pp.395-406.

[from the summary]: Although Fortran is not a pleasant language to use, it does have the advantages of universality and (usually) relative efficiency. The RATFOR language attempts to conceal the main deficiencies of Fortran while retaining its desirable qualities ... RATFOR is implemented as a preprocessor which translates this language into Fortran.

RATFOR is a prime example of the piggy-backing of other languages onto FORTRAN even though the authors seem to despise the original language! In many respects the objectives oyyf RATFOR were achieved in FORTRAN 77, though the preprocessor is still in active use in 1982.

(1975?) Engel, F., Jr., FORTRAN, in Encyclopedia of Computer Science, Belzer, Holzman & Kent (Eds), Vol.8, 1975?, pp.252-285.

(1976)* Greenfield, M.N., Background and Interpretation of the FORTRAN Draft Proposed Standard, unpublished report to ANSI Committee X3J3, 1976 February 9, 4pp.

(1976) anon., draft proposed ANS FORTRAN, BSR X3.9, X3J3/76, SIGPLAN Notices, ACM New York, Vol.11, No.3, 1976 March.

The complete issue of SIGPLAN Notices was taken up by this proposal which was eventually modified and became the 1978 ANSI Standard which is commonly known as FORTRAN 77.

(1977) Knuth, D.E., and Pardo, L.T., Early developments of programming languages, in Encyclopedia of Computer Science and Technology, Vol.7, Dekker, New York, 1977, pp.419-493. Also published in: "A History of Computing in the Twentieth Century", Metropolis, N., Howlett, J., and Rota, G-C., (Eds), Academic Press, New York, 1980, pp.197-273.

This history of programming languages differs from the other papers on the history of computing published in the same volume (Metropolis et al) in several ways. Whereas most of the other papers are personal recollections of events which led up to some specific development, or are reports of the "human side" of that history, the paper by Knuth and Pardo is a technological view of the development of programming languages from a distance. As a model of the type of technological reviews that should be conducted at this point in time over the recent history this must be regarded as a classic. That is not to say that one cannot find flaws with the presentation, but rather that other such reviews of the technology would be very acceptable. If there be a fault at all, it is the lack of continuity between languages in this presentation. The 20 languages presented are linked mainly by chronology and the authors' ability to show examples of the syntactic form through the use of a single algorithm. It would have made a vast improvement to have suggested or speculated on the parenthood of each language and to have attempted to construct a set of family trees for these languages. Perhaps this is a good topic for a follow-up paper.

The method presentation of the twenty languages is to describe the syntactic forms of the language, the machine for which it was constructed (since most pre-1950's languages were very machine dependent) and they show the possible program for a nonsense algorithm. The readers will find that the examples are the whole heart of the presentation and provide a vehicle for understanding of each language which endorses the views of several educators that languages can be taught by example (c.f. Wegner+). Starting with a discussion of pre-computer languages for the description of algorithms, the authors present brief descriptions of Plankalkul (Zuse, 1945), Flow diagrams (Goldstine and von Neumann, 1946), Composition (Curry, 1948), Short code (Mauchly et al, 1949), Intermediate Programming Language (Burks, 1950), Klammeransdrucke (Rutishauser, 1951), Formules (Bohme, 1951), Autocode (Glennie, 1952), A-2 (Hopper at al, 1953), Algebraaic Interpreter (Laning and Zierler, 1953), FORTRAN (Backus et al, 1954-57), Mark I Autocode (Brooker, 1954), PP-2 (Kamynin and Liubimskii, 1954), PP (Ershov, 1955), BACAIC (Grems and Proter, 1955), Kompiler 2 (Elsworth et al, 1955), ADES (Blum, 1956), IT (Perlis et al, 1956), Math-matic (Katz et al, 1956-58), and the language identified only by its US patent number 3,047,288 (Bauer and Samelson, 1956-58).

The summary to this paper is an excellent table of comparisons covering such topics as whether the language was actually implemented, its readability, the availability of control structures, the types of data structures included, the machine independence of the language and its impact on the field. There is no doubt in my mind that this paper should be required reading for all computer science students in addition to the modern text-books on comparative languages which start the comparison only as early as FORTRAN.

J.A.N. Lee, Blacksburg VA
Reprinted with permission, Computing Review No. XXXX.

--------------------

+ Wegner, P., Programming with Ada: An Introduction by Mean of Graduated Examples, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

(1978) Backus, J.W., Can programming be liberated from the von Neuman style? A functional style and its algebra of programs. CACM, Vol.21, No.8, August 1978, pp.613-641.

(1978) Brainerd, W., (Ed.), FORTRAN 77, CACM, Vol.21, No.10, 1978 October, pp.806-20.

This paper describes the second FORTRAN standard with the official title "American National Standard Programming Language FORTRAN, X3.9-1978" but more commonly known as FORTRAN 77. Included in the presentation is the many additional features of FORTRAN 77 with information as to why and how the standard was developed.

(1979) Backus, J.W., The history of FORTRAN I, II and III, Ann. Hist. Comput., Vol.1, No.1, 1979 July, pp.21-37.

This article summarizes the history of the development of FORTRAN I, II and III. The author, who was the leader of the groups which developed the first two compilers, explains the economic factors leading to the establishment of the FORTRAN project, its goals, and the mode of working of its implementations. The article makes it clear that the early FORTRAN efforts were efforts of compiler development rather than language design. The language was designed as the compiler was written and the compiler design was considered [to be] the hard job.

This lucidly written article is interesting not only for the facts presented about the history of FORTRAN (e.g., that efficiency of object code was more important in getting FORTRAN accepted than the design of the language) and the insight given into design of the language (e.g., that subscripts in a subscript variable were limited to three to increase compiler efficiency rather than because the IBM 704 has only three index registers), but also for its revelation of the mixture of clairvoyance, inventiveness, and naivete possessed by the implementation team (e.g., common expression elimination, the actual degree of optimization exhibited in the object code, and the feeling that debugging would all but [be] eliminated by the use of FORTRAN). The article is must reading for anyone considering language design today; it raises serious questions as to whether there is anything new under the sun. The kinds of things being said today are hauntingly reminiscent of the kinds of things quoted in the article as being said in "those" days.

D. Berry, Los Angeles CA,
Reprinted with permission, Computing Review 35,907.

(1979) Stegmann, C., Pathfinder, THINK, IBM Corp., 1979 July/August, pp.18-27.

An interview with John Backus on the 25th anniversary of the beginning of the FORTRAN project.

(1980) Backus, J.W., Programming in the 1950's - some personal

impressions. In "A History of Computing in the Twentieth Century," Metropolis, N., et al, Eds., Academic Press, New York, 1980, pp.125-135.

(1981) Hoare, C.A.R., The Emperor's Old Clothes, 1980 ACM Turing Award Lecture, CACM, Vol.24, No.2, 1981 February, p.77.

This reference is included as the latest reference to an anecdote which points a design flaw in the FORTRAN language which is much more apparent in 1981 than would have been considered in 1954. Hoare criticizes the lack of strong typing in FORTRAN and cites "... The story of the Mariner space rocket to Venus, lost because of the lack of compulsory declarations ...". Further research reveals two other references to this story which is explicitly stated by Horning [Horning, J., A Note on Program Reliability, ACM SIGSOFT, Software Engineering Notes, Vol.4, No.4, 1979 Oct., p.6]:

> "The first American Venus probe was lost due to a program fault caused [3] by the inadvertent substitution of a statement of the form
>
> DO 3 I = 1.3
>
> for one of the form
>
> DO 3 I = 1,3 "

where reference [3] is:
Meyers, G.J., Software Reliability: Principles and Practices, John Wiley and Sons, 1976, p.275.

(1981) Backus, J.W., The FORTRAN Session, In "The History of Programming Languages", Wexelblat, R.L., (Ed), Academic Press, New York, 1981, pp.25-74.

(1981)* Lee, J.A.N., (Ed)., Oral Interview with Florence Pessin

## Manufacturer's Language Reference Manuals

(1956) Backus, J.W., Beeber, R.J., Best, S., Goldberg, R., Haibt, L.M., Herrick, H.L., Nelson, R.A., Sayre, D., Sheridan, P.B., Stern, H., Ziller, I., Hughes, R.A., and Nutt, R., Programmer's Reference Manual, The FORTRAN Automatic Coding System for the IBM 704 EDPM, IBM Corp., New York, 1956 October 15.

(1957) [Mitchell, Grace E.]#, Programmer's Primer for FORTRAN Automatic Coding System for the IBM 704, IBM Corp., New York, 1957, Form No.32-0306.

(1958) anon., FORTRAN II for the IBM 704 Data Processing System, IBM Corp. Reference Manual, C28-6000, 1958.

(1961) anon, FORTRAN General Information Manual, IBM Corp., Data Processing Division, White Plains NY, F28-8074, 1961.

(1960) anon., IBM 709-7090 FORTRAN Monitor, IBM Corp. Manual, C28-6065, 1960.

(1962) anon., IBM 1620 FORTRAN Reference Manual, IBM Corp., Data Processing Division, White Plains NY, C26-5619-09, 1962.

(1964) anon., IBM Operating System/360: FORTRAN IV, IBM Corp., Data Processing Division, White Plains NY, C28-6515-2, 1964.

(1965) anon., Series 200 FORTRAN D Compiler, Honeywell Information Systems, File No.123.1305.001D.027, 1965 November.

(1966) anon., IBM 7090/7094 IBSYS Operating System - Version 13: FORTRAN IV Language, IBM Corp., Data Processing Division, White Plains NY, C28-6390-3, 1966 April.

(1966) anon., 3100/3200 Computer Systems Basic FORTRAN Reference Manual, Control Data Corp., Publ. No.60172000, 1966 July.

(1966)* anon., Time-Sharing FORTRAN Reference Manual, General Electric Information Systems Division IPC-206046A, 1966 August, Rev. 1966 October, pp.125.

(1968) anon., 3100/3200/3300/3500 Computer Systems FORTRAN Reference Manual, Control Data Corp., Publ. No.60057600C, 1968 November.

(1969) anon., UNIVAC 1108 FORTRAN V, Sperry Rand Corp., Publ. UP-4060, 1969.

## American and ISO Standards Documents

(1966) American Standard FORTRAN, American Standards Association X3.9-1966, Approved March 7, 1966.

(1966) American Standard Basic FORTRAN, American Standards Association X3.10-1966, Approved March 7, 1966.

(1978) American National Standard programming language FORTRAN, ANSI X3.9-1978, approved April 3, 1978.
Revision of ANSI X3.9-1966; X3.10-1966 on Basic FORTRAN was withdrawn.

Also worthy of note is the newsletter of FORTRAN enthusiasts, published on an irregular basis as: FOR-WORD, Meissner, L.P., (Ed), Lawrence Berkeley Lab., Berkeley CA.

1982
National
Computer
Conference
Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

5780

Reply to: John A. N. Lee
Pioneer Day Chairman

1981 December 16

Calvin C. Gotlieb
Dept. of Computer Science
University of Toronto
Toronto Ont M5S 1A7
CANADA

Dear Kelly,

In researching the history of FORTRAN for the 1982 Pioneer Day, I have come across a reference from Don Knuth to a paper which was published in 1958 on the topic of FORTRANSIT. This was published in the Proceedings of the Canadian Conf. Computing Data Process. (excuse the abbrevs but I'm not totally sure how to unravel them!) which our library has been unable to locate through inter-library loan. The exact reference is:

B.C. Borden, FORTRANSIT, a universal automatic coding system, Proc. Can. Conf. Comput. Data Process., Univ. of Toronto, 1958, pp.349-359.

Would it be possible for you to locate someone at Toronto who could obtain a copy of this article for me? We would be very willing to pay copying and mailing costs. It would appear from discussions with Bob Bemer and others of his crew, that this is the only paper which was ever published in a formal manner on this particular processor.

Many thanks,
Yours sincerely,

Jan

John A. N. Lee
Pioneer Day Chairman

xc: Bemer

Table 4.9. Reference List of Computer Models Surveyed

| No. | Computer name | Date introduced | CM |
|---|---|---|---|
| 1 | Harvard Mark I | 1944 | 1 |
| 2 | Bell Lab Computer Model IV | Mar. 1945 | |
| 3 | Eniac | 1946 | |
| 4 | Bell Lab Computer Model V | Late 1947 | |
| 5 | Harvard Mark II | Sept. 1948 | |
| 6 | Binac | Aug. 1949 | 2 |
| 7 | IBM CPC | 1949 | 1 |
| 8 | Bell Lab Computer Model III | 1949 | |
| 9 | SEAC | May 1950 | |
| 10 | Whirlwind I | Dec. 1950 | |
| 11 | Univac 1101 Era 1101 | Dec. 1950 | 2 |
| 12 | IBM 607 | 1950 | 1 |
| 13 | Avdiac | 1950 | |
| 14 | Adec | Jan. 1951 | |
| 15 | Burroughs Calculator | Jan. 1951 | 3 |
| 16 | SWAC | Mar. 1951 | |
| 17 | Univac I | Mar. 1951 | 2 |
| 18 | ONR Relay Computer | May 1951 | |
| | Fairchild Computer | June 1951 | |
| 20 | National 102 | Jan. 1952 | 4 |
| 21 | IAS | Mar. 1952 | |
| 22 | Maniac I | Mar. 1952 | |
| 23 | Ordvac | Mar. 1952 | |
| 24 | Edvac | Apr. 1952 | |
| 25 | Teleregister Special Purpose Digital Data Handling | June 1952 | |
| 26 | Illiac | Sept. 1952 | |
| 27 | Elcom 100 | Dec. 1952 | 13 |
| 28 | Harvard Mark IV | 1952 | |
| 29 | Alwac II | Feb. 1953 | 14 |
| 30 | Logistics Era | Mar. 1953 | |
| 31 | Oarac | Apr. 1953 | 10 |
| 32 | ABC | May 1953 | |
| 33 | Raydac | July 1953 | |
| 34 | Whirlwind II | July 1953 | |
| 35 | National 102A | Summer 1953 | 4 |
| 36 | Consolidated Eng. Corp. Model 36-101 | Summer 1953 | |
| 37 | Jaincomp C | Aug. 1953 | |
| 38 | Flac | Sept. 1953 | 6 |
| 39 | Oracle | Sept. 1953 | |
| 40 | Univac 1103 | Sept. 1953 | 2 |
| 41 | Univac 1102 | Dec. 1953 | 2 |
| 42 | Udec I | Dec. 1953 | 3 |
| 43 | NCR 107 | 1953 | 4 |
| 44 | Miniac | Dec. 1953 | |
| 45 | IBM 701 | 1953 | 1 |
| 46 | IBM 604 | 1953 | 1 |
| 47 | AN/UJQ-2(YA-1) | 1953 | |
| 48 | Johnniac | Mar. 1954 | |
| 49 | Dyseac | Apr. 1954 | |
| 50 | Elcom 120 | May 1954 | 13 |
| 51 | Circle | June 1954 | |
| 52 | Burroughs 204 & 205 | July 1954 | 3 |
| 53 | Modac 5014 | July 1954 | |
| 54 | Ordfiac | July 1954 | 13 |
| 55 | Datatron | Aug. 1954 | 3 |
| 56 | Modac 404 | Sept. 1954 | |
| 57 | Lincoln Memory Test | Dec. 1954 | |
| 58 | TIM II | Dec. 1954 | |
| 59 | Caldic | 1954 | |
| 60 | Univac 60 & 120 | Nov. 1954 | 2 |
| 61 | IBM 650 | Nov. 1954 | 1 |
| 62 | WISC | 1954 | |
| 63 | NCR 303 | 1954 | 4 |
| 64 | Mellon Inst Digital Computer | 1954 | |
| 65 | IBM 610 | 1954 | 1 |
| 66 | Alwac III | 1954 | 14 |
| 67 | IBM 702 | Feb. 1955 | 1 |
| 68 | Monrobot III | Feb. 1955 | 15 |
| 69 | Norc | Feb. 1955 | |
| 70 | Miniac II | Mar. 1955 | |
| 71 | Monrobot V | Mar. 1955 | 15 |
| 72 | Udec II | Oct. 1955 | 3 |
| 73 | RCA BIZMAC I & II | Nov. 1955 | 6 |
| 74 | Pennstac | Nov. 1955 | |
| 75 | Technitral 180 | 1955 | |
| 76 | National 102D | 1955 | 4 |
| 77 | Monrobot VI | 1955 | 15 |
| 78 | Modac 410 | 1955 | |
| 79 | Midac | 1955 | |
| 80 | Elcom 125 | 1955 | 13 |

Ref. Book: The International Computer Industry
Harman,A.J. Harv. U. 1971

Table 4.9 (continued)

| No. | Computer name | Date introduced | CM |
|---|---|---|---|
| 81 | Burroughs E101 | 1955 | 3 |
| 82 | Bendix G15 | Aug. 1955 | 12 |
| 83 | Alwac IIIE | Nov. 1955 | 14 |
| 84 | Readix | Feb. 1956 | |
| 85 | IBM 705, I, II | Mar. 1956 | 1 |
| 86 | Univac 1103A | Mar. 1956 | 2 |
| 87 | AF CRC | Apr. 1956 | 4 |
| 88 | Guidance Function | Apr. 1956 | |
| 89 | IBM 704 | Apr. 1956 | 1 |
| 90 | IBM 701 (CORE) | 1956 | 1 |
| 91 | Narac | July 1956 | |
| 92 | LGP 30 | Sept. 1956 | 16 |
| 93 | Modac 414 | Oct. 1956 | |
| 94 | Elecom 50 | 1956 | 13 |
| 95 | Udec III | Mar. 1957 | |
| 96 | George I | Sept. 1957 | |
| 97 | Univac File 0 | Sept. 1957 | 2 |
| 98 | Lincoln TXO | Fall 1957 | |
| 99 | Univac II | Nov. 1957 | 2 |
| 100 | IBM 705 III | Late 1957 | 1 |
| 101 | Teleregister Telefile | Late 1957 | |
| 102 | Recomp I | Late 1957 | |
| 103 | IBM 608 | 1957 | 1 |
| 104 | Mistic | 1957 | |
| 105 | Maniac | 1957 | |
| 106 | IBM 609 | 1957 | 1 |
| 107 | IBM 305 | Dec. 1957 | 1 |
| 108 | Corbin | 1957 | |
| 109 | Burroughs E103 | 1957 | 3 |
| 110 | AN/FSQ 7 & 8 | 1957 | |
| 111 | Alwac 880 | 1957 | 14 |
| 112 | Univac File I | Jan. 1958 | 2 |
| 113 | Lincoln CG24 | May 1958 | |
| 114 | IBM 709 | Aug. 1958 | 1 |
| 115 | Univac 1105 | Sept. 1958 | 2 |
| 116 | Lincoln TX2 | Fall 1958 | |
| 117 | Philco 2000-210 | Nov. 1958 | 5 |
| 118 | Recomp II | Dec. 1958 | |
| 119 | Burroughs 220 | Dec. 1958 | 3 |
| 120 | Mobidic | 1958–1960 | |
| 121 | Philco CXPO | 1958 | 5 |
| 122 | Monrobot IX | 1958 | 15 |
| 123 | GE 210 | June 1959 | 10 |
| 124 | Cyclone | July 1959 | |
| 125 | IBM 1620 | Oct. 1959 | 1 |
| 126 | NCR 304 | Nov. 1959 | 4 |
| 127 | IBM 7090 | Nov. 1959 | 1 |
| 128 | RCA 501 | Nov. 1959 | 6 |
| 129 | RW 300 | Nov. 1959 | |
| 130 | RPC 9000 | 1959 | 16 |
| 131 | Librascope Air Traffic | 1959 | 16 |
| 132 | Jukebox | 1959 | |
| 133 | Datamatic 1000 | 1959 | 8 |
| 134 | CCC Real Time | 1959 | |
| 135 | Burroughs E102 | 1959 | 3 |
| 136 | Burroughs D204 | 1959 | 3 |
| 137 | AN/TYK 6V BASICPAC | 1959 | |
| 138 | CDC 1604 | Jan. 1960 | 7 |
| 139 | Librascope 3000 | Jan. 1960 | 16 |
| 140 | Univac Solid State 80/91 1 | Jan. 1960 | 2 |
| 141 | Philco 2000-211 | Mar. 1960 | 5 |
| 142 | Univac Larc | May 1960 | 2 |
| 143 | Libratrol 500 | May 1960 | 16 |
| 144 | Monrobot XI | May 1960 | 15 |
| 145 | IBM 7070 | June 1960 | 1 |
| 146 | CDC 160 | July 1960 | 7 |
| 147 | IBM 1401 (Mag. Tape) | Sept. 1960 | 1 |
| 148 | AN/FSQ 31 & 32 | Sept. 1960 | 1 |
| 149 | Merlin | Sept. 1960 | |
| 150 | IBM 1401 (Card) | Sept. 1960 | 1 |
| 151 | Mobidic B | Fall 1960 | |
| 152 | RPC-4000 | Nov. 1960 | 16 |
| 153 | PDP-1 (MT) | Nov. 1960 | 9 |
| 154 | PDP-1 (PT) | Nov. 1960 | 9 |
| 155 | Packard Bell 250 (PT) | Dec. 1960 | |
| 156 | Honeywell 800 | Dec. 1960 | 8 |
| 157 | General Mills AD/ECW-57 | Dec. 1960 | |
| 158 | Philco 3000 | Late 1960 | 5 |
| 159 | Maniac III | Late 1960 | |
| 160 | Sylvania S9400 | Late 1960 | |
| 161 | Target Intercept | Late 1960 | 2 |
| 162 | Westinghouse Airborne | 1960 | |
| 163 | RCA 300 | 1960 | 6 |
| 164 | Mobidic CD & 7A AN/MYK | 1960 | |
| 165 | Litton C7000 | 1960 | |
| 166 | Libratrol 1000 | 1960 | 16 |
| 167 | GE 312 | 1960 | 10 |
| 168 | Diana | 1960 | |
| 169 | DE 60 | Feb. 1960 | |
| 170 | Burroughs D107 | 1960 | 3 |
| 171 | AN/USQ 20 | 1960 | 2 |
| 172 | AN/TYK 4V Compac | 1960 | 1 |

Table 4.9 (continued)

| No. | Computer name | Date introduced | CM |
|---|---|---|---|
| 173 | General Mills Apsac | Jan. 1961 | |
| 174 | Univac Solid State 80/90 II | Jan. 1961 | 2 |
| 175 | Bendix G20 & 21 | Feb. 1961 | 12 |
| 176 | RCA 301 | Feb. 1961 | 6 |
| 177 | BRLESC | Mar. 1961 | |
| 178 | GE 225 | Mar. 1961 | 10 |
| 179 | CCC-DDP-19 (Card) | May 1961 | |
| 180 | CCC-DDP-19 (MT) | May 1961 | |
| 181 | IBM Stretch (7030) | May 1961 | 1 |
| 182 | NCR 390 | May 1961 | 4 |
| 183 | Honeywell 290 | June 1961 | 8 |
| 184 | Recomp III | June 1961 | |
| 185 | CDC 160A | July 1961 | 7 |
| 186 | IBM 7080 | Aug. 1961 | 1 |
| 187 | RW 530 | Aug. 1961 | |
| 188 | IBM 7074 | Nov. 1961 | 1 |
| 189 | IBM 1410 | Nov. 1961 | 1 |
| 190 | Honeywell 400 | Dec. 1961 | 8 |
| 191 | Rice Univ. | Dec. 1961 | |
| 192 | Univac 490 | Dec. 1961 | 2 |
| 193 | AN/TYK 7V | 1961 | 1 |
| 194 | Univac 1206 | 1961 | 2 |
| 195 | Univac 1000 & 1020 | 1961 | 2 |
| 196 | ITT Bank Loan Process | 1961 | |
| 197 | George II | 1961 | |
| 198 | Oklahoma Univ. | Early 1962 | |
| 199 | NCR 315 | Jan. 1962 | 4 |
| 200 | NCR 315 CRAM | Jan. 1962 | 4 |
| 201 | Univac File II | Jan. 1962 | 2 |
| 202 | HRB-Singer Sema | Jan. 1962 | |
| 203 | Univac 1004 | Feb. 1962 | 2 |
| 204 | ASI 210 | Apr. 1962 | |
| 205 | Univac III | June 1962 | 2 |
| 206 | Burroughs B200 Series B270 & 280 | July 1962 | 3 |
| 207 | SDS 910 | Aug. 1962 | 11 |
| 208 | SDS 920 | Sept. 1962 | 11 |
| 209 | PDP-4 | Sept. 1962 | 9 |
| 210 | Univac 1107 | Oct. 1962 | 2 |
| 211 | IBM 7094 | Nov. 1962 | 1 |
| 212 | IBM 7072 | Nov. 1963 | 1 |
| 213 | IBM 1620 MOD III | Dec. 1962 | 1 |
| 214 | Burroughs B5000 | Dec. 1962 | 3 |
| 215 | ASI 420 | Dec. 1962 | |
| 216 | Burroughs B200 Series-Card System | Dec. 1962 | 3 |
| 217 | RW 400 (AN/FSQ 27) | 1962 | |
| 218 | CDC 3600 | June 1963 | |
| 219 | IBM 7040 | Apr. 1963 | 1 |
| 220 | IBM 7044 | July 1963 | 1 |
| 221 | RCA 601 | Jan. 1963 | 6 |
| 222 | Honeywell 1800 | Nov. 1963 | 8 |
| 223 | Philco 1000 Transac S1000 | June 1963 | 5 |
| 224 | Philco 2000-212 | Feb. 1963 | 5 |
| 225 | Librascope L3055 | Dec. 1963 | 16 |
| 226 | HW Electronics 15K | Feb. 1963 | |
| 227 | GE 215 | June 1963 | 10 |
| 228 | DDP-24 | June 1963 | 8 |
| 229 | CDC 3600 | June 1963 | 7 |
| 230 | UNIVAC 1050 | Sept. 1963 | 2 |
| 231 | UNIVAC 1004 | Sept. 1963 | 2 |
| 232 | PDP-5 | Oct. 1963 | 9 |
| 233 | IBM 1460 | Oct. 1963 | 1 |
| 234 | IBM 1440 | Nov. 1963 | 1 |
| 235 | Honewell 1400 | Dec. 1963 | 8 |
| 236 | ASI 2100 | Dec. 1963 | |
| 237 | SDS 9300 | Dec. 1963 | 11 |
| 238 | Burroughs 273 | Jan. 1964 | 3 |
| 239 | GE-235 | Jan. 1964 | 10 |
| 240 | IBM 7010 | Jan. 1964 | 1 |
| 241 | Burroughs B160-180 | Apr. 1964 | 3 |
| 242 | CDC 160G | Apr. 1964 | 7 |
| 243 | IBM 7094 II | Apr. 1964 | 1 |
| 244 | CDC 3200 | May 1964 | 7 |
| 245 | GE 415 | May 1964 | 10 |
| 246 | UNIVAC 1004 II, III | June 1964 | 2 |
| 247 | SDS-930 | June 1964 | 11 |
| 248 | GE 425 | June 1964 | 10 |
| 249 | GE 205 | July 1964 | 10 |
| 250 | Honeywell 200 | July 1964 | 8 |
| 251 | RCA 3301 | July 1964 | 6 |
| 252 | PDP-6 | July 1964 | 9 |
| 253 | CDC 6600 | Sept. 1964 | 7 |
| 254 | UNIVAC 418 | Sept. 1964 | 2 |
| 255 | NCR 315-100 | Nov. 1964 | 4 |
| 256 | GE 635 | Nov. 1964 | 10 |
| 257 | CDC 3400 | Nov. 1964 | 7 |
| 258 | Burroughs B5500 | Nov. 1964 | 3 |

Table 4.9 (continued)

| No. | Computer name | Date introduced | CM | No. | Computer name | Date introduced | CM |
|---|---|---|---|---|---|---|---|
| 259 | SDS 925 | Feb. 1965 | 11 | 286 | DDP-124 | Jan. 1966 | 8 |
| 260 | SDS 92 | Feb. 1965 | 11 | 287 | Honeywell 1200 | Jan. 1966 | 8 |
| 261 | CDC 3100 | Feb. 1965 | 7 | 288 | IBM 360/20 | Jan. 1966 | 1 |
| 262 | ASI 6020 | Mar. 1965 |  | 289 | UNIVAC 1005 II, III | Feb. 1966 | 2 |
| 263 | DDP-224 | Mar. 1965 | 8 | 290 | UNIVAC 1005 I | Feb. 1966 | 2 |
| 264 | DDP-116 | Apr. 1965 | 8 | 291 | Honeywell 120 | Feb. 1966 | 8 |
| 265 | GE 625 | Apr. 1965 | 10 | 292 | IBM 360/65 | Mar. 1966 | 1 |
| 266 | PDP-8 | Apr. 1965 | 9 | 293 | UNIVAC 494 | Mar. 1966 | 2 |
| 267 | PDP-7 | Apr. 1965 | 9 | 294 | SDS 940 | Apr. 1966 | 11 |
| 268 | IBM 360/40 | May 1965 | 1 | 295 | RCA Spectra 70/55 | July 1966 | 6 |
| 269 | IBM 360/30 | May 1965 | 1 | 296 | RCA Spectra 70/45 | July 1966 | 6 |
| 270 | NCR 315 RMC | July 1965 | 4 | 297 | RCA Spectra 70/35 | July 1966 | 6 |
| 271 | UNIVAC 1108 II | Aug. 1965 | 2 | 298 | Philco 200-213 | Oct. 1966 | 5 |
| 272 | GE 435 | Aug. 1965 | 10 | 299 | IBM 360/44 | Oct. 1966 | 1 |
| 273 | IBM 360/50 | Sept. 1965 | 1 | 300 | Honeywell 4200 | May 1967 | 8 |
| 274 | IBM 1130 | Sept. 1965 | 1 | 301 | SDS Sigma 7 | Dec. 1966 | 11 |
| 275 | NCR 590 | Sept. 1965 | 4 | 302 | PDP-8/S | Sept. 1966 | 9 |
| 276 | ASI 6240 | Oct. 1965 |  | 303 | PDP-9 | Dec. 1966 | 9 |
| 277 | UNIVAC 491 & 492 | Oct. 1965 | 2 | 304 | SDS Sigma 2 | Jan. 1967 | 11 |
| 278 | RCA Spectra 70/15 | Oct. 1965 | 6 | 305 | Burroughs B2500 | Feb. 1967 | 3 |
| 279 | Raytheon 520 | Oct. 1965 |  | 306 | Burroughs B3500 | May 1967 | 3 |
| 280 | IBM 360/75 | Nov. 1965 | 1 | 307 | UNIVAC 9300 | June 1967 | 2 |
| 281 | Honeywell 2200 | Dec. 1965 | 8 | 308 | UNIVAC 9200 | June 1967 | 2 |
| 282 | CDC 3800 | Dec. 1965 | 7 | 309 | Burroughs B6500 | Feb. 1967 | 3 |
| 283 | RCA Spectra 70/25 | Dec. 1965 | 6 | 310 | CDC 3500 | Sept. 1967 | 7 |
| 284 | Friden 6010 | Jan. 1966 |  |  |  |  |  |
| 285 | CDC 6400 | Jan. 1966 | 7 |  |  |  |  |

*Sources*: Reprinted with permission from articles by K. E. Knight in *Datamation*, Sept. 1966 and Jan. 1968, published and copyrighted by L. D. Thompson Publications, Inc., 35 Mason St., Greenwich, Conn. 06830 (except for the CM column, which was compiled mainly from appendixes in Knight, *A Study of Technological Innovation—The Evolution of Digital Computers*; and N. Hanover, *Economic Aspects of Computer Use*.

*Note*: Column CM lists the corporate manufacturer by number as follows:

| *Corporation Number* | *Name* |
|---|---|
| 1. | International Business Machine Corp. |
| 2. | Sperry Rand Corp. |
| 3. | Burroughs Corp. |
| 4. | National Cash Register Co. |
| 5. | Philco Corp. |
| 6. | Radio Corporation of America |
| 7. | Control Data Corp. |
| 8. | Honeywell |
| 9. | Digital Equipment Corp. |
| 10. | General Electric Co. |
| 11. | Scientific Data Systems |
| 12. | Bendix Corp. |
| 13. | Underwood Corp. |
| 14. | El-Tronics |
| 15. | Monroe-Calculating Machine Co. |
| 16. | General Precision Equipment Corp. |

Computer No. 218 is also listed as No. 229; it was not counted as two separate machines.

American Federation of Information Processing Societies, Inc. 210 Summit Avenue, Montvale, New Jersey 07645 201-391-9810

afips

ANNALS OF THE HISTORY OF COMPUTING

Editor-in-Chief, Bernard A. Galler
Assistant Editor-in-Chief, Nancy Stern

Reply to: Henry S. Tropp
Mathematics Department
Humboldt State University u
Arcata, CA 95521

January 15, 1982

Robert W. Bemer, Z4
Honeywell Information Systems
P.O. Box 6000
Phoenix, Arizona 85005

Dear Bob,

SOBEL is in the mail, Thank you.

Thanks for the copy of JAN's letter to Uta. As to your pioneering status, you are now a COT, whether you want to be or not. (COT = Certified Old Timer; no cracks about certified or certifiable).

Regards,

Hank

Henry S. Tropp

HST:jmb

**CONFERENCE CHAIRMAN**
Russell K. Brown, CDP
3420 Yoakum Boulevard
Houston, Texas 77006
713-524-3420

**PROGRAM CHAIRMAN**
Howard L. Morgan
Department of Decision Sciences
The Wharton School/CC
Philadelphia, PA 19104
215-243-7731

**PROGRAM VICE-CHAIRMAN**
Eric K. Clemons
Associate Professor of Decision Sciences
The Wharton School/CC
University of Pennsylvania
Philadelphia, PA 19104
215-243-7747

**PROFESSIONAL DEVELOPMENT SEMINARS**
Joseph Campisi
Aetna & Casualty
151 Farmington Avenue
Hartford, CT 06156
203-273-3611

**NCCC LIAISON**
Harvey Garner
University of Pennsylvania
Moore School D-2
Philadelphia, PA 19104
215-243-4787

**PIONEER DAY**
J.A.N. Lee
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
703-961-6931

**PLENARY SESSIONS**
Susan Rosenbaum
AT & T
[illegible]-52 C278
P.O. Box 3509
New Brunswick, NJ 08903
201-457-2664

**VICE-CHAIRMAN (PROGRAM ACTIVITIES)**
Robert Stirling
IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604
914-696-4251

**PROGRAM COMMITTEE**
Gene P. Altshuler
O. Peter Buneman
James E. Emery
Dennis Frailey
Robert Frankston
Randall Jensen
Beverly K. Kahn
Alan N. Smith
Amy D. Wohl

**AFIPS PROJECT MANAGER**
Sam Lippman
AFIPS
1815 N. Lynn Street
Suite 800
Arlington, VA 22209
703-558-3614

1982
National
Computer
Conference
Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

Reply to:

John A. N. Lee
Pioneer Day Chairman

1981 November 06

Dr. Uta C. Merzbach
Curator
Division of Mathematics
The National Museum of American History
Smithsonian Institution
Washington DC 20560

Dear Dr. Merzbach,

Many thanks for your response to me enquiry about the documents which were copied from the "SHARE-Verzuh" file; I am enclosing apersonal cheque for $4.80 to cover the costs of reproduction. I would be grateful if you could forward these materials to me at the addresses listed at the left.

The question of Bemer materials will have to await my receipt of further funding to finance a visit to Washington unless I can piggy-back a visit to the museum onto some other trip. One possibility is a one-day visit on December 9th next; if that date would be feasible then I will attempt to finance it in some way. In particular I am interested in the Bemer archives which relate to his work on the translator for FORTRANSIT. This subject has never been written up in any technical journal and thus needs to be reviewed for completeness in preparation for Pioneer Day and the succeeding publications.

I look forward to receiving the Verzuh materials and to visiting the museum again soon.

Yours sincerely,

John A. N. Lee

xc: D. McCracken, R. Bemer

encls

| | ASA<br>Basic | ASA | ASI<br>6000<br>Series | Burroughs<br>B5500 | Computer<br>Control<br>DDP-24,<br>116, 124, 224 | CDC<br>1604<br>3600<br>3800 | CDC<br>6000<br>Series | PDP-6 | EAI<br>8400 | GE<br>200<br>Series | GE<br>400<br>Series | GE<br>600<br>Series | Honey-<br>well<br>200 | Honey-<br>well<br>800<br>1800 | IBM<br>1401<br>1440<br>1460 | IBM<br>1410<br>7010 | IBM<br>7040<br>7044<br>(8K) | IBM<br>7040<br>7044<br>(16-32K) | IBM<br>7090<br>7094 | IBM<br>360<br>D level<br>E level | IBM<br>360<br>H level | NCR<br>315 | Philco<br>2000<br>Series | RCA<br>3301 | RCA<br>Spectra<br>70<br>Size A | RCA<br>Spectra<br>70<br>Size B | SDS<br>9300 | Univac<br>III | Univac<br>1107 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maximum statement number | 9999 | 99999 | 99999 | 99999 | 99999 | 99999 | 99999 | 99999 | 99999 | 32767 | 32767 | 99999 | 99999 | 32767 | 99999 | 99999 | 99999 | 99999 | 32767 | 99999 | 99999 | 99999 | 32767 | 99999 | 99999 | 99999 | 99999 | 32767 | 32767 |
| Maximum continuation cards | 5 | 19 | No limit | 9 | No limit | No limit | No limit | 19 | 19 | No limit | No limit | 19 | 9 | 19 | 9 | 9 | 4 | 9 | 19 | 19 | 19 | 19 | 19 | No limit | 19 | 19 | No limit | 9 | 19 |
| Specification statements must precede first executable statement | • | • | • | | • | • | • | • | | • | | • | • | • | | • | | | | • | | • | | | • | | | | |
| INTEGER constant, maximum digits | | | 7 | 11 | 7 | 14 | 18 | 11 | 5 | 6 | 7 | 11 | 20 | 13 | 20 | 20 | 11 | 11 | 11 | 10 | 10 | 11 | 12 | 7 | 10 | 10 | 7 | 6 | 11 |
| INTEGER maximum magnitude | | | $2^{23}-1$ | $2^{39}-1$ | $2^{23}-1$ | $2^{47}-1$ | $2^{59}-1$ | $2^{35}-1$ | $2^{18}-1$ | $2^{19}-1$ | $2^{23}-1$ | $2^{35}-1$ | $2^{119}-1$ | $2^{44}-1$ | $10^{20}-1$ | $10^{20}-1$ | $2^{35}-1$ | $2^{35}-1$ | $2^{35}-1$ | $2^{31}-1$ | $2^{31}-1$ | $10^{11}-1$ | $2^{39}-1$ | $10^{7}-1$ | $2^{31}-1$ | $2^{31}-1$ | $2^{23}-1$ | $10^{6}-1$ | $2^{35}-1$ |
| REAL constant, maximum digits | | | 11 | 11 | 7 | 11 | 15 | 8 | 7 | 9 | 8 | 9 | 20 | 12 | 20 | 18 | 9 | 9 | 9 | 7 | 7 | 12 | 11 | 8 | 7 | 7 | 12 | 10 | 9 |
| DOUBLE PRECISION constant, digits | | | | | 14 | 25 | 29 | 16 | 14 | 18 | | 19 | | 20 | | | | 16 | 16 | 16 | 16 | 21 | 21 | | 16 | 16 | 19 | | 17 |
| REAL, DOUBLE PRECISION magnitude | | | $10^{76}$ | $10^{69}$ | $10^{76}$ | $10^{308}$ | $10^{308}$ | $10^{38}$ | $10^{76}$ | $10^{76}$ | $10^{127}$ | $10^{38}$ | $10^{99}$ | $10^{76}$ | $10^{99}$ | $10^{99}$ | $10^{38}$ | $10^{38}$ | $10^{38}$ | $10^{75}$ | $10^{75}$ | $10^{150}$ | $10^{616}$ | $10^{99}$ | $10^{75}$ | $10^{75}$ | $10^{77}$ | $10^{40}$ | $10^{38}$ |
| Variable name maximum characters | 5 | 6 | 6 | 6 | 6 | 8 | 8 | No limit | 6 | 12 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | No limit | 6 | 6 | 6 | 6 | No limit | 6 | 6 |
| Mixed mode arithmetic permitted | | | • | | | • | • | • | • | • | | | | | | | | | | • | • | • | | | • | • | • | | • |
| Assigned GO TO | | • | • | • | • | • | • | • | • | • | • | • | • | • | | | | • | • | | • | • | • | • | | • | • | • | • |
| Logical IF, relations | | • | • | • | • | • | • | • | • | • | | • | • | • | • | • | • | • | • | | • | • | • | • | | • | • | • | • |
| DOUBLE PRECISION operations | | • | | | • | • | • | • | • | • | | • | | • | | | | • | • | • | • | • | • | | • | • | • | | • |
| COMPLEX operations | | • | | | • | • | • | • | • | • | | • | | • | | | | • | • | | • | • | • | | | • | • | | • |
| LOGICAL operations | | • | • | • | • | • | • | • | • | • | | • | • | • | • | | | • | • | | • | • | • | • | | • | • | • | • |
| Dimension data in type statements | | • | | • | • | | | • | • | • | • | • | • | • | | | • | • | • | • | • | • | • | | • | • | • | • | • |
| Labeled COMMON | | • | | • | • | • | • | • | • | | | • | • | • | | | | • | • | | • | • | • | • | | • | • | • | • |
| Maximum array dimensions | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 7 | 63 | 3 | 7 | 3 | 3 | 3 | 3 | 3 | 3 | 7 | 3 | 7 | No limit | 3 | 3 | 3 | 7 | No limit | 3 | 7 |
| Adjustable dimensions | | • | • | • | • | • | • | • | • | • | | • | | • | • | | | • | • | | • | • | • | • | | • | • | • | • |
| Zero and negative subscripts | | | | | | | | • | • | | | | | | | | | | | | | • | | | | | • | | |
| Subscripts may be any expression, with subscripted variables permitted | | | | • | | • | • | | | • | | | | | | | | | | | | • | | | | | • | | • |
| Subroutine multiple entries and/or nonstandard returns | | | | | | | • | | | | | • | | | | | | | • | | • | • | | | | • | • | | • |
| DATA statement | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | | | • | • | | • | • | • | • | | • | • | • | • |
| Object time FORMAT | | • | • | | • | • | • | • | • | • | • | • | • | • | • | | • | • | • | | • | • | • | • | | • | • | • | • |

1965

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Lee Amaya<br>SIAC<br>55 Water St.<br>New York, NY 10041<br>212-623-7751 | Lockheed Aircraft Corp.<br>Calif. Division |
| Paul Armer<br>Charles Babbage Institute<br>701 Welch Road, Suite 224<br>Palo Alto, CA 94304<br>415-328-0984 | Rand |
| Leo A. Aroian | Hughes Aircraft Co.<br>Research & Development Lab |
| J. E. Barry | Lockheed Aircraft Corp.<br>Missile Systems Division |
| R. W. Bemer<br>Honeywell Information Systems, Inc.<br>P. O. Box 6000<br>Phoenix, AZ 85005 | Lockheed Aircraft Corp.<br>Missile Systems Division |
| Ray Berman | North American Aviation, Inc.<br>Dept. 56-72 |
| ACM R. Boden | North American Aviation, Inc.<br>Field Laboratory<br>Santa Susana, CA |
| Elaine Boehm<br>deceased | IBM<br>Poughkeepsie |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Robert Bosak<br>Abacus<br>12301 Wilshire Bouelvard, Suite 201<br>Los Angeles, CA 90025<br>213-820-6955 | Lockheed Aircraft Corp.<br>Georgia Division |
| Harvey Bratman<br>SDC<br>2500 Colorado Avenue<br>Santa Monica, CA 90406<br>213-829-7511, Ext. 2133 | Lockheed Aircraft Corp.<br>California Division |
| Ed Braun | Lockheed Aircraft Corp.<br>Missile Systems Division |
| ACM Willard Bouricious | IBM<br>Programming Research Dept.<br>Poughkeepsie |
| A. John Carlson, Jr. | The Ramo-Wooldridge Corp. |
| John Caywood | Lockheed Aircraft Corp.<br>Accounting |
| John A. DeVries | Curtiss-Wright Corp.<br>Nuclear Power Dept.<br>Research Division |
| Robert Douthitt | Lockheed Aircraft Corp.<br>Missile Systems Division |

| NAME & ADDRESS | CO. ORIGINALLY AT SHARE I & II |
|---|---|
| Frank Engel, Jr. | Westinghouse Electric Corp.<br>Pittsburgh |
| Gerald Fine | Lockheed Aircraft Corp.<br>Georgia |
| Jim Fishman<br>General Motors Technical Center<br>Computer Science Department<br>Warren, MI 48090<br>313-575-3237 | General Motors Corp.<br>Special Problems Dept. |
| Lou Gatt Spiteri<br>Avenue Chauhtemoc 696<br>Cuidad Del Sol<br>Guadalajara, Jalisco<br>Mexico | Los Alamos Scientific Lab<br>Univ. of California |
| Julian J. Go~~a~~odpasture | Lockheed Aircraft Corp.<br>California Division |
| Gene Gordon | Rand |
| John Greenstadt | IBM<br>Applied Science Division<br>590 Madison<br>New York, NY |
| Irwin Greenwald<br>22836 Margarita Drive<br>Woodland Hills, CA<br>213-883-1494 | Rand |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| D.(Dan) P. Haggerty ACM | Lockheed Aircraft Corp.<br>Georgia Division |
| Ben F. Handy | Lockheed Aircraft Corp.<br>Missile Systems Division |
| Wesley Harker ACM | California Research Corp. |
| Donald E. Hart<br>General Motors Research<br>Computer Science Department<br>Warren, MI 48090<br>313-575-2643 | General MOtors Corp. |
| E. H. Hawkotte VE | Lockheed Aircraft Corp.<br>Missile Systems Division |
| Helmut Hoelzer | Redstone Arsenal |
| Robert Hughes | University of California<br>Radiation Lab<br>Livermore, CA |
| Edwin L. Jacks<br>G.M. Technical Center<br>500 Building, Room 307<br>Warren, MI 48090<br>313-575-8700 | General Motors Corp. |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Steve Jamison | IBM<br>Santa Monica |
| Nowell Johnson | Lockheed Aircraft Corp.<br>Accounting |
| Fletcher Jones<br>deceased | North American Aviation, Inc.<br>Dept. 56-22 |
| John Jordon | Boeing Airplane Company |
| Allen Keller | General Electric Corp.<br>Lynn, MA |
| M. Larkin | Lockheed Aircraft Corp.<br>Missile Systems Division |
| E. G. Law | North American Aviation, Inc. |
| Richard C. Luke<br>NASA Ames | Lockheed Aircraft Corp.<br>Burbank |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Essor Maso<br>Hughes Aircraft Company<br>Centinella & Teale Streets<br>Culver City, CA 90230<br>Bldg 6, M.S. D104<br>213-391-0711, Ext. 6938 | Hughes Aircraft Corp.<br>Research & Development Lab |
| John Matousek<br>Johns Mannville 2-21<br>P. O. Box 5108<br>Denver, CO 80217<br>303-979-1000, Ext. 2192 | Rand |
| Tom E. McCool | National Security Agency |
| Owen Mock<br>CSC<br>650 N. Sepulveda Blvd<br>El Segundo, CA 90245<br>213-678-0311, Ext. 1367 | North American Aviation, Inc.<br>Dept. 60 |
| Norman Moss | IBM<br>Pittsburgh Office |
| Thomas D. Mueller | California Research Corp. |
| Ray Nutt<br>CSC<br>650 N. Sepulveda Blvd<br>El Segundo, CA 90245<br>213-678-0311 | United Aircraft Corp. |
| Robert L. Patrick<br>9935 Donna<br>Northridge, CA 91324<br>213-349-2225 | General Motors Corp. |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Randall Porter | Boeing Airplane Company |
| Walter Ramshaw | United Aircraft Corp. |
| J. R. Reynolds<br>IBM E REGION | Lockheed Aircraft Corp.<br>Georgia Division |
| Ward Sangren | Curtiss-Wright Corp. |
| Helmut Sassenfeld ABM<br>SAFEGUARD | Redstone Arsenal |
| Walter C. Schlieser | Douglas Aircraft Co., Inc.<br>El Segundo, CA |
| J. C. (Cliff) Shaw<br>10912 Whitburn<br>Culver City, CA<br>213-839-8192 | Rand |
| Don Shell | General Electric Company<br>Aircraft Gas Turbine Division |

| NAME & ADDRESS | CO. ORIGINALLY WITH AT SHARE I & II |
|---|---|
| Tom Steel<br>AT&T<br>295 N. Maple Avenue<br>Basking Ridge, NJ 07920<br>201-221-5619 | Rand |
| Jack A. Strong<br>13940 Tahiti Way<br>Marina del Rey, CA<br>213-823-4705 or 213-823-5024 | North American Aviation, Inc.<br>Dept. 60 |
| Carol Tross | Lockheed Aircraft Corp.<br>California Division |
| Richard E. VonHoldt | Univ. of California<br>Livermore |
| Ed Vorhees | Los Alamos Scientific Lab<br>Univ. of California |
| Frank V. Wagner<br>Informatics, Inc.<br>21031 Ventura Blvd<br>Woodland Hills, CA 91364<br>213-887-9040 | North American Aviation, Inc.<br>Dept. 56-72 |
| William J. West | California Research Corp. |
| C. M. (Chuck) Wimberly | Lockheed Aircraft Corp.<br>Missile Systems Division |
| Peter A. Zaphyr | Westinghouse Electric Corp.<br>Pittsburgh |

1982
National
Computer
Conference
Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

Reply to: J. A. N. Lee
Pioneer Day Chairman

Robert W. Bemer
2 Moon Mountain Trail
Phoenix, AZ 85023

Dear Bob,

As you may remember, I contacted you some months ago regarding your giving me permission to freely examine and reproduce copies from the files which you deposited with the Smithsonian Institute and which referred to your work on the programming language FORTRAN (or FORTRANSIT). I would be grateful if you could provide me with that permission in writing since I am planning on requesting access to those materials in the near future. In reviewing my materials on FORTRAN it is clear that FORTRANSIT is not well represented in the documentation and I believe it is imperative to close the gap before next year's Pioneer Day.

Best wishes,

Jan

cc: D. D. McCracken

1982
National
Computer
Conference
Astrohall
Houston, Texas
June 7-10, 1982

American Federation of Information Processing Societies
Association for Computing Machinery
Data Processing Management Association
IEEE Computer Society
Society for Computer Simulation

Reply to:

CONFERENCE CHAIRMAN
Russell K. Brown, CDP
3420 Yoakum Boulevard
Houston, Texas 77006
713-524-3420

RAM CHAIRMAN
rd L. Morgan
Department of Decision Sciences
The Wharton School/CC
Philadelphia, PA 19104
215-243-7731

PROGRAM VICE-CHAIRMAN
Eric K. Clemons
Associate Professor of Decision Sciences
The Wharton School/CC
University of Pennsylvania
Philadelphia, PA 19104
215-243-7747

PROFESSIONAL DEVELOPMENT SEMINARS
Joseph Campisi
Aetna & Casualty
151 Farmington Avenue
Hartford, CT 06156
203-273-3611.

NCCC LIAISON
Harvey Garner
University of Pennsylvania
Moore School D-2
Philadelphia, PA 19104
215-243-4787

PIONEER DAY
J.A.N. Lee
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
703-961-6931 -5780

PLENARY SESSIONS
Susan Rosenbaum
T
C278
P.O. Box 3509
New Brunswick, NJ 08903
201-457-2664

VICE-CHAIRMAN (PROGRAM ACTIVITIES)
Robert Stirling
IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604
914-696-4251

PROGRAM COMMITTEE
Gene P. Altshuler
O. Peter Buneman
James E. Emery
Dennis Frailey
Robert Frankston
Randall Jensen
Beverly K. Kahn
Alan N. Smith
Amy D. Wohl

AFIPS PROJECT MANAGER
Sam Lippman
AFIPS
1815 N. Lynn Street
Suite 800
Arlington, VA 22209
703-558-3614

Dr. Uta Merzbach
Div. of Mathematics
Nat. Museum of America History
Smithsonian Institute
Washington, DC 20560

Dear Dr. Merzbach:

Many thanks for your assistance and that of Mrs. Jordan during my visit to the Smithsonian last week. I was disappointed that the ALGOL and COBOL files provided by Bob Bemer did not contain any reference to FORTRANSIT as I had hoped. On the other hand, there was a fair amount of information on SHARE/IBM relationships vis-a-vis FORTRAN which will help to supplement our other information and will perhaps lead to new areas of investigation when we receive the copies of the appropriate pages.

As we go into the new year, I am planning to document much of the material found in the files I have examined at the Smithsonian. I will send you copies so that you can use them to supplement the materials you possess.

Many thanks.

Yours Sincerely,

J.A.N. Lee
Professor

dgb

✓xc: Robert Bemer

# Computers and Automation

E. L. HARDER
FELLOW AIEE

IN THE BRIEF SPAN of years since the closing days of World War II, we have witnessed a technological development in computers of such broad proportions that it is still impossible to appraise its far reaching effects adequately. Perhaps, the best way to express the enormous influence of this revolution is simply to point out that practically all of man's actions in producing the necessities and luxuries of life fall into two categories, namely his mental and his physical work. All of his mental work has to do with the processing of information in some form or other, and the close relationship of the computer to the human brain and its functioning needs no elaboration here. It should have been expected that nearly all of the operations of the civilized world, in which the human brain is involved to a greater or lesser extent in carrying out the task, could be aided by a technological development which performs accurately and at high speed, some of the functions of the brain. This is variously called computing, or information- or data-processing. Its field of application is as broad as human knowledge and is far beyond the comprehension of any one individual. However, large teams of scientists and engineers are currently at work in many of these compartmented fields of knowledge, learning how to adapt and develop the automatic processing methods of the computer to the needs of their particular fields.

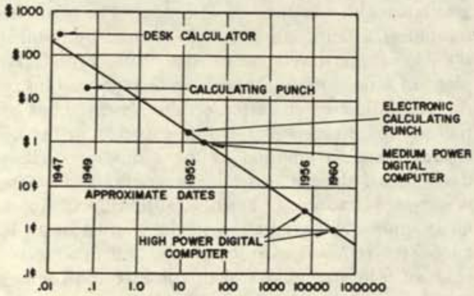


Fig. 1. Computation cost vs computer speed.

The revolution in computing has progressed to a point where two clearly distinct areas of computer automation are evident, the one associated with the automation of physical processes, controlling and measuring power machinery and vehicles, and the other having to do with the automation of man's clerical and mental work.

In the former, one can trace the stages of development starting with the primitive tool. Next came the power tool in which the forces of nature were harnessed to augment man's physical power but with the entire intelligence for its use remaining with the man. We have witnessed the gradual additions of intelligence to make this tool perform more and more automatically. The earlier elements of intelligence were simple computing devices, usually in the form of analog elements built directly into the mechanism of the tool. The potentialities of punched paper for sequencing processes was early recognized in the Jacquard Loom and the player piano. Most servomechanisms have some form of analog computing function built in. However, as the ratio of intelligence to power has increased, the computer is beginning to emerge as a distinct organ, separate from the power handling facilities and providing for the sequencing, the optimizing, the computing, and the data processing for the controlled process.

Dr. E. L. Harder is director, Analytical Department, Westinghouse Electric Corporation, East Pittsburgh, Pa.

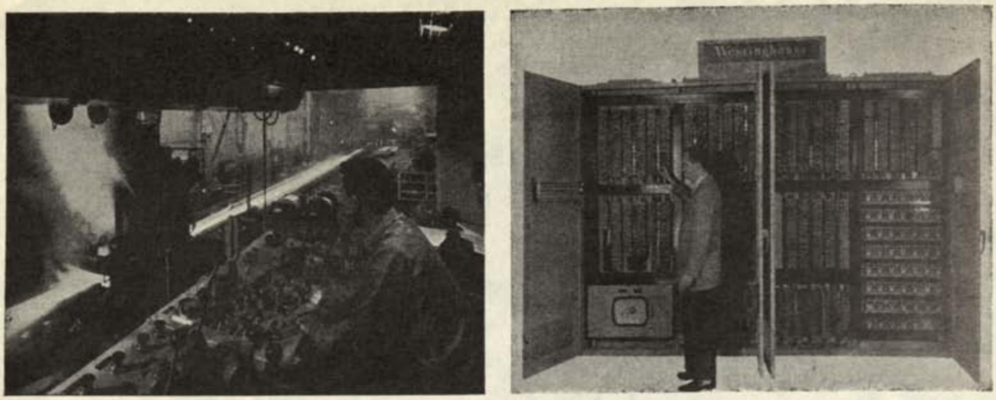


Fig. 2. PRODAC, programmed digital automatic control (similar to unit, right) at work controlling a reversing roughing mill (left, seen from control pulpit) of the Jones & Laughlin Steel Company at Aliquippa, Pa., near Pittsburgh.

In the field of computers for processing man's mental and clerical work, the early aids to calculation—the desk calculator, the slide rules, and the cash register—have grown through the stage of business machines capable of performing simple calculations on large numbers of similar documents. They have grown to the medium- and large-scale internally programmed, automatic computer, capable of carrying out long sequences of business, engineering, or military calculations. The high-speed internally programmed calculator brought, for the first time, a tremendously lowered unit cost per calculation (Fig. 1) with its explosive economic effect of broadening the base of applications to which computers might profitably be applied. For example, computation requiring 2 weeks on a desk calculator and costing $300 in 1947 will cost 1¢ on a high-powered computer of 1960, a reduction of 30,000 to 1. This single economic factor has been a major influence in extending the computer into the processing of complete sequences of business calculations such as the payroll, the inventory and stores program, and the accounting of a factory.

In engineering, the increased computing power and lower unit cost has been put to work in solving many previously intractable technical problems as well as carrying out the complete logic of design for more standardized products.

In addition to advances in the computer itself, still proceeding at a revolutionary rate, the science of programming is undergoing intensive development to reduce this tremendous bottleneck between the job to be done and the machine. It requires a little imagination to realize the vast potentialities that remain practically unexplored as the power of the mechanized logic of these computer and programming developments are brought to bear on the sequencing and optimizing of controlled processes.

In many industries, automation has reached a high state, with feedback control, or simply with sequenced controls, eliminating most of the manual operations previously required in mass production processes. The transfer machine for complete machining of an engine block, the integrated four or five stand tandem cold-rolling mill for steel strip, and the paper making machine are but a few examples. For large production of an invariable product, this form of automation has reached high development.

However, where the product or ingredients are variable, and where greater and more flexible intelligence can profitably be applied, computer techniques are being rapidly incorporated into the automated system. This is exemplified by the programmed digital automatic control for a blooming or slab mill (Fig. 2), the digitally controlled skin or profile mill for aluminum wing structures, or the automatic economic dispatching control of an electric power system.

It is these latter phases of automation of physical systems, together with the automation of mental and clerical work, that are to be treated. Both are closely associated with computer development. The relation of computers to automation will be treated by outlining the important developments that have been taking place in several areas. Taken together, these illustrate the revolutionary character of the development and indicate the progress that can be expected in the next several years. In addition to the computer itself, we shall discuss developments in programming; in engineering applications; in business, military, and Government applications, as well as in the translation of language, the retrieval of information, and other diverse fields of information. We shall consider the computer in data logging and control. We shall consider all of the accessories developed for data processing as well as the computer techniques themselves as a "bag of tricks" out of which much practical automation is being accomplished short of the full computer.

### DEVELOPMENTS IN THE COMPUTER

THE FIRST LARGE-SCALE, internally programmed, electronic digital computer was the ENIAC, completed about the end of World War II (Fig. 3). This machine had 18,000 vacuum tubes, and used vacuum tube high-speed storage. It used a 600-entry wired storage for a function, namely the drag function, since the computer was intended primarily for ballistic problems. Magnetic tapes were notably lacking at that time, but were soon developed. Computers installed since that time are shown in Table I. The four panels shown in the background in Fig. 3 are the initiating panels, cycling panels, and the two master programmers. On the right is shown a function table. The two master programmer panels determine the sequence of operation in the solving of a problem in which panel numbers are added, subtracted, multiplied, and divided. Also in these panels are stored numbers for use in a later stage of the solution.

The most notable developments s.nce that time have been the mercury delay line storage, then the Williams tube high-speed, random-access storage, and the magnetic-core storage which soon succeeded it and which now forms the. high-speed memory of most modern large-scale computers. For computers of medium to low speed and cost, the magnetic drum has been highly developed. More recently, the demand for larger random access, of lower cost per bit, principally for the storage of active inventories of business items, led to the development about 1956 of magnetic disc and multiple magnetic tape memories. The Ramac unit of The International Business Machines Corporation (IBM) is a disc memory operating in the fashion of a juke box and storing some 5 million 6-bit characters, arranged as 50,000 separately addressed, 100-character records. The average access time of 1/2 to 1/6 seconds, depending on the number of reading arms, is adequate for the transaction rate of many businesses having inventories within the 50,000 record size.

**Table 1. General Purpose Digital Computer Systems Installed January 1959**

| Computer System | Number* |
|---|---|
| Large Scale (Magnetic tapes and microsecond arithmetic) | 316 |
| Medium Scale (Magnetic tapes and millisecond arithmetic) | 348 |
| Small Scale (No magnetic tapes, but internally programmed) | 1,370 |
| Miscellaneous (Card calculators and others) | 5,166 |
| TOTAL | 7,200 |

*Figures were taken from John Diebold Associates publication *Automatic Data Processing Service Newsletter*, vol. III, no. 17, Jan. 12, 1959.

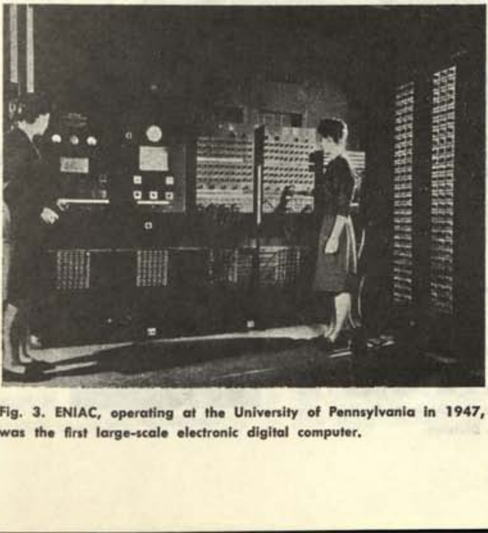


**Fig. 3. ENIAC, operating at the University of Pennsylvania in 1947, was the first large-scale electronic digital computer.**

Since 1956, solid-state techniques have been used to an increasing extent, so that as of 1959, several all-transistorized or magnetic computers are on the market with substantial increases in both speed and reliability.

Magnetic tapes are used on all large and many medium size computers today. The 15,000-cycle frequency (200 characters per inch times 75 inches per second) characteristic of 1956-57 has been stepped up by 4-to-1 to a 62,500-cycle frequency (555 characters per inch times 112.5 inches per second) or higher. Wide tapes are used on some computers further multiplying the information transfer rate from a single tape transport unit.

The first high-speed printer (600 lines per minute) was developed about 1954 for the UNIVAC computer. Facsimile printing at 1,000 lines per minute and still higher speed photographic printing from the Charactron have since been developed. Off-line printing has become the rule for most large installations, and for many medium installations with magnetic tape, to conserve the valuable computing time. Buffering to permit simultaneous read, write, and compute, first used for the serial business computers, is now available for high-speed, parallel scientific and engineering computers also, to increase the speed for problems with large input-output requirements.

Of note during the last 5 years has been the greatly increased use of small drum computers, priced under $100,000. Over 600 computers in this bracket have found their way into as many diverse applications, ranging all the way from research on "home permanents" to on-line reduction of data on a critical atomic facility. There is profound significance in this broadening of the base of computer usage—in having this many diverse groups extending computer technology into so many avenues of human effort.

### SAGE COMPUTER

LANDMARKS of present advance for large-scale systems in service are the computers of the SAGE system for continental air defense (Fig. 4). Working with unprecedented reliability on an around-the-clock schedule, such computers process constant streams of radar data, perform complex computations, and present visual displays of the air situation to U. S. Air Force personnel. Their duplex facilities assure their 24-hours a day primary air defense mission while scheduled maintenance is being performed and training programs carried out.

Most important for automation, and far more difficult to express, is the progress during this same period in computer techniques and accessories. The general-purpose computer represents an important use but far from the only use of these techniques. It is the availability of this tremendous array of adaptive and peripheral devices that takes the computer out of the laboratory and into practical automation.

## DATA LOGGING

THE ABILITY to process data rapidly implies the necessity to gather data rapidly and efficiently, hence the development of data logging equipment. Data logging requires, first of all, transducers from all types of measurements into analog or digital form. These include currents, voltages, power measurements, rates of flow, pressures, temperatures, weights, times, counts, thicknesses, compositions, speeds, hardness, viscosity, moisture content, density, concentration, and all manner of other physical, chemical, metallurgical, atomic or other conceivable properties. An ever-increasing range of such transducers is becoming available of improved quality and reliability. Logged data becomes much more valuable with the ability to process it rapidly, with increasing knowledge of what action to take as a result of the processing, and with automatic facilities to take the indicated action.

Generally, to gather such data, it is necessary first to schedule its taking. This requires programming or timing equipment. Often analog-to-digital conversion equipment is required where the basic transducers develop analog outputs. This equipment may be time shared among a large number of measurements, requiring accurate and high-speed conversion equipment.

Data logging usually requires the typing or printing of some or all of the data for monitoring purposes and the recording of some or all of it for later processing. Frequently, the availability of the data leads to a requirement for alarms at limiting values, and perhaps the more extensive action of automatically changing a set point or otherwise altering the process being monitored. The final records may be on paper tape, magnetic tape, punched cards, or other suitable medium. In some cases, the data logging feeds directly into a computer which processes the data on the spot and gives out only the desired processed results instead of the multitudinous items of input data.

Equipment in considerable variety is now available for all of the functions previously described. However,

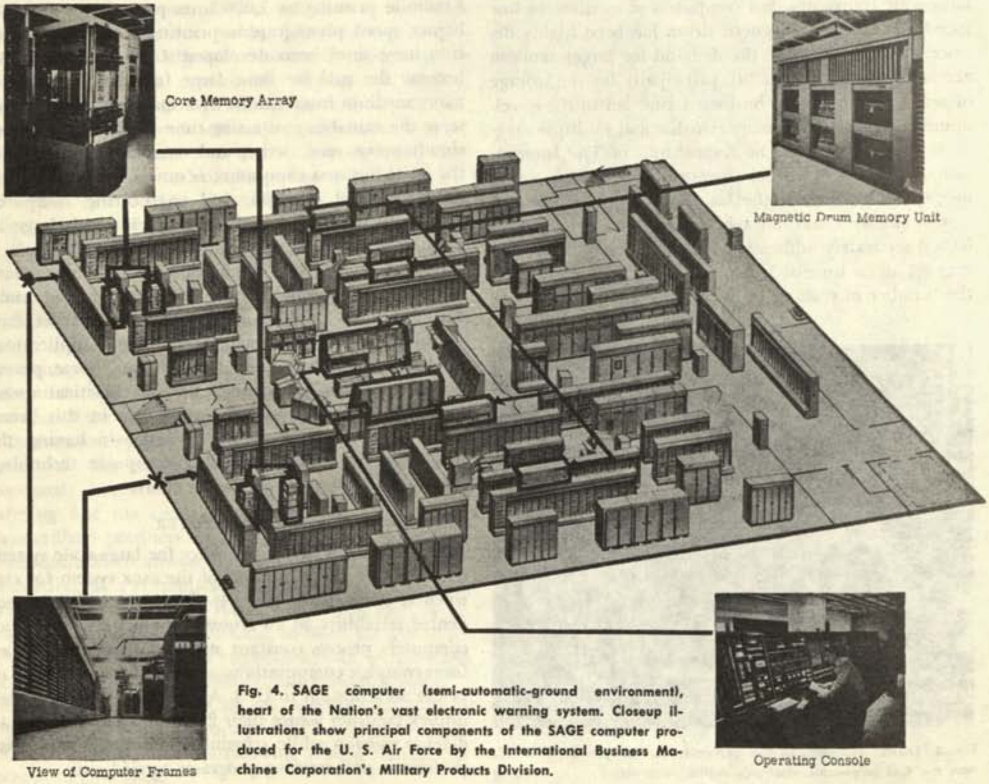


Fig. 4. SAGE computer (semi-automatic-ground environment), heart of the Nation's vast electronic warning system. Closeup illustrations show principal components of the SAGE computer produced for the U. S. Air Force by the International Business Machines Corporation's Military Products Division.

as the applications grow, the conditions to be met continue to expand and we should expect to see over the next several years tremendous advances in both ideas and equipment for filling the needs in this area of automation.

### DATA COMMUNICATION

THE LOGGING AND PROCESSING of data also implies a need for communicating it from one point to another, or from one computer to another at the same point. We may cite the continental defense requirements for transmission of data from many points to the central computers of the SAGE system (Fig. 4). In an industrial operation, we may have the transmissions from many district offices over a network of teletype circuits to a central order-processing computer. It may be necessary to extend the services of a central high-powered scientific computer to the engineering departments at other divisions of the company at remote locations up to several hundred miles away. It may be necessary to transmit data from field tests of a missile or a turbine installation to a remote computer, either on a permanent or temporary basis.

It may be necessary for a large number of telephone installation and service men to communicate with a central computer for their daily orders for supplies, or for the workers of a factory to communicate through time clocks with a central computer to avoid intermediate operations in processing their hours of work. It may be necessary for a branch of the military to keep a central computing facility continually appraised of the status of supplies throughout the world and, vice versa, to keep certain information current at numerous locations. It may be necessary for a transportation company to transmit reservation information to and from central points as well as operational information, flight plans, status of facilities and products.

All of these and innumerable other data-communication requirements associated with computers have received a great deal of attention and development during the last several years in order to provide economical and suitable services for all of the different classes of requirements. These range from the simple teletype circuits now with new checking features for data transmission, the 80-column card transceivers for using up to four 11-card-per-minute transmissions over four separate carriers on a telephone voice channel, and the magnetic-tape to magnetic-tape transmission at higher speeds utilizing the full capabilities of voice channels. Other variations include the cardaphone, moderate speed transmission of data over ordinary telephone circuits with a minimum of terminal equipment, and the rapid transmission of teletype over phone circuits by first "playing" it onto audio magnetic tape, then transmitting at a high rate over the phone circuit, and finally stepping back to the teletype rate from the received audio tape.

Automation of far-flung enterprises is obviously intimately interlinked with these developments in the communication of data, and as new requirements are continually arising, together with a pressing need for better solutions to the old problems, the developments in this area in 1959 are surging forward at an unprecedented rate. It can be expected that this communication adjunct to automation will see great advances in the next several years. In turn, the automation of communication circuit switching and accounting is benefiting by the advances in computing techniques.

### GENERAL AND SPECIAL PURPOSE COMPUTERS

ALTHOUGH LARGE-SCALE COMPUTERS originally were developed along separate lines for business and scientific purposes, two seemingly contradictory trends are now evident. One is the definite trend toward the common computer or computing center which has the proper complement of equipment to meet all of the large information processing requirements of the plant or group of plants. Originally, it was felt that the large input–output requirements and special storage requirements of the business problem required a separate type of machine from the engineering or scientific problems with their comparatively smaller input and output and very much greater computation requirements. It has since been found that even though the input–output of the engineering problem is small relative to the business problem, it may still require half of the total computing time and, thus, economically justifies high-speed input–output equipment, the same as the business problem. Thus, this distinction between the two is being rapidly obliterated.

Programmers of both business and engineering problems are finding that a large high-speed memory is extremely advantageous and time saving in the programming of problems of either kind and, thus, this requirement for an expensive element of the machine suggests the pooling of most problems on a single facility with very ample high-speed memory. The business operational problems of production control, and engineering which may combine in one close-knit operation the large computational problems of engineering design together with the references to large quantities of information on stocks, processes, time values, costs, and parts, naturally militates in this direction.

The supposition of low computing requirements of the business problem are based on compatibility with input–output speed. However, buffered computers with multiple tapes and higher speed tape transmission, may well raise the input–output speed to where business problems as well as engineering can profit from a lower unit cost per calculation with a high-speed computer. Thus, new computers on the market have remarkably improved abilities for handling combined loads of engineering and business problems.

Quite distinct from this trend is the appearance of on-line data-processing equipment such as the IBM RAMAC in which a comparatively inexpensive computer with large random access memory of $1/2$ second access or less, is applied for continuous use on a high activity business operation, and tends away from joint use consideration. This trend to special-purpose computers, is continuing over a broad area of operational and industrial problems. The specialized banking computers, the reservation computers for airlines and railroads, the large storage computers for air traffic control or for the continual processing of substantial inventories of many types, all trend toward the on-line computer, specialized or not. In industry, one finds the digitally controlled machine-tool director, intermediary between the large general-purpose computer for developing the tool path and the phase-modulated tape for the individual tool.

The PRODAC, programmed digital automatic control for mills and processes (Fig. 2), provides the necessary components of memory for storing the mill program, the necessary reading devices for quickly changing it to another program for a different mill operation, the necessary comparison or computing elements for making digital comparisons between the mill settings and the programmed values for actuation of the driving servos. Such a device has little use for the large repertoire of commands of a general purpose computer, but special attention must be given to reliability since the unscheduled maintenance of most general-purpose computers of the past could not be tolerated in such an application.

Thus, industrial control may utilize many of the computer techniques, incorporating them in particularly reliable forms especially for the important higher speed operations. The memories, the logic circuits, the programming and scheduling facilities, the reading devices, the data logging mentioned previously, including the analog-to-digital conversion equipment, all find their place in the automated mill, but frequently in specialized form. Somewhat further from the inner workings of the physical mill itself, the general-purpose computer finds its place in the planning, scheduling, and processing of the data on a "one step removed" basis. It is supplied with data and programs to be processed in accordance with the need of that particular industry. This mill or operational use of the general-purpose computer compares with the general-purpose computer in the manufacturing plant for the control of materials and stores, for the accounting operations, for production scheduling and optimizing, for payroll and records, and for the analysis of operating data.

### ANALOG COMPUTERS

THE GROWTH of analog computers, while less spectacular, has been steady. Most widely used are the electronic differential analyzers for the solution of regulation, dynamics and other problems of physical systems which can be expressed in differential equations. Network calculators and field models and the large passive element computers for transient and vibration analysis of systems and structures constitute another large class. The vast majority of all special-purpose computing devices associated with machinery, control, regulation, and instrumentation are analog in nature, at the present time. The use of analog computers for navigation and fire control directors, and for industrial control and simulation greatly exceeds the use of digital techniques in this area.

Principal developments responsible for the increased use of analog machines are: a full order of magnitude of accuracy improvement, the introduction of centralized control, and the substitution generally of static electronic devices for servo multipliers and function generators, together with revolutionary detail improvements.

The digital read-in and read-out and other automatic features have further enhanced its use while the development of analog–digital conversion equipment has led to the combined use of analog and digital facilities for large simulation problems requiring the capabilities of both computers.

Although primarily restricted to the engineering and scientific field, the number of analog computers in many large industrial establishments is quite comparable to the number of digital computers. A large analog installation is generally made up of a number of smaller analog computers, which can be used independently or connected through central control.

### DIGITAL COMPUTERS OF THE FUTURE

FROM WHAT HAS GONE BEFORE, it can be clearly seen that the future of computers does not hinge entirely on increases in speed, memory size, or logic, but also on the expansion and utilization of the tremendous range of facilities already developed, which are taking computer techniques and mechanized logic into all phases of industrial operation.

Nevertheless, speed and memory size and extent of logic, which can be economically justified, still have a strong bearing in some of the more interesting and sociologically important developments of the computer of the future. If the large-scale computer of 1958 is described as a commercial unit with a one megacycle rate, using vacuum tubes capable of adding 40,000 ten-digit numbers or multiplying 4,000 in a second, then its counterpart in the development laboratories is a solid-state computer with a pulse rate of 40 mc. Combined with new technological ideas, it has superior computing facilities as well as memories, buffering, and flexibilities, giving it an effective computing power of the order of 400 times the commercial unit of the last few years.

This laboratory unit with its 40-mc rate corresponds to a decision element speed of 25 millimicroseconds operating time. At this speed, the length and stray capacity of wires is certainly becoming important. However, with decision elements operating in 2½ millimicroseconds, which have been proved feasible using solid-state techniques, the computing circuitry of 1960 being developed in the laboratory will be up against serious time delays in the wire. Although the theoretical time for electromagnetic waves to travel one foot is one millimicrosecond, end effects increase this to 1.7 millimicroseconds for a one-foot connection. This is 7/10 as long as the operating time of a decision element of 2½ millimicroseconds. Developments on the horizon make it quite evident that this decision element time will decrease below 0.25 millimicroseconds. (A semiconductor switching in 0.05 millimicrosecond has been announced by *Electrical Design News*, January 1959, page 3.) The one foot of wire at this stage would represent a delay seven times as long as the element and would render the development of the element worthless.

This spells but one thing—microminiaturization. Although miniaturization studies are proceeding on all solid-state devices, a great deal of interest has been focused on the thin film and cryogenics.

The cryogenic element is based on the phenomena that wires held at about 420 F below zero can be switched into and out of superconductivity by a small magnetic field which can be produced simply by a current in one wire passing over another. The simplest cryotron is, therefore, a crossing of two wires. Different materials have different transition temperatures so that the controlling wires can always be kept superconducting and require no energy. Any number of elements can be switched from a single element. An external d-c current source provides the basic power through each wire.

Work at Massachusetts Institute of Technology reported by Dudley Buck at the Eastern Joint Computer Conference, Philadelphia, Pa., Dec. 1958, indicates there is reasonable possibility from experimental work that such cryotrons can be made in batch processes by selective etching of thin films of conducting material coated on insulators utilizing electron beam techniques to select the parts to be etched, and leaving a network of lines and elements so fine as to be invisible to the human eye with the most powerful optical microscope. The lines in this array would be 1/10 micron in width or 1,000 angstrom units compared with 4,000 to 8,000 angstroms, the wave length of visible light. Cryotron elements would have a minimum spacing of one micron apart in each direction giving a theoretical maximum of $10^8$ cryotrons per square centimeter on a thin film. Successive thin films can be shielded by superconducting layers. Allowing several cryotrons to a logic element and a tremendous space (relatively) for interconnections, a packing of $10^4$ logic elements per square centimeter should be attainable. The complete arithmetic unit logic of a present large-scale computer reduces to not over a cubic inch on this basis. Needless to say, the lead lengths in such an array would be amply short and the time constants should be short enough to permit another decade of advance below the 2½ millimicrosecond decision element time cited for 1960.

Philosophers, while admitting a faint similarity between some functions of the brain and a computer memory, have always pointed out the disturbing fact that a sufficient number of elements to compare with the brain would require a unit the size of a large office building. However, a spacing of cryotrons one micron apart on thin film compares directly with the density of neurons in animal nerve tissue.

The recent studies in which computers have been used to simulate self-organizing mechanisms in which information is stored through the connection of elements rather than their state provides a more interestingly close resemblance to the brain. Such mechanisms can be arranged to learn a desired response. They resemble the brain in that the removal of part deteriorates rather than completely destroys the functioning. The remainder contains the learning power to improve itself again. This presents the interesting possibility of logical structures of unprecedented reliability and of entirely new computer philosophy. Thin film cryogenics holds promise of sufficiently large arrays at reasonable cost to study self-organizing mechanisms adequately. This may enable us to learn more of the functioning of the human nervous system and also to derive ideas from nature for the further development of computer logic.

Development of the ideas, the logic, and the programming philosophies for utilizing these inherent capabilities will keep scientists busy for a long time. It is amply evident that space-age automation will not be hampered by any ceiling in the advance of computer technology.

### PROGRAMMING

THE ASSEMBLY PROGRAM, which is a necessary adjunct to any large computer installation today, brings to the programmer in the form of a package program most of the automatic programming developments to date. He writes his program in the simplest form and then "assembles it" on the computer. The assembly program converts his abbreviated program into a complete machine language program by performing the following typical operations, all formerly done by the programmer himself:

1. It converts the simple symbolic names for the commands into decimal or binary machine codes as required.
2. It brings into the program any needed routines from the library tape.
3. It allocates memory space and inserts correct addresses in all instructions.
4. It detects common errors.
5. It converts from our decimal number system to the binary number system if used.
6. It condenses the program to a tape or compact card form.

The compiler program, such as Fortran for the IBM *704*, goes a lot farther. In using an assembly program, the programmer must write each step (except for subroutines) albeit in abbreviated form. With the compiler, this is no longer necessary. He simply writes the mathematical statement for the program to be prepared and the compiler (Fortran means "formula translation") translates this into a machine program with an efficiency of 5- or 10-to-1 compared with writing out each step. Fortran is primarily for scientific or engineering problems.

In many fields, specialized languages are prepared enabling the programmer to express the problem as simply as possible in terms native to that field. Thus, a language has been written in which a steam system can be conveniently described to effect the heat balance solution with any arrangement of turbines, heaters, pumps, and boilers.

Another language has been written for dealing with words in a language for language translation. In this case, facility in dealing with "strings" of words is an objective.

A language for describing the desired tool path, rate of feed, and tolerance for digitally directed machine tools, permits the description of straight lines by two co-ordinate points, or one point and tangency to a circle of given center and radius. Similarly, other regular geometric shapes are used in a manner native to drafting.

A language was described recently for point mechanics, enabling any problem in masses, springs, levers, torques, and frictions to be simply described.

In each case, a program is then written to bridge from the problem statement in the use-oriented language to a machine program for solving the specific problem. In the mechanics language cited, the associated program prepares a Fortran program, which can then be compiled by Fortran into a machine language program. However, a compiler from the Fortran language could be written to other computers than the IBM *704* for which it was intended. This has, in fact, been done, "Fortransit" making the conversion of a Fortran written program to the "IT" compiler for the IBM *650*.

The universal language is the rallying point for work on machine compatibility; running programs on one machine that have been written for another. If written at a high enough and generalized enough level, hopefully in a "universal language," this may be done. The limitations today are so great that practically all programs are written in a language of the machine on which they are run. However an increasing number are written in a high-level language such as Fortran, and are far more susceptible to translation than direct machine language programs.

It is fervently hoped by everyone with a substantial stake in programs that good workable solutions to the compatibility problem will evolve in the next several years.

Considerable work has been done on compilers for business problems. Here the general concepts involve first a complete and orderly arrangement of all data into addressable records and files. Second, it must be possible to state explicitly what operations are to be performed on this data and the form and arrangement of outputs. Finally, from the characteristics of the specific machine, the compiler must prepare a working machine-language program that will accomplish the desired operation.

The concept of report generator implies the ability to specify conveniently and quickly a new derivation from the data needed for certain management decisions, a facility utterly hopeless without a high-speed computer. The possibilities of this are being actively explored by many management groups as a means of lessening routine data.

### DIGITAL CONTROL OF MACHINE TOOLS

THE IDEA of using numbers expressed as holes punched in cards or tapes to control the multiple motions of a machine tool, was first presented to the U. S. Air Force early in 1949. In the 10 years following, both the philosophy and equipment have been developed and over 100 contour milling machines are now either in use or under construction using the resulting principles. These are principally the skin mills and profile mills in the aircraft industry used for the machining of complex wing sections. Hundreds of other digital machine-tool applications are in use or under development for a wide variety of machine tools. Typical positioning applications are the placement of components on printed wiring boards, or the positioning of the work and tool turret in a large sheet-metal punch press.

For the contour milling operation, the calculation of the tool path presents an awesome problem, work on which started at MIT under Air Force contracts and has now been taken up by a joint industry effort. This is resulting in computer programs for converting the desired cuts into tool center paths.

To date, the starting point has been the engineering drawing. However, ideas for getting directly from a simpler specification of the desired shape to a tool path program for machining, bypassing the drawing, is developing. It appears that the rules for filets and curvatures, through which the draftsman develops from the basic requirements and dimensions into a finished object, may very well be programmed for a computer. However, this is for the space age ahead.

### ENGINEERING APPLICATIONS

IN MECHANICAL and electrical engineering, computers have been particularly important in the design and application of machinery and systems. Civil engineers have found it invaluable for the extensive earthwork calculations involved in highway and seaway construction and the calculation of structures.

In engineering design, the earliest problems attempted were the scientific problems which could be quite well-stated mathematically but many of which it was impossible to solve practically without the aid of the computer. These included the diffusion calculations for the nuclear reactor, the higher natural frequencies of turbines and compressor blades, and the modes, vibrations and frequencies of other more complex mechanical shapes. In design, the computer was widely used for the performance calculations in which the designer completely specifies a conformation of the device and

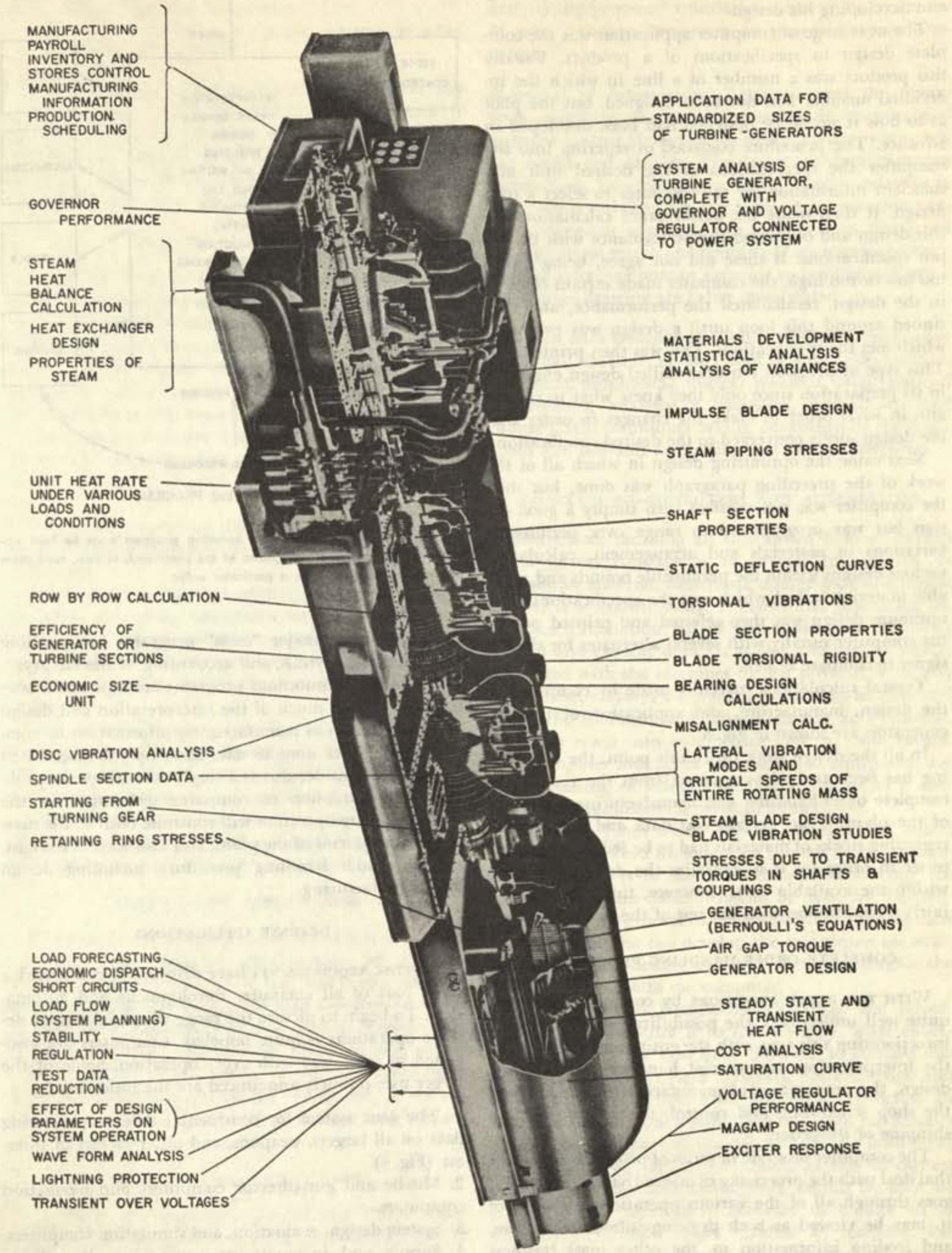


**Fig. 5. Typical calculations currently made by computer in the design, manufacture, and application of turbine generators. Exploded view of 135,000-kw machine built by Westinghouse Electric Corp., for the TVA Shawnee Station.**

calculates how this will perform as a step in revising and developing his design.

The next stage of computer application was the complete design to specifications of a product. Usually this product was a member of a line in which the individual member had not been designed, but the plan as to how it would be designed had been developed in advance. The procedure consisted of entering into the computer the specifications of the desired unit and sufficient information for the computer to select a trial design. It then made the performance calculations on this design and compared the performance with the input specifications. If these did not agree, being either too low or too high, the computer made certain changes in the design, recalculated the performance, and continued around this loop until a design was produced which met the specifications. This was then printed out. This type of program required skilled design engineers in its preparation since only they knew what to change and in what order to make the changes in order that the design surely converged to the desired specifications.

Next came the optimizing design in which all of the work of the preceding paragraph was done, but then the computer was not satisfied with simply a good design but was programmed to range over permissible variations in materials and arrangement, calculating various designs within the permissible bounds and available materials, all of which met the specifications. The optimum design was then selected and printed out by the computer usually with several alternates for the designer to compare as well.

Typical calculations currently made by computer in the design, manufacture, and application of turbine-generators are shown in Fig. 5.

In all the design work up to this point, the engineering has been treated as separate from the rest of the complete order-handling and manufacturing operation of the plant. True, certain cost data and information regarding stocks of materials had to be fed into the computer in order for it to optimize the design and keep within the available parts; however, this constituted a fairly trivial liaison with the rest of the operation.

### COMPLETE ORDER-HANDLING PROCEDURE

WITH THE DESIGN PROCEDURE by computer becoming quite well understood, the possibility presents itself of incorporating this step with the entire series of steps in the interpretation, commercial handling, engineering, design, the preparation of manufacturing information, the shop scheduling and control, the accounting and shipping of the order.

The computer program or series of programs in a plant that deal with the processing of one individual order as it goes through all of the various operations required of it, may be viewed as both drawing information from, and feeding information to, the other mass business programs of the plant as shown in Fig. 6. This has been simplified to show only a few of the many operations performed on an individual order and also only representative of the major "mass" programs for handling the materials, payrolls, and accounting of the factory.

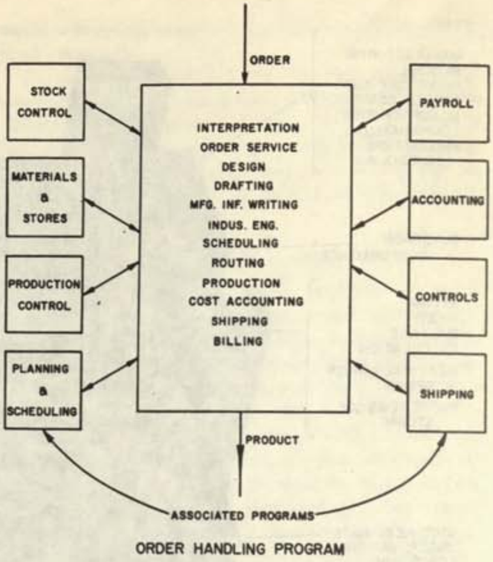


Fig. 6. A straight-through order handling program must be kept updated by the mass data programs of the plant and, in turn, feed them information developed on a particular order.

A number of ambitious programs are now under way for carrying out much of the interpretation and design and preparation of manufacturing information by computer. The work done to date has amply demonstrated the need to consider this as a single over-all system problem. The extension of computer technology in the manufacturing operation will continue both in the mass business problems of the plant, and also for the straight-through order handling procedure including design and manufacturing.

### DEFENSE APPLICATIONS

DEFENSE REQUIREMENTS have directed and financed a large part of all computer development and automation. To begin to picture the range of use in various defense operations is quite hopeless. Computers are associated in some way with every operation. Some of the larger uses publicly announced are the following:

1. The SAGE system for continental defense processing data on all targets, weapons, and other factors of interest (Fig. 4).
2. Missile and gun-director computers and navigation computers.
3. System design, evaluation, and simulation computers.
4. Supply and inventory computers for all military parts.
5. Air traffic control.
6. Logistic and operation research facilities.

7. Test data processing computers.

8. Computers for design of atomic reactors, missiles, aircraft, ships, and vehicles.

From depending on computers primarily for ballistic and range data in 1945, the defense organizations have come to look upon the computer as one of the most vital survival links in an atomic space age—depending on it for nearly all vital information and for all decisions that have to be preplanned.

### BUSINESS APPLICATIONS

A FEW of the more important business applications which have developed are the following. The computer control of inventory and stores has been profitable in many businesses. The payroll and all associated employee records are on computers, particularly where peripheral benefits can be obtained. Capital stock records, tool and facility records, and sales records analyzed for business control, are frequently computerized. Policy records and accounting of insurance companies and billing accounting of public utilities are both extensive uses.

Mail order houses and the repair parts and shipping stocks of many businesses are being placed on on-line computers.

All forms of accounting, traditionally using business machines, are being considered for larger scale computers and many profitable conversions have been made. Thus, computers are appearing in banks and commercial houses, hospitalization, and finance companies.

The use for operations research, for production planning and scheduling, for business decisions, for simulation of business operations to study alternatives, and for linear programming and optimizing, should be particularly mentioned. Here are the difficult problems, often of much greater payoff, in which much of the effort will be applied over the next several years.

### GOVERNMENT APPLICATIONS

THE BUREAU OF CENSUS was one of the first to encourage large-scale computer development for its use. UNIVACS were used for part of the 1950 census analysis. Other typical applications have been Government bond records, social security, veterans' records, income tax, air traffic control, map making, letter sorting, patent search, statistics of all kinds, as well as corporation, sales, and other tax accounting. These are, of course, additional to the business and defense applications cited previously.

### MISCELLANEOUS APPLICATIONS

*Translation of languages.* One of the truly monumental tasks, this is now in its infancy though one of vast social significance. Large random access, permanent memories are needed, permitting photographic and optical techniques. At present, teams of programmers and linguists are studying the linguistic rules and testing proposed translation algorithms by means of trial translator programs on general-purpose computers.

*Retrieval of Information.* Retrieval runs the gamut from searching classified information on chemical-formulas or United States patents to automatically abstracting books and articles. The latter involves character recognition and linguistic rules closely related to language translation for discovery of information content. Ingenious apparatus has already been developed for high-speed search and printing of abstracts, and engineering reference systems superior to pulling and printing a large drawing are being investigated.

*Research with Computers.* This is extending into many new areas: the study of dairy herd performance, medical, biological, geophysical, nuclear research, using planned experiments, analysis of variances, probability, and Monte Carlo techniques taking advantage of the high-speed and low-unit computing cost of the computers.

### CONCLUSIONS—COMPUTERS AND AUTOMATION

COMPUTERS, meaning all that broad field of devices whose prime function is to process information as opposed to material, constitutes a large part of automation as it is known today. It constitutes practically all of office automation, all of the automation of man's mental and clerical work. It is the intelligence which, associated with the machines for his physical work and his defense, removes the human limitations of time and space, the limitations of human sensing and reaction, in man's reach into outer space and into the future. The computer and all of the associated equipment for the processing of information elevates man to a new level of accomplishment through automation.

For those things which he can conceive and knows how to do, he need no longer be limited by the computational obstacles or the detailed decision making. He can now program these things to be done by the computer under his general control. Instead of making each design himself, he can develop the techniques for making an excellent design, and from there on relegate the routine design job to the computer.

This field holds tremendous challenge and opportunity. The computer is to the machine, what education is to man. We now have the tools to educate our machines. Engineers have accepted this challenge and in the last 10 years have made tremendous progress. In this time, the vista of possibilities has been continually expanding. The next 25 years should see the fulfillment of the dream of automation that is in but its earliest stage today. Computer intelligence should by then be applied wisely throughout industry to handle all of those types of tasks better handled by machine. Far from displacing man from employment, this will elevate him to a world of greater effectiveness and greater opportunity, as has the harnessing of the forces of nature in providing his physical power and transportation.

A Reprint From

ELECTRICAL ENGINEERING

Printed in the United States of America

American Institute of Electrical Engineers

The Institute assumes no responsibility for statements made by contributors.

MAY 1959

# NOTES

of the

# MEETING

on

# 705 SCIENTIFIC PROGRAMMING

## FOURTH SESSION GUIDE

FRIDAY
SEPTEMBER 13, 1957

SHERATON-PALACE HOTEL
SAN FRANCISCO, CALIFORNIA

## FRIDAY AFTERNOON SESSION

September 13, 1957

- - -

The Friday morning GUIDE meeting of 705 Scientific Programming convened in the GUIDE Suite, Room 2044, of the Sheraton-Palace Hotel and was called to order at 9:15 o'clock, a.m., with Mr. H. A. Thompson of The Texas Company presiding.

- - -

CHAIRMAN THOMPSON (The Texas Company): I would like to call this meeting to order, Gentlemen, and get started. We have a lot to discuss, a lot to do.

I would like to state at the outset that I am considerably gratified to see so many attending this meeting. I think that we have had a very good offering of personnel to serve on our FORTRAN working committee.

Thus far I have commitments for approximately five people, five whole people, not necessarily five individuals, and I think that as soon as we firm up our plans on what we are going to do in the next month, that even more people will join in because I know that there are some people that want to participate in this program that aren't at the meeting.

Yesterday at the Programming Committee meeting, I stated that there were seven aims -- or six aims, and I would like to at least partially tackle, or at least bring before you for your consideration, some of them.

Some of them, I think, we can actually put behind us, and those are:

(1) To establish the extent of industrial participation.

We can't do this completely because of what I said just a few minutes ago that there are people that want to get in on this that aren't represented at this time, but will come in later.

But the extent of industrial participation at this date is five men. We also want to establish the extent of IBM participation.

I don't know if Bob has been empowered to say anything?

MR. BEMER (IBM): Yes.

CHAIRMAN THOMPSON: I will list the other aims and we can back up and take that one up.

Number three, we would like to establish what this relationship between the people that are working for IBM and the people that are working for industry would ~~feel~~;

and, four, to establish their responsibilities.

Five, we will see if we can at least kick around the idea of which language level we should be shooting for.

Number six, what machine are we going to talk about, what configuration will we consider as a minimum in 705, so that we can design a system to fit that minimum machine.

Can anybody think of any additional items that would be proper to at least introduce at this time? This is probably more than we can get around to anyway.

MR. BEMER: I have a small one, the question of publication for the various people.

CHAIRMAN THOMPSON: We will have that as number seven. Anybody else?

(No response.)

CHAIRMAN THOMPSON: Well, with no further ado, let's back up, since we have established number one as best as we can thus far -- let's back up and pick up number two.

I would like to turn the forum over to Bob Bemer and see as to what he has to say on this line.

MR. BEMER: I have a chart here. It is by no reason complete, but what I am trying to do, I am trying to take the entire spectrum of equipment and computers from 650 up through STRETCH and make a chart for assemblies, business compilers and mathematical, scientific compilers.

I am running the machines this way (indicating) and across this way (indicating).

I have the language, the machine configuration requirements, time estimates, on when the Specs will be due, the rough language Specs due and the final language -- the final manuals, the preliminary manual and the primer reference.

If you are not familiar with what our basic principles are, how we put it together, the time for the work delivery, time for a workable system, delivery time for a good system.

The value of this chart is that there are a lot of blank spaces in incompatibility. The biggest planning space in the FORTRAN is, you see, there is absolutely nothing along that one, 1, 2 and 3.

Now, are there any GUIDE members here?

(No response.)

In this case, the IBM have a pretty nice position in this committee.

The 705 Model 3 is not so doggoned difficult or different from the Model 2, and since in order to do the Model 3 FORTRAN, we would have to put quite a number of people on the thing, and if we can do both projects at the same time by playing cagily, which I am sure we can do with the similarity of the machines, we should be able to invest three people into the project.

I think we can say at least four people from IBM, at least four people from IBM.

CHAIRMAN THOMPSON: Bob, I would like to ask a question. How many people do you think IBM would throw into this project if there were no industrial volunteers?

MR. BEMER: That's the four people I am talking about. In order to get any more than that we would have to hire people and I hope we can. If we are able to hire more people for this project, we will put them on.

MR. GEORGE W. KUSS (A. O. Smith Corporation): There are different deadlines though. The deadline for your model there would be about two years or a year and a half?

MR. BEMER: No, has anybody quoted a delivery?

MR. KUSS: Two and a half years. I understand the Government last summer made one earlier than that.

MR. BEMER: We will have to produce a system for the first machine delivered, whether it is the Government or not. So if, say, the Government got a machine in a year and a half or two years, we still have to have all the systems working by that time.

I have been thinking, estimating that if delivery on the Model 3 were somewhat a year and a half upward for the first machine and we could probably hope to complete this before nine months or a year. It is very close to simultaneous, and since we would take it for concurrent projects, the IBM commitment of four people should be just the same, no matter what.

MR. KUSS: If they are run concurrently, you could even do that.

MR. BEMER: A great many gadgets are similar in both of them and if you haven't violated any rules, then all we have to do is take the processor and to do two things to it.

One thing is to change the processor so it uses the Model 3 facilities, except the simultaneous output, and change the outward program that produces so that that takes advantage of it.

At least, then we would be safe.

MR. KUSS: Including the Autocoder subroutine. Then it would be a matter of changing the macros.

MR. BEMER: I am sure we will or should be able to have at least four people. I am not saying that we have four people put on immediately, but I will certainly work very much on it and Bill will give part of his time and Frank Williams will certainly give all of his time.

We can certainly pass that one.

CHAIRMAN THOMPSON: Don't you think it would add weight if we were able to set a date at this meeting for the first get-together of all of the people that are going to participate in this, plus all that are going to be working on it from IBM?

MR. BEMER: It certainly would. Not only a date, but a place.

CHAIRMAN THOMPSON: A date and a place?

MR. BEMER: Yes.

CHAIRMAN THOMPSON: And do you have any feeling for how many weeks hence this should be?

MR. BEMER: I think the timing depends on you people, when you can get commitments in your place and when you will be able to figure on whom you want to put on it and get rolling.

As far as I am concerned, we can do all we can do before we meet in a matter of two weeks.

CHAIRMAN THOMPSON: Two weeks?

MR. BEMER: Yes.

CHAIRMAN THOMPSON: Well --

MR. BEMER (interposing): I think the best place to do it, if it were possible, would be New York City, because we have all the people with FORTRAN experience there that could advise us during the initial phases of it in setting it up.

MR. KUSS: How long do you think we need for the group to get together?

MR. BEMER: I think it would be a week. I think it would be a week of damned hard work.

CHAIRMAN THOMPSON: Well, what is the feeling from the group, those that have made commitments or those that are just thinking they, perhaps, will come into it?

Is two weeks satisfactory as a target date for our first working committee to get together?

MR. BEMER: Here is a man that can authorize the people right now.

(Whereupon, at this time Mr. Dick Cline stepped into the room.)

MR. EDWARD B. BERNINGER (Procter & Gamble): Could you ask the people individually, you know, that are committed or almost committed?

CHAIRMAN THOMPSON: Yes.

MR. BERNINGER: If their companies are represented here?

CHAIRMAN THOMPSON: Yes. The Texas Company has committed a man.

The Eastman Kodak Company has committed one man.

General Electric has contributed one-half -- a woman.

(Laughter.)

Westinghouse Corporation has committed a man.

Standard Oil of New Jersey and A. O. Smith Corporation have committed two-fifths of a man.

MR. BEMER: Two-fifths of a man?

CHAIRMAN THOMPSON: Standard Oil of Ohio has committed an entire woman.

MR. KUSS: That's 20 percent of each time for two men?

CHAIRMAN THOMPSON: That's my list to date unless anybody is prepared at the moment to make commitments that they haven't announced to date.

(No response.)

MR. BERNINGER: I would say for Procter & Gamble, we can contribute computer time.

CHAIRMAN THOMPSON: That is good.

MR. KUSS: Who does the final assembly?

MR. BERNINGER: I am sure we can do assembly.

MR. BEMER: Where are you located?

MR. BERNINGER: We are in Cincinnati.

CHAIRMAN THOMPSON: I know that there are people that aren't at this meeting that will come forth very shortly and I estimate -- well, I won't make any estimation.

MR. WILLIAM M. SELDEN (IBM): I have a point of information. Two weeks from now would be the 30th of September to the 4th of October. What about the 7th of October to the 11th?

MR. BEMER: That would be very good.

CHAIRMAN THOMPSON: All right. It is the 7th to the 11th in New York City.

MR. BEMER: If possible, we will do it at some other place, if it is important, but I think for the initial stocking it would be advisable.

MR. SELDEN: We can travel, but all the people that wrote the original FORTRAN could not travel for a week just for speculation.

This would definitely be a disadvantage.

CHAIRMAN THOMPSON: This would be if it were -- I don't see why we can't decide on New York City between us right now. We have the various facilities there.

MR. BERNINGER: I am looking at the names of the companies who submitted their men. Kodak Company and the others are quite close, except The Texas Company.

MR. BEMER: You can't tell about the Texans in New York anyway.

(Laughter.)

CHAIRMAN THOMPSON: All right. Shall we take a quite vote on New York?

All in favor will say "Aye"?

(Whereupon, the majority of the conference participants indicated themselves to be in favor.)

And all opposed?

(No response.)

CHAIRMAN THOMPSON: The ayes have it.

MR. BEMER: Yes.

CHAIRMAN THOMPSON: All right. Then it will be in New York on October 7th to 11th and I will issue a general GUIDE letter as soon as I get back to Houston, inviting anybody to attend it that so desires to contribute materially to this project.

Now, let's back up to number two.

Dick, we have been talking about industrial commitments on this project. Would you like to say some words on IBM's participation?

MR. RICHARD L. CLINE (IBM): I have one person currently working in the scientific area.

A PARTICIPANT: Could you speak a little louder?

MR. CLINE: Louder?

A PARTICIPANT: Yes.

MR. CLINE: We have one person in this scientific area, and this individual is going to be planning the FORTRAN system for the Model 3.

Now, what I would like him to do is to work with you people during the planning stages for the FORTRAN system for the current machine.

This individual is Frank Williams.

MR. BEMER: You haven't been thinking of running them concurrently?

MR. CLINE: Pardon me?

MR. BEMER: You haven't been thinking of doing the whole thing as a single project?

MR. CLINE: Model 2 as well as Model 3?

MR. BEMER: Yes. That's the way I envisioned the project. There is nothing wrong with that. We just talk together.

MR. CLINE: I feel, at the same time, you plan the Model 2 FORTRAN, the Model 3 FORTRAN could be planned.

MR. BEMER: I think you could do more than that. You could make a common flow.

MR. CLINE: I see.

MR. BEMER: As far as the processing is concerned and making the plug-in, so that we just pull one out and plug another one in like a black box type of thing.

The black box for each thing is different for the two models, but there aren't really an awful lot of differences.

Further, the Model 2 FORTRAN will run on Model 3 if we take the minor precautions which we will take.

MR. CLINE: The input and output operations will be considerably different, but the planning through the flow chart stage will probably be the same.

MR. BEMER: Right. I don't see any reason why this thing can't be a whole blended project. Another reason it is good as a blended project is that it allows us to make better use of part-time men, where supposing somebody had a man they could put on full time for the 705 model 2, and he had this particular project, maybe somebody had a 30 or 40 percent man that they could put on the thing and he could be put on to convert that same piece to Model 3.

I think it would give us better use of partial assistants for that reason.

For that reason, I would be favoring that we would contribute our people towards both and that we are assured that we do it that way.

MR. CLINE: I will have the one individual to work with you in the planning area and this will carry through the flow charting stage.

MR. BEMER: Well, we will probably --

MR. CLINE (interposing): I hope to add one or two people to this area; as of now, there is nothing definite on that.

MR. BERNINGER: One point in talking about Model 2. Is this useful to Model 1?

MR. BEMER: We haven't settled that. We should go back to it when we talk about configuration.

MR. BERNINGER: I see.

MR. BEMER: I think, just to make sure that IBM does not fall short on this thing, Bill, Bill and I will probably spend a very great deal of time -- and we will spend a great deal of time until we can get some other people to help out on this thing.

If we do the actual coding ourselves, maybe one and, or one and a half programming.

CHAIRMAN THOMPSON: Well, it sounds like IBM participation is a little light. Dick, of course, I know you have personnel problems, too.

It seems like IBM has already announced that they are planning on a Model 3 FORTRAN and that you could throw in.

Certainly you were planning for more than one man later on in that development, I am sure?

MR. CLINE: Well, as I mentioned, we plan on adding people, but since the people don't exist as of this moment, I can't make any more definite commitment right now.

I do want to add, maybe, one or two people to this area.

4 MR. BEMER: If you are going to make a 705 FORTRAN sheet, we will have to add a lot more than this because it is now something like 25 to 30 man-years spent in 704 FORTRAN, and I estimated the 705 FORTRAN as 6 to 8 man-years.

You certainly couldn't do this 705, Model 3 FORTRAN in 6 to 8 man-years as an individual project.

And if you have two years' delivery date, that would be three to four people.

MR. CLINE: We are going to draw here on your people who are working on the FORTRAN system on the 709 and the experience that you people have, and also on the experience of this committee which is being set up right now.

MR. BEMER: Right, but --

MR. CLINE: (Interposing) So it isn't starting an entirely new project.

MR. BEMER: That's why we are not quoting 25 to 30 man-years or, say, 15 to 20 for the 705, being a similar machine, that is.

CHAIRMAN THOMPSON: I don't believe that industry should be expected to carry the major load on this particular thing, Dick.

I would be disappointed if IBM didn't permit even 10, 15 people.

MR. CLINE: I think the first thing that should be done is to sit down and review what has to be done.

CHAIRMAN THOMPSON: Right.

MR. CLINE: Yes. We have to get a good estimate of how long this is going to take and then go on from there, and I assume this is going to be done this week of October 7th?

MR. BEMER: Right.

MR. SELDEN: I assume we will be in better shape at that week to have comments on this because I believe that Bob and myself will be doing quite a little in looking into this matter before that meeting.

MR. BEMER: I get some other news for you, too.

(Laughter.)

CHAIRMAN THOMPSON: I think it is important for IBM to, maybe, even over-extend themselves at this point because it certainly is going to have an influence on the way the GUIDE letter, the GUIDE is going to be sent out, the letter that is going to be received.

If the tone of this letter is, "Well, industry is going to carry this ball and IBM is just going to sit back on their -- and use industry to do their work that they have already committed themselves to" --

They have already committed themselves to this at this meeting, and it is going to have an unfavorable reaction, I am afraid.

I think that, on the other hand, if IBM shows a hearty enthusiasm, that industry is going to react in kind.

MR. CLINE: Well, I agree with you on that point. Now, I don't know just exactly what in the line of commitments have been made so far by IBM. This was done where?

CHAIRMAN THOMPSON: This is being done right now by you.

MR. BEMER: We haven't done anything yet.

MR. KUSS: Originally, you said you would coordinate the project?

MR. CLINE: Yes. IBM would coordinate.

MR. BEMER: Well, I am very much desirous of seeing this thing going right. I think it is damn important, and we have to consider the way our programming is set up in IBM, specifically in this case between Dick and myself.

Dick is in charge of 705 program, per se, that is the period we sit on the other end, running the system.

Now, I do have people reporting to me and in emergency or if it were the proper way of doing it, of doing the things, we could put these people on so the project wouldn't suffer.

MR. SELDEN: I think that we are in IBM all agreed that this is a good project and we have possibly some slightly different views about the dangers of committing IBM positively to do something that we think it can do and feel that it should do, and then we can't.

CHAIRMAN THOMPSON: I don't see why you would have any qualms about this.

MR. BEMER: I don't have any qualms, mostly because I am brasher than Dick is.

MR. SELDEN: Dick doesn't have four people that can meet with you.

MR. CLINE: I think if it is a matter of 15 people, this is going quite high.

MR. BEMER: It is not a matter of 15 people. I think we should contribute four people on a full-time basis, at least, and possibly some part-time help from Bill and myself.

CHAIRMAN THOMPSON: Don't you think, Dick, as a result of this meeting, if it were determined that you ought to have 10, 15 people on this project, it would just be a matter of going to management and saying, "Gentlemen, this is what we need for this job, let's go out and get them."

I am sure the people upstairs would say, "Well, go and get them."

MR. CLINE: I think this is a case of determining just what percentage of the 705 users would use the system and how much they would use it. The commercial users -- the commercial users of the machine are the greatest percentage and, I think, this should help determine the percentage of effort that 705 applied programming is going to devote to this project.

MR. BEMER: Let me put in a couple of words here:

For one thing, this is a chicken-egg proposition. I don't think we can adequately determine the amount of effort of scientific work to be done on the 705's until we get a system they can use it with and sell it.

In other words, I don't think as much usage is being made of it unless we go out and talk it up a little bit. I really think you have to do that and present them something that they can use, and I think they will be damn glad to use it.

In the second place, we are not so much creating a FORTRAN as we are a mixed system, where not only the scientific use of it but the commercial use, the commercial user can come in and use FORTRAN, as well as, I think, there are many cases where you will find you have mixed application.

Certainly, the first to do something on this 650 was sort of on the borderline anyway.

The third thing is that Jack is not down at this meeting here, but on my way in, I stopped and took a shower at his room --

(Laughter.)

-- and I think he agrees with us on the importance of committing some people to such a project and that the projects will be done essentially concurrently.

So, we could split our effort this way

CHAIRMAN THOMPSON: Les, do you have something?

MR. LESTER W. CALKINS (U.S. Steel Corp.): In answer to Dick, I would just like to make a passing comment. I think that many managements around the country are bearing in mind that it is a company management as such, such as the 705's. They started out with high hopes of reaping a lot of dollar benefit from the 705.

Then, after they got into it for about eight or nine months, the hopes were still high, but they were somewhat lower after they started to face reality.

Then, after the machine came in and they found out actually what they could do, I think that the hopes were still high, but the picture didn't look quite so good.

So, I think, from the financial point of view, many managements around the country are starting to say:

"Well, why can't our engineering people now use these 705's with the full realization that some of the work that they do in terms of benefit would be tremendous by comparison to the number of heads ordered to roll."

So, there is a definite, a very definite feeling around the country now of the runners of this equipment to turn to engineering.

I think it is essentially that some thing be provided along this line to give them that vehicle, and I think it even goes further than that. I know that in United States Steel, in the case of United States Steel, we have some 650's and we have a 705, and we are starting our engineering effort now.

We want to select a language which they can get to use on our 650's to go with our 705's at the same time.

We want to be able to go to the machine that we think we would be able to use in the engineering, namely the 704, but if we don't have that vehicle and, with the accounting management turning to engineering, if we can't provide the system that we are talking about here, then we are in trouble.

Then, I think it is definitely IBM's responsibility here. I think they really owe it to the 705 people because the 705 has been more or less the pet project along the engineering line.

MR. JOHN B. SHEPPARD (Westinghouse Electric Corp.): Dick, I don't think it is so much the function of the amount of time used on the 705 as the value of the usage to the customer.

Now, I would say from some of my own studies so far that engineering time on the 705 pays off at least 5 to 1 as it begins data processing operations in our particular location, and these are the kinds of pay-offs we are looking for because it is very difficult to justify equipment if you are forced to do such justification strictly on data processes, data processing operations.

This backs up what Les is saying, but I have had very close and intimate relationship with this type of study for months now and I know exactly what this is meaning to us.

The engineering part of it, the scientific part of it, is a terrific lift to any justification and any "look-see" into future equipment that might cost more money to do a better job and so on.

We are all looking for it and we are all looking forward to it.

MR. CALKINS: To add one more thing, we have gone through a cost study now. The basis of this cost study goes something like this.

In other words, when you ask a lot of people how you are making out on your computer, they will say:

"Well, we are breaking even or are in the black."

What does that mean?

I think a lot of people are talking about the current savings versus current costs, but when you take the approach that you want to know what your accumulative savings are relative to your accumulative costs, I think you can count on your two hands the number of people that are really making out as a result of the study.

The only happy note was that we were not quite as shocked as we thought we were going to be, but when we start bringing the engineering people into the thing and some of the problems that they can perform, again backing up what John said, we are starting to show a far better picture.

MR. BEMER: Well, I can give you another thing on that:

Dr. DeCarl made a tour of various 705 installations. He was pretty frankly shocked at the same things you are talking about.

They weren't making any more profit then out of a machine that should be capable of a lot of profit. If you were thinking about just letting the scientific people go off on their own and just throw in applied science men to help the commercial people get straight on the profit, I have no doubt whatsoever that if we came up with a proposition, the doctor would say it on FORTRAN off 705 and no trouble of getting them.

MR. CLINE: I don't think there is any question here about the desirability of having such a program. This is why we decided to develop such program for the Model 3.

I think the whole point is -- the whole point is how long is this going to take?

How many people are the customers going to provide?

How many people is IBM going to provide to bring this to a successful conclusion?

MR. JAMES D. TUPAC (Rand Corporation): I think there is one point. As long as people are going to provide people, you should have gone on the presumption that the industry is going to provide nobody.

Make the assumption that industry wasn't going to provide any help at all and the fact that you have committed yourselves to putting out a system for the 705, Model 3, seems to me that in order to have something in any length of time, you must have thought that this is going to require eventually 6 or 8 or 10 people to get it finished.

Otherwise, the same thing would happen that is happening on the 704. It was two years after the delivery of the machine before something was out.

MR. CLINE: That's right, but this is a system for the Model 3 and we have not in the past gone back and done something for past machines that we are doing on the current machines.

If we did this, of course, it would require a tremendous effort.

MR. BEMER: That's exactly why we will be so grateful for GUIDE to provide this thing, but now I know we put in at least five man-years of effort on prints before we turned it over to Frank Williams for just the general ratings and prints and are nowheres near the magnitude of FORTRAN.

The way it has to be done, I think we have to double the thing, even starting from scratch, we have to throw in eight to ten man-years.

MR. TUPAC: That's right. I don't think you realize the magnitude of the job.

MR. BEMER: 705 people have never done a FORTRAN before.

MR. CHAMBERS: It seems to me not too long ago when we met at Poughkeepsie, some time was quoted there to us, something like 6 men, three man-years.

CHAIRMAN THOMPSON: Six to nine.

MR. CHAMBERS: Something previous, some previous minutes that my management has, and this sounded very encouraging to us.

MR. BEMER: I thought I said about six man-years.

MR. CHAMBERS: Well, you are wrong with that.

MR. BEMER: I hope I was not.

MR. CHAMBERS: But, evidently, something has come up since that meeting, primarily the Model 705, Model 3, which has amplified the system and that you want to make a little more powerful -- you want, perhaps, to make it a little more powerful than you had in mind at that time?

MR. BEMER: I think personally that the existence, the future existence of the Model 3, 705, demands that we have Model 1 and 2 for the carry-over into it.

MR. CHAMBERS: Absolutely.

MR. BEMER: We need that to get started.

MR. CHAMBERS: What I am trying to clarify is this: This is being increased. The previous estimate on the time necessary for the development.

MR. SELDEN: I think not. Pardon me, it is not for an estimate.

MR. BEMER: I don't think so, not for the original system. We are still talking six, possibly eight for the original system. We are talking about that and there would be a FORTRAN system for the Model 3 which if there were no co-operative effort, IBM would probably have to throw in another six or eight man-years of effort, possibly a little more, because it might turn out a more complicated process.

So that if you added them together for the two projects, this would turn out to 12 to 16 man-years of effort.

Now, knowing how the things go, I propose to almost put the two projects together and reap benefits of both and, perhaps, actually come within our estimate of the man-years.

MR. SHEPPARD: This was the question that was coming in my mind.

It certainly seems to me that FORTRAN was developed, Model 1 and 2, that it would be a much easier job, too.

MR. BEMER: Oh, yes.

MR. SHEPPARD: It would be a much easier job to convert it into Model 3. Maybe some could be direct conversion, maybe some could be some off-shoots from the Models to make it more powerful for the capabilities of the Model 3.

It would almost seem to me for the amount of time that you would put in the Model 3, F-5 language, that you could do it on the Model 2, and then carry it over.

CHAIRMAN THOMPSON: I think that for reasons of capability -- for reasons of compatibility on the current generation of IBM computers, that this should be considered as a crash project and that there are some very good reasons for getting out this Model or Model 2 FORTRAN just as soon as we can.

I am thinking from Bob's proposition that 6 to 12 months would be what we are talking about, and during that time -- since the FORTRAN language is already in existence -- there is no reason why training couldn't begin immediately and people could even start writing programs in anticipation of using their 705.

It could even be checked out by the 650 or 704 and be all set for the Model 3, 705.

MR. SHEPPARD: That's a very good point.

CHAIRMAN THOMPSON: I very strongly urge IBM to consider this as a crash project and as such throw in even over-commitments rather than play it coy and cautious.

MR. BEMER: Well, let's put it this way:

By the time everybody shows up with their personnel from October the 7th to 11th, I think, in fact I would about personally guarantee that the doctor will see that we have enough to make the GUIDE people entirely happy with the sharing of effort.

CHAIRMAN THOMPSON: All right. But for the purposes of this letter, soliciting further industrial help, what number can IBM authorize me to state as a reason?

MR. BEMER: When do you write the letter?

CHAIRMAN THOMPSON: As soon as I get home.

MR. BEMER: I can make a telephone call today, if you like.

CHAIRMAN THOMPSON: All right. Let's get it. I think we should get it settled today and make it on the high side, if you can.

MR. SHEPPARD: I know that from our standpoint, the only reason that Wally Chambers is here, is to make sure that we can come to a definite decision today as to how we are going to proceed on this.

If we are going to let it hang over to another GUIDE meeting or a special meeting somewhere else, we are way off beam.

I think we have to come to a decision today as to what to do, committing ourselves on this and get going.

CHAIRMAN THOMPSON: I don't know if you came late, John?

6 MR. SHEPPARD: I realize, I realize that I did.

CHAIRMAN THOMPSON: We have already decided that the first working committee will assemble in New York, October 7th to 11th, for the purposes of drawing up their sleeves and getting down to work, probably spend a week -- perhaps even more.

MR. BEMER: Yes. I got an idea. If you would like to have Bill Selden outline some of the various processes of FORTRAN so that people here that aren't too familiar with the operation might see the overall standing, the breakdown, the organization, the number of passes through the machine, the arithmetic standing and statements, how they would be accomplished in macros, the final merging --

I could excuse myself a bit here and come back with some reasonable work and word.

CHAIRMAN THOMPSON: All right. Why don't you do it? Bill, would you like to take over?

MR. SELDEN: I would. I would be very pleased to. I would like to start -- Incidentally, do you wish the stenotype transcript of my remarks?

CHAIRMAN THOMPSON: Why not? He is here.

MR. SELDEN: I am not afraid of being quoted, by the way.

(Laughter.)

I would like to get some indication of what you want to do. Do you want me to describe the 704 FORTRAN or 705, or would you like me to try both?

How long do you want me to take?

CHAIRMAN THOMPSON: You might want to consider that this is coffee break time and, maybe, this might be better starting at 10:30.

MR. SELDEN: All right.

CHAIRMAN THOMPSON: Let's break for coffee.

MR. SELDEN: Just a moment -- Oh, I'm sorry, please finish your remarks.

CHAIRMAN THOMPSON: During coffee break, decide in your own mind how best to present it and we will leave it to your judgment, as something between half an hour and an hour.

MR. SELDEN: Is half an hour or an hour going to be too long?

CHAIRMAN THOMPSON: It sounds like being very reasonable to me.

MR. SELDEN: All right, thank you.

- - -

(Morning coffee break.)

- - -

CHAIRMAN THOMPSON: Let's get started, Gentlemen. We can't run over 12:00 o'clock.

I think the most interesting item of business at the moment would be the results of Bob's call to New York.

MR. BEMER: I didn't call New York, didn't have to.

CHAIRMAN THOMPSON: Before Bill Selden takes over, we would like to know what you found out.

MR. BEMER: Listen to me, so I don't say anything wrong.

(Laughter.)

I will say it one way and you can correct me if there are any small knots in meanings. We don't know yet how many people it would take us to produce a 705, Model 3 FORTRAN system. We will start to determine that when we get back to New York because, after all, the machine has been only announced a very short time and we didn't have a chance to get together on it.

If one would consider the total expenditure of effort all the way along, IBM will commit at least the people it would have to do or would have to commit for the 705, Model 3 FORTRAN project, plus probably some extra help, if necessary, on this thing.

Now, when you merge these things together, we still have no clear definition of what will be valuable one way and what you people would feel is valuable the other way.

But as far as the commitment of personnel, we would -- the least we would have to do, if this whole project would have never come up, we would have as many people as if this whole project would have never come up and probably more so.

We will be in no way unfair to GUIDE, we hope.

CHAIRMAN THOMPSON: Well, speaking for The Texas Company, that sounds fair to me. If you pretend we are not even going to be around and you would throw in a number of people that it would take without us, I feel that we couldn't ask for anything more.

MR. BEMER: This is the overall project.

CHAIRMAN THOMPSON: Yes.

MR. BEMER: Right.

CHAIRMAN THOMPSON: George, did you want to say something?

MR. KUSS: That will take two or three years to get it.

MR. BEMER: It is probable that most of the people that we throw in will be considered on the Model 3 project. That's where they would go anyway as soon as we start working on it. We go back and forth between the two and this we would be able to work out in more detail in New York, and then we would know where we stand.

I am certain that we would not, in any way, be falling short in our commitment -- our responsibilities, I should say, to be fair to GUIDE and the whole business.

I think that's what we are really trying to say beforehand.

CHAIRMAN THOMPSON: Well, when you mention one man -- when you mentioned one man, it shocked me a little bit.

MR. BEMER: No.

MR. CLINE: That's one man we have now.

MR. BEMER: That's the original planning. That's in addition to that. Bill and myself will also help in the original planning.

When we meet in New York, we will be at least three people from IBM sitting in this thing.

CHAIRMAN THOMPSON: All right. Has anybody any further comment?

(No response.)

CHAIRMAN THOMPSON: Well, if nobody has any further comment on this item, I suggest that we move on and turn the floor over to Bill so that he can fill us in and give us some rough ideas of what is involved in this job.

- - - - -

## ADDRESS BY WILLIAM SELDEN

- - - - -

MR. SELDEN: Thank you. I don't know whether it is best up here or on the other end (indicating).

CHAIRMAN THOMPSON: Wherever you wish.

MR. SELDEN: I hope this will do.

CHAIRMAN THOMPSON: Fine.

MR. SELDEN: I would like to start with some definitions which will probably make this a little bit easier.

I would like to say that we ask questions if I am not making myself clear.

We talked about FORTRAN as a language. This is the expression of key-punching or arithmetic symbols and various additional symbols that you think as a FORTRAN language program.

The general language is the same for all computers. We have at the moment some detailed differences between the language of the 704 FORTRAN and the language of the 709 FORTRAN.

While we can say that F-4 language will be slightly different from F-9 language, but that the language for FORTRAN is common to both of them and, incidentally, we expect ascending compatibility through machine processes.

F-9 language is richer, has more expressions in the language than the F-4 language.

We speak about the object program. This is a program in the native machine language. 704 binary, 705, the "H" and all the five characters, 650 words or whatever this is.

This is the object program and we would like to speak of the FORTRAN system as the FORTRAN executive or the processing system of processing.

The 704 FORTRAN has several parts. I will describe this program:

I will describe this program not in terms of the language, but in the terms of the processor. By the way, there is a description of the paper that was presented in California, of the paper presented in California having some description of FORTRAN.

I didn't bring any copies of that with me. I regret that. If you are not familiar with it, we might pass a paper around and if you put your name and address on that, I would try to see that this got mailed to anybody that wanted it.

Would somebody start a paper? I can only say that this could be mailed out if it is still in stock.

MR. BEMER: Yes. We have quite a few of them.

MR. SELDEN: The FORTRAN is divided. F-4 is divided into six sections.

The first section is in two parts. There are various other peculiarities. The first thing that FORTRAN does is to read instructions, cards, translate the cards into symbolic instructions.

Then it examines the program that it has written. It is in symbolic language, to increase its efficiency, and then it transfers to the left the section which is essentially a SAP assembly program.

7 In our case it would transfer it to the Autocoder program. The first section of FORTRAN has two states.

The first state is the arithmetic scan. This occupies approximately a full man.

The second phase is the second state of the first section, and that is the portion that deals with non-arithmetic statements. It goes to format statements and so forth.

The second section of FORTRAN deals with the problem of Do's.

Do's is a section of coding modified by indexing.

The third section of FORTRAN merges the output of the arithmetic and non-arithmetic statements.

The fourth section of FORTRAN operates the resulting program in a "Monte Carlo" technique to determine the most frequently used branches.

The fifth section of FORTRAN, having discovered the most frequently traversed branches, alters the coding to be more efficient.

And the last section is the symbolic assembly, equivalent to the 705 symbolic.

A problem is taken in the optimization features and seems to us that the characteristics of the 705 are such that optimization will pay off much less in a 705 than in a 704 and 9.

So we are proposing for this project, for this project at least, a processor which does not have any very complicated -- even if any at all -- optimizer section.

MR. BERNINGER: That would be the sections 4 and 5?

MR. SELDEN: 4 and 5 would have no equivalent and in the F-5 language, in the F-5 processor.

MR. BERNINGER: Is this due to the fixed FORTRAN?

MR. SELDEN: The optimization in Sections 4 and 5 optimizes the assignment of three index registers.

Our thinking is that where indexing is used in the 705, we will use memory positions and have a fairly large number, perhaps 10 or 20 index registers.

MR. BEMER: Yes.

MR. SELDEN: And with this number, the increase of efficiency to optimization will be negligible.

MR. BEMER: This is sort of occurring,if you only had one index register you have a very definite time working everything through. You get more and more and more things that come in in an infinite number of index registers. You don't have any problem at all because you have something for everything.

However, a number of 10 or 15, like that, that does not expand too much memory for registers.

At the same time, it should provide, let's say, 99 percent of that infinite efficiency.

MR. SELDEN: I would also say that future FORTRAN processes are planned with the ability not to optimize a program and to first process and in many instances will not have optimizers, but in every case we think it is worthwhile to be able to avoid the optimization.

So, we don't think we are losing anything here.

The language of FORTRAN, next perhaps we have, and we have mostly completed and are finishing working on the proper size specifications for the F-9 language.

What improvements in the basic FORTRAN language will be present and at nine this is a matter of offending compatibility.

F-9 is a richer language than F-4. I hope that our project will encompass writing a program. If F-5 language is identically equal to F-9 language it can be done.

The one characteristic of F-4 processor is that it deals with the floating point, binary words of six characters in length, primarily dealing with binary of half-word lengths for indexing.

We do not regard this as a property of the language, but a property of the processor. It will be up to this committee in part to decide how many word lengths and what fixed point and floating point facilities you wish to put into FORTRAN F-5 processor.

I add here parenthetically that the way I envisage -- we envision -- I am not the only one that is doing it -- the way we envision the F-5 processor, it will be quite easy to add different word lengths and make the floating point or fixed point or whatever.

The Autocoder assembly system, I assume, would be used at the tail end of FORTRAN F-5. This would have to be a somewhat modified Autocoder in that, at least, system control would have to be changed.

There are possibilities of doing more of the work which is done in F-4 processor in FORTRAN via macro instructions in the Autocoder.

There is the possibility of improving the Autocoder in some areas where a small change to the Autocoder could represent a considerable saving in the work in the FORTRAN processor.

This is the sort of detail that I think we get into three weeks from now when our sleeves are rolled up.

The arithmetic scan of FORTRAN is the beginning point of the FORTRAN project to translate formulas. This is the best documented section of the FORTRAN.

It is the one we know the most about in terms of ideal float listings, float charts, and so on. Some of the other bits are a little bit full of --

I don't think in any case we wish to transcribe the coding. The method used in 704, FORTRAN, is readily available.

I have one copy of the paper describing it with me. The people that are writing F-9 processor have looked at this paper and preferred to write the F-9 processing section from this paper rather than from the machine.

So, I think we can assume that this is well defined.

There are other techniques of scanning formulas which will produce the same result that the arithmetic scan of F-4 does and might well be better adapted to the 705, which with its single character arrestability.

The man that wrote F-4, Pete Sheridan, is aware of these other schemes and before I came out here I discussed it with him and he is planning to talk with whoever writes the section, discussing his method and alternatives.

I think that whoever is assigned this one or one of these, one of these two persons' jobs I was talking about -- this pair of people, I think, should be given format of their import and told what theywant out in terms of macro instructions and be told, "Go to it."

The rest of us would not be involved in this, in which rules of the formula translation they use.

The process of the DO loops, the repeating back and forth in the indexing, is not quite so well documented, and there is no neat rule for this.

It is a collection of about a dozen rules with a few exceptions.

This will probably be well to follow here exactly in the approach of 704 FORTRAN, and that we have already had several people spending several months trying to find some other approach that would work -- they don't.

Maybe, if we were engaged in a theoretical study, we might all try to do it differently, but if we are working in a minimum time to get something out, I propose what we just follow what was done in this area.

The merge, and we will have to have an equivalent, I am sure, the optimizer will probably not have an equivalent in our initial thinking, but we should leave a gap where we can put one in in our ideas of how it should be set up, and we will have an Autocoder at the end.

Now, there are some further characteristics that I think we ought to discuss. I hate to say, "I," all the time. It is mostly the departmental agreements.

I think that we should not have two states in section 1. They should sort the source language into different tape files of the type of statement that is involved.

One type would be arithmetic formulas, one type would be DO and repeat, one type would be Autocoder language instructions.

We should then process each type of instruction, each class of instruction with a specific processor, one after the other, and, of course, the Autocoder language instructions simply get left on the side until the Autocoder comes in.

MR. BERNINGER: I have a question here.

Does this mean then that in the midst of algebraic formation, you can start writing in regular Autocoder?

MR. SELDEN: Not in the middle of a single formula, however.

CHAIRMAN THOMPSON: Wouldn't this detract somewhat from the featured compatibility?

8 MR. SELDEN: It absolutely kills it.

MR. BEMER: Wait a minute. This is the thing you write in the Autocoder yourself. Part of your program is a feature of F-9 language. It is not a feature of the F-4 as it exists.

The F-4 model 2 which is just starting to be written -- F-9 allows it to make a subroutine in either machine language or FORTRAN language and named; and later on that certain thing. It all depends on what language we use here and where this comes in.

In any event, since a subroutine written in Autocoder language must be called for by name, it doesn't detract from your compatibility because if you ran on another machine, you would have to code that subroutine in that machine language to correspond to it with getting the same name for that purpose for that machine.

MR. SELDEN: Yes. That is very true. Many ventures into Autocoder language will be ventures into macro instructions, and subroutines which can have equivalent functions in the Autocoder system for the other processes and add to memory instructions; for example, it would probably be very difficult to equate to any single 709 instruction.

CHAIRMAN THOMPSON: So, it doesn't kill compatibility if you are careful in the way you mix it up?

MR. SELDEN: In the general case it kills it; in the specific case, you can do it.

We wish to narrow down the area of Autocoder language statements. We wish to urge against their use; we wish to enrich F-5 and F-9 so that people will not be tempted to use them.

MR. BEMER: This is the same problem you are running into, what you are going to pay for optimization all the way through.

If you have, for instance, a primary dictionary and a primary way of doing something, say that you do it this way, the machine runs very quickly and will process yours rapidly, but if you do it the other way, it is going to take the secondary thing which is going to cut on your time and not be an efficient program.

The man has an option. Anything he does will work, but if he plays ball with the primary rules, he will make profit out of it.

The same thing occurs here. If you are very careful to make sure that everything you do is done in FORTRAN, no matter how crudely, you never have to worry about writing a subroutine to match the name in any other machine.

It is just a question of the price you wish to pay. If I write a subroutine in FORTRAN language and then use it, I have no problems at all.

If I write it in machine language and name it as a FORTRAN function, then when I run up against doing this on any other job, it will run into a stone wall and I will say that I haven't got this.

In that case, you will immediately have to go out and code that for that other machine, and it may be difficult remembering what you meant last year.

I don't know but one must code in machine language -- if one must code in machine language, you've got to pay a penalty on the compatibility.

MR. KUSS: Can I ask one question?

MR. SELDEN: Certainly.

MR. KUSS: In 705 typewriter and 650 and 704, a typewriter is a very valuable thing and we use it, we like to use it.

How do you take this into compatibility?

MR. SELDEN: The 709 standard machine has a printer on line for communication for the programming.

MR. KUSS: Which will be converted to the printer. Now, what do you do for compatibility in FORTRAN? Do you just delete that?

MR. BEMER: FORTRAN will just cause a sub-set of FORTRAN language; in other words, you cannot write an F-4 program into F-5 and expect to write on a 650, GED. Tom. You have to expect that or don't write in the first place.

MR. SELDEN: 650 with tapes and typewriters attached, which is a complete 650, could handle this problem.

I believe you could buy a 705 without tape drive and you would have a bad time.

MR. BEMER: I have a solution here. You know, I marked down "flag removal" for fast reassembly.

This is in connection with your patching and snapshot diagnosis, where you are writing a FORTRAN program and you suspect it may not run the first time because of some error in your coding or perhaps overly pessimistic on your part.

I do this so you put in snapshot statements. People use this and say, "I would like to see this answer here and this answer here, and this will give me some sort of a clue how far I ran along correctly."

If you identify this thing as a special type of statement in the processor in the first place, it can put it in as patches. Then, when you later on delete it, it doesn't go on reassembling the whole thing up to that point. It just pulls the patch out. This is possible.

Another place that we should take advantage of flagging -- and this could be done automatically as the preliminary program goes into the first time -- it is classified as to arithmetic statements, the various types of processes simultaneously, and we could look immediately to the operators and classify these as certain types of statements.

Then we would have with each processor a dictionary of the statement numbers or of types which are acceptable to that machine.

Further, we could join with this the dictionary of 704's or 650's of various types. This would be a very simple matter.

And, if you so desire, at the end of any processing, it could be pointed out to you that this program will not work on a

704 because of this and this statement.

I think that's a very simple matter to put in, don't you, Bill?

MR. SELDEN: Well, I wouldn't say that it is very simple. It can be done simply.

MR. BEMER: I think it is.

MR. SELDEN: We do feel that processing systems that IBM produces can no longer be expected to be finished and this is the end of it.

You tighten in the edges, and that is wrong. There has to be room for change and modification.

A great deal of the thinking on F-9 has gone into the problem of keeping the F-9 processor open-ended, so that more can be added to it, and writing, coding in a manner that it will be undestandable to somebody else taking it up later on.

This, we feel, is intensely important. I am sure all of you will agree with this who tried to change the program somebody else wrote eight months ago.

I would think that the output of the earlier sections of FORTRAN would primarily be a system of macro instructions and these macro instructions would go to the Autocoder and be processed.

I do not think that these macro instructions would be the same macro instructions we now have in the library.

I believe they should be a package, a system of macro instructions that is meant to work together, but is not meant to be particularly useful or helpful to anybody just wanting to use one for some reason.

I believe that the final object program coming from the F-5 processor will be a large volume of subroutine and in the area of data, but the effect of this will probably be very much the same as the effect of the print.

There is a very tightly coded package of subroutines and things in the interpreter and I believe that FORTRAN should produce some set of macro instructions which produces a similarly -- a similar package.

Now, this goes along with the idea of separate projects very definitely. We have complete precedents for separation of the object program and the macro instruction library, whatever you call it, from the prior processing that was our print that worked, that was our print that worked and there was no problem there.

The problem of this processor, I hope, will look -- I'm talking now about Model 3 processing in particular, and it would somewhat be in the program as if you were writing for a combination of FORTRAN and the Autocoder.

That will be a method of writing a program completely in Autocoder language but also be a method of requiring or writing a program completely, avoiding the Autocoder so that you will have essentially one assembly system.

Now, this is something that is probably not part of the immediate thinking of this group, to go into at this time, and it may be somewhat beyond the scope of this group to work on, to work on a major modification of the Autocoder for the Model 2.

Are there any questions? I talked about three-quarters of an hour.

MR. ELDON C. DODGE (Stanford Research Institute): What do you expect to do in regard to the format statements that are in F-4 input and output? Will that be similar, will that be similarly as flexible as F-4, for example?

MR. SELDEN: That is the most knotty problem we are looking at now.

MR. DODGE: I thought so.

MR. SELDEN: I need a format statement which is more understandable to get ascending compatibility.

We have a problem unless we keep the old format statement available. The seven F-4 format statement is so powerful in part because it depends in its operation on the characteristics of the copy loop and does not produce an efficient method of reading in on, out of the 709 or 705.

9 MR.BEMER: I have an idea here:

This problem is akin to the one we having moving between the F-4 and F-9 language where we require pre-processors.

Now, any deviation we would make from the F-4 language would require that we would pre-process on the 705, a source program meant for the 650 or 704.

This is probably a very fast-eyed process. Assume we wrote the F-4 language for the Model 2, 705, and assume further we wrote the F-9 language -- F-9 language for the Model 3, 705.

Then, compatibility between the two would have to be achieved by a pre-processor. It takes the source program you wrote this year for the Model 2 and converted to a source program for the Model 3 next year.

Now, this is a relatively simple problem actually because there is just a pretty much one to one correspondence.

We don't envisage any trouble at all between the 704 and 709.

Now, you can consider that even with identical languaging -- not identical, but almost identical languages, you might have the same problem.

Supposing we kept to the F-4 language for the Model 2 and we want to run something that was written for the 704 which had this particular type of format statement.

We might consider processing the 704 source program so that it produced a variable format statement which we would use for the 705 only. Then, all 704 users would have to have a pre-processor which converted format statements of our type back to the 704.

Now, this is a possibility, but it is not one that I like. I am just tossing it out to show you where we have the difficulties of compatibility.

I am also showing you that if you had any compatibility at all it might be well to make the big jump on the Model 2 and make the F-9 language for the Model 2 in which case your language would be identical between the Model 2 and the Model 3, except that you would have to use this pre-processor for the 704.

Now, if it should turn out --

CHAIRMAN THOMPSON: (Interposing) Excuse me, Bob; hasn't that already been written?

MR. BEMER: What?

CHAIRMAN THOMPSON: The pre-processor to go from F-4 to F-9?

MR. BEMER: It is being written for the 704 - 9 series. It will have to be written for the 705 series. If you write two languages, F-4, that bear the same relationship as the 709.

Now, if the format statement gives us considerable difficulty as it exists in the F-4 language, it will cause us then to come to a pre-processor for the F-4 and 650 program as it is.

Now, if you are going to have any pre-processor at all, you might as well have one, I think, that will do the work and not only change your format statement, but change the F-9 language so that you immediately, exactly are compatible with the 709 and with both models of the 705.

MR. SELDEN: I think there is no problem in F-4 to F-9 conversion, except the format statement.

I may be wrong in that.

MR. KUSS: F-9 requires copy loop, too?

MR. BEMER: No.

CHAIRMAN THOMPSON: No.

MR. BEMER: There is no such thing as a copy loop problem, fortunately.

MR. KUSS: When you go, I mean you require a complicated statement for the F-9 as you do for the F-4?

MR. BEMER: We are going to considerably ease up the F-4 format statement on the F-9. This is one of the statements that has not been completely specified in our F-9 language.

In fact, it is about the only remaining one we don't have because -- well, I don't like the F-4 format statement.

(Laughter.)

MR. SELDEN: I would like to suggest that we leave the precise details of the format statement for our meetings in two weeks from now at New York. It is a lower level.

MR. CHAMBERS: (Westinghouse Corporation) How about pre-processors for 705's, to 704, 705, to 650?

MR. BEMER: You need them in the averse direction if you had difficult format statements or anything difficult.

MR. CHAMBERS: Definitely.

MR. BEMER: For 704 you probably wouldn't have any difficulty because it is about the same power as the 705.

If you want to try to do this on the 650 equipment for transit without tapes, you couldn't do it no matter what you do with the pre-processor.

MR. CHAMBERS: What I mean is -- would it not work with F-5?

MR. SELDEN: For transit language it would work on 5. You could write a program using those portions of F-5 which were available and for transit and have no problem.

You would get as efficient an object program which you would find in most time writing.

A major specific difference here is the limitation on the number of columns of alphabetic information on the 650.

MR. CHAMBERS: In other words, it is primarily due to the fact that you enhance the source language?

MR. BEMER: Yes. The whole premise here is that we do not confine ourselves to what we did the last time.

If we know how to make something better and really cut down the coding effort we will do it, but we will make the old one compatible with the new through the pre-processor.

MR. CALKINS (United States Steel Corporation): Would you take any power away from the F-4 transit?

NEVER USE THIS PHRASE

MR. BEMER: Yes, we do, by God, we are cutting away from the 704 FORTRAN to statements which can be done by two other statements which presently do not exist.

We are not taking any power away. We are not taking any power away -- we can't.

MR. CALKINS: We can't?

MR. BEMER: That's the unwritten law.

(Laughter.)

CHAIRMAN THOMPSON: Looking to the future, Bob, wouldn't you recommend the F-9 to be identical with the F-5 for the pre-processors?

MR. BEMER: I would. Between any of the machines -- not machines, between the languages, one will have to have a pre-processor, but if the F-9 language is written and used that way for the 705, Model 2 and 3 both, you will have three-way compatibility between both models of the 705 and the 709 without any pre-processing.

This seems to me the most desirable way to do it.

I think we avoid a lot of bogging down type of detail by doing this.

It does mean this, though: This is the place we have to caution, that if during the period of running 705 problems in the F-9 language, you wish to put this back on the 704 and run it -- not on the 709, then your 704, now, that becomes a machine with theoretically less capability than either your 705's as far as FORTRAN is concerned, and you would have to have some means of going back to the F-4 language.

In other words, you would have to have a pre-processor in the downward direction which would not only convert the F-9 language, which then seems identical to F-5, down to the F-4, but you will also have to flag some statements in the F-9 that are not possible to the F-4 FORTRAN.

That's the cautionary problem.

MR. BERNINGER: In other words, this would give you some descending compatibility?

MR. BEMER: You could get a descending compatibility to the 704 by virtue of the pre-processor.

MR. KUSS: Rewriting some statements?

MR. BEMER: And a few statements that would be flagged before you can possibly run it.

I don't consider this a tremendous objection because here is the way I would do the things if I had a 704 program.

Now, let's say that I want to go up to 709, I dropped the cards in the hopper, my old source program runs through the pre-processor, and I now have a program that works on the 709, but really doesn't take any advantage of the 709 because it does the same doggoned thing the F-4 language did.

It is only converted in the superficial details of the subset on the F-9 language.

I would then look at this list and see -- "Well, there is a place where I can insert a better type of format statement, here is where I can knock off the whole section of coding because it is really a section I can save, that I can name once and no use repeating in the source language generally, and I say that I trim it down and add a few statements till I fix it up properly."

I have the option. I can run it as it is in no more efficiency as I got in 704 -- I can fiddle around with it in a language familiar with and add new statements and delte certain sections until I can really help it a lot.

But I haven't ~~restored~~ destroyed compatibility.

Now, I think that these pre-processors are not very difficult things to run or wouldn't be.

They will be quick for one thing and, once you have processed this program for another machine, you have now a source program language, a source language program for that machine, and you never have to do it again.

I would say that if I have a library of FORTRAN source programming in my 704 installation, I am going to get a 709 and shoot the 704 out, even if I am not, I will take that entire library and process it once and for all and produce a corresponding library of F-9 source programs, and then I won't ever have to use the pre-processor again.

As you can see, ostensibly, it is a one-type thing.

MR. SELDEN: I don't see why you do it at all, but then --

MR. BEMER: (Interposing) I don't see how you can get away without doing it.

MR. SELDEN: We obviously are not in complete agreement here.

(Laughter.)

MR. BEMER: You think that's any surprise?

(Laughter.)

MR. SELDEN: You have to translate the format statement or statements in the ascending, but if you have a more advanced matter available in the F-9, the improvement in the F-9 language will result in the same machine code at the end.

MR. BEMER: We are talking about superficial details such as card format, the way the instructions are actually written as an intelligible alternate.

MR. SELDEN: I agree -- Sorry, oh, yes.

MR. BEMER: An amplification on this alternate to process, it was incumbent, it is the thing to recognize, the old language as well as the new language. You turn a switch and say, "Okay," old language and new language. I propose to have a little separate processor here, a one-shot deal that converts, and we never have to worry about cluttering up the processor from the old into new.

MR. KUSS: Do you think it is possible to write a preprocessor to convert a ~~quick~~ PRINT program to F-5 language?

In other words, we have a large body of subroutines at the moment and --

MR. SELDEN (Interposing): I would think so. It is mostly three-address instructions that could be a three-element format.

MR. BEMER: I hate to be quoted on this right now, but it sounds intriguing. Yes, it might be capable. It would produce a presumably stupid format, but at least you can run them until you can recode them in FORTRAN or polish them up in short program.

This is a definite possibility because when we are looking through something and index register tags, it has a great similarity to the DOS and it would probably come out of there, yes.

That's for repeating some operations and we probably get that and, well, it seems very reasonable because, really, print is like a sub-set of FORTRAN. It lacks the full arithmetic scanning capacity of creating a program for a format that you yourselves have not broken down.

A PARTICIPANT: Do you have a print program which includes a great deal of symbolic?

MR. BEMER: All you can do in that case --

MR. KUSS (Interposing) -- is rewrite it.

MR. SELDEN: In the Spec instance of 705, I believe that the prints symbolic can easily be equated in Autocoder.

MR. BEMER: That's true, but he means the broken-up section.

MR. SELDEN: But it wouldn't go to 704.

MR. BEMER: The pre-processor should be able to recognize the print instructions and differentiate from the 705 symbolic instruction and, therefore, could group the sections and only process the print sections itself.

In this case you have to do something else about the machine language.

MR. KUSS: The Autocoder is just a column?

MR. SELDEN: Not a column, but very closely.

MR. BEMER: It still requires a subroutine.

MR. SELDEN: I am not tired of getting up and watching the conversation -- if there are no other questions, I am about through.

(No response.)

CHAIRMAN THOMPSON: Thanks very much, Bill. I would like to recognize John Sheppard.

John Sheppard has a couple of things he would like to say now.

MR. SHEPPARD: Well, the thing I would like to bring out is that I think as much as we are talking about this here in the last few minutes has been more probably subject for this New York meeting.

I think the thing that we are all here for is to come to some definite understanding as to the commitments, both as to the GUIDE and IBM in this whole determination.

I think we would like to go away from here with a very clear idea of where we stand on getting the thing started and how GUIDE and IBM are going to work together on it.

And the other thing that I would like to suggest, which probably would have to be taken up, that the minutes of this meeting would be made as immediately available to the members of this group, of the companies represented in this group, as possible, and not waiting for the general distribution of the proceedings because I think this is a much more needed subject than the general proceedings of the GUIDE conference here.

CHAIRMAN THOMPSON: I don't personally see any advantage in trying to edit an edition of this meeting since we can just send out the unexpurgated version.

We should be able to get this out pretty fast.

I will check with -- I guess I have to check with Tom Ford on that.

Now, as far as the other point that you bring up, John, on leaving here with a very clear idea of what is our plan or what our plans are, what the commitments are -- I think we are all pretty firm on the fact that we do have this first working committee meeting set for October 7th through 11th.

A letter is going to be issued by myself as soon as possible, making this known to not only you -- you already know it -- but also to other people that were not at this GUIDE meeting, so that they, too, can join into this project.

The people that have been committed already by industry have already been enumerated.

IBM has stated that they plan to throw in the same number of people on this project that they plan to throw into the FORTRAN for the Model 3 just as if industry would not be providing any people at all, is that right, Jack?

MR. JACK T. AHLIN (IBM): Will you restate our commitment again and make sure that I can understand it?

CHAIRMAN THOMPSON: You have already committed yourselves to create FORTRAN for the 705, Model 3?

MR. AHLIN: That's right.

CHAIRMAN THOMPSON: And you would want to get this FORTRAN system ready in time, at the same time that the machine is ready, which is, whatever it is, two and a half years, I believe it is, that is the figure that has been used?

MR. AHLIN: Yes.

CHAIRMAN THOMPSON: I don't know where it came from, but in order to turn out a FORTRAN system for the 705, Model 3, and have it ready in, say, two and a half years, you would need 10 to 15 people, or whatever you decided?

MR. BEMER: Some undetermined number.

CHAIRMAN THOMPSON: Some undetermined number?

MR. AHLIN: Yes.

CHAIRMAN THOMPSON: You turn this force loose as soon as you can get together, and for this project?

MR. AHLIN: That's what we are doing. Ordinarily, if we have a project which requires, say, I am just talking about 15 people at the peak of activity, we do not assign 15 people on date to this project.

I am sure you don't do this in your own operation. We phase in the people as they are needed as much as the initial work is of planning nature, and, frankly, we don't have 15 people that could be of great assistance to us in this area.

Generally, however, your understanding of our commitments are 100 percent correct.

CHAIRMAN THOMPSON: Does that satisfy you, John?

MR. SHEPPARD: Yes. There is only this one question as to possibly what plans there might be between GUIDE participation

and IBM participation in this endeavor, and I don't know whether
11 you are in a position to know this at the present time?

CHAIRMAN THOMPSON: Well, it sounds like this will vary at the beginning. It sounds like it is going to be 80 or 90 percent industrial and then the shift will swing to IBM more than ours, and it might continue that way.

MR. BERNINGER: The same point I think would be good for us to get at least a good target date as to when the Model 3 FORTRAN is scheduled to be ready.

This, of course, ties into the date of Model 3, when it is going to be shipped.

MR. BEMER: That's right. We could possibly give you a guess on that as to the delivery of the first Model 3 system.

MR. AHLIN: I don't think that is what we are interested in.

MR. BEMER: Pardon me?

MR. AHLIN: I don't think that is what we are interested in. I think the basic point here is one of the fact that we, I think, at this meeting, have fairly clearly stated what our objectives are in and they are in applied programming and what our means of achieving these objectives are.

One of our objectives is as we have indicated, to supply for the Model 3 a FORTRAN system within our basic framework and an attempt to make these things available in time.

In addition, we are anxious to be of literal assistance in what we can do in supplying the 705 FORTRAN for 705, Model 3, and

I am afraid that if we would make any specific commitment here, it would be doing a disservice to you because it would be a commitment or a statement, indeed, without proper information.

I don't think you want that type of thing.

The fact that we say that we are going to do this doesn't mean we are going to do it,because there are many slips that occur, as you probably realize in your own activity, and the fact that we are anxious to do it, is, I think, what we want to have, and we hope will be of consideration and interest to you.

I think we have the know-how and have the desire to do the projects, and with your assistance, I think, we are getting this thing off properly.

CHAIRMAN THOMPSON: I have an announcement to make.

If anyone has any change in the registration, a mistake in the company, name, or anything, Bill Smith requested that if you are aware of it, let him know. They are going to get it out for publication right away.

Les, do you have something to add?

MR. CALKINS: I would like to make a statement here in trying to sum up actually what was said here as far as IBM commitments were concerned.

I will make a statement as it is and let it be open for correction.

It is my understanding that IBM -- that IBM has committed themselves to arrive or to achieve a FORTRAN for the 705, Model

3.

The statement has been made and I understand this to be IBM's commitment, that they will have a FORTRAN for the 705, Model 1 and 2, and that they would put the number of people necessary as a minimum to produce this FORTRAN.

Those numbers of people would actually be independent to get this job done from the number of people supplied by industry.

In other words, IBM would supply the necessary personnel to do this job, whether or not industry had brought the question to them or not, is that correct?

MR. BEMER: No, this is not correct.

MR. CALKINS: That is not?

MR. BEMER: No.

MR. CALKINS: That is what I understood.

MR. SELDEN: I think you started off with Model 2 and you meant Model 3?

MR. CALKINS: I understand that IBM is doing it for Model 3?

MR. BEMER: IBM would not have been able to do it for Model 2.

MR. CALKINS: But that IBM will supply the men to do the necessary job for Model 2 and 1?

MR. KUSS: You will supply the same number for Model 3, and if industry helps, we will get it for Models 1 and 2 also?

Our participation in the 1 and 2 process has no reflection -- you're not going to have people on Model 3 because of 1 and 2?

You will apply the same number for Model 3?

MR. BEMER: That is right. Let me have a whack at it in stating this thing.

(Laughter.)

MR. CALKINS: It is kind of tough, Bob.

(Laughter.)

MR. BEMER: FORTRAN's Model 1 and 2, and Model 3:

IBM has to build Model 3 anyway, requiring independent people which we have not determined yet.

We have a great interest in seeing Model 1 and 2 FORTRAN produced for industry, and we will undoubtedly aid that project as well as the Model 3 that we do ourselves.

If doing it ourselves, that doesn't mean separate people. It may mean people phased, but in any event, if you take the FORTRAN projects, the two of these, one for the Model 1 and 2 and one for the Model 3, and lump them together, as an overall project, IBM will not put any less people on them for the simple reason that industry is aiding us.

MR. CALKINS: In effect, you are stating that you will have the necessary personnel to accomplish this as a lump project were industry not to supply anything to you?

MR. SELDEN: No, no, no. We accomplish only the Model 3 by industry participation. There will be two programs, and IBM's commitment will be to work for the joint effort as much as it would be working for the single effort by itself.

MR. BEMER: At least.

MR. SELDEN: At least.

MR. CALKINS: At least? That satisfies us.

MR. SHEPPARD: Pardon me. Except from one standpoint, and that is one of timing.

If these two projects are run concurrently, does that mean that the FORTRAN for Model 1 and 2 would not be available prior to Model 3?

MR. BEMER: No. We desire to have this FORTRAN of Model 1 and 2 done in a nine to twelve-month period from today.

MR. SHEPPARD: This is what I was hoping to get at, a target date.

MR. BEMER: This, I think, I have said before, and inasmuch as much of the work done on the Model 1 and 2 FORTRAN is applicable to the Model 3, it is only fair -- over the total disposition of personnel and effort, as we have stated -- that some of the people that would have otherwise to do the Model 3 are available to help on Model 2, so that the total effort by IBM is not diminished.

A PARTICIPANT: They have to step up the time schedule, is that right?

MR. BEMER: Probably a bit, not a great deal.

MR. AHLIN: Gentlemen, I think I would like to make a general comment here.

If a completely accurate recording of our conversations were taken here such as the people are doing in the other room, and if I read it at a later date, I would feel that IBM's commitment was one devised to leave many loopholes, and I am sure you have the same feeling.

(Laughter.)

MR. CALKINS: That is what we are trying to plug.

(Laughter.)

MR. AHLIN: However, I am sure that if I attended some of your companies' planning meetings, when you discussed programming, and I'd probably hear from you similar statements.

In all honesty, what we are trying to do here is not trying to be clever or evasive or leave loopholes. We are trying to be reasonable and, as I say, it would not be in any of our best interests if we made statements that we cannot live up to.

We have problems in an enormous programming effort, just as you have problems in a programming effort.

We are trying to be reasonable, we are trying to get these two jobs accomplished.

We stated clearly from time to time that we would not have done the job had it not been for your offer of assistance and your very determined interest in it.

We are certainly very influenced by the fact that you people are as interested in this thing as you are.

We have not, for example, gotten the same degree of, or expression of interest in our past for Autocoder. We don't expect it, we don't expect to get that interest. I frankly wish we would.

I would like to have the same type of assistance and interest in the Autocoder which, I think, is more fundamental to the 705 system than anything else.

That is something that has universal application in your work and any other work that is done.

As I say, this expression of interest and desire on your part certainly moves us as it should, certainly, and all I am really saying is that we are not trying to leave our statements for commitments full of loopholes, which we can later squeeze out of.

We are trying to be reasonable here.

Our basic objective is to make the thing available on time. We don't know what time is involved, how many people are needed, and this is something that we will have to determine.

If we are not successful, in the last analysis, we will show and recognize that our whole programming program is based on that it is going to be a family service, and if not, it is not going to be an acceptable one.

12 Many of your own service objects in your company have the same problem and your objectives must be the same as ours, and that is to be honest, to be reasonable, and I think that we get together, work together, and we can achieve our objectives.

I guess I will leave it there. Does that sound fair enough?

CHAIRMAN THOMPSON: Very good. Thanks. As far as I am concerned, I am willing to trust IBM to come through and I believe that the subject of commitments is fairly well defined.

It is fairly well defined except for people from industry that haven't had an opportunity to voice their interest and intent to participate.

MR. KUSS: Could we have something from Bob Bemer as to what he thinks the minimum machinery is going to be to assemble the FORTRAN? Somebody mentioned a drum before.

MR. SELDEN: That's why I want to do the two sections in series. No drum. I expect, if we use the Autocoder, it will be reasonable to assume to use the same machine the Autocoder requires which is 6-7 if you have a high-speed printer.

CHAIRMAN THOMPSON: 20K?

MR. SELDEN: 20K.

MR. AHLIN: Muddle 1 and 2.

CHAIRMAN THOMPSON: Muddle 1 and 2? No TRC?

MR. SELDEN: No TRC.

MR. BEMER: Well, TRC, if we want to get the thing done, it will run or bypass TRC's and we can do this, I think, within the time schedule.

It will be reasonable not to commit ourselves to TRC usage, otherwise you are unnecessarily complicating the problem.

CHAIRMAN THOMPSON: So we are switching them for a 705-1?

MR. BEMER: 705-1. What about the Model 2? Because, really, the memory, except for sizes, is not too different -- is not too difficult to take care of in many instances, except in the flow of the problems through the machine.

Do you think so, Bill?

MR. SELDEN: You are referring to the process or the object program?

MR. BEMER: Well, I am referring to both.

MR. SELDEN: Well, they are two separate questions. FORTRAN for 704 will run on a certain machine, but will allow you to make a program that will run on a machine larger than that.

MR. BEMER: Yes.

MR. SELDEN: The Autocoder does that. When I say "Autocoder," we will assume that this will happen. We will presumably first utilize the TRC as opposed to by-passing it in the case of the object program, and last do it in the processor.

We might do it in utilizing the TRC as opposed to merely by-passing it, and part of the processors before we do it in all processors.

MR. BEMER: Here, the possibility is of using in descending order, the possibility of 40K in the object program, the possibility of the TRC in the object program; the possibility of 40K in the processor, the posibility of the TRC in the processor.

I am thoroughly against the latter, but the other three are under consideration; but at least we could say this:

Until the committee gets going that we will work on the minimum machine which is the Model 1, 20K, without the TRC, and we will in all probability be able to produce programs for 40K, although we can settle it later.

At least, we will establish the lower bounds.

CHAIRMAN THOMPSON: I sure buy that.

MR. KUSS: One more question: Have you got any idea of how many hours for instruction the assembly pass will take?

MR. BEMER: You mean the output?

MR. KUSS: The whole FORTRAN.

MR. SELDEN: Produced instruction?

MR. KUSS: Produced instruction. It is more like what we have now.

MR. SELDEN: I will take a flying, and completely flying guess. It is something around a range of 100 to 250 instructions per minute.

CHAIRMAN THOMPSON: FORTRAN statements?

MR. SELDEN: Object program statements. This may, if we assume the 50 to 60 to 1 expansion that you get in FORTRAN, this is about 10 or 20 instructions a minute input.

Format FORTRAN statements, that is.

MR. BEMER: If that seems sort of low at this time, and will probably be low, however I think the coding facility will be so much greater, it wouldn't bother you any in de-bugging it.

MR. SELDEN: We hope that it will be possible to do a certain amount of partial rather than complete assemblage. We hope that, when I mention the improved Autocoder, one facility that could be improved is to put much of the routine, of the routines that are used -- the floating routine would be a constant relocated or constantly relocated by the Autocoder.

Or, in this case, not even relocated, be copied out so your first 500 or a thousand instructions produced would cost you 10 seconds, and then you start counting at the rate that I have mentioned after this.

CHAIRMAN THOMPSON: Are there any other major points, Bob, that you have?

MR. BEMER: Just one minor one that I brought up for consideration first.

That is the question of publication, manuals and credits and stuff like that.

I don't think that we need to say anything here on that. We can settle all that in the committee meeting, except that I think we ought to keep this in mind, that when we get this thing going that there will be something people will be watching in business because this will be the first co-operative effort that I know of in this magnitude by the commercial people on the 705.

I think we want to document this thing pretty well for articles. You may have noticed the articles on the PACT effort in the SEACM Journal.

I don't think we want to be blind to the possibility of showing the good of the good work when it gets done, but when we are pretty close, because we want to be ready to do something about it.

CHAIRMAN THOMPSON: Along this line, we have under consideration publishing all GUIDE correspondence and distributing to each installation on a monthly basis in initial stages.

This could possibly be covered by that Secretary to Membership letter, and then when things start coming to a head, certain of these could be gathered together and published in a partially complete report or something in that line.

MR. BEMER: Good.

CHAIRMAN THOMPSON: Are there any other questions?

MR. SHEPPARD: I would like to make a statement.

Maybe this should be in the form of a motion to this committee, that we accept the proposals that have been made here as a working basis, providing that this committee gives a mandate to the committee which will meet in New York to provide certain minimum results.

One, is a complete statement of the problem and the specifications for an F-5 FORTRAN.

Secondly, that manpower requirements are pretty closely determined at that time; and

Thirdly, that a definite time schedule is set up to accomplish this.

CHAIRMAN THOMPSON: I think these three points could be taken care of at the October meeting, Bob?

MR. BEMER: I do, under one circumstance.

We have something here that demands, like Jack says, the utmost co-operation.

We must have very good programmers. These people up here, except for one, are numbers to me, and if one installation does not provide a good programmer -- one that can really carry the ball, and so -- and he will drag the rest of us down.

IBM is in no position to crack a whip and say, "We will fire you," because you work for somebody else.

So, that's the only thing that does worry me about talking of time schedules and manpower determinations.

Although we can help all we want, we can't force you to do anything on this thing, and I don't want to.

MR. SELDEN: I think the schedule definitely has to be adopted by the committee rather than IBM. We cannot adopt a schedule.

CHAIRMAN THOMPSON: Well, I for one, having quoted some problems like most of you, abhor trying to state when I'm going to finish the program.

MR. SHEPPARD: We have found in our operation that programs get completed pretty much on schedule. We can set a two-year date and it will be finished in two years.

If you say one year or six months, it is not going to be finished.

We've got to have a target. If we don't, we will miss the boat completely.

13 CHAIRMAN THOMPSON: That target has been set by Bob.

MR. BEMER: All he wants is a target. I agree with you, John.

MR. SHEPPARD: Somehow it works.

CHAIRMAN THOMPSON: You can put a time schedule on it. If this serves to speed things up, I am for it.

MR. SHEPPARD: I realize that the engineering or scientific type of mind doesn't like schedules in any way, shape or form.

If you were an old production man like myself, you would recognize the need for schedules considerably more.

MR. CHAMBERS: I see Jack sitting here with his hands holding his face, and my main object here is to determine the prerequisite for the person that we are willing to contribute towards this effort.

We certainly do not want to put somebody on this effort who is going to hold it back. We don't want him to feel that, well, he is here and he can't do a proportion of work, at least a share along these lines.

I hope to leave here with a good idea or knowledge so that I can commit this person and feel sure that he is going to do a good job.

I wonder if Bill, Bob or Jack can steer us along these lines; perhaps send us literature?

CHAIRMAN THOMPSON: Why not make the statement right now?

MR. SELDEN: I think I will make statements, we all make them. What do you think, Bob?

MR. BEMER: I can illustrate this with a PACT project. The people that actually did the work, Charlie Baker, etcetera, were actually top-notch programmers in their companies that they could get out.

They were the best they could find and they worked.

I think if you settle for anything less than the very best, you can afford, it will not work.

MR. CHAMBERS: Well,"the very best"-- sometimes, this is a little general. Perhaps there is a previous type of programming that has a lot to do to determine what you mean by "the very best."

MR. BEMER: I don't think that has to do too much in this case.

MR. BERNINGER: In other words, a form of background?

MR. BEMER: Let's put it this way:

They don't have to have a college degree. This, I think, is not at all necessary. I think that you demand quite ingenious people who still will be able to restrict themselves in their ingenuity so that they won't make coding that nobody else can understand.

(Laughter.)

MR. CHAMBERS: Sometimes you get a person who may come up with a method in doing things and I am sure we have an individual who can do this, but it is a question of whether we want to take them away from the application of machine to engineering as far as our company is concerned.

MR. BEMER: Oh, well, now, let's make this very clear right now.

We are working on the FORTRAN language and processor to convert it to machine instructions.

This is not engineering. This is programming that is applicable to any type of work.

This is programming as such. Because we are producing a system that will be used by engineering people for the most part, does not mean that our programming itself will be engineering or related to engineering problems.

MR. SELDEN: I see no reason to favor a mathematician or engineer in the choice.

I think one or two people in the group would have to have some background. Among the people that IBM has already submitted or committed will be enough on the mathematical background needed.

Everyone else doesn't need it.

MR. CHAMBERS: Well, this is --

MR. SELDEN (Interposing): Well, somebody who can transfer any routine, something like this.

MR. CHAMBERS: I am reading the FORTRAN Manual as to what goes into the input, output machine and engineering application. We are not too concerned with this problem.

Probably our commercial people are more versed along these lines, at this time, the fact that they have these problems.

MR. AHLIN: What are your commercial people there?

MR. SHEPPARD: Well, you might say this:

I think that right now I don't think we are in a position where we have to determine the exact person in an organization that we are going to commit to this as long as we make a commitment.

I think that the requirements for the type of person that we want will come out of the October meeting in New York; and, at that time, we can determine who the specific person is within the organization.

MR. SELDEN: I think we should have the people there to start working, whoever walks into the door.

MR. BEMER: I was asked about some people who had been successful in this thing. They were sought off statistics and shop scheduling people, working on that.

However, in the FORTRAN business, we have several chess players, guitarists, and ex-advertising agency general factotum -- about anything you can imagine.

All they need is a little spark of ingenuity that can visualize many things walking around at the same time.

In other words, a sensible fellow with horse sense, a guy who doesn't mind doing some hard work, and has a reasonable amount of ingenuity and adaptability.

CHAIRMAN THOMPSON: A man who doesn't require much sleep?

MR. BEMER: That is a good point.

(Laughter.)

CHAIRMAN THOMPSON: Jack, could you add anything to these people?

MR. AHLIN: Let me just summarize just what has been said.

Correction: First of all, the person does not have to have a particular type of academic background.

I think this is completely aside from the choice of a man. He should undoubtedly have had some length of service in programming.

He should, in addition, have accomplished something -- and I would say he should have completed some successful programming.

At least, two things kind of go together, length of service and having done something.

(Laughter.)

Preferrably, we would like to have someone who has a very decided interest in programming, who'd rather do coding than anything else.

MR. BEMER: Anything else except one thing, please, Jack.

(Laughter.)

MR. AHLIN: Somebody who has written maintenance routine, transit routine, has written mathematical subroutine, or somebody like George here (indicating), somebody with that type of background, I think, is an outstanding candidate.

He should have ingenuity and foresight. I think we can impress on any person the need for adhering to certain programming conventions and standards which we ourselves have adopted.

Bob~~by here~~ BEMER is no longer the programmer -- at least, the type he was before -- and found out through his own realization that it doesn't pay to be cute in this business.

(Laughter.)

There are a lot of people who read what you do and this would be particularly true in something like this, and we just can't afford the luxury in having people determine as to what they did in being cute.

I think these points are very important.

CHAIRMAN THOMPSON: Does anybody else have anything?

MR. SELDEN: Qualifications or anything else?

MR. CARL L. BYHAM (Southern Railway System): In connection with this first meeting, is this to be such a meeting that it would concern only working people?

MR. BEMER: No.

CHAIRMAN THOMPSON: When John was talking about this --

MR. BYHAM: (Interposing) Or perhaps one of us would be there, too?

MR. BEMER: Yes. I think it would be very well if both sides,at least one or two.

CHAIRMAN THOMPSON: It occurred to me when John was talking that if you have three or four people, send all of them -- if you can afford it.

MR. SHEPPARD: I was thinking of the possibility of a team, possibly, an engineer and a data processing man maybe working as a team in our organization.

CHAIRMAN THOMPSON: I don't mean to commit all of them to the job, but then between themselves they can probably decide who's got the greatest appetite for it.

MR. SHEPPARD: We have an austerity program, too, you know?

(Laughter.)

CHAIRMAN THOMPSON: Well, it is adjournment time unless there's something else.

MR. MAURICE T. DEVLEN (Canadian Pacific Railway): I was wondering now, in this letter you are going to send out to the installations, whether you could have some statement of qualifications.

I think it would be very helpful.

CHAIRMAN THOMPSON: All right.

MR. SELDEN: Should be included that it is necessary for the working party to be there. That would be nice to have.

MR. SHEPPARD: There is a motion before the floor. There has been no action on my motion.

I didn't know whether it was passed approval or whether it was just plain ignoring it.

(Laughter.)

CHAIRMAN THOMPSON: I believe we touched on the points, but we didn't formalize the thing.

The three points were to review that this working group come up with a complete statement as to the minimum machine, the language and for --

MR. SHEPPARD: (Interposing) The minimum specifications.

CHAIRMAN THOMPSON: The minimum specifications; the manpower question clearly stated, and, if it is possible at that time, a time schedule.

Those in favor of setting forth these targets for this working committee, if this is the proper way to present this --

1 Any objections?

(No response.)

CHAIRMAN THOMPSON: Motion carried.

(Whereupon, at this time, the motion indicated above was adopted by the subcommittee.)

CHAIRMAN THOMPSON: Well, if there is no more comment, this will then conclude the meeting.

Thank you, Gentlemen.

(WHEREUPON, at 12:20 o'clock, p.m., the meeting adjourned sine die.)

- - -

SHARE V 56 MAY 9-11

(7) FORTRAN - John Backus of IBM World Headquarters reported on FORTRAN, characterizing the status of the FORTRAN system by noting the following:

a. FORTRAN will be ready for customer use by early August, 1956, at which time the complete system will be distributed, on magnetic tape, to all customers.

b. The system will have taken fourteen man-years to write and check out.

c. The program will be comprised of approximately 19,800 instructions. (These will all be "active" instructions, none going into the program produced by FORTRAN.) There are, as of May, 1956, 16,400 instructions written, of which 8,400 have been tested. There are currently twelve people working on the project.

d. A programmer's manual of FORTRAN will be completed approximately June 1, 1956.

e. A complete description of "internal" FORTRAN will be available in early September, 1956.

f. The "minimum machine" required by the FORTRAN system will be:

   1. 4096 CORE (There are no modifications presently under way to enable FORTRAN to take advantage of larger amounts of core storage.)
   2. Four (4) tapes
   3. Four (4) logical drums

g. It is anticipated that, for every FORTRAN instruction written, from five to twenty instructions will be generated by the system (1/5 - 1/20 compiling ratio). This is expected to result in a reduction of coding time by a factor of from three to thirty.

h. FORTRAN will have substantially the same specifications as those originally published.

Reports from Members

(1) PACT 1A - Owen Mock (NA) described the progress to date on PACT 1A, a generative compiler for the 704 being written through the cooperative effort of several member installations of SHARE. Mock noted the following:

a. PACT 1A programming will be completed sometime in June, 1956, at which time the system will be distributed to a limited number of installations for trial. When indications are given from these field trials that the bugs are out of PACT 1A, the system will be available to all 704 installations on request.

b. The PACT 1A manual will be completed sometime in June, 1956.

c. PACT 1A will have required six man-years to write and check out.

d. The system will be comprised of approximately 10,000 active instructions.

e. The "minimum machine" required for use of PACT 1A will be a 704 having:

   1. Three (3) tapes
   2. 8,192 words of other storage. (Either all core or a combination of core and drum.)

Agenda Item 12 - NUMBER OF CHARACTERS USED IN A SYMBOL

There was a brief introductory discussion in which it was made clear that the following is a continuation of informal discussions at the Second SHARE Meeting, and, in effect, a follow-up to SHARE mail proposal No. 1 (by General Electric - ballot mailed 4 October 1955).

It was moved and seconded that:

> "Programs distributed through SHARE shall use five or fewer characters as symbols, except when the programmer wishes to prevent heading (e.g., the symbol "COMMON")."

A spirited discussion followed. The stated objectives of the motion were to enable any distributed routine to be re-assembled with other routines into a final program. Shell (GE) explained various reasons why he doesn't like it, claiming that it doesn't meet the objectives which it pretends to. Heising (NY) asked what would happen when you tried to distribute this final, re-assembled program (with six character symbols), and somebody wanted to re-assemble it again with other routines. Ramshaw (UA) pointed out that it should not be done that way, serially, but rather in parallel; that is to say, by getting together all the individual routines in their original form and then re-assembling all at once.

Shell (GE) noted that he is preparing a program to translate CAGE symbolic cards into SHARE format. When this is completed, any subroutine he distributes will have only one symbol.

It was universally agreed that this was the most useful form which distributed subroutines could take.

The motion was passed.

Agenda Item 13 - COMPILERS

FORTRAN

Backus (NY) made a progress report on FORTRAN. Six thousand instructions have been coded in what they hope is final form. He expects that in its first edition FORTRAN will include eight to ten thousand instructions, which will be coded by January 1st. Some debugging will have been accomplished by then, and he estimates that it will be completely checked out some time in February. The minimum components necessary will be one 4096-word core, four tapes, one drum box, and either on-line or off-line output. It will produce symbolic instructions for subsequent assembly in the SHARE format. It is estimated that it will take six minutes to produce one thousand symbolic instructions. The symbols used will be the same ones that were used for variables. Planned for the second edition is the inclusion of formula numbers in the comments. He gave a brief rundown of changes from the latest printed specifications. These are summarized below in Part III.C. He also covered very hurriedly the techniques they are using, the most dramatic of which was the enormous number of tables set up. He ended by paying tribute to United Aircraft, and especially Roy Nutt, for their cooperation and assistance.

# FORTRAN

W. P. HEISING, IBM Corporation, New York, N. Y.

"The FORTRAN language is intended to be capable of expressing any problem of numerical computation. In particular, it deals easily with problems containing large sets of formulae and many variables, and it permits any variable to have up to three independent subscripts. However, for problems in which machine words have a logical rather than a numerical meaning it is less satisfactory, and it may fail entirely to express some such problems. Nevertheless, many logical operations not directly expressable in the FORTRAN language can be obtained by making use of provisions for incorporating library routines." This quotation is taken from "The FORTRAN Automatic Coding System for the IBM 704 EDPM," dated October 15, 1956. This first manual was a programmer's reference manual issued by the Programming Research Department of IBM. The original system and the original manual were the work of J. W. Backus, R. J. Beeber, S. Best, R. Goldberg, H. L. Herrick, R. A. Hughes (U.C.R.L., Livermore), L. B. Mitchell, R. A. Nelson, R. Nutt (United Aircraft), D. Sayre, P. B. Sheridan, H. Stern, and I. Ziller; all were associated with IBM except as noted.

Since that time, FORTRAN systems have been prepared for the IBM 650, 1401, 1410, 1620, 705/7080, 7030, 7070/7072/7074, and 709/7090/7094 systems as well as for equipment of many other manufacturers.

Although there have been many changes and additions over the years in the FORTRAN language and associated manuals, the basic structure and intent have been extended rather than altered in any fundamental way. Accordingly, the quotation taken from the original manual on the scope and intent of FORTRAN is as accurate today as when it was first written.

The completion of the original 704 FORTRAN system constitutes a significant achievement in the history of programming and has undoubtedly influenced later developments. Such now common terms as "source program," "object program," and "object machine" have passed into the general programming vocabulary from the original FORTRAN manual.

The 704 FORTRAN system was issued early in 1957 by the Programming Research Department of IBM and included as a part of it, a system editing program to introduce modifications. The master tape as issued was not directly used for compilation, but rather served as the principal input to the system edit program. The system edit program accepted as a secondary input "modification cards" which constituted a cumulative list of alterations to be made from the master tape. The output was a "working system tape" which would be used for actual system execution. This method was chosen so that small changes could be readily introduced to a large number of using installations by mailing a few punched cards rather than requiring the slow process of mailing back and forth magnetic tape reels. A system of numbering modifications is now in use so that each modification is numbered consecutively and the "modification level" of a user's system is indicated by, say, "level 76" indicating that numbered modifications through modification number 76 have been incorporated. Modifications are used to remove system errors or for adding improvements to performance in the way of speed of compiling, or execution, or new system or language facilities.

The system editing program also permits a using installation to add nonstandard modifications to the working system to meet local needs without modifying the master tape itself. The local systems programmer must be careful that numbered modifications do not conflict with any nonstandard modifications added locally.

The master tape is in absolute code, and in time either the accumulation of modifications becomes too bulky or the nature of a modification is such as to make desirable a new master tape. In such cases, the entire system is reassembled to make a new master tape, which is called a version with its own number, and a new set of numbered modifications starting from one is instituted. The older version is commonly superseded and is no longer maintained. Separate versions may also be issued for different system configurations of the same machine type, quite commonly for differing amounts of core storage. Accordingly, more than one version can be in current maintenance status at any given time.

The original concept of FORTRAN was developed by John Backus in 1954 and the original FORTRAN language was basically completed by mid-1955. At that time the instruction repertoire of the 704 was frozen and the main coding effort on the compiler began. The size of the effort and the time required exceeded initial estimates—a not uncommon situation when one is working in an entirely new area. The initial system released in 1957 was probably the most complex programming system ever produced up to that time, and the fact that the system comprised some 25,000 lines of code is not an accurate measure of the complexity, as the analysis particularly of index register assignment was especially intricate. During its initial period of use in 1957, many small errors came to light, and it was a difficult period for users and the authors alike until the most frequently encountered errors were diagnosed and corrected. The improvement effort responsibility was transferred from Programming Research to the Programming Systems Department (then called Applied Programming) early in 1958.

FORTRAN II, a new version with significant source language additions was officially released by Programming Systems in June, 1958, although the basic planning and much of the work had been done by the original authors.

FORTRAN systems for the 709 and the 650 were officially released later in 1958. During this same period, GUIDE was writing a FORTRAN section of the 705 Autocoder III system, and 705 FORTRAN was eventually tested and released with subsequent maintenance by IBM. 1620 FORTRAN was released in the fall of 1960, 7070 FORTRAN in 1960, and 7030 (Stretch) FORTRAN IV was shipped to customers in the summer of 1962.

A FORTRAN primer was published in 1957, primarily to introduce FORTRAN to scientists and engineers who were not computer specialists. It was specifically written to introduce the reader step-by-step to each part of FORTRAN and was extremely popular. It was succeeded by a FORTRAN General Information Manual which serves the same functions for a variety of machine types, and has separate chapters at the back giving additional information specific to each machine type.

My data are incomplete, especially for the earlier years. However, the figures in Table I for some of the "best-sellers" give some idea of the scope of the publishing activity.

These are only a few of the scores of manuals produced and excludes publications in German, French and other foreign

TABLE I

| *Manual* | *Period Covered* | *Number Distributed* |
|---|---|---|
| 704 FORTRAN Reference Manual | 1/58(?) to 3/61 | 61,932 |
| 704 FORTRAN II Reference Manual | 6/58 to 11/62 | 37,058 |
| 709 FORTRAN Reference Manual | 9/59 to 11/62 | 69,386 |
| FORTRAN General Information Manual | 10/61 to 8/62 | 61,081 |

languages. FORTRAN manuals in foreign languages are produced abroad. The most extensive FORTRAN foreign language documentation is probably the French material on the 7090 FORTRAN Monitor System. The IBM Computing Center in Paris not only produced manuals in French, but has modified the system so that all of the hundreds of compiler and execution diagnostics are produced in French as well as symbolic listings using French operation code mnemonics.

Contributions from FORTRAN users have unquestionably been the primary factor in pointing the way from FORTRAN in 1957 to the systems of today. The systems of today reflect the accumulated experience and suggestions of literally thousands of users, and compiler authors must make real efforts not to lapse into ivory towers, if their product is to make the transition from a toy to a workhorse.

The nature of user-author communication is enormously varied. At one end of the spectrum are the maintenance procedures. In these a user who believes he has detected a system error forwards a completed standard form together with a source program card deck and any other appropriate material. Upon receipt, the inquiry is logged, receipt is acknowledged, and the source deck is run to verify the reported behavior (this is to eliminate machine malfunction or nonstandard system modification as a possible cause). The results of the test are then analyzed and will normally be classified as (1) source program error, (2) known system error, or (3) previously unreported system error. In the case of source program error, the error will be pointed out to the sender. However, in some cases, additional action may follow. The reference manual may be incorrect, incomplete or possibly misleading, in which case later editions of the manual will be modified. Furthermore, a source program error which occurs frequently may indicate the desirability of an additional diagnostic for the compiler. A known error may have already been corrected by a later modification than the user has received (he reported his modification level). If the correction of an error requires extensive system changes that will result in considerable delay, it may be necessary to notify all users of a temporary restriction until the appropriate modifications can be programmed and tested.

The remoteness geographically between the support authority and the hundreds or even thousands of users makes the thoroughness of testing of modifications crucial, as the inadvertent introduction of other errors may inconvenience many people. It is definitely not sufficient to merely check that the program which brought the error to light is correctly handled—rather a whole battery of "typical" problems are recompiled and retested. In fact, an excellent source of "typical" programs are those user programs originally submitted through this procedure, since such programs tend to be sufficiently complex to exercise little used parts of the programming system.

Another aspect of the author-user communication is correspondence and personal contact. Especially influential has been SHARE, the 704/9/90/94 users group which has had a FORTRAN Committee since June, 1958, and hence has influenced FORTRAN over the longest period. Many significant additions were the work of the SHARE FORTRAN Committee members and many other additions were in response to needs which become apparent in SHARE FORTRAN Committee discussions. The FORTRAN Assembly Program (FAP) was written by U.C.L.A., Macro-FAP by Bell Laboratories, the symbolic debugging system by a committee of four using installations, and the double precision/complex package by Hughes Aircraft.

The most significant single addition, however, was the incorporation of the automatic monitor system of the Rocketdyne Division of North American Aviation. Indeed, every single aspect of the SHARE-IBM relationship had the same motif—users want a programming system, not merely a compiler. Although a compiler may be the largest single component of a programming system, it has probably received more than its proper share of attention in the literature relative to system components which perform more mundane but equally vital functions.

# DOCUMENTATION OF IPL-V

ALLEN NEWELL, Carnegie Institute of Technology, Pittsburgh, Penn.

IPL-V (Information Processing Language-V) is a programming language for list processing and symbol manipulation. It is the fifth of a series of programming languages that has developed as part of a research effort in artificial intelligence and simulation of cognitive processes. This research started in late 1954 at The RAND Corporation and Carnegie Institute of Technology and has remained centered there, so that these two organizations, or more properly the scientists thereof, can be considered the source of the language. The earlier IPL's were coded for the RAND JOHNNIAC, a unique machine of the Princeton class; IPL-V is the only one which has become a "public" language and where the necessary effort has been made to document and standardize the language.

IPL-V started out in late 1957 to be a "modified copy" of IPL-IV (then being implemented on JOHNNIAC) for the IBM 650, the computer available at Carnegie Tech. A running system was produced in early 1958.[1] It was then felt that a system should exist that would be usable both on the IBM 650 and on the IBM 704, which was also in use at RAND. This precipitated another iteration culminating in a preliminary version of the manual, which doubled as the specifications, in June 1958. The system became operational first on the 704 at the end of summer, 1959.

As shown below, IPL-V now exists for several different machines. In each case the impetus has come from some person who has wanted to use the language. As a result the maintenance of each machine version rests with the originating group for that

[1] HENSLEY, C. B., NEWELL, A., AND TONGE, F. M. 650 Information Processing Language. CIP Working Paper #9 (ditto), Jan. 1958. (No longer available.)

# SECTION 01

# INTRODUCTION

## 01.01 The Origin of the SHARE Organization

Upon getting thoroughly into the problem of preparations for the IBM 704, three installations in the Los Angeles area began to have informal discussions concerning their individual plans. Having been pleasantly surprised by the successful cooperative effort for the design and coding of Pact I, a favorable climate existed for a similar joint activity in connection with program development for the 704. Accordingly, Rand, Lockheed, and North American seriously began to consider standardization. A fortunate circumstance was the seminar held by IBM in Los Angeles during the week of August 8, 1955. This brought representatives of several other western installations together, and the idea was discussed among them.

The mutual respect that the participants in these discussions had for the programming competence of the others soon brought the realization that an "isolationist" attitude no longer existed, and almost all professed themselves as quite willing to accept the ideas of others, even to the extent of obsoleting things already done within their own installations. It was unanimously agreed that a full-scale attempt should be made to bring SHARE into being, Since it seemed almost too late to do it on a nationwide basis, extreme haste was necessary and the initial meeting of SHARE was called for the week of August 22, 1955.

In spite of such short notice, almost all potential 704 installations throughout the country responded with alacrity. All expressed a desire to participate, and attendance at the first meeting was gratifyingly large. Seventeen installations - the charter members of the organization - were represented:

| | |
|---|---|
| BA | Boeing Airplane Company |
| CL | Lockheed Aircraft Corporation, Burbank |
| CR | California Research Corporation |
| CW | Curtiss-Wright Corporation |
| GE | General Electric Company, Cincinnati |
| GL | Lockheed Aircraft Corporation, Marietta |
| GM | General Motors Research |
| HA | Hughes Aircraft Company |

| | |
|---|---|
| PK | International Business Machines Corporation, Poughkeepsie |
| NY | International Business Machines Corporation, New York |
| ML | Lockheed Aircraft Corporation, Van Nuys |
| NA | North American Aviation, Inc., Los Angeles |
| NS | National Security Agency |
| RS | The Rand Corporation, Santa Monica |
| UA | United Aircraft Corporation |
| LA | University of California Los Alamos Scientific Laboratory |
| LC | University of California Radiation Laboratory, Livermore |

One other installation, Douglas Aircraft Company - El Segundo Division, was also present at the first meeting. At that time, its 704 procurement picture was rather vague, but it subsequently became a member.

The name of the organization was selected with the naive hope that suitable words could be found which would match the initials, describe the aims of the organization, and, at the same time, be clever enough so that somebody would admit to originating them. Although many suggestions approximating this were propounded, nobody was really that smart, and so each member is free to interpret the initials in his own way. (It has been suggested that this is symbolic of one of SHARE's principles of "unity in essentials and freedom in accidentals!")

## 01.02 Advantages of SHARE Membership

A member of SHARE is closely united with one of the main streams of development of computer usage in the world. A substantial percentage of the major users of high-speed digital computing equipment is represented in the SHARE membership. The knowledge of what is going on in these installations is considered by many of those members to be an indispensable requirement for efficiently exploiting the 704 and the 709. In addition to the documentary information which is received through the mail, the informal conversations at the meetings are very productive. These discussions, involving some of the most brilliant 704 and 709 programmers in the world, are frequently extremely fruitful. It has been found that critical evaluation of one another's ideas by these men usually produces a distillation of their thoughts which is superior to any individual opinions.

Member installations should be able to do considerably less programming and checkout of utility routines, mathematical routines and complete systems. Almost all the utility type routines produced by its members are expected to be distributed through SHARE. Members have had the opportunity to have a voice in the specifications of these routines, and, because of the close contacts with other members, can keep as up-to-date as necessary on their progress. Even those members who have done a significant amount of utility programming have available to them a considerably more diversified library than they would otherwise. Moreover, the continual interchange of ideas among the members (representing most of the 704-709 programming talent in the world) has demonstrated that a much higher degree of computing sophistication is rapidly built up in an installation than would result if it maintained a splendid isolation.

SHARE has been able to provide IBM with well-thought-out and authoritative requests for changes to the 704, 709 and other associated equipment and believes that IBM will pay much more attention to such a united voice than to individual requests. However, SHARE members also distribute copies of their individual RPQ's through the organization. Officially, SHARE has decided not to extrapolate this activity to include the next generation of machines. However, the presence in one hotel of so much authoritative customer opinion (at SHARE meetings) is expected to be used by machine manufacturers. This is obviously an excellent spot to conduct sales research surveys with the expectation of obtaining highly meaningful information.

## SECTION 10.01

## APPENDIX

### 10.01 SYNOPSES OF PROCEEDINGS

1. FIRST MEETING August 22-26, 1955 Los Angeles, California

   a. Attendance - Eighteen installations that later become members and IBM Applied Science Division represented.

   b. SHARE Organization - The following officers were elected:

   Chairman: Jack Strong, North American, Los Angeles
   Vice-Chairman: Donald Shell, General Electric, Cincinnati
   Secretary: Fletcher Jones, North American, Los Angeles

   c. SHARE Standards Adopted

   1.) The SHARE operation code will be the IBM mnemonic code plus an extended operation list.

   2.) The IBM Assembly Program NYAPI will be modified to form the SHARE Assembler.

   3.) Binary card format.

   4.) Print wheel configuration as decided upon at the March 1955 701-704 symposium at the Rand Corporation.

   5.) Minimum 704 described. SHARE programs using component other than those of minimum 704 must so note in write-up.

   6.) Binary point location will be described by counting from left to right of a word.

   7.) Conventions in writing subroutines.

8.) Format of program writeup for SHARE distribution.

9.) Identification of program decks for SHARE distribution.

| | |
|---|---|
| Installation code | Columns 1-2 |
| Reserved for internal use | Columns 3-8 |
| Sequence number | Location specified in writeup |

10.) SHARE elementary function set will be comprised of the following routines:

Square root
Sine-cosine
Exponential
Logarithm (base e)
Arc tangent
Sinh-cosh
$X^a$

d. New Operations - It was agreed that SHARE members request of IBM these additional operations:

1) Copy and Add Logical word

2) Exclusive Or

3) Store Index in Address

4) Place Index in Address

5) Logical Right Shift

6) Store Tag

7) Backspace File

8) Read Tape Backward

e. Committees and Assignments

1) A sub-committee was designated to compile a glossary of terms to augment existing computing dictionaries.

2) Program assignments were made to members.

f. Papers Presented

| | |
|---|---|
| Cross Bar Switching | United Aircraft |
| Manual of operation for 704 using CAGE | General Electric, Evendale |
| 704 Regional Symbolic Assembly Program | Los Alamos |
| 704 Matrix Routine | Lockheed, Burbank |
| 704 Simulator on the 701 | Rand, Santa Monica |

2. SECOND MEETING September 12-13, 1955 Philadelphia, Pa.

a. Attendance - Sixteen member organizations and IBM Applied Science Division were represented. The number of members of SHARE was increased to twenty-one.

b. SHARE Standards Adopted

1) On the basis of new information, it was agreed to use the United Aircraft Assembly Program as the framework of the SHARE Assembler, instead of the IBM NYAPI as previously adopted.

2) The calling sequence form adopted at the first meeting should be a suggested form only and should not restrict the programmer.

3) Erasable storage symbol chosen -- COMMON

4) A SHARE program deck will not require that certain subroutines be available within the tape library for assembly.

5) Integer scaling will be specified as B=35, or a special data card form may be used.

6) Additions were made to the binary card form.

c. New Operations - A report from IBM was made concerning the additional operations requested at the first meeting.

1) Shortly to be added to list of standard operations

2-6) - Presently being engineered

7) Requires an RPQ from each installation

8) Impossible

IBM reported that half-word logic will be available at $500. 00 per month.

d. Committees

A report was submitted by the Glossary Sub-committee.

e. Papers including subroutines for elementary functions, data handling and diagnostics were presented by Lockheed, United Aircraft, North American, Los Alamos, Rand, California Research, IBM and General Electric.

3. THIRD MEETING November 10-11, 1955 Boston, Mass.

a. Attendance - Twenty-two members of SHARE and IBM Applied Science Division were represented.

b. SHARE Organization

1) Member is defined as an installation which has on hand or on order at least one 704. *

2) Quorum shall consist of at least two-thirds of members. Majority of quorum is necessary to pass any motion. ‡

3) Established percent of quorum is necessary to reconsider a previous decision and to overrule it.

c. SHARE Standards Adopted

1) Octal card form presented in the second proceedings will be the SHARE standard.

2) Material which is not in SHARE language will not be distributed by SHARE.

3) The standard library tape shall be Tape No. 1.

4) Procedures were established to transact SHARE business by mail.

* SHARE has since been expanded to include users of the 709 computer. See section entitled "The By-Laws of SHARE".

‡ Has since been changed; see "The By-Laws of SHARE".

5) All distributed decks which carry sequence numbers must use the following convention: Sequence numbers in self-loading decks shall start with zero; all others shall start with one.

6) Programs distributed through SHARE will use five or fewer characters as symbols, except when programmer wishes to prevent heading, as in erasable storage.

d. Request to IBM - SHARE requested IBM to use unused bits in the decrement field in a definite order.

e. Committees and Assignments

1) Committees were appointed to study the following:

a) Bibliography and index of SHARE distributed material.

b) Future 704 changes

c) Machine time charges

d) Periquip changes

e) Periquip reader wiring

f) Printer board standard

g) RPQ procedures

2) Various installations were charged with submitting mail proposals concerning:

a) Standing committee on mathematical analysis

b) New operations
Sense Copy Check
Index Register ADD instead of OR
Load Index with own address
Load Index with complement of own address.

c) Trapping mode console switch

f. Papers presented included descriptions of new routines, usage of peripheral equipment and compilers.

4. FOURTH MEETING February 6 and 10, 1956 San Francisco, California

a. Attendance - Twenty-seven member installations and IBM Applied Science Division were represented.

b. SHARE Organization

1) Don Shell resigned as vice-chairman.

2) Walter Ramshaw was elected new vice-chairman.

3) Standards were adopted for election of officers.

4) Non-members shall attend SHARE meetings by invitation only.

5) Statements were adopted describing the following:

a. Obligations and advantages of SHARE membership

b. Scope of SHARE activities for balance of 1956.

c. SHARE Standards Adopted

1) Card Form - binary, decimal, octal, chinese binary

| | | |
|---|---|---|
| Identification | Col | 73-80 |
| Information | Col | 1-72 |

2) On-line Board Wiring

| | |
|---|---|
| On-line Reader | 72-72 to accept adopted card form |
| On-line Punch | Col 2-9 offset gang punched identification columns |

3) Printer Board Wiring Diagrams

4) Tape record representing 80 column card should be 84 characters in length, the last 4 being blank.

5) Sense switch conventions

a. When a sense switch is used for control, the "down" position shall be the "unusual" case.

b. Sense switch No. 6 shall be used for trapping mode control.

6) A method for the exchange of statistical information concerning machine time charges was adopted.

d. Reports from IBM

1) It was restated that bits in decrement field of Type B instructions should not be used. These will be used by IBM in future machine changes and there is no order of probability of use.

2) Key punch code plates will be available which will print SHARE characters.

3) Notification of 704 changes will be distributed to SHARE in addition to appearing in 704 Information Bulletin.

e. RPQ Procedures

Certain legal ramifications render undesirable any joint action by SHARE in requesting machine changes. Future RPQ's originating in SHARE shall be submitted by each member, noting that the request is sponsored by SHARE.

It was urged that a member submit for SHARE distribution information on any RPQ submitted to IBM, if this is not in violation of proprietary or security standards.

Members were requested to submit an RPQ concerning a change to the automatic carriage control on the 717 printer.

An RPQ agreed upon by mail ballot concerning changes to the peripheral equipment was described.

f. Committees - The following committees were formed:

1) Education of Computer Personnel

2) Mathematical Methods

g. Papers presented

1) Programs written by Lockheed

2) SHARE Assembler Listing

5. FIFTH MEETING May 9-11, 1956 Chicago, Illinois

a. Attendance - Thirty-seven member installations were represented.

b. Format of Meeting

1) First Day - Panels on 704 experience, debugging, machine layout and CRT usage.

2) Second Day - 3 schedules of sub-committee meetings

3) Third Day - Reports from IBM concerning time clock, sequencing device on on-line punch, Chinese Binary, black box, 32 K word core, Fortran

Report on PACT 1A compiler

Reports from sub-committees

c. SHARE Standards Adopted

1) Change on SHARE standard 716 panel such that sense exit No. 1 be wired directly to skip to channel No. 1

2) Programs submitted after May 14, 1956 will include catalog entry cards. Format of cards and outline of classification approved.
(IBM Poughkeepsie will prepare catalog cards for prior programs.)

3) Chinese Binary Card Format - (Binary Cards to be distributed will remain in row-wise format.)

4) Absolute binary card decks will be distributed only for programs to be used from operator's console.

5) "CAC" or "CAD" is mnemonic code for Copy, Add and Carry Logical word instruction.

d. Recommendations to IBM

1) Overflow -- Underflow

a) No automatic stop on floating overflow and/or floating underflow feature is to be provided.

b) Execution of floating operations which do not overflow or underflow is not to take more time than at present.

c) Whenever a floating OF or UF occurs in AC and/or MQ the location of the instruction following the offending instruction is to be stored at location 0000 and control is to be transferred to some cell (not 0000 or 0001) in low end of memory.

2) IBM urged to study extensively magnetic tape life and reliability.

e. Committees and Assignments

1) Permanent Committee to prepare and maintain SHARE Reference Manual

2) Logical Data Processing Committee

3) Education Committee to prepare outline of course for training computer personnel

4) New Programs Committee

5) Instruction Mnemonics Committee

6) Permanent Catalog Committee to review methods to maintain SHARE catalog

7) Committee to originate and distribute Monthly Check List to strengthen communication

8) Committee to Collect Programming Statistics

f. Appendix

1) Machine Configuration Chart

2) Typical 704 layouts

3) Diagram of 32K work core frame

4) Report of Education Committee to ACM Council

6. SIXTH MEETING August 22-24, 1956 Denver, Colorado

a. Attendance - Fifty-two member installations were represented.

b. SHARE Organization

1) Question of legal status of SHARE referred to committee for investigation.

2) The Executive Board shall be composed of seven members, which number shall include the officers of SHARE, any officers of the year immediately passed who are not re-elected to office, and a number of members chosen by nomination and election, sufficient to complete the seven-member body. It shall act as an advisory body to the executive officer of SHARE

3) Executive Board elected for 1956-1957:

| | |
|---|---|
| Chairman: | Frank Engel (WH) |
| Vice-Chairman: | Randall Porter (BA) |
| Secretary: | Joanne Edson (CS) |
| Other Members of Executive Board: | Paul Armer (RS)<br>Fletcher Jones (NA)<br>Walter Ramshaw (UA)<br>Jack Strong (NA) |

c. SHARE Standards Adopted

1) Identification of program decks for SHARE distribution:

First card will be a REM card containing program title and installation code.

Symbolic decimal program deck:

| | | |
|---|---|---|
| Columns | 73-76 | Program identification |
| Columns | 77-80 | Sequence number |

Binary card identification not changed. (See First Meeting, 0. c. 9). Recommended that new identification be used for second 100 cards.

d. Reports from IBM

1) Chinese binary is ready for field testing.

2) Internal clock now available by RPQ.

3) Sequencing device for the on-line punch will be available soon by RPQ.

4) 704's are being changed in the field to permit checking of the longitudinal redundancy bits. A 30-cycle delay is needed.

5) 704's will be changed in the field to give program detectable external indication of physical end-of-tape.

6) The Copy instruction which is presently used will not be available on the 738 (Model III).

7) The 32000-word core will be changed in size.

8) Report on status of FORTRAN.

e. Recommendations to IBM

1) SHARE requests IBM to make no changes which affect existing 704 operations other than those already requested by SHARE.

2) SHARE urges IBM to distribute the 704 Information Bulletin monthly.

3) SHARE urges IBM to revise the Add and Carry Logical Word instruction in the manner as requested by SHARE, such that the Q bit is not cleared on execution.

f. Committees and Assignments

1) Mnemonics Committee dissolved. Its function can more efficiently be performed by IBM.

2) New committee formed to investigate diagnostic system for the prevention of machine stops.

3) Legality and Individual Membership Committee

4) Committee to prepare Schedule of Meetings.

g. Other topics discussed

1) Idle time and its relationship to efficient operation.

2) Tape reliability

3) Aptitude testing

4) Debugging

5) 704 layout

6) Critique of SHARE'S first year

h. Appendices

1) Attendance

2) Index of UA Library Programs

3) Mathematical Routine Questionnaire

4) Programmer Training Committee Questionnaire Results. Syllabus of Proposed Training Program

5) Diagram of 32000 Word Core Frame

6) An example of FORTRAN coding

7) Graphical representation of relationship between 704 waiting time and work load.

8) Preliminary report on General Electric tape reliability test.

9) Description of Midwestern Universities Research Association (MU)

10) Index

7. SEVENTH MEETING December 13-14, 1956 New York, New York

a. Attendance - Sixty-eight member installations were represented.

b. SHARE organization

1) Recommendation by Executive Board concerning government of SHARE was adopted.

2) A formal committee-subcommittee structure was approved.

c. SHARE Standards Adopted

1) Standard format for abstract cards for programs in preparation or revised.

d. Reports from IBM

1) Storage life and reliability of magnetic tape.

2) Description of 774 Tape Data Selector.

3) Overflow-underflow change requested by SHARE being worked on.

4) Chinese binary being field tested at North American.

5) Add and Carry Logical Instruction now available.

6) Store Zero instruction will now be maintained.

7) End of Tape Test instruction being put on all machines.

8) Reports on FORTRAN: Subroutine structure, distribution system, experience to date, future aspects, debugging, and input-output.

e. Recommendations to IBM

1) The use of mnemonic code SLT on the improved 704 is deplored, as it conflicts with a SHARE extended operation code.

f. Committees and Assignments

1) Education Committee reorganized as "Public Relations in the Computing Field" Committee.

2) 704 Model 3 System Committee founded to study the establishment of a uniform system as well as a uniform language in the next machines.

g. Other Topics Discussed

1) Indoctrination session for new members held prior to general meeting.

2) Proposal concerning legal counsel for SHARE.

3) Panel discussion of operator programs employing tape-to-tape operation, new debugging techniques, and snapshot dumping.

h. Appendices

1) Attendance

2) Report of the Education Committee

3) Chart of SHARE Committee Structure.

4) Committee Chairmen and Members

5) Report on status of SHARE Assembly Program

6) Panel discussion.

7) SHARE Monthly Checklist Questionnaire.

8) Chart of SHARE Machine Configurations.

9) Chart of Operating Characteristics.

8. EIGHTH MEETING April 24-26, 1957 Dallas, Texas

a. Attendance - Sixty-seven member installations were represented

b. SHARE Organization

1) New By-Laws adopted.

2) Slate of officers nominated for 1957 - 1958.

c. SHARE Standards Adopted

1) Method of distributing program decks on request.

2) System of program revision.

d. Reports from IBM

1) 727/792 tape unit compatibility on 704 and 709.

2) 150/500/1000 line printers and tape unit compatibility

3) 704/709 RAMAC

4) COMTRAN

5) Programmable trap interval timer.

e. Preparation for 709

1) Minimum 709 defined.

2) Objectives of system listed.

f. Other Topics discussed

1) Floating Point Trap (Floating underflow-overflow) - informal survey of installations ordering.

2) Decided not to add mean error-free time to machine performance statistics.

3) Report on SHARE Assembly Program.

4) Panel on direct input devices.

5) Panel on improvement of program material for SHARE distribution.

6) Panel on program checkout techniques.

7) Panel on programming tricks and conventions.

8) Panel on administration of computer facilities.

9) Operational reports on FORTRAN and PACT 1A.

10) Panel on unexpected arithmetic difficulties due to machine characteristics.

11) Panel on flow charting techniques.

12) Panel on techniques for handling completed program library.

g. Appendices

1) Attendance

2) By-Laws

3) Standing and Ad Hoc Committees

4) Transfer of Responsibility for SHARE Distribution

5) Machine configuration and Operating Characteristic Charts.

6) Supplementary Input-Output Functions for 704

7) FORTRAN Status Report

8) Report on SHARE Assembler

9) Results of Program Usage Questionnaire

10) Report of Committee on SHARE Distribution

11) Report of Committee on Public Relations in the Computer Field

12) Report of SHARE Reference Manual Committee

13) Report of Programmer Training Committee

14) Report of Mathematics Subcommittee

15) Guide for the preparation of program critiques

16) Report of Utility Programs Subcommittee

17) Report of 709 System Committee

9. NINTH MEETING October 1-3, 1957 San Diego, California

a. Attendance - Seventy-six member installations were represented.

b. SHARE Organization

1) Executive Board elected for 1957 - 1958 :

| | |
|---|---|
| Chairman: | F. V. Wagner (NA) |
| Vice-Chairman: | B. Ferber (CS) |
| Secretary: | H. S. Bright (WB) |
| Executive Board: | L. H. Amaya (CL) |
| | P. Armer (RS) |
| | W. A. Ramshaw (UA) |

(Ex-officio seventh member of Executive Board is the outgoing Chairman, F. Engel)

2) Quorum reduced to one-half of membership.

3) Amendment simplifying election procedure adopted. (New elections for all offices simultaneous; one election for board members regardless of number of nominees).

4) By-Laws rearranged to put all paragraphs on attendance together.

5) Future meetings will be scheduled in such a way as not to be contiguous with related major technical meetings. Each meeting will be three days in length, Monday - Wednesday or Wednesday - Friday. Two meetings will be held a year.

c. SHARE Standard Adopted

1) Standards for column binary agreed on:

Combination 9-7 punch in column 1 shall designate column binary card.

Card images on tape shall agree exactly with standard row binary except for bits 9 and 11 in the first word corresponding to the 9 - 7 control punch in the card.

The SHARE standard 714 board shall be wired to permit the program to look ahead to see if the next record is column binary or BCD.

d. Reports from IBM

1) Organization of Applied Programming Department of IBM.

2) 704 and 709 Publications

3) New library programs.

4) FORTRAN II plans.

5) COMTRAN, proposed IBM common language translator.

6) Special engineering appiications and special hardware requests.

e. Committee and Assignments

1) Ad Hoc committee formed to consider distribution of informal material within SHARE.

2) Ad Hoc committee on type wheel standards recommended 8 - 4 character change. Mail ballot will be taken.

f. Preparation for 709

1) Discussion of reasons for various choices of machine configurations.

2) Discussion of interchangeable sets of type wheels.

3) Panel discussion of 709 system.

4) Ad Hoc committee formed to study elimination of Q bit from 709.

g. Other Topics Discussed

1) Proposed FORTRAN Source Language Translator.

2) Survey of non-engineering applications of the 704.

3) Panel discussion on output generators.

4) Panel discussion on curve plotting techniques.

5) CORBIE automatic operator system.

6) Panel on experience with 32000-word core storage.

7) Panel on FORTRAN experience

8) Panel on Universal Computer Language.

h. Appendices

1) Summary of decisions

2) Attendance

3) SHARE committees

4) IBM reports

5) Non-Engineering 704 usage

6) Progress report on 709 Supervisory Control

7) SHARE members 704 and 709 Configurations.

8) Report of Mathematics Committee, including report of Scope Subcommittee.

9) Report of Utility Programs Subcommittee.

10) Report of the SHARE Reference Manual Committee.

11) Report of the Committee on Column Binary

12) Report of Committee on 709 Type Wheel Standards.

13) Report of Committee for Scheduling of Future SHARE Meeting.

14) Notes on Panel Discussion on Administration.

15) Notes from Meeting on Universal Computer Language.

16) Proposed FORTRAN Source Language Translator.

17) Summary of Answers to 704 Clock Questionnaire.

# FOREWORD

Upon getting thoroughly into the problem of preparations for the IBM 704, three installations in the Los Angeles area began to have informal discussions concerning their individual plans. Having been pleasantly surprised by the successful cooperative effort for the design and coding of Pact I, a favorable climate existed for a similar joint activity in connection with program development for the 704. Accordingly, Rand, Lockheed, and North American seriously began to consider standardization. A fortunate circumstance was the seminar held by IBM in Los Angeles during the week of August 8, 1955. This brought representatives of several other western installations together, and the idea was discussed among them.

The mutual respect that the participants in these discussions had for the programming competence of the others soon brought the realization that an "isolationist" attitude no longer existed, and almost all professed themselves as quite willing to accept the ideas of others, even to the extent of obsoleting things already done within their installations. It was unanimously agreed that a full-scale attempt should be made to bring SHARE into being. Since it seemed almost too late to do it on a nationwide basis, extreme haste was necessary, and the initial meeting of SHARE was called for the week of August 22, 1955.

In spite of such short notice, almost all potential 704 installations throughout the country responded with alacrity. All expressed a desire to participate, and attendance at the first meeting was gratifyingly large. The following Proceedings indicate the high degree of success achieved thus far.

## PROCEEDINGS OF THE FIRST MEETING OF SHARE

### Addenda et Errata

| Page | Line | Correction |
|---|---|---|
| 3 | 18 | Substitute "Computing Bureau" for "Programming Research." |
| 3 | 23 | Add "Data Processing Center" after "World Headquarters." |
| 4 | 4 | Delete "Robert Douthitt." |
| 9 | 26-27 | Strike out the remainder of the sentence beginning with "assembly program . . ." and substitute "decimal symbolic deck." |
| 10 | 29-33 | Delete this calling sequence and substitute: |

| LOC | OP | ADDR | TAG | DEC |
|---|---|---|---|---|
| $\Delta$ | TSX | | C | |
| $\Delta + 1$ | HTR | $q_1$ | | $q_2$ |
| $\Delta + 2$ | HTR | Y | | |
| $\Delta + 3$ | TRA | B | | |
| $\Delta + 4$ | Forwarding Location | | | |

| Page | Line | Correction |
|---|---|---|
| App. 2-1 | | Add (bottom of page):<br>"Components selected may be indicated by small decimal integers. Thus, Tape $(221)_8$ may be addressed as Tape 1, Drum $(302)_8$ as Drum 2, etc. No addresses need be written for RCD, RPR, WPR, WPU, WTV, SLF, SPR, SPT, SPU, CFF, IOD." |
| App. 5-1 | Program 5 | Write in "LA" under each heading across page. Add "Floating Point" under "Remarks." |
| App. 8-1 | 27 | Substitute "General Motors Research" for "General Motors Corporation." |
| App. 9-1 | 15 | Substitute "PK" for "PR" and substitute "Computing Bureau" for "Programming Research." |

Recorded
10-20-55
JEK

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

I.

## REPRESENTATION AT THE FIRST MEETING OF SHARE

BOEING AIRPLANE COMPANY
John Jordan
Randall Porter

CALIFORNIA RESEARCH CORPORATION - LA HABRA
Wesley Harker TOM MUELLER
William J. West

CURTISS-WRIGHT CORPORATION
John A. DeVries ✓
WARD SANGREN (FORMER OAK R)

DOUGLAS AIRCRAFT COMPANY, INC. - EL SEGUNDO DIVISION*
Walter C. Schlieser

GENERAL ELECTRIC COMPANY - AIRCRAFT GAS TURBINE DIVISION GE LYNN
Don Shell ALLAN HELLER

GENERAL MOTORS CORPORATION DON HART - CHF,
Jim Fishman BOB PATRICK
ED JACKS

HUGHES AIRCRAFT COMPANY
Leo A. Aroian
Essor Maso

INTERNATIONAL BUSINESS MACHINES CORPORATION - PROGRAMMING RESEARCH, POUGHKEEPSIE
Willard Bouricius ✓

INTERNATIONAL BUSINESS MACHINES CORPORATION - SANTA MONICA OFFICE*
Steve Jamison

INTERNATIONAL BUSINESS MACHINES CORPORATION - WORLD HEADQUARTERS
John Greenstadt ✓ ELAINE BOEHM STAN [illegible]

LOCKHEED AIRCRAFT CORPORATION - ACCOUNTING, BURBANK*
John Caywood
Norvell Johnson

LOCKHEED AIRCRAFT CORPORATION - BURBANK
Lee Amaya
Harvey Bratman LUKE
Carl Tross

LOCKHEED AIRCRAFT CORPORATION - GEORGIA
Robert Bosak
D. P. Haggerty
J. R. Raynolds ✓ FINE

I. REPRESENTATION (Continued)

LOCKHEED AIRCRAFT CORPORATION - MISSILE SYSTEMS, RESEARCH LABORATORIES
J. E. Barry
Ed Braun
~~Robert Douthitt~~
M. Lakin

LOCKHEED AIRCRAFT CORPORATION - MISSILE SYSTEMS, ENGINEERING DEPARTMENT
R. W. Bemer ✓
B. F. Handy ✓
E. H. Hawekotte
C. M. Wimberley

NATIONAL SECURITY AGENCY
Thomas E. McCool (GENE)

NORTH AMERICAN AVIATION, INC.
Ray Berman ✓
Robert Boden
Fletcher Jones ✓
Owen R. Mock ✓
Jack A. Strong ✓
Frank V. Wagner ✓

THE RAMO-WOOLDRIDGE CORPORATION*
A. John Carlson, Jr.

THE RAND CORPORATION
Paul Armer
Eugene Gordon
Irwin Greenwald ✓
John Matousek ✓
Cliff Shaw
Tom Steel

UNITED AIRCRAFT CORPORATION
Roy Nutt ✓ WALT RAMSHAW

UNIVERSITY OF CALIFORNIA - LIVERMORE
Robert Hughes DICK REINHOLD

UNIVERSITY OF CALIFORNIA - LOS ALAMOS
Louis Gatt ✓
Edward Voorhees ✓

REDSTONE
SASSENFELD
HOELZER

WESTINGHOUSE
FRANK ENGEL

*Nonparticipating

## POLICY COMMITTEE PROCEEDINGS

For purposes of definition, the "SHARE" Policy Committee is considered to be that part of the "SHARE" organization which has as its members at least that person from each 704 installation who is empowered to commit his organization on all matters involved in a cooperative activity. The chief duty of the Policy Committee is to direct the activities of the Working Committee, which is composed of individuals familiar with the details of machine methods and programming.

At the opening of the "SHARE" conference, Paul Armer of the Rand Corporation welcomed the attendees and gave a brief summary of the genesis and expected scope of the organization. There followed a discussion on the aims of "SHARE", during which the final agenda for the week to follow was evolved. The agenda was based on a fifteen-point program, involving the following topics:

1. Mnemonic operation code.
2. Assembly program.
3. Binary card format.
4. Utility programs.
5. Subroutines.
6. Conventions of use of index registers, indicators, and switches.
7. Print wheel format.
8. Code diagnostics.
9. Additional operations.
10. Binary point location description.
11. Language conformity.
12. Systems of use of utility programs.
13. Program identification.
14. Abstractions.
15. Continuance of SHARE activities.

An election of officers was called for, and nominations and voting followed. The following are the officers of the "SHARE" Policy Committee as elected August 22:

| | |
|---|---|
| Chairman: | Jack Strong |
| Vice-Chairman: | Donald Shell |
| Secretary: | Fletcher Jones |

Many representatives were not familiar with the methods in use at several of the installations. In order to acquaint everyone with methods currently in use and being planned, a cursory examination of each of the points on the agenda was undertaken. As each item on the program was surveyed, those who had pertinent programs, definite plans, or ideas gave a brief description of these. At the conclusion of this discussion, it was decided that the first three items on the agenda, mnemonic operation code, assembly program and binary card format, were prime instruments in any transfer of information, and, as such, should be given precedence over the other points to the extent of having discussion and basic decision on these items in the Policy Committee before discussion of any other points.

It was noted that four organizations had progressed in the design of machine methods to the point of having written assembly programs and adopted binary card forms and mnemonic operation codes. These organizations are United Aircraft Corporation, Los Alamos, General Electric-Evendale, and IBM. These items were taken up in order of appearance in the agenda, with representatives of the organizations mentioned giving a detailed picture of their particular method. The results of the presentations and subsequent discussions follow.

1. Mnemonic Operation Code

It was agreed that all the existing operation codes were good and easy to use, and that the choice between these was largely one of personal taste. Many representatives, however, were in favor of an extended order list, the use of which would enable a programmer to write a particular operation code which would designate the desired input-output or sensing device.

Only one installation of the four had a strong reason for not adopting a different code from that they were using. IBM has spent a great deal of money and time in the preparation of literature, training of customer engineers and other personnel, and the printing of circuit diagrams and reference material -- all using the New York IBM code. Through a telephone call to the New York IBM offices, it became evident that the mnemonic operation code was the one item on the agenda which was considered unchangeable by IBM. IBM, however, did not object to an extended order list or to the naming of operations to be requested in addition to those already used.

It was decided by a vote that IBM's operation code is that which will be used by "SHARE". It was agreed that the Working Committee would compile a mutually acceptable extended order list if such was desired by a majority of Working Committee members.

2. Assembly Program

It became apparent during the discussion and description of the available assembly programs that each of the installations had written their assembler with an eye to some particular feature or features. (While the main consideration of the Los Alamos assembly is the quick reassembly, G. E. desired a variable field card input and UAC a compounded address.) It was agreed that one assembly program could economically include most of the features desired. Towards the goal of constructing such an assembly program, it was decided that some existing assembler should be adopted as a framework on which to add the features considered to be most important. A vote was taken and the IBM assembly program was designated as that which will be the framework of the final "SHARE" assembly. The Working Committee was instructed to specify the necessary changes which would make the "SHARE" assembler acceptable to all.

3. Binary Card Format

The various binary card forms in use were discussed. John Greenstadt proposed a compromise of all those forms mentioned to that described in appendix (1). This was unanimously accepted as the "SHARE" binary card form.

Time permitted the discussion by the Policy Committee of several other points on the agenda. The results of these discussions follow.

1. Print Wheel Format

It was noted that, during the March, 1955, 701-704 Symposium held at The Rand Corporation, a final 704 print wheel configuration had been decided upon by all the then prospective 704 installations. This format was accepted by IBM representatives as that which they would implement in the design of the 704 print wheels. Through a phone conversation with the New York offices of IBM, it was found that the print wheel design now considered to be standard by IBM differs from that suggested in March. It was agreed without objection that a letter protesting the decision of IBM in this matter be sent from SHARE. It was also agreed that the letter from SHARE should include a request for the change of the print wheels to conform to the configuration accepted in March. It was further resolved that a copy of this letter would be sent to the IBM branch office servicing each of the SHARE members.

2. Standardization of Machine Configuration

It was thought to be appropriate to investigate the feasibility of having a "standard 704". This would provide every 704 machine with the same number of core frames, drums, etc. It was immediately apparent from the discussion of this subject that such standardization would not be practical with the 704. This is because use of computing equipment varies among the 704 installations, requiring a "variable machine." The thought was projected that this topic would be one of significance when new computing machinery is announced.

The discussion was very productive in that it supplied to all present the machine configurations on order by the installations represented. A chart showing the types of 704's on order by SHARE members may be found in Appendix (3).

For purposes of reference, a "minimum 704" was described. This configuration will be of aid to members in the writing of programs for distribution in SHARE. All the components of the 704 used in programs for distribution in SHARE, other than those components in the "minimum 704," must be noted in the program writeup. The composition of the "minimum 704" may be found in Appendix (4).

3. Binary Point Description

In the discussion of this subject, it was found that only two installations preferred to describe the location of the binary point by counting bit locations from the right of the binary word, the remaining installations having adopted the convention of counting from the left. It was decided by vote that the binary point will be described by counting from the left to the right of the word. Thus the binary word having the form xxx.xxxx shall be said to have a scale factor, or "q", of 3, and the binary word having the point 37 bits to the left of the rightmost bit shall be said to have a scale factor of -2.

4. Language Conformity

The discussion of this subject led to the conclusion that a glossary of terms used in connection with 701-704 computing is needed to augment existing dictionaries of computing terms. The Working Committee was charged with the compilation of such a glossary.

5. Continuance of "SHARE" and Method of Distribution of Information

It was observed during the discussion of this topic that subsequent meetings of SHARE should coincide with meetings of other organizations involved in computing and thereby lessen the burden of travel justification placed on SHARE representatives. It was noted that four or five meetings of great interest to computing personnel occur each year, and these would provide ample opportunity for SHARE conferences once the initial backlog of SHARE work has been accomplished. The second meeting of SHARE was scheduled for September 12th through 13th in Philadelphia. This meeting will immediately precede the Philadelphia meeting of the Association for Computing Machinery, September 13th through 15th.

It was decided that some member of SHARE be designated the distributing agent for inquiries, memoranda, information, and finished programs. To this end, the following was agreed upon:

a. For distribution of programs which have been checked out, written up and made ready for general dissemination within SHARE, Mr. George Petrie of IBM, Poughkeepsie, New York, will act as agent. All writeups, diagrams, etc, should be in a reproducible form.

b. For all other purposes, the secretary of the SHARE Policy Committee will act as the distribution and information agent. Items sent to the secretary for distribution should be in a reproducible form or should be reproduced before being sent to the secretary, whichever is most convenient.

c. On receipt of an item for distribution the agent involved will, within a reasonable time, process the item and send copies directly to the members of SHARE.

Having discussed the subjects listed above, the Policy Committee instructed the Working Committee to investigate and make decisions on the other points of the agenda, adding, where necessary, to the decisions of the Policy Committee on items previously covered. The Policy Committee adjourned.

## WORKING COMMITTEE PROCEEDINGS

The first item of business called for in the Working Committee was the election of officers. Nominations were made and voting followed. The officers of the SHARE Working Committee as elected August 24th are:

| | |
|---|---|
| Chairman: | Irwin Greenwald |
| Secretary: | Fletcher Jones |

The Working Committee had been instructed by the Policy Committee to dispose of the items on the agenda. For purposes of collecting all activities on each item examined by the Working Committee, the Proceedings will follow the outline furnished by the agenda. In this categorization, no attempt has been made to order the happenings chronologically.

1. Mnemonic Operation Code

A vote was taken to determine the desirability of an extended order list. The decision was made to design an extended order list to be added to the IBM operation codes. A subcommittee, composed of R. Bosak, D. Shell and J. Greenstadt, was organized for this purpose, and devised that which is to be found in Appendix 2. This was adopted, without objection, as the SHARE extended order list.

2. Assembly Program

The Policy Committee had charged the Working Committee with the modification of the NYAPI (IBM) assembler so as to satisfy the needs of all installations. After long discussion, it was decided that the IBM assembly should be modified to include the following features:

a. The ability to accept and print out a variable field.

b. The ability to facilitate a short reassembly-one which would not require the entering into the machine of the complete ~~assembly program and which would not require the number of passes necessary for the present IBM assembly program.~~ decimal symbolic deck.

c. The ability to accept and interpret compounded addresses and compounded decrements. This allows the algebraic combination of symbols in the address and decrement parts.

d. The feature which would allow the assembly program to punch out origin cards for library programs.

e. The ability to punch out origin tables which may be changed with a minimum of hand keypunching, or, that which is preferred, no keypunching whatsoever.

f. The ability to assign erasable storage in a quasi-automatic fashion. This would allow the programmer, when storing an answer, to call the address "the result of step n". Later references to this address would have the same form, i.e., "CLA R(n)". The assembly, in this case, would automatically assign absolute addresses to these references, making unnecessary the naming of erasable storage during coding.

John Greenstadt expressed confidence that all these features could be incorporated into NYAP1. John will delve deeper into the addition of these features upon his return to New York, and will give a report as to his findings at the second meeting of SHARE. Proponents of the suggested changes were asked to submit proposals at the next meeting as to the embodiment of their ideas into NYAP1.

3. Binary Card Format

There were no suggestions for addition to the decisions made on this subject in the Policy Committee.

4. Utility Programs,

5. Subroutines and

6. Conventions of Use of Index Registers and Indicators and Mode of Input-Output of Subroutines

It was decided that these three subjects should be taken up as one, since many problems involved in programming are common in these items.

There was much utility program and subroutine coding experience represented in the Working Committee, since many programmers do their first experimenting with a new machine by writing input-output programs and elementary function subroutines. Everyone with feelings for and experience in these items gave a summation of their ideas. It was decided that certain rules governing subroutines should be specified at this point in the discussion. These follow:

a. Fixed and floating point subroutines shall be separate entities.

b. Subroutine shall always be entered by a calling sequence.

c. The transfer point shall always be the first instruction in the subroutine.

d. Every effort will be made to use the following form of calling sequence:

| | | | LOC | OP | ADDR | TAG | DEC |
|---|---|---|---|---|---|---|---|
| ~~Δ~~ | ~~TSX~~ | ~~C~~ | Δ | TSX | | C | |
| ~~Δ+1~~ | ~~$q_1$~~ | ~~$q_2$~~ | Δ+1 | HTR | $q_1$ | | $q_2$ |
| ~~Δ+2~~ | ~~HTR~~ | ~~Y~~ | Δ+2 | HTR | Y | | |
| ~~Δ+3~~ | ~~TRA~~ | ~~B~~ | Δ+3 | TRA | B | | |
| Δ+4 | FORWARDING LOCATION | | | | | | |

As in the above sequence, index register C will always be that which is used in subroutines.

The argument(s) will be placed in the following units of the machine, in the order indicated: (1) accumulator, (2) MQ, (3) core storage location specified in linkage. Thus, if there is only one argument, it will be found in the accumulator; if there are two, they will be found in the accumulator and MQ; etc. Output of the subroutine shall be stored in the same fashion.

In step Δ+1, the q's represent scale factors; $q_1$ the scale factor of the input and $q_2$ that of the output.

Δ+2 is an example of a step indicating an input parameter is to be found in core storage. If there are several input parameters in core, the steps designating the addresses of these will be consecutive in the calling sequence. The operation associated with an input parameter shall always be a "HTR".

It was decided that there should be two returns to the master program; one if the execution of the subroutine was unsuccessful, and a successful completion return. This, it was pointed out, would eliminate stops in subroutines due to faulty scaling, etc. The first return location in our example sequence is at step Δ+3. Step Δ+4 is the site of the successful return in this instance.

e. If any index registers, other than "C", are used by the subroutine, these will be restored to their original condition within the subroutine before exiting.

f. If an unsuccessful return is made to the master program from a subroutine, bits defining the nature of the error will be placed in the accumulator. These bits patterns and their meanings will appear in the program write-up.

g. If any information is conveyed to the master program from the subroutine via the overflow triggers, these will be preset as needed within the subroutine. Notice will be given in the write-up if the overflow triggers convey information. If no information is conveyed via overflow triggers, the condition of overflow triggers on exiting from the subroutine is not guaranteed. These indicators may be used within the subroutine without restoration or setting.

h. Any sense lights used within the subroutine will be restored before exit to the master program.

i. A calling sequence for a floating point subroutine will appear as that in (d) above, with the exception that there will be no reference to scale factors.

j. Negative numbers appearing in the calling sequence for use in negative scale factor writing will be in the form of 2's complement.

k. If a double precision routine is being entered, the first argument will be found in the accumulator and MQ, any second argument in some core address specified in the calling sequence and the next consecutive address, etc. Output from the subroutine shall be stored in the same fashion.

l. In the one case not governed by input-output rules listed above, if a subroutine having n input values yields n + y output values, where $n \geq 2$ or the routine is double precision, the calling sequence must have y core addresses specified as storage sites.

m. A 12 in 80 punch will indicate end-of-file on card reader.

It was next decided that a basic set of elementary functions should be decided upon. To this end a subcommittee, composed of W. Harker, R. Nutt, L. Gatt and J. Greenstadt, studied the suggestions for those elementary functions thought to be necessary and arrived at the following list of necessary subroutines:

Square Root
Sine-cosine (one routine)
Exponential (e)
Logarithm (base e)
Arc tangent
Sinh-Cosh
$X^a$

These routines comprise the SHARE elementary function set.

Discussion next moved to the choice of abstractions, utility routines and mathematical subroutines. Many programs were suggested, but it was realized that initial efforts in the programming direction should not attempt to be all-inclusive, so decisions were made as to which programs were very useful or necessary. A chart, listing these programs, may be found in Appendix (5). Included in this list are those programs to which there already has been considerable effort devoted. These would have been finished by the installation concerned regardless, so their addition to the assignment sheet merely insures their dissemination in SHARE. Also, on this chart is the assignment of each program to an installation for programming. Assignment was on a volunteer basis. Those installations indicating that program specifications will be prepared by September 12 will also be responsible for the programming of the routine involved. Appendix (9) will be useful in "decoding" the assignment chart.

7. Print Wheel Format

This subject was discussed to a final conclusion in the Policy Committee meeting.

8. Code Diagnostics

After lengthy discussion and much divergence of opinion on this topic, it was decided that code diagnostics is a category which is highly specialized from one installation to another. Many different modes of diagnoses will be used in many different ways. It was decided, therefore, that there will be no SHARE diagnostic routine as such, but anyone writing such a routine is encouraged to submit this to SHARE for distribution. Several general diagnotic routines, as will be noted on the assignment chart, are to be written. These will be invaluable aids in initial machine use.

9. Additional Operations

There were many suggestions for operations to be added to the order list. It was pointed out that SHARE should weigh the usefulness of any suggested operation against the possibility that all operation bit arrangements might be depleted before experience on the 704 could guide a choosing of additional operations. Toward the goal of selecting the most useful operations to be added, a subcommittee composed of I. Greenwald, D. Shell and J. Greenstadt was instructed to investigate the suggestions and compile a list of operations which SHARE would request of IBM. The activities of this subcommittee resulted in the compilation of the following list:

RPQ - INSTAL CHG

RPQ BEING ENGINEERED

BY MID 21 SEPT

PAUL KNAPLUND NEW RPQ

1. Copy and add logical word
2. Exclusive or $\overline{\text{MATCH}}$ = 1, MATCH = 0
3. Store index in address PART
4. Place index in address PART OF ACCUM
5. Logical right shift
6. Store tag
7. Backspace file (On a backspace file instruction during a beginning-of-tape condition, a skip is desired. A skip is also desired on the backspace record instruction when a beginning-of-file condition is met.)
8. Read tape backward (If this instruction is adopted by IBM, it is expected that there also would be available a skip on beginning-of-file condition when backspace tape is given.) LOOKS IMPOSSIBLE

1,500/MO MAY 56? FIXED HALF WORD DATA BITS 16 & 17 IN BINARY CONS (TO OR FROM STORAGE)

RPQ - ADD TRAN MQ RESET 32 [illegible] CARD IN 4 PIECES

It was pointed out that perhaps, due to some unknown characteristics of the tape units, IBM would be unable to supply both (7) and (8). It was decided, in this event, that (7) is the operation of choice. It was indicated that the innovation of either (7) or (8) might make necessary the sacrifice of the high-speed rewind. The decision was made that SHARE will have traded well if either instruction should replace the high-speed rewind.

It was unanimously agreed that a form letter, asking that the instructions above be added to the operation list, would be sent from each installation to Dr. De Carlo of IBM.

## 10. Binary Point Location Description

Conclusive decisions on this topic were made in the Policy Committee.

## 11. Language Conformity

The Policy Committee had directed the Working Committee to compile a glossary of terms to augment existing computing dictionaries. It was decided that this should be done in a subcommittee. The Los Angeles vicinity installations will organize a subcommittee for purposes of compiling a glossary. This subcommittee will submit proposals at the second meeting of SHARE.

## 12. Systems of Use of Utility Programs

It was found that systems of utility program usage was largely an internal function with each installation and that SHARE's activities depended in no degree upon agreement on this subject. It was interesting to note, however, that, of those installations represented, only two did not adhere to the practice of entering utility programs into the master program at the time of assembly. References marginally associated with this subject may be found in III, 4.

13. Program Identification

It was agreed that some method of readily identifying the source of programs written for SHARE should be adopted. The write-up form to be used with SHARE programs partially accomplishes this purpose, but does not allow for the identification of decks of cards. It was decided that all decks of cards to be distributed in SHARE will have, in the first two columns, characters defining the originating installation. Characters were assigned to each SHARE installation. A list of these may be found in Appendix (9). It was decided that the six columns immediately following the installation characters would be reserved for internal use in each installation. These columns, however, will not contain the same data for two different SHARE programs written by the same installation.

In connection with card identification, it was decided that all decks sent to SHARE for distribution will have a sequence number in some location specified by the program write-up.

14. Abstractions

It was decided that, for the immediate purposes of SHARE, abstractions were not necessary and the programming of these should be put aside until more urgent work is accomplished. There will be, however, a proposal for a matrix abstraction rendered at the second meeting of SHARE.

15. Continuance of SHARE

This subject was discussed to a conclusion in the Policy Committee meeting.

Roy Nutt of UAC gave an interesting talk on the "Cross-Bar Switching" arrangement to be used at East Hartford. This is the method by which UAC plans to manipulate 727 tapes without having to shut off power. UAC will build the necessary apparatus at an approximate cost of $30,000. Their switching unit will control as many as 10 tapes at once.

The first meeting of the Working Committee of SHARE adjourned on Friday afternoon, August 26, 1955.

pg. 2-8 1

# MANUAL OF OPERATION

for

IBM 704

using

# CAGE

**Compiler and Assembler by General Electric**

Prepared and published by

Computer Techniques Development Subsection
Investigations Section
Aircraft Gas Turbine Development Department
General Electric Company
Evendale, Ohio

## TABLE OF CONTENTS

## INTRODUCTION

This volume is a manual of operation for the IBM Type 704 Electronic Data Processing Machine. It is written from the point of view of a programmer using CAGE--Compiler and Assembler by General Electric.

The manual is divided into two parts. The first part describes the essential characteristics of the 704 from the programmer's point of view. It duplicates in large measure the contents of IBM's own manual of operation for the 704. However, the nomenclature and especially the mnemonic code has been altered throughout to conform to that which must be used with CAGE.

The second part of the manual is a description of CAGE itself. It is written from the users viewpoint and is intended to describe in considerable detail the features of this essential tool.

The entire manual has been published in this loose leaf form in order that it may be a simple matter to make future additions and corrections. It is anticipated that the user of this manual will want to put program write-ups, listings and so forth, in the binder for ready reference. One might also want to do such things as removing pages 40 and 41 of part one and mounting them on heavy paper or cardboard for easy access. In general the attempt has been made to make the manual as easy to use as possible.

It is anticipated that additions and revisions to this manual will be necessary in the future. When such is the case, appendices will be issued to cover all required changes.

A great deal of the information in part one is covered by IBM copyrights and is used with their permission.

August 3, 1955

# PRELIMINARY REPORT ON REGIONAL-SYMBOLIC (R-S) 704

by Louis Gatt

R-S is a system of coding which incorporates the important features of symbolic coding and regional coding.*

The main advantage of symbolic coding is the ease with which insertions and deletions are made. However, a partial assembly is not easily made with a symbolic-type program. The important asset of regional coding is the ability to do an assembly on a set of instructions A, that refer to instructions B when the instructions B are not present. In regional coding, one can make corrections on a small portion of this program without loading the entire decimal deck. On the other hand, it is not convenient to add or delete instructions in a program using regional-type coding.

In using R-S, the programmer codes in symbolic, with the restriction that sequencing numbers be in ascending but not necessarily consecutive order, and the assembly program does a regional type assembly. Hence, we have all the conveniences of symbolic coding in making insertions and deletions and also the convenience of regional coding since partial assemblies and reassemblies are possible.

R-S will be equipped to produce relocatable type binary cards; these will be described in a later report.

To avoid circumlocution, a square root calculation is attached using the language of R-S, upon which the following discussion is based.

* This program evolved from a method suggested by Mr. Edward Voorhees. The author would like to express his appreciation to Mr. Floyd Johnston of IBM for his suggestions during the development of the detailed program.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| S | Q |  | R | O | O | T |  |

**704 Regional-Symbolic**

Programmer

Problem SQUARE ROOT

Date 8-3-55

Page 1 of 1

| UNPUNCHED REMARKS |  | LOCATION |  |  | OPER-ATION | ADDRESS |  |  | TAG | DECREMENT |  |  | ± DATA / REMARKS ± |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | REGION | SEQUENCE | F |  | REGION | SEQUENCE | F |  | REGION | SEQUENCE | F | I FRACTION u ℓ |
|  | 9 | 10 12 | 13 16 | 17 | 18 20 | 21 23 | 24 27 | 28 | 29 | 30 32 | 33 36 | 37 | 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 |
| ASSUME ENTRY BY: | 0 | 005 | 0000 |  | TZ | 000 | 0001 | 0 | C | 000 | 0000 |  | SQUARE ROOT |
| α TSX S.0 C |  |  | 1 |  | TM | 2 | 32 |  |  |  |  |  | ERROR STOP |
| WITH ARGUMENT IN ACC |  |  | 1 | 1 | XSD | 1 | 0 |  | C |  |  |  | STORE INDEX |
| EXIT WITH SQ.RT. IN ACC |  |  | 2 |  | ST | 1 | 1 |  |  |  |  |  | STORE ARG |
|  |  |  | 3 |  | NA | 2 | 30 |  |  |  |  |  |  |
|  |  |  | 5 |  | LR | 0 | 1 |  |  |  |  |  |  |
|  |  |  | 6 |  | A | 1 | 1 |  |  |  |  |  | X |
|  |  |  | 7 |  | LR | 0 | 1 |  |  |  |  |  |  |
|  |  |  | 8 |  | A | 2 | 31 |  |  |  |  |  | 64 1/2 AT 8 |
| WITH THIS INITIAL |  |  | 9 |  | SAX | 2 | 32 |  | C |  |  |  | 3 TO INDEX C |
| GUESS WE NEED |  |  | 12 |  | ST | 1 | 2 |  |  |  |  |  |  |
| ONLY 3 ITERATIONS |  |  | 13 |  | CA | 1 | 1 |  |  |  |  |  | X |
| FOR CONVERGENCE. |  |  | 14 |  | FD | 1 | 2 |  |  |  |  |  |  |
|  |  |  | 15 |  | SQ | 1 | 3 |  |  |  |  |  |  |
|  |  |  | 16 |  | CA | 1 | 3 |  |  |  |  |  |  |
|  |  |  | 17 |  | FA | 1 | 2 |  |  |  |  |  |  |
|  |  |  | 18 |  | S | 2 | 30 |  |  |  |  |  | DIVIDE BY 2 |
|  |  |  | 18 | 1 | TX | 2 | 12 |  | C | 0 | 1 |  |  |
|  |  |  | 19 |  | SDX | 1 | 0 |  | C |  |  |  | RELOAD C |
|  | 0 |  | 20 |  | T | 0 | 1 |  | C |  |  |  | EXIT |
|  | 8 |  | 30 |  |  |  |  |  |  |  |  |  | 001000000000 |
|  | 8 |  | 31 |  |  |  |  |  |  |  |  |  | 100400000000 |
|  | 0 |  | 32 |  | H | 0 | 3 |  |  |  |  |  |  |

704 Regional-Symbolic

Column 9 will have a control punch to be described later. The location, address and decrement each have two parts; viz. (1) Region number and (2) Sequencing number:

1. The region number is any set of 3 digits with the restrictions:
   a. Region 000 is reserved for absolute numbers.
   b. Region 001 is reserved for temporary storage.
   c. Region 002 is used in the address or decrement when referring to other instructions within the same region or program. It is possible to code in Region 2, but if one codes in Region 2, then it will be impossible to refer to it from a different region.

2. The sequencing number is the sum of the 4-digit sequence number plus the 1-digit fraction (F). The fraction is used to insert instructions between consecutive sequence numbers, e.g., 1.1 was inserted between 1.0 and 2.0. Nine insertions may be made between any two consecutive sequence numbers in this manner. If more than nine insertions are necessary, then a completely new region may be inserted. There are other techniques also such as renumbering some of the sequence numbers. Note that we may leave gaps in our code, as was done after 0003.

Complements of numbers

The 1's complement of a number is the number with ones replaced by zeros and zeros by ones. The 2's complement is equal to the 1's complement plus 1 in the right-most position.

A 12-punch in column F of either address or decrement will cause the 2's complement of the address or decrement to be computed regardless of the region number used with the address or decrement.

An 11-punch in column F of either address or decrement will cause the 1's complement of the address or decrement to be computed regardless of the region number used with the address or decrement.

It is desirable in many cases (especially with temporary or permanent storage) to have regions whose sequence numbers are necessarily consecutive. Regions 000 and 001 described above are of such a kind. Therefore, addresses will have to be computed on the basis of the type of region referred to. There are two types of regions, called C-regions and D-regions.

C-regions are those whose addresses are computed by adding the sequence number to the origin; the fraction is considered to be zero. C-regions are regions 000, 001, 800, 801, ..., 999 and those regions whose origins have been assigned but for which no non-origin cards (to be described later) have been entered in the first pass. For example, if an origin has been assigned to region 13 and no 0, 3, 4, or 8 cards (these are non-origin cards) have been entered for region 13 during pass 1 of the assembly, region 13 will thereafter be considered as a C-region.

D-regions are those regions whose locations are determined by their order in the decimal deck along with the assigned origin. Since insertions and deletions are very easily made with D-regions, D-regions will normally be used for instructions, and C-regions will be used for data and eraseable storage. However, the choice of the region numbers will be up to the programmer completely. If it is desired, one can code in region 800, ..., 999 with the understanding that no expansions or contractions are allowed. To emphasize, the C-regions are designed to make data handling with R-S simpler. Coding in region 000 is equivalent to absolute decimal coding.

During the assembly process, it is not necessary to have the region numbers appear in any order. For example, it is possible to assemble the following regions in the given order: regions 005, 083, 067, 005, 192, 005. Notice that region 005 appeared more than once. This is permissable and allows one to insert region 192 within region 005. The only restriction in repeating region 005 is that the first sequencing number of the second region 005 block be greater than the last sequencing number of the first region 005 block. This is in agreement with the earlier statement on Page 1 of this report restricting sequencing numbers of a region to be in ascending order.

Operations:

The Los Alamos mnemonic operations will be entered in columns 18-20. One letter operations are entered in column 18, two letter operations in columns 18 and 19. The unused columns of the operation field are to be left blank.

In any case of overlap between operation and decrement or operation and address, the operation takes precedence. For example,

# PROCEEDINGS

# OF THE

# SECOND MEETING

# OF

# SHARE

Philadelphia

September 12-13, 1955

## TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

I.

## REPRESENTATION AT THE SECOND MEETING OF SHARE

CALIFORNIA RESEARCH CORPORATION - La Habra
Thomas D. Mueller

CURTISS-WRIGHT CORPORATION
John A. DeVries
Ward Sangren

GENERAL ELECTRIC CORPORATION - Lynn
Allen Keller

GENERAL MOTORS CORPORATION
Donald E. Hart
Edwin L. Jacks
Robert L. Patreck

INTERNATIONAL BUSINESS MACHINES CORPORATION - COMPUTING BUREAU - POUGHKEEPSIE
Elaine Boehm
Willard Bouricius

INTERNATIONAL BUSINESS MACHINES CORPORATION - DATA PROCESSING CENTER - WORLD HEADQUARTERS
John Greenstadt

LOCKHEED AIRCRAFT CORPORATION - Burbank
Richard C. Luke

LOCKHEED AIRCRAFT CORPORATION - Georgia
Gerald Fine
Jack R. Reynolds

LOCKHEED AIRCRAFT CORPORATION - MISSILE SYSTEMS - ENGINEERING DEPARTMENT
Robert W. Bemer
Benjamin F. Handy, Jr.

NORTH AMERICAN AVIATION, INC.
Ray Berman
Fletcher Jones
E. G. Law
Owen R. Mock
Jack Strong

REDSTONE ARSENAL
Helmut Hoelzer
Helmut Sassenfeld

I. REPRESENTATION (Continued)

THE RAND CORPORATION
Irwin Greenwald
John F. Matousek

UNITED AIRCRAFT CORPORATION
Roy Nutt
Walter Ramshaw

UNIVERSITY OF CALIFORNIA - Livermore
Richard E. Von Holdt

UNIVERSITY OF CALIFORNIA - Los Alamos
Louis Gatt
Edward Voorhees

WESTINGHOUSE ELECTRIC CORPORATION - Pittsburgh
Frank Engel, Jr.
Peter A. Zaphyr

INTERNATIONAL BUSINESS MACHINES CORPORATION - Pittsburgh Office*
Norman Moss

* Non-participating

## II.

## PROCEEDINGS OF THE SECOND MEETING OF SHARE

The second meeting of SHARE convened on September 12th at the Adelphia Hotel, Philadelphia. Three installations other than those represented at the first meeting of SHARE signified their intention to become SHARE members by attending, thus increasing the number of participants to 21, and leaving only one known prospective 704 installation which has not as yet made its intentions clear regarding SHARE participation.

The first item of business was the compiling of an agenda. After discussion, the following agenda was adopted:

1. Method of use of relocatable binary cards
2. Binary card form
3. Report from Willard Bouricius on decisions by IBM
4. Additional features added to assembly
5. Subroutines and reports on assignments made at the last meeting
6. Dictionary Committee report
7. Errata on SHARE Proceedings of First Meeting
8. Write-up distribution

During preliminary discussion of the agenda, it became evident that new information in the area of assembly programs required the reopening of the subject in this session. Information on the assembly program written by Roy Nutt of United Aircraft Corporation was made available in the form of a program description distributed at the meeting. Roy gave supplementary facts, augmenting the write-up. On the basis of this new information concerning United Aircraft Corporation's assembler, several of the representatives made known their preference for a revision of the former decision to use IBM's NYAP1 in a modified form.

One of the foremost considerations in the change of assemblers was the fact that the United Aircraft assembly already has in its structure many of the ideas involved in the changes that were to be required of NYAP1. This fact contributed to a second consideration in that the revision of the United Aircraft assembly does not require as much time to be made ready for SHARE use as does NYAP1. John Greenstadt made an estimate of from two to five months as being that time required to change and check out the modified version of NYAP1. Roy Nutt estimated that the United Aircraft assembly could be modified to SHARE's satisfaction and checked out within one month.

II. PROCEEDINGS (Continued)

The discussion of the assembler lasted for a day, during which all ramifications of both assemblies and the advisability of the change were explored. As a final proposal from United Aircraft Corporation to supply SHARE with an assembly program, Walter Ramshaw submitted a "packaged" offering. This follows:

1. United Aircraft will use the SHARE mnemonic operation code for both internal operation and communication purposes.

2. United Aircraft will incorporate all changes suggested by the SHARE Council which are not of a nature contrary to the philosophy of the United Aircraft assembler.

3. A converter program which will change symbolic decimal cards from the type used in NYAP1 to those which may be used by the United Aircraft assembler will be written by North American Aviation.

4. All of the above items, including the complete checkout of the changed United Aircraft assembler, will be consummated by October 15th.

At this point it was decided to consider the necessary changes to the UAC assembly, so that Ramshaw and Nutt could better evaluate the steps necessary to change this program and give a new estimate of time required if this was indicated. During the discussion of the changes to be made, the following were evolved as required modifications to the UAC assembly program.

1. The assembly program should produce relocatable output.

2. An illegitimate operation code should be indicated by an error symbol.

3. The assembly program should punch origins for relocatable library routines.

4. The program should accept scaled decimal input.

5. The printing of library subroutines during assembly should be controllable.

It was considered that a change making possible the unrestricted use of sequential symbolic notation was extraneous to the philosophy of the present UAC assembler, and, as such, could not be attained.

By way of a counterproposal, John Greenstadt said a concerted effort would be made to change the NYAP1 to conform with those changes thought to be necessary by the SHARE Council. His estimate of the time involved was revised to indicate that he would be able to have the program modified and ready for use by November 15th.

Greenstadt enumerated the following items as being those advantages NYAP1 holds over the UAC assembly program.

1. Octal addresses and decrements may be written.

2. A binary library tape is available.

II. PROCEEDINGS (Continued)

3. The IBM assembler requires the use of fewer tapes when library programs are put in the program to be assembled, since the UAC assembler has an intermediate step in which the library programs to be used are taken off a complete library tape and transferred to another, thus providing faster access if these routines are to be used several times.

4. NYAP1 has in its make-up a more complete error description, in that particular errors are differentiated from others, whereas the UAC assembly program gives a single error indication for all recognized mistakes.

5. NYAP1 provides an optional sequence code since the location may be sequentially numbered, whereas the UAC program has the restriction that only approximately 1,000 locations may be numbered when using a 4,096 cell frame. Any expansion in the amount of core storage results in a similar expansion in the possible number of instructions having numbered locations in the UAC program. (An 8,192 cell frame will facilitate the naming of approximately 3,048 locations in a program using the UAC assembler.)

A vote was taken to decide on the assembler to be used for SHARE purposes. This resulted in the selection of the UAC assembly program as that which will be used by SHARE participants. There are strong indications from all quarters that the UAC program will also be used in the internal operation of the companies represented in SHARE.

The discussion of the assembly consumed much time, and, because of this, considerably shortened the time during which the items of the selected agenda were to be discussed. This resulted in brief discussions of only the following topics:

1. Calling sequence. It was strongly urged that the calling sequence form suggested at the first SHARE meeting be used only as an indication of a possible calling sequence, and that this should not restrict SHARE members in the writing of routines for SHARE.

2. Erasable symbol. It was decided that a special symbol denoting erasable storage should be devised. A subcommittee composed of Irwin Greenwald, John Greenstadt, and Roy Nutt arrived at a decision on this matter. The symbol chosen, COMMON, was accepted by the SHARE Council.

3. Tape restrictions during assembly. A SHARE program deck will not require that certain subroutines be available within the tape library for assembly.

4. Integer scaling. It was decided that integer scaling will be specified as B = 35 rather than a blank scaling field. As an alternative to this, a special data card form may be used for integers. Either of these two methods is acceptable for SHARE purposes.

5. Report from Dr. DeCarlo. Willard Bouricius gave the following report from Dr. DeCarlo:

II. PROCEEDINGS (Continued)

a. The print wheel configuration will be changed to conform with that desired by SHARE as expressed in a letter to Dr. DeCarlo on August 29th. All 704's with the exception of the first will have this change incorporated. The first machine is to be delivered to IBM, New York.

b. Items 2 through 6 on the list of new instructions requested for use with the 704 are presently being engineered at IBM. The opinion was given that there is reason to expect the addition of these new instructions within six months. Number 8 on the same list is considered by IBM to be impossible, while number 7 will require an RPQ from each installation. It was decided that the Secretary shall write a form RPQ requesting number 7 for submittal to IBM by all 704 installations. Number 1 of the same list will shortly be added to the standard list of instructions recognized by the 704.

c. Dr. DeCarlo made it known that half-word logic will be available on the 704 at an additional cost of $500.00 per month.

6. Binary card forms. Additions to the binary card convention were made. One is that column 21, 9 row will be used to indicate a relocatable table of origins. It was also decided that unused columns on the binary card should be left blank or made to be blank when sent for distribution to SHARE members. Slightly changed also was the convention regarding identification of binary cards. The convention is to be the same as decided upon at the first SHARE meeting with the exception that columns 7 and 8 will be used for a sequence code. A further addition is that all SHARE programs will use a standard origin of 0, with erasable storage having a standard origin of $(2000)_8$. This change will allow two or more SHARE programs to use the same erasable storage.

7. Assignment of work. It was noted that of all the programs assigned during the first meeting of SHARE only two were not described in write-ups submitted during the second meeting, and many for which no commitments were made were available in write-up form. Since time did not allow complete discussion of these write-ups and the described routines, it was urged that each representative read the descriptions and communicate by mail to the originator of each regarding any changes or additions thought to be necessary or useful.

8. Location and time of next meeting. There were no strong feelings for when and where the next meeting is to be held. It was decided that the Secretary would compile suggestions for an agenda as these are submitted, and, within a month from date, send to participating installations inquiries as to whether the tentative agenda requires a meeting. It is suggested that the Boston meeting of the Joint Computer Conference would be an opportune time for the next SHARE conference. Each representative is urged to send suggestions for the agenda to the Secretary.

The second meeting of SHARE adjourned on the evening of September 13, 1955.

United Aircraft Corporation

September 9, 1955

# 704 Symbolic Assembly Program UA SAP 1

Roy Nutt

704 instructions to be assembled by this program are written with references expressed as arithmetic combinations of symbols and/or decimal integers. A variable field format is used in which the parts of the instruction are given in the order address, tag and decrement. In addition to instructions, data in decimal, octal or Hollerith (BCD) form may be assembled, and library routines written in the same symbolic form may be conveniently incorporated into the program being assembled.

The following program to compute

$$P_N(x,y) = \sum_{j=0}^{N}\left[\sum_{i=0}^{i+j \leq N} a_{ij}\, x^{i}\right] y^{j}$$

is used as an example. (See page 9).

In order to describe the use of this assembly program, let us consider first a simplified explanation of symbolic assembly operation.

The procedure is divided into two parts;-the first examines the program to be assembled in order to define each symbol used in writing the program. The second part prepares the actual machine language program, punches it in binary form on cards and produces a printed copy of the program in symbolic form together with the corresponding octal machine language program.

During the first part a counter is used to specify the absolute location of each word in the program. Call this location counter L. L is set initially to an integer supplied to the assembly program by the program being assembled, henceforth L is increased by one for each word to be used by the program.

Simultaneously with this counting procedure a table is constructed. Each entry in this table defines a symbol used in the program as being equivalent to some integer. Entries to the table are made in two ways:

1. A symbol appears as the "symbolic location" of a word in the program being assembled and is assigned the value of L.

2. A symbol is defined by a pseudo operation.

UA

It is important to note that the order of the absolute instructions produced by symbolic assembly is determined solely by the order in which the symbolic instructions are read by the assembly program.

During the second part of the assembly process L is computed in exactly the same manner as it was during the first part. In addition all symbols in the symbolic program are replaced by the integer equivalences given in the table formed during the first part, thus producing an absolute program.

Note that this operation requires that each symbol be uniquely defined.

For use in the assembly program the following definitions are made:

Symbol: Any combination of not more than 6 Hollerith characters, none of which is + — * / , $ and at least one of which is non-numeric.

Integer (with respect to instructions): Any decimal integer less than 1000000.

The operation part of each instruction is specified by a standard abbrieviation of not more than 3 Hollerith characters.

A symbolic instruction should be identified by a symbol ("symbolic location") only if it is necessary to refer to this instruction in the program.

The address, tag and decrement parts of symbolic instructions are given in that order. In some cases the decrement, tag or address parts are not necessary, therefore the following combinations are permissible

OP

OP Address

OP Address, Tag

OP Address, Tag, Decrement

For example consider respectively instructions P4 -3, P4 -2, P4 +1 and P4 +2 in the illustration.

Note that the tag, if present, must be separated from the address by a comma and similarly the decrement, if present, must be separated from the Tag by a comma. For the few instructions which require a tag but no address, the address zero should be used, for example:

PDX 0,1

UA16

Similarly where a decrement is required with no tag a zero tag should be used as in

TXL A,0,B

The following card form is used by the assembly

| | |
|---|---|
| 1-8 | not used |
| 9-14 | symbol or blank |
| 15 | blank |
| 16-18 | abbrieviated operation or blank |
| 19 | blank |
| 20-80 | Variable field |

For instructions, expressions defining the address, tag and decrement are punched without blanks from column 20 on. The first blank to the right of column 20 defines the end of the instruction. All punching to the right of such a blank is considered to be a remark and has no effect on the assembly process.

If an instruction requires a symbolic location, the symbol used is punched in columns 9-14.

Arithmetic expressions

Arithmetic expressions in terms of symbols and integers may be used with some pseudo-instructions and to define address, tag or decrement parts of 704 instructions.

The following elementary operations may be used:

addition, indicated by +

subtraction, indicated by -

multiplication, indicated by *

division, indicated by /

No parenthetical expressions may be written.

UAI

$2^{35}$

Integral arithmetic modulo $2^{35}$ is used, hence

1. Multiplication is not commutative with division:

   A*B/C ≠ A/C*B

   except when C is a factor of A.
   Note that A/C*B implies (A/C) *B not A/(C*B).

2. Addition and subtraction are commutative:

   A+B-C = B-C+A

3. Multiplication and division are distributive with respect to each other but not with respect to addition or subtraction:

   A+B*C/D ≠ A*C/D+B*C/D

   Note that A+B*C/D implies A+(B*C/D) and not (A+B)*C/D

If the result of an expression is to be expressed in n binary places, its value is computed modulo $2^n$. If the residue is negative, its 2's complement is the result.

Hence if $v$ is the value of an expression, r is the result used and

$$m = |v| \bmod 2^n$$

$$\text{then} \quad r = \begin{cases} m & v \geq 0 \\ 2^n - m & v < 0 \end{cases}$$

For example the instruction at location P4 +2 in the illustration has a decrement part of -1. Here m=1, v=-1, n=15 so that

$$r=2^{15}-1$$

Consider also the tag part of instruction P4 -1 where

v=J+K=1+4=5

m=5, n=3

so that r=5

Pseudo instructions

Origin specification: ORG

The location counter L is set to the value of the expression appearing in the variable field. Each symbol appearing in the expression must have been previously defined (i.e. appeared in the symbol field, columns 9-14, of some instruction or pseudo-instruction preceding this origin specification).

If no origin specification is given for a program the initial value of L shall be zero.

Origin specification instructions may be used at will.

Synonym: SYN

The symbol appearing in 9-14 is assigned the integer value given by the expression appearing in the variable field. Each symbol used in this expression must be previously defined.

Decimal data: DEC

The decimal data beginning in column 20 is converted to binary and assigned to consecutive locations L, L+1,...

Decimal exponents are indicated by the letter E. The decimal point indicated by . is placed where desired. If either E or . or both appear in a decimal data word the conversion is made to 704 floating binary. If no E or . appears conversion is made to a binary integer. The binary point in storage is considered to be on the right hand end of a 704 word. Successive words of data on a card are separated by commas, and the first blank to the right of column 20 indicates that all punching to the right of this blank is a remark. Signs are indicated by + or - preceding the number or exponent. However it is not necessary to use the + sign.

For example 12.345 may be written as +12.345, 1.2345E1, 1234.5E-2, etc.

Octal data: OCT

The octal data beginning in column 20 is taken in binary integer form, the binary point considered to be on the right hand end of a 704 word, and assigned to consecutive storage locations L, L+1,...

Successive words are separated by commas and the first blank to the right of column 20 indicates that all punching to the right is to be considered a remark.

Hollerith data: BCD

Normally the 10 six character words of Hollerith information from col. 21-80 are read and assigned to locations L, L+1,...,L+9. If, however, less than 10 BCD words are desired, a word count v ($1 \leq v \leq 9$) is punched in column 20, in which case v words are read and assigned to locations L, L+1,..., L+v-1.

Block started by symbol: BSS

The block of storage extending from L to L+N-1, where N is the value of the expression beginning in column 20, is reserved by this operation. Each symbol in the expression for N must have been previously defined.

If a symbol is punched in 9-14, it is assigned the value L, corresponding to the first word of the block reserved.

Finally, L is replaced by L+N.

Block ended by symbol: BES

This operation is exactly the same as BSS, except that the value assigned to any symbol appearing in 9-14 is L+N, corresponding to the location of the first word following the block reserved.

Repeat: REP

Two expressions, the first beginning in column 20 and separated from the second by a comma, define two integers M and N. The block of instructions and/or data following the REP operation in locations L,L+1...L+M-1 is repeated N times, the repeated information being assigned to locations L+M, L+M+1,...., L+M+N-1. It is necessary that M ≤ 110.

Good

Library search: LIB

The library routine identified by the symbol in 9-14 is obtained from a library tape and inserted in the program being assembled. If the library routine requires k words of storage it will occupy locations L,L+1,....,L+k-1. The identification symbol is not entered in the table of symbols, but any symbols appearing in the library routine are entered and properly defined.

End of program: END

This operation must be the last read by the assembly program. The value of the expression beginning in column 20 is punched as the transfer address in a 704 binary control card.

Heading: HED

It is often convenient to combine several programs into one program. Two difficulties immediately arise. First, the symbolic references to data common to the several programs may differ in the individual programs. This can be easily corrected by the use of synonyms which equate the proper symbols.

Second, it may be that two or more of the individual programs use the same symbols for references which should be unique. In order to restore uniqueness, it is necessary to change the symbols in each program in some way. The heading operation accomplishes this result in the following manner.

The heading card supplies to the assembly program a single character (punched in column 9 of the HED card). Each symbol in the program following the HED operation is prefixed by this character except when a special indication to cancel the prefixing operation is given. A new heading operation will replace the prefix character. Thus three programs having non-unique symbols may be combined by giving the heading operation with a unique character before each program.

It is, however, sometimes necessary to make cross-references between the individual programs. To accomplish this, such references must be rewritten in the following way. Let H be a heading character and K be the symbol to which reference is to be made in the block headed by H. To refer to K from a part of the program not headed by H write

H$K

The special character $ indicates to the assembly program that K is to be prefixed by H instead of the prefix given by the last heading control.

It is important to note that if use is to be made of the Heading feature, all symbols used through out the program must be restricted to five or fewer characters.

### Operational features

As an aid to the programmer this assembly program gives some indications of erroneously prepared programs.

If a symbol used in the program is not defined, an asterisk is printed to the left of the symbolic instruction referring to that symbol. The value zero is used in expressions using the symbol.

A list of duplicated symbols is printed prior to the printing of the program. This list gives the symbol duplicated and the integer values assigned to it.

Other convenient features are:

Printing may be suppressed.

Single or double spacing is optional.

Assembly may be made from either a BCD tape, or from cards.

### Machine components required:

In any case the on line card reader, core storage and the on line card punch will be required. In addition if

1. No library references are made then assembly may be made with

   a. one tape (requiring one reading of the symbolic cards either from the on line or off line card reader) or

b. no tapes (requiring two readings of the symbolic cards on the on line card reader).

2. If library references are used two additional tapes are required. One of these is the library tape, the other is used to construct a sub-library written in the order required by the assembly. Hence either three or two Tapes will be necessary (see cases 1a and 1b.)

Reassembly features

Additions to a program which has been assembled are easily accomplished if the table of symbols which was punched during the initial assembly process has been saved. It is then necessary only to reload this table and assemble the new parts of the program. The original program need not be reloaded.

Furthermore any change to the original program which does not involve relocation of any part of the program, or any reassignment of symbols, may be made by assembly of only those parts of the program which are to be changed.

SHARE

# PROCEEDINGS OF THE THIRD MEETING OF SHARE
BOSTON, NOVEMBER 10-11, 1955

## TABLE OF CONTENTS

# I
# PROCEEDINGS

## MINUTES OF THIRD SHARE MEETING

The third meeting of SHARE was held in Room 406-8 of the Statler Hotel, Boston, on Thursday and Friday, November 10th and 11th, 1955. Twenty-two members of SHARE were represented:

| | |
|---|---|
| BA | Boeing Airplane Company |
| CF | Convair (Ft. Worth) |
| CS | Convair (San Diego) |
| CW | Curtiss-Wright Corporation |
| GE | General Electric AGT Division (Evendale) |
| GA | General Electric AGT Division (Lynn) |
| GT | General Electric MST Division (Lynn) |
| GS | General Electric LSTG Department (Schenectady) |
| GM | General Motors Corporation |
| PK | International Business Machines Corporation Computing Bureau (Poughkeepsie) |
| NY | International Business Machines Corporation Data Processing Center (New York) |
| CL | Lockheed Aircraft Corporation (California Division) |
| GL | Lockheed Aircraft Corporation (Georgia Division) |
| ML | Lockheed Aircraft Corporation (Missile Systems Division) |
| LA | Los Alamos Scientific Laboratory |
| NS | National Security Agency |
| NC | North American Aviation (Columbus) |
| NA | North American Aviation (Los Angeles) |
| RA | Redstone Arsenal |
| RS | The Rand Corporation (Santa Monica) |
| UA | United Aircraft Corporation |
| WH | Westinghouse Electric Corporation |

There were four absentees:

| | |
|---|---|
| CR | California Research Corporation |
| HA | Hughes Aircraft Corporation |
| LC | Livermore Radiation Laboratory |
| RL | The Rand Corporation (Systems Training Project) |

Chairman Strong presided. Due to the unavoidable absence of Secretary Jones, the Chairman appointed Wagner (NA) Acting Secretary. He immediately requested all members to:

1. Keep the Secretary continuously informed of an up-to-date, official name, address, and telephone number for the Secretary's use in distributing SHARE material. (He will keep the IBM distribution office informed.)

2. All material for distribution in SHARE should be submitted as original copy on white bond paper.

3. Supply the Secretary with copies on white bond paper of correspondence between members on subjects that might be of interest to other SHARE members. If the Chairman and Secretary consider these of sufficient general interest, they will be distributed.

The representatives of the member installations of SHARE plus interested nonmembers added up to 49 people, who are listed in Appendix B.

The first item of business was the Agenda. It was agreed to follow the list of items distributed October 26th with the addition of several new ones. The complete Agenda follows:

Agenda, Third Meeting of SHARE

1. Definition of SHARE Membership.
2. Definition of Quorum.
3. Reconsideration of Topics.
4. Processing Proposals by Mail.
5. Material Not in SHARE Language.
6. Definition of Machine Time Charges.
7. Progress on SHARE Assignments.
8. New Assignments.
9. Uniformity of Decimal Symbolic Cards.
10. Forms - Miscellaneous Card, Code Sheet, and Other.
11. Index Register "ADD" Instead of "OR".
12. Number of Characters Used in a Symbol.
13. Compilers.
14. Peripheral Equipment Usage.
15. 704 Experience.
16. Use of SHARE Assembly.
17. Changes to Periquip.
18. Unused Bits.
19. Trapping Mode Console Switch.
20. New 704 Operations and Characteristics.

Following this, Truman Hunter of IBM, who had made the arrangements for the meeting room, said a few short words of welcome.

The record of the discussions and action on the various items of the Agenda, which follows, is not necessarily in chronological order. Some items were considered out of sequence, and some were considered at various intervals throughout the two days of the meeting. For purposes of ready reference, however, everything concerning a particular item is grouped under it in the following record. Similarly, various miscellaneous subjects which were considered throughout the two-day period are grouped together in these Proceedings following Item 20 of the Agenda.

Appendix C, covering the Agenda Items 1 through 5, was then considered.

## Agenda Item 1 - DEFINITION OF SHARE MEMBERSHIP

A preliminary discussion brought out certain desirable changes to the solution proposed in Appendix C. In final form, it was moved and seconded that:

> "A member of SHARE must be an installation which has on hand or on order at least one 704 (or is actively campaigning for the placing of an order in the near future with noticeable positive effect to the extent that programming effort is being devoted to 704 methods). 'Installation' is defined as one or more 704's under the same administrative head who is empowered to select the machine methods to be used."

The discussion which followed brought out the following facts:

1. Existing members who meet this definition are those listed above. In the future, membership may be obtained by direct request to the Secretary of SHARE with an unqualified statement that the installation meets the above requirement. (The list compiled after the meeting is Appendix A of these Proceedings.)

2. It was announced that IBM is maintaining a list for the distribution of SHARE material to nonmembers of SHARE, consisting of its customers and others who request to be placed on it. SHARE encourages this activity, and, furthermore, will be happy to have such interested nonmembers attend all SHARE meetings until the attendance taxes the physical facilities available.

The above motion was unanimously passed with full knowledge of the implications concerned in (1) and (2) above.

## Agenda Item 2 - DEFINITION OF QUORUM

A preliminary discussion brought out certain desirable changes to the solution proposed in Appendix C. In final form, it was moved and seconded that:

> "At least two-thirds of the members of SHARE are necessary to constitute a quorum in order to transact business either at a SHARE meeting or by a mail vote. Unless otherwise specified, a simple majority of the quorum is necessary to pass any motion."

Agenda Item 2 (Continued)

The discussion which followed considered whether the majority should be of the quorum or of those voting (on the presumption that some would wish to abstain.) The argument in favor of permitting abstentions from voting was that, on many proposals, a member may be either uninformed or uninterested, and willing to accept the majority decision of only those who wish to vote, even if only one or two people. The opposing argument was that no SHARE member need ever be uninformed about a topic, but rather owes it as his duty to SHARE to be informed sufficiently to form a judgment as to what is best for the overall organization. It was stated that decisions thus reached will, in the long run, be better. It was also pointed out that it was the duty of the members not to inflict their will on minorities by unusually close votes, and that, when it was evident that this might be the case, a motion to postpone decision for further study could and should be resorted to.

The motion as stated above was passed.

Agenda Item 3 - RECONSIDERATION OF TOPICS

A preliminary discussion brought out certain desirable changes to the solution proposed in Appendix C. In final form, it was moved and seconded that:

> "Reconsideration of a decision made by SHARE may be introduced under the following conditions:
>
> A. When the topic is handled by mail or when advance notice by mail is given that the topic will be treated at a meeting, the approval of more than 50 percent of the quorum is required in order to reopen the subject for reconsideration.
>
> B. When the topic is brought up at a meeting without advance mail notice, 75 percent of the quorum must approve reopening the subject for reconsideration.
>
> C. A motion to change any previous decision of SHARE requires the approval of 75 per cent of the quorum in order to be put into effect."

This motion was passed.

Agenda Item 4 - PROCESSING PROPOSALS BY MAIL

A preliminary discussion brought out certain desirable changes to the solution proposed in Appendix C. In final form, it was moved and seconded that:

> "The business of SHARE may be transacted by mail as follows:
>
> A. The proposal is sent to the Secretary, and must include the name of any other member known to be seriously affected by it.
>
> B. The Secretary immediately sends a copy of this to those other members (but at least one) who are known to be seriously affected by, or to have a deep interest in, the subject.

Agenda Item 4 (Continued)

C. The interested members explore the ramifications of the proposal and mail their comments on it to the Secretary immediately. (Opposition comments may contain a statement that a defined counter-proposal will be submitted if the proposal is rejected.)

D. These comments are made known to the original proposer, and he is given a very brief period to submit a rebuttal if he wishes.

E. The Secretary mails copies of the proposal, the comments, and the rebuttal to all members of SHARE, together with a ballot to be marked "Yes" or "No". (Counter-proposals do not appear on the ballot.) The ballots are marked and returned to the Secretary.

F. Two weeks after the date of distribution of the ballots, the vote closes, and notices of results are sent to the members by the Secretary."

In the discussion which followed, two points were brought out:

1. No provision was made for marking the ballot "Abstain." It was noted that negligence to return the ballot would be potentially a "No" vote if a quorum were not obtained. Members were urged to return every ballot promptly in all cases.

2. In certain exceptional cases where the final balloting is done by mail, after much of the preliminary discussion has taken place at a meeting, the members at the meeting may authorize the ballot to take the form of a choice between alternate proposals.

With full realization of the above implications, the motion was passed.

Agenda Item 5 - MATERIAL NOT IN SHARE LANGUAGE

A preliminary discussion lead to the belief that the solution proposed in Appendix C should be completely reversed. In final form, it was moved and seconded that:

> "No material shall be distributed by SHARE itself (or by IBM under SHARE auspices) which is not in SHARE language."

The discussion brought out that this applied specifically to the following:

1. Write-ups of 704 programs must be in the format distributed with the First SHARE Proceedings. This format is illustrated by most of the write-ups which were in the Second SHARE Proceedings, and by all of those in this Proceedings.

2. Listings of routines, and associated decimal and binary cards, shall be in the format of the official SHARE assembly program (UA SAP1).

Agenda Item 5 (Continued)

3. Octonary card forms shall be those adopted at this meeting under Agenda Item 10.

4. Any other card forms shall be acceptable to a program previously distributed through SHARE.

5. Descriptions of methods, procedures, or write-ups on any other subject shall conform as closely as possible to conventions for terminology officially recommended by SHARE.

The motion was passed.

During the discussion of this motion, Butterworth (GT) suggested that a centrally prepared bibliography of material distributed through SHARE be compiled and distributed periodically. Engel (WH) suggested that each item submitted for distribution through SHARE be accompanied by an index card carrying its classification and a very concise description. It was pointed out that both these suggestions required the establishing of some classification system, which was a difficult task. Engel (WH) and Butterworth (GT) were appointed as a committee to study their suggestions carefully and to submit a recommendation to SHARE at the next SHARE meeting.

Agenda Item 6 - DEFINITION OF MACHINE TIME CHARGES

It was proposed that a standard set of definitions be adopted for recording the use of machine time. Appendix D was distributed as illustrative of this. It was argued that this was desirable for two reasons:

1. The record of machine time is an important factor in determining extra shift rental paid to IBM. It was contended that a standard method of doing this would be advantageous to all concerned. Opposing arguments pointed out that this was an individual matter which may presently be within the authority of each IBM District and/or Regional Manager. It was the consensus of the group that it would be unwise to upset this status quo, and consequently that this type of standardization was not suitable for consideration by SHARE. No formal motion was voted on, and this aspect of the matter was dropped.

2. Several members stated that such a standard set of definitions would be useful for statistical purposes in exchanging information on operating procedures in their installations. The motion was passed that:

> "A committee shall be appointed to study the possibilities of defining machine time charges for statistical records and present their recommendation before the next SHARE meeting."

Amaya(CL) and Engel (WH) were appointed to the committee.

Agenda Item 7 - PROGRESS ON SHARE ASSIGNMENTS

Progress on the assembly program is described under Agenda Item 16 below. Progress on all other utility routines was reported, and is described in Appendix E. Some revised write-ups and detailed progress reports are in Part III.A. of these Proceedings.

The SHARE Glossary Subcommittee reported that they had received no comments on it, and did not plan any revisions or additions until the need became evident.

The problem of getting 704 time for the checkout of subroutines assigned by SHARE was discussed. Bouricius (PK) stated that he would try to find some time on the Poughkeepsie machine for such checkout under the following conditions:

> A complete deck of cards shall be mailed to him. This is expected to be one complete deck and no additional cards of any sort will be added to it. It will be placed in the card reader exactly as it is received. It should contain as part of itself any necessary diagnostic routines. It must be accompanied by very simple, clear, and concise instructions to the operator.

If these requirements are fulfilled, the job will be run as soon as Bouricius (PK) can find a few spare 704 minutes, and the results will immediately be mailed back to the programmer.

Ramshaw (UA) agreed to donate to SHARE some of United Aircraft's free time after Thanksgiving under the same conditions.

A discussion was held concerning the time at which a routine should be considered checked out, so that a complete distribution could be made. It was agreed that cards for a routine should not be distributed until it was checked out on an actual 704. Exceptions to this may be made at the discretion of the programmer only if the routine is completely independent of timing considerations, and has been thoroughly checked out on the 701 by means of a 704 simulator routine. When it is determined that a program is ready for distribution, it should be sent to IBM New York, using the most up-to-date address for IBM SHARE Distribution, which always will appear on the latest list of SHARE members. (Currently this is Appendix A of these Proceedings.) The following should be sent:

1. Write-up.

2. Listing.

3. Complete deck or decks of binary cards.

4. Complete deck or decks of symbolic decimal cards. (The latter should be sent unless it is very large, and if it is the opinion of the issuing member that very few other installations would use the decimal cards. These can request them directly from the issuing member. For example, only a few installations have indicated that they wished to have copies of decimal cards for the SHARE assembly program, UA SAP1.)

Agenda Item 8 - NEW ASSIGNMENTS

No new assignments were made to write subroutines; however, several new ones were described verbally and offers made to distribute them through SHARE. These are included in Appendix E. Write-ups of some of these new routines are contained in Part III.A. of these Proceedings.

Other new assignments are described under various Agenda Items, and are summarized in Part I.C. of these Proceedings.

Agenda Item 9 - UNIFORMITY OF DECIMAL SYMBOLIC CARDS

Ramshaw (UA) reported that the standard SHARE card form for decimal symbolic cards (acceptable to UA SAP1) was in process of being printed. It was now in the mill in IBM, and as soon as he was informed of the appropriate IBM form number, he would make the latter available to all SHARE members so that all could order it, or copy it for use in SHARE distribution.

Agenda Item 10 - FORMS - MISCELLANEOUS CARD, CODE SHEET, AND OTHER

The motion was made and seconded that:

> "The octal card form presented in the Second SHARE Proceedings, Part III, Section 4.h., will be the accepted SHARE standard."

The discussion brought out that this form was predicated on the existing design of the periquip, and would undoubtedly have to be revised if any revisions were made to the latter. This was readily admitted by the proponents of the motion, but they insisted on the desirability of having a standard in the meantime. The motion was passed.

Several code sheet forms were presented. It was the consensus of the body that none of these could be decided upon as standard. Most of those present seemed to agree with the statement that was made that there will be no opposition if any installation took a form that they liked and revised it by putting their own name at the top.

A ready reference card containing condensed information about all the operations and the card, tape, and machine versions of the standard SHARE set of Hollerith characters was presented. It was the consensus of the body that this would be highly useful in its present arrangement. IBM volunteered to look into the possibility of having it reproduced in quantity both in 8-1/2" x 11" size printed on stiff cardboard for desk or wall use, and in a reduced size for pocket use. It is expected that they will distribute these to all their customers.

Agenda Item 11 - INDEX REGISTER "ADD" INSTEAD OF "OR"

After a brief discussion, it was decided to postpone consideration of this and discuss it under the general heading of any new 704 operations or characteristics under Agenda Item 20.

## Agenda Item 12 - NUMBER OF CHARACTERS USED IN A SYMBOL

There was a brie. introductory discussion in which it was made clear that the following is a continuation of informal discussions at the Second SHARE Meeting, and, in effect, a follow-up to SHARE mail proposal No. 1 (by General Electric - ballot mailed 4 October 1955).

It was moved and seconded that:

> "Programs distributed through SHARE shall use five or fewer characters as symbols, except when the programmer wishes to prevent heading (e.g., the symbol "COMMON")."

A spirited discussion followed. The stated objectives of the motion were to enable any distributed routine to be re-assembled with other routines into a final program. Shell (GE) explained various reasons why he doesn't like it, claiming that it doesn't meet the objectives which it pretends to. Heising (NY) asked what would happen when you tried to distribute this final, re-assembled program (with six character symbols), and somebody wanted to re-assemble it again with other routines. Ramshaw (UA) pointed out that it should not be done that way, serially, but rather in parallel; that is to say, by getting together all the individual routines in their original form and then re-assembling all at once.

Shell (GE) noted that he is preparing a program to translate CAGE symbolic cards into SHARE format. When this is completed, any subroutine he distributes will have only one symbol.

It was universally agreed that this was the most useful form which distributed subroutines could take.

The motion was passed.

## Agenda Item 13 - COMPILERS

### FORTRAN

Backus (NY) made a progress report on FORTRAN. Six thousand instructions have been coded in what they hope is final form. He expects that in its first edition FORTRAN will include eight to ten thousand instructions, which will be coded by January 1st. Some debugging will have been accomplished by then, and he estimates that it will be completely checked out some time in February. The minimum components necessary will be one 4096-word core, four tapes, one drum box, and either on-line or off-line output. It will produce symbolic instructions for subsequent assembly in the SHARE format. It is estimated that it will take six minutes to produce one thousand symbolic instructions. The symbols used will be the same ones that were used for variables. Planned for the second edition is the inclusion of formula numbers in the comments. He gave a brief rundown of changes from the latest printed specifications. These are summarized below in Part III.C. He also covered very hurriedly the techniques they are using, the most dramatic of which was the enormous number of tables set up. He ended by paying tribute to United Aircraft, and especially Roy Nutt, for their cooperation and assistance.

Agenda Item 13 (Continued)

PACT

Mock (NA) gave a brief history of PACT I (written for the 701), contrasting it with FORTRAN. He mentioned how helpful its actual production use at North American had been in evaluating this philosophy so that PACT IA (for the 704) could be more useful. He then launched into an elaboration of the write-up in Part III.C. below. He revised the statement therein that the work is now about two weeks behind schedule, and optimistically predicted that coding might be finished by December 15th. Thereafter, 704 availability will dictate completion of checkout. The minimum machine necessary for PACT IA will be a 4096-word core, and 4096 more words of storage, which may be on drum or in additional core storage. Four tapes are needed unless it is loaded from the on-line card reader, in which case only three tapes are used.

He mentioned briefly that the committee for PACT II has begun a few airy speculations. PACT II is currently thought of as a broad philosophy of compiling methods rather than as a program for a particular machine. He concluded by inviting all to participate in any of the PACT projects.

GENERAL COMMENTS

Mock (NA) pointed out that all PACT IA users also plan to use FORTRAN heavily. For future development work, he felt that the cooperative approach was an absolute necessity, not only to speed the programming labor and concentrate the talent, but in order to avoid overlooking any possible classes of problems peculiar to only a few installations.

Shell (GE) noted that compilers may very well eliminate much of the current simple clerical-type coding. However, it will create a need for two types of computer experts: one the high-level creative thinker needed to develop continuously improving techniques, and the other the expert at problem analysis who can exploit these techniques to the fullest for the applications in their company. He also noted that no compiler to date had made liberal use of logical manipulation. He stated it was his belief that we had only scratched the surface in making use of the powerful logical operations now available or on the horizon. He added that, in his opinion, this development will go hand in hand with micro-programming (using a very broad structure of a machine within which it is possible to create the operations needed for each particular application).

Agenda Item 14 - PERIPHERAL EQUIPMENT USAGE

Patrick (GM) presented the General Motors position as contained in their write-up, in Part III.B. below. Butterworth (GT) presented the ideas of General Electric MST Division, Lynn, in the write-up contained in the same Part III.B.

Ramshaw (UA) revealed that they were making excellent progress with their cross-bar switching arrangement for use of the periquip. However, they were only integrating one set of periquip with one machine instead of the double installation using 26 tapes that they had originally planned. The stumbling block had been cable length if the installation was on one floor, coupled with IBM's refusal to consider a two-floor installation.

Agenda Item 14 (Continued)

Mock (NA) elaborated briefly on North American's thoughts, summarized in Part III.B. below. **(Later NA and GM agreed on one program based on the GM method.)**

LIBRARY TAPE

This precipitated a discussion of standardizing on numbers for the input tape, the output tape, and the library tape. Mock (NA) recommended a standard convention concerning which tape number would be used at all times for the library tape. After considerable discussion as to the desirability of this standard, it was moved and seconded that:

> "At all times the standard library tape shall be Tape No. 1."

The motion was passed.

Shell (GE) asked if anyone had realized that if about 5 percent of the computing time on one shift is used for writing output tape, it will keep the off-line printer busy three shifts.

CLOCK

In connection with exploiting periquip, it was brought out that the need for some form of clock connected with the 704 was intensified. Shell (GE) noted that he had information that IBM was working on a clock to be attached to either the punch or the printer so that a time record could be punched or printed under program control. Ramshaw (UA) said it was his information that it was easier to do on the punch, and this was what IBM was working on. Mock (NA) noted that it ought to be on the printer because then you can have access to it with your program, and once you get it into memory, you can do anything with it that you want.

Hunter (NY) pointed out that there were two kinds of clocks possible. One was simply for the purpose of keeping time records of the jobs run. This was relatively easy and was the type that Ramshaw and Shell mentioned. Another kind of clock, however, which had been discussed, was a microsecond clock inside the machine for the purpose of keeping track of minute fragments of programs being executed. He asked for a show of hands as to who wanted what kind. Every installation present said that they wanted to have the timekeeping kind of clock, and, further, that they wanted to be able to have access to it with their programs. When Hunter (NY) asked how many people would be interested in a microsecond clock, only five or six installations expressed interest.

WIRING OF PERIQUIP READER

It was pointed out that the periquip reader was to be wired from columns 1-80. The motion was made and seconded that:

> "SHARE requests that the standard periquip reader be wired to start in column 9, go through column 80, and then come back to columns 1 through 8."

Agenda Item 14 (Continued)

In the discussion which followed, it was pointed out that no matter which way it was wired, it could, with appropriate programming, be usable by anybody. Mock (NA) pointed out that it would make it much easier when using the standard SHARE card forms if it were wired in accordance with the motion. McCool (NS) was asked about his installation (which would be very much affected). He claimed it was no trouble to live with things as they stood. Ryckman (GM) and Greenstadt (NY) were against the motion because as things were now it is easier to be compatible with a 705. Patrick (GM) contended it was much easier to use the standard wiring when mixing engineering and business problems on one tape. Mock (NA) and Nutt (UA) insisted that, since SHARE was primarily interested in engineering computing, the burden of extra programming should be thrown on those who wished to mix in business applications. The motion was tabled.

A subsequent motion was made that:

> "SHARE will use the periquip reader in the form in which it now exists; namely, wired from columns 1 through 80."

It was pointed out that, since we had tabled the previous motion, it would be contradictory to take action on this one. The problem was referred to a committee composed of Nutt (UA) and Keller (GE). They were instructed to study the problem and write up a recommendation for a mail vote as soon as possible. If it seems advisable to this committee, the ballot may be presented in the form of a choice between the two alternates.

Agenda Item 15 - 704 EXPERIENCE

This was preceded by a short discussion on the use of simulators, in which Nutt (UA) and Shell (GE) described the use of their two-frame simulator. Steel (RS) and Mock (NA) described experiences with the one-frame simulator. Strong (NA) mentioned that the two-frame 701 version of UA SAP1 had been rewritten by North American for a single frame, and would be distributed shortly. All agreed that working with a simulator was better than nothing, and, except for the terrible waste of 701 time, usually was a useful method of checkout.

All experience on an actual 704 to date has been on the Poughkeepsie prototype machine. It was emphasized that, being a prototype, this 704 was not representative of production machines. For example, the input-output components in general are old, modified 701 equipment. The machine is being used to give a final test to the production tape units as they are manufactured; consequently, a new set of untested tape units is frequently installed. Moreover, since the first production machine has not been delivered yet, one should expect to find bugs still existing in the basic circuitry, since it is the function of the prototype to uncover these.

Nutt (UA) stated that, apart from the above, he found the machine extremely reliable. It had made no errors while he was using it. Checkout on it was much like the 701 except that your running time on the machine was extremely short, and you had a long period of head-scratching before you were ready to go back with the correction to your program.

Agenda Item 15 (Continued)

Shell (GE) stated unqualifiedly that it was much more subtle to debug and required far more advance planning. He pointed out that main frame reliability was a matter of definition, since he and Nutt have each uncovered one logical bug in it, and he felt that there would be many more of these when different sequences of instructions were executed for the first time.

A subject previously mentioned in passing was reiterated at this time; namely, that IBM is making no effort to guarantee what will happen when a programmer asks the machine to execute an "outlaw" instruction. This is one which does not exist in the Manual of Operation, but which can be obtained (because of the various unused bits) by generating it inside the machine. Such instructions can only be used on a calculated risk basis, because even though they work 100 percent of the time today, the customer engineer may tune up the machine a little differently tomorrow, and they will not work.

Tom Jordan's (LA) comment on 704 experience was that his group had coded extensively from flow diagrams and checked their codes very carefully. This cut down on the quantity of errors, but the ones that were still in the code were much harder to find. He felt that he needed an extensive set of selective tracing routines. Greenstadt (NY) commented that he found core dumps adequate.

Bouricius (PK) noted that there were two schools of thought, each with highly competent proponents, one saying some form of selective tracing was most efficient, and the other saying that the core dump was all that was necessary. Bouricius stated that an accurate record of difficulties had failed to reveal any failures in core storage that were not explained by goofs in maintenance. He noted that there has been a difficulty when changing the tape mode too quickly from binary to BCD or vice versa. The synchronizer switched immediately, even though a COPY was still being executed. Use of a delay instruction will avoid this. However, it is believed that the latter will not be necessary on production 704's.

Strong (NA) asked if a complete set of periquip would be available in New York for use during the free time allotted to customers. The answer was yes.

Agenda Item 16 - USE OF SHARE ASSEMBLY

Nutt (UA) gave a progress report on the SHARE assembly program (UA SAP1). He stated that 704 availability had delayed complete checkout. However, it was almost entirely checked out. There were two trivial bugs still to be removed, and he was confident that they would be corrected on the next machine run. However, there was one major bug still outstanding in connection with relocatable binary cards. He was going to Poughkeepsie the next week, and was confident that UA SAP1 would be completely checked out by Friday, November 18. He thanked Bouricius (PK) for his outstanding cooperation in getting machine time. He noted that as soon as the checkout was complete, a small supplement to the existing write-up would be issued, but said this would contain no extraordinary new information. Nothing significant developed in the way of questions and answers.

Agenda Item 17 - CHANGES TO PERIQUIP

Ramshaw (UA) in'roduced this subject by describing the fact that United Aircraft had enormous numbers of programs that were used only occasionally. They felt it was quite impractical to save these on tape. They wished to store them on cards. Ideally, this would be similar to the 701 operation where the storage was on binary cards. However, United Aircraft wished to use the periquip for output and input of these cards; i.e., output once from core to tape to off-line punch to card file, and then input whenever necessary from card file to off-line reader to tape to core storage. With present periquip design, this can be done using easily read octal cards, five instructions per card (see Agenda Item 10). It can also be done, 10 instructions per card (very hard to read), using a much clumsier "quadroctal" card form. He proposed a change to all the periquip (reader, punch, and printer), which will permit handling twelve easily read instructions per card, as follows:

1. It must work with all current Hollerith characters plus the now forbidden combinations:

   8, 7, zone
   8, 6, zone
   8, 5, zone

2. It must change the parity check and the method of handling zero to be consistent with the main frame; thus BCD characters on tape will have a one-to-one correspondence with core storage.

He emphasized that he was not proposing any alteration in the 704 itself or in the synchronizer. This change would be exclusively in the periquip. He noted things which could not be done if this revision were made:

1. A revised set of periquip could not be used for a 702 or 705 installation which might be at the same location as a 701 or 704 installation. (Here 701 refers to one equipped with 727 tapes.)

2. The tapes which would be prepared for or by the revised periquip would have to be referred to as BCD' tapes, and, obviously, could not be obtained from or sent to a 702 or 705. Note, however, that the 701 or 704 can still produce or receive standard BCD tapes, which are completely interchangeable with any others.)

Strong (NA) immediately noted that what Ramshaw (UA) was proposing was only a compromise, and pointed out that as soon as the periquip was announced, North American Aviation had urgently requested IBM to revise it so as to work with the complete card image. Ramshaw admitted that this would be the best solution but felt that it would be too much to hope for, whereas he felt that his proposal could be obtained rather easily. It was announced that IBM was now investigating the card image deal on a column-by-column basis. Everyone agreed that the probability was extremely small of getting any type of revision installed with a switch so that the same piece of periquip could be used in either the original or the revised mode, because it would be very much harder and more expensive to do. It was agreed that SHARE should not make a decision as to what they would request IBM to do until more information was obtained as to the relative difficulty of the various possibilities.

Agenda Item 17 (Continued)

Nutt (UA) was appointed a committee of one to be the contact man on IBM activity, and he was empowered to submit to SHARE a proposal giving the pros and cons of all reasonable possibilities, and to make a recommendation. This will be done as soon as sufficient information can be obtained from IBM.

Agenda Item 18 - UNUSED BITS

The unused bits (12-17) in the decrement field of Type B instructions have precipitated an "Oklahoma land rush" among the users who wish to stake out a claim on them. The subject is bound up with the one covered under Agenda Item 20, "New Operations." Under that heading, the difficulties may be explained as follows: If the bits are used merely as labels, with the present decoding circuitry, then the number of new operations possible is relatively small because of the relatively small number of combinations possible. If, on the other hand, new decoding circuitry is considered when new operations are to be introduced, the number of bit combinations available for new operations is very much larger. To date, it seems that the simpler and more restricted method is being followed by IBM, and this divided the members into two camps - those who wished to set up a convention for the present situation, and those who wished to press for a change to the more unrestricted method. Here, under Agenda Item 18, however, the only situation considered is the present one, where these bits are used merely as labels.

Jordan (LA) stated his belief that efficient debugging would require several sophisticated selective tracing programs in which these bits could very profitably be put to use. Handy (ML) noted that his SHARE assignment of a double precision abstraction has to use these bits. Amaya (CL) pointed out that the recent formal announcement of half-word arithmetic for the 704 included the use of bits 16 and 17. Judd (NY) warned that any new instructions currently under consideration (including those previously requested by SHARE) may very well use them. Wagner (NA) reminded everyone that when this subject was discussed previously at the first SHARE meeting, IBM warned that those bits were not in the public domain, and could be used by programmers only at their own peril.

It was the consensus of the group, however, that in spite of this, some usefulness might be squeezed out of them, and consequently a convention for their use was desirable. It was moved and seconded that:

> "SHARE requests IBM to give the order of least probability of use of bits 12-17 in the decrement field of Type B instructions. If IBM has no firm opinions on the subject, SHARE requests that IBM use them from right to left (beginning with 17 and 16, which they have already used), and that SHARE programmers will use them (at their peril) from left to right (beginning with 12)."

The motion was carried.

Agenda Item 19 - TRAPPING MODE CONSOLE SWITCH

Ramshaw (UA) related that difficulties in checkout by manipulating the console have demonstrated the need for a button on it whose function will be to change the trapping mode trigger. He pointed out that this would enable your program to have access to the instruction counter, and thus get a printout of it.

An added suggestion was made that it be a two-position toggle switch, one of which enters trapping mode and one of which inhibits trapping. Many people commented that this seemed to be retrogressing in view of the fact that everyone paid at least lip service to the philosophy that all operation, including checkout, should be completely automatic. Ideally the console should be eliminated entirely, unless the customer engineers needed it. Shell (GE) very firmly stated that he does not want any button that operators can get their hands on, because they are probably going to push it the right way at the wrong time. No formal yes or no vote was taken. The Chairman asked for a show of hands as to who was interested, and nobody except Ramshaw (UA) indicated interest. The subject was tabled, and Ramshaw was invited, if he wished to push it further, to submit a complete write-up for decision by mail. (Later on, Cantrell (GS) evidenced interest.)

Agenda Item 20 - NEW 704 OPERATIONS AND CHARACTERISTICS

This subject, which started out as a single specific item on the Agenda (Item 11 - Index Register "ADD" Instead of "OR") soon developed into a major discussion of philosophy and policies in the future development of the logical structure of the 704. Discussions under almost every item on the agenda contained overtones of these future possibilities, and the topics which are recorded below were interspersed throughout both days.

These discussions concerned three categories of changes:

1. Those already formally requested by SHARE.

2. Proposals for specific new changes presented for consideration to this meeting.

3. Future 704 changes.

These will be considered in that order below.

CHANGES ALREADY REQUESTED BY SHARE

There were a total of eight of these requested, numbered 1 to 8 in the form letter which each installation was expected to send to IBM, and a copy of which was sent to each member by the Secretary 30 October 1955. It was brought out that events which had taken place since then had divided these into four groups: No. 1, Nos. 2-6, No. 7, and No. 8. In spite of our correspondence on this subject, present procedures make it imperative that the local IBM branch office at each installation will have filled out and forwarded to IBM an RPQ in connection with each of these groups. Each installation, therefore, must immediately check with their local branch office of IBM to see that three RPQ's have been filled out and submitted for their installation:

Agenda Item 20 (Continued)

A. This RPQ should request the proposed new operation No. 1, Copy and Add Logical Word.

B. This RPQ should request the group of new operations 2 through 6:

2. Exclusive OR.

3. Store Index in Address.

4. Place Index in Address.

5. Logical Right Shift.

6. Store Tag.

(1) C. This RPQ should request the proposed new operation No. 7, Backspace File on Tape (including the ability to skip one instruction when Backspace File is given during a beginning-of-tape condition). In addition, this RPQ should request that the 704 skip one instruction when a Backspace Record (BST) is given and a beginning-of-file condition is encountered.

It has been determined that proposed additional operation No. 8, Read Tape Backward, is impossible to achieve, and thus each installation should also check that no RPQ has been submitted for this. If one has been submitted, have it withdrawn.

In connection with this somewhat clumsy procedure, there was a lively discussion as to the possibility of streamlining it, since official requests by SHARE, representing its members, might be handled in one block. Shell (GE) suggested that any such procedure should be carefully scrutinized so that no violation of the anti-trust laws was involved. A committee was appointed consisting of Ramshaw (UA) and Greenstadt (NY) to investigate the possibility of streamlining this multiple RPQ procedure, whereby over 20 different branch offices do identically the same thing for each member installation of SHARE.

NEW OPERATIONS AND CHARACTERISTICS PROPOSED AT THIS MEETING

Index Register ADD Instead of OR (Agenda Item 11)

Steel (RS) had originally proposed under Agenda Item 11 a revision to the characteristics of the 704 when two index registers are used simultaneously. Rand wished this to produce the effect of an addition instead of the current logical OR. This was the subject of some preliminary correspondence during the last few months. Backus (NY) pointed out that this might cause extra machine cycles when it is used, or even possibly at any time when the index registers are used. Bouricius (PK) passed on the rumor that the engineers claimed that it would take a year to get it, and there is not space enough in the machine. Keller (GT) and Steel (RS) agreed that it could be a very valuable characteristic, and thought it should be investigated.

>>>>><<<<<

(1) Following the meeting, NA received a copy of IBM letter to Santa Monica Office from WHQ dated 11-14-55 on this subject, which is being distributed to SHARE.

Agenda Item 20 (Continued)

At this point, there was considerable discussion which is summarized below under "General Philosophy for Future 704 Changes."

The group decided to consider this proposed characteristic of the machine in conjunction with many others that would be investigated during the next few months, and that The Rand Corporation should take the responsibility for circulating by mail a complete set of arguments demonstrating its value.

Load Index with Own Address

Load Index with Complement of Own Address

Ramshaw (UA) moved and it was seconded that:

"SHARE requests IBM to add to the machine two new operations:

A. Load Own Address into Index Register.

B. Load Complement of Own Address into Index Register."

Shell (GE) stated that he liked the new instructions very much, but agreed with Cantrell (GS), Heising (NY), Porter (BA), and others who felt that a firm decision should not be made until after further study.

The motion was tabled, and the group decided that this would be considered in conjunction with proposals for other new operations during the next few months, and that United Aircraft take the responsibility for circulating by mail complete arguments demonstrating the value of these two proposed operations.

Sense Copy Check

Shell (GE) suggested that it would be desirable to remove the Copy Check Stop feature from the machine, and substitute for it a sense type instruction to test the Copy Check Trigger. It was agreed that this should be considered in conjunction with all new proposed operations during the next few months, and that General Electric AGT, Evendale, should take the responsibility for circulating complete arguments demonstrating its value.

FUTURE 704 CHANGES

There was considerable discussion as to the value of the SHARE organization in giving complete, careful consideration to requests for changes to the 704. It was pointed out that this value could quickly disappear if ill-considered requests were made of IBM which asked for a big engineering effort, and were followed by a loss of interest by SHARE members. Representatives of most installations present made speeches agreeing that they would be very careful not to do this. In return, IBM was urged to supply quick feedback so that those requesting would understand as early as possible the engineering and production difficulties and costs involved. It was further noted that IBM ought to remove any fear among the members of SHARE that a stock reply might be made that a proposal is extremely difficult, even before it has been closely scrutinized by competent people.

Agenda Item 20 (Continued)

All present agreed that some soul-searching was necessary on both sides, and pledged themselves to the fullest cooperation.

Hunter (NY) announced that IBM's research budget for 1956 included a thorough investigation of the addition of a new component to the 704. This would be a box, supplemental to the main frame, which will give more space for added circuitry. Some very vague information about it may be available by the end of December, and firm specifications should be ready in March. Among other things to fill it with, they plan to investigate approximately 50 new operations. It was agreed, therefore, that the SHARE organization, individually and collectively, should give careful thought to such new operations and characteristics of the machine as they consider most important, and advise IBM accordingly.

One major decision in this regard must be whether this box would result in every 704 being slightly different from every other one, or whether the SHARE organization wished to keep the machines as standard as possible. Shell (GE) had previously indicated that his present inclinations were along the line of rigid standardization. Amaya (CL) said that Lockheed was already almost committed to having a machine with half-word arithmetic. Bouricius (PK) noted the possibility that, if one does not insist on standardization, the same bits in the instructions can be used for different things in different machines. This precipitated another violent discussion of the philosophy of usage of the presently unused bits referred to under Agenda Item 18 above.

The matter was concluded as follows:

1. Additional changes would be given serious consideration during the next few months with a view towards referring final decisions to the next meeting of SHARE.

2. In the interim, all SHARE members were requested to consider very carefully any new operations or characteristics and distribute them by mail for comments.

3. Bouricius (PK), Hunter (NY), and Heising (NY) of IBM were requested to keep SHARE informed as early as possible of developments along this line. SHARE wants to know the new operations under consideration, those that will be more difficult and costly to obtain than others, and, in particular, whether any information can be obtained on the basic question of decoding the operations (i.e., whether the possible number of new operations on any one machine will be extremely limited or not).

MISCELLANEOUS ITEMS NOT ON THE AGENDA

Proposed Input-Output Buffer

Hunter (NY) announced that the proposed new box would very likely contain an input-output buffer which would permit direct communication from the tapes to core storage with a minimum interruption of computing. It is expected that this would probably take zero time during the execution of multiply or divide instructions, Miscellaneous Items (Continued)

and in other cases ·.ould periodically interrupt execution of instructions for 12 microseconds as each word is transmitted. He said that this would not render impossible the old-fashioned way of transmitting information by the COPY operation.

Delivery Schedules

Hunter (NY) denied a rumor about an additional production delay, and stated that the first four machines would be delivered in December. The first production machine (destined for IBM New York) is now under test operation at Poughkeepsie, and is working extremely well.

Proposal on Format for 704 Assembly

Greenstadt (NY) distributed Appendix G. He moved and it was seconded that:

> "The subject of the format of the official SHARE assembly program be reopened, and that the proposal contained in Appendix G be referred to a committee for evaluation."

In the discussion which followed, it was brought out that:

1. Backus (NY) stated that there was no question but that the FORTRAN routine itself will be distributed in SHARE language, and that FORTRAN would use SHARE language. Consequently, the symbolic instructions produced by FORTRAN will be accepted for assembly by UA SAP1. However, he fears that the routines produced may have too many symbols, even though he thinks that he can remove those symbols which are not referred to. Ramshaw (UA) argued that, at worst, this certainly could be done with one more tape pass, and Backus (NY) agreed and said it might even be done more easily. However, he still wanted to have the ability to assemble FORTRAN output with NY AP1 in those cases where the number of symbolic instructions was so large that it would be very inefficient to use UA SAP1.

2. It was the consensus of most of those who commented that the proposed changes to the SHARE format were not ones which it was necessary for every installation to make. In fact, it was only necessary at the IBM New York Computing Center to the extent that they continued to use NY AP1, and elsewhere if UA SAP1 becomes too clumsy to be used with FORTRAN. Consequently, it was felt by the majority of those commenting that these changes were entirely permissible on an internal basis within IBM New York Computing Center. They reasoned, therefore, that the motion was irrelevant to SHARE as a whole. The Chairman ruled that this was in fact true - the motion was irrelevant - and the matter was dropped. Subsequently Greenstadt (NY) agreed completely with this disposition of the matter.

Miscellaneous Items (Continued)

Proposed Standard Printer Board

Mock (NA) moved and it was seconded that:

> "SHARE shall select a standard configuration for the printer boards and designate it as such."

It was immediately agreed that this seemed to be desirable; it was referred to a committee consisting of Shell (GE), Nutt (UA), and Greenstadt (NY), who are requested to report as soon as possible, in order that the subject may be disposed of by mail.

Sequence Numbers in Distributed Decks

Steel (RS) moved and it was seconded that:

> "All distributed decks which carry sequence numbers must use the following convention: Sequence numbers in self-loading decks shall start with zero, and in all others shall start with one."

The motion carried.

Proposed Committee on Mathematical Analysis

Sangren (CW) proposed that a committee of SHARE be established to investigate current progress on mathematical analysis methods applicable to computing machines, and correlate and disseminate this information through SHARE. He was requested by the Chairman to submit such a proposal by mail, outlining completely the functions of such a committee.

Characteristics of Future Machines

Some members felt that SHARE should take on as one of its responsibilities the study of the desired characteristics of new machines, in order to give IBM unified authoritative guidance from its customers. Some members felt that this was beyond the scope of the SHARE organization. It was pointed out that this would have to be a continuous interchange of information, since no one likes to ask for "blue sky" capabilities which may be impossible to realize except at exorbitant cost in time and money. Consequently, some indications from IBM as to what they could do would be valuable.

Further discussion along these lines brought out that this involved a very delicate situation concerning information which IBM has every right to consider proprietary until they are ready to make a formal announcement, and that it was asking entirely too much of IBM to disclose these things prematurely.

Nevertheless, the general consensus was that SHARE should make some attempt to provide IBM with well-considered information as to the desires of its members. It was mentioned that the PACT group was also interested in doing this from the point of view of the machine characteristics needed for future compiling techniques. It was agreed that, if time permitted, this general subject would be placed on the Agenda for the next meeting.

CONCLUSION

Chairman Strong (NA), on behalf of the organization, and Hunter (NY) on behalf of IBM, thanked all member and nonmember participants for their attendance and cooperation during the meetings. Hunter (NY) offered on behalf of IBM to provide facilities for the next meeting.

The group agreed that, barring contingencies, the next meeting would be held in conjunction with the Western Joint Computer Conference in San Francisco during the second week of February, 1956. Since the conference is Tuesday, Wednesday, and Thursday, February 7, 8, and 9, it is probable that the SHARE meetings will be held Monday, February 6, and Friday, February 10.

The meeting adjourned at 4:15 p.m., Friday, November 11.

A summary of all decisions reached follows in Part I.B. A summary of new assignments is contained in Part I.C.

## PROCEEDINGS PART I.B.

## SUMMARY OF DECISIONS REACHED

| Subject | Description | Disposition | Agenda Item | Page |
|---|---|---|---|---|
| Assembly - Format for 704 | Subject of format of official SHARE assembly shall be reopened | Formal motion - ruled irrelevant and withdrawn | Misc. | I-A-22 |
| Bibliography of SHARE Materials | Committee appointed to study | No formal motion - referred to committee | 5 | I-A-7 |
| Ballots - Form of | See Mail - Processing Proposals by | | | |
| Bits - Unused (12-17) | SHARE requests IBM to use unused bits in the decrement field in a definite order | Formal motion - passed | 18 | I-A-16 |
| Checkout | Definition established of when a routine shall be considered checked out, and of what should be distributed | No formal motion - agreed on definition | 7 | I-A-8 |
| Clock | Timekeeping clock is most desired, rather than the microsecond type | No formal motion - show of hands indicated timekeeping type accessible to program is most wanted | 14 | I-A-12 |
| Distribution | | | | |
| Checked-Out Routines | What should be sent to IBM | No formal motion - list decided upon | 7 | I-A-8 |
| Nonmember | IBM is maintaining a nonmember distribution list for SHARE material | No formal motion - SHARE encourages such distribution | 1 | I-A-4 |

SUMMARY OF DECISIONS REACHED (Continued)

| Subject | Description | Disposition | Agenda Item | Page |
|---|---|---|---|---|
| Format | | | | |
| For 704 Assembly | See Assembly - Format for 704 | | | |
| For Distribution of Material | See Language - Material Not in SHARE | | | |
| Forms | | | | |
| Code Sheet | Form to be decided upon by each installation | No formal motion - agreed each installation could select and modify any they wished | 10 | I-A-9 |
| Decimal Symbolic Cards | The standard form in preparation by UA is desirable for distribution | No formal motion - UA's form will be available to all | 9 | I-A-9 |
| Octal Cards | Octal card form presented in Second SHARE Proceedings will be the accepted SHARE standard | Formal motion - passed | 10 | I-A-9 |
| Index Register ADD Instead of OR | Proposed revision to 704 characteristics | No formal motion - request for further arguments | 20 | I-A-17 |
| Installation | Defined - See Membership | | | |
| Language - Material not in SHARE | No material shall be distributed by SHARE (or by IBM under SHARE auspices) which is not in SHARE language | Formal motion - passed | 6 | I-A-7 |

SUMMARY OF DECISIONS REACHED (Continued)

| Subject | Description | Disposition | Agenda Item | Page |
|---|---|---|---|---|
| Library Tape | The standard library tape shall be Tape No. 1 | Formal motion - passed | 14 | I-A-12 |
| Mail - Processing Proposals by | Procedures established to transact SHARE business by mail | Formal motion - passed | 4 | I-A-5 |
| Mathematical Analysis Committee - Proposal for | Suggestion that such committee be set up | No formal motion - request for mail proposal | Misc. | I-A-22 |
| Membership - Definition of | Member shall be an installation which has on hand or on order at least one 704; "installation" defined | Formal motion - passed | 1 | I-A-4 |
| Operations - New | Committee on information from IBM appointed | No formal motion - committee set up | 20 | I-A-20 |
| Load Own Address into Index Register, and Load Complement of Own Address into Index Register | Add the two new operations stated | Formal motion - tabled for future consideration | 20 | I-A-19 |
| Sense Copy Check | Substitute an operation "Sense Copy Check" for the Copy Check Stop | No formal motion - request for further information | 20 | I-A-19 |
| Periquip | | | | |
| Changes to | Revise periquip to handle non-Hollerith code | No formal motion - referred to committee | 17 | I-A-15 |
| Reader Wiring | See Reader - Present Periquip | | | |

SUMMARY OF DECISIONS REACHED (Continued)

| Subject | Description | Disposition | Agenda Item | Page |
|---|---|---|---|---|
| Printer Boards - Standard | SHARE shall select a standard configuration for the printer boards and designate it as such | Formal motion - referred to committee | Misc. | I-A-22 |
| Proposals | See Mail - Processing Proposals by | | | |
| Quorum - Definition of | Quorum shall consist of two-thirds of members; majority of quorum necessary to pass any motion | Formal motion - passed | 2 | I-A-4 |
| Reader - Present Periquip | Two motions were made:<br>(1) Periquip Reader shall be wired from columns 9-80, and then 1-8<br>(2) Periquip Reader wired from columns 1-80 shall be the SHARE standard | Formal motions - referred to committee | 14 | I-A-12 |
| Reconsideration of Topics | Established percent of quorum necessary to reconsider previous decisions and overrule them | Formal motion - passed | 3 | I-A-5 |
| RPQ's | | | | |
| Streamlining of Procedures for | SHARE should be able to make requests of IBM in the name of its members | No formal motion - referred to committee | 20 | I-A-18 |
| Existing SHARE Requests | Existing SHARE requests for changes to the 704 require three RPQ's from each installation | No formal motion - Chairman instructed each installation to check their local IBM office | 20 | I-A-17 |
| Sequence numbers | Convention for sequence numbers in decks to be distributed | Formal motion - passed | Misc. | I-A-22 |
| Switch - Trapping Mode Console | Need for ETM-LTM switch | No formal motion - request for further information | 19 | I-A-17 |

SUMMARY OF DECISIONS REACHED (Continued)

| Subject | Description | Disposition | Agenda Item | Page |
|---|---|---|---|---|
| Symbols | Programs distributed through SHARE shall use five or fewer characters as symbols, except when programmer wishes to prevent heading | Formal motion - passed | 12 | I-A-10 |
| Tape, Library | See Library Tape | | | |
| Time Chages - Definition of Machine | (1) Standardization of machine charges for payment of shift rental | No formal motion - agreed to drop | 6 | I-A-7 |
| | (2) Committee appointed to study standardization of machine charges for exchange of statistical records | Formal motion - passed | 6 | I-A-7 |
| Vote | | | | |
| By Mail | See Mail - Processing Proposals by | | | |
| Majority | See Quorum | | | |

## PROCEEDINGS PART I.C.

## NEW COMMITTEES APPOINTED

| Committee Title | Members | Job | Due Date | Agenda Item | Page |
|---|---|---|---|---|---|
| Bibliography - Index | Engel (WH)<br>Butterworth (GT) | Establish classification system for SHARE material and study methods of distributing bibliography or index | Mail proposal - just before the next meeting | 5 | I-A-7 |
| Future 704 Changes | Bouricius (PK)<br>Hunter (NY)<br>Heising (NY) | Keep SHARE informed of development of any new 704 characteristics | Continuing | 20 | I-A-20 |
| Machine Time Charges | Amaya (CL)<br>Engel (WH) | Study the possibilities of defining machine time charges for statistical records and present recommendation | Mail proposal - just before the next meeting | 6 | I-A-7 |
| Periquip Changes | Nutt (UA) | Report on all reasonable possibilities for revisions to periquip and make recommendation | Mail proposal - as soon as possible | 17 | I-A-15 |
| Periquip Reader Wiring | Nutt (UA)<br>Keller (GE) | Recommend standard wiring of existing periquip reader | Mail proposal - as soon as possible | 14 | I-A-13 |
| Printer Boards | Shell (GE)<br>Nutt (UA)<br>Greenstadt (NY) | Recommend a standard configuration for printer boards | Mail proposal - as soon as possible | Misc. | I-A-22 |
| RPQ Procedure | Ramshaw (UA)<br>Greenstadt (NY) | Investigate possibility of streamlining procedure for multiple RPQ | Mail proposal - as soon as possible | 20 | I-A-18 |

## PROCEEDINGS PART I.C. (Continued)

## NEW ASSIGNMENTS TO INSTALLATIONS

| Installation (Code) | Job | Due Date | Agenda Item | Page |
|---|---|---|---|---|
| Curtiss-Wright (CW) | Submit proposal for a Standing Committee on Mathematical Analysis (Sangren) | Mail proposal - when convenient | Misc. | I-A-22 |
| General Electric AGT Dept., Evendale (GE) | Study further the new operation "Sense Copy Check" and circulate arguments by mail (Shell) | Mail proposal - just before the next meeting | 20 | I-A-19 |
| IBM New York (NY) | Investigate possibility of distributing, in two sizes, "List of Operations" cards for ready reference (Greenstadt) | Authorized to be done as soon as possible | 10 | I-A-9 |
| The Rand Corporation (RS) | Study further the new characteristic "Index Register ADD Instead of OR" and circulate arguments by mail (Steele) | Mail proposal - just before the next meeting | 20 | I-A-18 |
| United Aircraft (UA) | (1) Study further the new operations "Load Index with Own Address" and "Load Index with Complement of Own Address" and circulate arguments by mail (Ramshaw) | Mail proposal - just before the next meeting | 20 | I-A-19 |
| | (2) Study addition of Trapping Mode Console Switch and circulate arguments by mail (Ramshaw) | Mail proposal - when convenient | 19 | I-A-17 |

## PROCEEDINGS PART I.C. (Continued)

### SUMMARY OF PEOPLE RESPONSIBLE FOR NEW ASSIGNMENTS

| People | Assignment | Agenda Item | Page |
|---|---|---|---|
| Amaya (CL) | Machine Time Charges Committee | 6 | I-A-7 |
| Bouricius (PK) | Future 704 Changes Committee | 20 | I-A-20 |
| Butterworth (GT) | Bibliography - Index Committee | 5 | I-A-7 |
| Engel (WH) | (1) Machine Time Charges Committee | 6 | I-A-7 |
| | (2) Bibliography - Index Committee | 5 | I-A-7 |
| Greenstadt (NY) | (1) IBM (NY) will investigate distribution of "List of Operations" cards in two sizes. | 10 | I-A-9 |
| | (2) Printer Boards Committee | Misc. | I-A-22 |
| | (3) RPQ Procedure Committee | 20 | I-A-18 |
| Heising (NY) | Future 704 Changes Committee | 20 | I-A-20 |
| Hunter (NY) | Future 704 Changes Committee | 20 | I-A-20 |
| Keller (GE) | Existing Periquip Reader Wiring Committee | 14 | I-A-12 |
| Nutt (UA) | (1) Existing Periquip Reader Wiring Committee | 14 | I-A-13 |
| | (2) Periquip Changes Committee | 17 | I-A-15 |
| | (3) Printer Boards Committee | Misc. | I-A-22 |
| Ramshaw (UA) | (1) United Aircraft will study further the new operations "Load Index with Own Address" and "Load Index with Complement of Own Address" and circulate arguments by mail. | 20 | I-A-19 |
| | (2) United Aircraft will study further the addition of a Trapping Mode Console Switch and circulate arguments by mail. | 19 | I-A-17 |
| | (3) RPQ Procedure Committee | 20 | I-A-18 |

SUMMARY OF PEOPLE RESPONSIBLE FOR NEW ASSIGNMENTS (Continued)

| People | Assignment | Agenda Item | Page |
|---|---|---|---|
| Sangren (CW) | Curtiss-Wright will submit by mail a proposal for a Standing Committee on Mathematical Analysis. | Misc. | I-A-22 |
| Shell (GE) | (1) General Electric AGT, Evendale, will study the new operation "Sense Copy Check" and circulate arguments by mail. | 20 | I-A-19 |
| | (2) Printer Boards Committee | Misc. | I-A-22 |
| Steele (RS) | The Rand Corporation will study further the new characteristic "Index Register ADD Instead of OR" and circulate arguments by mail. | 20 | I-A-18 |

# II
# APPENDICES

## APPENDIX A

## SHARE ADDRESS LIST

### REVISED 11-20-55

BA
Mr. Randall Porter
Physical Research Staff
Boeing Airplane Company
Box 3107
Seattle 14, Washington
MOhawk 7080
Extension 7901

CF
Mr. Henry S. Wolanski
Electronic Computations Laboratory
Convair Division of General Dynamics Corporation
Fort Worth, Texas
SUnset 7311
Extension 7134

CL
Mr. Lee Amaya
Mathematical Analysis Department
Lockheed Aircraft Corporation
Burbank, California
STanley 72711
Extension 1256

CR
Mr. William J. West
California Research Corporation
Box 446
La Habra, California
OXford 71746
Extension 112

CS
Mr. W. G. Gerkin
Digital Computation Laboratory, Building 35
Convair Division of General Dynamics Corporation
San Diego, California
CYpress 66611
Extension 488
or 795

CW
Mr. John A. DeVries
Nuclear Power Department
Research Division
Curtiss - Wright Corporation
Brighton Road
Clifton, New Jersey
GRegory 13000

DA
Mr. Walter C. Schlieser
Engineering Computing Group, B-250
El Segundo Division
Douglas Aircraft Corporation
El Segundo, California
ORegon 76161
Extension 1151

GA
Mr. Allan I. Benson
Lynn Digital Computations
Aircraft Gas Turbine Division
General Electric Company
1000 Western Avenue
West Lynn 3, Massachusetts
LYnn 36000
Extension 3431

GE
Mr. Donald L. Shell
Investigations Section
Aircraft Gas Turbine Development Department
Building 305
General Electric Company
Cincinnati 15, Ohio
POplar 14100
Extension 514

GL
Mr. Robert Bosak
Mathematical Analysis Department, 72-22
Lockheed Aircraft Corporation
Marietta, Georgia
CYpress 1411
Extension 2574

GM
Mr. Donald E. Hart
Special Problems Department
General Motors Research
Box 188, North End Station
Detroit 2, Michigan
JEfferson 95000
Extension 2626

GS
Mr. Harry Cantrell
Large Steam Turbine-Generator Department
Building 273-257
General Electric Company
Schenectady, New York
FRanklin 42211
Extension 3025

GT
Mr. Richard A. Butterworth
Medium Steam Turbine, Generator and Gear Department
General Electric Company
920 Western Avenue
West Lynn 3, Massachusetts
LYnn 36000
Extension 530

LA
Mr. Edward A. Voorhees
University of California
Los Alamos Scientific Laboratory
P. O. Box 1663
Los Alamos, New Mexico
Los Alamos 23051

LC
Dr. Sidney Fernbach
Theoretical Division
Radiation Laboratory
University of California
Livermore, California

ML
Dr. Werner W. Leutert
Research Laboratories
Missile Systems Division
Lockheed Aircraft Corporation
7701 Woodley Avenue
Van Nuys, California

ATTENTION: Mr. Benjamin F. Handy

STanley 64210

NA
Mr. Jack Strong
Department 92
North American Aviation, Inc.
International Airport
Los Angeles 45, California

ORegon 83011
Extension 2701

NC
Miss Elizabeth U. Blackwell
Engineering Computing Group
Department 56
North American Aviation, Inc.
4300 East Fifth Avenue
Columbus 16, Ohio

ATTENTION: Mr. Philip Arnold

DOuglas 1851
Extension 1483

NS
Mr. William D. Nichols
10605 Lilac Place
Silver Spring, Maryland

JAckson 5-5800
Extension 377
(NSA)

NY
Mr. William Heising
Data Processing Center
International Business Machines
Corporation
590 Madison Avenue
New York 22, New York

PLaza 31900

PK
Dr. Willard Bouricius
Computing Bureau, Department 537
International Business Machines
Corporation
Poughkeepsie, New York

Poughkeepsie 6920
Extension 790M

RA
Dr. Helmuth Hoelzer
Computation Laboratory
Building 491
Redstone Arsenal
Huntsville, Alabama

JEfferson 46411
Extension 3115

ATTENTION: Dr. Helmut Sassenfeld

RL
Mr. John F. Matousek
System Training Project
The Rand Corporation
1905 Armacost Avenue
West Los Angeles, California

GRanite 88293
Extension 33

RS
Mr. Paul Armer
The Rand Corporation
1700 Main Street
Santa Monica, California

EXbrook 40251
Extension 447

ATTENTION: Mr. Irwin Greenwald

UA
Mr. Stuart Crossman
Computation Laboratory
Research Department
United Aircraft Corporation
400 Main Street
East Hartford 8, Connecticut

JAckson 84811
Extension 7754

ATTENTION: Mr. Walter Ramshaw

WH
Dr. Edward Harder
Analytical Department 4L39
Westinghouse Electric Corporation
East Pittsburgh, Pennslyvania

EXpress 12800
Extension 2151

ATTENTION: Mr. Frank Engel

| | | |
|---|---|---|
| Chairman | Mr. Jack Strong<br>Department 92<br>North American Aviation, Inc.<br>International Airport<br>Los Angeles 45, California | ORegon 83011<br>Extension 2701 |
| Vice Chairman | Mr. Donald L. Shell<br>Investigations Section<br>Aircraft Gas Turbine Development Department<br>Building 305<br>General Electric Company<br>Cincinnati 15, Ohio | POplar 14100<br>Extension 514 |
| Secretary | Mr. Fletcher R. Jones<br>Department 56-72<br>North American Aviation, Inc.<br>International Airport<br>Los Angeles 45, California | ORegon 83011<br>Extension 2651 |

IBM Distribution

| | |
|---|---|
| Dr. John Greenstadt<br>704 Library Group<br>Data Processing Center<br>Applied Science Division<br>International Business Machines Corporation<br>590 Madison Avenue<br>New York 22, New York | PLaza 31900<br>Extension 126 |

## APPENDIX B

### REPRESENTATION AT THE THIRD MEETING OF SHARE

BOEING AIRPLANE COMPANY - Seattle
John Jordan
Randall Porter

CHANCE VOUGHT AIRCRAFT CORPORATION *
H. Allen Wood

CONSOLIDATED-VULTEE AIRCRAFT CORPORATION - Fort Worth
Henry S. Wolanski

CONSOLIDATED-VULTEE AIRCRAFT CORPORATION - San Diego *
William Gerkin
H. W. Buckner

CURTISS-WRIGHT CORPORATION
Ward Sangren
John A. DeVries

GENERAL ELECTRIC COMPANY - Evendale
Don L. Shell
James A. Porter

GENERAL ELECTRIC COMPANY - Lynn
Richard A. Butterworth
Everett L. Roger
Allen Keller
Robert L. Cushman
Paul E. Tanner

GENERAL ELECTRIC COMPANY - Schenectady
Jane King
Harry Cantrell

GENERAL MOTORS CORPORATION - RESEARCH
George Ryckman
Robert L. Patrick

GENERAL MOTORS CORPORATION - ALLISON DIVISION *
Ladd Mathiason

INTERNATIONAL BUSINESS MACHINES CORPORATION - Boston *
L. P. Rosenberry

INTERNATIONAL BUSINESS MACHINES CORPORATION - London *
Michael Barnett

## APPENDIX B

REPRESENTATION (continued)

INTERNATIONAL BUSINESS MACHINES CORPORATION - New York
John Greenstadt
Harold Judd
John Backus
William P. Heising
Truman Hunter

INTERNATIONAL BUSINESS MACHINES CORPORATION - Poughkeepsie
Willard Bouricius
Edward Goldstein

LOCKHEED AIRCRAFT CORPORATION - CALIFORNIA DIVISION
Lee Amaya

LOCKHEED AIRCRAFT CORPORATION - GEORGIA DIVISION
Gerald Fine

LOCKHEED AIRCRAFT CORPORATION - MISSILE SYSTEMS DIVISION
Benjamin Handy

MASSACHUSETTES INSTITUTE OF TECHNOLOGY *
Dean Arden
John M. Frankovich

NORTH AMERICAN AVIATION INC. - Columbus
Paul H. Arnold

NORTH AMERICAN AVIATION INC. - Los Angeles
Jack A. Strong
Frank V. Wagner
Owen R. Mock

NATIONAL SECURITY AGENCY
John T. Powers
Thomas E. McCool
William D. Nichols

THE RAND CORPORATION
Tom B. Steel, Jr.

REDSTONE ARSENAL
Helmut Sassenfeld

UNITED AIRCRAFT CORPORATION
Roy Nutt
Walter Ramshaw

UNIVERSITY OF CALIFORNIA - Los Alamos
Thomas L. Jordan

## APPENDIX B

REPRESENTATION (continued)

WESTINGHOUSE ELECTRIC CORPORATION
Frank Engel, Jr.
Ben Mount
Edward Harder

* Nonparticipating

SHARE

**SHARE Inc.**
25 Broadway, Suite 750
New York, New York 10004
(212) 943-2130

The Equitable Life Assurance
Society of the United States
1285 Avenue of the Americas
New York, New York 10019
(212) 554-3678
August 8, 1975

THOMAS B. STEEL, JR.
Director

Dear colleague:

In classical SHARE tradition, as refined over the past twenty years, this letter is being sent very late. Nevertheless it is my hope that you will be able to participate in the twentieth anniversary of SHARE. The SHARE XLV Meeting will be held 1975 August 18-22 at the New York Hilton. If you can participate for the entire week, SHARE would be honored to have you register. If, however, you could only spend a brief time, Wednesday afternoon, August 20 is the time we plan to conduct a session, "An Information Processing Retrospective", at which we plan to review why SHARE started and how it has evolved, as well as consider what it might become. If it is at all possible we would like your participation at that session.

This letter is being sent to every participant at SHARE I that can be found, as well as all of the past Presidents of SHARE and a very few others. Many of you are already aware of these plans from earlier discussion and I hope you have already made your plans to participate.

Any of you who cannot be there but have some observations about SHARE and its two decades can communicate them to me and I will see that they are exposed at the meeting.

I am looking forward to seeing many of you at SHARE XLV.

Sincerely yours,

T. B. Steel

T. B. Steel, Jr.
SHARE Historian

# FIRST USA-JAPAN COMPUTER CONFERENCE PROCEEDINGS

OCTOBER 1972

TOKYO

COSPONSORED BY

AFIPS & IPSJ

-691

# A TRILOGY ON ERRORS IN THE HISTORY OF COMPUTING*

N. Metropolis and J. Worlton

(Los Alamos Scientific Laboratory, University of California,
Los Alamos, New Mexico)

## 1. INTRODUCTION

The critic who investigates the inadequacies of the history of computing is at once faced with an embarrassment of riches. Computer scientists seem determined to confirm the judgment of professional historians that scientists should not be depended upon to produce the histories of their own fields(1). Sarton, in an essay on "The Scientific Basis of the History of Science"(2), pays tribute to the "good amateurs" who work as hard in the field of history as they do in their own specialties, but complains that the amateur historian of science is more often

> ...a distinguished scientist who has become sufficiently interested in the genesis of his knowledge to wish to investigate it, but has no idea whatsoever of how such investigations should be conducted and is not even aware of his shortcomings. His very success in another domain, the fact that he has long passed the years of apprenticeship, make it difficult, if not impossible, for him to master a new technique. He generally lacks the humility of a beginner, and publishes his historical results with blind and fatuous assurance. This is amateurism at its worst.

Computer science is fortunate to have people trained in both history and computing to direct the major project on the history of computing at the Smithsonian Museum of History and Technology(3), but there is an essential role for the "good amateur" to play in preparing this history. The field is so broad and the professional historians so few that they cannot do all of the detailed work of collecting, organizing, and documenting that is necessary; further, much of the information is known only to the computing pioneers who are, by and large, amateurs in the field of history.

Although this paper emphasizes the inadequacies and misunderstandings in published accounts of the history of computing, it is not its purpose to discourage further efforts, but to encourage them and to emphasize that the history of computing deserves to be known as well as possible, and any knowledge short of what is attainable should be treated with the same contempt as we would treat half-baked knowledge in computing itself(2). Since the authors of this paper are amateurs in the field of history, the proposals made here for the improvement of work in this field are modest.

1. Allow no published error to go uncorrected. Only through a vigorous weeding process can we hope to stop the propagation of the seeds of error.

2. Do not publish conjectures as though they were facts. Lack of caution is one of the obvious marks of the "bad amateur."

3. Do not depend upon secondary sources. The error function for Nth-level repetition is monotonically increasing.

4. Remember that the basis of scientific history is bibliography. Start with a good bibliography and end with a better one.

Specific professional suggestions can be obtained from George Sarton's dual publication, "The Study of the History of Mathematics" and "The Study of the History of Science." The four basic suggestions noted above, however, will at least lead authors toward professional standards of history.

In the three studies which follow, we first take note of published errors or misunderstandings in the history of computing and then provide results of research intended to provide corrections.

## 2. BABBAGE AND THE ORIGINATORS OF MODERN COMPUTERS

### 2.1 A Question of Awareness

The creative genius that Charles Babbage exhibited in his design of general-purpose mechanical computing devices has surprised and delighted readers for well over a century. As noted by Bowden, it is difficult to determine whether Babbage's oddities were responsible for his failure to complete his machines or his failures made him odd(4); in either event, he is often accorded the honor of being the "Father of Computing." In recent articles, however, it has been claimed that those responsible for the development of modern computers were not only not influenced by the ideas of Babbage but that they were not even aware of his work.

The question of the "influence" of one person's work on that of another is often subjective and difficult to establish; however, it is possible to establish "awareness" by documenting references in the writings of the people concerned, and it is this point which is addressed here, i.e., are there references in their writings to the work of Charles Babbage? The claims noted below imply a uniformly negative answer,

*This work was done under the auspices of the United States Atomic Energy Commission.

Turing's paper on "Computing Machinery and Intelligence," first published in 1950, refers to both Babbage and Lady Lovelace, but this is as early as we are able to document Turing's awareness of Babbage from his own writings. However, in an interview among the authors, J. H. Wilkinson, and Leslie Fox, Wilkinson stated unequivocally that for all those working on the "ACE" project at the National Physical Laboratory, Teddington, England (N.P.L.), including Turing, "Babbage" was a household word and very much a topic of conversation "...very, very early on." One of the buildings at N.P.L. was called "Babbage Building," and the ACE project was referred to by Womersley as "Babbage's Dream." Since Wilkinson was Turing's closest associate on the logical design of ACE, his information on Turing's awareness of Babbage should be considered conclusive.

The published works of von Neumann contain no references to Babbage, but this is "negative evidence" in the sense that it does not resolve the question of an awareness of Babbage on the part of von Neumann. However, von Neumann's intense interest in and deep knowledge of history are well known, and the authors have been advised by Julian Bigelow, one of von Neumann's close associates on the IAS computer project at Princeton, that Babbage was the subject of a discussion among Hartree, von Neumann and himself on the occasion of Hartree's second visit to Princeton in 1947. Contacts between Turing and von Neumann would have provided opportunity for such information to be discussed, either before the War at Princeton or during the War, in England(93). The authors have been advised by the Earl of Halsbury that classified documentation exists in England on "...Turing's warwork on code breaking with electronic devices..." which may bear on this subject, but as yet efforts to have this material declassified have been unsuccessful(92). Thus, while no available documentary information supports it, there is some informal evidence to support the claim of a knowledge of Babbage on the part of von Neumann.

In summary, Calvert's assertion about a lack of awareness of Babbage among the inventors of the electronic computer in the United States seems indeed to be safe, but when the more general question of awareness among other early workers in this field is considered, a rather broad awareness of Babbage can be demonstrated.

## 3. THE STORED-PROGRAM CONCEPT AND EARLY IMPLEMENTATIONS

### 3.1 Control Modes

There has been some confusion in the literature concerning the origin of the stored-program concept and the early implementations thereof. Some of these errors are easy to correct, as for example, the notion that the Zuse Z3 was an electronic stored-program machine(57). In fact, the Z3 was an electromechanical computer which was indeed programmgesteuerte (program controlled), but whose program control was implemented through the use of eight-bit one-address instructions punched in 32-mm cinefilm, rather than in the 64 words of relay storage(6).

Some of the confusion concerning program control in early computers is derived from a lack of understanding of the stages through which program control in scientific computers evolved during the 1940's. Figure 1 is an idealized classification of this evolution, with the pioneering machines noted for each level; each of these levels is discussed briefly below.

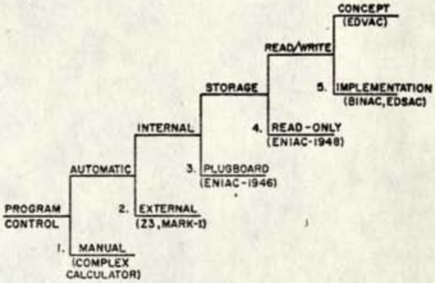


Fig. 1. Evolution of program control modes.

1. Manual control was used on the Bell Labs' "Complex Calculator" (1940), with instructions being entered through a teletypewriter keyboard. This device was thus more nearly related to modern desk calculators than to modern computers.

2. Automatic control of a calculational sequence was achieved in the Zuse Z3 (1941) and the Harvard Mark-I (1944) through the use of external program readers for film (Z3) and paper tape (Mark-I).

3. Internal control of calculations was first implemented in the ENIAC (1946) through the use of jackplugs and switches to route control signals. "Programming" for this machine consisted of making jackplug connections and setting switches.

4. Storage control of a computer was later implemented on the ENIAC (1948) through the use of a decoding matrix in conjunction with the read-only function tables.

5. Read-write memories for stored programs were first implemented in 1949 on the BINAC in the United States and the EDSAC in Great Britain. These machines were based on the design of the EDVAC (1945).

### 3.2 Concept vs. Implementation

A natural source of error concerning the history of the stored program is the failure to distinguish between the origins of the concept and its first implementation. The design group working at the Moore School of Electrical Engineering of the University of Pennsylvania produced the functional design of the EDVAC(58), which included acoustical delay lines to hold both programs and data, even prior to the completion of the ENIAC, so it is correct to credit the EDVAC design as being the first to employ the stored-program concept. However, construction of the EDVAC was delayed because of the departure of key personnel from the project after the war, and the machine was not completed until 1952(59). In 1948 an instruction decoder was added to the ENIAC at the

Ballistic Research Laboratories(60) which allowed the 312 words of read-only storage on the portable function tables to be used to hold instructions, and ENIAC became the first computer to operate with a read-only stored program. In 1949 both the BINAC in the United States(61) and the EDSAC in Great Britain(62) were successfully tested; these designs were based on that of EDVAC, with a dynamically modifiable stored program being executed out of the delay-line storage. Thus, it is correct to:

(a) credit the EDVAC design as the first to include the stored-program concept;

(b) credit the ENIAC as the first computer to be run with a read-only stored program; and

(c) credit the BINAC and the EDSAC as being the first computers to be run with a dynamically modifiable stored program.

Because item (b) is so little known, we have provided the basic information in the next section.

### 3.3 The Read-Only Stored-Program Control on the ENIAC

Each preparation of the ENIAC for a new problem was a time-consuming affair; the control consisted of a very large, distributed plugboard and manually-set switches. One part of the read-only memory--the so-called "function tables"--could store 312 numbers of twelve decimal digits effected by ten-position, manually-set switches. R. F. Clippinger(63) (not von Neumann as stated in references 60 and 64) suggested that the function tables might be used to store sequences of decimal digit pairs, each pair corresponding to one of a possible hundred instructions, and that the control might be implemented (once-and-for-all) to interpret and execute such pairs. To change problems, a new sequence would be introduced--a much simpler procedure than the jackplug and switch method.

A provisional plan by A. Goldstine for a control program exceeded the capacity of the ENIAC. A second approach by N. Metropolis and K. von Neumann was successful, but only because of a curious coincidence. On a preliminary visit to the Aberdeen Proving Ground in Maryland when the ENIAC had been moved from Philadelphia(*), Metropolis noticed a complete many-to-one decoder network nearing completion; it was intended to increase the capability of executing iterative loops in a program. It was also just what was needed to simplify considerably the decoding of digit pairs representing an instruction, and in fact, the new mode of control could be contained. The local authorities agreed to the change and the campaign was launched; after at least the expected number of program errors had been committed and eventually removed, the ENIAC achieved a read-only stored program. The time scale to change problem setups was reduced from hours to minutes. Moreover, maintenance procedures were simplified.

In the original ENIAC form of control a limited amount of parallel operation was possible; this was sacrificed in converting to strictly sequential execution. All the remaining flexibilities were available in the new modus operandi, however.

After some thorough testing, Metropolis and K. von Neumann put the first problem--the original Monte Carlo--to the ENIAC in its new form in early 1948.

### 3.4 Originators

Another point concerning the stored-program history which needs clarification is the unwarranted assumption that J. von Neumann alone deserves the credit for the stored-program concept. In his Turing lecture in 1967, Maurice Wilkes (who attended the Moore School lectures in 1946) gave the following description of the roles played by Eckert and Mauchly on the one hand, and von Neumann on the other:

> Eckert and Mauchly appreciated that the main problem was one of storage, and they proposed for future machines the use of ultrasonic delay lines. Instructions and numbers would be mixed in the same memory...von Neumann was, at that time, associated with the Moore School group in a consultative capacity...The computing field owes a very great debt to von Neumann. He appreciated at once...the potentialities implicit in the stored program principle. That von Neumann should bring his great prestige and influence to bear was important, since the new ideas were too revolutionary for some, and powerful voices were being raised to say that the ultrasonic memory would not be reliable enough, and that to mix instructions and numbers in the same memory was going against nature...Subsequent developments have provided a decisive vindication of the principles taught by Eckert and Mauchly...(65)

The historical document which is crucial to this discussion is the progress report on the EDVAC written in September 1945 by Eckert and Mauchly; the following information is taken from section "1.0 Historical Comments" of that report.

> ...in January, 1944, a "magnetic calculating machine" was disclosed...An important feature of this device was that operating instructions and function tables would be stored in exactly the same sort of memory device as that used for numbers...The invention of the acoustic delay line memory device by Eckert and Mauchly early in 1944 provided a way of obtaining large high-speed storage capacity with comparatively little equipment...Therefore, by July, 1944 it was agreed that when work on the ENIAC permitted, the development and construction of such a machine should be undertaken. This machine has come to be known as the EDVAC (Electronic Discrete Variable Computer)...During the latter part of 1944, and continuing to the present time, Dr. John von Neumann, consultant to the Ballistic Research Laboratory, has fortunately been available for consultation. He has contributed to many discussions on the logical controls of the EDVAC, has prepared certain

(*) A heroic achievement, watched by many, all named Thomas!