DATA Control

What the Engineer Should Know About Programming ·

How to Consider A Computer

Engineering is taking on a "new look." Computers are the logical and more powerful successors to the desk calculator and the slide rule, the previous working tools of the engineer. There is really only one major difference: because of their necessary size and cost to be so powerful, computers must be shared by a great many users. This means a new concept of shared system operation must be accepted by the engineer.

To help you get oriented, here are some vital considerations affecting present and future computer use in your work and some helpful sources of further highly specialized information.

BY ROBERT W. BEMER
Programming Research Department
International Business Machines

■ A computer should not be rented or purchased unless an automatic programming or coding system is furnished for its operation. The computer and the operational system constitute a matched pair, and one without the other is highly unsatisfactory from the point of view of getting work done at minimum cost.

For engineering work, any automatic system should contain provision for indexing and floating point operation, if these are not built in as hardware, for they are the two most vital features for easy usage. Indexing allows for algebraic array notation, which in turn makes for easy understanding of how a problem should be programmed. Floating point, although it may sometimes introduce either spurious accuracy or loss of it to the uninitiated, prevents a Gordian tangle of scaling difficulties from cluttering up the problem.

HOW CODING SYSTEMS HELP

Automatic coding systems have by no means reached their ultimate efficiency or sophistication, yet remarkable savings in programming costs have already been achieved, sometimes by an order of 50! For the best of the present systems it is a reasonable estimate to say that they can, in general, reduce the programming costs and time to a tenth of that required to code in stubborn machine language.

There have been many attempts to relieve the burden of programming through special coding systems of all types. The data sheet on computer coding systems is not only an interesting history of growth, but is also presented for the edification of those now entering the field with incomplete knowledge of what code to use for their machine. The time may come soon when you will be using a common language exclusive of the characteristics of any particular computer. Thus, with an automatic translator for each different computer, a running program may be produced for any desired machine from the single original problem and procedure statement in the common language. Credit is due to Dr. Saul Gorn of the Moore School of Electric Engineering for first championing these principles.

GOOD COMPUTER OPERATION IS STATE OF USER'S MIND

It is axiomatic that a computer should never stop, run useless problems or be subjected to manual oper-

ation and dial-twiddling. To do so deprives your fellow engineers of its benefit. Here are some detailed considerations pertinent to good computer operation:

- It can do only what it is explicitly ordered to do, and this ordering must be done eventually in its own machine language, which is all it can understand.

- The reliability of most present-day computers is so high that, if answers are not right or reasonable, the chances are at least 99 to 1 that it is somehow the user's fault. Wrong answers usually stem from wrong equations or misuse.

- Allow for growth when doing the original planning. Build in flexibility for changes, or else costs will soar if the entire problem must be re-programmed. A stored-program may always be corrected or augmented to give exactly what the engineer desires, including special report format for jobs where repetition justifies the effort.

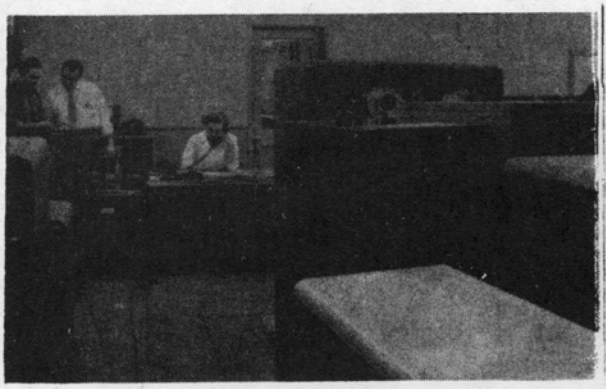
- For design studies, plan parameter variation carefully and allow flexibility in changing individual parameters. The computer may surprise you by showing that some parameter values for optimum conditions may be outside of the range expected or allowed for. To make certain that the program gives answers consistent with hand-computations, first test the program on the machine for a specific combination. Time this run. Then multiply this time by the total number of parameter combinations to see if the study is feasible in time and cost. If M parameters are combined for N values each, the total number of combinations is N^M.
For example, with 6 Parameters: 6 values for each will produce 46,656 combinations; 5 will produce 15,625; 4 will produce 4,096.
The moral: *Don't triple the cost* of your problem if you are engineer to draw a curve through one less point.

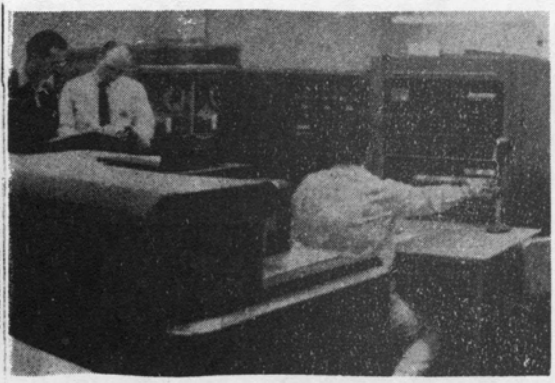
FUTURE COMPUTER LANGUAGES

New synthetic languages are in the process which will affect your use of computers. As problem-solving languages they will be much superior to present systems in these ways:

1. Even though the binary type of computer will probably be universal for both engineering and commercial work, the need for the user to know binary representation will effectively disappear. Logical decision will be the only remaining function which will not appear in decimal form, and even here facilities will be provided so that the programmer need not concern himself with the precise method of operation within the machine. Conversion from fixed point decimal to floating point binary for operation, and the converse for output, will all be done automatically by the synthetic language translator.

2. The elements of the synthetic language will be essentially algebraic, both arithmetic and logical, and linguistic so that procedures may consist of real sentences in a living language. Idiom will be such that the program will be operative in any spoken language, with minor changes





COMPUTER	SYSTEM NAME OR ACRONYM	DEVELOPED BY	CODE	SYSTEM TYPE				OPER. DATE	INDEXING	FL PT	SYMB.	ALGEB.	COMMENTS
				M.L.	ASSEM.	INTER.	COMP.						
I.B.M. 704	R-S	Los Alamos			X			Nov. 55	M2	M	1		
	Cage	General Electric			X		X	Nov. 55	M2	M	2		
	Fortran	I.B.M.	1				X	Jan. 57	M2	M	2	X	
	NYAP	I.B.M.			X			Jan. 56	M2	M	2		
	Pact 1A	(See Pact Group)					X	Jan. 57	M2	M	1		Modified Pact I for 704
	SAP	United Aircraft	1		X			Apr. 56	M2	M	2		Official Share Assembly
Sperry-Rand 1103A	Compiler I	Boeing, Seattle			X		X	Mar. 57		S	1	X	
	FAP	Lockheed M.S.D.	1	X		X		Oct. 56	S1	S	0		
	Mishap	Lockheed M.S.D.			X			Oct. 56		S	1		Magn. Tape Assembly + Correction
	Trans-Use	Holloman A.F.B.			X			Nov. 56		S	2		
	Use	Ramo-Wooldridge	1		X		X	Feb. 57	M/S1	M/S	2		Official for Use Organization
1103	Chip	Wright Field		X		X			S1	S	0		Similar to Flip
	Flip/Spur	Convair San Diego		X		X		Jun. 55	S1	S	0		Spur Unpacked, Twice as Fast
	Rawoop	Ramo-Wooldridge	1		X			Mar. 55	S1	—	1		One-Pass Assembly
	Snap	Ramo-Wooldridge	1		X	X		Aug. 55	S1	S	1		Used with Rawoop
I.B.M. 705	Autocoder	I.B.M.	1	X	X		X	Dec. 56	—	S	2		
	Fair	Eastman Kodak				X		Jun. 56	—	S	0		
	Print I	I.B.M.	1	X	X	X		Oct. 56	S2	S	2		
	Symb. Assem.	I.B.M.			X			Jan. 56	—		1		May be Assembled on Acctng. Equip.
	SOHIO	Std. Oil of Ohio		X	X	X		May 56	S1	S	1		
Sperry-Rand Univac I, II	A0	Remington Rand		X	X		X	May 52	S1	S	1		
	A1	Remington Rand		X	X		X	Jan. 53	S1	S	1		
	A2	Remington Rand		X	X		X	Aug. 53	S1	S	1		
	A3	Remington Rand	2	X	X		X	Apr. 56	S1	S	1		
	AT3	Remington Rand	2		X		X	Jun. 56	S1		2	X	Fortran-Like, Output To A3, I + II
	BO	Remington Rand	1,2	X	X		X	Dec. 56	S2	S	2		Runs on Univac I + II
	BIOR	Remington Rand		X	X		X	Apr. 55	—	—	1		Primarily Business Data-Processing
	GP	Remington Rand	1	X	X		X	Jan. 55	S2	S	1		For Expert Programmers
	NYU	New York University					X	Feb. 54	—	S	1		
	Relcode	Remington Rand		X	X			Apr. 56	—	—	1		
	Short Code	Remington Rand		X		X		Feb. 51	—	S	1		
	X-1	Remington Rand	2	X	X			Jan. 56	—	—	1		Runs on Univac I + II
I.B.M. 702	Autocoder	I.B.M.		X	X		X	Apr. 55	—	S	1		
	Assembly	I.B.M.			X			Jun. 54	—	—	1		May be Assembled on Acctg. Equip.
	Omnicode	G.E. Hanford	2		X	X	X	Propos	S2	S	2		Super-Script
	Script	G.E. Hanford	1	X	X	X	X	Jul. 55	S1	S	1		
I.B.M. 701	Acom	Allison G.M.				X							
	Bacaic	Boeing, Seattle	1,2		X		X	Jul. 55	—	S	1	X	
	Douglas	Douglas Sm			X			May 53	—	S	1		
	Dual	Los Alamos		X		X		Mar. 53	—	S	1		
	607	Los Alamos			X			Sep. 53	—	—	1		
	Flop	Lockheed Calif.		X	X	X		Mar. 53	—	S	1		
	Jcs 13	Rand Corp.			X			Dec. 53	—	S	1		Modification of 607
	Kompiler 2	Ucrl Livermore					X						
	Naa Assembly	N. Amer. Aviation			X								Also Assembles 704 Programs
	Pact I	(See Pact Group)	1				X	Jun. 55	S2	—	1		Most Programs run on Pact IA
	Queasy	Nots Inyokern				X		Jan. 55	—	S	—		
	Quick	Douglas Es				X		Jun. 53		S	0		Double Quick for Dbl Prec
	Seesaw												
	Shaco	Los Alamos			X	X		Apr. 53		S	1		
	So 2	I.B.M.			X			54					
	Speedcoding	I.B.M.	1		X	X		Apr. 53	S1	S	1		
Datatron	Ctc	Purdue Univ.	2				X	Incomp	S2	S	1	X	
	Dot I	Electro Data				X							
	Ugliac	United Gas Corp.				X							
I.B.M. 650	Ades II	Naval Ordnance Lab					X	Feb. 56	S2	S	1	X	
	Bacaic	Boeing, Seattle	2		X	X	X	Aug. 56	—	S	1	X	Must Process on 701
	Baltiac	M.I.T.		X	X		X	Jan. 56	S1		2		For all 650 models
	Bell	Bell Tel. Labs		X		X		Aug. 55	S1	S	0		
	Ctc	Carnegie Tech	1,2				X	Dec. 56	S2	S	1	X	Output Processed by soap
	Druco I	I.B.M.				X		Sep. 54		S	0		
	Flair	Lockheed Msd, Ga.		X		X		Feb. 55	S1	S	0		
	Mitilac	M.I.T.		X		X		Jul. 55	S1	S	2		For all 650 models
	Omnicode	G.E. Hanford	2			X	X	Dec. 56	S1	S	2		Must Process on drum 702
	Sir	I.B.M.				X		May 56	—	S	2		Operates with soap I, II
	Soap I	I.B.M.			X			Nov. 55	—		2		
	Soap II	I.B.M.	1		X			Nov. 56	(M)	(M)	2		For all 650 models + variations
	Speed coding	Redstone Arsenal		X		X		Sep. 55	S1	S	0		Resembles 701 speedcoding
	Spur	Boeing, Wichita		X	X	X		Aug. 56	(M)	S	1		For all 650 models
Whirlwind	Algebraic	M.I.T.					X		S2	S	1	X	
	Comprehensive	M.I.T.		X	X	X		Nov. 52	S1	S	1		
	Summer ses.	M.I.T.				X		Jun. 53	S1	S	1		
Midac	Easiac	Univ. of Michigan			X	X		Aug. 54	S1	S	1		
	Magic	Univ. of Michigan			X	X			S1	S	1		
Burroughs		Burroughs Lab					X				1		
Ferranti	Transcode	Univ. of Toronto	1		X	X	X	Aug. 54	M1	S	1		
Illiac	Dec order input	Univ. of Illinois	1		X			Sep. 52	S1	S	1		
Johnniac	Easy fox	Rand Corp.			X	X		Oct. 55		S	1		
Norc		Naval Ordnance Lab			X		X	Feb. 56	M2	M	1		
Seac	Base 00	Natl. Bur. Stds.		X		X							

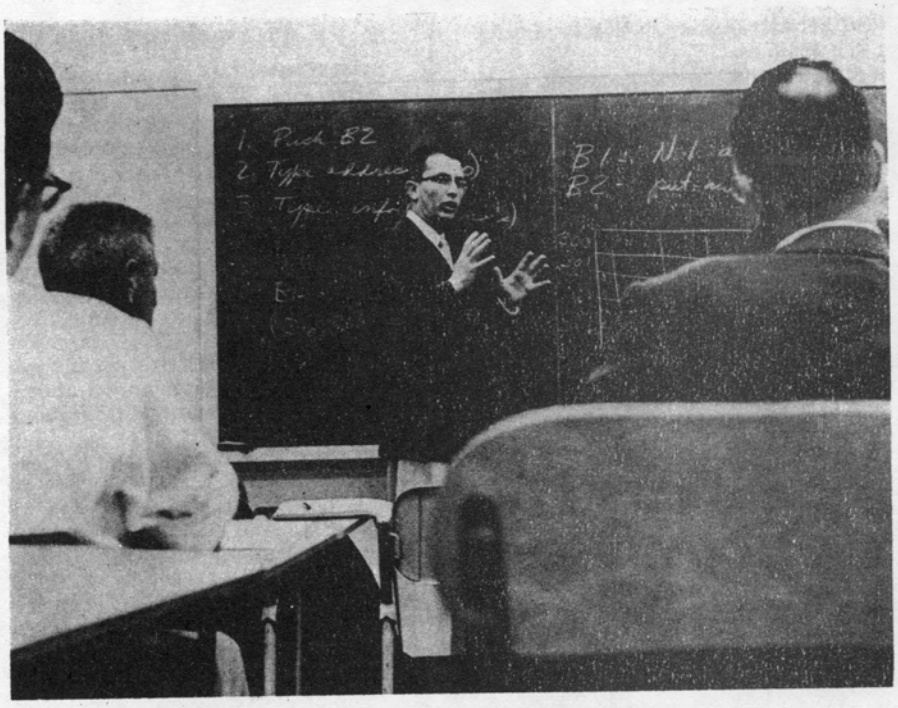
Pact Group Contains Douglas Sm, Es, Lb, Lockheed Cal, Nots, N. Amer., Rand

in the dictionary, much as the FORTRAN coding system has already been translated for use by the French.

● Much more intelligence will be built into the translators. The program may make a reasonable guess about the intent of the programmer when some omission or violation of language rules occurs. Learning procedures will be incorporated so that the translator may take advantage of statistics of previous operation to improve the program which it creates. This may extend as far as a form of creativity where the date processor may originate programs merely from the statement of the problem to be solved, rather than from a given procedure for this solution.

4. Flow-chart construction for procedures will be automatic, having been determined in their linkages by the before-and-after conditions for the components of procedure. Today, Monte Carlo techniques are already being used to determine frequencies of occurrence for various logical branchings in the procedure, with resultant optimization of the program by taking these factors into consid-●on. The efficient usage of the various and graded components of the computer system will be automatic; the programmer considers an infinite machine, which the processor arranges as it knows its own limitations and capabilities.

5. Not only will programs be created which write programs to do actual computation, but other levels



will be superimposed on these. Bootstrap methods are being considered which will allow even the person who develops the processor to have a good portion of his work done automatically by reference to and through previous work.

6. The actual operation of the computer will be under control of an integrated portion of the processor known as a supervisory routine. In some instances the program will not have been created by the processor prior to execution time, but will be created during a break in execution time under orders of this supervisory routine, which detects that no method is in existence in the program for a particular contingency. Although these supervisors will be on magnetic tape for a while, it is envisioned that they will be buried in the machine hardware eventually, to be improved by replacement like any other component.

FUTURE COMPUTER SYSTEMS

Future computer operation, which strongly influences the design of the programming languages, has some vitally interesting possibilities. In this glimpse, the picture presented here is dependent upon three axioms:

▸ Faster computers always lower the dollar cost per problem solved, but not all companies will be able to afford the high prices of the next generation of super-computers. They simply may not have enough problems to load one.

▸ Producing a spectrum of machines is a tremendous waste of effort and money on the part of both the manufacturers and the users.

▸ Availability of a huge central computer can eliminate the discrete acquisition of multiple smaller computers, homogenize the entire structure of usage, and allow a smaller and more numerous class of user into the act, thus tapping a market many times the size of presently projected with current practice in computer access.

Assuming the availability of practical micro-wave communication systems, it is conceivable that one or several computers, much larger than anything presently contemplated, could service a multitude of users. They would no longer rent a computer as such; instead they would rent input-output equipment, although as far as the operation will be concerned they would not be able to tell the difference. This peripheral equipment would perhaps be rented at a base price plus a variable usage charge on a non-linear basis. The topmost level of supervisory routine would compute these charges on an actual usage basis and bill the customer in an integrated operation. These program features are, of course, recognizable to operations research

---

EXPLANATION OF SYMBOLS

CODE 1) Recommended for this particular computer, sometimes for volume of usage or interchange, or for lack of better.
2) Common language to more than one computer.
SYSTEM TYPE (See "Terms Used In Automatic Coding")
INDEXING M) Actual index registers or B-boxes exist in machine hardware.
S) Simulated in the synthetic language.
1) Limited form, either stepped unidirectionally or by 1 word only, having certain registers applicable to only certain address positions, or not compound (by combination).
2) General form, where any address may be indexed by any 1 or a combination of registers, which may be freely incremented or decremented by any amount. Have associated loop termination instructions.
FLOATING PT. M) Inherent in machine hardware.
S) Simulated in the synthetic language.
SYMBOLISM 0) None.
1) Limited—either regional, relative or exactly computable.
2) Fully descriptive, where a word or symbol combination may be descriptive of the contents of the assigned storage.
ALGEBRAIC A single continuous algebraic formula statement may be made. The processor contains mechanisms for applying the associative and commutative laws of algebra to form the operative program.

people as the Scheduling and Queuing Problems.

Using commutative methods, just as motion pictures produce an image every so often for apparent continuity, entire plant operations might be controlled by such super-speed computers.

These future hardware capabilities (and few competent computer manufacturers will deny the feasibility, even today, of super-speed and interleaved programs) demonstrate a pressing need for an advanced common language system so all users concerned can integrate their particular operations into the complex of control demanded by an automated future.

Just one last prediction—the engineer who is going to be at the top of his profession in the years to come had better become a computer expert, too.

A WORKING GLOSSARY OF SOME AUTOMATIC CODING TERMS

AUTOMATIC CODING — Systems which allow programs to be written in a synthetic language especially designed for problem statement, which the processor translates to presumably the most efficient final machine language code for any given computer. Usually such a system will examine one entry at a time and produce some amount of coding which is determined by that entry alone.

AUTOMATIC PROGRAMMING—Systems further up the scale of complexity, where the computer program helps to plan the solution of the problem as well as supply detailed coding. Such systems usually examine many entries in parallel and produce optimized coding where the result of any single entry depends upon its interactions with other entries.

ASSEMBLER—An original generic name for a processor which converts, on a one-for-one basis, the synthetic language entries to machine instructions. This process occurs prior to the actual execution of the working program. It is a one-level processor which can combine several sections or different programs into an integrated whole, meanwhile assigning actual operation codes and addresses to the instructions.

COMPILER — Generally a more powerful processor than the assembler, although there is a great deal of confusion and overlapping of usage between the two terms. The compiler is capable of replacing single entries with pre-fabricated series of instructions or sub-routines, incorporating them in the program either in-line or in predetermined memory positions with standard mechanisms for entry from and exit to the main routine. Such compound entries are sometimes called "macro-instructions." The basic principle of a compiler is to translate and apply as much intelligence as possible ONCE before the running of the program, to avoid time-consuming repetition during execution. It produces an expanded and translated version of the original, or source program. According to the ACM, a compiler may also produce a secondary synthetic program for interpretation while running.

FLOATING POINT—Number notation whereby a number X is represented by a pair of numbers Y and Z in the form: $X = Y \cdot B^Z$ where B is the number base used. For floating decimal notation the base B is 10; for floating binary the base is 2. The quantity Y is called the fraction or mantissa, and in the best notation $O = Y - 1$. Z is an integer called the exponent or power.

GENERATOR—A generator is a program which writes other programs, usually on a selective basis from given parameters and skeletal coding. It may be either a character-controlled generator, so that it selects among several options according to a preset character matrix, or a pure generator, which writes a program on the basis of calculations which it makes from the input data. Almost all assemblers and compilers have generating elements in some form.

INDEX REGISTER — A register whose contents are used to automatically modify addresses incorporated in instructions just prior to their execution, the original instruction remaining intact and unmodified in memory. It may either be built in the hardware and circuitry of the computer or be simulated by the program. The original unmodified addresses are termed presumptive, the modified addresses are termed effective.

INTERPRETER—In contrast to an assembler, compiler or generator, a source program designed for interpretation is converted to an object program which is not in machine language when run. The interpreter itself is an executive program which must always be used in conjunction with the object program, and always resides in high-speed memory during execution of the problem. The object program is always subservient to the interpreter, which fabricates from the incompletely converted instruction the necessary parts of machine language instructions just before they are executed. Each entry in the interpretive language usually calls for the use of a complete subroutine, which is used again and again by filling in the missing parts of certain of its instructions from the object instruction.

MACHINE LANGUAGE—The wired-in circuitry language at a low logical level which is intelligible to the computer. It should seldom be used to code problems because of the difficulties of usage at this level and the tendency to error.

OBJECT PROGRAM—The output of the processor when it has translated the source program to either machine language or a second level synthetic language.

PROCESSOR—Also called a translator, this is a computer program which produces other programs, in contrast to programs which are working and produce answers.

SOURCE PROGRAM—The original program written to solve problems and produce answers, phrased in the synthetic language.