a Review of STRETCH

Memorandum for: Mr. J. W. Birkenstock
Mr. J. A. Haddad
Dr. C. C. Hurd
Mr. R. L. Palmer

February 8, 1956

SUBJECT: Stretch Proposal

Attached is a report reviewing the subject proposal prepared by a group of Applied Science, Engineering and Product Planning personnel. The report discusses certain inadequacies of the proposal and presents recommendations for their correction.

The attached report further finds that random access storage capacities in excess of 1,000,000 words are essential to a well-balanced computer having the speed of the Stretch machine. A program to develop random access storage devices in this area is recommended.

J. W. Backus

JWB/mms
Enclosure

CC: Miss E. M. Boehm - WHQ
Dr. C. R. DeCarlo - WHQ
Mr. S. W. Dunwell - Poughkeepsie
Mr. B. O. Evans - WHQ
Dr. J. L. Greenstadt - WHQ
Mr. J. E. Griffith - Poughkeepsie
Mr. J. B. Jackson - Richland, Washington
Mr. F. E. Johnston - Albuquerque
Mr. W. F. McClelland - Chicago
Mr. D. W. Pendery - WHQ
Mr. D. W. Sweeney - Endicott

COMPANY CONFIDENTIAL

A review of the
STRETCH PROPOSAL
February 7, 1956

The following study group has reviewed the subject proposal:

Applied Science:
J. W. Backus
J. L. Greenstadt
J. B. Jackson
F. E. Johnston
W. F. McClelland
D. W. Sweeney

Engineering:
E. M. Boehm

Product Planning:
J. E. Griffith

The study group finds the proposed computer weak in certain areas. These areas are listed below in order of importance. It is felt that the recommended changes in these areas should either be included in the original proposal or offered at additional cost in order to assure Los Alamos that a satisfactorily balanced machine will be available.

1. The total random access storage capacity of the machine (32,000 words) is entirely inadequate for a machine having the proposed arithmetic speeds.

   a. This 32,000 word storage capacity is already offered with a machine (the 704) having approximately 1/200 the arithmetic speed of the Stretch machine.

   b. The largest matrix which can be stored in the main storage units of the machine can be inverted in approximately 20 seconds.

   c. The access time of the high speed RAM units is such that it is not feasible to transmit blocks of information smaller than 10,000 words. A 32,000 word storage capacity then permits only three blocks of information to appear in storage.

c. cont.

A well balanced machine should have a storage capacity to contain at least ten blocks. The 704 main storage can contain seventeen blocks. A "block" of information from an auxiliary storage unit is here considered to have a number of words whose transmission time from the unit is twice the average access time of the unit.

Recommended storage capacity:
Sixteen 8,192 word two microsecond access storage units (instead of four such units). Total: 131,072 words.

2. The high speed (.5 microsecond access) storage capacity (1,024 words) is inadequate.

a. Storing instructions in the two microsecond access units causes the machine to operate at 1/3 to 1/10 its potential speed.

b. Few programs will fit in the proposed 1,024 words of .5 microsecond access units. Thus, a large portion of many programs will operate the machine at a small fraction of its top speed.

c. Dr. Kolsky of Los Alamos states that it is likely in future problems that hydrodynamics calculations which require 10,000 701 instructions may appear simply as subroutines.

Recommended .5 microsecond access storage capacity:
Eight 512 word units (instead of two such units). Total capacity: 4096 words.

3. Facilities for removing large volumes of information from the machine for later re-entry are inadequate.

a. Twenty minutes are required to dump 2,000,000 words from disc storage onto a 727 tape unit. Thus, when examination of certain printed information is necessary at some intermediate stage of a calculation before continuing, as much as 40 minutes may be required to dump and re-enter the appropriate information.

b. During the process of unloading one partially completed problem onto 727 tape, the machine may be otherwise useless if the next problem requires the use of the disc storage being unloaded.

c. Los Alamos anticipates frequent occurrence of the situations referred to in a. and b. above.

Recommended facilities for removal and re-entry of information into the machine:

Two magnetic tape units (instead of none) capable of recording or delivering more than 15,000 words per second.

4. Facilities for providing initial input information and recording final output at speeds commensurate with that of the machine are inadequate.

a. Loading a 180 x 180 matrix and recording its inverse on 727 tape requires 40 seconds. Inverting the matrix requires 20 seconds.

b. In the 704 two multiplications can be performed in the time required to enter one word from 727 tapes. In the Stretch machine, 800 multiplications can be performed in the time required to enter a word from 727 tapes.

Recommended facilities for providing initial input and final output at appropriate speeds:

Auxiliary equipment for recording input information on the high performance tapes recommended in item 3 above and for printing or preparing 727 tapes from information on the high performance tapes.

5. The ultra high speed (.2 microsecond access, transistor) storage capacity (16 words) may be inadequate.

a. The rate of flow of instructions from the .5 microsecond fast storage is frequently not quite adequate to operate the machine at full effectiveness. Therefore, it is often necessary to use ultra fast storage for all index quantities and intermediate results in order to obtain these quantities at the proper rate

without interfering with the flow of instructions.

Recommended ultra high storage capacity:
Sixty-four .2 microsecond transistor registers
(instead of sixteen such registers)

6. In general, the greatest weakness of the Stretch computer as proposed is its lack of adequate random access storage capacity. An increase of such capacity to 131,072 words is recommended because this is considered the largest economically feasible capacity obtainable using present methods of constructing random access storage devices. However, the study group wishes to emphasize in the strongest possible manner that it considers random access storage capacities in the neighborhood of 1,000,000 and more words essential to a well balanced machine with the speed of the Stretch computer. It seems clear that any proposal to Los Alamos offering random access storage capacity of about 1,000,000 words and any reasonable machine organization stands an excellent chance of being accepted even at increased cost.

It is suggested that a stong program in IBM to develop economically feasible 1,000,000 word random access storage devices would be of inestimable benefit to all areas of our future EDPM business: scientific, industrial, and commercial data processing. It is further suggested that IBM give careful consideration to indicating in its proposal to Los Alamos and any similar proposals its intention to embark on such a program for developing very large random access storage devices.

In order to supply information to the arithmetic unit of the Stretch machine at an adequate rate, the units of a very large random access storage require individual access times of less than ten microseconds. For certain classes of problems, multiplexing techniques of the look-ahead type would fail to provide an adequate flow of information unless the access time to an individual unit of storage is less than five microseconds. Increases in the speed of the arithmetic unit beyond those now proposed would require corresponding decreases in the access time to each unit of storage if the arithmetic unit is to be kept reasonably busy.

Below are listed some of the considerations which indicate the increasing importance of very large random access storage devices for data processing machines of the future.

1. A large proportion of even the problems of today in the scientific, industrial, and commercial areas require storage capacities in excess of 1,000,000 words. Most of this storage capacity today is necessarily in the form of non-random access storage such as tapes, drums and disc storage.

2. We are encountering many difficulties with the non-random access properties of these large capacity storage facilities. The frequent necessity to sort information stored on tapes and the slowness of the sorting process is an indication of the need even today of large random access storage.

3. Certain present day problems are such that solutions are completely not feasible unless the data can be stored in random access storage. Examples: (1) obtaining the roots of matrices. (2) Monte Carlo calculations. It is not unreasonable to assume that the higher speeds of future machines will make possible the solution of much larger problems of these types, thereby requiring random access storage capacities in excess of 1,000,000 words for their solution. Certain other types of calculations are such that they may require very much more machine time when non-random access storage must be used for data than the time required when adequate random access storage is available. Linear programming problems and other scheduling calculations are examples in the area of industrial problems. Certain file maintenance and sorting jobs are examples in the commercial area. In the scientific area an example which may very well constitute a large portion of the calculations performed at Los Alamos is that of the solution of hydrodynamics problems over a network of points whose neighbors change during the calculation.

4. The major time and cost involved in planning the solutions of most large problems is associated with planning the organization and transmission of information in non-random access storage. Planning the efficient use of this auxiliary storage is correspondingly the most difficult, if not impossible, task to have performed by an automatic programming system.

5. The best non-random access storage device now contemplated for the Stretch machine, the RAM with 6 microsecond per word transmission rate, cannot supply information at an adequate rate to keep the arithmetic unit busy for a very significant class of problems, those involving large matrices. In matrix multi-

plication where neither matrix fits the random access storage, the calculation proceeds at half speed.

6. The Stretch machine and other future machines will produce results at a rate which prohibits their complete human inspection. Thus, future computers must be able to locate interesting or exceptional results for detailed human examination. Their ability to do so will depend almost entirely on the ability to rearrange information rapidly according to a variety of criteria. Such an ability to rearrange implies large random access storage.

Copy 18 Kelley

Griffith

A SYSTEM DESCRIPTION

OF

PROJECT STRETCH

December 29, 1956

COMPANY CONFIDENTIAL

Data Processing Systems Planning:

F. S. Beckman
J. C. Gibson
R. Goldfinger
J. E. Griffith
B. L. Sarahan
D. W. Sweeney
I. Ziller

I. Introduction

This report describes, in summary, the presently accepted form of the Stretch computer. The form of the machine, as herein described, is the result of a rather dynamic convergence of opinions about what is desirable and opinions about what is feasible. Since these two sets of opinions do not always converge, the form of the machine must change in order that divergences are changed into convergences. The overall effect is thus to force a general convergence toward the form of the machine as it will be. The form described here will be subject to violent fluctuations in the future, for new hardware possibilities and opinions about what is basically desirable must change the form as much as possible, in order that the rate of convergence be as high as possible.

As is presently conceived, the Stretch computer will be made up of two computers. These computers are complete in themselves and capable of independent or integrated operation. The reason for this is that it allows a certain degree of independence in the planning and design of the system, thus simplifying some of the concepts of a system of such advanced capabilities. The principal difference between the two computers is that one is slower in operational speed than the other. The slow machine will be about 10 times as fast as the IBM Type 705 system and is designed for both commercial and scientific problems. The fast machine will be about 100 times as fast as the IBM Type 704 system, and it is slanted toward the specifically scientific or technical area of computation. It is this machine which will satisfy, for the most part, the AEC contract with IBM.

This report will describe the two machines separately, but it is expected that there will be many overlapping concepts in the two. The areas of overlap will serve to make the joint use of the two machines as practical as possible.

12/29/56

II. The High Speed Machine

A. Introduction

The High Speed Machine is composed of three basic units: memory, arithmetic unit, and decoder, all three interconnected by a bus system with appropriate input/output connection.

Present day opinion has established the feasibility of producing a memory of 2.0 usec. full cycle time in large enough quantities to be used in a machine such as is contemplated. However, in order to have computing speeds that are 100 times faster than the 704, it is necessary to have a basic arithmetic speed of the order of one microsecond per operation. This speed is somewhat incompatible with the 2.0 usec. memory cycle, as it implies that the arithmetic unit can chew up data twice as fast as the data can be obtained from the memory. There is also another problem: the instructions must also be procured from the same memory as the data, offering another block to the goal of getting a computer 100 times faster than the 704. We therefore appear to be in a quandry; the arithmetic speed of 1.0 usec is too fast for a machine with 2.0 usec. memory; it appears that the overall speed of the machine will be limited by the many necessary references to the 2.0 usec. memory.

The solution to this problem lies in the possibility of multiplexing the various units of the machine so that more than one unit can be operated at one time. In fact, this appears to be the only way of operating slower units so that they can keep up with faster units. The unit size of the 2.0 usec. memory aids this idea, for the memory will come in blocks of 8192 words, each block with its own memory register, so that the blocks may be multiplexed by adding sufficient control hardware. It can be shown that this can be done and that, on the average, the delay due to 2.0 usec. data references

can be made sufficiently small to be ignored.

However, there is still a problem; if instructions as well as data are stored in the 2.0 usec. memory, there will still be a delay due to the many instruction references that are necessary. Due to the ordered manner in which these references are taken, multiplexing does not offer an adequate solution.

For the Stretch machine, this problem will be ~~alleviated~~ alleviated by the introduction of the notion of a memory hierarchy, that is, a memory made up of units of various speeds. Specifically, a special memory with a cycle length of 0.5 usec. has been introduced as a special instruction storage. This memory will be much smaller in quantity than the 2.0 usec. memory, ~~for~~ because Instruction the ~~First~~ Memory (as the 0.5 usec. cycle memory is known) is more expensive and only enough is needed to contain the program.

Another even faster (0.2 usec. cycle) memory is included to serve as a block of index registers. This memory allows the indexing process to proceed at a very high rate and thus contributes to the overall speed of the machine by lessening the time needed to prepare the final effective address.

Thus, the computer has a memory made up of units of 3 different speeds - 2.0 usec., 0.5 usec., 0.2 usec. It is assumed that any combination of these may be specified for a particular installation.

The decoder is the unit which is concerned with obtaining and indexing instructions, multiplexing the various memories with each other and the arithmetic unit, and executing the operation called for in the most efficient manner. Due to the discrepancy between the 2.0 usec. memory cycle and the 1.0 usec average arithmetic cycle, it is necessary to buffer the decoder with two or three registers which contain indexed instructions ready for execution. The relative slowness of the memory can

be discounted by a buffered decoder, and the decoder will, as a general rule, allow the arithmetic unit to operate as if it were looking at a 1.0 usec. memory. If the speed of the 2.0 usec. memory could be increased to 1.0 usec. cycle time, the buffering could be excluded from the decoder with a small increase in overall performance of the machine.

Two new philosophies have been applied to the multiplexed operation of the machine: one, the asynchronous operation of the various units of the machine, including the memories, arithmetic unit, and input/output equipment; two, the non-sequential execution of the instructions to provide for a very high simultaneous use of the various units of the computer. These philosophies are manifested in the design of the decoder which serves as the central control for these simultaneous operations.

The arithmetic unit is a completely self-contained unit which will operate in either fixed or floating point notation. This arithmetic unit retains the original operands until the correct answer has been produced. Also, the arithmetic unit requires only as much time as is necessary to produce the correct answer; this means that the actual time needed to produce the correct answer depends on the operands themselves; in particular, the number of binary ones which make up the numbers determines the time required. This feature is particularly valuable in an asynchronous machine, for it allows the overall speed of the machine to vary with the size of the operands used. Thus, if the average operand is "small" in size, the computer will operate at a higher speed than if the average operand is "large" in size; also, the adjustment in speed is continuous and automatic, and the decoder takes advantage of any variations in speed.

12/29/56

At this point, we will now begin the description of a machine that is founded on the three basic units described above.

III. Information Flow

A. General

The computer is basically a bus system to which is connected the memories, decoder, arithmetic unit and I/O. Appropriate control is provided to switch each of these units to the bus at the proper time.

B. Memory

There will be, at most, memories of three different speeds. The two higher speed memories are necessary to account for the difference in the speed of the slowest (2.0 usec. - Data Memory) and the average operational speed (1.0 usec.) of the arithmetic unit. The memories are as follows:

a. Data Memory - 2.0 usec cycle

b. Instruction Memory - 0.5 usec cycle

c. Index Memory - 0.2 usec cycle

Since each type of memory is for a special purpose, the speed is specified accordingly.

1. Data Memory - 2.0 usecs. This memory will come in blocks of 8192 words, and each block will have its own memory register and memory address register. This type of memory may be increased in blocks of 8192 words and the addressing will be sequenced such that successive addresses will be in adjacent blocks of memory. In case an odd number of blocks is attached to the machine, one block may contain 8192 successively addressed words. The machine addressing system will provide for direct addressing of a location in a memory of the order of one million words,

12/29/56

and therefore, the data memory of the machine may be built up of 8192 word blocks until the maximum size is reached.

Since all memories are composed of 64-bit words, they are all compatible and any memory may be used for any type of storage such as data, instructions and index quantities.

2. Instruction Memory - 0.5 usec. This memory will be available in blocks of 512 words. It is expected that this memory will be used principally for instructions, but it can be used for any type of information desired. The addresses of locations in this memory will fall in the low order range of the one million locations provided in the address structure of the machine. It is expected that the maximum amount of Instruction Memory that can be installed on any one machine will be about 4000 words. It is probable that the addressing sequence will be interleaved across adjacent memory blocks as in the case of the Data Memory. However, if the timing is propitious another arrangement may be used; successive addresses will be placed in one block thus allowing a sequential placement within each memory block.

3. Index Memory - 0.2 usec. This memory will be available in blocks of 16 words. This memory is intended for use as index register storage and as auxiliary accumulators. It will be connected to the bus as an adjunct to the Data Memory and the Instruction Memory. The addressing sequence will be placed in the lowest-order range of memory addresses. Thus, the memory addressing sequence will start with the Index Memory, proceed without any gaps into the Instruction Memory, and finally extend into the Data Memory.

4. Address placement. It has been arbitrarily decided that the Data Memory will start at location 4096 and extend as far as necessary up to a maximum of 1,048,576.

12/29/56

The Instruction Memory will occupy those location from 4095 downward. It is assumed that a maximum of 4095 words of Instruction Memory will suffice. If any customer wishes more than 4095 words of Instruction Memory, the ~~dividing~~ dividing line may be moved up to 8192, but this should be done only as a last resort.

The first 4096 words of Data Memory are assumed to be unavailable, for these addresses are occupied by the Instruction Memory and the Data Memory. There is, therefore, the implication that the first 4096 words of Data Memory must be reserved for other uses, even if all of the address are not used. It is possible that there will be an engineering solution to this problem such that the loss is virtual and not real.

C. Memory Bus

The bus system in the Stretch computer will be, in a conceptual sense, one common, bidirectional system which is common to all memories, the decoder, the control arithmetic unit, the arithmetic unit, the I/O Exchange and, if necessary, the I/O Computer. Figure 1 gives a schematic view of this system:

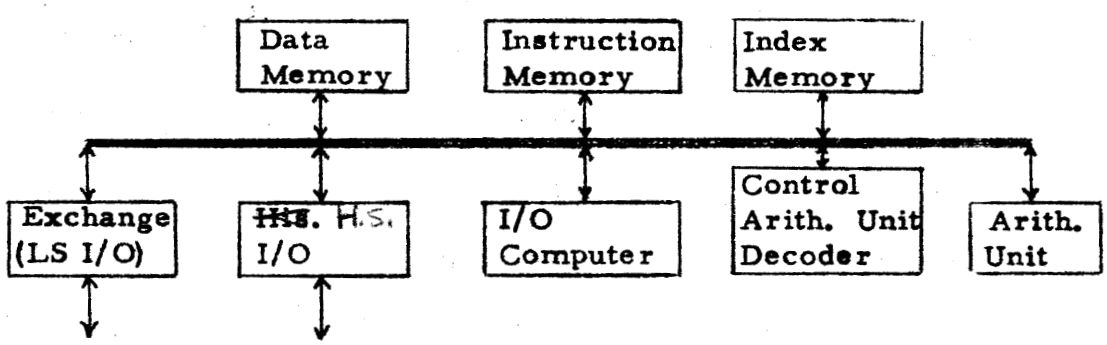


Figure 1

12/29/56

This system involves a bus which is common to all basic units thus allowing the transfer of information between any two units during any 0.2 usec. period.

The actual bus system, as finally designed, may include more than one bus as shown above, but the final decision must depend upon consideration of engineering feasibility and overall machine performance. For instance, in order to achieve the desired performance on a set of typical problems, it may be necessary to have a separate bus for transmitting instructions from the Instruction Memory to the decoder.

D. Instruction Flow

1. Instruction Counter. This counter will denote the location of the last instruction called from memory into the Instruction Register. This counter will be available for modification, and its contents may be read out at any time into the memory, Control Arithmetic Unit or Main Arithmetic Unit.

2. Instruction Register. This register receives the instruction from the memory register. All indexing and other address modification takes place in this register. A special Control Arithmetic Unit is associated with this register to perform the arithmetic required for indexing operations. The Instruction Register has associated with it a decoder which will decode and execute instructions which do not involve the Main Arithmetic Unit.

Instructions which refer to the Main Arithmetic Unit proceed on into the Decoder after their addresses have been properly modified. No address modification of any kind is possible in the Decoder itself.

12/29/56

3. Control Arithmetic Unit. This unit comprises a 26 bit accumulator with sign control. It is capable of addition or subtraction of signed 26 bit ¢ numbers in integer form. The normal address will be a 20 bit signed number, and any arithmetic operations will be accomplished in the high order (left hand) 20 bits. The low order six bits are for use in bit addressing wherein the address of any bit in memory may be represented by a word address of 20 bits and a bit address (of any bit within the word addressed) of six bits.

Multiplication operations may be performed on 10 bit integers but at a loss in time.

The normal mode of use will be such that the operation of the Control Arithmetic Unit will be specified by suitable modifiers in the instruction itself. However, a set of special commands will allow the unit to be programmed in a manner analogous to the Main Arithmetic Unit to handle cases not included in the instruction format. For example, the operations of multiplication and division in the Control Arithmetic Unit may be initiated only by one of the set of special commands. Shifting in the Control Arithmetic Unit will also not be possible except when specifically instructed.

4. Decoder. This unit consists of a decoding register and several buffer registers. The normal flow of instructions will be from the Instruction Register, where they were indexed, into the first Decoder buffer register, thence into the second ~~Decoder~~ decoder buffer register, thence into the third - - - etc. until the Decoder Register is reached. The number of Decoder buffer registers is dependent upon the relative speeds of the components in the machine and the desired degree of performance. In certain cases, it is possible to initiate a reference to Data Memory from some of

12/29/56

the buffer registers, but this action is dependent upon the instruction contents of the buffer registers and the Decoder Register and is not a feature that can be accounted for or used by the programmer.

The original location in memory of each instruction is carried along with it in the buffer and Decoder Register, and is lost only when the execution of an instruction has been completed and no automatic breakin actions are required.

5. Execution Control. The execution of the instruction will proceed until the end-of-execution signal is given at which time the option of breakin must be examined. If no breakin is possible or specified, the next instruction may proceed If breakin is taken, the next instruction may not be initiated until a special trigger is set by the program.

E. Data Flow

The data being transmitted to or from the Main Arithmetic Unit will always pass thru the S register which serves as a connection between the bus system and the Arithmetic Unit proper. If the data being transferred to the Arithmetic Unit is intended for any register other than the S register, then the data must be transmitted from the S register to the other register. Figure 2 shows the layout of the Main Arithmetic Unit and illustrates the registers which are available to the programmer.

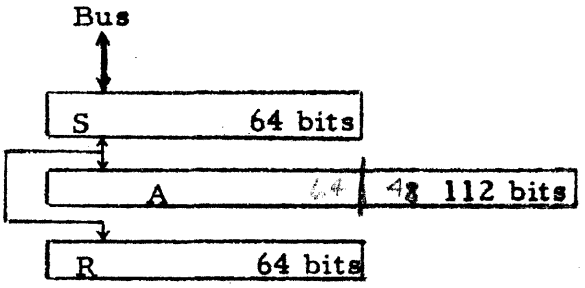


Figure 2

12/29/56

The S register is 64 bits long and serves as an operand register for arithmetic operations and also as a termination for the bus system. The R register is 64 bits long and is used to contain the remainder in divide operations. The A register is 112 bits long and its excess length is one mantissa length exactly so that floating point operations may be extended to produce double precision mantissas.

The Main Arithmetic Unit also includes a 96 bit parallel adder to produce double precision answers for the normal floating point mode of operation. In addition, there will be a shifting matrix included in the Main Arithmetic Unit.

This shifting matrix will be capable of shifting any group of adjacent bits to any other position in any other register. It will also be possible to shift any group of bits to any other position in the same register. There will be, in addition to shifting operations, appropriate masking facilities for both the transmitting and receiving register and optional control for all of these features.

Suitable checking hardware will be associated with the Main Arithmetic Unit such that all answers produced by arithmetic operations will be checked and corrected if possible. This implies that the original operands must be retained until a correct answer is produced, and thus there may exist some registers in the Main Arithmetic Unit which are necessary for producing checked answers. These, too, will probably be made available to the programmer if they exist, but the use of these registers will be severely limited by the implications of their principal use.

II. Word Format

There will be, of necessity, several different word formats in the Stretch Computer. This is necessary in order to provide flexibility and high speed performance. One may design a machine with only a few formats, but the invariable result is loss

of speed in some applications. In order, therefore, to provide a high level of performance in all problem areas, several word formats will be available in order to facilitate the use of the inherent speed of the machine. There will exist areas for which there is no optimum format, but if the chosen formats are properly compromised, any area may be handled without a serious sacrifice of performance. Since there may be only a limited number of different formats in a machine such as the Stretch, one wonders how to guarantee adequate performance in those cases for which no specific format is given.

One solution commonly taken is to eliminate the problem by providing a general format which includes most all possible cases. This solution provides a format filled with various modifiers which are used to define the format for any given operation. This approach is commonly known as "Microprogramming." When implemented, this solution usually requires a longer word length than would any other method and since some modifiers are not used in many operations, the space reserved must therefore be wasted. In order to gain the extreme flexibility of format and use, it is generally felt that the wasted space is not too serious a disadvantage.

However, there is another solution to the problem of providing flexibility and high speed, and this solution appears to be more consistent with the fundamental nature of a stored program computer than any other solution heretofore proposed. In order to understand it, however, it is first necessary to define what is meant by the "fundamental nature" of a computer.

The most essential characteristic of a stored program computer is that it can be made to look like any computer that we please (within very broad limits). This characteristic was unknown before the advent of stored program machines, (although

Turing proved in 1935 that such a machine could be made) and it is the principal reason why the power of stored program computer staggers our wildest flights of fancy. The ability to look like any other machine is a property hitherto unknown of any machine, and it is found to be the property a very large class of machines which we call computers. In actual practice the only limitation of any computer is the time required to perform the job. By this, we mean that any decent stored program computer may be programmed to look like any other computer we please, but there may be a loss of efficiency such that the simulated machine is too slow for any practical use. The limitation, therefore, is one of time. In other words, the computer is limited to simulation only of those machineswhose speed is fast enough to be useful.

Thus, the problem of designing a computer which can "look" like the broadest possible class of other machines is one of exploiting the hardware of the given computer to the fullest. The consideration of this problem turns out to be the consideration of a subject we now call "editing." Editing refers to the process of mapping information from one format into another.

In the past, the usefulness of having more than one format has been recognized and all stored program computers have been built with at least two formats (instruction and data) so as to save editing operations, thereby effectively increasing the speed of the machine. In time, it was recognized that having more than two formats would allow a given computer to simulate a larger class of useful machines, but adding formats costs hardware and money, thereby decreasing the application potential as much as slowing down the computer. The microprogram approach is to provide a large generalized format with modifiers to convert it into any of a very large class of formats is still too small in many cases; also, it may require more hardware

than does the case of few, but fixed, formats. In any case, the microprogram solution is clearly an extension of the fixed format philosophy and suffers from the same defect; that the fundamental nature of the stored program computer is not being exploited.

The Stretch Computer will incorporate a philosophy which is different from that in any presently existing or proposed machine. That is, it will manifest as an important part of its character the ability to simulate a much greater variety of computers than is possible by any other means; namely, it will have as a specific design feature the ability to operate in the interpretive (simulative) mode. The key to this ability is, as mentioned previously, the ability to edit in a fast and direct manner. The principal reason that present day computers run slow in the interpretive mode is that their ability to edit in a generalized manner is very deficient. By generalized editing, we mean the ability to map (transform) information in any format whatever to any other format, usually the machine's own format.

Present day computers spend by far the greater time in the interpretive mode editing instead of performing useful work. For certain often-occuring special cases, such as floating point arithmetic, special formats are defined, but no real effort has been made to facilitate the use of formats not foreseen by the machine's designers.

Recognizing that the most innate characteristic of a computer is its theoretical abilit to simulate any other computer desired, the Stretch computer will, therefore, contain the ability to operate in the interpretive mode at a speed much more commensurate with the speed of its components. This philosophy will, in particular, be of great help in applying the techniques of automatic programming to the use of large computers.

Therefore, the Stretch machine will have a limited number of word formats, but it will contain special instructions to facilitate the definition of, and operation with, an infinitely large number of other formats (these special instructions will be covered later with regard to logical operations and editing). Since the machine's speed in the

12/29/57

interpretive mode will be greater, relatively, than any present day computer, the Stretch Computer will be able to simulate a gigantic variety of other computers. Whereas in present day computers multiple formats are built in as an attempt to sidetrack the necessity of editing, the Stretch philosophy recognizes that the ability to edit is fundamentally necessary to any computer, and therefore, every attempt will be made to provide this facility in a form which recognizes that its importance is greater even than the basic arithmetic operations!

The following set of formats have been selected as a basis for carrying out the aforementioned philosophy; the set of four formats includes one instruction format, one index word (location) format, and two data formats.

A. Instruction Format. This format is composed of six fields as shown in Figure 3 and several interpretations are possible

x	operation	AI	Address	E	T
1	12	6	21	~~21~~ 12	12

Figure 3

for any given case. The fields and their respective uses are as follows:

Field

X - 1 bit - This is an unassigned bit for the programmer's use, and had no assigned machine function.

Operation - 12 bits - This field will contain the operation code and any operation classifiers needed.

Address Interpreter - 6 bits - This field will denote the interpretation to be taken on the Address, E, and T Fields.

Address - 21 bits - This field will contain a 20-bit address and a sign to facilitate a generalized form of indexing arithmetic.

12/29/56

Extension - This 12 bit field may serve as an extension of the Address field (for bit addressing) or as an Index Tag. The exact interpretation will be controlled by the AI field.

Tag - This 12 bit field will normally serve as an Index Tag, but may have other meanings as controlled by the AI field.

The exact interpretation of any combination of the Address, Extension, and Tag fields will be specified by the contents of the Address Interpreter field. The AI field will allow a maximum of 64 different interpretations to be taken on this particular format. In addition, the classifiers in the operation code will allow for many classes of operations such as floating point, fixed point, etc.

B. Index Word Format. The Stretch Computer does not have index registers as such. Instead, provision will be made for using any memory location as an index register. This will provide as many index" registers" as is convenient with a given memory size. The various speeds of memories will control the speed with which any given indexing operation takes place. The Index Memory is a special high speed memory intended for use in those cases where high speed indexing is desired and will effect a reference in 0.2 usec. It is not necessary to use Index Memory for indexing operations, and thus the Instruction Memory or Data Memory may be used instead for indexing operations. The only difference will be in the resulting speed of indexing operations.

The format of an Index Word will be as shown in Figure 4.

S	Compare	Add	Increment	Base
1	21		21	21

12/29/56

The field will be as follows:

S - 1 bit - unassigned at present.

Compare Address - 20 bits + sign - This field will contain an address which may be compared automatically with the Base Address. An alternate interpretation of this field will split it into two parts, a 10 bit count field and a 10 bit + sign count increment field. The count field may be automatically incremented with the contents of the Count Increment field. These two interpretations will be specified by the instruction itself.

Increment - 20 bits + sign - The contents of this field may be automatically added to the contents of the Base field upon command.

Base - 20 bits + sign - This field contains the address which will normally be added to the contents of the Control Accumulators as part of an indexing operation.

It should be noted that all fields in an Index Word are signed quantities and that all additions and subtractions performed as part of indexing operations are therefore algebraic operations. This facility allows addresses to be increased or decreased by complementary programming techniques. The operations performed on Address fields in an instruction itself are also algebraic and thus the Address field of an instruction is signed to permit a consistent scheme of indexing. However, all actual addresses sent to Memory Address Registers are interpreted as absolute values only; there are no signs which refer to memory locations.

C. Arithmetic Data Format - This is the standard format for data to be operated upon by the Main Arithmetic Unit. Figure 5 shows the fields which make-up this format.

P	S	Exp	Mantissa
4	3	9	48

Figure 5

Programmers' Bits - 4 bits - These four bits are reserved for use by the programmer. They are assigned for any automatic action.

Sign Field - 3 bits - This field contains the sign of the exponent, the sign of the mantissa, and an indicator bit to indicate numbers not representable in floating point notation, such as zero.

Exponent - 9 bits - For floating point arithmetic this field will contain the exponent of the number. The exponent range is $10^{\pm 152}$. For fixed point arithmetic, this field is left blank and is ignored by the Main Arithmetic unit.

Mantissa - 48 bits - This field will contain the mantissa of floating point numbers and the fractional representations of fixed point numbers. The interpretation of this field will be specified by the command itself.

The same format is used for both fixed and floating point arithmetic, but the interpretation, as specified by the instruction, alters the meaning of some of the fields.

D. Logical Data Format - This format is, in reality, no format at all. It is somewhat analoguous to the zero of an arithmetic system, and is included as the null class of formats. This format, defined in Figure 6, complements

64

Figure 6

the set of all other formats and is one of the most important of all formats, for it

serves as a blank format which the programmer may use to define his own format for a particular application. Since none of the bits are assigned to any give purpose, the programmer may, by interpretive coding, define his own format and use it to construct a language of his own choosing.

This format may be altered in two ways; by fixed point (48 bit) arithmetic and by logical manipulations upon the set of 64 bits. The instruction set will include suitable commands for efficiently performing any alteration desired by these means, thus carrying out the philosophy of providing means for efficiently operating the machine in the interpretive mode.

12/29/56

The Instruction Word

A general form of the instruction word has been proposed. This form is modified to include specific sets of instructions, which require the specification of varying amounts of information. The different portions of the general form of the instruction word perform somewhat different functions for certain sets of operations. A specification of various types of indexing also requires modification of the general form. This section will define the general form of the instruction words and specify the modifications required. A separate section will define the indexing modes applicable to this computing system.

A. General Format

The general form of the instruction word is as follows:

Field No.	1	2	3	4	5	6
Field Size	1	12	6	21	12	12

In this general form, the meaning assigned to the various fields is as follows:

Field No. 1 - Not used, available to programmer

2 - Operation Code

3 - Address Interpreters

4 - Word Address

5 - Tag Address #1

6 - Tag Address #2

This form provides for the usual mode of operation of specifying an operation to be performed and the address of the location of the data, with the provision of specifying the location of two indexing quantities which are used to modify the address

12/29/56

prior to the execution of the operation. Each of the fields is described in more detail in the following paragraphs.

Field Number 1

At the present time, one bit has been placed in the "not used" category. This bit location might be used by the programmer to signal special action either through subsequent programming or through operation of an automatic break-in feature.

Field Number 2 - Operation Code

The bits of the operation code field will define the operations which the machine is expected to perform. This field will probably contain certain bit locations which specify the various classes of instructions. The class of an instruction may indicate a variation in the interpretation of the instruction format. Other bits of the operation code might be used to indicate variations within the operation code itself, e.g., one bit within the Arithmetic Class of instruction might differentiate between fixed point and floating point operations. The bits of the operation code should be used in a manner which is most advantageous to the machine decoding of that operation. The specific operations which will be required for the Stretch system are defined in a later section of this report.

Field Number 3 - Address Interpreters

The six bits of this field are used for the specification of the manner in which the Word Address and the two Tag Addresses are to be used. The assignment of meaning to these bits will be dependent upon the engineering considerations involved in the design of mechanisms to handle the different modes of operation.

Field Number 4 - Word Address

The twenty-one bits of this field are used to specify in a binary number system a sign and a twenty-bit address of a memory location. Provision is thus made for

directly addressing somewhat more than a million words of memory. A sign is attached to the word address to simplify address modification. This bit has no meaning in the interpretation of the address.

Fields Numbers 5 and 6 - Tag Addresses #1 and #2

Each of these fields contains twelve bits. The twelve bits permit direct addressing of each machine register and each of the 0.2 us and 0.5 us memory locations. The normal use of these addresses will be to specify index registers for the modification of the word address. Any of the faster memory locations may be used as an index register within the general instruction form.

B. Memory Location Addressing

In most one-address instructions, the programmer is free to specify an operation code and the location of the data to be used in this operation. Provision is made for modifying this address by indexing. Two other modes of addressing are included in the specification of the Stretch system. These modes are immediate and indirect addressing. The usual mode of addressing will be referred to as direct addressing. In direct addressing, one specifies that this operation be performed on or with the contents of some memory location.

Immediate Addressing

Whenever constants in a direct addressing scheme are used arithmetically or in the modification of instructions, it is necessary to refer to memory to obtain the constant data word. In an immediate addressing mode, the bits of the Word Address (including the sign bit) of Field Number 4 are used as the data in the operation specified. The use of this addressing mode will provide considerable flexibility in the modification and manipulation of instructions or data.

Indirect Addressing

In indirect addressing the contents of the word at memory location X are used as the address of the word whose contents are to be used as data in the operation specified. In this mode of operation, the indexing specified is performed. The location specified by the modified indirect address may specify a word which again contains an indirect address. The process, that is, the search for an immediate or direct address, will continue until an address which is not indirect is encountered. This mode of addressing is useful in providing links between programs and in the specification of the location of data for sub-routines.

Bit Addressing

In many cases it is desirable to interrogate the condition of a bit location in memory. This bit location might be a single unit of a selector register. In these cases, it might be necessary or desirable to allow modification by the contents of two index registers. Some combination of bits in the Address Interpreter Field (Field Number 3) will specify that the address be interpreted as a fourteen-bit word address and a six-bit bit-position address. This type of address representation implemented by indexing will provide direct addressing of each bit within a million word memory. This mode of addressing also assumes a difference in the form of the index register, which difference will be described in a later paragraph.

C. Functions of the Tag Fields

Some comments concerning the function of the Tag Fields have already been made. The most straight-forward use of these two fields is for the specification of two index registers whose content values are to be used successively to modify the word address of the instruction. This use of the Tag Fields is useful in those cases where the word address is either direct or indirect. There may be some reason for

allowing indexing of immediate addresses, but the exact definition will be dependent upon programming examples. The following paragraphs will describe various other uses of the Tag Fields.

Two Address Operation with Single Indexing

The majority of instructions will likely require one-dimensional indexing. In these cases the index register to be used is specified by the contents of Tag Field #1. The following uses under these circumstances have been assigned to Tag Field #2.

a. Pre-Store - The contents of the receiving register (S) are stored in the location specified by the contents of Tag Field #2 following the Word Address indexing by the contents of the location specified by Tag Field #1 but prior to the execution of the instruction. Thus, a number taken from a 2.0μs memory location can be stored in a faster memory for future use during the execution of the first instruction in which the number is involved. It should be noted that this can be thought of as an exchange type instruction on the usual store class of instructions. For example, the original contents of a memory location are brought out and placed in a fast memory location and then the instruction is completed by the storage of the particular accumulator or register specified.

b. Post-Store - The contents of the high-order result register (A) are stored in the location specified by the contents of Tag Field #2 on the completion of the operation. The contents of Tag Field #1 specify the location of the index register to be used in modification of the word address. This mode is particularly useful in the storage of temporary results.

c. Limited Two Address - In this case, the use of Tag Field #1 is used to specify a normal indexing operation. The contents of the location specified by Tag Field #2 are transferred to the high-order result register (A) prior to the start of the operation. The low-order result register (B) may or may not be cleared, depending upon the specification mode. At the completion of the operation the contents of the high-order result register (A) are stored in the location specified by Tag Address #2. This mode permits a limited form of two-address operation.

[Actual two address]

Combined Use of Tag Fields

Many cases will require that index and other address modifications be performed using operands outside of the low 4,096 memory locations. In such cases, it is desirable to combine the two Tag Fields to form a second twenty-one bit word address. Three of the available twenty-four bits are not used. The uses of this second word address are:

a. Immediate Address - The signed number specified by the twenty-one bit second word address is combined with the normal word address to obtain a modified address for use with the operation code. This permits the indexing quantity to be carried within the instruction itself.

b. Indirect Address - The signed number specified by the twenty-one bit second word address is treated as an absolute value. This quantity specifies an indirect index address, that is, the

quantity specifies the location of the actual Tag Address to be used for indexing.

c. Direct Address - The signed number specified by the twenty-one bit second word address is treated as an absolute value. This quantity specifies the location of the index quantity to be used in modifying the word address. This mode of operation permits the use of any memory location within the computer system to be used as an index register. The use of a 2.0 microsecond memory location as an index register will obviously cost some time in the execution of the instruction.

Variable Field Operations

One of the most used logical operations involves the packing and unpacking of pieces of information of less than a word length into or from word length forms which can be handled by the system. This is normally done through the use of a mask or filter, which allows certain bit channels to be operative and the remainder to be non-operative. Because all of the instructions refer to full words, the actual instructions used to specify the operation of packing (for example), have little relation to the operation being performed. It is proposed that a limited number of operations (on the order of five or six) be provided with a variable field length type of operation.

In this mode of operation Tag Address #1 field is broken into two six-bit fields. The six-bit field nearer to the word address field specifies for this operation the starting point in the memory word, that is, information is required from the memory word beginning with the bit position specified. The second six-bit field specifies the number of information bits required. On these special instructions part of the operation code specifies the starting point in the accumulator. Indexing on a bit - basis is specified by Tag Address #2.

12/29/56

As an example of this mode of operation, the instruction

Clear and Add 25, + 17986, 60, 6, 0

is interpreted as:

1. No indexing
2. The 6 bits of memory word 17986, positions 55 through 60 are extracted.
3. The bit in position 60 is used for sign control.
4. The bits in positions 55 through 59 are placed in positions 21 through 25 respectively of the high-order accumulator register.

This example illustrates the need for specifying operations which include manipulation of either signed or unsigned quantities. It is expected that this type of operation will be extremely useful in assembling instructions.

A minimum list of operation codes would include Clear and Add, Add, Load, Store, and Unload.

Many of the advantages of this type of operation can be secured on other operations. In these cases, the starting point in the accumulator is assumed to be the low-order bit position (or the sign position). The word address and the two tag fields are used in the manner just described. Thus, the information field in the memory word is defined. Information sufficient to this specified length is provided to or taken from the accumulator during a given operation. This mode gives direct bit addressing of any bit within a million-word memory system. Indexing is based upon a combined word and bit address, and the form of this indexing is described in the next section.

There are additional implications of this mode of operation when one considers the use of any register as an addressed memory location. Many shift instructions are automatically made available by the commands associated with variable field length.

For example, the instruction

LOAD 14, + Address ACC, 15, 8, 0,

accomplishes a one position left shift in the accumulator of an eight-bit field without change of sign. A special operation might be required to set unused bits on left and/or right to zero during a simulated shifting operation.

The specification of field length is restricted to a maximum of 64 bits. The specified field of information may however, be contained in sequential bit positions of two adjacent words.

Comment on Word Addressing

All internal arithmetic and control registers and counters will be individually addressable. Each of these addresses will be in some normal addressing sequence, but this sequence may not necessarily be dense. The 0.2 us, 0.5 us and 2.0 us memory locations will be assigned addresses sequentially within each memory type. There will be no gaps in addressing between these different memory types. All of the internal registers and counters, the 0.2 us memory locations, and the 0.5 us memory locations will have addresses below decimal 4096.

Multiple Tag Words

The general instruction form provided for double indexing, that is, the modification of the word address by the address modifier values of two specified index registers. A special word format is required for those cases of multiple-level indexing to a degree greater than two.

The instruction contains an indication in the Address Interpreter field that multiple-level indexing will be required. Under these conditions, the contents of Tag Address #2 specify the location of a "multiple tag word." The location of a "multiple tag word" might also be specified by a combined address of Tag Address #1 and #2.

The multiple tag word has a format as follows:

a. Interpreters - 4 bits

b. Tag Address #1 - 12 bits

c. Tag Address #2 - 12 bits

d. Tag Address #3 - 12 bits

e. Tag Address #4 - 12 bits

f. Tag Address #5 - 12 bits

Thus, a maximum of six index registers can be specified with an instruction and a multiple tag word. The interpreter field (a) specifies the first of the five tags (counting from the left), which is to be used. The indicated tag and all of those to its right will be used in a normal manner. Any tag fields which are zero will not enter into the indexing operation.

Additional multiple tag words may be called for through the interpreter field (a). Indicators specify that the contents of Tag Address #5 be used to specify the next multiple tag word, or that the Tag Address fields #4 and #5 be combined to specify the next multiple tag word.

Geometrical Addressing

The use of the multiple tag words for obtaining multiple-level indexing is restricted only by the fact that the tags used be of the twelve-bit type, thus restricting the index registers to locations below 4096. In every other respect this is a completely general system of indexing. The following descriptions of geometrical addressing will illustrate a second means of multiple indexing which is only a little less general than the system just described.

This form applies to those instructions in which both tag address fields are available. The Tag Address #1 is used to specify the base address of a set of twelve

successive memory locations which may be index registers. The bits of Tag Address #2 indicate in sequence the index registers which are to be used. Thus, for example,

Clear and Add + 17286 490 101100001001,

where the first two addresses are in decimal and the last in binary ~~would~~ means that the word address (+ 17286) is to be modified by the contents of index registers 490, 492, 493, 498, and 501. This form of multiple-addressing has the advantage of compactness. The changing of indicator bits is simplified by bit addressing, and the direct modification of the word address or tag address is simplified by the variable-field type instructions.

Indexing

A method to modify word and bit addresses automatically is referred to by the word "indexing." In general, a word address is modified by the value contained in the index register specified by a secondary address within the instruction. In the general form of the instruction, the modification of the word address of the contents of two index registers is possible. On some of the other modes of operation, only single indexing is provided. This section will define in somewhat more detail the functions of an index register.

A. Index Register Type #1

Any register or memory location can be used as an index register. The index register is then sixty-four bits long, rather than simply the length of the word address. The additional bits may be used to specify other ~~pertinent~~ pertinent information with respect to loop cycling and the other uses of the index register concept. The Index Register Format Type #1 contains:

a. Compare Value	20 bits and sign
b. Value Modifier	20 bits and sign
c. Address Modifier Value	20 bits and sign
d. Not Used	1 bit

12/29/56

In cases where this format for indexing is specified, the address modifier value (field c) is added with regard to sign to the contents of the instructions word address field. The result is the effective address for the given instruction.

Instructions pertaining to the index registers of this type would include:

a. Modify Value - the contents of the Address Modifier Value (field c) are combined algebraically with the contents of the Value Modifier (field b), and the result replaces the original contents of the Address Modifier Value field.

b. Modify Value and Compare - the operation described above is performed, but the result is also compared with the contents of the Compare Value (field a). The equal, high, or low trigger of the index control is turned on. The contents of these triggers may be tested or stored on a subsequent instruction.

Any of the fields may be changed or reset by means of the variable-field type of instructions described elsewhere in this report. Normally, only the Address Modifier Value (field c) would be changed, but there will be many cases where it will be desirable to change the Value Modifier or the Compare Value. Field d may be used by the programmer for purposes of indication or problem control.

B. Index Register Type #2

In some cases, it is easier to control the repetition of a given process of a counter, rather than by relating a Compare Value to the Address Modifier Value. The index register format Type #2 provides for this mode of action. The contents of the index register are:

a. Count Modifier 9 bits and sign

b. Count Value	10 bits and sign
c. Value Modifier	20 bits and sign
d. Address Modifier Value	20 bits and sign
e. Not Used	1 bit

Indexing and the modification of the Address Modifier Value by the Value Modifier is accomplished as described for the format Type #1, and presumably with the same circuits. A special instruction is required:

Modify Count - the contents of the Count Modifier (field a) are combined algebraically with the Count Value (field b) and the result replaces the original contents of the Count Value field.

A test of the count value will usually be a zero test of the Count Value field. This is incorporated in the variable field length transfer instructions. The test of the count value for values other than zero would be performed by the compare type instructions. Any of the fields may be changed or reset by means of the variable-field length type of instructions described elsewhere in this report.

C. Index Register Type #3

The use of bit addressing, either in the specialized case of a word address broken between a truncated word address and a bit address or the more generalized form described previously, requires an address modifier value of 27 bits. The value contains a sign, twenty bits for the word address modifier, and six bits for the bit address modifier. The size of this value cannot be contained within either of the two previously specified formats. The Index Register Format Type #3 contains:

a. Count Modifier	9 bits and sign
b. Count Value	10 bits and sign

c. Value Modifier 14 bits and sign

d. Bit-Address Modifier Value 26 bits and sign

e. Not Used 1 bit

The special instruction used to combine the value modifier and the address modifier value must take account of the differences in field length. This instruction is called Modify Bit-Address Value and operates similar to the instruction Modify Value. The instruction Modify Count operates as described previously. The tests of Count Value will be handled by variable field length transfer or compare instructions. The Bit-Address Modifier Value can be tested by variable field length compare instructions. Any of the fields may be changed or reset by means of the variable field length type of instructions described elsewhere in this report.

IV. Input-Output Equipment

Every effort must be made to provide a general, easily usuable method of attaching input-output equipment to the computer portion of the system. It is difficult to predict the types of equipment which will be available in 1960 (particularly, when one assumes that the equipment of any manufacturer must be handled in some way). A common control element for equipment operating at or below the 15KC range is being designed. This element is called the Exchange.

The Exchange

The Exchange is designed to handle a wide variety of units with a minimum modification of the units. The entire complement of units can be connected permanently to the Exchange. A fixed number of channels (a number less than the number of units) is provided between this switching control and the memory portion of the Exchange.

Specific input or output units are connected to these action channels under program control. Simultaneous operation on all channels is permitted. Thus, the number of input-output devices operating at one time is limited by this fixed number of channels. Lower speed units (on the order to 20 characters per second) all operate into one channel and are always connected. Approximately six input and six output units operating in the 15 KC range can be handled simultaneously with several hundred units in the low speed range. The design of the Exchange should be such that the number of input-output units required can be used to determine the size of the Exchange.

The functions of the Exchange are:

a. To interconnect specific input-output units to the available channels.

b. To assemble the bits of characters into full computer words or input, and to initiate the action to place completed words in specific locations in the main memory.

c. To take full computer words from the main memory and to disassemble such words into characters for transmission to output devices.

d. To accept control information from a computer and to use this information to control the actions of input-output units connected to the Exchange.

e. To permit maximum efficiency in the use of channels by automatic assignment of input-output units to available channels.

f. To allow suitable error correction and detection devices to monitor the flow of information.

12/29/56

Exchange Instructions

The following instructions are needed for linking the operation of a computer with the Exchange:

a. Select for Writing - specifies an output device and the location of a control word. The output device is selected and the control word is transferred to the Exchange memory. The Exchange control circuits initiate and monitor further operations.

b. Select for Reading - specifies an input device and the location of a control word. The input device is selected and the control word is transferred to the Exchange memory. The Exchange control circuits initiate and monitor further operations.

c. Select for Control - indicates a particular control operation required for the specified input-output device. For example, this operation might be Backspace or Rewind in the case of magnetic tape units. The nature of this instruction is dependent upon the input-output devices connected to the Exchange.

d. Interrogate I/O Unit Status - specifies that the status bits associated with a given input-output device be made accessible to the computer working with the Exchange. The status includes information about unit not ready, busy, read-write error, transmission error, etc. Detailed examination of the status field is handled by the programmer.

e. Copy Control Word - specifies that the control word as it now appears in the Exchange memory be sent to a specified location in the main memory.

Additional Requirements

The following comments are made with respect to additional requirements which are to be imposed upon the Exchange. Most of these features are already included in the Exchange planning and are included here to illustrate further the flexibility of the Exchange.

It should be possible to distribute input information into several different regions of memory without requiring control on end-of-record signals. This is accomplished by chaining control words. It should also be possible to group information into one record from several different regions of memory. This again is accomplished by chaining control words.

If information is not required in a particular record, the flow of information from or to main memory should be eliminated. The action of the Exchange might continue, but the main memory is not imposed upon. This function is probably more closely associated with input.

Information coming from an input unit should be treated as a serial sequence of bits. This sequence should be placed in memory in groups of sixty-four bits. No provision should be made to standardize ~~byte~~ byte size or adjust the information to a specified word length with existing input-output devices. These operations can be handled by programming. It should be possible to influence future devices to meet the exact requirements of the Exchange in a more suitable way.

Medium - Speed Input-Output Devices

The following devices are considered in the medium speed range and will probably be used on the Exchange:

a. Type 727 Magnetic Tape Units (or an improved version of such units) to provide a communication link between this system and the 704-705 systems and certain auxiliary equipment.

12/29/56

b. Electronic Printer Plotter to permit page printing and print plotting with recording on microfilm. An associated visual display will be provided. The printing rate is 16,500 characters per second.

c. High-speed direct printer with the operational characteristics of the present 1000-lines per minute wire printer. This will be used for direct communication with the computer and for use as an auxiliary operated printer.

d. High-speed card reader for direct attachment to the computer or for use as an auxiliary card-to-tape converter.

e. Card Punch of highest speed available for direct attachment to the computer or for use as an auxiliary tape-to-card converter.

Low-Speed Input-Output Devices

The following devices are considered in the low-speed range and will probably be used with the Exchange. The number of units included in this category when one considers the Stretch Computer is probably extremely small. The number will probably increase as real-time applications are associated with the Stretch Computer.

a. Manual Keyboard and Typewriter - interrogation devices of this nature will be used to enter and to extract information from the system.

b. Telephone Lines - will usually form a natural link between the source of information and the computer system and between the computer system and the controlled elements.

c. Visual Displays - will be used to an increasing extent.

12/29/56

High-Speed Input-Output Devices

The Exchange is limited to handling information centered around the 15KC range. Such data transmission rates are clearly quite low with respect to the computing speeds of the Stretch Computer. The following devices will be needed to provide a somewhat more balanced system. The control equipment required may be an attachment to the Exchange or separate units.

a. Faster Magnetic Tapes - tapes operating at a rate of 100 times that of the 727 will be required. These will be used as additional external memory. A tape of 10 times that of the 727 will serve as the input-output connection to other computers. It will probably also be used as an auxiliary recording source for later conversion to printed form or card form. This speed tape should be ideal for handling the input-output requirements of a machine operating at about 0. 1 the rate of the Stretch machine.

b. Magnetic Disc Storage - with a capacity of a million words and the ability to transfer information between this disc storage and the main memory at a speed of 1 word every 4 microseconds. This storage is used to supplement the mangetic core memory for the storage of instructions and data.

Provision must be made for the efficient selection of these units and the rapid initiation of information transfer.

Input/Output COMPUTER

[redacted] For certain applications, it is desirable to a have a separate unit to handle input-output considerations. These applications are usually in the realm of problems which require a large amount of data to be processed or in the realm of real time problems. For problems in which it would be advantageous to handle input/output with a special program running concurrently with the working program, a special unit is available.

This unit is a full-sized computer in itself, and it is one which can operate alone or in conjunction with the Stretch Computer. The most distinguishing characteristic of it is that its speed is about 10X faster than 700 series machine and about 10X slower than the Stretch Computer. The Secondary Computer, as it is called, has 64 bit words and an assortment of word formats all of which are compatible with the Stretch machine.

However, the principle bias of the Secondary Computer is toward a combination commercial-scientific data processing machine. To this end, it contains hardware for both decimal and binary arithmetic, among other things. The Stretch machine makes no specific concession to commercial data processing problems, but the Secondary computer does, and such concessions are manifested in the provision for arranging, rearranging and transforming of data formats. These provisions, which are more elaborate in the Secondary computer than in the Stretch Computer, with the ability to perform arithmetic computations in both the binary and decimal modes will present to the commercial field a computer much more advanced than any machine now existing or conceived.

The ability of the Secondary Computer in the field of scientific data processing will be provided through built-in floating point arithmetic (in binary only) and very flexible indexing facilities. The arithmetic speeds of the Secondary computer will allow it to tackle problems considerably larger than are now possible on the 704, but the ultimate in scientific data processing ability will be furnished only by the Stretch computer, and no attempt will be made to increase the power of the Secondary computer for scientific problems beyond a factor of 10 over the 704.

Thus, the real emphasis on the Secondary computer is toward the commercial data processing field, and scientific problems will receive only a secondary consideration beyond the provision of certain necessary features such as floating point arithmetic.

Since the Secondary Computer will have superior ability to edit data for input-output operations, it is possible to use it as a fully programmed input-output computer for the Stretch machine, thus relieving the Stretch machine from that work which may be done in parallel with the computational work on the data. Since the requirements of such a machine are difficult to meet, a complete computer will be available to process input-output information.

During those periods when the Secondary computer is operating in conjunction with the Stretch computer, the two may communicate with each other through three media: tape, memory, and the bus system of the machines. A program to synchronize the action of the two machines will therefore not be limited by the method of communication between the two.

If there are periods when the two machines need not act in conjunction with each other, both may be operated independently of the others. In this case the two machines act as completely separate computer systems and each has all the basic facilities to make such operation feasible.

12/29/56

V. OPERATIONS

A. Floating Point Operations

The specifications of the actual arithmetic operations in floating point are easily defined since most problems will be programmed in normalized floating point. The requirements for special information about results which have not been easily available in other computers has been extended considerably.

There will be operations of add, subtract, multiply, and divide in single precision floating point with sign manipulation of one of the operands. Multiple precision operations can be easily programmed. There will also be special manipulation and test operations for all parts of the floating point data word.

Special information should be available to the programmer about the result of any floating point operation. This information can be categorized as follows:

1. Exponent overflow
2. Exponent underflow
3. Exponent in range but greater than a set value
4. Complete figure loss as a result of addition
5. Improper divisors

One a programmer has been "notified" that one of the above events has occurred the ability to propogate this exception should be his option. Therefore the following handling of exceptions is proposed.

1. Optional break-in if both operands are regular but the result out exponent overflows or underflows, or an imporper divide is attempted.

2. Optional break-in if figure loss is complete.

3. If either or both of the operands has an indication of exponent overflow or underflow the result will be predetermined by a set of fixed rules. If mantissas are zero they will be treated as if they were non-zero except if they are used as divisors.

4. Program Assignable bits can cause optional break-in as they flow into the S register from memory for use as operands but only those bits placed specifically by the program will be stored i.e., these bits will not be automatically transmitted through the arithmetic unit.

The rules for propogation of exponent overflow or underflow are given by the following figures. E is the symbol of exponent overflow, X is the symbol for a normalized floating point number, and e is the symbol for exponent underflow.

Add operations

Operands	E	x	e
E	E	E	E
x	E	*	x
e	E	x	e

* The result can cause optional break-in if exponent overflow or underflow, if exponent exceeds a set magnitude, or if figure loss is complete.

Multiply operations

Operands	E	x	e
E	E	E	E
x	E	*	e
e	E	e	e

* The result can cause optional break-in if exponent overflow or underflow, or if exponent exceeds a set magnitude.

Divide operation

Dividend / Divisor	E	x	e
E	E	e	e
x	E	*	e
e	E	E	E

* The result can cause automatic break-in if exponent overflow or underflow, if exponent exceeds a set magnitude. Optional break-in will also be caused if the divisor mantissa is zero.

Operation -

Assume a sign manipulator, M which has four states: use sign, invert sign, set sign plus, and set sign minus. The symbol S will be used for the operand in the S register, the sumbol A will be used for the 48 most significant bits of the A register and the symbol AB will be used for the complete A register. The symbol R will be used for the contents of the R register.

All add operations will shift the number with the smaller exponent right before addition. If the mantissa sum overflows the mantissa will be adjusted and the exponent increased by one. If figure loss occurs the mantissa will be normalized and the exponent decreased by the amount of normalization. The result will always be a double length mantissa in AB.

Add operation

MS + A into AB

MS - A into AB

MS + AB into AB

MS - AB into AB

MA + S into AB

MA - S into AB

MAB + S into AB

MAB - S into AB

Multiply operations

All multiply operations will produce a double length product in AB. The exponents will be added and a single leading zero in the product will be normalized.

MS • A into AB The two operands will be in S and R

MA • S into AB The two operands will be in S and R

Divide operations

All divide operations will produce a single length quotient in AB and a single length remainder in R. The quotient exponent will be the difference of the operand exponents. The remainder exponent will be the dividend exponent less the number of divide sub cycles taken to produce the full length quotient.

MB ÷ S into Quotient in B, Remainder inR

MA ÷ S into Quotient in B, Remainder inR

MS ÷ A into Quotient in B, Remainder in R

Special operations

Square root MS into B

Load MS from memory

Load MAB from memory

Load MR from memory

Store MS

Store MR

Store MA

Store MAB rounded

Store M (lower portion of B) this operation implies the generation of an exponent 48 less than B exponent.

Round MAB

Normalize AB

Make exponent S into floating point number in AB

Make mantissa S into exponent in AB less one with mantissa equal to one half

Borrow from mantissa A

Interchange exponent S and AB

Compare exponent S

Compare exponent AB

Compare exponent R

Compare Floating Point number

Count leading zeros of AB

Add to mantissa AB

Add to exponent AB

Insert Program bits in AB

Insert Program bits in AR

Set exponent of S

Set mantissa of S

This is only a specimen list of the special manipulative commands for floating point numbers. The actual detailing of these commands will depend upon the actual machine hardware.

There will also be a mode of floating point operations which will introduce a fixed error similar to rounding into all arithmetic results to facilitate the determination of significance. This mode will cause the automatic inversion of a predetermined bit in the result of all arithmetic operations. This inversion will take place before normalization. The operations necessary are:

Enter mode (The accumulator bit position is assigned by this instruction).

Leave mode.

Test mode.

V. OPERATIONS

B. Fixed Point Operations

The fixed point operations can be divided into two classes: those operations upon "full word" data, and those operations upon pieces of instructions and index words.

The set of operations proposed for the first class will be similar to the floating point operations and will be on a word length basis. The word length at the present time is 48 bits with sign and the 4 assignable bits. The remaining 11 bits will be carried along but not considered in the calculation. There will be optional break-in for the cases of overflow on addition and improper divisors. The results will be defined if break-in is not taken and summary indication will be made. The 4 assignable bits and the 11 unassigned bits will be maintained in the S register but will not be transmitted through the arithmetic unit.

The operations are:

MS + AB $\rightarrow$ AB

MS - AB $\rightarrow$ AB

M AB + S $\rightarrow$ AB

M AB - S $\rightarrow$ AB

MS · A $\rightarrow$ AB

MA · S $\rightarrow$ AB

M AB $\div$ S $\rightarrow$ Q in AB, R in R

MS $\div$ A $\rightarrow$ Q in AB, R in R

Load MS

Load M AB

Load MR

Store MS

Store MA

Store MR

Store MA rounded

Store M (lower portion of AB)

Compare

Transfer Zero

Transfer Non-Zero

Transfer Plus

Transfer Minus

The set of operations proposed for the second class is a set of variable field length operations. This set of operations should have in addition to the data address and an index tag; three six-bit addresses. These addresses will indicate the starting bit position in the S register, the field length, and the starting bit position in the A register. The address indicating the starting bit position in the S register is equivalent to the actual bit address of the storage position addressed. This allows the programmer to address any bit or group of consecutive bits in any addressable location. During any indexing operation using this set of operations the six position bit address will be appended to the low order positions of the data address and any index quantities will increment this entire field with appropriate carry or borrow propagation. Thus, the entire storage of the machine can be considered to be a continuous string of bits without word boundaries. This system implies that the high-order bit of a word has the bit address, 0, and the low-order bit of a word has the bit address, 63.

12/29/56

Since the index words will contain one or more signed increments in which the sign (or signs) will not be in the high order position of the word, but associated with the actual increment field; it is necessary that the set of instructions be of both a signed and an unsigned nature. If the unsigned mode is specified the field will be considered positive and all bits of the field will enter the data portion of the accumulator. If the signed mode is specified, one bit of the field will be combined appropriately with the accumulator sign and the remaining bits of the field will enter the data portion of the accumulator.

The set of commands are:

Load S unsigned

Load S signed

Load A unsigned

Load A signed

Add A unsigned

Add A signed

Subtract A unsigned

Subtract A signed

Store A unsigned

Store A signed

Compare A unsigned

Compare A signed

It will be noted that the operations Load S unsigned and Load S signed allow the programmer to obtain partial words for use in full length fixed point operations which have not been specified in the above list such as multiply and divide.

This set of commands will make un-necessary a host of special manipulative commands which are required in other computers e.g., shift, store address, load address, etc. They will also give the programmer a system much more flexible than has been attainable in the past except by complex programs to extract or insert partial words in memory. Besides being necessary to manipulate instructions and index quantities, these commands will be useful in the cases where tables are collapsed to eliminate un-necessary zeros in words or zero table values. The presence or absence of table values can be indicated by a bit array containing a one or zero, simplifying in many cases large volume table storage requirements. This set of commands will also be necessary when the computer is required to perform radix conversion or other input-output functions.

C. Data Transmission Operations

There should be two types of data transmit commands. One type to transmit a word or block of words from one memory location to another and a second type to interchange a word or block of words between two locations. It appears desirable that not only should both starting locations be indexable, but that different increments may be specified. The transmit command should be strictly a memory function and not disturb the arithmetic section. The interchange command may have to use the S register as an intermediate buffer register.

V. Operations

D. Logical operations. The Stretch computer will have a bank of selectors (triggers) available to the programmer to indicate the status of various component units of the machine. These triggers will be many in number and come in two classes: those which indicate the status of certain machine components and those which are controlled only by the program. In addition, the addressing scheme of the machine allows the addressing of any bit in any word in the memory, thus making it possible to look at the memory as a collection of bits, not words.

To facilitate operations on collections of bits, special commands are available, for example:

AND	$X	The contents of X are ANDed with the contents of the A Register
OR	X	Same for OR operation
Exclusive OR	X	Same for Excl. OR
Set	X	For each bit in the A register, set the coresponding bit in word X to a one.
Reset	X	Set to zero as above.
Invert	X	Invert as above.
Transfer if X ON		For each bit in the A register, examine the corresponding bit in Loc. X. If all bits examined are on, transfer.
Transfer if any X ON		Same if any one of the corresponding bits are ON.
Transfer if any X OFF		Same for OFF

12/29/56

The following are to be applied without reference to any other register - i. e. applied to the Loc. X only:

Transfer if all X ON

Transfer if all X OFF

Tr any X ON

Tr any X OFF

The following include geometrical to numerical address conversion:

Tr X, b, c, y ON - Starting at bit b in Loc X, examine the next c bits. When a bit is found ON, place n (the bit no. above b) in the CA and Tr to y.

For priority assignments etc.

Tr X, b, c, y OFF - Same for first zero.

Count Ones X, b, c Count the total no. of one bits from bit b in X to c. Place the no in the CA.

Count Zeros X, b, c Same for zeros.

All of the above commands may be used with bit addressing and variable field length addressing. These commands may refer to any bit in the memory as well as to any selector.

Provision for storing in a selector the result of a test on any bit or group of bits will be provided as follows:

Set Sel. Z ON if X ON
Set Sel Q OFF if X ON
Invert Sel. Q if X ON

} These commands will act if the bit or group of bits specified in X is ON.

12/29/56

Set Sel. Q ON if X OFF

Set Sel. Q OFF if X OFF

Invert Sel. Q. if X OFF

Same as above but X must be OFF

The selector,, Q, to be operated may itself be the bit or one of the group of bits addressed by X. These commands allow one to program any of the binary logical connectives not included in the machine as operations.

The selectors will be addressable in two modes: numerical and geometrical. The numerical mode is the usual mode of addressing selectors in present day machines. This mode relies on an assigned numerical address for each selector. Any selector may thus be interrogated by specifying its numerical address in the command. The geometrical mode of addressing depends on the positions of bits in a register. In this mode, a selector is assigned to a particular bit position in a register, and the selector may be interrogated by a command which refers to the register. Those selectors corresponding to the bit positions with ones in them will be examined, all selectors corresponding to the bit positions with zeros in them will not be examined. This addressing mode has the advantage of permitting the examination of many selectors at one time.

The above set of commands, along with the data transfer commands covered in the following section, constitute a very powerful means for logical control of programs and the rearrangement of information within the machine.

E. Testing and Transfer

There will be a complete set of "full" word length test and transfer commands for both fixed and floating point data such as compare, transfer zero, transfer minus, etc. These commands as well as the variable

length compare command and the index test commands have been discussed in other sections.

There will also be a set of variable length test commands. These will consist of a word address and an associated bit address and field length. The word and bit address will be indexable. The tests will be:

Test D, b, f for all ones.

Test D, b, f, for any one

Count all ones in D, b, f.

Count all zeros in D, b, f.

Count leading zeros in D, b, f.

Count leading ones in D, b, f.

These commands will set an indicator bit in the control section which can be interrogated for a subsequent possible transfer. It will be noted that this set of commands eliminates the need for special test commands for zero, non-zero, plus, minus, etc., when operating with the variable length arithmetic. It also allows the programmer to consider any portion of memory as a group of selector bits. This will probably eliminate the need for a special class of indicators (or selectors) which are program assignable.

One of the more important uses for logical and testing commands will be in the utilization of the machine break-in system. The Stretch Computer will have, for certain conditions, a means for usurping control of the machine in the event that logical conflicts arise. Some of these conflicts may arise out of input-output operations and some may arise from inconsistencies in the floating point arithmetic system.

In particular, the floating point arithmetic system is not capable of a completely consistent definition, for the definition must be made in terms of a given problem class. For instance, there is, at this time, no way to define in floating point notation a zero which is exactly analogous to the zero found in fixed point notation. In floating point notation there are many possible definitions of zero, and the best zero to be used depends upon the problem. Since more than one zero definition in the computer is not feasible, one must specify what the machine must do if the computer produces an undefined number as the result of an operation. Therefore, a reasonably general definition of the results of any given operation will be built into the computer. For those cases in which it is possible to specify an alternate definition, break-in will be provided.

The break-in, however, will be optional in every case, and the program must specify the modes to be used. The optional break-in feature will be composed of two parts: the Break-in Triggers which are directly associated with each condition upon which break-in is to be allowed, and the Break-in Control Register which contains a bit position for each Break-in Trigger. If the corresponding bit in the Break-in Control register is a one, the Breakin Trigger, when turned on, will force an automatic transfer of control to a specific location. If the corresponding bit of the Break-in control Register is a zero, no automatic action will be associated with the Break-in Trigger, and the machine will ignore that trigger.

This system allows the programmer to specify which particular occurrences will cause an automatic transfer of control. Thus, if the programmer cannot use the machine definition of the result of any given operation, he may, thru the specification of certain Break-in Triggers, generate his own definition in a subroutine and have it automatically applied whenever the condition arises.

The commands covered in the previous sections will enable the programmer to modify the Break-in Control Register at will and to test the Break-in Triggers in order to determine the status of the machine at any time. The Break-in Triggers will be selectors of the same type as discussed previously, and will be controllable to the extent implied by their use.

G. Data Transfer - The Stretch Computer will have available a special command for moving a group of bits from any position in any word to any position of any other word. The command itself will look as follows:

Transmit Data X, a, b, Y, c

This instruction will transmit b bits starting at location X, bit position a, to location Y starting at bit position c, All other bits at location Y remain undisturbed. Locations X and Y may be any register or any memory location, and may even be the same word. This instruction allows the transmission of a group of bits (not to exceed 64) from any word to any other word, effecting at the same time a shift of a specified number of positions to the right or left.

Another version of the same command exchanges two groups of bits:

Exchange Data X, a b, Y, c

This instruction will exchange, b bits starting at location X, bit position a with b bits starting at location Y, bit position c. All other bits in both X and Y remain undisturbed. No more than 64 bits may be exchanged in one operation.

Another version of this command will exchange blocks of words in memory,

Exchange Words X, n,Y

This command will interchange the locations of a block of n words, one block starting at location X and the other block starting at location Y. The two blocks may not overlap in memory.

All of the above data transfer commands make use of the Main Arithmetic Unit and its shift matrix, and therefore the contents of the registers will be destroyed by any data transfer commands. In addition, no other use can be made of the Main Arithmetic Unit while such operations are in process.

These commands allow the physical rearrangement of data to be facilitated with a minimum number of commands.

12/29/56

H. Compare Operations - Commands to perform data comparisons and subsequent trigger setting will be available. It will also be possible to compare two strings of bits both of which begin at arbitary locations within any two words. Usually, Hi, Lo, and Equal indications will be given through setting appropriate selectors or triggers. Both logical and arithmetic compare operations will be available.

I. Table Lookup Operations - Certain specialized table lookup operations are necessary to facilitate the use of tables to represent functions. For the Stretch Computer, the commands will be of two types: one type will search a table by comparing each entry against a standard, and the other type will effect a replacement of a given argument. The table search command will operate as follows:

Table Search X, a, b, c

This command will search a table starting at Word X, bit a. It will compare b bits (starting at position a) with the contents of the A register. If equality occurs, the word and bit address of the argument will appear in the Control Accumulator. If inequality occurs, the CA will be indexed C bits and the process repeated. If no equality occurs before a preset address in a special register occurs, the process will terminate.

This same command will appear in several modes of compare operation:

Conditions
Equal
Hi or Equal
Lo or Equal

12/29/56

In addition, the command will operate to find the highest or lowest value in a given table.

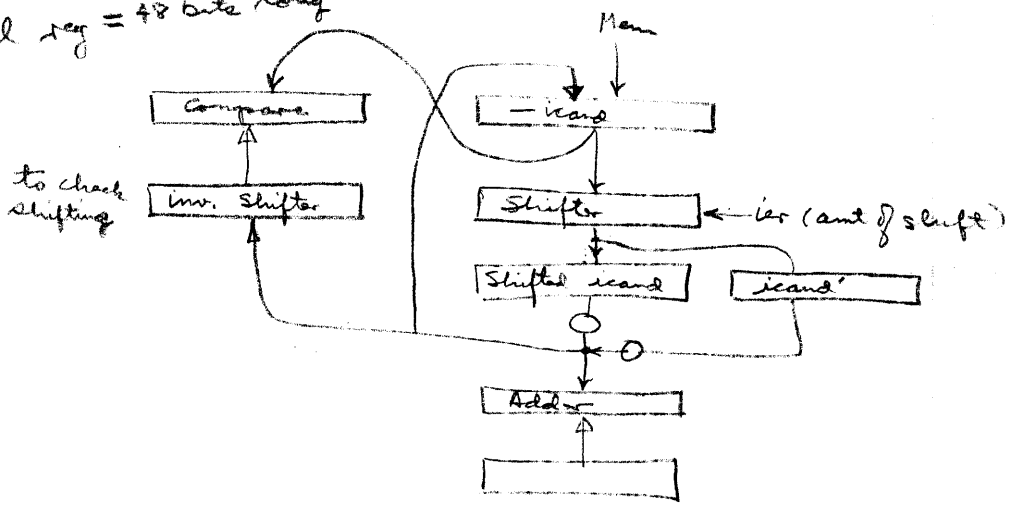
The replacement table lookup will look at m bits in the low order part of the lower half double length A register and will lookup in a table starting at X, a, a group of m bits which will then be placed in the high order end of the upper half of the A register. This cycle may be repeated a number, k, of times. If the cycle is repetitive the registers will shift appropriately between cycles so that a sequence of groups may be transformed by the lookup process.

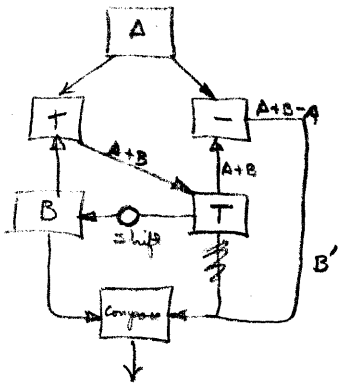
J. Shift commands

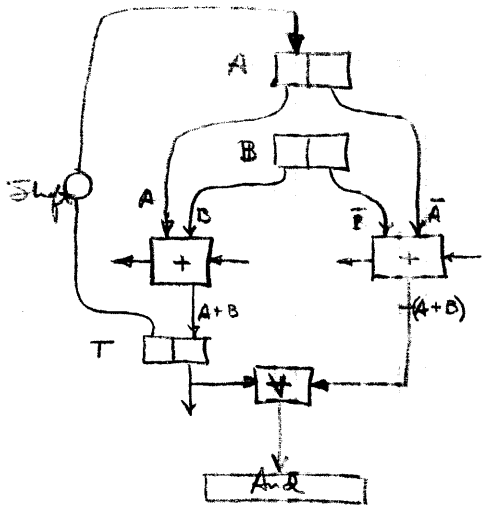
a) Logical DP, left

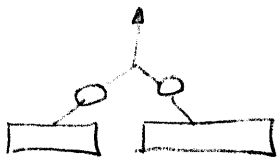
b) Ring DP, left

all reg = 48 bits long









Select ~ 10 μsec

Transfer  ~ 25 μsec

Shift ~ 25 μsec (Transistors)

(Cores)

VI. Automatic Program Interrupt Control

We shall, in the following, use the expression "break-in" to refer to the automatic interruption of a program in exceptional situations. In the STRETCH computer many new problems appear because of computer characteristics that have no precedent. For example, the asynchronous non-sequential control allows the execution of control instructions prior to arithmetic operations which normally would precede them. Thus, the recording of enough information to permit continuation of a program after a stop occurs is more complicated than in earlier computers where it is often adequate, for this purpose, to remember only the last executed instruction. In those situations where several operations are occurring simultaneously (e.g. computing and using input-output equipment), if break-in situations occur in more than one sequence of operations, some observation of priorities may be necessary in determining the ordering of the corrective or other operations undertaken by the computer. Rather than attempt to make automatic the perhaps difficult decisions involved in doing this, we propose that the programmer be responsible for this corrective action. This may not be very much of a burden, however, because it is expected that a general purpose library program can handle these situations in most cases, and it is proposed that such a program be an integral part of the break-in system. This program will be referred to below as the break-in library program. The desirability of having such a program, assumed to be present in all normal uses of the machine, will in part depend upon the amount of storage space that will be required to accommodate it. It may be feasible to store that part of the break-in program that may be called upon frequently for the more common break-in conditions in core storage (2 μ sec cycle), while those parts that handle more

exceptional situations can be held in auxiliary storage. If, on further investigation, a very general program seems to be feasible then, in several situations, as indicated below, it may be worth including in this program procedures which, if executed by the inclusion of more hardware, would add significantly to the complexity and cost of the computer.

It is assumed that a "Mask Word" register will be under the control of the programmer. The 64 bits of this word will refer to the status of as many "triggers" which respond to various machine states (defined below). In addition, an "Indicator Word" register will indicate the present status of these triggers. This register will also be addressable and modifiable by the program. When the logical product of these words is not zero, a break-in will occur. This will cause a transfer to a fixed location (the first instruction in the interrupt library program). It might be slightly more convenient to have the machine transfer automatically to a different location for each break-in situation. However, the cost of the additional hardware required to accomplish this should be considered and, if it is at all significant, the transfer to a fixed location should be made. The break-in program can then determine from the logical product of the Mask Word and Indicator Word the proper address to which to transfer. Since the first step in the break-in procedure may, as indicated below, always be the same (copying the decoder registers) it may be desirable to do this in any event.

Normally, many of the bits in the Mask Word will always be "on". Since a general break-in system is being proposed we suggest below the inclusion therein of the handling of several situations where the incorporation of enough hardware to accomplish automatic corrective action does not seem to be justified by the infrequent

need for this action. Thus, we propose that errors in the results of arithmetic operations and possibly errors in the transfer or retention of words (the desirability of handling this type of error by this means will depend upon the type of error-detecting code that is eventually specified and the cost of including automatic correction) be corrected by the break-in library program rather than by additional hardware.

As an example of a special operation that might be considered worthy of inclusion in a general break-in system (but possibly could not be justified if it, and all the hardware required to accomplish it, were considered alone), a special SUBROUTINE instruction has been suggested (by Mr. W. P. Heising). This instruction located at A would contain addresses B and C. It would cause the program to transfer to B and continue with the subroutine beginning at this location until an instruction from location C is called for. When this occurs the machine would automatically return control to the instruction at A + 1. If the instruction SUBROUTINE is given before the return transfer for a preceding SUBROUTINE instruction has been effected, the machine would "remember" enough information to permit this return to occur after execution of the most recent SUBROUTINE instruction and the return to the location following it. This could all be accomplished by additional hardware including a "compare address register" and a "comparator" together with the break-in library program. The compare address register would contain the address, C, that appears in the most recent SUBROUTINE instruction. The comparator would compare the contents of the instruction counter and the compare address register.

Thus, this type of instruction involves a break-in operation. However, to handle it in the same way as the other break-in situations described below involves more hardware. It would be necessary to set a selector or break-in trigger bit in the Mask Word when the instruction SUBROUTINE is given (hereinafter called the SUBROUTINE bit). The corresponding bit in the Indicator Word would be turned "on" when the comparator indicates agreement of the number appearing in the instruction location counter and the number, C, appearing in the most recent SUBROUTINE instruction. Because of the possibility of subroutines within subroutines, the interrupt library program would have to take appropriate action when a SUBROUTINE operation is encountered before the return condition of a previous SUBROUTINE operation. This would include storage of the most recent C address in the compare address register and the A and B addresses in the memory used by that part of the interrupt library program which performs this function. When a SUBROUTINE instruction is given, the SUBROUTINE bit in the Mask Word will be examined and, if a previous SUBROUTINE order has not been completed, a different bit in the Indicator Word will be turned on and a break-in transfer to the library program will occur.

A list of suggested break-in conditions follows. Each of these conditions corresponds to one bit in both the Mask Word and the Indicator Word. In an installation that includes a separate Input-output computer the detection of and remedial action for some of the break-in conditions listed below would be undertaken by this computer. Under group B, "E" represents a floating point number with an exponent overflow indication and "e" represents a floating point number with an exponent underflow indication.

A. Input-Output Conditions

1. An error in reading.
2. An error in writing.
3. An end of file indication in reading (before a specified number of records has been read).
4. An end of tape indication in writing.
5. Attempting to read or write beyond a maximum address (e.g. on a drum or RAMAC unit)

(In the above conditions it is assumed that a register will contain the address of the input-output unit involved, supplementing the information supplied by the break-in bit)

B. Exceptional Arithmetic Conditions

6. Positive exponent overflow in floating point multiplication.
7. Negative exponent overflow in floating point multiplication.
8. Positive exponent overflow in floating point division.
9. Negative exponent overflow in floating point division.
10. Positive exponent overflow in floating point addition or subtraction.
11. Negative exponent overflow in floating point addition or subtraction.
12. A zero mantissa result of a floating point addition or subtraction*.
13. A zero mantissa result of a floating point multiplication (if unnormalized arithmetic is used).*
14. A zero mantissa result of a floating point division (if unnormalized arithmetic is used).*

15. An exponent in the result of a floating point multiplication which exceeds some preset bound*.

16. An exponent in the result of a floating point division which exceeds some preset bound*.

17. An exponent in the result of a floating point addition or subtraction which exceeds some preset bound*.

18. An exponent in the result of a floating point multiplication which is less than some preset bound*.

19. An exponent in the result of a floating point division which is less than some preset bound*.

20. An exponent in the result of a floating point addition or subtraction which is less than some preset bound*.

21. Attempting to multiply a number of type E by one of type e.

22. Attempting to divide a number of type E by one of type E.

23. Attempting to divide a number of type e by one of type e.

24. Presence of an exponent overflow bit in one of the operands in an arithmetic operation.

25. Presence of an exponent underflow bit in one of the operands in an arithmetic operation.

26. Attempting a fixed point division where the divisor does not exceed the dividend.

27. Occurrence of an overflow in a fixed point addition or subtraction.

28. Occurrence of an overflow in a left shift operation.

29. Occurrence of an overflow in a floating to fixed point conversion.

30. Complete loss of significance in a floating to fixed point conversion.

*The procedure followed by the interrupt library program will depend upon the setting of indicator bits in the data word operands.

C. Arithmetic and Other Machine (or Programmer) Errors

31. An error in a fixed point addition or subtraction.

32. An error in a fixed point multiplication.

33. An error in a fixed point division.

34. An error in a floating point addition or subtraction.

35. An error in a floating point multiplication.

36. An error in a floating point division.

37. An error in a word (indicated by the error detection bits) being read out of a high speed storage unit.

38. An error in a word (indicated by the error detection bits) being written in a high speed storage unit.

39. Attempting to interpret an impossible operation code in an instruction.

40. Attempting to interpret an impossible address in an instruction.

D. Programmer Controlled Conditions

41. Break-in bit used by the SUBROUTINE instruction described above. This bit in the Indicator Word is set when the contents of the Instruction Counter are shown by a comparator to be equal to the contents of a "control word register".

42. Occurrence of a SUBROUTINE order before the completion of a previous SUBROUTINE order

43. Complete tracing break-in bit under programmed control. It causes a transfer of control to the interrupt library program after the execution of every instruction. The corresponding bit in the Indicator Word will normally always be on.

44. This bit is like (43) except that it is controlled by a manually operated switch.

45. Transfer tracing break-in bit under programmed control. It causes a transfer of control to the interrupt library program before the execution of all transfers.

46. This bit is like (45) except that it is controlled by a manually operated switch.

47. A manually controlled STOP switch sets this bit in the Indicator Word. Thus, manually stopping the calculator is made part of the break-in system.

48. A preset time of day has been reached by the real time clock.

49. A preset lapsed time interval is indicated by the real time clock.

50. These fifteen bits in the Indicator Word are under the control of manual
. switches, the program, real time devices and any parallel data pro-
. cessors (such as an input-output calculator) incorporated in a STRETCH
. system. Any of these can interrupt the high speed STRETCH com-
64. puter at any time, if the Mask Word permits it.

The first step in the break-in procedure will be to duplicate the five asynchronous control (or decoder) registers in five reserved memory registers. Since a program could not perform this function if its own instructions went through the decoder, it seems necessary to provide additional circuitry that would accomplish this when a break-in occurs. It will probably be desirable that the break-in program be informed of the original locations of the instructions in the decoder and therefore that these locations appear in the decoder registers as well as the instructions themselves.

If two break-in conditions occur simultaneously or if a second break-in occurs while a prior break-in situation is being corrected, no special difficulties are encountered. In both cases the break-in program can handle the different situations on a priority basis. If the corrective action for any break-in must be postponed, it will be the responsibility of the break-in program to "remember" the Indicator Word, the contents of the control registers and whatever other information is necessary in order to correct the situation at a later time.

The break-in program should probably be written so that the frequency and location of machine errors, occurrence of overflows and underflows and other information of interest to the engineer and the programmer will always be recorded. This would probably be an exorbitant requirement if break-in action were entirely automatic.