PRELIMINARY MANUAL

HARVEST SYSTEM

MAY 1, 1957

Company Confidential

IBM Research and Product Development
Poughkeepsie, N. Y.

TABLE OF CONTENTS

1. General Description of the Harvest System

Harvest is a complete general purpose data processing system consisting of input-output devices and control, memories, and a central processing computer. It is being designed in such a way that additional computer processing units can be incorporated into the system.

The Harvest system will provide an overall operating speed on data processing problems from 100 to 500 times faster than is possible on data processing equipment available today. If floating point features are added for scientific computing problems, Harvest will operate from 20 to 50 times faster than is possible on computing equipment commercially available today.

The Harvest design permits several input-output units, memory units, and arithmetic and logical units to function simultaneously. This is accomplished by providing sufficient independent controls in each of these units, and by providing an asynchronous mode of operation throughout the entire system.

The system is being designed to operate for extended periods of time unattended by either operating or engineering personnel. The reliability necessary for this type of operation will be achieved by the use of solid state components, by checking all critical operations throughout the system, by automatic error correction in important areas, and by a fault location system which is an integral part of the design.

All memory units in Harvest have a fixed word size of 64 binary information bits with additional non-information bits added for checking. However, controls are provided in the arithmetic and logical units of the system, hereafter called the computer, to permit addressing any bit in memory and to adjust automatically for data which crosses word boundaries. Thus to the programmer, each type of memory in Harvest may be considered as a continuous array of binary storage.

The computer operates in parallel on eight information bits, or any subset of them. This variable sized set, from one to eight bits, is called a byte. Controls are provided to permit arithmetic operations on fields of up to 64 bits, and logical operations on any size field.

Memory addressing is binary to permit maximum utilization of mapping and table reference techniques. However, for arithmetic

and logical operations binary, decimal, and other radices are provided.

Logical operations in the computer are performed at a rate of not more than 0.2 microsecond per byte operation. Arithmetic operations will be performed at the rate of 0.2 microsecond per 8-bit byte. The entire design of Harvest has been based on the principle that all processes throughout the system, including input-output, memory references, instruction control, etc., must be organized so as to permit a continuous flow of arithmetic and logical processes on data to proceed at a rate as close as practicable to the basic rate of 0.2 microsecond per logical operation.

The Harvest system is divided into three basic areas:

a) The input-output and other external units

b) The memory, consisting of multiple units with cycle times of 2.0 microseconds and 0.5 microseconds

c) The 0.1 microsecond registers and the central computer (See Figure 1.1)

1.1 Exchange - The Exchange is essentially a small, fixed-program computer which performs a function similar to that of a telephone central switching unit. It provides, by a switching network, a means of connecting up to 32 different types of input-output devices into the system. By time sharing, it permits up to 10 devices to communicate simultaneously with the main system. The Exchange assembles, where necessary, the short words from input units such as paper tape readers into full 64 bit words and, similarly, disassembles full words into appropriate short words for output devices.

In addition, the Exchange performs the function of capturing a memory cycle in the specified memory unit and of transferring words to or from that memory unit. It also does the necessary memory address bookkeeping to provide the correct location for the next word to be transferred to or from a memory unit for a specific input-output device These operations are performed for each of the several input-output devices that are simultaneously communicating with the main memory system.

1.2 External Units - The basic Harvest system will include at least the following input-output devices:

A 1000 card per minute card reader

A 155 card per minute card punch

A 500 line per minute printer (A printer of 1000 lines or more per minute is contemplated)

Four high performance tape units with a storage and speed capacity approximately 100 times the rate of the IBM 727 tape drive

A high performance disk memory with a total storage capacity of two million or more 64-bit words and a transfer rate between disk memory and internal memory approximately the same as that of the high performance tape units

Four tape units handling IBM 727 tapes

One manual interrogation unit

1.3 Memory - The memory system in Harvest consists of one or more units of 8,192 words of storage with a 2.0 microsecond read-write cycle, and one or more units of 512 words of storage with a complete read-write access time of 0.5 microseconds. In addition, one or more single word registers with 0.1 microsecond access time can also be provided. The addressing system allows for a total memory capacity of just over a million words of storage.

The basic Harvest system will contain eight 8,192 word units of 2.0 microsecond storage units, four 512 word units of 0.5 microsecond storage units and 16 single word 0.1 microsecond registers.

Each of the eight 2.0 and four 0.5 microsecond memory units is an independent, asynchronous memory device containing its own addressing mechanism, address register, and data register. In addition, the normal manner of address assignment for these memory units is such that consecutive memory word addresses are located in different memory units.

The purpose of this method of address assignments is directly related to one of the basic advantages of the multiple independent memory units. If successive memory references are made to different memory boxes, it will be possible to effect an average memory access time considerably less than the full cycle of 2.0 or 0.5 microseconds. For the 0.5 microsecond units, the average time will approach the bus time of the system. Distributing successive memory addresses through the memory units thus provides an automatic means of optimizing memory access time without concern on the part of the programmer.

1.3.1 Counting in Memory - The 0.5 microsecond memory units in Harvest provide a special function called "counting in memory". This feature permits any word or any binary partition of a word to be used as a counter under program control. In the counting mode of operation, the specification of a word and bit address causes a one to be added to the proper bit position of that word in memory during one memory read-write cycle. The cycle, however, may be longer than 0.5 microseconds. During the same memory cycle that the count in memory is accomplished, the memory data word to which the count is added is available for transfer to computer registers if desired.

1.3.2 Existence in Memory - The 2.0 microsecond memory units in Harvest provide a function similar in concept to counting in memory called "existence in memory". In the existence mode of operation, the specification of a word and bit address causes a one to be combined with the bit at the specified address by an inclusive OR logical operation during one memory cycle. During that same memory cycle, the data word addressed is transferred to a computer register. This function provides a rapid means of recording that a particular data item has occurred and of determining whether the same item occurred previously.

1.3.3 Clearing Memory - Under program control it is possible to clear major sub-sections of a memory unit with one instruction. In the 2.0 microsecond units, sections of 64 words and 1024 words can be cleared with one instruction. In the 0.5 microsecond units, sections of 8 words and 64 words can be cleared with one instruction.

1.4 The Computer - The Computer of the Harvest system consists of a number of different units interconnectable in a variety of ways under program control. Two basic modes of operation are provided in the Computer: the streaming mode and the regular instruction mode.

The streaming mode provides a means whereby a series or two independent series of bytes can be selected from memory in orderly but complex sequences, called streams. The streams may be routed through the units of the computer in any one of a number of possible paths; logical or arithmetic operations may be performed repeatedly on each successive byte or pair of bytes and an output stream may be generated and stored in memory in some other orderly sequence.

The Stream Units of the computer are used as input, output, and transfer registers for the regular stored program instructions of the Harvest system. Streaming and regular instructions may be readily intermixed.

Adequate indexing controls are provided to permit regular instructions to be executed with a minimum in loss of time due to indexing and general bookkeeping.

1.4.1 Stream Units - The computer of the Harvest system contains three Stream Units: two primarily for input from memory and one primarily for output to memory. Each of these Stream Units (SU's) will hold two full memory words of data. Each input unit contains a sufficient amount of built-in control to permit it to generate an output stream of bytes selected in an orderly but complex fashion from the words within the registers. The third SU can similarly store a stream of bytes in a complex fashion. The SU built-in controls permit automatic loading and unloading of the registers to and from memory without, normally, any interruption in the stream of bytes being generated or stored. In the streaming mode of operation, the SU's required are initially provided with the starting location of the streams in memory and the pattern by which the bytes are to be selected from or stored in memory. This setup is done for each SU to be used. The computer operation is set up by a "streaming" instruction which defines the logical or arithmetic

operation to be performed on the bytes and defines the flow path of the data to and from the various units of the computer. The operation is then executed repeatedly until a prespecified condition halts the streaming mode of operation.

1.4.2 Table Address Assembler - the Table Address Assembler provides a means of assembling bytes from one or more SU's together with the contents of a register and a program-controlled counter to form addresses for table lookup, counting in memory, and other similar memory reference functions.

The Table Address Assembler includes the necessary controls to enable it to capture a memory cycle in the memory unit specified by the assembled address, and to transmit the address to the address register of that unit.

The Table Address Assembler is intimately associated with the Table Extract Unit of the computer.

1.4.3 Table Extract Unit - The Table Extract Unit is a data storage register similar in design to the Stream Units. Its primary function is to receive data words from memory as addressed by the Table Address Assembler and to select from them the appropriate bytes for transfer to other units in the computer. The starting bit address in a memory word transferred into the Table Extract Unit is specified by the low order bits of the address assembled in the Table Address Assembler.

1.4.4 Statistical Accumulator and Statistical Counter - The Statistical Accumulator is provided primarily for accumulating specified bytes from words in the Table Extract Unit and for counting the number of accumulations made. Associated with the Statistical Accumulator is a built-in threshold comparator. A pre-determined threshold may be loaded into the threshold register as a part of the setup operation.

The Statistical Accumulator accepts a parallel input of up to eight binary digits. Its capacity is 26 binary digits and sign. The threshold register's capacity is

26 binary digits and sign.

The Statistical Counter is normally stepped for each entry into the accumulator but may be stepped as a result of other computer conditions. Its capacity is 18 binary digits.

The Statistical Accumulator and Statistical Counter are most frequently used in conjunction with counting in memory in a streaming mode of operation. It may also be used, however, under control of a regular instruction.

1.4.5 Logical Unit - The Logical Unit (LU) performs a variety of arithmetic and logical operations on the successive bytes or pairs of bytes as they flow through the computer in a streaming mode or regular mode of operation. The particular operation to be performed is specified by the operation code of the instruction. The various operations performed by this unit are described in detail later in this manual.

1.4.6 Match Recognition Units - During the streaming mode of operation, it is possible to monitor input and output streams for the occurrence of prespecified bytes. The Match Recognition Units provide a capacity for specifying four different special bytes at any one time. Recognition of a special byte normally causes a break-in on the streaming mode of operation.

1.4.7 Byte Masks - In order to provide a ready means of operating upon any size byte from 1 through 8 bits and of any subset of bits within a byte, several byte masks are provided at strategic points on the data flow lines within the computer. The mask format is specified as a part of the streaming setup.

1.4.8 Numbering Counter - A counter is provided which reads directly into the second Stream Unit. It is used for numbering records and sequencing through tables.

1.5 Additional Computers and Processing Units - As stated above, the Harvest system is being designed to permit adding computer processing units to the basic system. Such additional units will

be connected into the main bus system in such a way that they can share the memory units and the input-output exchange with the basic computer. These additional units will be asynchronous with respect to the rest of the system.

One embodiment of this concept, necessary to complete the Harvest system, is the addition of a high speed arithmetic computer to work in conjunction with the basic data-manipulating computer described in this manual. This arithmetic computer will be a completely independent computer, capable of being operated with or without the data-manipulating computer being connected into the system.

The normal means of transferring data between two computers operating in the one system will be through the memory units. However, direct communication paths will be provided to allow either computer to initiate an automatic break-in on the other computer's operation. Priority control will monitor memory references to insure that critical memory accesses are executed and simultaneous references are properly sequenced.

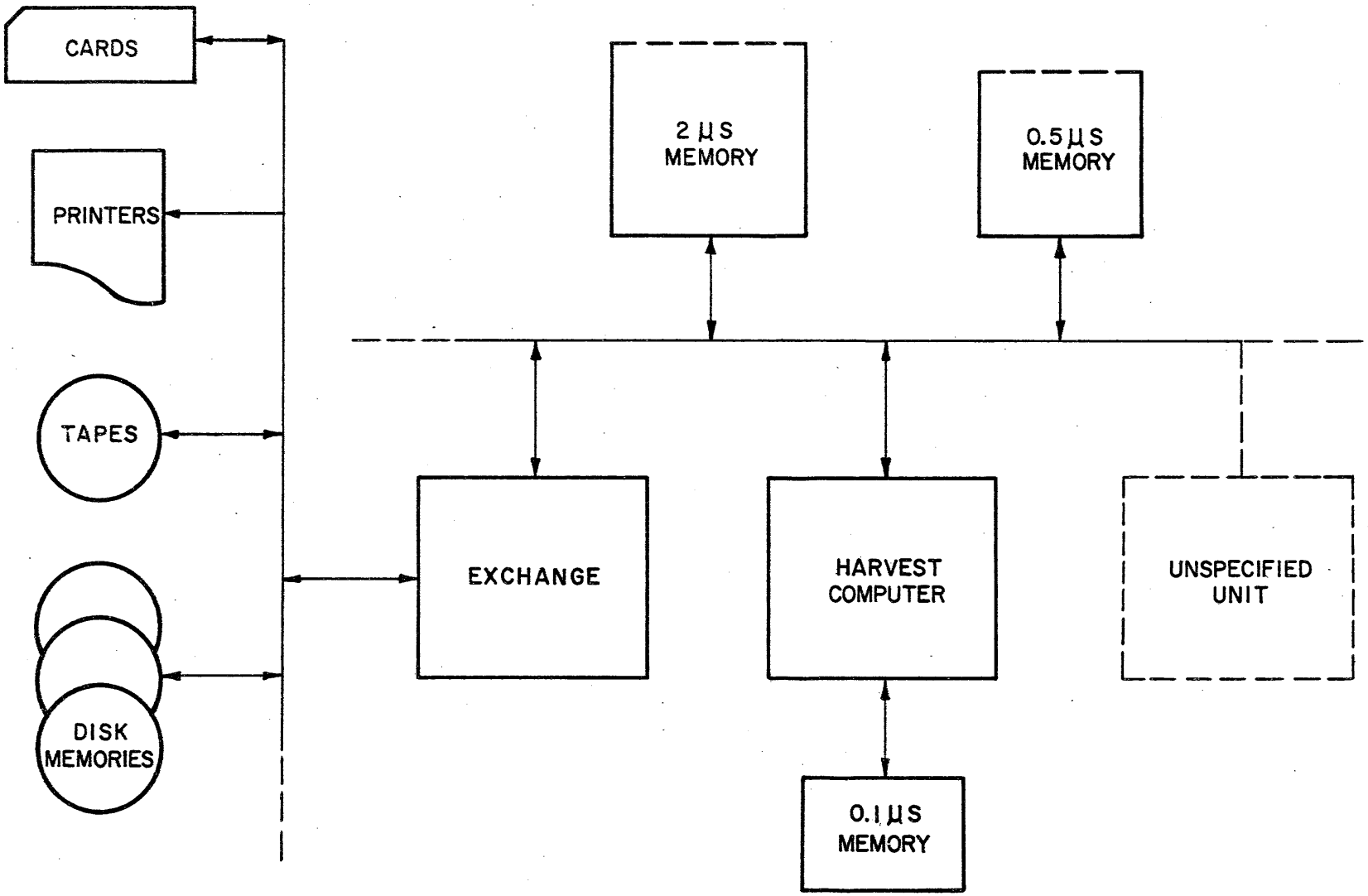
Other special processing units may be attached to this system which do not contain built-in instruction decoders and which are essentially adjuncts to one of the computers in the system. Such units will be addressed and be under control of the instruction set of the computer to which the unit is an adjunct.

1.6 The Harvest System As Part Of A Larger System - The design of the Harvest system is proceeding on the assumption of the existence of a larger assemblage which may contain several Harvest systems and perhaps other non-Harvest systems. It is assumed that the different Harvest systems will contain different groupings of Harvest units. As a part of this concept, the individual units of the Harvest system, i.e. the Exchange, the separate memory units, and the computers, are being designed in such a way that they can be detached from one system and attached to another system. In addition, the design will provide means of communication between system input-output lines whereby individual input-output devices can be simultaneously attached to more than one Exchange unit. In this way, all systems in the larger system can communicate with one another under break-in program control.

1.7 The busses in Harvest are an extremely critical part of the

overall system. Because different sections of the system, i. e. the Exchange, the Harvest computer, and other computers all operate simultaneously, and because extremely high transfer rates are required between 0.5 microsecond memories and computers, the demands on the bus system are high. Just how high, however, cannot be determined until the machine design has progressed further. Under any circumstances a priority system must be provided to protect the synchronous input-output devices, and to schedule simultaneous demands on the busses.

The asynchronous nature of the majority of the units makes it possible for any such unit to wait if a bus is in use at the moment the unit requires it. However, these waits obviously increase the operation time. Therefore the final bus system design must be such that waits of this type are kept to a minimum.



HARVEST SYSTEM

FIGURE 1.1

2. Exchange and External Units

The function of the Exchange is to direct and control the information flow between many input-output units or external memory units on the one side, and the internal memory on the other side. In addition, the Exchange provides a number of common control facilities to be time-shared among the external units, thus keeping these units as simple as possible.

The basic Exchange permits up to 32 external units to be connected to the system. These external units are of the kind which operate serially, one byte at a time, at any reading or writing rate which does not exceed 25,000 bytes per second. Up to 10 such units may operate simultaneously through the Exchange, each unit either reading or writing.

High-speed tapes and disks follow the same philosophy of control as the basic Exchange, but separate equipment is required because the higher speed does not permit as much time-sharing.

A further extension of the basic Exchange is contemplated which will allow a great many more units of much lower speed to be accommodated simultnaeously. Among these units will be manually operated devices and low-speed data units. Specifications for this section are not complete and will not be provided initially.

The Exchange provides a general method of connecting many different kinds of units to a computing system. This will include direct input from remote sources, output to remote stations, communication with other computers which are not a direct part of the system, and input-output units using techniques still under development.

To the programmer, the Exchange acts purely as an intermediary between the computer and the external units. It appears as a passive device which provides temporary storage of data and controls. The main thing which concerns the programmer is the amount of simultaneous operation and thus the overall rate of information flow provided by the Exchange. For normal operation, the instructions may be written as if they directly controlled the external unit concerned without regard to any other units operating at the same time.

To explain the philosophy behind the design of the Exchange, a brief description of the basic Exchange follows.

2.1 The Basic Exchange

A simplified diagram with the major information paths of the basic Exchange is shown in Fig. 2.1.

On receipt of the appropriate signals from the computer to initiate a reading or writing operation, the Exchange:

a) Selects a path to the external unit

b) Sets up the unit for reading or writing

c) Assembles incoming information into a memory word or breaks a memory word apart as needed for outgoing information

d) Initiates access to memory for successive words in a block of information

e) Advances the memory address from word to word

f) Keeps a count of the number of words transferred

g) Arranges to time-share all these facilities among many different external units

The Exchange transfers information in accordance with a few fixed and built-in rules. No attempt is made to have the Exchange perform editing operations on the data. Editing is a function of the computer program.

The external units are connected to the Exchange by means of a Cross-point Switch. This switch permits any one of up to 32 units to be connected to any one of up to 10 channels. Once this connection is established, the switch retains the connection until a block of information has been read or written. The connection is then released, and the channel becomes available for use by another unit. In this way any 10 of the units may operate simultaneously. If all channels happen to be in use when a unit is to be started, that unit automatically waits until another unit has finished.

Once a connection has been established for an input unit, information is allowed to flow in, a byte of 8 bits at a time. The Exchange assembles 8 successive bytes into a memory word. When a word is full, it is sent to the main internal memory.

For output, a word is obtained from main memory and sent, a byte at a time, through the Cross-point Switch to the output unit.

For either input or output, timing of successive bytes is entirely under the control of the external unit. Subject to its maximum rate of 25,000 bytes per second per unit, the Exchange furnishes service on request.

The heart of the Exchange is the Exchange Memory and associated circuits. Here the partially assembled or disassembled data words are stored for each external unit. When an input unit demands service, the corresponding data word is sent to the Word Register. From there, its contents go through the Data Word Shift Unit which shifts the entire word one byte to the left to make room for the new byte on the right. Both are returned to the Exchange Memory via the Memory Drivers.

The path for output information is essentially the same except that the output byte is extracted from the left end of the Word Register. The word is then shifted and returned to memory; this brings another byte into position for the next round.

Each data word has associated with it a count of the number of bytes assembled or disassembled. When this number reaches eight, an access to main memory is initiated.

Accesses to main memory are controlled by means of a control word which must be furnished by the computer program at the start of a block. The control word consists basically of a Word Address and a Word Count. Every time a data word must be transferred to and from main memory, the control word is obtained from the Exchange Memory to determine the desired address in main memory for the data word. Before returning the control word to the Exchange Memory, the Word Address is stepped up and the Word Count stepped down by one. This is done in the Control Word Modification Unit.

There are two ways of terminating a block of information. The

External Unit may signal the end of the block. For example, a tape may reach a gap on reading. If no such signal has been given, the Exchange terminates the block when the Word Count in the control word reaches zero.

To start the whole process of reading or writing a block of information, the computer must give a READ or WRITE instruction. The instruction contains an address which specifies the external unit and which also defines the location of the corresponding data and control words in the Exchange Memory. When the unit is ready to operate, the Exchange selects a channel through the Cross-point Switch. From then on the channel controls retain the address of the external unit to which the channel is connected; this permits the channel to gain access to the appropriate Exchange Memory location for every byte.

The READ or WRITE instruction also contains the address in main memory of the initial control word to be used by the Exchange. This control word address is first sent to the Exchange which then obtains the actual control word from memory. Moreover, when one control word is exhausted, it can cause another control word to be obtained from memory automatically.

The Exchange executes several functions, other than reading or writing, which have to do with external units. These will be described fully in later sections.

2.2 Instruction Control of External Units

The method of program control to be described in this section applies to all external units. The instructions follow the format of the arithmetic mode of operation in the computer.

When instructions apply to external units, the computer executes all address modification. It then sends the addresses and the decoded operation to the Exchange, which completes the execution of the instruction by obtaining the operand (control word) from memory and starting the external unit. This procedure permits the Exchange before it accepts an instruction to determine from its stored status indication bits whether the unit is ready, and to sandwich in the extra cycles necessary to start an operation into available time periods. If the unit is not ready (for instance, if the operator has stopped the unit manually for an indefinite time), the Exchange rejects the instruction. This permits a

program interruption.

The computer waits until the Exchange has signalled that it has accepted or rejected the instruction. If the Exchange happens to be quite busy, the computer may have to wait some tens of microseconds because it is more flexible in its operation than most external units, which cannot wait. Thus instructions involving the Exchange may take somewhat longer than regular instructions of a similar kind.

The computer never waits for the external unit to respond or to finish the operation, which may take milliseconds to minutes. The Exchange takes over full control and signals the computer when the operation is completed. The Exchange absorbs not only regular starting delays and operating times, but it also waits for a tape unit to rewind or space over a block, for a printer to skip lines, etc., before initiating a new operation. Thus an instruction which has been accepted by the Exchange may remain in the Exchange for some time before the operation can actually begin.

The basic principle is that the Exchange can accept an instruction from the computer just as soon as the previous operation for the same unit has been completed. This permits the computer program to continue without being held up by any delays which must be expected with external units. The Exchange will hold an instruction, if necessary, to avoid tying up a channel while a tape unit is rewinding, etc. The Exchange cannot, however, accept more than one instruction for a given unit at a time. Any attempt to do this results in a program interruption.

2.2.1 Control Word

READ and WRITE instructions include the address of a control word in memory to be used by the Exchange in executing the operation. The control word, as stored in memory, contains the address of the first data word in memory and the number of words to be transferred. Subsequent words are sent to consecutive higher memory addresses which the Exchange obtains by repeatedly adding one to the data word address in the control word.

Basically the control word follows the same format as the index words used by all instructions in the arithmetic

mode. Thus the same word may serve as a control word while reading a block of data, as an address modifier giving the base address of the block while computing, and, again, as a control word while writing that block.

The data word address is contained in the word address portion of the Value field, the bit address and sign portions being unused (see Figure 2. 2).

The word count consists of 15 bits in the Length field starting at the eighth bit from the right. The seven bits on the right of the Length field, corresponding to the number of bits and the sign, are left unused, as are the five bits on the left of the field. The Exchange is thus restricted to counting full words, up to a maximum of 2^{15} words. Counting in the Exchange consists of subtracting one from the word count each time a word is transferred until the count reaches zero.

Although the control word can only specify blocks of information starting at the left end of a full word in memory and containing an integral number of words (i. e., a multiple of 8 bytes), an external unit may terminate operation at any time before the word count reaches zero. If during reading the last word has not been completed, the Exchange finishes the shifting process, filling in zero bits on the right of the last word, before sending the word to memory.

The Index Tag (XT), at the extreme right, is the same tag bit which is available to the programmer during regular indexing for causing a program interruption. In the control word the Index Tag may be used to instruct the Exchange to obtain another control word after this one.

The Index Tag is set to 0 when a single control word defines the block of information for reading or writing. Sometimes it is desired to write a block which is pieced together from several scattered memory areas (grouping), or to read different portions of an incoming block into different memory areas (distribution). To do this, a different control word is required for each memory

area. Each control word gives the first address and the number of words in its area, and all but the last have their Index Tag set to 1.

For grouping and distribution, READ and WRITE specify the first control word to be used. The Exchange can be set up to obtain a new control word automatically when the word count on the current control word has reached zero. For this purpose the program must insert in each control word, in the otherwise unused center portion, the 20-bit address of the next control word. (See bottom of Figure 2.2). This is accomplished by a variable field length STORE instruction. The Index Tag must be a 1. This process continues until the last control word, with an Index Tag of 0, is encountered; when the last word count is exhausted, the Exchange signals the end of the operation.

The Next Control Word Address field overlaps the normal Value and Length fields in positions which are not used for control words but which could affect regular indexing. If there is interference between the two word formats, control words to be arranged for grouping and distribution should be set up separately from the words used for indexing. Such conflict can usually be avoided, however, because of the nature of indexing for input-output data. Control words provide the base address for reference to blocks of input-output data. For base addresses the Value field will always contain only full-word addresses, leaving zeros in the right-most seven bits; the Length field is not needed for indexing because base addresses are generally not subject to incrementing. Interference is thus avoided whenever the control words used for grouping and distribution are stored at memory addresses below 2^{13}, so that the high-order seven bits of the Next Control Word Address may also be left at zero.

The grouping and distribution feature is not available with high-speed tapes and disks because of time limitations.

Seven status bits are shown in Figure 2.2 near the right end of the control word in unused portions of the Length field. These are inserted automatically by the Exchange in the Exchange Memory to indicate the current status

of the corresponding external unit. Normally, programs do not require access to the status bits. Provisions have been made, however, for a program to interrogate the current control word during the progress of an external operation. The current status bits, defined in another section, will then be available to the program.

2.2.2 READ
WRITE

These instructions initiate a reading or writing operation. The Second Address specifies the external unit and the effective Word Address specifies the first control word to be used. The control word, in turn, supplies the information defining the data addresses in memory, and the maximum number of words to be transferred. Immediate addressing may not be used.

A block of information is terminated either by a signal from an external unit or by the word count in the last control word reaching zero, whichever happens first. If termination by word count occurs before the unit has reached the end of a block, the unit continues to the end by itself, independently of the Exchange, before permitting another operation to start.

2.2.3 CONTROL

The effective Word Address is sent as control information to the external unit specified by the Second Address.

The control information is decoded by the external unit to perform such functions as:

Rewind tape
Backspace tape
Space or skip on printer carriage
Turn on RESERVED light
Turn off RESERVED light

A single byte at the left end of the Word Address is sufficient to specify control for most units; the remainder is then ignored by the unit.

2.2.4 LOCATE

The effective Word Address is sent as an address to the external unit specified by the Second Address.

This instruction is used to set up external addresses on disk units, electronic printer-plotters, automatic tape cartridge changers, and similar devices. Access to the specified location is initiated, but the READ or WRITE instruction may follow at any time, whether access has been completed or not.

2.2.5 DISCONNECT

The Exchange will accept a DISCONNECT instruction while a READ, WRITE, CONTROL, or LOCATE instruction is still in progress for the external unit specified by the Second Address of the DISCONNECT instruction. Any operation involving that unit is terminated immediately and the Exchange is released. The unit continues by itself to the end of its block.

This instruction permits the program to free Exchange and memory facilities when the program has determined that the external unit should not or cannot complete a transfer of information started by a READ or WRITE instruction.

2.2.6 Control Word Interrogation

Any instruction in the arithmetic mode of program operation, which calls for a word from a specified location, may give the address of an external unit. The computer sends this address to the Exchange and causes the Exchange to transmit the current control word corresponding to that unit over the data bus; the word is stored in the Exchange at that instant. Thus, the control word will contain the current values of the data address and the word count, the next control word address, and indications of the status of the external unit. Interrogating the control word does not interfere with any operation in progress using that control word.

For example, specifying an external unit in the Second

Address of a TRANSMIT instruction or in the effective Word Address of a RECEIVE instruction, permits the current state of the control word to be stored in any memory location or in any register for subsequent operations. LOAD may be used to enter the word in Register A so that the status bits may be interrogated by means of conditional branch instructions. VFL instructions such as ADD and COMPARE also qualify. The decoding is done in the computer, and all these operations give rise to the same action in the Exchange.

Only a source address can specify a control word in this manner. Information cannot be returned to the Exchange by giving a destination address. Thus STORE and ADD TO MEMORY are instructions which do not qualify. Only READ or WRITE can enter a control word into the Exchange in the manner described in earlier sections.

2.3 Indicator and Status Interrogation

Whenever the Exchange finishes an operation, successfully or otherwise, it causes suitable Indicator Bits to be sent to the Indicator register together with the identification of the external unit, which appears in an External Unit Address register in the computer. The indicators will cause a program interruption if the Interrupt mechanism has been properly set up, thus permitting the program to take immediate action if desired.

In an alternate mode of operation, one or more of the indicators may be masked out, thus preventing program interruption, from these causes. Interrogation of the indicators may be performed at intervals. This mode may cause delays in external operations.

The purpose of the Interrupt system is to avoid the need for programming frequent interrogations during normal external operations. If it is desired to monitor the progress of a unit or to discover which units are ready to be used, the program can interrogate the current Status Bits residing in the control word location in the Exchange Memory.

The Indicator and Status Bits are defined below.

2.3.1 Exchange Indicators

The bits listed below are turned on whenever the Exchange rejects an instruction just given. Program Interrupt is available before the computer proceeds to the next instruction, and this defines the instruction at fault.

Interrupts caused by Exchange indicators do not interfere with any External Unit Indicators.

a) Select Reject

An instruction was given for a unit which was still selected as a result of a previous instruction. This rarely happens in normal programming.

b) Not Ready Reject

An instruction was given for a unit which was not in condition to be operated, i.e., the unit was Not Ready. A Not Ready Reject may usually be avoided by testing the Not Ready Status bit in the control word before issuing the first instruction to the unit.

2.3.2 External Unit Indicators

Whenever one of the indicator bits listed below is turned on, the address of the external unit concerned appears in the External Unit Address register.

Whenever one of these bits is interrogated, by an Interrupt or otherwise, it is immediately turned off. Whenever the External Unit Address register is interrogated and no further external unit indicators are on, an indication is given to the Exchange that another external unit Interrupt will be accepted by the Indicator Register.

Only one external unit Interrupt is sent to the computer at a time. If another interruption occurs before the first can be accepted by the computer, the Interrupt conditions

are stored by the Exchange for later presentation. Delayed interruptions are presented to the computer as if they had just occurred. In this manner, all external unit interruptions can be cleared without conflict. Delays can be avoided, or at least minimized, by suitable programming.

a) Normal End

This is the normal end-of-message signal indicating that an operation initiated for the unit has been successfully completed. This indicator being On implies that all other external unit indicators are Off.

b) Operator Signal

An operator's signal has been received from the indicated unit. This signal has no functional significance. It is interpreted by programming, in whatever manner is desired, to establish communication between the operator, who attends to a unit, and the computer.

c) End of File

The indicated unit has reached an end-of-file condition.

d) Cancel

The last operation initiated for the indicated unit has been terminated without success. This indication does not include data error. One use for this indicator is with inquiry stations to permit an operator to wipe out a partial entry by means of a Cancel key. This indicator also comes on after a DISCONNECT instruction has been given, to

indicate completion of that operation.

e) Data Error

The last operation initiated for the indicated unit has been terminated by a data error.

2.3.3 Status Bits

The status bits appear in bit positions 55 to 61 of the control words stored in the Exchange Memory. They are defined to perform specific functions in the Exchange and they may not all be useful to the program.

If any of the last five bits, 57 to 61, is on, the Exchange will reject further READ, WRITE, CONTROL, or LOCATE instructions for thst unit.

Bit 55 Assign Channel

The Exchange is waiting to assign a channel to the unit. This implies either that all channels are currently in use or that the unit is still busy finishing an operation independently of the Exchange.

Bit 56 Data Error

This bit corresponds exactly to the Data Error Indicator of the previous section.

Bit 57 Not Ready

The unit is not in a condition to be operated from the computer. This bit combines conditions such as: out of material, stacker full, operator stop, power off, control error, and mechanical malfunctioning.

Bit 58 Select for Read

A READ instruction is in progress.

Bit 59 Select for Write

A WRITE instruction is in progress.

Bit 60 Select for Control

A CONTROL instruction is in progress.

Bit 61 Select for Locate

A LOCATE instruction is in progress.

2.4 External Units

This section is restricted to a description of some of the units which are of design similar to existing units. As new units are developed, they will follow the same pattern of communication.

High-speed tape and disk units are under development at the present time for incorporation into the initial computer system. They are not described here because their detailed operating characteristics cannot yet be specified. The method of program control will, however, be the same as that used with the units which are described here.

External units are identified by unique addresses which are different from general memory and register addresses. The external unit addresses fall into the range (decimal) of 256 through 511.

The same addresses are also used to identify the corresponding control words stored in the Exchange. The distinction is evident from the nature of the instruction. If the Second Address of a READ instruction, for example, is 260, external unit #260 is referred to. If the Second Address of a TRANSMIT instruction is 260, the control word currently in the Exchange which belongs to unit #260 is transmitted over the data bus.

2.4.1 Card Reader

The card reader operates at a rate of 1000 cards per minute. It handles 80-column punched cards containing any kind of numeric, alphanumeric, or binary punching. Cards are fed row by row, 9-row first, past two reading

stations. At the first station all holes punched in the card are counted. At the second station, the information is read and transferred in 12 bursts of 10 bytes each to the Exchange. Another hole count is made and compared with the count previously obtained at the first station for a thorough check. From there on, the information continues to be checked by the regular parity checking equipment in the system.

A complete image of the punching in the card is created in memory, using a bit for each of the 960 punching positions, with a 1 for a hole and a 0 for no hole. This image occupies 15 words in memory. The first word (at the lowest address) consists of the "9"-bits of columns 1 to 64. The second word contains the remainder of the "9"-row and the "8"-bits of columns 1 to 48, etc.

2.4.2 Card Punch

The card punch operates at a rate of 155 cards per minute. The unit is mechanically similar to the IBM 535 punch which is used with the IBM 608 calculator.

The card punch creates any kind of 80-column card. Cards are fed row-by-row, 9-row first, past a reading station which is left unused in this device, then past the punching station, and finally a second reading station to check the punching. Like the card reader, the punch operates from a 15-word image in memory of the card to be punched.

The information sent to the punch is checked by means of parity bits. There the one-bits are counted as the card is punched. The card is read at a station following the punching station, and the number of holes in the card is compared with the previously accumulated count to obtain a thorough check of the punch.

The cards to be punched must be blank to begin with; otherwise the punching check will fail.

2.4.3 Printer

The printer operates at a rate of 500 lines of 120 characters per minute. The unit is mechanically similar to the

IBM 720 Wire Printer.

The printer is supplied with information from a memory area of 15 words containing 120 bytes of 8 bits each. The information is completely edited beforehand by the computer and translated to the 6-bit character code used by the IBM 720 (which is the same code as that used in the IBM 705 system). The 6-bit character code occupies the low-order (right-hand) bits of the 8-bit byte; the two high-order bits are zeros.

The nature of the printing mechanism requires that the printer receive first, every fourth character starting on the left, then, every fourth adjacent character, and so on. For this reason the characters to be printed must be arranged in an interleaved fashion in memory:

1, 5, 9, 13, ..., 113, 117, 2, 6, 10,

14, ..., 114, 118, 3, 7, ..., 116, 120

The numbers refer to the printing positions along the line. This arrangement can be readily programmed, simultaneously with character code translation, by means of the streaming mode.

Automatic single or double spacing for each line of printing is set up by CONTROL instructions. When other than single or double spacing is desired, a CONTROL instruction to skip to another line must precede the WRITE instruction which initiates the printing of that line.

2.4.4 Magnetic Communication Tape

The Magnetic Communication Tape (COT) units are used for reading and writing tape at 15,000 characters per second. A "character" consists of 6 information bits. The unit is mechanically similar to the IBM 729 tape unit with dual-gap heads, and it handles tapes compatible with those used on IBM 727 Tape Units in conjunction with existing auxiliary equipment.

On reading, each 6-bit character is entered into the Exchange in the low-order 6 bits of an 8-bit byte; the left

two positions are filled with 0's. If the number of characters in an incoming block of information is not a multiple of 8, the Exchange fills the remainder of the last word with zeros.

On writing, only the low-order 6 bits of each 8-bit byte are written, starting at the beginning of the first word. Blocks must consist of whole words, that is, multiples of 8 bytes.

COT units are intended primarily to read and write tapes which can be used on existing machines equipped with IBM 727 or 729 Tape Units. Since both odd-count and even-count parity (redundancy) checking is used on existing equipment, CONTROL instructions are provided for COT units to set up the odd- or even-count mode. When in the even-count mode, bytes consisting of only binary zeros cannot be written since they leave no recorded spots on the tape.

For special high-volume applications, a converter to change four consecutive 6-bit bytes to three 8-bit bytes, and vice versa, is planned. This converter would facilitate the entry of long streams of binary information or other differently coded data from COT tape.

2.4.5 Operating Stations

There will be an operator's panel on the computer which carries Power On and Off buttons and lights, initial program loading controls, and controls needed for computer maintenance.

In addition, there will be a separate console consisting of a number of keys and lights, and a bell. The meaning of the keys, lights, and bell is defined solely by the program. The console is connected to the system like any other input-output unit. The keys are scanned and the bits entered as input into the system on a signal from the operator. The lights and bell are set up by the computer as if they were output bits.

A complete operating station consists of such a console together with an inquiry station (keyboard for input and

typewriter for output). The computer may be operated with none, one, or several operating stations at nearby or remote locations.

2.4.6 Controls and Lights on External Units

(The following controls and lights apply to card readers, punches, printers and tape units, with the exception of printer carriages and automatic tape loading devices. Carriage controls are conventional. Tape loading controls have not yet been specified.)

a) Controls

1. SIGNAL

This key permits an operator to request computer intervention by causing an automatic program interruption. The address of the unit at which the Signal was given is made available to the program. The Signal has no direct function in the external unit.

2. START

This key turns on the Ready status if all the necessary operating conditions are satisfied.

3. STOP

This key turns off the Ready status at the end of any operation in progress, thus preventing further operation of the unit from the computer.

4. LOAD

This key brings the medium (cards or tape) to the proper starting position, provided Ready is off. (Run-in)

5. UNLOAD

This key brings the medium (cards or tape) to the point where it can be manually unloaded. (Run-out)

6. RESET

This key clears interlocks set by error (reject) signals.

b) Lights

1. STOP

This light is On when the power is on but the unit is not ready to operate under computer control. It is Off when Ready goes on. It comes on when the STOP key is pressed.

2. READY

The unit is ready to operate under computer control.

3. BUSY

The unit is busy executing a computer instruction.

4. RESERVED

This light is turned on and off by the computer program. It has no predetermined meaning, but its main purpose is to notify the operator that the unit has been reserved for use by the computer on a previously assigned program and that it should not be assigned to another program.

5. EMPTY

The unit is out of paper, at end of tape, out of cards (hopper empty), etc.

6. FULL

The stacker of a card reader or punch is full.

7. PROTECT

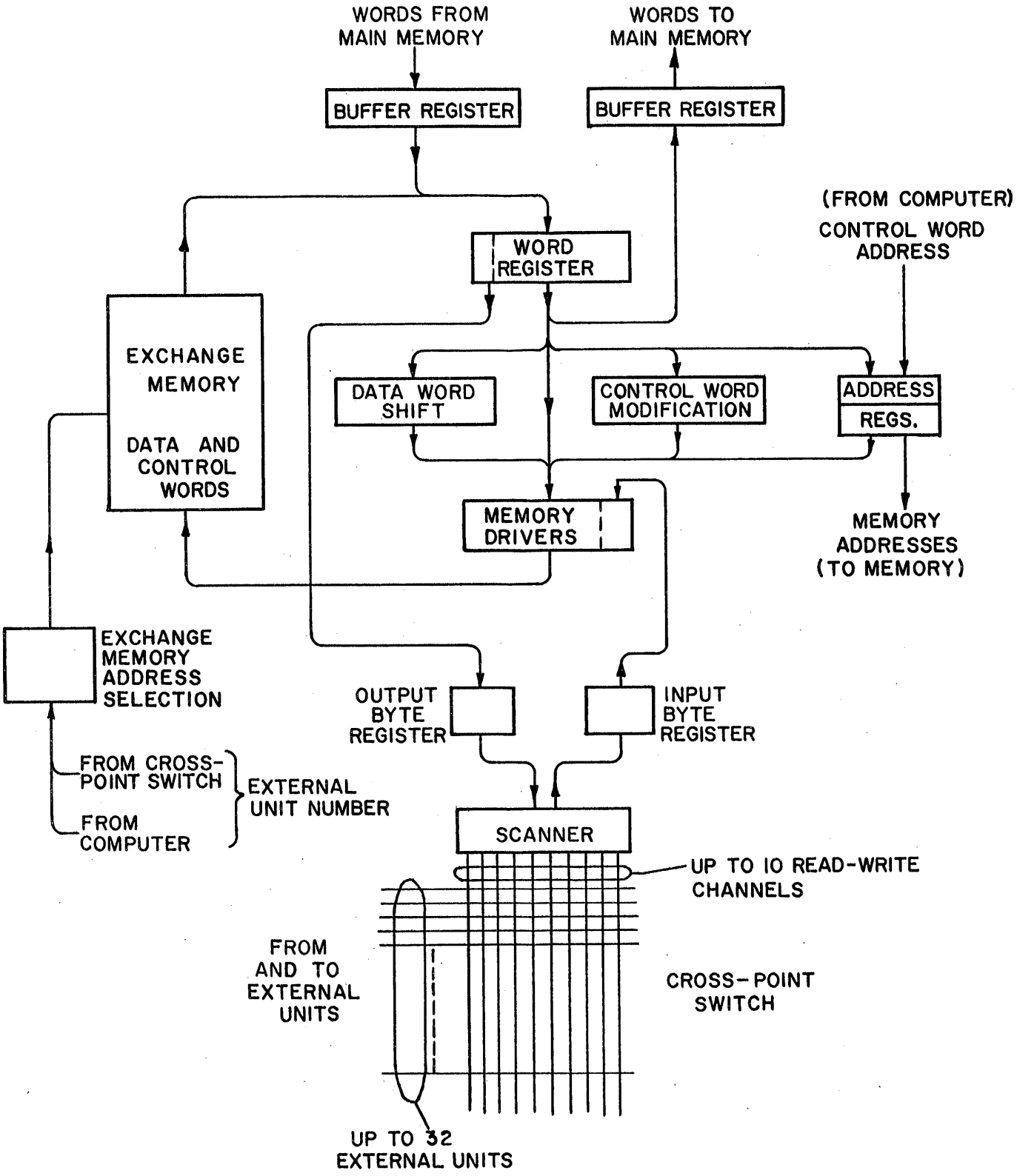
The file protection feature on a tape prevents writing.

8. SIGNAL

Pressing the SIGNAL key turns this light on. The first computer instruction to arrive turns it off. RESET also turns it off.

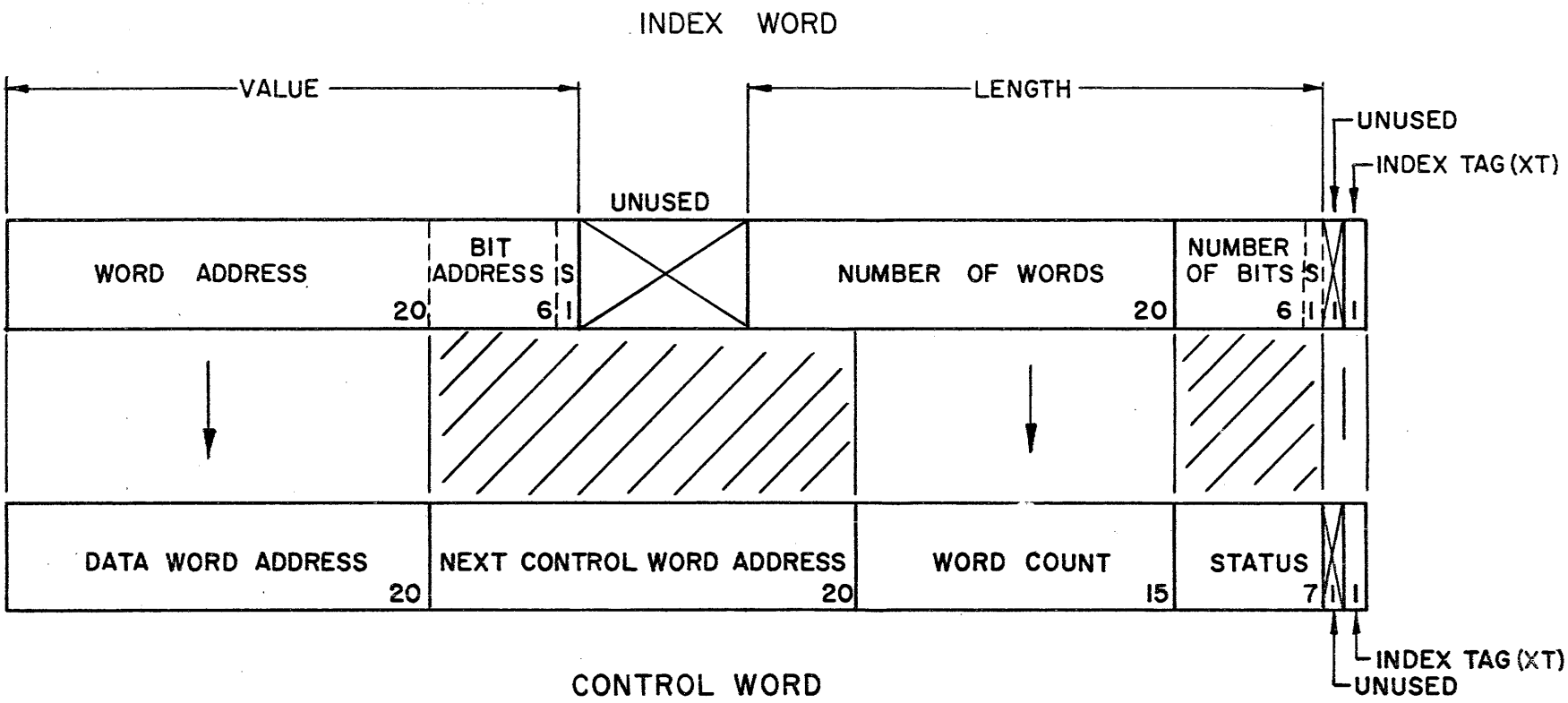
9. CHECK

Separate lights may distinguish between data, control, and mechanical errors detected by the built-in checking devices.



INFORMATION PATHS OF BASIC EXCHANGE

FIGURE 2.1



RELATION BETWEEN INDEX AND CONTROL WORDS

FIGURE 2.2

3. Memory

3.1 There are three types of internal memory in the Harvest system, excluding the special purpose memory in the Exchange. They are:

a) The 0.5 microsecond cycle type, in units of 512 words

b) The 2.0 microsecond cycle type, in units of 8,192 words

c) The 0.1 microsecond transistor registers

The 0.5 and 2.0 microsecond units are magnetic arrays.

The data word size in the memories is 64 bits. Additional bits are provided for checking. The bus system, connecting the memories with the remainder of the system, transmits 64-bit data words in parallel.

3.2 Memory is addressed in binary notation. 20 bits of address are provided in the instruction to address a memory word; the address capacity is thus 2^{20} or 1,048,576 words. The Harvest system will not initially be provided with this amount of memory. However, to facilitate the later expansion of memory capacity, the various types of memory have been assigned blocks of memory addresses as follows:

a)	1 through 255:	0.1 microsecond registers, (including computer registers)
b)	256 through 511:	Exchange memory and Input-Output units
c)	2,048 and up:	0.5 microsecond memory
d)	32,768 and up:	2.0 microsecond memory

3.3 The 2.0 and 0.5 microsecond memories will be grouped in banks of four independently operable memory units (provided four are available). For these banks, the address assignments will be such that the two lowest ordered bits of the word address will select one of the four memories of a particular bank. To

explain further, if an address counter starts at 2048 and is caused to step by one for six steps, the 0.5 microsecond memories addressed would be:

0. 2048 -- Memory 00

1. 2049 -- Memory 01

2. 2050 -- Memory 10

3. 2051 -- Memory 11

4. 2052 -- Memory 00

5. 2053 -- Memory 01

6. 2054 -- Memory 10

3.4 In addition to the normal ability to clear one word of memory per instruction, one can clear a block of words in a particular memory unit by a single instruction in about one memory cycle. The block sizes are 8 and 64 words for a 0.5 microsecond memory unit, and 64 and 1,024 words for a 2.0 microsecond memory unit. The blocks are non-overlapping for a particular size; they are addressed by specifying any word of the block, for either size block.

The addresses within a clearable block are spaced four apart. To clear consecutive addresses, the corresponding blocks within the other three memories of a bank must be cleared by giving the instruction four times, incrementing the address by one.

3.5 The 2.0 microsecond memory has five types of memory cycles. These are:

a) Read -- Transfer of data from memory to the designated register

b) Write -- Transfer of data from a register to memory

c) Or -- Inclusive Or-ing of a memory word with a word formed by the Table Address Assembler

d) Read and Or -- A combination of operations (a) and then (c)

e) Clear -- Clearing of the block designated by the base point address

3.6 The 0.5 microsecond memory has five different types of memory cycles. These are:

a) Read -- Transfer of data from memory to the designated register

b) Write -- Transfer of data from a register to memory

c) Count -- Adding the contents of memory to a word formed by the Table Address Assembler

d) Read and Count -- A combination of operations (a) and then (c)

e) Clear -- Clearing of the block designated by the base point address

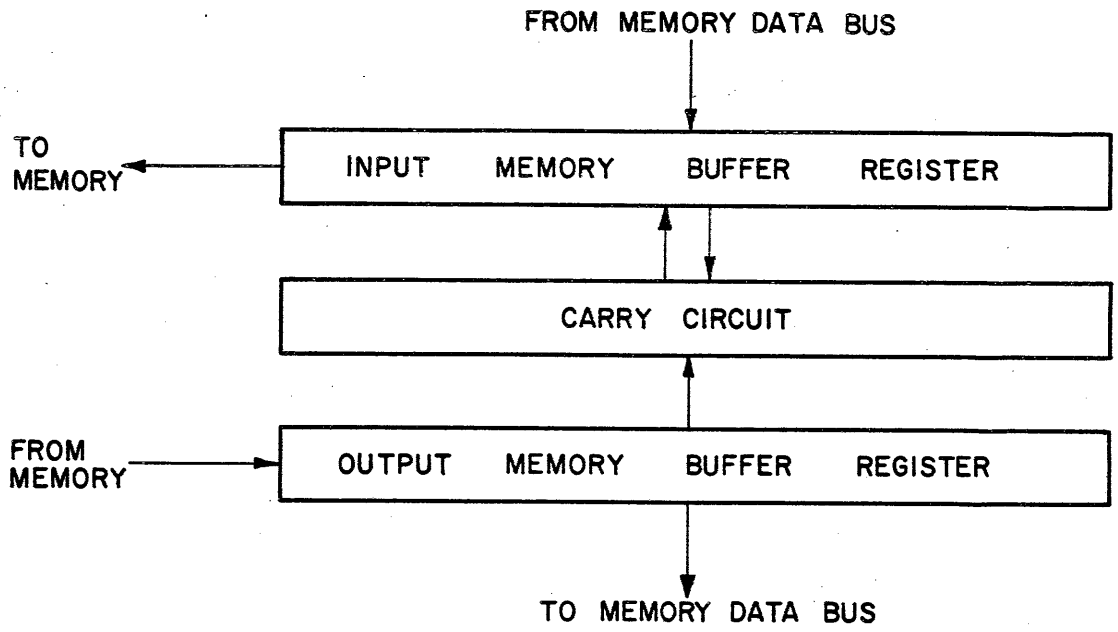
3.7 The Counting and Or-ing in memory is implemented by the type of logic shown in Figures 3.1 and 3.2. This logic causes the Counting or Or-ing cycles to be very similar to the basic Read and Write cycles except that additional time is likely to be required.

In the Read cycle, the word from the memory register enters the Output Memory Buffer register and is restored in memory via the Input Memory Buffer register. The output register holds the data until the memory bus transfer is made.

In the Write cycle the data word to be stored is entered into the input register at the same time as the address is entered in the Memory Address register. The memory cycle is initiated and the data is stored.

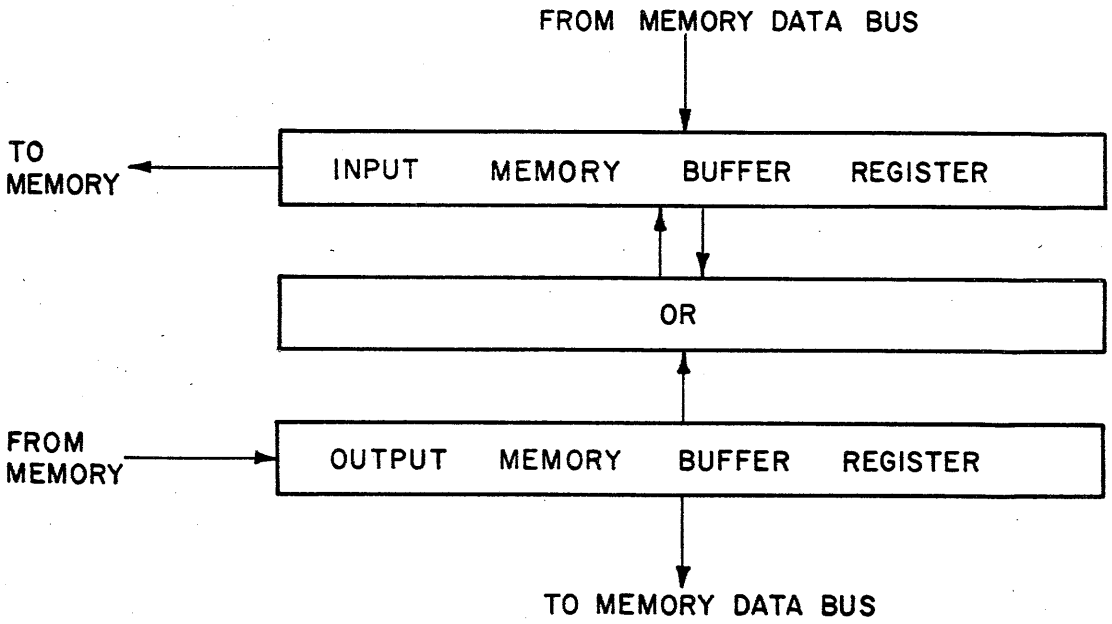
In the Count or Or cycles, the word containing all zeros except for one bit enters the input register as in a Write cycle. A Read type cycle is initiated in that the contents of the memory register are entered into the output register and the contents

sent to the Table Extract Unit if an extraction is to be performed. Independently of the transmission to the Table Extract Unit, the Or or Count is implemented. At the completion of this phase, the Write portion of the cycle is initiated and the revised word is stored in the addressed memory location.



MEMORY BUFFER REGISTER DATA FLOW WHEN COUNTING IN MEMORY

FIGURE 3.1



MEMORY BUFFER REGISTER DATA FLOW WHEN OR-ING TO MEMORY

FIGURE 3.2

4. The Harvest Computer

The computer unit of the Harvest system consists of a number of different, asynchronous units, 8-bit data busses connecting these units, 64-bit data busses between some of these units and the main bus system, and an array of controls and control lines interconnecting all of the computer units with themselves and with the other major parts of the system, i.e., Memory and the Exchange. Figure 4.1 shows the various units of the computer and the data lines connecting these units.

Registers R_1 through R_6 and R_r through R_s are identical 64-bit registers used in conjunction with the four switch matrices to disassemble 64-bit data words into 8-bit bytes or to assemble 8-bit bytes into 64-bit words. Their use is described in some detail in the Stream Unit and Table Extract Unit sections that follow.

The registers R_j through R_k and R_m through R_n are basically identical in nature to registers R_1 through R_6, but in some cases do not contain a full 64 bits of storage. They are used in conjunction with indexing (i.e. address modification) operations and with instruction look-ahead and break-in operations. The indexing arithmetic unit is described in the Indexing Units section that follows. The use of registers R_j through R_k is described in Chapters 5 and 6 under instruction control.

The layout of Figure 4.1 illustrates a basic design philosophy in the Harvest system. Every effort is being made to standardize units such as registers, adders, counters, and switches in order to simplify design and maintenance and to permit a substantial increase in flexibility by the presence of several units which accomplish identical functions.

4.1 Stream Units

4.1.1 The Stream Unit (SU) stands between the memory, which can emit or accept only full 64-bit words, and the arithmetic and logical organs which are designed for 8-bit words. It differs from more familiar registers in having a much more versatile switching facility and in having a certain amount of automatic control provided for its data input and output.

The switching facility in the Stream Unit provides a flexible and convenient way of handling less-than-full-word data units or bytes by allowing direct selection of the byte rather than the indirect "splitting off" process. Because the 64-bit word boundaries would appear to the byte user as artificial and annoying constraints, the memory is thought of not as consisting of 2^{20} words of 2^{6} bits each but rather of 2^{26} individual bits. Accordingly a byte is selected by giving its bit address, a 26 bit number. A byte may then be anywhere within a word and in particular may be partly in the end of one word and the beginning of the next.

The basic size was chosen large enough to express an alphanumeric character in parallel and at the same time be a power of 2. Byte sizes of less than 8 bits may be specified by masking operations subsequent to the SU selection, and fields of more than 8 bits are handled by successive bytes.

The streaming mode mentioned in the General Description is facilitated by an automatic indexing and memory reference control mechanism associated with each unit. The utility of this automatization depends upon the existence of regular patterns in the successive addresses from which the programmer wishes to obtain bytes. A discussion of some of these patterns and the mechanism for generating the appropriate sequence of addresses is given in the section on SU indexing. Hence the procedures and equipment involved in indexing will not be described here, even though they are employed in streaming operations.

Each 26 bit address presented to the SU by the indexing

mechanism is used in two different ways. In principle, the most significant 20 bits are used to select a memory word to be read into the register while the least significant 6 bits are used to select the starting bit position of the byte within that word. In practice, an additional register is provided into which the following memory word is read in case the byte referred to extends into it, and also to maintain smooth flow as the selected byte pattern moves through the first word and into the following.

4.1.2 Figures 4.2 and 4.3 show a byte selection mechanism which makes it possible to select a byte consisting of any 8 consecutive bits among the bits contained at the moment in the two-word register mentioned above. The byte selection mechanism also provides checking by means of simple parity. This scheme could be elaborated to include higher levels of parity for error correction if necessary. The selection is done in two levels to effect a considerable saving in equipment.

The first of the two levels of selection chooses one among 8 overlapping sets, each of which consists of 16 consecutive bits from among the bits contained in the two-word register and supplies these 16 bits, together with a parity bit, as input to the second level of selection. In the second level, any desired 8 consecutive bits are selected from among the 16 information bits which entered the level and these 8 bits form the basic "selected byte"; these 8 can be cut down to a byte of smaller size by a suitable masking operation after leaving the byte selection matrix.

Figure 4.2 shows the first-level selection matrix and the associated parity generating equipment for one word of 8 bytes. Most of the parity equipment would be needed anyway to check words read into the SU from memory.

It is assumed that the word from which the byte selection is to be made is already in the register. The path along which it came from the memory is not shown in the figure.

The word is thought of as being separated into bytes which are denoted by B_0, B_1, B_2, ..., B_7. The finally

selected byte need not, of course, coincide with any of these, but may consist of any 8 consecutive bits chosen from among the bits contained in the register. Each of the bytes B_i consists of 8 bits, each of which is connected to one of the vertical lines of the first-level selection matrix. Between the 8 vertical lines corresponding to the bits of B_i and the 8 vertical lines corresponding to the bits of B_{i+1} there is another vertical line which comes from $P_{i,\ i+1}$ which is the parity bit obtained by combining P_i with P_{i+1}, where P_i is the parity of Byte i. As can be seen from the figure, the parity for the whole word is also formed at the same time.

The first-level selection matrix also has 17 horizontal lines over which the 16 selected output bits of this level and the corresponding parity bit pass to be used as input to the second level.

Just which 17 vertical lines furnish the bits which appear as output on the 17 horizontal lines is determined by which of the 8 diagonal lines is energized. This, in turn, is determined by sending 3 of the 6 bits from the byte address to the First Level Decoder which then causes the proper diagonal line to be energized. It should be remembered that the byte addresses are furnished to the SU by the indexing equipment. Although the sketch of the first level matrix shows 17 x 72 intersections, only those marked by heavy dots involve any equipment. Each intersection may be thought of as an AND element which gives an output on its corresponding horizontal line when and only when its corresponding diagonal line and its corresponding vertical line are energized. Thus any one of 8 different overlapping sets of 16 consecutive information bits can be selected from among the bits present in the original word and sent, together with a corresponding parity bit, for use as input to the second-level selection matrix. It is clear from the figure that any 8 consecutive bits among those in the register will appear in at least one of these overlapping sets of 16 bits since successive sets overlap by 8 bits.

The remaining bits of the byte address pass from the

Byte Address Register to the Second Level Decoder where they select one of the 8 diagonal lines of the second-level selection matrix and cause this line to be energized. It should be noticed that the short diagonal lines at the upper right corner of the matrix are actually continuations of the longer diagonal lines below and to the left. Thus, each diagonal line actually passes through 17 dotted intersections. These dotted intersections have the same meaning as before. It is clear from the figure that selecting a suitable one of the diagonal lines to be energized will make it possible to use any 8 consecutive bits (among the 17 which enter the matrix) as the 8 bits of the "Selected Byte". The remaining 9 bits form the "Residue Byte". The 8 bits of the Selected Byte form the final output of byte selection equipment.

The parity P_S of the selected byte and the parity P_R of the residue byte are both formed and combined with each other to give the parity P of the whole 17-bit piece. Since a parity bit was included among the 17 bits whose combined parity is expressed by P, this combined parity will always be constant.

An analogous mechanism in Stream Unit C permits storage of a byte at any arbitrary memory location.

4.2 Table Address Assembler

The Table Address Assembler forms the addresses for the memory references during the table lookup operations. (See Figure 4.4) The addresses are formed by assembling data bytes from the byte busses together with a table base address and, optionally, the contents of a counter. The assembled address may be used in any of the following ways:

a) Counting or Or-ing in memory without table lookup, i.e., without making the contents of the memory register available to the Table Extract Unit

b) Table lookup for extracting a table entry

c) Table lookup combined with counting or Or-ing in memory

Before the unit is operated, the Base Address, Bit Address, Increment, and Length registers are set up.

The bit address selects the position in the Assembler register at which the first byte is to be entered. After each byte is entered, the bit address is incremented by the first level increment. This continues until the length for the first level is exhausted. If the controls indicate that a second source is to supply data also, the Table Address Assembler is switched to the other input and data is received and positioned from the second source until the second length is exhausted. Only two sources are permitted for each address formed.

After the data for one address has been assembled in the Assembler register, the contents of the Assembler register and the Base Address register are added together and entered into the Address Accumulator. The address in the Address Accumulator is then sent to the Word-Bit register where the word portion of the address is sent to memory to start the memory reference. The bit portion of the address is sent to the Table Extract Unit. The Assembler indexing controls are reset.

The Assignment Switches and the Memory Distributor (a four-cycle, 2-bit counter) permit the programmer to sequence through four memories in an orderly fashion and thus gain speed for table lookup. The table must, of course, be repeated in all four memories in corresponding memory cells.

When the Address Assignment Switch is in the off position, the bits from the Word Address register are not shifted. When the Address Assignment Swtiches are in the on position, the bits in the word portion of the address are shifted left two places to permit the contents of the Memory Distributor to be entered into the two low-order bits of the word address. The Distributor advances by one for each address formed.

If Counting or Or-ing to memory is to be done, the bit portion of the address is Or-ed to one less than the table entry cell size. This 6 bit number is then decoded and a one is inserted into the empty Add One register at the corresponding bit position. The contents of the Add One register are sent to memory.

4.3 Table Extract Unit

The Table Extract Unit is used with the Table Address Assembler.

Its function is to extract a table entry from the memory word obtained by the Table Address Assembler. (See Figure 4.5)

The Table Extract Unit consists of a 64 to 8 switch matrix and its index controls and one or more sets of Extract registers and Bit Address registers. The Table Extract Unit is basically similar to a Stream Unit, except that it does not have the ability to extract bytes which cross word boundaries and has only one level of index control. This index control is identical to the first level of Stream Units A, B, and C.

Because several memory references for table extract may be underway simultaneously, a separate Extract register is provided to receive each reference.

Sufficient mechanism will be provided to assure that the words are taken from the Table Extract Unit in the order in which the addresses were formed. The Bit Address registers corresponding to the multiple extract registers receive the bit addresses directly from the Table Address Assembler.

The contents of each table entry are made available a byte at a time by the index control and switch matrix.

4.4 General Indexing

Modification of the Operand Address portion of instructions in the arithmetic mode is accomplished in the General Indexing Unit shown in Figure 4.6.

Three distinct operations are performed in this Unit:

a) Modification of the Operand Address

b) Modification of the index value and the residual Length

c) Testing of the residual Length

All instruction operand addresses are indexable.

The General Indexing Unit contains three registers, each of which is made up of two or more fields:

a) The Instruction register, IR, includes:

1. A Word Address, WA

2. A Second Address, SA

3. An Index Address, IA

4. The Operation Code (which does not enter into consideration here)

The Second Address consists of two subfields:

(a) The Bit Address, BA

(b) The Field Length, FL

For some operations, the entire Second Address acts as a single address; at other times the Bit Address is appended to the Word Address to give a complete Operand Address.

b) The Index register, IX, contains:

1. The Value, V

2. The Residual Length, R

c) The Index Accumulator, IAC, is used to hold any of the following values:

1. The Operand Address

2. The modified Operand Address

3. The Increment

4. A new Value

5. A new Length

A 26 bit adder with sign control is incorporated in the General Indexing Unit. It is used to:

a) Form the modified address

b) Increment the Value

c) Decrement the Residual Length

d) Enter a new V or L into the IAC

The Limit Test mechanism signals whether R is positive, negative, or zero.

4.5 Stream Unit Indexing

Figure 4.7 shows the mechanism necessary for address modification in the Stream Units. Each SU has two built-in levels of indexing plus higher level indexing from memory. The second level, however, shares a pair of 6-bit adders and an Effective Address register with the first level. Moreover, all Stream Units share one pair of 20-bit adders.

The Effective Address register holds the address of the next byte to be read out.

Each indexing level for each SU has an Increment Register, I, a Residual Length Register, R, and a Length Register, L. The contents of these registers are used to modify the Effective Address according to the Control Sequence discussed below.

To keep track of the level in control of the address modification, each SU has a level control mechanism. This mechanism contains a number, X, designating the level in control. Thus "X + 1 replaces X" means control is passed to Level X + 1.

The initialization of the Effective Address Register, and the actual gating of the Effective Address into the Byte Address Register (one for each SU, used to select the desired memory word and the proper byte-line in the Cross Point Matrix) may be thought of as being part of a Zero-th level of control. The Control Sequence is as follows:

Level 0, Step (a) Initialize S

(b) Set X equal to 1

(c) Gate S into Byte Address Register and proceed to Level X, step (a)

Level X, Step (a) Is Read Suppress bit on?

If NO, read byte and proceed to Step (b)
If YES, proceed to Step (b)

(b) Replace R by R - I; replace S by S + I

(c) Is R = 0?

If NO, and Reset Suppress bit is on, go to Level 0, Step (c) (don't reset X)

If NO, and Reset Suppress bit is off, go to Level 0, Step (b) (do reset X)

If YES, proceed to Step (d)

(d) Replace R by L

(e) Is Reset Suppress bit on?

If NO, replace S by S-L (reset S)

If YES, replace X by X + 1 and proceed to Level X + 1, Step (a) (don't reset S)

The Read Suppress and Reset Suppress bits are associated with each level of indexing control. Their use is discussed in Chapter 6.

The following table indicates the number of bit positions (excluding the sign) in each register:

Level	Register			
	S	I	R	L
0	26*	---	---	---
1	---	8	20	20
2	---	26	26	26
Higher	---	26	26	26

*The S Register is the only register without a sign position.

Each register (S, I, R, and L) is divided into two parts; the low order six bits and the two, fourteen, or twenty (depending upon the register) high order bits. The low order bits refer only to the address of a particular bit within a word, while the high order bits refer to the word address itself.

The 20-bit adders are used only when a word address must change. The 6-bit adders are used whenever a bit address is changed. For example, suppose I contains a number less than 64. Then if the contents of I and S are added together, the result will not always mean a new word is to be addressed. However, when necessary a 20-bit adder cycle is captured and a new word is addressed.

4.6 Statistical Accumulator and Statistical Counter

4.6.1 The Statistical Accumulator can accumulate signed or unsigned bytes and test its contents against a threshold. Its mechanism is similar to that used in the first level of indexing for a Stream Unit. (See Figure 4.8) The nomenclature used in describing the indexing equipment is, of course, different from that used here. The correspondence is as follows:

Register in Indexing Mechanism	Function in Statistical Accumulator
S Register	Serves as Accumulator
I Register	Furnishes quantity to be accumulated

R Register	Serves as residual Threshold register
Two 26-bit adders	Add and subtract One for Accum. and one for Thr. Reg.

Modifications

1. Indication when quantity in R Register becomes zero or negative

2. Overflow indication on S Register

3. Indication when sign of S Register changes from + to -

4. Provision to suppress negative totals and reset Accumulator to zero when such totals occur (This is done or not at programmer's option.)

5. Accumulation Mode Control inserted between I Register and both adders

The initial contents of the Accumulator and the Threshold register are set up by means of a RECEIVE instruction. After this is done and the accumulation begins, each time a quantity is added into the Accumulator, the same quantity with inverse sign is added into the Threshold Register. Hence, as the accumulated total reaches or exceeds the preassigned threshold value, the quantity in the Threshold Register reaches zero or becomes negative. An indication of this is then given, allowing interruption if streaming is in progress.

Notice that when the input quantity and the quantity in the Accumulator are added, their sum is returned to the Accumulator. At the same time, the same quantity is subtracted from the Residual Threshold Value and the result is returned to the Residual Threshold Register. If the input quantity has a negative sign and the Accumulator

Mode Control is set for signed operation, the addition and subtraction operations are interchanged.

4.6.2 The Counter is an 18-bit binary counter with overflow indication allowing break-in. Like the Accumulator and the Threshold Register, it has an address and hence it is possible to load it initially or to store its contents by means of an ordinary TRANSMIT instruction. The source of its one-bit inputs is specified in the stream instruction.

4.7 The Logical Unit

The Logical Unit (LU) of the Harvest computer performs a variety of logical and arithmetic operations on a pair or a sequence of pairs of bytes. The streaming and arithmetic operations performed in the LU are described in Chapters 6 and 5, respectively.

In conjunction with each LU output data byte, the LU also generates an associated output control bit whose condition is determined by the operation being performed and by the input data bytes. It may be entered onto the byte bus for transfer to the Statistical Accumulator or Stream Unit C, stimulate a counter, or initiate control operations.

4.8 The Byte Masks

The basic byte size within the Harvest computer is 8 bits. In order to select the byte size needed in an operation, four byte masks are provided. In the streaming mode the programmer explicitly specifies the masks in the setup and may modify them later as required. In the arithmetic mode the specification of certain parameters automatically regulates the masks.

Byte Mask 1 is on the output byte bus from Stream Unit A; BM_2, on the output from B; BM_3, on the input to C; and BM_4, on the output from the Table Extract unit. In streaming each mask is able to select not just a byte size, but any subset of bits from the 8. When the data passes through BM_1, BM_2, and BM_4 those bits corresponding to 1's in the masks go through unchanged; those bits corresponding to 0's are made 0's. BM_3 has a different effect: the bits corresponding to 1's in the mask enter Stream Unit C for storage and eventually replace bits in

memory; the bits corresponding to 0's do not enter C and the memory bits are left unchanged.

The action of BM_1, BM_2, and BM_4 is accomplished by forming the logical And of the data and the mask. This necessitates registers for the mask and registers for the data. (The registers for the data are the same ones used for match recognition at this point.) BM_3, on the other hand, is involved in the actual gating of lines into Stream Unit C and thus is closely associated with its control mechanism.

4.9 Match Units

A Match Unit is a device which compares bytes passing on a byte bus with a prespecified 8-bit byte. Each Match Unit has a register containing the match character to be recognized. The match characters, the associated break-in control data, and the positioning of the Match Units are all specified in the setup of a stream operation.

There are four Match Units (W, X, Y, Z) and four points on the byte busses where Match Units may be connected: Match Stations 1, 2, 3, 4 (Figure 6.1). Each Match Unit can be connected to only one Match Station at any one time; however, any number of units (up to four) can be connected to the same Match Station. Each station contains a byte register. (The register usually is shared with the corresponding byte mask.) When the byte enters the register, it is compared with the match characters in each of the Match Units plugged to this station. Not until all the comparisons are declared negative is the data byte presented to the byte bus. If a match is obtained a signal is sent to the interrupt control and, optionally, to the Index Adjuster. In this second case nothing is put on the byte bus unless the break-in instruction specifies it.

4.10 The Numbering Counter

The Numbering Counter is a 16-bit binary unit counter which may be used to count control signals and transfer the count to the byte busses. This counter has two primary purposes: to provide a sequence of labels for forming records; and to provide the Table Address Assembler with a sequence of numbers in order to visit a series of tables or to count in a series of memory areas.

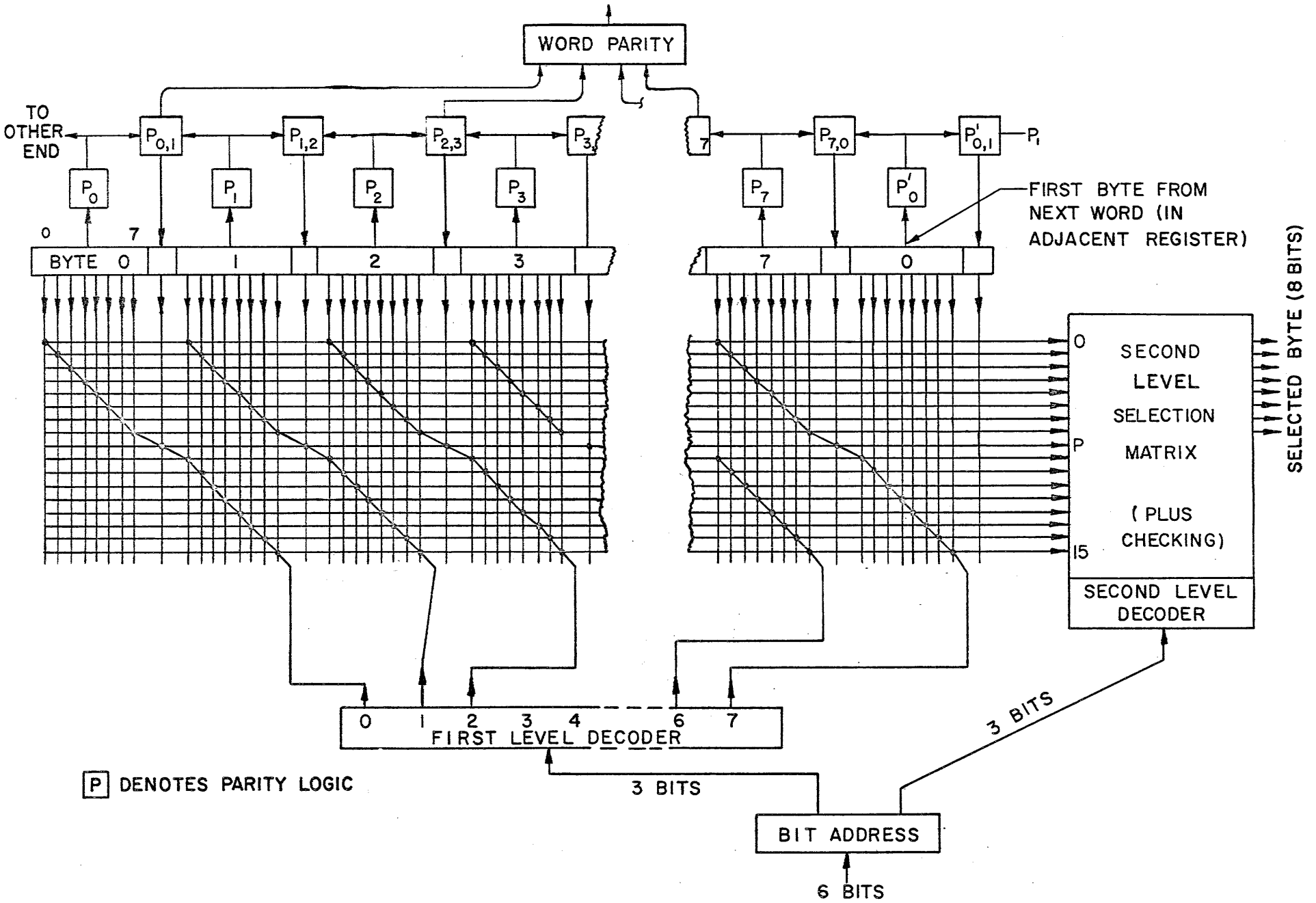
The counter may advance automatically on:

a) An address sent to memory

b) A second level advance of the indexing mechanism of Stream Unit A.

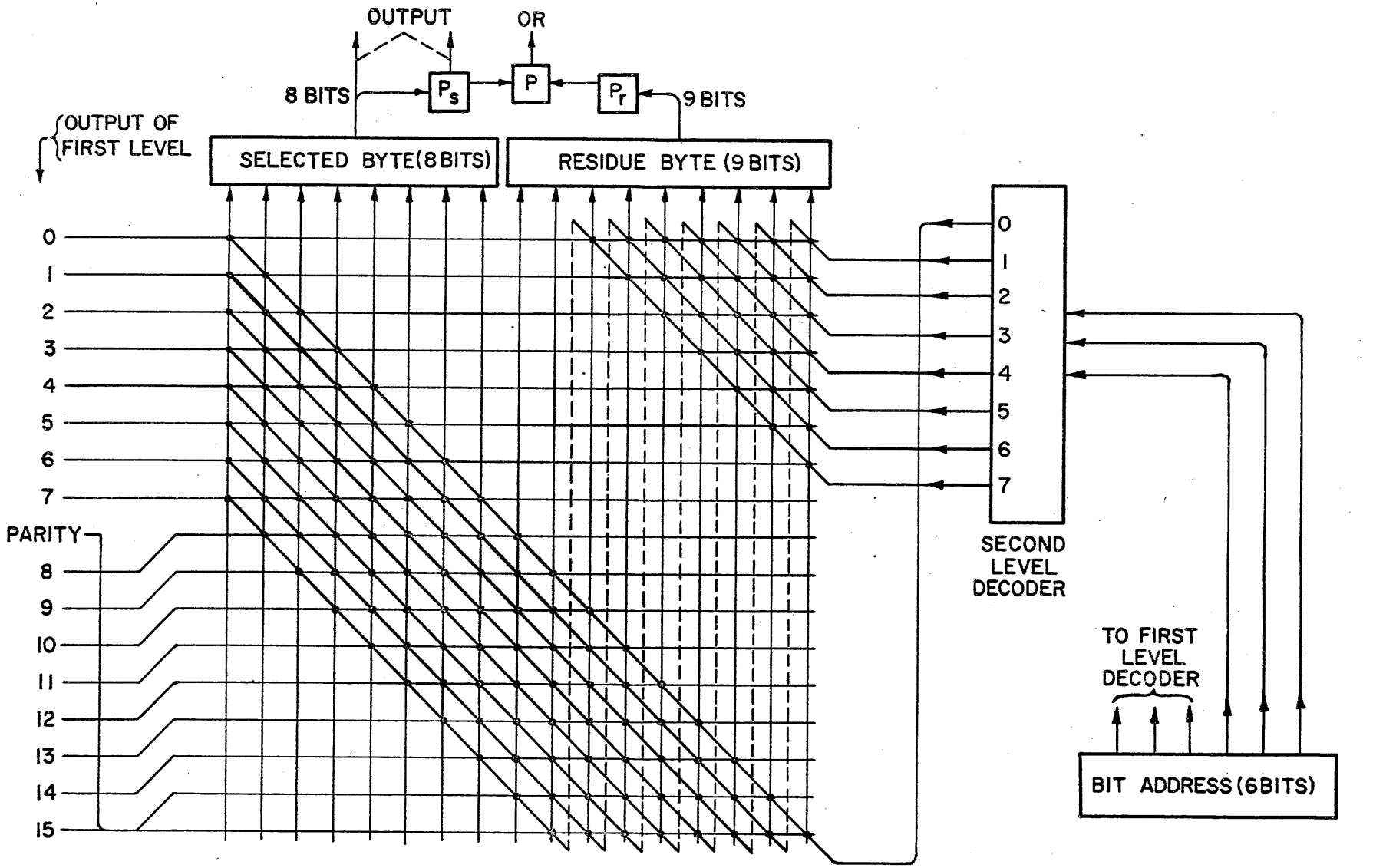
c) A signal from the LU

or may be advanced by regular instruction. The contents of the counter may be read onto the byte bus, under program control, via the Stream Unit B shift matrix. The 16 bits of the counter correspond to the first 16 bits of B. The Stream Unit B indexing controls govern the read out.



STREAM UNIT 2 LEVEL SELECTION

FIGURE 4.2



DETAIL OF SECOND LEVEL SELECTION MATRIX

FIGURE 4.3

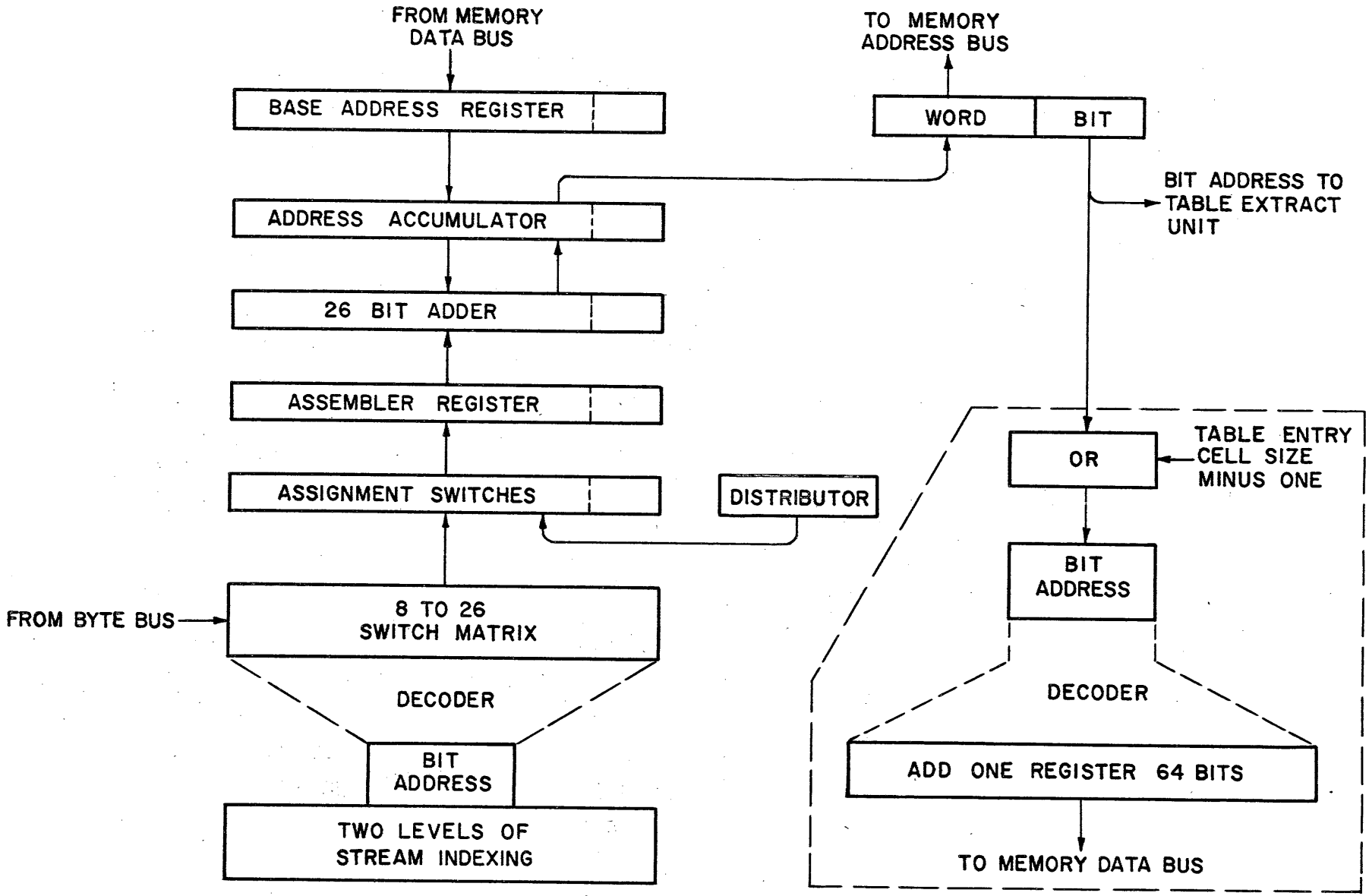


TABLE ADDRESS ASSEMBLER

FIGURE 4.4

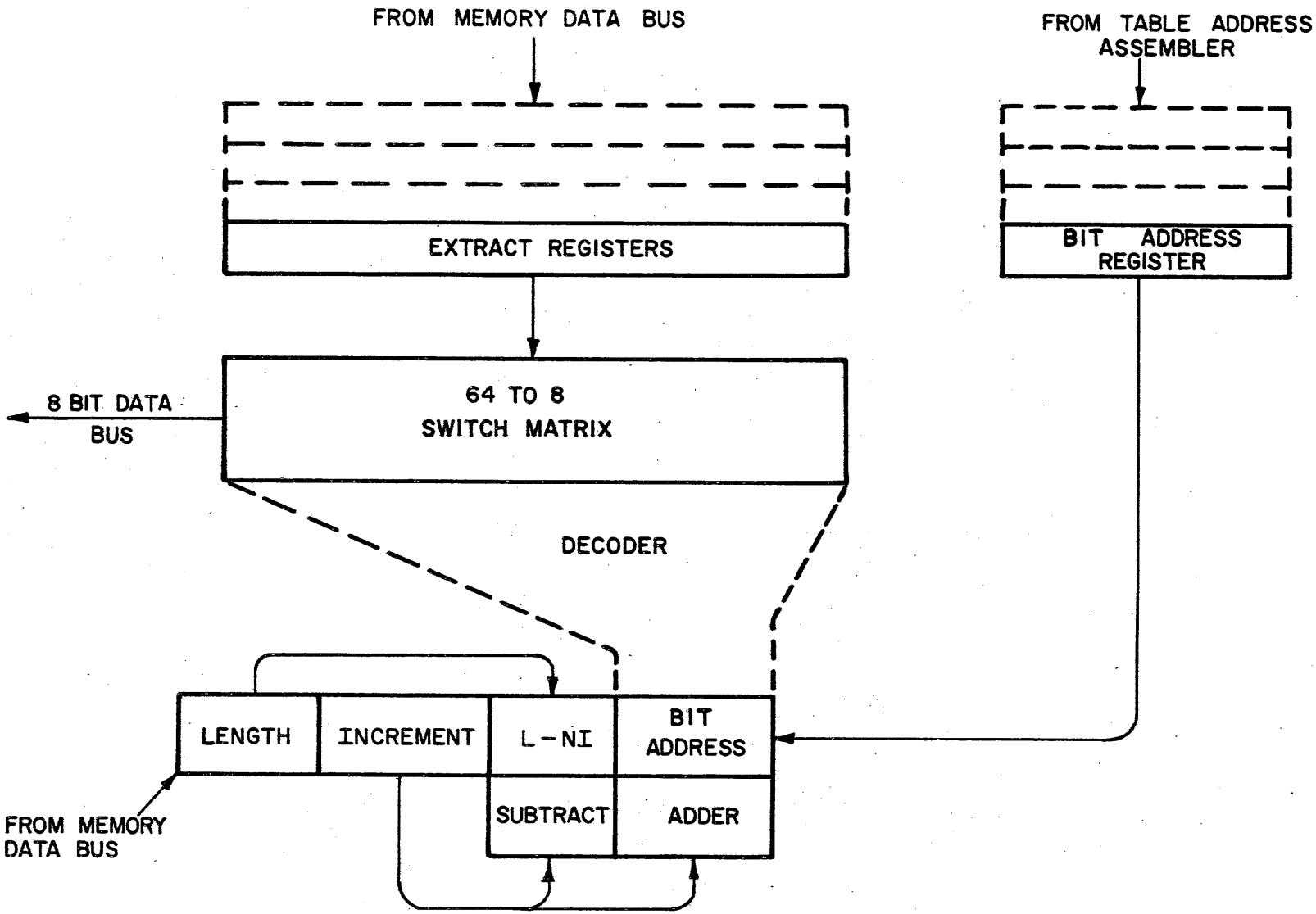
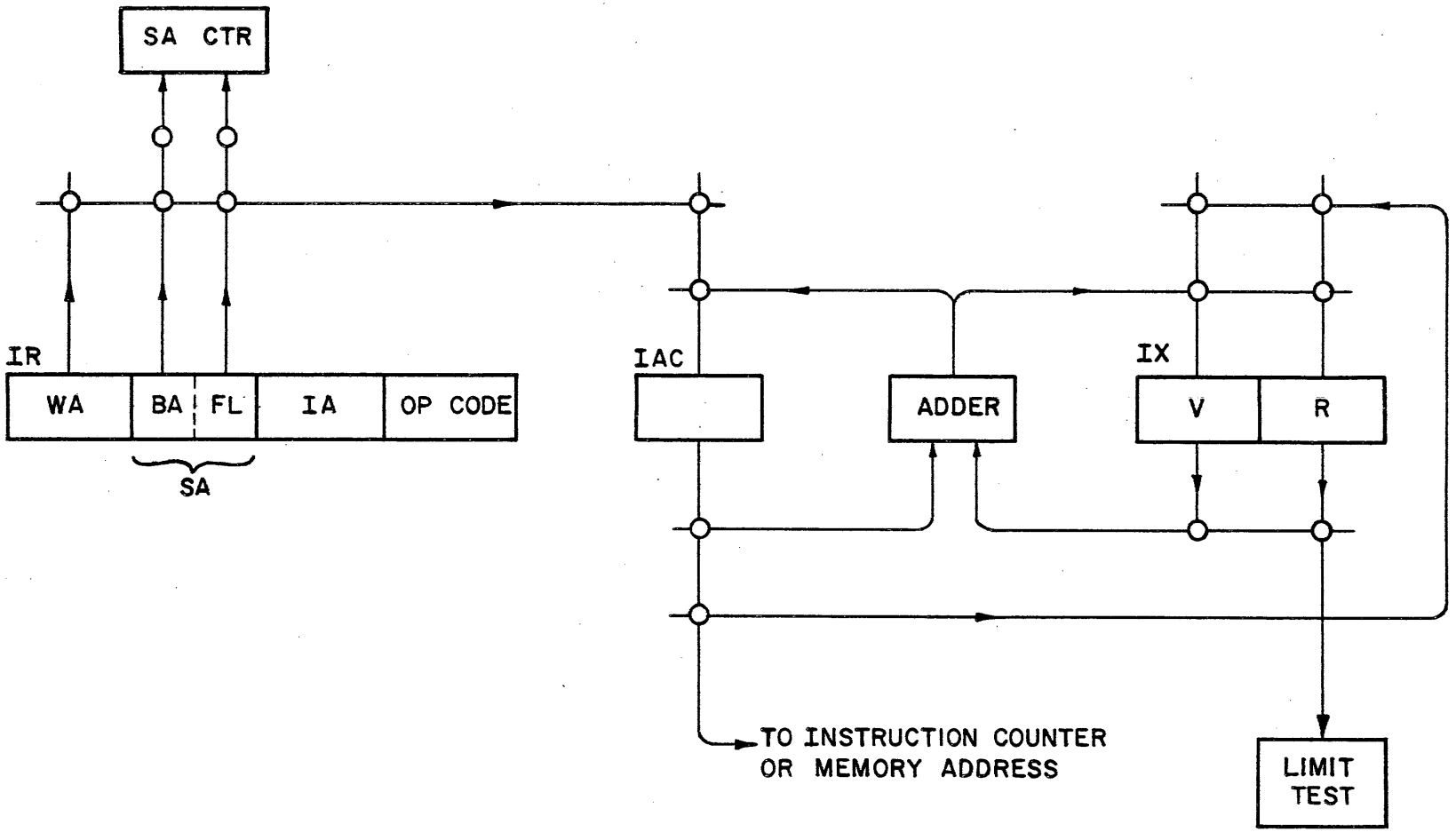


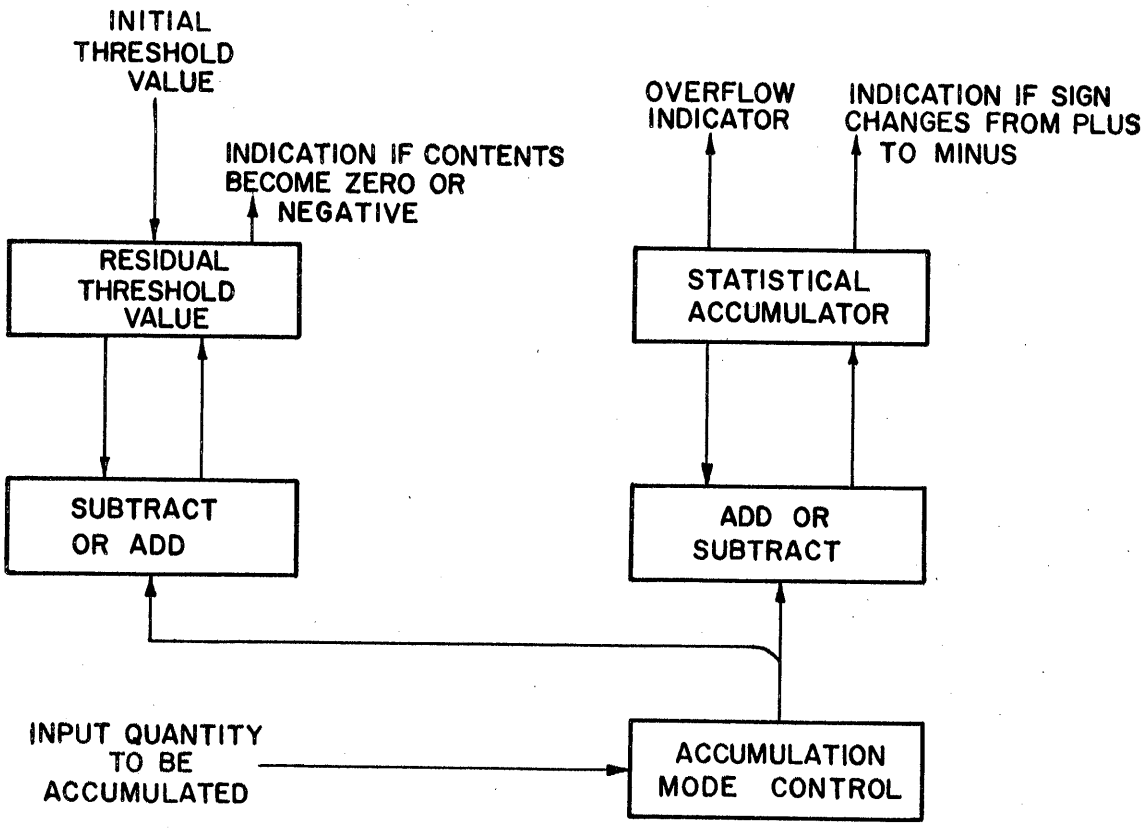
TABLE EXTRACT UNIT

FIGURE 4.5



GENERAL INDEXING

FIGURE 4.6



STATISTICAL ACCUMULATOR

FIGURE 4.8

5. System Operation in the Arithmetic Mode

5.1 General Description of Arithmetic Mode

In the arithmetic mode of operation, instructions deal with data in memory, a field at a time. A field may be of any length from one to 64 bits; it may start at any bit position in any word in memory and continue through that word and into the next higher word. Thus, fields of any length from one to 64 bits may be stored in adjacent memory positions, regardless of memory word boundaries, and instructions in the arithmetic mode can directly address such fields.

Variable field length (VFL) instructions are of the single-address type. The address is assumed to be in one of the internal computer registers. Among the VFL instructions are the arithmetic instructions: LOAD, ADD, STORE, MULTIPLY, DIVIDE, etc. To these are added instructions for rounding and for setting new fields lengths, which are used to align the radix points and cast out undesired digit positions.

There are non-arithmetic instructions included under the "arithmetic mode" because they resemble the arithmetic instructions in that their operation extends over variable fields not exceeding 64 bits. The CONNECT instruction, also VFL, provides the 16 "logical connectives" of two binary variables. Other non-arithmetic instructions deal only with full 64-bit memory words; the format of these instructions permits a limited second memory address without indexing. Included among them are instructions for transmitting full words between two locations, modifying index quantities, and branching.

Space has been left in the format for defining floating-point arithmetic operations, should a floating-point format prove desirable in this computer. Floating-point words would have a fixed length of 64 bits. Some references to floating-point features will be found in this manual to indicate how such features would be incorporated, but they are not defined in detail.

5.1.1 Data Registers and Paths

Three data registers, A, B, and C, are involved in arithmetic operations (Fig. 5.1). These are the same data registers, with their controls, which are used in the streaming mode of operation.

Each register is 128 bits long including sign positions. The registers communicate with memory in parallel, 64 bits at a time. The registers also communicate via the Logical Unit (LU), one byte of 8 or less bits at a time. Selection and alteration of bits out of a 64-bit memory word occur only over the paths between the registers, never on the way to or from memory.

Conversion between parallel and serial representations takes place in shift matrixes which, for simplicity, are not shown in Fig. 5.1. The registers themselves do not have shifting properties.

All three registers can transmit bytes to the LU, but only register C can receive and assemble bytes coming from the LU. To permit further operations on results without going through memory, a direct parallel path is provided between registers C and A.

The three registers serve the following general functions during arithmetic:

Register B receives the operand from the specified memory address.

Register A holds the unspecified operand and receives the result when the result is not to be returned to memory immediately.

Register C acts as temporary storage during arithmetic operations and for transfer of results to memory.

An ADD instruction may serve to illustrate these functions. One of the numbers is assumed to be already in A. The field length of this number is retained in the controls of A. The ADD instruction specifies a memory address and field length for the other number. The corresponding memory word (and possibly a second memory word) is loaded into B. The numbers are then stepped serially out of A and B into the LU, from right to left, starting at the sign byte. At the same

time the sum is stepped serially into C, starting at the right end of C. When the serial addition has been completed, the contents of C are transferred in parallel back into A, ready for another operation.

Exceptions to the above general functions are explained under specific instruction headings.

5.1.2 Binary and Decimal Arithmetic

A bit in the arithmetic instructions specifies either binary or decimal arithmetic. The adder in the LU automatically adjusts its mode of operation accordingly. The LU can accommodate up to 8 bits in parallel.

Numbers can be signed or unsigned as specified by one of the instruction modifier bits. For arithmetic purposes, the numeric part of a field in memory cannot exceed 60 bits in length without requiring more than one instruction for each arithmetic operation. A 4-bit sign byte (see below) can extend the word length to 64 bits for signed numbers, but unsigned numbers are restricted to 60 bits altogether. Within Register A, a double-length result can be developed.

Binary:

The numeric portion of binary fields can be of any length from 1 to 60 bits. The instructions specify a byte size which refers to the sign byte only; as explained below, the sign byte can consist of merely a sign bit or of several additional indicator bits depending on the byte size specified. This sign byte is on the right end of the number. Byte size has no meaning in unsigned binary numbers, which are always treated as positive.

Since the numeric portion of binary numbers is homogeneous, the machine automatically adds them in 8-bit bytes for maximum speed. Only at the ends, as defined by the starting address and field length, does the machine adjust its oper-

ation, when necessary, to take less than 8 bits.

VFL binary arithmetic can be used to modify signed or unsigned portions of index, instruction, and floating point words in unusual ways not covered by regular instructions.

Decimal:

Decimal digits are normally represented by the ten 4-bit binary integers, 0000 through 1001. Thus a byte size of 4 is specified for purely decimal data. For uniformity, the sign byte of a signed decimal field is assumed to be the same size as the decimal digits. Byte sizes of less than 4 bits are meaningless.

Byte sizes greater than 4 can be specified for decimal numbers. The extra bits are inserted to the left of the basic 4-bit decimal digit. For byte sizes of 5 or 6 bits, the extra bits are 1's; for 7 or 8 bit byte sizes, one or two zeros are inserted in addition. Thus, the digit "4" with a byte size of 6 appears as: 110100. With a byte size of 8, the coding is: 00110100. The sign byte is treated the same way. Arithmetic operations are automatically adjusted for byte size by the removal or insertion of the extra 1's.

Byte sizes of 6 to 8 bits permit decimal digits to be interspersed with alphabetic characters and still retain a uniform byte size. The use of two 1's on the left of the enlarged byte automatically places decimal digits into the generally accepted character sequence during a binary comparison.

The field length of decimal numbers may be any multiple of the byte size chosen, but it is restricted to multiples of the byte size and to a maximum of 60 or 64 bits, depending on whether the field is signed or not.

When the byte size for decimal numbers is 4, the machine adds two decimal digits (8 bits) of each number in parallel, whenever possible, to attain greater speed. This adjustment is again automatic.

5.1.3 Sign Byte

The sign byte can be from 1 to 8 bits long, as explained

in Section 5.1.2. Byte sizes 1, 2, and 3 are valid only in the binary mode. The eight possible bytes are:

Byte Size	Sign Byte
1	S
2	S Z
3	S Z T_2
4	S Z T_2 T_1
5	1 S Z T_2 T_1
6	1 1 S Z T_2 T_1
7	0 1 1 S Z T_2 T_1
0*	0 0 1 1 S Z T_2 T_1

where S = sign bit (0 for plus, 1 for minus)
Z = zero value bit (0 for zero, 1 for not zero)

T_1 and T_2 = Data Tag bits (if 1, each tag bit turns on a corresponding indicator bit for program interrupt)

* A code of 0 denotes a byte size of 8.

For unsigned numbers, the sign byte is missing in memory; but the sign is assumed to be plus, and the sign byte in the registers is set accordingly. During arithmetic operations the sign positions in the registers are handled as if all numbers were signed. Thus, an operand can be brought in without sign and the result stored with sign, or vice versa.

The field length includes all the bits of a number, including the sign byte if signed.

The Data Tag bits are retained during arithmetic operations. The corresponding tag bits of the two operands are Or-ed to form the tag bits of the result. Tag bits are stored in the sign byte, if any, along with the result.

5.1.4 VFL Numeric Formats

Fig. 5.2 gives examples of various numeric formats using arbitrary field lengths.

5.2 Operand Designation

The instruction format for the arithmetic mode is shown in Fig. 5.3. The left-most seven fields, containing 48 bits in all, de signate the operand or operands. The remaining 16 bits, starting at the operation code and extending to the programmer's Instruction Tag bit on the extreme right, specify the operation to be performed. The functions of these 16 bits will be discussed in later sections.

All address quantities are binary.

5.2.1 Addressing Convention

All variable field length instructions carry 26-bit addresses which allow them to address any bit in a working memory of up to 2^{20} (over 1 million) words of 2^{6} (64) bits each. Addresses follow a "left to right" sequence. Reading and writing operations deal with consecutive words addressed in ascending sequence. If writing starts in word address 2^{16}, the first word goes to word address 2^{16}, the next word to $2^{16} + 1$, the next to $2^{16} + 2$, and so on. Bit addressing follows the same pattern. The left-most bit of a word carries bit address 000000, the next bit 000001, and on to the last bit address 111111 of the word.

Consistent with this addressing scheme, fields and bytes within a field are addressed by the address of the left-most bit. Most streaming operations start at the byte whose left-most bit is located at the specified starting address, and step along, byte by byte, in ascending order of addresses. (The Exchange also operates in this manner when transmitting information to and from units a byte at a time, except that it is constrained to start at the left end of a full memory word.) A 20-bit word address and the corresponding 26-bit field address are related by simply adding 6 zeros to the right end of the word address.

Although arithmetic operations must proceed from right to left, in a descending order of addresses, the same addressing convention is still used. Numeric fields are addressed by the position of the leftmost, or high-order, bit. The correct starting position at the right end (low-

order) is calculated automatically.

5.2.2 Operand Address

The leftmost 20 bits of the instruction specify the word address of an operand in memory. The word address, after modification by indexing, is sent to memory to select a full 64-bit memory word. Special addresses permit the word address to select control words currently in the Exchange, and internal computer registers.

For variable field length operations, the 6-bit address part of the instruction extends the word address to a complete 26-bit operand address. During the execution of a VFL instruction, the bit address is used inside the computer to select from a full word the leftmost bit of the field. On storing, only full words are sent to memory to be stored at the word address; bits are assembled in the computer beforehand. Thus, the word address and the bit address perform different kinds of functions. Logically, however, the VFL operand address may be considered to be a single 26-bit number.

Non-arithmetic instructions, which do not operate on VFL data, have only the 20-bit word address to select the operand.

Whenever the effective operand address, after indexing, is zero, any reference to memory is automatically suppressed. Address zero is always considered to contain zero. When an operation calls for an operand from address zero, a zero operand is automatically provided in the correct format (VFL or floating point, unsigned or signed with appropriate sign byte). When an operation indicates storing an operand at address zero, nothing actually is stored.

5.2.3 Field Length

In VFL instructions, the 6 bit field length specifies the total number of bits in the operand, from 1 to 64. A field length of 64 bits is encoded as 000000.

For signed numbers, the field length includes the sign and tag bits. Thus signed and unsigned numbers with the same numeric bits differ in field length by the bits in the sign byte.

The field length is omitted from instructions dealing only with full words from memory.

5.2.4 Second Address

For other than VFL instructions, the bit address and field length parts of the instruction are combined and interpreted as a second 12-bit address. The use of this address depends on the instruction. Most frequently it is used as an address to designate internal registers, or a section of memory limited to word addresses below 4096 (2^{12}), or an external unit. Another limitation of the second address is that is cannot be indexed.

When used as a memory address, the second address is interpreted as a 20-bit address of which the high-order 8 bits are zeros.

5.2.5 Address Modification

The operand address can be modified before using it to select the operand by adding an address modifier (index value). The addition results in an effective address which is the one actually used in word and bit selection; the contents of the instruction as stored in memory, however, are left unchanged.

The index address part of the instruction contains 12 bits which specify the location in memory of an index word containing the modifier. The index word locations are thus limited to word addresses below 4096 (2^{12}).

The index word format is shown in Fig. 5.4. The value field on the left is the part that is added to the address part of the instruction for address modification, the remainder of the index word being ignored. The index value field consists of 27 bits: a word address, a bit address, and a sign. The addition is algebraic.

The address part of the instruction does not contain a sign bit and is always considered positive. The resultant effective address must be a positive quantity. Any invalid address turns on the Invalid Address indicator.

The same index word format is used to modify 26 or 20 bit operand addresses. When the operation is not of the VFL type, the 6 bits of the bit address are suppressed during the addition, but the sign is still effective.

The addition takes place in an index adder which is separate from the main arithmetic adder in the SLU. The execution of one instruction may overlap with the obtaining and modifying of the next instruction.

Negative index values are represented in absolute value form with sign; any complements occurring during addition are automatically converted to the absolute value form.

When the index address is zero, address modification is omitted. This is the normal method of specifying "no indexing".

5.2.6 Address Modification for Word Transmission

A special form of address modification allows both the effective Word Address and the Second Address to advance automatically, so as to permit a group of consecutively stored words to be transferred from one location to another, with one instruction. This mode of operation is explained in the section on Word Transmission.

The stepping up of addresses takes place within the computer and does not alter the instruction and index words in memory which are used to initiate the action.

5.2.7 Address Designators

The instruction format (Fig. 5.3) contains three address designators which modify the interpretation of the index address, the word address, and the second address, respectively. The previous descriptions of

these addresses apply when the designators are all set to zero. They are referred to as "direct" addresses. Other meanings are listed in the following table.

a. Index Address Designator (IAD - 1 bit)

IAD = 0 : Direct
IAD = 1 : Indirect

b. Operand Address Designator (OAD - 2 bits)

OAD = 00 : Direct
OAD = 01 : Indirect
OAD = 10 : Immediate

c. Second Address Designator (SAD - 1 bit)

SAD = 0 : Direct
SAD = 1 : Indirect

A direct index address specifies an index word in memory which is used directly to modify the operand address.

An indirect index address specifies a Multiple Index Address Word in memory (see Fig. 5.5). This word has a format similar to an instruction word and contains space for three index addresses in the fields corresponding to the WA, SA, and IA parts of the instruction. Each address can be used to specify a separate index word whose value is to be added to the address part of the basic instruction to form the final effective address.

The multiple index address word itself contains an IAD. If the new IAD = 1, the new IA part is used to fetch yet another multiple index address word. The process of multiple indexing continues until it encounters an IAD = 0. Then the last IA is interpreted as a direct index address, and this becomes the last step in the indexing process.

Multiple indexing proceeds in the order WA, SA, and IA (if direct). If les s than three index addresses are needed, the remaining address fields should be set to zero. A zero address in any of the three fields also terminates the multiple indexing process. Everything beyond the zero index address is then ignored.

Multiple indexing thus permits any number of indexing values to be added to the operand address. Each multiple index address word in the chain contributes two more index addresses except the last word which can contribute three. Another feature of multiple indexing is that there is one 20-bit address in each word which permits an indexing quantity anywhere in memory to be specified.

A direct operand address results in an effective address after modification which directly selects an operand.

An indirect operand address, after modification by indexing, causes an Indirect Address Word (Fig. 5.5) to be brought out of memory. This word has a format and contains another operand address, field length, index address, IAD and OAD. The new address so obtained may itself be indexed by means of the index address brought in with the new address word. The new OAD can also indicate "indirect", thus causing the new effective address to obtain yet another indirect address word. The process continues until the OAD indicates "direct" or "immediate (see below). The effective address obtained from the final address word then selects the actual operand.

An immediate operand address is used to permit specifying short operands immediately in the address part of the instruction, without reference to another memory location. The address may be modified by indexing, and the effective address is then sent out over the data bus to become the operand itself.

The bits appear on the data bus and in the receiving register in the left-most positions of the word. This exactly corresponds to the original position of the immediate address in the instruction. The maximum size of the operand is 27 bits of which the right-most bit is a sign. The instruction actually can supply only 20 meaningful bits in the high-order positions, but all the 27 bits of the effective address, including any sign supplied by the index value, are sent out as data. The rules for modification of immediate addresses are the same as for regular addresses.

With VFL operations, the bit address (after indexing) and the field length are used by Register B to specify the starting position and the number of bits of the immediate address to be used in the operation. Thus, the bit address appears both on the data bus and in the controls of Register B, and care must be taken with immediate addresses to produce meaningful results. Normally, this technique should be restricted to unsigned data of 20 bits or less.

Immediate operand addresses are not defined for branching or for operations which normally return data to memory (such as STORE).

A direct second address is placed into a Second Address Counter to be used as it stands.

An indirect second address refers to a memory location containing another instruction word. The second address part of the word is then substituted in the Second Address Counter. Both direct and indirect second addresses are limited to 12 bits. Indirect second addressing continues until SAD = 0.

When more than one non-zero address designator appears in an instruction, the sequence is as follows:

1. If SAD = 1, the second address substitutions are made first.

2. Multiple indexing, if IAD = 1, is taken care of next, to modifying the present operand address.

3. Then, if the operand address is indirect (OAD = 01), the indirect address word is obtained to replace in the Instruction register the left-most 4 7 bits, up to the OAD, of the original instruction. The right-most 17 bits, from the SAD on, are left unchanged.

4. The new contents of the instruction register are again decoded as if they were a new instruction, except for the second addres s. Steps 2 and 3 are repeated until the OAD indicates direct (00) or immediate (10).

Thus, each new operand address can be subjected to single or multiple indexing, using the indexing information carried with the indirect address word. This implies that the index address information to modify the final operand address must be stored with that address in the Indirect Address Word and cannot be obtained from the original instruction.

Indirect addressing and multiple indexing provide great flexibility, but possibly at the expense of speed. The loss in speed, if any, depends on the number of steps in the instruction modification process and on the duration of the previous instruction execution with which the instruction modification overlaps.

5.3 Index Modification

The instructions needed to advance and test index words stored in memory for subsequent use in modifying addresses of the instructions are referred to as index modification instructions. Most of them are two-address instructions, where the operand address specifies an index word containing the increment or replacement value, and the second address specifies the index word to be modified.

The operand address of an index modification instruction may itself be indexed to permit the use of a series of modifiers.

The words involved in index modification follow the index word format of Fig. 5.4. Each word consists of two 27-bit fields, named Value and Length. The remaining bits are not used during general indexing, but have significance when an index word is used as a control word in the Exchange.

5.3.1 Value and Length

The Value field is the field that is added to an operand address in the normal course of events. When the Value field is to be changed by regular increments, the Length field defines the end of the incrementing process.

Initially, the Value field is set to the Starting Value, usually zero. The Length field is set to the total

number of bits to be traversed from start to finish. During incrementing the increment is added to the Value and subtracted from the Length. When the Residual Length reaches zero, the end of the process is indicated.

Thus, both fields change continually during the process. The Residual Length always indicates the number of bits yet to be covered. Instructions for restoring both fields to their initial settings are provided.

Index words can also be used for simple counting without address modification. A special instruction is provided to advance the count and branch in one operation.

5.3.2 INCREMENT

The INCREMENT instruction specifies two index words, located in memory at the second address and the word address, each word containing two fields, V and L.

	V	L
Second Address:	Current Value	Residual Length
Word Address:	Increment	(Length)

The increment is added to the current value and subtracted from the residual length. All quantities are signed, and addition and subtraction are algebraic.

The Residual Length, after the subtraction, is tested for zero and sign, and the test is retained in the Indicator Register for subsequent use by a CONDITIONAL BRANCH instruction.

The word address may be designated as immediate, so that the operand address itself can serve as a 20-bit increment with a zero bit address and positive sign. If the immediate address is also indexed, a non-zero bit address and either sign may be provided. This technique often avoids the need for the second index word to specify the increment.

When an index word is used to hold the increment, the L field of that word does not affect the INCREMENT operation. It may serve to retain the initial setting of the Residual Length when the simple RESTORE oper-

ation (see below) does not apply.

5.3.3 Branching and Counting

The INCREMENT instruction includes a test to determine whether an indexing process has reached its limit. The result of the most recent test is retained in the Indicator register. A Conditional Branch instruction interrogates the Indicator register and alters the program depending on the bit.

To simplify counting, a Count operation is provided and explained in detail in the section on Branching. It is equivalent to an INCREMENT instruction followed by a Conditional Branch. The increment is implied to be + 2^6, i.e., a one is added in the low-order position of the word address. The maximum range of this count is 0 to 2^{20} - 1.

5.3.4 REPLACE V BY V
REPLACE L BY L
REPLACE L BY V
REPLACE V BY L

The REPLACE INSTRUCTIONS CAUSE EITHER THE V or L field of the index word at the second address to be replaced by either the V or L field of the index word at the word address. The word at the word address remains unchanged. Except for the specified field, the remainder of the word at the second address also remains unchanged.

Instruction	REPLACE Field at Second Address	BY Field at Word Address
REPLACE V BY V	V	V
REPLACE L BY L	L	L
REPLACE L BY V	L	V
REPLACE V BY L	V	L

If the entire index word is to be replaced, a TRANSMIT or RECEIVE instruction is used.

5.3.5 RESTORE

RESTORE performs the following operations on the index word at the word address:

a) V is added algebraically to L, the sum replacing L.

b) V is then set to zero.

Whenever an index word starts with V ≠ 0 and is modified by use of the INCREMENT or Count instructions, the index word may be restored to its initial setting by the RESTORE instruction. V is added to L, rather than simply replacing it, to permit returning to the original setting even if L has not yet reached zero or has gone beyond zero.

5.4 Arithmetic Operations

Only variable field length fixed-point arithmetic operations are defined here.

5.4.1 Arithmetic Instruction Format

Fig. 5.3 shows the positionsof the various parts of the instruction. The parts designating the operand are described in Section 5.2. The parts defining the operation are described below.

The operation code (6 bits) designates the basic operations. These operations are modified by the modifier bits listed below. The operation code also distinguishes between arithmetic, non-arithmetic and streaming operations. The modifier bits do not have the same meaning for non-arithmetic operations where they form merely an extension of the operation code.

Sign Modifier (SM - 2 bits: 54-55)

The operand sign is modified before starting the operation without changing the sign as stored in the original location. For ADD TO MEMORY, only, the sign modifier is applied to the number coming from register A.

SM = 00: Leave operand sign unchanged.
SM = 01: Invert operand sign.
SM = 10: Set operand sign to plus
SM = 11: Set operand sign to minus.

Format Modifier (FM - 2 bits: 56-57)

FM = 00: Unsigned variable field length
FM = 01: Signed variable field length
FM = 10: Unnormalized floating point
FM = 11: Normalized floating point

For incoming unsigned operands a plus sign is assumed, and the sign modifier bits then modify this plus sign. The result in the data register is always signed, but it may be stored, signed or unsigned. Thus with FM = 00, there is no distinction between SM = 00 and 10, or between SM = 01 and 11.

Decimal Modifier (DM - 1 bit: 58)

If DM = 0, the arithmetic operation is binary.
If DM = 1, the arithmetic operation is decimal.

Left Half Modifier (LM - 1 bit: 59)

If LM = 0, all 128 bits of Register A are used in the operation, starting at the right end.
If LM = 1, only the left 64 bits of Register A are used in the operation.

Byte Size (B.S. - 3 bits: 60-62)

Byte Size refers to the specified operand.
If B.S. = 000, the byte size is 8 bits.
If B.S. = 001 to 111, the byte size is 1 to 7 bits, respectively.
For binary numbers (DM = 0), the byte size indication applies only to the sign byte.
For decimal numbers (DM = 1), the byte size indication applies to the decimal digits as well as to the sign byte.

Instruction Tag (IT - 1 bit: 63)

Every instruction carries this bit. It is sent to the Indicator register to permit a program interruption if desired (see section on Interrupt). It does not affect the operation of the current instruction in any way.

5.4.2 Variable Field Length Arithmetic

All arithmetic operations, other than floating point, operate on a variable field length basis. These operations proceed as if the radix point were fixed at the right end of the number. Adjustment of the right end of the field may be needed to line up the point.

Single operands are limited to a maximum of 60 bits in length, not counting sign bits. Intermediate results in Register A can extend to 120 bits; results longer than 60 bits are stored in two parts, or they are first adjusted to reduce them to 60 bits or less.

Each operation produces a result of a specific length unless an overflow occurs. Overflow automatically increases the result field length to retain all significant bits but the Result Overflow Indicator is turned on. If the result field attempts to exceed 120 bits the Register Overflow Indicator is turned on. The field length of results in Register A is retained in the index controls from one operation to the next. Thus, although most instructions do not directly specify the field length in A, complete control is retained over the field lengths.

Keeping the field lengths as short as possible can save memory space and operating time.

5.4.3 LOAD

(SM = 00)

The specified operand is loaded into Register A, replacing any previous contents of A. The operand is left unchanged in the original location.

The operation places the one or two 64-bit memory words containing the operand into A without shifting or relocating the data. The starting bit address, the field length and byte size are retained in the register controls to define the field for subsequent operations.

If the field is signed, the sign byte is inspected and the proper indicators in the Indicator Register are set.

LOAD does not affect Registers B or C.

Sign Modifiers

SM = 01: Load with opposite sign.
SM = 10: Load with plus sign.
SM = 11: Load with minus sign.

5.4.4 ADD

(SM = 00)

The specified operand is added algebraically to the contents of Register A.

The sum is first produced in Register C and then returned to Register A, replacing the previous contents. The sum lines up with the right end of A and the rest of the bits are zero. The operand is left unchanged at its original location.

The field length of the result is defined as the length of the longer of the two numbers added. If an overflow occurs beyond the result field length as defined, the field length is increased by one bit (if binary) or byte (if decimal), and the Result Overflow Indicator is turned on.

When two numbers are added, the byte size of the result is the byte size of the number in A. This holds for either binary or decimal addition. When storing the result it may be converted to another byte size.

The Sign and Zero Indicators are set to correspond to the sum.

ADD alters Registers B and C.

Sign Modifiers

SM = 01: Subtract
SM = 10: Add absolute
SM = 11: Subtract absolute

5.4.5 STORE

(SM = 00)

The contents of Register A are stored at the specified operand address using the specified byte size. Only as many bits as specified in the operand field length are stored. Bits to the right or left of the specified field in memory remain unchanged. Register A remains unchanged.

If the specified field length is greater than the field length of the number in A, the remaining bit positions in memory are filled with zeros. If the specified field length is less than that in A, nothing is stored beyond the field length specified; the remainder of A is scanned and if any more one-bits are found the Store Overflow Indicator is turned on.

STORE alters Register C but not B.

Sign Modifiers

SM = 01: Store with opposite sign.
SM = 10: Store with plus sign.
SM = 11: Store with minus sign.

5.4.6 STORE C

The contents of Register C are stored at the specified operand address using the specified byte size. The rules for field length, overflow, and treatment of bits in memory are the same as for STORE.

STORE C is used primarily to store the remainder after DIVIDE.

STORE C alters the contents of B and C. Register A remains unchanged.

5.4.7 COMPARE

COMPARE is the same as ADD in all respects except that the sum is not returned to Register A which remains unchanged. The sum does appear in Register C and can be stored by using the STORE C instruction before another instruction intervenes which would alter C. The sign modifiers apply. The Sign and Zero Indicators are set to

correspond to the sum.

The chief use of COMPARE (with SM = 01) is to obtain a non-destructive algebraic comparison of a quantity (A) in Register A with the specified quantity (B) in memory. The fields can be of different lengths. The result is indicated by the Sign and Zero Indicators.

For example, with SM = 01:

Indicator On	Meaning
0	(A) = (B)
+ (non-zero)	(A) > (B)
-	(A) < (B)

For alphanumeric fields of 60 bits or less use COMPARE with SM = 01, FM = 00, and DM = 0.

COMPARE alters Register B.

5.4.8 DIMINISH

DIMINISH is the same as ADD in all respects except that, if the sum is negative, it is replaced by zero in Register A. The true sum appears in Register C and can be stored by using the STORE C instruction before another instruction intervenes which would alter C. The sign modifiers apply. The Sign and Zero Indicators are set to correspond to the true sum in C.

The chief use of DIMINISH is with an unsigned format and SM = 01.

DIMINISH alters Register B.

5.4.9 ADD TO MEMORY

(SM = 00)

The contents of Register A are added to the specified operand, the sum being returned to the specified operand address. Bits to the right or left of the specified Memory area remain unchanged. Register A remains unchanged.

The specified format modifier and byte size applies to the operand and the result.

The result has the same field length as the specified operand. If the sum is longer than the specified field length, nothing is stored beyond the field length specified; the remainder of the sum is scanned and, if any non-zero bits or bytes are found in the sum which were not stored, the Store Overflow Indicator is turned on.

The Sign and Zero Indicators are set to correspond to the sum as it is stored. The Data Tag on the operand originally in Memory controls the Data Tag Indicators. The Or of the operand Data Tags is the Data Tag of the result.

ADD TO MEMORY alters Register C but not B.

Sign Modifiers

The sign modifier alters the sign of the number coming from A.

S = 01:	Subtract from Memory.
S = 10:	Add absolute value of A to Memory.
S = 11:	Subtract absolute value of A from Memory.

5.4.10 MULTIPLY

The contents of Register A (the multiplicand) are multiplied by the specified operand (the multiplier). The product is formed in Register C after first clearing C. The product is returned to Register A, replacing its previous contents.

The length of the product is the length of the multiplicand plus the length of the multiplier. The byte size of the product is the byte size of the multiplicand.

Since the factor in Register A can have a length of more than 60 bits, it is possible for the product to exceed the register limit of 120 bits. If this occurs, the multiplication is stopped and the Register Overflow Indicator is turned on. In this case the multiplicand in A remains unchanged.

The Sign and Zero Indicators are set to correspond to the product. The Data Tag Indicators are set to correspond to the multiplier Data Tags.

5.4.11 LOAD AC

The contents of Register A are first transferred to Register C. Register A is then loaded with the specified operand, as described under LOAD.

LOAD AC is used to load the multiplicand prior to a CUMULATIVE MULTIPLY operation, preserving the previous contents of A for addition to the new product. No other operation should intervene which would alter the contents of Register C.

The transfer of A to C is made serially via the LU to line up the number at the right end of Register C, ready to be added to the new product.

Register B is not changed.

5.4.12 CUMULATIVE MULTIPLY

The contents of Register A (the multiplicand) are multiplied by the specified operand (the multiplier). The product is added algebraically to the previous contents of Register C. The sum is finally returned to Register A, replacing its previous contents.

The length of the result is the length of the field previously in C, or the sum of the multiplier and multiplicand lengths, whichever is longer. If an overflow occurs beyond the result field length as defined, the field length is increased by one bit (if binary) or byte (if decimal), and the Result Overflow Indicator is turned on.

The multiplicand should have been loaded by means of a LOAD AC instruction prior to the CUMULATIVE MULTIPLY instruction. No instruction should intervene which would alter the contents of Register C.

The Sign and Zero Indicators are set to correspond to the result. The Data Tag Indicators are controlled by the multiplier Data Tags.

Register B is altered.

5.4.13 DIVIDE

The contents of Register A (the dividend) are divided by the specified operand (the divisor). The quotient is returned to A, replacing the previous contents. The remainder appears in Register C.

If the remainder is to be retained, it can be stored by using STORE C. This instruction should follow DIVIDE before another instruction intervenes which would alter the contents of C.

Not counting any sign bytes, the number of bits (if binary) or digits (if decimal) in the quotient is given by the number of significant bits or digits in the dividend, minus the number of significant bits or digits in the divisor, plus one bit or digit. The number of significant divisor bits or digits excludes any zeros to the left of the most significant bit or digit in the divisor.

If the divisor is zero, the division is not executed and the Zero Divisor Indicator is turned on. No quotient is produced, and the dividend remains in A.

If the significant bits or digits in the divisor exceed the number of bits or digits in the dividend, the result is a zero quotient and a remainder equal to the dividend.

These rules treat the divisor as if it were an integer and produce meaningful quotients and remainders for any non-zero divisor.

The length of the remainder is the same as the length of the dividend.

The byte size of the quotient and remainder is the byte size of the dividend.

The division process consists of first transferring the contents of A to C, via the LU, to line up the dividend in C. The divisor is placed in B. During the division, the quotient is developed in the left end of C while the dividend is reduced toward the right end. This leaves both quotient and remainder

in C. The contents of C are then dumped into A, and the controls of A are set to the bit address and length of the quotient, while the controls of C are set to the bit address and length of the remainder.

The Sign and Zero Indicators are set to correspond to the quotient. The Data Tag Indicators are set to correspond to the divisor Data Tags.

Register B is altered.

5.4.14 SHORTEN

The field length of Register A is decreased by removing a number of bits from the right end of the numeric portion of the field in A. The sign byte is left in place. The number of bits to be removed is specified in the Field Length part of the SHORTEN instruction.

The whole number is passed through the LU to align it at the right end of Register A and to test the shortened number for zero. The Sign and Zero Indicators and the sign byte are set to correspond to the result.

The Operand and Index Address parts of the instruction are not used.

SHORTEN alters Register C but not B.

In the decimal mode the number of bits removed should be a multiple of the byte size of A.

5.4.15 ROUND

The field length of Register A is decreased by removing a number of bits from the right end of the numeric portion of the field in A and rounding the remaining portion.

ROUND is the same as SHORTEN except that a 1 is added to the absolute value of the remaining field, in the low-order position, if

(a) in binary arithmetic, the highest bit dropped is a 1, or

(b) in decimal arithmetic, the highest byte dropped is

a 5 or greater.

Any carry beyond the left end of the field increases the resultant field length by one bit or byte and turns on the Result Overflow Indicator.

ROUND alters C but not B.

5.4.16 LENGTHEN

The field length of Register A is increased by inserting a number of bits to the right of the low-order position of the numeric portion of the field in A. The sign byte is left in place. The number of bits to be inserted is specified in the Field Length part of the LENGTHEN instruction.

The inserted bits are zero bits in binary arithmetic and properly coded zero bytes in decimal arithmetic.

The whole number is passed through the LU to align it at the right end of Register A.

The Operand and Index Address parts of the instruction are not used.

LENGTHEN alters Register C but not B.

5.5 Binary Connectives

The LU can provide the 16 connectives of two binary variables, which are listed in Fig. 5.6. A and B represent a single bit from each of two operands. The four columns of one-bit results correspond to the four possible states of A and B. The rows represent the 16 possible connective functions, coded 00 to 15.

Among the common connectives are:

14	A or B
08	A and B
06	A EXCLUSIVE OR B
09	A MATCH B
03	NOT A
05	NOT B

The trivial connectives A, B, 0, and 1 are included for completeness.

Because of the VFL feature, it is possible to operate either on pairs of single bits or to perform the same operation on all pairs of bits of longer fields. A series of single-bit operations can be used to evaluate complex logical expressions. The multiple-bit operations lend themselves to operations on complex patterns of bits.

5.5.1 CONNECT

The bits of the specified operand (B) are combined with the bits of the field (A) in Register A, according to the Table of Connectives (Fig. 5.6). The result is returned to A, replacing the previous contents.

The Connective Code is specified by the 4 bits of the Connective Modifier (CM) part of the instruction. (See Fig. 5.3).

The length of the result is the length of the specified operand. The original field length of A is ignored.

If the result contains only zero-bits, the Zero Indicator is turned on. If the result contains one or more one-bits, the Plus (Non-zero) Indicator is turned on.

The LM modifier applies to CONNECT. The SM, FM, and DM modifiers and the Byte Size do not apply.

CONNECT alters registers B and C.

5.6 Word Transmission

A pair of two-address instructions, RECEIVE and TRANSMIT, are provided to move full memory words from one location to another. Only one of the two instructions is used for any given operation, the difference between them being which of the two addresses is interpreted as source and which as destination. The two addresses are the 20-bit Word Address and the 12-bit Second Address.

If the Index Address is zero, no indexing takes place and only one word is transmitted.

To transmit more than one word, the Index Address is used to specify an Index Word. The Value part of the Index Word is added

to the Word Address to obtain the actual starting location. (The Second Address cannot be so modified). The Length part specifies the number of words to be transmitted. For successive words, both the effective Word Address and the Second Address are advanced by one. There are a number of restrictions to this process:

(a) Words must be stored at consecutively numbered addresses, both at the source and destination.

(b) The Second Address cannot be indexed.

(c) The Second Address can only specify addresses below 4096 (2^{12}). This does include all registers and high-speed memories, however.

(d) Fields other than full 64-bit memory words cannot be moved this way.

This pair of word transmission instructions is provided to permit either the source address or the destination address to be unrestricted. VFL instructions or the streaming mode should be used for more flexible manipulation of data.

5.6.1 RECEIVE
TRANSMIT

RECEIVE and TRANSMIT both move full 64-bit memory words from one set of locations to another. The only distinction is which address is the source and which the destination:

	Source	Destination
RECEIVE	Word Address	Second Address
TRANSMIT	Second Address	Word Address

The effective Word Address, after indexing, and the Second Address specify the two locations for the first word to be transmitted.

The Second Address and the effective Word Address are then stepped-up by 1 (in terms of words) and 2^6 is subtracted from the Length part of the specified Index Word.

The new Length is tested for zero or negative contents. Additional words are transmitted and the addresses and the Length are modified until the Length becomes zero (including the Bit Address part) or negative (if the Bit Address part was not zero). The process then terminates.

If the Index Address is zero, the Word Address is not modified and only one word is transmitted.

The contents of the source address are not changed.

The data registers are not affected by the instructions unless they are specified by one or both of the addresses. If one of the data registers is specified, its controls are not changed.

The operation modifiers do not apply.

5.7 Branching

Immediately after the Instruction Counter contents have been sent to Memory to obtain a new instruction, the Instruction Counter is stepped up by one to indicate the location of what is normally the next instruction. The branch instructions, however, permit the instruction sequence to be altered by inserting a new address in the Instruction Counter; the next instruction will then be obtained from this new location.

The unconditional branch instruction, BRANCH, always inserts a new instruction location. In addition, the Second Address is used to specify a word where the old Instruction Counter setting can be stored. Since at this time the Instruction Counter has already been stepped up, what is stored at the Second Address is the location of what would have been the next instruction without branching. If the Second Address is set to zero, the Instruction Counter is not stored and some time is saved.

By changing a single bit in the Operation Code from 1 to 0, the BRANCH instruction is converted to NO OPERATION.

A second unconditional branch instruction, BRANCH ENABLE, is described in the section on program interruption.

The conditional branch instructions, BRANCH IF I ON/OFF and BRANCH IF A ON/OFF, replace the Instruction Counter contents

only if a specified condition is satisfied; if the condition is not satisfied, the Instruction Counter continues the present instruction sequence.

The conditional branch instructions perform a wide variety of tests by permitting any one bit in the Indicator Register or in Register A to be interrogated. The Indicator Register, which is described in another section, contains all of the machine conditions needed for testing. Any bit stored in memory may be tested by first loading the corresponding memory word into Register A and then using the conditional branch instructions. The Bit Address specifies which bit position in the desired register is to be tested.

By changing a single bit in the Operation Code from 1 to 0, the BRANCH IF I (A) ON instruction is converted to BRANCH IF I (A) OFF.

Another pair of conditional branch instructions, COUNT AND BRANCH IF ZERO/NOT ZERO, combine advancing a count in an index word with conditional branching to test the count. Here the bit address is not used. Instead, the Second Address specifies the location of the Index Word containing the count.

All branch instructions may be indexed. The effective word address is sent to the Instruction Counter when branching occurs. In the conditional branch instructions, the bit address participates in the indexing as if these were VFL instructions; only the effective Word Address and the effective Bit Address have different functions. The field length is not used.

If the effective Word Address is zero, no branching occurs.

5.7.1 BRANCH

After advancing by one from the location of the present instruction, the Instruction Counter contents are stored in the leftmost 20 bits of the word specified by the Second Address; the remainder of the word is left unchanged.

Branching then occurs; that is, the Instruction Counter contents are replaced by the effective Word Address part of the instruction.

If the effective Word Address is zero, no branching occurs.

The Instruction Counter contents, which are not replaced, are still stored.

If the Second Address is zero, the Instruction Counter is not stored.

5.7.2 NO OPERATION

This instruction does nothing except set the Instruction Tag Indicator with its own Instruction Tag bit. All other parts of the instruction are ignored.

The Operation Code for NO OPERATION may be changed to BRANCH by changing a single bit from 0 to 1. Thus the operand designation of the NO OPERATION instruction is frequently that of a BRANCH instruction, but no branching occurs.

5.7.3 BRANCH IF I ON
BRANCH IF I OFF
BRANCH IF A ON
BRANCH IF A OFF

The Instruction Counter is set to the effective Word Address, i.e., branching occurs, if a specified condition is satisfied. If the condition is not satisfied, the Instruction Counter is advanced by 1, i.e., no branching occurs.

BRANCH IF I ON tests the bit in the Indicator Register specified by the effective Bit Address. If the bit is On (1), branching occurs. If the bit is Off (0), no branching occurs.

BRANCH IF I OFF also tests bits in the Indicator Register. If the specified bit is Off (0), branching occurs. If the bit is On (1), no branching occurs.

BRANCH IF A ON and BRANCH IF A OFF are similar to the first two instructions, except that they test bits in Register A.

If the effective Word Address is zero, no branching occurs in any case.

5.7.4 COUNT AND BRANCH IF ZERO
COUNT AND BRANCH IF NOT ZERO

The Second Address of these instructions specifies an

Index Word. 2^6 is algebraically added to the Value part of the Index Word and 2^6 is subtracted from the Length part. The low-order six bits are not altered. The modified Index Word is returned to its storage location.

The entire Length field is tested for zero to determine whether branching should occur.

COUNT AND BRANCH IF ZERO causes branching if the resultant Length field is zero.

COUNT AND BRANCH IF NOT ZERO causes branching if the resultant Length field is not zero.

If the effective Word Address is zero, no branching occurs in any case, but the Index Word is still modified.

5.8 Indicator Register

The Indicator Register contains 64 bits. The individual bits are turned On (set to 1) or turned off (set to 0) at the time certain specified conditions occur in the computer system. Each bit is also turned Off when it is interrogated, either by a conditional branch instruction or by an automatic program interruption.

The entire Indicator Register contents may be sent out over the data bus by instructions giving the address of the Indicator Register. This may be for the purpose of storing the Register contents in Memory or for manipulation in the data registers. Interrogating the entire Indicator Register in this way resets the entire Register to zeros. Bits cannot be stored in the Indicator Register by addressing it.

For the purpose of program interruption, the bits in the Indicator Register have a built-in priority. Priority decreases from left to right. The conditions are listed below in the order of decreasing priority.

5.8.1 Internal Malfunctions

0. Control Error
1. Information Error
2. Memory Error

5.8.2 Control

3. Invalid Instruction

An instruction has not been executed because the operation is meaningless.

4. Invalid Address

An instruction has not been executed because the address is meaningless in conjunction with the specified operation.

5. Elapsed Time

The Elapsed Time Clock, an internal clock, has gone through zero.

5.8.3 Streaming (Bits 6-30)

These are the conditions which result from and can interrupt streaming operations. They are listed in section 6.10.

5.8.4 External Units

For Indicators 33 to 38 the "indicated unit" is that identified by the External Unit Address Register.

31. Exchange Error

32. Select Reject

The Exchange rejected an instruction because the unit was already selected by a previous instruction. This Indicator should not be masked so that program interruption may occur before the computer takes up the next instruction.

33. Not Ready Reject

The Exchange rejected an instruction because the unit was not in condition to be operated. This indicator should not be masked so that program interruption may occur before the computer takes up

the next instruction.

34. Normal End

The last operation initiated for the indicated unit has been successfully completed.

35. Operator Signal

An operator's signal has been received from the indicated unit.

36. End of File

The indicated unit has reached an end-of-file condition.

37. Cancel

The last operation initiated for the indicated unit has been terminated without success. This indication does not include data error.

38. Data Error

The last operation initiated for the indicated unit has been terminated by a data error.

39. Other Computer

This Indicator is turned on by the program in another Computer connected to the same memory bus system.

5.8.5 Index Word Modification

These indicators retain the result of the test made during the last modification of an Index Word.

40. Length less than zero

41. Length equal to zero

42. Length greater than zero

5.8.6 Arithmetic

These indicators retain the tests made during the last arithmetic operation which affected them.

43. Result Overflow

44. Store Overflow

45. Register Overflow

46. Zero Divisor

47. Result less than zero

48. Result zero

49. Result greater than zero

50. Exponent Overflow

51. Exponent near Overflow — Reserved for floating point arithmetic

52. Exponent Underflow

53. Exponent near Underflow

5.8.7 Programmer's Tags

54. Instruction Tag

55. Index Tag

56. Data Tag T1 — Part of VFL sign byte

57. Data Tag T2

5.8.8 Not assigned (Bits 58-63)

5.9 Program Interruption

All machine conditions relevant to operational programs are retained in a 64-bit Indicator Register (see Section 5.8). The conditions may be interrogated by conditional branch instructions inserted at specific points in a program. Alternatively, the conditions may be permitted to interrupt a program automatically, i.e.,

when they occur and without specific tests being inserted in the program.

The Program Interruption feature is so designed that a program written without detailed knowledge of the feature may be interrupted at any time and control turned over to a master program. The often intricate procedure for taking care of the various conditions causing interruption can be a task for the master program. On completing the special procedure, the master program can return control to the interrupted program at the precise point of interruption.

If several conditions exist at the same time, each of which may cause interruption, a built-in priority system permits interrogation of each condition in turn. The program is aware of only one interruption condition at a time, specifically the one that had the highest priority at the time of the Interrupt. Other conditions are retained in the Indicator Register. When another interruption is permitted to occur, the one with the highest priority at that time will function; this includes all conditions held over as well as any which occurred in the meantime.

A 64-bit Mask Register is provided which permits the programmer to choose the conditions which may or may not interrupt his program. A 1 in a given position of the Mask Register permits interruption by the corresponding position of the Indicator Register; a zero in the Mask Register will prevent Interrupt for that condition. Thus interruption requires that a 1 exist in corresponding positions in both registers. A conditional branch instruction may still be used to interrogate any Indicator Register position, regardless of the state of the corresponding Mask Register bit. Having once interrogated a given condition, either by interruption or by conditional branching, that bit in the Indicator Register is turned off (set to zero). Interrogation does not alter the Mask Register.

When an interruption occurs during streaming,the stream is stopped immediately at the byte at which the condition is recognized.

When a condition requiring interruption occurs during the execution of non-streaming instructions, the interruption becomes effective at the end of the execution of the current instruction and before the next instruction has had a chance to make any irreversible changes. The possibility of an Interrupt does not prevent the instruction controls from fetching the next instruction and performing address modification; but all permanent changes, such

as branching or index word modification, are held up until the execution of the previous instruction has been completed and it is verified that no Interrupt will take place.

The Instruction, Index, and Data Tags cause an interruption before the execution of the instruction which obtained the tags.

When an interruption occurs, the next instruction to be executed is selected in a special way. Its location is determined by adding the position of the bit in the Indicator Register, which caused the Interrupt, to the contents of one or more Interrupt Address Registers. It will be possible to set these up so that certain stream interruption conditions can refer to 0.1 registers. The Instruction Counter, which contains the location of what would have been the next instruction if no Interrupt had taken place, is not consulted. (See Figure 5.7)

The position in the Indicator Register of the bit which caused the interruption is given by the Leftmost One Identifier (LOI). The LOI scans the masked outputs of the Indicator Register from left to right, in the direction of decreasing priority. When the highest-priority bit to cause Interrupt is found, the LOI turns it off and makes available 6 bits which represent the position of the Interrupt bit in the Indicator Register.

The 6-bit output of the LOI is added to the 6 low-order positions of a 20-bit Interrupt Address Register. (By means of a RESET INTERRUPT instruction, this register must have been previously set up to contain the starting address of a table of 64 instructions representing the 64 possible actions to be taken as the result of an interruption.) The selected instruction is placed in the Instruction Register for decoding and execution in the normal manner, after which the Instruction Counter again takes over control. The Interrupt Address register is not altered.

Usually, the table will contain a set of BRANCH instructions which store the old contents of the Instruction Counter at a suitable memory location and place the start of the program appropriate to the Interrupt in the Instruction Counter. If the special instruction found in the table does not change the Instruction Counter, the next instruction to be executed will be that selected by the old Instruction Counter contents. If the interruption occurs during the execution of a streaming instruction, the Instruction Counter contents will specify the same instruction. If a sequence of arithmetic mode instructions is interrupted, the Instruction

Counter contents designate the instruction after the last one executed.

As soon as an interruption occurs, the interruption mechanism is shut off to prevent any further interruptions from interfering with the program which attends to the current interruption. The interruption mechanism is reactivated by giving a BRANCH ENABLE instruction; usually this is also the instruction which returns to the interrupted program. Interrupt is automatically enabled whenever an instruction is given which puts the machine into the streaming mode. The first interruption may occur before streaming starts.

The RESET INTERRUPT instruction does two things. First it disables the Interrupt feature so as to permit the program to set up a new Interrupt procedure. Then it enters the effective Word Address into the Interrupt Address Register. Generally this instruction is followed by a RECEIVE instruction to reload the Mask Register, and BRANCH ENABLE, to enable the interruption mechanism again.

If it is desired to ignore and reset a given interruption condition without special programming, a BRANCH ENABLE instruction with zeros in the Word, Second, and Index Addresses is placed in the corresponding position of the 64-word table. When this interruption occurs, the interruption feature is immediately enabled again and the original program is allowed to continue (no branching).

5.9.1 RESET INTERRUPT

The interruption mechanism is turned off to prevent program interrupts from occurring after this and subsequent instructions. The effective Word Address, after indexing, is entered into the Interrupt Address Register.

If the effective Word Address is zero, the Interrupt Address Register is not altered; the RESET INTERRUPT instruction then merely disables the interruption feature.

5.9.2 BRANCH ENABLE

The BRANCH ENABLE instruction is similar to BRANCH except that is also enables the program interruption feature to function again if a previous interruption or a

RESET INTERRUPT had disabled it.

The first interruption may occur immediately after the Instruction Counter has been set to its new value by the BRANCH ENABLE instruction.

If the effective Word Address is zero, no branching occurs. The Instruction Counter contents are still stored and program interruption is still enabled.

5.10 Other Instructions and Features

5.10.1 IDLE

The IDLE instruction causes the program to wait indefinitely with the Instruction Counter set at the next higher instruction location. No operations are executed by the computer.

If program interruption had been enabled previously, an interruption condition set up by an external unit can restart the program at the point defined by the interruption feature. Otherwise the program must be restarted manually by pressing a Start key.

Only the Operation Code portion of the instruction has any meaning.

5.10.2 CLEAR MEMORY LARGE
CLEAR MEMORY SMALL

The effective word address of the instruction designates a block of memory which is cleared; that is, the next time any word in this block is referred to, all its information bits will be found to be zero.

For 0.5 microsecond memory CLEAR MEMORY LARGE clears a block of 64 words; CLEAR MEMORY SMALL clears a block of 8 words.

For 2.0 microsecond memory CLEAR MEMORY LARGE clears a block of 1024 words; CLEAR MEMORY SMALL clears a block of 64 words.

The appropriate high-order bits designate the address of the block to be cleared. When any address belonging to a block is given, all addresses within the block and the memory unit are cleared.

When memory units are grouped in fours with addresses sequenced through the entire group, only a block in one of the units is cleared. Thus, when memory units are grouped in fours, the two low-order bits of the word address indicate which of the four units is cleared; the words cleared then occupy every fourth address.

5.10.3 Elapsed Time Clock

The Elapsed Time Clock is intended to measure elapsed time over relatively short intervals of a minute or less. It can be set to any value at any time, and program interruption is available when the time period has ended.

The Elapsed Time Clock consists of a counter 20 bits long which is continually stepped down by pulses originating from a stable oscillator. The oscillator operates at 16,384 (2^{14}) cycles per second, or a pulse about every 61 microseconds. No attempt is made to correct the clock for long-term drifts.

The fifteenth stage from the right measures time in seconds. A full cycle is 64 seconds.

As long as the power is on, the clock never stops. Whenever it goes from zero to all ones, it turns on the Elapsed Time Indicator in the Indicator Register to permit Program Interrupt if desired.

The clock may be set to a new value at any time by using regular instructions which give the address of the clock. This turns off the indicator. The clock then continues stepping down from that value, and the first indication of having gone through zero signals the end of the present time interval.

Similarly, the contents of the counter may be read out at any time, with regular instructions. This does not change the setting of the clock or the indicator. The counter contents appear as bits 0-19 on the data bus.

(To get a time-of-day indication, a more conventional clock may be connected to the Exchange as an external unit. Such an external clock could be properly regulated and supplied from a separate power source so as to provide continuity over long time intervals without interference from computer operations. It could also serve to calibrate the internal clock. An external clock is not provided as part of the basic system.)

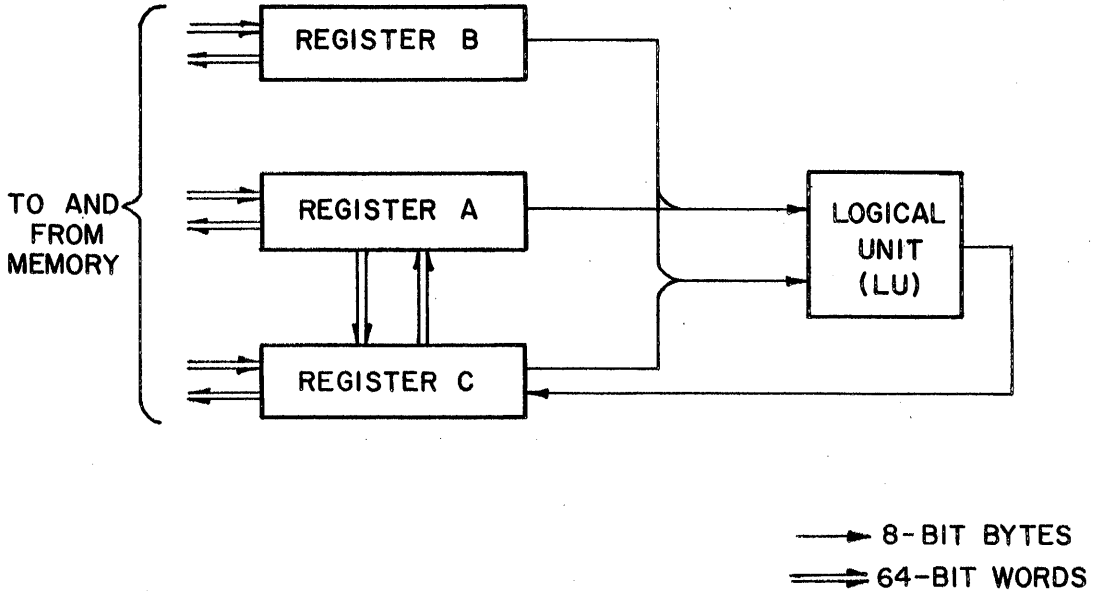
5.10.4 Preferred Alphanumeric Code

The Harvest computer has no built-in alphanumeric code because one of its outstanding characteristics is the ease with which it can translate from one code to another by programmed table look-up. It is expected that the computer will normally deal with external equipment using many different codes.

From the table look-up point of view, the only requirement of a usable code is that all distinguishable characters be represented by unique bit combinations. Certain additional characteristics are desirable, however. For sequencing operations, such as merging and sorting, a standard character sequence is desired, and it is at least preferable for the decimal digits to be directly usable for arithmetic operations in their coded form.

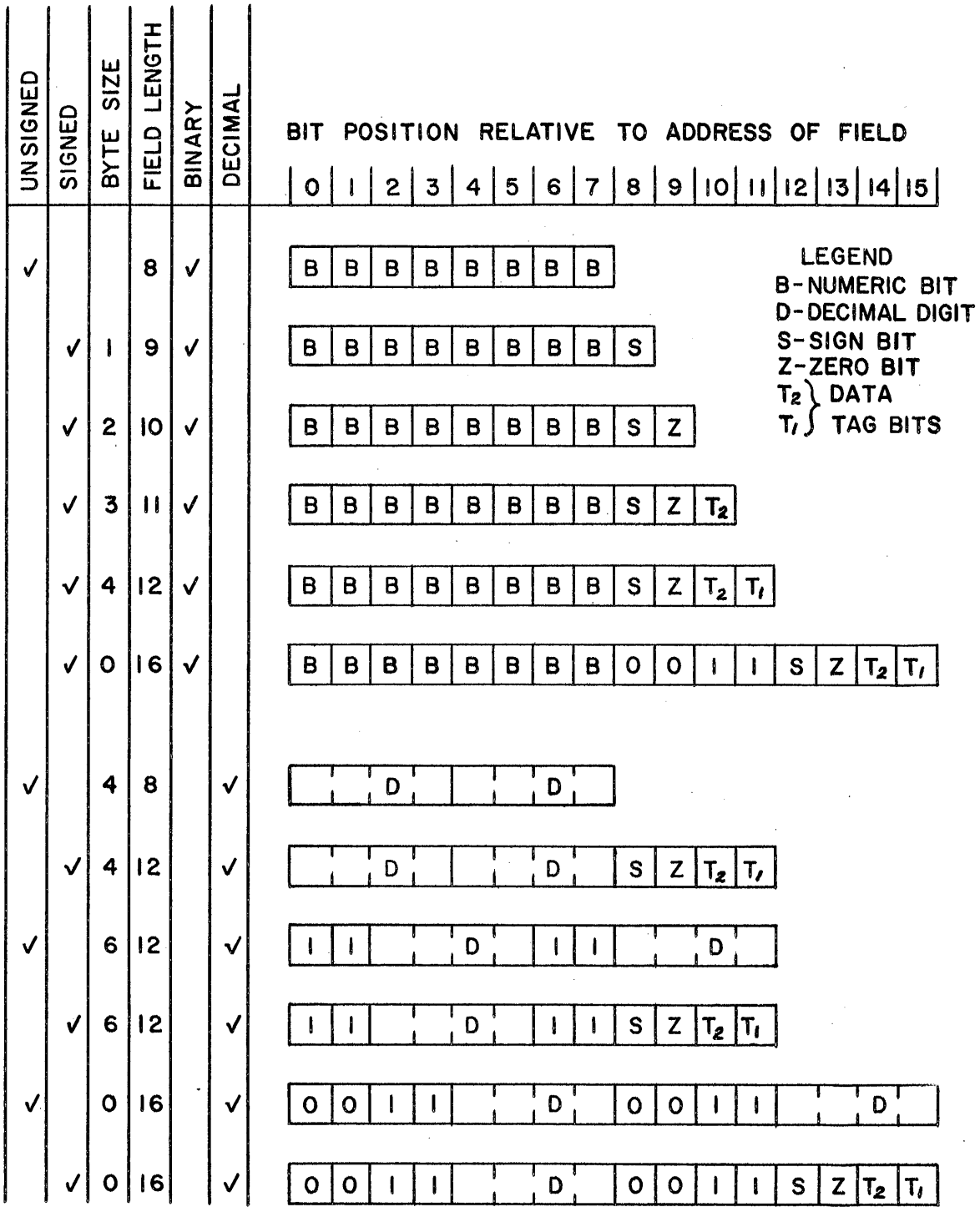
For this purpose, a preferred alphanumeric code has been drawn up in Figure 5.8. The six significant bits are shown in the leftmost six positions of an 8-bit byte. The dashes indicate that the rightmost two bits are arbitrary. When a six-bit byte is used, the two bits on the right are dropped.

This code exhibits the standard IBM character sequence when compared on a straight binary basis. The decimal digits are represented by the first ten binary integers in the middle four bit positions; the two ones on the left serve to preserve the alphanumeric sequence. Decimal arithmetic has been defined to provide these one bits automatically when larger than 4-bit bytes are chosen.



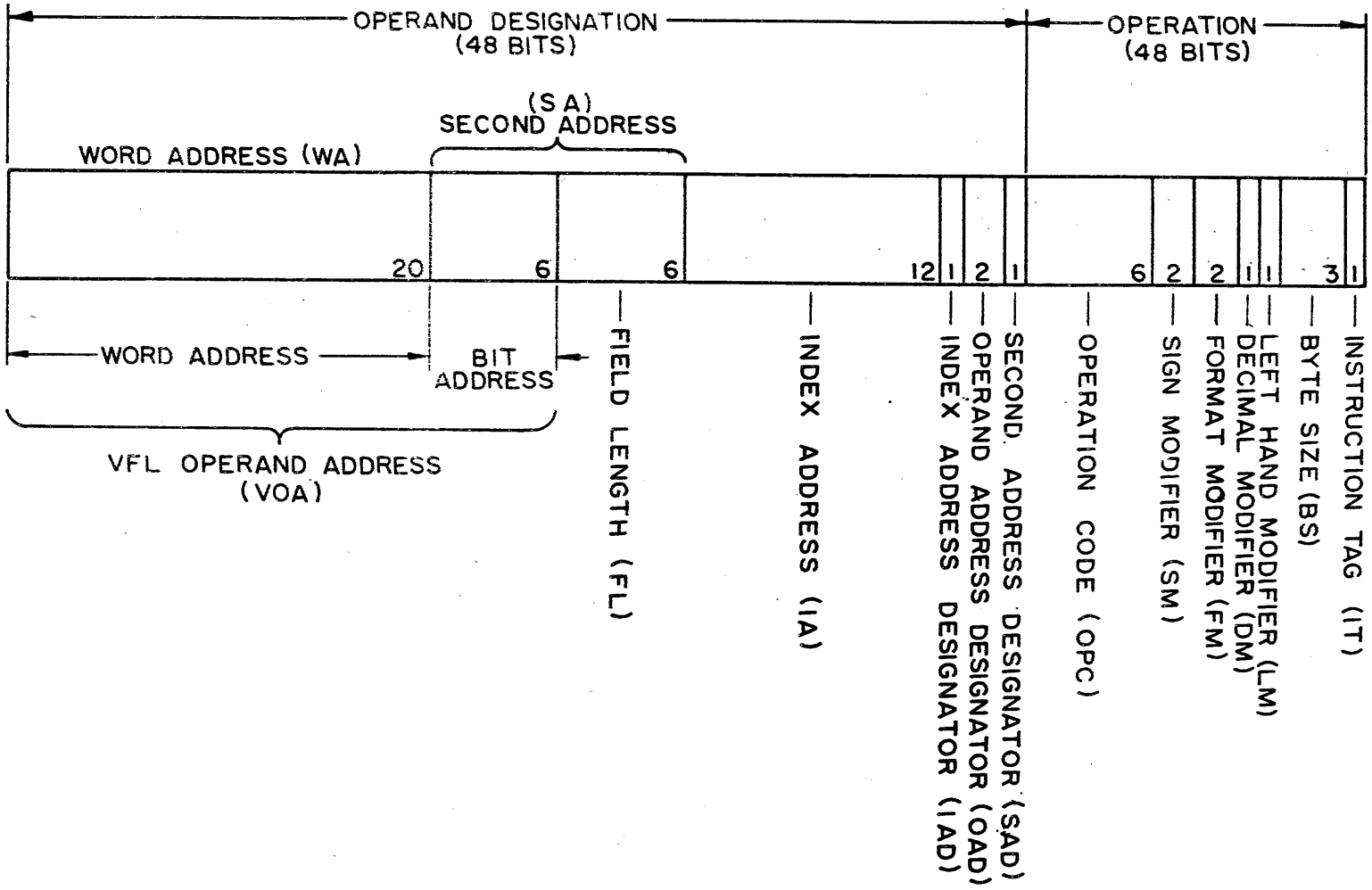
ARITHMETIC DATA PATHS

FIGURE 5.1



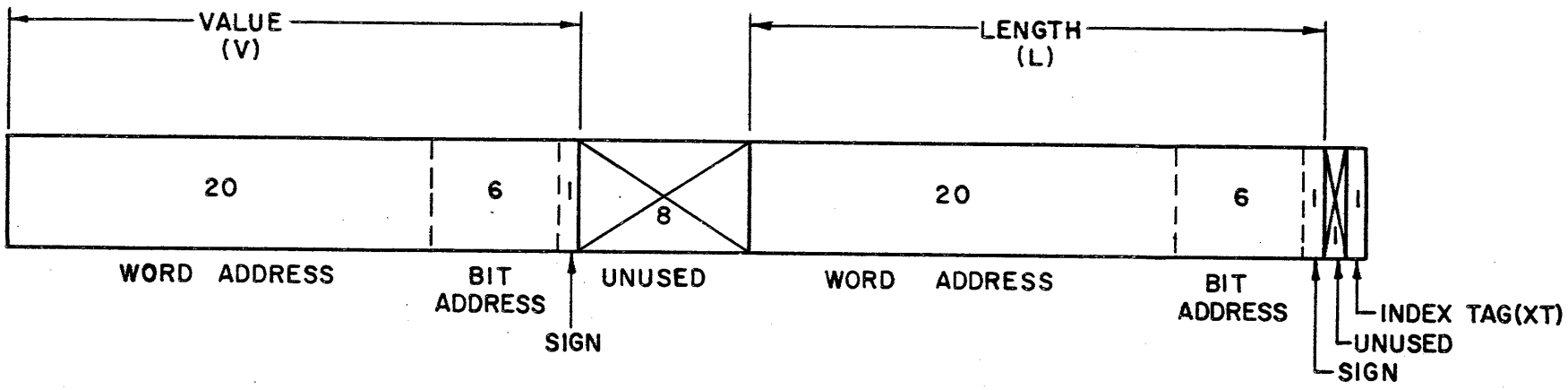
EXAMPLES OF VFL NUMERIC FORMATS

FIGURE 5.2



INSTRUCTION FORMAT
(ARITHMETIC MODE)

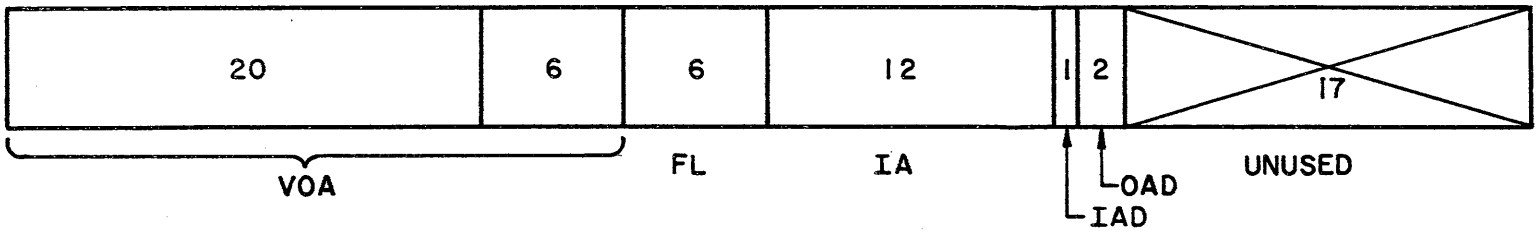
FIGURE 5.3



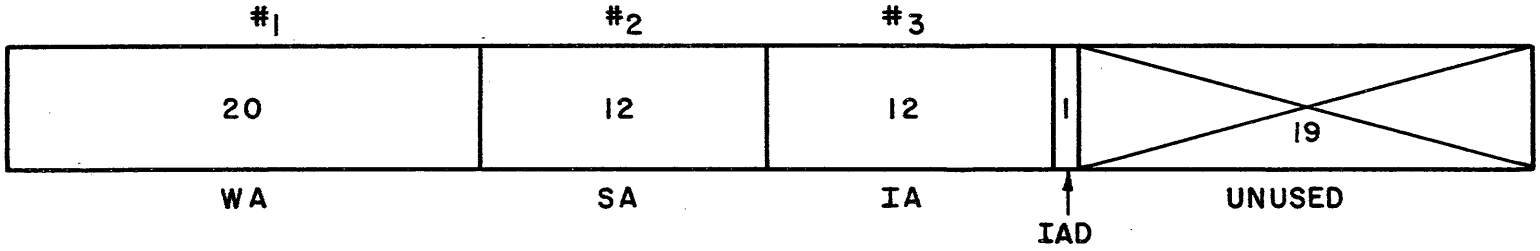
INDEX WORD FORMAT

FIGURE 5.4

INDIRECT ADDRESS WORD



MULTIPLE INDEX ADDRESS WORD



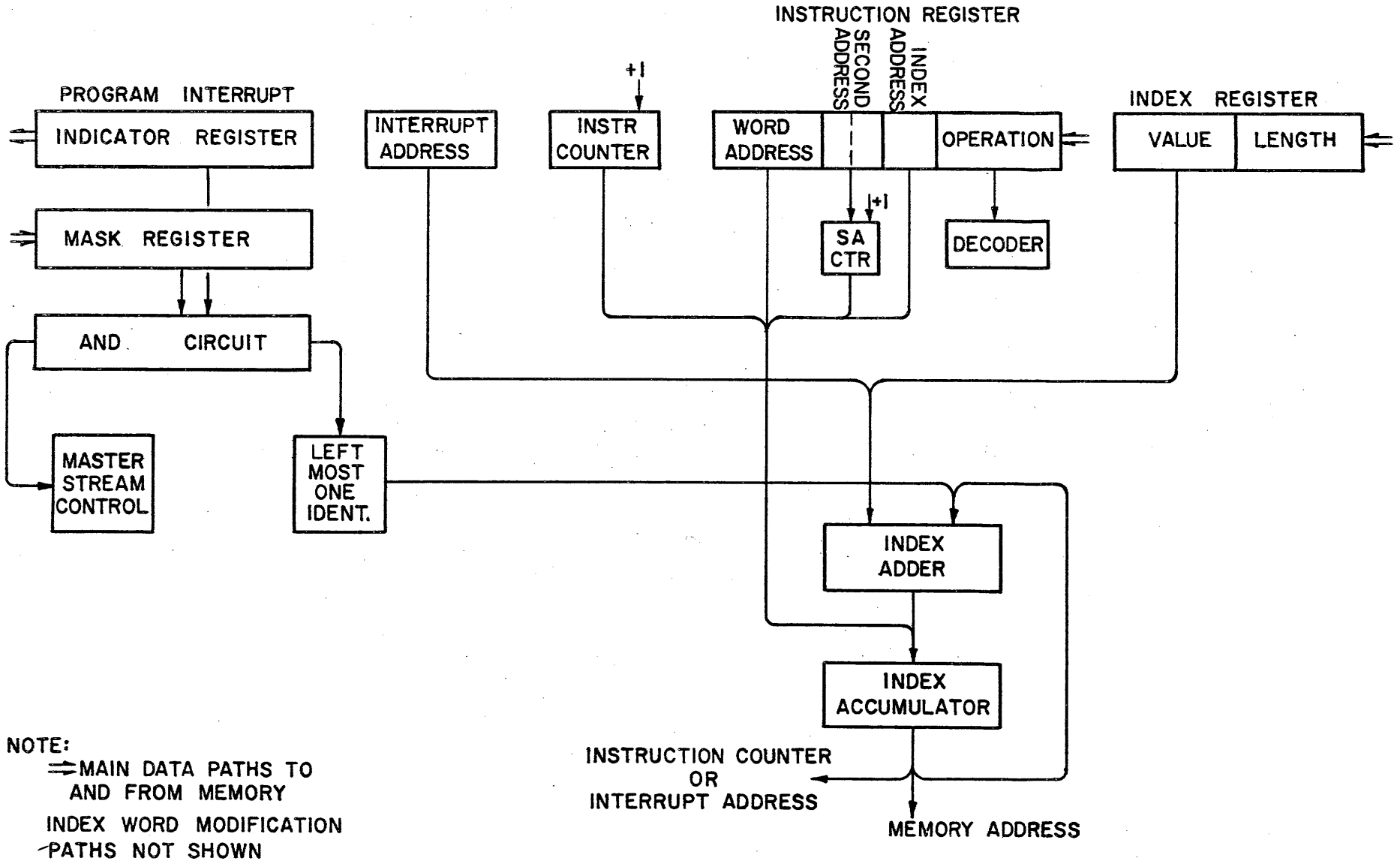
FORMATS DERIVED FROM INSTRUCTION WORD

FIGURE 5.5

(CM) CONNECTIVE CODE	LOGICAL FUNCTION	RESULT WHEN A B = 1 1	A B = 1 0	A B = 0 1	A B = 0 0
00	0	0	0	0	0
01	$\bar{A}\cdot\bar{B}$	0	0	0	1
02	$\bar{A}\cdot B$	0	0	1	0
03	$\bar{A}$	0	0	1	1
04	$A\cdot\bar{B}$	0	1	0	0
05	$\bar{B}$	0	1	0	1
06	$A \nu B$	0	1	1	0
07	$\bar{A} \vee \bar{B}$	0	1	1	1
08	$A\cdot B$	1	0	0	0
09	$A \equiv B$	1	0	0	1
10	B	1	0	1	0
11	$\bar{A} \vee B$	1	0	1	1
12	A	1	1	0	0
13	$A \vee \bar{B}$	1	1	0	1
14	$A \vee B$	1	1	1	0
15	1	1	1	1	1

TABLE OF CONNECTIVES

FIGURE 5.6



INSTRUCTION CONTROL SYSTEM

FIGURE 5.7

CHARACTER		BIT POSITION							
		0	1	2	3	4	5	6	7
(Blank)	b	0	0	0	0	0	0	0	0
	.	0	0	0	0	0	0	0	1
	¤	0	0	0	0	0	0	1	0
	&	0	0	0	0	0	0	1	1
	$	0	0	0	0	0	1	0	0
	*	0	0	0	0	0	1	0	1
	–	0	0	0	0	0	1	1	0
	/	0	0	0	0	0	1	1	1
	,	0	0	0	0	1	0	0	0
	%	0	0	0	0	1	0	0	1
	#	0	0	0	0	1	0	1	0
	@	0	0	0	0	1	0	1	1
		0	0	0	0	1	1	0	0
		0	0	0	0	1	1	0	1
		0	0	0	0	1	1	1	0
		0	0	0	0	1	1	1	1
	A	0	0	0	1	0	0	0	0
	B	0	0	0	1	0	0	0	1
	C	0	0	0	1	0	0	1	0
	D	0	0	0	1	0	0	1	1
	E	0	0	0	1	0	1	0	0
	F	0	0	0	1	0	1	0	1
	G	0	0	0	1	0	1	1	0
	H	0	0	0	1	0	1	1	1
	I	0	0	0	1	1	0	0	0
	J	0	0	0	1	1	0	0	1
	K	0	0	0	1	1	0	1	0
	L	0	0	0	1	1	0	1	1
	M	0	0	0	1	1	1	0	0
	N	0	0	0	1	1	1	0	1
	O	0	0	0	1	1	1	1	0
	P	0	0	0	1	1	1	1	1

CHARACTER		BIT POSITION							
		0	1	2	3	4	5	6	7
	Q	0	0	1	0	0	0	0	0
	R	0	0	1	0	0	0	0	1
	S	0	0	1	0	0	0	1	0
	T	0	0	1	0	0	0	1	1
	U	0	0	1	0	0	1	0	0
	V	0	0	1	0	0	1	0	1
	W	0	0	1	0	0	1	1	0
	X	0	0	1	0	0	1	1	1
	Y	0	0	1	0	1	0	0	0
	Z	0	0	1	0	1	0	0	1
		0	0	1	0	1	0	1	0
		0	0	1	0	1	0	1	1
		0	0	1	0	1	1	0	0
		0	0	1	0	1	1	0	1
		0	0	1	0	1	1	1	0
		0	0	1	0	1	1	1	1
	0	0	0	1	1	0	0	0	0
	1	0	0	1	1	0	0	0	1
	2	0	0	1	1	0	0	1	0
	3	0	0	1	1	0	0	1	1
	4	0	0	1	1	0	1	0	0
	5	0	0	1	1	0	1	0	1
	6	0	0	1	1	0	1	1	0
	7	0	0	1	1	0	1	1	1
	8	0	0	1	1	1	0	0	0
	9	0	0	1	1	1	0	0	1
		0	0	1	1	1	0	1	0
		0	0	1	1	1	0	1	1
		0	0	1	1	1	1	0	0
		0	0	1	1	1	1	0	1
		0	0	1	1	1	1	1	0
		0	0	1	1	1	1	1	1

PREFERRED ALPHANUMERIC CODE

FIGURE 5.8

6. System Operation in the Streaming Mode

6.1 General Description

The Harvest System is especially proficient for the performance of streaming operations in which a single simple operation is performed upon a large number of operands. In general these operands are bytes, but in some operations several of these are taken together as a field.

The serial (byte-wise) processing part of the Harvest System is shown in the diagram of Figure 6.1. This part of the system, the registers shown in Figure 4.1, and the instruction interpreting mechanism and general indexing mechanisms previously described constitute the Harvest Computer.

The machine in streaming mode is best understood by noting its analogy to a device in which ball bearings are conducted to and through various processing units by pipes. Single bytes of data are presented to the 8-bit bus by the Stream Switch Matrices A and B in the upper left-hand corner of the diagram. They pass through Byte Masks BM_1 and BM_2 which select subsets of the 8-bit bytes. They then pass either into the Logical Unit (LU) where they are combined in some prescribed fashion to make another byte, or into a Table Address Assembly unit where one or more bytes from A or B or both are combined to make a memory address. This address may in turn be used either to modify the contents of memory or to cause the contents of the addressed location to be presented to the byte bus by the Table Extract Unit.

The output bytes presented by the Logical Unit may be sent to an 8-64 converter which embeds them in words to be stored in memory. Alternately or simultaneously, they may be sent to a Statistical Accumulator (SACC) where they are accumulated with a current total, or back to the Table Address Assembler for use in a memory operation. The bytes presented to the bus from the Table Extract unit may also be sent to any of these destinations or back to the input of the Logical Unit.

Not only may bytes from memory be thus processed, but counts may be accumulated and presented to the busses for processing by the Numbering Counter (NCTR). At several Match Stations located along the bus system, the passing bytes may be

compared with one or more preselected Match Characters. If any byte is identical to one of the match characters, the streaming action may be interrupted or index levels adjusted. It should be observed that several successive bytes may be at different stations of the "pipeline" at any one instant.

The presentation of the original input bytes from memory is accomplished by the Stream Units, each of which consists of 1) one or more registers for the receipt of 64 bit words from memory, 2) a 64-8 Switch Matrix which selects any contiguous eight-bit subset of a 64-bit word or two consecutive 64-bit memory words, and 3) an indexing mechanism capable of selecting bytes from memory in orderly but complex fashions. The whole assembly may be considered as a device for mapping the memory of the system into a set of bytes that are presented as if read from a very high speed tape eight channels wide. This mapping from a fairly slow real memory into a very fast virtual tape may include overlapping, repetition, skipping, and other useful patterns.

A similar Stream Unit performs the opposite function of receiving the bytes presented to it and causing them to be stored at memory locations selected in a complex fashion without disturbing the contents of the rest of memory. The individual components of the Stream Units are shown on the diagram. The indexing mechanisms, together with those that control the Table Address Assembler and the Table Extract Unit, are shown as a single block labelled Index Controls. The 64-8 Switch Matrices of Stream Unit A, Stream Unit B, and the Table Extract Unit are labelled simply A, B, and E. The 8-26 Switch Matrix associated with the Table Address Assembler is labelled D, and the 8-64 Switch Matrix associated with Stream Unit C is labelled C. The registers themselves are shown only in dotted outline.

The byte sources, byte sinks, and byte processing units are linked by several possible data paths, shown in heavy lines in Figure 6.1. Selection of the paths which will be used for any particular operation is termed data gating. It is accomplished by designating which of the gates, symbolized by circles in the diagram, will be open to permit data to flow through them. A bit is provided in the stream instruction format for each of the gates; a one bit causes the corresponding gate to be open. The gates are numbered according to the address of the controlling

bit within the instruction word. Only one gating may be in effect during a single streaming operation; changes in gating are effected by stopping the stream and interpreting another stream instruction.

The SU's impart to data a structure of several levels. Each stream instruction provides for stopping the stream when some specified SU reaches the end of a byte, first level, or second level. Alternatively, no provision need be made for stopping the stream at the end of a level of indexing; in this case the stream is stopped only by interruption. Interruptions can be caused by any of the Match Units, by signals from the counters and Statistical Accumulator, by end of indexing levels in the SU's, or by external signals. Different causes of interruption automatically cause the program to proceed from distinctive points. Furthermore, certain conditions arising from the operation of the Logical Unit or Match Units may cause stream indexing to be modified automatically without interrupting streaming.

Some of the bits used to control these stream interruptions are set up at the time the streaming is set up. Others are specified in control gating; the diamond shaped symbols in Figure 6.1 show the gates that may be so controlled. Each gate is numbered with the address of the bit within the stream instruction that controls it.

6.2 Data Gating

The data paths that are to be used in a stream operation are specified by the fourteen bits in the stream instruction. Each bit corresponds to a gate in the data paths. Timing of bytes is controlled from downstream; that is, if A is connected to the Logical Unit, bytes pass from A only when the Logical Unit signifies that it is ready to receive a byte. Of course, A must also have a byte ready to present.

The actions of the gates are as follows:

Gate 0 is used to govern the parallel transfer of data into the 64-8 Switch Matrix. This matrix is shared by the B registers and by the Numbering Counter. When this bit is set to zero, Switch Matrix B operates from the registers of Stream Unit B. When it is one, the matrix presents bytes from the NCTR. This gate is completely independent of other gates.

Gates 1 and 2 govern the passage of bytes from Switch Matrices A and B, respectively, into the Logical Unit. The Logical Unit causes these bytes to be presented simultaneously whenever both Gate 1 and Gate 2 are open. Either Gate 1 or Gate 2 may be used alone for operations that require only one input.

Gates 3 and 4 govern the passage of bytes from Switch Matrices A and B respectively into the Table Address Assembler. When only one of these gates is open, the TAA causes bytes to flow from the corresponding unit until a complete address, as designated by the TAA first level length, is formed. This address is sent to memory and the TAA indexing controls are reset for the next repetition of the operation. When both Gate 3 and Gate 4 are open, bytes are taken from Switch Matrix A until the TAA first indexing level is ended, then bytes are taken from B until the table address is complete. Increment I_1 is used for spacing the bytes taken from A, increment I_2 is used for spacing the bytes taken from B. Gates 3 and 4 may be used in any combination with Gates 1 and 2.

Gates 5 and 13 govern the entry of the one-bit signal output from the Logical Unit into the high-order (most significant) bit position of the byte bus. Gate 5, when open, causes the one-bit signal to replace the high order bit of the normal LU output at a point below the Match Station associated with the LU and above all the gates that distribute the LU output byte. Gate 13, when open, causes the one-bit signal to replace the high-order bit of the normal output if Gate 7 is open. If Gate 7 is closed, the one-bit signal enters the high-order bit position and zeros are entered on the other bit positions. Gate 13 thus permits sending the one-bit LU output to the Statistical Accumulator even though the eight-bit LU output is being sent to the TAA or to C. Gates 5 and 13 may be open at the same time, and this mode of operation will prove desirable in some cases. The operation of either of these can be meaningful only when there is at least one input to the Logical Unit, since it will not emit a signal without some input. Gates 5 and 13 are independent of Gates 3 and 4.

Gates 6, 7, and 8 govern the passage of the byte output from the LU to C, to the Statistical Accumulator, or to the Table Address Assembler, respectively. The timing of the passage of a byte presented by the LU into one of the downstream units is also governed by the downstream unit, except that if more than one

of Gates 5, 13, 6, 7 and 8 are open, the output is passed only when all of the downstream units to which the LU is connected are ready for it. Gates 6, 7 and 8 are meaningfully open only when there is at least one input to the LU. Any combination of them may be open at the same time, and any of them may be open when Gate 5 is open. Gate 8 may not be open when Gate 3 is open. When Gates 5 and 7 are open, it is redundant to open Gate 13.

Gates 9, 10, 11 and 12 govern the passage of bytes from the Table Extract to the Logical Unit, C, the Statistical Accumulator, and the Table Address Assembler, respectively. Timing is governed by the readiness of all connected downstream units, as is the case with Gates 6, 7 and 8. Gate 9 may not be open when Gate 2 is open. Gate 10 may not be open when Gate 6 is open. Gate 11 may not be open when Gates 7 or 13 are open. Gate 12 may not be open when Gate 4 is open. Some caution must be used to insure meaningful and terminated operation whenever Gates 8 and 9 are open for the same operation or whenever Gate 12 is open.

When Gate 8 or Gate 12 is open, the Table Address Assembler operates with respect to the sequential acceptance of data and the information of addresses just as if Gate 3 or 4 respectively were open.

Gates R and $\overline{R}$ are not subject to program control. Gates R are open only during runout operations, at which time Gates $\overline{R}$ are closed.

6.3 Automatic Address Modification for Streaming

6.3.1 Patterns of Address Modification

As described earlier, each Stream Indexing Mechanism automatically performs a complex selection of bits from memory. These are received in the Stream register and presented, via the Switch Matrix, as a series of 8-bit bytes.

Generally speaking such a flow of data fits a well defined pattern which may be simple or complex. For example, a stream may consist of one-thousand consecutive six-bit bytes beginning at a particular address. This is a

simple stream that requires only one level of address modification.

A more complex stream may be described as follows (see Figure 6.2): read out three overlapping eight-bit bytes (each byte overlaps the last half of the preceding byte) and then skip six bits; repeat this three times, each time moving the effective starting point 22 bits, thus giving a six-bit skip; repeat this entire process five times using the original starting point; finally each time all previously described steps have been completed repeat them at a new starting point one-hundred bits from the original starting point, continuing such repetition until a total of ten cycles have been completed. This complex stream requires four levels of address modification.

A third and simpler type of stream pattern is characterized by the following example (See Figure 6.3). A record consists of twelve 4-bit decimal digits. It is desired to read out three digits, skip two, read out two, and then go to the next record. Five levels of address modification are used to produce this pattern.

6.3.2 Control of Parameters

In order to fully determine any of the stream patterns described above, the programmer must specify:

a) S = Start Point Address

b) BM = Byte Mask (regulates Byte Size)

and for each level, k, of address modification needed:

c) $I_{(k)}$ = Increment (in bits)

d) $L_{(k)}$ = Length (in bits), the product of the Increment and the number of iterations to be performed on the level

e) Several control bits (described in Section 6.3.8)

f) $A_{3(k)}$ = Address of 3rd-level setup words

During the operation an auxiliary quantity is generated in the index mechanism:

g) R_k = Residual Length, the Length minus the sum of all increments already made on this level. Initially $R_k = L_k$.

In the first of the above examples the values of the parameters are B = 6, I = 6, and N = 1000.

In the second example the byte size is B = 8. The first level of address modification requires that $I_1 = 4$, $N_1 = 3$ (N_1 giving the three eight-bit bytes, I_1 locating these bytes four bits apart), and $L_1 = 12$. The second level of modification gives $I_2 = 22$, $N_2 = 3$ (N_2 giving the three repetitions of the first level pattern, I_2 advancing the address over sixteen bits for the overlapped bytes plus six bits for the skip), and $L_2 = 66$. For the third level $I_3 = 0$, $N_3 = 5$ (five repetitions of the entire two level pattern using the same starting address) and by special definition, $L_3 = N_3 = 5$, while the fourth level has $I_4 = 100$, $N_4 = 10$ (ten repetitions of the entire three level pattern using starting addresses one-hundred bits apart), and $L_4 = 1000$.

For the third example, B = 4, since the characters are in 4-bit decimal notation. Now, referring to Figures 6.2 and 6.3, one can see that the pattern of address modification is fundamentally different for the second and third examples. In the second example, after each new second level increment is added, the entire cycle of first level incrementing and resetting takes place. In fact after each increment is added on any level, the entire cycle of incrementing and resetting for all lower levels takes place. Moreover, just prior to incrementing on a higher level, the next lower level is reset. (The length, L, is subtracted from the Effective Address.) However, in the third example after the entire first level cycle of incrementing takes place, the first level is not reset. The second level increment is added immediately, and again no reset occurs. Then the third level cycle takes place. The fourth level is used to reset the Effective Address (I_4 is negative). Now this four level pattern is repeated for each fifth level increment.

The values of the parameters for the third example, then, are as follows: $I_1 = 4$ (equal to the byte size) and $N_1 = 3$ (to read out three characters); $I_2 = 8$ (to skip two characters) and $N_2 = 1$; $I_3 = 4$ (the byte size again) and $N_3 = 2$ (to read out two characters); $I_4 = -28$ (to reset to beginning of the record) and $N_4 = 1$; finally, $I_5 = 48$ (to jump over twelve 4-bit decimal digits to the beginning of the next record) while N_5 = total number of records to be processed.

In particular, note that in the second example the first level Increment does not equal the Byte size (as it does in the first example). Negative Increments are also permissible, in which case the Length is also negative. Moreover, an Increment of zero (I_3 in the second example) is perfectly permissible (whence, as indicated above, by special definition, the Length equals the Number of Iterations), as is a value of zero for the Number of Iterations (in this case the Length is defined to be zero). The latter means only that for the level of modification that has N = 0, iterations will continue indefinitely. For example, it is desired to read a record of indefinite length a byte at a time. Here a special character might be appended to the record and N be set to zero. A Match Unit would be used to terminate streaming upon recognition of this end-of-record symbol.

6. 3. 3 Parameter Interpretation for Stream Unit C

With respect to Stream Unit C, the Start Point Address is the address at which the unit will start storing the output stream. The interpretation of the other parameters remains as described in Section 3 above.

6. 3. 4 Parameter Ranges

The parameters for the first level of control are limited to the following sizes: I must be on the range -255 to +255 (including the extreme values) while L = NI must lie on the range $-(2^{20} - 1)$ to $+(2^{20} - 1)$ (including the extremes). For the second and higher levels, the parameters range as follows: I may have one of the values $-(2^{26} - 1)$ through $+(2^{26} - 1)$, and NI may not be less than $-(2^{26} - 1)$ nor greater than $+(2^{26} - 1)$.

6.3.5 Levels of Control

There is no practical limit to the number of levels of automatic address modification that may be associated with the Stream Units. The control parameters for each of these are independent of those used with the other two.

6.3.6 Control Setup

The setup data for the streaming mode of operation of the Harvest Computer must include the address of the first word of a separate block of memory for each Stream Unit to be used.

The first word of each block used contains S, BM, and I_1; the second contains R_1, the control bits for level 1, and L_1; the next two contain A_3, I_2, R_2, the control bits for level 2, and L_2. If higher levels are present, they require two words each and contain I_k, R_k, control bits of level k, and L_k. Field A_3 designates the first control word for the third level. The second word and the words for higher levels must follow consecutively. The values of all parameters must be given in binary notation. A RECEIVE instruction is used to set up Harvest for streaming. At the appropriate point in the interpretation of this instruction the parameters for the first two levels of address modification are automatically entered into the Control registers of the particular Stream Unit being initialized. The parameters for the third and higher levels of control are automatically made available as needed, their temporary values being held in the memory words that held their initial values.

A RECEIVE instruction may be used to set up (or change the setup of) a particular Stream Unit. Here the RECEIVE instruction carries the address of the first word of the block of memory words that contains the control parameters. Finally, a TRANSMIT instruction may be used to load any Control Register associated with any Stream Unit directly from the Extract Register or to move the contents of any Control Register into the Extract Register.

6.3.7 Control Bits and Interrupt Signals

Associated with the parameters of each level of control

is an END OF LEVEL control bit. If on a particular control level this bit is set to the value one, then the particular Stream Unit being controlled by this set of parameters will stop reading out bytes after the last iteration is executed on the specified control level. In addition, the parameters of this control level will be reset; i. e., R will be replaced by L and if RESET SUPPRESS is OFF - see below - the Effective Address will be decremented by L. However, at this point the normal control sequence (cf. Chapter 4) is interrupted. An Indicator Bit that may be sensed for interruption is set to the value one. No further action takes place at this Stream Unit until initiated by subsequent programmed instructions (cf. Section 6. 3. 8, below).

END OF LEVEL control bits may be set on any number of control levels associated with a given Stream Unit. Each register has an associated End of First Level Indicator Bit, an End of Second Level Indicator Bit, and an End of Higher Level Indicator Bit.

Thus, if several END OF LEVEL control bits are set for levels higher than the second, the control sequence at the end of each such level will cause the End of Higher Level Indicator Bit to turn on. In this situation it will generally be necessary to determine by programmed instructions which level has caused the Indicator Bit to be set. If k-th and higher level END OF LEVEL bits are ones, the level that caused the Indicator Bit to turn on may be determined by testing the parameter R associated with the highest level concerned. If this R is equal to L, the associated level set the Indicator Bit; if R is not equal to L, a lower level set the Indicator Bit.

A second control bit associated with each level of control is the READ SUPPRESS bit. The second example of Section 6. 3. 2 above requires that bytes be read out prior to each first level advance only (i. e., prior to each time I_1 is added to the Effective Address). This means that READ SUPPRESS is On (set to one) for all levels higher than the first. The third example above requires byte readout prior to each advance of both the first and third levels. Thus READ SUPPRESS is On for the second, fourth and fifth levels.

Finally, a RESET SUPPRESS bit is available to the programmer. If this bit is On (set to the value one), then after the particular level concerned, the k-th level, say, has advanced N_k times, R_k is replaced by L_k, but the Effective Address is not decremented by L_k. Furthermore, N_{k+1} advances on the (k + 1)-th level will take place (with or without readout) before the first level can regain control. On the other hand, if this bit is Off, the Effective Address (after N_k advances on the k-th level) is decremented by L_k and control immediately reverts to the first level. The third example above has the RESET SUPPRESS bit On for the first four levels, but not for the fifth level, while the second example has the RESET SUPPRESS bit Off for all levels.

6.3.8 Programmed and Automatic Adjustment of Indexing Levels

Modification of the normal indexing control sequence, known as index adjustment, may be performed automatically during the course of streaming by certain signals, according to controls specified by the programmer in the stream instruction. These same modifications may be performed after a stream has been interrupted or has reached its normal stop.

The operations of index adjustment are:

0. Byte Repeat: R_1 is incremented by I_1 and the effective address of the next byte is decremented by I_1 so that the last byte is repeated.

1. First Level Advance: R_1 is decremented by I_1 and the effective address is incremented by I_1 so that the byte which would normally be next is skipped.

2. First Level Reset: R_1 is reset to L_1 and the effective address is decremented by $L_1 - R_1$.

3. First Level Runout: Operation (1) is repeated with the successive bytes being presented on the special Runout bus. Normal indexing control

sequences take place until the end of the first level is reached, when a second level advance is executed. The byte first presented is the same as the byte in the Logical Unit when the signal is given. The end-of-first-level Indicator is not turned on.

4. Second Level Advance: Operation (2) is executed, then R_2 is decremented by I_2 and the effective address is incremented by I_2.

5. Second Level Reset: Operation (2) is executed, then R_2 is reset to L_2 and the effective address is decremented by $L_2 - R_2$.

6. Second Level Runout: Operation (3) is repeated until the end of the second level is reached, when a third level advance is executed. The end-of-level Indicator is not turned on.

7. High Level Runout: Operation (6) is repeated until the end of a high level is reached. The end-of-high-level Indicator is not turned on.

The index adjustment operations are controlled by the Index Adjuster (ADJ). This can be gated to receive the one-bit output signal from the Logical Unit or the signal from any of the Match Units. It has three modes of response to its input signal. One of these is selected for a streaming operation by bits 44 and 45 of the instruction. The modes are:

00 Adjuster output is the same as input.

01 The Adjuster is set to one when the first one input is received and is only reset when its output initiates some action.

10 The Adjuster changes state from zero to one or one to zero each time as one is received.

A fourth mode, 11, may be specified in the instruction. In this mode the Adjuster emits a one output when the instruction is interpreted and before streaming actually

commences. It is not active during the streaming operation proper. This mode may be used for programmed adjustment of index levels, with or without any other action being specified in the stream instruction.

The output of the Adjuster goes to a bit in the Indicator Register and may cause stream interruption. The programmer may also specify any appropriate one of the eight index adjustment operations to be performed on each Stream Index Mechanism whenever the Adjuster output is a one. This selection is specified by bits 26-43 of the stream instruction as follows:

Bits	Value	Action
26 - 29	0000	No adjustment of Unit A
	1000	Operation (0)
	1001	Operation (1)
	...	...
	1111	Operation (7)
30 - 33	0000	No adjustment of Unit B
	1000	Operation (0)
	...	...
	1111	Operation (7)
34 - 37	0000	No adjustment for Mechanism C
	1000	Operation (0)
	...	...
	1111	Operation (7)
38 - 40	000	No adjustment for Mechanism D
	001	Operation (1)
	010	Operation (2)
	011	Operation (3)
	100	Operation (4)
	101	Operation (5)
	110	Operation (6)
	111	Operation (7)
41 - 43	000	No adjustment for Mechanism E
	111	Operation (0)
	001	Operation (1)
	010	Operation (2)
	011	Operation (3)

6.4 The Functioning of the Logical Unit in Streaming

During streaming one may obtain any possible function of a set of bytes by means of the table lookup facility. Nevertheless a separate Logical Unit is provided. There are several reasons: many functions are extremely simple and regular; table lookup slows down when references to memory occur in a random order; large tables can often be reduced in size by first performing a simple function on the operands; if a table is to be used, it must first be read into memory; built-in functions in the LU are always immediately available.

The LU accepts an input byte from either or both of two sources and produces one output byte. In a streaming operation the LU performs the same function upon each sequentially presented set of inputs. The function, routing of the bytes, and other details of stream operation are specified in advance by means of a series of setup instructions and by the stream instruction itself. The setup remains fixed until specifically altered.

In addition to the byte output the LU also produces an associated one-bit signal. This signal

a) may control the Index Adjuster

b) may cause the Statistical Counter or the Numbering Counter to step

c) may be gated into the output channel in place of the high order bit of the normal byte output

d) may be directly entered into the eighth bit position from the right of the Statistical Accumulator

e) may cause a stream interrupt.

The 8 lines from Registers 1 and 2 lead directly to the 8 positions of the LU. The lowest-numbered bit (the left-most or high-order bit) of a register goes to the left-most position of the LU. The 8-bit outputs from LU correspond bit by bit to the inputs. All 8 bits travel to each of the three possible recipients. The Statistical Accumulator receives the 8 in its

right-most (low-order) bits. Switch Matrix C eliminates unwanted bits by means of B_3. The Table Address unit assembles its data by Or-ing, and unwanted bits travelling as 0's have no effect there.

If the 1-bit generated in the LU is gated into the output in place of the normal output, it is put in the left-most bit. This is natural since all read-out is measured from the left. If this modified output is sent to the Statistical Accumulator - or if the one-bit is put in directly - the entry is into the 8th bit from the right. This means that the Statistical Accumulator will be stepping by 128 rather than by 1. A single bit, 62, in the stream instruction governs whether the one-bit signal from the LU will be that specified in the operation description (bit 47 is 0) or its inverse (bit 47 is 1).

The LU functions are listed below: (A and B represent the two inputs)

Title		Byte Output	1-bit output = 1 if
Stream Logic - Non Zero		Logical Connectives 0 - 15 as shown in Figure 5.6	Logical Connection is $\neq 0$
Stream Logic - odd parity		Logical Connectives 0 - 15	Parity of Logical Connectives = 1
	Modifiers		
Stream Select	0000	Max. (A, B)	A > B
	0001	Max. (A, B)	A = B
	0010	Max. (A, B)	A < B
	0100	Min. (A, B)	A > B
	0101	Min. (A, B)	A = B
	0110	Min. (A, B)	A < B
	1000	A when A = B, otherwise 0	A = B
	1001	A when A $\neq$ B, otherwise 0	A $\neq$ B
	1010	A - B if A > B, otherwise 0	A $\geq$ B
	1011	B - A if B > A, otherwise 0	B $\geq$ A

Title	Byte Output	1-bit output = 1 if
Stream Modular Add-Carry	A + B mod M	A + B $\geqslant$ M
Stream Modular Add-Zero	A + B mod M	A + B = M
Stream Modular Subtract	A - B mod M	A = B
Stream Modular Reverse Subtract	B - A mod M	A = B

For logical purposes bytes are regarded merely as an ordered set of bits. For comparison purposes bytes are regarded as unsigned - i.e., positive - numbers in binary form. For modular arithmetic bytes are regarded as unsigned numbers in binary form and as already reduced to a value between 0 and M - 1, inclusive.

For logic and comparison there is no restriction on B_1 and B_2 although one must carefully calculate what effects the masks introduce. For diminish and modular arithmetic B_1 must equal B_2 and have 1's in the left-most positions corresponding to the byte size and 0's in the waste positions on the right. For modular operations A, B and M are all represented by the same size byte, this size being just sufficient to represent M. In the LU they are positioned to the left. When M is specified in the setup, the pertinent bits are to the left; empties on the right are filled with 0's.

6.5 Table Lookup and Modification as a Part of Streaming

The Table Address Assembler and Table Extract Unit are provided to permit the reading and altering of tabular quantities stored in memory. Tabular quantities are stored in memory cells which are obtained by dividing memory words into submultiples of 1, 2, 4, 8, 16, 32, or 64 bits. Thus a memory cell is a Field of length 2^K, $0 \leqslant K \leqslant 6$, whose left-most bit has a bit address of $m \cdot 2^K$, $0 \leqslant m \leqslant 2^{(6-K)}-1$. Four operations are possible, and two bits, 46 and 47, of the streaming instruction indicate which is to be performed:

00 Clear the small block of memory containing the addressed location

10 Transfer the contents of the addressed memory word to the Table Extract Unit and set its controls to read out only that cell

11 Count or Or in the addressed memory cell and transfer the contents to the Table Extract Unit, setting its controls to read those contents.

The Table Address Assembler provides for the acceptance of bytes from each of two different sources. Each source may have its own byte size. As indicated in the section on Data Gating, bytes from one source are taken until a field from that source, defined by L_1 in the TAA controls, is complete; then bytes from a second source are accepted until the second field, defined by L_2 in the TAA controls, is complete. The address formed by the sequence of bytes thus assembled is Or-ed with the contents of a Table Base Address register which provides the bits necessary to complete a twenty-six bit address. A two-bit counter, the Memory Distributor (MD), may be so arranged that it steps after each complete address is formed and permits successively formed addresses to refer to successive memory boxes regardless of the byte data used to form the address. Bit positions 0 and 1 of the assembled address are discarded. (If either is different from zero, the Invalid Address Indicator is turned on.) Bit positions 2 - 19 of the formed address become bit positions 0 - 17 of the address sent to memory. The MD contents become bit positions 18 and 19 of the address sent to memory. Bit positions 20 - 25 of the address go straight to the Table Extract Unit.

When the complete address is formed, the word part is sent to memory and the TAA indexing controls are reset for another operation. The bit address is sent to the Table Extract Unit to govern the read-out of the cell contents.

The Table Address Assembler indexing controls are simple, providing only an initial address (S), a first level increment (I_1), a second level increment (I_2), a first level field length (L_1), and a second level field length (L_2). The sum of L_1 and L_2 may not exceed twenty-six bits.

The Table Extract Unit consists of a 64-8 Switch Matrix E, and one or more registers that receive the word transferred from memory. Associated with E is a set of first level indexing controls that provide an initial address, an increment, and a field

length. These controls are usually set from the information formed by the Table Address Assembler. No table entry may lie across the boundary between two 64-bit memory words, so that a 6-bit increment, and a 6-bit first level field length suffice.

Setup formats for the Table Address Assembler and the Table Extract Unit are shown in Figure 6.6. R_1 and R_2 are set to L_1 and L_2 at the time of stream setup. Other values may be stored and reloaded in case of stream interruption. The start point address for the Table Extract Unit is usually furnished by the Table Address Assembler in the table lookup process.

The Table Extract Unit is also used by a stream type instruction. STREAM INSERT BYTE causes the instruction word to be sent to the Table Extract Unit, whose controls are set to cause the eight bits 54 - 61 of the instruction to be entered on the byte bus. This is known as the immediate byte. Any Data and Control Gating may be used to direct the byte through a desired stream operation.

6.6 Operation of the Statistical Accumulator and Statistical Counter

These two units are used primarily for making certain simple statistical calculations upon a stream of bytes without slowing down the streaming process.

During the process of counting in memory, it is possible to obtain the previous count, f_i, from the memory and add this count into the Statistical Accumulator before returning the new count to the memory.

When using the Logical Unit to compare successive bytes from A with the corresponding bytes from B, it is possible to send the bit 1 or the bit 0 to C depending on whether the bytes are equal or not, while at the same time accumulating the sum of the successive bits in the Statistical Accumulator and counting their number in the Statistical Counter.

Both the Statistical Accumulator and the Statistical Counter have overflow indicators which allow interruption of streaming and transfer to ordinary programmed operation.

The Statistical Counter can be connected to count the entries made to the Statistical Accumulator, to count the 1's appearing

in the sequence of 1-bit signals generated by the Logical Unit, or to count the bytes entered into Stream Unit C.

A streaming operation specifies the gating which determines the sources of information for the Statistical Accumulator and the Statistical Counter. This gating has already been discussed elsewhere, but the method of setting up the initial contents of the special registers and their modes of operation will be described here.

The Statistical Accumulator and its Threshold Register are set up as shown in Figure 6. 7.

Only the left-most 2 of the 8 bits of the control field are needed in controlling the mode of operation.

The first control bit determines whether the Statistical Accumulator is to interpret the 8-bit numbers it receives for accumulation as unsigned 8-bit numbers, or as 7-bit numbers with signs on the right. A zero bit causes unsigned accumulation, a one bit causes signed accumulation. In unsigned accumulation, the 8 data bits enter the right hand 8 positions of the register (positions 18 to 25), not counting the sign position. In signed accumulation, the left 7 data bits enter positions 18 to 24 of the Accumulator and a zero bit is filled in position 25. The rightmost (sign) bit enters the sign position.

If the second bit is 1, the Statistical Accumulator will refuse to retain negative totals and will reset itself to 0 whenever such a total occurs. If it is 0, the Accumulator will retain negative totals.

An indicator will be turned on and stream interruption will be permitted whenever the sign of the accumulated total changes from plus to minus. No indication is given when the sign changes from minus to plus.

Interruption is optional under control of the interruption mask.

Whenever the total in the Statistical Accumulator equals or exceeds the threshold value, an indicator will be turned on, allowing a stream interruption.

The Statistical Counter is set up according to figure 6. 7.

Each of the three Statistical Registers has an address and can be used by means of regular instructions for clearing, loading, and

storing.

The STREAM INSERT BYTE can be used for entry of single bytes into the SACC. The SACC mode last specified by the control bits governs the STREAM INSERT BYTE instruction.

6.7 The Functioning of the Numbering Counter in Streaming

It is often desirable to assign a serial number to fields or records which are passed through the computer or which are formed during a streaming operation. It is also desirable to cause successively formed memory addresses to refer to different tables. For these and other purposes there is provided a Numbering Counter, NCTR, sixteen bits long. This counter is provided with an overflow detector which can cause stream interruption. It may be set to some initial value or stored by an ordinary Receive or Transmit instruction, as shown in Figure 6.7.

The NCTR may be connected to count fields read from Stream Unit A, to count the number of addresses sent to memory from the Table Address Assembler, or to count one-bits from the Logical Unit.

The contents of the Numbering Counter may be entered into the processor a byte at a time by using Switch Matrix B and the indexing controls for Stream Unit B. This is accomplished by setting Gate 0 to one.

6.8 The Control Gating

The diamond-shaped symbols on Fig. 6.1 represent gates in the one-bit control lines which may be opened to connect the various counters and their actuating signals.

Gates 14, 15, and 16 gate the signals that actuate the Numbering Counter. Gate 14 causes the NCTR to count the number of memory addresses formed in the Table Address Assembler. Gate 15 causes it to count bits whose value is one in the one-bit signal output of the Logical Unit. Gate 16 causes the NCTR to count fields passed from Stream Unit A as defined by the end of the first level of indexing. Only one of these three gates may be open at a time.

Gate 17 permits the one-bit signal from the Logical Unit to be sent to the Index Adjuster. Alternatively the ADJ may be actuated

by signals from the Match Units.

Gate 18 permits the two-bit contents of the Memory Distributor to be entered into the two low-order bit positions of the word portion of the address formed in the Table Address Assembler.

Gates 19, 20, and 21 gate the signals that actuate the Statistical Counter. Gate 19 causes the SCTR to count bytes entered into Stream Unit C. Gate 20 causes the SCTR to count the total number of signals, zeros, and ones, from the one-bit output of the Logical Unit. (See the action of Gate 15 described above.)

Gate 21 causes it to count the number of entries made into the Statistical Accumulator. Only one of these three gates may be open at a time.

6.9 Match Character Recognition

Match Units are provided to stop the stream when an end-of-stream marker passes or to modify the operation when particular characters are presented (especially those indicating uncertain data) or to modify the course of streaming based upon what is fetched in the table-look-up process.

Harvest provides four Match Units, W, X, Y and Z, each attachable to any of four Match Stations within the computer - the output of A (after BM_1), the output of B (after BM_2), the output of the LU, and the output of the Table Extract Unit (after BM_4). Several units may be connected at one position but no single unit can be connected to more than one position. These connections are specified by bits in the Match Unit itself.

Each Match Unit is supplied with an 8-bit match character in the set up shown in Figure 6.7. The unit signals when the bits on the byte bus match those in the match character. This signal, or its inverse, sets an indicator which may initiate stream interruption. Independently, the signal, or its inverse, can be sent to the Index Adjuster.

At each Match Station, the data byte is compared with all the Match Units positioned there. The data go no farther until it is ascertained that there is no match with any of the special characters at this position.

The Match Units monitoring the output of the Table Extract Unit

test every byte presented in the same fashion. If a decision is to be made by means of arbitrary bits entered in a table, these arbitrary bits must form a byte of the same size as that used in reading out the rest of the tabular entry and must form a character which does not appear among the other bytes.

6.10 The Stream Interruption System

Stream interruption is effected in much the same manner as ordinary program interruption (See Chapter 5).

The Selection of the first instruction after interruption does not alter the Instruction Counter; the Instruction Counter does not step to the next value after fetching a stream instruction until the instruction is completely executed (i. e., The stop condition is satisfied).

The first instruction after interrupt may be a single operation, after which the Instruction Counter fetches and continues execution of the stream instruction that was interrupted. Alternatively, the first instruction after interruption may be a BRANCH, which may store and replace the contents of the Instruction Counter.

The twenty-five stream indicator bits are as follows:

6	SCTR overflow
7	NCTR overflow
8	SACC overflow
9	SACC sign change from plus to minus
10	SACC equal to originally set Threshold
11	Match Unit W
12	Match Unit X
13	Match Unit Y
14	Match Unit Z
15	Index Adjuster
16	LU Signal
17	L_1A=End of first indexing level in Stream Unit A
18	L_1B
19	L_1C
20	L_2A =End of second indexing level in Stream Unit A
21	L_2B
22	L_2C
23	L_HA=End of a higher indexing level in Stream Unit A
24	L_HB
25	L_HC
26 - 30	Not assigned

7. System Operation in the Merging Mode

7.1 For relatively short records, sorting by moving the records is an efficient and orderly process. Since the Stream Units provide an automatic flow of data from memory, a large part of the bookkeeping associated with binary merge sorting can be relegated to the SU control mechanism. Several Merge operations are included in the Harvest Computer instruction set. They are described in Section 7.4 below.

It is assumed that records to be merged are in two blocks. It is desired to merge the two blocks of records into one block. The ordering of the records depends upon the relative values of their control fields.

7.2 Record Format

The Merge instructions used in Harvest permit handling a general record format. A control field or subfield may be no longer than 64 bits, and may be offset not more than 225 bits from the beginning of the record. If, however, there is only one control field and it occurs at the beginning of the record, the entire record may be treated as if it were the control field and its length is virtually unlimited.

A typical record is shown in Fig. 7.1e. Here the record consists of 76 bits with a 34 bit control field split into three parts. The first control subfield is 12 bits in length and is offset zero bits from the record beginning. The second subfield is 15 bits long and is offset 57 bits from the start of the record. The third subfield, 7 bits long, is offset 22 bits from the start.

7.3 Problem Parameters

There are six parameters associated with SU merging. They are:

a) L_1, Record Length

b) L_2, Input Block Length

c) L_3, Output Block Length

d) J_1, Control Field Offset

e) J_2, Control Field Length

f) W, Size of Output Working Area

All six parameters express (in binary notation) the number of bits in the respective block or field. When $J_2 = 64$, it is represented by 000000. $L_2 = nL_1$ where n is the number of records in each Input Block, while $L_3 = 2L_2$.

In Harvest, the first two parameters are used to control the Automatic Address Modification mechanisms in Stream Units A and B (see Section 4.5), while W is used to control Stream Unit C. Generally speaking, J_1 and J_2 are used by the Merge instructions to locate the control fields for comparison purposes. Thus the Merge instruction obtains and compares control fields of length J_2 located J_1 bits from the beginning of each of two records. These records enter A and B, one record in each. As the result of the comparison, a record from one of these registers is read into C for automatic storage in memory. The parameter L_1, then, is precisely the Length required to control the first level of address modification in all three SU's; L_2 is the Length required by the second level of A and B, and W controls the second level of C.

7.4 The Merge Instructions

The MERGE UP and MERGE DOWN instructions may be used for merging records for which it is assumed that the control field is the entire record. Six more Merge instructions are provided for more complicated operations. (See Fig. 7.2 for format.) The MERGE OFFSET UP and MERGE OFFSET DOWN instructions are used for merging records having a single control field offset from the beginning of the records. The MERGE SPLIT UP and MERGE SPLIT DOWN instructions are used in conjunction with MERGE BRANCH UP and MERGE BRANCH DOWN for merging records with split control fields. The UP and DOWN instructions produce ascending and descending sequences, respectively.

a) MERGE UP (DOWN)

The two records are compared. If the compared bytes are equal, the common byte goes to C.

If the byte from A is lower (higher) than the byte from B, move the remainder of the record from A without further comparison and reset B.

If the byte from A is higher (lower) than the byte from B, move the remainder of the record from B without further comparison and reset A.

If the records are identical, the bytes are considered as having passed from A to C, and B's controls are reset.

The operation stops only upon interruption, which is normally set up for End-of-Second-Level on A and B.

The index address field is not used.

These instructions assume that the control field is the entire record. J_1 and J_2 are not used. Three situations can arise:

1. the records are identical;

2. control fields are identical, but data fields are not;

3. control fields differ.

In the first case, it is immaterial which record is moved to SU C. In the second case, it is assumed that it is also immaterial since the sort is used to arrange control fields. If this assumption is unsatisfactory, a MERGE OFFSET must be used. In the third case, the decision as to which record to move is made on the basis of the relative sizes of the control fields which begin the record; and only that portion of the records common to both control fields has already been moved at the time the decision is made.

b) MERGE OFFSET UP (DOWN)

The two control fields specified by the offset and field length are compared, and one of the records is then moved to C.

If the control field from A is lower (higher) than or equal to the control field from B, move the record from A and reset B.

If the control field from A is higher (lower) than the control field from B, move the record from B and reset A.

The operation stops only upon interruption, which is normally set up for end of second level on A and B.

The index address field is not used.

c) MERGE SPLIT UP (DOWN)

The two control fields specified by the offset and field length are compared, and if they are not equal, one of the records is moved to C.

If the control field from A is lower (higher) than the control field from B, move the record from A and reset B.

If the control field is higher (lower) than the control field from B, move the record from B and reset A.

If the two control fields are equal, go the next instruction.

When the end of the comparison and subsequent record moving is reached, the operation stops. If the compared fields were unequal, the next instruction is taken from the location specified in the index address field of the instruction. If the compared fields were equal, the next instruction in sequence is executed.

d) MERGE BRANCH UP
(MERGE BRANCH DOWN)

The operation is identical to that of the MERGE OFFSET instructions except that it stops when the end of a first level of indexing is reached in A or B (i.e., after each record is moved to C). The next instruction is always taken from the location specified in the Index Address. This instruction is customarily used to terminate a sequence of MERGE SPLIT instructions. Interruption on the end of second index level is normally used to stop merge instruction loops.

7.5 Use of the Merge Instructions

Four examples follow, illustrating how four types of records may be merged using the instruction types described above. In all cases it is assumed that the ascending control field sequence is desired. (If the descending order were desired, the corresponding DOWN orders would be used.)

a) See Fig. 7.1a. A MERGE UP instruction is used.

The value of L_1 is 30 for all SU's.

b) See Fig. 7.1b. One MERGE OFFSET UP instruction is used. The parameter values are $J_1 = 8$ and $J_2 = 12$.

The value of L_1 is 28 for all SU's.

c) See Fig. 7.1c. A loop of two instructions is used.

The first is MERGE SPLIT UP with parameter values $J_1 = 10$ and $J_2 = 15$. The Index Address is the address of the instruction itself.

The second instruction is MERGE BRANCH UP with parameter values $J_1 = 30$ and $J_2 = 10$. The Index Address is the address of the

MERGE SPLIT UP instruction.

The value of L_1 is 50 for all SU's.

d) See Fig. 7.1d. A loop of three instructions is used.

The first is MERGE SPLIT UP with parameter values $J_1 = 21$ and $J_2 = 11$. The Index Address is the address of the instruction itself.

The second is also a MERGE SPLIT UP with parameter values $J_1 = 53$ and $J_2 = 10$. The Index Address is the address of the first MERGE SPLIT UP instructions.

The last instruction in the loop is MERGE BRANCH UP with parameter values $J_1 = 0$ and $J_2 = 14$. The Index Address is the address of the first MERGE SPLIT UP instruction.

The value of L_1 is 64 for all SU's.

7.6 Control Signals and Runout

After all records of one of two input blocks have passed on to Stream Unit C, it will be necessary to run out the records of the other block. This is accomplished by stream interruption. The End-of-Second-Level control bits must be set to One in SU's A and B. This will cause a program interruption and a transfer of control to one of two stream orders used to run out the block.

These orders may be STREAM LOGIC and have ADJ = 11. (cf. Section 6.3.9). The adjustment operations are Second Level Runout for Unit A or B (according as the End-of-Second-Level signal comes from Unit B or A) and Second Level Reset for Unit B or A.

7.7 Typical Control Sequence

a) MERGE OFFSET causes the control fields of two records to be compared. The control field from A is less than the one from B, so that the record in A is moved to C for storage.

b) This is the last record in the block moving through A. Therefore, the End-of-Second-Level-in-A Indicator Bit comes on, causing an interruption which transfers control to a

c) STREAM LOGIC instruction. The remainder of the block of records moving through B is moved to C for storage and the End-of-Second-Level-in-B interrupt signal is suppressed. The second level control in both A and B are reset.

d) Instruction control returns to the MERGE OFFSET instruction.

8. Sample Problems

8.1 Problem 1:

Given two records each containing an unknown number of five-bit bytes, and each record followed by an end of stream byte, it is desired to add corresponding pairs of bytes together, bit-by-bit modulo 2. The process is to be terminated as soon as the shorter stream runs out. Only every other group of four bytes is to be examined (i.e., the first, third, fifth, ..., group in each stream). Furthermore it is desired to store the resultant stream densely in memory, and also to count the number of zero bytes in the output stream. The end of stream byte is defined to be 11111.

The Setup Words needed and the values of the parameters in each are as follows:

a) Word 1: S = address of first bit in first record

B1 = 11111000

I_1 = 00000101+

b) Word 2: R_1 = 00000 00000 00000 10100+

E = 0, DS = 0, TS = 0

L_1 = 00000 00000 00000 10100+

c) Word 3: A_3 = zero

I_2 = 00000 00000 00001 01000+

d) Word 4: R_2 = zero

E = 0, DS = 1, TS = 0

L_2 = zero

e) Word 5: S = address of first bit in second record

B2 = 11111 000

I_1 = 00000101+

f) Word 6: same as Word 2

g) Word 7: same as Word 3

h) Word 8: same as Word 4

i) Word 9: S = address of location in which first bit in resultant stream is to be stored

B_3 = 11111000

I_1 = 00000101+

j) Word 10: R_1 = zero

E = 0, DS = 0, TS = 0

L_1 = zero

k) Words 11 through 15 not used; may be all zero

l) Word 16: W, X, = 11111000

W connection = 001

X connection = 010

Y, Z connections = 000

All other fields not used; may be all zero

m) Word 17 not used; may be zero

n) Word 18: SCTR = zero, NCTR = zero

o) Word 19: stream portion of Indicator Mask =

01000110000000000000000000

The computer is set up by a RECEIVE order (TRANSMIT might be used instead). The Word Address of this order is the location of Setup Word 1, while the Second Address is the address of S for Stream Unit A. The Index Address is the location of an Index

Word that has 19×2^6 in its Length field (i.e., L = 000000000000000100110000000+), and the rest zero.

The various fields of the Stream instruction are as follows:

a) Data Gating: 01100010000000

b) Control Gating: 01000000

c) Stop: 0011

d) ADJ to Index Controls: zero

e) ADJ Mode: 00

f) TAA Mode: 00

g) Operation Code: STREAM LOGIC NON-ZERO

h) Logical Connective: 0110

i) Modifier: zero

j) LU Signal Invert: 1

k) Programmer's Tag Bit: not used, may be zero

The stream portion of the Indicator Mask has been set to interrupt streaming if the SCTR overflows, or if either end-of-stream byte is recognized. In the former case, control is transferred to whatever routine the programmer has written to take care of this situation (such a program is not discussed here). In the latter case, if either stream runs out, control is transferred to a TRANSMIT order. This instruction has the address of the NCTR as its Second Address and the memory location at which it is desired to store the contents of the NCTR as its Word Address.

Thus the entire program consists of the following words and instructions:

a) Nineteen words of setup information

b) A RECEIVE instruction used to set up the Computer

c) An Index Word that delineates the nineteen word Setup block

d) The STREAM instruction

e) A BRANCH instruction that transfers control to a subroutine that takes care of NCTR overflow

f) Two TRANSMIT instructions that store NCTR contents in memory (one for Match Unit W interrupt, the other for Match Unit X interrupt)

g) The first instruction of the next program (may be in memory directly following second TRANSMIT instruction)

8.2 Problem 2:

Given two records each a thousand four-bit bytes in length, look up an eight-bit signed value in a table using one byte from each record to form the address. Accumulate all values found unless the value is 10101010. Count the number of values accumulated. If at any time the partial sum equals or exceeds 2^{20}, terminate the stream operation and branch to subprogram 1. If the total sum is less than 2^{20}, proceed to subprogram 2. Do not change any parameters already in the registers, except as needed for this problem.

The Setup Words needed for this problem and the values of their parameters are as follows:

a) Word 1: S = address of first bit in first record
B1 = 11110000
I_1 = 00000100+

b) Word 2: R_1 = 00000 00011 11101 00000+
E = 1, DS = 0, TS = 0
L_1 = 00000 00011 11101 00000+

c) Word 5: S = address of first bit in second record
B2 = 1111 0000
I_1 = 0000 0100+

d) Word 6: same as Word 2 except E = 0

e) Word 13: S = 01111
R_1 = 00100+
L_1 = 00100+
I_1 = 00100+

f) Word 14: BA = XXXXXXXXXXXXXXX0000000000
(where the X's represent the address of the first word of the table)

R_2 = 00100+
L_2 = 00100+
I_2 = 00100+

g) Word 15: S = zero
B4 = 11111111
R_1 = 000110+
L_1 = 000110+
I_1 = 000110+

h) Word 16: W = 10101010
W connection = 100
X, Y, Z, connections = 000
All other fields not used; may be all zero

i) Word 17: Accumulator = zero
Threshold = 00000100000000000000000000+
Bit number 27 = 1 (signed values)
Bit number 28 = (normal accumulation)

j) Word 18: SCTR = zero

k) Word 19: Stream portion of Indicator
Mask = 00001100000000000000000000

As in Problem 1, the Computer is set up by RECEIVE order. Here it is necessary to use several such instructions, as Words, 3, 4, 7 through 12 must not be entered into the machine. The first RECEIVE order and its Index Word load Setup Words 1 and 2 the second pair loads Words 5 and 6, the third loads Words 13 through 19.

The Stream instruction fields follow:

a) Data Gating: 00011000000100

b) Control Gating: 00000001

c) Stop: 0101

d) ADJ to Index Controls: zero

e) ADJ Mode: 00

f) TAA Mode: 10

g) Operation Code: any stream operation code except that for STREAM INSERT BYTE

h) Logical Connective: not used; may be zero

i) Modifiers, not used; may be zero

j) LU Signal Invert: not used; may be zero

k) Programmer's Tag Bit: not used; may be zero

The stream portion of the Indicator Mask has been set to interrupt streaming if the Accumulator contents equal or exceed the threshold, or if a special character is recognized. In the former case, control is transferred to the first instruction of subprogram 1. In the latter case control is transferred to an instruction which resets the first level in Unit E. This order calls for the next value to be read from the Table Extract Unit and read into Match Station 4. The previous special character had already been discarded. If this latest value is a normal byte, control reverts to the STREAM instruction. The Stream Stop field is used to terminate streaming on an End-of-1st-Level signal from Stream Unit A. Control is then transferred to the first instruction of subprogram 2. This latter transfer takes place only after any interruptions (caused by the last bytes being read out of Stream Units A and B) have been taken care of.

The words and instructions for this program, then, are as follows:

a) Eleven words of Setup information

b) Three RECEIVE instructions used to set up the computer.

c) Three Index Words that index the RECEIVE instructions through the Setup Words.

d) The table of values, and a RECEIVE instruction with associated Index Word to load the table in memory.

e) The STREAM instruction

f) A BRANCH instruction that transfers control to subroutine 1 upon SACC interrupt.

g) A RESET FIRST LEVEL IN UNIT E instruction that prevents a special character from being accumulated.

h) The first instruction of subprogram 2 (must follow STREAM instruction in memory)

9. Summary

9.1 Operating Speed

The operating speed of an asynchronous machine, such as Harvest, depends on a large number of factors. The problem itself contributes many variables to the speed equation. Other variables are introduced by the form of the machine and by the manner in which the problem is set up in it. Because of this complexity, a set of simple rules for estimating operating speeds must be developed. Later paragraphs in this section provide a preliminary set of such rules to be used until a more accurate set can be established.

As one of the steps in developing this system, a program is being written for the IBM 704 which will accurately time any given program and produce a timing chart showing what portions of the system are in operation at each instant. The program is being written in such a way that it can readily be changed to evaluate the effect of changes to the internal mechanism of the system.

Project Silo calls for determination of the speeds of the memories by August 1, 1957. It appears at this time that the schedule will be met, and that both the 0.5 and 2.0 microsecond memory cycles can be achieved. It is possible that both memories can be increased in size without sacrificing speed and with a reduction in the cost per word of memory. It is also possible that a considerable further saving in total memory cost might be made by assembling a very large memory unit, e.g. one of 65,000 words, with a cycle time somewhat longer than 2.0 microseconds. The possibilities in this direction will be known by July 1, 1957.

The bus mechanism over which the several parts of the system communicate with one another is not fully defined, and exact data on its operation will not be available until after an acceptable system has been bench tested.

The speed of addition and subtraction will be between 0.1 and 0.2 microseconds per 8 bit byte for binary arithmetic. Decimal arithmetic may be slower due to the conversion cycles necessary in the use of a basically binary adder for decimal arithmetic.

The time required to multiply two 32-bit binary numbers will lie between 5 and 15 microseconds. More exact data must wait upon detailing of the circuits involved.

The total time required to execute an instruction will depend upon the kind of memory in which it is stored, the extent to which the instruction is indexed, the location of the data, the complexity of the look ahead feature, the nature of the bus system, and the other simultaneous activities of the computer. Until more is known, the rules below should be used for estimating operating speeds. An endeavor has been made to be conservative in establishing these rules. In particular, it is expected that much of the memory reference time of the arithmetic mode instructions will be overlapped with the arithmetic time. To the extent that this is achieved, the times quoted here will be reduced.

Arithmetic Mode Instructions

1. Allow 1.0 microsecond for each data operand memory reference.

2. Allow 1.0 microsecond to execute a data addition, subtraction, or similar operation regardless of field sizes.

3. Allow 15 microseconds to execute a multiplication regardless of field sizes.

4. Allow 30 microseconds to execute a division regardless of field sizes.

5. Allow 1.0 microsecond for each level of indirect addressing.

6. Allow 1.0 microsecond for each level of indexing above the first.

7. Allow 1.0 microsecond to execute instructions without data operand addresses.

8. Allow 1.5 microsecond per word for word transfer instructions such as TRANSMIT and RECEIVE.

Streaming Mode Instructions

Choose the times as stated for the slowest of the following streaming operations which are concurrent.

1. Allow 0.2 microsecond per byte, or pair of bytes when the pair is operated on simultaneously, for Streaming operations within the Harvest Computer.

2. Allow 0.2 microsecond per byte for each data byte assembled in the TAA in table lookup operations which sequence through four 0.5 microsecond memories.

3. Allow 0.3 microseconds per byte for each data byte assembled in the TAA in random table lookup using four 0.5 microsecond memories.

4. Allow 0.5 microseconds per address assembled in the TAA in table lookup operations which sequence through four 2.0 microsecond memories.

5. Allow 1.0 microsecond per address assembled in the TAA in random table lookup operations using four 2.0 microsecond memories.

Merging Mode Instructions

1. Allow 1.0 microsecond per control field per record plus 1.0 microsecond per word per record for each record passed through the LU into Stream Unit C during Merge instructions.

9.2 Alphabetic List of Instructions

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
ADD	ADD Add operand into Register A.	2	5.4.4
ADM	ADD TO MEMORY Add Register A to operand in memory.	3	5.4.9

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
BAF	BRANCH IF A OFF Test a bit in Register A and branch if zero.	1	5.7.3
BAN	BRANCH IF A ON Test a bit in Register A and branch if one.	1	5.7.3
BEN	BRANCH ENABLE Store Instruction Counter, enable Interrupt, and branch.	1	5.9.2
BIF	BRANCH IF I OFF Test a bit in Indicator register and branch if zero.	1	5.7.3
BIN	BRANCH IF I ON Test a bit in Indicator register and branch if one.	1	5.7.3
BRA	BRANCH Store Instruction Counter and branch.	1	5.7.1
CBN	COUNT AND BRANCH IF NOT ZERO Count up in Index word and branch if residual Length not zero.	2	5.7.4
CBZ	COUNT AND BRANCH IF ZERO Count up in Index word and branch if residual Length zero.	2	5.7.4
CML	CLEAR MEMORY LARGE Clear a large memory block.	2	5.10.2
CMP	COMPARE Compare Register A with operand and set indicators.	2	5.4.7

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
CMS	CLEAR MEMORY SMALL Clear a small memory block.	2	5.10.2
CNC	CONNECT Form specified logical connection of Register A and operand, leaving result in A.	2	5.5.1
CTL	CONTROL Send control signals to external unit.	2	2.2.3
CUM	CUMULATIVE MULTIPLY Multiply Register A by operand, add product to Register C, and return result to A.	16	5.4.12
DIM	DIMINISH Add operand to Register A, and place sum in A if positive, otherwise zero.	2	5.4.8
DIS	DISCONNECT Disconnect external unit immediately.	2	2.2.5
DIV	DIVIDE Divide Register A by operand, placing quotient in A and remainder in C.	31	5.4.13
IDL	IDLE Wait for interruption.	-	5.10.1
INC	INCREMENT Add Increment to Value, subtract Increment from Length, and set indicators.	2	5.3.2
LAC	LOAD AC Transfer Register A to C and load A.	2	5.4.11

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
LNG	LENGTHEN Insert zeros to the right of Register A.	1	5.4.16
LOC	LOCATE Send address to external unit.	2	2.2.4
LOD	LOAD Load operand into Register A.	1	5.4.3
MBD	MERGE BRANCH DOWN Compare control field; move high or equal record and branch.	Note 1	7.4
MBU	MERGE BRANCH UP Compare control field; move low or equal record and branch.	Note 1	7.4
MDN	MERGE DOWN Compare entire record and move high or equal record.	Note 1	7.4
MOD	MERGE OFFSET DOWN Compare control field and then move high or equal record.	Note 1	7.4
MOU	MERGE OFFSET UP Compare control field and then move low or equal record.	Note 1	7.4
MPY	MULTIPLY Multiply Register A by operand and return product to A.	16	5.4.10
MSD	MERGE SPLIT DOWN Compare control field; if unequal move high record and branch, if equal go to next instruction.	Note 1	7.4

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
MSU	MERGE SPLIT UP Compare control field; if unequal move low record and branch, if equal go to next instruction.	Note 1	7.4
MUP	MERGE UP Compare entire record and move low or equal record.	Note 1	7.4
NOP	NO OPERATION Go to next instruction.	1	5.7.2
RCV	RECEIVE Consecutive words go from memory to locations at Second Address.	1.5 per word	5.6.1
RED	READ Start external unit reading per control word.	2	2.2.2
RES	RESTORE Add Value to Length, then set Value to zero.	2	5.3.5
RIN	RESET INTERRUPT Load Interrupt Address Register and turn off Interrupt.	1	5.9.1
RLL	REPLACE L BY L Replace Length at Second Address by Length at Word Address.	2	5.3.4
RLV	REPLACE L BY V Replace Length at Second Address by Value at Word Address.	2	5.3.4
RND	ROUND Remove bits from the right of Register A and round the remainder.	1	5.4.15

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
RVL	REPLACE V BY L Replace Value at Second Address by Length at Word Address.	2	5.3.4
RVV	REPLACE V BY V Replace Value at Second Address by Value at Word Address.	2	5.3.4
SHR	SHORTEN Remove bits from the right of Register A.	1	5.4.14
SIB	STREAM INSERT BYTE Put immediate byte on byte bus from Table Extract Unit.	1	6.5
SLP	STREAM LOGIC-ODD PARITY Form specified logical connection of bytes and signal if number of ones in result is odd.	Note 2	6.4
SLZ	STREAM LOGIC-NON ZERO Form specified logical connection of bytes and signal if result is not zero.	Note 2	6.4
SMC	STREAM MODULAR ADD-CARRY Add bytes using specified modulus and signal if carry.	Note 2	6.4
SMR	STREAM MODULAR REVERSE SUBTRACT Form difference of byte B minus byte A using specified modulus and signal if result is zero.	Note 2	6.4

Abbr.	Operation	Time Estimate (Microseconds)	Refer to Section
SMS	STREAM MODULAR SUBTRACT Form difference of byte A minus byte B using specified modulus and signal if result is zero.	Note 2	6.4
SMZ	STREAM MODULAR ADD-ZERO Add bytes using specified modulus and signal if result is zero.	Note 2	6.4
SSL	STREAM SELECT Compare bytes, giving either byte or the difference as result, as specified.	Note 2	6.4
STC	STORE C Store Register C in memory.	3	5.4.6
STO	STORE Store Register A in memory.	3	5.4.5
TMT	TRANSMIT Store consecutive words from Second Address in memory.	1.5 per word	5.6.1
WRI	WRITE Start external unit writing per control word.	2	2.2.2

Note 1: Allow 1.0 microsecond per control field per record plus 1.0 microsecond per word per record for each record passed through the LU into Stream Unit C during Merge instructions.

Note 2: Choose the times given in Section 9.1 under Streaming Mode Instructions, using the time of the slowest concurrent operation.

9.3 The Basic Harvest System

A summary of the functional units that comprise the basic Harvest System follows:

a) One Harvest Computer

b) Eight units of 8,192 words of 2.0 microsecond memory

c) Four units of 512 words of 0.5 microsecond memory

d) Sixteen single word registers with 0.1 microsecond access time

e) One Exchange

f) One 1000 card per minute card reader

g) One 155 card per minute card punch

h) One 500 line per minute printer

i) Six high performance tape mechanisms

j) One high performance disk memory

k) Four tape units handling IBM 727 tapes

l) One console with manual inquiry station

9.4 Stream Instruction Fields

Bits	Use
0-13	Data Gating
14-21	Control Gating

Constraints on Data and Control Gating are such that any instruction may specify only one open gate in each of the following groups:

2, 9
3, 8
4, 12
6, 10
7, 11
11, 13
14, 15, 16
19, 20, 21

Gates 0, 1, 5, 17 and 18 may each be opened or closed independent of other gating.

Bits	Use
22-25	Stop Codes
26-43	Index Adjuster

A	B	C	D	E
26-29	30-33	34-37	38-40	41-43
0000	No adjustment		000 No adjustment	No adjustment
1000	Repeat byte			
1001	Advance first level		001 Advance first level	Advance first level
1010	Reset first level		010 Reset first level	Reset first level
1011	Runout first level		011 Runout first level	Runout first level
1100	Advance second level		100 Advance second level	Not used
1101	Reset second level		101 Reset second level	Not used
1110	Runout second level		110 Runout second level	Not used
1111	Runout high level		111 Repeat byte	Repeat byte

Bits	Use
44-45	Adjuster Mode
46-47	TAA Mode
48-53	Op Code
54-57	Connectives

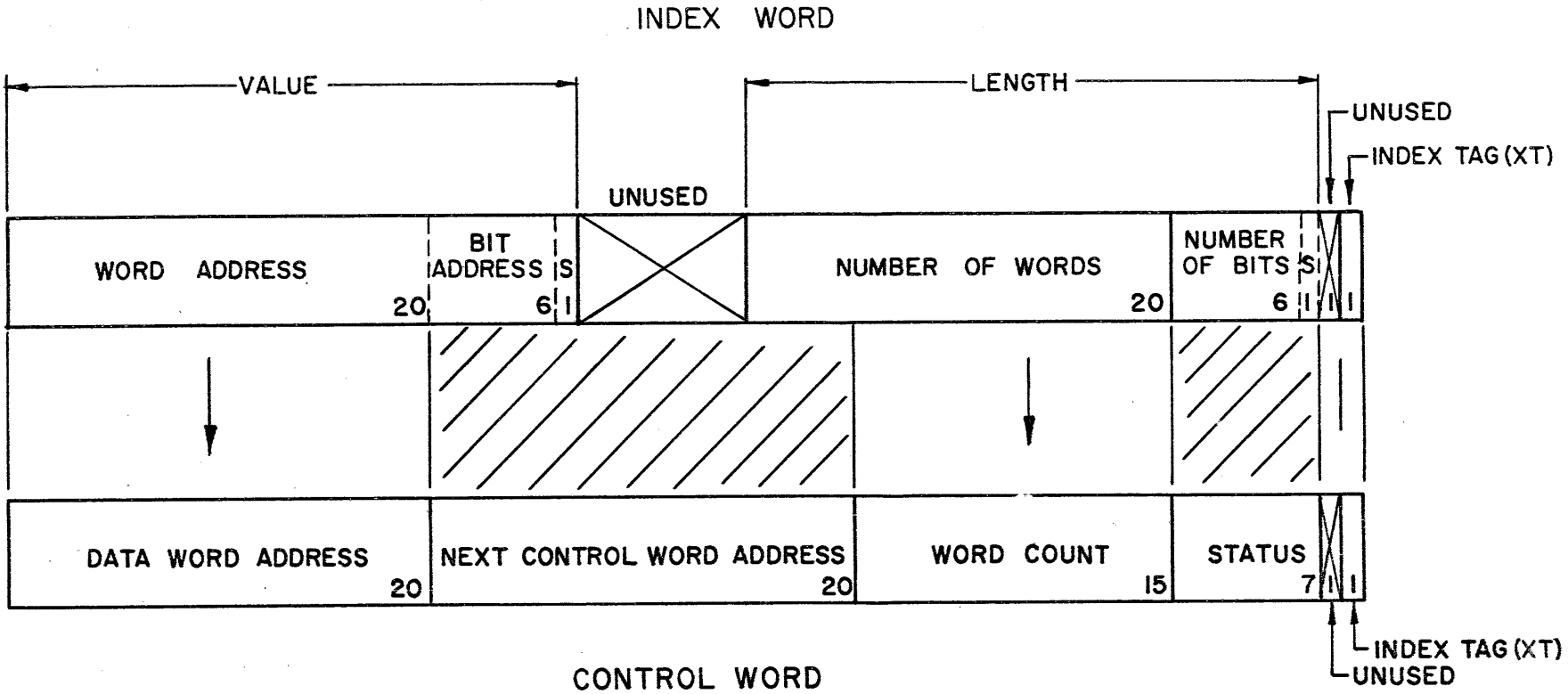
58-61 Modifiers for Stream Select Instruction

	Byte Output	Signal Output one if
0000	Max. (A, B)	$A > B$
0001	Max. (A, B)	$A = B$
0010	Max. (A, B)	$A < B$
0100	Min. (A, B)	$A > B$
0101	Min. (A, B)	$A = B$
0110	Min. (A, B)	$A < B$
1000	A if A = B, otherwise 0	$A = B$
1001	A if $A \neq B$, otherwise 0	$A \neq B$
1010	A - B if $A > B$, otherwise 0	$A \geq B$
1011	B - A if $B > A$, otherwise 0	$B \leq A$

or 54-61 Modulus or Immediate Byte

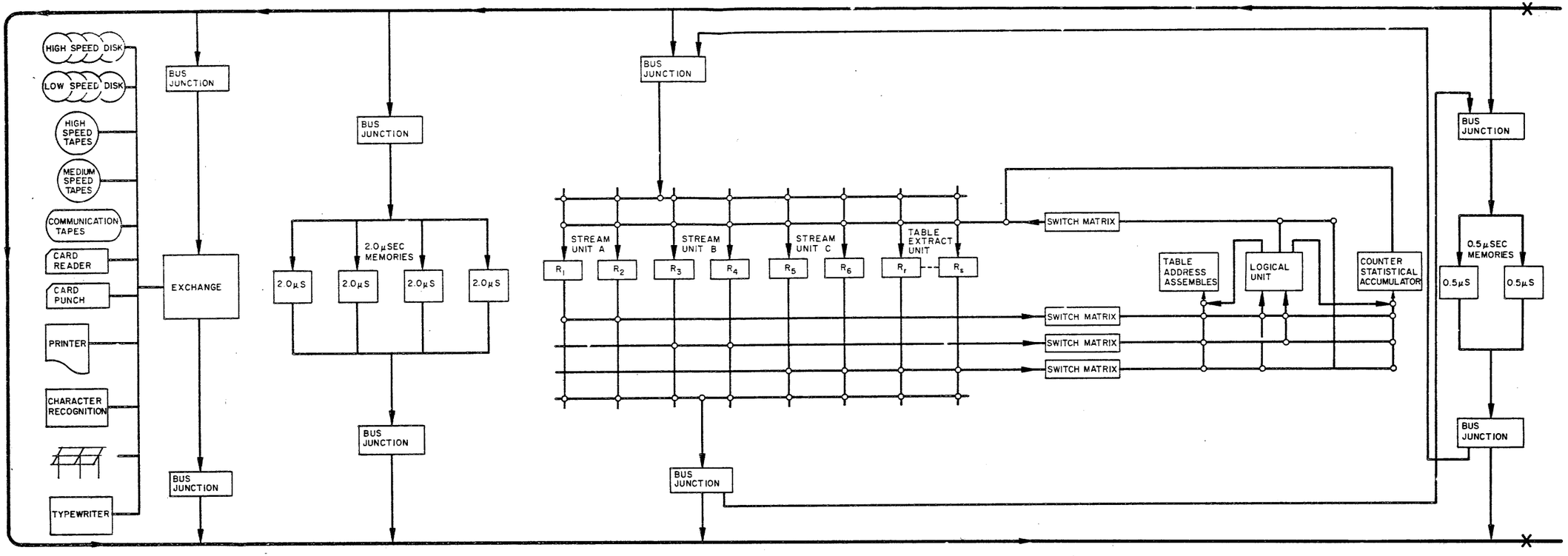
62 LU Invert

63 Instruction Tag

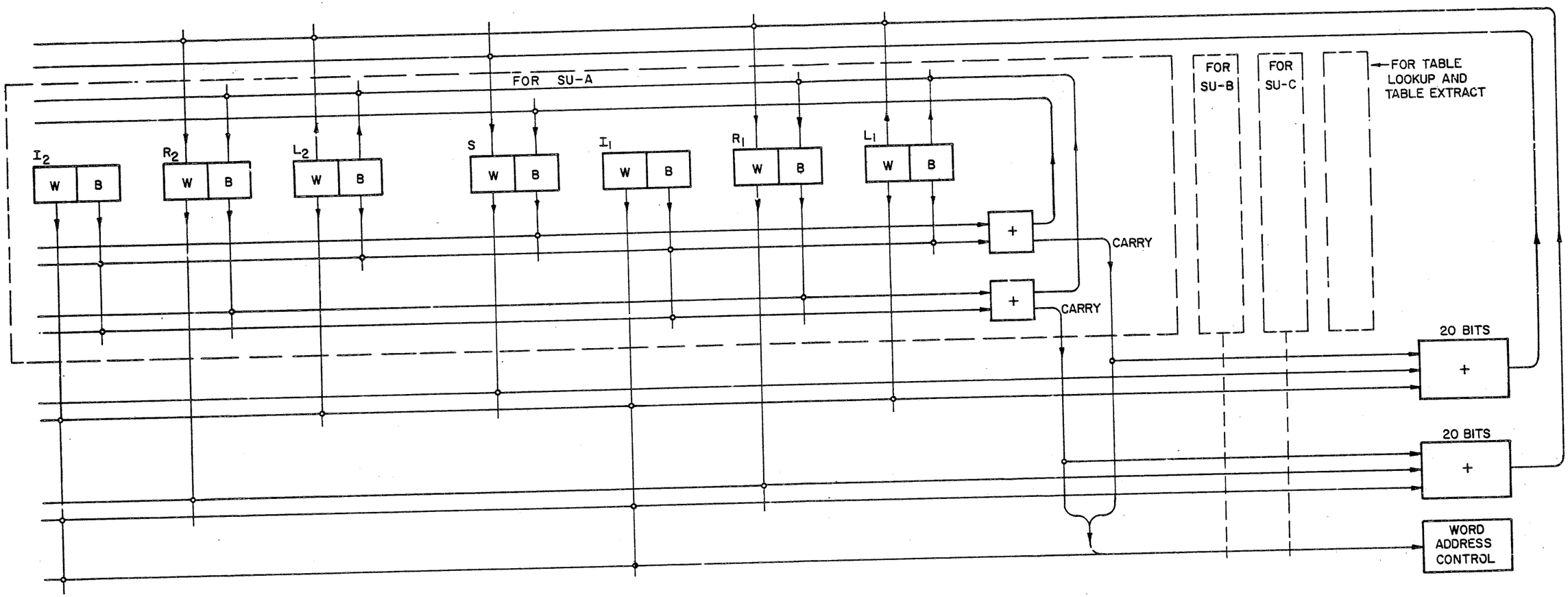


RELATION BETWEEN INDEX AND CONTROL WORDS

FIGURE 2.2

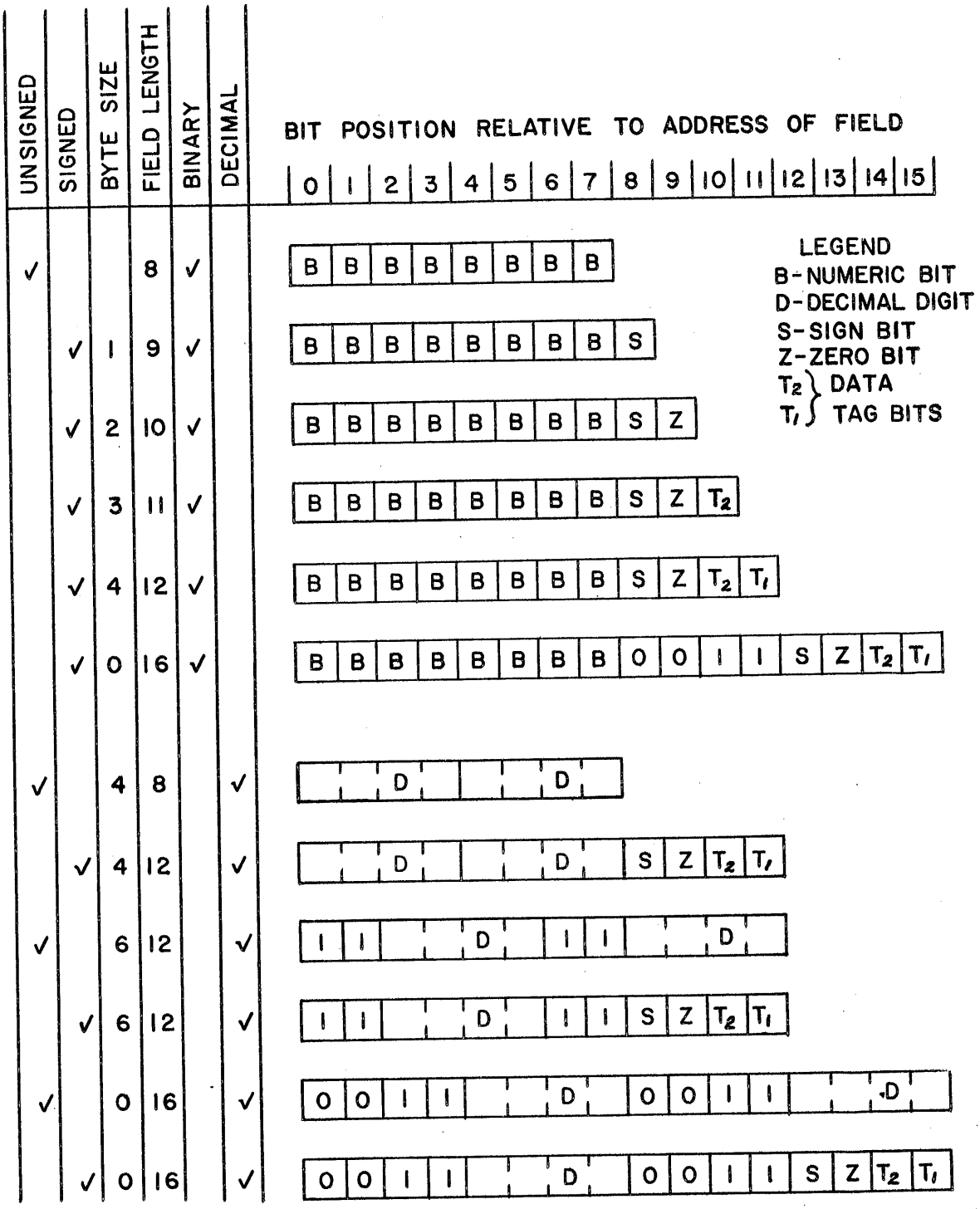


HARVEST SYSTEM – Figure 4.1



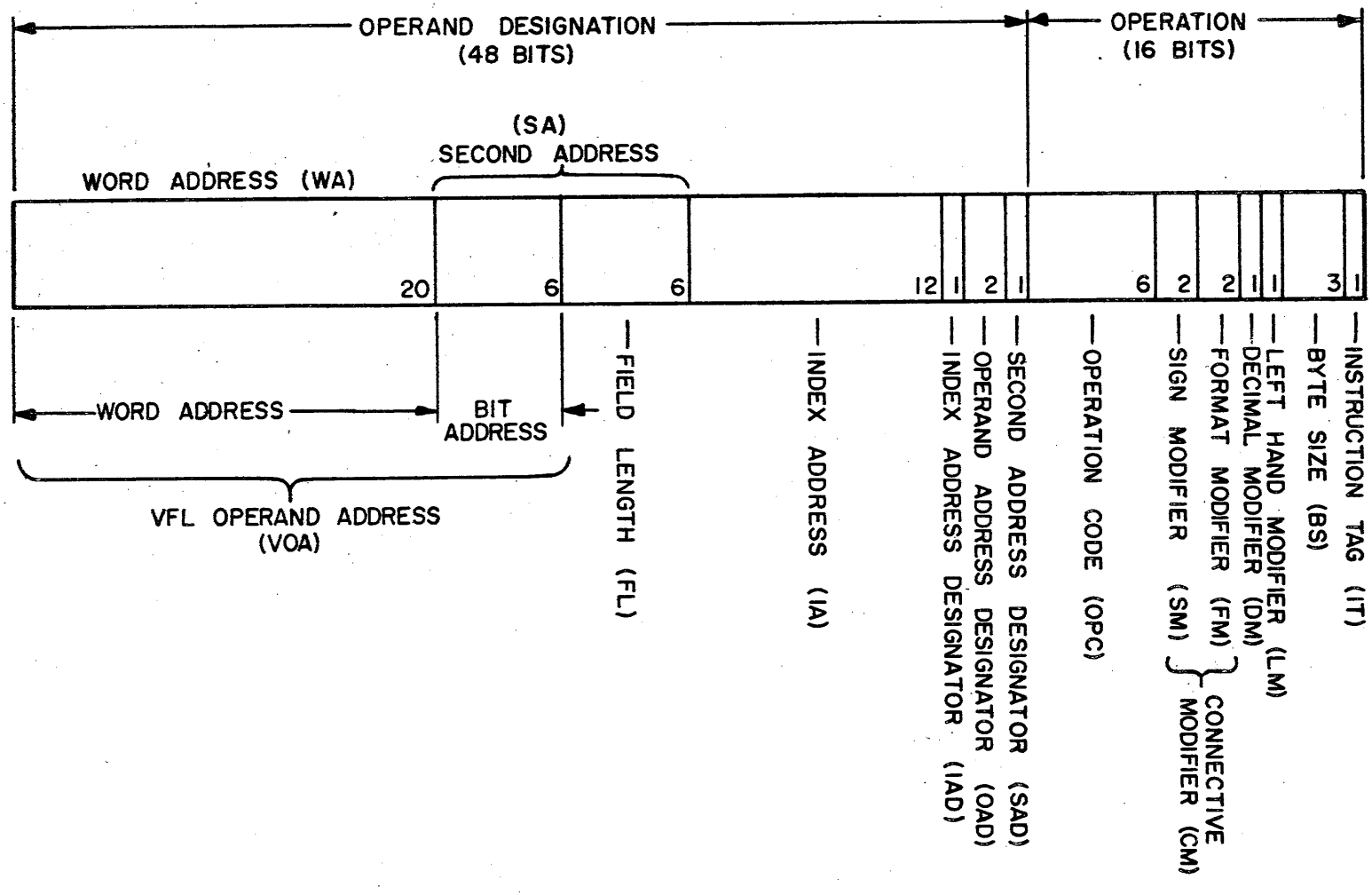
STREAM INDEXING

FIGURE 4.7



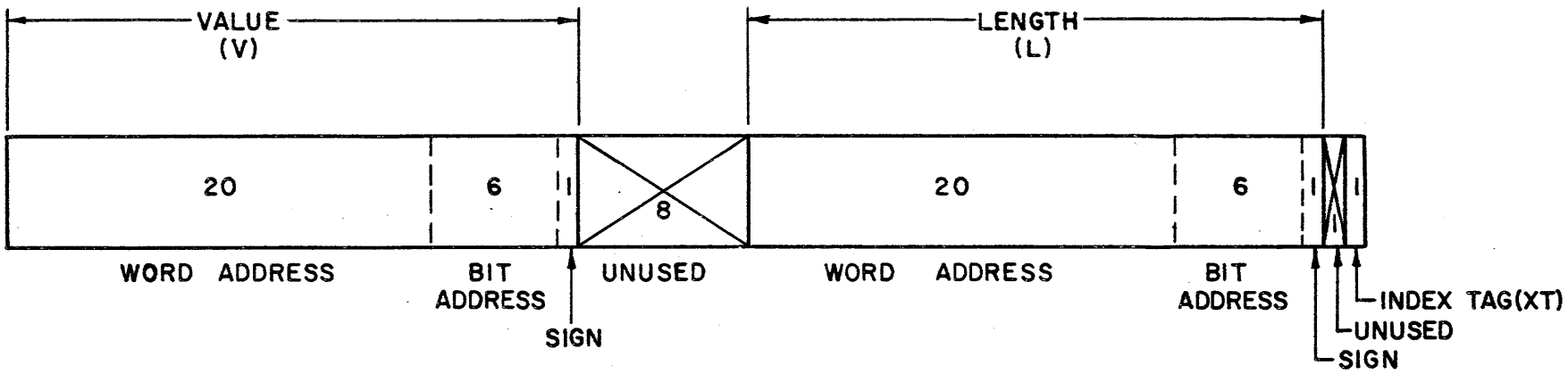
EXAMPLES OF VFL NUMERIC FORMATS

FIGURE 5.2



INSTRUCTION FORMAT
(ARITHMETIC MODE)

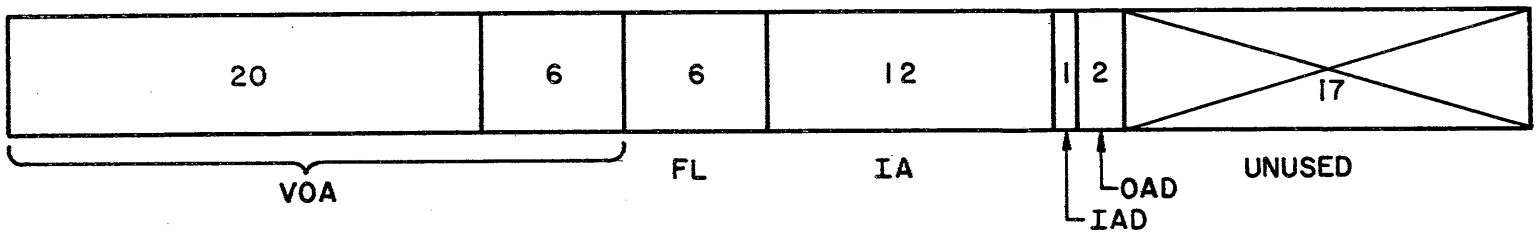
FIGURE 5.3



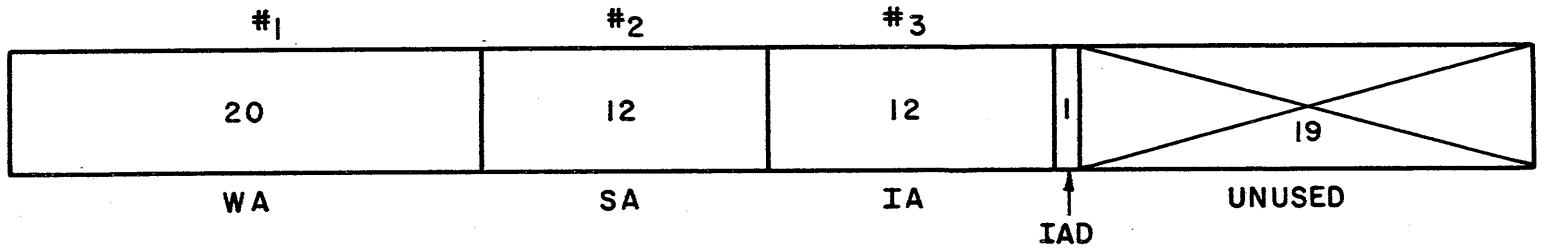
INDEX WORD FORMAT

FIGURE 5.4

INDIRECT ADDRESS WORD



MULTIPLE INDEX ADDRESS WORD



FORMATS DERIVED FROM INSTRUCTION WORD

FIGURE 5.5

(CM) CONNECTIVE CODE	LOGICAL FUNCTION	RESULT WHEN A B = 1 1	A B = 1 0	A B = 0 1	A B = 0 0
00	0	0	0	0	0
01	$\bar{A}\cdot\bar{B}$	0	0	0	1
02	$\bar{A}\cdot B$	0	0	1	0
03	$\bar{A}$	0	0	1	1
04	$A\cdot\bar{B}$	0	1	0	0
05	$\bar{B}$	0	1	0	1
06	$A \veebar B$	0	1	1	0
07	$\bar{A} \vee \bar{B}$	0	1	1	1
08	$A\cdot B$	1	0	0	0
09	$A \equiv B$	1	0	0	1
10	B	1	0	1	0
11	$\bar{A} \vee B$	1	0	1	1
12	A	1	1	0	0
13	$A \vee \bar{B}$	1	1	0	1
14	$A \vee B$	1	1	1	0
15	1	1	1	1	1

TABLE OF CONNECTIVES

FIGURE 5.6

CHARACTER		BIT POSITION							
		0	1	2	3	4	5	6	7
(Blank)	b	0	0	0	0	0	0	0	0
	.	0	0	0	0	0	0	0	1
	¤	0	0	0	0	0	0	1	0
	&	0	0	0	0	0	0	1	1
	$	0	0	0	0	0	1	0	0
	*	0	0	0	0	0	1	0	1
	–	0	0	0	0	0	1	1	0
	/	0	0	0	0	0	1	1	1
	,	0	0	0	0	1	0	0	0
	%	0	0	0	0	1	0	0	1
	#	0	0	0	0	1	0	1	0
	@	0	0	0	0	1	0	1	1
		0	0	0	0	1	1	0	0
		0	0	0	0	1	1	0	1
		0	0	0	0	1	1	1	0
		0	0	0	0	1	1	1	1
	A	0	0	0	1	0	0	0	0
	B	0	0	0	1	0	0	0	1
	C	0	0	0	1	0	0	1	0
	D	0	0	0	1	0	0	1	1
	E	0	0	0	1	0	1	0	0
	F	0	0	0	1	0	1	0	1
	G	0	0	0	1	0	1	1	0
	H	0	0	0	1	0	1	1	1
	I	0	0	0	1	1	0	0	0
	J	0	0	0	1	1	0	0	1
	K	0	0	0	1	1	0	1	0
	L	0	0	0	1	1	0	1	1
	M	0	0	0	1	1	1	0	0
	N	0	0	0	1	1	1	0	1
	O	0	0	0	1	1	1	1	0
	P	0	0	0	1	1	1	1	1

CHARACTER		BIT POSITION							
		0	1	2	3	4	5	6	7
	Q	0	0	1	0	0	0	0	0
	R	0	0	1	0	0	0	0	1
	S	0	0	1	0	0	0	1	0
	T	0	0	1	0	0	0	1	1
	U	0	0	1	0	0	1	0	0
	V	0	0	1	0	0	1	0	1
	W	0	0	1	0	0	1	1	0
	X	0	0	1	0	0	1	1	1
	Y	0	0	1	0	1	0	0	0
	Z	0	0	1	0	1	0	0	1
		0	0	1	0	1	0	1	0
		0	0	1	0	1	0	1	1
		0	0	1	0	1	1	0	0
		0	0	1	0	1	1	0	1
		0	0	1	0	1	1	1	0
		0	0	1	0	1	1	1	1
	0	0	0	1	1	0	0	0	0
	1	0	0	1	1	0	0	0	1
	2	0	0	1	1	0	0	1	0
	3	0	0	1	1	0	0	1	1
	4	0	0	1	1	0	1	0	0
	5	0	0	1	1	0	1	0	1
	6	0	0	1	1	0	1	1	0
	7	0	0	1	1	0	1	1	1
	8	0	0	1	1	1	0	0	0
	9	0	0	1	1	1	0	0	1
		0	0	1	1	1	0	1	0
		0	0	1	1	1	0	1	1
		0	0	1	1	1	1	0	0
		0	0	1	1	1	1	0	1
		0	0	1	1	1	1	1	0
		0	0	1	1	1	1	1	1

PREFERRED ALPHANUMERIC CODE

FIGURE 5.8

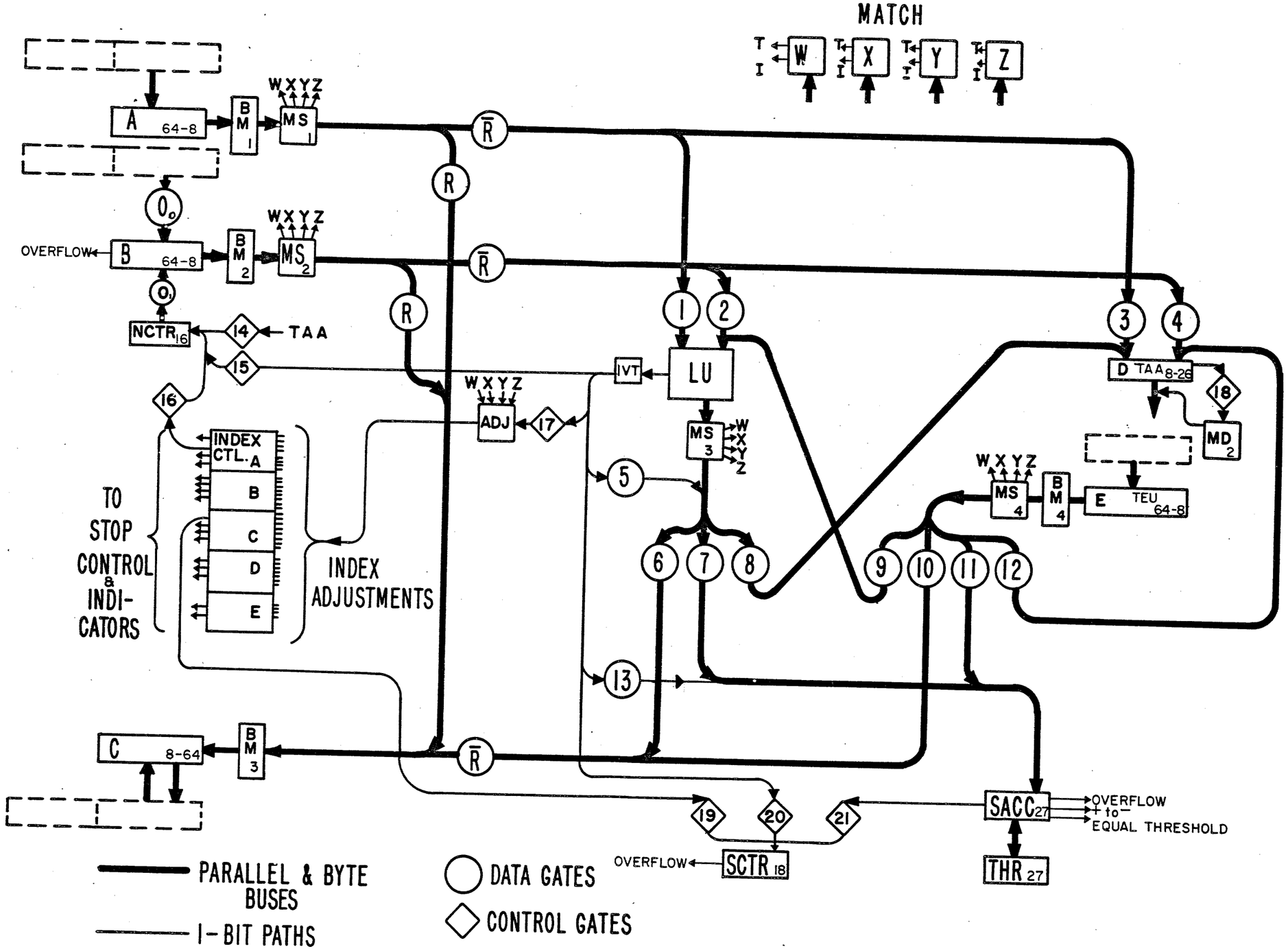
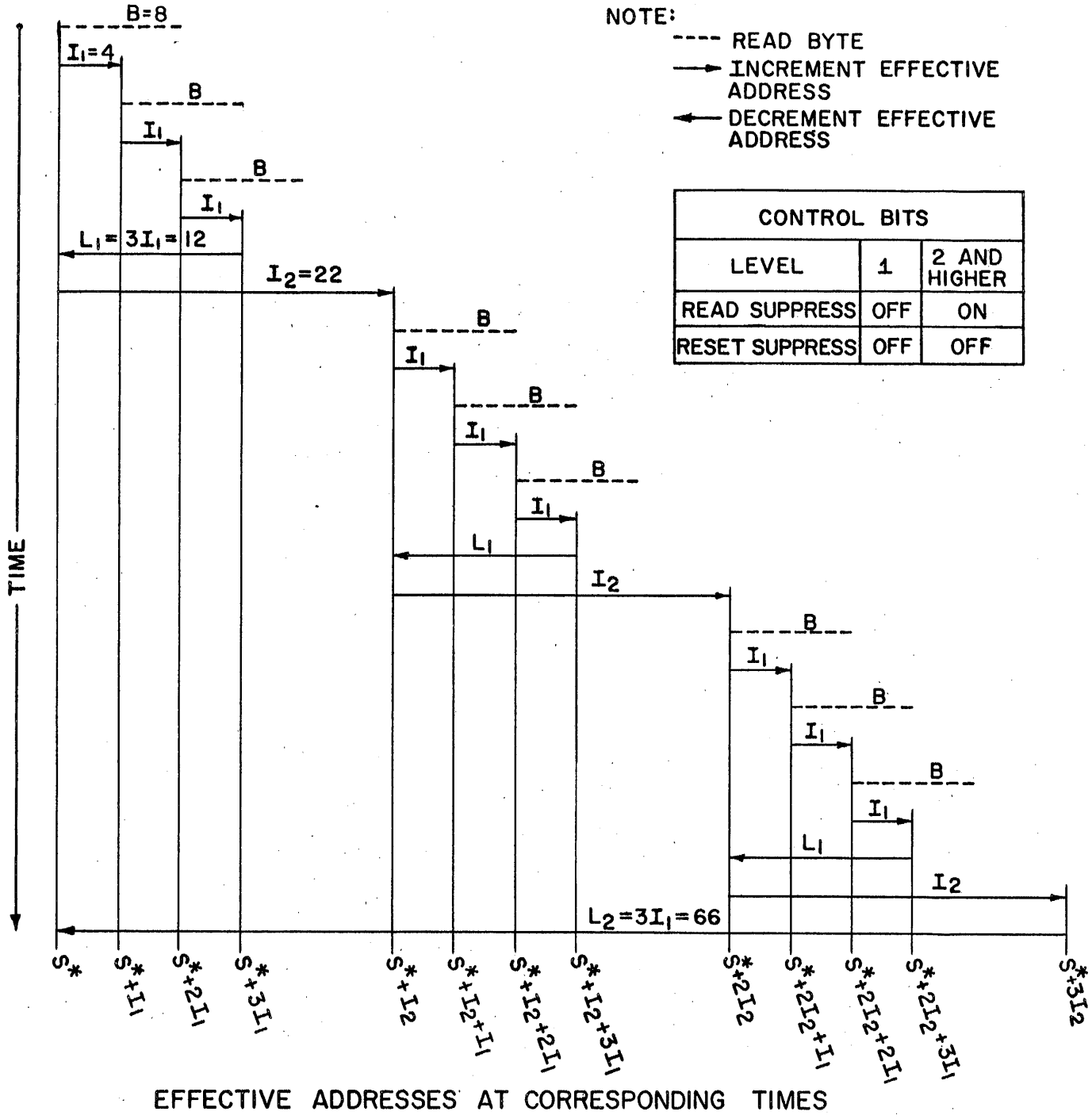
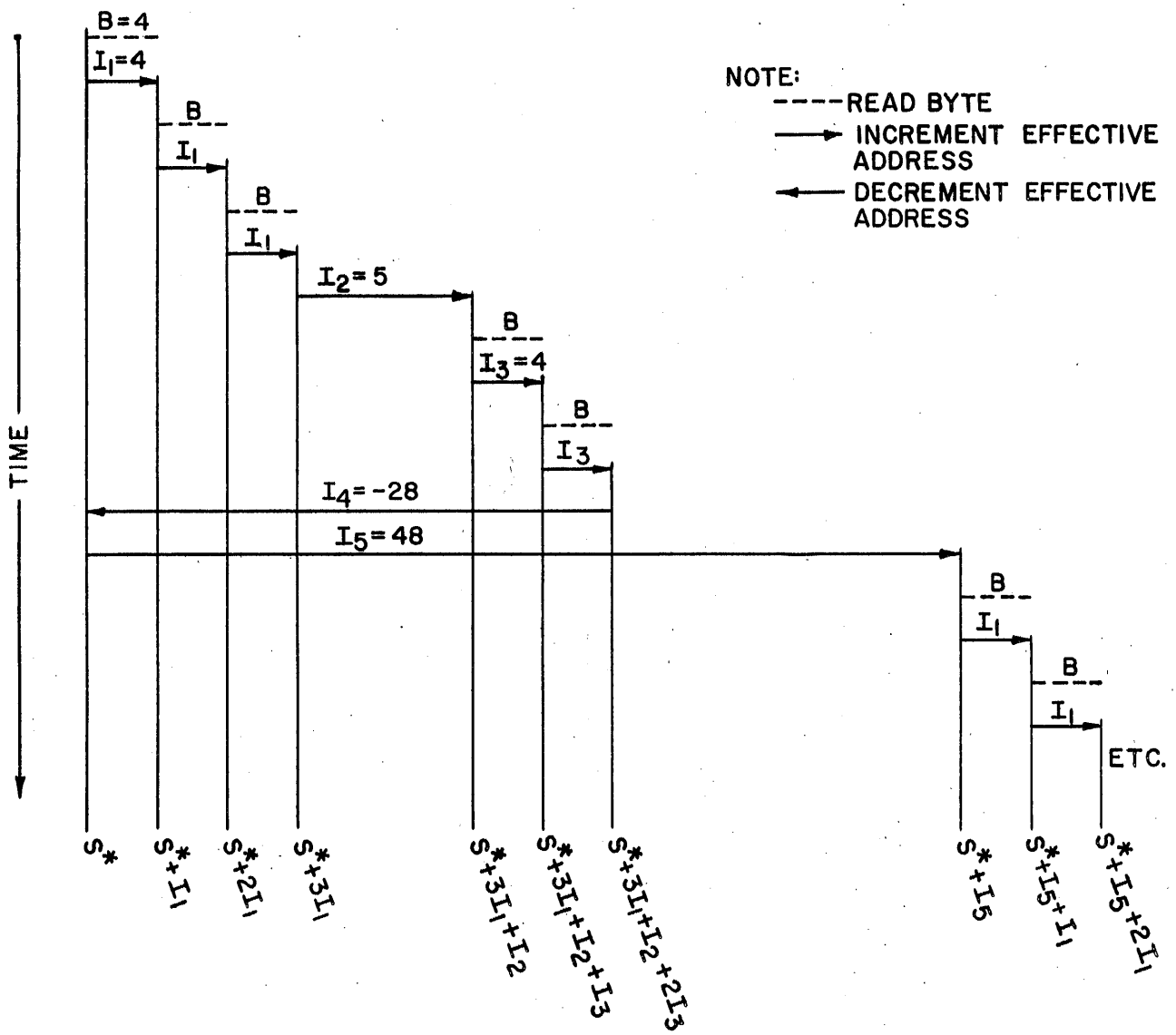


Figure 6.1 DATA and CONTROL PATHS in the STREAMING MODE



EXAMPLE 2, FIRST AND SECOND LEVELS OF ADDRESS MODIFICATION

FIGURE 6.2

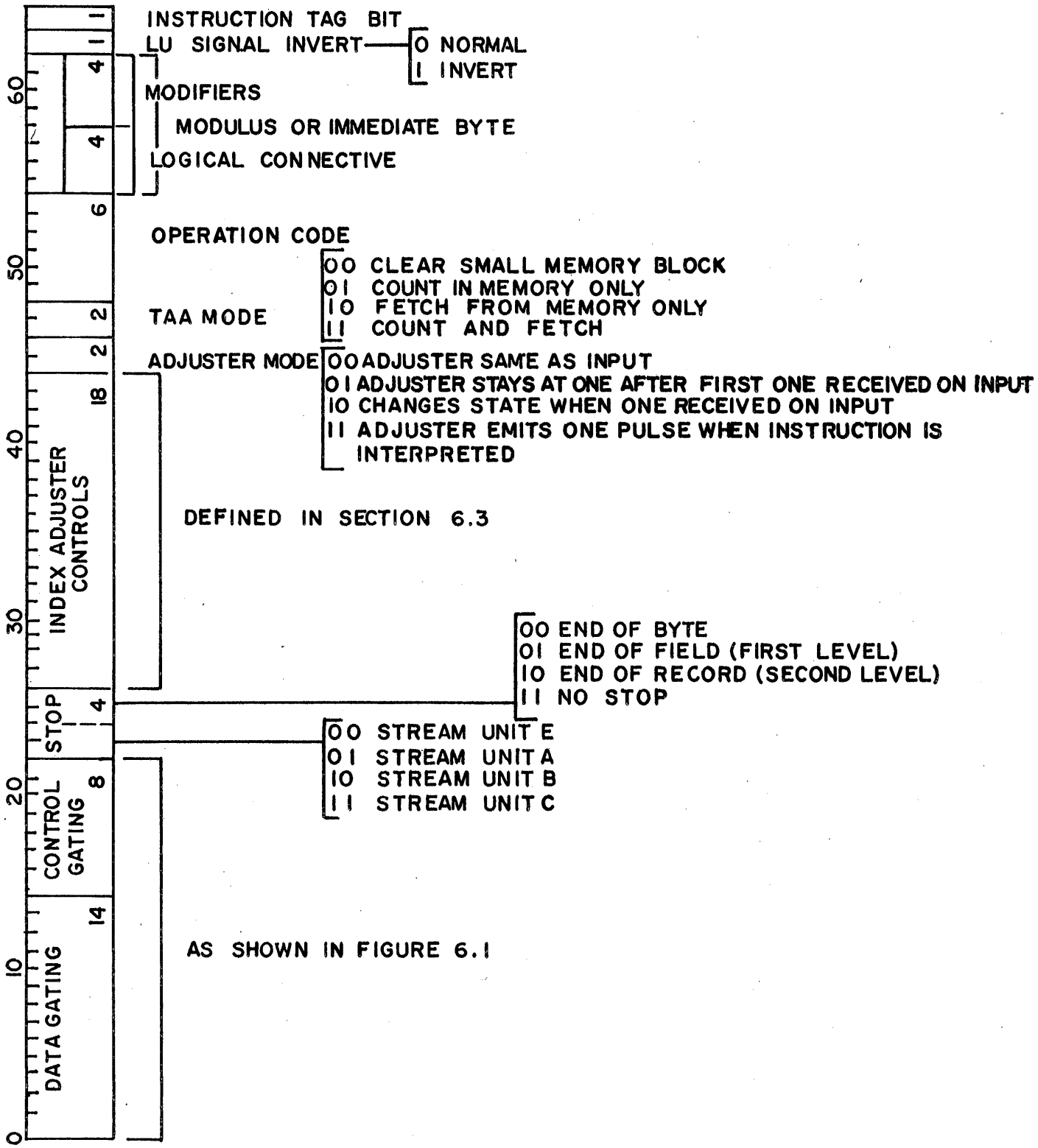


EFFECTIVE ADDRESSES AT CORRESPONDING TIMES

CONTROL BITS					
LEVEL	1	2	3	4	5
READ SUPPRESS	OFF	ON	OFF	ON	ON
RESET SUPPRESS	ON	ON	ON	ON	OFF

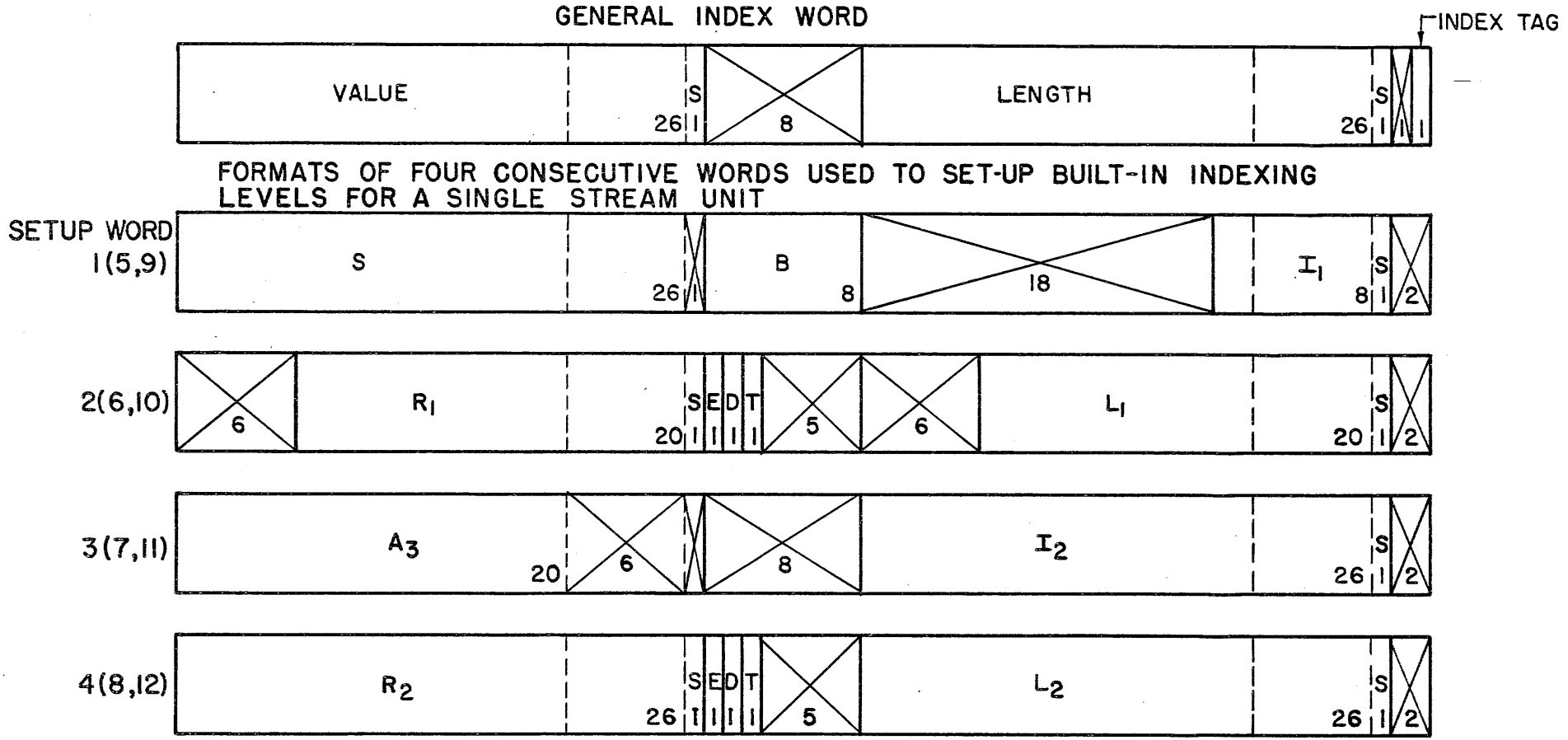
EXAMPLE 3, FIVE LEVELS OF ADDRESS MODIFICATION

FIGURE 6.3



STREAMING MODE INSTRUCTION FORMAT

FIGURE 6.4



STREAM UNIT SETUP FORMAT

FIGURE 6.5

TABLE ADDRESS ASSEMBLER SETUP FORMAT

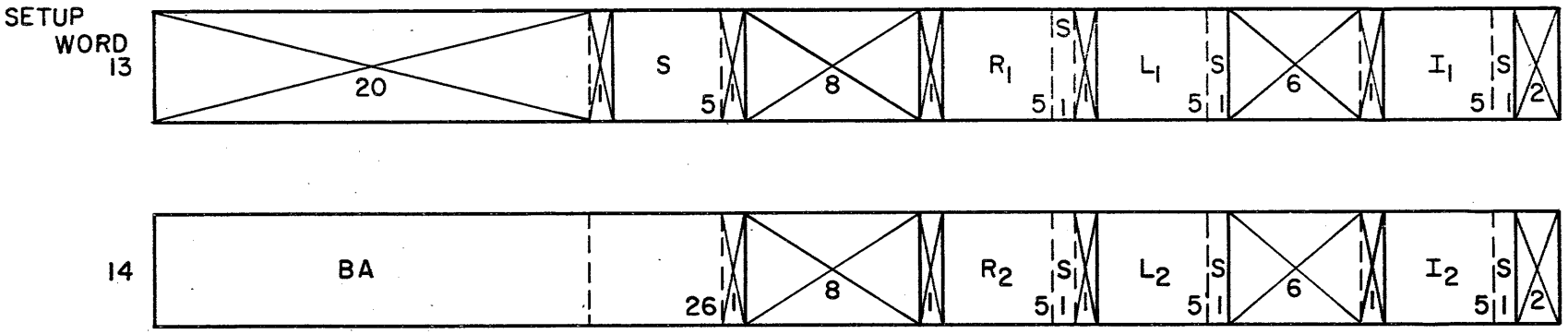
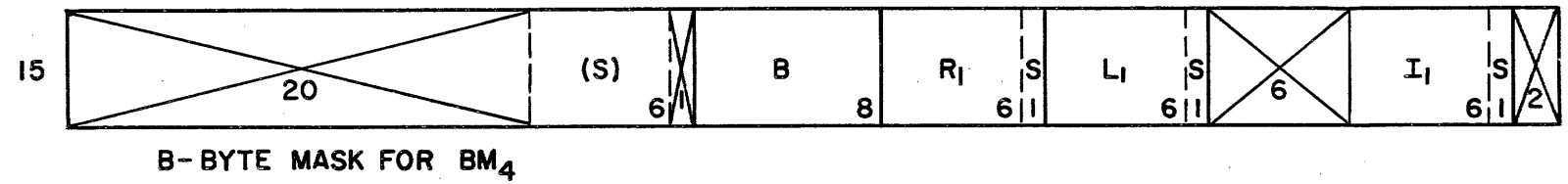


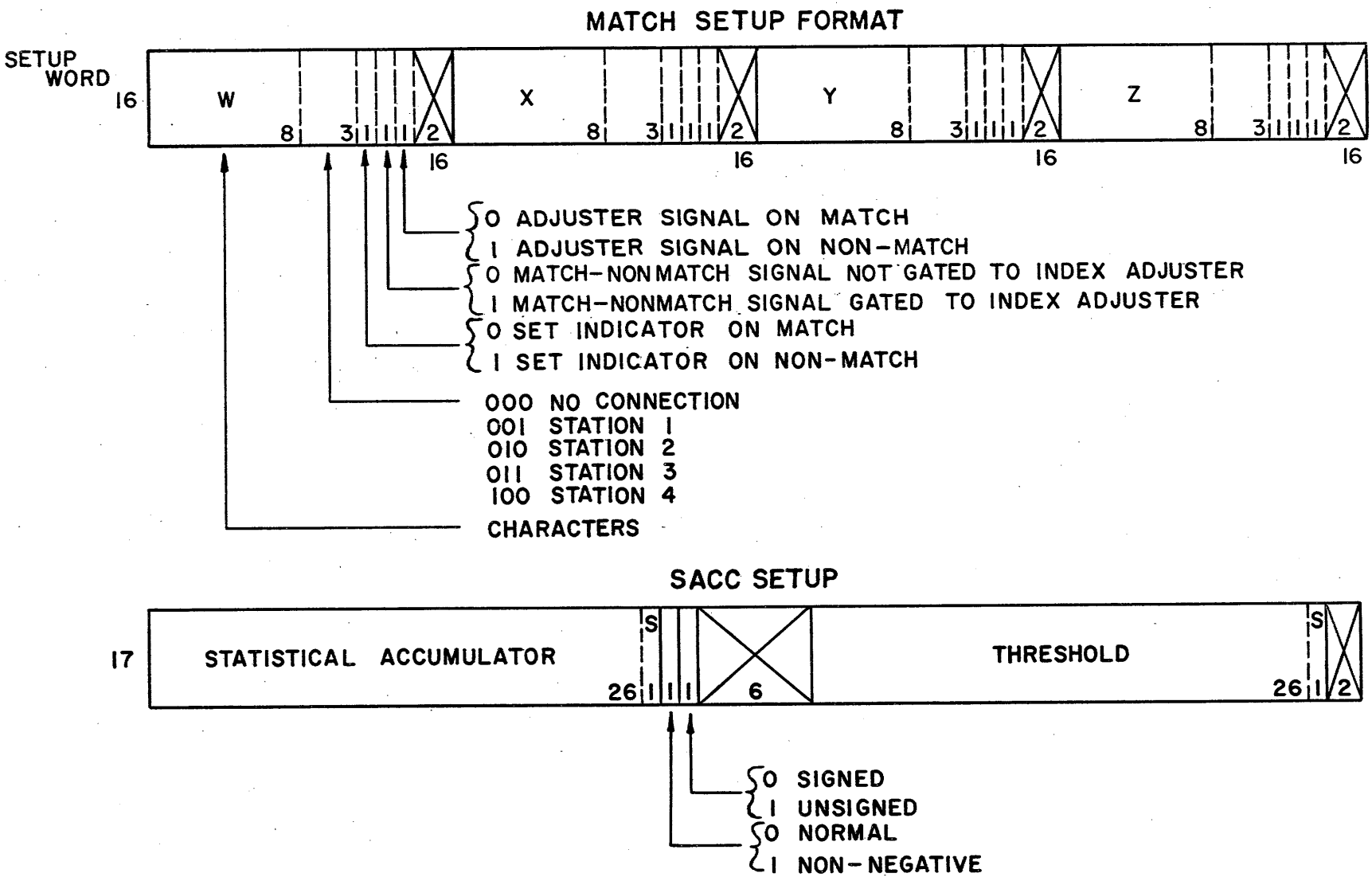
TABLE EXTRACT UNIT SETUP FORMAT



B-BYTE MASK FOR BM_4

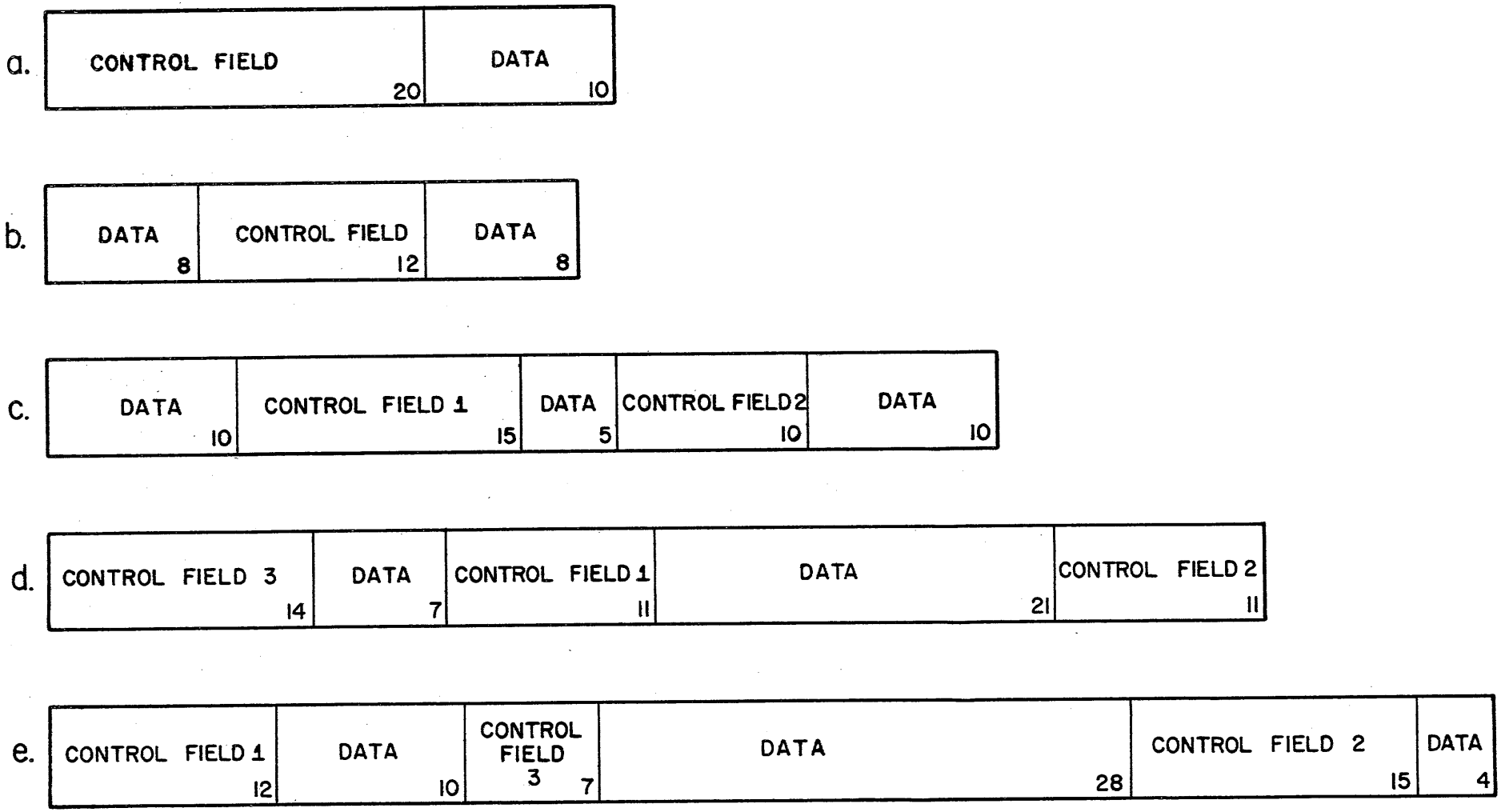
SETUP FORMATS

FIGURE 6.6



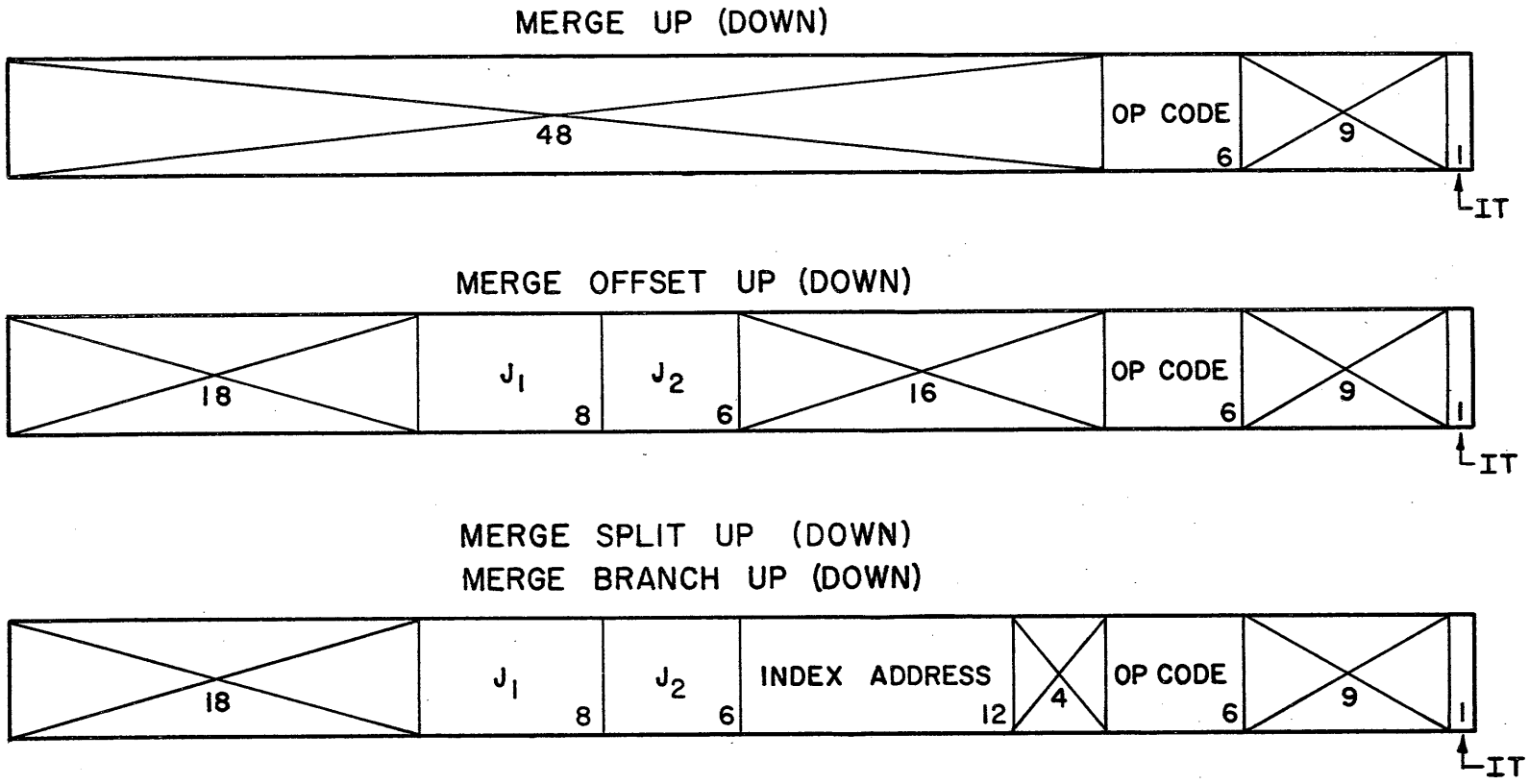
SETUP FORMATS

FIGURE 6.7



SOME TYPICAL RECORDS FOR MERGING

FIGURE 7.1



MERGE INSTRUCTION FORMATS

FIGURE 7.2

PRELIMINARY MANUAL - HARVEST SYSTEM

ERRATA

May 15, 1957

p. 1.5: 1.4, par 1, line 5
For "regular instruction mode" read "arithmetic mode"

p. 2.1: par 4, line 3
"simultnaeously" should be "simultaneously"

p. 2.13: 2.3.3, par 2, line 3
"thst" should be "this"

p. 4.3: 4.1.2, par 2, line 7
For "among the 16 information bits" read "among the first 15 of the 16 information bits"

p. 4.5: par 1, line 8
For "17 dotted intersections" read "17 intersections"

par 1, line 9
For "dotted intersections" read "intersections"

par 1, line 12
For "among the 17" read "among the 15 of the 16 information bits"

par 1, line 16
For "of byte" read "of the byte"

p. 4.6: par above 4.3, line 1
For "the bit portion ... cell size." read "the bit portion of the address (a 6-bit field) is Or-ed to a 6-bit number that has a value one less than the table entry cell size."

p. 4.9: 4.5, par 1, line 5
For "Address register with the" read "Address register, S, with the"

p. 4.10: Step (e) should read
"(e) Is Reset Suppress bit on?

If NO, replace S by S - L (reset S) and proceed to Step (f)

If YES, proceed to Step (f) (don't reset S)

(f) Replace X by X + 1 and proceed to new Level X, Step (a)"

p. 4.15: line 9
For "first" read "leftmost"

Fig 4.1: For "Table Address Assembles" read "Table Address Assembler"

Fig 4.7: For "W" read "WA" and for "B" read "BA".

p. 5.1: 5.1, par 2, line 6
For "fields" read "field".

p. 5.9: par 3, line 2
For "SLU" read "LU".

p. 5.12: Item 2
For "modifying" read "modify"

p. 5.15: 5.3.4, par 1, line 1
For "The REPLACE INSTRUCTIONS CAUSE EITHER THE V or L . . ." read "the REPLACE instructions cause either the V or L. . ."

p. 5.41: 5.10.4, par 3 should read
"For this purpose, a preferred alphanumeric code has been drawn up in Figure 5.8. The six significant bits are shown in the rightmost six positions of an 8-bit byte. The leftmost two bits are zero. When a 6-bit byte is used, the two bits on the left are dropped."

Fig 5.3: SM & FM fields combine to be "CONNECTIVE MODIFIER (CM)" field.

For "OPERATION (48 BITS)" read "OPERATION (16 BITS)".

p. 6.11: par 6.3.8
Item 2. should read "effective address is decremented by $L_1 - R_1$ and R_1 is reset to L_1"

p. 6.12: Item 5 should read "then effective address is decremented by L_2 - R_2 and R_2 is reset to L_2"

p. 6.13: Bits 38 - 40
The code 111 should be defined as Operation (0) as in Bits 41 - 43.

There is no Operation (7) for Mechanism D.

p. 6.16: par above 6.5, lines 1 and 3
For "B_1" and "B_2" read "BM_1" and "BM_2"

par above 6.5, line 8
For "in the setup" read "in the STREAM instruction"

6.5 Between 00 and 10 codes, insert the following:
"01 Count or Or in the addressed memory cell (without transferring contents to TEU)"

p. 6.20: 6.7, par 3, line 4
For "Gate 0" read "Gate 0_1"

Fig 6.1: In "MATCH" at top, use "ADJ" instead of "T"

Fig 6.2: For "S*" read "S".

For "$L_2 = 3I_1 = 66$" read "$L_2 = 3I_2 = 66$".

Fig 6.3: For "$I_2 = 5$" read "$I_2 = 8$".

For "S*" read "S".

Fig 6.4: For "01 END OF FIELD (FIRST LEVEL)" read "01 END OF FIRST LEVEL".

For "10 END OF RECORD (SECOND LEVEL)" read "10 END OF SECOND LEVEL".

Fig 6.5: Setup word 1(5, 9)
For "B" read "BM".

Fig 6. 5: (cont.)
Setup words 2(6, 10) and 4(8, 12)
For "D" read "D_S", and for "T" read "T_S".

Fig 6. 5, 6. 6, 6. 7:
Use "$\pm$" instead of "S" for sign bit (19 places)

Fig 6. 7: Setup word 17
Interchange functions of the two control bits

p. 7. 4: Item c), par 3, line 1
For "field is higher" read "field from A is higher".

p. 7. 6: Item d), par 3, lines 2 and 3
For "J_2 = 10" read "J_2 = 11".

p. 8. 1: Item a), line 2
For "B1" read "BM_1"

Item e), line 2
For "B2" read "BM_2"

p. 8. 2: Item i), line 3
For "B_3" read "BM_3"

p. 8. 3: 2 lines below Item k)
For "SCTR" read "NCTR"

p. 8. 4: 8. 2, par 1, lines 5 and 7
For "2^{20}" read "2^{15}".

Item a), line 2
For "B1" read "BM_1".

Item c), line 2
For "B2" read "BM_2".

p. 8. 5: Item f), line 1
Should be "BA = XXXXX XXXXX XXXXX 0000 0000 000"

p. 8. 5: (cont.)
Item g), line 2
For "B4" read "BM_4"

Item i), line 2
Should be "Threshold = 00000 00000 1 00000 00000 00000+"

Item i), line 3
Should be "Bit number 27 = 0(signed values)"

Item i), line 4
Should be "Bit number 28 = 0(normal accumulation)"

p. 6. 6: 6. 3. 2, item b)
For "Byte Size" read "Byte Size, BS"

6. 3. 2, item d)
For "number of iterations to be" read "Number of Iterations, $N_{(k)}$, to be".

p. 6. 7: par 2, line 2
For "B = 6" read "BS = 6".

par 3, line 1
For "B = 8" read "BS = 8"

par 4, line 1
For "B = 4" read "BS = 4"

Figs 6. 2 and 6. 3
For "B" read "BS".