N. Rochester

This is the best available form of the 8106 manual. We should have it entirely rewritten within six weeks time. Regards.

J. W. Franklin
Dept 251
Bldg 965
Systems Planning

THE IBM 8106 DATA PROCESSING SYSTEM

(Preliminary Operating Manual)

IBM CONFIDENTIAL

Product Development Laboratory
International Business Machines Corporation

Data Systems Division
Poughkeepsie, New York

Chapter 1

THE IBM 8106 DATA PROCESSING SYSTEM

Contents

Printed in U.S.A.

OK

Chapter 1

THE IBM 8106 DATA PROCESSING SYSTEM

1.1 INTRODUCTION

The IBM 8106 Data Processing System is designed to satisfy a wide range of applications. These applications may require a disk- or tape-oriented system; a system with great data-manipulation facility or a system for technical computation; a system for conventional daily operation or one with around-the-clock availability; a system for batch or in-line processing; a system for scheduled or short-notice processing; a system operating virtually unattended or requiring extensive man-machine communication.

To meet these diverse and often conflicting requirements at an efficient cost-to-performance ratio, two principles have been used in the systems design.

The first principle is modularity of systems components. A universal interconnection is provided between the central processor and input-output to provide maximum freedom in the selection of input-output units. The choice between a disk- or tape-oriented system, or any intermediate solution, no longer affects the equipment within the central processor. Similar remarks apply to other system alternatives. The modular principle also applies to the size and characteristics of storage units, and furthermore, extends to the number and characteristics of central processing units which may be part of a system. Thus, two or more central processors may communicate with each other and with input-output units in such a manner that continuous systems availability is insured. The central processors within a system need not be identical. For instance, increased technical computing performance may be achieved by incorporating one or more IBM 8112 processors within the system.

The second principle applied in the 8106 system design is integration of a large number of basic features which enable the system to satisfy different application requirements. It is expected that each of these features will be used to some degree in all installations. However, the particular emphasis which a given feature receives will determine the operating capability of the system.

FEATURES OF THE SYSTEM

Direct and indirect addressing
Indexing of floating-point operands
Data specification by control words
Chained control words
Program and data relocation
Multiple simultaneous input-output operation
Independent instruction streams for input-output channels
Multiplexed processors
Rapid program switching
Address protection
Program interrupt
Maskable indicators
Interval timer
Time clock
Instruction count
Program and data monitoring
Variable precision floating point
Noisy mode for floating-point significance test
Floating-point extremum handling
Binary and decimal fixed-point arithmetic
Signed and unsigned arithmetic
Arithmetic with single or multiple operands
Processing on fields or streams of data
Data flags
Code translation
Logical connectives
Compressed four-bit numeric data
Extended eight-bit alphanumeric characters
Processing till match, clash or count
Decisions on bits or bytes
Error scan
Automated fault location

The individual cost of each of these features has been reduced greatly by using a highly integrated design. Registers are shared by input-output control words, data control words, and operands. Input-output channel controls are shared by interrupt functions. Buses serve for serial, parallel, input-output, diagnostic, and initializing information flow. A fast-responding priority circuit permits simultaneous data flow through several input-output channels, flexibility in operand designation, immediate response to system alerts, and accurate fault location.

1.2 SYSTEM ORGANIZATION

A basic system is composed of a central processor, core storage, and input-output. Information moves between input-output devices and core storage via channels which are part of the central-processing unit. The central-processing unit further consists of registers, arithmetic devices, and control circuits necessary for performing operations upon data taken from core storage. The central processor is controlled by a succession of instructions which themselves come from core storage. Registers and controls are shared in performing the tasks of channel operations, data manipulation, and instruction modification.

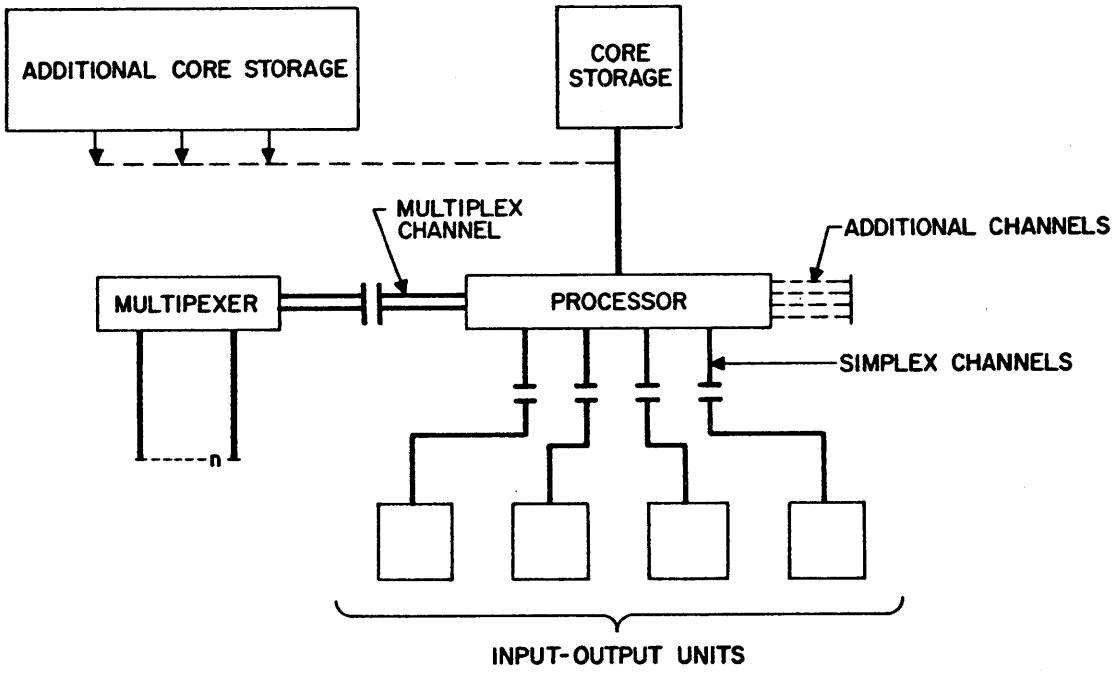


Figure 1.1 - System organization

1.2.1 Storage Units

The computing system uses core storage units with a read-write cycle time of two microseconds. A word consists of 64 information bits and eight nonaddressable redundancy bits. The redundancy bits are provided for error detection by the central processor and the input-output units.

The address space in instructions provides for processing directly on any operation any of 65,536 (2^{16}) word locations numbered 0 to 65,535 consecutively. Some of the low-order addresses are reserved for special purposes.

1.2.1.1 Core Storage

Storage units contain 4096, 8192, or 16,384 words. A system may contain several units of 16,384 words operating as one logical unit. Only one storage reference is in process at the same time. These units also have a two-microsecond read-write cycle time.

1.2.1.2 Shared Core Storage

The computing system permits the use of additional core-storage units which have two storage bus connections, but which are otherwise identical to the core-storage units mentioned above. They permit communication between two central-processing units. These processors may operate simultaneously, each with their own input-output units, and if desired, each with additional core storage connected to its own storage bus. The shared core-storage unit provides a link between the two storage bus systems. Messages can be exchanged between processors by placing them in the shared core-storage unit. In particular, the systems can be organized in such a fashion that, in the case of routine maintenance or malfunction of one processor, the other processor can keep the system in satisfactory operating condition. The use of several shared storage units permit more than two processors to be part of a system.

Shared core storage units contain 4096 or 16,384 words.

1.2.1.3 Core Storage With Fast Channels

Two types of input-output channels are provided, the types differing only in individual and in total rate of transmission through the channels. When high-performance channels are provided in the central-processing unit, one core-storage unit with fast channels must be employed in the system. This type of core-storage unit is identical with the unit mentioned under Storage Units, except that equipment is provided to permit assembling or disassembling of input-output data at a high data rate. The presence of the fast channels also serves to reduce the number of storage cycles required by a channel while transmitting data.

Core-storage units with fast channels contain 4096 or 16,384 words exclusive of the fast-channel equipment. A maximum of eight channels can be provided.

1.2.2 Input-Output

Input to the system passes from the input devices to storage through the channels of the central processor. These channels assemble complete 64-bit words from the flow of input information and store the assembled words in core storage locations. The central processor specifies the starting location and the number of input words to be read. While the central processor proceeds with computation, the channel completes the input operation and signals the central processor when transmission is finished.

The same channels operate similarly for output, fetching core-storage words and disassembling them for the output devices independent of computations in the central processor. External storage devices, such as tapes and disks, may be operated via a channel as if they were input and output units.

The basic central-processing unit has five channels which can operate independently. Four of these basic channels -- the simplex channels -- can be replaced by up to eight fast channels when a core-storage unit with fast channels is provided in the system. The simplex channels can accommodate the data rate of an IBM 729IV tape unit. A maximum of four units with this data rate can run simultaneously. The fast channels can run units with a data rate up to 300 kc. The fifth basic channel -- the multiplex channel -- can function either as a single channel or as a connection to a multiplexing device. When a multiplexing device is connected, the channel is logically and operationally equivalent to as many as 112 channels. These channels share a major portion of the channel equipment, and they accommodate the aggregate data rates required by communication equipment.

A wide variety of input-output units can be attached to the channels -- card readers, printers, magnetic tape units, disk units, communication equipment, displays, and operators' consoles. Some control units permit only one input-output device to be attached to a channel. Others, such as the magnetic tape-control unit or the disk-control unit, are designed to allow several units to be attached to a single channel. When this is done, only one of these units can be operated at a time. Control units designed for attachment to the multiplex channel accommodate many units simultaneously.

1.2.3 Central Processor

The central processor performs the basic tasks of instruction sequencing, operand addressing, and arithmetic or logical processing. A diagram of the central processor is shown in Figure 1.2 . Operations are specified one at a time by instructions taken from storage. The operands specified by the instruction are also obtained from storage. The arithmetic or logical process specified by the instruction is performed subsequently and the result returned to storage. The internal registers and data paths of the central processor permit parallel access to storage. Hence, the fetching of instructions and data, and the storing of results are performed in parallel. The high speed of storage permits rapid parallel operation. The actual processing operation, which requires great flexibility in the selection of successive digits or characters to be processed and in the various modes of processing, is performed serially: a digit or a character at a time. The combination of parallel data transmission and serial data processing allows for speed and flexibility in over-all operation.

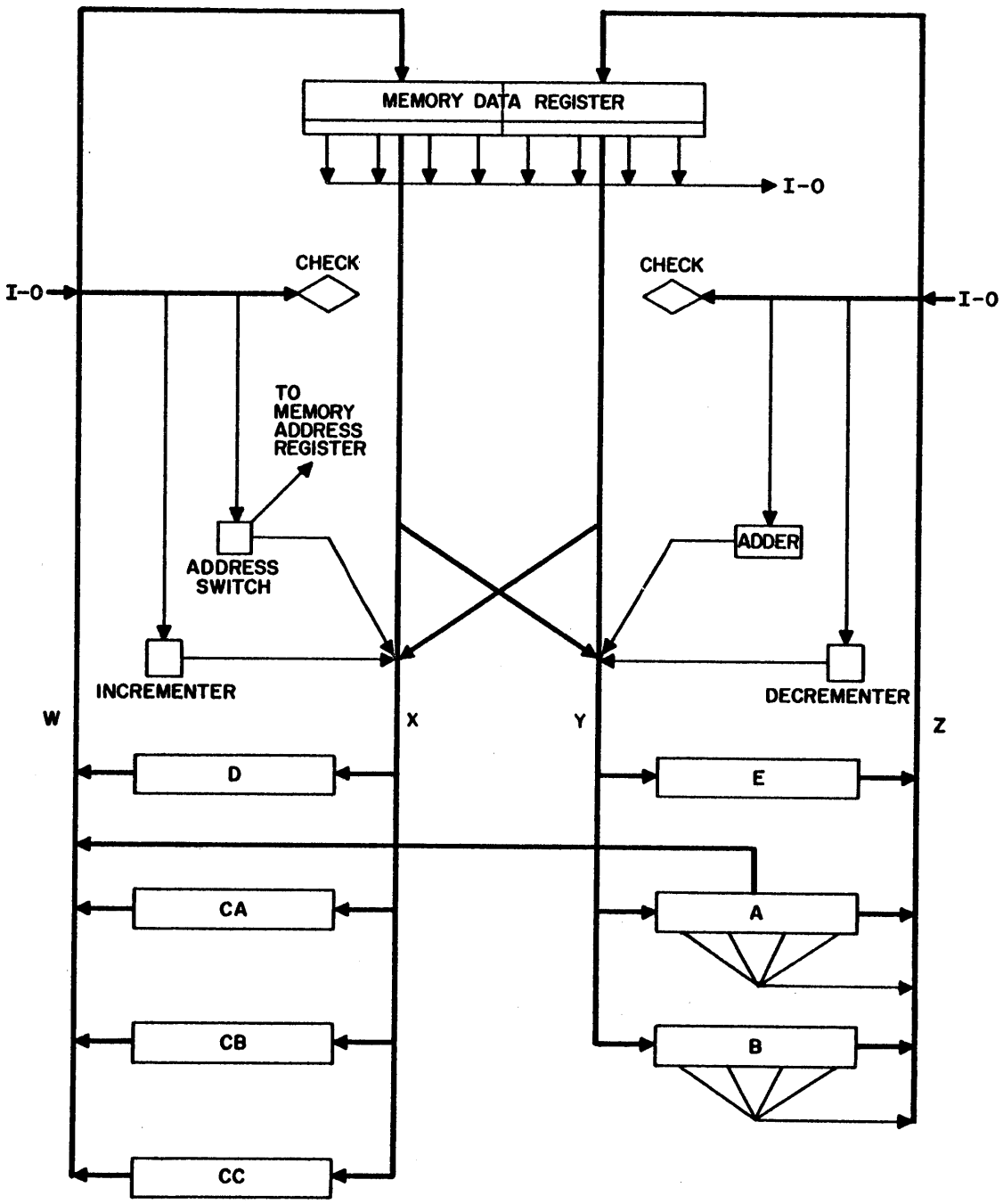


Figure 1.2 - Data paths

1.3 SYSTEM OPERATION

1.3.1 Instruction Sequencing

Instructions have a length of one, two, or three 32-bit half-words with each instruction half-word containing an address. Therefore, one-, two-, or three-address instructions are available. Instructions of any length can be intermixed without regard to word boundaries.

Instructions are taken in succession as specified by an instruction address. Alteration of the succession of instructions is possible by branching operations which can be controlled by several conditions. Automatic interruption of the normal sequence of instructions will occur under control of certain system alerts. Part of these alerts are caused by the program in execution and are identified by indicator bits. The programmer can selectively permit or prevent interruption caused by the program alerts by means of a set of mask bits. The condition bits, the indicator bits, and the mask bits identify the condition under which the program is currently executed. These bits form part of a program control word which contains the essential information which defines the state of the processor as it executes a given program. The instruction address, mentioned above, is also part of the program control word. The program control word further contains mode bits, a prefix address, and an instruction count.

The prefix address is a reference address for the program being executed. Some information belonging to this program, such as the floating-point accumulator, is stored relative to the prefix address. The prefix address also serves as a boundary address in the address-protecting procedure.

The instruction count is a tool for program diagnosis. The count is reduced by one for each instruction executed until zero is reached. When zero is reached, a program interruption is initiated.

The program control word permits rapid program switching. All information vital to the current program can be preserved by storing the program control word. When the program is to be resumed, the program control word need only be loaded into internal working registers to restore the processing unit to the state belonging to this program. Since the prefix address is part of the program control word, a change in prefix address can be part of a switch in program control words. This avoids dumping and loading of vital registers as part of the change from one program to another.

A program interruption is executed as a switch between the current program and the auxiliary program which belongs to the interruption. As an interruption occurs, the processor is placed in the disabled state and further interruptions are delayed. Any subsequent interruptions are stacked in storage and will occur in turn when the processor is placed again in the enabled state.

Interruptions may be caused by input-output units, by various program monitoring functions and by exceptional arithmetic results. The input-output interruptions indicate normal or special termination of a channel operation. The monitoring functions include an interval timer, the instruction count, and checks on valid data, addresses and operation codes. The exceptional arithmetic results include various types of overflows and are individually maskable by the programmer.

1.3.2 Operand Addressing

Operands are specified by the address parts of the instruction being executed. In floating-point and a few related operations, an accumulator is used and only one additional operand is specified. All other operations specify all operand and result locations, and do not use an accumulator. Fixed-point arithmetic and logical-processing operations, therefore, are two- or three-address operations.

The address specified by the instruction may further be modified by indirect addressing, and in the case of floating-point operands, by indexing.

Several operand formats can be used. The floating-point format occupies a full word. The operands of arithmetic or logical processes occupy fields up to 64 bits in length. Fields of different length may be assigned to adjacent locations in core storage, even if this means that a field lies partly in one word and partly in the next. Each field may be addressed directly by specifying its start and end in the instruction. The computer selects the words required and alters only the desired information. Since the field length is explicitly stated in each instruction rather than being implied by the data, there is no restriction on the coding of variable field-length data. In logical operations, long fields which do not have the restriction of a maximum length of 64 bits may be specified. Thus, editing operations may be performed upon records rather than individual fields. In arithmetic instructions, multiple fields may be specified and the arithmetic process is repeated for each field specified by the given instruction.

Individual characters in a field may occupy four or eight bits. Thus, a decimal digit may be compactly represented by a binary code of four bits, or it may be expanded to eight bits when intermixed with alphabetic information.

Operand addressing is subject to an address-monitoring system. All addresses below the prefix address of the current program are protected from store operations. This permits protection of a supervisory program against programming mistakes in a subject program. Because it is often very difficult to predict all the addresses which might be generated by indexing, indirect addressing or other address modification, the built-in address-monitoring facilities give far better protection than is possible by screening the program before execution. This is particularly true when the program is not yet error-free.

Systems required to process a wide variety of data in many different ways and on short notice may be arranged to store programs and files on external media. Because of overlapped processing and input-output operation, it is impossible to assign in advance the storage areas required for programs and data. In order to avoid time-consuming and logically complex address alterations, the relocation mode can be used. This mode has the additional advantage of isolating, and therefore protecting from each other, the instruction and data areas belonging to the different programs which may appear simultaneously in storage.

When programs are run in the relocated mode, all data and instruction addresses are translated prior to addressing storage. A translation table associated with the program allows individual relocation of several data areas and instruction areas. Relocation takes place in blocks of 256 words. The translation process permits continuous addressing of noncontiguous data areas. Relocation may be performed with all types of operands and every mode of address modification.

1.3.3 Arithmetic and Logical Processing

The instruction set includes elementary operations, such as ADD (A), MULTIPLY (M), and DIVIDE (D), which are altered by modifier bits. Thus, the operations "subtract" or "add absolute" are obtained by use of sign modifiers with an ADD (A) instruction, and are not provided as separate operations. The same modifiers permit controlling the sign of a number to be multiplied or divided.

1.3.3.1 Floating-Point Arithmetic

Floating-point arithmetic makes use of a specialized data format wherein numbers are represented as a signed exponent and a signed fraction which, together, occupy a full 64-bit word.

Floating-point instructions contain sign modifiers which permit any desired combination of operand signs. They also contain a normalization modifier which specifies the choice between normalized and unnormalized operation. Normalization shifts take place four bits at a time. Consequently, the operating radix for floating point is said to be hexadecimal, rather than binary. The exponent of the floating-point word is a hexadecimal number and denotes the power of 16 by which the fraction should be multiplied.

The fraction length may be either 32 or 48 bits as specified by the precision bit of the floating-point word. The 32-bit fraction length gives the fastest operating speed. The 48-bit fraction length makes it possible to compute in single-precision mode for a number of problems which would have to be done in double-precision otherwise. The two fraction lengths may be intermixed within one computation.

In order to simplify significance studies, a mode of operation called "noisy mode" is provided in which results are altered in a specified fashion. Consecutive runs of the same problem in standard and noisy mode permit an estimate of the significance of the results to be obtained.

Floating-point numbers cover a range between the positive and negative value of the fraction having the maximum exponent. Since the exponent range is finite, a discontinuity exists between positive and negative values of fractions having minimum exponent. Included in this range is the number zero. An extremum flag bit has been incorporated in the floating-point word format to provide a straightforward control of data exceeding the exponent range or falling within the range of discontinuity.

1.3.3.2 Fixed-Point Arithmetic

The class of fixed-point arithmetic operations is designed to facilitate arithmetic on other than the specialized floating-point numbers. The emphasis here is on versatility and economy of storage. Data may be signed or unsigned. For unsigned data, the sign is simply omitted in storage, thus saving space and avoiding the task of assigning signs where there are none. Unsigned numbers are treated as if they were positive.

All integer operations are available in either decimal or hexadecimal form by setting the radix modifier bit. Again, the term "hexadecimal" is used in preference to "binary" to indicate that addressing and processing proceed four bits at a time.

A radix-conversion operation is provided to facilitate the use of decimal input and output while retaining the advantages of hexadecimal operation within the machine. In this operation, the operand is an integer and may be converted either from hexadecimal to decimal or from decimal to hexadecimal.

1.3.3.3 Logical Processing

The logical-processing operations are divided into three groups: connective, translation, and alphameric-comparison operations.

Connective operations provide a simple and orderly fashion for performing operations which logically combine bits by "and," "or," and "exclusive or," as well as many other nonarithmetic data-handling operations. Each connective operation specifies two or three storage fields as in integer arithmetic. Two operand fields are logically combined bit for bit and placed in the result field. The result is tested for the all-zero condition. The instruction CONNECT FOR TEST (CT) allows fields to be tested without altering the contents of storage.

There are sixteen possible ways to combine or connect two bits, each of these can be specified with the connective operations. Besides the connectives "and," "or," and "exclusive or," there are connectives to invert, to replace, and to set bits to zero or one. While the term "logical connectives" suggests evaluation of elaborate expressions in Boolean algebra, the connective instructions have important every-day applications, such as assembling and testing input-output data. Their power lies in the ability to specify fields of any length and in any position in storage.

Several modifications, which may be combined in any desired way, are provided with the connective operations. One modification, the choice between storing the result or only testing the result, has been mentioned already. Another modification is the ability to connect a field with a single four- or eight-bit character. This single character is connected over and over with the successive characters of the field. A third modification is the match or clash option. In this option, the result characters are compared with a four- or eight-bit comparison character. An equality is called a match; an inequality, a clash. The operation is terminated either on the first match or on the first clash. This operation has the characteristics of a search operation.

Translation operations also proceed a character at a time. Either a four- or eight-bit character is selected from the operand field as the argument for the table from which the result character is obtained. The result character is four or eight bits independent of the size of the argument character. Successive argument characters are used to obtain a stream of result characters, which are placed in the result field.

The translation operations have direct application in code conversion operations. A second application is character-size alteration by translation from four- to eight-bit size, or vice versa. A third less obvious application is character reordering. Here, an operand stream of control characters is used to select, from a translation table containing the subject data, the result characters in a prescribed order. This reordering function has application in such editing functions as punctuation insertion.

A fourth application of the translation operations is logical manipulation. Two additional translation instructions, TRANSLATE WITH LEFT CARRY (TLL) and TRANSLATE WITH RIGHT CARRY (TLR) are provided to increase the logical power for this class of operations. In these operations, part of the result character is used as argument for the next character translation. The two operations differ in the direction of processing. These translation operations are also used in editing operations, and provide means for zero suppression and similar functions.

Alphameric-comparison operation differs from the numeric comparison operation (provided as part of the arithmetic operations) in the direction of processing, and in that the field length is unrestricted. Alphameric high-low comparisons are made from right to left by simple binary subtraction of successive characters. There is no fixed character code built into the computer. The only requirement is that the binary numbers representing each character fall into the comparing sequence desired for the application. If the code used for input or output does not conform to this comparing requirement, the translate instructions may be used to perform the desired conversion.

1.3.4 Control-Word and Decision Operations

Every instruction may have its address portions modified by indirect addressing, or substitutionary indexing. In the case of floating-point operations, additive indexing is provided as well. The information referred to in indirect addressing and in indexing is the value field of a control word. Normally, the instruction and the control word remain unchanged. To alter the control words is the function of control-word operations.

Each control word contains a count to keep track of the number of times a program loop has been traversed, or in the case of long fields, the number of times a word boundary has been passed. A third field in each control word specifies a refill address from which another control word may be loaded automatically.

Together, these three fields provide a very convenient indexing technique. At each traversal of a program loop, an increment is added to the control-word value and the count is stepped down by one. When the count reaches zero, the control word is reset by refilling it from the storage location containing the original value and count. All this may be done with one instruction at the appropriate point in the loop.

The instruction set permits many other control-word techniques. An important one is the use of the refill address to indicate the next control word in succession in a chain of control words. Such chains permit the computation to progress through a series of items which are not stored in the order in which they are to be used. Chaining can greatly simplify insertion, deletion, and sorting of items by not requiring rearrangement of data in storage.

In logical-processing operations, control words are used to specify operands which have a length greater than 64 bits. The refill address is used to chain together successive data fields. Thus, the control words permit entire records or groups of records to participate in a single operation.

The instruction set includes operations for incrementing the value field, decrementing the count field, and chaining. Several of these functions may be combined in one operation. The counting function may also be combined with fixed-point arithmetic in a set of very flexible control-word modification operations.

While control-word instructions are provided to change control words explicitly, it is also possible to take advantage of an implicit control-word modification mode. This mode is provided by the fact that control words which are used in processing operations are always stored back into a fixed storage location at the end of the operation. These so-called "final control words" are carefully updated to reflect correctly the point reached in processing the operand. The final control words can be used in subsequent operations involving the same record. This mode of operation may be applied to advantage in stepping along a string of data of various lengths without requiring a separate incrementing instruction at each step.

The branching operations alter the instruction counter so as to change the course of a program. The number of operations is not large, but modifiers are available to provide a great deal of flexibility. The branch instruction refers to the four-bit condition register which reflects the result of the instruction last executed. By means of four select bits, the branch decision may be based on the state of some or all of the condition bits. The various options include an unconditional branch and a NO OPERATION (NOP) instruction. Branch instructions may be coupled with an operation to store the program control word at any desired location before branching. This simplifies re-entry to a program from a subprogram.

The condition register may be loaded prior to a branch operation from any four-bit location in storage.

1.3.5 Data Transmission

The transmission operation provides the facilities to move a block of data from one set of addresses to another. The operation may be terminated by a control-word count, or by a match or clash. The match or clash condition is obtained by comparing one character in each word transmitted with a comparison character. Additional operations are provided to interchange the contents of storage locations and to move half-words.

There are two basic instructions for controlling input-output and external storage units: START CHANNEL (SRT) and RELEASE CHANNEL (RLS). Each instruction specifies the unit desired and the storage area involved in the data transmission.

The storage area is specified by a control word which contains the first address in storage and a count of the number of words to be transferred. The control word also specifies the operation to be performed by the channel. The control word contains a refill address which can specify the address of another control word. Control words can thus be chained together to define storage areas and channel operations to be executed with these storage areas. This makes it possible to proceed with a series of operations on a given channel without further attention by the main program.

Input-output control words have the same format as control words used in indexing and indirect addressing. This important feature greatly simplifies the processing and sorting of large files.

All instructions for operating external units are executed independently of the computer program. A number of data transfers can thus take place simultaneously, all sharing access to storage. Signalling functions inform the program when an external process is completed.

1.3.6 Maintenance

Throughout the processor, a high degree of checking is used for data and controls. The internal organization of the processor makes it possible to halt machine operation when an error is detected. The state of the internal registers therefore is preserved, and may be used to determine the type of error and the location of the fault which caused the error. To facilitate diagnosis, the contents of the machine registers are automatically scanned and placed in storage when an error is detected. If desired, the scanned information may be dumped on an output medium, thus producing an external record of all error counts which occur. This procedure aids in diagnosing intermittent errors.

When a permanent fault is known to exist in the processor, a scan-in, as well as a scan-out, may be employed -- internal registers of the processor are set according to data which resides in storage. Following this scan-in, a predetermined number of machine cycles is taken, followed by a scan-out. The scan-in information may be obtained from an input device; the scan-out information may be dumped on an output device, and a large number of scans may be taken in succession. Thus, an exhaustive equipment test may be performed in a short time.

Machine-scans may also be initiated by program control as an aid to diagnostic programs.

Chapter 2

OPERAND DESIGNATION

Contents

Contents Chapter 2 (continued)

Contents Chapter 2 (continued)

Chapter 2

OPERAND DESIGNATION

2.1 GENERAL DESCRIPTION

The central processor is capable of addressing 65,536 (2^{16}) words of storage, each containing 64 bits of information. Most of these locations contain general-purpose information, but some locations have been assigned for a special purpose. The information obtained from core storage may be used as data or as program.

Data may be alphameric or numeric. Depending upon the type of processing to be performed, the data format may be a full word or a half-word, a field or a record. A field has variable length, and may start within one word and continue through that word into the next higher addressed storage location. The length of a field is limited to 64 bits. A record also has variable length, but it is not limited to 64 bits. Thus, records and fields can cross word boundaries, and may be placed side by side in storage.

Processing of data proceeds four or eight bits at a time. Alphameric data consist of characters which are processed in a uniform manner. Numeric data consist of digits and, where needed, a sign. Floating-point data consist of a signed fraction and a signed exponent. Floating-point arithmetic operations make use of an accumulator. All other operations place their results in general-purpose storage.

Data may be moved internally by a set of data-transmission operations. To transmit data to and from the accumulator, LOAD ACCUMULATOR and STORE ACCUMULATOR operations are provided.

The program specifies computer operation by means of instructions and control words. Instructions are executed sequentially -- control words are consulted and updated in any desired sequence.

The program specifies processor operation by means of instruc- a variable-length instruction format is used. Instruction addresses may be modified by indexing or indirect addressing.

The index quantity or operand specification required for this address modification is placed in control words. Control words are also used to specify the addresses of the data which take part in input-output operations. A third use of control words is the specification of the details of instruction sequencing and interruption.

The contents of control words follow a standard pattern. A set of control-word modification operations is provided to change the various parts of control words.

Internal registers, in general, are not addressable and do not preserve information from one operation to the next. The only information kept in internal registers as the processor proceeds from one operation to the next is the content of the program control-word register. The program control word is fetched at the start of a program and stored at the end of a program. Among the information contained in the program control word is the prefix address. This address is used as a reference address, and specifies the location of the index registers and accumulator of a particular program. The program control word and its prefix address avoid the need for dumping and loading of registers in switching from one program to another.

Some applications require program and data to be placed at different storage locations as processing proceeds. To simplify storage allocation and address modification for these applications, a relocation mode may be used. In this mode, the addresses used in a program are considered symbolic. A translation procedure is used to change them to actual addresses each time they are used to address storage.

The programs executed by the processor often are known to contain mistakes. When these programs are executed, it is important to protect certain storage areas from erroneous store operations to preserve proper processor control. Address monitoring provides this storage protection. The extent of the monitored area is determined by the prefix address.

2.2 FORMAT

The system stores information in core storage in 64-bit words. Information transmission betweeen storage and processor is in parallel, a 64-bit full word or 32-bit half-word at a time. Information words of 64 bits are part of 72-bit machine words. The extra eight bits are single error detection check bits and are not available to the programmer. Within the central processing unit, each check bit is used as the odd-parity bit of eight consecutive information bits.

Words in storage are specified by a contiguous set of addresses numbered from 0 to 65,535 (2^{16}-1). Words are subdivided into sixteen groups, or "bytes," of four bits each. In each word, the bytes are numbered from 0 to 15, left to right. Byte 15 of a word may be considered adjacent to byte 0 of the next higher addressed word. The four bits of a byte are numbered from 0 to 3, left to right. Numbering of bytes and bits within a byte may be combined by numbering the 64 bits of a word from 0 to 63. The parity bits are not numbered.

The set of addresses includes the addresses of core storage used for special purpose. A list of these addresses is described in the storage assignment section. There are no processor registers which need addressing, since all necessary information is placed in storage in the course of the machine operation.

In some operations, information is processed a word at a time; in others, a half-word at a time. A half-word contains eight bytes of four bits each and corresponds either to the first half (bits 0-31) or the second half (bits 32-63) of a full word. In a third group of operations, information is processed a byte at a time. The byte size may be four or eight bits. Eight-bit bytes always consist of two consecutive even and odd numbered bytes. A group of bytes which is processed serially is called a field. A field may start with any byte in a word in storage, and continue through that word and into the word in the next higher addressed storage location.

Information, whether a full word, half-word, or field, is always addressed by the leftmost byte. The rightmost byte is either implied by the operation to be performed or specifically stated as part of the instruction.

In all serial processing operations, operand data are fetched from storage a half-word at a time or an eight-bit byte at a time. Results are always stored a half-word at a time. When operand and result data fields which overlap partially are chosen, caution should be taken that results do not replace operands before the latter are fetched for processing. When operand and result fields are identical, no complications arise except where noted.

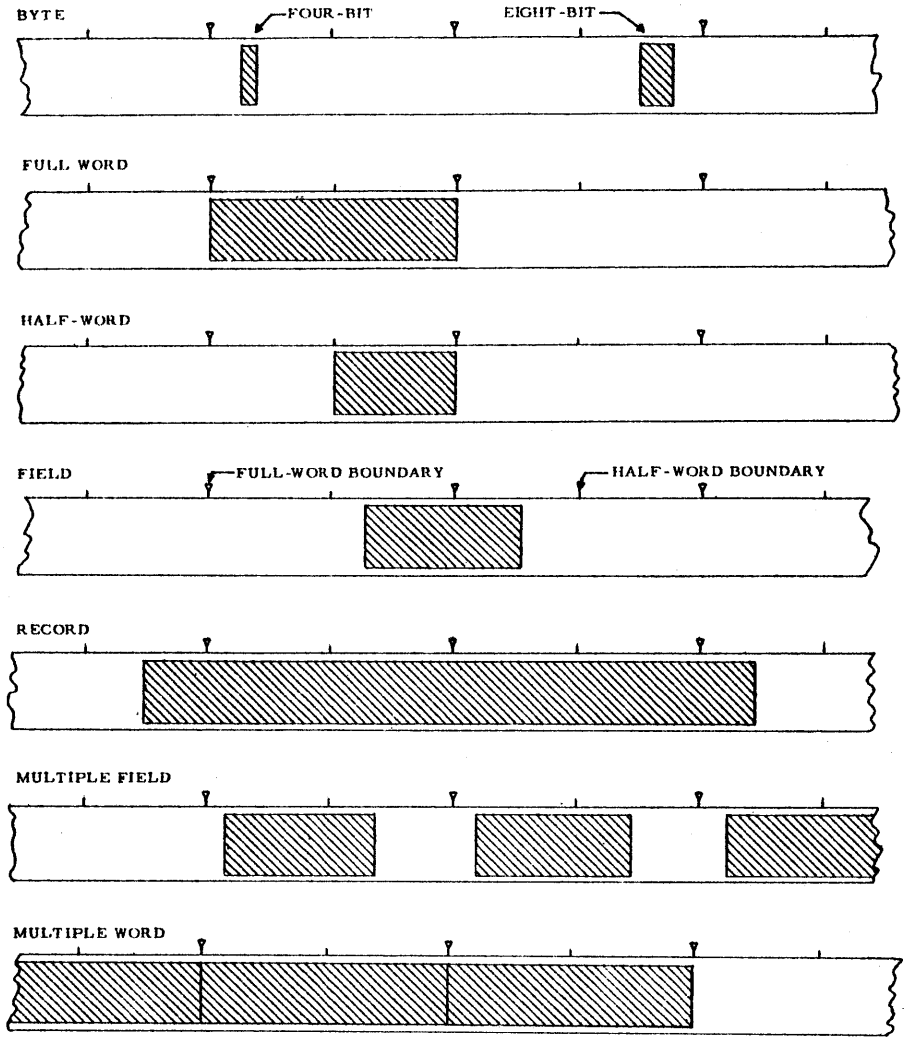


Figure 2.1 - Data formats

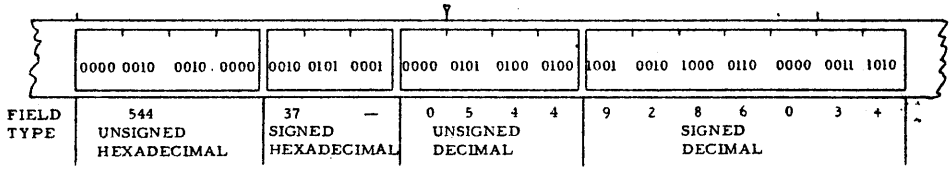


Figure 2.2 - Numeric fields

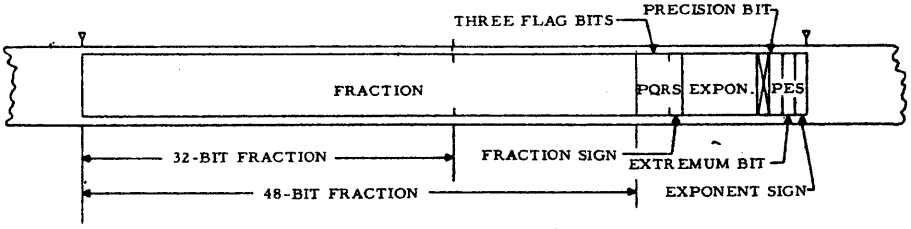


Figure 2.3 - Floating-point data format

2.2.1 Data

Seven data formats are used in the processor. These formats are closely related and may be considered variations upon the basic field format. Examples of these formats are shown in Figure 2.1 . The byte, word, and half-word format correspond to the basic subdivisions of storage in bytes, words, and half-words. The field format has a length of one to sixteen 4-bit bytes, or one to eight 8-bit bytes. The field format is not restricted by word or half-word boundaries, and may start and end with any byte within a word. In a record, the number of bytes is not limited. A record therefore may continue through many words in storage. A multiple field consists of a set of fields equal in length and with their left-hand boundaries sixteen bytes apart. The fields, therefore, occupy the same groups of bytes in successive full words. Multiple fields may cross word boundaries. The multiple-word format consists of one or more adjacent full words.

Fixed-point arithmetic operations process data four or eight bits at a time in hexadecimal or in decimal radix. In decimal arithmetic, the digits 0 through 9 are represented by the four-bit binary integers 0000 through 1001. Hexadecimal arithmetic makes use of the same four-bit size with the values 0 through 15 being represented by the binary integers 0000 through 1111. The processor uses the hexadecimal rather than binary radix, because processing proceeds four bits at a time. Therefore, the processing resolution is four bits and not one bit. Otherwise, hexadecimal processing is in all respects equal to binary processing.

Numbers may be signed or unsigned. When present, the sign of a number is specified by a single bit. A zero indicates a positive sign; a one, a negative sign.

With the sign, up to three flag bits may be used. Flag bits permit special identification of numbers and are placed to the left of the sign bit to form with this bit the four-bit sign byte.

For unsigned numbers, the sign byte is omitted and the sign is assumed to be plus. During arithmetic operations, all numbers are handled as if they were signed. When unsigned operands are obtained from storage, the implied sign is modified as specified by the instruction. Thus, an operand can be brought in without sign and the result stored with sign or vice versa.

In fixed-point arithmetic, the data fields specified have variable field length. For unsigned data, the entire field is treated as a positive integer. With signed data, the rightmost byte is interpreted as a sign byte. Figure 2.2 shows possible numeric fields.

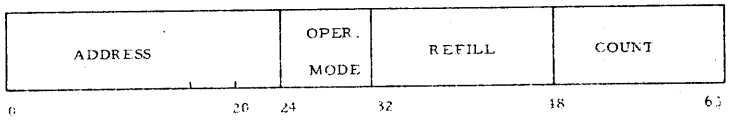
The format for floating-point arithmetic is shown in Figure 2.3 The two portions of the floating-point word are: left, the fraction; and right, the exponent. Both fraction and exponent are signed, and have the hexadecimal radix. The fraction has 32 or 48 numeric bits. Flag bits of the fraction sign byte serve for the entire floating-point word. The exponent has two numeric bytes. The flag bits of the exponent sign byte are used for the extremum bit, which indicates a true zero or infinite quantity, and for the precision bit, which specifies fraction length. The third flag bit is ignored.

Logical-processing operations usually process information four bits at a time, but occasionally processing proceeds eight bits at a time. Input-output operations always proceed eight bits at a time. Alphameric characters normally occupy an eight-bit byte. No preferred code sets other than those implied by a binary collating sequence are assumed by these processing operations.

2.2.2 Program

The processor program consists of instructions and control words used in the control section of the processor to specify the operation to be performed. They are stored in core storage and when desired may be operated upon as data. Instructions occupy one, two, or three half-words. They are referred to as "one-address," "two-address," and "three-address instructions." Each of these instruction types may start at bit address 0 or 32. Control words occupy a half-word or a full word and are distinguished as short or long control words. A long word always has bit address 0, while a short word may have bit address 0 or 32. The formats used for instructions and control words are shown in Figure 2.4 . In this figure, the bit address of all instructions and control words is assumed to be 0. In some cases, this bit address may be 32, in which case the addresses of the individual fields in the instruction differ from those shown. For convenience of presentation, all references of bits and fields will use the numbering of Figure 2.4 .

Instructions and control words follow a basic pattern upon which some variations are made. This pattern consists of four parts: the address, the operation or mode, the refill, and the count.



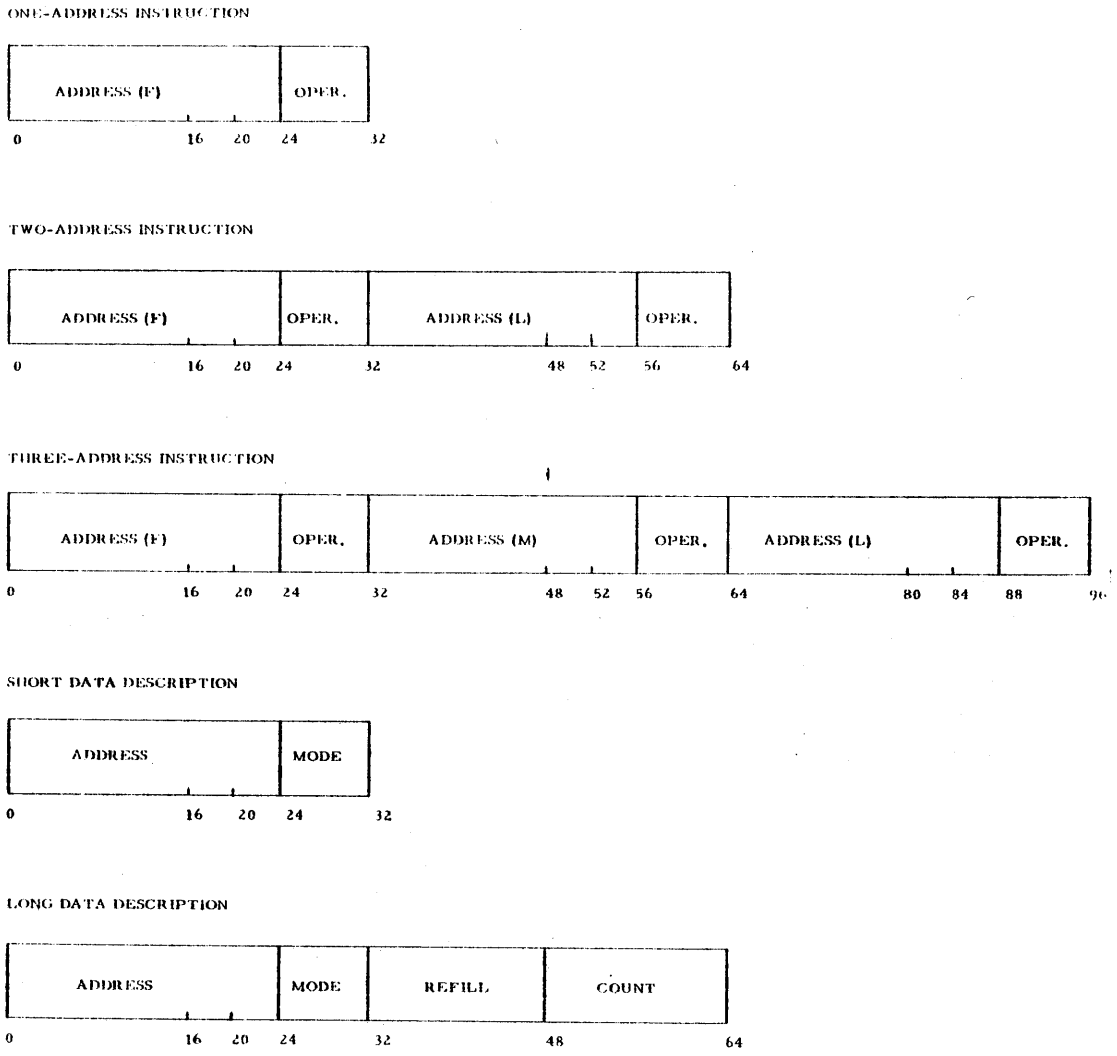


Figure 2.4 - Instruction and control word formats

The entire pattern occupies a full word and is the format of a long control word. The first two parts of the pattern occupy a half-word and can appear alone as an instruction half-word or as a short control word. Records, multiple fields, and multiple words can only be described by the long format. All other data can be described by the short format.

2.2.2.1 Instruction

The total length of an instruction in half-words is directly related to the number of addresses necessary to perform the operation. The half-words are distinguished as the first (F), middle (M), and last (L) half-word. Each half-word subsection consists of two parts: the address and the operation.

The address part consists of 24 bits and is leftmost in the instruction half-word. The first twenty bits specify the byte address, while the last four bits specify the limit of a field or in branch instructions the branching conditions. The operation part is rightmost in the instruction half-word and contains eight bits which are used as class, as code, and as modifier bits. Class bits specify type of addressing, instruction length, and operation class. Code bits specify basic operations within a class, whereas modifier bits specify modifications to these operations.

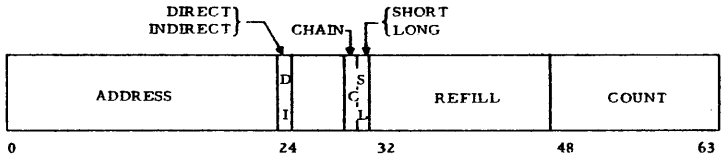
The last half-word of each instruction has bit 27 as a zero. All other instruction half-words have this bit as a one. This code bit may be used to separate an instruction stream into individual instructions. Bit 28 of the first instruction half-word is zero for two-address instructions and one for three-address instructions. The length of the instruction is determined from bits 27 and 28 of the first instruction half-word and is verified against bit 27 of the last instruction half-word. If, in the case of two- or three-address instructions, bit 27 is not found to be zero, interruption code Operation Code Invalid (OP) is given.

2.2.2.2 Control Word

Control words in general describe information. When this information is used as the operand of an instruction, the control word is called a "data description." Instructions refer to data descriptions by means of indirect addressing. Data descriptions may have the short or the long format.

Control words are also used for input-output control, for instruction-sequencing control, and for interruption control. These control words all have the long format.

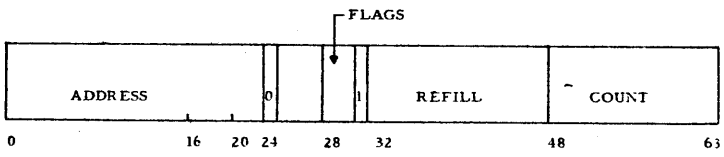
The leftmost 24 bits of a data description form the address part in the same way as for an instruction half-word. The next eight bits are the mode bits. Mode bits specify type of addressing and type of data description, including the distinction between a short and a long data description.



In a long data description, the right half of the full word contains the refill and the count fields, each sixteen bits in length.

Each field of a data description word is intended for a distinct purpose. The address field is used to address the operand or series of operands to which the data description refers. The address field can be changed by an increment which is either implied or explicitly specified. Implied increments normally increase the word address by one. Word address 65,535 ($2^{16}-1$) is followed by word address zero. The count field registers the number of times an increment is to be applied. The contents of the count field are reduced by one whenever a count is specified. A zero count is followed by a count of 65,535 ($2^{16}-1$). An initial count of zero is interpreted as 65,536 (2^{16}). The refill field contains the address of a new control word that can replace the original control word. Refill can be made conditional on the count reaching zero or it can be specified to occur unconditionally. By means of this refill operation a sequence may be established between control words. Such control words are said to be "chained." Chained control words may specify chained records, multiple fields, or multiple words. The operand addressing and the instruction set take full advantage of the data description fields for the above purposes. However, a different use of data description fields is possible if desired. For instance, the address field may be compared against a limit placed in the count or refill field. Sufficient operations are available to permit flexibility in this respect.

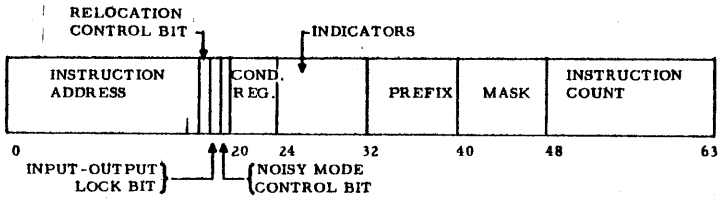
In input-output operations, a control word is used to control data transmission to and from the external units. In these operations, some of the mode bits are further defined, as described in Chapter 7.



The uniformity of format allows a control word to be used both for input-output and for internal processing operations.

The input-output format is also used for the interruption control words and in modified form for the program control word.

The program control word format is shown below. The address field of the program control word is shortened to seventeen bits, leaving room for mode and condition bits. The address field is used to specify the next instruction to be performed. Three mode bits are available to specify addressing, input-output, and floating-point operating modes. The condition bits refer to the final condition of the current instruction. The next two bytes are used for indicator bits. These bits record program alert situations. The refill field is used for the prefix field and the mask bits. The prefix field is a truncated full-word address. The eight low-order bits of this address, the prefix address, are omitted because they are always zero. Mask bits are used in conjunction with the indicator bits and specify which program alerts may cause an interruption. Further explanation of the purpose of these fields is found in Chapter 3.



2.3 INDIRECT ADDRESSING

The address part of an instruction half-word is normally used to address the data upon which the instruction operation is performed. This addressing mode is called "direct addressing." In another mode, bits 0-16 of the address can be used to address a data description which in turn contains an address part that can be used to address data. The process of using a first-level address to obtain a second-level address is called "indirect addressing." The second-level address can in turn refer to another data description of which, again, the address can be used. In this manner, it is possible to extend indirect addressing through many levels. Indirect addressing has a variety of applications as a programming tool. In addition, it should be noted that since some data formats can only be described by long data descriptions, indirect addressing is required to specify these formats.

Bit 24 of an instruction half-word or a data description indicates whether or not indirect addressing will take place. When bit 24 is zero, direct addressing takes place. When bit 24 is one, indirect addressing takes place. Indirect addressing applies only to the address which is part of the instruction half-word and may be specified for each instruction half-word independently. Indirect addressing is also independent of the operation to be performed. Conversely, the operation as specified in bits 27-31 of an instruction half-word is not altered by indirect addressing. Bits 25 and 26, when used as the byte size modifier and signed-unsigned modifier in two- or three-address instructions are replaced by the corresponding bits of the data description. Otherwise, bits 25 and 26 of the instruction half-word remain unchanged by indirect addressing.

Each time a data description is fetched in indirect addressing, bit 24 is inspected. When bit 24 is one, indirect addressing continues through another level, using bits 0-16 to address the next data description. When bit 24 of the data description is zero, a direct address is indicated. The data description is short or long accordingly as bit 31 is zero or one. A long data description is normally obtained from a full word address location. When bit 16 of the address referring to the control word is one, that is, when it refers to the right half of a full word address location, the right and left half of the control word are interchanged. The middle address (M) of a three-address operation can never refer to a long data description. For this address, bit 31 is not tested, and a short data description will always be assumed.

When a refill operation takes place, bit 24 of the new control word is again inspected, and any indirect addressing which may be specified is performed. At the end of the indirect addressing operation, bit 31 again is inspected to distinguish short and long data descriptions.

During an input-output operation, no indirect addressing takes place when a refill operation replaces an input-output control word by its successor. Bit 24 and 31 of the new control word are always ignored under these circumstances.

The last data description fetched by a refill or indirect addressing operation is retained in its entirety. The processor has enough register storage to retain the necessary instruction code and modifier bits, as well as all 64 bits of a long data description. As a result, the processor may return a long data description to storage at the end of an operation with updated address and count fields, but otherwise identical to the data description which was fetched. Storing of these final data descriptions is further described in Section 2.5 .

In indirect addressing, it is possible that through a programming error the addresses refer to each other in such a way that they form a loop. As a result, the processor cannot finish its operation and no interruption is accepted. To prevent the machine from being locked in this way, an instruction will be terminated if still active after one-tenth second, and an interruption code Unending Sequence will be stored in the interruption stream.

PROGRAMMING NOTE

The indirect-addressing action for bits 25 and 26 makes it possible to specify byte size and signed-unsigned options as part of a data description.

2.4 INDEXING

Floating-point arithmetic, accumulator and control-word instructions permit address modification by indexing. A total of 255 index words may be specified by a program.

Bits 16-23 of the instruction are used to indicate the index to be selected. When these bits are all zero, no indexing takes place, otherwise these bits are used as the index address. The index words are found at locations 1-255 following the prefix location of the program. Therefore, the address of an index word may be obtained by using the prefix field, bits 32-39 of the program control-word, as the high-order part, and the index field, bits 16-23 of the instruction as the low-order part.

Some of the index locations are special-purpose locations, such as final control-word and accumulator locations, and therefore serve a dual purpose.

Address modification is performed by adding bits 0-15 of the index word, the value field, to bits 0-15 of the instruction. Both fields are treated as unsigned hexadecimal integers and any overflow carry is ignored. Since only bits 0-15 of the index word are used in address modification, the other bits may be assigned in any desired way. In particular, when the control-word format is chosen, control-word operations may be used to change the index value.

It is permissible to specify both indexing and indirect addressing in one instruction. When both are specified, indexing takes place first, then the modified address is used in indirect addressing. Following the indirect-addressing operation, the new data description is inspected again for any additional indexing. Indexing and indirect addressing will alternate when each is specified in subsequent data descriptions. When no indexing is specified, indirect addressing takes place at once. The process terminates when no further indirect addressing is designated. The combined indexing-indirect addressing process is subject to the one-tenth of a second time limitation. When the time limit has been exceeded, the interruption code Unending Sequence (US) will be stored. When a refill operation takes place, both indexing and indirect addressing may be performed.

2.5 OPERAND ADDRESSING

The address part of an instruction half-word may be modified by indexing or indirect addressing, as has been described. The final address thus obtained is called the "effective address." When no address modification occurs, the address part of the instruction half-word is the effective address. An effective address always contains 24 bits, but depending upon the operation, less than 24 bits may be required. In each case, only the applicable part of the effective address is used, the nonapplicable part being ignored. When a long data description is used to address an operand, the refill and count field, as well as the effective address, are used to specify the operand.

The refill field of a long data description may be used to obtain a new data description. This new data description may be short or long and may specify indirect addressing or indexing, depending upon the state of bits 31, 24, and for indexable instructions, bits 16-23. The new data description may also introduce a new byte size and signed-unsigned modifier.

The refill field introduces the possibility of an unending chain of control words. Because of a programming error, this chain of control words may result in an unending operation. To prevent such an operation from locking the processor, the operation will be terminated after one-tenth second, at which time the interruption code Unending Sequence (US) will be stored.

The final values of long data descriptions are placed in core storage at the completion of the operation. These values are called "final data descriptions." The data description belonging to the first instruction address, called the "first final data description," is stored at the address two above the prefix address of the current program. The data description belonging to the last instruction address, called the "last final data description," is stored at the next higher word address. Short data descriptions are stored only in data-transmission operations.

The storing of final data descriptions makes it possible to use them in subsequent operations. The final data descriptions, furthermore, identify the results of match-type operations.

2.5.1 Byte

The operation LOAD CONDITION REGISTER addresses a single four-bit byte in memory. For this operation, the first twenty bits of the effective address are used to specify the addressed byte. In byte connect and in match-type operations, a single byte is also addressed. The addressed byte may contain four or eight bits. Accordingly, the first twenty or nineteen

bits of the operand address are used to specify the byte. The choice of byte size is indicated by the modifiers, bits 25, 57, and 89.

2.5.2 Word

A full-word operand is implied by floating-point operations, control-word operations, and full-word transmission operations. These operations use the first sixteen bits of the effective address.

2.5.3 Half-Word

A half-word operand is implied by branch and half-word transmission operations, and whenever indirect addressing is specified. For these operations, the first seventeen bits of the effective address are used. The sixteen high-order bits address a full word; the low-order bit addresses the first or second half of the full word.

2.5.4 Field

Operands of fixed-point or logical-processing operations normally have the field format. To specify a field, all 24 bits of the effective address are used. The sixteen high-order bits address a full word and the next four bits address the byte in the addressed word which is the leftmost byte of the field. Bits 20-23 address the rightmost byte of the field. This byte may be part of the addressed word or of the next higher addressed word in storage. The word-address of this byte need not be given since the field length never exceeds sixteen bytes. The leftmost and the rightmost bytes of the field thus defined are part of the operand. When the byte address of these two bytes is the same, a field length of one byte is implied.

2.5.5 Record

In logical-processing operations, a long data description may be used to address a record when processing from left to right. The leftmost byte of the record is addressed by bits 0-19 of the data description. The count field specifies the number of full words in which the record is placed. This number includes the words in which the leftmost and the rightmost bytes are located, although these words may not be fully occupied by the record. When no word boundaries are crossed by a record, the count is one. When a record crosses a word boundary, the count should be two. If, in this instance, a count of one instead of two is given, the processing will terminate either within the first word or at the word boundary.

The byte address of the rightmost byte of the record is specified by bits 20-23 of the data description.

Only the first and last address of an instruction may refer to a long data description and thereby specify a record. The middle address of a three-address instruction cannot specify a record.

Records may be changed by means of a refill function. After processing the last byte of a record, a new data description may be used to specify the record with which processing is to continue. The first byte of the new record follows the last byte of the current record without disruption of the logical-processing operation. The full-word location of the new data description is defined by the refill address of the current data description. Refill operation only takes place when bit 30 (the chain bit) of the current data description is one. When the chain bit is zero, no refilling takes place. As part of a refill operation, the byte size of the record may be changed.

A logical-processing operation may use both fields and records for operands and results. The operation is terminated when the first termination of a field or a record occurs. In some operations, termination may occur even earlier because of other conditions which will be discussed later.

The final data description placed in storage specifies the part of the record or records which remain to be processed; that is, the address, count, and refill field of the final data descriptions have been updated to refer correctly to these remaining record parts. If the entire record were processed, the byte beyond the rightmost byte of the record is addressed, and the final count is zero.

2.5.6 Multiple Field

The multiple-field format permits arithmetic operations to be repeated with a series of operands. A long data description is used to address the multiple field. The address part of the data description is used to address a field in the same manner as is the case for the field format. When the numeric operation on the specified field is completed, the word address of this field is incremented by one and the operation repeated with the newly defined field. At the same time, the count of the data description is reduced by one. When the count reaches zero, operation is terminated unless the chain bit is one, in which case the data description is refilled. When two multiple fields are used in an operation, the multiple field which terminates first will terminate the entire operation. When one of the operands in a multiple-field operation is specified by a single field, the data in that field is used repeatedly. The middle address of a three-address instruction cannot refer to a multiple field.

Multiple-field operation may be specified in all floating-point operations and in fixed-point operations with the exception of DIVIDE. Multiple field may also be specified in TRANSLATE WITH LEFT CARRY. The first and last operand should both be a field or both be a multiple field in all operations except ADD, TRANSLATE WITH LEFT CARRY, and LOAD, otherwise incorrect results will occur.

The final data descriptions specify the fields which remain to be processed. If all fields are processed, one field beyond the last field is addressed and the count is zero.

PROGRAMMING NOTE

The refill operation introduces an entire new record description and makes it possible to change byte size, signed-unsigned modifiers, length, and relative location of subsequent fields.

2.5.7 Multiple Word

In full-word data transmission, a long data description may be used to address multiple words. The address part of the data description is used to address a word, as in the case of full-word addressing. The initial count specifies the total number of words to be transmitted. The word address, bits 0-15, is incremented by one each time a word is transmitted. At the same time, the count is reduced by one. When the count reaches zero, operation is terminated provided the chain bit of the data description is zero. If the chain bit is one, the control word is refilled. When data descriptions are used for both operands of a transmission operation, the data description which terminates first will terminate the operation. When a data description is used for only one operand, the effective address of the other operand is incremented until the operation is terminated by the data description.

In data-transmission operations, the final data description, including address, count, and refill fields, specifies the words which remain to be transmitted. If all words are transmitted, one word beyond the last word is addressed and the final count is zero.

Two final data descriptions are stored even if only one long description is specified. A short data description or an instruction half-word when stored will have the full-word address correctly updated, and bits 24 and 31 will both be zero, and therefore can be used in indirect addressing. The other bits of this final data description are uncertain and should not be used.

Input-output control words also address multiple words. Data transmission in input-output operations is in several ways similar to internal transmission; chaining is provided and final control words are placed in storage. Input-output data transmission and multiple-word definition are described in Chapter 7.

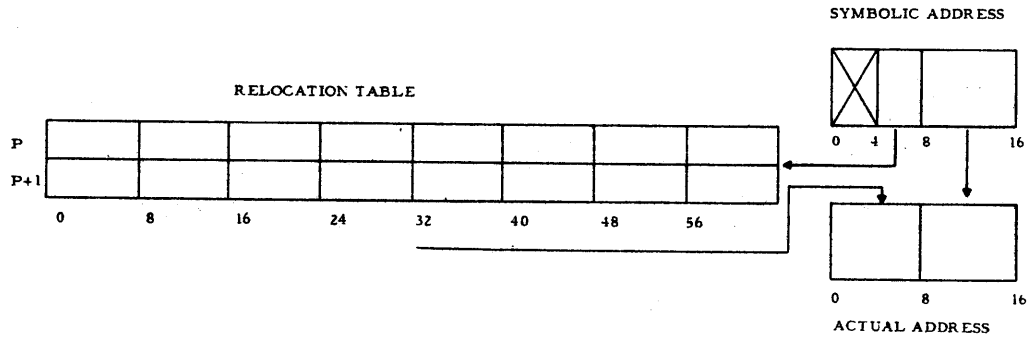
2.6 ADDRESS TRANSLATION

Address translation, provided to allow dynamic program relocation, only takes place when the relocation mode is specified in the current program control word. The relocation control bit, bit 17 of the program control word, when one, specifies the relocation mode; when zero, no address translation takes place. Input-output operations are not controlled by a program control word and therefore are never in the relocation mode.

In the relocation mode, all addresses supplied by and available to the program are treated as "symbolic addresses." These symbolic addresses are changed by means of a relocation table to "actual addresses" when storage is addressed. As a result of address translation, storage is divided into 256 blocks which can be individually assigned.

2.6.1 Relocation Table

The relocation table is used to translate a symbolic address into an actual address. The table has sixteen 8-bit entries and is placed in the two locations at the prefix address and the next higher word address.



Bits 0-3 of the symbolic address are ignored. Bits 4-7 of the symbolic address are used to select an eight-bit entry from the relocation table. This entry replaces bits 0-7 of the symbolic address, thus forming the actual address. Bits 8-15 of the symbolic address remain unchanged in the translation process.

An all-zero entry in the relocation table is called an "empty address." When such an address is obtained in the translation process, the operation is terminated, and the program is alerted by the interruption code Empty Address (EA). If several of the symbolic addresses used by an instruction are empty, only one interruption code will be stored. Following the detection of an empty address, the instruction address may point either to the next instruction in sequence or to the second or third instruction half-word of the current instruction.

2.6.2 Block Boundaries

Entries in the relocation table specify the addresses of 256-word storage blocks. The translation process makes it possible to refer with a set of continuous symbolic addresses to noncontiguous storage blocks. Because each address occurring in an operation is individually translated, the data participating in the operation may belong to different blocks. However, individual fields, records, multiple fields, and multiple words should be entirely within a block. When a block boundary is crossed, operation is terminated and the interruption code, Address Invalid (AD) is stored. Similarly, the instruction address may not be incremented across a block boundary. Again, the code Address Invalid (AD) will be given if this event occurs. In order to cross block boundaries for data, the refill feature should be used. In the case of instructions extending beyond a single block, a branch instruction should be used. An Address Invalid (AD) code will also be given when the address in the data descriptions stored in a transmission operation or the address in the program control word stored in a branch and preserve operation would have crossed a block boundary, even though the operand or instruction addresses had not yet crossed that boundary. In these cases the control word is not stored.

PROGRAMMING NOTE

Although a record or multiple field cannot cross a block boundary, the final control word will address a new block when termination occurs at the boundary.

2.6.3 Addresses Translated

Addresses are translated only when they are used to address storage. All addresses placed in storage, such as the addresses in final data descriptions or program control words, are stored in their updated symbolic form.

Address translation includes

1) all effective addresses used in fetching or storing data or instructions;

2) addresses used in fetching data descriptions or control words as part of indirect addressing;

3) the address of the initial control word of an input-output operation, including the RELEASE CHANNEL instruction; and

4) the direct address of a channel in an input-output instruction.

Addresses not translated are

1) data addresses of input-output operations;

2) addresses relative to the program prefix address, such as the accumulator, final control word, and index addresses;

3) auxiliary addresses generated as part of instruction execution; and

4) addresses used in storing an interruption code.

2.7 ADDRESS MONITORING

Information in core storage can be classified as data, instructions, and reference information. Information may be used for communicating with external units or with an operator, supervising and scheduling of programs, monitoring of program execution, or for executing problem programs. Furthermore, a difference between problem programs may be noted. Some programs are assumed to be operating properly and represent the working load of the system. Other programs are not yet in the proper operating form and are executed for debugging purposes. When a single task is performed by the system, usually a number of independent programs can be recognized which are executed in intermixed fashion. The system is furthermore capable of performing more than one task at a time, involving an even greater number of programs identified as independent units. When several programs are in core storage at one time and executed in intermixed fashion, errors in one program should not change data, instructions, or reference information belonging to another program or to the system. These errors are prevented by defining a monitored area for each program and prohibiting data storage at all locations within the monitored area.

2.7.1 Definition of Monitored Area

The execution of a program is controlled by a program control word. The prefix address, bits 32-39 of the program control word, is used to define the monitored area. Full words in locations up to and including one address beyond the prefix address are monitored for data-store operations. The monitored area therefore includes all locations below the relocation table (prefix and prefix plus one).

A prefix address of zero provides the equivalent of unmonitored operation. The next highest possible prefix address is location 256. For this address, all reference information used for program definition and communication with external units, as well as a limited amount of general-purpose storage, is monitored. When it is desired to monitor a larger amount of instructions or data, a higher prefix address can be chosen.

2.7.2 Action of Address Monitoring

The address-monitoring system is always active. It is independent of the enabled or disabled state of the interruption system, the relocation mode, or the type of instruction being executed. When an instruction specifies the storing of data at a protected address, the store operation is not executed and the protected storage location remains unchanged. The execution of the current instruction is terminated, and the program is alerted to the attempted store operation by means of the interruption code Data Store (DS).

Address monitoring applies only to store-type addresses, i. e., addresses which place information in storage. Even when the new content of storage is identical to the content prior to the store operation, the address is called a store-type address.

When the storing of an interruption code causes a program interruption from an enabled to a disabled program, a disabled program control word is used subsequently, and the prefix address of that program control word is used for all subsequent monitoring operations.

In general when a protected address is encountered during an operation, the operation is terminated. This may or may not result in complete suppression of the operation. In operations using chained data descriptions, part of the operation may already have been completed when the address alarm occurs, and hence, cannot be suppressed.

2.7.3 Addresses Monitored

All store-type addresses designated in the course of an operation are subject to address monitoring. An address associated with an operation but not actually used is not monitored. For example, the store program control word address which is part of a branch operation is not monitored if the branch is unsuccessful. Core-storage addresses used by input-output operations are not monitored. An address may cause both an Address Invalid (AD) and a Data Store (DS) interruption code to be stored.

Address monitoring includes:

1) the last effective addresses of all operations involving data storage, including the addresses obtained as a result of refill operations, and
2) the first and last effective address of a swap operation.

Not included in address monitoring are

1) addresses of comparison and connect for test operations;
2) channel addresses of input-output instructions, as well as all data addresses of input-output operations;
3) auxiliary addresses generated as part of instruction execution such as LOAD CONDITION REGISTER;

4) addresses used in storing an interruption code; and

5) the program control word address of an unsuccessful branch instruction.

PROGRAMMING NOTE

In the relocated mode, additional address protection is obtained by means of the relocation table. Only the areas specified by the table entries can be addressed; all other areas are protected. Since the relocation table is protected from store operations, the relocated program cannot change the storage assignment and address-protection specification.

2.8 CONTROL-WORD OPERATIONS

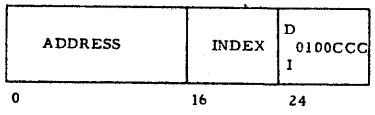
Operations are provided to change the contents of control words. These operations assume the control-word format previously described. The address part of the control word may be incremented by one, the count field may be diminished by one, and a conditional or unconditional refill may be performed.

The increment operation adds one in bit 15 of the address field of the control word. Any overflow carry from bit position 0 is ignored; address 0 follows address 65,535 ($2^{16}-1$). The increment operation is not changed by the relocation mode.

The count operation subtracts one from the count field, bits 48-63 of the control word, in bit position 63. A zero count is followed by a count of 65,535 ($2^{16}-1$). If the count remains or becomes zero, a condition bit is set.

The refill operation replaces the control word with the full word at the location specified by bits 32-47, the refill field of the control word. Refill occurs when specified by the instruction, either unconditionally or on the condition that the count field becomes zero. In these control-word operations, refill is not conditioned by the chain bit, bit 30 of the control word.

Control-word operations use the one-address format shown below. The control-word address is specified by bits 0-15 and may be modified by the index specified by bits 24-31. Neither modifiers nor long data descriptions are used.



2.8.1 Conditions

Condition bits 0, 1, and 2 are not used and are always set to zero for these operations. Condition bit 3 is used in control-word operations and identifies the state of the count field regardless of the control-word operation. The condition bit describes the control word after the control-word modification is completed, but before any refill takes place. In the operations REFILL and REFILL FROM ADDRESS, the control word is also described by the condition bit before refill takes place.

Condition bits always reflect the result of the primary operation specified in an instruction. The use of control words in addressing, indexing, or indirect addressing does not affect the setting of condition bits.

Result Count Zero

Condition bit 3 is set to one when the count field is zero. Otherwise, the bit is set to zero.

2.8.2 Indicators

No indicators can be turned on by control-word operations.

2.8.3 Operations

The following one-address instructions perform control-word operations.

INCREMENT ADDRESS (IA)

A one is added in bit 15 of the address field of the control word specified by the effective address.

DIMINISH COUNT (DC)

The count field of the control word specified by bits 0-15 of the effective address is reduced by one.

INCREMENT ADDRESS AND DIMINISH COUNT (IDC)

One is added in bit 15 of the control word specified by the effective address. The count field of the control word is also reduced by one.

INCREMENT ADDRESS AND REFILL (IR)
DIMINISH COUNT AND REFILL (DCR)
INCREMENT ADDRESS, DIMINISH COUNT, AND REFILL (IDCR)

These three instructions are identical to the preceding three except that the control word is refilled if the count remains or becomes zero.

REFILL (R)

The full word addressed by bits 0-15 of the effective address is replaced with the full word addressed by the refill field of the word at the effective address. The condition bits are set according to the original contents of the word in the effective address.

REFILL FROM ADDRESS (RA)

The full word addressed by the effective address is replaced with the full word addressed by the address field of the word at the effective address. The condition bits are set according to the original contents of the word at the effective address.

PROGRAMMING NOTE

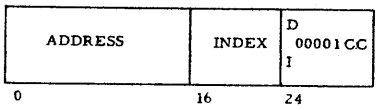
In addition to the control-word operations listed in this section, fixed-point arithmetic operations may be used to change the contents of control words. In particular, the vary and count modifiers of these operations are applicable to the control-word format.

2.9 ACCUMULATOR OPERATIONS

A pair of operations is provided to transmit full words to and from the accumulator. The accumulator is used with all floating-point operations, hence the data transmitted will normally have the floating-point format. The accumulator occupies a single full word, and is located at the prefix address plus four of the current program.

Grouped with this class of operations is a pair of address store operations employed to prepare addresses to be used in subsequent operations by indexing or indirect addressing. The addresses are stored in symbolic form when relocation is specified.

Accumulator operations use the one-address format shown below. A full-word address is specified by bits 0-15 and may be modified by the index specified by bits 16-23. The operations are specified by bits 30 and 31. Modifiers and long data descriptions are not used.



2.9.1 Condition

The condition register is not altered by accumulator operations.

2.9.2 Indicators

No indicators can be turned on by accumulator operations.

2.9.3 Operations

The following four operations form the class of accumulator operations.

LOAD ACCUMULATOR (LA)

The full word addressed by the effective address is placed in the accumulator location.

STORE ACCUMULATOR (SA)

The contents of the accumulator are placed in the location addressed by the effective address.

STORE EFFECTIVE ADDRESS, FIRST (SEAF)

Bits 0-15 of the effective address replace bits 0-15 of the first final control word location. The address of this location is two beyond the prefix address of the current program.

STORE EFFECTIVE ADDRESS, LAST (SEAL)

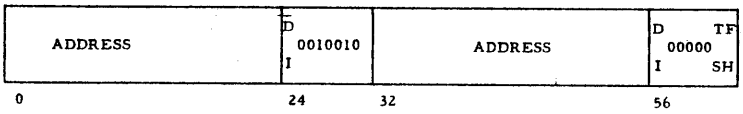
Bits 0-15 of the effective address replace bits 0-15 of the last final control word location. The address of this location is the prefix address plus three of the current program.

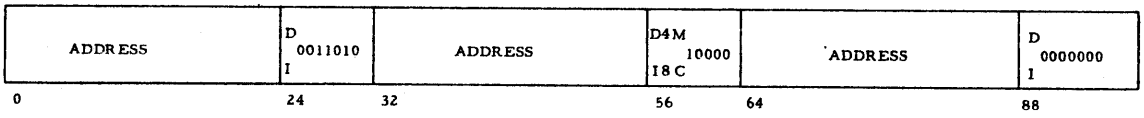
2.10 DATA TRANSMISSION

Instructions are available to transmit data from one storage location to another. Transmission may be unidirectional or bidirectional. In unidirectional transmission, data are removed from one storage location to another, the data remaining unchanged in the original storage location. The data originally occupying the receiving storage location are destroyed. In bidirectional transmission, data are interchanged so that both sets of data are preserved. Unidirectional transmission of words is specified by transmit operations; bidirectional transmission by the swap operations.

Data transmission may be performed one half-word or one full word at a time. When a full-word transmission operation is specified, either one word or multiple words may be transmitted. Transmission of multiple words may be terminated by a count, or by a match or clash condition.

A two-address instruction is used for all transmission operations, except those which involve comparison for match or clash. Transmission till match or clash requires a three-address instruction.





Bits 62 and 63 of the two-address instructions specify four operations. In the three-address instruction, bits 57 and 58 are used as modifiers to identify the type of match or clash condition required.

2.10.1 Multiple-Word Transmission

Transmission of multiple words always involves the use of at least one long data description, even when match or clash is specified. When no long data descriptions are specified by the transmission instruction, only one word is transmitted. Transmission is terminated when the count of any of the participating data descriptions is reduced to zero, or when the first match or the first clash occurs. When multiple words are transmitted, descriptions for both data areas are stored at the completion of the operation. Long data descriptions are stored in the usual manner. Short data descriptions, as well as instruction half-words, are stored also. Both of these have the word address pointing to the next word to be transmitted; bits 24 and 31 are always zero. The remaining bits of the full-word locations in which they are stored vary according to the details of execution and should not be used.

2.10.2 Match or Clash

Transmission of multiple words may be combined with byte comparison. One byte of each word transmitted is compared with a comparison byte. Equality of the two bytes is called a match; inequality, a clash. The bytes may be specified with a byte size of four or eight bits.

The comparison byte is the same for the entire operation and is specified by the middle effective address of the instruction. Bit 57 of the instruction designates the byte size of both bytes used in the comparison. When bit 57 is zero, a four-bit byte is used; when the bit is one, an eight-bit byte is used. Bits 0-19 of the middle effective address specify the address of the comparison byte with byte size four. For byte size eight, only bits 0-18 of the effective address are used. Similarly, bits 16-20 or bits 16-19 of the first effective address are used to specify the byte to be compared within each word transmitted.

The operation is terminated by the first occurring match or clash accordingly as bit 58 of the instruction is zero or one. Operation is also terminated when either multiple-word area is exhausted before the desired match or clash occurs. However, transmission of the word which caused the terminating match or clash is completed. The final data description stored does not address this word, but the next word to be transmitted.

PROGRAMMING NOTE

When the match byte represents an end character separating grouped records, the final data description identifies the start of a new record.

2.10.3 Conditions

Condition bits 0, 2, and 3 are used in data transmission operations, while condition bit 1 is always set to zero. The bits identify the cause which terminated the operation. When more than one cause occurs at the same time, all corresponding bits are turned on.

Match or Clash

Condition bit 0 is set to one when the operation is terminated by a match or clash condition; otherwise, the bit is set to zero.

First Count Zero

Condition bit 2 is set to one when the operation is terminated by a zero count in the long data description specifying the area from which data are transmitted; otherwise, the bit is set to zero.

Last Count Zero

Condition bit 3 is set to one when the operation is terminated by a zero count in the long data description specifying the area to which data are transmitted; otherwise, the bit is set to zero.

2.10.4 Indicators

No indicators can be turned on by data-transmission operations.

2.10.5 Operations

The four 2-address data-transmission instructions and the one 3-address instruction are listed below and on the following page.

SWAP HALF-WORD (SWH)

The half-word specified by the first and last effective addresses are exchanged. No final data descriptions are stored. All condition bits are set to zero.

SWAP (SW)

The full word specified by the first and last effective addresses are exchanged. No final data descriptions are stored, and all condition bits are set to zero.

TRANSMIT HALF-WORD (TMH)

The half-word specified by the first effective address is transmitted to the location specified by the last effective address. No final data descriptions are stored. All condition bits are set to zero.

TRANSMIT (TM)

The full word or group of words specified by the first effective address is transmitted to the locations specified by the last effective address. If one or two long data descriptions are used, the transmission is terminated when one of the multiple-word areas is exhausted. When no long data description is used, only one word is transmitted. The descriptions of the data used in the operation are stored at the end of the operation.

TRANSMIT TILL MATCH (TMM)
TRANSMIT TILL CLASH (TMK)

The word or group of words specified by the first effective address is transmitted to the locations specified by the last effective address. One byte of each word transmitted is compared with the comparison byte specified by the middle effective address. Operation proceeds until either one of the multiple-word areas is exhausted, or a match or clash occurs between the comparison byte and one of the bytes specified by the first effective address. The descriptions of the data used in the operation are stored at the end of the operation.

2.11 STORAGE ASSIGNMENT

The effective address of each instruction half-word provides for a sixteen-bit word address which permits addressing of 65,536 (2^{16}) storage locations. Several of these locations are reserved for specific purposes, but the remaining addresses are available as general-purpose core storage. The functions of each of the special locations are described in the following sections and the storage locations assigned to these functions are summarized in the table at the end of the chapter.

Storage assignment is either relative to address 0 or to the prefix address of the current program control word. When several programs are in storage concurrently, each may have a different prefix address in its program control word. Therefore, a different storage assignment is possible for each program. The storage assignment relative to the prefix coincides with the first addresses of storage when the prefix is zero.

2.11.1 Relocation Table

The relocation table used for address translation in the relocation mode is stored at the prefix address and the next higher address. The table contains sixteen 8-bit entries. Each entry represents the high-order eight digits of a storage address which is used to address a 256-word storage block. A total of sixteen storage blocks can be addressed via the table. If the number of storage blocks used in a relocated program is less than sixteen, the table entries not used can be made zero. The use of a zero entry alerts the program through the Empty Address (EA) interruption code.

When the prefix is zero, the relocation table is placed at locations 0 and 1. However, a program with prefix zero is normally of supervisory nature and is not relocated. Locations 0 and 1 are permanently protected against data storage and can be loaded only by input operations

2.11.2 Final Data Descriptions

Long data descriptions used in numeric and logical-processing operations, and all data descriptions or instruction half-words used in multiple-word transmission, are stored at the end of the operation. These are the first final and the last final data descriptions. The first final data description is stored at the location specified by the prefix plus two, and the last final data description is stored at the next higher full-word address. Since the prefix address may be 0, final data descriptions may be stored at locations 2 and 3. For this reason, these locations are not used for any other purpose. A second use of these locations is in storing bits 0-15 of the effective address of the instructions STORE EFFECTIVE ADDRESS FIRST and STORE EFFECTIVE ADDRESS LAST. A third use of the location at the prefix address plus two is in storing the instruction half-word or data description of an INTERRUPT INTENTIONALLY instruction. A full word is stored for these instructions, although the information in the right half-word is meaningless when no long data description is used.

During processing operations, the data descriptions are updated and occasionally stored in the final data description locations, since these locations are used as backup locations for internal registers. Either the entire control word or its right half is stored. The stored information usually lags behind the information in the internal registers. When an operation is terminated because of a Data Store (DS), Invalid Data (ID), Unending Sequence (US), or Address Invalid (AD) interruption, the stored information may be useful in diagnosing the program error.

2.11.3 Accumulator

Floating-point operations, and the accumulator load and store operations make use of an implied accumulator location which has the address of the prefix address plus four.

The accumulator occupies a full word. In the floating-point operations, the contents of the word are assumed to have the floating-point format.

2.11.4 Timer

The time clock and interval timer are stored in location 5. These are fetched and updated in the processor, and subsequently returned to storage.

2.11.4.1 Time Clock

A time clock is provided to measure time difference or duration over relatively long periods. It consists of a twenty-bit number which is stepped up by pulses from a 10-cps oscillator. The time clock is updated at the moment when the interval timer is also updated. Since the clock measures time in tenths of seconds, a full cycle is about 29 hours. The value of the clock occupies bits 0-19 of location 5. Each time the oscillator delivers a pulse, the contents of location 5 are read out, incremented by one in position 19, and returned to core storage. The clock runs continually while the processor is under program control, including the time the processor is waiting. When the clock reaches its maximum reading of all ones, the next oscillator pulse sets it to all zeros. No indication is given when the clock recycles to zero.

PROGRAMMING NOTE

The time clock can be used to obtain a time-of-day indication. A known external time is taken as a reference point and the time clock is set to a given constant at that time. The time of day can be obtained by reading the time-clock setting, and converting the amount to hours, minutes, and seconds.

2.11.4.2 Interval Timer

The interval timer is intended to measure elapsed time over relatively short intervals. The timer consists of a 16-bit number which is stepped down by pulses originating from a stable oscillator. The oscillator operates at 1024 (2^{10}) cps, that is, a pulse about every millisecond. It can be set to any value at any time, and an interruption code Time Signal (TS) is stored when the time period has ended. The value of the interval timer occupies bits 48-63 of location 5. Each time the oscillator delivers a pulse, the contents of location 5 are read out, decremented by one in position 63, and returned to core storage. Bits 48-53 show time in seconds. A full cycle is about one minute.

The timer runs whenever the machine is operating, including the time the processor waits. Whenever it goes from one to zero, it stores the interruption code Time Signal (TS). The timer remains at zero until changed by programming.

2.11.5 Interruption Control Words

Two control words are used by the interruption procedure: the store interruption control word and the fetch interruption control word. Both control words are specified and placed in their storage locations by the program which is used to initialize the processing system. The interruption procedure is described in Chapter 3.

2.11.5.1 Store Interruption Control Word

The store interruption control word is placed in location 6 and specifies where the interruption codes are to be stored. When a program alert occurs, the store interruption control word is fetched, used and updated by the processor and returned to storage after each interruption code is stored.

2.11.5.2 Fetch Interruption Control Word

The fetch interruption control word is placed in location 7 and specifies the next interruption code to be fetched from storage. When an interruption occurs while the processor is enabled or becomes enabled, the fetch interruption control word is fetched, used and updated by the processor, and returned to storage after an interruption code is fetched. Interruption codes are always stored before they are fetched.

2.11.6 Backup Locations

Locations 8, 9, and 10 are used as backup locations. Some internal registers are stored in and later retrieved from locations 8 and 9 during the execution of input-output operations. These locations should not be used for general-purpose storage, since their content may be destroyed by input-output operations. Location 10 is used in a similar fashion to store intermediate results of floating-point operations.

2.11.7 Scan Area

Diagnostic information is scanned into storage when a machine check occurs. To provide fault location, information may also be scanned from storage into the processor registers. The storage locations used for scan operations have addresses 16-31. These locations are also used in the initial program-loading procedure.

2.11.8 Program Control Word

Locations 32 and 33 are set aside for the current enabled and the current disabled program control word. When the machine is in the enabled state, reference is made to location 32 for the program control word. In the disabled state, reference is made to location 33 for the word.

The current program control word is kept in internal registers and updated in these registers. For certain instructions, the right half of the program control word is temporarily placed in storage using the appropriate disabled or enabled location as backup location. The control word which is not in use remains unchanged in storage. Program control words are always correctly updated and stored when the machine switches from the disabled state to enabled state or vice versa.

Locations 34-65 contain the initial disabled program control words which are fetched from the interruption byte stream under control of the fetch interruption control word. The control words in these locations are used but not changed by the processor.

The control words in locations 35-63 correspond to individual processor or input-output alerts. Locations 64 and 65 are shared by all input-output alerts of channels 16-127. Location 34 corresponds to interruption Code 2. This code does not represent a machine alert, but may be assigned by the programmer and placed in the interruption stream under program control.

2.11.9 Arithmetic Tables

The processor uses tables in storage for some of its arithmetic operations. The proper argument address for the table is formed during the operation, and depends upon the type of operation to be performed, as well as the value of the operand digits processed. Separate tables are available for hexadecimal and for decimal arithmetic operations.

Locations 72-75 are reserved for a radix conversion table used by the CONVERT operation.

Locations 76-79 are reserved for a quotient search table used by the division operations.

Locations 80-95 are reserved for a quotient prediction table used by floating-point division.

Locations 96-115 are reserved for a decimal multiplication table used by the fixed-point decimal multiplication and division operations. The equivalent of a hexadecimal multiplication table is provided by internal equipment.

2.11.10 Input-Output Area

Locations 124 and following are used for data buffers and control words required in input-output operations, the extent of the area depending upon the number of channels used in the system. A data-buffer location is used by the input-output section in the assembly of storage words from bytes obtained from the external units, or in selecting bytes from storage words in order to send them to external units. Two data-buffer words are used per channel. The control-word locations are used to store control words which control data transfer to and from the buffer area. The regular simplex channels use two data buffers and two control-word locations per channel. On the other hand, the multiplexed channels use one control-word location per channel and do not use any data-word locations.

2.11.10.1 Data-Buffer Locations

Locations 124-127 are used for the first data-buffer word and locations 128-131 are used for the second data-buffer word of channels 8-11.

2.11.10.2 Control-Word Locations

Locations 132-135 contain the auxiliary control words for channels 8-11. Locations 136 and following contain the main control words for channels 8 and following. A total of 120 channels (numbered 8-127) is possible such that the highest location used (address 255) would contain the control word for channel 127. The final control word for the input-output operations executed by a channel will reside in the main control-word location for that channel. The final input-output control word for a channel can be used by the programmer in a manner similar to the use of the final data description. The final input-output control word always refers to the next byte to be read or written by the channel.

2.11.11 Index Words

A total of 255 index words may be specified by a program. These index words are located in storage at the addresses prefix plus one to prefix plus 255. The index words are selected by bits 16-23 of the floating-point arithmetic, accumulator, and control-word instructions. An all-zero field indicates that no indexing is to take place.

2.11.12 Main Core Storage

All special-purpose storage locations operate as general-purpose storage. All locations in main storage are valid word addresses in any operation unless restrictions are imposed by the storage-protection feature.

Address 0 follows address 65,535 when either an implied or an explicit increment operation is performed. No indication will be given when this increment occurs as part of operand or instruction addressing, or in any control-word operation. When fixed-point arithmetic is used for address computation, the Oversized Result indicator will be set.

Addresses from 0 to 65,535 ($2^{16}-1$) are available for addressing main core storage. Not all addresses may be provided in a given installation. If less than the maximum amount of core storage is provided, consecutive addresses starting at location 0 are used. The entire effective address is always used and if it is above the limit of available storage, interruption code Address Invalid (AD) will be stored. If several of the addresses used by an instruction are invalid, only one interruption code will be stored. Following the detection of an invalid address, the instruction address may point either to the next instruction in sequence or to the second or third instruction half-word of the current instruction.

The Address Invalid (AD) check includes all addresses actually used to fetch or store data, including addresses specified in instruction and control words, and addresses generated as an operation proceeds.

Not checked for Address Invalid (AD) are

1) bits 0-7 of the channel address of an input-output instruction,

2) the control-word address of a RELEASE CHANNEL instruction,

3) the addresses of an unsuccessful branch instruction, and

4) a refill address which is not used.

STORAGE ASSIGNMENTS

0-1	Relocation table (prefix zero)
2	First final data description (prefix zero)
3	Last final data description (prefix zero)
4	Accumulator (prefix zero)
5	Time clock and interval timer
6	Store interruption control word
7	Fetch interruption control word
8-10	Backup locations
16-31	Scan area
32	Enabled program control word (PCW)
33	Disabled program control word (PCW)
34	PCW code 2
35	PCW instruction count (IC)
36	PCW time signal (TS)
37	PCW channel not operational (CNO)
38	PCW channel busy (CB)
39	PCW unended sequence (US)
40	PCW address invalid (AD)
41	PCW data store (DS)
42	PCW operation code invalid (OPI)
43	PCW invalid data (ID)
44	PCW intentional interruption (II)
45	PCW input-output alarm (IOA)
46	PCW empty address (EA)
47	PCW maskable indicator (MI)
32 + 2n	PCW channel n end (n=8-15) (CnE)
33 + 2n	PCW channel n special condition (n=8-15) (CnSC)
64	PCW channels 16-127 end (CE)
65	PCW channels 16-127 special end (CSE)
72-75	Radix conversion table
76-79	Quotient search table
80-95	Quotient prediction table
96-115	Decimal multiplication table
124-127	First data-buffer word channels 8-11
128-131	Second data-buffer word channels 8-11
132-135	Auxiliary control word channels 8-11
136-255	Main control word channels 8-127

Prefix +	0-1	Relocation table
	2	First final data description
	3	Last final data description
	4	Accumulator
	n	Index n (n=1-255)

Chapter 3

INSTRUCTION SEQUENCING

Contents

Contents Chapter 3 (continued)

Chapter 3

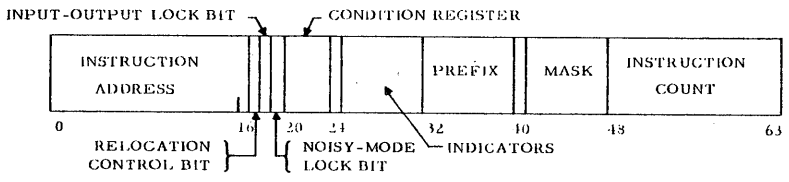
INSTRUCTION SEQUENCING

3.1 NORMAL SEQUENTIAL OPERATION

Normally, the operation of the processor is controlled by instructions taken in sequential order. An instruction is fetched from a core-storage location whose address is specified by the instruction address portion of a program control word. Subsequently, the instruction address is changed to address the next instruction in sequence. Any address modification required by the fetched instruction is performed and the modified instruction is then executed. The process is then repeated, using the new value of the instruction address. Deviations from this normal sequence of instructions may be caused by branching or program interruption.

3.1.1 Program Control Words

Each program or subprogram executed by the processor normally has its own program control word. The program being actively executed by the processor at any given time is called the "current program." Normally, the program control word for this program, called the "current program control word," is kept in internal registers. The other program control words are kept in core storage. Unique storage locations are provided for the program control words for the programs to be executed for each of the causes of program interruption. These programs are executed in the disabled mode, and their program control words are collectively called the "disabled program control words." They are kept in storage locations 34-65. In addition, the final status of the program control word for the current or last previous disabled program is kept in storage location 33. The program control word for the program being executed in the enabled mode, which is normally the basic program being processed by the processor, is kept in storage location 32. Switching of control of the processor from one program control word to another is accomplished by the program-interruption system and the program-switching instructions. A program control word has the format shown below.



3.1.1.1 Instruction Address

Bits 0-16 of a program control word contain the instruction address, which specifies the half-word in core storage from which the first section of the next instruction will be fetched. As each instruction is fetched, the instruction address is changed to address the next instruction in sequence in storage. The instruction address of a program control word may also be changed by branching operations.

3.1.1.2 Instruction Count

Bits 48-63 of a program control word contain the instruction count. This count is decreased by one each time an instruction is fetched. When this count reaches zero, interruption code Instruction Count Signal (ICS) is stored in the interruption stream to cause interruption at the completion of the current instruction if the system is enabled. Further discussion of program interruption can be found in a later section of this chapter. Once the count reaches zero, or if it is initially zero, it is not decremented further, and no more interruptions occur.

PROGRAMMING NOTE

The instruction count is intended principally as an aid to the debugging of programs. It may be used for this purpose in at least three ways. First, by setting the instruction count of the program being debugged to one and then switching to that program, a single instruction will be executed, and the program then interrupted. This allows the subject program to be traced instruction by instruction. Second, the instruction count may be used to break out of any possible infinite loops of instructions in the subject program. Third, the subject program can be interrupted at any desired point. For example, the instruction count can first be set to all ones. The subject program is then run until trouble develops. The instruction count at that time is noted. A number a few less than the complement of this count is then set in the instruction count of the subject program and the program is rerun. It will now be interrupted just before the trouble develops and it may be traced instruction by instruction from that point until the cause of the trouble is discovered.

3.1.1.3 Prefix Address

Bits 32-39 of a program control word contain the prefix for that program. This prefix is used to form the prefix address, which is a full-word address consisting of the prefix followed by eight zeros. The prefix address is therefore restricted to a multiple of 256. The prefix address performs two functions in the processor. First, it controls the protection of core-storage locations. Storing into all locations below the prefix address plus two is prevented, as discussed in the section entitled Address Monitoring, Chapter 2. Second, the implicit storage locations required by some operations are located relative to the prefix address. This allows a separate set of such locations for each program having a separate program control word. These locations are described in detail in the section on Storage Assignment, Chapter 2.

3.1.1.4 Indicators

Bits 25-31 of a program control word contain the indicators associated with the program controlled by that word. These indicators signal the occurrence of various exceptional conditions to the program. When an indicator is masked on and the interruption system is enabled, the program will be interrupted if that indicator is turned on. The indicators are described further in a later section of this chapter.

3.1.1.5 Mask

Bits 41-47 of a program control word contain the mask corresponding to the indicators in bits 25-31. As mentioned above, if a given bit of the mask is a one, then the corresponding indicator can cause a program interruption when the interruption system is enabled. If a bit of the mask is a zero, the corresponding indicator cannot cause an interruption.

3.1.1.6 Condition Register

Bits 20-23 of a program control word contain the condition register for the program controlled by that word. The bits of this register are set to describe the results of each operation. For example, after an addition they indicate whether the sum was positive, negative, or zero. The bits of the condition register can be tested by the operations BRANCH IF ANY and BRANCH IF NONE. The condition register can also be set to correspond to any byte in storage by the operation LOAD CONDITION REGISTER.

3.1.1.7 Relocation Control Bit

Bit 17 of a program control word is the relocation control bit. If this bit is zero, addresses are used directly. If this bit is one, addresses are relocated before being used to address core storage. Relocation was described in Chapter 2.

3.1.1.8 Input-Output Lock Bit

Bit 18 of a program control word is the input-output lock bit. If this bit is zero, input-output operations proceed normally. If this bit is a one, any input-output operation encountered in the program controlled by this word is suppressed, and interruption code Input-Output Alarm (IOA) is stored in the interruption stream. This feature allows all input-output activity to be forbidden to a specific program.

3.1.1.9 Noisy-Mode Control Bit

Bit 19 of a program control word is the noisy-mode control bit. If this bit is zero, any floating-point operations encountered in the current program are executed normally. If this bit is one, floating-point operations are executed in the noisy mode. The noisy mode provides a test of the significance of floating-point results. It is more fully described in connection with the floating-point operations.

PROGRAMMING NOTE

Bits 24 and 40 of a program control word are not used. Bit 24 is used in data descriptions to control indirect addressing. If bit 24 of a program control word is set to zero, then this word may be properly indirectly addressed by a branch instruction to cause a branch to the starting point of the program described by the program control word.

3.1.2 Instruction Loading

The fetching of instructions from storage is controlled by the current program control word. Normally, this word is kept in internal registers. Some operations require these registers for other uses, however. During such operations, all or part of the current program control word is stored in location 32 or 33.

3.1.2.1 Instruction-Loading Sequence

Upon the completion of an instruction, the next instruction must be fetched. If the appropriate program control word is not already contained in the internal registers, this control word is first fetched from storage. The half-word at the storage location addressed by the instruction-address part of the control word is next fetched. This half-word is examined to determine whether the instruction is of the one-, two-, or three-address type. If it is a multiple-address type, the remaining half-words of the instruction are fetched. As each half-word is fetched, any address modification called for is performed, and the instruction-address part of the program control word is incremented to address the next half-word in storage. As the first half-word of any instruction is fetched, the instruction count is decreased by one, unless it has already reached a value of zero. If the instruction is of the three-address type, or for certain two-address and floating-point instructions, the right half-word of the program control word is stored after the entire instruction has been fetched. (This is necessary to make the internal register available for use by some instructions of these types.) The operation called for by the instruction is then executed.

3.1.2.2 Tests for Instruction Validity

As an instruction is fetched, it is checked in a number of ways to determine whether or not it is a valid instruction. The first check is made to see if the instruction length, as determined from the first half-word, agrees with the other instruction half-words fetched from memory. As mentioned in Chapter 2, the last half-word of an instruction should contain a zero in bit position 27 of that half-word. If it does not, it is an invalid instruction, and results in the storage of interruption code Operation Code Invalid (OP) and the suppression of any operation called for by that instruction. The instruction address will have been updated according to the length specified in the first half-word.

If the first address of an input-output instruction is direct, rather than indirect referring to a control word, the instruction is invalid, and interruption code Operation Code Invalid (OP) is stored.

In addition to these checks, each address actually used to reference storage is checked to see that it addresses a location actually provided in the particular installation. If it does not, that is, if the address is above the range of available storage locations, interruption code Address Invalid (AD) is stored in the interruption stream.

3.2 BRANCHING

The sequence in which instructions are executed may be made conditional upon the results produced by previous instructions by means of the branching operations. In these operations, the branch is successful, that is, the instruction address of the current program control word is replaced by the branch address from the instruction, only if the specified condition is met. Otherwise, the sequence of instructions proceeds normally to the next instruction in storage. The condition tested by the branching instructions is the state of specified bits in the condition register.

3.2.1 Condition Register

The condition register occupies bits 20-23 of the current program control word. The four bits of this register are set by most operations to reflect the result of the operation. All four bits are always set as a group; never separately. They remain set until the next operation that sets the condition register.

The types of instructions that set the condition register are listed below. Each bit, 0-3, is set to one by the condition shown; to zero by the absence of that condition. Any bits not used by a particular operation are set to zero. For further details regarding the setting of the condition register reference may be made to the individual instruction descriptions.

Control-Word Instructions

0 Not used
1 Not used
2 Not used
3 Result count zero

Fixed-Point Arithmetic Instructions

0 Result less than zero or last field less than the first
1 Result zero or last field equal to the first
2 Result greater than zero or last field greater than the first
3 Result count zero

Floating-Point Arithmetic Instructions

0 Result less than zero
1 Result zero
2 Result greater than zero
3 Result infinite

Alphameric-Comparison Instructions

0 Last field less than the first
1 Last equal to first
2 Last greater than first
3 Not used

Connective Instructions

0 Operation terminated by match or clash
1 Result zero
2 Operation terminated by exhaustion of first field
3 Operation terminated by exhaustion of last field

Data-Transmission Instructions

0 Operation terminated by match or clash
1 Not used
2 Operation terminated by first count zero
3 Operation terminated by last count zero

Translation Without Carry Instructions

0 Not used
1 Not used
2 Operation terminated by exhaustion of first field
3 Operation terminated by exhaustion of last field

Translation With Carry Instructions

The condition register is set to the value of the last carry byte.

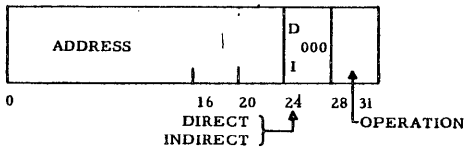
The condition register is not altered by any accumulator, branching, program-switching, or input-output operations. The condition register may be stored for later testing by preserving the program control word.

In addition to the operations listed above, an auxiliary operation, LOAD CONDITION REGISTER, is provided to set the condition register to any desired byte in storage so that this byte may be tested.

LOAD CONDITION REGISTER (LCR)

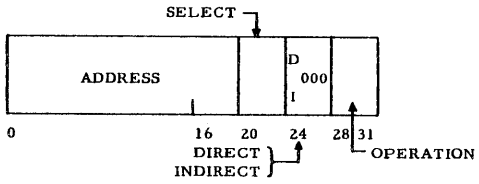
This instruction occupies a single half-word with the format shown below. The byte-address part, bits 0-19, specifies a single four-bit byte in storage. Bits 20-23 are ignored. Bit 24 specifies whether the byte address is direct or indirect. Bit 25 is ignored. Bits 26-31 contain the operation code.

The four-bit byte addressed by bits 0-19 of the effective address is fetched from storage and replaces the previous contents of the condition register.



3.2.2 Branching Operations

The branching instructions occupy a single half-word, with the format shown below. The byte-address part, bits 0-19, represents the branch address. The three low-order bits of this address are ignored. Bits 20-23 of the instruction select the bits of the condition register to be tested. Each bit in this part of the instruction selects the corresponding bit in the condition register. If the bit in the instruction is one, that bit of the condition register is tested. If the bit in the instruction is zero, the corresponding bit in the condition register is not tested. Bit 24 of the instruction determines whether the byte address is direct or indirect. Indirect addressing does not affect the selection of bits to be tested. Bit 25 of the instruction is ignored. Bits 26-31 contain the operation code.



BRANCH IF ANY (BA)

This operation causes the selected bits of the condition register to be tested. If any of the selected bits are on (set to one), the contents of the instruction-address part of the current program control word are replaced by bits 0-16 of the effective address of the instruction. If none of the selected bits are on, control proceeds normally to the next instruction in sequence.

BRANCH IF NONE (BN)

This operation causes the selected bits of the condition register to be tested. If none of the selected bits are on (set to one) the contents of the instruction-address part of the current program control word are replaced by bits 0-16 of the effective address of the instruction. If any of the selected bits are on, control proceeds normally to the next instruction in sequence.

PROGRAMMING NOTE

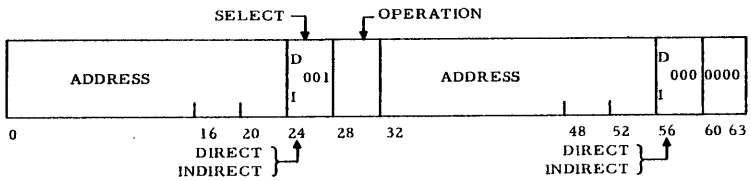
If bits 20-23 of the instruction are all set to zero, then BRANCH IF NONE becomes an unconditional branch instruction and BRANCH IF ANY becomes a "no operation" instruction.

3.2.3 Preservation of Program Control Word

Each of the one-address branching operations described above has a two-address counterpart that preserves the current program control word in a specified storage location. This stored word may later be used to return control to the instruction following the branching operation in normal sequence.

The program control word is preserved in a state pertinent to the next instruction in storage following the branching instruction. That is, the instruction-address part contains the address of the next half-word in storage following the last half-word of the branching instruction, and the instruction count includes the decrementing associated with fetching the branching instruction. If the processor is in the relocation mode, the instruction address is stored in symbolic form.

These operations have the format shown below. The first half-word is identical to the corresponding one-address branching instruction, except that the operation code is slightly different. The byte address in the last half-word, bits 32-51, specifies the location at which the program control word is stored. The four low-order bits of the address are ignored. Bit 56 specifies whether this address is direct or indirect. The limit part of the last half-word is not used. Bits 57, 58, and 60-63 are ignored. Bit 59 is a zero and identifies the last half-word.



BRANCH IF ANY AND PRESERVE PCW (BAP)

This operation is identical to BRANCH IF ANY, using the branch address in the first half-word, except that first the current program control word is stored at the full-word location addressed by bits 0-15 of the effective address of the last half-word of the instruction if the branch is successful. If the branch is not successful, the program control word is not stored.

BRANCH IF NONE AND PRESERVE PCW (BNP)

This operation is identical to BRANCH IF NONE, using the branch address in the first half-word, except that first the current program control word is stored at the full-word location addressed by bits 0-15 of the effective address of the last half-word of the instruction if the branch is successful. If the branch is not successful, the program control word is not stored.

3.3 PROGRAM SWITCHING

A separate program control word is normally used for the main program and for each of the programs or subprograms associated with the different causes for program interruption. Switching between these different programs is controlled by the program interruption system.

3.3.1 Enabled and Disabled Modes

The processor operates in one of two modes: enabled or disabled. In the enabled mode, the sequence of instructions is controlled by the enabled program control word stored in location 32, and the program interruption system is active, so that the occurrence of an interrupting condition stops the execution of the current program and causes a transfer to the disabled mode. This interruption action is described more fully under Program Interruption. In the disabled mode, the sequence of instructions is controlled by one of the disabled program control words, which are stored in locations 34-65, and the program interruption system is disabled. The occurrence of an interrupting condition does not then affect the execution of the current program until the processor is again placed in the enabled mode. In either mode, the occurrence of an interrupting condition causes a code to be stored in the interruption stream.

Usually, any interrupting condition should be responded to promptly. Therefore, a program is normally executed in the enabled mode. When an interruption occurs, the processor is switched to the disabled mode. This serves two purposes. First, it allows a new sequence of instructions to be executed under control of the disabled program control word corresponding to that cause for interruption without disturbing the enabled program. Second, it prevents further interruptions from occurring while the first interruption is being handled.

The interruption-handling routines are often very similar for broad groups of problems. For this reason, the disabled parts of a program can often be standardized, and only the enabled portion need be written anew for each problem. In such cases, the disabled program often takes the form of a "supervisory program," usually with additional features to take care of other common tasks, such as standard input and output routines and aids for program debugging. The enabled portion of the program, perhaps together with some information for the supervisory program, then becomes a "problem program" that is executed under control of the

supervisory program. Only the problem-program portion of the entire program need be written for each problem to be solved. Not all of the supervisory program must be executed in the disabled mode. Lengthier portions or ones requiring external responses may be executed in the enabled mode, but control is normally retained by the disabled portions.

To save time, the supervisory program may perform the function of coordinating the execution of a whole group of problem programs. These problem programs may be "uniprogrammed," that is, executed completely sequentially; or "multiprogrammed," that is, executed concurrently, with two or more problem programs in core storage simultaneously and with control switched from one to another according to their demands on input-output equipment.

Problem programs are very likely to contain mistakes, particularly during the period when they are being checked out. It is therefore important to protect the supervisory program, and in the multiprogrammed case, the other problem programs, from any undesired changes caused by these mistakes. This protection is provided by the relocation and address-monitoring facilities described in Chapter 2. On the other hand, the supervisory program is assumed to be trustworthy and essentially free from mistakes. For this reason, a number of operations are permitted in the disabled mode that are not permitted in the enabled mode. Most important of these is the ability to switch to any of the disabled program control words by means of the INTERRUPT INTENTIONALLY operation.

3.3.2 Requirements for Switching Programs

To switch effectively from one program to another, it is necessary to do more than simply start fetching instructions from a new area in storage. A number of other factors, in addition to the instruction address, are normally unique to a particular program, and therefore must be changed when programs are switched. For convenience, these factors are all grouped together in the program control word. The program interruption system provides the means for placing the processor under control of a different program control word.

The following factors, generally different for each program, determine the state of the processor for executing that program and are therefore included in the program control word--

1. The instruction address and instruction count, which indicate the point to which execution of the program has progressed.

2. The prefix, which identifies the area in storage assigned to this program.

3. The mask, which determines the indicators that can cause interruption.

4. The indicators and condition register, which describe results produced by the program, and upon which later branching operations in the program may depend.

5. The relocation, input-output lock, and noisy-mode control bits, which determine the mode in which the program is run.

One additional factor is important in determining the processing mode for a given program. This is whether the processor is enabled or disabled. This, of course, is also determined by the program control word, but by its location rather than its contents.

3.4 PROGRAM INTERRUPTION

The program interruption system provides the means by which the processor responds appropriately to external signals and to exceptional conditions arising within its own program. When such a signal or exceptional condition is detected, it causes an "interruption code," which is an eight-bit byte coded to identify the cause for interruption, to be stored in an area in storage called the "interruption stream." If the processor is in the enabled mode, an interruption occurs at the completion of the instruction being executed when an interruption code was stored. An interruption consists of a switch to the disabled mode, the examination of a byte in the interruption stream, and the initiation of the program corresponding to the value of that byte and hence to the cause for interruption. When this program has responded suitably to the interruption, it switches back to the enabled program, which continues at the point at which it was interrupted. The elements of the program interruption system and the sequence of actions that take place are described more fully in the following sections.

3.4.1 Storage of Interruption Codes

Interruption codes are stored in the interruption stream. The location and extent of the interruption stream is defined by the "store interruption control word," which is stored in location 6 of core storage. This word has a format similar to an input-output control word, and the storage of interruption codes is done by the input-output mechanism. Of bits 19-31, only bits 19, 27, 28, and 30 are used; the others are ignored. Bit 19, the storage-protection bit, is normally set to one. Bits 27 and 28 must be zero. The address in bits 0-18 is the location at which the next interruption code is to be stored.

Any of several types of conditions can cause an interruption code to be stored. These conditions include such events as

1. an attempt to execute an invalid instruction,

2. suppression of an instruction because of the present machine status,

3. termination of an input-output operation,

4. receipt of an external signal, and

5. occurrence of a specified exceptional data condition.

Immediately upon the detection of such a condition, an interruption code is stored. This may occur at any time during the execution of an instruction, regardless of whether the processor is in the enabled or disabled mode.

The address in bits 0-18 of the store interruption control word is incremented by a one in bit position 18 each time a byte is stored, unless the byte stored has a value of zero or one. The latter cases correspond to a switch back from the disabled to the enabled mode by the execution of an INTERRUPT INTENTIONALLY operation. In these cases, the control word is not incremented, so that the next interruption code is stored immediately following the last code that is neither zero nor one. In incrementing the control word, whenever a word boundary is crossed, the count in bits 48-63 is decreased by one. If the count becomes zero, the chain bit, bit 30, is tested. If it is a one, the control word is refilled from the full-word location addressed by bits 32-47, and further interruption codes are stored according to the new contents of location 6. If the chain bit is zero, the storage-protection bit is turned off. This prevents any further interruption codes from being stored, and would normally be a programming mistake.

If the address in the store interruption control word is invalid, the byte will not be stored, but the invalid address will cause another interrupting condition. The processor thus becomes caught in a loop, continually trying to store bytes.

PROGRAMMING NOTE

Normally, the store interruption control word is refilled with its own initial value. The interruption stream therefore doubles back on itself, and is never exhausted. It is only necessary to provide an area large enough that a given interruption code is serviced before it is written over with a new interruption code. The length of the area required will depend upon the frequency of interrupting conditions and the length of time that the processor may remain in the disabled mode. Normally, two or three words will be sufficient.

3.4.2 Interruption Action

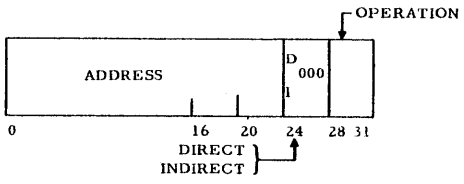
When the processor is in the enabled mode, an interruption occurs at the completion of execution of any instruction during whose execution an interruption code was stored. At this time, the instruction address in the enabled program control word will be that of the succeeding instruction, with the possible exception of Address Invalid and Empty Address interruptions. The interruption occurs even if the store interruption control word has been exhausted and its storage-protection bit turned off, so that the byte is not actually stored. An interruption is also initiated by the execution of an INTERRUPT INTENTIONALLY instruction in either the enabled or disabled mode.

The current program control word is stored in location 32 if the processor is in the enabled mode, or location 33 if it is in the disabled mode. The "fetch interruption control word" is then obtained from location 7. This control word has the same format as the store interruption control word and normally describes the same storage area. The byte addressed by this control word is fetched from storage. If this byte is neither zero nor one, the address of the control word is incremented, and if a word boundary is crossed the count is decremented, just as in the storage of interruption codes, and the processor is placed in the disabled mode. If the count becomes zero, the control word is refilled, regardless of the value of the chain bit. If this byte is zero or one, the control word is not incremented, and the processor is placed in the enabled mode.

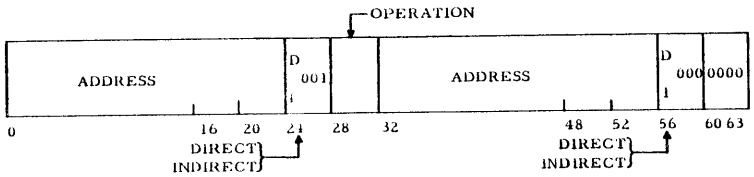
In either case, a new program control word is obtained. If the byte fetched has a value from 0-31, inclusive, the new program control word is obtained from location 32+x, where x is the value of the byte. If the byte fetched has a value of 32 or greater, the new program control word is obtained from location 64 if the byte has an even value, or 65 if the byte has an odd value. Location 64 thus corresponds to a Normal End on any of the multiplex input-output channels, while location 65 corresponds to a Special Condition on these same channels. If the byte fetched does not have a value of one, program execution proceeds using the new program control word. If the byte has a value of one, the processor enters a waiting status. This status is terminated by the occurrence of an interrupting condition.

3.4.3 Program-Switching Instructions

An intentional interruption to effect a switch to a different program control word can be initiated by a program-switching instruction. These instructions have the same format as the branch instructions. The one-address format is shown below.



The one-address instruction has a two-address counterpart that first preserves the current program control word, just as the branch instruction does. The format for this instruction is shown on the next page. The first half-word is identical to the one-address instruction except for bit 27. The last half-word contains the full-word address at which the current program control word is stored. The byte and limit parts of this half-word are not used.



INTERRUPT INTENTIONALLY (INT)

This instruction operates differently in the enabled and disabled modes.

If the processor is in the enabled mode, the full word containing the effective address is stored at the prefix address plus two, the first final control word location. If the processor is in the relocation mode, the effective address is stored in symbolic form. If direct addressing was specified, the first half-word is the instruction itself. The last half-word is unpredictable. If indirect addressing was specified, it is the full-word final data description regardless of bit 31. The interruption code for intentional interruption is then stored in the interruption stream, and an interruption is initiated. This interruption causes a switch to the disabled program controlled by the intentional interruption program control word.

If the program is in the disabled mode, the byte in storage addressed by bits 0-18 of the effective address is stored in the interruption stream. A program interruption is then initiated during which the current program control word is stored at location 33.

INTERRUPT INTENTIONALLY AND PRESERVE PCW (INTP)

The current program control word is stored at the full-word location specified by bits 0-15 of the effective address of the last half-word of the instruction. The operation then proceeds as in INTERRUPT INTENTIONALLY.

PROGRAMMING NOTES

The execution of INTERRUPT INTENTIONALLY, when the processor is in the enabled mode, serves to switch to the disabled mode and the supervisory program by way of the intentional interruption program control word. Thus, all entries to the supervisory program from the problem program are handled by the program that services intentional interruptions.

The execution of INTERRUPT INTENTIONALLY, when the processor is in the disabled mode, can switch to the enabled program or any desired interruption-servicing program, following the servicing of any previously stored interruption codes. If the byte addressed has a value of zero, control will be returned to the enabled program following the servicing of all interruptions. If the byte addressed has a value of one, the processor will go into a waiting status following the servicing of all previous interruptions. It remains in that state until another interruption occurs.

At this point, it is useful to examine the entire process by which interruptions are normally handled. Assume that the processor is executing the enabled program, and that the store and fetch interruption control words point to the same byte in the interruption stream. An interrupting condition then occurs. This immediately causes a code byte to be stored in the interruption stream and the store interruption control word to be advanced to the next byte. At the completion of the current instruction in the enabled program, an interruption will occur. The program control word for the enabled program is stored in location 32. The byte that was previously stored is now fetched from the interruption stream, and the fetch interruption control word is advanced to the next byte. Again both interruption control words point to the same byte.

The disabled program control word indicated by the value of the byte is then fetched, and the program pertinent to this cause for interruption is executed in the disabled mode. At the completion of this program, an INTERRUPT INTENTIONALLY instruction is executed, addressing a byte with a value of zero. This causes a zero byte to be stored in the interruption stream at the point indicated by the store interruption control word and initiates another program interruption. The byte is immediately refetched under control of the fetch interruption control word. This zero byte causes the new program control word to be fetched from location 32, and thereby causes the resumption of the enabled program.

If a second interrupting condition had occurred before the return to the enabled program, this code would have been stored in the interruption stream before the zero code stored by the INTERRUPT INTENTIONALLY instruction. It would then have been fetched before the zero code was fetched, and hence the disabled program for this interruption would be executed before control was returned to the enabled program. Since the interruption control words are not advanced upon storing or fetching a zero code, the INTERRUPT INTENTIONALLY instruction at the end of this second disabled program causes a zero code to be stored on top of the zero code stored by the first disabled program. This insures that

there is never more than one zero code in the interruption stream; and that this one, if it exists, is at the end of the stream.

3.4.4 Indicators

The indicators, located in bits 25-31 of a program control word, signal exceptional conditions that may occur during the execution of the program corresponding to that control word. If an indicator is on, the corresponding bit of the mask, located in bits 41-47 of the program control word, is on, and the processor is in the enabled mode; interruption code Maskable Indicator (MI) is stored in the interruption stream. Indicators can be turned on only by arithmetic operations. The conditions that turn on each of the indicators are listed below and on the next page. Indicators are not turned off automatically, but remain on until turned off by programming.

Data Flag P (PF)

This indicator, bit 25, is turned on when an operand is fetched that has a sign byte in which data flag P (the first bit) is on.

Data Flag Q (QF)

Bit 26 is turned on when an operand is fetched that has a sign byte in which data flag Q (the second bit) is on.

Data Flag R (RF)

This indicator, bit 27, is turned on when an operand is fetched that has a sign byte in which data flag R (the third bit) is on.

Generated Extremum Positive (GEP)

Bit 28 is turned on when the extremum bit of a floating-point result is generated as part of the arithmetic process and the exponent sign is positive, except when the extremum bit is generated by division by zero. The indicator is not turned on when the extremum bit is propagated as a result of inspecting the operand extremum bits.

Generated Extremum Negative (GEN)

Bit 29 is turned on when the extremum bit of a floating-point result is generated as part of the arithmetic process and the exponent sign is negative. The indicator is not turned on when the extremum bit is propagated as a result of inspecting the operand extremum bits.

Oversized Result (OR)

This indicator, bit 30, is turned on if the size of the significant result of a fixed-point or unnormalized floating-point arithmetic operation exceeds the result-field size. The indicator is not turned on when high-order zero digits only are left outside the result field. This indicator is also turned on if a negative result occurs when an unsigned result is specified, or if the restrictions on the field sizes of a FRACTIONAL MULTIPLY instruction are not met.

Zero Divisor (ZD)

Bit 31 is turned on if the divisor is zero. The division operation is then terminated.

PROGRAMMING NOTE

The servicing of the indicators will normally be different for each problem program. Thus, it is suggested that the routines to service them be part of each problem program. When the supervisory program found interruption code MI in the interruption stream, it would "and" the mask and the indicators of the enabled program to determine the indicator that caused the interruption, placing the result in a location accessible to the enabled program. It would then turn off, in the enabled program control word, the indicators that caused the interruption and would return control to the interruption-servicing portion of the problem program.

3.4.5 Interruption Codes

The conditions that cause each interruption code to be stored are listed below. Each code occupies an eight-bit byte. The number of each code in the list below represents the binary value of the bit pattern that is stored in this byte. The codes thus range from 0-255. The first sixteen codes, 0-15, are used for processing interruptions. The remaining codes, 16-255, are used for input-output channel interruptions. Each channel requires two codes, therefore a maximum of 120 channels can be accommodated.

TABLE--COMPUTER INTERRUPTIONS

Enabled Program (EP)

Code 0 can be stored only by the execution of an INTERRUPT INTENTIONALLY instruction in the disabled mode. When fetched from the interruption stream, it causes a return to the enabled program.

Programmed Wait (PW)

Code 1 can be stored only by the execution of an INTERRUPT INTENTIONALLY instruction in the disabled mode. When fetched from the interruption stream, it causes the processor to enter a waiting status. This status is terminated by the occurrence of an interrupting condition. At that time, an interruption occurs in the normal way.

Unassigned

Code 2 is not assigned, and is available for use by the programmer.

Instruction Count Signal (ICS)

Code 3 is stored when the instruction count in the current program control word is reduced to zero during the loading of an instruction.

Time Signal (TS)

This is Code 4 and is stored when the value of the interval timer goes from one to zero.

Channel Not Operational (CNO)

Code 5 is stored when an input-output instruction refers to a channel that is not provided in the system or to which no unit is connected, or addresses a unit that is not in an operational state.

Channel Busy (CB)

This code, Code 6, is stored when an input-output instruction refers to a channel that is busy executing a previous instruction.

Unended Sequence (US)

This is Code 7 and is stored when any operation is not completed in one second, due for example, to an unended sequence of refill addresses or indirect-addressing references. The operation is then terminated.

Address Invalid (AD)

Code 8 is stored when an operation (including the fetching of an instruction) attempts to make a reference to a core-storage location not provided in the system, except for the transmission of data in an input-output operation or the chaining of input-output control words. These latter operations cause a Channel n Special Condition interruption code byte to be stored. Address Invalid (AD) is also stored when, in the relocated mode, a data field, record, multiple field or multiple word crosses a block boundary, or the instruction address is incremented across a block boundary. When an invalid address is detected during the loading of an instruction, the operation is terminated immediately. At this time the instruction address of the program control word may not have been incremented for all half-words of the instruction.

Data Store (DS)

Code 9 is stored when an operation attempts to store data at a protected address. This protection is described in more detail in the section entitled Address Monitoring, Chapter 2.

Operation Code Invalid (OP)

This code, Code 10, is stored when an attempt is made to execute an instruction in which the instruction length specified in the first half-word disagrees with the code in the other half-words. The instruction is suppressed. This code is also stored if the first address of an input-output instruction is direct.

Invalid Data (IVD)

Code 11 is stored when a decimal arithmetic operation encounters a byte that does not contain a valid decimal digit. The operation is terminated.

Intentional Interruption (II)

Code 12 is stored when an INTERRUPT INTENTIONALLY operation is executed in the enabled mode.

Input-Output Alarm (IOA)

This is Code 13 and is stored when an attempt is made to execute an input-output operation when the current program control word has a one in bit position 18, the input-output lock bit. The operation is suppressed.

Empty Address (EA)

This code, Code 14, is stored when, during the translation of a symbolic address into an actual address in the relocation mode, an all-zero origin is obtained from the relocation table. When an empty address is detected during the loading of an instruction, the operation is terminated immediately. At this time the instruction address of the program control word may not have been incremented for all half-words of the instruction.

Maskable Indicator (MI)

Code 15 is stored when an indicator whose mask bit is one is turned on if the processor is in the enabled mode.

TABLE--INPUT-OUTPUT CHANNEL INTERRUPTIONS

For each input-output channel there are two interruption codes: End and Special Condition. These occupy a pair of codes, 2n and 2n + 1; where n is the channel number, from 8 to a maximum of 127. Thus, for example, the interruption code is 33 for a Special Condition on channel 16. The multiplex channels, numbered from 16 up to a maximum of 127, have only a single program control word for End and a single program control word for Special Condition, although distinct interruption codes are stored. Thus, when an interruption occurs from one of these channels, the byte in the interruption stream must be examined to determine the channel causing the interruption.

Channel n End (CnE)

Code 2n is stored when an input-output operation using channel n is terminated, provided that the suppress normal termination bit in the channel control word is zero and that the unit detected no special condition during the operation.

Channel n Special Condition (CnSC)

This is Code 2n + 1 and is stored when a special condition is detected on channel n. Such special conditions include the receipt of an Attention signal from a unit connected to that channel, the detection of an unusual condition by the external unit during the execution of an operation on that channel, and the detection by the processor of a program error associated with that channel. These conditions may be distinguished by examining the status bits in the control word for the channel and the sense information available from the adapter.

3.4.6 Effect of Alerts on Current Instruction

The storage of an interruption code or the turning on of an indicator constitutes an alert. As noted earlier, an alert may or may not cause an interruption following the completion or termination of the instruction currently being executed, depending upon the mode in which the processor is operating and the value of the mask. The different alerts are independent, that is, the occurrence of one alert does not affect other alerts. An alert may, however, cause the execution of the instruction to be terminated, thereby preventing the detection of a condition that would have resulted in another alert. Each cause for an alert actuates a single indicator or stores a single interruption code. Furthermore, each alert results from a single cause. It is possible, however, for more than one alert to occur during the execution of a single instruction, due to independent causes.

In general, the execution of the current instruction will be completed in the event of an alert that--

1. It is not caused by the current instruction. The Time Signal alert and the various channel alerts are of this type.

2. Is only a warning of a condition that is not necessarily an error. The data flag indicators are of this type.

3. Occurred because of the result of a completed execution. The Instruction Count Signal, Intentional Interruption, Generated Extremum Positive, Generated Extremum Negative, and Oversized Result alerts are of this type.

In general, the execution of the current instruction will be suppressed in the event of an alert which indicates that--

1. An operation is not properly defined or cannot be executed because of the present state of the machine. Channel Not Operational, Channel Busy, Unended Sequence, Address Invalid, Empty Address, and Operation Code Invalid alerts are of this type.

2. Completion of the execution might destroy information. Data Store and Input-Output Alarm alerts are of this type.

In some cases, conditions that would normally cause an alert which would suppress the execution of an instruction are not detected until the execution has progressed to a point where the instruction can no longer be completely suppressed. In these cases, the execution of the instruction

is terminated and the alert given as soon as the condition is detected. This is the situation in the following cases--

1. An Invalid Data alert.

2. An Address Invalid or Empty Address alert in operations using long data descriptions when the address results from stepping or refilling a valid address.

3. A Data Store alert when a protected address is encountered due to refilling.

Chapter 5

FLOATING-POINT ARITHMETIC

Contents

Chapter 5

FLOATING-POINT ARITHMETIC

5.1 GENERAL DESCRIPTION

A floating-point number consists of a signed exponent $\pm E$ and a signed fraction $\pm F$. The quantity expressed by this number is the product of the fraction and the radix 16 raised to the power of the signed exponent, or $\pm F \cdot 16^{\pm E}$. Both exponent and fraction are expressed in four-bit bytes representing hexadecimal digits. E is an integer in the range $-255 \leq E \leq 255$ and F is either an eight- or twelve-digit fraction with a hexadecimal point to the left of the high-order digit.

There are basically only four floating-point instructions: ADD, RESET ADD, MULTIPLY, and DIVIDE. These, together with LOAD ACCUMULATOR and STORE ACCUMULATOR, are all one-address instructions and are the only instructions to make use of an accumulator at an implied address. The accumulator is located at the address prefix +4. Its contents have the same format as any other storage locations, and may be used as an addressed operand in any instruction. By locating the accumulator relative to the program prefix, a separate accumulator for every program is provided. This obviates clearing and storing the accumulator contents in multiprogramming situations when passing from one program to another.

By means of modifiers, the basic instructions may be augmented to furnish a versatile floating-point performance. Provision is made for direct or indirect addressing, indexing, modifying the sign of the addressed operand, normalized or unnormalized operation, and the choice of eight or twelve hexadecimal digits of precision.

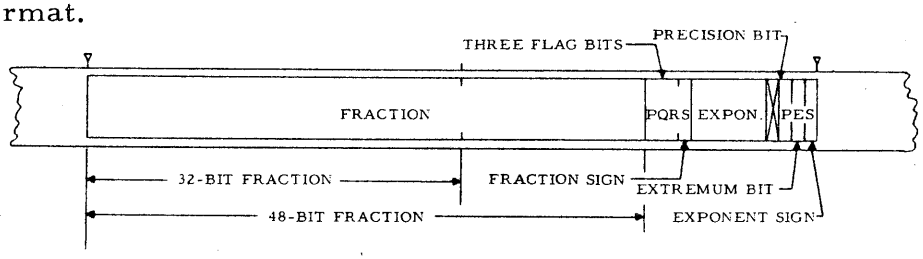
The fractions of all arithmetic results contain the standard eight or twelve digits, accordingly as the minimum of the two input precisions is eight or twelve digits. Not all of these digits need be significant; furthermore, as a result of calculation, the number of significant digits may be reduced. To simplify significance studies, a mode of operation called "noisy mode" is provided, by which standard results are altered in a specific manner. By processing a program, both in standard and in noisy mode, an estimate of the significance of the results may be obtained.

Floating-point numbers cover a range between the positive and negative values of the fraction having the maximum exponent. Since the exponent range is finite, a discontinuity exists between the positive and negative values of the fraction having the minimum exponent. Included in this range is zero. An extremum bit has been included in the exponent field to provide straightforward interpretation of data which exceed the exponent range or fall within the range of discontinuity.

The multiple-field feature may be used with all four floating-point instructions. By this means, a series of floating-point operations on operands stored in successive storage locations may be performed by one instruction which refers to a long data description.

5.1.1 Data Format

Floating-point numbers are represented by the following full-word format.



The fraction field occupies the first 48 bits of the data word. The fraction may be either of twelve hexadecimal digit precision, in which case it occupies the full field; or of eight-digit precision, in which case bits 32-47 will not influence, although in some circumstances they may be altered by, floating-point operations. The precision of the result fraction of a floating-point operation is the minimum of the two input precisions.

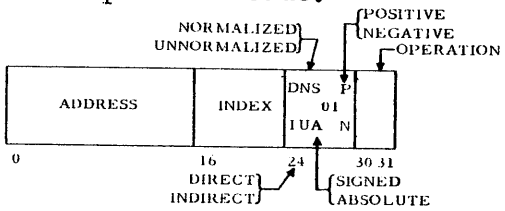
Bits 48-51 contain the fraction sign byte, consisting of three data flags P, Q, and R, and the fraction sign bit. The fraction is positive or negative accordingly as bit 51 is zero or one. Eight exponent bits occupy bits 52-59. Floating-point numbers whose magnitude ranges between 16^{255} and 16^{-256}, or approximately 10^{307} and 10^{-308}, may be represented to a precision of twelve hexadecimal digits.

The exponent sign byte occupies bits 60-63. Bit 60 is unassigned. Its value is ignored by the central processor. Bit 61, the precision bit, indicates the fractional precision, which is eight or twelve digits accordingly as this bit is zero or one. The extremum bit, indicative of an exponent outside the normal range, is assigned to bit 62. The exponent sign is represented by bit 63 in the same manner as bit 51 represents the fraction sign.

The contents of the accumulator have exactly the same format as data in any other storage location.

5.1.2 Instruction Format

All floating-point instructions are of the single address, half-word form. This single address, contained in bits 0-15, specifies one operand of the arithmetic operation to be performed. The second operand is always in the accumulator. Bits 16-23 specify the index address. The address field of the index register specified is added to the address field of the instruction to form an effective address (see Chapter 2 for full details). A zero index field indicates that the instruction is not indexed. Finally, bits 24-31 contain the operation code.



Bits 27 and 28 of the operation code contain the fixed code 01 to indicate that the instruction belongs to the floating-point class. Bit 24 is zero or one accordingly as the operand is directly or indirectly addressed. Bit 25 distinguishes between normalized (0) and unnormalized (1) modes of operation. The sign of the addressed fraction is utilized or ignored accordingly as bit 26 is zero or one. In either event the effective sign of the fraction is determined in conjunction with bit 29, the sign-modifier bit, as described on the next page. Finally, bits 30 and 31 specify one of the four basic floating-point instructions.

5.1.3 Sign Control

The sign of a fraction from storage is modified by bits 26 and 29 to form an effective sign as follows:

Bit 26	Bit 29	Modification
0	0	Retain the sign (bit 51) as it comes from storage
0	1	Invert the sign before operation
1	0	Impose a plus sign
1	1	Impose a minus sign

If bit 26 is zero, the original sign of the fraction is preserved or inverted accordingly as bit 29 is zero or one. Whenever bit 26 is one, bit 29 of the instruction replaces bit 51 of the fraction as its sign.

In no case is the sign of the fraction altered in core storage.

5.1.4 Normalization

A quantity can be represented with the greatest precision by a floating-point number of given fraction length when that number is normalized. A normalized floating-point number always has a nonzero, high-order fraction digit. If one or more high-order fraction digits are zero, the number is said to be in unnormalized form. The process of normalization consists of shifting the fraction until the high-order digit is nonzero and altering the exponent by the amount of the shift.

Bit 25 in the operation code field of the instruction format is zero or one accordingly as the operation is to be performed in the normalized or unnormalized mode.

When normalized operation is specified, the operands need not be normalized, but the result of the operation will be normalized, except when it has a zero fraction or its exponent lies outside the normal range. In these circumstances, which are characterized by the extremum bit of the result being one, normalization is suppressed.

When unnormalized operation is specified, the result fraction remains as it is without a normalization cycle. High-order zeros in the fraction are not eliminated. If a fraction overflow bit is produced, it is lost and the Oversized Result (OR) indicator is turned on.

A normalized fraction has a magnitude in the range $1 > F \geq 1/16$. The magnitude of an unnormalized fraction lies in the range $1 > F \geq 0$. Since hexadecimal numbers only are considered in floating-point operations, all shifts are an integral number of hexadecimal digit positions, i.e., all shifts are multiples of four bit positions.

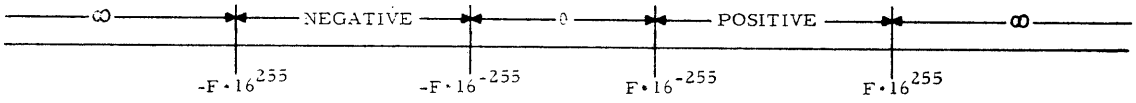
5.1.5 Data Flag Bits

It is sometimes desirable to mark certain data to indicate that special handling is required. In some circumstances it may be necessary to attach more than one mark to each data word and to treat the marks selectively. The flag bits P, Q, and R in the fraction sign byte permit such indicative marking. These flags, if present in a data word brought from storage, set corresponding indicators which permit selective action when the flag bits are encountered.

Flag bits present in an operand are not propagated into the result, i.e., the accumulator flag bits are always zero upon the completion of a floating-point instruction.

5.1.6 Numbers Outside the Normal Exponent Range

A number outside the normal exponent range is characterized by its extremum bit (bit 62) having the value one. In the interests of simplicity and consistency, all numbers with an extremum bit of one and a negative exponent sign are treated by the machine as zero. All numbers with an extremum bit of one and a positive exponent sign are treated by the machine as infinite. Such numbers will be referred to as zero and infinity, respectively. In both cases, the fraction and the fraction sign are ignored. Should a zero fraction be generated during an operation, the extremum bit is set to one and the exponent sign is set negative. No normalization of the fraction occurs.



Should the result of any floating-point instruction, both of whose operands are in the normal exponent range, lie within the range $-F \cdot 16^{-255}$ through $F \cdot 16^{-255}$ it is made zero by turning on the extremum bit. Similarly, if the result of such an instruction is greater than $+F \cdot 16^{255}$ or less than $-F \cdot 16^{255}$, it is made infinity by turning on the extremum bit. The programmer is alerted by means of the GEN or GEP indicator, respectively.

The table below defines the result of floating-point operations for every combination of ranges of the operands. Operands within the normal exponent range are designated by A or S, depending on whether they were in the accumulator or in storage at the beginning of the operation.

ADD

Storage \ Acc	∞	A	0
∞	Acc*	Sto	Sto
S	Acc	$A \pm S$	Sto
0	Acc	Acc	Acc

MULTIPLY

Storage \ Acc	∞	A	0
∞	Acc	Sto	Sto*
S	Acc	A · S	Acc
0	Acc*	Sto	Acc

DIVIDE

Storage \ Acc	∞	A	0
∞	Acc*	0	Acc
S	Acc	A/S	Acc
0	Acc	∞	∞*

If either or both operands are outside the normal exponent range, then, in the case of ADD or MULTIPLY, either the accumulator will be unchanged or else the addressed operand from storage will replace the contents of the accumulator. In DIVIDE, the accumulator will either be unchanged, or else will have its extremum bit and exponent sign adjusted to reflect the appropriate extremum condition.

The starred (*) cases above represent the most conspicuous or worst cases, since the result is mathematically ambiguous ($\infty - \infty$, $\infty \times 0$, ∞ / ∞, or $0/0$).

In the above cases, if either of the operands and the result have an extremum condition, the extremum is "propagated" and no indicator alert is made that such propagation occurred, although the corresponding bit in the condition register will be on. On the other hand, the result of $A \pm S$, $A \cdot S$, or A/S may be a normally representable number, or else an extremum condition may be "generated" and the extremum bit set to one. An indication of such generation is made separately for a positive and a negative exponent sign, these situations being known generally as exponent overflow or exponent underflow, respectively. In most cases, overflow represents an error in analysis or scaling, although on occasions a programmer

will wish to propagate it. The overflow condition is indicated by the Generated Extremum Positive (GEP) indicator. On the other hand, exponent underflow is expected and the programmer will usually wish to treat it as a zero. The underflow condition is indicated by the Generated Extremum Negative (GEN) indicator. In the case of a result with an intermediate exponent overflow that is subsequently removed by postnormalization, the result will be in the normal exponent range, and the GEP indicator will not be turned on.

5.1.7 Noisy Mode

The normalization process used with floating-point operations introduces zeros into the lower order fraction bits whenever left-shifting occurs. This procedure may result in a loss of significance during the course of a program. Assistance in the study of such effects is provided by means of the "noisy mode" of operation. Noisy mode provides for the introduction of hexadecimal fifteen rather than zero in the low-order digit positions during each hexadecimal left-shift associated with normalization. Extra dividend digits required during a division are also filled noisily. Processing a program both in standard and in noisy mode provides an estimate for the study of significance loss as a result of fraction truncation.

The choice of standard or noisy mode is specified by the setting of the noisy-mode control bit in the control word of the program. Standard or noisy mode is called for accordingly as this bit is zero or one. Noisy mode influences normalized operations only; unnormalized operations are unaffected.

5.1.8 Floating-Point Conditions

The condition register is affected by the result of a floating-point operation in the following manner.

Condition of arithmetic result	Condition bit set
In the normal exponent range and less than zero	0
Extremum negative (result zero)	1
In the normal exponent range and greater than zero	2
Extremum positive (result infinite)	3

These conditions are mutually exclusive and collectively exhaustive. After every floating-point instruction, there will always be exactly one nonzero bit in the condition register.

5.1.9 Floating-Point Indicators

Any of the indicators may be affected as a result of a floating-point operation. Indicators are not turned off automatically, but remain on until turned off by programming. The conditions that turn on each of the indicators are listed below.

Data Flag P (PF)

Bit 25 is turned on when an operand is fetched that has a fraction sign byte in which data flag P (the first bit) is on.

Data Flag Q (QF)

This indicator, bit 26, is turned on when an operand is fetched that has a fraction sign byte in which data flag Q (the second bit) is on.

Data Flag R (RF)

Bit 27 is turned on when an operand is fetched that has a fraction sign byte in which data flag R (the third bit) is on.

Generated Extremum Positive (GEP) (exponent overflow)

Bit 28 is turned on when an extremum positive result is generated by a floating-point operation. The indicator is not turned on when the extremum is propagated as a result of inspecting the operand extremum bits.

Generated Extremum Negative (GEN) (exponent underflow)

Bit 29 is turned on when an extremum negative result is generated by a floating-point operation. The indicator is not turned on when the extremum is propagated as a result of inspecting the operand extremum bits.

Oversized Result (OR)

This indicator, bit 30, is used only in unnormalized operation, when it is turned on for fraction overflow in ADD operations, or in DIVIDE operations if the dividend fraction is not less than the divisor fraction.

Zero Divisor (ZD)

Bit 31 is set when the divisor in a DIVIDE operation is zero. The division operation is then terminated. At the time of termination, the quotient will be in the extremum positive condition. In spite of this, the GEP indicator is not turned on. The ZD and GEP indicators are mutually exclusive.

5.2 OPERATIONS

There are only four floating-point operations; namely, ADD, RESET ADD, MULTIPLY, and DIVIDE.

ADD (A)

The addition of floating-point numbers consists of an exponent comparison and a fraction addition. The exponent of the addressed operand is subtracted from the exponent of the operand in the accumulator. The fraction of the number with the algebraically smaller exponent is offset right a number of hexadecimal digits equal to the exponent difference. If the exponent difference is greater than or equal to 12 (for 48-bit precision), or greater than or equal to 8 (for 32-bit precision) no addition takes place, and the number with the greater exponent is treated as the sum. These operations are equivalent to right-shifting the fraction with the smaller exponent until the exponents agree, truncating it to match the fraction with the larger exponent, and then adding.

The fraction sign of the addressed operand is modified by the sign modifiers prior to the addition. A 12- or 8-digit addition takes place accordingly as 48- or 32-bit precision is specified by the minimum of the two operand precisions. The larger of the two exponents is used as the exponent of the sum. For unnormalized operations, the addition is now complete. If there is an overflow digit, it does not enter the accumulator, but turns on the Oversized Result (OR) indicator.

For normalized operations, the entire sum fraction together with overflow is shifted to form a normalized fraction, and the exponent is adjusted accordingly. The normalization does not take place for a zero fraction. Instead, the GEN indicator is turned on. In the noisy mode, any left-shift during normalization will cause hexadecimal fifteens rather than zeros to fill the vacated low-order digits.

The exceptional cases, when one or both of the operands is outside the normal exponent range, are summarized in section 5.1.6. To recapitulate, if the accumulator operand is infinity, or the operand from storage is zero, the contents of the accumulator are accepted as the sum or difference. For the remaining situations, zero accumulator and nonzero storage operand, or infinite storage operand and noninfinite accumulator, a RESET ADD operation is performed.

All the indicators except Zero Divisor (ZD) may be affected by a floating-point addition. However, the Oversized Result (OR) indicator can only be turned on for an unnormalized ADD operation in which a fraction overflow occurs.

PROGRAMMING NOTE

Floating-point comparisons may be simulated by means of a subtraction, after which the result of the comparison will be reflected in the condition register. No account is taken of the sign of the result if it is in the extremum positive or extremum negative range.

RESET ADD (RADD)

The RESET ADD instruction is equivalent to an ADD instruction for which the operand in the accumulator has a twelve-digit precision and is always in the extremum negative, or zero, condition. If the addressed operand has a positive extremum, it is simply loaded into the accumulator without any modifications; if the operand is zero, the zero condition is propagated into the accumulator, and no loading takes place. Finally, if the operand is in the normal exponent range, its sign is modified under the control of bits 26 and 29 of the instruction word before loading. After loading it is normalized, if normalization is specified. The noisy mode operates for RESET ADD instruction by filling with fifteens the low-order digits vacated by left-shifts.

The LOAD accumulator instruction is closely related to RESET ADD, but the latter has more versatility as a result of the various modifiers, such as sign manipulation and normalization. The price of this versatility is a longer instruction execution time. RESET ADD also differs from LOAD in that the former leaves the accumulator flag bits P, Q, and R set to zero, after setting flag indicators, if any.

The indicators affected by RESET ADD are the data flag indicators PF, QF, and RF, and the Generated Extremum Negative (GEN) indicator. GEN can only be turned on if the addressed operand is a new data word with zero fraction and normal-range exponent.

PROGRAMMING NOTE

The LOAD ACCUMULATOR and STORE ACCUMULATOR instructions result in simple transmissions of data between the effective address in storage and the accumulator location of prefix +4. If the data in storage is already in the form desired, and no modifications such as sign manipulation or normalization are necessary, then a LOAD instruction is faster and more direct than a RESET ADD. If it is required to modify data in the accumulator before storing the data, a RESET ADD instruction addressing the accumulator itself is an effective means of doing so.

MULTIPLY (M)

The operand specified by the effective address, the multiplicand, is multiplied by the operand in the accumulator, the multiplier. The exponent, fraction, and sign of the product replace the original contents of the accumulator.

Multiplication of floating-point numbers consists of an exponent addition and a fraction multiplication. The sum of the exponents is used as the exponent of the unnormalized intermediate result. Depending on the precision specified, two 8-digit or two 12-digit fractions are multiplied to form an 8- or 12-digit product, respectively. This product is normally made up of the high-order digits only of the double-length product. For normalized operation, however, if the highest order digit is zero, it is disregarded, and the product is filled out by taking in the highest of the low-order product digits. The remaining low-order digits of the complete double-length product are not preserved even if the normalization of the product requires more than one left-shift. The product sign is determined by the rules of algebra, using the accumulator sign and the effective sign of the addressed operand.

If a normalized product is required, the intermediate product fraction is postnormalized and the exponent adjusted accordingly. When noisy mode is specified, low-order digits are filled with hexadecimal fifteens in the usual manner. For unnormalized operation, no further action is taken on the intermediate product.

The exceptional cases, when one or both of the operands are outside the normal exponent range, are summarized in section 5.1.6. If the accumulator operand is infinity, or if it is zero and the operand from storage is noninfinite, the contents of the accumulator are accepted as the product. For the remaining situations, infinite storage operand and noninfinite accumulator operand, or zero storage operand and normal-range accumulator operand, the operand from storage is brought into the accumulator.

The indicators affected by the MULTIPLY operation are the data flag indicators PF, QF, and RF, and the generated extremum indicators, GEP and GEN.

DIVIDE (D)

The operand in the accumulator, the dividend, is divided by the operand from storage, the divisor. The sign, fraction, and exponent of the quotient replace the original contents of the accumulator. No remainder is retained by this operation.

In the normalized mode, the DIVIDE instruction operates in such a manner as to utilize all the digits of the dividend, regardless of the relative magnitude of the dividend and divisor fractions. If relative shifting of the fractions is required, the exponent quotient is adjusted accordingly. The final quotient is always in normalized form.

In the noisy mode, additional dividend digits required as the division progresses are supplied as hexadecimal fifteens instead of zeros.

If the dividend fraction in an unnormalized DIVIDE is not less than the divisor fraction, the Oversized Result (OR) indicator is turned on, and the operation is terminated. If the divisor fraction is the greater, the division proceeds as in the normalized case, except that leading zeros in the quotients are not suppressed, and the dividend is not offset left.

In all cases the quotient fraction is truncated after the appropriate number of digits (eight or twelve depending upon the precision specified) are obtained. No rounding of the quotient fraction takes place.

The quotient exponent is the difference of the dividend and divisor exponents. If the quotient exponent overflows or underflows, it will turn on the GEP or GEN indicator, respectively. The sign of the quotient is determined by the rules of algebra, using the accumulator sign and the effective sign of the addressed operand.

The exception cases, when one or both the operands are outside the normal exponent range, are summarized in section 5. 1. 6. In no case is the accumulator fraction disturbed. If the dividend is infinite, or the divisor is zero, the quotient will be in the extremum positive condition. A zero divisor also causes the Zero Divisor (ZD) indicator to be turned on. In this case, the GEP indicator is not turned on, since the operation of the two indicators is mutually exclusive. A zero dividend and nonzero divisor will put the quotient in the extremum negative condition.

All the indicators are affected by the DIVIDE operation. The conditions for turning on the various indicators have been mentioned above.

PROGRAMMING EXAMPLE

The example below exhibits the quotients obtained when the four-digit dividend $0001, 0100, 1000, 0111 \times 16^{0}$ is divided by the four-digit divisor $0011, 0101, 0010, 0110 \times 16^{0}$, using the modifications indicated.

Modification	Quotient obtained
Unnormalized	$0000, 0110, 0010, 1101 \times 16^{0}$
Normalized	$0110, 0010, 1101, 1111 \times 16^{-1}$
Normalized, noisy mode	$0110, 0010, 1110, 0100 \times 16^{-1}$

Chapter 7

INPUT-OUTPUT OPERATIONS

Contents

Contents Chapter 7 (continued)

Chapter 7

INPUT-OUTPUT OPERATIONS

7.1 GENERAL DESCRIPTION

Data are recorded on and read from external documents by input-output units. Operation of these units is controlled by adapters which may be designed to control a single input-output unit, several units of the same type, or several different units. The facilities required in the central-processing unit for the connection of an adapter are called "channels." A channel initiates input-output operations, controls the flow of information between an adapter and storage, and alerts the program when an input-output operation is completed.

The design of an adapter depends on the units which it controls. The interface between an adapter and the processor, however, is standard for all types of adapters; no modification of the processor is necessary for the connection of any adapter. All adapters and channels are controlled by the same set of instructions and control words. The START CHANNEL instruction initiates an operation or a sequence of operations which uses a single channel. The instruction specifies the channel to be used and the first control word in a chain of control words. Each control word specifies a storage area and the operation to be executed by the channel. Thus, a chain of control words may specify several operations, each using one or more storage areas.

When an input-output instruction is given, the processor determines whether the channel is available. If not, an alert is given before the program is resumed. If the channel is available, a channel control word, which is used to direct the operation, is formed from the first control word in the chain, and the program is resumed. No other instructions are required for the execution of the input-output operation. Thus, it is possible for data processing to proceed simultaneously with several input-output operations. The only effect of overlapped input-output operations on the program is increased execution time.

All adapters operate serially, sending or receiving one 8-bit byte at a time. A ninth bit, the parity bit, is provided with each byte transmitted between an adapter and the processor to check the transmission. The channel assembles eight bytes into a storage location during an input operation. During an output operation, words from storage are disassembled into eight bytes.

Data recorded on an external document may be divided into "blocks." The length of a block depends on the document, e. g., a block may be a card, a line of printing, or the information recorded between two consecutive gaps on a tape. An input-output operation is always terminated at the end of the block or when all the storage areas specified for the operation are exhausted, whichever occurs earlier.

When an operation is terminated, a test is made to determine if any "special condition" occurred during its execution. If a special condition is detected, an alert is given and the sequence of operations is not continued. If no special condition is detected, an alert may or may not be given, and the next operation in the sequence, if any, is initiated. When a sequence of input-output operations is terminated, the channel control word is found in a fixed location. Three status bits in the channel control word indicate the conditions causing termination.

7.2 CHANNELS

7.2.1 Number of Channels

Three types of channels may be provided in the processor: multiplex, regular simplex, and fast simplex channels. One multiplex channel is provided in every processor. It can be connected to a single adapter, or it can, without modification, accommodate multiplexing facilities which can provide up to 112 channels for the attachment of adapters. Any number of these adapters can be operated simultaneously, provided the capacity of the multiplex channel is not exceeded. Simplex channels are suitable only for the connection of a single adapter. The processor is provided with either two to four regular simplex channels or with four to eight fast simplex channels. Fast channels differ from regular channels only in the maximum data rate which can be accommodated. All three types of channels are controlled by the same instructions and control words.

7.2.2 Channel Capacity

Adapters connected to the processor are serviced on a priority basis. The lower the channel address, the higher the priority. An adapter connected to the multiplex channel or a group of adapters connected to a multiplexer attached to the multiplex channel are assigned lower priority than adapters connected to the simplex channels.

The capacity of a channel, i.e., the maximum rate of information transmission, is determined by the number of channels of higher priority in operation and the byte rates of the adapters connected to them. A limitation is imposed on both the average byte rate and the maximum service time of the adapters in operation. (Average byte rate and maximum service time are defined in an appendix to this manual.) The table below gives the number of adapters that can be operated simultaneously on a regular simplex channel if none of these has an average byte rate exceeding the specified value or a maximum service time less than the specified value.

Average byte rate (Bytes per second)	Maximum service time (microseconds)	Number of adapters
222,000	9	1
129,000	11	2
91,000	13	3
70,000	15	4

If none of the simplex channels are in operation, an adapter with an average byte rate of 143,000 and a maximum service time of 22 usec can be operated. The capacity of the fast simplex channels will be specified in a later edition of the manual.

The capacities shown on the previous pages are based on the assumption that chaining is not specified (chaining is described in section 7.4.4). If chaining is specified, the average byte rate will be reduced.

If the average byte rate of an adapter exceeds the capacity of the channel to which it is connected or if the maximum service time of the adapter is less than the service time of the channel, an "overrun" occurs and information is lost. The adapter detects this condition and gives an Unusual End signal at the end of the operation which causes the channel to alert the program.

PROGRAMMING NOTE

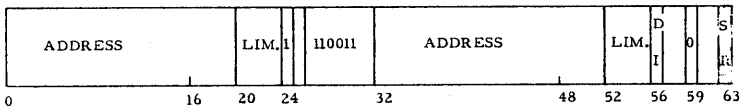
It is possible to operate four IBM 729 IV tape units simultaneously, if byte conversion (eight-bit mode) is specified for all of the operations. Without byte conversion (six-bit mode), two 729 IV units can be operated simultaneously.

7.3 INSTRUCTIONS

An input-output instruction specifies the channel to be used and the location of the chain of control words which specify the operations to be executed.

7.3.1 Instruction Format

Input-output instructions have the format shown below. Bits 17-23, 25, 49-55, 57, 58, 60-62 are ignored.



The first address gives the location of the first control word in the chain of control words which specify the sequence of input-output operations to be executed. This address must be indirect, and may be followed by other indirect addresses before the first control word is reached. The control word is always obtained from the full-word location specified by bits 0-15 of the last indirect address. Bit 16 of the last indirect address normally is zero. If bit 16 is one, the left and right halves of the control word are interchanged as obtained from storage. If the first address is direct (i.e., bit 24 is zero), an Operation Code Invalid alert will be given and the instruction will be suppressed.

The last instruction address may be either direct or indirect as specified by bit 56. Bits 40-47 of the effective address give the address of the channel to be used. The simplex channels are assigned consecutive addresses beginning with address 8. If a single adapter is connected to the multiplex channel, the channel is assigned address 16. If a channel multiplexer is attached, the channels it provides may be assigned any of the addresses 16-127. If the specified channel address is 0-7 or 128-255, or any unassigned address from 8-127, a Channel Not Operational alert will be given, and the instruction will be suppressed.

7.3.2 Instructions

If bit 63 in the instruction is off, the START CHANNEL instruction is specified; RELEASE CHANNEL is specified if the bit is on.

START CHANNEL (SRT)

This instruction initiates the sequence of input-output operations specified by the chain of control words which it addresses. If the addressed channel is busy or not operational, the appropriate interruption code is stored and no operation is initiated.

RELEASE CHANNEL (RLS)

If the addressed channel is busy, this instruction causes information transfer to be terminated immediately. Any alert which results from an input-output operation terminated by this instruction is not suppressed. The first address given in this instruction is not used in the operation, but may cause an Empty Address alert if the instruction is given in the relocation mode.

7.3.3 Instruction Alerts

One of the following alerts is given when a specified sequence of input-output operations cannot be initiated. Other alerts, described in section 7.5, are given when an operation is terminated or when an Attention signal is received from an input-output unit.

If conditions exist that could cause more than one of these alerts, only the alert appearing first on the list below is given.

The first two alerts in this list can be caused by any instruction and are fully described in Chapter 3. They are described here as they apply to input-output instructions. All other alerts listed here can be caused only by input-output instructions.

When any of these alerts occurs, the instruction causing it is suppressed.

Address Invalid (AD)

This alert is given if the processor detects an invalid, indirect address while obtaining the control word or channel address specified by an input-output instruction.

Unended Sequence (US)

This alert results if the program is not resumed within one second after an input-output instruction is given.

Input-Output Alarm (IOA)

If the input-output lock bit of the current program control word is on, an input-output instruction will cause this code to be stored in the interruption stream. This alert permits a supervisory program to monitor all input-output instructions given by a problem program before they are executed. Thus, a problem program can be prevented from using an input-output unit or an area of storage assigned to another program.

Channel Not Operational (CNO)

This interruption code is stored if the adapter connected to the addressed channel is not operational. It is also stored if no adapter is connected to the addressed channel, or if the channel is not provided in the system.

Channel Busy (CB)

This interruption code is stored when an instruction addresses a channel which is executing a sequence of operations initiated by a previous instruction.

7.4 CONTROL WORDS

A channel is capable of executing a series of operations, such as read or write, specified by a chain of control words and initiated by a START CHANNEL instruction.

One or more consecutive control words are used to specify each input-output operation. Each control word defines an area of storage to be used. The first control word associated with an operation specifies the operation, and the last control word indicates whether another operation follows. Operations are executed in the order specified in the chain of control words.

7.4.1 Operations

Each channel is capable of executing four operations: read, sense, write, and control. During read and sense operations (input operations), information flows from the adapter to storage; during write and control operations (output operations), information flows in the opposite direction. Data are transferred by a read or write operation, and sense and control operations are used to transfer control information.

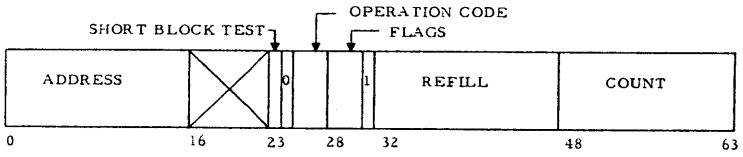
A control operation is used to change the mode of operation of an adapter, to select a unit connected by the adapter or to initiate any operation other than reading or writing, such as rewind of a tape, seek of a sector in a disk file, and stacker selection of a card device. A control operation is terminated as soon as the controlling information is sent to the adapter, and the channel and adapter are available for further use. The unit however may require more time to complete the control operation thus initiated, and in that case is not ready for further operation.

A sense operation is used to transfer control information from the adapter to storage. This information can be used to determine if certain conditions occurred during the preceding operation, such as out of material, data check, or unit not ready.

The amount of information transferred during a sense or control operation is limited by the design of the adapter. These operations are also terminated when the specified storage area is exhausted if this occurs first.

7.4.2 Control-Word Format

The control words which specify an input-output operation have the format shown below. The contents of bits 16 - 22 are ignored.



7.4.2.1 Address

Bits 0 - 15 specify the first location in the storage area defined by the control word.

7.4.2.2 Short-Block Test

If bit 23 is on in the first control word which specifies an input operation, a test is made to determine if the operation is terminated before all of the storage areas defined for the operation are exhausted. If a short block is detected a special condition alert is given at the end of the operation.

If bit 23 is off the test is not made. This bit is ignored in all other control words.

7.4.2.3 Operation Code

Bits 25 - 27 specify the operation to be executed and whether or not an alert should be given at the end of an operation during which no special conditions occur. In all control words used to specify an operation, the state of all bit positions 26 must be the same; i.e., all ones or all zeros. The same rule applies to bit 27. If they are different, the operation will be terminated when the area defined by the previous control word is exhausted; a special condition alert will be given, and the sequence of operations will not be continued.

Suppress normal termination

If bit 25 is on in the last control word used to specify an operation, and if no special conditions occur during the operation, no alert will be given when the operation is terminated. If this bit is off in the last control word, an alert will always be given at the end of the operation.

Data / Control

If bit 26 is off, data are transferred between the adapter and storage. If bit 26 is on, control information is transferred.

Direction

If bit 27 is zero an input operation is specified; a one specifies an output operation.

7.4.2.4 Flags

Bits 28 - 30 are used to specify variations of the basic operations, to indicate the last control word associated with an operation, and to indicate the last control word in a chain.

Skip

Bit 28 is used to suppress the transfer of information to storage during an input operation.

Multiple

In the last control word associated with an operation, bit 29 specifies whether or not another operation follows. In all other control words bit 29 can be used to facilitate the operation of input-output units which are incapable of terminating data transfer immediately when signalled to stop by the computer.

Chain

Bit 30 indicates the last control word associated with an operation.

7.4.2.5 Refill

If either the chain flag or the multiple-operation flag is on, bits 32 - 47 contain the address of the next control word in the chain.

7.4.2.6 Count

Bits 48 - 63 specify the number of 64-bit storage locations in the area defined by the control word.

7.4.3 Definition of Storage Area

The number of locations in the area of storage defined by a control word is specified by the count field. If the count field is zero, the area contains 65,536 (2^{16}) locations. Locations in the area have consecutive addresses beginning with the address specified in the control word. During an input-output operation, the locations in the defined area are used in order of increasing addresses. Location 65,535 is followed by location 0.

Any storage location provided in a system can be used in executing an input-output operation. If a reference is made to a location not provided in the system, however, the operation is terminated immediately and an alert is given.

7.4.4 Chaining

The chain flag, bit 30, in each control word specifies whether or not it is the last control word associated with an operation.

If the chain flag is one, a new control word is obtained from the location specified in the refill field when the defined area is exhausted. The operation is then continued using the storage area defined by the new control word.

If the chain flag in a control word is zero, the operation specified by the control word is terminated when the defined storage area is exhausted, and if the data recorded on the external document is divided into blocks, the document is advanced to the beginning of the next block before another operation is executed using the unit. Therefore, operations using these units must start at the beginning of a block.

The chaining feature permits information to be rearranged as it is transferred between storage and an input-output unit. It also permits a block of information to be transferred to or from noncontiguous areas of storage, and is necessary for the skipping and overrun features.

7.4.5 Multiple Operations

The last control word of an operation is identified by a zero in bit 30, the chain flag. If bit 29, the multiple flag, in this control word is on, another operation follows. This operation is initiated when the previous operation is terminated, unless a special condition is detected at the end of the previous operation. If the multiple flag and chain flag are zero, the end of the chain of control words has been reached.

This feature permits the program to initiate the transmission of multiple blocks of data with a single instruction. It also permits a single instruction to select a particular input-output unit connected to a channel and then to transfer data to or from this unit.

7.4.6 Stopping

During output operations a Stop signal is sent to the adapter when the data area defined by the last control word associated with the operation is exhausted. Following the Stop signal no more information is

transferred to the adapter. During input operations a Stop signal is not sent because the amount of information to be transferred is normally determined by the input unit. If additional data is transmitted by the unit, it is not placed in storage and a long-block condition is indicated at the end of the operation.

For those input units which do not have a natural block length, a Stop signal will be provided when the area defined by a control word with both bits 29 and 30 on is exhausted. Because bit 30, the chain bit, is on another control word is obtained, and any additional data transmitted by the adapter is placed in the storage area defined by this control word This feature also provides a means for suppressing a long-block test for those units which terminate data transfer as soon as a Stop signal is received. In that case the area defined by the additional control word is not used.

7.4.7 Skipping

Bit 28, the skip flag, specifies whether or not information is transferred to storage during an input operation. If the skip flag is zero, a normal input operation is specified.

If the skip flag is one, no information is transferred to storage. The operation of the adapter is continued, however, and the computer counts the bytes transmitted by the adapter. If the chain flag is one, a new control word is obtained when enough bytes have been received to fill the defined storage area. If the skip flag in the new control word is off, subsequent bytes are transferred to storage. The skip flag is ignored during output operations.

The skipping feature, when combined with chaining, permits selected portions of a block of information from an input unit to be placed in storage.

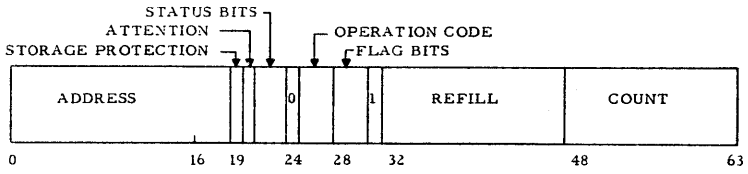
7.5 TERMINATION

The operation of a channel is controlled by the "channel control word" stored in location n + 128, where n is the channel address. When an operation is initiated, the first control word specifying the operation is stored in the channel control-word location. As the operation progresses, the channel control word is modified repeatedly. When the storage area specified by the control word is exhausted, either the next control word in the chain is stored in the channel control-word location and the operation proceeds or the operation is terminated. When the operation is terminated the channel control word indicates any special conditions which occurred during the operation and the number of word and/or bytes in the defined storage areas which were not used. The channel control word can be ex-examined by the program at any time. The following section describes its status at the end of an operation. The way in which the channel control word is modified during an operation is described in section 7.6.

When an input-output operation is terminated, the program may be alerted by means of the interruption system. The code stored in the interruption stream identifies the channel, and distinguishes between a normal and special termination.

7.5.1 Channel Control Words

The format of a channel control word at the end of an operation is shown below.



7.5.1.1 Address

If no special condition occurs during an operation in which all of the defined areas are used, bits 0 - 18 address the first byte following the last area defined. If an operation is terminated by the adapter before the defined areas are exhausted, this field addresses the first byte in the defined areas which was not used.

7.5.1.2 Storage Protection

If bit 19 is zero during an input operation, bytes will not be placed in storage. The storage protection bit is always zero at the end of an operation.

7.5.1.3 Attention

Bit 20 is turned on when an Attention signal is received from the input-output unit. It is not altered by the initiation of an operation or by chaining. It is never turned off by the computer, and therefore, must be turned off by the program when the Attention signal is serviced.

7.5.1.4 Status Bits

Bits 21 - 23, are turned on during an operation if certain special conditions occur. They are reset when the next operation is initiated.

Unusual end

Bit 21 is turned on at the end of an operation if the adapter indicates that it detected a special condition during the operation. Some of these special conditions are--

a. An operation is specified which the addressed unit is not designed to execute (e.g., a read operation for a printer).

b. An operation is specified which the addressed unit is not capable of executing because of its present status (e.g., a read operation for a tape unit which is rewinding).

c. An undefined control code is sent to the adapter during the execution of a control operation.

d. A data error is detected by the adapter. The adapter makes this check even though the channel is skipping or the defined storage areas have been exhausted.

e. The adapter detects a malfunction of one of its components.

f. The unit reaches an out-of-material condition (e.g., an empty card hopper, a full card stacker, the end of a tape).

g. A tape mark sensed.

h. A byte of information is lost during the operation because of timing restrictions.

Program check

Bit 23 is turned on if the computer detects any of the following programming errors during an operation.

a. Short block. The number of bytes transmitted during an input operation is insufficient to fill the storage area or areas specified. This test is made only if bit 23 is on in the first control word associated with the operation. When a short block is detected, the address field gives the location of the first byte in the defined areas not used.

b. Long block. During an input operation the adapter tries to transmit one or more bytes to storage after the defined storage area or areas are filled. The extra bytes are not placed in storage. The count will always be zero and the chain flag will be off in the channel control word. The long-block test can be suppressed by a procedure described in section 7.4.6.

c. Invalid data address. The computer attempts to transmit information to or from a storage location not provided. Operation is terminated immediately, and the invalid data address appears in the channel control word. This test is suppressed when skipping during an input operation.

d. Invalid refill address. During an input-output operation the computer attempts to obtain a control word from a storage location not provided. The operation is terminated immediately, and the invalid refill address appears in the channel control word. If the address specified for the first control word associated with an operation is invalid, the operation is not initiated and an Address Invalid (AD) alert is given.

e. Incorrect operation code. The operation code specified in a control word, other than the first control word associated with an operation, differs from the corresponding bits in the first control word. The operation is terminated immediately. Although the control word containing the incorrect operation code is inspected, it is not placed in the channel control-word location.

The preceding control word, as modified by the operation will be found in the channel control-word location. The count will be zero, and the chain flag will be on.

f. Unending sequence of control words. If an input-output operation is terminated by the adapter before all the control words associated with the operation are used and if no special conditions have been detected, the computer obtains the remaining control words one by one until a control word is found in which the chain flag is off. If this process requires more than one second, a programming error is indicated.

7.5.1.5 Operation Code

Bits 25-27 identify the last operation executed.

7.5.1.6 Flag Bits

Bits 28-30 contain the flags specified in the last control word used.

7.5.1.7 Refill

Bits 32-47 contain the refill address specified in the last control word used.

7.5.1.8 Count

Bits 48-63 are normally zero at the end of an operation. If an operation is terminated by the adapter or by an invalid data address, the count indicates the number of fully or partially unused storage locations in the area defined by the last used control word.

7.5.2 Terminating Alerts

The code stored in the interruption stream by either of the two alerts described below identifies the channel with which the alert is associated.

Channel n Normal End (CnNE)

This alert is given when an input-output operation using channel n is terminated normally and the suppress normal termination bit, bit 25, is off in the last control word used to specify the operation. If a subsequent operation is specified by the chain of control words, it is initiated immediately. If the operation is terminated normally and the suppress normal termination bit is on, no alert is given.

Channel n Special Condition (CnSC)

This alert is given when an input-output operation using channel n is terminated, if either the attention bit or any of the status bits are on in the channel control word at the end of the operation. If a subsequent operation is specified by the chain of control words, it is not executed. This alert is also given if an Attention signal is received from channel n when the channel is not busy.

7.6 CHANNEL OPERATION

Input operations using the simplex channels and all operations using the multiplex channel are controlled by the channel control word associated with the channel used for the operation. An "auxiliary control word" associated with each simplex channel controls output operations using the channel. The channel control word associated with the channel is stored in location n + 128 where n is the channel address, and the auxiliary control word associated with a simplex channel is stored in location n + 124. Two other storage locations, n + 116 and n + 120, associated with each simplex channel, are used as buffers.

Three types of sequences--setup, information transfer, and ending--are executed by a channel during an input-output operation. A setup sequence is initiated by a START CHANNEL instruction or may follow an ending sequence. During this sequence the first control word used to specify an operation is fetched from storage, modified, and stored in the channel control-word location or the auxiliary control-word location. Then the appropriate command is sent to the adapter.

An information-transfer sequence is initiated by a service request from the adapter. During this sequence a byte is transmitted to or from the adapter, and the channel control word may be modified. For a simplex channel, information may be transferred between the defined storage area and a buffer location, and if an output operation is in progress, the auxiliary control word may be modified.

An End or Unusual End signal from the adapter initiates an ending sequence. During this sequence the adapter is reset in preparation for another operation. If the conditions for an alert exist, a byte is stored in the interruption stream. If another operation is specified by the chain of control words and no special conditions occurred during the current operation, a setup sequence follows this sequence.

The contents of the fixed locations associated with a channel can be inspected by the program at any time except when one of the above sequences is being executed. This is useful for determining the progress of a current operation. Furthermore, if the prefix address is zero, the program can alter the contents of the channel and auxiliary control words, and thus, change the course of an operation. Only data-transmission instructions should be used for this purpose.

7.6.1 Simplex-Channel Sequences

During the setup sequence for an input operation using a simplex channel, bits 16-18 of the channel control word are set to zero. The

Attention bit in the channel control word is not altered by this sequence or any of the other sequences described in this section. A Read or Sense command is sent to the adapter.

During the setup sequences for an output operation using a simplex channel, the first control word specifying the operation is modified, as described above for an input operation, and stored in the channel control-word location. The control word is then further modified by incrementing the word address (bits 0 - 15) by one and decrementing the count by one, and the resulting control word is stored in the auxiliary control-word location. The first two words in the defined area are transferred to the buffer locations associated with the channel, and a Write or Control command is sent to the adapter.

An information-transfer sequence for an input operation transmits a byte from the adapter to one of the buffer locations associated with the channel. When the buffer is filled, its contents are transmitted to the storage location addressed by the channel control word, provided the storage-protection bit is on and the skip bit is off. Then the address in the channel control word is incremented and the count is decremented. Thus, the control word always addresses the location into which the word being assembled is to be placed.

An information-transfer sequence for an output operation transmits a byte from one of the buffers associated with the channel to the adapter. When a full word has been sent to the adapter, the auxiliary control word is transmitted to the channel control-word location. Then, if the storage-protection bit is on, the address in the auxiliary control word is incremented, the count decremented, and the buffer refilled from the storage location designated by the new address. Thus, during an output operation, the channel control word always addresses the location containing the word being disassembled, and the auxiliary control word addresses the next location in the defined storage area.

During an ending sequence, bits 16 - 18 in the channel control word are set to address the byte following the last byte used.

If a buffer is partly filled during an input operation, its entire contents are transferred to the storage location specified by the address in the channel control word. The contents of the unfilled byte positions in this buffer are not predictable.

7.6.2 Multiplex-Channel Sequences

The setup sequence for all operations using the multiplex channel is identical to the setup of an input operation for a simplex channel.

During an information-transfer sequence, a byte is transmitted from the byte position in storage specified by the byte address in the control word (bits 0 - 18) to the adapter if the direction bit (bit 27) is on. Transmission in the opposite direction takes place if the direction and skip bits are off. Next, the byte address is incremented by one byte. If a carry results from bit 16 to bit 15 (i.e., a word boundary is crossed), the count is decremented by one.

During an ending sequence, bits 16 - 18, which specify the next byte to be transferred, remain unchanged. If a word is partially filled during an input operation the unused bytes are not changed.

Chapter 8

MANUAL CONTROL AND SYSTEM MAINTENANCE

Contents

Contents Chapter 8 (Continued)

Chapter 8

MANUAL CONTROL AND SYSTEM MAINTENANCE

8.1 GENERAL DESCRIPTION

Initial program loading, recovery from processor malfunctions, and system maintenance are accomplished by means of a "scanner." The controls of the scanner are independent of the processing controls, but the scanner uses the main data paths of the processor. The operations executed by the scanner are called "system operations." These are executed in a sequence determined by manual controls, rather than by a stored program. The scanner can load a program into storage from an input unit and initiate its execution. When a malfunction is detected by a checking circuit in the processor, the scanner records the state of most of the processor triggers in storage or on an external document and then restarts the program at a predetermined place. The scanner can detect and locate faults in the data paths, or the processing or input-output controls by executing a diagnostic procedure specified on external documents. The results of this procedure are recorded on external documents.

A system panel is provided for manual control of the system by an operator or a customer engineer. The panel can be located on the right-hand side of a universal console, or it can be an independent unit. It contains the keys, switches, and lights necessary to control the processor power supplies and the scanner. A set of lights indicates the state of the system and signals the operator if a malfunction occurs from which the processor cannot recover automatically. A complete set of controls is provided for the manual location of faults in the scanner controls by means of a single-cycling technique, and similar but rudimentary controls are provided for the processing and input-output controls. Facilities have also been provided for executing any operation or part of an operation repeatedly so that an oscilloscope can be used to locate faults.

8.2 SYSTEM PANEL

The lights, keys, and switches described in this section are provided for manual control of the central-processing unit and for system maintenance. These controls are located on the system panel, illustrated in Figure 8.1, which can be attached to the right-hand side of a universal console or can be an independent unit. The controls are directly connected to the processor.

8.2.1 Power Controls

POWER ON, CPU STORAGE Key and Light

Depressing this key turns on the d-c power supplies for the processor and storage units in the proper sequence. It also closes the line contactors in the input-output adapters. A light mounted inside the key indicates the d-c power is on.

SYSTEM POWER OFF Key

Depressing this key turns off the d-c power supplies for the processor and storage units in the proper sequence, and turns off power to the input-output adapters.

EMERGENCY POWER OFF Pull Switch

Pulling this switch turns off all power immediately beyond the entry terminal in every unit in the system.

CPU FUSE, CPU THERMAL, STORAGE FUSE, and STORAGE THERMAL Lights

These four lights indicate that a thermal-protection switch is open in the processor or a storage unit, or a fuse is blown in one of these units.

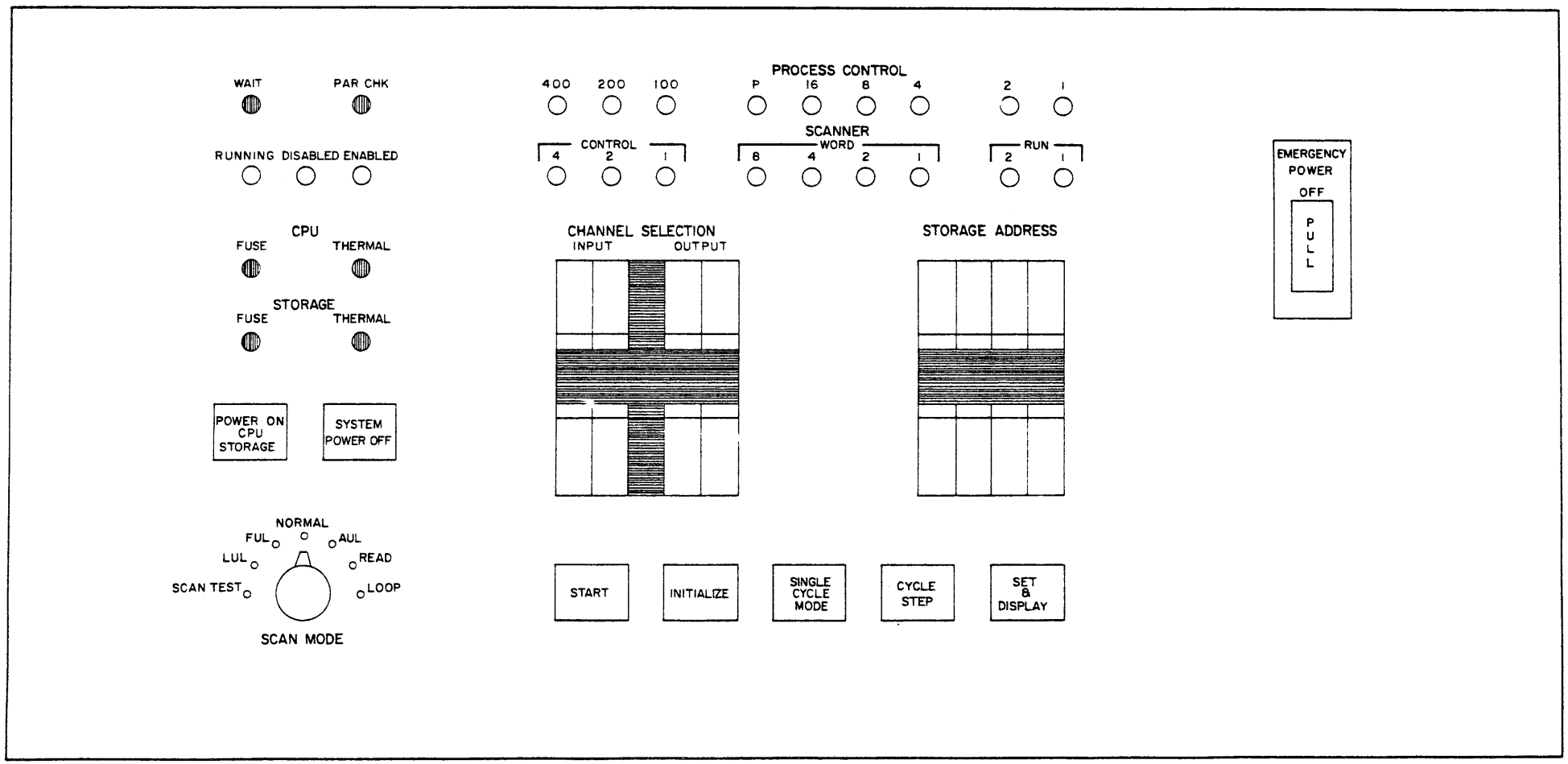


Figure 8.1 - System panel

8. 2. 2 System Mode Switch

This is a seven-position rotary switch which is used to select the system mode. The modes are listed below along with their uses.

NORMAL and AUL (Automatic Unload)

These modes are used for normal productive operation. They are discussed in sections 8. 4 and 8. 5.

FUL Mode

This mode can be used by the operator to record the state of the triggers in the processor. It is discussed in section 8. 5. 2.

SCAN TEST and LUL (Load and Unload)

These modes are used in the mechanized maintenance procedures for the system and are discussed in section 8. 6.

READ and LOOP

These modes are used in the manual maintenance procedures for the system and are discussed in section 8. 7.

8. 2. 3 Operator Controls

These lights, keys, and switches are used by the operator to control the system during normal productive operation.

RUNNING Light

This light is turned on for 1/10th second every time an instruction is obtained from storage, except an instruction following a successful branch instruction.

WAIT Light

This light is on when the processor is in the initial or waiting state.

ENABLED Light

This light is on if the processor is enabled.

DISABLED Light

This light is turned on every time the processor enters the disabled mode. It remains on as long as the processor is disabled, but never less than 1/10th second.

PARITY CHECK Light

This light is turned on for 1/10th second every time the processor or a storage unit detects a parity error.

INITIALIZE Key

Depressing this key places the system in the initial state, which is described in section 8.3.

START Key

When the system is in the initial state, depressing this key causes the processor to execute the sequence of system operations specified by the system-mode switch. This key is inoperative when the system is in the running or waiting state.

INPUT CHANNEL SELECTION Switches

These two 16-position switches specify the channel to be used for input data during the initial program-loading procedure, or a mechanized or manual maintenance procedure. The positions of these switches are designated 0, 1,.....,9, U, V,.., Z.

OUTPUT CHANNEL SELECTION Switches

These two 16-position switches specify the channel to be used for output data when recording the state of the processor triggers in the AUL or FUL mode, or during a mechanized maintenance procedure. The sixteen positions of these switches are designated 0, 1,....., 9, U, V,.., Z.

8.2.4 Customer Engineering Controls

SINGLE CYCLE Key and Light

This key is used to place the processor in either the automatic or single-cycle mode. These two modes combine with the seven modes which can be selected by means of the SYSTEM MODE switch to give a total of fourteen systems modes. If the system is in the automatic mode,

depressing this key places the system in the single-cycle mode. Depressing the key a second time returns the system to the automatic mode. A light mounted inside the key indicates that the system is in the single-cycle mode.

STORAGE ADDRESS Switches

These four 16-position switches specify a storage location. They are used in conjunction with the SET AND DISPLAY key for displaying the contents of the specified location. The STORAGE ADDRESS switches are also used in the read mode to specify the storage location to be loaded.

SET AND DISPLAY Key

Depressing this key causes the contents of the storage location specified by the STORAGE ADDRESS switches to be displayed in the DATA REGISTER lights of the storage unit. This key is operational only when the system is in the single-cycle mode.

CYCLE STEP Key

When the processor is placed in the single-cycle mode it stops. Each time this key is depressed, the processor is advanced one cycle. This key is not operational when the processor is in the automatic mode.

PROCESS CONTROL Lights

These lights partially indicate the state of the processing controls. They are used for manual maintenance of the system.

SCANNER Lights

These lights indicate the state of the scanner controls and are used for manual maintenance of the scanner.

8.3 SYSTEM STATES

There are three system states: initial, running, and waiting. These states and the transitions between them are shown in Figure 8.2.

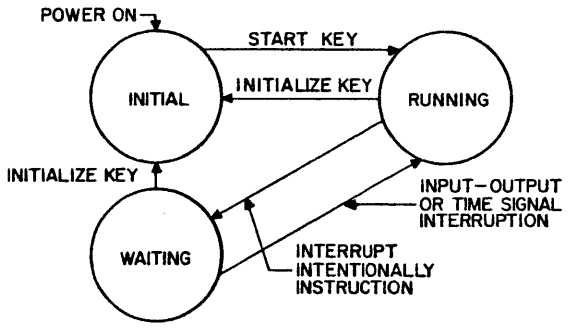


Figure 8.2 - System states

8.3.1 Initial State

The system can be placed in the initial state at any time by depressing the INITIALIZE key. This terminates all input-output operations in progress and resets the adapters to a predetermined state. If a program is in progress, its execution is terminated without completing the current operation. The contents of the triggers in the processor are then placed in storage locations 17-31 in the manner illustrated in Figure 8.3. If a time clock and interval timer are provided in the processor, they are stopped.

When the system is in the initial state all parts of the system are inactive. When power is restored to the system after being turned off, the system will be in the initial state. Depressing the START key causes the system to leave the initial state and enter the running state.

PROGRAMMING NOTE

When the system is placed in the initial state the program is stopped before completion of the current instruction. Therefore, the program normally cannot be resumed.

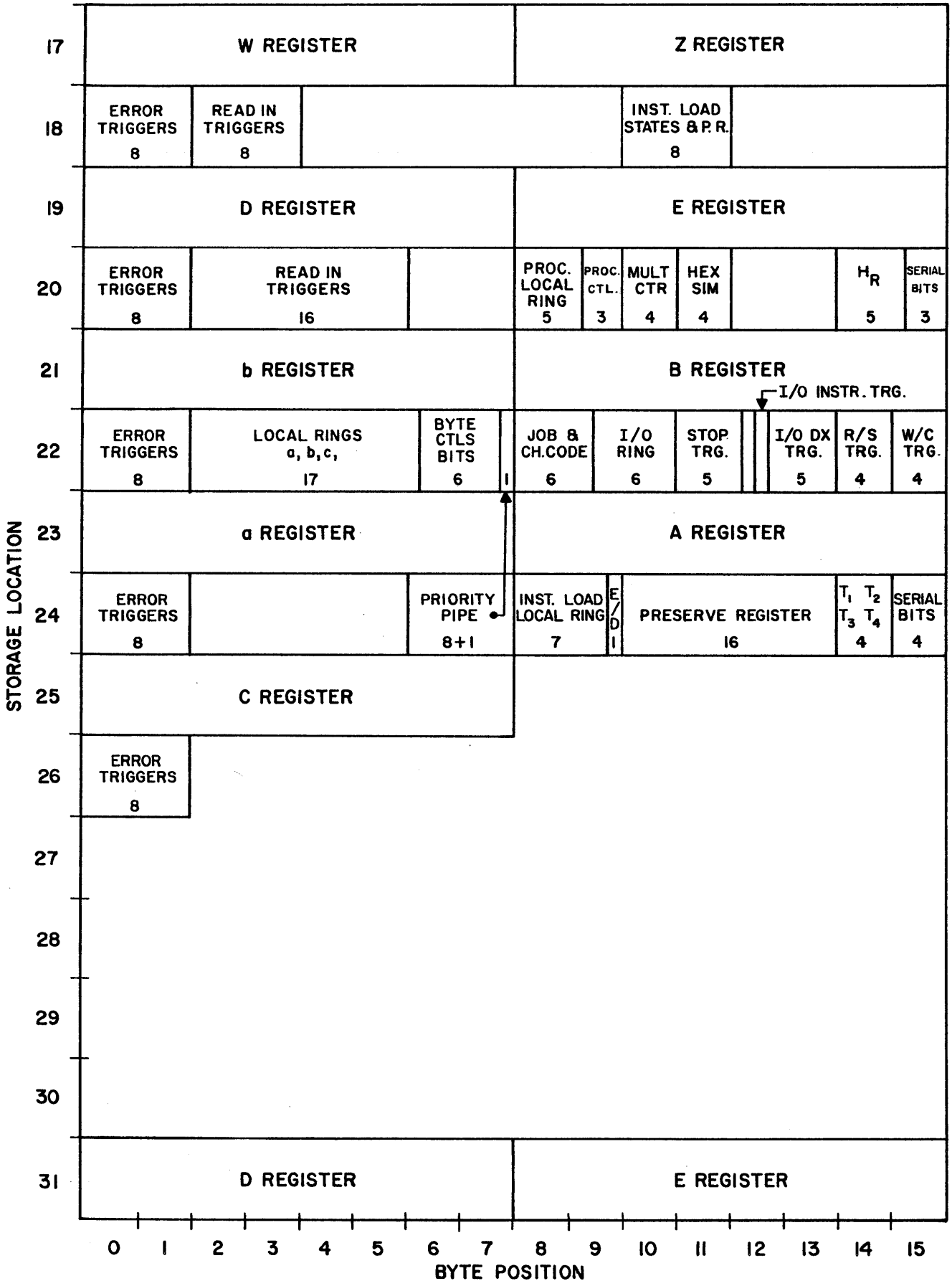


Figure 8.3 - Scan-out pattern

8.3.2 Running State

When the system is in the running state, the processor is always busy executing an instruction or a system operation. The input-output channels may or may not be in operation. Instructions and system operations are never executed in the initial or waiting states.

8.3.3 Waiting State

If an INTERRUPT INTENTIONALLY instruction which addresses a byte containing the code Programmed Wait (PW) is executed, the system is placed in the waiting state. In this state, no instructions are executed, but input-output operations may continue, and the time clock and interval timer will continue to run. The system is returned to the running state when an Input-Output Alarm or Time Signal alert occurs.

8.3.4 System Status Lights

The RUNNING and WAIT lights indicate the state of the system as follows:

Running	Wait	State
On	Off	Running normally
Off	On	Programmed waiting or initial state
Off	Off	Trouble--the processor is hung up because of a malfunction of the controls or it is executing only successful branch instructions.

8.4 INITIAL PROGRAM LOADING

Provision has been made for loading a program into storage and initiating its execution under external control. This procedure is necessary when the power is turned on, and to recover from some programming mistakes and system malfunctions. The program can be loaded from any input unit connected to the machine which is capable of transmitting all possible eight-bit bytes to the processor (e.g., a card-reader or tape unit). Loading a program from a unit which is capable of transmitting only a restricted set of bytes (e.g., a console typewriter) is not possible.

Program loading is initiated by placing the system in the initial state by depressing the INITIALIZE key or by turning the power on. The input unit from which the program is to be loaded is selected by means of the INPUT CHANNEL SELECTION switches located on the system panel. These switches may be set either before or after the INITIALIZE key is depressed. The system must be placed in either the normal or AUL mode.

Once the processor is placed in the initial state, it is inactive until the START key is depressed. This permits the operator to load the program deck or tape on the selected input unit. When the START key is depressed, the following sequence of system operations is executed. This sequence and all other sequences of system operations that can be executed in the normal or AUL mode are illustrated in Figure 8.4 .

1. One block of data is read from the selected unit, and the first fifteen words are placed in storage locations 16-30. The first word of this block gives the contents of the program control word at the beginning of the program being loaded. The remaining fourteen words read from the specified unit contain the program to be initiated. If the block contains more than fifteen words, the additional words are not placed in storage. No termination alert is given by the channel.

2. The word in location 16 is transmitted to the D and E registers, and the processor controls are set to a predetermined state.

3. The system enters the running state, and the program is initiated with the execution of the instruction specified by the program control word.

PROGRAMMING NOTE

The initial loading procedure does not alter the arithmetic tables. If these are incorrect, they must be corrected by the initial program before a decimal multiplication, a conversion or a division operation can be executed. The initial loading procedure also does not alter the interruption control words. These words should be loaded to specify where the interruption codes are to be stored.

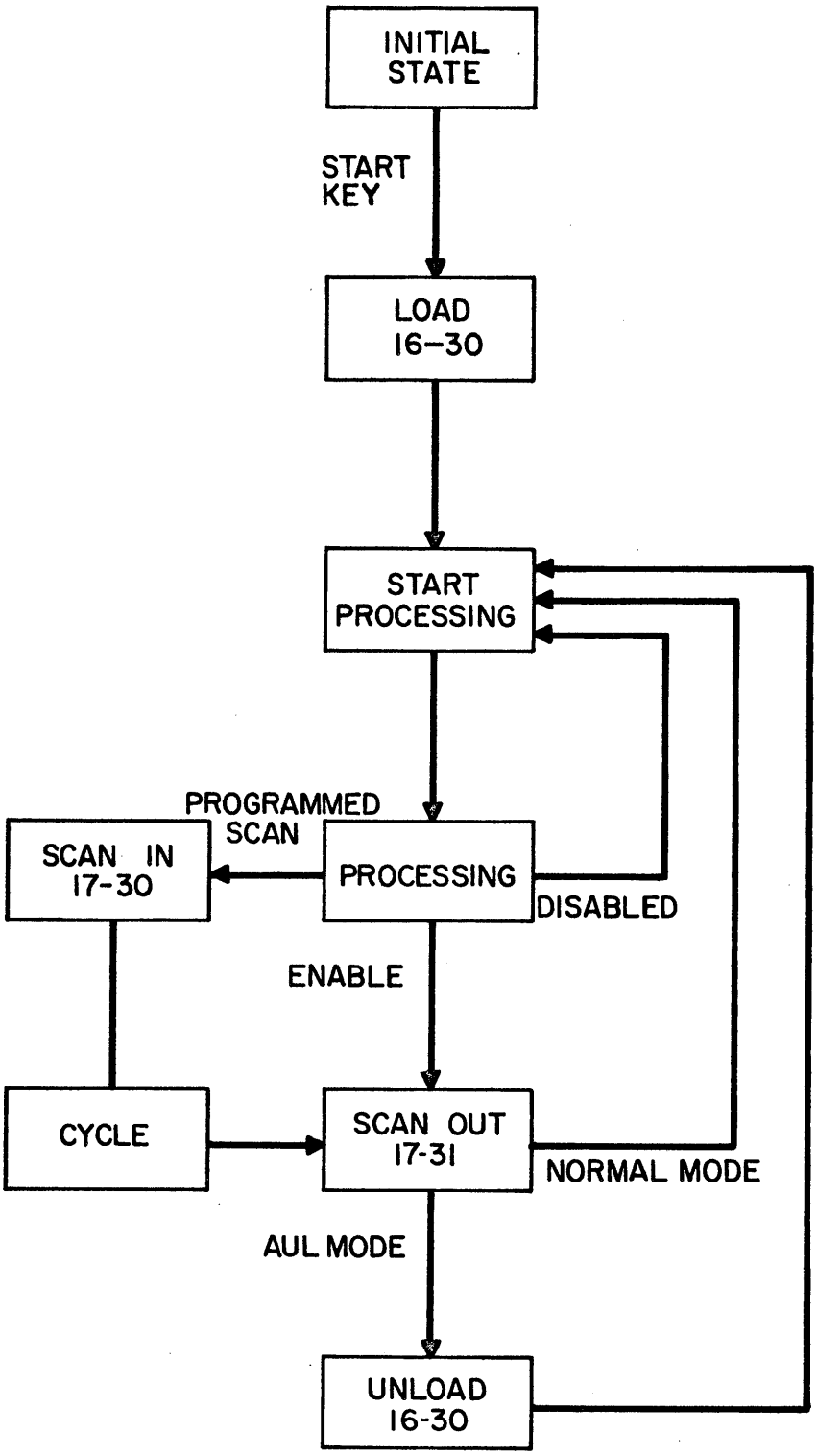


Figure 8.4 - Normal or AUL mode

8.5 CHECKING

Equipment is provided for automatic detection of system malfunctions and for recording the information needed for fault location. When a malfunction is detected the contents of processor triggers are placed in storage and, if desired, this information is transmitted to an output unit. Then processing is restarted with a program control word obtained from a fixed location.

8.5.1 Extent of Checking

Data is represented in the processor by eight-bit bytes, to which a ninth odd parity bit is added. Every byte used in an arithmetic or a logical operation and every byte placed in storage is checked for correct parity. In order to provide continuity of data checking the parity is preserved in storage and in the processor registers, and whenever a byte is changed a new parity bit is generated independently of the new byte. Logical circuits are designed such that single-component faults will always produce incorrect parity. The processor contains eight check circuits so that eight bytes can be checked simultaneously as they are placed in storage. These circuits, which are shown on the data-flow diagram given in appendix II, are also used to check serial arithmetic and processing operations. The control section of the processor has been designed so that malfunctions in a large part of controls will result in parity errors in the data being processed. Therefore the data-check circuits also provide a large degree of control checking.

Errors introduced with input data will normally be detected by the input units or their adapters. Data-transmission errors detected during input-output operations do not initiate the recording of the processor state or the program restart procedure.

8.5.2 Machine State Recording

If a malfunction is detected while the system is enabled, processing is stopped during the cycle in which the malfunction is detected, and the contents of processor registers and triggers remain unchanged while they are recorded in storage locations 17-31. The register contents are placed in storage with correct parity. Any necessary parity corrections are indicated by data bits in the following storage location. These bits and any other bits which have no parity associated with them are grouped in eight-bit bytes and have a correct parity bit affixed to them during the recording process. Figure 8.3 shows the contents of locations 17-31 after the recording operation has taken place.

During the recording process no input-output sequences are executed. As a result an overrun condition may occur in an unbuffered adapter and lead to a subsequent Unusual End from the adapter.

If the processor is disabled when a malfunction is detected, the state of its registers and triggers is not recorded in storage. Since the processor is always disabled before it is restarted following the detection of a malfunction, a recording of the processor state is protected from replacement by recordings resulting from later malfunctions.

Malfunctions are detected only when the processor is executing instructions or input-output operations in either the normal mode or the automatic unload (AUL) mode. In the normal mode the program is restarted after the recording operation. In the AUL mode the contents of location 17-31 are transmitted to the unit attached to the channel selected by the OUTPUT CHANNEL SELECTION switches after the recording process and before the program is restarted. The selected output unit should be reserved for this purpose when the processor is in the AUL mode.

The FUL mode permits the operator to obtain the state of the processor at any moment. This is accomplished by depressing the INITIALIZE key which stops the processor and records its state in storage locations 17-31. Then an output channel is selected by means of the OUTPUT CHANNEL SELECTION switches and the system is placed in the FUL mode. When the START key is depressed the contents of locations 16-30 are transmitted to the selected output unit. The sequence of system operations executed in the FUL mode is illustrated in Figure 8.5. The initial program-loading procedure must be used to restart the processor.

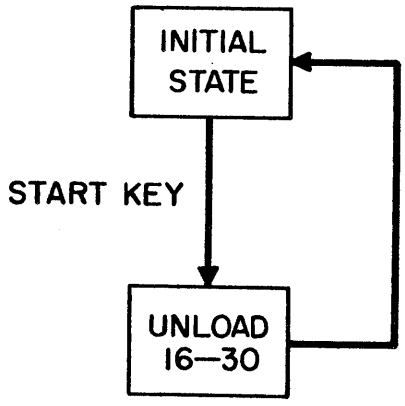


Figure 8.5 - FUL mode

8.5.3 Program Restart

After a malfunction is detected and the required scan-out and unload procedures are executed, the processor is disabled and a new program control word is obtained from location 16. Then normal processing is restarted. Processing starts with the instruction specified by the new program control word. All input-output operations in process when the malfunction was detected are resumed.

8.5.4 Programmed Fault Location

An instruction, SCAN, is provided to facilitate the location of processor faults by a program. If this instruction is given when the system is disabled, the system operations shown in Figure 8.4 are executed. Normal processing is stopped and the contents of storage location 17-30 are placed in the processor registers and triggers in the manner illustrated in Figure 8.6. The processor is then advanced the number of cycles specified in bits 58-63 of location 29, and the resulting state is recorded in locations 17-31. If the system is in the normal mode, processing is then restarted with the instruction specified by the program control word in location 16. If the system is in the AUL mode, the contents of locations 16-30 are transmitted to the unit selected by the OUTPUT CHANNEL SELECTION switches before processing is restarted.

All input-output operations in progress when this instruction is given are suspended while the system operations described above are executed. When processing is restarted the input-output operations which were in progress are resumed. If an unbuffered adapter is in operation an overrun condition may occur. If a SCAN instruction is given when the system is enabled, none of the system operations described above are executed, and processing is continued.

The scanner also provides a means for placing data with incorrect parity in storage. This data can then be used to test the operation of the checking circuits. By means of a scan-in operation, the processor controls are placed in a state that cannot occur in normal operation. When the processor is advanced, the contents of registers a-A and b-B are or-ed and placed in the storage location specified in register c. In this way any eight-bit byte, except all zeros, can be stored with even parity.

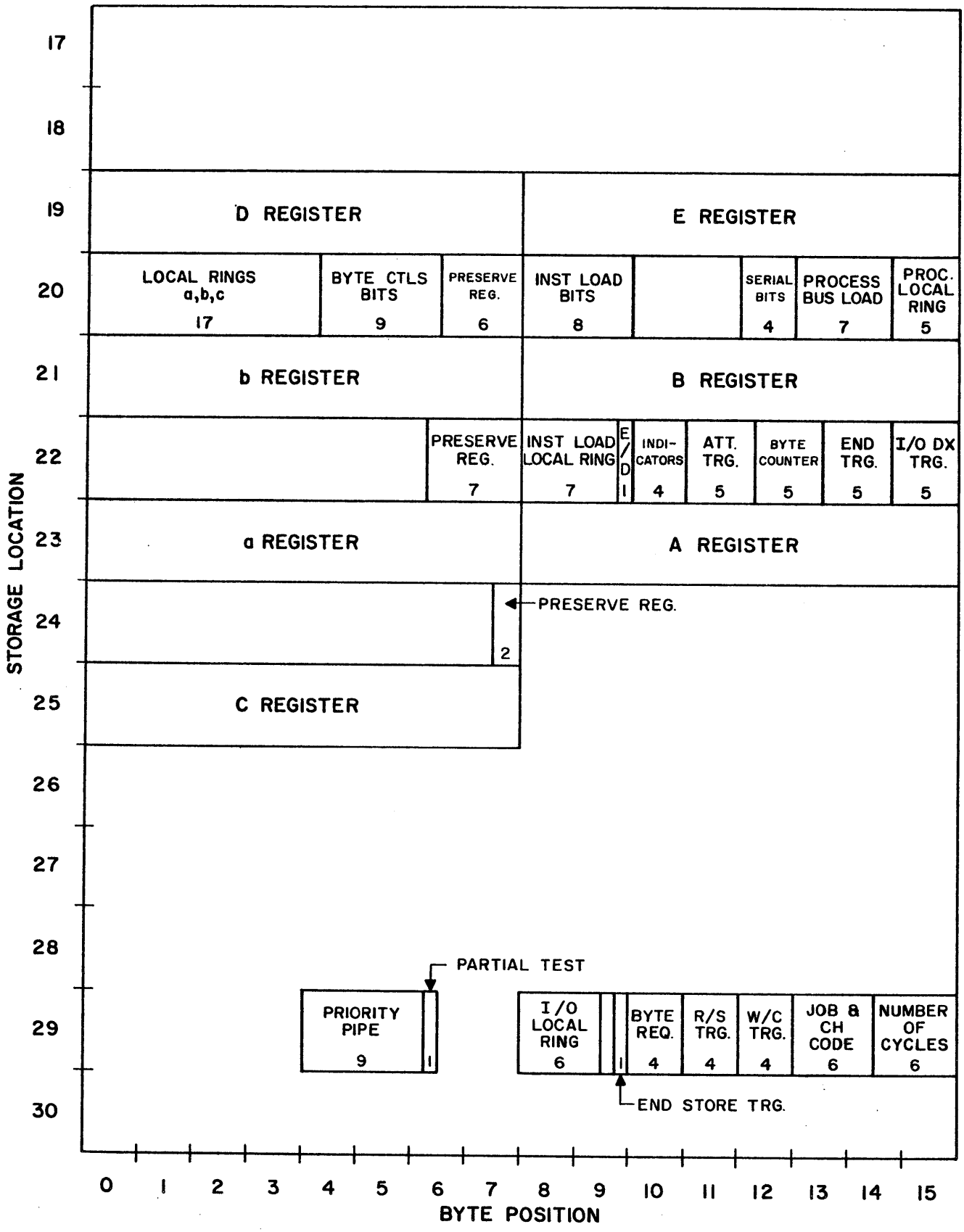


Figure 8.6 - Scan-in pattern

8.6 MECHANIZED FAULT LOCATION

The LUL and scan-test modes facilitate the location of processor faults by mechanized procedures. The sequence of system operations which can be executed in these modes is illustrated in Figure 8.7.

8.6.1 Scan Test

The scan-test mode is used to locate faults in the processor data paths, and in the processing and input-output controls. The fault location procedure is initiated by placing the system in the initial state and selecting the scan-test mode. An input and an output channel are then selected, and a deck or a tape which specifies the fault-location procedure is loaded on the input unit. When the START key is depressed one block of data is read by the input unit and the first fifteen words are placed in storage locations 16-30. If the block contains more than fifteen words the additional words are ignored. The contents of locations 17-30 are then placed in the registers and triggers of the processor using the pattern shown in Figure 8.6. If the full test is specified, i.e., bit 25 in storage location 29 is off, the processor is advanced by the number of cycles specified in bits 58-63 of location 29. The contents of the processor triggers are then recorded in locations 17-31, and a second block is read by the input unit and compared with locations 16-30. If any differences are detected, the contents of storage locations 16-30 are transmitted to the selected output unit. The procedure is repeated until one of the input-output units signals an Unusual End. If the contents of the processor triggers are identical with the second block, no information is transmitted to the output unit and the process is repeated immediately. If a partial test is specified, i.e., bit 25 in location 29 is on, the processor is not advanced and its state is not recorded. The second block read from the input unit is compared with the information obtained from the first block.

8.6.2 Load and Unload

The LUL mode is used to locate faults in the scanner controls, the storage unit, and some of the data paths. The procedure is the same as in the scan test except that as soon as a block of data has been read from the input unit and placed in location 16-30, it is immediately transmitted to the output unit. The procedure is repeated until one of the input-output units signals an Unusual End.

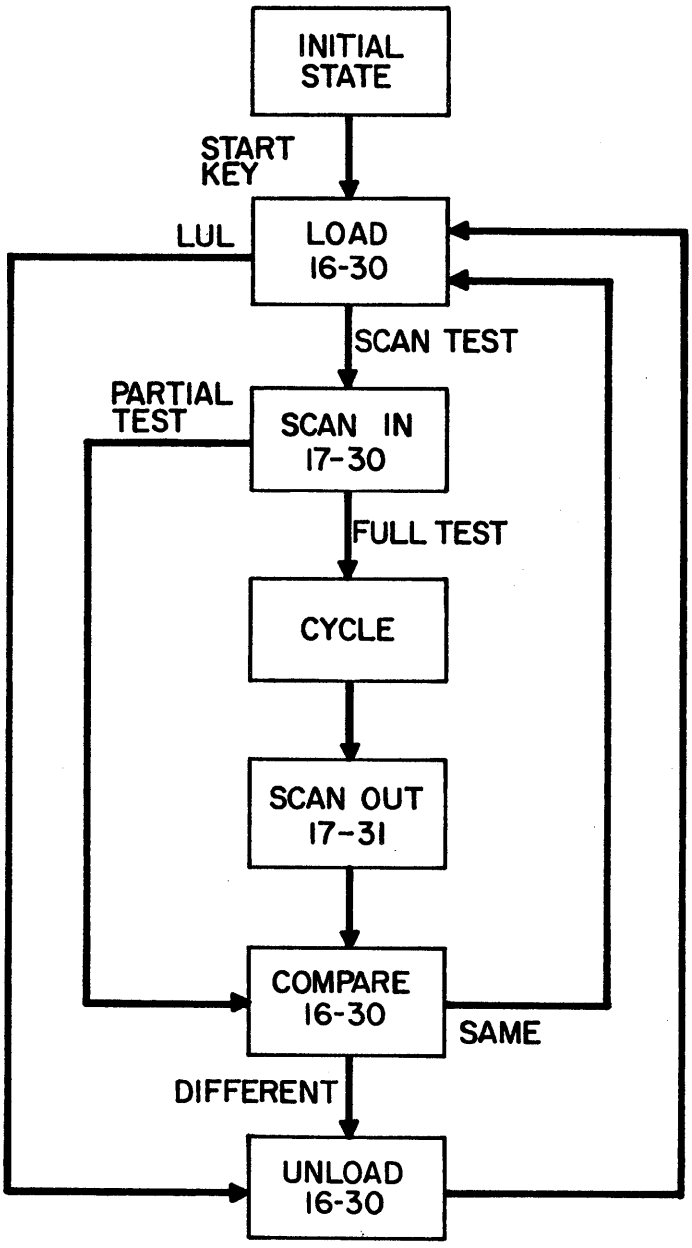


Figure 8.7 - Scan test or LUL mode

8.7 MANUAL FAULT LOCATION

Manual fault location is necessary when the processor is incapable of executing a mechanized fault-location procedure or when the mechanized procedure does not locate a fault with sufficient precision. Two system modes, loop and read, and the customer engineering controls on the system panel are provided to facilitate these procedures.

8.7.1 Single Cycling

Controls are provided for single cycling the processor and for displaying a specified storage location. When the processor is operated in this manner, the states of all the triggers in the scanner controls and the most important triggers in the processing and input-output controls are displayed on the system panel. During manual fault-location procedures it is often desirable to enter the information into storage. This can be accomplished by means of the read mode. The system is placed in the initial state and the read mode selected. Bytes of information can then be entered from the selected input unit and placed in the location specified by the STORAGE ADDRESS switches. The first byte is placed in the left-hand byte position of this location, and the subsequent bytes are placed in successive byte positions to the right.

8.7.2 Scoping

The loop mode is provided to facilitate fault location by means of an oscilloscope. The sequence of system operations executed in this mode is shown in Figure 8.8. The system is placed in the initial state and the loop mode selected. When the START key is depressed a block of data is read by the selected input unit and placed in storage locations 16-30. The contents of locations 17-30 are then transmitted to the processor triggers, and the processor is advanced the number of cycles specified in bits 58-63 of location 29. The scan-in and cycle operations are then repeated continuously, thus providing the repetitive signals required by an oscilloscope. The process can be stopped by depressing the INITIALIZE key.

Chapter 9

TIMING

Contents

Chapter 9

TIMING

9.1 GENERAL DESCRIPTION

Timing in the machine is governed by a clock which provides 1-usec intervals for processing cycles and 2-usec intervals for cycles requiring access to storage. Performance is substantially enhanced by overlapping processing and storage whenever possible. If processing takes place during a storage cycle, the processing cycle is extended to coincide with the storage cycle. The execution of any instruction requires an integral number of processing and storage cycles. However, before storage may be used, it must be established that no activity having a higher priority, such as input-output, is engaging storage. This priority request always requires a processing cycle, which whenever possible is overlapped with a storage cycle from another source.

Even if there is no input-output activity, overlapping of processing and storage cycles is still possible, since there are in the processor several control areas whose activities can overlap. The amount of overlapping depends therefore on the complete state of the machine, and succinct rules cannot always be given for determining it. For this reason the instruction times presented in this chapter are provided in the form m(n), where m is the maximum time required for an instruction, assuming no overlap, and (n) is the time when the overlapping conditions are the most favorable. Where m and (n) are equal, only m is written. All times are in microseconds (usec).

Should an activity of higher priority be using storage or some part of the processor, the instruction currently being executed will wait until the facilities are available. No account of this variable waiting time is taken in assessing the time required to execute an instruction, since this time is included in the instruction time of some other activity.

9.2 TIMING OF INSTRUCTIONS

The complete time required to execute an instruction is influenced by such factors as indirect addressing, relocating, indexing, crossing of word boundaries, and the refilling of control words. Except where indicated, the basic times ignore these factors. In the absence of specific information under each instruction, the following times should be added to any calculation.

Indirect Addressing and Relocation

For the first level of indirect addressing, or for relocated addresses, 2 usec for all but the last address of any instruction, for which the time is 3(2) usec; 3(2) usec for each subsequent level of indirect addressing or for relocated refill addresses.

Indexing

For each indexed address 9(4) usec, except for floating-point instructions for which the last level of indexing takes 7(2) usec.

Interruptions

For all interruptions, 12(9) usec are required to place the appropriate code in the interruption stream, except for I/O interruptions, for which the figure is 8(6) usec; and a further 12(7) usec to fetch the code from the interruption stream and enter the disabled mode.

Multiple Field or Record

Where the multiple-field or record format is used 3(2) usec in addition to the indirect addressing time, are required to place each control word initially in a location relative to the prefix. Furthermore, 6(4) usec must be added each time a control word is fetched, updated, and restored; namely after each set of fields is processed in a multiple-field operation, or when a word boundary is crossed in a record operation. If two control words require updating at the same time, only 10(8) usec are needed. Each refill of a control word takes 3(2) usec.

In the case of the multiple-field format, the instruction time saved for all but the first set of operands is 3(2) usec for one-address, 5(4) usec for two-address, and 14(12) for three-address instructions.

Miscellaneous

No additional time is spent on address monitoring, condition and indicator setting, or noisy mode.

9.2.1 Control Word and Address Arithmetic Instructions

Instructions		Basic time
INCREMENT ADDRESS		10(7)
DIMINISH COUNT		10(7)
INCREMENT ADDRESS AND DIMINISH COUNT		10(7)
DIMINISH COUNT AND REFILL	Refill	12(9)
	No refill	10(7)
INCREMENT ADDRESS, DIMINISH COUNT AND REFILL	Refill	12(9)
	No refill	10(7)
REFILL		12(9)
REFILL FROM ADDRESS		12(9)
STORE EFFECTIVE ADDRESS, FIRST/LAST		6(4)

9.2.2 Decision Instructions

Instructions		Basic time
BRANCH IF ANY/NONE		3(2)
INTERRUPT INTENTIONALLY	Enabled	6(4)
	Disabled	12(8)
LOAD CONDITION REGISTER		7(4)
BRANCH IF ANY/NONE AND PRESERVE PCW	Successful	9(6)
	Unsuccessful	5(4)
INTERRUPT INTENTIONALLY AND PRESERVE PCW	Enabled	12(8)
	Disabled	18(12)

9.2.3 Transmit and Swap Instructions

Instructions	Basic time single operation	Basic time multiple operation
TRANSMIT	11(8)	13(8) + 4W
TRANSMIT TILL MATCH/CLASH	20(16)	20(14)+ 5W
TRANSMIT HALF-WORD	11(8)	----
SWAP	16(13)	----
SWAP HALF-WORD	17(13)	----

Where W is the number of words transmitted.

The multiple-word transmission instruction times do not include the initial indirect addressing to the control words. The control words are kept in the processor registers, and hence do not have to be fetched and stored after each word is transmitted. Each refill of a control word requires 3(2) usec for the first field or 2 usec for the last field.

TRANSMISSION EXAMPLES

Transmit a block of ten words, the last field only indirectly addressing a control word.

Calculation = 13(8) + 10x4 + (3)2 = 56(50) usec

Transmit three blocks of ten words each, to a thirty-word area, the first field being indirectly addressed.

		Calculation		
Transmit	=	13(8) + 30x4 + 2	=	135 (130)
Refill	=	2x3(2)	=	6 (4)
		Total time (usec)	=	141 (134)

9.2.4 Fixed-Point Arithmetic Instructions

Instructions		Basic time
LOAD		14(10) + N(0)
ADD		14(10) + N(0) + [N_ℓ(0) + 7(5)] R
CONVERT		18(14) + [22+ 3N_ℓ] N_f
NUMERIC COMPARE		14(10) + N(0)
ADD AND PLACE		23(18) + N(0) + [N_ℓ(0) + 7(5)] R
MULTIPLY	Decimal	23(18)+N_m [10 +3N_f]
	Hexadecimal	23(18)+N_m [6 +N_f]
FRACTIONAL MULTIPLY	Decimal	23(18)+N_m [10+3N_f]
	Hexadecimal	23(18)+ N_m [6+N_f]
DIVIDE	Decimal	23(18)+N_ℓ [18+ 12N_m]
	Hexadecimal	23(18)+N_ℓ [18+4N_m]

Where N is the number of digits in the longest field addressed,

N_f is the number of digits in the first field addressed,

N_m is the number of digits in the middle field addressed,

N_ℓ is the number of digits in the last field addressed, and

R is one or zero accordingly as recomplementation is or is not required. (Recomplementation is required if the effective signs of the operands differ and the first operand is greater in absolute magnitude than the second.)

In the above formulas, quantities in brackets are factors of a product; those in parentheses are lower bounds (due to overlap) on the immediately preceding numbers. In calculating the number of digits in the various fields addressed, the sign byte should not be included. Processing of the sign, whether present or implied, is included in the fixed-point times.

The following times must be added whenever eight-bit byte, half-word, or full-word boundaries are crossed.

Instructions	Field addressed	Eight-bit boundary	Half-word boundary	Full-word boundary
LOAD, ADD, CONVERT, NUMERIC COMPARE, and ADD AND PLACE	First	---	3(2)	3(2)
	Middle	3(2)	---	---
	Last	---	3(2)	6(4)
MULTIPLY, and FRACTIONAL MULTIPLY (decimal)	First	---	$2N_m$	$2N_m$
	Middle	2	---	---
	Last	---	$2N_r$	$8(6)N_r$
MULTIPLY, and FRACTIONAL MULTIPLY (hexadecimal)	First	---	$3(2)N_m$	$3(2)N_m$
	Middle	2	---	---
	Last	---	$3(2)N_r$	$9(6)N_r$
DIVIDE (decimal)	First	---	$8N_l$	$8N_l$
	Middle	---	$8N_l$	$8N_l$
	Last	---		
DIVIDE (hexadecimal)	First	---	$12(8)N_l$	$12(8)N_l$
	Middle	---	$12(8)N_l$	$12(8)N_l$
	Last	---	---	---

Where N_r is the number of digits to the right of the boundary specified.

The signed-unsigned, byte, sign-inversion, and vary modifiers do not influence fixed-point instruction times. The count modifier will add 7(5) usec to the instruction time; plus 2 usec if a refill is required.

FIXED-POINT ARITHMETIC EXAMPLES

Accumulate the sum of nine unsigned fields, eight of which are in the first address position and one in the last address position. Assume that the first field, of seven-digit length, crosses a half-word boundary and that the last field, of 11-digit length, crosses a full-word boundary.

Operation performed	Calculations	
ADD	8x14(10) - 7x5(4) + 8x11(0) + 2	= 167(54)
Cross half-word boundary (1st field)	8x3 (2)	= 24(16)
Cross full-word boundary (last)	8x6 (4)	= 48(32)
Store initial CW	3 (2)	= 3(2)
Update CW	8x6 (4)	= 48(32)
Total time (usec)		= 290(146)

Note: If the first and last fields were positioned more suitably, 48(32) usec could be saved.

Add a constant field (middle address) to a series of fifty fields (first address) and place the result in a series of fifty fields (last address). Assume that the first and last fields cross no half- or full-word boundaries, and that the middle field crosses only one eight-bit boundary (best case). All fields have four digits.

Operation performed	Calculations	
ADD AND PLACE	50x23(18)+49x14(12)+50x4(0)+5(4)	= 669(316)
Cross eight-bit boundary	50x3 (2)	= 150(100)
Store initial CWs	2x3 (2)	= 6(4)
Update two CWs	50x10(8)	= 500(400)
	Total time (usec) =	1325(820)

Fractional multiply (hexadecimal) the first field (16 four-bit bytes; fifteen digits and a sign) by the middle field (16 four-bit bytes; fifteen digits and a sign), and place the result in the last field (16 four-bit digits; unsigned). Let each field occupy a full word.

Operation performed	Calculations	
FRACTIONAL MULTIPLY	23(18) + 15[6(4) + 15]	= 338(303)
Cross half-word boundary	15x3(2) + 8x3(2)	= 69(46)
Cross eight-bit boundary	7x2	= 14(14)
	Total time (usec) =	411(363)

9.2.5 Floating-Point Arithmetic Instructions

Instructions	Basic time (8-digit precision)	Basic time (12-digit precision)
ADD	26(12)	34
RESET AND ADD	26(12)	34
MULTIPLY	100(28)	280
DIVIDE	274	880
LOAD ACCUMULATOR	7(6)	7(6)
STORE ACCUMULATOR	7(6)	7(6)
Operations		
Normalization	$14(4)N_0 + 6(4)$	$14(4)N_0 + 6(4)$
Recomplementation	20(6)	24(6)
Normalization and recomplementation	$14(4)N_0 + 20(10)$	$14(4)N_0 + 24(10)$

Where N_0 is the number of digit positions to be normalized.

For all floating-point instructions certain conditions, such as propagated extremum, cause the instruction to be terminated early. If the fractional part of the accumulator is unaltered, such early terminations result in an instruction time of 12-14 usec. If the accumulator fraction is replaced, the instruction time is 18-20 usec. Reference to the section in Chapter 5 on the treatment of numbers outside the normal exponent range will determine whether or not the accumulator fraction is altered.

For floating-point instructions, no additional time is required when operands cross half-word boundaries. For one level of indexing add 7(2) usec. For more than one level of indexing, the last level takes 7(2) usec and all earlier levels 9(4) usec.

FLOATING-POINT ARITHMETIC EXAMPLE

Add in the relocation mode two levels of indexing and one level of indirect addressing.

Operation performed	Calculations
ADD (32-bit fraction)	= 26(12)
Index	= 9(4)
Relocate	= 3(2)
Indirect	= 3(2)
Index	= 7(2)
Relocate	= 3(2)
Total time (usec)	= 51(24)

9.2.6 Logical Instructions

Instructions	Basic time
CONNECT/BYTE CONNECT	13(10) + N(0)
CONNECT/BYTE CONNECT TILL MATCH/CLASH	22(18) + N(0)
CONNECT/BYTE CONNECT FOR TEST	13(10) + N(0)
CONNECT/BYTE CONNECT FOR TEST TILL MATCH/CLASH	22(18) + N(0)
CONNECT/BYTE CONNECT AND PLACE	22(18) + N(0)
TRANSLATE	20(16) + 3(2)N
TRANSLATE WITH CARRY LEFT/RIGHT	21(17) + 3(2)N
ALPHAMERIC COMPARE	13(10) + N_h(0)

Where N is the number of bytes in the shortest field or record addressed, and

N_h is the number of identical high-order bytes in two operands.

The following times must be added wherever half-word and full-word boundaries are crossed.

Operations	Field addressed	Eight-bit boundary	Half-word boundary	Full-word boundary
Translation	First	---	2	2
	Last	---	2	5(4)
All others	First	---	3(2)	3(2)
	Middle	3(2)	---	---
	Last	---	3(2)	6(4)

TRANSLATION EXAMPLE

Translate 80 eight-bit characters from one code to another, both fields starting on full-word boundaries.

Operation performed	Calculations	
TRANSLATE	20(16) + 80x3(2) + 5(4)	= 265(180)
Cross boundaries	19x2 + 10x2 + 9x5(4)	= 103(94)
Update CWs	10x10(8)	= 100(80)
Store initial CWs	2x3(2)	= 6(4)
	Total time (usec)	= 474(358)

Note: if both the first and last fields did not cross a word boundary at the same time, the updated CWs time would be 10x6(4)+10x6(4).

9.2.7 Input-Output Instructions

Instructions		Basic time
START CHANNEL		9(7)
RELEASE CHANNEL		11(9)
Control-Word Operations		
Read or Sense	Simplex channel	3(2) + 2B + 13(11) W
	Multiplex channel	3(2) + 12(9)B
Write or Control	Simplex channel	7(6) + 2B + 16(14) [W - 1]
	Multiplex channel	3(2) + 12(9)B

Where B is the number of bytes processed, and
W is the number of words processed.

Attention Signals		
Channel not busy (Including storage of instruction byte)		14(11)
Channel busy	Simplex channel with Write or Control	11(8)
	Otherwise	7(3)

Operations	
End (bits 29 and 30 of CW set 00 or 10 regardless of SNT command)	8(6)
Store interruption byte for input-output termination	8(6)
Refill (bits 29 and 30 of CW set 01 or 11)	6(4)
Restart (bits 29 and 30 of CW set 10)	8(5)

Normally, each refill of an input-output control word requires 3(2) usec. However, if an operation ends before all data word areas (defined by 01 or 11 codes) are used, each refill required before a control word containing a zero in bit 30 is reached, will require 8(6) usec.

The skip flag does not influence the execution time of any input-output instruction.

The Attention signals refer to signals initiated at the input-output units themselves. The time required to store interruption codes resulting from Attention signals is included.

The times quoted refer to the time during which storage and/or processor are engaged by input-output activities. The real-time elapsed between the beginning and the end of an input-output operation depends on the units addressed.

INPUT-OUTPUT EXAMPLE

An input-output chain of Read's on a simplex channel.

Commands		Setting of bits 29, 30	SNT
Control (select)		10	On
Read		01	--
Read	(750 characters)	01	--
Read		10	Off
Read	(1000 characters)	10	Off
Control	(rewind)	00	On

Operations performed	Calculations		
START CHANNEL		=	9 (7)
Control	7(6) + 2x2 + 16x0	=	11 (10)
End		=	8 (6)
Refill CW		=	8 (5)
Read	3(2) + 750x2 + 94x13(12)	=	2725(2630)
End		=	8 (6)
Refill CWs	2x6(4) + 8(5)	=	20 (13)
Read	3(2) + 1000x2 + 125x13(12)	=	3628(3502)
End		=	8 (6)
Refill CW		=	8 (5)
Control	7(6) + 1x2 + 16x0	=	9 (8)
End		=	8 (6)
Store two End bytes in interruption stream		=	16 (12)
	Total time (usec)	=	6466(6216)

9.2.8 Time Clock and Interval Timer

The interval timer is updated every 1/1024 of a second and the time required for this updating is 10(7) usec. The time clock, updated approximately every 1/10 of a second, is advanced simultaneously with the interval timer, and hence has no call on machine time.

8.8 MARGINAL CHECKING

Jack plugs have been provided in the processor so that a portable buck-boost d-c supply can be connected in series with the -12 volt power supply for two independent sections of the processor.

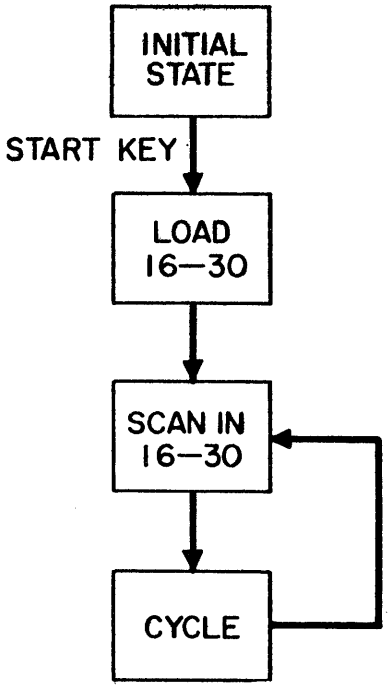


Figure 8.8 - Loop mode

APPENDIX I

8106 INSTRUCTION CODE

When an option exists between choices of a modifier, the first mentioned option is indicated by a zero in the appropriate bit position. The symbol used in a bit position always indicates that the second of the two choices is used when the bit is a one. Bit positions marked z may be set to either one or zero.

ONE-ADDRESS INSTRUCTIONS

Bits 24-31

Floating-Point Arithmetic (indexed)

```
i u a 0 1 n c c
i                       Direct-Indirect
  u                     Normalized-Unnormalized
    a                   Sign-Absolute
          n             Positive-Negative
            0 0         RESET ADD
            0 1         ADD
            1 0         MULTIPLY
            1 1         DIVIDE
```

Accumulator Operations (indexed)

```
i z 0 0 0 1 c c
i                       Direct-Indirect
            0 0         LOAD ACCUMULATOR
            0 1         STORE ACCUMULATOR
            1 0         STORE EFFECTIVE ADDRESS, FIRST
            1 1         STORE EFFECTIVE ADDRESS, LAST
```

Decision Operations

```
i z 0 0 0 0 c c
i                       Direct-Indirect
            0 0         INTERRUPT INTENTIONALLY
            1 0         LOAD CONDITION REGISTER
            0 1         BRANCH IF ANY
            1 1         BRANCH IF NONE
```

Control-Word Operations (indexed)

```
i z 1 0 0 c c c
i                       Direct-Indirect
          0 0 0         REFILL FROM ADDRESS
          0 0 1         REFILL
          0 1 0         DIMINISH COUNT
          0 1 1         DIMINISH COUNT AND REFILL
          1 0 0         INCREMENT ADDRESS
          1 0 1         INCREMENT ADDRESS AND REFILL
          1 1 0         INCREMENT ADDRESS AND DIMINISH COUNT
          1 1 1         INCREMENT ADDRESS, DIMINISH COUNT,
                              AND REFILL
```

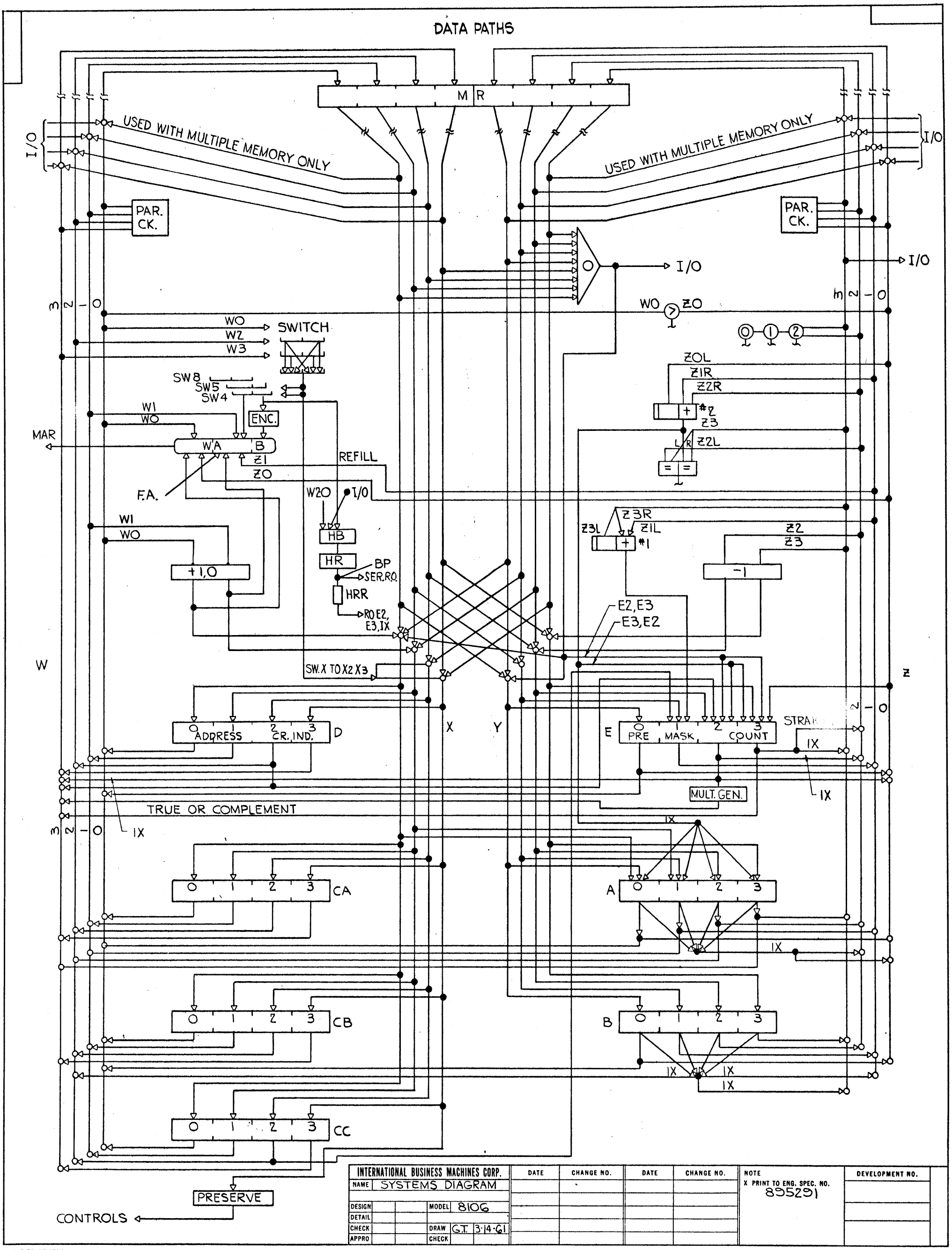
TWO-ADDRESS INSTRUCTIONS

Bits 24-31	Bits 56-63	
Integer Arithmetic		
i 8 u 1 0 1 c c	i 8 u 0 d n c v	
i	i	Direct-Indirect
8	8	Four-Eight
u	u	Signed-Unsigned
	d	Hexadecimal-Decimal
	n	Positive-Negative
	c	Count
	v	Vary
0 0		LOAD
0 1		ADD
1 0		CONVERT
1 1		COMPARE
Connective Operations		
i 8 1 1 0 0 0 c	i z t 0 x x x x	
i	i	Direct-Indirect
8		Four-Eight
	t	Store-Test
	x x x x	Connective code
z 0		CONNECT
8 1		BYTE CONNECT
Alphameric-Comparison Operations		
i z 1 1 0 0 1 0	i z z 0 z z z z	
i	i	Direct-Indirect
Transmission Operations		
i z 0 1 0 0 1 0	i z z 0 z z c h	
i	i	Direct-Indirect
	h	Full word-Half word
	0	TRANSMIT
	1	SWAP
Decision Operations		
i z 0 1 0 0 c c	i z z 0 z z z z	
i	i	Direct-Indirect
0 0		INTERRUPT INTENTIONALLY AND PRESERVE PCW
0 1		BRANCH IF ANY AND PRESERVE PCW
1 1		BRANCH IF NONE AND PRESERVE PCW
Channel Operations		
i z 1 1 0 0 1 1	i z z 0 z z z c	
i	i	Direct-Indirect
	0	START CHANNEL
	1	RELEASE CHANNEL

THREE-ADDRESS INSTRUCTIONS

Bits 24-31	Bits 56-63	Bits 88-95	
Integer Arithmetic			
i 8 u l l l c c	i 8 u l z z z z	i 8 u 0 d n c v	
i	i	i	Direct-Indirect
8	8	8	Four-Eight
u	u	u	Signed-Unsigned
		d	Hexadecimal-Decimal
		n	Positive-Negative
		c	Count
		v	Vary
0 0			ADD AND PLACE
0 1			FRACTIONAL MULTIPLY
1 0			MULTIPLY
1 1			DIVIDE
Connective Operations			
i 8 l l l 0 c c	i 8 c l z z z z	i z t 0 x x x x	
i	i	i	Direct-Indirect
8	8		Four-Eight
	c		Match-Clash
		t	Store-Test
		x x x x	Connective code
z 0 0	z z		CONNECT AND PLACE
z 0 1	8 z		BYTE CONNECT AND PLACE
z 1 0	8 c		CONNECT TILL MATCH-CLASH
8 1 1	8 c		BYTE CONNECT TILL MATCH-CLASH
Translation Operations			
i 8 0 1 1 0 c c	i 8 z l z z z z	i 8 z 0 z z z z	
i	i	i	Direct-Indirect
8	8	8	Four-Eight
0 0			TRANSLATE
0 1			TRANSLATE WITH LEFT CARRY
1 1			TRANSLATE WITH RIGHT CARRY
Transmission Operations			
i 8 0 1 1 0 1 0	i 8 c l z z z z	i z z 0 z z z z	
i	i	i	Direct-Indirect
8	8		Four-Eight
	c		Match-Clash

DATA PATHS



INTERNATIONAL BUSINESS MACHINES CORP.						DATE	CHANGE NO.	DATE	CHANGE NO.	NOTE	DEVELOPMENT NO.
NAME	SYSTEMS DIAGRAM									X PRINT TO ENG. SPEC. NO. 895291	
DESIGN			MODEL	8106							
DETAIL											
CHECK			DRAW	G.T.	3-14-61						
APPRO			CHECK								

←≪ Fold out

Appendix II - Data flow paths in the 8106

APPENDIX II

DATA FLOW PATHS IN THE 8106

APPENDIX III

CORRECTIONS TO THE MANUAL AND MACHINE CHANGES

This appendix contains final corrections to the manual and machine changes to document the current state of planning in the 8106. Some of the corrections are "rewrites" to clarify or re-emphasize a point; others are changes in hardware. Each correction is stated, then the appropriate sections of the manual are rewritten to incorporate the change. The rewritten material is designated by page number and line number from the top (counting all headings and lines) so that the holder of a manual may paste the correction over the original copy, if desired.

Hardware Change--CONVERT Instruction

Bit 28 of the last half-word of instruction refers to the radix of the middle field. If bit 28 is zero, conversion is from decimal to hexadecimal; if one, vice versa.

(p 4.4, replace paragraphs starting at line 6)

In the last half-word of instruction, bit 28 specifies whether the radix is decimal (1) or hexadecimal (0). In the CONVERT instruction, bit 28 of the last half-word refers to the radix of the middle field. If bit 28 is zero, conversion is from decimal to hexadecimal; if one, vice versa. Bit 29, in conjunction with bit 26 of the first half-word of instruction, determines the effective sign of the first operand, as described under Sign Control.

Finally, bits 30 and 31 of the last half-word of instruction are the count and vary modifiers, respectively.

(p 4.10, replace paragraph starting at line 28)

The CONVERT operation enables data to be converted from decimal to hexadecimal, or vice versa. The data field addressed by the first effective address is converted as an integer from decimal to hexadecimal if the radix modifier is zero; and in the opposite manner if the modifier is one. The result is placed in the field specified by the last effective address. If the last field is not large enough to contain the significant digits of the result, the Oversized Result indicator will be turned on. Note that bit 28 of the last half-word of this instruction refers to the radix of the middle field.

Clarification--ADD and ADD AND PLACE Instructions

A zero result in an ADD or an ADD AND PLACE instruction will have the sign of the second field.

(p 4.10, replace paragraph starting at line 23)

The sign modifier and the signed-unsigned modifier provide means for performing subtraction and absolute-value operations. A zero signed result will have the sign of the last field.
(end of paragraph)

(p 4.11, replace three lines starting at line 18)

Subtraction and absolute-value operations may be performed, as in ADD. A zero result will have the sign of the middle field.
(end of paragraph)

Clarification--Condition Register Setting

The zero-result bit in the condition register will be turned on if the result field (A) is zero, even if the Oversized Result indicator is on.

(p 4.8, replace paragraph starting at line 9)

Bit 1 is on when the arithmetic result is zero (plus or minus) even if the Oversized Result indicator is on. Also, bit 1 is on in numeric-comparison operations when the second operand is equal to the first (plus zero = minus zero).
(end of paragraph)

Hardware Change--Instruction Codes

Interchange the codes for FRACTIONAL MULTIPLY and ADD AND PLACE.

(p A-7, interchange codes starting at line 12)

01 (for ADD AND PLACE)
00 (for FRACTIONAL MULTIPLY)

Hardware Change--Programmed Scan

If the processor is disabled, the input-output operation START CHANNEL 1 will initiate the sequence of system operations scan-in, cycle, and scan-out. Processing will then be resumed with a new program control word obtained from location 16. If the processor is enabled, a START CHANNEL 1 instruction will cause a Channel Not Operational alert.

(Correction to p 7.6 has been combined with another correction for that page)

(p 8.14, replace three lines starting at line 9)

If the input-output instruction START CHANNEL 1 is given when the system is disabled, the programmed scan procedure shown in Figure 8.4 is executed.

(p 8.14, replace three lines starting at line 18)

location 16. If the processor is enabled, a START CHANNEL 1 instruction will cause a Channel Not Operational alert.

(end of paragraph)

Clarification--Storage and CPU Thermal Lights

The Storage and the CPU Thermal Lights will turn on before the thermal-protection switch (which shuts off all power) is tripped.

(p 8.2, add to end of the page)

The Storage Thermal Light and the CPU Thermal Light will turn on before the thermal-protection switch (which shuts off all power) is tripped.

Hardware Change--Time Signal and Interval Timer

The input-output instruction START CHANNEL 0 causes the contents of the Interval Timer to be decremented by one every millisecond. When the count reaches zero, the Time Signal interruption will be stored in the interruption stream. The decrementation continues. Decrementation of the Interval Timer is stopped when the input-output instruction RELEASE CHANNEL 0 is executed. When in operation, the Interval Timer consumes approximately one per cent of the available time. When inoperative, no time is consumed by the timer.

(p 7.6, replace paragraphs starting at line 1)

START CHANNEL (SRT), START CHANNEL 1, and 0

The START CHANNEL instruction initiates the sequence of input-output operations specified by the chain of control words which it addresses. If the addressed channel is busy or not operational, the appropriate interruption code is stored and no operation is initiated.

The instruction START CHANNEL 1 initiates a programmed scan procedure if the processor is in the disabled mode. If the processor is enabled, this instruction causes a Channel Not Operational alert.

Another instruction--START CHANNEL 0--causes the contents of the Interval Timer to be decremented by one every millisecond. When the count of the timer reaches zero, a Time Signal interruption will be stored.

RELEASE CHANNEL (RLS) and RELEASE CHANNEL 0

If the addressed channel is busy, RELEASE CHANNEL causes information transfer to be terminated immediately. Any alert which results from an input-output operation terminated by this instruction is not suppressed. The first address given in this instruction is not used in the operation, but it may cause an Empty Address alert if the instruction occurs in the relocation mode.

The instruction RELEASE CHANNEL 0 terminates the decrementation of the Interval Timer.

(p 3.21, replace paragraph starting at line 27)

This is code 4 and it is stored when the value of the Interval Timer goes from one to zero. Decrementation of the timer can be initiated by the special instruction START CHANNEL 0 and can be stopped by the special instruction RELEASE CHANNEL 0.

(Continued--hardware change on the interval timer)

(p 2.36, add to end of page)

The input-output instruction START CHANNEL 0 initiates the decrementation of the Interval Timer by one every millisecond. Decrementation of the timer is stopped when the instruction RELEASE CHANNEL 0 is executed. During operation, the Interval Timer consumes approximately one per cent of available time. When inoperative, it consumes no time.

Clarification--Unended Sequence (US)

Interruption code 7 is stored when any operation is not completed in one-tenth of a second; for example, if a START CHANNEL or RELEASE CHANNEL instruction is still indirecting after the time limit.

(p 3.22, replace three lines starting at line 9)

This is code 7 and it is stored when any operation is not completed in one-tenth second, due, for example, to an unended sequence of refill addresses or indirect-addressing references. The operation is terminated.

Clarification--System Mode Switch

The System Mode Switch is a six-position rotary switch used to select the system mode. The normal mode is used for production operation.

(p 8.4, replace paragraphs starting at line 1)

8.2.2 System Mode Switch

This is a six-position rotary switch used to select the system mode. The modes are listed below along with their uses.

NORMAL Mode

The normal mode is used for normal production operation. It is discussed in section 8.4 .

(Continued--system mode switch clarification)

(p 8.10, replace line 15)

depressed. The system must be placed in the normal mode.

(p 8.10, replace line 21)

in the normal mode are illustrated in Figure 8.4 .

(p 8.11, change caption of figure)

Figure 8.4 - The normal mode

(p 8.13, replace paragraph starting at line 9)

Malfunctions are detected only when the processor is executing instructions or input-output operations in the normal mode. The program is restarted after the recording operation.

(delete rest of paragraph)

(p 8.3, correct Figure 8.1 by deleting the reference to AUL on the scan mode switch)

(p 8.11, correct Figure 8.4 by deleting reference to AUL, the connecting lines to and from the block labeled UNLOAD 16-30, and the block itself.)

Clarification--Fixed-Point Instructions

The multiple-field feature is available for all fixed-point instructions, except DIVIDE. Only with an ADD, a LOAD, or a NUMERIC COMPARE instruction may either the first operand or the last, or both, employ the multiple-field format.

(p 4.3, replace paragraph starting at line 11)

The multiple-field feature is available for all fixed-point instructions, except DIVIDE. Only with ADD, LOAD, or NUMERIC COMPARE may either the first operand or the last, or both, employ a multiple-field format. With CONVERT, ADD AND PLACE, MULTIPLY, or FRACTIONAL MULTIPLY, the first and last operand must be in the same format; both must be single fields or multiple fields. When one of the operands in these four instructions is a single field, the instruction will terminate after a single operation. The middle address of a three-address instruction cannot refer to a multiple field. Should the middle address refer to a long data description, only the left half-word will be used as a short data description.

Clarification--Selecting the Four Bits of an Eight-Bit Byte

In the eight-bit mode, the low-order bit of the limit address determines which four bits of the eight-bit byte are to be used in numeric processing.

(p 4.5, replace paragraph starting at line 7)

The low-order bit of the byte and limit addresses is ignored in addressing the eight-bit byte. In the eight-bit mode, the low-order bit of the limit address determines which four bits of the eight-bit byte are to be used. The left or right four bits of the eight-bit byte will be used accordingly as the low-order bit of the limit address is zero or one.

Clarification--System Panel

A system panel is provided for manual operation of the system by an operator or customer engineer. This panel can be located on the right-hand side of the universal console, or it can be located in an independent unit which contains a suitable plug for connection to the main frame.

(p 8.1, replace paragraph starting at line 18)

A system panel is provided for manual control of the system by an operator or customer engineer. The panel can be located on the right-hand side of the universal console, or it can be located in an independent unit which contains a suitable plug for connection to the main frame. The panel contains the keys, switches, and lights necessary to control the processor, the power supplies, and the scanner. A set of lights indicates the state of the system and signals the operator if a malfunction occurs from which the processor cannot recover automatically. A complete set of controls is provided for the manual location of faults in the scanner controls by means of a single-cycling technique, and similar but rudimentary controls are provided for the processing and input-output controls. Facilities have also been provided for executing any operations or part of an operation repeatedly so that an oscilloscope can be used to locate faults.

Clarification--Chapter 5, addition of a page, 5.13

The Sign of a Zero Exponent

In cases where normalization causes a zero exponent, the sign of the exponent is the same as that of the nonzero exponent before normalization. Where no normalization is required, the sign of a zero exponent is the same as that of the exponent of the operand originally in the accumulator, with two exceptions. One, in ADD, if the addressed operand has a zero exponent, and the exponent of the accumulator operand is less than zero, the sign of the zero exponent is preserved. Two, in RESET ADD, the sign of a zero exponent of the addressed operand is carried into the accumulator.

Operands with Zero Fraction

(follows last paragraph of section 5.1.6)

An operand with a zero fraction and exponent in the normal range will behave like any other number with a normal-range exponent. However, a Generated Extremum Positive condition will take precedence over a Generated Extremum Negative condition. For example, if the sum of the exponents in a multiplication is greater than 255, the Generated Extremum Positive indicator will be turned on. The Generated Extremum Negative indicator will not be affected, in spite of the fact that if the multiplication were completed a zero fraction would result.

Clarification--FRACTIONAL MULTIPLY Instruction

The correct operation of FRACTIONAL MULTIPLY requires that the field addressed by the first effective address contain the same number of digits as the result field addressed by the last effective address; e.g., a first field consisting of eight 8-bit bytes (eight digits) with a last field of sixteen 4-bit bytes (sixteen digits) will turn on Oversized Result.

(p 4.9, replace paragraphs starting at line 1)

The indicator will also be turned on if a negative result arises when an unsigned result has been specified. Oversized Result will be turned on in FRACTIONAL MULTIPLY if the number of digits in the field addressed by the first effective address differs from the number of digits in the result field addressed by the last effective address. Oversized Result will also be turned on if the number of digits in the middle effective address is not a multiple of the field addressed by the first effective address.

Zero Divisor

This indicator is turned on when a divisor is zero. The operation is suppressed, except that the sign byte is placed in a signed quotient field.

(p 4.12, replace paragraph starting with line 11)

Correction operation of this instruction requires that the field addressed by the first effective address contain the same number of digits as the result field addressed by the last effective address; e.g., a first field consisting of eight 8-bit bytes (eight digits) and a last field of sixteen 4-bit bytes (sixteen digits) would be incorrect. Also, the number of digits in the field addressed by the middle effective address must be a multiple of the number of digits of the first field. If either of these conditions is not met, Oversized Result will be turned on.

Editor's Error--Operand Designation

(p 2.1, replace paragraph starting with line 28)

To permit operations which require one, two, or three addresses, a variable-length instruction format is used. Instruction addresses may be modified by indexing or indirect addressing.

APPENDIX IV

THE IBM 8104 DATA PROCESSING SYSTEM

THE IBM 8104 DATA-PROCESSING SYSTEM

1. Description

The IBM 8104 will be the smallest scientific member of the 8000 series of systems compatible with the IBM 8106. While basically a scientific computer, the data-processing ability of the 8104 will be such that it can be used for either scientific or data-processing applications. The system can operate in either single- or double-precision mode, and the programmer can specify the mode through the program control word.

The single-precision format occupies a half-word--32 bits with a 24-bit fraction, and the double-precision format occupies a full-word--64 bits with a 48-bit fraction. Thus, to change precision, the programmer will at most reassemble his program and modify his data. Furthermore, prior to running a single-precision 8104 program on the 8106, the programmer will have to reassemble his data and place it in the double-precision format. All programs not using single-precision floating point will be load-deck (i.e., binary) compatible with the 8106.

Assuming a 40 per cent use of single precision and a 60 per cent use of double precision, the 8104 will have a performance of approximately 1.2 times that of the IBM 709 on a "Gibson" mix.

2. Instruction Set

The instruction set of the 8104 will be a subset of the 8106 instruction set, and will utilize the same machine codes. The following is a list of the 8104 instructions and modifiers.

One-Address Instructions

FLOATING ADD -- Modal
FLOATING MULTIPLY -- Modal
FLOATING DIVIDE -- Modal
LOAD ACCUMULATOR -- Modal
STORE ACCUMULATOR -- Modal
INCREMENT ADDRESS
DIMINISH COUNT
INCREMENT ADDRESS AND DIMINISH COUNT
LOAD CONDITION REGISTER
BRANCH IF ANY
BRANCH IF NONE
INTERRUPT INTENTIONALLY

Two-Address Instructions

ADD
NUMERIC COMPARE
TRANSMIT
TRANSMIT HALF-WORD
BRANCH IF ANY AND PRESERVE PCW
BRANCH IF NONE AND PRESERVE PCW
INTERRUPT INTENTIONALLY AND PRESERVE PCW
BYTE CONNECT
CONNECT
BYTE CONNECT FOR TEST
CONNECT FOR TEST

Three-Address Instructions

MULTIPLY
FRACTIONAL MULTIPLY
DIVIDE
TRANSLATE
TRANSLATE WITH RIGHT CARRY
TRANSLATE WITH LEFT CARRY

Instruction Modifiers

Direct-Indirect -- can be used with all instructions
Indexing -- same restrictions as in the 8106
Sign inversion -- can be used with all arithmetic instructions
Signed-Unsigned/Absolute -- can be used with all arithmetic instructions
Normalized-Unnormalized -- can be used with all floating-point instructions
Radix -- hexadecimal only
Vary -- can be used with fixed-point ADD
Byte size -- all logical instructions can specify a byte size of four or eight bits. However, the following restrictions must be observed. For the TRANSLATE instruction, the table and destination field must specify the same byte size. TRANSLATE WITH CARRY LEFT/RIGHT instructions must have a table byte size of eight, and a source and destination byte size of four
Connectives -- all sixteen logical connectives are provided

When the precision-mode bit in the program control word is on (i.e., the system is operating in the single-precision floating-point format) all the one-address instructions designated "Modal" in the list on page 1 will have their effective address shifted right one bit position (i.e., halved) when the address is placed in the storage-address register. The low-order bit will be used to determine which half-word of the 32,768 possible words addressable by the remaining 15 bits is to be used. Thus, the 8104 can address a possible 65,536 floating-point operands, regardless of the precision mode being used.

All fixed-point arithmetic instructions (ADD, MULTIPLY, DIVIDE, and NUMERIC COMPARE) will operate only on half-word formats (i.e., 32-bits). If unsigned arithmetic is specified, the sign byte, which occupies four bits, is ignored. Variable field length is possible with all of the logical connectives and the translation instructions. Fields operated upon by the connectives must fall within a half-word, and fields operated upon by the translation instructions must fall within a full word. All data descriptions indirectly addressed by the 8104 will have bit 31 set to zero by the assembly program. Thus, if an 8104 program is run on an 8106, the data descriptions will be interpreted by the 8106 as short data descriptions.

The 8104 will have 255 index words, with indirect addressing and indexing having the same relationship as in the 8106.

3. Operational Characteristics

All floating-point operations will use a 64-bit register accumulator. An eight-bit serial adder will perform all arithmetic.

Storage will have an access time of 8 usec, and a storage word will contain 32 information bits and 4 check bits. Two storage words will be required for one logical word. The minimum storage size will be 2,048 logical words. Initially, the system will be offered with storage sizes of 2,048, 4,096, or 8,192 logical words.

The 8104 will have an interruption system like the one provided in the 8106, with an interruption queue, and store and fetch interruption control words. Program control words will be compatible with those of the 8106. There will be some form of storage protection on the 8104 which will be compatible with the storage-protection concept of the 8106.

A minimum of four channels will be provided, all of which can operate simultaneously. One will be used for a program-controllable typewriter attached to the console as a standard device. Two of the channels will have a maximum data rate of at least 7,000 bytes per second (bps), and the fourth will have a maximum data rate of 50,000 bps. The following two channels are field installable: a second high-speed channel which can operate alone at 50,000 bps and at a reduced rate if the other high-speed channel is operating, and a multiplexing channel with a maximum data rate of at least 7,000 bps. All input-output units will be connected to the system through the DSD standard interface.

The input-output control words will have a count but no refill address, nor will there be chaining between control words.

The operations and maintenance console will be a part of the central-processing unit.

4. Installation Characteristics

A. The 8104 will be housed in a rack-and-panel arrangement. Normal colors will be available.

B. Cabling and raised-floor requirements will be the same as those for the 8106.

C. Power requirements will be approximately 3 kva. Voltage, frequency variation, and transients will be the same as for the 8106.

D. Temperature, humidity, vibration, and shock limitations will be the same as for the 8106.

5. Noise Criteria

The 8104 will not exceed the maximum noise level specified by DSD Acoustical Standard 1-1-1710-6.

APPENDIX V

THE IBM 8108 HIGH-SPEED FLOATING-POINT ATTACHMENT

THE IBM 8108 HIGH-SPEED FLOATING-POINT ATTACHMENT

1. General Description

The IBM 8108 will be an attachment to the IBM 8106 to provide increased internal performance for all floating-point and indexing operations. Floating-point operations of the 8106 will utilize the 8108 to gain speed, but will produce results logically and mathematically identical to those achievable in the 8106 alone.

The following table indicates the increased performance obtainable through attachment of an 8108 to the 8106 central-processing unit (times are in microseconds).

	8106 Instruction Time		8108 Instruction Time	
Instructions	8-digit precision	12-digit precision	8-digit precision	12-digit precision
RESET ADD	26	34	10	10
ADD	26	34	10	10
MULTIPLY	100	280	19	24
DIVIDE	274	880	24	28
LOAD/STORE ACCUMULATOR	7	7	5	5
Operation				
Indexing	7	7	3	3

The 8108 will act in lieu of, and not in addition to, the floating-point arithmetic facilities of the 8106. The same set of floating-point instructions available on the 8106 will operate in the 8106-8108 combination. Complete upward compatibility from the 8106 to the 8106-8108 combination will be the design objective. Thus the operational characteristics will be identical with those described in Chapter 5 of the current 8106 operating manual. In fact, the 8108 will be strictly a parallel floating-point engine equipped with a double-length (24 hexadecimal digits) accumulator, and facilities for sign and exponent manipulation. All decoding of instructions, address modification other than indexing, and transferring of data between the 8108 and storage in the 8106 will remain the responsibility of the 8106; and the times for such functions will remain unaffected.

2. Upwards Program Compatibility

All numeric quantities involved in the input to and output from the 8108 will be identical to those involved in floating-point operations on the 8106. The only possible program incompatibility acceptable will be that due to the increased speed at which programs will be run in the 8108. Thus, a program executed in the 8106-8108 combination might overtake an input-output operation and cause a condition that could not exist in the 8106 system alone.

Since the 8108 will have a hardware accumulator while the 8106 has an implied accumulator at storage location prefix + 4, certain conditions will be specified under which the 8108 accumulator will be stored at the location prefix + 4, and under which the location prefix + 4 will be loaded into the hardware accumulator, if strict program compatibility is to be maintained. To this end a trigger will be provided which will be off or on accordingly as the accumulator contents are in prefix + 4 or in 8108 hardware. The situations resulting from the shuttling of accumulator contents between the 8108 accumulator register and 8106 storage location prefix + 4 are described below.

a) The 8108 accumulator is stored in prefix + 4 and the trigger is turned off when --

1) the current instruction is multiple-address, or

2) there is an interruption.

b) The contents of location prefix + 4 are loaded into the 8108 accumulator and the trigger is turned on when the current instruction is a floating-point or STORE ACCUMULATOR instruction.

c) The LOAD ACCUMULATOR instruction loads the 8108 accumulator and turns on the trigger.

The effect of these arrangements will be to leave the accumulator contents in the 8108 accumulator registers for strings of floating-point instructions, and to leave the 8108 accumulator register dormant for strings of nonfloating-point instructions.

Two minor areas of possible incompatibility are covered by the following restrictions.

1) The 8108 accumulator register will be stored in location prefix + 4 for any instruction, including a floating-point instruction whose first effective address is prefix + 4, whenever the trigger is on. This, for example, preserves the legitimacy of squaring the contents of the accumulator by means of a floating MULTIPLY instruction addressing location prefix + 4.

2) Index addresses 0, 1, 4, and 5 are rendered inoperative. This answers any difficulties raised by attempting to index the backup locations for the 8108 accumulator register.

The loading or storing of the 8108 accumulator will add 3 usec to the time of the causative operation.

3. Double-Precision Arithmetic

To take advantage of the double-precision potential inherent in the 8108 double-length accumulator, a precision-mode bit will be provided in the program control word. There will be no extra instruction codes, but the floating-point instructions will be interpreted slightly different in the double-precision mode. Factors common to all floating-point instructions in the double-precision mode are --

a) Whenever the high-order accumulator fraction and the accumulator exponent are stored in location prefix + 4, the low-order accumulator fraction, furnished with the appropriate exponent, will be stored in location prefix + 5. If the accumulator exponent is less than or equal to -244, the resulting floating-point number in prefix + 5 will be in the extremum negative condition. Similarly, whenever the contents of prefix + 4 are loaded into the 8108 accumulator, the contents of location prefix + 5 will be loaded into the low-order accumulator. The exact conditions under which this storing and loading take place have been described.

b) During normalization of a floating-point result, the entire 24-digit accumulator fraction is shifted. Only if the entire result fraction is zero will the Generated Extremum Negative indicator be turned on and the extremum bit be given a value of one.

c) There is no double-precision noisy mode and no eight-digit double-precision mode.

d) Direct-indirect addressing, indexing, signed-unsigned, and normalized-unnormalized modifications to the basic floating-point instructions will be available in the double-precision mode.

No special load or store instructions will be provided for the low-order accumulator; the low-order accumulator fraction will be loaded and stored by programming. A complete double-precision number may be loaded into the accumulator by the LOAD ACCUMULATOR instruction followed by an ADD instruction. The TRANSMIT instruction, whose first address is prefix+ 5, will serve to store the low-order accumulator in an acceptable amount of time (17 usec).

With the double-precision mode, floating-point operations in which double-precision results are obtained from double-precision operands may conveniently be programmed.

Comments pertinent to the individual floating-point instructions and the LOAD/STORE ACCUMULATOR instructions are listed below.

LOAD ACCUMULATOR

This instruction will load the high-order accumulator and clear the low-order accumulator.

STORE ACCUMULATOR

This instruction will store the high-order accumulator and leave the low-order accumulator undisturbed.

RESET ADD

The low-order accumulator will be cleared at the start of this instruction.

ADD

If the exponent of the operand from storage is greater than or equal to the exponent of the operand in the accumulator, the double-precision accumulator operand will be added to the single-precision storage operand to give a double-precision sum. If, however, the storage exponent is less than the accumulator exponent, the low-order fraction of the accumulator operand will be lost. In this case, a single-precision accumulator operand will be added to a single-precision storage operand to give a double-precision sum.

MULTIPLY

The low-order fraction of the accumulator operand will be ignored. A single-precision accumulator operand will be multiplied by a single-precision storage operand to yield a double-precision product.

DIVIDE

A double-precision dividend in the accumulator will be divided by a single-precision divisor from storage to yield a single-precision quotient. The remainder will not be preserved, but may be obtained by programming. At the completion of a division, the low-order accumulator fraction will be cleared.