IBM 8104 Data Processing System

Preliminary Manual of Operation
January 10, 1961

IBM Product Development Laboratory

Poughkeepsie, N. Y.

Chapter 1

IBM 8104 Data Processing System

Contents

Chapter 2

Operand Designation

Contents

(Contents - Chapter 2)

Chapter 3

Instruction Sequencing

Contents

Chapter 4

Variable Field-Length Data Handling

Contents

Chapter 5

Floating Point Arithmetic

Contents

Chapter 6

Input-Output Operations

Contents

Chapter 1

IBM 8104 Data Processing System

1.1 Introduction

The IBM 8104 Data Processing System provides technical processing ability at a low overall systems cost. While a typical systems configuration would include a limited number of low cost input-output units, the system, being part of the 8000 series, permits attachment of higher performance external devices. Similarly, the memory capacity can be expanded as the size of the problems to be solved grows. Finally, the 8104 is purposely designed to provide an easy transition to the more powerful 8106 and 8112 processing systems.

1.2 System Organization

The basic system consists of a central processing unit, a core storage unit and input-output devices. Information moves between input-output devices and core storage under control of channels which are part of the central processing unit.

1.2.1 Storage Units

The computing system uses core storage units with a read-write cycle time of 8 μseconds. A word consists of 16 information bits and 2 non-addressable redundancy bits. The address space in instructions provides for addressing of 2^{16} (65,536) word locations. Storage units consist of 8192 or 16,384 words. Multiple storage units can be used up to the maximum addressing capacity.

1.2.2 Input and Output

Input to the system passes from the input devices to core storage through the channels of the central processing unit. The channel assembles 16-bit words from the 8-bit input information and stores the assembled words in core storage. The program of the central processor starts the input operation. When transmission is finished, the processor is signalled. The channels operate similarly for output. The processor proceeds with computation while the channel completes the operation.

The basic system has one channel. Individual input or output devices can be attached via their control units to the channel. The channel also permits attachment of a multiplexing unit. With this unit, the channel becomes logically equivalent to several channels which can run simultaneously. The system can be expanded by the addition of simplex channels. A total of two of these channels can be provided. Simplex channels have no multiplexing abilities. Their data rate, however, is much higher than the rate of the multiplex channel. The multiplex channel can process a maximum data rate of 10,000 characters per second. The simplex channels can process a maximum rate of 80,000 characters per second.

1.2.3 Central Processing Unit

The central processing unit is designed to provide flexible floating point arithmetic operation as required for technical computation. Normalized and unnormalized operation is provided. Numbers outside the normal exponent range, such as zero and quasi-infinite, are handled in a straightforward manner by using an extremum bit. Full sign modification is provided for all operands. A noisy mode is available to perform significance tests. The floating point format consists of a 32-bit fraction portion and 16 bits for exponent, fraction sign, exponent sign, and extremum bit. Since technical computation also requires efficient decision instructions, a complete set of branch instructions is provided. Variable-field-length arithmetic and logical operations are available to facilitate housekeeping and editing. These operations include fixed point arithmetic and logical connective operations. A maximum field length of 16 bits is allowed.

All instructions occupy 32 bits.

Besides the functions provided by the instruction set, several overall machine features are incorporated in the processor.

An interrupt mechanism is provided to allow input-output alerts, program alerts, and result alerts to interrupt the computation. The result alerts set a group of four indicators. The programmer has the option to allow an interruption by these result alerts, or not, depending upon the setting of four mask bits. As an optional feature, an interval timer and a real time clock are provided.

Address modification by indirect addressing is provided for all instructions. The floating point instructions furthermore permit address modification by additive indexing. Since vital machine information is stored at the first addresses of storage, an address monitoring function is provided which prevents erroneous store operations to all locations with addresses below 1024 or below 2048.

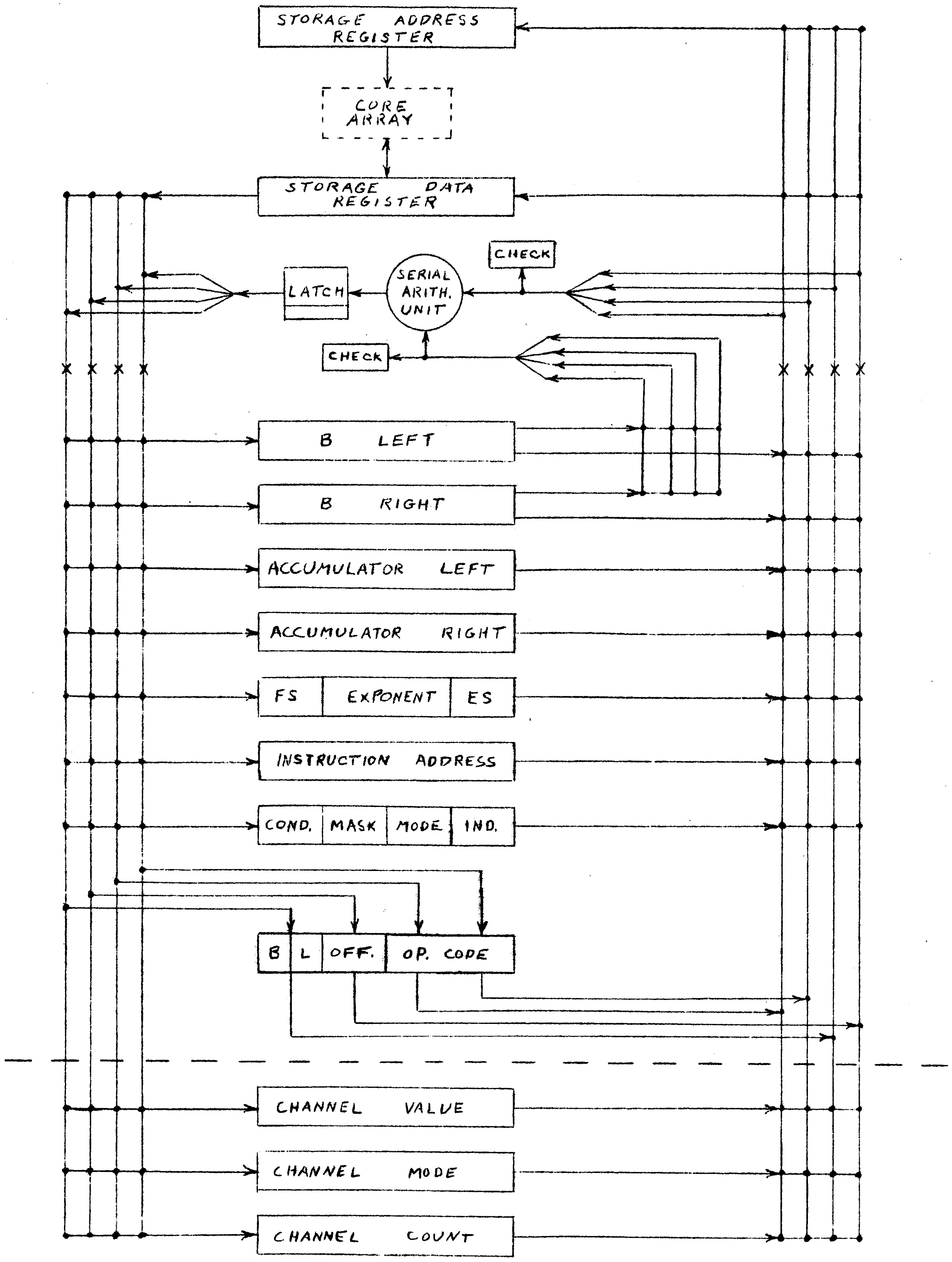


Figure 1 - 8104 Data Paths

Chapter 2

Operand Designation

2.1 Formats

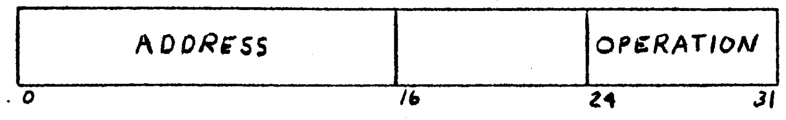
The computer stores information in core storage in 16-bit locations. Information transmission between storage and the computer is in parallel, 16 bits at a time. Each 16-bit word is part of an 18-bit machine word; the extra 2 bits are check bits, not available to the programmer, which permit single error detection. Within the central processing unit, each check bit is used as the odd-parity bit of 8 consecutive information bits.

Locations in storage are specified by a contiguous set of addresses, which are numbered from 0 to 65,535 (2^{16}-1). Each location is subdivided into 4 groups, called bytes, of 4 bits each. The bytes of each location are numbered, from left to right, 0 to 3. Byte 3 of a location may be considered adjacent to byte 0 of the next higher addressed location. The four bits of each byte are numbered, from left to right, 0 to 3. Bits may also be numbered with respect to a full storage location, from left to right, 0 to 15.

Some information words may occupy more than one consecutive storage location. The addresses of such words are the addresses of their leftmost storage locations.

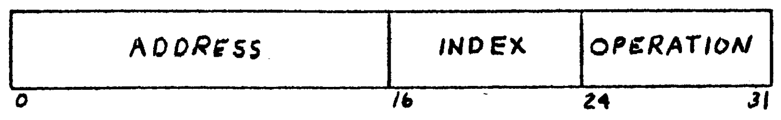
2.1.1 Instruction Formats

Every instruction word occupies two consecutive storage locations. The left location specifies the address of the instruction, and bytes 2 and 3 of the right location specify the operation. Bytes 0 and 1 of the right location may be used to specify various other functions of the operation.



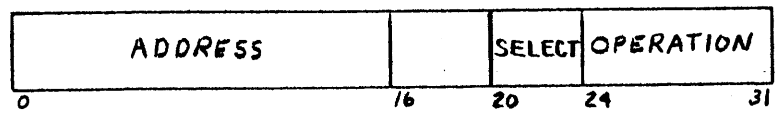
2.1.1.1 Indexable Instructions

The accumulator instructions, the control word arithmetic instructions, and the floating point instructions are indexable. Other instructions are not indexable. The index register used is specified by bits 16-23 of the instruction word.



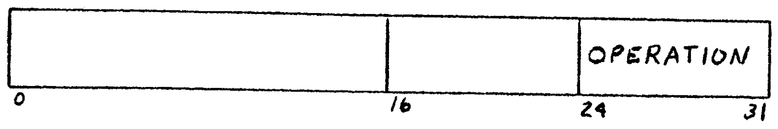
2.1.1.2 Decision Instructions

The branching and program-control-word-storing operations are decision instructions. These operations are conditional upon the value of selected bits in the condition register. Bits 20-23 of the instruction word select the bits of the condition register to be tested. Bits 16-19 are not used.



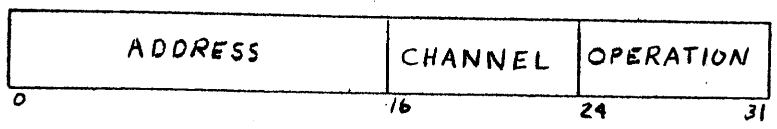
2.1.1.3 Interruption Instructions

INTERRUPT INTENTIONALLY, ENABLE, and ENABLE AND WAIT are interruption instructions. Only the operation code of these instructions is used. Bits 0-23 of the instruction word are not used.



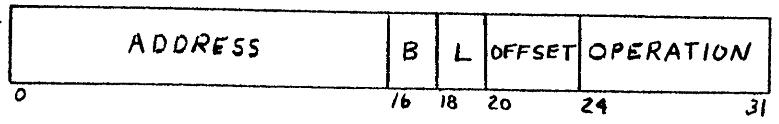
2.1.1.4 Input-Output Instructions

The I-O channel is specified by bits 16-23 of the instruction word.



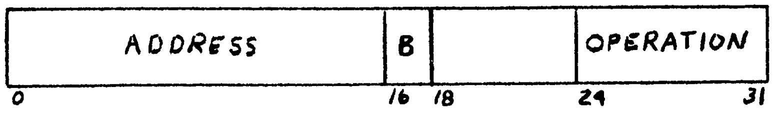
2.1.1.5 Variable Field Length Instructions

Variable field length, or VFL, instructions all specify a location, a byte, a limit, and an offset. Bits 16 and 17 of the instruction word specify a byte in the addressed storage location; this specified byte is the leftmost byte of the field. Bits 18 and 19 specify the position of the rightmost byte in the field. Bits 20-23 of the instruction word specify the offset.



2.1.1.6 Byte Address Instructions

The instructions LOAD EXPONENT SIGN, LOAD FRACTION SIGN, and LOAD CONDITION REGISTER all use a two bit field, bits 16 and 17 of the instruction word, to indicate a byte within the addressed location, byte 0, 1, 2, or 3. This byte is the operand.



2.1.2 Data Format

Data formats may either be a single byte, a field o~~r~~f one to four consecutive bytes, or a word of one, two, or three consecutive storage locations.

2.1.2.1 Single Byte Format

The conditional instructions and the instructions LOAD EXPONENT SIGN, LOAD FRACTION SIGN, and LOAD CONDITION REGISTER use the single byte operand format. Bits 16 and 17 of the instruction word are used to specify the byte within the addressed location.

2.1.2.2 Variable Field Format

The variable field (VFL) format and details are explained in Chapter 4.

2.1.2.3 Single Location Formats

The operations REFILL, INCREMENT BY ONE, INCREMENT BY TWO, INCREMENT BY THREE, DECREMENT, STORE LEFT HALF-FRACTION, LOAD LEFT HALF-FRACTION, STORE RIGHT HALF-FRACTION, and LOAD RIGHT HALF-FRACTION use an operand of a single full location.

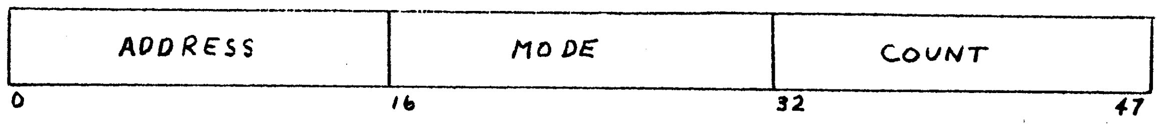
2.1.2.4 Double Location Formats

The operations STORE FRACTION and LOAD FRACTION use an operand of two consecutive locations.

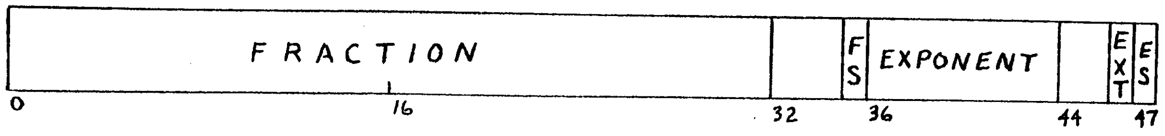
2.1.2.5 Triple Location Formats

STORE ACCUMULATOR, LOAD ACCUMULATOR, the floating point operations, and the input-output operations all use an operand of three full storage locations. The accumulator itself occupies storage locations 256, 257, and 258.

The input-output operations use a control word format with the address in the left location, the mode in the second location, and the count in the right location.



In the floating point data format, the exponent, fraction sign, exponent sign, and extremum bit occupy the rightmost location. Bit 35 specifies the fraction sign, bits 36-43 specify the exponent, bit 46 is the extremum bit, and bit 47 specifies the exponent sign. The fraction occupies the two left full locations.



2.2 Indexing

255 index registers are available, each occupying a full storage location. The addresses of these locations are 1 through 255. Indexing may be specified by any indexable instruction by using the address of an index register in bits 16-23 of the right location of the instruction; if address 0 is used no indexing will result.

When indexing is specified, prior to the execution of an instruction, the address portion of the instruction is fetched and summed, unsigned, with the contents of the index register. The low order 16 bits of the sum form the effective address of the instruction. The address portion of the instruction, however, remains unchanged in core storage.

2.3 Indirect Addressing

The address part of an instruction is normally used to address the data upon which the instruction operation is performed. This addressing mode is called direct addressing. In another mode of addressing, bits 0-15 of the address can be used to address a new word occupying two storage locations. This word in turn contains an address part which 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 new word 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.

Bit 24 of an instruction indicates whether 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 may be specified for each instruction independently and applies only to the address which is part of that instruction. Indirect addressing is also independent of the operation to be performed. Conversly the operation, as specified in bits 24-31 of an instruction, is not altered by indirect addressing.

Each time a word is fetched in indirect addressing, bit 24 is inspected. When bit 24 is one, indirect addressing continues through another level, using bits 0-15 to address the next control word. In each level the address is modified to be coherent with the operation. That is, if the initial instruction is indexable, bits 16-23 of the control word of each level will be used for indexing at that level prior to further indirect addressing. If a byte address or byte limit is specified in the initial instruction, a byte address and byte limit must be specified in the last level control word. Channel number, condition selection, and offset are executed as specified in the initial instruction word. When bit 24 of the control word is zero, a direct address is indicated at that level.

In indirect addressing, it is possible that the addresses refer to each other in such a way that they form a loop. As a result the computer cannot finish its operation and no interruption is accepted. This situation would be the result of a programmer's error. In order to prevent the machine from being locked up in this way, indirect addressing will be terminated if still active after one millisecond and interruption code USA, Unending Sequence of Addresses, will be stored in the interruption stream.

2.4 Accumulator Operations

The accumulator operations are indexable. The condition register is not altered by these operations, no indicators are turned on.

2.4.1 STORE ACCUMULATOR

The contents of the accumulator are stored into three consecutive locations beginning with the addressed location. The addressed word may not overlap an accumulator location.

2.4.2 LOAD ACCUMULATOR

The contents of three consecutive locations beginning with the addressed location are loaded into the accumulator.

2.4.3 STORE FRACTION

The contents of the left two locations of the accumulator are stored into two consecutive locations in storage beginning with the addressed location. The addressed word in storage may not overlap an accumulator location.

2.4.4 LOAD FRACTION

The contents of two consecutive locations in storage beginning with the addressed location are loaded into the left two accumulator locations. The exponent portion of the accumulator remains unchanged.

2.4.5 STORE LEFT HALF-FRACTION

The contents of the leftmost accumulator location is stored into the addressed location. An accumulator location may not be addressed.

2.4.6 LOAD LEFT HALF-FRACTION

The contents of the addressed location are loaded into the leftmost accumulator location. The remainder of the accumulator remains unchanged.

2.4.7 STORE RIGHT HALF-FRACTION

The contents of the second accumulator location are stored into the addressed location. An accumulator location may not be addressed.

2.4.8 LOAD RIGHT HALF-FRACTION

The contents of the addressed location are loaded into the second accumulator location. The remainder of the accumulator remains unchanged.

2.5 Control Word Operations

There are five control word operations, all indexable. For each operation, the condition register is set to reflect the result, no indicators are affected.

2.5.1 INCREMENT BY ONE, INCREMENT BY TWO, and INCREMENT BY THREE.

The contents of the location specified are incremented by one, two, or three, respectively.

2.5.2 DECREMENT

The contents of the location specified are decremented by one.

2.5.3 REFILL

The contents of the location at the effective address of the instruction are interpreted as a refill address. The contents of the location at the refill address replace the contents of the location at the effective address of the instruction.

2.6 Address Monitoring

Information which is in core storage can be classified as data, instructions and reference information. Information may be used for instruction definition, communication with external units or with an operator, supervision and scheduling of programs, monitoring of program execution, or problem program execution itself. It is desirable that errors in a program will not cause any changes in data, instructions or reference information belonging to the computer system as a whole. This protection is provided by defining a monitored area and preventing data stores to all locations within the monitored area.

2.6.1 Definition of Monitored Area

The execution of a program is controlled by a program control word. Bit 25 of the program control word is used to define the monitored area. When bit 25 is zero locations 256 through 1023 are monitored for data store operations. When bit 25 is one the monitored area is extended such that locations 256 through 2047 are monitored for data store operations.

2.6.2 Action of Address Monitoring

The address monitoring system is active only when the computer is in the enabled mode. When the computer is in the enabled mode and an instruction specifies the storing of data at a protected address, the store operation is not executed and the protected storage location, therefore, 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 DS, Data Store. Address monitoring is only active for store-type operations. These are operations in which the contents of an addressed storage location are subject to change.

Programming Note:

In general when a protected address is encountered during an operation the operation is terminated. This results in complete suppression of the operation.

2.6.3 Addresses Monitored

All addresses used in store-type operations which are specified in the course of an operation are subject to address monitoring. Although an address which is associated with an operation is not actually used, it will nevertheless, be monitored. For example, the store program control word instruction is monitored whether the instruction test is successful or not. Core storage addresses which are used by input-output operations are not subject to address monitoring.

Address monitoring includes the last effective addresses of all store-type operations including the refill operations. Not included in address monitoring are:

1) Channel addresses of input-output instructions as well as all data addresses of input-output operations.

2) Auxiliary addresses generated as part of instruction execution.

3) Addresses used in storing an interruption code.

2.7 Storage Assignment

The effective address of each instruction provides for a 16 bit address. This permits addressing of 65,536 (2^{16}) storage locations. Several of these locations are used for specific purposes. The remaining addresses are available as general purpose core storage. The functions of the special locations are described in the following sections.

2.7.1 Control Words

There are three types of control words: program control words, interruption control words, and channel control words. The formats and details of the program control words (PCW) and the interruption control words (ICW) are found in Chapter 3 and of the channel control words (CCW) are found in Chapter 6. The storage assignments of all control words are listed in 2.7.6.

2.7.2 Accumulator

The accumulator occupies locations 256, 257, and 258. It may be addressed only for fetching information, not for storing. Attempts to store in these locations result in the storage of an Invalid Address interruption code.

Floating point words occupy all three accumulator locations in the same format as floating point data in general storage.

VFL words occupy consecutive bytes of the accumulator. The offset is the number of bytes to the right of the VFL word and to the left of bit 32 of the accumulator. The sign of a VFL word in the accumulator is kept in bit 35.

2.7.3 Time Clock

The time clock is optionally provided to measure time difference or duration over relatively long periods. This clock consists of location 264 and is stepped by pulses originating from a one cycle per second oscillator. The time clock is updated at the next time that the interval timer is updated. The clock measures time in seconds. A full cycle is about 18 hours. Each time the oscillator delivers a pulse the contents are read out, incremented by 1 and returned to core storage. The clock runs continually while the computer is under program control including the time the computer 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 0.

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 at a later time can be obtained by reading the time clock setting and converting the amount to hours, minutes and seconds taking into account the 18 hour recycle time.

2.7.4 Interval Timer

The interval timer is an optional feature intended to measure elapsed time over relatively short intervals. The timer consists of a number 16 bits long in location 266; which is stepped down by pulses originating from a stable oscillator. It can be set to any value at any time and interruption code TS, time signal, is stored when the time period has ended. Each time the oscillator delivers a pulse the contents are read out, decremented by 1 and returned to core storage. The oscillator operates at 1024 (2^{10}) cycles per second, or a pulse about every millisecond. Bits 0-6 show time in seconds. A full cycle is about 1 minute.

The timer runs whenever the machine is operating and includes the time the computer is waiting. Whenever it goes from one to zero it stores the interruption code TS in storage. The timer stops upon reaching zero.

2.7.5 Main Core Storage

All special purpose memory locations operate as general purpose memory. Addresses from 0 to 65,535 ($2^{16}-1$) are available for addressing main core storage. Not all of these 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 this effective address is above the limit of available memory, interruption code AD, Address Invalid, will be stored. All locations in main memory are valid word addresses in any operation, except a store may not be executed to a location of the accumulator.

2.7.6 Storage Assignment List

Location	Assignment	
0	None	
1-255	Index Registers 1-255	
256-258	Accumulator	
259	None	
260-261	Intentional Interruption PCW	
262-263	Disabled PCW	
264	Time Clock	
265	Interval Timer	
266-267	None	
268-269	Enabled PCW	
270-271	Time Signal PCW	
272-274	Store ICW	
275	None	
276-277	Channel Not Operational PCW	
278-279	Channel Busy PCW	
280-282	Fetch ICW	
283	None	
284-285	Unended Sequence of Addresses PCW	
286-287	Address Invalid PCW	
288-~~289~~ 290	Initial Value ICW	
291	None	
292-293	Operation Code Invalid PCW	
294-295	Data Store PCW	
296-317	None	
318-319	Maskable Indicator PCW	
(8n) - (8n + 2)	Channel n CCW	$40 \leq n \leq 127$
8n + 3	None	
(8n + 4) - (8n + 5)	Channel n End PCW	
(8n + 6) - (8n + 7)	Channel n Special Condition PCW	

Chapter 3

Instruction Sequencing

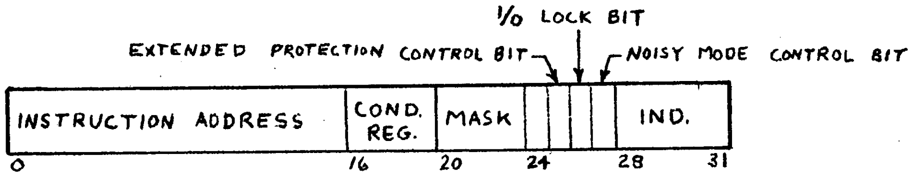
3.1 Normal Sequential Operation

Normally, instructions are taken in sequential order from successive locations in storage. These locations are specified by the instruction address portion of a program control word. This address is incremented each time an instruction is fetched.

3.1.1 Program Control Words

Each program or subprogram executed by the computer normally has its own program control word. Separate locations in core storage are provided for the program control words for the program being executed in the enabled mode, which is normally the basic program being processed by the computer, and for each of the programs that are executed as a result of program interruption. Switching of control of the computer 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.



3.2 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. The condition that is tested is the state of specified bits in the condition register.

3.2.1 Condition Register

The condition register occupies bits 16-19 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 conditions recorded in the condition register are listed below.

Control Word Instructions

0	(Not used)
1	Result address zero
2	Result address greater than zero
3	(Not used)

Floating Point Arithmetic Instructions

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

Variable Field-Length Arithmetic Instructions

0	Result less than zero
1	Result zero
2	Result greater than zero
3	(Not used)

Connective Instructions

0	(Not used)
1	Result zero
2	Result greater than zero
3	(Not used)

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

3.2.1.1 LOAD CONDITION REGISTER

The byte address part of this instruction, bits 0-17, specifies a single 4-bit byte in storage. This byte is placed in the condition register. Bits 18-23 of the instruction are not used.

3.2.2 Branching Operations

Bits 20-23 of the branching instructions 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. Bits 16-19 of the instruction are not used.

3.2.2.1 BRANCH IF ANY

If any of the selected bits of the condition register are on, the contents of the instruction address part of the current program control word are replaced by bits 0-15 of the effective address of this instruction. If none of the selected bits are on, control proceeds normally to the next instruction in sequence.

3.2.2.2 BRANCH IF NONE

This instruction is similar to BRANCH IF ANY, except that the branch is successful if none of the selected bits of the condition register are on, and unsuccessful if any are on.

Programming Note:

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

3.2.3 Storage of Program Control Word

The current program control word can be stored for a later return of control to the current point in the program. The instruction address stored is four greater than the address of the current instruction. This allows a branching instruction to be associated with the program-control-word-storing instruction. The address stored is thus the proper one for a return of control to the instruction following the branching instruction.

The program-control-word-storing instructions are conditional in the same way as the branching instructions. Selected bits of the condition register are tested, and if the test is satisfied, the program control word is stored. Otherwise, it is not stored.

3.2.3.1 STORE PCW IF ANY

If any of the selected bits of the condition register are on, the current program control word is stored in the word at the effective address of the instruction and the following word. If none of the selected bits are on, the program control word is not stored.

3.2.3.2 STORE PCW IF NONE

This instruction is similar to STORE PCW IF ANY, except that the current program control word is stored if none of the selected bits of the condition register are on, and is not stored if any are on.

3.3 Program Interruption

The program interruption system provides the means by which the computer 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 8-bit byte coded to identify the cause for interruption, to be stored in an area in storage called the interruption stream. At the same time, the interruption trigger, a control trigger in the circuits of the computer, is turned on if the computer is in the enabled mode. This trigger is tested at the completion of each instruction. If it is on, an interruption occurs. An interruption consists of a switch to the disabled mode; an 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.

3.3.1 Storage of Interruption Codes

Interruption codes are stored in an area of storage called the interruption stream. The location and extent of the interruption stream is defined by the initial value of the interruption control word. This initial value is stored in words 288 through 290 of core storage and has a format similar to that of an input-output control word.

Upon the detection of an interrupting condition, an interruption code is stored. This may occur at any time during the execution of an instruction regardless of whether the computer is in the enabled or disabled mode. Each code is stored at the address currently contained in bits 0-16 of the store interruption control word. This control word is stored in words 272-274 of core storage and has the same format as its initial value stored in words 288-290. Each time a byte is stored, unless the byte had a value of 65 or 66, the address of the store interruption control word is incremented by one. Each time this address crosses a word boundary, the count in bits 32-47 of the control word is decremented by one. When this count reaches zero, the control word is reset to its initial value.

3.3.2 Interruption Action

The interruption trigger is tested at the completion of each instruction. If it is on, an interruption results. The current program control word is stored in locations 268-269 if the computer is in the enabled mode, or locations 262-263 if it is in the disabled mode. The fetch interruption control word is obtained from locations 280-282, and the interruption trigger is turned off. This control word has the same format and initial value as the store interruption control word. The byte addressed by this control word is fetched from storage. If this byte has a value of neither 65 nor 66, the address of the control word is incremented, and if necessary reset, just as in the storage of interruption codes, and the computer is placed in the disabled mode. If this byte is 65 or 66, the control word is not incremented, and the computer is placed in the enabled mode. In either case, a new program control word is obtained from storage locations determined by the value of the byte fetched. If the value of this byte is x, then the program control word is obtained from locations $4x+2$ and $4x+3$ if x is odd, or locations $4x+4$ and $4x+5$ if x is even. Program execution then proceeds using this new program control word. If the byte has a value of 65, the computer enters a waiting status during which it executes no instructions. This status is terminated by the occurrence of an interrupting condition.

3.3.3 Program Switching Instructions

An intentional interruption to effect a switch to a different program control word can be initiated by a program switching instruction. There are three such instructions. They differ only in the value of the interruption codes that they cause to be stored.

3.3.3.1 INTERRUPT INTENTIONALLY

This instruction causes an interruption code with a value of 64 to be stored in the interruption stream. It also turns on the interruption trigger. This instruction is used to switch from the enabled to the disabled mode. The address portion of the instruction is not used.

3.3.3.2 ENABLE

This instruction is identical to INTERRUPT INTENTIONALLY, except that an interruption code with a value of 66 is stored. This instruction is used to switch from the disabled to the enabled mode.

3.3.3.3 ENABLE AND WAIT

This instruction is identical to INTERRUPT INTENTIONALLY, except that an interruption code with a value of 65 is stored. This instruction is used to cause the computer to wait for an external signal before proceeding. This instruction is valid only if the computer is in the disabled mode. If executed in the enabled mode, it causes an Invalid Operation Code interruption byte to be stored.

3.3.4 Indicators

The indicators, located in bits 28-31 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 20-23 of the program control word, is on; and the computer is in the enabled mode; interruption code 79, Maskable Indicator, is stored in the interruption stream. Indicators can be turned on only by arithmetic operations. Indicators are not turned off automatically, but remain on until turned off by programming. The indicators are listed below.

28	Generated Extremum Positive
29	Generated Extremum Negative
30	Oversized Result
31	Zero Divisor

3.3.5 Interruption Codes

The interruption codes are listed below. Each code occupies an 8-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 first sixteen codes, from 64 to 79, are used for computer interruptions. The remaining codes, from 80 to 255, are used for input-output channel interruptions. Each channel requires two codes, therefore a maximum of 88 channels can be accomodated.

Computer Interruptions

64	Intentional Interruption
65	Programmed Wait
66	Enabled Program
67	Time Signal
68	Channel Not Operational
69	Channel Busy
70	Unended Sequence of Addresses
71	Address Invalid
72	Operation Code Invalid
73	Data Store
74	
75	
76	Not assigned.
77	
78	
79	Maskable Indicator

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 40 to a maximum of 127.

2n	Channel n End
2n + 1	Channel n Special Condition

3.4 Manual Control

A limited number of control keys and lights are located on the operator's console. These allow manual control of the computer by the operator for initial loading of the program and for maintenance purposes. The lights indicate the state of the machine and whether it is operating properly.

3.4.1 System States

There are three system states: running, initial, and waiting. The system can be placed in the initial state at any time by depressing the INITIALIZE key. When the system is in the initial state, it can be placed in the running state by depressing the START key. The waiting state is entered by executing an ENABLE AND WAIT instruction, and the system is returned to the running state by the first alert encountered.

The RUNNING light, described later, indicates when the system is in the running state.

3.4.2 System Initialization

The INITIALIZE key may be used for program loading or for manual termination of computer operation. Depressing this key stops the execution of instructions by the system. Since operation is terminated before the completion of the current instruction, the original program cannot normally be resumed. No further instructions are executed until the START key is depressed.

The INITIALIZE and START keys are located on the maintenance panel which is part of the operator console.

3.4.3 Initial Power On

When the power is turned on, the computer is in the initial state. All addressable locations may contain random bits not necessarily with correct parity. No instructions are executed until the START key is depressed.

3.4.4 System Status Lights

Two lights are provided to show the status of the system.

1. RUNNING light. This light is turned on when the power is on and the system is not in the waiting or initial state.

2. INACTIVE light. This light is turned on when the power is on but the instruction counter is not being incremented.

Thus, the four possible states of these lights are interpreted as follows:

RUNNING	INACTIVE	
Off	Off	Power off.
On	Off	Running normally.
Off	On	Waiting or initial state.
On	On	Trouble: The system is hung up by repeated machine checks, interruptions, or branch instructions, or by a lost control signal.

The status lights are located on the maintenance panel.

Chapter 4

Variable Field-Length Data Handling

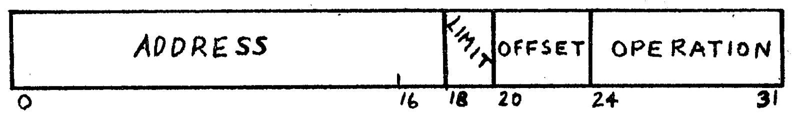
4.1 General Description

The variable field-length operations operate on a variable-length data field, which may start at any byte position in storage. The instruction contains an 18-bit word and byte address that defines the leftmost byte of the storage field. The instruction also specifies the length of the field, which may be from one to four bytes, by specifying the position of the rightmost, or limit, byte within the word.

The second operand in these operations is a field in the fraction portion of the accumulator. Since this field may be located anywhere in the fraction portion of the accumulator, the instruction format also contains an offset field that defines the low-order byte of the accumulator operand. The low-order positions of the two operands are aligned for processing. The results of the variable field length operations replace the accumulator operand.

4.2 Instruction Format

The variable field-length operations have the instruction format shown below.



Bits 0-17 specify the byte address of the leftmost byte of the storage operand. Bits 18 and 19 specify the rightmost, or limit, byte of the storage operand. This byte may be in either the same word or the next word following the one that contains the leftmost byte, subject to the restriction that the storage field is no greater than four bytes in length. Bit 24 of the instruction specifies direct or indirect addressing. If this bit is zero, the byte address and limit are used directly. If this bit is one, bits 0-15 of the instruction are used to obtain an indirect address. This address replaces both the byte address and limit, but not the offset, specified in the instruction.

Bits 20-23 of the instruction specify the offset of the accumulator operand. The first of these bits is ignored. The remaining three specify the number of bytes between the rightmost byte of the accumulator operand and the right-hand end of the fraction portion of the accumulator.

4.3 Arithmetic Operations

Two variable field-length arithmetic operations are provided: ADD and RESET ADD. Both leave the result in the accumulator. Storage of results is accomplished with the accumulator-storing operations described in Chapter 2.

4.3.1 Sign Handling

The variable field-length operations address unsigned fields in storage. If signed arithmetic is desired, the sign of the factor in storage is contained in the exponent sign byte of the accumulator. This sign can be examined by the variable field-length arithmetic operations to provide full algebraic sign control.

4.3.1.1 LOAD FRACTION SIGN

This instruction addresses a single byte in storage. The limit and offset fields of the instruction are ignored. The byte is placed in the fraction sign byte of the accumulator. The remainder of the accumulator is left undisturbed. The condition register is not altered.

Modifier:

Negative Sign. If this modifier bit is on, the sign bit (rightmost bit) of the byte is inverted before placing in the accumulator.

4.3.1.2 LOAD EXPONENT SIGN

This instruction is identical to LOAD FRACTION SIGN, except that the byte is loaded into the exponent sign byte of the accumulator, instead of the fraction sign byte. The same modifier applies.

4.3.2 ADD

The instruction addresses a field in storage. This field is added algebraically to the contents of the accumulator at the specified offset. The result replaces the contents of the accumulator to the left of the offset. If the sign of the accumulator is changed, the entire accumulator is recomplemented. The sign of the accumulator operand is contained in the fraction sign byte of the accumulator. The sign of the storage operand is contained in the exponent sign byte of the accumulator. If significant bits of the storage operand overlap the left end of the accumulator, due to an offset greater than 4, the Oversized Result indicator is turned on. This indicator is also turned on if a carry is propagated beyond the left end of the accumulator. The condition register is set according to the resulting contents in the entire accumulator.

Modifiers:

Unsigned. If this modifier bit is on, the sign bit of the storage operand is ignored, and taken as positive.

Negative. If this modifier bit is on, the sign bit of the storage operand is inverted before use. This modifier acts after the unsigned modifier.

4.3.3 RESET ADD

This operation is identical to ADD, except that first the entire accumulator, including fraction sign, exponent, and exponent sign, is cleared to zeros. Consequently, the unsigned modifier has no effect on the result.

4.4 Connective Operations

Each of the connective operations can specify any one of the sixteen possible binary connectives. Two operands, one in storage and one in the accumulator, are combined bit for bit, using the connective specified, to produce a result field. This result field may replace the operand field in the accumulator or may be tested and then discarded.

4.4.1 Logical Connectives

The particular connective to be used is specified by bits 28-31 of the instruction. The code in these four bits is composed of the result bits obtained for each of the possible combinations of m and a, where m is a bit from the storage operand, and a is the corresponding bit from the accumulator operand. In this code, the first bit represents the result when m and a are both zero. The second bit represents the result when m is zero and a is one. The third bit represents the result when m is one and a is zero. The fourth bit represents the result when m and a are both one.

4.4.2 CONNECT

The addressed storage operand is combined with the accumulator field specified by the offset in accordance with the logical connective specified. The result field replaces the accumulator operand. The remainder of the accumulator remains unchanged. The condition register is set according to the value of the result field placed in the accumulator. The contents of the remaining portions of the accumulator do not affect the setting of the condition register. The Oversized Result indicator cannot be turned on by a CONNECT operation.

4.4.3 CONNECT FOR TEST

This operation is identical to CONNECT, except that the result field is not placed in the accumulator, but is used only to set the condition register and is then discarded. The accumulator remains unchanged.

Programming Note:

Variable length fields are stored by combining the field to be stored with the remaining portions of the word or words into which it is to be stored. This combination is done in the accumulator, using connective and accumulator loading operations. The full words are then stored using accumulator storing operations. The sign of a result in the accumulator may be combined with the field to be stored by using a CONNECT operation that addresses the first byte of location 258 (the fraction sign of the accumulator), thus moving the sign to the fraction portion of the accumulator.

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 $\pm F\ 16^{\pm E}$. The exponent is an eight bit binary number. The fraction bits are grouped together in four bit bytes to form an eight digit fraction with the hexadecimal point to the left of the high order digit.

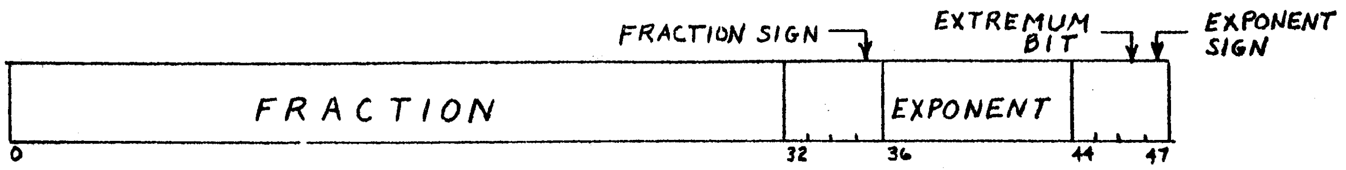
There are four basic floating-point instructions, ADD, RESET ADD, MULTIPLY, and DIVIDE, supplemented by LOAD and STORE instructions that are used to transmit data between the accumulator and storage locations. By means of modifiers, the basic instructions may be manipulated to furnish a versatile floating-point performance. Provision is made for the direct or indirect addressing of data in conjunction with indexing, modification of the sign of the addressed operand, and the choice of normalized or unnormalized operation.

The fractions of all arithmetic results contain eight hexadecimal digits. Not all of these digits need be significant; furthermore, as a result of calculation, the number of significant digits may be reduced. In order 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.

5.1.1 Data Format

Floating point numbers are represented in the following 48-bit format.



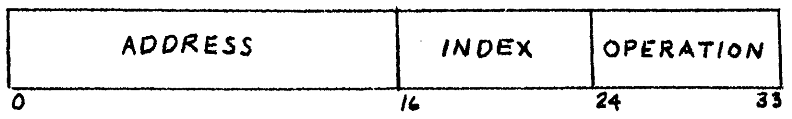
The fraction occupies the first 32 bits of the data field. Bits 32-35 contain the fraction sign byte, of which only bit 35 is used to indicate the fraction sign. The fraction is positive or negative according as bit 35 is zero or one. Bits 32-34 are ignored by the machine. The eight bits of the exponent are located in bit positions 36-43. Floating point numbers whose magnitudes range between 16^{255} and 16^{-256}, or approximately 10^{307} and 10^{-308} may be represented to a precision of eight hexadecimal digits.

The exponent sign byte occupies bits 44-47. Bits 44 and 45 are unassigned and ignored by the machine. The extremum bit, indicative of an exponent outside the normal range, is assigned to bit 46. The exponent sign is represented by bit 47 in the same manner as bit 35 represents the fraction sign.

A floating point number is stored in three consecutive 16-bit words. The non-usage of bits 32-35 and 46-47 is motivated partly by the desire to provide a measure of compatibility of data format with that of larger machines, and partly by the need to treat fractions and exponents as fixed point numbers. For the latter reason, the exponent and the fraction signs cannot be placed together in a single byte. The unassigned bits correspond to facilities, such as the provision of data flags, that are available only in the larger machines.

5.1.2 Instruction Format

All floating point instructions are of single address, 32-bit form. This single address, contained in bits 0-15 of the instruction field, specifies one operand of the floating-point operation to be formed. The second operand is always in the accumulator. Bits 16-23 of the instruction 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. 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 instruction code contain the fixed code 11 to indicate that the instruction belongs to the floating-point class. Bit 24 is zero or one according 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 data is utilized or ignored according as bit 26 is zero or one. In either event, the effective sign of the operand is determined in conjunction with bit 29, the sign modifier bit, as described in (5.1.3) . Finally, bits 30 and 31 specify one of the four basic floating-point instructions.

5.1.3 Sign Control

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

Bit 26	Bit 29	
0	0	retain the sign (bit 35) 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 operand is preserved or inverted according as bit 29 is zero or one. Whenever bit 26 is one, bit 29 of the instruction replaces bit 35 of the operand as its sign.

In no case is the sign of the operand 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 non-zero, high-order fraction digit. If one or more high-order fraction digits are zero, the number is not in normalized form. The process of normalization consists of shifting the fraction until the high-order digit is non-zero, 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 according 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 is 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, the 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 $1 > F \geqslant 1/16$. The magnitude of an unnormalized fraction lies in the range $1 > F \geqslant 0$. 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 The Treatment of Numbers Outside the Normal Exponent Range

A number outside the normal exponent range is characterized by its extremum bit (bit 46) 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 if they were zero. All numbers with an extremum bit of one and a positive exponent sign are treated by the machine as infinite. Such numbers are referred to as True Zero (TZ) and Quasi Infinity (QI), 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.

Since the floating-point representation is quasi-logarithmic in nature, the quantity zero is not strictly representable. By suitable definition of machine operations, any number in the range ($\pm F\ 16^{-255}$) is given the arithmetic properties of zero. Similarly, any number greater than $F\ 16^{256}$ is constrained by the machine specifications to have the arithmetical properties of infinity.

The following table defines the results 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	QI	A	TZ
QI	Acc*	Sto	Sto
S	Acc	A$\pm$S	Sto
TZ	Acc	Acc	Acc

MULTIPLY

Storage \ Acc	QI	A	TZ
QI	Acc	Sto	Sto*
S	Acc	A•S	Acc
TZ	Acc*	Sto	Acc

DIVIDE

Storage \ Acc	QI	A	TZ
QI	Acc*	TZ	Acc
S	Acc	A/S	Acc
TZ	Acc	QI	QI

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 $0/0$).

Programming Note:

In the above cases, if either of the operands and the result has an extremum condition, the extremum is propagated and no indication is made that such propagation occurred, although, of course, the corresponding bit in the condition register (5. 1.7~~6~~) will be turned 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 occasion 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 true zero. The underflow condition is indicated by the Generated Extremum Negative (GEN) indicator.

5.1.6 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 according as this bit is zero or one. Noisy mode influences normalized operations only; unnormalized operations are unaffected.

5.1.7 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 (zero result)	1
In the normal exponent range and greater than zero	2
Extremum Positive (infinite result)	3

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

5.1.8 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.

Generated Extremum Positive (GEP) (Exponent Overflow) This indicator 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) This indicator 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 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) This indicator 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 indications are mutually exclusive.

5.2 Operations

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

5.2.1 ADD

The addition of two floating point numbers consists of an exponent comparison and a fraction addition. The exponent of the addressed operand is subtracted from the operand in the accumulator. The fraction of the number with the algebraically smaller exponent is shifted right a number of digit positions equal to the exponent difference, truncated and added to the fraction of the number with the greater exponent. If the exponent difference is greater than or equal to eight, no addition of fractions takes place, and the number with the greater exponent is treated as the sum.

The fraction sign of the addressed operand is modified by the sign modifiers (5. 1. 3) prior to the addition. 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 (5.1.5). 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 (5. 1. 5). 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 non-zero storage operand, or infinite storage operand and non-infinite 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.

5.2.2 RESET ADD

The RESET ADD instruction is equivalent to an ADD instruction for which the operand in the accumulator is always in the extremum negative, or True Zero, condition. If the addressed operand has a positive extremum, it is simply loaded into the accumulator without any modifications; if the operand is True Zero, the True 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, that may influence it. The price of this versatility is a longer instruction execution time.

The only indicator that can be affected by RESET ADD is the Generated Extremum Negative (GEN) indicator, and then only in the exceptional case that the addressed operand is a new data word with zero fraction and normal exponent range.

Programming Note:

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

5.2.3 MULTIPLY

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. The two 8-digit fractions of the operands are multiplied to form an 8-digit product. 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. 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 post-normalized 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 is outside the normal exponent range, are summarized in (5. 1. 5). If the accumulator operand is infinity, or it is zero and the operand from storage is non-infinite, the contents of the accumulator are accepted as the product. For the remaining situations, infinite storage operand and non-infinite accumulator operand, or zero storage operand and normal range accumulator operand, the operand from storage is brought into the accumulator.

The MULTIPLY operation affects the generated extremum indicators GEP and GEN. They are turned on for exponent overflow and underflow, respectively.

5.2.4 DIVIDE

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 a normalized DIVIDE, both dividend and quotient are pre-normalized and their exponents adjusted accordingly. If, after pre-normalization, the dividend fraction is greater than or equal to the divisor fraction, the quotient must be normalized by an effective right shift and its exponent increased by one. If the divisor fraction is the greater, no such post-normalization adjustment is necessary to obtain the quotient in final normalized form.

In the noisy mode, additional dividend bytes required as the division progresses are supplied noisily.

For an unnormalized DIVIDE operation, the divisor is pre-normalized and the dividend is shifted in synchronism with the divisor. If the resulting dividend fraction is greater than or equal to the divisor fraction, the Oversized Result (OR) indicator is turned on. If the divisor fraction is the greater, the division proceeds. The quotient thus obtained is not post-normalized.

The quotient exponent is the difference of the dividend and divisor exponents, augmented by one in the normalized case when the dividend fraction is not less than the divisor fraction. If the quotient exponent overflows or underflows, the GEP or GEN indicator, respectively, will be turned on. 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 exceptional cases, when one or both the operands are outside the normal exponent range, are summarized in (5. 1. 5). In no case is the accumulator fraction disturbed. If the dividend is infinite, or the division 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. A zero dividend and non-zero 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.

Chapter 6

Input-Output Operations

Data is recorded on and read from external media by input-output units. The 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 central processing unit contains the facilities necessary to control the flow of information between adapters and core storage. Some of these facilities are also used for other functions of the computer.

The design of an adapter depends on the units which it controls. The interface between adapter and computer, however, is the same for all types of adapters; no modification of the computer is necessary for the connection of any type of adapter. A common set of instructions is used by the computer to control all types of adapters.

All adapters operate serially, sending or receiving one 8-bit byte at a time. Two bytes are required to transmit a storage word of 16 bits. A parity bit is provided with each byte transmitted between adapter and computer to check the transmission.

When an input-output instruction is given, the program is delayed a few microseconds while the computer determines whether the operation can be executed. If the input-output operation cannot be executed, an alert is given before the program is resumed. Once an operation has been initiated, no other instructions are needed for its completion, including the assembly or disassembly of information and the housekeeping of addresses. The program is alerted when the operation is completed. 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 computer program is increased execution time.

6.1 Capacity

6.1.1 Number of Channels

The facilities required in the central processing unit for the connection of an adapter are called a channel. The central processing unit provides one channel, the multiplex channel, which can be connected to a single adapter, or it can, without modification, accommodate multiplexing facilities. These facilities can provide up to 86 channels for the attachment and simultaneous operation of adapters. Two optional channels, called simplex channels, suitable only for the connection of a single adapter, can be provided in the central processing unit.

The simplex channels are assigned addresses 40 and 41. If a single adapter is connected to the multiplex channel, the channel is assigned the address 42. If a channel multiplexer is attached, the channels it provides may be assigned any subset of the addresses 42 through 127, as determined by the design of the multiplexer.

6.1.2 Channel Capacity

The capacity of a channel, i.e., the maximum rate of information transmission, is determined by the number of channels concurrently in operation and the byte rates of the units connected to these channels. The simplex and multiplex channels operate in different ways and, therefore, have different capacities.

The capacity of channel 40 (or the capacity of channel 41 if channel 40 is not in operation) is 80,000 bytes per second. If both simplex channels are in operation, the capacity of channel 41 is 44,000 bytes per second and their combined byte rates cannot exceed 100,000 bytes per second. If neither of the simplex channels is in operation and if no interruptions occur during an input-output operation, the capacity of the multiplex channel is 18,000 bytes per second.

If the byte rate of an adapter exceeds the capacity of the channel to which it is connected, or if the total byte rate of all channels in operation is too high, information may be lost. Other timing restrictions must be fulfilled for successful operation, and information may be lost even though the byte rate of a unit is less than the capacity of the channel to which it is connected. If information is lost during an input-output operation because of timing restrictions, an indication is given at the end of the operation.

Programming Note:

It is possible to operate a single type 729 IV tape unit or to operate simultaneously two type 7330 tape units using the simplex channels. A single 7330 tape unit may be operated at low density on the multiplex channel if neither of the simplex channels is in operation.

6.2 Channel Operation

Each of the channels 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 is transferred by a read or write operation, and the sense and control operations are used to transfer control information.

An input-output operation is initiated by a START CHANNEL instruction which specifies the channel to be used and the location of a control word which defines the area of storage to be used and the operation to be executed.

When a START CHANNEL instruction is given, the computer tests the status of the specified channel. If the channel is either busy or not operable, an alert is given. If the channel is operable and not busy, the computer fetches the specified control word and stores it in a register (simplex channel) or in the channel control word locations (multiplex channel) associated with the channel. The appropriate command is then sent to the adapter and information transfer is begun.

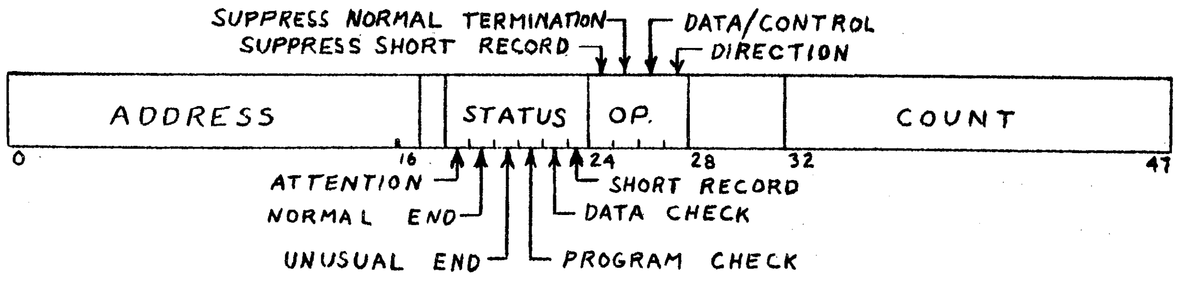
Data recorded on an external medium is divided into blocks. The length of a block depends on the medium; e.g., a block may be a card, a line of printing, or the information recorded between two consecutive gaps on tape. Reading and writing operations always start at the beginning of a block and are terminated at the end of the block or when the specified storage area is exhausted, whichever occurs first.

The amount of information which can be 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.

When an input-output operation is terminated, the channel control word indicates how much of the specified storage area was used and any special conditions which occurred during the operation. This control word is stored in locations 8n, 8n + 1, and 8n + 2, where n is the channel address.

6.2.1 Control Word Format

Input-output control words have the format shown below.



Bits 0 through 16, Address:

This field in the original control word specifies the first 8-bit byte in storage to be used. In the final control word, it addresses the byte following the last byte used.

Bits 18 through 23, Status Bits:

At the end of an input-output operation, these bits indicate the occurrence of certain conditions during the execution of the operation. Their contents at the beginning of an operation is ignored.

18 Attention: This bit indicates that an Attention signal has been received from the channel. It can be turned on at any time whether the channel is busy or idle.

19 Normal End: This bit indicates that no unusual conditions occurred during the execution of the input-output operation just terminated. An Attention signal is not considered an unusual condition.

20 Unusual End: This bit indicates that an unusual condition was detected by the adapter during the execution of an operation. A list of the conditions which may be detected by the adapter is given in section 6.4.2 .

21 Program Check: This bit is turned on if the channel attempts to transmit information to or from a storage location which is not provided. The operation is terminated immediately.

22 Data Check: This bit is turned on if the channel receives a byte with incorrect parity from an adapter during an input operation. It is never turned on during an output operation.

23 Short Record: This bit indicates that an input operation was terminated by the adapter because the end of the block on the external medium was reached before the storage area defined by the control word was filled. It is never turned on during an output operation.

<u>Bits 24 through 27, Operation Code:</u>

These bits specify the input-output operation to be executed and modify the interruption given at the end of the operation. They are not altered during the operation.

24 Suppress Short Record: For input operations, if this bit is zero and the Short Record bit is on at the end of the operation, a Special Condition interruption is given. If this bit is on, a Normal End interruption is given unless another special condition occurred during the operation. Output operations are not affected by the Suppress Short Record bit.

25 Suppress Normal Termination: If this bit is on, all Normal End interruptions are suppressed. Special condition interruptions are not affected.

26 Data/Control: If this bit is zero, data is transferred between the adapter and storage. If the bit is on, control information is transferred to or from the adapter. Control information transmitted to the adapter can be used to select a particular unit attached to the adapter, change the mode of operation of the adapter, or to initiate certain operations such as, rewind a tape. Control information transferred to storage is used to determine the status of the adapter following an Attention signal or an Unusual End.

27 Direction: If this bit is zero, an input operation is specified and information is transferred from the adapter to storage. A one specifies an output operation during which information is transferred to the adapter.

Bits 32 through 47, Count:

This field in the original control word specifies the number of 16-bit locations in the defined storage area. In the final control word, it indicates the number of locations in the defined area which were not used. If either the original storage area or the unused storage area at the end of an operation begins with the righthand byte of a storage location, this location is included in the count.

The original contents of bit 17 and bits 28 through 31 are ignored by the channel.

6.2.2 Definition of Core Storage Area

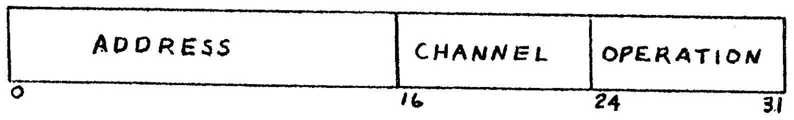
The defined storage area contains one or more bytes with consecutive addresses. The control word addresses the byte with the lowest address. If this is the lefthand byte of a storage location, the area contains an even number of bytes, and this number is twice the count given in the control word. If the control word addresses the righthand byte of a location, the number of bytes in the area is odd and one less than twice the count. During the input-output operation, the bytes in the defined area are used in order of increasing addresses beginning with the byte addressed by the control word. The last byte in the defined area is always the righthand byte of a storage location.

If the count is zero, it is interpreted as 65,536 (2^{16}). Thus, it is impossible for a control word to define an area containing no bytes. A zero count provides a convenient way of indicating that the amount of information transferred is to be governed by the block size of the external medium.

When executing an input-output operation, any storage location can be used. If a reference is made to a location above the limit of available storage, however, the operation is terminated immediately and the Program Check bit (bit 21) in the channel control word is turned on.

6.3 Instructions

Input-output instructions have the format shown below.



Bits 0 through 15 specify the location of the control word which defines the area of core storage to be used. This address is direct if bit 24 is zero; otherwise, it is indirect.

The channel to be used is specified by bits 16 through 23. The instruction will be rejected and code CNO, Channel Not Operational, will be stored in the interruption stream if the channel address is 0 through 39 or any unassigned address above 39.

The operation code is given in bits 25 through 31.

6.3.1 START CHANNEL

This instruction initiates the input-output operation specified by the control word which it addresses. If the adapter connected to the addressed channel controls two or more input-output units, the instruction applies to the unit last selected. If the addressed channel is busy or not operational, the appropriate interruption code is stored and no operation is initiated.

6.3.2 RELEASE CHANNEL

If the addressed channel is busy, this instruction causes information transfer to be terminated immediately. The only interruption code which can be stored as a result of a RELEASE instruction is CNO, Channel Not Operational. The interruption code which results from an input-output operation terminated by this instruction, however, is not suppressed. The control word address given in this instruction is ignored.

6.4 Input-Output Alerts

In order to permit the effective simultaneous execution of a program and several input-output operations, the latter are interlocked with the program by means of the interruption system. Two types of alerts are used, general alerts and channel alerts.

6.4.1 General Alerts

For these five alerts, the code stored in the interruption stream does not indicate the channel with which the causing condition is associated. Since these alerts can occur only during the initiation of an input-output operation, the channel can be determined from the last instruction executed before interruption , if the computer is enabled.

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 three alerts in this list can be caused by other instructions. They are described here only as far 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 causing instruction is suppressed.

Code 71, Address Invalid (AD)

This alert results if the control word address of an input-output instruction refers to a core storage location not provided in the system.

Code 70, Unended Sequence of Addresses (USA)

This alert results if a sequence of indirect control word addresses is not terminated by a direct address within one millisecond.

Code 72, Operation Code Invalid (OP)

If the input-output lock bit of the current program control word is one, any input-output instruction will cause this code to be stored in the interruption stream. This feature permits the 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 core storage assigned to another program.

Code 68, 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 unit is connected to the addressed channel, or if the channel is not provided in the system.

Code 69, Channel Busy (CB)

This interruption code is stored when an instruction addresses a channel which is busy.

6.4.2 Channel Alerts

The codes stored in the interruption stream by the two alerts described below identify the channel with which the alert is associated. These codes are of the form 2n + a where "n" is the channel address and "a" denotes the type of alert.

Code 2n, Channel n Normal End (CnNE)

This code is stored when an input-output operation using channel n is terminated, unless suppression of normal termination is specified in the control word or a special condition occurred during the operation. A short record is not considered a special condition if the Suppress Short Record bit is on in the control word.

Code 2n + 1, Channel n Special Condition (CnSC)

This code is stored when an input-output operation using channel n is terminated if a special condition occurred during the operation. It is also stored immediately if an Attention signal is received from channel n when the channel is not busy.

Special conditions which can occur during an input-output operation are:

1. An Attention signal may be received from the adapter.

2. The adapter may indicate the end of the operation by an Unusual End signal.

3. The channel may attempt to transmit information to or from a location which is above the limit of available storage.

4. A byte of information with incorrect parity may be detected by the channel during an input operation.

5. During an input operation, the end of the block of data may be reached before the count in the control word is reached. This is considered a special condition only if the Suppress Short Record bit in the control word is zero.

   All special conditions which occur during an operation are indicated by the status bits of the channel control and at the end of the operation.

If a Channel n Special Condition alert is caused by either an Attention signal or an Unusual End signal, the program can determine the exact cause of the alert by means of a sense execution. Several possible causes of an Unusual End are:

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

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

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

4. A data error is detected by the unit.

5. The adapter detects the malfunction of some of its components.

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

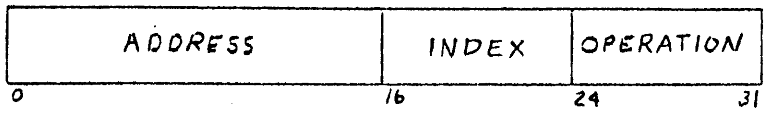
7. A tape mark has been sensed.

8. A byte of information was lost during the operation because of timing restrictions.

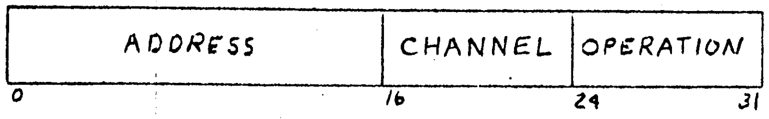
Appendix

Instruction Formats

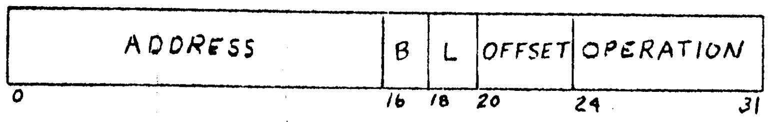
Floating point, control word, and accumulator operations.



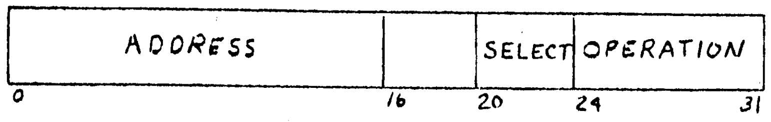
Channel operations.



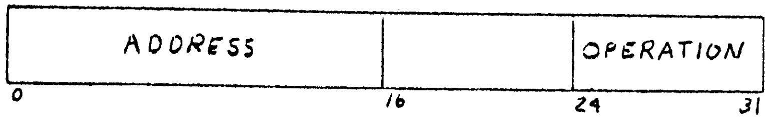
Variable-field-length operations.



Decision operations.

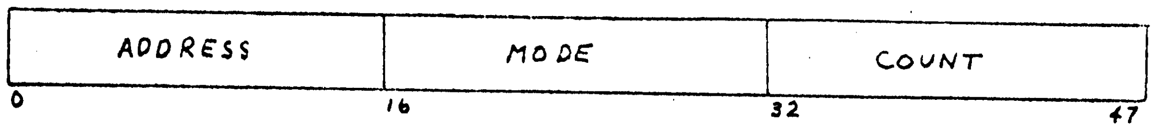


Interruption operations.

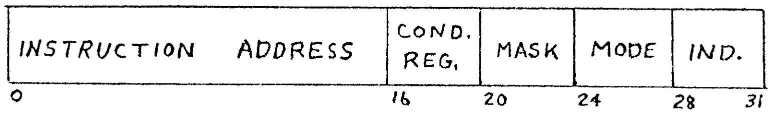


Control Word Formats

Channel control word.

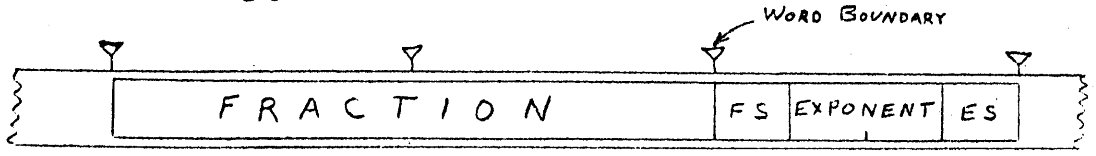


Program control word.

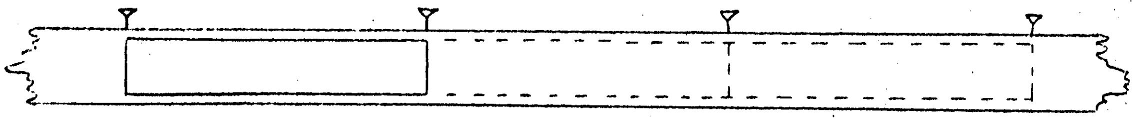


Data Formats

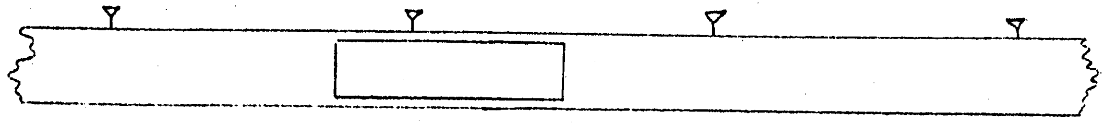
Floating point.



Full word.



Variable-length field.



Operation Codes

Code bits 24-31

Floating Point Arithmetic Operations

i u a 1 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

i 0 0 1 0 s c c

i	direct-indirect
s	load-store
0 0	accumulator
0 1	right half-fraction
1 0	left half-fraction
1 1	fraction

Control Word Operations

i 0 1 1 0 c c c

i	direct-indirect
0 0 0	diminish one
0 0 1	increment one
0 1 0	increment two
0 1 1	increment three
1 0 0	refill

Channel Operations

i 1 0 1 0 0 0 c

i	direct-indirect
0	start
1	release

(Operation Codes)

Load Condition Register Operation

```
i 0 1 0 1 0 1 0

i                    direct-indirect
```

Load Sign Operations

```
i 0 0 0 1 n 1 c

i                    direct-indirect
          n          positive-negative
              0      fraction sign
              1      exponent sign
```

Variable-Field-Length Arithmetic Operations

```
i 0 a 0 1 n 0 c

i                    direct-indirect
    a                signed-unsigned
          n          positive-negative
              0      reset add
              1      add
```

Connective Operations

```
i 1 t 0 x x x x

i                    direct-indirect
    t                store-test
        x x x x      connective code
```

Decision Operations

```
i 0 0 0 0 1 c c

i                    direct-indirect
            0 0      branch if any
            0 1      branch if none
            1 0      store PCW if any
            1 1      store PCW if none
```

Interruption Operations

```
i 0 0 0 0 0 c c

i                    direct-indirect
            0 0      interrupt intentionally
            0 0      enable and wait
            0 0      enable
```

Operation Times

	Cycles	µsec
Floating Point Operations		
1. Add and Reset Add	11	110
2. Multiply	38	380
3. Divide	50	500
Accumulator Operations		
1. One word transferred	3	30
2. Two words transferred	4	40
3. Three words transferred	5	50
Control Word Operations		
1. Increment or Decrement	4	40
2. Refill	5	50
Channel Operations		
1. Start Channel (simplex)	5	50
2. Start Channel (multiplex)	8	80
3. Start Channel (unsuccessful)	2	20
4. Release channel	2	20
Variable-Field-Length Operations		
1. Without word boundary crossover	4	40
2. With word boundary crossover	5	50
Sign Byte Load and Condition Load	4	40
Interruption and Decision Operations		
1. Unsuccessful	2	20
2. Successful interruptions	2	20
3. Successful branches	3	30
4. Successful store PCW	4	40
Indexing	2	20
Indirect Addressing	2	20