

PART 1 LANGUAGE DEFINITION

CHAPTER 1 INTRODUCTION

CHAPTER 2 MACHINE STRUCTURE

- 2.1 Introduction
- 2.2 Main Store
- 2.3 Non-Addressable Main Store
- 2.4 Scratch-Pad Store
- 2.5 Program Control and Arithmetic Unit
- 2.6 Input/Output Control
- 2.7 Read-Only Store

CHAPTER 3 THE USERCODE LANGUAGE

- 3.1 Program Structure
- 3.2 Usercode Coding Conventions
- 3.3 Basic Usercode Concepts
- 3.4 Expressions
- 3.5 Machine Instruction Statements
- 3.6 Translator Instruction Statements
- 3.7 Program Sectioning and Linking

CHAPTER 4 DATA FORMATS

- 4.1 Introduction
- 4.2 Packed Decimal
- 4.3 Zoned Decimal
- 4.4 Fixed Point
- 4.5 Floating Point
- 4.6 Logical

CHAPTER 5 INSTRUCTION FORMATS

- 5.1 Introduction
- 5.2 RR Format
- 5.3 RX Format
- 5.4 RS Format
- 5.5 SI Format
- 5.6 SS Format

CHAPTER 6 EXECUTABLE (MACHINE) INSTRUCTIONS

- 6.1 Introduction
- 6.2 Fixed Point Instructions
 - 6.2.1 Load Word (LR) (L)
 - 6.2.2 Load Halfword (LH)
 - 6.2.3 Load and Test (LTR)
 - 6.2.4 Load Complement (LCR)

- 6.2.5 Load Positive (LPR)
- 6.2.6 Load Negative (LNR)
- 6.2.7 Load Multiple (LM)
- 6.2.8 Add Word (AR) (A)
- 6.2.9 Add Halfword (AH)
- 6.2.10 Add Logical (ALR) (AL)
- 6.2.11 Subtract Word (SR) (S)
- 6.2.12 Subtract Halfword (SH)
- 6.2.13 Subtract Logical (SLR) (SL)
- 6.2.14 Compare Halfword (CH)
- 6.2.15 Compare Word (CR) (C)
- 6.2.16 Multiply Word (MR) (M)
- 6.2.17 Multiply Halfword (MH)
- 6.2.18 Divide (DR) (D)
- 6.2.19 Convert to Binary (CVB)
- 6.2.20 Convert to Decimal (CVD)
- 6.2.21 Store Word (ST)
- 6.2.22 Store Halfword (STH)
- 6.2.23 Store Multiple (STM)
- 6.2.24 Shift Left Single (SLA)
- 6.2.25 Shift Right Single (SRA)
- 6.2.26 Shift Left Double (SLDA)
- 6.2.27 Shift Right Double (SRDA)
- 6.2.28 Interrupt Action

6.3 Decimal Arithmetic Instructions

- 6.3.1 Add Decimal (AP)
- 6.3.2 Subtract Decimal (SP)
- 6.3.3 Zero and Add (ZAP)
- 6.3.4 Compare Decimal (CP)
- 6.3.5 Multiply Decimal (MP)
- 6.3.6 Divide Decimal (DP)
- 6.3.7 Pack (PACK)
- 6.3.8 Unpack (UNPK)
- 6.3.9 Move with Offset (MVO)
- 6.3.10 Interrupt Action

6.4 Logical Instructions

- 6.4.1 Move (MVI) (MVC)
- 6.4.2 Move Numerics (MVN)
- 6.4.3 Move Zones (MVZ)
- 6.4.4 Compare Logical (CLR) (CL) (CLI) (CLC)
- 6.4.5 And (NR) (N) (NI) (NC)
- 6.4.6 Or (OR) (O) (OI) (OC)
- 6.4.7 Exclusive OR (XR) (X) (XI) (XC)
- 6.4.8 Test under Mask (TM)
- 6.4.9 Insert Character (IC)
- 6.4.10 Store Character (STC)
- 6.4.11 Load Address (LA)
- 6.4.12 Translate (TR)
- 6.4.13 Translate and Test (TRT)
- 6.4.14 Edit (ED)
- 6.4.15 Edit and Mark (EDMK)

- 6.4.16 Shift Left Single Logical (SLL)
- 6.4.17 Shift Right Single Logical (SRL)
- 6.4.18 Shift Left Double Logical (SLDL)
- 6.4.19 Shift Right Double Logical (SRDL)
- 6.4.20 Interrupt Action

6.5 Branching Instructions

- 6.5.1 Branch on Condition (BCR)(BC)
- 6.5.2 Branch and Link (BALR)(BAL)
- 6.5.3 Branch on Count (BCTR)(BCT)
- 6.5.4 Branch on Index High (BXH)
- 6.5.5 Branch on Index Low or Equal (BXLE)
- 6.5.6 Execute (EX)
- 6.5.7 Interrupt Action

6.6 Privileged Instructions

- 6.6.1 Introduction
- 6.6.2 Load Scratch Pad (LSP)
- 6.6.3 Store Scratch Pad (SSP)
- 6.6.4 Program Control (PC)
- 6.6.5 Idle (IDL)
- 6.6.6 Diagnose (DIG)
- 6.6.7 Start Device (SDV)
- 6.6.8 Halt Device (HDV)
- 6.6.9 Test Device (TDV)
- 6.6.10 Check Channel (CKC)
- 6.6.11 Insert Storage Key (ISK)
- 6.6.12 Set Storage Key (SSK)
- 6.6.13 Write Direct (WRD)
- 6.6.14 Read Direct (RRD)
- 6.6.15 Interrupt Action

6.7 Processor State Control Instructions

- 6.7.1 Introduction
- 6.7.2 Interrupt Action
- 6.7.3 Supervisor Call (SVC)
- 6.7.4 Set Program Mask (SPM)

6.8 Floating Point Instructions

- 6.8.1 Load and Test (LTCR)(LTDR)
- 6.8.2 Load Complement (LCER)(LCDR)
- 6.8.3 Load Positive (LPER)(LPDR)
- 6.8.4 Load Negative (LNCR)(LNDR)
- 6.8.5 Add Normalised (AER)(AE)(ADR)(AD)
- 6.8.6 Add Unnormalised (AUR)(AU)(AWR)(AW)
- 6.8.7 Subtract Normalised (SER)(SE)(SDR)(SD)
- 6.8.8 Subtract Unnormalised (SUR)(SU)(SWR)(SW)
- 6.8.9 Compare (CER)(CE)(CDR)(CD)
- 6.8.10 Halve (HER)(HDR)
- 6.8.11 Store (STE)(STD)
- 6.8.12 Multiply (MER)(ME)(MDR)(MD)

- 6.8.13 Divide (DER)(DE)(DDR)(DD)
- 6.8.14 Load (LER)(LE)(LDR)(LD)
- 6.8.15 Interrupt Action
- 6.8.16 Normalisation

CHAPTER 7 TRANSLATOR INSTRUCTIONS

- 7.1 Extended Mnemonic Codes
- 7.2 Base Register Usage
 - 7.2.1 Use base register (USING)
 - 7.2.2 Drop base register (DROP)
- 7.3 Equating Symbols with Expressions
 - 7.3.1 Equate Symbol (EQU)
- 7.4 Defining Storage and Constants
 - 7.4.1 Defining Constants (DC)
 - 7.4.2 Defining Storage (DS)
- 7.5 Programming Sectioning and Control Instructions
 - 7.5.1 Start Translation (START)
 - 7.5.2 End Translation (END)
 - 7.5.3 Identify Control Section (CSECT)
 - 7.5.4 Identify Dummy Section (DSECT)
 - 7.5.5 Identify Entry Point Symbol (ENTRY)
 - 7.5.6 Identify External Symbol (EXTRN)
 - 7.5.7 Set Location Counter (ORG)
 - 7.5.8 Begin Literal Pool (LTORG)
 - 7.5.9 Input Sequence Check (ISEQ)
- 7.6 Control of Translator Listing
 - 7.6.1 Identify Translator Output (TITLE)
 - 7.6.2 Print Optional Data (PRINT)
 - 7.6.3 Translator Option (AOPTN)
 - 7.6.4 Start New Page (EJECT)
 - 7.6.5 Space Listing (SPACE)
 - 7.6.6 Assembler Log (ALOG)
- 7.7 Define Channel Command Word (CCW)
- 7.8 Conditional No Operation (CNOP)
- 7.9 Define Common Control Section (COM)
- 7.10 Macro Storage within the Translator
 - 7.10.1 Pre-Store Library Macros (MCALL)
 - 7.10.2 Store Macros by Priority (MPRTY)

CHAPTER 8 INPUT/OUTPUT OPERATION

- 8.1 Introduction
- 8.2 Input/Output Channels
 - 8.2.1 Selector Channels
 - 8.2.2 Multiplexor Channel
 - 8.2.3 Input/Output Initiation
 - 8.2.4 Channel Servicing

- 8.2.5 Channel Address Word (CAW)
- 8.2.6 Channel Command Word (CCW)
- 8.3 Input/Output Channel Registers
- 8.4 Input/Output Instructions
 - 8.4.1 Start Device Instruction
 - 8.4.2 Halt Device Instruction
 - 8.4.3 Test Device Instruction
 - 8.4.4 Check Channel Instruction
- 8.5 Input/Output Status Indicators
 - 8.5.1 Condition Code
 - 8.5.2 Channel Status Byte
 - 8.5.3 Standard Device Byte
 - 8.5.4 Sense Bytes
- 8.6 Channel Servicing
 - 8.6.1 Servicing a Data Transfer
 - 8.6.2 End and Chaining Servicing
 - 8.6.3 Interrupt Servicing

CHAPTER 9 PROGRAM INTERRUPT

- 9.1 Introduction
- 9.2 Processor States
- 9.3 Processor State Registers
 - 9.3.1 Program Counter
 - 9.3.2 General Registers
 - 9.3.3 Floating-Point Registers
 - 9.3.4 Interrupt Status-Registers
 - 9.3.5 Interrupt Mask Registers
 - 9.3.6 Program Mask Registers
 - 9.3.7 Register Addressing
 - 9.3.8 Interrupt Flag Register
- 9.4 Interrupt Conditions
- 9.5 Interrupt Mechanisation
 - 9.5.1 Automatic Interrupt
 - 9.5.2 Program Controlled Interrupt
 - 9.5.3 General Notes on Program Interrupt

APPENDIX A1 List of Instructions

APPENDIX A2 Character Codes

APPENDIX A3 Reserved Store Locations

APPENDIX A4 Sample coding sheet

APPENDIX A5 Hexadecimal conversion and useful numbers

APPENDIX A6 Index

1. Introduction

The 4-50 Usercode language provides a simple and flexible means of writing a program. It contains all the features normally to be expected from an Assembly system, such as mnemonic operation codes, symbolic and relative addressing, flexible constant declarations and program linking instructions.

The Usercode program is translated into machine code by the 4-50 Usercode Translator, which also performs many useful clerical functions for the programmer and contains extensive error checking. Several versions of the Translator are available for use with different operating systems. The input to the Translator (the source language) may be on punched cards, paper tape or magnetic tape. These media are also available to store the output from the Translator (the object program). The Translator produces a printed record of the source and object programs and a list of errors detected.

2. Machine structure

2.1 INTRODUCTION

This section is intended to give the programmer an outline of the structure of the hardware of system 4-50.

The 4-50 consists of main store, non-addressable main store, scratch-pad store, program control and arithmetic unit, input-output control, and read-only store.

2.2 MAIN STORE

The main store of the system 4-50 is central storage for both data to be processed and the controlling instructions. The smallest addressable unit of information in the main store is one byte (eight bits).

The store cycle time is the time required to access and transfer a halfword (4-50) from the main store to the store register and regenerate the information in the main store. The store cycle time is 1.4 microseconds. The store size may be from 16KB to 256KB (Kilobytes).

The first 128 locations of the main store are reserved for hardware use and must not be used by the program.

2.3 NON-ADDRESSABLE MAIN STORE

A non-addressable main store is an addition to the main store and cannot be addressed by programming. It contains the subchannel registers that control the operation of input/output devices on the multiplexor channel. (See Chapter 8 for full details of the multiplexor channel). A set of three 32-bit registers services each device on the multiplexor channel. The number of subchannel register sets and the number of devices that can be connected to the multiplexor channel are determined by the capacity of the main store, which is given in the table below:

MAIN STORE CAPACITY AND MULTIPLEXOR SETS/DEVICES

Capacity of Main Store (Bytes)	Number of Multiplexor Subchannel Register Sets/Devices
16,384 (16KB)	64
32,768 (32KB)	128
65,536 (64KB)	256
131,072 (128KB)	256
262,144 (256KB)	256

2.4 SCRATCH-PAD STORE

The scratch-pad store is a storage device consisting of 128 four-byte words, the cycle time of which is 300 nanoseconds. Each word in the scratch-pad store is uniquely addressed.

The following registers are contained in the scratch-pad store: (see Appendix 3 for a diagram of these register assignments):-

1. Processor Utility Registers - All locations designated as processor utility registers are used by the processor for program control and cannot be used by the program.

2. General Registers - These locations are the general registers for each processor state. These registers are used by the program for base addressing, for indexing, or for storing operands.

Note: The 4-50 has four processor states that pertain to system and program interrupts. (See Chapter 9 for details of interrupts).

3. Interrupt Mask Registers - An Interrupt Mask register for each processor state permits or inhibits 32 interrupt conditions.

4. Interrupt Status Registers - An Interrupt Status register for each processor state stores interrupt identification information and operational control information. This register contains indications of the last state interrupted, the protection key, the decimal mode (ISO or EBCDIC), the privileged mode bit, and the supervisor call identification.

5. Program Counter - A Program Counter for each processor state contains the main store address of the next instruction to be executed, the condition code, the instruction length code, and the program mask.

6. Input/Output Channel Registers - A set of six registers for each selector channel controls input/output operation. A set of four registers for the multiplexor channel controls initiation and termination of input/output operations on the multiplexor channel (see Chapter 8 for input/output operations).

7. Floating-Point Registers - Four Floating-Point registers (each is two words long) are used in floating-point arithmetic.

8. Interrupt Flag Register - One Interrupt Flag register is provided. When an interrupt condition occurs, a bit associated with this condition is set in the Interrupt Flag register.

2.5 PROGRAM CONTROL AND ARITHMETIC UNIT

The program control and arithmetic unit interprets and executes the instructions stored in the main store. Registers and indicators monitor the sequence of operations, perform automatic accuracy checks, and communicate with the standard interface in the control of input/output devices.

2.6 INPUT/OUTPUT CONTROL

The 4-50 communicates with all input-output devices through the standard interface.

The 4-50 can have up to three selector channels (optional). Each selector channel contains two standard interface trunks. Each standard interface trunk controls one device subsystem (from 1 to 16 devices). All selector channels can operate simultaneously.

In addition to the selector channels, a multiplexor channel is standard equipment on the 4-50; it contains eight standard interface trunks, each of which controls one device subsystem. All trunks on the multiplexor can operate simultaneously. Also, the multiplexor channel and all selector channels can operate simultaneously.

Input/output operations are described fully in Chapter 8.

2.7 READ-ONLY STORE

The read-only store is a standard feature of the 4-50. It consists of 2048 54-bit words, a 12-bit address register, and a 54-bit store register. The effective cycle time is 480 nanoseconds with a 54-bit access. Like the scratch-pad store, the read-only store is not accessible by the programmer.

The stored logic contained in the read-only store controls the elementary operations of the 4-50. Because of the fast access time, significant reductions in instruction execution timings are possible.



3. The usercode language

3.1 PROGRAM STRUCTURE

The unit of translation is known as a Usercode module. A complete Usercode program, as written by the programmer, consists of one or more such modules. Each such module consists of one or more control sections and is translated into one object code module. Each control section consists of Usercode statements which are of five kinds:

- (i) machine instruction statements
- (ii) Translator instruction statements
- (iii) comment statements
- (iv) macro instruction statements
- (v) macro definition language statements.

3.1.1 Machine Instruction Statements

These are symbolic representations of System 4-50 machine instructions. The Translator produces an equivalent machine instruction in the object code module for each machine instruction statement in the Usercode module. These statements are discussed in more detail in chapter 3.5.

3.1.2 Translator Instruction Statements

The Translator, in addition to its translation function provides auxiliary functions that assist the programmer in documenting programs, in controlling storage location address assignment, in program sectioning and linking, in data and storage field definition and in controlling the Translator itself. With a few exceptions these statements do not result in the generation of any object code by the Translator. They are discussed in more detail in chapter 7.

3.1.3 Comment Statements

These consist of an asterisk in the begin column (column 1), followed by any sequence of Usercode characters. They communicate nothing to the Translator.

3.1.4 Macro Instruction Statements

These cause the Translator to retrieve a specially coded symbolic macro definition either from the macro library or from those macro definitions supplied in the Usercode module. The macro definition together with the operands of the macro instruction define a set of Usercode statements which are generated and substituted for the macro instruction statement for translation. This procedure is known as macro expansion.

Certain system macro definitions are provided as part of the macro library. Macro definitions may be added to the macro library by a system maintenance program. These macro definitions may be referenced by any module to be translated. Furthermore macro definitions may be supplied in a Usercode module. These definitions may only be referenced within

that module and must appear within the module prior to their first use.

3.1.5 Macro Definition Language Statements

These statements are described fully in section 4.3.1 of the Software Manual.

3.2 USERCODE CODING CONVENTIONS

3.2.1 Statement Boundaries

Statements are written in columns 1 to 71 of a statement line. Column 1 is referred to as the 'begin' column and column 71 is referred to as the 'end' column. One continuation line may be used by any statement except a macro instruction statement which may use up to 49 continuation lines. A continuation line is used by writing a non-blank character, not part of the statement, in the column after the end column, and continuing the statement on the next line starting at column 16, referred to as the 'continue' column. Columns before the 'continue' column must be blank.

For comment cards, that is, cards which have an asterisk in column 1, column 72 does not function as a 'continue' indicator and the succeeding card must also have an asterisk in column 1.

The nominal layout of a statement is as follows:

Usercode instructions:

columns	1- 8	name
	10-14	operation
	16-71	operand, comments. At least one blank column must separate the operand and comment.
	72	continuation indicator
	73-80	serial number

Comment cards:

columns	1	asterisk
	2-72	comment
	73-80	serial number

The Usercode Translator accepts a "free format" instruction statement, such that column 1 indicates the name field start but the operation may begin anywhere provided that at least one blank column separates the last character of the actual name from the first character of the operation: in particular, if column 1 is blank the first non-blank column is taken as the start of the operation field. Similarly, at least one blank column is sufficient to separate the operation from the operand, and the operand from the comments field.

3.2.2 Statement format

Any statement except a comment statement consists of, from left to right, a name entry, an operation entry, one or more operand entries and a comments entry. The entries must be separated by one or more blanks.

Name entry This is a symbol used to label a statement. It is optional but if present must begin in the begin column. If the begin column is blank no name is assumed present.

Operation entry This consists of up to 5 non blank characters and specifies the machine operation, Translator function or macro action desired. It is mandatory and may not appear on a continuation line, neither must the blank which follows it. Valid operation codes for machine and Translator instructions are listed in Appendix 1.

Operand entries These either identify and describe data to be acted upon by a machine instruction by indicating storage locations, masks, storage area lengths or types of data etc., or they convey information to the Translator. The number and type of the operands depend on the particular instructions (see Appendix 1). If more than one operand is present they are separated by commas. Operands may be optional or mandatory. An operand may contain blanks only in a character constant, a character self defining term or a character literal.

Comment entry This entry may contain any characters from the Usercode character set. Where all operand entries are optional and all are omitted, if a comments entry is desired it must be separated from the operation entry by a comma preceded and followed by one or more blanks.

Identification-sequence entries These are optional and do not form part of the statement line. If used they normally occupy columns 73 to 80.

3.2.3 Usercode Character Set

There are 29 letters namely A to Z, @ , # and \$.

There are 10 digits namely 0 to 9.

There are 12 special characters namely + - , = . * () ' / & space (plus, minus, comma, equals sign, period, asterisk, left parenthesis, right parenthesis, quote, solidus, ampersand, space)

In addition any EBCDIC character (see Appendix 2) may be used in a character constant, a character self defining term or a character literal.

3.3 BASIC USERCODE CONCEPTS

3.3.1 Terms

Symbols, symbol length attribute references, location counter references, self defining terms, and literals are examples of terms. Each may be used in an expression.

3.3.2 Attributes

Associated with each term or expression are four attributes ; a value attribute or value, a length attribute, a relocatability attribute, and a control section attribute. These attributes are either inherent in a term or become associated with a term or expression by virtue of its appearance within the module.

Value attribute This is purely numeric and denotes a storage location address, a general register number, or other information required by a statement.

Length attribute Where a value attribute refers to a storage location address it may be necessary to represent the length in bytes of the storage area, the address of whose first byte is denoted by the value attribute. This purpose is served by the length attribute.

Relocatability attribute Terms are said to be absolute or relocatable. An absolute term is one whose value does not depend on which storage locations are assigned by the Translator to the object code module. A relocatable term is one whose value would increase (or decrease) by n were the storage location addresses assigned by the Translator to the object code module to be increased (or decreased) by n.

Relocation is the loading of an object code module into storage locations not identical with those assigned by the Translator.

Control Section attribute. An absolute term has no such attribute. A relocatable term has a control section attribute which depends on the kind of term and on its definition within the module.

3.3.3 The location counter

A location counter is used to assign storage location addresses to object code. As each machine instruction or data item is translated the location

counter is first aligned to the proper boundary, if such alignment is necessary, and then incremented by the length of the translated item. Thus it always points to the next available location.

This point is discussed in more detail in Chapter 3.7.1.

3.3.4 Symbol

A symbol consists of a letter followed by up to 7 characters each of which may be either a letter or a digit. It may denote a storage location address or an arbitrary value.

Symbol Definition A symbol may be defined once per module in either of two ways, either by its appearance as the name entry for some statement, once only except in the case of the name of a control section (see START, CSECT, DSECT), or by its appearance as the operand entry for one or more EXTRN statements. The definition of a symbol also involves the definition of its attributes. These attributes are maintained by the Translator in an internal table - the symbol table - to which reference is made when any other use is made of the symbol. All symbols used in an operand entry (except the operand of V type address constants) must be defined in the module.

Symbol attribute definition When defined as the name entry of an EQU statement, the attributes of the symbol are those of the operand entry. In particular, the symbol may be relocatable or absolute.

When defined as the name entry of any other statement, the value attribute is the value of the location counter after any alignment has been performed, but before incrementing. Therefore the value attribute is the address of the leftmost storage location assigned to the translated statement. The length attribute is the length in bytes of the storage field whose address is denoted by the symbol. Such a symbol is relocatable. The control section attribute is that of the control section in which the definition occurs.

The value assigned to a symbol defined in an EXTRN statement (an external symbol) is zero. The length attribute of an external symbol is one. An external symbol is relocatable. Each external symbol is considered to have a control section attribute different from that of any other symbol within the module.

The value of a symbol may not be negative nor be more than 524287.

3.3.5 Symbol length attribute reference

This term is written as a symbol preceded by the two characters L' (L quote). Its value is the length attribute of the symbol. Its length attribute is one. It is absolute, and has no control section attribute.

3.3.6 Location counter reference

This term is written as a single asterisk. Its attributes are exactly those which a symbol would have which appeared as a name entry to the statement in which the reference appears, were a name entry possible.

In the case of such a reference within an ORG instruction, the value is that of the location counter for the control section in which the statement appears, the length attribute is undefined, the term is relocatable and its control section attribute is that of the control section in which the statement appears.

3.3.7 Self defining term

This may be one of four kinds, binary, hexadecimal, decimal, and character. Its value is not assigned by the Translator but is inherent in the term. Its length attribute is one. It is absolute, and has no control section attribute.

Binary self defining term This term is written as 1 to 24 binary digits (that is 0 or 1) enclosed in single quotes and preceded by the letter B. Its value is that represented by the binary number.

Hexadecimal self defining term This term is written as 1 to 6 hexadecimal digits enclosed in single quotes and preceded by the letter X. Its value is that represented by the hexadecimal number.

Decimal self defining term This term is written as 1 to 8 decimal digits. Its value is that represented by the decimal number, and may not exceed 16,777,215 (or $2^{24}-1$).

Character self defining term This term is written as 1 to 3 EBCDIC characters enclosed in single quotes and optionally preceded by the letter C. Any of the 256 punch combinations may be used for the characters. However, because the characters quote and ampersand have special significance to the Translator, for each single quote or ampersand desired in such a term, two consecutive quotes or two ampersands must be written. Each such pair of quotes or ampersands and each remaining character in the sequence of three is converted to its 8 bit equivalent (see Appendix 2) and the value represented by the resulting 24 bit binary number is the value of the term.

3.3.8 Literal

This term is written as an equals sign followed by any valid form for an operand entry to a DC statement, with the exceptions of S-type address constants and constants with zero duplication factor. The appearance of a literal in a source language statement directs the Translator to assemble the constant specified by the literal into a "literal pool". The value attribute of the literal is the address of the first storage location occupied by the assembled constant. Its length attribute is that of the constant. The term is relocatable. Its control section attribute is that of the control section in which the literal pool is placed. This will usually be the first control section of the module, at the end of which, beginning at a double word boundary, the literal pool is placed by the Translator. However, by the use of the LTORG statement, the programmer may arrange for a literal pool to be placed elsewhere, and for multiple pools.

In general, a literal may appear wherever a storage location address is permitted.

Restrictions:

1. Only one literal may appear in any one machine instruction statement.
2. A literal term may not be combined with other terms.
3. A literal may not be specified in an address constant.

Duplicate literals The occurrence of duplicate literals within the same literal pool is avoided in some cases by the Translator. A literal is not assembled into the current literal pool if the character string which represents its definition contains no asterisk, and is identical with that of a literal which has previously been assembled into that literal pool. In this case, its value attribute is that of the previously occurring and identical literal.

3.3.9 Inner expressions

Terms are used in combination with arithmetic operators to form expressions. A term may be a symbol, a symbol length attribute reference, a location counter reference, a self defining term, or a literal. A term may also be a relocatable or absolute expression enclosed within single parentheses. This expression is said to be inner to the expression in which it is used as a term. An inner expression may itself contain inner expressions used as terms; these further inner expressions are said not to be inner to the first expression.

3.3.10 Pairing

Two relocatable terms within an expression are said to be paired if they have the same control section attribute and if just one is preceded by a minus sign. A term within an inner expression may not be paired with a term not within that inner expression.

3.4 EXPRESSIONS

3.4.1 Construction of expressions

This proceeds according to the following rules.

1. An expression must begin and end with a term.
2. Each relocatable term may be preceded by or followed by + or -.
3. Each absolute term may be preceded by or followed by + - * /
4. Each operator must be followed by a term.
5. A compound expression (see Chapter 3.4.3) may not contain a literal.
6. The number of terms within an expression is limited according to the rule given under 'Value of an expression' below.
7. The final value of an expression must be less than 2^{19} . Intermediate values must be less than 2^{31} . Only absolute and complex relocatable expressions may have a negative value, and this must exceed -2^{19} .

3.4.2 Operator precedence

In the evaluation of an expression the order of evaluation of operators is governed by their precedence, which is now defined.

1. Any operator within an inner expression has greater precedence than any operator which precedes the inner expression.
2. Otherwise, * and / have equal precedence, + and - have equal precedence and * and / each have greater precedence than either + or -.

3.4.3 Expression attributes

An expression has value, length, relocatability and control section attributes whose derivation from those of its constituent terms is defined below. In addition it has the attribute of being simple or compound.

A simple expression is one containing just one term, which is not an inner expression.

A compound expression is any other expression.

The derivation of the remaining attributes is now defined.

Value of an expression The order of evaluation proceeds according to the normal arithmetic convention.

The expression is scanned from left to right and as each term (including inner expressions) is encountered, it is evaluated before the following operator is inspected.

As each operator is encountered it is not performed but its precedence is used to determine the performance of all previously encountered but as yet unperformed operators as follows. If there is no such operator, scanning proceeds. If the most recent such operator is of lower or equal precedence, then scanning proceeds. If the most recent such operator is of greater precedence then it is performed and the comparison with the current operator proceeds with the next most recent such operator.

When the last term of an expression is encountered, all previously encountered, but as yet unperformed operators are performed in the reverse order of their occurrence. The last term of an inner expression only causes operators within the inner expression to be so performed.

Rule: The number of encountered but unperformed operators may at no point exceed 3. (This implies in addition that no more than three nested levels of parentheses may be used.)

Division by zero is valid and yields a zero result.

Division yields an integer result and any fractional part of the result will be lost.

Absolute and complex relocatable expressions (see below) may have a negative value; relocatable expressions may not.

Length attribute of an expression. This is that of its leftmost term.

Relocatability of an Expression

An expression is either absolute or relocatable; it is relocatable if its value at program execution time is dependent upon the position of the program within memory.

For example if A,B,C are symbols then A,A+6 are simple relocatable, requiring only one 'compensation'. The expressions A+B, 2*A, are complex relocatable because a double compensation is needed as the program is loaded. If A,B are paired (see 3.3.10), then their need for 'compensations' cancel and the expression becomes absolute for example, A-B. Note that 2*A-B is simple relocatable, being equivalent to A+(A-B).

In instruction operands **only** positive absolute or simple relocatable expressions are permitted, because there is only one means of compensating, viz. the base register for the operand. Certain instructions **require** one or both of their operands to be absolute, because no base register is involved.

3.5 MACHINE INSTRUCTION STATEMENTS

Each machine instruction may be written symbolically as a Usercode statement. The symbolic format is similar to, but does not duplicate, the actual machine instruction format and thus varies according to the particular machine instruction. Within each basic format further variations are possible. The machine instruction formats are defined in Chapter 5.

Machine instruction statements are written in a similar way to other Usercode statements, as defined in Chapter 3.2.

3.5.1 Instruction alignment and checking

Each machine instruction is aligned automatically by the Translator to lie on a halfword boundary. If any bytes are skipped for this reason they are made to contain hexadecimal zero. Each expression within a machine instruction statement that specifies a storage location address is checked to ensure that it refers to an appropriate boundary for the machine instruction.

3.5.2 Machine instruction operand subfields

The information assembled into a machine instruction is of several kinds as follows:-

the number of a general register used as a base;

the number of a general register used as an index;

an 8 bit or 4 bit value, known as immediate data which is actually assembled into the machine instruction and is required during its execution, being, for example, input and output trunk and unit numbers, or comparison masks;

a length of a storage field;

a displacement which is added, during instruction execution, to the sum of the contents of the base and the contents of the index, if any, to form the storage location address of the operand of the machine instruction;

the number of a general register which is neither a base nor an index.

Explicit and implicit formats When required in a machine instruction, some of the above information may be present (this is known as explicit specification), or omitted when a suitable value is assumed by the Translator, (this is known as implicit specification). The subfields which can be implicit are the base, the index and the length subfields. The format used for each combination of possibilities is defined in the following table. Immediate data when required, may never be omitted and except for trunk and unit numbers is written as an absolute expression whose value, always positive and not exceeding 255, depends on the particular instruction.

Table 1 : Explicit and implicit formats

	Explicit base	Implicit base
explicit length	D(L,B)	S(L)
implicit length	D(,B)	S
explicit index	D(X,B)	S(X)
implicit index	not possible	S
neither length nor index	D(B)	S

where B and X are absolute expressions having non-negative values less than 16;

L is an absolute expression having a non-negative value either less than 256 or less than 16 depending on the instruction;

D is an absolute expression having a non-negative value less than 4096;

S is a relocatable or absolute expression having a non-negative value less than 524288 (2^{19}).

Base field This may be specified explicitly as an absolute expression having a non-negative value less than 16. In this case the displacement field must be written as an absolute expression having a non-negative value less than 4096.

If the base is to be implicit, it is omitted and the storage location address is written in place of the displacement, as a relocatable or absolute expression having a non-negative value less than 524288 or 2^{19} . A suitable base is provided automatically by the Translator as follows;

Automatic base register provision This proceeds according to the following rules.

(a) If the expression is absolute and has a value between 0 and 4095 inclusive, then register zero is always used.

(b) Otherwise a register is available for use as an implied base for an expression occurring in a given statement of a module if previously, and within the module, it has been declared in a suitable USING statement, and not since declared in another USING statement or DROP statement.

A suitable USING statement is one which associates the register with an expression whose relocatability and control section attributes are the same as those of the original expression and whose value differs from that of the original expression by between 0 and 4095 inclusive.

(c) If more than one register is thus available, that yielding the lowest displacement is used.

(d) If more than one register yields the same displacement the highest numbered is used.

Index field This may be specified explicitly as an absolute expression having a non-negative value less than 16. If omitted, a value of zero is assumed.

Immediate data This may not be omitted if required by the machine instruction and is always written as a separate operand entry either, in the case of trunk and unit numbers, as an absolute expression followed by a parenthesised absolute expression, both expressions having non-negative values less than 16; or in all other cases as a single absolute expression having a non-negative value less than 256. An example of the first form is 3(4) and an example of the second form is B'1010'.

Length subfield This may be specified explicitly as an absolute expression whose value, non-negative and less than 256 (sometime less than 16 depending on the machine instruction), is the length in bytes of the storage field in question. The value assembled into the machine instruction is one less than the value of the expression. If the length is implicit the value assembled is one less than the length attribute of the expression.

A length subfield of one may be specified as either zero or one as desired.

Displacement subfield This is described in 'Base Field' above.

Other registers These are specified, as a complete operand, as an absolute expression having a non-negative value less than 16.

3.6 TRANSLATOR INSTRUCTION STATEMENTS

These are defined in Chapter 7.

3.7 PROGRAM SECTIONING AND LINKING

It is often convenient, or necessary, to write a program in independent

parts, combining these parts after Translation. These parts are called control sections (CSECTS) and the Translator processes a source module made up of one or more CSECTS. The resultant object module is combined with others by the Composer program, whose unit of data is the CSECT.

The Usercode Language provides control statements which will appear in a preface to the object module, indicating how Composer is to process inter-CSECT references between different object modules. These are the ENTRY and EXTRN statements, together with V-type address constants; in addition to the CSECT statement there are also DSECT and COM (for COMMON) statements.

3.7.1 Use of ENTRY/EXTRN

When an instruction in one object module is to address symbolic data (see 3.7.2) in a second module, that symbol is declared as an EXTRN in the first module and an ENTRY in the second. A special base register must be reserved in the first module, as the link will be effected by Composer's setting the appropriate ENTRY address in the address constant to load the base register. Thus the first module must contain a sequence of instructions in the following form:

	EXTRN	ALPHA	
	USING	ALPHA,8	
	L	8,BETA	
	MVC	GAMMA,ALPHA	
<u>or</u>	B	ALPHA	
	⋮		
	⋮		
BETA	DC	A(ALPHA)	

where 8 is the reserved base register number.

3.7.2 Use of V-constants

When an instruction in one object module is to branch to a symbolic instruction in a second module, the symbol is declared as a V-constant in the first module and an ENTRY in the second, and a register is used for the actual branch. The sequence of coding in the first module is

	L	8,ALPHA	
	BR	8	go to BETA
	⋮		
	⋮		
ALPHA	DC	V(BETA)	

Composer will suitably adjust the value of the constant ALPHA.

NOTES

- A In certain circumstances EXTRNs and V-constants may be used interchangeably. This depends upon how Composer will interpret them, and the Composer manual for the particular operating regime should be consulted.
- B In certain regimes, the use of V-constants for branching is mandatory, as the destination CSECT will not necessarily be in

memory; the constant ALPHA becomes the address of a control routine to fetch the CSECT.

3.7.3 Use of COMMON

This is designed to assist the Composition of modules derived from various language processors, for example, FORTRAN. A COMMON control section behaves like a CSECT containing only 'define storage' statements, and generates only a storage allocation requirement without corresponding object code.

3.7.4 Use of DSECT (Dummy CSECT)

This is used to define an imaginary area of storage, which either corresponds to (1) part of a CSECT in this module or (2) to part or whole of a CSECT in another module.

It exists at the source code level only, and no object code is generated. It is addressed like a CSECT, that is, requires its own base register even in case (2).

(1) Used in this manner, a DSECT can assign symbolic names to addresses which continually alter at execution time, for example, the components of a record in a multi-record input area. In this case the "base register" is set up by the File Control Package.

Alternatively, the DSECT is contained in a subroutine (or a macro) to allow that subroutine to address many different sets of parameters by a common set of names. The appropriate base register is to be set before entry to the routine, which if it is a macro will contain the USING statement for the DSECT where the register can be supplied as a symbolic parameter.

(2) In this manner, no EXTRNS/ENTRYS are needed for addressing the other module's CSECT and symbols representing consecutive memory are evaluated by the Translator to employ the base register declared for the other CSECT, with displacements derived from the source code representation of that CSECT. However, the base register cannot be set up in this module's CSECT and must be established by instructions obeyed in the other CSECT at execution time.

3.7.5 Modules containing multiple CSECT,DSECT,COM statements.

CSECTS are defined by their name, and statements for several CSECTS can be interleaved provided that a new CSECT statement with the appropriate name precedes the resumption of coding for the previously defined CSECT. Text is output in the same order as the source, but the loading address ensures continuity of object code. Different CSECTS are placed in adjacent memory, in order of occurrence; similarly with different COM statements, except that the first COM will be positioned after the last CSECT. DSECTS do not generate object code, but can be interrupted and resumed as for CSECTS. An independent location counter is maintained by the Translator for all CSECTS/DSECTS/COMMONS, resolved by reference to the subset for this CSECT of current USING statements.

3.7.6 Addressing CSECTS within a module

Although the Translator places two or more CSECTS in adjacent memory, separate base registers are required for each even if the first occupies less than 4096 bytes. This is because in certain operating regimes Composer may separate the CSECTS. No references should be made to a point later than that defined by the final Usercode statement in the CSECT, and references to a point earlier than the CSECT start will be flagged by the Translator.

3.7.7 Special Properties of the First CSECT in a module

(1) Its tentative starting location may be specified absolutely by employing the START statement instead of CSECT.

(2) In the absence of further information, all literals in the module will be assembled so as to occur at the end of this CSECT (see LTORG statement). This will require the programmer to maintain the appropriate base register for the CSECT during the remainder of the module.

3.7.8 Limits on Number of Control Statements

The Translator's limits are: not more than 100 ENTRY Statements; not more than 255 distinct CSECT+DSECT+COM+EXTRN+V-constant statements; not more than 32 distinct CSECT+DSECT+COM statements per module translated.

The Composer's limits will depend upon the operating regime and the amount of store available to Composer.

4. Data formats

4.1 INTRODUCTION

The basic unit of information in the 4-50 is a byte, which is the smallest addressable unit. A byte consists of eight information bits and one parity bit. The parity bit ensures the accuracy of all bytes accessed by the processor. Odd parity is used in both processors.

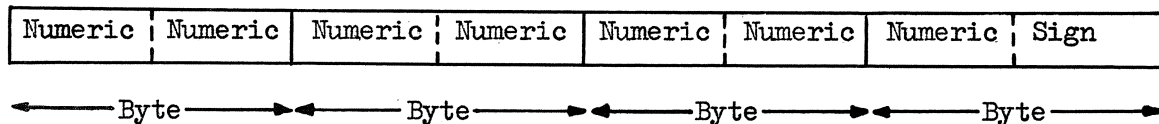
The internal code representation is either the Extended Binary-Coded-Decimal Interchange Code (EBCDIC) or the American Standard Code for Information Interchange (ASCII also referred to as ISO code) as specified by program.

There are several distinct formats for data in the main store. Further explanation of each format appears in the instruction sections of this manual.

The formats for data in System 4-50 are packed decimal, zoned decimal, fixed and floating point, and logical.

4.2 PACKED DECIMAL

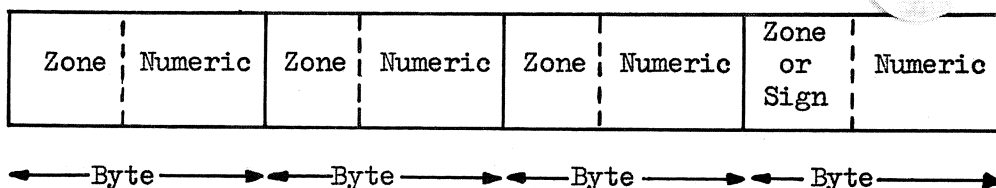
In packed-decimal format, one byte represents two numeric digits. All decimal numerics must be packed because all decimal arithmetic functions operate on this format. The numerals, zero to nine, are coded $(0000)_2$ to $(1001)_2$ and are the only legitimate digits in packed-decimal format.



The rightmost half-byte (four bits) of a field represents the sign. The only EBCDIC code that is machine-generated to represent the plus (+) sign of a positive field is $(1100)_2$. The code $(1101)_2$ is the only EBCDIC code that is machine-generated to represent the minus (-) sign of a negative field. It should be noted, however, that the codes $(1010)_2$, $(1110)_2$ and $(1111)_2$ representing plus (+) signs and the code $(1011)_2$ representing a minus (-) sign are accepted by the machine. (This variety permits the processor to handle other than EBCDIC code). Nevertheless, if an arithmetic operation is performed on a field, the sign of the result will be in EBCDIC code.

4.3 ZONED DECIMAL

In zoned format, one byte represents one alphanumeric digit. Alphanumeric data must be in zoned format. The rightmost half-byte (four bits) is the number and the leftmost half-byte (four bits) is the zone.

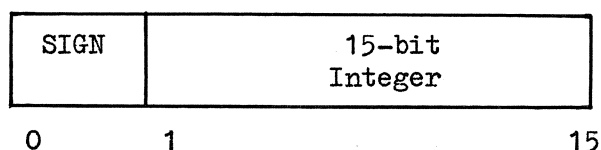


When changing from packed decimal format to zoned format, the code $(1111)_2$ is generated in the zone portion of a field. The zone portion of the rightmost byte of a numeric field is the sign of the field.

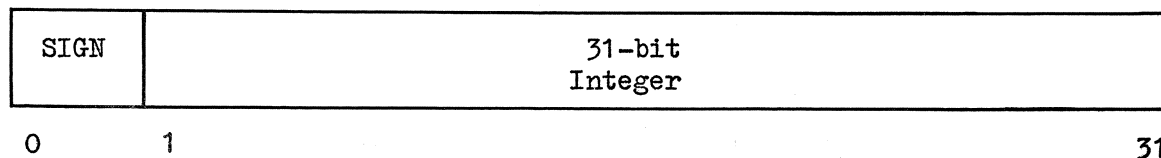
4.4 FIXED POINT

A fixed-length format of a one-bit sign followed by the integer field makes up fixed-point numbers. In one of the general registers, the number is a 31-bit integer field. The complete 32-bit register is occupied by the fixed-point quantity and sign. A 64-bit operand, with a 63-bit integer field, is used by some shift, multiply, and divide instructions. A pair of adjacent registers, addressed by the even address of the leftmost register, contains these longer operands. The sign-bit of the rightmost register becomes part of the integer field. The same register can be specified for both operands in register-to-register operations (except for the Divide instructions). In the main store, fixed-point operands are in either a 32-bit word or a 16-bit halfword. The integer fields are then either 31 bits or 15 bits. Radix conversion operations always use a 64-bit decimal field. Integral storage boundaries for these units of data must be observed. Halfword, full-word, or double-word operands are addressed with one, two, or three low-order address bits set to zero. Half-word operands are extended to full words when they are fetched from the main store and used as a full-word operand.

Halfword Fixed-Point Number



Full-word Fixed-Point Number



4.5 FLOATING POINT

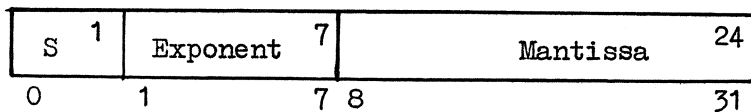
Floating-point numbers are fixed in length and are either full-word (short) or double-word (long) in format.

The first bit in both formats is the sign of the mantissa. A 1 bit represents a minus sign and a 0 bit represents a plus sign. The next seven bits represent the exponent. The mantissa contains six hexadecimal

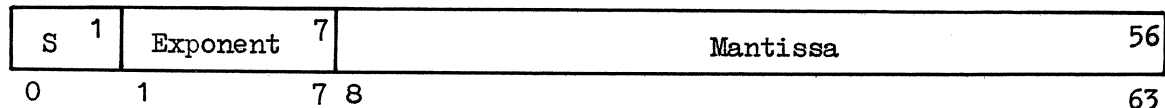
digits (short floating-point number) or 14 hexadecimal digits (long floating-point number).

The short format allows for faster processing and uses less storage. Because floating-point registers are 64 bits long, the rightmost 32 bits are ignored when dealing with short operands. When the short format is specified, all operands and the result are 32 bits long. When using the long format, which provides greater precision, all operands are 64 bits long and require the full register.

Short Floating-Point Number



Long Floating-Point Number



4.6 LOGICAL

Data in general registers usually involves the entire 32 bits. There is no distinction made between sign and numeric bits. In some operations, only the least significant eight bits of the general register are involved, and in another case, the least significant 24 bits are involved. In addition, there are some shift operations in which an even/odd numbered pair of general registers is involved.

The storage data in store-to-register operations resides in either a 32-bit word or an eight-bit byte. A word must be oriented on word boundaries (that is, the address of the 32-bit word must have the two rightmost bits zero).

The storage data in store-to-store operations have a variable length format and can have a field size of up to 256 bytes starting at any byte location. Processing is from left to right.

Instructions that specify an operand that is part of the immediate instruction being executed are restricted to a field size of one eight-bit byte.

The Translate and Test and the Edit and Mark instructions imply the use of General Register 1*. An address of 24 bits may be placed in this register during the execution of these instructions. The Translate and Test instruction also implies the use of General Register 2 where an insertion of an eight-bit function byte may be placed during the execution of the instruction.

*When these instructions are executed in P3, General Registers 13 and 14 are used; in P4, General Registers 9 and 10 are used.

Overlapping of fields in store-to-store operations may or may not affect the operands of the various instructions. The execution of some logical instructions does not change the operands. Other instructions, such as Move, Edit, and Translate, replace one operand with new data, and this data is handled one eight-bit byte at a time. This procedure enables the user to determine the effect overlapping fields have on the execution of the instruction. Unpredictable results can occur while overlapping fields are being edited. Overlapping fields are valid for all other operations.

5. Instruction formats

5.1 INTRODUCTION

The five basic instruction formats express, in general terms, the operation to be performed as follows:

RR = register-to-register

RX = register-to-indexed main store

RS = register-to-main store

SI = main store and immediate operand operation

SS = main store to main store

The detailed layout of each class is shown below.

0		7		15		31		47					
1st halfword locn. 1st byte		2nd byte		2nd halfword location		3rd halfword location							
0		7		11		15				RR Format			
Op. code		R1		R2									
0		7		11		15		19		31		RX Format	
Op. code		R1		X2		B2		D2					
0		7		11		15		19		31		RS Format	
Op. code		R1		R3		B2		D2					
0		7		15		19		31				SI Format	
Op. code		I2		B1		D1							
0		7		11		15		19		31		SS Format	
Op. code		L1		L2		B1		D1		B2			D2
				L									

Notation

R1, R2, R3	4-bit register address (0→15)
X2	4-bit index register address (0→15)
B1, B2	4-bit base register addresses (0→15)
D1, D2	12-bit displacements (0→4095)
I2	8-bit immediate operand
L1, L2	4-bit length specifications (0→15)
L	8-bit length specification (0→255).

5.2 RR FORMAT

The contents of the general register specified by R1 is the first operand. The contents of the general register specified by R2 is the second operand. In floating-point operations, R1 designates the address of the floating-point register that contains the first operand. R2 designates the floating-point register that contains the second operand. The first and second operands can be the same and are designated by identical R1 and R2 addresses.

5.3 RX FORMAT

The contents of the general register specified by R1 is the first operand. To obtain the address of the second operand, the contents of the general registers specified by X2 and B2 are added to the D2 field. In floating-point operations, R1 designates the floating-point register that contains the first operand.

5.4 RS FORMAT

The RS format is used by shift instructions, branching instructions, and load/store multiple instructions.

Shift Instructions

The contents of the general register specified by R1 is the first operand. The contents of the general register specified by B2 are added to the D2 field. The sum specifies the number of bits of shifting to be done by the shift operation. The R3 field is ignored.

Branching Instructions

The contents of the general register specified by R1 is the first operand. The contents of the general register specified by B2 are added to the D2 field to obtain the branch address. The contents of the general register specified by R3 is the third operand.

Load/Store Multiple Instructions

The R1 and R3 fields specify the general register boundaries. The contents of the general register specified by B2 are added to the D2 field to obtain the main store address of the second operand.

5.5 SI FORMAT

The contents of the general register specified by B1 are added to the contents of the D1 field to obtain the address of the first operand. The second operand is the immediate eight-bit byte in the I2 field of the instruction. (An immediate operand is a piece of data which is actually stored with an instruction and used by that instruction, as opposed to an operand which is defined by means of an address).

5.6 SS FORMAT

The contents of general register specified by B1 are added to the contents of the D1 field to obtain the address of the leftmost byte of the first operand. The L1 field specifies the number of additional bytes in the operand that are to the right of the first operand address. To obtain the second operand address, the contents of the general register specified by B2 are added to the contents of the D2 field. The L2 field specifies the number of additional bytes in the operand that are to the right of the second operand address.

The L field specifies the number of additional bytes that are to the right of the first and the second operand addresses. An operand length in machine form is thus one less than the true value. This is to utilise every possible bit configuration, allowing real values in range 1-16 for L1, L2, and 1-256 for L.

NOTES

1. A zero appearing in the X2, B1 or B2 fields indicates an absence of the corresponding address or shift-amount component. An instruction can specify the same general register both for address modification and for operand location.
2. Address modification is completed before the execution of an operation.
3. The results replace the first operand (except in Store Character instruction), where the result replaces the second operand.
4. A variable-length result is never stored outside the field specified by the address and length.
5. The contents of all registers and main store locations not specified by an instruction remain unchanged except for the Edit and Mark instruction and the Translate and Test instructions. These instructions automatically use certain general registers.

6. Executable (machine) instructions

6.1 INTRODUCTION

The machine instructions for SYSTEM 4-50 can be divided into the following categories:-

- a) Fixed point instructions
- b) Decimal arithmetic instructions
- c) Logical instructions
- d) Branching instructions
- e) Privileged instructions
- f) Processor state control instructions
- g) Floating point instructions

The instructions in each group are listed below. Following this is a list of the symbols used in the detailed descriptions of each instruction which follow later in this chapter. Each detailed description begins with the name of the instruction at the top of the page and the mnemonic in the top corner. The rest of the description is divided into the following headings:-

General Description - a description of what the action does.

Formats - the Machine and Usercode formats of the instruction. The figures associated with each part of the operand in the machine format represent the number of bits in that part. Those positions marked by a diagonal line must be set to zero.

Condition Code - the possible condition code settings and their meanings after the instruction has been executed.

Timing - the time taken to execute the instruction.

Interrupt Action - the interrupt conditions which can occur with any particular instruction. The last sub-section of each group gives details of the interrupt conditions in that particular group.

Notes - any comments or special points about the instruction.

		<u>Chapter</u>
<u>Fixed-Point Instructions</u>		6.2
A	Add Word	6.2.8
AH	Add Halfword	6.2.9
AL	Add Logical	6.2.10
AIR	Add Logical	6.2.10
AR	Add Word	6.2.8
C	Compare Word	6.2.15
CH	Compare Halfword	6.2.14
CR	Compare Word	6.2.15
CVB	Convert to Binary	6.2.19
CVD	Convert to Decimal	6.2.20
D	Divide	6.2.18

		<u>Chapter</u>
DR	Divide	6.2.18
L	Load Word	6.2.1
LCR	Load Complement	6.2.4
LH	Load Halfword	6.2.2
LM	Load Multiple	6.2.7
LNR	Load Negative	6.2.6
LPR	Load Positive	6.2.5
LR	Load Word	6.2.1
LTR	Load and Test	6.2.3
M	Multiply Word	6.2.16
MH	Multiply Halfword	6.2.17
MR	Multiply Word	6.2.16
S	Subtract Word	6.2.11
SH	Subtract Halfword	6.2.12
SL	Subtract Logical	6.2.13
SLA	Shift Left Single	6.2.24
SLDA	Shift Left Double	6.2.26
SLR	Subtract Logical	6.2.13
SR	Subtract Word	6.2.11
SRA	Shift Right Single	6.2.25
SRDA	Shift Right Double	6.2.27
ST	Store Word	6.2.21
STH	Store Halfword	6.2.22
STM	Store Multiple	6.2.23

Decimal Arithmetic Instructions

6.3

AP	Add Packed Decimal	6.3.1
CP	Compare Packed Decimal	6.3.4
DP	Divide Packed Decimal	6.3.6
MP	Multiply Packed Decimal	6.3.5
MVO	Move With Offset	6.3.9
PACK	Pack	6.3.7
SP	Subtract Packed Decimal	6.3.2
UNPK	Unpack	6.3.8
ZAP	Zero and Add Packed	6.3.3

Logical Instructions

6.4

CL	Compare Logical	6.4.4
CLC	Compare Logical	6.4.4
CLI	Compare Logical	6.4.4
CLR	Compare Logical	6.4.4
ED	Edit	6.4.14
EDMK	Edit and Mark	6.4.15
IC	Insert Character	6.4.9
LA	Load Address	6.4.11
MVC	Move	6.4.1
MVI	Move	6.4.1
MVN	Move Numerics	6.4.2
MVZ	Move Zones	6.4.3
N	AND	6.4.5
NC	AND	6.4.5

	<u>Chapter</u>
NI AND	6.4.5
NR AND	6.4.5
O OR	6.4.6
OC OR	6.4.6
OI OR	6.4.6
OR OR	6.4.6
SLDL Shift Left Double Logical	6.4.18
SLL Shift Left Single Logical	6.4.16
SRDL Shift Right Double Logical	6.4.19
SRL Shift Right Single Logical	6.4.17
STC Store Character	6.4.10
TM Test Under Mask	6.4.8
TR Translate	6.4.12
TRT Translate and Test	6.4.13
X Exclusive OR	6.4.7
XC Exclusive OR	6.4.7
XI Exclusive OR	6.4.7
XR Exclusive OR	6.4.7
<u>Branching Instructions</u>	6.5
BAL Branch and Link	6.5.2
BALR Branch and Link Register	6.5.2
BC Branch on Condition	6.5.1
BCR Branch on Condition Register	6.5.1
BCT Branch on Count	6.5.3
BCTR Branch on Count Register	6.5.3
BXH Branch on Index High	6.5.4
BXLE Branch on Index Low or Equal	6.5.5
EX Execute	6.5.6
<u>Privileged Instructions (Standard)</u>	6.6
DIG Diagnose	6.6.6
IDL Idle	6.6.5
LSP Load Scratch Pad	6.6.2
PC Program Control	6.6.4
SSP Store Scratch Pad	6.6.3
<u>Privileged Instructions (Optional)</u>	6.6
ISK Insert Storage Key	6.6.11
RDD Read Direct	6.6.14
SSK Set Storage Key	6.6.12
WRD Write Direct	6.6.13
<u>Privileged Instructions (Input/Output)</u>	6.6
CKC Check Channel	6.6.10
HDV Halt Device	6.6.8
SDV Start Device	6.6.7
TDV Test Device	6.6.9
<u>Processor State Control Instructions</u>	6.7
SPM Set Program Mask	6.7.4
SVC Supervisor Call	6.7.3

<u>Floating Point Instructions</u>		<u>Chapter</u>
		6.8
AD	Add Normalised (Long)	6.8.5
ADR	Add Normalised (Long)	6.8.5
AE	Add Normalised (Short)	6.8.5
AER	Add Normalised (Short)	6.8.5
AU	Add Unnormalised (Short)	6.8.6
AUR	Add Unnormalised (Short)	6.8.6
AW	Add Unnormalised (Long)	6.8.6
AWR	Add Unnormalised (Long)	6.8.6
CD	Compare (Long)	6.8.9
CDR	Compare (Long)	6.8.9
CE	Compare (Short)	6.8.9
CER	Compare (Short)	6.8.9
DD	Divide (Long)	6.8.13
DDR	Divide (Long)	6.8.13
DE	Divide (Short)	6.8.13
DER	Divide (Short)	6.8.13
HDR	Halve (Long)	6.8.10
HER	Halve (Short)	6.8.10
LCDR	Load Complement (Long)	6.8.2
LCER	Load Complement (Short)	6.8.2
LD	Load (Long)	6.8.14
LDR	Load (Long)	6.8.14
LE	Load (Short)	6.8.14
LER	Load (Short)	6.8.14
LNDR	Load Negative (Long)	6.8.4
LNER	Load Negative (Short)	6.8.4
LPDR	Load Positive (Long)	6.8.3
LPER	Load Positive (Short)	6.8.3
LTDR	Load and Test (Long)	6.8.1
LTER	Load and Test (Short)	6.8.1
MD	Multiply (Long)	6.8.12
MDR	Multiply (Long)	6.8.12
ME	Multiply (Short)	6.8.12
MER	Multiply (Short)	6.8.12
SD	Subtract Normalised (Long)	6.8.7
SDR	Subtract Normalised (Long)	6.8.7
SE	Subtract Normalised (Short)	6.8.7
SER	Subtract Normalised (Short)	6.8.7
STD	Store (Long)	6.8.11
STE	Store (Short)	6.8.11
SU	Subtract Unnormalised (Short)	6.8.8
SUR	Subtract Unnormalised (Short)	6.8.8
SW	Subtract Unnormalised (Long)	6.8.8
SWR	Subtract Unnormalised (Long)	6.8.8

Symbols used in Usercode instruction formats.

- D - Arithmetic expression, specifying the address of the leftmost byte of the operand, relative to the base address in the register specified by B. In the range 0 to 4095.
- B - Arithmetic expression, specifying the register that will provide the base address of the operand. In the range 0 to 15.
- L - Arithmetic expression, specifying the length of the operand in bytes. If a length is allowed for both operands in an instruction then it is in the range 1 - 16; if a length is allowed only for the first operand in an instruction then it is in the range 1 to 256.
- X - Arithmetic expression, specifying the register whose contents are used for address modification in RX instructions. In the range 0 to 15.
- R - Arithmetic expression, a self defining term or a symbol, specifying a register. In the range 0 to 15.
- I - Arithmetic expression or a self defining term, specifying an immediate operand.
- M - Arithmetic expression or a self defining term, specifying a mask.
- S - an absolute or relocatable expression.
- T(U) - Arithmetic expression (arithmetic expression) specifying the trunk and unit number of a device.

DXB format

The allowable forms for an operand which has DXB format are:-

D(X,B)
S(X)
S

DLB format

The allowable forms for an operand which has DLB format are:-

D(L,B)
D(,B)
S(L)
S

DB format

The allowable forms for an operand which has DB format are:-

D(B)
S

Symbols used in Timing

- N1 - number of bytes in first operand
- N2 - number of bytes in second operand
- N - number of bytes in operand
- B - total number of bytes processed
- R - number of registers loaded or stored
- T - number of 4-bit shifts
- V - total number of bits shifted
- F - number of fill characters inserted
- K - number of control characters in pattern field
- D - number of digits in source field.

6.2 FIXED POINT INSTRUCTIONS

6.2.1 LOAD WORD

General Description

The operand specified by the second address is loaded into the general register specified by the first address.

Machine Formats

(RR)

OP	8	R1	4	R2	4
----	---	----	---	----	---

$OP = (18)_{16}$

(RX)

OP	8	R1	4	X2	4	B2	4	D2	12
----	---	----	---	----	---	----	---	----	----

$OP = (58)_{16}$

Usercode Formats

(RR) LR R, R
(RX) L R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 LR 2.88 ; L 8.88.
4-70 LR 0.90 ; L 1.80.

Interrupt Action

Address error:

Addressing (RX format).
Specification (RX format).

Note

The operand in the register or main store location specified by the second address remains unchanged.

LH

6.2.2 LOAD HALFWORD

General Description

The halfword operand in the main store specified by the second address is loaded into the general register specified by the first address.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(48) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

LH R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 7.92

4-70 : 2.2

Interrupt Action

Address error:

Addressing.

Specification.

Notes

1. When the halfword (second operand) is fetched from the main store, it is expanded to a full word by propagating the sign-bit value through the 16 high-order positions of the receiving register.
2. The operand specified by the second address is unaltered.

6.2.3 LOAD AND TEST

General Description

The operand in the register specified by the second address (R2) is loaded into the general register specified by the first address (R1). The condition code is determined by the magnitude and the sign of the loaded operand.

Machine Format

(RR)	OP	8	R1	4	R2	4	OP=(12) ₁₆
------	----	---	----	---	----	---	-----------------------

Usercode Format

LTR R, R

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - not used.

Timing (μsecs)

4-50 : 5.28
4-70 : 0.9

Interrupt Action

None.

Notes

1. The same register can be specified for both R1 and R2. If this is done, the operation is equivalent to a test with no data movement.
2. The operand specified by the second address is unaltered.

LCR

6.2.4 LOAD COMPLEMENT

General Description

The two's complement of the operand in the register specified by the second address is loaded into the general register specified by the first address. The condition code is determined by the magnitude and the sign of the loaded operand.

Machine Format

(RR)	OP	8	R1	4	R2	4	OP=(13) ₁₆
------	----	---	----	---	----	---	-----------------------

Usercode Format

LCR R, R

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : 5.28
4-70 : 0.9

Interrupt Action

Fixed-point overflow.

Notes

1. Zero operands remain constant and unchanged under complementation.
2. A fixed-point overflow interrupt occurs when the maximum negative number is complemented.
3. The operand specified by the second address is unaltered.

6.2.5 LOAD POSITIVEGeneral Description

The operand in the register specified by the second address is made positive, if negative, and loaded into the general register specified by the first address. In loading the absolute value of the operand, negative numbers are complemented and positive numbers remain unaltered. The magnitude of the absolute value determines the condition code.

Machine Format

(RR)	OP	8	R1	4	R2	4	OP=(10) ₁₆
------	----	---	----	---	----	---	-----------------------

Usercode Format

LPR R, R

Condition Code

- 0 - result is zero.
- 1 - not used.
- 2 - result greater than zero.
- 3 - overflow on complement.

Timing (μsecs)

4-50 : 6.24

4-70 : 0.9

Interrupt Action

Fixed-point overflow.

Notes

1. A fixed-point overflow interrupt exists if a maximum negative number is complemented.
2. The operand specified by the second address is unaltered.

LNR

6.2.6 LOAD NEGATIVE

General Description

The two's complement of the operand in the register specified by the second address is loaded into the general register specified by the first address. In loading the operand value, positive numbers are complemented and negative numbers remain unaltered. The magnitude of the loaded value determines the condition code setting.

Machine Format

(RR)	OP	8	R1	4	R2	4	OP=(11) ₁₆
------	----	---	----	---	----	---	-----------------------

Usercode Format

(RR) : LNR R, R

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - not used.
- 3 - not used.

Timing (μsecs)

4-50 6.24
4-70 0.9

Interrupt Action

None.

Notes

1. A zero operand is not altered and retains a positive sign.
2. The operand specified by the second address is unaltered.

6.2.7 LOAD MULTIPLEGeneral Description

The set of general registers, beginning with the register specified by the first address (R1) and ending with the register specified by the third address (R3), is loaded with operands from the main store. The second address specifies the main store location of the first word to be loaded. Loading of the general registers continues in the ascending order of their addresses beginning with the register specified by R1. As many words as needed are fetched from the main store location specified, continuing up to, and including, the register specified by R3.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(98) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

LM R, R, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 6.0 + 2.88R

4-70 : 2.0 + 0.54R

Interrupt Action

Address error:

Addressing.

Specification.

Notes

1. If R1 and R3 specify the same register, only one word is loaded.
2. If the register specified by R3 is less than the register specified by R1, wrap-around occurs from register 15 to 0.
3. The operands specified by the second address are unaltered.

AR,A

6.2.8 ADD WORD

General Description

The operand specified by the first address is added to the operand specified by the second address and the sum is placed in the general register specified by the first address. The magnitude and the sign of the sum determine the condition code setting.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(1A) ₁₆				
OP	8	R1	4	R2	4							
(RX)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(5A) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR) AR R, R
(RX) A R, DXB

Condition Code

- 0 - sum is zero.
- 1 - sum is less than zero.
- 2 - sum is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : AR 5.28
A 8.88
4-70 : AR 1.1
A 2.1

Interrupt Action

Fixed-point overflow.

Address error:

Addressing (RX format).
Specification (RX format).

Notes

1. All 32 bits of both operands participate in the addition. If the high-order numeric bit position of the result and the carries out of the sign bit position disagree, an overflow condition exists. The overflow does not alter the sign bit created by the carries.
2. A negative overflow results in a positive sum and a positive overflow results in a negative sum with overflow bits being lost.

3. A zero result is always positive.
4. The operand specified by the second address is unaltered.

6.2.9 ADD HALFWORDGeneral Description

The halfword operand specified by the second address is added to the operand specified by the first address and the sum is placed into the register specified by the first address. The sign and the magnitude of the sum determine the condition code setting.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(4A) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

AH R, DXB

Condition Code

- 0 - sum is zero.
- 1 - sum is less than zero.
- 2 - sum is greater than zero.
- 3 - overflow.

Timing (μsecs)

- 4-50 : 7.92
- 4-70 : 2.4

Interrupt Action

Fixed-point overflow.

Address error:

Addressing.
Specification.

Notes

1. The halfword in the main store location specified by the second address is expanded to full-word length prior to the addition by propagating the sign bit value through the high-order 16 positions. The addition is completed by adding all 32 bits of both operands.
2. An overflow exists if the high-order numeric result bit and the carry out of the sign-bit position disagree. The sign is not corrected after overflow occurs. A negative overflow results in a positive sum and a positive overflow results in a negative sum with the overflow bits being lost.
3. The operand specified by the second address is unaltered.

6.2.10 ADD LOGICALGeneral Description

The operand specified by the second address is logically added (32-bit unsigned) to the operand specified by the first address. The sum is placed in the general register specified by the first address. The condition code is determined by the relation of the sum to a zero number and the occurrence of a carry out of the sign bit position. An overflow on such carries is not recognized and does not set an interrupt condition.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(1E) ₁₆				
OP	8	R1	4	R2	4							
(RX)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(5E) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR) ALR R, R
(RX) AL R, DXB

Condition Code

0 - sum is zero and no carry.
1 - sum is not zero and no carry.
2 - sum is zero with a carry.
3 - sum is not zero with a carry.

Timing (μsecs)

4-50 : ALR 4.80
AL 8.40
4-70 : ALR 1.1
AL 2.1

Interrupt Action

Address error:

Addressing (RX format).
Specification (RX format).

Notes

1. All 32 bits of the operands participate in the logical addition.
2. The operand specified by the second address is unaltered.

2. The difference between a maximum negative number and another maximum negative number is zero with no overflow.
3. When the same register is specified for R1 and R2, the operation is equivalent to clearing R1 to zero.
4. The operand specified by the second address is unaltered.

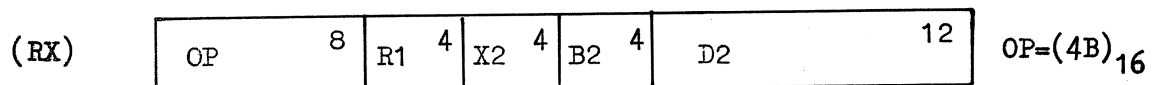
SH

6.2.12 SUBTRACT HALFWORD

General Description

The halfword operand specified by the second address is expanded and subtracted from the operand specified by the first address (R1). The difference is placed in the general register specified by R1. The sign and the magnitude of the difference determine the condition code setting.

Machine Format



Usercode Format

SH R, DXB

Condition Code

- 0 - difference is zero.
- 1 - difference is less than zero.
- 2 - difference is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : 7.92
4-70 : 2.4

Interrupt Action

Fixed-point overflow.

Address error:

Addressing.
Specification.

Notes

1. The halfword in the main store location specified by the second address is expanded to full-word length by propagating the sign bit value through the 16 high-order positions.
2. The subtraction is completed by adding the one's complement of the second operand and a one in the low-order position of the first operand. All 32 bits are involved in the operation.
3. An overflow exists if the high-order numeric result bit and the carry out of the sign bit position disagree.
4. The operand specified by the second address is unaltered.

6.2.13 SUBTRACT LOGICALGeneral Description

The operand specified by the second address is logically subtracted (32-bit unsigned) from the operand specified by the first address. The difference is placed in the general register specified by the first address. The condition code is determined by the relation of the sum to a zero number and the occurrence of a carry out of the sign bit position. An overflow on such carries is not recognized and does not set an interrupt condition.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(1F) ₁₆				
OP	8	R1	4	R2	4							
(RX)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(5F) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR) SLR R, R
(RX) SL R, DXB

Condition Code

- 0 - not used.
- 1 - difference is not zero and no carry.
- 2 - difference is zero with a carry.
- 3 - difference is not zero with a carry.

Timing (μsecs)

4-50 : SLR 4.80
 SL 8.40
4-70 : SLR 1.1
 SL 2.1

Interrupt Action

Address error:

Addressing (RX format).
Specification (RX format).

Notes

1. A zero difference cannot occur without a carry out of the sign position
2. Logical subtraction is accomplished by adding the one's complement of the second operand and a one in the low-order position of the first operand.

SLR,SL

3. All 32 bits of the operands participate in the logical subtraction without change to the resulting sign bit.
4. The operand specified by the second address is unaltered.

6.2.14 COMPARE HALFWORDGeneral Description

The operand specified by the first address is compared with the halfword operand expanded to a full word, specified by the second address. Both operands remain unaltered. The result of the comparison determines the condition code setting.

Machine Format

(RX)	OP ⁸	R1 ⁴	X2 ⁴	B2 ⁴	D2 ¹²	OP=(49) ₁₆
------	-----------------	-----------------	-----------------	-----------------	------------------	-----------------------

Usercode Format

CH R, DXB

Condition Code

- 0 - operands are equal.
- 1 - the operand specified by the first address is low.
- 2 - the operand specified by the first address is high.
- 3 - not used.

Timing (μsecs)

4-50 : 7.44
4-70 : 2.4

Interrupt Action

Address error:

Addressing.
Specification.

Notes

1. The halfword in storage specified by the second address is expanded to full-word length by propagating the sign bit value through the 16 high-order positions.
2. Both operands are considered as 32-bit signed integers and the comparison is algebraic.

6.2.15 COMPARE WORDGeneral Description

The operand specified by the first address is compared with the operand specified by the second address. Both operands remain unaltered. The result of the comparison determines the condition code setting.

Machine Formats

OP	8	R1	4	R2	4
----	---	----	---	----	---

OP=(19)₁₆

OP	8	R1	4	X2	4	B2	4	D2	12
----	---	----	---	----	---	----	---	----	----

OP=(59)₁₆

Usercode Formats

(RR) CR R, R
(RX) C R, DXB

Condition Code

- 8 0 - operands are equal.
4 1 - the operand specified by the first address is low.
2 2 - the operand specified by the first address is high.
1 3 - not used.

Timing (μsecs)

4-50 : CR 4.80
C 8.40
4-70 : CR 1.1
C 2.1

Interrupt Action

Address error:

Addressing (RX format).
Specification (RX format).

Note

Both operands are considered as 32-bit signed integers and the comparison is algebraic.

6.2.16 MULTIPLY WORDGeneral Description

The operand (multiplicand) specified by the first address is multiplied by the operand (multiplier) specified by the second address. The double-length product is loaded into the register specified by the first address, which must be an even number, and the next odd-numbered register.

Machine Formats

(RR)	OP	8	R1	4	R2	4	OP=(1C) ₁₆				
(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(5C) ₁₆

Usercode Formats

(RR) MR R, R
(RX) M R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : MR 62.52
M 65.64
4-70 : MR 5.8
M 6.6

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Notes

1. The first address must always refer to the even-numbered register of an even/odd pair. The multiplicand is taken from the odd-numbered register of the pair. The original contents of the even-numbered register, which is replaced by the product, is ignored. An overflow cannot occur.
2. Only when two maximum negative numbers are multiplied does the product exceed 62 significant bits. This product produces 63 significant bits.
3. In two's-complement notation, the sign bit is propagated right, up to the first significant product bit.

4. The sign of the product is determined algebraically. A zero result is always positive.
5. The least significant digit of the product goes into the odd-numbered register.
6. The operand specified by the second address (multiplier) is unaltered except when the first and second addresses specify the same (even numbered) register. In this case the multiplier is taken from the even register, the multiplicand is taken from the odd register and the product is placed into the even/odd pair.

6.2.17 MULTIPLY HALFWORDGeneral Description

The operand (multiplicand) specified by the first address is multiplied by the halfword operand (multiplier) specified by the second address. The product of the operands replaces the contents of the register specified by the first address.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(4C) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

MH R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 35.40

4-70 : 6.6

Interrupt Action

Address error:

Addressing.

Specification.

Notes

1. The halfword operand in the main store is expanded to a full word before multiplication by propagating the sign bit value through the 16 high-order positions. Both operands are considered as 32-bit signed integers. The multiplicand is replaced by the low order 32 bits of the product.
2. The product usually occupies 46 bits of significance except when both operands are maximum negative numbers and occupy 47 bits.
3. The bits to the left of the 32 low-order bits of the product are not tested for significance. No overflow indication is given. Since the bits to the left of the low-order 32 are ignored, the sign of the result may differ from the true sign of the product, if the product exceeds 32 bits.
4. The operand specified by the second address is unaltered.
5. A zero product is always positive.

DR,D

6.2.18 DIVIDE

General Description

The double-word operand (dividend) specified by the first address is divided by the operand (divisor) specified by the second address. The quotient and remainder replace the double-word operand in the registers specified by the first address. The register specified by the first address must be the even-numbered register of an even/odd pair.

Machine Formats

(RR)

OP	8	R1	4	R2	4
----	---	----	---	----	---

 OP=(1D)₁₆

(RX)

OP	8	R1	4	X2	4	B2	4	D2	12
----	---	----	---	----	---	----	---	----	----

 OP=(5D)₁₆

Usercode Formats

(RR) DR R, R
(RX) D R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : DR 90.81
 D 94.89
4-70 : DR 10.8
 D 11.6

Interrupt Action

Address error:

Addressing (RX format).
Specification

Divide Error.

Notes

1. The dividend, a 64-bit signed integer, is replaced by a 32-bit signed quotient and a 32-bit signed remainder; the remainder is placed in the even-numbered register and the quotient is placed in the odd-numbered register. The divisor is a 32-bit signed integer and is unaltered.
2. A divide error interrupt occurs when the magnitude of the dividend to the divisor is such that the quotient cannot be expressed by a 32-bit signed integer. (The divisor must be greater than twice the absolute value of the first word of the dividend).

3. The sign of the quotient is determined algebraically except that a zero quotient, as a zero remainder, is always positive.
4. The remainder has the same sign as the dividend.

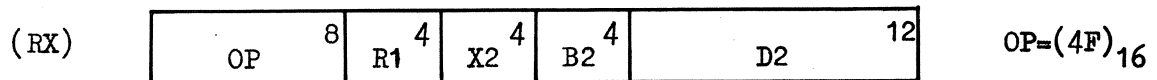
CVB

6.2.19 CONVERT TO BINARY

General Description

The radix of the double-word operand in the main store specified by the second address is converted from decimal to binary notation and loaded into the general register specified by the first address. The operand in the main store is treated as a right-justified signed integer before and after the conversion.

Machine Format



Usercode Format

CVB R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 91.20

4-70 : 11.1

Interrupt Action

Address error:

Addressing.
Specification.

Data error.

Divide error.

Notes

1. The double-word operand in main store (15 digits plus sign) must be in the packed decimal format. The operand is checked for valid sign and digit codes. The sign representation depends on the current decimal code (ISO or EBCDIC).
2. The maximum decimal number that can be converted and still be contained in a 32-bit register is (2,147,483,647)₁₀ positive and (2,147,483,648)₁₀ negative. A larger decimal number causes a divide error interrupt.
3. Negative decimal zero is converted to positive binary zero.
4. The operand specified by the second address remains unaltered in main store.

6.2.20 CONVERT TO DECIMALGeneral Description

The radix of the operand specified by the first address is converted from binary to decimal notation and stored at the double-word main store area specified by the second address. The operand is treated as a right-justified signed integer before and after the conversion.

Machine Format

(RX)	8	4	4	4	12	
	OP	R1	X2	B2	D2	OP=(4E) ₁₆

Usercode Format

CVD R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 68.88 to 91.92
4-70 : 14.0

Interrupt Action

Address error:

Addressing.

Specification.

Protection.

Notes

1. The result is placed in the double-word main store location in the packed decimal format of 15 digits plus sign.
2. The low-order four bits of the result are the sign which is generated according to the current decimal code, EBCDIC or ISO.
3. The maximum binary number (32-bit signed integer) that can be converted is (2,147,483,647)₁₀ positive and (2,147,483,648)₁₀ negative. No overflow can occur.

ST

6.2.21 STORE WORD

General Description

The operand in the general register specified by the first address is stored in the main store location specified by the second address.

Machine Format

(RX)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(50) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Format

ST R, DXB

Condition Code

Unchanged.

Timing (µsecs)

4-50 : 7.44

4-70 : 1.7

Interrupt Action

Address error:

Addressing.

Specification.

Protection.

Notes

1. The complete contents (32 bits) of the general register specified by the first address are placed unaltered in main store.
2. The operand specified by the first address is unaltered.

6.2.22 STORE HALFWORD

General Description

The rightmost half (16 bits) of the operand in the general register specified by the first address is stored unaltered in the halfword main store location specified by the second address.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(40) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

STH R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 5.04
4-70 : 3.0

Interrupt Action

Address error:

Addressing.
Specification.
Protection.

Notes

1. The 16 high-order bits of the operand specified by the first address field are ignored by the operation.
2. The operand specified by the first address is unaltered.

STM

6.2.23 STORE MULTIPLE

General Description

The operands in the set of general registers, beginning with the register specified by the first address (R1) and ending with the register specified by the third address (R3), are stored in main store locations starting with the location specified by the second address. The second address refers to the main store location where the first operand (word) is to be stored. Storing of the operands continues in the ascending order of the register number specified by R1, up to and including R3, storing as many words as indicated in the main store locations that immediately follow the initial operand.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(90) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

STM R, R, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 6.0 + 2.88R

4-70 : 1.4 + 0.54R

Interrupt Action

Address error:

Addressing.

Specification.

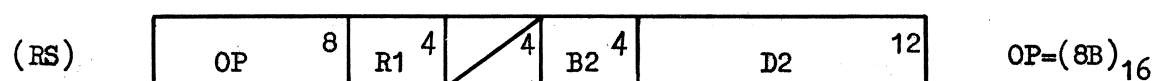
Protection.

Notes

1. If the same register is specified for R1 and R3 only one word is stored.
2. If R3 is less than R1, the register addresses wrap around from 15 to 0. For instance, all registers can be stored by making R3 one less than R1.
3. The operands in the set of registers designated are unaltered.

6.2.24 SHIFT LEFT SINGLEGeneral Description

The integer portion of the operand in the general register specified by the first address is shifted left the number of positions specified by the second address. The second address is used as a count and not to address data. The low-order six bits of the second address constitute the count. The remaining bits are ignored.

Machine FormatUsercode Format

SLA R, DB

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - overflow.

Timing (μsecs)

- 4-50 : Under 16 bits : $10.08 + 0.48(V)$
- 16-31 bits : $13.20 + 0.48(V-16)$
- 32-47 bits : $16.32 + 0.48(V-32)$
- 48-63 bits : $19.44 + 0.48(V-48)$
- 4-70 : $0.9 + 0.12T$

Interrupt Action

Fixed-point overflow.

Notes

1. All 31 bit positions of the integer are shifted. The sign is not altered. Zeros are inserted in the right-hand end of the operand for each shift.
2. If a bit is shifted out of the left-hand end that is not identical to the sign bit, a fixed-point overflow condition exists.

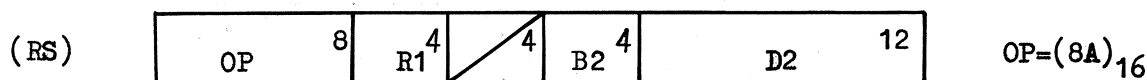
SRA

6.2.25 SHIFT RIGHT SINGLE

General Description

The integer portion of the operand in the general register specified by the first address is shifted right the number of positions specified by the second address. The second address is used as a count and not to address data. The low-order six bits of the second address field constitute the count. The remaining bits are ignored.

Machine Format



Usercode Format

SRA R, DB

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - not used.

Timing (μsecs)

- 4-50 : Under 16 bits : $8.16 + 0.48(V)$
- 16-31 bits : $10.32 + 0.48(V-16)$
- 32-47 bits : $12.48 + 0.48(V-32)$
- 48-63 bits : $12.48 + 0.48(V-48)$
- 4-70 : $0.9 + 0.12T$

Interrupt Action

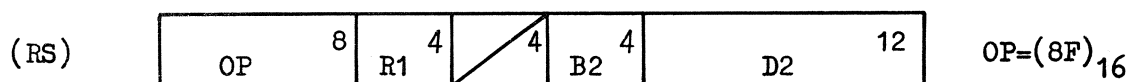
None.

Notes

1. All 31 bit positions of the integer are shifted. The sign is not altered. The sign bit is propagated through the positions vacated in the left end of the operand. The bits shifted out to the right are lost.
2. Shifting to the right is equivalent to low-order truncation or division by powers of two.
3. Shifts greater than 31 cause all significant bits to be lost. A zero for positive numbers and a minus one for negative numbers is the result of such shifts.
4. Fixed-point positive numbers go towards zero; Fixed-point negative numbers go towards minus one.

6.2.26 SHIFT LEFT DOUBLEGeneral Description

The integer portion of the double-word operand specified by the first address and the first address plus one is shifted left the number of positions specified by the second address. The first address specifies an even-numbered register of an even/odd pair that contains the 63-bit integer to be shifted. The second address is used as a count and not to address data. The low-order six bits of the second address field constitute the count. The remaining bits are ignored.

Machine FormatUsercode Format

SLDA R, DB

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - overflow.

Timing (μsecs)

- 4-50 : Under 16 bits : $11.04 + 0.96(V)$
- 16-31 bits : $15.12 + 0.96(V-16)$
- 32-47 bits : $19.20 + 0.96(V-32)$
- 48-63 bits : $23.28 + 0.96(V-48)$
- 4-70 : $1.6 + 0.12T$

Interrupt Action

Fixed-point overflow.

Address error:

Specification.

Notes

1. All 63 bit positions of the integer are shifted. The sign bit (position 0) in the even register is not altered. Zeros are inserted in the right-hand end of the double-word operand for each shift.
2. If a bit is shifted out of the left-hand end that is not identical to the sign bit, a fixed-point overflow condition exists.

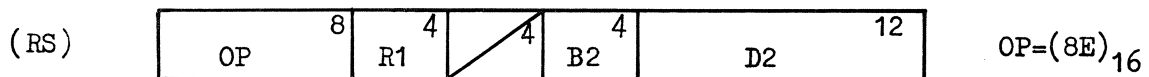
SRDA

6.2.27 SHIFT RIGHT DOUBLE

General Description

The integer portion of the double-word operand specified by the first address and the first address plus one is shifted right the number of positions specified by the second address. The first address specifies an even-numbered register of an even/odd pair that contains the 63-bit integer to be shifted. The second address is used as a count and not to address data. The low-order six bits of the second address constitute the count. The remaining bits are ignored.

Machine Format



Usercode Format

SRDA R, DB

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - not used

Timing (μsecs)

- 4-50 : Under 16 bits : $9.36 + 0.96(V)$
- 16-31 bits : $12.48 + 0.96(V-16)$
- 32-47 bits : $15.60 + 0.96(V-32)$
- 48-63 bits : $18.72 + 0.96(V-48)$
- 4-70 : $1.6 + 0.12T$

Interrupt Action

Address error:

Specification.

Notes

1. All 63 bit positions of the integer are shifted. The sign bit in the leftmost position of the even-numbered register is not altered. This sign bit is propagated through the positions vacated in the left end of the double-word operand. The bits shifted out to the right are lost.
2. A shift count of zero provides a double-word sign and magnitude check.

6.2.28 INTERRUPT ACTION

The following interrupt conditions can occur as a result of fixed-point instructions:

Address Error

Addressing:- An address error interrupt occurs when an address specifies a location outside the available main store. The operation is terminated at the point of error. The result data and the condition code, if produced, are unpredictable.

Specification:- An address error interrupt occurs when an instruction specifies:

1. A full-word operand that is not located on a 32-bit boundary.
2. A halfword operand that is not located on a 16-bit boundary.
3. A double-word operand that is not located on a 64-bit boundary.
4. A register with an odd-numbered address when using an even/odd pair containing a 64-bit operand.

The instruction is suppressed. The condition code, data in main store, and registers remain unchanged.

Protection:- An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is suppressed and the condition code and data in the registers and main store are unaltered. The only exception is the Store Multiple instruction which is terminated. The amount of data stored is unpredictable. (This interrupt can only occur if the store protect feature is installed).

Data Error

A data error interrupt occurs when an invalid digit or sign code of the decimal operand is encountered in the Convert to Binary instruction. The operation is suppressed and the condition code and data in the register and main store are unaltered.

Fixed-Point Overflow

A fixed-point overflow interrupt occurs when the results overflow in sign control, add, subtract or shift operations. The operation is completed by placing the truncated result in the register and setting Condition Code 3. Overflow bits are lost. If the fixed point program mask bit is reset, interrupt will not occur and the flag in the Interrupt Flag Register (IFR) will not be set.

Divide Error

A divide error interrupt occurs when the quotient would exceed the register size in division, or the result of a Convert to Binary instruction exceeds 31 bits. The operation is suppressed and the data in the registers remains unaltered.

6.3 DECIMAL ARITHMETIC INSTRUCTIONS

6.3.1 ADD DECIMAL

General Description

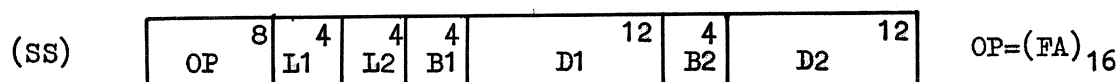
The operand specified by the second address is added algebraically to the operand specified by the first address. The result is stored in the field specified by the first address. The sign and the magnitude of the sum determine the condition code.

The operands can be variable in length up to 16 bytes and must be in packed format. If operands overlap, their rightmost byte location must coincide.

The addition of the two operands can cause decimal overflow. Two conditions which cause overflow are:

- a) a carry out of the high-order position of the result.
- b) a second operand that is larger than the first operand and significant result positions are lost.

Machine Format



Usercode Format

AP DLB, DLB

Condition Code

- 0 - sum is zero.
- 1 - sum is less than zero.
- 2 - sum is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : 15.36 + 1.80N1 + 0.42N2
 4-70 : 4.2 + 0.69N1 + 0.36N2

Interrupt Action

Address error:

Addressing.
 Protection.

Data error.
 Decimal overflow.

Notes

1. High-order zeros are supplied for either operand during instruction execution.
2. All signs and digits are checked for validity.
3. The operand specified by the second address is unaltered.
4. Processing is from right to left.
5. A zero result is always positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the correct result.

6.3.2 SUBTRACT DECIMAL

General Description

The operand specified by the second address is subtracted algebraically from the operand specified by the first address. The result is stored in the field specified by the first address. The sign and the magnitude of the difference determine the condition code.

The operands can be variable in length up to 16 bytes and must be in packed format. If operands overlap, their rightmost byte location must coincide.

The subtraction of two operands can cause decimal overflow.

Machine Format

(SS)	8	4	4	4	12	4	12	
	OP	L1	L2	B1	D1	B2	D2	OP=(FB) ₁₆

Usercode Format

SP DIB, DIB

Condition Code

- 0 - difference is zero.
- 1 - difference is less than zero.
- 2 - difference is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : $15.36 + 1.80N1 + 0.42N2$

4-70 : $4.2 + 0.69N1 + 0.36N2$

Interrupt Action

Address error:

Addressing.

Protection.

Data error.

Decimal overflow.

Notes

1. High-order zeros are supplied for either operand during instruction execution.
2. All signs and digits are checked for validity.
3. The operand specified by the second address is unaltered.
4. Processing is from right to left.
5. A zero difference is always positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the correct difference.

6.3.3 ZERO AND ADD

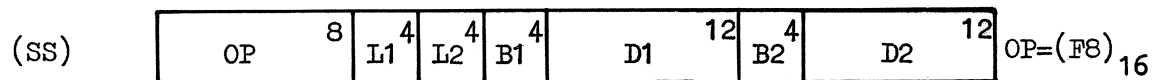
General Description

The operand specified by the second address is loaded into the location specified by the first address. The operation is equivalent to an addition to zero and the result of the addition determines the condition code.

The operands may be variable in length up to 16 bytes and must be in packed format. High-order zeros are provided when necessary. Operands may overlap if their rightmost byte locations coincide, or if the rightmost byte of the first operand is to the right of the rightmost byte of the second operand.

A second operand that is longer than the first operand causes overflow.

Machine Format



Usercode Format

ZAP DLB, DLB

Condition Code

- 0 - result is zero.
- 1 - result is less than zero.
- 2 - result is greater than zero.
- 3 - overflow.

Timing (μsecs)

4-50 : 15.48 + 1.08N1 + 0.42N2

4-70 : 4.7 + 0.25N1 + 0.20N2

Interrupt Action

Address error:

Addressing.
Protection.

Data error.
Decimal overflow.

ZAP

Notes

1. Only the second operand is checked for valid sign and digit codes.
2. The second operand is unaltered.
3. Processing is from right to left.
4. A zero result is positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the second operand.

6.3.4 COMPARE DECIMAL

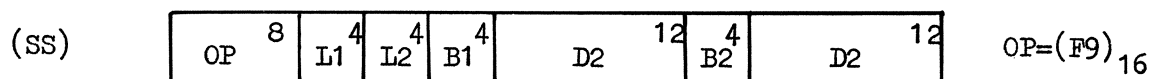
General Description

The operand specified by the first address is algebraically compared with the operand specified by the second address. The results of the comparison determine the condition code.

The operands may be variable in length up to 16 bytes and must be in packed format. The shorter operand is extended with high-order zeros when the operands are unequal in length. If operands overlap, their right-most byte location must be identical.

Overflow cannot occur as a result of this operation.

Machine Format



Usercode Format

CP DLB, DLB

Condition Code

- 0 - the fields are numerically equal.
- 1 - the first operand is algebraically less than the second operand.
- 2 - the first operand is algebraically greater than the second operand.

Timing (μsecs)

- 4-50 : 16.8 + 1.08N1 + 0.42N2
- 4-70 : 4.2 + 0.42N1 + 0.42N2

Interrupt Action

Address error:

Addressing.

Data error.

Notes

1. All signs and digits are checked for validity.
2. Both operands are unaltered.
3. Comparison is from right to left.
4. A positive zero compares equally to a negative zero.

6.3.5 MULTIPLY DECIMALGeneral Description

The operand specified by the first address (multiplicand) is multiplied by the operand specified by the second address (multiplier). The product is stored in the location of the first operand, right-justified.

The operands may be variable in length and must be in packed format. Operands can overlap if their rightmost byte locations coincide.

The second operand (multiplier) must be shorter than the first operand (multiplicand) and must not exceed eight bytes in length (15 digits plus sign). Otherwise, an address error (specification) occurs.

The multiplicand must have high-order zero ^{bytes} ~~digits~~ equal to the number of ~~digits~~ in the multiplier, or a data error occurs. The maximum product size is 31 digits.

Machine Format

(SS)	OP	8	L1	4	L2	4	B1	4	D1	12	B2	4	D2	12	OP=(FC) ₁₆
------	----	---	----	---	----	---	----	---	----	----	----	---	----	----	-----------------------

Usercode Format

MP DLB, DLB

Condition Code

Unchanged.

Timing (usecs)

4-50 : $28.49 + 16.96N1 - 14.35N2 + 2.34N2 (N1-N2)$
 4-70 : $5.5 + 1.27N1 + 2.09N2 + 2.7N2 (N1-N2)$

Interrupt Action

Address error.

Addressing.

Protection.

Specification.

Data error.

Notes

1. All signs and digits are checked for validity.
2. The second operand is unaltered unless operands overlap.
3. Overflow cannot occur.
4. The sign of the product is determined by the rules of algebra, even if one, or both, operands are zero; that is, minus zero is a possible result.

6.3.6 DIVIDE DECIMALGeneral Description

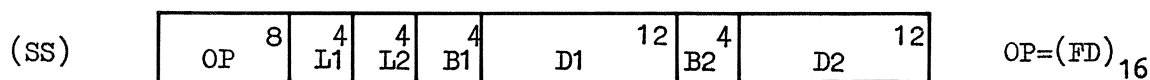
The operand specified by the first address (the dividend) is divided by the operand specified by the second address (the divisor) and the result (quotient plus remainder) replaces the first operand. The quotient is placed leftmost in the first operand field. The remainder, which has a size equal to the divisor size, is placed rightmost in the first operand field.

The operands may be variable in length and must be in packed format. Overlapping is allowed if the rightmost byte locations are identical. The second operand (the divisor) must be shorter than the first operand (the dividend) and must not exceed eight bytes in length (15 digits plus sign). If either rule is not observed, an address error (specification) occurs.

The dividend must have at least one high-order zero. Otherwise, a decimal divide error occurs.

Together, the quotient and remainder occupy the entire dividend field after division. Therefore, the address of the quotient field is the address of the dividend field and its size in bytes is $L1 - L2 - 1$. The quotient and remainder are signed integers which are right-aligned in the first operand.

No overflow can occur. A quotient that is larger than the number of digits allowed causes a decimal divide error.

Machine formatUsercode Format

DP DLB, DLB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : $26.33 + 36.71N1 - 35.14N2 + 5.40N2 (N1-N2)$
 4-70 : $8.04N1 - 7.8N2 + 5.42N2 (N1-N2) - 1.3$

Interrupt Action

Address error:

Addressing.

Protection.

Specification.

Data Error.

Decimal divide error.

Notes

1. All signs and digits are checked for validity.
2. The second operand is unaltered.
3. The sign of the quotient is determined by the rules of algebra from dividend and divisor signs. The sign of the remainder has the same value as the dividend sign.
4. The first address plus $(L1 - L2)$ specifies the address of the remainder. The length of the remainder is specified by $L2$.

PACK

6.3.7 PACK

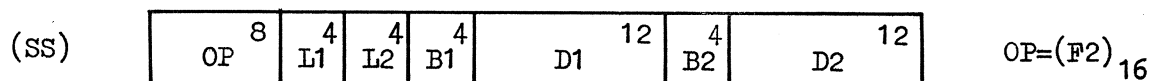
General Description

The operand specified by the second address is converted from zoned format to packed format and the result is placed in the location specified by the first address.

The operand specified by the second address must be in zoned format. The sign is obtained from the zone portion of the rightmost byte of the second operand and is placed in the rightmost four bits of the first operand (result field). All other zones are ignored. The four-bit numeric portions (stripping the four-bit zone) of each byte are then placed adjacent to the sign, and to each other, to fill the result field.

The result is extended with high-order zeros if the second operand field is shorter than the first. If the first operand field is not large enough to contain all the significant digits from the second operand field, the remaining digits are ignored. The operands may overlap.

Machine Format



Usercode Format

PACK DLB, DLB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 8.88 + 1.92N1 + 0.96N2

4-70 : 3.6 + 0.48N1 + 0.17N2

Interrupt Action

Address error:

Addressing.

Protection.

Notes

1. Signs and digits are not checked for validity.
2. The second operand is not changed except when the operands overlap.
3. Processing is from right to left, one byte at a time.

UNPK

6.3.8 UNPACK

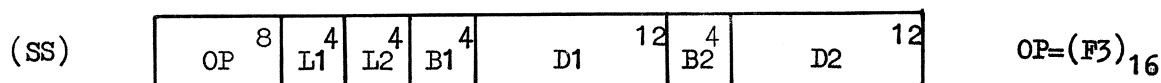
General Description

The operand specified by the second address is converted from packed format to zoned format and the result is placed in the location specified by the first address.

Each of the eight-bit bytes of the packed, second-operand field represents two four-bit digits. Each of the four-bit digits is stored in a byte of the first operand field in the low-order four-bit positions. If the Decimal Code is EBCDIC, a zone code of 1111 is inserted into the high-order four bits of each byte. If the Decimal Code is ISO, a zone code of 0101 is inserted. These zones are inserted in all but the zone portion of the right-most byte, which receives the sign of the packed operand.

If the first operand is not large enough to receive the significant digits of the second operand, the remaining digits are ignored. The second-operand field is extended with zero digits before unpacking.

Machine Format



Usercode Format

UNPK DLB,DLB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 9.90 + 0.96N1 + 0.90N2

4-70 : 3.6 + 0.23N1 + 0.48N2

Interrupt Action

Address Error:

Addressing.

Protection.

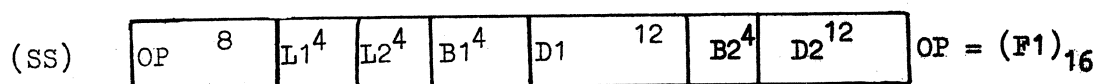
Notes

1. Signs and digits are not checked for validity.
2. The second operand is not altered, except when operands overlap.
3. Processing is from right to left.

6.3.9 MOVE WITH OFFSETGeneral Description

The operand specified by the second address is offset 4 bits to the **left** (a 1-digit left shift) and is placed to the left of, and adjacent to, the low-order four bits of the operand specified by the first address.

If the first operand is not large enough to receive all bytes of the second operand, the remaining bytes are ignored. If the second operand is shorter than the first operand, the second operand is extended with high-order zeros. The first and second operands may overlap.

Machine FormatUsercode Format

MVO DLB, DLB

Condition Code

Unchanged

Timing (μsecs)

4-50 : 11.04 + 1.92N1 + 0.96N2

4-70 : 3.6 + 0.48N1 + 0.20N2

Interrupt Action

Address error:

Addressing.

Protection.

Notes

1. Signs and digits are not checked for validity.
2. The second operand is not changed except when operands overlap.
3. Processing is from right to left.
4. The initial low-order 4-bit digit of the operand specified by the first address is left unaltered.

6.3.10 INTERRUPT ACTION

The following interrupt conditions can occur as a result of a decimal arithmetic instruction.

Address Error

Addressing:-

An address error interrupt exists when an address specifies a location outside the available main store of the particular installation. The operation is terminated at the point of error. The result data and the condition code are unpredictable.

Specification:-

An address error interrupt exists when a multiplier or divisor size exceeds 15 digits plus sign; or when the multiplier size or the divisor size is equal to, or greater than, the multiplicand or dividend size, respectively. The instruction is suppressed. The condition code, data in the main store, and registers remain unchanged.

Protection:-

An address error interrupt exists when the protection key and the storage key of the result location do not match. The operation is terminated. The result data and condition code are unpredictable. (This interrupt can occur only if the store protect feature is installed.)

Data Error

A data error interrupt exists in decimal arithmetic when an invalid sign (not greater than nine) or digit code (not zero to nine) is detected in an operand, a multiplicand has insufficient high-order zeros, or there is incorrect overlapping of operands. The operation is terminated. The result data and the condition code setting are unpredictable.

Decimal Overflow

A decimal overflow interrupt exists when the result field of an Add Decimal, Subtract Decimal, or Zero and Add instruction is too small to contain the overflow data. The operation is completed by ignoring the overflow data, and setting the condition code to 3. If the decimal overflow program mask bit is reset, interrupt will not occur and the flag in the IFR will not be set.

Divide Error

A divide error interrupt occurs when the quotient is greater than the specified data field, including division by zero, or the dividend does not have one leading zero. Division is suppressed and the dividend and divisor remain unchanged in main store.

6.4 LOGICAL INSTRUCTIONS6.4.1 MOVEGeneral Description

For the SI format Move instruction, the immediate byte in the I2 field of the instruction being executed is stored in the main store location specified by the first address.

To process the SS format Move instruction, the source field specified by the second address is moved into the destination field specified by the first address. This format is used for a main store to main store move.

Machine Formats

(SI)	OP	8	I2	8	B1	4	D1	12
------	----	---	----	---	----	---	----	----

OP=(92)₁₆

(SS)	OP	8	L	8	B1	4	D1	12	B2	4	D2	12
------	----	---	---	---	----	---	----	----	----	---	----	----

OP=(D2)₁₆
Usercode Formats

(SI) MVI DB, I
(SS) MVC DLB, DB

Condition Code

Unchanged.

Timing (µsecs)

4-50 : MVI 5.04
MVC 8.94 + 1.44N
4-70 : MVI 1.9
MVC 3.1 + 0.45N

Interrupt Action

Address error:

Addressing.

Protection.

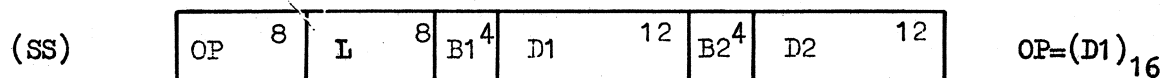
MVI,MVC

Notes

1. The bytes being moved are not inspected or changed.
2. Processing is from left to right and overlapping of fields is permitted.
3. The second operand is not altered, unless operands overlap in the SS format.
4. It is possible to propagate one byte through an entire field by having the first operand address specify one location to the right of the second operand address.

6.4.2 MOVE NUMERICSGeneral Description

The low-order four bits of each byte in the source operand specified by the second address are placed into the low-order four bits of the corresponding byte of the destination operand specified by the first address.

Machine FormatUsercode Format

MVN DLB, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 9.90 + 2.22N

4-70 : 3.1 + 0.79N

Interrupt Action

Address error:

Addressing.

Protection.

Notes

1. The numerics are not changed or checked for validity.
2. The operand specified by the second address is not altered, unless operands overlap.
3. Processing is from left to right.
4. The high-order four bits of the source and destination operand bytes are not altered.
5. The operand fields may overlap in any way and may be variable in length.

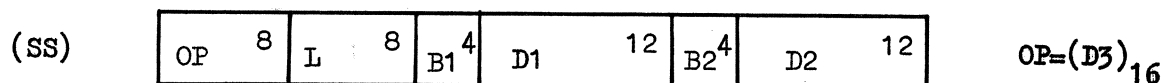
MVZ

6.4.3 MOVE ZONES

General Description

The high-order four bits of each byte in the source operand specified by the second address are placed into the high-order four bits of the corresponding byte of the destination operand specified by the first address.

Machine Format



Usercode Format

MVZ DLB, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 9.90 + 2.22N

4-70 : 3.1 + 0.79N

Interrupt Action

Address error:

Addressing.

Protection.

Notes

1. The zones are not changed or checked for validity.
2. The operand specified by the second address is not altered, unless operands overlap.
3. Processing is from left to right.
4. The low-order four bits of the source and destination operand bytes are not altered.
5. The operand fields may overlap in any way and may be variable in length.

6.4.4 COMPARE LOGICALGeneral Description

The operand specified by the first address is logically compared with the operand specified by the second address. The result of the comparison determines the condition code. These instructions process all bits as part of an unsigned binary quantity. All codes are valid and the instruction is terminated on inequality or when the operand bytes have been exhausted.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>R2⁴</td></tr></table>	OP	8	R1 ⁴	R2 ⁴	OP=(15) ₁₆						
OP	8	R1 ⁴	R2 ⁴									
(RX)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>X2⁴</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12	OP=(55) ₁₆			
OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12						
(SI)	<table><tr><td>OP</td><td>8</td><td>I2</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td></tr></table>	OP	8	I2	8	B1 ⁴	D1	12	OP=(95) ₁₆			
OP	8	I2	8	B1 ⁴	D1	12						
(SS)	<table><tr><td>OP</td><td>8</td><td>L</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12	OP=(D5) ₁₆
OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12			

Usercode Formats

(RR)	CLR R,R
(RX)	CL R,DXB
(SI)	CLI DB,I
(SS)	CLC DLB,DB

Condition Code

- 0 - the operands are equal.
- 1 - the first operand is less than the second operand.
- 2 - the first operand is greater than the second operand.
- 3 - not used.

Timing (μsecs)

4-50	: CLR 4.8
	CL 8.4
	CLI 6.0
	CLC 8.96 + 1.44B
4-70	: CLR 1.1
	CL 4.9
	CLI 2.0
	CLC 3.1 + 0.84N

CLR,CL
CLI,CLC

Interrupt Action

Address error:

Addressing (RX, SI, SS only).
Specification (RX only).

Notes

1. Both operands are unaltered.
2. In the SI format, the immediate byte in the I2 field of the instruction being executed is the second operand.
3. Processing is from left to right and can extend to field lengths of 256 bytes.
4. The operation can be used for alphanumeric comparisons.

NR,N
NI,NC

Timing (μsecs)

4-50 :	NR	5.28
	N	8.4
	NI	6.96
	NC	$8.95 + 2.22N$
4-70	NR	1.1
	N	2.0
	NI	2.8
	NC	$3.1 + 0.79N$

Interrupt Action

Address error:

Addressing (RX, SI, SS only).
Protection (SI, SS only).
Specification (RX only).

Notes

1. The second operand is unaltered, unless operands overlap in the SS format.
2. In the SI format, the immediate byte in the I2 field of the instruction being executed is the second operand.
3. Processing is from left to right.
4. All operands and results are valid.
5. The "AND" instruction is also used to set a bit to zero.

6.4.6 ORGeneral Description

This instruction performs a logical "OR" operation on two operands bit-by-bit according to the following rules:

Rules for Logical "OR" Operation

If Bit in First Operand is	And Bit in Second Operand is	Then Bit in result is
0	0	0
0	1	1
1	0	1
1	1	1

The logical result of the operation is placed in the location specified by the first address and determines the condition code.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>R2⁴</td></tr></table>	OP	8	R1 ⁴	R2 ⁴	OP=(16) ₁₆						
OP	8	R1 ⁴	R2 ⁴									
(RX)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>X2⁴</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12	OP=(56) ₁₆			
OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12						
(SI)	<table><tr><td>OP</td><td>8</td><td>I2</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td></tr></table>	OP	8	I2	8	B1 ⁴	D1	12	OP=(96) ₁₆			
OP	8	I2	8	B1 ⁴	D1	12						
(SS)	<table><tr><td>OP</td><td>8</td><td>L</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12	OP=(D6) ₁₆
OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12			

Usercode Formats

(RR) OR R,R
 (RX) O R, DXB
 (SI) OI DB, I
 (SS) OC DLB, DB

Condition Code

0 - result is zero.
 1 - result is not zero.
 2 - not used.
 3 - not used.

OR,O
OI,OC

Timing (μsecs)

4-50 : OR 5.28
 O 8.4
 OI 6.96
 OC 8.95 + 2.22N
4-70 : OR 1.1
 O 2.0
 OI 2.8
 OC 3.1 + 0.79N

Interrupt Action

Address error:

Addressing (RX, SI, SS only).
Protection (SI, SS only).
Specification (RX only).

Notes

1. The second operand is unaltered, unless operands overlap in the SS format.
2. In the SI format, the immediate byte in the I2 field of the instruction being executed is the second operand.
3. Processing is from left to right.
4. All operands and results are valid.
5. The "OR" instruction is also used to set a bit to one.

6.4.7 EXCLUSIVE OR

General Description

These instructions perform an Exclusive "OR" operation on two operands bit-by-bit according to the following rules:

Rules for Exclusive "OR" Operation

If Bit of First Operand is	And Bit of Second Operand is	Then Bit in Result is
0	0	0
0	1	1
1	0	1
1	1	0

The modulo-two sum (binary addition without carries) of the operation is placed in the location specified by the first address and determines the condition codes.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>R2⁴</td></tr></table>	OP	8	R1 ⁴	R2 ⁴	OP=(17) ₁₆						
OP	8	R1 ⁴	R2 ⁴									
(RX)	<table><tr><td>OP</td><td>8</td><td>R1⁴</td><td>X2⁴</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12	OP=(57) ₁₆			
OP	8	R1 ⁴	X2 ⁴	B2 ⁴	D2	12						
(SI)	<table><tr><td>OP</td><td>8</td><td>I2</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td></tr></table>	OP	8	I2	8	B1 ⁴	D1	12	OP=(97) ₁₆			
OP	8	I2	8	B1 ⁴	D1	12						
(SS)	<table><tr><td>OP</td><td>8</td><td>L</td><td>8</td><td>B1⁴</td><td>D1</td><td>12</td><td>B2⁴</td><td>D2</td><td>12</td></tr></table>	OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12	OP=(D7) ₁₆
OP	8	L	8	B1 ⁴	D1	12	B2 ⁴	D2	12			

Usercode Formats

(RR) XR R,R
(RX) X R, DXB
(SI) XI DB, I
(SS) XC DLB, DB

Condition Code

0 - result is zero.
1 - result is other than zero.
2 - not used.
3 - not used.

XR,X
XI,XC

Timing (μsecs)

4-50 : XR 5.28
 X 8.4
 XI 6.96
 XC 8.95 + 2.22N
4-70 : XR 1.1
 X 2.0
 XI 2.8
 XC 3.1 + 0.79N

Interrupt Action

Address error:

Addressing (RX, SI, SS only).
Protection (SI, SS only).
Specification (RX only).

Notes

1. The second operand is unaltered, unless operands overlap in the SS format.
2. In the SI format, the immediate byte in the I2 field of the instruction being executed is the second operand.
3. Processing is from left to right.
4. All operands and results are valid.
5. These instructions may be used to complement a number (one's complement).

6.4.8 TEST UNDER MASKGeneral Description

The operand (byte) specified by the first address is tested against the immediate I field (byte) as a mask. The result determines the condition code. The I field is used as an eight-bit mask and is made to correspond one-for-one with the bits of the byte in main store that is specified by the first address.

A bit in the byte being examined is said to be selected when the corresponding mask bit is a one. When the mask bit is a zero, the bit in main store is ignored.

Machine Format

(SI)	OP	8	I2	8	B1	4	D1	12	OP=(91) ₁₆
------	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

TM DB, I

Condition Code

- 0 - selected bits all zero or mask is all zero.
- 1 - selected bits mixed zero and one.
- 2 - not used.
- 3 - selected bits all one's.

Timing (μsecs)

4-50 : 6.48
4-70 : 2.0

Interrupt Action

Address error:

Addressing.

Note

The operands are unaltered.

IC

6.4.9 INSERT CHARACTER

General Description

The eight-bit byte specified by the second address is loaded into the rightmost byte of the general register specified by the first address. The remaining bits of the register are unaltered.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(43) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

IC R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 5.52

4-70 : 3.1

Interrupt Action

Address error:

Addressing.

Note

The operand specified by the second address is not altered or inspected.

6.4.10 STORE CHARACTERGeneral Description

The rightmost eight-bit byte of the general register specified by the first address is stored into the main store location specified by the second address.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(42) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

STC R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 5.04

4-70 : 2.9

Interrupt Action

Address error:

Addressing.

Protection.

Note

The operand specified by the first address is not altered or inspected.

6.4.11 LOAD ADDRESSGeneral Description

The final main store address specified by the second operand is loaded into the rightmost 24 bits of the general register specified by the first address (R1). The leftmost eight bits of the register are set to zeros.

The contents of the registers specified by the X2 and B2 fields are added to the contents of the D2 field of the instruction to obtain an address. This is the address that is loaded into the register specified by the first address. Any carry beyond the rightmost 24 bits is ignored.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(41) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

LA R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 7.92

4-70 : 1.3

Interrupt Action

None

Notes

1. All specified address arithmetic is computed before loading.
2. R1, X2 and B2 may specify the same register; however R1 only may specify register 0.
3. This instruction can be used to increment the low-order 24 bits of a general register (other than 0) by the contents of the D2 field. The register to be incremented is specified by R1, and either X2 (with B2 set to zero) or B2 (with X2 set to zero). Since R1 and X2 or B2 must specify the same register, register zero cannot be incremented (a zero in the B2 or X2 field indicates that the corresponding address component is absent).
4. Main store is not accessed by this instruction.

6.4.12 TRANSLATEGeneral Description

The variable length operand specified by the first address is translated, byte-for-byte, according to the byte translation table specified by the second address. The result replaces the bytes in the field specified by the first address.

The bytes of the first operand are termed the argument bytes. Bytes of the first operand are selected for translation from left-to-right, one byte at a time. Each argument byte is added to the second operand address, which is the starting location of a translation table. This sum, in turn, addresses a byte location within the table containing a function byte. The function byte at this location replaces the original argument byte of the first operand.

The operation terminates when the first operand bytes have been exhausted.

Machine Format

(SS)	OP	8	L	8	B1	4	D1	12	B2	4	D2	12	OP=(DC) ₁₆
------	----	---	---	---	----	---	----	----	----	---	----	----	-----------------------

Usercode Format

TR DIB, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 6.24 + 5.04N

4-70 : 3.1 + 1.53N

Interrupt Action

Address error:

Addressing.

Protection.

Notes

1. The translation table is unaltered unless overlap occurs.
2. The field to be translated and the translation table are addressed by their leftmost byte.

3. The length of a table, in general, must be 256 bytes, unless the domain of argument bytes is limited to a specific subset by the program and data.
4. The L field specifies the length of the first operand minus one (binary 00000001 = 2 bytes).

6.4.13 TRANSLATE AND TESTGeneral Description

The variable length operand, which is specified by the first address, is used as the argument (byte-by-byte) to reference a list (functions) specified by the second address. The functions referenced are inspected for zero or non-zero. If a non-zero is encountered, the address of the argument byte is loaded into General Register 1 (General Register 13 in P3; General Register 9 in P4) and the function byte is loaded into the rightmost end of General Register 2 (General Register 14 in P3; General Register 10 in P4). Whenever zeros are encountered in the function list, the operation proceeds to the next byte. The first operand is unaltered.

The bytes of the first operand are termed the argument bytes. Processing of the first operand is from left-to-right, one byte at a time. Each argument byte is added to the second operand, which is the starting location of the translate table. This sum, in turn, addresses a byte location within the table, which is termed a function byte. Then, the function byte retrieved from the table is inspected for all zeros.

If the function byte is all zeros, the operation proceeds to the next argument byte and continues processing. If the function byte is not all zeros, the instruction inserts the address of the argument byte in the low-order 24 bits of General Register 1 (13 or 9) and inserts the retrieved non-zero function byte in the low-order eight-bits of General Register 2 (14 or 10). The high-order eight bits of General Register 1 (13 or 9) and high-order 24 bits of General Register 2 (14 or 10) are unaltered.

The operation terminates when a (non-zero) function byte is accessed or when the first operand field is exhausted.

Machine Format

(SS)	OP	8	L	8	B1	4	D1	12	B2	4	D2	12	OP=(DD) ₁₆
------	----	---	---	---	----	---	----	----	----	---	----	----	-----------------------

Usercode Format

TRT DLB, DB

Condition Code

- 0 - accessed function bytes all zeros.
- 1 - a non-zero function byte is encountered before the first operand field is exhausted.
- 2 - the last function byte is non-zero.
- 3 - not used.

Timing (µsecs)

4-50 : 11.04 + 4.08B

4-70 : 4.8 + 1.36N

Interrupt Action

Address error:

Addressing.

Notes

1. The variable length field specified by the first address is unaltered.
2. If non-zero functions do not occur, General Registers 1 (13 or 9) and 2 (14 or 10) are unaltered.
3. The first operand and the translation table are addressed by their leftmost bytes.
4. The length of the table, in general, must be 256 bytes, unless the domain of argument bytes is limited to a specific subset by the program and data.
5. The L field specifies the length of the first operand minus one.
6. This instruction is useful for scanning input streams and locating delimiters for variable length records and fields.
7. In processor states P1 and P2, General Registers 1 and 2 are used. In processor state P3, General Registers 13 and 14 are used. In processor state P4 General Registers 9 and 10 are used.

6.4.14 EDITGeneral Description

The variable length source field specified by the second address is changed from packed format to zoned format with the results edited under the control of a mask pattern. The result of the operation replaces the mask pattern specified by the first address and determines the condition code.

The L field applies to the mask pattern (first address field). The source digits are processed left-to-right, one byte at a time. The leftmost four bits of each byte are examined first and the rightmost four bits of each byte are held available for the next mask character that calls for digit examination. Immediately after the leftmost four bits have been examined, the rightmost four bits are checked for a sign code. When one of the sign codes is encountered, these bits are no longer treated as a digit. A new character is fetched from the mask pattern for the next digit to be examined.

Machine FormatUsercode Format

ED DLB, DB

Editing Rules

Editing includes sign control, punctuation control, zero suppression or check protection, and also facilitates blanking of all-zero fields. In addition, multiple fields of digits can be edited in one operation, and numeric data can be combined with alphabetic and special characters.

Editing rules depend on the control code, significance, and the source digit, and are given as follows:

Editing Rules

Control Codes	Hexadecimal Code	Decimal Code	Function
Filler	Any	Any	*Replaces leading zeros.
Start Significance	21	33	Stops replacement of leading zeros. Also acts as a digit select code.
Digit Select	20	32	Specifies digit position in data (replaced by filler code if appears after a negative sign has been sensed).
Field Separator	22	34	Indicates editing of a new field is to begin (replaced by filler code).
Insertion Character	Any	Any	Inserted in the result.

*The most common filler characters are the blank and the asterisk.

1. Source digits are examined only when a digit select code (20)₁₆ or a start significance code (21)₁₆ is encountered in the mask pattern.
2. Significance is established either:
 - (a) upon encountering a non-zero digit in the source field.
 - (b) after encountering a start significance code (21)₁₆ within the mask pattern.
3. If significance has not been established, every control code or insertion character encountered in the mask pattern (including the start significance code) is replaced by the filler character.
4. If significance has been established, every digit select code (20)₁₆ or start significance code (21)₁₆ encountered in the mask pattern is replaced by a digit from the source field, which is expanded by attaching a zone.
5. If significance has been established, every insertion character (other than the digit select, start significance, or field separator codes) encountered within the mask pattern is left in place without alteration.
6. Significance is disestablished by:
 - (a) encountering a field separator code (22)₁₆ in the mask pattern.
 - (b) encountering a positive (plus) sign within the rightmost four bits of a source field byte.

7. A negative (minus) sign within the rightmost four bits of a source byte does not disestablish significance. Additional digit select codes encountered in the mask pattern are replaced by filler characters, but insertion characters are left in place without alteration.

8. Field separator codes $(22)_{16}$ are always replaced by the filler character.

Note: The filler character is obtained from the mask pattern as part of the editing operation. The first character (leftmost byte) of the mask pattern is used as a filler character and is left unchanged in the result, except:

- (a) when it is a digit select code.
- (b) when it is a start significance code.

In these codes, a source digit is examined and, when non-zero, inserted in the result field.

To facilitate blanking out all-zero result fields, or triggering negative field special processing, the condition code is used to indicate the sign and zero status of the last field edited. All digits examined are tested for zero, and the presence, or absence, of an all-zero source field is indicated in the condition code at the termination of the editing operation. Sign significance is also indicated by the condition code.

Condition Code

0 - indicates a zero source field regardless of whether or not significance is established.

1 - indicates non-zero result field with significance established to indicate less than zero.

2 - indicates non-zero result field with no significance established to indicate greater than zero.

3 - not used.

Note: The condition code setting reflects only the field following the last (rightmost) field separator code of the mask pattern for multiple-field-editing operations.

Timing (μ secs)

4-50 : $10.56 + 3.00N1 + 1.92N2 - 0.12F - 0.6K$

4-70 : $3.1 + 0.6K + 0.93D$

Interrupt Action

Address error:

Addressing.

Protection.

Data error.

Notes

- A The leftmost four-bits of any source field byte must be a valid digit, otherwise a data error interrupt occurs.
- B The rightmost four-bits of any source field byte can be either a digit or a sign.
- C Multiple field editing is possible by using the field separator code within the mask pattern.
- D The zones of the expanded source digits can be either EBCDIC or ISO , as specified by the mode code. When the mode code specifies EBCDIC, zone code 1111 is generated. When the mode code specifies ISO , the zone code 0101 is generated.
- E The rightmost four bits of any source field byte can be a digit or sign as follows:

Codes	Definition
0000 → 1001 1010, 1100, 1110, 1111 1011, 1101	Digits Plus Sign Minus sign

- F Overlapping of fields yields unpredictable results.

6.4.15 EDIT AND MARKGeneral Description

The variable length source field specified by the second address is changed from packed format to zoned format and the results are edited under control of a mask pattern. The result of the operation replaces the mask pattern specified by the first address and determines the condition code. In addition, the address of each first significant result digit is stored in General Register 1 (General Register 13 in P3; General Register 9 in P4).

The operation of this instruction is identical to the Edit instruction except for the additional function of inserting a byte address in General Register 1 (13 or 9). The destination address of the digit that establishes significance within the source field being edited is loaded into the rightmost 24 bits of General Register 1 (13 or 9). The leftmost eight bits are unaltered. The address is not loaded when significance is forced by recognition of the start significance code in the mask pattern.

The Edit and Mark instruction facilitates the insertion of floating currency symbols, sign indicators, relational operators, and other editing symbols (\$, +, -, <, >, etc.). The address loaded into the register is one byte to the right of the address where such a symbol would be inserted. (The Branch on Count instruction, with zero in the R2 field, can be used to reduce the loaded address by one.)

Because the address is not loaded when significance is forced by the start significance code, the address of the byte immediately to the right of the start significance code in the mask pattern field should be loaded in General Register 1 (13 or 9) before an Edit and Mark instruction is executed.

Machine Format

(SS)	OP ⁸	L ⁸	B1 ⁴	D1 ¹²	B2 ⁴	D2 ¹²	OP=(DF) ₁₆
------	-----------------	----------------	-----------------	------------------	-----------------	------------------	-----------------------

Usercode Format

EDMK DLB, DB

Condition Code

0 - indicates a zero source field whether or not significance is established.

1 - indicates non-zero result field with significance established to indicate less than zero.

2 - indicates non-zero result field with no significance established to indicate greater than zero.

EDMK

3 - not used.

Timing (μsecs)

4-50 : $13.44 + 3.00N1 + 1.92N2 - 0.12F - 0.6K$

4-70 : $3.9 + 0.6K + 0.93D$

Interrupt Action

Address error:

Addressing.

Protection.

Data error.

Notes

- A All notes of the Edit instruction are applicable to the Edit and Mark instruction.
- B The address of the byte is loaded each time significance is established and a non-zero character is inserted into the result field.
- C The address is loaded into the rightmost 24 bits of General Register 1 (13 or 9). The leftmost eight bits are unaltered.
- D When a single instruction is used to edit multiple fields, the address of the first significant digit of each field is loaded into the register. However, only the address of the last field processed will be available upon completion of the instruction.
- E In processor states P1 and P2, General Register 1 is used. In processor state P3, General Register 13 is used. In processor state P4, General Register 9 is used.

6.4.16 SHIFT LEFT SINGLE LOGICALGeneral Description

The entire contents of the general register specified by the first address are shifted left the number of bit positions specified by the second address. The R3 field is ignored.

The second address does not refer to a main store location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(89) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

SLL R,DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : Under 16 bits : $7.92 + 0.48 (V)$
 16-31 bits : $11.04 + 0.48 (V-16)$
 32-47 bits : $14.16 + 0.48 (V-32)$
 48-63 bits : $17.28 + 0.48 (V-48)$

4-70 : $0.9 + 0.12T$

Interrupt Action

None.

Notes

1. High-order bits of the register are shifted out and lost.
2. Zeros are placed into the right end of the register.
3. All 32 bits of the specified register are shifted.

SRL

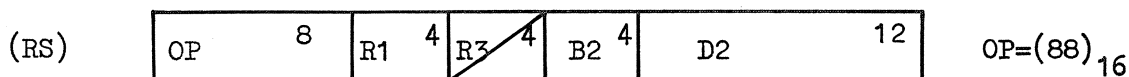
6.4.17 SHIFT RIGHT SINGLE LOGICAL

General Description

The entire contents of the general register specified by the first address are shifted right by the number of bit positions specified by the second address. The R3 field is ignored.

The second address does not refer to a main store location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Machine Format



Usercode Format

SRL R, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : Under 16 bits : $8.88 + 0.48(V)$
16-31 bits : $11.04 + 0.48(V-16)$
32-47 bits : $13.20 + 0.48(V-32)$
48-63 bits : $13.20 + 0.48(V-48)$
4-70 : $0.9 + 0.12T$

Interrupt Action

None.

Notes

1. Low-order bits of the register are shifted out and lost.
2. Zeros are placed into the left end of the register.
3. All 32 bits of the specified register are shifted; that is, the operation is unsigned.

6.4.18 SHIFT LEFT DOUBLE LOGICALGeneral Description

The entire contents of the double-length operand (two general registers) -even/odd specified by the first address are shifted left the number of bit positions specified by the second address. The R3 field is ignored.

The second address does not refer to a main store location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(8D) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

SLDL R, DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : Under 16 bits : $7.68 + 0.96(V)$
 16-31 bits : $11.76 + 0.96(V-16)$
 32-47 bits : $15.84 + 0.96(V-32)$
 48-63 bits : $19.92 + 0.96(V-48)$
 4-70 : $1.6 + 0.12T$

Interrupt Action

Address error:

Specification.

Notes

1. The first address must specify an even-numbered register.
2. All 64 bits of the double-length operand are shifted.
3. High-order bits are shifted out and lost.
4. Zeros are placed into the low-order end of the odd-numbered register.

SRDL

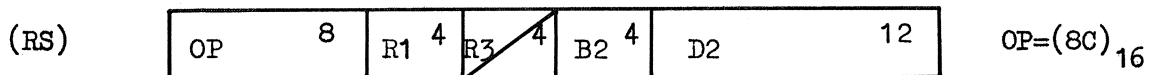
6.4.19 SHIFT RIGHT DOUBLE LOGICAL

General Description

The entire contents of the double-length operand (two general registers) -even/odd specified by the first address are shifted right the number of bit positions specified by the second address. The R3 field is ignored.

The second address does not refer to a main store location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Machine Format



Usercode Format

SRDL R, DB

Condition Code

Unchanged

Timing (μsecs)

4-50 : Under 16 bits : $7.44 + 0.96(V)$
 16-31 bits : $10.56 + 0.96(V-16)$
 32-47 bits : $13.68 + 0.96(V-32)$
 48-63 bits : $16.80 + 0.96(V-48)$
4-70 : $1.6 + 0.12T$

Interrupt Action

Address error:

Specification.

Notes

1. The first address must specify an even-numbered register.
2. All 64 bits of the double-length operand are shifted.
3. Low-order bits are shifted out and lost.
4. Zeros are placed into the high-order end of the even-numbered register.

6.4.20 INTERRUPT ACTION

The following interrupt conditions can occur as a result of logical instructions:

Address Error

Addressing:-

An address error interrupt occurs when an address specifies a location outside the available store. At the point of error the operation is terminated. The result data and condition code, if affected, are unpredictable.

Specification:-

An address error interrupt occurs when a full-word operand is not located on a word boundary in a storage-to-register operation, or when an odd register is specified as the first register in an instruction which performs an operation on an even/odd pair of general registers. The operation is suppressed.

Protection:-

An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is suppressed and the condition code, registers and main store are unaltered. The variable-length store-to-store instructions are the only exception, in which case the operation is terminated and the result data and the condition code setting are unpredictable. (This interrupt can only occur if the store protect feature is installed.)

Data Error

A data error occurs if a digit code of the second operand in the Edit instruction or Edit and Mark instruction is invalid. The operation is terminated, and the result data and condition code setting are unpredictable.

BCR,BC

6.5 BRANCHING INSTRUCTIONS

6.5.1 BRANCH ON CONDITION

General Description

If the condition code is set to any of the conditions specified by the four-bit mask field (M or M1), the P counter is replaced by the branch address. If the four-bit mask field is not equivalent to the condition code settings, branching does not occur and the next instruction in sequence is executed. The branch is initiated whenever the condition code has a corresponding mask bit set.

Machine Formats

(RR)	<table><tr><td>OP</td><td>8</td><td>M1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	M1	4	R2	4	OP=(07) ₁₆
OP	8	M1	4	R2	4			

(RX)	<table><tr><td>OP</td><td>8</td><td>M</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	M	4	X2	4	B2	4	D2	12	OP=(47) ₁₆
OP	8	M	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR) BCR M,R

(RX) BC M,DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : BCR Branch 3.84
 No Branch 3.36

 BC Branch 4.56
 No Branch 4.56

4-70 : BCR 2.2
 BC 3.4

Interrupt Action

None

Notes

1. The four-bit mask in M1 corresponds, left-to-right, with the four condition codes:

Instruction Bit	Condition Code
8	0
9	1
10	2
11	3

2. If all mask bits are set ($M1 = F_{16}$), an unconditional branch is effected.
3. When all mask bits are zero, or if R2 in the RR format is zero, the instruction is a no-op.
4. When a branch occurs, the leftmost eight-bit portion of the 32-bit P counter (Instruction Length Code (ILC), CC and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.
5. The contents of the registers specified by the second address are unaltered.

For extended mnemonic codes of the BC instruction see Section 7.1.0
Page 7-2.

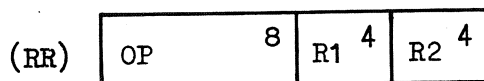
BALR,BAL

6.5.2 BRANCH AND LINK

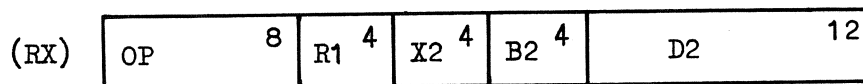
General Description

The entire 32-bit contents of the P counter are loaded into the general register specified by R1. Then, the program branches to the instruction address specified by the branch address. The instruction length counter, the condition code, the program mask, and the updated instruction address are stored. However, when branching occurs, only the instruction address is replaced.

Machine Formats



OP=(05)₁₆



OP=(45)₁₆

Usercode Formats

(RR) BALR R,R

(RX) BAL R,DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : BALR Branch 4.80
 No Branch 3.84
 BAL 5.52
4-70 : BALR 2.0
 BAL 3.1

Interrupt Action

None

Notes

1. The P counter is stored without branching in the RR format when the R2 field is zero.
2. When a branch occurs, the leftmost eight-bit portion of the 32-bit P counter (ILC, CC, and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.
3. The contents of the register specified by the second address are unaltered.

6.5.3 BRANCH ON COUNTGeneral Description

The contents of the general register specified by the R1 field are algebraically decremented by one. The contents of the register are examined, and if the contents are zero, no branching occurs. If the contents are not zero, the instruction address in the P counter is replaced by the branch address and branching occurs.

Machine Formats

(RR)	OP	8	R1	4	R2	4	OP=(06) ₁₆				
(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(46) ₁₆

Usercode Formats

(RR) BCTR R,R
(RX) BCT R,DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : BCTR Branch 5.76
 No Branch 5.28
 BCT Branch 7.92
 No Branch 6.96
4-70 : BCTR 1.6
 BCT 2.8

Interrupt Action

None.

Notes

1. The subtraction executes as in fixed point arithmetic with all 32 bits participating.
2. An initial count of zero in the R1 field results in branching, because subtraction occurs before testing the contents of the register. If the value is zero, branching occurs and the result is minus one. To effect a no branch, the contents of the R1 field must be 1.
3. The contents of the registers specified by the second address are unaltered.

BCTR,BCT

4. When branching occurs, the leftmost eight-bit portion of the 32-bit P counter (ILC, CC, and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.
5. In the RR format, if the R2 field is zero, counting is performed without branching.
6. If a negative number appears in R1, an overflow condition occurs when this field is decremented. However, this overflow is ignored.
7. Overflow from a maximum negative number to a maximum positive number is ignored.

6.5.4 BRANCH ON INDEX HIGHGeneral Description

The operand specified by the third address (R3) is added to the operand specified by the first address (R1) and the sum is algebraically compared with the operand specified by the third address (R3), if R3 specifies an odd register. If R3 specifies an even register, the sum is algebraically compared with R3 + 1. If the sum is low or equal, branching does not occur and the next instruction is executed. If the sum is high, the instruction address in the P counter is replaced by the branch address and branching occurs.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(86) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

BXH R,R,DB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : Branch 11.60
 No Branch 11.12
 4-70 : 3.1

Interrupt Action

None.

Notes

1. The sum replaces the operand specified by the first address (R1) regardless of the comparison. The sum replaces (R1) after the comparison has been made.
2. Overflow is not recognized.
3. The contents of the register specified by R3 or R3 + 1 are unaltered.
4. When a branch occurs, the leftmost eight-bit positions of the 32-bit P counter (ILC, CC, and mask) are unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

BXLE

6.5.5 BRANCH ON INDEX LOW OR EQUAL

General Description

The operand specified by the third address (R3) is added to the operand specified by the first address (R1) and the sum is algebraically compared with the operand specified by the third address (R3), if R3 specifies an odd register. If R3 specifies an even register, the sum is algebraically compared with R3 + 1. If the sum is high, branching does not occur and the next instruction in sequence is executed. If the sum is low or equal, the instruction address in the P counter is replaced by the branch address and branching occurs.

Machine Format

(RS)	OP	8	R1	4	R3	4	B2	4	D2	12	OP=(87) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

BXLE R,R,DB

Condition Code

Unchanged.

Timing (μ secs)

4-50 : 11.6 (Branch or No Branch)
4-70 : 3.1

Interrupt Action

None.

Notes

1. The sum replaces the operand specified by the first address (R1) regardless of the comparison. The sum replaces (R1) after the comparison has been made.
2. Overflow is not recognized.
3. The contents of the register specified by R3 or R3 + 1 are unaltered.
4. When a branch occurs, the leftmost eight-bit positions of the 32-bit P counter (ILC, CC, and mask) are unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

6.5.6 EXECUTEGeneral Description

The instruction in the location specified by the second address is modified by the contents of the register specified by the first address. Then, the modified instruction is executed and control is returned to the instruction following the Execute instruction.

Machine Format

(RX)	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(44) ₁₆
------	----	---	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

EX R,DXB

Condition Code

May be set by the instruction being modified and executed.

Timing (μsecs)

4-50 : 6.96 + object instruction execution time

4-70 : 2.3

Interrupt Action

Address error:

Addressing.

Specification.

Notes

1. Bits 8-15 of the subject instruction are "OR"ed with bits 24-31 of the register specified by the first address (R1).
2. If R1 is zero, no modification takes place.
3. The ILC is set to two (length of the Execute) and the P counter is set to the address of the instruction following the Execute instruction.
4. The contents of R1 and the subject instruction in main store are unaltered.
5. Interrupts are inhibited until the subject instruction has been completed.
6. When the subject instruction is a successful branching instruction, the P counter is updated by the branch address.

6.5.7 INTERRUPT ACTION

Interrupts can occur as a result of an Execute instruction only. The interrupt conditions are as follows:

Address Error

Addressing:-

An address error interrupt occurs when the branch address of an Execute instruction is outside the main store for the particular installation, or if an Execute instruction is attempted to perform another Execute instruction. The operation is suppressed and the condition code, registers, and main store are unaltered.

Specification:-

An address error interrupt occurs if the branch address of an Execute instruction is not on a halfword boundary. The operation is suppressed and the condition code, registers, and main store are unaltered.

6.6 PRIVILEGED INSTRUCTIONS

6.6.1 INTRODUCTION

The instructions described in this section are called privileged instructions and can only be executed if the non-privileged mode bit (bit position 15 in the Interrupt Status register) for the current state is zero.

In addition to the standard privileged instruction set, inclusion of the store protect and/or the direct control optional features cause additional privileged instructions to be added.

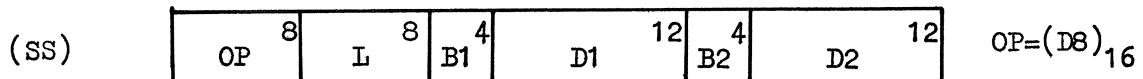
LSP

6.6.2 LOAD SCRATCH PAD

General Description

Operands from main store, starting with the storage location specified by the second address, are loaded in the scratch-pad store starting at the location specified by the first address.

Machine Format



Usercode Format

LSP DLB, DB

Condition Code

Unchanged except when the P counter in scratch-pad store is loaded.

Timing (usecs)

4-50 : 7.20 + 2.88R

4-70 : 1.6 + 0.54R

Interrupt Action

Privileged operation.

Address error:

Addressing.

Specification.

Notes

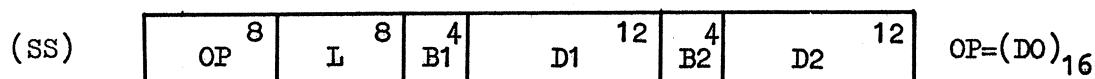
1. The L field provides an eight-bit count specifying the number of scratch-pad store locations to be loaded. An initial count of zero specifies one word to be loaded.
2. The first address specifies scratch-pad store words 0 to 127 by the seven rightmost bits of the address. The bits to the left of the seven-bit address must be zero.
3. The second address must be on a word boundary. (This is a program restriction.)
4. The processor uses utility registers in scratch-pad store to execute this instruction. If these registers are included in the range of this instruction results are unpredictable.

6.6.3 STORE SCRATCH PAD

General Description

Operands from the scratch-pad store, starting with the location specified by the first address, are stored in main store locations, starting with the location specified by the second address.

Machine Format



Usercode Format

SSP DLB, DB

Condition Code

Unchanged

Timing (usecs)

4-50 : 7.20 + 2.88R

4-70 : 1.1 + 0.54R

Interrupt Action

Privileged operation.

Address error:

Addressing.

Specification.

Protection.

Notes

1. The L field provides an eight-bit count specifying the number of scratch-pad store locations to be stored. An initial count of zero specifies one word to be stored.
2. The first address specifies scratch-pad store words 0 to 127 by the seven rightmost bits of the address. The bits to the left of the seven-bit address must be zero.
3. The second address must be on a word boundary. (This is a program restriction.)

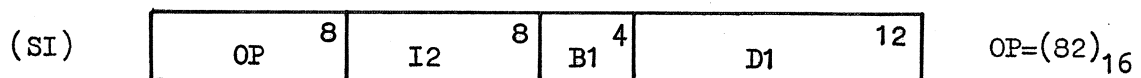
PC

6.6.4 PROGRAM CONTROL

General Description

This instruction specifies the termination of program execution in the current state, and the initiation of another state under control of the immediate byte in the I2 field. The address computed from the B1/D1 address components of the instruction is stored in the P counter of the state being terminated (bit positions 8-31).

Machine Format



Usercode Format

PC DB, I

Condition Code

The condition code indicators of the state being terminated are preserved in the state's P counter. The condition code in the P counter of the initiated state is then used to set the condition code indicators.

Timing (μsecs)

4-50 : 7.44

Interrupt Action

Privileged operation.

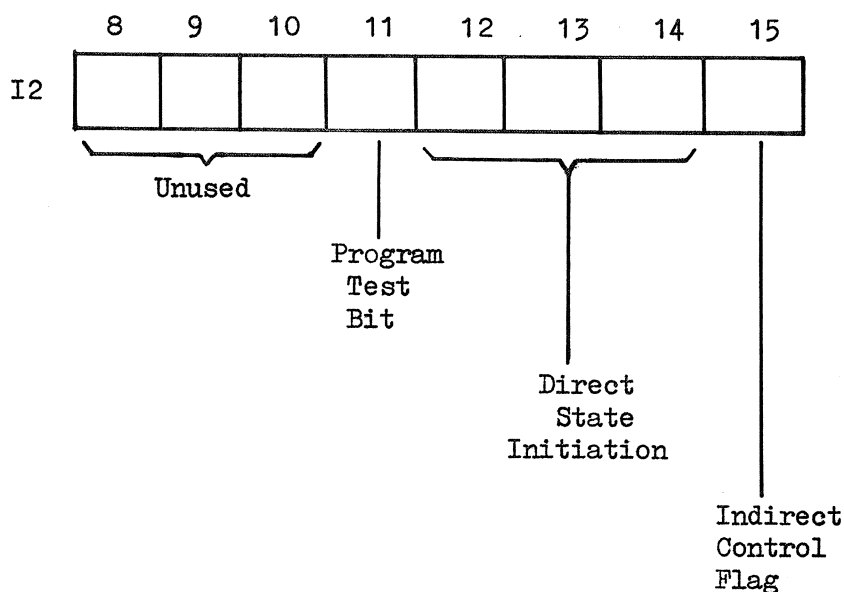
Address error:

Addressing.

Specification.

Note

1. The immediate byte in the I2 field of the instruction is divided into four subfields as follows:



Bits 8 to 10 are unused. The three bit unused portion must be zero.

Bit 11 is the program test bit. If bit 11 = 1, the program test mode is initiated. The program test interrupt bit is set in the Interrupt Flag register of the initiated state.

The scan of the Interrupt Flag register in the initiated state is delayed until after the first instruction of the initiated state is executed, at which time the scan is made in normal priority.

If bit 11 = 0, the program test mode is not initiated.

Bits 12 to 14 are the direct state initiation bits. The three-bit direct state initiation codes that may be specified are as follows:

- 000 - Go to Machine Condition State P4.
- 001 - Go to Interrupt Control State P3.
- 010 - Go to Interrupt Response State P2.
- 011 - Go to Processing State P1.

Programming Note: The leftmost bit of the three-bit direct state initiation field must be zero. (This is a programming restriction.)

Bit 15 is the indirect control flag bit. If indirect state control is specified (bit 15 = 1), the three-bit direct state initiation field is ignored. The three-bit interrupted state identifier (ISI), which indicates the last state interrupted, specifies the state to be initiated. This information is contained in the Interrupt Status register of the state being terminated.

If bit 15 = 0, direct state initiation is used.

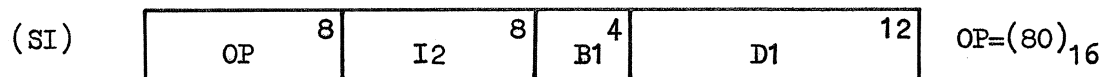
IDL

6.6.5 IDLE

General Description

This instruction effects an idle mode within the processor by continuously branching back to itself.

Machine Format



Usercode Format

IDL DB, I

Condition Code

Unchanged.

Timing (usecs)

4-50 : 6.0

Interrupt Action

Privileged operation.

Notes

1. When this instruction is operating with the I field zero, the Idle light of the console is on.
2. Any interrupt occurring while the idle mode is in effect is taken (if permitted via the Interrupt Mask register).
3. The B1 and D1 fields of this instruction must be zero.
4. For normal programming, the I field must be zero. For maintenance programming, bits within the I field, have the following meaning:

 Bit 15 = 1-set alarm inhibit.
 Bit 14 = 1-reset alarm inhibit.
 Bit 13 = 1-set inhibit simultaneity.
 Bit 12 = 1-reset-inhibit simultaneity.

6.6.6 DIAGNOSE

General Description

The purpose of this privileged instruction is to provide a means for facilitating maintenance techniques on the Central Processor. It is provided mainly for engineering purposes and cannot be used to detect program errors.

Although this instruction is included in the manual for the sake of completeness, it will not normally be required by the programmer, and thus no further description is given.

SDV

6.6.7 START DEVICE

General Description

The contents of the general register specified by B1 are added to the D1 field. The resultant sum identifies the channel and device to which the instruction applies. These are specified by bit positions 21 to 31 of the sum. The I-field is not used and must be zeros.

The channel address word in main store location 72 contains the protection key to be used and the address of the first channel command word. The channel command word designated by the channel address word specifies the operation to be performed, the main store area to be used, and the action to be taken when the operation is completed. The condition code indicates the result of the instruction.

Machine Format

(SI)	<table><tr><td>OP⁸</td><td>I2⁸</td><td>B1⁴</td><td>D1¹²</td></tr></table>	OP ⁸	I2 ⁸	B1 ⁴	D1 ¹²	OP=(9C) ₁₆
OP ⁸	I2 ⁸	B1 ⁴	D1 ¹²			

Usercode Format

SDV DB, I

Condition Code

0 - input/output operation initiated and channel proceeding with execution.

1 - status bits stored in scratch-pad store.

2 - busy or interrupt pending.

3 - inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Timing (μsecs)

4-50 : Multiplexor 33.36 + CRT*
Selector 27.60 + CRT*

4-70 : 0.9

* CRT = Channel Response Time (average 2 μs)

Interrupt Action

Privileged operation.

Notes

1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No.1
0	1	0	Selector No.2
0	1	1	Selector No.3
1	0	0	Selector No.4
1	0	1	Selector No.5
1	1	0	Undesignated
1	1	1	Selector No.6

Bit positions 24 to 31 specify one of 256 possible devices.

2. The standard device byte and the channel status byte stored by the previous input/output instruction in scratch-pad store is destroyed if the condition code at the completion of the Start Device instruction is 0 or 1.
3. Status storage (channel status byte and standard device byte), if required, occurs before the Start Device instruction terminates.
4. Condition Code 0 is set under the following conditions:
 - a) The device control electronics and the device specified are available.
 - b) The Start Device instruction specifies a Sense command to a device that is inoperable.
5. Condition Code 1 indicates that either the channel status byte or the standard device byte has been stored in the channel registers in the scratch-pad store for the specified channel.

The channel status byte is stored under the following conditions:

- a) A parity error occurs while accessing the Channel Address Word (CAW) or a Channel Command Word (CCW). The channel control check bit in the channel status byte is set.
- b) The Store Protection feature is not installed and the key in the CAW is not zero. The program check bit in the channel status byte is set.
- c) The main store address specified in the CAW is not on a double word boundary. The program check bit in the channel status byte is set.

- d) The main store address in the CCW specifies an address outside the available store for the system. The program check bit in the channel status byte is set.

The standard device byte is stored under the following conditions:

- a) The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.
- b) The Start Device instruction specifies a command which is other than a Sense command and the addressed device is inoperable. The device inoperable bit in the standard device byte is set.
- c) The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek to a random access device). The device busy bit and the device end bit in the standard device byte are set.

6. Condition Code 2 is set under the following conditions:

- a) A selector channel is specified that is busy.
- b) A selector channel is specified that has an interrupt pending (termination or external device request).
- c) The multiplexor channel is specified and it is operating in burst mode.
- d) The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device.
- e) The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.
- f) A burst mode operation is directed to the multiplexor and there is a termination interrupt pending on one of the attached device control electronics.

7. Condition Code 3 is set under the following conditions:

- a) A selector channel is specified that is not in the system.
- b) The specified device control electronics is inoperable.

8. If the condition code is 1, 2 or 3 the input/output operation is not initiated.

9. Parity errors that occur while fetching the CAW or CCW or that occur after the input/output operation has been initiated do not cause a machine check interrupt. A channel interrupt occurs and the program is notified of the error via the channel status byte.
10. If the first CCW is a Transfer in Channel command the Start Device instruction terminates and the condition code is set to 0. However, the specified device control electronics recognizes this command as an illegal operation and causes a channel interrupt to occur.

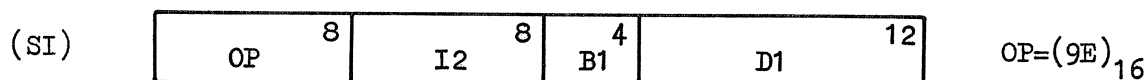
HDV

6.6.8 HALT DEVICE

General Description

The contents of the general register specified by B1 are added to the D1 field, and the resultant sum identifies the channel to be halted. The channel is specified by bit positions 21 to 23 of the sum. If a multiplexor is specified, bit positions 24 to 31 of the sum identify the device to be halted. The I field is not used and must be zeros. Buffered devices operating off-line, and independent of the channel/device control electronics, cannot be stopped by using this instruction. The condition code specifies the results of the instruction.

Machine Format



Usercode Format

HDV DB, I

Condition Code

- 0 - not busy.
- 1 - standard device byte stored in the scratch-pad store.
- 2 - termination accepted.
- 3 - inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Timing (μsecs)

4-50 : Multiplexor 10.32 + CRT
Burst 5.52 + CRT
Selector 6.00 + CRT

where CRT = Channel Response Time (average 2 μs)

4-70 : 0.9

Interrupt Action

Privileged operation.

Notes

1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No.1
0	1	0	Selector No.2
0	1	1	Selector No.3
1	0	0	Selector No.4
1	0	1	Selector No.5
1	1	0	Undesignated
1	1	1	Selector No.6

Bit positions 24 to 31 specify one of 256 possible devices.

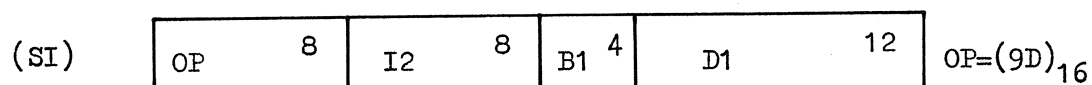
2. If a device operating on a selector channel is to be halted, the device number does not have to be specified.
3. The channel address word in main store location 72 and the channel command word are not used by this instruction.
4. A termination interrupt occurs when any input/output operation is terminated. Status bits are stored in the scratch-pad store when the termination interrupt occurs.
5. All five flags in Channel Command Register II (CCR-II) are cleared if the Halt Device instruction is accepted. Therefore, upon termination, the incorrect length counter in the channel status byte is set if the count is not zero.
6. A Halt Device instruction that specifies a multiplexor channel that is operating in the burst mode must specify a device that is operating in the burst mode.
7. Condition Code 0 is set under the following conditions:
 - a) The device control electronics or the device specified on the multiplexor channel is not busy. No termination is required.
 - b) A selector channel or the multiplexor channel operating in burst mode is specified and it is not busy. No termination is required.
 - c) The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. No termination is required.
8. Condition Code 1 indicates that the specified device is on the multiplexor channel and that the standard device byte has been stored in the channel registers in the scratch-pad store for the multiplexor channel. The channel status byte is never stored.

The standard device byte is stored under the following conditions:

- a) The specified device indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.
 - b) The specified device is busy but the device control electronics is not busy (that is, tape rewinding). The device busy bit in the standard device byte is set.
 - c) The specified device is inoperable. The device inoperable bit in the standard device byte is set.
9. Condition Code 2 is set under the following conditions:
- a) A selector channel is specified that is busy.
 - b) The multiplexor channel is specified and it is operating in the burst mode.
 - c) The multiplexor channel is specified and the addressed device control electronics and device are busy.
10. Condition Code 3 is set under the following conditions:
- a) A selector channel is specified that it is not in the system.
 - b) The specified device control electronics is inoperable.
11. Status storage (standard device byte), if required, occurs before the Halt Device instruction terminates.

6.6.9 TEST DEVICEGeneral Description

The contents of the general register specified by B1 are added to the D1 field. The resultant sum identifies the channel and device to which the instruction applies. These are specified by bit positions 21 to 31 of the sum. The I-field is not used and must be zeros. The condition code specifies the results of the instruction.

Machine FormatUsercode Format

TDV DB, I

Condition Code

- 0 - available.
- 1 - standard device byte stored in the scratch-pad store.
- 2 - busy or interrupt pending.
- 3 - inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Timing (μsecs)

4-50 : Multiplexor 8.40 + CRT
 Selector 8.88 + CRT

where CRT = Channel Response Time (2 μs average)

Interrupt Action

Privileged operation.

Notes

1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No.1
0	1	0	Selector No.2
0	1	1	Selector No.3
1	0	0	Selector No.4
1	0	1	Selector No.5
1	1	0	Undesignated
1	1	1	Selector No.6

Bit positions 24 to 31 specify one of 256 possible devices.

2. The channel address word in main store location 72 and the channel command word are not used by this instruction.
3. Status storage (standard device byte), if required, occurs before the Test Device instruction terminates.
4. Condition Code 0 is set if the device control electronics and the device are available.

Note: There may be pending interrupts on the multiplexor channel that would prohibit a burst mode operation to be initiated.

5. Condition Code 1 indicates that the standard device byte has been stored in the channel registers in the scratch-pad store for the specified channel. The channel status byte is never stored by this instruction.

The standard device byte is stored under the following conditions:

- (a) The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.
 - (b) The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek to a random access device). The device busy bit in the standard device byte is set.
 - (c) The specified device is inoperable. The device inoperable bit in the standard device byte is set.
6. Condition Code 2 is set under the following conditions:
 - (a) A selector channel is specified that is busy.

- (b) A selector channel is specified that has an interrupt pending (termination or external device request.)
- (c) The multiplexor channel is specified and it is operating in burst mode.
- (d) The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device.
- (e) The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.

7. Condition Code 3 is set under the following conditions:

- (a) A selector channel is specified which is not in the system.
- (b) The specified device control electronics is inoperable.
- (c) A device is specified that is not in the system.

CKC

6.6.10 CHECK CHANNEL

General Description

The contents of the general register specified by B1 are added to the D1 field, and the resultant sum identifies the input/output channel to be tested. This is specified by bit positions 21 to 23 of the sum. Only the channel is tested.

Machine Format

(SI)	<table><tr><td>OP</td><td>8</td><td>I2</td><td>8</td><td>B1</td><td>4</td><td>D1</td><td>12</td></tr></table>	OP	8	I2	8	B1	4	D1	12	OP=(9F) ₁₆
OP	8	I2	8	B1	4	D1	12			

Usercode Format

CKC DB, I

Condition Code

- 0 -(a) The specified selector channel is not busy and has no interrupts pending.
(b) The specified multiplexor channel is not operating in the burst mode.
- 1 - The specified selector channel has an external device request interrupt pending.
- 2 -(a) The specified selector channel is busy or has a terminating interrupt pending.
(b) The specified multiplexor is operating in the burst mode.
- 3 - A selector channel is specified that is not in the system.

Timing (μsecs)

4-50 : Multiplexor 5.52
Selector 6.48

Interrupt Action

Privileged operation.

Notes

- 1. The address portion of this instruction specifies the channel to be tested as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No.1
0	1	0	Selector No.2
0	1	1	Selector No.3
1	0	0	Selector No.4
1	0	1	Selector No.5
1	1	0	Undesignated
1	1	1	Selector No.6

2. The channel address word in main store location 72 and the channel command word are not used by this instruction.
3. The device address (bit positions 24 to 31 of the sum) is not used by this instruction.
4. Status bits (channel status byte and standard device byte) are not stored in the scratch-pad store by this instruction.
5. Current operations proceeding in the specified channel are unaffected by this instruction.

ISK

6.6.11 INSERT STORAGE KEY

General Description

The storage key of the 2,048-byte main store block, which is located at the address contained in the general register specified by the second address, is inserted in the general register specified by the first address.

Machine Format

(RR)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(09) ₁₆
OP	8	R1	4	R2	4			

Usercode Format

ISK R, R

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 5.28

Interrupt Action

Privileged operation.

Address error:

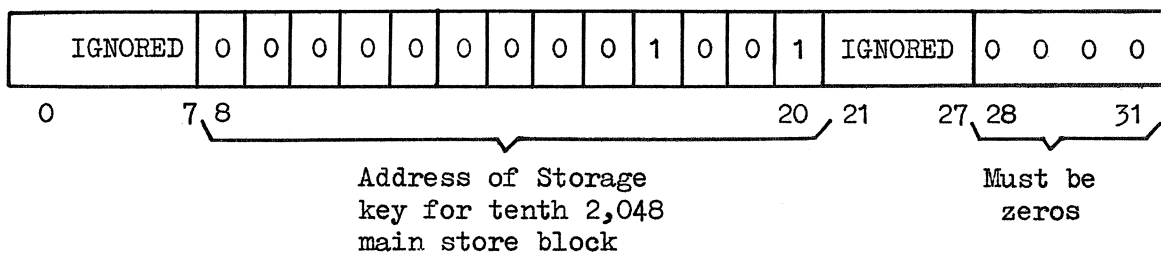
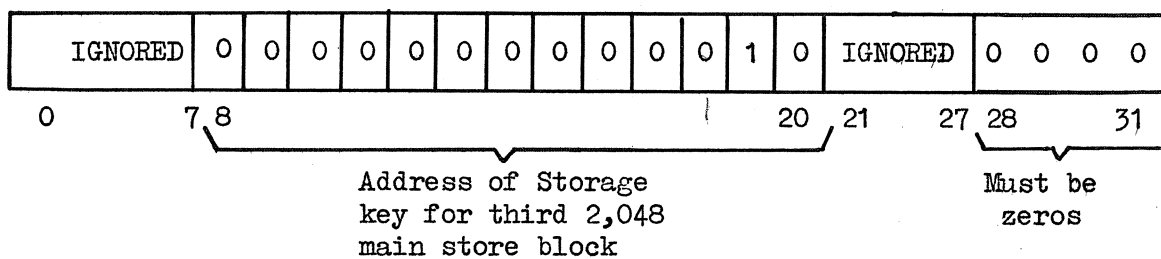
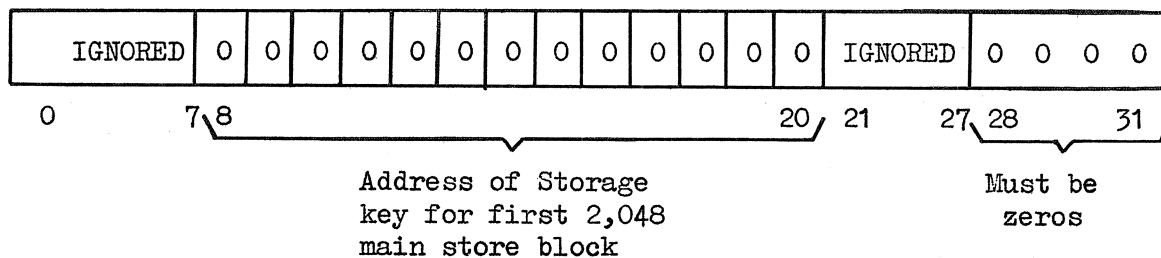
Addressing.
Specification.

Operation code trap (if the store protect feature is not installed).

Notes

1. The general register specified by the second address (R2) contains the location of the 2,048-byte main store block in bits 8 to 20. Bits 0 to 7 and 21 to 27 are ignored. Bits 28 to 31 must be zero.
2. When the four-bit storage key is inserted into bits 24 to 27 of the general register specified by the first address, bits 0 to 23 are unaltered and bits 28 to 31 are made zero.
3. The address of the storage key for a specific 2,048-byte main store block is specified in R2 by a binary count as shown in the following examples:

Storage Key Address in R2



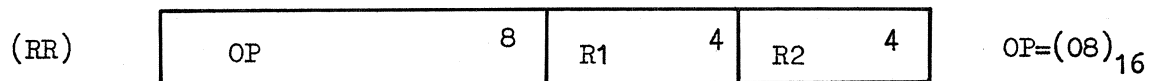
SSK

6.6.12 SET STORAGE KEY

General Description

The storage key of a 2,048-byte main store block located at the address contained in the general register specified by the second address is set according to the value contained in the register specified by the first address.

Machine Format



Usercode Format

SSK R,R

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 5.28

Interrupt Action

Privileged operation.

Address error:

Addressing.
Specification.

Operation code trap (if the store protection feature is not installed).

Notes

1. Bits 8 to 20 of the register specified by the second address contain the location of the storage key for a 2,048-byte main store block. Bits 0 to 7 and 21 to 27 are ignored. Bits 28 to 31 must be zero.
2. Bits 24 to 27 of the general register specified by the first address contain the four-bit storage key to be assigned. Bits 0 to 23 and 28 to 31 are ignored.
3. The address of the storage key for a specific 2,048-byte main store block is specified in R2 by a binary count (see examples under Insert Storage Key description).

6.6.13 WRITE DIRECTGeneral Description

The eight-bit byte specified by the first address is accessed and transmitted to all units via the Static Out lines. The eight-bit I field specifies the Signal Out lines to be pulsed. The Static Out lines remain as specified until the next Write Direct instruction.

Machine Format

(SI)	OP	8	I2	8	B1	4	D1	12	OP=(84) ₁₆
------	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

WRD DB,I

Condition Code

Unchanged.

Timing (μsecs)

4-50 : To be supplied later.

Interrupt Action

Privileged operation.

Address error:

Addressing.

Operation code trap (if Direct Control option is not installed).

Notes

- Each trunk has only one Signal Out line and is pulsed according to the following pattern:

<u>I-Field</u>	<u>Trunk(s) Pulsed</u>
Bit 0 = 1	Six
Bit 1 = 1	Five
Bit 2 = 1	Four
Bit 3 = 1	Three
Bit 4 = 1	Two
Bit 5 = 1	One
Bit 6 = 0	Reserved (Must be zero)
Bit 7 = 0	Reserved (Must be zero)

WRD

More than one I-Field bit may be set to 1 providing pulses for sending over more than one direct control trunk. This permits sending the same byte to all processors connected to the transmitting processor.

2. A processor cannot Write Direct to itself. The I-Field bit associated with the transmitting processor must always be reset to zero. (This is a programming restriction.)

6.6.14 READ DIRECTGeneral Description

The eight-bit I field specifies one of up to five possible sets of Direct Control trunks to be sampled. The sampled eight-bit byte is transferred to the main store location specified by the first address from the Static In lines.

Machine Format

(SI)	OP	8	I2	8	B1	4	D1	12	OP=(85) ₁₆
------	----	---	----	---	----	---	----	----	-----------------------

Usercode Format

RDD DB,I

Condition Code

Unchanged.

Timing (μsecs)

4-50 : To be supplied later.

Interrupt Action

Privileged operation.

Address error:

Addressing.

Protection.

Operation code trap (if Direct Control option is not installed).

Notes

- Each of the six Direct Control trunks has a set of Direct In Lines which are sampled according to the following pattern:

<u>I-Field</u>	<u>Trunk Sampled</u>
Bit 0 = 1	Six
Bit 1 = 1	Five
Bit 2 = 1	Four
Bit 3 = 1	Three
Bit 4 = 1	Two
Bit 5 = 1	One
Bit 6 = 0	Unused (Must be zero)
Bit 7 = 0	Unused (Must be zero)

The program must specify only one I-Field bit set to 1, otherwise results of the instruction are unpredictable.

RDD

2. A processor cannot Read Direct to itself. The I-Field bit associated with the receiving processor must always be reset to zero. (This is a programming restriction.)
3. This instruction may be prolonged by the presence of a HOLD signal. If so, timer updating may be skipped. However, input/output servicing will not be affected.

6.6.15 INTERRUPT ACTION

The following interrupt conditions can occur as a result of a privileged instruction:

Address Error

Addressing : An address error interrupt occurs when an address specifies a location outside the available main store of the particular installation. The operation is terminated at the point of error. The result data and condition code, if produced, are unpredictable. If the address of an instruction is invalid, the operation is suppressed.

Specification : An address error interrupt occurs when:

1. A Load Scratch Pad or Store Scratch Pad instruction specifies a first or second address which is not on a word boundary.
2. Bits 28 to 31 of the second operand of a Set Storage Key or Insert Storage Key instruction are not zero.
3. The store protection feature is not installed and the protection key in the Interrupt Status register for the current program state is not zero.
4. The Program Control instruction specifies an instruction address which is not on a halfword boundary.

In these error interrupt conditions, the operation is suppressed. The data in the main store and registers is unchanged.

Protection : An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is terminated. The result data is unpredictable. (This interrupt can occur only if the store protection feature is installed.)

Privileged Operation

A privileged operation interrupt occurs if execution of any privileged instruction is attempted and the non-privileged mode bit (bit position 15 in the Interrupt Status register) for the current state is 1. The operation is suppressed and the condition code, registers, and main store are unaltered.

Operation Code Trap

An operation code trap interrupt occurs under the following conditions:

1. The store protection feature is not installed and an attempt to execute a Set Storage Key or Insert Storage Key instruction is made.
2. The direct control feature is not installed and an attempt to execute a Write Direct or Read Direct instruction is made.

6.7 PROCESSOR STATE CONTROL INSTRUCTIONS

6.7.1 INTRODUCTION

There are two control instructions that can be used in the Processing State (P1). These instructions are Supervisor Call, and Set Program Mask. These instructions can also be executed in any other state.

The Supervisor Call instruction enables the program to switch from any state to the Interrupt Control State (P3). Through this operation a program in any processor state can communicate with and initiate the Interrupt Control State (P3) programs.

The Set Program Mask instruction permits the user to specify whether or not the program is to be interrupted for any of the following errors:

1. significance error.
2. exponent underflow.
3. decimal overflow.
4. fixed-point overflow.

The execution of the Set Program Mask instruction causes the condition code and program mask bits in the P counter of the state in which the system is operating to be set to the value specified by the instruction. This instruction always changes the condition code.

6.7.2 INTERRUPT ACTION

No error interrupts can occur as a result of using the instructions in this section. The Supervisor Call instruction causes an interrupt, but this interrupt is the desired result of its execution.

SVC

6.7.3 SUPERVISOR CALL

General Description

The R1 and R2 fields provide an interruption code and this code is placed into the rightmost byte of the Interrupt Status Register (ISR) of the program state in which this instruction is issued. The supervisor call interrupt flag bit (priority 21) is set in the Interrupt Flag register and a program interrupt may occur depending on the associated mask bit in the Interrupt Mask Register of the current state.

Machine Format

(modified RR)

OP	8	L	8
----	---	---	---

OP = (OA)₁₆

Usercode Format

SVC L

Condition Code

Unchanged.

Timing (μsecs)

4-50 : 2.88

Interrupt Action

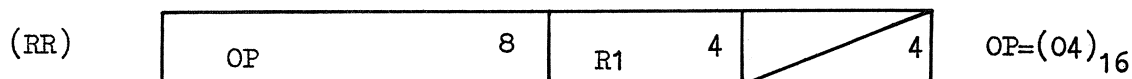
None

Notes

1. If a higher priority interrupt is honoured upon executing this instruction, the flag bit (priority 21) will be set and the Supervisor Call byte stored in the ISR so that when it is honoured the results are independent of any higher priority interrupts.
2. Under normal circumstances, SVC will occur only within library macros and so is of no interest to the programmer. The values of L have no intrinsic significance but are merely identifiers to show which Supervisor routine is to be given control, and are thus determined by which operating regime is being used.

6.7.4 SET PROGRAM MASKGeneral Description

Bits 2-7 of the general register specified by the first address establish new program masks and condition code setting for the current program state.

Machine FormatUsercode Format

SPM, R

Condition Code

The condition code is set according to bits 2 and 3 of the general register specified by R1 as follows:

Condition Code Setting

2	3	Result
0	0	Set condition code 0 (zero).
0	1	Set condition code 1.
1	0	Set condition code 2.
1	1	Set condition code 3.

Timing (μsecs)

4-50 : 2.88

Program Mask

The program mask is set according to bits 4-7 of the general register specified by R1 as follows:

Program Mask Setting

Bit	Result
4	Fixed-point overflow.
5	Decimal overflow.
6	Exponent underflow.
7	Significance error.

Note

The contents of the P-counter and the register specified by the first address are unaltered.

LTER,LTDR

6.8 FLOATING POINT INSTRUCTIONS

6.8.1 LOAD AND TEST

General Description

The operand in the floating-point register specified by the second address is loaded into the floating-point register specified by the first address. The sign and magnitude of the loaded operand determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(32) ₁₆
OP	8	R1	4	R2	4			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(22) ₁₆
OP	8	R1	4	R2	4			

Usercode Formats

(RR Short) LTER R, R
(RR Long) LTDR R, R

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - result mantissa is greater than zero.
- 3 - not used.

Timing (μsecs)

4-50 : LTER (Short) 6.00
 LTDR (Long) 8.16
4-70 : LTER (Short) 1.0
 LTDR (Long) 1.6

Interrupt Action

Address error:

Specification.

Notes

1. If R1 and R2 are equal, the operation is equivalent to a test without data movement.
2. The operand specified by the second address is unaltered.
3. Short operands do not alter the low-order half of the register specified by the first address.

6.8.2 LOAD COMPLEMENTGeneral Description

The operand in the floating-point register specified by the second address is loaded into the floating-point register specified by the first address and the sign is changed to the opposite value. The sign and magnitude of the loaded operand determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(33) ₁₆
OP	8	R1	4	R2	4			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(23) ₁₆
OP	8	R1	4	R2	4			

Usercode Formats

(RR Short) LCER R, R
(RR Long) Lcdr R, R

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - result mantissa is greater than zero.
- 3 - not used.

Timing (μsecs)

4-50 LCER (Short) 6.0
LCDR (Long) 8.16
4-70 LCER (Short) 1.1
LCDR (Long) 1.8

Interrupt Action

Address error:

Specification.

Notes

1. The exponent and mantissa are unaltered.
2. Short operands do not alter the low-order half of the registers specified by the first address.

LPER LPDR

6.8.3 LOAD POSITIVE

General Description

The operand in the floating-point register specified by the second address is loaded into the floating-point register specified by the first address and the operand sign is made plus.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(30) ₁₆
OP	8	R1	4	R2	4			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(20) ₁₆
OP	8	R1	4	R2	4			

Usercode Formats

(RR Short) LPER R, R
(RR Long) LPDR R, R

Condition Code

- 0 - result mantissa is zero.
- 1 - not used.
- 2 - result mantissa is greater than zero.
- 3 - not used.

Timing (μsecs)

4-50 : LPER (Short) 5.52
 LDER (Long) 7.68
4-70 : LPER (Short) 1.1
 LPDR (Long) 1.6

Interrupt Action

Address error:

Specification.

Notes

1. The exponent and mantissa are unaltered.
2. Short operands do not alter the low-order half of the register specified by the first address.

6.8.4 LOAD NEGATIVE

General Description

The operand in the floating-point register specified by the second address is loaded into the floating-point register specified by the first address and the operand sign is made minus.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(31) ₁₆
OP	8	R1	4	R2	4			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(21) ₁₆
OP	8	R1	4	R2	4			

Usercode Formats

(RR Short) LNER R, R
(RR Long) LNDR R, R

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - not used.
- 3 - not used.

Timing (µsecs)

4-50 : LNER (Short) 5.52
LNDR (Long) 7.68
4-70 : LNER (Short) 1.1
LNDR (Long) 1.6

Interrupt Action

Address error:

Specification.

Notes

1. The exponent and mantissa are unaltered.
2. Short operands do not alter the low-order half of the register specified by the first address.

AER,AE
ADR,AD

6.8.5 ADD NORMALIZED

General Description

The operand specified by the second address is added to the operand in the floating-point register specified by the first address. The normalized sum is loaded into the register specified by the first address. The sign and magnitude of the sum determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3A) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7A) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2A) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6A) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) AER R, R
(RX Short) AE R, DXB
(RR Long) ADR R, R
(RX Long) AD R, DXB

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - result mantissa is greater than zero.
- 3 - result exponent overflows.

Timing (μsecs)

4-50 : AER 16.08
 AE 19.20
 ADR 22.63
 AD 27.69
4-70 : AER 2.4
 AE 3.6
 ADR 2.9
 AD 4.0

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Significance error.

Exponent overflow.

Exponent underflow.

Notes

1. To perform normalized addition, the computer must scale the two operands. Scaling consists of comparing the exponents of the two operands. If they do not agree, the mantissa with the smaller exponent operand is shifted right. Its exponent is increased by one for each digit right-shifted, until the two exponents agree. Then, the mantissae are added algebraically to form an intermediate sum. If an overflow carry occurs, the intermediate sum is right-shifted one digit and its exponent is increased by one. If this causes an overflow, an exponent overflow interrupt condition occurs.

For short operands, the intermediate sum consists of seven hexadecimal digits and a possible carry. The low-order digit is the guard digit which is retained from the mantissa which is shifted right. Only one guard digit participates in the mantissa addition. The guard digit is zero if no shift occurs.

For long operands, the intermediate sum consists of fourteen hexadecimal digits and a possible carry. No guard digit is retained.

2. After addition, the intermediate sum is left-shifted until all high-order zero hexadecimal digits have been eliminated. The vacated low-order digits are made zero and the exponent is decremented by one for each zero digit shifted. If no left-shift takes place, the intermediate sum is truncated to the proper mantissa length. If the exponent underflows (exceeds -64) during normalization, the floating-point number is made true zero and an exponent underflow interrupt occurs.
3. No normalization is performed when the intermediate sum is zero. The sum mantissa is unaltered and a significance error interrupt occurs. If a significance error interrupt is prohibited by the interrupt mask, the quantity is made true zero and a significance error interrupt does not occur.
4. Initial operands need not be in normalized form.
5. The sign of the sum is determined by the rules of algebra. A zero sum is always plus.
6. Short operands do not alter the low-order halves of the registers specified by the address fields.

6.8.6 ADD UNNORMALIZED

General Description

The operand specified by the second address is added to the operand in the floating-point register specified by the first address. The unnormalized sum is loaded into the register specified by the first address. The sign and magnitude of the loaded sum determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3E) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7E) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2E) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6E) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) AUR R, R
(RX Short) AU R, DXB
(RR Long) AWR R, R
(RX Long) AW R, DXB

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - result mantissa is greater than zero.
- 3 - result exponent overflows.

Timing (μsecs)

4-50 : AUR 15.84
AU 18.96
AWR 21.77
AW 26.81
4-70 : AUR 2.3
AU 3.4
AWR 2.8
AW 3.9

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Exponent overflow.

Significance.

Notes

1. The Add Unnormalized is similar to the Add Normalized, except that the sum is not normalized by this instruction and exponent underflow cannot occur.

SER,SE
SDR,SD

6.8.7 SUBTRACT NORMALIZED

General Description

The operand specified by the second address is subtracted from the operand in the floating-point register specified by the first address. The normalized difference is loaded into the register specified by the first address. The sign and magnitude of the difference determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3B) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7B) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2B) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6B) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) SER R, R
(RX Short) SE R, DXB
(RR Long) SDR R, R
(RX Long) SD R, DXB

Condition Code

0 - result mantissa is zero.
1 - result mantissa is less than zero.
2 - result mantissa is greater than zero.
3 - result exponent overflows.

Timing (μsecs)

4-50 : SER 16.08
 SE 19.20
 SDR 22.63
 SD 27.69
4-70 : SER 2.4
 SE 3.6
 SDR 2.9
 SD 4.0

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Significance error.

Exponent overflow.

Exponent underflow.

Notes

1. The Subtract Normalized is the same as the Add Normalized, except that the sign of the second operand is changed to the opposite value before addition. A zero difference is always positive.

6.8.8 SUBTRACT UNNORMALIZED

General Description

The operand specified by the second address is subtracted from the operand in the floating-point register specified by the first address. The unnormalized difference is loaded into the register specified by the first address. The sign and magnitude of the difference determine the condition code.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3F) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7F) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2F) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6F) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) SUR R, R
 (RX Short) SU R, DXB
 (RR Long) SWR R, R
 (RX Long) SW R, DXB

Condition Code

- 0 - result mantissa is zero.
- 1 - result mantissa is less than zero.
- 2 - result mantissa is greater than zero.
- 3 - result exponent overflows.

Timing (μsecs)

4-50 : SUR 15.84
 SU 18.96
 SWR 21.77
 SW 26.81
 4-70 : SUR 2.3
 SU 3.4
 SWR 2.8
 SW 3.9

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Significance error.

Exponent overflow.

Notes

1. Subtract Unnormalized differs from Subtract Normalized only in that the difference is not normalized before it is loaded into the result register.
2. Exponent underflow cannot occur.

CER,CE CDR,CD

6.8.9 COMPARE

General Description

The operand in the floating-point register specified by the first address is algebraically compared to the operand specified by the second address. The result determines the condition code.

Machine Formats

(RR Short)	OP 8	R1 ⁴	R2 ⁴		OP=(39) ₁₆
(RX Short)	OP 8	R1 ⁴	X2 ⁴	B2 ⁴	D2 12 OP=(79) ₁₆
(RR Long)	OP 8	R1 ⁴	R2 ⁴		OP=(29) ₁₆
(RX Long)	OP 8	R1 ⁴	X2 ⁴	B2 ⁴	D2 12 OP=(69) ₁₆

Usercode Formats

(RR Short) CER R, R
(RX Short) CE R, DXB
(RR Long) CDR R, R
(RX Long) CD R, DXB

Condition Code

- 0 - operands are equal.
- 1 - operand specified by the first address is less than the one specified by the second address.
- 2 - operand specified by the first address is greater than the one specified by the second address.
- 3 - not used.

Timing (μsecs)

4-50 : CER 12.24
CE 15.36
CDR 18.48
CD 23.52
4-70 : CER 1.6
CE 2.7
CDR 2.2
CD 3.2

Interrupt Action

Address error:
Addressing (RX format).
Specification.

Notes

1. Comparison takes into account the sign, exponent, and mantissa of each number. Exponent inequality is not decisive for magnitude determination since the mantissæ may have different numbers of leading zeros. The operands are scaled, as in Subtract Normalized, and if the mantissa of each operand is zero, the numbers are considered equal regardless of the sign and exponent.
2. Both operands are unaltered.

HER,HDR

6.8.10 HALVE

General Description

The operand in the floating-point register specified by the second address is divided by two. The quotient is loaded into the floating-point register specified by the first address.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(34) ₁₆
OP	8	R1	4	R2	4			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(24) ₁₆
OP	8	R1	4	R2	4			

Usercode Formats

(RR Short) HER R, R
(RR Long) HDR R, R

Condition Code

Unchanged.

Timing (μsecs)

4-50 : HER 6.00
 HDR 8.16
4-70 : HER 1.1
 HDR 1.8

Interrupt Action

Address error:

Specification.

Notes

1. The difference between the Halve instruction and a Divide instruction with a divisor of two, is that no normalization and no zero mantissa testing takes place. The sign and exponent are unaltered and the mantissa is shifted right one bit.
2. Short operands do not alter the low-order half of the result register.

6.8.11 STOREGeneral Description

The contents of the floating-point register specified by the first address are stored in the main store location specified by the second address.

Machine Formats

(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(70) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(60) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RX Short) STE R, DXB

(RX Long) STD R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : STE 8.40
STD 11.28

4-70 : STE 1.8
STD 2.2

Interrupt Action

Address error:

Addressing.

Specification.

Protection.

Notes

1. The first operand is unaltered.
2. Short operands do not alter the low-order half of the register specified by the second address.

MER,ME MDR,MD

6.8.12 MULTIPLY

General Description

The operand in the floating-point register specified by the first address is multiplied by the operand specified by the second address. The normalized product is loaded into the register specified by the first address.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3C) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7C) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2C) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6C) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) MER R, R
(RX Short) ME R, DXB
(RR Long) MDR R, R
(RX Long) MD R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : MER 46.40
ME 49.42
MDR 181.51
MD 186.55
4-70 : MER 4.9
ME 6.2
MDR 10.1
MD 11.5

Interrupt Action

Address error.

Addressing (RX format).
Specification.

Exponent overflow.

Exponent underflow.

Notes

1. The exponents of the two operands are added, and the sum is reduced by 64 to form an intermediate exponent. The mantissae are normalized as described in the Add Normalize instruction, and multiplied to form an intermediate mantissa. The intermediate mantissa is then normalized (reducing its exponent by one for each digit left shifted) to form the final product.
2. The sign of the product is determined by the rules of algebra.
3. If the product mantissa is zero, the final product is made true zero.
4. If the final product exponent is greater than 127, an exponent overflow interrupt occurs.
5. If final product exponent is less than zero, an exponent underflow interrupt occurs.
6. For short operands, the low-order half of the register specified by the first address is used in the calculation of the intermediate mantissa. The product mantissa has the full 14 digits as in the long format and the two low-order digits are always zero.

DER,DE
DDR,DD

6.8.13 DIVIDE

General Description

The operand (dividend) in the floating-point register specified by the first address is divided by the operand divisor specified by the second address. The normalized quotient is stored in the register specified by the first address. The remainder is not retained

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(3D) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(7D) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(2D) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(6D) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) DER R, R
(RX Short) DE R, DXB
(RR Long) DDR R, R
(RX Long) DD R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : DER 79.88
DE 83.00
DDR 275.68
DD 280.27
4-70 : DER 8.3
DE 9.3
DDR 17.6
DD 18.6

Interrupt Action

Address error:

Addressing (RX format).
Specification.

Exponent overflow.

Exponent underflow.

Divide error.

Notes

1. The exponents of the two operands are subtracted and the difference is increased by 64 to form an intermediate exponent. The mantissae are normalized as described in the Subtract Normalize instruction, and divided to form the mantissa of the intermediate quotient. The intermediate exponent and mantissa are normalized to form a final quotient.
2. If the dividend (first operand) is zero, the quotient is made true zero.
3. If the divisor (second operand) is zero, a divide error interrupt occurs.
4. The sign of the quotient is determined by the rules of algebra.
5. If the final quotient exponent is less than zero, the final quotient is made true zero and an exponent underflow interrupt occurs.
6. If the final quotient exponent exceeds 127, an exponent overflow interrupt occurs.
7. For short operands, the low-order halves of the registers are unaltered.

LER,LE LDR,LD

6.8.14 LOAD

General Description

The operand specified by the second address is loaded into the floating-point register specified by the first address.

Machine Formats

(RR Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(38) ₁₆				
OP	8	R1	4	R2	4							
(RX Short)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(78) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			
(RR Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>R2</td><td>4</td></tr></table>	OP	8	R1	4	R2	4	OP=(28) ₁₆				
OP	8	R1	4	R2	4							
(RX Long)	<table><tr><td>OP</td><td>8</td><td>R1</td><td>4</td><td>X2</td><td>4</td><td>B2</td><td>4</td><td>D2</td><td>12</td></tr></table>	OP	8	R1	4	X2	4	B2	4	D2	12	OP=(68) ₁₆
OP	8	R1	4	X2	4	B2	4	D2	12			

Usercode Formats

(RR Short) LER R, R
 (RX Short) LE R, DXB
 (RR Long) LDR R, R
 (RX Long) LD R, DXB

Condition Code

Unchanged.

Timing (μsecs)

4-50 : LER 6.72
 LE 9.84
 LDR 8.64
 LD 13.68
 4-70 : LER 0.9
 LE 1.8
 LDR 1.6
 LD 1.9

Interrupt Action

Address error:

Addressing (RX format).
 Specification.

Notes

1. The operand specified by the second address is unaltered.
2. Exponent overflow, underflow, or lost significance cannot occur.
3. The low-order half of the register specified by the first address is unaltered when short operands are used.

6.8.15 INTERRUPT ACTION

The following interrupt conditions can occur as a result of a floating-point instruction.

Address Error

Addressing:- An address error interrupt occurs when an address in the RX instruction format specifies a location outside the available main store. The operation is terminated at the point of error. The result data and the condition code (if affected) are unpredictable.

Specification:- An address error interrupt occurs if a short operand is not located on a word boundary or a long operand is not located on a double-word boundary. An address error interrupt also occurs if a floating-point register other than 0, 2, 4 or 6 is specified. The instruction is suppressed. The condition code, the data in main store, and the registers remain unchanged. Address restrictions do not apply to the X2, B2 and D2 components of the instruction.

Protection:- An address error interrupt occurs when the protection key and the storage key of the result location do not match. The operation is suppressed. The condition code, the data in the main store, and the registers remain unchanged. (This interrupt can only occur if the store protection feature is installed).

Significance Error

A significance error interrupt occurs when the result mantissa of an add or subtract operation is zero. A program interrupt occurs if the significance error mask bit in the Interrupt Mask Register of the current state is set to 1. The operation is completed, the exponent is unaltered, and the interrupt is taken. If the significance error mask bit is zero, the interrupt is prohibited and the operation is completed by setting the result to true zero (zero sign, zero exponent, and zero mantissa). In either case, the condition code is set to zero.

Divide Error

A divide error interrupt occurs if division by zero is attempted.

Exponent Overflow

An exponent overflow interrupt occurs when the result exponent overflows and the mantissa is not zero. The operation is terminated and the result data is unpredictable. Addition and subtraction set the condition code to 3. Multiplication and division do not affect the condition code setting.

Exponent Underflow

An exponent underflow interrupt occurs when the result exponent is less than zero and the result mantissa is not zero. The operation is completed by setting the result to true zero (zero sign, zero exponent, and zero mantissa). Addition and subtraction set the condition code to zero. Multiplication and division do not affect the condition code setting.

6.8.16 Normalization

A floating-point number with a mantissa containing a non-zero, high-order, hexadecimal digit is called a normalized number. An unnormalized number has one or more high-order hexadecimal zero digits in the mantissa. To change an unnormalized number into a normalized number, the mantissa is shifted to the left until the high-order digit is non-zero. Then, the exponent is decremented by the number of digits shifted.

Generally, normalization occurs when the intermediate arithmetic result is changed to the final result. However, in multiplication and division operations, normalization occurs before the arithmetic process.

Floating-point operations are performed with, or without, normalization. Most operations are performed in only one way; however, addition and subtraction may be performed either way as specified.

When normalization is not performed, high-order zeros in the result mantissa are not eliminated. Depending on the original operands, the result may, or may not, be normalized.

Initial operands in both normalized and unnormalized operations need not be in normalized form. Because normalization takes place on hexadecimal digits, the three high-order bits of a normalized mantissa can be zero.

7. Translator Instructions

Just as machine instructions are used to request the computer to perform a sequence of operations during program execution time, so Translator instructions are requests to the Translator to perform certain operations during the translation. Translator instruction statements, in contrast to machine instruction statements, do not cause machine instructions to be included in the object code module. Some such as DS and DC generate no machine instructions but, by affecting the location counter, cause storage areas to be set aside for constants and other data. Others such as EQU and SPACE are effective only at translation time; they generate nothing in the translated module and do not affect the location counter.

The following is a list of all Translator instructions.

ALOG
AOPTN
CNOP
CCW
COM
CSECT
DC
DROP
DS
DSECT
EJECT
END
ENTRY
EQU
EXTRN
ISEQ
LTORG
MCALL
MPRTY
ORG
PRINT
SPACE
START
TITLE
USING

7.1 EXTENDED MNEMONIC CODES

For the convenience of the programmer, the Translator provides extended mnemonic codes, which allow conditional branches to be specified mnemonically as well as through the use of the BC machine instruction. These extended mnemonic codes specify both the machine branch instruction and the condition on which the branch is to occur. The codes are not part of the set of machine instructions, but are translated into the corresponding operation and condition combinations. The allowable extended mnemonic codes are shown in the following table:

Extended Codes	Meaning	Operand Format	Equivalent BC Mask (decimal)
B	Branch Unconditional	DXB	15 0123
BR	Branch Unconditional (R→R)	R	15
NOP	No Operation	DXB	0
NOPR	No Operation (R→R)	R	0
BH	Branch on High	DXB	2 2
BL	Branch on Low	DXB	4 1
BE	Branch on Equal	DXB	8 0
BNH	Branch on Not High	DXB	13 0,1,3
BNL	Branch on Not Low	DXB	11 0,2,3
BNE	Branch on Not Equal	DXB	7 1,2,3
BO	Branch on Overflow	DXB	1 3
BP	Branch on Plus	DXB	2 2
BM	Branch on Minus	DXB	4 1
BZ	Branch on Zero	DXB	8 0
BO	Branch if Ones	DXB	1 3
BM	Branch if Mixed	DXB	4 1
BZ	Branch if Zeros	DXB	8 0

7.2 BASE REGISTER USAGE

Whenever an address is specified in a machine-instruction statement, the Translator may be required to determine whether there is a register suitable for use as a base. Two instructions are provided to inform the Translator which base registers are available and what value they are assumed to hold. These instructions do not result in the generation of any machine language code by the Translator.

7.2.1 USE BASE REGISTER

General Description

The USING instruction indicates to the Translator the general register(s) that are available for use as base registers, and the value(s) that the register(s) are assumed to contain at object time.

Format

The format of the USING instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	USING	Expressions of the form: V, r1, r2, r3, r4, r5, ---, rx

Specification Rules

Name Field : Not used.

Operation Field : USING

Operand Field : Contains the base address value and the register(s) to be assigned. Operand V must be an absolute or relocatable expression, literals are not permitted. The values that are assumed in the base registers r1 to rx will be in the form of V, V+4096, V+8192, V+12288, etc. The expressions used to indicate the registers r1 to rx must be absolute and have values between 0-15. Any number of registers may be specified in one USING statement.

Notes

1. The USING instruction may be used as often as needed and at any point in the source module to indicate to the Translator changes in the register(s) or their value(s).
2. Since the USING instruction does not actually load the assigned general registers, it is the user's responsibility to ensure that the register(s) are loaded with the value(s) specified in the USING instruction.
3. General register 0 may be used as a base register only if operand V is a relocatable expression in the first control section of the source module, and if used, General register 0 is specified in the operand r1.
4. The Translator will assume register 0 to contain a zero value if specified and that all other registers stated in the USING instruction will be in increments of 4096.
5. Nonrelocatable modules result from usage of register 0. The program may be made relocatable at a later time by replacing the register 0 notation in the USING statement, loading the register with a relocatable value and retranslating the module.

USING

6. Whenever an address is specified in a Usercode instruction, the Translator scans a list of current USING Statements to find registers covering the portion of the program containing the address, that is, specifying a label in the same control section as the address and which occurs not more than 4095 bytes before the address. If two or more registers are found, the one giving the smallest displacement is selected, if two or more registers give the same displacement, the highest number register is selected.

7.2.2 DROP BASE REGISTER

General Description

The DROP instruction specifies previously available registers that may no longer be used as base registers.

Format

The format of the DROP instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	DROP	Absolute Expression(s) of the form r1, r2, r3, r4, r5 --- rx

Specification Rules

Name Field : Not used.

Operation Field : DROP

Operand Field : Contains absolute expressions indicating the register(s) to be dropped. Any number of registers may be dropped with one DROP instruction.

Notes

1. It is not necessary to DROP a general register before changing its value with another USING instruction.
2. A dropped register may be made available again through another USING instruction.

EQU

7.3 EQUATING SYMBOLS WITH EXPRESSIONS

The EQU instruction allows a symbol to be defined by assigning to it the attributes of an expression. It does not result in the generation of any machine language code by the Translator.

7.3.1 EQUATE SYMBOL

General Description

The EQU instruction is used to define a symbol by assigning to it the attributes of an expression in the operand field.

Format

The format of the EQU instruction statement is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A symbol	EQU	An expression

Specification Rules

Name Field : Any valid symbol

Operation Field : EQU

Operand Field : The expression in the operand field may be either absolute or relocatable. Any symbols appearing in the expression must be previously defined.

Note

The symbol in the name field is given the same attributes as the expression in the operand field.

7.4 DEFINING STORAGE AND CONSTANTS

The DC instruction is used to provide constant data in storage. It results in the generation of constants in storage by the Translator. The DS instruction is used to reserve storage area.

7.4.1 DEFINING CONSTANTSGeneral Description

The DC (Define Constant) instruction is used to provide constant data in storage. One or more of a variety of constants may be specified in a single DC instruction.

Data in the form of a character, hexadecimal, binary, decimal, fixed-point, or floating-point constant can be entered into a program through the use of this instruction.

System 4-50, through its wide variety of data representation, normally requires descriptive information regarding (1) the type of constant to be assembled, (2) the number of times the constant is to be duplicated, (3) the amount of storage to reserve for this constant, and (4) the actual constant itself to be stated in the Operand field of the DC instruction. Data constants are generally called constants, unless they are created from storage addresses, in which case they are called address constants.

Format

The format of the DC instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A symbol or blank	DC	A single operand in the following format:

The operand consists of four subfields. The first three describe the constant (some or all may be omitted, depending on the constant), and the fourth subfield provides the constant or constants. Note that more than one constant may be specified in the fourth subfield for some types of constants. No spaces may occur within any of the subfields (unless provided as characters within a character constant), nor may they occur between the subfields of an operand.

The subfields of a DC operand are written in the following sequence.

1	2	3	4
Duplication factor	Type	Modifier	Constant(s)

Specification Rules

Name Field : Contains a symbol or is left blank. If a symbol is used to name the constant, it is assigned the leftmost byte (after alignment) address of the first or only constant. The length attribute depends on two things; the type of constant being defined and the presence of a length specification. Implied lengths are assumed for the various constant types in the absence of a length specification. If more than one constant is defined or a duplication factor is specified the length

attribute is the length in bytes (specified or implied) of the first constant. Boundary alignment also varies according to the type of constant being specified and the presence of a length specification.

Some constants are aligned at various word boundaries (half, full or double) in the absence of a length specification. If length is specified, no boundary alignment occurs for such constants. Bytes that must be skipped in order to align the field at the proper boundary are not considered to be part of the constant.

Any bytes skipped in aligning constants are not zeroed.

Operation Field : DC

Operand Field : The operand field consists of four subfields as follows:

Note Literal Definitions

All subsequent specifications are applicable to writing literals, the only differences being:

- 1) The literal is preceded by an = sign.
- 2) The duplication factor may not be zero.
- 3) S - type address constants may not be specified.

Operand Subfield 1 : Duplication Factor

The duplication factor may be omitted. If specified, it causes the constant(s) to be generated the number of times indicated by the factor. The factor must be specified by a decimal self defining term. The duplication factor is applied after the constant has been developed into its proper format.

A duplication factor of zero is permitted except in a literal and can be used to force the Location Counter to a full-word, half-word or double-word boundary when specified with the appropriate field-type.

Operand Subfield 2 : Type

The type subfield defines the type of constant being specified. From this type specification, the Translator determines how it is to interpret the constant and translate it into the appropriate machine format. The type is specified by a letter code as shown below. If no type is specified C is assumed. Further information about these constants is provided under "Operand Subfield 4" below.

Operand Subfield 3 : Modifier

This field specifies the type of modification to be applied against the constant. Three types of modifiers are available:

1. Ln = Length Modifier. This type specifies the number of bytes of the store to be assigned to the constant. This modifier can be used with all types of constants. No boundary alignment is provided when a length modifier is used.
2. Sn = Scale Modifier. This type of modifier specifies the amount of internal scaling to be performed on the constant. The scale modifier can be used with fixed point (F, H) and floating-point (E, D) constants only.

The modifier is written as Sn, where n is a decimal value. The decimal value may be preceded by a sign; if none is present, a plus sign is assumed. The maximum values for scale values are summarized in the table at the end of chapter 7.4.1.

Scale Modifier for Fixed-Point Constant: The scale modifier specifies the power of two by which the constant must be multiplied after it has been converted to its binary representation. Just as multiplication of a decimal number by a power of 10 causes the decimal point to move, multiplication of a binary number by a power of two causes the binary point to move. This multiplication has the effect of moving the binary point away from its assumed position in the binary field; the assumed position being to the right of the rightmost position.

Thus, the scale modifier indicates either of the following: (1) the number of binary positions to be occupied by the fractional portion of the binary number, or (2) the number of binary positions to be deleted from the integral portion of the binary number. A positive scale of x shifts the integral portion of the number x binary positions to the left, thereby reserving the rightmost x binary positions for the fractional portion. A negative scale shifts the integral portion of the number right, thereby deleting rightmost integral positions. If a scale modifier does not accompany a fixed-point constant containing a fractional part, the fractional part is lost. For example, if the decimal portion of the number 987.65 is to be retained, the statement DCF S8'987.65' is required. The statement DCF'987.65' without the scale modifier S8 retains only the integral portion, 987, of the number.

Where positions are lost because of scaling (or the lack of scaling), rounding occurs in the leftmost bit of the last portion. The rounding is reflected in the rightmost position saved.

Scale Modifier for Floating-Point Constant: Only a positive scale modifier can be used with a floating-point constant. This modifier indicates the number of hexadecimal positions that the fraction is to be shifted to the right. Note that this shift amount is in terms of hexadecimal positions, each of which is four binary positions. (A positive scaling actually indicates that the point is to be moved to the left. However, a floating-point constant is always converted to a

fraction, which is hexadecimally normalized. The point is assumed to be at the left of the leftmost position in the field. Because the point cannot be moved left, the fraction is shifted right.)

Thus, scaling that is specified for a floating-point constant provides an assembled fraction that is unnormalized, that is, contains hexadecimal zeros in the leftmost positions of the fraction. When the fraction is shifted, the exponent is adjusted accordingly to retain the correct magnitude. When hexadecimal positions are lost, rounding occurs in the leftmost hexadecimal position of the lost portion. The rounding is reflected in the rightmost hexadecimal position saved.

3. En = Exponent Modifier. This modifier denotes the power of 10 by which the constant is to be multiplied before its conversion to the proper internal format. The modifier is written as En where n is a decimal value. The decimal value may be preceded by a sign; if none is present, a plus sign is assumed. The maximum values for exponent modifiers are summarized in the table at the end of chapter 7.4.1.

This modifier is not to be confused with the exponent modifier of the constant itself as defined under Fixed Point Constants and Floating Point Constants, (see (f) and (g) below.) Both are denoted as En. The exponent modifier affects each constant in the operand, whereas the exponent written as part of the constant only pertains to that constant. Thus, a constant can be specified with an exponent of +2, and an exponent modifier of +5 can precede the constant. In effect, the constant has an exponent of +7.

Note: There is a maximum value, both positive and negative, (listed in Summary of Constants, below) for exponents. This applies both to exponent modifier and exponents specified as part of the constant, or to their sum if both are specified.

Operand Subfield 4 : Constant

This subfield supplies the constant (or constants) described by the subfields that precede it. A data constant (all types except A, Y, S and V) is enclosed by single quotation marks. An address constant (types A, Y, S and V) is enclosed by parentheses. For types F, H, E, D, P and Z, two or more constants may be specified; each must be separated by commas and the entire sequence of constants must be enclosed by single quotation marks. Thus, the format for specifying the constant(s) is one of the following:

Single Constant

'constant'
(constant)

Multiple Constants (F,H,E,D,P,Z type only)

'constant,...,constant'

All constant types except character (C), hexadecimal (X), binary (B), packed decimal (P), and zoned decimal (Z), are aligned on the proper boundary unless a length modifier is specified. In the presence of a length modifier, no boundary alignment is performed. If an operand specifies more than one constant, any alignment applies to the first

constant only. Thus, for an operand that provides five full-word constants, the first would be aligned on a full-word boundary, and the rest would automatically fall on full-word boundaries.

The total storage requirement of an operand is the product of the length, times the number of constants in the operand, times the duplication factor (if present), plus any bytes skipped for boundary alignment.

If an address constant contains a Location Counter reference, the Location Counter value that is used is the storage address of the first byte the constant will occupy. If an address constant is specified (and contains a Location Counter reference) with a duplication factor, the constant is duplicated with a varying Location Counter value, the Location Counter being stepped on by the length of the constant each time the constant is duplicated. If an address constant in a literal contains a Location Counter reference, the Location Counter value that is used is the storage address of the first byte of the instruction containing the literal. If an address constant in a literal is specified with a duplication factor and contains a Location Counter reference, the constant is duplicated with a varying Location Counter value, the Location Counter being stepped on by the length of the constant each time the constant is duplicated.

The subsequent text describes each of the constant types.

(a) Character Constant C - Type. Any of the 256 punch combinations may be specified in a character constant. A string of these is enclosed in single quotation marks and optionally preceded by a letter C.

Only one character constant may be specified per operand.

Special consideration must be given to representing quotation marks and ampersands as characters. Each single quotation mark or ampersand specified as a character in the constant must be represented by a pair of single quotation marks or ampersands. Only one single quotation mark or ampersand appears in storage. The maximum length of a character constant is 256 bytes. No boundary alignment is performed. Each character is translated into one byte. The double quotation marks and ampersands count as one character. If no length modifier is given the size in bytes of the character constant is equal to the number of characters in the constant. If a length modifier is provided, the result varies as follows. If the number of characters in the constant exceeds the specified length or is greater than the number which can be held on two cards, as many rightmost bytes as necessary are dropped; if the number of characters is less than the specified length, the excess rightmost bytes are filled with spaces.

(b) Hexadecimal Constant X - Type. A hexadecimal constant comprises of one or more of the hexadecimal digits which are 0 - 9 and A - F, enclosed by single quotation marks and preceded by a letter X.

Only one hexadecimal constant may be specified per statement.

The maximum length of a hexadecimal constant is 256 bytes (512 hexadecimal digits). No boundary alignment is performed.

Constants that contain an even number of hexadecimal digits are translated as one byte per pair of digits. If an odd number of digits is specified the leftmost byte has the leftmost four bits filled with a hexadecimal zero while the rightmost four bits contain the odd (first) digit.

If no length modifier is given, the implied length of the constant is half the number of hexadecimal digits in the constant (assuming that a hexadecimal zero is added to an odd number of digits). If a length modifier is given the constant is handled as follows. If the number of hexadecimal digits is too great to fit into the specified length, the necessary leftmost bytes are dropped; if the number of hexadecimal digits does not fill the specified length the necessary bytes are added to the left and filled with hexadecimal zeros.

(c) Binary Constant B - Type. A binary constant consists of a string of binary digits (1 and 0) enclosed in quotation marks and preceded by the letter B. Only one binary constant may be specified in a statement. Duplication and length modifier may be specified.

The maximum length of a binary constant is 256 bytes.

The implied length of a binary constant is the number of bytes occupied by the constant including any padding necessary. Padding or truncation takes place on the left. The padding bit used is 0.

(d) Decimal Constants P-Type and Z-Type

A decimal constant is written as a signed or unsigned decimal value. If the sign is omitted, a plus sign is assumed.

The decimal point may be written in or omitted from the constant. The placing of a decimal point in the definition does not affect the translation of the constant in any way because the decimal point is not translated with the constant.

P - Type. In packed decimal constants (P - type) each pair of decimal digits is translated into one byte. The rightmost digit and the sign are translated into the rightmost byte. The bit configuration for the digits is identical to the configurations for the hexadecimal digits 0-9.

If an even number of packed decimal digits is specified, one digit will be left unpaired, because the rightmost digit is paired with the sign. Therefore, in the leftmost byte, the leftmost four bits will be set to zeros and the rightmost four bits will contain the odd (first) digit.

Z - Type. In zoned decimal constants (Z - type) each decimal digit is translated into one byte. The translation is done according to the character set shown in Appendix 2. The rightmost byte contains the sign as well as the rightmost digit.

For both packed and zoned decimal constants, a plus sign is translated into the hexadecimal digit C, and a minus sign into the hexadecimal digit D.

The maximum length of a decimal constant is 16 bytes. No boundary alignment is performed. If no length modifier is given the implied length for either constant is the number of bytes the constant occupies. If a length modifier is given, the constant is handled as follows. If the constant requires fewer bytes than the length specified, the necessary number of bytes is added to the left. For zoned decimal constants, the decimal digit zero is placed in each added byte. For packed decimal constants, the bits of each added byte are set to zero.

If the constant requires more bytes than the length specified, the necessary number of leftmost digits is dropped.

(e) Address Constants A-Type, Y-Type, S-Type, V-Type.

An address constant is a storage address that is translated into a constant. An address constant, unlike other types of constant, is enclosed in parentheses. There are four types of address constants, A, Y, S and V.

An A-type constant normally holds the address in a full word, aligned on a full word boundary.

A Y-type constant normally holds the address in a half word aligned on a half word boundary.

An S-type constant holds the address in base and displacement form in a half word aligned on a half word boundary.

A V-type constant is used to reserve storage for the address of an external symbol and normally holds the address in a full word on a full word boundary.

A-Type. This constant is specified as an absolute, relocatable or complex relocatable expression, (the expression may be single or compound). The value of the expression is calculated as explained in chapter 3.4 with one exception; the maximum value of the expression may be $2^{31}-1$. The implied length of an A-type constant is four bytes, and the value is placed in the rightmost portion, the remaining bits being padded with zeros. Alignment is to a full word boundary, unless a length is specified. A length modifier of one to four bytes may be specified, in which case no alignment will occur.

Y-Type. A Y-type address constant has the characteristics and format of the A-type constant discussed above except for the following:

1. The constant is assembled as a half word and aligned to a half word boundary.
2. The implied length is two bytes.
3. A length modifier of one or two bytes may be used.

S-Type. The S-type address constant is used to store an address in base-displacement form.

The constant may be specified into two ways:

- 1) As an absolute or relocatable expression, for example S(BETA)
- 2) As two absolute expressions, the first of which represents the displacement value and the second, which is enclosed in inner parentheses, the base register, for example S(400(13)).

The address value represented by the expression in 1) will be broken down by the Translator into the proper base register and displacement value. An S-type constant is assembled as a half-word and aligned on a half-word boundary. The leftmost four bits of the assembled constant represent the base register designation, the remaining 12 bits the displacement value.

A length modifier of two bytes may be specified when no alignment will take place.

S-type address constants may not be specified as literals.

V-Type. This constant reserves storage for the address of an external symbol that is used in branching to other modules. It may be used to activate the automatic segmentation facilities bounded by Composer. See Section 2.4.1, Appendix Q, of the DOS Manual. The constant is specified as one relocatable symbol which need not be identified by an EXTRN statement. Whatever symbol is used is assumed to be an external symbol by virtue of the fact that it is supplied in a V-type address constant.

Note that specifying a symbol as the operand of a V-type constant does not constitute a definition of the symbol for this module.

The implicit length of a V-type constant is 4 bytes and alignment is to a full word. A length modifier of three or four bytes may be specified, in which case there will be no alignment. The value of the assembled constant will be zero until the module has been through Composer.

(f) Fixed-Point Constants (F- and H- Type)

A fixed-point constant is written as a decimal number, which may be followed by a decimal exponent if desired. The number can be an integer, a fraction, or a mixed number (that is, one with integral and fractional portions). The format of the constant is as follows:

1. The number is written as a signed or unsigned decimal value. The decimal point is placed before, within, or after the number, or it is omitted, in which case the number is assumed to be an integer. A positive sign is assumed if an unsigned number is specified. Unless a scale modifier accompanies a mixed number or fraction, the fractional portion is lost.
2. The exponent is optional. If specified, it is written immediately after the number as E_n , where n is an optionally signed decimal value specifying the exponent of the factor 10. The exponent can be in the range -85 to +75. If an unsigned exponent is specified, a plus sign is assumed. The exponent causes the value of the constant to be adjusted by

is assumed. The exponent causes the value of the constant to be adjusted by the power of 10 that it specifies before the constant is converted to its binary form.

The number is converted to its binary equivalent and is assembled as a full word or halfword, depending on whether the type is specified as F or H. It is aligned at the proper full word or halfword boundary if a length is not specified. An implied length of four bytes is assumed for a full word (F) and two bytes for a halfword (H). However, any length up to and including eight bytes can be specified for either type of constant by a length modifier, in which case no boundary alignment occurs.

Maximum and minimum values, exclusive of scaling, for fixed-point constants are:

LENGTH	MAX.	MIN.
8	$2^{63}-1$	-2^{63}
4	$2^{31}-1$	-2^{31}
2	$2^{15}-1$	-2^{15}
1	2^7-1	-2^7

The binary number occupies the rightmost portion of the field in which it is placed. The unoccupied portion (that is, the leftmost bits) is filled with the sign. That is, the setting of the bit designating the sign is the setting for the bits in the unused portion of the field. If the value of the number exceeds the length, the necessary leftmost bits are dropped. A negative number is carried in 2's complement form.

If the rightmost portion of the number must be dropped as a result of scale modifiers, rounding occurs. Any duplication factor that is present is applied after the constant is converted to its binary format and assembled into the proper number of bytes.

(g) Floating-Point Constants (E- and D-Type)

A floating-point constant is written as a decimal number, which is followed by a decimal exponent, if desired. The number can be an integer, a fraction, or a mixed number (that is, one with integral and fractional portions). The format of the constant is as follows:

1. The number is written as a signed or unsigned decimal value. The decimal point is placed before, within, or after the number, or it is omitted. If the decimal point is omitted, the number is assumed to be an integer. A positive sign is assumed if an unsigned number is specified.

2. The exponent is optional. If specified, it is written immediately after the number as E_n , where n is an optionally signed decimal value specifying the exponent of the factor 10. The exponent can be in the range -85 to +75. If an unsigned exponent is specified, a positive sign is assumed.

Machine format for a floating-point number is in two parts: the portion containing the exponent, which is sometimes called the characteristic, followed by the portion containing the fraction, which is sometimes called the mantissa. Therefore, the number specified as a floating-point constant must be converted to a fraction before it can be translated into the proper format. For example, the constant 27.35E2 represents the number 27.35 times 10 to the 2. Represented as a fraction, it would be .2735 times 10 to the 4, the exponent having been modified to reflect the shifting of the decimal point. The exponent is also affected by the presence of an exponent modifier, as explained under the heading Operand Subfield 3: Modifiers. Thus, the exponent is also altered before being translated into machine format. After the constant is converted into the proper exponent and fraction, each is translated into its binary equivalent and arranged in machine floating-point format.

The translated constant is placed in a full word or a double word, depending on whether the type is specified as E or D. It is aligned at the proper full-word or double-word boundary if a length is not specified. An implied length of four bytes is assumed for a full word (E) and eight bytes is assumed for a double word (D). However, any length up to and including eight bytes can be specified for either type of constant by a length modifier, in which case no boundary alignment occurs.

Within the portion of the floating-point field allocated to the fraction, the hexadecimal point is assumed to be to the left of the leftmost hexadecimal digit and the fraction occupies the leftmost portion of the field. The fraction is normalized (no leading hexadecimal zeros), unless scaling is specified. If the rightmost portion of the fraction must be dropped because of length or scale modifiers, rounding will occur. Negative fractions are carried in true representations, not in the 2's complement form.

SUMMARY OF CONSTANTS

Type	Implied Length (Bytes)	Alignment	Length Modifier Range (Lm)	Specified by	Number of Constants per Operand	Exponent Modifier Range (Em)	Scale Modifier Range (Sm)	Truncation/ Padding Side
C	as needed	byte	1 to 256	characters	one			right
X	as needed	byte	1 to 256	hexadecimal digits	one			left
B	as needed	byte	1 to 256	binary digits	one			left
F	4	word	1 to 8	decimal digits	multiple	-85 to +75	-187 to +346	left
H	2	half word	1 to 8	decimal digits	multiple	-85 to +75	-187 to +346	left
E	4	word	1 to 8	decimal digits	multiple	-85 to +75	0 to 2L-2 ₍₁₎	right
D	8	double word	1 to 8	decimal digits	multiple	-85 to +75	0 to 2L-2 ₍₁₎	right
P	as needed	byte	1 to 16	decimal digits	multiple			left
Z	as needed	byte	1 to 16	decimal digits	multiple			left
A	4	word	1 to 4	any expression	one			left
V	4	word	3 or 4	relocatable symbol	one			left
S	2	half word	2 only	one absolute or relocatable expression or two absolute expressions: exp (exp)	one			left
Y	2	half word	1 to 4	any expression	one			left

(1) L is length of constant. Negative scaling is not permitted.

7.4.2 DEFINING STORAGE

General Description

The DS (Define Storage) instruction is used to reserve areas of storage and to assign names to those areas.

Format

The format of the DS instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Symbol or blank	DS	one operand in a format described below

Name field : The attributes of the symbol, if any, are defined exactly as for the DC instruction.

Operation field : DS

Operand field : Any valid DC operand may be written as a DS operand. In addition, the data, subfield 4 may be omitted.

The data, subfield 4 will in no case be assembled into the reserved storage, nor will the reserved storage be set to zeros.

Note The use of a zero duplication factor in the examples DS OH, DS OF, DS OD will cause the Translator to **force** alignment of the location counter to half, full, double-word boundaries respectively and is an alternative to the CNOP instruction.

7.5 PROGRAM SECTIONING AND CONTROL INSTRUCTIONS

This section describes the instructions necessary for the Translator to provide facilities for symbolically linking separately translated programs or program sections. It also describes the instructions which give the programmer control over the Location Counter value, the placing of literals and checking of statement sequence.

START

7.5.1 START TRANSLATION

General Description

The START instruction may be used to give a name to the first (or only) control section of a module. There may be only one START instruction in a module. It is also used to specify a tentative starting location for the module. The START instruction may not be preceded by any type of statement that affects, or depends upon, the setting of the Location Counter. Control sections are discussed in more detail in Chapter 3.7.

Format

The format of the START instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A symbol or blank	START	A self-defining term or blank

Specification Rules

Name Field: If a symbol names the START instruction, the symbol is established as the name of the control section. If not, the control section is considered to be unnamed.

The symbol in the Name Field is a valid relocatable symbol whose value represents the address of the first byte of the control section. It has a length attribute of one.

Operation Field: START

Operand Field: The Translator uses the self-defining value specified by the operand as the tentative starting location of the program. If the operand is omitted, the Translator sets the tentative starting location of the program to zero. If necessary, the Translator will round the specified value up to a multiple of 8.

Note (5A Regime only)

To use the 5A system in the most efficient way, the operand in a START instruction should be 12504 (decimal) or above. This ensures that Job Control (located before address 12500) is not overwritten and so need not be fetched between jobs.

END

7.5.2 END TRANSLATION

General Description

The END instruction terminates the translation of a module. It may also designate a point in the module or in a separately translated module to which control may be transferred after the module is loaded. The END instruction must always be the last statement in the source module.

Format

The format of the END instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used.	END	A relocatable expression or blank.

Specification Rules

Name Field : Not used.

Operation Field : END

Operand Field : Any expression whose value specifies the point to which control is transferred when loading the object module (normally, the first instruction statement in the module).

Note

If the operand contains an external symbol, only a single-term, relocatable expression is allowed.

7.5.3 IDENTIFY CONTROL SECTION

General Description

The CSECT instruction identifies the beginning or the continuation of a control section. All statements that follow the CSECT instruction are translated as part of that control section until another statement that identifies a different control section is encountered. Control sections are discussed in more detail in Chapter 3.7.

Format

The format of the CSECT instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A symbol or blank	CSECT	Not used. See below.

Specification Rules

Name Field : The symbol entered in the Name Field establishes the name of the control section. If omitted, the section is considered to be unnamed. The symbol in the Name Field is a valid relocatable symbol, the value of which is the address of the first byte of the control section. It has a length attribute of one.

A previously named CSECT may be resumed at any time by another CSECT statement with the same name. Thus statements from different control sections may be interspersed. Text is output in the same physical order as the input source, but the associated loading address will place resumptions of CSECTS in contiguous store.

If the CSECT is unnamed, that is, the name field is blank, this can be resumed by a CSECT statement with a blank name field. However, any ENTRY names defined in this CSECT will be flagged; this is because Composer must be presented with CSECTS with unique names and an unnamed CSECT is potentially not unique.

Operation Field: CSECT

Operand Field: Not used. Any entry appearing in this field will be treated as a comment.

DSECT

7.5.4 IDENTIFY DUMMY SECTION

General Description

A dummy section represents a control section that is translated but is not part of the object module. A dummy section is a convenient means of describing the layout of an area of storage without actually reserving the storage. (It is assumed that the storage is reserved either by some other part of this module or else by another module). The DSECT instruction identifies the beginning or resumption of a dummy section. More than one dummy section may be defined per module, but each must be named. Control sections are discussed in more detail in Chapter 3.7.

Format

The format of the DSECT instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A symbol	DSECT	Not used. See below.

Specification Rules

Name Field : The symbol entered in the Name Field establishes the name of the dummy control section. The symbol in the Name Field is a valid relocatable symbol whose value is the address of the first byte of the dummy section. A length attribute of one is assigned.

Operation Field : DSECT.

Operand Field : Not used. Any entry appearing in this field will be treated as a comment.

Additional Information

Program statements belonging to dummy sections can be interspersed throughout the module or can be written as a unit. In either use, the appropriate DSECT instruction should precede each set of statements. When multiple DSECT instructions with the same name are encountered, the first instruction is considered to initiate the dummy section and the rest to continue it. Symbols that name statements in a dummy section may be used in USING instructions. Therefore, they may be used in program elements (e.g. machine instructions and data definitions) that specify storage addresses.

Note: A symbol that names a statement in a dummy section may be used in an A-type address constant only if it is paired with another symbol from the same dummy section.

Dummy Section - Location Assignment

A Location Counter determines the relative locations of named program elements in a dummy section. The Location Counter is always set to zero at the beginning of the dummy section and the location values assigned to symbols that name statements in the dummy section relate to the initial statement in the section.

Addressing Dummy Sections

The programmer may wish to describe the format of an area whose storage location is not determined until the module is executed. He can describe the format of the area in a dummy section, and he can use symbols defined in the dummy section as the operands of machine instructions. To refer to the storage area, he does the following :

1. Provides a USING statement that specifies a general register, which the Translator can assign to the machine instructions as a base register, and that specifies a value from the dummy section, which the Translator assumes is contained in the base register.
2. Ensures that the same register is loaded with the actual address of the storage area.

The values assigned to symbols defined in a dummy section relate to the initial statement of the section. Thus, all machine instructions that refer to names defined in the dummy section will, at execution time, refer to storage locations that relate to the address loaded into the register.

ENTRY

7.5.5 IDENTIFY ENTRY POINT SYMBOL

General Description

The ENTRY instruction identifies a linkage symbol that is defined in this module but may be used by some other module.

Format

The format of the ENTRY instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	ENTRY	A relocatable symbol that also appears as a statement name.

Specification Rules

Name Field : Not used.

Operation Field : ENTRY

Operand Field : Any symbolic name to identify an entry point to the Translator.

Notes

1. The symbol used in the Operand Field may be used by other modules as operand entries.
2. ENTRY statements may not contain a symbol defined in an unnamed control section or in a dummy section.
3. The name of a control section need not be identified by an ENTRY instruction when another module uses it as an entry point. The Translator will automatically include the section names in the Control Dictionary.
4. A module may not declare more than 100 symbols by ENTRY instructions.

7.5.6 IDENTIFY EXTERNAL SYMBOL

General Description

The EXTRN instruction identifies a linkage symbol that is used by this module but defined in some other module. Each external symbol, including the names of control sections, used by this module and defined in some other module must be identified.

Format

The format of the EXTRN instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	EXTRN	A relocatable symbol

Specification Rules

Name Field : Not used.

Operation Field : EXTRN

Operand Field : Any relocatable symbol defined in some other module. It may not appear as the name of a statement in the module containing the EXTRN statement.

Notes

1. External symbols, when used in an expression, may not be paired. The Translator processes them as though they originated from different control sections.
2. A symbol used in a V-type address constant need not be defined by an EXTRN statement.
3. A symbol may be declared in more than one EXTRN statement.
4. The length attribute of a symbol defined as an EXTRN is 1.

7.5.7 SET LOCATION COUNTER

General Description

The ORG instruction alters the setting of the Location Counter for the current control section.

Format

The format of the ORG instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	ORG	A relocatable expression or blank.

Specification Rules

Name Field : Not used.

Operation Field : ORG

Operand Field : Any relocatable expression composed of previously defined symbols. The unpaired relocatable symbol must be defined in the same control section in which the ORG statement appears.

Notes

1. The Location Counter is set to the value of the expression in the Operand field.
2. Omission of an Operand entry causes the Location Counter to be set one byte higher than the maximum location assigned for the control section up to this point.
3. An ORG statement must not specify a location below the beginning of the current control section.

7.5.8 BEGIN LITERAL POOL

General Description

The LTORG instruction causes all literals thus far encountered in the source module to be assembled at appropriate boundaries starting at the first double-word boundary following the LTORG statement.

Format

The format of the LTORG instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Symbol or blank.	LTORG	Not used.

Specification Rules

Name Field : Any symbol representing the first byte of the relocated Literal pool. It has a length attribute of one.

Operation Field : LTORG.

Operand Field : Not used.

Notes

1. Literals are listed and translated in the object module when the LTORG statement is encountered. Literals not covered by a LTORG statement are listed and assembled when the END card is detected.
2. Literals used after the last LTORG statement in a module are placed at the end of the first control section. If there are no LTORG statements in a module, all literals used in the module are placed at the end of the first control section. In these circumstances, the programmer must ensure that the first control section is always addressable. This means that the base address register for the first control section should not be changed through use in subsequent control sections. If the programmer does not wish to reserve a register for this purpose, he may place a LTORG statement at the end of each control section thereby ensuring that all literals appearing in that section are addressable.
3. A maximum of 32 LTORG instructions may be specified.

ISEQ

7.5.9 INPUT SEQUENCE CHECK

General Description

The ISEQ instruction checks the sequence of source input cards.

Format

The format of the ISEQ is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	ISEQ	Two decimal values of the form L,r or blank.

Specification Rules

Name Field : Not used.

Operation Field : ISEQ

Operand Field : Contains two decimal values in the form L,r.

L specifies the leftmost column of the input card to be checked.

r specifies the rightmost column of the input card to be checked.

Notes

1. Sequence checking begins with the first card following the ISEQ statement. Comparison of adjacent cards make use of the eight-bit internal collating sequence.
2. Any ISEQ with a blank operand terminates the operation.
3. Statements generated by macros are not included in the sequence check. (Source deck macro definitions will be checked.)
4. The field specified by operands L and r must not be greater than seven bytes.
5. Operand L must be greater than the end column of the source statement plus one.
6. Operand r must be equal to or greater than L.

7.6 CONTROL OF TRANSLATOR LISTING

When a module is translated, the Translator lists the program with any errors present. The following instructions are provided to give the programmer control of this listing. They allow him to suppress any parts of this listing, to give a title to the listing and to insert page changes and blank lines. These instructions do not result in the generation of any machine language code by the Translator.

TITLE

7.6.1 IDENTIFY TRANSLATOR OUTPUT

General Description

The TITLE statement is used to identify a Translator listing.

Format

The format for the TITLE statement is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
One to four characters	TITLE	A sequence of characters, enclosed in single quotation marks.
or		
Not used		

Specification Rules

Name Field : One to four characters, or not used. Used for naming columns 73-76 of the object card images of the module. Only the first TITLE card of a module should have a name entry. Name fields on subsequent TITLE cards should be blank.

Operation Field : TITLE

Operand Field : Contains the title of the module to be printed on the Translator listings. Maximum entry is 100 characters enclosed in single quotation marks.

Notes

1. A module may contain more than one TITLE statement. Each statement provides the heading for the listing of the statements that follow it until another TITLE card is read.
2. Each new TITLE card causes a page change before the header is printed.
3. The first TITLE statement will be used for all page headings until the next TITLE statement is encountered, no matter where the first statement is placed.

7.6.2 PRINT OPTIONAL DATA

General Description

The PRINT instruction controls printing of the Translator listing.

Format

The format of the PRINT instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	PRINT	One to four operands.

Specification Rules

Name Field : Not used.

Operation Field : PRINT.

Operand Field : One to four of the following terms can be used in the operand field:

- SINGLE - Text listing is single spaced.
- DOUBLE - Text listing is double spaced.
- ON - A listing is printed.
- OFF - No listing is printed.
- GEN - All statements generated by macro instructions are printed.
- NOGEN - Statements generated by macro instructions are not printed. However, the macro instruction itself and MNOTE messages will appear in the listing.
- DATA - Constants are printed in full in the listing.
- NODATA - Up to eight bytes (16 hexadecimal digits) or the first constant, whichever is shorter, of the translated data is printed on the listing.

Example Until the first PRINT statement is encountered, the options ON, NODATA, GEN, SINGLE are assumed.

AOPTN

7.6.3 TRANSLATOR OPTION

General Description

The AOPTN instruction is used to control the listing output by the Translator.

Format

The format of the AOPTN instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	AOPTN	One or more of the specified options, separated by commas.

Name Field : Not used.

Operation Field : AOPTN

Operand Field : Each of the following options may be specified in separate AOPTN instructions or appear as multiple operands (separated by commas) in a single card.

- NOLIST - The Translator listing will not appear (statements containing errors will, however, be printed).
- NOERR - The error flags will not appear in the Translator listing (A statement indicating the number of errors will, however, be printed).
- NOSYM - The symbol table will not appear in the Translator listing.
- LITERAL - This indicator is ignored.

Note If NOLIST and NOERR are both specified, then no information will be output to the listing unit. Therefore, a listing device is not required.

7.6.4 START NEW PAGE

General Description

The EJECT instruction causes the next line of the listing to appear at the top of a new page. This instruction provides a convenient way to separate routines in the program listing.

Format

The format for the EJECT instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
	EJECT	Not used; should be blank but will be treated as a comment.

Specification Rules

Name Field : Not used.

Operation Field : EJECT

Operand Field : Blank.

Notes

1. If the next line of the listing normally appears at the top of a new page, the EJECT statement has no effect.
2. The EJECT statement is not printed; except for the case where it has been flagged, when it is printed like a normal source statement without paper movement.

SPACE

7.6.5 SPACE LISTING

General Description

The SPACE instruction inserts one or more blank lines in the listing.

Format

The format for the SPACE instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used.	SPACE	A decimal value or blank

Specification Rules

Name Field : Not used.

Operation Field : SPACE.

Operand Field : A decimal number that indicates the number of lines to be skipped. A blank operand entry causes a skip of one line. If the space count exceeds the number of lines remaining on the page, a page change results.

Note

1. The SPACE statement is not printed; except for the case where it has been flagged, when it is printed as a normal source statement without paper movement.

7.6.6 ASSEMBLER LOG

An ALOG (Assembler Log) card is accepted for compatibility, but ignored.

CCW

7.7 DEFINE CHANNEL COMMAND WORD

General Description

This instruction provides a convenient way to define and generate an eight-byte channel command word aligned at a double word boundary.

Format

The format of the CCW is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
A Symbol or Not Used	CCW	Four operands, separated by commas, specifying the contents of the channel command word in the format described below.

Specification Rules

Name Field : Any symbol or not used.

Operation Field : CCW

Operand Field : All four operands must appear. They are written, from left to right, as follows:

1. An absolute expression that specifies the command code. This expression's value is right-justified in byte 1.
2. An absolute or relocatable expression that specifies the data address. The value of this expression is right-justified in bytes 2-4.
3. An absolute expression that specifies the flags for bits 32-36 and zeros for bits 37-39. The value of this expression is right-justified in byte 5 (Byte 6 is set to zero).
4. An absolute expression that specifies the count. The value of this expression is right-justified in bytes 7-8.

Notes

If there is a symbol in the name field of the CCW instruction, it is assigned the address value of the leftmost byte of the channel command word. The length attribute of the symbol is eight. The internal machine format of a channel command word is as follows:

Byte	Bits	Usage
1	0-7	Command code
2-4	8-31	Data Address
5	32-36 37-39	Flags Must be zero
6	40-47	Set to zero
7-8	48-63	Count

CNOP

7.8 CONDITIONAL NO OPERATION

General Description

The CNOP instruction allows the programmer to align an instruction at a specific word boundary. If any bytes must be skipped to align the instruction properly, the Translator ensures an unbroken instruction flow by generating no-operation instructions. This facility is useful in creating calling sequences consisting of a linkage to a subroutine followed by parameters such as channel command words (CCW).

Format

The format of the CNOP instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used.	CNOP	Two decimal terms of the form b,w.

Specification Rules

Name Field : Not used.

Operation Field : CNOP.

Operand Field : Two operands in the form of b,w where:

b specifies the byte in a word or double word in which the Location Counter is to be set. Values of 0,2,4, or 6 may be specified.

w specifies whether the byte b is a word (four bytes) or double word (eight bytes). The following pairs are valid combinations.

b,w	Specifies
0,4	Beginning of a word
2,4	Middle of a word
0,8	Beginning of a double word
2,8	Second half word of a double word
4,8	Middle (third half word) of a double word
6,8	Fourth half word of a double word

CNOP

Double Word							
Word				Word			
Halfword		Halfword		Halfword		Halfword	
Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte
 0,4 0,8		 2,4 2,8		 0,4 4,8		 2,4 6,8	

CNOP Alignment of Double Word Using (0,4,2,4)

Note

The CNOP instruction ensures the alignment of the Location Counter setting to a halfword, word, or double-word boundary. If the Location Counter is already properly aligned, the CNOP instruction has no effect. If the specified alignment requires the Location Counter to be incremented, one to three no-operation instructions are generated, each of which uses two bytes.

COM

7.9 DEFINE COMMON CONTROL SECTION

General Description

The COM instruction identifies and reserves a common area storage that may be referred to by independently translated modules that have been composed together. The common area may be broken into subfields by use of DC and DS statements. Names of subfields are defined relative to the beginning of the common section, as in the DSECT instruction.

Format

The format of the COM instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Symbol or blank	COM	not used: see below

Specification Rules

Name Field: the symbol entered establishes the name of the common area. The symbol in the name field is a valid relocatable symbol whose value is the address of the first byte of the area. The symbol has a length attribute of one.

operation field: COM

operand field : not used. Any entry appearing in this field will be treated as a comment.

Additional Information

As with CSECT and DSECT, common areas can be interrupted and resumed within a module, prefacing each resumption by another COM statement with the appropriate name in the name field. Symbols that name subfields in the common area may be used in USING instructions and as operands of machine instructions and data definitions that specify storage address.

Notes

1. No instructions or constant appearing in a common area are translated into object code. Data can be placed in the common area only through execution of the program.
2. The Translator assigns tentative start address to common areas which place all common areas after all CSECTS; several common areas are assigned start address in the order of occurrence of the area.

7.10 Macro Storage within the Translator

This section describes the instructions provided to give the programmer control of the Translator's storage areas during macro expansion. These can be used to reduce time for access to the macro library where many different macros are called, or the same macro is called more than once. These instructions are accepted on all 4-50, 4-70 machines but are ignored where the operating system is disc-oriented. Tape-oriented systems are assumed in this section.

The Translator uses a storage area and a work tape as the extension of that area to hold copies of macros taken from the macro library. It is clearly to the programmer's advantage to have frequently called macros held in the storage area, and seldom called macros on the work tape; macros which are called once only are preferably not to be copied to its area, but processed directly from the macro library.

The storage area is cumulative, and cannot be released for use by another macro. Macros presented as part of the source module are always placed in the storage area. The normal macro call line will cause a copy of that macro to be added to the area during expansion, unless the macro is in a priority group specified by the instruction MPRTY as 'not to be copied'. The storage area can be 'pre-filled' by the MCALL instruction at any stage of processing source statements.

MCALL

7.10.1 PRE-STORE LIBRARY MACROS

General Description

The MCALL instruction causes the Translator to search the macro library for a list of macros, and place these macros in its storage area. When a macro call is subsequently encountered, the macro will be processed from the storage area, thus minimising tape search time.

Format

The format of the MCALL instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
not used	MCALL	Symbols separated by commas.

Specification Rules

Name field: not used

operation field: MCALL

operand field: Macro names (that is, symbols) separated by commas; the order of names has no significance.

Notes

1. As many MCALL statements as are desired can occur in the module. If a macro is specified which has already been stored by the Translator, for example, by a previous MCALL statement, it is ignored.
2. MCALL statements may occur within macro definitions (that is, as a model line)
3. The instruction is processed by the Translator as follows: one complete scan is made of the library, and if any macro in the library is specified by MCALL it is copied into the storage area. Hence macros are stored in the order that they are held in the library. When the library is created, macro names and priorities are allocated to place the most frequently used ones at the start of the library.
4. The current MPRTY instruction has no effect on the processing of MCALL.

MPRTY

7.10.2 STORE MACROS BY PRIORITY

General Description

The instruction MPRTY is used to control selective storage of macros, based upon their priority group. (In the library, macros are held in four groups representing roughly their frequency of use). It is used to prevent the storage area being filled too rapidly, or to exclude certain macros from being stored.

Format

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
not used	MPRTY	a 4-digit combination of zeros and ones

Specification Rules

Name field: not used

operation field: MPRTY

Operand field: The four digits from left to right represent 'yes/no' indicators for each of the priority groups.

Notes

1. As many MPRTY instructions can be written as are desired. The Translator assumes a pre-set value of 1100, that is, inhibit storage of any priority 3 or 4 macro.
2. The current MPRTY state has no effect on the processing of MCALL instructions, or the storage of source module macros.

PUNCH

7.11 PUNCH A CARD

General Description

The PUNCH instruction is used so as to include Composer control data within the object module, rather than supplying it as part of the Job Stream control data.

Format

The format of the PUNCH instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used	PUNCH	A sequence of characters, enclosed in single quotation marks.

Specification Rules

Name field : not used
Operation field : PUNCH
Operand field : The first 80 characters between the quotation marks are used to generate a card (or card image), where the first character after the left-hand quote represents column 1 of the card.

Notes

1. The punched card normally appears at the same point in the object module as the PUNCH statement in the source. Where the PUNCH statement is preceded only by Translator control instructions (so that no object code has yet been generated) the punched card will precede all other Translator output.
2. To represent a single quotation mark or ampersand within the operand, two such characters must be written (cf. character constants).
3. This instruction is not available in the 5A operating regime.

MTRAC

7.12 MACRO TRACE

General Description

The MTRAC Translator control instruction allows the programmer to observe the process of macro expansion, by displaying the effective conditional transfer instructions in the macro language (AIF, AIFB, AGO, AGOB) as they are encountered. This is done by printing the source statement with a Y or an N in the 80th position to indicate that a branch did or did not take place at that point. Also, the source statements for SETA, SETB and SETC instructions will be printed.

Format

The format of the MTRAC instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used.	MTRAC	Not used.

Specification Rules

Name Field : Not used.
Operation Field : MTRAC
Operand Field : Not used.

Notes

1. If the instruction PRINT NOGEN is in force, MTRAC does not produce output.
2. MTRAC can occur either inside or outside macro definitions.
3. MTRAC is cancelled by the instruction NTRAC.

NTRAC

7.13 CANCEL MACRO TRACE

General Description

The NTRAC Translator Control instruction cancels MTRAC.

Format

The format of the NTRAC instruction is as follows:

<u>NAME</u>	<u>OPERATION</u>	<u>OPERAND</u>
Not used.	NTRAC	Not used.

Specification Rules

Name Field : Not used.
Operation Field : NTRAC
Operand Field : Not used.

8. Input/output operation

8.1 INTRODUCTION

The 4-50 can control a variety of input/output devices. All the input/output devices function independently of normal processor operation. This simultaneous operation is achieved by processor input/output channels that control input/output operations. The control electronics of each peripheral device is connected to an input/output channel via the Standard Interface. This interface permits all peripheral equipment (with the exception of remote communications and random access devices) to be attached to any channel in the 4-50. Remote communication devices must be connected to the multiplexor channel. Random access devices must be connected to a selector channel.

After an input/output operation is initiated by the program, data is transferred, byte-by-byte, between the processor and the peripheral device. This data transfer over the standard interface is controlled by the applicable input/output channel, freeing the computer to continue the program. Each of the channels on the 4-50 can interrupt normal process or operations.

8.2 INPUT/OUTPUT CHANNELS

The 4-50 has two types of input/output channels, selector channels and a multiplexor channel.

8.2.1 Selector Channels

Up to three selector channels (optional) can be attached to a 4-50. Each selector channel can address up to 256 peripheral devices.

On the 4-50, each selector channel has two standard interface trunks. Each standard interface trunk can be connected to the control electronics of an input/output device. A device control electronics controls one device (that is, card reader, printer), or a number of devices (that is tape controller: up to 16 tape stations).

Only one device can operate on a selector channel at one time. However, all selector channels can operate simultaneously with, and independently of, normal processor operation.

The multiplexor channel operates simultaneously with selector channels and independently of normal processor operation.

Control and operation of each input/output device connected to the multiplexor channel is done through a set of subchannel registers contained in non-addressable main store.

In addition to the subchannel registers, four 32-bit registers, called multiplexor registers, are provided in scratch-pad store. These registers are used for subchannel initiation and termination. Upon servicing a termination interrupt of a device connected to the multiplexor channel, the information which pertains to the completed operation is transferred from the non-addressable main store to the scratch-pad store.

The multiplexor registers in scratch-pad store are called:

Channel Address Register (CAR)
Channel Command Register-II (CCR-II)
Channel Command Register-I (CCR-I)
Assembly/Status Register

Each selector channel is controlled and operated via four 32-bit registers. These registers are located in scratch-pad store and are called:

Channel Address Register (CAR)
Channel Command Register-II (CCR-II)
Channel Command Register-I (CCR-I)
Assembly/Status Register

All the information that is required to control selector channel operation is contained in these registers. Data is transferred between the selector channel and the peripheral device one byte at a time.

8.2.2 Multiplexor Channel

The multiplexor channel is standard on the 4-50 and can address up to 256 devices.

The multiplexor channel has eight standard interface trunks each of which can be connected to a device control electronics. This permits the multiplexor channel to operate devices on all seven or eight trunks simultaneously. The limit as to the number of input/output devices that can be connected is determined by the device control electronics. A ninth trunk is provided on the multiplexor for exclusive use by the monitor typewriter.

Although the multiplexor channel can handle slow-speed devices on a time-sharing basis, it can accommodate fast devices through a burst mode. Burst mode operation is specified by the program, and causes a transfer of data to occur between a specific device and the main store without time-sharing the multiplexor channel with other input/output devices. If a program is to specify burst mode, a program check is made that other devices on the multiplexor channel have completed operation. This ensures that data is not lost.

Data is transferred between the multiplexor channel and each peripheral device one byte at a time.

Note: When a burst mode operation is executed the subchannel registers are not utilized. The input/output operation is similar to a selector channel operation and is controlled entirely by the multiplexor registers in scratch-pad store.

8.2.3 Input/Output Initiation

All input/output operations are executed by the selected channel and are independent of normal processor operation. Prior to initiation of an input/output operation, the program must supply information concerning the operation. The program must store information in the main store, such as the type of operation (read, write, etc), the data area address in the main store at which to begin the operation, and the number of bytes to be transferred by the channel. This information is called the Channel Command Word (CCW).

After the Channel Command Word is stored in the main store, the address of this CCW must be stored in a standard main store location. This standard location is called the Channel Address Word (CAW) and is main store locations 72 to 75.

Once the Channel Address Word and the Channel Command Word have been assembled, the input/output operation can be initiated.

All input/output operations are initiated by executing a Start Device instruction or by manually depressing the LOAD button on the console. Execution of the Start Device instruction causes the information contained in the Channel Address Word (CAW) and the Channel Command Word (CCW) to be transferred to the input/output channel registers in the scratch-pad store for the specified selector channel. If the specified channel is the multiplexor channel, this information is transferred to the subchannel registers in non-addressable main store for the specified device. Once this has been accomplished, the Start Device instruction terminates and the input/output operation has been initiated. Completion of the input/output operation is under control of the channel, and normal processor operation can proceed.

8.2.4 Channel Servicing

Servicing a Data Transfer When an input/output operation has been initiated and the input/output device control electronics is ready to send or receive a data byte, the channel asks the processor for a service request. When the processor permits the service request, a data transfer occurs. This servicing permits the transfer of a data byte between main store and the input/output device to occur. It also updates the information in the input/output channel registers or the subchannel registers (multiplexor) to prepare for the next data byte.

End and Chaining Servicing When an input/output operation has been completed, the channel asks the processor for another service request. This service request is required so that the channel can (1) tell the device control electronics to set a channel interrupt condition, or (2) check the current command to see if chaining is specified, and if it is to initiate the next command.

Interrupt Servicing If an input/output operation has been completed and chaining has not been specified, the input/output device control electronics causes the appropriate channel interrupt flag to be set in the Interrupt Flag register. If the Interrupt Mask register for the current processor state permits the interrupt, it is taken. At this time the channel asks the processor for another service request. This service request is required so that the channel can transfer information concerning the status of the device and the channel to the input/output channel registers in scratch-pad store. If the interrupt is caused by a device on the multiplexor channel, the appropriate subchannel registers are transferred from the non-addressable main store to the scratch-pad store.

Because all input/output servicing (servicing a data transfer, end and chaining servicing, and interrupt servicing) requires that the channel utilize main store, scratch-pad store and non-addressable main store (multiplexor devices), normal processor operation is held-off until the servicing has been completed. Servicing is time-shared with normal mode processing.

Servicing Priority Because input/output operations on all selector channels and the multiplexor channel proceed simultaneously, the processor must constantly scan the channels to determine their servicing status. If servicing is required by a channel, scanning is stopped and the input/output device is serviced. After a device is serviced, scanning is resumed.

Each selector channel and the multiplexor channel has a scanning priority. If servicing is required by devices on more than one channel, the channel with the highest priority is serviced first. The priority is as follows:

Selector Channel No. 1
Selector Channel No. 2
Selector Channel No. 3
Multiplexor Channel

The devices on the multiplexor have a priority depending upon the standard interface trunk to which they are connected; the lower the standard interface trunk in the scanning sequence, the higher the priority.

After a device has been serviced, scanning always resumes with Selector Channel No. 1. With this scanning technique, the devices with the shortest holding time (high-speed devices) must be connected to the channel with the highest scan priority. Servicing of a device connected to the multiplexor channel may be temporarily interrupted by a selector

channel service request. If this occurs, all selector channels requiring service are served before multiplexor channel servicing resumes.

The most optimum connection of device control electronics to selector channels and the multiplexor channel depends on the requirements of each installation. However, a general rule is to connect the device control electronics which control devices with the highest data transfer requirements to the channels with the highest priority. The remaining device control electronics are connected in descending order of data transfer requirements to descending priority sequence of channels.

8.2.5 Channel Address Word (CAW)

The Channel Address Word (CAW) is used by the Start Device instruction (see Privileged Instructions Chapter 6.6), and specifies the address of the first Channel Command Word (CCW) used to control the operation of the input/output device. If the Store Protect option is installed, the store protection key must also be stored in the CAW before a Start Device instruction is issued.

The CAW must be stored in main store locations 72 to 75 before executing a Start Device instruction and has the following format:

Key	0000	Address of CCW
0	3 4	7 8 31

Bit Positions 0 to 3 contain the store protection key. It is used to ensure that data is not being transferred to a protected store area. If the Store Protect option is not installed, these bits must be zero.

Bit Positions 4 to 7 are reserved for future expansion.

Bit Positions 8 to 31 contain the main store address of the initial channel command word.

8.2.6 Channel Command Word (CCW)

The Channel Command Word (CCW) supplies the information for controlling the operation of the input/output device. This information must be stored in main store by the program before a Start Device instruction is issued. The CCW consists of two 32-bit words in main store that must be aligned on a double word boundary. The CCW has the following format:

Command Code	Address of First Data Byte or Address of Next CCW if Command is a Transfer in Channel
0	7 8 31

Flags	Reserved for Future Expansion	Byte Count
32	36 37	47 48 63

Bit Positions 0 to 7 contain the command code, which specifies the operation to be performed by the input/output device. (See table below).

Command Code Operations

Command Code								Operation
0	1	2	3	4	5	6	7	Bit Position
M	M	M	M	0	0	0	1	Sense
M	M	M	M	M/O	0	1	0	Read Reverse
M	M	M	M/B	M/O	0	1	1	Write
M	M	M	M/B	M/O	1	0	0	Write Erase
M	M	M	M/B	M/O	1	0	1	Read
M	M	M	M	0	1	1	1	Write Control
M	M	M	M	1	0	0	1	Transfer in Channel

Notes:

- Any command code other than the ones shown in the table is illegal and must not be programmed. If this rule is violated, the resulting effect on the channel and device is unpredictable. If one of the legal commands is issued to a device which is not capable of accepting the operation (that is, a Write command is issued to a card reader), the command, after being accepted, is terminated by the device control electronics. A channel interrupt occurs and the sense byte(s) indicate the illegal operation.
- The bit position designated as "B" indicates that the specified device is connected to the multiplexor channel and the multiplexor is to be operated in the burst mode. If this position is a 1 bit, the multiplexor channel is locked-on to the selected device, and the servicing of other devices connected to the multiplexor channel is inhibited. A burst mode can only be initiated when it is specified in the first command of a chain. Subsequent commands, linked by chaining, cannot initiate a burst mode. However, if the first command in a chain specifies burst mode, all commands in the chain are executed under burst mode conditions.
- Bit positions designated as M (modifier) indicate variations of the operation and are unique to the specific input/output device.

Definition of these M bits is provided in the applicable input/output device reference manuals.

An explanation of the commands shown in the above table is as follows:

Sense (0001) - Information is transferred from the specified input/output device control electronics to the main store. The information transferred indicates unusual conditions that occurred as a result of the last operation performed by the device. (The information received is defined in the individual input/output device reference manuals). The address specified by the CCW is the leftmost main store location of the input area.

Note: Parity is not checked on data transferred to the main store by this command.

Read Reverse (0010) - Information is transferred from the specified input/output device to the main store in descending order. The address specified by the CCW is the rightmost main store location of the input area.

Write (0011) - Information is transferred from the main store to the specified input/output device. The address specified by the CCW is the leftmost main store location of the output area.

Write Erase (0100) - Information is transferred from the main store to the specified input/output device control electronics. Data is not written to tape and the tape is erased in accordance with the byte count (applicable to magnetic tapes only). The address specified by the CCW is the leftmost main store location of the output area.

Read (0101) - Information is transferred from the specified input/output device to the main store in ascending order. The address specified by the CCW is the leftmost main store location of the input area.

Write Control (0111) - Information is transferred from the main store to the specified input/output device control electronics. The device control electronics interprets this information as control information and initiates a function not involving a data transfer. The address specified by the CCW is the leftmost main store location of the output area.

Transfer in Channel (1001) - This command provides chaining of CCW's that are not located in adjacent double word main store. An actual branch to the address of the next CCW is taken. The branch address (specified in bits 8 to 31 of the Channel Command Word) must specify a double word location. (Bits 29 to 31 must be zero.) This command cannot be the first command in a chain. A Transfer in Channel command may address another Transfer in Channel command.

Note: The flag bits are ignored if a Transfer in Channel command is specified. The flag bits of the preceding command remain effective.

Bit Positions 8 to 31 (see CCW format) contain the address of the first byte in the main store at which the input/output operation begins, or if the command is a Transfer in Channel, the main store address of the next CCW to be executed. The address of the first byte of the next data segment can also be specified if data chaining.

Bit Positions 32 to 36 are the flag bits and have the following significance:

1. Bit position 32 is the Chain Data flag (CD). In addition to transferring data to and from a single main store area, 4-50 can read into, or write from, many non-contiguous areas of main store by executing one Start Device instruction. When data chaining is specified by setting this bit, a chain (series of Channel Command Words in sequence) is used and each Channel Command Word designates an area of main store at which to continue the current operation. When one Channel Command Word has a lapsed byte count, the next Channel Command Word in sequence is automatically fetched. The current operation is continued at the main store area specified by the new Channel Command Word. The command code of the new CCW is ignored unless it specifies a Transfer in Channel. If any of the following channel status byte conditions occur, data chaining is suppressed (see Channel Status Byte for further definition):

Program Check

Protection Check

Channel Control Check

Channel Data Check (if the operation is a write)

Incorrect Length Condition

When data chaining, the chain data flag in the last Channel Command Word must be reset. This causes the data chain to be terminated upon completion of the operation specified by this CCW.

2. Bit position 33 is the Chain Command flag (CC). The 4-50 can perform several operations to an input/output device by executing one Start Device instruction. When command chaining is specified by setting this bit, a chain (series of Channel Command Words in sequence) is used and each Channel Command Word specified by one Channel Command Word is completed, the next Channel Command Word in sequence is automatically fetched and the operation specified is initiated. If any of the following conditions occur, command chaining is suppressed:

- (a) Channel status byte conditions (see channel status byte for further definition).

Incorrect Length Condition and suppress length flag is zero.
Program Check
Protection Check
Channel Control Check

- (b) Standard device byte conditions (see standard device byte for further definition).

Secondary Indicator is set
Device Inoperable is set
Device End is not set

When command chaining, the chain command flag in the last Channel Command Word must be reset. This causes the command chain to be terminated upon completion of the operation specified by this CCW.

3. Bit position 34 is the Suppress Length Indication flag (SLI). Incorrect length occurs in the 4-50 when the number of bytes specified in the Channel Command Word is not equal to the number of bytes sought by, or sent from, the input/output device. (When a command or chain of commands terminates, the data byte count has not lapsed.) An example of an incorrect length condition is a tape read which terminates on a gap before the byte count has lapsed. If the SLI bit is set, the program does not receive an indication of an incorrect length upon termination of the input/output operation. If the SLI bit is reset, the program receives an indication of an incorrect length upon termination of the input/output operation. This indication is contained in the channel status byte.

Notes

1. If the SLI bit is set and the chain data flag of the final CCW in a chain is reset, the incorrect length indication is suppressed, if it occurs.
2. If the chain data flag of a CCW is set and an incorrect length condition occurs, the program is notified of the condition regardless of the setting of the SLI flag.
3. Bit position 35 is the Skip flag (SKIP). In conjunction with data chaining, portions of a block of information can be suppressed during an input operation. If this bit is set, the transfer of data to main store specified by this command is suppressed. This bit can be used only with Read, Read Reverse or Sense commands.
4. Bit position 36 is the Program Controlled Interrupt flag (PCI). During data and command chaining, the 4-50 has the ability to notify the program of the progress of chaining through an interrupt when a Channel Command Word is fetched. When this bit

is set, a channel interrupt occurs when the Channel Command Word is fetched from main store and the first data byte has been transferred. This flag is ineffective if the channel is the multiplexor operating in burst mode.

5. Bit positions 37 to 47 are reserved for future expansion and must be set to all zeros by the program.
6. Bit positions 48 to 63 contain the count of the number of bytes to be transferred to or from the main store during the input/output operation (from 0 to 65,536 bytes). An initial count of zero specifies the maximum number of bytes to be transferred.

8.3 INPUT/OUTPUT CHANNEL REGISTERS

The Channel Address Word (CAW) and the Channel Command Word(s) (CCW) are stored by the program in main store. However, when an input/output operation is initiated, the information contained in the CAW and the first CCW is transferred to the scratch-pad input/output channel registers for the channel specified by the Start Device instruction. Because the access speed in the scratch-pad store is faster than the main store, faster servicing of input/output devices is possible. These registers also eliminate the need for the program to reset Channel Command Words, because incrementing and decrementing addresses and byte count is done in the scratch-pad store. These registers allow the input/output operation to proceed under control of the specified channel, thereby permitting normal mode processing to continue.

Input/Output Channel Registers

Register	Selector Channel	Multiplexor Channel	
	Scratch-Pad Store	Scratch-Pad Store	Non-Addressable Main Store
Channel Address Register (CAR)	1 per selector channel	1 per multiplexor channel	1 per device
Channel Command Register-I (CCR-I)	1 per selector channel	1 per multiplexor channel	1 per device
Channel Command Register-II (CCR-II)	1 per selector command	1 per multiplexor channel	1 per device
Assembly/Status Register	1 per selector channel	1 per multiplexor channel	None

The format for each of these four 32-bit registers is as follows:

Device No.	Address of next CCW
0	7 8 31

Bit Positions 8 to 31 contain the address of the next Channel Command Word if chaining is specified. This information is obtained by incrementing the address of the first CCW by eight. The address of the first CCW is obtained from the Channel Address Word (CAW).

0000	Command Code	Data Address of First Byte or Location of new CCW if Command is Transfer in Channel
0	3 4	7 8 31

Bit Positions 4 to 7 contain the command code. This code is obtained from the Channel Command Word. The commands are defined as follows:

Bit Positions 8 to 31 contain the address of the initial byte in the main store at which the operation begins; or contains the branch address if the command is a Transfer in Channel. This information is obtained from the Channel Command Word.

Channel Command Register-II (CCR-II)

Flags					000			Channel Status Byte							Byte Count					
0			4	5			7	8							15	16				31

Bit Positions 0 to 4 contain the flags. The flags are obtained from the Channel Command Word. The flag bits are defined as follows:

- Bit 0 - Chain data flag (CD)
- Bit 1 - Chain command flag (CC)
- Bit 2 - Suppress length indicator flag (SLI)
- Bit 3 - Skip flag (SKIP)
- Bit 4 - Program controlled interrupt flag (PCI)

Bit Positions 5 to 7 are reserved for future use.

Bit Positions 8 to 15 contain the channel status byte. The bits of the channel status byte are generated as a result of the input/output operation and are defined as follows:

- Bit 8 - Program Controlled Interrupt
- Bit 9 - Incorrect Length
- Bit 10 - Program Check
- Bit 11 - Protection Check
- Bit 12 - Channel Data Check
- Bit 13 - Channel Control Check
- Bit 14 - Reserved for use by the processor
- Bit 15 - Termination Interrupt

(For a detailed description of the above see Channel Status Byte below).

Bit Positions 16 to 31 contain the number of bytes of main store to or from which data is transferred. This information is obtained from the Channel Command Word. The count can range from 0 bytes to 65,536 bytes.

Assembly/Status Register

Data Bytes															Standard Device Byte					
0															23	24				31

When the device status is stored as a result of an input/output operation, bit positions 24 to 31 of the assembly/status register are used to store the standard device byte. The bits of the standard device byte supply status information pertaining to the device control electronics and the input/output device and are defined as follows:

Bit 24 - External Device Request Interrupt Pending
Bit 25 - Terminating Interrupt Pending
Bit 26 - Device Busy
Bit 27 - Control Busy (not applicable)
Bit 28 - Device End
Bit 29 - Secondary Indicator
Bit 30 - Device Inoperable
Bit 31 - Status Modifier

(For a detailed description of the above, see Standard Device Byte below.)

8.4 INPUT/OUTPUT INSTRUCTIONS

There are four processor instructions which are concerned with input/output operations. They are Start Device, Halt Device, Check Channel and Test Device. These instructions are executed by the processor and the results, in the form of condition codes, are available upon instruction completion. It should be noted that the condition code settings indicate the results of the instruction and not the results of the input/output operation that the instruction may be initiating. The channel continues off-line to accomplish the input/output operation as specified by the instruction. However, during this time the processor continues executing subsequent instructions.

8.4.1 Start Device Instruction

The Start Device instruction is a privileged operation and, therefore, can be executed only if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to zero. This instruction is executed in the normal mode. Continuation of program execution is delayed until the Start Device instruction has been terminated.

Upon execution of a Start Device instruction, the following events occur. (See figure 1).

Block 1. If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to zero, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs (if permitted).

Block 2. If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 3. If the specified channel is a selector channel that is busy or has an interrupt pending (termination or external device request) or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

FIGURE 1 FUNCTIONAL LOGIC OF START DEVICE INSTRUCTION

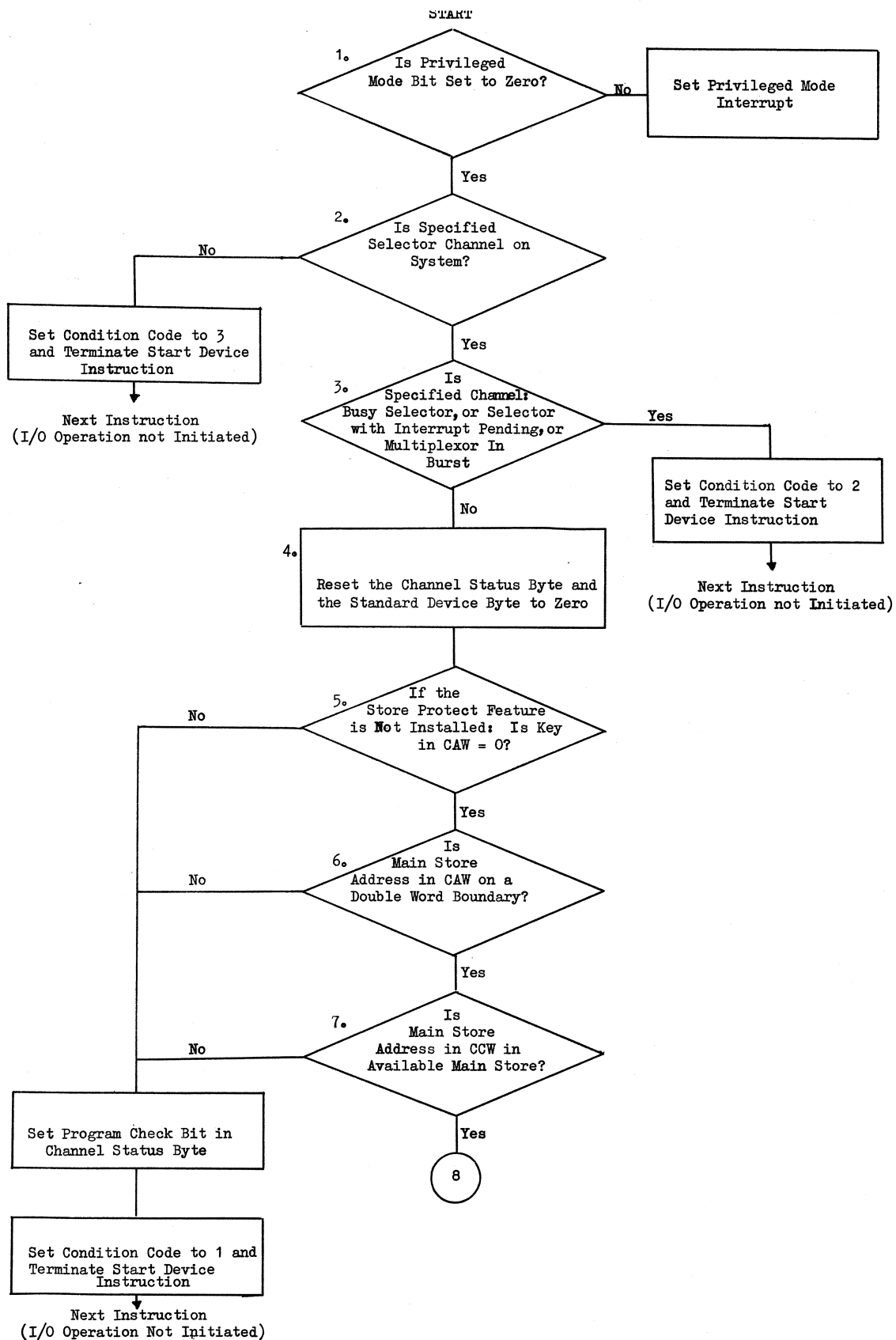
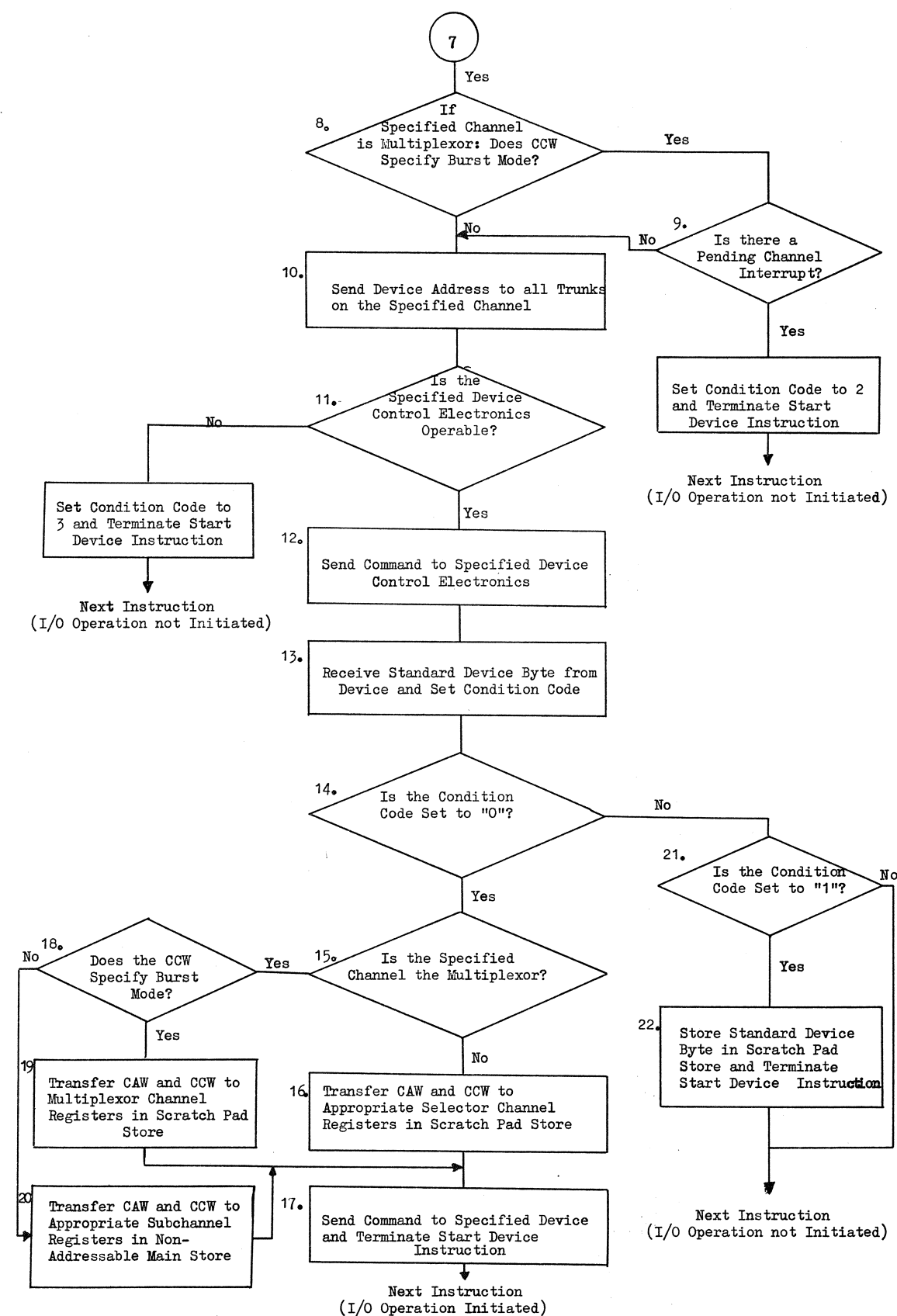


FIGURE 1(cont.)



Block 4. The channel status byte and the standard device byte for the specified channel are reset to zeros in the appropriate channel registers.

Block 5. If the Store Protect feature is not installed, the key in the Channel Address Word (CAW) is tested to see if it is equal to zeros. If it is not zeros, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated, and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 6. The main store address in the Channel Address Word is tested to see if it is on a double word boundary. If it is not, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 7. The main store address in the Channel Command Word (CCW) is tested to see if it is within the available main store for the system. If it is not, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 8. If the specified channel is the multiplexor channel, the command code in the Channel Command Word is tested to see if a burst mode operation has been specified.

Block 9. If a burst mode operation has been specified, a test is made to see if there is a terminating interrupt pending on any of the trunks on the multiplexor. If a terminating interrupt is pending, the condition code is set to 2, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 10. The device address as specified in the Start Device instruction is sent to all trunks on the addressed channel.

Block 11. A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the condition code is set to 3, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 12. If the specified device control electronics is operable, the command code from the Channel Command Word is sent to the specified device control electronics.

Block 13. After receiving the command code, the device control electronics sends the standard device byte to the processor. This standard device byte is not stored in the channel registers in the scratch-pad store. It is used to set the proper condition code as follows:

Condition Code	Definition
3	Device control electronics is inoperable
2	A termination interrupt pending condition exists in the device control electronics on the multiplexor channel.
2	The device control electronics is busy working with the specified device.
2	The device control electronics is busy working with a device other than the one specified.
1	An external device request interrupt pending condition exists in the device control electronics on the multiplexor channel.
1	The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek).
*1	The specified device is inoperable.
0	The specified device and control electronics is available.

*If the command is a Sense, the condition code is set to 0 permitting the operation to be initiated.

Block 14. A test is made to see if the condition code is set to 0 (input/output operation can be initiated).

Block 15. If the condition code is zero, a test is made to see if the specified channel is the multiplexor channel.

Block 16. If the specified channel is a selector channel, the Channel Address Word is fetched from main store locations 72 to 75 and stored in the appropriate channel address register. Using the main store address specified in the CAW, the Channel Command Word is fetched from the main store and stored in the appropriate channel command registers.

Block 17. The command is sent to the specified device control electronics and the Start Device is terminated (with the condition code set to 0). The input/output operation is initiated and proceeds under control of the appropriate channel and registers in the scratch-pad store and non-addressable main store (multiplexor devices). Normal program execution of the next instruction continues simultaneously with the input/output operation.

Block 18. If the specified channel is the multiplexor channel, the command code in the Channel Command Word is tested to see if a burst mode operation has been specified.

Block 19. If a burst mode operation has been specified, the Channel Address Word is fetched from main store locations 72 to 75 and stored in the channel address register for the multiplexor channel. Using the main store address specified in the CAW, the Channel Command Word is fetched and stored in the channel command registers for the multiplexor channel.

Block 20. If a burst mode operation has not been specified, the Channel Address Word and the Channel Command Word are fetched from the main store and stored in the subchannel registers in the non-addressable main store for the device specified.

Block 21. If the condition code is not set to 0, a test is made to see if the condition code is set to 1.

Block 22. If the condition code is set to 1, the standard device byte is transferred to the channel registers for the channel specified, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Notes on Start Device Instruction

1. The channel status byte and the standard device byte are not stored if the condition codes are 0, 2, 3.
2. If the specified channel and device can be initiated (CC = 0) the contents of the Channel Address Word and Channel Command Word are loaded into the appropriate channel registers and the command is sent to the device. The legality of the command is not determined at initiation time. If the device gets an illegal command, the operation is terminated and a channel interrupt occurs. The standard device byte (stored in the appropriate channel registers when the interrupt is taken) indicates a secondary indicator. A Sense command must be issued to bring the Sense byte(s) into main store. The Sense byte(s) indicate the illegal operation.

3. If execution of this instruction causes the channel status byte or the standard device byte to be stored, the program must inhibit interrupts on this channel until the status byte has been analysed or moved from the channel registers. If interrupts are permitted and one occurs the standard device byte and the channel status byte are destroyed.

8.4.2 Halt Device Instruction

The Halt Device instruction is a privileged operation and can be executed only if the mode bit (bit position 15 of the Interrupt Status register) for the current state is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until termination is accepted by the device control electronics. When the device control electronics receives the termination, it causes a channel interrupt to occur. Both the channel number and the device number must be specified in the instruction. Because the Channel Address Word is not referred to by the Halt Device instruction, both the Channel Address Word and a Channel Command Word are not required.

Upon execution of a Halt Device instruction, the following events occur (see Figure 2).

Block 1. If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to zero, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs (if permitted).

Block 2. If the specified channel is a selector channel which is not available on the system, the condition code is set to 3, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 3. If the specified channel is a selector channel that is busy or if the specified channel is the multiplexor that is operating in the burst mode, the Chain Command (CC) flag in CCR-II is reset, the device control electronics is told to set an end condition, the condition code is set to 2, the Halt Device instruction is terminated, and program control is transferred to the next instruction.

Notes:

1. Setting an end condition causes the device to be halted on servicing the next data transfer (see Servicing a Data transfer).
2. The Chain Command flag must be reset to suppress chaining during termination (see Chaining and End Servicing below).

Block 4. If the specified channel is not the multiplexor channel, the condition code is set to 0, the Halt Device instruction is terminated and program control is transferred to the next instruction.

FIGURE 2. FUNCTIONAL LOGIC OF HALT DEVICE INSTRUCTION

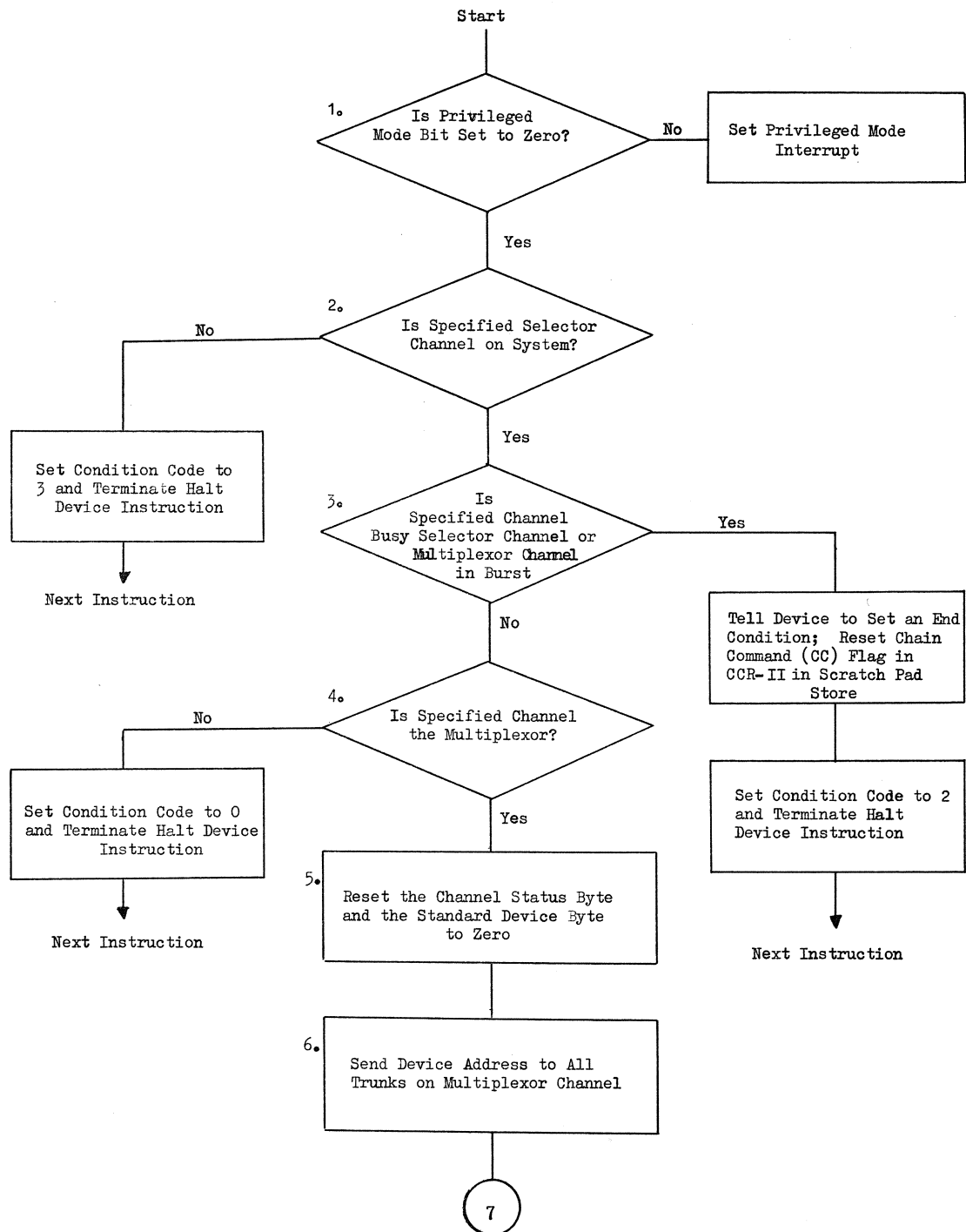
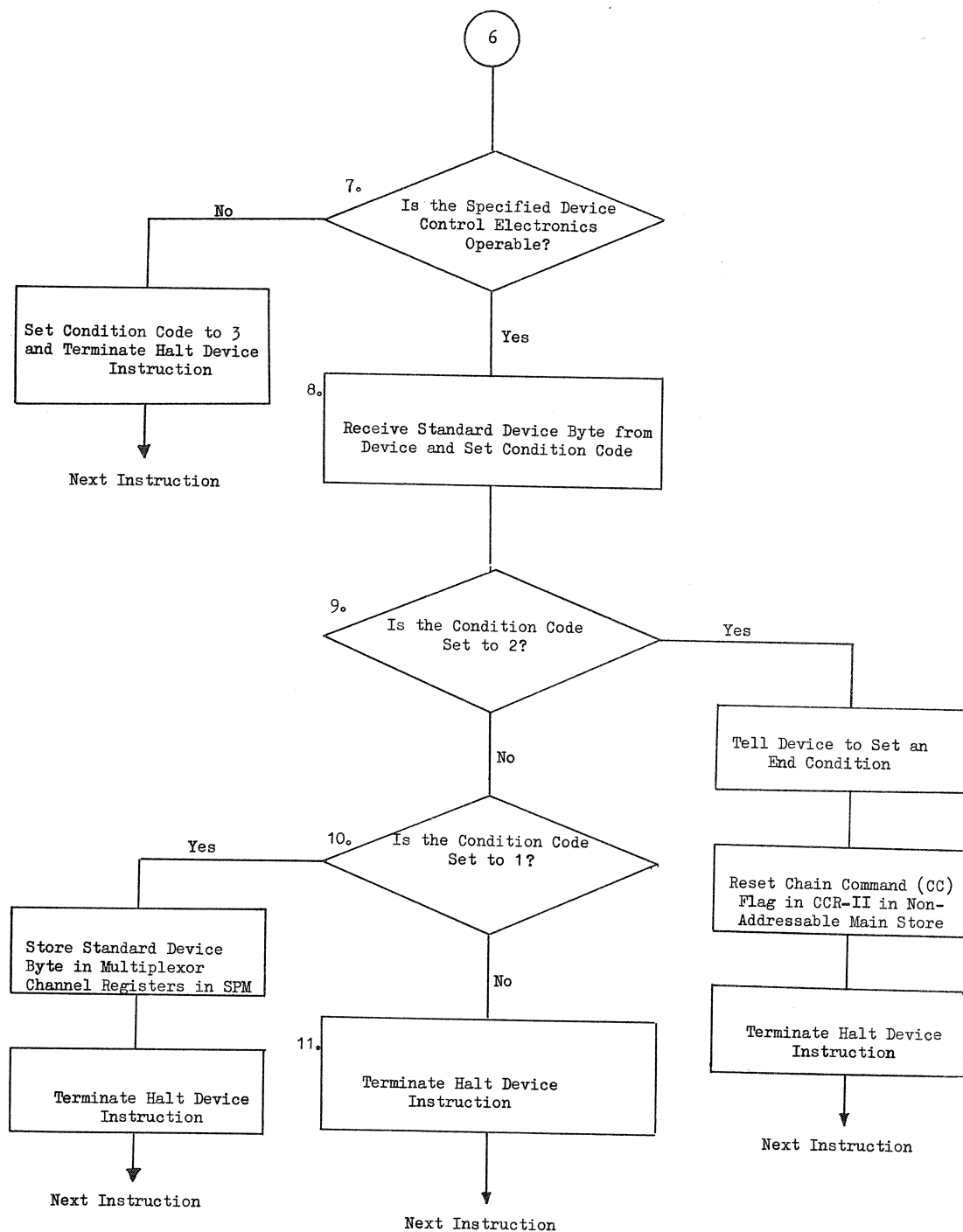


FIGURE 2(cont.)



Block 5. If the specified channel is the multiplexor channel, the channel status byte and the standard device byte are reset to zeros in the multiplexor channel registers.

Block 6. The device address as specified in the Start Device instruction is sent to all trunks on the multiplexor channel.

Block 7. A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not the condition code is set to 3, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 8. If the specified device control electronics is operable, it sends the standard device byte to the processor. This standard device byte is not stored in the channel registers. It is used to set the proper condition code as follows:

Condition Code	Definition
3	Device control electronics is inoperable.
0	A termination interrupt pending condition exists in the device control electronics.
2	The device control electronics is busy working with the specified device.
0	The device control electronics is busy working with a device other than the one specified.
1	An external device request interrupt pending condition exists in the device control electronics.
1	The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek).
1	The specified device is inoperable.
0	The specified device and control electronics is available.

Block 9. A test is made to see if the condition code is set to 2 (input/output operation can be terminated). If it is, the device control electronics is told to set an end condition, the Chain Command (CC) flag in CCR-II in the appropriate sub-channel register is reset and control is transferred to the next instruction.

Notes:

1. Setting an end condition causes the device to be halted on servicing next data transfer (see Servicing a Data Transfer).
2. The Chain Command flag must be reset to suppress chaining during termination (see Chaining and End Servicing section, below).

Block 10. If the condition code is set to 1, the standard device byte is transferred to the assembly/status registers for the multiplexor channel, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 11. If the condition code is not set to 1 (it is 0, 3) the Halt Device instruction is terminated and program control is transferred to the next instruction.

Notes on Halt Device Instruction

1. The channel status byte is not stored as a result of this operation. However, the incorrect length bit in the channel status byte may be set.
2. The standard device byte is not stored if the condition codes are 0, 2, 3.
3. If an interrupt pending (termination or external device request) condition exists on a specified selector channel, the condition code is set to zero.
4. The channel and device are terminated at the next data service request (see Servicing a Data Transfer).
5. The Channel Address Word (CAW) and Channel Command Word (CCW) are not used by this instruction.
6. If execution of this instruction causes the standard device byte to be stored in the multiplexor channel registers, the program must inhibit interrupts from the multiplexor channel until the standard device byte has been analysed or moved from the channel registers. If interrupts are permitted and one occurs, the standard device byte is destroyed.

8.4.3 Test Device Instruction

The status of an input/output device can be tested by executing a Test Device instruction. The Test Device instruction is a privileged operation and can be executed only if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until the instruction is terminated.

Both the channel number and the device number must be specified in the instruction. Because the Channel Address Word is not referred to by the Test Device instruction, the Channel Address Word and a Channel Command Word are not required.

Upon execution of a Test Device instruction, the following events occur (see figure 3).

Block 1. If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to 0, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs, if permitted.

Block 2. If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 3. If the specified channel is a selector channel that is busy or has an interrupt pending (termination or external device request); or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 4. The channel status byte and the standard device byte for the specified channel are reset to zeros in the appropriate channel registers.

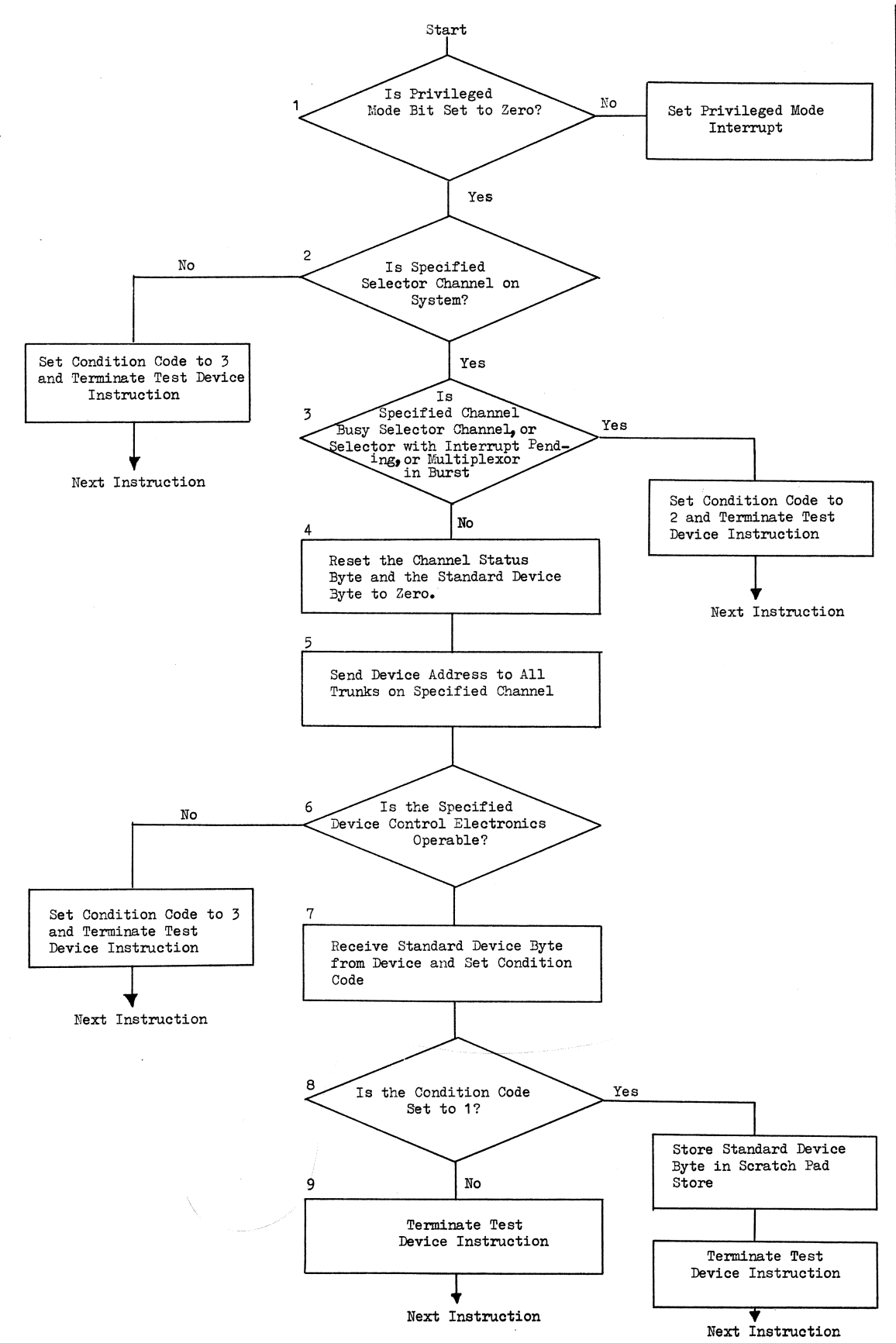
Block 5. The device address as specified in the Test Device instruction is sent to all trunks on the addressed channel.

Block 6. A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the condition code is set to 3, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 7. If the specified device control electronics is operable, it sends the standard device byte to the processor. This standard device byte is not stored in the channel registers. It is used to set the proper condition code as follows:

Condition Code	Meaning
3	Device control electronics is inoperable.
2	A termination interrupt pending condition exists in the device control electronics on the multiplexor channel.
2	The device control electronics is busy working with the specified device.

FIGURE 3 FUNCTIONAL LOGIC OF TEST DEVICE INSTRUCTION



Condition Code	Meaning
2	The device control electronics is busy working with a device other than the one specified.
1	An external device request interrupt pending condition exists in the device control electronics on the multiplexor channel.
1	The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek).
1	The specified device is inoperable.
0	The specified device and control electronics is available.

Block 8 A test is made to see if the condition code is set to 1. If it is, the standard device byte is transferred to the channel registers for the channel specified, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 9 If the condition code is not set to 1, the Test Device instruction is terminated and control is transferred to the next instruction.

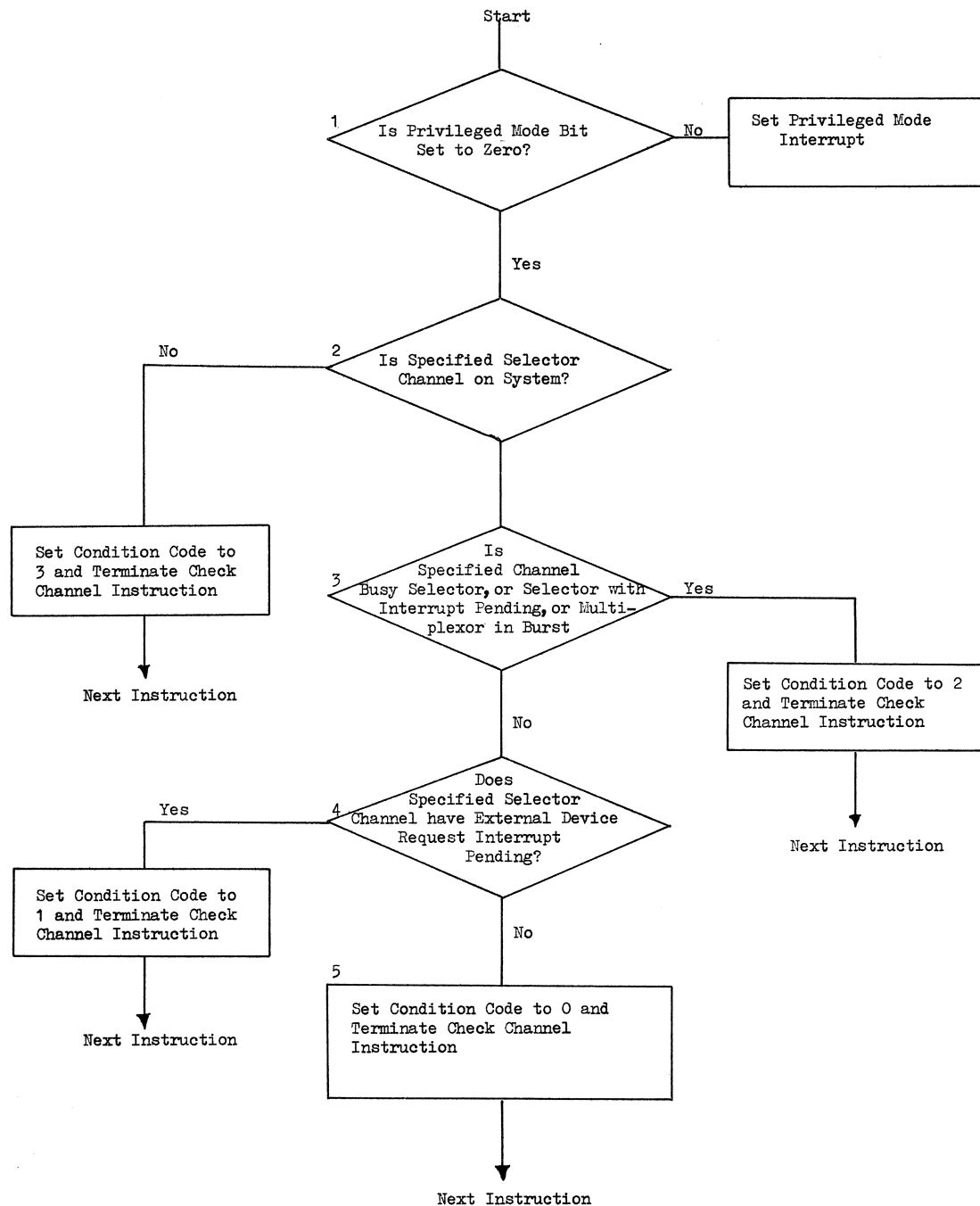
Notes on Test Device Instruction:

1. The channel status byte is not stored as a result of this operation.
2. The standard device byte is not stored if the condition codes are 0, 2, or 3.
3. The Channel Address Word (CAW) and Channel Command Word (CCW) are not used by this instruction.
4. If execution of this instruction causes the standard device byte to be stored in the multiplexor channel registers, the program must inhibit interrupts from the multiplexor channel until the standard device byte has been analysed or moved from the channel registers. If interrupts are permitted and one occurs, the standard device byte is destroyed.

8.4.4 Check Channel Instruction

The status of an input/output channel can be tested by executing a Check Channel instruction. The Check Channel instruction is a privileged operation and can only be executed if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until the instruction is terminated.

FIGURE 4 FUNCTIONAL LOGIC OF CHECK CHANNEL INSTRUCTION



Only the channel number must be specified in the instruction. Because the Channel Address Word is not referred to by the Check Channel instruction, the Channel Address Word and a Channel Command Word are not required.

Upon execution of a Check Channel instruction, the following events occur (see figure 4).

Block 1 If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to 0, the privileged operation bit is set in the Interrupt Flag register and interrupt occurs if permitted.

Block 2 If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 3 If the specified channel is a selector channel that is busy or has a termination interrupt pending; or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 4 If the specified channel is a selector channel that has an external device request interrupt pending, the condition code is set to 1, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 5 If the specified channel is a selector channel that is not busy and has no interrupts pending; or is the multiplexor channel that is not operating in the burst mode, the condition code is set to 0, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Notes on Check Channel instruction:

1. The channel status byte and the standard device byte are never stored by this instruction.
2. The Channel Address Word (CAW) and the Channel Command Word (CCW) are not used by this instruction.

8.5 INPUT/OUTPUT STATUS INDICATORS

Three levels of status information are available to the program to control input/output operation. The first pertains to the setting of the condition code when an input/output instruction is issued. The second level provides more detailed information by storing the channel status byte and the standard device byte in the appropriate input/output channel registers in the scratch-pad store. The third level of status information is generated by, and stored in, the device control electronics until a Sense command is issued. At that time the status information (Sense bytes) are transferred to main store similar to a data transfer.

8.5.1 Condition Code

The condition code is set by the input/output instructions and can be tested by the Branch On Condition instruction. It should be noted that the condition code settings indicate the result of the input/output instructions only. They do not indicate the results of the input/output operation. Condition Code settings for all input/output instructions are as follows:

Start Device Instruction Condition Code Settings

Condition Code	I/O Operation Initiated	Meaning
0	Yes	1. The device control electronics and the device specified are available. 2. The Start Device instruction specifies a Sense command to a device that is inoperable.
1	No	This condition code indicates that either the channel status byte or the standard device byte has been stored in the channel registers for the specified channel. The channel status byte is stored under the following conditions: 1. A parity error occurs while accessing the Channel Address Word or a Channel Command Word. The channel control check bit in the channel status byte is set. 2. The Store Protect feature is not installed and the key in the CAW is not zero. The program check bit in the channel status byte is set. 3. The main Store address specified in the CAW is not on a double word boundary. The program check bit in the channel status byte is set. 4. The main store address in the CCW specifies an address outside the available store for the system. The program check bit in the channel status byte is set. The standard device byte is stored under the following conditions: 1. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external

Condition Code	I/O Operation Initiated	Meaning
		device request interrupt pending bit in the standard device byte is set. 2. The Start Device instruction specifies a command which is other than a Sense command and the addressed device is inoperable. The device inoperable bit in the standard device byte is set. 3. The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek to a random access device). The device busy bit and the device end bit in the standard device byte is set.
2	No	1. A selector channel is specified that is busy. 2. A selector channel is specified that has an interrupt pending (termination or external device request). 3. The multiplexor channel is specified and it is operating in burst mode. 4. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device. 5. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. 6. A burst mode operation is directed to the multiplexor and there is a termination interrupt pending on one of the attached device control electronics.
3	No	1. A selector channel is specified that is not in the system. 2. The specified device control electronics is inoperable.

Halt Device Instruction Condition Code Settings

Condition Code	I/O Operation Terminated	Meaning
0	No	1. The device control electronics or the device specified on the multiplexor channel is not busy. No termination is required. 2. A selector channel or the multiplexor channel operating in burst mode is specified and it is not busy. No termination is required. 3. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. No termination is required.
1	No	This condition code indicates that the specified device is on the multiplexor channel and that the standard device byte has been stored in the channel registers for the multiplexor channel. The channel status byte is never stored. The standard device byte is stored under the following conditions: 1. The specified device indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set. 2. The specified device is busy but the device control electronics is not busy (that is, tape rewinding). The device busy bit in the standard device byte is set. 3. The specified device is inoperable. The device inoperable bit in the standard device byte is set.
2	Yes	1. A selector channel is specified that is busy. 2. The multiplexor channel is specified and it is operating in the burst mode. 3. The multiplexor channel is specified and the addressed device control electronics and device are busy.

Condition Code	I/O Operation Terminated	Meaning
3	No	1. A selector channel is specified that it is not in the system. 2. The specified device control electronics is inoperable.

Test Device Instruction Condition Code Settings

Condition Code	Meaning
0	The device control electronics and the device are available. Note: There may be pending interrupts on the multiplexor channel that would prohibit a burst mode operation being initiated.
1	This condition code indicates that the standard device byte has been stored in the channel registers for the specified channel. The channel status byte is never stored by this instruction. The standard device byte is stored under the following conditions: 1. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set. 2. The specified device is busy but the device control electronics is not busy (that is, tape rewinding, off-line seek to a random access device). The device busy bit in the standard device byte is set. 3. The specified device is inoperable. The device inoperable bit in the standard device byte is set.
2	1. A selector channel is specified that is busy. 2. A selector channel is specified that has an interrupt pending (termination or external device request). 3. The multiplexor channel is specified and it is operating in burst mode. 4. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device.

Condition Code	Meaning
	5. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.
3	1. A selector channel is specified which is not on the system. 2. The specified device control electronics is inoperable. 3. A device is specified that is not in the system.

Check Channel Instruction Condition Code Setting

Condition Code	Meaning
0	1. The specified selector channel is not busy and has no interrupts pending. 2. The specified multiplexor channel is not operating in the burst mode.
1	The specified selector channel has an external device request interrupt pending.
2	1. The specified selector channel is busy or has a terminating interrupt pending. 2. The specified multiplexor is operating in the burst mode.
3	A selector channel is specified that is not in the system.

8.5.2 Channel Status Byte

The channel status byte is stored in Channel Command Register-II (bit positions 8 to 15) for the appropriate channel. It contains information concerning the status of the channel when a channel interrupt occurs, or at the completion of a Start, Halt or Test Device instruction if the condition code indicates that Status is stored. The bit significance of the channel status byte is as follows:

Bit Position 8 is the program controlled interrupt bit. When set, this bit indicates that a Channel Command Word was accessed which had the program controlled interrupt flag bit set. A channel interrupt occurs for the appropriate channel while the input/output operation specified by the Channel Command Word is being executed.

Notes

1. The program controlled channel interrupt occurs after the first data byte has been transferred.
2. If a Channel Command Word that specifies a burst mode operation is fetched and the program controlled interrupt flag bit is set, the program controlled interrupt does not occur until completion of the burst operation.

Bit Position 9 is the incorrect length bit. When set, this bit indicates that when the input/output operation was terminated, the byte count specified in the channel command was not equal to the number of bytes received from, or sent to, the input/output device. The incorrect length indicator can be set only if the suppress length indicator flag bit in the Channel Command Word is reset to 0.

The following conditions cause the incorrect length bit to be set:

1. Count High on Input (Read, Read Reverse, Sense). The main store area specified by the Channel Command Word is not completely filled by transmission from the device. The final byte count in Channel Command Register-II is greater than zero.
2. Count High on Output (Write, Write Control). Data in the main store area specified by the Channel Command Word is not completely transferred and the device terminated. The final byte count in Channel Command Register-II is greater than zero.

Notes

1. If incorrect length occurs during command chaining and the Suppress Length Indicator flag bit of the current command is reset, the incorrect length bit is set.
2. If incorrect length occurs during the last command of a chain (the Chain Data Flag bit is reset), and the Suppress Length Indicator flag of the command is set, the incorrect length bit is not set.

Bit Position 10 is the program check bit. When set, this bit indicates that a programming error was detected by the channel.

The following conditions cause the program check bit to be set:

1. Invalid Channel Command Word Address. The addressed Channel Command Word is not located on a double word boundary.
2. Invalid Channel Command Word Address. The addressed Channel Command Word is outside the available main store for the particular installation.
3. Invalid Data Address. The main store location specified by the data address in the Channel Command Word is outside the available main store for the particular installation.

4. Invalid Key. The store protection key in the Channel Address Word is not zero and the system does not have the Store Protect option installed.

Notes:

1. If a program check error occurs during input/output initiation, the operation is suppressed and the program is notified of the error by the condition error setting.
2. If a program check error occurs while the input/output operation is in progress, the operation is terminated and a channel interrupt occurs for the specified channel.
3. If a program check error occurs during chaining (command or data), a channel interrupt occurs for the specified channel and chaining is suppressed.

Bit Position 11 is the protection check bit. When set, this bit indicates that the channel tried to store data in a protected main store area. The operation is terminated and a channel interrupt occurs for the specified channel. If chaining (command or data) is in progress, it is suppressed.

Bit Position 12 is the channel data check bit. When set, this bit indicates that a parity error was detected in the channel, in main store, non-addressable main store or in scratch-pad store. Reading of characters with bad parity going into store are replaced with the systems error byte (hexadecimal FF), and the input/output operation is completed. For parity error characters going to a device, (writing) the invalid character is transferred unchanged, the operation is terminated and a channel interrupt occurs for the specified channel. (The transfer of sense byte(s) to store is not checked for parity).

Bit Position 13 is the channel control check bit. When set, this bit indicates that a machine malfunction has occurred affecting the channel controls. Conditions which cause this bit to be set are parity error in the Channel Command Word, data address, or contents of the Channel Command Word. The operation is terminated and a channel interrupt occurs for the specified channel. If chaining (command or data) is in progress, it is suppressed.

Bit Position 14 is reserved for use by the processor.

Bit Position 15 is the termination interrupt bit. When set, this bit indicates that a termination interrupt has been effected.

Note: The channel status byte is reset only when an input/output operation is initiated.

8.5.3 Standard Device Byte

The standard device byte is stored in scratch-pad store in the Assembly/Status register (bit positions 24 to 31) for the appropriate channel.

This byte indicates the status of a device after an input/output operation. It may also indicate a device request interrupt.

The standard device byte is automatically stored when:

1. An input/output interrupt is serviced (request or termination).
2. An input/output operation is attempted and the condition code indicates that status bits are stored (channel status byte, standard device byte).

The standard device byte is defined as follows:

Bit Position 24 is the external device request interrupt pending bit. When set, this bit indicates that a random access device, a data exchange control or a communications device requires servicing.

Bit Position 25 is the termination interrupt pending bit. When set, this bit indicates that a termination interrupt condition exists in an input/output device.

Bit Position 26 is the device busy bit. When set, this bit indicates that the specified device is busy and cannot accept another operation.

Bit Position 27 is the control busy bit. Not applicable.

Bit Position 28 is the device end bit. When set, this bit indicates that the specified device has terminated. Another operation can be accepted by the device if the device busy bit (26) is not set.

Bit Position 29 is the secondary indicator bit. When set, this bit indicates that the specified device has additional indicators to be tested. These indicators can be brought into main store by using the Sense command.

Bit Position 30 is the device inoperable bit. When set, this bit indicates that the specified device is inoperable.

Bit Position 31 is the status modifier bit. This bit is used with chaining (command or data). When set, this bit indicates that the next Channel Command Word is skipped. This bit is set as a result of device termination.

8.5.4 Sense Bytes

The sense byte, or bytes, are brought into main store from an input/output device by using the Sense command. These bytes contain status information for the device referred to.

8.6 CHANNEL SERVICING

The following sections explain in detail the three types of channel servicing which may be performed during input/output operations. They are: servicing a data transfer, end and chain servicing, and interrupt servicing.

Because channel servicing required that the channel utilize main store, scratch-pad store and non-addressable main store (multiplexor devices), normal mode processing is held off until the servicing has been completed. Consequently, channel servicing is time-shared with normal mode processing. Between service requests, normal mode processing is resumed, or another channel is permitted to service its device(s).

Channel servicing for a device on the multiplexor channel (multiplex mode) requires more time than channel servicing for a device on a selector channel. To balance the system's throughput rate, multiplexor channel servicing is segmented to permit selector channel servicing to break-in if any selector channels require servicing. After all selector(s) demanding service have been satisfied, multiplexor servicing is resumed. This technique ensures that the interference to selector channel servicing caused by the multiplexor channel is no greater than that of an additional selector channel.

8.6.1 Servicing a Data Transfer

Once an input/output operation has been initiated, it proceeds under control of the appropriate channel and registers in scratch-pad store and non-addressable main store (multiplexor devices). When an input/output operation has been initiated and the input/output device is ready to send or receive a data byte, it asks the processor for a service request. When the processor honours this service request, servicing of a data transfer occurs.

Because servicing a data transfer requires that the channel utilize main store, scratch-pad store and non-addressable main store (multiplexor devices), normal mode processing is held off until the servicing has been completed. Servicing of a data transfer is time-shared with normal mode processing. Between service requests, processing is resumed, or another channel is permitted to service its device(s).

If a burst mode operation has been initiated to the multiplexor channel, the channel operates similar to a selector and only one device is serviced. Service requests by devices other than the one operating in burst mode are ignored until the multiplexor channel is operating in the multiplexor mode. This occurs at the conclusion of the burst operation when the last data byte has been serviced (prior to interrupt).

Servicing of a data transfer causes the following events to occur (see figure 5).

Block 1. If the service request comes from a device control electronics connected to the multiplexor channel which is operating in the multiplex mode, the processor gets the device address and fetches the appropriate sub-channel registers from non-addressable main store. These registers are placed in processor utility registers in scratch-pad store. (They are not sent to the multiplexor channel registers in scratch-pad store.) If the service request comes from a device control electronics connected to the multiplexor channel which is operating in the burst mode or from a device connected to a selector channel, the appropriate channel registers in scratch-pad store are used to service the data transfer.

FIGURE 5 FUNCTIONAL LOGIC OF SERVICING A DATA TRANSFER

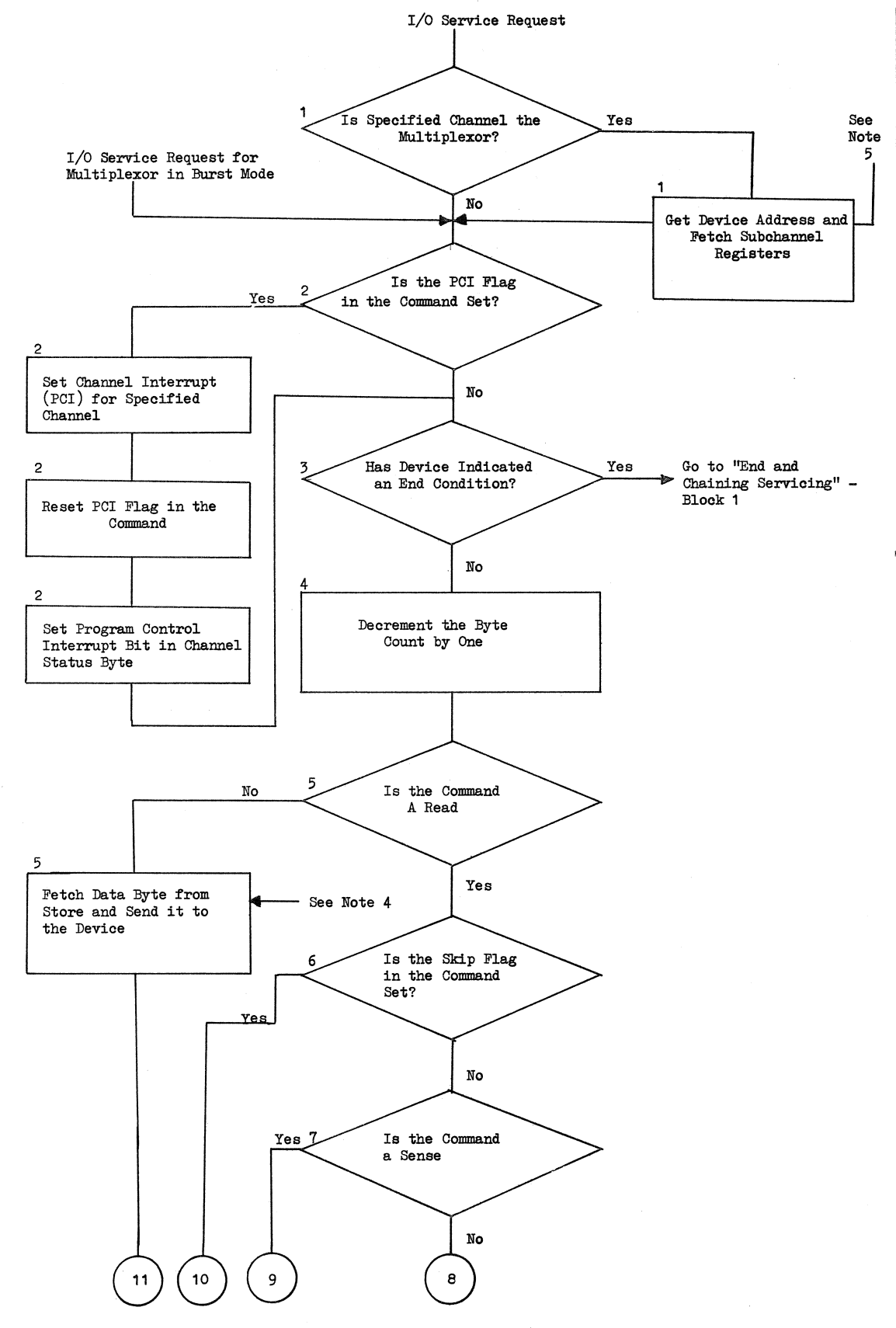
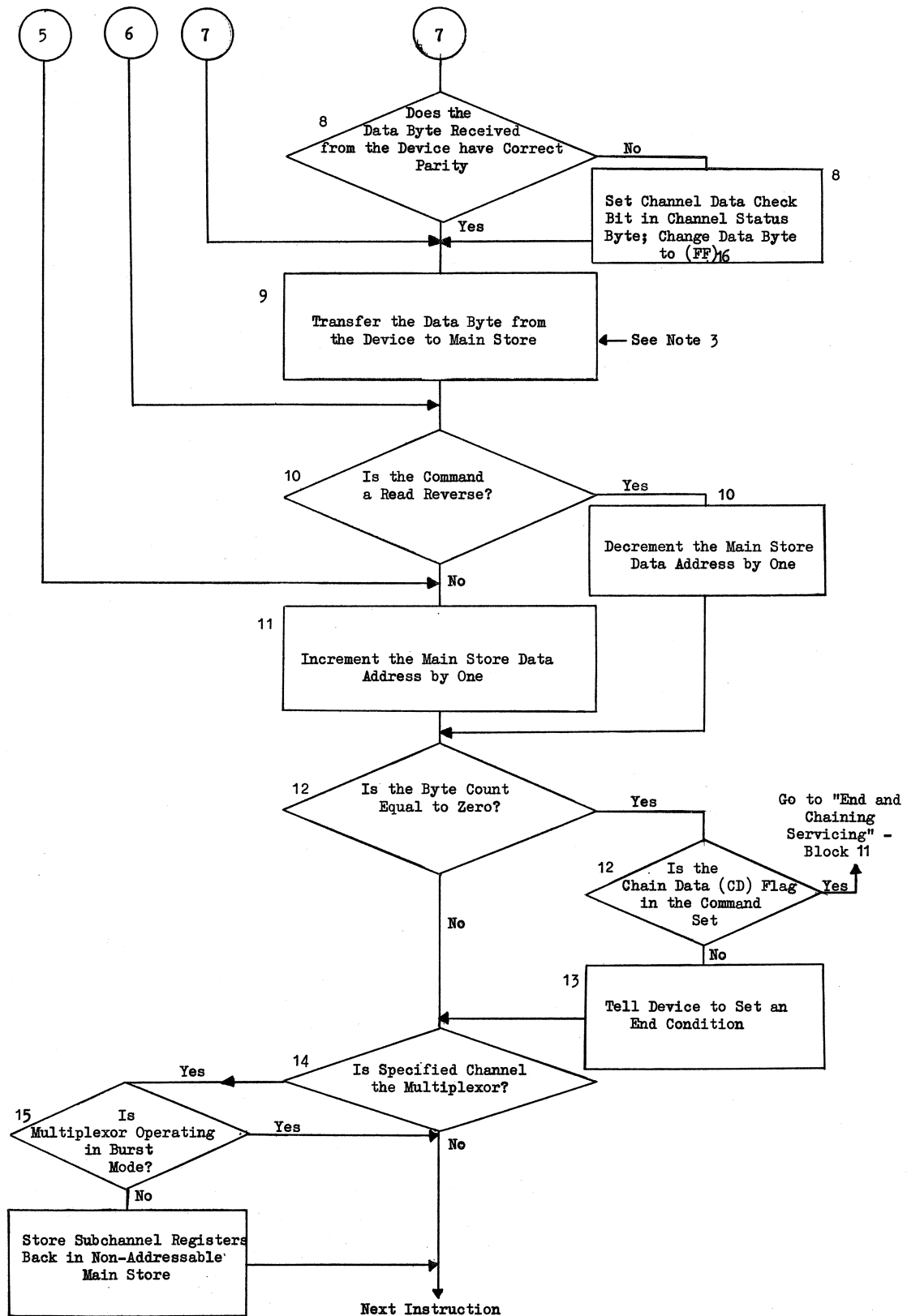


FIGURE 5(cont.)



Block 2. A test is made to see if the Program Controlled Interrupt (PCI) flag is set. If it is, the channel interrupt bit is set in the Interrupt Flag register and an interrupt occurs, if permitted. The PCI flag is reset and the program control interrupt bit is set in the channel status byte.

Block 3. A test is made to see if the device control electronics requesting service has indicated an end condition. An end condition is indicated when one of the following occurs:

1. The processor reaches a byte count lapse. If this occurs, the processor tells the device control electronics to indicate an end condition on the next data service request.
2. The device has completed the input/output operation (that is, a gap is sensed on tape). If this occurs, the device control electronics automatically indicates an end condition. (In this case the byte count is never zero.)

If an end condition has been indicated, the processor goes to End and Chaining Servicing (see figure 6A, Block 1).

Note: Certain error conditions cause the processor to tell the device control electronics to indicate an end condition on the next data service request (see Notes 3, 4, 5, 6 on Servicing a Data Transfer).

Block 4. If the device control electronics has not indicated an end condition, the byte count is decremented by one.

Block 5. A test is made to see if the command is a read. A read command can be any one of the following:

Read Forward
Read Reverse
Sense

All other commands (except Transfer in Channel) are write commands. If the command is a write, the data byte is fetched from main store and sent to the device. Control is then transferred to Block 11.

Block 6. If the command is a read, a test is made to see if the SKIP flag is set. If it is, transfer of the data byte to main store is bypassed and control is transferred to Block 10.

Block 7. If the SKIP flag is not set, a test is made to see if the command is a Sense. If it is, parity checking of the data byte is bypassed and control is transferred to Block 9.

Block 8. If the command is not a Sense, a test is made to see if the data byte received from the device has correct parity. If it does not, the channel data check bit in the channel status byte is set and the data byte is converted to (FF)₁₆. The input/output operation continues.

Block 9. The data byte is transferred to the main store address specified.

Block 10. A test is made to see if the command is a Read Reverse. If it is, the main store address is decremented by one.

Block 11. If the command is not a Read Reverse, the main store address is incremented by one.

Block 12. A test is made to see if the byte count has lapsed. If it has, a test is made to see if the Chain Data flag is set. If it is, the processor goes to End and Chaining Servicing (see figure 6B, Block 11).

Block 13. If the Chain Data flag is not set, the processor tells the device control electronics to indicate an end condition on the next data service request.

Block 14. A test is made to see if the service request was honoured for a device on the multiplexor channel. If it was not, program control continues with the next instruction or with the instruction that was interrupted due to the service request.

Block 15. If the service request was honoured for a device on the multiplexor channel, a test is made to see if it is a burst mode operation. If it is not a burst mode operation the sub-channel registers are sent back to non-addressable main store. In either case, program control continues with the next instruction or with the instruction that was interrupted due to the service request.

Notes on Servicing a Data Transfer.

1. All input/output data service requests are honoured depending on the channel's position in the priority sequence.
2. The following tests occur when a data byte is transferred to main store:
 - (a.) The main store address to which the data byte is to be transferred is tested to see if it is in a store protected area (Store Protect feature must be installed). If it is, the protection check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).
 - (b.) The main store address to which the data byte is to be transferred is tested to see if it is in available main store for the system. If it is not, the program check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).
3. The following tests occur when a data byte is transferred from main store:

- (a.) The main store address from which the data byte is to be transferred is tested to see if it is in available main store for the system. If it is not, the program check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).
 - (b.) The data byte to be transferred is checked for correct parity. If parity is not correct, the data check bit in the channel status byte is set and the device control electronics is told to set an end condition on the next data service request (see Block 13).
- 4. If a main store parity error occurs while fetching the subchannel registers, the channel control check bit in the channel status byte is set, and the device control electronics is told to set an end condition on the next data service request (see Block 13).
 - 5. If a scratch-pad store parity error occurs during the servicing of a data transfer, the channel control check bit in the channel status byte is set and the device control electronics is told to set an end condition on the next data service request (see Block 13).

8.6.2 End and Chaining Servicing

End and chaining servicing is required when the input/output operation specified by the current command has been completed (normally or abnormally). Entry to this servicing always comes from "servicing a data transfer". The following conditions cause end and chaining servicing to take place:

- 1. A device control electronics has indicated an end condition. This end condition is recognized in Servicing a Data Transfer.
- 2. The byte count in the current command has lapsed and the Chain Data (CD) flag in this command is set. If this condition occurs, entry to End and Chaining Servicing occurs at a point which bypasses the normal end servicing with no chaining and the end servicing with command chaining.

For input/output operations that do not specify chaining, end servicing is used so that the processor can tell the appropriate device control electronics to set an interrupt condition. This interrupt condition is in turn reported to the processor and the appropriate flag in the Interrupt Flag register is set, at which time the interrupt is taken, if permitted.

For input/output operations that specify chaining (command or data), this servicing does one of the following:

- 1. If the current command specifies command chaining (the CC flag in the command is set) this service is used to fetch the next command in the chain and to send this new command to the input/output device.

2. If the current command specifies data chaining (the CD flag in the command is set) this service is used to fetch the next command in the chain so that the current operation can be continued.

End and Chaining Servicing causes the following events to occur (see 6).

Block 1. Entry to this block occurs when the input/output device control electronics has indicated an end condition. This end condition is recognized during servicing a data transfer and is generated:

1. automatically by the device, or
2. by the device on command from the processor.

The processor receives the standard device byte from the device control electronics. This standard device byte is used by the processor for testing purposes. It is not stored in the channel registers.

Block 2. The standard device byte is tested to see if the device busy bit is set and the device end bit is reset. This condition normally arises in buffered devices, (that is, card punch, printer) when the buffer has been loaded and the completion of the operation is off-line (no more data has to be sent between the processor and the device control electronics). If this condition exists, the processor tells the device to set another end condition and ask for another service request when the device is no longer busy. Control is then transferred to Block 14.

Block 3. If the device is not busy, a test is made to see if the Chain Command (CC) flag is set. If it is not, control is transferred to Block 8 which causes termination of the command to occur.

Block 4. If the Chain Command (CC) flag is set, a test is made to see if one of the following bits is set in the channel status byte:

Program Check bit

Protection Check bit

Data Check bit (This bit is checked only if the current operation is a write)

Channel Control Check bit

If any of the above bits are set (except the data check bit on a Read) control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 5. If none of the bits tested in the channel byte are set, a test is made to see if the Chain Data (CD) flag is set. If the Chain Data flag is set, control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 6. If the Chain Data (CD) flag is not set the standard device byte is tested to see that the following conditions are present:

FIGURE 6 FUNCTIONAL LOGIC OF END AND CHAINING SERVICING

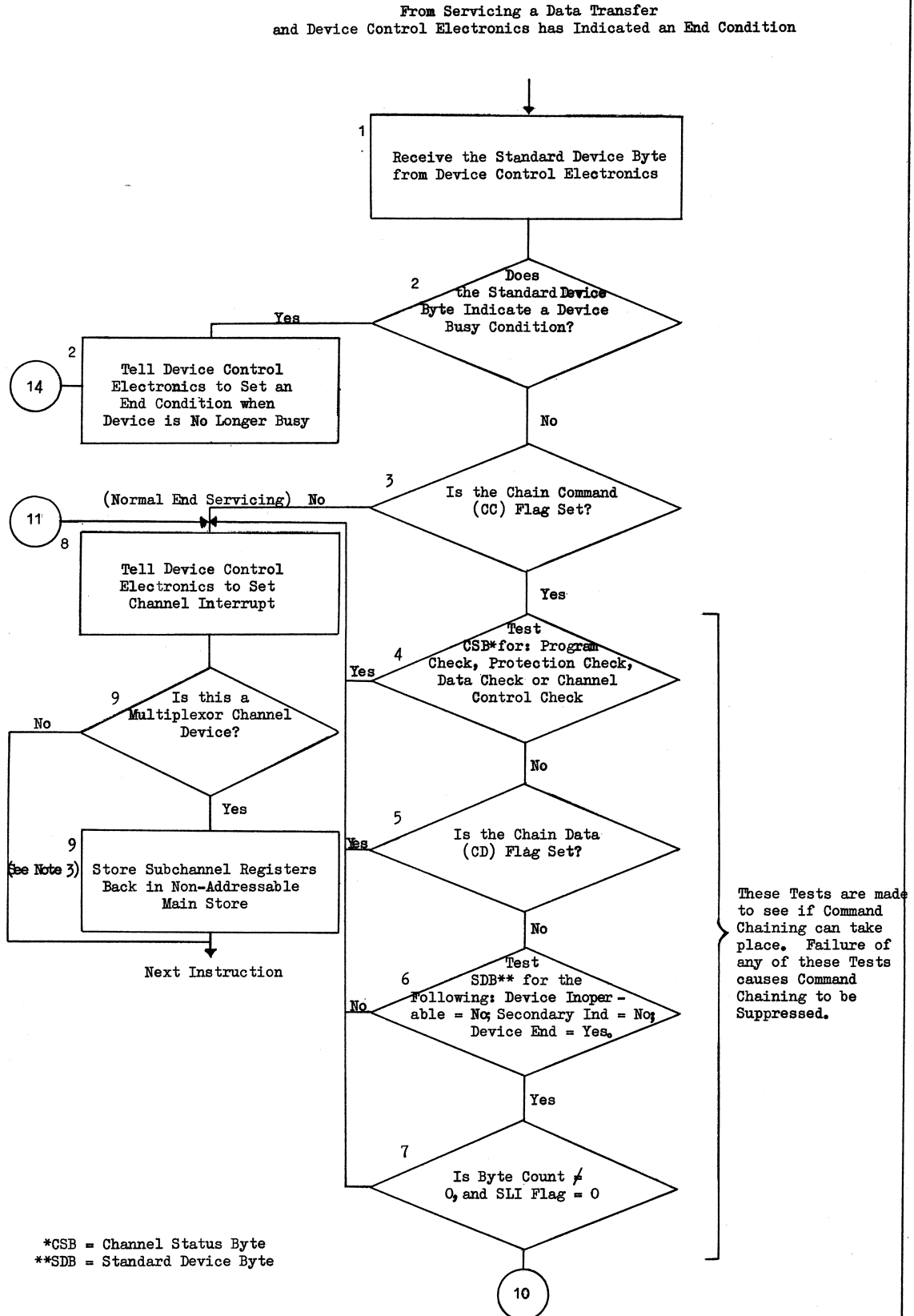
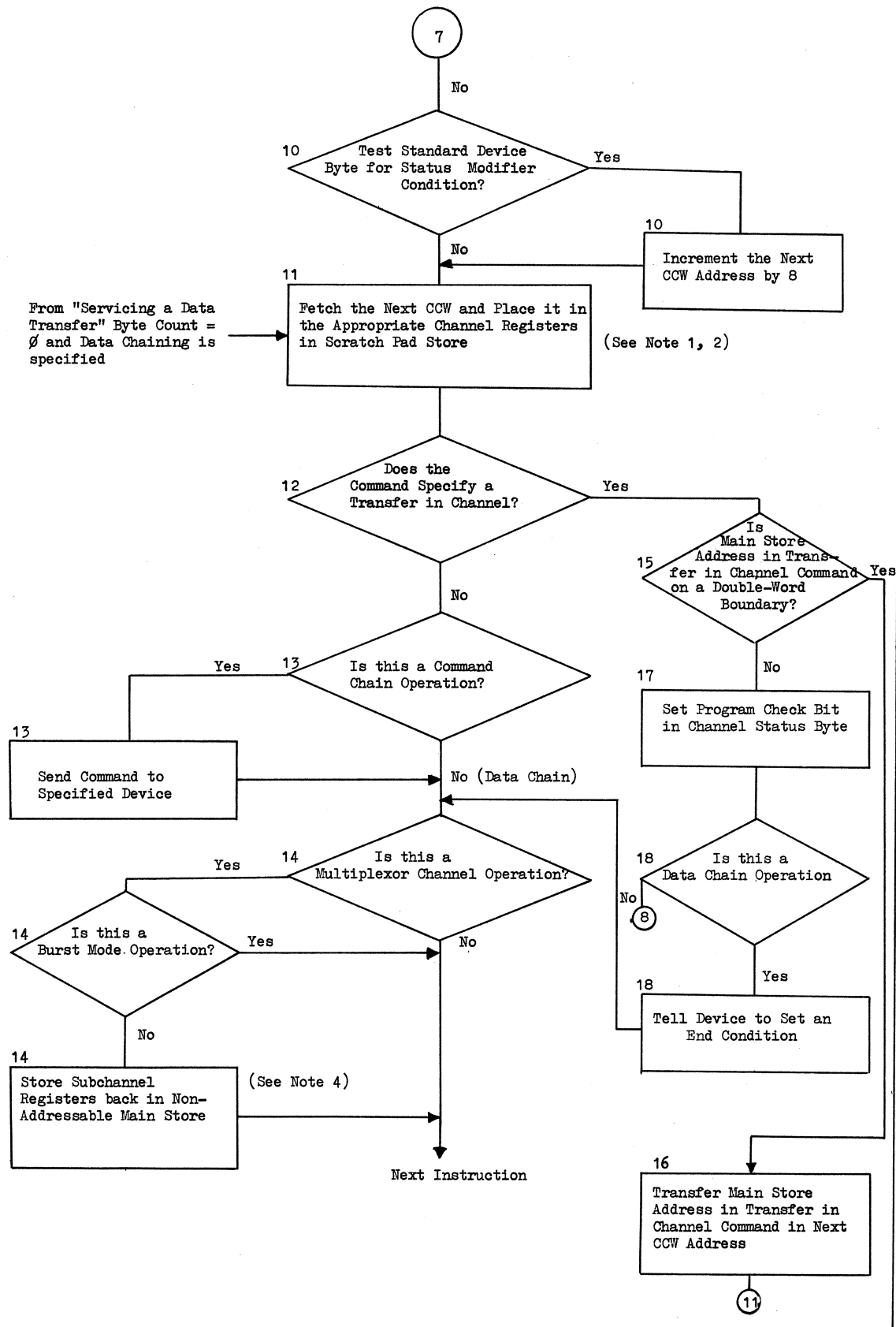


FIGURE 6 (cont)



Device is operable
Secondary indicator is not set
Device end is set

If any of the above conditions is not present, control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 7. If all of the conditions tested in the standard device byte are present, a test is made to see if the byte count is not equal to zero and the Suppress Length Indicator (SLI) flag is equal to zero. If these conditions are present, the program desires an indication of incorrect length, and control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 8. Entry to this block occurs under the following conditions:

- (a) A device control electronics has indicated an end condition, the device is not busy and the chain command flag bit is not set.
- (b) A device control electronics has indicated an end condition and the chain command flag is set. However, a condition is present which causes command chaining to be suppressed.

The processor tells the device control electronics to set a channel interrupt condition for the appropriate channel.

Block 9. A test is made to see if the device is on the multiplexor channel. If it is, the subchannel registers are sent back to non-addressable main store. In either case, program control continues with the next instruction or with the instruction that was interrupted due to chaining and/or end servicing.

Note: If the operation that was terminated was a burst mode operation, the burst mode is completed at this point and other multiplex mode operations can be directed to devices on the multiplexor channel. The processor does not have to wait for the burst mode terminating interrupt to occur.

Block 10. Entry to this block occurs when command chaining is to take place. The standard device byte is tested to see if the status modifier bit is set. If it is, the next Channel Command Word (CCW) address is incremented by eight. (The next Channel Command Word in sequence is skipped).

Block 11. In addition to continuing command chaining processing, entry to this block occurs from Servicing a Data Transfer when the following conditions are present:

- (a) The byte count is equal to zero.
- (b) The Chain Data (CD) flag is set.

The next Channel Command Word (CCW) is fetched from main store and placed in the appropriate channel registers. The next Channel Command Word address is incremented by eight.

Block 12. A test is made to see if the next command in sequence is a Transfer in Channel command.

Block 13. If the command is not a Transfer in Channel command, a test is made to see if this is a command chain or a data chain operation. If it is a command chain operation, the new command is sent to the specified device control electronics. (This is not required if this is a data chain operation.)

Block 14. A test is made to see if the chaining servicing has occurred for a device on the multiplexor channel. If it has, a test is made to see if it is a burst mode operation. If it is not a burst mode operation, the subchannel registers are sent back to non-addressable main store. In all cases, program control continues with the next instruction, or with the instruction that was interrupted due to the chaining servicing.

Block 15. If the next command in sequence is a Transfer in Channel command, the main store address specified by the Transfer in Channel command is tested to see if it is on a double-word boundary.

Block 16. If the main store address specified in the Transfer in Channel command is on a double word boundary, this address is placed in the next Channel Command Word address and control is transferred to Block 11 which fetches the CCW specified by the Transfer in Channel command.

Block 17. If the main store address specified in the Transfer in Channel command is not on a double word boundary, the program check bit is set in the channel status byte.

Block 18. A test is made to see if this is a data chain operation. If it is, the device is told to set an end condition on the next data service request and control is transferred to Block 14 to complete the end servicing. If this is a command chain operation (the device has already indicated an end condition) control is transferred to Block 8 where the device control electronics is told to set an interrupt condition.

Notes on End and Chaining Servicing:

1. The following test occurs when the next Channel Command Word is fetched:

The main store address specified is tested to see if it is in available main store for the system. If it is not, the program check bit in the channel status byte is set; and, if data chaining, the device is told to set an end condition on the next data service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

2. If a main store parity error occurs when fetching the next Channel Command Word, the channel control check bit in the channel status byte is set; and, if data chaining, the device control electronics is told to set an end condition on the next data service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

3. If a scratch-pad store parity error occurs when storing the subchannel registers back in non-addressable main store the channel control check bit in the channel status byte is set.
4. If a scratch-pad store parity error occurs when storing the subchannel registers back in non-addressable main store, the channel control check bit in the channel status byte is set; and, if data chaining, the device control electronics is told to set an end condition on the next service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

8.6.3 Interrupt Servicing

Interrupt servicing occurs when the appropriate flag in the Interrupt Flag register has been set, and the Interrupt Mask register for the current state permits the interrupt and it is taken. This service is required to:

1. Obtain the standard device byte from the device control electronics (if applicable) and store it in the appropriate input/output channel registers.
2. Fetch the appropriate subchannel registers from non-addressable main store if the interrupt is due to a multiplexor channel device. The subchannel registers are stored in the multiplexor channel registers in scratch-pad store.

There are three kinds of channel interrupts. They are as follows:

Programmed Control Interrupt This interrupt occurs when a Channel Command Word is fetched and the program controlled interrupt flag bit is set. This interrupt condition has no effect upon the input/output operation specified by the Channel Command Word. The standard device byte and the subchannel registers are not stored.

Device Request Interrupt This interrupt occurs as a result of a condition arising in an input/output device control electronics. It may occur independently of a processor initiated input/output operation. Examples of this type of interrupt are as follows:

1. A remote processor wishes to send data via a Data Exchange Control. The Data Exchange Control initiates the channel interrupt. (This interrupt occurs independent of a processor initiated input/output operation).
2. The processor initiates an off-line seek to a random access device. When the seek is complete, the random access device control electronics initiates a channel interrupt. (This interrupt occurs in conjunction with a processor initiated input/output operation).

When an external device request interrupt occurs, the standard device byte and the subchannel registers (if a multiplexor device) are stored in the appropriate input/output channel registers.

Terminating Interrupt This interrupt occurs when an input/output operation initiated by the processor has terminated. When this interrupt occurs, the standard device byte and the subchannel registers (if a multiplexor device) are stored in the appropriate input/output channel registers. This is the final servicing of the channel and device. At the completion of this servicing, the channel is free to accept another operation. The contents of the input/output channel registers must be utilized by the program before another operation is initiated. (When another operation is initiated, the contents of these registers are altered.) The following information is available in the input-output channel registers for interrogation by the program.

Channel status byte
Standard device byte
Byte count
Address of next CCW
Low-order 4 bits of the command code
Device number

Interrupt servicing causes the following events to occur (see figure 7).

Block 1. The device control electronics is asked for the address of the device requiring interrupt servicing.

Block 2. A test is made to see if the device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the processor generates a standard device byte of all zeros. Control is then transferred to Block 4.

Block 3. If the device control electronics is operable, it sends the standard device byte to the processor.

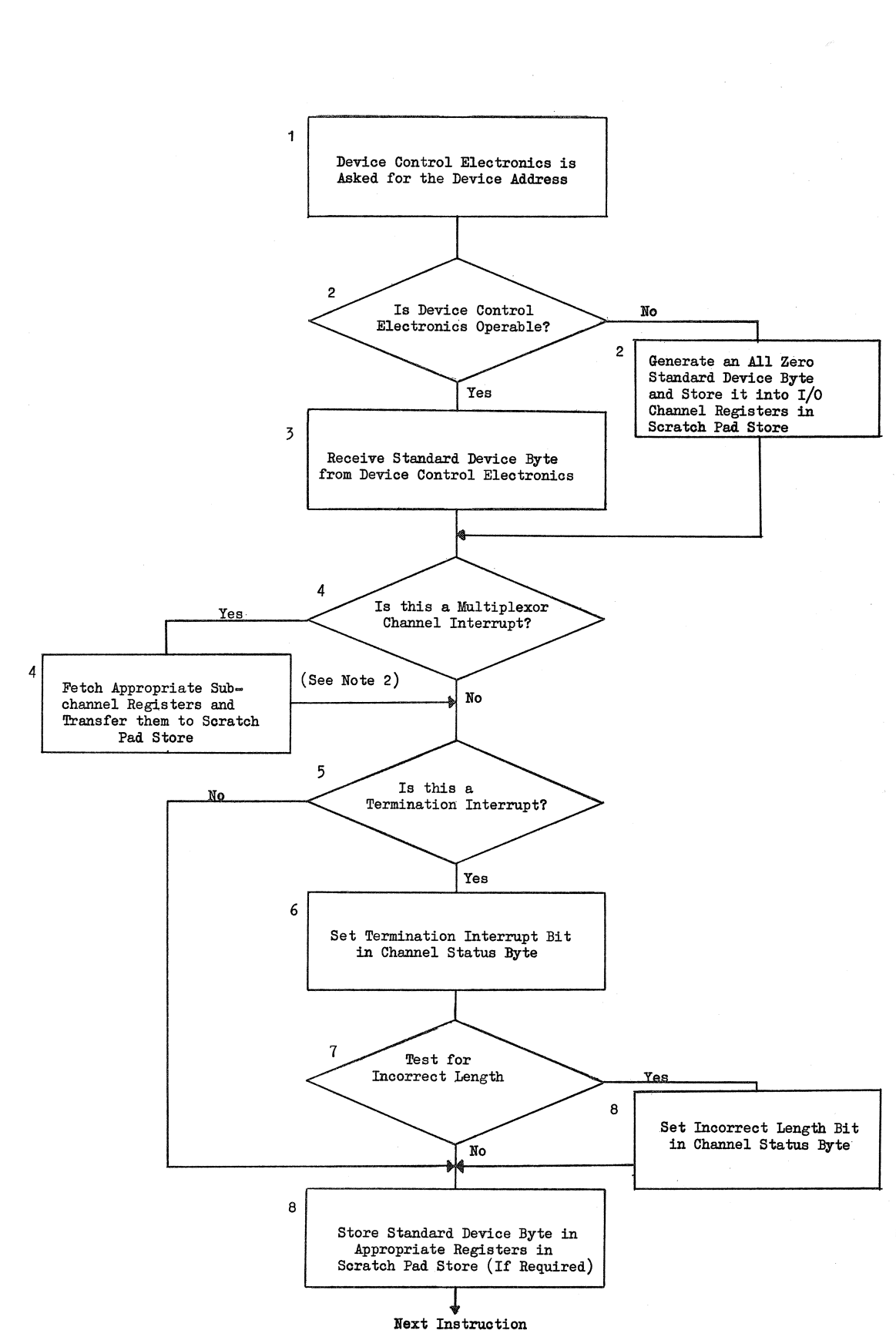
Block 4. If the service request comes from a device control electronics connected to the multiplexor channel, the processor uses the device address to fetch the appropriate subchannel registers in non-addressable main store. The subchannel registers are stored in the input/output channel registers in scratch-pad store for the multiplexor channel.

Block 5. A test is made to see if this is a terminating interrupt. If it is not (it is a program controlled or a device request interrupt) control is transferred to Block 8.

Block 6. If the interrupt is a terminating interrupt, the termination interrupt bit in the channel status byte is set.

Block 7. A test is made to see if the byte count is not equal to zero and the Suppress Length Indicator (SLI) flag is equal to zero. If these conditions are present, the program desires an indication of incorrect length bit in the channel status byte is set.

FIGURE 7 FUNCTIONAL LOGIC OF INTERRUPT SERVICING



Block 8. The standard device byte is stored in the appropriate input/output channel registers and program control continues with the next instruction.

Notes on Interrupt Servicing:

1. The device address is always stored in the input/output channel registers in scratch-pad store if the interrupt is due to a device connected to the multiplexor channel. If the interrupt is due to a device on a selector channel, the device address is stored only if it is a device request interrupt.
2. If a main store parity error occurs when fetching the subchannel registers, the channel control check bit in the channel status byte is set.

9. Program interrupt

9.1 INTRODUCTION

Program interrupts occur as a result of errors in data or instruction specifications, input/output operations, external signals, equipment malfunctions or arithmetic errors. The instruction being executed at the time of the interrupt can be completed, suppressed, or terminated depending on the cause of the interrupt.

An interrupt can be inhibited or permitted in any state through programming. If an interrupt occurs and is permitted, conditions existing in the interrupted state are automatically stored. Control is then passed to the Interrupt Control State P3 or Machine Condition State P4, depending on the cause of the interrupt. (See Processor States below.) The priority of the interrupt is established and an analysis is made to determine the proper linkage to the Interrupt Response State P2 so that the interrupt may be processed. After interrupt processing is completed, control is returned to the state which was last interrupted, and normal processing is resumed.

If several interrupts occur at the same time, the one having the highest priority is processed. The remaining interrupts are processed in turn, depending on their priority.

9.2 PROCESSOR STATES

The 4-50 has four processor states that provide control of system and program interrupts. Programs can be executed in any one of the states, because each state is completely independent and has its own set of registers. The processor states and their functions are as follows:

Processing State P1 The Processing State P1 interprets and executes the user's program. This processing state is the problem-oriented state.

Interrupt Response State P2 The Interrupt Response State P2 performs specific program tasks as dictated by the Interrupt Control State P3.

Interrupt Control State P3 The Interrupt Control State P3 is automatically entered when an interrupt is recognized that is other than one caused by a machine check or power failure. In this state, programming is responsible for performing a detailed analysis of the cause of the interrupt and establishing its priority. After these functions are performed, linkage is provided to the related interrupt processing routine in the Interrupt Response State P2.

Machine Condition State P4 The Machine Condition State P4 is entered whenever a machine check or power failure occurs. In this state, programming analyses the cause of a machine interrupt and establishes its priority so that an indication of the cause of interrupt can be given to the operator.

9.3 PROCESSOR STATE REGISTERS

Registers are provided in the scratch-pad store, for each processor state as given in the table below:

Table 1: Processor State Registers

Register	State			
	P1	P2	P3	P4
Program Counter	1	1	1	1
General Registers	16	16	6	5
Floating-Point Registers	4	*	*	*
Interrupt Status Register	1	1	1	1
Interrupt Mask Register	1	1	1	1

*Floating-point instructions executed in any of the processor states use the floating-point registers assigned to P1.

Because each processor state has its own general registers, Interrupt Status Register and Interrupt Mask Register, storing and reloading these registers is not necessary during interrupt processing.

9.3.1 Program Counter

The Program Counter (P counter) is a 32-bit register that is located in the scratch-pad store. A separate P counter is provided for each of the four processor states.

The format of the P counter is as follows:

IILC	CC	Program Mask	Next Instruction Address
0	1 2 3 4	7 8	31

Bit Positions 0 and 1 contain the Instruction Length Code. When an interrupt occurs and is taken, or a Program Control instruction is executed, the length of the last instruction executed in the terminated state, before the interrupt condition occurred, is stored in bit positions 0 and 1 as given in the table below. The Instruction Length Code is always generated from the operation code of the instruction.

Table 2: Instruction Length Codes

IILC	Length in Bytes
01	Two-byte instruction
10	Four-byte instruction
11	Six-byte instruction

Notes

1. If the interrupt condition is an operation code trap, the length of the instruction causing the interrupt is generated from the operation code and is stored in bit positions 0 and 1 as given in table 2.
2. The instruction length code is unpredictable if the interrupt was caused by one of the following:
 - Power Failure
 - Machine Check
 - Address Error (only if the address error was caused by an invalid instruction address)

Bit Positions 2 and 3 contain the condition code. When an interrupt occurs or a Program Control instruction is executed, the condition code is moved from a machine register, where it is maintained for instruction execution, and stored in this field of the P counter of the state being terminated. The condition code in this field of the P counter of the state being initiated is moved into a machine register where it is maintained for possible future use.

Bit Positions 4 to 7 contain the program mask. When an interrupt occurs or a Program Control instruction is executed, the program mask is moved from the machine register, where it is maintained for instruction execution, and stored in bits 4 to 7 of the P counter of the state being terminated. The program mask in this field of the P counter of the state being initiated is moved into the machine register where it is maintained for possible future use.

Bit Positions 8 to 31 contain the next instruction address. This field stores the address of the next instruction in the main store to be staticized by the appropriate processor state. Each time an instruction is staticized, the P counter is updated to the next instruction. This field is left intact whenever an interrupt requires switching to a new processor state.

9.3.2 General Registers

A separate set of general registers is assigned to each processor state. Each general register is 32 bits long. Sixteen general registers are assigned to P1 and P2, six general registers are assigned to P3 and five general registers are assigned to P4. These registers serve as operands, base address registers, or index registers.

9.3.3 Floating-Point Registers

Four floating-point registers are provided. Each floating-point register is 64 bits long (double length). These registers are used only in floating-point arithmetic. The floating-point registers can be used by any of the processor states.

9.3.4 Interrupt Status Registers

The Interrupt Status register is a 32-bit register. A separate register is provided for each of the four processor states.

The format of each Interrupt Status register is as follows:

ISI		000		PI		KEY		A		EB		N		00000000				Call	
0	2	3	5	6	7	8	11	12	13	14	15	16				23	24		31

Bit Positions 0 to 2 contain the Interrupt State Identifier. When an interrupt occurs, the number of the processor state being interrupted is stored in this field of the processor state being initiated as given in table 3.

Table 3. Interrupt State Identifier Codes

ISI	Definition
000	P4 was interrupted
001	P3 was interrupted
010	P2 was interrupted
011	P1 was interrupted

Bit Positions 3 to 5 are not used and must be zeros.

Bit Positions 6 and 7 contain the program indicators. When an interrupt occurs due to a parity error in Main Store or Scratch Pad Store the program indicators are stored in this field in P4 as given in Table 4.

Table 4. Program Indicator Codes

Program Indicators	Definition
00	Neither error has occurred
01	Scratch Pad Store parity error has occurred
10	Main Store parity error has occurred
11	Scratch Pad Store parity error and Main Store parity error have occurred.

Bit Positions 8 to 11 contain the store protection key. This field is set by the program to indicate the desired protection key. When an interrupt occurs or a Program Control instruction is executed, the store protection key is extracted from this field of the processor state being

initiated and placed in a machine register where it performs the store protect function. The four-bit key provides a possible 15 keys ranging from (1)₁₆ to (F)₁₆. Each 2,048-byte block of main store has its individual machine register for the protection key. When the key related to the current processor state and the key related to the main store block are equal, or either is zero, the main store block accepts a data store. Conversely, if the keys do not match, and neither is zero, an address error (protection) interrupt occurs.

Note: If the store protect feature is not installed, this field must be zero.

Bit Position 12 designates the internal decimal code. When an interrupt occurs or a Program Control instruction is executed, the decimal code (either ISO or EBCDIC) for the processor state being initiated is established by the setting of this bit. If the bit is 1, ISO Code is established; if the bit is 0, EBCDIC is established.

Note: The setting of this Decimal Code does not affect any automatic translation of data read into or written from the processor. The Decimal Code is used to determine what zone configuration (ISO or EBCDIC) is to be established internally when executing the decimal arithmetic instruction set, the Edit instruction, and the Edit and Mark instruction.

Bit Positions 13 and 14 are used when an Emulator feature is included in the system. If an Emulator feature is not installed, this field must be zero or an address error interrupt occurs.

Bit Position 15 is the non-privileged mode bit. This field is set by the program to indicate the privileged status of the processor state being initiated. If N = 0, the initiated processor state runs in the privileged mode, allowing execution of the privileged instructions; if N = 1, the processor state runs in the non-privileged mode, inhibiting the execution of the privileged instructions.

Bit Positions 16 to 23 are not used and must be zeros.

Bit Positions 24 to 31 is the call field. This field is set during the execution of a Supervisor Call instruction. The R1 and R2 field of this instruction provide a code which is placed into the call field of the Interrupt Status register of the processor state in which the Supervisor Call instruction is issued. This code provides linkage to the program required to accomplish the purpose of the Supervisor Call instruction.

9.3.5 Interrupt Mask Registers

The Interrupt Mask register is a 32-bit register. A separate register is provided for each of the four processor states. Each bit in the Interrupt Mask register is associated with an interrupt condition. A 0 bit in any bit position in this register inhibits the associated interrupt condition; a 1 bit in any bit position in this register permits the associated interrupt condition.

Notes

1. The Power Failure and Machine Check interrupts must be inhibited in the Machine Condition State P4. The mask bits in the Interrupt Mask register for these interrupt conditions must always be zero. This is a program restriction.
2. The Address Error interrupt must be inhibited in the Interrupt Control State P3. The mask bit in the Interrupt Mask register for this interrupt condition must always be zero. This is a programming restriction.

9.3.6 Program Mask Registers

In addition to the Interrupt Mask register, a Program Mask register is also provided for each state. The Program Mask register is not contained in the main store or the scratch-pad store. It is a separate machine register which is set by the non-privileged instruction, Set Program Mask, and it applies to the following interrupt conditions:

Significance error.

Exponent underflow.

Decimal overflow.

Fixed-point overflow.

The program mask bit settings have priority over the bit settings in the Interrupt Mask register for the above four program interrupts. A 0 bit in any bit position in this register cancels the interrupt condition if it occurs. A 1 bit in any bit position in this register indicates that the Interrupt Mask register is to be examined. If an interrupt condition occurs and is inhibited by the Interrupt Mask register, it remains pending until it is serviced (permitted).

9.3.7 Register Addressing

Register addressing in each of the processor states is given in Table 5.

Table 5. Register Addressing in the Processor States

Register Number	Processor States			
	P1	P2	P3	P4
0	GR	GR	IMR, P1 State	Processor Utility
1	GR	GR	ISR, P1 State	Processor Utility
2	GR	GR	P counter, P1 State	Processor Utility
3	GR	GR	Interrupt Flag Register	Processor Utility
4	GR	GR	IMR, P2 State	Processor Utility
5	GR	GR	ISR, P2 State	Processor Utility
6	GR	GR	P counter, P2 State	Processor Utility
7	GR	GR	GR	Processor Utility
8	GR	GR	IMR, P3 State	GR
9	GR	GR	ISR, P3 State	GR
10	GR	GR	P counter, P3 State	GR

Register Number	Processor States			
	P1	P2	P3	P4
11	GR	GR	GR	GR
12	GR	GR	GR	IMR, P4 State
13	GR	GR	GR	ISR, P4 State
14	GR	GR	GR	P counter, P4 State
15	GR	GR	GR/Weight	GR/Weight

GR = General Register
IMR = Interrupt Mask Register
ISR = Interrupt Status Register

Notes

1. The P counter, Interrupt Status register, and Interrupt Mask register for processor state P1, P2 and P3 can be addressed by register notation (R1,R2 or R3 field of an instruction) in processor state P3 only. The P counter, ISR and IMR for processor state P4 can be addressed by register notation in processor state P4 only. Because the P counter, the ISR's and the IMR's are contained in the scratch-pad store, they can be addressed in any of the processor states by using the Load Scratch Pad instruction and the Store Scratch Pad instruction. However, these instructions are privileged instructions and the processor state in which they are executed must be running in the privileged mode. (Bit position 15 of the appropriate Interrupt Status register must be set to zero.)
2. Floating-Point registers may be addressed by floating-point instructions only, and are addressed as 0, 2, 4 and 6 in all processor states.

9.3.8 Interrupt Flag Register

The Interrupt Flag register is a 32-bit register. There is only one Interrupt Flag register. When an interrupt condition occurs, a bit associated with the specific interrupt is set in the Interrupt Flag Register. If the corresponding bit in the Interrupt Mask register for the current state is set, an interrupt occurs.

If the interrupt condition is one of the four program interrupts, the corresponding bit in the Program Mask register must also be set to cause an interrupt.

The Interrupt Flag register is scanned on a priority basis and the highest priority interrupts are serviced first. Each interrupt condition is assigned a specific weight which is put into the rightmost eight bits of General register Number 15 of the initiated state (P3 or P4). This weight can be used by the program to enter the proper interrupt routine.

General register number 15 in P3 or P4 is cleared and reloaded each time an interrupt occurs.

Table 6 lists the priority, the Interrupt Flag register position, the program state initiated, and the weight of each of the interrupt conditions.

Table 6. Interrupt Conditions and Priority

Priority	Interrupt Condition	Flag *Bit	State Initiated	Weight
1	Power Failure	2^0	P4	0
2	Machine Check	2^1	P4	4
3	External Signal No. 1	2^2	P3	8
4	External Signal No. 2	2^3	P3	12
5	External Signal No. 3	2^4	P3	16
6	External Signal No. 4	2^5	P3	20
7	External Signal No. 5	2^6	P3	24
8	External Signal No. 6	2^7	P3	28
9	Not Specified	2^8	P3	32
10	Selector Channel No. 1	2^9	P3	36
11	Selector Channel No. 2	2^{10}	P3	40
12	Selector Channel No. 3	2^{11}	P3	44
13	Selector Channel No. 4	2^{12}	P3	48
14	Selector Channel No. 5	2^{13}	P3	52
15	Selector Channel No. 6	2^{14}	P3	56
16	Multiplexor Channel	2^{15}	P3	60
17	Elapsed Time Clock	2^{16}	P3	64
18	Console Interrupt Request	2^{17}	P3	68
19	Not Specified	2^{18}	P3	72
20	Not Specified	2^{19}	P3	76
21	Supervisor Call Instruction	2^{20}	P3	80
22	Privileged Operation	2^{21}	P3	84
23	Op-Code Trap	2^{22}	P3	88
24	Address Error (Protect, Addressing, Specification)	2^{23}	P3	92
25	Data Error	2^{24}	P3	96
26	Exponent Overflow	2^{25}	P3	100
27	Divide Error	2^{26}	P3	104

Priority	Interrupt Condition	Flag *Bit	State Initiated	Weight
28	Significant Error**	2^{27}	P3	108
29	Exponent Underflow**	2^{28}	P3	112
30	Decimal Overflow**	2^{29}	P3	116
31	Fixed Point Overflow**	2^{30}	P3	120
32	Test Mode	2^{31}	P3	124

* 2^0 = The rightmost bit in the Interrupt Flag register.

**Note: These interrupt conditions can be masked by two separate masks. The first, the program mask, is a four-bit, non-privileged mask which can be set by program, that can be used to cancel the interrupt condition when it occurs. The second mask is composed of bits 2^{30} to 2^{27} of the 32-bit Interrupt Mask register associated with the state in which the processor is operating. If the Program Mask prohibits the interrupt it is cancelled. If the Program Mask permits the interrupt, the Interrupt Mask register is scanned. Like all the other interrupt conditions, the masks of the 32-bit Interrupt Mask register leave these four interrupt conditions pending if the associated mask bits are zeros.

9.4 INTERRUPT CONDITIONS

A description of the individual interrupt conditions is given in table 7. More detailed information concerning the interrupt conditions is given in the instruction descriptions. Some interrupt conditions arise from input/output channel operations, and these conditions are further discussed in the Input/Output Operational Control section.

Note: When an interrupt condition occurs, the current instruction can be suppressed or it can be terminated. When an instruction is suppressed, the condition code setting that existed before the instruction was attempted remains unchanged. Data in main memory and the general registers specified by the instruction also remain unchanged. When an instruction is terminated, the condition code setting and data in the general registers and/or main store are unpredictable.

Table 7 Interrupt Conditions

Priority No.	Condition	Flag Bit	Explanation
1	Power Failure	2^0	A power failure interrupt occurs when there is a power failure in the processor or main store caused by a line failure. Any instruction being executed at the time of interrupt is terminated. It is a program restriction that the mask bit in processor state P4 for this interrupt condition must always be zero when this interrupt occurs. This permits the program to operate in processor state P4 for the purpose of closing down the machine during a one-millisecond interval between power failure and actual power loss to the system.
2	Machine Check	2^1	The machine check interrupt occurs when a machine fault or malfunction is detected. Any instruction being executed at the time of interrupt is terminated. It is a program restriction that the mask bit in processor state P4 for this interrupt condition must always be zero when this interrupt occurs. The following conditions can cause a machine check interrupt to occur: Scratch-Pad Store Parity Error - This error can occur when data is read from the Scratch-Pad Store. Main Store Parity Error - This error can occur when data or instructions are read from main store. If a main store parity error occurs during an I/O data transfer, this interrupt condition does not occur. A channel interrupt occurs and the program is notified of the condition via the channel status byte.

Priority No.	Condition	Flag Bit	Explanation
3 4 5 6 7 8	External Signal No.1 External Signal No.2 External Signal No.3 External Signal No.4 External Signal No.5 External Signal No.6	² ₃ ² ₄ ² ₅ ² ₆ ² ₇ ² ₇	The external signal interrupt occurs when a signal is received on an external line (1-6) associated with the Direct Control option. Any instruction being executed at the time of interrupt goes to completion.
9	Not Used	⁸ ₂	
10 11 12 13 14 15 16	Selector Channel No.1 Selector Channel No.2 Selector Channel No.3 Selector Channel No.4 Selector Channel No.5 Selector Channel No.6 Multiplexor Channel	⁹ ₂ ¹⁰ ₁₁ ¹¹ ₁₂ ¹² ₁₃ ¹³ ₁₄ ¹⁴ ₁₅ ¹⁵ ₂	This interrupt provides the means by which the processor can receive and act upon signals from input/output devices connected to a Selector Channel (1-6) or the Multiplexor Channel. This interrupt can occur as a result of the termination (normal or abnormal) of an input/output operation or at the request of an input/output device. It can also occur as the result of a program controlled interrupt. Any instruction being executed at the time of interrupt goes to completion. (Selector Channels are optional.)
17	Elapsed Time Clock	¹⁶ ₂	This interrupt occurs when the Elapsed Time Clock counts downward from positive to negative, indicating that its maximum range has been reached. Any instruction being executed at the time of interrupt goes to completion. (The Elapsed Time Clock is an option.)
18	Console Interrupt Request	¹⁷ ₂	This interrupt is controlled by the Console Interrupt key on the operator's console. Any instruction being executed at the time of interrupt goes to completion.

Priority No.	Condition	Flag Bit	Explanation
19	Not Used	2^{18}	
20	Not Used	2^{19}	
21	Supervisor Call	2^{20}	This interrupt results from the execution of the Supervisor Call instruction. The P counter and the Interrupt Status register of the interrupted state are updated normally. The rightmost eight bits of the Interrupt Status register of the state in which the instruction is executed receives the R1,R2 field of the Supervisor Call instruction.
22	Privileged Operation	2^{21}	This interrupt occurs when a privileged instruction is attempted and the current processor state is in non-privileged mode. (Bit position 15 of the Interrupt Status register is set.) The instruction is suppressed. The privileged instructions are: Diagnose Start Device Test Device Halt Device Check Channel Program Control Load Scratch Pad Store Scratch Pad Idle Set Storage Key } If the Store Protect option is installed. Insert Storage Key } Write Direct } If the Direct Control option is installed. Read Direct }

Priority No.	Condition	Flag Bit	Explanation								
23	Operation Code Trap	2 ²²	This interrupt occurs when an operation code that is either not assigned or not available on the processor is attempted. No operation is performed. The length of the instruction upon which the trap occurred is determined by the instruction length code field of the P counter of the terminated state as follows: <table><tr><td>ILC</td><td>Length in Bytes</td></tr><tr><td>01</td><td>Two-byte instruction</td></tr><tr><td>10</td><td>Four-byte instruction</td></tr><tr><td>11</td><td>Six-byte instruction</td></tr></table> Note: The ILC is always generated from the operation code of the instruction.	ILC	Length in Bytes	01	Two-byte instruction	10	Four-byte instruction	11	Six-byte instruction
ILC	Length in Bytes										
01	Two-byte instruction										
10	Four-byte instruction										
11	Six-byte instruction										
24	Address Error	2 ²³	Three conditions cause an address error interrupt to occur. They are : address error, specification error, and protection error. Addressing - An address error (addressing) interrupt occurs when: 1. An address specifies any part of data, an instruction or control word outside the available main store for the particular installation. The instruction operation is terminated for an invalid data address, and the results of the instruction are unpredictable. The instruction operation is suppressed for an invalid instruction address. 2. An Execute instruction specifies another Execute instruction to be performed. The operation is suppressed.								

Priority No.	Condition	Flag Bit	Explanation
24	Address Error (Cont'd)	2 ²³	3. The first operand address field of an instruction designates an odd register address for a pair of general registers that contain a double word operand. The operation is suppressed. 4. A floating-point instruction addresses a floating-point register other than 0,2,4,6. The operation is suppressed. Specification - An address error (specification) interrupt occurs when: 1. A data, instruction, or control word address does not specify a doubleword, word, halfword, or byte boundary as required by the particular instruction concerned. The operation is suppressed. 2. The multiplier or divisor in decimal arithmetic exceeds 15 digits and sign. The operation is suppressed. 3. The first operand field is not longer than the second operand field in decimal division or multiplication. The operation is suppressed. 4. Bit positions 28 to 31 of the second operand of a Set Storage Key or Insert Storage Key instruction are not zero. The operation is suppressed. 5. The Store Protect option is not installed and the protection key in the Interrupt Status register is not zero. The operation is suppressed. 6. The Program Control instruction specifies an instruction address which is not on a halfword boundary. The operation is suppressed.

Priority No.	Condition	Flag Bit	Explanation
24	Address Error (Cont'd)	2^{23}	Protection - An address error (protection) interrupt occurs when the storage key and the protection key of the result main store location do not match, and neither is zero. The operation is suppressed if the first main store location specified that the instruction is in a protected area. The operation is terminated with unpredictable results if the instruction is in progress when the protected area is addressed. (This interrupt can only occur if the Store Protect option is installed.) Notes: 1. If an address error type interrupt occurs during an input/output operation (after initiation), an address error interrupt does not occur. Instead, a channel interrupt occurs for the appropriate channel. 2. It is a program restriction that the mask bit in processor state P3 for this interrupt condition must always be zero when this interrupt occurs.
25	Data Error	2^{24}	This interrupt occurs when any of the following conditions occur: 1. The sign or digit codes of operands in decimal arithmetic, editing, or Convert To Binary instructions are incorrect. 2. Fields overlap incorrectly in decimal arithmetic. 3. A decimal multiplicand has too many high-order, significant digits. The operation is terminated (suppressed if the operation is a Convert To Binary instruction) upon detection of any of the above.

Priority No.	Condition	Flag Bit	Explanation
26	Exponent Overflow	2^{25}	The exponent overflow interrupt occurs when the result exponent of floating-point addition, subtraction, multiplication, or division is greater than 127. The operation is terminated.
27	Divide Error	2^{26}	The divide error interrupt occurs when any of the following occur: 1. A quotient exceeds the general register size in fixed-point division, including division by zero. The division is suppressed. 2. The result of a Convert To Binary instruction exceeds one word. The conversion is completed by ignoring information which is outside the general register size. 3. A quotient exceeds the specified data field size in decimal divide. The division is suppressed. 4. Floating-point division is attempted with a divisor whose mantissa is zero. The operation is suppressed.
28	Significance Error	2^{27}	This interrupt occurs when the result mantissa of a floating-point add or subtract instruction is zero. If the interrupt is permitted (by the program mask and the interrupt mask) the operation is completed, the exponent is unaltered, and the interrupt is taken. If the interrupt is inhibited by the program mask, the interrupt condition is cancelled and the operation is completed by setting the result to true zero (zero sign, zero exponent and zero mantissa). If the interrupt is permitted by the

Priority No.	Condition	Flag Bit	Explanation
28	Significance Error (Cont'd)	2^{27}	program mask but inhibited by the interrupt mask, the interrupt remains pending and the operation is completed by setting the result to true zero (zero sign, zero exponent and zero mantissa).
29	Exponent Underflow	2^{28}	This interrupt occurs when the result exponent of a floating-point addition, subtraction, multiplication, or division is less than zero. The operation is completed by making the result true zero (zero sign, zero exponent, and zero mantissa). If the interrupt is inhibited by the program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.
30	Decimal Overflow	2^{29}	This interrupt occurs when the result field is too small to contain the result of a decimal operation. The operation is completed by ignoring the overflow data. If the interrupt is inhibited by the program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.
31	Fixed-Point Overflow	2^{30}	This interrupt occurs when a high-order carry occurs or high-order significant bits are lost in fixed-point addition, subtraction, shifting, or sign control operations. The operation is completed by ignoring the overflow data. If the interrupt is inhibited by the

Priority No.	Condition	Flag Bit	Explanation
31	Fixed-Point Overflow (Cont'd)	2^{30}	program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.
32	Test Mode	2^{31}	This interrupt provides program control over the processor during program testing. The program test interrupt flag is set by the Program Control instruction. When the interrupt flag bit and the related interrupt mask bit in the state to be initiated are both set, an interrupt occurs after the first instruction that is executed in the initiated processor state.

9.5 INTERRUPT MECHANIZATION

There are two ways of causing a change of processor state. They are:

1. Automatic Interrupt: effected when any interrupt condition described in table 7 occurs, and is permitted.
2. Program Controlled Interrupt: effected when a Program Control instruction is executed.

Whenever the processor state is changed, either by automatic interrupt or by the execution of a Program Control instruction, some machine conditions must be stored in the P counter and the Interrupt Status register of the terminated state for possible use when the state is initiated again. In addition, certain machine conditions associated with the state being initiated must be extracted from the P counter and the Interrupt Status register of the new state.

All the storing and extracting required when processor states are changed is accomplished by hardware.

9.5.1 Automatic Interrupt

When an automatic interrupt condition occurs, the following events occur: (See figure 1.)

Block 1 A check is made to see if the interrupt condition is one of the following four:

Significance Error

Exponent Underflow

Decimal Overflow

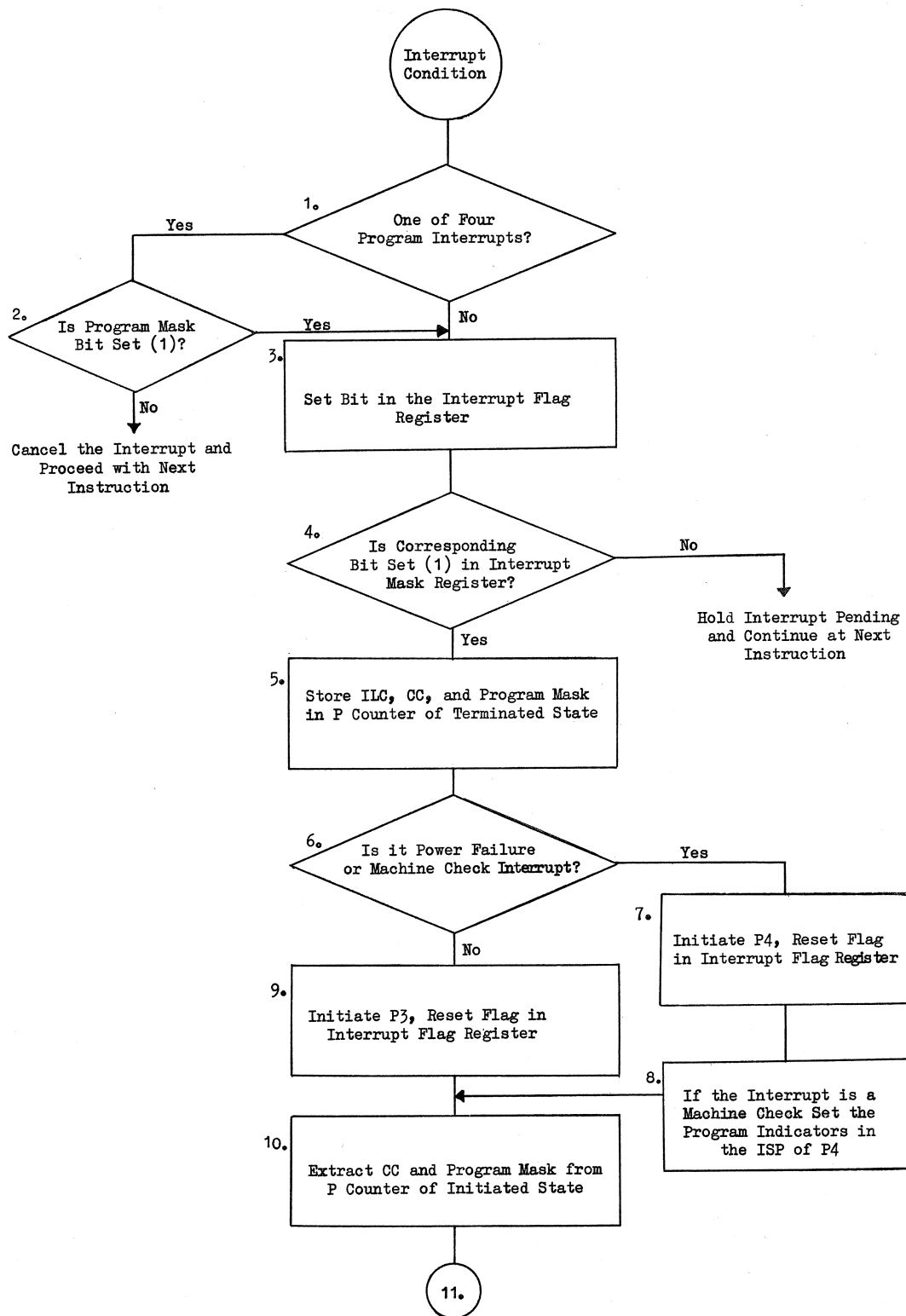
Fixed-Point Overflow

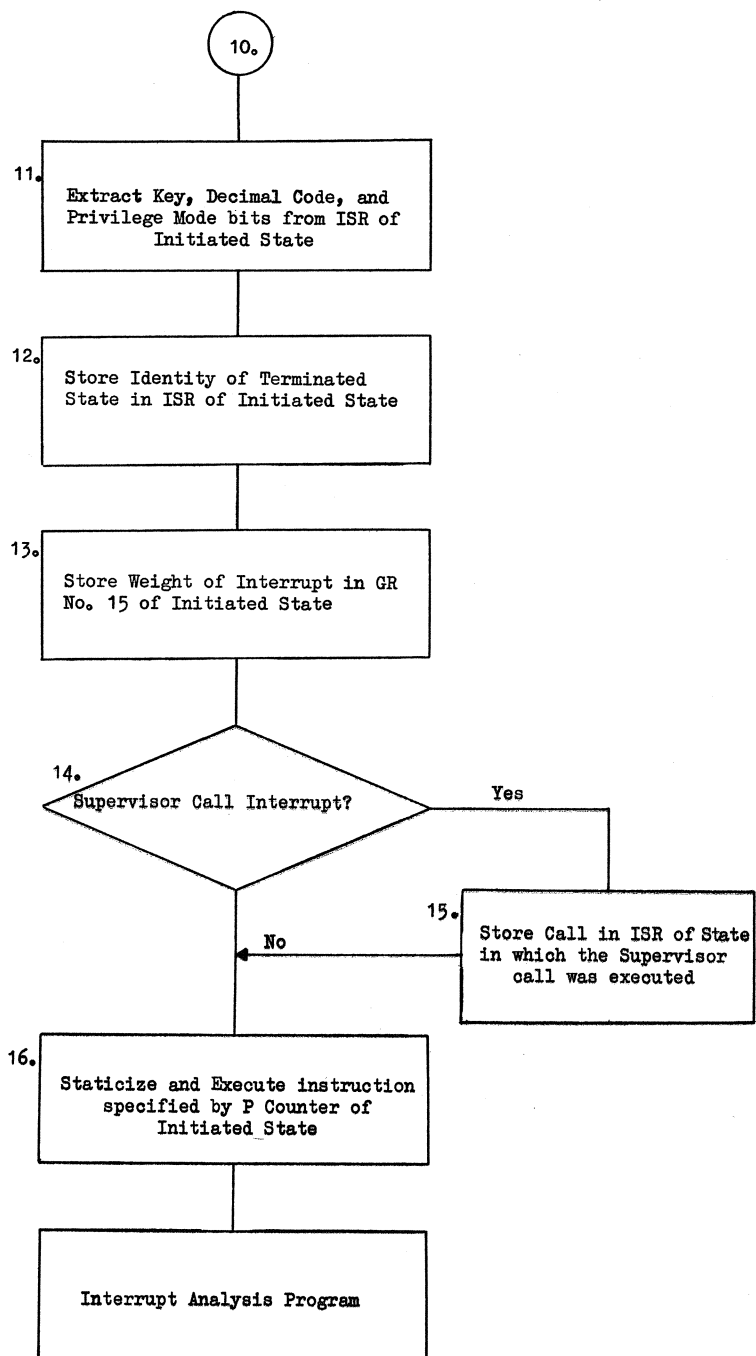
Block 2 If the interrupt condition is one of the above, the program mask (machine register) for the current program state is checked to see if the interrupt is permitted. If the program mask indicates that the interrupt is inhibited (mask = 0), the interrupt condition is cancelled and the next instruction in the current processor state is executed.

Block 3 If the interrupt condition is not one of the four program interrupts, or is one of the four program interrupts but the program mask indicates that the interrupt is to be permitted (mask = 1), the specific bit associated with the interrupt condition is set in the Interrupt Flag register.

Block 4 The bit in the Interrupt Flag register is compared with the corresponding bit in the Interrupt Mask register for the current state. If the bit in the Interrupt Mask register is reset (0), the interrupt condition remains pending and the next instruction in the current processor state is executed. The interrupt remains pending until the mask is changed to a permit status and the interrupt is serviced.

Figure 1 Functional Logic of Automatic Interrupt





Block 5 If the bit in the Interrupt Mask register is set, the interrupt is taken and information (ILC, CC, program mask) is stored in the P counter of the state being terminated.

Blocks 6 and 7 If the interrupt condition is a power failure or a machine check, the Machine Condition State P4 is initiated. The flag in the Interrupt Flag register is reset.

Block 8 If the interrupt is a Machine Check, the Program Indicators are stored in the Interrupt Status register of P4.

Block 9 If the interrupt condition is not a power failure or machine check, the Interrupt Control State P3 is initiated. The flag in the Interrupt Flag register is reset

Block 10 The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate hardware registers.

Block 11 The store protection key, the decimal code and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate registers.

Block 12 The state being terminated is identified to the state being initiated by setting an interrupted state identifier code in the Interrupt Status register of the initiated state.

Block 13 The weight of the condition causing the interrupt is stored in general register number 15 of the initiated state (P3 or P4).

Block 14 and 15 If the interrupt condition is a Supervisor Call, the R1 and R2 fields of the Supervisor Call instruction are stored in the rightmost eight-bits of the Interrupt Status register of the state in which the instruction is executed.

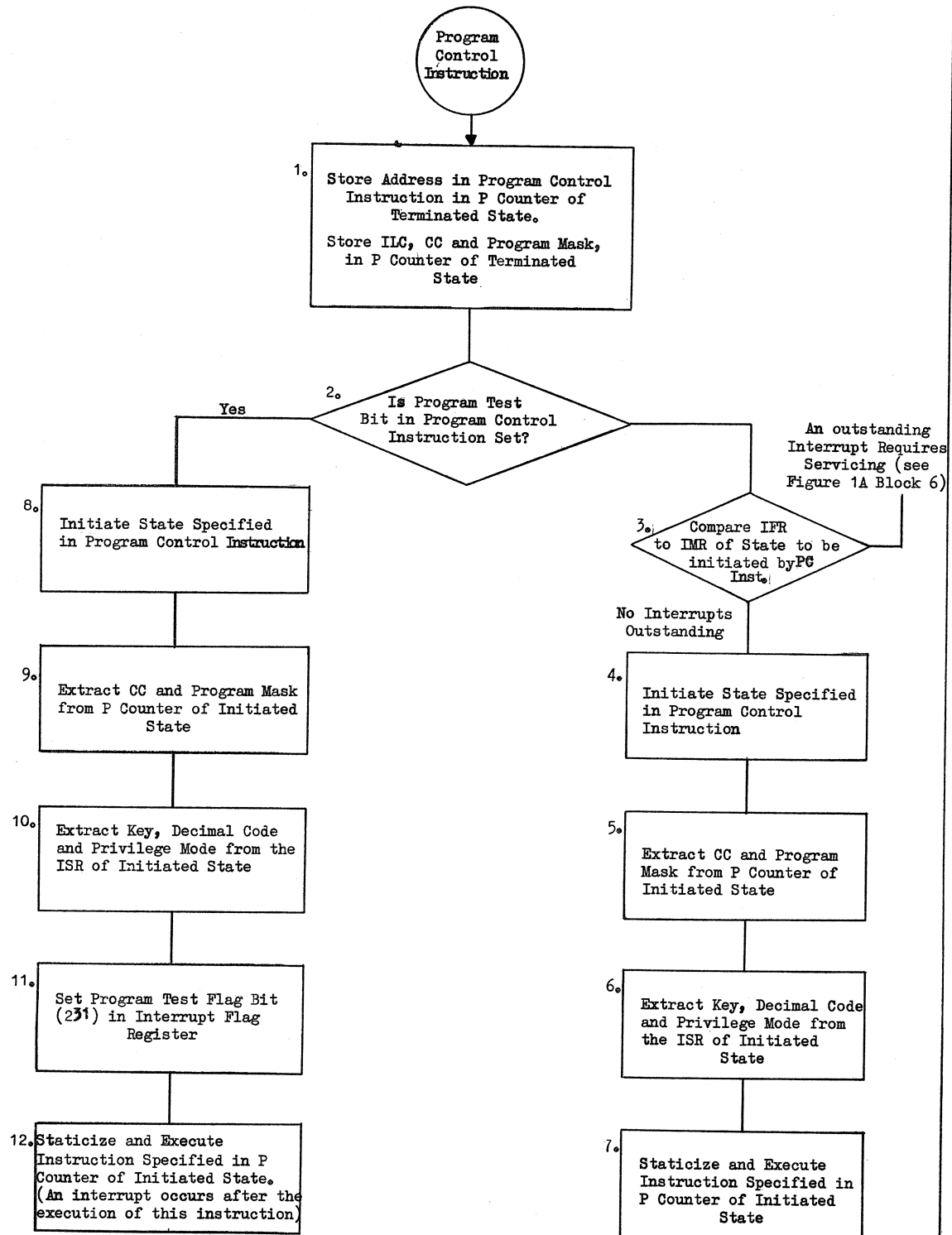
Block 16 The instruction at the address specified in the P counter of the initiated state is staticized and executed.

9.5.2 Program Controlled Interrupt

The Program Control instruction transfers the program from one processor state to another. This instruction is a privileged operation and can be executed only if the state in which the processor is operating is in the privileged mode (bit position 15 of the Interrupt Status register = 0). When a Program Control instruction is executed, the following events occur. (See Figure 2.)

Block 1 The address (B1,D1) specified in the Program Control instruction is stored in the P counter of the terminated state. The length of the last instruction executed in the terminated state, the condition code setting, and the program mask is stored in the P counter of the terminated state.

FIGURE 2 FUNCTIONAL LOGIC OF PROGRAM CONTROL INSTRUCTION



Block 2 A check is made to see if the program test bit in the Program Control instruction is set.

Block 3 If the program test bit is not set, the Interrupt Mask register for the state to be initiated by the Program Control instruction is compared to the Interrupt Flag register. If an interrupt condition has occurred, the events described under automatic interrupt take place (see figure 1A, block 3).

Note: If an interrupt is outstanding in the state to be initiated by the Program Control instruction, the number of the initiated state specified by the Program Control instruction is stored in the interrupt status identifier field of the Interrupt Status register of the initiated state (P3 or P4).

Block 4 If an interrupt condition is not outstanding in the state to be initiated by the Program Control, instruction control is transferred to the state specified by the Program Control instruction (directly or indirectly - See Program Control instruction).

Block 5 The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate machine registers.

Block 6 The store protection key, the decimal code and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate machine registers.

Block 7 The instruction at the address specified in the P counter of the initiated state is staticized and executed.

Block 8 If the program test bit is set, control is transferred to the state specified by the Program Control instruction (directly or indirectly - see Program Control Instruction).

Block 9 The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate registers.

Block 10 The store protection key, the decimal code, and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate registers.

Block 11 The program test flag bit (2^31) in the Interrupt Flag register is set.

Block 12 The instruction at the address specified in the P counter of the initiated state is staticized and executed.

Notes

1. When a Program Control instruction has the program test bit set, the first instruction of the initiated state is always executed before any interrupt is taken.

2. If the initiated state permits the program test interrupt (via the Interrupt Mask register), a program test interrupt occurs after the first instruction in the initiated state is executed.
3. An interrupt condition can occur while executing the first instruction of the initiated state. If it does, and is permitted, it is serviced before the program test interrupt.

9.5.3 General Notes on Program Interrupt

1. The decimal mode in the 4-50 is either ISO or EBCDIC as specified by bit 12 in the Interrupt Status register. When an automatic interrupt occurs or a Program Control instruction is executed, the decimal mode is not stored in the Interrupt Status register of the terminated state. The mode of the state being initiated is determined by the mode bit in its own Interrupt Status register.

Consequently, to change mode, the mode bit of the Interrupt Status register associated with the appropriate state must be altered by the program, and that state must be initiated either by an interrupt condition or a Program Control instruction. This is the method available to the program for changing the mode.

2. The interrupt flags are scanned to determine whether or not an interrupt shall occur if the Interrupt Mask register associated with the current state or the Interrupt Flag register are written into by the program.

3. Changing the protection key, decimal mode, or privileged mode fields in the Interrupt Status register does not change the protection key, machine mode, or privileged mode bits of the associated processor state. To change the status of the processor, the state concerned must be initiated by an interrupt condition or a Program Control instruction.

A1. List of instructions

This Appendix contains a list of the instructions available on System 4-50, in alphabetical order. The symbols used in the 'Instruction Type' column are as follows:-

RR	}	Machine instruction formats.
RS		
RX		
SI		
SS		
TRANS		Translator instruction.
EBC		Extended branch on condition instruction.
MAC		Instruction which is a member of the Macro language.

The symbols used in the operand format column are:-

- D - Arithmetic expression, specifying the address of the leftmost byte of the operand, relative to the base address in the register specified by B. The expression has a value in the range 0 to 4095.
- B - Arithmetic expression, specifying the register that will provide the base address of the operand. The expression has a value in the range 0 to 15.
- L - Arithmetic expression, specifying the length of the operand in bytes. If a length is allowed for both operands in an instruction, then the value of the expression lies in the range 1 to 16; if a length is permitted for the first operand only, then it is in the range 1 to 256.
- X - Arithmetic expression, specifying the register whose contents are to be used for address modification in an RX instruction. The expression has a value in the range 0 to 15.
- R - Arithmetic expression, self defining term or a symbol specifying a register, in the range 0 to 15.
- I - Arithmetic expression or self defining term specifying an immediate operand.
- M - Arithmetic expression or self defining term specifying a mask.
- S - An absolute or relocatable expression.
- LRE - Logical or relational expression.
- SS - Sequence symbol.

DXB Format

The allowable forms for an operand which has DXB format are:-

D(X,B)
S(X)
S

DLB Format

The allowable forms for an operand which has DLB format are:-

D(L,B)
D(,B)
S(L)
S

DB Format

The allowable forms for an operand which has DB format are:-

D(B)
S

MNEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
A	Add word	5A	RX	R, DXB	0 - Sum zero 1 - Sum less than zero 2 - Sum greater than zero 3 - Overflow	6-14
AD	Add normalized (long)	6A	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-132
ADR	Add normalized (long)	2A	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-132
AE	Add normalized (short)	7A	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-132
AER	Add normalized (short) floating point	3A	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-132
AGO	Unconditional branch forward	-	MAC	SS	-	*(4-24)
AGOB	Unconditional branch backwards	-	MAC	SS	-	*(4-26)
AH	Add halfword	4A	RX	R, DXB	0 - Sum zero 1 - Sum negative 2 - Sum positive 3 - Overflow	6-16
AIF	Conditional branch forward	-	MAC	LEE, SS	-	*(4-20)

* Page references in brackets, on this and the subsequent pages, refer to the page number in the Macro Reference Manual (Section 4.3.1)

MEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
ALFB	Conditional branch backward	-	MAC	LRE, SS	-	*(4-22)
AL	Add logical	5E	HX	R, DXB	0 - Sum zero, no carry 1 - Sum not zero, no carry 2 - Sum zero, with carry 3 - Sum not zero, with carry	6-17
ALOG	Assembler log	-	TRANS	-	Unchanged	7-34
ALR	Add logical	1E	PR	R, R	0 - Sum zero, no carry 1 - Sum not zero, no carry 2 - Sum zero, with carry 3 - Sum not zero, with carry	6-17
ANOP	No operation	-	MAC	-	-	*(4-28)
AOPTN	Translator option	-	TRANS	One or more of the specified options separated by commas	Unchanged	7-34
AP	Add decimal	FA	SS	-	0 - Sum zero 1 - Sum negative 2 - Sum positive 3 - Overflow	6-41
AR	Add word	1A	RR	R, R	0 - Sum zero 1 - Sum negative 2 - Sum positive 3 - Overflow	6-14
AU	Add unnormalized (short) floating point	7E	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-134
AUR	Add unnormalized (short) floating point	3E	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-134
AW	Add unnormalized (long) floating point	6E	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-134

INSTRUCTION CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
AWR	Add unnormalized (long) floating point	2E	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-134
B	Unconditional branch	47	EBC	DXB	Unchanged	7-2
BAL	Branch and link	45	RX	R, DXB	Unchanged	6-90
BAIR	Branch and link	05	RR	R, R	Unchanged	6-90
BC	Branch on condition	47	RX	M, DXB	Unchanged	6-88
BCR	Branch on condition	07	RR	M, R	Unchanged	6-88
BCI	Branch on count	46	RX	R, DXB	Unchanged	6-91
BCIR	Branch on count	06	RR	R, R	Unchanged	6-91
BE	Branch on equal	47	EBC	DXB	Unchanged	7-2
BH	Branch on high	47	EBC	DXB	Unchanged	7-2
BL	Branch on low	47	EBC	DXB	Unchanged	7-2
BM	Branch on minus	47	EBC	DXB	Unchanged	7-2
BNE	Branch on not equal	47	EBC	DXB	Unchanged	7-2
BNI	Branch on not high	47	EBC	DXB	Unchanged	7-2
BNI	Branch on not low	47	EBC	DXB	Unchanged	7-2
BO	Branch on plus	47	EBC	DXB	Unchanged	7-2
BP	Branch on plus	47	EBC	DXB	Unchanged	7-2
BR	Unconditional branch	07	EBC	R	Unchanged	7-2
BKH	Branch on index high	86	RS	R, R, DB	Unchanged	6-93

MEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
BXIE	Branch on index low or equal	87	RS	R, R, DB	Unchanged	6-94
BZ	Branch on zero	47	BEC	DXB	Unchanged	7-2
C	Compare word	59	RX	R, DXB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - not used	6-24
CCW	Channel command word	-	TRANS	Command word definition	-	7-38
CD	Compare long	59	RX	R, DXB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-140
CDR	Compare long	29	RR	R, R	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-140
CE	Compare short	79	RX	R, DXB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-140
CDR	Compare short	39	RR	R, R	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-140
CH	Compare halfword	49	RX	R, DXB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-23

INSTRONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONVATION CODE SETTING	PAGE REF.
CKC	Check channel	9F	SI	DB, I	0 - I/O channel available 1 - Interrupt pending in selector channel 2 - selector channel busy or multiplexor channel operating in burst mode 3 - I/O channel inoperable	6-114
CL	Compare logical	55	RX	R, DXB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-61
CLC	Compare logical	D5	SS	DLB, DB	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-61
CLI	Compare logical	95	SI	DB, I	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-61
CLR	Compare logical	15	RR	R, R	0 - Operands equal 1 - First operand low 2 - First operand high 3 - Not used	6-61
CROP	Conditional No-operation	-	TRANS	b, w where b=byte within word, w=4 or 8 for full or double word		7-40
COM	Define common control section	-	TRANS	-	Unchanged	7-42
CP	Compare decimal	F9	SS	DLB, DLB	0 - Fields algebraically equal 1 - First operand low 2 - First operand high 3 - Not used	6-47

INSTRUCTION CODE	INSTRUCTION CODE	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
CR	Compare word	19	RR	R, R	0 - Operands equal 1 - First operand low 2 - Second operand low 3 - Not used	6-24
CSECT	Identify control section	-	TRANS	-	-	7-23
CVB	Convert to binary	4P	RR	R, DIB	Unchanged	6-30
CVD	Convert to decimal	4E	RR	R, DIB	Unchanged	6-31
D	Divide	5D	RR	R, DIB	Unchanged	6-28
DC	Define constant	-	TRANS	Constant definition	-	7-8
DD	Divide (long) floating point	6D	RR	R, DIB	Unchanged	6-146
DDR	Divide (long) floating point	2D	RR	R, R	Unchanged	6-146
DE	Divide (short) floating point	7D	RR	R, DIB	Unchanged	6-146
DER	Divide (short) floating point	3D	RR	R, R	Unchanged	6-146
DIG	Diagnose	83	SI	DB, I	Unchanged	6-103
DP	Divide decimal	FD	SS	DIB, DIB	Unchanged	6-50
DR	Divide	1D	RR	R, R	Unchanged	6-28
DROP	Drop base register	-	TRANS	R, R, ----	-	7-4
DS	Define storage	-	TRANS	Storage definition	-	7-19
DSECT	Identify dummy section	-	TRANS	-	-	7-24

MEMEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	TYPE TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
ED	Edit	DE	SS	DLEB, DB	0 - Indicates zero source field whether or not significance is established 1 - Non-zero result field with significance established to indicate less than zero 2 - Non-zero result field with no significance established to indicate greater than zero 3 - Not used	6-77
EDMK	Edit and mark	DF	SS	DLEB, DB	0 - Indicate zero source field whether or not significance is established 1 - Non-zero result field with significance established to indicate less than zero 2 - Non-zero result field with no significance established to indicate greater than zero 3 - Not used	6-81
EJECT	Start new page	-	TRANS	-	-	7-35
END	End assembly	-	TRANS	relocatable expression or blank	-	7-22
ENTRY	Identifier entry point symbol	-	TRANS	relocatable symbol	-	7-26
EQU	Equate	-	TRANS	expression	-	7-6
EX	Execute	44	HX	R, DXB	May be set by instruction being modified and executed	6-95

HEXONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPER AND FORMAT	CONDITION CODE SETTING	PAGE REF.
EXTEN	Identify external symbol	-	TRANS	relocatable symbol	-	7-27
HDR	Halve (long) floating point	24	RR	R, R	Unchanged	6-142
HNV	Halt device	9E	SI	DB, I	0 - Channel or subchannel not busy 1 - Status stored in the scratch-pad 2 - Termination accepted 3 - Channel or subchannel inoperable	6-108
HER	Halve (short) floating point	34	RR	R, R	Unchanged	6-142
IC	Insert character	43	RX	R, DXB	Unchanged	6-70
IDL	Idle	80	SI	DB, I	Unchanged	6-102
ISSQ	Input sequence checking	-	TRANS	Two decimal values separated by commas, or blank	-	7-30
ISY	Insert storage key	09	RR	P, R	Unchanged	6-116
L	Load word	58	RX	R, DXB	Unchanged	6-7
LA	Load address	41	RX	R, DXB	Unchanged	6-72
LCBR	Load complement (long) floating point	23	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Not used	6-129

MEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
LCER	Load complement (short) floating point	33	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Not used	6-129
LCR	Load complement	13	RR	R, R	0 - Result zero 1 - Result negative 2 - Result positive 3 - Not used	6-10
LD	Load (long) floating point	68	RX	R, DMB	Unchanged	6-148
LDR	Load(long) floating point	28	RR	R, R	Unchanged	6-148
LE	Load (short) floating point	78	RX	R, DMB	Unchanged	6-148
LER	Load (short) floating point	38	RR	R, R	Unchanged	6-148
IH	Load halfword	48	RX	R, DMB	Unchanged	6-8
LM	Load multiple	98	RS	R, R, DB	Unchanged	6-13
LNDR	Load negative (long) floating point	21	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Not used 3 - Not used	6-131
LNDR	Load negative (short) floating point	31	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Not used 3 - Not used	6-131

MEMORIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPRAND FORMAT	CONDITION CODE SETTING	PAGE REF.
LNR	Load negative	11	RR	R, R	0 - Result zero 1 - Result negative 2 - Not used 3 - Not used	6-12
LPFR	Load positive (long) floating point	20	RR	R, R	0 - Mantissa zero 1 - Not used 2 - Mantissa positive 3 - Not used	6-130
LPFR	Load positive (short) floating point	30	RR	R, R	0 - Mantissa zero 1 - Not used 2 - Mantissa positive 3 - Not used	6-130
LPR	Load positive	10	RR	R, R	0 - Result zero 1 - Not used 2 - Result positive 3 - Overflow	6-11
LR	Load word	18	RR	R, R	Unchanged	6-7
LSP	Load scratch-pad	D8	SS	DLB, DB	Unchanged	6-98
LTR	Load and test (long) floating point	22	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Not used	6-128
LTR	Load and test (short) floating point	32	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Not used	6-128
LTRG	Begin literal pool	-	TRANS	-	-	7-29
LTR	Load and test	12	RR	R, R	0 - Result zero 1 - Result negative 2 - Result positive 3 - Not used	6-9
M	Multiply word	5C	RX	R, RDB	Unchanged	6-25

INSTRUCTION CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CON. ACTION CODE SETTING	PAGE REF.
MACRO	Macro header statement	-	MAC	-	-	*(2.2)
MCALL	Pre-store library macros	-	TRANS	Symbols separated by commas	Unchanged	7-44
MD	Multiply (long) floating point	6C	RX	R, DXB	Unchanged	6-144
MDR	Multiply (long) floating point	2C	RR	R, R	Unchanged	6-144
ME	Multiply (short) floating point	7C	RX	R, DXB	Unchanged	6-144
MEND	Macro trailer statement	-	MAC	-	-	*(2.13)
MER	Multiply (short) floating point	3C	RR	R, R	Unchanged	6-144
MEXIT	Macro definition exit	-	MAC	-	-	*(4-30)
MH	Multiply halfword	4C	RX	R, DXB	Unchanged	6-27
MNOTE	Error message request	-	MAC	Error code, error message enclosed in quotation marks	-	*(4-32)
MP	Multiply decimal	FC	SS	DLB, DLB	Unchanged	6-48
MPRTY	Store macros by priority	-	TRANS	4 digit combination of zeros and ones	Unchanged	7-45
MR	Multiply word	1C	RR	R, R	Unchanged	6-25
MTRAC	Macro trace	-	MAC	Not used	Unchanged	7-47
MVC	Move characters	D2	SS	DLB, DB	Unchanged	6-57

MEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
MVI	Move immediate	92	SI	DB, I	Unchanged	6-57
PTN	Move numerics	D1	SS	DLB, DB	Unchanged	6-59
MVO	Move with offset	F1	SS	DLB, DLB	Unchanged	6-55
MVZ	Move zones	D3	SS	DLB, DB	Unchanged	6-60
N	And	54	RX	R, DXB	0 - Result zero 1 - Result non-zero 2 - Not used 3 - Not used	6-63
NC	And	D4	SS	DLB, DB	0 - Result zero 1 - Result non-zero 2 - Not used 3 - Not used	6-63
N1	And	94	SI	DB, I	0 - Result zero 1 - Result non-zero 2 - Not used 3 - Not used	6-63
NOP	No operation	47	RX	DXB	Unchanged	7-2
NOPR	No operation	07	RR	R	Unchanged	7-2
NR	And	14	RR	R, R	0 - Result zero 1 - Result non-zero 2 - Not used 3 - Not used	6-63
NTFAC	Cancel macro trace	-	TRANS	-	Unchanged	7-48
O	Or	56	RX	R, DXB	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-65
OC	Or	D6	SS	DLB, DB	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-65

MEMORIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
OI	Or	96	SI	DB, I	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-65
OR	Or	16	RR	R, R	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-65
ORG	Set location counter	-	TRANS	Relocatable expression or blank	-	7-28
PACK	Pack	P2	SS	DLB, DLB	Unchanged	6-52
PC	Program control	82	SI	DB, I	CC of state being terminated is stored in P-counter CC of state being initiated used to set CC indicator	6-100
PRINT	Print optional data	-	TRANS	One to three operands	-	7-33
PUNCH	Punch a card	-	TRANS	A sequence of characters enclosed in single quotation marks	Unchanged	7-46
RDD	Read direct	85	SI	DB, I	Unchanged	6-121
S	Subtract word	5B	RX	R, DXB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-18
SD	Subtract normalized (long) floating point	6B	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-136
SDR	Subtract normalized (long) floating point	2B	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-136

MNEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	COMMAND FORMAT	CONDITION CODE SETTING	PAGE REF.
SDV	Start device	9C	SI	DB, I	0 - I/O operation started and channel proceeding 1 - Status bits stored in scratch-pad 2 - Channel or sub-channel busy 3 - Channel or sub-channel inoperable	6-104
SE	Subtract normalized (short) floating point	73	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-136
SER	Subtract normalized (short) floating point	3B	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-136
SET	Set arithmetic	-	MAC	An arithmetic expression	-	*(4-3)
SETB	Set binary	-	MAC	0 or 1 or a logical or relational expression enclosed within parentheses	-	*(4-3)
SETC	Set character	-	MAC	Up to eight characters enclosed by a pair of single quote marks	-	*(4-6)
SH	Subtract halfword	4B	RX	R, DXB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-20
SL	Subtract logical	5F	RX	R, DXB	0 - Not used 1 - Result non-zero, no carry 2 - Result zero with carry 3 - Result non-zero with carry	6-21

HEXONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
SIA	Shift left single	8B	RS	R, DB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-35
SIDA	Shift left double	8F	RS	R, DB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-37
SIDL	Shift left double logical	8D	RS	R, DB	Unchanged	6-85
SIL	Shift left single logical	99	RS	R, DB	Unchanged	6-83
SLR	Subtract logical	1F	RR	R, R	0 - Not used 1 - Result non-zero, no carry 2 - Result zero with carry 3 - Result non-zero with carry	6-21
SP	Subtract decimal	FB	SS	DIB, DLB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-43
SPACE	Space listing	-	TRANS	A decimal value or blank	-	7-36
SPM	Set program mask	04	RR	R	CC set according to bits 2 and 3 of the register specified	6-127
SR	Subtract word	1B	RR	R, R	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-18

MEMORIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
SRA	Shift right single	8A	RS	R, DB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Not used	6-36
SRDA	Shift right double	8E	RS	R, DB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Not used	6-36
SRDL	Shift right double logical	8C	RS	R, DB	Unchanged	6-86
SRL	Shift right single logical	88	RS	R, DB	Unchanged	6-84
SSK	Set storage key	08	RR	R, R	Unchanged	6-118
SSP	Store scratch pad	D0	SS	DIB, DB	Unchanged	6-99
ST	Store word	50	RX	R, DXB	Unchanged	6-32
START	Start assembly	-	TRANS	Self-defining term or blank	-	7-21
STC	Store character	42	RX	R, DXB	Unchanged	6-71
STD	Store (long) floating point	60	RX	R, DXB	Unchanged	6-143
STE	Store (short) floating point	70	RX	R, DXB	Unchanged	6-143
STH	Store halfword	40	RX	R, DXB	Unchanged	6-33
STM	Store multiple	90	RS	R, R, DB	Unchanged	6-34
SU	Subtract unnormalized (short) floating point	7F	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-138

MNEMONIC CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
SUR	Subtract unnormalized (short) floating point	3F	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-138
SVC	Supervisor call	0A	RR	R, R	Unchanged	6-126
SW	Subtract unnormalized (long) floating point	6F	RX	R, DXB	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-138
SWR	Subtract unnormalized (long) floating point	2F	RR	R, R	0 - Mantissa zero 1 - Mantissa negative 2 - Mantissa positive 3 - Exponent overflow	6-138
TDV	Test device	9D	SI	DB, I	0 - Available 1 - Status bits stored in scratch-pad 2 - Channel or sub-channel busy 3 - Channel or sub-channel inoperable	6-111
TITLE	Identify assembly output	-	TRANS	Sequence of characters enclosed in single quote marks	-	7-32
TM	Test under mask	91	SI	DB, I	0 - Selected bits all zero, or mask all zero 1 - Selected bits mixed zero and one 2 - Not used 3 - Selected bits all one	6-69
TR	Translate	DC	SS	DXB, DB	Unchanged	6-73

MICRO- CODE	INSTRUCTION NAME	MACHINE CODE	INSTRUCTION TYPE	OPERAND FORMAT	CONDITION CODE SETTING	PAGE REF.
TFT	Translate and test	DD	SS	DLB, DB	0 - All accessed function bytes all zeros 1 - Non zero function bytes encountered 2 - Last function byte non-zero 3 - Not used	6-75
UNPK	Unpack	F5	SS	DLB, DLB	Unchanged	6-54
USING	Use base register	-	TRANS	S, R, R, R,	-	7-3
WRD	Write direct	84	SI	DB, I	Unchanged	6-119
X	Exclusive or	57	RX	R, DXB	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-67
XC	Exclusive or	D7	SS	DLB, DB	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-67
XI	Exclusive or	97	SI	DB, I	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-67
XR	Exclusive or	17	RR	R, R	0 - Result zero 1 - Result non-zero 2 - First operand high 3 - Not used	6-67
ZAP	Zero and add	F8	SS	DLB, DLB	0 - Result zero 1 - Result negative 2 - Result positive 3 - Overflow	6-45

A2. Character codes

This Appendix will be supplied later.

A3. Reserved store locations

There are only two reserved locations in the System 4-50. They are bytes 72-75 which form the Channel Address Word, and bytes 80-83, which constitute the Timer.

However, the lower end of the store is used by the Supervisor. The size of the Supervisor varies according to the operating system, but may be up to 20,000 (for the 4-50 Disc Operating System).

The layout of the Scratch Pad Store and the assignment of registers is shown in the diagram overleaf.

Scratch Pad Layout and Register Assignment

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	General Purpose Register No. 0 P2	General Purpose Register No. 1 P2	General Purpose Register No. 2 P2	General Purpose Register No. 3 P2	General Purpose Register No. 4 P2	General Purpose Register No. 5 P2	General Purpose Register No. 6 P2	General Purpose Register No. 7 P2	General Purpose Register No. 8 P2	General Purpose Register No. 9 P2	General Purpose Register No. 10 P4	General Purpose Register No. 11 P4	Interrupt Mask Register P4	Interrupt Status Register P4	Program Counter P4	General Purpose Register No. 15 (Weight) P4
1	Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Status Register	Processor	Processor	Utility	General Purpose Register No. 8 P4	General Purpose Register No. 9 P4	Channel Address Register	Channel Command Register II	Channel Command Register I	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I
2	Interrupt Mask Register P1	Interrupt Status Register P1	Program Counter P1	Interrupt Flag Register	Interrupt Mask Register P2	Interrupt Status Register P2	Program Counter P2	General Purpose Register No. 7 P3	Interrupt Mask Register P3	Interrupt Status Register P3	Program Counter P3	General Purpose Register No. 11 P3	General Purpose Register No. 12 P3	General Purpose Register No. 13 P3	General Purpose Register No. 14 P3	General Purpose Register No. 15 (Weight) P3
3	Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Selector No. 2	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I	Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I	Pre-Fetch Channel Command Register I
4	General Purpose Register No. 0 P2	General Purpose Register No. 1 P2	General Purpose Register No. 2 P2	General Purpose Register No. 3 P2	General Purpose Register No. 4 P2	General Purpose Register No. 5 P2	General Purpose Register No. 6 P2	General Purpose Register No. 7 P2	General Purpose Register No. 8 P2	General Purpose Register No. 9 P2	General Purpose Register No. 10 P2	General Purpose Register No. 11 P2	General Purpose Register No. 12 P2	General Purpose Register No. 13 P2	General Purpose Register No. 14 P2	General Purpose Register No. 15 P2
5	Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Selector No. 4	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I	Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I	Pre-Fetch Channel Command Register I
6	General Purpose Register No. 0 P1	General Purpose Register No. 1 P1	General Purpose Register No. 2 P1	General Purpose Register No. 3 P1	General Purpose Register No. 4 P1	General Purpose Register No. 5 P1	General Purpose Register No. 6 P1	General Purpose Register No. 7 P1	General Purpose Register No. 8 P1	General Purpose Register No. 9 P1	General Purpose Register No. 10 P1	General Purpose Register No. 11 P1	General Purpose Register No. 12 P1	General Purpose Register No. 13 P1	General Purpose Register No. 14 P1	General Purpose Register No. 15 P1
7	Floating-Point Register No. 0	Floating-Point Register No. 0	Floating-Point Register No. 2	Floating-Point Register No. 4	Floating-Point Register No. 6				Processor Utility	Processor Utility	Channel Address Register	Channel Command Register II	Channel Command Register I	Pre-Fetch Channel Command Register II	Pre-Fetch Channel Command Register I	Pre-Fetch Channel Command Register I

*Word Address is in Hexadecimal; e.g., 2A = Program Counter P3.

A4. Sample coding sheet

This Appendix contains a sample of the coding sheet used for Usercode programs.

SYSTEM 4 USERCODE CODING SHEET

SYSTEM 4 USERCODE CODING SHEET

Program Identifier

Programmer's Name.....

Sheet No. of Sheets

Date.....

NAME				OPERATION			OPERANDS/COMMENTS				COZT	IDENTIFICATION No.	
1				8	10	14	16	39	40	71		73	80

A5. Hexidecimal conversion and useful numbers

This Appendix contains a Hexadecimal to Decimal Conversion table and tables of some useful numbers to high degrees of precision. None of the hexadecimal values have been rounded.

The tables provided are:-

- A5.1 Hexadecimal - Decimal Conversion
- A5.2 Logarithms to Base 10
- A5.3 Logarithms to Base e
- A5.4 Logarithms to Base 2
- A5.5 Useful Functions

A5.1 HEXADECIMAL-DECIMAL NUMBER CONVERSION

This table contains the necessary reference information for the conversion of decimal numbers to hexadecimal numbers and the conversion of binary numbers to decimal or hexadecimal.

Example 1

$$(00111010)_2 = (3A)_{16} = (58)_{10}$$

Example 2

$$(F6)_{16} = (11111100)_2 = (252)_{10}$$

In the conversion of a hexadecimal number to its decimal value the marks (0-F) represent a multiplier and their positions (reading right to left) within the hexadecimal number represent the exponent of the base. Each mark is multiplied by the base raised to the appropriate power and the summation of their product is the decimal value of the number.

Example 3

$$(36F)_{16} = 3 (16^2) + 6 (16^1) + 15 (16^0)$$

F

$$(36F)_{16} = 3 (256) + 6 (16) + 15 (1) = (879)_{10}$$

To convert hexadecimal to binary substitute the binary equivalent of the hexadecimal mark into its appropriate position as follows:

$$(3 \quad 6 \quad F)_{16} = (0011 \quad 0110 \quad 1111)_2$$

The table provides for direct conversion of decimal and hexadecimal numbers in these ranges:

Hexadecimal	Decimal
000 to FFF	0000 to 4095

For numbers outside the range of the table, add the following values to the table figures:

Hexadecimal	Decimal	Hexadecimal	Decimal
1000	4096	8000	32768
2000	8192	9000	36864
3000	12288	A000	40960
4000	16384	B000	45056
5000	20484	C000	49152
6000	24576	D000	53248
7000	28672	E000	57344
		F000	61440

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
FO	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

A5.2 LOGARITHMS TO BASE 10

DECIMAL VALUES

Log 2	0.30102 99956 63981 19521 37388 94724
Log e	0.43429 44819 03251 82765 11289 18917
Log π	0.49714 98726 94133 85435 12682 88291

HEXADECIMAL VALUES

0.4D1 04D 427 DE7 FBC C47 C4A CD6
0.6F2 DEC 549 B94 38C A9A ADD 558
0.7F4 536 CC6 E8E ECO 32F EDA EE3

A5.3 LOGARITHMS TO BASE e

DECIMAL VALUES

Log 2	0.69314 71805 59945 30941 72321 21458
Log π	1.14472 98858 49400 17414 34273 51353
Log 10	2.30258 50929 94045 68401 79914 54684

HEXADECIMAL VALUES

0.B17 217 F7D 1CF 79A BC9 E3B 398
1.250 D04 8E7 A1B DOB D5F 956 C6B
2.4D7 637 76A AA2 B05 BA9 5B5 8AE

A5.4 LOGARITHMS TO BASE 2

DECIMAL VALUES

Log e	1.44269 50408 88963 40735 99246 81002
Log π	1.65149 61294 72318 79804 32792 95108
Log 10	3.32192 80948 87362 34787 03194 29489

HEXADECIMAL VALUES

1.715 476 52B 82F E17 77D OFF DA1
1.A6C 873 498 DDF 75B ODB 2E4 F08
3.526 9E1 2F3 4E6 2BF 924 AFD BFD

A5.5 USEFUL FUNCTIONS

DECIMAL VALUES

$\sqrt{2}$	1.41421 35623 73095 04880 16887 24210
$\sqrt{3}$	1.73205 08075 68877 29352 74463 41506
π	3.14159 26535 89793 23846 26433 83280
$1/\pi$	0.31830 98861 83790 67153 77675 26745
$\sqrt{\pi}$	1.77245 38509 05516 02729 81674 83341
$\sqrt{2\pi}$	2.50662 82746 31000 50241 57652 84811
$\sqrt{\frac{\pi}{2}}$	1.25331 41373 15500 25120 78826 42406
$\sqrt{\frac{1}{\pi}}$	0.56418 95835 47756 28694 80794 51561
$\frac{1}{\sqrt{2}}$	0.39894 22804 01432 67793 99460 59934
$\sqrt{\frac{2}{\pi}}$	0.79788 45608 02865 35587 98921 19869
e	2.71828 18284 59045 23536 02874 71353
$1/e$	0.36787 94411 71442 32159 55237 70161
γ	0.57721 56649 01532 86060 65120 90082

HEXADECIMAL VALUES

1.6A0 9E6 67F 3BC C90 8B2 FB1 367
1.BB6 7AE 858 4CA A73 B25 742 D70
3.243 F6A 888 5A3 08D 313 198 A2E
0.517 CC1 B72 722 OA9 4FE 13A BE9
1.C5B F89 1B4 EF6 AA7 9C3 B05 20D
2.81B 263 FEC 4EO B2C AF9 483 F5D
1.40D 931 FF6 270 596 57C A41 FAE
0.906 EBA 821 4DB 688 D71 D48 A7F
0.662 114 CF5 OD9 423 43F 2CF 140
0.CC4 229 9EA 1B2 846 87E 59E 280
2.B7E 151 628 AED 2A6 ABF 715 881
0.5E2 D58 D8B 3BC DF1 ABA DEC 783
0.93C 467 E37 DBO C7A 4D1 BE3 F81

γ = Eulers constant.

A6. Index

Add Decimal Instruction (AP)	6.3.1
Add Halfword Instruction (AH)	6.2.9
Add Logical Instructions (ALR,AL)	6.2.10
Add Normalized Instructions (AER,AE,ADR,AD)	6.8.5
Add Unnormalized Instructions (AUR,AU,AWR,AW)	6.8.6
Add Word Instruction (AR,A)	6.2.8
Address Constants	7.4.1
AND Instruction (NR,N,NI,NC)	6.4.5
Assembler Log Instruction (ALOG)	7.6.6
Assembly Status Register	8.3
Attributes	3.3.2
Begin Literal Pool Instruction (LTORG)	7.5.8
Binary Constants	7.4.1
Binary Self Defining Term	3.3.7
Branch and Link Instructions (BALR,BAL)	6.5.2
Branch on Condition Instructions (BCR,BC)	6.5.1
Branch on Count Instructions (BCTR,BCT)	6.5.3
Branch on Index High Instruction (BXH)	6.5.4
Branch on Index Low or Equal Instruction (BXLE)	6.5.5
Branching Instructions	6.5
Channel Address Register	8.3
Channel Address Word	8.2.5
Channel Command Register	8.3
Channel Command Word	8.2.6
Channel Status Byte	8.5.2
Character Constants	7.4.1
Character Self Defining Term	3.3.7
Check Channel Instruction (CKC)	6.6.10;8.4.4
Common Control Section (COM)	7.9
Comments Statements	3.1.3
Compare Decimal Instruction (CP)	6.3.4
Compare Halfword Instruction (CH)	6.2.14
Compare Instructions (CER,CE,CDR,CD)	6.8.9
Compare Logical Instructions (CLR,CL,CLI,CLC)	6.4.4
Compare Word Instructions (CR,C)	6.2.15
Condition Code	8.5.1
Conditional No Operation Instruction (CNOP)	7.8
Constants	7.4.1
Control Section Attribute	3.3.2
Control Sections	3.7
Convert to Binary Instruction (CVB)	6.2.19
Convert to Decimal Instruction (CVD)	6.2.10
Data Formats	4
- Fixed Point	4.4
- Floating Point	4.5
- Logical	4.6
- Packed Decimal	4.2
- Summary	4.7
- Zoned Decimal	4.3

Decimal Arithmetic Instructions	6.3
Decimal Constants	7.4.1
Decimal Self Defining Term	3.3.7
Define Channel Command Word Instruction (CCW)	7.7
Define Common Control Section (COM)	7.9
Define Constant Instruction (DC)	7.4.1
Define Storage Instruction (DS)	7.4.2
Diagnose Instruction (DIG)	6.6.6
Divide Decimal Instruction (DP)	6.3.6
Divide Instructions (DER, DE, DDR, DD)	6.8.13
Divide Instructions (DR, D)	6.2.18
Drop Base Register Instruction (DROP)	7.2.2
Duplicate Literals	3.3.8
Edit and Mark Instruction (EDMK)	6.4.15
Edit Instruction (ED)	6.4.14
End Translation Instruction (END)	7.5.2
Equate Symbol Instruction (EQU)	7.3.1
Exclusive OR Instructions (XR, X, XI, XC)	6.4.7
Execute Instruction (EX)	6.5.6
Expressions	3.4
Extended Mnemonic Codes	7.1
Fixed Point Constants	7.4.1
Fixed Point Instructions	6.2
Floating Point Constants	7.4.1
Floating Point Instructions	6.8
Floating Point Registers	2.4
General Registers	2.4
Halt Device Instruction (HDV)	6.6.8; 8.4.2
Halve Instructions (HER, HDR)	6.8.10
Hexadecimal Constants	7.4.1
Hexadecimal Self Defining Term	3.3.7
Identify Control Section Instruction (CSECT)	7.5.3
Identify Dummy Section Instruction (DSECT)	7.5.4
Identify Entry Point Symbol Instruction (ENTRY)	7.5.5
Identify External Symbol Instruction (EXTRN)	7.5.6
Identify Translator Output Instruction (TITLE)	7.6.1
Idle Instruction (IDL)	6.6.5
Inner Expression	3.3.9
Input/Output Channel Registers	2.4; 8.3
Input/Output Channels	8.2
Input/Output Channels - Channel Servicing	8.2.4; 8.6
Input/Output Channels - Multiplexor	8.2.2
Input/Output Channels - Selector	8.2.1
Input/Output Control	2.6
Input/Output Operation	8
Input Sequence Check Instruction (ISEQ)	7.5.9
Insert Character Instruction (IC)	6.4.9
Insert Storage Key Instruction (ISK)	6.6.11

Instructions - Alignment of	3.5.1
- Branching	6.5
- Decimal Arithmetic	6.3
- Fixed Point	6.2
- Floating Point	6.8
- Formats of	5
- List of	6.1
- Logical	6.4
- Privileged	6.6
- Processor State Control	6.7
Interrupt Mask Registers	2.4
Interrupt Status Registers	2.4
Interrupts	9
- Automatic	9.5.1
- Conditions	9.4
Length Attribute	3.3.2
Literals	3.3.8
Literals-Duplicate	3.3.8
Load Address Instruction (LA)	6.4.11
Load and Test Instruction (LTR)	6.2.3
Load and Test Instructions (LTER, LTDR)	6.8.1
Load Complement Instruction (LCR)	6.2.4
Load Complement Instructions (LCER, LCDR)	6.8.2
Load Halfword Instruction (LH)	6.2.2
Load Instructions (LER, LE, LDR, LD)	6.8.14
Load Multiple Instruction (LM)	6.2.7
Load Negative Instruction (LNR)	6.2.6
Load Negative Instructions (LNER, LNDR)	6.8.4
Load Positive Instruction (LPR)	6.2.5
Load Positive Instructions (LPER, LPDR)	6.8.3
Load Scratch Pad Instruction (LSP)	6.6.2
Load Word Instructions (LR, L)	6.2.1
Location Counter	3.3.3
Logical Instructions	6.4
Machine Instruction Statements	3.1.1; 3.5
Machine Instructions - Operand Subfields	3.5.2
Macro Definition Language Statements	3.1.5
Macro Instruction Statements	3.1.4
Macro Storage within Translator	7.10
Main Store	2.2
Main Store - Non Addressable	2.3
Move Instructions (MVI, MVC)	6.4.1
Move Numerics Instruction (MVN)	6.4.2
Move with Offset Instruction (MVO)	6.3.9
Move Zones Instruction (MVZ)	6.4.3
Multiplexor Channel	8.2.2
Multiply Decimal Instruction (MP)	6.3.5
Multiply Halfword Instruction (MH)	6.2.17
Multiply Instructions (MER, ME, MDR, MD)	6.8.12
Multiply Word Instructions (MR, M)	6.2.16
Non Addressable Main Store	2.3

OR Instructions (OR,O,OI,OC)	6.4.6
Pack Instruction (PACK)	6.3.7
Pre-Store Library Macros (MCALL)	7.10.1
Print Optional Data Instruction (PRINT)	7.6.2
Privileged Instructions	6.6
Processor State Control Instructions	6.7
Processor State Registers	9.3
Processor States	9.2
Processor Utility Registers	2.4
Program Control and Arithmetic Unit	2.5
Program Control Instruction (PC)	6.6.4
Program Counter	2.4; 9.3.1
Program Interrupts	9
Program Linking	3.7
Program Sectioning	3.7
Program Structure	3.1
Read Direct Instruction (RDD)	6.6.14
Read Only Store	2.7
Registers - Floating Point	2.4; 9.3.3
- General	2.4; 9.3.2
- Input/Output Channel	2.4
- Interrupt Flag	2.4; 9.3.8
- Interrupt Mask	2.4; 9.3.5
- Interrupt Status	2.4; 9.3.4
- Processor Utility	2.4
- Program Counter	2.4; 9.3.1
Relocatability Attribute	3.3.2
Scratch Pad Store	2.4
Selector Channels	8.2.1
Self Defining Terms	3.3.7
Sense Bytes	8.5.4
Set Location Counter Instruction (ORG)	7.5.7
Set Program Mask Instruction (SPM)	6.7.4
Set Storage Key Instruction (SSK)	6.6.12
Shift Left Double Instruction (SLDA)	6.2.26
Shift Left Double Logical Instruction (SLDL)	6.4.18
Shift Left Single Instruction (SLA)	6.2.24
Shift Left Single Logical Instruction (SLL)	6.4.16
Shift Right Double Instruction (SRDA)	6.2.27
Shift Right Double Logical Instruction (SRDL)	6.4.19
Shift Right Single Instruction (SRA)	6.2.25
Shift Right Single Logical Instruction (SRL)	6.4.17
Space Listing Instruction (SPACE)	7.6.5
Standard Device Byte	8.5.3
Start Device Instruction (SDV)	6.6.7;8.4.1
Start New Page Instruction (EJECT)	7.6.4
Start Translation Instruction (START)	7.5.1
Statement Boundaries	3.2.1
Statement Format	3.2.2
Store Character Instruction (STC)	6.4.10
Store Halfword Instruction (SH)	6.2.22
Store Instructions (STE,STD)	6.8.11

Store Macros by Priority (MPRTY)	7.10.2
Store Multiple Instruction (STM)	6.2.23
Store Scratch Pad Instruction (SSP)	6.6.3
Store Word Instruction (ST)	6.2.21
Subtract Decimal Instruction (SP)	6.3.2
Subtract Halfword Instruction (SH)	6.2.12
Subtract Logical Instructions (SLR, SL)	6.2.13
Subtract Normalized Instructions (SER, SE, SDR, SD)	6.8.7
Subtract Unnormalized Instructions (SUR, SU, SWR, SW)	6.8.8
Subtract Word Instructions (SR, S)	6.2.11
Supervisor Call Instruction (SVC)	6.7.3
Symbol	3.3.4
Terms	3.3.1
Test Device Instruction (TDV)	6.6.9; 8.4.3
Test Under Mask Instruction (TM)	6.4.8
Translate and Test Instruction (TRT)	6.4.13
Translate Instruction (TR)	6.4.12
Translator Instruction Statements	3.1.2, 7
Translator Option Instruction (AOPTN)	7.6.3
Unpack Instruction (UNPK)	6.3.8
Use Base Register Instruction (USING)	7.2.1
Value Attribute	3.3.2
Write Direct Instruction (WRD)	6.6.13
Zero and Add Instruction (ZAP)	6.6.3

English Electric Leo Marconi is a member of English Electric which also includes English Electric Automation and the Marconi Companies. English Electric and its members have an unrivalled breadth of experience and research and development facilities which place them in the forefront of electronics and automation activities of every kind.

English Electric-Leo-Marconi Computers Limited
An English Electric Company

United Kingdom

Kidsgrove, Stoke-on-Trent, Staffordshire
Kidsgrove 2041/2141/3131

Portland House, Stag Place, London SW1
VICtoria 2299 STD 01

City

Enterprise House, 66 Wilson Street, London EC2
BIShopsgate 6862 STD 01

Midlands

65 New Street, Birmingham 2
MIDland 4008 STD 021

South West

36/38 Baldwin Street, Bristol
Bristol 21406 STD OBR 2

North West/North Midlands

St. Anne's House, Alderley Road, Wilmslow, Cheshire
Lindow Park 7475

North East

Tithe House, Horsforth, Nr. Leeds, Yorkshire
Horsforth 4678 STD OWR 34

Scotland & Ireland

Colston Road, Bishopbriggs, Glasgow
BIShopbriggs 3232 STD 041

Bureau Division

Kidsgrove, Stoke-on-Trent, Staffordshire
Kidsgrove 2041/2141/3131

London Bureau

Hartree House, Queensway
London W2
BAYswater 9451 STD 01

Training Division

Radley House, 35-39 South Ealing Road, London W5
EALing 7262/6066 STD 01

Overseas

Australia

Australian Computers Pty. Ltd, ADC House
77 Pacific Highway, North Sydney, New South Wales
Cables: Leotronic, Sydney. Tel 929-8155

Australian Computers Pty. Ltd
320/322 St Kilda Road, Melbourne SC1, Victoria
Cables: Leotronic, Melbourne. Tel 697191

Australian Computers Pty. Ltd
PO Box 1363H, Adelaide, South Australia
Cables: Enelectico, Adelaide. Tel 51-2421

New Zealand

English Electric-Leo Data Systems Ltd
PO Box 3667, Wellington C2
Cables: Leodata, Wellington, NZ. Tel 55-640

South Africa

English Electric Leo-Computer Services Pty. Ltd
PO Box 4164, Johannesburg
Cables: Leotronic, Johannesburg. Tel 838-3581

Argentina

Intercontinental Data Services S.A.I.y C
Calle Florida, Casilla de Correo 128
Buenos Aires, Argentina

© All rights reserved 1966

The policy of English Electric-Leo-Marconi Computers Limited is one of continuous development and improvement of its products and services, and the right is therefore reserved to alter or amend the information contained in this publication without notice. Although every effort has been made to avoid ambiguities, should any statement in this publication appear to be capable of more than one interpretation the statement must be referred to the Company for clarification. The Company cannot accept responsibility for wrongful interpretation of any statement which has not been referred to them for clarification.

