3100

Control Data® 3100 Computer System

Preliminary Reference Manual

3100 Computer Instruction Index

Mnemonic & Octal Code		Name	Page
HLT	00	Unconditional Stop	5-4
SJ1-6		Selective Jump 1-6	5-4
RTJ		Return Jump	5-5
UJP	01	Unconditional Jump	5-5
IJI	02	Index Jump, Incremental	5-5
IJD		Index Jump, Decremental	5-5
AZJ	03	Compare A with Zero	5-6
AQJ		Compare A with Q	5-7
ASE	04	Skip if (A) = y	5-9
QSE		Skip if (Q) = y	5-9
ISE		Skip if $(B^b) = y$	5-9
ASG	05	Skip if (A) ≧ y	5-9
QSG		Skip if (Q) ≧ y	5-9
ISG		Skip if $(B^b) \geqq y$	5-9
MEQ	06	Masked Equality Search	5-11
MTH	07	Masked Threshold Search	5-12
SSH	10	Storage Shift	5-12
ISI		Index Skip, Incremental	5-8
ISD		Index Skip, Decremental	5-8
ECHA	11	Enter A, Character Address	5-8
SHA	12	Shift A	5-10
SHQ		Shift Q	5-10
SHAQ	13	Shift AQ	5-11
SCAQ		Scale AQ	5-11
ENA	14	Enter A	5-9
ENQ		Enter Q	5-9
ENI		Enter Index	5-9
INA	15	Increase A	5-9
INQ		Increase Q	5-9
INI		Increase Index	5-9
XOA	16	Exclusive OR of A and y	5-9
XOQ		Exclusive OR of Q and y	5-9
XOI		Exclusive OR of Index and y	5-9
ANA	17	AND of A and y	5-9
ANQ		AND of Q and y	5-9
ANI		AND of Index and y	5-9
LDA	20	Load A	5-14
LDQ	21	Load Q	5-14
LACH	22	Load A, Character	5-14
LQCH	23	Load Q, Character	5-14
LCA	24	Load Complement A	5-15
LDAQ	25	Load AQ	5-15
LCAQ	26	Load Complement AQ	5-15
LDL	27	Load A Logical	5-15
ADA	30	Add to A	5-18
SBA	31	Subtract from A	5-18
ADAQ	32	Add to AQ	5-19
SBAQ	33	Subtract from AQ	5-19
RAD	34	Replace Add	5-18
SSA	35	Selectively Set A	5-13
SCA	36	Selectively Complement A	5-13
LPA	37	Logical Product A	5-13
STA	40	Store A	5-15
STQ	41	Store Q	5-15
SACH	42	Store A, Character	5-16
SQCH	43	Store Q, Character	5-16
SWA	44	Store Word Address	5-16
STAQ	45	Store AQ	5-16
SCHA	46	Store Character Address	5-16
STI	47	Store Index	5-16
MUA	50	Multiply A	5-18
DVA	51	Divide A	5-18
CPR	52	Compare	5-13
---	53	Inter-Register Transfers, 24 Bit	5-17
LDI	54	Load Index	5-15
MUAQ	56	Multiply AQ	5-19
DVAQ	57	Divide AQ	5-19
FAD	60	Floating Point Add	5-20
FSB	61	Floating Point Subtract	5-20
*FMU	62	Floating Point Multiply	5-20
*FDV	63	Floating Point Divide	5-20
*LDE	64	Load E	5-22
*STE	65	Store E	5-22
*ADE	66	Add to (E)	5-22
*SBE	67	Subtract from (E)	5-23
*SFE	70	Shift E	5-21
*EZJ		E Zero Jump	5-22
*EOJ		E Overflow Jump	5-22
*SET		Set D Register	5-22
SRCE	71	Search Character Equality	3-6
SRCN		Search Character Inequality	3-6
MOVE	72	Move Data	3-7
INPC	73	Input, Character Block to Storage	3-8
INAC		Input, Character to A	3-8
INPW	74	Input, Word Block to Storage	3-8
INAW		Input, Word to A	3-8
OUTC	75	Output, Character Block from Storage	3-8
OTAC		Output, Character from A	3-8
OUTW	76	Output, Word Block from Storage	3-8
OTAW		Output, Word from A	3-8
---	77	Sense, Select, Interrupt and Control functions	5-24

*Trapped instructions. See also Chapters 3 and 5.

3100

Control Data® 3100 Computer System

Preliminary Reference Manual

Record of Revisions

REVISION	NOTES

Address comments concerning this Manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington
St. Paul, Minnesota 55112

or use Comment Sheet located in the rear of this book.

Pub. No. 60108400
July, 1964

CONTENTS

CHAPTER 1. SYSTEMS HARDWARE DESCRIPTION

System Concepts 1-1
Summary of 3100 Characteristics 1-1
3100 Computer System 1-2
Peripheral Equipment 1-7

CHAPTER 2. SYSTEMS SOFTWARE DESCRIPTION

3100 SCOPE 2-1
3100 COMPASS 2-1
3100 Data Processing Package 2-2
3100 COBOL 2-3
3100 FORTRAN 2-3
3100 Generalized Sort/Merge Program 2-4
Basic System 2-4

CHAPTER 3. PROGRAMMING FEATURES

Program Interrupts 3-1
Special Power Failure Interrupts 3-3
Trapped Instructions 3-3
Integrated Register File 3-4
Real-Time Clock 3-5
Block Operations 3-5

CHAPTER 4. OPERATING FEATURES

Displays and Indicators 4-1
Switches 4-3

CHAPTER 5. REPERTOIRE OF INSTRUCTIONS

General Information 5-1
Instructions 5-3

APPENDIXES

A. 3100 Compass
B. BASIC Assembler Coding Procedure
C. Number Systems
D. Table of Powers of Two
E. Octal-Decimal Integer Conversion Table
F. Octal-Decimal Fraction Conversion Table
G. Definition of I/O Interface Signals
H. 3100 System Character Set
I. Peripheral Equipment Code

FIGURES

1-1 CONTROL DATA 3101 Desk Console 1-3
1-2 Parity Bit Assignment 1-3
1-3 Word Addressed Instruction Format .. 1-4
1-4 Character Addressed Instruction Format 1-4
1-5 Storage addressing and Data Paths of Typical Installation 1-6
3-1 Search Operation 3-6
3-2 Move Operation 3-7
3-3 Initial Steps of I/O Sequence 3-8
3-4 Input, Character or Word to A 3-9
3-5 Output, Character or Word from A ... 3-9
3-6 Input, Character Block to Storage 3-11
3-7 Input, Word Block to Storage 3-11
3-8 Output, Character Block from Storage 3-12
3-9 Output, Word Block from Storage 3-12
4-1 Integrated Console 3-1
4-2 Temperature Warning Designations for Fully Expanded 3104, Front View ... 4-3
4-3 3104 Console Keyboard 4-5
5-1 General Machine Code Instruction Formats 5-1
5-2 Indirect Addressing Routine 5-2

TABLES

1-1 Optional Memory Configuration 1-2

1-2 Properties of Arithmetic and Control Registers 1-6

1-3 Tape Transport Characteristics 1-7

1-4 Tape Transport Controllers 1-7

1-5 Card Reader Characteristics 1-7

1-6 Card Reader Controller Characteristics 1-8

1-7 Card Punch Characteristics 1-8

1-8 Card Punch Controller Characteristics 1-8

1-9 Paper Tape Reader Punch Characteristics 1-8

1-10 Line Printer Characteristics 1-8

1-11 Printer Controller Characteristics 1-8

3-1 Interrupt Mask Bit Assignments....... 3-2

3-2 Interrupt Priority........................... 3-2

3-3 Representative Interrupt Codes........ 3-3

3-4 List of Trapped Instructions............ 3-3

3-5 Integrated Register File Assignments. 3-4

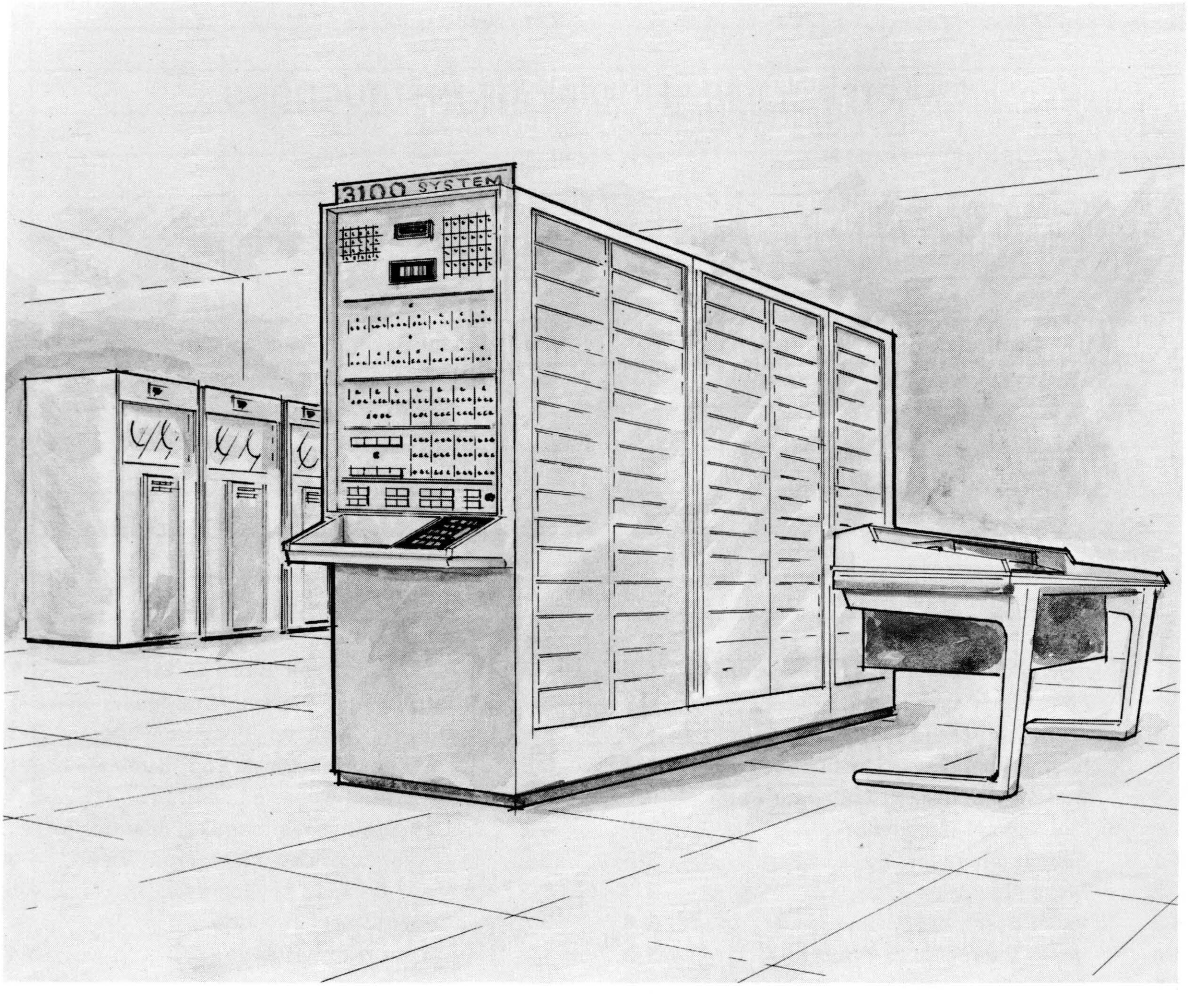
3-6 Block Operations 3-5

4-1 Register Displays........................ 4-2

4-2 Main Console Operator Switches 4-4

4-3 Keyboard Switches 4-6

4-4 Maintenance Switches 4-7



1

Systems Hardware Description

System Concepts

The CONTROL DATA* 3100 is a medium-size, solid-state, general-purpose digital computing system. Advanced design techniques used in the system provide for fast solutions to data processing, scientific and real-time problems. Modular construction is utilized by 3100 computers to permit adaptation to the design requirements of exacting installations.

The 3100 is program compatible to the CONTROL DATA 3200 computer system and is consistent with the Input/Output Specification for the 3000 computer series. An integrated register file and block control system are used in all 3100 computers and trapped instructions include those pertinent to BCD, floating point, and 48-bit precision multiply and divide.

A complete line of peripheral equipment may be incorporated into a 3100 system, including the following:

—CONTROL DATA 601, 604 and 607 Tape Transports which are ½-inch magnetic tape units that can handle binary or BCD data, recording at densities up to 800 bits per inch with tape speeds from 37.5 inches/second to 150 inches/second reading forward or backward.

—CONTROL DATA 405 Card Reader which reads cards at a 1200 card per minute rate.

—CONTROL DATA 501 Line Printer which prints 136 character lines at up to 1000 lines per minute.

Also available are a paper tape reader/punch, a medium speed line printer, and an I/O typewriter.

Summary of 3100 Characteristics

GENERAL

—Stored-program, general-purpose computer
—Parallel mode
—Solid-state logic
—Real time clock
—Program interrupt

COMPUTER

—Complete repertoire of instructions, word and character oriented
—Three index registers
—Arithmetic
    fixed point 24-bit precision
    fixed point 48-bit precision add and subtract
    fixed point 48-bit precision multiply and divide. (trapped)
    floating point with 36-bit coefficient, biased exponent, and 1-bit coefficient sign (trapped)
    BCD (trapped)
—Typical instruction execution time
    fixed point 24-bit addition, 3.5 μs with storage access

STORAGE

—Magnetic core memory
—Word size
    24-bit words with four characters per word
    4 parity bits, one per character
—4,096 words/16,384 characters, basic memory size; expandable to 8,192, 16,384, or 32,768 words
—1.75 μsec complete cycle time
—1.0 μsec storage access time
—Indirect addressing

INPUT/OUTPUT

—Standard
    one 12-bit bidirectional I/O channel
—Optional
    3 additional 12-bit channels, or
    2 additional 12-bit channels and
    1 additional 24-bit channel
—Data transfer rate up to 3.3 megabits/second
—Mediums
    magnetic tape, punched cards, paper tape, printed forms

CONSOLES

—Standard
    integrated console with binary displays and detachable keyboard
—Optional
    separate desk console including detachable keyboard and on line typewriter

SOFTWARE

—Operating system: 3100 SCOPE
—Assembly program: 3100 COMPASS
—Basic System (3104 4K memory oriented)
—3100 Data Processing Package used with the assembly program under the operating system; business and I/O Macros
—Business language compiler: 3100 COBOL
—Scientific language compiler: 3100 FORTRAN

*Registered Trademark of Control Data Corp.

3100 Computer System

A 3100 computer system consists of combination logic modules selected by the customer to best fit his needs.

3104 COMPUTER

The 3104 computer contains arithmetic and control logic to perform 24-bit precision fixed point arithmetic, 48-bit precision fixed point addition and subtraction, Boolean, character and word handling, and decision making operations. The computer will also execute BCD, floating point and 48-bit precision multiply and divide as trapped instructions.

The 3104 computer uses a panel type console integrated with the main frame of the computer. The panel is mounted at one end of the cabinet and is equipped with binary displays, control switches, monitor loudspeaker and removable keyboard to facilitate remote operation.

A 4,096 word memory and a 12-bit bidirectional data channel are also incorporated in the 3104.

COMMUNICATION CHANNELS (I/O)

The following I/O channels are available for 3100 computing system.

3106 Communications Channel

The 3106 is a bidirectional, 12-bit, parallel data channel. Up to eight peripheral equipment controllers may be connected in parallel to one channel. One module may be installed in the main computer cabinet. Additional modules must be contained in adjacent cabinets. A maximum of four 3106 channels may be used with any 3100 system.

3107 Communications Channel

In lieu of two 3106 channels, one 3107 may be used. The 3107 is a bidirectional, buffered 12- or 24-bit data exchange communication channel. It features 12 to 24 bit assembly, disassembly and permits attachment of one to eight peripheral controllers to a 3100 system. Only one 3107 can be used per 3100 computer system.

CONSOLES

Two consoles are available for use in the 3100 computer system. They are electrically compatible; however, only one type may be used in a system.

Integrated Console

The integrated console, standard on a 3104 computer, is a panel type mounted on the end of the main computer frame. This console features binary displays monitor loudspeaker and a removable keyboard for remote operation. The 3192 on-line monitor typewriter which connects directly to the computer, is ordered separately when the integrated console is used in a system.

3101 Desk Console

The 3101 desk console is electrically identical to the integrated console but features a condensed display and control unit mounted above an on-line monitor typewriter included with this console. The 3101 is optional in 3100 systems. Figure 1-1 illustrates a 3101 desk console.

OPTIONAL STORAGE

A customer may select a combination of magnetic core storage (MCS) modules to increase the total storage capacity of his 3104 computer system to 8,192, 16,384 or 32,768 words. The following storage modules are available:

3108—Optional 4,096 word (16,384 characters) MCS memory module.

3109—Optional 8,192 word (32,768 characters) MCS memory module.

3103—Optional 16,384 word (65,536 characters) MCS memory module.

Memory configurations are shown in table 1.

Table 1-1. Optional Memory Configurations

Total Expanded Memory Capacity	Memory Modules Required in Addition to 4K Memory in 3104
8K	3108
16K	3108 and 3109
32K	3108, 3109 and 3103

STORAGE CHARACTERISTICS

Storage modules in a 3104 computer system are composed of fields, consisting of 4,096 words, 28 bits per word. A particular system may have 1, 2, 4, or 8 such fields. These fields operate together as one large storage system during the execution of stored programs.

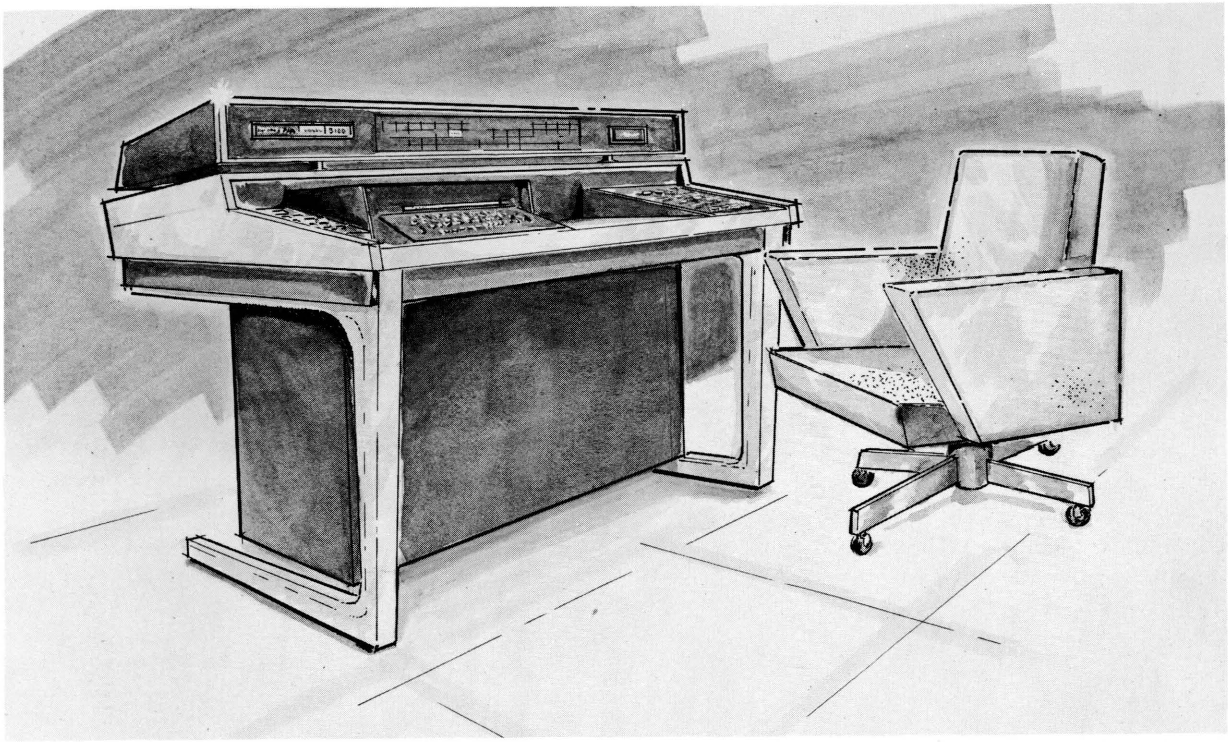


Figure 1-1.
Control Data 3101 Desk Console

Storage Word

Storage words contain 28 bits. Twenty-four of these are for information; four are for parity.

Parity

For parity checking purposes, each storage word is broken into four 6-bit groups, each of which has one parity bit associated with it. Figure 1-2 shows parity bit assignments.

During each write cycle, a parity bit is stored along with each group. When part or all of a word is next read from storage, the appropriate parity bit(s) accompany the word to the control section of the chassis where it is checked for a loss or gain of bits. The 3100 uses odd parity. That is, the total number of "1's" in a character, plus the parity bit, is always an odd number. Any failure to produce the correct parity during read operations causes a memory fault indication that is followed by an immediate program halt. This halting may be avoided by use of the Disable Parity switch. An indicator light on the storage module control panel indicates a parity error.

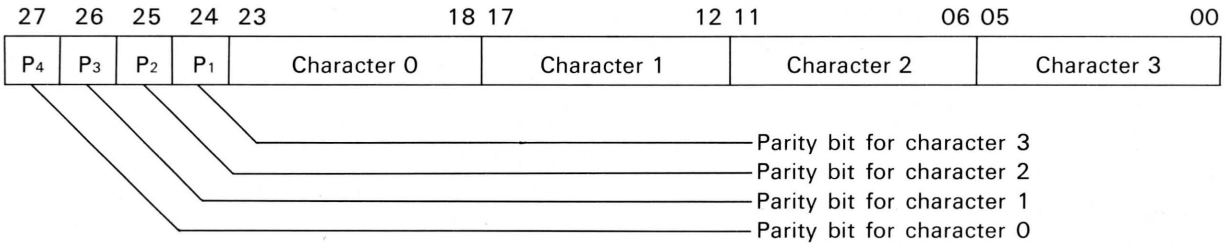

27 26 25 24 23 18 17 12 11 06 05 00
P_4 P_3 P_2 P_1
Character 0
Character 1
Character 2
Character 3
Parity bit for character 3
Parity bit for character 2
Parity bit for character 1
Parity bit for character 0

Figure 1-2. Parity Bit Assignments

Storage Addressing

Most instructions used with the 3100 computer refer to a unique storage word or to a character within a particular word.

Word Addressing

Figure 1-3 shows the format of a word addressed instruction.

Character Addressing

Figure 1-4 shows the format of a character addressed instruction.

Storage Sharing

Two 3104 computers may share the use of a common storage module. A switch on each storage module control panel allows the operator to give exclusive control to the right- or to the left-hand computer. A middle position on this switch actuates a two-position priority scanner. The requests are honored by storage control on a nonpriority basis. Neither computer has priority over the other. The computer being serviced by the current storage cycle relinquishes control to the awaiting computer at the end of the cycle. Either computer can therefore be delayed a maximum of one storage cycle. A two-position scanner within each computer determines whether main control or block control has access to the storage module; thus a similar program delay may occur within either computer.

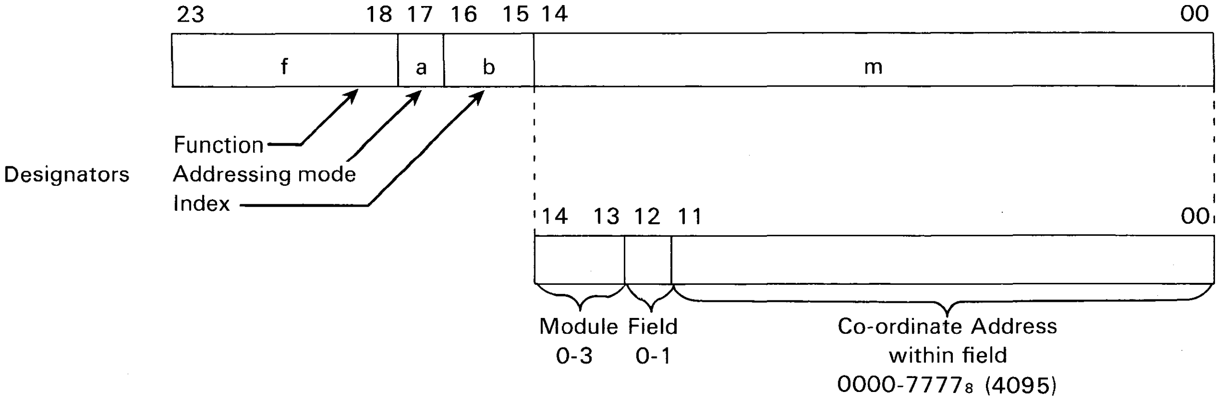


Figure 1-3. Word Addressed Instruction Format

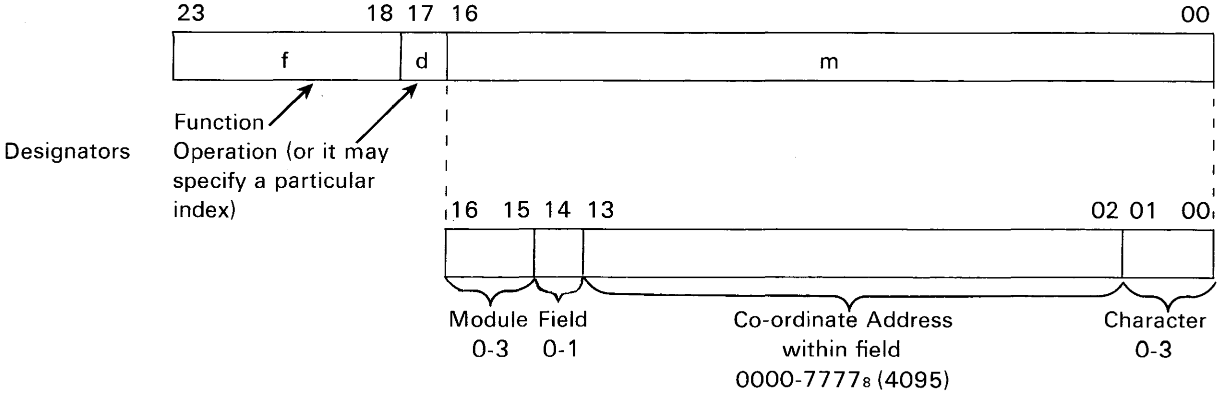

23
18 17 16
00
f
d
m
Designators
Function
Operation (or it may specify a particular index)
16 15 14 13
02 01 00
Module 0-3
Field 0-1
Co-ordinate Address within field 0000-7777₈ (4095)
Character 0-3

Figure 1-4. Character Addressed Instruction Format

Registers Associated with Storage

Two registers are associated with each storage module: S and Z.

- The 13-bit S Register contains the address of the word being currently processed. Bit 12 specifies field 0 or field 1. Bits 00-11 specify the co-ordinates of the word.
- The 28-bit Z Register is the storage restoration and modification register.

Read/Write Control

During a normal memory cycle, all bits of a word referenced by the (S) are read out of core storage in parallel, loaded into Z, used for some purpose, then written back into storage, intact. Five modes exist in the 3100 computer for storage modification. In all cases, assume that Z is initially in the cleared state.

Single-Character Mode. Any one character may be inhibited during the read cycle. New data is then loaded into the corresponding character position of Z and the whole (Z) is stored.

Double-Character Mode. The upper, middle, or lower half of a word is inhibited during the read cycle. New data is loaded into the unfilled half of Z and the whole (Z) is stored.

Triple-Character Mode. Either of the two possible triple-character groups may be inhibited during the read cycle. New data is then loaded into the corresponding character positions of Z and the whole (Z) is stored.

Full-Word Mode. The whole word is inhibited during the read cycle. A new word is entered into Z and the (Z) is stored.

Address Mode. The lower 15 or 17 bits of a word may be inhibited during the read cycle. A new word or character address is then loaded into Z, and the whole (Z) is stored.

After all write cycles, Z is cleared unless the computer has stopped as the result of a memory parity error.

ARITHMETIC SECTION

The arithmetic section of the 3100 computer consists of two operational registers. They are displayed on the console and each may be loaded from the entry keyboard. These registers are the:

A — arithmetic register

Q — auxiliary arithmetic register

The A register (accumulator) is the principal arithmetic register. Some of the more important functions of A are:

1. All arithmetic and logical operations use the A register in formulating a result. The A register is the only register with provisions for adding its contents to the contents of a storage location or another register.
2. *Shifting* — A may be shifted to the right or left separately or in conjunction with Q. Right shifting is open-ended; the lowest bits are discarded and the sign is extended. Left shifting is circular; the highest order bit appears in the lowest order stage after each shift; all other bits move one place to the left.
3. *Control for conditional instructions* — A holds the word which conditions jump and search instructions.

The Q register is an auxiliary register and is generally used in conjunction with the A register. The principal functions of Q are:

1. Providing temporary storage for the contents of A while A is used for another arithmetic operation.
2. Forming a double-length register, AQ.
3. Shifting to the right or left, separately or in conjunction with A.
4. Serving as a mask register for 06, 07, and 27 instructions.

Both A and Q may load, or be loaded from any of the three index registers without the use of storage references.

CONTROL SECTION

The control section contains five operational registers. As in the arithmetic section, these registers are displayed on the console and loaded from the entry keyboard. They are the:

F — program control register

P — program address counter

B^1 through B^3 — index registers

The program control register, F, holds an instruction during the time it is being executed. After executing an instruction, an *exit, jump exit,* or *skip exit* is performed. An exit advances the count in P by one and executes the next instruction specified by the contents of P. A jump exit executes the instruction at the storage location specified by the execution address of the jump instruction. The execution address is, in this case, entered into P and used to specify the starting location of a new sequence of instructions. A skip exit advances the count in P by two, bypassing the next sequential instruction and executing the following one.

The P register is the program address counter. It provides program continuity by generating in sequence the storage addresses which contain the individual instructions. Usually at the completion

Table 1-2. Properties of Arithmetic and Control Registers

Register	No. of Stages	Modulus	Complement Notation	Arithmetic	Result
A	24	2^{24}-1	one's	additive	signed*
Q	24	2^{24}-1	one's	additive	signed
F	24	2^{24}-1	**	**	**
P	15	2^{15}-1	one's	additive	unsigned
B^1-B^3	15	2^{15}-1	one's	additive	unsigned

of each instruction, the count in P is advanced by one to specify the address of the next instruction.

The three index registers, B^1 through B^3, provide storage for quantities which are used in a variety of ways, depending on the instruction. The B registers have no provisions for arithmetic operations. In a majority of instructions they hold quantities to be added to the execution address. All address modifications are performed in the Adder.

Table 1-2 is a summary of the properties of the A, Q, F, P, and B registers.

A sixth operational register, closely related to the control section, is the communications register. Quantities to be entered into any of the above registers or into storage from the entry keyboard are temporarily held in the communications register until the transfer button is pushed. If a mistake is made while entering data into the communications register, the Keyboard Clear button may be used to clear this register.

COMPUTER ORGANIZATION

All modules of the 3100 computer except the console are connected in parallel to a common bidirectional data bus. The address registers of all storage modules are connected in parallel to main control by the address bus. Figure 1-5 is a block diagram of storage addressing and data paths within a typical computer installation.

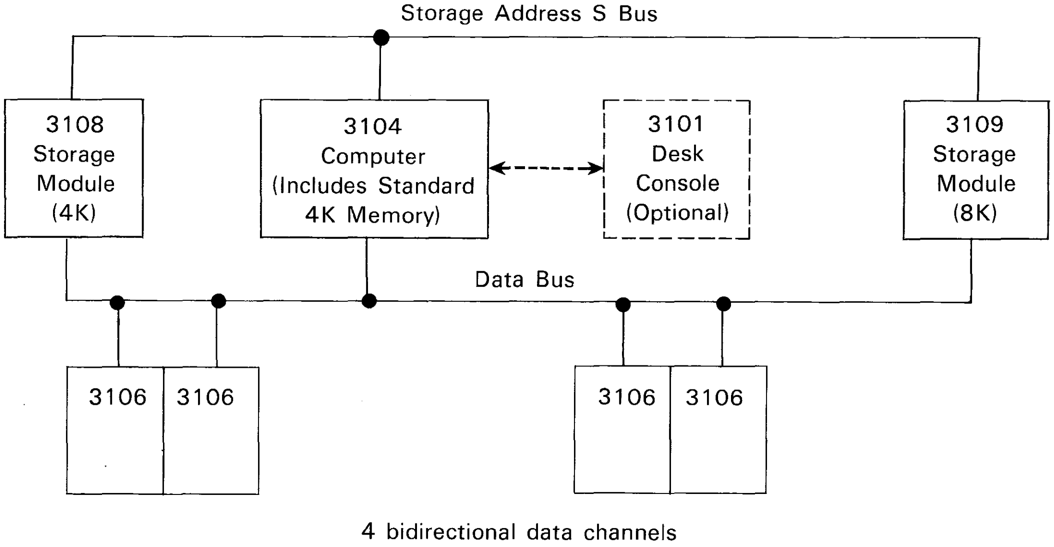

Storage Address S Bus
3108 Storage Module (4K)
3104 Computer (Includes Standard 4K Memory)
3101 Desk Console (Optional)
3109 Storage Module (8K)
Data Bus
3106
3106
3106
3106
4 bidirectional data channels

Figure 1-5. Storage Addressing and Data Paths of Typical Installation

*NOTE: The result of an arithmetic operation in A satisfies A $\leq 2^{23}$-1, since A always is treated as a signed quantity. When the result in A is zero, it is always represented by 00000000.

**NOTE: Only the lower 15 or 17 bits of F are modified, depending on whether word or character addressing is being used. The results are unsigned.

Peripheral Equipment

Peripheral equipment is available for handling magnetic tape, punched cards, punched paper tape, and printed forms. Other pieces of equipment for the 3100 computer system are a program controlled I/O typewriter, an incremental plotter, and a Satellite coupler. For details on any particular piece of peripheral equipment, refer to the reference manual concerning that equipment.

MAGNETIC TAPE

Magnetic tape is processed on either the CONTROL DATA 601, 604 or 607 Tape Transports. A variety of tape transport controllers is available, each with a different capability.

Tape Transports

Table 1-3 lists the operating characteristics of the 601, 604 and the 607 Tape Transports. Tapes may be read forward or backward with both models.

Tape Transport Controllers

Tape transport controllers are differentiated by the number of read/write controls they contain and by the number of tape transports that they can control. Eight types are available (see table 1-4).

The tape transport controllers marked by a dagger (†) will most commonly be selected for a 3100 computer system. A multi-channel controller may be used for buffered communication between two or more computers in a multi-computer installation.

Table 1-3. Tape Transport Characteristics

Characteristic	601	604	607
Tape length	2400 feet	2400 feet	2400 feet
Tape width	½-inch	½-inch	½-inch
Tape speed	37.5 inches/sec	75 inches/sec	150 inches/sec
Word size including one parity bit	7 bits	7 bits	7 bits
Bit density	200, or 556 bpi	200, 556, or 800 bpi	200, 556, or 800 bpi
Maximum bit transfer rate	7.5 or 20.85 kc	15, 41.65, or 60 kc	30, 83.3, or 120 kc

Table 1-4. Tape Transport Controllers

Model Number	No. of Read Write Controls	Maximum No. of Tape Transports
†3127	1	4
†3228	1	4
†3229	1	8
3621	2	8
3622	2	16
3623	4	8
3624	4	16
3625	3	8
3626	3	16

PUNCHED CARDS

Cards are read with a Control Data 405 Card Reader and punched with an IBM 523 or 544 Card Punch.

Card Reader. Table 1-5 lists the operating characteristics of the 405 card reader.

Card Reader Controllers. Two card reader controllers are available. Table 1-6 lists the characteristics of each. Both types of controllers are mounted on chassis within the 405 cabinet.

Table 1-5. Card Reader Characteristics

Speed — 80 column cards	1200 cpm
Speed — 51 column cards	1600 cpm
Reading method	photo-electric, column-by-column
Verification method	double read — comparison
Card separation and picking method	pneumatic
Card capacity — main tray	4000 cards
Card capacity — reject tray	240 cards

Table 1-6.
Card Reader Controller Characteristics

Characteristics	3248	3649
BCD Conversion	Yes	Yes
Checking	Yes	Yes
Full card buffer	No	Yes
No. of read controls	1	2

Card Punches. Table 1-7 lists the operating characteristics of the 523 and 415 card punches.

Card Punch Controllers. Two types of card punch controllers may be used. Each type is mounted in its own peripheral equipment cabinet. Table 1-8 lists the controller characteristics.

Table 1-7. Card Punch Characteristics

Characteristics	523	415
Speed-80 column cards	100 cpm	250 cpm
Card hopper capacity	800 cards	1200 cards
Punch method	Mechanical, row-by-row	

Table 1-8.
Card Punch Controller Characteristics

Characteristics	3245	3644
Checking	No	Yes
Full card buffer	No	Yes
No. of Write controls	1	2

PUNCHED PAPER TAPE

A unit frequently used for reading programs into storage and for recording data from storage is the 3691 Paper Tape Reader Punch. Table 1-9 lists the characteristics of this device.

Table. 1-9.
Paper Tape Reader Punch Characteristics

Reading speed	350 characters/sec
Punching speed	110 characters/sec
No. of read/write controls	1

PROGRAM CONTROLLED I/O TYPEWRITER

The 3692 Program Controlled I/O Typewriter has one read/write control. It differs from the online 3192 typewriter in that it must be connected to the computer via a 3106 Communication Channel.

INCREMENTAL PLOTTER

The 3293 Incremental Plotter can make 300 .01 inch steps per second. Form width is 11 inches.

PRINTED FORMS

The 501 High Speed Line Printer is available for all 3100 computer systems. The printer and controller characteristics are listed in Tables 1-10 and 1-11.

Table 1-10. Line Printer Characteristics

Characteristics	501
Printing speed	1000 lpm
No. of characters	64
No. of columns	120

Table 1-11. Printer Controller Characteristics

Characteristics	3256	3659
No. of write controls	1	2
Full line buffer	Yes	Yes

SATELLITE COUPLER

The 3682 Satellite Coupler permits direct connection between any two standard 12-bit bidirectional channels, or channel converters. With the addition of a 3681 Data Channel Converter, a 160-A Computer may be used as a satellite to the 3100 computer system.

2

Systems Software Description

There are various programming language techniques which facilitate writing programs for the CONTROL DATA 3100 Computer System. The following pages contain a synopsis of the methods listed below.

- 3100 SCOPE
  Monitor System
- 3100 COMPASS
  Assembler
- 3100 DATA PROCESSING PACKAGE
  Macro Instructions, Generalized I/O
- 3100 COBOL
  Business Language Compiler
- 3100 FORTRAN
  Scientific Language Compiler
- 3100 GENERALIZED SORT/MERGE PROGRAM
  Operates Under 3100 SCOPE
- BASIC SYSTEM
  Basic Assembler, Basic FORTRAN II

3100 SCOPE

SCOPE is the operating system for the CONTROL DATA 3100 Computer. Modular in structure, the system provides efficient job processing while minimizing its own memory and time requirements. Programming with the operating system is simplified by the use of control cards which are included with program decks. Among the functions performed by SCOPE are the following:

JOB PROCESSING

- processes stacked or single jobs
- controls I/O and interrupt requests
- monitors compilations and assemblies
- loads and links object subprograms
- stores accounting information
- initiates recovery dumps
- prepares overlay tapes

EQUIPMENT ASSIGNMENTS

- logical unit references
- physical unit assignment at run time
- drivers for all standard peripheral equipment
- system units which facilitate job processing and minimize monitor programming

DEBUGGING AIDS

- extensive diagnostics
- octal corrections
- snapshot dumps
- recovery dumps

LIBRARY PREPARATION AND EDITING

- prepare a new library
- edit an existing library
- list the contents of a library

3100 COMPASS

COMPASS is the comprehensive assembly system for the CONTROL DATA 3100 Computer. Operating under 3100 SCOPE, it assembles relocatable machine language programs. The program may consist of subprograms, each of which may be independently assembled. Refer to Appendix A for 3100 COMPASS coding procedures. COMPASS source language includes the following features:

Operation codes — Machine operations are written as one or more mnemonic or octal subfields.

Addressing — Expressions, used as addresses, may represent either word or character locations. Expressions consist of symbols, constants, and special characters connected by + and −.

Data storage — A data area, shared by subprograms, may be specified and loaded with data in the source program.

Common storage — A common area may be designated to facilitate communication among subprograms.

Data definitions — Constants may be defined as octal, decimal, double-precision, integer or floating-point numbers; BCD words, BCD characters; or contiguous strings of bits.

Library access — Library routines may be called by reference to their entry points or by inclusion of macros in the source program (data processing macros).

Listing control — The format of the assembly listing may be controlled by pseudo instructions.

Diagnostics — Diagnostics for source program errors are included with the output listing.

Macro instructions — Macros may be defined in the source program or entered into the library; the sequence of instructions will be inserted whenever the macro name appears in the operation field.

THE ASSEMBLER

The 3100 COMPASS assembly program converts programs written in 3100 COMPASS source language into a form suitable for execution under the 3100 SCOPE operating system. Source program input may be on punched cards or in the form of card images on magnetic or paper tape. The output from the assembler includes an assembly listing and a relocatable binary object program on punched cards or magnetic tape.

EQUIPMENT CONFIGURATION

The assembly system, which is stored on the SCOPE library tape, is designed to operate on a 3100 computer with a minimum of 8,192 words of storage. In addition to the SCOPE library unit, the following input/output equipment is required:

Input unit: card reader, magnetic tape, or paper tape

Scratch unit: magnetic tape (may also be used for output)

Listable output unit: magnetic tape or printer

Object program output unit: magnetic tape or card punch

PROGRAM STRUCTURE

Source programs may be divided into subprograms which are assembled independently. All location symbols except COMMON and DATA symbols are local to the subprogram in which they appear, unless they are declared as external symbols. Locations which will be referenced by other subprograms are declared as entry points. For example, if subprogram IGOR references locations KIEV and MINSK in subprogram DEMETRI, KIEV and MINSK must be declared external symbols in subprogram IGOR and entry points in subprogram DEMETRI.

The links among subprograms are associated by the SCOPE loader. As each subprogram is loaded, all external symbols and entry points are entered into a symbol table. When an external symbol is found which matches an entry point already entered in the table, or an entry point is found which matches an external symbol, linkage between the two points is established.

If any external symbols are not matched with entry points after the last subprogram is loaded, the library tape is searched for routines with the names of unmatched symbols. If these routines are found, they are loaded and linked to the other subprograms. If external symbols remain for which there has been no corresponding entry, the job is terminated and an error message written by the system.

3100 Data Processing Package

The Data Processing Package is composed of a set of data processing routines and a generalized input/output system.

DATA PROCESSING ROUTINES

The data processing routines, called macros, are used in COMPASS assembly language programs to do particular data handling jobs; included are the following:

TRANSMIT	Transmits any string of up to 4,095 characters from one place in memory to another.
COMPARE	Compares any string of up to 4,095 characters with any other string and sets a register to indicate whether the first string is lower, equal, or higher than the second.
EDIT	Moves a numeric field to a receiving field with report editing.
MULTIPLY	Multiplies two BCD numbers and stores the result in a third.
DIVIDE	Divides one BCD number by another and stores the result in a third.

GENERALIZED INPUT/OUTPUT SYSTEM

The 3100 Generalized Input/Output System is a series of library routines which provide complete input/output control for data processing. These routines are used in COMPASS assembly programs; they simplify programming while offering versatile data handling and optimum usage of internal storage space and processing time. Complete, partial or no buffering may be designated, depending upon the amount of storage the programmer has available; multi-file reels or multi-reel files may be read or written; fixed or variable length logical or physical records may be processed; and magnetic tape, paper tape, cards or printer may be used for input/output units. Both labeled and unlabeled tapes may be handled. The input/output macros perform the following functions:

OPENI	Opens an input file
OPENO	Opens an output file
READ	Reads one logical record into the record area
WRITE	Writes one logical record from the record area
READI	Reads one logical record into a specified area in memory
WRITEF	Writes one logical record from a specified area in memory
CLOSE	Closes a reel or file

In addition to the input/output operations, the programmer also describes the files to be processed through use of macros.

FIELDESC Defines logical records, buffers, logical units, recording density and rerun requirements.

LABELING Describes file label and tape retention time (prevents accidental destruction of tapes).

VARIABLE Indicates whether the size of a variable length record is determined by a record mark or a key field.

SHAREBUF Allows user to let files share the same areas in storage.

MULTIFIL Defines multi-file reels.

The I/O System interprets each set of instructions, refers to the file description, and then initiates the requested operation; it controls buffering, transmission errors, and logical-physical record divisions.

3100 COBOL

COBOL is a programming system designed to facilitate the solution of business data processing problems. To use COBOL, the programmer describes the problem in a language resembling English; the 3100 COBOL processor translates this source language input into relocatable machine language for program execution.

The 3100 COBOL language contains the elements set forth in the official Department of Defense *Report Describing COBOL 1961,* plus many of the features defined as elective COBOL.

A COBOL source program is specified in four divisions: IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE. The IDENTIFICATION division identifies the name, author, date, and so forth of the program. The ENVIRONMENT division defines the computer configuration required for both compilation and execution. The DATA division describes the format of the data files which the program is to process. The PROCEDURE division contains a sequence of statements which describe the processing to be performed.

The 3100 COBOL compiler is a three-pass system. No object code is produced until the entire source program has been thoroughly analyzed. Wherever possible, in-line coding is produced. Depending on the needs of the program, the compiler provides an input/output system which allows variable length records, up to two buffer areas per file, multi-file reels, multi-reel files, rerun procedures, and so forth. In general, the features of the 3100 COBOL input/output system correspond to those described for the Data Processing package.

3100 FORTRAN

The 3100 FORTRAN system incorporates a problem-oriented language that facilitates simple algebraic solution of mathematical or scientific problems.

3100 FORTRAN programs are written as a sequence of statements, using familiar arithmetic operations and English expressions. Large programs may be written independently in sections, the sections tested, then executed together.

Statements are available to reserve areas of memory for variables and arrays. Strings of values may be loaded with the program for reference during the program execution. Equivalence statements allow the same areas of memory to be identified with different variables and arrays during the execution of a program.

Type statements specify the mode in which values are to be stored. The possible types include: REAL, INTEGER, and CHARACTER. The programmer may also declare a special mode, type OTHER, to handle information which does not conveniently conform to the standard modes.

Arithmetic expressions are indicated by arithmetic sign and algebraic names. For example, A+B−C means add A to B and subtract C. Logical and relational operators are available for use in expressions which may be true or false.

Statements are usually executed in sequence. However, control statements may be used to transfer to another part of the program.

Sets of statements which are to be executed several times with minor changes or increments may be written once with a statement to indicate how many times they are to be repeated, and if they are to be changed each time.

Input/output operations provide a means to read information into the machine from various sources and to record results on a selected output device. If buffered input/output operation is specified, other operations may continue while information is read in or out.

Facilities are also available to transfer a number of characters from one area of memory to another, and to test machine conditions through calls to 3100 FORTRAN library functions.

The 3100 FORTRAN compiler produces machine language programs which may be executed immediately or stored for execution at a later date.

3100 Generalized Sort/Merge Program

The GENERALIZED SORT/MERGE PROGRAM organizes data on magnetic tape into one continuous predetermined order. SORT/MERGE operates under the 3100 SCOPE operating system. Control cards, read from the standard input unit, contain file descriptions and SORT/MERGE specifications.

SORT/MERGE orders fixed or variable length tape records, blocked or unblocked, written in either BCD or binary mode, according to a specified collating sequence. BCD and binary collating sequences are provided within SORT/MERGE, or the user may specify his own. The resultant output file may be merged with other presorted files in a final merge pass, or, if a number of presorted files exist, the merge phase only can be performed.

The SORT/MERGE can transfer to user prepared subroutines which perform the following functions:

- edit acceptable records
- reject records
- check nonstandard labels
- modify nonstandard labels
- generate messages for the operator
- write secondary output file (edit sorted records)
- prepare summary file (summarize sorted records)
- terminate the sort process

The SORT/MERGE checks standard header and trailer labels and provides rerun dumps.

The SORT/MERGE contains an internal sort phase and a merge phase. The sort uses the tournament replacement technique which makes maximum use of available core storage and takes advantage of existing bias in the data. The method of merging, which is selected by the user; can be normal balanced or polyphased with either forward or backward reading.

Basic System

The BASIC system is designed for the CONTROL DATA 3104 computer with a standard 4K internal storage memory. This system may also be used with the 3104 computers equipped with expanded memory modules up to 32K. Appendix B provides coding procedures for the BASIC Assembler. Included in the BASIC system are:

BASIC ASSEMBLER
BASIC LOADER
BASIC FORTRAN II

BASIC ASSEMBLER AND LOADER

The BASIC Assembler language forms a subset of the COMPASS language. Although designed primarily for use on the 3104 with a 4K memory, it can readily be used on larger systems. Object programs produced by the BASIC Assembler are loaded by the BASIC Loader or can be loaded by 3100 SCOPE. Source language programs must be prepared as complete entities if they are to be loaded by the BASIC loader. As a result, facilities for referencing external storage areas (COMMON, DATA) and external program elements (ENTRY, EXT, macros) are not used in BASIC Assembler language, nor are a few of the more complex pseudo instructions (VF, IF). All other features of the language are similar: operation codes, addressing, data definitions, listing control, and so forth.

To assemble a BASIC Assembler program, the following configuration is required:

4K words of storage

Input unit: card reader, magnetic tape or paper tape (used for source language input, library, and BASIC Assembler)

Listable output unit: printer, magnetic tape, paper tape, typewriter

Object program output unit: card punch, magnetic tape, paper tape, typewriter (all output may be written on one tape unit if desired)

BASIC FORTRAN II

BASIC FORTRAN II is a problem-oriented language that performs familiar mathematical operations in arithmetic expressions and replacement statements. The source language provides substantial power and flexibility through a variety of statements. BASIC FORTRAN II is compatible with other FORTRAN II systems and provides many of the features incorporated in 3100 FORTRAN.

3

Programming Features

This chapter discusses the following programming features of the 3100 computer system:

- program interrupts
- special power failure interrupt
- trapped instructions
- integrated register file
- real-time clock
- block operations

Program Interrupts

The interrupt control section of the 3104 computer provides for testing whether certain internal and external conditions exist without having these tests in the main program. Examples of these conditions are internal faults and external equipment end-of-operation. Near the end of each RNI cycle, a test is made for these conditions. If one of the conditions exists, execution of the main program halts. The contents of the Program Address register, P, are stored and an interrupt routine is initiated. This interrupt routine, which has been initially stored in memory, takes the necessary action for the condition and then jumps back to the next unexecuted step in the main program.

There are three major types of interrupts in the 3100 Computer System—normal interrupts (including internal and external conditions), trapped instruction interrupts, and a special power failure interrupt.

Normal interrupts are the only interrupts that are completely under the programmer's control. These interrupts are of two types—internal and external. The following paragraphs describe the interrupt causing conditions, the Interrupt Mask register, interrupt control, and interrupt processing.

INTERNAL INTERRUPTS

Seven internal conditions may be set to cause an interrupt. These conditions and their definitions are:

- *Arithmetic Overflow Fault*
  The Arithmetic Overflow fault is set when the capacity of the adder is exceeded. Its capacity, including sign, is 24 or 48 bits for 24-bit precision and 48-bit precision, respectively.
- *Divide Fault*
  The divide fault sets if a quotient, including sign, exceeds 24 or 48 bits for 24-bit precision or 48-bit precision, respectively. Therefore, attempts to divide by too small a number result in a divide fault.
- *Exponent Overflow/Underflow Fault*
  During a trapped floating point multiplication and division, the Exponent Overflow/Underflow is set if the exponent exceeds 2^{10}-1.
- *BCD Fault*
  A BCD Fault is set if a BCD Trapped instruction is executed.
- *I/O Channel Interrupts*
  Any of the four possible I/O channels will generate an interrupt:
  1) Upon reaching the end of an input or output block, or
  2) Upon receiving an End of Record (Disconnect) signal from an external device.
- *Search/Move Interrupt*
  The Search/Move interrupt is generated during a 71 or 72 instruction:
  1) Upon the completion of an equality or inequality search, or
  2) Upon the completion of a block move.
- *Real-Time Clock Interrupt*
  The Real-Time Clock interrupt is generated when the clock reaches a prespecified time that has been stored in register 32 of the register file.

EXTERNAL INTERRUPTS

Three external conditions may cause interrupts. These are:

- *External I/O Interrupts*
  The External I/O interrupt is set when an Interrupt signal is received from any of eight peripheral equipment controllers connected to any of the four possible I/O channels (there may be a total of 32 lines). The interrupt remains set until the computer directs the originating device to turn it off.

- *Manual Interrupt*
  The Manual interrupt is set by a switch on the computer console. This interrupt is not masked because it is assumed that this switch will be pressed only when an interrupt is desired.

- *Associated Computer Interrupt*
  If two computers are sharing a storage module, either computer may interrupt the other by executing a 7757xxxx instruction. This interrupt is not masked. It clears out as soon as it is recognized.

INTERRUPT MASK REGISTER

The programmer can choose to honor or ignore an interrupt by means of the Interrupt Mask register. All but two of the normal interrupt conditions are represented by the 12 Interrupt Mask register bits. The mask is selectively set with instruction 7752xxxx, and selectively cleared by instruction 7753xxxx. See Table 3-1 for mask bit assignments.

Table 3-1. Interrupt Mask Bit Assignments

Mask Bit	Conditions Represented
00-07	External Interrupts on Channel 0-3, and I/O Channel Interrupts, Channels 0-3
08	Real-Time Clock Interrupt
09	Exponent Overflow and BCD Faults
10	Arithmetic Overflow and Divide Faults
11	Search/Move Completion

As previously explained, the Manual Interrupt and the associated computer interrupt are not masked. The contents of the Interrupt Mask register may be transferred to the upper 12 bits of the A register for display purposes with instruction 772c0000 or 773c0000.

INTERRUPT CONTROL

Through use of the 3104 computer repertoire of instructions, the program can recognize, sense, and clear interrupts, and enable or disable interrupt control.

Enabling or Disabling Interrupt Control

The programmer has master control over normal interrupts. Instruction 7774---- enables the system; instruction 7773---- disables it. After recognizing an interrupt and entering the interrupt sequence, other interrupts are disabled automatically, just as if a 7773---- had been executed. When leaving the interrupt subroutine, the interrupt must again be enabled by the 7774---- instruction. After 7774----, one more instruction may be performed before the interrupt enable takes effect.

INTERRUPT PRIORITY

An order of priority exists between the various interrupt conditions. As soon as an interrupt becomes active, the computer scans the priority list until it reaches an interrupt that is active. The computer processes this interrupt and the scanner returns to the top of the list where it waits for another active interrupt to appear. Table 3-2 lists the order of priority.

Sensing Interrupts

The programmer may selectively sense interrupts, independent of the Interrupt Mask register, by using instruction 774cxxxx. Sensing the presence of internal faults automatically clears them.

Table 3-2. Interrupt Priority

Priority	Type of Interrupt
1	Arithmetic Overflow or Divide fault
2	Exponent Overflow or BCD fault
3 - 66	External I/O Interrupts*
67 - 74	I/O Channel Interrupts**
75	Search/Move Interrupt
76	Real-Time Clock Interrupt
77	Manual Interrupt
78	Adjacent Computer Interrupt

NOTES:

*There are eight interrupt lines on each of the four possible I/O channels, or 32 lines in all. On any given channel, a lower numbered line has priority over a higher numbered line. Likewise a lower numbered channel has priority over a higher numbered channel. Summarizing, line 0 of channel 0 has highest priority of all external I/O Interrupts, and line 7 of channel 3 has the lowest.

**A lower numbered I/O channel interrupt has priority over a higher numbered I/O channel interrupt.

Clearing Interrupts

I/O channel interrupts must be selectively cleared by instruction 7750xxxx. The real-time clock, arithmetic, and search/move completion interrupts may be cleared by:

- Sensing, after which the interrupts are automatically cleared.
- Using instruction 7750xxxx.
- Master clearing.

In instruction 7750xxxx, xxxx represents the mask. The manual and associated computer interrupts are automatically cleared when they are recognized.

INTERRUPT PROCESSING

Four conditions must be met before a normal interrupt can be processed:

- With the exception of the manual interrupt and adjacent computer interrupt, a bit representing the interrupt condition must be set to "1" in the Interrupt Mask register.
- The interrupt system must have been enabled.
- An interrupt-causing condition must exist.
- The interrupt scanner must reach the level of the active interrupt on the priority list.

When an active interrupt has met the above conditions, the following takes place:

- The instruction in progress proceeds until the point is reached in the RNI cycle where an interrupt can be recognized. At this time the count in P has not been advanced nor has any operation been initiated. When an interrupt is recognized, the address of the current unexecuted instruction in P is stored in address 00004.
- A number representing the interrupt-causing condition is stored in the lower 12 bits of address 00005 without modifying the upper bits. Table 3-3 lists the octal codes which are stored for each interrupt condition.
- Program control is transferred to address 00005 and an RNI cycle is executed.

Table 3-3. Representative Interrupt Codes

Conditions	Representative Codes
External interrupt	00LC*
I/O channel interrupt	010C
Real-time clock interrupt	0110
Arithmetic overflow fault	0111
Divide fault	0112
Exponent overflow fault	0113
BCD fault	0114
Search/move interrupt	0115
Manual interrupt	0116
Adjacent computer interrupt	0117

*L = line 0-7
*C = channel numbers 0-3

Special Power Failure Interrupts

Failure of primary power is detected by the computer, and a special routine is executed prior to shutdown so that no data will be lost. This operation takes 30 ms; 16 ms detection and 14 ms for processing a special power failure interrupt.

NATURE OF THE INTERRUPT

The Power Failure interrupt overrides any other interrupt (internal or external), regardless of the state of the interrupt control.

PROCESSING THE INTERRUPT

Since this interrupt overrides all others, the address in which the present contents of P are stored and the address to which the program control is transferred must be different than that for a normal interrupt. When a Power Failure interrupt occurs, the machine stores the contents of P in address 00002 and transfers program control to address 00003.

Trapped Instructions

The 3104 computer processes 3200 type BCD, floating point, and 48-bit precision multiply and divide instructions by means of implemented software. These instructions, listed in table 3-4 and in Chapter 5 are called trapped instructions.

The following operations take place when a trapped instruction is detected:

- (P + 1) is stored in address 00010
- The upper 6 bits of F are loaded into the lower 6 bits of address 00011; the upper 18 bits remain unchanged.
- Program control is transferred to address 00011 and an RNI cycle is executed.

Table 3-4. List of Trapped Instructions

Machine Code	Mnemonic Code	Instruction Function
56	MUAQ	Multiply AQ, 48-bit Precision
57	DVAQ	Divide AQ, 48-bit Precision
60	FAD	Floating Point Add
61	FSB	Floating Point Subtract
62	FMU	Floating Point Multiply
63	FDV	Floating Point Divide
64	LDE	Load E
65	STE	Store E
66	ADE	Add to E
67	SBE	Subtract from E
70	SFE	Shift E
	EZJ, EQ	E Zero Jump, E = 0
	EZJ, LT	E Zero Jump, E < 0
	EOJ	E Overflow Jump
	SET	Set D Register

Integrated Register File

The Integrated Register File is a 64 word (24 bits per word) memory located in the upper 64 addresses of storage. Although the programmer has access to all registers in the file with the 53 instruction, certain registers are reserved for specific purposes (see table 3-5). All reserved registers may be used for temporary storage if their use will not disrupt other operations that are in progress.

The contents of any register in the file may be inspected by transferring them to the A register.

Table 3-5. Integrated Register File Assignments

Register Numbers	Reserved For
00-03	Current character or word address (channel 0-3 control)
10-13	Last character or word address ± 1 depending on the instruction (channel 0-3 control)
20	Current character address (search control)
21	Source address (move control)
22	Clock, current time
23	Current character address (type control)
24	Current character address (auto-load/ dump control)
25-27	Temporary storage
30	Last character address + 1 (search control)
31	Destination address (move control)
32	Clock interrupt mask
33	Last character address + 1 (type control)
34	Last character address + 1 (auto-load/ dump control)
35-77	Temporary storage

NOTE: Register numbers correspond to upper 64 word locations in memory. Unused registers, located between register assignments are used for temporary storage.

Real-Time Clock

The real-time clock is a 24-bit counter that is incremented each millisecond and has a period of 16,777,216* milliseconds. The clock, which is controlled by a 1 kilocycle signal, starts as soon as the Run button on the console has been pushed. The current time is stored in register 22 of the register file. It is removed from storage, updated, and compared with the contents of register 32 once each millisecond. When the clock time equals the time specified by the clock mask, an interrupt is set.

When necessary, the real-time clock may be reset to any 24-bit quantity including zero by loading A, then entering (A) into register 22.

Block Operations

Block operations are of three types—Search, Move, and Input/Output. These operations use the computer block controls and, with the exception of those operations dealing with the A register, certain reserved registers in the register file. Block operations, with the exception of inputs to A and outputs from A, are buffered. After the Search/Move or I/O control has been activated, the computer can return to its main program and continue until an interrupt is generated or the program senses for block operation completion. This section presents all block operations (see table 3-6) and includes machine code instruction formats, instruction descriptions, and flow charts.

Table 3-6. Block Operations

SRCE SRCN MOVE	Search/Move instructions, character addressed and buffered	
INAC INAW	Character input Word input	Unbuffered input to, and output from A
OTAC OTAW	Character output Word output	
INPC INPW	Character input Word input	Buffered input to, and output from storage
OUTC OUTW	Character output Word output	

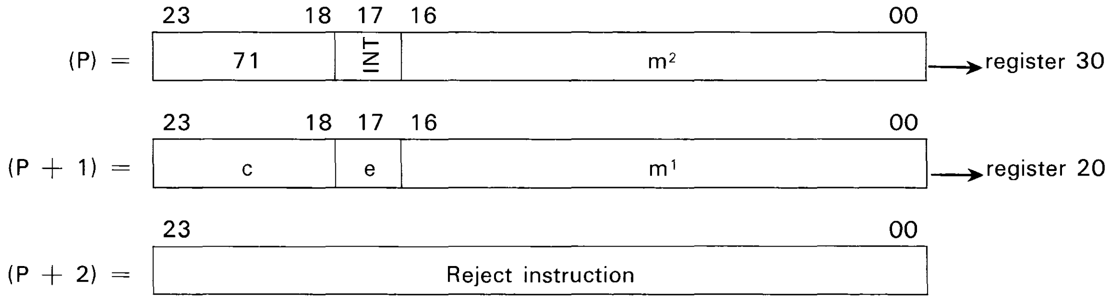
*16,777,216 milliseconds equals approximately 4 hours and 40 minutes.

SEARCH

SRCE, SRCN Search F = 71

This instruction initiates a search through a block of character storage addresses looking for equality or inequality with character ‘c’. It is composed of three words, including the two main block instruction words plus a one word reject instruction.

As a search operation progresses, m^1 is incremented until the search terminates when either a comparison occurs between the search character ‘c’ and a character in storage, or until $m^1 = m^2$. If a comparison does occur, the address of the satisfying character may be determined by inspecting m^1. To do this, transfer the contents of register 20 to A with instruction 53 (see figure 3-1).



INT = "1" for interrupt upon completion
m^2 = last character address of the search block, plus one
c = 00-77₈, BCD code of search character
e = "0" for SRCE, search for character equality
e = "1" for SRCN, search for character inequality
m^1 = first character address of the search block

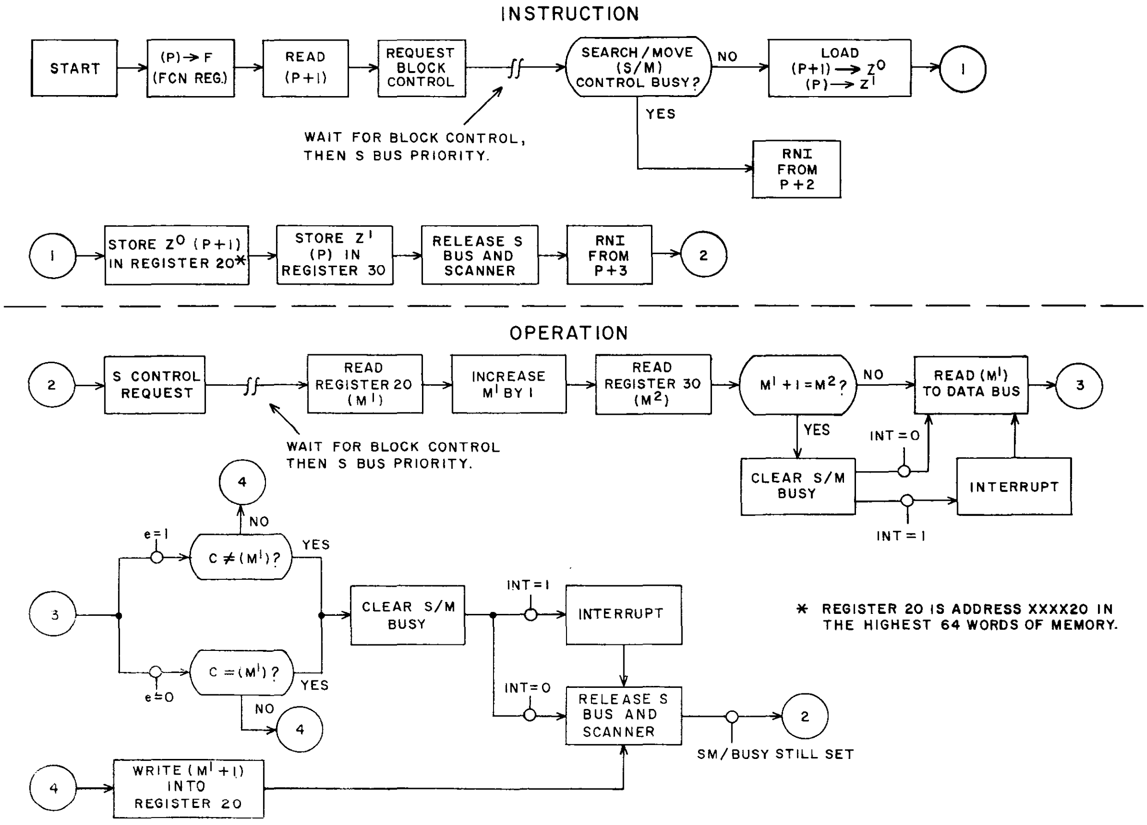


Figure 3-1. Search Operation

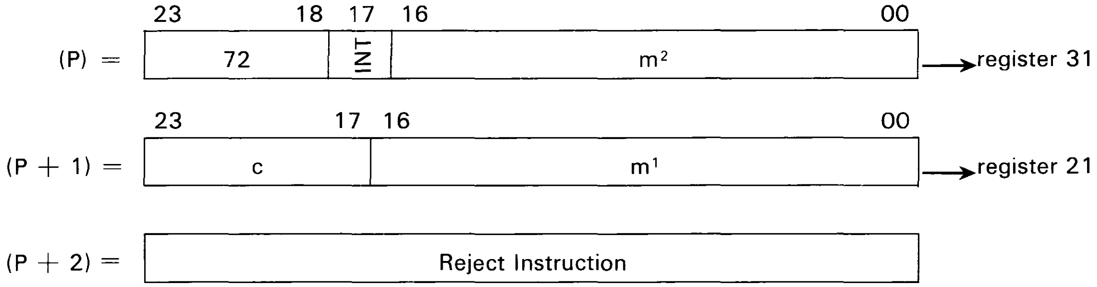
Note: Instructions 71 and 72 are mutually exclusive. Attempts to execute one while the other is in progress will cause a reject to P + 2.

MOVE

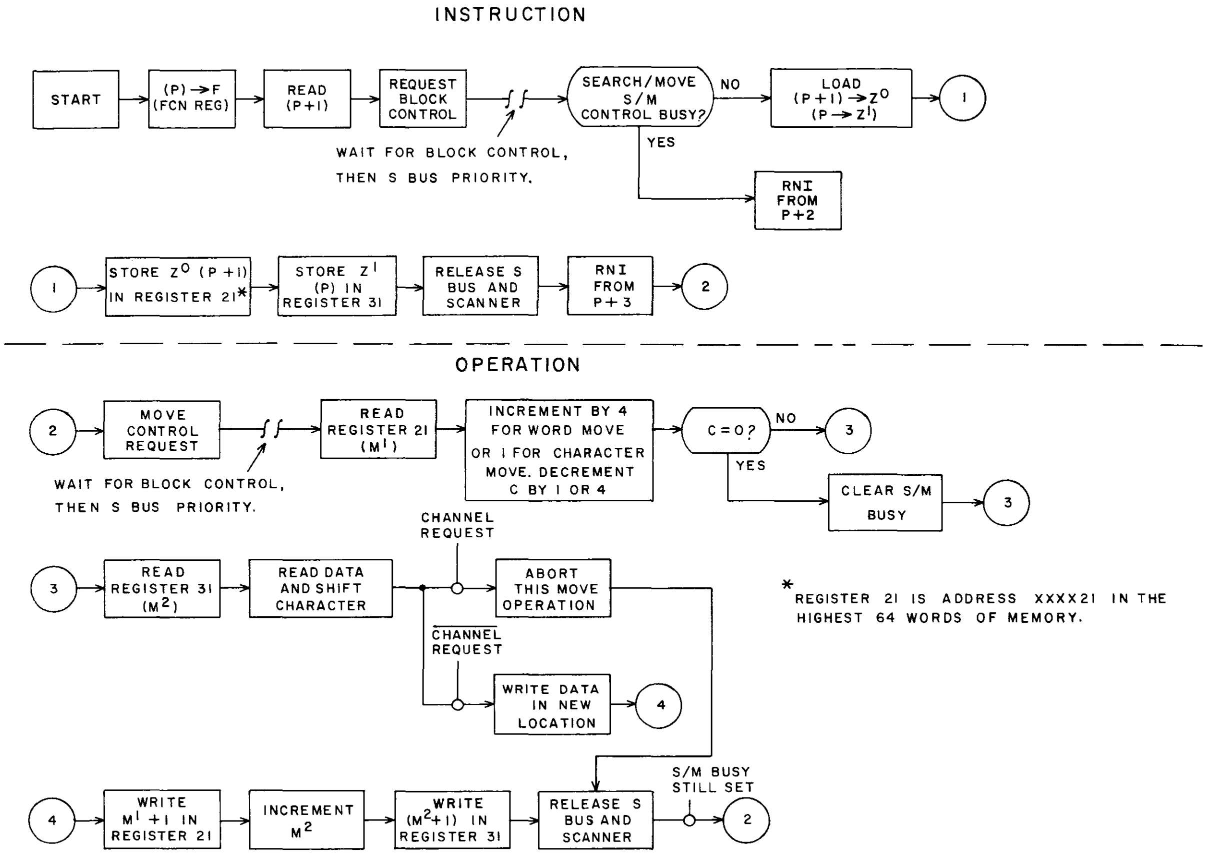
f = 72

This instruction is used to move a block of data, 'c' characters long, from one area of storage to another. It is composed of three words.

As a move operation progresses, m¹ and m² are incremented and 'c' is decremented until c = 0 (see figure 3-2). 128 characters or 32 words may be moved. When bits 00 and 01 of m¹ and m² are "0" and field length is a multiple of four characters, data is moved word by word. This reduces move time by 75% over a character by character move.



INT = "1" for interrupt
m^2 = first address of character block destination
c = field length of block, 0-177₈*
m^1 = first address of character block source



INSTRUCTION
START
(P)→F (FCN REG)
READ (P+1)
REQUEST BLOCK CONTROL
WAIT FOR BLOCK CONTROL, THEN S BUS PRIORITY.
SEARCH/MOVE S/M CONTROL BUSY?
NO
YES
LOAD (P+1)→Z⁰ (P→Z¹)
1
RNI FROM P+2
1
STORE Z⁰ (P+1) IN REGISTER 21*
STORE Z¹ (P) IN REGISTER 31
RELEASE S BUS AND SCANNER
RNI FROM P+3
2
OPERATION
2
MOVE CONTROL REQUEST
WAIT FOR BLOCK CONTROL, THEN S BUS PRIORITY.
READ REGISTER 21 (M¹)
INCREMENT BY 4 FOR WORD MOVE OR 1 FOR CHARACTER MOVE. DECREMENT C BY 1 OR 4
C=0?
NO
3
YES
CLEAR S/M BUSY
3
CHANNEL REQUEST
3
READ REGISTER 31 (M²)
READ DATA AND SHIFT CHARACTER
ABORT THIS MOVE OPERATION
CHANNEL REQUEST
WRITE DATA IN NEW LOCATION
4
*REGISTER 21 IS ADDRESS XXXX21 IN THE HIGHEST 64 WORDS OF MEMORY.
S/M BUSY STILL SET
4
WRITE M¹+1 IN REGISTER 21
INCREMENT M²
WRITE (M²+1) IN REGISTER 31
RELEASE S BUS AND SCANNER
2

Figure 3-2. Move Operation

*1-177₈ represents a field length of 1 to 127 characters; 0 represents a field length of 128 characters.

INPUT/OUTPUT

Instructions 73 through 76 enable the computer to communicate with peripheral equipment via the I/O channels. These instructions are of two distinct types: those that deal with the A register and those that deal with storage. They all begin with the series of steps shown in figure 3-3. See the 77 instruction in chapter 5 for details on the preliminary operations—connecting to I/O equipment (77.0), sensing status of I/O equipment (77.2), and selecting function of I/O equipment (77.1).

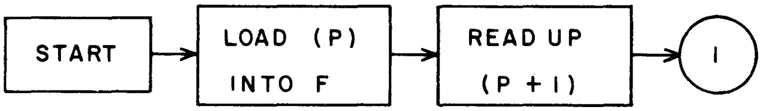


Figure 3-3. Initial Steps of I/O Sequence

Operations with A

Operations with A are unbuffered. They have a common machine code format.

INAC Input, Character to A f = 73

A 6-bit character is read from a peripheral device and loaded into the lower 6 bits of A. A is cleared previous to the input and the upper 18 bits remain cleared (see figure 3-4).

INAW Input, Word to A f = 74

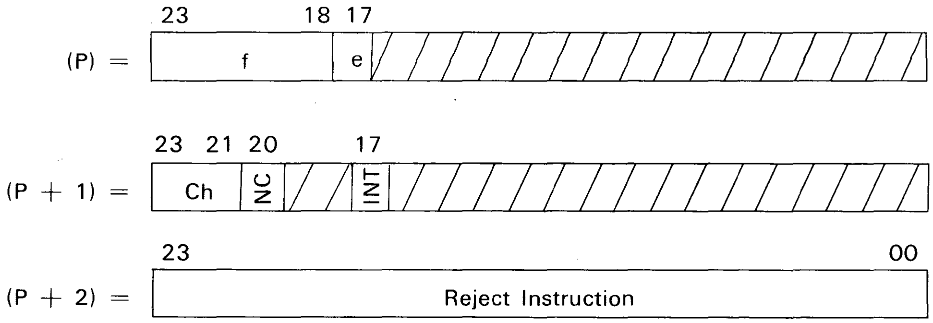
A 12 or 24-bit word is read from a peripheral device and loaded into the lower 12 bits or into all of A. Word size depends upon the type of I/O channel in use. A is cleared previous to an input and in the case of a 12-bit input, the upper 12 bits remain cleared (see figure 3-4).

OTAC Output, Character from A f = 75

A character from the lower 6 bits of A is sent to a peripheral device (see figure 3-5). (A) is retained.

OTAW Output, Word from A f = 76

A word from the lower 12 bits or from all of A is sent to a peripheral device. Word size depends upon the type of I/O channel in use (see figure 3-5). (A) is retained.



f = operation code 73-76
INT = "1" for interrupt on completion
Ch = I/O channel x; where x = 0-3
NC = "1" for no BCD conversion
e = "1" for operations with A

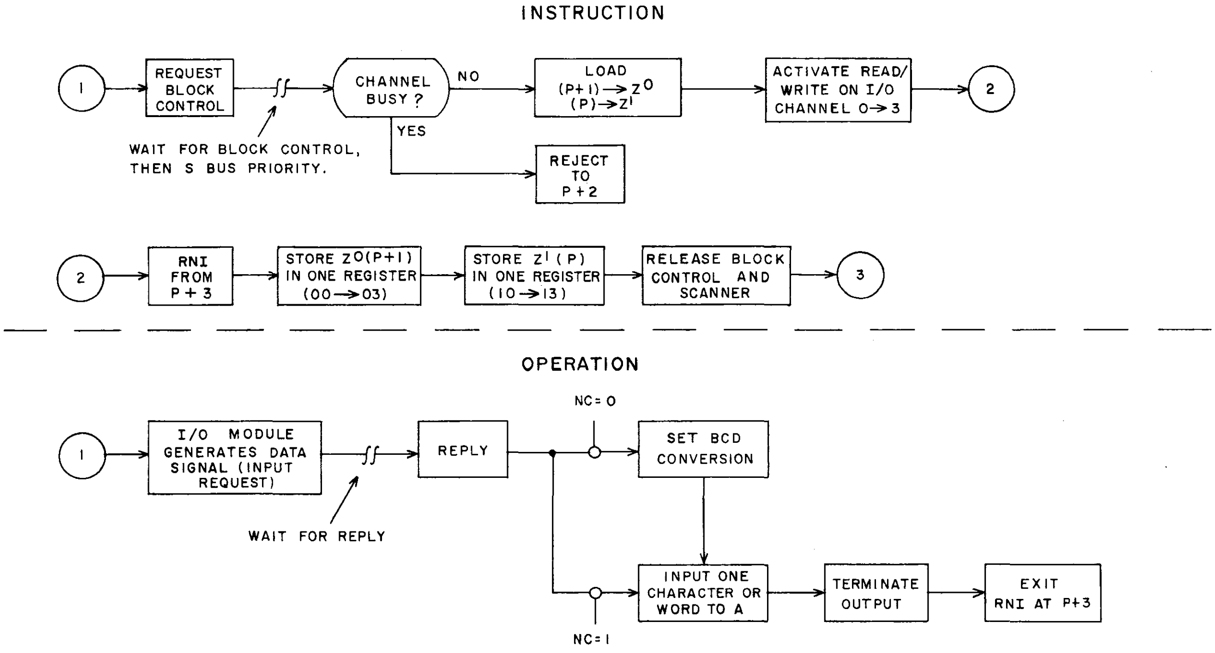

INSTRUCTION
1
REQUEST BLOCK CONTROL
WAIT FOR BLOCK CONTROL, THEN S BUS PRIORITY.
CHANNEL BUSY ?
NO
YES
LOAD (P+1)→Z0 (P)→Z1
REJECT TO P+2
ACTIVATE READ/WRITE ON I/O CHANNEL 0→3
2
2
RNI FROM P+3
STORE Z0(P+1) IN ONE REGISTER (00→03)
STORE Z1 (P) IN ONE REGISTER (10→13)
RELEASE BLOCK CONTROL AND SCANNER
3
OPERATION
1
I/O MODULE GENERATES DATA SIGNAL (INPUT REQUEST)
WAIT FOR REPLY
REPLY
NC=0
SET BCD CONVERSION
INPUT ONE CHARACTER OR WORD TO A
NC=1
TERMINATE OUTPUT
EXIT RNI AT P+3

Figure 3-4. Input, Character or Word to A

INSTRUCTION

SAME INSTRUCTION FORMAT AS INPUT, CHARACTER OR WORD TO A (FIGURE 3-4)

OPERATION

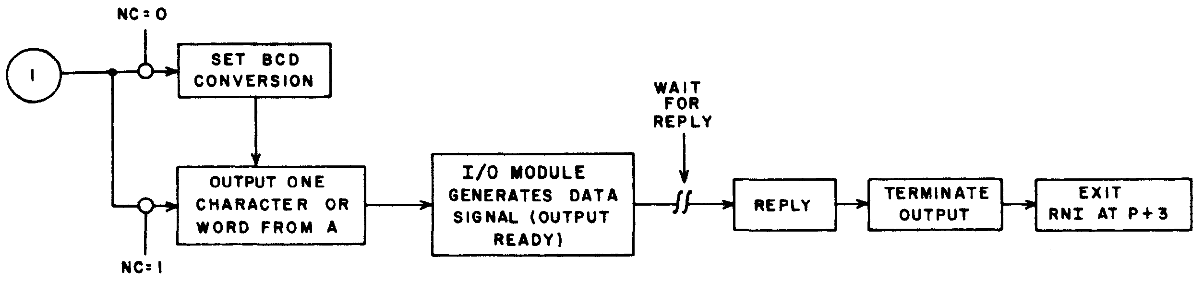

NC=0
1
SET BCD CONVERSION
OUTPUT ONE CHARACTER OR WORD FROM A
NC=1
I/O MODULE GENERATES DATA SIGNAL (OUTPUT READY)
WAIT FOR REPLY
REPLY
TERMINATE OUTPUT
EXIT RNI AT P+3

Figure 3-5. Output, Character or Word from A

Operations with Storage

These operations are buffered. Main computer control relinquishes control of the I/O operations and returns to the main program as soon as Read or Write has been activated. They have a common machine code format.

During the execution of word addressed I/O instructions, the addresses m^1 and m^2 are shifted left two places to the upper 15 bits of the 17-bit address positions. From this time on, they are treated as character addresses.

Registers 00-17$_8$ of the register file are reserved for buffered I/O operations; the last octal digit of the register designator corresponds to I/O channel x through which data is being transferred. 00-07 hold the current character or word address, and 10-17$_8$ hold the last character or word address, ± 1, depending on the operation.

INPC Input, Character Block to Storage f= 73

This is a character addressed instruction; 6 or 12-bit characters are read from peripheral equipment and stored in memory (refer to figure 3-6).

If H=0, there is 6 to 24-bit assembly. If H=1, there is 12 to 24-bit assembly. During this 12 to 24-bit assembly, the lowest bit of each character address is not read. This ensures that assembled characters are in either the upper or the lower half of a storage word.

M^2 = last character address of input data block, plus one (minus one, for backward storage).

INPW Input, Word Block to Storage f = 74

This is a word addressed instruction with the addresses initially placed in the lower 15 bits of the instruction words.

Depending upon the I/O module capability, 12 or 24-bit words are read from a peripheral device and stored in memory (refer to figure 3-7).

If N = 0, there is 12 to 24-bit assembly. The first word of a block is stored in the upper half of a storage address for store forward and in the lower half for store backward.

If N = 1, there is no assembly; a straight 12 or 24-bit data transfer occurs. A 12-bit word will be stored in the lower half of a storage address.

M^2 = last word address of input data block, plus one (minus one, for backward storage).

OUTC Output, Character Block from Storage f = 75

This is a character addressed instruction. Storage words are disassembled into 6 or 12-bit characters and sent to a peripheral device (refer to figure 3-8).

If H = 0, there is 24 to 6-bit disassembly. If H = 1, there is 24 to 12-bit disassembly.

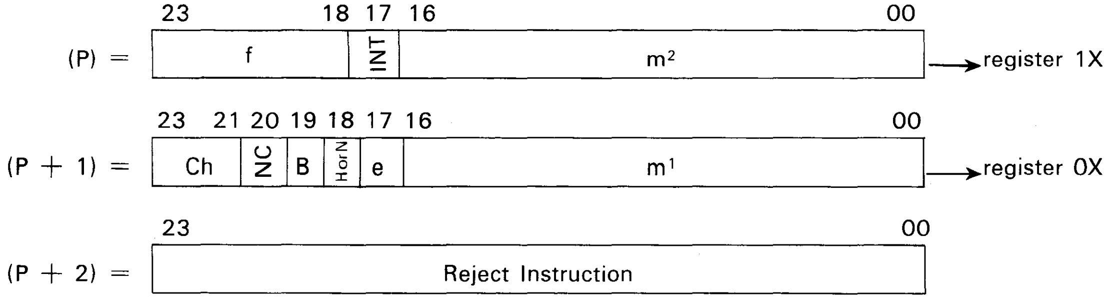
M^2 = last character address of output data block, plus one (minus one for load backward).

OUTW Output, Word Block from Storage f = 76

This is a word addressed instruction with the addresses initially placed in the lower 15 bits of the instruction words. Words are read from storage and sent to a peripheral device (refer to figure 3-9).

If N = 0, there is 24 to 12-bit disassembly. If N = 1, there is a straight 12 or 24-bit data transfer depending on the I/O module capabilities. If an attempt is made to send 24 bits over a 12-bit I/O channel, the upper 12 bits will be lost.

M^2 = last word address of output data block, plus one (minus one for load backward).

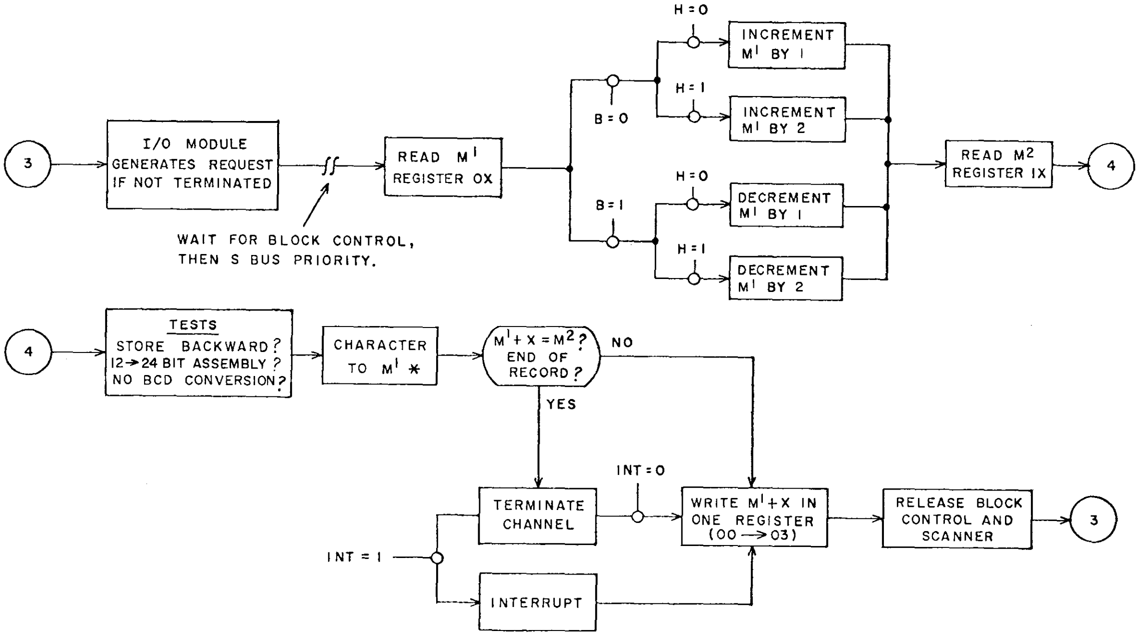


f = operation code 73-76
INT = "1" for interrupt
m^2 = (see individual instruction)
Ch = I/O channel X; where X = 0-3
NC = "1" for no BCD conversion
B = "1" for store backward
H or N = (see individual instruction)
e = "0" for operations with storage
m^1 = first character or word address of I/O data block; becomes current address as I/O operation is carried out.

INSTRUCTION

SAME INSTRUCTION FORMAT AS INPUT, CHARACTER OR WORD TO A (FIGURE 3-4)

OPERATION



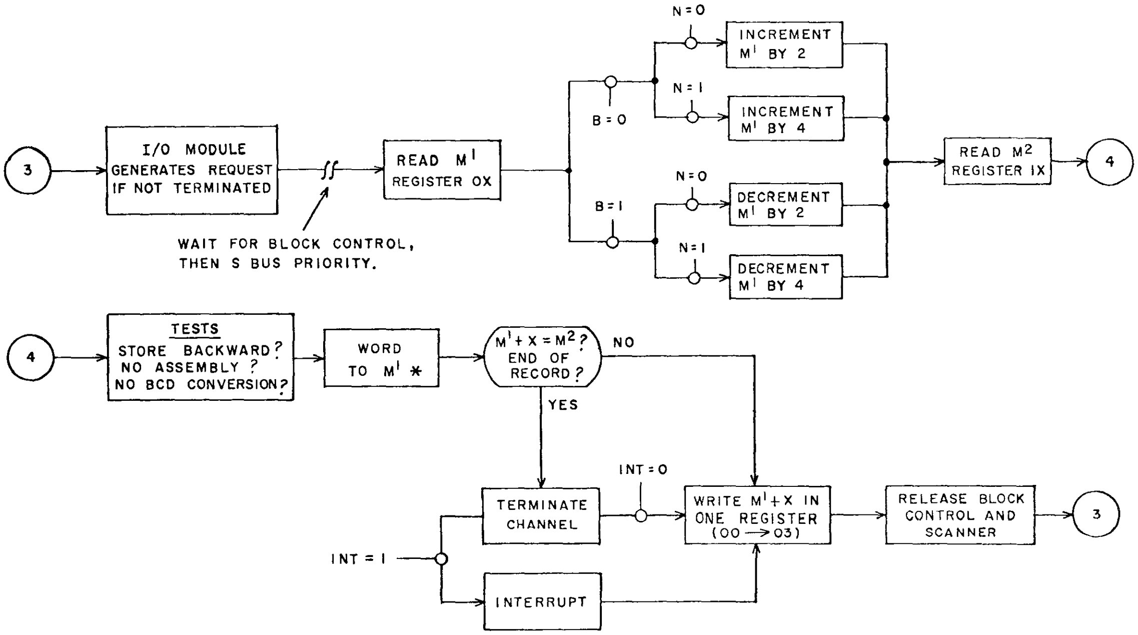
* BCD CONVERSION IF NC=0

Figure 3-6. Input, Character Block to Storage

INSTRUCTION

SAME INSTRUCTION FORMAT AS INPUT, CHARACTER OR WORD TO A (FIGURE 3-4)

OPERATION



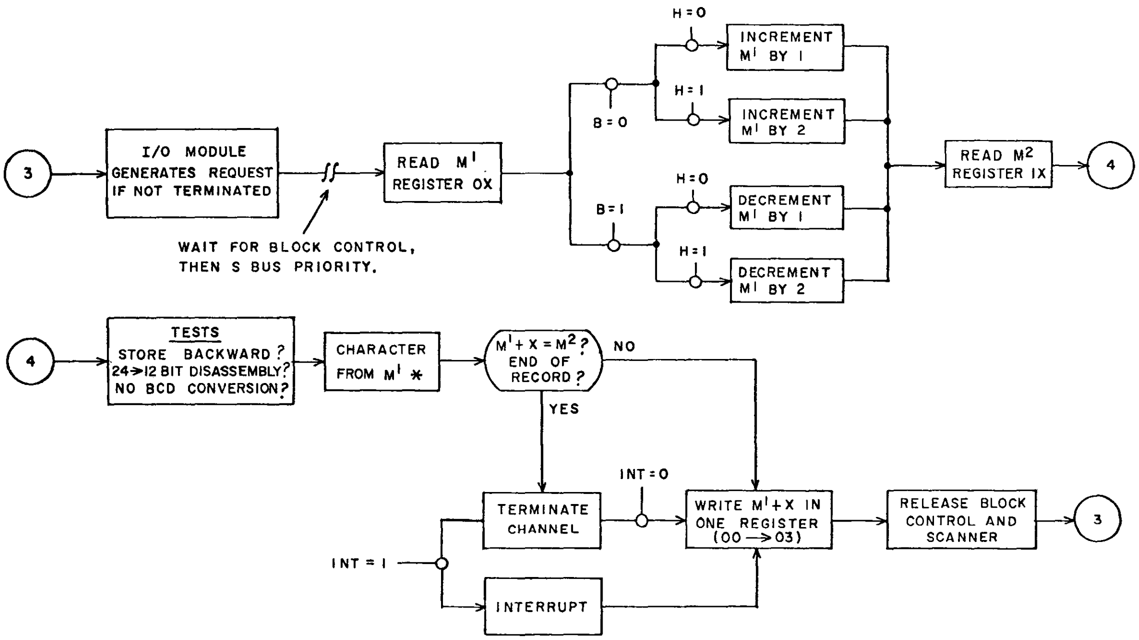
* BCD CONVERSION IF NC=0

Figure 3-7. Input, Word Block to Storage

INSTRUCTION

SAME INSTRUCTION FORMAT AS INPUT, CHARACTER OR WORD TO A (FIGURE 3-4)

OPERATION



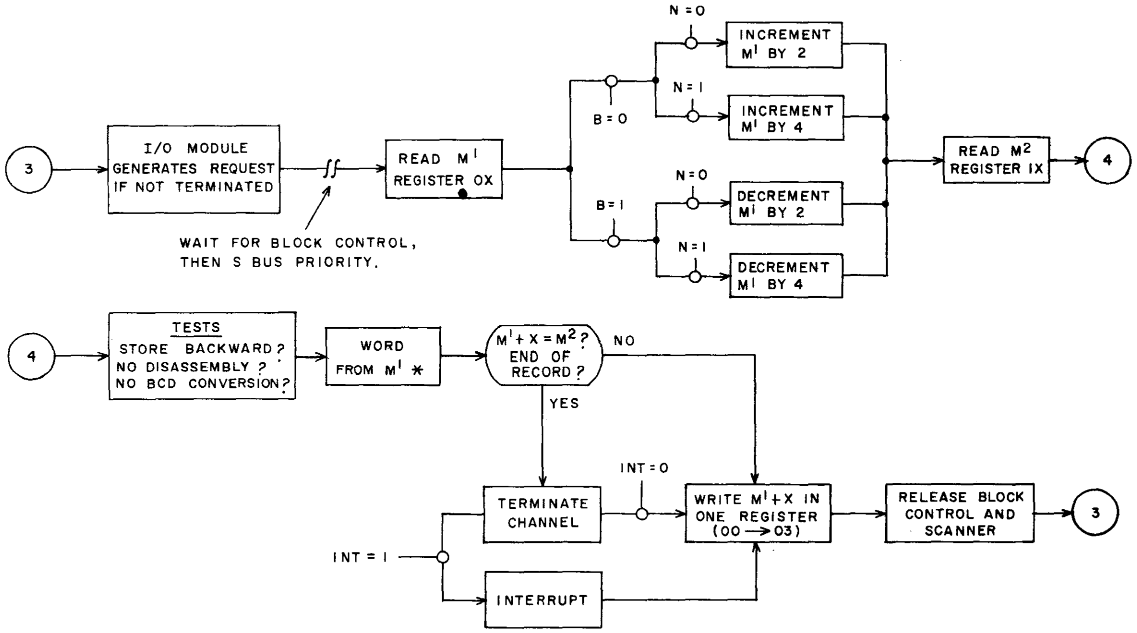
* BCD CONVERSION IF NC=0

Figure 3-8. Output, Character Block from Storage

INSTRUCTION

SAME INSTRUCTION FORMAT AS INPUT, CHARACTER OR WORD TO A (FIGURE 3-4)

OPERATION



* BCD CONVERSION IF NC=0

Figure 3-9. Output, Word Block from Storage

4

Operating Features

Two consoles, functionally identical to each other, are available for the 3100 computer system — the standard Integrated Console or the Optional 3101 Desk Console. This chapter defines the switches and indicators used on these consoles as well as explains the use of the entry keyboard. The basic differences in these consoles lie in their physical structures; they are electrically and logically identical.

Displays and Indicators

Seven rows of indicator lights are used to display the operational registers of the 3104 on the integrated console. Status lights, manual controls and a keyboard are also provided. Figure 4-1 is a view of the integrated console and table 4-1 describes the register displays. The 3101 desk console is electrically and logically identical to the integrated console; however the displays and switches are located above the on-line monitor typewriter.

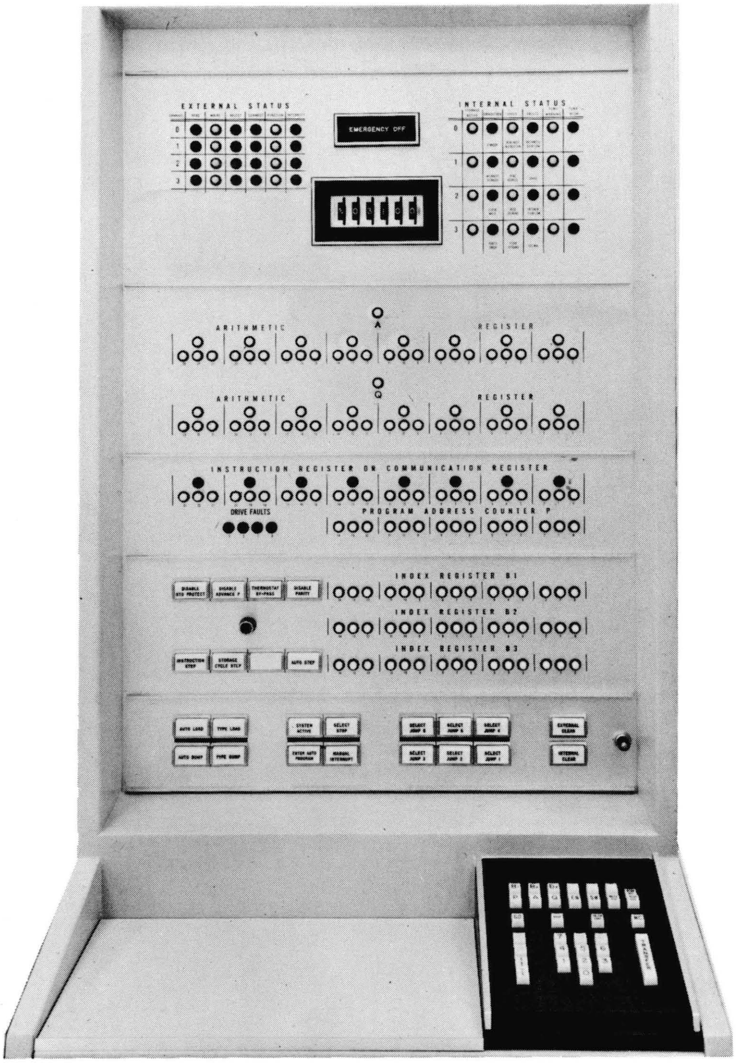
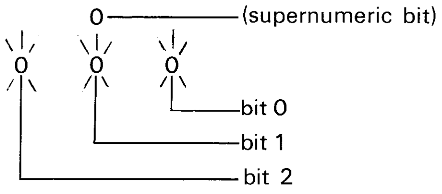


Figure 4-1. Integrated Console

Table 4-1. Register Displays

Register	Binary Capacity	Description
Program Address Counter	15 bits	Program Address register display panel.
Indexes B^1-B^3	15 bits	Index register display panels.
Instruction register or Communications register	24 bits	1) When one of the Step modes of operation is used, the contents of the Instruction register are shown. 2) In Stop mode, when the keyboard is active, the contents of the Communication register are shown. 3) In Run mode, when the keyboard is active, the contents of the Communication register are shown.
A and Q registers	48 bits	Displays the contents of each register.

On the integrated console, three indicator lights represent each digit. The digit configuration is as follows:



EXTERNAL STATUS INDICATORS

The external status indicators display the existing condition of I/O channels 0-3. Conditions displayed are Read, Write, Reject, Connect, Function, and Interrupt.

INTERNAL STATUS INDICATORS

Six columns of internal condition indicators are mounted on the consoles.

1 Storage Active
For addressing purposes, all possible word sections of memory are designated by digits 0-3. Digit zero indicates 4K or 8K storage. Digits 0 and 1 indicate 16K storage and 0 to 3 32 K storage. Whenever one of these storage sections becomes active, the corresponding indicator light is lit.

2 Conditions
Standby means that the main power switch is on, but individual supplies are still off.
Interrupt Disabled is lit whenever interrupt is disabled by the 77 instruction.

3 Cycle
Four cycles are represented: Read Next Instruction, Read Address, Read Operand, and Store Operand. These indicators are lit whenever the cycles are in progress.

4 Faults
This column represents the four arithmetic faults: Arithmetic Overflow, Divide, Exponent Overflow, and Decimal (BCD—always occurs when a BCD instruction is executed).

5 Temp Warning, and

6 Temp High

Looking at the front of a fully expanded 3104 computer, the cabinet sections are designated by digits 0-3 (see figure 4-2). The Temp Warning lights indicate that the section in question is approaching the upper limit of the normal operating temperature range of the computer. This is only a warning; the computer is not disabled. The Temp High indicators light when the safe operating temperature is exceeded in the sections they represent. At the same time, the power will be cut off unless the Thermostat Bypass switch has been pressed.

Temperature Indicator 2	Temperature Indicator 1	Temperature Indicator 0	Temperature Indicator 3	
8K Memory, and I/O Logic	Block Control, Interrupt, 4K Memory and I/O Logic.	Main Control and Arithmetic Logic	16K Memory and I/O Logic	Power Panel

Figure 4-2. Temperature Warning Designations for Fully Expanded 3104, Front View

Switches

The console switches are divided into two groups —those used for normal operation of the computer and those used primarily for maintenance purposes.

OPERATOR SWITCHES

Operational switches are found on the main console and the entry keyboard.

Main Console: Table 4-2 lists and describes the main console operator switches.

Entry Keyboard: The entry keyboard at the console replaces the Set and Clear push buttons that are on most CONTROL DATA computers for the manual entry of information. Figure 4-3 shows the 3104 console keyboard. Table 4-3 lists and describes the keyboard switches.

Table 4-2. Main Console Operator Switches

Switch Name	Quan.	Illum.	Description
Emergency Off (momentary)	1		Removes power from the whole system.
Breakpoint Address Selector Run Mode Selector	1		Lefthand dial of the six section, eight position switch. Permits the selection of two modes: 1) Breakpoint Mode a) OFF b) Instruction Address c) OFF d) Operand Address 2) Run Mode e) OFF f) Register Number* g) OFF h) Storage Address *Registers 00000-00077 only.
Breakpoint Address, Register File Number, or Storage Address	5		Five eight position thumb-wheel switches can be set to octal addresses 00000-77777 for modes 1 or 2 above.
Auto Load (momentary)	1	yes	Provides for the automatic loading of storage from a designated device. Active whether machine is running or stopped.
Auto Dump (momentary)	1	yes	Provides for the automatic dumping of storage into a designated device. Active whether machine is running or stopped.
Type Load (momentary)	1	yes	Provides for the loading of storage from the on-line I/O typewriter. Active whether machine is running or stopped.
Type Dump (momentary)	1	yes	Provides for the dumping of storage into the on-line I/O typewriter. Active whether machine is running or stopped.
Select Stop 1	1	yes	Stops the computer when the Selective Stop instruction is read.
Manual Interrupt (momentary)	1	yes	Forces the computer into an interrupt routine if the computer is in Run. If the computer is stopped when the switch is pressed, it will go into an interrupt routine as soon as it is restarted.
Select Jump 1-6	6	yes	Provides the manual conditions for executing a program jump on the Selective Jump instruction.
External Clear (momentary)	1	yes	Master clears all external equipments, the I/O channels to which they are attached, and all controls in the data channels.
Internal Clear (momentary)	1	yes	Master clears internal conditions and registers.

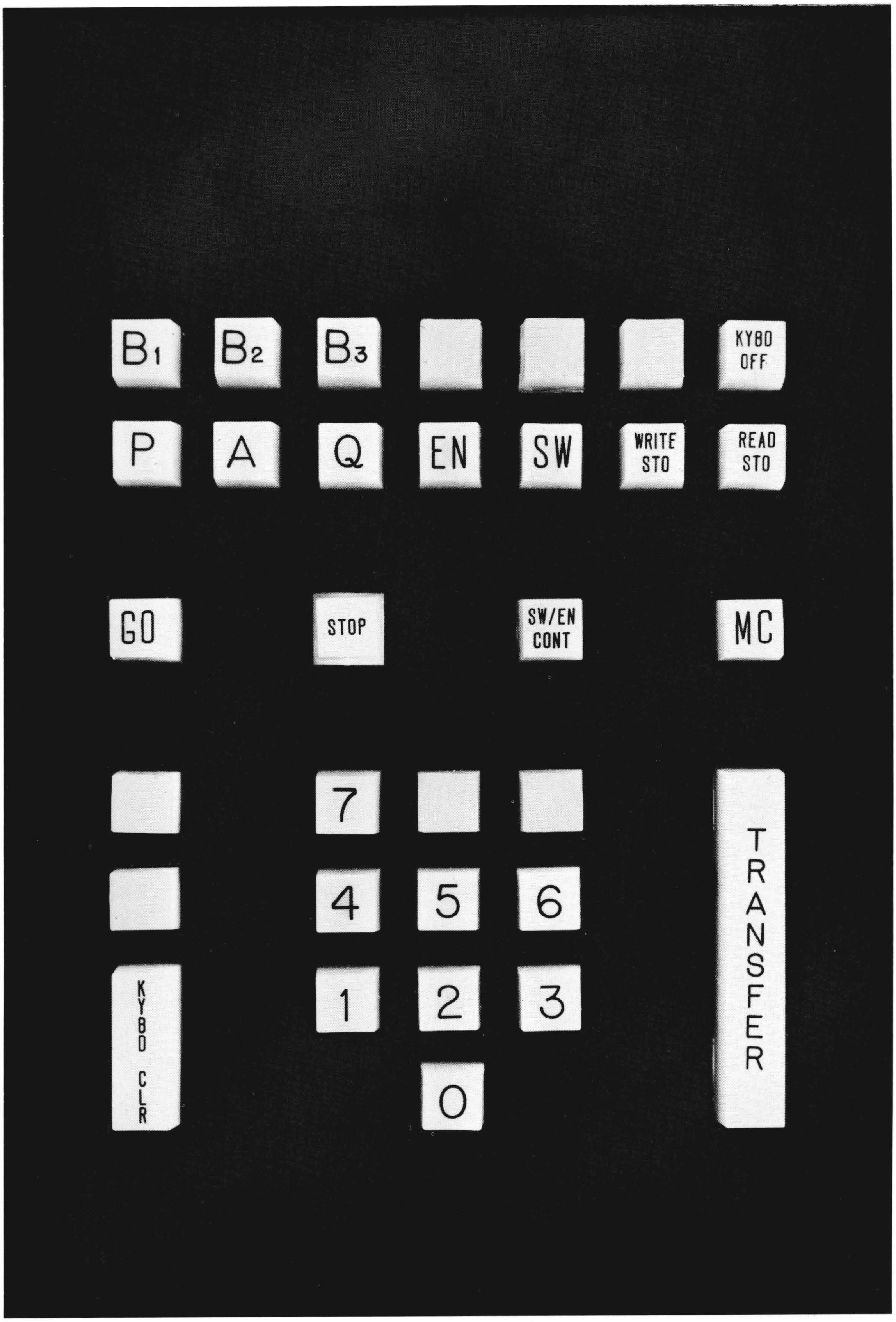


Figure 4-3. 3104 Console Keyboard

Table 4-3. Keyboard Switches

Operational Control Switches

Switch Name	Illum.	Description
Keyboard Off	Yes	Deactivates all keyboard controls. Disables Keyboard Active indicator.
Keyboard Clear (momentary)		Clears the Communications register and keyboard control settings.
Go (momentary)	Yes	Starts the computer at address to which P register has been set. Indicator is lit while computer is executing instructions. Not used for Sweep or Enter.
SW/EN Go (momentary)	Yes	Enables sweep or enter operations to proceed through storage.
Stop (momentary)	Yes	Brings the computer to a halt at the end of the current instruction. Indicator is lit when computer is forced to a Halt or Stop.
Transfer (momentary)		Enables the transfer of data between the Communications register and a selected register or storage location.
MC (momentary)		Performs both an internal and external master clear. Disabled when computer is in Go mode.

Register Selection Switches

Switch Name	Illum.	Description
B^1-B^3	Yes	Enables the manual entry of data from the keyboard into index registers B^1-B^3.
P	Yes	Enables the manual entry of an address from the keyboard into the P register.
A	Yes	Causes both A and Q to be displayed, but enables entry only into A.
Q	Yes	Causes both A and Q to be displayed, but enables entry only into Q.

Mode Selector Switches

Switch Name	Illum.	Description
Enter *	Yes	Enables the manual entry of information into storage while machine is stopped. First address of sequence is first entered into P. Pushing Transfer advances P.
Sweep *	Yes	Enables instructions to be read from consecutive storage locations; they are not executed. First address of sequence is first entered into P. Pushing Transfer advances P.
Write Storage	Yes	Enables keyboard entry into the storage location specified by the thumb-wheel switches. Entry occurs each time the Transfer key is pressed whether the computer is running or stopped.
Read Storage	Yes	Causes the display of the contents of the storage register location specified by the thumb-wheel switches. The word is displayed when the Transfer key is pressed whether the computer is running or stopped.

Table 4-3. Keyboard Switches (cont'd)

Digit and Sign Selector Switches		
Switch Name	**Illum.**	**Description**
0-7 (momentary)		All of these buttons, when pressed one at a time, allow entry of that particular digit into the Communications register.

* All register selection switches are disabled when either the Enter or Sweep switch is depressed.

MAINTENANCE SWITCHES

Maintenance switches are all located on the main console. Table 4-4 lists and describes the maintenance switches.

Table 4-4. Maintenance Switches

Switch Name	Illum.	Description
Disable Storage Protect	Yes	Disables the circuitry that normally protects the contents of storage.
Disable Advance P	Yes	Disables advancement of the count in the P register. When the Go button on the keyboard is pressed, the same instruction is repeated. Press a second time to release function.
Thermostat Bypass	Yes	Allows computation to proceed regardless of unfavorable ambient temperatures.
Disable Parity	Yes	Disables the recognition of parity errors from all storage modules.
Instruction Step	Yes	Enables the operator to step through the program instruction by instruction.
Storage Cycle Step	Yes	Enables the operator to step through an instruction one storage cycle at a time.
Auto Step	Yes	Enables many instructions to be executed in a low speed Run mode. The speed is regulated by the Auto Step Speed control on the console.

5

Repertoire of Instructions

General Information

INSTRUCTION WORD FORMATS

Instructions 00-70 and 77 use one 24-bit word each; instructions 71-76 use two 24-bit words. In general, the upper 6 bits hold the identifying function code 'f'. Instruction formats are of two types —word and character. Figure 5-1 shows the general formats for word and character oriented instructions. Instructions 70-77 use several additional symbols that are defined when they occur.

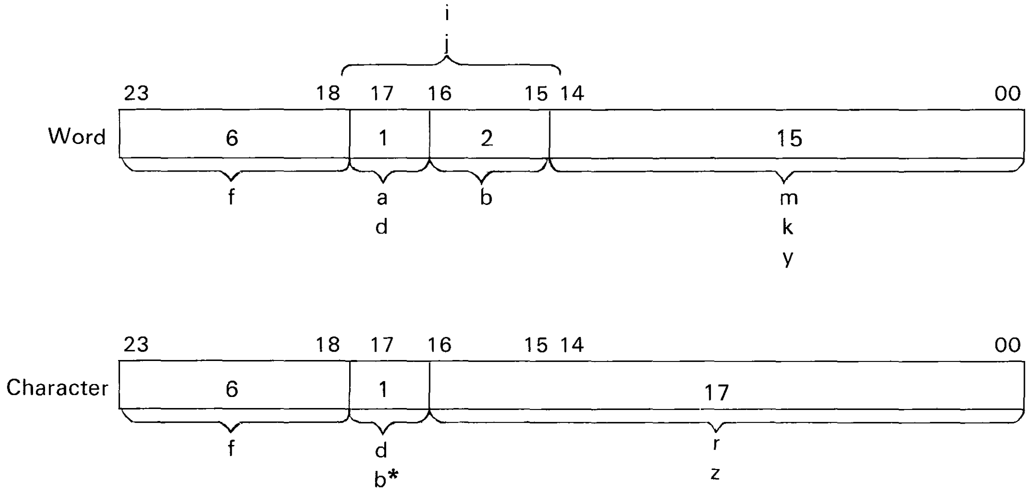


Figure 5-1. General Machine Code Instruction Formats

SYMBOL DEFINITIONS

a = addressing mode designator (a = 0, direct addressing; a = 1, indirect addressing)

b = index designator (unless otherwise stated)

d = operation designator (see individual instructions)

f = function code (6 bits, octal 00 to 77)

i = interval designator

j = jump, stop, or skip condition designator (see individual instructions)

k = shift count
m = word execution address } unmodified
r = character execution address

y = 15-bit operand

z = 17-bit operand

In some instructions, the execution address 'm' or 'r', or the shift count 'k' may be modified by adding to them the contents of an index register, B^b. The 2-bit designator 'b' specifies which of the three index registers is to be used. Symbols representing the respective modified quantities are M, R and K.

$M = m + (B^b)$
$R = r + (B^b)$
$K = k + (B^b)$

In each case, if b = 0, then M = m, R = r and K = k.

ADDRESSING MODES

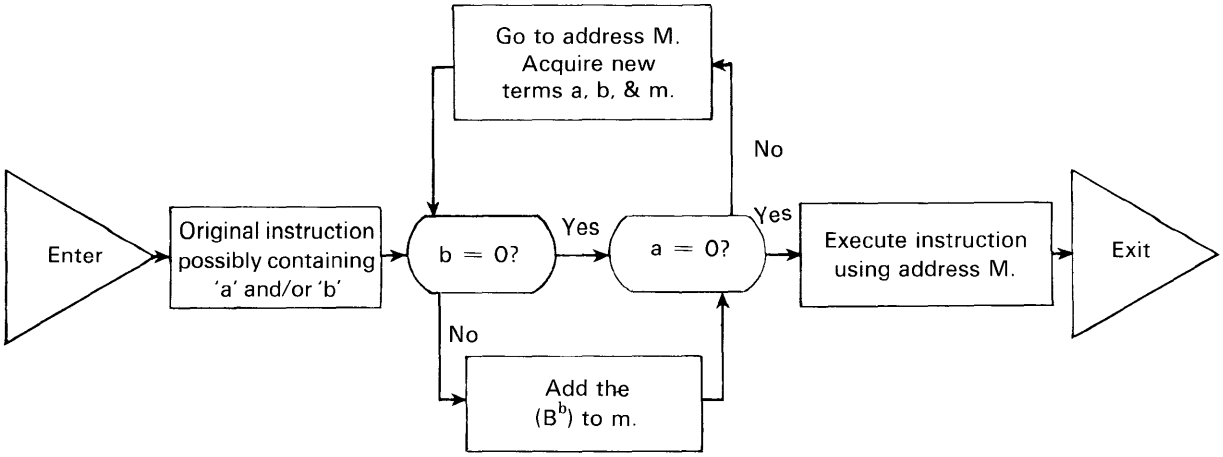
Three modes of addressing are used in the 3100 computer: no address, direct address, and indirect address.

No Address. This mode is used when an operand 'y' or a shift count 'k' is placed directly into the lower portion of an instruction word. Symbols 'a' and 'b' are not used as addressing mode and index designators with any of the no address instructions.

Direct Address. A direct address instruction is any instruction in which an operand address 'm' is stored in the lower portion of the initial instruction word. This mode is specified by making 'a' equal to zero. In many instructions, address 'm' may be modified (indexed) by adding to it the contents of register B^b; $M = m + (B^b)$.

*When used in this position, 'b' calls for the use of a specific index register.

Indirect Address. It is possible to use indirect addressing only with instructions that require an execution address 'm'. For applicable instructions, indirect addressing is specified by making 'a' equal to one. Several levels (or steps) of indirect addressing may be used to reach the execution address; however, execution time is delayed in direct proportion to the number of steps. The search for a final execution address continues until 'a' equals zero. It is important to note that direct (or indirect) addressing and address modification are two distinct and independent steps. In any particular instruction, one may be specified without the other. Figure 5-2 shows the indirect addressing routine for a 3100 computer.


Go to address M. Acquire new terms a, b, & m.
No
Enter
Original instruction possibly containing 'a' and/or 'b'
b = 0?
Yes
a = 0?
Yes
Execute instruction using address M.
Exit
No
Add the (B^b) to m.

Note: Unless it is otherwise stated, indirect addressing follows the above routine throughout the repertoire of instructions.

Figure 5-2. Indirect Addressing Routine

READ NEXT INSTRUCTION SEQUENCE (RNI)

An abbreviation, RNI, is used throughout the repertoire of instructions to indicate the read next instruction sequence. This is a sequence of steps taken by the control section to advance the computer to its next program step. For an extensive description of this sequence, consult the 3100 Customer Engineering Manual.

INDEX OF INSTRUCTIONS

In this chapter the instructions are grouped and arranged in the following order. Those marked with an asterisk are trapped instructions.

Each group of instructions is introduced with an index as well as a group description whenever it is necessary. Individual instructions are all presented in the same basic format:

Heading, which includes the assembly language mnemonic† and instruction name.

Machine code instruction format

Instruction description

Comments (when necessary)

Approximate instruction execution time. Add 1.75 μsec for indirect addressing. Instructions shown without execution times are indeterminate at this time.

STOP AND JUMPS	00-03
REGISTER OPERATIONS WITHOUT STORAGE REFERENCE	04-05, 10-17
STORAGE TEST	06-07, 10, 52
LOGICAL INSTRUCTIONS WITH STORAGE REFERENCE	35-37
LOAD	20-27, 54
STORE	40-47
INTER-REGISTER TRANSFER	
24-bit Precision	53
ARITHMETIC, FIXED POINT, 24-BIT PRECISION	30-31, 34, 50-51
ARITHMETIC, FIXED POINT, 48-BIT PRECISION	32-33, *56-57
ARITHMETIC, FLOATING POINT	*60-63
BCD OPERATIONS	*64-70
BLOCK OPERATIONS	71-76
MISCELLANEOUS OPERATIONS	77

†Some assembly code mnemonics may be modified by one or more of the following codes:

EQ	Equal
GE	Greater than or equal
I	Indirect addressing
LT	Less than
NE	Not equal
S	Extend sign of operand; use full 24-bit register in skip instructions
B	Backward read or write
H	Half assembly or disassembly (12-24)
INT	Interrupt when completed
N	No assembly or disassembly (24-24)
NC	No internal BCD conversion

Instructions

STOP AND JUMPS

Operation Field		Address Field	Interpretation
00	HLT	m	Unconditional stop; RNI from address 'm'
	SJ1	m	Jump if key 1 is set
	SJ2	m	Jump if key 2 is set
	SJ3	m	Jump if key 3 is set
	SJ4	m	Jump if key 4 is set
	SJ5	m	Jump if key 5 is set
	SJ6	m	Jump if key 6 is set
	RTJ	m	Return jump
01	UJP, I	m, b	Unconditional jump
02	IJI	m, b	Index jump; increment index
	IJD	m, b	Index jump; decrement index
03	AZJ, EQ	m	Compare A with zero; jump if (A) = 0
	NE		Compare A with zero; jump if (A) ≠ 0
	GE		Compare A with zero; jump if (A) ≥ 0
	LT		Compare A with zero; jump if (A) < 0
	AQJ, EQ	m	Compare A with Q; jump if (A) = (Q)
	NE		Compare A with Q; jump if (A) ≠ (Q)
	GE		Compare A with Q; jump if (A) ≥ (Q)
	LT		Compare A with Q; jump if (A) < (Q)

NOTE: Two additional Jump instructions, EZJ and EQJ, are described under the BCD instructions.

A Jump instruction causes a current program sequence to terminate and initiates a new sequence at a different location in storage. The Program Address register, P*, provides the continuity between program steps and always contains the storage location of the current program step.

When a Jump instruction occurs, P is cleared and a new address is entered. In most jump instructions, the execution address 'm' specifies the beginning address of the new program sequence. The word at address 'm' is read from storage, placed in F, and the first instruction of the new sequence is executed.

Some of the Jump instructions are conditional upon a register containing a specific value or upon the position of the Jump key on the console. If the criterion is satisfied, the jump is made to location 'm'. If it is not satisfied, the program proceeds in its regular sequence to the next instruction.

HLT Halt

	23 18	17 15	14 00
Format:	0 0	0	m

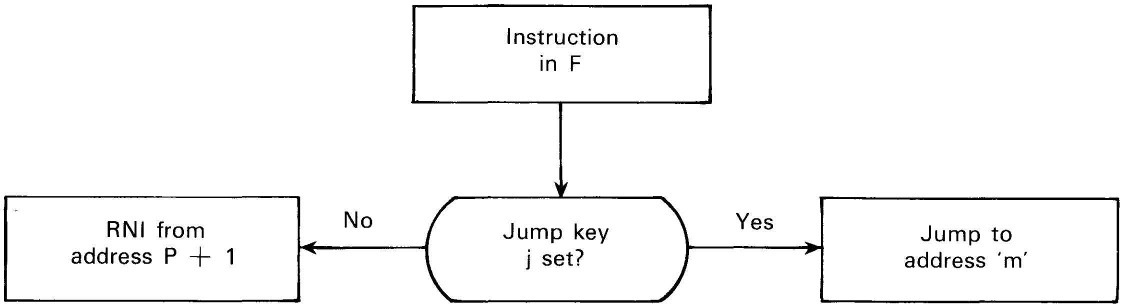
Instruction Description: Unconditionally stop at this instruction. Upon restarting, RNI from address 'm'. Indirect addressing and address modification are not applicable. (Approximate execution time: 1.8 μs.)

SJ1-6 Selective Jump

	23 18	17 15	14 00
Format:	0 0	j	m

j = 1 to 6

Instruction Description: Jump to address 'm' if Jump key j is set; otherwise, RNI from address P + 1. Indirect addressing and address modification are not applicable. (Approximate execution time: 1.8 μs.)



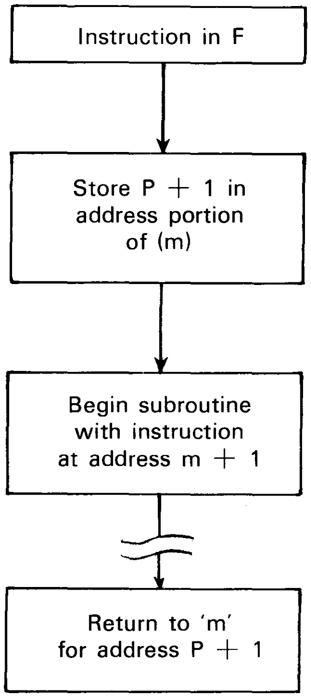
*Throughout this manual, the term (P) refers to the contents of the word addressed by P. This term shall be used because the more descriptive term, ((P)), becomes awkward when used frequently.

RTJ Return Jump

Format:

23 18	17 15	14 00
0 0	7	m

Instruction Description: The address portion of (m) is replaced with the return address P + 1. Jump to location m + 1 and begin executing instructions at that location. Indirect addressing and address modification are not applicable. (Approximate execution time: 3.5 μs.)


Instruction in F
Store P + 1 in address portion of (m)
Begin subroutine with instruction at address m + 1
Return to 'm' for address P + 1

UJP Unconditional Jump

Format:

23 18	17	16 15	14 00
01	a	b	m

Instruction Description: Unconditionally jump to address M. Indirect addressing and indexing are available. (Approximate execution time: 1.8 μs.)

IJI Index Jump, Incremental

Format:

23 18	17	16 15	14 00
02	d	b	m

d = 0
b = 1-3
m = jump address

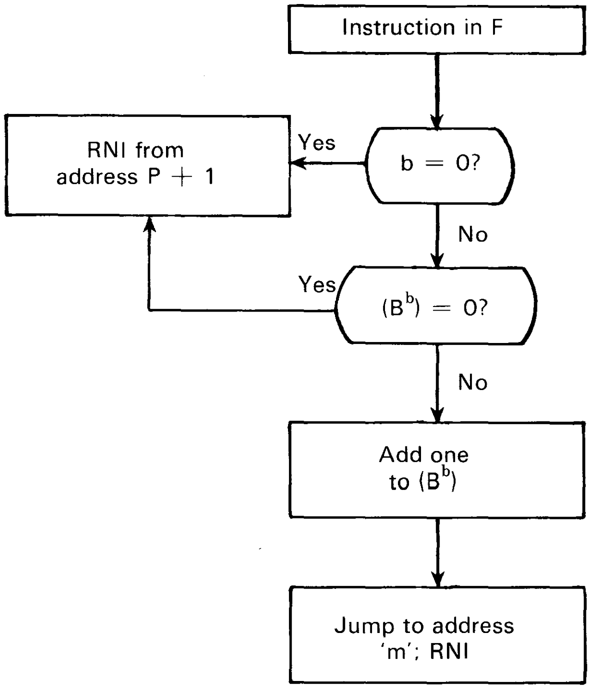
Instruction Description: If b = 0, this becomes a no-op instruction; RNI from address P + 1. If b≠0, (B^b) is examined.

1 If (B^b) = 0, the jump test condition is not satisfied; RNI from address P + 1.

2 If (B^b) ≠ 0, the jump test condition is satisfied. One is added to (B^b); jump to address 'm' and RNI.

Indirect addressing and jump address modification are not applicable. (Approximate execution time: 2.6 μs.)

Comments: The counting operation is done in a one's complement additive accumulator. Negative zero is not generated because the count progresses . . ., 77775, 77776, 00000, stopping at positive zero. If negative zero is initially in B^b, the count progresses 77777, 00001, etc. In this case, the counter must pass through its entire range to reach positive zero.



IJD Index Jump, Decremental

Format:

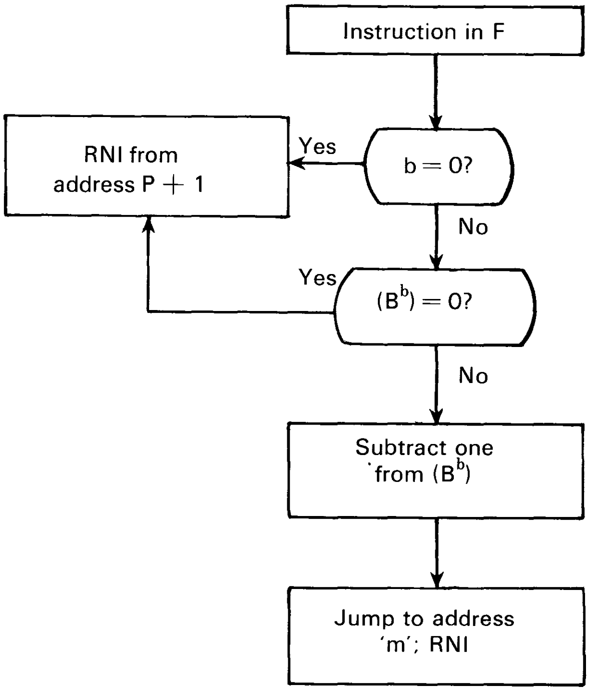
23 18	17	16 15	14 00
02	d	b	m

d = 1
b = 1-3
m = jump address

Instruction Description: If b=0, this becomes a no-op instruction; RNI from address P + 1. If b≠0, (B^b) is examined.

1 If (B^b) = 0, the jump test condition is not satisfied; RNI from address P + 1.
2 If (B^b) ≠ 0, the jump test condition is satisfied. One is subtracted from (B^b); jump to address 'm' and RNI.

Indirect addressing and jump address modification are not possible. (Approximate execution time: 2.6 μs.)



Comments: If negative zero is initially in B^b, the count must be decremented from 77777 to 00000 before the program will RNI from P + 1.

AZJ, Condition Compare A With Zero, Jump

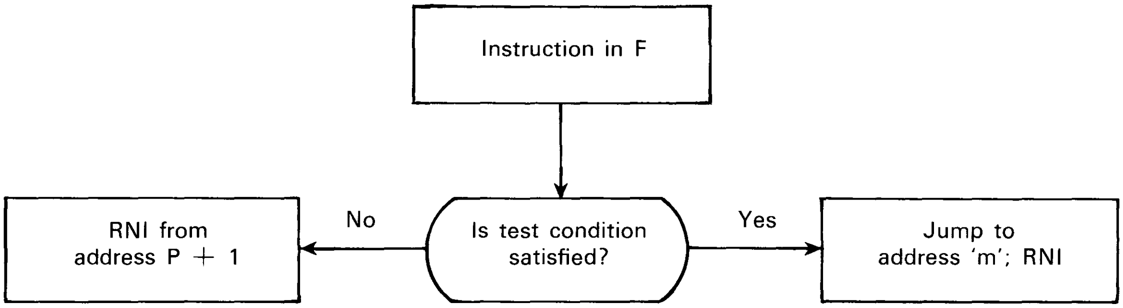
	23 18	17	16 15	14 00
Format:	03	d	j	m

d = 0
j = 0-3
m = jump address

Instruction Description: The quantity in A is compared algebraically with zero for an equality, inequality, greater than or less than condition (see table). If the test condition is true, the program jumps to address 'm'. If the test condition is not true, RNI from address P + 1. Indirect addressing and address modification are not applicable. (Approximate execution time: 2.6 μs.)

Condition Mnemonic	j	Test Condition
EQ	0	(A) = ± 0
NE	1	(A) ≠ ± 0
GE	2	(A) ≥ + 0
LT	3	(A) < + 0

Comments: Positive and negative zero give identical results in this test when j=0 or 1.



AQJ, Condition Compare A With Q, Jump

Format:

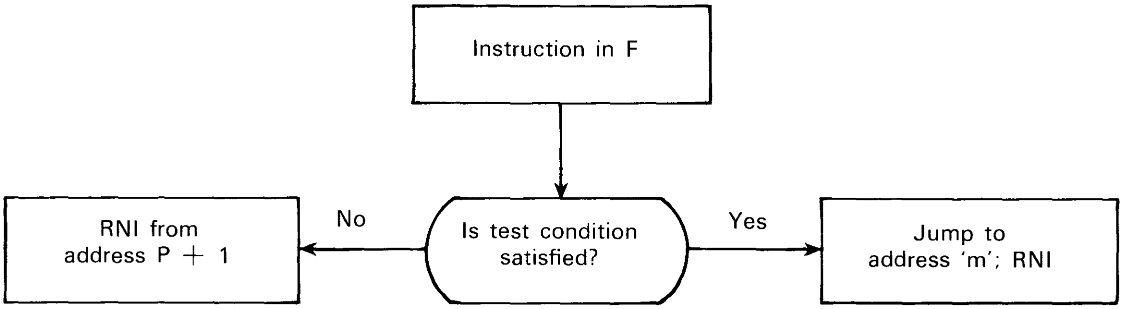
23 18	17	16 15	14 00
03	d	j	m

d = 1
j = 0-3
m = jump address

Instruction Description: The quantity in A is compared with the quantity in Q for an equality, inequality, greater than or less than condition (see table). If the test condition is true, the program jumps to address 'm'. If the test condition is not true, RNI from address P + 1. Indirect addressing and address modification are not applicable. (Approximate execution time: 2.6 μs.)

Condition Mnemonic	j	Test Condition
EQ	0	(A) = (Q)
NE	1	(A) ≠ (Q)
GE	2	(A) ≥ (Q)
LT	3	(A) < (Q)

Comments: This instruction may be used to test the contents of Q by placing an arbitrary value in A for the comparison. Positive and negative zero give identical results in this test when j=0 or 1.



REGISTER OPERATIONS WITHOUT STORAGE REFERENCE

Operational Field		Address Field	Interpretation
04	ASE, S	y	Skip next instruction if (A) = y
	QSE, S	y	Skip next instruction if (Q) = y
	ISE	y, b	Skip next instruction if (B^b) = y
05	ASG, S	y	Skip next instruction if (A) ≥ y
	QSG, S	y	Skip next instruction if (Q) ≥ y
	ISG	y, b	Skip next instruction if (B^b) ≥ y
14	ENA, S	y	Enter A with y
	ENQ, S	y	Enter Q with y
	ENI	y, b	Enter index with y
15	INA, S	y	Increase A by y
	INQ, S	y	Increase Q by y
	INI	y, b	Increase index by y
16	XOA, S	y	EXCLUSIVE OR of A and y
	XOQ, S	y	EXCLUSIVE OR of Q and y
	XOI	y, b	EXCLUSIVE OR of index and y
17	ANA, S	y	AND of A and y
	ANQ, S	y	AND of Q and y
	ANI	y, b	AND of index and y
10	ISI	y, b	Index skip, incremental
	ISD	y, b	Index skip, decremental
11	ECHA, S	y	Enter A with 17-bit character address
12	SHA	y, b	Shift A
	SHQ	y, b	Shift Q
13	SHAQ	y, b	Shift AQ
	SCAQ	y, b	Scale AQ

ISI Index Skip, Incremental

Format:

23 18	17	16 15	14 00
10	d	b	y

d = 0
b = 1 to 3

Instruction Description: If (B^b) = y, clear B^b and skip to address P + 2; if not, add one to (B^b) and RNI from address P + 1. (Approximate execution time: 2.6 μs.)

ISD Index Skip, Decremental

Format:

23 18	17	16 15	14 00
10	d	b	y

d = 1
b = 1 to 3

Instruction Description: If (B^b) = y, clear B^b and skip to address P + 2; if not, subtract one from (B^b) and RNI from address P + 1. (Approximate execution time: 2.6 μs.)

ECHA, S Enter Character Address Into A

Format:

23 18	17	16 00
11	d	z

Instruction Description: Clear A, then enter a 17-bit quantity 'y' (usually a character address) into A.

When d = 1, the sign is extended. (Approximate execution time: 1.8 μs.)

Instructions 04, 05, and 14-17 all refer to a register. Table 5-1 indicates the register and includes the corresponding operation codes, assembly mnemonics, and designators. When both 'd' and 'b' equal zero in instructions 04 and 05, zero is compared with 'y' and the instructions proceed just as if (B^b) was zero. Instructions 14-17 are no-ops when both 'd' and 'b' equal zero.

Table 5-1. Register Summary

Operation Codes and Mnemonics								
04	05	14	15	16	17	d	b	Register
ISE	ISG	ENI	INI*	XOI	ANI	0	1-3	B^b, no sign extended on B^b or 'y'
ASE,S	ASG,S	ENA,S	INA,S	XOA,S	ANA,S	1	†0	A, sign extension on 'y'
QSE,S	QSG,S	ENQ,S	INQ,S	XOQ,S	ANQ,S	1	†1	Q, sign extension on 'y'
ASE**	ASG**	ENA	INA	XOA	ANA	1	†2	A, no sign extension on 'y'
QSE**	QSG**	ENQ	INQ	XOQ	ANQ	1	†3	Q, no sign extension on 'y'

*Sign extension on 'y' and B^b

**Only the lower 15 bits of A or Q are used.

† When d = 1, 'b' does not serve in its usual role of index designator.

ASE, S Skip Next Instruction if (A) = y
QSE, S Skip Next Instruction If (Q) = y
ISE Skip Next Instruction If (B^b) = y

Format:

23 18	17	16 15	14 00
04	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: If the register contents equal 'y'; skip to address P + 2; if not, RNI from address P + 1. (Approximate execution time: 2.6 μs.)

ASG, S Skip Next Instruction If (A) ≥ y
QSG, S Skip Next Instruction If (Q) ≥ y
ISG Skip Next Instruction If (B^b) ≥ y

Format:

23 18	17	16 15	14 00
05	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: If the register contents are equal to or greater than 'y' skip to address P + 2; if not, RNI from address P + 1. (Approximate execution time: 2.6 μs.)

ENA, S Enter A with y
ENQ, S Enter Q with y
ENI Enter Index with y

Format:

23 18	17	16 15	14 00
14	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: Clear the register and enter 'y' directly into the register. (Approximate execution time: 1.8 μs.)

INA, S Increase A by y
INQ, S Increase Q by y
INI Increase Index by y

Format:

23 18	17	16 15	14 00
15	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: Add 'y' to the register contents. (Approximate execution time: 1.8 μs.)

XOA, S EXCLUSIVE OR of A and y
XOQ, S EXCLUSIVE OR of Q and y
XOI EXCLUSIVE OR of B^b and y

Format:

23 18	17	16 15	14 00
16	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: Enter the selective complement (the EXCLUSIVE OR function) of 'y' and register contents into the register. (Approximate execution time: 1.8 μs.)

ANA, S AND of A and y
ANQ, S AND of Q and y
ANI AND of B^b and y

Format:

23 18	17	16 15	14 00
17	d	b	y

See table 5-1 for 'b', 'd', and register.

Instruction Description: Enter the logical product (the AND function) of 'y' and the register contents into the register. (Approximate execution time: 1.8 μs.)

SHA Shift A
SHQ Shift Q

Format:

23 18	17	16 15	14 00
12	d	b	k

d = 0, SHA
d = 1, SHQ
b = index designator; $K = k + (B^b)$.

Instruction Description: (B^b) and k, with their signs extended, are added. (Even if b=0, the sign of 'k' is still extended.) The sign and magnitude of the 24-bit sum determine the direction and magnitude of shift. The computer only senses bits 00-05 and 23 of the sum for this information. To shift left, the magnitude of shift is placed in 'k'; to shift right, the complement of the magnitude of shift is placed in 'k'.

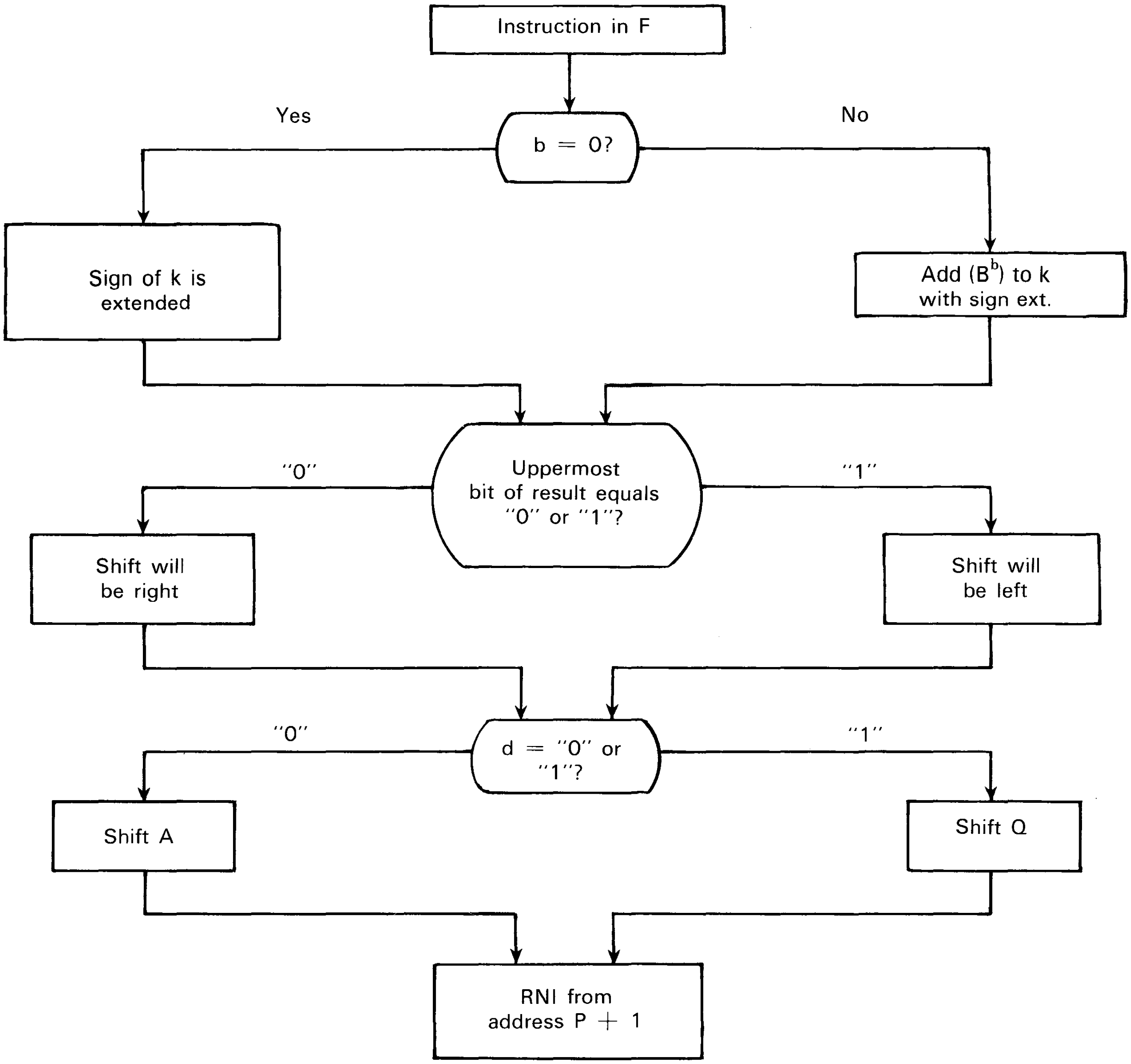
Examples: (b = 0 in both cases):

Shift left six positions:	k = 00006
Shift right six positions:	k = 77771

(Approximate execution time: 1.8 to 3.8 μs.)

Comments:
During left shifts, bits reaching the top of the A or Q register are brought end around. Therefore, a left shift of 24 places results in no change in (A) or (Q); a left shift of greater than 24 places results in an effective shift of K-24 (or K-48) places.

During right shifts, the sign bit is extended and the low order bits are discarded. A right shift of 23 or more places results in (A) or (Q) becoming all "0's" or all "1's", depending on the sign.



SHAQ Shift AQ

	23 18	17	16 15	14 00
Format:	13	d	b	k

d = 0
b = index designator; K = k + (B^b)

Instruction Description: The contents of A and Q are shifted together as one 48-bit register. (B^b) and 'k', with their signs extended, are added. (Even if b = 0, the sign of 'k' is still extended.) The sign and magnitude of the 24-bit sum determine the direction and magnitude of shift. The computer only senses bits 00-05 and 23 of the sum for this information. To shift left, the magnitude of shift is placed in 'k'; to shift right, the complement of the magnitude of shift is placed in 'k'.

Examples: (b = 0 in both cases):

Shift left three places:	k = 00003
Shift right three places:	k = 77774

(Approximate execution time: 2.6 to 5.1 μs.)

Comments:
During left shifts, bits reaching the top of the A register are brought end around to the lowest bit of Q. Therefore, a left shift of 48 places results in no change in (AQ); a left shift of greater than 48 places results in an effective shift of K-48 places.

During right shifts, the sign bit is extended and the low order bits are discarded. A right shift of 47 or more places results in (AQ) becoming all "0's" or all "1's", depending on the sign.

SCAQ Scale AQ

	23 18	17	16 15	14 00
Format:	13	d	b	k

d = 1
b = index designator
K = k—shift count; $(K \rightarrow B^b)$

Instruction Description: (AQ) is shifted left end around until the highest 2 bits (46 and 47) are unequal. If (AQ) should initially equal positive or negative zero, 48 decimal (60 octal) shifts are executed before the scale instruction terminates. During scaling, the computer makes a shift count. A quantity 'K', called the residue, equals 'k' minus the shift count. If b = 0, this quantity is discarded; if b = 1 to 3, the residue is placed in index B^b. (Approximate execution time: 2.6 to 5.1 μs.)

STORAGE TEST

Operation Field		Address Field	Interpretation
06	MEQ	m, b_1	Masked equality search
07	MTH	m, b_2	Masked threshold search
10	SSH	m	Storage shift
52	CPR, 1	m, b	Compare (within limits test)

MEQ Masked Equality Search

	23 18	17 15	14 00
Format:	06	i	m

i = 0 to 7, interval designator
m = unmodified storage address

Instruction Description: This instruction uses index B^1 exclusively. M = m + (B^1). (A) is compared with the logical product of (Q) and (M).
Instruction Sequence:

Decrement (B^1) by 'i'.

Test to see if (B^1) changed sign from positive to negative.

If so, RNI from P + 1; if not, test to see if A = Q.M.

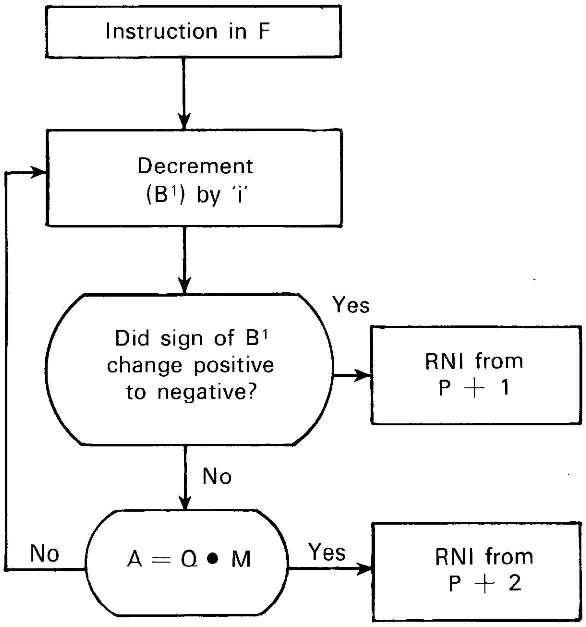
If A = Q.M, RNI from P + 2; if not, repeat sequence.

(Approximate execution time: 3.5 μs + n*-1.8 μs.)

Comments: 'i' is represented by 3 bits allowing a decrement interval selection of 1 to 8.

i	interval
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8

*n = number of words searched



MTH Masked Threshold Search

	23 18	17 15	14 00
Format:	07	i	m

i = 0 to 7, interval designator
m = unmodified storage address

Instruction Description: This instruction uses index B^2 exclusively. M = m +(B^2). (A) is compared with the logical product of (Q) and (M).

Instruction Sequence:

Decrement (B^2) by 'i'.

Test to see if (B^2) changed sign from positive to negative.

If so, RNI from P+1; if not, test to see if A ≥ Q • M.

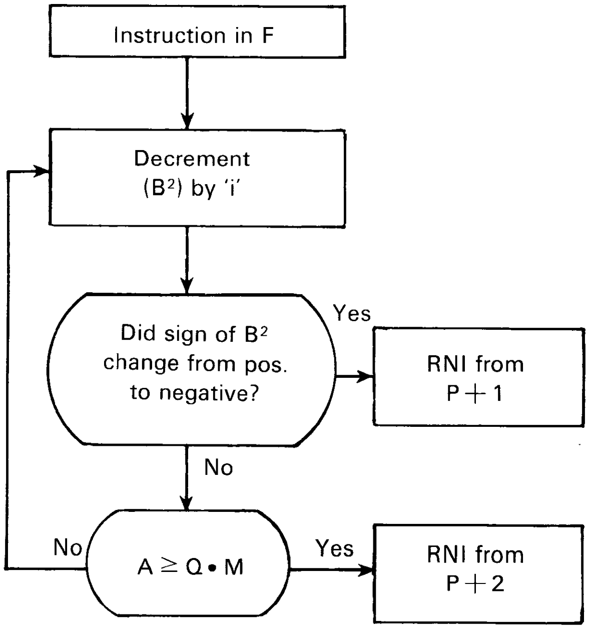
If A≥ Q • M, RNI from P+2; if not, repeat sequence.

(Approximate execution time: 3.5 μs + n*-1.8 μs.)

Comments: 'i' is represented by 3 bits allowing a decrement interval selection of 1 to 8.

i	interval
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8

*n = number of words searched

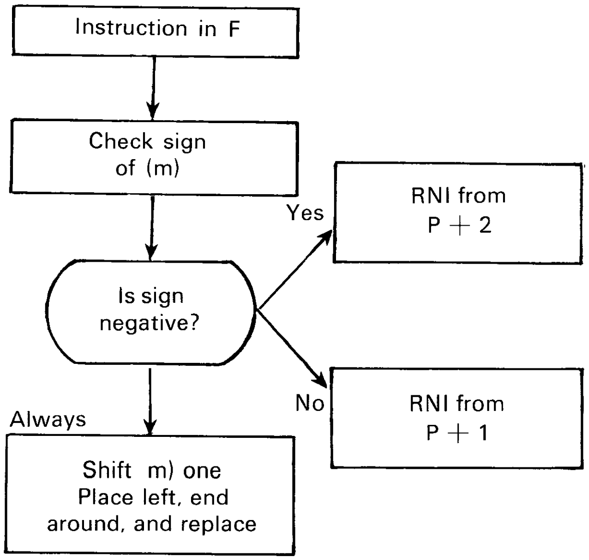


SSH Storage Shift

	23 18	17 15	14 00
Format:	10	0	m

m = storage address

Instruction Description: Sense bit 23 of (m). If (m) is negative, RNI from P + 2; if positive, RNI from P + 1. In both cases, shift (m) one place left, end around, and replace it in storage. (Approximate execution time: 5.3 μs.)



CPR Compare (Within Limits Test)

Format:

23 18	17	16 15	14 00
52	a	b	m

a = addressing mode designator
b = index designator
m = storage address

Instruction Description: (M) is tested to see if it is within the limits specified by A (upper limits) and Q (lower limits). The sequence of comparisons and the action taken are as follows:

1. Subtract (M) from (A) and place the difference in A. If A is negative, RNI from address P + 1; if not,
2. Subtract (Q) from (M) and place the difference in A. If A is negative, RNI from address P + 2; if not,
3. RNI from address P + 3.

Final State of Registers: (A) and (Q) remain unchanged. The address to which control proceeds, upon completion of the instruction is given by the following table:

Condition	Control Given To
(M) > (A)	P + 1
(Q) > (M)	P + 2
(A) ≥ (M) ≥ (Q)	P + 3

LOGICAL INSTRUCTIONS WITH STORAGE REFERENCE

Operation Field		Address Field	Interpretation
35	SSA, I	m, b	Selectively set A
36	SCA, I	m, b	Selectively complement A
37	LPA, I	m, b	Logical product A

SSA Selectively Set A

Format:

23 18	17	16 15	14 00
35	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Selectively set "1's" in A for all "1's" at address M. (Approximate execution time: 3.5 μs.)

SCA Selectively Complement A

Format:

23 18	17	16 15	14 00
36	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Selectively complement corresponding bits in A for all "1's" at address M. (Approximate execution time: 3.5 μs.)

LPA Logical Product A

Format:

23 18	17	16 15	14 00
37	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Replace A with the logical product of (A) and (M). (Approximate execution time: 3.5 μs.)

LOAD

Operation Field		Address Field	Interpretation
20	LDA, I	m, b	Load A
21	LDQ, I	m, b	Load Q
22	LACH	m, B^1	Load A, Character
23	LQCH	m, B^2	Load Q, Character
24	LACM, I	m, b	Load A, Complement
25	LDAQ, I	m, b	Load AQ
26	LAQC, I	m, b	Load AQ, Complement
27	LDL, I	m, b	Load A, Logical
54	LDI, I	m, b	Load Index

LDA Load A

Format:

23 18	17	16 15	14 00
20	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Load A with a 24-bit quantity from storage address M. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing and address modification are available.

LDQ Load Q

Format:

23 18	17	16 15	14 00
21	a	b	m

a = addressing mode designator
b = index designator; $M = m + (B^b)$
m = storage address

Instruction Description: Load Q with a 24-bit quantity from storage address M. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing and address modification are available.

LACH Load A, Character

Format:

23 18	17	16 00
22	b	r

16 02	01 00
00000-77777	0-3
word address	character designator

b = index designator. If b = 1, 'r' is modified by index register B^1; $R = r + (B^1)$.

Instruction Description: Load bits 0 to 5 of A with the character from storage specified by character address R. A is cleared prior to the load operation. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing is not applicable. Characters in storage are specified in the following manner:

23 18	17 12	11 06	05 00
0	1	2	3

character designators

LQCH Load Q, Character

Format:

23 18	17	16 00
23	b	r

16 02	01 00
00000-77777	0-3
word address	character designator

b = index designator. If b = 1, r is modified by index register B^2; $R = r + (B^2)$.

Instruction Description: Load bits 0 to 5 of Q with the character from storage specified by character address R. Q is cleared prior to the load operation. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing is not applicable. Characters in storage are specified in the following manner:

23 18	17 12	11 06	05 00
0	1	2	3

character designators

LCA Load A, Complement

Format:

23 18	17	16 15	14 00
24	a	b	m

a = addressing mode designator
b = index designator
m= storage address; M = m + (B^b)

Instruction Description: Load A with the complement of a 24-bit quantity from storage address M. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing and address modification are available.

LDAQ Load AQ

Format:

23 18	17	16 15	14 00
25	a	b	m

a = addressing mode designator
b = index designator
m= storage address; M = m + (B^b)

Instruction Description: Load registers A and Q with the two words from addresses 'M' and M + 1, respectively. Address 77777 should not be used. (Approximate execution time: 5.2 μs.)

LCAQ Load AQ, Complement

Format:

23 18	17	16 15	14 00
26	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Load registers A and Q with the complement of the two words from addresses M and M + 1, respectively. (Approximate execution time: 5.2 μs.)

LDL Load A, Logical

Format:

23 18	17	16 15	14 00
27	a	b	m

a = addressing mode designator
b = index designator

Instruction Description: Load A with the logical product (the AND function) of (Q) and the contents of address M. (Approximate execution time: 3.5 μs.)

LDI Load Index

Format:

23 18	17	16 15	14 00
54	a	b	m

a = addressing mode designator
b = index designator
m = storage address

Instruction Description: Load index register $B^b$with the lower 15 bits of storage address 'm'. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing, but no address modification, is possible. During indirect addressing only 'a' and 'm' are inspected. Symbol 'b' from the initial instruction specifies which index register is to be loaded with the storage address contents.

STORE

Operation Field		Address Field	Interpretation	
40	STA, I	m, b	Store A	2.5 μsec
41	STQ, I	m, b	Store Q	
42	SACH	m, B^2	Store A, character	
43	SQCH	m, B^1	Store Q, character	
44	SWA, I	m, b	Store 15-bit word address	
45	STAQ, I	m, b	Store AQ	3.8 μsec
46	SCHA	m, b	Store 17-bit character address	2.5 μsec
47	STI, I	m, b	Store index	

STA Store A

Format:

23 18	17	16 15	14 00
40	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Store (A) in storage address M. (Approximate execution time: 3.5 μs.)

STQ Store Q

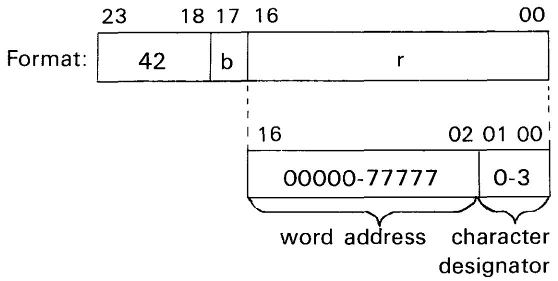
Format:

23 18	17	16 15	14 00
41	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Store (Q) in storage address M. (Approximate execution time: 3.5 μs.)

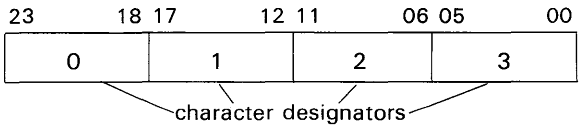
SACH Store A, Character



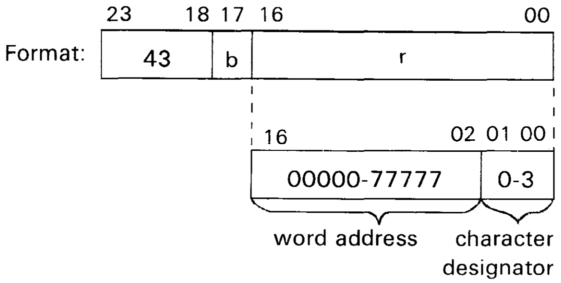
b = index designator. If b = 1, r is modified by index register B^2; $R = r + (B^2)$.

Instruction Description: Store the contents of bits 0 to 5 of the A register in the specified character address. All of A and the remaining three characters in storage remain unchanged. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing is not applicable. Characters in storage are specified in the following manner:



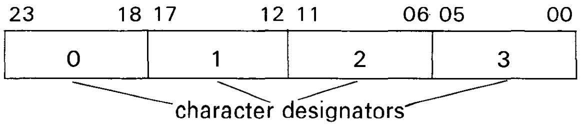
SQCH Store Q, Character



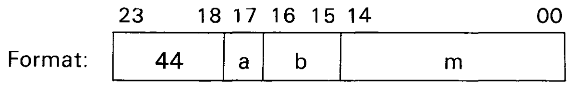
b = index designator. If b = 1, r is modified by index register B^1; $R = r + (B^1)$.

Instruction Description: Store the contents of bits 0 to 5 of the Q register in the specified character address. All of Q and the remaining three characters in storage remain unchanged. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing is not applicable. Characters in storage are specified in the following manner:



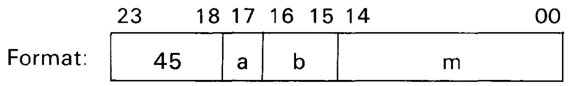
SWA Store Word Address



a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Store the lower 15 bits of (A) in the designated address M. The upper 9 bits of M remain unchanged. (Approximate execution time: 3.5 μs.)

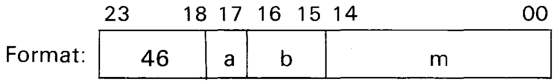
STAQ Store AQ



a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Store the contents of registers A and Q in storage addresses M and M + 1, respectively. Address 77777 should not be used. (Approximate execution time: 5.2 μs.)

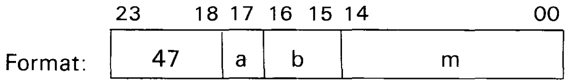
SCHA Store Character Address



a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Store the lower 17 bits of (A) in the designated address M. The upper bits of M remain unchanged. (Approximate execution time: 3.5 μs.)

STI Store Index



a = addressing mode designator
b = index designator
m = storage address

Instruction Description: Store the (B^b) in the lower 15 bits of storage address 'm'. The upper 9 bits of 'm' remain unchanged. (Approximate execution time: 3.5 μs.)

Comments: Indirect addressing, but no address modification, is possible. During indirect addressing only 'a' and 'm' are inspected. Symbol 'b' from the initial instruction specifies which index register is to have its contents stored. If b = 0, zeros are stored in m.

INTER-REGISTER TRANSFER 24-Bit Precision

Operational Field		Address Field	Interpretation
53	TIA	b	Transfer (B^b) to A
	TAI	b	Transfer (A) to B^b
	TMQ	m	Transfer (Register m) to Q
	TQM	m	Transfer (Q) to Register m
	TMA	m	Transfer (Register m) to A
	TAM	m	Transfer (A) to Register m
	TMI	m, b	Transfer (Register m) to B^b
	TIM	m, b	Transfer (B^b) to Register m
	AQA	- - -	Transfer (A) + (Q) to A
	AIA	b	Transfer (A) + (B^b) to A
	IAI	b	Transfer (B^b) + (A) to B^b

The 53 instruction is used to move data between the A and Q (arithmetic) registers, the index registers, and the register file.

TIA Transfer (B^b) to (A)
TAI Transfer (A) to (B^b)

Format:

23 18	17	16 15	14 12	11 00
53	d	b	0	- - -

d= 0, TIA; d = 1, TAI
b = index designator, 1 to 3
Comment: No sign extension.

TMQ Transfer (Register v) to Q
TQM Transfer (Q) to Register v

Format:

23 18	17	16 15	14 12	11 06	05 00
53	d	- - -	1	- - -	v

d = 0, TMQ; d= 1, TQM
v = register number, 00-77

TMA Transfer (Register v) to A
TAM Transfer (A) to Register v

Format:

23 18	17	16 15	14 12	11 06	05 00
53	d	- - -	2	- - -	v

d = 0, TMA; d = 1, TAM
v = register number, 00-77

TMI Transfer (Register v) to B^b
TIM Transfer (B^b) to Register v

Format:

23 18	17	16 15	14 12	11 06	05 00
53	d	b	3	- - -	v

d = 0, TMI; d = 1, TIM
b = index designator, 1 to 3
v = register number, 00-77

AQA Transfer (A) + (Q) to A

Format:

23 18	17 15	14 12	11 00
53	0	4	- - -

AIA Transfer (A) + (B^b) to A

Format:

23 18	17	16 15	14 12	11 00
53	0	b	4	- - -

b = index designator, 1 to 3

Instruction Description: The sign of (B^b) is extended.

IAI Transfer (A) + (B^b) to B^b

Format:

23 18	17	16 15	14 12	11 00
53	1	b	4	- - -

b = index designator, 1 to 3

Instruction Description: The sign of the original (B^b) is extended prior to the addition. The upper 9 bits are lost when the sum is placed in B^b.

ARITHMETIC, FIXED-POINT, 24-BIT PRECISION

Operation Field		Address Field	Interpretation
30	ADA, I	m, b	Add to A
31	SBA, I	m, b	Subtract from A
34	RAD, I	m, b	Replace add
50	MUA, I	m, b	Multiply A
51	DVA, I	m, b	Divide A

ADA Add to A

Format:

23 18	17	16 15	14 00
30	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Add a 24-bit quantity located at address M to (A). The sum appears in A. (Approximate execution time: 3.5 μs.)

SBA Subtract from A

Format:

23 18	17	16 15	14 00
31	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Subtract a 24-bit quantity located at address M from (A). The difference appears in A. (Approximate execution time: 3.5 μs.)

RAD Replace Add

Format:

23 18	17	16 15	14 00
34	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Replace the quantity in M with the sum of (M) and (A). The original (A) remain unchanged. (Approximate execution time: 5.2 μs.)

MUA Multiply A

Format:

23 18	17	16 15	14 00
50	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Multiply (A) by the quantity at address M. The 48-bit product appears in QA with the lowest order bits in A. (Approximate execution time: 14.5 μs.)

DVA Divide A

Format:

23 18	17	16 15	14 00
51	a	b	m

a = addressing mode designator
b = index designator
m = storage address; $M = m + (B^b)$

Instruction Description: Divide the 48-bit quantity in AQ by the quantity at storage address M. The quotient appears in A and the remainder with its sign extended appears in Q. If a divide fault occurs, this operation halts and the program advances to the next instruction. The final contents of A and Q are meaningless if this happens. (Approximate execution time: 15.0 μs.)

ARITHMETIC, FIXED-POINT, 48-BIT PRECISION

Operation Field		Address Field	Interpretation	
32	ADAQ, I	m, b	Add to AQ	
33	SBAQ, I	m, b	Subtract from AQ	
56	MUAQ, I	m, b	Multiply AQ	Trapped Instructions
57	DVAQ, I	m, b	Divide AQ	

Comments: **Instructions 56 and 57 are trapped in 3100 computer systems.**

Registers A and Q serve together as a 48-bit register with the highest order bits in A.

ADAQ Add to AQ

	23 18	17	16 15	14 00
Format:	32	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Add the 48-bit contents of addresses M and M + 1 to (AQ). The sum appears in AQ. (Approximate execution time: 5.2 μs.)

SBAQ Subtract from AQ

	23 18	17	16 15	14 00
Format:	33	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 48-bit combined contents of addresses M and M + 1 from (AQ). The difference appears in AQ. (Approximate execution time 5.2 μs.)

MUAQ Multiply AQ

	23 18	17	16 15	14 00
Format:	56	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply (AQ) by the 48-bit operand in M and M + 1. The 96-bit product appears in AQE.

DVAQ Divide AQ

	23 18	17	16 15	14 00
Format:	57	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Divide (AQE) by the 48-bit contents of addresses M and M + 1. The answer appears in AQ and the remainder with its sign extended appears in E. If a divide fault occurs, this operation halts and the program advances to the next instruction. The final contents of AQ and E are meaningless if this happens.

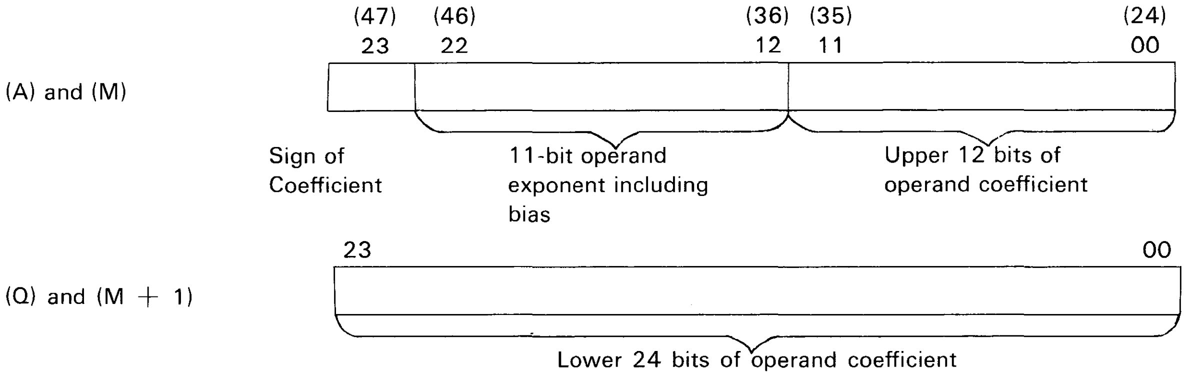
ARITHMETIC, FLOATING POINT

Operational Field		Address Field	Interpretation
60	FAD, I	m, b	FP addition to AQ
61	FSB, I	m, b	FP subtraction from AQ
62	FMU, I	m, b	FP multiplication of AQ
63	FDV, I	m, b	FP division of AQ

This group of instructions is trapped in 3100 computer systems. The E and D registers are simulated in 3100 memory by appropriate software and are not separate physical entities.

Comments: All floating point operations in the 3100 computer involve the A and Q registers plus two consecutive storage locations, 'm' and m + 1. The A and Q registers are treated as one 48-bit register.

Operand Formats: The AQ register and the storage address contents have identical formats.


(47) 23
(46) 22
(36) 12
(35) 11
(24) 00
(A) and (M)
Sign of Coefficient
11-bit operand exponent including bias
Upper 12 bits of operand coefficient
23
00
(Q) and (M + 1)
Lower 24 bits of operand coefficient

Exponent Equalization: During floating point addition and subtraction, the exponents involved are equalized prior to the operation. The coefficient of the algebraically smaller exponent is automatically shifted right until the exponents are equal.

Rounding: Rounding modifies the coefficient answer by adding one to AQ for positive answers or by subtracting one for negative answers. Rounding is necessary since the coefficient answer may contain more than 36 bits. The condition for rounding is inequality of the sign bits of AQ and E. This means that the next lower significant bit to the right of the number in AQ is equal to or greater than one-half. Coefficient arithmetic may yield rounded answers from zero to 2^{37}.

Normalizing: Normalizing brings the above answer back to a fraction from one-half to one with the binary point to the left of the 36th bit. The magnitude of the final normalized number in AQ will range from 2^{36} to 2^{37}-1. Normalizing is performed by either a right shift or the required number of left shifts for add and subtract or a one place right or left shift for multiply and divide. The exponent is corrected for every shift. The residue in E is not shifted.

Exponent Overflow: It is possible to sense exponent overflow and/or to use this overflow to cause an interrupt. Sensing this fault automatically clears the Exponent Overflow indicator.

FAD FP Addition to AQ

Format:

23 18	17	16 15	14 00
60	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Add the contents of M and M + 1 to (AQ). The rounded and normalized sum appears in AQ.

FSB FP Subtraction from AQ

Format:

23 18	17	16 15	14 00
61	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 48-bit contents of 'M' and M + 1 from (AQ). The rounded and normalized difference appears in AQ.

FMU FP Multiplication of AQ

Format:

23 18	17	16 15	14 00
62	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply (AQ) by the 48-bit contents of 'M' and M + 1. The rounded and normalized product appears in AQ.

FDV FP Division of AQ

Format:

23 18	17	16 15	14 00
63	a	b	m

a = addressing mode designator
b = index designator
m = storage address; M = m + (B^b)

Instruction Description: Divide (AQ) by the 48-bit contents of M and M + 1. The rounded and normalized quotient appears in AQ. The remainder with sign extended appears in the E register.

BCD

Operational Field		Address Field	Interpretation
70	SFE	k, b	Shift E
	EZJ, EQ	m	E zero jump, E = 0
	EZJ, LT	m	E zero jump, E < 0
	EOJ	m	E overflow jump
	SET	y	Set D register
64	LDE	m, b[1]	Load E
65	STE	m, b[2]	Store E
66	ADE	m, b[3]	Addition to E
67	SBE	m, b[3]	Subtraction from E

This group of instructions is trapped in 3100 computer systems. The E and D registers are simulated in 3100 memory by appropriate software and are not separate physical entities.

Formats: These instructions handle 4-bit BCD characters rather than whole 24-bit words. These characters are placed into the simulated E register and into storage in the following ways:

1 E^D Register (Simulated Configuration).

53	51												00
±	13	12	11	10	9	8	7	6	5	4	3	2	1

Sign of E

Overflow character position

BCD Characters

The simulated 53-bit E^D register can hold 12 regular BCD characters plus one overflow character. The E^D register can never be displayed on the consoles.

2 Storage

	23 18	17 12	11 06	05 00
(M) =	0	1	2	3

Character Positions

Each 24-bit storage word may be divided into four character positions of 6 bits each. The lower 4 bits of each position may hold a 4-bit character; the upper 2 bits are reserved for the sign designator, one per field. For each field, the sign accompanies the least significant character. 10xxxx specifies negative; any other combination, positive. The upper 2 bits of all other characters in the field must equal zero. The most significant character precedes the least significant character of a field in storage.

Field Length: The field length is specified by the contents of the simulated 4-bit D register. Any number 1-12 (0001-1100) is legal.*

Illegal Characters: By definition, any character greater than 9(or 11_8) is illegal. Characters are tested for legality during:

1. Loading into E,
2. Storing as they leave E, and
3. Addition and subtraction as they leave E and storage for processing by the adder.

BCD Fault: The BCD fault will occur if:

1. A sign is present in any character position other than the least significant, or
2. An illegal character is sensed during the execution of an instruction, or
3. The contents of D exceed 12 (will set only during a SET instruction).

SFE Shift E

	23 18	17	16 15	14 00
Format:	70	d	b	k

d = 0
b = index designator
k = shift designator

Instruction Description: This instruction shifts BCD characters within the E register in one character (4-bit) steps. 'k' and the contents of index B^b are added to modify the shift designator; $K = k + (B^b)$. The computer senses bits 00-03 and 23 of the sum.

*Although a fault will occur, D may equal 13 for the storage of 13 characters. The following sequence should be followed in storing 13 characters:

1. Set D (BCD fault will occur)
2. Sense for BCD fault (this clears the BCD Fault indicator)
3. Issue STE instruction

If the BCD fault is disregarded and there is an attempt to load, add, or subtract 13 characters, only the lower 12 characters will be used. No additional fault will occur.

Direction of Shift: Shifting is left if bit 23 is zero; right if it is one. Shifts are end off in both directions.

Magnitude of Shift: For a left shift, the lower 4 bits of the sum specify the shift magnitude; for a right shift, the lower 4 bits of the complement of the sum specify the shift magnitude.

Examples:

If K = 00000006, shift left 6 character positions.
If K = 77777771, shift right 6 character positions.

EZJ, EQ R Zero Jump, (E) = O
EZJ, LT E Zero Jump, (E) < O

Format:

23 18	17	16 15	14 00
70	d	j	m

d = 1
j = jump test designator
m = jump address

Instruction Description: This instruction compares (E) with zero. If the test condition is true, jump to address m; if not, RNI from address P + 1. See the table below for test conditions.

Mnemonic	j	Test Condition
EQ	0	(E) = 0
LT	1	(E) < 0

EOJ E Overflow Jump

Format:

23 18	17 15	14 00
70	6	m

m = jump address

Instruction Description: Jump to address 'm' if digit 13 of the E^D register receive a character indicating that E^D has overflowed; if not, RNI from address P + 1.

SET Set D Register

Format:

23 18	17 15	14 00
70	7	y

y = field length designator

Instruction Description: Place the lower 4 bits of 'y' in the simulated 4-bit D register. (D) remains set until a new quantity is entered. In other words, during a series of load and store operations dealing with equal size fields, (D) need only be set once.

LDE Load E^D

Format:

23 18	17	16 00
64	b	r

16 02	01 00
00000-77777	0-3
Word Address	Character position within the word

If b=0, r is the unmodified direct address.
If b=1, r is modified by (B^1); $R=r+(B^1)$.

Instruction Description: Load the 53-bit E^D register (includes sign of E^D) with a field of up to 12 numeric BCD characters from storage. Characters are read consecutively, starting with the least significant character (at address R + (D − 1) and continuing until the most significant character (at address R) is in E^D. (E^D) is shifted right as loading progresses. The sign is acquired along with the least significant character. Before executing this instruction, specify field length with a SET (70.7) instruction.

STE Store E^D

Format:

23 18	17	16 00
65	b	r

If b=0, r is the unmodified direct address.
If b=1, r is modified by the (B^2); $R=r+(B^2)$.

Instruction Description: Store a field of up to 13 numeric BCD characters from the 53-bit E^D register (includes sign of E^D). Storage begins with the least significant character and the sign. As it continues, (E^D) is shifted right, end off, until the field is stored. Before executing this instruction, specify field length with a SET (70.7) instruction.

ADE Addition to (E^D)

Format:

23 18	17	16 00
66	b	r

If b=0, r is the unmodified direct address.
If b=1, r is modified by (B^3); $R=r+(B^3)$.

*Instruction Description**: A field of up to 12 stored numeric characters may be added to (E^D). The sum appears in E^D. Stored characters are in consecutive character positions of adjacent storage addresses. 'R' specifies the most significant character of a field. The 4-bit D register specifies field length.

SBE Subtraction from (E^D)

Format:

23 18	17	16 00
67	b	r

If b=0, r is the unmodified direct address.
If b=1, r is modified by (B^3); R=r+(B^3).

*Instruction Description**: A field of up to 12 stored numeric characters may be subtracted from (E^D). See instruction 66 for remainder of description.

BLOCK OPERATIONS—SEARCH, MOVE, AND I/O

Operation Field		Address Field	Interpretation
71	SRCE,INT	c, m^1, m^2	Search character equality
	SRCN,INT	c, m^1, m^2	Search character inequality
72	MOVE,INT	c, m^1, m^2	Move c characters from m^1 to m^2
73	INPC,NC,INT,B,H	ch, m^1, m^2	Input, character block to storage
	INAC,NC,INT	ch	Input, character to A
74	INPW,NC,INT,B,N	ch, m^1, m^2	Input, word block to storage
	INAW,NC,INT	ch	Input, word to A
75	OUTC,NC,INT,B,H	ch, m^1, m^2	Output, character block from storage
	OTAC,NC,INT	ch	Output, character from A
76	OUTW,NC,INT,B,H	ch, m^1, m^2	Output, word block from storage
	OTAW,NC,INT	ch	Output, word from A

Comments: These instructions have the following characteristics in common:

1. They are composed of three words, including the two main block instruction words plus a one word reject instruction.
2. Addresses required for the execution of the instruction set are located within the instruction set.
3. Constants such as field lengths and BCD codes for search characters are within the instruction set.
4. They can all be set to cause an interrupt upon completion.

See chapter 3, Programming Features, for a description of the Block instructions.

*The A and Q registers are not used for these instructions.

SENSING, CONTROL AND INTERRUPT

Operation Field		Address Field	Interpretation
77.0	CON	x, ch	connect
77.1	SEL	x, ch	select
77.2	COPY	x, ch; x = 0	copy external status
77.2	EXS	x, ch; x ≠ 0	external sense
77.3	CINS	x, ch; x = 0	copy internal status
77.3	INS	x, ch; x ≠ 0	internal sense
77.4	INTS	x, ch	interrupt sense
77.50	INCL	x	interrupt clear
77.51	IOCL	x	I/O clear
77.52	SSIM	x	selectively set interrupt mask
77.53	SCIM	x	selectively clear interrupt mask
77.57	IAPR		interrupt associated processor
77.6	PAUS		pause
77.70	SLS		selective stop
77.71	SFPF		*set FP fault
77.72	SBCD		*set BCD fault
77.73	DINT		disable interrupt control
77.74	EINT		enable interrupt control
77.75	CTI		console typewriter in
77.76	CTO		console typewriter out
77.77	UCS		unconditional stop

Comments: 77 is an instruction that handles sensing, selecting, interrupt and control functions not covered by instructions 00-76.

The general format for all sub-divisions of the 77 instruction is:

	23 18	17 15	14 12	11 00
Format:	77	0-7	0-3	0000-7777
	Operation Code	Operation Code Modifier	Channel designator or special usage	Comparison mask, function code, etc.

Throughout this instruction, the term Busy may mean:

channel writing,
channel reading,
I/O equipment Reject on channel,
last Connect on channel not yet recognized, or
last Function on channel not yet recognized.

77.0 c xxxx Connect Channel to I/O Equipment

	23 18	17 15	14 12	11 00
Format:	77	0	c	xxxx

c = I/O channel designator 0-3.

xxxx = 12-bit connect code. Bits 09-11 select one of eight controllers which may be attached to the selected channel. Bits 00-08 select one of a possible 512 units which may be connected to the selected controller.

*Used for software simulation of the optional arithmetic packages.

Instruction Description: Channel c is checked for Busy. If Busy is present, a reject instruction is read from address P + 1. If channel c is not Busy, a 12-bit connect code is sent on channel c along with a connect enable; then the next instruction is read from address P + 2.

77.1 c xxxx Select Function of I/O Equipment

Format:

23 18	17 15	14 12	11 00
77	1	c	xxxx

c = I/O channel designator 0-3.

xxxx = 12-bit function code. Each piece of peripheral equipment has a unique set of function codes to specify operations within that device. Refer to the individual peripheral equipment manuals for these codes.

Instruction Description: Channel c is checked for Busy. If Busy is present, a reject instruction is read from address P + 1. If channel c is not Busy, a 12-bit function code xxxx is sent on channel c along with a function enable; then the next instruction is read from address P + 2.

COPY Interrupt Mask and External Status to A. EXS (Bits 00 through 11 specifying sensing of Interrupt Mask Register).

Format:

23 18	17 15	14 12	11 00
77	2	ch	0 0 0 0

ch = I/O channel designator 0-3

Instruction Description: This is a dual purpose instruction:

A The external status code from I/O channel C is loaded into the lower 12 bits of A.

B The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. See Table 5-4. RNI from address P + 1.

CINS Interrupt Mask and Internal Status to A. INS (Bits 00 through 11 specifying sensing of Interrupt Mask Register).

Format:

23 18	17 15	14 12	11 00
77	3	ch	0 0 0 0

ch = I/O channel designator 0-3

Instruction Description: This is a dual instruction:

A The internal status code of the computer is loaded into the lower 12 bits of A.

B The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. See Table 5-4. RNI from address P + 1.

INTS Sense Interrupt

Format:

23 18	17 15	14 12	11 00
77	4	ch	xxxx

ch = I/O module, channels 0-3.

xxxx = sense mask. Bits 00-03 of the mask represent interrupt lines from the designated I/O channel; bits 08-11 represent internal interrupt conditions.

Instruction Description: Sense for the interrupt conditions listed in table 5-4. If "1" bits appear on the interrupt lines in any of the same positions as "1" bits in the mask, RNI from address P + 1. If comparison does not occur in any of the bit positions, skip to address P + 2. Internal interrupts are cleared as soon as they are sensed.

Table 5-4. Interrupt Sensing Mask

Comparison Mask Bit Positions	Definitions
00-07	I/O line 0-3 interrupt active
*08	Clock interrupt
*09	Exponent overflow or BCD fault
*10	Arithmetic overflow or divide fault
*11	Search/move completion interrupt

INCL Clear Interrupt

Format:

23 18	17 15	14 12	11 00
77	5	0	xxxx

Instruction Description: The interrupt faults defined by xxxx are cleared (see table 5-5). Note that only internal I/O channel interrupts are cleared by this instruction.

Table 5-5. Interrupt Mask Register

Bit Positions	Definitions
00-07	I/O channel 0-7 interrupts (internal and external)
08	Clock interrupt
09	Exponent overflow or BCD fault
10	Arithmetic overflow or divide fault
11	Search/move completion interrupt

*FFs associated with these faults are cleared as soon as the conditions are sensed.

SSIM Selectively Set Interrupt Mask Register

	23 18	17 15	14 12	11 00
Format:	77	5	2	xxxx

Instruction Description: Selectively sets the Interrupt Mask register according to xxxx. For each "1" bit in xxxx, the corresponding bit position in the Interrupt Mask register is set to "1" (see table 5-5).

SCIM Selectively Clear Interrupt Mask Register

	23 18	17 15	14 12	11 00
Format:	77	5	3	xxxx

Instruction Description: Selectively clears the Interrupt Mask register according to xxxx. For each "1" bit in xxxx, the corresponding bit position in the Interrupt Mask register is set to "0" (see table 5-5).

IAPR Interrupt Associated Processor

	23 18	17 15	14 12	11 00
Format:	77	5	7	----

Instruction Description: This instruction sends an interrupt signal to a processor (computer). The interrupt remains active until it is recognized.

SFPF Set Floating Point Fault

	23 18	17 15	14 12	11 00
Format:	77	7	1	----

Instruction Description: The floating-point fault flip-flop is set to indicate that a floating point fault has occurred. This instruction is used when floating-point arithmetic is simulated and causes a flip-flop to set whenever a fault is sensed. The setting of this flip-flop causes bit 09 to be set in the Interrupt register and permits a normal hardware interrupt.

SBCD Set BCD Fault

	23 18	17 15	14 12	11 00
Format:	77	7	2	----

Instruction Description: This instruction exists for the same reason as 77.71. In this case the BCD Fault flip-flop is set.

DINT Disable Interrupt Control

	23 18	17 15	14 12	11 00
Format:	77	7	3	----

Instruction Description: Interrupt control is enabled. This instruction allows one more instruction to be executed before any interrupt can take place.

EINT Enable Interrupt Control

	23 18	17 15	14 12	11 00
Format:	77	7	4	----

Instruction Description: Interrupt control is enabled. This instruction allows one more instruction to be executed before any interrupt can take place.

Appendix A

3100 Compass

This appendix describes the capabilities of the 3100 COMPASS assembly system and is not intended as a final system description.

This information is preliminary and subject to change without notice.

Coding Procedures

3100 COMPASS subprograms are written on standard coding sheets. A subprogram consists of symbolic or octal machine instructions and pseudo instructions. Symbolic machine instructions are alphabetic mnemonics for each of the 3100 machine instructions. Pseudo instructions are COMPASS instructions used for the following operations:

subprogram identification and linkage
data definition (constants conversion)
data storage
system calls
assembler control
output listing control
macro definition

INSTRUCTION FORMAT

A COMPASS instruction may contain location, instruction, address, comment, and identification fields.

LOCATION FIELD

A symbol in the location field (LOCN) is placed in columns 1-8. A symbol identifies the address of an instruction or data item.

Location field symbols may be blank or consist of one to eight alphabetic or numeric characters; the first character must be alphabetic. Embedded blanks are ignored in location symbols. The following are examples of location symbols:

A
H3
ABCDEFGH
P1234567

A single * in the location field signifies a line of comments.

OPERATION CODE FIELD

The operation code field (OP) consists of any of the 3100 mnemonic or octal instruction codes with modifiers, or any macro or pseudo instructions. The field begins in column 10 and ends at the first blank column. If a modifier is used, a comma must separate the operation code from the modifier; no blank columns may intervene. A blank operation field or a blank in column 10 results in a machine word with zeros in the operation field.

ADDRESS FIELD

The address field begins before column 41 anywhere after the blank which terminates the operation field and ends at the first blank column. It is composed of one or more subfields, depending upon the instruction. Subfields, which are separated by commas on the coding form, specify the following quantities:

m or n	word address
r or s	character address
y	operand (15-bit)
z	operand (17-bit)
b or i	index register or interval quantity
c	character
v	register file location
ch	channel
x	function code or comparison mask

The interpretations of the address subfields for each set of 3100 instructions are described in the table on page A-2.

An m, n, r, s, y or z subfield may contain:

- a location symbol
- the symbol ** which causes each bit in the subfield to be set to one
- the symbol * which causes the assembler to insert the relocatable address of that instruction in the address field
- an integer constant
- an arithmetic expression
- a literal

b SUBFIELD—The index field (b) specifies an index register 1-3; or a symbol or expression which results in one of these digits may be used. Some instructions require a particular index register. If the b subfield is used with the octal operation codes, 0-7 may be used.

c SUBFIELD—The character field may contain any octal or decimal number, expression, or a symbol which is equivalent to a 6-bit binary number. Octal numbers must be suffixed with the letter B.

ch SUBFIELD—The channel field may contain one digit 0-3 to designate an input/output channel, or a symbol equated to one of these digits, or an expression resulting in one of the digits.

x SUBFIELD—The code field may contain any of the interrupt or input/output codes or comparison mask. Either decimal numbers, octal numbers suffixed with the letter B, symbols, or expressions resulting in constants may be used.

v SUBFIELD—The register file subfield specifies a location which may be 00_8-77_8. Any legal coding which results in a value 00_8-77_8 may be used.

i SUBFIELD—In the MEQ and MTH instructions, this subfield specifies a decrement interval quantity of 1-8.

COMMENTS FIELD

Comments may be included with any instructions. A blank column must separate them from the last character in the address field and they may extend to column 72. Comments have no effect upon compilation, but are included on the assembly listing.

INSTRUCTION

		00-70	71 (Search)	72 (Move)	73-77 (I/O)
FIELD	m, n	word address			first word address, last word address + 1
	b	index register			
	y or z	operand		operand	
	c		character		
	r	character address	address of first character	first character address of source field	first character address
	s		address of last character + 1	first character address of receiving field	last character address + 1
	ch				channel
	x				I/O or interrupt code
	i	Interval quantity			

IDENTIFICATION FIELD

Columns 73-80 may be used for sequence numbers or for program identification. This field has no effect upon assembly.

Pseudo-Instructions

MONITOR CONTROL

The following pseudo instructions provide communication between 3100 COMPASS subprograms and the monitor. Some are required in every subprogram; others are optional. Unless otherwise noted, each instruction may have a location field and an address field.

IDENT m—appears at the beginning of every COMPASS subprogram. The address field contains the name of the subprogram, which may be a maximum of eight alphanumeric characters, the first being alphabetic. A symbol in the location field is illegal and will result in an error flag (L) on the listing.

END m—marks the end of every subprogram. When a program (consisting of one or more subprograms) is assembled for execution, one of the subprogram END cards must contain a location symbol in the address field to indicate the first instruction to be executed in the program. Only one END card can contain an address field symbol. A term in the location field is ignored.

FINIS—terminates an assembly operation. It is a signal to the assembler that no more programs are to be assembled. The FINIS card is placed after the last END card of the last subprogram in the source program.

SYMBOL ASSIGNMENTS

The pseudo instructions, EQU; EQU,C; ENTRY; and EXT define symbols as equal to other symbols, or values or identify symbols used to communicate with subprograms. Linkage between symbols in separate subprograms is provided by the monitor system. These pseudo instructions may appear anywhere between an IDENT and an END pseudo instruction.

EQU m—assigns the result of the expression in the address field to the symbol in the location field. The result is a 15-bit address.

The following forms are allowed:

```
symbol   EQU   symbol
symbol   EQU   constant (octal or decinal)
symbol   EQU   expression (address arithmetic)
```

Example:

```
OUT   EQU   JUMP + 2
```

If JUMP is assembled to address 00100, OUT will be assigned the value 00102.

Numerical constants must follow the rules for symbolic instructions. Address arithmetic is permitted. A location field symbol may be equated to a decimal or octal constant.

EQU, C m—is similar to EQU, except that the result is a 17-bit address.

ENTRY m—defines location symbols which are referenced in other subprograms. These symbols, called entry points, must be placed in the address field of an ENTRY pseudo instruction. Any number of locations may be declared as entry points in the same ENTRY instruction. If two or more names appear in the address field, they must be separated by commas. No spaces (blanks) can appear within a string of symbols. The address field of the ENTRY pseudo instruction may be extended to column 72 and the location field must be blank. Only word-location symbols (15-bits) may be used.

Example:

```
ENTRY   SYM1,SYM2,SYM3
```

SYM1, SYM2, SYM3 can now be referenced by other subprograms.

EXT m—Symbols used by a subprogram defined in another subprogram are declared as external symbols by placing them in the address field of an EXT pseudo instruction. Only word-location symbols (15-bit) may be used. For example, to use the external symbols SYM1, SYM2, SYM3 in subprogram A, the following pseudo instruction would be written in subprogram A:

```
EXT   SYM1,SYM2,SYM3
```

These symbols must be declared as ENTRY points in some other subprogram or subprograms which are loaded for execution with subprogram A. The address field may be extended to column 72; symbols are separated by commas. No spaces (blanks) can appear in a string of symbols. The location field of an EXT must be blank.

Address arithmetic cannot be performed on external symbols.

Example:

```
          IDENT     CAIRO
          ENTRY     DEED, FFI
          EXT       ABE, DAVID
FFI       SJ1       **
            •
BEN       EQU       HAKIM
            •
DEED      LDA       ABE
            •
            •
            •
          RTJ       DAVID
            •
            •
          END
          END       FFI
          FINIS
```

LISTING CONTROL

The pseudo instructions which provide listing control for assembly listings are shown below. These instructions will not appear on the assembly listing and may be placed anywhere in a program.

SPACE—controls line spacing on an assembly listing. A decimal constant in the address field designates the number of spaces to be skipped before printing the next line. If the number of spaces to be skipped is greater than the number of lines remaining to be printed on a page, the line printer skips to the top of the next page. A symbol in the location field is ignored.

EJECT—causes the line printer to skip to the top of the next page when the assembled program is listed. A symbol in the location field is ignored.

REM—is used to insert program comments in an assembly listing. The address field can be extended to column 72. Any standard key punch character can be used in the comments. If the comments are to be written on more than one line, successive REM pseudo instructions must be used. A symbol in the location field is ignored.

NOLIST—causes the assembler to discontinue writing a listing of the program, starting with this instruction.

LIST—causes the assembler to resume listing the program. This instruction is used after a NOLIST instruction; it is not necessary to use it to obtain a complete listing of a program.

MACRO INSTRUCTIONS

MACRO—defines the beginning of a sequence of instructions that will be inserted by the assembler in the source program whenever the location symbol of the MACRO instruction appears in an operation field. The end of the sequence of instruction is marked by an ENDM pseudo instruction. For example, if the sequence

```
HOPE  MACRO  (PA, MA)
      LDA    PA
      INA    24B
      STA    MA
      ENDM
```

were defined and the following instructions appeared in the same program

```
STA   GARAGE
HOPE  (DW21 D6)
LDA   FARM
```

the assembled output would be

```
STA   GARAGE
LDA   DW21
INA   24B
STA   D6
LDA   FARM
```

ENDM—defines the end of a macro sequence.

LIBM—names library macros.

NAME (p_1,...,p_n)—is used to reference macros. The parameters p_1,...,p_2 are used by the routine, and name is a system defined macro.

DATA STORAGE ASSIGNMENTS

The following pseudo instructions reserve storage areas for blocks of data. BSS reserves storage blocks within the subprogram in which it appears. If these storage areas are to be referenced by other subprograms, the name assigned to the block is declared as an entry point in the program containing the block, and as an external symbol in the program referencing the block. Only work location symbols may be used. COMMON identifies storage areas to be referenced by more than one subprogram. DATA specifies special areas which may be preloaded with data; EXT and ENTRY are not needed to reference COMMON or DATA areas. Address arithmetic may be used, but all symbols must have been defined before the instruction is encountered.

BSS m—reserves a storage area of length m in a subprogram on a common or data storage area. The address field may contain any expression which results in a constant. The resultant constant specifies number of words to be used. The address field of the first word of the reserved area is assigned the location field term of the BSS instruction. Other words or characters in the area may be referenced by addressing arithmetic or by indexing.

BSS, C m—reserves a character storage area of length m in a subprogram. The address field is similar to the address field of BSS pseudo instruction. However, the resultant constant specifies the number of character positions to be reserved.

COMMON—assigns location terms following it to a common storage block until a DATA or PRG pseudo instruction is encountered. ORGR, BSS and BSS, C are the only pseudo instructions which may follow a COMMON pseudo instruction.* Location and address fields of a COMMON pseudo instruction should be blank. COMMON may not be preset with data. The following example illustrates the foregoing pseudo instructions:

*Occurrence of any other machine or data definition command causes the command and its successors to be assembled into the subprogram area.

Example:

```
            IDENT      BURKE
            COMMON
       A    BSS        20
       B    BSS        10
       C    BSS        6
             •
             •
             •
            END
            IDENT      SPINOZA
            COMMON
MARKET      BSS        5
STREET      BSS        13
SINGER      BSS        4
            END
```

During execution, one area in storage is assigned as common. All common storage may be filled repeatedly during program execution. A storage location assigned to the nth word in COMMON in subprogram 1 is the same location assigned to the nth word in common in subprogram 2. If the two subprograms in the above example were loaded together, the memory assignments would be as shown in the following table:

Locations in memory relative to the beginning of common	Name in subprogram BURKE	Name in subprogram SPINOZA
1-5	A ——→ A+4	MARKET ——→ MARKET+4
6-18	A+5 ——→ A+17	STREET ——→ STREET+12
19-20	A+18 ——→ A+19	SINGER ——→ SINGER+1
21-22	B ——→ B+1	SINGER+2 ——→ SINGER+3
23-30	B+2 ——→ B+9	
31-36	C ——→ C+5	

DATA—assigns all location symbols following it to a data block until a COMMON or PRG pseudo instruction is encountered. Data described by OCT; BCD; BCD,C; DEC; DECD and VFD pseudo instructions may be assembled into a DATA block. Areas may be reserved within a DATA block by the BSS and BSS,C pseudo instructions. The following is an example of a DATA pseudo instruction coded within a subprogram:

Example:

```
             •
             •
             •
            LDA        APRESMOI
            DATA
CONS        OCT        10,11,12,13
            PRG        *
            STA        LEDELUGE
```

A data area named CONS would be reserved and the octal constants 10, 11, 12, and 13 loaded into the four words in this area. In the source program, STA LEDELUGE would appear in the next location after LDA APRESMOI.

PRG—terminates the definition of a COMMON or DATA area.

CONSTANTS

Octal, decimal, and BCD constants may be inserted in a 3100 COMPASS program by using the pseudo instructions listed below. Location terms may be used and the address field may extend to column 72, if necessary.

OCT $m_1,m_2,\ldots,m_n$—inserts octal constants into consecutive machine words. A location term is optional; if present, it will be assigned to the first word. The address field consists of one or more consecutive subterms, separated by commas. Each subterm may consist of a sign (+ or −or none), followed by up to eight octal digits. Each constant is assigned to a separate word. If a location term is present, it will be assigned to the first word. If less than eight digits are specified, the constant is right justified in the word and leading zeros inserted.

DECD $m_1,m_2,\ldots,m_n$—converts decimal constants to equivalent 48-bit binary values and stores them in consecutive groups of two machine words. Each constant may be written in either fixed or floating point format.

The decimal numbers to be converted are written in the address field of the DECD instruction as follows:

Floating point constant consists of a signed or unsigned decimal integer of 14 digits. It is identified as a floating point constant by a decimal point which may appear anywhere within the digital string. A binary scale factor or decimal scale factor (indicated by B ± b or D ± d, respectively) is permitted. The result after scaling must not exceed the capacity of the hardware (approximately $10 \pm^{308}$).

Fixed point constant format is similar to that of the DEC single precision constants. Up to 14 decimal digits may be specified, expressing a value the magnitude of which is less than 2^{47}. Decimal and binary scale factors may be used. Low order bits are not lost; the signed 48-bit binary result is stored in two consecutive computer words.

No spaces may occur within a number, including its associated scale factors, as a space indicates the end of the constant. Plus signs may be omitted. Any number of constants may appear in a DECD instruction. Successive constants are separated by commas.

Examples:

LOCN	Op	Address Field	Comments
CONST A	DECD	−12345.	FLOATING PT CONST
CONST B	DECD	+12345	FIXED PT CONST
CONST C	DECD	−12345.D+5	FLOATING PT CONST, DECSCALE
CONST D	DECD	12345D−3	FIXED PT CONST, DECSCALE
CONST E	DECD	+12345B+8	FIXED PT CONST, BINSCALE
CONST F	DECD	+12345.D12B−18	FLOATING PT CONST, DECBIN SC

DEC $m_1,m_2,\ldots,m_n$—inserts 24-bit decimal integer constants in consecutive machine words. The D and B scaling is identical to the DECD scaling, but only positive integer values less than 2^{33} may be used. If a location term is present, it is assigned to the first constant.

BCD $n,c_1c_2,\ldots,c_{4n}$—inserts binary-coded decimal characters into consecutive words. If a location term is present, it will be assigned to the first word. The address field consists of a single digit n, which specifies the number of four-character words needed to store the BCD constant, followed by a comma and the BCD characters. The next 4n character positions after the comma will be stored. Any character string which terminates before column 73 may be used; n is restricted accordingly.

BCD,C $n,c_1c_2,\ldots,c_n$—places n characters in the next available m character positions in memory. If the previous instruction were also a BCD,C instruction, the next character position is defined as the one which follows the last position used by the previous instruction. If a location symbol is used, it will be assigned to the first character position in this field. If the previous instruction were not a BCD,C instruction, the next character position would be the first character position (0) of the next available word. Any character string which terminates before column 73 may be used; n is restricted accordingly.

VFD $m_1n_1/v_1\ldots/m_pn_p/v_p$—assigns data in continuous strings of bits rather than in word units. Octal numbers, character codes, program locations and arithmetic values may be assigned consecutively in memory, regardless of word breaks. The address field consists of one or more data fields. In each data field m specifies the mode of the data, n the number of bits allotted, and v the value. Four modes are allowed:

- **O** Octal number. If it is preceded by a minus sign, the one's complement form is stored.
- **H** Hollerith character code. The field length must be a multiple of six. Any printable character may appear in the v field except blanks or commas. Either a space or comma immediately succeeds the last character.
- **A** Arithmetic expression or decimal constant. The v field consists of an expression formed according to the rules for address field arithmetic, with the following restrictions:

1 n must be $\leq$ 24 and $|v| \leq 2^{n-1}-1$ unless a relocatable expression is used, in which case, n = 15 for word addresses and n = 17 for character addresses.

2 When a relocatable expression is used, it must be placed in the correct position in the address portion of a word to insure that it will be relocated by the loader.

C Character Expression.

Example:

VFD 012/-737,A27/A-X+B,H24/+A3 ,A15/NAME+2,H12/BQ

A, X, and B are non-relocatable symbols. Four words are generated, with the data placed as follows:

23	12	0
70 40		[A*

WORD 1

23	14	8	0
-X + B	20	21	0

WORD 2

23		14	0
3	60	[NAME + 2]	

WORD 3

23		12	0
B	Q	0 0 0 0	

WORD 4

The VFD address field is terminated by the first blank column not within a Hollerith field.

ADDITIONAL PSEUDO INSTRUCTIONS

Additional lines of coding may be generated by the following pseudo instructions:

IFZ m,n—n succeeding lines of coding will be assembled if m is zero. The integer n must be a positive numerical integer and m may be a symbol, an address arithmetic symbol, or a literal. If m is nonzero, n succeeding lines of coding will be bypassed by the assembler.

IFN m,n—n succeeding lines of coding will be as sembled if m is nonzero; n must be a positive numerical integer and m may be a symbol, address arithmetic symbol or literal. If m is zero, n succeeding lines of coding will be bypassed by the assembler.

The pseudo instructions, IFT and IFN, may be used within the range of a MACRO definition only.

IFT m,n,p—p succeeding lines of coding will be generated if character string m equals character string n. The integer p must be a positive numerical integer and m and n may be a formal parameter or a literal. If m $\neq$ n, p succeeding lines of coding will be bypassed.

IFF m,n,p—p succeeding lines of coding will be generated if m $\neq$ n. The integer p must be a positive integer and m and n may be a formal parameter or a literal. If m = n, p succeeding lines of coding will be bypassed.

ORGR m—the value in the address field will be assembled as the beginning location for subsequent instructions. The value may be in program, data area, or common area mode. The occurrence of a mode change pseudo operation, COMMON, DATA or PRG, terminates ORGR and subsequent instructions are assembled in the new mode.

NOP—No operation. An ENI y, O instruction is inserted.

TITLE—the information beginning in the address field is printed at the head of each page of the output listing which follows. The first page of listing may be titled by presenting the TITLE card immediately following the IDENT card.

ASSEMBLY LISTING FORMAT

An assembly listing contains the source program instructions and the corresponding octal machine instructions. The addresses assigned to each subprogram are relative addresses only. Absolute addresses are assigned when the program is loaded by the monitor loader. All common blocks are assigned consecutively, starting at relative location 00000. The range of locations assigned to the machine instructions (first word address and last word address plus one) are given at the beginning of each subprogram. Following this is a list of all entry points and external symbols, and the address assignments for all COMMON and DATA pseudo instructions. References to external symbols are strung together by the assembler. The monitor loader assigns the proper absolute addresses.

The address of each instruction word is the leftmost field for each instruction in the assembled listing. (Error codes appear to the left of this field.) External address field symbols are indicated by an X immediately to the left of the octal address field of each instruction. P indicates Program Relocatable, and C indicates Common. Subsequent next fields from left to right on the listing are an 8-digit location contents field, a 2-digit operation code, a l-digit b-subfield, a 5-digit address, and a l-digit character position. The remaining fields correspond to those in the symbolic source program. Listing format:

location	location contents	op	b	addr	char pos	source line
5 or 6 digits	8	2	1	5	1	80

ERROR CODES

The following error codes may appear as the leftmost field on an assembled listing:

Code

A Illegal character or expression in the address field.

D Same symbol used in more than one location field term. Only the first symbol is recognized; the remainder are ignored. A list of doubly defined symbols appears on the assembled listing.

F Symbol table is full. No more location field symbols will be recognized. Also designates overflow of MACRO parameter table.

O Illegal operation code. Zeros are substituted for the operation code.

U Undefined symbol. The assembler assigns the symbol to a region following the last program entry. A list of undefined symbols will appear on the output listing.

C An attempt was made to preset COMMON.

L A symbol appears in the location field when not permitted, a symbol is missing in the location field when one is required, or an illegal location symbol appears.

M A modifier appears in the location field when not permitted, a modifier is missing in the operation field when one is required, or an illegal modifier appears in the operation field.

T A character address symbol was used in an address subfield requiring a word symbol; significant bits are lost.

TABLE A-1
3100 COMPASS and BASIC ASSEMBLER MACHINE INSTRUCTIONS

Operation Field		Address Field	Instruction
00	HLT	m	unconditional stop; read next instruction from location m
	SJ1	m	jump if key 1 is set
	SJ2	m	jump if key 2 is set
	SJ3	m	jump if key 3 is set
	SJ4	m	jump if key 4 is set
	SJ5	m	jump if key 5 is set
	SJ6	m	jump if key 6 is set
	RTJ	m	return jump
01	UJP, I	m, b	unconditional jump
02	IJI	m, b	index jump; increment index
	IJD	m, b	index jump; decrement index
03	AZJ, EQ	m	compare A with zero; jump if (A) = 0
	NE		jump if (A) ≠ 0
	GE		jump if (A) ≥ 0
	LT		jump if (A) ≤ 0
	AQJ, EQ	m	compare A with Q; jump if (A) = (Q)
	NE		jump if (A) ≠ (Q)
	GE		jump if (A) ≥ (Q)
	LT		jump if (A) ≤ (Q)
04	ASE, S	y	skip next instruction, if (A) = y
	QSE, S	y	skip next instruction, if (Q) = y
	ISE	y, b	skip next instruction, if (B^b) = y
05	ASG, S	y	skip next instruction, if (A) ≥ y
	QSG, S	y	skip next instruction, if (Q) ≤ y
	ISG	y, b	skip next instruction, if (B^b) ≦ y
06	MEQ	m, i	masked threshold search
07	MTH	m, i	masked equality search
10	ISI	y, b	index skip; increment index
	ISD	y, b	index skip; decrement index
	SSH	m	storage shift
11	ECHA, S	z	enter A with 17-bit character address
12	SHA	y, b	shift A
	SHQ	y, b	shift Q
13	SHAQ	y, b	shift AQ
	SCAQ	y, b	scale AQ
14	ENA	y	enter A
	ENI	y, b	enter index
	ENQ	y	enter Q
15	INA	y	increase A
	INI	y, b	increase index
	INQ	y	increase Q
16	XOA, S	y	exclusive OR y and (A)
	XOQ, S	y	exclusive OR y and (Q)
	XOI	y, b	exclusive OR y and (B^b)
17	ANA, S	y	logical product (AND) of y and (A)
	ANQ, S	y	logical product (AND) of y and (Q)
	ANI	y, b	logical product (AND) of y and (B^b)

TABLE A-1 — (cont.)

Operation Field		Address Field	Instruction
20	LDA, I	m, b	load A
21	LDQ, I	m, b	load Q
22	LACH	r, 1	load A character
23	LQCH	r, 2	load Q character
24	LCA, I	m, b	load A complement
25	LDAQ, I	m, b	load AQ (double precision)
26	LCAQ, I	m, b	load AQ complement (double precision)
27	LDL, I	m, b	load logical
30	ADA, I	m, b	add to A
31	SBA, I	m, b	subtract from A
32	ADAQ, I	m, b	add to AQ
33	SBAQ, I	m, b	subtract from AQ
34	RAD, I	m, b	replace add
35	SSA, I	m, b	selectively set A
36	SCA, I	m, b	selectively complement A
37	LPA, I	m, b	logical product with A
40	STA, I	m, b	store A
41	STQ, I	m, b	store Q
42	SACH	r, 2	store character from A
43	SQCH	r, 1	store character from Q
44	SWA, I	m, b	store 15-bit word address from A
45	STAQ, I	m, b	store AQ
46	SCHA, I	m, b	store 17-bit character address from A
47	STI, I	m, b	store index
50	MUA, I	m, b	multiply A
51	DVA, I	m, b	divide AQ (48 by 24)
52	CPR, I	m, b	within limits test
53	TIA	b	transmit (B^b) to A
	TAI	b	transmit (A) to B^b
	TMA	v	transmit (high speed memory) to A
	TAM	v	transmit (A) to high speed memory
	TMQ	v	transmit (high speed memory) to Q
	TQM	v	transmit (Q) to high speed memory
	TMI	v, b	transmit (high speed memory) to B^b
	TIM	v, b	transmit (B^b) to high speed memory
	AQA		transmit (A) + (Q) to A
	AIA	b	transmit (A) + (B^b) to A
	IAI	b	transmit (B^b) + (A) to B^b
54	LDI, I	m, b	load index
56*	MUAQ, I	m, b	multiply AQE (96 by 48)
57*	DVAQ, I	m, b	divide AQE (48 by 48)
60*	FAD, I	m, b	floating add to AQ
61*	FSB, I	m, b	floating subtract from AQ

*Trapped instructions.

TABLE A-1 — (cont.)

Operation Field		Address Field	Instruction
62*	FMU, I	m, b	floating multiply AQ
63*	FDV, I	m, b	floating divide AQ
64*	LDE	r, 1	load E
65*	STE	r, 2	store E
66*	ADE	r, 3	add to E
67*	SBE	r, 3	subtract from E
70*	SFE	y, b	shift E
	EZJ, EQ	m	compare E with zero; jump if E = 0
	LT		compare E with zero; jump if E < 0
	EOJ	m	jump to m on E overflow
	SET	y	set D to value of y
71	SRCE, INT	c, m_1, m_2	search character equality
	SRCN, INT	c, m_1, m_2	search character inequality
72	MOVE, INT	ℓ, r, s	move ℓ characters from r to s
73	INPC, INT, B, H, A or NC	ch, r, s	input character block to memory
	INAC, A or NC	ch	input character to A
74	INPW, INT, B, N, A or NC	ch, m, n	input word block to memory
	INAW, A or NC	ch	input word to A
75	OUTC, INT, B, H, A or NC	ch, r, s	output character block from memory
	OTAC, A or NC	ch	output character from A
76	OUTW, INT, B, N, A or NC	ch, m, n	output word block from memory
	OTAW, A or NC	ch	output word from A
77.0	CON	x, ch	connect
77.1	SEL	x, ch	select
77.20	COPY	x, ch x=0	copy status
77.2	EXS	x, ch x≠0	external sense
77.3	INS	x, ch x≠0	internal sense
77.30	CINS	x=0	copy internal status
77.4	INTS	x, ch	interrupt sense
77.50	INCL	x	interrupt clear
77.51	IOCL	x	I/O clear
77.52	SSIM	x	selective set interrupt mask
77.53	SCIM	x	selective clear interrupt mask
77.57	IAPR	x	interrupt associated processor
77.6	PAUS	x	pause
77.70	SLS		selective stop
77.71	SFPF		set floating point fault
77.72	SBCD		set BCD fault
77.73	DINT		disable interrupt control
77.74	EINT		enable interrupt control
77.75	CTI		console typewriter in
77.76	CTO		console typewriter out
77.77	UCS		unconditional stop

*Trapped instructions.

Appendix B

Basic Assembler Coding Procedures

Basic Assembler Coding Procedures

BASIC Assembler programs are written in a manner similar to 3100 COMPASS programs. Each program is a complete entity and may be designed for any 3100 equipment configuration. Object programs produced by the BASIC Assembler for the 3104 4K storage configuration are loaded by the BASIC Loader; those for 8K or larger configurations are loaded by 3100 SCOPE.

INSTRUCTION FORMAT consists of the following fields:

LOCATION FIELD—from one to six alphabetic or numeric characters; the first character must be alphabetic.

OPERATION FIELD—any of the 3100 mnemonic instruction codes with modifiers, or the BASIC Assembler pseudo instructions.

ADDRESS FIELD—from one to six character location symbols, the special ** or * symbol, an integer constant, or an expression (address arithmetic) consisting of two terms.

COMMENTS FIELD—may be included with any instruction. A full line of comments may be inserted by placing an asterisk in the location field.

IDENTIFICATION FIELD—sequence number or program identification.

Pseudo-Instructions

PROGRAM IDENTIFICATION is provided for each program.

IDENT m appears at the beginning of a BASIC Assembler program. The address field contains the name of the subprogram.

END m marks the end of the program. The address field may contain a symbol which is used as the entry point to the program.

SYMBOL ASSIGNMENTS for each program.

EQU m equates an undefined symbol to a defined word address symbol.

EQU, C m equates an undefined symbol to a defined character address symbol.

ORGR m assembles the value specified in the address field as the beginning location for subsequent instructions. A symbol in the address field must be defined elsewhere in the program.

NOP m inserts a "do-nothing" instruction. The address field may contain a symbol.

LISTING CONTROL for assembly listings.

SPACE controls line spacing.

EJECT moves the line printer to the top of the next page.

REM is used to insert program comments.

NOLIST suppresses the output listing lines.

LIST resumes printing after a NOLIST instruction.

DATA STORAGE ASSIGNMENTS

BSS m reserves a block of words of length m.

BSS, C m reserves a block of characters of length m.

DATA DEFINITION

OCT m inserts an octal constant into a machine word.

DEC m inserts a single precision decimal constant into a machine word. Decimal and binary scaling is permitted.

DECD m inserts a double precision decimal constant into two consecutive machine words. Floating or fixed point numbers are allowed, also decimal and binary scaling.

BCD n,$c_1c_2...c_{4n}$ inserts binary-coded decimal characters into consecutive words.

BCD, Cn,$c_1c_2...c_n$ inserts binary-coded decimal characters into the next available n character positions in storage.

ASSEMBLY LISTING FORMAT

An assembly listing contains the source program and corresponding octal machine instructions. The program may be loaded absolutely, beginning at location 00000 or relocated into memory relative to some location other than 00000. Error codes correspond to 3100 COMPASS error codes; A, D, F, L, M, O and T codes are included.

Appendix C

Number Systems

- **ARITHMETIC**
- **CONVERSIONS**
- **FIXED POINT AND FLOATING POINT NUMBERS**

Number Systems

Any number system may be defined by two characteristics, the radix or base and the modulus. The radix or base is the number of unique symbols used in the system. The decimal system has ten symbols, 0 through 9. Modulus is the number of unique quantities or magnitudes a given system can distinguish. For example, an adding machine with ten digits, or counting wheels, would have a modulus of 10^{10}-1. The decimal system has no modulus because an infinite number of digits can be written, but the adding machine has a modulus because the highest number which can be expressed is 9,999,999,999.

Most number systems are positional, that is, the relative position of a symbol determines its magnitude. In the decimal system, a 5 in the units column represents a different quantity than a 5 in the tens column. Quantities equal to or greater than 1 may be represented by using the 10 symbols as coefficients of ascending powers of the base 10. The number 984_{10} is:

$$\begin{array}{rcrcr} 9 \text{ x } 10^{2} & = & 9 \text{ x } 100 & = & 900 \\ +8 \text{ x } 10^{1} & = & 8 \text{ x } 10 & = & 80 \\ +4 \text{ x } 10^{0} & = & 4 \text{ x } 1 & = & \underline{4} \\ & & & & 984_{10} \end{array}$$

Quantities less than 1 may be represented by using the 10 symbols as coefficients of ascending negative powers of the base 10. The number 0.593_{10} may be represented as:

$$\begin{array}{rcrcr} 5 \text{ x } 10^{-1} & = & 5 \text{ x } .1 & = & .5 \\ +9 \text{ x } 10^{-2} & = & 9 \text{ x } .01 & = & .09 \\ 2\ \ 3 \text{ x } 10^{-3} & = & 3 \text{ x } .001 & = & \underline{.003} \\ & & & & 0.593_{10} \end{array}$$

BINARY NUMBER SYSTEM

Computers operate faster and more efficiently by using the binary number system. There are only two symbols 0 and 1; the base = 2. The following shows the positional value:

... 2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
=32	=16	=8	=4	=2	=1	Binary point

The binary number 0 1 1 0 1 0 represents:

$$\begin{array}{rcrcr} 0 \text{ x } 2^{5} & = & 0 \text{ x } 32 & = & 0 \\ +1 \text{ x } 2^{4} & = & 1 \text{ x } 16 & = & 16 \\ +1 \text{ x } 2^{3} & = & 1 \text{ x } 8 & = & 8 \\ +0 \text{ x } 2^{2} & = & 0 \text{ x } 4 & = & 0 \\ +1 \text{ x } 2^{1} & = & 1 \text{ x } 2 & = & 2 \\ +0 \text{ x } 2^{0} & = & 0 \text{ x } 1 & = & \underline{0} \\ & & & & 26_{10} \end{array}$$

Fractional binary numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

	2^{1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}...
Binary Point	= ½	= ¼	= ⅛	= 1/16	= 1/32

The binary number 0.10 110 may be represented as:

$$\begin{array}{rcrcrcr} 1 \text{ x } 2^{-1} & = & 1 \text{ x } 1/2 & = & 1/2 & = & 8/16 \\ +0 \text{ x } 2^{-2} & = & 0 \text{ x } 1/4 & = & 0 & = & 0 \\ +1 \text{ x } 2^{-3} & = & 1 \text{ x } 1/8 & = & 1/8 & = & 2/16 \\ +1 \text{ x } 2^{4} & = & 1 \text{ x } 1/16 & = & 1/16 & = & \underline{1/16} \\ & & & & & & 11/16_{10} \end{array}$$

OCTAL NUMBER SYSTEM

The octal number system uses eight discrete symbols, 0 through 7. With base eight the positional value is:

... 8^{5}	8^{4}	8^{3}	8^{2}	8^{1}	8^{0}
32,768	4,096	512	64	8	1

The octal number 513_{8} represents:

$$\begin{array}{rcrcr} 5 \text{ x } 8^{2} & = & 5 \text{ x } 64 & = & 320 \\ +1 \text{ x } 8^{1} & = & 1 \text{ x } 8 & = & 8 \\ +3 \text{ x } 8^{0} & = & 3 \text{ x } 1 & = & \underline{3} \\ & & & & 331_{10} \end{array}$$

Fractional octal numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

8^{-1}	8^{-2}	8^{-3}	8^{-4} ...
1/8	1/64	1/512	1/4096

The octal number 0.4520 represents:

$$\begin{array}{rcrcrcr} 4\text{x}8^{-1} & = & 4\text{x}1/8 & = & 4/8 & = & 256/512 \\ +5\text{x}8^{-2} & = & 5\text{x}1/64 & = & 5/64 & = & 40/512 \\ +2\text{x}8^{-3} & = & 2\text{x}1/512 & = & 2/512 & = & \underline{2/512} \\ & & & & & & 298/512 = 149/256_{10} \end{array}$$

Arithmetic

ADDITION AND SUBTRACTION

Binary numbers are added according to the following rules:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ with a carry of 1

The addition of two binary numbers proceeds as follows (the decimal equivalents verify the result):

Augend	0111	(7)
Addend	+0100	+(4)
Partial Sum	0011	
Carry	1	
Sum	1011	(11)

Subtraction may be performed as an addition:

8 (minuend)		8 (minuend)
-6 (subtrahend)	or	+4 (10's complement of subtrahend)
2 (difference)		2 (difference - omit carry)

The second method shows subtraction performed by the "adding the complement" method. The omission of the carry in the illustration has the effect of reducing the result by 10.

One's Complement. The 3100 performs all arithmetic and counting operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in one's complement notation.

The one's complement representation of a number is found by subtracting each bit of the number from 1. For example:

1111	
-1001	9
0110	(one's complement of 9)

This representation of a negative binary quantity may also be obtained by substituting "1's" for "0's" and "0's" for "1's".

The value zero can be represented in one's complement notation in two ways:

$0000 \longrightarrow 00_2$	Positive (+) Zero
$1111 \longrightarrow 11_2$	Negative (−) Zero

The rules regarding the use of these two forms for computation are:

1 Both positive and negative zero are acceptable as arithmetic operands.

2 If the result of an arithmetic operation is zero, it will be expressed as positive zero.

One's complement notation applies not only to arithmetic operations performed in A, but also to the modification of execution addresses in the F register. During address modification, the modified address will equal 77777_8 only if the unmodified execution address equals 77777_8 and $b = 0$ or $(B^b) = 77777_8$.

MULTIPLICATION

Binary multiplication proceeds according to the following rules:

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Multiplication is always performed on a bit-by-bit basis. Carries do not result from multiplication, since the product of any two bits is always a single bit.

Decimal example:

multiplicand	14	
multiplier	12	
partial products	28	
	14	(shifted one place left)
product	168_{10}	

The shift of the second partial product is a shorthand method for writing the true value 140.

Binary example:

multiplicand (14)	1110	
multiplier (12)	1100	
partial products	0000	
	0000	shift to place
	1110	digits in proper
	1110	columns
product (168_{10})	10101000_2	

The computer determines the running subtotal of the partial products. Rather than shifting the partial product to the left to position it correctly, the computer right shifts the summation of the partial products one place before the next addition is made. When the multiplier bit is "1", the multiplicand is added to the running total and the results are shifted to the right one place. When the multiplier bit is "0", the partial product subtotal is shifted to the right (in effect, the quantity has been multiplied by 10_2).

DIVISION

The following examples shows the familiar method of decimal division:

```
                 14    quotient
divisor      13 |185   dividend
                 13
                 55    partial dividend
                 52
                  3    remainder
```

The computer performs division in a similar manner (using binary equivalents):

```
                    1110       quotient (14)
divisor      1101 | 10111001   dividend
                    1101
                     10100
                      1101
                       1110    partial dividends
                       1101
                         11    remainder (3)
```

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

Conversions

The procedures that may be used when converting from one number system to another are power addition, double dabble, and substitution.

Recommended Conversion Procedures (Integer and Fractional)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Double Dabble
Decimal to Octal	Double Dabble
Binary to Octal	Substitution
Octal to Binary	Substitution

GENERAL RULES

$r_i > r_f$: use Double Dabble, Substitution

$r_i < r_f$: use Power Addition, Substitution

r_i = Radix of initial system

r_f = Radix of final system

POWER ADDITION

To convert a number from r_i to $r_f (r_i < r_f)$ write the number in its expanded r_i polynomial form and simplify using r_f arithmetic.

EXAMPLE 1 Binary to Decimal (Integer)

$$\begin{aligned} 010\ 111_2 &= 1\ (2^4)\ +0(2^3) + 1(2^2) + 1(2^1) + 1(2^0) \\ &= 1\ (16)\ +0(8)\ +1(4)\ +1(2)\ +1(1) \\ &= 16 \quad +0 \quad +4 \quad +2 \quad +1 \\ &= 23_{10} \end{aligned}$$

EXAMPLE 2 Binary to Decimal (Fractional)

$$\begin{aligned} .0101_2 &= (2^{-1}) + 1(2^{-2}) + 0(2^{-3}) + 1(2^{-4}) \\ &= 0 \quad +1/4 \quad +0 \quad +1/16 \\ &= 5/16_{10} \end{aligned}$$

EXAMPLE 3 Octal to Decimal (Integer)

$$\begin{aligned} 324_8 &= 3(8^2) + 2(8^1) + 4(8^0) \\ &= 3(64) + 2(8) \ + 4(1) \\ &= 192 \ + 16 \ + 4 \\ &= 212_{10} \end{aligned}$$

EXAMPLE 4 Octal to Decimal (Fractional)

$$\begin{aligned} .44_8 &= 4(8^{-1}) + 4(8^{-2}) \\ &= 4/8 \quad + 4/64 \\ &= 36/64_{10} \end{aligned}$$

DOUBLE DABBLE

To convert a whole number from r_i to $r_f (r_i > r_f)$:

1. Divide r_i by r_f using r_i arithmetic
2. The remainder is the lowest order bit in the new expression
3. Divide the integral part from the previous operation by r_f
4. The remainder is the next higher order bit in the new expression
5. The process continues until the division produces only a remainder which will be the highest order bit in the r_f expression.

To convert a fractional number from r_i to r_f:

1. Multiply r_i by r_f using r_i arithmetic
2. The integral part is the highest order bit in the new expression
3. Multiply the fractional part from the previous operation by r_f
4. The integral part is the next lower order bit in the new expression
5. The process continues until sufficient precision is achieved or the process terminates.

EXAMPLE 1 Decimal to Binary (Integer)

45 ÷ 2 = 22	remainder 1; record	1
22 ÷ 2 = 11	remainder 0; record	0
11 ÷ 2 = 5	remainder 1; record	1
5 ÷ 2 = 2	remainder 1; record	1
2 ÷ 2 = 1	remainder 0; record	0
1 ÷ 2 = 0	remainder 1; record	1
Thus: $45_{10} = 101101_2$		101101

EXAMPLE 2 Decimal to Binary (Fractional)

.25 x 2 = 0.5;	record	0
.5 x 2 = 1.0;	record	1
.0 x 2 = 0.0;	record	0
Thus: $.25_{10} = .010_2$		.010

EXAMPLE 3 Decimal to Octal (Integer)

273 ÷ 8 = 34	remainder 1; record	1
34 ÷ 8 = 4	remainder 2; record	2
4 ÷ 8 = 0	remainder 4; record	4
		421
Thus: $273_{10} = 421_8$		

EXAMPLE 4 Decimal to Octal (Fractional)

.55 x 8 = 4.4;	record	4
.4 x 8 = 3.2;	record	3
.2 x 8 = 1.6;	record	1
-- --		-
-- --		-
		.431 . . .

Thus: $.55_{10} = .431\ldots_8$

SUBSTITUTION

This method permits easy conversion between octal and binary representations of a number. If a number in binary notation is partitioned into triplets to the right and left of the binary point, each triplet may be converted into an octal digit. Similarly each octal digit may be converted into a triplet of binary digits.

EXAMPLE 1 Binary to Octal

Binary =	110	000	.	001	010	
Octal =	6	0	.	1	2	

EXAMPLE 2 Octal to Binary

Octal =	6	5	0	.	2	2	7
Binary =	110	101	000	.	010	010	111

Fixed Point and Floating Point Numbers

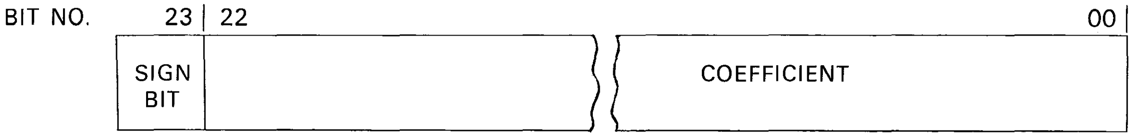
(The following information is for reference only and does not necessarily imply computer capability).

Any number may be expressed in the form kB^n, where k is a coefficient, B a base number, and the exponent n the power to which the base number is raised.

A fixed point number assumes:

1. The exponent n = 0 for all fixed point numbers.
2. The coefficient, k, occupies the same bit positions within the computer word for all fixed point numbers.
3. The radix (binary) point remains fixed with respect to one end of the expression.

A 3100 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any 3100 fixed point number designates the sign of the coefficient (23 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.



The radix (binary) point is assumed to be immediately to the right of the lowest order bit (00).

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of places shifted) is expressed as the power of the base. For example, $5{,}100{,}000_{10}$ may be expressed as 0.51×10^7, 0.051×10^8, 0.0051×10^9, etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are multiplied, the result is a number < 1. If the same two fractions are added, subtracted, or divided, the result may be greater than one and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

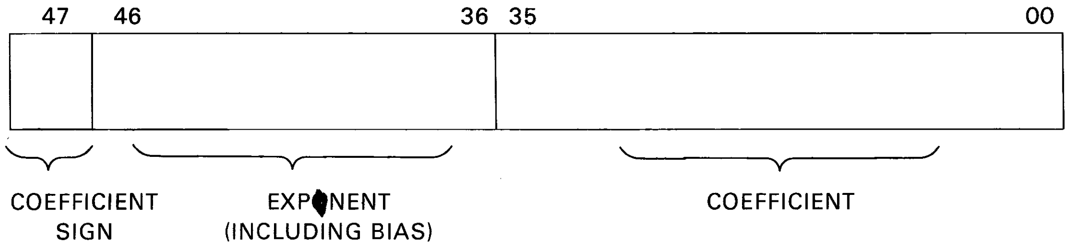
As an alternative to fixed point operation, a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

By shifting the radix point and increasing or decreasing the value of the exponent, widely varying quantities which do not exceed the capacity of the machine may be handled.

Floating point numbers within the computer are represented in a form similar to that used in "scientific" notation, that is, a coefficient or fraction multiplied by a number raised to a power. Since the computer uses only binary numbers, the numbers are multiplied by powers of two.

$F \bullet 2^E$ where: F = fraction
E = exponent

In floating point, different coefficients need not relate to the same power of the base as they do in fixed point format. Therefore, the construction of a floating point number includes not only the coefficient but also the exponent.



Coefficient. The coefficient consists of a 36-bit fraction in the 36 lower-order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than 1/2 but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent. The floating point exponent is expressed as an 11-bit quantity with a value ranging from 0000_8 to 3777_8. It is formed by adding a true positive exponent and a bias of 2000_8 or a true negative exponent and a bias of 1777_8. This results in a range of biased exponents as shown below.

True Positive Exponent	Biased Exponent	True Negative Exponent	Biased Exponent
+0	2000	-0	2000*
+1	2001	-1	1776
+2	2002	-2	1775
--	----	--	------
--	----	--	------
+1776	3776	-1776	0001
$+1777_8$	3777_8	-1777_8	0000_8

*Minus zero is sensed as positive zero by the computer and is therefore biased by 2000_8 rather than 1777_8.

The exponent is biased so that floating point operands can be compared with each other in the normal fixed point mode.

As an example, compare the unbiased exponents of $+52_8$ and $+0.02_8$ (Example 1).

EXAMPLE 1

	Number = $+52$	
0	0 0 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient
	Number = $+0.02$	
0	1 1 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

In this case $+0.02$ appears to be larger than $+52$ because of the larger exponent. If, however, both exponents are biased, (Example 2) changing the sign of both exponents makes $+52$ greater than $+0.02$.

EXAMPLE 2

	Number = $+52_8$	
0	1 0 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient
	Number = $+0.02_8$	
0	0 1 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

When bias is used with the exponent, floating-point operation is more versatile since floating-point operands can be compared with each other in the normal fixed point mode.

CONVERSION PROCEDURES

Fixed Point to Floating Point

1. Express the number in binary.
2. Normalize the number. A normalized number has the most significant 1 positioned immediately to the right of the binary point and is expressed in the range $1/2 \leq k < 1$.
3. Inspect the sign of the true exponent. If the sign is positive add 2000_8 (bias) to the true exponent of the normalized number. If the sign is negative add the bias 1777_8 to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.
4. Assemble the number in floating point.
5. Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive the assembled floating point number is the true representation.

EXAMPLE 1 Convert $+4.0$ to floating point

1. The number is expressed in octal.
2. Normalize. $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$.
3. Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = $2000 + 3$.
4. Assemble number in floating point format.
   Coefficient = $400\ 000\ 000\ 000_8$
   Biased Exponent = 2003_8
   Assembled word = $2003\ 400\ 000\ 000\ 000_8$

5 Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

EXAMPLE 2 Convert -4.0 to floating point

1 The number is expressed in octal.

2 Normalize. -4.0 = -4.0 x 8^0 = -0.100 x 2^3

3 Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.

4 Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased Exponent = 2003_8
Assembled word = 2003 400 000 000 000_8

5 Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for
-4.0 = 5774 377 777 777 777_8

EXAMPLE 3 Convert 0.5_{10} to floating point

1 Convert to octal. 0.5_{10} = 0.4_8

2 Normalize. 0.4 = 0.4 x 8^0 = 0.100 x 2^0

3 Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 0.

4 Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased Exponent = 2000_8
Assembled word = 2000 400 000 000 000_8

5 Since the sign of the coefficient is positive, the floating point representation of +0.5_{10} is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777_8 instead of 2000_8 because of the negative exponent. The 3200, however, recognizes -0 as +0 and biases the exponent by 2000_8.

EXAMPLE 4 Convert 0.04_8 to floating point

1 The number is expressed in octal.

2 Normalize. 0.04 = 0.04 x 8^0 = 0.4 x 8^{-1} = 0.100 x 2^{-3}

3 Since the sign of the true exponent is negative, add 1777_8 (bias) to the true exponent. Biased exponent = 1777_8 + (-3) = 1774_8.

4 Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased Exponent = 1774_8
Assembled word = 1774 400 000 000 000_8

5 Since the sign of the coefficient is positive, the floating point representation of 0.04_8 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point Format

1 If the floating point number is negative, complement the entire floating point word and record the fact that the quantity is negative. The exponent is now in a true biased form.

2 If the biased exponent is equal to or greater than 2000_8 subtract 2000_8 to obtain the true exponent. If less than 2000_8 subtract 1777_8 to obtain true exponent.

3 Separate the coefficient and exponent. If the true exponent is negative the binary point should be moved to the left the number of bit positions indicated by the true exponent. If the true exponent is positive, the binary point should be moved to the right the number of bit positions indicated by the true exponent.

4 The coefficient has now been converted to fixed binary. The sign of the coefficient will be negative if the floating point number was complemented in step one. (The sign bit must be extended if the quantity is placed in a register.)

5 Represent the fixed binary number in fixed octal notation.

EXAMPLE 1 Convert floating point number 2003 400 000 000·000_8 to fixed octal

1 The floating point number is positive and remains uncomplemented.

2 The biased exponent > 2000_8, therefore subtract 2000_8 from the biased exponent to obtain the true exponent of the number. 2003 - 2000 = +3

3 Coefficient = 400 000 000 000_8 = $.100_2$. Move binary point to the right 3 places. Coefficient = 100.0_2.

4 The sign of the coefficient is positive because the floating point number was not complemented in step one.

5 Represent in fixed octal notation.
100.0 x 2^0 = 4.0 x 8^0.

EXAMPLE 2 Convert floating point number 5774 377 777 777 777_8 to fixed octal

1 The sign of the coefficient is negative, therefore, complement the floating point number.
Complement = 2003 400 000 000 000_8

2 The biased exponent (in complemented form) > 2000_8, therefore subtract 2000_8 from the

biased exponent to obtain the true exponent of the number. 2003 - 2000 = +3

3 Coefficient = 4000 000 000 000$_8$ = 0.100$_2$
Move binary point to the right 3 places.
Coefficient = 100.0$_2$

4 The sign of the coefficient will be negative because the floating point number was originally complemented.

5 Convert to fixed octal. -100.0$_2$ = -4.0$_8$

EXAMPLE 3 Convert floating point number
1774 400 000 000 000$_8$
to fixed octal

1 The floating point number is positive and remains uncomplemented.

2 The biased exponent < 2000$_8$, therefore subtract 1777$_8$ from the biased exponent to obtain the true exponent of the number. 1774$_8$ - 1777$_8$ = -3

3 Coefficient = 400 000 000 000$_8$ = .100$_2$
Move binary point to the left 3 places.
Coefficient = .000100$_2$

4 The sign of the coefficient is positive because the floating point number was not complemented in step one.

5 Represent in fixed octal notation.
.000100$_2$ = .04$_8$

Appendix D

Table of Powers of Two

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

Appendix E

Octal-Decimal Integer Conversion Table

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

0000	0000
to	to
0777	0511
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

1000	0512
to	to
1777	1023
(Octal)	(Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE

2000 to 2777 (Octal) — 1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2100	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1519	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

3000 to 3777 (Octal) — 1536 to 2047 (Decimal)

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

4000	2048
to	to
4777	2559
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

5000	2560
to	to
5777	3071
(Octal)	(Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE

6000	3072
to	to
6777	3583
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

7000	3584
to	to
7777	4095
(Octal)	(Decimal)

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3496	3497	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

Appendix F

Octal-Decimal Fraction Conversion Table

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

Appendix G

Definition of I/O Interface Signals

DEFINITION OF I/O INTERFACE SIGNALS

This appendix defines the signals that are exchanged between the 3106 Data Channel and external equipment. There are three classes of signals: bidirectional, 3106 to external equipment, and external equipment to 3106.

Bidirectional Signals	
Data Bits	The 12 lines which carry data are bi-directional, and perform as follows: 1. In a Read (input) operation, data is transmitted from the external equipment to the 3106. 2. In a Write (output) operation, data is transmitted from the 3106 to the external equipment. 3. The Connect and Function codes are transmitted from the 3106 to the external equipment via the 12 data lines.
Parity Bit	A parity bit accompanies each 12 bits of data transmitted between the 3106 and external equipment. Odd parity is used; thus the total number of "1's" transmitted is always an odd number.
3106 to External Equipment	
Read	Static "1" signal produced by 3106 during a Read operation.
Write	Static "1" signal produced by 3106 during a Write operation.
Connect	Static "1" signal sent to external equipment when 12-bit Connect code is available on data lines. Signal drops when external equipment returns Reply or Reject.
Function	Static "1" signal sent to external equipment when 12-bit Function code is available on data lines. Signal drops when external equipment returns Reply or Reject.
Data Signal	Static "1" signal sent to external equipment during both Read and Write operations. Signal drops conditionally when Reply is received from external equipment. 1. In a Read operation, Data Signal indicates that 3106 is ready to accept a 12-bit word from external equipment. 2. In a Write operation, Data Signal indicates the 3106 has placed a 12-bit word on the data lines.

3106 to External equipment (Cont.)	
Master Clear	"1" signal from computer which returns channel and external equipment to zero initial conditions and disconnects external equipment.
Computer Running	Static "1" when computer is operating.
External Equipment to 3106	
Reply	Static "1" signal produced by external equipment in response to a Connect, Function, or Data Signal. Signal drops when Connect, Function, or Data Signal drops. 1. If connection can be made when Connect signal is received, external equipment connects and returns a Reply. 2. If specified function can be performed when Function signal is received, external equipment initiates function and returns a Reply. 3. In a Read operation, external equipment sends a Reply as soon as it has placed a 12-bit word on the data lines in response to the Data Signal. 4. In a Write operation, external equipment sends a Reply as soon as it samples the data lines in response to the Data Signal.
Reject	Static "1" signal produced by external equipment in response to a Connect or Function signal, if the connection cannot be made or the function cannot be performed at the time that the external equipment receives the respective signal.
End of Record	Static "1" signal produced by external equipment during a Read operation. This signal is produced in response to the Data Signal, if the end of the specified block of data has been reached.
Parity Error	Static "1" signal produced if the total number of "1's" in the 12 data bits plus the parity bit is not an odd number.
Status Bits	The external equipment uses the 12 status lines to indicate its condition.
Interrupt Lines	A "1" signal on an interrupt line indicates that an external equipment has reached a predetermined condition. A 3106 may communicate with a maximum of 8 external equipments, and each external equipment uses 1 interrupt line.

Appendix H

3100 System Character Set

3100 SYSTEM CHARACTER SET

Char.	Int. BCD	Ext. BCD	Holl.	Char.	Int. BCD	Ext. BCD	Holl.
0	00	12	0	P	47	47	11-7
1	01	01	1	Q	50	50	11-8
2	02	02	2	R	51	51	11-9
3	03	03	3	S	62	22	0-2
4	04	04	4	T	63	23	0-3
5	05	05	5	U	64	24	0-4
6	06	06	6	V	65	25	0-5
7	07	07	7	W	66	26	0-6
8	10	10	8	X	67	27	0-7
9	11	11	9	Y	70	30	0-8
A	21	61	12-1	Z	71	31	0-9
B	22	62	12-2	=	13	13	3-8
C	23	63	12-3	-(dash)	14	14	4-8
D	24	64	12-4	+	20	60	12
E	25	65	12-5	+0	32	72	12-0
F	26	66	12-6	.	33	73	12-3-8
G	27	67	12-7	)	34	74	12-4-8
H	30	70	12-8	-(minus)	40	40	11
I	31	71	12-9	-0	52	52	11-0
J	41	41	11-1	$	53	53	11-3-8
K	42	42	11-2	*	54	54	11-4-8
L	43	43	11-3	(space)	60	20	blank
M	44	44	11-4	/	61	21	0-1
N	45	45	11-5	,	73	33	0-3-8
O	46	46	11-6	(	74	34	0-4-8

Appendix I

Peripheral Equipment Codes

Peripheral Equipment

FUNCTION AND STATUS RESPONSE CODES

The following tables list the function and status response codes for the 3248/405 Card Reader and the 322X and 362X Magnetic Tape Controllers.

Function and status response codes for other 3200 peripheral equipments can be found in the reference manuals of these equipments.

3248/405 CARD READER CODES

FORMAT AND OPERATIONAL FUNCTION CODES

0001	Negate Hollerith to Internal BCD Conversion
0002	Release Negate Hollerith to Internal BCD Conversion
0004	Gate Card to Secondary Station
0005	Function Clear

INTERRUPT FUNCTION CODES

0020	Interrupt on Ready and Not Busy
0021	Release Interrupt on Ready and Not Busy
0022	Interrupt on End of Operation
0023	Release Interrupt on End of Operation
0024	Interrupt on Abnormal End of Operation
0025	Release Interrupt on Abnormal End of Operation

STATUS REPLIES

XXX1	Reader Ready
XXX2	Reader Busy
XXX4	Binary Card
XX1X	End of File (Card)
XX2X	Stacker Full or Jammed or Fail to Feed
XX4X	Hopper Empty
X1XX	End of File (Switch)
X2XX	Interrupt—Ready and Not Busy
X4XX	Interrupt—End of Operation
1XXX	Interrupt—Abnormal End of Operation
2XXX	Read Compare or Pre-read Error

322X AND 362X MAGNETIC TAPE CONTROLLERS

FORMAT FUNCTION CODES

0000	Release
0001	Binary
0002	Coded
0003	556 BPI Density
0004	200 BPI Density
0006	800 BPI Density
0005	Clear
0041	Reverse Read
0040	Clear Reverse Read

TAPE MOTION FUNCTION CODES

0010	Rewind
0011	Rewind Unload
0012	Backspace
0013	Search End of File Mark Forward
0014	Search End of File Mark Backward
0015	Write End of File Mark
0016	Skip Bad Spot

INTERRUPT FUNCTION CODES

0020	Interrupt On Ready and Not Busy
0021	Release Interrupt on Ready and Not Busy
0022	Interrupt on End of Operation
0023	Release Interrupt on End of Operation
0024	Interrupt on Abnormal End of Operation
0025	Release Interrupt on Abnormal End of Operation

STATUS REPLIES

XXX1	Ready
XXX2	Read/Write Control and/or Busy
XXX4	Write Enable
XX1X	File Mark
XX2X	Load Point
XX4X	End of Tape
X1XX	Density ("1" in bit 6 indicates 556 BPI, "0" in bit 6 and bit 7 indicates 200 BPI)
X2XX	Density ("1" in bit 7 indicates 800 BPI)
X4XX	Lost Data
1XXX	End of Operation
2XXX	Transverse or Longitudinal Parity Error

COMMENT SHEET

CONTROL DATA 3100 COMPUTER SYSTEM
PRELIMINARY REFERENCE MANUAL
PUB. NO. 60108400

FROM NAME : ______________________________

BUSINESS
ADDRESS : ______________________________

COMMENTS: (DESCRIBE ERRORS, SUGGESTED ADDITION OR DELETION AND INCLUDE PAGE NUMBER, ETC.)

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
8100 34TH AVENUE SOUTH
MINNEAPOLIS 20, MINNESOTA

ATTN: TECHNICAL PUBLICATIONS DEPT.
COMPUTER DIVISION
PLANT TWO

CUT ALONG LINE

FOLD FOLD

STAPLE STAPLE

CONTROL DATA SALES OFFICES

ALAMOGORDO • ALBUQUERQUE • ATLANTA • BOSTON • CAPE CANAVERAL
CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS • DAYTON
DENVER • DETROIT • DOWNEY, CALIF. • HONOLULU • HOUSTON • HUNTSVILLE
ITHACA • KANSAS CITY, KAN. • LOS ANGELES • MINNEAPOLIS • NEWARK
NEW ORLEANS • NEW YORK CITY • OAKLAND • OMAHA • PALO ALTO
PHILADELPHIA • PHOENIX • PITTSBURGH • SACRAMENTO • SALT LAKE CITY
SAN BERNARDINO • SAN DIEGO • SEATTLE • WASHINGTON, D.C.

INTERNATIONAL OFFICES

FRANKFURT, GERMANY • HAMBURG, GERMANY • STUTTGART, GERMANY
GENEVA, SWITZERLAND • ZURICH, SWITZERLAND • CANBERRA, AUSTRALIA
MELBOURNE, AUSTRALIA • SYDNEY, AUSTRALIA • ATHENS, GREECE
LONDON, ENGLAND • OSLO, NORWAY • PARIS, FRANCE • STOCKHOLM, SWEDEN
MEXICO CITY, MEXICO, (REGAL ELECTRONICA DE MEXICO, S.A.)
OTTAWA, CANADA, (COMPUTING DEVICES OF CANADA, LIMITED) • TOKYO, JAPAN,
(C. ITOH ELECTRONIC COMPUTING SERVICE CO., LTD.)

CONTROL DATA
CORPORATION
8100 34th AVENUE SOUTH, MINNEAPOLIS, MINNESOTA 55440

Pub. No. 60108400

Litho in U