

MASS REFERENCE MANUAL

CONTROL DATA[®]
MP-32
COMPUTER SYSTEMS

	REVISION RECORD
REVISION	DESCRIPTION
01	Original release describing the general operation of the MP
(76-05-21)	Micro ASSembler (MASS) and providing necessary instruction
	for preparing programs for assembly.
02	This manual is being reissued to clarify and elaborate on
(77-08-15)	use of the Micro ASSembler (MASS) to generate binary micro-
	code to be used on the MP-32 and on the Micro Simulator
	(MPSIM) which is used to debug MP-32 microcode.
03	Remove references to the Micro Simulator (MPSIM), since the
(82-12-30)	software no longer exists.
Document No. 17328900	

Revision letters I, O, Q, and X are not used.

COPYRIGHT 1976, 1977, 1982

CONTROL DATA CORPORATION

All Rights Reserved

Printed in the United States of America

CONTROL DATA CORPORATION

SYSTEMS TECHNOLOGY DIVISION

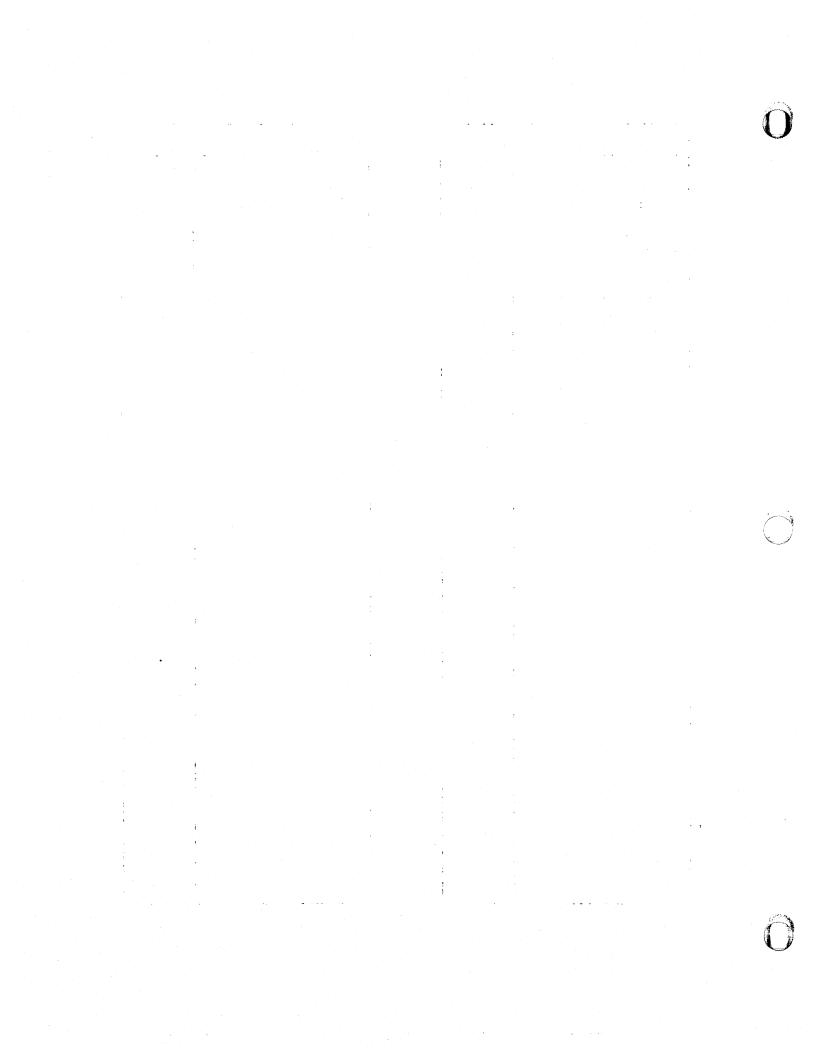
215 Moffett Park Drive

Sunnyvale, California 94086

LIST OF EFFECTIVE PAGES

Page	Revision
Cover Title Page ii through xii 1-1 through 1-2 2-1 through 2-10 3-1 through 3-22 4-1 through 4-6 A-1 through A-19 B-1 through B-2 C-1 through C-2 D-1 through D-2 E-1 through E-2 E-3 through E-4 F-1 through F-4 G-1 through G-6 H-1 through H-4 Glossary 1 through Glossary 6 Comment Sheet	- 02 02 02 02 02 02 02 02 02 01 02 02 02 02 02

Page	Revision
•	·
	·
	1



PREFACE

The MP Micro ASSembler (MASS) is a component of the MP-60 Software package. It is intended to assemble microcode for the CDC Microprogrammable Processor, herein called the MP. The assembler operates under control of the CYBER 170/70L/6000 NOS/BE 1.0 Operating System and the CYBER 170/70L/6000 NOS 1.0 Operating System. A separate version of the MP Micro Assembler executes on the MP-60 Series computer.

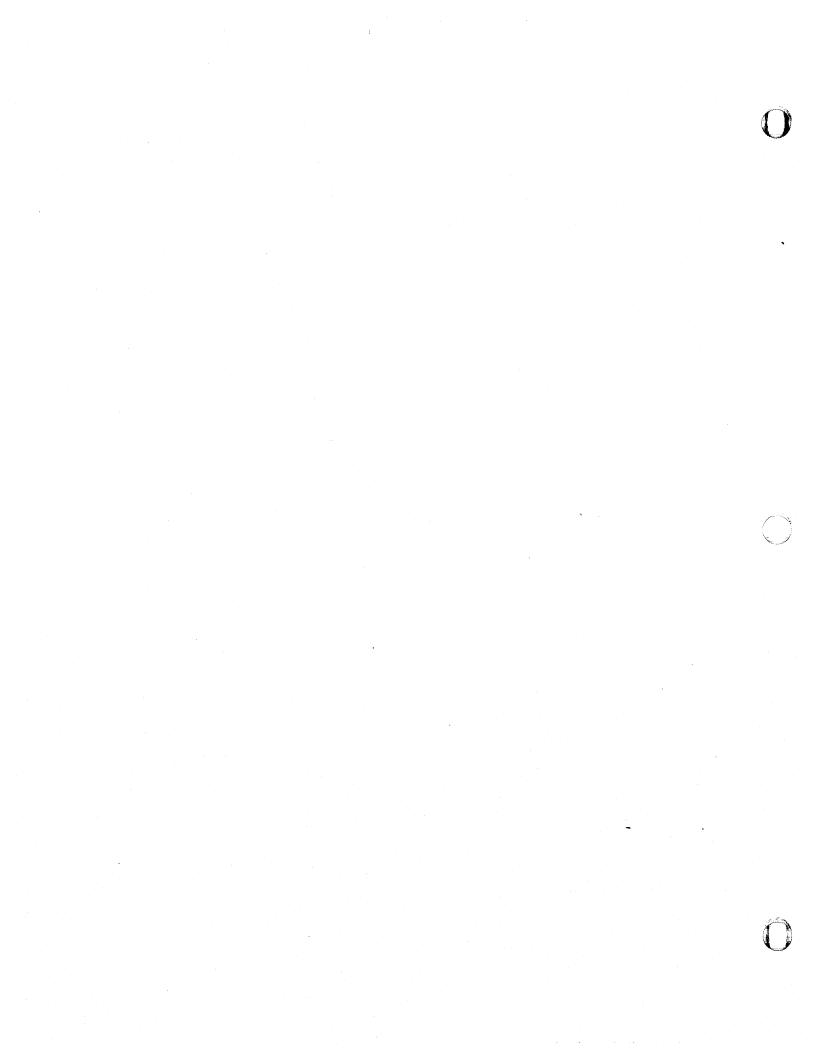
This assembler manual is to be used in conjunction with the Basic System Controller Model 65109 Hardware Reference Manual (see list of publications below). This manual will describe the general operation of the assembler and provide necessary instruction for preparing programs for assembly. No attempt is made here to provide a programmer's guide; therefore, examples will be limited. It is assumed that the reader is already familiar with the operation of the MP computer. A detailed description of the MP computer may be found in the above-mentioned Basic System Controller Model 65109 Hardware Reference Manual.

Following is a list of related manuals:

<u>Manuals</u>	CDC Publication No.
NOS 1.0 Reference Manual Vol. I	60435400
NOS 1.0 Reference Manual Vol. II	60445300
NOS/BE 1.0 Reference Manual	60493800
Basic System Controller Model 65109 Hardware Reference Manual	41618400

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

17328900-02 v/vi



CONTENTS

Section	<u>Title</u>		
1	INTRODUCTION		
f			
2	LANGUAGE STRUCTURE	2-1	
	Statement Format	2-1	
	Q Field	2-1	
	Location Field	2-4	
	Symbols	2-4	
	Constants	2-5	
	Decimal Constants	2-6	
	Octal Constants	2-6	
	Hexadecimal Constants	2-6	
	Pseudo Instructions	2-7	
	Mnemonic Instructions	2-7	
3	PSEUDO INSTRUCTIONS	3-1	
	Assembler Control	3-1	
	IDENT	3-1	
	TITLE	3-1	
	END	3-2	
	FINIS	3-2	
	OPTN	3-2	
	FILE	3-4	
	ENDF	3-4	
	Listing Control	3-4	
	EJECT	3-5	
	SPACE	3-5	
	BOX	3-5	
	EBOX	3-6	
	NOLIST	3-6	
	LIST	3-6	
	REM	3-7	

Section	<u>Title</u>	Page
	Memory Management and Symbol Definition	3-7
	EQU	3-8
	SET	3-9
	ORG	3-9
	Data Definition	3-10
	OCT	3-10
	DEC	3-11
	HEX	3-11
	VFD	3-12
	Programming Information	3-12
	CMP1	3-13
	CMP2	3-13
	Conditionals	3-13
	ENDIF	3-14
	IZD	3-14
	IND	3-14
	IGD	3-14
	ILD	3-15
4	MNEMONIC MICRO INSTRUCTIONS	4-1
•	MP Machine Organization	4-1
	Microinstruction Formats	4-1
	F Field	4-4
	ALU Operations	4-4
	Logical Operations	4-5
	Arithmetic Operations	4-5
	Shift Operations	4-8
	Scale Operations	4-11
	A, B and D Fields	4-13
	A Field	4-13
	B Field	4-15
	D Field	4-17

Section	<u>Title</u>	Page
	Examples of F, A, B, and D Fields	4-19
	S Field	4-19
· _	Examples	4-21
	C Field	4-22
	Jump Instructions	4-22
	Flag Setting and Clearing Commands	4-24
	Register Setting Instructions	4 - 25
	Register Control Commands	4 - 25
	Selector 3 (Shift Commands)	4-26
	Transforms	4-27
	Other C Codes	4-30
	Example of C Field Coding	4-31
	M Field	4-31
	T Field	4 - 36
	Assembler Processing of M and T Fields	4-38
	Sequential Addressing	4 - 38
	Jump Addressing	4-38
	Return Addressing	4-40
5 M/	AIN MEMORY INTERFACE	5-1
	Memory Accessing	5-1
	Page Index File	5-1
	Memory Page Addressing	5 - 3
	Page Status	5-3
	Memory Protect Logic	5-3
	Program State Registers	5 - 5
	CPU State Register	5-5
	DMA State Registers	5 - 7
	Coding Examples for Memory Interface	5-8

<u>Section</u>	<u>Title</u>	<u>Page</u>	
	AP PE NDI CES		
A	MASS Control Card	A-1	
В	Format of Deadstart Output Deck	B-1	
C	Default Codes Generated for Blank Fields	C-1	

Section	Title	Page		
D	Selecting Nonconflicting Mnemonics			
E	Mass Output Listing			
F	Sample Listing Including Map and Cross Reference			
G	Assembly Diagnostics	F-1 G-1		
Н	Microinstruction Timing	H-1		
	FIGURES			
	•			
2-1	MP Coding Form	2-3		
4-1	Central Processing Unit Detailed Block Diagram	4-2		
4-2	Microinstruction Formats	4-3		
4-3	Examples of Logical Operations	4-7		
4-4	Example of an Overflow Operation and Timing	4-7		
4-5	Examples of Shift Operations	4-10		
4-6	Examples of Scale Operations	4-12		
4-7	Examples of the Use of F, A, B and D Fields	4-20		
4-8	Example of S Field Coding	4-23		
4-9	Examples of S and C Field Conflicts	4-32		
4-10	Example of Multiply Coding	4-33		
4-11	Example of Divide Coding	4-34		
4-12	Example of Sequential Addressing	4-39		
4-13	Further Examples of Sequential Addressing	4-39		
4-14	Examples of Jump Addressing	4-42		
4-15	Examples of Return Addressing	4-42		
5-1	Creation of the Actual Physical Address	5-2		
5-2	Page Index File Entry Bit Assignments	5-4		
5-3	CPU State Register Bit Assignments	5-7		
5-4	Examples of Reading and Writing a Page Index File	5-9		
5-5	Examples of Reading and Writing State Registers	5-10		
5-6	Example of Reading Location O Using State O	5-11		
5-7	Example of Reading Location O Using State 2 and	5-12		
	Page ladox File			

17328900-02 xi

Section	<u>Title</u>		
5-8	Example of Placing a Location's Address in the	5-13	
	Location		
5-9	TK Transform/Lookahead Example	5-14	
5-10	TMA Transform/Interrupt Example	5-16	
D-1	Examples of Conflicting Mnemonic Selection and	D-2	
	Assembler Error Codes		
F-1	Sample Listing	F-2	
F-2	Map and Cross Reference	F-3	
G-1	Assembler Diagnostics Example	G-5	
H-1	Microinstruction Classification	H-3	
	TABLES		
2-1	Source Statement Fields	2-2	
2-2	Summary of Instructions and Bit Settings	2-8	
B-1	Deadstart Deck Format	B-2	
Clossary		G-1	



The assembler for the CONTROL DATA Microprogrammable Processors provides the mnemonic language necessary for the programmer to write a microprogram. The assembler translates symbolic source program instructions into object machine instructions and provides a listing of assembly results.

The characteristics of the assembler as written for the CYBER 170/70L/6000 and MP-60 Series computers are described. This assembler is based on the MICRO-71 assembler for the MPP computer.

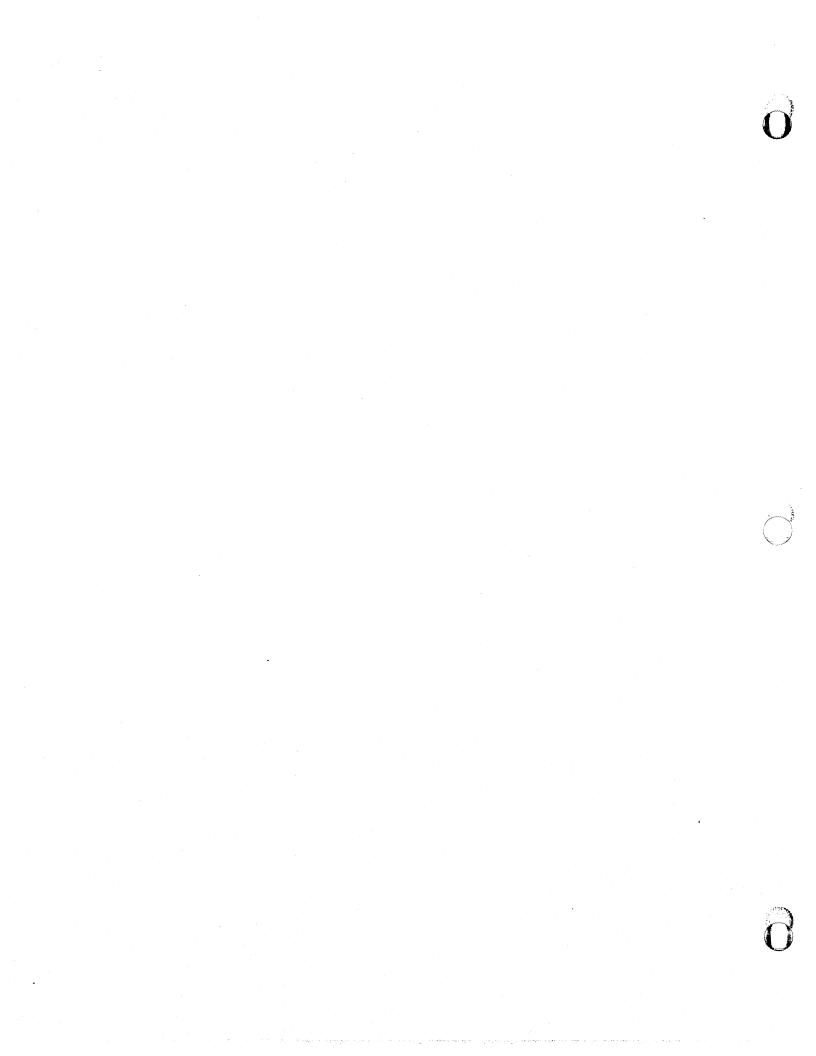
Input to this assembler consists of one or more source programs followed by a FINIS card. Each program begins with an IDENT card and is terminated with an END card. Each program is coded using these basic elements:

- Symbols
- Constants
- Pseudo Instructions
- Mnemonic Instructions

The basic elements are punched into a card in specific fields, always left-justified within the field.

Output from the assembler consists of the following:

- Assembly listing including diagnostics
- Zero location map
- Deadstart object time
- User generate micro binary



STATEMENT FORMAT

A source input statement to the MP assembler (MASS) consists of 11 fields, as contained in Table 2-1 and as illustrated in the coding form depicted in Figure 2-1. Of these fields, the Q (qualifier), location, and comment fields are used to improve the documentation of the assembled microinstructions, while the remaining eight fields correspond to the eight fields of the MP microinstruction shown in the Basic System Controller Model 65109 Hardware Reference Manual. The eight fields used on the input form are in the same order that the programmer will tend to use in preparing microinstructions for a microprogram.

Information entered in each field (if anything is entered) is entered left-justified with blank fill. Information which is not entered left-justified will not be processed correctly by the assembler.

Q FIELD

The Q field is column 1 of the input coding form. This field may contain an asterisk (*), dollar sign (\$), plus sign (+), minus sign (-), or it may be blank.

An asterisk (*) or dollar sign (\$) specifies that the rest of the input source statement is a remark and that the remaining 79 columns contain comments. This qualifier allows the remarks card to be printed on the listing with no effect on the assembler object code output. One line is skipped before and after a single comment card or group of comment cards.

A plus sign (+) in the Q field locates the resulting microinstruction as the upper instruction of a microinstruction pair.

A minus sign (-) in the Q field locates the resulting microinstruction as the lower instruction of a microinstruction pair.

A blank in the Q field locates the resulting microinstruction in the next available half of a microinstruction pair.

17328900-02

TABLE 2-1. SOURCE STATEMENT FIELDS

FIELDS	COLUMNS	COMMENTS
Q	1	The qualifier field may specify whether the (qualifier) statement is a comment, an upper instruction, or lower instruction.
Location	2 through 9	The location field specifies the statement's symbolic address in this program.
F (function)	11 through 16	The function field specifies a logical, arithmetic, shift or scale operation that is performed by the arithmetic and logic unit (ALU) on two sources and placed in a destination.
A	17 through 22	Specifies the A source of the function.
В	23 through 28	Specifies the B source of the function.
D	29 through 34	Specifies the destination of the result of the ALU.
s	35 through 40	The special field provides special instruction modes that either:
	·	• Extend the A, B, and D fields, or
		• provide a special command which is performed in parallel with the data transfers taking place in the ALU.
C (constant)	41 through 49	The constant field specifies another special command that is performed independently of the rest of the instruction; it is executed in parallel with the rest of the instruction.
М	50	The mode field specifies the addressing method for obtaining the next instruction pair: sequential, jump or return.
T (test)	51 through 55	The test field is the conditional branch of the instruction and specifies which instruc- tion (upper or lower) of the next instruction pair to execute. The test and branch are executed after the rest of the instruction has executed.
Comment	56 through 80	The comment field is used for remarks that are printed as part of the list output.

Figure 2-1. MP Coding Form

17328900-02 2-3

Each micromemory location (address) contains space for two microinstructions. These are referred to as the upper and the lower microinstruction. If column 1 is blank, the statement is assigned to the next available microinstruction position (either upper or lower). If column 1 contains a plus (+) sign or any character other than -, \$, *, or blank, that statement is assigned to the next available upper microinstruction location. If column 1 contains a minus (-) sign, that statement is assigned to the next available lower microinstruction location. The use of + and - results in possibly skipping an available microinstruction location. The skipped location is assembled as all zeros and not listed on the listing.

LOCATION FIELD

The location field is in columns 2 through 9 of the input coding form. This field may be left blank or it may contain a symbol. If a symbol is included in the field, it must be entered left-justified and follow the definition of a symbol.

A symbol in the location field is assigned the value of the location of the corresponding microinstruction or constant, or the value of the expression starting in columns 17 with a SET or EQU pseudo instruction.

A symbol in the location field of a microinstruction takes on the upper/lower quality of the actual microinstruction location. This quality is used in coding jumps in the C field of a microinstruction.

SYMBOLS

A symbol is a set of characters that identifies a value and its associated qualities. A symbol can be a maximum of eight characters. A symbol must contain a non-numeric character to distinguish it from a constant. A symbol cannot include the following characters:

, = + - * / blank

- 0
- The process of associating a symbol with a value and qualities is known as symbol definition. This can occur in two ways:
 - 1. A symbol used in the location field (columns 2 through 9) of a symbolic machine instruction or certain pseudo instructions is defined as an address having the current value of the location counter and having an upper or lower quality.
 - 2. A symbol used in the location field of definition pseudo instructions EQU and SET is defined as having the value and quality derived from an expression starting in column 17 of the instruction. The SET pseudo instruction assigns an attribute of redefinability to a symbol. Unless a symbol was previously defined with a SET instruction, a second attempt to define it with a different value produces a duplicate definition fatal error flag.

A symbol may be used in the location field (columns 2 through 9), the S field (columns 35 through 40), or in the C field (columns 41 through 49) of a microinstruction on the input form.

A symbol is undefined when it has never appeared in the location field, or if it is equated to an undefined symbol. The assembler identifies the use of undefined symbols on the assembly listing.

Examples:

HCNYL	Legal
TAG	Legal
1234	Illegal (will be interpreted as a constant)
*12.3	Illegal (contains an asterisk)
XYZ/3P	Illegal (contains a slash)
ß=3	Illegal (contains an equal sign)

CONSTANTS



Constants are used to represent numbers and may be used in the S and C fields of the input form. Constants may also be used on the right side

of the several pseudo instructions. The MP assembler recognizes three types of numeric constants: decimal, octal and hexadecimal. Decimal constants have no suffix, octal constants have B as a suffix, and hexadecimal constants have X as a suffix. The numeric constant is represented by a string of digits within the number base of the constant and is always a whole integer number. Constants must fit in the field's length except with the VFD pseudo.

DECIMAL CONSTANTS

A decimal constant consists of a string of decimal digits. If the constant is larger than the field width within the microinstruction, the high order bits will be discarded.

Examples:

999	Legal
98A	Illegal (contains an alphabetic character)
12.1	Illegal (contains a decimal point)

OCTAL CONSTANTS

An octal constant consists of a string of octal digits that are suffixed with the letter B.

Examples;

123B	Legal
77B	Legal
019B	Illegal (contains a non-octal digit)

HEXADECIMAL CONSTANTS

A hexadecimal constant consists of a string of hexadecimal digits and is suffixed with the letter X. The hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Examples:

77BX	Legal
1GX	Illegal (contains a non-hexadecimal digit)
ABC	Illegal (has no X suffix)
77B	Will be interpreted as an octal 77

PSEUDO INSTRUCTIONS

Pseudo instructions direct the MP assembler to perform specific functions. They do not generate MP instructions. They define assembler control, listing control, data definition, and other operations. Pseudo instructions are defined in Section 3 of this manual.

MNEMONIC INSTRUCTIONS

Mnemonic instructions allow the programmer to use convenient names to specify the binary information to be inserted in each field of MP micro-instruction. The list of mnemonic instructions recognized by the MP assembler for each field of the instruction is given in Table 2-2. Detailed usage of the mnemonic instructions is given in Section 4 of this manual.

Also allowed are user definable opcodes that can be used in place of the assembler opcodes. MASS will try to match the opcodes first from the symbol table and then from the opcode set in MASS.

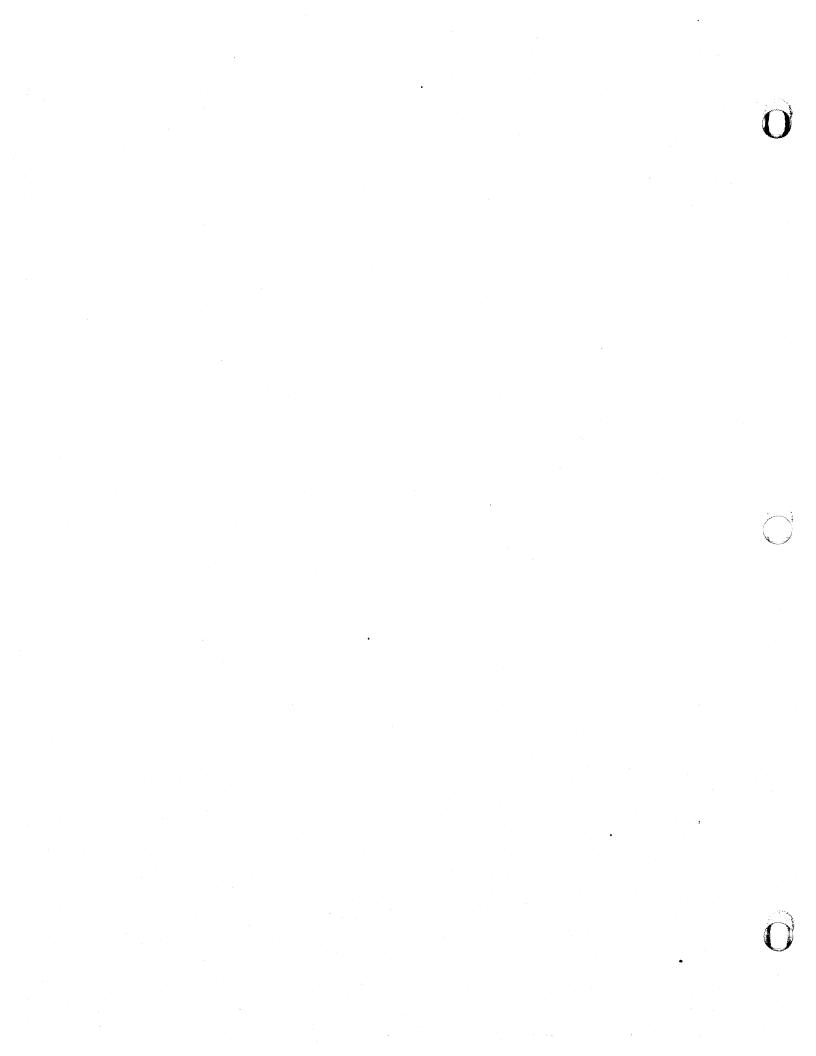
Whether or not a user-defined opcode is illegal or not is dependent on which field the opcode is to be used. Each field has associated with it a mask to define the bit fields. The masks for the fields are shown as follows:

C 1.FF		0.00
0	M R R R R R R R R R R R R R R R R R R R	/K L/K L/C L/C L/C L/C L/C L/C L/C L/C
• F k. I • I • I		
S	NOP RD2 RPT RD3 RD1 LBEA FZWR APDP OPP GATEI HALT RTJ CLRNP	
. E. 8F		
07.F	012E 4 5 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
89 · 1 · 3 · 1 · 1 · 1 · 1 · 1 · 1 · 1 · 1	2000	
A .1CF	0000044440000	
3FFF	000 000 000 000 000 000 000 000 000 00	3000 3008 3010 3040 3050 3680 3690
<u>.</u>	A A + - B A A + - B A A - B B B B B B B B B B B B B B B B B B B	AROE ARSE AREA AGROE AGREA SLOE SLOE SDLOE SDLOE
C	# # # # # # # # # # # # # # # # # # #	

FIELD	MASK
М	C0001FFF
F	3FFF0000
A	01C00F00
В	00380FFF
D	00070F00
T	0000E08F
S	00000F00
C	C00010FF

The user-defined opcode is usually established with an EQU and the bits must fit within the mask or an error will result.

Example: ROTE EQU 11110000X -- legal only in the F Field



Pseudo instructions are instructions to the MP assembler and normally do not result in any microcode output (the only exceptions are the HEX, DEC, OCT and VFD pseudo instructions). A pseudo instruction consists of the pseudo operation code, which is coded in the F field of the input form (columns 11 through 16), plus additional information coded in the other fields of the input form. The detailed field usage is given under each pseudo instruction.

ASSEMBLER CONTROL

These pseudo instructions define and control the operation of the MP assembler, but do not generate code in the object program.

IDENT

This pseudo instruction provides program identification and must be used as the first instruction of each program. Text in columns 17 through 71 of the IDENT card is printed as the first line at the top of each page of the output listing. The location field is ignored.

Example:

11	17							
IDENT	CDC	844	DISK	FILE	CONTROLLER	EMULATION	FOR	MP

TITLE

This pseudo instruction provides page heading information and may be used anywhere in the program. The TITLE pseudo causes a page eject and a new heading to be printed at the top of the page. This heading consists of columns 17 through 23 from the IDENT card, followed by one space and the contents of columns 17 through 63 from the TITLE card.

3-1

Example:

11	17	_				
TITLE	DISK	FILE	CONTROLLER	EMULATION	FOR	MP

END

The END pseudo instruction signals the end of this program for assembly and must be the last instruction in a program. It causes the assembler to proceed with the 'complete assembly' process. On completion of the assembly process, the assembler is reset and continues reading input information to obtain the next microprogram of a batch to assemble. The total assembly process is completed on detecting a FINIS pseudo instruction.

Example:

FINIS

The FINIS pseudo instruction signals the completion of a batch of assemblies by the MP assembler and returns control to the host computer operating system. The MP assembler accepts the FINIS pseudo instruction in either columns 10 or 11.

Examples (either of the following):

OPTN

The OPTN pseudo instruction informs MASS of the control and input/output options for this assembly. The OPTN pseudo instruction may appear



anywhere in the program. The requested options are separated by commas with the first blank or column 50 terminating the scan. Most of the options once set remain in effect for the complete assembly and, therefore, more than one OPTN pseudo can be specified. Only the first character is used to determine the requested option (L and LIST are equivalent option requests). Any illegal options are flagged as errors and ignored.

The following options may be specified by the OPTN pseudo:

- C The C (CROSS) option requests a cross-reference listing on the list file (OUTPUT).
- D The D (DEAD) option requests a deadstart format file on the object file (PUNCH).
- M The M (MAP) option requests a listing of unused (zero) micromemory locations on the list file (OUTPUT). This option is available only if either deadstart or simulate output is requested.
- N The N (NOSUM) option requests that no checksums of micromemory be generated. If the checksum is generated, the result is an exclusive OR of all of micromemory and is stored in the first (lowest) available unused (zero) micromemory location.
- The S (SIMULATE) option requests that a binary file of micromemory, file 1, file 2 and the symbol table be written on the simulation object output file (SIMFIL for input by the MP Simulator (MPSIM).
- L,L=xx The L (LIST) option requests a listing on the list file (OUTPUT). If the L=xx option is used, and the xx is not equal to 61, no further output listing is produced.
 - I The I (INPUT) option requests from a file named INPUT.

The default options are L and I. File names in parenthesis are defaults.

Example:

11 17

OPTN CROSS, DEAD, LIST, INPUT

FILE

The FILE pseudo is used to place constants in file registers 1 or 2. Card column 17 contains either 1 (or blank) or 2 to specify which register file the succeeding constants go into. The FILE pseudo is used to create constants for loading into the file registers and may appear anywhere in the program.

Example:

11	17
FILE	1
FILE	2
FILE.	

Assumes file 1

ENDF

The ENDF pseudo is used to terminate the assembly of instructions into the register file areas.

Example:

LISTING CONTROL

The listing output for the MP assembler is controlled by the following pseudo instructions. These pseudo instructions may appear anywhere in the source input between IDENT and END pseudo instructions.



EJECT

The EJECT pseudo instruction causes the listing to eject to the top of the next page and start a new page. The EJECT pseudo instruction card is not printed.

Example:

SPACE

The SPACE pseudo instruction causes blank lines to be printed. The SPACE pseudo instruction is not printed. The number of blank lines to be listed is defined in the A field of the pseudo instruction. The A field may contain a constant or a predefined symbol.

Example:

_	2	11	17
	NUMBER	EQU	2
	•	SPACE	6
		SPACE	NUMBER
		I .	

The first SPACE pseudo instruction would cause six blank lines to be printed. The second would cause two lines to be printed if NUMBER has been defined to be 2 by an EQU or SET pseudo instruction.

BOX

This pseudo instruction is used in conjunction with EBOX to provide emphasis for comments in the listing. This pseudo instruction is not printed; instead, a line of asterisks will be listed. All succeeding cards will have asterisks in columns 1 and 80 to create comment cards. Only an EBOX or END pseudo instruction following a BOX pseudo instruction will be executed. The listing will be spaced one line before printing the first line of asterisks for the BOX command.

NOTE

A BOX command will turn all succeeding microinstructions to comment cards until EBOX is encountered.

Example:

EBOX

This pseudo instruction causes a line of asterisks to be listed rather than this pseudo instruction. In addition, the automatic assignment of asterisks to columns 1 and 80 started by the BOX pseudo instruction will be terminated. One blank line will be listed after the line of asterisks.

Example:

NOLIST

The NOLIST pseudo stops further listing until a LIST or END pseudo is encountered, except for cards which cause diagnostics.

Example:

LIST

The LIST pseudo starts listing after being stopped by the NOLIST pseudo.

Example:



REM

The REM pseudo is another way to create comments. The entire card is considered a comment card except there is no automatic line space before the REM card, or after unless it immediately follows a comment card.

Example:

1	11	17		
THIS	REM	IS	A	COMMENT

MEMORY MANAGEMENT AND SYMBOL DEFINITION

These pseudo instructions define symbols and provide for controlling the allocation of micromemory for the object code output.

The EQU, SET, VFD, continuals and ORG pseudo instructions require an address expression that begins in card column 17 and may continue through column 50. The expressions are made up of operands separated by operators. The operands may be constants, asterisks or previously defined symbols. The operators are +, -, * and / (add, subtract, multiply and divide). Expressions are evaluated from left to right; there is no hierarchy of operators. Parentheses are not allowed for grouping within an expression. The expression terminates on the first blank character. The results of the expression and any intermediate computations are truncated integers within the range of the host processor; for each EQU and SET, the range is 0 to 2^{20} (19 bits).

Example:

A-B/C*D+E	Legal (if A=8, B=3, C=4, D=6, E=3, value is 9)
F	Legal
*	Legal (current location counter)
**5	Legal (current location counter times 5)
*+ 5	Legal (current location counter + 5)

All micromemory and file register definitions are defined as 2*value. In an expression, the micromemory/file register definitions are adjusted during multiplication and division.

Note that symbols for file and memory locations are stored in the symbol table at twice the memory location, if upper of the microinstruction pair, and twice the memory location, plus 1, if lower of the microinstruction pair. Symbols for file locations are always twice the file location. This implies that if the label "STUFF" is associated with micromemory location OOC upper, then the instruction LABEL EQU STUFF+5 would generate a hexadecimal 11 for LABEL. If the EQU was changed to ORG, then LABEL would be associated with 011 upper (not 00E lower).

Any symbol appearing as an operand in the expression of an EQU, SET, ORG instruction must have appeared and been defined in the location field prior to its use in the expression.

EQU

The EQU pseudo instruction assigns a value corresponding to the expression beginning in column 17 to the symbol appearing in the location field (columns 2 through 9). The symbol in the location field of a micromemory location or file takes on an upper or lower quality matching that of the expression in the A field. The use of the qualifier field in column 1 of an EQU card has no effect on the quality of the symbol defined by the equate operation.

Example:

2	11	17	
VALUE	EQU	4	VALUE = 4
LABEL	EQU	VALUE*VALUE/2	LABEL = 8
LABEL.1	EQU	LABEL+1*2	LABEL.1 = 18
P.43X	EQU	VALUE/3	P.43X=1
LOCATE	EQU	*	LOCATE = current location counter

SET

The SET pseudo instruction assigns a value corresponding to the expression beginning in column 17 to the symbol appearing in the location field. The symbol in the location field takes on an upper or lower quality-matching that of the expression in the A field. The use of qualifier field (column 1) has no effect on the upper or lower quality of the symbol. Unlike the EQU pseudo, a symbol defined by a SET pseudo may be redefined by a later SET. A symbol defined by SET may not appear in the location field of an EQU pseudo.

Examples:

	2	11	17		
	VALUE	SET	4	VALUE = 4	
	LABEL	SET	VALUE*VALUE/2	LABEL = 8	
	LABEL	SET	LABEL+1*2	LABEL = 18	
	P.43X	SET	VALUE/3	P.43X = 1	
	LOCATE	SET	*	LOCATE = current location	counter
	LOCATE	SET	*+ 5	LOCATE = current location	counter +5
-	1				

ORG

The ORG pseudo instruction is used to assign a starting value to the micromemory location counter. The micromemory location counter provides for automatic allocation of microinstructions to successive upper and lower locations, unless the allocation is changed by the coding of a plus sign or minus sign in column 1 of the microinstruction input card. When ORG is encountered, all instructions and data following the ORG pseudo instruction are assembled in consecutive upper and lower micromemory locations, starting with the upper location of the address specified by the expression beginning in column 17. The ORG may be used as many times as desired. If use of the ORG pseudo instruction causes some instruction to be assembled into a non-zero location (i.e., assembly over an already assembled location), an error is flagged and the number of the card that previously caused the location to be assembled is printed for cross-reference. The most recent instruction does, however, overlay the previously assembled instruction.

17328900-02

Examples:

		
_	11	17
•	ORG	100+ABC
	ORG	FF3X
	ORG	TAG

Set program location counter to upper of 150 decimal if ABC = 50.

Set program location counter to upper of FF3₁₆.

Set program location counter to the

value TAG (provided TAG is defined).

DATA DEFINITION

Three data definition pseudo instructions are provided so the programmer can define 32-bit constants to be inserted in the micromemory at the current location specified by the micromemory location counter. The pseudo commands are DEC, OCT and HEX for decimal, octal and hexadecimal constant generation. The pseudo commands are coded in the F field of the coding form, and a string of digits in the number base is included in columns 17 through 28. Comments may start in any column after 29. The string of digits may include a minus sign (-). Embedded blanks are ignored. The string of digits is converted in its number base to a 32-bit binary number. The result is complemented (one's complement) if a minus sign exists anywhere in the string. A symbol may be placed in the location field for referencing the constant, and the qualifier field may have a plus, a minus or a blank to control the micromemory allocation.

An error is indicated if the string of digits contains any digit not in the number base.

OCT

The OCT pseudo instruction causes the string of digits starting in column 17 to be converted from octal representation to binary and stored at the current micromemory location. A symbol in the location field is optional. Occurrence of any character other than 0 through 7 or minus in the string will cause an error to be indicated. The string is terminated by a blank.



Example:

 11	17
OCT	123
OCT	-123

Create 0000000123_8 in the current location Create 37777777654_8 in the current location

DEC

The DEC pseudo instruction causes the string of decimal digits in columns 17 through 28 to be converted from decimal representation to binary and stored as a 32-bit number in the current micromemory location. A symbol in the location field is optional. Occurrence of any character, other than 0 through 9, or minus in the string, will cause an error to be indicated. The string is terminated by the first blank.

Example:

DEC 10
DEC -10

Create .0000000A (hex) in the current location. Create FFFFFFF5 (hex) in the current location.

HEX

The HEX pseudo instruction causes the string of hexadecimal digits in columns 17 through 28 to be converted from hexadecimal to binary representation and stored as a 32-bit number in the current micromemory location. A symbol in the location field is optional. Occurrence of any character other than 0 through 9, A through F or minus in the string will cause an error to be indicated. The string is terminated by the first blank.

Example:

#EX DEAD

HEX DEAD

Create 0000DEAD (hex) in the current location. Create FFFF2152 (hex) in the current location.

The VFD pseudo instruction is used to create specific bit patterns within a micromemory word.

The VFD pseudo generates one word (32-bits) from one or more specifications in columns 17 through 50. The format of the instruction is:

VFD
$$s_1, m_1, s_2, m_2, \ldots s_n, m_n$$

where s_i is the number of bits of the value of m_i to be included in the word, and

m; is an expression.

The sum of the s_i's must equal 32. If the number of bits necessary to contain the value of an expression is less than the corresponding size s, the value is right-justified with zero fill. If the number of bits is larger than the size s, bits are truncated from the left.

Like the EQU, SET, ORG pseudo instructions, any symbol appearing as an operand in an expression MUST have been defined in the location field prior to its use in the instruction.

Example:

•	Sapatitati						
	Result	2	11	17			
	-	A	EQU	10			
		·	ORG	8			
	E0A0A803	x	VFD	4,-2,8,A,2,X,6,X+2,1,1,11,*-5			
	80000000		VFD	32,X			
	FFFFFFB		VFD	32,-5			
	00007000		VFD	16,0,4,7,12,0			
		1	L .	•			

PROGRAMMING INFORMATION

The programming information pseudo instructions provide the programmer with additional information in the output listing.



The MASS analyzes each microinsruction for its execution time in the variable cycle length of the MP. This timing information is printed immediately preceding the first column of the card listing on the assembler printout. This timing is represented in a cycle count of the 56 nanosecond basic cycle time of the MP. The following pseudo instructions allow the programmer to notify the assembler of the mode of operation for timing purposes.

If no timing pseudo instructions are used, the assembler assumes two's complement operation. In addition, some instructions take a different amount of time to execute, depending on whether they are executed on a 16- or 32-bit machine. See the Basic System Controller Model 65109 Hardware Reference Manual for these differences.

CMP1

The CMPl pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if the MP were operating in the one's complement mode. CMPl may be used anywhere and as often as desired in a program.

CMP2

The CMP2 pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if the MP were operating in the two's complement mode. CMP2 may be used anywhere and as often as desired in a program.

CONDITIONALS

Instructions in a program can be skipped or assembled according to the value of a symbol or expression at assembly time by the use of conditionals. In the following conditional pseudos, m is the symbol or expression to be evaluated; any symbols used must be previously defined. The tag is a symbol and is optional; if used, it is separated from m by a comma.

17328900-02

All instructions following the conditional pseudo instruction are assembled when the test condition is true and skipped when the condition is false.

Skipping is terminated either by the matching ENDIF or an END pseudo.

One blank line is output on the listing before each conditional.

ENDIF

This pseudo is used to terminate the conditional skipping of code to be assembled; otherwise, it has no effect. If there is no tag in the location field, any skipping in effect is terminated. If a tag appears in the location field, only the conditional(s) which specify that tag will be terminated. One blank line is skipped after an ENDIF.

Format:

١	2	11	17
	tag	ENDIF	

IZD

IZD causes the coding lines following the IZD pseudo to be assembled only if the value of $\,\mathrm{m}\,$ is zero.

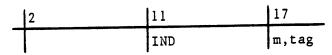
Format:

2	11	17
	IZD	m,tag

IND

IND causes the coding following the IND to be assembled only if the value of $\, m \,$ is not zero.

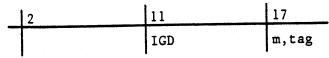
Format:



IGD

IGD causes the coding following the IGD to be assembled only if the value of m is greater than or equal to zero.

Format:



ILD

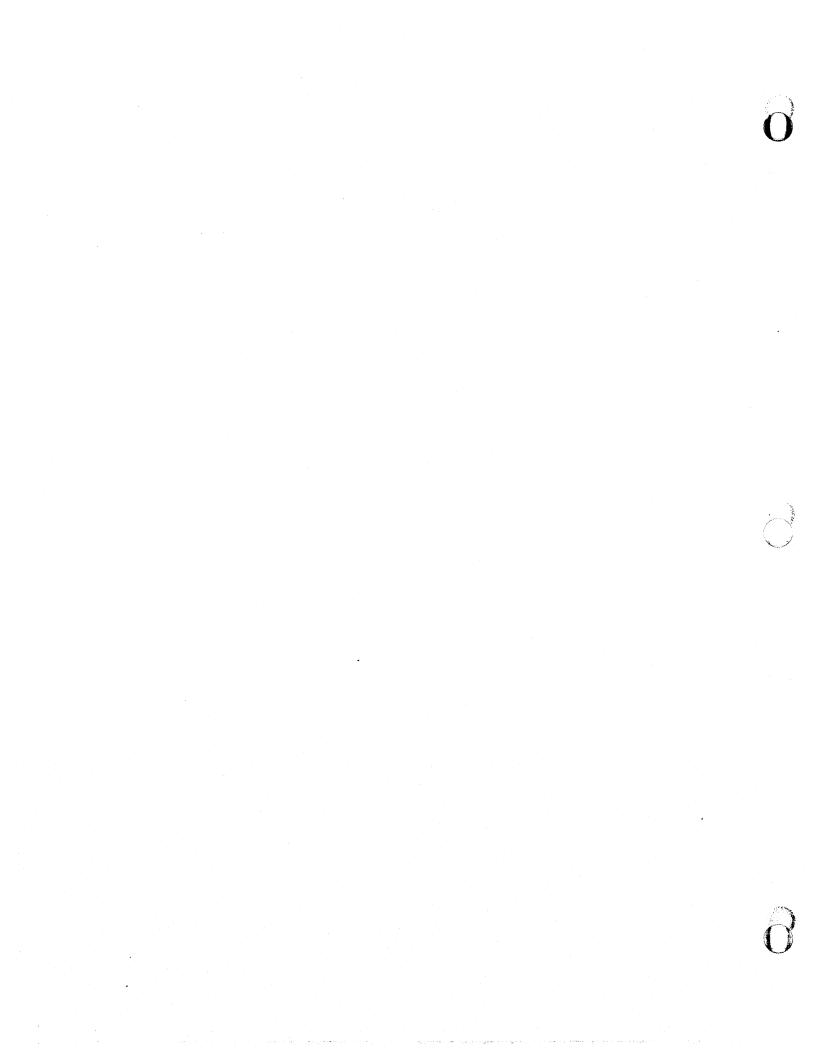
TLD causes the coding following the ILD to be assembled only if the value of m is less than zero.

Format:

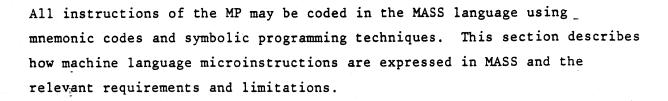
 2	11	17
	ILD	m,tag

Example of Conditional:

2	11	17
XYZ	EQU	1
JKL	EQU	0
PQR	EQU	-1
	IZD	XYZ, DWD
	•	
	•	Code Not Assembled
	•	
DWD	ENDIF	
	IGD	XYZ,ART
	•	
	•	Code Assembled
	IND	XYZ,ART
	•	
	•	Code Assembled
	ILD	JKL-XYZ,ART
	•	
	•	Code Not Assembled
ART	ENDIF	Terminates the IGD, IND and ILD
BUFF	EQU	*
	ORG	BUFF+X
	IGD	*-BUFF-30
	•	
-	•	Code not assembled if value
	•	of X is greater than 29.
	ENDIF	



MNEMONIC MICRO INSTRUCTIONS



MP MACHINE ORGANIZATION

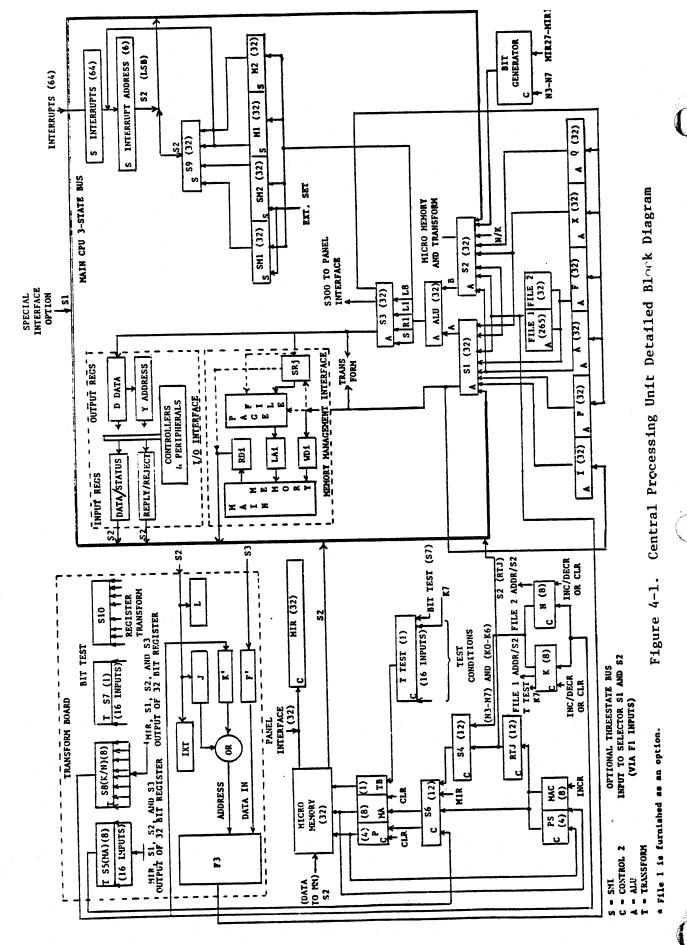
Figure 4-1. gives the organization of the registers and the ALU in the MP, including transforms.

MICROINSTRUCTION FORMATS

Figure 4-2. shows the format of the microinstructions for the MP. Each microinstruction is 32 bits in length. The micromemory is organized into 64-bit micromemory locations; each location contains two microinstructions, an upper microinstruction and a lower microinstruction. The determination of which microinstruction of a pair to execute is specified by the T (test) field of the previously executed microinstruction.

Each microinstruction consists of five major divisions:

- M field, which specifies the mode of selecting the next microinstruction pair and also selects different formats for interpretation of the S and C fields of the microinstruction.
- ALU control, which consists of the F, A, B and D fields; these fields specify two sources, a destination and an ALU operation to be performed. ALU control may be included in all microinstructions.



			Format	-^	-	» ^ـــــر	-		
31					П		[s		
30						resi	res		
8						Adc	Adc		
7	7)			O		l le l	l ue	기	r Z
9	၁			$ \cdot $	0	cro	cro	Value for K	Value for N
24 25 26 27 28 29 30 31						Next Micromem Address	Next Micromem Address	Valı	Valu
24				TVT	T/T	Nex	Nex		
23							П		
1 22	လ			တ	Page or Constant	S	Page	S	S
20 2					Pag		1 ²		
19 20 21 22 23				0		0	1	0	
18								-	:
6 17	T								
12				Not	KK/				
10 11 12 13 14 15 16 17 18	Q		의	Any C-Field Command Not Listed Below	TMAK/ J/				
13		-	C-Field Mnemonic	omu	>=	du	фш		
1 12	В	Shift Control	Mne	Ö g C	TN/ TK/ TMA/ GITMAK/ K'/	Same Page Jump	Cross Page Jump		
0 1	Ŧ		eld	Fiel Bel	'K/ MAF	age	Page	×	W Z
9 1		TO TO	E-0	Any C-Field Listed Below	/ 1 GIT	ne I	880		
80	V	1	-	Any	T	Sar	Cr		
2		2	ion						
5 6		Sale	ruct						
4	ഥ	g Sc	inst	n ntial	n ntial			ntia	ntial
က		Shift and Scale	Microinstruction Sequencing	Return Sequentia	Return Sequentia	Jump	Jump	Sequential	Sequential
2		Shif	Σ "	Se Se	Se Re	ر ا	٠ <u>ـ</u>	Š	Š
1	M			0-	0-	0	0		
0				00	وع				

F1g. 4-2. Microinstruction Formats

1.44.1.1

- T field, which specifies a condition to be tested in selecting the upper or lower of the next microinstruction pair for execution.
 The T field may also specify an unconditional selection.
- S field, which specifies a special operation to be executed in parallel with ALU control, a page for a microinstruction jump or transform, or extension of the A, B and/or D field coding.
- C field, which specifies additional operations to take place in parallel with other operations, a jump address, or a constant for transfer to the N or K register.

The six microinstruction formats in Figure 4-2. are selected based on the information coded in the M and C fields of the input form. In all cases, the ALU control portion of the microinstruction may be coded regardless of the format selected. Basic test conditions may be coded in all formats; extended tests may only be coded in format 1.

F FIELD

The F field specifies either ALU operations on the output of selector 1 and selector 2 (the A field and the B field respectively), or shift/scale operations in the A or AQ registers. In the case of shift and scale operations, the A and B fields of the microinstruction must not be specified.

ALU OPERATIONS

The ALU operations are either logical or arithmetic, and combine two source inputs. The result is routed to a single destination. The two inputs are called the A source and the B source. The A source is referred to as the A input or selector 1 (S1) and the B source as the B input or selector 2 (S2).

LOGICAL OPERATIONS

The logical operations perform bit-by-bit combinations of the A input and B input for delivery to the destination. An example of the use of the logical operations is shown in Figure 4-3.

Assume: 0 0 1 1 A input bit values 0 1 0 1 B input bit values

F Code	Bit Results of the Operation	Operation
ZERO	0 0 0 0	Zeros output
A.B	0 0 0 1	Logical product of A and B
AB	0 0 1 0	Logical product of A and not B
A	0 0 1 1	Transfer A
-AB	0 1 0 0	Logical product of not A and B
В	0 1 0 1	Transfer B
EOR	0 1 1 0	EXCLUSIVE OR of A and B
A+B	0 1 1 1	INCLUSIVE OR of A and B
-AB	1 0 0 0	Logical product of not A and not B
-EOR	1 0 0 1	Negation of EXCLUSIVE OR of A and B
- B	1 0 1 0	Transfer one's complement of B
A+-B	1 0 1 1	INCLUSIVE OR of A and not B
- A	1 1 0 0	Transfer one's complement of A
-A+B	1 1 0 1	INCLUSIVE OR of not A and B
-A+-B	1 1 1 0	INCLUSIVE OR of not A and not B
ONE	1 1 1 1	One's output

ARITHMETIC OPERATIONS

The arithmetic operations of addition and subtraction are specified in this set of F codes. Two options are provided. The first option provides an arithmetic overflow test if the letter T is included in the F code. This

sets the overflow capture bit on the transform logic if an add or subtract operation generates an arithmetic overflow. The overflow condition exists if the signs of the A source and B source are the same and the sign of the result is different (addition), or if the signs of the A and B sources differ and the sign of the result is the same as the B source (subtraction). An example of an overflow condition is shown in Figure 4-4.

The second option provides for a forced carry into the ALU logic unit, coded as a +. This option is used to emulate multiple-precision arithmetic.

The arithmetic operations are performed in one's or two's complement arithmetic and can operate on single-precision operands using the main ALU. Selection of one's complement or two's complement mode (SM101)* is explained in the Basic System Controller Model 65109 Reference Manual.

The arithmetic codes are as follows:

F Code	Operation
SUB	Subtract B input from A input.
ADD	Add A and B inputs.
SUBT	SUB with overflow test.
ADDT	ADD with overflow test.
SUB-	SUB with forced carry-in (generally the same as a SUB).

^{*}SM100 means status/mode register 1, bit position 00. Bit positions are numbered from left to right, with 00 as the left-most bit.

		***	* *	***				
PLE	COMMENTS	诉法律的特殊的特殊的法法法的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的特殊的		计存储设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设	X = (Q) EOR (A) COMPLEMENT X		COMMENTS	
EXAM	¥	* * *		*			T T	
LOGICAL OPERATIONS EXAMPLE	ပ	***	SN	*********			ပ	
GICAL	s	* * * * * * *	EXAMPLES OF LOGICAL OPERATIONS	****			w	
	0	* * *	SAL OF	***	××	3	0	
HASS	60	* * * * *	1001	****		ď	6	
1.0	•	* * * *	ES OF	***	< ×		4	
ER PSD VERSION 1.0 HASS	L.	****	EXAMPL	* * * * * *	EOR - A	ω	u.	
HBLER PSD	LABEL	*****	» • 1	****			LABEL	
CYBER/NP-60 MICRO ASSENBL	CYCLE				m m	m	CYCLE	
IW O	· ·				5 2000	3 200	W	
/HP-6	HEX Image				5326	. 4A2.	HEX IMAGE	
CYBER	HEX				019 L 5326 2 01A U 40C6 0	ن	HEX	
	Ξ							

Figure 4-3. Example of Logical Operations

EXAMPLES FOR MASS REFERENCE MANUAL

COMMENTS

Ħ

ပ

0

L

LABEL .

CYCLE

HEX IHAGE

HEX

CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 MASS

			* * *	OVERF	LOW A	N N	etttettettettettettettettettet	# H H H H H H H H H H H H H H H H H H H	S		esseessa arannessa parannessa parannessa parannessa parannessa parannessa parannessa parannessa parannessa par OVERFLOW AND TIMING EXAMPLES Berrannessa parannessa parannessa parannessa parannessa parannessa parannessa parannessa parannessa parannessa
000 U 7265 0000	0000 55	٠		ADOT	۰	đ	⋖				OVERFLOW SAVED IN S/M
000 L 7265 2000	0002 59	ın ,		ADDT CMP2	۰	a a	<				
			•	NOTE	THAT	1 HE	OVERFLOW	BIT	STAYS S	ET IN SZP	NOTE THAT THE OVERFLOW BIT STAYS SET IN S/H UNTIL EXPLICATLY CLEARED
HEX HEX	HEX C	CYCLE	LABEL	L.	4	.	0	S	ပ	Ī	COMMENTS

Figure 4-4. Example of an Overflow Operation and Timing

F	Code		
_			

ADD+	ADD with forced carry-in (A+B+l for two's complement; A+B+l for one's complement if both numbers are positive; otherwise, A+B for one's complement if one number is negative).
SUB-T	SUB- with overflow test.
ADD+T	Add+ with overflow test.
A-	Subtract 1 from A input.
A-T	Subtract 1 from A with overflow test.
A+	Add 1 to A.

Operation

SHIFT OPERATIONS

A+T

Shift operations shift the A or the AQ register left (L) or right (R) the number of bit positions specified by the contents of the N register. Additional shift modifier postfixes are as follows:

Add 1 to A with overflow test.

- OE Enter zero (right or left) for each shift cycle.
- 1E Enter one (right or left) for each shift cycle.
- SE Extend initial value of bit 0 for each shift cycle.
- EA Perform end-around shift.

The number of bits shifted is determined by the count in N at the start of the shift instruction. If the N register is zero, no shift occurs. The N register can be set in the (same) instruction by placing N=value in the C field; the value set affects the following instruction. The shift operations are various combinations of shift A or A/Q, left or right, end-around or end-off, sign-extended or not sign-extended and entry of a O to I in the vacated bit position. The A, B and D fields must be left blank.

An example of the shift operation is shown in Figure 4-5.

The shift operation is coded as the F field mnemonic. The D field of the microinstruction normally should be left blank to generate a NOP because the shift is performed in the A or AQ register without use of the ALU. The shift options are coded in the A and B fields of the microinstruction, and these fields of the coding form must be left blank.

The legal shift mnemonics using the above codes are as follows:

F Code	Operation
AR0E	Shift A right with zero entry.
ARSE	Shift A right with sign extended.
AREA	Shift A right, end-around.

HEX M M H	HEX CYCLE IMAGE COUNT	N. T.	CYBER/NP-50 41CKU ASSERBLE F A B HEX HEX CYCLE LABEL F A B ANL IMAGE GOUNT # SHIFT EXAMPLE, # THEN THE FOLLOW # AQ END AROUND # INSTRUCTION COM ####################################	F A B SHIFT EXAMPLE, THEN THE FOLLOW AQ END AROUND INSTRUCTION COM	SHIFT EXAMPLE, THEN THE FOLLOW AQ END AROUND ANSTRUCTION COF	B PLE A DLLOWI DUND 6	SHIFT EXAMPLE, ASSUME (N) IS 6 IN THEN THE FOLLOWING INSTRUCTION LEFA ENDER OF PROPERTY OF A PERSON OF	(N) I TRUCT	CTION LEFT AFTER	D S (: HT ***********************************		D S (: MT COMMENTS ASSUME (N) IS 6 INITIALLY 6 PLACES. N=FF AFTER APPLETION.		DIAGNOSTICS
0 100 L	001 U 7CDG 0000 5+N		AQLEA	AQLEA SHIFT MOS CLEARING	NOS H	r signi	AQLEA ************************************	8-91	T CHA	8-BIT CHARACTER INTO Q AFTER	NTO 0	AREAREMENT CHARACTER INTO Q AFTER	* * * *	
001 L D803	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Z M +	ZERO AQLEA ************************************	ZERO AQLEA ************************************	ZERO AQLEA *********** PLAGE HOST SIG HAKE ALL OTHER	SIGN	Q ************ IFICANT BI BITS ZERO*	**************************************	N=R N TO	AQLEA AQLEA ************************************		N=8 secure express contractions of the second contractions of the second contraction of the seco	• • •	
002 C 003 U HEX MML	L D000 303F U 7D40 0000 5 HEX CY IMAGE CO	3 3 5+N CVCLE COUNT	LABEL	AGROE	. ≪	100	•	ທ	2 0	E E	L.	COMMENTS	OIA	DIAGNOSTICS

17328900-02

F Code	<u>Operation</u>
ALOE	Shift A left with 0 entry.
AL1E	Shift A left with 1 entry.
ALEA	Shift A left, end-around.
AQR0E	Shift AQ right with 0 entry.
AQRSE	Shift AQ right with sign extended.
AQREA	Shift AQ right, end-around.
AQL0E	Shift AQ left with 0 entry.
AQLEA	Shift AQ left, end-around.

SCALE OPERATIONS

Scale operations are similar to shift operations but are conditioned by the two bits at the scale point being different. Scale examines N. If N=0, the scale operation is terminated. If $N\neq 0$, then the contents of A register bits 0 and 1 are examined. If the bits are different, the scale is stopped. Otherwise, the scale operation then takes place as a left shift of A or AQ and continues until either the bits at the scale point are different or until the contents of N are reduced to 0. At completion, the difference of the initial value of N and the final value of N is the number of bits that A or AQ was shifted. The A, B and D fields of the coding form should be left blank so the assembler can insert the correct values.

The examples depicted in Figure 4-6 assume that a number is positioned in the A/Q registers and must be scaled. In the one's complement example, an end-around scale is used to provide for the propagation of the correct value for the least significant bits. In the two's complement example, a zero entry scale is used.

		1 9 -			
COMMENTS	TERRETERRETERRETERRETERRETERRETERRETER	SET MAXIMUM SHIFT NUMBER OF SHIFIS = N = 64	nterterentertertertertertertertertertertertertert	SET MAXINUM SHIFT	COHMENTS
Ħ	SCALE EXAMPLE ONES COMPLEMENT ARITHMETIC	,	**************************************		Ħ
ပ	IT ARI	→ 9=N	T ARI	→ 9=N	ပ
s	PLEMEN		PLEHE		v
0	ES COM		OS CON		0
. 60	4PLE ON		APLE T		ω
Ø	E EXA!	٠	E EXA	ñ	4
u.	SCAL	SOLEA	SCAL	SOLDE	L.
LABEL			* * *		LABEL
CYCLE		N+S 0		N + W	CYCLE
HEX IMAGE		003 L D800 3040		004 L D800 3040	HEX
HEX		0 04 N		0 50 0 0 004 L	HE

Figure 4-6. Examples of Scale Operations

Scale operations are as follows:

F Code	Operation
SLOE	Scale A register left with 0 entry.
SLIE	Scale A register left with 1 entry.
SLEA	Scale A register left, end-around.
SDL0E	Scale AQ registers left with 0 entry.
SDLEA	Scale AQ registers left, end-around.

A, B, AND D FIELDS

The A, B and D fields of the microinstruction specify sources and destinations of information in the MP organization. In all three fields, there are basic mnemonics and extended mnemonics. In the object language output, the A, B and D fields occupy three bits each and thus allow only for specifying one of eight different sources and destinations. Since more than eight sources and destinations may be specified in each field, the S field is used also to provide alternate coding interpretation for the 3-bit number in the A, B and D fields. Alternate codes are referred to as A', B', D', D" and DD". The MP assembler accepts any of the specified alternate mnemonics for the fields and provides an automatic S field setting in the object code output. The result of the use of the S field to specify operations, as well as alternate decodings of the A, B and D fields, leads to a possible conflict of mnemonics. The resolution of this conflict is described in Appendix D of this manual. The assembler also provides diagnostic messages if any conflict occurs in the source program.

A FIELD

The A field specifies a source of information to the output of selector l and is available for the A input of the ALU.

A Code	Operation
F2	Initial value of N at start of microinstruction specifies address in File 2.*
P	Use contents of P register as A source.
I	Use contents of I register as A source.
X	Use contents of X register as A source.
A	Use contents of A register as A source.
F	Use contents of F register as A source.
Fl	Initial value of K at start of microinstruction specifies address in File 1.*
MEM	Contents of main memory are transferred. Data must have been read in a previous microinstruction.

The following A codes (A' codes) are extended mnemonics; the S field must be left blank.

A Code	Operation
SM1	Use contents of SM register 1 as A source.
SM2	Use contents of SM register 2 as A source.
Ml	Use contents of interrupt mask register 1 as A source.
M2	Use contents of interrupt mask register 2 as A source.

^{*}RESTRICTION: Value of addressing register (N or K) cannot have been modified by a C field increment or decrement command in the preceding microinstruction. In addition, whichever file is used during current microinstruction, this file cannot have been written in the preceding microinstruction.

B FIELD

The B field specifies a source of information to the output of selector 2 and is available for the B input of the ALU.

B Code	Operation
-ZERO	Use all zeros as output of selector.*
<u>F</u> N	Use N register as bits 16 through 23 of selector 2 with zeros as rest of selector 2.*
BG	Use single bit set to 1 with rest 0's. The bit to set is controlled by either value of lower five bits of C field of this microinstruction or contents of N register as elected by setting of status/mode bits.
K	Use K register as lower 8 bits of selector 2 with zeros as rest of selector 2.*
N,K	Use N and K registers as lower 16 bits of selector 2 with zeros as rest of selector 2. K fills bits 24-31 and N fills bits 16-23.*
X	Use contents of X register as B source.
Q	Use contents of Q register as B source.
F	Use contents of F register as B source.
F1	Use contents of a word in File 1 as B source input. Value of K at start of microinstruction specifies address in File 1.**
F2	Use contents of a word in File 2 as B source input. Value of N at start of microinstruction specifies address in File 2.**

^{*}Selection of this B code option is controlled by setting bits 28 and 29 of the microinstruction in the C field. Other C field commands may be given if they match in bits 28 and 29. The assembler allows this operation.

17328900-02 4-15

^{**}RESTRICTION: In the preceding microinstruction, the addressing register (N or K) cannot have been incremented or decremented and whichever file is read (F1 or F2), cannot have been written.

Operation

B Code

MEM

Contents of main memory is used as B source. Data must have been read in a previous microinstruction. Restriction: If extended A code is used, data from A source is also used as B source, suppressing memory data.

O

The following B codes (B' codes) are extended mnemonics and the S field must be left blank. If these extended mnemonics are used, it is not possible to use extended mnemonics in the A or D field.

B Code	Operation
CRTJ	Read the complement of the RTJ register in the lower 12 bits of selector 2.
INRD	Read data/status through I/O TTY card from peripheral devices.
INRS	Read responses from I/O TTY card.
MMU	Transfer data from micromemory to X register as a destination. Data is passed through ALU, and F field must make reference to B source. D field, if specified, will also contain the data from micromemory.* Next microinstruction is always next sequential upper instruction.
INTA	Read complement of address of current highest priority active interrupt having its corresponding mask bit set to output of selector 2. Address consists of bit number of mask bit that controls interrupt. An INTU test command must have been given in the preceding microinstruction.

^{*}The address of micromemory is determined by the transform number specified in the C field or NK, and the upper or lower location is specified by the test field of this instruction.



D FIELD

The D field specifies the destination of information from the main organization of the MP. There are four sources of information for delivery to the specified destination. These are:

- The optionally shifted output of the ALU. This shifting occurs
- in a shifting network (selector 3) that provides the shift on the output of the ALU.
- The direct (unshiftable) output of the ALU.
- The output of selector 1 (input to the selector is specified by the A field).
- The output of selector 2 (input to the selector is specified by the B field).

D Code	<u>Operation</u>
NOP	Do not transfer any information to any destination.
P	Transfer output of ALU as shifted by selector 3 to P register.
I	Transfer output of selector l to I register.
Q	Transfer output of ALU as shifted by selector 3 to Q register.
Fl	Transfer output of ALU as shifted by selector 3 to F register, and then write in file 1 during first part of next micro-instruction.
	Location in file 1 to receive information is specified by
Ξ	contents of K register on completion of this microinstruction.
	Note that if next instruction specifies Fl in A or B field, an erroneous result will be received; the information is
	available in the F register and should be obtained
	from there.

17328900-02 4-17

A	Transfer outpu	t of ALU as	shifted by selector	3 to A register.
X	Transfer outpu	t of ALU as	shifted by selector	3 to X register.
F	Transfer outpu	t of ALU as	shifted by selector	3 to F register.

Operation

The following extended D (D' codes) may be specified in the same microinstruction as an extended A code. The assembler provides the proper S field setting.

D Code	Operation
MMU	Transfer output of selector 2 to 32-bit micromemory word. Micromemory address is specified by transform corresponding to number contained in C field of this microinstruction. Upper or lower micromemory word is specified by T field of this microinstruction. Next microinstruction is taken from next sequential upper location.
M1	Transfer output of ALU to Ml register.*
M2	Transfer output of ALU to M2 register.*
SMl	Transfer output of ALU to SMl register*
SM2	Transfer output of ALU to SM2 register.*
IOD	Transfer output of selector 3 (S3) to I/O data register on TTY card.
IOA	Transfer output of selector 3 (S3) via the I/O data register to I/O address register. Destroys contents of I/O data register.

D Code

^{*}Outputs to the mask and status/mode registers are not shiftable.

EXAMPLES OF F, A, B, AND D FIELDS

The assembler output listing shown in Figure 4-7 demonstrates the basic use of the fields discussed above. In some cases, the S and C fields will also be used to demonstrate common programming errors that are detected by the assembler.

S FIELD

The special field (S field) is coded in columns 35 through 40 of the assembly coding form. If this field is not used by the assembler to specify alternate translations for the A, B or D fields, it may be used by the programmer to specify a special instruction or it may contain a constant or a programmer defined symbol. The S field mnemonics specifying alternate A, B or D field codings are not normally used by the programmer and are automatically generated as required by the assembler.

These S codes specify operations taking place in parallel with the F and C codes.

S Code	Operation
NOP	No operation specified in S field.
RPT	If contents of N register are not equal to zero, decrement N by one and take current microinstruction pair, using upper/lower decision implied by T field. If contents of N register equals zero, decrement N by one and use microinstruction sequencing specified by M field and T field.
L8EA	Selector 3 is set to provide shift of data left eight bits, end around. This provides shift of ALU output to P, A, F, X and Q registers if used as destinations.

LABEL	LL	< ■	6	s o	s	Σ υ	¥	COMMENTS	DIAGNOSTICS	
	F A A	8, 0	FIELD E	EXAMPLES	* * * * * * * * * * * * * * * * * * *			descentestan		
	ADD	⋖	ø	. ≪				A= (A)+(Q)		
	A REM	SMZ		×		K=3		X= (S/N REG 2) ALSO SET K=3		
	A00 A00 A00 A00+		F1 8G ZER0	Iaaa		K=6 31 CLRK		F1(6) = 2*F1(3) P=(P)+1 32 BIT MP P=(P)+1 USES C FIELO P=(P)+1 C FIELO OPEN		
	ONE	. ◀		.						
	A + 8	SH1	១១១ ១១១១	N X X E E E E E E E E E E E E E E E E E E		19) ed 2		X=(Q) X = BIT 3 OF X SET BIT 1 OF SM1 GLEAR BIT 4 OF SM2		
•	FOLL	OM I NG	FOLLOWING CODE IN	ERROR					0	
	A00 A00	SM1	CRTJ	⋖ ⋖		K=31		A AND B BOTH REQUIRE S C FIELD USED BY B FIELD	CBC	
LABEL	L	<	80	0	S	ပ	Ħ	COMMENTS		

U 7449 000C U 5441 2040 U 4702 0000

409 U 4702

0006 201F

407 U F184 (L 7051 2 403 U 7449 0

Figure 4-7. Examples of the Use of F, A, B, and D Fields

CYCLE

HEX IHAGE

HEX HEX

409 L 5426 2000 40A U 5C06 0003 L 5615 2A01 40B U 5A97 0A04

CYCLE

HEX Image

HEX HAL

S Code	Operation
F2WR	Write contents of the F register into file 2 at address
	specified by contents of N register at start of this
	microinstruction. Must not be specified if preceding
	microinstruction used F2 as a destination code.
GATE I	Transfer output of selector 1 to I register.
HALT	If SM109 (halt bit) is set to 1, complete this microinstruc-
	tion and halt operation.
RTJ	Transfer address of next sequential microinstruction pair to
	RTJ register.
CLRNP	Clear N register and set page register in P-MA register so
	next microinstruction will be executed from page 0. This is
	normally used with J or R in M field of microinstruction.
RD1	Gate data specified by lookahead register 1, 2 or 3 onto the
RD2	main bus for use by the next instruction at selector 2. MEM
RD3	in the A or B field would allow the data to be stored in a
	register specified in the D field.

The following S codes are not normally used by the microprogrammer because they are set by the assembler to handle extended A, B and D field mnemonics.

- AP (for A' codes)
- BP (for B' codes)
- DP (for D' codes)
- APDP (for A' and D' code combination)
- DPP (for D" codes)

EXAMPLES

The code shown in Figure 4-8 will count the number of 1 bits in the A register. Assuming two's complement arithmetic, the total number of 1-bits will be left in the Q register.

The code at HERE adds X to itself to get a left shift of 1, and the COL in the T field checks for carry out of the high order bit if it is a 1. If there is no carry out, the upper instruction at HERE is repeated. If there is a carry out, the lower instruction is performed, which adds one to the A register and jumps back to HERE. Each execution of the instruction at HERE counts N down by one. When N is counted down to zero, control goes to the next sequential upper instruction, since a two's complement adder is assumed, leaving all zeros in the X register after the 32 additions of X to itself.

C FIELD

The C field is a multipurpose field and its interpretation is dependent on the M field.

JUMP INSTRUCTIONS

If the M field is coded with a J, the C field is taken as a jump address. It may contain a constant, which expresses page and location within the page. In this case, the T field must be coded to specify the upper or lower location to be jumped to. The C field may contain a symbol, which identifies a page or location within the page and an upper or lower property. In this case, the T field may be left blank and the assembler will select the correct upper or lower designation. If a T field value is given, it will override the assembler selection.

The page to be jumped to is examined by the assembler, along with the page of the jump instruction. If the jump is within the same page, format 2 with bit 19 clear (refer to Figure 4-2) is selected, allowing use of the S field for special instructions. Otherwise, format 2 with bit 19 set is selected.

Coding
Field
S
of
Example
4-8.
Figure

соинеитѕ	在在这个中的中的中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中	CLEAR Q. SET RPT COUNT 1 TO A	COUNT A 1 BIF	NEXT INSTRUCTION	сомнеитѕ
000				K	8
Σ		2	, cor		Ħ
U	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	N=32 31	HERE		U
v	* *		RPT		S
0	XANPLE	♂ ∢	×a	MOP	G
80	PEAT E	98	×ø		œ
a	,		׫		⋖
L.	FIE S FIE	ZEROB	ADD		L
LABEL	* * * * * * * *		+HERE	•	LABEL
CYCLE		m m	in ±	m	CYCLE
<i>5</i> 5		1020 101F	C280 2016	0000	
HEX IMAGE		016 L 0803 3020 017 U 5415 201F	016 U 70DE C	019 U 5800 0000	HEX INAGE
HEX		016 L 017 U	010 U	019 U	HEX

FLAG SETTING AND CLEARING COMMANDS

The Status/Mode registers contain bits which are individually associated with various status and operation modes. The individual bits in the Status/Mode register which can be turned on/off are called flag bits.

Flag Number	SM Reg. No.	Bit No.	Function
0	1	00	Double Precision
1	1	01	One's Compliment
2	1	02	Bit Generation from N Register
3	1	03	Adder Split
4	1	16	Lock Memory
5	1	17	MP60 Bit Register
6	1	18	Monitor (Program) Mode
7	1	19	Reserved
. 8	2	00	Enable Auto Data Transfer
9	2	01	Read I/O Strobe
10	2	02	Write I/O Strobe
11	2	03	Sterm I/O Strobe
12	2	16	Test/XMC MPM (MPM Test Mode)
13	2	17	Isolate BSC (BSC Test Mode)
14	2	18	Reserved
15	2	19	Page Address From S3

The commands are written as follows:

SETF/m CLRF/m

where m is a constant or a symbol. m is converted to binary and the least significant four bits are used to specify the flag to be set (set to 1) or cleared (set to 0).



REGISTER SETTING INSTRUCTIONS

The register setting instructions set the K and the N register and are coded as follows:

- K=value
- N=value

where value may be a constant or a symbol. The constant or symbol is expressed in binary, and the lower eight bits of the binary value are inserted in the C field of the microinstruction as the value to be set into the specified register.

If the S field contains the CLRNP command, the result of a K= or N= in the C field will be zero in the specified register, along with a zero in the N register.

REGISTER CONTROL COMMANDS

The register control commands provide for incrementing, decrementing, and clearing the K or the N register. The commands are as follows:

<u>C Code</u>	Operation
INCK	Increment K register by 1.
INCN	Increment N register by 1.
DECK	Decrement K register by 1.
DECN	Decrement N register by 1.
CLRK	Clear K register.
CLRN	Clear N register.

If the T field specifies a test of the register, the test is made on the initial value of the register before the control command.

SELECTOR 3 (SHIFT COMMANDS)

The following C field commands specify a left or right shift one bit position of data transferred from the ALU through selector 3 destined to P, A, F or the X register. The contents of the Q register may be combined with the destination register to make a double length register, with Q register as lower order bits. These shifts are end off with provision for entry of O, 1 or the inverse of the sign of the ALU output. The Q register may not be used as a destination register in any of the following commands and obtain correct results. Also, these commands cannot be performed when S field contains L8EA.

First Part	Direction	Ent	ry To Open Bit
R (shift destination register)	L (left)	0E	Enter zero
RQ (shift destination and Q)	R (right)	1E	Enter one
		XN	Enter complement of ALU sign

The allowable mnemonics for the shift operations are as follows:

C Code	<u>Operation</u>
RQLXN	Shift destination and Q left with complement of ALU sign entered into lowest bit position of Q. This command is used in divide iteration.
RQR1E	Shift destination and Q right and enter 1 in sign position of destination register.
RQROE	Shift destination and Q right and enter 0 in sign position of destination register.

-	C Code
- A 35	

Operation

RLOE Shift destination left and enter 0 in lowest bit position.

RLIE Shift destination left and enter 1 in lowest bit position.

RROE Shift destination right and enter 0 in sign bit position.

RRIE Shift destination right and enter 1 in sign bit position.

TRANSFORMS

Transforms are used to provide a means of forming micromemory addresses or constants from any pattern of bits from selectors, registers or data paths of the processor. The addresses may be used to sequence micromemory or to set the N or K registers.

The transform instructions in the C field are written as follows:

TMA/j TK/j TN/j TR/j TMAK/j GITMAK/j

where j is a symbol or a transform number. The symbol or number is expressed in binary, and the lower four bits of the number are inserted in the microinstruction. The S field may contain a symbol or a constant to specify the page used in the MA transform. For the TN/ and TK/ instructions, the S field must contain a symbol or a constant to avoid an informative diagnostic from MASS.

C Code Operation

TMA/j The micromemory transform begins execution at one of the 256 microinstructions (of the current micromemory page) specified by either S2 or the IXT register. The transform number j specifies the register and the bit field as follows:

4-27

OPERATION

TRANSFORM NUMBER j

0	Not Defined
1	Micro Addr = Bits 0-5 of S2
2	Micro Addr = Bits 1, 4-10 of IXT
3	Not Defined
4	Not Defined
5	Not Defined
6	Not Defined
7	Micro Addr = Bits 24-31 of S2

TK/j Load the K or N register and the K' register with data

TN/j from S2, the CPU state register/IXT register, the MIR or
the F' register. The following table describes the
various fields that are loaded into the K or N register
from bits 0-7 of S8. Bits MSB-7 of S8 are used to load
the K' register and the K' register is always loaded
whether the N or the K register is specified. The following chart summarizes the bits that are loaded into the K
and K', or the N and K' register.

The symbols under the transform number denote whether the transform is wirewrapped (Δ), clad (\bigcirc) or unassigned (OPEN). The number under each of the bit numbers of S8 denotes the register and the bit number which the N or K is loaded. Example: Bit 7 of transform 0 shows S231. This means bit 31 of S2 is loaded into the least significant bit of K, K' or N, K' register.

Bit ll of Status Mode register l is =0 if File 3 is to be addressed and =1 if File l is to be addressed.

Generally, transforms 2, 3, 4, 5 and 7 are used to assist in decoding MP-60 type assembly instruction (see MP-60 Reference Manual, Publication No. 14306500A, pages 6-1, 6-4). Transform 2 is used to decode the B field; Transform 3,

		7				1883				פאש
		9				FP30				פאס
		5				'67d±				CND
Í		4				£628				CND
	Q	3				LZdd				CND
1	1/0	2				• FP26 SMICED		·		GND
	58	7		i		QN9				GND
Ē	·	0				QND				GND
	ror	A				ans				1822
	SELECTOR	8				GND				05 SC
9	SEL	۲			·	an9				6275
1		q				GND				8228
***		E				פאס				TZZC
		#SB				IIIIVS				ZWIII
	XFORM	ИО	B OPEN	9 OPEN	A OPEN	<i>b</i>	C OPEN	D OPEN	E OPEN	О
Г										
- 1		7	1 222	2215	OZIXI	OTIXI	VI.IXI	ELTXI	MIRBL	TELXI
		9	2525	2512	OZ1XI	OTIXI 601XI	EXT/3	ZITZI ZITZI	MIESO	OETXI LETXI
		9	٥٤٦٤	\$175	61TXI	601XI	E/1X]	בגבוב	MIE30	0ETX1
	0	2 6	62.22 02.52	†175 E175	611XI 611XI	20TXI 70TXI 80TXI 80TXI	CND EXTI1 EXTIZ	GND GND SITII	WIE30	CETXI CETXI
	1/0	4 5 6	8222 6222	\$215 \$215 \$215 \$215 \$210 \$210 \$20 \$20 \$20	USAS USASI TITII BITXI BITXI	2837 LXTO LXTO7 801X1 801X1	5831 CMD TXT/1 CMD CMD	CPU SRSI GND GND IXT II	WIRZS WIRZ9	ISAS ISASI IXTZI SZTXI EXTZ9
	1/2	3 4 5 6	7.5.22 8.5.22 6.5.52 0.5.52	\$215 \$215 \$215 \$211 \$211	05.92 09.09.09.09.09.09.09.09.09.09.09.09.09.0	US SER SO OS SER SER SER SER SER SER SER SER SER SE	2830 CAD CAD CAD CAD CAD CAD	CPU 5R30 CPU 6ND 6ND EXT II	WIE 30 WIE 33 WIE 33	05 N S S S S S S S S S S S S S S S S S S
	58 1/	2 3 4 5 6	0552 6252 6252 7252 7252 7252	\$218 \$283 \$283 \$283 \$283 \$283 \$283 \$283 \$28	25.42 05.42 05.42 04.7 15.42 61.1X.1 81.1X.1	CSAZ CG) CG) CG) CG) CG) CG) CG) CG)	2829 CRD CRD CRD CRD CRD CRD CRD CRD CRD CRD	CPU SR29 CPU SR30 CPU GND GND GND	WIE 30 WIE 23 WIE ZZ WIE ZZ WIE ZZ	0572 0582 0772 0583 17172 17173 17173 17173
	TOR 58 1/	1 2 3 4 5 6	05.72 62.72 7.72 7.2.72 7.2.72 7.2.72 7.2.72 7.2.72 7.2.72 7.2.72 7.2.72 7.2.72	\$21\$ \$21\$ \$21\$ \$21\$ \$283 \$283 \$283 \$283 \$283 \$283 \$283 \$28	85.82 85.82 65.82 09.0 05.82 09.0 15.82 17.17 81.17 81.171	2828 2829 CPU 2829 CRU 2830 CRU 2831 LXTO1 201X1 201X1	CRU SRZ9 CPU SRZ9 CPU SRZ1 CPU SRZ3 CRU CRD CRD SRZ3 CRU SRZ3 SRZ3 SRZ3 SRZ3 SRZ3 SRZ3 SRZ3 SRZ3	CPU SRZ8 CPU SRZ0 CPU SRZ1 CPU GND CRU GND	WIE 30 WIE ZZ WIE ZZ WIE ZZ WIE ZZ	28.29 28.29 28.20 28
	TOR 58 1/	0 1 2 3 4 5 6	0528 6228 9228 4228 5228 5228	\$218 \$283 \$283 \$283 \$283 \$283 \$283 \$283 \$28	85.82 95.82 95.82 090 15.82 17.171 81.171	8282 CPU CR2 CR3 CR3 CR3 CR3 CR3 CT0TXI B0TXI	2829 CRD CRD CRD CRD CRD CRD CRD CRD CRD CRD	5828 CPU CRU CRU CRU CRU CRU CRU CRU CR	WIE 30 WIE ZZ WIE ZZ WIE ZZ WIE ZZ WIE ZZ	28.29 (PU 58.29 (PU 58.30 (CPU 58.30 (
	58 1/	A O 1 2 3 4 5 6	0528 6228 9228 4228 5228 +228 5228	\$21\$ \$21\$ \$21\$ \$21\$ \$283 \$283 \$283 \$283 \$283 \$283 \$283 \$28	75,82 81,82 81,82 0,93 05,82 0,93 0,582 0,93 15,82 11,71 81,131	5826 5826 5826 5820 5830 5831 15106 15100 15100	2828 CPU 5830 CPU 5800 CPU 580	5829 5829 5829 5820 5830 5830 680 680 680 680 680 680 680 68	WIE 30 WIE 53 WIE 54 WIE 54 WIE 52 WIE 52 WIE 54	1X1Z9 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3 1X1Z3
	LECTOR S8 T/	B A O 1 2 3 4 5 6	0528 6228 9228 4228 5228 +228 5228	\$21.4 \$28.7 \$28.2 \$28.0 \$28.0 \$28.0 \$28.0 \$28.0 \$28.0 \$28.0 \$2.0 \$2.0 \$2.0 \$2.0 \$2.0 \$2.0 \$2.0 \$2	CFA2 CFA2	CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU	EXTIS EXTI EXTI CPU CPU SRZB SRZB CPU SRZB SRZB CPU SRZB SRZB CPU SRZB	CPU SRZ7 CPU SRZ9 CPU SRZ0 CPU SRZ1 CPU SRZ1 CPU SRZ1 CPU SRZ1 CPU	WIE 30 WIE 53 WIE 54 WIE 54 WIE 54 WIE 54 WIE 54 KE 65	1X128 1X128 1X128 1X128 1X128 1X128 1X128 1X128 1X128 1X128 1X128
	LECTOR S8 T/	C B A O 1 2 3 4 5 6	0528 6228 9228 4228 5228 4228 7228 7228 7228	\$21\$ \$21\$ \$21\$ \$21\$ \$26 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2 \$2	045 CPAS CPAS U45 BSAS U45 CPAS U5AS U5AS U5AS U5AS U5AS U5AS U5AS U5	CAND 1 X X Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z	EXTIS EXTII EXTII CPU CPU CRU CRU CRU CRU CRU CRU	CAD CAD CAD CAD CAD CAD CAD CAD CAD CAD	WIE 30 WIE 53 WIE 54	1X1Z9 1X1Z9 1X1Z1 1X1Z1 1X1Z1 CFU
	LECTOR S8 T/	D C B A O 1 2 3 4 5 6	0528 6228 9228 4228 5228 4228 7228 7228 7228	\$21\$ \$21\$ \$21\$ \$21\$ \$250 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$28300 \$283	6ND 6ND 6ND 6ND 6ND 6ND 6ND 6ND	GND CPU SRZB SRZB CPU SRZB S	EXTIS EXTII EXTII CAU CAU CRU CRU CRU CRU CRU CRU	6 N D CPU CPU CPU CPU CPU CPU CPU CP	WIE 30 WIE 53 WIE 54 WIE 54 WIE 54 WIE 54 WIE 54 KE 55 KE 57 KE 57 KE 57	6/112 6/112 1/172 1/

the sub-opcode field; Transform 4, the index field for half-word addressing; Transform 5, the index field for character addressing; and Transform 7, the C field decode.

TR/j The register transform is used to route data directly into File 3 from either S3 or from another register.

Presently, only transform 0 is available, which puts bits 0-31 of S3 into File 3.

TMAK/j Transfer data to the K register as specified by the wiring of the K/N transform j and then perform a TMA/j.

GITMAK/j Gate the output of selector l to the IXT register and then perform a TMAK/j transform.

OTHER C CODES

C Code	Operation
LWAn	Load word address into lookahead register n from page file and S3. If write memory mode is selected, the data in the register in S1 (A field) is used.
LHAn	Load half-word address into lookahead register n.
LCAn	Load character address into lookahead register n.
LBAn	Load bit address into lookahead register n.
WPF	Load the page file (specified by address in S3) with the data word from S1.
RPF	Gate the contents of the page file (specified by address in S3) to the main CPU bus for use in S1 or S2 during the following instruction cycle.
WSR	Load the state register specified by S3 with data from S1.

RSR

Gate the contents of the state register (specified by address in S3) onto the main CPU bus for use in S1 or S2 during the following instruction cycle.

EXAMPLE OF C FIELD CODING

If the \tilde{M} field is coded with a J, the C field is used as the address of the micromemory jump destination. The examples in Figure 4-9 show legal and illegal use of the S field in conjunction with the C field.

Examples of multiply and divide codes implemented in an MP which uses one's complement arithmetic are shown in Figures 4-10 and 4-11.

M FIELD

The M field is coded in column 50 of the assembler coding form. The mnemonics in the M field specify the method of selecting the next microinstruction pair (upper and lower) to be executed on completion of the current microinstruction.

If the M field is left blank, MASS assumes that a sequential (S) specification is required.

The code selected in the M field may be overridden in two cases:

- If the T field has the code *L, the lower microinstruction of the current microinstruction pair is selected regardless of the M field code.
- If the C field specifies the transforms TMA/, TMAK/ or
 GITMAK/, then the transform addressing takes precedence over the M field code.

HEX	H H H	CYOLE LABEL	LABEL	u.	⋖	60	0	v	ပ	Ĭ	соннеитѕ	DIAGNOSTIC
בי היים ביים ביים ביים ביים ביים ביים בי	nnt 1122.	. 00 3		IDENT	IDENT MASS	•	×	HALT	HALT 400X	7	JUHP TO OTHER BANK, S HOLDS PAGE INFORMATION	6 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
000	000 L 9E00 2700 001 U 5800 0573	7334		Α 4 η	SH1			LBEA	400X LBEA RORIE RTJ THA/2	ה ה	BOTH SHIFT SAME RESULT S FIELD USE INHIBITED	CSC DIAGNOSTI(
HEX	HEX	CYCLE	LABEL	L.	⋖	æ	•	v .	ပ	¥ .	COMMENTS	

Figure 4-9. Examples of S and C Field Conflicts

		COMPLEMENT			IIVE	GET POS X AND SIGN IN K CLEAR A, SET TIMES COUNT HAKE FIRST STEP TEST	ION LOOP	SIGN TEST REM	A Q S	
so.	HULTIPLY A BY X, PRODUCT TO AQ. HULTIPLY A BY X, PRODUCT TO AQ. HULTIPLY WORKS ON POSITIVE NUMBERS SO PROVIDE LEAD IN TO CALCULATE SIGN OF NEGATIVE INPUTS AND CORRECT AT END. CODE IS WRITTEN FOR ONES COMPLEMENT INPUT NUMBERS.	INDICATE ONES G		IGN	COMP Q FOR POSITIVE CHECK SIGN OF X	GET POS X AND S CLEAR A, SET TI HAKE FIRST STEP	MULTIPLY ITERATION LOOP HULTIPLY ITERATION	ON POS SIG	POS RESULT RECOMP NEG RESULT, COMP NEXT INSTRUCTION	S
COMMENTS	O AQ. NUMBERS SO PROVIDE LEAD IN INPUTS AND CORRECT AT END. OMPLEMENT INPUT NUMBERS.	INDICAL		CHECK SIGN	CHECK S	GET POS CLEAR / MAKE F	MULTIPI MULTIPI	EXIT	POS RENEGREGATION	COMMENTS
Ŧ	PROVID CORRECTION			ONS	ONS	צרמר	רמר	SKZU	77	Ī
ပ	HULTIPLY A BY X, PRODUCT TO AQ. HULTIPLY A BY X, PRODUCT TO AQ. HULTIPLY HORKS ON POSITIVE NUMBERS SO PROVIDE LEA CALCULATE SIGN OF NEGATIVE INPUTS AND CORRECT AT CODE IS HRITTEN FOR ONES COMPLEMENT INPUT NUMBERS ************************************			K=0	INCK	GECK N=31 RQROE	RORDE		EXIT	ů.
w __	UCT TO A ITIVE NU ATIVE IN NES COMP						RPI			Ø
S	HULTIPLY A BY X, PRODUCT TO HULTIPLY MORKS ON POSITIVE POLCULATE SIGN OF NEGATIVE CODE IS HRITTEN FOR ONES GOD ***********************************			ď	a	׫	44	∢	∢ ♂	0
₩.	HULTIPLY A BY X, F HULTIPLY HORKS ON CALCULATE SIGN OF CODE IS HRITTEN FC		30X		ď		*		đ	6
⋖	MULTIPLY MULTIPLY CALCULAT CODE IS	CMP1		∢	×	ZERO	A ADD A	4	V *	
Ŀ	MU MU CA CO	2	ORG	⋖	8 4	ZE	4 4	1	-A -B SET	•
LABEL				MULT	.◆	•	+ 1		EXIT	LABEL
CYCLE					M &	m m m	מים,	n	n m	CYCLE
				0000	0000	301F 20F2	22F2 22F2	6000	2059	
HEX IMAGE				0F03	1 4423 SEC0	40C6 0805 5800) 5F05 7110	U 4105	U 8105 L 6A23	HEX IMAGE
HEX			0 30	030 0	031 U	032 U L 033 U	034 U	0.35	036 1	HEX

Figure 4-10. Example of Multiply Coding

										_													
	COMPLEMENT REP ETS BIT IN SML IF PRESENT F QUOTIENT	CHECK SIGN	IF HEG. COMPLEMENT A IF POS. LEAVE ALONE		SIGN IN F. SET	E X IN I, SET 2S ICATE THOS COMPLE	GET POSITIVE X		SHIFT. AQ LEFT 1 Test divide overflom	SET BIT 14 DIVIDE OVERFLO			HANY SUBTRACTS				SET ONES COMP.	COMPLEMENT QUOTIENT		CHECK REMAINDER SIGN	DONE NO CHANGE		COMMENT
Ħ	COMPLEMENT SETS BIT IN SE QUOTIENT	SNU	នន			SNU			scól	SNU		3 5	1 TO MANY	ระ			SNU			SNU	, J J	;	Ë
v	CHES ONES SIGN O				N=29	CLRF/1			ROL XN ROL XN	ROL XN		ROLXN POLXN	1 LEFT AND HAY BE	RROE			SETF/1		REHAINDER AND CORRECT A		EXIT	: : !	ပ
	TS OF X. USES OVERFLOW TEST TO CALCULATE		0.00	5								RPT	EFT AN						AND				v
S	CONTENTS OF X. HAKES OVERFLOM USED TO CALCUI									ASHL	9					ENT			INDER		⋖		6
6	TENTS KES OVE SED TO	14	<		c u	. н	×	TOM.	44	⋖ Ø	ON LO	~ <	A IS	44	•	QUOTIENT			REHA				
•	F GOT				σu	_		OVERFLOW	×	× 60	RATIC	××	T 10M	×	×	P		<i>a</i>	9	•			€
< <	VIDE AQ BY CONTENTS OF X. IS ROUTINE HAKES OVERFLON REGISTER IS USED TO CALCUL	<	. ⋖		>	××	×	FOR	« «	A SH1	DIVIDE ITERATION LOOP	<<	CORRECTION	⋖ ⋖	⋖	K SIGN	L.		K SIGN	H	<	•	<
L.	DIVIDE AQ BY THIS ROUTINE F REGISTER I	•	. 4		80 6	A C	- A -	CHECK	A SUB	A00 A+8	01710	A DD SUB	END	A 00	A 00	CHECK	⋖	CHP1	CHECK	EOR	4-	SET	L.
LABEL	DIVIDE AQ BY CONTENTS THIS ROUTINE MAKES OV F REGISTER IS USED TO	DIVIDE			•					• 1	•	• 1	•	+ 1	•	•		•		1	•	EXIT	LABEL
M F			. 10	•	•	m æ	m		m w	15 →		w w		æ m	•		•	**			m	m,	CYCLE
CYCLE		6		000	0000	3010	0002		0070 COF 0	C070 240E		0270		2000	000		5051	000		000	2037		
HEX IMAGE			4105 2000	00.	4A23 00		12 900%		5F05 0	7110 C 5615 2		7110 6		7110 2 5F05 4	7110 0		5640	. K		52 A 8	8105		HE X I HAGE
Ä				L 58		L 02	ر ج		2 J) D J		> -		27	9		_	, , =	,)	ے	
HEX		,	1 to 0		050	051	0 51		250	053		924		8 0	926	•	25.	057		150	80	059	HEX

Figure 4-11. Example of Divide Coding

M Code

Operation

S or blank

÷

Sequential addressing. If T field is left blank, TMASS will select the next sequential microinstruction either from from current pair or from next microinstruction pair. If the T field is not blank and does not contain a *L (select current lower instruction), the upper instruction of the next sequential microinstruction pair within current page of micromemory will be selected. (Note that location 00_{16} in current page follows location FF.)

J

Jump addressing. Next microinstruction pair will be selected from micromemory location specified by symbol or constant coded in C field of current microinstruction.

MASS will select the correct type of jump based on current program address and jump destination. If the jump address is in the current page, a within-page jump will be performed and the S field is available for coding. If the jump address is in another page, the assembler will use the S field to specify the page in a page-jump instruction. If T field is left blank, MASS will select correct upper or lower T value for the symbol.

R

Return addressing. Next microinstruction pair is selected from location in current page specified by contents of RTJ register. The four MSB bits specify the page and and the remaining eight bits specify the address within the page. The RTJ register must have been previously set up by an RTJ in the S field. If the current S field contains CLRNP, the next microinstruction pair is selected from page 0 as specified by contents of RTJ register.

T field must be filled in with an upper/lower selection.

T FIELD

The mnemonics in the T field specify the selection of the upper or the lower microinstruction from the next microinstruction pair for execution on completion of the current microinstruction, except *L. The final letter of the mnemonic specifies the upper (U) or the lower (L) microinstruction to be selected if the condition specified by the first letters of the mnemonic is true.

When micromemory is being read or written as an operand, the T field is used to address the referenced micromemory location and the upper instruction in the next sequential microinstruction pair is always selected.

The following T field mnemonics may be included in any microinstruction:

T Code	Operation
*L	Select the lower of current microinstruction pair. This operation overrides M field addressing mode.
U	Select upper of next pair.
L	Select lower of next pair.
KZU	If initial contents of the K register is zero, select upper of next pair. Cannot be used with C field INCK.

T Code	<u>Operation</u>
NZU	If initial contents of the N register is zero, select upper of next pair. Cannot be used with C field INCN.
NU	If output of ALU is negative on completion of current microinstruction, select upper of next pair.
ZL	If output of the ALU is zero on completion of current microinstruction, select lower of next pair.

The following T field mnemonics are usable only in format 1 as shown in Figure 4-2. The use of these mnemonics with other formats, such as a jump or N= or K= in the C field, will cause an assembler diagnostic.

T Code Operation

LQL If at start of this microinstruction the least significant bit of Q is 1, take the lower of next microinstruction pair.

INTU If there is an interrupt, and the corresponding mask bit in the mask register is set, execute the upper of the next microinstruction pair.

BTU If the bit selected by the least significant four bits of the C field is set, execute the upper of the next micro-instruction pair. Presently, only the following bits are selectable:

C Field Value	Test
2	Bit Register
5	Selective skip bit of FCR
9	Monitor/Program Mode Bit
ਾ F	Macro RUN/STOP Bit

COL If there is a carryout of ALU from arithmetic operation, take lower of next pair.

K7L If the least significant bit of K register is set, execute lower of next microinstruction pair; if clear, execute upper of next microinstruction pair.

OVFL If overflow exists, execute the lower of the next microinstruction pair; if not, execute upper of the next microinstruction pair.

ERL If any memory fault occurred, take the lower of the next microinstruction pair.

ASSEMBLER PROCESSING OF M AND T FIELDS

SEQUENTIAL ADDRESSING

If the M and T fields are left blank, the assembler will assume sequential addressing mode and will choose a T code in the object code output to execute the next sequential microinstruction. If the current microinstruction is an upper, the *L code will be inserted for the T field. If the current microinstruction is a lower, a U code will be inserted in the T field.

The example in Figure 4-12 shows assembly output of two sequences of code to show two ways of specifying sequential addressing.

In Figure 4-13, the instructions with NOP coded in the D field are not executed, but the other instructions with coding are executed. This example shows how it is possible to interleave two paths of program flow through one set of micromemory locations.

JUMP ADDRESSING

If a J is coded in the M field, the C field is interpreted as the jump destination address. The C field may contain a symbol or a constant. A symbol is carried as an actual micromemory location address and has an upper or lower property as well. A constant is interpreted as an upper micromemory address. If no T field value is specified, the assembler provides the correct T field value for the location addressed. However, the default T field selection is overriddened if the programmer specifies a value in the T field.

The assembler compares the page of the jump destination address with the page of the current microinstruction. If the page numbers are the same, a within-page jump is coded and the S field may be used for additional instructions. If the pages are different, a page jump is coded and the page number is extracted from the constant or symbol value and inserted

R.										**********				~		
	COMMENTS			ANDTE SAME MICRO CODE GENERATED.			COMMENTS	8u	COMMENTS	中央的现在中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央中央					COMMENTS	
	¥	9. H		C00E	Su Su	Set	Ħ	iressi	Ī	* * *	S	St	S	ţ	#	
	U			SAME MICRO			U	Sequential Addressing	ပ						U	
	S	SEQUENTIAL		NOTE S			Ø	Seque	W	**************************************	MON	MON	M ON		v	
C)	ED SEC	× 4 0	* *	׫	æ	0	Example of		IAL ADE	×	. ◀	ø	×	6	
	es .	GENERATED	æ	SEQUENCE.	a		a	Examp		***********************		ø			ത	
	4	ASSEMBLER GENERATED	« ×	**************************************	⋖	×	4	4-12.	⋖	essessessessesses FURTHER SEQUENTIAL	≪ .		* *	0	4	
	ı.	ASSE	< ∞ <	* *	₹ 8	⋖ .	i.	Figure 4-12.	L		⋖	80	4	ZERO	L.	
	LABEL	* * *	•		+ 1	•	LABEL	Œ	LABEL	***	•	•	•	•	LABEL	
	CVCLE		m m m		m m	m	CYCLE		CYCLE		8 M	m m	20 20 30	£ 00	CYCLE	
			2000		2000	0000			е 10 10		16 2000 10 2000	55 4000	3 2000	0000 90	HEX Inage	
	HEX I HAGE		5F06 5425 5EC3		5F06 5425	5EC3	HEX IMAGE		NEX Image		U 5F06 L 5800	U 5425 L 5800	U 5800 L 5EC3	U 5806	HEX	
	HEX	÷	0 000 U L 0 007 U		D 800	N 600	HEX		HEX		0 0 A 0	800	000	000	HEX	

Figure 4-13. Further Examples of Sequential Addressing

in the S field for the object code. The location within a page is coded in the C field of the object code. If the programmer has used a value in the S field and a page jump is coded, a diagnostic will be generated.

The example in Figure 4-14 shows four methods of arriving at a specific microinstruction.

RETURN ADDRESSING

Return addressing causes control to be returned to the microinstruction pair specified by the contents of the RTJ register. The programmer must specify a value in the T field to get a correct return location (upper or lower of the microinstruction pair). The RTJ register may be set any time by placing the mnemonic RTJ in the S field of a microinstruction. The address stored into the RTJ register is that of the next sequential micromemory word following the instruction with RTJ in the S field. Both page and address within a page are stored in the RTJ register.

The example in Figure 4-15 shows use of return addressing. In this example, a jump is made to the routine SUB which tests the value in the A register. Return is to the lower of the following microinstruction pair if the value in the A register is negative, and returns to the upper of the pair if the value is positive.



1						
COMMENTS	电电电电路 化二甲基苯甲基苯甲基苯甲基苯甲基苯甲基苯甲基苯甲基甲基苯甲基甲基甲基甲基甲基甲基甲					CONHENTS
Ŧ	* *	7733		っっ も		H
ပ	* *	LOCA LOCA LOCA LOCB		LOCB LOCB LOCB LOCA		ပ
S	* * * * * * * * * * * * * * * * * * *		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			v
0						o ,
8	A DORESSING				⋖ 50	.
⋖ .	A 00 F				LOCA	⋖
u_	AUUL				EQU	u.
LABEL	中国在中国中国的中国中国中国中国中国中国的中国中国中国中国中国中国中国中国中国中		+1004		LOCAA	LABEL
CYCLE		. M M M M	m m	m m m m	,	CYCLE
0 0		2003 2003 2003 2003	0000	\$000 \$000 \$000 \$000	? • •	
HEX IHAGE		U 9800 U 9800 U 9800	U 5800	U 9800 U 9800		HEX IMAGE
HEX		400 L	1 204	+0+	4 0 3 4 0 3	HEX

Figure 4-14. Examples of Jump Addressing

. u + i

OYAG				6	CR P												•	2 - 3 - 6 ₁₀
	化化学 经存储证据 医皮肤	\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$		ANK	INK (S ERROR) 1K N PAGE 4 4 (S ERROR)		IN PAGE 4	(Cont.d)	COMMENTS		7 1 1	RETURN LOCATION LOWER	RETURN LOCATION (A -)	•	TEST A	RETURN LOWER IF A IS +	RETURN TO UPPER IF A IS . ELSE RETURN TO LOMER	O
COMMENTS	10 10 10			JUMP IN BANK JUMP OTHER B	JUMP OTHER BA JUMP THIS BAN JUMP TO OU II	START PAGE 4	INSTRUCTION	Jump Addressing (Co	E E	中央中央的共享公司 医克勒氏性 医克勒氏性 医克勒氏性 医克勒氏性 医克勒氏性 医克勒氏性 医克勒氏性 医克勒氏征 医克克氏征 医克勒氏征 医克克氏征 医克克克氏征 医克克氏征 医克克克克克氏征 医克克氏征 医克克克氏征 医克克氏征 医克克克克氏征 医克克克氏征 医克克克氏征 医克克克氏征 医克克克氏征 医克克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克克克氏征 医克克克氏征 医克克克克克氏征 医克克克氏征 医克克氏征 医克克克克克克氏征 医克克氏征 医克克克氏征 医克克氏征 医克克氏征 医克克氏征 医克克氏征氏征氏征 医克克氏征氏征性原生原生原生原生原生原生原生原生原生原生原生原生原生原生原生原生原生原生原生	٦-	22 GZ	SUB2 J		D X	RU	RNU	Return Addressing
Ħ	14 14 14 15 17 17 17 18	10 10 10 10 10 10 10 10		77				Jump Ad	ى د	********************	RTJ SUB1		RTJ SU					Return
ပ	1	1		A LOCNO				Examples of	0			N 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0						les of
S	1	s \$}		LBEA	LSEA	!	MON		æ	ADORESSING								المريا
٥		A STATE STAT		×	×			re 4-14.	∢	RETURN ADORE					⋖		A 27	re 4-15.
8			200X	4	⋖	X007		Figure	اد. د	**************************************					≪		8	Figure
۱.		JUMP E	ORG	⋖	⋖	ORG			LABE		+ 1	+ 1	• • 1	+ 1	•	. + 1	SUB	·
LABEL					LOCAB		LOCN2		CYCLE	· .	2512 3 2000 3	2000 3	2514 3 2000 3	20000			0000	
CYCLE	COUNT				mmmm		۳ ۱		HEX IMAGE		U 9800	U 5600	U 9800	L 5800	:		- -	
	INAGE			U 9F06 4501	L 9800 3400 U 9F06 2500 L 9800 2000 U 9800 3400	n n 2 n n 2 n n 2 n n 2 n n 2 n n 2 n n 2 n n 2 n n 2 n n 2 n n n 2 n n n 2 n n n 2 n n n 2 n n n 2 n n n 2 n n	0 2800 0000		H K		0 0 E	0 0 6	010	011		013	710	
HEX	Ŧ		200		202	6	9 0 9 9											U



The Memory Interface performs the following basic functions:

- Converts a programmer address to a physical address.
- Decouples microprocessor and memory timing, using look-ahead.
 - Formats address and data for full-word, half-word and character operations.

MEMORY ACCESSING

Supporting up to 4 million words of main memory and providing capabilities for virtual memory, memory protection and dynamic allocation is achieved by logically dividing memory into blocks, or pages. Each page consists of 2048 contiguous words. To access a maximum memory of 4 million words requires 2048 pages and an address of 22 bits. The required 22-bit address is divided into an upper and lower portion of eleven bits each. The upper portion selects one of the 2048 possible pages, while the lower portion selects one of the 2048 words within the selected page. The lower portion comes from 11 word address bits of the macroinstruction. The upper portion is supplied from an entry in the page index file.

Figure 5-1 shows the creation of the actual physical address from the macroinstruction operand fields and the page index file.

PAGE INDEX FILE

The Page Index File is a 1024, 17-bit bipolar random access memory. The Page Index File is micro-accessible (see WPF, RPF instructions) and is logically divided into 32 groups of 32 entries each. The CPU Program State Register, when in State Mode (or 5 bits from the macroinstruction, when in Direct Mode), selects one of the 32 groups. Five bits from the upper part of the word address in the macroinstruction selects one of the 32 entries in the group. This was done to help assist in the emulation of the MP-60 instruction set.

17328900-02 5-1

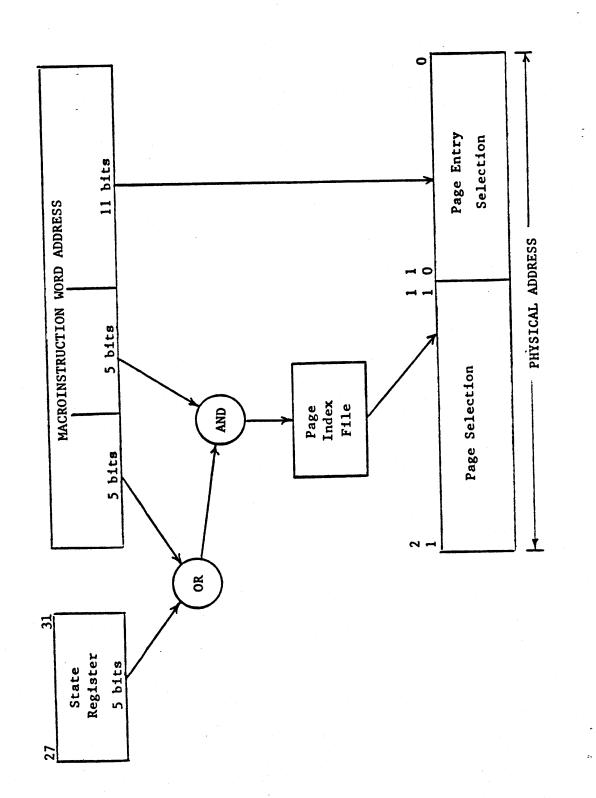


Figure 5-1. Creation Of The Actual Physical Address





Each of the 32 groups represents an individual state. Each of the entries in that group represents a logical page of 2048 words, so a state has a logical maximum size of 65K words.

Along with providing page addresses, the Page Index File supplies odd parity information to the memory protect logic.

MEMORY PAGE ADDRESSING

As shown in Figure 5-2., each Page Index File entry contains an 11-bit address for a particular page of main memory. Together with 11 bits from the word address in the macroinstruction, this provides addressing for over 4 million words of main memory. Although the Page Index File can hold only 1024 of the 2048 possible page addresses at any one time, the file can be changed by microprogram instructions to allow access of the maximum memory present. Actual memory address generation is discussed under MEMORY ACCESSING.

The page address parity logic generates an odd parity bit on the 11-bit page address as it is loaded from selector 1. This bit is stored at bit position 15 when the page address is stored in the page index file. When this page file entry is accessed, the parity bit is sent with the page address to the holding register.

PAGE STATUS

Bit positions 17 and 20 hold the MODIFIED and ACCESSED status bits respectively. The page status logic monitors activity of the Page Index File. When a Page Index File entry is accessed for a read operation, the page status logic sets the ACCESSED bit in that entry. On a write operation, both the ACCESSED and MODIFIED bits are set in the corresponding Page Index File entry.

MEMORY PROTECT LOGIC

Bits 16, 18 and 19 specify the protect state associated with each page. These bits are monitored by the protect logic to ensure that no illegal memory operations are allowed to occur. These bits are used as follows:

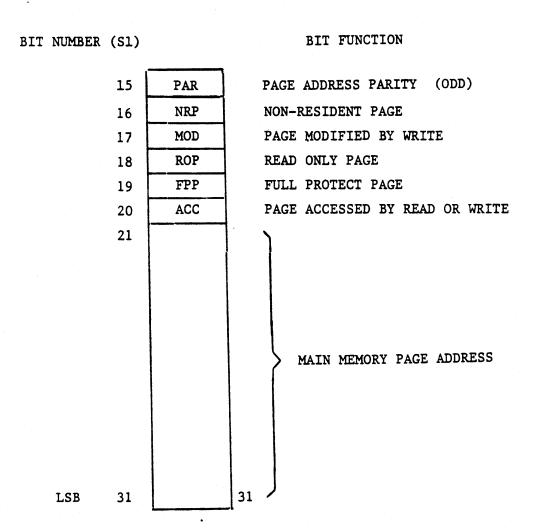


Figure 5-2. Page Index File Entry Bit Assignments

- Non-Resident Page Bit 16 indicates that the page is physically or logically not present in the system and cannot be accessed.
- Read Only Page Bit 18 indicates that the page may only be accessed by read instructions.
- Full Protect Page Bit 19 indicates that the page is not available for any access.

PROGRAM STATE REGISTERS

The memory interface contains one (1) CPU State Register and four (4) DMA State Registers, one for each DMA port. These state registers perform the following functions:

- Provide the upper half of the Page Index File address for main memory addressing.
- Provide mode control.
- Report fault conditions.

CPU STATE REGISTER

The CPU State Register is accessible by using the WSR, RSR mnemonics in the C field. The CPU State Register bit assignments are shown in Figure 5-3 and are used as follows:

- Mode Control Bits 20-22 are mode control bits for the three lookahead address/data register pairs of the CPU. Setting a bit puts the corresponding lookahead register pair in the write mode. Clearing a bit puts the corresponding lookahead register pair in the read mode.
- Addressing Bits 27-31 form the upper portion of the Page Index File address, selecting one of 32 groups of 32 Page Index File entries. Refer to MEMORY ACCESSING section for a discussion of address generation. Bits 27-31 are also routed to the backpanel for use by the transform module. These bits are undefined after a master clear.

- Fault Reporting Bits 15-26 are used to report faults that have occurred during CPU memory operations. The bits are set by the fault detection hardware and may be cleared by using a read-modify-restore microcoded sequence.
- Bits 13-14 These bits are constantly updated and have no significance until a DMA error occurs (bit 23 is set). When this happens, they contain the number of the DMA port which caused the error. Bit 23, when set, prevents any changes to bits 13-14 until a WSR (Write State Register) function is issued to the CPU State Register. These bits are undefined after a Master Clear.
- Bit 15 This bit will be set when a timeout error occurs, along with one of the lookahead error bits (bits 23-26) indicating which one generated the timeout error. This bit is cleared with a WSR instruction or a Master Clear.
- Bits 16-19 These bits, all of which could be set simultaneously, indicate a parity error on the corresponding data character(s). One of the lookahead bits (bits 23-26) would be set at the same time, indicating which lookahead register set caused the error and, at the same time, blocking any further setting of data parity error flags by subsequent errors. A Master Clear or a WSR instruction will clear these bits and remove the inhibit update signal.
- * Bits 23-26 These error flags are set on any error to indicate which lookahead register set generated the error. In addition, these bits prevent any further update of any other error information until the CPU clears these bits with a WSR instruction or a Master Clear. Any bit set will inhibit the updating of state register bits 16-19 (Data Parity bits). Bit 23 will inhibit the updating of state register bits 13-14 (DMA port number in error). Presently, only lookahead 1 is testable using T field logic. ERL in the T field would force execution to start in the lower of the next microinstruction pair on a fault.

BIT NUMBER	BIT ASSIGN	MENT
13-14	DMA PORT NUMBER IN	ERROR
15	TIMEOUT ERROR	-
_ 16	CHAR 0	DATA PARITY ERROR
17	CHAR 1	DATA PARITY ERROR
18	CHAR 2	DATA PARITY ERROR
19	CHAR 3	DATA PARITY ERROR
20	LOOKAHEAD 1	READ/WRITE BIT
21	LOOKAHEAD 2	READ/WRITE BIT
22	LOOKAHEAD 3	READ/WRITE BIT
23	DMA (LOOKAHEAD 0)	ERROR FLAG
24	LOOKAHEAD 1	ERROR FLAG
25	LOOKAHEAD 2	ERROR FLAG
26	LOOKAHEAD 3	ERROR FLAG
27-31	UPPER 5-BIT PAGE I	NDEX FILE ADDRESS PORTION

DMA STATE REGISTERS

The DMA State Registers are used to perform two functions for the CPU: they provide Page Index File information for main memory addressing and transfer mode control (half word/full word) information to the DMA port logic. The selection of the DMA State Register to be read or written is determined by the least significant two bits from S3. All mode bits and set of address bits are written to a Data State Register with a WSR instruction. All mode bits and one set of address bits are read back with each RSR instruction. Bit assignments are shown below.

BIT NUMBER	BIT ASSIGNMENT
23	TRANSFER MODE BIT PORT 0
24	TRANSFER MODE BIT PORT 1
25	TRANSFER MODE BIT PORT 2
26	TRANSFER MODE BIT PORT 3
-	
27-31	UPPER 5-BIT PAGE INDEX FILE REGISTER PORTION

- Bits 23-26 These are transfer mode bits selecting either half word (0) or full word (1) operation for the corresponding DMA port.
- Bits 27-31 Upper 5-bit Page Index File address portion.

CODING EXAMPLES FOR MEMORY INTERFACE

Figure 5-4 shows coding for reading and writing the contents of a Page Index File entry. Reading and writing the program and DMA state registers are shown in Figure 5-5. Reading physical location 00 using state 2 and Page Index File entry 34 are shown in Figures 5-6 and 5-7. Placing the address of a main memory word within the word for the first 65,536 locations is shown in Figure 5-8. Figure 5-9 is an example of using a TK transform and lookaheads to perform a main memory cycle. Figure 5-10 shows a TMA transform and the processing of interrupts.



-	* * * * * * * * * * * * * * * * * * * *		* * *	-	* * *				
COMMENTS	encent		papasananananananananananananananananana	PAGE FILE TO A-REGISTER	raterrecerrecerrecerrecerrecerrecerrecer		PF=X X=X+1 DOES X = 1024		COMMENTS
Ħ	INS AD		INS AD		ADDRESS	3	12	= ₇	Ŧ
ر ن	X-CONTAINS	N PF	X-CONTAINS X-CONTAINS	RPF	SERVICE TO THE PAGE FILE ENTRY WITH TTS ADDRESS OF THE PAGE FILE ENTRY WITH TTS ADDRESS OF THE PAGE FILE ENTRY WITH TTS ADDRESS OF THE PAGE FILE ENTRY WITH THE PAGE FILE FILE ENTRY WITH THE PAGE FILE FILE FILE FILE FILE FILE FILE FIL		WPF CNE C1024	LOAD	ပ
vs	ENTRY		ENTRY.		ENTRY				w
0	FILE E		FILE EP	< 4 €	FILE	>	· ××		0
65	MRITING A PAGE FILE	×	A PAGE F	*	ATTING THE PAGE		9 9 8 8		₩.
æ	ING A	⋖	READING A	HEM	ING T	31	×××		€ ,
L.	MRITING	6	READ	8 4		SET EQU	A DO A B B		L.
LABEL		•	* * *			ONE C1024	L0A3	•	LABEL
CYCLE		m		m m		M		юю	CYCLE
		002C		202E		200		2000	
HEX		5518 002C		5418 5FC5		5806	SEC 0 7 006 5 006	5800	HEX I HAGE
HEX		0 37 U		037 L 038 U		01F 015		0 38 U	HEX

Figure 5-4. Examples of Reading and Writing a Page Index File

17328900-02

÷

MT COMMENTS	RESPONDENCIONE CONTRACTOR REGISTER AND RESPONDENCIONE ADDRESS A-DATA CONTRACTOR CONTRACT		在我们的现在分词,我们是不是不是有的,我们们们的现在,我们们的,我们们的,我们们的,我们们的,我们们的,我们们的,我们们的,我们们		READING THE CPU STATE REGISTER, HOVING IT TO DMA 3 STATE AND INCREMENT IT BACK INTO THE CPU STATE REGISTER READING THE GPU STATE AND STATE REGISTER STATE AND STATE AND STATE AND STATE REGISTER REGISTER FRANCES OF THE FRANCES OF	A=4	STATS LIGO=X	X + X + X + X + X + X + X + X + X + X +	CPU STATE=X	HT COMMENTS
ပ	**************************************	MSR	* * * * * * * * * * * * * * * * * * * *	RSR	ER, HOVING IT TO	FOUR	RS S	ONE	HSR	v
v	STER		REGISTER		REGIST BACK					w ·
0	E REGI		E REGI	⋖	E CPU STATE INCREMENT IT	4	: :	××		•
œ	peessersersersersersersersersersersersers	×	E STATE		E CPU INCRE	a	3	₹ ₩ 9		₩
<	ING TH	⋖	ING THE	E W	READING THE STATE AND	29		∢ ×		<
L.	WRIT	8	READING	• ≪	READ STAT	SET	b	8 0 0 0	ZERO	t. .
LABEL						FOUR				LABEL
CYCLE		m		H 0			m m	203		CYCLE
HEX I MAGE		U 5518 0020		5800 2026 SFCS 000			5415	553E 2020	5800	HEX IMAGE
HEX	<u>-</u>	0 3C O		0 3C L 0 30 U		010 01F	030 L		934	HEX

Figure 5-5. Examples of Reading and Writing State Registers

	* * * *								
COMMENTS	enscheterencenterenterenterencenterenterenter	CLEAR CPU State Register		CLEAR PAGE	FILE ENTRY	LOAD ADDRESS REGISTER 1		INFORM AMI-NEED DATA Done, Data to A-Reg.	COMMENTS
Ī	# C D # # D # # D # # D # # D # # D # # D # # D # # D # # D # # D # # D # # D # # D # # D								Ħ
	O USI					5	HERE		
ပ	N ZER	M SR		HPF		LHA1	05 0		ن
S	CATIC						INOM 9	R01	S
0	LEMORY LO	« ×					DELAY OR PROCESSING MOULD GO HERE	⋖	٥
©	ICAL			×		×	OR PE		.
⋖	PHYS	⋖		¥)EL AY	HE	4
L ,	READ	ZERO ZERO		8		REN	ANA	⋖	i.
LABEL	* * *		, •		• •		* * *		LABEL
CYCLE		mm		. PO		M		mm	CYCLE
		2020		0020		2000		0000	
HEX Image		9065 7 9080 0 9802		041 U 5518 002C		041 L 5418 2000		042 U 5800 L 5FC5	HEX IMAGE
× _		ë D 4		.n		ب ب		ž U	<u>ہ ×</u>
HEX		50		70		70		70	HEX

Figure 5-6. Example of Reading Location O Using State O

=

COMMENTS	**************************************	BIT FOR BG VALUE OF 2 BIT FOR BG VALUE OF 32 A=2 STATE=2 A=32+2	PAGE (34) =0	BIT FOR BG VALUE OF 6192 x=00010000000000000000 LOAD ADORESS REGISTER 1		INFORH AHI-NEED DATA DONE, DATA TO A-REG. COMMENTS
Ī	9 7 M					Ï
	HENORY LOCATION ZERO USING FE 2 AND PAGE FILE ENTRY 34.			CE192 LHA1	OR PROCESSING WOULD GO HERE	
ပ	ZER	TWO MSR C32	HPF	93	09 0	. U
· s	**************************************				MON	RO1
	LOC	•			SING	_
0	HORY 2 AN	< <		× ×	OCES	∢ 0
co	AL ME	98 98		9 ×	OR PR	00
4	**************************************	0 9 A 4	⋖	0	DELAY	Z 4
ie.	READ PHYSICAL ME CPU STATE	SET SET B ZERO A+B	ZERO	SET B REN	ANY	≪ 1 ⊾
LABEL		TH0	• 1	C8192		LABEL
CYCLE		mmm	m	m in		CYCLE COUNT
22		001E 2020 001A	202C	2000		00
HEX I MAGE		5415 5900 5715	2900	U 5416 0		U 5800 CL 5FC5 N
HEX		01E 01A 043 U	044 1	012 045 0		D46 THEX
	•					

Figure 5-7. Example of Reading Location O Using State 2 and Page Index File

COMMENTS	protestatestatestatestatestatestatestatest	BIT FOR BG VALUE OF 65536 CLEAR CPU STATE REGISTER	INITIALIZE PAGE REGISTER		NRITE MEMORY X=X+1 TEST COMPLETE		DONE		COMMENTS
Æ	144444 00RESS	5	3		7.0		רר		Ē
	THEIR AL	N=16 MSR	WPF		HRFW ONE C65536		DONE		ڻ ن
S	S NITH		RPT						S
0	CATION	∢ ×	<		, ×				0
6	CAL LO		98		9 8				©
⋖	weekekkekke MRITE PHYSICAL ************************************	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	44		×××				4
L	WRITE	SET SET ZERO ZERO	A ADD		A 000 A . 8			EQU	u.
LABEL	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	ONE C65536	 •	•	+8EGIN	•		DONE	LABEL
CYCLE		mm	m w		mwm		10	າ	CYCLE
55		1010 2020	002C 221F		0020 201F 200F		20402	640	
HEX IMAGE		5906	U 5F00 L 7115		SECO 7006 DCCF		9800		HEX IMAGE
		ا د			2 7 2		9	ں ر	×⊒
HEX HE		01F 00F 047	8 7 0		8†0 6†0		840	040	HEX

Example of Placing a Location's Address in the Location Figure 5-8.

-

Ξ

ပ

S

0

æ

LABEL

CYCLE

HEX Inage

HES

CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0

IK TRANSFOPM / LOOKAHEAD EXAMPLE

IDENT

######################################	EXFCUTE AN ASSEMULY INSTRUCTION WHICH AN INCLUSIVE ON THE ELEMENT OF THO ARPAYS. THE ASSEMULY INSTRUCTION IS ASSUMED TO BEEN PEAD FROM MAIN NEMORY AND IS RESIDING IN THE IXT REGISTRA	THE A FIELD (BITS 11-15) OF THE ASSEMBLY INSTRUCTION SPECIFIES THE REGISTER WHICH CONTAINS THE ADDRESS OF ONE OF THE ARRAYS AND THE ATTAINS TON ADDRAY. THE B FIFLD (BITS 16-20) SPECIFIED THE	REGISTER WHICH CONTAINED THE ADDRESS OF THE SECOND ARRAY. THE GRISTER WHICH CONTAINS THE LOOP COUNT (RITS 27-31) SPECIFIED WHICH REGISTER CONTAINS THE LOOP COUNT (I.E. HOW MANY ARRAY ELEMENTS TO INCLUSIVE-OR). ESSENTIALLY THE CODE DOES THE FOLLOWING:	DO 10 I=1,CF Jarray(af)=Iarray(af).or. Iarray(bf)	IT IS ASSUMED THAT THE FOLLOWING MAS ALREADY OCCURRED:	LWAI=A ARRAY ADDRESS X REG=ADDRESS OF A ARRAY
**************************************	EXFCUTE AN ASSE ELEMENT OF TWO BEEN PEAN FPOM	REGISTER WHICH	REGISTER WHICH CONTAINED THE C FIELD (RITS 27-31) SPECIFI COUNT (I.E. HOW MANY ARRAY E THE CODE DOES THE FOLLOWING	10 146	IT IS ASSUMED	×

K=8 FIELD REG NUMBER	FETCH B ARRAY ADDRESS K=C FIELD REG NUMBER(LOOP COUNT)	DROP LOOP COUNT, NEXT LOWER IF ZERO		BUMP A ARRAY ADDR READ LAST A ARRAY ELEMENT	I=A ARRAY ELEMENT, GO FINISH OFF
		72		ب	· ~
TK/BF	LHAS	TK/8F			A08.20 J
B FIELD DECODE C FIELD DECODE				R01 R01	
ELO.0 ELO 0		⋖ .		L	-
2 8 FI	F1 .	F1	A=LOOP COUNT K=B FIELD REG NUMBER X=LWAI=A ARRAY ADDRESS LWA3=B ARRAY ADDRESS	×	HEN
77			UNT REG ARRA		
60n	•	-	PCO FIELD A1=A	ż	
£ 5			+ A=100P + K=B FI + X=LWA1 + LWA3=B	+ AOR. 10	
•	n m 1	m .co		3 M	m
	1032 202C	1037 F032		002 U 78C7 4400 L 5800 2400	003 U 95C2 2006
	5 T 8 D	5800		7867	9562
200 200 200		2 2 -		02 U	D 60
66	ā	ō			0

Figure 5-9. TK Transform/Lookahead Example

DIAGNOSTICS

COMMENTS

Ħ

ပ

S

LABEL

CYCLE

HEX IMAGE

HEX HH:

I=A ARRAY ELEMENT, LWA1=A ARRAY ADDR BUMP A ARRAY ADDR INCLUSIVE-OR A, B ARRAY ELEMENTS

LHA1 LHA3

R03

X=LWA1=A ARRAY ADDRESS LWA3=B ARRAY ADDRESS F=A ARRAY ADDRESS+1 K=8 FIELD REG NUMBER

A=LOOP COUNT

003 U 95C2 2006

003 L 55EA 2024 004 U 7984 032C L 568F 2000

AOR.10 JZL DECREMENT LOOP COULT INTO HER F LAST LAST HRITE RESULT INTO COULT	A AOR.10 RD3 F LMA2 D S C
RD3	F RO3 LMA2 O S C MT
F LMA2	F LHAZ LHAZ O S C MT

Figure 5-9. TK Transform/Lookahead Example (Cont'd)

-

DIAGNOSTICS

COMMENTS

H

CYBER/MP-60 MICRO ASSEMBLER PSD VERSION 1.0 TMA / INTERRUPT EXAMPLE

LABEL

CYCLE

HEX Inage

HEX

Example
Interrupt
TMA Transform/Interrupt E
TMA
Figure 5-10.

DIAGNOSTICS

IDENT THA / INTERRUPT EXAMPLE

MASS CONTROL CARD

The MASS control card causes the MASS Assembler to be loaded and executed under control of the MPX operating system. This appendix describes the parameter options available to the programmer.

The card format is as follows:

*MASS(I=u, L=u, P=u, X=u)

Parameter definitions:

• I=u

Source input is from logical unit or file u. If =u is absent, standard output is assumed. Input must be specified.

• L=u

Assembly listing is on logical unit or file u. If =u is absent, standard output will be used. If this parameter is absent, no listing is generated; however, a listing of error lines will be output to standard output.

• P=u

Deadstart binary object output is on logical unit or file u. If =u is absent, standard punch is assumed. If P is absent, no binary output is generated.

• X=u

Pure binary output is put on logical unit or file u. If =u is absent, standard load and go file is assumed. If X is absent, no binary is generated. u, if present, must specify a disk file or magnetic tape unit.

17328900-02

EXECUTION ON A CYBER 70/70L/6000 SERIES COMPUTER

The following program call card causes execution of the MASS Assembler on a CYBER system under NOS or NOS/BE. It assumes the MASS Assembler is part of the system library.

MASS, P₁, P₂, P₃, P₄.

P ₁	is the name of the file on whi	ch the
•	microprogram source resides.	The default
	is INPUT.	

- P₂ is the name of the file on which the assembler writes the source listing. The default is OUTPUT.
- p₃ is the name of the file on which the assembler writes the object output. The default is PUNCH.
- p4 is the name of the file on which the assembler writes the simulation object output. The default is SIMFIL.

Example:

For source cards in the job deck, punch on file DWD, no simulation object output, standard listing, the program call card is:

MASS,,,DWD.

FORMAT OF DEADSTART OUTPUT DECK

The deadstart deck contains both the data to be loaded into micromemory and control characters to direct the loading. The data is binary generated by MASS of a source program and is grouped 4, 32 bit words per card and punched in every other column in hollerith.

The control characters of the deadstart deck are directed to the Function Control Register (FCR), which has access to the CPU similar to the way switches on a conventional panel have access to the CPU. Table B-1 shows the control characters that are issued to the FCR to initialize the files, the registers and micromemory. See the BASIC SYSTEM CONTROLLER MODEL 65109 HARDWARE REFERENCE MANUAL for additional and more detailed information on the FCR and the operation of the panel interface.

The typical load sequence to enter a deadstart deck into the MP32 is to put the deadstart deck into the card reader, do a Master Clear and begin loading by enabling bit 4 of SM2 (usually connected to a switch).

B - 1

TABLE B-1. DEADSTART DECK FORMAT

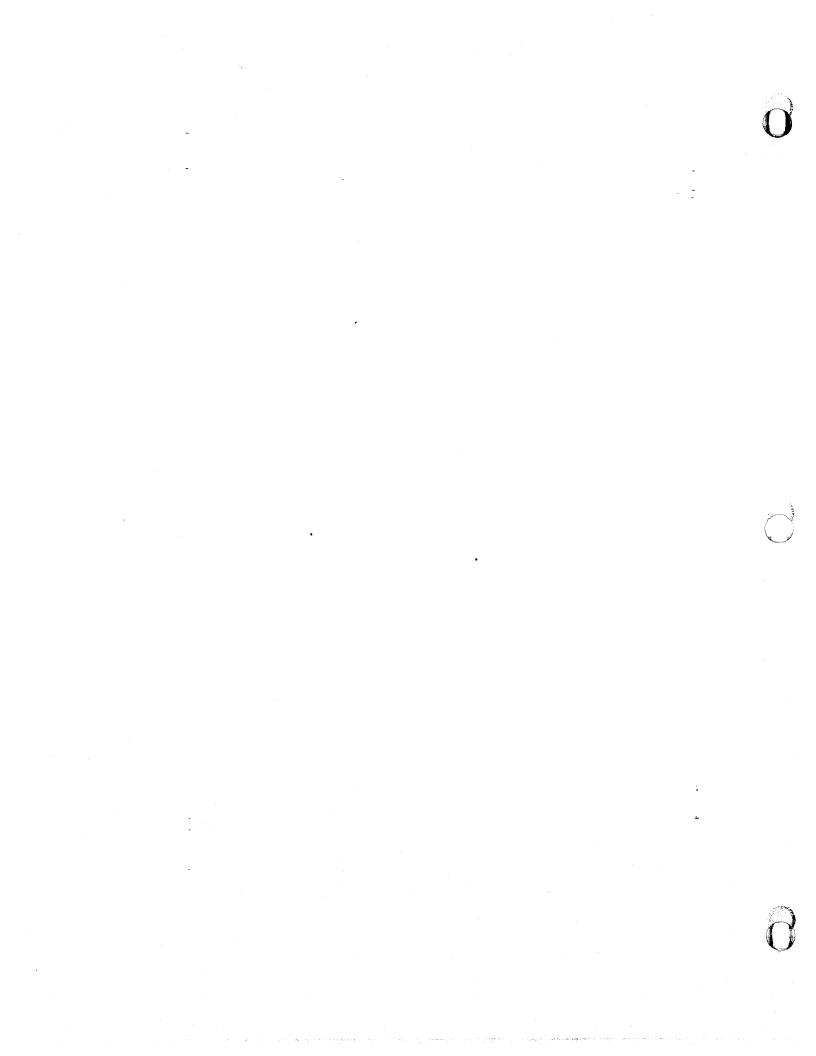
TYPE	CARDS	MEANING
1	K1008900G	Set up FCR register. Select N register for display 0. Select FCR for display 1. Select MICRO mode. Suppress console transmit. Enable micromemory write.
2	L0000G	Clear N register.
3	J02G	Select K register for display 0.
4	L0000G	Clear K register.
5	J0CG	Select micromemory for display 0.
6	L	Begin load of micromemory.
7	Data	Micromemory data in blocks of 256 32-bit micro- instructions. Each instruction is terminated with a G. The number of instructions per card depends on the spacing.
8	J01G	Select N register for display 0.
9	LXXXXG	Set N register to the number of 256 instruction blocks loaded.
for th	e remaining partia	re repeated until all full blocks are output. Data l block is then punched using cards 5 through 9, ontain less than 256 instructions.
10	K9A088000G	Set up new FCR register. After all data has been punched:
		Select RTJ register in display 0. Select SM2 register in display 1. Select MICRO mode. Suppress console transmit. Disable micromemory write.
11	K0000000G	Clear SM2. This card stops the deadstart hardware from reading more cards. Even though the MP16 has only 16 bits in the SM2 register, 32 bits of zeros (eight characters) are punched on the card to ensure total clearing of SM2 in case the machine is an MP32.

DEFAULT CODES GENERATED FOR BLANK FIELDS

Each field of the assembly input may be left blank, with the following values substituted by MASS:

<u>Field</u>	Default V	<u>'alue</u>
F	A	If A field contains a mnemonic and B field blank.
	В	If A field is blank and B field contains a mnemonic.
	ZERO	If both A and B fields are blank.
A	X	
В	X	
D	NOP	
S	NOP	
C	00	
М	S 00	If C field is not K= or N=. If C field is K= or N=.
T	Selected	to provide correct sequencing or jumping.
	*L	If upper instruction and M field is S or blank.
	U	If lower instruction and M field is S or blank.
	U	If M field is R.
- -	U	If M field is J and C field is constant.
	U	If M field is J and C field is a symbol with upper quality.
	L	If M field is J and C field is a symbol with lower quality.

17328900-02 C-1/2



SELECTING NONCONFLICTING MNEMONICS

The proper selection of a mnemonic for a field depends on the mnemonics used in the other fields (Table D-1. lists permissible combinations or codes). This is an inherent characteristic of the MP and is a result of maximizing the information content in the instruction repertoire. This means that the most frequently used operands require less space than the less frequently used operands.

TABLE D-1. LEGAL F, A, B, D AND S COMBINATIONS

F FIELD	A FIELD	B FIELD	D FIELD	S FIELD
Arithmetic or logical	A	В	D	Unu s ed
	A *	В	D	AP
	A	В'	D	ВР
	A	В	ס'	DP
	A'	В	ם'	APDP
	A	В	D''	DPP
	A	В	DD	ממ
Shifts or scale	Ъ	ъ	ъ	Unused
	Ъ	Ъ	NOP	Unused

Where:

Ъ

is a blank field

A, A', B, B', D, D", DD

are types of A, B and D field codes

AP, BP, DP, APDP, DPP, DD

are values supplied by the assembler

Each instruction also consists of a maximum of four independent and concurrently executed functional operations; this further reduces the effective instruction execution time. Their associated fields are also dual, in the sense that they can be used to specify less frequently used operands for other fields. The assembler flags ambiguously coded instructions with a diagnostic. See Figure D-1 for examples of conflicting mnemonics.

	化多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多多	*******		1	1						
DIAGNOSTICS	COMMENTS	Ī	ပ	S	o	60	4	t. .	LABEL	CYCLE	ist
										•	

SAS

SAS SAS SAS

HEX CY IHAGE CO	CY GL E.	LABEL	u.	4	60	0	S	ပ	ī	COMMENTS
		* * * * * * * * * * * * * * * * * * *	EXAMPLES BLANK FIE	ES OF C	CONFLI	CONFLICTING HNE	MNE MO	NIC SELEC	CTION A	AHPLES OF CONFLICTING MNEMONIC SELECTION AND ASSEMBLER ERROR CODES ** ANK FIELDS ARE AVALIABLE FOR USAGE
		V ****	9 4	ANU	F 15.0		7 *	****	*****	CONT. T.L.J. J. S.
•	u		00	۵	F.2		HALT			LEGAL
	n 4	•	V	. I	F2	, ,,,	HALT			ILLEGAL
	ع د		A 00	H	F2	—				LEGAL
2800			4	•	CRIJ	×	,			ווייים אוני אייים אוני
000	و.		⋖	4	CALJ	×	LBEA			ILLEGAL
2900	m		⋖	⋖ :	× :	I	4 4 0			TLLEGAL
900	m		⋖	⋖ :	× (Į,	44 154			I EGAL
2400	د ج		A 4	SH1	g 0	SM1	RIJ			ILLEGAL
	•		•	!		ı		1	1	
		******	*****	***	****	*********	* * * * * * * * * * * * * * * * * * * *	****		化化化物 化化化物 化二氯化物 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性 医多种性
				AND C PIELU	****************	111111111111111111111111111111111111111	*	******	*****	7-UNFILLS=3. Anteracent
3021	m							N=33	· >	LEGAL
	(E 12	101	ILLEGAL
3021	n m	LA81						526	rar	LEGAL
9966	, ,							LAB1	1101	ILLEGAL
7	•						1			
			* *	AND C FIELD C	essectestestestestestestestestestestestestest	FLICT				
					3					ILLEGAL
2008	m m				zz			108		LEGAL
		* * * * * * * * * * * * * * * * * * * *	THINS	FUNC	FUNCTION AND	4D A OR	8 F.	sereseresereseresereseresereseresereser	LICT	**************************************
2000	ν ν • • Σ Σ		ALEA	⋖	ø					ILLEGAL Illegal
		1 88:1	L	•	· œ	0	S	ပ	H	COMMENTS
	COUNT		. ·	L	ı					

* Figure D-1. Examples of Conflicting Mnemonic Selection and Assembler Error Codes

DIAGNOSTICS

ASF BSF

CBC

IIL

TIL

The output listing produced by MP MASS consists of a side-by-side listing of micromemory location, machine language, source card image, error codes and sequence number.

The information is assigned to the following columns of the listing.

Column	Contents and Meaning
1	Standard print format control.
2-4 (MML)	Micromemory location of assembled instruction, or value specified in EQU or ORG command. This is a three-digit hexadecimal number, with the
	first digit specifying micromemory page and the next two digits specifying location within the page.
5	Micromemory overlap indicator. This column is blank if micromemory location is used only once in microprogram. This column contains 1 if
	micromemory location had contained information previous to assembly of the line of code causing overlap. A count of the number of micromemory overlaps is included on the last page of assembly listing for information.
6	Upper/lower indicator. U specifies that micro-instruction is assembled to the upper micro-
	instruction of location specified in columns 2 through 4. L specifies that the microinstruction is assembled to the lower microinstruction.

Column	Contents and Meaning
8-16	Assembled microinstruction in hexadecimal code.
(IMAGE)	
19	Timing information. MASS calculates execution
(CYCLE)	time for each microinstruction and indicates
COUNT)	time as the number of minor cycles to execute
•	the statement.
24-103	Source card image.
104-113	Sequence number supplied by COSY package (if applicable.
115	Diagnostic error codes generated by MASS for source statement.

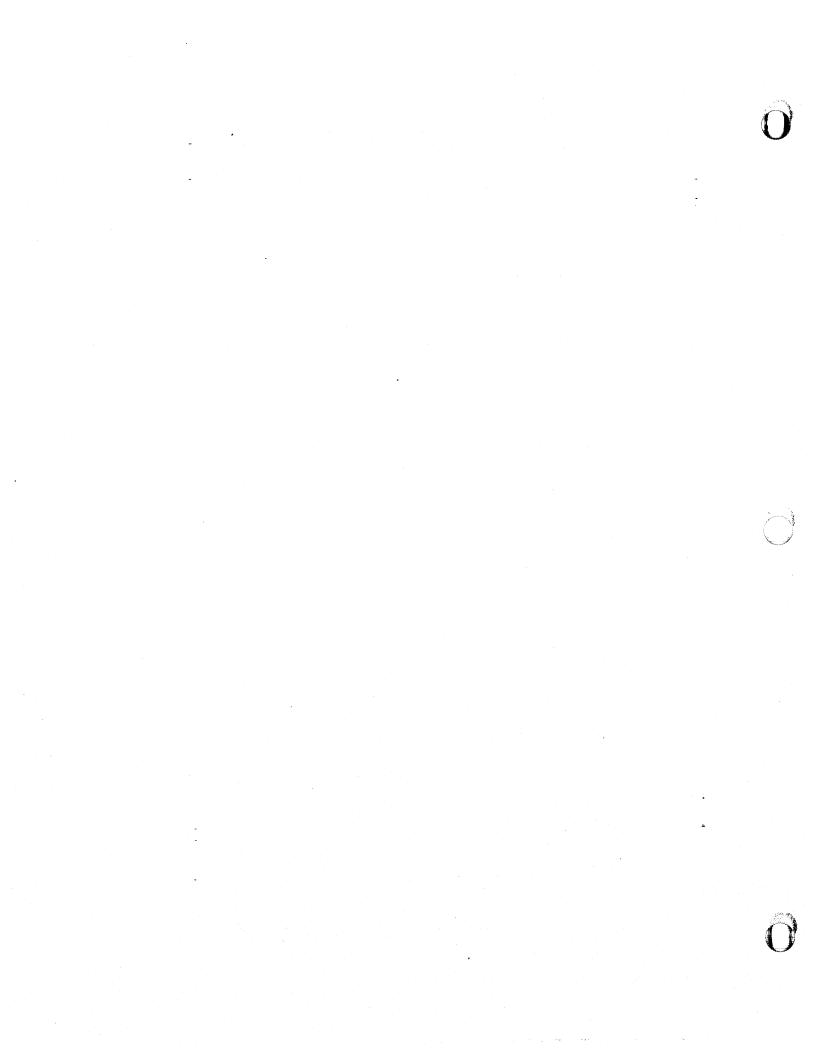
MASS MAP

MASS will generate a micromemory assignment map if the M(AP) option on the OPTN card is specified. The map will show blocks of 256 micromemory words (1 micromemory page) where each micromemory word can hold 2 instructions (an upper and a lower instruction). Asterisks (*) will be printed out for each micromemory location assigned and be put in either the upper or lower position. Remember also that unless a NOSUM option is specified, the checksum of micromemory is placed in the lowest vacant memory location, but would not show on the map.

MASS CROSS-REFERENCE

MASS will generate a cross-reference table if the C(ROSS) option on the OPTN card is specified. The cross-reference listing will contain an alphabetic listing of all the labels that occurred within the program. Each label will have a set of entries describing what its use was, the value associated with it and the page that it appeared on. The label use is shown as follows:

MA-C	MICRO ADDRESS	-	CODE
MA-D	MICRO ADDRESS	-	DATA
MA-O	MICRO ADDRESS	-	ORG
MA-E	MICRO ADDRESS	-	EQUATE
MA-S	MICRO ADDRESS	-	SET
F1-C	FILE 1	-	CODE
F1-D	FILE 1	-	DATA
F1-0	FILE 1	-	ORG
F1-E	FILE 1	-	EQUATE
F1-S	FILE 1	-	SET
F2-C	FILE 2	_ '	CODE
F2-D	FILE 2	-	DATA
F2-0	FILE 2	-	ORG
F2-E	FILE 2		EQUATE
F2-S	FILE 2	-	SET
COND	CONDITIONAL		
UNKN	UNKNOWN		
SPCE	SPACE		
SFLD	S FIELD		
JUMP	JUMP		
K=	K REGISTER =		
N=	N REGISTER =		
CFLD	C FIELD		
BTU	BIT TEST UPPER		
BG	BIT GENERATOR		
CLO	CLEAR LOW ORDER		



SAMPLE LISTING INCLUDING MAP AND CROSS REFERENCE

The following notes are keyed to the fields (columns) of the sample listing:

Note	Contents
A	Assembler identification; host machine type (CYBER 32/6000/CYBER 70L/CYBER 170), assembler version number.
В	Value (in hexadecimal) of the expression on an EQU or an ORG card.
C	Micromemory location (in hexadecimal) assigned to this microinstruction. The HEX MML column contains three digits. The first is the page address; the second two are the micromemory address within the page. The next
	column specifies U for upper-half word or L for lower-half word.
D	The contents in hexadecimal of the location.
E	A number indicating the length of time in cycles required to execute this instruction. (See Basic System Controller Model 65109 Hardware Reference Manual.)
F	Card image. The fields on the card are indicated by the notations: LABEL (location), F, A, B, D, S, C, MT and COMMENTS.
G	If the assembler detects an error in the information coded on the source card, the error code(s) is (are) printed on this part of the listing. There is room to print up to four error codes on the listing.

F-1

7. 05/24/76	DIAGNOSTICS	←					9	·						>	DIAGNOSTICS	
AND CROSS REFERENCE 00.21.37.	COMMENTS		AND CROSS REFERENCE	计多数 计多数 医多种	LEAD IN TO ** AT END. **	计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设计设	INDICATE ONES COMPLEMENT		CHECK SIGN	COMP Q FOR POSITIVE CHECK SIGN OF X	GET POS X AND SIGN IN K CLEAR A, SET TIMES COUNT MAKE FIRST STEP TEST	MULTIPLY ITERATION LOOP MULTIPLY ITERATION LOOP	EXIT ON POS SIGN TEST REM	POS RESULT RECOMP A NEG RESULT, COMP Q NEXT INSTRUCTION	COMMENTS	经存货的 医克勒氏性 医二甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基甲基
	H.		ND CROS	*****	PROVIDE CORRECT	***			SNC	SNU	Stat	רפר	SKZU	77	F F	******
PROGRAM HITH MAP	U		۵.	******	X* PRODUCT TO AQ. ON POSITIVE NUMBERS SO PROVIDE LEAD OF NEGATIVE INPUTS AND CORRECT AT E N FOR ONES COMPLEMENT INPUT NUMBERS.	***			K=0	INCK	DECK N=31 RQR0E	RQR0E RQR0E		EXIT	v	***
PROS	s	-	PROGRAH WITH MAP EAD, HAP, NOSUM, SIHU, L.I	****	PRODUCT TO AQ. POSITIVE NUMB NEGATIVE INPU	***	u u	-				RP1 RP1			v	
u 10	0		PRC HAP. NC	****	ON POSITIVE OF NEGATIVE	****		ŀ	σ	σ	× ⋖	≪ ≪	⋖	< ₫	0	
T LOWAY D	9 8		PLE S,DEAD	***	A BY X. F MORKS ON SIGN OF			<u>د</u>		σ		×		ø	80	20000
•	- 1		EXAMPLE CROSS,D	* * *	HULTIPLY A BY HULTIPLY WORK CALCULATE SIG		30X	START	<	×	×	4 4	⋖	« *	•	1 2
MOTOGON GOO	Vens.		IDENT OPTN	***	HULTIPLY HULTIPLY CALCULAT	****	CMP1 EQU	ORG	<	8 <	-A ZERO	A 00	4-	- A SET SET	u.	****
- 1					* * * * 1		START		MULT	•,	• ,	+ 1		EXIT	LABEL	
4	TICKU ASSEMBLEY CYCLE LABE	COUNT	·				L	ப	•	m s	- mmm	, ww	m	юю	CYCLE	
	1			-			-		0000					2037		-
	MP-50 HFX	IMAGE						<u> </u>	0F03	4 A 2 3	4006				HEX IMAGE	-
	2	E E		-			0 30	B 630	C 030 U	0 31	032 U		0.35.0	34 36	HEX	

Figure F-1. Sample Listing

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF ~ PROGRAM WITH MAP AND CROSS REFERENCE 11.36.12. 77/08/01.PAGE 0 ***** MA-0/0000 / 1 MICRO HEHORY MAP FOR EXAMPLE MA-S/0037 / MA-C/0030 / MA-E/0030 / NAME EXIT HULT START PAGE

~

PROGRAM WITH MAP AND CROSS REFERENCE 11.36.12. 77/08/01.PAGE

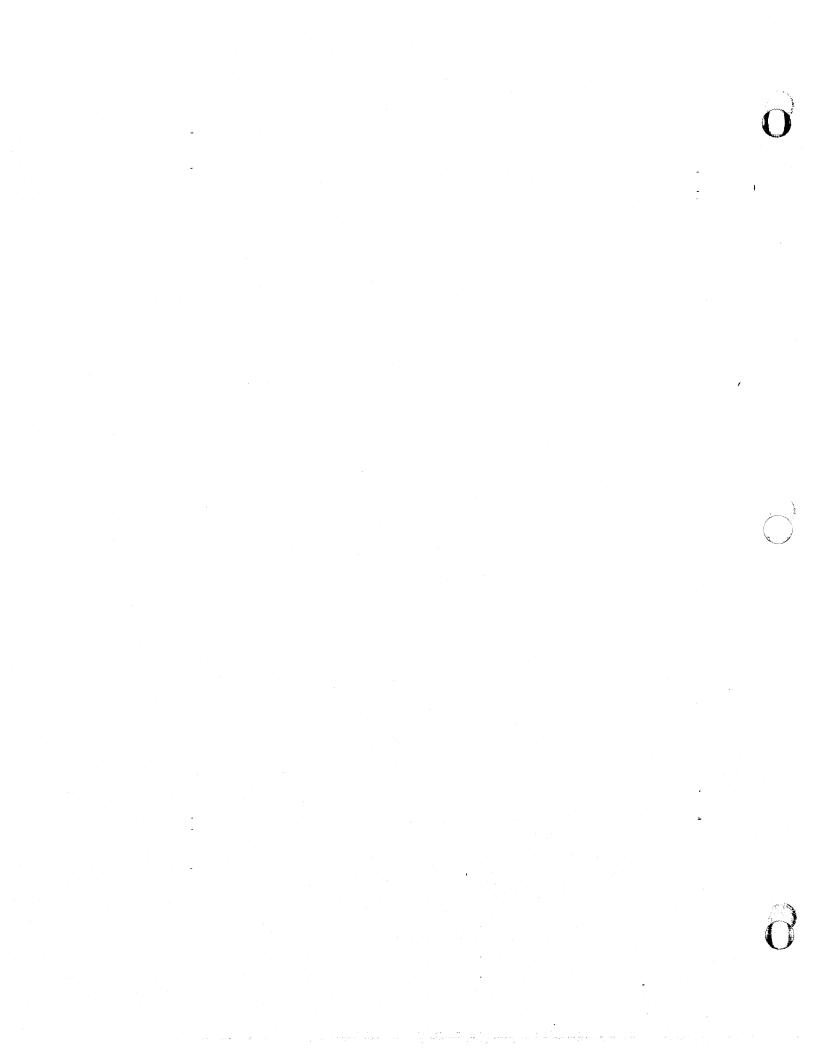
JUMP/0036L/

JUMP/0036 /

GROSS REFERENCE LISTING FOR EXAMPLE

TYPE/VALUE/PAGE

Map And Cross Reference Figure F-2.



ASSEMBLY DIAGNOSTICS

The assembler prints error codes to flag and diagnose incorrect assembly statements. When a statement is in error, one to four error codes are listed to the right of the statement describing the problem. Figure G-1. is an example of a listing containing error codes.

If any errors occur on a page, the last line printed on the page and all subsequent pages of coding contains the message "PREVIOUS ERROR(S) ON PAGE n", where n is the page number of the most recent page to contain errors. The final coding page contains the message "n LINES CONTAIN ERROR(S)" where n is the total number of all lines containing errors, or "NO ERRORS" if no errors were detected.

MASS ASSEMBLY ERRORS

The following error codes are generated by MASS:

Error Code	Meaning
ASF	Mnemonic must not be included in A field be- cause A field is set by F field (shift or scale).
AU	A field mnemonic is undefined.
BSF	Mnemonic must not be included in B field because B field is set by F field (shift or scale).
ви	B field mnemonic is undefined.
CBC	C, B field conflict. B field is N, K, ZERO, or N,K; C field must match B field requirement for bits 28 and 29.

G - 1

Error Code	Meaning
CER	Constant error. Illegal character in HEX, DEC or OCT pseudo instruction.
CMC	C and M field conflict. Command desired in C field cannot be expressed with instruction sequencing mode specified in M field.
CRP	Cannot reach page requested. S field must be left blank so page jump may be encoded in instruction.
CSC	C and S field conflict. The C field requires the S field which has already been set by an A, B, D or S code or a combination.
CU	C field contains undefined symbol.
DDS	Symbol has been defined more than once, or with other than SET pseudo.
DSC	D field requires use of S field, which is already set due to a mnemonic in A or B field.
DU	D field mnemonic is undefined.
ELU	Location field in EQU or SET pseudo instruction is blank.
EXP	Illegal character, symbol or operator in expression, or missing operator.

Error Code	Meaning
FBG	Incorrect number of bits specified in VFD.
FOV	This file location is out of range for the current
	file or was previously defined. If previously
	defined, this instruction does not replace the
	previous one.
FU	F field mnemonic is undefined.
MEM	Micromemory overflow.
MOV	Micromemory overlap. This micromemory location
	was defined previously. This instruction replaces
	the previous one. This error indication appears
	only if binary output has been selected.
MU	M field is undefined.
OPN	Illegal option specified on the OPTN card.
SAB	Both A and B fields require use of S field.
SAS	Mnemonic coded for S field cannot be assembled
	because S field is already set by A, B or D field.
su	S field contains undefined mnemonic or symbol.
TIL	T field code is illegal for jump command or for
	use in K= or N= command.
TNS	T field must be specified for M field mnemonic R.
TU	T field mnemonic is undefined.

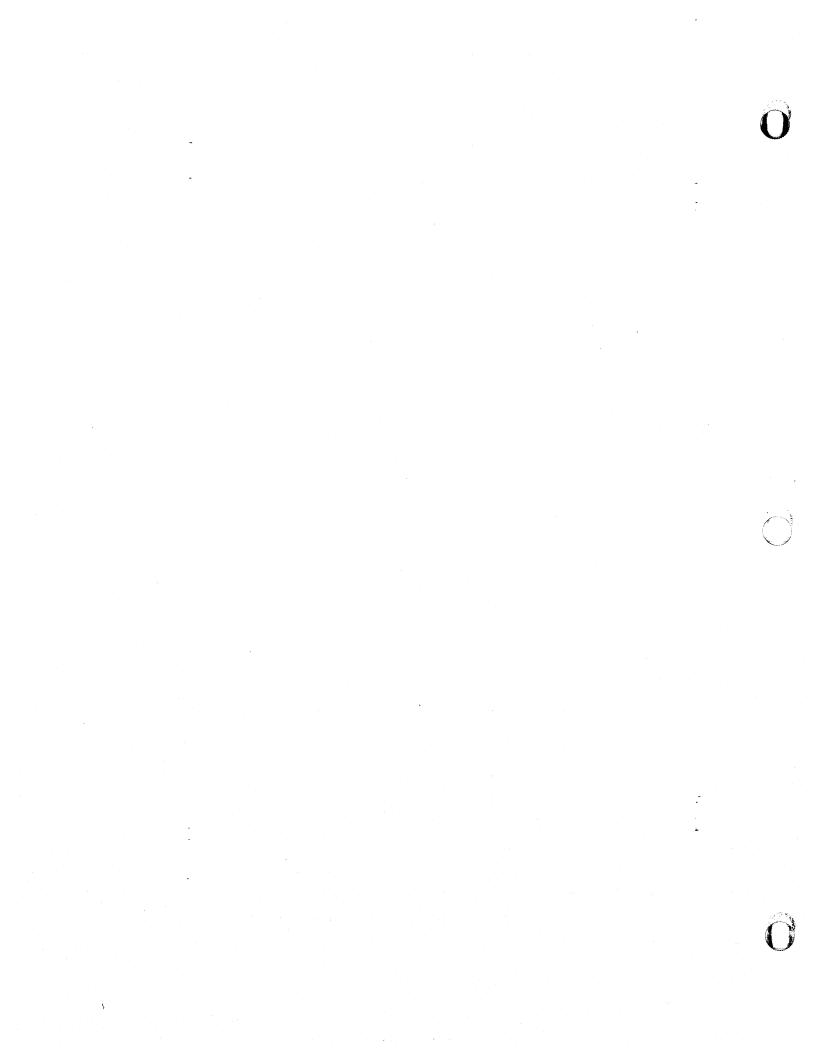
The following messages may appear in the job's dayfile of a CYBER machine:

REQUESTED FL EXCEEDS USER FL
BINARY SUPPRESSED -- INCREASE FL =42000 + 1000 * HIGHEST PAGE

The amount of central memory needed for MASS to produce binary output has exceeded the amount that can be requested by the program. Set the CM parameter on the job card to the value of 42000, plus the result of 10008 times the sum of the highest micromemory page number (of 512 words) plus one.

Thouse Fig. Thouse For the Processes Fig.	HEX	HEX	CYCLE	LABEL			 	· 6 0	0		2	±	сомментя	DIAGNOSTICS
TROUBLE FOUND FAME FROM FAME FAM						JENT G	IAGNO		XAMPL		THE MP		æ	
TROUGLE EQU FAMELE TROUGLE EQU FAMELE TROUGLE EQU FAMELE FAME				:	•	*	***	***	:	:	***	• • • • • • • • • • • • • • • • • • • •	***	
TROUBLE EDU SAN CA AX				•		I AGNO!	STIC E	KAHPLE		•	***************************************	****	***************************************	
U SADO 0000 3	FF.			TROUB	0 111		EAD							EXP
See	¥0.				Ö		×							
1	A0	5800	8 6 6 6	SYM					9				FIRST SYMBOL DUPLICATE	
5 5 5 5 5 5 5 5 5 5		5800	•	SYH							NAS	7	JCATE SYNBOL OF DUPLICATE	S00
1		5800		100		۵				Q O				ij
1		5800				_	٠.	-					A AS A) A 8
1		2980					-		XHX				DESTIN	202
1		5860								2	JOE			23
U 5550 0000 3 A A B C C C C E LABEL F A B D S C ONFELTOR U 5550 0000 3 A A B D C C A D A B D C C C C C C C C C C C C C C C C C C		5690										х 81г	ILLEGAL M CC ILLEGAL T CC	
1	-	3 2			₹								5 1	ASF
U 5500 0700 4 5 SM1 INRD HALT AF S CONFLICT L 5E10 2500 4		7C98 5E30	00		¥			XINTA					MUST BE BLANK CAN*T BOTH BE	8SF SA3
U 5810 0300 3		5400	0.5			٠.		TON		HALT				SAS
L 5E08 2008 3 U 5650 0000 3 ORG 10 U 5650 0000 3 HEX CYCLE LABEL F A B D S C HT CONHENTS I 5600 2007 3 ORG 10 ORG 10 ORG 10 ORG 10 ORG 10 ORG 000		5801	20					T NK C	IOA	GATEI			y N	SAS
L 9830 2530 3 U 0000 300F 3 U 0000 200C 3 U		5508	e (z			DECK N=35		ON A	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
U 565D 0000 3 A+B P X A ORG 0VER EXISTING CODE HEX CYCLE LABEL F A B D S C HT COMMENTS IMAGE COUNT PREVIOUS ERROR(S) ON PAGE 0 ***********************************		9830								RTJ	X 00 1	•	NEED S	CRP.
U 565D 0000 3 A+B P X A DRG OVER EXISTING CODE HEX CYCLE LABEL F A B D S G HT COMMENTS IMAGE COUNT THATE COUNT THE CO		9800	20								N=12		ILLEGAL T FOR	111
U 565D DODO 3 A+B P X A ORG OVER EXISTING CODE END HEX CYCLE LABEL F A B D S G HT COMMENTS IMAGE COUNT IM		9800 2	၁								100	2		
U 565D 0000 3 A+B P X A ORG OVER EXISTING CODE END HEX CYCLE LABEL F A B D S G HT COMMENTS IMAGE COUNT THAGE COUNT THAGE COUNT THAGE ON PAGE 0 ***********************************	0 A				ō		0 1							
END HEX CYCLE LABEL F A B D S G HI COMHENTS IMAGE COUNT	O.A.	2650	00					×	⋖				ORG OVER EXISTING CODE	NOW
HEX CYCLE LABEL F A B D S G HT COMMENTS IMAGE COUNT	8													
HEX CYCLE LABEL F A B D S G HT COMMENTS IMAGE COUNT THAGE T					ŭ	C.								
Secretarian PREVIOUS ERROR(S) ON PAGE 0 secretarian se	Ä H	HEX	CYCL				æ	80	0	S	ပ	Ŧ	СОННЕИТЅ	DIAGNOSTICS
		***	•			æ	EVIOUS	ERRO	1(2)	ON PAGE		\$ 1 \$ 1 \$ 1		

Figure G-1. Assembler Diagnostics Example



The basic CPU microinstruction execution time is 168 nanoseconds. Some microinstructions have longer execution times to allow certain operations to be completed. The microinstructions have been grouped according to execution times as requiring 3, 4, 5, 6, 5 + n and 9 56-nanosecond cycles.

The classification of microinstructions is shown in Figure H-1. This figure provides execution time by types for all legal combinations of microcommands which extend the basic cycle time.

Exceptions to the execution time, as listed in Figure H-1, are:

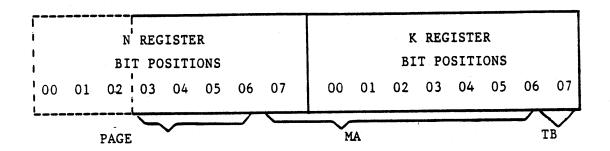
- The combination of one's complement arithmetic with ADD+ or ADD+T arithmetic operation requires four 56-nanosecond cycles.
- The MMU command is included under read/write micromemory operand commands; therefore, it requires nine, rather than four, cycles.

Analysis of a microprogram for execution time starts by classifying each of the microinstructions. This is done by using the microinstruction classification table or by examining the assembler output listing.

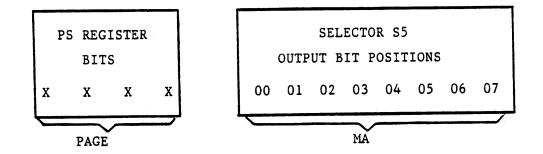
Two addressing modes are available for operand references via status mode bit 113 (SM113) as follows:

• SM113 = 0 - The contents of the combined N/K registers are used to reference operands as indicated. The least significant bit of K (bit 07) determines which 32-bit half-word is referenced via T' code 010.

H-1



• SM113 = 1 - An MA transform via S5 determines the 64-bit word. The 32-bit half-word to be referenced must be selected by a T field test.



XXXX = Page in which referencing microinstruction resides.

```
READ/WRITE MICROMEMORY OPERANDS
      SHIFT OR SCALE
         TMA, TMAK, GITMAK, ZL, NU, COL,
           ADD OR SUBTRACT OR
             A', B', NU, ZL, COL,
                 ONE'S COMPLEMENT
                  INSTRUCTION CYCLES
   N
     N
        N
           N
              N
                  3
N
N
   N
      N
        N
           N
              Y
                  3
           Y
   N
      N
              N
           Y
N
   N
      N
         N
              Y
      N
           N
   N
              N
                  5
           N
              Y
         Y
   N
      N
              N
           Y
         Y
              Y
                  6
   N
      N
   N
      Y
         N
           N
              N
                  5
N
N
   N
      Y
         N
           N Y
                  5
      Y
                  5
           Y
N
   N
         N
              N
                  5
N
      Y
           YY
   N
        N
                  5
N
   N
      Y
        Y
           N
              N
      Y
        Y
                  5
N
   N
           N
              Y
           Y
   N
     Y
        Y
                  6
N
              N
     Y
        Y
           Y
                  6
N
   N
              Y
   Y
      X
        X
           X
              X
                 5 + n where n = number of shifts
N
      X
         X
            Х
                    N = No
                    Y = Yes
                    X = Don't Care
```

Figure H-1. Microinstruction Classification



GLOSSARY

<u>Item</u>	<u>Definition</u>
A	A register. General purpose register that may be shifted individually or in conjunction with the Q register.
A field	Field of MASS input form for specifying input to selector 1.
ALU	Arithmetic logic unit. Capable of performing addition and subtraction, as well as logical operations on output of selector 1 and selector 2.
A/Q	A register, Q register or the combined A/Q register. The A and Q registers are shift registers.
A source	Register or data item selected as output of selector 1.
B field	Field of MASS input form for specifying input to selector 2.
BG	Bit generator. Specialized circuitry that generates a value consisting of single bit set to 1 with rest of bits set to 0.
B source	Register or data item selected as output of selector 2.
Bit test	Output of selector 7 (located on transform board), which allows any specified condition or bits to be used for selection of next upper or lower microinstruction.
C field	Field of MASS input form for specifying instructions, constants or jump address.
Decr	Decrement to reduce contents of a register by one. May affect N or K register.

Item	<u>Definition</u>
F	F register. General purpose, word length register. Note that the contents of F will be destroyed when File 1, 2 is accessed.
F'	Holds data for File 3 write.
F field	Field of MASS input form for specifying ALU operation or shift or scale operation.
F1	File 1. 256-word register file, which is addressed by contents of K register.
F2	File 2. 32-word register file, which is addressed by contents of N register.
F3	File 3. High-speed micromemory which is addressed by the K' or J register.
I	I register. General purpose register; is loaded with output of selector 1.
Incr	Increment to increase contents of a register by one. May affect N or K register.
INP	Input. One of possible paths of transferring information from external devices to selector 4 and thus to selector 2 input to ALU.
Input bus	Means of transferring information to selector 1 or selector 2.
Interrupt	A condition that is tested by the microinstruction by use of interrupt system, consisting of mask registers and interrupt
	address generator.
IXT	Temporary holding register used to decode macroinstructions.

<u>Item</u>	<u>Definition</u>
J	J register. Used to address File 3.
K	K register. An 8-bit counter that may be set, incremented, decremented and tested for zero. Also used to address file 1.
Κ'.	Contains image of the K register in the lower 8 bits and is also used to address File 3.
L	Holding register for constant increment for J.
Ll	Shift input to a selector; specifies left shift one bit position to generate output.
L8	Shift input to a selector; specifies left shift eight bit positions to generate output.
LAi	Main memory lookahead register; i = 0 if DMA lookahead, 1 to 3 if CPU lookahead.
LHW	Shift input to a selector; specifies left shift a half word, end around, to generate output.
M field	Field of MASS input form for specifying mode of obtaining next microinstruction pair.
Ml	Mask register. Word-sized register to control recognition of interrupts. An interrupt is recognized if its corresponding mask bit of Ml or M2 is set to 1.
м2	Mask register. Word-sized register to control recognition of interrupts. An interrupt is recognized if its corresponding mask bit of Ml or M2 is set to 1.
MA	Micromemory address used to address micromemory.

17328900-02 GLOSSARY-3

<u>Item</u>	Definition
MAC	Micromemory address counter. Counter that is always updated to contain address of next sequential microinstruction pair.
MIR	Microinstruction register. Register that holds current microinstruction being executed.
мм	Micromemory.
N	N register. An 8-bit counter that may be set, incremented, decremented and tested for zero. Used as shift and scale counter, used to address file 2 and used to control repeat operation.
One's Complement	The radix-minus-one complement in binary notation.
P	P register. General purpose register named P (not a pro- gram counter).
P-MA	Page-micromemory address register. 12-bit register that contains 4-bit page and 8-bit micromemory address of current microinstructions.
Page	Unit of micromemory containing 256 locations of 512 microsstructions.
PS	Page number saved of the previous microinstruction
	executed
PTR	Paper tape reader.

Q register. General purpose register that may also be used

in conjunction with A register in shifting.

Q

)	Item	<u>Definition</u>
	R1	Shift input to a selector; specifies right shift one bit position to generate output.
	RDi	Read from main memory holding register; i = register number from 1 to 3.
	RTJ	Return register. 12-bit register used to hold a micromemory address for returning from micro-subroutine.
	S field	Field of the MASS input form for specifying special operation taking place in parallel with other portions of microin-struction.
	S1	Selector 1. Selects A input to ALU.
	S2	Selector 2. Selects B input to ALU.
	s3	Selector 3. Provides shifting of ALU output going to P, A, F, X, Q, Fl or F2 destinations.
	S4	Selector 4. Selects one of four external inputs to selector 6.
	S 5	Selector 5. Used in performing MA transforms.
	S6	Selector 6. Used in selecting source of next microinstruction.
	S7	Selector 7. Selects one of the 16 possible bit tests.
	S8	Selector 8. Used to select values in K and N transform.
	S 9	Selector 9. Selects status/mode or mask register for input to selector 1.

Item	<u>Definition</u>
SM1	Status/mode l register. Status and mode control.
SM2	Status/mode 2 register. Status and mode control.
SRj	State register; j = DMA state registers 0 to 3 and CPU state register.
SW	Console switches.
T field	Field of MASS input form for specifying selection criteria for next microinstruction.
TB	Test bit used to identify upper or lower of micro-instruction pair.
Transform	Facility provided by selector 5 to generate micromemory addresses and selector 8 to load K and N registers.
Two's Complement	The radix complement in binary notation.
WDi	Write-to-main-memory holding register; i = register number from 1 to 3.
X	X register. General purpose word length register.

COMMENT SHEET

TITLE: MASS Reference Manual

PUBLICATION NUMBER: 17328900

REVISION: 03

NAME:

COMPANY:

STREET ADDRESS:

CITY:

STATE:

ZIP CODE:

Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 8241

MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Systems Technology Division 215 Moffett Park Drive Sunnyvale, California 94086



FOLD

FOLD