ASSEMBLY LANGUAGE

The Programmer's Companion

PRIME

The Programmer's Companion

The Programmer's Companion is a new series of pock
size, quick-reference guides to Prime software produc

Published by Prime Computer, Incorporated
145 Pennsylvania Avenue Framingham, MA 01701

Produced by Prime Technical Publications Departmer
3 Newton Executive Park Newton, MA 02162

First Printing, August 1978

TABLE OF CONTENTS

Prime Macro Assembler 2
MA Pseudo-Operations 2
MA Error Diagnostics 9
nstruction Summary Charts 12
)ata Structures 39
rocessor Characteristics 47
nstruction Formats (S,R,V Modes) 56
nstruction Formats (I-Mode) 60
.ddressing Mode Summaries and
low Charts (S,R,V Modes) 62
lotes 87

PRIME MACRO ASSEMBLER

PMA treename [-option-1] [-option-2] . . . [option-n]

Invokes the PMA program For more information refer to tl PMA Programmer's Guide

Option	Meaning
-INPUT treename	Input treename
-LISTING treename	Listing treename
-BINARY treename	Object file treename
-EXPLIST	Generates full assembly listi (overrides the pseudo-operatic NLST), and forces listing fi generation
-ERRLIST	Generates errors-only listing ar forces listing file generation
-XREFS	Omits from the listing symbo which have been defined but n used

PMA PSEUDO-OPERATIONS

How to use this listing

Type of pseudo op
AD =address definition
AC =assembly control
CA =conditional assembly
DD =data defining
LC =listing control
LI =literal control
LO =loader control
MD=macro definition
PL =program linking
SA =storage allocation
SD =symbol definition

Pseudo op format

[label] END address-expression **A**

Terminates assembly of source program

Explanation of pseudo op along with information on par meters, etc

ABS **A**

Sets assembly and loading mode to absolute *Not in SEG or SEG mode*

[label] AP address-expression [modifier] **AD**

Generates an argument pointer in the form used by the 64V/32I Procedure Call (PCL) instruction Argument is an argument variable written in memory reference format SEG/SEGR mode

modifier

S	Set argument store bit
SL	Set argument store bit Last argument
*S	Set argument store bit Argument is indirect
*SL	Set argument store bit Argument is indirect and last
*	Intermediate indirect argument
	Intermediate argument

label-1] { **BACK** / **BACK TO** } **label-2** **CA**

Directs assembler to start assembly from label-2 Label 2 must precede this statement in the source text Macros only

[label] BCI { **string** / **# string** } **DD**

Defines ASCII character strings by packing with specified ASCII characters two per word starting with the most significant 8 bits

label] { **BSS** / **BES** / **BSZ** } **expression** **SA**

Allocates a block of words of the size specified in expression starting at the current location count If there is a label it is assigned to either the first word of the block (BSS and BSZ) or to the last word of the block (BES) For BSZ all words within the block are set to zeros

C64R **AC**

Directs assembler to flag any instructions and/or memory reference not compatable with 64R addressing mode

[label] CALL symbol **PL**

In non-64V or 32I modes CALL generates a JST to **symbol**, which is defined by the assembler as external In 64V/32I mode CALL generates a PCL instruction

ENT symbol **PL**

Provides a conditional ENT capability

[label] COMM symbol [(size)], **SA**

Defines FORTRAN-compatible COMMON areas

D16S **LC**

Directs assembler and loader to use 16S mode

D32R **LC**

Directs assembler and loader to use 32R mode

D32S **LC**

Directs assembler and loader to use 32S mode

D64R **LC**

Directs assembler and loader to use 64R mode

D64V **LC**

Directs assembler and loader to use 64V mode

D32I **LC**

Directs assembler and loader to use 32I mode

[label] DAC address-expression **AI**

Generates a 16 bit pointer in S/R mode

[label] DATA [(absolute-expression-1)] absolute expression-2,... **DI**

Define **expression-2 expression-1** times Expression-1 is assume to be 1 if omitted

DDM **LC**

Directs the assembler and loader to use the default addressin mode

[label] DEC decimal-integer-constant,... **DI**

Defines **decimal integers.**

DUII absolute-expression-1, absolute-expression-2 **LC**

Triggers the loading of the UII package **Expression-1** is a bit mas defining instruction sets the UII package emulates and, **expression-2** is a bit mask defining the hardware sets that must b present to execute the UII package

bit number	
1-9	*Must be* 0
10	Prime 500
11	Prime 400
12	Undefined
13	Double-precision floating point
14	Single-precision floating point
15	Prime 300 only
16	High speed arithmetic

DYNM [absolute-expression-1] [svmbol [(absolute expression-2)] [absolute-expression-3]] **SD**

Declare stack relative symbols **Expression 1** is the stack header size **symbol** is the name **expression-2** is the number of words to allocate and **expression-3** is the stack offset

[label] ECB entry-point [link-base] displacement, n-arguments [stack-size] [kevs] **PL**

Generates an entry control block to define a procedure entry SEG/SEGR mode

EJCT **LC**

Causes the listing device to eject the page

ELSE **CA**

Reverses the condition set up by an IFxx statement until the matching ENDC statement is reached

END **AC**

Terminates assembly of the source program

ENDC **CA**

Defines the end of a conditional assembly area set up by an IFxx statement

ENDM **MD**

Terminates the assembly of a macro definition

[label] ENT svmbol-1 [symbol-2] **PL**

Links subroutine entry points to external names used in CALL XAC or EXT statements in calling programs **Svmbol 1** is external name **symbol-2** is internal name If symbol 2 is missing the internal name is assumed to be the same as the external name See also SUBR pseudo op

symbol { ***EQU expression [symbol expression]*** / **EQU svmbol expression** } **SD**

Defines **symbols** whose value may not be changed during assembly (see SET)

[label] EXT symbol **SD**

Identifies external **symbol**

FAIL **CA**

Generates an F error

[label] FIN **LI**

Controls the placement of literal pools

{GO / GO TO} **label** **CA**

Causes suspension of assembly of all subsequent statements until **label** is found *Macros only*

[label] HEX hexadecimal-constant **DD**

Defines **hexadecimal constant**

[label] IF absolute-expression statement **CA**

Provides ability to selectivity assemble code on the basis of a test

[label] {**IFM (minus)** / **IFP (plus)** / **IFZ (zero)** / **IFN (non-zero)**} **expression** **CA**

Sets specific text to control code assembly Continue assembly (to ENDS or ELSE) if **expression** test is true

[label] IP address-expression **AD**

Generates a 32 bit indirect point SEG/SEGR mode

LINK **AC**

Places subsequent code in the linkage frame *SEG/SEGR mode only*

LIR absolute-expression

Controls library program loading **Expression** is a bit mask defining instruction groups that are to trigger loading See DUII pseudo op for bit assignments

LIST **LC**

Causes all statements to be listed except those generated by a macro expansion

LSMD **LC**

Lists macro calls plus any data generated by macros

LSTM **LC**

Lists macro call statements plus all lines generated by the macro expansion including code and data values

label MAC noise words **MD**

Begins the definition of the macro named by the label

N64R **LO**

Informs the loader that the program is *not* to be loaded in 64R mode

NLSM **LC**

Lists only the macro call *does not* list statements generated by macro expansion Ignored if -EXPLIST Command line option is specified

NLST **LC**

Inhibits listing of all subsequent statements until a LIST is encountered Ignored if -EXPLIST command line option is specified

[label] OCT octal-constant, ***DD***

Defines **octal integers**

[label] ORG address-expression **AC**

Sets the assembler location count equal to the value of **address-expression**

PCVH **LC**

Directs the assembler to print symbol values in the cross reference in hexadecimal instead of octal

PROC **AC**

Places subsequent code in procedure segment SEG/SEGR mode

REL **AC**

Sets assembly and loading mode to relocatable *Not in SEG/SLGR mode*

RLIT **LI**

Directs the assembler to optimize literal allocation for relative addressing modes (32R, 64R 64V and 32I modes)

[label] SAY ASCII-text-string **MD**

Cause the text string to be printed in the listing

SCT absolute-expression **MD**

Assembles selected code groups based on expression

value	assembly condition
0	Assemble from the SCT statement to the first % marker then skip to the %/ line
1	Skip to the first % marker, assemble from there to the second % marker then skip to %/ marker
n	Skip to the nth % marker if any, assemble from there to marker n+1 then skip to %/ marker If there is no nth% marker proceed as for -n
-n	Skip to the %2 marker, if any, and assemble from there to the next % marker then skip to the %/ line If there is no %2 marker skip to %/ line

[label] SCTL expression 1, expression 2 **MD**

Assembles selected code groups based on comparison between **expression-1** and the rest of the list. See SCT for expression values.

SDM absolute-expression **LO**

Directs the loader to set its default addressing mode to **expression**.

expression	
0	16S mode
1	32S mode
2	64R mode
3	32R mode

SEG **AC**

Directs the assembler to create a 64V segmented mode assembly module.

SEGR

Directs the assembler to create a 32I assembly module.

[label] SETB start-address size **LO**

Specifies the **start-address** and the **size** of a base area for out of range indirect address links.

[label] SUBR symbol **PL**

Links entry points to external names used in CALL, XAC or EXT. Synonomous with ENT pseudo op.

[symbol] SET { **expression [symbol expression]** / **symbol expression** } **SD**

Defines **symbols** whose values may be redefined during assembly.

[label] VFD absolute expression-1 absolute-expression-2, **DD**

Permits 16 bit words to be formed in subfields of varying length by pairs of constants. The first expression gives the subfield size, the second gives the value.

[label] XAC symbol **AD**

Generates a 16 bit pointer to the external **symbol**.

[symbol] XSLT { **expression [symbol expression]** / **symbol expression** } **SD**

Same as SET, but symbols are not listed in the cross reference listing.

ERROR DIAGNOSTICS

ERR	DESCRIPTION
C	INST IMPROPERLY TERMINATED
F	BAD TERMINATOR ON ARGUMENT # EXPRESSION (MACRO CALL) ILLEGAL OPERATOR ON STACK PUSH POP FAIL PSEUDO OP
G	GOTO ERROR WITHIN MACRO END/ENDM PSEUDO OP WITHIN GOTO SKIP AREA
I	GENERIC I/O OR SHIFT HAS TAG MODIFIER TAG MODIFIER FIELD NOT PERMITTED ON 32I MODE FIELD INSTR SHORT INSTRUCTION SPECIFIER (#) CAN'T MAKE SHORT 64V MODE LDX CLASS INSTR BAD TAG MODIFIER FIELD 64V MODE TAG MODIFIER NOT PERMITTED ON BRANCH INSTR SEG MODE COMMON OR EXTERNAL REF BAD INDIRECT OR INDEX AP/IP, INDEX SPECIFIER INVALID TAG MODIFIER NOT PERMITTED ON 32I BRANCH
L	IMPROPER LABEL (CONSTANT/TERMINATOR IN LABEL FIELD) EXTERNAL VARIABLE PRESENT IN LITERAL BAD ARGUMENT IN EQU SET OR XSET
M	LABEL DEFINED MORE THAN ONCE
N	END WITHIN MACRO OR IF
O	UNRECOGNIZED OPCODE OR I ONLY OPCODE IN NON 32I MODE 64V MODE MEMORY REF NOT ON 64V MODE S/R MODE MEMORY REFM NOT IN S/R MODE
P	MISMATCHED PARENTHESIS
Q	AP NOT IN 64V/32I MODE IP NOT IN 64V/32I MODE ENDM PSEUDO OP NOT IN MACRO
R	STACK OVERFLOW MULTIPLY DEFINED MACRO OR MACRO NAME FIELD EMPTY
S	LOAD MODE INSTRUCTION WOULD REQUIRE DESECTORIZATION INDIRECT DAC IN C64R MODE

T 32I MODE TAG MODIFIER SYNIAX FRROR

U UNDEFINED VARIABLE IN ADDRESS FIFLD / EXPRFSSION
UNDFIINFD VARIABLE IN ORG/SFIB

V BIT FIELD IN BIT INST OUR OF RANGF
UNRECOGNIZED OPERATOR IN EXPRESSION
FIELD ADDRESS INST FAR OUT OF RANGE
I/O INST FUNCTION CODE / DLVICF ADDR OUT OF RANGE
SHIFI INSI SHIFT COUNT OUT OF RANGE
FIELD ADDRESS INSI NO COMMA FOLLOW ING FAR SPEC
32I MODE RFGISTER GENERIC NO COMMA AFIER REGISTER #
32I MODE FPR REGISTER GENFRIC NO COMMA AFTER REGISTER #
32I MODE BIT TEST INSTR NO COMMA AFTER REGISTER #
32I MODE BIT TEST INSTR NO COMMA AFIER BIT #
32I MODE GFN REGISIER MEMORY RFF BAD DEIIMITER
32I MODE SHIFT INSTR BAD DEIIMITFR
BAD SHIFI COUNT IN 32I MODE SHIFI INSTR
BAD IAG MODIFIER IN 32I MODE SHIFT
BAD DELIMITER AFTER REGISTER # IN 32I MODI PIO INSTR
OPEN PARENTHESIS MESSING ON DFIB ARGUMENT
CLOSE PARENIHESIS MESSING OF DFTB ARGUMFNT
label MISSING ON IFTF IFIT IFVT IFVF
NAME NOT FOUND IN IFTF IFTT IFVI IFVF
ABS/REI ILLEGAL IN SEG MODE
SEG/SEGR AFTFR CODE HAS BEFN GEN FRAIED
PROC/LINK FOUND OUTSIDF OF SEG MODE
FIELD OUI OF RANGE ON DDM PSEUDO OP
BAD ARGUMENT FOLLOWING EXT
END WITHIN MACRO
SYNTAX ERROR IN DYNM PSEUDO OP
BAD ARGUMENT ON SUBROUIINE (SUBR) STAIEMENT
VFD PSEUDO OP 16 BITS NOI DEFINED
UNTERMINATED CHARACIER STRING
EXPRESSION OVERFIOW ON FLOAIING PT NORMALIZE
EXPRESSION OVERFLOW ON FLOATING PI RE NORMALIZE

SCALED BINARY LOSS OF SIGNIFICANCE
FLOATING POINT NUMBER OUT OF RANGE
BCI REPEAT COUNT ERROR
BCI COUNT VARIABLE TYPE ERROR
CALL CONTAINS CONSTANT OR TERMINATOR IN ADDR FIELD
COMMON (COMN) PSEUDO OP HAS BAD ADDRESS FIELD
DEC DATA DBP HEX OR OCT REPEAT COUNT ERROR
DEC/OCT PSUEDO OP HAS BAD OPERATOR
RLIT FOUND AFTER CODE HAS BEEN GENERATED
NO LABEL ON DFTB

X 32I MODE GENERAL REGISTER SPECIFICATION ERROR

Y PHASE ERROR

Z ILLEGAL ABSOLUTE REFERENCE IN SFI MODE
SEG MODE ABSOLUTE REF NOT PERMITTED UNLESS 0 7
AP/IP ABSOLUTE REF INVALID
MORE THAN 1 EXTERNAL NAME IN AN EXPRESSION
INCORRECT EXPRESSION MODE FOR GIVEN INSTRUCTION
EXPRESSION MODE ERROR
>1 OPERATOR NON ABS/REI OR RIGHT-HAND OP NOT ABS/REI
EXTERNAL NAME NOT PERMITTED

INSTRUCTION SUMMARY

This chart contains a complete list of instructions for the Prime 100 through 500 Each instruction is followed by its octal code, format function information on addressing mode and hardware availability and a one line description of the instruction

The columns in the list are as follows

R — RESTRICTIONS

blank — regular instruction

R — instruction causes a restricted mode fault if executed in other than ring 0

P — instruction may cause a fault depending on address

W — writable control store instruction may be programmed in wcs to cause a fault

M — Machine specific — use only on specified CPU Usually an instruction reserved for operating system, such as EPMJ

MNEM — a mnemonic name recognized by the assembler PMA

OPCODE — Octal operation code of the instruction The codes are indented so that I/O instructions are isolated from generics, and the memory reference and register instructions of the P500 are sorted apart from the MR instructions of the P100-400

RI — Register (R) and Immediate (I) forms available (P500 memory reference instructions only), Y = YES N = NO

FORM — Format of instruction

MNEMONIC	DEFINITION
GEN	Generic
AP	Address Pointer
BRAN	Branch
IBRN	I-mode Branch
CHAR	Character
DECI	Decimal
PIO	Programmed I/O
SHFI	Shift
MR	Memory Reference — non I-mode
MRFR	Memory Reference — Floating Register
MRNR	Memory Reference — Non Register
RGEN	Register Generic

UNC — Function of instruction

MNEMONIC	DEFINITION
ADMOD	Addressing Mode
BRAN	Branch
CHAR	Character
CLEAR	Clear field
DECI	Decimal Arithmetic
FIELD	Field Register
FLOAT	Floating Point Arithmetic
INT	Integer
INTGY	Integrity
IO	Input Output
KEYS	Keys
LOGIC	Logical Operations
LTSTS	Logical Test and Set
MCTL	Machine Control
MOVE	Move
PCTLJ	Program Control and Jump
PRCEX	Process Exchange
QUEUE	Queue Control
SHIFT	Register shift
SKIP	Skip

MODE Addressing modes in which instruction functions as defined

- **S** Sectored
- **R** Relative
- **V** 64V (P400 P500)
- **I** 32I (P500)

2 3 4 5 How instruction is implemented on each CPU (100 thru 500) on each CPU Codes are

- — Not implemented *Do not* use this mnemonic on this CPU
- H Implemented by standard hardware
- O Implemented by hardware option *or* UII library if option is not present
- U Implemented by UII library

— How instruction affects C and L bits codes are

- — C and L are unchanged
- **1** C = unchanged L = carry
- **2** C = overflow status L = carry
- **3** C = overflow status L = unspecified
- **4** C = status extention L = unspecified
- **5** C = result L = unspecified
- **6** C = *unspecified* L = *unspecified*
- **7** C = loaded by instruction L = loaded by instruction
- **8** C = cleared by instruction L = unspecified

CC — How instruction affects condition codes codes are

- — condition codes are not altered
- 1 condition codes are set to reflect the result of arithme tic operation or compare
- 4 condition codes are set to reflect result of branch, con pare or logicize operand state
- 5 condition codes are indeterminant
- 6 condition codes are loaded by instruction
- 7 special results are shown in condition codes for th instruction

DESCRIPTION — a brief description of the instruction

A	02	YY	MRGR	INT	I	—	—	—	—	H	2	1	Add Fullword
A1A	141206		GEN	INT	**SRV**	H	H	H	H	H	2	1	**Add One to A**
A2A	140304		GEN	INT	SRV	H	H	H	H	H	2	1	Add Two to A
ABQ	141716		AP	QUEUE	**V**	—	—	—	H	H	—	6	**Add to Bottom of Queue**
ABQ	131		RGEN	QUEUE	I	—	—	—	—	H	—	7	Add to Bottom of Queue
ACA	141216		GEN	INT	**SRV**	H	H	H	H	H	2	1	**Add C-Bit to A**
ADD	06		MR	INT	SRV	H	H	H	H	H	2	1	Add
ADL	06 03		MR	INT	**V**	—	—	—	H	H	2	1	**Add Long**
ADLL	141000		GEN	INT	V	—	—	—	H	H	2	1	Add Link Bit to L
ADLR	014		RGEN	INT	**I**	—	—	—	—	H	—	7	**Add Link to R**
AH	12	**YY**	MRGR	INT	I	—	—	—	—	H	2	1	Add Halfword
ALFA 0	001301		GEN	FIELD	**V**	—	—	—	H	H	6	5	**Add Long Integer to Field Address**
ALFA 1	001311		GEN	FIELD	V	—	—	—	H	H	6	5	Add Long Integer to Field Address
ALL	0414XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	**A Left Logical**
ALR	0416XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	A Left Rotate
ALS	0415XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	**A Left Shift**
ANA	03		MR	LOGIC	SRV	H	H	H	H	H	—	—	AND
ANL	03 03		MR	LOGIC	**V**	—	—	—	H	H	—	—	**AND Long**
ARFA	161171		RGEN	FIELD	I	—	—	—	—	H	—	7	Update Field Address **Register**
ARGT	000605		GEN	PCTLJ	**VI**	—	—	—	H	H	—	—	**Argument Transfer**
ARL	0404XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	A Right Logical

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	ARR	0406XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	A Right Rotate
	ARS	0405XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	A Right Shift
	ATQ	141717		AP	QUEUE	V	—	—	—	H	H	—	6	Add to Top of Queue
	ATQ	135		RGEN	QUEUE	I	—	—	—	—	H	—	7	Add to Top of Queue
	BCEQ	141602		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC = 0
	BCGE	141605		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC ≥ 0
	BCGT	141601		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC > 0
	BCLE	141600		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC ≤ 0
	BCLT	141604		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC < 0
	BCNE	141603		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if CC •NE• 0
	BCR	141705		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if C-Bit = 0
	BCS	141704		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if C-Bit = 1
	BDX	140734		BRAN	BRAN	V	—	—	—	H	H	—	—	Decrement X; Branch if X = 0
	BDY	140724		BRAN	BRAN	V	—	—	—	H	H	—	—	Decrement Y; Branch if Y = 0
	BEQ	140612		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if A = 0
	BFEQ	141612		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if F = 0
	BFEQ	122		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F = 0
	BFGE	141615		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if F ≥ 0
	BFGE	125		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F ≥ 0
	BFGT	141611		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if F > 0
	BFGT	121		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F > 0

BFLE	141610	BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if F ≤ 0**
BFLE	120	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F ≤ 0
BFLT	141614	BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if F < 0**
BFLT	124	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F 0
BFNE	141613	BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if F •NE• 0**
BFNE	123	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if F •NE• 0
BGE	140615	BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if A ≥ 0**
BGT	140611	BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if A 0
BHD1	144	IBRN	BRAN	I	—	—	—	—	H	—	—	**Decrement H by One; Branch if H = 0**
BHD2	145	IBRN	BRAN	I	—	—	—	—	H	—	—	Decrement H by Two, Branch if H 0
BHD4	146	IBRN	BRAN	I	—	—	—	—	H	—	—	**Decrement H by Four; Branch if H = 0**
BHEQ	105	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if H 0
BHEQ	112	IBRN	BRAN	I	—	—	—	—	H	—	4	**Branch if H = 0**
BHGT	111	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if H 0
BHI1	140	IBRN	BRAN	I	—	—	—	—	H	—	—	**Increment H by One; Branch if H ≠ 0**
BHI2	141	IBRN	BRAN	I	—	—	—	—	H	—	—	Increment H by Two Branch if H = 0
BHI4	142	IBRN	BRAN	I	—	—	—	—	H	—	—	**Increment H by One; Branch if H = 0**
BHLE	110	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if H 0
BHLT	104	IBRN	BRAN	I	—	—	—	—	H	—	4	**Branch if H < 0**
BHNE	113	IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if H is not equal to 0
BIX	141334	BRAN	BRAN	V	—	—	—	H	H	—	—	**Increment X and Branch**
BIY	141324	BRAN	BRAN	V	—	—	—	H	H	—	—	Increment Y and Branch

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	BLE	140610		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if A ⩽ 0
	BLEQ	140702		BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if L = 0**
	BLGE	140615		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch is L ⩾ 0
	BLGT	140701		BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if L > 0**
	BLLE	140700		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if L ⩽ 0
	BLLT	140614		BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if L < 0**
	BLNE	140703		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if L •NE• 0
	BLR	141707		BRAN	BRAN	V	—	—	—	H	H	—	—	**Branch if L-Bit = 0**
	BLS	141706		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if L-Bit 1 (Set)
	BLT	140614		BRAN	BRAN	V	—	—	—	H	H	—	4	**Branch if A < 0**
	BMEQ	141602		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if Magnitude 0
	BMGE	141706		BRAN	BRAN	V	—	—	—	H	H	—	—	**Branch if Magnitude is ⩾ 0**
	BMGT	141710		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if Magnitude is 0
	BMLE	141711		BRAN	BRAN	V	—	—	—	H	H	—	—	**Branch if Magnitude is ⩽ 0**
	BMLT	141707		BRAN	BRAN	V	—	—	—	H	H	—	—	Branch if Magnitude is < 0
	BMNE	141603		BRAN	BRAN	V	—	—	—	H	H	—	—	**Branch if Magnitude is •NE• 0**
	BNE	140613		BRAN	BRAN	V	—	—	—	H	H	—	4	Branch if A •NE• 0
	BRBR	N040-077		IBRN	BRAN	I	—	—	—	—	H	—	—	**Branch if R bit n = 0**
	BRBS	*N000-037*		IBRN	BRAN	I	—	—	—	—	H	—	—	*Branch if R bit n = 1*
	BRD1	134		IBRN	BRAN	I	—	—	—	—	H	—	—	**Decrement R by One; Branch if R = 0**
	BRD2	135		IBRN	BRAN	I	—	—	—	—	H	—	—	Decrement R by Two, Branch if R 0
	BRD4	136		IBRN	BRAN	I	—	—	—	—	H	—	—	**Decrement R by Four, Branch if R = 0**

	BREQ	10_		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if R 0
	BRGE	105		IBRN	BRAN	**I**	—	—	—	—	H	—	4	**Branch if R ⩾ 0**
	BRGT	101		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if R 0
	BRI1	130		IBRN	BRAN	**I**	—	—	—	—	H	—	—	**Increment R by one and branch if 0**
	BRI2	131		IBRN	BRAN	I	—	—	—	—	H	—	—	Increment R by 2 and branch if 0
	BRI4	132		IBRN	BRAN	**I**	—	—	—	—	H	—	—	**Increment R by 4 and branch if 0**
	BRLE	100		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if R 0
	BRLT	104		IBRN	BRAN	**I**	—	—	—	—	H	—	4	**Branch if R < 0**
	BRNE	103		IBRN	BRAN	I	—	—	—	—	H	—	4	Branch if R •NE• 0
	C	61	**YY**	MRGR	INT	**I**	—	—	—	—	H	1	1	**Compare Fullword**
R	CAI	000411		GEN	IO	SRVI	H	H	H	H	H	—	—	Clear Active Interrupt
	CAL	141050		GEN	CLEAR	**SRV**	H	H	H	H	H	—	—	**Clear A Left**
	CALI	000705		AP	PCHI	V	—	—	—	H	H	7	6	Call Fault Handler
	CAR	141044		GEN	CLEAR	**SRV**	H	H	H	H	H	—	—	**Clear A Right**
	CAS	11		MR	SKIP	SRV	H	H	H	H	H	1	1	Compare A and Skip
	CAZ	140214		GEN	SKIP	**SRV**	H	H	H	H	H	1	1	**Compare A with Zero**
	CEA	000111		GEN	PCHI	SR	H	H	H	H	H	—	—	Compute Effective Address
	CGT	001314		BRAN	BRAN	**V**	—	—	—	H	H	6	5	**Computed GOTO**
	CGT	026		BRAN	BRAN	I	—	—	—	—	H	—	7	Computed GOTO
	CH	71	YY	MRGR	INT	**I**	—	—	—	—	H	1	1	**Compare Halfword**
	CHS	140024		GEN	INT	SRV	H	H	H	H	H	—	—	Change Sign
	CHS	040		RGEN	INT	**I**	—	—	—	—	H	—	—	**Change Sign**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	CLS	11 03		MR	LOGIC	V	—		—	H	H	1	1	Compare
	CMA	140401		GEN	LOGIC	**SRV**	H	H	H	H	H	—	—	**Complement A**
	CMH	04		RGEN	LOGIC	I		—	—	—	H	—	—	Complement
	CR	056		RGEN	CLEAR	**I**	—	—	—	—	H	—	—	**Clear**
	CRA	140040		GEN	CLEAR	**SRV**	H	H	H	H	H		—	Clear A
	CRB	140015		GEN	CLEAR	**SRV**	H	H	H	H	H	—	—	**Clear B**
	CRBL	01		RGEN	CLEAR	I	—	—	—	—	H		—	Clear High Byte 1
	CRBR	063		RGEN	CLEAR	**I**	—	—	—	—	H	—	—	**Clear High Byte 2**
	CRE	141404		GEN	CLEAR	V	—		—	H	H	—	—	Clear L
	CREP	10 02		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Call Recursive Entry Procedure**
	CRHI	0 4		RGEN	CLEAR	I			—	—	H	—	—	Clear Left Half Register
	CRHR	055		RGEN	CLEAR	**I**	—	—	—	—	H	—	—	**Clear Right Half Register**
	CRL	140010		GEN	CLEAR	**SRV**	H	H	H	H	H	—	—	Clear Long
	CRLE	141410		GEN	CLEAR	**V**	—	—	—	H	H	—	—	**Clear L and E**
	CSA	1403 0		GEN	MOVE	**SRV**	H	H	H	H	H		—	Copy Sign of A
	CSR	041		RGEN	MOVE	**I**	—	—	—	—	H	—	—	**Copy Sign**
R	CXCS	001714		GEN	WCS	**V**	—	—	—	H	H	—	—	Control Extended Control **Store**
	D	62	**YY**	MRGR	INT	**I**	—	—	—	—	H	3	1	**Divide Fullword**
	DAD	06		MR	INT	**SR**	O	O	H	H	H		1	Double Add
	DBL	000007		GEN	INT	**SR**	H	H	H	H	H	—	—	**Double Precision**
	DBLE	06 1 5		RGEN	FLT	I	—	—	—		H	—	—	Convert Single to Double

DFAD	06 02		MR	FLPT	RV	U	U	H	H	H	3	5	**Double Floating Add**
DFC	05 07	YY	MRFR	FLPT	I	—	—	—	—	H	—	1	**Double Floating Compare**
DFCM	130 174		CFN	FLPT	RV	—	O	O	H	H	3	5	**Double Floating Complement**
DFCM	144 154		RCIN	FLPT	I	—	—	—	—	H	3	1	**Double Floating Complement**
DFCS	11 02		MR	FLPT	RV	U	U	H	H	H	6	5	**Double Floating Compare and Skip**
DFD	31 33	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	**Double Floating Divide**
DFDV	17 0_		MR	FLPT	RV	U	O	O	H	H	3	5	**Double Floating Divide**
DFL	01 03	YY	MRFR	FLPT	I	—	—	—	—	H	—	—	**Double Floating Load**
DFLD	0_ 0_		MR	FLPT	RV	U	U	H	H	H	—	—	**Double Floating Load**
DFLX	15 02		MR	FLPT	V	—	—	—	H	H	—	—	**Load Double Floating Index**
DFM	15 _7	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	**Double Floating Multiply**
DFMP	16 02		MR	FLPT	RV	U	O	O	H	H	3	5	**Double Floating Multiply**
DFS	21 _3	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	**Double Floating Subtract**
DFSB	07 02		MR	FLPT	RV	U	U	H	H	H	3	5	**Double Floating Subtract**
DFST	11 13	NN	MRFR	FLPT	I	—	—	—	—	H	—	—	**Double Floating Store**
DFST	04 02		MR	FLPT	RV	U	U	H	H	H	—	—	**Double Floating Store**
DH	7_	YY	MRCR	INT	I	—	—	—	—	H	3	1	**Divide Halfword**
DH1	130		RGEN	INT	I	—	—	—	—	H	2	1	**Decrement Half Register by 1**
DH2	131		RGEN	INT	I	—	—	—	—	H	_	1	**Decrement Half Register by 2**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	DIV	17		MR	INT	V	—	—	—	H	H	3	5	Divide
	DIV	17		MR	INT	SR	O	O	H	H	H	3	5	Divide
	DLD	02		MR	MOVE	SR	O	O	H	H	H	—	—	Double Load
	DM	60	NN	MRNR	INT	I	—	—	—	—	H	—	1	Decrement Fullword
	DMH	70	NN	MRNR	INT	I	—	—	—	—	H	—	1	Decrement Halfword
	DR1	124		RGEN	INT	I	—	—	—	—	H	2	1	Decrement by One
	DR2	125		RGEN	INT	I	—	—	—	—	H	2	1	Decrement by Two
	DRX	140210		GEN	SKIP	SRV	H	H	H	H	H	—	—	Decrement and Replace Index
	DSB	07		MR	INT	SR	O	O	H	H	H	2	1	Double Subtract
	DST	04		MR	MOVE	SR	O	O	H	H	H	—	—	Double Store
	DVL	17 03		MR	INT	V	—	—	—	H	H	3	5	Divide Long
	E16S	000011		GEN	ADMOD	SRVI	H	H	H	H	H	—	—	Enter 16S Mode
	E32I	001010		GEN	ADMOD	SRV	—	—	—	—	H	\	\	Enter 32I Mode
	E32R	001013		GEN	ADMOD	SRVI	H	H	H	H	H	—	—	Enter 32R Mode
	E32S	000013		GEN	ADMOD	SRVI	H	H	H	H	H	—	—	Enter 32S Mode
	E64R	001011		GEN	ADMOD	SRVI	H	H	H	H	H	—	—	Enter 64R Mode
	E64V	000010		GEN	ADMOD	SRVI	—	—	—	H	H	—	—	Enter 64V Mode
	EAA	01 01		MR	MOVE	R	U	U	H	H	H	—	—	Effective Address to A
	EAFA 0	001300		AP	FIELD	VI	—	—	—	H	H	—	—	Load Field Address Register 0
	EAFA 1	001310		AP	FIELD	VI	—	—	—	H	H	—	—	Load Field Address Register 1
	EAI	01 01		MR	MOVE	V	—	—	—	H	H	—	—	Effective Address to L

	EALB	13 02		MR	PCTLJ	V	—	—	—	H	H	—	—	Effective Address to LB
	EAR	63	NN	MRGR	PCTLJ	I	—	—	—	—	H	—	—	Effective Address to Register
	EAXB	52	NN	MRNR	PCTLJ	I	—	—	—	—	H	—	—	Effective Address to Temporary Base
	EAXB	12 02		MR	MOVE	V	—	—	—	H	H	—	—	Effective Address to Temporary Base
R	EIO	34	NN	MRGR	IO	I	—	—	—	—	H	—	2	Execute I/O
R	EIO	14 01		MR	IO	V	—	—	—	H	H	—	—	Execute I/O
R	EMCM	000503		GEN	INIC Y	SRVI	—	H	H	H	H	—	—	Enter Machine Check Mode
R	ENB	000401		GEN	IO	SRVI	H	H	H	H	H	—	—	Enable Interrupts
	ENTR	01 03		MR	PCTLJ	R	H	H	H	H	H	—	—	Enter Recursive Procedure Stack
	EPMJ	000217		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Paging Mode and Jump
M	EPMX	000237		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Paging Mode and Jump to XCS
	ERA	05		MR	LOGIC	SRV	H	H	H	H	H	—	—	Exclusive OR to A
	ERL	05 03		MR	LOGIC	V	—	—	—	H	H	—	—	Exclusive OR to L
M	ERMJ	000701		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Restricted Execution Mode and Jump
M	ERMX	000721		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Restricted Execution Mode and Jump to WCS
R	ESIM	000415		GEN	IO	SRVI	H	H	H	H	H	—	—	Enter Standard Interrupt Mode
R	EVIM	000417		GEN	IO	SRVI	H	H	H	H	H	—	—	Enter Vectored Interrupt Mode
M	EVMJ J	000703		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Vectored Mode and Jump
M	EVMX	000723		MR	MCTL	SR	—	—	H	—	—	—	—	Enter Virtual Mode and Jump to WCS

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	FA	14 16	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	Floating Add
	FAD	06 01		MR	FLPT	RV	U	U	H	H	H	3	5	Floating Add
	FC	04 06	YY	MRFR	FLPT	I	—	—	—	—	H	—	1	Floating Compare
	FCM	140530		GEN	FLPT	RV	—	O	O	H	H	3	5	Floating Complement
	FCM	100 110		RGEN	FLPT	I	—	—	—	—	H	3	1	Floating Complement
	FCS	11 01		MR	FLPT	RV	U	U	H	H	H	6	5	Floating Compare and Skip
	FD	30 32	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	Floating Divide
	FDBL	140016		GEN	FLPT	V	—	—	—	H	H	—	—	Convert Single to Double Float
	FDV	17 01		MR	FLPT	RV	U	O	O	H	H	3	5	Floating Divide
	FL	00 02	YY	MRFR	FLPT	I	—	—	—	—	H	—	—	Floating Load
	FLD	02 01		MR	FLPT	RV	U	U	H	H	H	—	—	Floating Load
	FLOT	140550		GEN	FLPT	RV	—	O	O	H	H	6	5	Float
	FLT	105 115		RGEN	FLPT	I	—	—	—	—	H	—	—	Convert Integer to Floating
	FLTA	140532		GEN	FLPT	V	—	—	—	H	H	3	5	Convert Integer to Floating
	FLTH	102 112		RGEN	FLPT	I	—	—	—	—	H	—	—	Convert Halfword to Floating
	FLTL	140535		GEN	FLPT	V	—	—	—	H	H	8	5	Convert Long Integer to Floating
	FLX	15 01		MR	FLPT	RV	U	U	H	H	H	—	—	Load Double Word Index
	FM	24 26	YY	MRFR	FLPT	I	—	—	—	—	H	3	1	Floating Multiply
	FMP	16 01		MR	FLPT	RV	U	O	O	H	H	3	5	Floating Multiply
	FRN	140534		GEN	FLPT	RV	U	O	O	H	H	3	5	Floating Round
	FRN	107 117		RGEN	FLPT	I	—	—	—	—	H	3	1	Floating Round

	FS	20 22	YY	MRFR	FLPT	**I**	—	—	—	—	H	3	1	**Floating Subtract**
	FSB	07 01		MR	FLPT	**RV**	U	U	H	H	H	3	3	Floating Subtract
	FSGT	140515		GEN	FLPT	**RV**	—	O	O	H	H	—	4	**Floating Skip if > 0**
	FSLT	140514		GEN	FLPT	RV	—	O	O	H	H	—	1	Floating Skip if < 0
	FSMI	140512		GEN	FLPT	**RV**	—	O	O	H	H	—	4	**Floating Skip if Minus**
	FSNZ	140511		GEN	FLPT	RV	—	O	O	H	H	—	1	Floating Skip if Not Zero
	FSPL	140513		GEN	FLPT	**RV**	—	O	O	H	H	—	4	**Floating Skip if Plus**
	FST	10 12	**NN**	MRFR	FLPT	I	—	—	—	—	H	—	—	Floating Store
	FST	04 01		MR	FLPT	**RV**	U	U	H	H	H	6	6	**Floating Store**
	FSZT	140510		GEN	FLPT	RV	—	O	O	H	H	—	1	Floating Skip if Zero
R	**HLT**	000000		GEN	MCTL	**SRVI**	H	H	H	H	H	—	—	**Halt**
	I	11	**YN**	MRGR	MOVE	I	—	—	—	—	H	—	—	Interchange Register and Memory-Fullword
	IAB	000201		GEN	MOVE	**SRV**	H	H	H	H	H	—	—	**Interchange A and B**
	ICA	141340		GEN	MOVE	SRV	H	H	H	H	H	—	—	Interchange Characters in A
	ICBL	065		RGEN	MOVE	**I**	—	—	—	—	H	—	—	**Interchange Bytes and Clear Left**
	ICBR	066		RGEN	MOVE	I	—	—	—	—	H	—	—	Interchange Bytes and Clear Right
	ICHL	060		RGEN	MOVE	**I**	—	—	—	—	H	—	—	**Interchange Halves and Clear Left**
	ICHR	061		RGEN	MOVE	I	—	—	—	—	H	—	—	Interchange Halves and Clear Right
	ICL	141140		GEN	MOVE	**SRV**	H	H	H	H	H	—	—	**Interchange and Clear Left**
	ICR	141240		GEN	MOVE	**SRV**	H	H	H	H	H	—	—	**Interchange and Clear Right**
	IH	51	**YN**	MRGR	MOVE	**I**	—	—	—	—	H	—	—	**Interchange Memory and Register-Halfword**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	IH1	126		RGEN	INT	I	—	—	—	—	H	2	1	Increment by One
	IH2	127		RGEN	INT	I	—	—	—	—	H	2	1	Increment by Two
	ILE	141414		GEN	MOVE	V	—	—	—	H	H	—	—	Interchange L and E
	IM	40	NN	MRNR	INT	I	—	—	—	—	H	—	1	Increment Fullword
	IMA	13		MR	MOVL	SRV	H	H	H	H	H	—	—	Interchange Memory and A
	IMH	0	NN	MRNR	INT	I	—	—	—	—	H	—	1	Increment Halfword
R	INA	54		PIO	IO	SR	H	H	H	H	H	—	—	Input to A
R	INBC	001215		AP	PRCEX	VI	—	—	—	H	H	6	5	Interrupt Notify
R	INBN	001215		AP	PRCEX	VI	—	—	—	H	H	6	5	Interrupt Notify
R	INEC	001216		AP	PRCEX	VI	—	—	—	H	H	6	5	Interrupt Notify
R	INEN	001214		AP	PRCEX	VI	—	—	—	H	H	6	5	Interrupt Notify
R	INH	001001		GEN	IO	SRVI	H	H	H	H	H	—	—	Inhibit Interrupts
	INK	000043		GEN	KEYS	SR	H	H	H	H	H	—	—	Input Keys
	INK	070		RGEN	KEYS	I	—	—	—	—	H	—	—	Save Keys
	INT	140554		GEN	FLPT	V	—	O	O	H	H	3	5	Fix as Integer
	INT	103 113		RGEN	FLPT	I	—	—	—	—	H	—	—	Convert Floating to Integer
	INTA	140531		GEN	FLPT	V	—	—	—	H	H	3	5	Convert Floating to Integer
	INTH	101 111		RGEN	FLPT	I	—	—	—	—	H	—	—	Convert Floating to Halfword Integer
	INTL	140533		GEN	FLPT	V	—	—	—	H	H	3	5	Convert Floating to Integer Long
	IR1	122		RGEN	INT	I	—	—	—	—	H	2	1	Increment by One
	IR2	123		RGEN	INT	I	—	—	—	—	H	2	1	Increment by Two

	IRB	064		RGEN	MOVE	**I**	—	—	—	—	H	—	—	**Interchange Bytes**
	IRH	057		RGEN	MOVE	**I**	—	—	—	—	H	—	—	**Interchange Halves**
	IRS	12		MR	SKIP	**SRV**	H	H	H	H	H	—	—	**Increment Memory Replace and Skip**
R	**IRTC**	000603		GEN	IO	**VI**	—	—	—	H	H	7	6	**Interrupt Return**
R	**IRTN**	000601		GEN	IO	**VI**	—	—	—	H	H	7	6	**Interrupt Return**
	IRX	140114		GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Increment and Replace Index**
R	**ITLB**	000615		GEN	MCTL	**VI**	—	—	—	H	H	—	—	**Invalidate STLB entry**
	JDX	15 02		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump and Decrement Index**
	JEQ	02 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if = 0**
	JGE	07 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if > 0**
	JGT	05 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if > 0**
	JIX	15 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump and Increment Index**
	JLE	04 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if ≤ 0**
	JLT	06 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if < 0**
	JMP	51	NN	MRNR	PCTLJ	**I**	—	—	—	—	H	—	—	**Jump**
	JMP	01		MR	PCTLJ	**SRV**	H	H	H	H	H	—	—	**Jump**
	JNE	03 03		MR	PCTLJ	**R**	U	U	H	H	H	—	—	**Jump if •NE• 0**
	JSR	73	NN	MRGR	PCTLJ	**I**	—	—	—	—	H	—	—	**Jump to Subroutine**
	JST	10		MR	PCTLJ	**SR**	H	H	H	H	H	—	—	**Jump and Store**
	JSX	35 03		MR	PCTLJ	**RV**	H	H	H	H	H	—	—	**Jump and Store Return in Index**
	JSXB	61	NN	MRNR	PCTLJ	**I**	—	—	—	—	H	—	—	**Jump and Set XB**
	JSXB	14 02		MR	PCTLJ	**V**	—	—	—	H	H	—	—	**Jump and Set XB**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	JSY	14		MR	PCTJ	V	—	—	—	H	H	—	—	Jump and Store Y
	L	01	YY	MRGR	MOVE	I	—	—	—	—	H	—	—	Load
	LCEQ	141503		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC Equal to 0 and Set A
	LCEQ	153		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC = 0 and Set R
	LCGE	141504		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC ⩽ 0 and Set A
	LCGE	154		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC ⩾ 0 and Set R
	LCGT	141505		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC > 0 and Set A
	LCGT	155		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC > 0 and Set R
	LCLE	141501		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC ⩽ A
	LCLE	151		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC ⩽ 0 and Set R
	LCLT	141500		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC < 0 and Set A
	LCLT	150		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC < 0 and Set R
	LCNE	141502		GEN	LTSTS	V	—	—	—	H	H	—	—	Test CC •NE• 0 and Set A
	LCNE	152		RGEN	LTSTS	I	—	—	—	—	H	—	—	Test CC •NE• 0 and Set R
	LDA	02		MR	MOVE	SRV	H	H	H	H	H	—	—	Load A
	LDAR	44	NN	MRGR	MOVE	I	—	—	—	—	H	—	—	Load Addressed Register
	LDC	162 172		RGEN	CHAR	I	—	—	—	—	H	—	7	Load Character
	LDC 0	001302		CHAR	CHAR	V	—	—	—	H	H	—	7	Load Character
	LDC 1	001312		CHAR	CHAR	V	—	—	—	H	H	—	7	Load Character
	LDL	02 03		MR	MOVE	V	—	—	—	H	H	—	—	Load Long
P	LDLR	05 01		MR	MOVE	V	—	—	—	H	H	—	—	Load From Addressed Register
	LDX	35		MR	MOVE	SRV	H	H	H	H	H	—	—	Load Index
	LDY	25 2		MR	MOVE	V				H	H			Load

LEQ	003	RGLN	LISIS	I	—	—	—	—	H	—	4	**Test R 0 and Set R**
LF	140416	GEN	LTSTS	SRV	H	H	H	H	H	—	5	**Logic Set A False**
LF	016	RGEN	LISIS	I	—	—	—	—	H	—	4	Logic Set R False
LFEQ	141113	GEN	LISTS	V	—	—	—	H	H	—	4	**Test F 0, Set A**
LFEQ	023 033	RGEN	LISIS	I	—	—	—	—	H	—	4	Test F 0 Set R
LFGE	141114	GEN	LTSTS	V	—	—	—	H	H	—	4	**Test F ⩾ 0, Set A**
LFGE	024 034	RGEN	LISIS	I	—	—	—	—	H	—	4	Test F > 0 Set R
LFGT	141115	GEN	LTSIS	V	—	—	—	H	H	—	4	**Test F > 0, Set A**
LFGT	025 035	RGEN	LISTS	I	—	—	—	—	H	—	4	Test F > 0 Set R
LFLE	141111	GEN	LTSTS	V	—	—	—	H	H	—	4	**Test F ⩽ 0, Set A**
LFLE	021 031	RGEN	LISIS	I	—	—	—	—	H	—	4	Test F < 0, Set R
LFLI 0	001303	BRAN	FIELD	VI	—	—	—	H	H	—	—	**Load Field Length Register 0**
LFLI 1	001313	BRAN	FIELD	VI	—	—	—	H	H	—	—	Load Field Length Register 1
LFLT	141110	GEN	LTSTS	V	—	—	—	H	H	—	4	**Test F < 0, Set A**
LFLT	020 030	RGEN	LISIS	I	—	—	—	—	H	—	4	Test F < 0 Set R
LFNE	141112	GEN	LTSTS	V	—	—	—	H	H	—	4	**Test F •NE• 0, Set A**
LFNE	022 032	RGEN	LISIS	I	—	—	—	—	H	—	4	Test F •NE• 0 Set R
LGE	140414	GEN	LTSTS	SRV	H	H	H	H	H	—	4	**Test A ⩾ 0, Set A**
LGL	004	RGEN	LISIS	I	—	—	—	—	H	—	4	Test R 0 Set R
LGT	140415	GEN	LTSTS	SRV	H	H	H	H	H	—	4	**Test A > 0, Set A**
LGT	005	RGEN	LISIS	I	—	—	—	—	H	—	4	**Test R > 0, Set R**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	LH	11	YY	MRGR	MOVE	I	—	—	—	—	H	—	—	Load Halfword
	LHEQ	013		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH = 0; Set RH
	LHGE	004		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH ≥ 0; Set RH
	LHGT	015		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH > 0; Set R
	LHL1	04	YN	MRGR	MOVE	I	—	—	—	—	H	—	—	Load Halfword Left Shifted by 1
	LHL2	14	YN	MRGR	MOVE	I	—	—	—	—	H	—	—	Load Halfword Left Shifted by 2
	LHLE	011		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH ≤ 0; Set RH
	LHLT	000		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH < 0; Set RH
	LHNE	012		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test RH •NE• 0; Set RH
	LLE	140411		GEN	LTSTS	SRV	H	H	H	H	H	—	4	Test A ≤ 0; Set A
	LLE	001		RGEN	LTSTS	I	—	—	—	—	H	—	4	Test R < 0; Set R
	LLEQ	141513		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L = 0; Set A
	LLGE	140414		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L ≥ 0; Set A
	LLGT	141515		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L > 0; Set A
	LLL	0410XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	Long Left Logical
	LLLE	141511		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L ≤ 0; Set A
	LLLT	140410		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L < 0; Set A
	LLNE	141512		GEN	LTSTS	V	—	—	—	H	H	—	4	Test L •NE• 0; Set A
	LLR	0412XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	Long Left Rotate
	LLS	0411XX		SHFT	SHIFT	SRV	H	H	H	H	H	4	5	Long Left Shift
	LLT	140410		GEN	LTSTS	SRV	H	H	H	H	H	—	4	Test A < 0; Set A

	LLI	000		RGEN	LTSTS	**I**	—	—	—	—	H	—	4	**Test R < 0 Set R**
R	**LMCM**	000501		GEN	INTGY	**SRVI**	—	H	H	H	H	—	—	**Leave**
	LNE	140412		GEN	LTSTS	**SRV**	H	H	H	H	H	—	4	**Test A •NE• 0, Set A**
	LNL	002		RGEN	LTSTS	**I**	—	—	—	—	H	—	4	**Test R •NE• 0 Set R**
R	**LPID**	000617		GEN	MCTL	**VI**	—	—	—	H	H	—	—	**Load Process ID**
M	**LPMJ**	000215		MR	MCTL	**SR**	—	—	H	—	—	—	—	**Leave Paging Mode and Jump**
M	**LPMX**	000235		MR	MCTL	**SR**	—	—	H	—	—	—	—	**Leave Paging Mode and Jump to XCS**
R	**LPSW**	000711		SP	MCTL	**VI**	—	—	—	H	H	7	6	**Load Program Status Word**
	LRL	0400XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	**Load Right Logical Shift**
	LRR	0402XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	**Long Right Rotate**
	LRS	0401XX		SHFT	SHIFT	**SRV**	H	H	H	H	H	4	5	**Long Right Shift**
	LT	140417		GEN	LTSTS	**SRV**	H	H	H	H	H	—		**Set A 1**
	LT	017		RGEN	LTSTS	**I**	—	—	—	—	H	—	4	**Set R 1**
R	**LWCS**	001710		GEN	MCTL	**V**	—	—	—	H	H	—	—	**Load Writable Control Store**
	M	42	**YY**	MRGR	INT	**I**	—	—	—	—	H	3	1	**Multiply Fullword**
R	**MDII**	001305		GEN	INTGY	**VI**	—	—	—	H	H	—	—	**Inhibit Interleaved**
R	**MDIW**	001324		GEN	INTGY	**VI**	—	—	—	H	H	—	—	**Write Interleaved**
R	**MDRS**	001306		GEN	INTGY	**VI**	—	—	—	H	H	—	—	**Read Syndrome Bits**
R	**MDWC**	001307		GEN	INTGY	**VI**	—	—	—	H	H	—	—	**Load Write Control Register**
	MH	32	YY	MRGR	INT	**I**	—	—	—	—	H	3	1	**Multiply Halfword**
M	**MIA**	64	NN	MRGR	MCTL	**I**	—	—	—	—	H	—	—	**Microcode Entrance**
M	**MIA**	12 01		MR	MCTL	**V**	—	—	—	H	H	—	—	**Microcode Entrance**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
M	MIB	74	NN	MRGR	MCTL	I	—	—	—	—	H	—	—	Microcode Entrance
M	MIB	13 01		MR	MCTL	V	—	—	—	H	H	—	—	Microcode Entrance
	MPL	16 03		MR	INT	V	—	—	—	H	H	—	1	Multiply Long
	MPY	16		MR	INT	V	—	—	—	H	H	—	1	Multiply
	MPY	16		MR	INT	SR	0	0	H	H	H	3	1	Multiply
	N	03	YY	MRGR	LOGIC	I	—	—	—	—	H	—	—	AND
R	NFYB	001211		AP	PRCEX	VI	—	—	—	H	H	6	5	Notify
R	NFYE	001210		AP	PRCEX	VI	—	—	—	H	H	6	5	Notify
	NH	13	YY	MRGR	LOGIC	I	—	—	—	—	H	—	—	AND Halfword
	NOP	000001		GEN	MCTL	SRV	H	H	H	H	H	—	—	No Operation
	NRM	000101		GEN	INT	SR	H	H	H	H	H	—	—	Normalize
	O	23	YY	MRGR	LOGIC	I	—	—	—	—	H	—	—	OR
R	OCP	14		PIO	IO	SR	H	H	H	H	H	—	—	Output Control Pulse
	OH	33	YY	MRGR	LOGIC	I	—	—	—	—	H	—	—	OR Halfword
	ORA	03 02		MR	LOGIC	V	—	—	—	H	H	—	—	Inclusive OR
R	OTA	74		PIO	IO	SR	H	H	H	H	H	—	—	Output from A
	OTK	000405		GEN	KEYS	SR	H	H	H	H	H	7	6	Restore Keys
	OTK	071		RGEN	KEYS	I	—	—	—	—	H	7	6	Restore Keys
	PCL	41	NN	MRNR	PCTLJ	I	—	—	—	—	H	—	—	Procedure Call
	PCL	10 02		MR	PCTLJ	V	—	—	—	H	H	7	6	Procedure Call
	PID	000211		GEN	INT	SR	0	0	H	H	H	—	—	Position for Integer Divide

	PID	052		RGEN	INT	**I**	—	—	—	—	H	—	—	**Position for Integer Divide**
	PIDA	000115		CEN	INT	**V**	—	—	—	H	H	—	—	Position for Integer Divide
	PIDH	053		RGEN	INT	**I**	—	—	—	—	H	—	—	**Position for Integer Divide**
	PIDL	000305		CEN	INT	V		—	—	H	H	—	—	Position Long for Integer **Divide**
	PIM	000205		GEN	INT	**SR**	0	0	H	H	H	—	—	**Position After Multiply**
	PIM	0		RGEN	INT	I	—	—	—	—	H		1	Position After Multiply
	PIMA	000015		GEN	INT	**V**	—	—	—	H	H	3	5	**Position After Multiply**
	PIMH	1		RGEN	INT	I	—		—	—	H		1	Position After Multiply
	PIML	000301		GEN	INT	**V**	—	—	—	H	H	3	5	**Position After Multiply Long**
	PRTN	000611		CEN	POINT	VI	—		—	H	H		6	Procedure Return
	RBQ	141715		AP	QUEUE	**V**	—	—	—	H	H	—	6	**Remove From Bottom of Queue**
	RBQ	133		AP	RGEN	I			—	—	H		7	Remove From Bottom of Queue
	RCB	140200		GEN	KLYS	**SRV**	H	H	H	H	H	5	—	**Clear C-Bit (Reset)**
R	RMC	000021		GEN	INTCY	SRVI	—	H	H	H	H			Clear Machine Check
	ROT	24	**NN**	MRGR	SHIFT	I	—	—	—	—	H	4	—	**Rotate**
	RRST	000717		AP	MCTL	VI	—	—	—	H	H	—	—	Register Restore
	RSAV	000715		AP	MCTL	**VI**	—	—	—	H	H	—	—	**Register Save**
	RTN	000105		CEN	POINT	SR	H	H	H	H	H	—	—	Return
	RTQ	141714		AP	QUEUE	**V**	—	—	—	H	H	—	6	**Remove From Top of Queue**
	RTQ	13		RGEN	QUEUE	I	—	—	—	—	H		7	Remove From Top of Queue
	S	22	**YY**	MRGR	INT	**I**	—	—	—	—	H	2	1	**Subtract**
	S1A	140110		CEN	INT	SRV	H	H	H	H	H	—	1	Subtract One from A

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	S2A	140310		GEN	INT	SRV	H	H	H	H	H	2	1	Subtract Two from A
	SAR	10026X		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip on A Bit Clear
	SAS	101260		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip on A Bit Set
	SBL	07 03		MR	INT	V	—	—	—	H	H	2	1	Subtract Long
	SCA	000041		GEN	INT	SR	H	H	H	H	H	—	—	Load Shift Count into A
	SCB	1406000		GEN	KEYS	SRV	H	H	H	H	H	5	—	Set C-Bit in Keys
	SGL	000005		GEN	INT	SR	H	H	H	H	H	—	—	Single Precision
	SGT	100220		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip if A Greater Than Zero
	SH	32	YY	MRGR	INT	I	—	—	—	—	H	2	1	Subtract Halfword
	SHA	15	NN	MRGR	SHIFT	I	—	—	—	—	H	4	—	Shift Arithmetic
	SHL	05	NN	MRGR	SHIFT	I	—	—	—	—	H	4	—	Shift Logical
	SHL1	076		RGEN	SHIFT	I	—	—	—	—	H	4	1	Shift H Left One
	SHL2	077		RGEN	SHIFT	I	—	—	—	—	H	4	1	Shift H Left Two
	SHR1	120		RGEN	SHIFT	I	—	—	—	—	H	4	1	Shift H Right One
	SHR2	121		RGEn	SHIFT	I	—	—	—	—	H	4	1	Shift H Right Two
	SKP	100000		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip
R	SKS	34		PIO	IO	SR	H	H	H	H	H	—	—	Skip if Satisfied
	SL1	072		RGEN	SHIFT	I	—	—	—	—	H	4	1	Shift R Left One
	SL2	073		RGEN	SHIFT	I	—	—	—	—	H	4	1	Shift R Left Two
	SLE	101220		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip if A Less Than or Equal to Zero
	SLN	101100		GEN	SKIP	SRV	H	H	H	H	H	—	—	Skip if LSB Nonzero (A(16)=1)

	SLZ	100100	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if LSB Zero (A(16) 0)**
	SMCR	100200	GEN	INTGY	**SRV**	—	H	H	H	H	—	—	**Skip on Machine Check Clear**
	SMCS	101200	GEN	INTCY	**SRV**	—	H	H	H	H	—	—	**Skip on Machine Check Set**
	SMI	101400	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if A Minus**
R	**SMK**	170020	PIO	IO	**SR**	H	H	H	H	H	—	—	**Send Mask**
R	**SNR**	100240	GEn	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip on Sense Switch Clear**
R	**SNS**	101240	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip on Sense Switch Set**
	SNZ	101040	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if A Non Zero**
	SPL	100400	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if A Plus**
R	**SR1**	100040	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 1 Clear**
	SR1	074	RGEN	SHIFT	**I**	—	—	—	—	H	4	1	**Shift R Right One**
R	**SR2**	100010	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 2 Clear**
	SR2	075	RGEN	SHIFT	**I**	—	—	—	—	H	4	1	**Shift R Right Two**
R	**SR3**	100004	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 3 Clear**
R	**SR4**	100002	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 4 Clear**
	SRC	100001	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if C Bit is Clear**
R	**SS1**	101020	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 1 Clear**
R	**SS2**	101010	GEN	SKIP	**SRV**	H	H	H	H	H	—		**Skip if Sense Switch 2 Clear**
R	**SS3**	101004	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 3 Clear**
R	**SS4**	101002	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Sense Switch 4 Clear**
	SSC	101001	GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if C-Bit is Set**
	SSM	140500	GEN	INT	**SRV**	H	H	H	H	H	—		**Set Sign Minus**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
	SSM	042		RGEN	INT	**I**	—	—	—	—	H	—	—	**Set Sign Minus**
	SSP	140100		CIN	INT	**SRV**	H	H	H	H	H	—	—	**Set Sign Plus**
	SSP	043		RGEN	INT	**I**	—	—	—	—	H	—	—	**Set Sign Plus**
R	**SSR**	100036		GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Any Sense Switch is Clear**
R	**SSS**	101036		GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if Any Sense Switch is Set**
	ST	21	NN	MRGR	MOVE	**I**	—	—	—	—	H	—	—	**Store**
	STA	04		MR	MOVE	**SRV**	H	H	H	H	H	—	—	**Store A**
	STAC	001200		AP	MOVE	**V**	—	—	—	H	H	—	"	**Store A Conditionally**
	STAR	54	NN	MRGR	MOVE	**I**	—	—	—	—	H	—	—	**Store Addressed Register**
	STC	166 176		RGEN	CHAR	**I**	—	—	—	—	H	—	"	**Store Character**
	STC 0	001322		CHAR	CHAR	**V**	—	—	—	H	H	—	7	**Store Character**
	STC 1	001332		CHAR	CHAR	**V**	—	—	—	H	H	—	7	**Store Character**
	STCD	137		RGEN	MOVE	**I**	—	—	—	—	H	—	7	**Store Conditional Fullword**
	STCH	136		RGEN	MOVE	**I**	—	—	—	—	H	—	7	**Store Conditional Halfword**
	STEX	001315		GEN	PCTLJ	**V**	—	—	—	H	H	6	5	**Stack Extend**
	STEX	02"		RGEN	PCTLJ	**I**	—	—	—	—	H	—		**Stack Extend**
	STFA 0	001320		AP	FIELD	**VI**	—	—	—	H	H	—	—	**Store Field Address Register**
	STFA 1	001330		AP	FIELD	**VI**	—	—	—	H	H	—	—	**Store Field Address Register**
	STH	31	NN	MRGR	MOVE	**I**	—	—	—	—	H	—	—	**Store Halfword**
	STL	04 03		MR	MOVE	**V**	—	—	—	H	H	—	—	**Store Long**
	STLC	001204		AP	MOVE	**V**	—	—	—	H	H	—	7	**Store L Conditionally**
P	**STLR**	03 01		MR	MOVE	**V**	—	—	—	H	H	—	—	**Store L into Addressed Register**
	STX	15		MR	MOVE	**SRV**	H	H	H	H	H	—	—	**Store X**
	STY	35 02		MR	MOVE	**V**	—	—	—	H	H		—	**Store Y**

	SUB	07		MR	INT	**SRV**	H	H	H	H	H	2	1	**Subtract**
	SVC	000505		GEN	PCTLJ	**SRVI**	H	H	H	H	H	—	—	**Supervisor Call**
	SZE	100040		GEN	SKIP	**SRV**	H	H	H	H	H	—	—	**Skip if A Zero**
	TAB	140314		GEN	MOVI	**V**	—	—	—	H	H	—	—	**Transfer A to B**
	TAK	001015		GEN	KEYS	**V**	—	—	—	H	H	7	6	**Move A to Keys**
	TAX	140504		GEN	MOVE	**V**	—	—	—	H	H	—	—	**Transfer A to X**
	TAY	140505		GEN	MOVE	**V**	—	—	—	H	H	—	—	**Transfer A to Y**
	TBA	140604		GEN	MOVE	**V**	—	—	—	H	H	—	—	**Transfer B to A**
	TC	046		RGEN	INT	**I**	—	—	—	—	H	3	1	**Two's Complement R**
	TCA	140407		GEN	INT	**SRV**	H	H	H	H	H	2	1	**Two's Complement A**
	TCH	047		RGEN	INT	**I**	—	—	—	—	H	3	1	**Two-s Complement H**
	TCL	141210		GEN	INT	**V**	—	—	—	H	H	2	1	Two-s Complement Long
	TFLL 0	001323		GEN	FIELD	**V**	—	—	—	H	H	—	—	**Transfer Field Length Register to L**
	TFLL 1	001333		GEN	FIELD	**V**	—	—	—	H	H	—	—	Transfer Field Length Register to L
	TFLR	163,173		RGEN	FIELD	**I**	—	—	—	—	H	—	7	**Move Field Length**
	TKA	001005		GEN	KEYS	**V**	—	—	—	H	H	—	—	**Move Keys to A**
	TLFL	001321		GEN	FIELD	**V**	—	—	—	H	H	—	—	**Transfer L to Field Length Register**
	TLFL	001331		GEN	FIELD	**V**	—	—	—	H	H	—	—	Transfer L to Field Length Register
	TM	44	**NN**	MRNR	MCTL	**I**	—	—	—	—	H	—	1	**Test Memory**
	TRFL	165,175		RGEN	FIELD	**I**	—	—	—	—	H	—	7	Update Field Length
	TSTQ	141757		AP	QUEUE	**V**	—	—	—	H	H	—	6	**Test Queue**
	TSTQ	104		RGEN	QUEUE	**I**	—	—	—	—	H	—	7	Test Queue
	TXA	141034		GEN	MOVE	**V**	—	—	—	H	H	—	—	**Transfer X to A**
	TYA	141124		GEN	MOVE	**V**	—	—	—	H	H	—	—	**Transfer Y to A**
R	**VIRY**	000311		GEN	INTGY	**SRVI**	—	0	H	H	H	5	6	**Verify**
R	**WAIT**	000315		AP	PROCEX	**VI**	—	—	—	H	H	—	—	**Wait**

R	MNEM	OP CODE	RI	FORM	FUNC	MODE	1	2	3	4	5	C	CC	DESCRIPTION
W	**WCS**	0016XX		GEN	MCTL	**RV**	—	—	—	0	0	—	—	**Writeable Control Store**
	X	43	YY	MRGR	LOGIC	**I**	—	—	—	—	H	—	—	**Exclusive OR**
	XAD	001100		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal Add**
	XBTD	001145		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Binary to Decimal Conversion**
	XCA	140104		GEN	MOVE	**SRV**	H	H	H	H	H	—	—	**Exchange and Clear A**
	XCB	140240		GEN	MOVE	**SRV**	H	H	H	H	H	—	—	**Exchange and Clear B**
	XCM	001102		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal Compare**
	XDTB	001146		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal to Binary Conversion**
	XDV	001107		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal Divide**
	XEC	01 02		MR	PCTLJ	**RV**	—	—	H	H	H	—	—	**Execute**
	XED	001112		DECI	DECI	**VI**	—	—	—	—	H	X	X	**Numeric Edit**
	XH	53	YY	MRGR	LOGIC	**I**	—	—	—	—	H	—	—	**Exclusive OR Halfword**
	XMP	001104		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal Multiply**
	XMV	001101		DECI	DECI	**VI**	—	—	—	U	H	X	X	**Decimal Move**
	ZCM	001117		CHAR	CHAR	**VI**	—	—	—	U	H	X	X	**Compare Character Field**
	ZED	001111		CHAR	CHAR	**VI**	—	—	—	—	H	X	X	**Character Edit**
	ZFIL	011116		CHAR	CHAR	**VI**	—	—	—	U	H	X	X	**Fill Character Field**
	ZM	43	NN	MRNR	CLEAR	**I**	—	—	—	—	H	—	—	**Clear Fullword**
	ZMH	53	NN	MRNR	CLEAR	**I**	—	—	—	—	H	—	—	**Clear Halfword**
	ZMV	001114		CHAR	CHAR	**VI**	—	—	—	U	H	X	X	**Move Character Field**
	ZMVD	001115		CHAR	CHAR	**VI**	—	—	—	U	H	X	X	**Move Equal Length Fields**
	ZTRN	001110		CHAR	CHAR	**VI**	—	—	—	U	H	X	X	**Translate Character Fields**

DATA STRUCTURES

S=S-Mode R=R-Mode V=V-Mode I=I-Mode

Word Length

16 bits (SRV)
32 bits (I)

Byte Length

8 bits (SRVI)

Character Strings

Variable length collection of bytes from 1 to 2**17-1

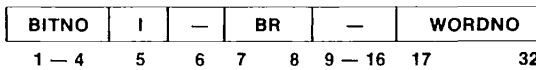
Numbers

- Unsigned 16 bit integers (SRV)
- Unsigned 32 bit integers (SRVI)
- Unsigned 64 bit integers (I)
- Signed 16-bit integers (SRVI)
- Signed 31 bit integers (SR)
- Signed 32-bit integers (VI)
- Signed 64 bit integers (VI)
- Floating Point — Single Precision, 32 bits (RVI)
- Floating Point — Double Precision 64 bits (RVI)
- Decimal — one to 63 digits in five forms (VI)

Decimal Control Word Format (VI)

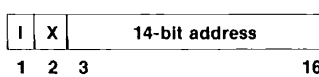
A (Bits 1-6) — Field 1, number of digits
E (Bits 14-16) — Field 1 decimal data type
B (Bit 9) — If set sign of field 1 is treated as opposite of its actual value
C (Bit 10) — If set, sign of field 2 is treated as opposite of its actual value *(XAD XMP XDV XCM only)*
D (Bit 13) — Round flag *(XMV only)*
F (Bit 17 22) — Field 2 number of digits
H (Bit 30-32) — Field 2 decimal data type
G (Bits 23-29) — Scale differential *(XAD XMV XCM* and number of multiplier digits in XMP)
T (Bit 12) — Generate positive results always
— Unused, *must* be zero

Address Pointer (AP) (VI)

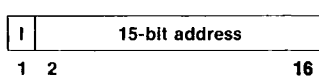


BITNO (Bits 1-4) — Bit number
I (Bit 5) — Indirect bit
BR (Bits 7-8) — Base register 00=PB, 01=SB 10=LB, 11=XB
WORDNO (Bit 17-32) — Word number offset from base register contents

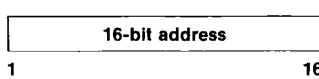
Indirect Word — One Word Memory Reference



16S



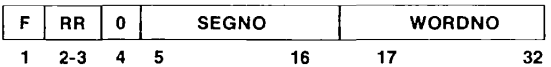
32S
32R



64R
64V

I (Bit 1) — Indirect Bit
X (Bit 2) — Index Bit

ndirect Pointer — Two Word Memory
leference (IP) (VI)



F (Bit 1) — Generate pointer fault if set. In the fault case the entire first word (bits 1-16) forms a fault code and no other bits are inspected

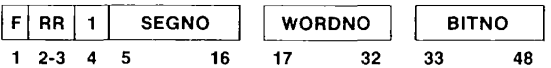
RR (Bits 2-3) — Ring of privilege — controls access rights

Bit 4 = 0 — No third word. Bit number portion of effective address is zero

SEGNO (Bits 5-16) — The segment number portion of the effective address

WORDNO (Bit 17-32) — The word number portion of the effective address

ndirect Pointer — Three Word Memory
leference (IP) (VI)



F (Bit 1) — Generate pointer fault if set. In the fault case the entire first word (bits 1-16) forms a fault code and no other bits are inspected

RR (Bits 2-3) — Ring of privilege — controls access rights

Bit 4 = 1 — The third word is present and gives the bit number portion of the effective address

SEGNO (Bits 5-16) — The segment number portion of the effective address

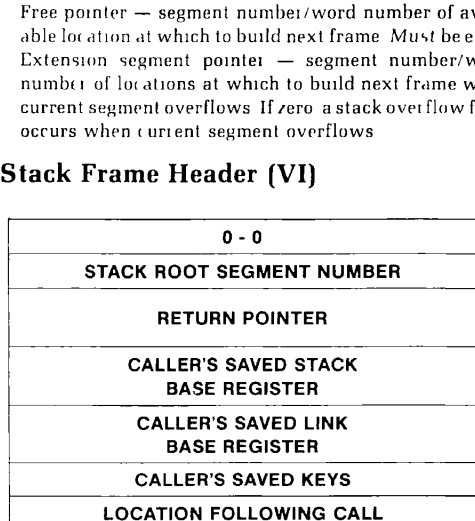
WORDNO (Bits 17-32) — The word number portion of the effective address

BITNO (Bits 33-36) — The bit number portion of the effective address

Stack Segment Header (VI)

Word	Meaning
0,1	Free pointer — segment number/word number of available location at which to build next frame. *Must* be even.
2,3	Extension segment pointer — segment number/word number of locations at which to build next frame when current segment overflows. If zero, a stack overflow fault occurs when current segment overflows.

PCL Stack Frame Header (VI)



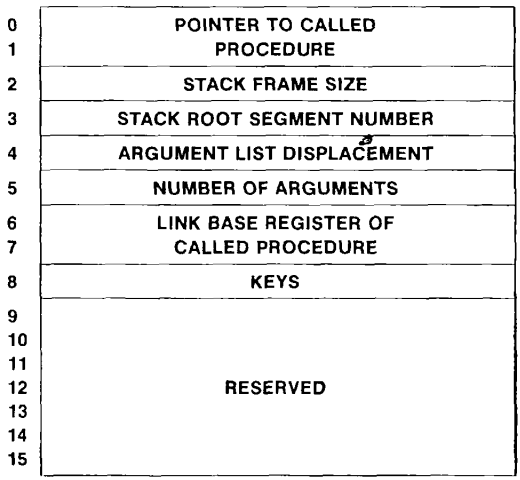
Word	Meaning
0	Flag bits — set to zero by PCL when frame is created.
1	Stack root segment number — for locating free pointer.
2,3	Return pointer — segment number/word number of location following call and argument sequence which created this frame.
4,5	Caller's saved stack base register.
6,7	Caller's saved link base register.
8	Caller's saved keys.
9	Word number of location following call — beginning of argument transfer templates, if any.

CALF Stack Frame Header (VI)

0	FLAG BITS
1	STACK ROOT SEGMENT NUMBER
2 3	RETURN POINTER
4 5	CALLER'S SAVED STACK BASE REGISTER
6 7	CALLER'S SAVED LINK BASE REGISTER
8	CALLER'S SAVED KEYS
9	LOCATION FOLLOWING CALL
10	FAULT CODE
11 12	FAULT ADDRESS
13 14 15	RESERVED

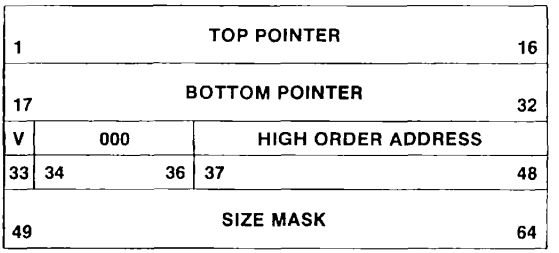
Word	Meaning
0	Flag bits — set to one by CALF instruction
1	Stack root segment number — for locating free pointer
2,3	Return pointer — segment number/word number of location return
4,5	Caller's saved stack base register
6,7	Caller's saved link base register
8	Caller's saved keys
9	Word number of location following call — beginning of argument transfer templates, if any
10	Fault code
11	Fault address
13-15	Reserved

Entry Control Block (ECB) (VI)



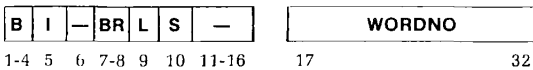
Word	Meaning
0,1	Pointer (ring segment word number) to the first executable instruction of the called procedure
2	Stack frame size to create (in words) *Must* be even
3	Stack root segment number If zero, keep same stack
4	Displacement in new frame of where to build argument list
5	Number of arguments expected
6,7	Called procedure's link base (location of called procedure's linkage frame less 400)
8	CPU keys desired by called procedure
9-15	Reserved *must* be zero

Queue Control Block (VI)



Bits	Meaning
1-16	Top pointer-read
17-32	Bottom Pointer-Write
33 (V)	Virtual/physical control bit 0 = physical queue 1 = virtual queue
34-36	Reserved — *must* be zero
37-48	Queue data block address Segment number if virtual queue High order physical address bits if physical queue
49-64	Mask — value 2**K-1 size = 2**K

Argument Transfer Template (AP) (VI)



B (Bits 1-4) — Bit number
I (Bit 5) — Indirect
BR (Bits 7 8) — Base register
00 = Procedure base (PB)
01 = Stack base (SB)
10 = Link base (LB)
11 = Temporary base (XB)
L (Bit 9) — Last template for this call
S (Bit 10) — Store argument address Last template for this argument
WORDNO (Bits 17-32) — Word number offset from base register

PROCESSOR CHARACTERISTICS

Registers (S)

Prime 100, 200 and 300 registers are 16 bits wide. All the program visible registers are physically located in high speed memory and are addressed as memory locations 0-37. In restricted mode (normal user operation) *only* 0-7 are accessable.

Memory Address	Register Designation	Function
0	X	Index Register
1	A	Arithmetic Register
2	B	Extension Arithmetic Register
3		
4		
5		
6	VSC	Visible Shift Count
7	P	Program Counter
10	PMAR *(Prime 300 only)*	Page Map Address Register
11	FCODE	Fault Code
12	FAR	Fault Address Register
13-17	Reserved	
20-37	DMA '20 '22 '36 (8 total)	Word Pairs for DMA channels (address and word counts)

Registers (R)

Prime 100, 200 and 300 registers are 16 bits wide. All the program visible registers are physically located in high speed memory and are addressed as memory locations 0-37. In restricted mode (normal user operation) *only* 0-7 are accessible.

Memory Address	Register Designation	Function
0	X	Index Register
1	A	Arithmetic Register
2	B	Extension Arithmetic Register
3	S	Stack Register
4	FLTH	Floating Point Accumulator — High
5	FLTL	Floating Point Accumulator — Low
6	FEXP	Floating Point Exponent

7	P		Program Counter
10	PMAR *(Prime 300 only)*		Page Map Address Register
11	FCODE		Fault code
12	PFAR		Page Fault Address Register
13 17	Reserved for microprogram		
20 37	DMA 20 22 (8 total)	36	Word Pairs for DMA channels (address and word counts)

Registers (VI)

Prime 400/500 registers are 32 bits wide Short form instructions reference the same registers as in Rmode

Register addresses used in LDLR and STLR instructions are doubleword addresses The notation 2 H means the high or left 16 bits of register address 2 while 2 L means the low or right 16 bits

The following registers should not be written into by STLR instructions or anomalous behavior will result

PB The procedure base should be changed *only* via LPSW or programmed transfers of control

keys The keys should be changed *only* via LPSW or the various mode control operations

modals The modals should be changed *only* via LPSW or the various mode control operations In no case should an LPSW ever attempt to change the current register set bits of the modals

NOTICE — Numbers in parentheses ( ) show P300 Address Mapping

/I-Mode Register Description:

MICROCODE SCRATCH S0			DMX RS1			CURRENT REGISTER SET (CRS) RS2 RS3			
ADR	HIGH	LOW	ADR	HIGH	LOW	ADR	ADR	HIGH	LOW
0	TR0	—	40	—		100	140	GR0	—
1	TR1	—	41	—	—	101	141	GR1	—
2	TR2	—	42	—	—	102	142	GR2(DATH)	(2 BIT)
3	TR3	—	43	—	—	103	143	GR3(FH)	(EI)
4	TR4	—	44	—	—	104	144	GR4	—
5	TR5	—	45	—	—	105	145	GR5(3 S Y)	—
6	TR6		46	—	—	106	146	GR6	—
7	TR7	—	47	—	—	107	147	GR7(0 X)	—
0	RDMX1	—	50	—	—	110	150	FAR0(18)	—
1	RDMX2	—	51	—	—	111	151	FLR0	—
2	—	RATMPL	52	—	—	112	152	FAR1(4)	(7)
3	RSGT1	—	53	—	—	113	153	FLR1 VSC(6)	—
4	RSGT2	—	54	—	—	114	154	PB	—
5	RECC1	—	55	—	—	115	155	SB(14)	(15)
6	RECC2	—	56	—	—	116	156	LB(16)	(17)
7	—	RLOIV	57	—	—	117	157	XB	—
0	ZERO	ONE	60	(20)	(21)	120	160	DTAR3(10)	—
1	PBSAVE	—	61	—	—	121	161	DTAR2	—
2	RDMX3	—	62	(?)	(23)	122	162	DTAR1	
3	RDMX4	—	63	—	—	123	163	DTAR0	—
4	CC77	CC77	64	(24)	(25)	124	164	KEYS	(MODALS)
5	—	—	65	—	—	125	165	OWNER	—
6	—		66	(26)	(?)	126	166	ECODE(11)	—
7	—	—	67	—	—	127	167	FADDR	(12)
0	PSWPB	—	70	(30)	(31)	130	170	TIMER	—
1	PSWKEYS		71	—	—	131	171	—	—
2	PPA PI A	PCBA	72	(32)	(33)	132	172	—	—
3	PPB PI B	PCBB	73	—	—	133	173	—	—
4	DSWRMA	—	74	(34)	(35)	134	174	—	—
5	DSWSTAT	—	75	—	—	135	175	—	—
6	DSWPB	—	76	(36)	(37)	136	176	—	—
7	RSAVPTR	—	77	—	—	137	177	—	—

Definitions

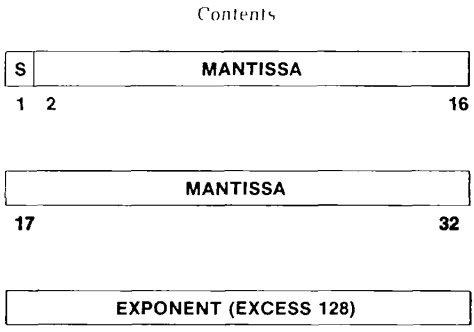
TR	Temporary Registers TR7 — Saved return pointer on a halt (automatic save)
RDMX	Register DMX RDMX1 — Used by DMC buffer start pointer RDMX2 — REA at time of DMX trap RDMX3 — Save RD during DMQ RDMX4 — Used as working register
RATMPL	Read Address Trap Map to rP Low
RSGT	Register Segmentation Trap RSGT1 — SDW2 / address of Page Map RSGT2 — contents of Page Map / DSW2
REOIV	Register End of Instruction Vector
ZERO/ONE	Constants

PBSAVE	Procedure Base SAVE saved return pointer when return pointer used elsewhere
C377	Constant
PSWPB	Processor Status Word Procedure Base return pointer for interrupt return (also used for Prime 300 compatibility)
PSWKEYS	Processor Status Word KEYS KEYS for interrupt return (also used for Prime 300 compatibility)
PPA	Pointer to Process A
PLA	Pointer to Level A
PCBA	Process Control Block A
PPB	Pointer to Process B
PLA	Pointer to Level B
PCBB	Process Control Block B
DSWRMA	Diagnostic Status Word RMA RMA at last Check Trap
DSWSTAT	Diagnostic Status Word STATus
DSWPB	Diagnostic Status Word Procedure Base Return pointer or PBSAVE at last check
RSAVPTR	Register SAVE Pointer Location of Register Save Area after Halt
GR	General Register
FAR1	Field Address Register
FLR1	Field Length Register
FAR2	Field Address Register
FLR2	Field Length Register
PB	Procedure Base PBH — RPH PBL — 0
SB	Stack Base
LB	Link Base
XB	Temp (auxiliary) base
DTAR	Descriptor Table adr reg
KEYS	See below
MODALS	See below
OWNER	OWNER
FCODE	Fault CODE
FADDR	Fault ADDRess
TIMER	TIMER

General Registers — 32 bits (I)

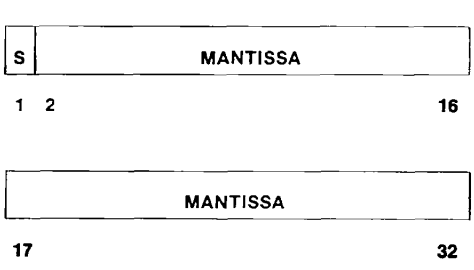
The eight general registers are numbered from 0 7 1 7 may be used for index registers All are used as fixed point and logical accumulators in register to memory and register to register operations

Floating Point Register — Single Precision (R)



Floating Point Register — Double Precision (R)

Prime 300



'02

'06

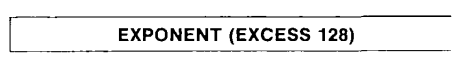
Floating Point Register — Single Precision (V

Register *Contents*

12H

12L

13H



Floating Point Register — Double Precision (V)

Register *Contents*

12H

12L

13L

13H

EXPONENT (EXCESS 128)

49 64

loating Point Registers — 64 bits (I)

he two floating point registers are numbered 0 and 1 They ıre ıed as single and double precision accumulators in register to emory and register to register operations The two floating point gisters overlap the two field length and address registers There re any operation which changes floating point register 0 stroys field length and address register 0 and vice versa The ime is true for floating point register 1 and field length and ldress register 1

ase Registers (VI)

PB — Procedure Base
SB — Stack Base
LB — Link Base
XB — Temporary Base

0	RING	0	SEGNO	WORDNO
1	2 3	4	5 16	17 32

RING (Bits 2 3) — Ring Number
SEGNO (Bits 5 16) — Segment Number
WORDNO (Bits 17 32) — Word Number

ıeld Address and Length Registers (VI)

0	RING	0	SEGNO	WORDNO	LENGTH	BITNO	0	LENGTH
1	2-3	4	5-16	17-32	33-48	49-52	53-59	60-64

RING (Bits 2 3) — Ring Number
SEGNO (Bits 5 16) — Segment Number
WORDNO (Bits 17 32) — Word Number
LENGTH (Bits 33 48 60 64) — Length
BITNO (Bits 49 52) — Bit Number

Field Registers — 64 bits (I)

The field registers, numbered 0 and 1, have the same meanii as in V-mode They are distinguished from the floating poi registers by the instructions which use them See comme under Floating Point Registers

Keys (S,R)

Process status information is available in a word called tl keys, which can be read or set by the program Its format is follows

C	DBL	—	Mode	0	Bits 9-16 of location 6
1	2	3	4-6	7-8	9 — 16

C (Bit 1) — Set by arithmetic error conditions
DBL (Bit 2) — Single Precision, 1 — Double Precision
MODE (Bits 4-6) — The current addressing mode as follow
000 = 16S
001 = 32S
011 = 32R
010 = 64R
110 = 64V
100 = 32I

C-Bit (SR)

Bit 1 in the keys Set by arithmetic error conditions (Bit 1)

Keys (VI)

C	0	L	MODE	F	X	LT	EQ	DEX	0 — 0	I	S
1	2	3	4-6	7	8	9	10	11	12 - 14	15	16

C (Bit 1) — C-Bit
L (Bit 3) — L-Bit
MODE (Bits 4-6) — Addressing Mode
000 = 16S
001 = 32S
011 = 32R
010 = 64R
110 = 64V
100 = 32I
F (Bit 7) — Floating point exception disable
0 = take fault
1 = set C bit

X (Bit 8) — Integer Exception enable
0 = set C-bit
1 = take fault

LT (Bit 9) — Condition code bits

EQ (Bit 10)
LT = negative
EQ = equal

DEX (Bit 11) — Decimal exception enable
0 = set C bit
1 = take fault

I (Bit 15) — In dispatcher — set/cleared *only* by process exchange

S (Bit 16) — Save done — set/cleared *only* by process exchange

-Bit (VI)

Set by error conditions in arithmetic operations

-Bit (VI)

Set by an arithmetic or shift operation except IRS IRX, DRX Equal to carry out of the most significant bit (bit 1) of an arithmetic operation

ondition Code Bits (VI)

The two condition-code bits are designated 'EQ' and LT" EQ is set if and only if the result is zero, if overflow occurs, EQ reflects the state of the result after truncation rather than before LT reflects the extended sign of the result (before truncation if overflow) and is set if the result is negative

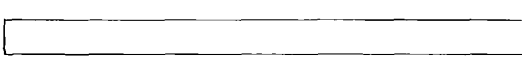
lodals (VI)

E	V	0	CURREG	MIO	P	S	MCK
1	2	3-8	9-11	12	13	14	15-16

E (Bit 1) — Interrupts enabled
V (Bit 2) — Vectored-interrupt mode
CURREG (Bits 9-11) — Current register set (set/cleared *only* by process exchange)
MIO (Bit 12) — Mapped I/O mode
P (Bit 13) — Process-exchange mode
S (Bit 14) — Segmentation mode
MCK (Bits 15-16) — Machine check mode

INSTRUCTION FORMATS (SRV)

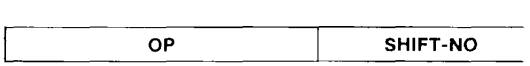
GENERIC (SRV)



1 1

Bits 3 6 are *always* zero

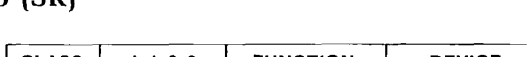
SHIFT (SR)



1 10 11 1

OP (Bits 1-10) — Opcode — Bits 3-6 are *always* ze
SHIFT-NO (Bits 11-16) — Two s complement of the numb
of places to be shifted 0=63 plac

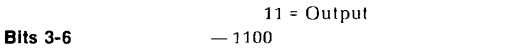
I/O (SR)



1 2 3 6 7 10 11 1

CLASS (Bits 1 2) — Type of I/O instruction
00 = Control
01 = Sense
10 = Input
11 = Output
Bits 3-6 — 1100
FUNCTION (Bits 7-10) — Subdivision of class Device
dependent
DEVICE (Bits 11 16) — Device type

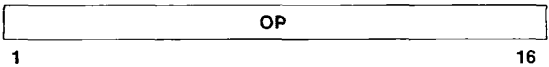
DECIMAL (V)



1 1

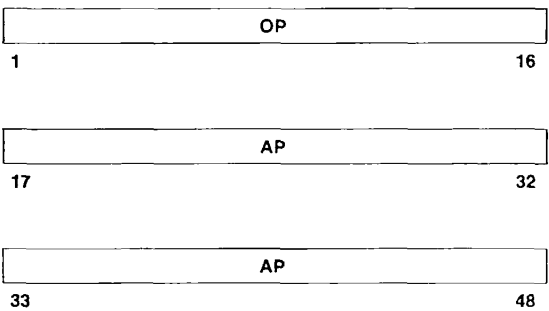
OP (Bits 1-16) — Opcode This instruction uses previously s
up field registers and a previously set
control word in register L (General Regist
2 in 32 1 mode)

CHARACTER (V)



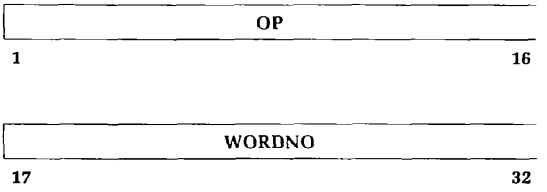
OP (Bits 1-16) — Opcode. This instruction uses previously set up field registers.

GENERIC AP (V)



OP (Bits 1-16) — Opcode
AP Bits (17-48) — Address Pointer — see AP in Data Structures

BRANCH (V)



OP (Bits 1-16) — Opcode
WORDNO (Bits 17-32) — Word number offset from procedure base register

Memory Reference Instruction Format (SRV)

Type	No Words	S	D	CB	Mode
Basic	1	0	0 777	—	SR
Sector Relative	1	1	0 777	—	S
Procedure Relative	1	1	241 to +256 -224 to +256	—	R
Stack Postincrement/ Predecrement	1	1	256 to 241	2 3	R
Base Register Relative	1	0	0 777	—	V
Long Reach	2	1	-256 to -241	0 2	R
Stack Relative	2	1	256 to 241	1 3	R
Base Registers	2	1	-256 to 224	—	V

Memory reference formats for S R and V modes are shown in the Addressing Mode Summaries Field mnemonics are

- **I** — Indirect bit
- **X** — Index bit
- **Y** — Second index bit (V mode only)
- **OX** — Opcode Extention
- **S** — Sector bit
- **D** — Displacement field
- **CB** — Class bits
- **A** — 16 bit address word — two word instructions
- **SP** — Stack Pointer
- **SB** — Stack Base register
- **LB** — Link Base register
- **XB** — Temporary Base register
- **PB** — Procedure Base register

INSTRUCTION FORMATS (I)

The three primary instruction formats and their subcategories are discussed below

Non Register Generic:

These instructions are a subset of the V-mode generics and are decoded in the same way

Register Generic:

These instructions operate on the specified register, which may be a general, field or floating register This class includes the branch instructions, where the branch address, in the second word, is a 16-bit procedure base displacement

Memory Reference:

There are three types of memory reference instructions

MRNR *Fixed Point Logical* Data location 2nd word *Memory*

OP	R	AD	S	B	D
1-6	7-9	10-11	12-14	15-16	17-36

MRGR *Fixed Point Logical* Data location 2nd word *Immediate Register Memory*

OP	110	OP	AD	S	B	D
1-3	4-6	7-9	10-11	12-14	15-16	17-36

MRFR *Floating* Data location 2nd Word *Immediate Register Memory*

OP	110	OP	FR	OP	AD	S	B	D
1-3	4-6	7	8	9	10-11	12-14	15-16	17-36

D	S	B	Effective Address/Instruction Type
3	>0	—	(D+B) *+S *(indirect post index)*
3	0	—	(D+B) * *(indirect)*
2	>0	—	(D+B+S) * *(pre index indirect)*
2	0	—	(D+B) * *(indirect)*
1	>0	—	D+B+S+ *(indexed)*
1	0	—	D+B *(direct)*
0	≥0	0	REG-REG *(S is operand)*
0	0	1	Immediate Type 1
0	>0	1	Immediate Type 2
0	0	2	Immediate Type 3
0	1	2	Floating Reg Source *(FR0)*
0	2	2	*Undefined (will not generate UII)*
0	3	2	Floating Reg source *(FR1)*
0	4-7	2	Undefined *(will not generate UII)*
0	—	3	Undefined *(will not generate UII)*

ield Mnemonics:

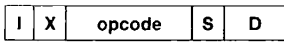
- **OP** — Opcode
- **R** — Destination register
- **AD** — Address computation code
- **S** — Source register
- **B** — Base register
- **FR** — Floating register
- **D** — Displacement

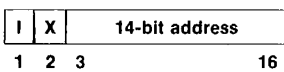
ADDRESSING MODE SUMMARIES AND FLOW CHARTS (SRV)

16S SUMMARY

Address Length 14 bits, 16K word address space

Format

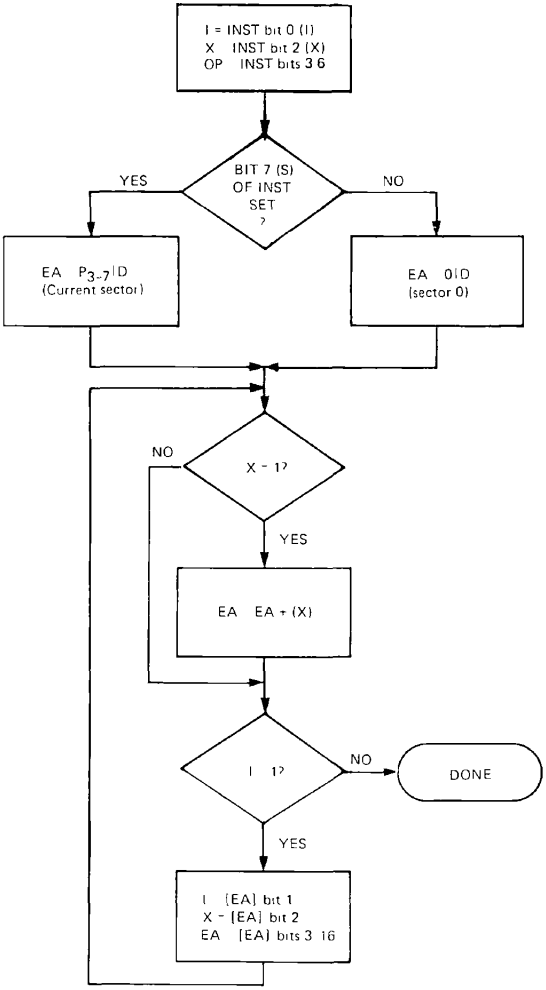




Indexing Multiple levels In an indirect word the index calculation is done *before* the indirection

Indirection Multiple levels

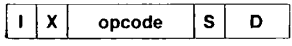
I	X	S	D	EA	Assembler Notation	Type
0	0	0	0 to '777	**0|D**	**LDA ADDR**	*Direct*
0	1	0	0 to '777	**0|D+X**	**LDA ADDR,1**	*Indexed*
1	0	0	0 to '777	**I (0|D)**	**LDA ADDR,***	*Indirect*
1	1	0	0 to '777	**I (0|D+X)**	**LDA ADDR,1***	*Indirect preindexed*
0	0	1	0 to '777	**P|D**	**LDA ADDR**	*Direct*
0	1	1	0 to '777	**P|D+X**	**LDA ADDR,1**	*Indexed*
1	0	1	0 to '777	**I (P|D)**	**LDA ADDR,***	*Indirect*
1	1	1	0 to '777	**I (P|D+X)**	**LDA ADDR,1***	*Indirect preindexed*


I = INST bit 0 (I)
X INST bit 2 (X)
OP INST bits 3 6
BIT 7 (S) OF INST SET ?
YES
NO
EA P3-7|D
(Current sector)
EA 0|D
(sector 0)
X - 1?
NO
YES
EA EA + (X)
I 1?
NO
DONE
YES
I [EA] bit 1
X - [EA] bit 2
EA [EA] bits 3 16

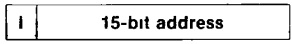
32S (INCLUDES 32R WHEN S=0) SUMMARY

Address Length 15 bits 32K word address space

Format



Instruction Word

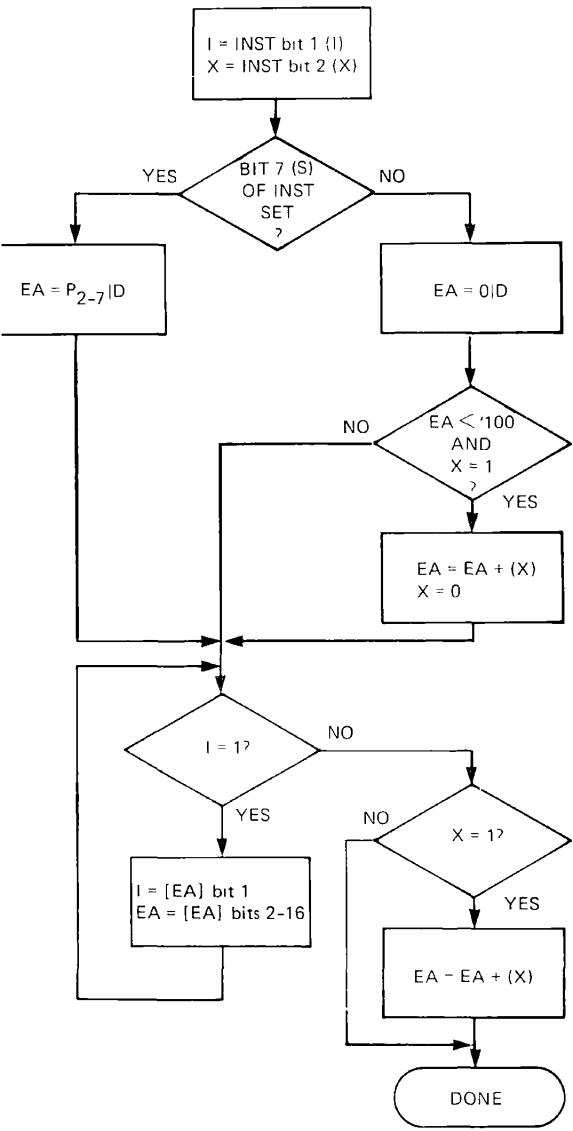


Indirect address word

Indexing One level The 15-bit indirect address word eliminat the X bit Done after all indirection is complete, except for the sp cial case shown in the table below

Indirection Multiple levels

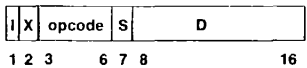
I	X	S	D	EA	Assembler Notation	Type
0	0	0	0 to '777	**0|D**	**LDA ADDR**	*Direct*
0	1	0	0 to '777	**0|D+X**	**LDA ADDR,1**	*Indexed*
1	0	0	0 to '777	**I (0|D)**	**LDA ADDR,***	*Indirect*
1	1	0	0 to '77	**I (0|D+X)**	**LDA ADDR,1***	*Indirect preindexed*
1	1	0	100 to 777	**I (0|D)+X**	**LDA ADDR,*1**	*Indirect postindexed*
0	0	1	0 to '777	**P|D**	**LDA ADDR**	*Direct*
0	1	1	0 to '777	**P|D+X**	**LDA ADDR,1**	*Indexed*
1	0	1	0 to '777	**I (P|D)**	**LDA ADDR,***	*Indirect*
1	1	1	0 to 777	**I (P|D)+X**	**LDA ADDR,1***	*Indirect postindexed*


I = INST bit 1 (I)
X = INST bit 2 (X)
BIT 7 (S) OF INST SET ?
YES
NO
EA = P2-7 ID
EA = 0 ID
EA < '100 AND X = 1 ?
NO
YES
EA = EA + (X)
X = 0
I = 1?
NO
YES
X = 1?
NO
YES
I = [EA] bit 1
EA = [EA] bits 2-16
EA = EA + (X)
DONE

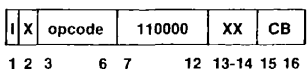
32R SUMMARY

Address Length 15 bits 32K word address space

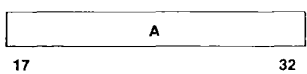
Format



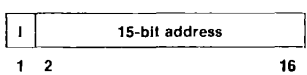
Instruction Word
S 0 or S-1
D≥-240



Instruction Word
S-1
D< 240



Address Word
Long Reach and
Stack Relative

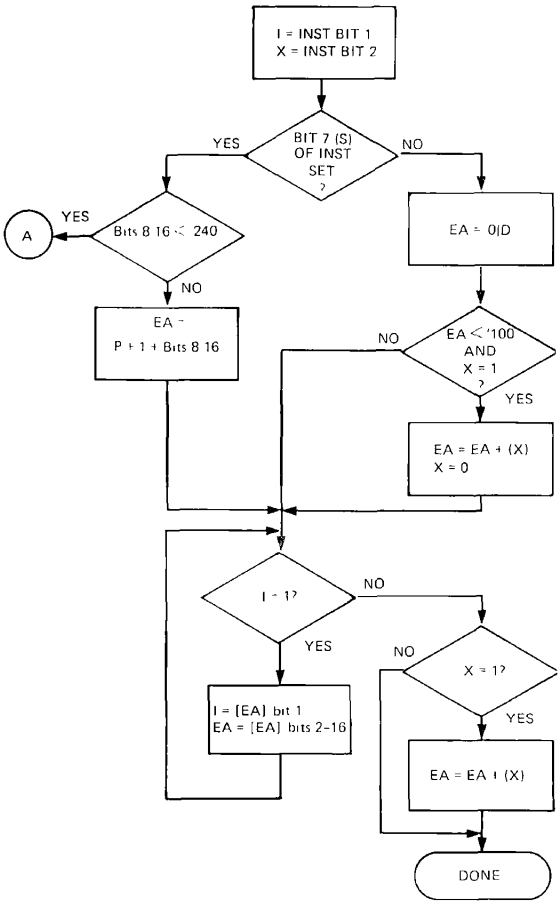


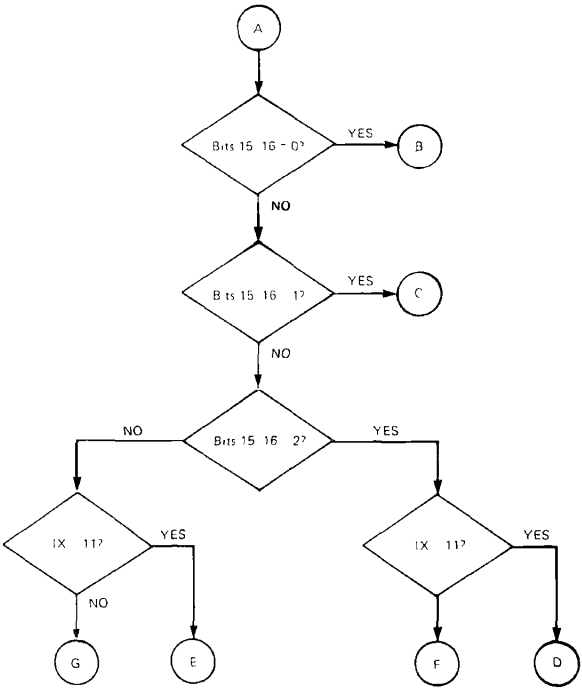
Indirect Address
Word

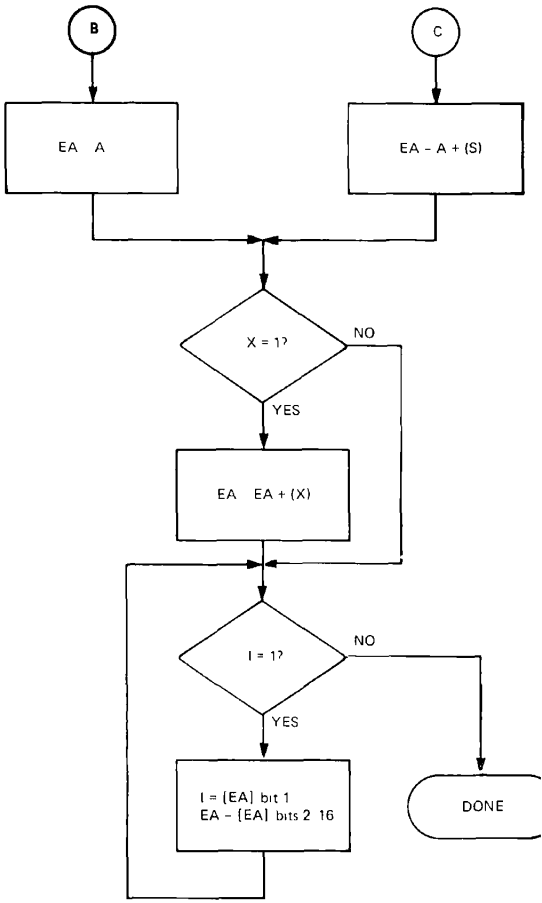
Indexing One level

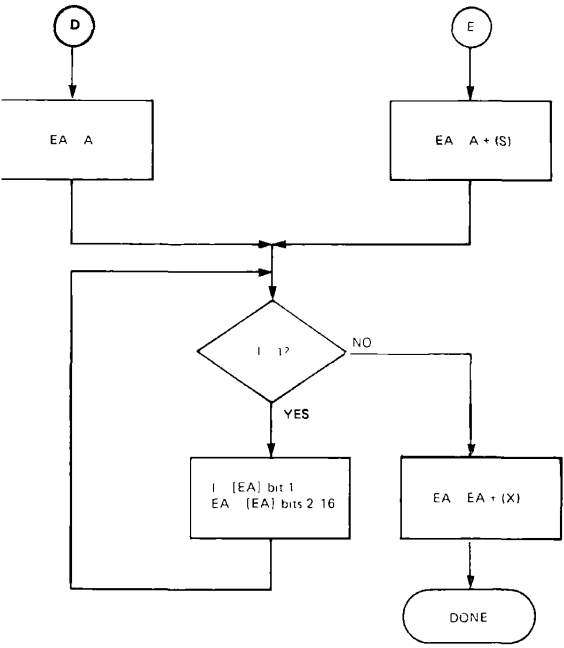
Indirection Multiple levels

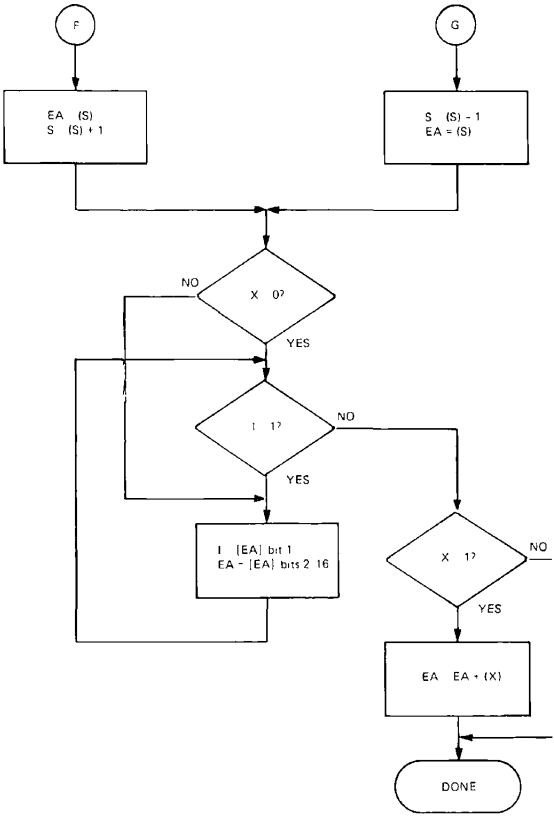
X	S	CB	D	FA	Assembler Notation	Type
0	0	—	0 to 777	0|D	LDA ADDR	*Direct*
1	0	—	0 to 777	0|D+X	LDA ADDR 1	*Indexed*
0	0	—	0 to 777	I (0|D)	LDA ADDR *	*Indirect*
1	0	—	0 to 77	I (0|D+X)	LDA ADDR 1*	*Indirect preindexed*
1	0	—	100 to 777	I (0|D)+X	LDA ADDR *1	*Indirect postindexed*
0	1	—	-240 to +256	P+D	LDA ADDR	*Direct*
1	1	—	-240 to +256	P+D+X	LDA ADDR 1	*Indexed*
0	1	—	-240 to +256	I (P+D)	LDA ADDR *	*Indirect*
1	1	—	-240 to +256	I (P+D)+X	LDA ADDR *1	*Indirect postindexed*
0	1	2	——	SP	LDA @+	*Postincrement*
1	1	2	——	I (SP)+X	LDA @+ *1	*Postincrement indirect postindexed*
0	1	2	——	I (SP)	LDA @+ *	*Postincrement indirect*
0	1	3	——	SP-1	LDA -@	*Predecrement*
1	1	3	——	I (SP-1)+X	LDA -@ *1	*Predecrement indirect postindexed*
0	1	3	——	I (SP-1)	LDA -@ *	*Predecrement indirect*
0	1	0	——	A	LDA% ADDR	*Direct long reach*
1	1	0	——	A+X	LDA% ADDR X	*Indexed long reach*
0	1	0	——	I (A)	LAD% ADDR *	*Indirect long reach*
1	1	2	——	I (A+X)	LDA% ADDR X*	*Indirect preindexed long reach*
1	1	2	——	I (A)+X	LDA% ADDR,*X	*Indirect postindexed long reach*
0	1	1	——	A+SP	LDA @+ADDR	*Direct stack relative*
1	1	1	——	A+SP+X	LDA @+ADDR X	*Indexed stack relative*
0	1	1	——	I (A+SP)	LDA @+ADDR *	*Indirect stack relative*
1	1	1	——	I (A+SP+X)	LDA @+ADDR X*	*Indirect preindexed stack relative*
1	1	3	——	I (A+SP)+X	LDA @+ADDR *X	*Indirect postindexed stack relative*


I = INST BIT 1
X = INST BIT 2
BIT 7 (S) OF INST SET ?
YES
NO
A
YES
Bits 8 16 < 240
NO
EA = 0ID
EA = P + 1 + Bits 8 16
EA < '100 AND X = 1 ?
NO
YES
EA = EA + (X)
X = 0
I = 1?
NO
YES
I = [EA] bit 1
EA = [EA] bits 2-16
X = 1?
NO
YES
EA = EA + (X)
DONE


A
Bits 15 16 = 0?
YES
B
NO
Bits 15 16 = 1?
YES
C
NO
Bits 15 16 = 2?
NO
YES
IX = 11?
YES
NO
IX = 11?
YES
G
E
F
D


B
C
EA A
EA - A + (S)
X = 1?
NO
YES
EA EA + (X)
I = 1?
NO
YES
I = (EA) bit 1
EA - (EA) bits 2 16
DONE

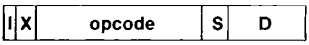

D
E
EA A
EA A + (S)
I 1?
NO
YES
I [EA] bit 1
EA [EA] bits 2 16
EA EA + (X)
DONE


F
G
EA (S)
S (S) + 1
S (S) - 1
EA = (S)
NO
X 0?
YES
I 1?
NO
YES
I [EA] bit 1
EA - [EA] bits 2 16
X 1?
NO
YES
EA EA + (X)
DONE

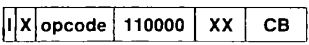
64R SUMMARY

Address Length 16 bits 64K word address space

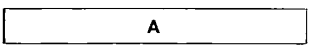
Format



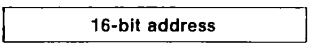
Instruction Word
S=0 or S=1
D ⩽-240



Instruction
S=1
D<-240



Address Word
Long Reach and
Stack Relative

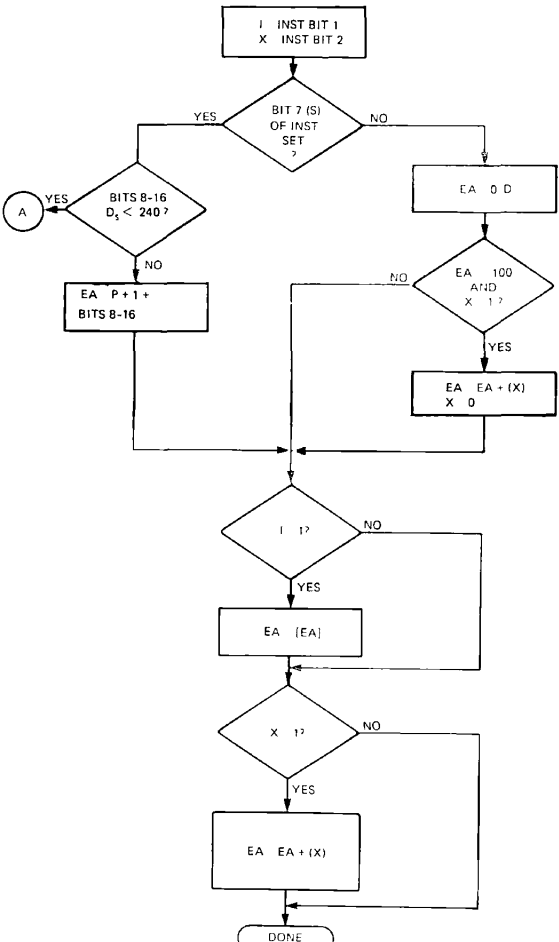


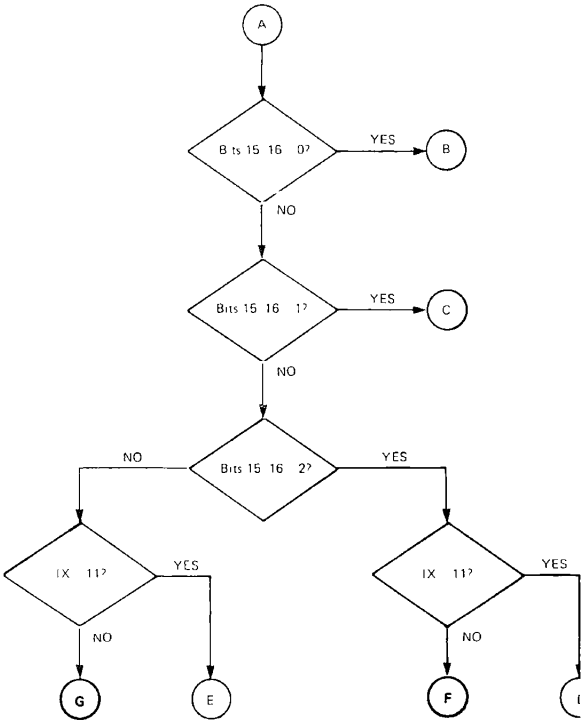
Indirect Address
Word

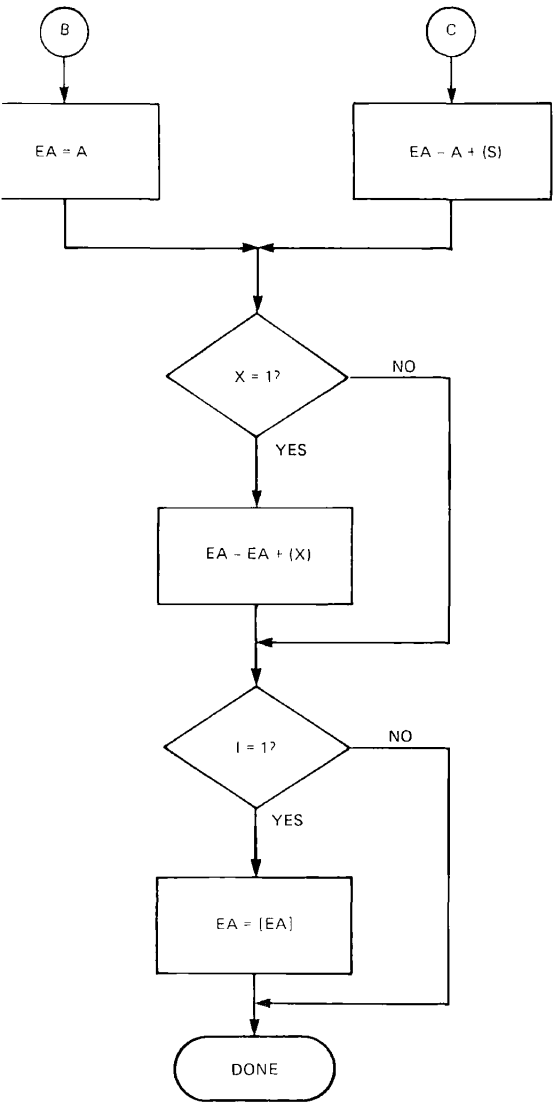
Indexing One level

Indirection One level

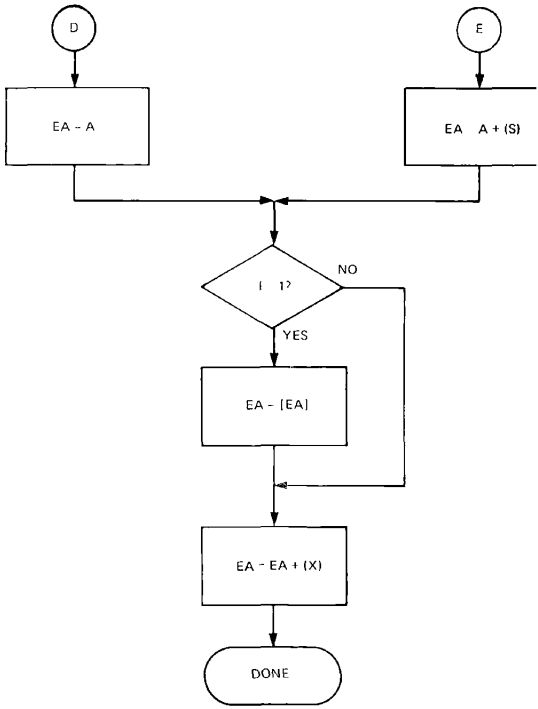
I	X	S	CB	D	EA	Assembler Notation	Type
0	0	0		0 to '777	0|D	LDA ADDR	Direct
0	1	0		0 to '777	0|D+X	LDA ADDR,1	Indexed
1	0	0	—	0 to '777	I (0|D)	LDA ADDR,*	Indirect
1	1	0	—	0 to '77	I (0|D+X)	LDA ADDR,1*	Indirect preindexed
1	1	0	—	100 to 777	I (0|D)+X	LDA ADDR*1	Indirect postindexed
0	0	1	—	-240 to +256	P+D	LDA ADDR	Direct
0	1	1	—	-240 to +256	P+D+X	LDA ADDR,1	Indexed
1	0	1	—	-240 to +256	I(P+D)	LDA ADDR,*	Indirect
1	1	1	—	-240 to +256	I (P+D)=X	LDA ADDR,*1	Indirect postindexed
0	0	1	2	——	SP	LDA @+	Postincrement
0	1	1	2	——	I (SP)+X	LDA @+,*1	Postincrement indirect postindexed
1	0	1	2	——	I (SP)	LDA @+,*	Postincrement, indirect
0	0	1	3	——	SP-1	LDA -@	Predecrement
0	1	1	3	——	I (SP-1)+X	LDA -@,*1	Predecrement indirect postindexed
1	0	1	3	——	I (SP-1)	LDA -@,*	Predecrement indirect
0	0	1	0	——	A	LDA% ADDR	Direct long reach
0	1	1	0	——	A+X	LDA% ADDR,X	Indexed long reach
1	0	1	0	——	I (A)	LDA% ADDR,*	Indirect long reach
1	1	1	0	——	I (A+X)	LDA% ADDR,X*	Indirect preindexed long reach
1	1	1	0	——	I (A)+X	LDA% ADDR,*X	Indirect postindexed long reach
0	0	1	1	——	A+SP	LDA @+ADDR	Direct stack relative
0	1	1	1	——	A+SP+X	LDA @+ADDR,X	Indexed stack relative
1	0	1	1	——	I (A=SP)	LDA @ADDR,*	Indirect stack relative
1	1	1	1	——	I (A+SP+X)	LDA @+ADDR,X*	Indirect preindexed stack relative
1	1	1	3	——	I (A+SP)+X	LDA @+ADDR,*X	Indirect postindexed stack relative

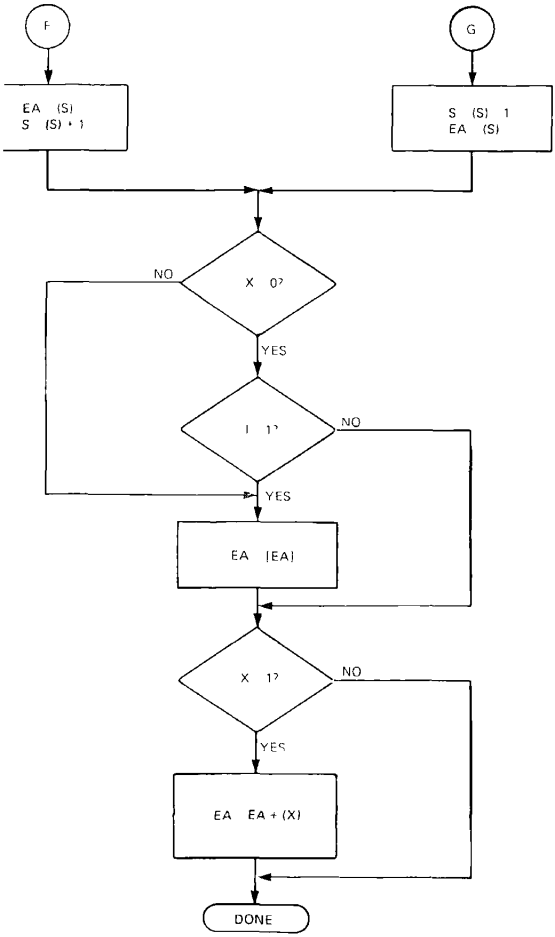

I INST BIT 1
X INST BIT 2
BIT 7 (S) OF INST SET ?
YES
NO
BITS 8-16 D$_s$ < 240 ?
YES
A
NO
EA P + 1 + BITS 8-16
EA 0 D
EA 100 AND X 1 ?
NO
YES
EA EA + (X)
X 0
I 1?
NO
YES
EA [EA]
X 1?
NO
YES
EA EA + (X)
DONE


A
Bits 15 16 0?
YES
B
NO
Bits 15 16 1?
YES
C
NO
Bits 15 16 2?
NO
YES
IX 11?
YES
NO
G
E
IX 11?
YES
NO
F


B
C
EA = A
EA = A + (S)
X = 1?
NO
YES
EA = EA + (X)
I = 1?
NO
YES
EA = (EA)
DONE

64R FLOWCHART 4

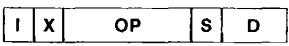



F
G
EA (S)
S (S) + 1
S (S) 1
EA (S)
X 0?
NO
YES
I 1?
NO
YES
EA [EA]
X 1?
NO
YES
EA EA + (X)
DONE

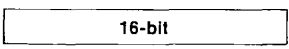
64V PROCEDURE RELATIVE (One Word, S=1)

Address length 16 bits 64K word address space

Format



Instruction Word



Indirect address w

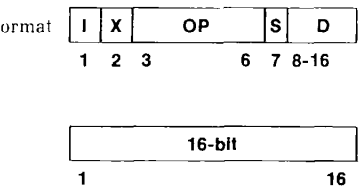
Indexing One level

Indirection One level

I	X	S	D	EA	Type
0	0	1	-224 to +256	**P+D**	Direct
0	1	1	-224 to +256	**P+D+X**	*Indexed*
1	0	1	-224 to +256	**I (P+D)**	*Indirect*
1	1	1	-224 to +256	**I (P+D)+X**	*Indirect, postindexe*

64V BASE REGISTER RELATIVE (One Word, S=0)

Address Length 3 64K segments

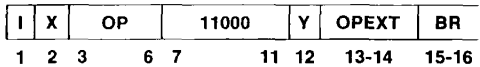


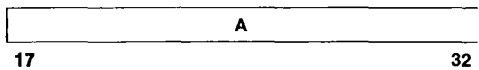
X	S	D	EA	Type
0	0	0-'7	**register location**	*Direct*
		'10-'377	**SB+D**	
		'400-'377	**LB+D**	
1	0	0-'377	**if D+X<'10 then EA=register location else SB+D+X**	*Indexed*
		'400-'777	**LB+D+X**	
0	0	0-'7	**I (REG)**	*Indirect*
		'10 '777	**I (PB|D)**	
1	0	0-'77	**I (PB|D+X)**	*Indirect, preindexed*
1	0	'100-'777	**I (PB|D)+X**	*Indirect postindexed*

64V TWO WORD MEMORY REFERENCE

Address Length 28 bits, 4096 64K segments

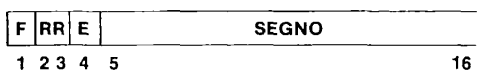
Format

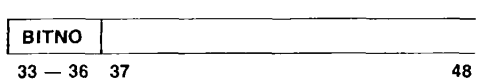




Indexing X and Y

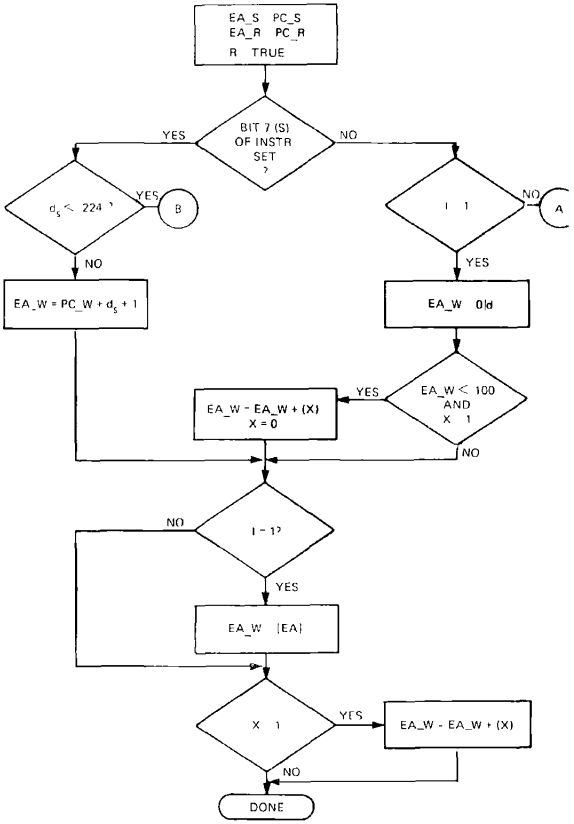
Indirection 48 bit word

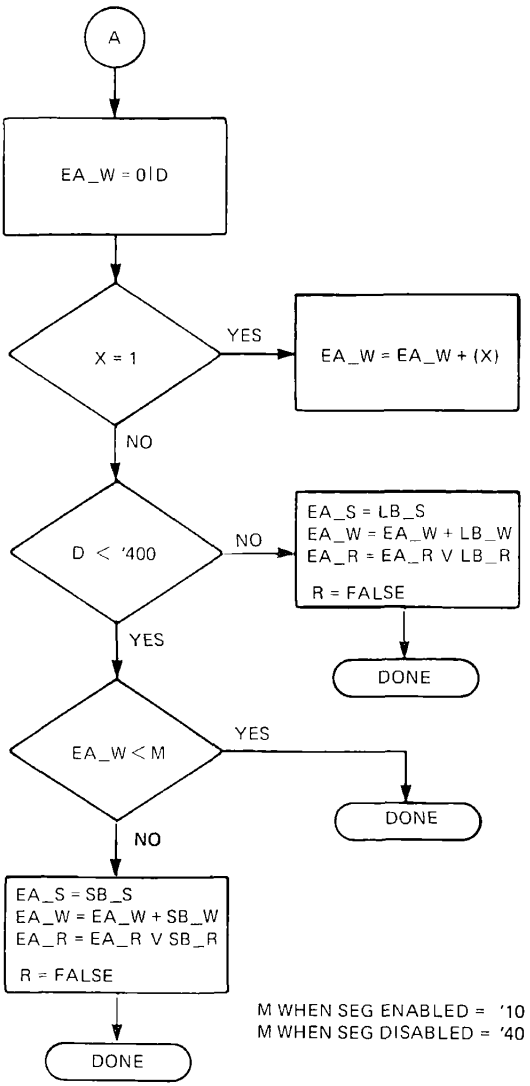


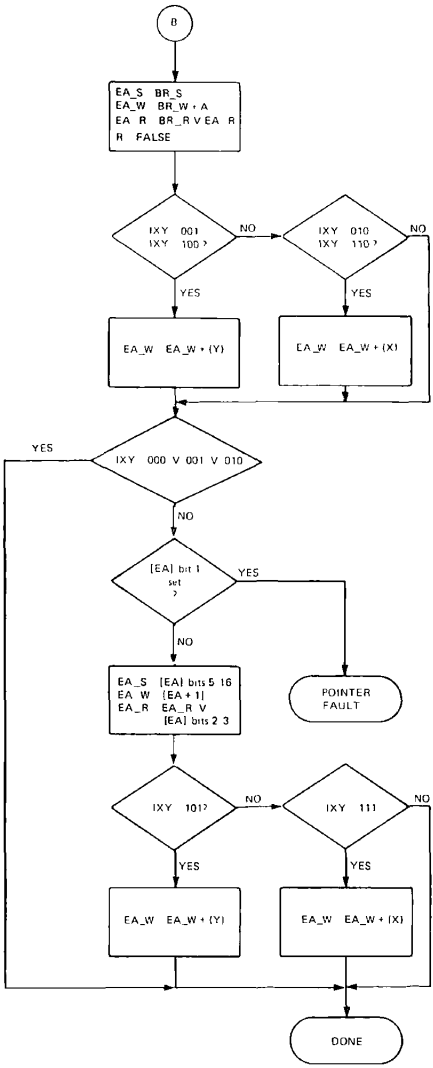


I	X	Y	BR	EA	Meaning
0	0	0	0	**PB | D**	
			1	**SB+D**	*Direct*
			2	**LB+D**	
			3	**XB+D**	
0	0	1	0	**PB |D+Y**	
			1	**SB+D+Y**	*Indexed by Y*
			2	**LB+D+Y**	
			3	**XB+D+Y**	
0	1	0	0	**PB |D+X**	
			1	**SB+D+X**	
			2	**LB+D+X**	*Indexed by X*
			3	**XB+D+X**	
0	1	1	0	**I (PB |D)**	
			1	**I (SB+D)**	*Indirect*
			2	**I (LB+D)**	
			3	**I (XB+D)**	
1	0	0	0	**I (PB |D+Y)**	
			1	**I (SB+D+Y)**	*Pre indexed by Y*
			2	**I (LB+D+Y)**	
			3	**I (XB+D+Y)**	
1	0	1	0	**I (PB |D)+Y**	
			1	**I (SB+D)+Y**	*Post-indexed by Y*
			2	**I (LB+D)+Y**	
			3	**I (XB+D)+Y**	
1	1	0	0	**I (PB |D+X)**	
			1	**I (SB+D+X)**	*Pre-indexed by X*
			2	**I (LB+D+X)**	
			3	**I (XB+D+X)**	
1	1	1	0	**I (PB |D)+X**	
			1	**I (SB+D)+X**	*Post indexed by X*
			2	**I (LB+D)+X**	
			3	**I (XB+D)+X**	

Note LDX and STX instructions may only be direct or indirect


EA_S PC_S
EA_R PC_R
R TRUE
BIT 7 (S) OF INSTR SET ?
YES
NO
$d_5 < 224$?
YES
B
NO
EA_W = PC_W + d_5 + 1
I 1
NO
A
YES
EA_W Old
EA_W < 100 AND X 1
YES
EA_W = EA_W + (X)
X = 0
NO
I = 1?
NO
YES
EA_W (EA)
X 1
YES
EA_W = EA_W + (X)
NO
DONE


A
EA_W = 0|D
X = 1
YES
EA_W = EA_W + (X)
NO
D < '400
NO
EA_S = LB_S
EA_W = EA_W + LB_W
EA_R = EA_R V LB_R
R = FALSE
DONE
YES
EA_W < M
YES
DONE
NO
EA_S = SB_S
EA_W = EA_W + SB_W
EA_R = EA_R V SB_R
R = FALSE
DONE
M WHEN SEG ENABLED = '10
M WHEN SEG DISABLED = '40


B
EA_S BR_S
EA_W BR_W + A
EA_R BR_R V EA_R
R FALSE
IXY 001
IXY 100 ?
NO
IXY 010
IXY 110 ?
NO
YES
YES
EA_W EA_W + (Y)
EA_W EA_W + (X)
YES
IXY 000 V 001 V 010
NO
[EA] bit 1
set
?
YES
NO
EA_S [EA] bits 5 16
EA_W [EA + 1]
EA_R EA_R V
[EA] bits 2 3
POINTER
FAULT
IXY 101?
NO
IXY 111
NO
YES
YES
EA_W EA_W + (Y)
EA_W EA_W + (X)
DONE

Credits.

Research and copy
Rosemary Shields

Design and production
William I Agush

Typesetting
JL Associates

Printing and binding
MARK BURTON

FDR3340