BASIC AND EXTENDED 3A PROCESSOR INSTRUCTION SET

3A PROCESSOR COMMON SYSTEMS

CONTENTS PAGE

1. GENERAL . . . 3

2. DESCRIPTION OF BASIC 3A INSTRUCTION SET . . . 4

ADDRESSING SCHEME . . . 4

EXECUTION TIMES FOR INSTRUCTIONS . . . 5

TYPES OF INSTRUCTIONS . . . 5

GENERAL INSTRUCTION FORMAT . . . 5

CLASSES OF INSTRUCTIONS . . . 8

A. Data Transfer Instructions . . . 8

Memory-to-Register . . . 9

Register-to-Memory . . . 13

Register-to-Register . . . 17

B. Branch Instructions . . . 19

Unconditional Short Transfer . . . 20

Unconditional Long Transfer . . . 21

Conditional Short Transfer . . . 21

Conditional Long Transfer . . . 22

Relative Transfer . . . 24

Subroutine . . . 25

C. Arithmetic Instructions . . . 29

Addition . . . 30

Subtraction . . . 32

NOTICE
Not for use or disclosure outside the
Bell System except under written agreement

Printed in U.S.A.

CONTENTS PAGE

D. Logic Instructions 33

Boolean Functions 33

Comparison 37

Rotation 39

Zeros Test 41

E. Bit Operational Instructions 41

Setting of a Bit 42

Clearing of a Bit 43

Testing of a Bit 46

F. Input/Ouput Instructions 51

Control 51

Test 53

G. Maintenance Instructions—Used to Control the System While in Maintenance Programs . . 54

H. Miscellaneous 64

3. GENERAL PURPOSE PHASE 1 EXTENSIONS TO THE 3A PROCESSOR INSTRUCTION SET 67

DESCRIPTION OF THE GENERAL PURPOSE EXTENSIONS TO THE BASIC 3A INSTRUCTION SET . . 67

A. Data Transfer Instructions 68

B. Branch Instructions 90

C. Arithmetic Instructions 94

D. Bit Operational Instructions 99

E. Input/Output Instructions 102

F. Direct Memory Access (DMA) Instructions 114

G. Communications Instructions 121

H. Miscellaneous 156

4. GLOSSARY 157

Figures **PAGE**

1. Three Types of Instructions in the Basic and Extended 3A Instruction Set 5

2. General Format of the Instruction Set 7

Tables

A. Definitions of Terms Used in Description of Basic Instructions 159

B. Definitions of Terms Used in Description of Extended Instructions 160

C. Alphabetic Listing of Basic Instructions 161

D. Alphabetic Listing of Extended Instructions 163

E. Alphabetic Listing of Basic and Extended Instructions 165

F. Definition of Addressable Special Registers 169

APPENDIXES 1 THROUGH 5

1. GENERAL

1.01 This section describes the basic and extended processor instruction set used in the 3A Central Control (3A CC). The following types of information are contained in this section:

- The format, function, options and requirements, and the approximate execution times of each instruction.

- The operation (OP) code for each instruction expressed in binary, octal, and hexadecimal representation.

Note: For the purpose of this section, when machine language is expressed in:

(a) Binary notation, the number is subscripted with a 2 (0110001_2)

(b) Octal notation, the number is subscripted with an 8 (061_8)

(c) Hexadecimal notation, the number is subscripted with a 16 (31_{16}).

- Table A defines common abbreviations and terms used in the description of the basic instructions.

- Table B defines common abbreviations and terms used in the description of the extended instructions.

- The mnemonic codes of the basic instructions are listed alphabetically in Table C with references to the descriptive paragraph on each instruction.

- The mnemonic codes of the extended instructions are listed alphabetically in Table D with references to the descriptive paragraph on each instruction.

- Table E is a combined alphabetical listing of the mnemonic codes of basic and extended instructions listed separately in Tables C and D, respectively.

- Table F defines those registers which are accessed by special register instructions.
- The appendixes contain miscellaneous information vital to the use of the instructions contained herein.

Additional information on the instruction set and 3A coding and conventions can be found in the following system software sections:

Section 254-340-100—Introduction to 3A Language

Section 254-340-104—Program Listing Organization and Usage.

1.02 ♦This section is being reissued to include LALLI, STALLI, LDX, LDK, STDK, GETFR, and RELFR in the 3A Processor Instruction Set. Revision arrows have been used to denote significant changes.♦

1.03 A glossary of abbreviations, acronyms, terms, and definitions is given in Part 4 of this section.

2. DESCRIPTION OF BASIC 3A INSTRUCTION SET

ADDRESSING SCHEME

2.01 The 3A CC can access over a million words of memory via a 20-bit addressing arrangement. This 20-bit address is normally generated by incrementing the program address (PA) register by one on each memory access. However, any 20-bit address can be generated or obtained by any one of the following methods:

(a) The 20-bit address may be contained within a double word (two 16-bit words) instruction. In this case, bits 3 through 0 of the first word of the instruction correspond to bits 19 through 16 of the address and bits 15 through 0 of the second word of the instruction correspond to bits 15 through 0 of the address. The addressing range is any location within 1,048,576 words.

(b) The 20-bit address may be generated by adding a 4-bit value (N) or a 12-bit value (K) to a general register pair (RP). The address base is continued in any RP from 0 to 14; (RP must be even).

(c) The 20-bit address may be generated by adding the contents of one general register to RA. The addressing range is any location from RA to (RA + 65,535).

(d) The 20-bit address may be generated by adding or subtracting an 8-bit number (OFFSET) to or from the contents of PA. The addressing range is any location from PA to (PA ±255).

(e) The 20-bit address may be generated by adding the contents of one of the general registers to PA + 1. The addressing range is any location from (PA + 1) to (PA + 65,535).

(f) The 20-bit address may be generated by combining eight bits within the instruction for bits 7 through 0 of the address and 12 bits stored in the microprogram store for bits 8 through 19 of the address. The addressing range is any location within a 255-word boundary depending on the constant in microprogram store.

2.02 The basic instruction set is stored in the main store and each instruction is fetched via the main store bus when needed. Within each instruction is a 7-bit operation (OP) code (paragraph 2.09). This 7-bit OP code points the microprogram control of the 3A CC to a starting address for a microsequence which performs the desired function of the instruction.

(b) RN—Register and immediate operand

(c) RxR—References memory by adding an index register to an address register pair

(d) RxN—References memory by adding N or K to an address register pair

(e) CM—Communications instructions

(f) RI—Register and immediate data

(g) MM—Memory-to-memory

(h) SL—Specified 20-bit data to load a register pair of reference memory

(i) SB—Subroutine instructions

(j) SS—Specified 8-bit offset in branch operation

(k) MS—Miscellaneous instructions.

Each instruction format contains two parity bits, one branch allowed (BA) bit, and a 7-bit OP code.

2.07 All words in the system have odd parity over eight bits and two parity bits over each 16-bit word. The two parity bits of each word are parity (P_L) on the low eight bits (0 through 7) and parity (P_H) on the high eight bits (8 through 15).

2.08 The branch allowed (BA) bit is used in the program transfer or branch process. When a branch instruction occurs, a hardware check ensures that the BA bit of the instruction being branched to is set. If for some reason the BA bit is not set, an error is indicated in the error register of the 3A CC. The BA bit appears in bit 15 of the first word of an instruction.

2.09 The 7-bit OP code field in all instructions specifies the function to be performed and is used to access a set of microinstructions which accomplishes the function indicated by the OP code. The OP code is located in bits 8 through 14 of the first word of each instruction. The OP code in this document is shown as seven binary digits, its octal equivalent, and its hexadecimal representation. In other words, an OP code of 00110001_2 is expressed as 061_8 in octal and 31_{16} in hexadecimal.

2.10 The remaining bits of single word, double word, and triple word instructions contain different arrangements and types of information depending on the particular instruction. This information may include registers, immediate data, addresses, etc. Registers are specified in decimal. Address and data fields may be specified using octal or hexadecimal qualifiers—the default case being decimal. See the explanations for each instruction to determine the type of information given and the formats of the instructions.

2.11 The register address pair (RA) is the general register address pair of either R12 and R13 or R14 and R15 used to contain a 20-bit address (bits 3 through 0 in R12 or R14 and bits 15 through 0 in R13 or R15).

EXECUTION TIMES FOR INSTRUCTIONS

2.03 The execution time of an instruction always includes the store fetch of the next instruction. The execution time given for each instruction depends on the following items:

(a) Microcycle time (execution time of a microinstruction—150 nanoseconds)

(b) Effective store access time (time from the beginning of the microcycle that initiates a store fetch to the end of the microcycle during which the ***data ready*** bit is set—1.05 microseconds)

(c) Number of microinstructions required in the microsequence for the instruction

(d) Amount of overlap possible between store operations and the microsequence for the instruction.

TYPES OF INSTRUCTIONS

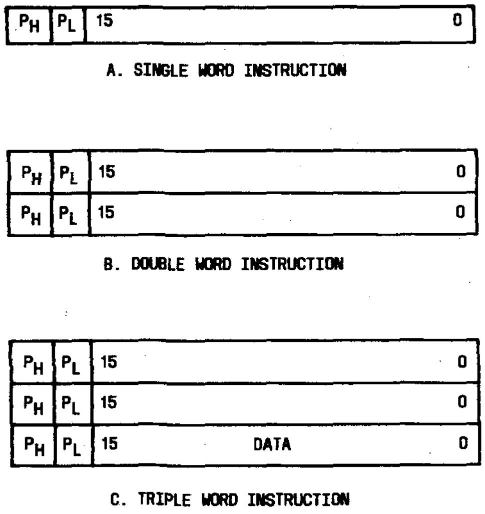
2.04 The three types of instructions in the 3A basic and extended instruction set are the single (full) word instructions, the double word instructions, and the triple word instructions (Fig. 1). These combinations of instructions provide a powerful instruction set. Single word instructions are the most commonly used within the system. Double word instructions are often used because either 16 bits of data or a 20-bit address is required in an instruction. Triple word instructions are used in multiple register manipulations.

GENERAL INSTRUCTION FORMAT

2.05 The 3A instructions are general purpose in nature to enable reading from or writing into any of the general registers. Since most of the instructions allow any general register to be used, it is not necessary to move the data to a special register to perform a function.

2.06 The general formats (Fig. 2) for the instructions in the 3A CC systems are:

(a) RR—Register-to-register



A. SINGLE WORD INSTRUCTION

B. DOUBLE WORD INSTRUCTION

C. TRIPLE WORD INSTRUCTION

Fig. 1—Three Types of Instructions in the Basic and Extended 3A Instruction Set

INCOMPLETE [illegible]
BEST COPY AVAILABLE

RR- REGISTER TO REGISTER

P_H	P_L	BA	OP CODE	R_X	R_Y
1	1	1	7	4	4

RN- REGISTER AND IMMEDIATE OPERAND

P_H	P_L	BA	OP CODE	R_X	N
1	1	1	7	4	4

RXR- REFERENCES MEMORY BY ADDING AN INDEX REGISTER TO AN ADDRESS REGISTER PAIR

					INDEX
P_H	P_L	BA	OP CODE	R_X	R_Y
1	1	1	7	4	4

RXN- REFERENCES MEMORY BY ADDING 4-BIT INDEX (N) TO AN ADDRESS REGISTER

					INDEX
P_H	P_L	BA	OP CODE	R_X/OPT	N
1	1	1	7	4	4

CM- COMMUNICATIONS

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	RP_X	OPT
P_H	P_L	K			R_X
1	1	12			4

RI- REGISTER AND IMMEDIATE DATA

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	R_X	OPT
P_H	P_L	I			
1	1	16			

MM- MEMORY TO MEMORY

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	0	0
P_H	P_L	0		RP_X	RP_X
1	1	8		4	4

SL- SPECIFIES 20-BIT DATA TO LOAD A REGISTER PAIR OF REFERENCE MEMORY

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	R_X	L(19-16)
P_H	P_L	L(15-0)			
1	1	16			

SB- SUBROUTINE (ONE WORD)

P_H	P_L	BA	OP CODE	OPT	OPT
1	1	1	7	4	4

SB- SUBROUTINE (TWO WORD)

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	OPT	OPT
P_H	P_L	OPT			
1	1	16			

SS- SPECIFIES 8-BIT OFFSET IN BRANCH OPERATION

P_H	P_L	BA	OP CODE	OFFSET
1	1	1	7	8

MS- MISCELLANEOUS (ONE WORD)

P_H	P_L	BA	OP CODE	OPT	OPT
1	1	1	7	4	4

MS- MISCELLANEOUS (TWO WORD)

1	1	1	7	4	4
P_H	P_L	BA	OP CODE	OPT	OPT
P_H	P_L	OPT			
1	1	16			

Fig. 2—General Format of the Instruction Set

CLASSES OF INSTRUCTIONS

2.12 The instructions used in the 3A CC are broken down into classes according to functions performed. These classes are:

(a) Data transfer instructions

(b) Branch instructions

(c) Arithmetic instructions

(d) Logic instructions

(e) Bit operation instructions

(f) Input/output instructions

(g) Maintenance instructions.

The instructions are described within the class to which they apply. Within each class, both types of instructions, single word or double word, are used.

A. Data Transfer Instructions

2.13 The data transfer class of instructions controls the exchange of information within the system. These instructions are divided into the following subclasses:

(a) ***Memory-to-Register***—This subclass involves the transfer of data from a memory location to a 3A CC register.

(b) ***Register-to-Memory***—This subclass involves the transfer of data from a 3A CC register to a memory location.

(c) ***Register-to-Register***—This subclass involves a transfer of one of the 3A CC registers to any one of the other 3A registers.

(d) ***Memory-to-Memory***—This subclass involves a transfer of data from one memory location to another memory location.

Memory-to-Register

2.14 *LL Rx,Y*

Format:

P_H	P_L	BA	14 OP CODE 0110001_2 061_8 31_{16} 8	7 R_X 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This instruction loads a general register (Rx) with the contents of a memory location Y. The contents of memory location Y remain unchanged.

Approximate execution time: 3.75 microseconds.

2.15 *LAL Rx,Y,RA*

Format:

P_H	P_L	BA	14 OP CODE $0110010_2/0110011_2$ $062_8/063_8$ $32_{16}/33_{16}$ 8	7 R_X 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This instruction loads the contents of a memory location (Y) into a general register (Rx) and loads the address of location Y into RA. Memory at location Y is unchanged. Bits 15 through 4 of general register 12/14 are not changed by this instruction.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 32_{16} (062_8) is used. When RA = R14, bit 8 is 1 and OP CODE 33_{16} (063_8) is used. RX should not be equal to either member of the RA register pair.

Approximate execution time: 3.90 microseconds.

2.16 *L Rx,N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1000000_2/1000001_2$ $100_8/101_8$ $40_{16}/41_{16}$	7 R_X 4	3 N 0

Function: This instruction loads a general register (Rx) with the contents of a memory location which is determined by adding a number, N(0 to 15), to the contents of RA. RA is loaded with the address of the memory location.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 40_{16} (100_8) is used. When RA = R14, bit 8 is 1 and OP CODE 41_{16} (101_8) is used.

Approximate execution time: 2.55 microseconds.

2.17 *LA Rx,N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1000010_2/1000011_2$ $102_8/103_8$ $42_{16}/43_{16}$	7 R_X 4	3 N 0

Function: This instruction loads a general register (Rx) with the contents of a memory location which is determined by adding N to the contents of RA and then loads the new address back into RA. Bits 15 through 4 of general register 12/14 are not changed by this instruction.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 42_{16} (102_8) is used. When RA = R14, bit 8 is 1 and OP CODE 43_{16} (103_8) is used. RX should not be equal to either member of the RA register pair.

Approximate execution time: 2.55 microseconds.

2.18 *LX Rx,Ry(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_x 4	3 R_y 0
			$1000100_2/1000101_2$ $104_8/105_8$ $44_{16}/45_{16}$		

Function: This instruction loads a general register (Rx) with the contents of a memory location which is determined by adding the contents of a general register (Ry) to RA. (Ry can be the same register as Rx).

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 44_{16} (104_8) is used. When RA = R14, bit 8 is 1 and OP CODE 45_{16} (105_8) is used.

Approximate execution time: 2.55 microseconds.

2.19 *LAX Rx,Ry(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_x 4	3 R_y 0
			$1000110_2/1000111_2$ $106_8/107_8$ $46_{16}/47_{16}$		

Function: This instruction loads a general register (Rx) with the contents of a memory location, which is determined by adding the contents of a general register (Ry) to RA, and then updating RA (loading the address of memory location into the RA). Bits 15 through 4 of the general register 12/14 are not changed by this instruction. (Ry can be the same register as Rx).

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 46_{16} (106_8) is used. When RA = R14, bit 8 is 1 and OP CODE 47_{16} (107_8) is used. Rx should not be equal to either member of the RA register pair.

Approximate execution time: 2.55 microseconds.

2.20 *LI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 00001111_2 007_8 07_{16} 8	7 R_X 4	3 0001_2 01_8 1_{16} 0
P_H	P_L	15	I		0

Function: This instruction loads a general register (Rx) with immediate data (I).

Options and requirements: Since many other instructions use OP CODE 07_{16} (007_8), the LI option is indicated by the 1_{16} (01_8) in bits 0 through 3 of the first word.

Approximate execution time: 2.40 microseconds.

2.21 *LN Rx,N*

Format:

P_H	P_L	BA	14 OP CODE 8 0000110_2 006_8 06_{16}	7 R_X 4	3 N 0

Function: This instruction loads a general register (Rx) with 4 bits of immediate data, N. The remaining bits (15 through 4) of the general register are cleared.

Approximate execution time: 1.20 microseconds.

Register-to-Memory

2.22 *STL Rx,Y*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td>14 OP CODE
0111001_2
071_8
39_{16} 8</td><td>7 R_X 4</td><td>3 BITS (19-16) OF Y 0</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="4">15 BITS (15-0) OF Y 0</td></tr>
</table>

Function: This instruction stores the contents of a general register (Rx) into a memory location Y. The contents within Rx remain unchanged.

Approximate execution time: 3.75 microseconds.

2.23 *STAL Rx,Y(RA)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td>14 OP CODE
$0111010_2/0111011_2$
$072_8/073_8$
$3A_{16}/3B_{16}$ 8</td><td>7 R_X 4</td><td>3 BITS (19-16) OF Y 0</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="4">15 BITS (15-0) OF Y 0</td></tr>
</table>

Function: This instruction stores the contents of a general register (Rx) into a memory location Y, and loads the address of location Y into RA. The contents within Rx remain unchanged. Bits 15 through 4 of general register 12/14 are not changed by this instruction.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE $3A_{16}$ (072_8) is used. When RA = R14, bit 8 is 1 and OP CODE $3B_{16}$ (073_8) is used. Rx should not be equal to either member of the RA register pair.

Approximate execution time: 4.05 microseconds.

2.24 *ST Rx,N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1001000_2/1001001_2$ $110_8/111_8$ $48_{16}/49_{16}$	7 R_X 4	3 N 0

Function: This instruction stores the contents of a general register (Rx) into a memory location which is determined by adding a number, N (0 to 15), to the contents of RA. The contents within the general register (Rx) remain unchanged.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 48_{16} (110_8) is used. When RA = R14, bit 8 is 1 and OP CODE 49_{16} (111_8) is used.

Approximate execution time: 2.70 microseconds.

2.25 *STA Rx,N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1001010_2/1001011_2$ $112_8/113_8$ $4A_{16}/4B_{16}$	7 R_X 4	3 N 0

Function: This instruction stores the contents of a general register (Rx) into a memory location, which is determined by adding N (0 to 15) to the contents of RA, and loads the address of this memory location into RA. The contents within the general register remain unchanged. Bits 15 through 4 of general register 12/14 are not changed by this instruction.

Options and requirement: When RA = R12, bit 8 is 0 and OP CODE $4A_{16}$ (112_8) is used. When RA = R14, bit 8 is 1 and OP CODE $4B_{16}$ (113_8) is used. Rx should not be equal to either member of the RA register pair.

Approximate execution time: 2.70 microseconds.

2.26 *STX Rx,Ry(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1001100_2/1001101_2$ $114_8/115_8$ $4C_{16}/4D_{16}$	7 R_x 4	3 R_y 0

Function: This instruction stores the contents of a general register (Rx) in a memory location which is determined by adding general register (Ry) to RA. The contents within the general register (Rx) from which memory was loaded remain unchanged.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE $4C_{16}$ (114_8) is used. When RA = R14, bit 8 is 1 and OP CODE $4D_{16}$ (115_8) is used.

Approximate execution time: 2.70 microseconds.

2.27 *STAX Rx,Ry(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1001110_2/1001111_2$ $116_8/117_8$ $4E_{16}/4F_{16}$	7 R_x 4	3 R_y 0

Function: This instruction stores the contents of a general register (Rx) in a memory location, which is determined by adding general register (Ry) to RA, and loads the address of the memory location into RA. The contents within the general register from which memory was loaded remain unchanged. Bits 15 through 4 of general register 12/14 are not changed by this instruction.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE $4E_{16}$ (116_8) is used. When RA = R14, bit 8 is 1 and OP CODE $4F_{16}$ (117_8) is used. Rx should not be equal to either member of the RA register pair.

Approximate execution time: 2.70 microseconds.

2.28 *STM Rx,N(RA),M*

Format:

P_H	P_L	BA	14 OP CODE 8 $0000100_2/0000101_2$ $004_8/005_8$ $04_{16}/05_{16}$	7 R_X 4	3 N 0
P_H	P_L	15 MASK 0			

Function: This instruction stores under mask the contents of a general register (Rx) into memory location Y, which is determined by adding a 4-bit number (N) to RA. This operation is accomplished via insertion masking. The new information obtained from the masking operation is loaded into R0 before it is stored into memory location Y. Therefore, the original information in R0 is destroyed.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 04_{16} (004_8) is used. When RA = R14, bit 8 is 1 and OP CODE 05_{16} (005_8) is used.

Approximate execution time: 5.40 microseconds.

2.29 *STVM Rx,N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 $1100110_2/1100111_2$ $146_8/147_8$ $66_{16}/67_{16}$	7 R_X 4	3 N 0

Function: This instruction stores under variable mask the contents of a general register (Rx) into memory location Y, which is determined by adding a 4-bit number (N) to RA. The variable mask is obtained from R0. This operation is accomplished via insertion masking. The new information obtained from the masking operation is loaded into R0 before it is stored into memory location Y. Therefore, the original information in R0 is destroyed.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 66_{16} (146_8) is used. When RA = R14, bit 8 is 1 and OP CODE 67_{16} (147_8) is used.

Approximate execution time: 5.55 microseconds.

Register-to-Register

2.30 *LRS Rx,RS*

Format:

P_H	P_L	BA	14 OP CODE 8 01000010_2 042_8 22_{16}	7 R_X 4	3 R_S 0

Function: This instruction loads a general register (Rx) with the contents of a special register (RS). RS is one of the 16-bit special registers (Table A).

Approximate execution time: 1.20 microseconds.

2.31 *LSR RS,Rx*

Format:

P_H	P_L	BA	14 OP CODE 8 00011111_2 017_8 $0F_{16}$	7 R_S 4	3 R_X 0

Function: This instruction loads a special register (RS) with the contents of one of the general registers (Rx). The contents of the general register remain unchanged. RS is one of the 16-bit special registers (Table A).

Approximate execution time: 1.20 microseconds.

2.32 *LR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8 00011100_2 014_8 $0C_{16}$	7 R_X 4	3 R_y 0

Function: This instruction loads one general register (Rx) with the contents of another general register (Ry). The contents of the general register from which the data was loaded (Ry) remain unchanged.

Approximate execution time: 1.20 microseconds.

2.33 *LRM Rx,Ry,M*

Format:

P_H	P_L	BA	14 OP CODE 8 00111101_2 035_8 $1D_{16}$	7 R_x 4	3 R_y 0
P_H	P_L	15 MASK 0			

Function: This instruction loads the logical AND of Ry and the mask into Rx. Ry is unchanged.

Approximate execution time: 2.40 microseconds.

2.34 *PACK RS*

Format:

P_H	P_L	BA	14 OP CODE 8 0010100_2 024_8 14_{16}	7 R_S 4	3 0000_2 0 00_8 0_{16}

Function: This instruction loads a 20-bit special register (RS) with 20 bits of data from R2 and R3. R2 bits (3 through 0) are gated to RS bits (19 through 16). R3 bits (15 through 0) are gated to RS bits (15 through 0).

Approximate execution time: 1.20 microseconds.

2.35 *EXR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8 0001110_2 016_8 $0E_{16}$	7 R_x 4	3 R_y 0

Function: This instruction exchanges the contents of a general register (Rx) with the contents of another general register (Ry).

Approximate execution time: 1.125 microseconds.

2.36 *UNPK RS*

Format:

P_H	P_L	BA	14 OP CODE 8 0010101_2 025_8 15_{16}	7 0000_2 4 00_8 0_{16}	3 R_S 0

Function: This instruction unloads 20 bits of data from a special register (RS) into R2 and R3. Bits 19 through 16 of RS are loaded into bits 3 through 0 of R2 and bits 15 through 0 of RS are loaded into bits 15 through 0 of R3. Bits 15 through 4 of R2 are cleared by this instruction.

Approximate execution time: 1.20 microseconds.

2.37 *IRM Rx,Ry,M*

Format:

P_H	P_L	BA	14 OP CODE 8 0011100_2 034_8 $1C_{16}$	7 R_X 4	3 R_y 0
P_H	P_L	15 MASK 0			

Function: This instruction inserts Ry into Rx on a bit-by-bit basis. If a mask bit = 1, then the logical AND of that equivalent bit in Ry with that mask bit is inserted into that equivalent bit of Rx. If a mask bit = 0, then the corresponding bit in Rx is unaffected. Ry is unchanged.

Approximate execution time: 2.40 microseconds.

B. Branch Instructions

2.38 The branch or transfer class of instructions controls the flow of program in the system. These instructions are divided into the following subclasses:

(a) Unconditional short transfer—Branching to a new address that is obtained by adding or subtracting an 8- or 4-bit number to or from an address.

(b) Unconditional long transfer—Branching to a new 20-bit address that is contained within the branch instruction.

(c) Conditional short transfer—If a certain situation or condition exists, branching to a new address which is obtained by adding or subtracting an 8-bit number to or from an address.

(d) Conditional long transfer—If a certain situation or condition exists, branching to a new 20-bit address which is contained within the branch instruction.

(e) Relative transfer—Branching to a new address that is obtained by adding a 16-bit register to an address.

(f) Subroutine—Branching to an instruction sequence that performs a specific function and then returns to the main routine.

Unconditional Short Transfer

2.39 *B Y*

Format:

P_H	P_L	BA	14 OP CODE 8	7 OFFSET 0
			$1010110_2/1010111_2$ $126_8/127_8$ $56_{16}/57_{16}$	

Function: This instruction causes a transfer to memory location Y which is determined by adding a number (offset) to the contents of the program address (PA) register.

Options and requirements: If bit 8 is 0, the number of offset is positive and causes a forward transfer. The forward transfer is associated with OP CODE 56_{16} (126_8). If bit 8 is 1, the offset is negative and causes a backward transfer. The backward transfer is associated with OP CODE 57_{16} (127_8).

Approximate execution time: 1.80 microseconds.

2.40 *BR N(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8	7 $0000_2/0001_2$ 4	3 N 0
			1010101_2 125_8 55_{16}	$00_8/01_8$ $0_{16}/1_{16}$	

Function: This instruction causes a transfer to memory location Y which is determined by adding a number (N) to RA. This instruction only causes a forward transfer.

Options and requirements: When RA = R12, the 0_{16} (00_8) option is used in bits 7 through 4. When RA = R14, the 1_{16} (01_8) option is used in bits 7 through 4.

Approximate execution time: 2.70 microseconds.

Unconditional Long Transfer

2.41 ***BL Y***

Format:

P_H	P_L	BA	14 OP CODE 0111110_2 076_8 $3E_{16}$ 8	7 0000_2 00_8 0_{16} 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This instruction causes a transfer or branch long to memory location Y.

Options and requirements: Since other instructions use OP CODE $3E_{16}$ (076_8) the BL option is indicated by the 0_{16} (00_8) in bits 7 through 4 of the first word.

Approximate execution time: 2.70 microseconds.

Conditional Short Transfer

2.42 ***BC Y***

Format:

P_H	P_L	BA	14 OP CODE 8 $1011000_2/1011001_2$ $130_8/131_8$ $58_{16}/59_{16}$	7 OFFSET 0

Function: This is a branch ***on condition*** instruction. In other words, when the condition flip-flop (CF) is set to 1, this instruction causes a transfer to a memory location which is determined by adding a number (offset) to the contents of the PA register. If the CF is equal to 0, no branch or transfer occurs and the next sequential instruction is executed.

Options and requirements: If bit 8 is 0, the offset is positive and causes a forward transfer. The forward transfer is associated with OP CODE 58_{16} (130_8). If bit 8 is 1, the offset is negative and causes a backward transfer. The backward transfer is associated with OP CODE 59_{16} (131_8).

Approximate execution time: 1.95 microseconds for CF = 1; 1.50 microseconds for CF = 0.

2.43 *BNC Y*

Format:

P_H	P_L	BA	14 OP CODE 8 $1011010_2/1011011_2$ $132_8/133_8$ $5A_{16}/5B_{16}$	7 OFFSET 0

Function: This instruction is a branch ***on not condition.*** When the CF is not set (CF = 0), this instruction causes a transfer to a memory location which is determined by adding a number (offset) to the contents of the PA register. When the CF is set (CF = 1), no branch occurs and the next sequential instruction is executed.

Options and requirements: If bit 8 is 0, the offset is positive and causes a forward transfer. The forward transfer is associated with OP CODE $5A_{16}$ (132_8). If bit 8 is 1, the offset is negative and causes a backward transfer. The backward transfer is associated with OP CODE $5B_{16}$ (133_8).

Approximate execution time: 1.50 microseconds for CF = 1; 1.95 microseconds for CF = 0.

Conditional Long Transfer

2.44 *BCL Y*

Format:

P_H	P_L	BA	14 OP CODE 8 1010000_2 120_8 50_{16}	7 0000_2 4 00_8 0_{16}	3 BITS (19-16) 0 OF Y
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This is a branch ***on condition*** instruction. When the condition flip-flop (CF) is set, this instruction causes a transfer or branch long to memory location Y. If the CF is not set, no transfer occurs and the next sequential instruction is executed.

Approximate execution time: 2.70 microseconds for CF = 1; 1.80 microseconds for CF = 0.

2.45 *BNCL Y*

Format:

P_H	P_L	BA	14 OP CODE 1010001_2 121_8 51_{16} 8	7 0000_2 00_8 0_{16} 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This instruction is a branch ***on not condition.*** When the CF is not set (CF = 0), this instruction causes a transfer or branch long to memory location Y. When the CF is set (CF = 1), no branch occurs and the next sequential instruction is executed.

Approximate execution time: 1.80 microseconds for CF = 1; 2.70 microseconds for CF = 0.

2.46 *BX Rx,Y*

Format:

P_H	P_L	BA	14 OP CODE 0111100_2 074_8 $3C_{16}$ 8	7 R_X 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: When Rx is not equal to zero, this instruction causes Rx to be decremented by one and a transfer to memory location Y to occur. If Rx is equal to zero, no transfer occurs and the next sequential instruction is executed.

Approximate execution time: 2.55 microseconds for Rx = 0; 2.70 microseconds for Rx ≠ 0.

Relative Transfer

2.47 *BRX Rx(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 1010101_2 125_8 55_{16}	7 $0010_2/0011_2$ 4 $02_8/03_8$ $2_{16}/3_{16}$	3 R_X 0

Function: This instruction causes a transfer to memory location Y which is determined by adding the contents of a general register (Rx) to RA. This instruction only causes forward transfers.

Options and requirements: When RA = R12, the 2_{16} (02_8) option is used in bits 7 through 4. When RA = R14, the 3_{16} (03_8) option is used in bits 7 through 4.

Approximate execution time: 2.55 microseconds.

2.48 *BPAX Rx*

Format:

P_H	P_L	BA	14 OP CODE 8 1011100_2 134_8 $5C_{16}$	7 0000_2 4 00_8 0_{16}	3 R_X 0

Function: This instruction causes a transfer to memory location Y which is determined by adding a general register (Rx) to the PA + 1. This instruction only causes forward transfers.

Approximate execution time: 1.65 microseconds.

Subroutine

2.49 *BSA Y*

Format:

P_H	P_L	BA	14 OP CODE 0111110_2 076_8 $3E_{16}$ 8	7 0001_2 01_8 1_{16} 4	3 BITS (19-16) OF Y 0
P_H	P_L	15 BITS (15-0) OF Y 0			

Function: This instruction causes the hold-get (HG) register to be decremented by 16; the return address to be stored in words 0 and 1 of the HG area; and a transfer to memory location Y to occur. Bits 4 through 15 of HG area word 0 are cleared.

Options and requirements: Since other instructions use OP CODE $3E_{16}$ (076_8), the BSA option is indicated by the 1_{16} (01_8) in bits 7 through 4 of the first word.

Approximate execution time: 6.15 microseconds.

2.50 *BSAI X*

Format:

P_H	P_L	BA	14 OP CODE 1110110_2 166_8 76_{16} 8	7 X 0

Function: This instruction causes the HG register to be decremented by 16; the return address to be stored in words 0 and 1 of the HG area; and a transfer to a new address to occur. The address is determined by an 8-bit field (X-low eight bits) in the instruction and 12 bits (12 high bits) in the *microprogram* store. Bits 4 through 15 of the HG area word 0 are cleared.

Approximate execution time: 5.70 microseconds.

2.51 *BTSA*

Format:

P_H	P_L	BA	14 OP CODE 8 1011101_2 135_8 $5D_{16}$	7 0000_2 4 00_8 0_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction sets the CF to 1 if register 0 is zero, and sets CF to 0 if register 0 is nonzero. The OP CODE FIL bit is set equal to bit 15 in word 0 of the HG area. It branches to the return address stored in words 0 and 1 of the HG area. The HG counter is incremented by 16.

Options and requirements: Since other instructions use OP CODE $5D_{16}$ (135_8), the BTSA option is indicated by the 0_{16} (00_8) in bits 7 through 4.

Approximate execution time: 4.95 microseconds.

2.52 *BTSAN N*

Format:

P_H	P_L	BA	14 OP CODE 8 1011101_2 135_8 $5D_{16}$	7 0011_2 4 03_8 3_{16}	3 N 0

Function: This instruction loads the return code N into bits (3 through 0) of register 0. Bits 15 through 4 of register 0 are set to zero. The CF is set to one if register 0 is zero and to zero if register 0 is nonzero. The OP CODE FIL bit is set equal to bit 15 in word 0 of the HG area. A branch is made to the return address which is stored in words 0 and 1 of the HG area. Increments HG counter by 16.

Options and requirements: Since other instructions use OP CODE $5D_{16}$ (135_8), the BTSAN option is indicated by the 3_{16} (03_8) in bits 7 through 4.

Approximate execution time: 5.10 microseconds.

2.53 *BTSAG*

Format:

P_H	P_L	BA	14 OP CODE 8 1011101_2 135_8 $5D_{16}$	7 0001_2 4 01_8 1_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction loads registers 2 through 15 from words 2 through 15 of the HG area. The CF is set to one if register 0 is zero, and the CF is set to zero if register 0 is nonzero. The OP CODE FIL bit is set equal to bit 15 in word 0 of the HG area. It causes a branch to the return address stored in words 0 and 1 of the HG area. It causes a branch to the return address stored in words 0 and 1 of the HG area. A 16 is added to the HG counter.

Options and requirements: Since other instructions use OP CODE $5D_{16}$ (135_8), the BTSAG instruction is indicated by the 1_{16} (01_8) in bits 7 through 4.

Approximate execution time: 31.20 microseconds.

2.54 *BTSAGN N*

Format:

P_H	P_L	BA	14 OP CODE 8 1011101_2 135_8 $5D_{16}$	7 0010_2 4 02_8 2_{16}	3 N 0

Function: This instruction loads registers 2 through 15 from words 2 through 15 of the HG area. Return code N is loaded into bits 3 through 0 of register 0. Bits 15 through 4 of register 0 are set to zero. The CF is set to one if register 0 is zero and to zero if register 0 is nonzero. The OP CODE FIL bit is set equal to bit 15 in word 0 of the HG area. A branch to the return address stored in words 0 and 1 of the HG area is made. A 16 is added to the HG counter.

Options and requirements: Since other instructions use OP CODE $5D_{16}$ (135_8), the BTSAGN instruction is indicated by the 2_{16} (02_8) in bits 7 through 4.

Approximate execution time: 31.35 microseconds.

2.55 *HA*

Format:

P_H	P_L	BA	14 OP CODE 8 1110011_2 163_8 73_{16}	7 0001_2 4 01_8 1_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction stores R2 through R15 in words 2 through 15 of the HG area. Words 0 and 1 are used to store the return address. The registers stored remain unchanged.

Options and requirements: Since other instructions use OP CODE 73_{16} (163_8) the HA instruction is indicated by the 1_{16} (01_8) in bits 7 through 4.

Approximate execution time: 28.55 microseconds.

2.56 *GA*

Format:

P_H	P_L	BA	14 OP CODE 8 1110011_2 163_8 73_{16}	7 0000_2 4 00_8 0_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction loads R2 through R15 from words 2 through 15 of the HG area.

Options and requirements: Since other instructions use OP CODE 73_{16} (163_8), the GA instruction is indicated by the 0_{16} (00_8) in bits 7 through 4.

Approximate execution time: 26.55 microseconds.

2.57 *HN Rx,N*

Format:

P_H	P_L	BA	14 OP CODE 8 1110101_2 165_8 75_{16}	7 R_X 4	3 N 0

Function: This instruction stores the contents of a general register (Rx) in word N of the HG area.

Approximate execution time: 2.70 microseconds.

2.58 *GN Rx,N*

Format:

P_H	P_L	BA	14 OP CODE 8 1110100_2 164_8 74_{16}	7 R_X 4	3 N 0

Function: This instruction loads the contents of word number N of the HG area into a general register (Rx).

Approximate execution time: 2.55 microseconds.

2.59 *PIE*

Format:

P_H	P_L	BA	14 OP CODE 8 1011101_2 135_8 $5D_{16}$	7 0100_2 4 04_8 4_{16}	3 0000_2 0 00_8 0_{16}

Function: The program interrupt end instruction clears the block interrupt bit in the system status register; restores the OP CODE FIL bit which was saved in bit 15 of word 0 in the HG area; transfers to the return address saved in words 0 and 1 of the HG area; and increments the HG counter by 16.

Options and requirements: Since other instructions use OP CODE $5D_{16}$ (135_8), the PIE instruction is indicated by the 4_{16} (04_8) in bits 7 through 4.

Approximate execution time: 4.20 microseconds.

C. Arithmetic Instructions

2.60 The arithmetic instructions either add or subtract the contents of one register, 16 bits of immediate data, or 4 bits of immediate data to or from another register. Therefore, the arithmetic instructions are divided into the following subclasses:

(a) Addition

(b) Subtraction.

Addition

2.61 *AI Rx,I*

Format:

P_H	P_L	BA	14	OP CODE 00001111_2 007_8 07_{16}	8	7	R_X	4	3	0000_2 00_8 0_{16}	0
P_H	P_L	15				I					0

Function: This instruction adds 16 bits of immediate data (I) to the contents of a general register (Rx) and stores the results in that general register. If an overflow or carry beyond bit 15 occurs, the CF is set to 1. If no carry or overflow occurs, the CF is cleared.

Options and requirements: Since other instructions use OP CODE 07_{16} (007_8), the AI option is indicated by the 0_{16} (00_8) in bits 3 through 0 of the first word.

Approximate execution time: 2.40 microseconds.

2.62 *AN Rx,N*

Format:

P_H	P_L	BA	14	OP CODE 0001001_2 011_8 09_{16}	8	7	R_X	4	3	N	0

Function: This instruction adds 4 bits of data (N) to the contents of a general register (Rx) and then stores the results in that general register. If a carry or overflow occurs, the CF is set to 1. If a carry or overflow does not occur, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.63 *AR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8 0000011_2 003_8 03_{16}	7 R_x 4	3 R_y 0

Function: This instruction adds the contents of one general register (Rʏ) to the contents of another general register (Rx) and stores the results in the register (Rx). Rʏ remains unchanged.

Approximate execution time: 1.20 microseconds.

2.64 *A1S Y*

Format:

P_H	P_L	BA	14 OP CODE 8 0111110_2 076_8 $3E_{16}$	7 0010_2 4 02_8 2_{16}	3 BITS (19-16) 0 OF Y
P_H	P_L	BITS (15-0) OF Y			

Function: This instruction adds ●ne to the contents of memory location Y and stores the result in memory location Y. If a carry or overflow occurs, the CF is set to 1. If a carry or overflow does not occur, the CF is cleared.

Options and requirements: Since other instructions use OP CODE $3E_{16}$ (076_8), the A1S option is defined by the 2_{16} (02_8) in bits 7 through 4 of the first word.

Approximate execution time: 5.40 microseconds.

Subtraction

2.65 *SN Rx,N*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_X 4	3 -N 0
			0001000_2 010_8 08_{16}		

Function: This instruction subtracts a number (N) from the contents of a general register (Rx) and stores the result in Rx. The subtraction is actually accomplished by adding the 2s complement in N to Rx. If a carry or overflow occurs, the CF is set to 1. If a carry does not occur, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.66 *SR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_X 4	3 R_y 0
			0001010_2 012_8 $0A_{16}$		

Function: This instruction subtracts the contents of one general register (Ry) from the contents of another general register (Rx) and stores the result in Rx. The subtraction is actually accomplished by adding the 2s complement of Ry to Rx. If a carry or overflow occurs, the CF is set to 1. If a carry or overflow does not occur, the CF is cleared. The contents of Ry remain unchanged.

Approximate execution time: 1.35 microseconds.

2.67 *SI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 00001111_2 007_8 07_{16} 8	7 R_x 4	3 0000_2 00_8 0_{16} 0
P_H	P_L	15		-I	0

Function: This instruction subtracts the immediate data (I) from the contents of a general register (Rx) and stores the result in Rx. The subtraction is actually accomplished by adding the 2s complement in I to Rx. If a carry or overflow occurs, the CF is set to 1. If a carry or overflow does not occur, the CF is cleared.

Approximate execution time: 2.40 microseconds.

D. Logic Instructions

2.68 The logic instructions are a powerful class of instructions divided into the following subclasses:

(a) Boolean functions of two variables—For example, the contents of one register (variable A) are ANDed to the contents of another register (variable B) to perform the Boolean function A-B.

(b) Rotation—Shifting the contents of a register left or right.

(c) Comparison—Matching of two variables.

(d) Zeros test—Testing for all zeros or locating the least significant bit of a word.

Boolean Functions

2.69 *COM Rx*

Format:

P_H	P_L	BA	14 OP CODE 8 00110111_2 033_8 $1B_{16}$	7 R_x 4	3 R_x 0

Function: This instruction complements the contents of a general register (Rx) and stores the results in the same general register. If the complement stored in Rx is equal to zero, the CF is set. If the complement stored in Rx is not equal to zero, the CF is cleared.

Approximate execution time: 1.125 microseconds.

2.70 *COM Rx,RY*

Format:

P_H	P_L	BA	14 OP CODE 8 0011011_2 033_8 $1B_{16}$	7 R_X 4	3 R_Y 0

Function: This instruction complements the contents of one general register (RY) and stores the results in another general register (Rx). If the complement stored in Rx is equal to zero, the CF is set. If the complement stored in Rx is not equal to zero, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.71 *NI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 8 0000111_2 007_8 07_{16}	7 R_X 4	3 0010_2 0 02_8 2_{16}
P_H	P_L	15		I	0

Function: This instruction ANDs 16 bits of immediate data (I) to a general register (Rx) and stores the results in the same general register. If the result is equal to zero, the CF is set to 1. If the result does not equal zero, the CF is cleared.

Options and requirements: Since OP CODE 07_{16} (007_8) is used by other instructions, the NI instruction is indicated by the 2_{16} (02_8) in bits 3 through 0 of the first word.

Approximate execution time: 2.40 microseconds.

2.72 *NR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 00110000_2 030_8 18_{16} 8	7 R_X 4	3 R_Y 0

Function: This instruction ANDs one general register (Ry) to another general register (Rx) and stores the result in Rx. The contents of the other general register (Ry) remain unchanged. If the results equal zero, the CF is set to 1. If the results do not equal zero, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.73 *OI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 0000111_2 007_8 07_{16} 8	7 R_X 4	3 0011_2 03_8 3_{16} 0
P_H	P_L	15 I 0			

Function: This instruction performs the logic operation of Inclusive OR between 16 bits of immediate data (I) and a general register (Rx). The result is stored in Rx. If the result is zero, the CF is set to 1. If the result does not equal zero, the CF is cleared.

Options and requirements: Since other instructions use OP CODE 07_{16} (007_8), the OI instruction is indicated by the 3_{16} (03_8) in bits 3 through 0 of the first word.

Approximate execution time: 2.40 microseconds.

2.74 *OR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8 00110011_2 031_8 19_{16}	7 R_x 4	3 R_y 0

Function: This instruction performs the logical operation of Inclusive OR between one general register (Rx) and another general register (Ry). The result is stored in Rx and the contents of Ry remain unchanged. If the results equal zero, the CF is set to 1. If the results do not equal zero, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.75 *XI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 8 00001111_2 007_8 07_{16}	7 R_x 4	3 0 0100_2 04_8 4_{16}
P_H	P_L	15 I 0			

Function: This instruction performs the logic operation of Exclusive OR between 16 bits of immediate data (I) and one of the general registers (Rx). The result is stored in Rx. If the result is equal to zero, the CF is set to 1. If the result does not equal zero, the CF is cleared.

Options and requirements: Since other instructions use OP CODE 07_{16} (007_8), the XI instruction is indicated by the 4_{16} (04_8) in bits 3 through 0 of the first word.

Approximate execution time: 2.40 microseconds.

2.76 *XR Rx,Ry*

Format:

P_H	P_L	BA	14	OP CODE 0011010_2 032_8 $1A_{16}$	8	7	R_x	4	3	R_y	0

Function: This instruction performs the logic operation of Exclusive OR between one general register (Rx) and another general register (Ry). The result is stored in Rx and the contents of Ry remain unchanged. If the result is equal to zero, the CF is set to 1. If the result does not equal zero, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.77 *ZR Rx*

Format:

P_H	P_L	BA	14	OP CODE 0000110_2 006_8 06_{16}	8	7	R_x	4	3	0000_2 00_8 0_{16}	0

Function: This instruction zeroes the contents of a general register (Rx). The CF is unchanged.

Approximate execution time: 1.20 microseconds.

Comparison

2.78 *CR,Rx,Ry*

Format:

P_H	P_L	BA	14	OP CODE 0100000_2 040_8 20_{16}	8	7	R_x	4	3	R_y	0

Function: This instruction compares the contents of one general register (Rx) to the contents of another general register (Ry) on a bit-by-bit basis. If all the bits match, the CF is set to 1. If one of the bit positions mismatch, the CF is cleared. The contents of both Rx and Ry remain unchanged.

Approximate execution time: 1.20 microseconds.

2.79 *CRM Rx,Ry,M*

Format:

P_H	P_L	BA	14 OP CODE 0011110_2 036_8 $1E_{16}$ 8	7 R_x 4	3 R_y 0
P_H	P_L	15	M		0

Function: This instruction compares the contents of one general register (Rx) to the contents of another general register (Ry) on a bit-by-bit basis for every bit in the mask that is one. If all the bits compared match, the CF is set to 1. If one or more of the compared bits mismatch, the CF is cleared. The contents of Rx and Ry are unchanged by this instruction.

Approximate execution time: 2.40 microseconds.

2.80 *CI Rx,I*

Format:

P_H	P_L	BA	14 OP CODE 0000111_2 007_8 07_{16} 8	7 R_x 4	3 0101_2 05_8 5_{16} 0
P_H	P_L	15	I		0

Function: This instruction compares the contents of one general register (Rx) to the immediate data (I) in the second word of this instruction on a bit-by-bit basis. If all the bits match, the CF is set to 1. If one of the bits mismatch, the CF is cleared. The contents of Rx remain unchanged.

Approximate execution time: 2.40 microseconds.

2.81 *CIRM Rx,I,N,M*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td>14 OP CODE 00111111_2 037_8 $1F_{16}$ 8</td><td>7 R_X 4</td><td>3 N 0</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="2">15 MASK 8</td><td colspan="2">7 I 0</td></tr>
</table>

Function: This instruction causes the contents of a general register (Rx) to be rotated right by N positions. After rotation, the low eight bits are compared to the immediate data (I) on a bit-for-bit basis for each bit in the mask that is one. If all the bits compared match, the CF is set to 1. If any one of the bits compared mismatch, the CF is cleared. The contents of Rx remain unchanged.

Approximate execution time: 2.40 microseconds.

Rotation

2.82 *RL Rx,Ry*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td>14 OP CODE 00100010_2 022_8 12_{16} 8</td><td>7 R_X 4</td><td>3 R_y 0</td></tr>
</table>

Function: This instruction rotates the contents of a general register (Rx) left by N bit positions. The amount of rotation is determined by the low four bits of another general register (Ry). The contents of Ry remain unchanged.

Approximate execution time: 2.70 microseconds.

2.83 *RR Rx,Ry*

Format:

P_H	P_L	BA	14	OP CODE 00100011_2 023_8 13_{16}	8	7	R_X	4	3	R_y	0

Function: This instruction rotates the contents of a general register (Rx) right by N bit positions. The amount of rotation is determined by the low four bits of another general register (Ry). The contents of Ry remain unchanged.

Approximate execution time: 1.50 microseconds.

2.84 *RRN Rx,N*

Format:

P_H	P_L	BA	14	OP CODE 00100001_2 021_8 11_{16}	8	7	R_X	4	3	N	0

Function: This instruction rotates the contents of a general register (Rx) right by N bit positions. The amount of rotation (N) is determined by bits 3 through 0 of this instruction.

Approximate execution time: 1.20 microseconds.

2.85 *RLN Rx,N*

Format:

P_H	P_L	BA	14	OP CODE 00100001_2 021_8 11_{16}	8	7	R_X	4	3	16-N	0

Function: This instruction rotates the contents of a general register (Rx) left by 16-N bit positions.

Approximate execution time: 1.20 microseconds.

Zeros Test

2.86 *TZ Rx*

Format:

PH	PL	BA	14 OP CODE 8 0011000$_2$ 030$_8$ 18$_{16}$	7 Rx 4	3 Rx 0

Function: This instruction tests a general register (Rx) for all zeros. If Rx is equal to zero, the CF is set. If Rx is not equal to zero, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.87 *FLZ Rx,Ry*

Format:

PH	PL	BA	14 OP CODE 8 0100001$_2$ 041$_8$ 21$_{16}$	7 Rx 4	3 Ry 0

Function: This instruction detects the least significant zero in one of the general registers (Rx). If a zero is found, its location is placed in bits 3 through 0 of another general register (Ry), the CF is set to one, and bits 15 through 4 of Ry are zeroed. The zero in Rx is set to 1. If a zero is not found, the CF is cleared and Ry is unchanged.

Approximate execution time: 1.95 microseconds for zero found; 1.125 microseconds for no zero found.

E. Bit Operational Instructions

2.88 The bit operational instructions involve the performing of a function on a single bit and are divided into the following subclasses:

(a) Set instructions

(b) Clear instructions

(c) Test instructions.

Setting of a Bit

2.89 *SCF*

Format:

P_H	P_L	BA	14 OP CODE 8	7 0000_2 4	3 0000_2 0
			0100000_2	00_8	00_8
			040_8	0_{16}	0_{16}
			20_{16}		

Function: This instruction sets the CF flip-flop to 1.

Approximate execution time: 1.20 microseconds.

2.90 *SOP*

Format:

P_H	P_L	BA	14 OP CODE 8	7 0000_2 4	3 0001_2 0
			0001101_2	00_8	01_8
			015_8	0_{16}	1_{16}
			$0D_{16}$		

Function: This instruction sets the OP CODE FIL bit. The setting of this bit provides an additional 128 OP codes for use besides the 3A common system OP codes.

Options and requirements: Since other instructions use OP CODE $0D_{16}$ (015_8), the SOP instruction is indicated by option 1_{16} (01_8) in bits 0 through 3.

Approximate execution time: 1.20 microseconds.

2.91 *SBN Rx,B*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_X 4	3 B 0
			0101100_2		
			054_8		
			$2C_{16}$		

Function: This instruction sets the selected bit (B) to 1 in a general register (R_X). The remaining bits of the general register are unchanged.

Approximate execution time: 1.20 microseconds.

2.92 *SBR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_x 4	3 R_y 0
			0101101_2 055_8 $2D_{16}$		

Function: This instruction sets a bit (B) in a general register (Rx). The location of bit B is determined by the low four bits of another general register (Ry). The remaining bits of the general register (Rx) are unchanged.

Approximate execution time: 1.20 microseconds.

2.93 *SBS N(RA),B*

Format:

P_H	P_L	BA	14 OP CODE 8	7 B 4	3 N 0
			1100010_2/1100011 142_8/143_8 62_{16}/63_{16}		

Function: This instruction sets bit B in a memory word. The location of the word is determined by adding a number (N) to the contents of RA. The remaining bits of the memory word are unchanged.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 62_{16} (142_8) is used. When RA = R14, bit 8 is 1 and OP CODE 63_{16} (142_8) is used.

Approximate execution time: 4.35 microseconds.

Clearing of a Bit

2.94 *ZCF*

Format:

P_H	P_L	BA	14 OP CODE 8	7 0000_2 4	3 0000_2 0
			00010011_2 011_8 09_{16}	00_8 0_{16}	00_8 0_{16}

Function: This instruction clears the CF flip-flop by setting it to zero.

Approximate execution time: 1.20 microseconds.

2.95 *ZBN Rx,B*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_X 4	3 B 0
			0100100_2 044_8 24_{16}		

Function: This instruction clears a bit (B) in a general register (Rx). The remaining bits of the general register (Rx) are unchanged.

Approximate execution time: 1.20 microseconds.

2.96 *ZBR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8	7 R_X 4	3 R_y 0
			0100101_2 045_8 25_{16}		

Function: This instruction clears a bit (B) in a general register (Rx). The location of bit B is determined by the low four bits of another general register (Ry). The remaining bits of the general register (Rx) are unchanged.

Approximate execution time: 1.20 microseconds.

2.97 *ZBS N(RA),B*

Format:

P_H	P_L	BA	14 OP CODE 8	7 B 4	3 N 0
			1100100_2/1100101_2 144_8/145_8 64_{16}/65_{16}		

Function: This instruction clears bit B in a memory word. The location of the word is determined by adding a number (N) to the contents of RA. The remaining bits of the memory word are unchanged.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 64_{16} (144_8) is used. When RA = R14, bit 8 is 1 and OP CODE 65_{16} (145_8) is used.

Approximate execution time: 4.35 microseconds.

2.98 *ZOP*

Format:

P_H	P_L	BA	14 OP CODE 8 0001101_2 015_8 $0D_{16}$	7 0000_2 4 00_8 0_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction clears the OP CODE FIL bit which points to the common systems set of OP codes associated with the 3A.

Options and requirements: Since other instructions use OP CODE $0D_{16}$ (015_8), the ZOP instruction is indicated by the 0_{16} (00_8) in bits 0 through 3.

Approximate execution time: 1.20 microseconds.

Testing of a Bit

2.99 *ICF Rx,N*

Format:

P_H	P_L	BA	14 OP CODE 8 0110000_2 060_8 30_{16}	7 R_X 4	3 N 0

Function: This instruction tests and inserts the state of the CF into bit N of a general register (Rx). If the CF is equal to 1, bit N of Rx is set. If the CF is equal to 0, bit N of Rx is cleared. The remaining bits of the general register (Rx) are unchanged.

Approximate execution time: 1.20 microseconds.

2.100 *TRPL Rx*

Format:

P_H	P_L	BA	14 OP CODE 8	7 0000_2 4	3 R_x 0
			1011110_2 136_8 $5E_{16}$	00_8 0_{16}	

Function: This instruction tests the low parity (PL) bit (parity on bits 7 through 0) of a general register (Rx). If the PL is equal to 1, the CF is set to 1. If the PL is equal to 0, the CF is cleared.

Options and requirements: Since other instructions use OP CODE $5E_{16}$ (136_8), the TRPL instruction is indicated by the 0_{16} (00_8) in bits 7 through 4.

Approximate execution time: 1.20 microseconds.

2.101 *TRPH Rx*

Format:

P_H	P_L	BA	14 OP CODE 8	7 0001_2 4	3 R_x 0
			1011110_2 136_8 $5E_{16}$	01_8 1_{16}	

Function: This instruction tests the high parity (PH) bit (parity on bits 15 through 8) of a general register (Rx). If the PH is equal to 1, the CF is set to 1. If the PH is equal to 0, the CF is cleared.

Options and requirements: Since other instructions use OP CODE $5E_{16}$ (136_8), the TRPH instruction is indicated by the 1_{16} (01_8) in bits 7 through 4.

Approximate execution time: 1.20 microseconds.

2.102 *TSRPL RS*

Format:

P_H	P_L	BA	14 OP CODE 8 1011111_2 137_8 $5F_{16}$	7 0000_2 4 00_8 0_{16}	3 R_S 0

Function: This instruction tests the P_L of a special register (RS). If the P_L is equal to 1, the CF is set to 1. If the P_L is equal to 0, the CF is cleared. The special register may be one of the registers shown in Table A.

Options and requirements: Since other instructions use OP CODE $5F_{16}$ (137_8), the TSRPL instruction is indicated by the 0_{16} (00_8) in bits 7 through 4.

Approximate execution time: 1.20 microseconds.

2.103 *TSRPH RS*

Format:

P_H	P_L	BA	14 OP CODE 8 1011111_2 137_8 $5F_{16}$	7 0001_2 4 01_8 1_{16}	3 R_S 0

Function: This instruction tests the P_H (parity on bits 8 throug 19) of a special register (RS). If the P_H is equal to 1, the CF is set to 1. If the P_H is equal to 0, the CF is cleared.

Options and requirements: Since other instructions use OP CODE $5F_{16}$ (137_8), the TSRPH instruction is indicated by the 1_{16} (01_8) in bits 7 through 4.

Approximate execution time: 1.20 microseconds.

2.104 *TBN Rx,B*

Format:

P_H	P_L	BA	14 OP CODE 8 0101000_2 050_8 28_{16}	7 R_X 4	3 B 0

Function: This instruction tests bit B in a general register (R_X). If the bit is equal to 1, the CF is set. If the bit is equal to 0, the CF is cleared.

Approximate execution time: 1.20 microseconds.

2.105 *TBR Rx,Ry*

Format:

P_H	P_L	BA	14 OP CODE 8 0101001_2 051_8 29_{16}	7 R_x 4	3 R_y 0

Function: This instruction tests a bit in a general register (R_X). The bit location is determined by the low four bits of another general register (R_Y). If the bit tested is equal to 1, the CF is set to 1. If the bit tested is equal to 0, CF is cleared.

Approximate execution time: 1.20 microseconds.

2.106 *TBS N(RA),B*

Format:

P_H	P_L	BA	14 OP CODE 8 $1010010_2/1010011_2$ $122_8/123_8$ $52_{16}/53_{16}$	7 B 4	3 N 0

Function: This instruction tests bit B of a memory location which is determined by adding a number (N) to the contents of RA. If the bit tested is equal to 1, the CF is set. If the bit tested is equal to 0, the CF is cleared.

Options and requirements: When RA = R12, bit 8 is 0 and OP CODE 52_{16} (122_8) is used. When RA = R14, bit 8 is 1 and OP CODE 53_{16} (123_8) is used.

Approximate execution time: 3.30 microseconds.

2.107 *TCC1*

Format:

P_H	P_L	BA	14 OP CODE 8 1011111_2 137_8 $5F_{16}$	7 0001_2 4 01_8 1_{16}	3 1111_2 0 17_8 F_{16}

Function: This instruction sets the CF equal to zero if this instruction is being executed in CC 0. The CF is set equal to one if this instruction is being executed in CC 1.

Options and requirements: Since other instructions use OP CODE $5F_{16}$ (137_8), the TCC1 instruction is indicated by the F_{16} (17_8) in bits 3 through 0.

Approximate execution time: 1.20 microseconds.

F. Input/Output Instructions

2.108 The input/output (I/O) instructions control the status and transmission of an I/O message to a peripheral unit via an I/O channel. The I/O instructions are divided into the following subclasses:

(a) Control

(b) Test.

Control

2.109 *SIO*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0000_2 4 00_8 0_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction initiates the transmission of an I/O message over one of the I/O channels. This is accomplished in the following sequence; (1) Idle the I/O channel defined in R9 (bits 15 through 10) and load the I/O status register from R9 (bits 9 through 0); (2) Load I/O data (IOD) register from R10; (3) Load R11 from IOD and perform a match on R10 and R11 via use of the data manipulation logic (DML); (4) Transmit message.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the SIO instruction is indicated by the 0_{16} (00_8) option in bits 7 through 4 of this instruction.

Approximate execution time: 1.20 microseconds.

2.110 *SMIO*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0010_2 4 02_8 2_{16}	3 0000_2 0 00_8 0_{16}

Function: Same as SIO instruction (see paragraph 2.109) except that transmission is a ***maintenance*** type message.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the SMIO instruction is indicated by the 2_{16} (02_8) option in bits 7 through 4 of this instruction.

Approximate execution time: 1.20 microseconds.

2.111 *ZIO*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0111_2 4 07_8 7_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction idles an I/O channel defined in bits 15 through 10 of R9.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the ZIO instruction is indicated by the 7_{16} (07_8) in bits 7 through 4.

Approximate execution time: 1.20 microseconds.

Test

2.112 *TIO*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0100_2 4 04_8 4_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction tests an I/O channel to determine if an I/O message is present or not. If an I/O message is present, the contents of IOD are gated to R11 and a parity check is performed. The CF is set and the I/O channel is idled. If an I/O message is not present, the CF is cleared and R11 is unchanged. The contents of R10 are not changed by this instruction.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the TIO instruction is indicated by option 4_{16} (04_8) in bits 7 through 4 of this instruction.

Approximate execution time: 1.20 microseconds.

2.113 *TMIO*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0101_2 4 05_8 5_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction is the same as the TIO instruction (see paragraph 2.112) except that it tests for the presence of a ***maintenance*** type message.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the TMIO instruction is indicated by the 5_{16} (05_8) in bits 7 through 4 of this instruction.

Approximate execution time: 1.20 microseconds.

2.114 *TCH*

Format:

P_H	P_L	BA	14 OP CODE 8 0100111_2 047_8 27_{16}	7 0110_2 4 06_8 6_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction tests an I/O channel defined in R9 to determine if it is in the idle state or not. If the channel is in the idle state, the CF is set to 1. If the channel is not in the idle state, the CF is cleared.

Options and requirements: Since other instructions use OP CODE 27_{16} (047_8), the TCH instruction is indicated by the 6_{16} (06_8) in bits 7 through 4 of this instruction.

Approximate execution time: 1.20 microseconds.

G. Maintenance Instructions—Used to Control the System While in Maintenance Programs

2.115 *HALT*

Format:

P_H	P_L	BA	14 OP CODE 8 10111101_2 135_8 $5D_{16}$	7 0110_2 4 06_8 6_{16}	3 0000_2 0 00_8 0_{16}

Function: This instruction causes the 3A microcode to loop until it is initialized or interrupted. If the 3A is interrupted, the interrupt will return to the HALT instruction.

2.116 *MI*

Format:

P_H	P_L	BA	14	OP CODE 0010111_2 027_8 17_{16}	8	7	SPARE	0
P_H	P_L	15		R_x	8	7	R_y	0

Function: This instruction sets the microinterpret (MINT) flip-flop to indicate the interpret mode. This blocks the normal gating between the microstore and the TO and FROM fields of the MIR and enables several main memory words to be used as microinstructions. Rx is gated to the MIR TO field and RY is gated to the MIR FROM field, and the indicated micro operation is performed. This process is repeated for each main memory word until the micro operation which clears the MINT is performed.

Options and requirements: Only a linear string of main memory words may be performed during the microinterpret cycle and the last word must clear the MINT bit. Interrupts are not processed when the 3A is in the interpret mode.

Approximate execution time: 1.20 microseconds per main memory access.

2.117 *MIS*

Format:

P_H	P_L	BA	14	OP CODE 0010110_2 026_8 16_{16}	8	7	SPARE	0
P_H	P_L	15		R_x	8	7	R_y	0

Function: This instruction sets the MINT which blocks normal gating between the microstore and the TO and FROM fields of the MIR and to indicate the interpret mode. Rx is gated to the MIR TO field and Ry is gated to the MIR FROM field. After the micro operation is performed, the MINT is cleared and the next sequential instruction is performed. Interrupts are not processed when the 3A is in the interpret mode.

Approximate execution time: 2.40 microseconds.

2.118 *MSTF N(RA)*

Format:

P_H	P_L	BA	OP CODE 0000001_2 001_8 01_{16}							$0000_2/0001_2$ $00_8/01_8$ $0_{16}/1_{16}$				N			
P_H	P_L	BECO	CWO	BDSR1	BDSR0	IS01	IS00	UPD1	UPD0	IDL1	IDL0	RW1	RW0	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. The effective address for this instruction is computed by adding N to RA.
2. The contents of the main memory status register are saved.
3. The main memory status register is loaded from the second word of this instruction.
4. The indicated read/write function using register 0 as the destination/source for the data is performed.
5. The previously saved contents of the main memory status register are restored.
6. The CF is cleared.
7. This instruction keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Since other instructions use OP CODE 01_{16} (001_8), the MSTF instruction is indicated by the 0 stored in bit 5 of the first word of this instruction. When RA = R12, bit 4 of the first word is 0. When RA = R14, bit 4 of the first word is 1.

Approximate execution time: 5.40 microseconds.

2.119 *MSTFX Rx(RA)*

Format:

P_H	P_L	BA	OP CODE 0000001_2 001_8 01_{16}							$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$				R_X			
P_H	P_L	BECO	CWO	BDSR1	BDSR0	ISO1	ISO0	UPD1	UPD0	IDL1	IDL0	RW1	RW0	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding Rx to RA.
2. Saves the contents of the main memory status register.
3. Loads the main memory status register from the second word of this instruction.
4. Performs the indicated read/write operation using register 0 as destination/source for the data.
5. Restores the previously saved contents of the main memory register.
6. Clears the CF.
7. Keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Since other instructions use OP CODE 01_{16} (001_8), the MSTFX instruction is indicated by the 1 stored in bit 5 of the first word of this instruction. When RA = R12, bit 4 of the first word is 0. When RA = R14, bit 4 of the first word is 1.

Approximate execution time: 5.25 microseconds.

2.120 *STAF N(RA)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1111100_2
174_8
$7C_{16}$</td><td colspan="4">$0000_2/0001_2$
$00_8/01_8$
$0_{16}/1_{16}$</td><td colspan="4">N</td></tr>
<tr><td>P_H</td><td>P_L</td><td>BECQ</td><td>CW0</td><td>BDSR1</td><td>BDSR0</td><td>IS01</td><td>IS00</td><td>UPD1</td><td>UPD0</td><td>IDL1</td><td>IDL0</td><td>RW1</td><td>RW0</td><td>MM21</td><td>MM20</td><td>MM11</td><td>MM10</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding N to RA.
2. Saves the contents of the main memory status register.
3. Loads the main memory status register from the second word of this instruction.
4. Performs the indicated read/write operation using register 0 as destination/source for the data.
5. Parity is corrected on the data received from the store before it is placed in RO.
6. Clears the CF.
7. If a store error C or A bus parity error occurs:

    (a) The CF is set equal to one.

    (b) The ER is cleared.
8. Restores the previously saved contents of the main memory status register.
9. Caution: This instruction clears the block hardware check bit in the system state register.
10. This instruction keeps an internal timer to prevent the 3A from hanging.
    If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.65 microseconds.

2.121 *STAF Rx(RA)*

Format:

P_H	P_L	BA	OP CODE 1111100_2 174_8 $7C_{16}$							$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$				R_X			
P_H	P_L	BECO	CWO	BDSR1	BDSRO	ISO1	ISOO	UPD1	UPDO	IDL1	IDLO	RW1	RWO	MM21	MM2O	MM11	MM1O
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding Rx to RA.

2. Saves the contents of the main memory status register.

3. Loads the main memory status register from the second word of this instruction.

4. Performs the indicated read/write operation using register 0 as destination/source for the data.

5. Parity is corrected on the data received from the store before it is placed in R0.

6. Clears the CF.

7. If a store error C or a bus parity error occurs:

   (a) The CF is set equal to one.

   (b) The ER is cleared.

8. Restores the previously saved contents of the main memory status register.

9. Caution: This instruction clears the block hardware check bit in the system state register.

10. This instruction keeps an internal timer to prevent the 3A from hanging.
    If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.50 microseconds.

2.122 *CONL N(RA)*

Format:

P_H	P_L	BA	OP CODE 1111010_2 172_8 $7A_{16}$							$0000_2/0001_2$ $00_8/01_8$ $0_{16}/1_{16}$				N			
P_H	P_L	BECO	CWO	BDSR1	BDSRO	ISO1	ISOO	UPD1	UPDO	IDL1	IDLO	RW1	RWO	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding N to RA.
2. Saves the contents of the main memory status register.
3. Reads the off-line store without store error correction or processor error correction.
4. Writes the complement of the received data into the off-line store.
5. Restores the previously saved contents of the main memory status register.
6. Clears the condition flip-flop.
7. This instruction keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.50 microseconds.

2.123 *CONL Rx(RA)*

Format:

P_H	P_L	BA	OP CODE 1111010_2 172_8 $7A_{16}$							$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$				R_X			
P_H	P_L	BEC0	CW0	BDSR1	BDSR0	IS01	IS00	UPD1	UPD0	IDL1	IDL0	RW1	RW0	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding Rx to RA.
2. Saves the contents of the main memory status register.
3. Reads the on-line store without store error correction or processor error correction.
4. Writes the complement of the received data into the on-line store.
5. Restores the previously saved contents of the main memory status register.
6. Clears the condition flip-flop.
7. This instruction keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.35 microseconds.

2.124 *COFL N(RA)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1111011_2
173_8
$7B_{16}$</td><td colspan="4">$0000_2/0001_2$
$00_8/01_8$
$0_{16}/1_{16}$</td><td colspan="4">N</td></tr>
<tr><td>P_H</td><td>P_L</td><td>BECO</td><td>CWO</td><td>BDSR1</td><td>BDSRO</td><td>IS01</td><td>IS00</td><td>UPD1</td><td>UPDO</td><td>IDL1</td><td>IDLO</td><td>RW1</td><td>RWO</td><td>MM21</td><td>MM20</td><td>MM11</td><td>MM10</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding N to RA.
2. Saves the contents of the main memory status register.
3. Reads the on-line store without store error correction or processor error correction.
4. Writes the complement of the received data into the on-line store.
5. Restores the previously saved contents of the main memory status register.
6. Clears the condition flip-flop.
7. This instruction keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.80 microseconds.

2.125 *COFL Rx(RA)*

Format:

P_H	P_L	BA	OP CODE 1111011_2 173_8 $7B_{16}$							$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$				R_x			
P_H	P_L	BECO	CWO	BDSR1	BDSRO	ISO1	ISOO	UPD1	UPDO	IDL1	IDLO	RW1	RWO	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction results in the following functions:

1. Computes the effective address for this instruction by adding Rx to RA.
2. Saves the contents of the main memory status register.
3. Reads the on-line store without store error correction or processor error correction.
4. Writes the complement of the received data into the on-line store.
5. Restores the previously saved contents of the main memory status register.
6. Clears the condition flip-flop.
7. This instruction keeps an internal timer to prevent the 3A from hanging. If the instruction times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 7.65 microseconds.

2.126 *NOP*

Format:

P_H	P_L	BA	14 OP CODE 8 0001100_2 014_8 $0C_{16}$	7 0000_2 4 00_8 0_{16}	3 0000_2 0 00_8 0_{16}

Function: No operation

Approximate execution time: 1.20 microseconds.

H. Miscellaneous

2.127 *COFLX Rx(RA)*

Format:

			14 OP CODE 8	7 4	3 0
P_H	P_L	BA	1111011_2 173_8 $7B_{16}$	$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$	R_X

Function: This instruction effects a complement write the off-line store at location RA indexed by Rx:

1. Compute the effective address for this instruction by adding Rx to RA.
2. Save the contents of the main memory status register.
3. Read the off-line store without store error correction or processor error corrections.
4. Write the complement of the received data into the off-line store.
5. Restore the previously saved contents of the main memory status register.
6. Zero the condition flip-flop.
7. This instruction starts an internal timer to prevent the CC from hanging. If the timer times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 an 1 for RA = 14.

Approximate execution time: 10.05 microseconds.

2.128 *CONLX Rx(RA)*

Format:

P_H	P_L	BA	14 OP CODE 8 1111010_2 172_8 $7A_{16}$	7 4 $0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$	3 0 R_X

Function: This instruction effects a complement write the on-line store at location RA indexed by Rx:

1. Compute the effective address for this instruction by adding Rx to RA.
2. Save the contents of the main memory status register.
3. Read the on-line store without store error correction or processor error correction.
4. Write the complement of the received data into the on-line store.
5. Restore the previously saved contents of the main memory status register.
6. Zero the condition flip-flop.
7. This instruction starts an internal timer to prevent the CC from hanging. If the timer times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined by the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 9.75 microseconds.

2.129 *STAFX Rx(RA)*

Format:

P_H	P_L	BA	OP CODE 1111100_2 174_8 $7C_{16}$							$0010_2/0011_2$ $02_8/03_8$ $2_{16}/3_{16}$				R_X			
P_H	P_L	BECO	CWO	BDSR1	BDSR0	ISO1	ISO0	UPD1	UPD0	IDL1	IDL0	RW1	RW0	MM21	MM20	MM11	MM10
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store access function using register 0 at location RA indexed by Rx:

1. Compute the effective address for this instruction by adding Rx to RA.
2. Save the contents of the main memory status register.
3. Load the main memory status register from the second word of this instruction. The isolate bits are left as they were upon entry.
4. Perform the indicated read/write operation using register 0 as destination/source for the data.
5. Parity is corrected on the data received from the store before it is placed in register 0.
6. Zero the condition flip-flop.
7. If a store error C occurs, the condition flip-flop is set to a 1 and the error register is cleared.
8. Restore the previously saved contents of the main memory status register.
9. This instruction starts an internal time to prevent the CC from hanging. If the timer times out, register 0 is set to all ones.

Options and requirements: Bit 4 is determined the value of RA. It is 0 for RA = 12 and 1 for RA = 14.

Approximate execution time: 9.3 microseconds.

3. GENERAL PURPOSE PHASE 1 EXTENSIONS TO THE 3A PROCESSOR INSTRUCTION SET

3.01 Extended instructions are added to the basic set to facilitate the implementation of the extended operating system and to enhance its functional capabilities.

3.02 The added (extended) instructions provide the following features:

- Supervisor call
- Subroutine call and return via register links
- Shift
- Move block
- Signed arithmetic
- Test and set
- Breakpoint execute
- Improved base register addressing capabilities
- Communications
- Parallel channel
- Direct memory access.

DESCRIPTION OF THE GENERAL PURPOSE EXTENSIONS TO THE BASIC 3A INSTRUCTION SET

3.03 The information contained in Parts 1 and 2 of this section applies to the extended instruction set except as specifically detailed.

General Instruction Format (See Paragraphs 2.05 Through 2.11)

3.04 Specific information and the precise format(s) for each of the extended instructions follows.

Classes of Instructions (See Paragraph 2.12)

3.05 The extended instructions are described within the class to which they apply. Definitions of terms used in the description of extended instructions are given in Table B.

A. Data Transfer Instructions (See Paragraph 2.13)

Memory-to-Register

3.06 *LALL Y* (Load all registers from memory)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0101011_2
053_8
$2B_{16}$</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction loads registers R0, R1, . . ., R15 from memory, at locations Y, Y+1, . . ., Y+15, respectively.

Approximate execution time: 27.9 microseconds.

3.07 ➧*LALLI Y*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0101011_2
053_8
$2B_{16}$</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: Load all registers indirect. Load registers R0 through R15 from memory starting at the address specified by the 20 bits at locations Y and Y+1.◆

3.08 *LB Rx,RPx*

Format:

P_H	P_L	BA	OP CODE 01111111_2 077_8 $3F_{16}$	RP_X	0010_2 02_8 2_{16}
P_H	P_L	00000000_2 000_8 00_{16}		RP_X	R_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction loads a byte into Rx from a memory location derived from RPx:

- Compute the word address. See Appendix 1.
- Load register byte (bits 7 through 0 of Rx) with the contents of buffer byte.
- Bits 15 through 8 of Rx are unchanged.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE $3F_{16}$ (077_8), the LB option is indicated by the 2_{16} (02_8) in bits 0 through 3 of the first word.

Approximate execution time: 5.40 microseconds.

3.09 *LBA Rx,RPx*

Format:

P_H	P_L	BA	OP CODE 0111111_2 077_8 $3F_{16}$	RP_X	0011_2 03_8 3_{16}
P_H	P_L	00000000_2 000_8 00_{16}		RP_X	R_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction loads a byte into Rx from a memory location derived from RPx and increment count:

- Compute the word address. See Appendix 1.
- Load register byte (bits 7 through 0 of Rx) with the contents of buffer byte.
- Increment byte count by 1.
- Bits 15 through 8 of Rx are unchanged.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE $3F_{16}$ (077_8), the LBA option is indicated by the 3_{16} (03_8) in bits 0 through 4 of the first word.

Approximate execution time: 5.55 microseconds.

3.10 ♦*LDK RPx,K(RPy)*

Format:

P_H	P_L	BA	OP CODE 0001011_2 013_8 $0B_{16}$	1000_2 10_8 8_{16}	RP_Y
P_H	P_L	K			RP_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: Load register pair at base plus offset. Load registers RPx and RPx+1 from memory at the address formed by adding the offset K to the 20-bit base address in RPy.♦

3.11 ▶*LDX RPx,Rx(RPy)*

Format:

P_H	P_L	BA	OP CODE 0000010_2 002_8 02_{16}	1000_2 10_8 8_{16}	RP_Y
P_H	P_L	0		R_X	RP_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: Load register pair from memory at base plus index. Load registers RPx and RPx+1 from memory at the address formed by adding the contents of index register Rx to the 20-bit base address in RPy.◀

3.12 *LMP Y* (See paragraph 3.15 for function of this instruction.)

Format:

P H	P_L	BA	OP CODE 0101011_2 053_8 $2B_{16}$	0000_2 00_8 0_{16}	Y 19-16
P_H	P_L	Y 15-0			
P_H	P_L	DATA			
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Approximate execution time: 5.25 microseconds.

3.13 ***LMPK K(RPx)*** (See paragraph 3.15 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.3 microseconds.

3.14 ***LMPX Rx(RPx)*** (See paragraph 3.15 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0001_2
01_8
1_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.3 microseconds.

3.15 Function of the instructions in paragraphs 3.12 (LMP Y), 3.13 [LMPK K(RPx)], and 3.14 [LMPX Rx(RPx)]: The instructions load multiple registers as specified by the data word following the instruction in the following manner:

- Compute the effective address (EA).
- Pick up the data word which ***must*** immediately follow this instruction and increment the program address (PA) to point to the instruction following the data word (3-word instruction).
- Scan each bit in the data word from bits 0 through 15. If the bit is a one, load the register corresponding to the bit position with the contents of memory at location EA+m where m is the number of ones in the bits of the data word which are less significant than the bit being operated on.
- The initial values of Rx, RPx, and RPx+1 are used to compute the EA. These registers may be loaded during execution of the instruction.

3.16 ***LRP RP$_X$,Y*** (See paragraph 3.19 for function of this instruction.)

Format:

<table>
<tr><td>P$_H$</td><td>P$_L$</td><td>BA</td><td colspan="7">OP CODE
0110100_2
064_8
34_{16}</td><td colspan="4">RP$_X$</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P$_H$</td><td>P$_L$</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.75 microseconds.

3.17 ***LRPK RP$_X$,K(RP$_Y$)*** (See paragraph 3.19 for function of this instruction.)

Format:

<table>
<tr><td>P$_H$</td><td>P$_L$</td><td>BA</td><td colspan="7">OP CODE
00010111_2
013_8
$0B_{16}$</td><td colspan="4">0100_2
04_8
4_{16}</td><td colspan="4">RP$_y$</td></tr>
<tr><td>P$_H$</td><td>P$_L$</td><td colspan="12">K</td><td colspan="4">RP$_X$</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 7.02 microseconds.

3.18 ***LRPX RP$_X$,R$_X$(RP$_Y$)*** (See paragraph 3.19 for function of this instruction.)

Format:

<table>
<tr><td>P$_H$</td><td>P$_L$</td><td>BA</td><td colspan="7">OP CODE
0000010_2
002_8
02_{16}</td><td colspan="4">0100_2
04_8
4_{16}</td><td colspan="4">RP$_y$</td></tr>
<tr><td>P$_H$</td><td>P$_L$</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R$_X$</td><td colspan="4">RP$_X$</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 7.05 microseconds.

3.19 Function of the instructions in paragraphs 3.16 (LRP RPx,Y), 3.17 [LRPK RPx,K(RPy)], and 3.18 [LRPX RPx,Rx(RPy)]: The instructions load a register pair from memory (20 bits) in the following manner.

- Compute the effective address (EA).
- Load 20 bits of address information into registers RPx, RPx+1 from memory at locations EA and EA+1.

  Contents of EA bits 3 through 0 are loaded into RPx bits 3 through 0.

  Contents of EA+1 bits 15 through 0 are loaded into RPx+1 bits 15 through 0.
- The upper 12 bits of RPx are unchanged.
- If RPx equals RPy, the initial value is used to determine the EA.

3.20 *LRPI RPx,YI*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0101111_2
057_8
$2F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">YI 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">YI 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction loads a register pair with immediate data (20 bits) in the following manner:

- Load 20 bits of immediate data into registers RPx and RPx+1.
- The upper 12 bits of RPx are unchanged.

Approximate execution time: 2.7 microseconds.

3.21 *LRPAK RPx,K(RPy)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100011_2
043_8
23_{16}</td><td colspan="4">0100_2
04_8
4_{16}</td><td colspan="4">RP_Y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load pair RPx from memory at location RPy displaced by K and update RPy:

1. Compute the effective address (EA = K+RPy).

2. Update RPy.

   EA bits 19 through 16 are loaded into RPy bits 3 through 0.

   EA bits 15 through 0 are loaded into RPy+1 bits 15 through 0.

   RPy bits 15 through 4 are unaffected.

3. Load RPx with 20 bits of data from memory location EA.

   Contents of EA bits 3 through 0 are loaded into RPx bits 3 through 0.

   Contents of EA+1 bits 15 through 0 are loaded into RPx+1 bits 15 through 0.

   RPx bits 15 through 4 are unaffected.

3.22 *LRPAX RPx,Rx(RPy)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
00100000_2
020_8
10_{16}</td><td colspan="4">0100_2
04_8
4_{16}</td><td colspan="4">RP_Y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">0</td><td colspan="4">R_X</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load register pair RPx from memory at location RPy indexed by Rx and update RPy:

1. Compute the effective address (EA = Rx+RPy).

2. Update RPy.

   EA bits 19 through 16 are loaded in RPy bits 3 through 0.

   EA bits 15 through 0 are loaded into RPy+1 bits 15 through 0.

   RPy bits 15 through 4 are unaffected.

3. Load RPx with 20 bits of data from memory location EA.

   Contents of EA bits 3 through 0 are loaded into RPx bits 3 through 0.

   Contents of EA+1 bits 15 through 0 are loaded into RPx+1 bits 15 through 0.

   RPx bits 15 through 4 are unaffected.

3.23 ***LXAK Rx,K(RPx)*** (See paragraph 3.25 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100011_2
043_8
23_{16}</td><td colspan="4">0001_2
01_8
1_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.0 microseconds.

3.24 ***LXAX Rx,Ry(RPx)*** (See paragraph 3.25 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0010000_2
020_8
10_{16}</td><td colspan="4">0001_2
01_8
1_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_y</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.0 microseconds.

3.25 Function of instructions in paragraphs 3.23 [LXAK Rx,K(RPx)] and 3.24 [LXAX Rx,Ry(RPx)]: The instructions load a register from memory and update register pair in the following manner:

- Compute the effective address (EA).
- Update 20 bits of RPx, RPx+1 to the EA. The upper 12 bits of RPx are unchanged.
- Load Rx with the contents of memory at the EA.

If Rx equals RPx or RPx+1, the register just updated will be overwritten by the contents of memory at the EA.

3.26 *LXK Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 0001011_2 013_8 $0B_{16}$	0001_2 01_8 1_{16}	RP_X
P_H	P_L	K			R_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction loads a register from memory in the following manner:

- Compute the effective address (EA).
- Load Rx with contents of memory at the EA.

If Rx equals RPx or RPx+1, the initial contents (before load) will be used to compute the EA.

Approximate execution time: 4.65 microseconds.

3.27 *LXX Rx,Ry(RPx)*

Format:

P_H	P_L	BA	OP CODE 0000010_2 002_8 02_{16}	0001_2 01_8 1_{16}	RP_X
P_H	P_L	00000000_2 000_8 00_{16}		R_y	R_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction loads a register from memory in the following manner:

- Compute effective address EA.
- Load Rx with contents of memory at the EA.

If Rx equals RPx or RPx+1, the initial contents (before load) will be used to compute the EA.

Approximate execution time: 4.65 microseconds.

Register-to-Memory Instructions

3.28 *STALL Y*

Format:

P_H	P_L	BA	OP CODE 0101011_2 053_8 $2B_{16}$	01	Y 19-16
P_H	P_L	Y 15-0			
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction stores all registers in memory. It stores the contents of registers R0, R1, . . ., R15 into memory locations Y, Y+1, . . ., Y+15, respectively.

Approximate execution time: 28.5 microseconds.

3.29 ♦*STALLI Y*

Format:

P_H	P_L	BA	OP CODE 0101011_2 053_8 $2B_{16}$	0110_2 06_8 6_{16}	Y 19-16
P_H	P_L	Y 15-0			
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: Store all registers indirect. Store registers R0 through R15 into memory starting at the address specified by the 20 bits at locations Y and Y+1.♦

3.30 *STB Rx,RPx*

Format:

P_H	P_L	BA	OP CODE 0111111_2 077_8 $3F_{16}$	RP_X	0100_2 04_8 4_{16}
P_H	P_L	0	000000000_2 000_8 00_{16}	RP_X	R_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction stores a byte from register (Rx) into memory location derived from the RPx:

- Compute the word address. See Appendix 1.
- Store register byte into buffer byte (other byte remains unchanged).
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE $3F_{16}$ (077_8), the STB option is indicated by the 4_{16} (04_8) in bits 0 through 4 of the first word.

Approximate execution time: 6.45 microseconds.

3.31 *STBA Rx,RPx*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0111111_2
077_8
$3F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0101_2
05_8
5_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">RP_X</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction stores a byte from register (Rx) into memory location derived from RPx and increment count:

- Compute the word address. See Appendix 1.
- Store register byte into buffer byte (other byte remains unchanged).
- Increment byte count by 1.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE $3F_{16}$ (077_8), the STBA option is indicated by the 5_{16} (05_8) in bits 0 through 3 of the first word.

3.32 ♦*STDK RPx,K(RPy)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0001011_2
013_8
$0B_{16}$</td><td colspan="4">1001_2
11_8
9_{16}</td><td colspan="4">RP_Y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: Store register pair in memory at base plus offset. Store registers RPx and RPx+1 into memory at the address formed by adding the offset K to the 20-bit base address in RPy.♦

3.33 ♦*STDX RPx,Rx(RPy)*

Format:

P_H	P_L	BA	OP CODE 0000010_2 002_8 02_{16}	1001_2 11_8 9_{16}	RP_Y
P_H	P_L	0		R_X	RP_X
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: Store register pair at base plus index. Store registers RPx and RPx+1 into memory at the address formed by adding the contents of index register Rx to the 20-bit base address in RPY.♦

3.34 *STMP Y* (See paragraph 3.37 for function of this instruction.)

Format:

P H	P_L	BA	OP CODE 0101011_2 053_8 $2B_{16}$	0001_2 01_8 1_{16}	Y 19-16
P_H	P_L	Y 15-0			
P_H	P_L	DATA			
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Approximate execution time: 5.25 microseconds.

3.35 ***STMPK K(RPK)*** (See paragraph 3.37 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0100_2
04_8
4_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.3 microseconds.

3.36 ***STMPX Rx(RPx)*** (See paragraph 3.37 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0011_2
03_8
3_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.3 microseconds.

3.37 Function of instructions in paragraphs 3.34 (STMP Y), 3.35 [STMPK K(RPx)], and 3.36 [STMPX Rx(RPx)]: The instructions store multiple registers as specified by the data word following this instruction in the following manner:

- Compute the effective address (EA).
- Pick up the data word which ***must*** immediately follow this instruction and increment the PA to point to the instruction following the data word.
- Scan the data word from bits 0 through 15. If the bit is one, store the contents of the register corresponding to the bit position into memory at location EA+m where m is the number of ones in lower bit positions of this word.

3.38 ***STRP Y*** (See paragraph 3.41 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0110101_2
065_8
35_{16}</td><td colspan="4">RP_X</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 7.2 microseconds.

3.39 ***STRPK*** $RP_X,K(RP_Y)$ (See paragraph 3.41 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0001011_2
013_8
$0B_{16}$</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">RP_y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 7.5 microseconds.

3.40 ***STRPX*** $RP_X,R_X(RP_Y)$ (See paragraph 3.41 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0000010_2
002_8
02_{16}</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">RP_y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 7.5 microseconds.

3.41 Functions of instructions in paragraphs 3.38 (STRP Y), 3.39 [STRPK RPx,K(RPy)], and 3.40 [STRPX RPx,Rx(RPy)]:

The instructions store a register pair in memory (20 bits) in the following manner:

- Compute the effective address (EA).
- Store 20 bits of address information from registers RPx, RPx+1 into memory at locations EA and EA+1.

$$[EA] \longleftarrow [EA]_{15\text{-}4} \quad (RP_X)_{3\text{-}0}$$

$$[EA+1] \longleftarrow (RP_X+1)$$

- The upper 12 bits of the contents of location EA remain unchanged.

3.42 *STRPAK PRx,K(RPy)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100011₂
043₈
23₁₆</td><td colspan="4">0101₂
05₈
5₁₆</td><td colspan="4">RP_Y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store the contents of register pair RPx into memory location RPy displaced by K and update register pair RPy:

1. Compute the effective address (EA = K+RPy).
2. Update RPy

   EA bits 19 through 16 are loaded into RPy bits 3 through 0.

   EA bits 15 through 0 are loaded into RPy+1 bits 15 through 0.

   RPy bits 15 through 4 are unaffected.

3. Store 20 bits of the contents of RPx into memory locations EA and EA+1.

   RPx bits 3 through 0 are loaded into EA bits 3 through 0.

   RPx+1 bits 15 through 0 are loaded into EA+1 bits 15 through 0.

   EA bits 15 through 4 are unaffected.

3.43 *STRPAX RPx,Rx(RPy)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0010000_2
020_8
10_{16}</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">RP_Y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store the contents of register pair RPx into memory location RPy indexed by Rx and update RPy:

1. Compute the effective address (EA = Rx+RPy).

2. Update RPy

   EA bits 19 through 16 are loaded into RPy bits 3 through 0.

   EA bits 15 through 0 are loaded into RPy+1 bits 15 through 0.

   RPy bits 15 through 4 are unaffected.

3. Store 20 bits of the contents of RPx into EA and EA+1.

   RPx bits 3 through 0 are loaded into EA bits 3 through 0.

   RPx+1 bits 15 through 0 are loaded into EA+1 bits 15 through 0.

   EA bits 15 through 4 are unaffected.

3.44 *STXAK Rx,K(RPx)* (See paragraph 3.46 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100011_2
043_8
23_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.85 microseconds.

3.45 ***STXAX Rx,Ry(RPx)*** (See paragraph 3.46 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0010000_2
020_8
10_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_y</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.85 microseconds.

3.46 Function of the instructions in paragraphs 3.44 [STXAK Rx,K(RPx)]and 3.45 [STXAX Rx,Ry(RPx)]:

The instructions store a register in memory and update a register pair in the following manner:

- Compute the effective address (EA).
- Update 20 bits of RPx and RPx+1 to the EA. The upper 12 bits of RPx are unchanged.
- Store the contents of Rx into memory at the EA.

If Rx equals RPx or RPx+1, the contents (after update) will be stored.

3.47 ***STXK Rx,K(RPx)***

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
00010111_2
013_8
08_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction stores the contents of a register into memory. It computes the effective address (EA) and stores the contents of Rx into memory at the EA.

Approximate execution time: 4.5 microseconds.

3.48 *STXX Rx,Ry(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0000010_2
002_8
02_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">Ry</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction stores the contents of a register into memory. It computes the effective address (EA) and stores the contents of Rx into memory at the EA.

Approximate execution time: 4.5 microseconds.

Register-to-Register Instructions

3.49 *RPAK K(RPx)* (See paragraph 3.51 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100011_2
043_8
23_{16}</td><td colspan="4">0011_2
03_8
3_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 4.2 microseconds.

3.50 *RPAX Rx(RPx)* (See paragraph 3.51 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">14 OP CODE 8
0010000_2
020_8
10_{16}</td><td colspan="4">0011_2
03_8
3_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">Rx</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 4.2 microseconds.

3.51 Function of instructions in paragraphs 3.49 [RPAK K(RPx)] and 3.50 [RPAX Rx(RPx)]:

The instructions update a register pair in the following manner:

- Compute the effective address (EA).
- Update 20 bits of RP, RP+1 to the EA. The upper 12 bits of RP are unchanged.

Memory-to-Memory Instructions

3.52 *MBLK RPx,RPy*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
01111111_2
077_8
$3F_{16}$</td><td colspan="4">0000_2
00_8
0_{16}</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">RP_X</td><td colspan="4">RP_y</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction moves a block of memory in the following manner:

- The number of memory words (N) to be moved is read from register 0.
- Move N words from memory starting at the location specified by (RPy) to memory starting at the location specified by (RPx).
- Decrement R0 and increment RPx and RPy as the instruction progresses until (R0) equals zero.
- Test for interrupts after each move; if found, reset PA to point to this instruction for continued execution after the interrupt routine is completed.

Requirements and Notes:

- R0 must be initialized. If it is zero, no data will be moved. A positive 16-bit quantity is assumed.
- R0, RPx, and RPy are modified: (R0) ← 0; (RP) ← (RPx) + N; (RPy) ← (RPy) + N.
- In the panel step mode, only one move is executed per "execute" step.
- Write protect is not checked by this instruction.

B. Branch Instructions

Branch-and-Link

3.53 ***BAL RPx,Y*** (See paragraph 3.56 for function of this instruction.)

Format:

<table>
<tr><td>PP_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0101110_2
056_8
$2E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 3.45 microseconds.

3.54 ***BALK RPx,K(RPY)*** (See paragraph 3.56 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0000_2
00_8
0_{16}</td><td colspan="4">RP_y</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">RP_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 4.65 microseconds.

3.55 ***BALX RPx,Rx(RPY)*** (See paragraph 3.56 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0100110_2
046_8
26_{16}</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">RPy</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">Rx</td><td colspan="4">RPx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 4.2 microseconds.

3.56 Function of the instructions in paragraphs 3.53 (BAL RPx,Y), 3.54 [BALK RPx ,K(RPy)], and 3.55 [BALX RPx,Rx(RPy)]:

The instructions effect a branch and link in the following manner:

- Compute the effective address (EA).
- Set the 20-bit address of the following instruction into registers RPx, RPx+1. The upper 12 bits of RPx are unchanged.
- Branch to the EA.
- The BA bit must be set in the instruction at the EA.
- The BA bit must be set in the instruction following the BAL, BALX, or BALK.
- If RPx equals RPy, the initial value is used to compute the EA.

Subroutine Instructions

3.57 *RET RPx*

Format:

P_H	P_L	BA	OP CODE 1100001_2 141_8 61_{16}	0000_2 00_8 0_{16}	RPx
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction effects a return from subroutine and test R0 in the following manner:

- Set CF to 1 if the contents of R0 = 0 or to 0 if the contents of R0 ≠ 0.
- Branch to the 20-bit address contained in RPx, RPx+1 (RPx is even).
- The BA bit must be set at the destination.

Approximate execution time: 2.7 microseconds.

3.58 *RETSN RPx,N*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0111101_2
075_8
$3D_{16}$</td><td colspan="4">RP_X</td><td colspan="4">N</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a return from subroutine, set and test R0 in the following manner:

- Set the contents of R0 to the value N.

- Set the CF to 1 if N = 0 or to 0 if N ≠ 0.

- Branch to the 20-bit address contained in RPx, RPx+1 (RPx is even).

Approximate execution time: 3.0 microseconds.

Miscellaneous

3.59 *SVC N*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1100001_2
141_8
61_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">N</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a supervisor call in the following manner:

- Set bit 2 of the system status register to block interrupts.

- Execute a BSAI 2×(32+N) instruction [locations $X(40_{16})$ through $X(5F_{16})$].

- The BA bit should be set in the instruction following the SVC unless the SVC routine terminates with a PIE instruction.

Approximate execution time: 6.6 microseconds.

3.60 ***XQT Y*** (See paragraph 3.62 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0001_2
01_8
1_{16}</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.25 microseconds + instruction.

3.61 ***XQTEI Y*** (See paragraph 3.62 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0011_2
03_8
3_{16}</td><td colspan="4">Y 19-16</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">Y 15-0</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.25 microseconds + instruction.

3.62 Function of instruction in paragraphs 3.60 (XQT Y) and 3.61 (XQTEI Y):

The instructions effect a breakpoint execute and return in the following manner:

- If XQTEI (only), enable interrupts (zero BIN bit in system status register).
- Load PA with the 20-bit address from the contents of locations Y and Y+1 of memory.
- Load store instruction register (SIR) with the contents of location Y+2.

Note: The CF and all general purpose registers are unchanged. This is similar to a branch indirect via Y ***except*** the first word of the instruction is picked up from location Y+2 but executed as if it were located at the address specified in Y and Y+1.

- These instructions were designed to enhance the use of breakpoints by debugging programs. They permit resumption of program execution from a breakpoint without restoring or simulating the instruction formerly located at the breakpoint.
- Without permitting interrupts, start execution of the instruction in the instruction register.

C. Arithmetic Instructions

3.63 *AAR Rx,Ry*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0110111_2
067_8
37_{16}</td><td colspan="4">R_x</td><td colspan="4">R_y</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects an arithmetic add register in the following manner:

- Add the contents of Ry to the contents of Rx and place the result in Rx.
- Set the CF equal to:

  (a) One if arithmetic overflow occurs assuming bit 15 is the sign bit and two's complement representation for negative numbers.

  (b) Zero if no arithmetic overflow occurs.

Approximate execution time: 1.95 through 3.0 microseconds.

3.64 *A1SK K(RPx)* (See paragraph 3.66 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0001011_2
013_8
$0B_{16}$</td><td colspan="4">0000_2
00_8
0_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 6.75 microseconds.

3.65 ***A1SX Rx(RPx)*** (See paragraph 3.66 for function of this instruction.)

Format:

P_H	P_L	BA	OP CODE 0000010_2 002_8 02_{16}	0000_2 00_8 0_{16}	RPx
P_H	P_L	00000000_2 000_8 00_{16}		Rx	0000_2 00_8 0_{16}
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Approximate execution time: 6.75 microseconds.

3.66 Function of the instructions in paragraphs 3.64 [A1SK K(RPx)] and 3.65 [A1SK Rx(RPx)]:

The instructions add one to store and test result in the following manner:

- Add one to the contents of memory at the effective address.

- Set the CF equal to:

  (a) One when the result is zero.

  (b) Zero when the result is not zero.

3.67 ***CAR Rx,Ry***

Format:

P_H	P_L	BA	OP CODE 1100000_2 140_8 60_{16}	R_X	R_y
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction effects a compare arithmetic in the following manner:

- Arithmetically compare the contents of registers Rx and Ry. Bit 15 contains the sign and two's complement arithmetic is used.

- Set the CF to:

  (a) One if (Rx) < (Ry).

  (b) Zero if (Rx) ≥ (Ry).

Approximate execution time: 3.15 microseconds.

3.68 *CASR Rx,Ry*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0110110_2
066_8
36_{16}</td><td colspan="4">R_X</td><td colspan="4">R_y</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a compare arithmetic and execute 3-way branch in the following manner:

- Arithmetically compare the contents of Rx and Ry. Bit 15 specifies the sign and two's complement representation is used for negative numbers.

- Branch to:

  (a) NA if (Rx) < (Ry).

  (b) NA+1 if (Rx) = (Ry) (BA must be set).

  (c) NA+2 if (Rx) > (Ry) (BA must be set).

Where NA is the next address (immediately following this instruction).

Options and requirements:

- Caution must be used with this instruction. Two instructions which immediately follow ***must*** be single-word instructions. The BA bit must be set on the second and third instructions which follow.

Approximate execution time: 2.85 through 5.7 microseconds.

3.69 ***SAR Rx,Ry***

Format:

PH	PL	BA	OP CODE 0111000_2 070_8 38_{16}	Rx	Ry
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction effects an arithmetic subtract register in the following manner:

- Subtract the contents of Ry from the contents of Rx and store the result in Rx.
- Set the CF equal to:

  (a) One if arithmetic underflow occurs assuming bit 15 is the sign bit and two's complement representation is used for negative numbers.

  (b) Zero if no arithmetic underflow occurs.

Approximate execution time: 2.4 through 3.45 microseconds.

3.70 ***S1S Y*** (See paragraph 3.73 for function of this instruction.)

Format:

PH	PL	BA	OP CODE 0101011_2 053_8 $2B_{16}$	0100_2 04_8 4_{16}	Y 19-16
PH	PL	Y 15-0			
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Approximate execution time: 5.85 microseconds.

3.71 ***S1SK K(RPx)*** (See paragraph 3.73 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0001011_2
013_8
$0B_{16}$</td><td colspan="4">0110_2
06_8
6_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.85 microseconds.

3.72 ***S1SX Rx(RPx)*** (See paragraph 3.73 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0000010_2
002_8
02_{16}</td><td colspan="4">0110_2
06_8
6_{16}</td><td colspan="4">RPx</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">Rx</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.85 microseconds.

3.73 Function of the instructions in paragraphs 3.70 (S1S Y), 3.71 [S1SK K(RPx)], and 3.72 [S1SX Rx(RPx)]:

The instructions subtract one from store and test result in the following manner:

- Subtract one from the contents of store at the effective address and test the result for zero.
- Set the CF equal to:

  (a) One when the result is zero.

  (b) Zero when the result is not zero.

D. Bit Operational Instructions

3.74 *SLL Rx,Ry*

Format:

P_H	P_L	BA	OP CODE 1010100_2 124_8 54_{16}	0100_2 04_8 4_{16}	0000_2 00_8 0_{16}
P_H	P_L	00000000_2 000_8 00_{16}		Rx	Ry
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction effects a shift Rx logical an amount determined by 4 bits of Ry in the following manner:

- The amount of the shift (N) is determind by the low 4 bits of Ry.
- Shift Rx left by N bit positions.
- Fill the low N bits with zero.
- CF is unchanged.

Approximate execution time: 6.0 microseconds.

3.75 *SLLN Rx,N*

Format:

P_H	P_L	BA	OP CODE 1010100_2 124_8 54_{16}	0101_2 05_8 5_{16}	0000_2 00_8 0_{16}
P_H	P_L	00000000_2 000_8 00_{16}		R_X	N
17	16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0

Function: This instruction effects a shift Rx left logical by N bit positions in the following manner:

- Shift Rx left by N bit positions.
- Fill the low N bits with zero.
- CF is unchanged.

Approximate execution time: 6.0 microseconds.

3.76 *SRL Rx,Ry*

Format:

<table>
<tr><td>PH</td><td>PL</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0010_2
02_8
2_{16}</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>PH</td><td>PL</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">R_y</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a shift Rx right logical an amount determined by the low 4 bits of Ry in the following manner:

- The amount of the shift (N) is determined by the low 4 bits of Ry.
- Shift Rx right by N bit positions.
- CF is unchanged.

Approximate execution time: 4.65 microseconds.

3.77 *SRLN Rx,N*

Format:

<table>
<tr><td>PH</td><td>PL</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0000_2
00_8
0_{16}</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>PH</td><td>PL</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">N</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a shift Rx right logical by N bit positions in the following manner:

- Shift Rx right by N bit positions.
- Fill upper N bits with zero.
- CF is unchanged.

Approximate execution time: 4.2 microseconds.

3.78 ***TSBSK B,K(RPx)*** (See paragraph 3.80 for function of this instruction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0001011_2
013_8
$0B_{16}$</td><td colspan="4">0111_2
07_8
7_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">B</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.4 microseconds.

3.79 ***TSBSX B,Rx(RPx)*** (See paragraph 3.80 for function of this instruuction.)

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
0000010_2
002_8
02_{16}</td><td colspan="4">0111_2
07_8
7_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="8">00000000_2
000_8
00_{16}</td><td colspan="4">R_X</td><td colspan="4">B</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Approximate execution time: 5.25 microseconds.

3.80 The instruction in paragraphs 3.78 [TSBSK B,K(RPx)] and 3.79 [TSBSX B,Rx(RPx)] effects a test and set bit B in memory in the following manner:

- These instructions operate on word WX at the effective address.
- Set the CF equal to bit B in word WX.
- Set bit B in word WX.
- All other bits of WX are unchanged.

E. Input/Output Instructions

Parallel Channel (I/O) Instructions

3.81 *RD*

Format:

P_H	P_L	BA	OP CODE 11011100_2 154_8 $6C_{16}$	0000_2 00_8 0_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction effects normal read data from parallel channel in R9 into R11:

- Load data from the device defined in R9 bits (15 through 0) into R11. Parity is initially checked then regenerated.

- This is the normal (nonmaintenance) read instruction.

- The CF is set to:

(a) One if no errors are detected.

(b) Zero if:

—A 10-microsecond device time-out occurs.

—The device reports an error.

—Bad parity on the received data.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the RD option is indicated by the 00_{16} (000_8) in bits 0 through 7.

Approximate execution time: 3.3 microseconds.

3.82 *RDP*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0000_2 00_8 0_{16}	0001_2 01_8 1_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE						PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY				DEVICE ADDRESS 5 BITS + ODD PARITY					
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a maintenance read data from parallel channel defined in R9 into R11:

- Load data from the device defined in R9 (15 through 0) into R11.
- This is the maintenance (nonnormal) read instruction.
- Parity is not checked; also bad parity is not reported.
- The CF is set to:

  (a) One if no errors are detected.

  (b) Zero if:

  —A 10-microsecond device time-out occurs.

  —The device reports an error.
- The correct parity bits for the received data in R11 are generated by the processor.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the RDP option is indicated by the 01_{16} (001_8) in bits 0 through 7.

Approximate execution time: 3.3 microseconds.

3.83 *WD*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0010_2 02_8 2_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction effects a transmit (write) data from R10 to the device defined in R9 via parallel channel:

- Transmit the data in R10 bits (15 through 0) to the device defined in R9 (15 through 0).
- This is the normal write instruction.
- The CF is set to:

(a) One if:
—No errors are detected.

(b) Zero if:

—A 10-microsecond device time-out occurs.

—The device reports an error.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the WD option is indicated by 20_{16} (040_8) in bits 0 through 7.

Approximate execution time: 3.0 microseconds.

3.84 *SDC*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0111_2 07_8 7_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction sends a command in R10 to the device defined in R9 via the parallel channel:

- Transmit the data in R10 (15 through 0) to the device defined in R9 (15 through 0). Assert the CP control lead. Note that R10 (0) must be a "zero" for commands to a peripheral device. R10 (0) set to a "one" is used only for DBS commands.

- The CF is set to:

  (a) One if no errors are detected.

  (b) Zero if:

  —A 10-microsecond device time-out occurs.

  —The device reports an error.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SDC option is indicated by the 70_{16} (160_8) in bits 0 through 7.

Approximate execution time: 3.15 microseconds.

3.85 *SS*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0011_2 03_8 3_{16}	0010_2 02_8 2_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction effects a normal sense status of device defined by R9 via parallel and store in R11:

- Load status received from the device defined in R9 (15 through 0) into R11 (15 through 0). Parity is checked and regenerated.

- The CF is set to:

  (a) One if no errors are detected.

  (b) Zero if:

  —A 10-microsecond device time-out occurs.

  —The device reports an error.

  —Bad parity on the received status.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SS option is indicated by the 32_{16} (062_8) in bits 0 through 7.

Approximate execution time: 3.75 microseconds.

3.86 *SSP*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0011_2 03_8 3_{16}	0011_2 03_8 3_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY
17	16	15 14 13 12 11 10	9 8 7 6	5 4 3 2 1 0

Function: This instruction effects a maintenance sense status of device defined by R9 via the parallel channel and store in R11:

- Load status received from the device defined in R9 (15 through 0) into R11 (15 through 0).
- Parity is regenerated but errors are not reported.
- The CF is set to:

(a) One if no errors are detected.

(b) Zero if:

—A 10-microsecond device time-out occurs.

—The device reports an error.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SSP option is indicated by the 33_{16} (063_8) in bits 0 through 7.

Approximate execution time: 3.75 microseconds.

3.87 *SSA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0011_2 03_8 3_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction effects a maintenance sense status of device defined by R9 via the parallel channel and store in R11:

- Load status received from the device defined in R9 (15 through 0) into R11 (15 through 0). Parity checked and regenerated.
- The received device address R11 (5 through 0) is compared to the transmitted device address R9 (5 through 0).
- The CF is set to:

(a) One if no errors are detected.

(b) Zero if:

—A 10-microsecond device time-out occurs.

—The device reports an error.

—Bad parity is received.

—Device address match fails.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SSA option is indicated by the 30_{16} (060_8) in bits 0 through 7.

Approximate execution time: 4.8 microseconds.

3.88 *SSPA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0011_2 03_8 3_{16}	0001_2 01_8 1_{16}

R9 FORMAT

<table>
<tr><td>P_H</td><td>P_L</td><td colspan="6">MAIN CHANNEL SELECT 3/6 CODE</td><td colspan="4">PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY</td><td colspan="6">DEVICE ADDRESS 5 BITS + ODD PARITY</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a maintenance sense status of device defined in R9 via the parallel channel, store in R11, and check device address:

- Load status received from the device defined in R9 (15 through 0) into R11 (15 through 0).
- Parity is regenerated but errors are not reported.
- The received device address R11 (5 through 0) is compared to the transmitted device address R9 (5 through 0).
- The CF is set to:

(a) One if no errors are detected.

(b) Zero if:

—A 10-microsecond device time-out occurs.

—The device reports an error.

—Device address match fails.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SSPA option is indicated by the 31_{16} (061_8) in bits 0 through 7.

Approximate execution time: 4.8 microseconds.

3.89 *TBSY*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	0001_2 01_8 1_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction sets the CF to condition of device defined by R9:

- The device defined by R9 (15 through 0) is interrogated.
- The CF is set to:

(a) One if no errors are detected and the device reports busy.

(b) Zero if the device is not busy OR the device reports errors.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the TBSY option is indicated by the 10_{16} (020_8) in bits 0 through 7.

Approximate execution time: 3.3 microseconds.

3.90 *IDBS*

Format:

PH	PL	BA	OP CODE 11011100_2 154_8 $6C_{16}$	1010_2 12_8 A_{16}	0000_2 00_8 0_{16}

R9 FORMAT

PH	PL	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY
17	16	15 14 13 12 11 10	9 8 7 6	5 4 3 2 1 0

Function: This instruction initializes all the DBSs on the parallel channel defined by R9:

- The INIT lead associated with the parallel subchannel defined in R9 (15 through 0) is asserted. All DBSs on the subchannel assume the "device on" state.
- The CF is set to:

(a) One if no errors are detected.

(b) Zero if:

—The parallel channel is controlled by the DMA and the DMA fails to release.

—DBS reports errors.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the IDBS option is indicated by the $A0_{16}$ (240_8) in bits 0 through 7.

Approximate execution time: 2.1 microseconds.

3.91 *IID*

Format:

P_H	P_L	BA	OP CODE 11011100_2 154_8 $6C_{16}$	1001_2 11_8 9_{16}	0000_2 00_8 0_{16}

R9 FORMAT

<table>
<tr><td>PH</td><td>PL</td><td colspan="6">MAIN CHANNEL SELECT 3/6 CODE</td><td colspan="4">PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY</td><td colspan="6">DEVICE ADDRESS 5 BITS + ODD PARITY</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction identifies the interrupting device:

- The interrupt status of all the peripherals associated with the subparallel channel defined in R9 (15 through 0) are interrogated.

- Each device interrupt status is reported via a preassigned INF lead (one for each of the 16 devices). The resulting 16-bit word is complemented and stored in R11 — a "0" identifies an interrupting device. The interrupt flag in the subparallel channel (SPCH) is reset by the IID. The interrupt flags in the peripheral devices themselves are not cleared.

- The CF is set to:

  (a) One if no errors are detected.

  (b) Zero if the parallel channel is controlled by the DMA and the DMA fails to release or if any device reports errors.

- Proper parity is generated.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the IID option is indicated by the 90_{16} (220_8) in bits 0 through 7.

Approximate execution time: 3.3 microseconds.

3.92 *CMC N*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	1011_2 13_8 B_{16}	N

R9 FORMAT

P_H	P_L	MAIN CHANNEL SELECT 3/6 CODE	PARALLEL SUBCHANNEL SELECT 3 BITS + ODD PARITY	DEVICE ADDRESS 5 BITS + ODD PARITY
17	16	15 14 13 12 11 10	9 8 7 6	5 4 3 2 1 0

Function: This instruction issues a maintenance channel N to the parallel channel defined in R9: (for maintenance programmers only).

- Issue a maintenance command to the parallel channel as defined in R9 bits (15 through 0).

N (Command Code)	ACTION
0	Send pulse MD0—not used
1	Send pulse MD1—clear channel
2	Send pulse MD2—initialize channel, remove control signal
3	Send pulse MD3—acknowledge interrupt, command strobe
4	Send pulse MD4—sense status
5	Send pulse MD5—read data
6	Send pulse MD6—data present (write)
7	Send pulse MD7—command present
8	Set CF to equal SYNC bit
9	Set CF to equal BUSY bit
A	Set CF to equal ER bit
B	Load INF (15 through 0) into R11 with regenerated parity

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the CMC option is indicated by the $0B_{16}$ (013_8) in bits 4 through 7.

Approximate execution time: 1.20 microseconds.

F. Direct Memory Access (DMA) Instructions

3.93 *MDMA*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101100_2.
154_8
$6C_{16}$</td><td colspan="4">0101_2
05_8
5_{16}</td><td colspan="4">1010_2
12_8
A_{16}</td></tr>
</table>

R9 FORMAT

<table>
<tr><td>P_H</td><td>P_L</td><td colspan="6">DMA MAIN
CHANNEL ADDRESS</td><td colspan="4">1111_2
17_8
F_{16}</td><td>*
P</td><td>0</td><td colspan="4">DEVICE
ADDRESS</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a read write memory:

- If bits (3 through 0) of the DMA parameter register are loaded with a read command:

  Load the contents of the memory location specified by the present address low (PAL) or present address high (PAH) registers into the DMA data buffer register.

- If bits (3 through 0) of the DMA parameter register are loaded with a write command:

  Store the contents of the DMA data buffer register into the memory location specified by the PAL or PAH.

- The device address in bits (3 through 0) of R9 specifies which word of the PAL, PAH, PARM, and DTB registers is used for the instruction.

- Set the CF to zero if it takes the DMA too long to respond. Otherwise set CF to one.

* P = Bit for odd parity over bits (5 through 0).

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the MDMA option is indicated by the $5A_{16}$ (132_8) in bits 0 through 7.

Approximate execution time: 3.0 + X microseconds
$0 \leq X \leq 3.6$ microseconds.

3.94 *WDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	SEE TABLE

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS	SEE TABLE	* P	1	SEE TABLE

17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Function: This instruction effects a write DMA register:

- Load the specified DMA register from R10.

INTRODUCTION BITS (7-0)	DMA REGISTER		R9 (9-6)	R9 (5)	R9 (4)	R9 (3-0)
51_{16} (121_8)	PAL	Present address low	3_{16} (03_8)	*	1	Device address
54_{16} (124_8)	PAH	Present address high	6_{16} (06_8)	*	1	Device address
51_{16} (121_8)	FAL	Final address low	C_{16} (14_8)	*	1	Device address
54_{16} (124_8)	FAH	Final address high	9_{16} (11_8)	*	1	Device address
51_{16} (121_8)	DTB	Data buffer	A_{16} (12_8)	*	1	Device address
51_{16} (121_8)	MDT	Memory data	5_{16} (05_8)	0	1	3_{16} (03_8)
51_{16} (121_8)	EDT	Error data	5_{16} (05_8)	0	1	6_{16} (06_8)
51_{16} (121_8)	MSK	Mask	5_{16} (05_8)	0	1	C_{16} (14_8)
40_{16} (100_8)	CTL	Control	5_{16} (05_8)	0	1	9_{16} (11_8)
51_{16} (121_8)	STS	Status	5_{16} (05_8)	0	1	A_{16} (12_8)
40_{16} (100_8)	STE	State	5_{16} (05_8)	0	1	5_{16} (05_8)

- The DMA register is read back and matched with R10. A mismatch causes a DML mismatch interrupt.

- Set CF to zero if it takes DMA too long to respond. Otherwise set CF to one.

- R9 and R11 are changed by this instruction.

* P = Bit for odd parity over bits (5 through 0).

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the selected WDMA option is indicated by the octal values in bits 0 through 7, as shown in the above table.

Approximate execution time: $6C51_{16}$ (66121_8) 5.55 + 2 X microseconds
$6C54_{16}$ (66124_8) 6.0 + 2 X microseconds
$6C40_{16}$ (66100_8) 1.35 microseconds
$0 \le X \le 3.6$ microseconds.

3.95 *RDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	SEE TABLE

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS	SEE TABLE	* P	1	SEE TABLE
17	16	15 14 13 12 11 10	9 8 7 6	5	4	3 2 1 0

Function: This instruction effects a read DMA register:

- Read the specified DMA register into R11.

INTRODUCTION BITS (7-0)	DMA REGISTER		R9			
			(9-6)	(5)	(4)	(3-0)
56_{16} (126_8)	PAL	Present address low	3_{16} (03_8)	*	0	Device address
56_{16} (126_8)	PAH	Present address high	6_{16} (06_8)	*	0	Device address
56_{16} (126_8)	FAL	Final address low	C_{16} (14_8)	*	0	Device address
56_{16} (126_8)	FAH	Final address high	9_{16} (11_8)	*	0	Device address
57_{16} (127_8)	DTB	Data buffer	A_{16} (12_8)	*	0	Device address
57_{16} (127_8)	MDT	Memory data	5_{16} (05_8)	1	0	3_{16} (03_8)
57_{16} (127_8)	EDT	Error data	5_{16} (05_8)	1	0	6_{16} (06_8)
56_{16} (126_8)	MSK	Mask	5_{16} (05_8)	1	0	C_{16} (14_8)
41_{16} (101_8)	CTL	Control	5_{16} (05_8)	1	0	9_{16} (11_8)
41_{16} (101_8)	STS	Status	5_{16} (05_8)	1	0	A_{16} (12_8)
41_{16} (101_8)	STE	State	5_{16} (05_8)	1	0	5_{16} (05_8)

- Parity error causes a data bus parity error interrupt.
- Set CF to zero if it takes DMA too long to respond. Otherwise set CF to one.

* P = Bit for odd parity over bits (5 through 0).

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the selected WDMA option is indicated by the octal values in bits 0 through 7, as shown in the above table.

Approximate execution time: $6C56_{16}$ (66126_8) 3.6 + X microseconds
$6C57_{16}$ (66127_8) 3.6 + X microseconds
$6C41_{16}$ (66101_8) 1.2 microseconds
$0 \leq X \leq 3.6$ microseconds.

3.96 *TDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$							0101_2 05_8 5_{16}				1000_2 10_8 8_{16}			

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS						0000_2 00_8 0_{16}				0	1	0000_2 00_8 0_{16}			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a test DMA decoder:

- R10 should contain a nonzero constant.
- Test the decoder for reading and writing the DMA register.
- Set the CF to zero if the test failed or if it takes too long for the DMA to respond. Otherwise set CF to one.
- R9 and R11 are changed by this instruction.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the TDMA option is indicated by the 58_{16} (130_8) in bits 0 through 7.

Approximate execution time: 5.55 + 2 X microseconds
$0 \leq X \leq 3.6$ microseconds.

3.97 *IDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	1011_2 13_8 B_{16}	0000_2 00_8 0_{16}

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS						*	*	*	*	*	*	*	*	*	*
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects an initialize DMA:

- DMA is put into the inactive state.
- The DMA control, status, state, and mask registers are cleared.
- All internal control flip-flops are cleared.
- The my bus control and other bus control circuits are initialized.
- A grand reset signal is generated to the PCH in the DMA.
- All DMAR signal receivers are cleared.
- The counter in the add-on circuit is cleared.

* = Do not care.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the IDMA option is indicated by the $B0_{16}$ (260_8) in bits 0 through 7.

Approximate execution time: 1.2 microseconds.

3.98 *SDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	1011_2 13_8 B_{16}	0101_2 05_8 5_{16}

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS						*	*	*	*	*	*	*	*	*	*
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a step DMA:

- The single-step-mode bit of the control register must be set.
- Step DMA to its next state.
- This instruction is used for maintenance only.
- CF is unchanged.

* = Do not care.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the SDMA option is indicated by the $B5_{16}$ (265_8) in bits 0 through 7.

Approximate execution time: 1.2 microseconds.

3.99 *LDMA*

Format:

P_H	P_L	BA	OP CODE 1101100_2 154_8 $6C_{16}$	1011_2 13_8 B_{16}	0010_2 02_8 2_{16}

R9 FORMAT

P_H	P_L	DMA MAIN CHANNEL ADDRESS	*	*	*	*	*	*	*	*	*
17	16	15 14 13 12 11 10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a lock DMA:

- DMA is put into the inactive state.
- The state register is cleared.
- This instruction is used for maintenance only.
- CF is unchanged.

* = Do not care.

Options and requirements: Since other instructions use OP CODE $6C_{16}$ (154_8), the LDMA option is indicated by the $B2_{16}$ (262_8) in bits 0 through 7.

Approximate execution time: 1.2 microseconds.

G. Communications Instructions

3.100 *LCC Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character:

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.
- Increment character count.
- No block check.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCC option is indicated by the 0_{16} (00_8) in bits 0 through 3 of the first word.

Approximate execution time: 16.20 microseconds.

3.101 *LCS Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0001_2
01_8
1_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character and check status:

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8])

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- No block check.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCS option is indicated by the 1_{16} (01_8) in bits 0 through 3 of the first word.

Approximate execution time: 16.80 microseconds.

3.102 *LCX Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0010_2
02_8
2_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with translation:

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- No block check.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCX option is indicated by the 2_{16} (02_8) in bits 0 through 3 of the first word.

Approximate execution time: 19.95 microseconds.

3.103 *LCB Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0011_2
03_8
3_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with status and translation:

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.
- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- No block check.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCB option is indicated by the 3_{16} (03_8) in bits 0 through 3 of the first word.

Approximate execution time: 20.55 microseconds.

3.104 *LCC1 Rx,K(RPx)*

Format:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P$_H$	P$_L$	BA	OP CODE 1101110$_2$ 156$_8$ 6E$_{16}$							RP$_X$				0100$_2$ 04$_8$ 4$_{16}$			
P$_H$	P$_L$	K												R$_X$			

Function: This instruction effects a load character and check (CRC1):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (CRC1).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE 6E$_{16}$ (156$_8$), the LCC1 option is indicated by the 4$_{16}$ (04$_8$) in bits 0 through 3 of the first word.

Approximate execution time: 29.95 microseconds.

3.105 *LCS1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0101_2
05_8
5_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with status and check (CRC1):

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) $\neq$ (EA [15 through 8]).

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (CRC1).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCS1 option is indicated by the 5_{16} (05_8) in bits 0 through 3 of the first word.

Approximate execution time: 29.85 microseconds.

3.106 *LCX1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0110_2
06_8
6_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with translation and check (CRC1):

- See Appendix 5.

- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.

- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).

- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).

- Update block check (CRC1).

- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCX1 option is indicated by the 6_{16} (06_8) in bits 0 through 4 of the first word.

Approximate execution time: 33.00 microseconds.

3.107 *LCB1 Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101110_2 156_8 $6E_{16}$							RP_X				0111_2 07_8 7_{16}			
P_H	P_L	K												R_X			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a load character with status, translation, and check (CRC1):

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- Update block check (CRC1).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCB1 option is indicated by the 7_{16} (07_8) in bits 0 through 3 of the first word.

Approximate execution time: 33.60 microseconds.

3.108 *LCCL Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1000_2
10_8
8_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">R_X</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character and check (LRC):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8) the LCCL option is indicated by the 8_{16} (10_8) in bits 0 through 3 of the first word.

Approximate execution time: 18.75 microseconds.

3.109 *LCSL Rx,K(RPx)*

Format:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P_H	P_L	BA	OP CODE 1101110_2 156_8 $6E_{16}$							RP_X				1001_2 11_8 9_{16}			
P_H	P_L	K												R_X			

Function: This instruction effects a load character with status and check (LRC):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCSL option is indicated by the 9_{16} (11_8) in bits 0 through 3 of the first word.

Approximate execution time: 19.35 microseconds.

3.110 *LCXL Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101110_2 156_8 $6E_{16}$							RP_X				1010_2 12_8 A_{16}			
P_H	P_L	K												R_X			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a load character with translation and check (LRC):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.
- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCXL option is indicated by the A_{16} (12_8) in bits 0 through 3 of the first word.

Approximate execution time: 22.50 microseconds.

3.111 *LCBL Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1011_2
13_8
B_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with status, translation, and check (LRC):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) $\neq$ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCBL option is indicated by the B_{16} (13_8) in bits 0 through 3 of the first word.

Approximate execution time: 23.10 microseconds.

3.112 *LCC2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1100_2
14_8
C_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character and check (CRC2):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]; = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCC2 option is indicated by the C_{16} (14_8) in bits 0 through 3 of the first word.

Approximate execution time: 29.25 microseconds.

3.113 *LCS2 Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101110_2 156_8 $6E_{16}$							RP_X				1101_2 15_8 D_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a load character with status and check (CRC2):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCS2 option is indicated by the D_{16} (15_8) in bits 0 through 3 of the first word.

Approximate execution time: 29.85 microseconds.

3.114 *LCX2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1110_2
16_8
E_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with translation and check (CRC2):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4])
- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.
- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8), the LCX2 option is indicated by the E_{16} (16_8) in bits 0 through 3 of the first word.

Approximate execution time: 33.00 microseconds.

3.115 *LCB2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101110_2
156_8
$6E_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1111_2
17_8
F_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a load character with status, translation, and check (CRC2):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Load Rx (7 through 0) with:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of Rx (15 through 8) remains unchanged.

- Increment character count.
- Add contents of Rx (7 through 0) from Step 3 to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- Test the contents fetched in Step 5. If bit 15 = 0, it is a control character; branch to control address. If bit 15 = 1, place bits 7 through 0 into Rx (7 through 0).
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6E_{16}$ (156_8) the LCB2 option is indicated by the F_{16} (17_8) in bits 0 through 3 of the first word.

Approximate execution time: 33.60 microseconds.

3.116 *SCC Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP_X				0000_2 00_8 0_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character:

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.
- No block check.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8) the SCC option is indicated by the 0_{16} (00_8) in bits 0 through 3 of the first word.

Approximate execution time: 18.30 microseconds.

3.117 *SCS Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0001_2
01_8
1_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status:

- See Appendix 5.

- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) $\neq$ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [18 through 4]) = (EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.

- No block check.

- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCS option is indicated by the 1_{16} (01_8) in bits 0 through 3 of the first word.

Approximate execution time: 18.90 microseconds.

3.118 *SCX Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP$_X$				0010_2 02_8 2_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character with translation:

- See Appendix 5.
- Add contents of Rx (7 through 0) to translation table address (TTA) then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 2) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte is unchanged.
- Increment character count.
- No block check.
- Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCX option is indicated by the 2_{16} (02_8) in bits 0 through 3 of the first word.

Approximate execution time: 22.05 microseconds.

3.119 *SCB Rx,K(RPx)*

Format:

<table>
<tr><td>P$_H$</td><td>P$_L$</td><td>BA</td><td colspan="7">OP CODE
1101111$_2$
157$_8$
6F$_{16}$</td><td colspan="4">RP$_X$</td><td colspan="4">0011$_2$
03$_8$
3$_{16}$</td></tr>
<tr><td>P$_H$</td><td>P$_L$</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status and translation:

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx (7 through 0]).
- If translated character (fetch of Step 3) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 9]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- No block check.
- Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE 6F$_{16}$ (157$_8$), the SCB option is indicated by the 3$_{16}$ (03$_8$) in bits 0 through 3 of the first word.

Approximate execution time: 22.65 microseconds.

3.120 *SCC1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0100_2
04_8
4_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character and check (CRC1):

- See Appendix 5.

- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.

- Update block check (CRC1).

- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8) the SCC1 option is indicated by the 4_{16} (04_8) in bits 0 through 3 of the first word.

Approximate execution time: 31.35 microseconds.

3.121 *SCS1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0101_2
05_8
5_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status and check (CRC1):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.
- Update block check (CRC1).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8) the SCS1 option is indicated by the 5_{16} (05_8) in bits 0 through 3 of the first word.

Approximate execution time: 31.95 microseconds.

3.122 *SCX1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">0110_2
06_8
6_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with translation and check (CRC1):

- See Appendix 5.
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 3) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (CRC1). Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCX1 option is indicated by 6_{16} (06_8) in bits 0 through 3 of the first word.

Approximate execution time: 35.10 microseconds.

3.123 *SCB1 Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP_X				0111_2 07_8 7_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character with status, translation, and check (CRC1):

- See Appendix 5.
- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 3) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4])= 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (CRC1). Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCB1 option is indicated by the 7_{16} (07_8) in bits 0 through 3 of the first word.

Approximate execution time: 35.70 microseconds.

3.124 *SCCL Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP$_X$				1000_2 10_8 8_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character and check (LRC):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCCL option is indicated by the 8_{16} (10_8) in bits 0 through 3 of the first word.

Approximate execution time: 20.85 microseconds.

3.125 *SCSL Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1001_2
11_8
9_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status and check (LRC):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.
- Update block check (LRC).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCSL option is indicated by the 9_{16} (11_8) in bits 0 through 3 of the first word.

Approximate execution time: 21.45 microseconds.

3.126 *SCXL Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1010_2
12_8
A_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with translation and check (LRC):

- See Appendix 5.
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 2) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check.
- Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCXL option is indicated by the A_{16} (12_8) in bits 0 through 3 of the first word.

Approximate execution time: 24.60 microseconds.

3.127 *SCBL Rx,K(RPx)*

Format:

<table>
<tr><td>P$_H$</td><td>P$_L$</td><td>BA</td><td colspan="7">OP CODE
1101111$_2$
157$_8$
6F$_{16}$</td><td colspan="4">RP$_X$</td><td colspan="4">1011$_2$
13$_8$
B$_{16}$</td></tr>
<tr><td>P$_H$</td><td>P$_L$</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status, translation, and check (LRC):

- See Appendix 5.

- Set CF to:

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status contents of Rx (15 through 8) ≠ (EA [15 through 8]).

- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).

- If translated character (fetch of Step 3) bit 15 = 0, it is a control character—branch to control address.

- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.

- Update block check.

- Contents of Rx remains unchanged.

- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE 6F$_{16}$ (157$_8$), the SCBL option is indicated by the B$_{16}$ (13$_8$) in bits 0 through 3 of the first word.

Approximate execution time: 25.30 microseconds.

3.128 *SCC2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1100_2
14_8
C_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character and check (CRC2):

- See Appendix 5.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCC2 option is indicated by the C_{16} (14_8) in bits 0 through 3 of the first word.

Approximate execution time: 31.35 microseconds.

3.129 *SCS2 Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP$_X$				1101_2 15_8 D_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character with status and check (CRC2):

- See Appendix 5.
- Set CF to:

  (a) One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).

  (b) Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).

- Store character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.

- Increment character count.
- Update block check (CRC2).
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCS2 option is indicated by the D_{16} (15_8) in bits 0 through 3 of the first word.

Approximate execution time: 31.95 microseconds.

3.130 *SCX2 Rx,K(RPx)*

Format:

P_H	P_L	BA	OP CODE 1101111_2 157_8 $6F_{16}$							RP_X				1110_2 16_8 E_{16}			
P_H	P_L	K												Rx			
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Function: This instruction effects a store character with translation and check (CRC2):

- See Appendix 5.
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 2) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address if character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (CRC2).
- Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCX2 option is indicated by the E_{16} (16_8) in bits 0 through 3 of the first word.

Approximate execution time: 35.01 microseconds.

3.131 *SCB2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1101111_2
157_8
$6F_{16}$</td><td colspan="4">RP_X</td><td colspan="4">1111_2
17_8
F_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a store character with status, translation, and check (CRC2):

- See Appendix 5.
- Set CF to

  One and branch to error address if actual status in Rx (15 through 8) does not equal expected status; contents of Rx (15 through 8) ≠ (EA [15 through 8]).
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx (7 through 0]).
- If translated character (fetch of Step 3) bit 15 = 0, it is a control character—branch to control address.
- Set CF to:

  Zero and branch to error address of character count = character limit (EA + 2 [15 through 4]) = (EA + 4 [15 through 4]).
- Store translated character contents of Rx (7 through 0) into:

  (Word Address [7 through 0]) if (EA + 2 [4]) = 1

  (Word Address [15 through 8]) if (EA + 2 [4]) = 0

  Contents of other byte remains unchanged.
- Increment character count.
- Update block check (CRC2). Contents of Rx remains unchanged.
- Representative examples are shown in Appendix 5.

Options and requirements: Since other instructions use OP CODE $6F_{16}$ (157_8), the SCB2 option is indicated by the F_{16} (17_8) in bits 0 through 3 of the first word.

Approximate execution time: 35.70 microseconds.

3.132 *CRC1 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1110000_2
160_8
70_{16}</td><td colspan="4">RP_X</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a check character (CRC1):

- See Appendix 5.
- Update block check (CRC1). The synchronous data link control (SDLC) polynomial X16+X12+X5+1 is used.
- Block check is at EA rather than EA+6.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE 70_{16} (160_8), the CRC1 option is indicated by the 0_{16} (00_8) in bits 0 through 3 of the first word.

Approximate execution time: 16.05 microseconds.

3.133 *CRC2 Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1110000_2
160_8
70_{16}</td><td colspan="4">RP_X</td><td colspan="4">0001_2
01_8
1_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a check character (CRC2):

- See Appendix 5.
- Update block check (CRC2). The binary synchronous communications (BSC) polynomial X16+X15+X2+X1 is used.
- Block check is at EA rather than EA+6.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE 70_{16} (160_8), the CRC2 option is indicated by the 1_{16} (01_8) in bits 0 through 3 of the first word.

Approximate execution time: 16.05 microseconds.

3.134 *XLAT Rx,K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1110000_2
160_8
70_{16}</td><td colspan="4">RP_X</td><td colspan="4">0010_2
02_8
2_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">Rx</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: This instruction effects a translate character.

- See Appendix 5.
- Add contents of Rx (7 through 0) to translation table address (TTA), then fetch (TTA + Rx [7 through 0]).
- If translated character (fetch of Step 2) bit 15 = 0, it is a control character—branch to control address.
- Load translated character into Rx (7 through 0) contents of Rx (15 through 8) remains unchanged.
- Representative examples are shown in Appendix 5.
- CF is unchanged.

Options and requirements: Since other instructions use OP CODE 70_{16} (160_8), the XLAT option is indicated by the 2_{16} (02_8) in bits 0 through 3 of the first word.

Approximate execution time: 10.35 microseconds.

H. ♦Miscellaneous

3.135 *GETFR K(RPx)*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0110_2
06_8
6_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="12">K</td><td colspan="4">0000_2
00_8
0_{16}</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: Store registers in a stack frame in the following manner:

1. Add the length of the current stack frame (in memory at the address pointed to by RPx) to the current address in RPx to point to the new stack frame.

2. Save K (the length of the new frame) in the first entry of the new frame.

3. Compare K to the upper 12 bits of RPx (remaining stack free space). If not enough space remains, abort the operation by executing an SVC 15 instruction.

4. Save the original contents of RPx in the second and third entries of the new frame.

5. Decrement the upper 12 bits of RPx by K.

6. Increment the frame address portion of RPx by K to point to the first entry of the new frame.

7. Scan each bit of the data word in the instruction (starting with bit 0) and, if set, save the corresponding register in the new frame starting at the fourth entry.

1. ***This instruction is available only on releases G1C and later.***

2. ***The initial value of the length field of RPx should be one less than the stack length because the new frame length is stored on the stack before testing for overflow.***

3. ***Registers RPx and RPx+1 must not be specified within the data word since the values after update are saved. This will cause problems when the frame is released. (It will lock the frame pointer at the current frame.)***

4. ***The value of K must be greater than or equal to the number of registers to be saved; otherwise, part of the frame will be overwritten.♦***

3.136 ♦*RELFR RPx*

Format:

<table>
<tr><td>P_H</td><td>P_L</td><td>BA</td><td colspan="7">OP CODE
1010100_2
124_8
54_{16}</td><td colspan="4">0111_2
07_8
7_{16}</td><td colspan="4">RP_X</td></tr>
<tr><td>P_H</td><td>P_L</td><td colspan="16">DATA</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Function: Restore previously saved registers from the stack frame in the following manner:

1. Save the pointer to the current frame in a temporary location.
2. Point to the previous frame by loading RPx and RPx+1 from the current frame.
3. Scan each bit of the data word in the instruction (starting with bit 0) and, if set, load the register from the stack frame beginning at the fourth entry.

> ***1. This instruction is available only on releases G1C and later.***
>
> ***2. Registers RPx and RPx+1 must not be specified since this will result in the loss of the frame pointer.***♦

4. GLOSSARY

4.01 The following terms and definitions are used in this section to describe the program instruction set.

Address—A number (20 bits) that identifies a location in a storage device or equipment unit.

Binary—Base 2 numbering system, using digits 0 and 1.

Bit (contracted from ***b***inary dig***it***)—The binary unit of information which is represented by one of two possible conditions such as the digits 0 and 1, high potential or low potential, on or off.

Clear—To restore a storage device or area to the "zero" state.

Complement—Put bit positions in opposite state, eg, 0 to 1, 1 to 0.

Conditional Transfer—An instruction which causes the program control to either process the next sequential instruction or to jump to some other specified instruction, depending upon the result of some previous operation or condition.

Decrement—Subtract from.

Exclusive OR—A logic operation for combining binary words in accordance with the following rule: given binary words A and B, the result is a word that has a 1 only in those bit positions in which either A or B but not both have a 1.

$$\begin{array}{lr} \text{eg,} & A = 1100_2 \\ & B = 1010_2 \\ \hline & A \oplus B = 0110_2 \end{array}$$

Flip-Flop—A device capable of assuming two stable states (set or clear), thereby storing a bit of information. It remains in either state until a signal changes it to the other state.

Gate—A circuit which has the ability to produce an output dependent upon specified type or the coincident nature of the input(s).

Hexadecimal—Base 16 numbering system, using the digits 0 through F (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Hold-Get Area—An area in main store used to store the general registers 2 through 15 and the return address. It is used to store information to prevent its loss when the program transfers from one level of subroutine to another.

Immediate Data—Information that is usually part of an instruction and is directly available for use.

Inclusive OR—A logical operation for combining binary words in accordance with the following rule: given binary words A and B, the result is a binary word that has a 1 only in those bit positions in which either A or B or both have a 1.

$$\begin{array}{lr} \text{eg,} & A = 1100_2 \\ & B = 1010_2 \\ \hline & A+B = 1110_2 \end{array}$$

Indexing—Algebraically adding a given number (index) to another number or address.

Insertion Masking—The process of inserting bit information from one location to another under mask. For each bit position containing a 1 in the mask, the corresponding bit position in a register (R_X) is replaced by the result of the logical product (AND) of the mask bit and the corresponding bit of another register (R_Y). For each bit position containing a 0 in the mask, the corresponding bit position of the register (R_X) remains the same. Algebraically, insertion masking can be represented as follows:

$(R_Y \bullet M)+(R_X \bullet M)$

where:

M is Mask
● is an AND function
+ is an OR function
⌉ is a NOT function

Example: (Using 8-bit register for simplicity)

Contents of R_Y	$= 10101010_2$
Mask	$= 00001111_2$
Original Contents of R_X	$= \underline{01010101_2}$
New Contents of R_X (after insertion masking)	$= 01011010_2$

Instruction—One or more words which direct a unit or resource to perform a particular function such as the clearing of a register, the gating of the information from one place to another, etc.

Logical Product (AND)—A logical operation for combining binary words in accordance with the following rule: given binary words A and B, the result is a binary word that has a 1 only in those bit positions in which both A and B have a 1.

$$\text{e g,} \quad \begin{array}{r} A = 1100_2 \\ \underline{B = 1010_2} \\ A \bullet B = 1000_2 \end{array}$$

Mask—A binary word used to specify the bit positions of a word which are to be operated on (0 means no modification; 1 means perform specified operation on that bit position of the word).

Microinstruction—A 32-bit word stored in the microstore of the 3A CC.

Microsecond—One 1 millionth of a second (0.000001 sec).

Microsequence—A set of microinstructions which perform the function indicated by an instruction from memory.

Mnemonic—A combination of letters which convey the essence of the command. Some of these are:

A = Add
B = Branch
C = Compare
COM = Complement
EX = Exchange
G = Get
H = Hold
I = Insert
L = Load

M = Maintenance
N = AND
O = OR
RL = Rotate Left
RR = Rotate Right
ST = Store
T = Test
X = Exclusive OR
Z = Zero

Octal—Base 8 numbering system, using the digits 0 through 7.

Off-Line—A condition in which equipment may or may not be operating correctly but is not called on to perform its primary function.

On-Line—A condition in which equipment is performing its primary function.

Read—To retrieve the information stored in a memory device.

Register—A functionally associated set of memory elements such as flip-flops. A word repository.

Sequential—A manner of action, or operation, of equipment in which instructions are set up in a sequence (following a specific space or time pattern) and are fed consecutively to equipment.

Set—To place a storage device in the "one" state.

Subroutine—A sequence of programmed instructions to perform a particular function which may be common to several programs or called from two or more places in a single program.

Unconditional Transfer—An instruction which causes the program control to jump to a specified instruction.

Word—A set of characters which occupies one location in storage and is treated by the system as a unit.

Write—To insert information into a memory device.

TABLE A

DEFINITIONS OF TERMS USED IN DESCRIPTION OF BASIC INSTRUCTIONS

TERM	MEANING
R_X or R_Y	One of the 16 general registers (R0 through R15) within the 3A CC.
RA	Either the pair of general registers R12 and R13 or the pair of general registers R14 and R15 are used to contain a 20-bit address.
RA = R12	General register address pair of R12 and R13 are used to contain a 20-bit address (bits 3 through 0 of R12 and bits 15 through 0 of R13).
RA = R14	General register address pair of R14 and R15 are used to contain a 20-bit address (bits 3 through 0 of R14 and bits 15 through 0 of R15).
I	Immediate data
OFFSET	8-bit index number
N	4-bit number (used as immediate data or index number)
M	Mask data bits
X	Low eight bits of an address
RS	May be any one of the special registers listed below: DB — Display Buffer SAR — Store Address Register PA — Program Address Register MCHTR — Maintenance Channel Transmit Receive Register (Load Only) AK — Address Mask AI — Address Input ER — Error Register SS — System Status Register MCHB — Maintenance Channel Buffer Register DK — Data Mask DI — Data Input TI — Timer (Read Only) IM — Interrupt Mask IS — Interrupt Set (Read Only) MS — Maintenance State MMSR — Main Memory Status Register (Read Only)

TABLE B

DEFINITIONS OF TERMS USED IN DESCRIPTION OF EXTENDED INSTRUCTIONS

TERM	MEANING
(R)	Contents of register R
(RP)	Contents of register pair RP
[ADDR]	Contents of memory at location ADDR
EA	Effective address (computed address)
B	4-bit quantity
N	4-bit quantity
M	8- or 16-bit mask
Y	20-bit address
YI	20-bit quantity
K	12-bit quantity
X( )	Hexadecimal value within ( ) (anything other than X with parentheses)
EA( )	Bits of parameter before ( )
1234	Decimal value 1234
GR	Specific general register
RA	Base register pair (12 and 13 or 14 and 15)
RP	Any register pair, as 0 and 1, 2 and 3, 4 and 5, 6 and 7, 8 and 9, 10 and 11, 12 and 13, or 14 and 15
RS	Special register
PA	Program address
I	Immediate
OPF	Opcode fill bit

TABLE C

ALPHABETIC LISTING OF BASIC INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
AI R_X , I	2.61	CR R_X , R_Y	2.78
AN R_X , N	2.62	CRM R_X , R_Y , M	2.79
AR R_X , R_Y	2.63	EXR R_X , R_Y	2.35
A1S Y	2.64	FLZ R_X , R_Y	2.87
B Y	2.39	GA	2.56
BC Y	2.42	GN R_X , N	2.58
BCL Y	2.44	HA	2.55
BL Y	2.41	HALT	2.115
BNC Y	2.43	HN R_X , N	2.57
BNCL Y	2.45	ICF R_X , N	2.99
BPAX R_X	2.48	IRM R_X , R_Y , M	2.37
BR N(RA)	2.40	L R_X , N(RA)	2.16
BRX R_X (RA)	2.47	LA R_X , N(RA)	2.17
BSA Y	2.49	LAL R_X , Y, RA	2.15
BSAI X	2.50	LAX R_X , R_Y (RA)	2.19
BTSA	2.51	LI R_X , I	2.20
BTSAG	2.53	LL R_X , Y	2.14
BTSAGN N	2.54	LN R_X , N	2.21
BTSAN N	2.52	LR R_X , R_Y	2.32
BX R_X , Y	2.46	LRM R_X , R_Y , M	2.33
CI R_X , I	2.80	LRS R_X , RS	2.30
CIRM R_X , I, N, M	2.81	LSR RS, R_X	2.31
COFL N(RA)	2.124	LX R_X , R_Y (RA)	2.18
COFL R_X (RA)	2.125	MI	2.116
COFLX R_X , (RA)	2.127	MIS	2.117
COM R_X	2.69	MSTF N(RA)	2.118
COM R_X , R_Y	2.70	MSTFX R_X (RA)	2.119
CONL N(RA)	2.122	NI R_X , I	2.71
CONL R_X (RA)	2.123	NOP	2.126
CONLX R_X (RA)	2.128	NR R_X , R_Y	2.72

TABLE C (Contd)

ALPHABETIC LISTING OF BASIC INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
OI R_X , I	2.73	STM R_X , N(RA), M	2.28
OR R_X , R_Y	2.74	STVM R_X , N(RA)	2.29
PACK RS	2.34	STX R_X , R_Y (RA)	2.26
PIE	2.59	TBN R_X ,B	2.104
RL R_X , R_Y	2.82	TBR R_X , R_Y	2.105
RLN R_X , N	2.85	TBS N(RA), B	2.106
RR R_X , R_Y	2.83	TCC1	2.107
RRN R_X , N	2.84	TCH	2.114
SBN R_X , B	2.91	TIO	2.112
SBR R_X , R_Y	2.92	TMIO	2.113
SBS N(RA), B	2.93	TRPH R_X	2.101
SCF	2.89	TRPL R_X	2.100
SI R_X , I	2.67	TSRPH RS	2.103
SIO	2.109	TSRPL RS	2.102
SMIO	2.110	TZ R_X	2.86
SN R_X , N	2.65	UNPK RS	2.36
SOP	2.90	XI R_X , I	2.75
SR R_X , R_Y	2.66	XR R_X , R_Y	2.76
ST R_X , N(RA)	2.24	ZBN R_X , B	2.95
STA R_X , N(RA)	2.25	ZBR R_X , R_Y	2.96
STAF N(RA)	2.120	ZBS N(RA), B	2.97
STAF R_X (RA)	2.121	ZCF	2.94
STAFX R_X (RA)	2.129	ZIO	2.111
STAL R_X , Y(RA)	2.23	ZOP	2.98
STAX R_X , R_Y (RA)	2.27	ZR R_X	2.77
STL R_X , Y	2.22		

TABLE D

ALPHABETIC LISTING OF EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
AAR R_X, R_Y	3.63	LCS2 R_X, K(RP_X)	3.113
A1SK K(RP_X)	3.64	LCX R_X, K(RP_X)	3.102
A1SX R_X(RP_X)	3.65	LCXL R_X, K(RP_X)	3.110
BAL RP_X, Y	3.53	LCX1 R_X, K(RP_X)	3.106
BALK RP_X, K(RP_Y)	3.54	LCX2 R_X, K(RP_X)	3.114
BALX RP_X, R_X(RP_Y)	3.55	LDK RP_X, K(RP_Y)	3.10
CAR R_X, R_Y	3.67	LDX RP_X, R_X(RP_Y)	3.11
CASR R_X, R_Y	3.68	LDMA	3.99
CMC N	3.92	LMP Y	3.12
CRC1 R_X, K(RP_X)	3.132	LMPK K(RP_X)	3.13
CRC2 R_X, K(RP_X)	3.133	LMPX R_X(RP_X)	3.14
GETFR K(RP_X)	3.135	LRP RP_X, Y	3.16
IDBS	3.90	LRPAK RP_X, K(RP_Y)	3.21
IDMA	3.97	LRPAX RP_X, R_X(RP_Y)	3.22
IID	3.91	LRPI RP_X, YI	3.20
LALL Y	3.06	LRPK RP_X, K(RP_Y)	3.17
LALLI Y	3.07	LRPX RP_X, R_X(RP_Y)	3.18
LB R_X, RP_X	3.08	LXAK R_X, K(RP_X)	3.23
LBA R_X, RP_X	3.09	LXAX R_X, R_Y(RP_X)	3.24
LCB R_X, K(RP_X)	3.103	LXK R_X, K(RP_X)	3.26
LCBL R_X, K(RP_X)	3.111	LXX R_x, R_y (RP_x)	3.27
LCB1 R_X, K(RP_X)	3.107	MBLK RP_X, RP_Y	3.52
LCB2 R_X, K(RP_X)	3.115	MDMA	3.93
LCC R_X, K(RP_X)	3.100	RD	3.81
LCCL R_X, K(RP_X)	3.108	RDMA	3.95
LCC1 R_X, K(RP_X)	3.104	RDP	3.82
LCC2 R_X, K(RP_X)	3.112	RET RP_X	3.57
LCS R_X, K(RP_X)	3.101	RELFR RP_X	3.136
LCSL R_X, K(RP_X)	3.109	RETSN RP_X, N	3.58
LCS1 R_X, K(RP_X)	3.105	RPAK K(RP_X)	3.49

TABLE D (Contd)

ALPHABETIC LISTING OF EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
RPAX $R_X(RP_X)$	3.50	STB R_X, RP_X	3.30
SAR R_X, R_Y	3.69	STBA R_X, RP_X	3.31
SCB R_X, $K(RP_X)$	3.119	STDK RP_X, $K(RP_Y)$	3.32
SCBL R_X, $K(RP_X)$	3.127	STDX RP_X, $R_X(RP_Y)$	3.33
SCB1 R_X, $K(RP_X)$	3.123	STMP Y	3.34
SCB2 R_X, $K(RP_X)$	3.131	STMPK $K(RP_X)$	3.35
SCC R_X, $K(RP_X)$	3.116	STMPX $R_X(RP_X)$	3.36
SCCL R_X, $K(RP_X)$	3.124	STRP Y	3.38
SCC1 R_X, $K(RP_X)$	3.120	STRPAK RP_X, $K(RP_Y)$	3.42
SCC2 R_X, $K(RP_X)$	3.128	STRPAX RP_X, $R_X(RP_Y)$	3.43
SCS R_X, $K(RP_X)$	3.117	STRPK RP_X, $K(RP_Y)$	3.39
SCSL R_X, $K(RP_X)$	3.125	STRPX RP_X, $R_X(RP_Y)$	3.40
SCS1 R_X, $K(RP_X)$	3.121	STXAK R_X, $K(RP_X)$	3.44
SCS2 R_X, $K(RP_X)$	3.129	STXAX R_X, $R_Y(RP_X)$	3.45
SCX R_X, $K(RP_X)$	3.118	STXK R_X, $K(RP_X)$	3.47
SCXL R_X, $K(RP_X)$	3.126	STXX R_X, $R_Y(RP_X)$	3.48
SCX1 R_X, $K(RP_X)$	3.122	SVC N	3.59
SCX2 R_X, $K(RP_X)$	3.130	S1S Y	3.70
SDC	3.84	S1SK $K(RP_X)$	3.71
SDMA	3.98	S1SX $R_X(RP_X)$	3.72
SLL R_X, R_Y	3.74	TBSY	3.89
SLLN R_X, N	3.75	TDMA	3.96
SRL R_X, R_Y	3.76	TSBSK B, $K(RP_X)$	3.78
SRLN R_X, N	3.77	TSBSX B, $R_X(RP_X)$	3.79
SS	3.85	WD	3.83
SSA	3.87	WDMA	3.94
SSP	3.86	XLAT R_X, $K(RP_X)$	3.134
SSPA	3.88	XQT Y	3.60
STALL Y	3.28	XQTEI Y	3.61
STALLI Y	3.29		

TABLE E

ALPHABETIC LISTING OF BASIC AND EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
AAR R_X , R_Y	3.63	CIRM R_X , I, N, M	2.81
AI R_X , I	2.61	CMC N	3.92
AN R_X , N	2.62	COFL N(RA)	2.124
AR R_X , R_Y	2.63	COFL R_X (RA)	2.125
A1SK K(RP_X)	3.64	COFLX R_X (RA)	2.127
A1SX R_X (RP_X)	3.65	COM R_X	2.69
A1S Y	2.64	COM R_X , R_Y	2.70
B Y	2.39	CONL N(RA)	2.122
BAL RP_X , Y	3.53	CONL R_X (RA)	2.123
BALK RP_X , K(RP_Y)	3.54	CONLX R_X (RA)	2.128
BALX RP_X , R_X (RP_Y)	3.55	CR R_X , R_Y	2.78
BC Y	2.42	CRC1 R_X , K(RP_X)	3.132
BCL Y	2.44	CRC2 R_X , K(RP_X)	3.133
BL Y	2.41	CRM R_X , R_Y , M	2.79
BNC Y	2.43	EXR R_X , R_Y	2.35
BNCL Y	2.45	FLZ R_X , R_Y	2.87
BPAX R_X	2.48	GA	2.56
BR N(RA)	2.40	GETFR K(RP_X)	3.135
BRX R_X (RA)	2.47	GN R_X , N	2.58
BSA Y	2.49	HA	2.55
BSAI X	2.50	HALT	2.115
BTSA	2.51	HN R_X , N	2.57
BTSAG	2.53	ICF R_X , N	2.99
BTSAGN N	2.54	IDBS	3.90
BTSAN N	2.52	IDMA	3.97
BX R_X , Y	2.46	IID	3.91
CAR R_X , R_Y	3.67	IRM R_X , R_Y , M	2.37
CASR R_X , R_Y	3.68	L R_X , N(RA)	2.16
CI R_X , I	2.80	LA R_X , N(RA)	2.17

TABLE E (Contd)

ALPHABETIC LISTING OF BASIC AND EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
LAL R_X, Y, RA	2.15	LMPX R_X (RP_X)	3.14
LALL Y	3.06	LN R_X, N	2.21
LALLI Y	3.07	LR R_X, R_Y	2.32
LAX R_X, R_Y (RA)	2.19	LRM R_X, R_Y, M	2.33
LB R_X, RP_X	3.08	LRP RP_X, Y	3.16
LBA R_X, RP_X	3.09	LRPAK RP_X, K(RP_X)	3.21
LCB R_X, K(RP_X)	3.103	LRPAX RP_X, R_X (RP_X)	3.22
LCBL R_X, K(RP_X)	3.111	LRPI RP_X, YI	3.20
LCB1 R_X, K(RP_X)	3.107	LRPK RP_X, K(RP_Y)	3.17
LCB2 R_X, K(RP_X)	3.115	LRPX RP_X, R_X (RP_Y)	3.18
LCC R_X, K(RP_X)	3.100	LRS R_X, RS	2.30
LCCL R_X, K(RP_X)	3.108	LSR RS, R_X	2.31
LCC1 R_X, K(RP_X)	3.104	LX R_X, R_Y (RA)	2.18
LCC2 R_X, K(RP_X)	3.112	LXAK R_X, K(RP_X)	3.23
LCS R_X, K(RP_X)	3.101	LXAX R_X, R_Y RP_X)	3.24
LCSL R_X, K(RP_X)	3.109	LXK R_X, K(RP_X)	3.26
LCS1 R_X, K(RP_X)	3.105	LXX R_X, R_Y (RP_X)	3.27
LCS2 R_X, K(RP_X)	3.113	MBLK RP_X, RP_Y	3.52
LCX R_X, K(RP_X)	3.102	MDMA	3.93
LCXL R_X, K(RP_X)	3.110	MI	2.116
LCX1 R_X, K(RP_X)	3.106	MIS	2.117
LCX2 R_X, K(RP_X)	3.114	MSTF N(RA)	2.118
LDK RP_X, K(RP_Y)	3.10	MSTFX R_X (RA)	2.119
LDX RP_X, R_X (RP_Y)	3.11	NI R_X, I	2.71
LDMA	3.99	NOP	2.126
LI R_X, I	2.20	NR R_X, R_Y	2.72
LL R_X, Y	2.14	OI R_X, I	2.73
LMP Y	3.12	OR R_X, R_Y	2.74
LMPK K(RP_X)	3.13	PACK RS	2.34

TABLE E (Contd)

ALPHABETIC LISTING OF BASIC AND EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
PIE	2.59	SCS2 R_X, $K(RP_X)$	3.129
RD	3.81	SCX R_X, $K(RP_X)$	3.118
RDMA	3.95	SCXL R_X, $K(RP_X)$	3.126
RDP	3.82	SCX1 R_X, $K(RP_X)$	3.122
RET RP_X	3.57	SCX2 R_X, $K(RP_X)$	3.130
RELFR RPx	3.136	SDC	3.84
RETSN RP_X, N	3.58	SDMA	3.98
RL R_X, R_Y	2.82	SI R_X, I	2.67
RLN R_X, N	2.85	SIO	2.109
RPAK $K(RP_X)$	3.49	SLL R_X, R_Y	3.74
RPAX $R_X(RP_X)$	3.50	SLLN R_X, N	3.75
RR R_X, R_Y	2.83	SMIO	2.110
RRN R_X, N	2.84	SN R_X, N	2.65
SAR R_X, R_Y	3.69	SOP	2.90
SBN R_X, B	2.91	SRL R_X, R_Y	3.76
SBR R_X, R_Y	2.92	SRLN R_X, N	3.77
SBS N(RA), B	2.93	SR R_X, R_Y	2.66
SCB R_X, $K(RP_X)$	3.119	SS	3.85
SCBL R_X, $K(RP_X)$	3.127	SSA	3.87
SCB1 R_X, $K(RP_X)$	3.123	SSP	3.86
SCB2 R_X, $K(RP_X)$	3.131	SSPA	3.88
SCC R_X, $K(RP_X)$	3.116	ST R_X, N(RA)	2.24
SCCL R_X, $K(RP_X)$	3.124	STA R_X, N(RA)	2.25
SCC1 R_X, $K(RP_X)$	3.120	STAF N(RA)	2.120
SCC2 R_X, $K(RP_X)$	3.128	STAF R_X (RA)	2.121
SCF	2.89	STAFX R_X (RA)	2.129
SCS R_X, $K(RP_X)$	3.117	STAL R_X, Y(RA)	2.23
SCSL R_X, $K(RP_X)$	3.125	STALL Y	3.28
SCS1 R_X, $K(RP_X)$	3.121	STALLI Y	3.29

TABLE E (Contd)

ALPHABETIC LISTING OF BASIC AND EXTENDED INSTRUCTIONS

INSTRUCTION	PARAGRAPH REFERENCE	INSTRUCTION	PARAGRAPH REFERENCE
STAX R_X, R_Y (RA)	2.27	TBSY	3.89
STB R_X, RP_X	3.30	TCC1	2.107
STBA R_X, RP_X	3.31	TCH	2.114
STDK RP_X, K(RP_Y)	3.32	TDMA	3.96
STDX RP_X, R_X (RP_Y)	3.33	TIO	2.112
STL R_X, Y	2.22	TMIO	2.113
STM R_X, N(RA), M	2.28	TRPH R_X	2.101
STMP Y	3.34	TRPL R_X	2.100
STMPK K(RP_X)	3.35	TSBSK B, K(RP_X)	3.78
STMPX R_X (RP_X)	3.36	TSBSX B, R_X (RP_X)	3.79
STRP Y	3.38	TSRPH RS	2.103
STRPAK RP_X, K(RP_Y)	3.42	TSRPL RS	2.102
STRPAX RP_X, R_X (RP_Y)	3.43	TZ R_X	2.86
STRPK RP_X, K(RP_Y)	3.39	UNPK RS	2.36
STRPX RP_X, R_X (RP_Y)	3.40	WD	3.83
STVM R_X, N(RA)	2.29	WDMA	3.94
STX R_X, R_Y (RA)	2.26	XI R_X, I	2.75
STXAK R_X, K(RP_X)	3.44	XLAT R_X, K(RP_X)	3.134
STXAX R_X, R_Y (RP_X)	3.45	XQT Y	3.60
STXK R_X, K(RP_X)	3.47	XQTEI Y	3.61
STXX R_X, R_Y (RP_X)	3.48	XR R_X, R_Y	2.76
SVC N	3.59	ZBN R_X, B	2.95
S1S Y	3.70	ZBR R_X, R_Y	2.96
S1SK K(RP_X)	3.71	ZBS N(RA), B	2.97
S1SX R_X (RP_X)	3.72	ZCF	2.94
TBN R_X, B	2.104	ZIO	2.111
TBR R_X, R_Y	2.105	ZOP	2.98
TBS N(RA), B	2.106	ZR R_X	2.77

TABLE F

DEFINITION OF ADDRESSABLE SPECIAL REGISTERS*

REG	TO FIELD FUNCTION		FROM FIELD FUNCTION		
0	GB ==> MCMTR	22 BIT PATH	TI	==> GB	18 BIT PATH PL=0; PH=0
1	GB ==> SAR	22 BIT PATH	SAR	==> GB	22 BIT PATH
2	GB ==> PA	22 BIT PATH	PA	==> GB	22 BIT PATH
3	GB ==> MCHB	22 BIT PATH	MCHB	==> GB	22 BIT PATH
4	UNASSIGNED		MMSR	==> GB	14 BIT PATH PL=0; PH=1
5	GB ==> AK	22 BIT PATH	AK	==> GB	22 BIT PATH
6	GB ==> AI	22 BIT PATH	AI	==> GB	22 BIT PATH
7	GB ==> DK	18 BIT PATH	DK	==> GB	18 BIT PATH
8	GB ==> DI	18 BIT PATH	DI	==> GB	18 BIT PATH
9	GB ==> DB	22 BIT PATH	DB	==> GB	22 BIT PATH
10	GB ==> ER	22 BIT PATH	ER	==> GB	22 BIT PATH
11	GB ==> DB IF DISPLAY BIT IN SS IS 1	22 BIT PATH	UNASSIGNED		
12	GB ==> IM	18 BIT PATH	IM	==> GB	18 BIT PATH
13	GB ==> SS_S	A 1 SETS SS	IS	==> GB	18 BIT PATH
14	GB ==> MS	18 BIT PATH	MS	==> GB	18 BIT PATH
15	GB ==> SS_R	A 1 RESETS SS	SS	==> GB	22 BIT PATH PL=CC; PH=CC

* Registers which are accessed with the special register instructions.