

This item is from Box 42,
error on the stamp *

Space Shuttle Model AP-101 C/M Principles of Operation

Prepared Under
P.O. M4J7XMA-483019

30 January 1979

IBM File No. 6246156B

DATE: 12/15/74
P.O. NO.: M4J7XMA-483019
IBM NO.: 6246156

 Federal Systems Division, Owego, New York 13827

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED IN ACCORDANCE WITH THE
WICHITA STATE UNIVERSITY LIBRARIES
POLICY

MS 87-08 Box 42
FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives



CHANGE HISTORY SHEET

| CHANGE NUMBER | SYM | PAGE | REVISION | APPROVED BY | DATE |
|---------------|-----|-------|--|-------------|------|
| Class II | B | i | Nom chg | | |
| Class II | B | ii | New page | | |
| Class II | B | etc | New page | | |
| Class II | B | 2-6 | Nom chg | | |
| Class II | B | 2-7 | Nom chg | | |
| Class II | B | 2-8 | Nom chg | | |
| Class II | B | 2-9 | Nom chg | | |
| EDCP | | | | | |
| 79-001 | B | 2-10 | Add "double words", nom chg | | |
| Class II | B | 2-12 | Nom chg | | |
| Class II | B | 2-13 | Nom chg | | |
| EDCP | | | | | |
| 58-1 | | | | | |
| 59-8, | | | | | |
| 59-15 | B | 2-14 | Add notes | | |
| EDCP | | | | | |
| 58-1, | | | | | |
| 59-8, | | | | | |
| 59-15 | B | 2-15 | Add note on BSR, DSR | | |
| EDCP | | | | | |
| 58-1 | B | 2-16 | Nom chg, added info on storage protect | | |
| EDCP | | | | | |
| 58-1 | B | 2-17 | Add note on PSW | | |
| Class II | B | 2-18 | Nom chg | | |
| EDCP | | | | | |
| 79-001 | B | 2-19 | Add notes on PSW, nom chg | | |
| EDCP | | | | | |
| 79-001 | B | 2-20 | Add notes on interrupts | | |
| EDCP | | | | | |
| 79-001 | B | 2-21 | Add notes on interrupts | | |
| EDCP | | | | | |
| 79-001 | B | 2-22 | Add notes on interrupts | | |
| Class II | B | 2-23 | Nom chg | | |
| EDCP | | 2-25/ | Add notes on system reset, IPL, | | |
| 79-001 | B | 26 | growth | | |
| Class II | B | 3-1 | Nom chg | | |
| Class II | B | 3-2 | Nom chg | | |
| Class II | B | 4-1 | Nom chg | | |
| Class II | B | 4-2 | Nom chg | | |
| Class II | B | 4-3 | Nom chg | | |
| Class II | B | 4-4 | Nom chg | | |
| Class II | B | 4-5 | Nom chg | | |
| Class II | B | 4-6 | Nom chg | | |

| |
|--------|
| NUMBER |
| PAGE |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION FROM THE ARCHIVAL DEPARTMENT

MS 87-08 1 Box 42
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

| IBM CHANGE HISTORY SHEET | | | | | |
|--------------------------|-----|-------------|-------------------------------|-------------|------|
| CHANGE NUMBER | SYM | PAGE | REVISION | APPROVED BY | DATE |
| Class II | B | 4-7 | Nom chg | | |
| Class II | B | 4-8 | Nom chg | | |
| Class II | B | 4-9 | Nom chg | | |
| Class II | B | 4-10 | Nom chg | | |
| Class II | B | 4-14 | Nom chg | | |
| Class II | B | 4-15 | Nom chg | | |
| Class II | B | 4-17 | Nom chg | | |
| Class II | B | 4-18 | Nom chg | | |
| Class II | B | 4-19 | Nom chg | | |
| Class II | B | 4-20 | Nom chg | | |
| Class II | B | 4-21 | Nom chg | | |
| Class II | B | 4-25 | Nom chg | | |
| Class II | B | 4-36 | Nom chg | | |
| Class II | B | 4-27/ 28 | Nom chg | | |
| Class II | B | 5-1 | Nom chg | | |
| EDCP | | | | | |
| 79-001 | B | 5-2 | Chg description of BIX | | |
| Class II | B | 5-3 | Nom chg | | |
| Class II | B | 5-8 | Nom chg | | |
| Class II | B | 5-9 | Nom chg | | |
| Class II | B | 5-10 | Nom chg | | |
| Class II | B | 6-1 | Nom chg | | |
| EDCP | | | | | |
| 79-001 | B | 6-2 | Nom chg, add programming note | | |
| Class II | B | 6-3 | Nom chg | | |
| Class II | B | 6-4 | Nom chg | | |
| Class II | B | 6-5 | Nom chg | | |
| Class II | B | 6-6 | Nom chg | | |
| Class II | B | 6-7 | Nom chg | | |
| Class II | B | 6-8 | Nom chg | | |
| Class II | B | 7-3 | Nom chg | | |
| Class II | B | 7-7 | Nom chg | | |
| Class II | B | 7-11 | Nom chg | | |
| EDCP | | | | | |
| 58-1, 59-12 | B | 7-12 | Add programming note | | |
| Class II | B | 7-13 | Nom chg | | |
| Class II | B | 7-14 | Nom chg | | |
| Class II | B | 7-15 | Nom chg | | |
| Class II | B | 7-16 | Nom chg | | |
| Class II | B | 7-17 | Nom chg | | |

NUMBER

PAGE

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

| IBM CHANGE HISTORY SHEET | | | | | |
|--------------------------|-----|------|---|-------------|------|
| CHANGE NUMBER | SYM | PAGE | REVISION | APPROVED BY | DATE |
| EDCP | | | | | |
| 79-001 | B | 8-1 | Define normalized number | | |
| Class II | B | 8-2 | Nom chg | | |
| Class II | B | 8-3 | Nom chg | | |
| Class II | B | 8-4 | Nom chg | | |
| Class II | B | 8-5 | Nom chg | | |
| Class II | B | 8-6 | Clarify floating point condition code | | |
| Class II | B | 8-7 | Nom chg | | |
| Class II | B | 8-8 | Nom chg | | |
| Class II | B | 8-9 | Nom chg | | |
| Class II | B | 8-10 | Nom chg | | |
| EDCP | | | | | |
| 059-1 | B | 8-11 | Delete compare (Long Operands) | | |
| EDCP | | | | | |
| 059-1 | B | 8-12 | Delete compare (Long Operands) | | |
| EDCP | | | | | |
| 58-1, 59-13 | B | 8-12 | Add programming and condition code note | | |
| EDCP | | | | | |
| 58-1, 59-10 | B | 8-13 | Add programming note | | |
| Class II | B | 8-14 | Nom chg | | |
| EDCP | | | | | |
| 059-2 | B | 8-16 | Chg description of DIVIDE | | |
| EDCP | | | | | |
| 59-5, 59-16 | B | 8-17 | Modify description of DIVIDE (SHORT) | | |
| Class II | B | 8-19 | Nom chg | | |
| Class II | B | 8-20 | Nom chg | | |
| EDCP | | | | | |
| 58-1, 59-11 | B | 8-21 | Add programming note | | |
| EDCP | | | | | |
| 59-4 | B | 8-22 | Add programming note | | |
| EDCP | | | | | |
| 59-4 | B | 8-23 | Chg description of Mid Value Select | | |
| EDCP | | | | | |
| 59-4 | B | 8-24 | Chg description of Mid Value Select | | |
| Class II | B | 8-25 | Nom chg | | |
| Class II | B | 8-26 | Nom chg | | |
| EDCP | | | | | |
| 58-1, 59-14 | B | 8-28 | Add programming note | | |

NUMBER
PAGE

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER OR BY ANY MEANS

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE STATE WILL NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE ARCHIVAL DEPARTMENT

| IBM | | CHANGE HISTORY SHEET | | | |
|-----------------------|-----|----------------------|--------------------------------------|-------------|------|
| CHANGE NUMBER | SYM | PAGE | REVISION | APPROVED BY | DATE |
| EDCP 58-1 | B | 8-30 | Add programming note | | |
| Class II | B | 8-31/ | | | |
| | | 8-32 | Nom chg | | |
| EDCP 58 | B | 9-0 | Add description of DETECT | | |
| Class II | B | 9-1 | Nom chg | | |
| EDCP 58-1, 59-9 | B | 9-3 | Add programming notes | | |
| EDCP 58-1, 59-9 | B | 9-4 | Chg description of MVH | | |
| Class II | B | 9-5 | Add programming notes | | |
| EDCP 79-001 | B | 9-6 | | | |
| Class II | B | 9-7 | | | |
| Class II | B | 9-8 | | | |
| EDCP 79-001 | B | 9-9 | Add note on hardware anomaly | | |
| EDCP 79-001 | B | 9-10 | Add note on hardware anomaly | | |
| EDCP 58-1 | B | 9-11 | Add programming note | | |
| EDCP 58 | B | 10-1 | Chg description of I/O Channel Reset | | |
| EDCP 58-1, 59-3, 59-7 | B | 10-2 | Chg description and programming note | | |
| Class II | B | 11-1/ | | | |
| | | 2 | Nom chg | | |
| Class II | B | 12-1 | Nom chg | | |
| Class II | B | 12-3/ | | | |
| | | 4 | Nom chg | | |
| Class II | B | 13-1 | Nom chg | | |
| Class II | B | 13-2 | Nom chg | | |
| Class II | B | 14-1/ | | | |
| | | 2 | Nom chg | | |

NUMBER
PAGE

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

TABLE OF CONTENTS

| Section | Page |
|---|------|
| 1 INTRODUCTION | 1-1 |
| 2 AP-101 C/M STRUCTURE | 2-1 |
| MAIN STORAGE | 2-1 |
| INFORMATION FORMATS | 2-1 |
| ADDRESSING | 2-1 |
| INFORMATION POSITIONING | 2-2 |
| CENTRAL PROCESSING UNIT | 2-2 |
| PROGRAM ADDRESSABLE REGISTERS | 2-2 |
| FIXED-POINT DATA REPRESENTATION | 2-4 |
| INSTRUCTION FORMATS | 2-4 |
| RR FORMAT INSTRUCTIONS | 2-6 |
| SRS FORMAT INSTRUCTIONS | 2-6 |
| SI INSTRUCTIONS | 2-8 |
| RI INSTRUCTIONS | 2-8 |
| RS FORMAT INSTRUCTIONS | 2-9 |
| EXPANDED ADDRESSING | 2-15 |
| PROGRAM EXECUTION | 2-16 |
| STORAGE PROTECTION FEATURES | 2-16 |
| INSTRUCTION MONITOR FEATURE | 2-17 |
| MACHINE STATUS | 2-17 |
| PROGRAM STATUS WORD | 2-17 |
| INTERRUPTS | 2-19 |
| GENERAL SYSTEM OPERATION | 2-23 |
| 3 CPU I/O | 3-1 |
| DIRECT MEMORY ACCESS OPERATION | 3-1 |
| PROGRAM-CONTROLLED INPUT/OUTPUT OPERATION | 3-1 |
| PROGRAM-CONTROLLED I/O INSTRUCTION | 3-1 |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITER'S PERMISSION THE MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER IN ANY REPOSITORY

MS 87-08 BOX 42 FF 46

Table of Contents (cont)

| Section | | Page |
|---------|--|------|
| 4 | FIXED-POINT ARITHMETIC | 4-1 |
| | ADD | 4-1 |
| | ADD HALFWORD | 4-2 |
| | ADD HALFWORD IMMEDIATE | 4-3 |
| | ADD TO STORAGE | 4-4 |
| | COMPARE | 4-4 |
| | COMPARE BETWEEN LIMITS | 4-5 |
| | COMPARE HALFWORD | 4-6 |
| | COMPARE HALFWORD IMMEDIATE | 4-7 |
| | COMPARE IMMEDIATE WITH STORAGE | 4-8 |
| | DIVIDE | 4-9 |
| | EXCHANGE UPPER AND LOWER HALFWORDS | 4-10 |
| | INSERT ADDRESS LOW | 4-11 |
| | INSERT HALFWORD LOW | 4-12 |
| | LOAD | 4-13 |
| | LOAD ADDRESS | 4-13 |
| | LOAD ARITHMETIC COMPLEMENT | 4-14 |
| | LOAD FIXED IMMEDIATE | 4-15 |
| | LOAD HALFWORD | 4-16 |
| | LOAD MULTIPLE | 4-17 |
| | MODIFY STORAGE HALFWORD | 4-18 |
| | MULTIPLY | 4-18 |
| | MULTIPLY HALFWORD | 4-19 |
| | MULTIPLY HALFWORD IMMEDIATE | 4-20 |
| | MULTIPLY INTEGER HALFWORD | 4-21 |
| | STORE | 4-22 |
| | STORE HALFWORD | 4-22 |
| | STORE MULTIPLE | 4-23 |
| | SUBTRACT | 4-24 |
| | SUBTRACT FROM STORAGE | 4-25 |
| | SUBTRACT HALFWORD | 4-26 |
| | TALLY DOWN | 4-27 |
| 5 | BRANCHING | 5-1 |
| | BRANCH AND LINK | 5-1 |
| | BRANCH AND INDEX | 5-2 |
| | BRANCH ON CONDITION | 5-3 |
| | BRANCH ON CONDITION BACKWARD | 5-4 |
| | BRANCH ON CONDITION (EXTENDED) | 5-5 |
| | BRANCH ON CONDITION FORWARD | 5-6 |
| | BRANCH ON COUNT | 5-7 |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY
SPECIAL COLLECTIONS
WICHITA STATE UNIVERSITY LIBRARIES

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER FOR ANY PURPOSES
SPECIAL COLLECTIONS
WICHITA STATE UNIVERSITY LIBRARIES

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

Table of Contents (cont)

| Section | Page |
|--|------|
| BRANCH ON COUNT BACKWARD | 5-8 |
| BRANCH ON OVERFLOW AND CARRY | 5-8 |
| BRANCH ON OVERFLOW AND CARRY FORWARD | 5-9 |
| 6 SHIFT OPERATIONS..... | 6-1 |
| NORMALIZE AND COUNT | 6-1 |
| SHIFT LEFT LOGICAL | 6-3 |
| SHIFT LEFT DOUBLE LOGICAL | 6-3 |
| SHIFT RIGHT ARITHMETIC | 6-4 |
| SHIFT RIGHT DOUBLE ARITHMETIC | 6-5 |
| SHIFT RIGHT DOUBLE LOGICAL | 6-5 |
| SHIFT RIGHT LOGICAL | 6-6 |
| SHIFT RIGHT AND ROTATE | 6-7 |
| SHIFT RIGHT DOUBLE AND ROTATE..... | 6-7 |
| 7 LOGICAL OPERATIONS..... | 7-1 |
| AND | 7-1 |
| AND HALFWORD IMMEDIATE | 7-2 |
| AND IMMEDIATE WITH STORAGE | 7-3 |
| AND TO STORAGE | 7-4 |
| EXCLUSIVE OR | 7-5 |
| EXCLUSIVE OR HALFWORD IMMEDIATE | 7-6 |
| EXCLUSIVE OR IMMEDIATE WITH STORAGE | 7-7 |
| EXCLUSIVE OR TO STORAGE | 7-7 |
| OR | 7-8 |
| OR HALFWORD IMMEDIATE | 7-9 |
| OR TO STORAGE..... | 7-10 |
| SEARCH UNDER MASK | 7-11 |
| SET BITS | 7-12 |
| SET HALFWORD | 7-13 |
| TEST BITS | 7-14 |
| TEST REGISTER BITS | 7-14 |
| TEST HALFWORD | 7-15 |
| ZERO BITS | 7-16 |
| ZERO REGISTER BITS | 7-17 |
| ZERO HALFWORD..... | 7-17 |
| 8 FLOATING-POINT OPERATIONS | 8-1 |
| DATA FORMAT | 8-1 |
| NUMBER REPRESENTATION | 8-2 |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR BY ANY MEANS ELECTRONIC OR MECHANICAL INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

Table of Contents (cont)

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, NOR FACED IN ANY MANNER.

Section

Page

NORMALIZATION 8-3

FLOATING-POINT SECOND OPERANDS 8-3

FLOATING-POINT REGISTERS 8-3

FLOATING-POINT INSTRUCTIONS 8-5

CONDITION CODE 8-6

INDICATORS 8-6

FLOATING-POINT ARITHMETIC EXCEPTIONS 8-6

ADD (LONG OPERANDS) 8-7

ADD (SHORT OPERANDS) 8-9

COMPARE (SHORT OPERANDS) 8-12

CONVERT TO FIXED-POINT 8-12

CONVERT TO FLOATING-POINT 8-13

DIVIDE (EXTENDED OPERANDS) 8-14

DIVIDE (SHORT OPERANDS) 8-17

LOAD (LONG OPERANDS) 8-19

LOAD (SHORT OPERANDS) 8-20

LOAD COMPLEMENT (SHORT OPERANDS) 8-20

LOAD FIXED REGISTER 8-21

LOAD FLOATING IMMEDIATE 8-21

LOAD FLOATING REGISTER 8-22

MID VALUE SELECT (SHORT OPERANDS) 8-23

MULTIPLY (EXTENDED OPERANDS) 8-24

MULTIPLY (SHORT OPERANDS) 8-26

SUBTRACT (LONG OPERANDS) 8-28

SUBTRACT (SHORT OPERANDS) 8-29

STORE (LONG OPERANDS) 8-30

STORE (SHORT OPERANDS) 8-31

9 SPECIAL OPERATIONS 9-1

 DETECT 9-1

 INSERT STORAGE PROTECT BITS 9-1

 LOAD PROGRAM STATUS 9-3

 MOVE HALFWORD OPERANDS 9-4

 SET PROGRAM MASK 9-6

 SET SYSTEM MASK 9-6

 STACK CALL 9-7

 STACK RETURN 9-9

 SUPERVISOR CALL 9-11

 TEST AND SET 9-12

 TEST AND SET BITS 9-13

10 INTERNAL CONTROL OPERATIONS 10-1

 INTERNAL CONTROL 10-1

MS 87-08 Box 40 42 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

Table of Contents (cont)

| Section | Page |
|---|------|
| 11 EFFECTIVE ADDRESS GENERATION SUMMARY CHART | 11-1 |
| 12 AP-101 C/M INSTRUCTION REPERTOIRE | 12-1 |
| 13 AP-101 C/M OP CODE ASSIGNMENTS | 13-1 |
| 14 AUTOMATIC INDEX ALIGNMENT DESCRIPTION | 14-1 |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR FOR ANY PURPOSES OTHER THAN
 RESEARCH

MS 87-08 1 Box 1042 1 FF 46 1

Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

LIST OF ILLUSTRATIONS

| Figure | | Page |
|--------|---|------|
| 2-1 | Instruction and Operand Bit Numbering | 2-2 |
| 2-2 | General Register Addresses | 2-3 |
| 2-3 | Fixed-Point Operand Formats | 2-4 |
| 2-4 | Basic Instruction Formats | 2-5 |
| 2-5 | The RR Instruction Formats | 2-6 |
| 2-6 | SRS Instruction Format | 2-6 |
| 2-7 | SRS Halfword Addressing | 2-7 |
| 2-8 | SRS Fullword Addressing | 2-7 |
| 2-9 | SI Instructions | 2-8 |
| 2-10 | RI Instructions | 2-8 |
| 2-11 | RS Instruction Formats | 2-9 |
| 2-12 | Displacement Alignment for Extended Addressing | 2-9 |
| 2-13 | Automatic Index Alignment | 2-11 |
| 2-14 | Displacement Alignment for Indexed Addressing | 2-12 |
| 2-15 | The Contents of Indirect Address Storage Modification Word | 2-13 |
| 2-16 | The Contents of Index Register X | 2-13 |
| 2-17 | Fullword Indirect Address Pointer | 2-14 |
| 2-18 | Expanded Addressing | 2-15 |
| 2-19 | PSW Fields | 2-18 |
| 2-20 | Interrupt Codes | 2-20 |
| 2-21 | Preferred Storage Area Assignments | 2-24 |
| 2-22 | CPU Mode Switching | 2-24 |
| | | |
| 6-1 | Shift Count | 6-1 |
| 6-2 | Normalize and Count Execution | 6-2 |
| | | |
| 8-1 | Floating-Point Second Operand in Main Storage | 8-3 |
| 8-2 | Floating-Point Operands in Registers | 8-4 |
| 8-3 | Combinations of Fractional Precision for Floating-Point Operands | 8-4 |
| 8-4 | Condition Code Setting for Floating-Point Arithmetic | 8-6 |
| | | |
| 9-1 | Move Halfword Execution | 9-5 |
| 9-2 | Current STACK Status — Prior to SCAL | 9-7 |
| 9-3 | STACK Status — Upon Completion of SCAL | 9-8 |

Section 1
INTRODUCTION

The AP-101 C/M (description is for AP-101C and AP-101, monolithic version) is a high-speed general-purpose computer intended primarily for real-time applications such as guidance, navigation, control, and data processing. The AP-101 C/M is a member of the advanced System/4 Pi family of digital computers. This family shares and is unified by extensive design experience, proven technology base, and common manufacturing processes.

This Principles of Operation manual provides a direct comprehensive description of the system structure; the arithmetic, logical, branching, and status switching; and the interruption system. This publication defines and describes features common to all AP-101 C/M computers. These features are the basis for IBM-developed support software and are compatible with compiler development efforts now in process.

Execution times and nonstandard features and functions are described in separate documents. This is because aerospace computers characteristically include user defined features such as unique input/output channels, and special discretes. These will be incorporated into the AP-101 C/M as pluggable options. Furthermore, the AP-101 C/M is microprogrammed and is designed to permit incorporation of additional instructions and operations without redesign and requalification. Such extensions are also described separately.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER FOR ANY PURPOSES

MS 87-08 Box 4042 FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

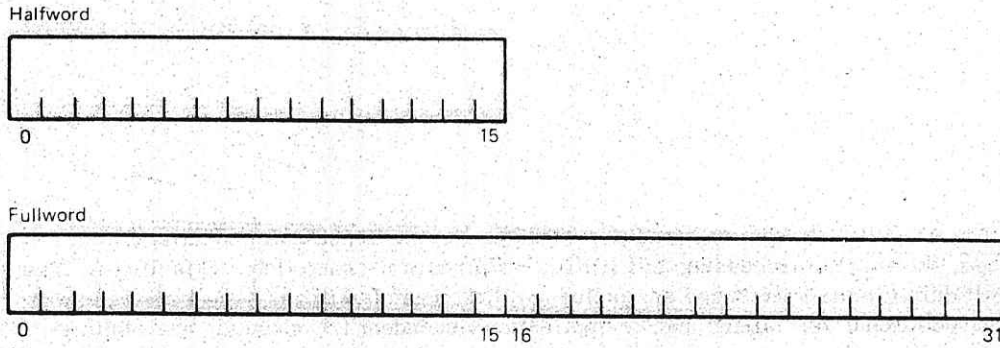


Figure 2-1. Instruction and Operand Bit Numbering

INFORMATION POSITIONING

Fullword operands must be located in main storage on even halfword boundaries. That is, the least significant bit of the operand address, when expressed in binary, must always be zero. Fullword instructions may begin at any address.

CENTRAL PROCESSING UNIT

The central processing unit (CPU) contains facilities for addressing main storage, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the desired order, and for initiating the communication between storage and external devices.

The control section guides the CPU through the functions necessary to execute the program.

PROGRAM ADDRESSABLE REGISTERS

Two sets of eight fixed-point general registers and one set of eight floating-point registers are under explicit program control. The contents of one or more of these registers (32 bits) participate in most CPU operations.

Conceptually, an additional doubleword status register, called the program status word (PSW), is the focal point for machine status. The contents of the PSW are updated during each instruction. Consequently, the PSW reflects current machine status following every instruction. The PSW participates implicitly in status switching, branching operations, and address calculations.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, NOR BE LOANED, REPRODUCED, OR
DISTRIBUTED IN ANY MANNER.

MS 87-08

Box 40 42

FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

In addition to the PSW and the general and floating-point registers, the CPU also contains working registers used for storage addressing, storage buffering, shift and iteration counting, and operand storage. These registers are of no direct concern to the programmer and are not described herein.

The contents of the PSW specify which of the two sets of general registers is in current use. Only the contents of the selected general register set can participate in arithmetic operations and the contents of unselected sets of general registers can not be altered by a program. An alternate set of general registers can be selected by changing the PSW. Only one set of the fixed-point, general-purpose registers and the floating-point registers are available to the program at any one time.

General register contents can be used interchangeably as operands for arithmetic, logical, and shifting operations, or as base and index registers for relative addressing. Each set of general registers is numbered from 0 through 7 and is addressed as shown in Figure 2-2.

| General Register Number | Register Function | | |
|-------------------------|-------------------|------------|-------|
| | Operand | Base | Index |
| 0 | 000 | 00 | None |
| 1 | 001 | 01 | 001 |
| 2 | 010 | 10 | 010 |
| 3 | 011 | 11 or None | 011 |
| 4 | 100 | | 100 |
| 5 | 101 | | 101 |
| 6 | 110 | | 110 |
| 7 | 111 | | 111 |

Figure 2-2. General Register Addresses

Note that general registers 4 through 7 cannot contain base addresses and that general register 0 cannot contain an index.

For some operations, an even/odd pair of general registers are linked to form a 64-bit doubleword register. The most significant half of a doubleword operand is contained in the even-numbered register; the least significant half of the doubleword in the next higher odd-numbered register. Doubleword operands are addressed by specifying the even numbered address of the register containing the most significant portion of the operand.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17, U.S. CODE)
COPIES PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED
REPRODUCED ANYWHERE ANY MANNER ANY REASON

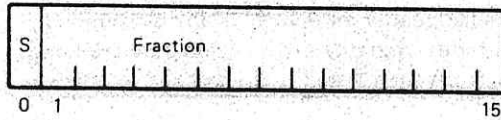
MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FIXED-POINT DATA REPRESENTATION

Data representation is fractional, with negative numbers represented in two's complement form. A halfword operand is 15 bits plus sign; a fullword operand is 31 bits plus sign, as shown in Figure 2-3.

In fractional data representation, the binary point is immediately to the right of the sign.

Fixed-Point Halfword Operand



Fixed-Point Fullword Operand

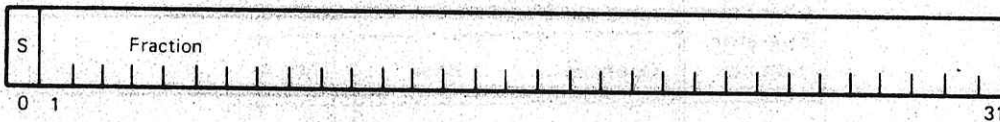


Figure 2-3. Fixed-Point Operand Formats

INSTRUCTION FORMATS

The length of an instruction format can be either one or two halfwords. Long format instructions provide maximum range and extended flexibility for addressing storage operands. Short instructions are used to (1) specify register-to-register operations, and (2) specify storage operands in cases where a small displacement is sufficient and complete address modification capability is not required.

Instruction formats overlap. Programs are written so that in many instances any given operation can be coded using either a halfword or a fullword instruction. In such cases, maximum use of halfword instructions results in increased storage efficiency and performance.

The three basic instruction formats are as shown in Figure 2-4. Halfword instructions are automatically selected by the assembler unless otherwise specified by the programmer.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONSWITHOUT WRITTEN PERMISSION FROM WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

MS 87-08

Box 42

FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

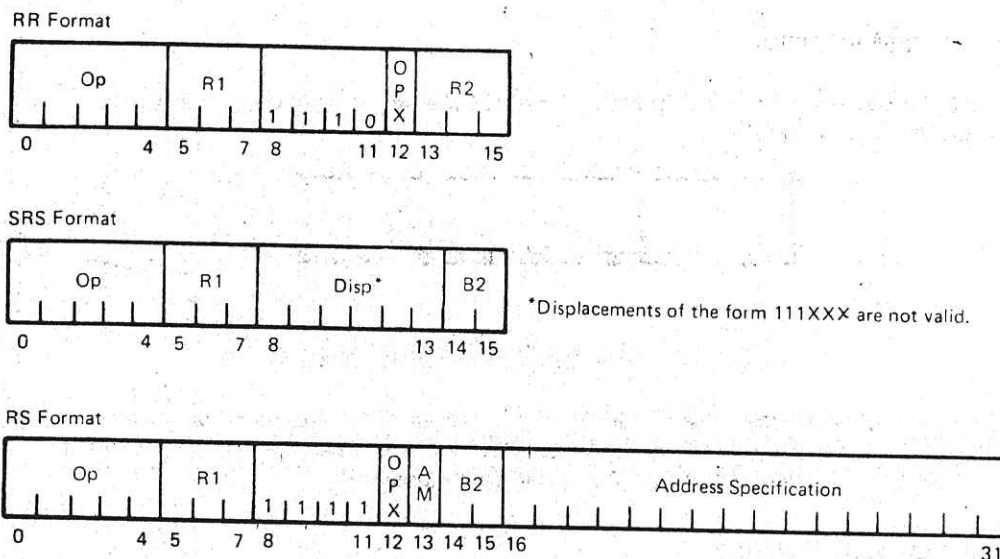


Figure 2-4. Basic Instruction Formats

The fields within the instruction formats usually are used as described below. The exceptions are described in conjunction with the individual formats and instructions.

| | |
|-----------------------|---|
| Op | This 5-bit field defines an operation, or the class of operation, to be performed by the CPU. |
| R1 | This 3-bit field designates the register containing the first operand. Except for operations which alter main storage, the result usually replaces the first operand. |
| R2 | This 3-bit field appears only in the RR format. It is used to specify a general register containing either the second operand or the address of the second operand. |
| B2 | This 2-bit field specifies the register containing the base address. |
| Disp | In halfword SRS format instructions, this 6-bit field is called the displacement. For the SRS format, the displacement is added to the base address specified by the B field to obtain a storage address. |
| OPX | This bit is an extension of the OP field. |
| AM | This field designates one of two fullword format addressing options. |
| Address Specification | The second halfword of a fullword instruction is specified as either extended or indexed addressing. |

RR FORMAT INSTRUCTIONS

The RR format instructions (Figure 2-5) permit the specification of operations that use two general registers.

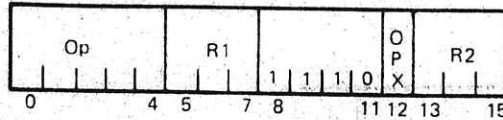
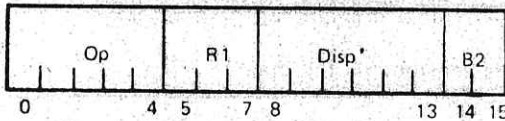


Figure 2-5. The RR Instruction Formats

The operation normally uses as operands the contents of two general registers. The R2 field specifies the second operand while the R1 field specifies the first operand. The result of the operation usually replaces the first operand.

SRS FORMAT INSTRUCTIONS

The SRS instruction format (Figure 2-6) is a compression of the RS format. It provides base plus displacement storage addressing.



* Displacements of the form 111XXX are not valid.

| B2 | Register Containing Base |
|----|--------------------------|
| 00 | General Register 0 |
| 01 | General Register 1 |
| 10 | General Register 2 |
| 11 | General Register 3 |

Figure 2-6. SRS Instruction Format

The R1 field specifies the first operand register address. The 19-bit effective address (EA) of the second operand is developed as follows:

- Step 1 First the positive integer contained in the displacement field is added to the contents of the base contained in the general register specified by B2.

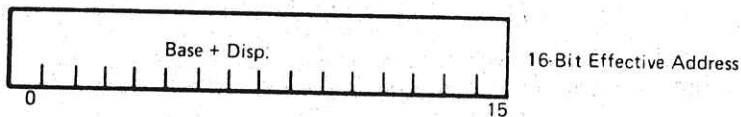
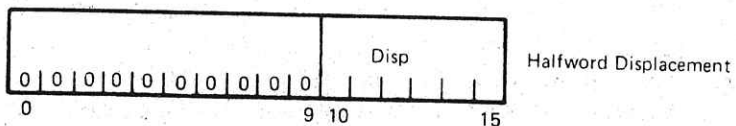
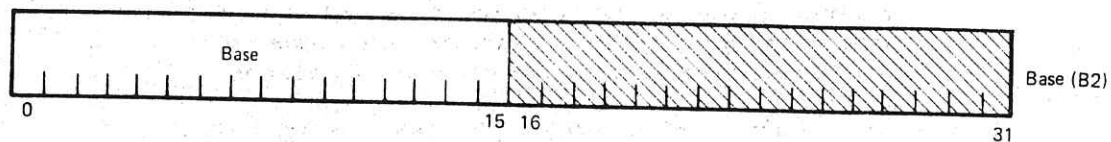
When addressing halfword operands, the least significant bit of the displacement field (instruction bit 13) is aligned with base register bit 15. The 16-bit result is the sum of the base and the displacement, aligned as shown in Figure 2-7.

When addressing fullwords operands using the SRS format, the least significant bit of the displacement field is aligned with base register bit 14 as shown in Figure 2-8.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE LIBRARY

MS 87-08 BOX 42 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives




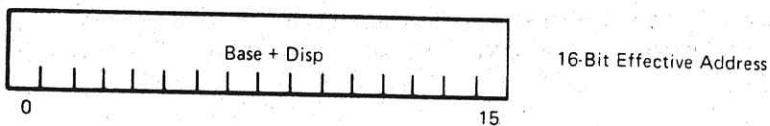
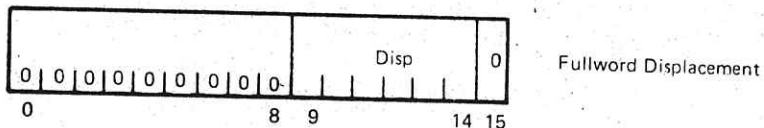
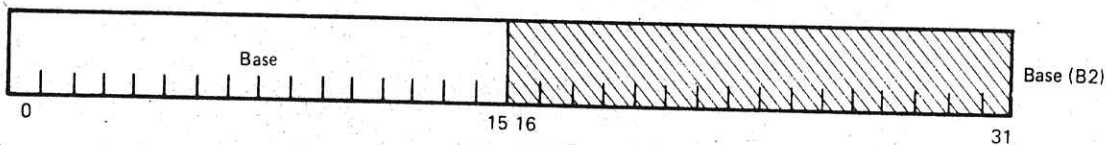
 The low-order half of the general register containing the base does not participate in SRS addressing.

Figure 2-7. SRS Halfword Addressing




 The low order half of the general register containing the base does not participate in SRS addressing.

Figure 2-8. SRS Fullword Addressing

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION FROM THE NATIONAL ARCHIVES
 REPRODUCED IN ANY FORM OR BY ANY MEANS

MS 87-08 BOX 42 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

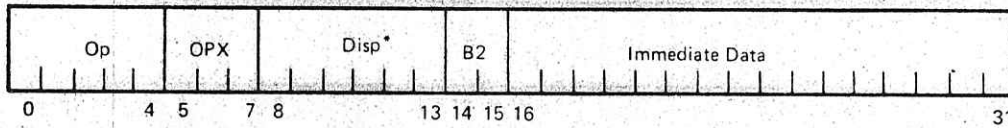
Even though the addition of a base and the fullword displacement in a halfword address, bit 15 is ignored when addressing fullword second operands. As a result, the same fullword address is obtained regardless of the contents of base bit position 15.

- Step 2 The 16-bit result of the addition of the base and displacement is expanded (see Expanded Addressing) to a 19-bit effective address (EA), and this is the address of the second operand.

Except for store instructions, the result of operation between the first operand (the contents of general register R1) and the second operand (the contents of the EA) replace the first operand for SRS format operations. The first operand replaces the second operand for store instructions.

SI INSTRUCTIONS

Direct initialization, modification, and testing of main storage is possible through the use of an immediate data halfword appended to an SRS instruction. See Figure 2-9.



*Displacements of the form 111XXX are not valid.

Figure 2-9. SI Instructions

The address of the halfword second operand is developed in the normal manner for SRS instructions using halfword addressing. Except for test instructions, the result of the operation between the halfword second operand and the immediate data replaces the second operand. The second operand is not altered for test instructions. The first operand is never altered for SI instructions.

RI INSTRUCTIONS

Using an immediate data halfword appended to an RR instruction (Figure 2-10) permits direct initialization, modification, and testing of the most significant 16 bits contained in a general register.

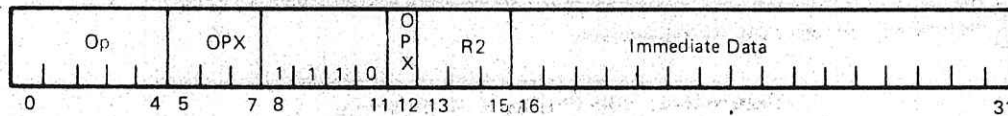


Figure 2-10. RI Instructions

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED
REPRODUCED IN WHOLE OR IN PART FROM ANY SOURCE

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

Except for test instructions, the result of the operation between the second operand and the immediate data replaces the second operand. The second operand is not altered for test instructions. The immediate data first operand is never altered for RI instructions.

RS FORMAT INSTRUCTIONS

There are two major classes of RS instructions, extended and indexed addressing modes, differing in the techniques used to specify the second operand. See Figure 2-11.

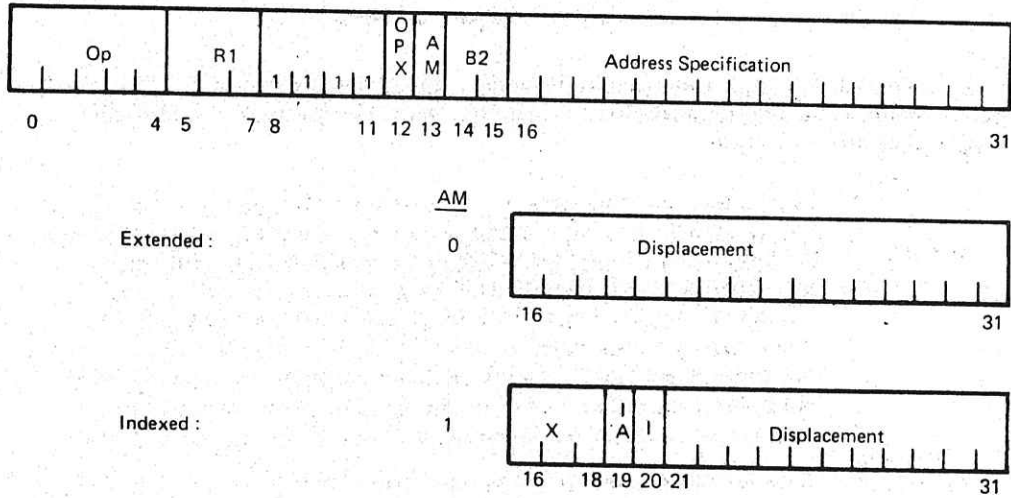


Figure 2-11. RS Instruction Formats

Extended addressing is specified when RS format bit 13 (AM) equals 0. This addressing mode provides a full 16-bit halfword displacement. The base and displacement are aligned as shown in Figure 2-12 when base addressing is performed.

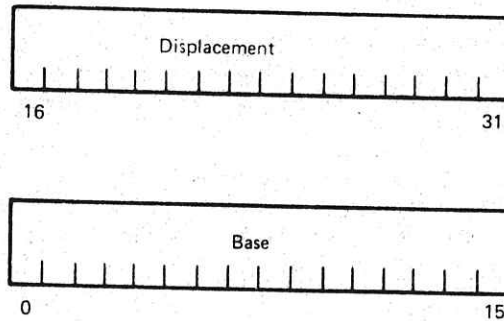


Figure 2-12. Displacement Alignment for Extended Addressing

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED FROM THE ORIGINAL SOURCE COLLECTION
WITH OUT THE PERMISSION OF THE WICHITA STATE UNIVERSITY LIBRARIES

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

Aside from the size and alignment of the displacement, RS extended addressing differs from SRS addressing in two other respects:

- 1) The alignment of the displacement is the same whether addressing double word, fullword or halfword operands.
- 2) When B2 equals 11, base addressing is not performed. In this case the displacement is instead used directly as the address. Then the resulting 16-bit EA is expanded (See Expanded Addressing) to a 19-bit EA. Bit 15 of the operand effective address is always treated as zero when addressing fullword operands.

Indexed addressing is specified by RS format bit 13 (AM) equal to 1. This addressing mode contains three additional fields. Normally, they contribute to the effective address generation as follows:

- | | |
|----|---|
| X | This 3-bit field specifies one of seven general registers containing the index. Indexing is not performed when X is equal to 000. An index is contained in the upper halfword of a general register. The index is automatically aligned as illustrated in Figure 2-13. For additional information on index alignment, see Section 14. Consistent with the restrictions that apply to register usage and indirect addressing, general register contents can be used interchangeably as either a base or an index or both. When indirect addressing is specified, indexing follows indirect addressing. |
| IA | This format bit, when a one, specifies indirect addressing. Indirect addressing is not performed when this bit is zero. |
| I | This format bit, in conjunction with X and IA, specifies various address modes which are explained below. |

The development of the EA for the indexed mode of operand addressing is explained in detail in the subsequent steps:

- 1) Indexed addressing is specified by RS format bit 13 (AM) equal to 1. This addressing mode provides an 11-bit displacement. The base and displacement are aligned as shown in Figure 2-14 when indexed addressing is performed.

The displacement is aligned so that bit 31 corresponds to base or index bit 15 and displacement bit 21 corresponds to base or index bit 5. The displacement is expanded to 16 bits by appending five leading zeros.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

COPY PROVIDED BY

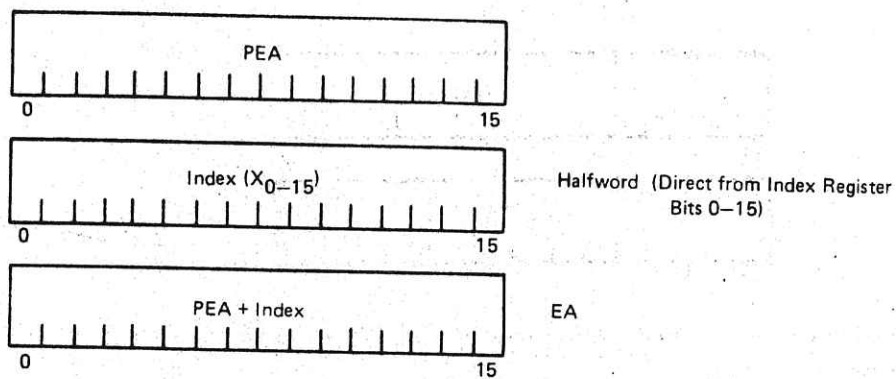
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

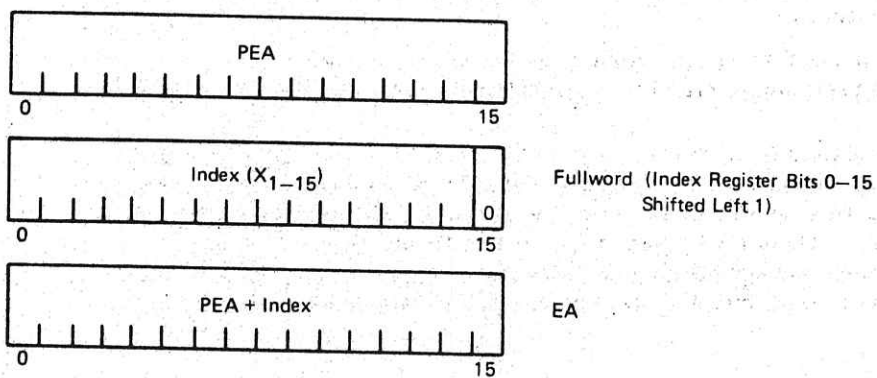
MS 87-08

Box 4042

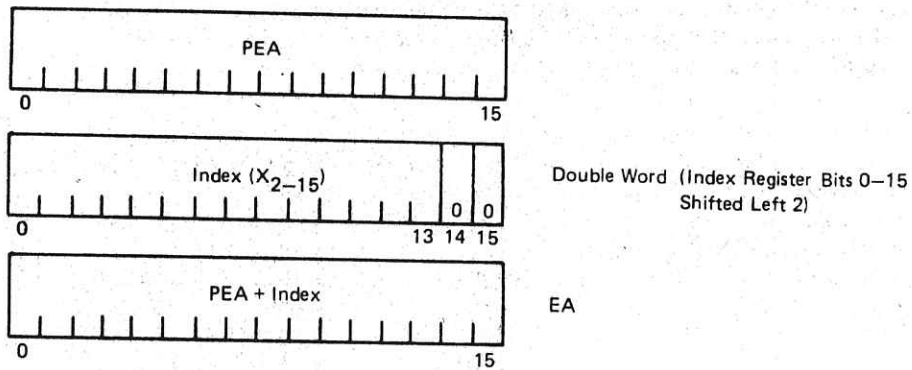
FF 46



a. Halfword Index Alignment



b. Fullword Index Alignment



c. Double Word Index Alignment

Figure 2-13. Automatic Index Alignment

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED IN ACCORDANCE WITH THE NATIONAL ARCHIVES ACT OF 1986
WITHOUT WRITTEN PERMISSION FROM THE NATIONAL ARCHIVES

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

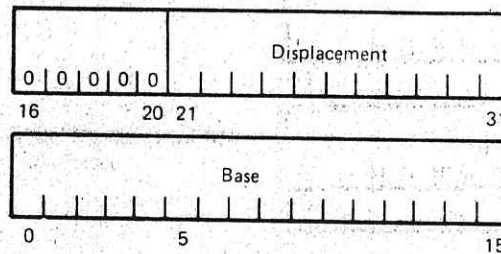


Figure 2-14. Displacement Alignment for Indexed Addressing

- 2) If B2 is not equal to 11, the 16-bit base, contained in the higher order half of the specified register, is added to the aligned displacement. This results in a preliminary effective address (PEA) whereby the $PEA = (B) + \text{Displacement}$.
If B2 is equal to 11, the aligned displacement is added to zero. This result is the preliminary effective address (PEA), whereby the $PEA = \text{Displacement}$.
- 3) If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a zero, then the 16-bit result of Step 2 is added to the contents of the updated instruction counter (IC) to form the 16-bit EA whereby $EA = \text{updated IC} + PEA^*$. (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section, with the exception that the Branch Sector Register (BSR) bits are used instead of the Data Sector Register (DSR) bits.)
- 4) If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the 16-bit result of Step 2 is subtracted from the contents of the updated IC to form the 16-bit EA whereby $EA = (\text{updated}) IC - PEA^*$. (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section with the exception that the Branch Sector Register (BSR) bits are used instead of the Data Sector Register (DSR) bits.)
- 5) If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a zero, then Indirect Addressing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used as the address of a main-storage halfword. This halfword is then fetched and expanded to 19-bits by using expanded addressing to form the EA. $EA = MS(PEA)$. Functional equivalency to preindexing capability can be obtained through modification of the base.

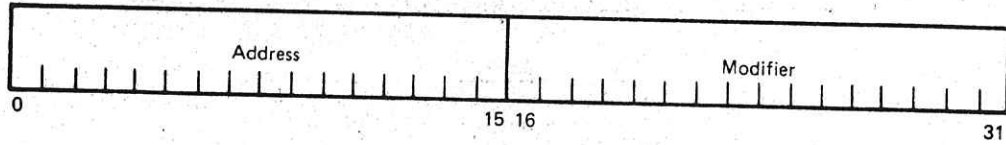
*Usage of B2 equal to 11 (no base) is encouraged in the relative addressing mode. Usage of B2 not equal to 11 may be changed in future computers.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, WITH OR WITHOUT PERMISSION.

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

- 6) If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a one, Indirect Addressing is performed as described in Step 5 with a full word main storage pointer. Then, storage modification is automatically performed. The indirect address is contained in a full word and must have an even address. A modifier is contained in bits 16 through 31. An address is contained in bits 0 through 15. The modifier is added to the address and the resulting modified address replaces bits 0 through 15 of the indirect address word. (See Figure 2-15.)

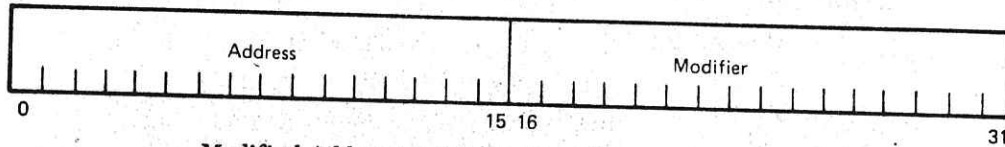


$$\text{Modified Address} = \text{MS (PEA)} \leftarrow \text{MS (PEA)} + \text{MS (PEA} + 1)$$

Figure 2-15. The Contents of Indirect Address Storage Modification Word

- 7) If the X field is not all zeros, IA (bit 19) is a zero and I (bit 20) is a zero, the most significant 16-bits of the general register specified by the X field are aligned, and then added to the 16-bit result of Step 2 (PEA) to form the 16-bit EA (see Figure 2-13). (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section.)
- 8) If the X field is not all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the most significant 16 bits of the general register specified by the X field are aligned, and then added to the 16-bit result of Step 2 (PEA) to form the 16-bit EA (see Figure 2-13). (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section.) (The modifier is added to the address and the resulting modified address replaces bits 0 through 15 of the index register after the EA is determined.)

Figure 2-16 illustrates the address and modifier format in the index register.



$$\text{Modified Address} = (X)_{0-15} \leftarrow (X)_{0-15} + (X)_{16-31}$$

Figure 2-16. The Contents of Index Register X

- 9) If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a zero, Indirect Addressing (IA) with post-indexing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used to fetch a main-storage halfword. The index contained in the general register specified by X is aligned and then added to the fetched halfword to form the 16-bit EA (see Figure 2-13). This EA is then expanded to a 19-bit EA by using expanded addressing. Functional equivalency to preindexing capability can be obtained through modification of the base.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

MS 87-08 Box 4046 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 TITLE 17 U.S. CODE
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION

10) If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a one, a direct addressing mode is defined using a 32-bit fullword indirect address pointer as follows:

a) First, the PEA from Step 2 must locate a fullword indirect address pointer, with the format as illustrated in Figure 2-17.

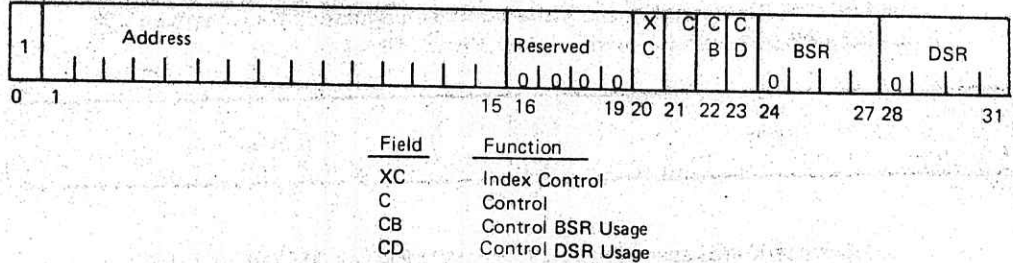


Figure 2-17. Fullword Indirect Address Pointer

b) If C (bit 21) equals 0, XC (bit 20) equals 1, and the instruction is not a branch type instruction, the 19-bit EA equals the 4-bit DSR with the 15-bit address field appended. When C (bit 21) equals 0, XC (bit 20) equals 0, and the instruction is not a branch type instruction, the 19-bit EA equals the 15-bit address field added to the index value in indexing register X with the result appended to the fullword indirect address pointer's DSR. The current PSW's DSR is not changed.

If C (bit 21) equals 0 and the instruction is a branch type instruction, the current PSW's BSR in conjunction with bits 0 through 15 of the fullword indirect address pointer will be used to form the BA. If XC = 0, post-indexing will occur. When C (bit 21) equals zero, CB and CD are reserved and should be set to zero.

c) If C (bit 21) equals 1 and the instruction is a branch type instruction and the branch is taken, the BSR and DSR fields selectively replace the corresponding fields in the current PSW, based on the CB and CD bit values as follows:

| CB | CD | Result |
|----|----|--|
| 0 | 0 | Use current PSW's BSR to form the BA. |
| 0 | 1 | Replace the current PSW's DSR with this DSR. Form the BA normally. |
| 1 | 0 | Replace the current PSW's BSR with this BSR before forming the BA. |
| 1 | 1 | First, replace the current PSW's DSR with this DSR. Then, replace the current PSW's BSR with this BSR before forming the BA. |

d) When C (bit 21) equals 1 and XC (bit 20) equals 1, postindexing is not performed. When C (bit 21) equals 1 and XC (bit 20) equals 0, the BA calculation includes a final addition of the index value in index registers X.

If C (bit 21) equals 1, XC equals 1, and the instruction is not a branch, the 19-bit EA equals the current PSW's DSR and the 15-bit field appended. If XC = 0, postindexing will occur.

MS 87-08 BOX 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

The results of indexed mode RS operations normally replace the first operand except for store operation where the first operand replaces the second operand. The second operand is unaltered for nonstore operations, and the first operand is unaltered for store operation.

EXPANDED ADDRESSING

The addressing philosophy accommodates $64K \times$ halfword addresses since a full 16-bit address is provided. Extending the addressing range beyond 64K halfword locations up to 512K halfword locations is provided by utilizing PSW bits.

Expanding to 19 bits is achieved by replacing the high-order bit of a 16-bit address with 4 bits, as shown in Figure 2-18. Data operand addresses are extended to 19 bits by specifying either a 4-bit Data Sector Register (DSR) or an implied DSR. When the high-order bit of a 16-bit data address is 1, a 4-bit DSR (PSW bits 28 through 31) is selected to replace the high-order bit. When the high-order bit of a 16-bit data address is a 0, an implied DSR containing 0000 is selected. Note that indirect addressing locates the indirect address pointer as if the pointer were a data operand. Branch addresses are extended to 19 bits in an equivalent manner. When the high-order bit of a 16-bit branch address is a 1, a 4-bit Branch Sector Register (BSR—PSW bits 24 through 27) is selected to replace the high-order bit. When the high-order bit is a 0, an implied BSR containing 0000 is selected. The high-order bit of both the BSR and DSR must be zero.

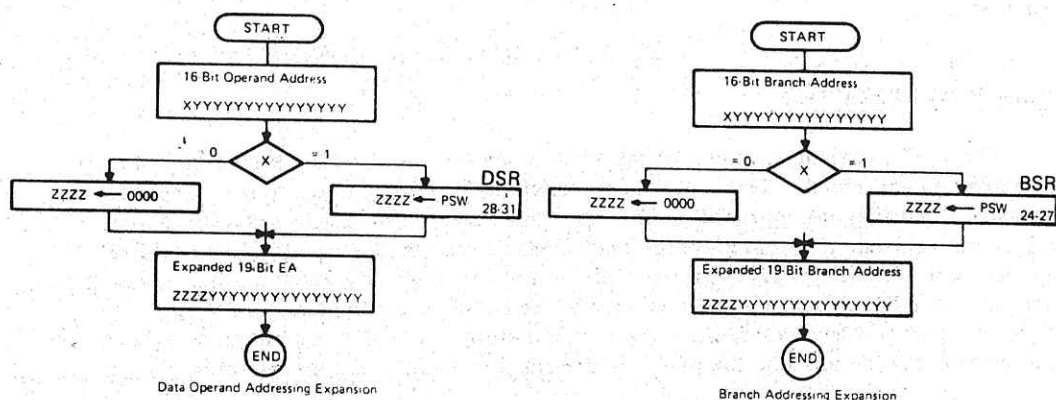
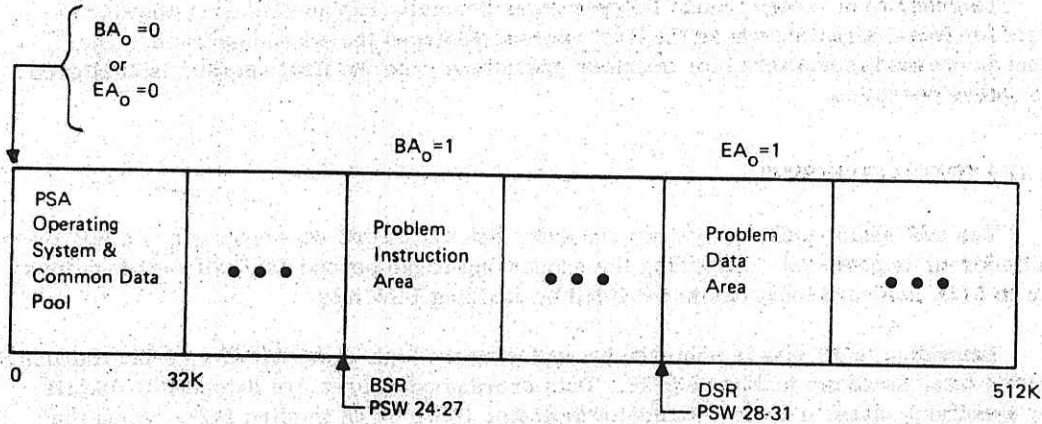


Figure 2-18. Expanded Addressing

*K = 1024

Pictorially, main storage can be visualized as follows:



This permits efficient communication from the problem program to the operating system, the preferred storage area, (PSA) or a common data area.

It should be cautioned that instruction address incrementing or address calculations used to form the EA are performed on the low 16 bits only and will not alter the BSR and DSR. This BSR or DSR may be altered only via a PSW swap, special instruction operations (SVC, LPS) or by use of the indirect address pointer described in this section.

PROGRAM EXECUTION

The CPU program consists of instruction and control words specifying the operations to be performed. This information resides in main storage and addressable registers and may be operated on as data. Instruction execution control is as defined under the section on Machine Status and General System Operation. Insert Storage Protect Bits, Load Program Status, Internal Control and Set System Mask instructions are privileged instructions and can only be executed in the Supervisor State. The Program Status Word determines the current state of the CPU and the Supervisor Call instruction can be used by the problem program to enter the Supervisor State.

STORAGE PROTECTION FEATURES

The storage protection feature prevents modification of specific main storage locations. Any location which could, for example, contain constant data or program instructions can be selectively protected from Store operations without restricting the use of other areas. Traps on store operations to specific data words can be inserted during program checkout. A privileged instruction, Insert Storage Protect Bits, is provided to set or reset the protection bit associated with each halfword of main storage. Attempting to store data in a protected location will result in a program interrupt unless it is previously masked by setting the machine check mask (PSW bit 45) to zero. In this case the store operation does not occur.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THE NATIONAL ARCHIVES WILL NOT REPRODUCE OR
REPRODUCE INFORMATION CONTAINED IN THIS DOCUMENT

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

INSTRUCTION MONITOR FEATURE

The storage protection bit described can also be used to flag an inadvertent attempt to execute, as instructions, data stored in unprotected areas. The feature will ensure that no program will continue to execute data as program instructions. An attempt to fetch an instruction word which is unprotected will result in an interrupt if PSW bit 34 is a one. The feature can be masked by a System Mask Bit (bit 34 of the PSW). During program checkout, this feature permits use of special software to aid debugging.

MACHINE STATUS

System status can be altered by the occurrence of interrupts, by the program, by manual intervention, and by external units such as another CPU. A doubleword register within the CPU contains a program status word (PSW) and is the focal point for CPU and system status conditions.

PROGRAM STATUS WORD

The program status word (PSW), contains the basic information required for proper program execution. The 64-bit PSW includes the next instruction address, the current condition code, the carry and overflow indicators, the system mask for interrupts, and other fields significant to CPU operations. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PSW is called the "current PSW." By storing the current PSW during an interruption, the status of the CPU can be preserved for subsequent use. By loading a new PSW or part of a PSW, the state of the CPU can be initialized or changed. Figure 2-19 shows the PSW format.

The overall status of the CPU is preserved in the current PSW and the contents of the general registers. The PSW is automatically retained upon taking an interrupt. It is the programmer's responsibility to preserve the contents of the general registers when necessary.

Certain other conditions that contribute to an overall system status situation are not automatically preserved when a CPU is interrupted. These conditions involve additional units and include the dynamic state of all other interrupts, the state of real time counters, and I/O system status.

Masking is accomplished by setting the appropriate PSW bit to zero.

PSW Fields

The PSW fields (Figure 2-19) are defined as follows:

- 1) Instruction Address — Bits 0 through 15 and 24 through 27 of the PSW contain the information to determine the address of the next instruction to be executed. The machine architecture makes provision to address 262,144 fullwords. However, the space shuttle hardware implementation allows for addressing a maximum of 131,072 fullwords.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

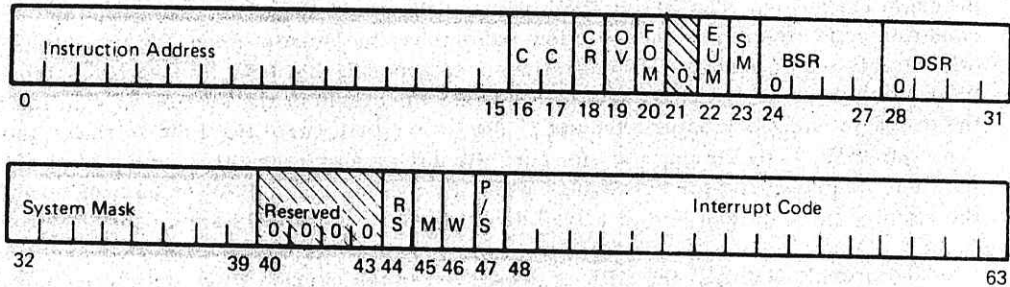
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
UNIVERSITY ARCHIVES
REPRODUCED IN ACCORDANCE WITH THE ARCHIVES
WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE REPRODUCED

MS 87-08 Box 104 FF 48

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION

2) CPU Status

| Bit | Use |
|--------|---|
| 16, 17 | Condition code for certain arithmetic, logical and I/O instructions |
| 18 | Carry status bit indicator |
| 19 | Overflow status bit indicator |
| 20 | Fixed-point Arithmetic Overflow Mask |
| 21 | Reserved |
| 22 | Floating-Point Exponent Underflow Mask |
| 23 | Significance Mask |



| | | | | |
|-------|--|-------|--|---------------|
| 0-15 | Next Instruction Address | 36 | External Interrupt 1 Mask | } System Mask |
| 16-17 | Condition Code | 37 | External Interrupt 2 Mask | |
| 18 | Carry Indicator | 38 | External Interrupt 3 Mask | |
| 19 | Overflow Indicator | 39 | External Interrupt 4 Mask | |
| 20 | Fixed-Point Arithmetic Overflow Mask | 40-43 | Reserved | |
| 21 | Reserved | 44 | Register Set (GR set 0 or 1) | |
| 22 | Floating Point Exponent Underflow Mask | 45 | Machine Check Mask | |
| 23 | Significance Mask | 46 | Wait State Bit (Wait/Process) | |
| 24-27 | Branch Sector Register | 47 | Problem/Supervisor State Control Bit | |
| 28-31 | Data Sector Register | 48-63 | Interrupt Code for Program Machine Check and Special External Interrupts | |
| 32 | Counter 1 Mask | | | |
| 33 | Counter 2 Mask | | | |
| 34 | Instruction Monitor Mask | | | } System Mask |
| 35 | External Interrupt 0 Mask | | | |

Figure 2-19. PSW Fields

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITER PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS ELECTRONIC OR MECHANICAL

- 3) Branch Sector Register — Bits 24 through 27 replace the high-order bit of a branch address when that bit is a 1. Otherwise, an implied sector register of 0000 replaces the high-order bit.
- 4) Data Sector Register — Bits 28 through 31 replace the high-order bit of a data address when that bit is a 1. Otherwise, an implied sector register of 0000 replaces the high-order bit.
- 5) System Mask — Bits 32 through 39 are mask bits. The first two bits of the System Mask are normally assigned to the two counters and the third to the instruction Monitor Feature. The remaining five masks include I/O end conditions, other application dependent items, such as, a manual interrupt key and timer overflow conditions. The instruction SET SYSTEM MASK is provided for modifying this field.
- 6) Reserved — bits 40 through 43 are reserved.
- 7) Register Select Field — The register select field, bit 44, controls which of two sets of general registers is in current use. When this bit is a zero, then register set 0 is used; when this bit is one, then register set 1 is used. The set of general registers in current use can be selected when a new PSW is loaded. This can result from the execution of the PSW load instruction or from an interrupt.
- 8) Machine Check Mask — Bit 45 is the mask bit which is used to inhibit machine check interrupts. (See Figure 2-20). When this bit is a one, then machine check interrupts, store protect interrupts, or external 1 interrupts detected by the CPU (see *note on Figure 2-20) are inhibited.
- 9) Wait State — Bit 46 determines the wait or processing (run) states. When this bit is a zero, the CPU is in the processing state. When this bit is a one, the CPU is in the Wait State.
- 10) Problem/Supervisor — Bit 47 determines the problem or supervisor states. When this bit is a zero, the CPU is in the supervisor state and privileged instructions can be executed. When this bit is a one, the CPU is in the problem state and attempts to execute privileged instructions are inhibited resulting in an interrupt.
- 11) Interrupt Code — Bits 48 through 63 are reserved for the interrupt code. Program and machine check interrupt conditions and associated interrupt codes are given in Figure 2-20.

INTERRUPTS

- 1) Power — This interrupt occurs when primary power is removed from the system for any reason. The current PSW, the general register set 1 & 2, the floating point registers, the counters 1 & 2 and the operational register are put away (stored) in main storage for future reference. Figure 2-21 shows the PSA assignments including putaway. When primary power is restored, operation is initiated with the "power on PSW" (if the power-up mode is defined as Run). This power-up condition is explained in General System Operation.

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY

THIS MATERIAL IS UNCLASSIFIED
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED
 XEROXED, REPRODUCED, OR TRANSMITTED IN ANY FORM OR BY ANY MEANS

| Interrupt Priority | Class | Old PSW | New PSW | Not Maskable | Mask Bit | Pending | Interrupt Code | Interrupt Accept Time | CPU/IOP/AGE Generated | Interrupt |
|--------------------|-------|---------|---------|--------------|----------|---------|----------------|--------------------------|-----------------------|---|
| 1A | PWR | - | 0004 | X* | - | - | - | Immediate | CPU | Power On |
| 1B | PWR | - | 0014 | X** | - | - | - | Immediate | CPU | System reset |
| 3 | MC | 0040*** | 0044 | - | 45 | - | 0003 | End of MS cycle | CPU | CPU (encountered) storage parity for main store (MS) access in the CPU unit |
| 4 | MC | 0040*** | 0044 | - | 45 | - | 0002 | End of I/O cycle | CPU | IOP (encountered) storage parity for MS access in either the IOP or CPU units |
| 5 | MC | 0040*** | 0044 | - | 45 | - | 0001 | End of MS cycle | CPU | CPU extended (IOP unit) memory address parity |
| 6 | MC | 0040*** | 0044 | - | 45 | - | 0004 | End of MS cycle | CPU | CPU extended (IOP unit) memory data parity |
| 9 | MC | 0040*** | 0044 | - | 45 | - | 0005 | Immediate | CPU | CPU ROS parity |
| 7 | PE | 0048*** | 004C | X | - | - | 0003 | Immediate | CPU | CPU address specification |
| 8 | PE | 0048*** | 004C | - | 45 | - | 0007 | End of MS cycle | CPU | CPU store protection violation |
| 11 | PE | 0048*** | 004C | X | - | - | 0000 | During instr fetch | CPU | Illegal operation |
| 11 | PE | 0048 | 004C | X**** | - | - | 0001 | During addr generation | CPU | Privileged instruction |
| 11 | PE | 0048 | 004C | - | 20 | - | 0004 | During instr execution | CPU | Fixed point overflow |
| 11 | PE | 0048 | 004C | - | 23 | - | 0005 | During instr execution | CPU | Significance |
| 11 | PE | 0048 | 004C | X | - | - | 0006 | During instr execution | CPU | Divide or convert inputs not normalized |
| 11 | PE | 0048 | 004C | - | 22 | - | 0008 | During instr execution | CPU | Exponent underflow (floating point or convert) |
| 11 | PE | 0048 | 004C | X | - | - | 000A | During instr execution | CPU | Exponent overflow (convert) |
| 11 | PE | 0048 | 004C | X | - | - | 000B | During instr execution | CPU | Exponent overflow (floating point) |
| 11 | PE | 0048 | 004C | X | - | - | 000C | During instr execution | CPU | Invalid divide zero divisor (floating point) |
| 12 | SC | 0058 | 005C | X | - | - | - | Address generation | CPU | Supervisor call |
| 14 | SYS | 0060 | 0064 | - | 32 | X | - | End of instr | CPU | Real-time CLK 1 |
| 15 | SYS | 0068 | 006C | - | 33 | X | - | End of instr | CPU | Real-time CLK 2 |
| 10 | PE | 0070*** | 0074 | - | 34 | - | - | Beginning of instr fetch | CPU | Instruction monitor (Masking can only be performed in supervisor state.) |
| 16A | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | Watchdog timer (IOP group 1 exception) |
| 16B | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | IOP voter (IOP group 1 exception) |
| 16C | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | C/M idle (IOP group 1 exception) |
| 16D | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | IOP ROS parity (IOP group 1 exception) |
| 16E | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | IOP fault (IOP group 1 exception) |
| 16F | SYS | 0078 | 007C | - | 35 | X | - | End of instr | IOP | Spare (IOP group 1 exception) |
| 17A | SYS | 0080 | 0084 | - | 36 | X | 0000 | End of instr | IOP | PCI/PCO Channel parity (IOP group 2 exception) |
| 17B | SYS | 0080 | 0084 | - | 36 | X | 0000 | End of instr | IOP | DMA instruction read parity (IOP group 2 exception) |
| 17C | SYS | 0080 | 0084 | - | 36 | X | 0000 | End of instr | IOP | DMA data read parity (IOP group 2 exception) |
| 17D | SYS | 0080 | 0084 | - | 36 | X | 0000 | End of instr | IOP | Burst DMA word count excess (IOP group 2 exception) |
| 17E | SYS | 0080 | 0084 | - | 36 | X | 0000 | End of instr | IOP | O overflow (IOP group 2 exception) |
| 17F | SYS | 0080 | 0084 | - | 36 | X | 0000 | During instr | IOP | DMA timeout (IOP group 2 exception) |
| 17G | SYS | 0080 | 0084 | - | 36 | X | 0003 | End of instr | CPU | DMA address specification |
| 17H | SYS | 0080 | 0084 | - | 36 | X | 0004 | End of instr | CPU | DMA store protect violation |
| 17I | SYS | 0080 | 0084 | - | 36&45 | X | 0002 | End of instr | CPU | DMA data write parity |
| 17J | SYS | 0080 | 0084 | - | 36&45 | X | 0001 | End of instr | CPU | PCI data parity |
| 17L | SYS | 0080 | 0084 | - | 36 | X | 0005 | End of instr | CPU | DMA address parity |
| 17K | SYS | 0080 | 0084 | - | 36 | X | 0006 | End of instr | AGE | AGE interrupt |
| 18A-18L | SYS | 0088 | 008C | - | 37 | X | - | End of instr | IOP | IOP programmed interrupts (1-12) |
| 19A-19D | SYS | 0090 | 0094 | - | 38 | X | - | End of instr | IOP | Spare (4) |
| 20A-20D | SYS | 0098 | 009C | - | 39 | X | - | End of instr | IOP | Spare (4) |
| 2A | PWR | - | - | X | - | - | - | End of instr | CPU | Power off interrupt |
| 2B | PWR | 0010 | - | X | - | - | - | End of instr + 100 μS | CPU | Power off interrupt delayed (POID) |
| 13 | PWR | 0010 | - | X | - | - | - | End of instr | CPU | Initiate putaway |

NOTE: *CPU must not be in halt mode. ***Contains address of next instruction or second half of existing full-word instruction.
 CPU must be in halt mode. **Only occurs when in problem state: PSW 47-1

Figure 2-20. Interrupt Codes

- FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
- TITLE 17 U.S. CODE
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS
- 2) Machine Check — When not masked, this interrupt class occurs following the detection of a malfunction. The current instruction is then terminated and the interrupt taken. A diagnostic procedure may then be initiated. When masked the interrupt does not remain pending.
 - 3) Program — This class of interrupt arises from improper specification or use of instructions or data. Bits 20, 22, and 23 in the PSW are provided to permit masking program interrupts due to arithmetic exceptions such as fixed point overflow. Bit 34 in the PSW is provided to permit masking the instruction monitor interrupt. Bit 45 of the PSW (Machine Check Mask) masks a store protection violation. When masked, program interrupts do not remain pending. When invalid instruction or address detection is provided, the resulting program interrupts cannot be masked.
 - 4) Supervisor Call (SVC) — This interrupt results from the execution of the SVC instruction. The 16-bit effective address is placed in the interruption code of the old PSW. This instruction can be used to switch from the problem to the supervisor state.
 - 5) System — This class of interrupt results from program counter time outs and conditions outside the CPU. Provision is made for 7 interrupt levels within this class, and each is provided with a unique set of PSWs and a mask bit. Two are program counters and 5 are external interrupts.

Any number of the 5 external interrupt conditions may be grouped into a single level by the external equipment. In the event of simultaneous external interrupt conditions, the lowest numbered (bit within the system mask field in the PSW) interrupt is taken first. These interrupts remain pending when masked except when the machine check mask bit is one.

The two program interval timers are each 32 bits wide and decrement. The lower 16 bits (least significant halfword) of each counter resides in 16-bit binary hardware counters that count down by one every microsecond. The high 16 bits (most significant halfword) of each counter resides in main store. The high halfword lies in main store location 00B0 for counter 1 and main store location 00B1 for counter 2. Every 65 ms when the low halfword (in the hardware counter) passes from 0000 (hex) to FFFF (hex) an interrupt occurs which can cause the high halfword in main store (via microcode) to be decremented by one. This interrupt is transparent to the programmer until the high halfword in main store equals 0000 (hex). When such an interrupt occurs, the high halfword is decremented to FFFF (hex) and a PSW swap occurs, telling the programmer that the counter has timed out. Note that if the interrupt is masked the high halfword will not be decremented by the microcode. The low halfword continues to count down. The interrupt although remains pending and if unmasked within 65 ms, the upper halfword will be decremented without a loss of a count.

The counters can be loaded and read by the Internal Control instruction, described in Section 10.

Interrupt Handling

The machine check, program, SVC, and each system interrupt have two related PSWs called "old" and "new" in unique main store locations. This zone of main store is referred to as a preferred storage area (PSA), which is illustrated in Figure 2-21.

In all cases and interruption involves merely storing the current PSW in its old position and making the PSW at the new position the current PSW. The old PSW holds all the necessary status information in the system existing at the time of interruption. If, at the conclusion of the interruption routine, there is an instruction to make the old PSW the current PSW, the system is restored to the state prior to the interruption, and the interrupted routine continues. This means the programmer must clear the fixed point overflow indicator before being reloaded. Note that it is possible to switch to the alternate set of general registers when the PSW swap takes place. This set of registers is defined by bit 44 in the new PSW.

Interruptions can only be taken when the CPU is interruptable for a given source. The system mask, machine check mask bit, floating-point exponent underflow mask, the significance mask, and the fixed-point overflow mask bits in the PSW define the interruptable state of the CPU with respect to those sources. When masked, system interrupts remain pending while machine check and program interrupts are ignored.

The power transient, certain program interrupts, and the SVC interrupt cannot be masked.

Interrupt Priority

Simultaneous interrupt requests are honored by the CPU. The smaller the hardware priority number the higher the priority. It should be noted that many of the interrupts listed in Figure 2-20 have the same priority number, this is because these interrupts are mutually exclusive and priority has no meaning.

When more than one unmasked interrupt requests service, the action consists of storing the old PSW and fetching the new PSW belonging to the interruption which is taken first. This new PSW subsequently is stored without any instruction execution and the next interruption PSW is fetched. This process continues until no more interruptions are to be serviced. When the last interruption request has been serviced, instruction execution is resumed using the PSW last fetched. The order of execution of the interruption subroutines is, therefore, the reverse of the order in which the PSWs are fetched. Machine check and power transient, when they occur, do not allow any other interrupt to be taken.

The above priority is used to resolve race conditions due to multiple interrupt conditions. Since separate mask bits and PSW pairs are provided for each external interrupt source, the priority in handling these interrupts is actually determined by the content of the new PSWs. When a PSW swap occurs, further action in regard to system (and machine check) interrupts is determined by the mask fields in the new PSW.

Interrupt Masking

Individual masking of several of the interrupt types is possible. When masked off, the interruption is either ignored or remains pending for later execution. The masking capability for each of the interrupt types is as follows:

- 1) Power Transient — Cannot be masked off.

FOR RESEARCH USE ONLY
THIS MATERIAL IS UNCLASSIFIED

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08

Box 4042

FF 46

- FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
- 1) COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
- WITHOUT WRITEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.
- 2) Machine Check — Can be masked off by setting the machine check mask bit 45 in the PSW equal to zero. When masked off normal instruction sequencing occurs, and the interrupts do not remain pending.
 - 3) Program — Three of the 11 program interrupts are capable of being masked off; fixed-point arithmetic overflow, exponent underflow, and significance, by setting the appropriate mask bits in the PSW equal to zero. When masked off these interrupts do not remain pending. Also, the storage protect interrupt can be masked via the machine check mask (PSW bit 45). Note that if a PSW with both Fixed Point Overflow Indicator and mask (bits 19 and 20) set is used, the interrupt will occur.
 - 4) Supervisor Call — Cannot be masked off.
 - 5) System — Each level of external interrupts can individually be masked off by setting the corresponding system mask bit in the PSW equal to zero. Interrupts that are masked remain pending.

Preferred Storage Area (PSA) Assignments

The contents of the PSA are shown in Figure 2-21 with the main store address expressed in hexadecimal notation. The following PSA locations must not be store protected:

- 1) Power off interrupt PSW
- 2) All old PSW locations
- 3) Main store location 0087 (used by microprogram for I/O operations)
- 4) Counter 1 & 2, high halfword locations 00B0 & 00B1
- 5) Putaway locations (00C0 through 0103).

In addition, MS location 0087 must be set initially to zero for use by self test

GENERAL SYSTEM OPERATION

The various states entered by the computer and their relationship to the basic operator controls are shown in Figure 2-22. The basic controls provided for the operator are power-on, initial program load (IPL) and the system reset key. Among the many controls available, these functions have special significance because of their relationship to an unconditional system reset sequence. These functions each produce a system reset sequence which applies to the computer, I/O channels, and peripherals. Further operation within the system differs, however, as explained in the following sections.

Power-On

One of two modes of operation must be specified for the system at power-on. The first results in a system reset followed by the computer entering the stop state. In this state, instructions are not processed, interrupts are not accepted, and system timers are not updated. This state is termed "manual" because further operation must be determined by the operator.

The second mode at power-on enters the run state after the system reset is complete. The instruction stream is initiated and interrupts are processed. The computer can be removed from the run state by certain instructions, interruptions, and by manual intervention.

System Reset

The system reset function rests the computer system to a known state such that processing can be initiated without the presence of machine checks, except for those caused by subsequent machine malfunctions. The system reset function causes the following:

- CPU pending interrupts are reset
- Internal timers are reset to all ones (1's)
- Status registers are reset

IPL

The use of the IPL function is independent of the current state of the system. IPL first causes a system reset function.

Operating State

The run state and wait state shown in Figure 2-22 are collectively termed the operating state for the system. When the computer is in the run state, instructions are executed in the normal manner. An instruction may be encountered or an interrupt processed that forces the computer into the wait state. The computer does not execute instructions in the wait state, but it is interruptable when not masked. System timers are updated and input/output operations continue in the wait state.

The wait state may also be entered after completing IPL or by special operating intervention via the stop state. (Dotted lines on Figure 2-22). This action is the result of the wait bit being set in the controlling PSW.

Program State Alternatives

Certain other states exist within the CPU that contribute to its overall status. These states are directly related to program operation and are

- 1) Masked or Interruptable State — The computer may be masked for certain interrupt conditions at any given time. These conditions generally remain pending within the system until the masked condition is changed by the program. Certain error conditions cannot be masked off, while other error conditions such as program checks are ignored when specifically masked.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPIES PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION

- 2) Supervisor or Problem State — In the supervisor state all instructions are valid. In the problem state, I/O and certain other instructions are invalid, and their use produces an error interrupt. This state is controlled by a bit in the PSW. The SVC instruction is provided to switch from problem to supervisor state. The LOAD PSW instruction is used to switch from supervisor to problem state.
- 3) General Register Selection — Bit 44 in the current PSW selects the set of general registers in current use.

ARCHITECTURAL GROWTH

Throughout this Principles of Operation manual, architecture conventions are defined or facilities are marked "reserved" to retain flexibility for future implementations and extensions. The computer operates in conformance to this manual when architecture definitions are followed consistently. Hardware operation when these rules are violated are not defined and are properly outside the scope of this manual to retain flexibility of implementation. "Programmer discovered" operations that violate or go beyond the definitions described herein but produce "useful" functions should not be used and should be considered "reserved" because the results obtained may vary from computer to computer or even release levels for one computer depending upon options selected or the design release level to which the hardware is manufactured.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW
(TITLE 17, U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

PHOTOCOPIED BY PERMISSION OF THE AUTHOR. THIS MATERIAL MAY NOT BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08

Box 42

FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

The fields of the CW are defined as follows:

- ID:** For an input operation, bit 0 must be coded as 0. For an output operation, this bit must be coded as 1.
- Command (M):** Bits 1-31 specify the particular operation to be performed, and can be used to expand the basic input and output operations. For example, they can be coded to specify sense and control operations. Additionally, DMA I/O operations can be initialized by a Direct I/O. In executing an input operation, the channel (1) transmits the 32-bit CW to the external device; and (2) subsequently loads 32 bits of information, transmitted from the addressed device, into general register R1. In executing an output operation, the channel (1) transmits the CW to the external device, and (2) subsequently transmits bits 0-31 of general register R1 to the addressed device. The specific definition of the command bits is described in the Principles of Operation for PCI/PCO, MSC, & BCE. The only restriction placed on the system design is the definition of bit 0.

Each control unit connected to the channel is required to accept the CW, decode the control unit and device address, and perform the input or output operation defined by the command field. The device address field identifies, for example, the flight control subsystem, the radar altimeter, the navigation sensors, the displays, or the mass storage unit. The number and the types of devices connected to the channel and their address assignments depend on the system configuration.

If the IO handshaking operation does not complete within 9 microseconds for CW & DATA OUT transfers or 6 microseconds for data in transfers, the Program Controlled instruction will terminate and the condition code will be set to reflect the time-out.

RESULTING CONDITION CODE

- 00 Operation successful
01 Interface time-out error: operation not successful

INDICATORS

The overflow and carry indicators are not changed by this instruction.
Program Interrupt — Privileged instruction

PROGRAMMING NOTE

This is a privileged instruction and can only be executed when the CPU is in the supervisor state.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL,
INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

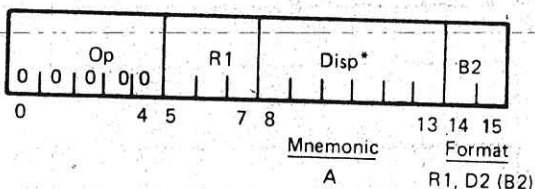
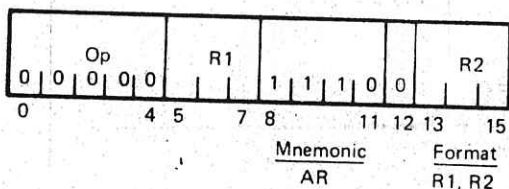
Section 4
FIXED-POINT ARITHMETIC

For all of the following sections, [@] [#] indicates that the use of indirect addressing and/or autoindexing is optional. For example, M specifies direct addressing without autoindexing, while M# specifies direct addressing with autoindexing.

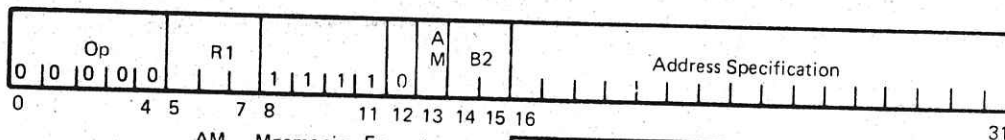
The arithmetic instruction set performs binary arithmetic on fixed-point, fractional operands. Fullword operands are signed and 32 bits long. Negative quantities are represented in twos complement form.

Halfword operands are 16 bits long. Within the CPU, a halfword operand from storage is developed into a fullword operand prior to instruction execution. This is done by using the contents of the halfword second operand location as the most significant 16 operand bits and generating 16 low-order zeros. This result is the second operand.

ADD

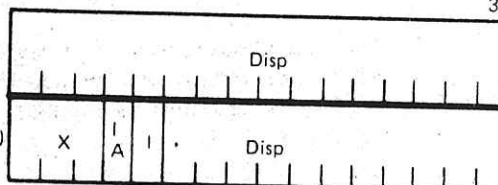


*Displacements of the form 111XXX are not valid.



Extended: 0 A R1, D2 (B2)

Indexed: 1 A [@] [#] R1, D2 (X2, B2)



FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
UNWRITTEN PERMISSION. THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE ARCHIVES DEPARTMENT.

MS 87-08 BOX 42 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The fullword second operand is added to the contents of general register R1. The result replaces the contents of general register R1. The second operand is not changed.

RESULTING CONDITION CODE

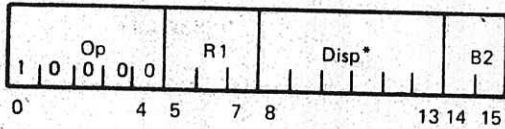
- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (>0).

INDICATORS

The overflow indicator is set to one if the magnitude of the sum is too large to be represented in the general register; that is, greater than $1-2^{-31}$ or less than -1. If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of the general register.

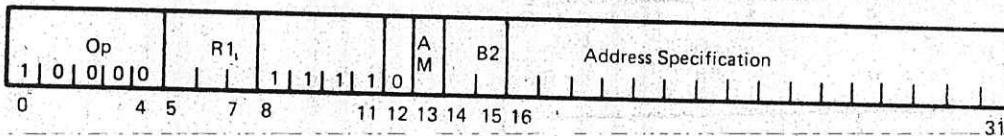
Program Interrupt - Fixed point overflow

ADD HALFWORD

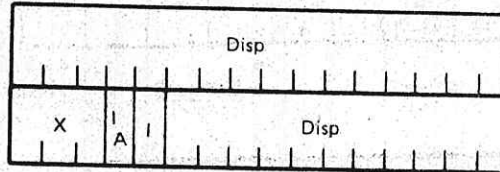


*Displacements of the form 111XXX are not valid.

Mnemonic: AH
Format: R1, D2 (B2)



| | AM | Mnemonic | Format |
|-----------|----|------------|-----------------|
| Extended: | 0 | AH | R1, D2 (B2) |
| Indexed: | 1 | AH (@) [#] | R1, D2 (X2, B2) |



DESCRIPTION

The halfword second operand is first developed into a fullword operand by appending 16 low-order zeroes. This fullword operand is then added to the contents of general register R1. The result replaces the contents of general register R1. The second operand is not changed.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

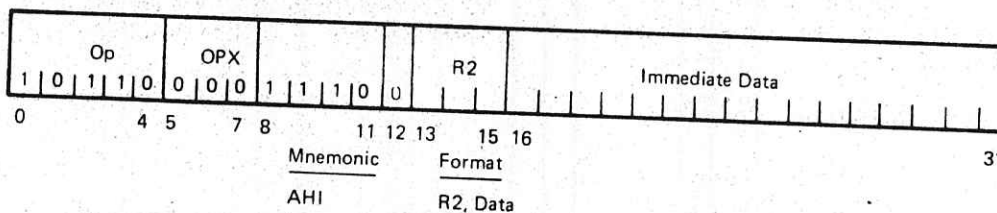
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (>0).

INDICATORS

The overflow indicator is set to one, if the magnitude of the sum is too large to be represented in the general register; that is, greater than $1-2^{-31}$ or less than -1 . If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of the general register.

Program Interrupt — Fixed point overflow

ADD HALFWORD IMMEDIATE

DESCRIPTION

Instruction bits 16 through 31 are treated as immediate data. The halfword immediate data is first developed into a fullword operand by appending 16 low-order zeroes. The resulting fullword operand is then added to the contents of general register R2. The result replaces the contents of general register R2. The immediate operand is not changed.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (>0).

INDICATORS

The overflow indicator is set to one if the magnitude of the sum is too large to be represented in the general register; that is, greater than $1-2^{-31}$ or less than -1 . If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of the general register.

Program Interrupt — Fixed point overflow

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)

COPY PROVIDED BY

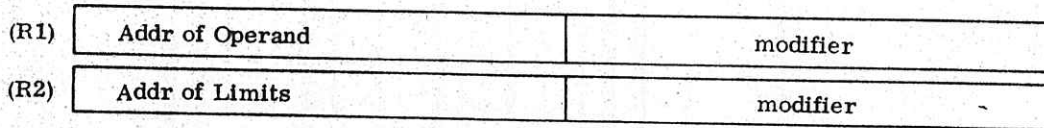
WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

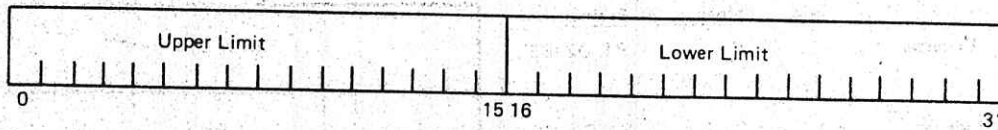
MS 87-08

Box 4042

FF 46



The address of a 16-bit two's complement integer operand is contained in bits 0 through 15 of general register R1. The address of a fullword with the following format containing the upper and lower limits is contained in bits 0 through 15 of the general register R2:



These limits are 16-bit two's complement integers.

In bits 16 through 31 of general registers R1 and R2 are 16-bit two's complement integer modifiers. After the addresses in bits 0 through 15 have been used to locate the operands, each modifier is added to the most significant 16 bits of the registers. The result replaces the most-significant 16 bits. The modifier is not changed, overflows and carry out of the most-significant address bit are ignored.

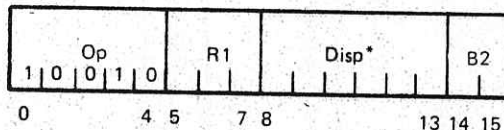
RESULTING CONDITION CODE

- 00 Within Limits: Lower Limit ≤ Operand ≤ Upper Limit
- 01 Above Upper Limit: Operand > Upper Limit
- 11 Below Lower Limit: Operand < Lower Limit

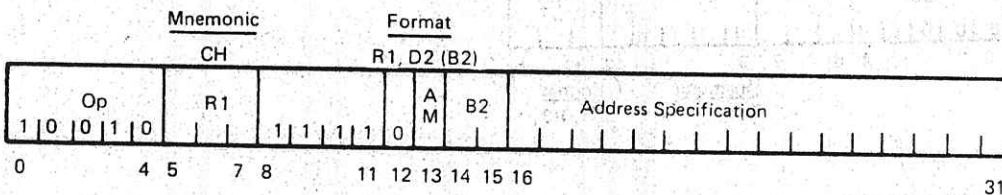
INDICATORS

The overflow and carry indicators are not changed by this instruction.

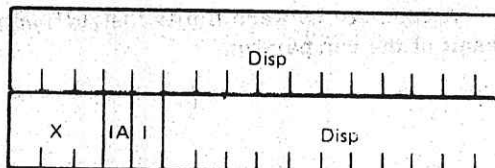
COMPARE HALFWORD



*Displacements of the form 111XXX are not valid.



| | AM | Mnemonic | Format |
|-----------|----|-----------|----------------|
| Extended: | 0 | CH | R1, D2 (B2) |
| Indexed: | 1 | CH(@) [#] | R1, D2 (X2,B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED
 REPRODUCED IN ANY MANNER WITHOUT PERMISSION OF THE REPRODUCING AGENCY

MS 87-08 BOX 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The halfword second operand is first developed into a fullword operand by appending 16 low-order zeros. This fullword operand is then algebraically compared with the contents of general register R1. The contents of the general register and main storage are not changed at the end of instruction execution.

RESULTING CONDITION CODE

- 00 The contents of general register R1 equals the developed fullword operand
- 11 The contents of general register R1 are less than the developed fullword operand
- 01 The contents of general register R1 are greater than the developed fullword operand.

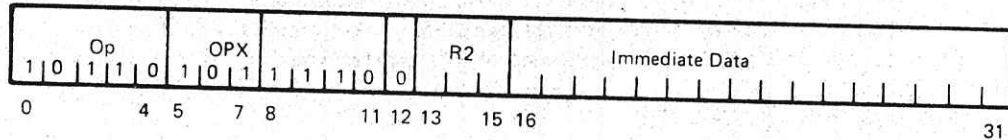
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

After development, all 32 bits of the fullword operand participate in the comparison.

COMPARE HALFWORD IMMEDIATE



| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| CHI | R2, Data |

DESCRIPTION

Instruction bits 16 through 31 are treated as immediate data. This halfword of immediate data is first developed into a fullword operand by appending 16 low-order zeros. This fullword operand is then algebraically compared with the contents of general register R2. The contents of the general register and main storage are not changed at the end of instruction execution.

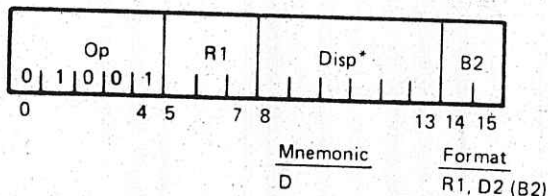
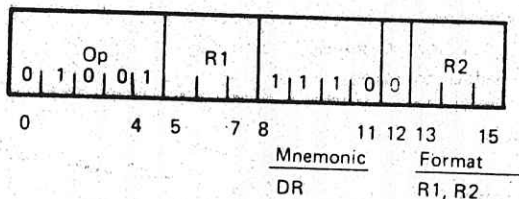
RESULTING CONDITION CODE

- 00 The contents of general register R2 equals the developed fullword operand
- 11 The contents of general register R2 are less than the developed fullword operand
- 01 The contents of general register R2 are greater than the developed fullword operand.

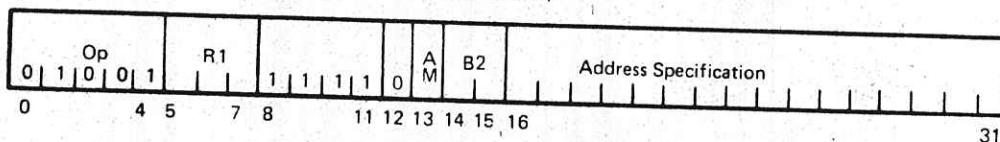
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17, U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FASHION, NOR FACED, IN ANY MANNER.

MS 87-08 Box 40
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

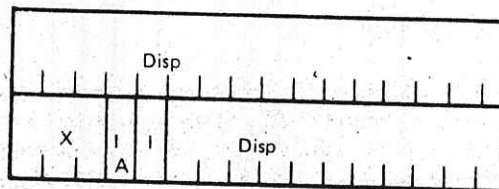
DIVIDE



*Displacements of the form 111XXX are not valid.



| | AM | Mnemonic | Format |
|-----------|----|---------------|-----------------|
| Extended: | 0 | D | R1, D2 (B2) |
| Indexed: | 1 | D [@] [#] | R1, D2 (X2, B2) |



DESCRIPTION

The first operand, a 64-bit, signed 2's complement dividend, is contained in the even/odd general register pair R1 and R1 ⊕ 1. The most-significant portion is in R1. When R1 indicates an odd general register, the first operand is developed by appending 32 low-order zeros to the contents of R1. The second operand is the divisor.

The first operand is divided by the second operand. The unrounded quotient replaces the contents of general register R1. The remainder is not developed. When R1 is even, specifying an even/odd general register pair, the contents of R1 ⊕ 1 are indeterminate at the end of instruction execution. When R1 is odd, R1 ⊕ 1 is never changed. The second operand is not changed.

When the relative magnitude of dividend and divisor is such that the quotient cannot be expressed as a 32-bit signed fraction, an overflow is generated. In this event, the contents of both R1 (and R1 ⊕ 1 when R1 is even) are indeterminate upon instruction termination.

RESULTING CONDITION CODE

The code is not changed.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

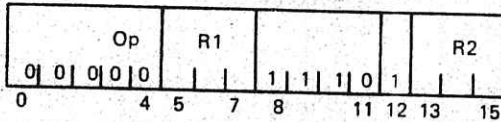
MS 87108 - Box 4045
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives
 FF 46

INDICATORS

The overflow indicator is set to one when the quotient cannot be represented, or when division by zero is attempted. The dividend is destroyed in these cases. If the overflow indicator already contains a one, it is not changed. The carry indication has no significance following execution and is indeterminate.

Program Interrupt — Fixed point overflow

EXCHANGE UPPER AND LOWER HALFWORDS



Mnemonic
XUL

Format
R1, R2

DESCRIPTION

The upper halfword of general register R1 is exchanged with the lower halfword of general register R2. Bits 0 through 15 of general register R1 replace bits 16 through 31 of general register R2 while simultaneously bits 16 through 31 of general register R2 replace bits 0 through 15 of general register R1.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

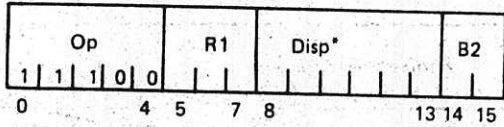
The overflow and carry indicators are not changed.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

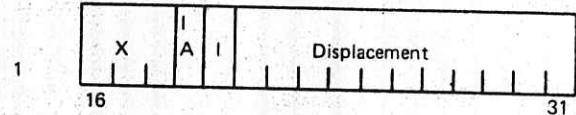
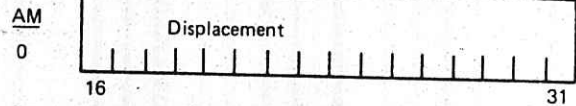
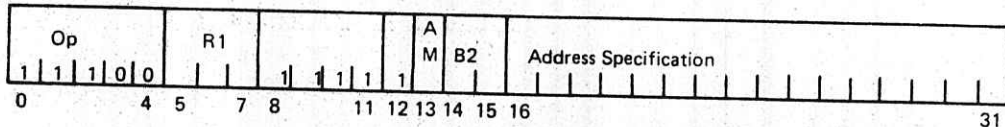
MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

INSERT ADDRESS LOW



* Displacements of the form 111XXX are not valid.

| Mnemonic | Format |
|----------|-------------|
| IAL | R1, D2 (B2) |



| | AM | Mnemonic | Format |
|-----------|----|-----------------|-----------------|
| Extended: | 0 | IAL | R1, D2 (B2) |
| Indexed: | 1 | IAL [@] [#] | R1, D2 (X2, B2) |

DESCRIPTION

A 16-bit effective address is developed in the normal manner without expanding to 19-bits. This address itself replaces the 16 low-order bits of general register R1. The 16 high-order bits of general register R1 are not changed.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

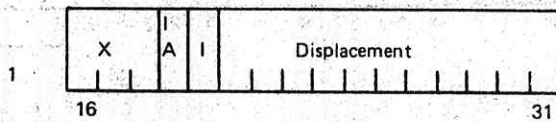
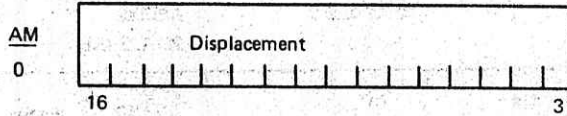
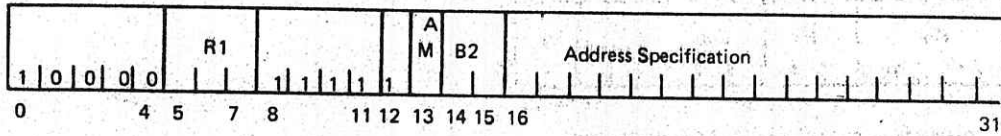
The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

1 COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

MS 87-08 BOX 1042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

INSERT HALFWORD LOW



| | AM | Mnemonic | Format |
|-----------|----|---------------------------|-------------|
| Extended: | 0 | IHL | R1, D2 (B2) |
| Indexed: | 1 | IHL (@)[#]R1, D2 (X2, B2) | |

DESCRIPTION

The halfword second operand replaces the contents of bits 16-31 of general register R1. Bits 0-15 of general register R1 are not changed. The second operand is not changed.

RESULTING CONDITION CODE

The code is not changed.

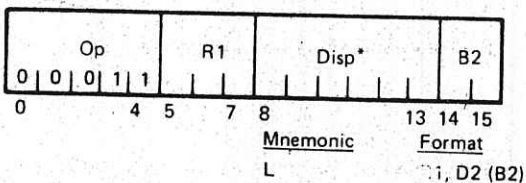
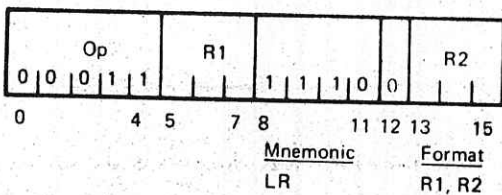
INDICATORS

The overflow and carry indicators are not changed by this instruction.

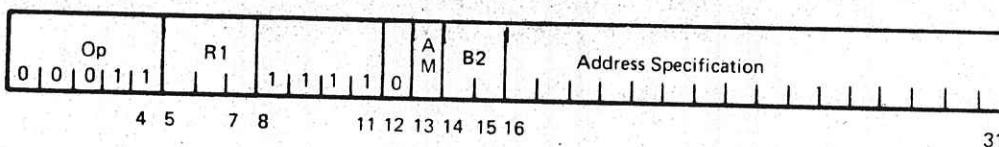
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

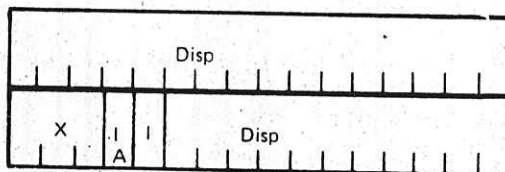
LOAD



*Displacements of the form 111XXX are not valid.



| | <u>AM</u> | <u>Mnemonic</u> | <u>Format</u> |
|-----------|-----------|-----------------|-----------------|
| Extended: | 0 | L | R1, D2 (B2) |
| Indexed: | 1 | L [@] [#] | R1, D2 (X2, B2) |



The fullword second operand is placed in general register R1. The second operand is not changed.

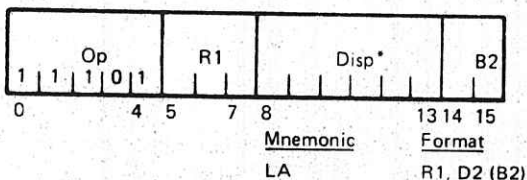
RESULTING CONDITION CODE

- 00 The second operand is zero
- 11 The second operand is negative
- 01 The second operand is positive (> 0).

INDICATORS

The overflow and carry indicators are not changed by this instruction.

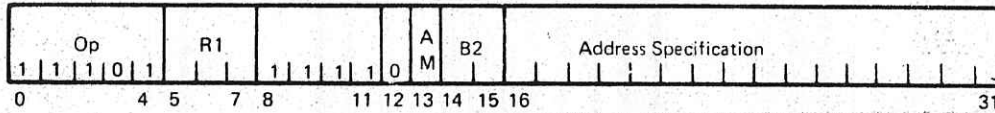
LOAD ADDRESS



*Displacements of the form 111XXX are not valid.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 FILE 17 US CODE
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY INFORMATION SYSTEM

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives



| | AM | Mnemonic | Format | Disp | |
|-----------|----|----------------|-----------------|------|------|
| Extended: | 0 | LA | R1, D2 (B2) | | |
| Indexed: | 1 | LA [@] [#] | R1, D2 (X2, B2) | X | Disp |

DESCRIPTION

A 16-bit effective halfword address is developed in the normal manner without expanding to 19-bits. This address itself replaces the 16 high-order bits of general register R1. The 16 low-order bits of general register R1 are zeroed.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

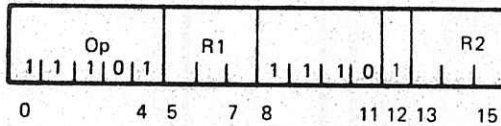
The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

When R1 = B2, it is possible to increment R1 by the displacement field.

In the RS format when B2 = 11 and AM = 0, this is functionally equivalent to a LOAD HALFWORD IMMEDIATE instruction. In this case, bits 16 through 31 are treated as immediate data. The immediate data is expanded to 32 bits by appending 16 low-order zeros. This resulting fullword operand replaces the contents of general register R1.

LOAD ARITHMETIC COMPLEMENT



| Mnemonic | Format |
|----------|--------|
| LCR | R1, R2 |

DESCRIPTION

The two's-complement of the fullword second operand replace the contents of general register R1. Complementation is accomplished by adding the one's complement of the fullword second operand and a low-order one.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY MEDIUM

MS 87-08 Box 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

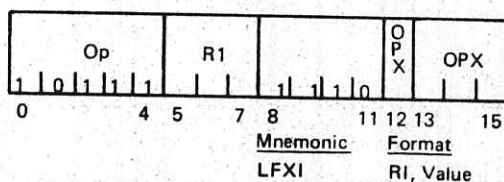
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (> 0).

INDICATORS

The overflow indicator is set to one when the maximum negative number is complemented. If the overflow indicator already contains a one, it is not altered by this instruction. The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of general register. The carry indicator will only be set when the operand is zero.

Program Interrupt — Fixed point overflow

LOAD FIXED IMMEDIATE

DESCRIPTION

A fixed-point literal value is loaded into the general register specified by R1.

The immediate values are -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 or 13. The immediate is loaded into bits 0 through 15 of general register R1. Bits 16 through 31 of general register R1 are set to zero.

| <u>OPX (Bits 12, 13, 14 & 15)</u> | <u>Immediate Value → R1</u> |
|---------------------------------------|-----------------------------|
| (hex) | (hex) |
| 0 | FFFE0000 |
| 1 | FFFF0000 |
| 2 | 00000000 |
| 3 | 00010000 |
| 4 | 00020000 |
| 5 | 00030000 |
| 6 | 00040000 |
| 7 | 00050000 |
| 8 | 00060000 |
| 9 | 00070000 |
| A | 00080000 |
| B | 00090000 |
| C | 000A0000 |
| D | 000B0000 |
| E | 000C0000 |
| F | 000D0000 |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REMOVED FROM ARCHIVE
REMOVED FROM ARCHIVE

DESCRIPTION

The halfword second operand is developed into a fullword operand by appending 16 low-order zeros. The resulting fullword operand replaces the contents of general register R1. The second operand is not changed.

RESULTING CONDITION CODE

- 00 The fullword operand is zero
- 11 The fullword operand is negative
- 01 The fullword operand is positive (>0).

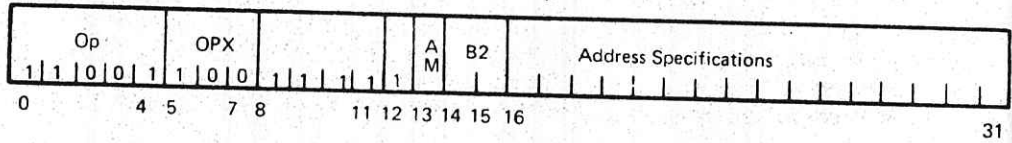
INDICATORS

The overflow and carry indicators are not changed by this instruction.

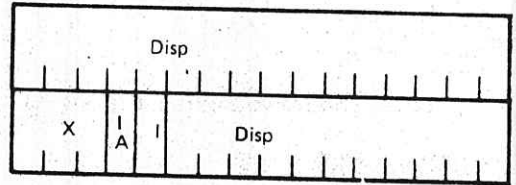
PROGRAMMING NOTE

This instruction clears the low-order half of general register R1.

LOAD MULTIPLE



| | AM | Mnemonic | Format |
|-----------|----|----------------|-------------|
| Extended: | 0 | LM | D2 (B2) |
| Indexed: | 1 | LM [@] [#] | D2 (X2, B2) |



DESCRIPTION

All eight general registers are loaded from the eight fullword locations starting at the fullword, second operand address. The general registers are loaded in ascending order.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

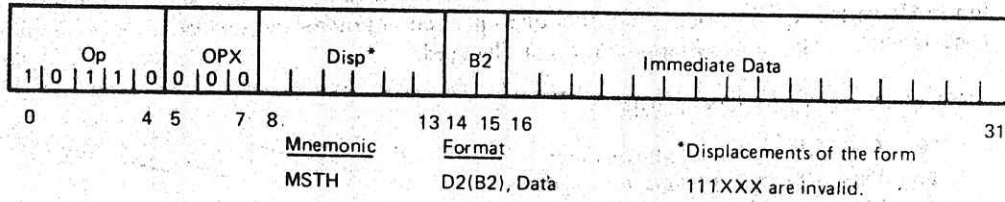
This instruction will always have halfword index alignment and will be excluded from automatic index alignment.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITH OUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY INFORMATION SYSTEM

MS 87-08 BOX 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

MODIFY STORAGE HALFWORD



DESCRIPTION.

Instruction bits 16 through 31 are treated as immediate data representing a 2's complement integer. This immediate data is added to the halfword main storage operand. The result replaces the halfword main storage operand. The contents of the general registers are not changed. Only the contents of the halfword main storage operand location is altered.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (>0).

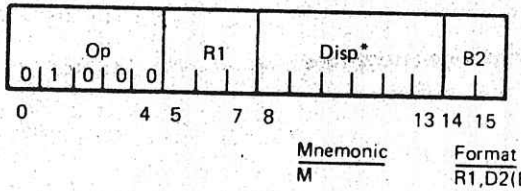
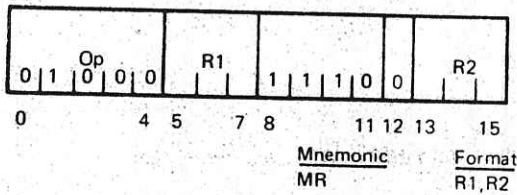
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

The MSTH immediate data (mask) is algebraically added to the halfword operand in main storage. Tally up and tally down is thus possible.

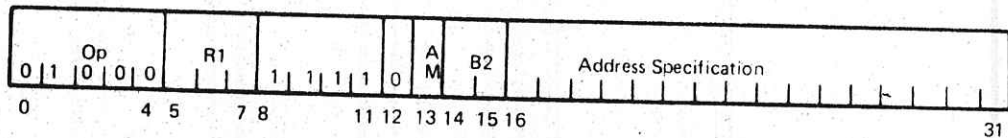
MULTIPLY



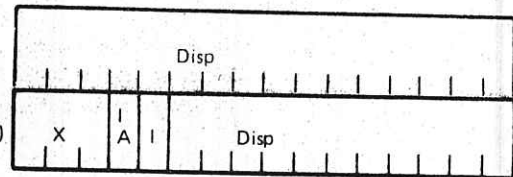
*Displacements of the form 111XXX are not valid.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 TITLE 17, U.S. CODE
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WICHITA, KANSAS 67260-0044
 NOT TO BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives



| | AM | Mnemonic | Format |
|-----------|----|-----------|--------------|
| Extended: | 0 | M | R1,D2(B2) |
| Indexed: | 1 | M (@) [#] | R1,D2(X2,B2) |



DESCRIPTION

The product of the multiplier (the second operand) and the multiplicand (the first operand) replaces the multiplicand. Both multiplier and multiplicand are 32-bit signed 2's complement fractions. The product is a 64-bit, signed 2's complement fraction and occupies an even/odd register pair when the R1 field references an even-numbered general register. When R1 is odd, only the most significant 32 bits of the product is saved in general register R1.

RESULTING CONDITION CODE

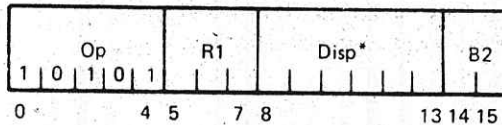
The code is not changed.

INDICATORS

The overflow indicator is set to one when -1 is multiplied by -1. If the overflow indicator already contains a one, it is not altered by this instruction.

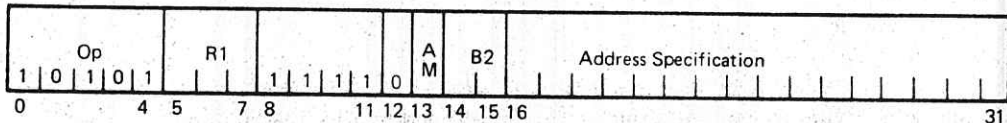
Program Interrupt - Fixed point overflow

MULTIPLY HALFWORD

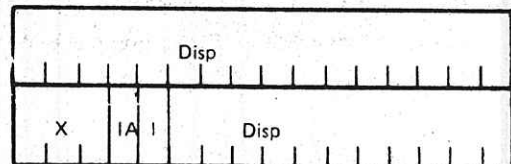


*Displacements of the form 111XXX are not valid.

| Mnemonic | Format |
|----------|-----------|
| MH | R1,D2(B2) |



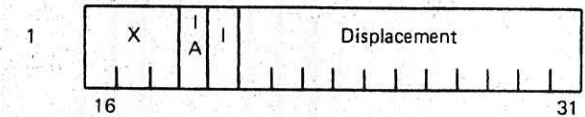
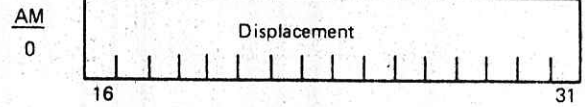
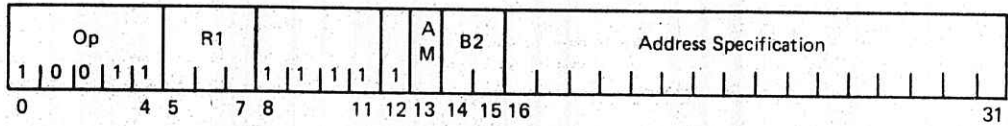
| | AM | Mnemonic | Format |
|-----------|----|------------|--------------|
| Extended: | 0 | MH | R1,D2(B2) |
| Indexed: | 1 | MH (@) [#] | R1,D2(X2,B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR PLACED IN ANY MEDIUM

MS 87-08 BOX 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

MULTIPLY INTEGER HALFWORD



| | AM | Mnemonic | Format |
|-----------|----|-------------|-----------------|
| Extended: | 0 | MIH | R1, D2 (B2) |
| Indexed: | 1 | MIH [@] [#] | R1, D2 (X2, B2) |

DESCRIPTION

The product of the multiplier (the two's complement signed integer halfword second operand) and the two's complement signed integer halfword multiplicand (the contents of bits 0 through 15 of general register R1) replaces the multiplicand. An intermediate product is formed as a 31-bit signed integer. This product is algebraically shifted left 15 places, to form a two's complement signed halfword integer product. This halfword product replaces bits 0 through 15 of general register R1. Bits 16 through 31 of general register R1 are zeroed.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

Program Interrupt – Fixed point overflow

The overflow indicator is set when the upper 16 bits of the intermediate product does not equal all ones or all zeroes. If the overflow indicator already contains a one, it is not altered by this instruction.

PROGRAMMING NOTE

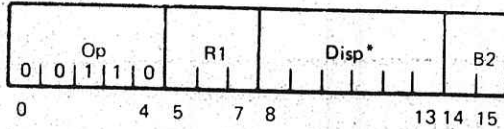
If I, J, and K are halfword operands, the equation $I*J+K$ may be solved with the following code:

```
LH   R1,I
MIH  R1,J
AH   R1,K
```

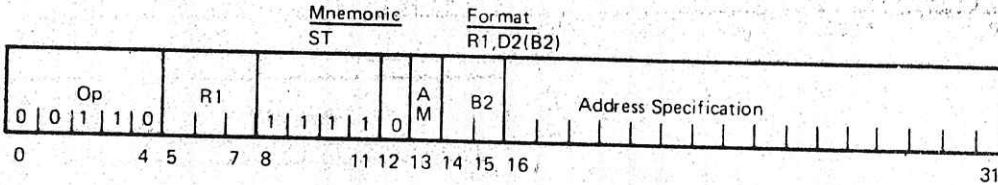
WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY.
 FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY

MS 87-08
 Box 1042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

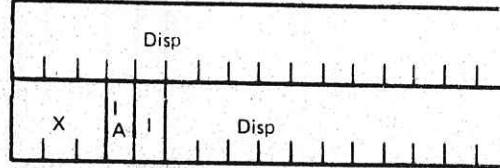
STORE



*Displacements of the form 111XXX are not valid.



| | AM | Mnemonic | Format |
|-----------|----|----------|---------------|
| Extended: | 0 | ST | R1,D2,(B2) |
| Indexed: | 1 | ST @ # | R1,D2 (X2,B2) |



DESCRIPTION

The contents of general register R1 are stored at the fullword second operand location. The contents of general register R1 are not changed.

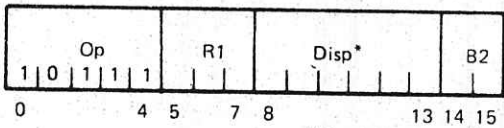
RESULTING CONDITION CODE

The code is not changed.

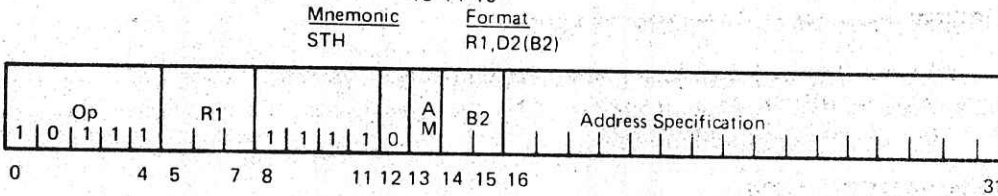
INDICATORS

The overflow and carry indicators are not changed by this instruction.

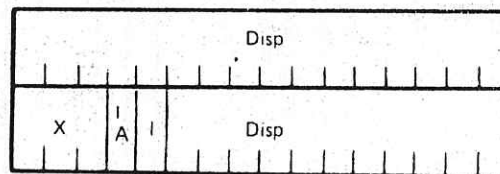
STORE HALFWORD



*Displacements of the form 111XXX are not valid.



| | AM | Mnemonic | Format |
|-----------|----|-----------|---------------|
| Extended: | 0 | STH | R1,D2 (B2) |
| Indexed: | 1 | STH @ # | R1,D2 (X2,B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 TITLE 17, U.S. CODE
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 ALL RIGHTS RESERVED. THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE
 WICHITA STATE UNIVERSITY LIBRARIES.

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The most significant 16 bits (bits 0 through 15) of general register R1 are stored at the halfword second operand location. No other storage location is altered. The contents of general register R1 are not changed.

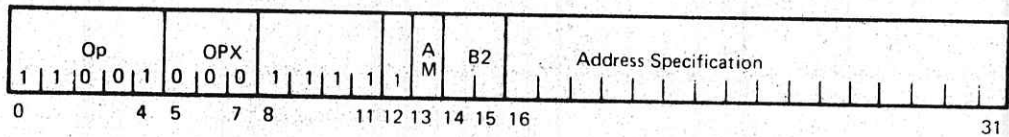
RESULTING CONDITION CODE

The code is not changed by this instruction.

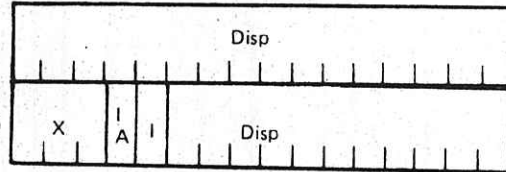
INDICATORS

The overflow and carry indicators are not changed by this instruction.

STORE MULTIPLE



| | AM | Mnemonic | Format |
|-----------|----|-------------|-----------|
| Extended: | 0 | STM | D2(B2) |
| Indexed: | 1 | STM (@) [#] | D2(X2,B2) |



DESCRIPTION

All eight general registers are stored at the eight fullword locations starting at the fullword second operand address. The general registers are stored in ascending order.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

This instruction is excluded from automatic index alignment. Indexes will always specify the halfword.

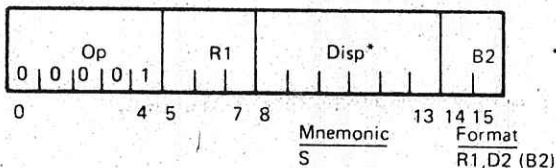
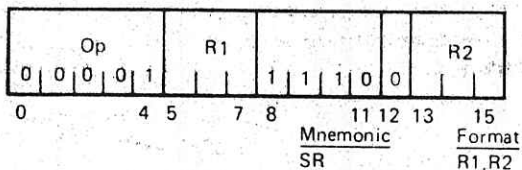
FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE PROTECTED BY COPYRIGHT LAW (TITLE 17 U.S. CODE)

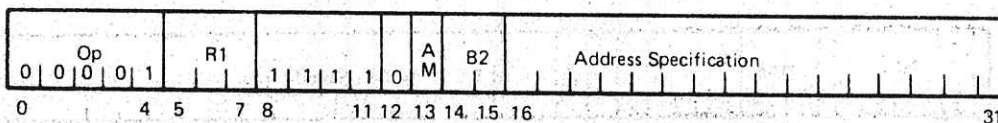
COPY PROVIDED BY WICHITA STATE UNIVERSITY LIBRARIES SPECIAL COLLECTIONS

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

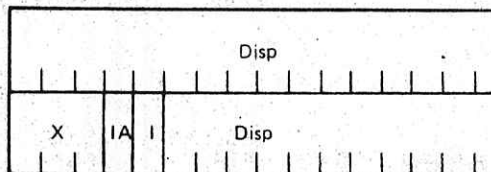
SUBTRACT



*Displacements of the form 111XXX are not valid.



| | | | |
|-----------|-----------|-----------------|---------------|
| | <u>AM</u> | <u>Mnemonic</u> | <u>Format</u> |
| Extended: | 0 | S | R1,D2 (B2) |
| Indexed: | 1 | S@[I] [#] | R1,D2 (X2,B2) |



DESCRIPTION

The fullword second operand is subtracted from the contents of general register R1. The result replaces the contents of general register R1. The second operand is not changed.

Subtraction is performed by adding the one's-complement of the second operand and a low-order one to form the two's complement for the fullword. This fullword is added to the first operand. All 32 bits of both operands participate as in ADD. The overflow, carry, and condition code indicators reflect the result of this addition.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (> 0).

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY INFORMATION SYSTEM

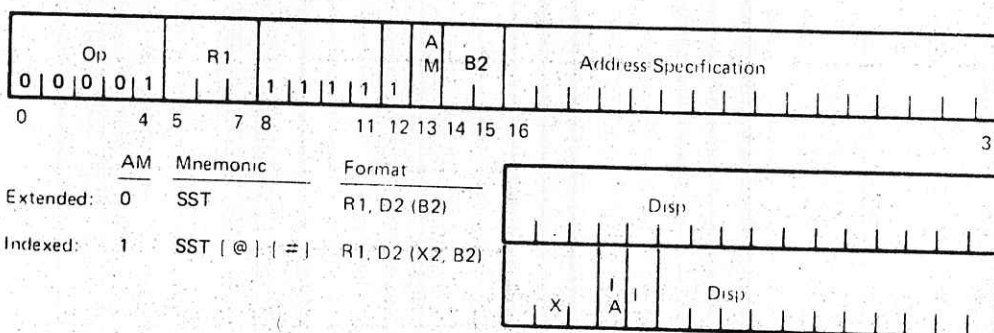
MS 87-08 Box 42 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

INDICATORS

The overflow indicator is set to one if the magnitude of the difference is too large to be represented in R1; that is, greater than $1-2^{-31}$ or less than -1 . If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of R1.

Program Interrupt — Fixed point overflow

SUBTRACT FROM STORAGE



DESCRIPTION

The contents of general register R1 is subtracted from the fullword second operand. The result replaces the contents of the second operand location. The first operand is not changed.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (>0).

INDICATORS

The overflow indicator is set to one if the magnitude of the sum is too large to be represented in the second operand location. That is, greater than $1-2^{-31}$ or less than -1 . If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of the result.

Program Interrupt — Fixed point overflow

FOR RESEARCH USE ONLY

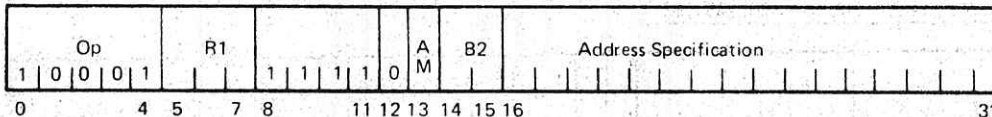
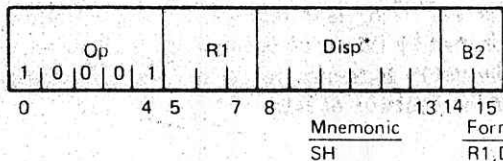
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

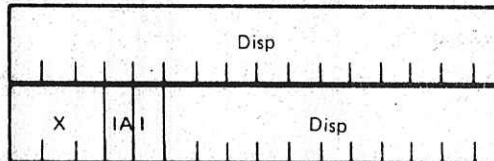
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

SUBTRACT HALFWORD



| | AM | Mnemonic | Format |
|-----------|----|------------|--------------|
| Extended: | 0 | SH | R1,D2,(B2) |
| Indexed: | 1 | SH (@) ≠ | R1,D2(X2,B2) |



DESCRIPTION

The halfword second operand is first developed into a fullword operand by appending 16 low-order zeroes. This second operand is then subtracted from the contents of general register R1. The result replaces the contents of general register R1. The second halfword operand is not changed.

Subtraction is performed by adding the ones complement of the developed fullword operand and a low-order one to form the fullword twos complement. This fullword is added to the first operand.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is negative
- 01 The result is positive (> 0).

INDICATORS

The overflow indicator is set to one if the magnitude of the sum is too large to be represented in R1; that is, greater than $1-2^{-31}$ or less than -1. If the overflow indicator already contains a one, it is not altered by this instruction. (Overflow can be reset by testing or by loading the PSW.) The carry indicator is set to indicate whether or not there is a carry out of the high-order bit position of R1.

Program Interrupt — Fixed point overflow

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE SWATHRAL WAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

MS 87-08 Box 4046 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

For the RR format, the branch address is contained in bits 0 through 15 of general register R2, if R2 ≠ 0. This 16-bit branch address is expanded to a 19-bit branch address. (See Expanded Addressing.)

RESULTING CONDITION CODE

The code is not changed.

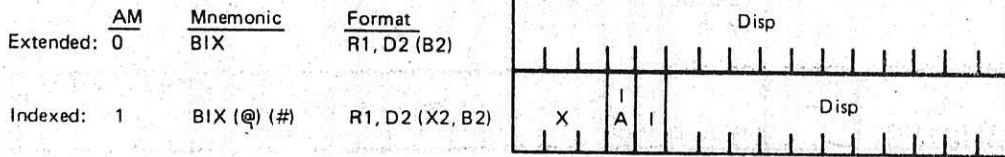
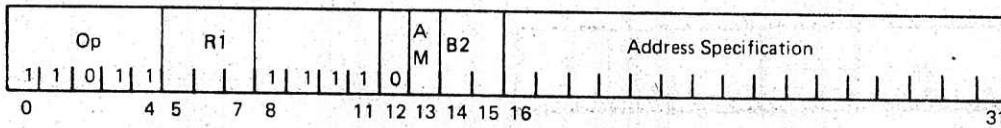
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

The assembly instruction BALR R1, 0 causes the address (instruction counter and BSR) of the next sequential instruction to be stored in bits 0 through 15, and 24 through 27 of general register R1. In this particular case, no branch is taken.

BRANCH AND INDEX



DESCRIPTION

"Bits 0 through 15 of the general register specified by R1 contain an index. Bits 16 through 31 of general register R1 contain a count. An effective address is computed in the normal manner for the extended class. (For the indexed addressing mode, the fullword indirect address pointer must contain zero's in bit locations 22 and 23.) Next, the index is incremented by one. Then the count is decremented by one. If the count prior to update is greater than zero, a branch to the effective address is taken. If the count prior to update is less than or equal to zero, no branch occurs."

RESULTING CONDITION CODE

The code is not changed.

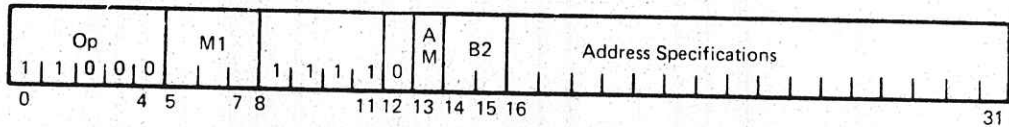
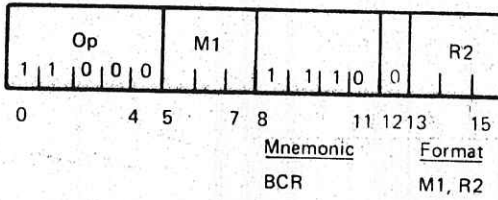
INDICATORS

The carry and overflow indicators are not changed by this instruction.

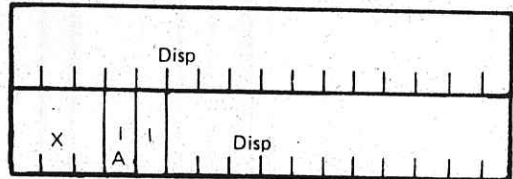
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE REPRODUCED
 REPRODUCED IN ANY MANNER WITHOUT PERMISSION

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

BRANCH ON CONDITION



| AM | Mnemonic | Format |
|-------------|----------------|-----------------|
| Extended: 0 | BC | M1, D2 (B2) |
| Indexed: 1 | BC [@] [=] | M1, D2 (X2, B2) |



DESCRIPTION

This instruction tests the PSW condition code status bits. Instruction bits 5 through 7 (the M1 field) specify which condition code (bits 16 and 17 of the PSW) is to be tested. Instruction bit 5 tests for a code equal 00, instruction bit 6 tests for a code equal 1i, and instruction bit-7 tests for a code equal 01. Whenever the condition code test is **successful, the branch is taken. Thus, when more than one bit of the M1 field is a one, the branch is taken for any successful test. (e.g., M1 = 111 always branches, M1 = 000 never branches.)**

The branch address is contained in bits 0 through 15 of general register R2 for the RR format. This 16-bit branch address is expanded to a 19-bit branch address. (See Expanded Addressing.)

RESULTING CONDITION CODE

The condition code was set following all arithmetic, logical, test, and compare instructions, and otherwise remains unchanged unless the program status word is altered. The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WISCONSIN STATE UNIVERSITY LIBRARY
 SPECIAL COLLECTIONS
 PERMISSION TO REPRODUCE THIS MATERIAL MAY NOT BE OBTAINED FROM
 THE NATIONAL ARCHIVES WITHOUT PERMISSION FROM THE NATIONAL ARCHIVES

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

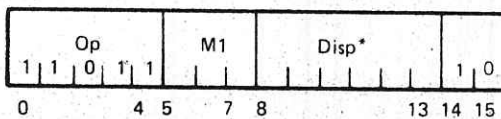
PROGRAMMING NOTE

The result and test conditions are shown as follows:

| | M1 Field (Test) | | |
|-------------------------------|-----------------|-----|-----|
| | (5) | (6) | (7) |
| <u>Arithmetic & Tally</u> | | | |
| Zero | 1 | 0 | 0 |
| Negative | 0 | 1 | 0 |
| Positive (>0) | 0 | 0 | 1 |
| <u>Logical</u> | | | |
| Zero | 1 | 0 | 0 |
| Not Zero | 0 | 1 | 0 |
| <u>Test</u> | | | |
| Zero | 1 | 0 | 0 |
| Mixed | 0 | 1 | 0 |
| All ones | 0 | 0 | 1 |
| <u>Compare</u> | | | |
| Equal | 1 | 0 | 0 |
| $0_1 < 0_2$ | 0 | 1 | 0 |
| $0_1 > 0_2$ | 0 | 0 | 1 |

It is possible to combine tests. For example, following the MSTH instruction, an M1 field of 1 0 1 specifies branch on non-negative (zero or positive).

BRANCH ON CONDITION BACKWARD



*Displacements of the form 111XXX are not valid.

| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| BCB | M1, D2 |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR BY ANY MEANS.

MS 87-08 BOX 42 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

This instruction tests the PSW condition code status bits. Instruction bits 5 through 7 (the M1 field) specify which condition code (bits 16 and 17 of the PSW) is to be tested. Instruction bit 5 tests for a code equal 00, instruction bit 6 tests for a code equal 11, and instruction bit 7 tests for a code equal 01. Whenever the condition code test is successful, the branch is taken by subtracting the Disp from the updated IC. Thus, when more than one bit of the M1 field is a one, the branch is taken for any successful test (e.g., M1=111 always branches).

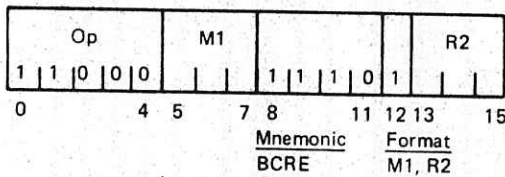
RESULTING CONDITION CODE

The condition code was set following all arithmetic, logical, test, and compare instructions, and otherwise remains unchanged unless the program status word is altered. The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

BRANCH ON CONDITION (EXTENDED)



DESCRIPTION

This instruction tests the PSW condition code status bits. Instruction bits 5 through 7 (the M1 field) specify which condition code (bits 16 and 17 of the PSW) is to be tested. Instruction bit 5 tests for a code equal 00, instruction bit 6 tests for a code equal 11, and instruction bit-7 tests for a code equal 01. Whenever the condition code test is successful, the branch is taken. Thus, when more than one bit of the M1 field is a one, the branch is taken for any successful test. (e.g., M1 = 111 always branches.)

When the branch is taken, PSW bits 0 through 15 and bits 24 through 31 are replaced by corresponding bits in general register R2.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE #7 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 THE MATERIAL MAY NOT BE REPRODUCED
 WITHOUT PERMISSION

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

RESULTING CONDITION CODE

The condition code was set following all arithmetic, logical, test, and compare instructions, and otherwise remains unchanged unless the program status word is altered. The code is not changed by this instruction.

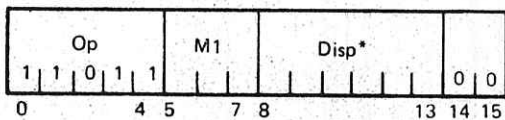
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

This instruction is similar to the RR version of the BRANCH ON CONDITION instruction. It is provided to facilitate subroutine returns across sector boundaries after general register R2 had been initialized by the use of the BRANCH AND LINK instruction.

BRANCH ON CONDITION FORWARD



*Displacements of the form 111XXX are not valid.

| | |
|----------|--------|
| Mnemonic | Format |
| BCF | M1, D2 |

DESCRIPTION

This instruction tests the PSW condition code status bits. Instruction bits 5 through 7 (the M1 field) specify which condition code (bits 16 and 17 of the PSW) is to be tested. Instruction bit 5 tests for a code equal 00, instruction bit 6 tests for a code equal 11, and instruction bit 7 tests for a code equal 01. Whenever the condition code test is successful, the branch is taken by adding the Disp to the updated IC. Thus, when more than one bit of the M1 field is a one, the branch is taken for any successful test (e.g., M1=111 always branches).

RESULTING CONDITION CODE

The condition code was set following all arithmetic, logical, test, and compare instructions, and otherwise remains unchanged unless the program status word is altered. The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 1700 WEST WASHINGTON, NORTH AND NORTHWEST CORNERS
 WICHITA, KANSAS 67260-0001

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

This instruction tests the PSW overflow and carry indicator status bits. The M1 field, instruction bits 5 through 7 specifies the test. Instruction bit 6 is tested against PSW bit 18 (carry), and instruction bit 7 is tested against PSW bit 19 (overflow). Whenever a specified bit of the PSW is a one, the test is successful and the branch is taken. Thus, when both indicators are tested by M1 = 011, the branch is taken if either indicator contains a one. A one in instruction bit 5 inverts the logic, causing bits 6 and 7 to test the PSW bits for zero.

For the RR format, the branch address is contained in bits 0 through 15 of general register R2. This 16-bit branch address is expanded to a 19-bit branch address. (See Expanded Addressing.)

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

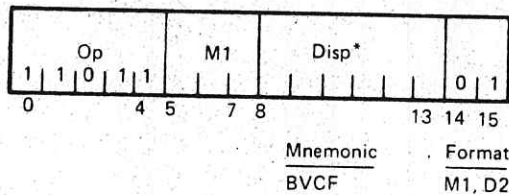
The overflow indicator is set 0 by this instruction. The carry indicator is not changed by this instruction.

PROGRAMMING NOTE

The possible combinations of test conditions are shown as follows:

| <u>M1 Field</u> | <u>Test Conditions</u> |
|-----------------|---------------------------------------|
| 5 6 7 0 0 0 | Branch never taken (no operation) |
| 0 0 1 | Branch on Overflow |
| 0 1 0 | Branch on Carry |
| 0 1 1 | Branch either on Overflow or on Carry |
| 1 0 0 | Branch |
| 1 0 1 | Branch On No Overflow |
| 1 1 0 | Branch On No Carry |
| 1 1 1 | Branch On No Overflow and No Carry |

BRANCH ON OVERFLOW AND CARRY FORWARD



*Displacements of the form 111XXX are not valid.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17, U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE ORIGINAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 1045 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

This instruction tests the PSW overflow and carry indicator status bits. Instruction bits 5 through 7 specify the test. Instruction bit 6 is tested against PSW bit 18, and instruction bit 7 is tested against PSW bit 19. Whenever a specified bit of the PSW is a one, the test is successful and the branch is taken by adding the Disp to the updated IC. Thus, when both indicators are tested by M1 = 011, the branch is taken if either indicator contains a one. A one in instruction bit 5 inverts the logic, causing bits 6 and 7 to test the PSW bits for zero.

The branch address is formed by adding the displacement to the updated instruction counter.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

The overflow indicator is set 0 by this instruction. The carry indicator is not changed by this instruction.

PROGRAMMING NOTE

The possible combinations of test conditions are shown as follows:

| <u>M1 Field</u> | <u>Test Conditions</u> |
|-----------------|---------------------------------------|
| 5 6 7 | |
| 0 0 0 | Branch never taken (no operation) |
| 0 0 1 | Branch on Overflow |
| 0 1 0 | Branch on Carry |
| 0 1 1 | Branch either on Overflow or on Carry |
| 1 0 0 | Branch |
| 1 0 1 | Branch On No Overflow |
| 1 1 0 | Branch On No Carry |
| 1 1 1 | Branch On No Overflow and No Carry |

Section 6

SHIFT OPERATIONS

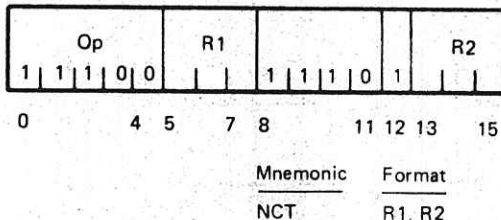
Shift instructions use the halfword format. The shift count is defined by the count field, as shown in Figure 6-1.

| Instruction Bits 8 through 13 | Shift Count Determined By |
|-------------------------------|--|
| 000000 (Zero) | No Operation |
| 000001-110111 (1 through 55) | Instruction bits 8 through 13 |
| 111000 (56) | Bits 10 through 15 of general register 0 |
| 111001 (57) | Bits 10 through 15 of general register 1 |
| 111010 (58) | Bits 10 through 15 of general register 2 |
| 111011 (59) | Bits 10 through 15 of general register 3 |
| 111100 (60) | Bits 10 through 15 of general register 4 |
| 111101 (61) | Bits 10 through 15 of general register 5 |
| 111110 (62) | Bits 10 through 15 of general register 6 |
| 111111 (63) | Bits 10 through 15 of general register 7 |

Figure 6-1. Shift Count

If the shift count is 56 through 63, bits 10 through 15 of the corresponding general register (0 through 7) designate the shift count. When specified using the count field, the maximum shift count allowed for shift operations is 55. Shifts of up to 63 positions are allowed, when general register 0 through 7 is used to specify a computed shift.

NORMALIZE AND COUNT



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17, U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

First, all bits (0 through 31) of general register R1 are set to zero. For each position that the contents of general register R2 are shifted, to the left, the high-order half of general register R1 bits (0 through 15) is incremented by 1. The shift terminates when bit position 0 \neq bit position 1 of general register R2. If the contents of general register R2 are initially zero, a count of zero is entered in general register R1. Zeros are entered into the vacated low-order bits of general register R2. Upon completion of this instruction, the count is contained in bits 0 through 15 of general register R1.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The carry indicator will be zero at the end of the operation, if general register R2 contains zero. The carry indicator will be one at the end of the operation, if the shift is terminated by the detection of bit position one not equal to bit position 0 of the general register R2. The overflow indicator is not changed by this instruction.

PROGRAMMING NOTE

If the initial condition of general register R2 was such that bit position 0 is not equal to bit position 1, the count in the high-order bit of general register R1 is zero, the carry indicator is one, and there is no shift. If the initial condition of R2 was all ones, the count is 31, the carry is one and R2 contains 80000000.

This instruction is executed as shown below in Figure 6-2.

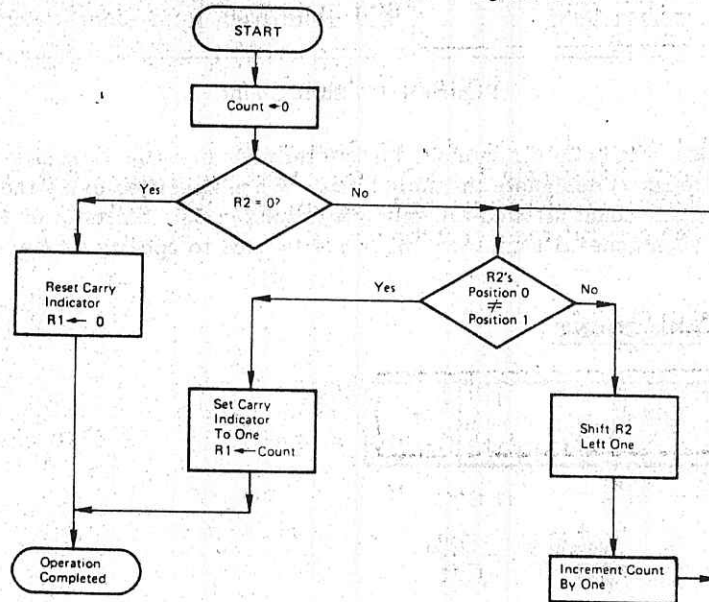
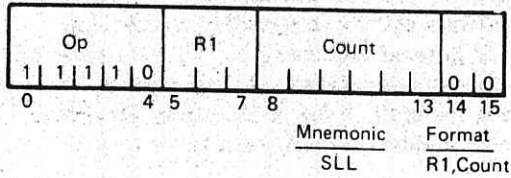


Figure 6-2. Normalize and Count Execution

SHIFT LEFT LOGICAL



DESCRIPTION

The contents of general register R1 are shifted left, as specified by the shift count Figure 6-1. Zeros are entered into the vacated low-order bits of general register R1. Bits leaving the high-order bit (bit 0 of general register R1) position are entered in the carry indicator. (See indicators below.) Bits shifted out of the carry indicator are lost. Only the contents of general register R1 are changed.

RESULTING CONDITION CODE

The code is not changed by this instruction.

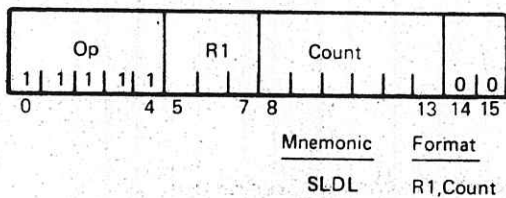
INDICATORS

The carry indicator is set to one for each one, and to zero for each zero, shifted left from the high-order position of general register R1. The overflow indicator is not changed by this instruction.

PROGRAMMING NOTE

When the shift count n is greater than 31, then the result of the shift of general register R1 is zero.

SHIFT LEFT DOUBLE LOGICAL



RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
TITLE 17 U.S. CODE

COPY PROVIDED BY
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
SPECIAL COLLECTIONS
THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED WITHOUT PERMISSION

MS 87-08 Box 1042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The contents of the even/odd pair of general registers (R1 and R1 ⊕ 1) are shifted left as a 64-bit register. The number of positions shifted is specified by the shift count. Bits shifted out of bit position zero, of general register R1 ⊕ 1, are entered into bit position 31 of general register R1. Zeros are entered into the vacated low-order bits of general register R1 ⊕ 1. Bits leaving the high-order bit position (bit position 0 of general register R1) are shifted into the carry indicator. Bits shifted out of the carry indicator are lost.

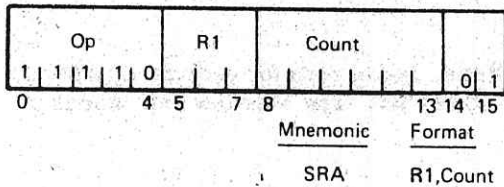
RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The carry indicator is set to one for each one, and to zero for each zero, shifted left from the high-order bit position of general register R1. The overflow indicator is not changed by this instruction.

SHIFT RIGHT ARITHMETIC



DESCRIPTION

The contents of general register R1 are shifted right the number of places indicated by the shift count. Bits equal to the sign are entered into vacated high-order bit positions. Bits shifted out of bit position 31 of general register R1 are lost.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

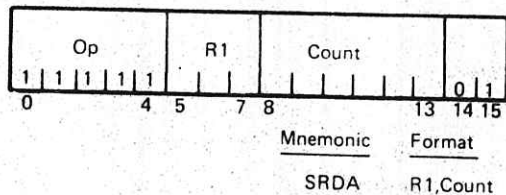
The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS ELECTRONIC OR MECHANICAL
 INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM

MS 87-08 Box 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

PROGRAMMING NOTE

A shift right of n is equivalent to dividing the contents of general register R1 by 2^n .

SHIFT RIGHT DOUBLE ARITHMETIC

DESCRIPTION

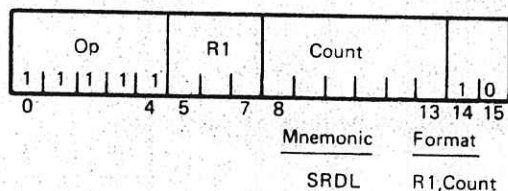
The contents of an even/odd pair of general registers (R1 and $R1 \oplus 1$) are shifted right as a 64-bit register. The number of positions shifted is specified by the shift count. Bits shifted out of bit position 31, of general register R1, are entered into bit position 0 of general register $R1 \oplus 1$. Bits equal to the sign are entered into vacated high-order bit positions. Bits shifted out of bit position 31 of general register $R1 \oplus 1$ are lost.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

SHIFT RIGHT DOUBLE LOGICAL

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

WITHOUT WRITING PERMISSION FROM THE
SPECIAL COLLECTIONS DIVISION
1000 W. WILSON AVENUE
WICHITA, KANSAS 67260-0001

DESCRIPTION

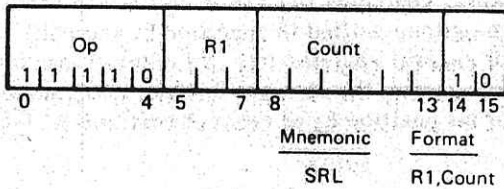
The contents of an even/odd pair of general registers (R1 and R1 ⊕ 1) are shifted right, as a 64-bit register. The number of positions shifted is specified by the shift count. Zeros are entered into all vacated high-order bit positions. Bits shifted out of bit position 31, of general register R1, are entered into bit position 0 of general register R1 ⊕ 1. Bits shifted out of bit position 31 of general register R1 ⊕ 1 are lost.

The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

SHIFT RIGHT LOGICAL



DESCRIPTION

The contents of general register R1 are shifted right the number of places indicated by the shift count. Zeros are entered into all vacated high-order bit positions. Bits shifted out of bit position 31 of general register R1 are lost.

RESULTING CONDITION CODE

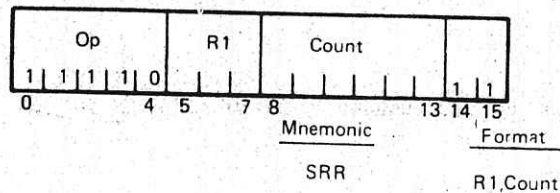
The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

SHIFT RIGHT AND ROTATE

DESCRIPTION

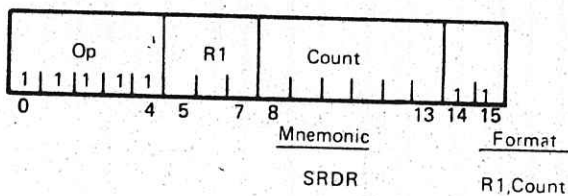
The contents of general register R1 are shifted right the number of places indicated by the shift count. Bits shifted out of bit position 31 are entered into bit position 0. The general register thus becomes a circular register and no bits are lost.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

SHIFT RIGHT DOUBLE AND ROTATE

DESCRIPTION

The contents of an even/odd pair of general registers (R1 and $R1 \oplus 1$) are shifted right, as a 64-bit register. The number of positions shifted is specified by the shift count. Bits shifted out of bit position 31 of general register R1 are entered into bit position 0 of general register $R1 \oplus 1$. Bits shifted out of bit position 31 of general register $R1 \oplus 1$ are entered into bit position 0 of general register R1. Thus, the two registers become a single, circular, 64-bit register, and no bits are lost.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

When the shift count equals 32, the contents of general register R1 and R1 ⊕ 1 are exchanged.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
REPRODUCED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED IN WASHINGTON, NOT PLACED IN WAREHOUSE

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

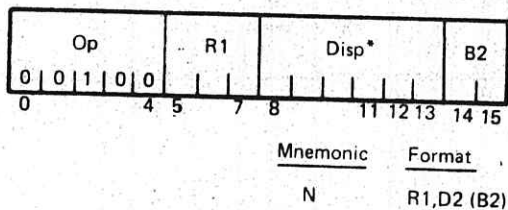
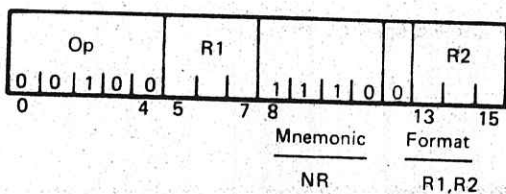
Section 7

LOGICAL OPERATIONS

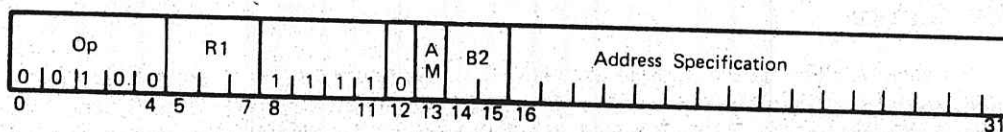
A set of instructions is provided for the logical manipulation of data. Fullword operands consist of 32 bits. Halfword immediate and storage operands are developed into fullword operands by appending 16 low-order zeros. The sign position is treated in the same manner as any other position.

There is no interdependence between bits for logical operations: that is, the result in position i is independent of bit j in either operand when $i \neq j$.

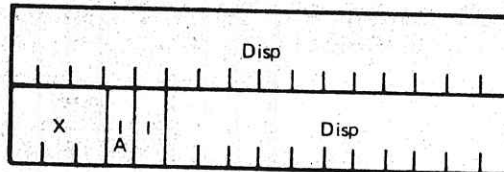
AND



* Displacements of the form 111XXX are not valid.



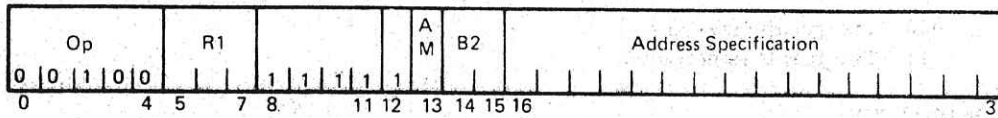
| | | | |
|-----------|----|-----------|----------------|
| Extended: | AM | Mnemonic | Format |
| | 0 | N | R1,D2 (B2) |
| Indexed: | 1 | N [@] [#] | R1,D2 (X2, B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR BE PLACED IN ANY INFORMATION SYSTEM

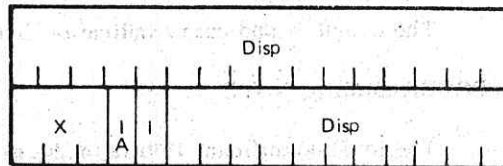
MS 87-08 Box 1042 FF 48
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

AND TO STORAGE



Extended: AM 0 Mnemonic NST Format R1,D2(B2)

Indexed: 1 NST [@] [#] R1,D2(X2,B2)



DESCRIPTION

The logical product (AND) of the fullword second operand and the contents of general register R1 is formed bit-by-bit. The result replaces the second operand. The contents of the general register is not changed. The following table defines the AND operation.

| AND | |
|---------|---------|
| Storage | 1 1 0 0 |
| R1 | 1 0 1 0 |
| Result | 1 0 0 0 |

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WITH-OUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY INFORMATION SYSTEM

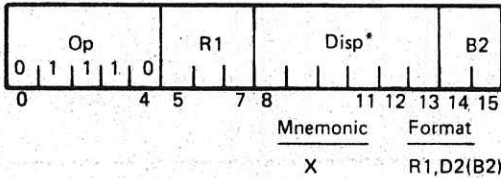
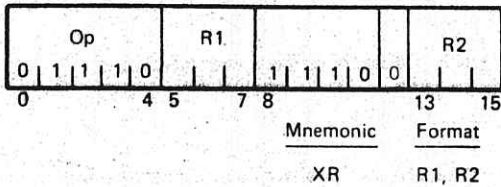
MS 87-08

Box 42

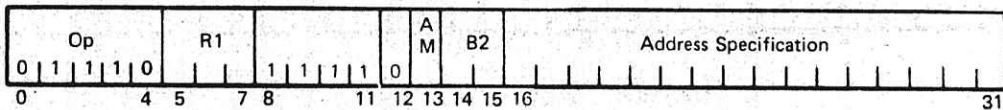
FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

EXCLUSIVE OR

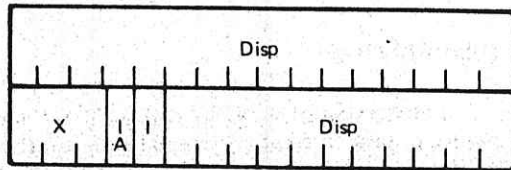


* Displacements of the form 111XXX are not valid.



Extended: AM Mnemonic Format
 0 X R1,D2(B2)

Indexed: 1 X (@) (#) R1,D2(X2,B2)



DESCRIPTION

The modulo-two sum (Exclusive OR), of the fullword second operand and the contents of general register R1, is formed bit-by-bit. The result replaces the contents of general register R1. The second operand is not changed. The following table defines the Exclusive OR operation.

| Exclusive OR | |
|--------------|---------|
| Storage | 1 1 0 0 |
| R1 | 1 0 1 0 |
| Result | 0 1 1 0 |

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17, U.S. CODE)
 COPY PROVIDED BY:
 MICHIGAN STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 REPRODUCTION PERMISSIONS UNIT
 300 SOUTH ZEEB AVENUE
 EAST LANSING, MICHIGAN 48824

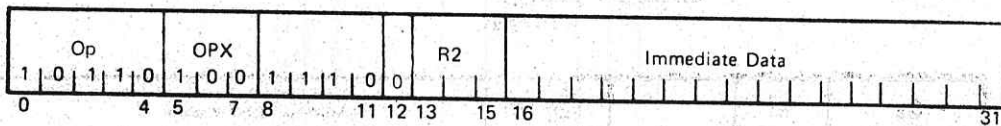
MS 87-08 Box 404 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Michiana State University Libraries, Special Collections and University Archives

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

The one's complement of the general register is obtained when the second operand contains all ones.

EXCLUSIVE OR HALFWORD IMMEDIATE

| Mnemonic | Format |
|----------|---------|
| XHI | R2,Data |

DESCRIPTION

Instruction bits 16 through 31 are treated as immediate data. The halfword of immediate data is first developed into a fullword by appending 16 low-order zeros. The modulo-two sum (Exclusive OR) of this fullword operand and contents of general register R2 is formed bit-by-bit. The result replaces the contents of general register R2. The immediate operand is not changed. The following table defines the Exclusive OR operation.

| Exclusive OR | |
|----------------|---------|
| Immediate Data | 1 1 0 0 |
| R2 | 1 0 1 0 |
| Result | 0 1 1 0 |

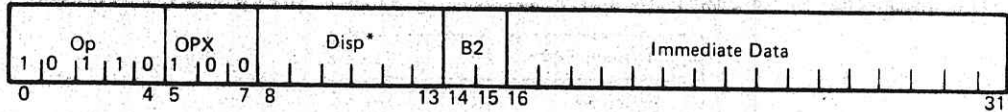
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

EXCLUSIVE OR IMMEDIATE WITH STORAGE



Mnemonic Format * Displacements of the form
 XIST D2(B2),Data 111XXX are invalid.

DESCRIPTION

Bits 16 through 31 of this instruction are treated as halfword immediate data. The modulo-two sum (Exclusive OR) of this halfword immediate data and the halfword main storage operand is formed bit-by-bit. The result replaces the halfword main storage operand.

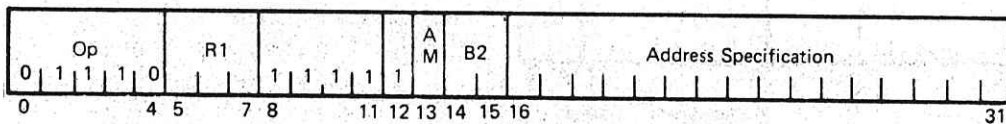
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

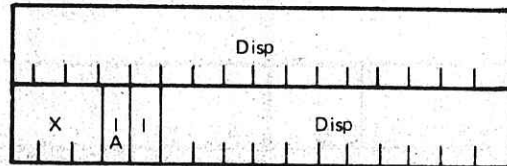
The overflow and carry indicators are not changed by this instruction.

EXCLUSIVE OR TO STORAGE



Extended: AM Mnemonic Format
 0 XST R1,D2(B2)

Indexed: 1 XST[@] [#] R1,D2(X2,B2)



THIS MATERIAL MAY BE PROTECTED BY COPYRIGHT LAW (TITLE 17 U.S. CODE)
 WICHITA STATE UNIVERSITY LIBRARIES SPECIAL COLLECTIONS
 DR. JAMES E. TOMAYKO COLLECTION OF NASA DOCUMENTS
 WICHITA STATE UNIVERSITY LIBRARIES, SPECIAL COLLECTIONS AND UNIVERSITY ARCHIVES
 MS 87-08 Box 1042 FF 46

DESCRIPTION

The modulo-two sum (Exclusive OR) of the fullword second operand and the contents of general register R1 is formed bit-by-bit. The result replaces the second operand. The contents of the general register is not changed. The following table defines the Exclusive OR operation.

| Exclusive OR | |
|--------------|---------|
| Storage | 1 1 0 0 |
| R1 | 1 0 1 0 |
| Result | 0 1 1 0 |

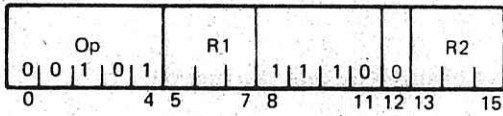
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

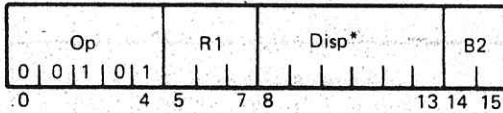
INDICATORS

The overflow and carry indicators are not changed by this instruction.

OR

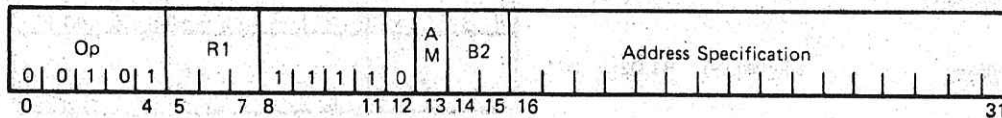


Mnemonic: OR
Format: R1,R2

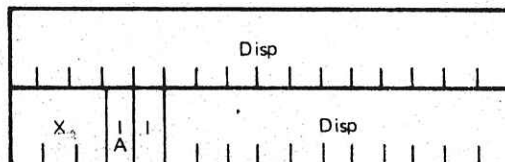


* Displacements of the form 111XXX are not valid.

Mnemonic: 0
Format: R1,D2(B2)



Extended: AM 0 Mnemonic 0 Format R1,D2(B2)
Indexed: 1 0 [@] [#] R1,D2(X2,B2)



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER NOR PLACED IN ANY PUBLIC DOMAIN

MS 87-08 BOX 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The logical sum (OR) of the fullword second operand and the contents of general register R1 is formed bit-by-bit. The result replaces the contents of general register R1. The second operand is not changed. The following table defines the OR operation.

| OR | |
|---------|---------|
| Storage | 1 1 0 0 |
| R1 | 1 0 1 0 |
| Result | 1 1 1 0 |

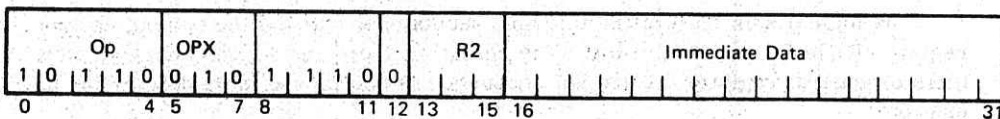
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

OR HALFWORD IMMEDIATE



Mnemonic: OHI
Format: R2,Data

DESCRIPTION

Instruction bits 16 through 31 are treated as immediate data. The halfword of immediate data is first developed into a fullword operand by appending 16 low-order zeroes. The logical sum (OR) of the fullword operand and the contents of general register R2 is formed bit-by-bit. The result replaces the contents of general register R2. The immediate operand is not changed. The following table defines the OR operation.

| OR | |
|----------------|---------|
| Immediate Data | 1 1 0 0 |
| R2 | 1 0 1 0 |
| Result | 1 1 1 0 |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHTIA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR BY ANY MEANS

MS 87-08
 Box 46
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

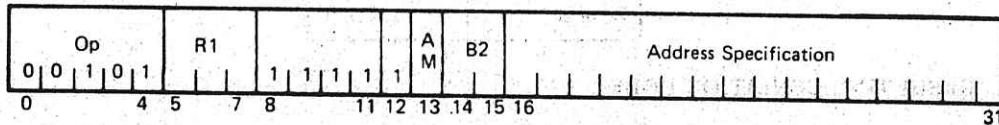
RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

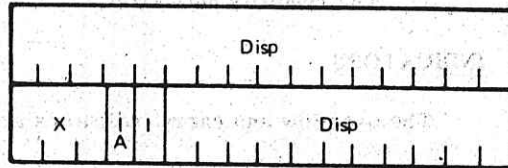
The overflow and carry indicators are not changed by this instruction.

OR TO STORAGE



Extended: $\frac{AM}{0}$ Mnemonic OST Format R1,D2(B2)

Indexed: 1 OST [@] [#] R1,D2(X2,B2)



DESCRIPTION

The logical sum (OR) of the fullword second operand and the contents of general register R1 is formed bit-by-bit. The result replaces the second operand. The contents of general register R1 are not changed. The following table defines the OR operation.

| OR | |
|---------|---------|
| Storage | 1 1 0 0 |
| R1 | 1 0 1 0 |
| Result | 1 1 1 0 |

RESULTING CONDITION CODE

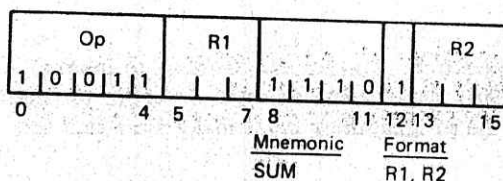
- 00 The result is zero
- 11 The result is not zero.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY.

MS 87-08 Box 42 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

SEARCH UNDER MASK

DESCRIPTION

A variable search of an array under control of fields in a mask for specific bit patterns is performed. A two's complement 16-bit integer count is contained in bits 0 through 15 of the general register specified by R2. (This must be a positive number for correct execution of this instruction).

The address of an array (A_i) is contained in bits 0 through 15 of the even general register of the even/odd pair specified by R1. A two's complement integer modifier is contained in bits 16 through 31. After each A_i has been located via bits 0 through 15, the modifier is added to the most-significant 16 bits of general register R1. This result replaces the most-significant 16 bits. The modifier is not changed. A 16-bit mask (M) is contained in bits 0 through 15 of the odd general register specified by R1 $\oplus 001$ while field values (FV) are contained in bits 16 through 31.

The following equation is solved.

$$(A_i \wedge M) \oplus (FV \wedge M)$$

where

$i = 1, \dots, \text{count}$

\wedge = logical AND function

\oplus = logical Exclusive-OR function.

$A_i \wedge M$ extracts bits selected by the mask out of the array. $FV \wedge M$ extracts bits selected by the mask also. These latter bits are compared with $A_i \wedge M$. If they are equal, the comparison continues until the count is exhausted. The condition code reflects the result of this operation.

If the comparison indicates an inequality, the instruction is terminated with the address of the inequality operand located in general register R1.

RESULTING CONDITION CODE

- 00 All array items matched
- 11 An array item miss-matched and general register R1 has the address where it failed.

INDICATORS

The overflow and carry are not changed by this instruction.

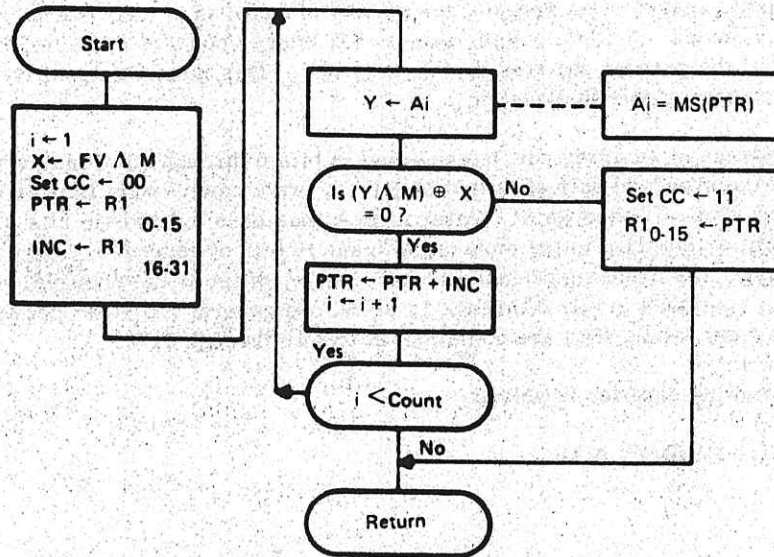
FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
COPY PROVIDED BY
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER NOR PLACED IN ANY MEDIUM

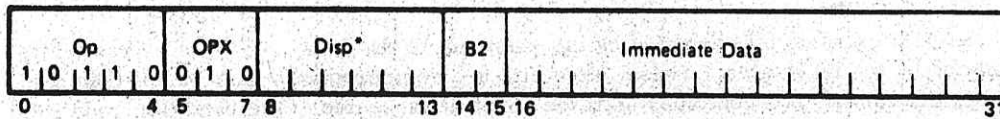
PROGRAMMING NOTE

This is a variable length instruction execution. Care must be taken to insure proper interrupt response by using sufficiently small count values. In order to assure proper completion of the putaway routine, the programmer must make sure that the count values do not exceed eight.

The following flowchart indicates how this instruction is executed:



SET BITS



| | | |
|-----------------|---------------|---|
| <u>Mnemonic</u> | <u>Format</u> | * Displacements of the form 111XXX are invalid. |
| SB | D2(B2),Data | |

DESCRIPTION

Bits 16 through 31 of this instruction are treated as halfword immediate data. The logical sum (OR) of the immediate data and the halfword main storage operand is formed bit-by-bit. The result replaces the halfword main storage operand.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 REPRODUCED BY ANY PERSON
 WITHOUT WRITTEN PERMISSION
 FROM THE NATIONAL ARCHIVES
 TITLE 17 U.S. CODE
 COPYRIGHT LAW
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITH OUT WRITTEN PERMISSION
 FROM THE NATIONAL ARCHIVES
 REPRODUCED BY ANY PERSON
 WITHOUT WRITTEN PERMISSION
 FROM THE NATIONAL ARCHIVES

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

INDICATORS

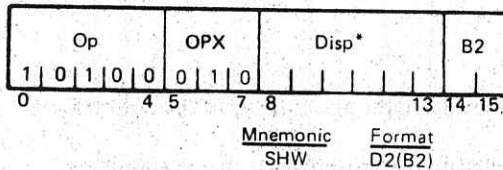
The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

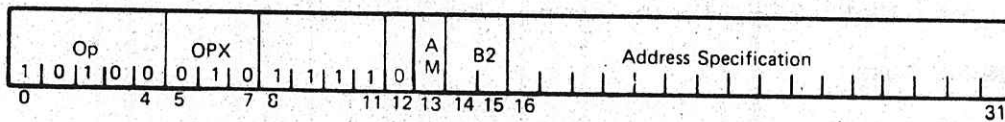
The one bits in the halfword mask specify the bits of the halfword second operand that are set one. The result replaces the halfword second operand. The following table defines this instruction.

| SET BITS | |
|----------|---------|
| Mask | 1 1 0 0 |
| Storage | 1 0 1 0 |
| Result | 1 1 1 0 |

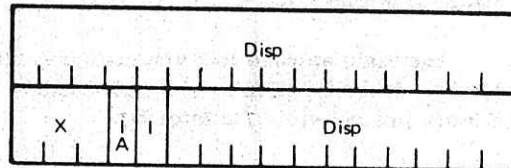
SET HALFWORD



* Displacements of the form 111XXX are not valid.



| | | | |
|-----------|---------|-----------------|------------------|
| Extended: | AM 0 | Mnemonic SHW | Format D2(B2) |
| Indexed: | 1 | SHW[@] [=] | D2(X2,B2) |



DESCRIPTION

The halfword main storage operand is set to all ones.

RESULTING CONDITION CODE

The condition code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 MICHIGAN STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

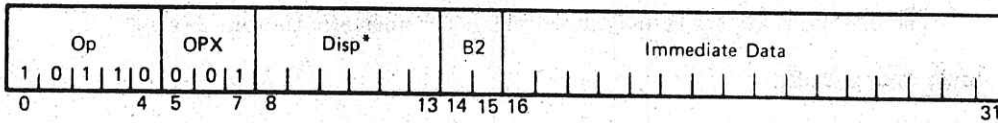
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

This instruction is the same as the TEST BITS instruction with the mask equal to all ones.

ZERO BITS



Mnemonic Format * Displacements of the form 111XXX are invalid.
 ZB D2(B2),Data

DESCRIPTION

The logical complement of bits 16 through 31 of this instruction is ANDed to the halfword main storage operand and is formed bit-by-bit. The result replaces the halfword main storage operand.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

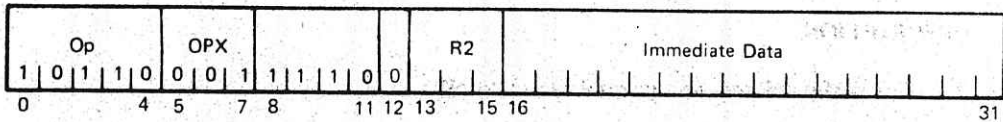
The one bits in the halfword immediate data specify the bits of the halfword main storage operand that are set zero. The result replaces the halfword main storage operand. The following table defines this instruction:

| ZERO BITS | |
|----------------|---------|
| Immediate Data | 1 1 0 0 |
| Storage | 1 0 1 0 |
| Result | 0 0 1 0 |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE
 WICHITA STATE UNIVERSITY LIBRARIES

MS 87-08 Box 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

ZERO REGISTER BITS



| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| ZRB | R2,Data |

DESCRIPTION

First, the halfword immediate data is expanded to a fullword by appending 16 low-order zeros. The logical complement of this fullword is then ANDed to the contents of general register R2. The result replaces general register R2.

RESULTING CONDITION CODE

- 00 The result is zero
- 11 The result is not zero.

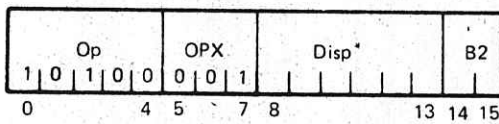
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

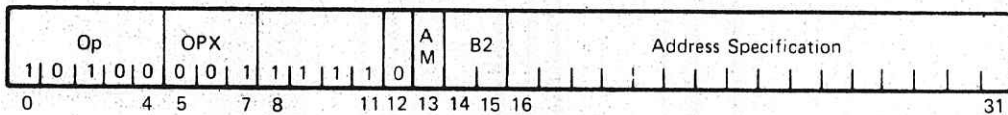
The one bits in the halfword immediate data specify the bits in the general register that are set zero. Bits 16 through 31 of general register R2 are not changed by this instruction.

ZERO HALFWORD

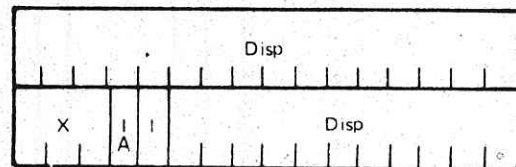


* Displacements of the form 111XXX are not valid.

| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| ZH | D2(B2) |



| | <u>AM</u> | <u>Mnemonic</u> | <u>Format</u> |
|-----------|-----------|-----------------|---------------|
| Extended: | 0 | ZH | D2 (B2) |
| Indexed: | 1 | ZH(@) [=] | D2(X2,B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL IS NOT TO BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 1092 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The halfword second operand is set to all zeros.

RESULTING CONDITION CODE

The condition code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

This instruction is similar to the ZERO BITS instruction with the mask equal to all ones.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT
REPRODUCED IN ANY FORM OR BY ANY MEANS

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FLOATING-POINT OPERATIONS

The floating-point instruction set is used to perform calculations on operands with a wide range of magnitude and to yield results scaled to preserve precision.

A floating-point number consists of a signed exponent and a signed fraction. The quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the exponent. The exponent is expressed in excess 64 binary notation; the fraction is expressed as a sign-magnitude hexadecimal number having a radix point to the left of the high order digit.

The floating-point instruction set provides for loading, adding, subtracting, comparing, multiplying, dividing, and storing. Short operands generally provide faster processing and require less storage than long operands. On the other hand, long operands provide greater precision in computation. Operations may be either register to register or storage to register. All floating-point instructions are part of the floating-point feature including the two data conversion instructions. A normalized number is one in which the high-order hexadecimal digit of the fraction is not zero or the fraction is all zero and the characteristic is the smallest possible value (zero).

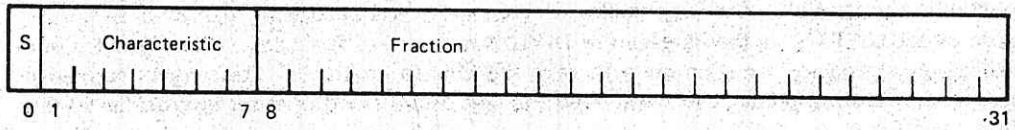
Maximum precision is preserved in addition, subtraction, multiplication, and division because all results are normalized.

The condition code is set as a result of all compare, add, subtract, and load operations.

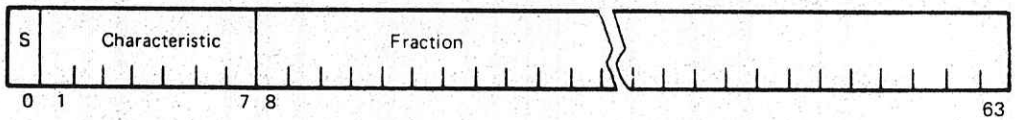
DATA FORMAT

Floating-point data occupy a fixed-length format which may be either a fullword short format or a double word long format. Both formats may be used in main storage.

Short Floating-Point Number



Long Floating-Point Number



FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

MS 87-08 Box 4042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY:

WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE

The first bit in either format is the sign bit (S). The subsequent seven bit positions are occupied by the characteristic. The fraction field may have either six or fourteen hexadecimal digits.

Although final results have six fraction hexadecimal digits in short-precision, intermediate results may have one additional low-order digit. This low-order digit, the guard digit, increases the precision of the final result.

NUMBER REPRESENTATION

The fraction of a floating-point number is expressed in hexadecimal digits. The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit. To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1 through 7 of both floating-point formats, indicates this power. The bits within the characteristic field can represent numbers from 0 through 127. To accommodate large and small magnitudes, the characteristic is formed by adding 64 to the actual exponent. The range of the exponent is thus -64 through +63. This technique produces a characteristic in excess 64 notation.

Both positive and negative quantities have a true fraction, the difference in sign being indicated by the sign bit. The number is positive or negative accordingly as the sign bit is zero or one.

The range covered by the magnitude (M) of a normalized floating-point number is:

in short precision $16^{-65} M (1-16^{-6}) \cdot 16^{63}$, and
 in long precision $16^{-65} M (1-16^{-14}) \cdot 16^{63}$,
 or approximately $5.4 \cdot 10^{-79} M 7.2 \cdot 10^{75}$.

The short and long precisions contain 6.2 and 15.5 decimal digits respectively.

A number with zero characteristic, zero fraction, and plus sign is called a true zero. A true zero may arise as the result of an arithmetic operation because of the particular magnitude of the operands. A true zero is forced when one or both operands of MULTIPLY or the dividend in DIVIDE has a zero fraction. The sign of a sum, difference, product, or quotient with zero fraction is positive. The proper representation of a floating point zero when used for any of the floating point operations is the true zero form.

MS 87-08

Box 1042

FF 46

NORMALIZATION

A quantity can be represented with the greatest precision by a floating-point number of given fraction length when that number is normalized. Therefore, all floating-point arithmetic operations require normalized operands. A normalized floating-point number has a nonzero high-order hexadecimal fraction digit. If one or more high-order fractional hexadecimal digits are zero, the number is said to be unnormalized unless it is a true zero. The process of normalization consists of shifting the fraction left until the high-order hexadecimal digit is nonzero and reducing the characteristic by the number of hexadecimal digits shifted. A zero fraction cannot be normalized, and its associated characteristic therefore remains unchanged when normalization is called for. A floating point word of all zeros is defined as a true zero.

Normalization usually takes place when the intermediate arithmetic result is changed to the final result. This function is called postnormalization, and it is performed as part of instruction execution.

PROGRAMMING NOTE

It is the programmer's responsibility to ensure that floating-point operands are normalized prior to instruction execution. Since normalization applies to hexadecimal digits, the three high-order bits of a normalized number may be zero.

FLOATING-POINT SECOND OPERANDS

The short 32-bit second operand has a fullword effective address. The long 64-bit second operand must start at an even boundary halfword address. Figure 8-1 illustrates floating-point data placement in main storage.

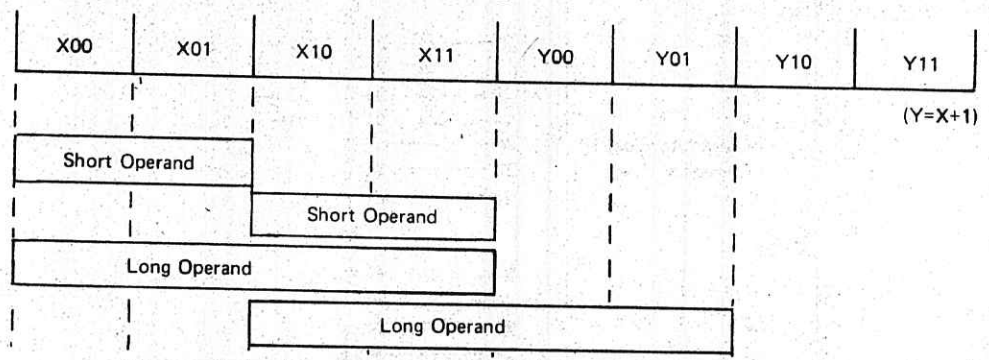


Figure 8-1. Floating-Point Second Operand in Main Storage

FLOATING-POINT REGISTERS

The registers used for floating-point arithmetic are distinct or separate registers from those used for fixed-point arithmetic. Register designation may be even or odd for short operands.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL WILL NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS

MS 87-08 Box 1042 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

The first operand is contained in floating-point register R1 when the second operand is a short 32-bit operand. If the second operand is a long or extended operand, the first operand is contained in the pair of floating-point registers specified by R1 and $R1 \oplus 001$, where \oplus indicates the Logical OP function. See Figure 8-2.

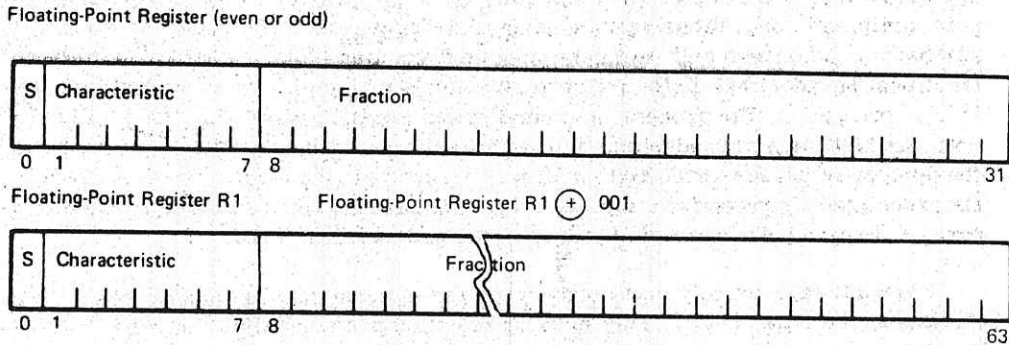


Figure 8-2. Floating-Point Operands in Registers

A comprehensive set of floating-point instruction is available for both short and long operands. Figure 8-3 summarizes the various combinations of fractional precision used for the floating-point operands. For further detail, see the individual instructions.

| Instructions | Short 2nd Operand | | Long 2nd Operand | |
|---------------------|-------------------|----------------|------------------|----------------|
| | Operand | | Operand | |
| | Result | 1 2 | Result | 1 2 |
| RRs | | | | |
| A/S | 24 ← | 24 ± 24 | 56 ← | 56 ± 56 |
| C | | $24 : 24$ | | |
| M | 24/48 ← | 24×24 | 56 ← | 31×31 |
| D | 24 ← | $24 \div 24$ | 31 ← | $56 \div 31$ |
| Convert to Floating | 24 ← | 32 | | |
| Convert to Fixed | 32 ← | 24 | | |
| L | 24 ← | 24 | | |
| SRSs | | | | |
| A/S | 24 ← | 24 ± 24 | | |
| M | 24/48 ← | 24×24 | | |
| D | 24 ← | $24 \div 24$ | | |
| L | 24 ← | 24 | | |
| ST | 24 → | 24 | | |
| RSs | | | | |
| A/S | 24 ← | 24 ± 24 | 56 ← | 56 ± 56 |
| C | | $24 : 24$ | | |
| M | 24/48 ← | 24×24 | 56 ← | 31×31 |
| D | 24 ← | $24 \div 24$ | 31 ← | $56 \div 31$ |
| L | 24 ← | 24 | 56 ← | 56 |
| ST | 24 → | 24 | 56 → | 56 |

Figure 8-3. Combinations of Fractional Precision for Floating-Point Operands

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF
 WICHITA STATE UNIVERSITY LIBRARIES

MS 87-08 Box 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

FLOATING-POINT INSTRUCTIONS

The floating-point arithmetic instructions and their mnemonics, and descriptions follow. The following table indicates when the condition code is set and the exceptions in operand designations, data, or results that cause a program interruption.

| Name | Mnemonic | Type | Exceptions |
|--|---------------------------------|------------|------------|
| Add (Long Operands) | AEDR | RR C | U, E, S |
| Add (Long Operands) | AED | RS C | U, E, S |
| Add (Short Operands) | AER | RR C | U, E, S |
| Add (Short Operands) | AE | SRS, RS C | U, E, S |
| Compare (Short Operands) | CER | RR C | |
| Compare (Short Operands) | CE | RS C | |
| Convert to Fixed-Point | CVFX | RR C | O |
| Convert to Floating-Point | CVFL | RR C | S |
| Divide (Extended Operands) | DEDR | RR | U, E, FK |
| Divide (Extended Operands) | DED | RS | U, E, FK |
| Divide (Short Operands) | DER | RR | U, E, FK |
| Divide (Short Operands) | DE | SRS, RS | U, E, FK |
| Load (Long Operands) | LED | RS C | |
| Load (Short Operands) | LE | SRS, RS C | |
| Load (Short Operands) | LER | RR C | |
| Load Complement (Short Operands) | LECR | RR C | |
| Load Fixed Register | LFXR | RR | |
| Load Floating Immediate (Short Operands) | LFLI | RR | |
| Load Floating Register (Short Operands) | LFLR | RR | |
| Mid Value Select (Short Operands) | MVS | RS C | |
| Multiply (Extended Operands) | MEDR | RR | U, E |
| Multiply (Extended Operands) | MED | RS | U, E |
| Multiply (Short Operands) | MER | RR | U, E |
| Multiply (Short Operands) | ME | SRS, RS | U, E |
| Store (Long Operands) | STED | RS | |
| Store (Short Operands) | STE | SRS, RS | |
| Subtract (Long Operands) | SEDR | RR C | U, E, S |
| Subtract (Long Operands) | SED | RS C | U, E, S |
| Subtract (Short Operands) | SER | RR C | U, E, S |
| Subtract (Short Operands) | SE | SRS, RS, C | U, E, S |
| Notes | | | |
| C | Condition code is set | | |
| E | Exponent-overflow exception | | |
| FK | Floating-point divide exception | | |
| O | Overflow | | |
| S | Significance exception | | |
| U | Exponent-underflow exception | | |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPIES PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08

Box 4042

FF 46

CONDITION CODE

The results of floating-point add, compare, subtract, convert, load, and mid-value select operations are used to set the condition code. Multiplication, division, and storing leave the code unchanged. The condition code can be used for decision-making by subsequent branch-on condition instructions.

The condition code can be set to reflect the type of results for floating-point arithmetic. The states 00, 11, or 01 indicate that the result is zero, less than zero, or greater than zero. A zero result is indicated whenever the result fraction is zero, including a forced zero. State 10 is never set by floating-point operations. The compare instruction indicates the relative arithmetic magnitude of the first operand (R1) and the second operand (called $\phi 2$). (See Figure 8-4).

| | 00 | 11 | 01 |
|------------------|---------------------|---------------------|---------------------|
| Add S/L | zero | < zero | > zero |
| Compare S/L | (R1) = ($\phi 2$) | (R1) < ($\phi 2$) | (R1) > ($\phi 2$) |
| Load S/L | zero | < zero | > zero |
| Subtract S/L | zero | < zero | > zero |
| Converts | zero | < zero | > zero |
| Mid Value Select | within | above | below |

Figure 8-4. Condition Code Setting for Floating-Point Arithmetic

INDICATORS

The overflow and carry indicators are not changed by floating-point instructions.

FLOATING-POINT ARITHMETIC EXCEPTIONS

Invalid operation codes, operand designations, data, or results cause a program interruption. When the interruption occurs, the current PSW is stored as an old PSW, and a new PSW is obtained. The interruption code in the old PSW identifies the cause of the interruption. The following exceptions cause a program interruption in floating-point arithmetic.

Protection: Each halfword in main storage can be protected with a storage protection bit. The operation is terminated on a store violation.

Addressing: An address designates an operand location outside the available storage for the installed system. In most cases, the operation is terminated. The result data and the condition code, if affected, are unpredictable and should not be used for further computation.

Exponent Overflow: The result exponent in addition, subtraction, multiplication, or division exceeds 127 (16^{63}), and the result fraction is not zero. The operation is terminated and a program interrupt occurs.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT

(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED IN WHOLE OR IN PART FROM THE
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION. THIS MATERIAL MAY BE
REPRODUCED IN WHOLE OR IN PART FROM THE
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

MS 87-08

Box 42

FF 46

Exponent Underflow: The result exponent in addition, subtraction, multiplication, or division is less than zero (16^{-64}), and the result fraction is not zero. The operation is terminated, and a program interruption occurs if the exponent-underflow mask bit (PSW bit 22) is one.

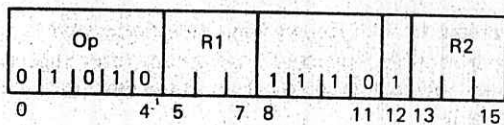
The setting of the exponent-underflow mask also affects the result of the operation. When the mask bit is zero, the sign, exponent, and fraction are set to zero, thus making the result a true zero. When the mask bit is one, the fraction and exponent results are unpredictable.

Significance: The result fraction of an addition, subtraction, certain multiplies by zero or convert to floating-point is zero. A program interruption occurs if the significance mask bit (PSW bit 23) is one. The mask bit affects also the result of the operation. When the significance mask bit is a zero, the operation is completed by replacing the result with a true zero. When the significance mask bit is one, the operation is completed without further change to the characteristic of the result. In either case, the condition code is set to 00.

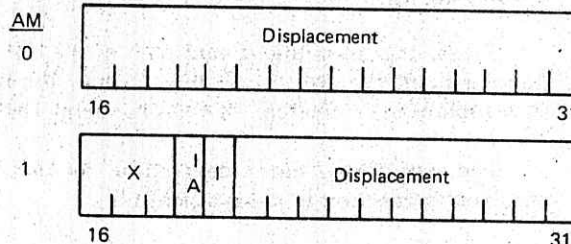
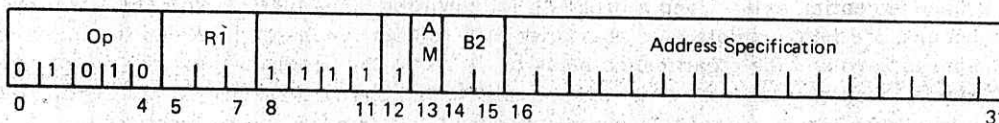
Floating-Point Divide: When division by a true zero is attempted, the division is suppressed. The condition code and data in registers and storage remain unchanged.

Un-normalized Inputs for Divide: When division is performed with un-normalized inputs, the un-normalized inputs interrupt will occur. The exception to this rule occurs when the divisor is un-normalized and the final quotient characteristic exceeds 127. In this case, the exponent overflow interrupt will occur in lieu of the un-normalized input interrupt.

ADD (LONG OPERANDS)



Mnemonic Format
AEDR R1, R2



| | AM | Mnemonic | Format |
|-----------|----|-----------------|-----------------|
| Extended: | 0 | AED | R1, D2 (B2) |
| Indexed: | 1 | AED [@] [#] | R1, D2 (X2, B2) |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS

MS 87-08
Box 1042
FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The second operand is added to the first operand, and the normalized sum is placed in the first operand location.

The long 64-bit second operand is added with the contents of the even/odd floating-point-register pair specified by the even register R1. The normalized result is placed into even/odd floating-point register R1.

Addition of two floating-point numbers consists of a characteristic comparison and a fraction addition. The characteristics of the two operands are compared, and the fraction characteristics of the two operands are compared, and the fraction with the smaller characteristic is right-shifted; its characteristic is increased by one for each hexadecimal digit of shift, until the two characteristics agree. The fractions are then added algebraically to form an intermediate sum. If an overflow carry occurs, the intermediate sum is right-shifted one hexadecimal digit, and the characteristic is increased by one. If this increase causes a characteristic overflow, an exponent-overflow exception is signaled, and a program interruption occurs.

The long intermediate sum consists of 15 hexadecimal digits and a possible carry.

After the addition, the intermediate sum is left-shifted as necessary to form a normalized fraction; vacated low-order digit positions are filled with zeros and the characteristic is reduced by the amount of shift.

If normalization causes the characteristic to underflow, characteristic and fraction are made zero, an exponent-underflow exception exists, and a program interruption occurs if the corresponding mask bit is one. If no left shift takes place the intermediate sum is truncated to the proper fraction length.

When the intermediate sum is zero and the significance mask bit is one, a significance exception exists, and a program interruption takes place. No normalization occurs; the intermediate sum characteristic remains unchanged. When the intermediate sum is zero and the significance mask bit is zero, the program interruption for the significance exception does not occur; rather, the characteristic is made zero, yielding a true zero result. Exponent underflow does not occur for a zero fraction.

First, the least-significant part of the intermediate sum replaces the contents of floating-point register R1 \oplus 001. Then, the most significant part of the intermediate sum replaces the contents of floating-point register R1

The sign of the sum is derived by the rules of algebra. The sign of a sum with zero result fraction is always positive.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17, U.S. CODE)
COPY PROVIDED BY
WICHITA STATE UNIVERSITY LIBRARIES

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITH-OUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED OR
REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08

Box 404

FF 46

RESULTING CONDITION CODE

- 00 Result fraction is zero
- 11 Result is less than zero
- 01 Result is greater than zero

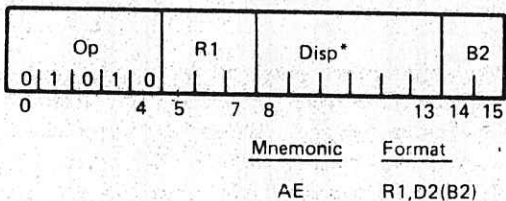
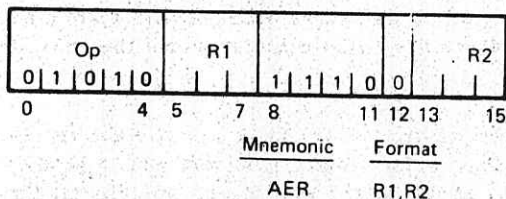
PROGRAM INTERRUPTIONS

- Significance
- Exponent Overflow
- Exponent Underflow

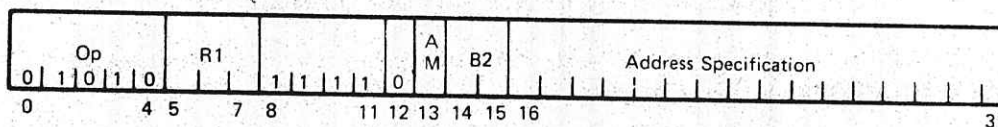
PROGRAMMING NOTE

Interchanging the two operands in a floating-point addition does not affect the value of the sum.

ADD (SHORT OPERANDS)

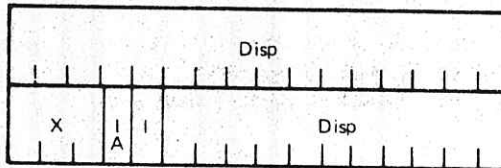


* Displacements of the form 111XXX are not valid.



Extended: $\frac{AM}{0}$ Mnemonic AE Format R1,D2(B2)

Indexed: 1 AE (@) [=] R1,D2(X2,B2)



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER OR PLACED IN ANY REPOSITORY

Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives
 MS 87-08 Box 4042 FF 46

DESCRIPTION

The short second operand is added to the short first operand, and the six digit normalized sum is placed in the first operand location.

Addition of two floating-point numbers consists of a characteristic comparison and a fraction addition. The characteristics of the two operands are compared, and the fraction with the smaller characteristic is right-shifted; its characteristic is increased by one for each hexadecimal digit of shift, until the two characteristics agree. The fractions are then added algebraically to form an intermediate sum. If an overflow carry occurs, the intermediate sum is right-shifted one digit, and the characteristic is increased by one. If this increase causes a characteristic overflow, an exponent-overflow exception is signaled, and a program interruption occurs.

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

After the addition, the intermediate sum is left-shifted as necessary to form a normalized fraction, vacated low-order digit positions are filled with zeros and the characteristic is reduced by the amount of shift.

If normalization causes the characteristic to underflow, characteristic and fraction are made zero, an exponent-underflow exception exists, and a program interruption occurs if the corresponding mask bit is one. If no left shift takes place, the intermediate sum is truncated to the proper fraction length.

When the intermediate sum is zero and the significance mask bit is one, a significance exception exists, and a program interruption takes place. No normalization occurs; the intermediate sum characteristic remains unchanged. When the intermediate sum is zero and the significance mask bit is zero, the program interruption for the significance exception does not occur; rather, the characteristic is made zero, yielding a true zero result. Exponent underflow does not occur for a zero fraction.

The sign of the sum is derived by the rules of algebra. The sign of a sum with zero result fraction is always positive.

RESULTING CONDITION CODE

- 00 Result fraction is zero
- 11 Result is less than zero
- 01 Result is greater than zero

FOR RESEARCH USE ONLY

MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITH-OUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR

REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF WICHITA STATE UNIVERSITY LIBRARIES

MS 87-08

Box 4042 FF 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17, U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED WITHOUT PERMISSION FROM WICHITA STATE UNIVERSITY LIBRARIES

RESULTING CONDITION CODE

- 00 Operands are equal
- 11 First operand is low
- 01 First operand is high

PROGRAMMING NOTE

Numbers with zero fraction compare equal even when they differ in sign or characteristic.

In comparing very small numbers (characteristic of 00 hexadecimal) which would result in an exponent underflow in a subtract instruction, the condition code will be set to 00 (equal) even though the number is visually not equal. For example, a comparison of 00100000 and 001FFFFF would yield a condition code of 00 (equal).

CONVERT TO FIXED-POINT

| | | | | | | | | | | | | | | |
|----------|---|---|---|----|---|---|---|--------|----|----|--|----|--|--|
| Op | | | | R1 | | | | R2 | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | | | |
| 0 | | 4 | 5 | | 7 | 8 | | 11 | 12 | 13 | | 15 | | |
| Mnemonic | | | | | | | | Format | | | | | | |
| CVFX | | | | | | | | R1, R2 | | | | | | |

DESCRIPTION

The second operand is normalized short 32-bit floating-point operand using the sign magnitude floating-point representation. The second operand is converted to fixed-point by an unnormalization operation in order to have its characteristic equal to a hexadecimal 44 [1000100 (2)]. Its sign bit is placed into the sign bit of general register R1. Next, bits 8 through 39 of the intermediate value are converted from sign-magnitude representation to two's complement and placed into bits 1 through 31 of general register R1.

A convert overflow occurs when a floating-point second operand is not properly converted to fixed-point. This occurs when the characteristic is larger than 44 hexadecimal 1000100 (2) or when bit 8 of the intermediate value is a 1 unless the number is negative and bits 9 through 31 are zero. The value of R1 is unchanged.

CONDITION CODE

- 00 Bits 0 through 15 of the result in general register R1 is zero.
- 11 Bits 0 through 15 of the result in general register R1 is negative
- 01 Bits 0 through 15 of the result in general register R1 is positive.

ANOMALY NOTE

A floating-point value of 41100000 is converted to a fixed-point 00010000 but gives a condition code of 00.

INDICATORS

The overflow and carry indicators are not changed.

PROGRAM INTERRUPTS

Convert overflow.

PROGRAMMING NOTE

Refer to the CONVERT TO FLOATING instruction.

CONVERT TO FLOATING-POINT

| | | | | | | | | | | | | | |
|----|---|---|----|---|--|---|-----------------|----|---------------|----|---|----|--|
| Op | | | R1 | | | | R2 | | | | | | |
| 0 | 0 | 1 | 1 | 1 | | | 1 | 1 | 1 | 0 | 1 | | |
| 0 | | | 4 | 5 | | 7 | 8 | 11 | 12 | 13 | | 15 | |
| | | | | | | | <u>Mnemonic</u> | | <u>Format</u> | | | | |
| | | | | | | | CVFL | | R1, R2 | | | | |

DESCRIPTION

The second operand is a 32-bit two's complement number with its binary point considered to be between bits 15 and 16. It is converted to sign magnitude floating-point representation and placed into floating-point register R1.

First, the sign bit of the fixed-point number is placed into the sign bit of the intermediate result shown below. Then, bits 0 through 31 of the fixed-point number are converted from two's complement representation to the magnitude of a sign-magnitude representation, and then placed into bits 8 through 39 of the intermediate result. The characteristic in bits 1 through 7 of the intermediate result is set to 1000100 (2). Finally, the resulting intermediate number is normalized and only a short floating-point representation (bits 0 through 31) is developed and placed into the floating point register R1.

CONDITION CODE

- 00 The floating-point result is zero.
- 11 The floating-point result is negative.
- 01 The floating-point result is positive (>0).

INDICATORS

The overflow and carry indicators are not changed by this instruction.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY

SPECIAL COLLECTIONS

UNWRITTEN PERMISSION. THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED WITHOUT PERMISSION FROM THE ARCHIVES

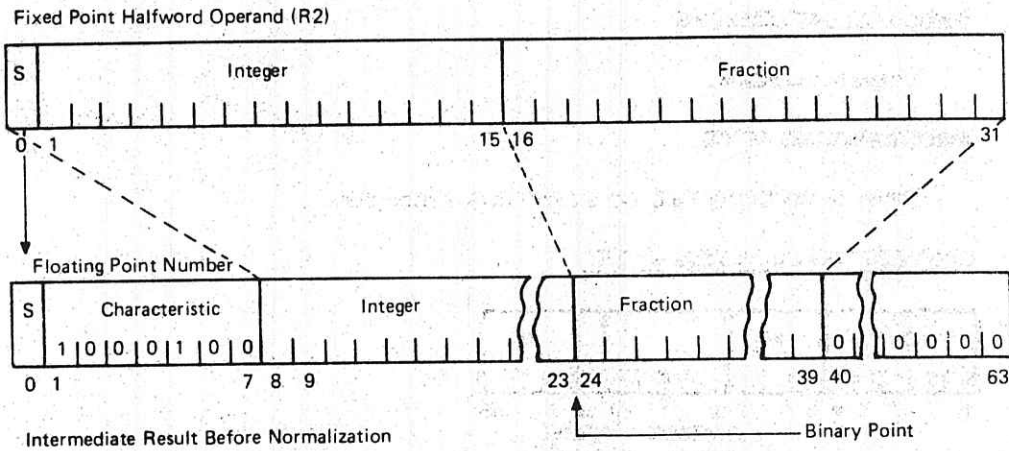
MS 87-08

Box 42

FF 46

PROGRAM INTERRUPT

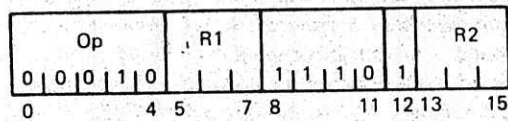
Significance



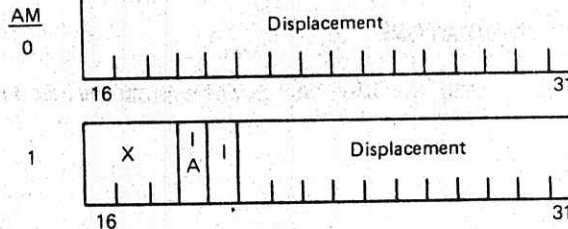
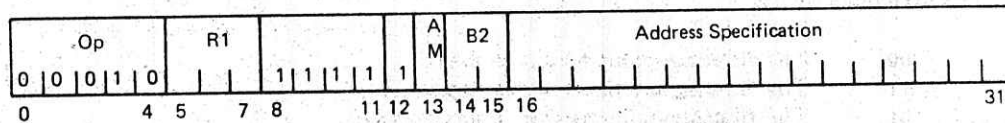
PROGRAMMING NOTE

Since the significance interrupt will occur when converting a zero, the programmer may want to mask this interrupt before doing a CVFL by setting the significance mask (bit 23 of the PSW) to zero. Thus, the significance interrupt would occur only for add or subtract floating, if not masked during the execution of those instructions.

DIVIDE (EXTENDED OPERANDS)



Mnemonic Format
DEDR R1, R2



| | AM | Mnemonic | Format |
|-----------|----|-------------|-----------------|
| Extended: | 0 | DED | R1, D2 (B2) |
| Indexed: | 1 | DED [@] [#] | R1, D2 (X2, B2) |

FOR RESEARCH USE ONLY
 MATERIAL MAY BE
 PROTECTED BY COPYRIGHT-LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED
 REPRODUCED, MANIPULATED, OR PLACED IN ANY MEDIUM

MS 87-08 Box 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The dividend (the long first operand) is divided by the divisor (the quasi-extended second operand) and replaced by the quotient. No remainder is preserved.

The first operand is located in bits 0 through 63 of the even/odd pair of floating point registers specified by R1. The first operand is divided by the divisor. This quasi-extended divisor is limited to 31 fraction bits. This quasi-extended divisor is formed from a long floating-point operand by truncating the fraction portion of the second operand to 31 bits and then rounding into the 31st bit based upon the 32nd bit. The quasi-extended quotient replaces the dividend. This quotient replaces bits 0 through 38 of the even/odd pair of floating-point registers specified by R1. (Bits 39 through 63 are set to zero.)

A floating-point division consists of a characteristic subtraction and a fraction division. The difference between the dividend and divisor characteristics plus 64 is used as an intermediate quotient characteristic. The sign of the quotient is determined by the rules of algebra.

Postnormalizing the intermediate quotient is never necessary with both dividend and divisor being normalized, but a right-shift may be called for. The intermediate quotient characteristic is adjusted for the shifts. All dividend fraction digits participate in forming the quotient, even if the normalized dividend fraction is larger than the normalized divisor fraction. The quotient fraction is truncated to 31 bits.

A program interruption for exponent overflow occurs when the final quotient characteristic exceeds 127 and the operation is terminated. This interruption will take precedence over all other program interruptions for this instruction.

A program interruption for exponent underflow occurs if the final-quotient characteristic is less than zero. The characteristic, sign, and fraction are made zero, and the interruption occurs if the corresponding mask bit is one. Underflow is not signaled for the intermediate quotient or for the operand characteristics during prenormalization.

When division by a true zero divisor is attempted, the operation is suppressed. The dividend remains unchanged, and a program interruption for floating-point divide occurs. When the dividend is a true zero, the quotient fraction will be zero. The quotient sign and characteristic are made zero, yielding a true zero result without taking the program interruptions for exponent underflow and exponent overflow. The program interruption for significance is never taken for division.

When division is performed with un-normalized inputs, the un-normalized inputs interrupt will occur.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR

REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08

Box 4042 FF 46

CONDITION CODE

The code remains unchanged.

PROGRAM INTERRUPTIONS

- Exponent Overflow
- Exponent Underflow
- Floating-Point Divide Exception
- Unnormalized inputs

PROGRAMMING NOTES

Fraction division proceeds as in fullword fixed-point division with formation of a 32-bit signed quotient using a 32-bit signed divisor and a 64-bit signed dividend. The magnitude of the dividend fraction is adjusted to ensure that the magnitude of the divisor exceeds the magnitude of the dividend. The quotient is converted to a normal extended precision floating point operand with low-order fraction bits set to zero.

Rounding of the quasi-extended divisor means adding the 32nd bit in the fraction part of the floating-point operand to the 31st bit and propagating all possible carries.

There are several cases when the quotient fraction may exceed 31 bits. These situations occur with specific data patterns. The quotient will be correct but the low-order fraction bits (39-63) will not be set to zero as stated in paragraph two of the description.

HARDWARE ANOMALY

1. Due to an anomaly in the microcode implementation of this instruction whereby internal status bit 21 is not cleared when there is a zero dividend, the programmer must take steps to correct or avoid that condition. Usually, the best way to do this is to test the dividend before executing the Divide and if it is zero, do not perform the Divide. Thus, status bit 21 will never be left set equal to one. Another alternative would be to calculate the reciprocal of the divisor, then multiply by the reciprocal instead of dividing.

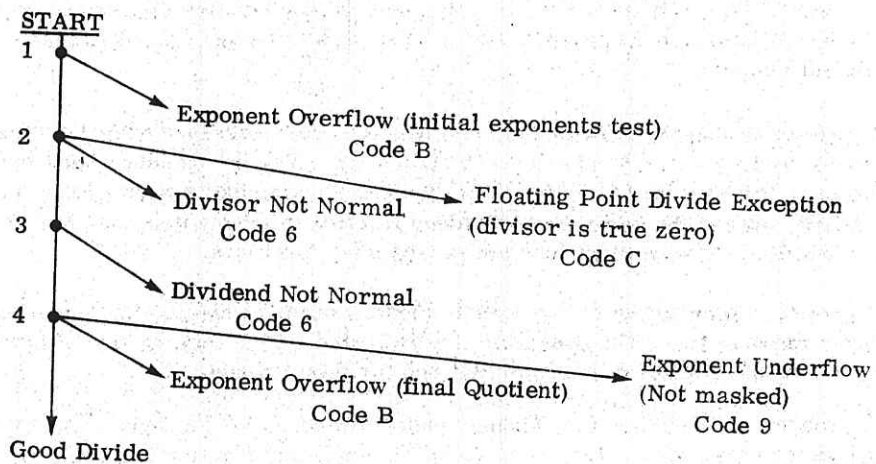
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER ON ANY MEDIA IN ANY REPOSITORY

MS 87-08 Box 42 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

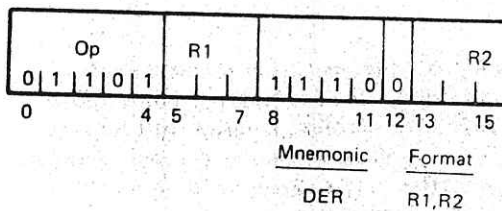
2. The extended form of the floating point divide (DED, DEDR) does not always produce a quotient which is accurate to 31 bits. The operands which would produce an incorrect result cannot be precisely defined; however, the following observations can be made:
 - a. If the divisor's fraction is less than hexadecimal .8000 0000, then the quotient will be correct.
 - b. If the divisor's fraction is greater than or equal to .8000 0000, then there exists a possibility of an inaccurate quotient.
 - c. The value of the dividend does not affect the accuracy of the result.
 - d. The inaccuracy can occur as early as bit 25 in the fraction (origin 0) and may be in any of the last seven bits, 25-31.
 - e. The short precision divide (DE, DER) does not have this problem.

For those situations where accuracy to the full 31 bit precision is required, it is recommended that reciprocals of constants be stored and the extended form of the floating point multiply be used instead of the divide. For those conditions where the divisor is a variable, it will be necessary to use a work-around to preserve the accuracy.

3. The divide instruction interrupt hierarchy for both long and short operands is given in the diagram below:



DIVIDE (SHORT OPERANDS)



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

When division is performed with un-normalized inputs, the un-normalized inputs interrupt will occur.

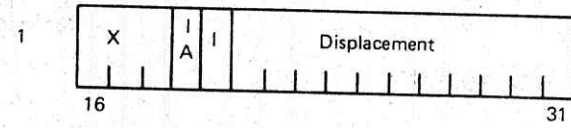
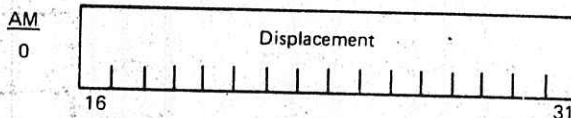
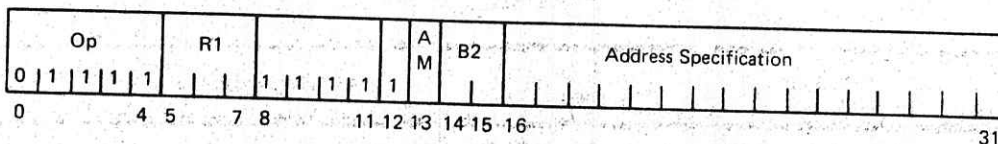
CONDITION CODE

The code remains unchanged.

PROGRAM INTERRUPTS

- Exponent Overflow
- Exponent Underflow
- Floating-Point Divide Exception
- Un-normalized Inputs

LOAD (LONG OPERANDS)



| | AM | Mnemonic | Format |
|-----------|----|-------------|-----------------|
| Extended: | 0 | LED | R1, D2 (B2) |
| Indexed: | 1 | LED [@] [#] | R1, D2 (X2, B2) |

DESCRIPTION

The long second operand is placed in the long first operand register. The second operand is not changed.

First, bits 32 through 63 of the doubleword main storage operand are loaded into floating-point register R1 ⊕ 001. Then, bits 0 through 31 of the doubleword main storage operand are loaded into floating-point register R1. Exponent overflow, exponent underflow, or lost significance cannot occur.

CONDITION CODE

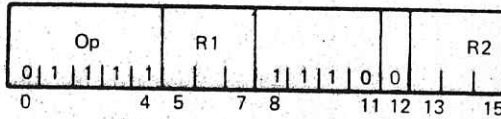
- 00 The second operand is a true zero
- 11 The second operand is negative
- 01 The second operand is positive (> 0)

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED WITHOUT THE WRITTEN PERMISSION OF THE
 WICHITA STATE UNIVERSITY LIBRARIES

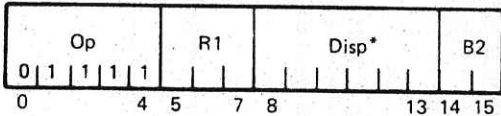
MS 87-08 Box 4042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17, U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

LOAD (SHORT OPERANDS)

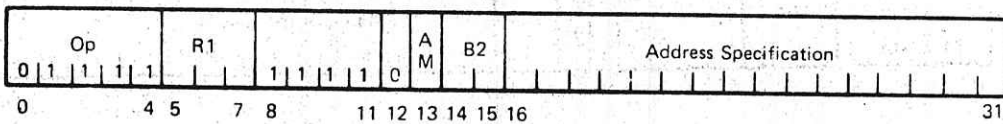


Mnemonic Format
 LER R1,R2

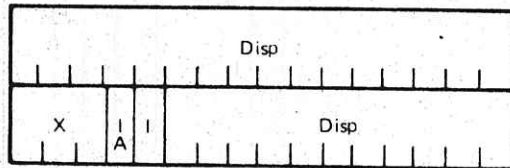


Mnemonic Format
 LE R1,D2(B2)

* Displacements of the form 111XXX are not valid.



| | | | |
|-----------|---------|------------------------|------------------------|
| Extended: | AM 0 | Mnemonic LE | Format R1,D2(B2) |
| Indexed: | 1 | Mnemonic LE (@) [≠] | Format R1,D2(X2,B2) |



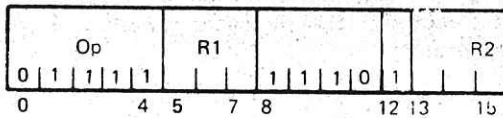
DESCRIPTION

The second operand is placed in floating-point register R1. The second operand is not changed. The overflow, underflow, and carry indicators are not changed by this instruction.

RESULTING CONDITION CODE

- 00 The second operand is a true zero
- 11 The second operand is negative
- 01 The second operand is positive (> 0)

LOAD COMPLEMENT (SHORT OPERANDS)



Mnemonic Format
 LECR R1,R2

MS 87-08 Box 4044 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The arithmetic complement of the fullword second operand replaces the contents of floating-point register R1. The sign bit of the second operand is inverted, while the characteristic, the fraction, and register R1 \oplus 001, are not changed. Indicators are unchanged by this instruction.

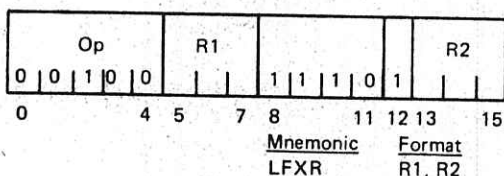
RESULTING CONDITION CODE

- 00 The result is a true zero
- 11 The result is negative
- 01 The result is positive (> 0)

PROGRAMMING NOTE

If this instruction is used to load a true zero, the condition code is set to 11 indicating a negative result and the result will equal hexadecimal 80000000. To avoid this condition, a test for zero operand should be made prior to the LECR and if the operand is zero, branch around the LECR.

LOAD FIXED REGISTER



DESCRIPTION

The fullword contents of the floating-point register specified by R2 is loaded into the general register specified by R1.

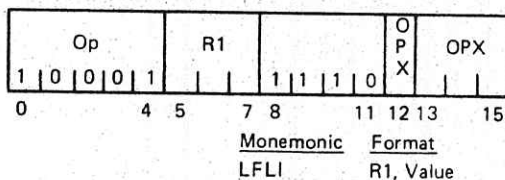
RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

LOAD FLOATING IMMEDIATE



DESCRIPTION

A floating-point immediate value is loaded into the floating-point register specified by R1.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

MS 87-08

Box 1042

#F 46

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

The immediate values are 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11., 12., 13., 14., and 15.

OPX (bits 12, 13, 14, 15)

(hex)

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- A
- B
- C
- D
- E
- F

Immediate Values → R1

(hex)

- 4100 0000
- 4110 0000
- 4120 0000
- 4130 0000
- 4140 0000
- 4150 0000
- 4160 0000
- 4170 0000
- 4180 0000
- 4190 0000
- 41A0 0000
- 41B0 0000
- 41C0 0000
- 41D0 0000
- 41E0 0000
- 41F0 0000

RESULTING CONDITION CODE

The code is not changed by this instruction.

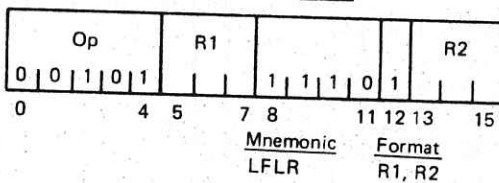
INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

The result of a LFLZ with zero immediate value does not produce a true zero result.

LOAD FLOATING REGISTER



DESCRIPTION

The fullword contents of the general register specified by R2 are loaded into the floating-point register specified by R1.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, ELECTRONIC OR MECHANICAL, INCLUDING
 PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND
 RETRIEVAL SYSTEM.

MS 87-08 BOX 46 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

The normalized product of multiplier (a quasi-extended second operand) and multiplicand (a quasi-extended first operand) replaces the multiplicand.

The first operand is located in bits 0 through 38 of the even/odd pair of floating-point register specified by the even register R1. This operand is multiplied by the second operand. For the RR format, the second operand is located in bits 0 through 38 of the even/odd pair of floating-point registers specified by R2. (Bits 39 through 63 do not participate except during rounding. See Programming Notes.) For the RS format, the second operand is located in bits 0 through 38 of the main storage extended operand. The extended product replaces bits 0 through 63 of the even/odd pair of floating-point registers specified by R1 and R1 \oplus 001.

The multiplication of two floating-point numbers consists of a characteristic addition and a fraction multiplication. (Participation of multiplicand and multiplier fraction bits is limited to 31 bits, except as used for rounding. See Programming Notes. Fraction multiplication proceeds as in fixed point full word multiplication, but produces only a 62-bit fraction product.) The sum of the characteristic less 64 is used as the characteristic of an intermediate product.

The sign of the product is determined by the rules of algebra.

The product fraction is normalized by post-normalizing the 62-bit intermediate product, if necessary, then truncating the product to 56 bits. The intermediate product characteristic is reduced by the number of left-shifts.

Exponent overflow occurs if the final product characteristic exceeds 127. The operation is terminated, and a program interruption occurs. The overflow exception does not occur for an intermediate product characteristic exceeding 127 when the final characteristic is brought within range because of normalization.

Exponent underflow occurs if the final product characteristic is less than zero. If the floating-point exponent underflow mask is a one, a program interruption occurs. If the mask bit is zero, the result is made a true zero.

When all digits of the intermediate product fraction are zero, the product sign and characteristic are made zero, yielding a true zero result. No interruption for exponent underflow or exponent overflow can occur when the result fraction is zero. The program interruption for lost significance is never taken for multiplication.

CONDITION CODE

The code remains unchanged.

PROGRAM INTERRUPTION

Exponent Overflow
Exponent Underflow (occurs prior to zero operand test)

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED OR REPRODUCED ANYWHERE FOR ANY PURPOSES

The multiplication of two floating-point numbers consists of a characteristic addition and a fraction multiplication. The sum of the characteristics less 64 is used as the characteristic of an intermediate product. The sign of the product is determined by the rules of algebra.

The product fraction is normalized by postnormalizing the intermediate product, if necessary. The intermediate product characteristic is reduced by the number of left-shifts. For short operands (six-digit fractions), the product fraction has the full 14 digits of the long format with the two low-order fraction digits accordingly always zero.

Exponent overflow occurs if the final product characteristic exceeds 127. The operation is terminated, and a program interruption occurs. The overflow exception does not occur for an intermediate product characteristic exceeding 127 when the final characteristic is brought within range because of normalization.

Exponent underflow occurs if the final product characteristic is less than zero. If the floating-point exponent underflow mask is a one, a program interrupt occurs. If the mask bit is zero, the result is made a true zero.

When all 14 digits of the intermediate product fraction are zero, the product sign and characteristic are made zero, yielding a true zero result. No interruption for exponent underflow or exponent overflow can occur when the result fraction is zero. The program interruption for lost significance is never taken for multiplication.

The least significant part of the product fraction replaces the contents of floating-point register R1 \oplus 001. Then, the most significant part of the intermediate product fraction replaces the contents of floating-point register R1.

CONDITION CODE

The code remains unchanged.

PROGRAM INTERRUPTIONS

Exponent Overflow
Exponent Underflow (occurs prior to zero operand tests)

PROGRAMMING NOTES

Interchanging the two operands in a floating-point multiplication does not affect the value of the product.

When either the multiplicand or multiplier is a true zero, the result is normally forced to a true zero without requiring the hardware to enter the longer multiply algorithm. When multiplying a true zero by another true zero or a true zero by any number with a characteristic less than 64 (hexadecimal 40), the exponent underflow interrupt will occur (if not masked) and the product is not computed. Masking the interrupt will generate a true zero product.

Notice that the MULTIPLY (short) instruction uses two registers for its result if R1 was even. This allows the programmer to use the additional precision without going to the extended form of the MULTIPLY.

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

1 COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

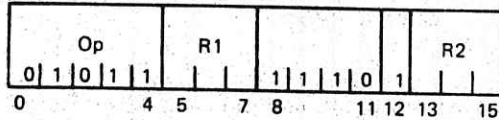
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE REPRODUCED
REPRODUCED IN ANY FORM OR BY ANY MEANS WITHOUT PERMISSION

MS 87-08

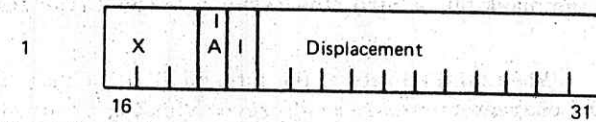
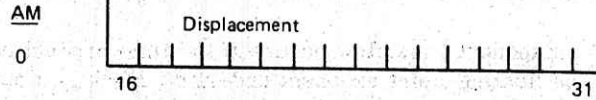
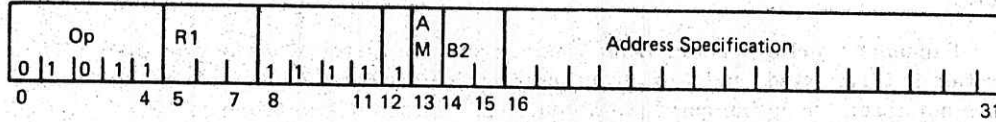
Box 1042

FF 46

SUBTRACT (LONG OPERANDS)



Mnemonic: SEDR
Format: R1, R2



| | AM | Mnemonic | Format |
|-----------|----|-----------------|-----------------|
| Extended: | 0 | SED | R1, D2 (B2) |
| Indexed: | 1 | SED [@] [#] | R1, D2 (X2, B2) |

DESCRIPTION

The long second operand is subtracted from the long first operand, and the normalized difference is placed in the first operand location.

The long 64-bit second operand is subtracted from the contents of floating-point register pair specified by the even register R1 and R1 ⊕ 001. The normalized result is placed into floating-point registers R1 and R1 ⊕ 001.

The SUBTRACT (long operand) is similar to ADD (long operand), except that the sign of the second operand is inverted before addition.

The sign of the difference is derived by the rules of algebra. The sign of a difference with zero result fraction is always positive.

RESULTING CONDITION CODE

- 00 Result fraction is zero
- 11 Result is less than zero
- 01 Result is greater than zero

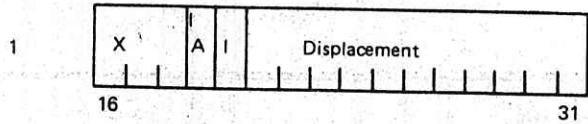
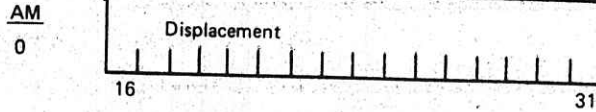
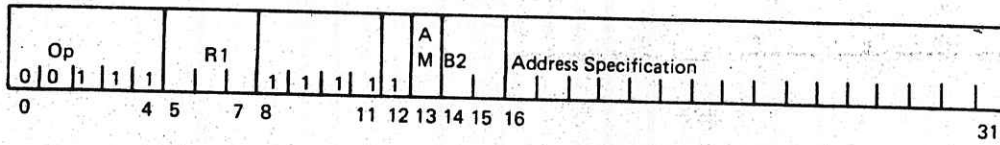
PROGRAM INTERRUPTIONS

- Significance
- Exponent Overflow
- Exponent Underflow

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NEAR EXACT OR ANY OTHER MANNER

MS 87-08
 Box 4042
 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

STORE (LONG OPERANDS)



| | AM | Mnemonic | Format |
|-----------|----|------------------|-----------------|
| Extended: | 0 | STED | R1, D2 (B2) |
| Indexed: | 1 | STED [@] [#] | R1, D2 (X2, B2) |

DESCRIPTION

The long first operand is stored at the long second operand location. The first operand is not changed.

The first operand is located in the even/odd pair of floating-point registers specified by the even register R1. First, bits 0 through 31 of floating-point register R1⁺ are stored into the second fullword of the doubleword storage area starting with the second operand fullword address. Bits 0 through 31 of floating-point register R1 are stored in the fullword specified by the second operand fullword address.

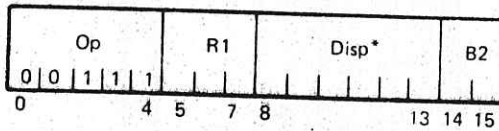
CONDITION CODE

The code remains unchanged.

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER FOR ANY PURPOSES
 REPRODUCTION OF THIS MATERIAL MAY BE PERMITTED BY THE
 WICHITA STATE UNIVERSITY LIBRARIES

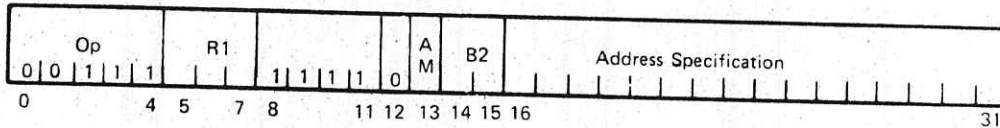
MS 87-08 BOX 1042 FF 46
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

STORE (SHORT OPERANDS)

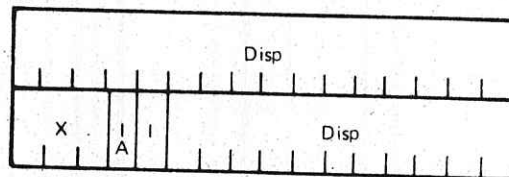


Displacements of the form 111XXX are not valid.

Mnemonic: STE
Format: R1,D2(B2)



Extended: AM 0 Mnemonic STE Format R1,D2(B2)
Indexed: 1 STE(@) [=] R1,D2(X2,B2)



DESCRIPTION

The contents of floating-point register R1 is stored at the second operand location. The contents of R1 is not changed. The overflow and carry indicators are not changed by this instruction.

RESULTING CONDITION CODE

The code is not changed.

8-31/8-32

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE PROTECTED BY COPYRIGHT LAW (TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES SPECIAL COLLECTIONS

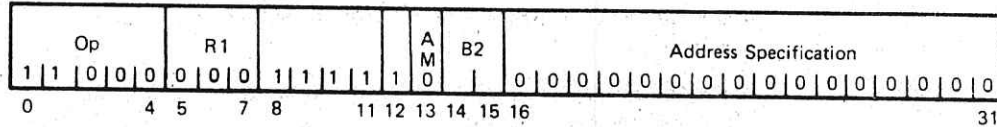
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR REPRODUCED IN ANY MANNER NOR BE LOANED, REPRODUCED, OR

MS 87-08 Box 104 FF 46
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

SPECIAL OPERATIONS

This section describes the special instructions. These instructions make possible the use of efficient pseudo subroutines, permit the specification of storage protection, perform status switching, and control I/O.

DETECT



DESCRIPTION

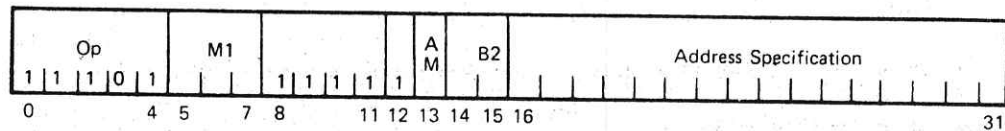
The B2 field uniquely selects one of four special microprogram routines. The selected micro-routine is executed. These routines are used to perform built-in diagnostic functions to verify the proper functioning of the CPU hardware.

Since the instruction is not intended for normal program usage, DETECT has no mnemonic. This is a privileged operation and can only be executed when the CPU is in the Supervisor state.

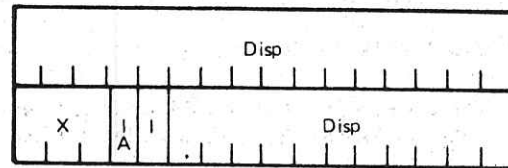
PROGRAM INTERRUPTION

Privileged operation

INSERT STORAGE PROTECT BITS



| | AM | Mnemonic | Format |
|-----------|----|------------------|--------------|
| Extended: | 0 | ISPB | M1,D2(B2) |
| Indexed: | 1 | ISPB [@] [≠] | M1,D2(X2,B2) |



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION. THIS MATERIAL MAY NOT BE REPRODUCED
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, WITHOUT
 PERMISSION FROM THE ARCHIVE DEPARTMENT.

MS 87-08 Box 42 1 FP 42
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

DESCRIPTION

Bits 5 through 7, the M1 field, are decoded to set or reset the protection bit associated with each halfword in main-storage as specified by the EA. The contents of the specified location, however, are not changed.

The following defines the combinations of the M1 field and the corresponding result:

| <u>M1 Field</u> | <u>Result</u> |
|-----------------|--|
| 000 | Reset the storage protection bit for the halfword second operand. |
| 001 | Reset the storage protection bits for both halfwords in the fullword second operand. |
| 010 | Set the storage protection bit for the halfword second operand. |
| 011 | Set the storage protection bits for both halfwords in the fullword second operand. |
| 100 | Illegal |
| 101 | Illegal |
| 110 | Illegal |
| 111 | Illegal |

This is a privileged operation and can only be executed when the CPU is in the supervisor state.

RESULTING CONDITION CODE

The code is not changed by this instruction.

INDICATORS

The carry and overflow indicators are not changed by this instruction.

PROGRAM INTERRUPTIONS

Illegal operation

PROGRAMMING NOTES

The low-order bit in the EA is used to specify the halfword when M1 is 000 or 010. When M1 is 001 or 011, the low-order bit of the EA should be 0 and will be ignored.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER FOR ANY PURPOSES EXCEPT AS MAY BE
PERMITTED BY THE ARCHIVES AND UNIVERSITY ARCHIVES

MS 87-08

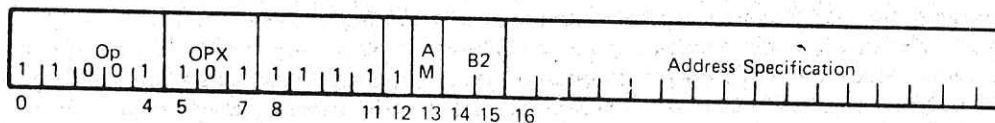
Box 42

FF 42

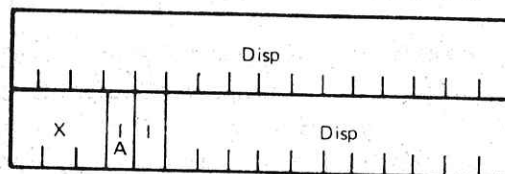
This instruction will always have halfword alignment and will be excluded from automatic index alignment.

The illegal M1 field patterns (100, 101, 110, and 111) leave the storage protect override bit set on which means that storage protected locations can be written into without getting a store protect violation. The condition will occur until the next valid ISPB is executed.

LOAD PROGRAM STATUS



| | AM | Mnemonic | Format |
|-----------|----|------------|-----------|
| Extended: | 0 | LPS | D2(B2) |
| Indexed: | 1 | LPS[@] [=] | D2(X2,B2) |



DESCRIPTION

Two fullwords starting at the location designated by the fullword operand address replace the contents of the program status registers on the CPU, as described under Program Status word. (Section 2, Figure 2-19).

RESULTING CONDITION CODE

The code is set or defined by the new PSW.

INDICATORS

The carry and overflow indicators are set or defined by the new PSW.

PROGRAMMING NOTE

This is a privileged operation and can only be executed when the CPU is in the supervisor state. This instruction will always have halfword index alignment and will be excluded from automatic index alignment.

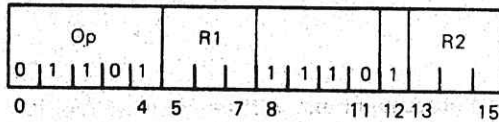
PSW bits 40 through 43 are not changed by the load operation.

PROGRAM INTERRUPT

If PSW bits 19 and 20 are both set, a fixed-point overflow will occur.

RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 REPRODUCED BY U.S. CODE
 TITLE 17, SECTION 105
 COPYRIGHT LAW
 STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WICHITA STATE UNIVERSITY
 WICHITA, KANSAS 67260-0001

MS 87-08
 Dr. James E. Tornayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives
 Box 42
 FF 42

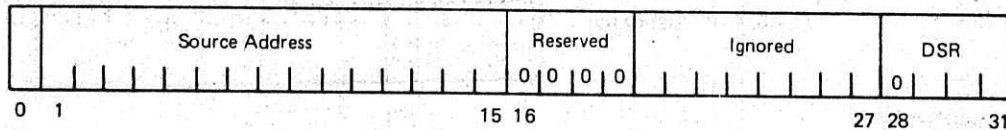
MOVE HALFWORD OPERANDS

| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| MVH | R1, R2 |

DESCRIPTION

Bits 0 through 15 of the general register specified by R1 contain the destination address. (This is analogous to the RR Format Branch Instructions except when bit 0 of general register R1 is a one; in that case the DSR in the current PSW is used.) Bits 16 through 31 of R1 contain a count of halfwords to be moved which must be greater than zero. Since its representation uses a signed 2's complement integer format, bit 16 (the sign bit) should be zero. A negative count (bit 16 equals 1) indicates no data will be moved.

The content of the general register specified by R2 is as follows:



When bit 0 in R2 is zero, the source address uses an implied DSR of all zeros.
When bit 0 in R2 is one, the source address uses the DSR contained in bits 28-31.

Data (a block of contiguous halfwords) is moved a halfword at a time from a source whose address is determined by concentrating the value of the DSR in R2 with the Source Address in R2 and adding to it the value of the count in bits 16 through 31 of R1 which is decremented by one for each halfword moved. The data is moved to the destination whose address is determined by adding to the operand address (Bits 0 through 15 of R1) the current value of the count. The move is completed when the count becomes zero. See Figure 9-1.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

The overflow and carry indicators are not changed.

PROGRAMMING NOTES

As in all instructions, main store addresses (for source and destination) must not be expected to cross 32K sector boundaries, because this instruction will not modify the DSR's. If this is ever attempted, the result is quite predictable in that operands will be used from the first 32K main storage locations.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT

(TITLE 17 U.S. CODE)
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
WITHOUT WRITTEN PERMISSION THE MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

MS 87-08 Box 42 FF 47
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

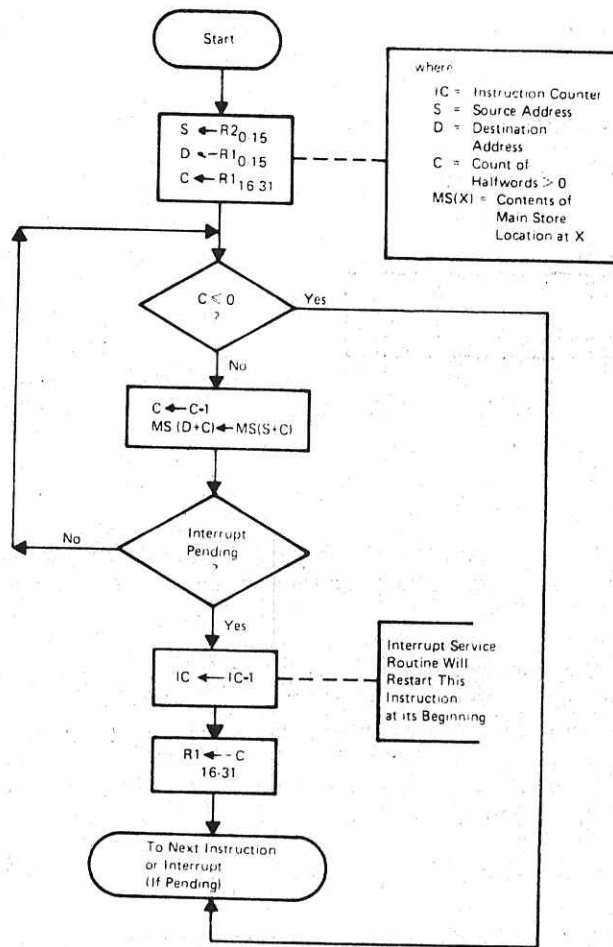


Figure 9-1. Move Halfword Execution

Because the MOVE HALFWORD instruction can execute for a long time, it has been designed to be interruptible. The following interrupts are typical of those interrupts which may break into the sequence of moves before the instruction is finished:

1. Initial power off signal (POI) from power supply.
2. Counter 1 or 2 interrupts.

When MOVE HALFWORD ends prematurely due to any of the above pending interrupts, the instruction counter will be decremented such that when the interrupt is taken the old PSW contains the instruction address of the move instruction. Also, when this instruction is interrupted, the count in R1 is modified to reflect the number of halfwords remaining to be moved. This will allow returning to the move instruction so that it can continue to be executed from where it was interrupted.

The programmer is encouraged to have both source and destination address low-order bits set the same. This will enable the instruction to accelerate execution by using fullword transfers for the majority of the move.

HARDWARE ANOMALY

External 1 interrupt "Old-PSW" can be invalid when any of the following interrupts occur:

1. I/O Interface Address Parity

DESCRIPTION

The halfword second operand replaces bits 32 to 47 of the PSW. This is a privileged operation and can only be executed when the CPU is in the supervisor state.

RESULTING CONDITION CODE

The code is not changed by this instruction.

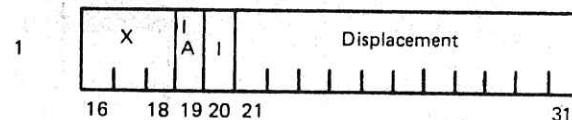
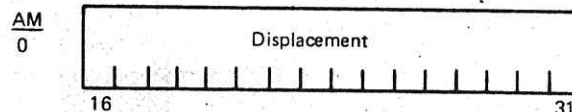
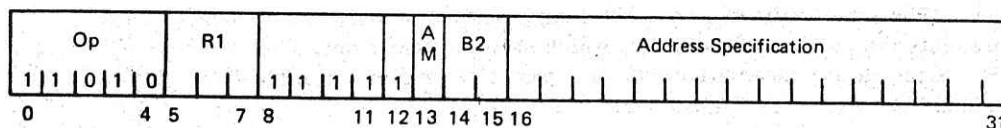
INDICATORS

The carry and overflow indicators are not changed by this instruction.

PROGRAMMING NOTE

Bits 5 through 7 are not used by this instruction. It is recommended that these bits be set to zero.

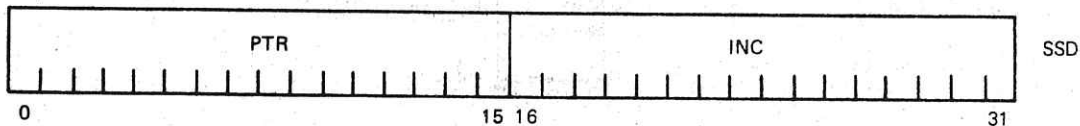
STACK CALL



| | AM | Mnemonic | Format |
|-----------|----|--------------|-----------------|
| Extended: | 0 | SCAL | R1, D2 (B2) |
| Indexed: | 1 | SCAL (@) [#] | R1, D2 (X2, B2) |

DESCRIPTION

This instruction for calling subroutines automatically controls saving bits 0 through 31 of the current PSW, the 8 general registers and programmer's temporary work space in main storage. When the Stack Call (SCAL) instruction is to be used, or the corresponding Stack Return (SRET), general register R1 must contain a Stack Status Descriptor word (SSD), as follows:



FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION FROM THE WICHITA STATE UNIVERSITY LIBRARY ARCHIVES

MS 87-08 Box 42 FF 47
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

First a branch address is computed. A save area address on the stack is computed from values in the SSD in R1 as:

$$SA = PTR + INC$$

(This save area address must be an even boundary halfword address.) Then the first two halfwords of the current PSW, and eight GPRs are automatically stored in the 18 halfwords beginning at location SA.

The SSD in R1 is now updated, as:

$$PTR = SA;$$

$$INC = 18.$$

Finally, the next instruction is taken from the branch address. This is essentially a BAL instruction which provides an automatic call stack function.

PROGRAMMING NOTE

PTR is a normal 16-bit address which is the location of a particular place in the stack. (The stack utilizes a variable-length portion of contiguous storage.) INC represents the number of halfwords which have currently been used in the stack beyond PTR. Since its representation uses a signed 2's complement integer format, its sign bit should be zero. See Figure 9-2.

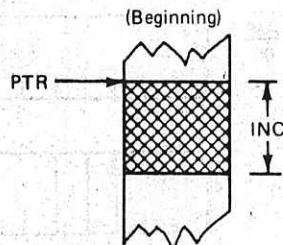


Figure 9-2. Current STACK Status - Prior to SCAL

When SCAL is executed, the new stack save address is calculated from $PTR + INC$, (SA), and then the current PSW and eight general registers are automatically saved in the new stack save area pointed to by SA, so that the stack now appears as in Figure 9-3. Then the PTR in R1 is updated to the value in SA and INC set at 18.

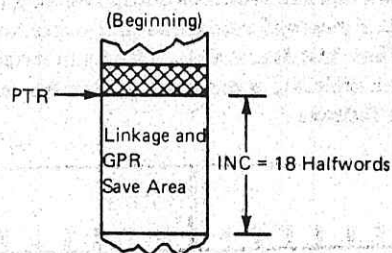


Figure 9-3. STACK Status - Upon Completion of SCAL

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
COPYRIGHT PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, OR PLACED IN ANY REPOSITORY.

MS 87-08

Box 42

FF 47

The programmer is free to use additional space in the stack, by simply using R1 as a base, and an offset which is greater than 18 (to avoid destroying the saved GPR contents). However, this additional information will be lost if he issues another SCAL without specifically adjusting INC in R1 to include this new space.

When SRET is executed, the first 2 halfwords of the PSW and the eight GPRs are automatically loaded from the save area at location PTR (in R1). Note that this restores R1 to contain the SSD it had just prior to the last SCAL, which means that the stack is automatically restored to the state of Figure 9-2.

Refer to STACK RETURN.

HARDWARE ANOMALY

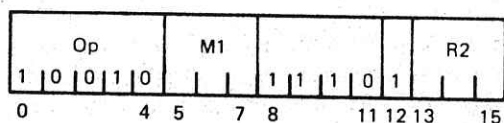
External 1 interrupt "Old PSW" can be invalid when any of the following interrupts occur:

1. I/O Interface Address Parity
2. DMA Data Parity
3. PCI Data Parity

PROGRAM INTERRUPTION

Specification
Protection

STACK RETURN



| Mnemonic | Format |
|----------|--------|
| SRET | M1, R2 |

DESCRIPTION

When SCAL is used to call a subroutine, the complementary branch instruction SRET is used to leave the calling subroutine and return to the conditions prior to the last SCAL. This is a conditional branch instruction in the RR format which provides the first two halfwords of the PSW and restores the registers (GPR's) to the same state as existed at the time of the SCAL.

The instruction execution first matches the M1 field against the condition code to determine if the branch should be taken. If the branch should not be taken, the instruction terminates at this point. The remaining description applies when the branch should occur.

The stack pointer address, PTR, is located in bits 0 through 15 of the general register specified by R2. (This address must be an even boundary halfword address.) The first two halfwords of the stack are moved into the active PSW. Next, all eight general purpose registers are loaded from the current stack save beginning at location PTR + 2 as specified in R2. Finally, instruction execution continues from the address indicated by the active PSW.

CONDITION CODE

The value in the corresponding field is loaded from the stack.

INDICATORS

The value in the corresponding field is loaded from the stack.

PROGRAMMING NOTES

The following notes are intended to amplify and clarify the use of the stack and extended call facility.

- Since the stack is located in main store, any area of the stack can be accessed by standard addressing techniques (i. e. , using R1 as a base).
- While the primary purpose of the stack is automatic register saving and restoring, it also provides automatic allocation and de-allocation of temporary work space, a function often required for efficient use of storage, and for use of reentrant programs. Note that the INC value in the SSD does not have to be modified to use this work space; simply addressing relative to the base in R1 allows this. The INC value only needs to be adjusted if the information in the stack space needs to be preserved during a subsequent SCAL.
- The total stack space (i. e. , the space taken up by the total stack at any given time) is variable. It grows and shrinks as a function of the depth of the call tree and the amount of workspace used by the various programs. However, in the overall data structure of the total application, there must inevitably be a fixed limit on the amount of main store which can be allocated to the stack. Such limit would presumably be based on either statistics of usage plus a safety factor, or else on a detailed analysis of the usage of all possible call chains. In both cases (the latter as an error detection mechanism) it is important to have some mechanism to stop the call chain if through some peculiar circumstances the stack should exceed its allocated space. Unfortunately, there does not appear to be any fool-proof scheme. However, most such situations would be caught by appending a few words at the end of the allocated space which have the store protect bit on. Any attempt to store into the stack beyond its limit would result in a protection violation and interrupt.
- Since the PSW and the eight general purpose registers are automatically restored on SRET, it is not possible to return results directly to the calling program in the registers. Rather, the value to be returned in a register must be stored into the appropriate slot in the general purpose register save area in the stack. Then, when the registers are restored, the calling program will, in fact, find the value in the register. At the same time, additional values can be returned to the calling program in the work space in the stack, since the calling program can access that space by addressing relative to the base in R1 (SCAL). (There must, of course, be an agreed-upon convention as to the specific locations in the work space.) Note — the floating-point registers are not affected by SCAL and SRET so variables can be passed in these registers.

FOR RESEARCH USE ONLY

THIS MATERIAL IS

PROTECTED BY COPYRIGHT LAW

(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

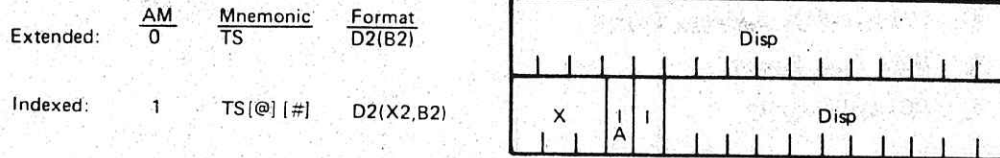
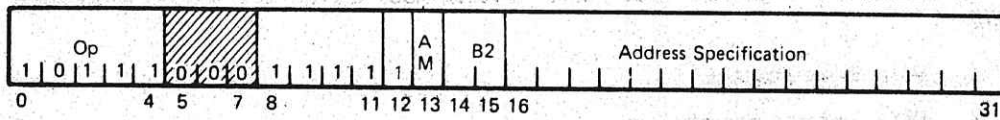
MS 87-08

Box 42

FF 47

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

TEST AND SET



DESCRIPTION

Bits in the halfword second operand are tested to set the condition code, and the second operand is set to all ones. No other access to this location is permitted between the fetch and the storing of all ones.

RESULTING CONDITION CODE

- 00 The bits are all zeros
- 11 The bits are mixed with zeros and ones
- 01 The bits are all ones.

INDICATORS

The carry and overflow indicators are not changed by this instruction.

PROGRAMMING NOTES

TS can be used for the controlling and sharing of a common storage area by more than one program. To accomplish this, a halfword can be designated as control. The desired interlock can be achieved by establishing a program convention in which a zero halfword indicates that the common area is available, but a one means that the area is being used. Each using program then must examine this halfword by means of a Test and Set before making access to the common area. If the test sets the condition code to 00, the area is available for use; if it sets the condition code either 01 or to 11, the area cannot be used. Because Test and Set permits no access to the test halfword between the moment of fetching (for testing) and the moment of storing all ones (setting),

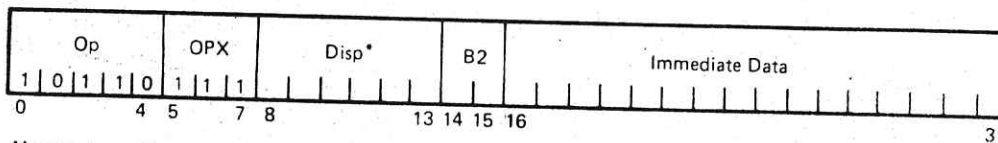
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 COPIED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 REPRODUCED FROM NASA ARCHIVES REPOSITORY

MS 87-08 Box 42 FF 47
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

the possibility is eliminated of a second program testing the halfword before the first program is able to reset it. Selective bits can be tested by using the TEST AND SET BITS instruction.

Bits 5 through 7 are not used by this instruction. It is recommended that these bits be set to zero.

TEST AND SET BITS



Mnemonic: TSB
 Format: D2(B2),Data
 *Displacements of the form 111XXX are invalid.

DESCRIPTION

Bits 16 through 31 of this instruction are treated as halfword immediate data. The immediate data is logically tested with the halfword second operand. The logical sum (OR) of the immediate data and the halfword second operand is formed bit-by-bit. The result replaces the halfword second operand. No other access to this location is permitted between the fetching of the operand and the storing of the result.

RESULTING CONDITION CODE

- 00 Either the bits selected by the immediate data are zeros or the immediate data is all zeros
- 11 The bits selected by the immediate data are mixed with zeros and ones
- 01 The bits selected by the immediate data are all ones.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAMMING NOTE

The one bits in the halfword mask specify the bits of the halfword second operand that are set one. The result replaces the halfword second operand. The following table defines this instruction.

| TEST AND SET BITS | |
|-------------------|---------|
| Mask | 1 1 0 0 |
| Storage | 1 0 1 0 |
| Result | 1 1 1 0 |

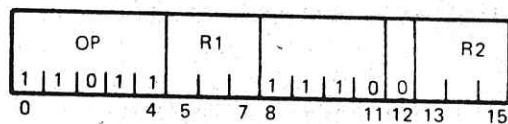
FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPIES PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WICHITA STATE UNIVERSITY LIBRARY OR
 WICHITA STATE UNIVERSITY ARCHIVES PERMISSION

MS, 87-08 Box 42, FF 47
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

INTERNAL CONTROL OPERATIONS

A CPU instruction will initiate an Internal Control operation that will perform the following functions, depending on the control word (CW) coding:

- A fullword will be transferred between general register R1 and counter 1 or 2. The high halfword of general register R1 (the most significant halfword) is transferred to or from the main store halfword location 00B0 for counter 1 or 00B1 for counter 2. The low halfword of general register R1 (the least significant halfword) is transferred to or from a 16-bit hardware binary counter 1 or counter 2. Section 2 contains a description of counter operations.
- An AGE command word, specified by bits 16 through 31 of the CW (R2), will be transferred to the AGE interface, and a halfword will be transferred to or from bits 0 through 15 of a general register (R1) and the AGE interface.
- Four discretes will be transferred from bits 0 through 3 of a general register (R1) to the I/O interface.
 - 0 - XMIT Disable
 - 1 - BCE Disable
 - 2 - Spare 1
 - 3 - Spare 2
- I/O channel reset. The channel reset operation issues a reset to the IO. The IO and CPU uses the signal to reset the IO/CPU interface logic. If an external interrupt 0 has occurred, this command must not be executed until IOP level A interrupt register has been read.

INTERNAL CONTROL

| | |
|-----------------|---------------|
| <u>Mnemonic</u> | <u>Format</u> |
| ICR | R1,R2 |

DESCRIPTION

This instruction transfers a fullword to or from the general register specified by R1. Operations are further defined by a control word contained in bits 0 through 31 of the general register specified by R2. The CW format is shown below.

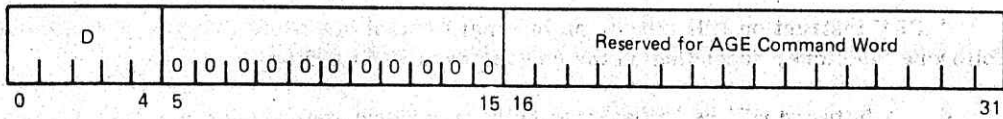
FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

MS, 87-08
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives
 Box 421
 FF 471

CONTROL WORD (CW)



| Legal D Command | Meaning |
|-----------------|-----------------|
| 00000 | Read Counter 1 |
| 00001 | Read Counter 2 |
| 00101 | Read AGE |
| 01000 | Write Counter 1 |
| 01001 | Write Counter 2 |
| 01100 | Write Discretes |
| 01101 | Write AGE |
| 10000 | Channel Reset |

No data transfer is associated with the Channel Reset operation.

RESULTING CONDITION CODE

The code is not changed.

INDICATORS

The overflow and carry indicators are not changed by this instruction.

PROGRAM INTERRUPTIONS

Illegal operation

PROGRAMMING NOTES

This is a privileged operation and can only be executed when the CPU is in the supervisor state.

The illegal operation program interruption will occur if the following illegal commands are used: 00010, 00011, 00100, 00110, 00111, 01010, 01011, 01110, and 01111.

Commands of the form LXXXX other than 10000 are reserved and should not be used. The illegal operation program interruption does not occur; instead a channel reset is performed.

When using either Counter 1 of Counter 2 as a counter (rather than as an incremental timer), a possibility exists that the counter could be in error during a single read by 65.536 microseconds (low order bit of location 00B0 or 00B1). This problem can be avoided by doing two consecutive reads and making comparisons to pick the correct reading.

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
REPRODUCED BY ANYONE
WITHOUT PERMISSION FROM
THE NATIONAL ARCHIVES

REPRODUCED BY ANYONE
WITHOUT PERMISSION FROM
THE NATIONAL ARCHIVES

(TITLE 17 U. S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

THE UNIVERSITY OF MICHIGAN LIBRARIES

REPRODUCED BY ANYONE WITHOUT PERMISSION FROM THE NATIONAL ARCHIVES

MS 87-08

Box 42

FF 47

Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

EFFECTIVE ADDRESS GENERATION SUMMARY CHART

| SRS, SI Formats | RS Format | | | | | |
|-----------------|----------------------------|---------------------------|--------------|--------------|-----------------------------------|--------------------------------|
| | Extended Addressing (AM=0) | Indexed Addressing (AM=1) | | | | |
| | | IA | I | X=000 | X=000 | |
| B2≠11 | EA=(B)+DISP | EA=(B)+DISP | PEA=(B)+DISP | | | |
| | | | 00 | | EA=IC+PEA | EA=(X) ₀₋₁₅ +PEA |
| | | | 01 | | EA=IC-PEA | EA=(X) ₀₋₁₅ *PEA |
| | | | 10 | | EA=MS(PEA) | EA=MS(PEA)+(X) ₀₋₁₅ |
| | | 11 | | EA=MS(PEA)** | EA=MS(PEA)***+(X) ₀₋₁₅ | |
| B2=11 | EA=(B)+DISP | EA=DISP | PEA=DISP | | | |
| | | | 00 | | EA=IC+PEA | EA=(X) ₀₋₁₅ +PEA |
| | | | 01 | | EA=IC-PEA | EA=(X) ₀₋₁₅ *PEA |
| | | | 10 | | EA=MS(PEA) | EA=MS(PEA)+(X) ₀₋₁₅ |
| | | 11 | | EA=MS(PEA)** | EA=MS(PEA)***+(X) ₀₋₁₅ | |

Definitions

- EA Effective address, main storage address of second operand
- PEA Preliminary effective address
- (RN) Contents of bits 0-15 of general register specified by B2 or X
- RN General register "N", where N = 0 to 7
- (B) Contents of bits 0-15 of general register specified by the B2 field
- B2 B field of SRS, SI, or RS format instruction
- MS() Contents of the main storage location specified by the contents of the parenthesis
- DISP Displacement field of instruction
- X X field of RS format instruction with indexed mode of addressing
- (X)₀₋₁₅ Most significant halfword (bits 0-15) of the content of index register X automatically aligned.
- AM AM (addressing mode) field of RS format instruction
- IA IA (indirect address) field of RS format instruction with the indexed mode of addressing
- I I field of RS format instruction with indexed mode of addressing
- IC Updated Instruction Counter
- * Automatic Index Modification
- ** Automatic Storage Modification
- *** Direct Storage Addressing with/without Post Indexing

| X | INDEX VALUE | X | INDEX VALUE |
|-----|-------------|-----|-------------|
| 000 | Zero | 100 | (R4) |
| 001 | (R1) | 101 | (R5) |
| 010 | (R2) | 110 | (R6) |
| 011 | (R3) | 111 | (R7) |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 REPRODUCED BY ANYONE
 WITHOUT WRITTEN PERMISSION
 FROM THE NATIONAL AERONAUTICS
 AND SPACE ADMINISTRATION
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 COPY PROVIDED BY
 (TITLE 17 U.S. CODE)
 PROTECTED BY COPYRIGHT LAW

MS 187-08 Box 42 FF42
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives

AP-101 C/M INSTRUCTION REPERTOIRE

| <u>Name</u> | <u>Mnemonics</u> | <u>Format</u> |
|--------------------------------------|------------------|---------------|
| <u>Fixed-Point Operations</u> | | |
| Add | AR, A | RR, SRS, RS |
| Add Halfword | AH | SRS, RS |
| Add Halfword Immediate | AHI | RI |
| Add to Storage | AST | RS |
| Compare | CR, C | RR, SRS, RS |
| Compare Between Limits | CBL | RR |
| Compare Halfword | CH | SRS, RS |
| Compare Halfword Immediate | CHI | RI |
| Compare Immediate with Storage | CIST | SI |
| Divide | DR, D | RR, SRS, RS |
| Exchange Upper and Lower Halfwords | XUL | RR |
| Insert Address Low | IAL | SRS, RS |
| Insert Halfword Low | IHL | RS |
| Load | LR, L | RR, SRS, RS |
| Load Address | LA | SRS, RS |
| Load Arithmetic Complement | LCR | RR |
| Load Fixed Immediate | LFXI | RR |
| Load Halfword | LH | SRS, RS |
| Load Multiple | LM | RS |
| Modify Storage Halfword | MSTH | SI |
| Multiply | MR, M | RR, SRS, RS |
| Multiply Halfword | MH | SRS, RS |
| Multiply Halfword Immediate | MHI | RI |
| Multiply Integer Halfword | MIH | RS |
| Store | ST | SRS, RS |
| Store Halfword | STH | SRS, RS |
| Store Multiple | STM | RS |
| Subtract | SR, S | RR, SRS, RS |
| Subtract from Storage | SST | RS |
| Subtract Halfword | SH | SRS, RS |
| Tally Down | TD | SRS, RS |
| <u>Branch Operations</u> | | |
| Branch and Link | BALR, BAL | RR, RS |
| Branch and Index | BIX | RS |
| Branch on Condition | BCR, BC | RR, RS |
| Branch on Condition Backward | BCB | SRS |
| Branch on Condition (Extended) | BCRE | RR |
| Branch on Condition Forward | BCF | SRS |
| Branch on Count | BCTR, BCT | RR, RS |
| Branch on Count Backward | BCTB | SRS |
| Branch on Overflow and Carry | BVCR, BVC | RR, RS |
| Branch on Overflow and Carry Forward | BVCF | SRS |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
MICHIGAN STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION FROM THE COPYRIGHT OWNER

MS 187-08

Box 42

1 FH 471

| <u>Name</u> | <u>Mnemonics</u> | <u>Formats</u> |
|-------------------------------------|------------------|----------------|
| <u>Shift Operations</u> | | |
| Normalize and Count | NCT | RR |
| Shift Left Logical | SLL | SRS |
| Shift Left Double Logical | SLDL | SRS |
| Shift Right Arithmetic | SRA | SRS |
| Shift Right Double Arithmetic | SRDA | SRS |
| Shift Right Logical | SRL | SRS |
| Shift Right Double Logical | SRDL | SRS |
| Shift Right and Rotate | SRR | SRS |
| Shift Right Double and Rotate | SRDR | SRS |
| <u>Logical Operations</u> | | |
| AND | NR, N | RR, SRS, RS |
| AND Halfword Immediate | NHI | RI |
| AND Immediate with Storage | NIST | SI |
| AND to Storage | NST | RS |
| Exclusive-OR | XR, X | RR, SRS, RS |
| Exclusive-OR Halfword Immediate | XHI | RI |
| Exclusive-OR Immediate with Storage | XIST | SI |
| Exclusive-OR to Storage | XST | RS |
| OR | OR, O | RR, SRS, RS |
| OR Halfword Immediate | OHI | RI |
| OR to Storage | OST | RS |
| Search Under Mask | SUM | RR |
| Set Bits | SB | SI |
| Set Halfword | SHW | SRS, RS |
| Test Bits | TB | SI |
| Test Register Bits | TRB | RI |
| Test Halfword | TH | SRS, RS |
| Zero Bits | ZB | SI |
| Zero Register Bits | ZRB | RI |
| Zero Halfword | ZH | SRS, RS |

| <u>Name</u> | <u>Mnemonics</u> | <u>Formats</u> |
|--|------------------|----------------|
| <u>Floating-Point Operations</u> | | |
| Add (Long Operand) | AEDR, AED | RR, RS |
| Add (Short Operands) | AER, AE | RR, SRS, RS |
| Compare (Short Operand) | CER, CE | RR, RS |
| Convert to Fixed-Point | CVFX | RR |
| Convert to Floating-Point | CVFL | RR |
| Divide (Extended Operand) | DEDR, DED | RR, RS |
| Divide (Short Operand) | DER, DE | RR, SRS, RS |
| Load (Long Operand) | LED | RS |
| Load (Short Operand) | LER, LE | RR, SRS, RS |
| Load Complement (Short Operand) | LECR | RR |
| Load Fixed Register | LFXR | RR |
| Load Floating Immediate | LFLI | RR |
| Load Floating Register | LFLR | RR |
| Mid Value Select (Short Operands) | MVS | RS |
| Multiply (Extended Operand) | MEDR, MED | RR, RS |
| Multiply (Short Operand) | MER, ME | RR, SRS, RS |
| Subtract (Long Operand) | SEDR, SED | RR, RS |
| Subtract (Short Operand) | SER, SE | RR, SRS, RS |
| Store (Long Operand) | STED | RS |
| Store (Short Operand) | STE | SRS, RS |
| <u>Special Operations</u> | | |
| Detect ^P | - | RS |
| Insert Storage Protect Bits ^P | ISPB | RS |
| Load Program Status ^P | LPS | RS |
| Move Halfword Operands | MVH | RR |
| Set Program Mask | SPM | RR |
| Set System Mask ^P | SSM | RS |
| Stack Call | SCAL | RS |
| Stack Return | SRET | RR |
| Supervisor Call | SVC | RS |
| Test and Set | TS | RS |
| Test and Set Bits | TSB | SI |
| <u>Internal Control Operations</u> | | |
| Internal Control ^P | ICR | RR |
| <u>I/O Operations</u> | | |
| Program Controlled Input/Output ^P | PC | RR |

P: Privileged Instruction

12-3/12-4

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)
WICHITA STATE UNIVERSITY LIBRARIES
SPECIAL COLLECTIONS
REPRODUCED FROM THE ORIGINAL COPY NOT BE COPIED OR
REPRODUCED WITHOUT PERMISSION FROM WICHITA STATE UNIVERSITY LIBRARIES

MS 187-08
Box 42
FF42
Dr. James E. Tomayko Collection of NASA Documents
Wichita State University Libraries, Special Collections and University Archives

AP-101 C/M OP CODE ASSIGNMENTS

| OP 2,3 | OP0, OP1 | | | |
|--|--|--|--|---|
| | 00 | 01 | 11 | 10 |
| OP 04 = 1 | | | | |
| 00 | SRS SUBTRACT RR SUBTRACT RR ₂ COMP BTWN LMTS RS SUBTRACT RS ₂ SUB FRM STO | SRS DIVIDE RR DIVIDE RR ₂ COMP FL ST RS DIVIDE RS ₂ COMP FL ST | SRS BROV & CRY RR BROV & CRY RR ₂ SET PROG MSK RS BROV & CRY RS ₂ LM, STP, LPS, SM, SVC | SBS SUB HW RR LOAD FL IMM RR ₂ LOAD FL IMM RS SUB HW RS ₂ SET SYST MASK |
| 01 | SRS LOAD RR LOAD RS LOAD | SRS SUBTRACT FL ST RR SUBTRACT FL ST RR ₂ SUBTRACT FL LN RS SUBTRACT FL ST RS ₂ SUBTRACT FL LN | SRS BR RELATIVE RR ICR 1/0 RR ₂ PC RS BLX RS ₂ | SRS LOAD HW RR ₂ SUM RS LOAD HW RS ₂ MIH |
| 11 | SRS STORE FL ST RR CONV TO FXD RS STORE FL ST | SRS LOAD FL ST RR LOAD FL ST RR ₂ LOAD COMP FL ST RS LOAD FL ST RS ₂ LOAD FL LN | SRS REG SH DBL SRS COMP SH DBL | SRS STO HW RR LOAD FX IM RR ₂ LOAD FX IMM RS STO HW RS ₂ TST & SET |
| 10 | SRS OR RR OR RR ₂ LOAD FLTG REG RS OR RS ₂ OR TO STORE | SRS DIVIDE FL ST RR DIVIDE FL ST RR MOVE HALFWORD OPS RS DIVIDE FL ST TEST 2 (LRS) | SRS LOAD ADDRESS RR ₂ LOAD ARITH COMP RS LOAD ADDRESS RS ₂ INSTR PROT BITS | SRS MULTIPLY HW TEST 3 (RR ₂) RS MULTIPLY |
| OP 04 = 0 | | | | |
| 00 | SRS ADD RR ADD RR ₂ XU&L HW RS ADD RS ₂ ADD TO STORE | SRS MULTIPLY RR MULTIPLY RS MULTIPLY | SRS BR ON COND RR BR ON COND RR ₂ BR ON COND EXT RS BR ON COND RS ₂ DETECT | SRS ADD HW RR ₂ RS ADD HW |
| 01 | SRS COMP DVD FL LN RR COMP RR ₂ RS COMP RS ₂ DVD FL LN | SRS ADD FL ST RR ADD FL ST RR ₂ ADD FL LN RS ADD FL ST RS ₂ ADD FL LN | SRS BR ON CT RR BR ON CT TEST 4 (RR) RS BR ON CT RS ₂ STACK CALL | SRS COMP HW RR ₂ STACK RTRN RS COMP HW |
| 11 | SRS STORE RR ₂ MPY FL LN RS STORE RS ₂ MPY FL LN | SRS XOR RR XOR RS XOR RS XOR TO STORE | SRS RG SH SING SRS COMP SH SING | IEXP RR, RS R1 = OPX |
| 10 | SRS AND RR AND RR ₂ LOAD FX RG RS AND RS ₂ AND TO STORE | SRS MULTIPLY FL ST RR MULTIPLY FL ST TEST 1 (RR) RS MULTIPLY FL ST RS ₂ MID VALUE SLCT | SRS INSRT ADD LO RR BR & LNK RR ₂ NORM & CNT RS ₂ INSRT ADD LO | IMPL SRS, RS R1 = OPX TEST 3 (LRS) |
| <p>Notes: OP12 = 1 Causes either RR₂ or RS₂ Operations FL ST - Floating-Point (Short Operands) FL LN - Floating-Point (Long Operands) HW - Halfwords</p> <p>Op Code 00011 with OP12 = 1 is reserved</p> | | | | |

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 MICHIGAN STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION FROM THE ORIGINAL OWNER BE COPIED OR
 REPRODUCED IN ANY MANNER OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.
 MS 87-08 1 Box 42 1 FF 47
 Dr. James E. Tomayko Collection of NASA Documents
 Michigan State University Libraries, Special Collections and University Archives

AP-101 C/M OP Code Assignments (cont)

| <u>OP</u> | <u>R1=OPX</u> | <u>RR₂</u> | <u>RS₂</u> |
|-----------|---------------|-----------------------|-----------------------|
| 11001 | 000 | Set Program Mask | Store Multiple |
| 11001 | 001 | Reserved | Supervisor Call |
| 11001 | 010 | Reserved | Reserved |
| 11001 | 011 | Reserved | Reserved |
| 11001 | 100 | Reserved | Load Multiple |
| 11001 | 101 | Reserved | Load Program Status |
| 11001 | 110 | Reserved | Reserved |
| 11001 | 111 | Reserved | Reserved |

IMPLIED IMMEDIATE

| | | | |
|-------|-----|---------------|---------|
| 10100 | 000 | Tally Down | SRS, RS |
| 10100 | 001 | Zero Halfword | SRS, RS |
| 10100 | 010 | Set Halfword | SRS, RS |
| 10100 | 011 | Test Halfword | SRS, RS |
| 10100 | 100 | Reserved | |
| 10100 | 101 | Reserved | |
| 10100 | 110 | Reserved | |
| 10100 | 111 | Reserved | |

EXPLICIT IMMEDIATE

| <u>OP</u> | <u>R1=OPX</u> | <u>RR</u> | <u>RR₂</u> | <u>SRS</u> |
|-----------|---------------|---------------------|-----------------------|------------------------------|
| 10110 | 000 | Add Half Immediate | Reserved | Modify Storage Halfword |
| 10110 | 001 | Zero Register Bits | Reserved | Zero Bits |
| 10110 | 010 | OR Half Immediate | Reserved | Set Bits |
| 10110 | 011 | Test Register Bits | Reserved | Test Bits |
| 10110 | 100 | XOR Half Immediate | Reserved | XOR Immediate With Store |
| 10110 | 101 | Comp Half Immediate | Reserved | Compare Immediate With Store |
| 10110 | 110 | AND Half Immediate | Reserved | AND Immediate With Store |
| 10110 | 111 | Mult Half Immediate | Reserved | Test and Set Bits |

FOR RESEARCH USE ONLY
THIS MATERIAL MAY BE
PROTECTED BY COPYRIGHT LAW
(TITLE 17 U.S. CODE)

COPY PROVIDED BY
MICHIGAN STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION FROM THE NATIONAL ARCHIVES
REPRODUCED FROM THE NATIONAL ARCHIVES

WITHOUT WRITTEN PERMISSION FROM THE NATIONAL ARCHIVES
REPRODUCED FROM THE NATIONAL ARCHIVES

MS 87-08

Box 42

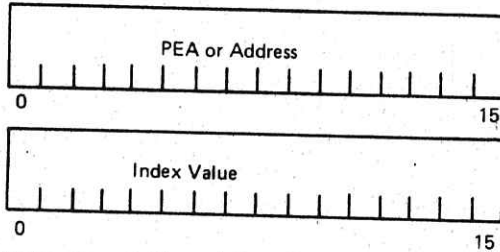
FF 47

Section 14

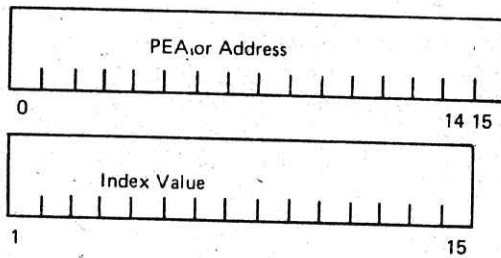
AUTOMATIC INDEX ALIGNMENT DESCRIPTION

Index alignment occurs automatically. That is, bits 0 through 15 of the general register specified by X specify entities. The identity of this entity is explicitly defined by the particular operation being executed.

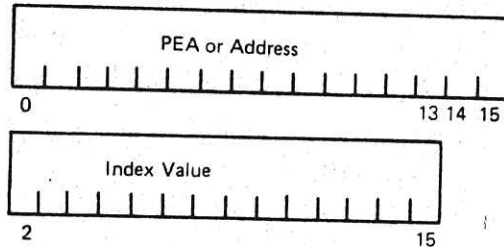
Halfword operations align index value bit 15 with the least-significant bit of the PEA (described in Section 2) or the ADDRESS portion of an indirect address pointer. (described in Section 2). It should be noted that the LOAD MULTIPLE, STORE MULTIPLE, LOAD PROGRAM STATUS, and INSERT STORAGE PROTECT BITS instructions are excluded from automatic index alignment and have a halfword index alignment.



Likewise, fullword operations functionally shift the index value one position to the left, prior to alignment. Note that bit 0 of the index value is lost.



Likewise, doubleword operations functionally shift the index value two positions to the left prior to alignment. Note that bits 0 and 1 of the index value are lost.



14-1/14-2

NASA-JSC

FOR RESEARCH USE ONLY
 THIS MATERIAL MAY BE
 PROTECTED BY COPYRIGHT LAW
 (TITLE 17 U.S. CODE)
 COPY PROVIDED BY
 WICHITA STATE UNIVERSITY LIBRARIES
 SPECIAL COLLECTIONS
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR
 REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION

MS1 87-08 Box 42
 Dr. James E. Tomayko Collection of NASA Documents
 Wichita State University Libraries, Special Collections and University Archives
 FF 47