AP-101S
PATCH FUNDAMENTALS
OTP155

CHUCK WOHL

OCT. 2004

Class Overview

- **Class Name: AP101s Patch Fundamentals**
- **Class Number: OTP155**
- **Prerequisites:**
  - Basic AP101s Assembler knowledge is helpful.
  - Good general knowledge of the FSW.
- **Intended Audience:**
  - All FSW, but specifically those that develop FSW Patches.

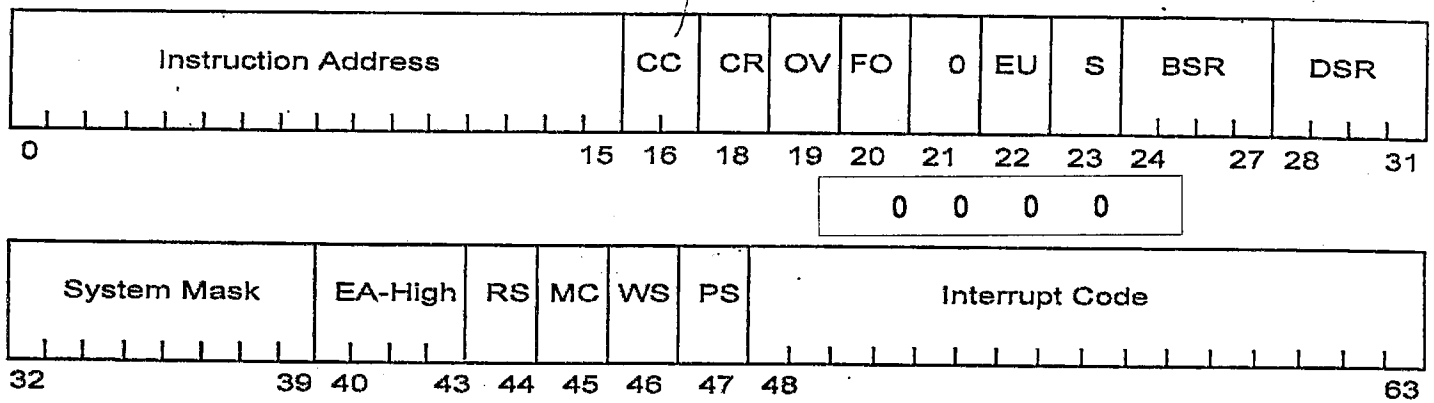
CLASS CONTENTS

- AP-101S ABCs
  - Program Status Word (PSW) Basics
    - PASS Applications
  - GPC Memory Access
    - Address Expansion (using DSR / DSE / BSR)
  - General Register Use
- INSTRUCTION ABCs
  - Basic Instruction Formats and CPU Operation
  - Instruction Format Exceptions
- Basic Addressing Modes (Adequate for most patches)
- BRANCHING - Testing Condition Codes (CC)
- SNEAKY INSTRUCTIONS

CLASS CONTents

- RESTRICTIONS / PRECAUTIONS
- WHY PATCH?
  - Facility Authorizations / Considerations
- WHAT / HOW / WHERE
  - In-Place
  - Patch Area
- KIS? Principle
  - Caution / Tips on Patch Installation
- SAMPLE PATCH DEVELOPMENT

AP-101S: PROGRAM STATUS WORDS (PSWs)



Current PSW located at Loc 0010 upon FEID LSTOP & GPC Power OFF in STBY or RUN

Nominal 3rd HW of PSW
FCOS = A00C
Non-FCOS = FC05
Inside PROT Block = A005

0-15	Next Instruction Address
16-17	Condition Code
18	Carry Indicator
19	Overflow Indicator
20	Fixed-Point Arithmetic Overflow Mask *
21	Reserved
22	Floating Point Exponent Underflow Mask *
23	Significance Mask *
24-27	Branch Sector Register
28-31	Data Sector Register
32	Counter 1 Mask *
33	Counter 2 Mask *
34	Instruction Monitor Mask *
35	Ext. 0 Interrupt Mask (Level A) *
36	Ext. 1 Interrupt Mask (Level B) *
37	Ext. 2 Interrupt Mask (Level C) *
38	Ext. 3 Interrupt Mask (Unused) *
39	Ext. 4 Interrupt mask (Unused) *
40-43	SVC parameter list high-order EA Bits
44	Register Set **
45	Machine Check Mask *
46	Wait State Bit ***
47	Problem/Supervisor State Control Bit ****
48-63	Interrupt Code for Program, Machine Check and Special Ext. Interrupts or 16-bit SVC parameter list address for SVC instructions

* 0 = Inhibited
1 = Enabled

** 0 = Application Non-FCOS
1 = FCOS

*** 0 = Processing State
1 = Wait State

**** 0 = Supervisor State → Execute "Privileged Instructions"
1 = Problem State (HAL/S types)

AP-101S MEMORY; SECTOR LOCATIONS

- ADDRESSED BY HALFWORD LOCATIONS: 00000 - 7FFFF
- HALFWORD = 16 BITS (2 BYTES); FULLWORD = 32 BITS
- GPC MEMORY SECTORS = 8000 HEX (32768 DEC.) HALFWORDS LONG

Sector #'s

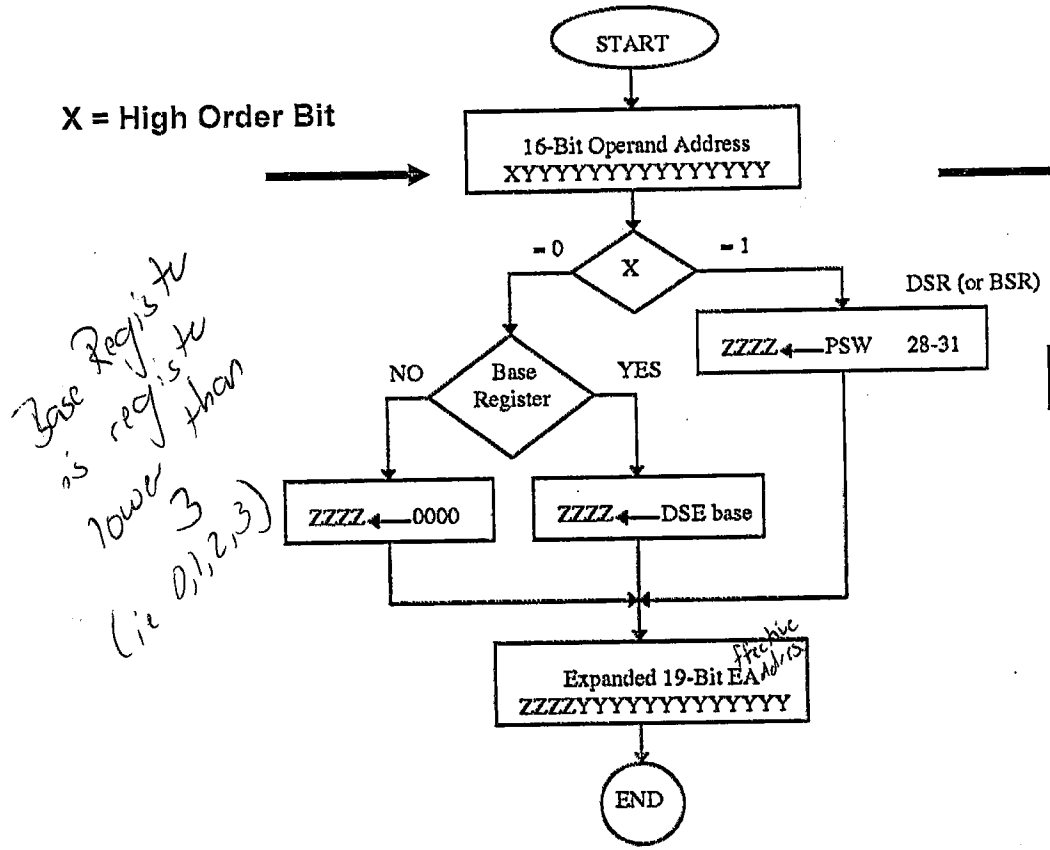
BSR/DSR/DSE	MEMORY LOC.	BSR/DSR/DSE	MEMORY LOC.
0	00000 - 07FFF	8	40000 - 47FFF
1	08000 - 0FFFF	9	48000 - 4FFFF
2	10000 - 17FFF	A	50000 - 57FFF
3	18000 - 1FFFF	B	58000 - 5FFFF
4	20000 - 27FFF	C	60000 - 67FFF
5	28000 - 2FFFF	D	68000 - 6FFFF
6	30000 - 37FFF	E	70000 - 77FFF
7	38000 - 3FFFF	F	78000 - 7FFFF

NOTE: DSEs for all Regs in both Reg sets are cleared by microcode during GPC microcode initialization when the GPC is moded from HALT to STBY. DSEs must be loaded via the LDM or LXAR / LXA instructions.

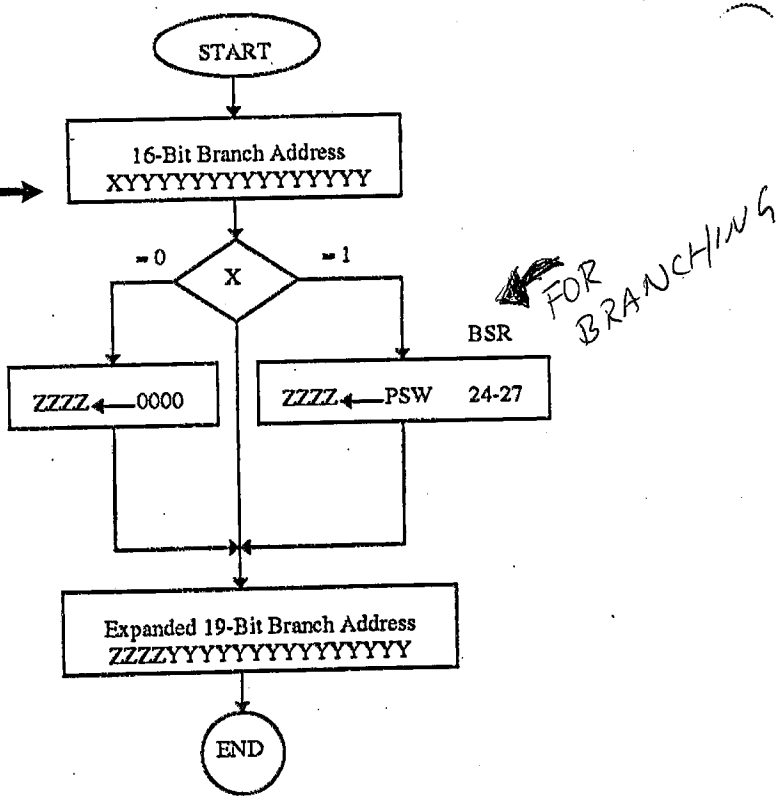
MEMORY ADDRESSING Using DSRs / DSEs / BSRs

16 bit Operand Address = 9CCA; DSR = 1; ADDR = 09CCA
16 bit Operand Address = 1CCA; DSE = 7; ADDR = 39CCA

X = High Order Bit



Data Operand Addressing Expansion

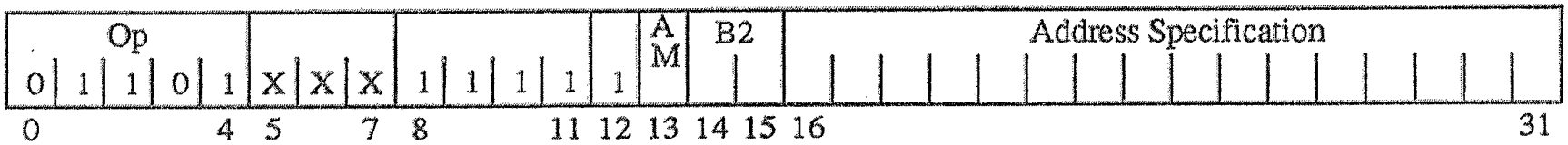


Branch Addressing Expansion

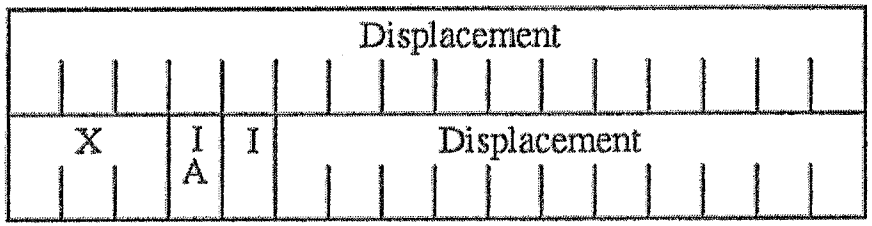
NOTE: DSEs can also be used with other REGs in MVH/SCAL/SRET Instructions

DSE LOAD For Regs 0 - 3 (DATA REMOTE)

9.13 LOAD DSE MULTIPLE



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

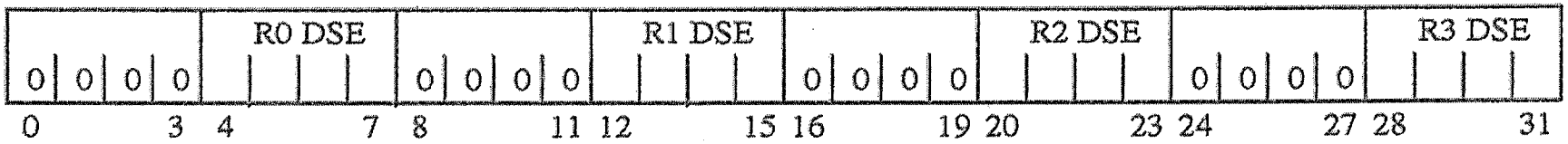


DESCRIPTION:

The four Data Sector Extensions (DSE) corresponding to R0-R3 of the current register set are initialized from the fullword second operand.

The format of the fullword second operand is:

C.G. $ZDSESET and $ZDSECLR



PROCEDURE GRR_RCS (DSR/DSE Values ??)

```
0079E6-0079EB    #EGEIORB          ****    0006(    6)      PDE (Program Directory Entry)
0079E6-0079EB    #EGEIORB+0000     0000    0000     DDB2    0051    79EC    8003
```

GEI: BSR = 5; DSR =1
First 8 bits of 2nd HW Zero'd by FPMSCHED

```
02DEE1-02DEE2    $0GEIORB+012F     D0FF    3AC0     021E76  SCAL@# R0,X'02C0'(R1,R3)
```

stack call ≈ branch instruction

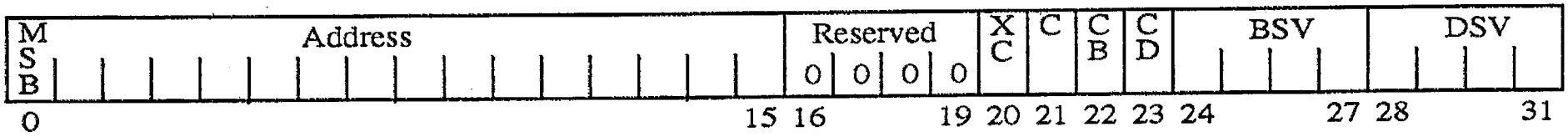
```
0002C0-0002C1    #ZGRRRCS          ****    0002(    2)      ZCON
0002C0-0002C1    #ZGRRRCS+0000     9E76    0E40
```

GRR: FW Ind Addr Ptr. -Use BSV ; Not DSV
See Next Page - (Ref. P.O.O. Pg 2-13 &14)

```
021E76-02238B    #CGRRRCS          ****    0516(    1302)   PROCEDURE       GRR_RCS
021E76           #CGRRRCS+0000     ST#6    EQU      *
021E76-021E77    #CGRRRCS+0000     E9F3    8BA4     LA ≃ LHI  R1,X'8BA4'    (See Next Page-#D)
021E78-021E79    #CGRRRCS+0002     B6E1    7FFF     NHI       R1,X'7FFF'    (Clear HOB)
021E7A           #CGRRRCS+0004     B914             STH       R1,5(R0)      (Save R1 in Stack)
021E7B-021E7C    #CGRRRCS+0005     E0FB    001C     IAL       R0,X'001C'
021E7D           #CGRRRCS+0007     EB91             LA        R3,X'0024'(R1)
021E7E-021E7F    #CGRRRCS+0008     68FB    04F6     LDM       X'04F6'       ($ZDSESET)

0004F4-0004F5    $ZDSECLR          ****    0002
0004F4-0004F5    $ZDSECLR+0000     0000    0000
0004F6-0004F7    $ZDSESET          ****    0002
0004F6-0004F7    $ZDSESET+0000     0007    0007     R0 & R2 DSEs = 0; R1 & R3 DSEs = 7
```

ADDR MODES: ZCON FORMAT & EXAMPLE



Field	Function
X_C	Index Control
C	Control to allow PSW modification
C_B	Control BSV Usage
C_D	Control DSV Usage
BSV (Branch Sector Vector)	Selectively replaces BSR in PSW
DSV (Data Sector Vector)	Selectively replaces DSR in PSW
MSB (Most Significant Bit)	Determines type of address expansion

ZCON Example from previous chart = 9E76 0E40

BRANCH	XC	C	CB	CD	FUNCTION
YES	1	1	1	0	BSR = BSV and BA = BSR [ADDR] Ref. Pg. 63

#DGRRRCS - Declared as "DATA_ REMOTE"

```
38BA4-038BF1   #DGRRRCS          ****     004E(    78)      DATA
38BA4-038BC5   #DGRRRCS+0000     ???      ADCONS,LITERALS,ETC.    ???
38BA4-038BB3   #DGRRRCS+0000     5AB0     3D1E     5B84     3A58     CF00     576E
38BB4-038BC3   #DGRRRCS+0010     0000     000F     FFFF     FFF0     FFFF     FFFF
38BC4-038BC5   #DGRRRCS+0020     0000     0001
38BC8-038BC9   #DGRRRCS+0024     ---      LOCAL    BLOCK    DATA     (BLOCK   GRR_RCS_RM_1
38BC8-038BC9   #DGRRRCS+0024     0000     0012
38BCA  #DGRRRCS+0026   CGRS_OX_TEMP_LM          0FA0
38BCB  #DGRRRCS+0027   CGRS_FU_TEMP_LM          0AA0
38BCC  #DGRRRCS+0028   CGRS_FVRCS_FU_LEAK_TEMP_LT_ORB      *4100
38BCD  #DGRRRCS+0029   CGRS_FVRCS_OX_LEAK_TEMP_LT_ORB      *4100
38BCE  #DGRRRCS+002A   CGRS_LVRCS_FU_LEAK_TEMP_LT_ORB      *4100
38BCF  #DGRRRCS+002B   CGRS_LVRCS_OX_LEAK_TEMP_LT_ORB      *4100
38BD0  #DGRRRCS+002C   CGRS_RVRCS_FU_LEAK_TEMP_LT_ORB      *4100
```

R1 usually always contains pointer to #D data area

DSR / DSE INITIALIZATION - PGM ARA

```
000A42-000A47    #EARAGPC          ****    0006(    6)     PDE
000A42-000A47    #EARAGPC+0000     0000    0000     8BC0   0087    1308    8001
```

ARA: BSR = 8 ; DSR = 7
FPMSCHED Clears 1st 8 Bits of 2nd HW

```
040BC0           $0ARAGPC+0000     ST#8    EQU      *      BLOCK
040BC0-040BC1    $0ARAGPC+0000     E9F3    81B4     LA     R1,X'81B4'
        #DARAGPC
040BC2-040BC3    $0ARAGPC+0002     B6E1    7FFF     NHI    R1,X'7FFF'
040BC4           $0ARAGPC+0004     B914             STH    R1,5(R0)  SAVE REG 1 in Stack
040BC5-040BC6    $0ARAGPC+0005     E0FB    0018     IAL    R0,X'0018'
040BC7           $0ARAGPC+0007     EB51             LA     R3,X'0014'(R1)
        #DARAGPC+14      (LOCAL    BLOCK
040BC8-040BC9    $0ARAGPC+0008     68FB    04D2     LDM    X'04D2'   $ZDSESET+0000
040BCA           $0ARAGPC+000A     BB24             STH    R3,9(R0)  SAVE REG 3 in Stack
```

```
0004D0-0004D1    $ZDSECLR          ****    0002
0004D0-0004D1    $ZDSECLR+0000     0000    0000
0004D2-0004D3    $ZDSESET          ****    0002
0004D2-0004D3    $ZDSESET+0000     0007    0007
```

R0 & R2 DSEs = 0; R1 & R3 DSEs = 7

REGISTER USE in Instruction Formats

RULES For Registers - Operands, Bases, Indices

General Register Number	Register Function			
	Operand	Base Reg		Index
		SRS	RS	
0	000	00	00	NULL
1	001	01	01	001
2	010	10	10	010
3	011	11	NULL	011
4	100	--	--	100
5	101	--	--	101
6	110	--	--	110
7	111	--	--	111

NULL - NOT TREATED as Base Reg or IndeX

REG CONTENTS - Addresses / Data ??

Per P.O.O Par. 2.2.1: "General register contents can be used interchangeably as operands for arithmetic, logical, and shifting operations, or as base and index registers for relative addressing."

```
040C02   $0ARAGPC+0000   ST#8     EQU    *          BLOCK
040C02   $0ARAGPC+0000   E9F3     81BC   LA≈LHI     R1,X'81BC'          #DARAGPC
040C04   $0ARAGPC+0002   B6E1     7FFF   NHI        R1,X'7FFF'          32767
040C06   $0ARAGPC+0004   B914            STH        R1,5(R0)  SAVE REG  1
040C07   $0ARAGPC+0005   E0FB     0018   IAL        R0,X'0018'
040C09   $0ARAGPC+0007   EB61            LA         R3,X'0018'(R1)      #DARAGPC+24
040C0A   $0ARAGPC+0008   68FB     04F6   LDM        X'04F6'  $ZDSESET+0000
```

```
040C11   $0ARAGPC+000F   ST#234   EQU    *          ASSIGN  003820
040C11   $0ARAGPC+000F   9A05     LH     R2,1(R1)   LITERAL:H'10370',=X'2882'
***      COMPOOL         BASEREG:       =Y(#PCZ2COM+0478)
040C12   $0ARAGPC+0010   9D02     LH     R5,0(R2)   CZ2B_DIA_RM1
040C13   $0ARAGPC+0011   BDA1     STH    R5,X'0028'(R1)   ARAB_RM1
```

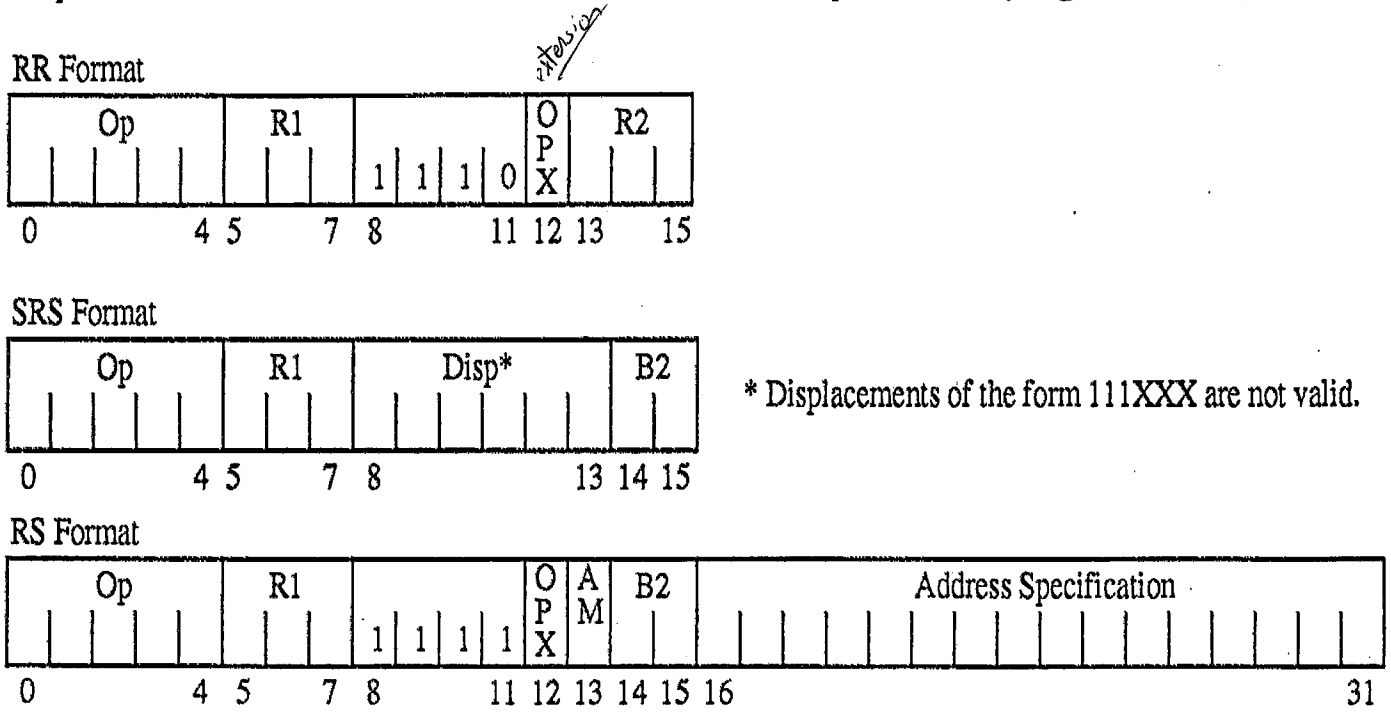
```
0381BC-0381D1   #DARAGPC+0000   ???    ADCONS,LITERALS,ETC.   ???
0381BC-0381CB   #DARAGPC+0000   23F2   2882   28E0   1FD4   23F6   9C72
```

ABCs: BASIC INSTRUCTION FORMATS

OP : This 5-bit field defines an 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 : In the RR format, R2 is used to specify a reg containing either the 2nd operand or the address of the 2nd operand (e.g. BCTR).



* Displacements of the form 111XXX are not valid.

B2 in SRS = R0-R3

B2 in RS = R0 - R2
See Pg 21-22

Figure 2-4. Basic Instruction Formats

BASIC INSTRUCTION FORMATS

B2 : 2-Bit field specifies the register containing the Base Address, e.g. #D data area; beginning of GPC's GST or ICC Buffer, etc.

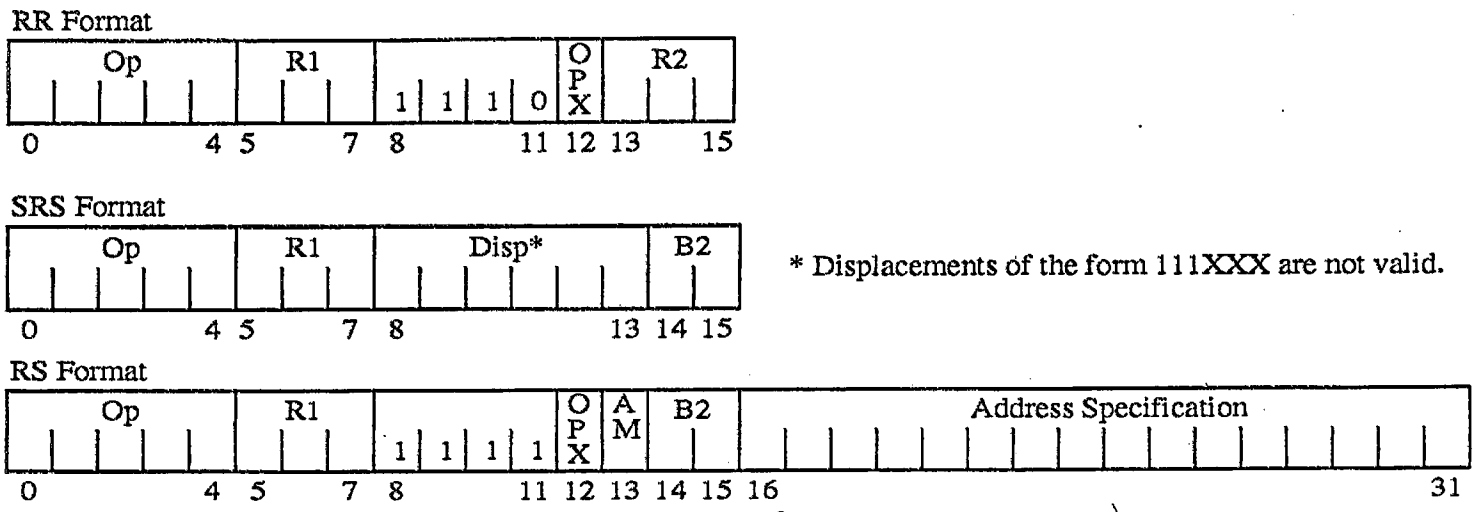
Disp : In SRS format, B2 + Disp* provides memory address; e.g. loc. of operand.

(BEWARE !!! B2 + Disp* alignment based on HW or FW operation-See pg 18 &19)

OPX : Extension of OP field (May be multiple bits, e.g. replace R1 or R2 field)

AM : Designates Extended vs Indexed Addressing

Addr Spec : If AM = 0, 16 bit displacement; if AM = 1, indexed addressing modes



* Displacements of the form 111XXX are not valid.

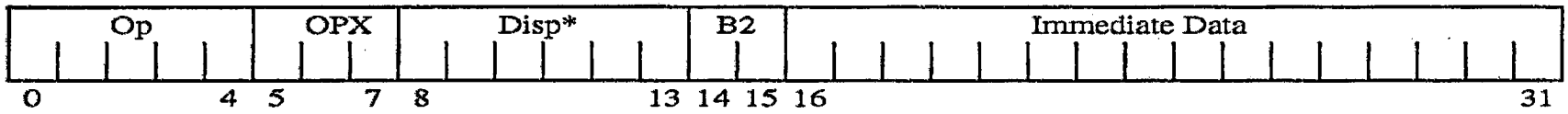
Figure 2-4. Basic Instruction Formats

SI / RI FORMATS

2.2.6 SI INSTRUCTIONS "Storage Immediate"

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.

EXAMPLE: TEST BITS (TB)



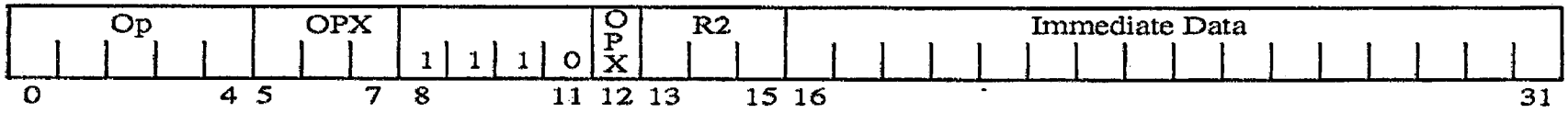
* Displacements of the form 111XXX are not valid.

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.

2.2.7 RI INSTRUCTIONS "Register Immediate"

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.



EXAMPLE: TRB MHI

Figure 2-10. RI Instructions

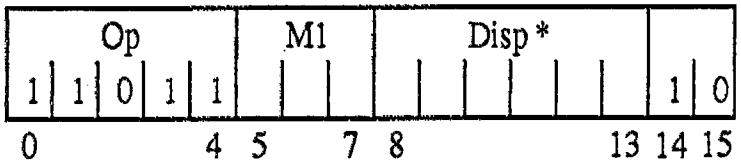
Except for test instructions, the result of the operation between the second operand and the immediate data replaces the second operand. The second operand is not altered for test instructions. The immediate data first operand is never altered for RI instructions.

INSTRUCTION FORMAT FIELD EXCEPTIONS

- M 1 Field :Replaces R1 Field (Bits 5-7) in "most" Branch Instructions (SRS or RS Formats) (More Detail later when we cover BRANCHING)
- COUNT: Replaces the DISP field in SRS format SHIFT instructions, e.g. SLL, SRL, SRA, etc.

Plus other BC Instructions

5.4 BRANCH ON CONDITION BACKWARD

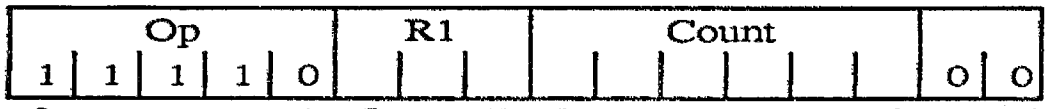


* Displacements of the form 111XXX are not valid.

Mnemonic	Format
BCB	M1,D2

Bits 14 - 15 Vary with Instruction; Refer to P.O.O.

6.2 SHIFT LEFT LOGICAL



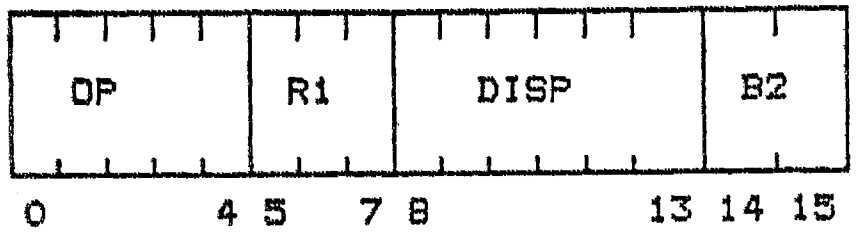
Mnemonic	Format
SLL	R1,Count

ADDRESSING MODES

Addressing Mode	Inst. Type	Address Loc. (16 Bit prior to Expansion)
Direct Addressing	SRS	Operand is located at Base + Disp
B2 = 3	RS	ADDR SPEC = Effective Address / Immediate Data
Extended Addressing	RS	Operand is located at Base + 16 Bit Disp
Indexed Addressing	RS	Operand is located at : Base + Disp + Index
Indexed Addressing with Index Mod.	RS	Same as above, and then: Index = Index + Constant
IC Relative Addressing	RS	Operand is located at: Base + Disp + IC
Indirect Addressing	RS	Address of Operand is at: Base + Disp
Indirect Addressing with Storage Mod.	RS	Same as above, and then: Address = Address + Constant
Indirect Addressing with Post Indexing	RS	Same as Indirect addressing except: Address = Address + Constant
ZCON Addressing	RS	Indirect Addressing with option to modify DSR or BSR

ADDR MODES: SRS DIRECT

SRS Format Addressing



Address of operand is determined from :

1) Contents of base register
2) 6 bit displacement field
3) Size of operand (half,full)

B2 = Regs 0, 1, 2, or 3 (for SRS)

Address of operand is computed as follows :

STEP 1 :

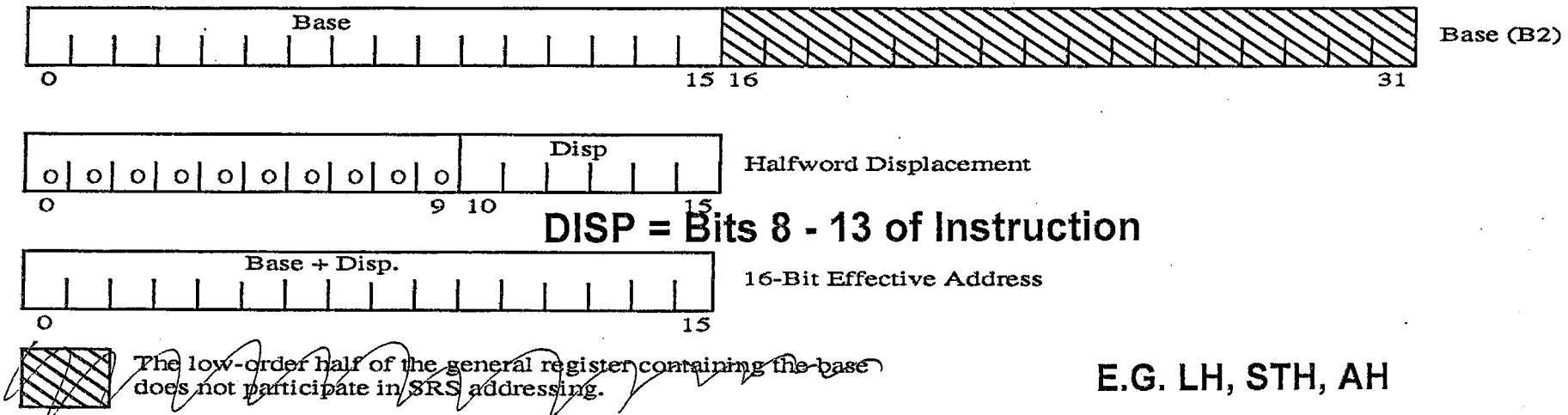
the contents of the base register (bits 0 - 15)
\+ 6 bit displacement aligned according to operand size

16 bit effective address

STEP 2 :

16 bit effective address is expanded to a 19 bit address using the DSR or DSE.

SRS OPERAND ADDRESSING

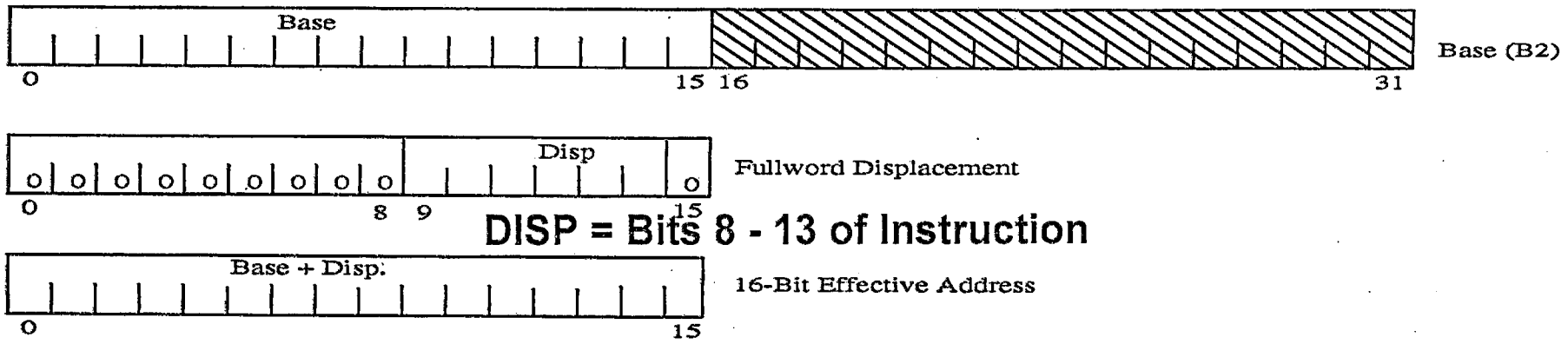


DISP = Bits 8 - 13 of Instruction

E.G. LH, STH, AH

ignor cross out

Figure 2-7. SRS Halfword Addressing



DISP = Bits 8 - 13 of Instruction

SRS Fullword Addressing, e.g. L, ST, A

SRS OPERAND EXAMPLES

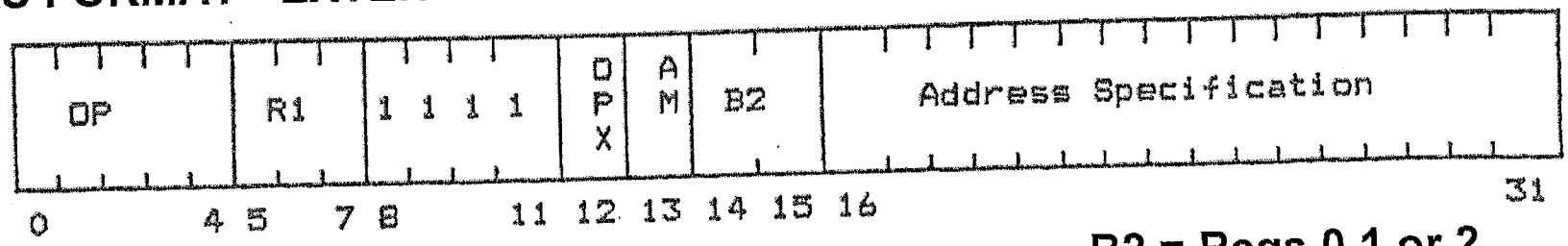
ASSEMBLER INSTRUCTIONS	OBJECT CODE:
L R4 , 8 (R3)	1C13
LH R4, 4 (R3)	9C13

OP	R1	DISP	B2
0 0 0 1 1	1 0 0	0 0 0 1 0 0	1 1
1 0 0 1 1	1 0 0	0 0 0 1 0 0	1 1

Reg 3 = 2020 0000 R3 DSE= 7
DISP = 4
16 Bit OPERAND EFFECTIVE ADDRESS:
Load Operand Addr = 2028
Load HW Operand Addr = 2024
19 Bit Operand Expanded Addresses use DSE = 7
Reg 4 Bits 0-31 loaded with FW contents of 3A028
Reg 4 Bits 0-15 loaded with HW contents of 3A024; Bits 16-31 = 0

ADDR MODES: RS EXTENDED ADDRESSING

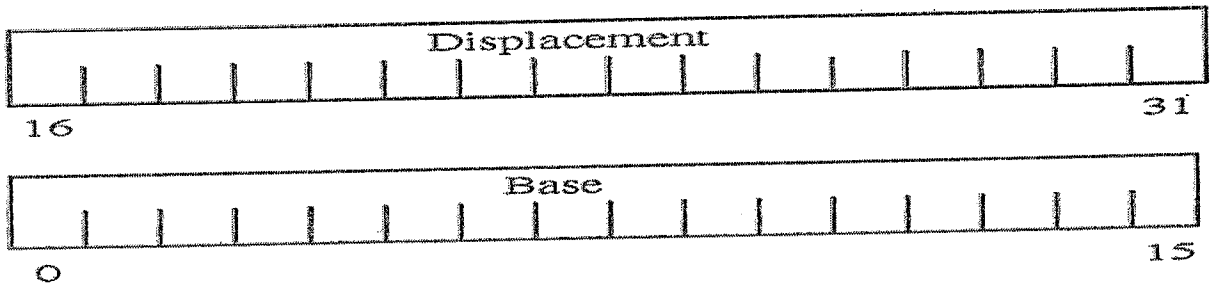
RS FORMAT - EXTENDED ADDRESSING



B2 = Regs 0,1,or 2

Where Address Specification is :
When AM bit = 0
(Extended Addressing)

Extended Addressing is performed when the AM Bit = 0 (and B2 NQ 3). This mode provides a full 16 bit Displacement to be added to the Base. Unlike the SRS, the Displacement is always aligned right justified with the Base regardless of Operand size (HW, FW, DW).

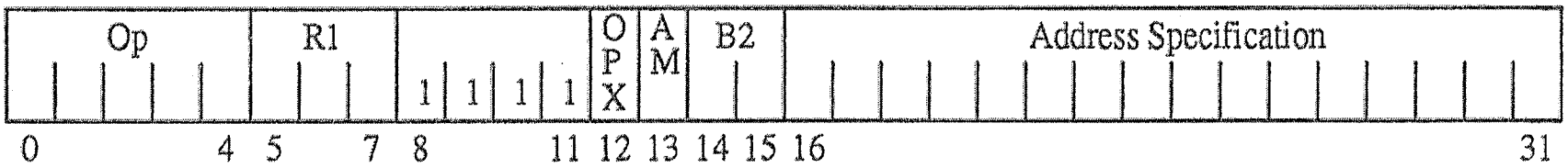


Reg 0, 1, or 2

ADDR MODES: RS FORMAT B2 = 3

RS FORMAT with B2 Field = "11" (3)

Base + Disp Addressing is NOT PERFORMED



IF B2 = "11": The Address Specification field is used directly as the Effective Address or as Immediate Data

EXAMPLE 1:

L R5, X'26F6' 1DF3 26F6 (B2 = 3 & AM = 0)

Loc. 26F6 = 1234 C6C6

Reg 5 loaded with contents of Loc 26F6 (DSE NOT APPLICABLE- WHY?)

b/c B2 is not really a base register

R5 = 1234 C6C6

EXAMPLE 2:

LA R5,X'26F6" EDF3 26F6 (B2 = 3 & AM = 0)

R5 = 26F6 0000 (Addr. Spec. Field loaded into bits 0-15 of R1)

Executes as a Load Halfword Immediate - LHI (ADDR or DATA)

INTRO to BRANCH ON PSW CC BITS via M1

	M1 FIELD (TEST)		
	(5)	(6)	(7)
ARITHMETIC & TALLY			
ZERO	1	0	0
NEGATIVE	0	1	0
POSTIVE (>0)	0	0	1
LOGICAL			
ZERO	1	0	0
NOT ZERO	0	1	0
TEST			
ZERO	1	0	0
MIXED	0	1	0
ALL ONES	0	0	1
COMPARE			
EQUAL	1	0	0
01 < 02	0	1	0
01 > 02	0	0	1

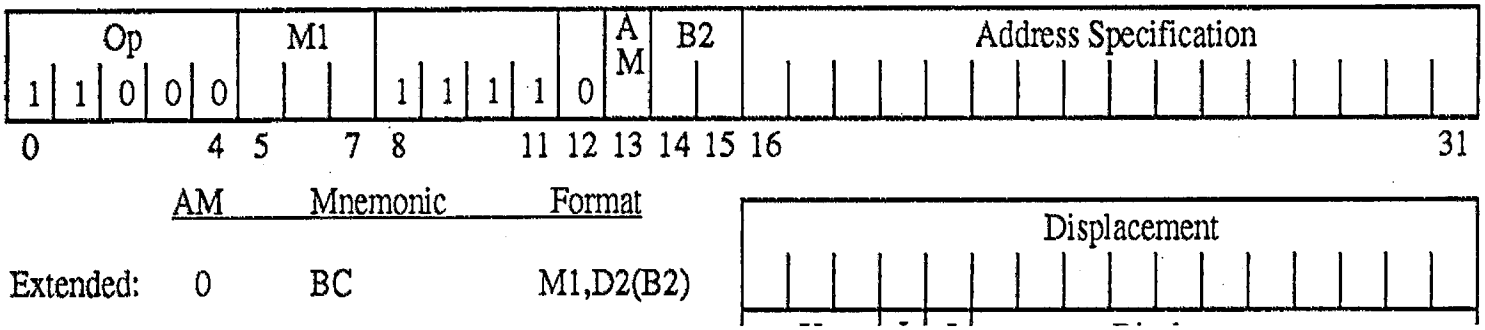
M1 FIELD = 111 (7)
ALWAYS BRANCH.

(Check P.O.O. for other M1 Field use, e.g. ISPB Instruction.)

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 TO 00, INSTRUCTION BIT 6 TESTS FOR A CODE EQUAL TO 11, AND INSTRUCTION BIT 7 TESTS FOR A CODE EQUAL TO 01.

BRANCHING: SIMPLE and EASY (BC)

- BRANCH ON CONDITION: EXTENDED ADDRESSING (MOST SIMPLE / LEAST RISKY method to Branch anywhere within the current sector of execution)



Conditional or Unconditional Branch to Patch Space or return from Patch; e.g. AM =0; B2 = 3; and 16 Bit Displacement = Start of allocated Patch Space or Location for return. High order bit of Displacement = 1 ensures use of BSR of current PSW and branches within current sector.

FOR EXAMPLE: ASSUME BSR = 6; Code execution at Loc. 3720C

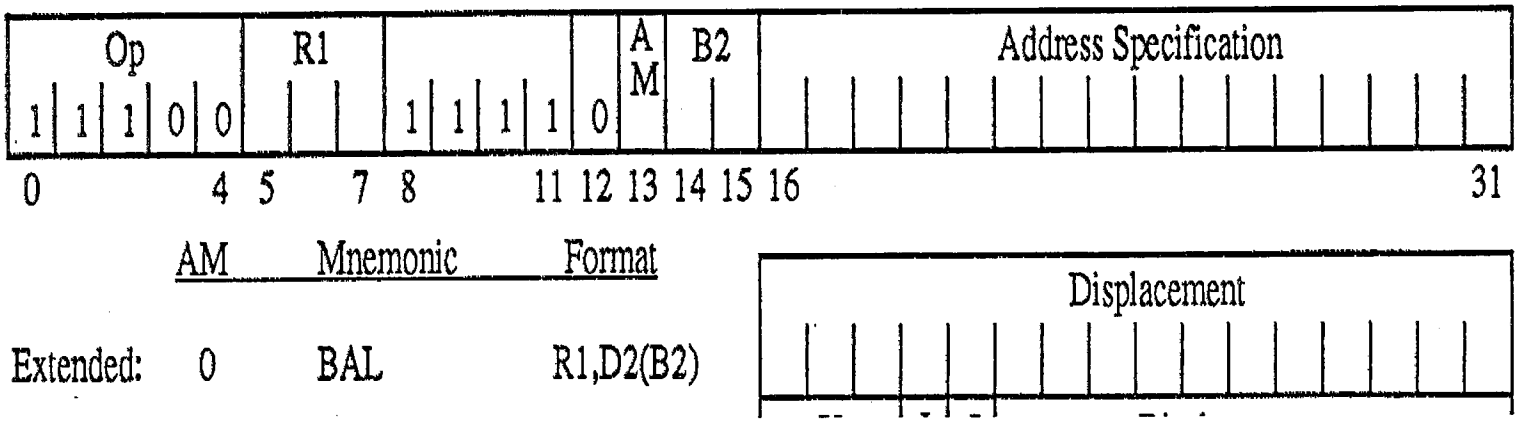
3720C C7F3 F800 Unconditional Branch to Loc. 37800

3720E - - - - - - - -

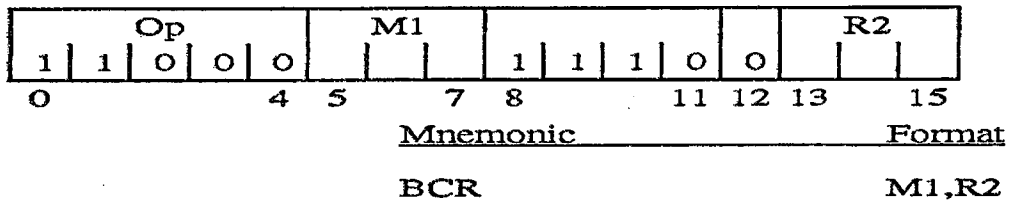
37800 C7F3 F20E Unconditional Branch back to Loc. 3720E

BRANCHING: Branch and Link & BCR

BRANCH AND LINK: EXTENDED ADDRESSING



Simple method to branch anywhere within the current sector of execution. Recommend B2 = 3. 16 Bit Displacement = "Branch To" Address, e.g. the start of allocated Patch Space. Upon execution, the contents of the Current PSW (pointing to the next sequential location) including BSR and DSR is stored in the Register specified in R1 Field. Commonly used with a BCR instruction to return from Patch Area.



R2 same Reg as R1 in BAL.
RETURNS to LOC after BAL.

BRANCHING: BC vs BAL- PATCH PROS / CONS

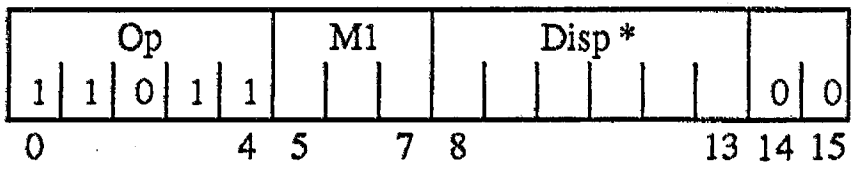
- PROS / CONS : Branch on Condition (BC) - C7F3 XXXX
  - PRO: A BC to branch to the patch space uses No additional Register for Patch : The Number of Registers available for Patch use may be limited without storing / reloading register contents as part of the patch
  - CON: Requires use of fullword BC to branch back to patched code. (Normally not a real limitation, i.e. memory space in patch area is normally not a concern)

- PROS / CONS of a BAL / BCR Pair - E7F3 XXXX
  - PRO: A BAL / BCR Pair use only three (3) HWs to branch to patch space and back to original code, i.e. BCR is an HW instruction
  - CON: A BAL uses a register to save the return address and BCR must return to location following BAL which MAY NOT be desirable. (Beware of using Reg 7 in FCOS patch or Reg 4 in HAL Process)

BRANCHING: Instruction Counter Forward

Used when Branching short distance (37 Hex HWs or Less) from the current PSW Address. NOTE: PSW ADDR is pointing to the next location after the BCB when Disp is calculated.

5.6 BRANCH ON CONDITION FORWARD



Mnemonic	Format
BCF	M1,D2

* Displacements of the form 111XXX are not valid.

DF00 or D800 can be used to NO-OP an Instruction

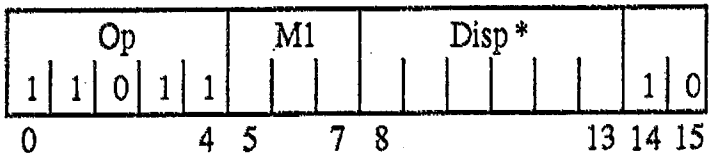
DESCRIPTION:

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).

BRANCHING: Instruction Counter Backward

Used when Branching short distance (37 Hex HWs or Less) from the current PSW Address. NOTE: PSW ADDR is pointing to the next location after the BCB when Disp is calculated.

5.4 BRANCH ON CONDITION BACKWARD



* Displacements of the form 111XXX are not valid.

Mnemonic	Format
BCB	M1,D2

DESCRIPTION:

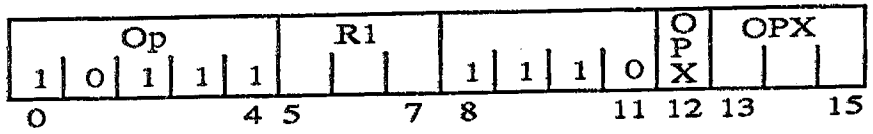
This instruction tests the PSW condition code status bits. Instruction bits 5 through 7 (the M1 field) specify which condition code (bits 16 and 17 of the PSW) is to be tested. Instruction bit 5 tests for a code equal 00, instruction bit 6 tests for a code equal 11, and instruction bit 7 tests for a code equal 01. Whenever the condition code test is successful, the branch is taken by 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).

ADDR. MODES - SRS / RS REVIEW 1

- SRS Operand Addressing (Pg 18-20)
  - All Examples are HW Operands
- RS Format Operand Addressing (B2 = 3) (Pg. 21)
  - ADDR SPEC Treated as Immediate Data / Eff. Addr.
- RS Format Extended Addressing (Pg. 22)

```
                                  ST#18     EQU     *
E9F3   088E   00088E                       LA      R1,X'088E'
B914          000005                       STH     R1,5(R0)
E0FB   001C                                IAL     R0,X'001C'
EB99          0008B4                       LA      R3,X'0026'(R1)
BB24          000009                       STH     R3,9(R0)
                                  ST#524    EQU     *
1DF3   26F6   0026F6                       L       R5,X'26F6'
25F1   0212   000AA0                       N       R5,X'0212'(R1)
```

4.17 LOAD FIXED IMMEDIATE



Mnemonic	Format
LFXI	R1,Value

CONSIDER USING LHI - Does Require FW Instr.

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.

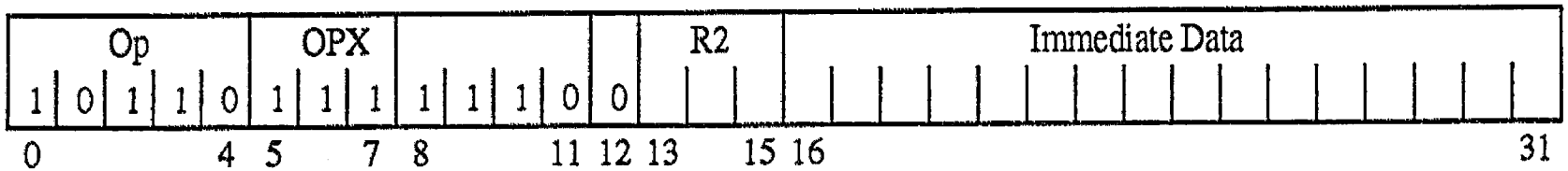
OPX (Bits 12, 13, 14, & 15) (hex)	Immediate Value --> R1 (hex)
0	FFFE0000
1	FFFF0000
2	00000000
3	00010000
4	00020000
5	00030000
6	00040000
7	00050000
8	00060000
9	00070000
A	00080000
B	00090000
C	000A0000
D	000B0000
E	000C0000
F	000D0000

OPX = 8; R1 = 0006 0000

MULTIPLY (Fixed Pt): M, MR, MH, or MHI

After a Multiply Instruction (Not Floating Pt.), other than the Multiply Integer Half (MIH), the PRODUCT needs to be shifted Right One if the RESULT is to be interpreted as an Integer. (Sneaky - Results NOT Intuitively Obvious) (SAM 19)

4.23 MULTIPLY HALFWORD IMMEDIATE



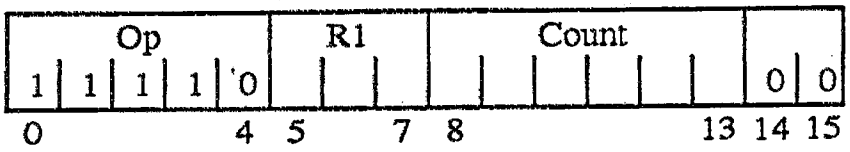
Mnemonic	Format
MHI	R2,Data

Example provided after Shift Instructions

DESCRIPTION:

Instruction bits 16 through 31 are treated as immediate data. This halfword of immediate data is the multiplier. The contents of bits 0 through 15 of general register R2 are the halfword multiplicand. The product of the multiplier and the multiplicand is a 32-bit signed fraction. Both multiplier and multiplicand are 16-bit signed twos complement fractions. This product is saved in general register R2.

LOGICAL SHIFT OPERATIONS



Shift Right Logical - SRL
Similar Operation to SLL

Mnemonic	Format
SLL	R1,Count

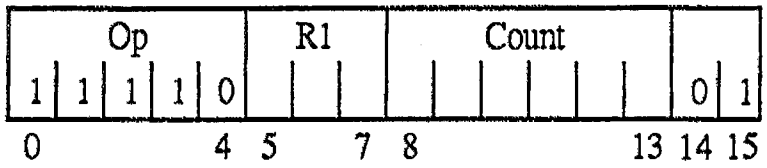
DESCRIPTION:

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.

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

ARITHMETIC SHIFT INSTRUCTIONS

6.4 SHIFT RIGHT ARITHMETIC



Mnemonic	Format
SRA	R1,Count

Refer to SAM 19 Attachment Par. 4.

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.

EXAMPLE: F519 SRA R5, 6

R5 Prior to Shift = 1000 0000 0000 0000 0000 0000 1100 0000

R5 After Shift = 1111 1110 0000 0000 0000 0000 0000 0011

BEWARE: Do NOT recommend Patch which requires use and manipulation of NEGATIVE Fixed Point numbers since data is represented in Two's (2s) Complement Format. Too Risky for Patch unless absolutely required.

MULTIPLY OPERATION / SHIFT RESULTS

EXAMPLE OF MULTIPLY and SHIFT RESULTS:

OBJECT:	INSTRUCTION	RESULT:
ECF3 0002	LHI R4,X'0002'	R4 = 0000 0000 0000 0010 0000 0000 0000 0000
B7E4 000A	MHI R4, 10	

Multiply Operation: Multiplier = (10 Dec.) 0000 0000 0000 1010

R4 = (2 to -14) X Multiplier (2 to -14 + 2 to -12) = 2 to -28 + 2 -26)

R4 = 0000 0000 0000 0000 0000 0000 0010 1000 (Value of 28 Hex = 40 Dec)

F405	SRA	R4, 1	R4 = 0000 0000 0000 0000 0000 0000 0001 0100
			(R4 FW Integer Value = 14 Hex = 20 Dec)

(The Following shifts are for SHIFT DEMO ONLY)

F46C	SLL	R4, 27	R4 = 1010 0000 0000 0000 0000 0000 0000 0000
F40D	SRA	R4, 3	R4 = 1111 0100 0000 0000 0000 0000 0000 0000
F40C	SLL	R4, 3	R4 = 1010 0000 0000 0000 0000 0000 0000 0000

PATCH RESTRICTIONS / PRECAUTIONS

- **UNDERSTAND USER NOTE 37547 Before Proceeding with Patch Consideration, Design, and Implementation Method (BEWARE !!)**

GPC MAIN MEMORY CAN BE PATCHED IN REAL-TIME BY USING THE GPC MEMORY DISPLAY (G901 OPS DISPLAY AND SPEC 000), BY UPLINK, BY TCS AND BY SACS. USERS SHOULD BE AWARE OF THE FOLLOWING RESTRICTIONS ASSOCIATED WITH THE PATCHING OF CODE OR DATA USING THESE CAPABILITIES:

A. PATCHING SINGLE OR MULTIPLE 16-BIT HALFWORDS IN A SINGLE GPC, CS OR AN RS, SHOULD BE AVOIDED IF THE CODE OR DATA BEING PATCHED IS CONCURRENTLY BEING USED BY OTHER CPU OR IOP PROGRAMS. EXAMPLES OF POTENTIAL PROBLEMS INCLUDE:

-CYCLIC PROGRAMS MAY NEGATE OR MODIFY THE EFFECT OF A PATCH TO A DATA AREA IN USE.

-IF THE PATCH IS TO AN I/O PROGRAM AREA OR BUFFER BEING REFERENCED OR MODIFIED BY THE IOP, THE RESULT COULD BE ERRONEOUS OR NON-HOMOGENEOUS INPUTS OR OUTPUTS.

-IN A CS THE APPLICATION OF THE PATCH WILL NOT BE APPLIED AT THE SAME POINT IN THE CS CODE EXECUTION IN ALL SET MEMBERS.

B. CONCURRENT PATCHING OF MULTIPLE HALFWORDS (INCLUDES TWO 16-BIT HALVES OF A 32-BIT CODE/DATA WORD) VIA UPLINK, TCS OR SACS SHOULD ONLY BE PERFORMED WHEN THE "CONTIGUOUS" OPTION IS USED

- CONTINUE ON SUPPLEMENT PAGE -

PATCH RESTRICTIONS / PRECAUTIONS

- **UN 37057 Continued:**

WHICH ALLOWS MULTIPLE HALFWORD PATCHING WITHOUT INTERRUPTION. SINCE THE "CONTIGUOUS" OPTION IS NOT AVAILABLE WITH THE ONBOARD GPC MEMORY SPEC, MULTIPLE HALFWORD PATCHING SHOULD BE AVOIDED WHEN USING THE ONBOARD SPEC. EXAMPLES OF POTENTIAL PROBLEMS INCLUDE:

-IN A SINGLE GPC OR RS, PATCHING OF SUCCESSIVE HALFWORDS MAY NOT OCCUR CONCURRENTLY BUT BE INTERRUPTED BY OTHER PROCESSING WHICH MAY USE PARTIALLY PATCHED CODE OR DATA.

-IN DISSIMILAR GPC'S OF A CS, PATCHING OF SUCCESSIVE HALFWORDS MAY NOT ONLY BE INTERRUPTED IN EACH GPC, BUT ALSO BE APPLIED AT DIFFERENT POINTS IN THE CS CODE WITHIN SET MEMBERS.

C. THE PATCHING OF MULTIPLE CONTIGUOUS HALFWORDS USING THE "CONTIGUOUS" OPTION (ONLY AVAILABLE WITH UPLINK, TCS AND SACS) SHOULD BE AVOIDED IF THE PATCH IS TO BE APPLIED TO AN I/O AREA (MSC PROGRAMS, BCE PROGRAMS, AND I/O BUFFERS). THERE IS A POTENTIAL FOR THE IOP TO BE ACCESSING OR STORING INTO THESE AREAS WHILE THE PATCH IS TAKING PLACE.

VIOLATION OF THE RESTRICTIONS CAN RESULT IN A PATCH WHICH SOMETIMES WORKS BUT SOMETIMES DOES NOT. THIS IS DUE TO DYNAMIC INTERACTION OF THE PROGRAM BEING PATCHED, THE PATCH PROGRAM, I/O ACTIVITY, AND TIMER AND I/O INTERRUPTS. FAIL-TO-SYNC, DIVERGENT COMPUTATIONS, OR COMPLETE LOSS OF THE PROGRAM ARE POSSIBLE RESULTS.

USER RESPONSE:

DESIGN PATCHES TO ABIDE BY THE ABOVE RESTRICTIONS. EXCEPTIONS SHOULD BE HANDLED ON AN INDIVIDUAL BASIS.

PATCH DEVELOPMENT: CAUTIONS

- IF you compile your logic changes to determine the Object code for the Patch, BEWARE !!!!!
  - Compiler optimization may cause other instructions to be changed
  - Requires DETAILED COMPARISON / ANALYSIS of surrounding code

- Pay EXTREMELY CLOSE ATTENTION to use of Registers / Don't Destroy the Contents of Regs used after patch instructions, for example:
  - Reg 0 : Stack Space for HAL programs
  - Reg 1: #D Program Data Area
  - Reg 4: Return Reg for Procedure Returns to calling program
  - Reg 7: FCOS Return Reg to calling program
  - CHECK ALL REGS !!!!! DO IT AGAIN !!!!!
  - Be AWARE of DSE Contents
- Be Aware of Problem vs Supervisor State and use of Privileged Instructions, e.g. SSW UI/SC CANNOT EXECUTE Privileged Instructions (See P.O.O.)

KIS? - KEEP IT SIMPLE (MORE CAUTIONS)

- Steer Clear of Logic / Instructions requiring complex or extensive data manipulation, e.g. Two's Complement Data, Divide Instruction, etc. (Refer to detailed example of Divide Operation in SAM 19 Attachment)

- Beware of Loop Counts and Iterative Operations (e.g. Decrement before or after Test or Operation), i.e. Branch on Counts, Move Halfword, etc.

- Beware of Indexed Addressing Modes

- Beware of changing contents of Stack Data, Stack Pointers, or Manipulating data from the Stack

- Review and HEED SAM 19 - Coding Constraints and Restrictions

- Review SAM 23 - FSW Patch Considerations and Constraints

OPERATIONAL ENVIRONMENT / INSTALLATION

- Environment Considerations / Patch Formats
  - FEID
    - LSTOP and PATCHFC Cards
  - Orbiter Pre-Launch
    - MM Universal Patch Format (UPF) - Output from Tool
  - Orbiter Flight
    - GMEM (GPC MEMORY Keyboard Items). MCC may convert to Uplink Format (GMEM OP Codes) - REMEMBER UN 37547. Provide EXPLICIT WARNINGS / DIRECTIONS ref. Patch Installation (if required).

    most always
  - Labs (SAIL, JAEL, SMS, KATS)
    - GPC MEMORY Keyboard Items
    - Raw Hex Code via Facility Unique AGE Application (GIC, SID, etc.)
      - BEWARE of "Patch on the Fly" Capability
    - UPF

RUN TIME PATCH INSTALLATION

- Patch Installed during PASS Code Execution
  - PATCH with Patch Space
    - Install / Apply Code in Patch Area First
    - Install / Apply Branch to Patch Area
  - In-Line Code
    - Conform to precautions in UN 37547 (Especially if multiple HWs must be patched, i.e. DO NOT USE SPEC 0 !!!!!! - NOT EVEN for a FW Branch to Patch Area !!!!)

↓ issues w/ interrupt protection

WHAT / HOW / WHERE

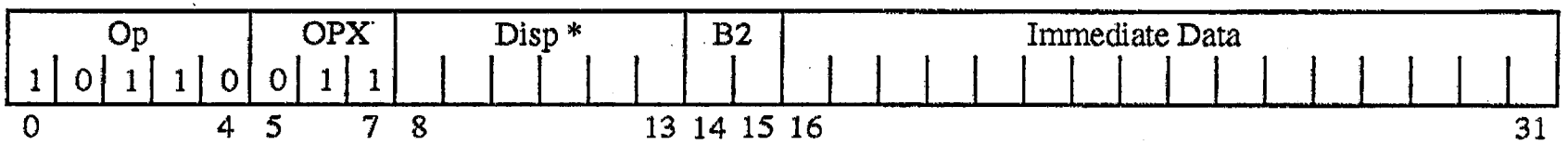
- PATCH REQUIREMENT / CODE ANALYSIS
  - Delete Logic (In-Place)
    - No-Op one or more assembler instructions
    - Skip or Branch Around multiple locations, e.g. multiple assembler instructions to "delete" an "IF" statement
  - Modify Logic / Data Value (Possibly In-Place; Patch over old code / data)
    - Reference a different parameter
    - Change Condition (= , >, <, combination)
    - Change Data Value
  - New Logic or New Data Value (Probable Patch Space Required)
    - Assembler Instructions to add new Logic (IF ?) Statement or Combination of Above
  - OTHER (?)

SAMPLE PATCH PROBLEM CODE

- NOTE: The sample problems based on this code are intended only to highlight patch techniques; the actual logic of the patches for the Shuttle would NOT be valid !!!
- Source Code from ARAGPCSW - Executes cyclically every 960 ms

SRN	STMT	HAL CODE
006600	301	ARAB_NEW_DISC(2) = ON;
006630	302	IF (ARAB_OLD_DISC(6) = ON) AND
		(ARAB_NEW_DISC(2) = OFF) AND
		(ARAB_OLD_DISC(8) = OFF) THEN DO;
006670	304	ARAB_OLD_ DISC(8) = ON;
006680	305	END;
006690	306	ELSE DO;
006700	307	ARAB_OLD_DISC(8) = OFF;
006710	308	END;

7.15 TEST BITS



Mnemonic	Format
TB	D2(B2),Data

* Displacements of the form 111XXX are invalid.

DESCRIPTION:

Bits 16 through 31 of this instruction are treated as immediate data. This halfword immediate data is logically tested with the halfword main storage operand. A one in the immediate data tests the corresponding bit in the halfword main storage operand. The halfword main storage operand is not changed. The result of the test is given in the condition code.

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.

SAMPLE CODE - ASSEMBLER LISTING

- **ASSEMBLED CODE** (can be deceiving; consider looking @ DASS)

```
0000088                                 ST#301   EQU     *
00088 B27D 0001        001F                      SB      31(R1),1
000008A                                 ST#302   EQU     *
000008A                                 LBL#57   EQU     *
0008A B39D 0400        0027                      TB      39(R1),1024
0008C DC24             0096                      BCF     4,*+10
000008D                                 LBL#60   EQU     *
0008D B37D 0001        001F                      TB      31(R1),1
0008F DB18             0096                      BCF     3,*+7
0000090                                 LBL#63   EQU     *
00090 B39D 0100        0027                      TB      39(R1),256
00092 DB0C             0096                      BCF     3,*+4
0000093                                 LBL#66   EQU     *
0000093                                 LBL#65   EQU     *
0000093                                 ST#303   EQU     *
0000093                                 ST#304   EQU     *
00093 B29D 0100        0027                      SB      39(R1),256
0000095                                 ST#305   EQU     *
0000095                                 ST#306   EQU     *
00095 DF08             0098                      BCF     7,*+3
0000096                                 LBL#64   EQU     *
0000096                                 ST#307   EQU     *
00096 B69D FEFF        0027                      NIST    39(R1),-257
```

SAMPLE CODE - DASS LISTING

- **CODE from DASS:**

```
040C8A          $0ARAGPC+0088                          ST#301   EQU   *                    THEN
040C8A-040C8B   $0ARAGPC+0088   B27D  0001  0381DB              SB    X'001F'(R1),1
040C8C          $0ARAGPC+008A                  (F )    ST#302   EQU   *                    IF
040C8C-040C8D   $0ARAGPC+008A   B39D  0400  0381E3              TB    X'0027'(R1),X'0400'
040C8E          $0ARAGPC+008C   DC24        040C98              BCF   4,9
040C8F-040C90   $0ARAGPC+008D   B37D  0001  0381DB              TB    X'001F'(R1),1
040C91          $0ARAGPC+008F   DB18        040C98              BCF   3,6
040C92-040C93   $0ARAGPC+0090   B39D  0100  0381E3              TB    X'0027'(R1),X'0100'
040C94          $0ARAGPC+0092   DB0C        040C98              BCF   3,3
040C95          $0ARAGPC+0093                          ST#303   EQU   *                    THEN
040C95          $0ARAGPC+0093                          ST#304   EQU   *                    ASSI
040C95-040C96   $0ARAGPC+0093   B29D  0100  0381E3              SB    X'0027'(R1),X'0100'
040C97          $0ARAGPC+0095                          ST#305   EQU   *                    END
040C97          $0ARAGPC+0095   DF08        040C9A              BCF   7,2
040C98          $0ARAGPC+0096                          ST#306   EQU   *                    ELSE
040C98          $0ARAGPC+0096                  (F )    ST#307   EQU   *                    ASSI
040C98-040C99   $0ARAGPC+0096   B69D  FEFF  0381E3              NIST  X'0027'(R1),X'FEFF'
```

MEMORY MAP for PATCH AREA SAMPLE PROB.

- **DASS MEMORY MAP SHOWING PATCH AREA**

```
0472B2-04731D  EXP       **** 006C(  108)  H A L   L I B R A R Y   C O D E
04731E-04736D  LOG       **** 0050(   80)  H A L   L I B R A R Y   C O D E
04736E-04739D  SQRT      **** 0030(   48)  H A L   L I B R A R Y   C O D E
04739E-0473A3  VV0DN     **** 0006(    6)  H A L   L I B R A R Y   C O D E
0473A4-0473A9  VV0SN     **** 0006(    6)  H A L   L I B R A R Y   C O D E
0473AA-0473B7  VV9S3     **** 000E(   14)  H A L   L I B R A R Y   C O D E
0473B8-0473FD  VV10D3    **** 0046(   70)  H A L   L I B R A R Y   C O D E
0473FE-04742B  VV10S3    **** 002E(   46)  H A L   L I B R A R Y   C O D E
04742C-04746B  CASPV     **** 0040(   64)  H A L   L I B R A R Y   C O D E
04746C-0474BB  CPAS      **** 0050(   80)  H A L   L I B R A R Y   C O D E
0474BC-0474CD  CPASP     **** 0012(   18)  H A L   L I B R A R Y   C O D E
0474CE-0474DB  GTBYTE    **** 000E(   14)  H A L   L I B R A R Y   C O D E
0474DC-0474F1  STBYTE    **** 0016(   22)  H A L   L I B R A R Y   C O D E
0474F2-047591  $Y028001  **** 00A0(  160)  P A T C H
047592-047593  --------  **** 0002(    2)  C H E C K S U M
048000-04800F  #Y029001  **** 0010(   16)  P A T C H
048010-048294  #PCDTANN  **** 0285(  645)  DATA
048296-048297  --------  **** 0002(    2)  C H E C K S U M
048298-0482DB  #PCDZANN  **** 0044(   68)  DATA
0482DC-048657  #PCDKANN  **** 037C(  892)  DATA
048658-048888  #PCDSANN  **** 0231(  561)  DATA
04888A-04888B  --------  **** 0002(    2)  C H E C K S U M
050000-05000D  $Y131000  **** 000E(   14)  P A T C H
05000E-05000F  --------  **** 0002(    2)  C H E C K S U M
050010-050A63  #PCDQANN  **** 0A54( 2644)  DATA
050A64-050A65  --------  **** 0002(    2)  C H E C K S U M
```

SAMPLE PATCH PROBLEM 1

- PROBLEM 1: Patch Requirement: "Functionally" Delete the first discrete check in SRN 6630, i.e. the check of (ARAB_OLD_DISC(6) = ON)

MULTIPLE SOLUTIONS (Each may have unique issues / considerations):

LOC	ARA. DISP	OBJECT CODE	ASSEMBLY	COMMENT
40C8C	$0ARAGPC.8A *	DF08	BCF 7, 2	BR to 40C8F

(Branch Disp ??? - IC = ??)

OR

40C8C	$0ARAGPC.8A	C7F3 8C8F (*)	BC 7, x'8C8F'	BR to 40C8F

(Installation Issue ???) (*) $0ARAGPC.008D (PATCHFC Format)

OR

40C8E	$0ARAGPC.8C	DF00	BC 7, 0	No-OP
40C8E	$0ARAGPC.8C	DC00	BC 4, 0	No Eff. Br

SAMPLE PATCH PROBLEM 2

- Problem 2: Patch Requirement- Add an additional "AND" condition to the original IF Statement at SRN 006630. Make the Patch In-Line Code, i.e. do NOT use a Patch Area.

SRN	STMT	HAL CODE
006600	301	ARAB_NEW_DISC(2) = ON;
006630	302	IF (ARAB_OLD_DISC(6) = ON) AND
		(ARAB_NEW_DISC(2) = OFF) AND
		(ARAB_OLD_DISC(7) = OFF) AND 
		(ARAB_OLD_DISC(8) = OFF) THEN DO;

PATCH:

LOC	ARA DISP	OBJECT CODE	ASSEMBLY	COMMENT
40C93	$0ARAGPC.91	- - - - 0300	X'0300'	Test Mask
(Effective Instruction)		(B39D) 0300	TB X'0027'(R1), X'0300'	
				(Tests both bits 7 & 8)

SAMPLE PATCH PROBLEM 3

- Problem 3: Patch Requirement- Add an additional "AND" condition to the original IF Statement at SRN 006630. Given: ARAB_NEW_DISC(1) = Bit 15; Patch Space Assigned: $Y028001 + 0E.

SRN	STMT	HAL CODE
006600	301	ARAB_NEW_DISC(2) = ON;
006630	302	IF (ARAB_OLD_DISC(6) = ON) AND
		(ARAB_NEW_DISC(2) = OFF) AND
		(ARAB_NEW_DISC(1) = ON) AND
		(ARAB_OLD_DISC(8) = OFF) THEN DO;

Plagiarize surrounding Code when possible. USE DASS

PATCH AREA CODE:

LOC	CSECT. DISP	OBJECT CODE	ASSEMBLY	COMMENT
47500	$Y028001.0E	B37D 0002	TB X'001F'(R1),X'0002'	New Check
47502	$Y028001.10	C4F3 8C98 *	BC 4,X'8C98'	BR NQ ON (1)
47504	$Y028001.12	B39D 0100	TB X'0027'(R1),X'0100'	RESTORE
47506	$Y028001.14	C7F3 8C94 *	BC 7, X'8C94'	RETURN

* ($0ARAGPC. nn)

Sample Patch Problem 3

- Branch to Patch Area

LOC	ARA DISP	OBJECT CODE	ASSEMBLY	COMMENT
40C92	$0ARAGPC. 90	C7F3 F500 *	BC 7,X'F500'	BR to Patch
		* $Y028001. 0E		

- INSTALLATION ISSUES ? REAL TIME, i.e. Code Executing
  - Order of Installation (Patch Space First then Branch to Patch)
  - Redundant Set ??
  - Common Set ??
  - SPEC 0 ??
  - UPLINK GMEM SCATTER ??
  - Uplink GMEM CONTIGUOUS ??

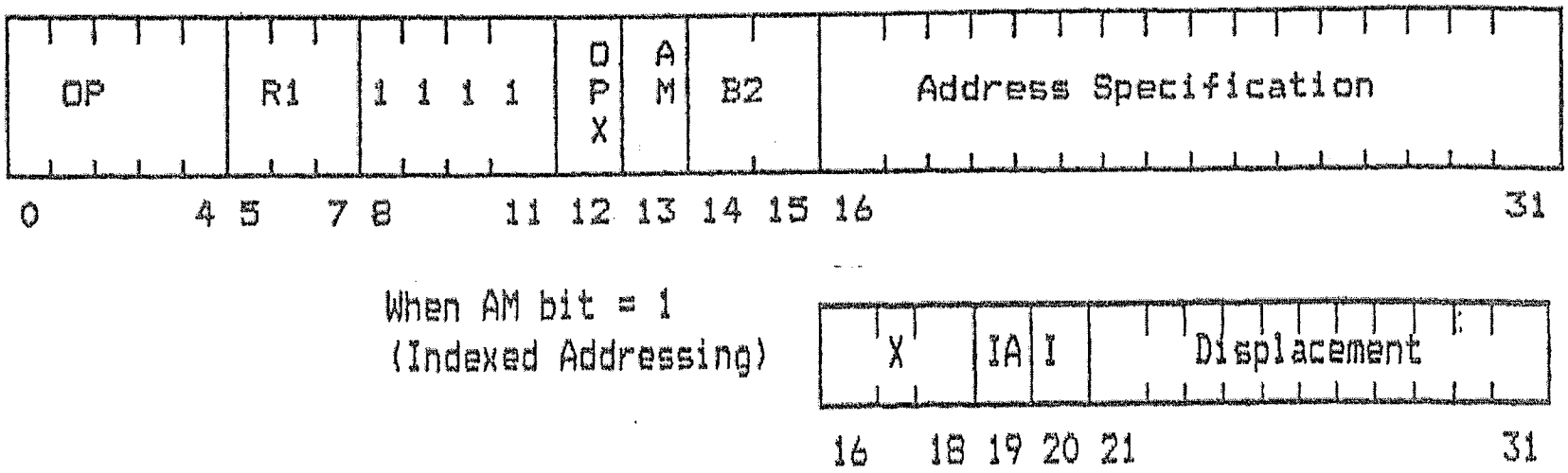
EPILOGUE / ACKNOWLEDGEMENTS

- IMPORTANT PATCH TOPICS NOT COVERED
  - Use of PATCHFC in FEID Patch Installation
  - Use of MM Patch Build Tool (FBLDBAP ?) (Willie ???)
  - Inspection / Verification of PATCH LOG contents (Willie ???)
  - Conversion of G3 Patch to G3 Archive Format (See "G3 ARCHIVE PATCHING FOR DUMMIES" by Nathan Bright)

- Some Data Plagiarized from: "PATCH METHODOLOGY" ; "Original" Author Unknown
- Other Training/References Available for AP-101 Assembler Use / Development:
  - AP101 Assembler Language Introduction OTP275 (8/99) - Connie Steed
  - AP101 Assembler Language Introduction OTP275.03 (2/01) - Alisha Witty
  - AP101 Assembler User's Guide
- Space Shuttle Model AP-101S Principles Operation with Shuttle Instruction Set (SFOC-OE0001) - THE FINAL AUTHORITY

BACKUP CHARTS - Not Covered This Session

ADDR MODES: RS INDEXED ADDRESSING



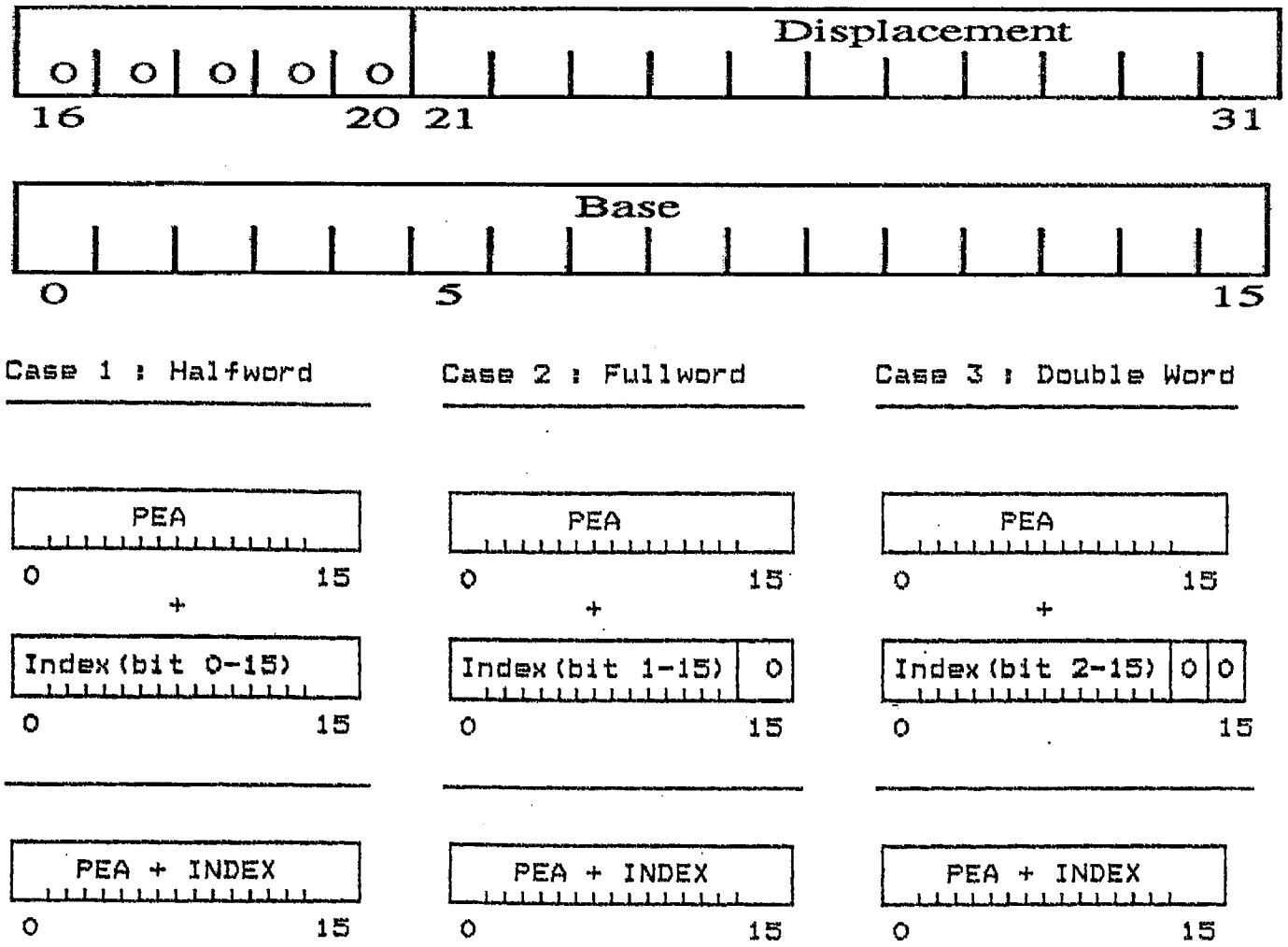
Indexed Addressing is performed when the AM Bit = 1. This mode provides an 11 bit Displacement to be added to the Base. The Displacement is always aligned right justified with the Base regardless of Operand size (HW, FW, DW). This provides the PEA (Preliminary Effective Address). If X is NQ 0, i.e. an Index Reg (1-7) is used, Bits 0-15 of the Index are aligned with the PEA based upon the type of Operand, i.e. HW, FW, DW. Index is shifted one bit to left for FW Operand Addressing and 2 bits to left for Doubleword (DW) Operand Addressing. See page 55.

ADDR MODES: RS INDEX ADDRESSING TYPES

AM	X	IA	I	Addressing Type	Function
0	---	-	-	Extended Addressing	EA = PEA
1	000	0	0	IC Relative	EA = IC + PEA
1	000	0	1	IC Relative	EA = IC - PEA
1	000	1	0	Indirect Addressing	EA = MS(PEA)
1	000	1	1	Indirect Addressing with Storage Modification	EA = MS(PEA) MS(PEA) = MS(PEA) + MS(PEA+1)
1	$\overline{000}$	0	0	Indexed Addressing	EA = (X) + PEA
1	$\overline{000}$	0	1	Indexed Addressing with Index Modification	EA = (X) + PEA (X.low) = (X.low) + (X.high)
1	$\overline{000}$	1	0	Indirect Addressing with Post Indexing	EA = MS(PEA) + (X)
1	$\overline{000}$	1	1	ZCON Addressing	EA = determined from MS(PEA)

ADDR MODES: 16 Bit EA = PEA + Index

Preliminary Effective Address (PEA) = Disp + [Base (if B2 NQ 3)]



ADDR MODES: IC RELATIVE

ASSUME: DSR = 1 Register 2 = 2000 0000

BSR = 4 Register 3 = 000A 0000

DSEs = 0 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20004	1CF7 000C	L 4, 12(0,3)	1	000	0	0	R4=1212 3456
20006	....						

20012 1212 3456

Sector 0 ADDR:	Contents:	Sector 1 ADDR:	Contents:
0000C	1111 0000	08008	ABAB FFFF
		0800A	000C CF00

02004 A688 0E30 (ZCON)

02006

ADDR MODES: INDIRECT

ASSUME: DSR = 1 Register 2 = 2000 0000

BSR = 4 Register 3 = 000A 0000

DSEs = 0 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20006	1CF6 1010	L 4,16(0,2)	1	000	1	0	R4=0001 1111

Sector 0 ADDR:	Contents:
00010	1111 0000
02010	800A 0000
02012	8008 0002

Sector 1 ADDR:	Contents:
08008	ABAB FFFF
0800A	0001 1111

ADDR MODES: INDIRECT with STORAGE MOD.

ASSUME: DSR = 1 Register 2 = 2000 0000

BSR = 4 Register 3 = 000A 0000

DSEs = 0 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20008	1CF6 1812	L 4, 18(0,2)	1	000	1	1	R4=1212 1111

Sector 0 ADDR:	Contents:
00012	1111 0000
02010	800A 0000
02012	8008 0002
(02012	800A 0002) (After Instruction Execution)
02014	2004 0000

Sector 1 ADDR:	Contents:
08008	1212 1111
0800A	000C CF00

ADDR MODES: INDEXED ADDRESSING

ASSUME: DSR = 1 Register 2 = 2000 0000

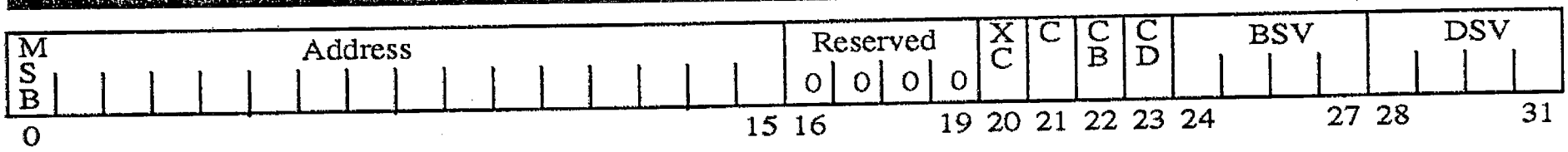
BSR = 4 Register 3 = 000A 0000

DSEs = 0 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
2000A	1CF6 A014	L 4, 20(5, 2)	1	101	0	0	R4=1230 1111

Sector 0 ADDR:	Contents:		Sector 1 ADDR:	Contents:
00014	82F0 0031		08008	ABAB FFFF
			0800A	000C CF00
02014	2004 0000	(PEA)		
02016	001C 0000	←		
02018	1230 1111	← + Index (R5) - Offset for FW		

ADDR MODES: ZCON FORMAT & EXAMPLES



Field	Function
X_C	Index Control
C	Control to allow PSW modification
C_B	Control BSV Usage
C_D	Control DSV Usage
BSV (Branch Sector Vector)	Selectively replaces BSR in PSW
DSV (Data Sector Vector)	Selectively replaces DSR in PSW
MSB (Most Significant Bit)	Determines type of address expansion

BRANCH	XC	C	CB	CD	FUNCTION
NO	0	0	-	-	EA = DSV [(X) + ADDR]
YES	0	1	0	0	BA = BSR [(X) + ADDR]
YES	1	1	0	1	DSR = DSV and BA = BSR [ADDR]
YES	0	1	1	0	BSR = BSV and BA = BSR [(X) + ADDR]
YES	1	1	1	0	BSR = BSV and BA = BSR [ADDR]

ADDR MODES: ZCON OPERAND ADDRESSING

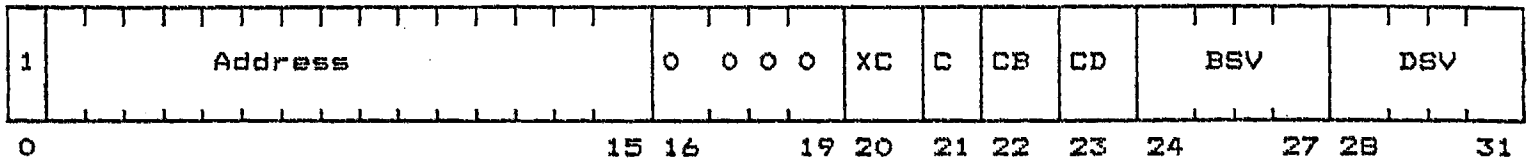
ASSUME: DSR = 1 Register 2 = 2000 0000

BSR = 4 Register 3 = 000A 0000

DSEs = 0 Register 5 = 0002 0000 (Refer to Pg. 54)

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20000	1CF6 B804	L 4, 8(5 , 2)	1	101	1	1	R4=2111 1111

Sector 0 ADDR:	Contents:	Sector 3 ADDR:	Contents:
02004	800A 0003	1800A	ABAB FFFF
		1800C	0101 1010
ZCON		1800E	2111 1111



Branch?	XC	C	CB	CD	Function
No	0	0	–	–	EA = DSV || [(X) + Addr]

ADDR MODES: ZCON BRANCH ADDRESSING

ASSUME: DSR = 1 Register 2 = 2000 0000 DSEs = 0

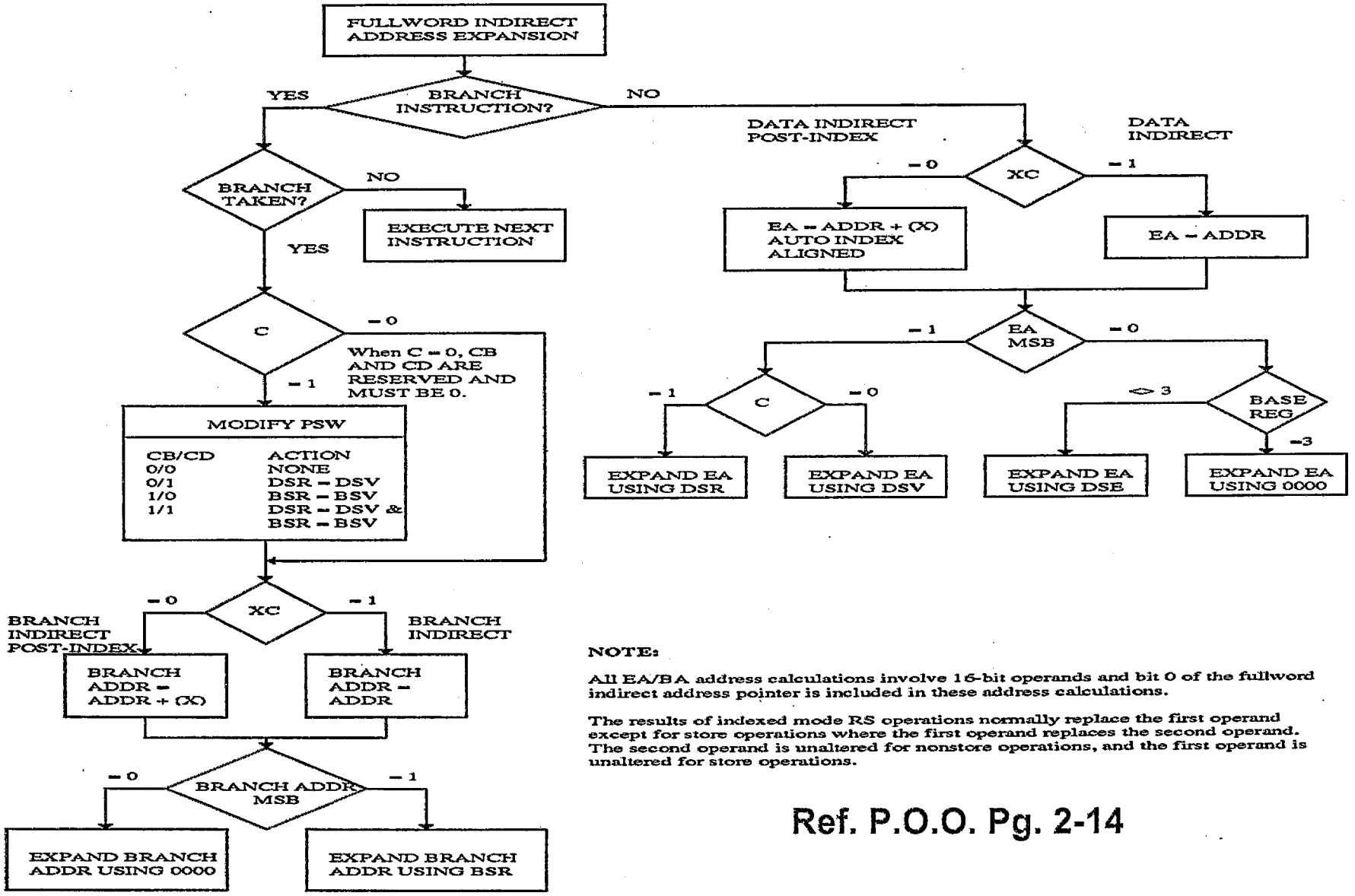
BSR = 4 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20002	C7F6 B80A	BC 7, 10 (5 , 2)	1	101	1	1	BA=27FF0
20004	EAF3 FFFF	???? (Review ?)					??????
27FF0	C7F3 8004	BC 7, X'8004'	0				BA=?? (Quiz)

Sector 0:	Contents:	XC	C	CB	CD	Branch Addr. (BA) Calc.
0200A	FFEE 0400 (ZCON)	0	1	0	0	BSR [Index + ZCON ADD]
						4 [0002 + FFEE]

Ref. ZCON Addressing Flow Pg. 63 or P.O.O. Pg 2-14

ADDR MODES: ZCON ADDRESS CALCULATION



NOTE:

All EA/BA address calculations involve 16-bit operands and bit 0 of the fullword indirect address pointer is included in these address calculations.

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

Ref. P.O.O. Pg. 2-14

ADDR MODES: ZCON BRANCH with DSR CHANGE

ASSUME: DSR = 1 Register 2 = 2000 0000 DSEs = 0

BSR = 4 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20004	C7F6 B80C	BC 7, 12(5, 2)	1	101	1	1	BA=20800 and DSR = 2
20800	9CF3 0140	LH R4,X'0140"	0				R4=(Quiz)

Sector O:	Contents:	XC	C	CB	CD	Branch Addr. Calculation
00140	8600 0000					
02000	8800 0D02	1	1	0	1	BSR [ZCON ADDR]

NOTE: Current PSW DSR changed from 1 to 2 (DSR = DSV in ZCON)

ADDR MODES: ZCON BRANCH with BSR CHANGE

ASSUME: DSR = 1 Register 2 = 2000 0000 DSEs = 0

BSR = 4 Register 5 = 0002 0000

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I	RESULT
20006	C7F6 B80E	BC 7, 14(5, 2)	1	101	1	1	BA=2800C and BSR = 5
2800C	9CF3 0140	LH R4,X'0140"	0				

Sector O:	Contents:	XC	C	CB	CD	Branch Addr. Calculation
0200E	800A 0650	0	1	1	0	BSR [Index + ZCON ADD]
						5 [0002 + 800A]

NOTE: Current PSW BSR changed from 4 to 5 (BSR = BSV in ZCON)

ADDR MODES: ZCON BRANCH with BSR CHANGE

ASSUME: DSR = 1 Register 1 = 821C 0000 (at SCAL) DSEs = 0

BSR = 8 Register 3 = 8270 0000

Register 0 = 1458 0014 (Stack Pointer)

ADDR:	OBJECT	INSTRUCTION	AM	X	IA	I
1EB7C	E9F3 8414	LA R1, X'8414'	0			
→ 446A5	D0FF 39D6	SCAL R0,X'01D6' (R1, R3)	1	001	1	1
446A7	68FB 04F4	LDM X'04F4'	0			

Sector O:	Contents:	XC	C	CB	CD	Branch Addr. Calculation
001D6	EB7C 0E30	1	1	1	0	BSR [ZCON ADD]
						3 [EB7C]

(Most common form of PASS ZCON for Procedure calls)

NOTE: Current PSW BSR changed from 8 to 3 (BSR = BSV in ZCON)

OI-29 PASS ZCON EXAMPLES - G1 DASS

Example PASS ZCONs for STS-110 (OI-29) DASS:

```
0001CA-0001CB   #QMKEM  +0000          EE00  0E80
0001CC-0001CD   #QVV0SN  **** 0002(    2)  H A L   L I B R A R Y   Z C O N
0001CC-0001CD   #QVV0SN +0000          F3A4  0E80
0001CE-0001CF   #QDMOD   **** 0002(    2)  H A L   L I B R A R Y   Z C O N
0001CE-0001CF   #QDMOD  +0000          F194  0E80
0001D0-0001D1   #ZDGOGSE **** 0002(    2)  ZCON
0001D0-0001D1   #ZDGOGSE+0000          C716  0E80
0001D2-0001D3   #ZARYMFB **** 0002(    2)  ZCON
0001D2-0001D3   #ZARYMFB+0000          DE98  0E80
0001D4-0001D5   #ZDCDDOW **** 0002(    2)  ZCON
0001D4-0001D5   #ZDCDDOW+0000          C30A  0E80
0001D6-0001D7   #ZDCDDG1 **** 0002(    2)  ZCON
0001D6-0001D7   #ZDCDDG1+0000          EB7C  0E30
0001D8-0001D9   #ZDCDDG2 **** 0002(    2)  ZCON
0001D8-0001D9   #ZDCDDG2+0000          DEF8  0E30
0001DA-0001DB   #ZDCDDG3 **** 0002(    2)  ZCON
0001DA-0001DB   #ZDCDDG3+0000          EB7C  0E30
0001DC-0001DD   #ZDCDDG8 **** 0002(    2)  ZCON
0001DC-0001DD   #ZDCDDG8+0000          DEF8  0E30
0001DE-0001DF   #ZDCDDG9 **** 0002(    2)  ZCON
0001DE-0001DF   #ZDCDDG9+0000          BA40  0E50
```

BRANCHING : TEST of PSW CC BITS via M1

	M1 FIELD (TEST)		
	(5)	(6)	(7)
ARITHMETIC & TALLY			
ZERO	1	0	0
NEGATIVE	0	1	0
POSTIVE (>0)	0	0	1
LOGICAL			
ZERO	1	0	0
NOT ZERO	0	1	0
TEST			
ZERO	1	0	0
MIXED	0	1	0
ALL ONES	0	0	1
COMPARE			
EQUAL	1	0	0
01 < 02	0	1	0
01 > 02	0	0	1

M1 FIELD = 111 (7)
ALWAYS BRANCH.

(Check P.O.O. for other M1 Field use, e.g. ISPB Instruction.)

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 TO 00, INSTRUCTION BIT 6 TESTS FOR A CODE EQUAL TO 11, AND INSTRUCTION BIT 7 TESTS FOR A CODE EQUAL TO 01.

IC RELATIVE ADDRESSING REVIEW

```
➔ 019829         FCMSWMON+001B   D9EA                               PC    R1,R2
  01982A         FCMSWMON+001C   DBEC                               PC    R3,R4
  01982B-01982C  FCMSWMON+001D   EEF3  001F                         LHI   R6,X'001F'
  01982D-01982E  FCMSWMON+001F   E6FB  FFFF                         IAL   R6,X'FFFF'
  01982F-019830  FCMSWMON+0021   EDF3  4000                         LHI   R5,X'4000'
  019831         FCMSWMON+0023   DEE5                               ICR   R6,R5
  019832-019833  FCMSWMON+0024   1EF7  0012  019846                 L     R6,X'0012'   ←
  019834-019835  FCMSWMON+0026   EDF3  4800                         LHI   R5,X'4800'
  019836         FCMSWMON+0028   DEE5                               ICR   R6,R5
  019837-019838  FCMSWMON+0029   B3E3  2000                         TRB   R3,X'2000'
  019839         FCMSWMON+002B   DE04        01983B                 BCF   6,1   ←
  01983A         FCMSWMON+002C   DF04        01983C                 BCF   7,1   ←
  01983B         FCMSWMON+002D                        #@LB6         DS    0H
  01983B         FCMSWMON+002D   DF4E        019829                 BCB   7,X'0013'   ←
  01983C         FCMSWMON+002E                        #@LB8         DS    0H
  01983C-01983D  FCMSWMON+002E   E8F3  0002                         LHI   R0,2
  01983E-01983F  FCMSWMON+0030   B8F3  0197  000197                 STH   R0,X'0197'
  019840-019841  FCMSWMON+0032   98F3  0140  000140                 LH    R0,X'0140'
  019842-019843  FCMSWMON+0034   ECF3  FC05                         LHI   R4,X'FC05'
  019844         FCMSWMON+0036   BC18                               STH   R4,6(R0)
  019845         FCMSWMON+0037   C7E7                               BCR   7,R7
➔ 019846-019847  FCMSWMON+0038   FFFF  FFFF           FCMPC2MX      DC    F          -1
```

DIs

NOT RU
BCB ?

Back-up / Backup Charts

- Following Data May be covered by Willie Allen Material

WHY PATCH? - AUTHORIZATIONS

- CHIT (GMEM) Request for Code Correction or Operational Workaround during Shuttle Mission
  - MCC / MER / MMT Approval
- DR Correction/ CR Change After Last Source Build, Final Load or Comp Load Release
  - SASCB Approval
- Software Approval Sheet (SAS) for Lab Operations / Testing (SAIL, SMS, etc.)
  - T&O Board Approval
- Engineering Evaluation of Prototype Solution, Potential Source Change, or Lab Test Support
  - Individual Discretion

PATCH AREAS by CSECT NAME

- TYPE 1 : PCH 04 5 P 1 (Used for FEIDTest Support, e.g. Dead/Live PICS)
  - Phase
  - Sector
  - Protected (U for Unprotected)
  - Seq #

NOT for General Use - DO NOT USE for any Delivered Patches

- TYPE 2 : $Y (or) #Y 05 4 000 (or 001) (e.g. $Y054001)
  - Code Patch Area
  - Data Patch Area
  - Phase
  - Sector
  - Seq #

TYPE 2 and TYPE 3 PATCH SPACE Managed by MM BUILD GROUP. See Willie Allen for space Allocation. Do NOT USE $T (OPS 0 ONLY)

- Type 3 : Same as Type 2 except Seq # > 001 - Allocated by MMU Build Pgm (NOT Available at Link-Edit / Comprint ; Limited FEID J-Step Test Capability)

DASS - PATCH AREA IDs

M E M O R Y M A P --- GNC2

$X030001 000654	$X050001 000602	$Y022001 010000	$Y023001 01C8F4	$Y024001 020000
$Y033001 01DEC4	$Y034001 020E6A	$Y052001 017FEA	$Y053001 01FFEA	$Y054001 027F7A
$Y057001 03FFEA	$Y131000 050000	$ZDSECLR 0004F2	$ZDSESET 0004F4	$OAIBGPC 040000

?

027F7A-027FF1	$Y054001	****	0078(120)	P A T C H	PROCEDURE GRN_OMS_ACTUATOR_FDI
027FF2-027FFD	PCH054P0	****	000C(12)	P A T C H	
027FFE-027FFF	--------	****	0002(2)	C H E C K S U M	
028000-028009	#Y025000	****	000A(10)	P A T C H	
02800A-02800B	--------	****	0002(2)	C H E C K S U M	
02800C-0280AC	#CGRNOMS	****	00A1(161)	PROCEDURE GRN_OMS_ACTUATOR_FDI	

Length = 120 HW

MMU BUILD LISTING - Ph 5 LB 18 Sector 4

MMU BUILD ALLOCATED PATCH AREAS (Ph 5 LB 18) →

								$Y054003	027F58	000002	51630	30C2
								$Y054002	027F5A	000020	51630	30C4
								$Y054001	027F7A	000078	51630	30E4
								PCH054P0	027FF2	00000C	51630	315C
5	19	PROT	51631	51631	01E1	5958		#CGRNOMS	02800C	0000A1	51631	0000