FLOATING POINT
SYSTEMS, INC.

Programmers Reference Manual Part Two

860-7319-000

by FPS Technical Publications Staff

Programmers Reference Manual Part Two

860-7319-000

1st Edition, January 1978

Publication No. FPS-7319

NOTICE

The material in this manual is for information purposes only and is subject to change without notice.

Floating Point Systems, Inc. assumes no responsibility for any errors which may appear in this publication.

Printed in U.S.A.

Table of Contents

AP-120B Instructions

Unconditional Fields

Each of the following fields may be used in any given instruction word.

Unconditional Fields

Each of the following fields may be used in any given instruction word.

Field Name	COND	DISP (Branch Displacement)	DPX	DPY
Octal Code				
0	NOP	(0-37)	NOP	NOP
1	# E-68		DB E-123	DB E-126
2	BR E-69		FA E-124	FA E-127
3	BINTRQ E-70		FM E-125	FM E-128
4	BION E-70			
5	BIOZ E-71			
6	BFPE E-71			
7	RETURN E-72			
10	BFEQ E-73			
11	BFNE E-74			
12	BFGE E-75			
13	BFGT E-76			
14	BEQ E-77			
15	BNE E-78			
16	BGE E-79			
17	BGT E-80			

Field Name	DPBS	XR (DPX Read Index)	YR (DPY Read Index)	XW (DPX Write Index)	YW (DPY Write Index)	FM
Octal Code						
0	ZERO . . . E-129	(0-7)	(0-7)	(0-7)	(0-7)	NOP
1	INBS . . . E-130					FMUL
2	VALUE* . . E-131					
3	DPX . . . E-132					
4	DPY . . . E-133					
5	MD E-134					
6	SPFN . . . E-135					
7	TM E-136					

Field Name	M1	M2	MI	MA	DPA	TMA
Octal Code						
0	FM E-99	FA E-100	NOP	NOP	NOP	NOP
1	DPX . . . E-99	DPX . . . E-100	FA E-137	INCMA . . E-140	INCDPA . . E-143	INCTMA . . E-146
2	DPY . . . E-99	DPY . . . E-100	FM E-138	DECMA . . E-141	DECDPA . . E-144	DECTMA . . E-147
3	TM E-99	MD E-100	DB E-139	SETMA . . E-142	SETDPA . . E-145	SETTMA . . E-148

*This instruction uses a 16-bit immediate VALUE as a constant or address (in bits 48-63 of this instruction). The YW, FM, M1, M2, MI, TMA, and DPA fields are then disabled for this instruction word.

SPEC Fields

One of the SPEC Fields may be used per instruction word. The S-Pad Fields (B, SOP, SOP1, SH, SPS, and SPD) are then disabled for this instruction.

Field Name	SPEC	STEST	HOSTPNL	SETPSA
Octal Code				
0	STEST E-31	BFLT E-31	PNLLIT E-39	JMPA* E-45
1	HOSTPNL E-39	BLT E-32	DBELIT E-39	JSRA* E-46
2	SPMDA E-6	BNC E-33	DBHLIT E-40	JMP* E-47
3	NOP	BZC E-34	DBLLIT E-40	JSR* E-47
4	NOP	BDBN E-35	NOP	JMPT E-48
5	NOP	BDBZ E-36	NOP	JSRT E-48
6	NOP	BIFN E-37	NOP	JMPP E-49
7	NOP	BIFZ E-37	NOP	JSRP E-50
10	SETPSA E-45	NOP	SWDB E-41	NOP
11	PSEVEN E-52	NOP	SWDBE E-42	NOP
12	PSODD E-55	NOP	SWDBH E-43	NOP
13	PS E-58	NOP	SWDBL E-44	NOP
14	SETEXIT E-66	BFL0 . . . . . . E-38	NOP	NOP
15	NOP	BFL1 . . . . . . E-38	NOP	NOP
16	NOP	BFL2 . . . . . . E-38	NOP	NOP
17	NOP	BFL3 . . . . . . E-38	NOP	NOP

Field Name	PSEVEN	PSODD	PS	SETEXIT
Octal Code				
0	RPS0A* E-52	RPS1A* E-55	RPSLA* E-58	NOP
1	RPS2A* E-52	RPS3A* E-55	RPSFA* E-59	SETEXA* E-66
2	RPS0* E-52	RPS1* E-55	RPSL* E-60	NOP
3	RPS2* E-52	RPS3* E-55	RPSF* E-60	SETEX* E-66
4	RPS0T E-52	RPS1T E-56	RPSLT E-60	NOP
5	RPS2T E-53	RPS3T E-56	RPSFT E-61	SETEXT E-67
6	NOP	NOP	RPSLP E-61	NOP
7	NOP	NOP	RPSFP E-61	SETEXP E-67
10	WPS0A* E-53	WPS1A* E-56	LPSLA* E-62	NOP
11	WPS2A* E-53	WPS3A* E-56	LPSRA* E-63	NOP
12	WPS0* E-54	WPS1* E-57	LPSL* E-64	NOP
13	WPS2* E-54	WPS3* E-57	LPSR* E-64	NOP
14	WPS0T E-54	WPS1T E-57	LPSLT E-64	NOP
15	WPS2T E-54	WPS3T E-57	LPSRT E-65	NOP
16	NOP	NOP	LPSLP E-65	NOP
17	NOP	NOP	LPSRP E-65	NOP

Formats for partial words (PSEVEN, PSODD, PS Fields) E-51

*This instruction uses a 16-bit integer VALUE (in bits 48-63 of the instruction word). The YW, FM, M1, M3, MI, MA, TMA, and DPA Fields are then disabled for this instruction word.

I/O Fields

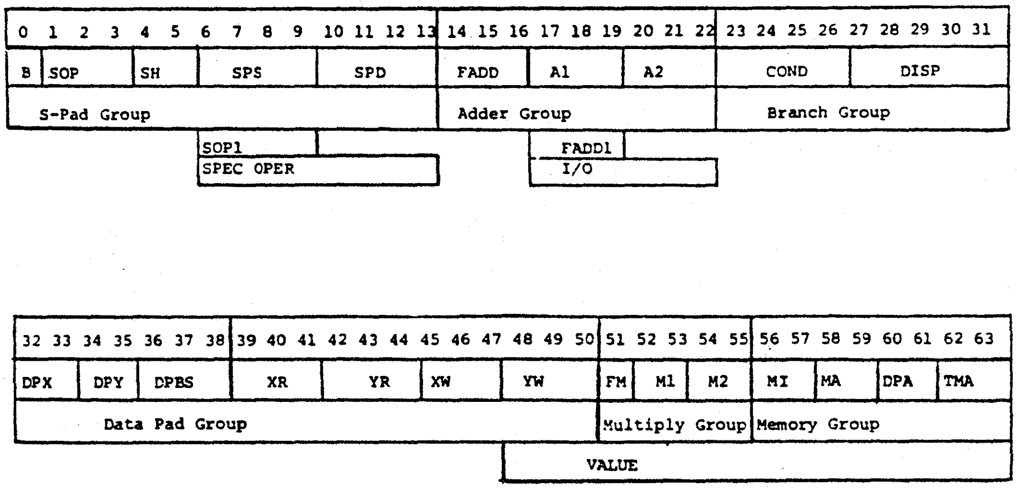
One of the I/O Fields may be used per instruction word. The Floating Adder Fields (FADD, FADD1, A1, and A2) are then disabled for this instruction word.

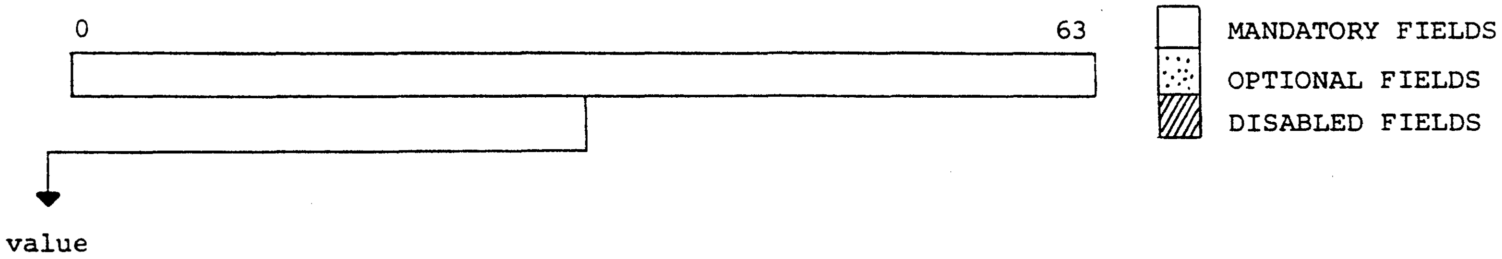
Field Name	I/O	LDREG	RDREG	INOUT
Octal Code				
0	LDREG E-101	NOP	RPSA E-105	OUT E-107
1	RDREG E-105	LDSPD E-101	RSPD E-105	SPNOUT E-109
2	SPMDAV E-7	LDMA E-101	RMA E-106	OUTDA E-110
3	REXIT E-7	LDTMA E-102	RTMA E-106	SPOTDA E-111
4	INOUT E-109	LDDPA E-102	RDPA E-107	IN E-112
5	SENSE E-115	LDSP E-103	RSPFN E-107	SPININ E-112
6	FLAG E-121	LDAPS E-103	RAPS E-108	INDA E-113
7	CONTROL E-8	LDDA E-103	RDA E-108	SPINDA E-114

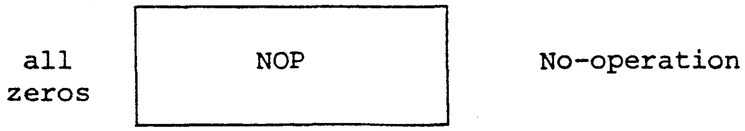
Field Name	SENSE	FLAG	CONTROL
Octal Code			
0	SNSA E-115	SFL0 . . . . . . E-121	HALT E-8
1	SPINA E-115	SFL1 . . . . . . E-121	IORST E-9
2	SNSADA E-115	SFL2 . . . . . . E-121	INTEN E-9
3	SPNADA E-115	SFL3 . . . . . . E-121	INTA E-10
4	SNSB E-118	CFL0 . . . . . . E-122	REFR E-10
5	SPINB E-118	CFL1 . . . . . . E-122	WRTEX E-11
6	SNSBDA E-119	CFL2 . . . . . . E-122	WRTMAN E-11
7	SPNBDA E-120	CFL3 . . . . . . E-122	NOP

NOP E-5

AP-120B Instruction Field Layout





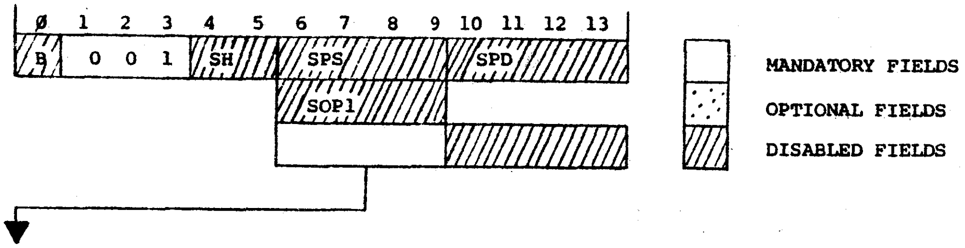


Assembler format: NOP

Effect: No operation is performed

Description: The assembler recognizes this mnemonic and will insert an all zeros instruction which is a NOP.

CONTROL (from SPEC)



VALUE

2 SPMDA SPIN WHILE MAIN DATA BUSY

Assembler
Format: SPMDA

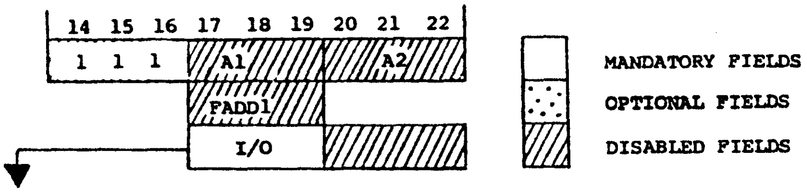
Effect: "SPIN" while MAIN DATA BUSY

Description: When specified, SPMDA causes the AP-120B to suspend program execution until MAIN DATA MEMORY (MD) completes its READ or WRITE cycle and becomes available for the next READ/WRITE operation. Using this op-code in an instruction immediately following one that initiates an MD READ operation, results in the data from that operation being available for use during the present instruction. It has no effect on a MD READ/WRITE operation in the same instruction.

```
    Thus:   LDMA; DB=100
            SPMDA; DPX(0)<MD
```

results in the contents of MD(100) being loaded into DPX(0).

CONTROL (from I/O)



VALUE

2 SPMDAV SPIN WHILE MAIN DATA BUSY

Assembler
Format: SPMDAV

Effect: "SPIN" while MAIN DATA BUSY

Description: When specified, SPMDA causes the AP-120B to suspend program execution until MAIN DATA MEMORY (MD) completes its READ or WRITE cycle and becomes available for the next READ/WRITE operation. Using this op-code in an instruction immediately following one that initiates an MD READ operation, results in the data from that operation being available for use during the present instruction. It has no effect on a MD READ/WRITE operation in the same instruction.

```
Thus:   LDMA; DB=100
        SPMDA; DPX(0)<MD
```

results in the contents of MD(100) being loaded into DPX(0).

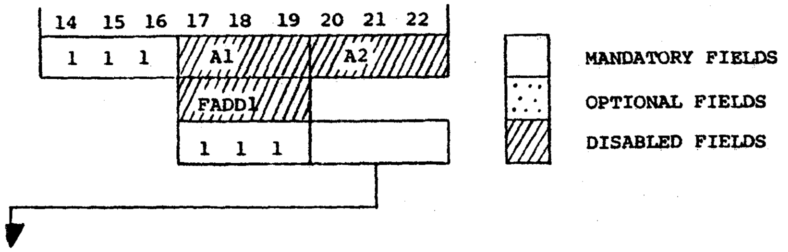
3 REXIT READ EXIT

Assembler
Format: REXIT

Effect: SRS(SRA)→PNLBS

Description: This instruction is similar to SETEXIT, but allows reading instead. The SRA is NOT changed by this instruction. This is generally used with a LDSPNL to load into S-PAD register.

CONTROL FIELD (CONTROL)



ALUE

Ø HALT

HALT AP-120

Assembler Format: HALT

example: INCMA; MI<FA; HALT

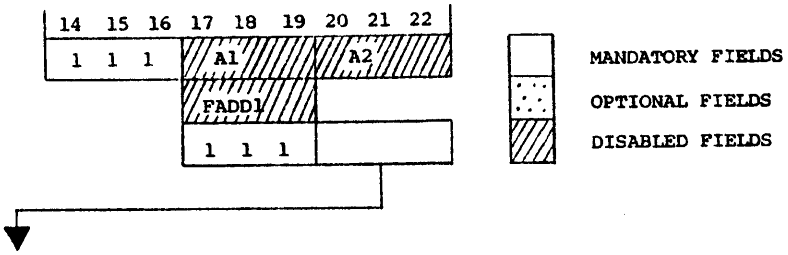
Effect: $1 \rightarrow FN_{bit\ Ø}$; clear RUN INDICATOR

Description: The AP-120B program execution will be halted after completion of the current instruction word. (See note.) AP-120B RUN INDICATOR (RUN) cleared and PANEL FUNCTION REGISTER(bit 0) set. When halted, PSA will point to the next instruction to be executed and it will have been entered into the instruction register. SPFN will reflect the operation in that instruction.

†NOTE: if the current instruction "SPINS" while waiting for I/O or MEMORY, HALT will not be effective until the "SPIN" cycle is finished and the instruction completed (as in the above example).

WARNING: Due to timing problems in the JSR instruction, it should not follow a HALT. Thus, the recommended programming practice is to place a NOP (no operation, all zeros instruction) after every HALT instruction.

CONTROL FIELD (CONTROL)



VALUE

1 IORST RESET I/O DEVICES IMMEDIATELY

Assembler
Format: IORST

Effect: Clear I/O device logic and timing and all four of the general FLAGS.

Description: Effects are device dependent. No effect on Host interface, TMRAM, or IOP16.

2 INTEN INTERRUPT ENABLE

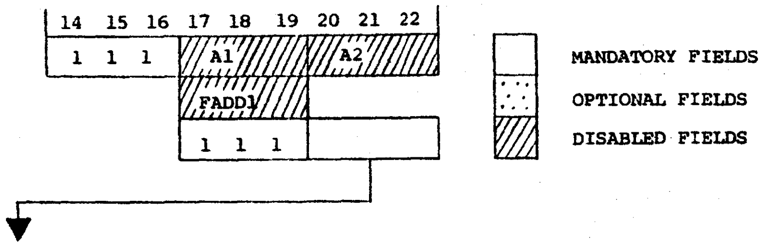
Assembler
Format: INTEN

Effect: If CTL$^{\text{Bit Ø5}}$ is already set to "1", generate interrupt to HOST-CPU. If not, no effect.

Description: This is used in conjunction with the CTL05 interrupt. (See I/O, PROGRAMMED INTERRUPTS.)

When an INTEN is executed, the AP-120B will attempt to set CTL05 interrupt. If CTL(Bit 05) is already set, then the AP-120B will generate an interrupt to HOST-CPU. The state of CTL(Bit 05) is not altered by this instruction.

CONTROL FIELD (CONTROL)



VALUE

3 INTA INTERRUPT ACKNOWLEDGE

Assembler Format: INTA

Effect: (DA interrupting IODEVICE) → INBS

Description: The interrupting I/O DEVICE enables its Device Address onto the INBS MANTISSA (Bits 20 to 27) for the current instruction cycle. Used to identify the interrupting I/O device after an interrupt is detected via the BINTRQ instruction.

4 REFR MEMORY REFRESH REQUEST

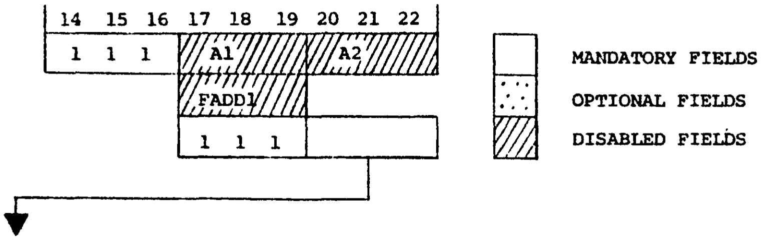
Assembler Format: REFR

Effect: REFRESH MD; reset REFR CTR

Description: REFR initiates a REFRESH cycle to MAIN DATA MEMORY (MD). The REFRESH COUNTER (REFR CTR) is reset to zero. This has the effect of synchronizing the REFRESH timing with a running AP-120B program. It can be used either to eliminate REFRESH interference with programmed memory accesses or simply to stabilize the REFRESH timing in order to facilitate hardware fault tracing with an oscilloscope. Floating Point Systems supplied APEX drivers always generate a REFR before starting the user called micro-code.

NOTE: NON-PROGRAMMED REFRESH cycles occur every 31usec for 8K bank-pair MD memory and every 15.5usec for 32K bank-pair MD memory.

CONTROL FIELD (CONTROL)



VALUE

5 WRTEX WRITE EXPONENT ONLY

Assembler
Format: WRTEX

example: DPX (2)< TM; WRTEX

Effect: Restricts DPX, DPY, or MI fields to write exponent bits only.

Description: When specified with a concurrent WRITE DPX, DPY, or MI operation, WRTEX restricts the writing to EXPONENT(Bits 02-11) only. (See S-PAD group, WRTEXP for further description of effect.)

6 WRTMAN WRITE MANTISSA ONLY

Assembler
Format: WRTMAN

Example: MOV 5,5; SETMA; MI<FA; WRTMAN

Effect: Restricts DPX, DPY, or MI fields to WRITE MANTISSA bits only.

Description: When specified with a concurrent WRITE DPX, DPY or MI operation, WRTMAN restricts the writing to MANTISSA(Bits 00-27) only. (See S-PAD group for further discussion.)

CONTROL FIELD (CONTROL)
continued...

7 IOINTA I/O INTERRUPT ACKNOWLEDGE

Assembler
Format: IOINTA

Effect: (DA interrupting IODEVICE) INBUS

Description: The interruptiong I/O Device enables its Device Address onto the INBS Mantissa (Bits 20 to 27) for the current instruction cycle.

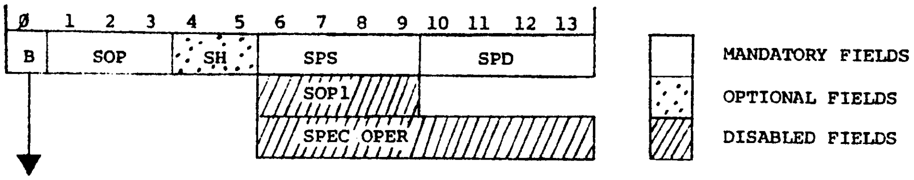
S-PAD TIMING RULES

1. SPFN for an instruction with an S-PAD operation is the result of that operation.

2. SPFN is stored back into SP(SPD) only once - at the end of the instruction in which the S-PAD operation took place (not stored at all if No-Load specified). Similarly, the N, Z, and C S-PAD condition bits are set only once for each S-PAD operation.

3. SPFN for an instruction without an S-PAD operation is the result of performing the last previous S-PAD instruction over again, using the current value of SP(SPD) as possibly modified by the original S-PAD operation. SP(SPD) is not altered if no S-PAD operation is specified. This modified SPFN value would be apparent if an SPFN utilizing instruction were executed (e.g., RSPFN, SETMA, etc.).

BIT REVERSE FIELD



VALUE

Ø No Operation

1 $\boxed{\&}$ BIT REVERSE the contents of S-PAD SOURCE REGISTER before using

Assembler Format: < & > (Brackets indicate optional use with S-PAD operations.)

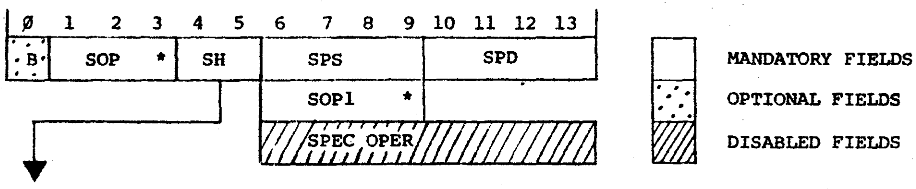
Effect: Example: ADD & 6,5

Effect: BIT-REVERSE(SP_{SPS})→ SOURCE INPUT FOR CURRENT S-PAD OPERATION

Description: The contents of the S-PAD SOURCE REGISTER (SP[SPS]) are BIT-REVERSED and shifted before being used as the SOURCE OPERAND in the current S-PAD operation.

The number of shifts performed depends on the size of the complex data array being processed. The programmer must load the applicable shift value into the BIT-REVERSE field of the APSTATUS Register before specifying the BIT-REVERSE operation. (See S-PAD SUMMARY BIT-REVERSE FIELD for more details.) (See also APSTATUS SUMMARY.)

SHIFT FIELD



* MAY BE USED WITH EITHER
SOP OR SOP1 FIELDS

VALUE

Ø No Operation

1 L LEFT SHIFT S-PAD OUTPUT (SPFN) ONCE. ZERO FILL.

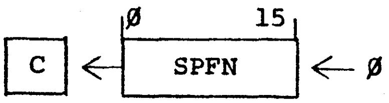
Assembler
Format: <L> (Brackets indicate optional use with S-PAD operations)

Example: SUBL 5,6

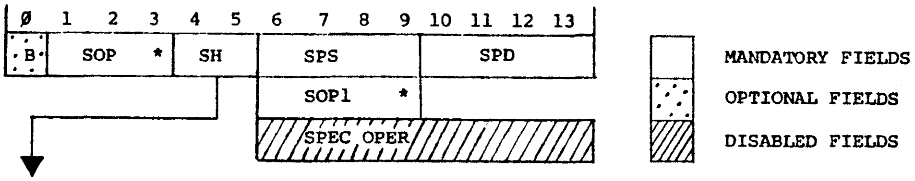
Effect: SPFN→ LEFT SHIFTED ONCE→ SPFN

Description: The S-PAD RESULT (SPFN) is logically shifted left one place. The right-most bit is set to zero. The bit shifted off the left end is stored in the S-PAD CARRY BIT, (C) - overriding any carry that resulted from the specified arithmetic operation.

Excepting possible OVERFLOW, the shift has the effect of a multiplication by two. The carry bit (C), bit 7 of the AP INTERNAL STATUS REGISTER (APSTATUS), may be tested during the next instruction cycle.



SHIFT FIELD (SH)



* MAY BE USED WITH EITHER SOP OR SOP1 FIELDS

VALUE

2 R RIGHT SHIFT S-PAD FUNCTION (SPFN) ONCE. ZERO FILL.

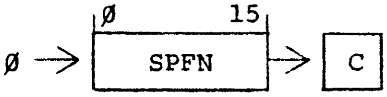
Assembler Format: <R> (Brackets indicate optional use with S-PAD operations)

Example: SUBR 5,6

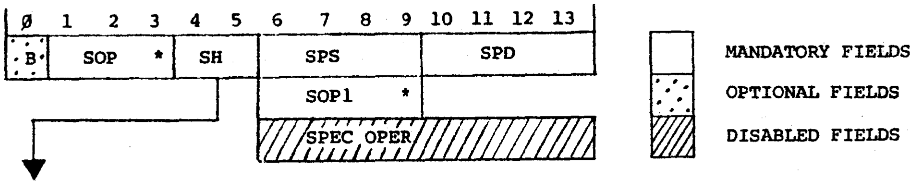
Effect: SPFN right-shifted once→ SPFN

Description: The S-PAD RESULT (SPFN) is logically shifted right one place. A zero is shifted into the left-most bit. The bit shifted off the right end is set into the S-PAD CARRY BIT.

The instruction has the effect for unsigned numbers, of a division by two. Bit C of the AP INTERNAL STATUS REGISTER (bit 7, APSTATUS) reflects the condition of S-PAD CARRY and may be tested during the next instruction cycle.



SHIFT FIELD



* MAY BE USED WITH EITHER SOP OR SOP1 FIELDS

VALUE

3 RR — RIGHT SHIFT S-PAD FUNCTION (SPFN) twice. Zero fill.

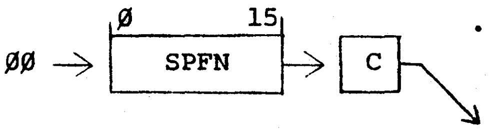
Assembler Format: <RR> (Brackets indicate optional use with S-PAD operations)

Example: SUBRR 5,6

Effect: SPFN → right shifted twice → SPFN

Description: The contents of the S-PAD ALU RESULT are logically shifted right two times before being enabled onto the SPFN data path. Zeros are filled into the left-most two bits. The second bit shifted off the end is set into the S-PAD ALU CARRY BIT.

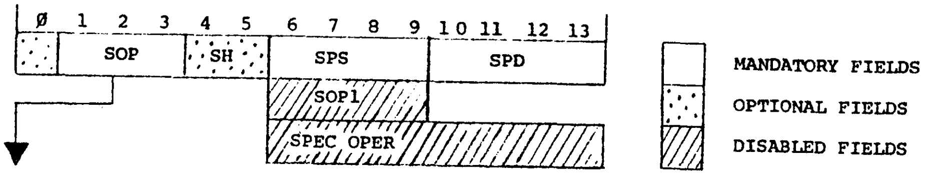
The instruction has the effect for unsigned numbers of a division by four. Bit C (bit 7) of the AP INTERNAL STATUS REGISTER (APSTATUS) reflects the condition of the S-PAD CARRY BIT and may be tested during the next instruction cycle.



- REFLECTS THE CONTENTS OF THE SECOND BIT SHIFTED "OFF THE END"

FIRST BIT SHIFTED "OFF THE END" IS LOST

S-PAD OPERATIONS FIELD



VALUE

Ø See, S-PAD OPERATIONS 1

1 See, SPECIAL OPERATIONS

2 ADD ADD S-PAD SOURCE REGISTER AND S-PAD DESTINATION REGISTER

Assembler Format: ADD<#> <&> SPS,SPD

Effect: $(SP_{SPS})^{\dagger}$ plus $(SP_{SPD}) \rightarrow SPFN^{\dagger\dagger}$;

$SPFN \rightarrow (SP_{SPD})$ unless S-PAD NO-LOAD(#) is specified

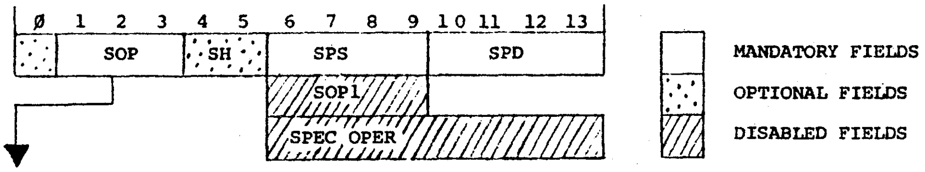
Description: The contents of S-PAD SOURCE REGISTER (SP_{SPS}) are added with the contents of S-PAD DESTINATION REGISTER (SP_{SPD}). The result of the operation, (SPFN) is stored back into the specified S-PAD DESTINATION REGISTER unless an S-PAD NO-LOAD (#) is specified.

Appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If $(SP_{SPD})+(SP_{SPS}) \geq 2^{16}$ then carry=1

† *(SP_{SPS}) may be optionally BIT-REVERSED, (see BIT-REVERSE FIELD)*

†† *SPFN from the ADD may be optionally shifted,(see SHIFT FIELD)*



VALUE

3 SUB

SUBtract S-PAD SOURCE REGISTER from S-PAD DESTINATION REGISTER

Assembler Format: SUB< sh >< # > < & > sps,spd

Effect: (SP_{SPD}) minus $(SP_{SPS})^{\dagger} \rightarrow SPFN^{\dagger\dagger}$;

SPFN→ (SP_{SPD}) unless S-PAD NO-LOAD (#) is specified

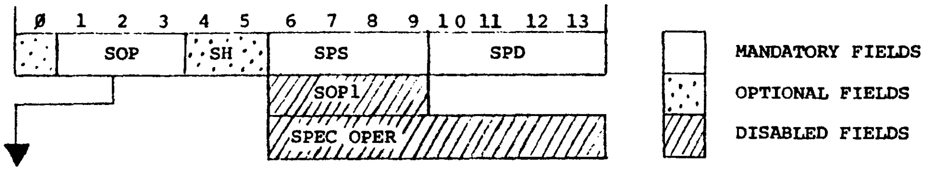
Description: The contents of the S-PAD SOURCE REGISTER are subtracted from the contents of the S-PAD DESTINATION REGISTER. The result of the operation is stored back into the S-PAD DESTINATION REGISTER unless a S-PAD NO-LOAD (#) is specified.

Appropriate bits (N,Z,C) are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If $(SP[SPD])+(\overline{SP[SPS]})+1 \geq 2^{16}$ then C=1 else 0. If a shift is specified, then C is set to the carry from that shift.

[†] (SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.
[††] SPFN from the SUB may be optionally shifted. See SHIFT FIELD.

S-PAD OPERATIONS FIELD



VALUE

4 MOV

MOVE S-PAD SOURCE REGISTER TO S-PAD DESTINATION REGISTER

Assembler Format: MOV < sh > < # > < & > sps,spd

Effect: $(SP_{SPS})^{\dagger} \rightarrow SPFN^{\dagger\dagger}$; SPFN→ (SP_{SPD}) unless S-PAD NO-LOAD is specified.

Description: SPFN is set to the contents of the S-PAD SOURCE REGISTER (SP[SPS]); SPFN is stored into the S-PAD DESTINATION REGISTER unless an S-PAD NO-LOAD (#) is specified.

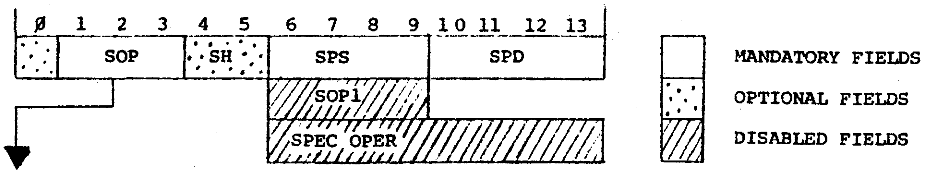
Appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If [(SP[SPD]) AND (SP[SPS])] + [(SP[SPD]) OR (SP[SPS])] ≥ 2^{16}

then, C=1 else 0

† (SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.
†† SPFN from the MOV may be optionally shifted. See SHIFT FIELD.

S-PAD OPERATIONS FIELD



VALUE

5 AND

AND S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler Format: AND < sh > <#> < & > sps,spd

Effect: $(SP_{SPS})^{\dagger}$ AND $(SP_{SPD}) \rightarrow SPFN^{\dagger\dagger}$;

SPFN → (SP_{SPD}) unless S-PAD NO-LOAD is specified.

Description: The contents of the S-PAD SOURCE REGISTER (SP_{SPS}) are logically ANDed with the contents of the S-PAD DESTINATION REGISTER (SP_{SPD}). A bit by bit comparison is made between the contents of the two operands and if both respective bits are "1", a "1" is recorded into the corresponding bit of the result (SPFN). All other combinations result in "Ø" being recorded into the respective bit of SPFN. The result of the operation (SPFN) is stored into SP_{SPD} unless an S-PAD NO-LOAD (#) is specified.

The appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If $[(SP_{SPD}) \text{ AND } (\overline{SP_{SPS}})] + (SP_{SPD}) \geq 2^{16}$ then CARRY=1

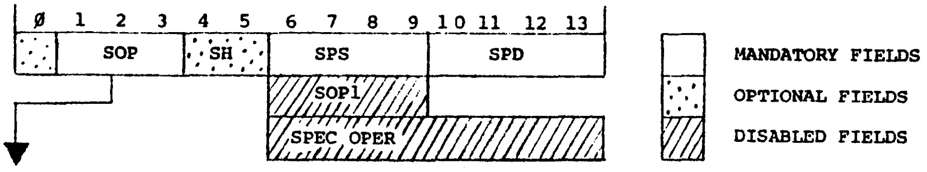
TRUTH TABLE

SP_{SPS}	SP_{SPD}		SPFN
▼	▼		▲
Ø	Ø	▶	Ø
Ø	1	▶	Ø
1	Ø	▶	Ø
1	1	▶	1 ▼

[†] *(SP_{SPS}) may be optionally BIT-REVERSED, See BIT-REVERSE FIELD*

[††] *SPFN from the AND may be optionally shifted, See SHIFT FIELD*

S-PAD OPERATIONS FIELD



VALUE

6 **OR** OR S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler Format: OR < sh > < # > < & > sps,spd

Effect: $(SP_{SPS})^{\dagger}$ OR $(SP_{SPD}) \rightarrow$ SPFN;††

SPFN → (SP_{SPD}) unless NO-LOAD is specified.

Description: The contents of the S-PAD SOURCE REGISTER (SP[SPS]) are logically ORed with the contents of the S-PAD DESTINATION REGISTER (SP[SPD]). A bit-by-bit comparison is made between the contents of the two operands and if either one of the respective bits = "1," then a "1" is recorded in the corresponding bit of the result (SPFN). All other combinations result in a "0" being recorded into the respective SPFN bit position. The result of the operation (SPFN) is stored into SP(SPD) unless S-PAD NO-LOAD (#) is specified.

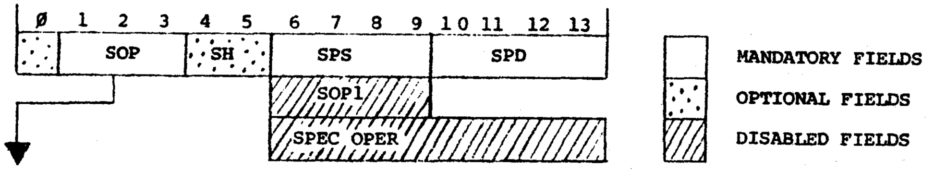
Additionally, S-PAD ALU CARRY BIT is set to "0." The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set and may be tested during the next instruction cycle.

TRUTH TABLE

SP_{SPS} ▼	SP_{SPD} ▼		SPFN ▲
Ø	Ø	▶	Ø
Ø	1	▶	1
1	Ø	▶	1
1	1	▶	1

†(SP[SPS]) may be optionally BIT-REVERSEd. See BIT-REVERSE FIELD.
††SPFN from the OR may be optionally shifted. See SHIFT FIELD.

S-PAD OPERATIONS FIELD



VALUE

7 | EQV | EQUIVALENCE S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler Format: EQV < sh >< # > < & > sps,spd

Effect: $(SP_{SPS})^{\dagger}\ \overline{XOR}\ (SP_{SPD}) \rightarrow SPFN^{\dagger\dagger}$;

$SPFN \rightarrow (SP_{SPD})$ unless NO-LOAD is specified.

Description: The contents of the S-PAD SOURCE REGISTER (SP[SPS]) and the S-PAD DESTINATION REGISTER (SP[SPD]) are compared on a corresponding bit position basis for equal value. If the corresponding bits both equal "0," or both equal "1," then the respective bit of the result (SPFN) is set to "1." All other combinations result in a "0" being set into the corresponding bit of SPFN. The result of the operation (SPFN) is then written into (SP[SPD]) unless S-PAD NO-LOAD (#) is specified.

The appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

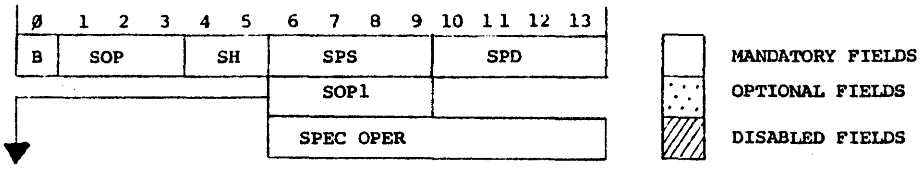
CARRY BIT EQUATION: If $(SP[SPD])+(SP[SPS]) \geq 2^{16}$ then CARRY=1

TRUTH TABLE

SP_{SPS} ▼	SP_{SPD} ▼		SPFN ▲
Ø	Ø	▶	1
Ø	1	▶	Ø
1	Ø	▶	Ø
1	1	▶	1

[†] (SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.
[††] SPFN from the EQV may be optionally shifted. See SHIFT FIELD.

S-PAD OPERATION 1 FIELD



VALUE

1 **WRTEXP** RESTRICT WRITE TO EXPonent only into DPX, DPY or MI

Assembler Format: WRTEXP

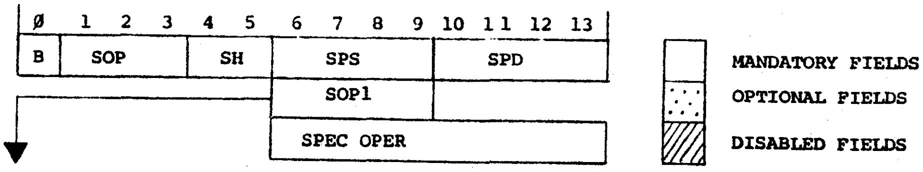
Example: DPX (-2) FA; WRTEXP

Effect: Restricts DPX,DPY or MI field to write EXPONENT bits only.

Description: WRTEXP restricts writing of the pertinent MEMORY INPUT REGISTER into EXPONENT bits 02-11 only. WRTEXP used in conjunction with a DPX, DPY or MI WRITE operation.

When used in conjunction with a WRITE DPX or WRITE DPY operation, this operation has the effect of concatenating a portion of the input data with the value most recently written into DPX or DPY irrespective of XW or YW. Thus, if the last WRITE into DPX placed a floating point 1.0 into DPX(-2) and in this instruction we WRITE DPX(0) in conjunction with the WRTEXP Op-Code, the net effect is to concatenate the EXPONENT portion of the current input with the MANTISSA from the 1.0 of the preceding DPX WRITE operation and place the result in DPX(0). WRTHMN, WRTLMN act in a similar fashion with the exception that they use different portions of the input argument. WRTEX and WRTMAN from the I/O group also work in a similar manner.

S-PAD OPERATIONS 1 FIELD



VALUE

2 **WRTHMN** RESTRICT WRITE TO HIGH MANTISSA only into DPX, DPY or MI

Assembler Format: WRTHMN

Example: WRTHMN ; DPY<FM

Effect: Restricts DPX, DPY or MI fields to WRITE HIGH MANTISSA bits only (MANTISSA$^{\text{bits ØØ-11}}$).

Description: WRTHMN restricts the writing to the HIGH MANTISSA only, (MANTISSA$^{\text{BITS ØØ-11}}$) of the pertinent MEMORY INPUT REGISTER. WRTHMN is used in conjunction with a DPX, DPY or MI WRITE operation. (See example above).

NOTE: See WRTEXP for a description of the effect of this operation on DPX or DPY.

3 **WRTLMN** RESTRICT WRITE to LOW MANTISSA only into DPX, DPY or MI FIELDS

Assembler Format: WRTLMN

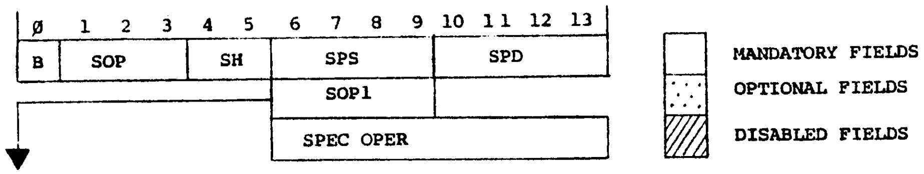
Example: WRTLMN ; SETMA; MI<MD

Effect: Restricts DPX, DPY or MI fields to WRITE LOW MANTISSA only (MANTISSA$^{\text{bits 12-27}}$).

Description: WRTLMN restricts writing to the LOW MANTISSA only (MANTISSA[bits 12-27]) of the pertinent MEMORY INPUT REGISTER. WRTLMN is used in conjunction with a DPX, DPY or MI WRITE operation. (See example above.)

NOTE: See WRTEXP for a description of the effect of this operation on DPX or DPY.

S-PAD OPERATIONS 1 FIELD



VALUE

4 through 7 No Operation

10 CLR CLEAR S-PAD DESTINATION REGISTER

Assembler Format: CLR < sh><#> spd

Effect: Ø→ SPFN; Ø→ SP_{SPD} unless NO-LOAD (#) is specified.

Description: The S-PAD OUTPUT (SPFN) is forced to all zeros and bit "Z" of the AP INTERNAL STATUS REGISTER is set to "1" (bit 5, APSTATUS).

SP(SPD) is cleared unless S-PAD NO-LOAD (#) is specified.

CARRY BIT EQUATION: If SP(SPD) is negative then CARRY=1.

11 INC INCREMENT S-PAD DESTINATION REGISTER

Assembler Format: INC< sh >< # >spd

Effect: (SP_{SPD}) + 1 → SPFN; and, unless NO-LOAD is specified,

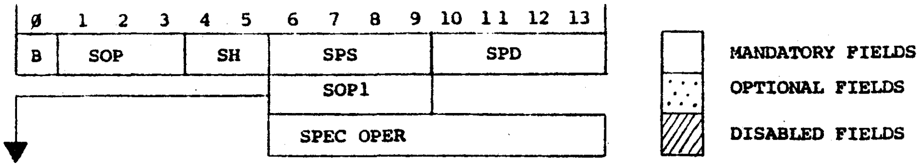
SPFN → (SP_{SPD}).

Description: The contents of the S-PAD DESTINATION REGISTER (SP[SPD]), plus ONE are enabled onto the S-PAD FUNCTION (SPFN). SPFN is stored into the S-PAD DESTINATION REGISTER unless S-PAD NO-LOAD is specified.

The appropriate bits of the APSTATUS Register are set and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If (SP[SPD]) was -1, then CARRY=1 else 0.

S-PAD OPERATIONS 1 FIELD



VALUE

12 DEC

DECREMENT S-PAD DESTINATION REGISTER

Assembler Format: DEC < sh> < #> spd

Effect: $(SP_{SPD})-1 \rightarrow SPFN$; and, unless NO-LOAD is specified,

$(SPFN) \rightarrow SP_{SPD}$

Description: The contents of the S-PAD DESTINATION REGISTER minus ONE is set to the S-PAD Function (SPFN). The result (SPFN) is stored into the S-PAD DESTINATION REGISTER unless S-PAD NO-LOAD (#) is specified.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) will be set and may be tested during the next instruction cycle.

CARRY BIT EQUATION: Unless SP_{SPD} was Ø, CARRY=1.

13 COM

COMPLEMENT S-PAD DESTINATION REGISTER

Assembler Format: COM < sh >< # > spd

Effect: $\overline{(SP_{SPD})} \rightarrow SPFN$; unless NO-LOAD is specified,

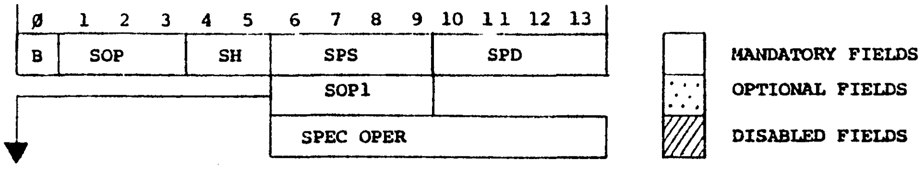
$SPFN \rightarrow (SP_{SPD})$

Description: The ONE's COMPLEMENT of the contents of S-PAD DESTINATION REGISTER are enabled onto the S-PAD Function. The result (SPFN) is stored into SP_{SPD} unless S-PAD NO-LOAD (#) is specified in the instruction.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) will be set and may be tested during the next instruction cycle.

CARRY BIT EQUATION: Unless SP_{SPD} was Ø, CARRY=1.

S-PAD OPERATIONS 1 FIELD



VALUE

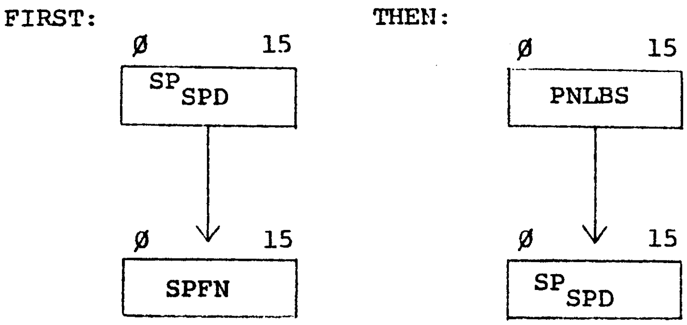
14 LDSPNL LOAD S-PAD DESTINATION REGISTER from the PANEL BUS

Assembler
Format: LDSPNL spd

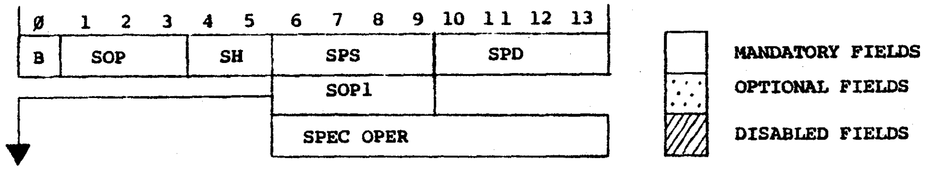
Effect: $(SP_{SPD}) \rightarrow SPFN; PNLBS \rightarrow SP_{SPD}$

Description: First, the S-PAD Function is set to the old contents of the S-PAD DESTINATION REGISTER. Then, whatever is enabled onto the PANEL BUS is loaded into the S-PAD DESTINATION REGISTER.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP_{SPD} and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of SP_{SPD} as loaded by this instruction cycle



S-PAD OPERATIONS 1 FIELD



VALUE

15 **LDSPE** Load S-PAD DESTINATION REGISTER from DATA PAD BUS - EXPONENT

Assembler Format: LDSPE spd[†]

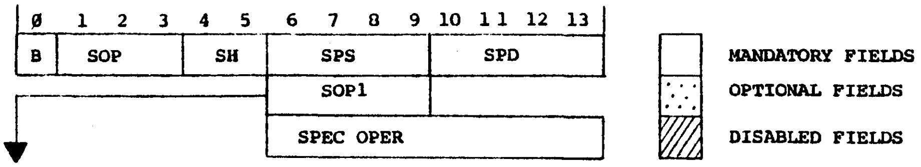
Effect: $(SP_{SPD}) \rightarrow$ SPFN; then, $(DPBS^{EXP}) -512 \rightarrow SP_{SPD}$

Description: First, the SPFN is set to the old contents of the S-PAD DESTINATION REGISTER. Then the EXPONENT portion of the DATA PAD BUS (bits 02-11), BIAS inverted, is loaded into the S-PAD DESTINATION REGISTER (bits 06-15). The inverted EXPONENT BIAS BIT is extended into the remaining portion of SP(SPD) (bits 00-05).

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP(SPD) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of SP(SPD) as loaded by this instruction.

[†] This transformation converts a BIASED EXPONENT from a Floating Point word into its TWO's COMPLEMENT equivalent.

S-PAD OPERATIONS 1 FIELD



VALUE

16 LDSPI

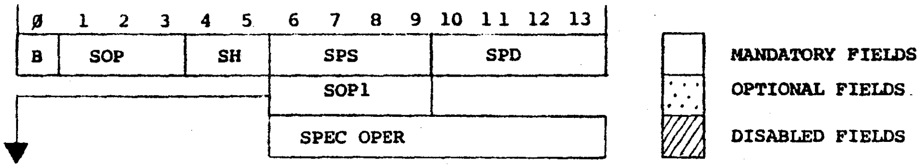
Load S-PAD DESTINATION REGISTER from DATA PAD BUS - INTEGER

Assembler Format: LDSPI spd

Effect: $(SP_{SPD}) \rightarrow$ SPFN, then $(DPBS_{LOW\ MANTISSA}) \rightarrow SP_{SPD}$

Description: First, SPFN is set to the old contents of the S-PAD DESTINATION REGISTER. Then the contents of the DATA PAD BUS - LOW MANTISSA, a 16-bit integer, are loaded into the S-PAD DESTINATION REGISTER.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP_{SPD} and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, the SPFN for that instruction will be the new contents of SP_{SPD} as loaded by this instruction.



VALUE

17 **LDSPT** Load S-PAD DESTINATION REGISTER from DATA PAD BUS - TABLE LOOK UP BITS

Assembler Format: LDSPT spd

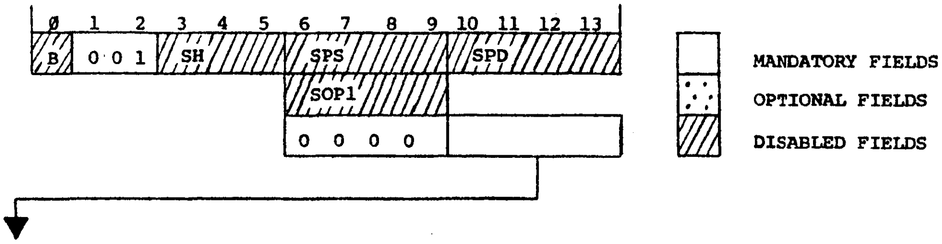
Effect: $(SP_{SPD}) \rightarrow$ SPFN; then $(\text{DPBS MANTISSA}^{\text{Bits } \emptyset 2-\emptyset 8}) \rightarrow SP_{SPD}$

Description: First, the SPFN is set to the old contents of S-PAD DESTINATION REGISTER. Then the DATA PAD BUS - TABLE LOOK UP bits (MANTISSA [bits 02-08]) are loaded into bits 09-15 of the S-PAD DESTINATION REGISTER. SP(SPD) bits 00-08 are cleared to zero.

LDSPT may be used to calculate memory addresses for use with a look up table. It extracts the seven most significant unknown bits from a positive, normalized, non-zero Floating Point number.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP(SPD) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of SP(SPD) as loaded by this instruction.

SPECIAL TEST FIELD (STEST)



VALUE

Ø **BFLT** BRANCH on FLOATING ADDER LESS THAN ZERO

Assembler Format: BFLT targ

Effect: If FA < 0; then (PSA) + (DISP† - BIAS) → PSA (where BIAS = 20_8).

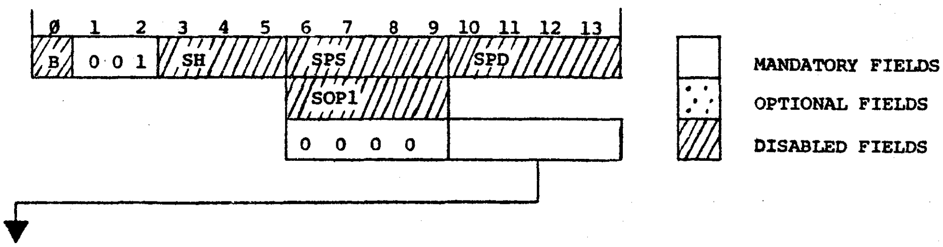
Description: CONDITIONAL RELATIVE BRANCH. BFLT will cause a program branch if the FADDR Result (FA) available during the previous instruction was less than zero.

This instruction tests the FA-NEGATIVE Bit (FN) of the APSTATUS Register. If FN is equal to "1" (indicating that FA was negative during the previous instruction) a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with Biased contents of the DISP field of the current instruction word. If FN is equal to "0," this instruction will have no effect.

The BRANCH TARGET ADDRESS must lie within -20(8) to +17(8) locations relative to the current PROGRAM SOURCE ADDRESS.

DISP=Instruction Word (BITS 27-31) is computed as follows: DISP = targ - (PSA) + BIAS. Note that if FN was altered via a LDAPS instruction, at least one cycle must intervene before testing it with this instruction. This restriction applies to all BRANCH instructions that test conditions appearing in APSTATUS.

SPECIAL TEST FIELD (STEST)



VALUE

1 **BLT** BRANCH if S-PAD FUNCTION is LESS THAN ZERO

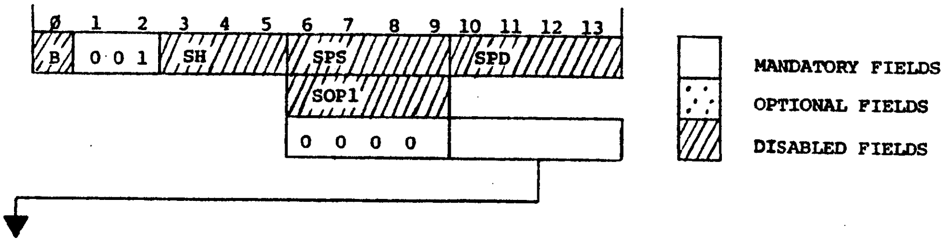
Assembler Format: BLT targ

Effect: If SPFN < 0 ; then (PSA) + (DISP† - BIAS) → PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. BLT will cause a program branch if the result of the last S-PAD operation (SPFN) was less than zero.

This instruction tests the condition of the SPFN-NEGATIVE Bit (N) of the APSTATUS Register. If "N" is equal to "1" (indicating that SPFN of the last previous instruction was negative), a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word. If "N" is equal to "0," this instruction will have no effect.

SPECIAL TEST FIELD (STEST)



VALUE

2 BNC BRANCH if S-PAD CARRY is equal to "1"

Assembler Format: BNC targ

Effect: If S-PAD CARRY = 1; then (PSA) + (DISP[†] - BIAS) → PSA (Where BIAS = 20_8).

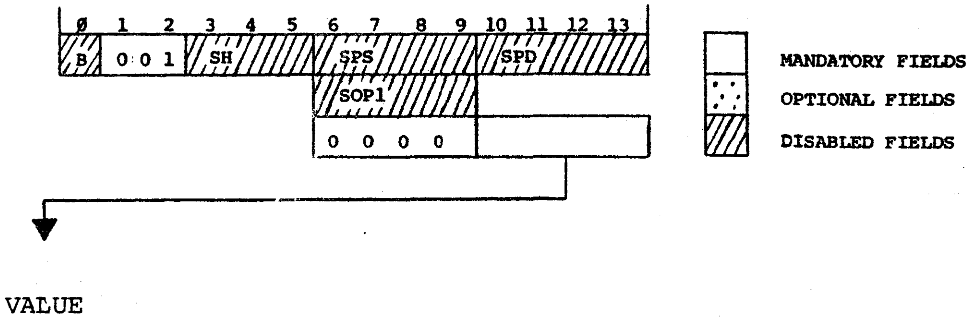
Description: CONDITIONAL RELATIVE BRANCH. BNC will cause a program branch if the S-PAD CARRY Bit (C) of the APSTATUS Register is equal to "1."

Bit "C" will be equal to "1" if either:

*the S-PAD CARRY Bit was set to "1" as a result of the last S-PAD operation and no S-PAD SHIFT was specified, or

*a shift occurred during the last S-PAD operation and the last bit shifted "off the end" of the S-PAD Result was equal to "1."

SPECIAL TEST FIELD (STEST)



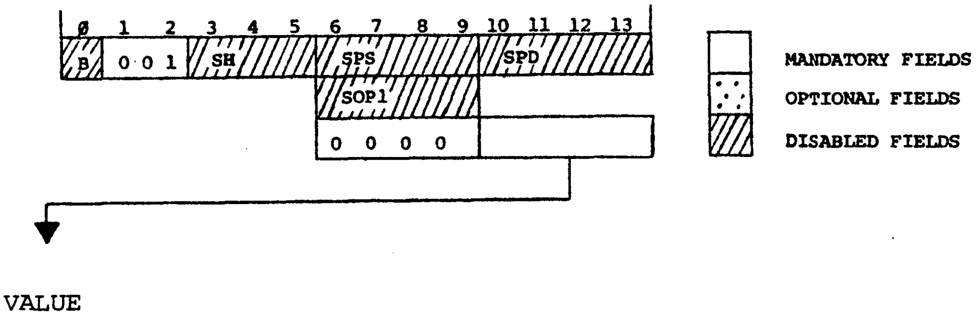
3 BZC BRANCH on S-PAD CARRY equal to ZERO

Assembler Format: BZC targ

Effect: If S-PAD CARRY = Ø; then (PSA) + (DISP[†] - BIAS) → PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. BZC will cause a program branch if the S-PAD CARRY Bit (C) of the APSTATUS Register is equal to zero.

SPECIAL TEST FIELD (STEST)



4 **BDBN** BRANCH if DATA PAD BUS is NEGATIVE

Assembler Format: BDBN targ

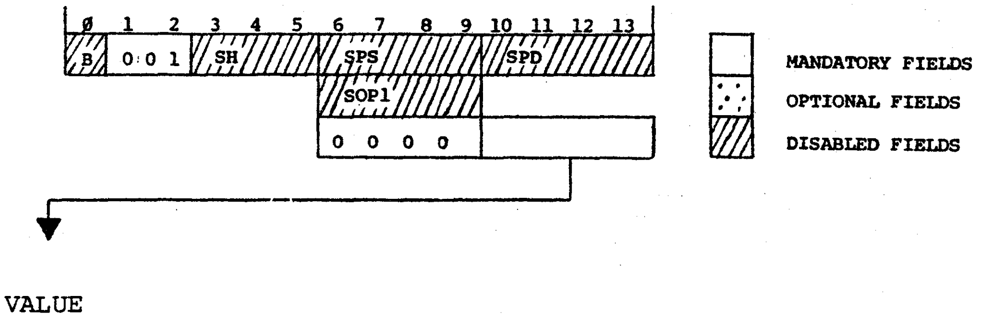
Effect: If (DB) <0.0; then (PSA) + (DISP[†] - BIAS) → PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. The sign of the DATA PAD BUS (MANTISSA) (DB[MANT]bit 00) is tested as to its state during the preceding instruction. If DB(MANT)Bit 00 was negative, (e.g.,=1), a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word. If DB(MANT)Bit 00 was "0," this instruction will have no effect.

NOTE: Since any data enabled onto DB is not latched, the programmer must re-enable the particular data onto DB one instruction cycle before attempting to test it with this instruction.

Note that instructions in the PS field (RPSF, LPSL, etc.) cannot be used to enable data onto Data Pad Bus for testing with this instruction.

SPECIAL TEST FIELD (STEST)



5 BDBZ BRANCH if DATA PAD BUS is POSITIVE and UNNORMALIZED

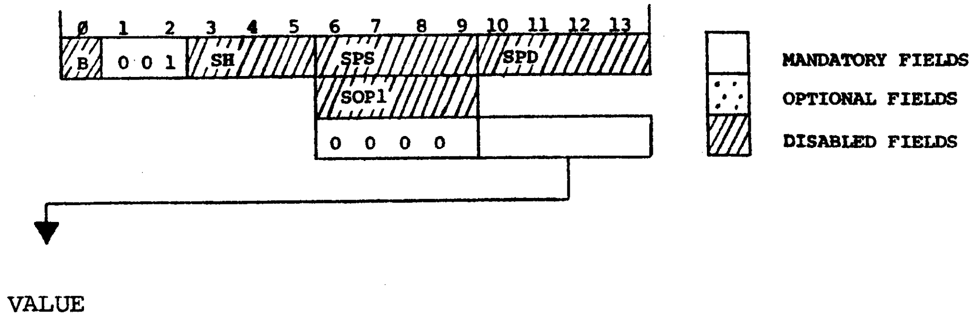
Assembler
Format: BDBZ targ

Effect: If DB^{MANT} Bits ØØ,Ø1 = "Ø", then (PSA) + (DISP[†] - BIAS) → PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. BDBZ will cause a program branch to occur to the BRANCH TARGET ADDRESS (targ), formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word, if the sign of the DATA PAD BUS(MANTISSA)(DB[MANT]Bit 00), enabled during the preceding instruction, was positive (e.g.,=0) and DB(MANT)Bit 01 was also equal to "0," (indicating an UNNORMALIZED MANTISSA). If either or both Bits equal(s) "1," this instruction will have no effect.

NOTE: Since any data enabled onto DB is not latched, the programmer must re-enable the particular data onto DB one cycle before attempting to test it by this instruction.

SPECIAL TEST FIELD (STEST)



6 **BIFN** BRANCH if INVERSE FFT FLAG = 1

Assembler Format: BIFN targ

Effect: If IFFT (APSTATUS $^{\text{Bit 11}}$) = 1; then (PSA) + (DISP† - BIAS)→PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. BIFN will cause a program branch if the Inverse FFT Flag (IFFT) of the APSTATUS Register is set to "1."

IFFT is APSTATUS(bit 11) and can be set by an LDAPS instruction. (See LDREG, I/O.) It is normally set to "1," along with APSTATUS(bit 12) (FFT), only during an INVERSE FAST FOURIER TRANSFORM.

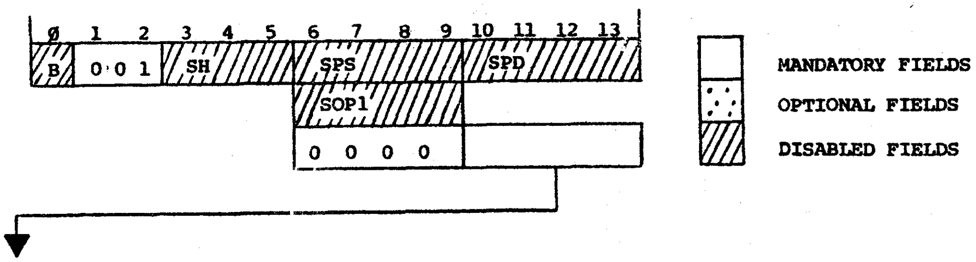
7 **BIFZ** BRANCH if IFFT FLAG = 0

Assembler Format: BIFZ targ

Effect: If IFFT (APSTATUS$^{\text{Bit 12}}$) = Ø; then (PSA) = (DISP† - BIAS)→PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH. BIFZ will cause a program branch if the INVERSE FFT Flag (IFFT) of the APSTATUS Register is cleared to zero.

SPECIAL TEST FIELD (STEST)



'ALUE

10 through 13		No Operation
14	BFLØ	BRANCH if GENERAL FLAG #Ø = 1
15	BFL1	BRANCH if BENERAL FLAG #1 = 1
16	BFL2	BRANCH if GENERAL FLAG #2 = 1
17	BFL3	BRANCH if GENERAL FLAG #3 = 1

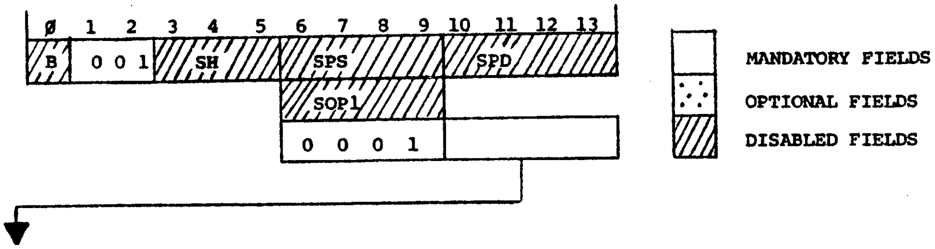
Assembler Format: BFLn targ

Effect: If FLAG n=1; then (PSA) + (DISP† - BIAS) → PSA (Where BIAS = 20_8).

Description: A branch will occur to program location "targ" (assembler format) if flag "n" is set to "1." Flag "n" must have been set or cleared two cycles before the current instruction cycle in order to be tested, i.e., at least one cycle must intervene between a set or clear flag instruction and a branch flag instruction.

Note: The CONDITIONAL RELATIVE BRANCH instructions test the condition of either of four GENERAL FLAGS (0,1,2,3) available for use in the AP. These flags may be "set" or "cleared" by software instructions. (See FLAG, I/ O.)

HOST/PANEL FIELD (HOSTPNL)



VALUE

Ø PNLLIT TRANSFER PANEL BUS to the LITES REGISTER

Assembler Format: PNLLIT

Effect: (PNLBS) → LITES

Description: The current data enabled onto the 16-bit PANEL BUS are loaded into the 16-bit LITES REGISTER.

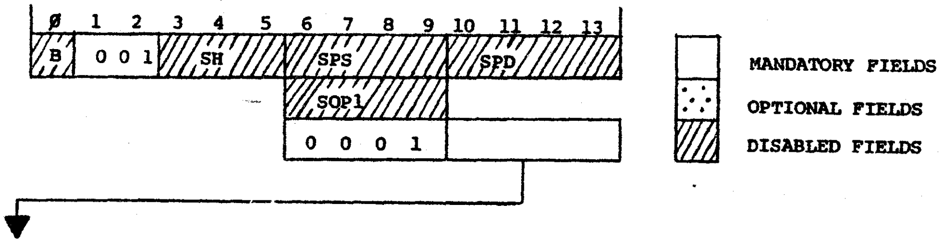
1 DBELIT TRANSFER DATA PAD BUS^{EXPONENT} to the LITES REGISTER, via PANEL BUS

Assembler Format: DBELIT

Effect: (DB^{EXP}) PNLBS → LITES; right justified.

Description: The current data enabled onto the 10-bit DATA PAD BUS^{EXPONENT} are loaded into the 16-bit LITES REGISTER - right justified. The transfer is via the PANEL BUS.

HOST/PANEL FIELD (HOSTPNL)



VALUE

2 **DBHLIT** TRANSFER DATA PAD BUS - HIGH MANTISSA to the LITES REGISTER, via PANEL BUS

Assembler Format: DBHLIT

Effect: (DB^{HMANT}) → PNLBS → LITES; right justified

Description: The current data enabled onto the $\text{DATA PAD BUS}^{\text{HIGH MANTISSA}}$ $(\text{MANTISSA}^{\text{Bits ØØ-11}})$ are loaded into the 12 right-most bits of the LITES REGISTER. The transfer is via PANEL BUS.

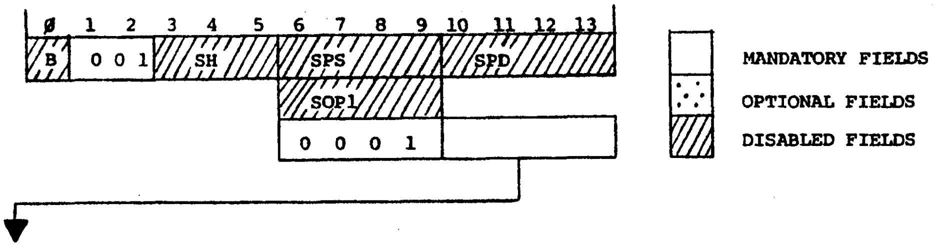
3 **DBLLIT** $\text{TRANSFER DATA PAD BUS}^{\text{LOW MANTISSA}}$ to the LITES REGISTER, via PANEL BUS

Assembler Format: DBLLIT

Effect: (DB^{LMANT}) → PNLBS → LITES

Description: The current data enabled onto the $\text{DATA PAD BUS}^{\text{LOW MANTISSA}}$ $(\text{MANTISSA}^{\text{Bits 12-27}})$ is loaded into the LITES REGISTER,via the PANEL BUS.

HOST/PANEL FIELD (HOSTPNL)



VALUE

4 through 7 — NOP — No Operation

1Ø — SWDB — TRANSFER SWITCHES to the DATA PAD BUS, via PANEL BUS

Assembler Format: SWDB

Example: DPX(1) < DB; SWDB

Effect:

$$\left.\begin{array}{l}(\text{SWR}^{\text{Bits Ø6-15}}) \longrightarrow \text{DB}^{\text{EXP}} \text{ Bits Ø2-11,}\\ (\text{SWR}^{\text{Bits Ø4-15}}) \longrightarrow \text{DB}^{\text{MANT}} \text{Bits ØØ-11,}\\ (\text{SWR}) \longrightarrow \text{DB}^{\text{MANT}} \text{Bits 12-27.}\end{array}\right\} \text{ via PNLBS}$$

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

SWR(bits 06-15) are enabled onto the DATA PAD BUS(EXPONENT),

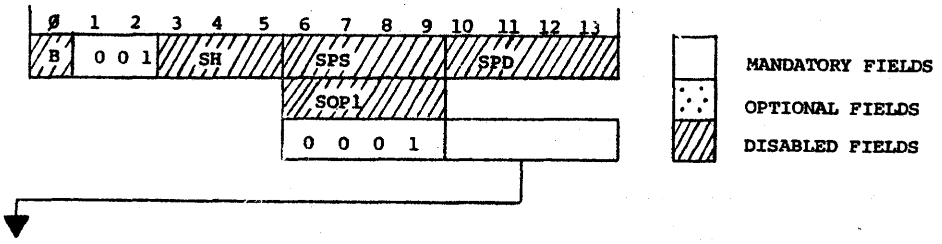
SWR(bits 04-15) are enabled onto the DATA PAD BUS(HIGH MANTISSA), and

SWR(bits 00-15) are enabled onto the DATA PAD BUS(LOW MANTISSA).

The transfer is via PNLBS.

This instruction is used concurrently with a write from DATA PAD BUS operation to transfer the contents of SWR into a designated memory location or Register. Use of this instruction disables any current DB or PNLBS source-enabling operation.

HOST/PANEL FIELD (HOSTPNL)



VALUE

11 $\boxed{\text{SWDBE}}$ TRANSFER SWITCHES to the DATA PAD BUS$^{\text{EXPONENT}}$ via PANEL BUS

Assembler Format: SWDBE

Example: DPY(-2) <DB; SWDBE

Effect:

$(\text{SWR}^{\text{Bits } Ø6\text{-}15}) \longrightarrow \text{DB}^{\text{EXP}}$ Bits Ø2-11,

$(\text{SWR}^{\text{Bits } Ø4\text{-}15}) \longrightarrow \text{DB}^{\text{MANT}}$ Bits ØØ-11,

$(\text{SWR}) \longrightarrow \text{DB}^{\text{MANT}}$ Bits 12-27.

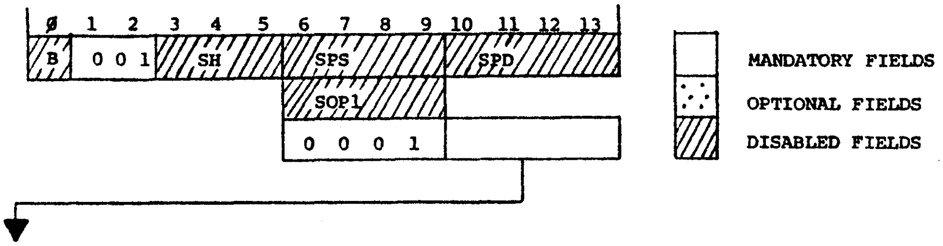
via PNLBS; WRTEXP[†] is forced

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) and a WRTEXP is forced.

A WRTEXP is forced as part of this instruction, restricting the writing to the EXPONENT portion of the designated memory location.

[†] see, SOP1 for a detailed description of WRTEXP

HOST/PANEL FIELD (HOSTPNL)



VALUE

12 SWDBH TRANSFER SWITCHES to DATA PAD BUS HIGH MANTISSA

Assembler Format: SWDBH

Example: SETMA; MI<DB; SWDBH

Effect:

(SWR$^{\text{Bits Ø6-15}}$) ⟶ DB$^{\text{EXP}}$ Bits Ø2-11,
(SWR$^{\text{Bits Ø4-15}}$) ⟶ DB$^{\text{MANT}}$ Bits ØØ-11,
(SWR) ⟶ DB$^{\text{MANT}}$ Bits 12-27.

} via PNLBS;WRTHMN[†] is forced

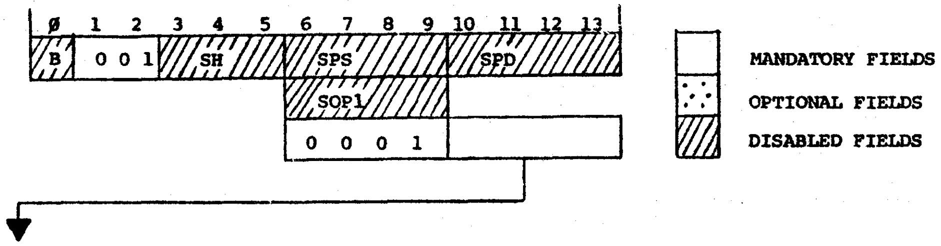
Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

The transfer is via PNLBS.

A WRTHMN is forced as part of this instruction, restricting the writing to the HIGH-MANTISSA portion (MANTISSA[bits 00-11]) of the designated memory location, only.

[†] See SOP1 for a detailed description of WRTHMN

HOST/PANEL FIELD (HOSTPNL)



VALUE

14 through 17	NOP	No Operation
13	SWDBL	TRANSFER SWITCHES to DATA PAD BUS$^{\text{LOW MANTISSA}}$

Assembler Format: SWDBL

Example: SWDBL; DPY(1)<DB

Effect:

(SWR$^{\text{Bits Ø6-15}}$) ⟶ DB$^{\text{EXP}}$ Bits Ø2-11,
(SWR$^{\text{Bits Ø4-15}}$) ⟶ DB$^{\text{MANT}}$ Bits ØØ-11,
(SWR) ⟶ DB$^{\text{MANT}}$ Bits 12-27.

via PNLBS; WRTLMN[†] is forced

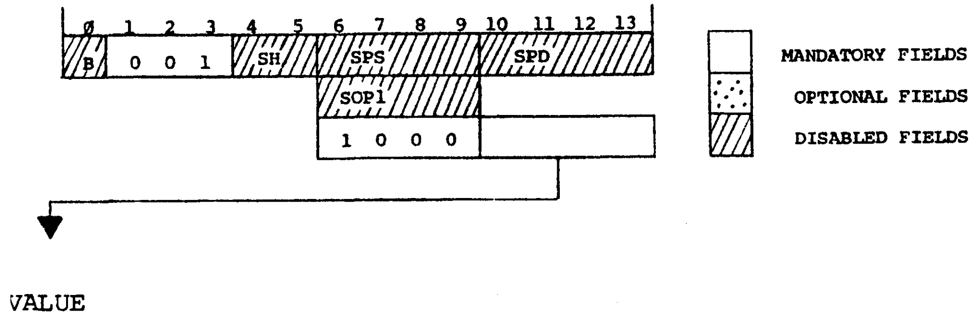
Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

The transfer is via PNLBS.

A WRTLMN is forced as part of this instruction, restricting the writing to the LOW-MANTISSA portion (MANTISSA[bits 12-27]) of the designated memory location only.

[†] See SOP1 for a detailed description of WRTLMN

SET PROGRAM SOURCE ADDRESS (SETPSA)



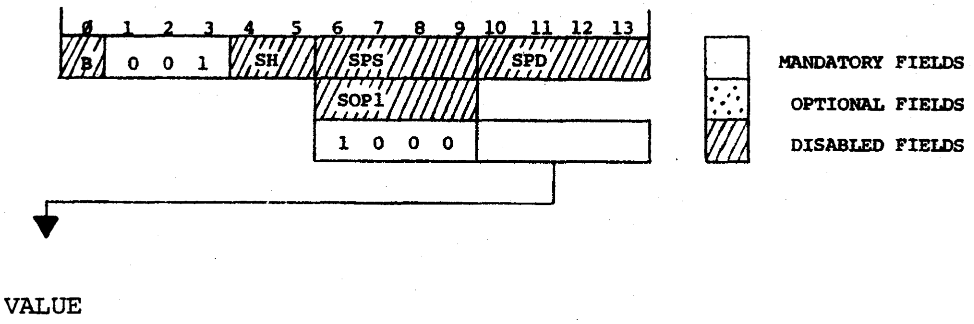
Ø JMPA JUMP (ABSOLUTE)

Assembler Format: JMPA adr

Effect: (VALUE) → PSA

Description: JMPA forces the address contained in the least-significant 12 bits of the VALUE field (Instruction Word$^{\text{Bits 48-63}}$) into the 12-bit PROGRAM SOURCE ADDRESS REGISTER (PSA). The effect is an ABSOLUTE JUMP to the location in program memory contained in the VALUE field.

SET PROGRAM SOURCE ADDRESS (SETPSA)



1 JSRA JUMP to SUB ROUTINE (ABSOLUTE)

Assembler
Format: JSRA adr

Effect: (SRA) + 1 → SRA; (PSA) + 1 → SRS_{SRA}; (VALUE) → PSA

Description: SUB-ROUTINE JUMP. First, the contents of the current PROGRAM SOURCE ADDRESS (PSA), plus "1," are saved by incrementing the SUB-ROUTINE STACK ADDRESS REGISTER (SRA) and storing the current [(PSA) + 1] into the "last-in" location of the SUB-ROUTINE RETURN STACK (SRS).

The least-significant 12 bits of the VALUE field (Instruction Word[bits 48-63]) are then loaded into the 12-bit PROGRAM SOURCE ADDRESS REGISTER (PSA). The program then jumps to that location. (See also RETURN, BRANCH.)

WARNINGS: The JSR instructions have a timing problem that causes certain instruction sequences to execute improperly. The following sequences should be avoided by the programmer:

1) Instruction from PS, PSEVEN or PSODD followed by a JSR.

```
example:  LDPSL
          JSRA
```

2) HALT instruction or a Panel Breakpoint before a JSR.

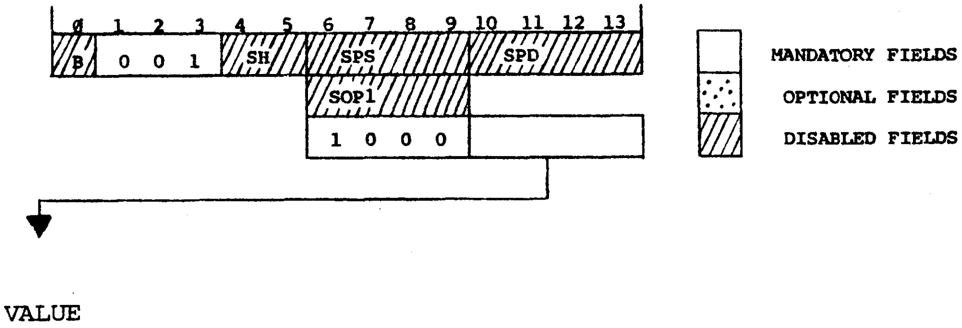
```
example:  HALT
          JSRT
```

3) Execution of consecutive RETURN instructions. This appears in the coding as a JSR followed by a RETURN.

```
example:  JSR
          RETURN
```

These cases all result in a mismatch between the subroutine Return address pointers (SRA) and program execution, and thus will cause severe debugging problems if extreme caution is not taken to avoid them.

SET PROGRAM SOURCE ADDRESS (SETPSA)



2 JMP JUMP (RELATIVE)

Assembler
Format: JMP adr

Effect: (VALUE) + (PSA) → PSA

Description: UNCONDITIONAL RELATIVE JUMP. The address contained on the least-significant 12 bits of the VALUE field (Instruction Words$^{\text{Bits 48-63}}$) is added to the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA). The address thus formed is loaded into PSA and the program jumps to that location.

3 JSR JUMP TO SUB ROUTINE (RELATIVE)

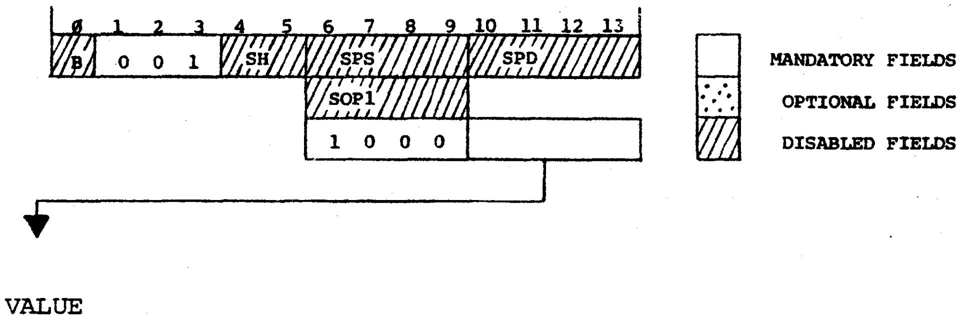
Assembler
Format: JSR adr

Effect: (SRA) + 1 → SRA; (PSA) + 1 → SRS_{SRA};
(VALUE) + (PSA) → PSA

Description: RELATIVE SUB-ROUTINE JUMP. First, the current contents of the PROGRAM SOURCE ADDRESS (PSA) plus "1" are saved by incrementing the SUB-ROUTINE ADDRESS REGISTER (SRA) and storing [(PSA) + 1] into the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS_{SRA}).

The address contained in the least-significant 12 bits of the VALUE field (Instruction Word$^{\text{Bits 48-63}}$) is added to the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA). The address thus formed is loaded into PSA and the program jumps to that location, (see also, RETURN, BRANCH).

SET PROGRAM SOURCE ADDRESS (SETPSA)



4 JMPT JUMP to location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler
Format: JMPT

Example: ADD 1,2; SETTMA

Effect: (TMA) → PSA

Description: UNCONDITIONAL ABSOLUTE JUMP. The contents of the PROGRAM SOURCE REGISTER (PSA) are replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA).

5 JSRT JUMP to SUB-ROUTINE at location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler
Format: JSRT

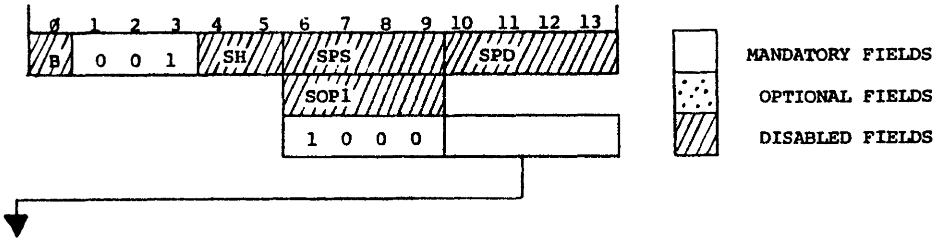
Example: ADD 1,2; SETTMA
JSRT

Effect: $(\text{SRA}) + 1 \rightarrow \text{SRA}; \ (\text{PSA}) + 1 \rightarrow \text{SRS}_{\text{SRA}}; \ (\text{TMA}^{\text{Bits } \emptyset 4\text{-}15}) \rightarrow \text{PSA}$

Description: UNCONDITIONAL ABSOLUTE SUB-ROUTINE JUMP. First, the current PROGRAM SOURCE ADDRESS plus "1" is saved by incrementing the SUB-ROUTINE ADDRESS POINTER (SRA) and storing [(PSA) + 1] into the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS_{SRA}).

The contents of PSA are then replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA). The program then jumps to the new PROGRAM SOURCE location. (See also, RETURN, BRANCH).

SET PROGRAM SOURCE ADDRESS (SETPSA)



VALUE

6 JMPP

JUMP to location indicated by SWITCH REGISTER, via PANEL BUS (ABSOLUTE)

Assembler Format: JMPP

Effect: $(SRW^{\text{Bits Ø4-15}}) \rightarrow \text{PNLBS} \rightarrow \text{PSA}$

Description: UNCONDITIONAL ABSOLUTE JUMP. The current contents of the least-significant 12 bits of the SWITCH REGISTER (SWR) are loaded into the PROGRAM SOURCE ADDRESS REGISTER (PSA) via the PANEL BUS (PNLBS).

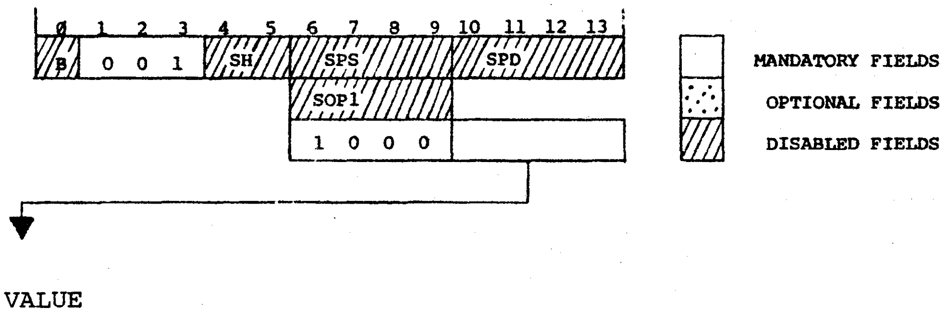
The program then jumps to the new PROGRAM SOURCE location.

WARNING: Propagation delays inherent when executing JMPP may disallow proper decoding of certain target instructions. The perferred alternative sequence is as follows:

t_n SWDB; LDTMA

t_{n+1} JMPT

SET PROGRAM SOURCE ADDRESS (SETPSA)



7 **JSRP** JUMP to SUB-ROUTINE pointed by the SWITCH REGISTER

Assembler Format: JSRP

Effect: $(SRA) + 1 \rightarrow SRA$; $(PSA) + 1 \rightarrow SRS_{SRA}$;
$(SWR^{\text{Bits Ø4-15}}) \rightarrow PNLBS \rightarrow PSA$

Description: UNCONDITIONAL ABSOLUTE SUB-ROUTINE JUMP. First, the current PROGRAM SOURCE ADDRESS (PSA) plus "1" is "saved" by incrementing the SUB-ROUTINE ADDRESS POINTER (SRA) by "1" and storing [(PSA) + 1] into the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS_{SRA}).

Then the contents of the PROGRAM SOURCE ADDRESS REGISTER are replaced by the current contents of the least significant 12 bits of the SWITCH REGISTER (via the PANEL BUS). The program then jumps to the new PROGRAM SOURCE location. (See also, RETURN, BRANCH).

WARNING: Propagation delays inherent when executing JSRP may disallow proper decoding of certain target instructions. The preferred alternative sequence is as follows:

t_n SWDB; LDTMA
t_{n+1} JSRT

PSEVEN, PSODD AND PS FIELDS

The following instructions available in the PSEVEN and PSODD fields involve PROGRAM SOURCE partial-word transfers via the PANEL BUS. Addressing for a given instruction may be RELATIVE or ABSOLUTE. (See SPEC SUMMARY.) All instructions within these fields require two cycles tc execute and instructions from other fields which reference PNLBS for addressing or use PNLBS as a transfer-conduit should not be concurrently specified.

Formats for the various PS partial-words is given below:

Please note the warnings in the HALT, BDBN, BDBZ and JSRA descriptions for WRT the PS, PSEVEN and PSODD instructions.

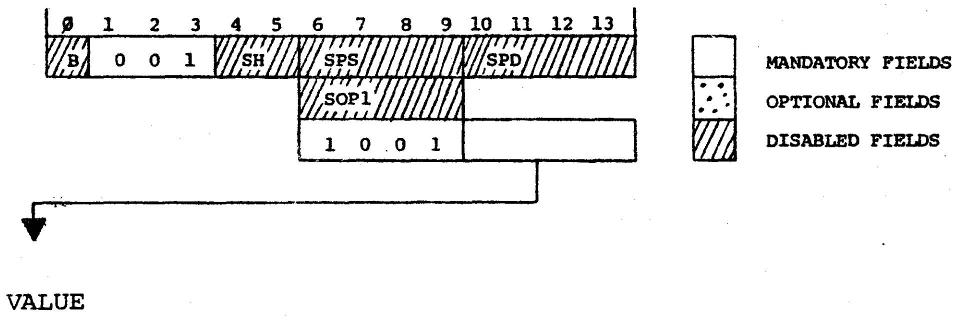
NOTES

1) When VALUE or TMA is used as an addressing subscript, the least-significant 12 bits are used.
Example:
PS(Q0)(VALUE)=PS(Q0)(VALUE)(bits 52-63)
PS(Q0)(TMA)=PS(Q0)(TMA)(bits 04-15)

TMA is TMA Register, not the Table Memory Address which may be modified by the FFT bit APSTATUS.

2) PS Quarter 0 is Bits 00 to 15 = PS(Q0)
1 16 to 31 = PS(Q1)
2 32 to 47 = PS(Q2)
3 48 to 64 = PS(Q3)

PROGRAM SOURCE - EVEN FIELD (PSEVEN)



Ø **RPSØA** READ PROGRAM SOURCE$^{\text{QUARTER Ø}}$ to LITES (ABSOLUTE)

Assembler Format: RPSØA adr

Effect: $(PS^{\text{QØ}}_{\text{VALUE}})$ → PNLBS → LITES

1 **RPS2A** READ PROGRAM SOURCE$^{\text{QUARTER 2}}$ from LITES (ABSOLUTE)

Assembler Format: RPS2A adr

Effect: $(PS^{\text{Q2}}_{\text{VALUE}})$ → PNLBS → LITES

2 **RPSØ** READ PROGRAM SOURCE$^{\text{QUARTER Ø}}$ to LITES (RELATIVE)

Assembler Format: RPSØ adr

Effect: $(PS^{\text{QØ}}_{\text{VALUE + PSA}})$ → PNLBS → LITES

3 **RPS2** READ PROGRAM SOURCE$^{\text{QUARTER 2}}$ to LITES (RELATIVE)

Assembler Format: RPS2 adr

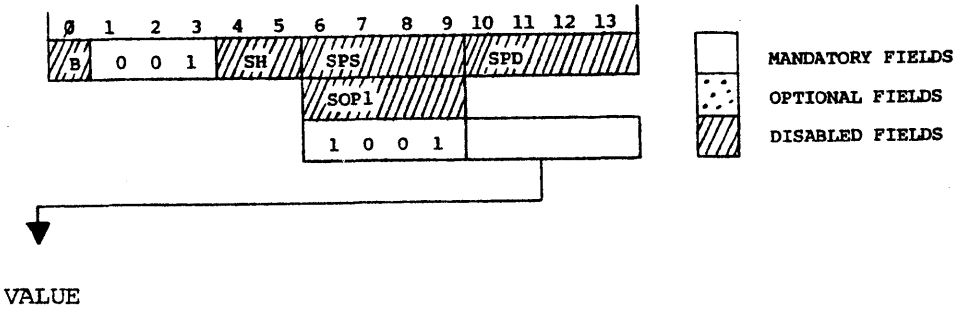
Effect: $(PS^{\text{Q2}}_{\text{VALUE + PSA}})$ → PNLBS → LITES

4 **RPSØT** READ PROGRAM SOURCE$^{\text{QUARTER Ø}}$ to LITES from the location specified by TABLE MEMORY ADDRESS

Assembler Format: RPSØT

Effect: $(PS^{\text{QØ}}_{\text{TMA}})$ → PNLBS → LITES

PROGRAM SOURCE - EVEN FIELD (PSEVEN)



5 RPS2T

READ PROGRAM $\text{SOURCE}^{\text{QUARTER 2}}$ to LITES from the location specified by TABLE MEMORY ADDRESS

Assembler Format: RPS2T

Effect: (PS^{Q2}_{TMA}) → PNLBS → LITES

6 and 7 NOP

No Operation

1Ø WPSØA

WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER Ø}}$ from the SWITCHES (ABSOLUTE)

Assembler Format: WPSØA adr

Effect: (SWR) → PNLBS → $PS^{QØ}_{VALUE}$

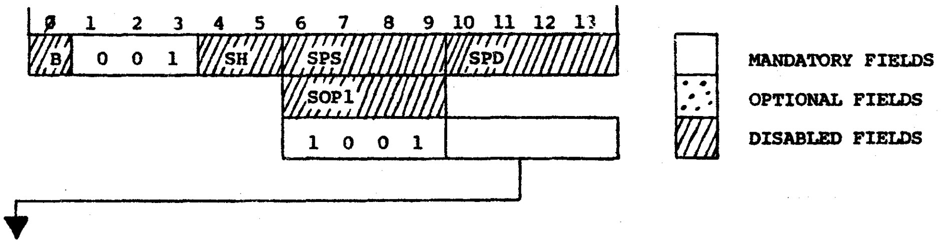
11 WPS2A

WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER 2}}$ from the SWITCHES (ABSOLUTE)

Assembler Format: WPS2A adr

Effect: (SWR) → PNLBS → PS^{Q2}_{VALUE}

PROGRAM SOURCE - EVEN FIELD (PSEVEN)



VALUE

12 WPSØ

WRITE PROGRAM SOURCE$^{\text{QUARTER Ø}}$ from the SWITCHES (RELATIVE)

Assembler Format: WPSØ adr

Effect: (SWR) → PNLBS → $PS^{QØ}_{VALUE + PSA}$

13 WPS2

WRITE PROGRAM SOURCE$^{\text{QUARTER 2}}$ from the SWITCHES (RELATIVE)

Assembler Format: WPS2 adr

Effect: (SWR) → PNLBS → $PS^{Q2}_{VALUE + PSA}$

14 WPSØT

WRITE PROGRAM SOURCE$^{\text{QUARTER Ø}}$ from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPSØT

Effect: (SWR) → PNLBS → PS^{Q2}_{TMA}

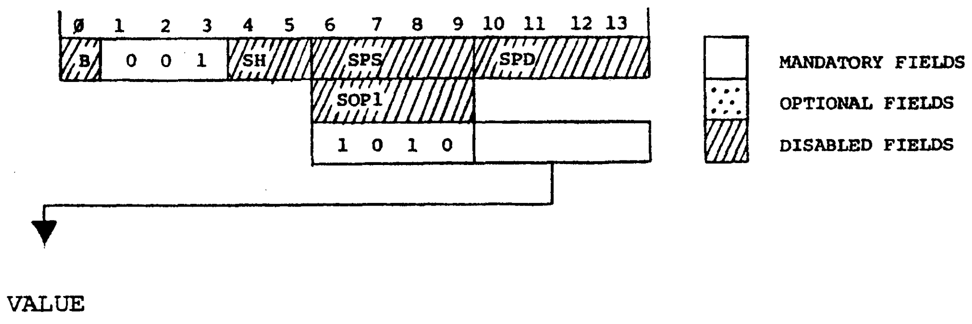
15 WPS2T

WRITE PROGRAM SOURCE$^{\text{QUARTER 2}}$from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPS2T

Effect: (SWR) → PNLBS → PS^{Q2}_{TMA}

PROGRAM SOURCE - ODD FIELD (PSODD)



Ø RPS1A READ PROGRAM SOURCE$^{\text{QUARTER 1}}$ to LITES (ABSOLUTE)

Assembler Format: RPS1A adr

Effect: (PS^{Q1}_{VALUE}) → PNLBS → LITES

1 RPS3A READ PROGRAM SOURCE$^{\text{QUARTER 3}}$ from LITES (ABSOLUTE)

Assembler Format: RPS3A adr

Effect: (PS^{Q3}_{VALUE}) → PNLBS → LITES

2 RPS1 READ PROGRAM SOURCE$^{\text{QUARTER 1}}$ to LITES (RELATIVE)

Assembler Format: RPS1 adr

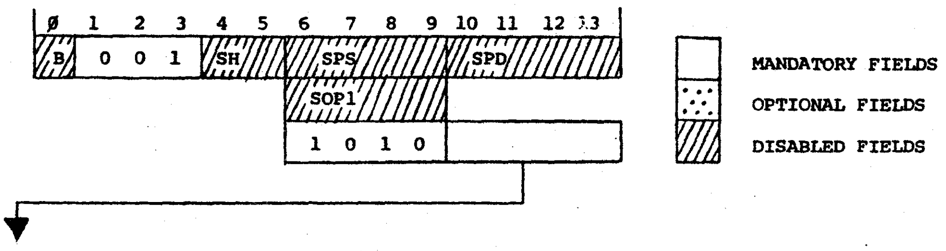
Effect: $(PS^{Q1}_{VALUE + PSA})$ → PNLBS → LITES

3 RPS3 READ PROGRAM SOURCE$^{\text{QUARTER 3}}$ to LITES (RELATIVE)

Assembler Format: RPS3 adr

Effect: $(PS^{Q3}_{VALUE + PSA})$ → PNLBS → LITES

PROGRAM SOURCE - ODD FIELD (PSODD)



VALUE

4 RPS1T

READ PROGRAM SOURCE$^{\text{QUARTER 1}}$ to LITES from the location specified by TABLE MEMORY ADDRESS

Assembler Format: RPS1T

Effect: $(\text{PS}^{Q1}_{\text{TMA}}) \rightarrow \text{PNLBS} \rightarrow \text{LITES}$

5 RPS3T

READ PROGRAM SOURCE$^{\text{QUARTER 3}}$ to LITES from the location specified by TABLE MEMORY ADDRESS

Assembler Format: RPS3T

Effect: $(\text{PS}^{Q3}_{\text{TMA}}) \rightarrow \text{PNLBS} \rightarrow \text{LITES}$

6 and 7 NOP

No Operation

1Ø WPS1A

WRITE PROGRAM SOURCE$^{\text{QUARTER 1}}$ from the SWITCHES (ABSOLUTE)

Assembler Format: WPS1A adr

Effect: $(\text{SWR}) \rightarrow \text{PNLBS} \rightarrow \text{PS}^{Q1}_{\text{VALUE}}$

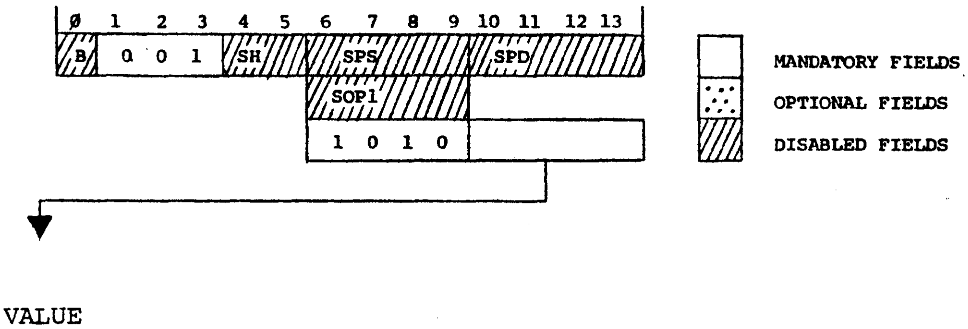
11 WPS3A

WRITE PROGRAM SOURCE$^{\text{QUARTER 3}}$ from the SWITCHES (ABSOLUTE)

Assembler Format: WPS3A adr

Effect: $(\text{SWR}) \rightarrow \text{PNLBS} \rightarrow \text{PS}^{Q3}_{\text{VALUE}}$

PROGRAM SOURCE - ODD FIELD (PSODD)



12 WPS1 WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER 1}}$ from the SWITCHES (RELATIVE)

Assembler Format: WPS1 adr

Effect: (SWR) → PNLBS → $\text{PS}^{\text{Q1}}_{\text{VALUE + PSA}}$

13 WPS3 WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER 3}}$ from the SWITCHES (RELATIVE)

Assembler Format: WPS3 adr

Effect: (SWR) → PNLBS → $\text{PS}^{\text{Q3}}_{\text{VALUE + PSA}}$

14 WPS1T WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER 1}}$ from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPS1T

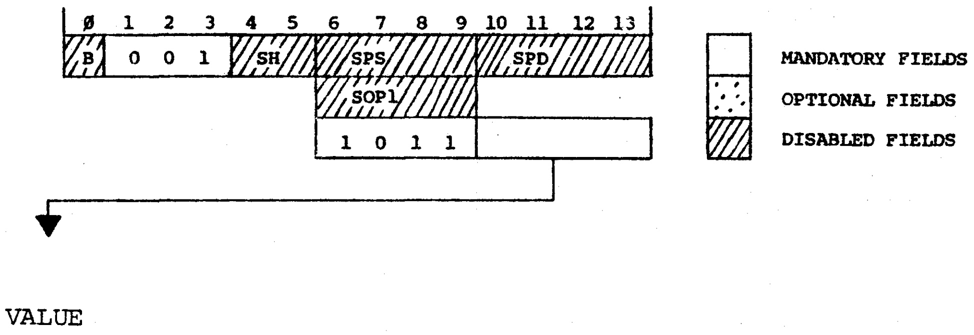
Effect: (SWR) → PNLBS → $\text{PS}^{\text{Q1}}_{\text{TMA}}$

15 WPS3T WRITE PROGRAM $\text{SOURCE}^{\text{QUARTER 3}}$ from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPS3T

Effect: (SWR) → PNLBS → $\text{PS}^{\text{Q3}}_{\text{TMA}}$

PROGRAM SOURCE FIELD (PS)



Ø **RPSLA** READ PROGRAM SOURCE$^{\text{LEFT HALF}}$ to DATA PAD BUS (ABSOLUTE)

Assembler Format: RPSLA adr

Effect: $(PS^{LH}_{VALUE}) \rightarrow DB$

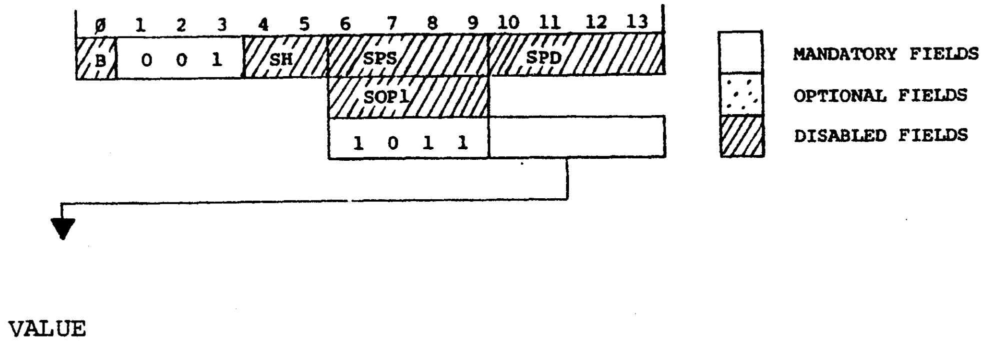
Description: PROGRAM SOURCE(LEFT HALF) (PS LH]=PS bits 00-31]), as addressed by the least-significant 12 bits contained in the VALUE field (Instruction Word bits 48-63]), is enabled onto DATA PAD BUS (DB).

The transfer is executed in the following manner:

* ZEROS ⟶ DB(EXP)Bits 02-07
* PS(Bits 00-03) ⟶ DB(EXP)Bits 08-11
* PS(Bits 04-31) ⟶ DB(MANT)Bits 00-27

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



1 RPSFA

READ PROGRAM SOURCE$^{\text{FLOATING-POINT LITERAL}}$ to DATA PAD BUS (ABSOLUTE)

Assembler Format: RPSFA adr

Effect: $(PS^{FPL}_{VALUE}) \rightarrow DB$

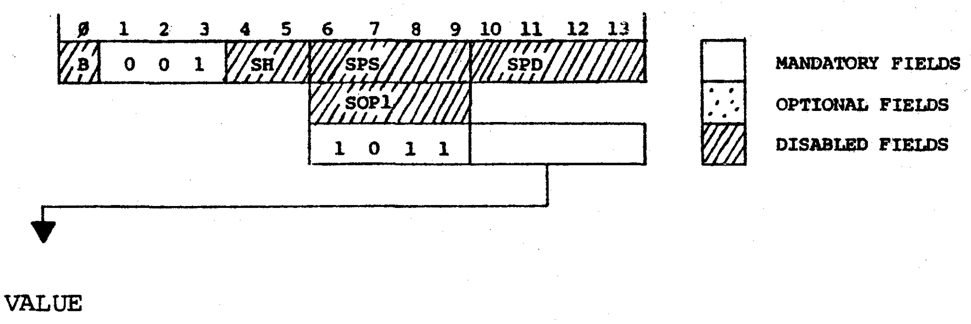
Description: PROGRAM SOURCE(FLOATING-POINT LITERAL) (PS[FPL]=PS bits [26-63]), as addressed by the least-significant 12 bits contained in the VALUE field (Instruction Word bits 48-63]), is enabled onto DATA PAD BUS (DB).

The transfer is executed in the following manner:

* PS(Bits 26-35)	DB(EXP)Bits 02-11
* PS(Bits 36-63)	DB(MANT)Bits 00-27

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



2 RPSL

READ PROGRAM $\text{SOURCE}^{\text{LEFT HALF}}$ to DATA PAD BUS (RELATIVE)

Assembler Format: RPSL adr

Effect: $(PS^{LH}_{VALUE + PSA}) \rightarrow DB$

3 RPSF

READ PROGRAM $\text{SOURCE}^{\text{FLOATING-POINT LITERAL}}$ to DATA PAD BUS (RELATIVE)

Assembler Format: RPSF adr

Effect: $(PS^{FPL}_{VALUE + PSA}) \rightarrow DB$

This instruction requires two cycles to execute.

4 RPSLT

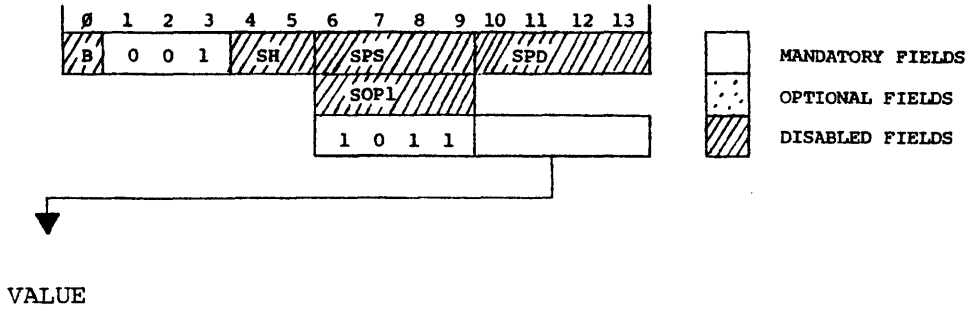
READ PROGRAM $\text{SOURCE}^{\text{LEFT HALF}}$ to DATA PAD BUS from the location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler Format: RPSLT

Effect: $(PS^{LH}_{TMA}) \rightarrow DB$

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



5 RPSFT

READ PROGRAM SOURCE$^{\text{FLOATING-POINT LITERAL}}$ to DATA PAD BUS from the location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler Format: RPSFT

Effect: $(PS^{FPL}_{TMA}) \rightarrow DB$

This instruction requires two cycles to execute.

6 RPSLP

READ PROGRAM SOURCE$^{\text{LEFT HALF}}$ to DATA PAD BUS from the address contained on PANEL BUS (ABSOLUTE)

Assembler Format: RPSLP

Effect: $(PS^{LH}_{PNLBS}) \rightarrow DB$

This instruction requires two cycles to execute.

7 RPSFP

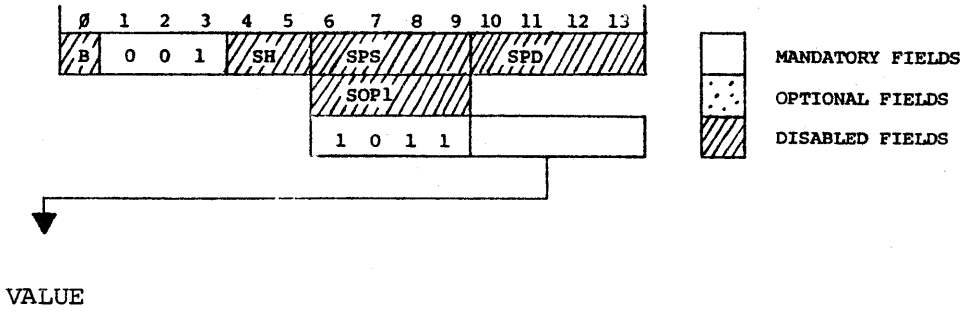
READ PROGRAM SOURCE$^{\text{FLOATING-POINT LITERAL}}$ to DATA PAD BUS from the address contained on PANEL BUS (ABSOLUTE)

Assembler Format: RPSFP

Effect: $(PS^{FPL}_{PNLBS}) \rightarrow DB$

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



1Ø LPSLA LOAD PROGRAM SOURCE$^{\text{LEFT HALF}}$ from DATA PAD BUS (ABSOLUTE)

Assembler Format: LPSLA adr

Effect: (DB) → PS^{LH}_{VALUE}

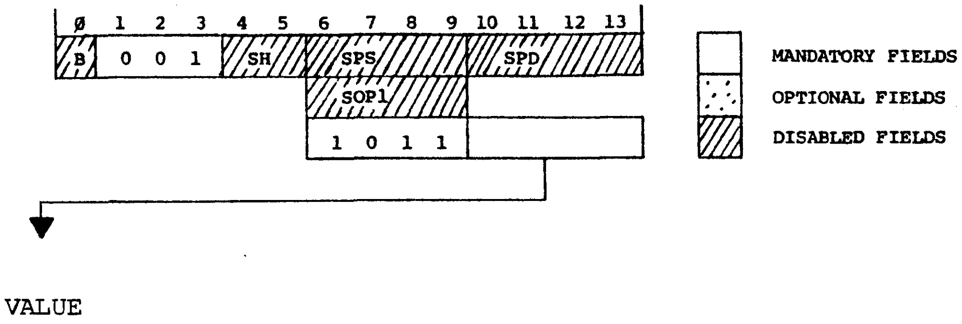
Description: The right-most 32 bits of the DATA currently enabled onto DATA PAD BUS (DB) are loaded into the PROGRAM SOURCE(LEFT HALF) (PS LH]=PS bits 00-31]).

The transfer is executed in the following manner:

* DB(EXP) Bits 08-11 → PS(Bits 00-03)
* DB(MANT) Bits 00-27 → PS(Bits 04-31)

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



11 LPSRA

LOAD PROGRAM SOURCE$^{\text{RIGHT HALF}}$ from DATA PAD BUS (ABSOLUTE)

Assembler
Format: LPSRA adr

Effect: (DB) → PS^{RH}_{VALUE}

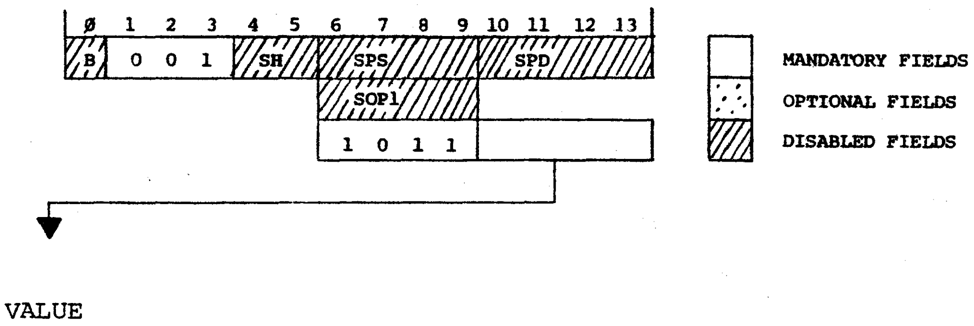
Description: The right-most 32 bits of data currently enabled onto the DATA PAD BUS (DB) are loaded into the PROGRAM SOURCE(RIGHT HALF) (PS [RH]=PS bits [48-63]) as addressed by the least-significant 12 bits of the VALUE field (Instruction Word bits 48-63]).

The transfer is executed in the following manner:

* DB(EXP) Bits 08-11 ⟶ PS(Bits 32-35)
* DB(MANT) Bits 00-27 ⟶ PS(Bits 36-63)

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



12 **LPSL** LOAD PROGRAM SOURCE$^{\text{LEFT HALF}}$ from DATA PAD BUS (RELATIVE)

Assembler Format: LPSL adr

Effect: $(DB) \rightarrow PS^{LH}_{VALUE + PSA}$

This instruction requires two cycles to execute.

13 **LPSR** LOAD PROGRAM SOURCE$^{\text{RIGHT HALF}}$ from DATA PAD BUS (RELATIVE)

Assembler Format: LPSR adr

Effect: $(DB) \rightarrow PS^{RH}_{VALUE + PSA}$

This instruction requires two cycles to execute.

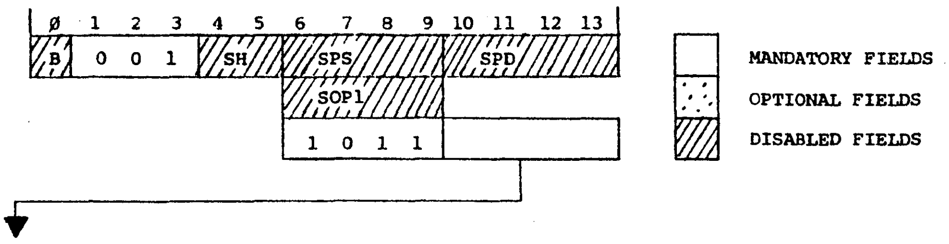
14 **LPSLT** LOAD PROGRAM SOURCE$^{\text{LEFT HALF}}$ from DATA PAD BUS as addressed by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler Format: LPSLT

Effect: $(DB) \rightarrow PS^{LH}_{TMA}$.

This instruction requires two cycles to execute.

PROGRAM SOURCE FIELD (PS)



VALUE

15 LPSRT

LOAD PROGRAM SOURCE$^{\text{RIGHT HALF}}$ from DATA PAD BUS as addressed by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler Format: LPSRT

Effect: $(DB) \rightarrow PS^{RH}_{TMA}$

This instruction requires two cycles to execute.

16 LPSLP

LOAD PROGRAM SOURCE$^{\text{LEFT HALF}}$ from DATA PAD BUS at the address contained on PANEL BUS (ABSOLUTE)

Assembler Format: LPSLP

Effect: $(DB) \rightarrow PS^{LH}_{PNLBS}$

This instruction requires two cycles to execute.

17 LPSRP

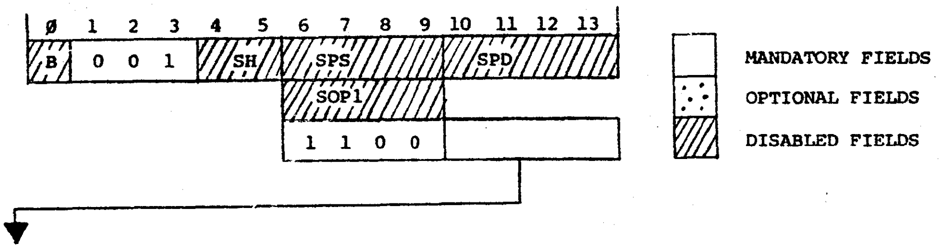
LOAD PROGRAM SOURCE$^{\text{RIGHT HALF}}$ from DATA PAD BUS at the address contained on PANEL BUS (ABSOLUTE)

Assembler Format: LPSRP

Effect: $(DB) \rightarrow PS^{RH}_{PNLBS}$

This instruction requires two cycles to execute.

SET EXIT FIELD (SETEXIT)



VALUE

Ø NOP No Operation

1 SETEXA SET EXIT ADDRESS (ABSOLUTE)

Assembler Format: SETEXA

Effect: (VALUE) → SRS_{SRA}

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS[SRA]), the "last-in" address, are replaced by the least-significant 12 bits of the VALUE field (Instruction Word[bits 48-63]). SRA not affected.

2 NOP No Operation

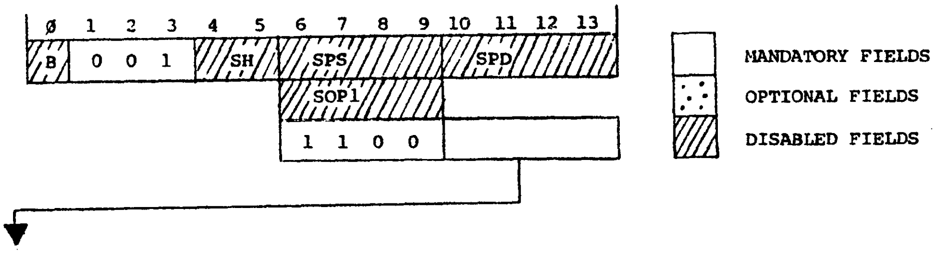
3 SETEX SET EXIT ADDRESS (RELATIVE)

Assembler Format: SETEX adr

Effect: (VALUE = PSA) → SRS_{SRA}

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced with the address formed by adding the current contents of the PROGRAM SOURCE ADDRESS (PSA) to the least-significant 12 bits contained on the VALUE field.

SET EXIT FIELD (SETEXIT)



VALUE

4 NOP No Operation

5 SETEXT SET EXIT ADDRESS from TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler
Format: SETEXT

Effect: (TMA) → SRS_{SRA}

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA).

6 NOP No Operation

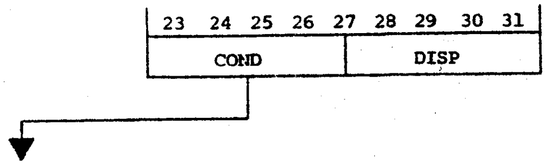
7 SETEXP SET EXIT ADDRESS from PROGRAM SOURCE ADDRESS plus ONE (ABSOLUTE)

Assembler
Format: SETEXP

Effect: (PSA + 1) → SRS_{SRA}

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced by the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA) plus "1"

BRANCH GROUP (BRANCH)



VALUE

Ø No Operation

1 $\boxed{\#}$ S-PAD DESTINATION REGISTER NO-LOAD

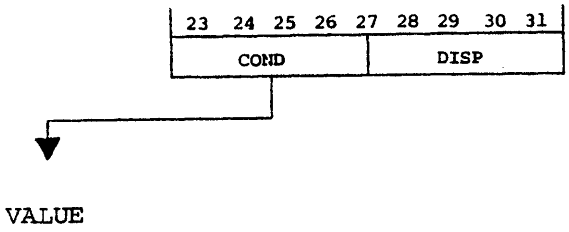
Assembler Format: < # >

(Example: ADDL# 4,5)

Effect: (SPFN)→ SP_{SPD} is inhibited.

Description: This instruction inhibits the normal loading of the current S-PAD OPERATION result (SPFN) back into the S-PAD DESTINATION REGISTER (SP_{SPD}) specified during the S-PAD Operation. (See S-PAD summary)

NOTE: Brackets indicate optional use with S-PAD operations.



2 **BR** UNCONDITIONAL BRANCH (RELATIVE)

Assembler Format: BR disp

(Examples: BR LOOP)

Effect: $(PSA) + (DISP^{\dagger} - BIAS) \rightarrow PSA$

(Where BIAS = 20_8).

Description: UNCONDITIONAL RELATIVE BRANCH

The program will branch to the target location "disp" (Assembler Format) by adding the current PSA to the BIASED value contained in the DISPlacement field of the instruction word.

NOTE: The BRANCH TARGET ADDRESS must be within a range of -20(8) to +17(8) locations relative to the current PROGRAM SOURCE ADDRESS (PSA).

DISP = bits 27-31 of the current instruction word and is completed as follows:

† DISP = disp - PSA + BIAS.

BRANCH GROUP (BRANCH)

23	24	25	26	27	28	29	30	31
COND					DISP			

VALUE

3 BINTRQ BRANCH ON INTERRUPT REQUEST FLAG NON-ZERO

Assembler
Format: BINTRQ disp

Effect: If INTRQ = 1, then (PSA) + (DISP - BIAS)→PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if Interrupt Request.

If the INTERRUPT REQUEST FLAG (INTRQ) equals "1," then the program will branch to the target location "disp."

This instruction can be used in conjunction with a succeeding INTA instruction (see I/O group) to identify the interrupting I/O device.

4 BION BRANCH ON I/O DATA READY FLAG NON-ZERO

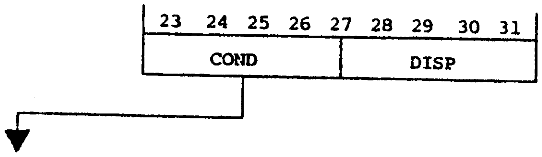
Assembler
Format: BION disp

Effect: If IODRDY (DA) = 1, then (PSA) + (DISP - BIAS)→ PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH is I/O Device Ready.

If the I/O DATA READY FLAG (IODRDY) of the I/O device specified by the I/O DEVICE ADDRESS REGISTER (DA) is "1," then the program will branch to the target location "disp."

BRANCH GROUP (BRANCH)



VALUE

5 BIOZ BRANCH ON I/O DATA READY FLAG ZERO

Assembler Format: BIOZ disp

Effect: If $(\text{IODRDY}_{\text{DA}}) = \emptyset$, then (PSA) + (DISP - BIAS)→PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if I/O Device not ready.

If the I/O DATA READY FLAG (IODRDY) of the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA) equals "0," then the program will branch to the target location "disp."

6 BFPE BRANCH ON FLOATING POINT ERROR

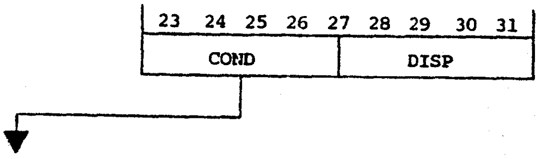
Assembler Format: BFPE disp

Effect: If OVF, UNF, or DIVZ = "1", then (PSA) + (DISP - BIAS)→ PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if Floating Point Error.

The OVERFLOW (OVF), UNDERFLOW (UNF), and DIVIDE BY ZERO (DIVZ), FLAGS (bits 0, 1, 2 of the APSTATUS REGISTER) are tested. If any of the three flags = "1," then a branch will occur to the target location "disp."

BRANCH GROUP (BRANCH)



VALUE

7 RETURN RETURN FROM SUB-ROUTINE

Assembler
Format: RETURN

Effect: $(SRS_{SRA}) \rightarrow PSA; \ (SRA) -1 \rightarrow SRA$

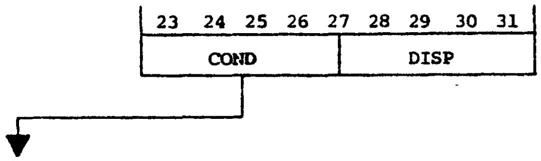
Description: UNCONDITIONAL RETURN JUMP.

The address contained in the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS) is forced into the PROGRAM SOURCE ADDRESS REGISTER (PSA) and the program branches to that program location.

The SUB-ROUTINE ADDRESS POINTER REGISTER (SRA) is then decremented by "1," and will point to the next "last-in" SUB-ROUTINE ADDRESS in event of another RETURN instruction. RETURN effects a "RETURN" from the last SUB-ROUTINE call.

NOTE: Two or more RETURNS may not be executed in time sequential instructions. There must be at least one instruction cycle between the last RETURN instruction and the next one, e.g., the following coding example is illegal in that it results in having the visible RETURN instruction execute immediately after the RETURN instruction in "sub" that brings the processor back to this level.

```
illegal code:       JSR sub
                    RETURN
```



VALUE

10 **BFEQ** BRANCH on FLOATING ADDER EQUAL ZERO

Assembler Format: BFEQ disp

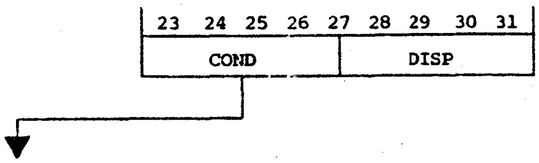
Example: BFEQ .-3

Effect: If FA = 0.0, then (PSA) + (DISP - BIAS)→ PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if FA equal to zero.

BFEQ will cause a PROGRAM BRANCH if the FLOATING ADDER Result (FA) available during the previous instruction was equal to 0.0. This instruction tests the FZ FLAG (bit 3 of APSTATUS) as set by the previous instruction. If FZ is equal to "1," (i.e., FA during the last instruction was equal to 0.0), then the program will branch to the target location "disp."

BRANCH GROUP (BRANCH)



VALUE

11 **BFNE** BRANCH on FLOATING ADDER NOT EQUAL to ZERO

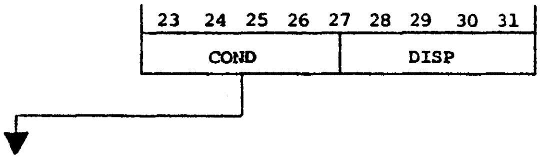
Assembler
Format: BFNE disp

Example: BFNE HELP+6

Effect: If FA ≠ 0.0, then (PSA) + (DISP - BIAS) → PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if FA not equal to zero.

BFNE will cause a PROGRAM BRANCH if the FLOATING ADDER Result for the previous instruction was not equal to 0.0. This instruction tests the FZ flag (bit 3 of APSTATUS) as set by the previous instruction. If FZ equals "0" (i.e., FA for the last instruction was not equal to 0.0), the branch will occur to the target location "disp."



VALUE

12 **BFGE** BRANCH on FLOATING ADDER GREATER or EQUAL to ZERO

Assembler
Format: BFGE disp

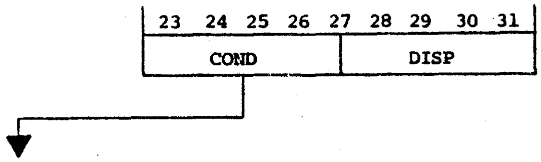
Effect: If FA> 0.0, then (PSA) + (DISP - BIAS)→ PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if FA greater than or equal to zero.

BFGE will cause a PROGRAM BRANCH if the FLOATING ADDER Result (FA) for the previous instruction was greater than or equal to 0.0.

This instruction tests the condition of the FLOATING ADDER NEGATIVE (FN) FLAG (bit 4 of APSTATUS) as set by the previous instruction.

If FN equals "0," (indicating that FA was not negative, i.e., greater than or equal to zero, during the last instruction cycle), a branch will occur to the TARGET ADDRESS "disp."



VALUE

13 **BFGT** BRANCH on FLOATING ADDER GREATER THAN ZERO

Assembler Format: BFGT disp

Effect: If FA > 0.0, then (PSA) + (DISP - BIAS) → PSA

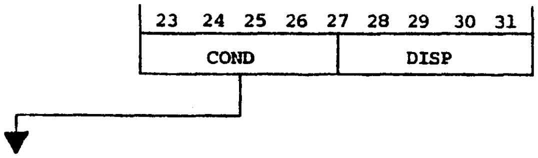
(Where BIAS = 20_8)

Description: CONDITIONAL RELATIVE BRANCH if FA greater than zero.

BFGT will cause a PROGRAM BRANCH to occur if the last FLOATING ADDER RESULT (FA) for the previous instruction was greater than 0.0.

The instruction tests the FLOATING ADDER ZERO (FZ) and FLOATING ADDER NEGATIVE (FN) flags (bits 3 and 4 of APSTATUS) as set by the previous instruction.

If both flags equal "0" (indicating that FA during the last instruction was greater than zero), then the program will branch to the target location "disp."



VALUE

14 BEQ

BRANCH on S-PAD RESULT EQUALS ZERO

Assembler
Format: BEQ disp

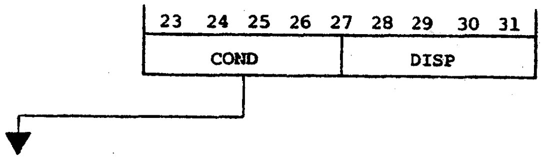
Effect: If SPFN = Ø, then (PSA) + (DISP - BIAS)→ PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if SPFN equals zero.

BEQ will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was equal to zero.

This instruction tests the S-PAD ZERO FLAG (Z) (bit 5 of APSTATUS) as set by the previous instruction. If Z equals "1" (indicating that SPFN of the last S-PAD operation was equal to zero), then a branch will occur to the target location"disp."

BRANCH GROUP (BRANCH)



VALUE

15 BNE BRANCH on S-PAD RESULT NON-ZERO

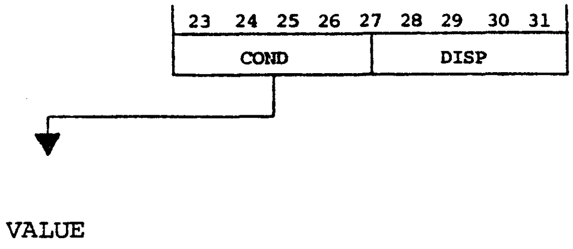
Assembler Format: BNE disp

Effect: If SPFN $\neq \emptyset$, then (PSA) + (DISP + BIAS) → PSA (Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if SPFN not equal to zero.

BNE will cause a PROGRAM RELATIVE BRANCH if the result of the last previous S-PAD operation (SPFN) was not equal to 0.

The instruction tests the S-PAD ZERO FLAG (Z) (bit 5 of APSTATUS) as set by the previous instruction. If Z equals "0" (indicating that SPFN of the last S-PAD operation was not equal to zero), then a branch will occur to the target location "disp."



16 BGE BRANCH on S-PAD RESULT GREATER or EQUAL to ZERO

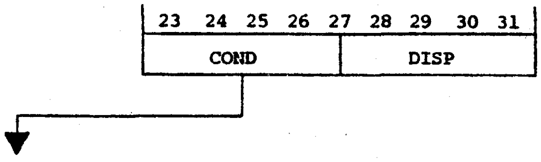
Assembler
Format: BGE disp

Effect: If SPFN > Ø, then (PSA) + (DISP - BIAS) → PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if SPFN greater than or equal to zero.

BGE will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was greater than or equal to zero.

The instruction tests the S-PAD NEGATIVE FLAG (N) (bit 6 of APSTATUS) as set by the previous instruction. If N is equal to "0" (indicating SPFN of the last operation was zero or greater), a branch will occur to the target location "disp."



VALUE

17 BGT BRANCH on S-PAD RESULT GREATER THAN ZERO

Assembler
Format: BGT disp

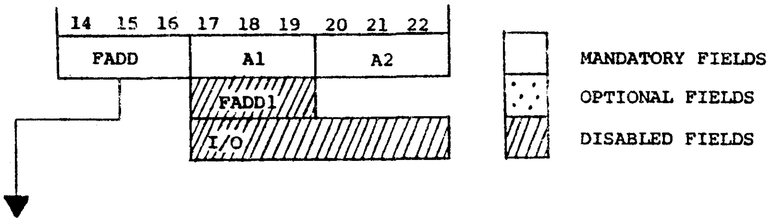
Effect: If SPFN > Ø, then (PSA) + (DISP - BIAS) → PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if SPFN greater than zero.

BGT will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was greater than zero.

The instruction tests the S-PAD ZERO (Z) and S-PAD NEGATIVE (N) flags (bits 5, 6 or APSTATUS) as set by the previous instruction. If both flags equal "0" (indicating SPFN of last S-PAD operation was greater than zero), then a branch will occur to the target location "disp."

FLOATING ADDER GROUP (FADD)



VALUE

0 See FADD1 field.

1 FSUBR FLOATING-POINT SUBTRACT REVERSE;

A2 minus A1.

Assembler Format: FSUBR < A1, A2 > (< > indicates optional fields)

Effect: (A2) - (A1)

FSUBR TM,MD

Description: FSUBR reverses the order of operands in a FLOATING-POINT SUBTRACTION. The contents of A1 REGISTER (A1) undergo a FLOATING-POINT SUBTRACTION from the contents of A2 REGISTER (A2).

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT is available as the FLOATING-ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

2 FSUB FLOATING-POINT SUBTRACT; A1 minus A2.

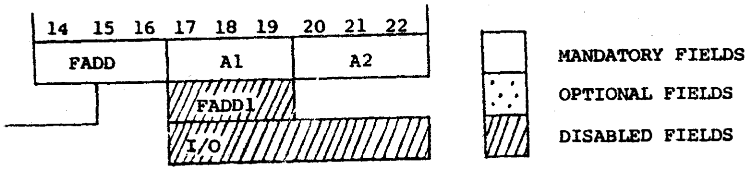
Assembler Format: FSUB < A1, A2 >

Effect: (A1) - (A2)

Description: The contents of A2 REGISTER (A2) undergo a FLOATING-POINT SUBTRACTION from the contents of A1 REGISTER (A1).

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

LOATING ADDER GROUP (FADD)



UE

3 FADD FLOATING-POINT ADD; A1 plus A2.

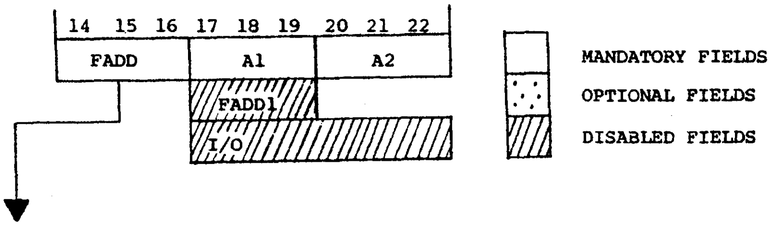
Assembler Format: FADD < A1, A2 >

Effect: (A1) + (A2)

Description: The contents of A1 REGISTER undergo a FLOATING-POINT ADDITION with the contents of A2 REGISTER.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

FLOATING ADDER GROUP (FADD)



VALUE

4 FEQV FLOATING-POINT LOGICAL EQUIVALENCE;
A1 with A2.

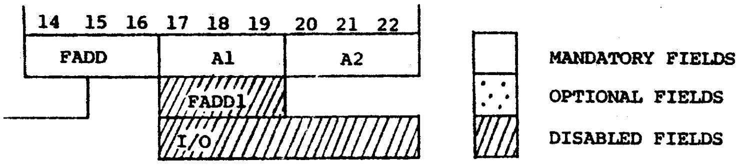
Assembler
Format: FEQV < A1, A2 >

Effect: (A1) $\overline{\text{XOR}}$ (A2)

Description: The MANTISSAS of A1 and A2 are compared for EQUIVALENCE in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2, including the three bits of residue, undergo a "bit by bit" comparison. When corresponding bits of A1 and A2 are equal; (i.e., both "0"s or both "1"s), a "1" is written into the corresponding bit of the RESULT. All other combinations result in a "0" being written.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)



JE

5 FAND

FLOATING-POINT AND; A1, A2.

Assembler
Format: FAND < A1, A2 >

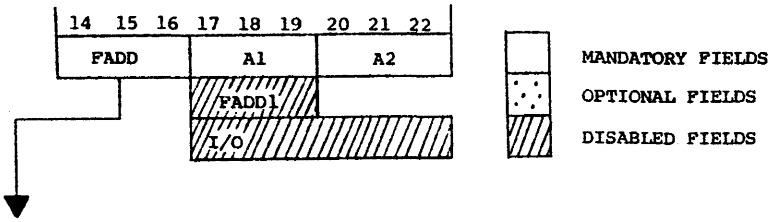
Effect: (A1) and (A2)

Description: The MANTISSAS of A1 and A2 are logically ANDED in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2 undergo a "bit by bit" comparison. When corresponding bits of A1 and A2 both equal "1," then a "1" is written into the corresponding bit position of the RESULT. All other combinations result in a "0" being written.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

FLOATING ADDER GROUP (FADD)



VALUE

6 | FOR | FLOATING-POINT OR; A1 or A2.

Assembler Format: FOR < A1, A2 >

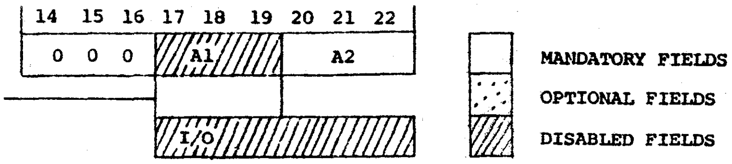
Effect: (A1) OR (A2)

Description: The contents of A1 and A2 are ORed in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2, including the three bits of residue, undergo a "bit by bit" comparison. When either or both corresponding bits of A1 and A2 equal "1," a "1" is written into the corresponding bit position of the RESULT. When neither corresponding bit is equal to "1," a "0" is written.

The NORMALIZED and CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

FLOATING ADDER GROUP (FADD)



LUE

No-operation.

FIX

FIX A2 to an INTEGER (result rounded)

Assembler
Format: FIX <A2> (< > indicates optional field)

Effect: Convert (A2) to a 28-Bit Two's Complement integer.
Example: FIX MD

Description: The contents of A2 are FIXED to an integer in the following manner:

1) An exponent of 28 (apparent value = 1034[octal]) is forced into A1(EXPONENT). A1(MANTISSA) = 0. A2 contains the selected argument to be FIXED.†

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT along with the larger of the two input exponents.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted one position while preserving the MANTISSA-SIGN.

4) The PRELIMINARY-RESULT is then CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

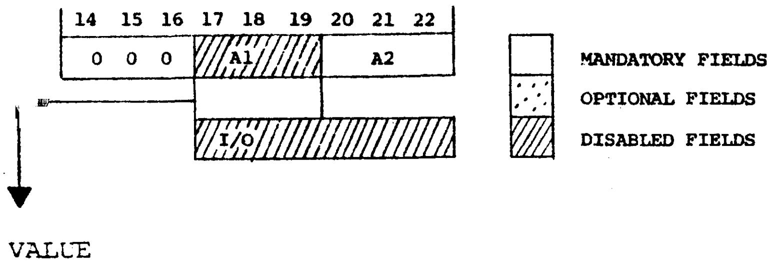
If the TRUE-VALUE of A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033[octal] or 539). If RESULT(MANTISSA) is≤2(-28), then a FLOATING-POINT ZERO is forced as the result.

†The TRUE-VALUE of A2(EXPONENT) must not exceed +27. (APPARENT-VALUE≤1033(octal). If the TRUE-VALUE of A2(EXPONENT)≥28, the following RESULT will be obtained:

RESULT(EXPONENT)=A2(EXPONENT)minus "1;" RESULT(MANTISSA-SIGN)=A2 (MANTISSA-SIGN),

RESULT(MANTISSA Bits 01-26)=A2(MANTISSA Bits 02-27);RESULT (MANTISSA Bit 27)=0.

FLOATING ADDER GROUP (FADD)



2 FIXT

FIX and TRUNCATE A2 to an INTEGER

Assembler Format: FIXT <A2>

Effect: Convert (A2) to a 28-Bit Two's Complement INTEGER; TRUNCATE (sign magnitude)

Example: FIXT DPX(3)

Description: The contents of A2(MANTISSA) are FIXED to an integer and the RESULT is TRUNCATED in the following manner:

1) Current SPFN(Bits 06-15) plus BIAS are forced into A1(EXPONENT). A1(MANTISSA)=0 . A2 contains the selected argument to be FIXED.†

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

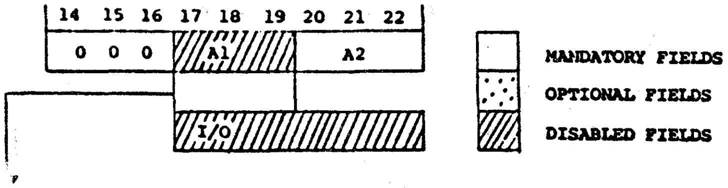
4) The Truncation truth table logic is enabled for this operation (see Floating Point Arithmetic theory) and a TRUNCATED RESULT becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

If the TRUE-VALUE of A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033[octal] or 539). If RESULT(MANTISSA) is<2(-27), then a FLOATING-POINT ZERO is forced as the result.

†TRUE-VALUE of A2(EXPONENT) must not exceed +27. (APPARENT-VALUE 539). If TRUE-VALUE of A2(EXPONENT) 28, the following RESULT will be obtained:

RESULT(EXPONENT)=A2(EXPONENT)minus "1"; RESULT(MANTISSA-SIGN)=A2(MANTISSA-SIGN), RESULT(MANTISSA Bits 01-26)=A2(MANTISSA Bits 02-27);RESULT(MANTISSA Bit 27)=0.

FLOATING ADDER GROUP (FADD)



VALUE

3 FSCLT FLOATING-POINT SCALE of A2; TRUNCATE

Assembler Format: FSCLT A2

Effect: Shift $A2^{MANTISSA}$ right and increment ($A2^{EXPONENT}$) until $A2_E$ = (SPFN + BIAS) - 1; result TRUNCATED.Converts an FPN to a 28-Bit Two's Complement integer within a dynamic range of 2↑27.

Description: The contents of A2(MANTISSA) are SCALED in the following manner:

1) Current SPFN(Bits 06-15) plus BIAS(=512) are forced into A1(EXPONENT). A1(MANTISSA)=0†. A2 contains the selected argument to be SCALED.††

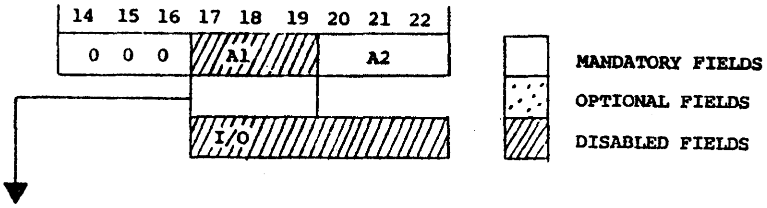
2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

4) The CONVERGENT-ROUNDING logic is inhibited for this operation and a TRUNCATED RESULT becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY).

If the TRUE-VALUE OF A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033[octal] or 539). If RESULT(MANTISSA) is<2(-28), then a FLOATING-POINT ZERO is forced as the result.

†Current SPFN(Bits 06-15) must equal maximum A2 Exponent plus "1" in order to obtain a correct result from this operation.
††TRUE-VALUE of A2(EXPONENT) must not exceed the value of current SPFN minus "1." If it does, the result obtained will be the same as in the case of FIX or FIXT when A2(EXP)≥ 28.



VALUE

4 FSM2C FORMAT-CONVERSION; A2 from SIGNED-MAGNITUDE to TWO'S COMPLEMENT†

Assembler Format: FSM2C A2

Effect: Converts (A2) from SIGNED-MAGNITUDE to TWOS-COMPLEMENT.

Description: The contents of A2 REGISTER are converted from SIGNED-MAGNITUDE to TWOS-COMPLEMENT format in the following manner:

1) If A2(MANTISSA) is negative, then A2(MANTISSA bit 00) (MANTISSA-SIGN) remains unchanged while A2(MANTISSA Bits 01-27) undergo a TWOS-COMPLEMENT conversion.

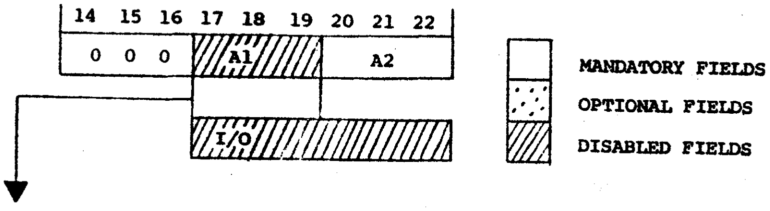
2) If A2(MANTISSA)is positive, A2(MANTISSA) is unchanged.

The normalized RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group operation. This operation can result in Floating Point underflow if the exponent is large and negative and the mantissa unnormalized. (See FADDR SUMMARY).

†See FLOATING-POINT THEORY, Types of notation.

This operation can result in FLOATING-POINT underflow if the exponent is large and negative and the mantissa unnormalized.

FLOATING ADDER 1 FIELD (FADD)



'ALUE

F2CSM

FORMAT CONVERSION; A2 from TWOS-COMPLEMENT to SIGNED-MAGNITUDE.†

Assembler Format: F2CSM A2

Effect: Convert (A2) from TWOS-COMPLEMENT to SIGNED-MAGNITUDE.

Description: The contents of A2 REGISTER are converted to SIGN-MAGNITUDE format in the following manner:

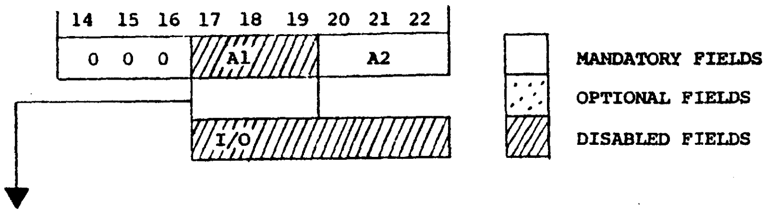
1) If A2(MANTISSA)is positive, A2(MANTISSA)is unchanged.

2) If A2(MANTISSA)is negative, then A2(MANTISSA Bit 00) (MANTISSA-SIGN) remains unchanged while A2(MANTISSA Bits 01-27) undergo a TWOS-COMPLEMENT conversion.

The normalized and convergently rounded RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY). Both underflow and overflow are possible as a result of this operation.

† See FLOATING-POINT SUMMARY, Types of Notation

FLOATING ADDER 1 FIELD (FADD)



VALUE

6 FSCALE

FLOATING-POINT SCALE of A2 (rounded result)

Assembler Format:

Shift $(A2^{MANTISSA})$ right and increment $(A2^{EXPONENT})$ until $(A2^{EXPONENT}) = (SPFN)$ + BIAS -1

Description: The contents of A2 are SCALED in the following manner:

1) Current SPFN(Bits 06-15)plus BIAS(=512)are forced into A1(EXPONENT). A1(MANTISSA)=0†. A2 contains the selected argument to be SCALED.††

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

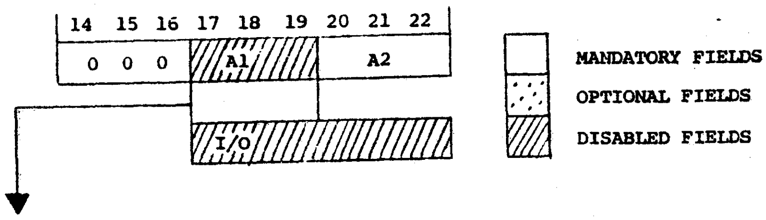
3) The PRELIMINARY-RESULT(EXPONENT)is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA)is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

4) The PRELIMINARY-RESULT is then CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

If RESULT(MANTISSA)is $\leq 2(-28)$, then a FLOATING-POINT ZERO is forced as the result.

† Current SPFN(Bits 06-15)must equal maximum plus "1" in order to obtain a correct result from this operation.

†† TRUE-VALUE OF A2(EXPONENT)must not exceed the value of current SPFN minus "1."



'ALUE

7 FABS FORMAT CONVERSION; A2 to ABSOLUTE VALUE.

Assembler Format: FABS A2

Effect: (A2) → ABSOLUTE VALUE

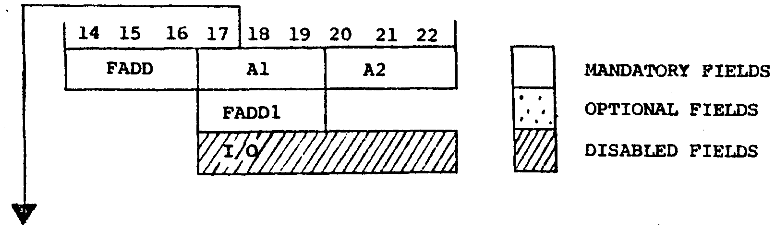
Description: The contents of A2 are converted to ABSOLUTE VALUE format in the following manner:

1) If A2(MANTISSA)is positive, A2(MANTISSA) is unchanged. A2 becomes the PRELIMINARY RESULT.

2) If A2(MANTISSA)is negative, then A2(MANTISSA Bits 00-27) undergo a TWOS-COMPLEMENT conversion. (See FLOATING-POINT SUMMARY -- Types of Notation).

The PRELIMINARY RESULT is then normalized and CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY). Both overflow and underflow are possible as a result of this operation.

FLOATING-ADDER GROUP (A1)



FLOATING ADDER REGISTER INPUT SUMMARY
A1

VALUE

Ø NC

NO CHANGE: The contents of A1 during the last FADDR operation are used as the A1 REGISTER OPERAND.

Assembler Format: FADD NC, <A2>Note: If no A1 operand is specified, NC is implied.

Effect: A1 is unaltered from previous operation.

1 FM

The current FLOATING-MULTIPLIER OUTPUT (FM) is the A1 REGISTER OPERAND.

Assembler Format: FSUB FM, <A2>

Effect: (FM) → A1

2 DPX (idx)

$\text{DATA PAD X}_{(\text{DPA} + \text{XR} -4)}$ is the A1 REGISTER OPERAND.

Assembler Format: FSUBR DPX (idx), <A2>

Effect: $[\text{DPX}_{(\text{DPA} + \text{XR} -4)}] \rightarrow \text{A1}$

3 DPY (idx)

$\text{DATA PAD Y}_{(\text{DPA} + \text{YR} -4)}$ is A1 REGISTER OPERAND.

Assembler Format: FOR DPY (idx), <A2>

Effect: $[\text{DPY}_{(\text{DPA} + \text{YR} -4)}] \rightarrow \text{A1}$

4 TM

The current contents of the TABLE MEMORY REGISTER (TMREG) is the A1 REGISTER OPERAND (See MEMORY GROUP SUMMARY - TM)

Assembler Format: FAND TM, <A2>

Effect: (TMREG) → A1

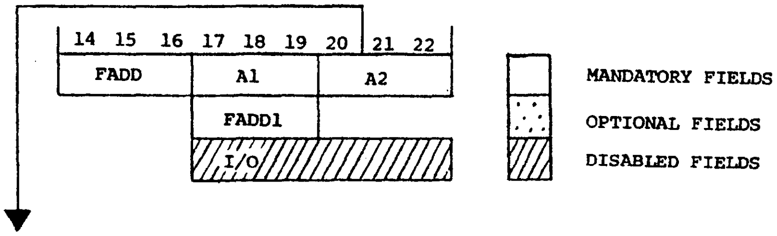
5,6 and 7 ZERO

FLOATING-POINT ZERO (0.0) is the A1 REGISTER OPERAND.

Assembler Format: FEQV ZERO, <A2>

Effect: 0.0 → A1

FLOATING ADDER GROUP



FLOATING ADDER REGISTER INPUT SUMMARY

A2

OCTAL VALUE

Ø NC

The contents of A2 during the last FADDR operation are used as the A2 REGISTER operand.

Assembler Format: FADD <A1>, NC

Effect: (A2) is unaltered from previous operation.

1 FA

The current FLOATING ADDER OUTPUT (FA) is the A2 REGISTER operand.

Assembler Format: FIX FA

Effect: (FA) → A2

2 DPX (idx)

DATA PAD X $_{\text{(DPA + XR -4)}}$ is the A2 REGISTER operand.

Assembler Format: FSCLT DPX(idx)

Effect: $[\text{DPX}_{\text{(DPA + XR -4)}}] \rightarrow \text{A2}$

3 DPY (idx)

DATA PAD Y $_{\text{(DPA + YR -4)}}$ is the A2 REGISTER operand.

Assembler Format: FABS DPY(idx)

Effect: $[\text{DPX}_{\text{(DPA + YR -4)}}] \rightarrow \text{A2}$

4 MD

The current contents of the MAIN DATA MEMORY OUTPUT REGISTER (MDREG) are used as the A2 REGISTER operand. (See MEMORY GROUP SUMMARY - MD).

Assembler Format: F2CSM MD

Effect: (MDREG) → A2

5 ZERO

FLOATING POINT ZERO (0.0) is the A2 REGISTER operand.

Assembler Format: FADD <A1>, ZERO

Effect: 0.0 → A2

6 MDPX (idx)

"Split-word" transfer to A2. (mantissa of DPX)

(1) The SPAD FUNCTION (SPFN) plus the BIAS-VALUE (512) forms the EXPONENT portion of the A2 OPERAND.

(2) The MANTISSA portion of DATA PAD $X_{(DPA + XR -4)}$ forms the MANTISSA portion of the A2 OPERAND.

Assembler Format: FSUB <A1>, MDPX (idx)

Effect: (SPFN) + 512 → $A2^{EXPONENT}$;

$[DPX_{(DPA + XR -4)}] \rightarrow A2^{MANTISSA}$

7 EDPX (idx)

" Split-word" transfer to A2. (exponent of DPX)

(1) The EXPONENT of DATA PAD $X_{(DPA + XR -4)}$ forms the EXPONENT portion of the A2 OPERAND.

(2) The 2 least-significant bits of SPAD FUNCTION $_{(SPFN\ bits\ 14,15)}$ are put into $A2^{MANTISSA}$ bits ØØ, 01. The remainder of $A2^{MANTISSA}$ is zeroed.

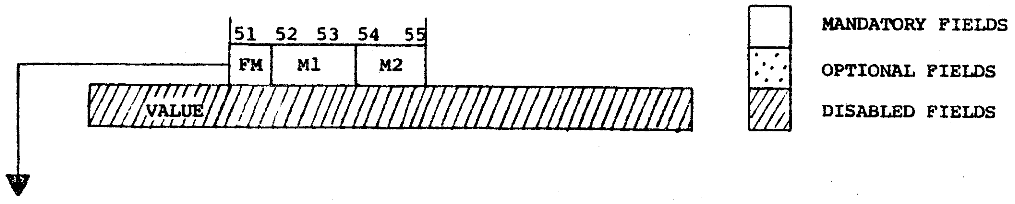
EDPX (idx) is used to generate either a $\pm\frac{1}{2}$ or -1 MANTISSA value.

Assembler Format: FSUBR <A1>, EDPX (idx)

Effect: $[DPX_{(DPA + XR - 4)}^{EXPONENT}] \rightarrow A2^{EXPONENT}$;

$(SPFN_{Bits\ 14,\ 15}) \rightarrow A2^{MANTISSA}$ bits ØØ, 01;

Øs → $A2^{MANTISSA}$ bits 01-27.



VALUE

Ø No Operation

1 **FMUL** FLOATING POINT MULTIPLY

Assembler Format: FMUL <M1,M2>[†]

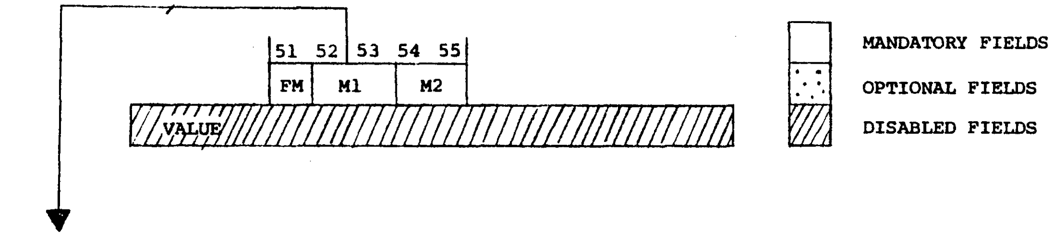
Effect: (M1) * (M2)

Description: A FLOATING POINT MULTIPLY (FMUL) is initiated using the operands selected by the M1 and M2 fields.

The CONVERGENTLY-ROUNDED result becomes available as FM 1 cycle after it has been "pushed" through the 3 stage pipeline by two subsequent FMUL operations. (See, FMULR SUMMARY).

[†] M1 and M2 operands need not be specified if a "dummy" is desired.

M1 REGISTER SUMMARY



VALUE

Ø FM

FLOATING MULTIPLIER OUTPUT is the M1 REGISTER OPERAND

Assembler Format: FMUL FM,<M2>

Effect: (FM) → M1

1 DPX(idx)

DATA PAD X is the M1 REGISTER OPERAND

Assembler Format: FMUL DPX(idx),<M2>

Effect: $(DPX_{(DPA + idx)}) \rightarrow M1$

2 DPY(idx)

DATA PAD Y is the M1 REGISTER OPERAND

Assembler Format: FMUL DPY(idx),<M2>

Effect: $(DPY_{(DPA + idx)}) \rightarrow M1$

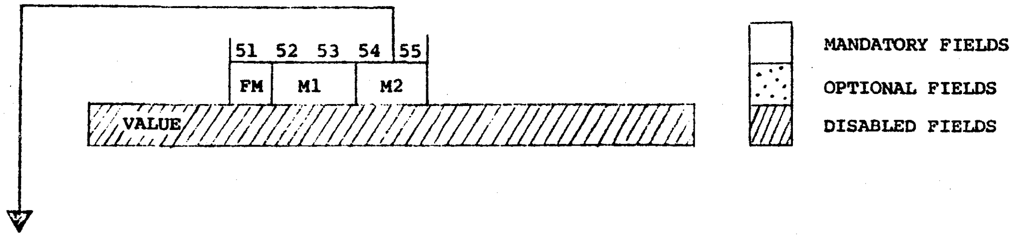
3 TM

TABLE MEMORY OUTPUT REGISTER is the M1 REGISTER OPERAND

Assembler Format: FMUL TM,<M2>

Effect: (TMREG) → M1

M2 REGISTER SUMMARY



VALUE

Ø **FA**

FLOATING ADDER OUTPUT is the M2 REGISTER INPUT

Assembler Format: FMUL<M1>,FA

Effect: (FA) → M2

1 **DPX(idx)**

DATA PAD X as M2 REGISTER INPUT

Assembler Format: FMUL<M1>, DPX(idx)

Effect: $(DPX_{(DPA\ +\ idx)}) \rightarrow M2$

2 **DPY(idx)**

DATA PAD Y is the M2 REGISTER INPUT

Assembler Format: FMUL <M1>, DPY(idx)

Effect: $(DPY_{(DPA\ +\ idx)}) \rightarrow M2$

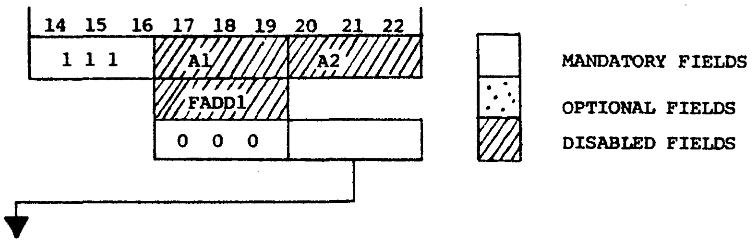
3 **MD**

MAIN DATA MEMORY REGISTER is the M2 REGISTER INPUT

Assembler Format: FMUL<M1>, MD

Effect: (MDREG) → M2

LOAD REGISTER FIELD (LDREG)



VALUE

Ø No Operation

1 LDSPD LOAD S-PAD DESTINATION ADDRESS

Assembler
Format: LDSPD

Example: LDSPD;DB=DPX(-3)

Effect: (DPBS$^{\text{MANTISSA}}$ bits 24-27)→ SPD

Description: DPBS(MANTISSA)bits 24-27 replace the contents of the four-bit S-PAD DESTINATION ADDRESS REGISTER (SPD) as of the next instruction cycle.

NOTE: A current S-PAD operation is unaffected by this instruction. However, if an S-PAD operation is executed on the next instruction cycle, the assembled S-PAD DESTINATION ADDRESS (SPD) to be used in the S-PAD operation will be replaced with the contents of SPD produced as a result of this instruction.

2 LDMA LOAD MEMORY ADDRESS from the DATA PAD BUS;
INITIATE A MEMORY CYCLE

Assembler
Format: LDMA

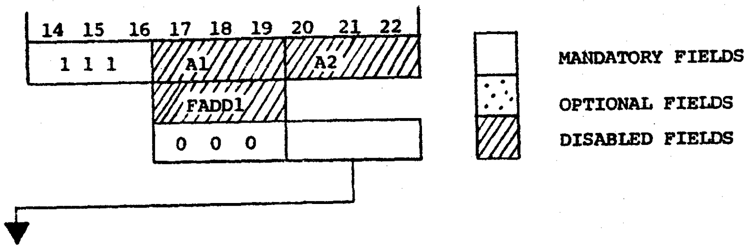
Example: LDMA;DB=DPX(-1)

Effect: (DPBS$^{\text{MANTISSA}}$ bits 12-27)→MA

Description: DPBS(LOW MANTISSA) bits 12-27 are loaded into the MAIN DATA MEMORY ADDRESS REGISTER (MA) effective as of the next instruction cycle. A MAIN DATA (MD) MEMORY cycle is initiated using the new contents of MA. (See MEMORY GROUP SUMMARY.)

NOTE: This op-code supersedes INCMA and DECMA in the same instruction. It makes SETMA redundant since it would now load from DB instead of SPFN due to the use of the LDREG field.

LOAD REGISTER FIELD (LDREG)



VALUE

3 LDTMA

LOAD TABLE MEMORY ADDRESS from DATA PAD BUS

Assembler Format: LDTMA

Example: LDTMA;DB=DPY(-4)

Effect: (DPBS$^{\text{MANTISSA}}$ bits 12-27) → TMA

Description: The DATA PAD BUS(LOW MANTISSA) (DPBS[MANTISSA]bits 12-27) is loaded into the TABLE MEMORY ADDRESS REGISTER (TMA) effective as of the next instruction cycle.

Two cycles later the contents of the TABLE MEMORY location specified by the two new contents of TMA will become available as the contents of TABLE MEMORY OUTPUT REGISTER (TMREG).

NOTE: This op-code supersedes INCTMA and DECTMA in the same instruction. It makes SETTMA redundant.

4 LDDPA

LOAD DATA PAD ADDRESS from DATA PAD BUS

Assembler Format: LDDPA

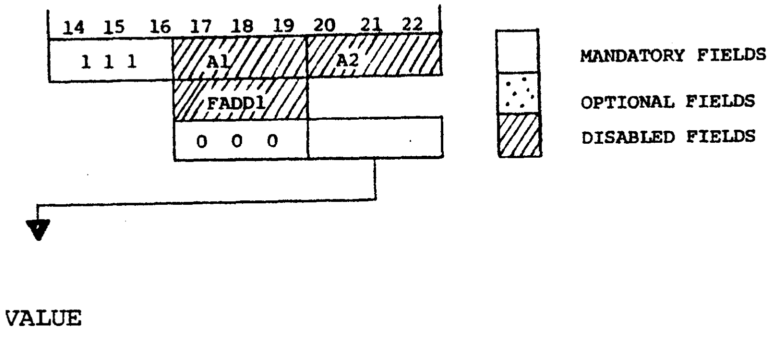
Example: LDDPA;DB=DPX(3)

Effect: (DPBS$^{\text{MANTISSA}}$ bits 21-27) → DPA

Description: The contents enabled onto the DATA PAD BUS(MANTISSA)bits 12-27 are loaded into the DATA PAD ADDRESS REGISTER (DPA). The change in DPA is effective as of the next instruction cycle.

NOTE: LDDPA supersedes INCDPA and DECDPA. It makes SETDPA redundant.

LOAD REGISTER FIELD (LDREG)



5 LDSP LOAD S-PAD DESTINATION REGISTER from DATA PAD BUS

Assembler Format: LDSP

Example: LDSP;DB=VALUE

Effect: $(\text{DPBS}^{\text{MANTISSA}}$ bits 12-27$)\rightarrow \text{SP}_{\text{SPD}}$

Description: The data currently enabled onto the DATA PAD BUS(MANTISSA)bits 12-27 are loaded into SP(SPD).

SPD is selected either via current S-PAD operation or a preceding LDSPD.

NOTE: LDSP supersedes LDSPNL, LDSPE and LDSPT. It makes LDSPI redundant. However, one of these op-codes could be used to select SPD if the immediately preceding instruction was not an LDSPD.

When combined with an S-PAD operation, LDSP results in the inclusive OR of SPFN and DPBS[MANTISSA](12-27) being written into SP(SPD).

6 LDAPS LOAD APSTATUS REGISTER from DATA PAD BUS

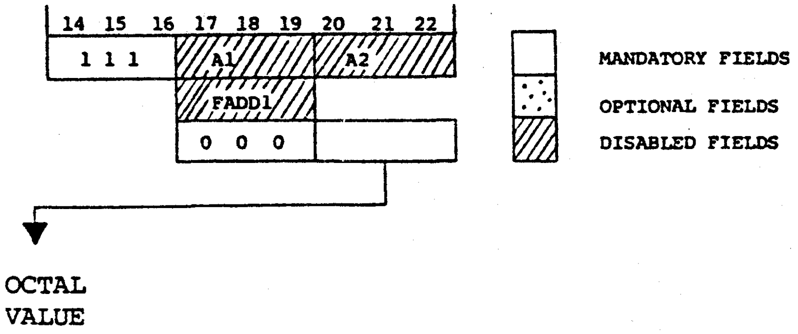
Assmebler Format: LDAPS

Example: LDAPS;DB=DPY(2)

Effect: $(\text{DPBS}^{\text{MANTISSA}}$ bits 12-27$)$ -- APSTATUS

Description: The data currently enabled onto the DATA PAD BUS(MANTISSA) bits 12-27 are loaded into the APSTATUS REGISTER (APSTATUS). The new contents of APSTATUS may be tested. Two cycles later, i.e., at least one cycle must intervene between the LDAPS and a related test. Refer to the I/O Group Summary for a complete description of the effects of LDAPS.

LOAD REGISTER FIELD (LDREG)



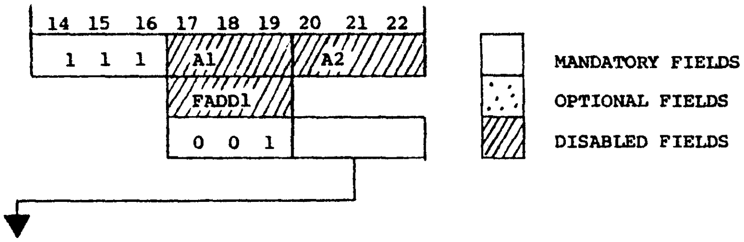
7 LDDA LOAD DEVICE ADDRESS from DATA PAD BUS.

Assembler Format: LDDA

Example: LDDA;DB=VALUE

Effect: The least significant 8 bits of the data currently enabled onto the DATA PAD BUS (DPBS) (DATA PAD BUS$^{\text{MANTISSA}}$ bits 2Ø-27) are loaded into the 8 bit DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

READ REGISTER FIELD (RDREG)



VALUE

Ø RPSA READ PROGRAM SOURCE ADDRESS onto the PANEL BUS.

Assembler Format: RPSA

Effect: (PSA)→ PNLBS

Description: The contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA) are enabled onto the PANEL BUS (bits Ø4-15). The value of PSA READ is the absolute address of this instruction.

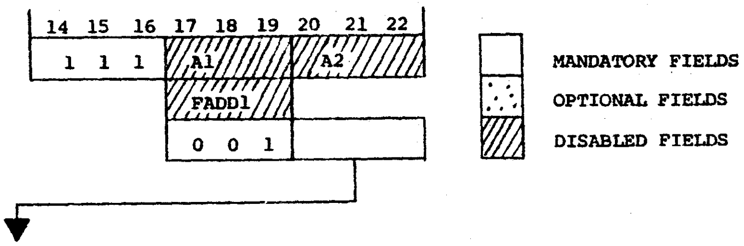
1 RSPD READ SPAD DESTINATION ADDRESS REGISTER onto the PANEL BUS.

Assembler Format: RSPD

Effect: (SPD)→ PNLBS

Description: The contents of the currently designated S-PAD DESTINATION ADDRESS (SPD) are enabled onto the PANEL BUS (PNLBS) (bits 12-15).

READ REGISTER FIELD (RDREG)



VALUE

2 RMA READ MEMORY ADDRESS REGISTER onto the PANEL BUS

Assembler
Format: RMA

Effect: (MA)→ PNLBS

Description: The contents of the 16-bit MEMORY ADDRESS REGISTER (MA) are enabled onto the PANEL BUS (PNLBS) (bits ØØ-15).

3 RTMA READ TABLE MEMORY ADDRESS onto the PANEL BUS

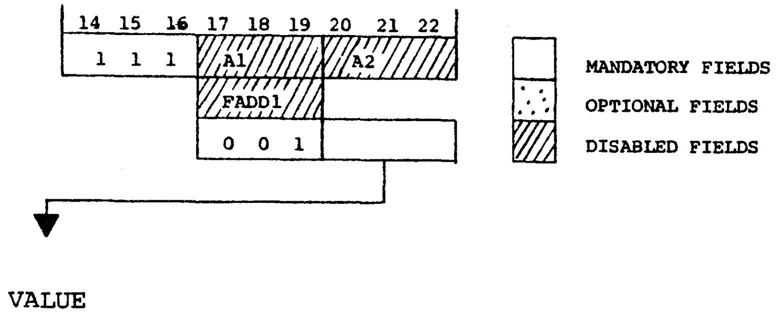
Assembler
Format: RTMA

Effect: (TMA)→PNLBS

Description: The contents of the 16-bit TABLE MEMORY ADDRESS (TMA) are enabled onto the PANEL BUS (PNLBS) (bits 00-15).

The value read by this instruction is the contents of TMA as modified by the FFT and IFFT Bits in the APSTATUS Register (APSTATUS Bits 11 and 12). If APSTATUS Bit 12 is a "0," then this instruction reads the unmodified contents of TMA. Thus, this instruction always reads the actual address being presented to TABLE MEMORY. This is distinct from the JMPT, JSRT, LPSLT, LPSRT, etc. op-codes in the SPEC group that always use the unmodified contents of TMA Register.

READ REGISTER FIELD (RDREG)



4 RDPA READ DATA PAD REGISTER onto the PANEL BUS

Assembler
Format: RDPA

Effect: (DPA)→ PNLBS

Description: The contents of the six-bit DATA PAD ADDRESS REGISTER (DPA) are enabled onto the PANEL BUS (PNLBS) (bits 10-15).

NOTE: DPA appears to be six bits wide for the purposes of LDDPA, SETDPA, INCDPA, DECDPA and RDPA. Only the least significant five bits, however, are effective in addressing DATA PAD.

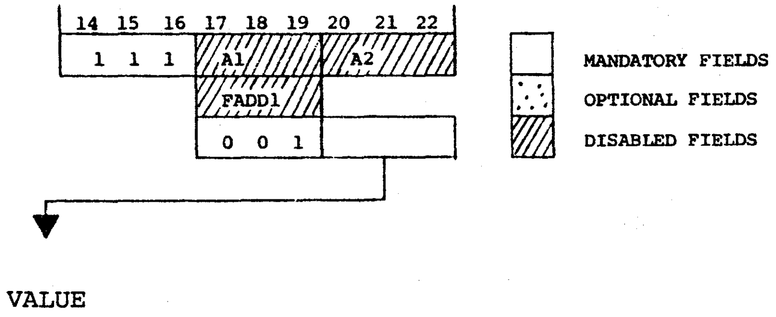
5 RSPFN READ S-PAD FUNCTION onto the PANEL BUS

Assembler
Format: RPSFN

Effect: (SPFN)→ PNLBS

Description: The result of the current S-PAD operation (SPFN) is enabled onto the PANEL BUS (bits ØØ-15) for this instruction cycle.

READ REGISTER FIELD (RDREG)



6 RAPS READ APSTATUS REGISTER onto the PANEL BUS

Assembler
Format: RAPS

Effect: (APSTATUS)→ PNLBS

Description: The contents of the 16-bit APSTATUS REGISTER (APSTATUS) are enabled onto the PANEL BUS (PNLBS) (bits ØØ-15) during the current instruction cycle.

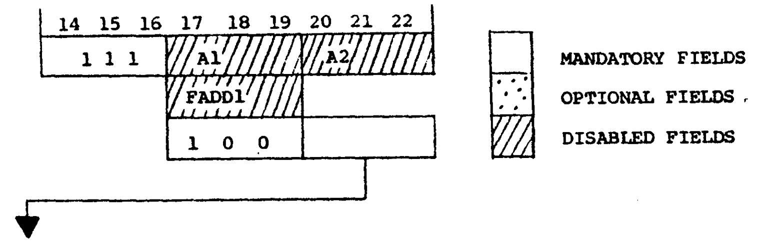
7 RDA READ DEVICE ADDRESS onto the PANEL BUS

Assembler
Format: RDA

Effect: (DA)→ PNLBS bits Ø8-15; Ø's→PNLBS bits ØØ-Ø7

Description: The current contents of the eight-bit DEVICE ADDRESS REGISTER (DA) are enabled onto the PANEL BUS (PNLBS), bits 08-15. Zeros are enabled onto the remaining left-most bits of PNLBS, bits 00-07.

INPUT/OUTPUT FIELD (IN/OUT)



VALUE

Ø **OUT** OUTPUT to I/O DEVICE

Assembler Format: OUT

Example: DB=DPX;OUT

Effect: (DPBS)→ INBS→ I/O DEVICE (DA)

Description: The data enabled onto the DATA PAD BUS (DPBS) during this instruction is enabled onto the INPUT/OUTPUT BUS (INBS) where it becomes available for output to the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA).

1 **SPNOUT** SPIN until I/O DEVICE READY;then OUTPUT DATA

Assembler Format: SPNOUT

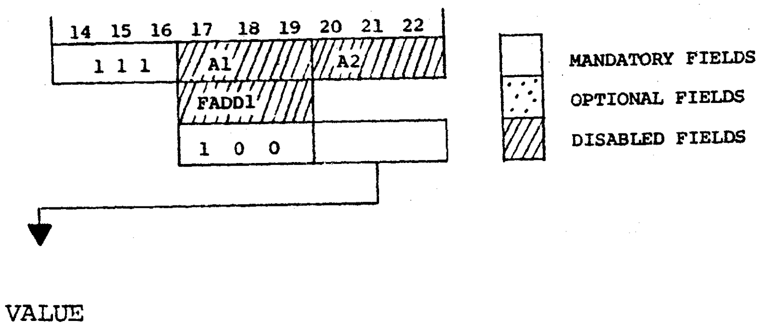
Example: DA=DPY(3);SPNOUT

Effect: SPIN UNTIL $\mathrm{IODRDY}_{(\mathrm{DA})}$ = 1; then, (DPBS)→ INBS→ $\text{I/O DEVICE}_{(\mathrm{DA})}$

Description: First, the condition of the I/O DATA READY FLAG (IODRDY) is tested. If IODRDY = 0, then a SPIN will occur. When IODRDY = 1, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS) where it becomes available for output to the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA).

NOTE: If IODRDY(DA) never equals "1", an infinite SPIN condition will occur. This can only be cleared by an Interface Reset.

INPUT/OUTPUT FIELD (IN/OUT)



2 OUTDA OUTPUT DATA; then SET DEVICE ADDRESS from SPFN

Assembler Format: OUTDA

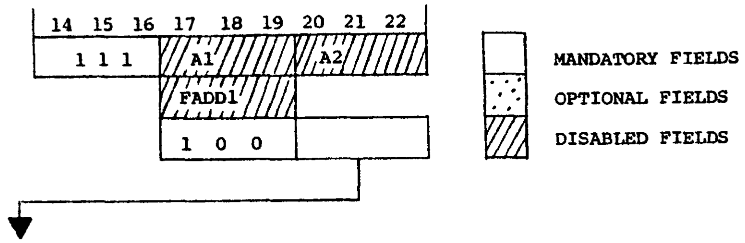
Example: MOV 5,5;DB=DPX(-2);OUTDA

Effect: (DPBS)→INBS→ IODEVICE$_{(DA)}$; then, (SPFN)→DA

Description: First, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS), where it becomes available for output to the I/O DEVICE specified by the current contents of the I/O DEVICE ADDRESS REGISTER (DA).

Then, bits Ø8-15 of the current SPAD operation result (SPFN) are loaded into the 8 bit I/O DEVICE ADDRESS REGISTER; effective as of the next instruction cycle.

INPUT/OUTPUT FIELD (IN/OUT)



'ALUE

3	SPOTDA	SPIN until I/O DEVICE is READY; then OUTPUT DATA; then SET DEVICE ADDRESS from SPFN.

Assembler Format: SPOTDA

Example: MOV 3,3;DB=MD;SPOTDA

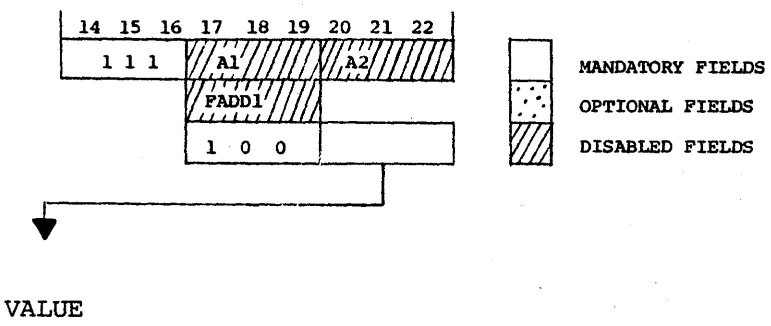
Effect: SPIN until IODRDY (DA)=1; then (DPBS)→INBS→I/O DEVICE$_{(DA)}$;then (SPFN)→DA

Description: First, the condition of the I/O DATA READY FLAG (IODRDY[DA]) is tested. If IODRDY(DA)=0, then a SPIN will occur until IODRDY(DA)=1. When IODRDY(DA)=1, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS) for output to the I/O DEVICE specified by the current contents of the I/O DEVICE ADDRESS REGISTER (DA).

Then, the eight right-most bits (bits 08-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

NOTE: If IODRDY(DA) never equals "1", then an infinite SPIN will occur.

INPUT/OUTPUT FIELD (IN/OUT)



4 IN — INPUT DATA from I/O DEVICE$_{(DA)}$

Assembler
Format: IN

Example: IN;DPX(2)<DB; DB = INBS

Effect: I/O DEVICE$_{(DA)}$ → INBS

Description: The data input from the I/O DEVICE specified by the I/O DEVICE ADDRESS RGISTER (DA) is enabled onto the INPUT/OUTPUT BUS (INBS).

NOTE: In order to be used internally by the AP-120B, the data input onto the INBS must be enabled onto the DPBS. This is achieved by a concurrent software instruction to that effect, such as in the assembler format example above. (Note that the assembler would equate the form DPX(2)<INBS to the two instructions shown above).

5 SPININ — SPIN UNTIL I/O DEVICE is READY; then INPUT DATA.

Assembler
Format: SPININ

Example: SPININ;DPX<INBS

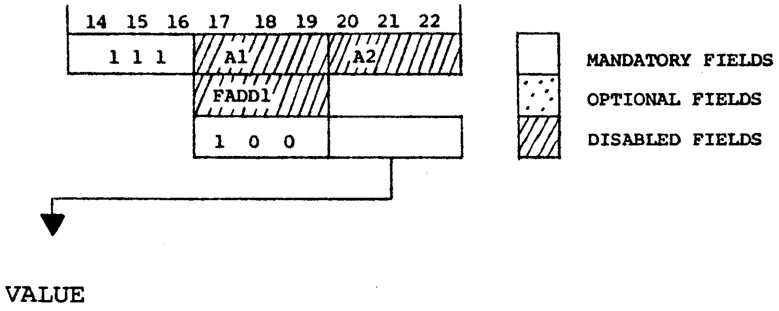
Effect: SPIN until IODRDY$_{(DA)}$=1; then (I/O DEVICE$_{(DA)}$) → INBS

Description: First, a SPIN is executed until the I/O DATA READY FLAG (IODRDY), for the I/O DEVICE SPECIFIED by the I/O DEVICE ADDRESS REGISTER (DA), is equal to "1."

Then, (when IODRDY[DA]=1), the data from the I/O DEVICE (DA) is enabled onto the INPUT/OUTPUT BUS (INBS).

NOTE: If IODRDY (DA) never equals "1," an infinite SPIN loop will occur.

INPUT/OUTPUT FIELD (IN/OUT)



6 INDA INPUT DATA; then SET DEVICE ADDRESS from SPFN

Assembler
Format: INDA

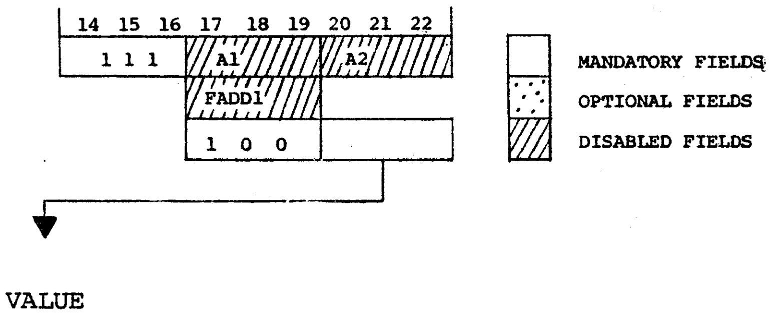
Example: MOV 5,5;INDA;DPY<INBS

Effect: (I/O DEVICE$_{(DA)}$)→ INBS; then (SPFN)→DA .

Description: First, the data input from the I/O DEVICE specified by the current contents of the I/O DEVICE REGISTER (DA), is enabled onto the INPUT/OUTPUT BUS (INBS).

Then, bits 08-15 of the current S-PAD operation result (SPFN) is set into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

INPUT/OUTPUT FIELD (IN/OUT)



7 SPINDA

SPIN UNTIL I/O DEVICE READY; then INPUT DATA; then SET DEVICE ADDRESS from SPFN.

Assembler Format: SPINDA

Example: MOV 6,6;SPINDA;DB=INBS

Effect: SPIN until IODRDY=1, then (I/O DEVICE$_{(DA)}$) → INBS; then (SPFN) → DA

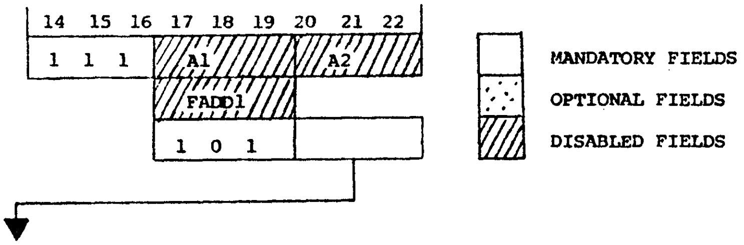
Description: First, a SPIN is executed until the I/O DATA READY FLAG(IODRDY) of the I/O DEVICE specified by the contents of the I/O DEVICE ADDRESS REGISTER (DA) is equal to "1."

When IODRDY(DA)=1, the data from the I/O DEVICE(DA) is input onto the INPUT/OUTPUT BUS (INBS).

Then the right-most bits of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

If IODRDY (DA) never equals "1," an infinite SPIN loop will be incurred.

I/O SENSE FIELD (SENSE)



VALUE

Ø **SNSA** SENSE "A"; set $IODRDY_{(DA)}$ from "A"

Assembler Format: SNSA

Effect: $(A_{(DA)}) \rightarrow IODRDY_{(DA)}$

Description: CONDITION "A" of the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA) will be "sensed" and the I/O DATA READY FLAG (IODRDY[DA]) will be set to reflect the state of "A" (either "1" or "0"). These conditions are device dependent and may not necessarily be implemented for all I/O devices. Refer to the appropriate device manual for further information. Note that some devices (e.g., PIOP) use the SNSA instruction to perform control functions.

1 **SPINA** SPIN UNTIL "A" NON-ZERO

Assembler Format: SPINA

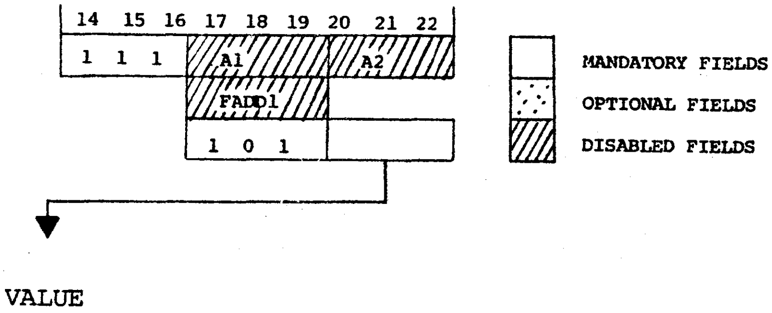
Effect: $(A_{(DA)}) \rightarrow IODRDY_{(DA)}$; if $IODRDY_{(DA)}=Ø$ SPIN UNTIL $IODRDY_{(DA)}=1$

Description: First, CONDITION "A" of I/O DEVICE(DA) is sensed and the I/O DATA READY FLAG (IODRDY) is set accordingly (to either "1" or "0").

If IODRDY(DA) equals "0," a SPIN condition will occur and will continue until condition "A" is equal to "1."

NOTE: If "A" never equals "1," an infinite SPIN will occur.

I/O SENSE FIELD (SENSE)



2 SNSADA

SENSE "A" SET to I/O READY FLAG; then set DEVICE ADDRESS from SPFN.

Assembler Format: SNSADA

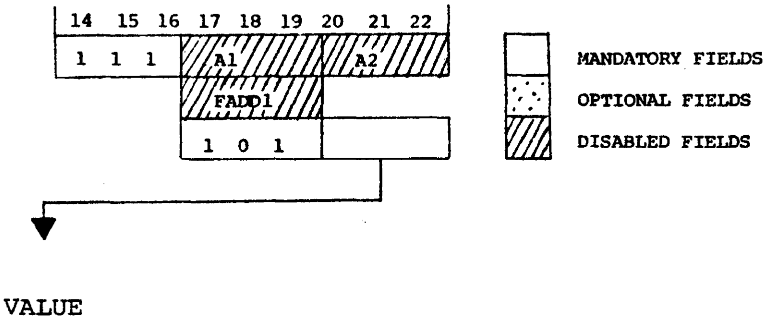
Example: MOV 5,5;SNSADA

Effect: $(A_{(DA)}) \rightarrow \mathrm{IODRDY}_{(DA)}$; then (SPFN) → DA

Description: First, CONDITION (A) for the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA) is sensed and its content (either "1" or "Ø") is set into the I/O DEVICE DATA READY FLAG ($\mathrm{IODRDY}_{(DA)}$).

Then, the right-most bits (bits Ø8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

I/O SENSE FIELD (SENSE)



3 **SPNADA** SPIN UNTIL "A" NON-ZERO; then SET DEVICE ADDRESS from SPFN

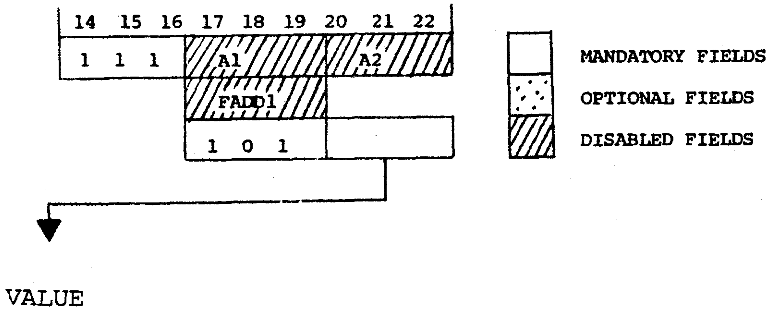
Assembler Format: SPNADA

Effect: $(A_{(DA)}) \rightarrow$ IODRDY$_{(DA)}$; if IODRDY$_{(DA)}$=Ø
SPIN until IODRDY=1; then (SPFN)→DA

Description: First, CONDITION "A" of I/O DEVICE$_{(DA)}$ is sensed and set into the I/O DATA READY FLAG (IODRDY$_{(DA)}$). If IODRDY then equals "Ø", a SPIN condition will occur until IODRDY$_{(DA)}$=1. At that time, the 8 right-most bits (bits Ø8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective next instruction cycle.

NOTE: An infinite "SPIN" will occur if "A" never equals "1".

I/O SENSE FIELD (SENSE)



4 SNSB SENSE I/O DEVICE CONDITION "B", SET $IODRDY_{(DA)}$ from "B"

Assembler Format: SNSB

Effect: $(B_{(DA)}) \rightarrow IODRDY_{(DA)}$

Description: CONDITION "B" (B) of the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA) will be "sensed" and the I/O DATA READY FLAG (IODRDY) for that I/O DEVICE will be set to reflect the contents of "B" (either "1" or "Ø").

5 SPINB SPIN UNTIL "B" is NON-ZERO

Assembler Format: SPINB

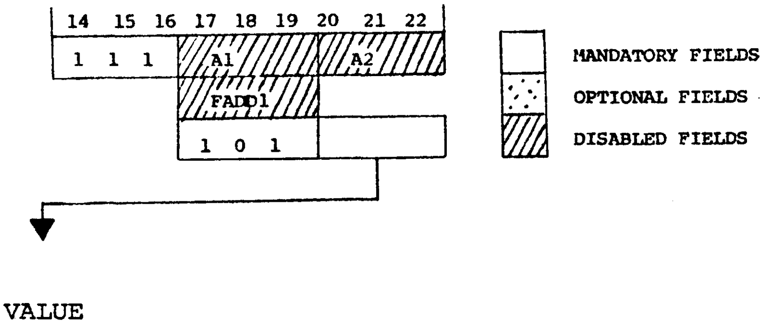
Effect: $(B_{(DA)}) \rightarrow IODRDY_{(DA)}$; if $IODRDY_{(DA)}$=Ø SPIN until $IODRDY_{(DA)}$=1.

Description: First, CONDITION "B" of I/O $DEVICE_{(DA)}$ is sensed and the I/O DATA READY FLAG (IODRDY) is set accordingly, (to either "1" or "Ø").

If $IODRDY_{(DA)}$ then equals "Ø", a SPIN condition will occur until "B" equals "1", at which time the $IODRDY_{(DA)}$ is set accordingly.

NOTE: If "B" never equals "1", an infinite SPIN condition will occur.

I/O SENSE FIELD (SENSE)



6 SNSBDA

SENSE "B" set to I/O DATA READY FLAG, then SET DEVICE ADDRESS from SPFN.

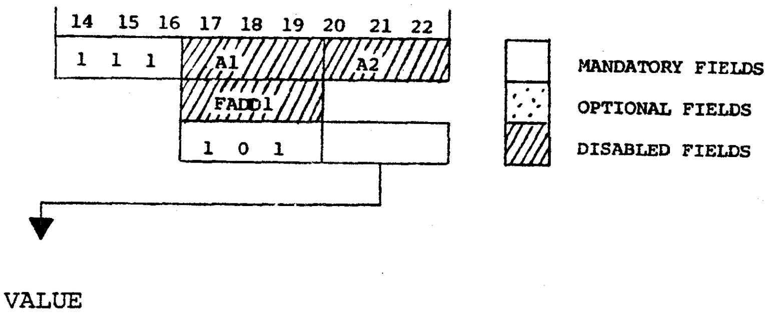
Assembler Format: SNSBDA

Effect: $(B_{(DA)}) \rightarrow IODRDY_{(DA)}$; then (SPFN)→ DA

Description: First, condition "B" of the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER$_{(DA)}$ is sensed and its content (either "1" or "Ø") set into the I/O DEVICE DATA READY FLAG (IODRDY$_{(DA)}$).

Then, the right-most bits (bits Ø8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective next instruction cycle.

I/O SENSE FIELD (SENSE)



7 SPNBDA SPIN UNTIL "B" is NON-ZERO; SET DEVICE ADDRESS from SPFN

Assembler Format: SPNBDA

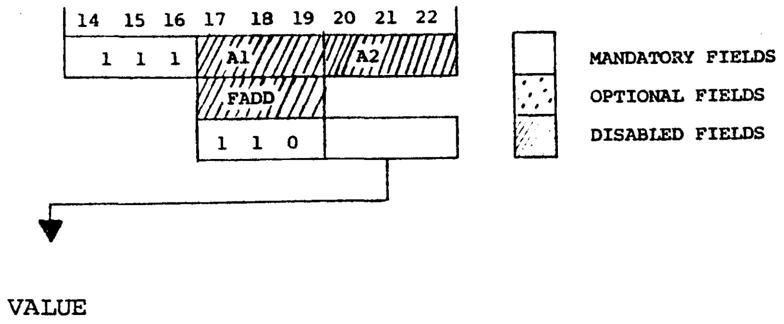
Effect: $(B_{(DA)}) \rightarrow IODRDY_{(DA)}$; if $IODRDY_{(DA)}$=Ø, SPIN until IODRDY=1; then (SPFN)→DA

Description: First, CONDITION "B" of I/O $DEVICE_{(DA)}$ is sensed and set into the I/O DATA READY FLAG ($IODRDY_{(DA)}$). If $IODRDY_{(DA)}$ then equals "Ø", a SPIN condition will occur until "B" equals "1", at which time the $IODRDY_{(DA)}$ is set accordingly.

Then the 8 right-most bits (bits Ø8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective next instruction cycle.

NOTE: If "B" never equals "1", an infinite SPIN condition will occur.

SET/CLEAR FLAG FIELD (FLAG)

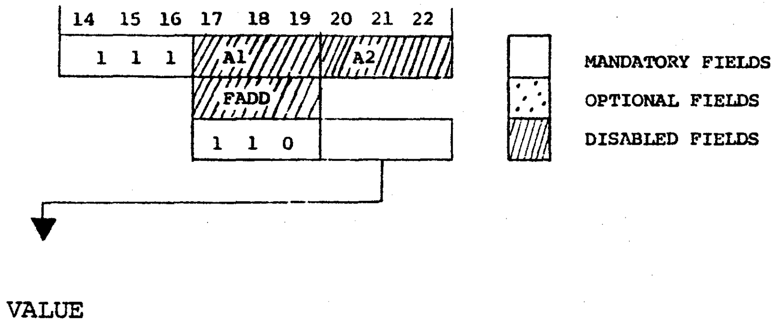


The following instructions set one of four GENERAL FLAGS (0,1,2,3) to a "1". These flags may be tested and branched upon accordingly by software instructions (see STEST SPEC).

Ø	SFLØ	Set GENERAL FLAG Ø to "1"
1	SFL1	Set GENERAL FLAG 1 to "1"
2	SFL2	Set GENERAL FLAG 2 to "1"
3	SFL3	Set GENERAL FLAG 3 to "1"
	Assembler Format:	SFLn
	Effect:	1→ FLAG_n

NOTE: Effective two instruction cycles later. At least one cycle must intervene between an SFLN and a Related Branch instruction.

SET/CLEAR FLAG FIELD (FLAG)



The following instructions clear to "0" one of the four GENERAL FLAGS (0,1,2,3) available for use in the AP. These flags may be tested and branched upon by software instructions (see STEST SPEC).

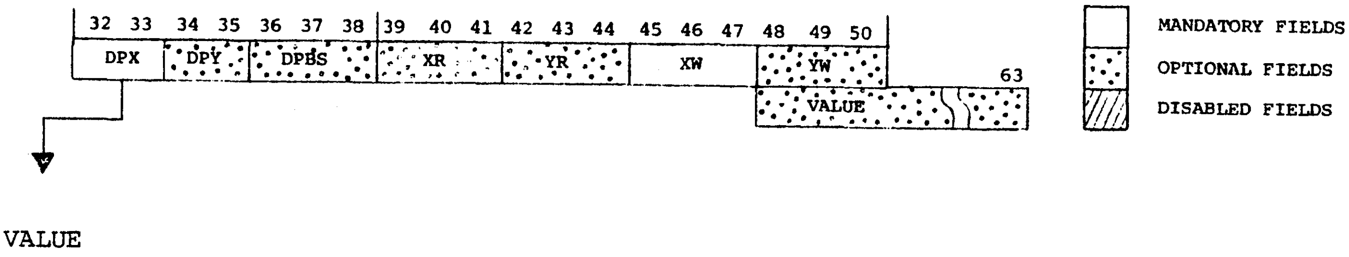
4	CLFØ	Clear GENERAL FLAG Ø to "Ø"
5	CLF1	Clear GENERAL FLAG 1 to "Ø"
6	CLF2	Clear GENERAL FLAG 2 to "Ø"
7	CLF3	Clear GENERAL FLAG 3 to "Ø"

Assembler Format: CLFn

Effect: $\emptyset \rightarrow FLAG_n$

NOTE: Effective two instruction cycles later.

DATA PAD X (DPX)



Ø No Operation

1 DPX(idx)[†] < DB STORE DATA PAD BUS into DATA PAD X

Assembler Format: DPX(idx) → DB

Example: DPX(-3) < DB; DB=MD

Effect: $(\mathrm{DPBS}) \rightarrow \mathrm{DPX}_{(\mathrm{DPA}) + \mathrm{idx}}$

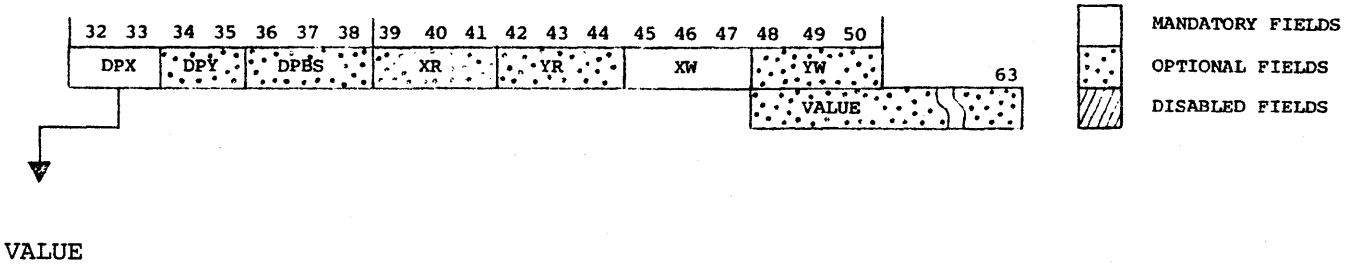
Description: The data currently enabled onto the DATA PAD BUS (DB) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the XWRITE FIELD (XW) minus 4 (the BIAS value).

Normally, a DPBS enable instruction is used concurrently with this instruction. If so, the two instructions can be expressed in shorthand notation (e.g., the example above can be expressed as follows: DPX[-3]<MD).

Note: All bits are written unless WRTEXP, WRTHMN , or WRTLMN is set.

[†] idx: An integer in a range from -4 to +3.
XW = idx + 4

DATA PAD X (DPX)



2 DPX(idx) < FA

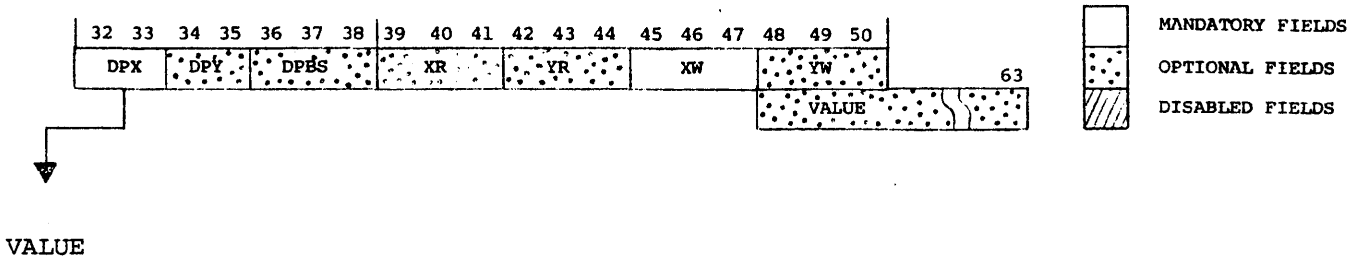
STORE FLOATING ADDER OUTPUT into DATA PAD X

Assembler Format: DPX(idx) < FA

Effect: $(FA) \rightarrow DPX_{(DPA) + idx - 4}$

Description: The current FLOATING ADDER result (FA) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.

DATA PAD X (DPX)



3 DPX(idx) < FM

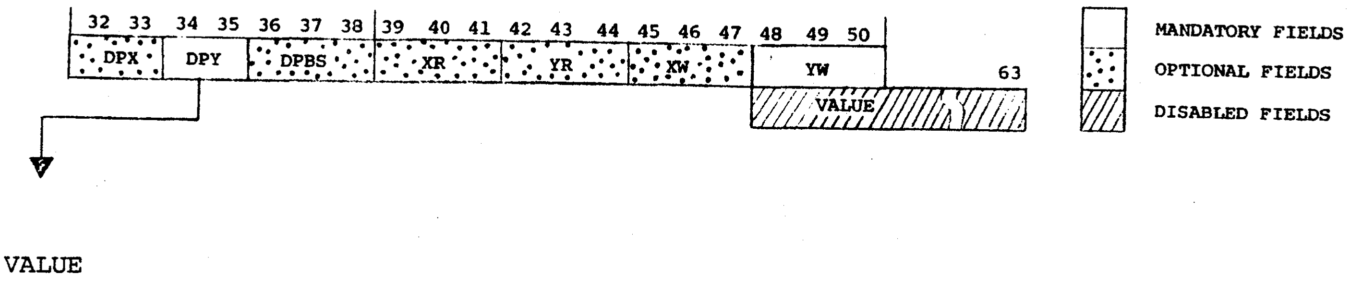
STORE FLOATING MULTIPLIER OUTPUT into DATA PAD X

Assembler Format: DPX(idx)< FM

Effect: $(FM) \rightarrow DPX_{(DPA) + idx}$

Description: The current FLOATING MULTIPLIER OUTPUT (FM) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.

DATA PAD Y (DPY)



1 DPY(idx)[†] < DB

STORE DATA PAD BUS into DATA PAD Y

Assembler Format: DPX(idx) < DB

Example: DPY(+3) < DB; DB=MD

Effect: (DB)→ DPY (DPA) + idx

Description: The data currently enabled onto the DATA PAD BUS (DB) is written into DATA PAD Y (DPY) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the Y WRITE FIELD (YW)[††] minus 4 (the BIAS value).

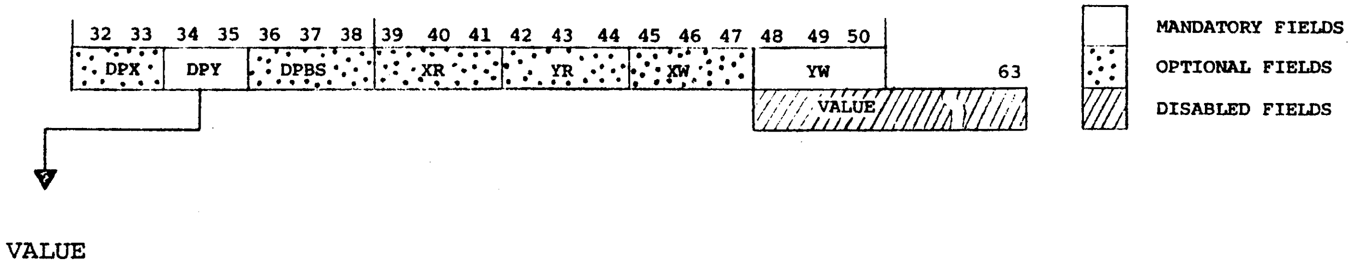
Normally, a DPBS enable instruction is used concurrently with this instruction. If so, the two instructions can be expressed in shorthand notaton (e.g., the example above can be expressed as follows: DPY[+3]<MD).

Note: All bits written unless WRTEXP, WRTHMN, or WRTLMN is set.

[†] idx: An integer in a range from -4 to +3.
YW = idx+4.

[††] If VALUE field is used during this instruction, the XW field is referenced instead. And if a DPX (idx) write instruction is used concurrently, the indices specified by the programmer must be equal. Errors in this regard are flagged by the assembler.

DATA PAD Y (DPY)



2 DPY(idx) < FA

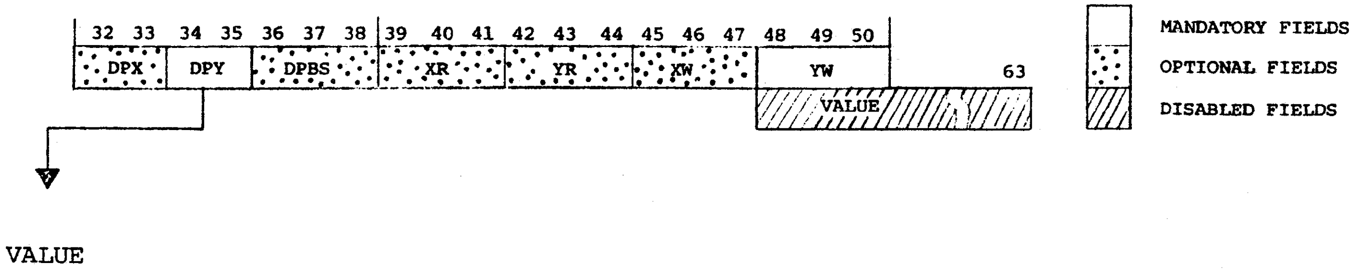
STORE FLOATING ADDER OUTPUT into DATA PAD Y

Assembler Format: DPY(idx)< FA

Effect: $(FA) \rightarrow DPY_{(DPA) + idx}$

Description: The current FLOATING ADDER OUTPUT (FA) is written into DATA PAD Y at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.

DATA PAD Y (DPY)



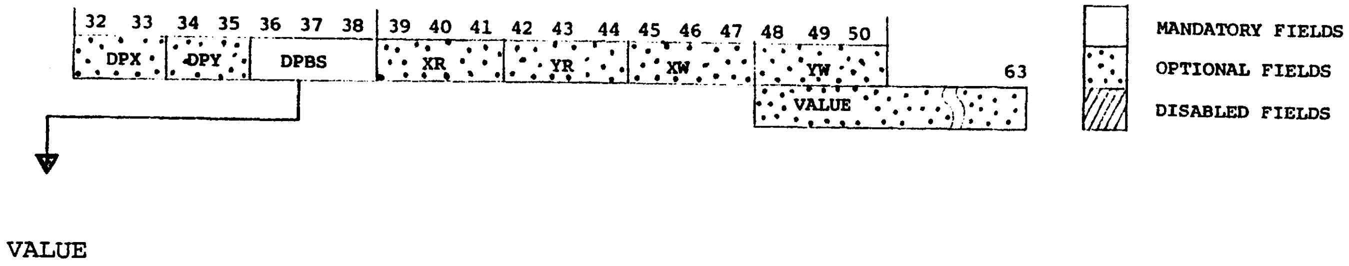
3 DPY(idx) < FM

STORE FLOATING MULTIPLIER OUTPUT into DATA PAD Y

Assembler Format: DPY(idx) < FM

Effect: $(FM) \rightarrow DPY_{(DPA)\ +\ idx}$

Description: The current contents of the FLOATING MULTIPLIER OUTPUT (FM) are written into DATA PAD Y (DPY) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.



Ø **DB = ZERO** ENABLE FLOATING POINT ZERO onto the DATA PAD BUS

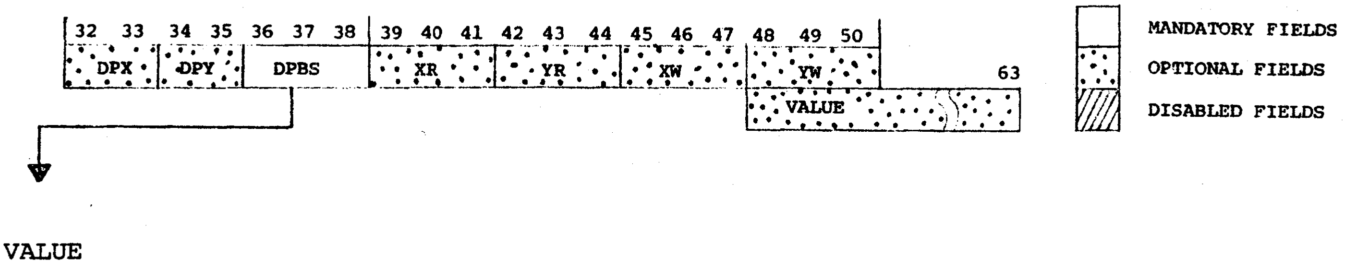
Assembler Format: DB = ZERO (see Data Pad summary)

Effect: 0.0 → DPBS

Description: FLOATING POINT ZERO (0.0) is enabled onto the DATA PAD BUS (DPBS) during the current instruction cycle.

Note that this is the default condition for the DPBS field. Thus, if DPBS is not specified via a "DB = " mnemonic, and no other DATA PAD BUS enable field (e.g. RPSF, see SPEC) is utilized, then there will be a zero on the DATA PAD BUS.

DATA PAD BUS (DPBS)



1 DB = INBS

ENABLE INBUS onto the DATA PAD BUS

Assembler Format: DB = INBS (see Data Pad summary)

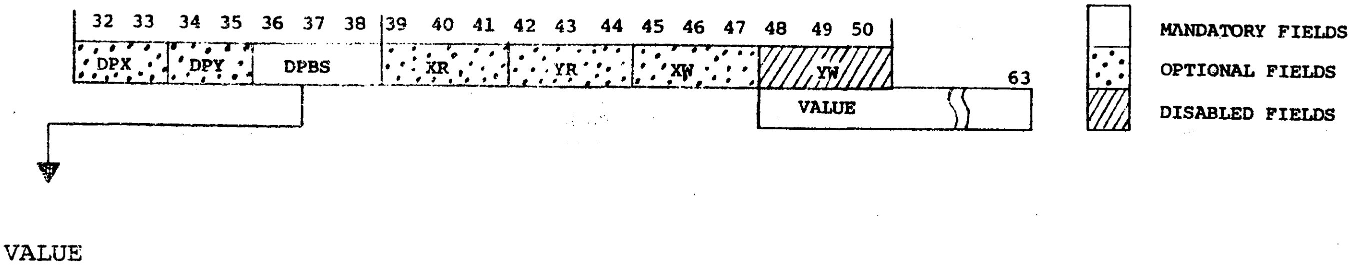
Example: IN: DPX(-3) DB; DB=INBS

Effect: (INBS)→ DPBS

Description: The contents of the 38-bit INPUT/OUTPUT BUS (INBS) are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

This instruction is used to transfer data from INBS to DPBS during a current input operation (See example, above).

It must be accompanied by an I/O group INPUT instruction (IN, SPININ, INDA or SPINDA).



2 $\boxed{\text{DB = VALUE}^{\dagger}}$ ENABLE VALUE onto the DATA PAD BUS

Assembler Format: DB = val (where val is 16 bit value contained in Value field).

Example: DB = 25
DB = -1

Effect: (VALUE bits 54-63) → $\text{DPBS}^{\text{EXPONENT}}$
(VALUE bits 49-63) → $\text{DPBS}^{\text{MANTISSA}}$ bits 13-27
(VALUE bit 48 (sign) → $\text{DPBS}^{\text{MANTISSA}}$ bits 00-12

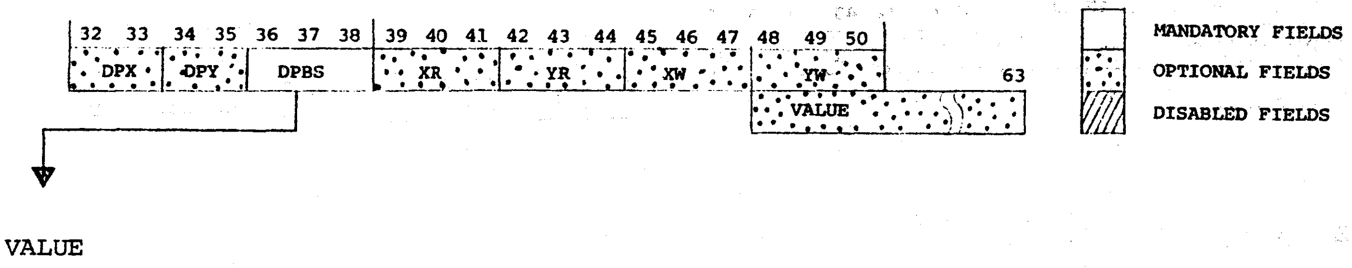
Description: The 16-bit value contained in the VALUE field of the instruction word is enabled onto the DATA PAD BUS (DPBS) during the current instruction cycle, in the following manner:

1) The right-most 15 bits of the VALUE field are enabled onto DPBS (MANTISSA) bits 13-27. The left-most remaining bit (the sign bit) is extended and enabled onto DPBS (MANTISSA) bits 00-12.

2) The right-most 10 bits of the VALUE field are enabled onto the DPBS (EXPONENT) bits 02-11.

† YW field is disabled for the current instruction cycle.

DATA PAD BUS (DPBS)



3 $\boxed{\text{DB = DPX(idx)}^{\dagger}}$ ENABLE DATA PAD X onto the DATA PAD BUS

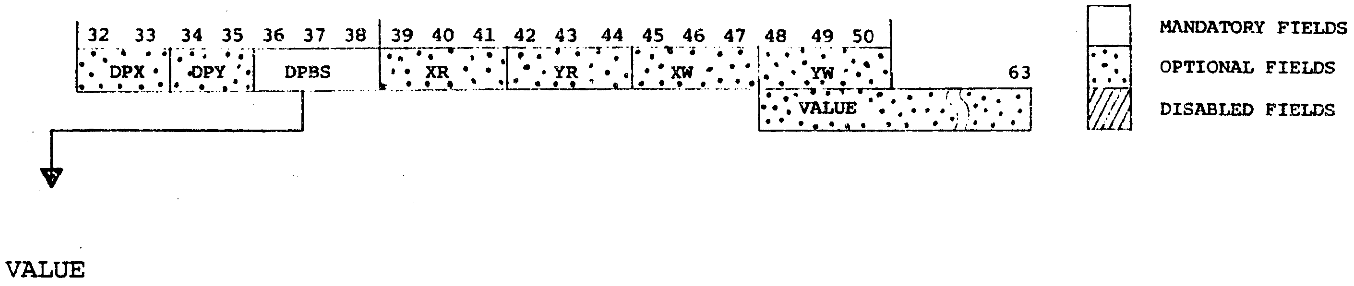
Assembler Format: DB = DPX(idx)

Effect: $(\text{DPX}_{(\text{DPA}) + \text{idx}}) \rightarrow \text{DPBS}$

Description: The contents of the DATA PAD X (DPX) location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPS), plus the contents of the X READ FIELD (XR) minus 4, are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

† idx: An integer in a range from -4 to +3.
XR = idx + 4

DATA PAD BUS (DPBS)



4 DB = DPY(idx)[†] ENABLE DATA PAD Y onto the DATA PAD BUS

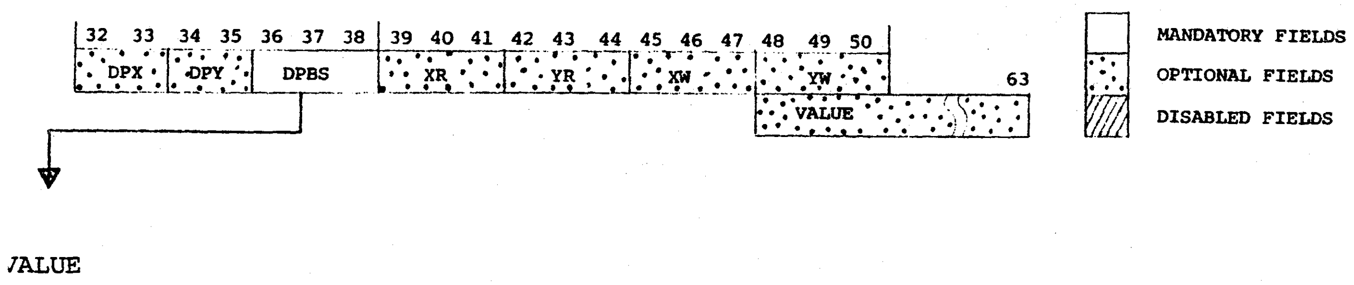
Assembler Format: DB = DPY(idx)

Effect: $(DPY_{(DPA) + idx}) \rightarrow DPBS$

Description: The contents of the DATA PAD Y (DPY) location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the Y READ FIELD (YR) minus 4, are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

[†] idx: An integer in a range from -4 to +3.
YR = idx + 4

DATA PAD BUS (DPBS)



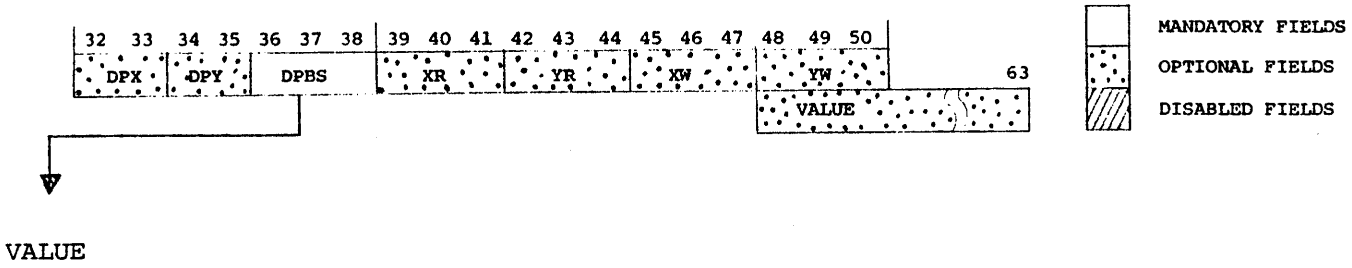
5 **DB = MD** ENABLE MAIN DATA MEMORY onto the DATA PAD BUS

Assembler
Format: DB=MD

Effect: (MDREG) → DPBS

Description: The contents of the MAIN DATA MEMORY (MD) location entered into the MEMORY OUTPUT REGISTER (MDREG) during this instruction cycle are enabled onto the DATA PAD BUS for the current instruction cycle. (See MEMORY GROUP SUMMARY for the set-up requirements for a MAIN DATA MEMORY (MD) READ operation.)

DATA PAD BUS (DPBS)



6 **DB = SPFN** ENABLE SPAD FUNCTION onto the DATA PAD BUS

Assembler Format: DB = SPFN

Effect:

(SPFN$_{\text{bits 7-15}}$) → DPBS$^{\text{EXPONENT}}$ bits Ø3-11

(SPFN$_{\text{bit 6}}$) INVERTED[†] → DPBS$^{\text{EXPONENT}}$ bit Ø2

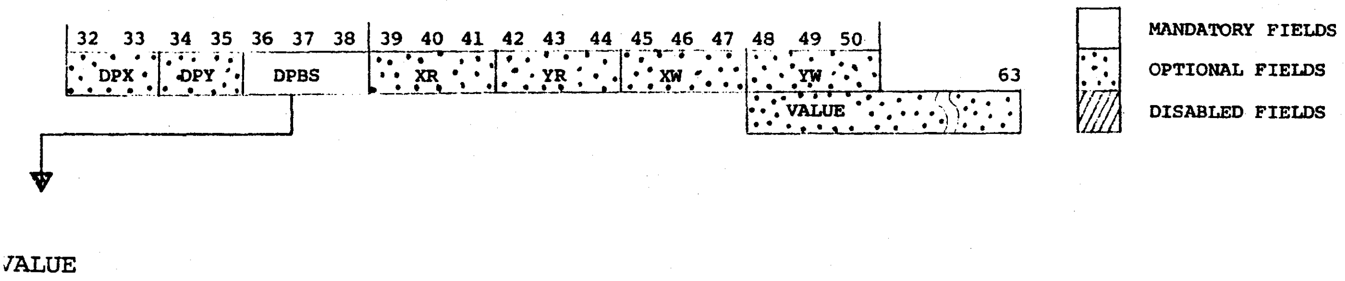
(SPFN$_{\text{bits Ø1-15}}$) → DPBS$^{\text{MANTISSA}}$$_{\text{bits 13-27}}$

(SPFN$_{\text{bit Ø (sign)}}$) → DPBS$^{\text{MANTISSA}}$$_{\text{bits 00-12}}$

Description: The 10 right-most bits of the S-PAD FUNCTION (SPFN bits [6-15]) plus BIAS , are enabled onto the DATA PAD BUS (EXPONENT) (DPBS [EXP] bits 02-11). The 15 right-most bits of SPFN (bits 01-15) are concurrently enabled onto the DPBS (MANTISSA bits) 13-27 and the left-most remaining bit (SPFN [bit 0]) is extended and enabled onto the left-most remaining bits of the DPBS (MANTISSA) (bits 00-12). SPFN is enabled onto DPBS during the current instruction only.

[†] BIAS: For this operation, a BIAS of 512(10) is added by inverting SPFN(bit)06.

DATA PAD BUS (DPBS)



7 DB = TM

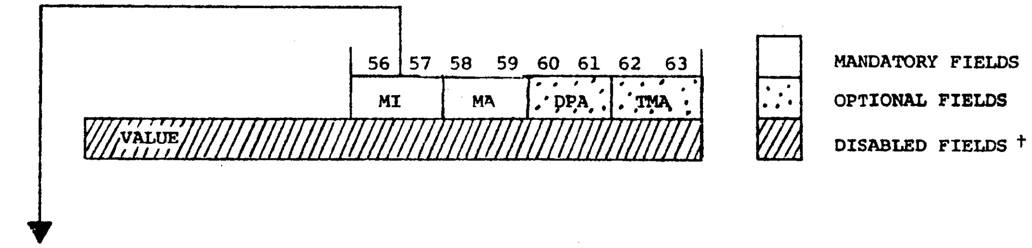
ENABLE TABLE MEMORY onto the DATA PAD BUS

Assembler
Format: DB=TM

Effect: (TMREG)→ DPBS

Description: The contents of the TABLE MEMORY (TM) location currently entered into the TABLE MEMORY REGISTER (TM REG) is enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle. (See MEMORY GROUP SUMMARY-TM for the set-up requirements for a TABLE MEMORY [TM] READ operation).

MEMORY INPUT GROUP (MI)



VALUE

Ø No Operation

1 MI<FA WRITE FLOATING ADDER RESULT to MAIN DATA MEMORY; VIA MEMORY INPUT REGISTER

Assembler Format: MI<FA

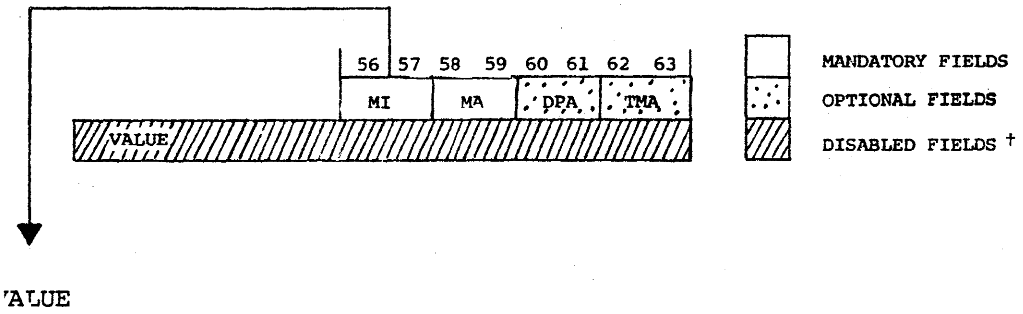
Example: MOV 5,5; SETMA; MI<FA

Effect: $(FA) \rightarrow MI \rightarrow MD_{(MA)}$

Description: The FLOATING ADDER RESULT (FA) currently available is loaded into the MEMORY INPUT REGISTER (MI) and written into the MAIN DATA MEMORY (MD) location specified by the current contents of the MEMORY ADDRESS REGISTER (MA). (See: MEMORY GROUP SUMMARY - MD for the necessary timing implications).

NOTE: The contents of the MEMORY ADDRESS REGISTER (MA), which specifies the MD location to be written, are determined by the mandatory and simultaneous SETMA, INCMA, DECMA or LDMA instruction being processed.

MEMORY INPUT GROUP (MI)



2 MI<FM

WRITE FLOATING MULTIPLY RESULT to MAIN DATA MEMORY; VIA MEMORY INPUT REGISTER

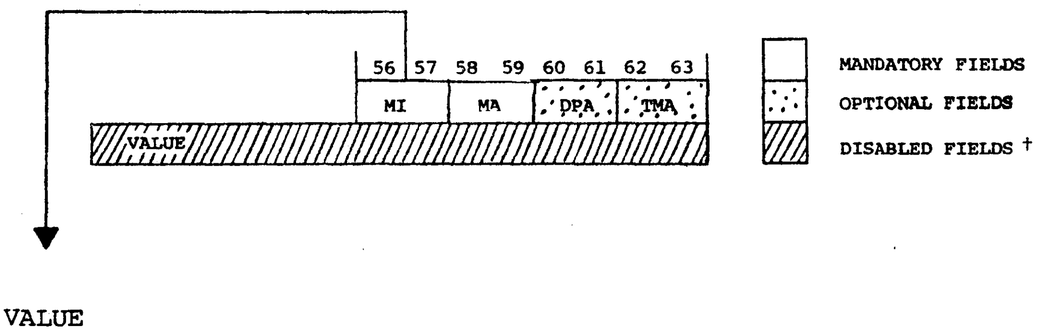
Assembler Format: MI<FM

Example: INCMA; MI<FM

Effect: $(FM) \rightarrow MI \rightarrow MD_{(MA)}$

Description: The FLOATING MULTIPLIER RESULT (FM) currently available is loaded into the MEMORY INPUT REGISTER (MI) and written into MAIN DATA MEMORY (MD) at the location specified by the current contents of the MEMORY ADDRESS REGISTER (MA).

MEMORY INPUT GROUP (MI)



3 MI< DB

WRITE DATA PAD BUS to MAIN DATA MEMORY; VIA MEMORY INPUT REGISTER

Assembler Format: MI< DB

Example: DECMA; MI< DB; DB=ZERO

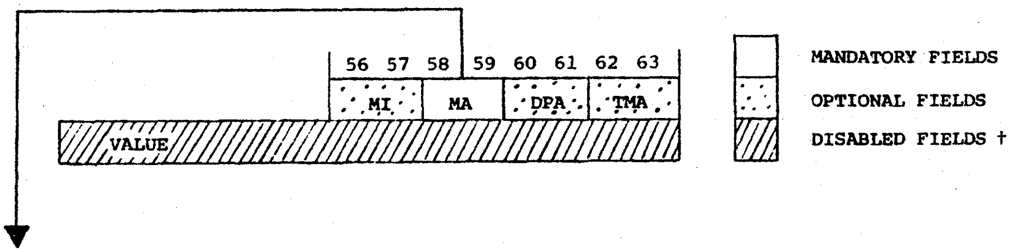
or: DECMA; MI< ZERO

Effect: $(DPBS) \rightarrow MI \rightarrow MD_{(MA)}$

Description: The current data enabled onto the DATA PAD BUS(DB) is loaded into the MEMORY INPUT REGISTER(MI) and written into MAIN DATA MEMORY (MD) at the location specified by the current contents of the MEMORY ADDRESS REGISTER (MA).

This instruction should be used in conjunction with a DPBS ENABLE instruction unless a zero is desired.

MEMORY ADDRESS GROUP (MA)



VALUE		
Ø		No Operation
1	INCMA	INCREMENT MEMORY ADDRESS REGISTER; INITIATE MAIN DATA MEMORY

Assembler
Format: INCMA

Example: INCMA; MI <FA

Effect: (MA) + 1→MA; initiate MD cycle

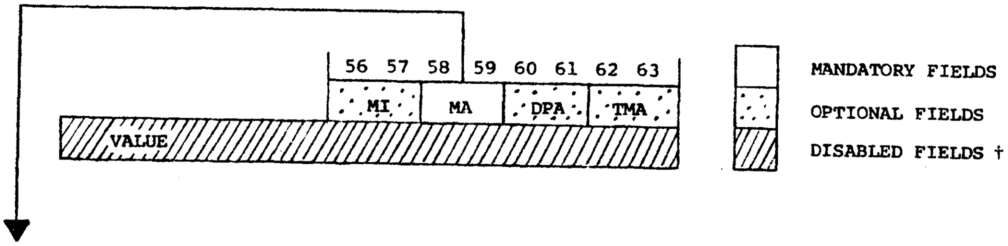
Description: The contents of the MEMORY ADDRESS REGISTER (MA) are incremented by "1" and a MAIN DATA MEMORY (MD) cycle is initiated.

If used concurrently with an MI group instruction, an MD WRITE is initiated using the incremented value of MA as the MD location specifier.

If used without a concurrent MI group instruction, then an MD READ is initiated using the incremented value of MA as the MD location specifier.

(See MEMORY GROUP SUMMARY - MD, for the necessary memory timing implications involved in the use of this field.)

MEMORY ADDRESS GROUP (MA)



VALUE

2 DECMA

DECREMENT MEMORY ADDRESS REGISTER; INITIATE MAIN DATA MEMORY

Assembler Format: DECMA

Example: DECMA;MI<FM

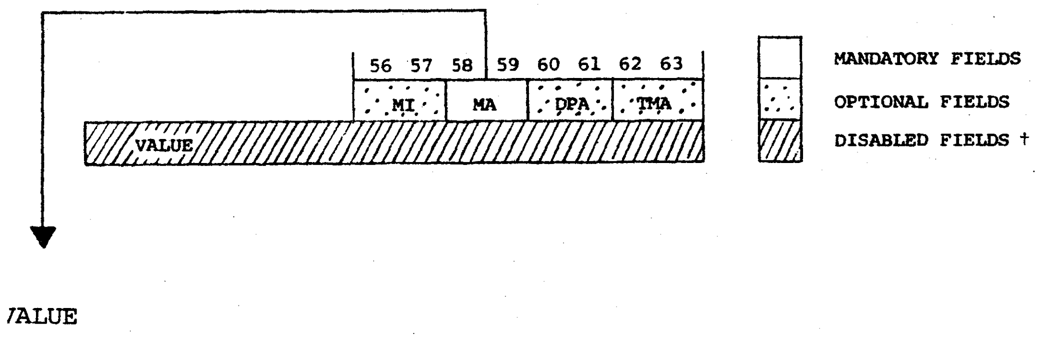
Effect: (MA)-1→MA; initiate MD cycle

Description: The contents of MEMORY ADDRESS REGISTER (MA) are decremented by "1" and a MAIN DATA MEMORY (MD) cycle is initiated.

If used with an MI group instruction, an MD WRITE is initiated using the decremented contents of MA as the MD location specifier.

If used without a concurrent MI group instruction, then an MD READ is initiated using the decremented contents of MA as the MD location specifier.

MEMORY ADDRESS GROUP (MA)



3 SETMA

SET MEMORY ADDRESS REGISTER FROM THE S-PAD
FUNCTION: INITIATE MAIN DATA MEMORY

Assembler Format: SETMA

Example: MOVE 0,0; SETMA; MI<TM

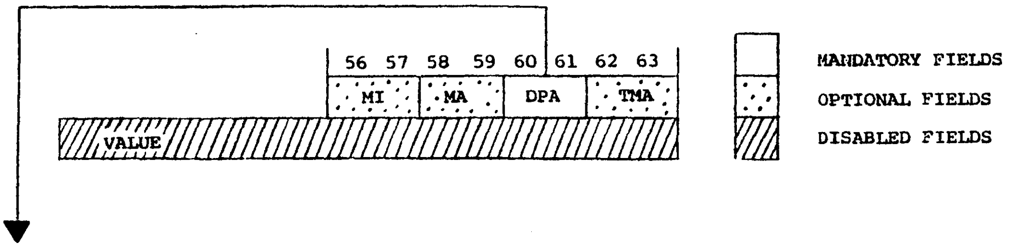
Effect: (SPFN)→MA; or, if LDREG field is being used, then (DPBS)→MA, instead.

Description: The contents of the MEMORY ADDRESS REGISTER (MA) are replaced by the S-PAD OUTPUT (SPFN) of the current S-PAD operation.

However, if an LDREG field instruction (see I/O) is used during the same instruction, then the contents currently enabled onto the DATA PAD BUS (DPBS), and not SPFN, are loaded into the MEMORY ADDRESS REGISTER (MA).

If used with an MI group instruction, an MD WRITE will be initiated. If used without an MI group instruction, then an MD READ will be initiated. In either case, the newly-formed value of MA will be used as the MD location specifier.

DATA PAD ADDRESS GROUP (DPA)



VALUE

Ø No Operation

1 INCDPA INCREMENT DATA PAD ADDRESS

Assembler:
Format: INCDPA

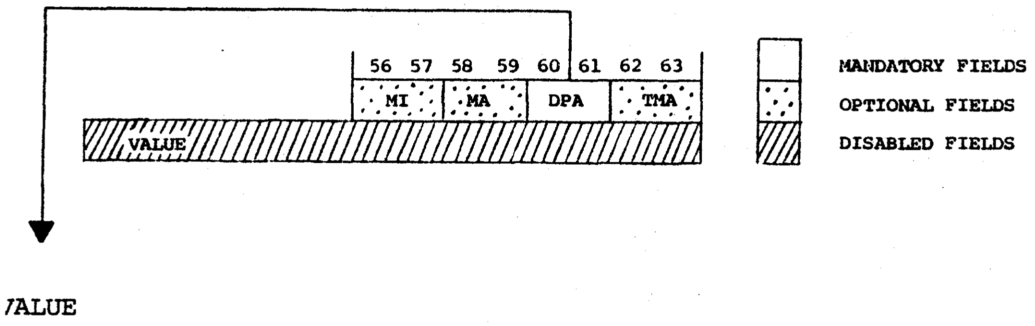
Effect: (DPA)+1→ DPA

Description: The contents of the DATA PAD ADDRESS REGISTER (DPA) are decremented by "1."

DPA will not be affected during the current instruction cycle but will be available as of the next instruction cycle.

(See DATA PAD GROUP SUMMARY - DPA.)

DATA PAD ADDRESS GROUP (DPA)



2 DECDPA DECREMENT DATA PAD ADDRESS REGISTER

Assembler
Format: DECDPA

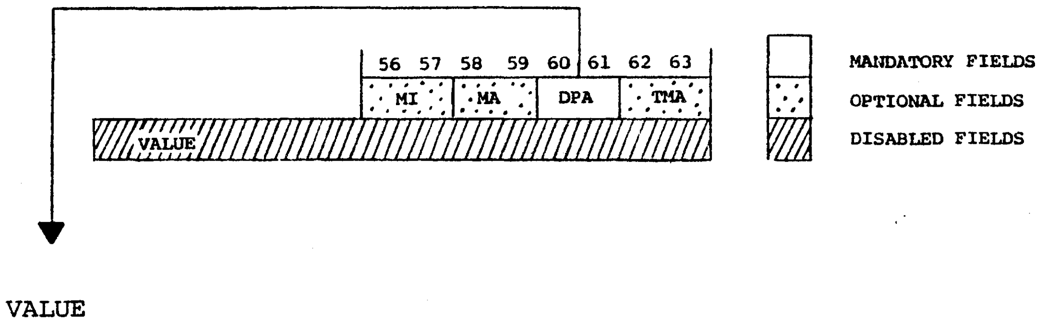
Effect: (DPA)-1→ DPA

Description: The contents of the DATA PAD ADDRESS REGISTER (DPA) are incremented by "1."

DPA will not be affected during the current instruction cycle but will be available as of the next instruction cycle.

(See DATA PAD GROUP SUMMARY - DPA.)

DATA PAD ADDRESS GROUP (DPA)



3 SETDPA — SET DATA PAD ADDRESS REGISTER from SPFN

Assembler Format: SETDPA

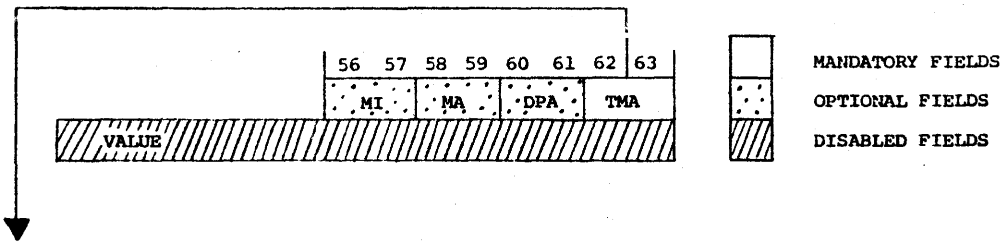
Effect: (SPFN) → DPA; or, if LDREG instruction is being used, then (DPBS) → DPA, instead.

Description: The current S-PAD (SPFN) is loaded into the DATA PAD ADDRESS REGISTER (DPA). However, if an LDREG field instruction (see I/O) is used during the same instruction cycle, the data enabled onto the DATA PAD BUS (DPBS) during this cycle, and not SPFN, will be loaded into the DPA.

DPA will not be affected during the current instruction cycle but will be availaɒle as of the next instruction cycle.

(See DATA PAD GROUP SUMMARY - DPA.)

TABLE MEMORY ADDRESS GROUP (TMA)



VALUE

Ø No Operation

1 INCTMA INCREMENT TABLE MEMORY ADDRESS REGISTER

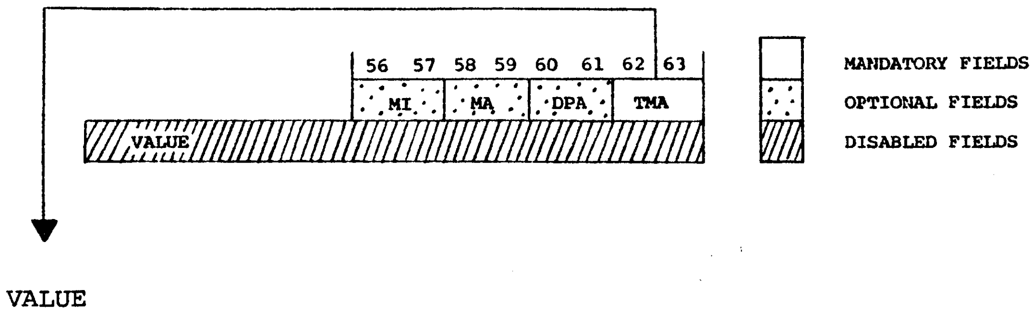
Assembler
Format: INCTMA

Effect: (TMA)+1→ TMA

Description: The contents of the TABLE MEMORY ADDRESS REGISTER (TMA) are incremented by one.

The contents of TM (TMA) are available in TMREG two cycles after execution of this instruction. The modified contents of TMA are available on the next instruction cycle.

TABLE MEMORY ADDRESS GROUP (TMA)



2 DECTMA DECREMENT TABLE MEMORY ADDRESS REGISTER

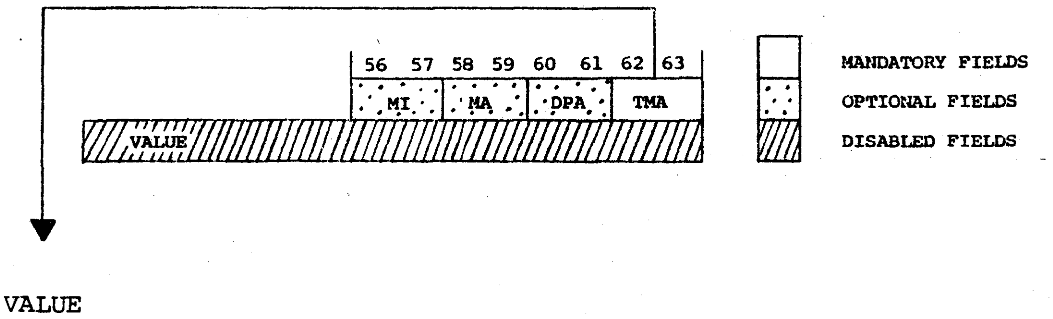
Assembler
Format: DECTMA (See, MEMORY GROUP SUMMARY - TMA)

Effect: (TMA)-1→ TMA

Description: The contents of the TABLE MEMORY ADDRESS REGISTER (TMA) are decremented by one.

The contents of TM (TMA) are available in TMREG two cycles after execution of this instruction.

TABLE MEMORY ADDRESS GROUP (TMA)



3 SETTMA

SET TABLE MEMORY ADDRESS REGISTER from S-PAD FUNCTION

Assembler Format: SETTMA

Effect: (SPFN)→TMA; or, if LDREG field is being used, then (DPBS)→TMA, instead.

Description: The contents of the result of the current S-PAD operation (SPFN) are loaded into the TABLE MEMORY ADDRESS REGISTER (TMA). However, if an LDREG field instruction is used concurrently (see I/O), then the data enabled onto the DATA PAD BUS (DPBS), and not SPFN, is loaded into TMA.

The contents of TMA are available as TM two cycles after execution of this instruction.

INDEX

Notice to the Reader

- Help us improve the quality and usefulness of this manual.

- Your comments and answers to the following READERS COMMENT form would be appreciated.

To mail: fold the form in three parts so
that Floating Point Systems'
mailing address is visible; seal.

Thank you

READERS COMMENT FORM

Document Title ______________________________

Your comments and answers will help us improve the quality and usefulness of our publications. If your answers require qualification or additional explanation, please comment in the space provided below.

How did you use this manual?

( ) AS AN INTRODUCTION TO THE SUBJECT
( ) AS AN AID FOR ADVANCED TRAINING
( ) TO LEARN OF OPERATING PROCEDURES
( ) TO INSTRUCT A CLASS
( ) AS A STUDENT IN A CLASS
( ) AS A REFERENCE MANUAL
( ) OTHER ______________________

Did you find this material . . .

	YES	NO
● USEFUL?	( )	( )
● COMPLETE?	( )	( )
● ACCURATE?	( )	( )
● WELL ORGANIZED?	( )	( )
● WELL ILLUSTRATED?	( )	( )
● WELL INDEXED?	( )	( )
● EASY TO READ?	( )	( )
● EASY TO UNDERSTAND?	( )	( )

Please indicate below whether your comment pertains to an addition, deletion, change or error; and, where applicable, please refer to specific page numbers.

Page	Description of error or deficiency

From:

Name ______________________
Firm ______________________
Address ______________________
Telephone ______________________

Title ______________________
Department ______________________
City, State ______________________
Date ______________________

First Class
Permit No. A-737
Portland,
Oregon

BUSINESS REPLY

No postage stamp necessary if mailed in the United States

Postage will be paid by:

FLOATING POINT SYSTEMS, INC.

P. O. Box 23489
Portland, Oregon 97223

Attention: Technical Publications

FLOATING POINT
SYSTEMS, INC.

CALL TOLL FREE 800-547-1445
P.O. Box 23489, Portland, OR 97223
(503) 641-3151, TLX: 360470 FLOATPOINT PTL