ARP

ARITHMETIC PROCESSOR

CHAPTER 5

5.0 INTRODUCTION

The Arithmetic Processor (ARP) is a programmable unit capable of performing very high speed fixed point arithmetic operations. The speed of the ARP results from the use of pipelining techniques, overlapped move and arithmetic operations, and the inclusion of a very fast 128 word Temporary Register File. All arithmetic operations may involve integers, scaled fractions, or (to a limited extend) a combination of integer and scaled fraction operands.

5.1 ORGANIZATION OF THE ARP.

Figure 5.1 provides a general block diagram of the ARP. Those elements of the ARP which are shown in black in Figure 5.1 are common to all processors. These elements are discussed in Section 2.6. The elements which are unique to the ARP are shown in red in Figure 5.1. These elements are the Arithmetic Unit and the Temporary Register File.

The heart of the ARP is the Arithmetic Unit which is supported by a Temporary Register File that provides 128 words of high-speed local storage (25 nanosecond cycle time). The Arithmetic Unit communicates with the DATA MULTIBUS via a Bus Store Register S and a Bus Load Register L.

A block diagram of the Arithmetic Unit and Temporary Register File is presented in Figure 5.2.

Two distinct types of operations take place in the Arithmetic Unit. These are arithmetic operations and move operations. The Arithmetic Unit is designed to execute an arithmetic instruction of the general form:

$$R = \pm(A \pm B) * C \pm E$$

in 175 nanoseconds.

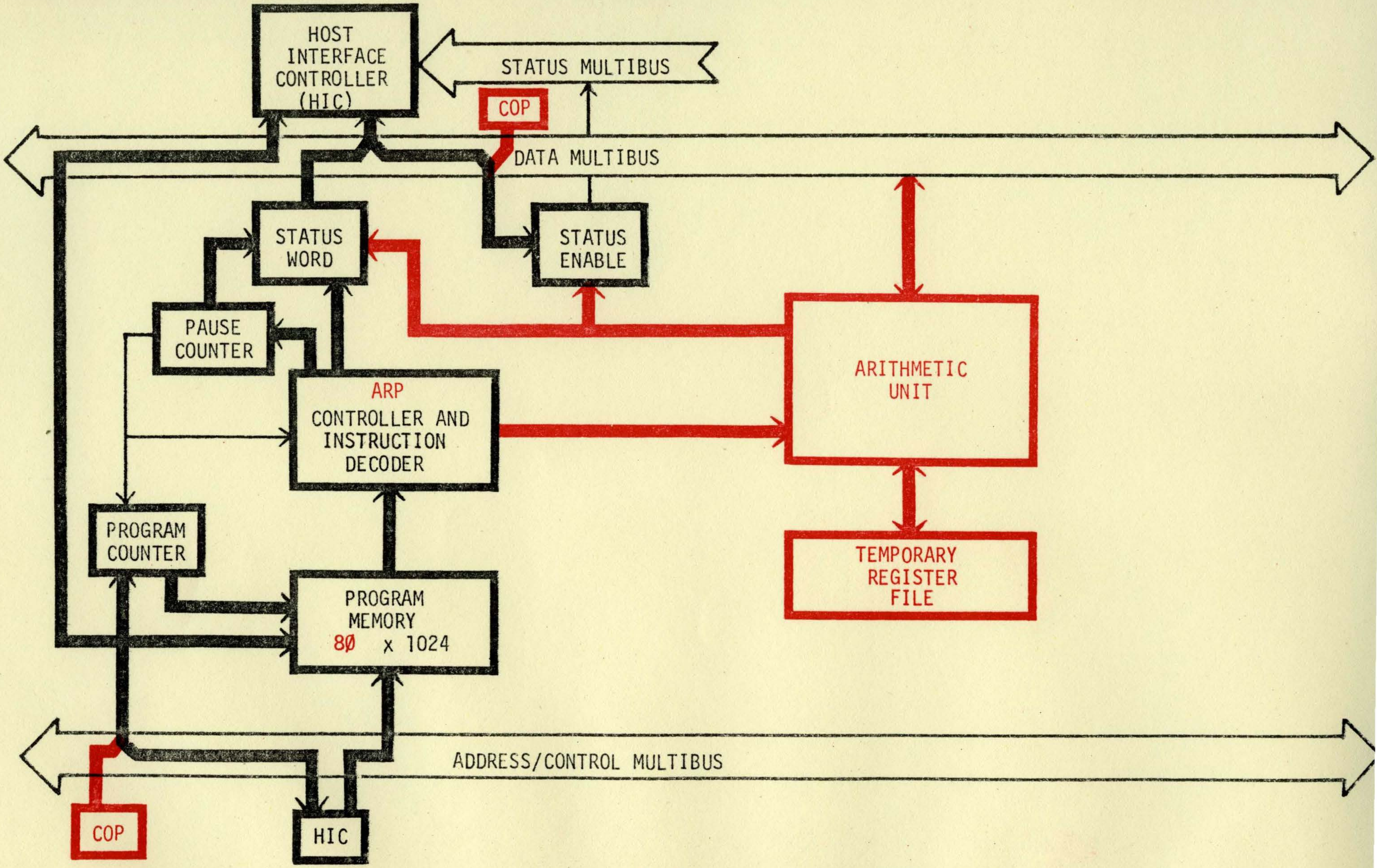


FIGURE 5.1 ARP BLOCK DIAGRAM

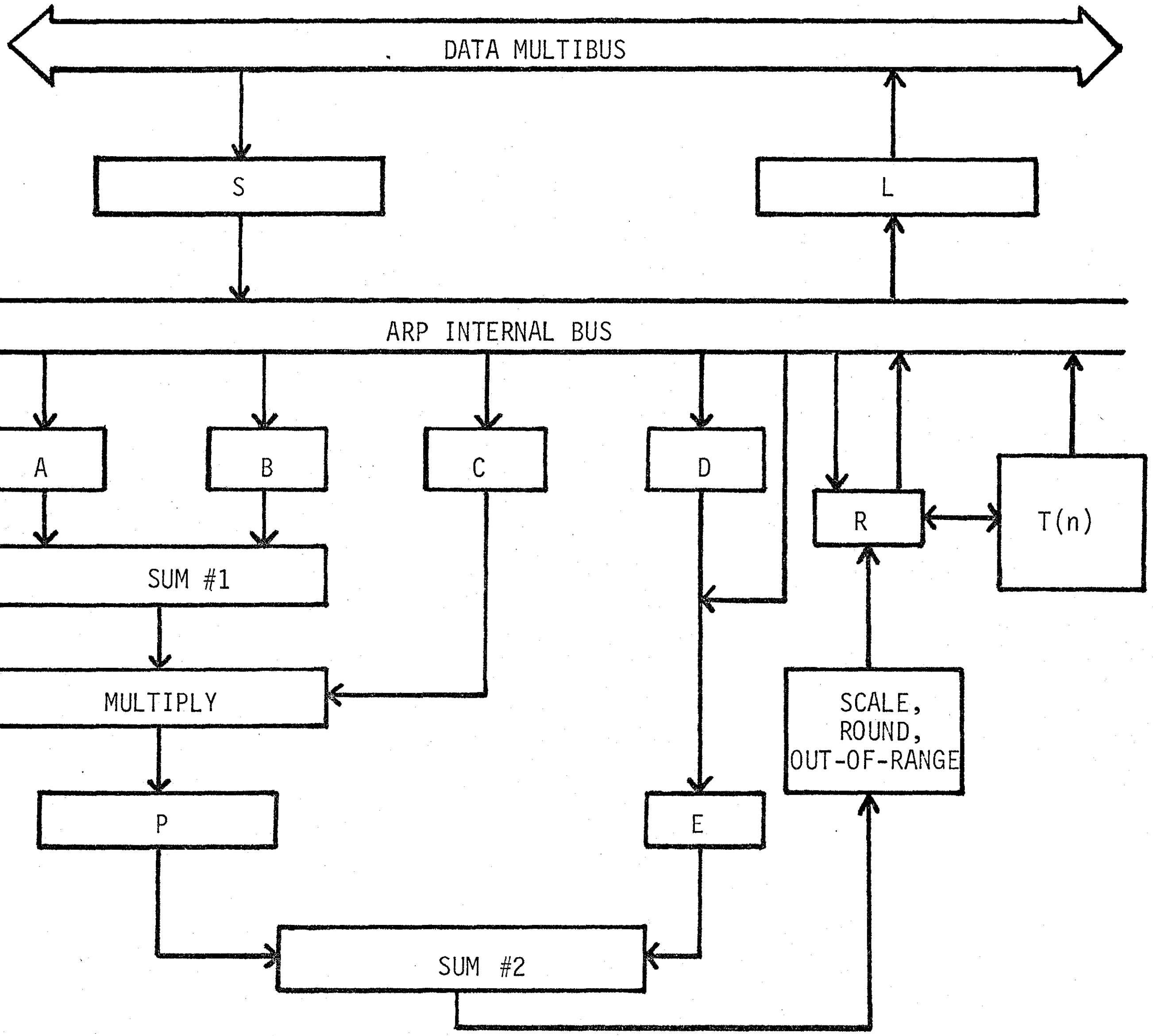


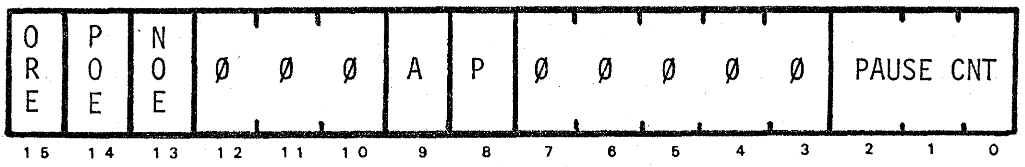
FIGURE 5.2 BLOCK DIAGRAM OF THE ARITHMETIC UNIT AND TEMPORARY REGISTER FILE

Data movement between the Arithmetic Unit and the MULTIBUS, between the Arithmetic Unit and the Temporary Register File, and between registers within the Arithmetic Unit itself is handled by data move instructions. Data move operations and arithmetic operations are overlapped (i.e., performed concurrently) within the Arithmetic Unit. These operations will be discussed in detail in subsequent sections.

5.1.1 PROCESSOR ADDRESS

The processor address for the ARP is Ø3.

5.1.2 PROCESSOR STATUS WORD



BIT(S)	DESCRIPTION
Ø-2	Remaining PAUSE count
3-7	UNASSIGNED
8	ARP is present when set
9	ARP is active when set
1Ø-12	UNASSIGNED
13*	Negative out-of-range error on some instruction
14*	Positive out-of-range error on some instruction
15*	Out-of-range error (one of the above two conditions)

*Once set these bits remain set until cleared by a READ operation.

The ARP Processor Status Word (PSW) contains information on the current status of the ARP. The PSW is a read-only register.

5.1.2.1 OUT-OF-RANGE ERRORS

Three out-of-range error signals are generated in the Arithmetic Unit of the ARP. These are:

a.) The Negative Out-of-Range Error (NOE)
b.) The Positive Out-of-Range Error (POE),
and c.) The Out-of-Range Error (ORE).

The NOE signal is generated if the result of an arithmetic instruction is a negative number which requires more than 16 bits for its two's-complement, binary representation. Similarly, the POE signal is generated if the result of an arithmetic instruction is a positive number which requires more than 16 bits for its two's-complement, binary representation. The ORE signal is generated if either an NOE or a POE condition occurs.

5.1.3 PROCESSOR STATUS ENABLE

EOR	EPO	ENO	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BIT	DESCRIPTION
Ø-12	UNASSIGNED
13	Enable the NOE signal from the Arithmetic Unit to the AER line in the STATUS MULTIBUS.
14	Enable the POE signal from the Arithmetic Unit to the AER line in the STATUS MULTIBUS.
15	Enable the ORE signal from the Arithmetic Unit to the AER line in the STATUS MULTIBUS.

The Processor Status Enable Register (PSE) is a write-only register. This register has the same address as the ARP Processor Status Word.

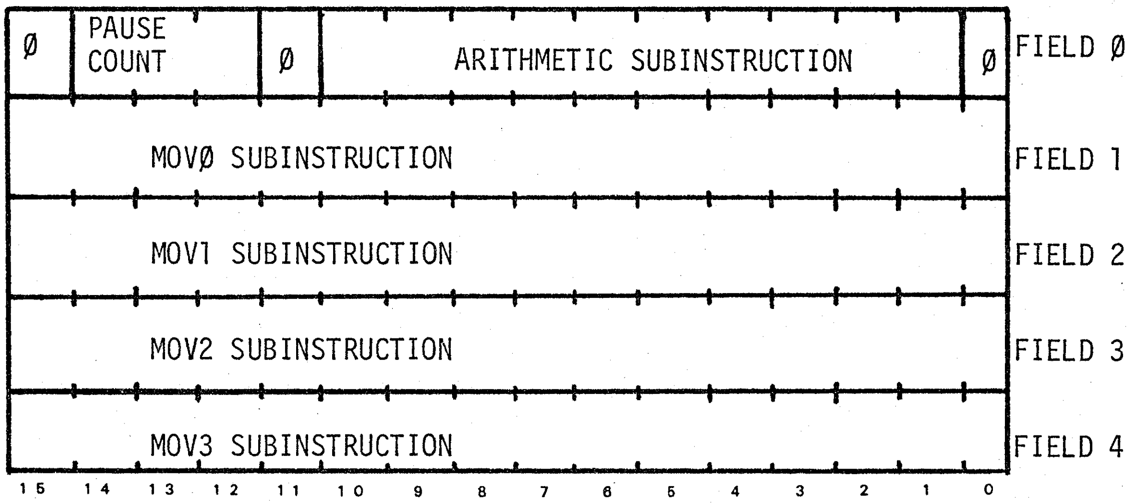
5.1.4 PROGRAM MEMORY

The ARP program memory contains 1,024 words. Each 80-bit instruction word is divided into five 16-bit fields. An ARP instruction is normally loaded as a sequence of five 16-bit words (one per field) from the host processor. However, each field is individually addressable and can be accessed from the host processor for a READ or WRITE operation.

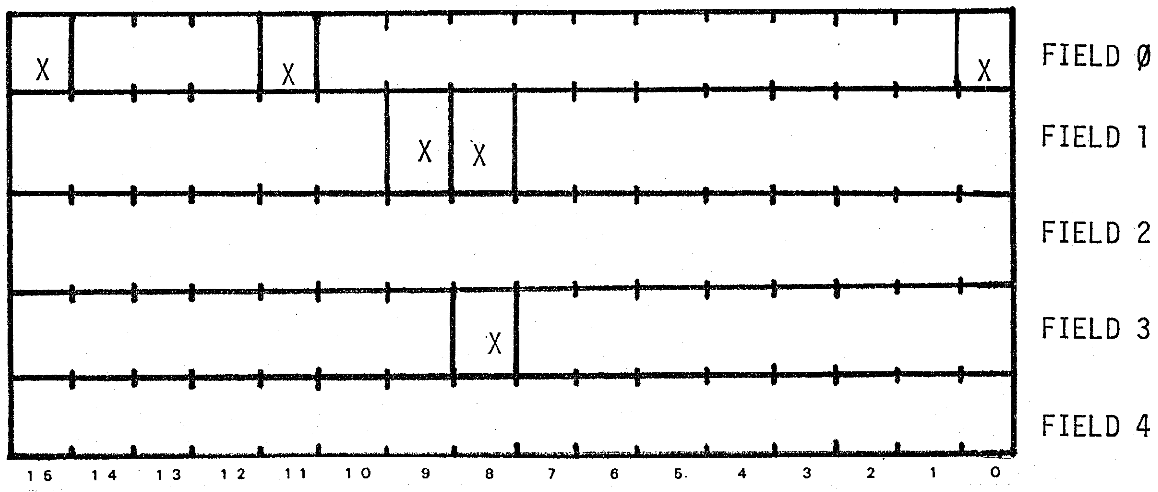
5.2 THE ARP INSTRUCTION SET

5.2.1 ARP INSTRUCTION WORD FORMAT

The ARP instruction word format is:



Within this format, certain bits are unused. Each of these unused bits is denoted by an "X" in the following diagram.



The unused bits cannot be set and are always read back as a Ø.

5.2.2 MICROPROGRAMMING ARP INSTRUCTIONS

The ARP can be micro-programmed; that is, the ARP instruction can contain one or more of the following:

< ARITHMETIC > subinstruction
< MOVØ > subinstruction
< MOV1 > subinstruction
< MOV2 > subinstruction
< MOV3 > subinstruction
< PAUSE > subinstruction

The general form of an ARP instruction is:

```
<ARITHMETIC> <MOVØ> <MOV1> <MOV2> <MOV3> <PAUSE>
```

Examples of ARP instructions:

```
IA (A+B)*C; MOVØ TØ,A; MOV1 S,R; MOV2 R,T6,D
FA (A)*C+E; MOVØ T1,A; MOV2 T2,B,D; MOV3 R,E; PAUSE 2
```

5.2.3 ARP ARITHMETIC SUBINSTRUCTION

A block diagram of the Arithmetic Unit in the ARP is shown in Figure 5.2. This Arithmetic Unit performs operations of the quasi-general form:

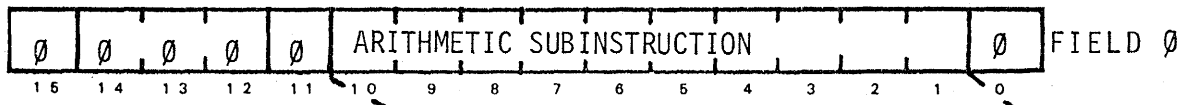
$$R = \pm (A \pm B) * C \pm E$$

For the sake of convenience, this quasi-general form of the arithmetic subinstruction is used in various places in this manual. The precise description of the allowed arithmetic expressions which can be evaluated in the Arithmetic Unit of the ARP is contained in the Backus-Naur Form (BNF) definition of the arithmetic subinstruction presented in Figure 5.3.

< ARITHMETIC INSTRUCTION >

ARITHMETIC SUBINSTRUCTION BNF AND BIT PATTERNS

$$< \text{OP CODE} > \underbrace{\pm (A\pm B)*C\pm E}_{\text{Quasi-General Form}}$$



BACKUS-NAUR FORM (BNF)

	10	9	8	7	6	5	4	3	2	1
< ARITHMETIC SUBINSTRUCTION > :: = FA < EXPRESSION >	Ø	Ø								
| FASL < EXPRESSION >	Ø	1								
| FASR < EXPRESSION >	1	Ø								
| IA < EXPRESSION >	1	1								
| no op† (X=Ø,1)	X	X	X	Ø	Ø	X	Ø	X	X	X
< EXPRESSION > :: = |< PRODUCT > + E††			Ø	1						
|< PRODUCT > - E††			1	1						
|< PRODUCT > + D			Ø	1						
|< PRODUCT > - D			1	1						
|< PRODUCT >			Ø	Ø						
< PRODUCT > :: = | + <SUM> *C | < SUM > *C					Ø	Ø	1			
| - <SUM> *C					Ø	1	1			
| + <SUM> *MC††† | < SUM > *MC					1	Ø	1			
| - <SUM> *MC					1	1	1			
| + <SUM> | < SUM >					1	Ø	Ø			
| - <SUM>					1	1	Ø			
| Ø					Ø	1	Ø			
					Ø	Ø	Ø			
< SUM > :: = | (A+B)								Ø	1	1
| (A-B)								1	1	1
| (A)								1	Ø	1
| (A+1)								Ø	Ø	1
| (B)								Ø	1	Ø
| (-B)								1	1	Ø
| 1								Ø	Ø	Ø
| Ø								1	Ø	Ø

(See also next page)

FIGURE 5.3 DEFINITION OF THE ARITHMETIC SUBINSTRUCTION

† On a no op or on an operation which yields Ø, $R_{1.75} \leftarrow Ø$.

†† Since D is automatically moved to E during the MØ subinterval of each instruction cycle, a move of a data word to E via a MOV1, MOV2, or MOV3 subinstruction must be microprogrammed with any arithmetic subinstruction which specifies E as an operand.

††† MC = |C|

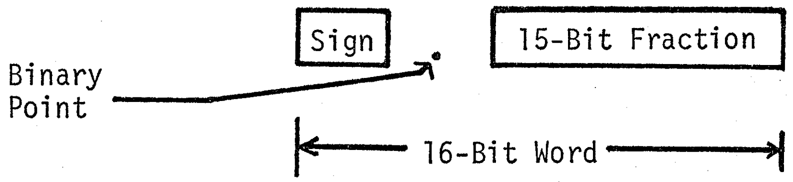
EXAMPLES WITH BIT PATTERNS

< ARITHMETIC INSTRUCTION >	BIT PATTERN (BITS 1Ø-1)
FA -(A+B)*C-D	ØØ 11 Ø11 Ø11
IA (A)+E	11 Ø1 1ØØ 1Ø1
FASR (A)*C+D	1Ø Ø1 ØØ1 1Ø1

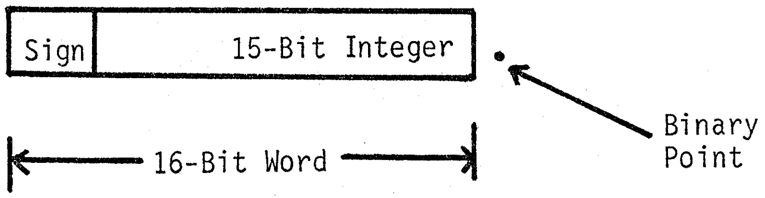
FIGURE 5.3 (CONTINUED) DEFINITION OF THE ARITHMETIC SUBINSTRUCTION

All arithmetic operations are performed in two's-complement, fixed point format. The Arithmetic Unit has two computational modes. These are the scaled fraction mode and the integer mode.

In the scaled fraction mode, each operand must lie in range -1.0 to + $(1.0-2^{-15})$ and the result, R, of the computation is a 16-bit number which also lies in this range. The format for a scaled fraction word is:



In the integer mode at least one of the operands must be an integer in the range -2^{15} to $+(2^{15} - 1)$. The format for an integer word is:



The integer mode allows all the operands to be integers and also allows certain mixed operations involving both integer and scaled-fraction operands. If all the operands are integers, the result, R, of the computation is an integer in the range -2^{15} to $+(2^{15}-1)$. For the allowable computations involving a mixture of integers and scaled-fraction operands, the result, R, is a scaled fraction in the range -1.0 + $(1.0 -2^{-15})$.

There are three arithmetic sub-instructions which pertain to the scaled-fraction mode. These are designated by the symbols FA, FASL, and FASR in the BNF description of the arithmetic subinstruction (Figure 5.3). There is one arithmetic subinstruction, designated by the symbol IA, which pertains to the integer mode.

As part of the precise definition of each of these four arithmetic subinstructions, a diagram is provided which specifies the word length and word format at each step through the flow of operations in the Arithmetic Unit. The basic form of these diagrams is shown in Figure 5.4, and thus to the diagram included as part of each arithmetic subinstruction definition:

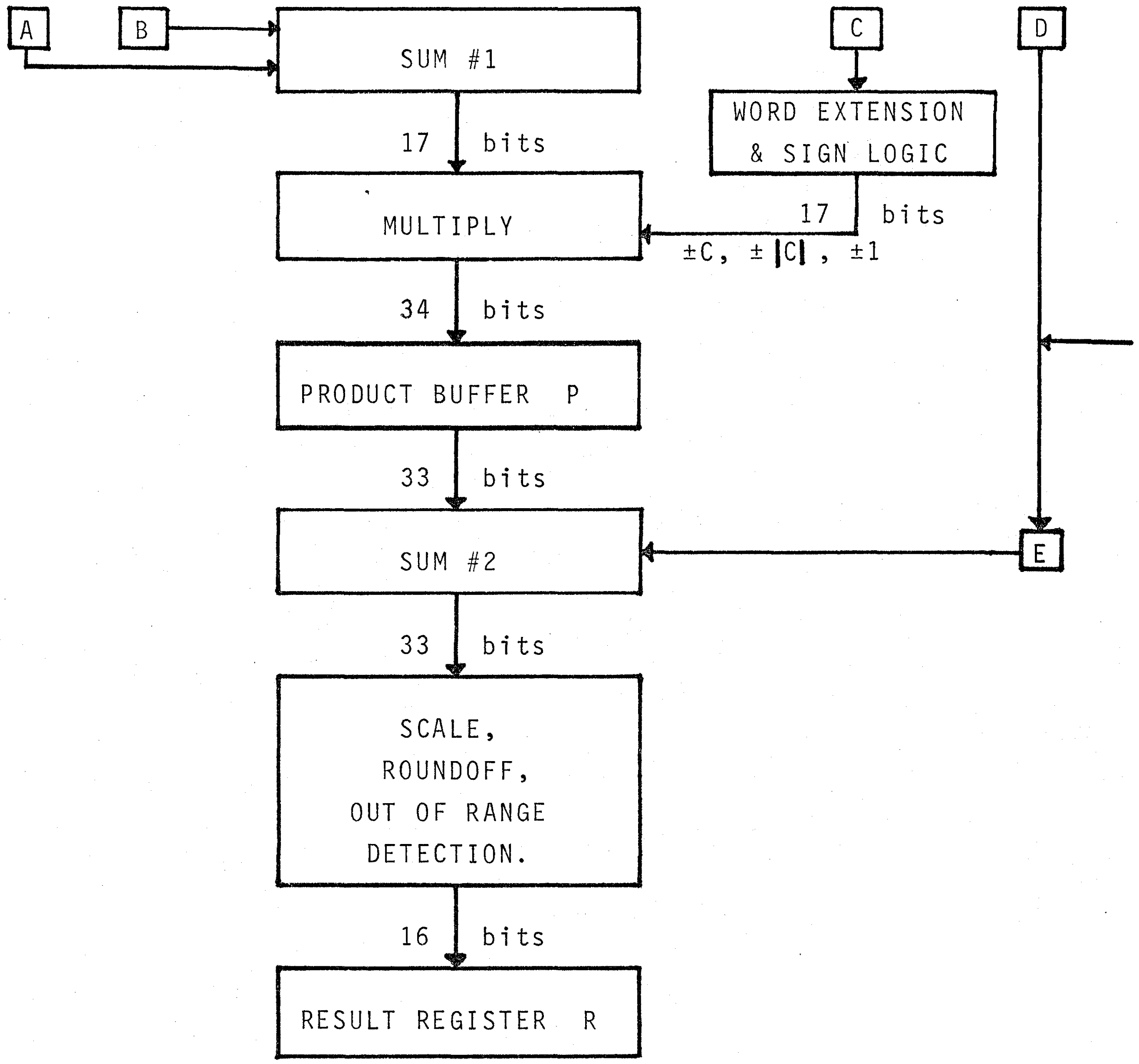


FIGURE 5.4 SPECIFICATION OF WORD LENGTHS IN THE ARITHMETIC UNIT

1. The Term < PRODUCT > in Figure 5.3 defines the possible outputs of the MULTIPLY operation in the Arithmetic Unit. < PRODUCT > may be viewed as specifying the multiplication of < SUM > by one of the following: $+C$, $-C$, $+|C|$, $-|C|$, $+1.\emptyset$, $-1.\emptyset$, or $\emptyset$. The selection of the appropriate input to MULTIPLY is one of the functions performed by the block labelled "Word Extension and Sign Logic".

2. In two's-complement format, the most positive number which can be represented with a given number of bits is one LSB (least significant bit) smaller than the magnitude of the most negative number which can be represented with that number of bits. This poses a problem if one wishes to compute $-C$ or $|C|$ when C takes on its most negative value. This problem is solved by converting operand C from a 16-bit number to a 17-bit number by extending the sign bit of C. This extension makes it possible to represent any value in the range -2.0 to $(2.0 - 2^{-15})$ in scaled fraction format or any value in the range -2^{16} to $+(2^{16}-1)$ in integer format. Clearly, this is adequate to handle $-1.\emptyset \leq C \leq +1.\emptyset$ or $-2^{15} \leq C \leq +2^{15}$. This word extension of operand C is performed by the block labelled "Word Extension and Sign Logic".

3. The MULTIPLY unit is a true 17-bit by 17-bit multiplier which produces a 34-bit product. However, consideration of the worst-case situation (for the possible range of each of the input operands) shows that all possible MULTIPLY outputs can be expressed with 33-bits (i.e., the two MSB's will always be the same). Therefore, in transferring the output of MULTIPLY to the PRODUCT BUFFER, the MSB is dropped and the PRODUCT BUFFER holds a 33-bit word.

4. Since D is automatically moved to E during the M$\emptyset$ subinterval of each instruction cycle, a MOV1, MOV2, or MOV3 subinstruction which moves a data word into E must be microprogrammed with any arithmetic subinstruction which explicitly specifies E as an operand.

5.2.3.1 FRACTION ARITHMETIC SUBINSTRUCTIONS

The three arithmetic subinstructions which have scaled-fraction operands and produce a scaled-fraction result are the following:

FA	Fraction Arithmetic
FASL	Fraction Arithmetic Shifted Left
FASR	Fraction Arithmetic Shifted Right

FA < EXPRESSION >

FRACTIONAL ARITHMETIC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	Ø	Ø	Ø	Ø	Ø	Ø	EXPRESSION								Ø	FIELD Ø

OPERATION:

```
$P_1 \leftarrow \pm (A_\emptyset \pm B_\emptyset)*C_\emptyset$
IF ROUND $(P_1 \pm E_1) \leq +1.\emptyset - 2^{-15}$
    THEN IF ROUND $(P_1 \pm E_1) \geq -1.\emptyset$
        THEN $R_{1.75} \leftarrow$ [ROUND $(P_1 \pm E_1)$]
        ELSE $R_{1.75} \leftarrow -1.\emptyset$
    ELSE $R_{1.75} \leftarrow +1.\emptyset - 2^{-15}$
    WHERE [ROUND $(P_1 \pm E_1)$] DENOTES THE LOW ORDER 16-BITS OF ROUND $(P_1 \pm E_1)$.
```

ERROR CONDITIONS:

Positive Out-of-Range Error (POE), if ROUND $(P_1 \pm E_1) > +1.\emptyset - 2^{-15}$

Negative Out-of-Range Error (NOE), if ROUND $(P_1 \pm E_1) < -1.\emptyset$

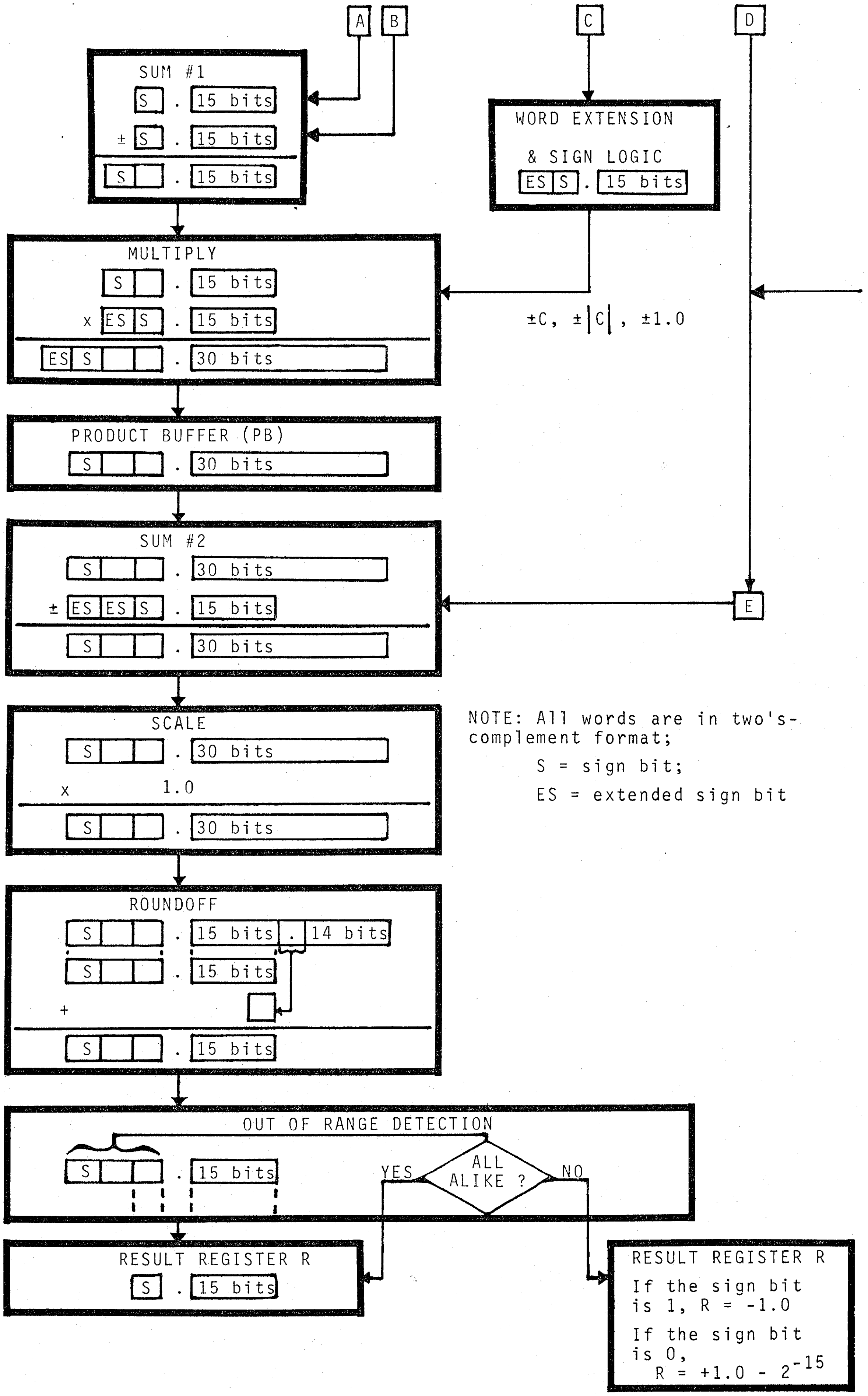
Out-of-Range Error (ORE), if either POE or NOE occurs.

DESCRIPTION:

The arithmetic operations defined by < EXPRESSION > are performed. Figure 5.5 shows the word length and word format at each step through the flow of operations in the Arithmetic Unit.

EXAMPLE:

FA(A+B)*MC



NOTE: All words are in two's-complement format;

S = sign bit;

ES = extended sign bit

FIGURE 5.5 FRACTIONAL ARITHMETIC (FA)

FASR < EXPRESSION >

FRACTIONAL ARITHMETIC SHIFTED RIGHT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	Ø	Ø	Ø	Ø	Ø	1	EXPRESSION								Ø	FIELD Ø

OPERATION:

```
$P_1 \leftarrow \pm (A_\emptyset \pm B_\emptyset)*C_\emptyset$
IF ROUND $((P_1 \pm E_1)/2) \leq +1.\emptyset - 2^{-15}$
    THEN IF ROUND $((P_1 \pm E_1)/2) \geq -1.\emptyset$
        THEN $R_{1.75} \leftarrow$ [ROUND $((P_1 \pm E_1)/2)$]
        ELSE $R_{1.75} \leftarrow -1.\emptyset$
    ELSE $R_{1.75} \leftarrow +1.\emptyset - 2^{-15}$
    WHERE [ROUND$((P_1 \pm E_1)/2)$] DENOTES THE LOW ORDER 16-BITS OF ROUND $((P_1 \pm E_1)/2)$.
```

ERROR CONDITIONS:

Positive Out-of-Range Error (POE), if ROUND $((P_1 \pm E_1)/2) > +1.0 - 2^{-15}$

Negative Out-of-Range Error (NOE), if ROUND $((P_1 \pm E_1)/2) < -1.\emptyset$

Out-of-Range Error (ORE), if either POE or NOE occurs.

DESCRIPTION:

The arithmetic operations defined by < EXPRESSION > are performed. Figure 5.6 shows the word length and word format at each step through the flow of operations in the Arithmetic Unit.

EXAMPLE:

FASR (A-B)+E

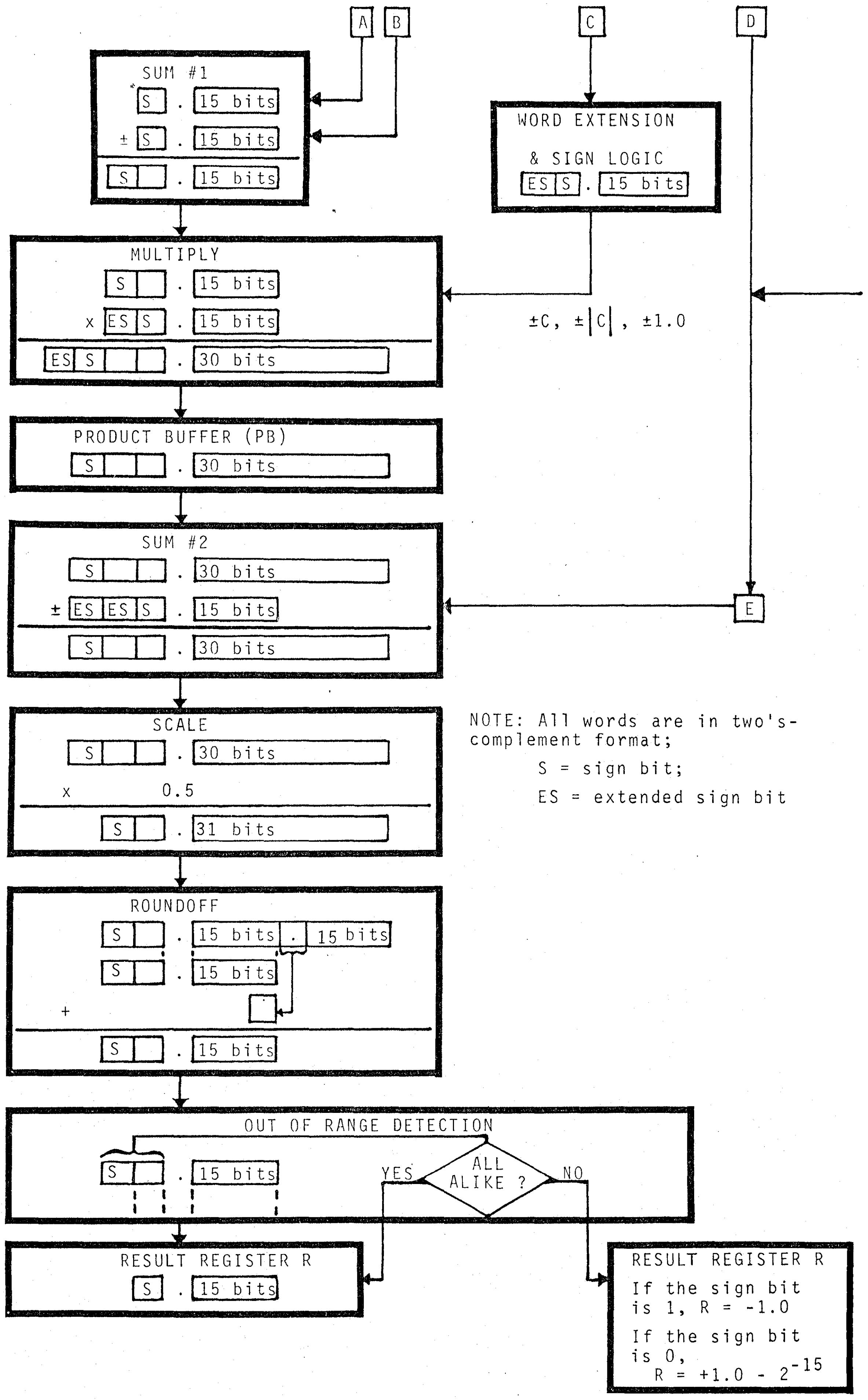


FIGURE 5.6 FRACTIONAL ARITHMETIC SHIFTED RIGHT (FASR)

FASL < EXPRESSION >

FRACTIONAL ARITHMETIC SHIFTED LEFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	Ø	Ø	Ø	Ø	1	Ø	EXPRESSION								Ø	FIELD Ø

OPERATION:

```
$P_1 \leftarrow \pm (A_Ø \pm B_Ø) * C_Ø$
IF ROUND $((P_1 \pm E) * 2) \leq +1.Ø - 2^{-15}$
    THEN IF ROUND $((P_1 \pm E_1) * 2) \geq -1.Ø$
        THEN $R_{1.75} \leftarrow$ [ROUND $((P_1 \pm E_1) * 2)$]
        ELSE $R_{1.75} \leftarrow -1.Ø$
    ELSE $R_{1.75} \leftarrow +1.Ø - 2^{-15}$
    WHERE [ROUND$((P_1 \pm E_1) * 2)$] DENOTES THE LOW ORDER 16-BITS OF ROUND $((P_1 \pm E_1) * 2)$
```

ERROR CONDITIONS:

Positive Out-of-Range Error (POE), if ROUND $((P_1 \pm E_1) * 2) > +(1.Ø - 2^{-15})$

Negative Out-of-Range Error (NOE), if ROUND $((P_1 \pm E_1) * 2) < -1.Ø$

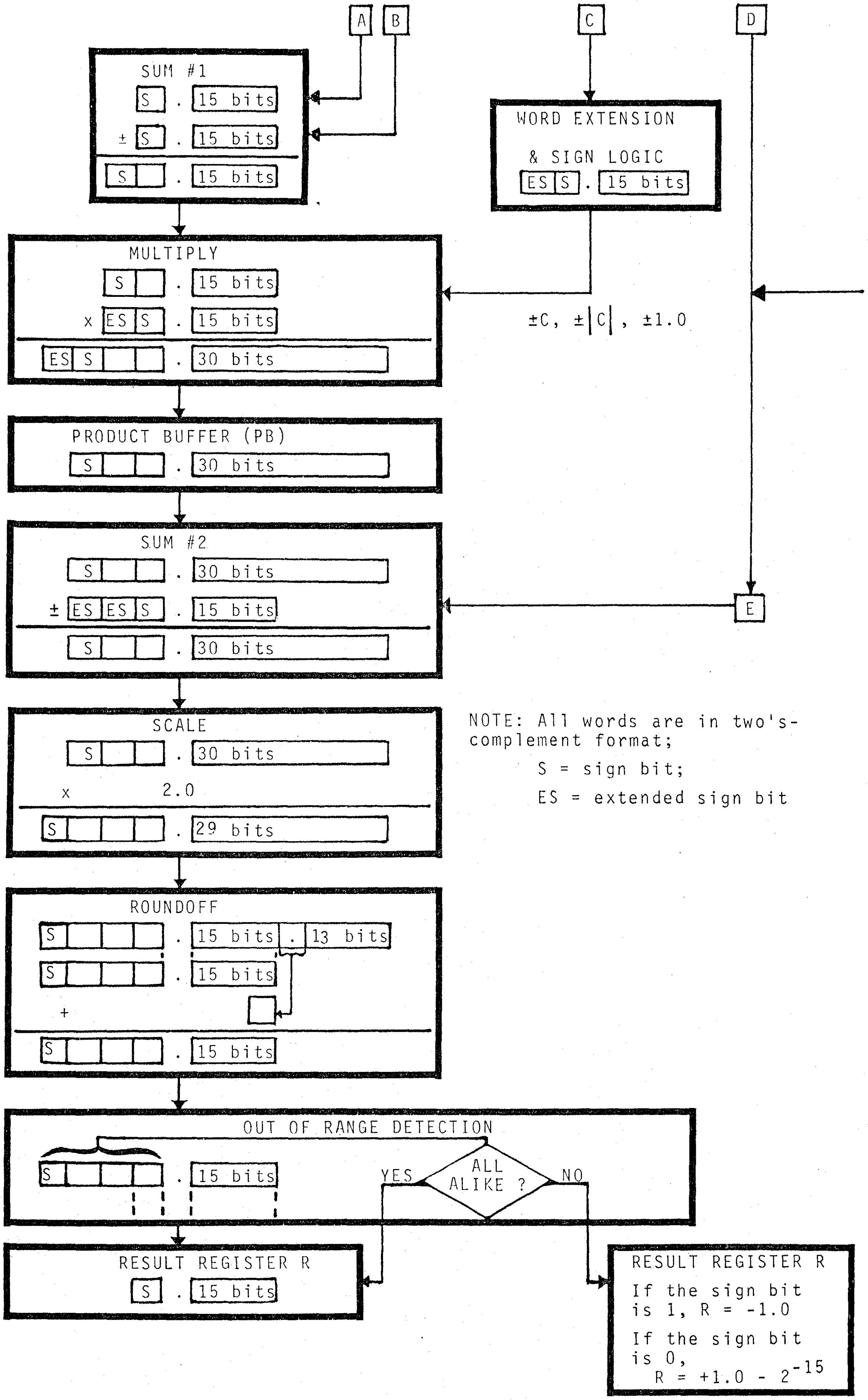
Out-of-Range Error (ORE), if either POE or NOE occurs.

DESCRIPTION:

The arithmetic operations defined by < EXPRESSION > are performed. Figure 5.7 shows the word length and word format at each step through the flow of operations in the Arithmetic Unit.

EXAMPLE:

FASL -(B)*MC+D



NOTE: All words are in two's-complement format;
S = sign bit;
ES = extended sign bit

FIGURE 5.7 FRACTIONAL ARITHMETIC SHIFTED LEFT (FASL)

5.2.3.2 INTEGER ARITHMETIC SUBINSTRUCTION

The arithmetic subinstruction which deals with integer operands and produces an integer result or which deals with mixed (i.e., scaled fraction/integer) operands and produces a scaled fraction result is the following:

IA	Integer Arithmetic

IA < EXPRESSION >

INTEGER ARITHMETIC (INTEGER OPERANDS ONLY)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
∅	∅	∅	∅	∅	1	1	EXPRESSION								∅	FIELD ∅

OPERATION:

```
$P_1 \leftarrow \pm(A_\emptyset \pm B_\emptyset)*C_\emptyset$
IF $(P_1 \pm E_1) \leq +2^{15} - 1$
    THEN IF $(P_1 \pm E_1) \geq -2^{15}$
        THEN $R_{1.75} \leftarrow [(P_1 \pm E_1)]$
        ELSE $R_{1.75} \leftarrow -2^{15}$
    ELSE $R_{1.75} \leftarrow +2^{15} - 1$
    WHERE $[(P_1 \pm E_1)]$ DENOTES THE LOW-ORDER 16-BITS OF $(P_1 \pm E_1)$.
```

ERROR CONDITIONS:

Positive Out-of-Range Error (POE), if $(P_1 \pm E_1) > +2^{15} - 1$

Negative Out-of-Range Error (NOE), if $(P_1 \pm E_1) < -2^{15}$

Out-of-Range Error (ORE), if either POE or NOE occurs.

DESCRIPTION:

The arithmetic operations defined by < EXPRESSION > are performed. Figure 5.8 shows the word length and word format at each step through the flow of operations in the Arithmetic Unit.

EXAMPLE:

IA (A)*C+D

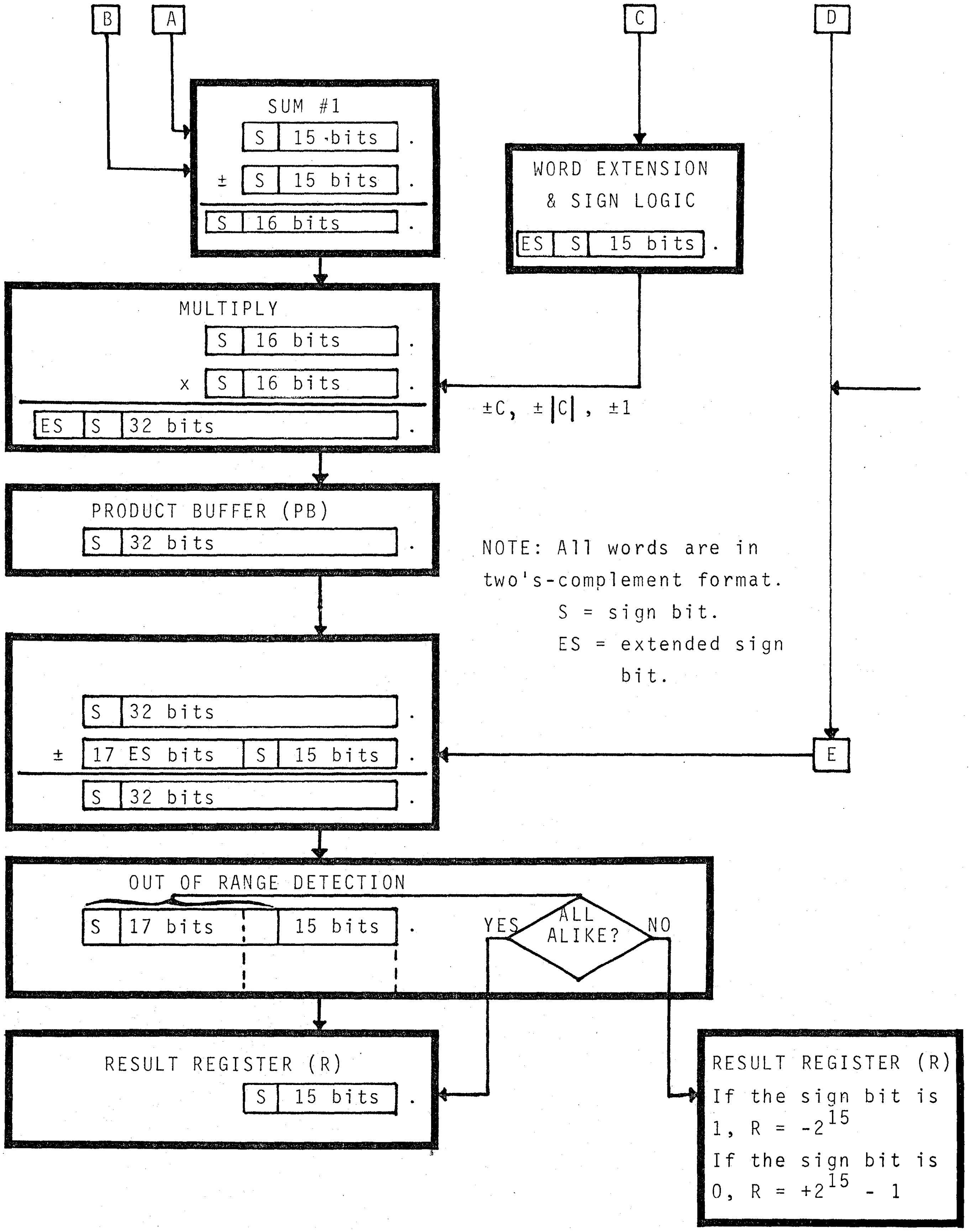


FIGURE 5.8 INTEGER ARITHMETIC (IA)

IA < EXPRESSION >

INTEGER ARITHMETIC (MIXED INTEGER/SCALED FRACTION OPERANDS)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	Ø	Ø	Ø	Ø	1	1	EXPRESSION								Ø	FIELD Ø

OPERATION:

```
$P_1 \leftarrow \pm (A_Ø \pm B_Ø) * C_Ø$
IF $(P_1 \pm E_1) \leq +1.Ø - 2^{-15}$
    THEN IF $(P_1 \pm E_1) \geq -1.Ø$
        THEN $R_{1.75} \leftarrow [(P_1 \pm E_1)]$
        ELSE $R_{1.75} \leftarrow -1.Ø$
    ELSE $R_{1.75} \leftarrow +1.Ø - 2^{-15}$
    WHERE $[(P_1 \pm E_1)]$ DENOTES THE LOW-ORDER 16-BITS OF $(P_1 \pm E_1)$
```

ERROR CONDITIONS:

Positive Out-of-Range Error (POE), if $(P_1 \pm E_1) > +1.Ø - 2^{-15}$

Negative Out-of-Range Error (NOE), if $(P_1 \pm E_1) < -1.Ø$

Out-of-Range Error (ORE), if either POE or NOE occurs.

DESCRIPTION:

The arithmetic operations defined by < EXPRESSION > are performed. The following mixed mode operations between integer (I) and scaled-fraction (F) operands are legal:

a.) ±(F±F)*I±F
b.) ±(I±I)*F±F

Figure 5.9 shows the word length and word format at each step through the flow of operations in the Arithmetic Unit for a.). The word length and word format is the same for b.) as for a.) from the output of MULTIPLY onward.

EXAMPLE:

IA -(-B)*C-D

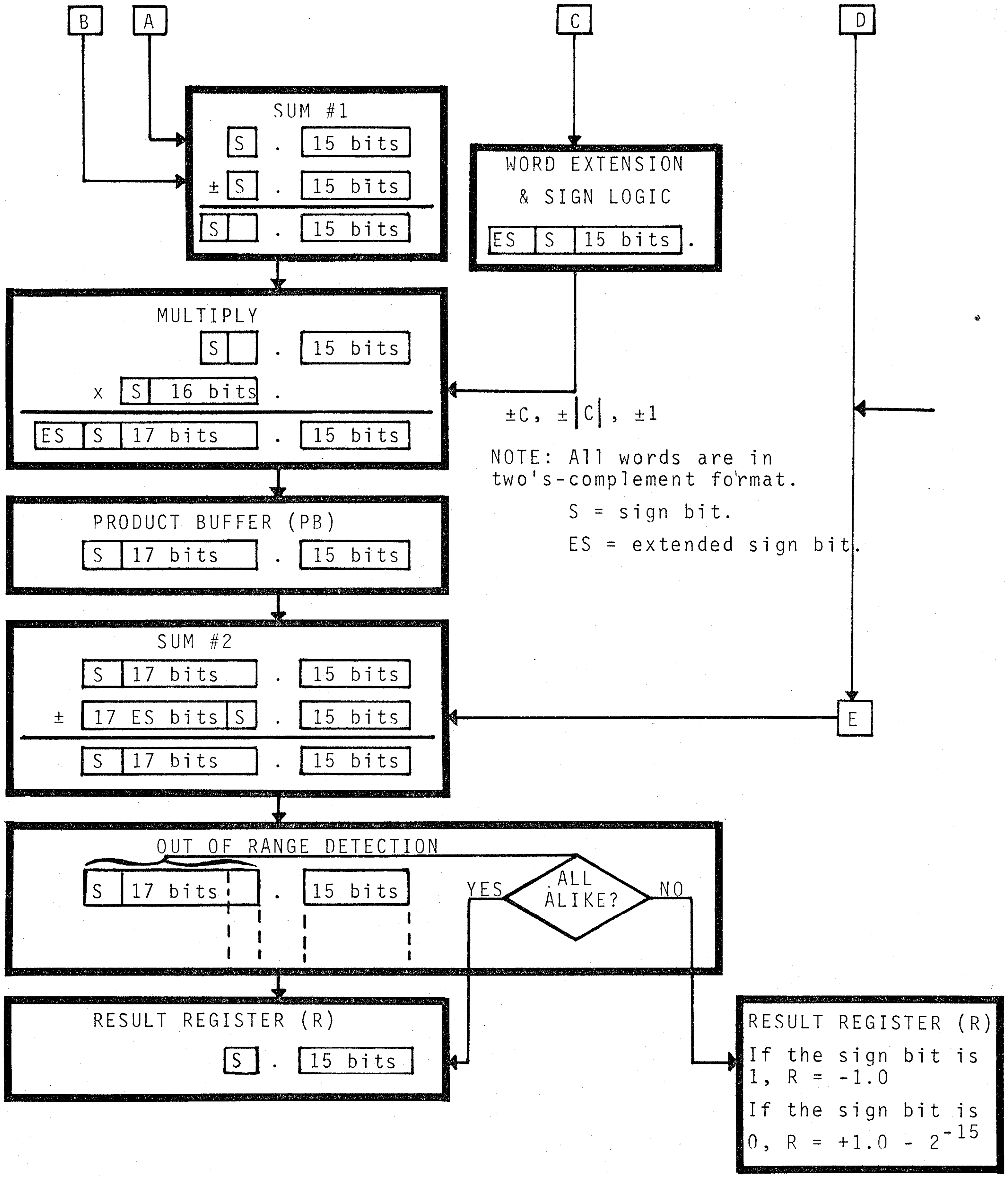


FIGURE 5.9 INTEGER ARITHMETIC (IA) MIXED MODE CASE

5.2.4 ARP MOV SUBINSTRUCTIONS

The ARP has an internal data bus (see Figure 5.2) which interconnects the operand registers A,B,C,D, and E, the Result Register R, the Bus Store Register S, the Bus Load Register L, and the 128 registers T(n) in the Temporary Register File. Up to four data move operations may be made via this internal data bus per instruction cycle. During each of the four internal data move intervals, one data word may be moved from a specified source register (SR) to one or more specified destination registers (DR's). Each data move operation is specified by a MOV subinstruction.

Four of the five fields in the ARP instruction word are used to specify the four possible data move operations which may be performed as part of an ARP instruction.

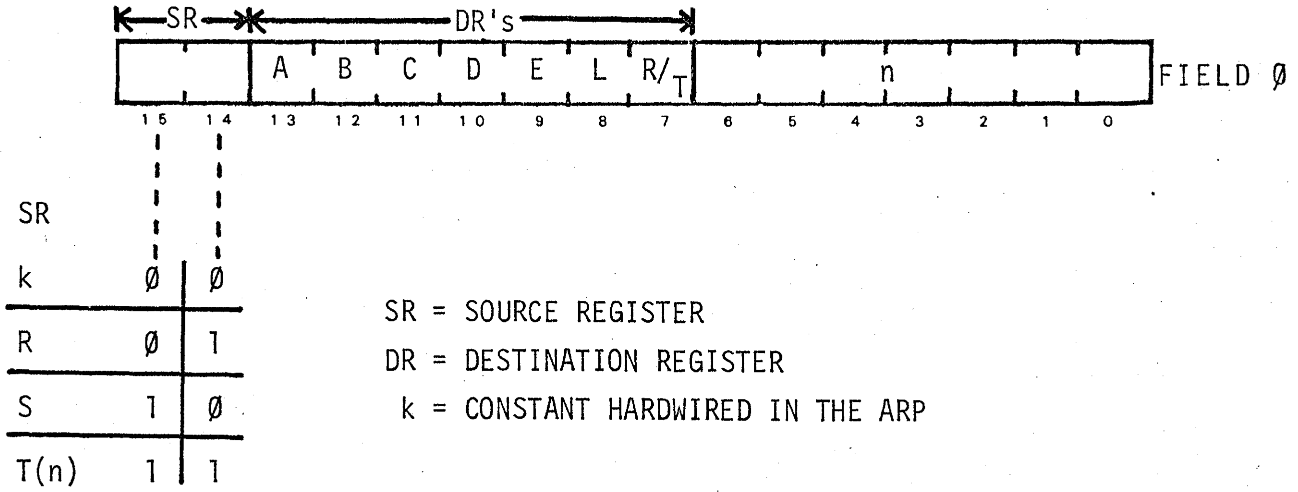
Note: If E is an operand in the Arithmetic subinstruction of an ARP instruction, then data must be moved to E by means of a MOV1, MOV2, or MOV3 subinstruction which is microprogrammed as part of that ARP instruction.

EXAMPLE:

```
    FA (A+B)*C+E; MOV1 S,E
```

5.2.4.1 MOV FIELD FORMAT

The general format of a MOV field in the ARP instruction word is:



SR		
k	Ø	Ø
R	Ø	1
S	1	Ø
T(n)	1	1

SR = SOURCE REGISTER

DR = DESTINATION REGISTER

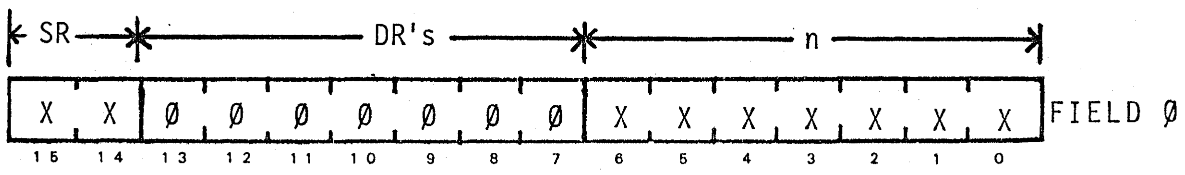
k = CONSTANT HARDWIRED IN THE ARP

BIT(S)	DESCRIPTION
Ø-6	Specify the address, n, of one of the 128 registers, T(n), in the Temporary Register File.
7	Specifies R or T(n) as a destination depending on the SR selected in accordance with the following table:

SR	DR	IMPLIED MOVE
R	T(n)	T(n) ← R
S	R	R ← S
T(n)	R	R ← T(n)

BIT(S)	DESCRIPTION
8	Specifies the Bus Load Register L as a DR.
9	Specifies E as a DR.
10	Specifies D as a DR.
11	Specifies C as a DR.
12	Specifies B as a DR.
13	Specifies A as a DR.
14,15	Specify the SR according to the above table.

The no-op format for a MOV field in the ARP instruction word is:



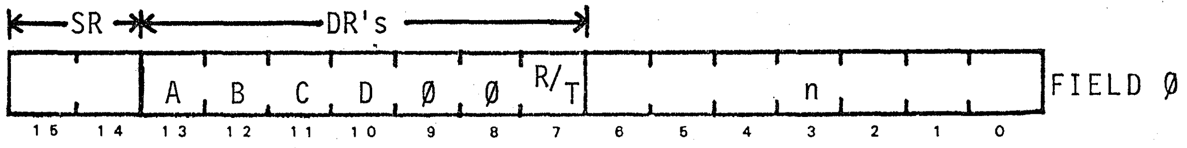
X = Ø,1

5.2.4.2 THE MOV INSTRUCTION

The four ARP MOV instructions are:

MOVØ	Move in subinterval MØ
MOV1	Move in subinterval M1
MOV2	Move in subinterval M2
MOV3	Move in subinterval M3

MOVØ SR, DR[,...[,DR]]



OPERATION:

$DR_{.25}, DR_{.25}, \ldots, DR_{.25} \leftarrow SR_{\emptyset}$

Note: Timing Relationship to the DATA MULTIBUS:

If the SR is S, then:

$DR_{.25}, \ldots, DR_{.25} \leftarrow S_{\emptyset}$

is equivalent to:

$DR_{.25}, \ldots, DR_{.25} \leftarrow DM_{\emptyset}$

ERROR CONDITION(S):

None

DESCRIPTION:

Move the contents of the source register, SR, to a list of destination registers. See paragraph 5.2.4.1 for the definition of the field format.

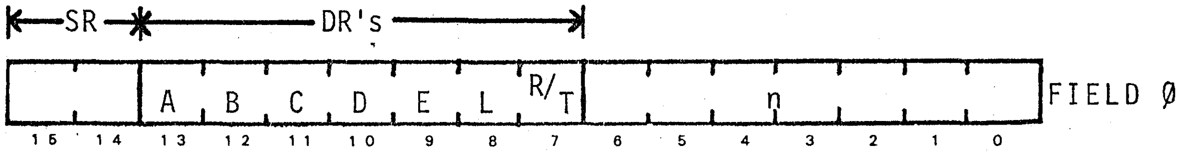
Notes:
1. Since there is an automatic move from D to E during the MØ subinterval, E is not a valid destination in a MOVØ subinstruction.

2. The Bus Load Register L is not a valid destination in a MOVØ subinstruction.

EXAMPLE:

MOVØ R,B,T(n)

MOV1 SR, DR[,...[,DR]]



OPERATION:

$DR_{.5}, DR_{.5}, \ldots, DR_{.5} \leftarrow SR_{.25}$

Notes: Timing Relationships to the DATA MULTIBUS:

1. If the SR is S, then:

   $DR_{.5}, \ldots, DR_{.5} \leftarrow S_{.25}$

   is equivalent to:

   $DR_{.5}, \ldots, DR_{.5} \leftarrow DM_{\emptyset}$.

2. If L is a DR, then:

   $DR_{.5}, \ldots, L_{.5} \leftarrow SR_{.25}$

   is equivalent to:

   $DR_{.5}, \ldots, DM_{1.\emptyset} \leftarrow SR_{.25}$

ERROR CONDITION:

If L is a DR, BUS conflict (DATA).

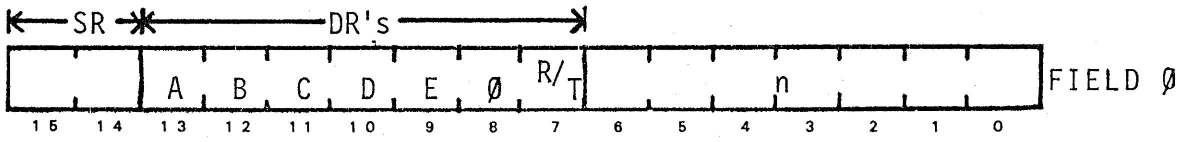
DESCRIPTION:

Move the contents of the source register, SR, to a list of destination registers. See paragraph 5.2.4.1 for definition of the field format.

EXAMPLE:

MOV1 T(n), D,L

MOV2 SR, DR[,...[,DR]]



OPERATION:

$DR_{.75}, DR_{.75}, \ldots, DR_{.75} \leftarrow SR_{.5}$

Note: Timing Relationship to the DATA MULTIBUS:

If the SR is S, then:

$DR_{.75}, \ldots, DR_{.75} \leftarrow S_{.5}$

is equivalent to:

$DR_{.75}, \ldots, DR_{.75} \leftarrow DM_{.5}$

ERROR CONDITION(S):

None

DESCRIPTION:

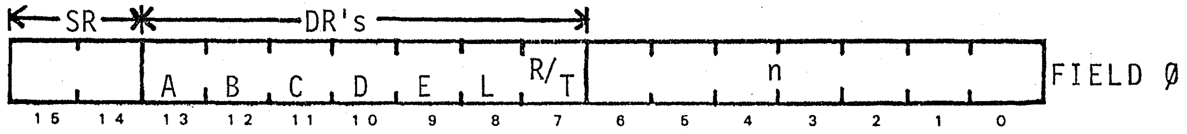
Move the contents of the source register, SR, to a list of destination registers. See paragraph 5.2.4.1 for the definition of the field format.

Note: The Bus Load Register L is not a valid destination in a MOV2 subinstruction.

EXAMPLE:

MOV2 S,B,D,R

MOV3 SR, DR[,...[,DR]]



OPERATION:

$DR_1, DR_1, \ldots, DR_1 \leftarrow SR_{.75}$

Notes: Timing Relationships to the DATA MULTIBUS :

1. If the SR is S, then:

   $DR_1, \ldots, DR_1 \leftarrow S_{.75}$

   is equivalent to:

   $DR_1, \ldots, DR_1 \leftarrow DM_{.5}$

2. If L is a DR, then:

   $DR_1, \ldots, L_1 \leftarrow SR_{:75}$

   is equivalent to:

   $DR_1, \ldots, DM_{1.5} \leftarrow SR_{.75}$

ERROR CONDITION:

If L is a DR, BUS conflict (DATA).

DESCRIPTION:

Move the contents of the source register, SR, to a list of destination registers. See paragraph 5.2.4.1 for the definition of the field format.

EXAMPLE;

MOV3 R,E

5.2.5 PAUSE/NOP INSTRUCTIONS

The PAUSE instruction may be microprogrammed with any other ARP instruction or it may be issued as a stand-alone instruction. The NOP instruction is only used as a stand-alone instruction. The NOP and PAUSE instructions are defined below.

PAUSE d

PAUSE FOR d INSTRUCTION CYCLES

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ø	d			Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	FIELD Ø

OPERATION:

None

ERROR CONDITIONS:

None

DESCRIPTION:

Suspend execution for d instruction cycles following the PAUSE instruction. This has the same effect as (d+1) NOP's.

EXAMPLE:

```
.
.
.
PAUSE 6
.
.
.
```

NOP

NO OPERATION

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

FIELD Ø
Through FIELD 4

OPERATION:

None

ERROR CONDITIONS:

None

DESCRIPTION:

Suspends execution for one instruction cycle.

EXAMPLE:

NOP