AC	MQ	Q	Mnd.
10111	00010	1	sub.] — II
11011	10001	0	ASR.]

Ans

Eg. -9×27

$27 = 011011$

$-9 = 001001$ → take complement (2's)

110111

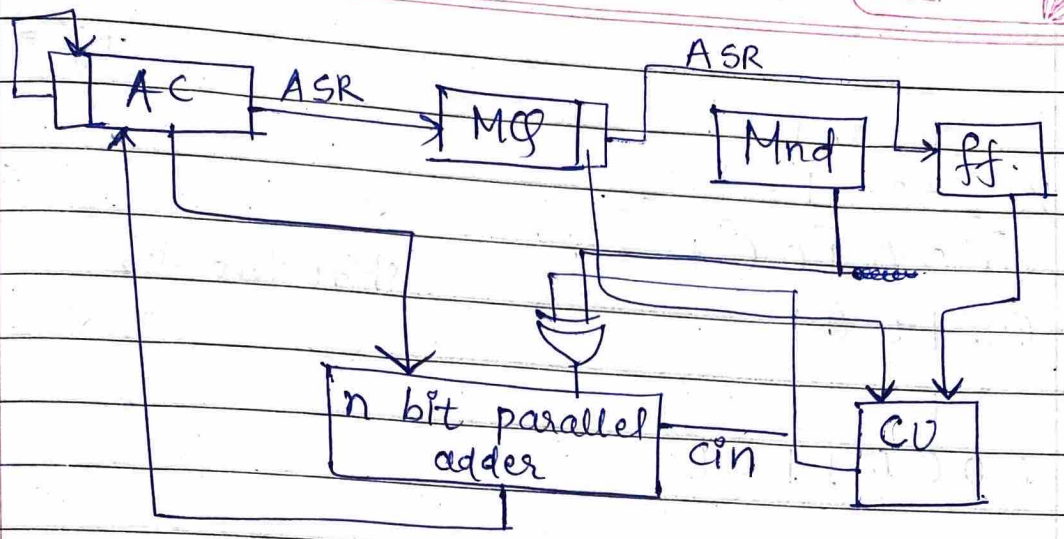
AC	MQ	(ff) Q	Mnd
00000	011011	0	110111
001001	011011	0	SUB
000100	101101	1	ASR ✓ 1
000010	010110	1	ASR ✓ 2
111001	010110	1	Add.
111100	101011	0	ASR ✓ 3
000101	101011	0	SUB
000010	110101	1	ASR ✓ 4
000001	011010	1	ASR ✓ 5
111000	011010	1	Add.
111100	001101	0	ASR ✓ 6

Ans. 111100001101 → 2's comp.

00001101

128 64 32 16 8 4 2 1 = 243

→ Stream adder can be used as substractor



(Boot's Algorithm)

Advantage

(+ve) & (-ve) numbers are treated as same.

eg $27 \times (-9)$

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

0 1 1 0 1 1 (consecutive 1's.)

one greater value of highest power of 1's → $2^5 - 2^3 + 2^2 - 2^0$

$= 27$

Bit pair recoding

0 1 1 0 1 1 0 (group 2)

+1 0 -1 +1 0 -1

$2^1\ 2^0$

0 1 1 0 1 1 0 (group 3)

2 -1 -1

0 1 1

+1 0

$2^1\ 2^0 = 2$.

eg

1 1 0 1 1 1 ←

× 1 1 0 1 1 0

0 -1 +1 0 -1 ← take 2's comp. with

0 0 0 0 1 0 0 1

0 0 0 0 0 0 0 0

1 1 1 1 0 1 1 1

0 0 0 0 1 0 0 1

and so on.

DATE:
PAGE:

```
    1 1 0 1 1 1
X     +2  -1  -1
-------------------
  0 0 0 0 1 0 0 1
0 0 0 0 1 0 0 1
1 1 1 0 1 1 1 0
-----------------
1 1 1 1 0 0 0 0 1 1 0 1
```

+2 = (+1)2

same number and then shift right

shift two bits.

→ Group of 3

000
0 0
1x0+2°x0=0

001
0 +1
2x0 + 2°x1

0	0	0	0XMultiplier
0	0	1	+1 X M
0	1	0	+1 X M
0	1	1	+2 X M
1	0	0	-2 X M
1	0	1	-1 X M
1	1	0	-1 X M
1	1	1	0 X M

eg −15 x 14

14 = 01110

−15 = 01111 → 1's

10000

\+ 1

10001

10001

00111 0 0

+1 0 −2

10001

X +1 0 −2

00000 1110 (−1)X2

00000 0000

11111 0001

1100011110

−2 =

DATE:
PAGE:

Division (Restoring division Algo.)

```
      0001
101 ) 10011011
     -101
     -ve
     +101
       10
      -101
       -ve
      +101
     +0010
      -101
       -ve
      +101
       1001
```

take 1 and subtract

```
 101
-101
(-ve)
```

① AC − DVR = +ve
= 2A − DVR

② AC − DVR = −ve
2 (A + DVR) − DVR
= 2A + DVR

	AC	Divind.	Dvr
	0000	1000	0011
LS	0001	000_ ← don't care	
	1110	0000	
	+0011		
	0001		
	0010	000_	
	−0011		
	1111	0000	
	+0011		
	0010		
LS	0100	000_	
	−0011		
	0001	0001	
LS	0010	001_	
	−0011		

if (−ve)
↳ 1111 + 0011 = 0010 ← Reminder

0010 ← Quo

DATE:
PAGE:

for, Unsigned Bit

Eg Multiply (10) X (13)
M Q

10 = 01010
13 = 01101

C	A	Q	
0	00000	01101	
0	01010	01101	
0	00101	00110	SR
0	00101	00110	
0	00010	10011	SR
0	01100	10011	
0	00110	01001	SR
0	10000	01001	
0	01000	00100	SR
0	01000	00100	
0	00100	00010	SR

```
 00010
 01010
 -----
 01100

 00110
 01010
 -----
 10000
```

Eg Multiply (29) X (21)
M Q

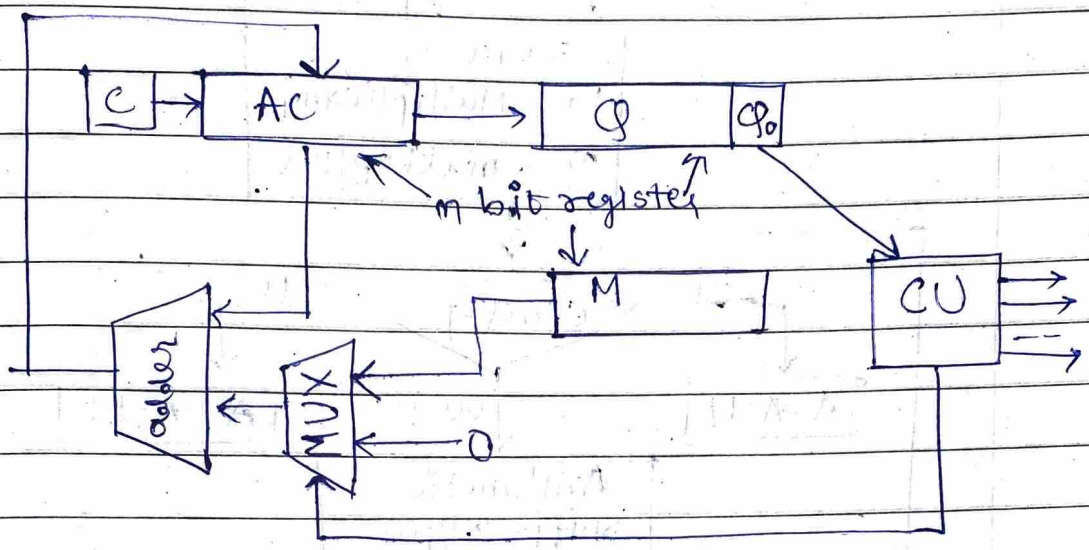
29 = 011101
21 = 010101

C	A	Q	
0	000000	010101	
0	011101	010101	
0	001110	101010	SR
0	001110	101010	
0	000111	010101	SR
0	100100	010101	
0	010010	001010	SR
0	010010	001010	
0	001001	000101	SR
0	100110	000101	
0	010011	000010	SR

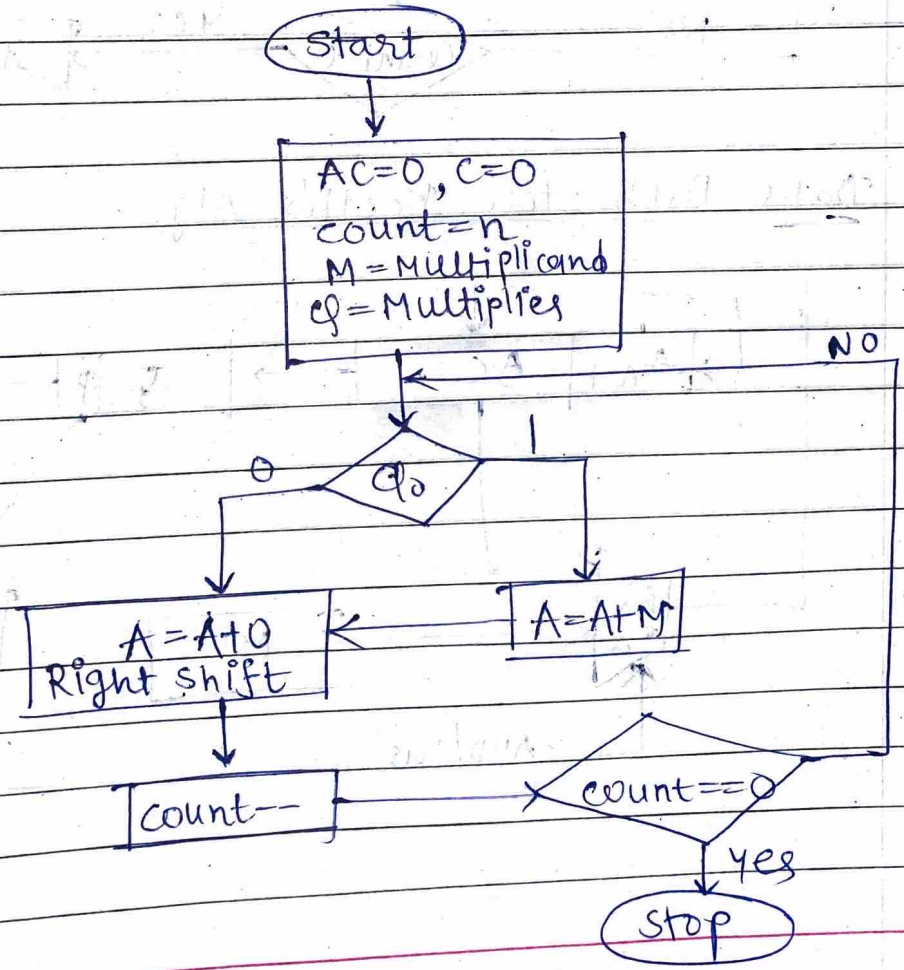
```
  11111
 000111
 011101
 ------
 100100

 001001
 011101
 ------
 100110
```

0	0 1 0 0 1 1	0 0 0 0 1 0	
0	0 0 1 0 0 1	1 0 0 0 0 1	SR

Circuit :

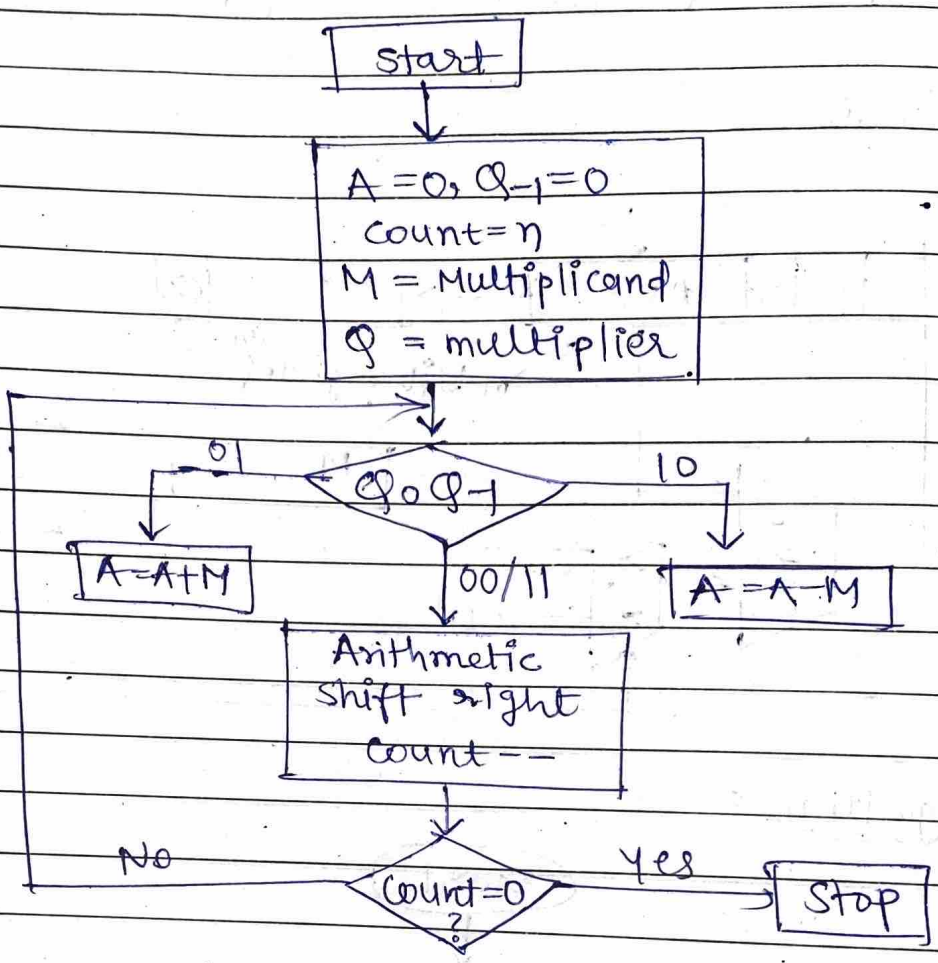


Algorithm :

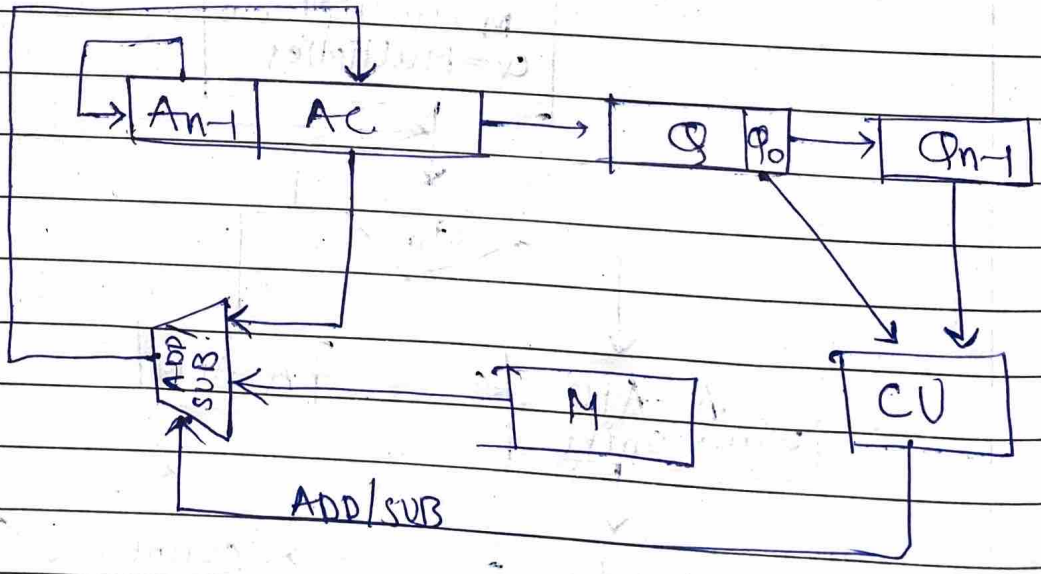


DATE:
PAGE:

Booth's Algorithm



→ Data Path for Booth's Algo.



DATE:
PAGE:

- Using Bit-Pair Recoding of Booth's Multiplication

Eg (+13) X (-22)

13 = 001101 M

-22 = 010110

101001

+ 1

101010 Q

001101

+1 +1 +1

010101

+1 +1 +1

001101

001101

001101

0100010001

→ Non-Restoring division Algo.

8/3. Eg

	AC	(8) Dvnd	(3) dvsr
	0000	1000	0011
LS	0001	000_	
−	0011		
1	1110	0000 ← if -ve put 0	
LS	1100	00_0	
+	0011		
	1111	0000 ← negative	
LS	1110	00_0_	
+	0011		
	0001	000_1	
	0010	001_0	
−	0011		
	1111		
+	0011		
	0010		

DATE:
PAGE:

13/6

AC	Dvnd	Dvsr
0000	1101	0110
0001	101_	
− 0110	101_	
2's comp. → 1011		
0111	010_	
+ 0110		
(-ve) 1101	0100	
1010	100_	
+ 0110		
(+ve) 0000	1001	
0001	001_	
+ 0110		
0111		
− 0110	0010	Quo.
(-ve) 1011		
+ 0110		
0001	reminder if no. is (-ve). else don't add the dvsr	

- Dvd = Q × Dvsr + Rem
  13 = 2 × 6 + 1

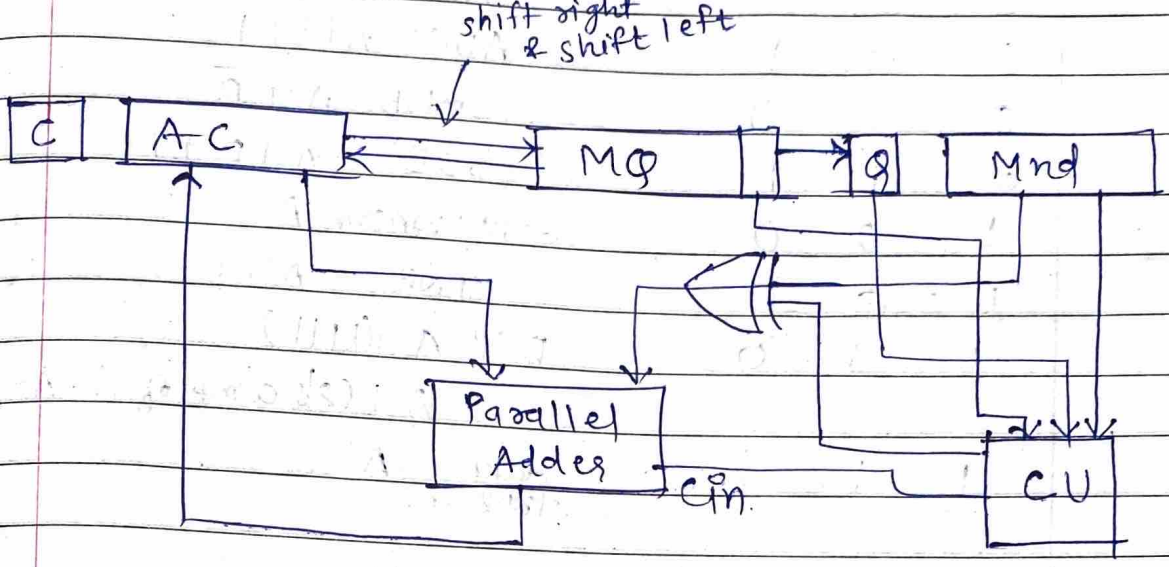
- if 13/(−6) then,
  sign of Q = sign of Divr × sign of dvnd.
  sign of Rem = sign of the dvnd.

eg No.	Dvnd	Q	Dvsr	Rem
7/3	7	2	3	1
-7/3	-7	-2	3	-1
7/-3	+7	-2	-3	+1
-7/-3	-7	+2	-3	-1

Sequential ALU

Assume: fixed point representation

→ Generalized Block diagram of Add., sub, Mul, div. operations.



⟹ Floating point no. can be represented as

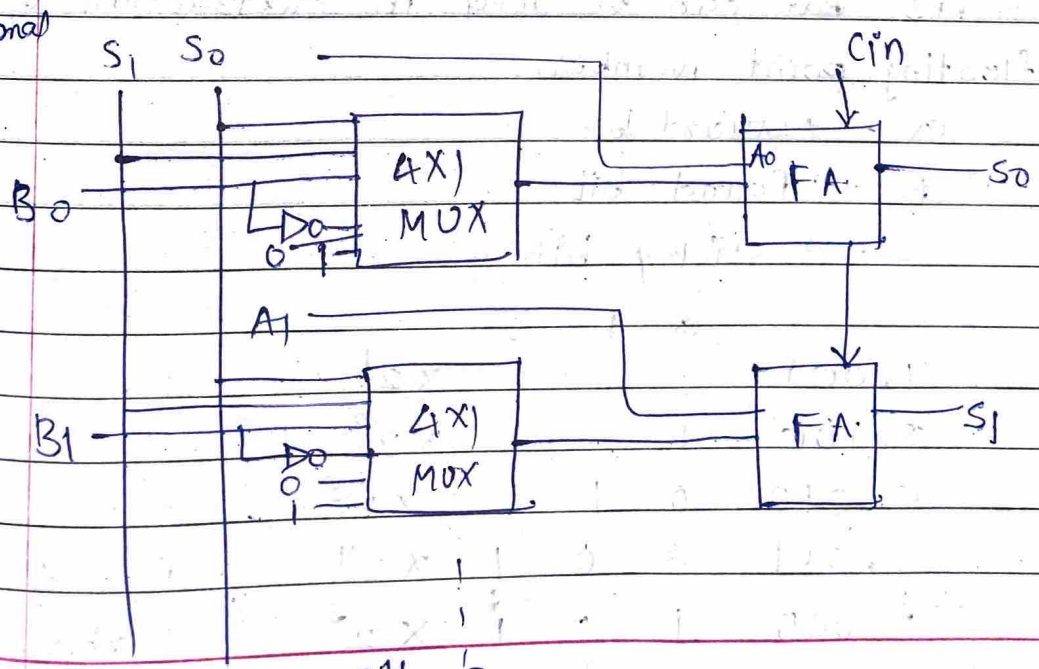
$$X = \underbrace{X_M}_{\text{Mantissa}} \cdot 2^{\overbrace{XE}^{\text{Exponential}}}$$

Assume, $X = X_M \cdot 2^{XE}$

$+ Y = Y_M \cdot 2^{YE}$

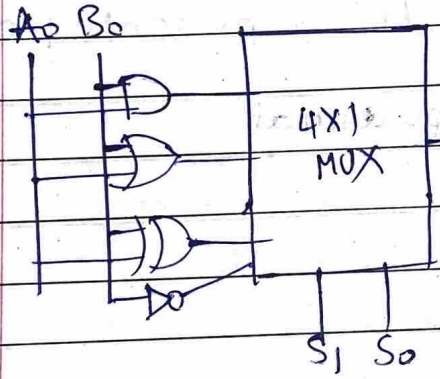
Exponent should be same for add/sub. operation.

⟹ Combinational ALU



Other 2

S_1	S_0	C_{in}	
0	0	0	ADD A+B
0	0	1	ADC A+B+1
0	1	0	SUBB A+$\bar{B}$
0	1	1	SUB A+$\bar{B}$+1
1	0	0	data transfer A
1	0	1	INR A+1
1	1	0	DCR A +(1111)
			A+(2's comp. of 1)=A-1
1	1	1	data transfer A



A_0 and B_0
A_0 or B_0
A_0 XOR B_0

→ 3 ffs can be used in subtraction of two floating point Number.

G → Guard bit
R → Round bit
S → sticky bit

	G	R	S	
1.001				$\times 2^1$
0.100	1			$\times 2^2$
0.010	0	1		$\times 2^3$
0.001	0	0	1	$\times 2^4$
0.000	1	0	1	$\times 2^6$

if 1 is passed through sticky bit then that 1 will be retain

DATE:
PAGE:

→

	G	R	S	
1.000	0	0	0	$\times 2^5$
− 0.000	1	0	1	$\times 2^5$
0.111	0	1	1	$\times 2^5$

2^{-3}
2
2^{-4}

$2^{-1}\ 2^{-2}\ 2^{-3}$ $\quad 2^{-4}\ 2^{-5}$

LS. 1.110 1 1

rounded up → 1.111×2^4

$2^{-4} < 2^{-4} + 2^{-5}$

↳ rounded up

11 → rounded

10 → remain 0 itself

→

	R	S	
1.000	0	0	$\times 2^5$
− 0.000	1	1	$\times 2^5$
0.111	0	1.	$\times 2^5$

not rounded up → $1.110 \quad 1.0 \times 2^4$

Num:

1.001×2^1

	R	S	
1.001			$\times 2^1$
0.100	1		$\times 2^2$
0.010	0	1	$\times 2^3$
0.001	0	1	$\times 2^4$
0.000	1	1	$\times 2^5$

	G	R	S	
1.000	0	0	0	$\times 2^5$
0.000	1	1	1	$\times 2^5$
0.111	0	0	1	$\times 2^5$

$2^{-3}\ 2^{-4}\ 2^{-5}$

$1.110 \quad 0\ 0\ 1 \times 2^4$

1.110×2^4

1.000×2^5
-1.111×2^1

$2^{-4} > 2^{-5}$ ⇒ retain 0.

If 1.0111.1

+ 1

Ans ⇒ 0.100

DATE:
PAGE:

→ In case of excess 3 (Excess k) code.

3 → 6

7 → 10

10+3 16. (we get)

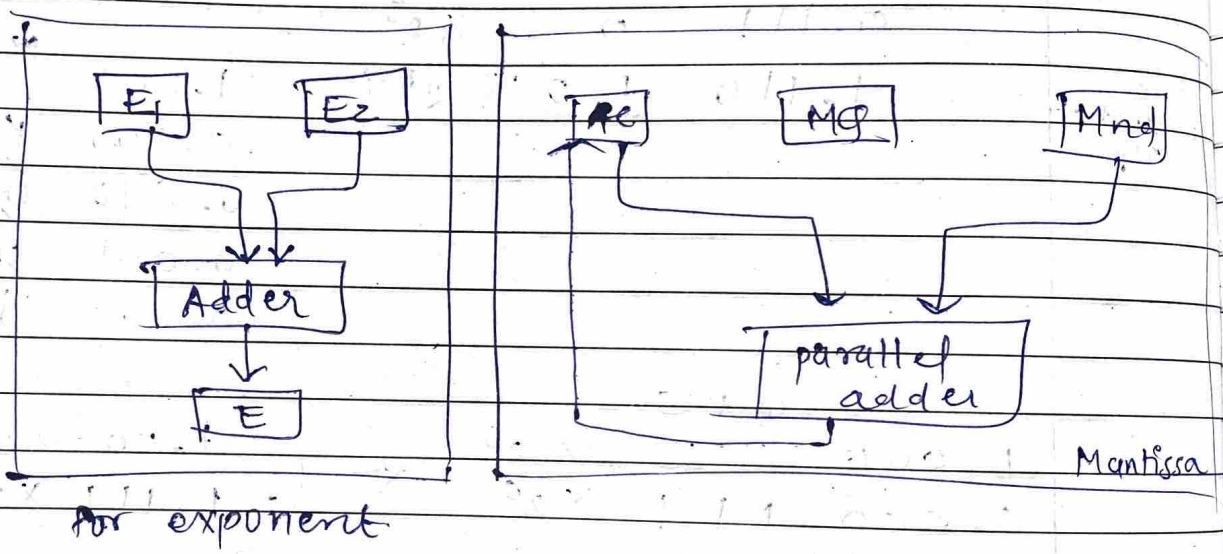
= (13)

↑ right answer

so, we have to subtract 3 from 16. = 13.

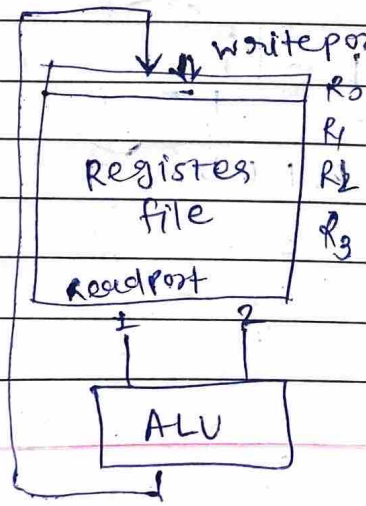
3+3+7+3−3. = 13.

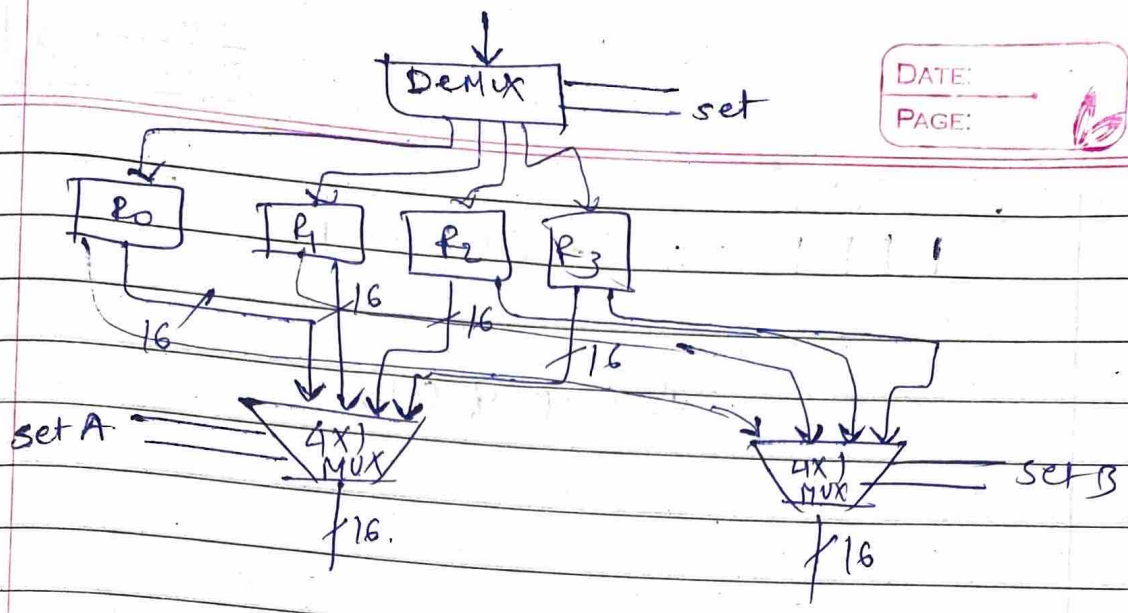
because it would be doubly biased.



→ Other component is register of Processor.

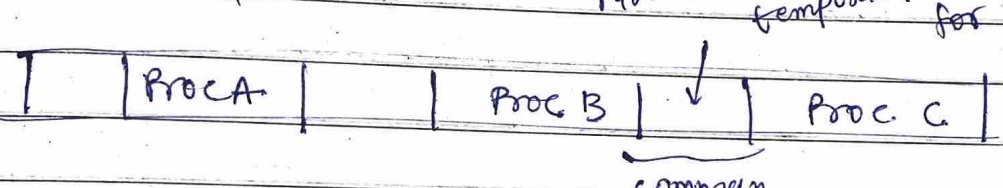
word size = 16 bits





pipelining is easily implemented in RISC processors.

- In RISC, register are divided into windows, these windows are known → overlapped Registers window.



in any prog. no. of arg. proc. called is 8.

current window pointer | 1 | 0 | 0 |

$$-X = \alpha^n - X$$

$$-x = \alpha^{n-1} - X_{n-2} \quad \text{and} \quad +x = -\alpha^{n-1} + X_{n-2} \ldots X_1 X_0$$

2's comp of any no. $= x = -\alpha^{n-1} X_{n-1} + \sum_{i=0}^{n-2} \alpha^i X_i$

$-2^3\ 2^2\ 2^1\ 2^0$

1 1 1 1 $= -8+7 = -1$

2's comp of (-1)