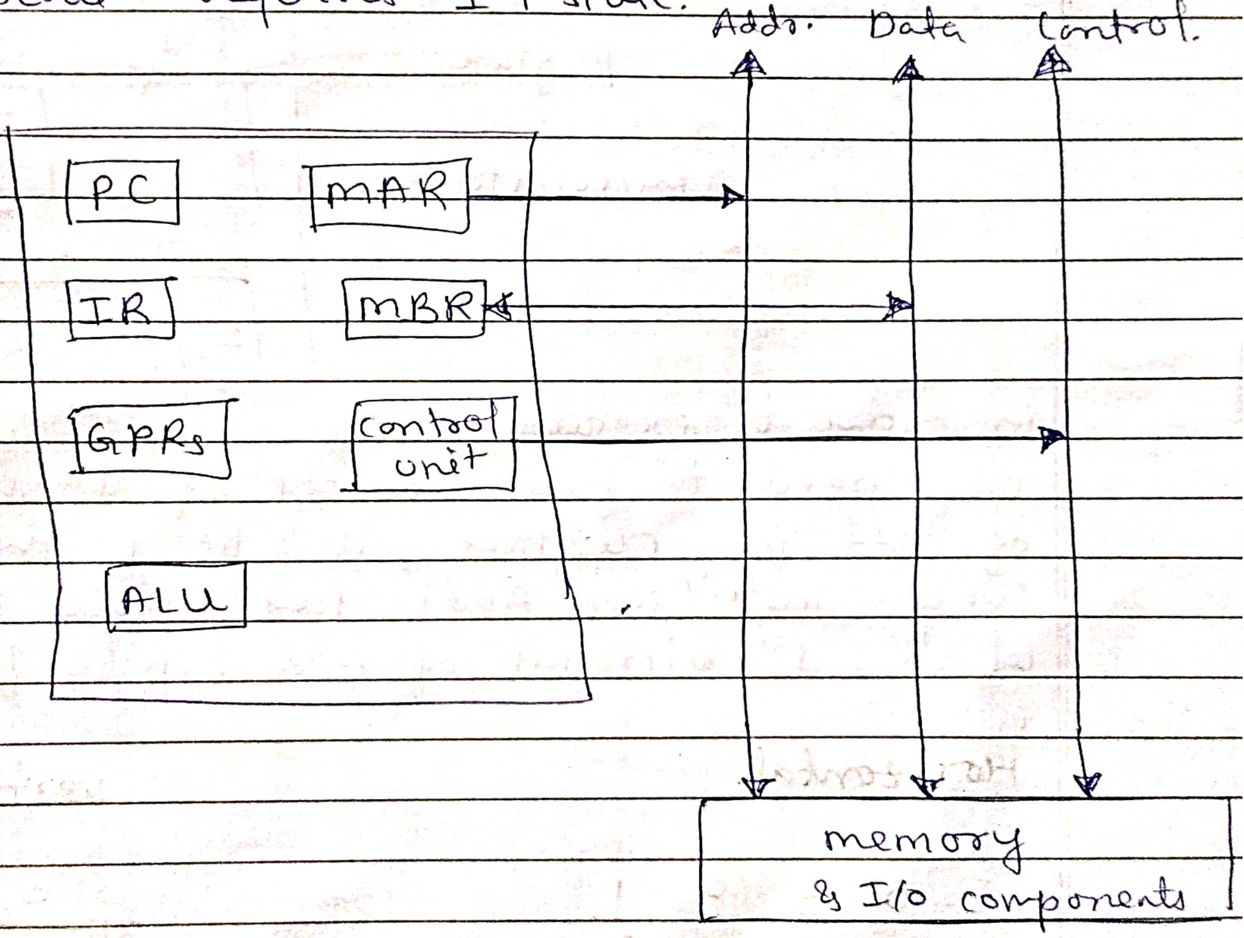
Microoperations

Program is a set of instructions
An instruction requires tiny-tiny activities performed by processor called micro-operations
To perform microoperation we need control signal produced by CU.

- ~~μ-ops for fetching~~.

- 1 μ operat^n requires 1 T-state.



fetching,

T1 : MAR ← PC
T2 : MBR ← memory (instr.)
T3 : IR ← MBR
PC ← PC+1

} 2 μops. takes place in the same T-state as they are completely independent.

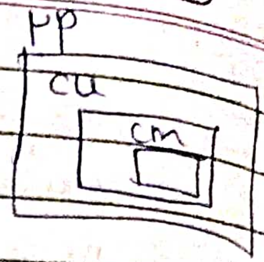
Program

Instr. (Add)

micro-operations

control signals — CU

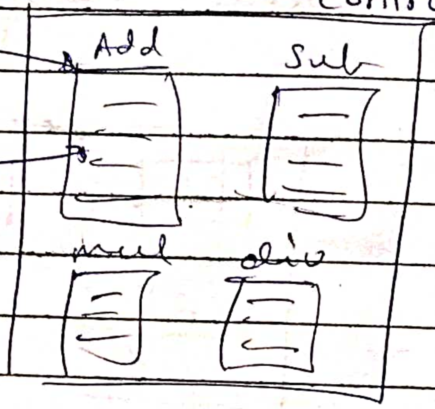
within control unit of processor



control memory

microprogram

microinstr.



In order to execute Add, we need to find address of 1st microinstruct^n of Add in CU. This will be unique.
Every instr. (say Add) has unique opcode using which 1st microinst^n of Add can be found out.

Horizontal.

... 0 0 0 1

~~C~~ C_3 C_2 C_1 C_0

0 0 1 1

C_1 C_0

both at a time can be released.

Vertical

0 0 0 0

decoder 4:16

C_0 C_{15}

only 1 out of 16 can be selected.

Horizontal faster than vertical.
vertical more efficient than horizontal.

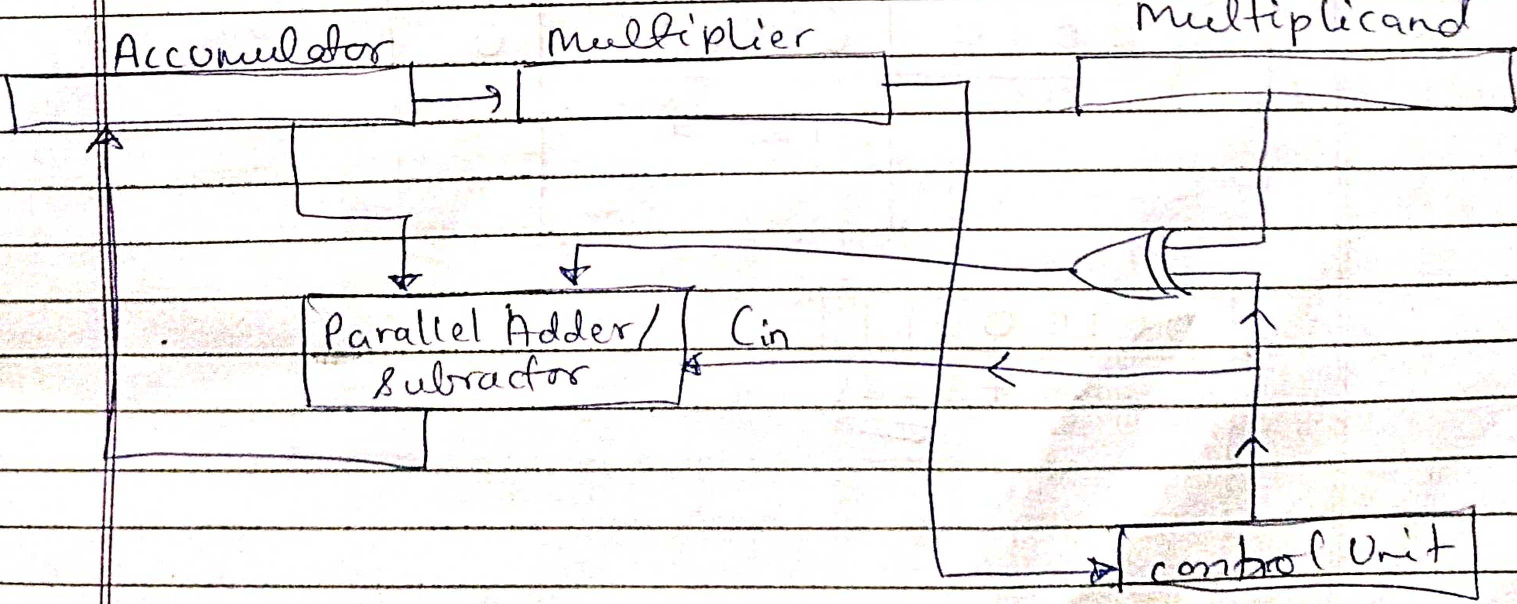
- ## Multiplication & Booth's Algo.

5 → Multiplicand
x 3 → Multiplier
15 → Product.

```
     5       0101
   x 3     x 0011
           ------
             0101
            0101x
           0000xx
          0000xxx
         --------
          0001111   → 15.
```

```
      0000  0000.  → Acc.
 +    0101  0000.
 R.S  0010  1000
 +    0101
      0111  1000
 R.S  0011  1100
 +    0000
      0011  1100
 R.S  0001  1110
      0000
      0001  1110
 R.S  0000  1111   → Acc.
```

Only for unsigned no.'s



Page No.
Date

5	0101		001111
× −3	1101		110001
−15	0101		
	0000×		110001
	0101××		
	0101×××		

$\boxed{1000001} = -63$ ×

−32+16+1

17

1+2+4+8

	A (0) Accumulator	Q (7) Multiplier	Imagine Q_{-1}	(5) Multiplicand
	0000	0111	0	0101
1) 0 to 1 subract M	+1011			
	1011	0111	0	
RS	1101	1011	1	
2) 1 to 1 RS.	1110	1101	1	
3) 1 to 1 RS.	1111	0110	1	
4) 1 to 0 add M	+0101			
	0100	0110	1	
RS.	0010	0011	0.	

$\underbrace{0010 \quad 0011}_{\text{Result}}$

$\boxed{00100011}$

32 + 3 = 35

Ans : -35

	A (0) Accumulator	Q (-7) Multiplier	Q_{-1}	M (5) Multiplicand
	0000	1001	0	0101
1) 0 to 1 subtraction	+ 1011			
	1011	1001	0	
RS	1101	1100	1	
2) 1 to 0 add M	+ 0101			
	① 0010	1100	1	
RS.	0001	0110	0	
3) 0 to 0	0000	1011	0	
RS.				
4) 0 to 1	1011			
sub M	1011	1011	0	
RS	1101	1101	1	

11011101 = -35

-128+

64+16+8+4

+1

64

29

93

128

93

-35

Divisor ↓

left shift!!

Restoration!!! for unsigned

Page No.	
Date	

Dividend ← 4) 64 (1 ← Quotient.

6 _ 4
− 4
2

8) 64 (0

6 _ 4
− 8
−2
Restoration { + 8
6

- 64 / 4. steps needed = (no. of ~~digits~~ bits in dividend). At each step left shift the dividend make an attempt to subract the divisor .... if result is +ve (0 inclusive) the step is successfull, & quotient bit is 1. No restoration req. if the result is −ve &, step is unsuccessful & quotient bit is 0. Restorat^n is performed by adding back the divisor.

	A (0) Accumulator	Q (7) Dividend	Divisor (M) (3)
	0000	0111	0011
1). LS	0000	111_	
sub m	+ 1101		
	1101		
∴ msb = 1 −ve	1101		
∴ unsuccessful	+ 0011		
Restoration	0000	1110	
2) LS	0001	110_	
sub m	+ 1101		
−ve → restorat^n	1110		
	+ 0011	1100	
	0001		
3) LS	0011	100_	
sub m.	+ 1101		
+ve.	0000	1001	

1)

2)

3)

4)

unsigned

Page No.	
Date	

4). LS 0001 001_

Sub. m +1101

-ve → restoration 1110 0010

+0011

0001 0010

remainder quotient

3) 7 (2 → Quotient

−6

1 → remainder.

Non-Restoring for unsigned.

	A(0) Accumulator	Q(7)	M(3)
	0000	0111	0011
1). LS.	0000	111_	
-ve	1101		
unsucc. next add,	1101	1110	
2) ~~Add.~~	~~1101~~		
LS	1011	110_	
Add m	+0011		
-ve unsucc. next add	1110	1100	
3). LS	1101	100_	
Add m	+0011		
+ve succ. next sub.	0000	1001	
4) LS	0001	001_	
Sub m	+1101		
-ve unsucc.	1110	0010	
	+0011		
	0001 → R.	2 → Q.	

	A (0)	Q (6)	M (2)
	0000	0110	0010
1) LS	0000	110_	
sub m	+1110		
-ve unsu.	1110		
next add.		1100.	
2) LS	1101	100_	
add m	+0010		
-ve unsc.	1111		
next add	1111	1000.	
3) LS	1111	000_	
add m	+ 0010		
+ve - succ.	0001		
next sub.	0001	0001	
4) LS	0010	001_	
sub m.	+ 1110		
+ve - succ.	0000	0011	
	R = 0.	3 → Q	

- **Restoring division for signed no.**

No. of steps = no. of bits in the dividend
At each step, perform a LS, then compare the sign of Acc. (A) & Divisor (M). if A & M are of same sign then subract M else add M

Q=0 restorat" req. ← if sign of A changes, then unsuccessfull
else if " of A remains same ~~its successfull~~ (OR)
If the entire Dividend becomes 0 then successfull

↓

Q bit 1
no restorat^n

7 ÷ (-3)
7 = 0111   3 = 0011
-7 = 1001   -3 = 1101

1111 if MSB of Q is 1.
remainder sign = dividend sign
quotient = +ve

Page No.
Date

	A	Q (7)	M (-3)
Initialize	0000	0111	1101
1) LS	0000	111_	
Add m	+ 1101		
sign change,	1101	1110	
unsucc., restore!!	0000	1110	
2) LS	0001	110_	
Add m	+ 1101		
sign change,	1110	110_	
unsucc., restore!!	0001	1100	
3) LS	0011	100_	
Add m	+ 1101		
same sign (A), succ.	1 0000	1001	
4) LS.	0001	001_	
Add m	1101		
sign change	1110		
unsucc.	0001	0010	

R = 1 (sign of dividend.

Q = 2 (always +ve)

~~1101~~
~~1101~~
~~0010~~

1
~~1101~~
~~0011~~
~~0000~~

0011

~~1110~~
~~0011~~
~~0001~~

$-8+4+2+1$
$= -8$

(4)

Initialize	A	Q(-7)	M(3)
	1111	1001	0011
1) LS.	1111	001_	
Add m	+0011		
sign change, uns.	0010		
restore	1111	0010	
2) LS	1110	010_	
Add m	+0011		
sign change, R.	0001		
	1110	0100	
3) LS	1100	100_	
Add m	+0011		
same sign, succ.	1111	1001	
4) LS.	1111	001_	
Add m	+0011		
sign change, R	0010		
	1111	0010	

$-1 \rightarrow R$ $2 \rightarrow Q$

$14 \div (-2)$

Initialize	A	Q(14)	M(-2)
	00000	01110	11110
1) LS	00000	1110_	
	+11110		
	11110		
	00000	11100	
2) LS	00001	1100_	
	+11110		
	11111		
	00001	11000	
3) LS	00011	1000_	
	+11110		
	1x 00001	10001	
4) LS	00011	0001_	
	+11110		
	1x 00001	00011	
5) LS	00010	0011_	
	+11110		
	1x 00000	00111	

0 → R   7 → Q

01110
10010

00010

$(-14) \div 2$

Initialize	A	Q(-14)	M(2)
	11111	10010	00010
1) LS	11111	0010_	
	+ 00010		
	(1)~~x~~ 00001		
	11111	00100	
2) LS	11110	0100_	
	+ 00010		
	(1)~~x~~ 00000		
	11110	01000	
3) LS	11100	1000_	
	+ 00010		
	11110	10001	
4) LS	11101	0001_	
	+ 00010		
	11111	00011	
5) LS	11110	0011_	
	+ 00010		
	(1)~~x~~ 00000	00111	
	~~11110~~	~~00110~~	

Dividend = 0

Q → R

Q → 7

Page No.	
Date	

- Floating point number formats.
  → any no. where the position of the point is not fixed is called a floating point no.

floating point	fixed point no.
73.2	7.32
649	6.49
.847	8.47
5.63	5.63

- Process of converting floating point to fixed point no. is called normalization.

$73.2 \longrightarrow 7.32 \times 10^1$

$649 \longrightarrow 6.49 \times 10^2$

$.847 \longrightarrow 8.47 \times 10^{-1}$

$5.63 \longrightarrow 5.63 \times 10^0$

*Now point can be ignored.

$\underbrace{563}_{\text{mantisa}}$ 0 → exponent → Tells position of point with reference to referenced portion

847 → m
−1 → E

S   m   TE E

$0101.001 \rightarrow (-1)^0\, 1.01001 \times 2^2$

$11111.01 \rightarrow (-1)^0\, 1.111101 \times 2^4$

$0.00101 \rightarrow (-1)^0\, 1.01 \times 2^{-3}$

$-10.01 \rightarrow (-1)^1\, -1.001 \times 2^1$

Normalized form. → $\boxed{(-1)^{S}\, 1.\underline{m} \times 2^{\overline{E}}}$

S → sign → 0 → +ve
1 → −ve

S	E	m
1	1	001

- IEEE - 754 - 32 bit format

(Single precision / Short Real)

→ biased exponent = True exponent (TE) + bias

Bias 127

S	E	m
(1)	(8)	(23)

- IEEE - 754 - 64 bit format

bias 1023

S	E	m
(1)	(11)	(52)

bias value $= \frac{255 \text{ (full range)}}{2} = 127$ {single precision}

bias value $= \frac{2047 \text{ (full range)}}{2} = 1023$ {double precision}

ex:- $0101.001 \rightarrow 1.01001 \times \cancel{10}\, 2^{2}$

in single precision

0	127+2=129	~~1~~ 0100100 ___ 0
S	E (8)	m (23)

ex:- $0.00101 \rightarrow 1.01 \times \cancel{10}\, 2^{-3}$

in double precision

0	1023 − 3 = 1020	0100 ——— 0
S	E (11)	M (52)

① $(14.125)_d \rightarrow$ single precision

61110

1110.001

normalised?

0.125 × 2 = 0.25 (0)
0.25 × 2 = 0.5 (0)
0.5 × 2 = 1.0 (1)

$\underbrace{(-1)^{0}}_{S} \times \underbrace{1.110001}_{m} \times 2^{3 \rightarrow E}$

bias = 127 + 3 = ~~124~~ 130

in binary 124 → 11111100

10000010

0	10000010	1100010...0
S	E	m

64
72
136
16
112
8
4
128
12

② Convert $(2A3B)_H \rightarrow$ single precision.

0010 1010 0011 1011.

$(-1)^{0 \rightarrow S} \times \underbrace{1.0101000111011}_{m} \times 2^{13 \rightarrow E}$

bias E = 127 + 13 = 140.

10001100

0	10001100	0101000111011...0
S	E	m.

Errors in floating point no.'s are called exception

S	E	M
1	8 bit	23 bit

$1.0 \times 2^{-}$

		Exp
TE = 0	BE = 127	OK
TE = 1	BE = 128	OK
TE = 2	BE = 129	OK
⋮	⋮	
TE = 127	BE = 254	OK
TE = 128	BE = 255	1111 1111
TE = 129	BE = 256 (255)	1111 1111 (255)
TE = 130	BE = 257 (255)	1111 1111 (255)
⋮		

	E	M	S
$\infty \rightarrow$	1111 1111	$\underbrace{0000 \ldots 0}_{23\ bit}$	0
$-\infty \rightarrow$	1111 1111	0000 ... 0	1

overflow exception

NaN no. → Not a Number

when all 8 bits of E are 1's.

$0.1 \Rightarrow 1.0 \times 2^{-}$

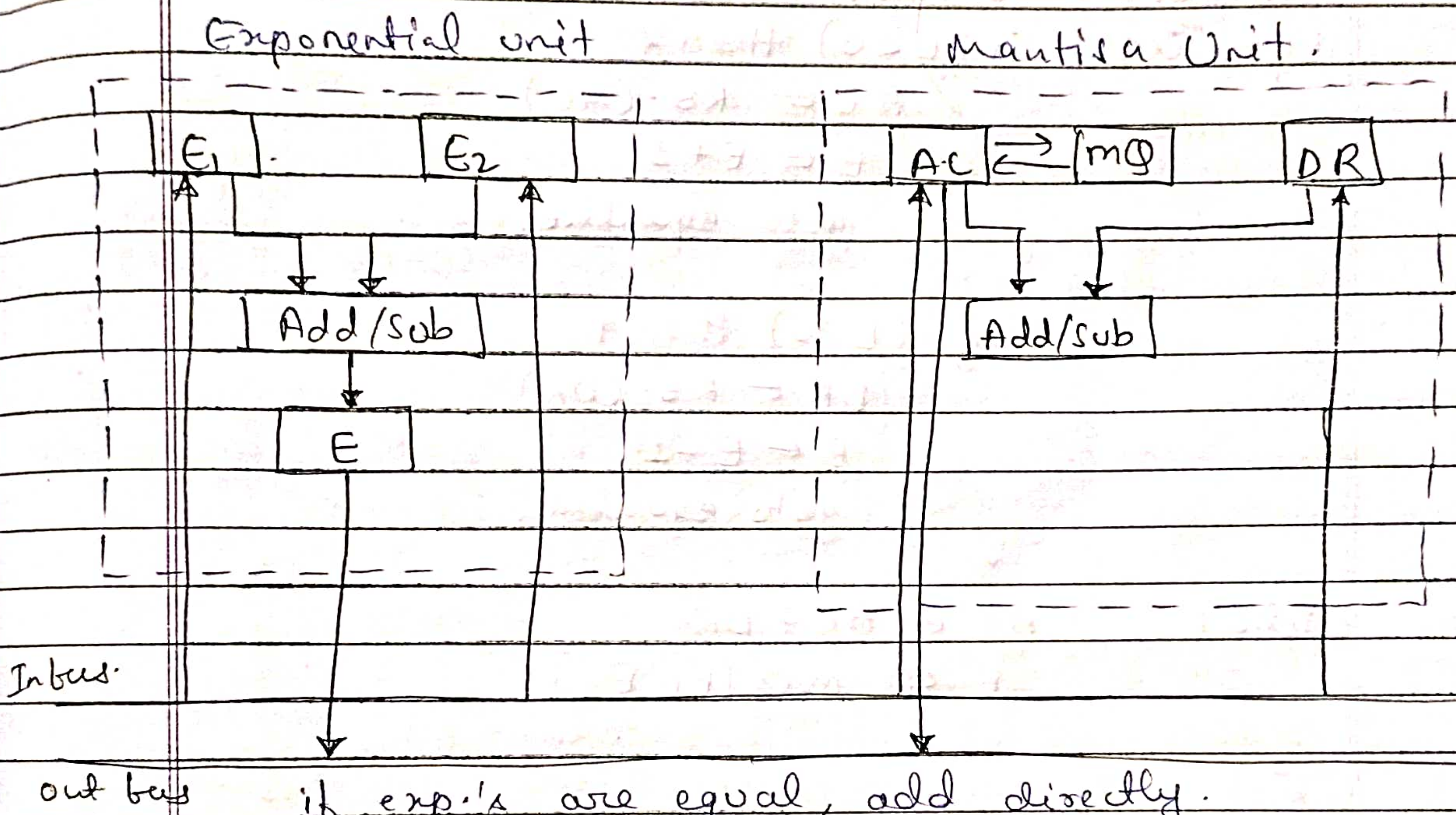
TE = -1	BE = 126	OK
TE = -2	BE = 125	OK
⋮	⋮	
TE = -126	BE = 1	OK
TE = -127	BE = 0	0000 0000
TE = -128	BE = -1 (0)	0000 0000
TE = -129	BE = -2 (0)	0000 0000
TE = -130	BE = -3 (0)	0000 0000
⋮		

underflow exception

DeNormal no.

	S	E	M
ZERO.	0	0000 0000	$\underbrace{0000 \ldots 0}_{23}$

Number	Exception	Exponent	Mantissa
Normal	-None-	$0 < E < 255$	Anything
NaN	Overflow	1111 1111	Anything
∞	Overflow	1111 1111	0000...0
De-Normal	Underflow	00000000	Anything
zero	underflow	00000000	000...0



if exp.'s are equal, add directly.
else make exp's equal then add.

ex. $4 \times 10^3 + 2 \times 10^4$ ↳ get the lower exp. high

i.e. $0.4 \times 10^4 + 2 \times 10^4 = 2.4 \times 10^4$.

Page No.
Date

$X = X_m \cdot 2^{X_E}$
$Y = Y_m \cdot 2^{Y_E}$

Inputs - $AC \leftarrow X_m$; $E_1 \leftarrow X_E$; AC-overflow $\leftarrow 0$
$DR \leftarrow Y_m$; $E_2 \leftarrow Y_E$; Error $\leftarrow 0$.

Compare : $E \leftarrow E_1 - E_2$.
Equalise : if $(E<0)$ then {
$AC \leftarrow RS\cdot(AC)$
$E \leftarrow E+1$
goto equalise.
}
if $(E>0)$ then {
$DR \leftarrow RS\cdot(DR)$
$E \leftarrow E-1$
goto equalise.
}
Add : $AC \leftarrow AC + DR$
$E \leftarrow \max(E_1, E_2)$.

case (I).

AC, DR, E_1, E_2

4×10^{3}
$+\, 2 \times 10^{4}$

$\rightarrow E_1 - E_2 < 0 \Rightarrow E = -1$

$\underbrace{(4 \rightarrow 0\cdot4)}_{R\cdot S.} \times 10^4$

$E = E+1$
$E = 0$.

$AC \leftarrow 0\cdot4 + 2 \Rightarrow AC = 2\cdot4$
$E \leftarrow \max(E_1, E_2) \Rightarrow 10^4$
$2\cdot4 \times 10^4$.

Page No.	
Date	

Case (I)

AC $\leftarrow$ 4 $\times 10^{3}$ $\rightarrow E_1$

DR $\leftarrow$ + 2 $\times 10^{5}$ $\rightarrow E_2$

$E_1 - E_2 < 0 \quad (-2) = E$

$AC \leftarrow R.S(AC)$

$0.4 \times 10^{4} \rightarrow E_1 \qquad E = E+1$

$2 \times 10^{5} \rightarrow E_2 \qquad E = -1$

R.S.

$0.04 \times 10^{5} \qquad \cancel{E_1 - E_2 < 0} \quad E = -1 < 0$

$2 \times 10^{5} \qquad E = E+1$

$2.04 \times 10^{5} \qquad E = 0$.

Case (II)

AC $\leftarrow$ 4 $\times 10^{4}$ $\rightarrow E_1$

DR $\leftarrow$ + 2 $\times 10^{3}$ $\rightarrow E_2$

$E_1 - E_2 = E = 1 > 0$.

$DR \leftarrow RS(DR)$

$\cancel{0.2 \times 10^{4}} \qquad 4 \times 10^{4} \qquad E = E+1$

$0.2 \times 10^{4} \qquad E = 0$

4.2×10^{4}

overflow case $\rightarrow$

4.0×10^{3}

$+ 9.0 \times 10^{3}$

13.0×10^{3}

>1 digits to the left of point

overflow: if (AC-overflow = 1) then {

if (E = Emax) then goto Error;

AC $\leftarrow$ RS. (AC)

E $\leftarrow$ E+1.

goto End.

}

Normalise: (if AC normalised) then

goto End;

Zero : if (AC=0) then E $\leftarrow$ 0;

$$\begin{array}{r} 4.2 \times 10^3 \\ - \; 4.0 \times 10^3 \\ \hline 0.2 \times 10^3 \end{array}$$

$2.0 \times 10^2 \leftarrow 0.2 \times 10^3$ underflow.

Underflow : if ($E > E_{min}$) then {
$AC \leftarrow LS(AC)$
$E \leftarrow E - 1$
goto normalise.
}

$$\begin{array}{r} 4.02 \times 10^3 \\ - \; 4.00 \times 10^3 \\ \hline 0.02 \times 10^3 \end{array}$$

Error : Error $\leftarrow 1$

End : End of process.

- **Memory Hierarchy -**

Primary memory = Physical memory = Main memory.

RAM (Random access memory)
→ Read & write both
→ volatile memory = temporary

ROM. (Read only Memory)
→ U can't write.
→ info. stored in ROM is permanent.
→ Non-volatile memory = permanent mem.

★ RAM dominates over ROM in primary memory.

Pri. mem.

RAM
4-8 GB

ROM
4-8 MB

holds

99.99% primary mem. = RAM

★ BIOS loads the OS from the HDD into the RAM.

when the PC starts, OS has to run, OS (windows) is stored in HDD, OS has to be loaded to RAM so that processor can play it through RAM.

Program that fetches whole info. of OS to RAM.
→ called as ~~booting program~~
BIOS (basic i/p o/p system) program.

Page No.			
Date			

sec. mem.

HD (Hard disk) 1 TB FD (Floppy Disk) 1.44 MB CD (Compact Disk) 700 MB DVD (Digital versatile disc) 4.7 GB	→ Non-Volatile → Writable!! → Thus sec. mem. has characteristics of RAM + ROM. → sec. memory cheaper than primary mem.

RAM/ROM are chip memories.

HD / FD / CD / DVD are disk memories.

HD / FD → magnetic disks.

CD / DVD → optical disks.

* chip memories more expensive than disk memories as they use semiconductor technology while disk uses magnetic fields & uses just a disk.

* RAM are faster than ROM.

RAM → uses semiconductor tech., pure electronic way, hence faster.

ROM → uses disk type, mechanical energy for rotation required, which makes it slow!!

* Operate on fast memory (RAM), storing on cheap memory (ROM).
That's why we need primary & secondary memory both.

* ~~p~~ processor never operates on secondary memory.

SOG

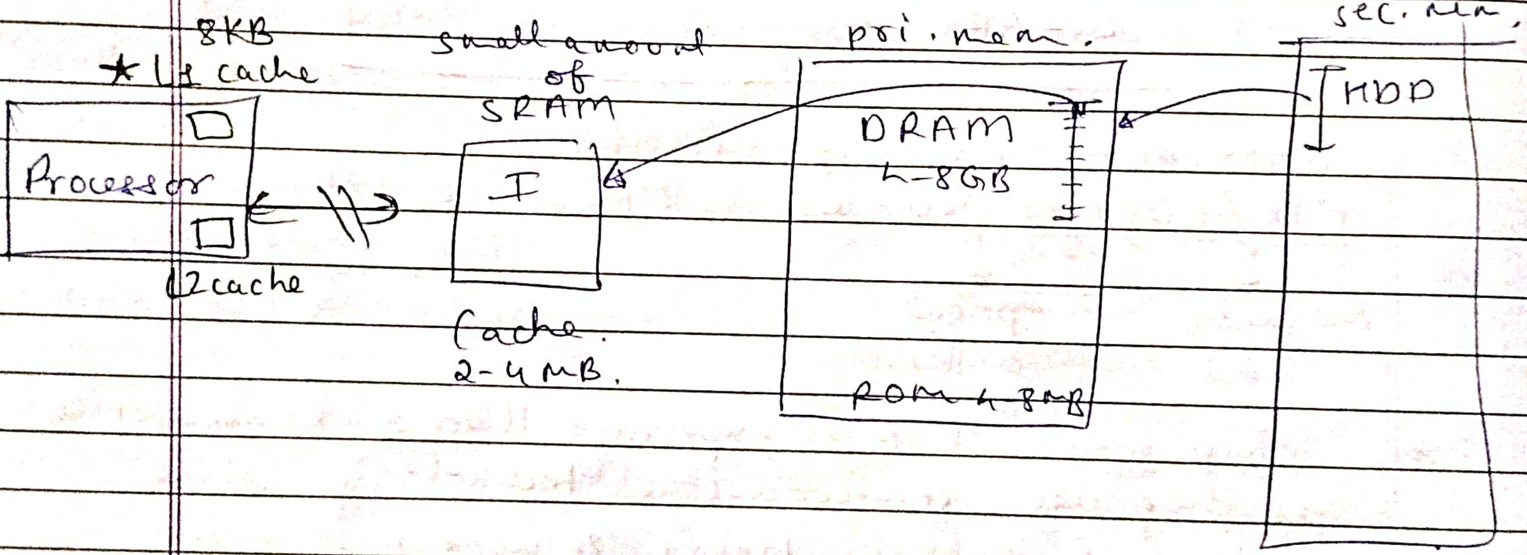
→ memory divided into sections called pages.

SRAM (static RAM)
DRAM (Dynamic RAM)
SDRAM (synchronous dynamic RAM).

* SRAM faster than DRAM,
uses flipflops uses capacitor.

CO7

* SRAM super expensive x100 times DRAM.
↳ lakhs of rupees probably even more.
* So our PC has DRAM!!



* You want information it's stored in bag (secondry memory) from that what you need for your current use you keep it in your pocket (pri. mem.) from that what you need right now is there in your hand (cache!!).

* Pendrive is chip memory
  ↳ more expensive than disk type memories
  uses flash ROM.
  ↳ compact as compared to CD, DVD etc.

- movie stored in sec. mem. say HDD.
  we don't load entire movie into pri. mem. or RAM (coz DRAM needs ~~other~~ some space to do other activities also).
  Therefore, memory is divided in sections called pages. These pages are loaded in DRAM & executed.
  So whenever we play a movie, whole movie is not available at that very instant, only ~~2-3~~ 3-5 minutes are loaded & as these minutes ~~gets~~ starts getting exhausted. for ex. only 1 min. of watch is left, another page will be loaded in RAM & it goes on.
  In DRAM this page is not entirely utilized at one time, user sees 10 sec. of movie not whole 2-3 min.
  So these 10 secs of that page is loaded to cache & processor pics it from cache as it is faster (SRAM).
  What if cache takes directly data from sec. storage?
  It will take 10 secs after ending it has to wait for loading another 10 secs from sec. storage. which is a very big NO-NO!!
  This problem is solved by primary mem. or DRAM.

• Virtual Memory Management using Paging

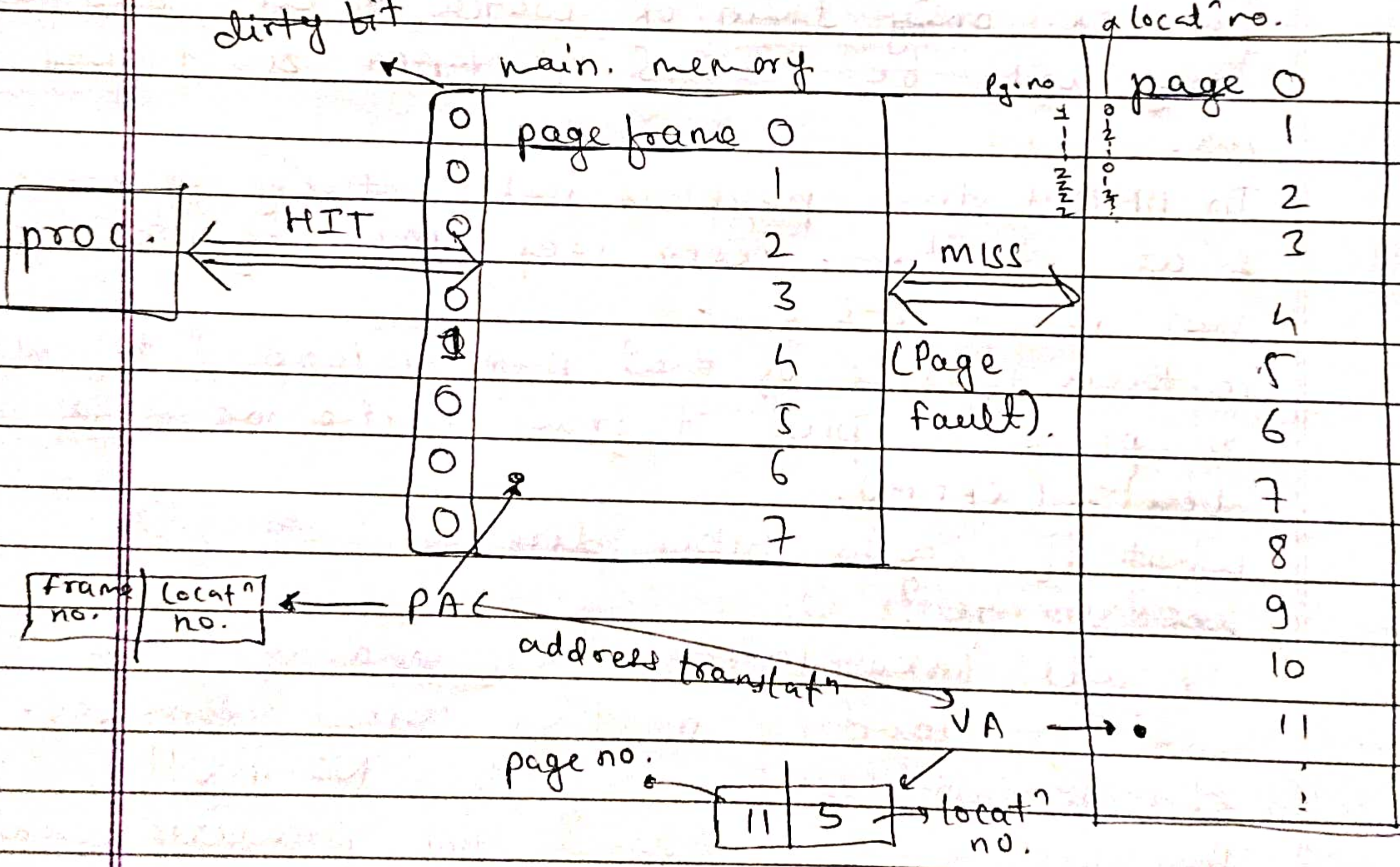
Page No.
Date

- Info. is stored in sec. memory from where via virtual memory management it comes into main memory at that time paging is done.

- Paging is a V.M management system, it tells how information comes from V.M into physical memory.

- Typical size of a page – 4KB.

VM divided into equal blocks called pages.
main. memory divided into equal blocks called page frames
Page frame & page are of same size



- Whenever pp wants info., it'll first try its luck to find the info. in main. mem. itself, if it gets it in the main. mem. that event is called a HIT.

Page No. | Date

If it doesn't get from main mem. then processor goes to the ~~s~~ V.M. that event is called a MISS (Page fault) →

(N1) HIT: desired page available in main memory.
(N2) MISS (Page fault): when desired page not available in main memory, ~~the~~ pp goes to VM copies page to main mem.

- $\text{HIT ratio} = \dfrac{\text{no. of hits}}{\text{total attempts}} = \dfrac{N1}{N1+N2}$.

in most computer HIT ratio is 95–97% (0.95–0.97)

- suppose page (0–7) are loaded to Page frame (0–7) now page 8 is supposed to be load but main mem. is completely occupied. Processor scans all the pages over main mem. & sees which page is no longer relevant/required, that page will be removed from main mem. & page 8 will be loaded, this is page replacement.

→ Whenever there is a page fault and there are no empty frames available in the main memory and old page from main memory is replaced by a new page from VM. This is called page replacement

- By default all dirty bits are 0. As long as read operation are happening it'll remain 0 but as soon as write happens to some page i.e. it gets modified, the dirty bit corresponding to such page becomes 1 & such a page is known as dirty page.

MIS T1 | MIS T2 Academic

person

A	B	C
1	x	y
2	a	b
3	c	d

join $\times$

A	E	F
1	P	Q
2	R	S
3	T	U

Page No.	
Date	

50 ← 100 $\times$ → 100 natural join.

select A from T1, T2;
where T1.A = T2.A

cash of T1 & T2

T1.A G T2.A G

	A	B	C	A	E	F
R1	1	x	y	1	P	Q
R2	1	x	y	2	R	S
	1	x	y	3	T	U
	2	a	b	1	P	Q
	2	a	b	2	R	S
	2	a	b	3	T	U

⟹ R

frequency

Page frame 0	5
1	9
2	1
3	4
4	7
5	23
6	18
7.	14

← page 2 gets replaced in LFU.

- Page replacement policies

i) FIFO- Page that came in first will go out first. oldest page from main mem. will get replaced. fair but not the best!!

ii) Least Recently Used (LRU)-
least recently used page is replaced!!

iii) Least frequetly used (LFU)- (Algo. used in engj's)

- Thrashing- hit ratio is dropped drastically.
sol^n- change page replacement policies

Page Table (address translation)

Frame no.	Page no.

- Translation lookaside buffer (TLB) -
Page table present in physical memory.
Most recently used 32 entries from page table copied to processor in small table called translation Lookaside buffer (TLB).
Processor simultaneously searches the TLB & page table. (lookaside).

Virtual address (logical address)

Page no. desired	Location within page
(20)	(18)

Page	Table
–	–
–	–
20	35
–	–
–	–
–	–

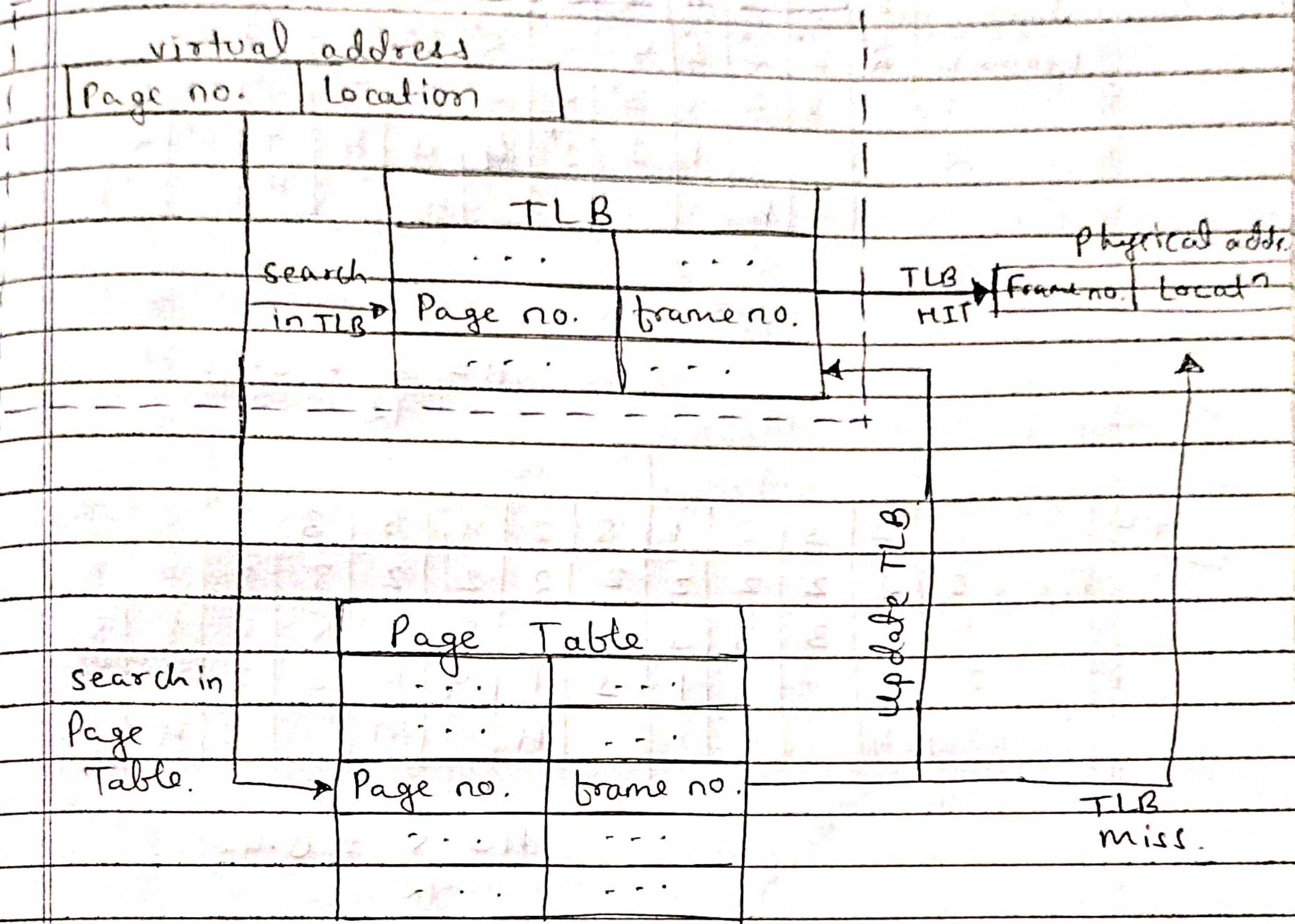
Physical address

Page frame no.	Location within page
(35)	(18)

- Paging Mechanism.

Page Table / TLB used for address translation gives mapping of virtual addr. (page no.) & physical addr. (page frame no.)

• Translation look-aside buffer (TLB)



Principle of temporal locality!!

FIFO

	2	3	2	1	5	2	4	5	3	2	5	2
Frame 0	2*	2*	2*	2*	5	5	5*	5*	3	3	3	3*
1		3	3	3	3*	2	2	2	2*	2*	5	5
2				1	1	1*	4	4	4	4	4*	2
			H					H		H		

↑ Hit.

$$H = \frac{3}{12} = 0.25$$

LRU

	2	3	2	1	5	2	4	5	3	2	5	2
frame 0	2	2	2	2	2	2	2	2	3	3	3	3
1		3	3	3	5	5	5	5	5	5	5	5
2				1	1	1	4	4	4	2	2	2
			H			H		H			H	H

$$H = \frac{5}{12} = 0.42...$$

LFU

	2	3	2	1	5	2	4	5	3	2	5	2
frame 0	2^1	2^1	2^2	2^2	2^2	2^3	2^3	2^3	2^3	2^4	2^4	2^5
1		3^1	3^1	3^1	5^1	5^1	5^1	5^2	5^2	5^2	5^3	5^3
2				1^1	1^1	1^1	4^1	4^1	3^{*1}	3^1	3^1	3^1
			H			H		H		H	H	H

$$H = \frac{6}{12} = 0.5.$$

OPT

	2	3	2	1	5	2	4	5	3	2	5	2
Frame 0	2	2	2	2	2	2	4	4	4	2	2	2
1		3	3	3	3	3	3	3	3	3	3	3
2				1	5	5	5	5	5	5	5	5
			H			H		H	H		H	H

$$H = 6/12 = 0.5.$$

Memory Interleaving

Interleaving - Having multiple memory chips to form one big memory.

higher order -

A_2	A_1 A_0	
0	0 0	0
0	0 1	1
0	1 0	2
0	1 1	3

A_2 | $A_1 A_0$

A_2	A_1 A_0	
1	0 0	4
1	0 1	5
1	1 0	6
1	1 1	7

lower order -

odd bank.

0 0	1
0 1	1
1 0	1
1 1	1

Even bank

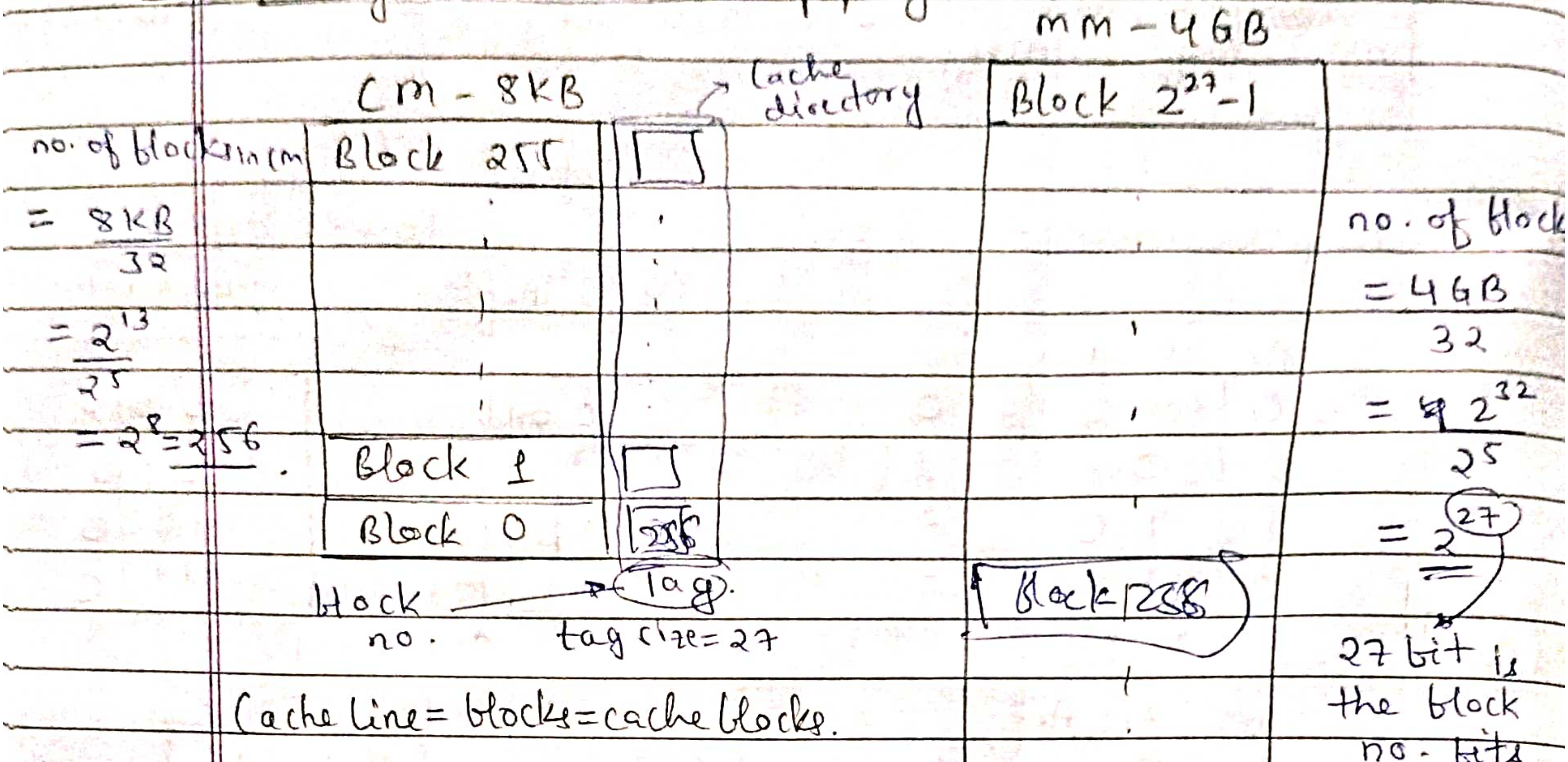
0 0	0
0 1	0
1 0	0
1 1	0

$A_2 A_1$ | A_0

Principle of spatial locality.

* Cache mapping Techniques -

1. Fully Associative Mapping -



Cache Line = blocks = cache blocks.

m.m addr. (32)

Block no. (27)	Locatn (5)

→ Fully associative means, full cache memory is associative. Any block of m.m can be mapped into any block of cache memory.

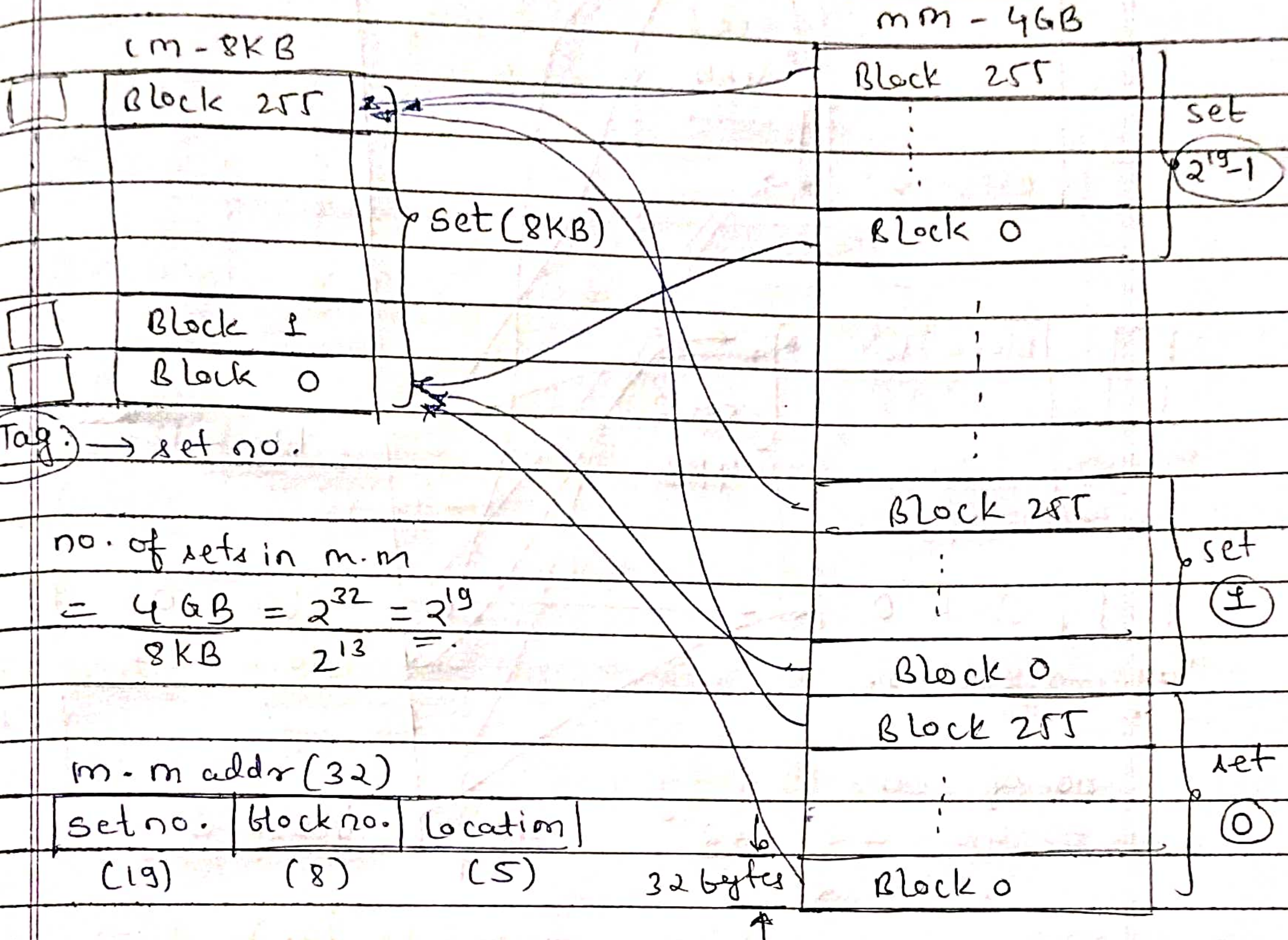
→ most flexible hence most slowest!

Tag → will tell you, which block of m.m has been mapped with given cache block.

Collection of tags is called Cache directory. → present in cache memory.

- Tag size = 27 bit } strongest
- Searches = 256. } drawbacks.
- Hit Ratio = } Best possible hit ratio ~~along~~ amongst all.

2) One way set associative (Direct mapping) -



no. of sets in m.m

$$= \frac{4\,GB}{8\,KB} = \frac{2^{32}}{2^{13}} = 2^{19}$$

m.m addr (32)

set no.	block no.	location
(19)	(8)	(5)

- Tag - ~~8 bit~~ 19 bit
- Searches - 1
- Hit ratio - ↳ Has worst hit ratio.

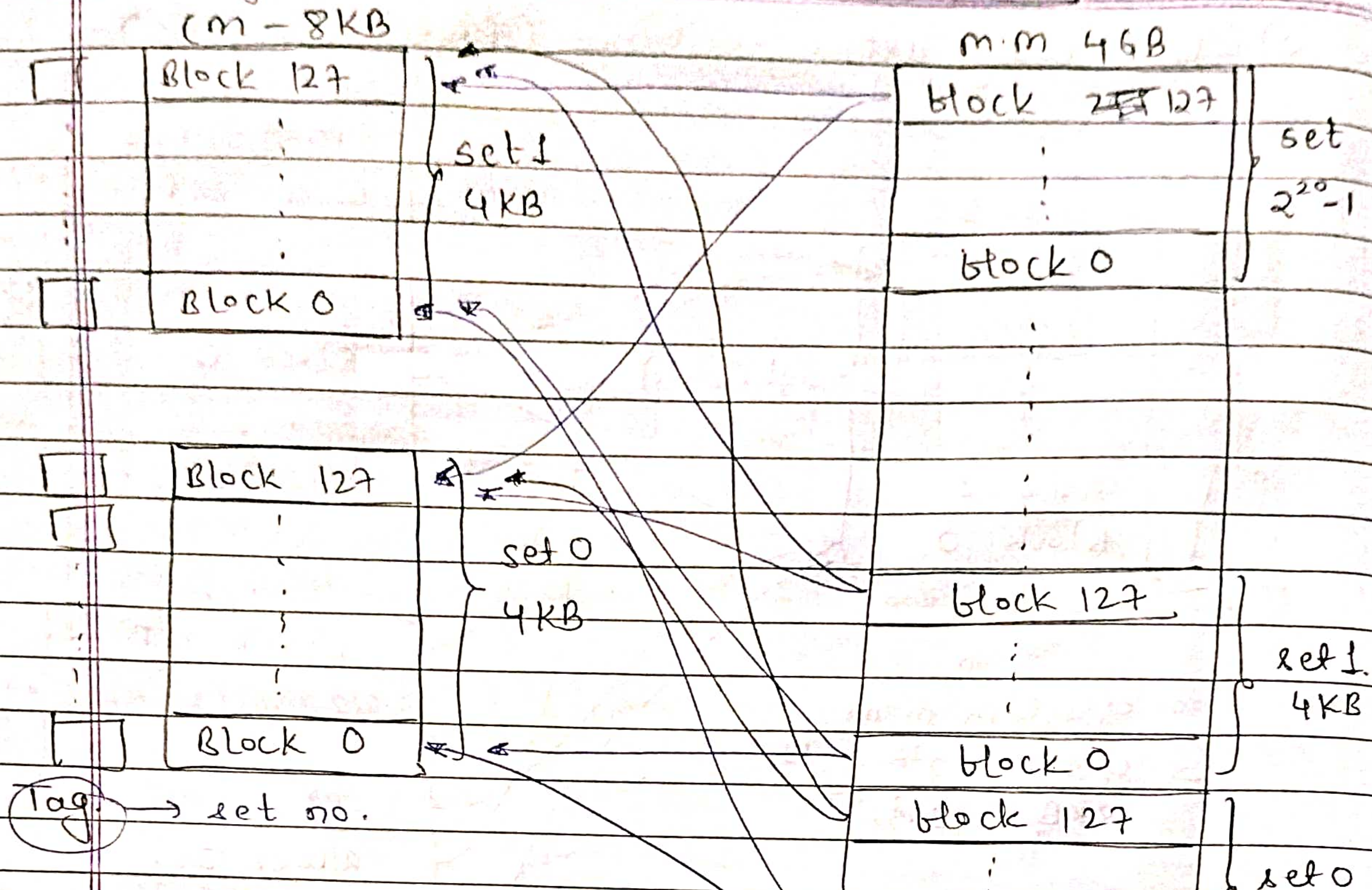
32 bit
↑
1 block → 4 byte.
$2 \times 2^{22} \cdot 2^{24}$

$$m.m = \frac{2^{24}}{2^{2}} = 2^{22}$$

2^{24} → total size. 2^{22} → no. of blocks. 2^{2} → 1 block size.

mm. address: | 22 bit | word (2 bit) |

3) 2-way set associative (Pentium's Cache)



no. of sets -

$$= \frac{4GB}{4KB} = \frac{2^{32}}{2^{12}} = 2^{20}$$

m.m addr. (32)

set	blockno.	Locat^n
(20)	(7)	(5)

- Tag = 20 bit.
- Searches = 2.
- Hit ratio = ↳ way better than one way set associative!!

$$\frac{2^{13}}{2^5} = 2^8$$

256

set	block no.	locat^n.
11	11 bit.	2 bit.

$2^7 \times 2^5$ 128+128

Conclusion -

	Searches	Tag size
2 way	2	20
4 way	4	21
8 way	8	22
16 way	16	23
32 way	32	24
64 way	64	25
128 way	128	26
256 way.	256	27. → fully associative!!

* Bus Contention | Bus Arbitration | Priority Scheme -

- Multiprocessor system → Closely coupled system
  → Loosely coupled system

Closely - multiple processors like 8086 & 8087 use a common bus.

Loosely - Every processor has its own local bus to connect with Local memory & IO devices & all the processor which is the modules basically are connected to each other via system bus.

$\frac{2^{14}}{2^2} = 2^{12}$

4096 → 2048

$\boxed{2048}$ 2048.

$2^{11} \times 2^2$

m.m → 16 MB → 2^{24}

c.m → 16KB → 2^{14}

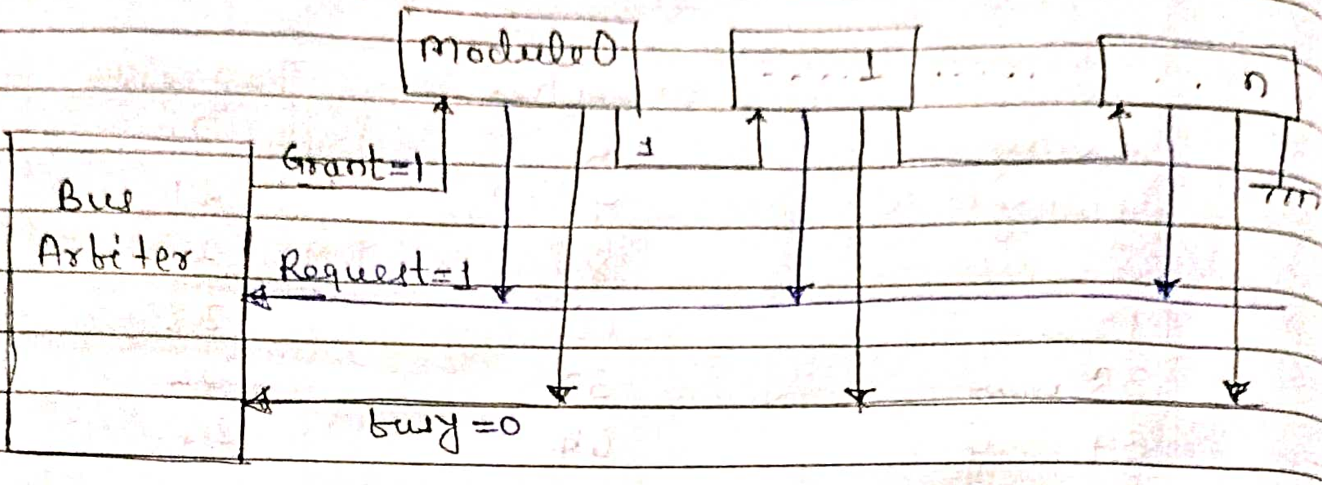
block → → 2^2

$= \frac{2^{24}}{2^{14}} \quad \frac{2^{\cancel{2}}}{\cancel{2}^{12}}$ = 10.

$\frac{2^{13}}{}$ $\frac{2^{13}}{}$.

very usefull for small no. of systems

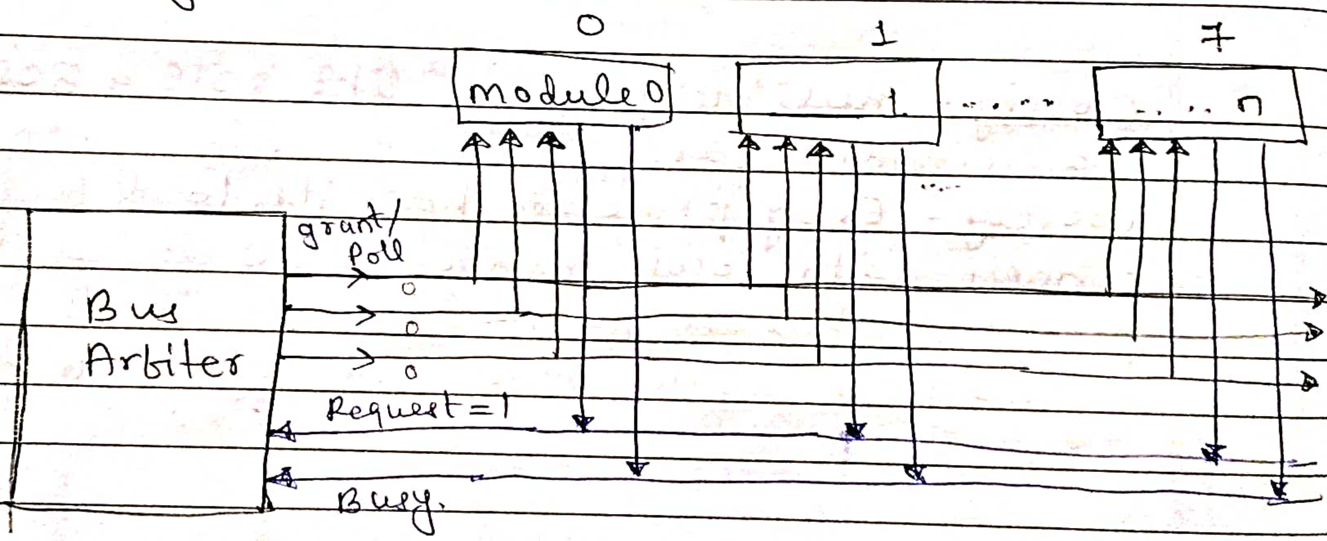
1. Daisy chaining -



- Advantage -
  → Simplest.

- Drawback -
  → Circulat^n of grant is slow process. (Poor Performance)
  → Poor reliability.
  → Poor priority mechanism.

2. Polling - → asking!



- Advantage -

- Drawback -

01011 → 11

1 11011 → (-5)

-55

(-5)

(11) - 16

00101

11011 → (_ _ _ _)

0101
1011
2 8

1 1 1 1 0 0 1 0 0 1

-5

11011 0

0-1+1 0-1

1 1 0 1 1

0 1 1 0 1 1 1

1 00 1 0 0 1

0 → 32 → 0 → 1 32
1 → 0 → +1 16
48
0 → 0 → 0 4
1 → 1 → 0 52
11011
0-1+1 0-1

11010

0-1+1 -1 0

5

10 2×n

01011 +11

× 0-1 1 0-1 (-5) 0001+0001

(5) 010

100

1 1 1 1 1 1 0 1 0 1
0 0 0 0 0 0 0 0 0 x
0 0 0 0 1 0 1 1 x x
1 1 1 0 1 0 1 x x x
0 0 0 0 0 0 x x x x
1 1 1 1 0 0 1 0 0 1 (-55)

Booth's

11

01011 - 01011

10101

10000 -32

10000 +32

0110111

32+16+4+2+1

48+7 = 55

0 1 0 1 0

1 0 1 1 0

A	Q	Q_{-1}	M
0000	1101	0	0111
1001			
1) 10-			
sub. M 1001	1101		
R.S. 1100	1110	1	
2) add 0111			
m 0011	1110		
RS 0001	1111	0	
3) sub M 1001			
1010	1111 RS		
RS 1101	0111	1	
4) RS 1110	1011		

1001

2^5

2^5

00010101

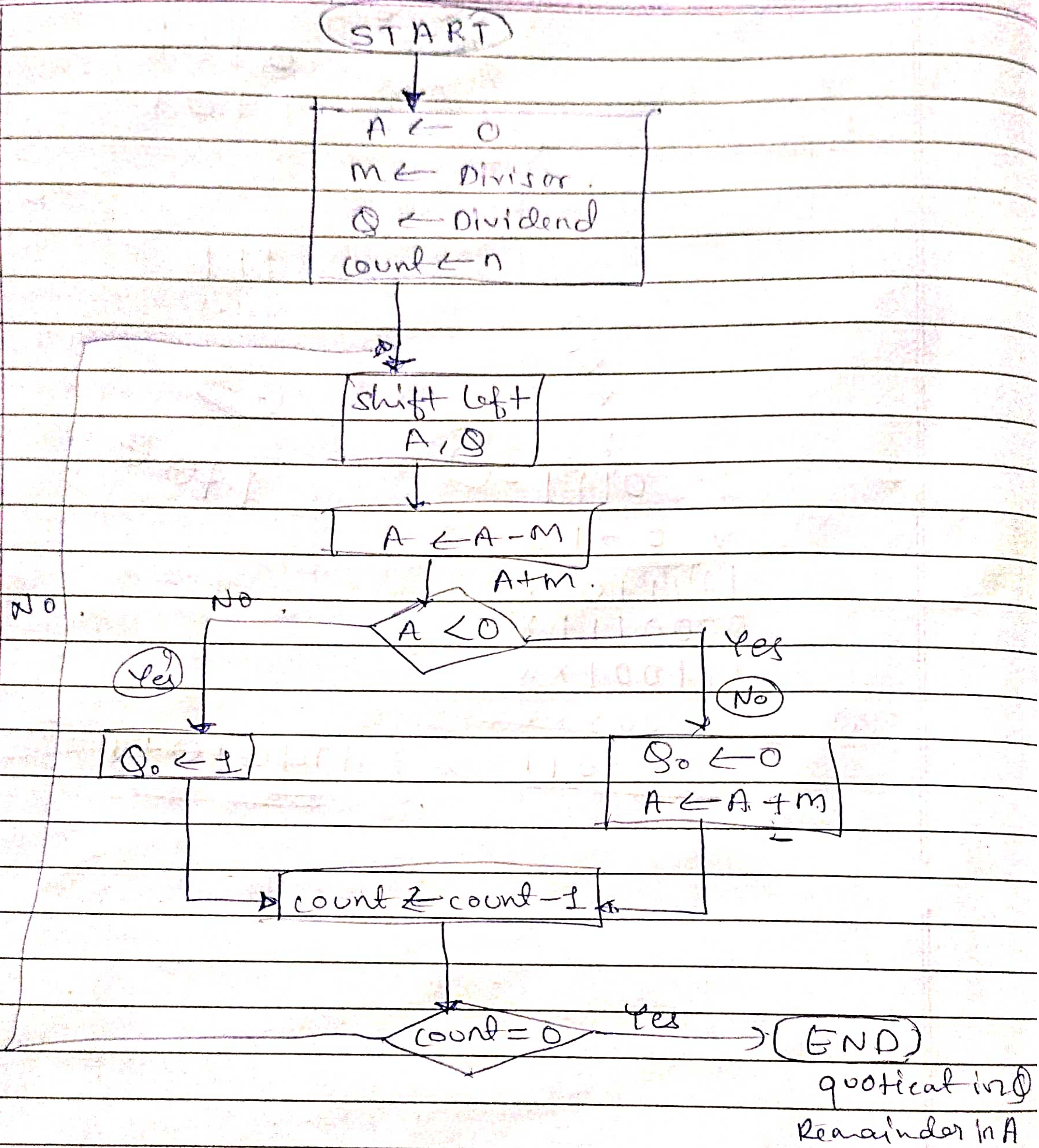
$16 + 4 + 1 = 21$

-21

Q	Q_{-1}	recoding
1 0		-1
0 1		+1
0 0		0
1 1		0

1101 0

0 -1 +1 -1



unsigned
binary division.

A	Q	
0000	0111	M = 3. M = 0011

1) 0000 1110

−M 1101

1101 1110 → Q_0.

+ 0011

0000 1110

2) 0001 1100

1101

1110

0011 → M

0001 1100

3) 0011 1000

1101

0000 1000 $Q_0 \to 1$.

1 ← ?

4) 0001 0001 $Q_0 \to 1$.

350 – stallings

1.

Restoring & Non-Restoring of Unsigned.

8/3

A   Q   M ← 0011

0000   1000   2's → 1101

1).

L.S.   0001   000_

sub.   1101

1110   0000

0011

2). 0001   0000

2)

L.S.   0010   000_

sub.   1101

1111   0000

0011

0010   0000

3

L.S.   0100   000_

sub M   1101

0001   0001

4

L.S.   0010   001_

1101

1111   0010

0010   0010

2   2 → Q

R.

Non-restoring for unsigned.

stage 1 :- ① If sign of A is 0, shift A & Q left one bit position & sub M from A, otherwise shift A & Q left & add M to A.

② Now, if the sign of A is 0, set q_0 to 1; otherwise, set q_0 to 0.

Restore!

stage 2 :- If the sign of A is 1, add M to A.

A	Q	
0000	1000	M → 0011
		2's → 1101

①

LS 0001 000_

sub 1101

1110 0000

add next.

② 1100 000_

add. 0011

1111 0000

add next.

③ 1110 000_

add 3. 0011

0001 0001

sub next.

④ LS 0010 001_

sub last. 1101

1111 0010

0011

0010

2

2

SIGN DIVISION

sign of D = sign of V

Dividend	divisor

D +ve	V +ve	Q +ve	R +ve
D +ve	V -ve	Q -ve	R +ve
D -ve	V +ve	Q -ve	R -ve
D -ve	V -ve	Q +ve	R -ve

sign !!

-145 / 13

D = -145	Dv → 13 → 01101
10011

A	Q

111101	111000	0 0 0 1 0 0 1 0 1
111101	101111	1 0 1 1 1 1 0 0 1

+ 110011

111101
001101
001010	101111

128
127
008

Dv → 13

01101
145
128
007

LS.	111011	011110
+ 001101
001000	011111

LS.	010000	111110
+ 001101
011101	111110

111011 011110

145
128
017

101101111

LS.	111011	11111_
+ 001101
001000	111111

LS 010001 11111_
001101
011110 111111

LS 111101 11111_
001101
001010 111111

LS 010101 11111_
001101
100010 111110

011110

11
~~1+45~~
12

~~111011~~ ~~011111~~

LS 110110 11110_
001101
000011 111110
000011
110110

LS 101101 111000
001101
111010 111001

LS. 110101 11001_
001101
000010 110010
110101 110010

LS

-11

15/9

11	101011	10010
	001101	
	111000	100101

15.	110001	00101
	001101	
	111110	001011

(-2).

000010

2

11

110101

-11

8+3

11

S	E'	m

0 → +ve
1 → -ve.

8 bit signed exponent in excess -127 representatⁿ

23 bit mantissa fraction.

$$\text{value} = \pm 1 \cdot M \times 2^{E'-127}.$$

@ single precision,

40 - 127 = ...

8 + 32 = 40

| 0 | 00101000 | 001010 . . . 0 |

value = $1.001010...0 \times 2^{-87}$

→ increased exponential & mantissa.

| S | E' | M |

1 → -ve
0 → +ve

11 bit access
1023
exponent

52 bit
mantissa fraction

~~acc~~

(c) double precision & IEEE formats.

$B = 1.M = 1.b_{-1}b_{-2} \cdots b_{-23}$

$V(B) = 1 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-23} \times 2^{-23}$

$E' = E + 127$ } excess -127 format.

$-126 \le E \le 127$

$-1022 \le \bar{E} \le 1023$

| 0 | 10001000 | 0010110 . . 0 |

↳ There is no 1 to the left of point.

value = $+0.0010110 \cdots \times 2^{9}$

(a) unnormalized value.

8 + 128 → 136 - 127 → 9.

| 0 | 10000101 | 0110 . . . |

value = $+1.0110 \cdots \times 2^{6}$

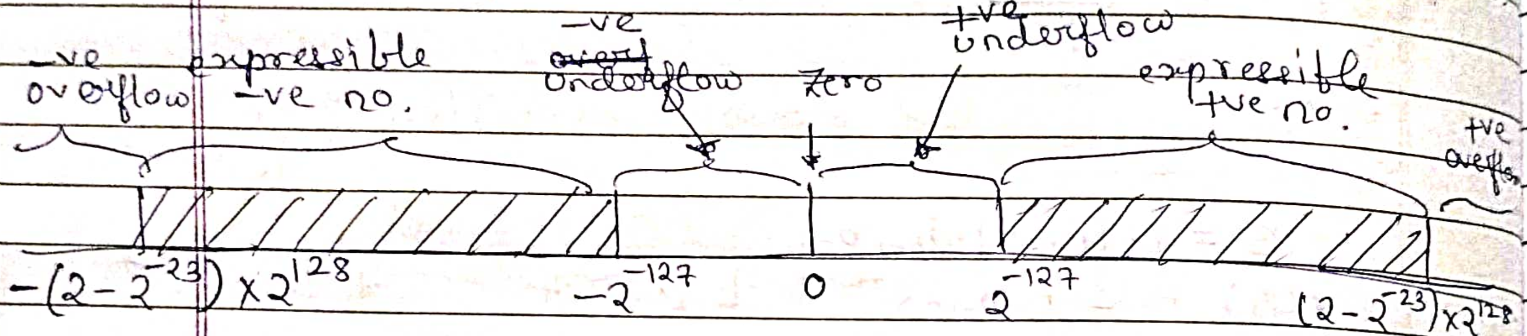
(b) normalized

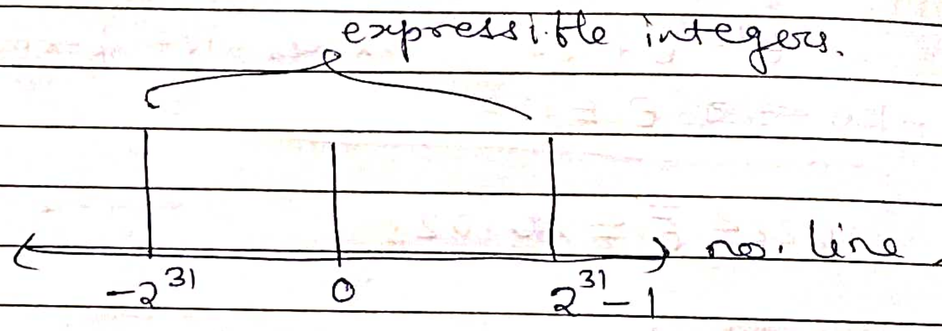
1 + 4 + 128 → 133 - 127 = 6.

if $exp < -126$ then underflow
or if $exp > 127$ then overflow.

$s = \pm, \pm 0$ $E' = 0$, $m = 0$ → 0 no.
$s = \pm, \pm\infty$ $E' = 255$, $m = 0$ → ∞ no

$E' = 255$, $m \neq 0$, value called Not a No. (NaN)





storage	32	64	128
Expo.	8	11	15
bias	127	1023	16383
max expo.	127	1023	16383
min expo.	-126	-1022	-16382
normal no. range	10_{-38}, 10_{+38}	10_{-308}, 10_{+308}	10_{-4932}, 10_{+4932}
Mantissa length	23	52	112
no. of exponent.	254	2046	32766
no. of fractions	2_{23}	2_{52}	2_{112}

Add / subract -

1. Choose the no. with the smaller exponent & shift its mantissa right a no. of steps equal to the difference in exponents
2. Set the exponents of the result equal to the larger exponent.
3. Perform addition / subraction on the mantissa & determine the sign of the result
4. Normalize the resulting value, if necessary

Multiply -

1. Add exponents & subract 127 to maintain the excess - 127 representatn
2. Multiply the mantissas & determine the sign of the result.
3. Normalize the resulting value, if necessary

Divide -

1. Subract the exponents & add 127 to maintain

Q. express the following no.'s in IEEE 32 bit floating point format.

2+127 → 129

0+127

01~~0~~0.0 × 2^2

1.01~~0~~ 0 × 2^{2}

1.010 × 2^{2}

2+127

-5, -6, -1.5, 384, 1/16, -1/32

E'

-5. | 1 | 10000001 | 010 ---- 0 |

s e M

= ~~+~~ - 1.(010...0) × 2^{2}

0101.0

1.010 × 2^{2}

2+127 = 129

0 0 0 0 0 0 0 1

-6. | 1 | 10000001 | ~~01~~10 ---- 0 |

s e

= -1.(0110..) × 2^2

-1.5 → ~~0101~~

0001.0101 × 2^{0} → 0+127

| 1 | 01111111 | 010.... 0 |

s e

2+127 → 129

0110.0

1.10 × 2^{2}

101

e' - 127 =

1.101

5

- 1 0 0 0 0 1 1 1

0 1 1 1 1 0 1 1

4+2+1

123

64

59

59

32

27

27

16

11

0011

3

1+2+8+16+32+64

-5 | 1 | 10000001 | 010 . . . . 0 |

$= -1.(010\ldots0) \times 2^2$

-6 | 1 | 10000001 | 10 . . . . 0 |

$\rightarrow 0110.0 \rightarrow \underbrace{1.10}_{\text{mantissa}} \times 2^{2} \rightarrow \text{expo} = 2+127 = 129.$

$= -1.(10\ldots0) \times 2^2$

-1.5 $\rightarrow 0001.0101 \times 2^{0} \rightarrow 0+127 = 127.$

| 1 | 01111111 | 01010 . . . . . . 0 |

s e

$= -1.(01010\ldots0) \times 2^0$

384. $\rightarrow 110000000.0.$

$\rightarrow 1.10 \times 2^{8} \rightarrow 127+8 = 135$

| 0 | 10000111 | 10 . . . . 0 |

$= 1.(100\ldots0) \times 2^8.$

$1/16 = 0.0625 \rightarrow$ ~~0.062~~ $0.0625 \times 2 = 0.125$ (0)

$0.125 \times 2 = 0.25$ (0)

0000.0001 $0.25 \times 2 = 0.5$ (0)

0.0001 $0.5 \times 2 = 1$ (1).

$\rightarrow 1.0 \times 2^{-4} \rightarrow -4+127$

$\rightarrow 123.$

| 0 | 01111011 | 0 . . . . 0 |

$= +1.(0\ldots0) \times 2^{-4}.$

1 1 0 0 0 0 0 0 0

Page No

Date

$\frac{1}{32} \times 4$

256

128

$-1/32 = -0.03125$

$0.03125 \times 2 = 0.0625$ 0

$0.0625 \times 2 = 0.125$ 0

$0.125 \times 2 = 0.25$ 0

-00.00001

$0.25 \times 2 = 0.5$ 0

$0.5 \times 2 = 1$ 1

$\text{value} = -1.0 \times 2^{-5} \rightarrow 127-5$

$\hookrightarrow 122$

| 1 | 01111010 | 0 . . . . . 0 |