PROCESSOR ORGANISATION AND ARCHITECTURE (POA)

Unsigned Binary subtraction

eg① 32 - 85

32 = 0100000 —1's C→ 1011111

85 = 1010101

subtracting using 1's complement.

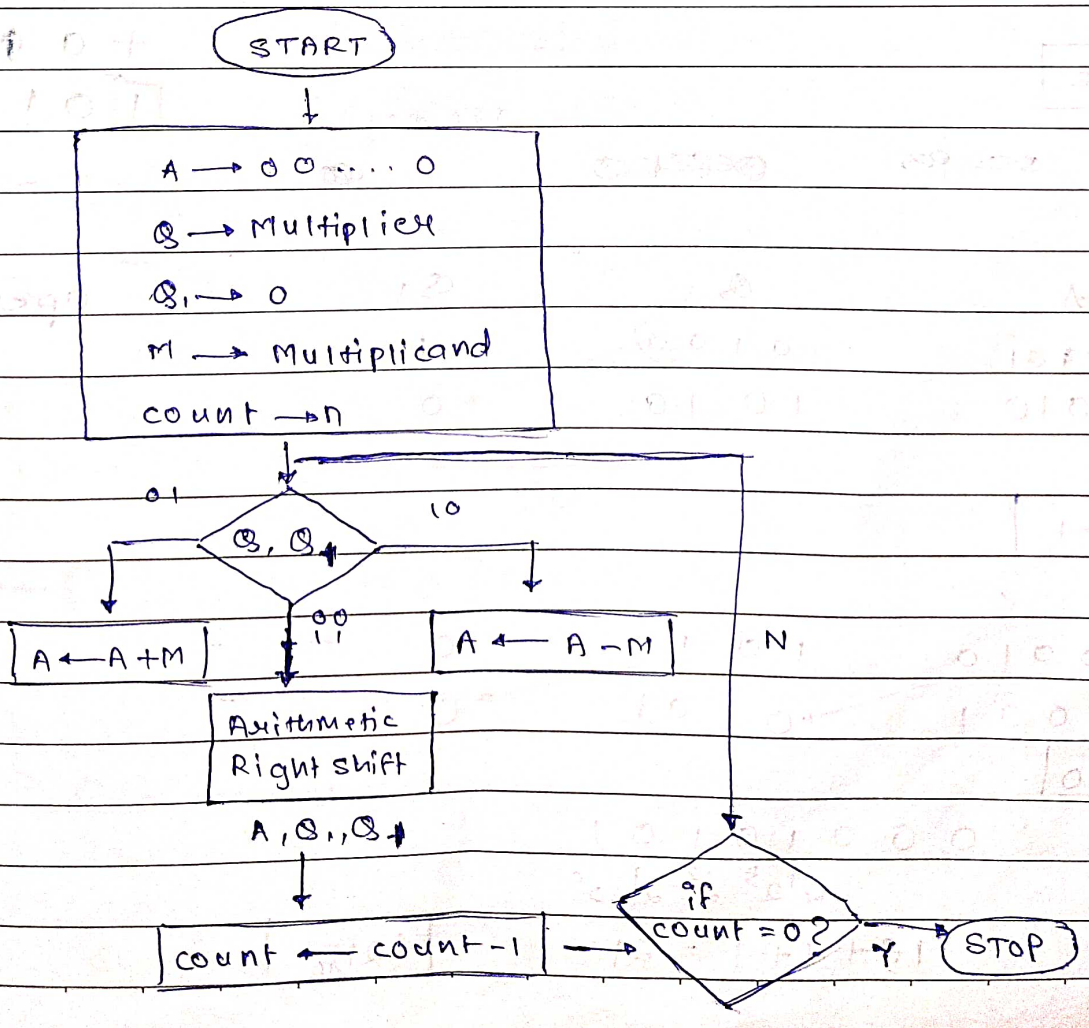
```
  1 1 1 1 1 1
  1 0 1 0 1 0 1
+ 1 0 1 1 1 1 1
-------------------
1 0 1 1 0 1 0 0
↳ Final carry
```

```
  0 1 1 0 1 0 0
+               1
-------------------
  0 1 1 0 1 0 1
```

1+4+16+32

= 53

BOOTH'S ALGORITHM FOR SIGN MULTIPLICATION:



1st bit must be written in fnout as it is

1 1 0 1 → discard } Arithmetic Right shift

1 1 1 0

Page No.:
Date:

Q.1 7×3

7 → Multiplicand – M

3 → Multiplier – Q

3 0011

7 0111 $\xrightarrow{1's\ C}$ 1000 $\xrightarrow{2's\ C}$ 1001 (–M)

	A	Q	Q_1	operation
Step 1	0000	0011	1	0000
				+ 1001
count = 4				1001

$[A \rightarrow A + (-M)]$

	A	Q	Q_1	operation
Step 2 (Right shift)	1001	0011	1	
	1100	1001		
count = 3				

	A	Q	Q_1	operation
Step 3 (Right shift)	1100	1001	1	
	1110	0100	1	1110
count = 2				+ 0001
				1 0101

$[A \rightarrow (A + M)]$

discard carry if any

operation

	A	Q	Q_1	operation
Step 4 (Right shift)	0101	0100	0	
	0010	1010	0	
count = 1				

00

	A	Q	Q_1	operation
Step 5 (Right shift)	0010	1010	0	
	0001	0101	0	
count = 0				

0 0 0 1 0 1 0 1

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

$16 + 4 + 1 = 21$

$(21)_{10}$

Q.2 $7 \times (-3)$

M = 7, Q = 3

7 → 0111 → 1's C → 1000 → 2's C → 1001 (−M)

3 → 0011 → 1's C → 1100 → 2's C → 1101 (−3)

	A	Q	Q1 = 0	operation
count 4	0000	1101	0	0000
				+ 1001
				1001

10

[A → A + (−M)]

ARS	1001	1101	0	1100
				+ 0111
count 3				1|0011

01

[A → A + M]

↳ discard.

ARS	0011	1110	1	0001
				+ 1001
count = 2				1010

10

[A → A + (−M)]

ARS	1010	1111	0	
count = 1	1101	0111	1	
ARS	1101	0111	1	
count = 0	1110	1011	1	

If MSB of A is 1 it indicates the answer is −ve

1110 1011

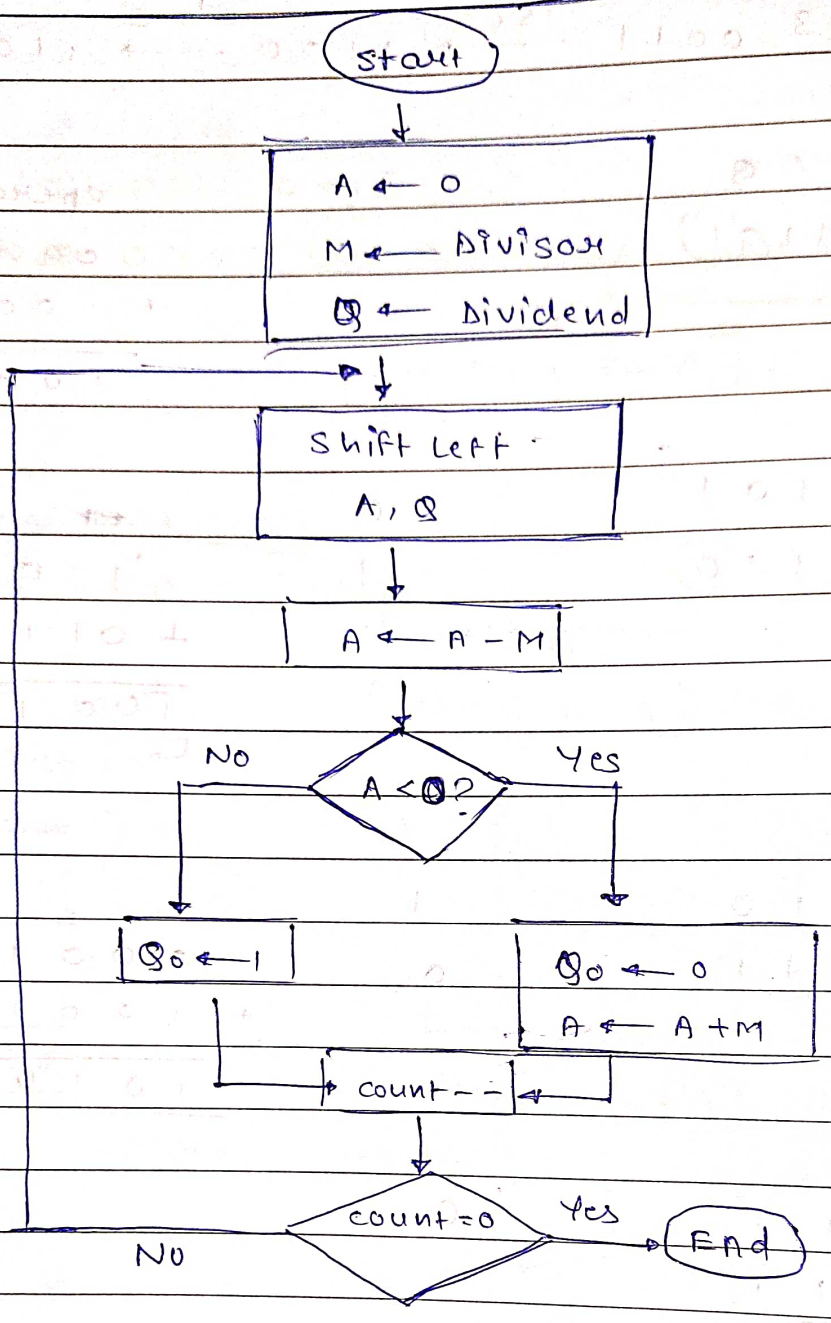
2's C

0001 0101 = 21

1's C

$-(21)_{10}$

DIVISION USING RESTORING METHOD:



Q.1 13/5

13 → 01101 → 1's C → 00101 → 11010 → 2's C → 11011 (-M)

5 → M

13 → Q

A	Q	M	Action
00000	01101	00101	
00000	1101_	00101	Left shift
+11011	Q0		A = A-M
11011			
-ve 1 11011	11010	00101	A<0 – Yes
discard + 00101			Q0 ← 0
100000			A = A+M

count = 0

	A	Q	M	Action
count=4	0 0 0 0 0	1 1 0 1 0	0 0 1 0 1	
	0 0 0 0 1	1 0 1 0 _	0 0 1 0 1	Left Shift
	+ 1 1 0 1 1	Q_0		A = A − M
-ve	(1) 1 1 0 0			
	1 1 1 0 0	1 0 1 0 0	0 0 1 0 1	A < 0 — Yes
	+ 0 0 1 0 1			$Q_0 \leftarrow 0$
discard	(1) 0 0 0 0 1			A = A + M

	A	Q	M	Action
count=3	0 0 0 0 1	1 0 1 0 0	0 0 1 0 1	
	0 0 0 1 1	0 1 0 0 _	0 0 1 0 1	Left Shift
	+ 1 1 0 1 1	Q_0		A = A − M
-ve	(1) 1 1 1 0			
	1 1 1 1 0	0 1 0 0 0	0 0 1 0 1	A < 0 — Yes
	+ 0 0 1 0 1			$Q_0 \leftarrow 0$
discard	(1) 0 0 0 1 1			A = A + M

	A	Q	M	Action.
count=2	0 0 0 1 1	0 1 0 0 0	0 0 1 0 1	
	0 0 0 1 0	1 0 0 0 _	0 0 1 0 1	Left Shift
	+ 1 1 0 1 1	Q_0		A = A − M
discard	(1) 0 0 0 0 1			
+ve				A ≮ 0 — No
	0 0 0 0 1	1 0 0 0 1	0 0 1 0 1	$Q_0 \leftarrow 1$
count=1	⟶ End cycle			

	A	Q	M	Action
	0 0 0 0 1	1 0 0 0 1	0 0 1 0 1	
	0 0 0 1 1	0 0 0 1 _	0 0 1 0 1	Left Shift
	+ 1 1 0 1 1	Q_0		A = A − M
-ve	(1) 1 1 1 0 0			

```
  1 1 1 1 1 0      0 0 0 1 0      0 0 1 0 1      A < 0 — Yes
+ 0 0 1 0 1                                     Q0 ← 0
                                                A = A + M
  1 0 0 0 1 1
```

discard (1)

count = 0 ⟶ End

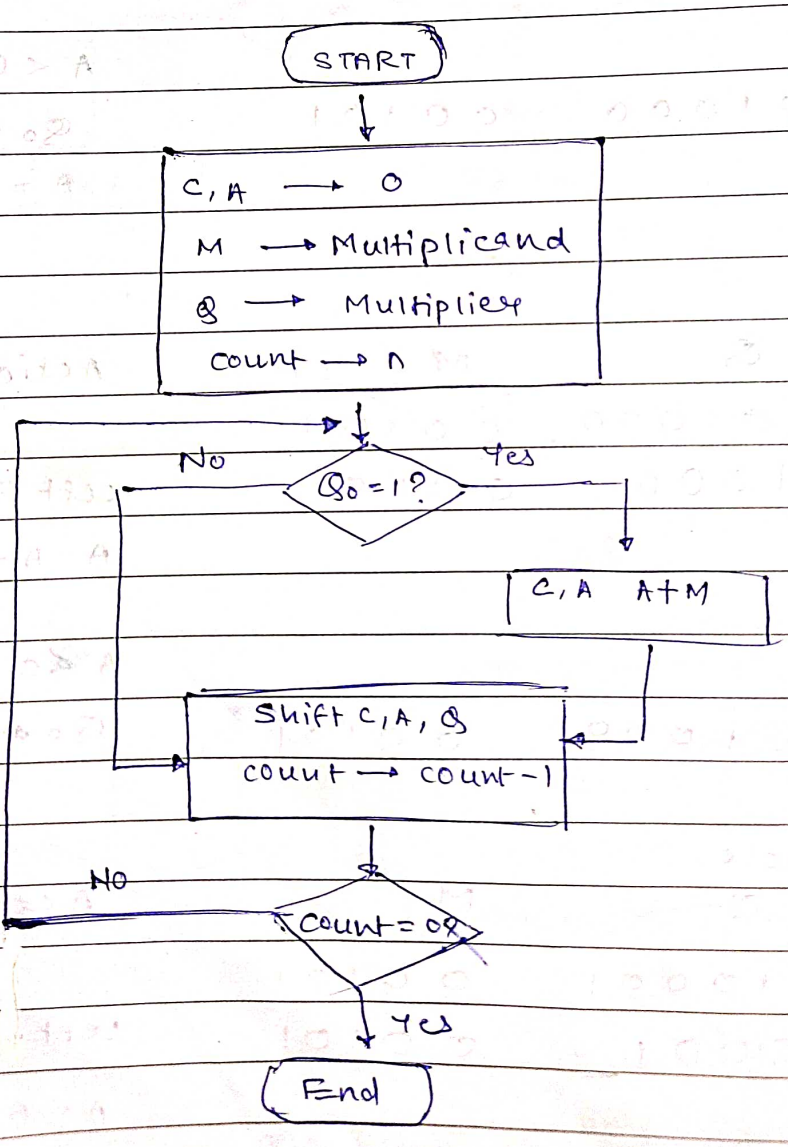
```
  0 0 0 1 1      0 0 0 1 0      0 0 1 0 1
      S              S
```

Remainder Quotient

Quotient = $(00010)_2 = (2)_{10}$

Remainder = $(00011)_2 = (3)_{10}$

UNSIGNED MULTIPLICATION :



Q.1 13×11 (Q → Multiplier, M → Multiplicand.)

$13 = 1101$ $\quad A = 0000$

$11 = 1011$ $\quad C = 0$

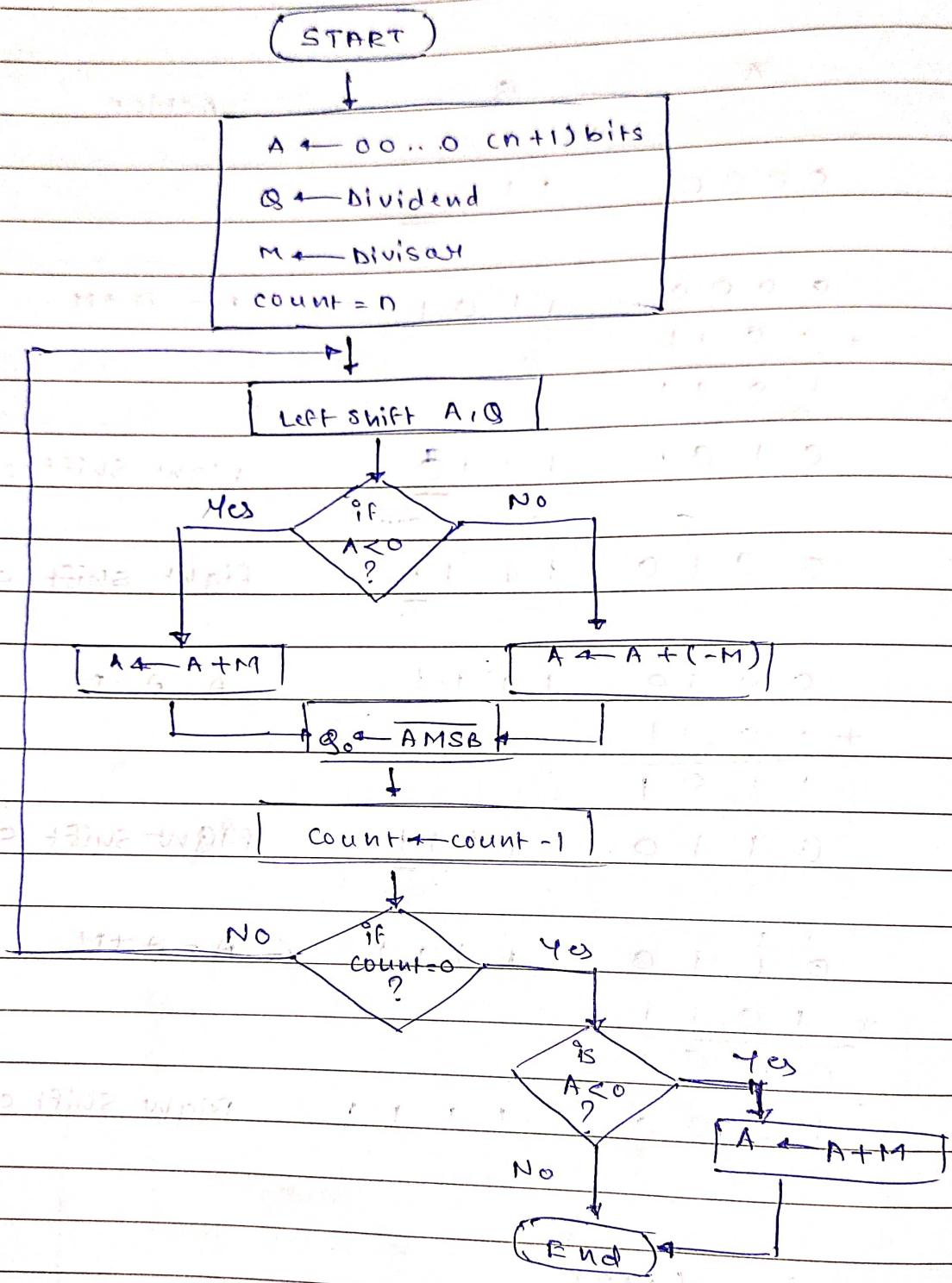
count	carry	A	Q	Action
count=4	0	0000	1101	
			$\overline{Q_0}=1$	
	0	0000	1101	$A = A+M$
		$+\ 1011$		
		1011		
	0	0101	1110	Right shift C, A, Q
			$\overline{Q_0}\neq 1$	
count=3	0	0010	1111	Right shift C, A, Q
			$\overline{Q_0}=1$	
count=2	0	0010	1111	$A = A+M$
		$+\ 1011$		
		1101		
	0	0110	1111	Right shift C, A, Q
			$\overline{Q_0}=1$	
count=1	1	0110	1111	$A = A+M$
		$+\ 1011$		
		0001		
	0	1000	1111	Right shift C, A, Q
count=0				

$(10001111)_2 = (143)_{10}$

Date:

YOUVA

DIVISION USING NON-RESTORING METHOD:



Q. 11 ÷ 3 → Q = Dividend, 11 = 01011

M = Divisor, 3 = 00011 (M), (−3) = 11101 (−M)

	A	Q	Action
	00000	1011	
A +ve	00001	011_	Left Shift
	00001	0110	A = A − M
	+ 11101		
AMSB	11110	AMSB	
count = 3			
A −ve	11100	110_	Left Shift
	11100	1100	A = A + M
	+ 00011		
AMSB	11111	AMSB	
count = 2			
A −ve	11111	100_	Left Shift
	11111	1001	A = A + M
	+ 00011		
discard	10010	AMSB	
	AMSB		
count = 1			
A +ve	00101	001_	Left Shift
	00101	0011	A = A − M
	+ 11101		
discard	00010	AMSB	
	AMSB		
count = 0			
	00010	0011	
	A	Q	

↳ A is (+ve) when count = 0

No need for A = A + M

∴ Remainder = A = 00010 = $(2)_{10}$

Quotient = Q = 00011 = $(3)_{10}$

MEMORY ORGANISATION

RAM [RANDOM ACCESS MEMORY]

① S-RAM [STATIC RAM]: (133 Mega Hz – clock frequency)

1 bit fetched

a] Normally used for cache memory
b] Faster
c] Expensive
d] Transistors, Flipflops i.e more components
e] Low density device
f] Low Latency (delay)

② D-RAM [DYNAMIC RAM]:

1 bit fetched

a] Normally used for main memory
b] Slower
c] Less Expensive.
d] capacitors i.e less components
e] High density device.
f] High Latency (delay)

③ SD-RAM [SYNCHRONOUS DYNAMIC RAM]:

a] Synchronises with clock of CPU

1 bit fetched

- Input-Output Clock Frequency ⟶ 100 – 133 MHz (connection with other devices)
- Internal Clock Frequency (within the same device) ⟶ 100 – 133 MHz

[Because of delay in D-RAM we use SD-RAM]

④ DDR [DOUBLE DATA RATE]: (184 pins)

fetches double than normal RAM

⑤ DDR2:

Double than DDR (240 pins)

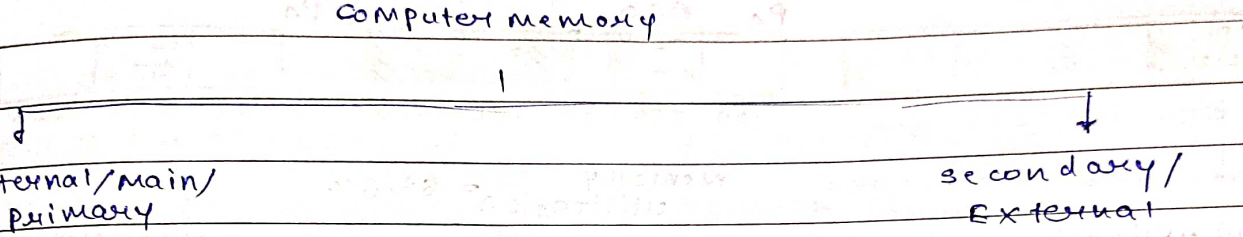
⑥ DDR3:

Double than DDR2 (240 pins)

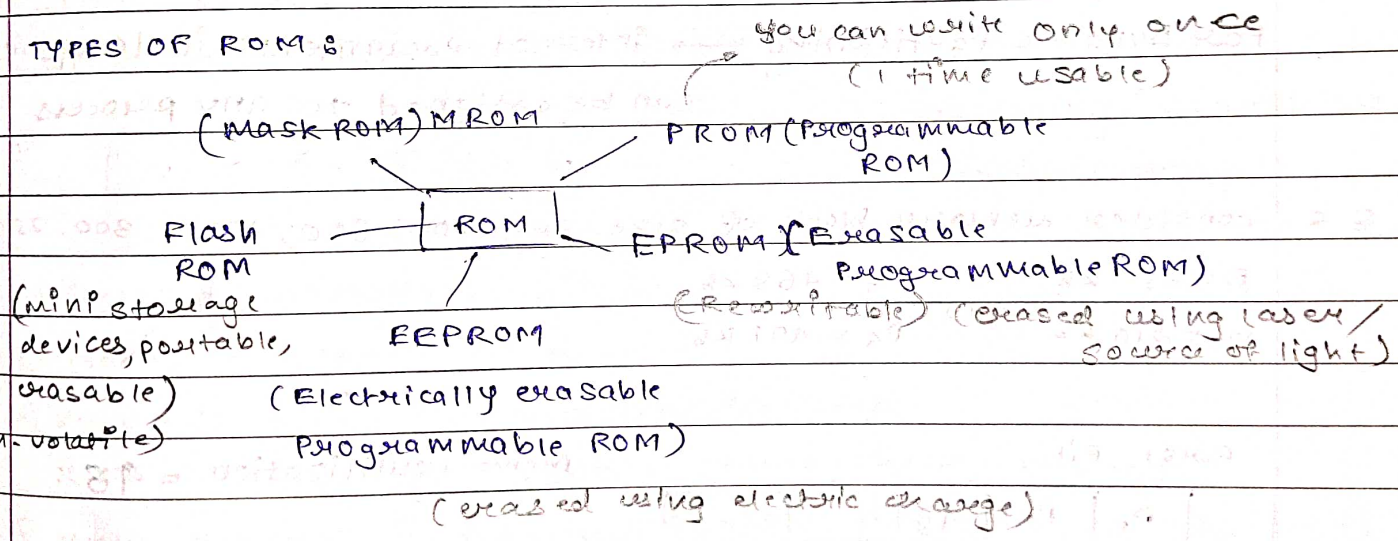
- RAM
  - SRAM
  - DRAM
    - SDRAM
      - DDR
      - DDR2
      - DDR3

ROM (READ ONLY MEMORY)

1. Non-volatile
2. mainly to start / boost up the computer.



TYPES OF ROM:



MEMORY ALLOCATION TECHNIQUES:

1. First-Fit
2. Best-Fit
3. Worst-Fit

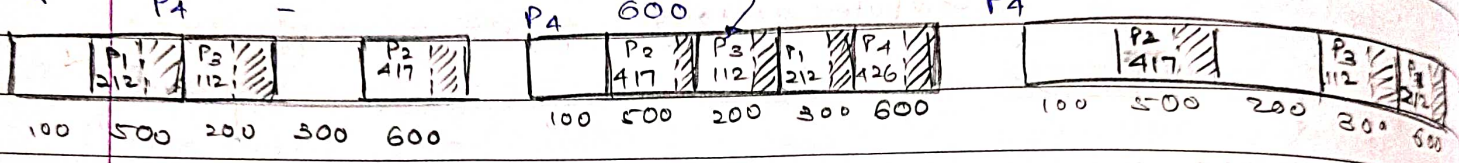
Page No.:
Date:

Q.1 consider memory block of size 100, 500, 200, 300 and 600 KB each

P_1 = 212 KB P_3 = 112 KB

P_2 = 417 KB P_4 = 426 KB

First-Fit		Best-Fit		Worst-Fit	
P_1	500	P_1	300	P_1	600
P_2	500	P_2	500	P_2	500
P_3	200	P_3	200	P_3	300
P_4	–	P_4	600	P_4	



//// → internal fragmentation.

Memory utilization = 68.6%

Memory utilization = 212+112+417 / 1700 = 43.5%

Memory utilization = 49.5%

Imp For Fixed partitioning → internal fragments cannot be assigned to any process

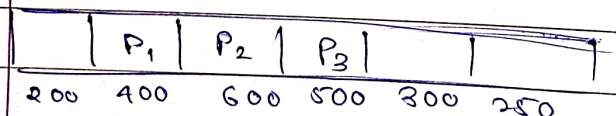
For Dynamic Partitioning → internal fragments if large enough can be assigned to any process

Q.2 consider memory block of size 200, 400, 600, 500, 300, 250

P_1 = 357 Kb P_3 = 468 Kb

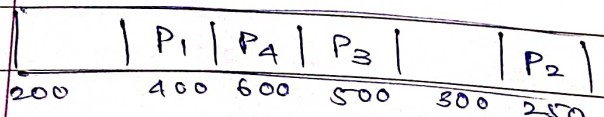
P_2 = 210 Kb P_4 = 491 Kb

First Fit



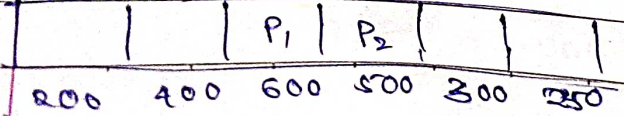
Memory utilization = 48%

Best Fit



Memory utilization = 67%

Worst Fit



Memory utilization 25.2%

LEVEL OF MEMORY :

Level 1 / Register memory

Level 2 / Cache memory

Level 3 / Primary memory

Level 4 / Secondary memory

INTERLEAVED MEMORY :

Successive words are stored in different memory bank

HIGH-ORDER INTERLEAVING :

(1) MSB of the memory address decides memory banks

(2) LSB bits are sent as addresses to each chip.

LOW-ORDER INTERLEAVING

(1) LSB of the memory address decides memory banks.

(2) Here ~~pro~~ problem of high order interleaving is solved.

(3) ~~faster~~ than higher since consecutive bits gets distributed.

0 1 2 3 → processing / searching becomes easy

4 5 6 7

ASSOCIATIVE MEMORY :

(1) Address - Address of w

(2) Keywords - How many digits to check

The bits that we want to search/check we keep it 1

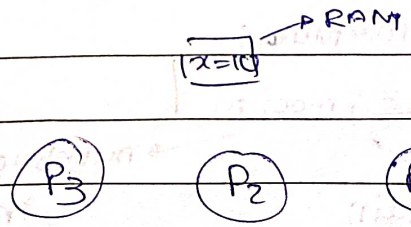
(3) match registers - The files that matches are assigned flag 1 and rest flag 0.

CACHE MAPPING :

(1) Direct

(2) Fully

(3) Associative.



to solve this cache mapping is used.

if P_1 fetches $x = 10$ then increments it $x = 11$ but before it tells RAM P_2 fetches it

① DIRECT MAPPING:

64 32 16 8 4 2 1

0 1 00	116	
0 1 01	117	
0 1 10	118	
0 1 11	119	

↓ line L1

29 → 5 6 → 116 → B29 → block 29

4 | 2

3	2	2
tag	lines	OFF

So block B_{29} is in line L1

L0	B_0	B_4	B_8	w_0	w_1	w_2	w_3	B_0
L1	B_1	B_5	B_9	w_4	w_5	w_6	w_7	B_1
L2	B_2	B_6	B_{10}	w_8	w_9	w_{10}	w_{11}	B_2
L3	B_3	B_7	B_{11}			⋮		⋮

Each line stores only 1 block at a time and specific block only

eg: L0 will have B_0, B_4, B_8... only

It won't store B_1 in it.

Each block has 4 words $= 2^{(2)} \rightarrow$ 2 bits

∴ Each line stores 1 block
= 4 words
= $2^{(2)} \rightarrow$ 2 bits.

if ~~these are~~ total 128 size $= 2^{(7)} \rightarrow$ 7 bits

3	2	2
tag	line	offset

4 bits	2 bits	2 bits
↓ tag	↓ line	↓ word off
		offset

Formula:

$i = j \bmod m$

- i: cache line number
- j: main memory block number
- m: number of lines in the cache

CACHE MAPPING:

Which data goes where in the main memory

1. Direct Mapping
2. Fully Associative Mapping
3. Set Associative Mapping

(2) FULLY ASSOCIATIVE MAPPING:

Direct mapping's drawback is solved → any block can go in any line of cache

| t bits | w bits |

t bits → tag
w bits → word - offset

which ever is the ~~avaible~~ available place there.

Problem — Mapping is flexible and fast
Searching is inconvinient as whole cache need to be searched,

(3) SET ASSOCIATIVE MAPPING:

Set is fixed but within the set it can be placed in any line
Mapping is flexible.
While searching only particular lines of a fixed set need to be searched

Formula:

$$m = v * k$$
$$i = j \bmod v$$

| t bits | s bits | w bits |

t bits → tag
s bits → set number
w bits → word - offset

8086 MICROPROCESSOR

OVERVIEW:

16 - bit microprocessor

20 - address lines

16 - data lines.

Two modes of operation:

1. min mode
2. max mode

It has an instruction queue, which is capable of storing 6 instruction bytes.

Available in 3 versions

8086 → 5 MHz

8086-2 → 8 MHz

8086-1 → 10 MHz

Using 2 stages of pipelining → Fetch stage, Execute stage

256 vectored interrupts

Consits of 29,000 transistors.

CS = IP	All RHS are corresponding offsets of LHS.
SS - SP/BP	
DS - SI	
ES - DI	