

Ans 1.) Machine : 16-bit words, Instruction format : 4-bit
opcode + 12-bit direct address.

Opcode given :

- 0001 = Load AC from memory
- 0010 = Store AC to memory
- 0110 = Subtract from AC

Program (3 instructions). PC is set to 300, so instruction at 301, 302, 303:

1) Load AC, 940

→ opcode = 0001
→ address 940 in 12-bit binary = 001110101100
→ instruction word: 0001001110101100

2) Subtract M[941] from AC

→ opcode = 0110
→ address 941 = 001110101101
→ instruction: 0110001110101101

3) Store AC to 941

→ opcode = 0010
→ address 941 = 001110101101
→ instruction = 0010001110101101

Assume Memory : $M[940] = 10, M[941] = 5$. AC initial unknown.

Fetch :-

Inst @ 301: Load AC, 940

• Fetch -
• MAR \leftarrow PC(301), memory read $M[301] \rightarrow MBR \leftarrow$

memory [301], IR \leftarrow MBR, PC \leftarrow 302.

• Decode -

→ opcode = 940

$\rightarrow \text{MAR} \leftarrow 940$, read memory: $\text{MBR} \leftarrow M[940]$
 $\rightarrow \text{AC} \leftarrow \text{MBR} (\text{AC} = 10)$
 $\rightarrow \text{Result: AC} = 10$

Inst @ 302:

- Fetch: $\rightarrow \text{MAR} \leftarrow \text{PC}(302)$, $\text{MBR} \leftarrow \text{memory}[302]$, $\text{IR} \leftarrow \text{MBR}$, $\text{PC} \leftarrow 302$.
- Decode:
 - $\rightarrow \text{MAR} \leftarrow 941$, read $M[941] \rightarrow \text{MBR} = 5$
 - $\rightarrow \text{AC} \leftarrow \text{AC} - \text{MBR} = 10 - 5 = 5$
 - $\rightarrow \text{Result: AC} = 5$

Inst @ 303:

- Fetch: $\rightarrow \text{MAR} \leftarrow \text{PC}(303)$, read instruction to $\text{MBR} \rightarrow \text{IR} \leftarrow \text{MBR}$, $\text{PC} \leftarrow 304$.
- Decode:
 - $\rightarrow \text{MAR} \leftarrow 941$, $\text{MBR} \leftarrow \text{AC}(5)$, write memory: $M[941] \leftarrow \text{MBR}(5)$.
 - $\rightarrow \text{Result: } M[941] \text{ becomes } 5$

Final State: $\text{AC} : \sim 5$; $M[940] = 10$; $M[941] = 5$

Ans 2) Steps	Operations	Explanation
1	$\text{MAR} \leftarrow \text{PC}$, $\text{MBR} \leftarrow M[\text{MAR}]$, $\text{IR} \leftarrow \text{MBR}$, $\text{PC} \leftarrow \text{PC} + 1$	Fetch Instruction
2	Decide opcode & address	Identify operation
3	For Load	$\text{MAR} \leftarrow 940$, $\text{MBR} \leftarrow M[940]$, $\text{AC} \leftarrow \text{MBR}$
4	For Subtract	$\text{MAR} \leftarrow 941$, $\text{MBR} \leftarrow M[941]$, $\text{AC} \leftarrow \text{AC} - \text{MBR}$
5	For Store	$\text{MAR} \leftarrow 941$, $\text{MBR} \leftarrow \text{AC}$, $M[\text{MAR}] \leftarrow \text{MBR}$

Name: Pawhali

Regd. Number: 2341013223

Ans 3) Given,

Cache access time = 50 ns
 Main memory access = 550 ns
 Reads = 80%
 Writes = 20%
 Read hit = 0.9

Now,
 $AAT = 0.8 [0.9 \times 50 + 0.1 \times 550] + 0.2 \times 550$
 $= 0.8 \times 100 + 110$
 $= 190 \text{ ns}$
 i.e. Average Access Time = 190 ns.

Ans 3) Job	CPU t.	Time (ms)	Memory (MB)
Job 1	10	10	150
Job 2	20	20	100
Job 3	15	15	125

Total mean = 375 MB.

(a) Uniprogramming: ~ Runs one by one
 $\rightarrow \text{CPU time used} = 10 \times 0.7 + 20 \times 0.1 + 15 \times 0.1 = 10.5 \text{ min}$

Total time = 45 min
 $\rightarrow \text{CPU util} = 10.5/45 = 23.3\%$

$\rightarrow \text{Mem util} = (150 \times 10 + 100 \times 20 + 125 \times 15) / 45 = 29.9\%$

$\rightarrow \text{Throughput} = 3/45 = 0.067 \text{ job/min}$

(b) Multiprogramming: ~ All jobs in memory \rightarrow total mean mem = $375/400 = 93.75\%$

$\rightarrow \text{CPU util} = 1 - (0.3 \times 0.9 \times 0.9) = 75.7\%$

$\rightarrow \text{Throughput} = 3/20 = 0.15 \text{ job/min}$

Name: Pawhali

Regd. Number: 2341013223

Job	Total	CPU	I/O (each)
Job1	23	3	10
Job2	29	5	12
Job3	14	4	5

Uniprogramming :-

$$\text{Total} = 66 \text{ ms}, \text{CPU} = 12 \text{ ms}$$

$$\rightarrow \text{CPU Util} = 18.2\%$$

Multiprogramming :-

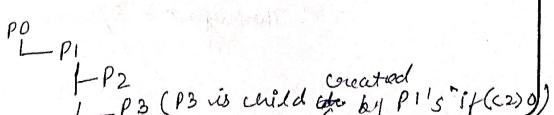
overlapping CPU/I/O reduces idle time
 → CPU busy 12 ms / total 30 ms
 $= 40\%$

1. CPU Utilisation improves from 18.2% → 40%

Ans 6) a) Trace:

- Start at P0
- C1 = fork(); → creates P1 (child)
- In P0: C1 > 0 → skip C2 = fork();
 $(\text{so } C2 \text{ stays } 0)$
- In P1: C1 == 0 → executes C2 = fork();
 $\text{which creates } P2.$
- In P1: C2 > 0 (true) → executes fork() producing P3.
- In P2: C2 == 0 → does not execute the if (C2 > 0) fork.

Graph:-



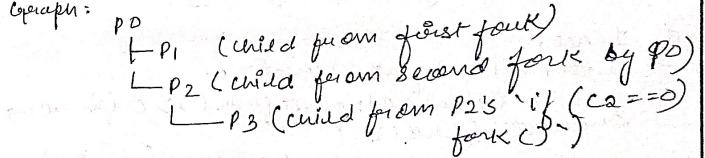
Name: Pushali

Regd. Number: 2341013223

fork()

- b) → Start: P0 (C1 = 1, C2 = 1).
- C1 = fork(); → creates P1.
- In parent P0: C1 != 0 true → do C2 = fork(); creates P2.
- In P0: after that C2 > 0 so if (C2 == 0) false → no else fork.
- In P2: C2 == 0 true → executes fork() creating P3.
- In child P1: C1 == 0 → skip second fork (C2 stays 1 → if (C2 == 0) false).

Graph:-



c) Trace:-

- Start P0
- first fork() creates A:
- In parent P0: first fork() returns 0 → (true || ...) short-circuits; parent then executes fork() inside the if (the third fork) producing C.
- In child A: first fork() returned 0 → evaluate secnd fork() (the right operand of ||)
 \rightarrow this second fork() (call it B) creates B (child of A). Results:

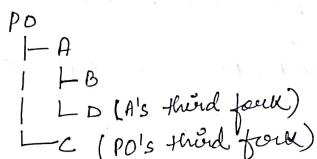
Name: Pushali

Regd. Number: 2341013223

→ In A: second fork() returned $> 0 \rightarrow (0 \parallel 0) = \text{true} \rightarrow A \text{ executes third fork() producing } D.$

→ In B: second fork() returned 0 $\rightarrow (0 \parallel 0) = \text{false} \rightarrow B \text{ doesn't execute the third fork().}$

Graph:



d) If all:

→ Start: PO. first fork() creates A.
 \rightarrow In PO (parent): first fork() > 0 (true) \rightarrow evaluate !fork();
 \rightarrow PO executes second fork() (call it C).
 In parent PO, second fork() returns $> 0 \rightarrow !> 0 \rightarrow \text{false} \rightarrow$ first if condition false \rightarrow PO doesn't enter inner if.

\rightarrow In child C: inherits the state where the first fork() return was > 0 (true), second the first fork() return was > 0 (true), overall fork() return $> 0 \rightarrow !0 \rightarrow \text{true} \rightarrow$ overall true & true So, C enters the if inner if.

\rightarrow In A: first fork() returned 0 (false) \rightarrow short-circuit & So A does not execute second fork and does not enter the inner if.

Name: Poushali

Regd. Number: 2341013223

→ Now only process C continues into the inner if:
 \rightarrow evaluates if (fork() || fork()) fork();

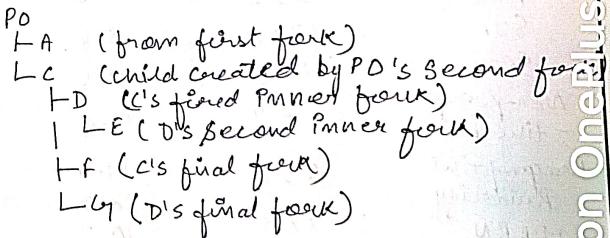
\rightarrow first fork here creates D.
 \rightarrow In C: first inner-fork returned > 0 (true || ...) short-circuits \rightarrow condition true \rightarrow C runs final fork() producing F.

\rightarrow In D: first inner-fork returned 0 \rightarrow D evaluates the second fork() producing E.

\rightarrow In D: second inner-fork returned $> 0 \rightarrow (0 \parallel 0) = \text{false} \rightarrow D \text{ executes final fork() producing } G.$

\rightarrow In E: second inner-fork executes 0 $\rightarrow (0 \parallel 0) = \text{false} \rightarrow E \text{ does not execute final fork().}$

Graph:



Name: Poushali

Regd. Number: 2341013223

Process	Arrival	Burst
P ₁	0	3
P ₂	1	1
P ₃	3	3
P ₄	4	X

Average waiting time = $1 \rightarrow$ total waiting = 4.
After simulation under SJF:

$$\cdot w_1 = 1, w_2 = 0, w_3 = 1+x, w_4 = 0 \\ \rightarrow 1 + 0 + (1+x) + 0 = 4 \Rightarrow x = 2$$

Ans: $x = 2$ ms

Process	Arrival	Burst	Priority
P ₁	0	11	1
P ₂	0	8	0
P ₃	12	2	3
P ₄	2	6	2
P ₅	3	16	4

Algorithm	Avg Turnaround	Avg Waiting	Avg Response
FCFS	23.2	14.6	14.6
SJF	17.0	8.4	8.4
SRTF	17.0	8.4	8.4
Non-preemptive - time priority	22.4	15.8	15.8
Preemptive Priority	23.0	14.4	14.4
HRRN	18.8	10.2	10.2
RR ($q=2$)	23.8	15.2	15.2

Name: Poushali

Regd. Number: 2341013223

Minimum Avg Waiting Time : 8.4 (SJF/SRTF)			
Algorithm	Process	Arrival	Burst
FCFS	P ₁	0	4
SJF	P ₂	0	2
SRTF	P ₃	1	3
Non-preemptive priority	P ₄	2	2

Algorithm	Avg Turnaround	Avg Waiting
FCFS	6.75	4.00
SJF	5.25	2.50
SRTF	5.25	2.50
Non-preemptive priority	6.00	3.25
Preemptive Priority	6.00	3.25
HRRN	6.50	3.75
RR ($q=2$)	7.00	4.25

Minimum Avg Waiting Time: 2.5 (SJF/SRTF)

Ans 10) Given

- Queues:
 - $\rightarrow Q_1 \rightarrow$ Round Robin (quantum=3)
 - $\rightarrow Q_2 \rightarrow$ Round Robin (quantum=5)
 - $\rightarrow Q_3 \rightarrow$ FCFS
- All processes arrive at time 0

Name: Poushali

Regd. Number: 2341013223

Process	Burst Time
P ₁	8
P ₂	22
P ₃	4
P ₄	12

Stepwise Execution

Q1 (RR, q=3):

- P₁: 0-3 → rem 5 → to Q₂
- P₂: 3-6 → rem 19 → to Q₂
- P₃: 6-9 → rem 1 → Q → to Q₂
- P₄: 9-12 → rem 9 → to Q₂

Q2 (RR, q=5):

- P₁: 12-17 → finishes
- P₂: 17-22 → rem 14 → to Q₃
- P₃: 22-23 → finishes
- P₄: 23-28 → rem 9 → to Q₃

Q3 (FCFS):

- P₂: 28-42 → finishes
- P₄: 42-46 → finishes

Result

Process	Completion	Turnaround	Waiting
P ₁	17	17	9
P ₂	42	42	20
P ₃	23	23	19
P ₄	46	46	34

Average Turnaround Time = 32.2

Average Waiting Time = 20.5