24-7-19

Delta

Parallel computing - Doing several computation tasks simultaneously.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{31}b_{13} \cdots & & \\ & & \\ & & \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

→ If we have single processor, it will have 'n' computations.

→ And we have 'n^2' terms, so it will be ($O(n^3)$) complexity.

→ If we are having 'n' processors, then the operations can be done simultaneously or parallely

→ To multiply 2 matrices, if we have 3 processors: P_1, P_2, P_3

→ For computing 1st row :-

Processor ID ←	K=1	K=2	K=3
	P_1	P_2	P_3
Initialize^n	$C_{11} = 0$	$C_{12} = 0$	$C_{13} = 0$
	+	+	+
1st iterat^n	$a_{11}b_{11}$	$a_{12}b_{12}$	$a_{11}b_{13}$
	+	+	+
2nd iterat^n	$a_{12}b_{21}$	$a_{12}b_{22}$	$a_{12}b_{23}$
	+	+	+
3rd iteration	$a_{13}b_{31}$	$a_{13}b_{32}$	$a_{13}b_{33}$

Delta

→ It is of O(n), ∴ faster
→ Similarly, we can compute other rows.
→ Total computation time = $O(n^2)$, ∴ faster than sequential.

Eg: Supercomputers: Sunway Taihu delight (1.45GB)

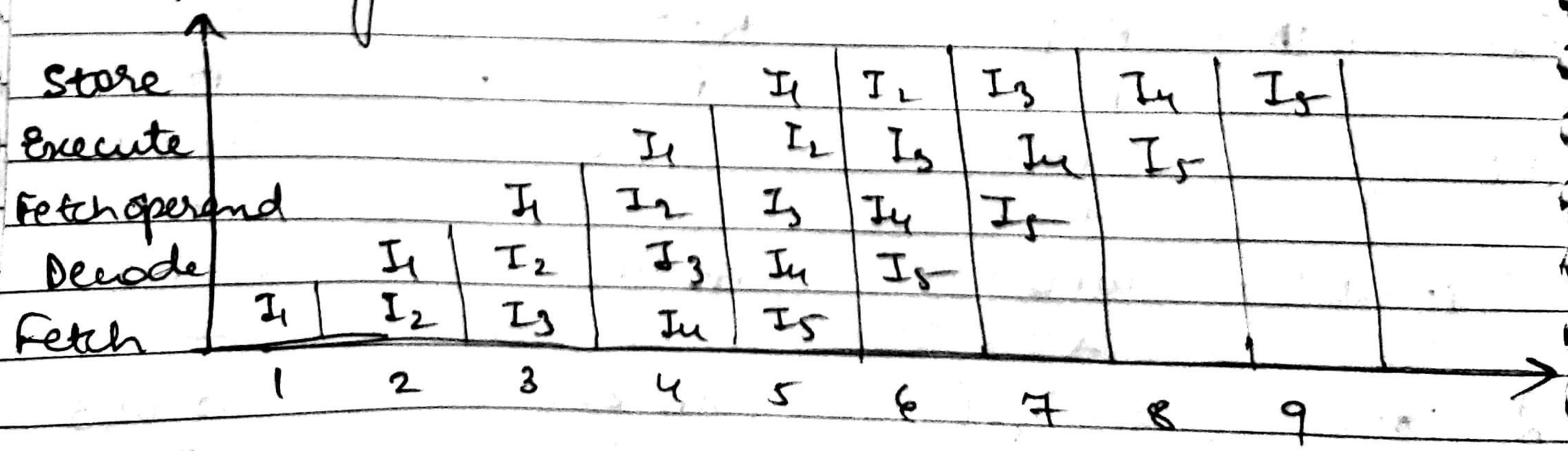
Q How the processor will know that it has to perform a certain operation?

Ans: We have parallel algorithms & the processors should have unique ID which is fixed.

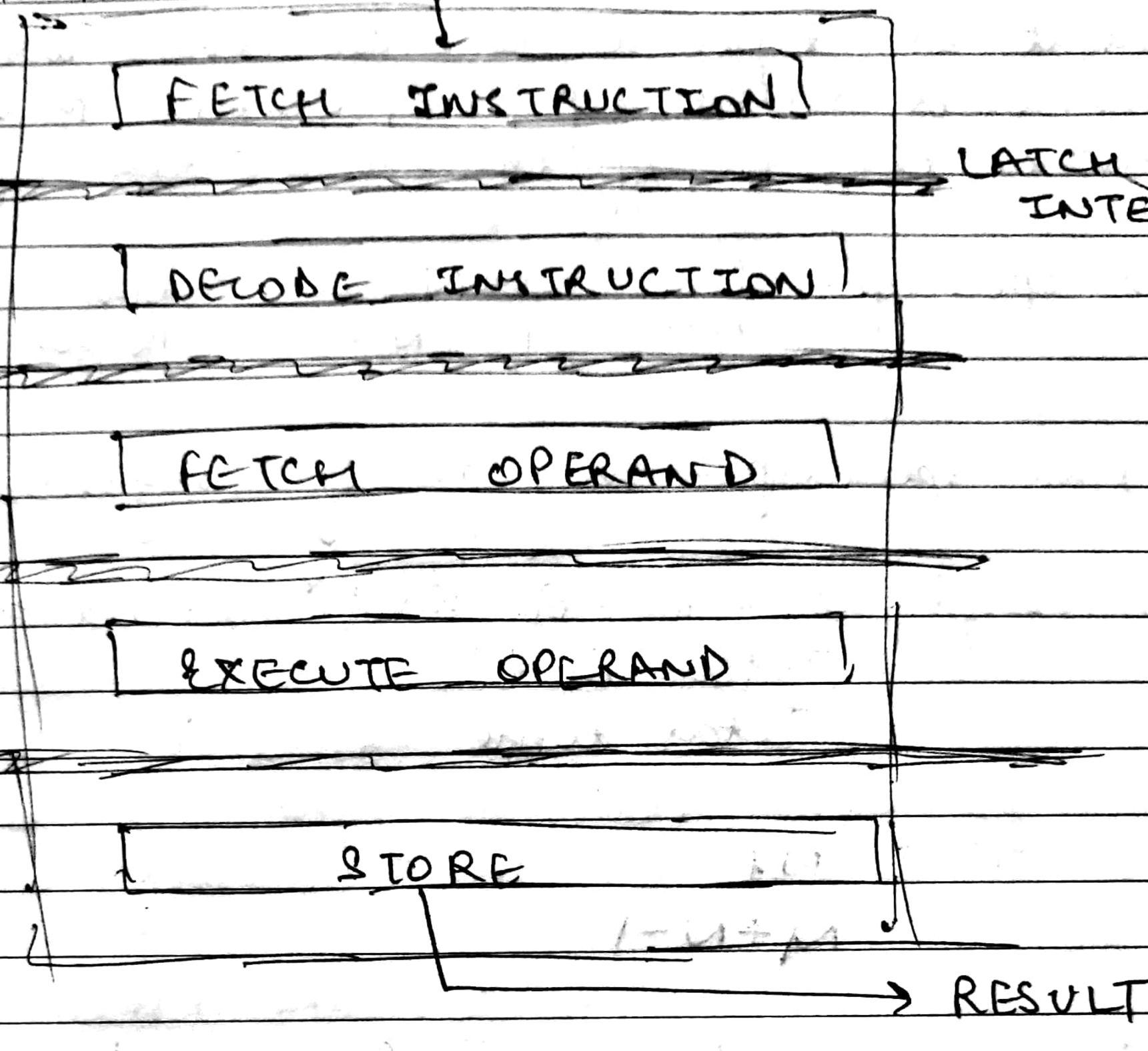
```
for(K=1; K<=n; K++)
    for(i=1; i<=n; i++)
        c[i][K] = 0
    for(j=1; j<=n; j++)
        c[i][j] += a[i][K] + b[K][j]
```

~~26-7-19~~

Pipelining



Delta



Let No of Inst^r = N

No. of Stages = M

① Time = M + (N-1) CC → clock cycle

for 1st ← for remaining N-1 inst^r being in each cycle one inst^r is executed

$$T = \frac{M+N-1}{\text{Frequency}}$$

② Speed up = $\frac{\text{Time taken w/o pipeline}}{\text{Time taken w/ pipeline}} = \frac{MN \text{ cc}}{M+N-1 \text{ cc}}$

$$S = \frac{MN}{M+N-1}$$

→ Benefit of pipeline is only there when pipeline is filled up, i.e, after M cycles.
→ i.e more the no. of inst^r, more the speedup

$$\text{Max. Speedup} = \lim_{N\to\infty} \frac{MN}{M+N-1} = \lim_{N\to\infty} \frac{M}{\frac{M}{N}+1-\frac{1}{N}}$$

Max Speedup = M = no. of segments.

③ Throughput - No. of inst^r executed per sec.

$$= \frac{\text{Total inst}^r}{\text{Total time}}$$

$$T = \frac{Nf}{M+N-1}$$

$$\text{Max } T = \lim_{N\to\infty} \frac{Nf}{M+N-1} = \lim_{N\to\infty} \frac{f}{\frac{M}{N}+1-\frac{1}{N}}$$

Max. Throughput = f

→ i.e higher the frequency, higher the throughput
→ Hence, we prefer system w/ higher freq

Q How to increase frequency?

→ By increasing the no. of stages / segments, since amount of work to be done in each stage decreases.
→ Higher the V, higher the heat generation.
∴ increase V as per available cooling.

And hence, concept of multicore was generated

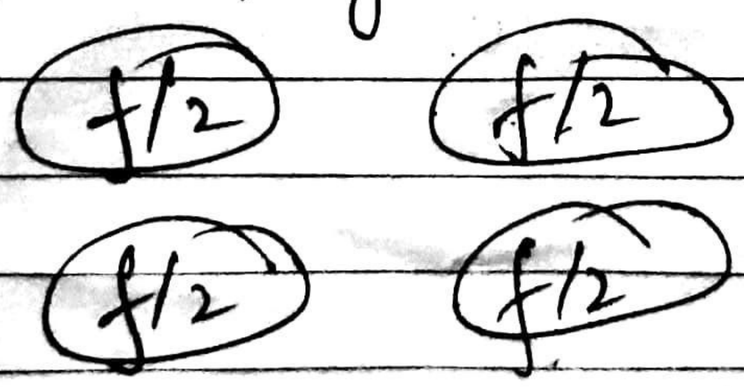
29-7-19

④ Average No. of instr executed per clockcycle

$$= \frac{N}{\frac{N+M-1}{M}} = \frac{MN}{M+N-1}$$

Max = M [1 instr cycle (IC) = M cc]

☆ → If multicore, you use V of 1/2 the original V.



$4 \times f/2 = 2f$ ⇒ faster and heat also less

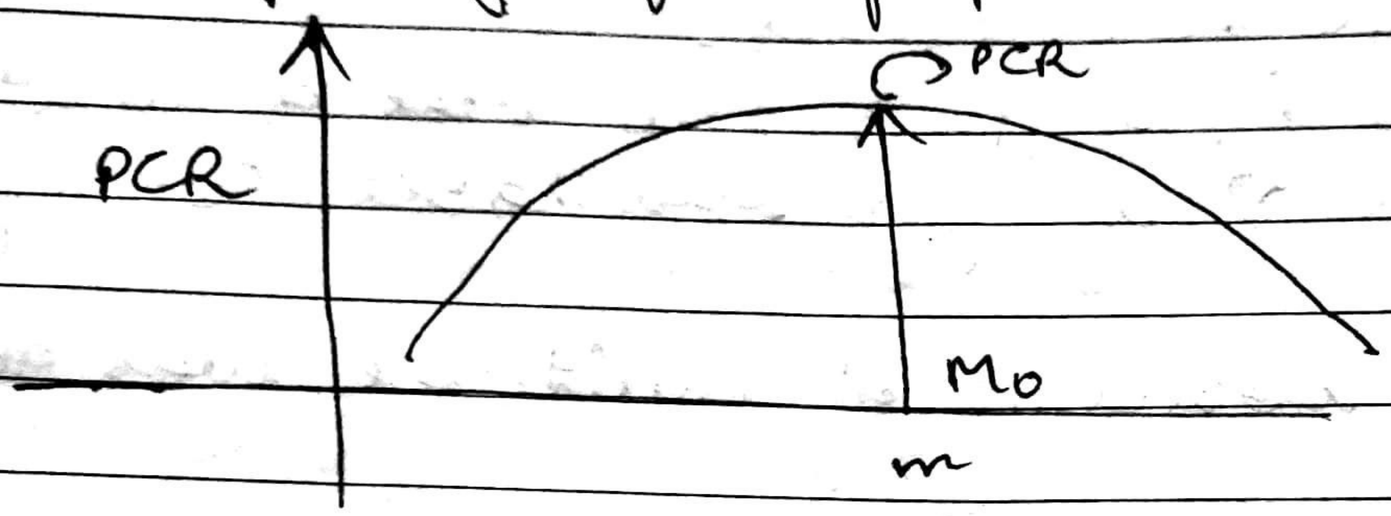
⑤ Performance Cost Ratio (PCR) = $\frac{f}{\text{Total Cost}}$

↳ Throughput → frequency

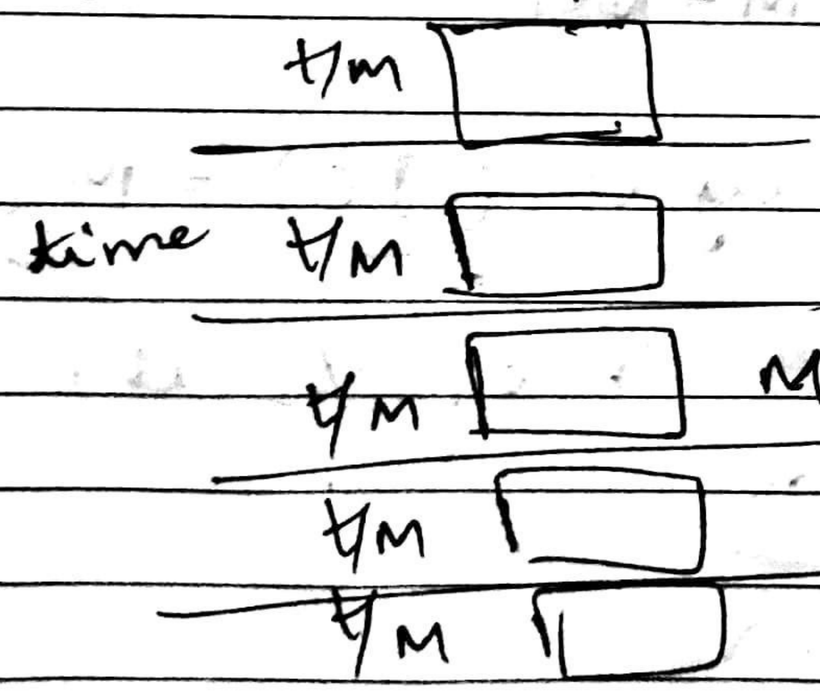
$$PCR = \frac{f}{c+mh}$$

Dt.:
Pg.: Delta

C = cost of digital gates
h = cost of latch
m = no. of stages
f = freq. of pipeline



After mo, even if throughput increases, PCR decreases since cost increases. ∴ we use no. of stages as Mo (optimal no. of stages)



t = time taken by non-pipelined H/W
Let 'd' = time to store result into the latches

Then for each stage, time taken = t/m + d

$$f = \frac{1}{t/m + d}$$

$$PCR = \frac{1}{(t/m + d) * (c + mh)}$$

At what value of m will PCR be max?

Dt.:
Pg.:

Delta

$$\frac{dPCR}{dM} = \frac{h}{(t/M+d)(c+mh)^2} - \frac{t}{m^2(t/M+d)^2(c+mh)}$$

Maximizing $\frac{dPCR}{dM} = 0$

$$\frac{h}{c+mh} = \frac{t}{(t/M+d)m^2}$$

$$h = \frac{tc + mht}{tm + m^2d}$$

$$mht + m^2hd = tc + mht$$

$$m = \sqrt{\frac{tc}{dh}}$$

$\frac{d^2PCR}{dM^2} < 0$ hence maximum value of m is

$$m = \sqrt{\frac{tc}{dh}}$$

31-7-18

→ Frequency decided by every stage must have approx. same work.

→ Time period of pipeline = max ($t_1 + t_2 + t_3 + \dots t_n$) + latch

One branch causes M-1 extra CC

No. of instr = N

No. of stages = M

Prob. that an instr will cause branching = S

No. of instr = N

No. of stages = M

I_1
I_2
I_3
I_4
I_5
I_6 → branch Jmp Next
I_7
I_8 Next I_{12}
I_9 I_{13}
I_{14}
I_{15}
I_{16}

extra CC

Store					1	2	3	4	5	6	①	②	③	⑨	12	11
execute				1	2	3	4	5	6	7				12	11	10
DI			1	2	3	4	5	6	7	8			12	11	10	9
ID		1	2	3	4	5	6	7	8	9		12	11	10	9	8
IF	I_1	I_2	3	4	5	6	7	8	9	10	12	11	10	9	8	7

Dt.:
Pg.:

Delta

① Time = $M + N - 1 + NS(M-1)$

↳ how many branching (alg)

② speedup = $\frac{MN \text{ cc}}{M+N-1+NS(M-1) \text{ cc}}$

Max $\lim_{N\to\infty} \frac{M}{\frac{M}{N} + 1 - \frac{1}{N} + S(M-1)} = \frac{M}{1+S(M-1)}$

if $S = 0$, Max speedup $= M$

③ Throughput $= \frac{Nf}{M+N-1+SN(M-1)}$

Max. $\lim_{N\to\infty} \frac{f}{1+S(M-1)}$

④ Avg. no. of instr executed per cc =

$$\frac{MN}{N+M-1+NS(M-1)}$$

$$= \frac{M}{1+S(M-1)}$$

→ pyq question.
(starting)

Delta

Parallel & Distributed System

No. of Instructions: N

No. of stages in a pipeline = m

Probability that an Instruction is a branch instruction = p

Prob. that a branch inst^r is unconditional branch = q

" " " " " " conditional branch = r

" " " a conditional branch will cause branching = s

Find: Time, Throughput, Speedup, Avg. no. of instruction executed per inst^r set.

Avg. no. of branch Instruction = Np

Avg. no. of conditional branch Inst^r = Npr

" " " unconditional " " = Npq

Avg. no. of branching from cond. branch = Nprs

Avg. no. of Inst^r executed per IC = [redacted]

Avg. no. of branch = Npq + Nprs = Np(q + rs)

Time = m + N - 1 + Np(q + rs)(m - 1) clock cycles

Frequency

$$\text{Throughput} = \frac{Nf}{m + N - 1 + Np(q + rs)(m-1)}$$

$$\text{Speedup} = \frac{\text{Max Throughput}}{\text{Throughput}} =$$

$$\lim_{N \to \infty} \frac{Nf}{M + N - 1 + NP(q + rs)(m-1)}$$

$$= \frac{f}{1 + p(q + rs)(m-1)}$$

if $p = 0$, then max. throughput $= f$

$$\text{Speedup} = \frac{NM}{M + N - 1 + NP(q + rs)(M-1)}$$

$$\text{Max Speedup} = \lim_{N \to \infty} \frac{MN}{M + N - 1 + Np(q + rs)(M-1)}$$

$$= \frac{M}{1 + p(q + rs)(M-1)}$$

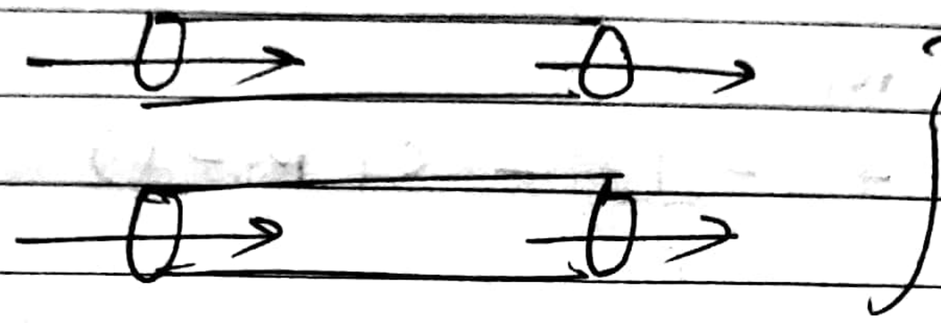
If $p = 0$, Max speedup $= M$.

13-9-19

Scealar pipeline → 1 clock 1 Job
↳ every clock 1 inst^r execute.

Supersecalas pipeline -

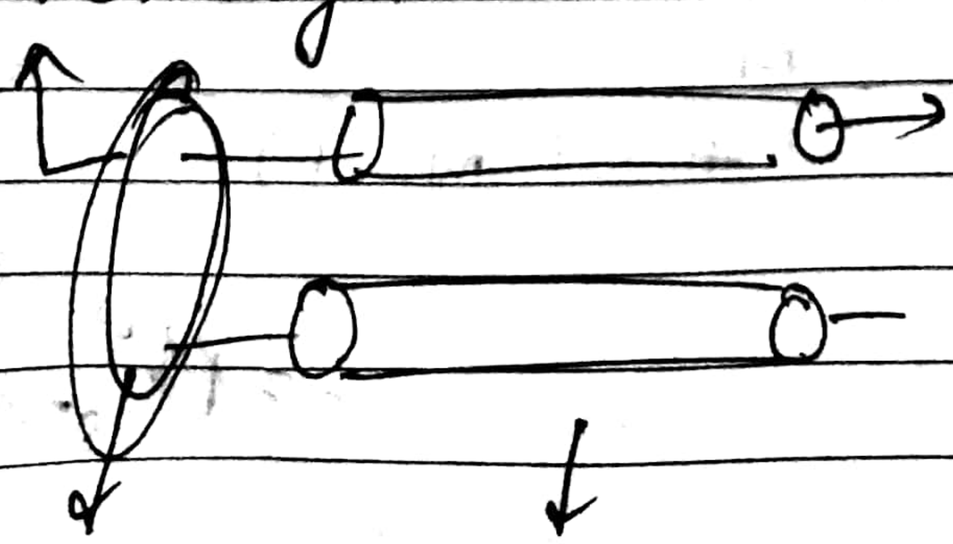
In one clock cycle, more than 1 inst^r can be completed. Process will be fast.
Suppose we have 2 pipeline:



Can have multiple pipelines.
Degree → no. of pipeline.
Degree n → It can perform n Jobs per clock cycle.

Disadvantage!
As instructions as stored in memory & at one time only one inst^r can access memory hence there will be bottleneck.

Soln :-
Have a common hardware for fetching inst^r from memory.

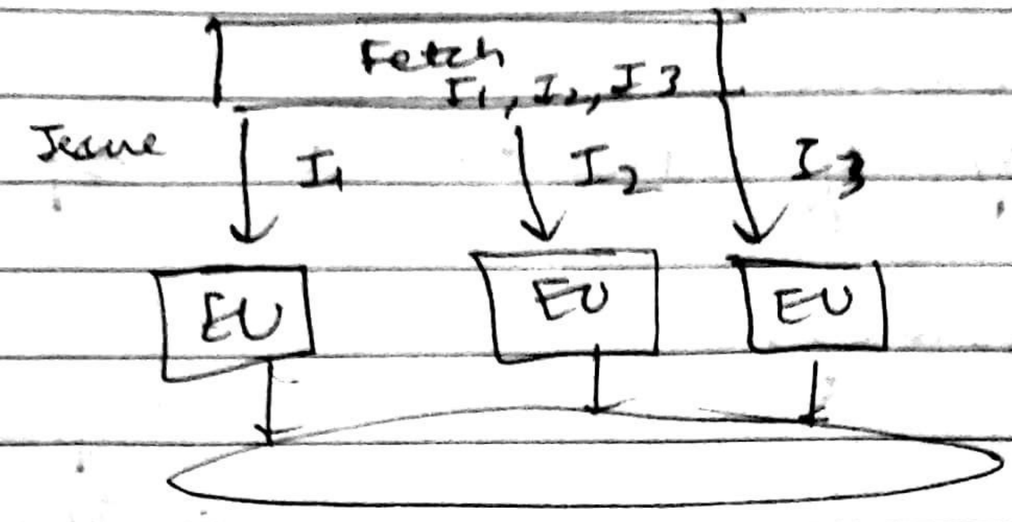


store in processor

Dt.:
Pg.:

Delta

Fetch & issue to free execution unit



All I_1, I_2, I_3 are executing at same time but they should be independent of one another.

Hence we perform Dependency analysis to know which inst^r should be performed parallel execution.

If one inst^r is dependent on other so the next inst^r is execute Hence it goes out of order.

Correctness of program should be maintained.

Commit Unit

By using out of order execution, it becomes faster and store the result in order and present it in correct order to user.

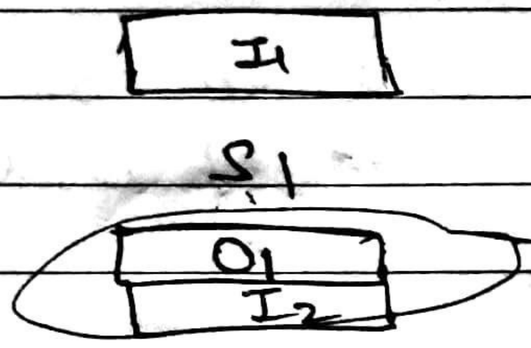
All processors are using out of order execution to save time.

Split cache for Structure Hazard.

Memory of data
& memory of instr are ~~=~~ separated

Dependency Analysis

Input Set = I
Output Set = O
Instr = S



→ output of S_1 is input of S_2

① Flow Dependency → for S_2 to execute, it is imp to get output of S_1.

$O_1 \cap I_2 \neq \phi$

Flow independent if $O_1 \cap I_2 = \phi$
Representation : →

② Anti dependency $O_2 \cap I_1 \neq \phi$

output of $S_2(O_2)$ is overlapping with input of $S_1(I_1)$

Anti independent if $O_2 \cap I_1 = \phi$
Representation : →

③ Output Dependency $O_2 \cap O_1 \neq \phi$

Both try to write at same position/file

DELTA Notebook

Date: Page: Delta

Representation: →

Output independent if $O_2 \cap O_1 = \phi$

Bernstein's Conditions

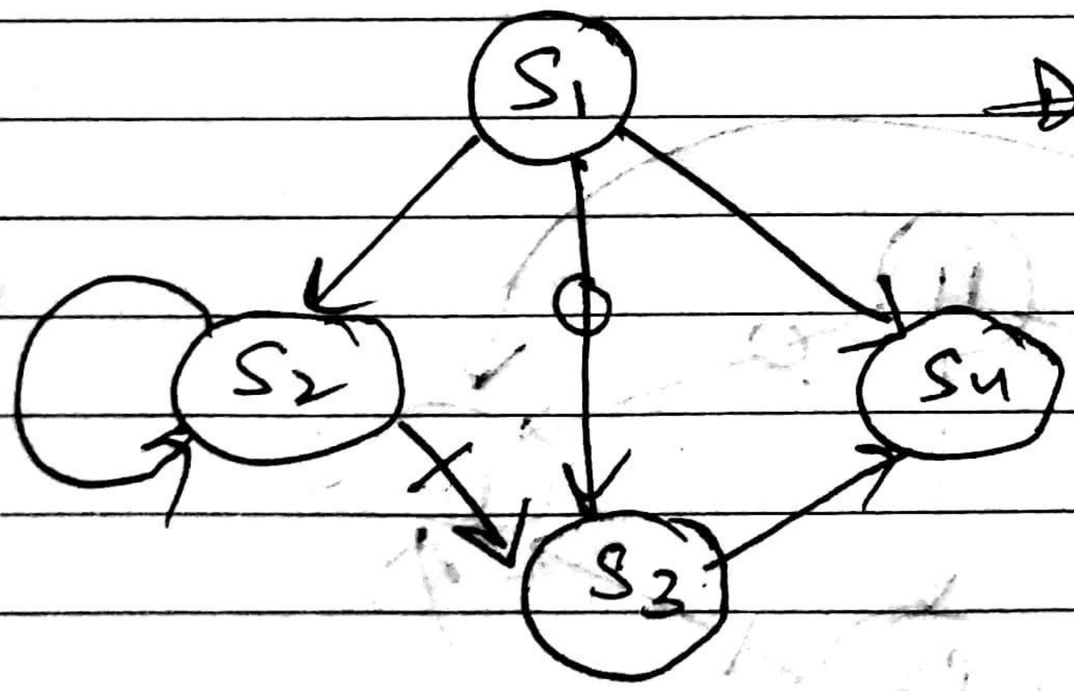
For S_1 & S_2 to execute simultaneously if

$$\begin{bmatrix} O_1 \cap I_2 = \phi \\ O_2 \cap I_1 = \phi \\ O_2 \cap O_1 = \phi \end{bmatrix}$$

eg → (1)

S1: load R1, A
S2: Add R2, R1
S3: MOV R1, R3
S4: Store B, R1

Dependence Graph



$P_1 \| P_2 \| P_3 \| P_4 \cdots \| P_K$ if and only if $P_i \| P_j \ \forall \ i \neq j$

can be scheduled for parallel execution

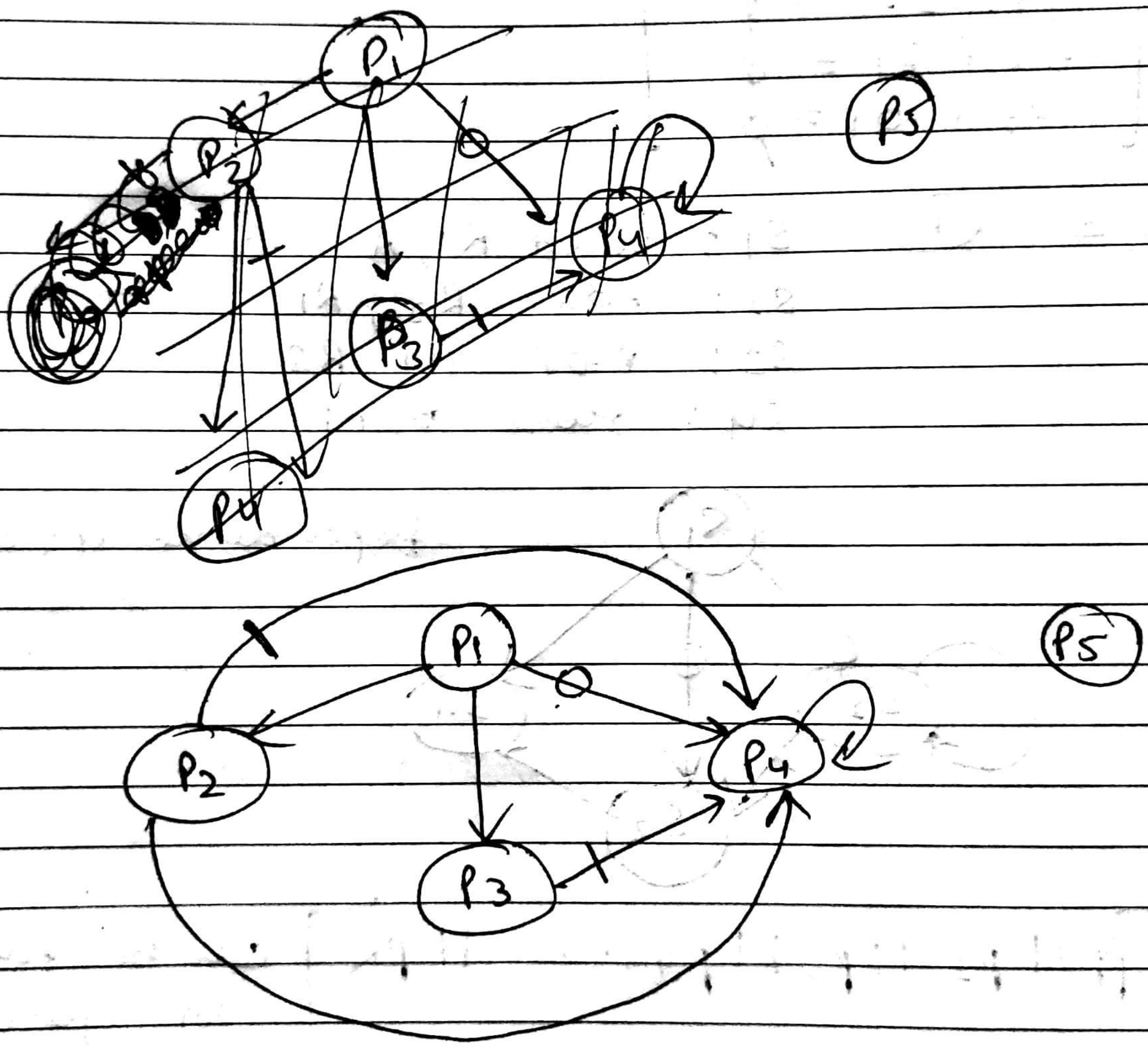
↳ each pair can be parallelly executed

Dt.:
Pg.:

Delta

(2)

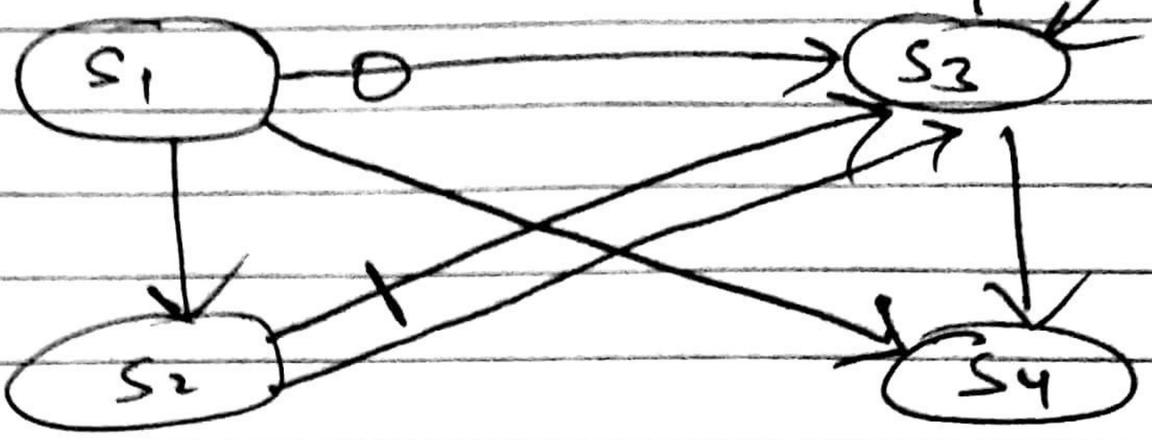
P1 : C = D X E
P2 : M = G + C
P3 : A = B + C
P4 : C = C + M
P5 : F = G - E



Q) S1 : A = B + D
S2 : C = A x 3
S3 : A = A + C
S4 : E = A/2

Dt.:
Pg.:

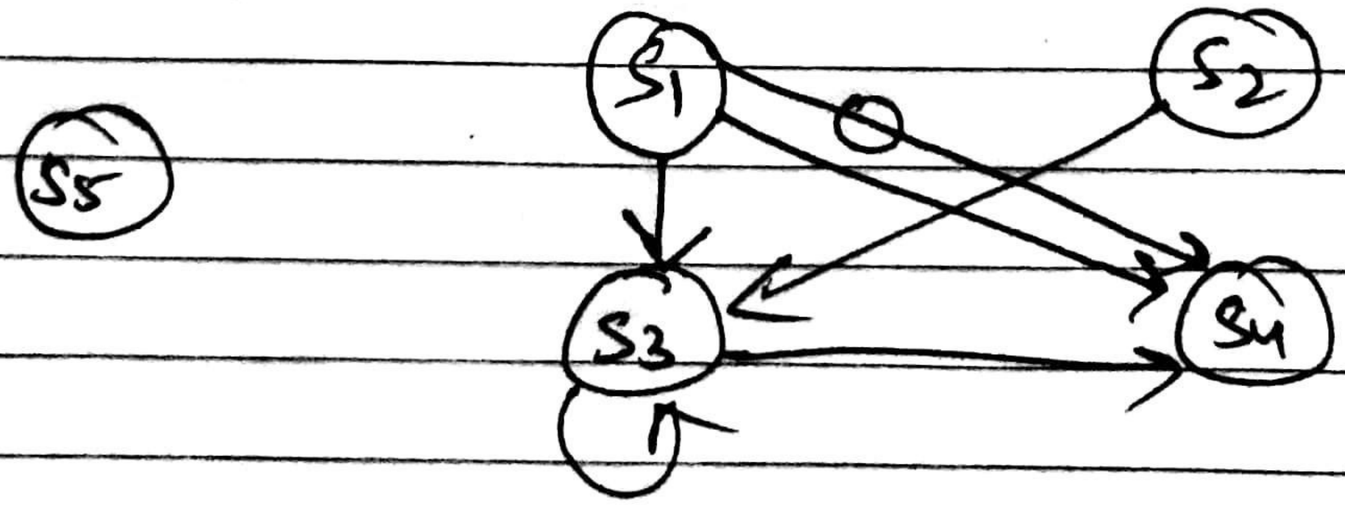
Delta



Q S1: Load R1, 1024 R1 ← 1024
S2: Load R2, M(10) R2 ← MEM(10)
S3: Add R1, R2 R1 ← R2 + R1
S4: Store M(1024), R1 MEM(1024) ← R1
S5: Store MEM(R1), 1024 MEM(R1) ← 1024

① Flow : S1 → S3 S1 → S4
S2 → S3 S3 → S4.

② Output : S1 → S4



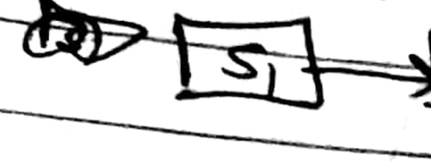
3-9-19

Dt.:
Pg.:

Delta

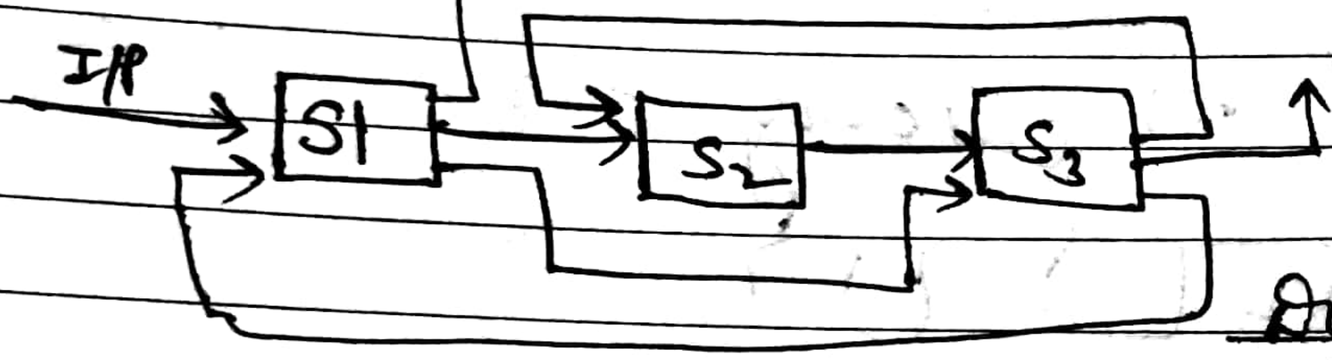
Reservation Table of Pipeline

~~Linear Pipeline~~



	1	2	3	4	5	6
S1	X1	X2	X3			
S2		X1	X2	X3		
S3			X1	X2	X3	
S4				X1	X2	X3
S5					X1	X

output X



Dynamic Pipeline and non-linear ||

Reservation Table for X

	1	2	3	4	5	6	7	8
S1	X					X		X
S2		X		X				
S3			X		X		X	

Reservation Table for X

	1	2	3	4	5	6
S1	X				X	
S2			X			
S3		X		X		X

Choose time latency (avoid) so that there is no collision. latency should be as min as possible as then Throughput will be maximum.

(MAL) → Minimum Average latency

After 1 cycle = Latency = 1

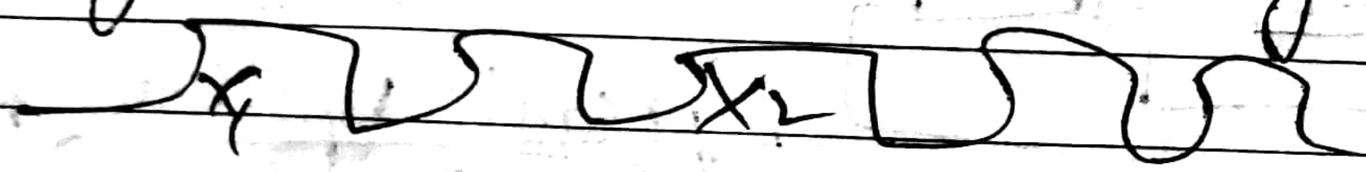
Dt.: Pg.:

Delta

Reservation Table of X

	1	2	3	4	5	6	7	8	
S1	X1	X2				X1	X2	X1	X2
S2		X1	X2	X1	X2				
S3			X1	X2	X1	X2	X1	X2	

After 2 C.C. ⇒ Latency = 2

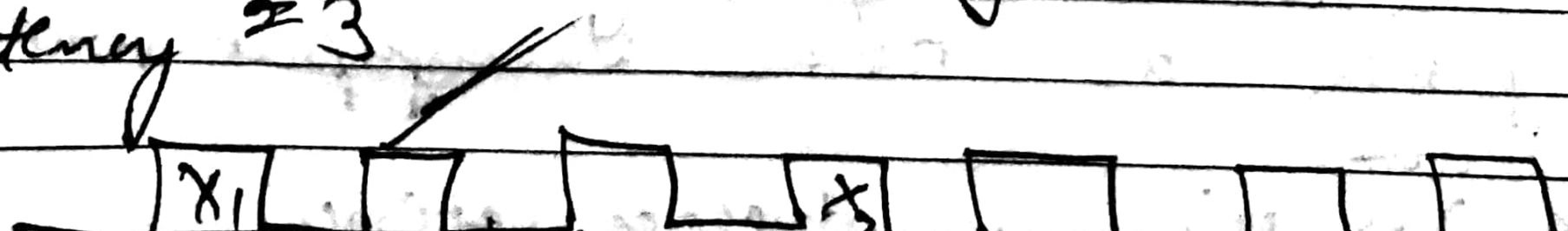


	1	2	3	4	5	6	7	8
S1	X1		X2			X1		X1/X2
S2		X1		X1/X2		X2		
S3		X1		X1/X2		X1/X2		

This Latency which causes collision → Forbidden Latency

" " " not " " → Permissible Latency.

Latency = 3



	1	2	3	4	5	6	7	8	9	10	11
S1	X1			X2		X1		X1		X2	X2
S2		X1		X1	X2		X2				
S3			X1		X1	X2	X1	X2		X2	

Permissible latency = 1, 3, 6, 8, ...
Forbidden " = 2, 4, 5, 7

Collision vectors

Collision Free scheduling.

max forbidden.

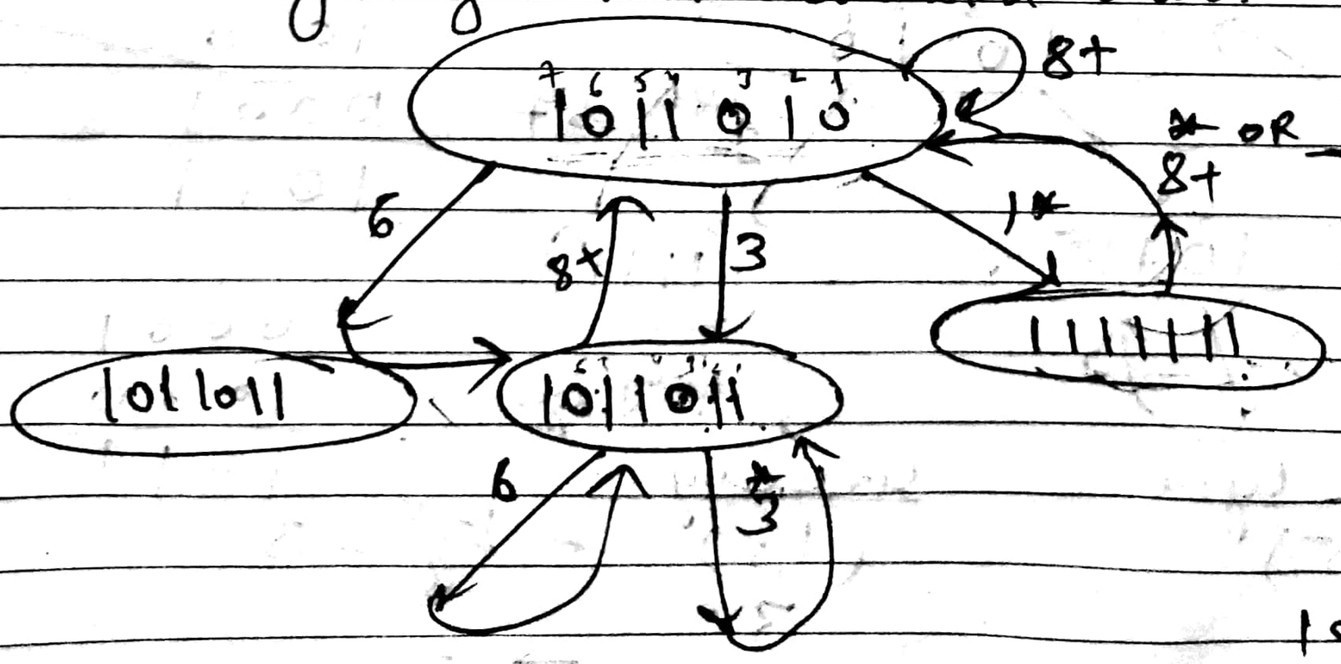
7 6 5 4 3 2 1
1 0 1 1 0 1 0 ⇒ Initial Collision vector

	1	2	3	4	5	6	7	8	
S_1	X					X		X	2, 5, 7
S_2		X		X					2
S_3			X		X		X		2, 4

Forbidden latency.

Directly by getting pair wise difference.

Starting stage → initial collision vector.



1011010
OR 0101101
1111111

1011010
0000001
1011011

1011010
0010011
1001001

1011010
0010111
1001101

0101101
1011010
1111111

0010111
1011010
1001101

1011010
0000000
1011010

That simple cycle which contain min edge from each node

Greedy Cycle

Simple cycle
(1, 8)
(8)
(3, 6)
(3, 8)
(6, 8)
(6)
(3)

(1, 8)
(3)
" " = 3

Avg ~~lateny~~ latency = $\frac{1+8}{2}$
= 4.5

Ans ⇒ 3

Reservation Table for Y

	1	2	3	4	5	6	
S_1	X				X		4
S_2			X				
S_3	X			X		X	2, 4

4 3 2 1
1 0 1 0

5↑

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

5↑ 3 1* 5*↑

1 0 1 1

1 1 1 1

outgoing edges

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

Simple cycle
(1, 5)
5
3
(3, 5)

Greedy
(1, 5) → $\frac{1+5}{3}$ = 3
3
→ 3

Delta

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
S_1	X_1			X_2		X_1	X_3	X_1	X_2	X_4	X_2	X_3	X_5	X_3
S_2		X_1		X_1	X_2		X_2	X_3		X_3	X_4	X_4		
S_3			X_1		X_1	X_2	X_1	X_2	X_3	X_2		X_4	X_3	

Efficiency $= \frac{8}{9} \times 100$

$= 88.88\%$

Throughput $= \frac{1}{MAL * 3} = \frac{1}{MAL}$ Jobs/sec

(1 ref.)

Time Period

↳ Min & Max Value

Dt.: Pg.: Delta

Bounds on MAL

(1) The MAL is lower bounded by maximum no. of check marks in any row of Representation Table.

> Maximum no of check marks in any Row of RT. $\leq$ MAL $\leq$ Avg latency of any greedy cycle in the SD.

(2) MAL is lower than or equal to the average latency of any greedy cycle in the ~~state~~ diagram. (SD)

(3) Average latency of any greedy cycle is upper bounded by no. of 1's in initial collision vector + 1

..... $\leq$ No. of 1's in Initial collision vector + 1

For X

$\frac{3}{\uparrow} \leq MAL \leq 5$ $\Rightarrow$ Ans as calculated before = 3 (optimized)

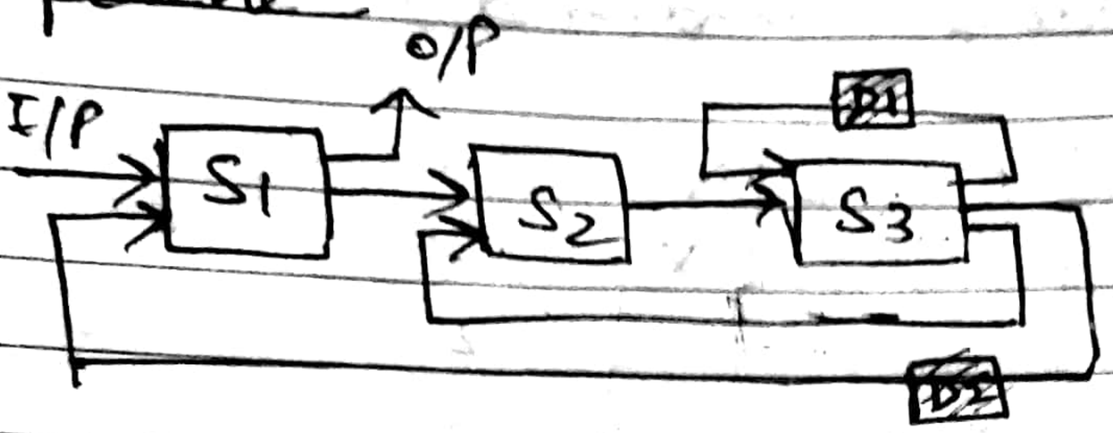
For Y

(3) $\leq MAL \leq 3$ $\Rightarrow$ Ans as calculated before = 3 (optimized)

Dt.:
Pg.:

Delta

Pipeline



Reservation Table

	1	2	3	4	5
S_1	X				X
S_2		X		X	
S_3			X	X	

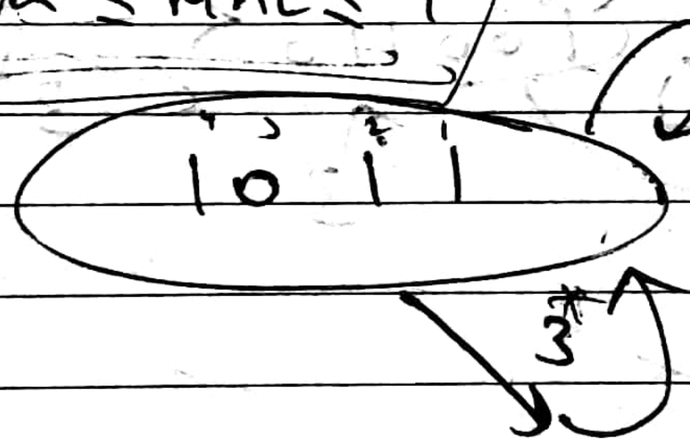
4 3 2 1
1 0 1 1

4
2
11

Bounds

$2 \leq MAL \leq 4$

5+



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

Simple cycle	Greedy
3	3
5	

MAL = 3

Hence not optimal as if MAL = 2 we get max. throughput.

Hence to optimize MAL we insert delays

① Find RT after inserting delay
② Then find MAL

DELTA Notebook

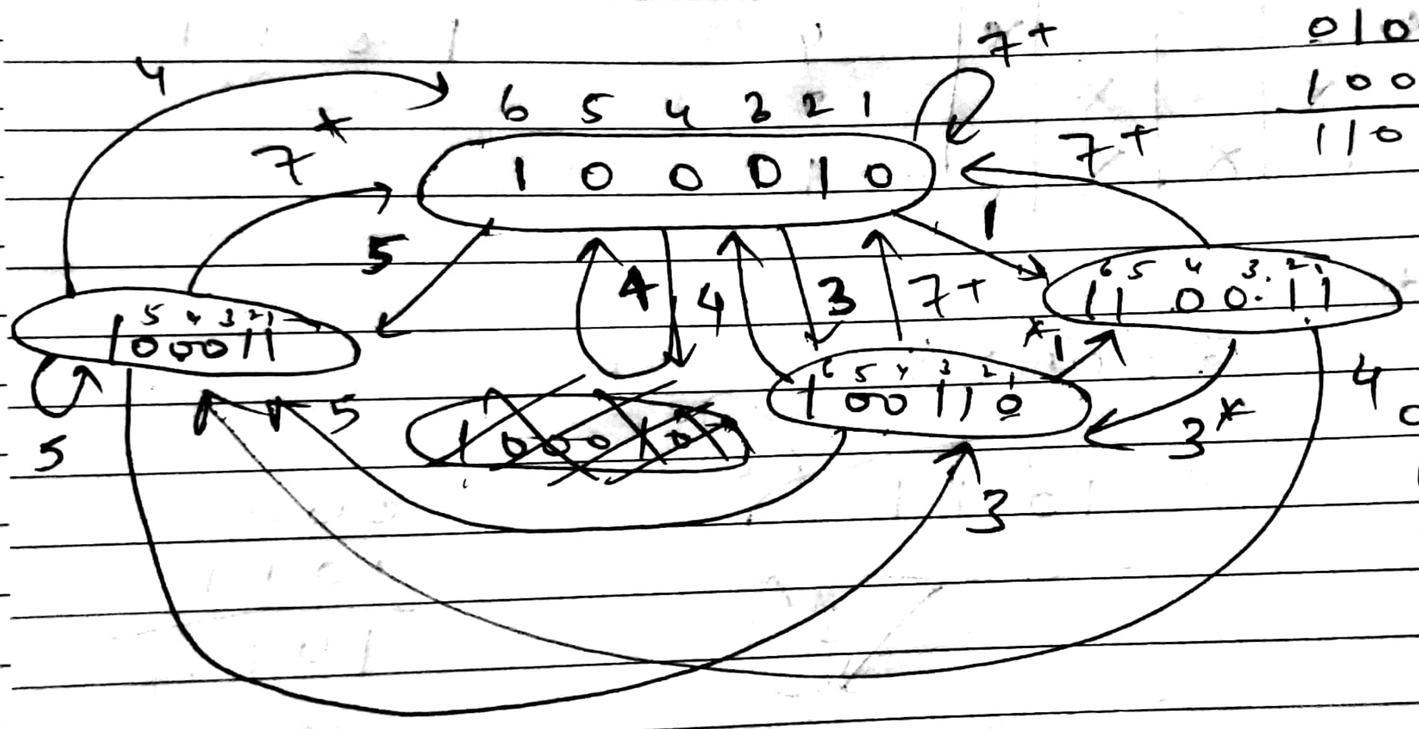
Use any no. of delays wherever we get optimal
Delays can improve/optimize MAL.

Dt.:
Pg.:

Delta

Reservation Table (Modified)

	1	2	3	4	5	6	7	
S1	X						X	6
S2		X		X				2
S3			X		X			2
D1				D1				
D2						D2		



Simple cycle
1, 7
3, 7
4
5, 7
7
3, 1

Greedy cycle
1, 3 = (2)
3
3

MAL = 2

000100
100010
100110

010001
100010
110011

000010
100010
100010

600001
100010
100011

000110
100010
100110

000011
100010
100011

010011
100010
110011

000010
100010
100010

DELTA Notebook

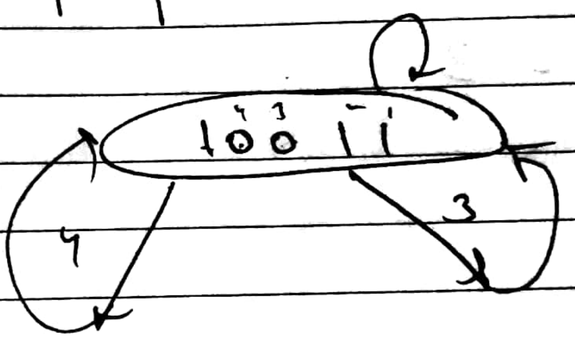
Dt.:
Pg.:

Assignment /simulate

Q WAP to find collision free scheduling of a pipeline

Q1

	1	2	3	4	5	6	
S_1	X					X	5
S_2		X		X			2
S_3			X				
S_4				X	X		1



5 4 3 2 1

1 0 0 1 1

MAL = 3

00010
10011
10011
00001
10011
10011

Throughput = $\frac{1}{MAL * T}$

$= \frac{1}{3 \times 20\,ns} = \frac{10^9}{60}$ Jobs/sec

$= \frac{1000}{5} \times 10^5$ Jobs/sec

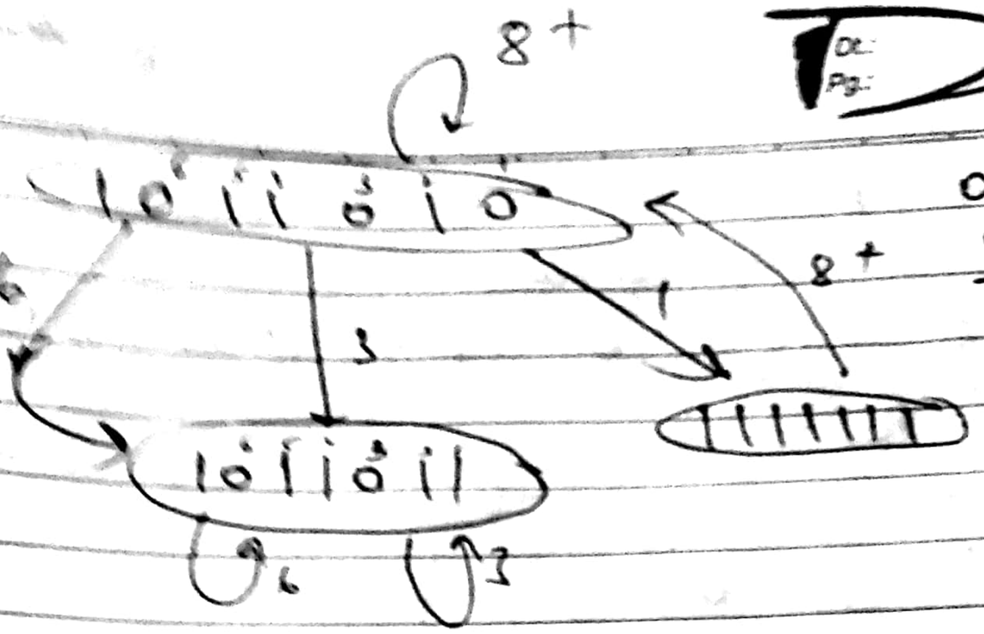
Q2

	1	2	3	4	5	6	7	8	
S_1	X					X		X	2, 7, 5
S_2		X		X					2
S_3			X		X		X		2, 4

7 6 5 4 3 2 1

1 0 ϕ ϕ 0 1 0

Delta



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

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

MAL => $\frac{1+8}{2} = 4.5$

OR (3).

MAL = 3