Cache Mapping

Q1 Explain direct mapping, associative mapping & set associative mapping in brief.

Ans **Direct Mapping :-**

The simplest technique, which maps each block of main memory into only one possible cache line. or In Direct mapping, assign each memory block to a specific line in the cache.

If a ~~new~~ line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed.

The cached is used to store the tag field whereas the rest is stored in the main memory. Direct mapping's performance is directly proportional to the Hit ratio.

Formula used for mapping is :-

$$i = j \% m$$

where: i = cache line number
j = main memory block number
m = number of lines in the cache.

Tag	Lines	Block Offset

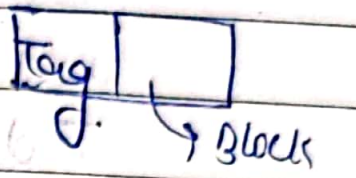
Associative / Fully Associative mapping

→ Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache. In this case the control logic interprets a memory address simply as a tag & a word field.

Adv: → fast & easy to implement.
→ we need replacement algo to maximize hit ratio.

Date __/__/__
Page

Disadv :- → Complex circuitry req. to examine the tags of all cache line in parallel.
→ No. of comparison increases.



Set Associative mapping :-

→ It is a mixture of direct mapping & fully associative mapping. by arrangement lines of a cache into sets.

→ The sets are persistent using a direct mapping scheme. However the lines within each set are treated as a small fully associative cache where any block that can save in the set can be stored to any line inside the set.

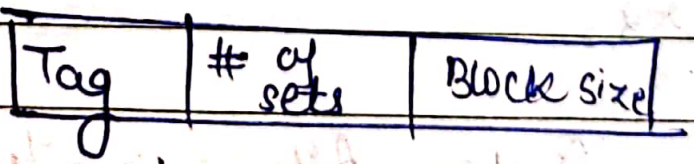


Fig: memory box.

Date __/__/__
Page

Q.2 For a direct mapped cache design with a 32 bit address & byte addressable memory the following bits of the address are used to access the cache : Tag (22 bits) Index (5 bits), Offset (5 bits). Ans the following:-

i) What is the cache line size (in words)?

Ans Given :- Offset = 5 bits, & {block offset = cache line size}

∴ Size of cache line = 2^5 words
= 32 words

ii) How many entries (cache lines) does the cache have? Cache line are also called blocks.

Ans Given Index bits = 5 bits
Index bits = bits required to present the number of cache lines.

∴ Num of cache lines = 2^5 = 32 lines

Q3) Consider a direct mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find the number of bits in tag.

Ans) Given: Cache size = 16 KB

Bits require to represent cache size = $2^4 \times 2^{10}$

$= 2^{14}$

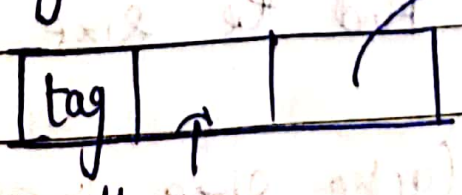
= 14 bits

Block Size = 256 bytes

main memory size = 128 KB

tag bits = ?

On direct mapping :-



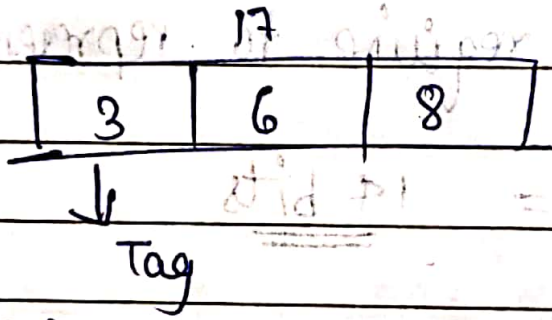
Date __/__/__
Page

# of bits require for block size = 8 bits

PA size (in bits) = 17 bits.

# of cache lines = Cache size / block size = 16 KB / 256 B

$= \frac{2^{14}}{2^8} = 6$ bits



∴ Tag bits = 3 bit

Q) Consider a direct mapped cache of size 512 KB with block size 1KB. There are 7 bits in the tag. And the size of main memory?

Ans Given:- Cache size = 512 KB

Block size = 1KB

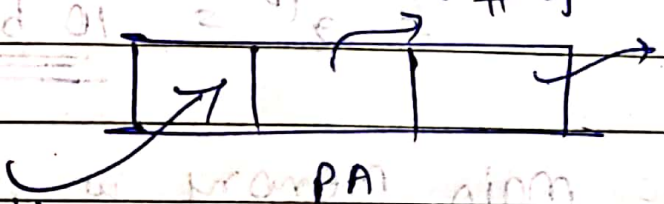
Tag bits = 7 bits.

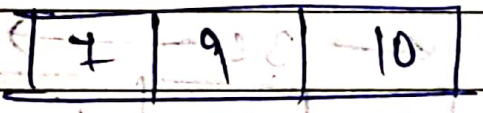
bits for block size = 10 bits

of cache line = $\frac{\text{Cache size}}{\text{Block size}} = \frac{512\,KB}{1\,KB}$

= 9 bits

PA is denoted by :-





∴ # of bits in PA (main mem) = 7 + 9 + 10

= 26 bits

∴ main mem size = $2^{26} = 2^6 \cdot 2^{20}$ = 64 MB

Q5) Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32 bit address. Cal. the number of bits needed

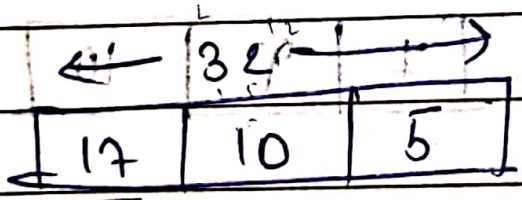
for cache indexing & no. of tag bits?

Ans) Given: Cache size = 32 KB
Block size = 32 bytes

# of cache lines = $\frac{32\ KB}{32\ B} = 1000$

$= 2^{10} = 10$ bits

Given: main memory is represent by 32 bits.



∴ Tag bits = 17 bits
Cache Indexing bits = 10 bits

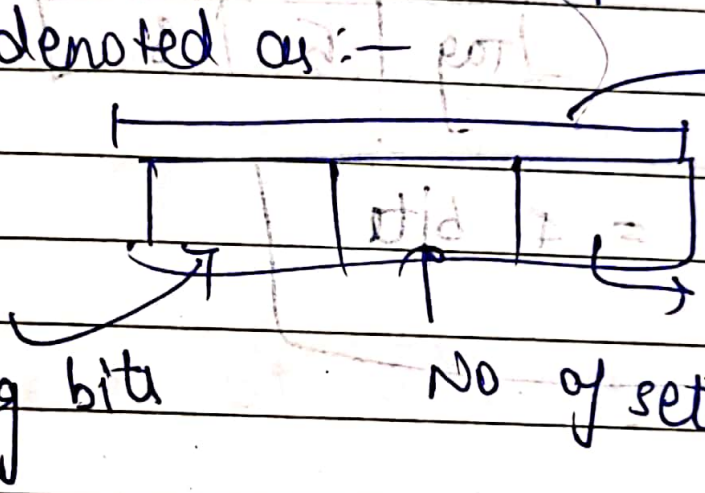
Q6) Consider a 2 way set associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is

128 KB. And number of tag bits?

Ans Given: Cache size = 16 KB
Block size = 256 bytes
Main mem. size = 128 KB.

In set associative mapping:—

PA is denoted as:—



PA in bits = $2^7 \cdot 2^{10}$ = 17 bits

~~Number of cache lines = 16 KB = 16000 = 256 B 64 16~~

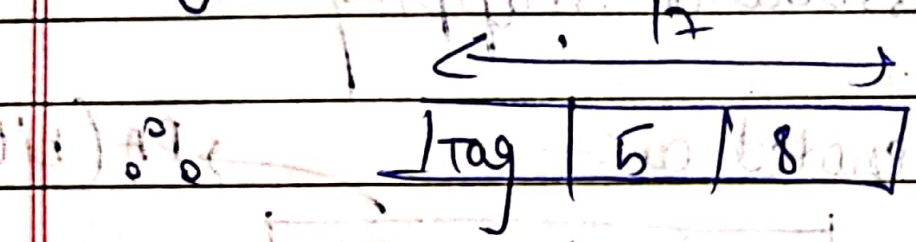
# of cache lines = $\frac{16 \text{ KB}}{2^8 \text{ B}} = \frac{2^{14}}{2^8} = 2^6$

= 64 lines

# of sets = $\frac{64}{2}$ { # of cache line } / { set size }

= 32

# of req. to represent # of sets = 5 bit



Tag bits	= 4 bits

Q1) Consider a 8 way set associative mapped cache of size 512 KB with block size 1 KB. There are 7 bits in the tag field size of main memory?

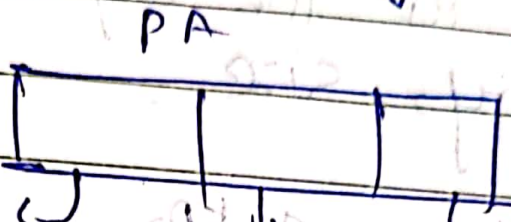
Ans) Given: Cache size = 512 KB

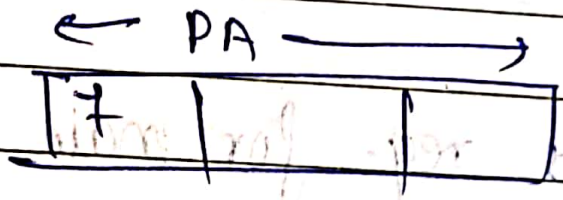
Block size = 1 KB

Tag bits = 7 bit

PA size = ?

In set associative mapping PA is denoted by:-



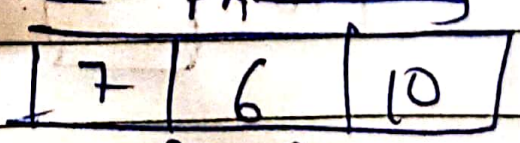


Block size in bits = 1 KB = 10 bits

# of cache lines = $\frac{512\ KB}{1\ KB}$ = 512

# of sets = $\frac{\text{Cache Line}}{\text{set size}}$ = $\frac{512}{8}$ = 64

$\therefore$ bits req. for sets = 6 bits



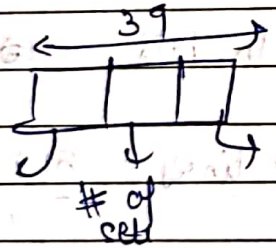
PA = 7 + 6 + 10 = 23 bits

PA size = 2^{23} = 8 MB //

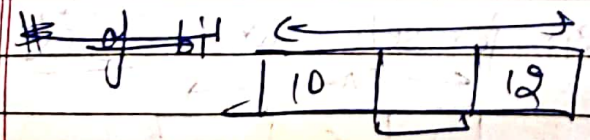
Q8) Consider a 4 way set associative mapped cache with block size 4 KB. The size of main memory is 16 GB & there are 10 bits in the tag. Find the cache memory size.

Ans) Block size = 4 KB
main mem size = 16 GB
Tag bits = 10 bits

# of bits req. for main mem = $2^4 \cdot 2^{30} = 34$ bits



# of bits for block size = $2^2 \cdot 2^{10} = 2^{12}$ → 12 bits



∴ # of bits req. for # of sets = 34 − 22 = 12 bits

∵ # of sets = 2^{12}

$$\#\text{ of sets} = \frac{\#\text{ of cache lines}}{\text{set size}}$$

$$2^{12} = \frac{\#\text{ of cache lines}}{4}$$

$2^{12} \cdot 2^2$ = # of cache lines

2^{14} = # of cache lines.

∴ Cache mem. size = # of cache line * Block size

$= 2^{14} * 2^{12}$

$= 2^{26} = 2^6 \cdot 2^{20} =$ 64 MB

Q9) Consider a 4 way set associative mapped cache with block size 4 KB. The size of main memory is 16 GB & there are 10 bits in the tag. Find the size of cache memory?

Ans Given :- Block size = 4KB

main. mem. size = 16 GB

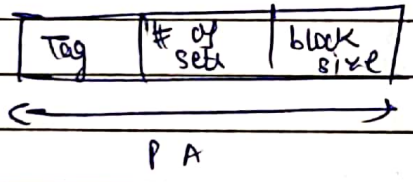
Tag bits = 10

To find. Size of cache memory = ?

given: 4-way set associative map.

∴ the set associative mapping

main mem is represented as :-

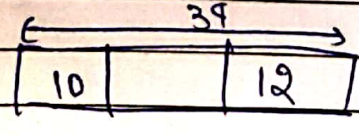


∴ # of bits req. to req rep. PA = $2^4 \cdot 2^{30}$

= 34 bits

block size (in bits) = $2^2 \cdot 2^{10}$ = 12 bits

# of cache blocks =



∴

Solved Previously

Q10) Consider a system with 2 level caches. Access time of level 1 cache, level 2 cache & main memory are 1 ns, 10 ns and 500 ns respectively. The hit rates of level 1 & level 2 caches are 0.8 & 0.9 respectively what is the avg access tym of sys. ignoring the search time within the cache ?

Ans Level 1 : Access tym (t_1) = 1 ns.

hit ratio (h_1) = 0.8

Level 2 : Access tym (t_2) = 10 ns

hit ratio (h_2) = 0.9

Mem. access tym (t_3) = 500 ns.

~~EMAT = (hit ratio) $h_1 t_1$ + (1 - h_1) $t_2 \cdot h_2$~~

~~+ (1 - h_1)(1 - h_2) $\cdot t_3$~~

Tavg = $h_1 t_1$ + (1 - h_1)($t_1 + t_2$) h_2

+ (1 - h_1)(1 - h_2)($t_1 + t_2 + t_3$)

= (0.8)(1) + (0.2)(1+10) 0.9 + (0.2)(0.1)(1+10+500)

= 13 ns //

Pipelining :-

Q1 What is arithmetic pipeline & instruction pipeline ?

Ans1 Arithmetic Pipeline :-

- Arithmetic pipeline divides an arithmetic problem into various sub-problems for execution in various pipeline segments
- Its used for floating point operations multiplication & various other operations

Instruction Pipeline :-

- In this a stream of instructions can be executed by overlapping fetch, decode & execute phase of an instruction cycle.
- This type of technique is used to inc. the throughput of computer system.
- An inst. pipeline needs inst. from the memory while previous inst. are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously

Q2 Explain floating point addition stages using pipeline.

Ans1 Floating point addition is performed by arithmetic pipeline.

The add^n operation is being divided to various suboperation as stages :-

i) Compare the exponents by subtraction
ii) Allign the mantissas
iii) Add the mantissas
iv) Normalize the result.

Let's take an example :-

Let $X = A * 2^a = 0.9504 * 10^3$

$Y = B * 2^b = 0.8200 * 10^2$

where A & B = mantissa

a & b = exponents

Step 1: Comparing exponents by Substraction :-

The diff of exponents i.e. $3 - 2 = 1$ determines how many times the mantissa associated with the smaller exponent must be shifted to right.

Step 2: Allign the mantissa as :-

The mantissa associated with smaller exponent is shifted according to diff of exponents determined in Step 1.

$\therefore X = 0.9504 * 10^3$

$Y = 0.08200 * 10^2$

Step 3: Add mantissas :-

$Z = X + Y = 1.0324 * 10^3$

Step 4: Normalize the result :-

$Z = 0.10324 * 10^4$

Q3) Explain the following terms: latency, throughput, throughput in ideal or max cond^n, speedup ratio, speedup time in ideal or max. cond^n, efficiency.

Ans Latency :- After how much time the new input is given to the system.

Throughput :- Number of operation performed/unit time.

$$\text{Throughput} = \frac{n}{(K+n-1)\cdot tp}$$

In ideal case :- $K \ll n$; then ignore $(K-1)$

$$\text{Throughput}_{(\text{ideal})} = \frac{1}{tp}$$

Speed up Ratio :- Performance of pipeline is given by speed up ratio.

$$S = \frac{\text{Non Pipeline Processing tym}}{\text{Pipeline}}$$

$$S_{\text{ideal}} = \frac{tn}{tp} \qquad \text{Savg} = \frac{n * tn}{(K+n-1)tp}$$

Efficiency :- It's the %age of time when processor are busy over total time taken.

$$\eta = \frac{n}{[K+n-1]}$$

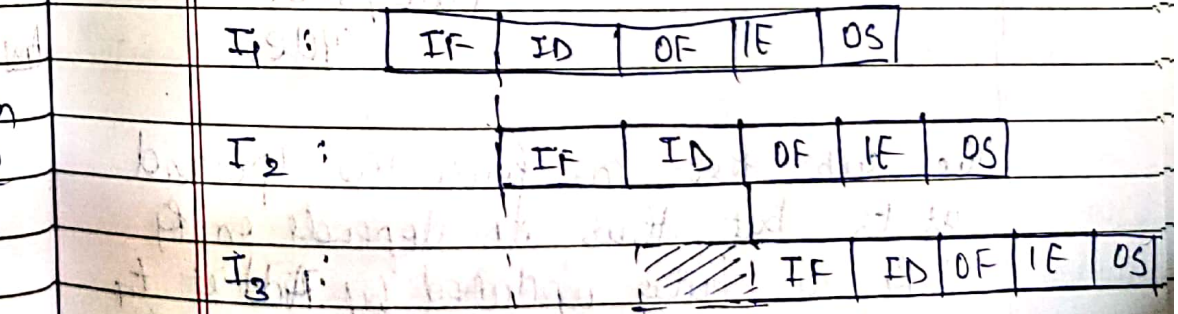
→ In ideal cond^n $K \ll n$ ignore $K-1$

$$\therefore \eta_{\text{ideal}} \to 1$$

Q4 Explain Instruction hazards & their solutions with neat & clear diagrams

Ans There are 3 types of instruction hazards :-

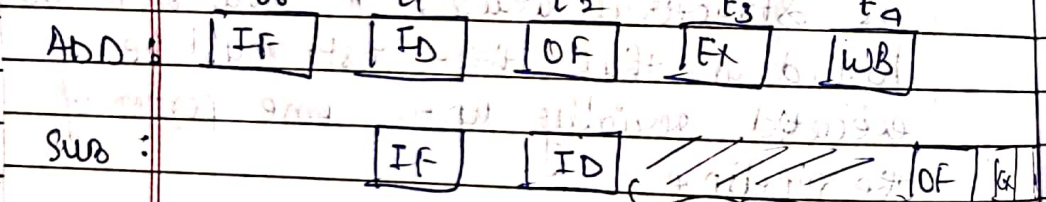
i) Resource Conflict / Structural Hazard :- When 2 pipeline instruction or even more want to access the same resource it results in resource hazards. It is also termed as structural hazard. A solution to this hazard is that these inst. must be executed serially upto some portion of the pipeline



ii) **Data Hazard :-** It's a cond^n when accessing an operand location creates conflict. Consider that you have 2 Inst :-

ADD R_1, R_2, R_3
SUB A_1, R_1

Obs. the Inst. the result of add is stored to R_1 after execution. This R_1 act as an operand for Sub. Inst.



The sub. Inst. can fetch the operand at t_3 but thus it depends on R_1 & R_1 is being updated by Add at t_4 ∴ Sub has to wait idle until t_4 is executed.

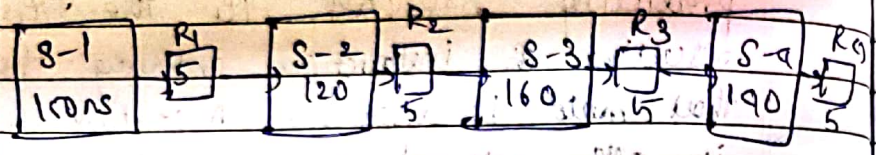
iii) **Control Hazard / Branch difficulty :-** occurs when Inst. pipelining fails in predicting branches i.e. the Inst. resulting in instructions entering the pipeline that must be discarded. A control hazard is often referred to as a branch hazard.

Conditional hazard Solution :-

i) Stall
ii) Prediction
iii) Dynamic Branch Prediction
iv) Reordering Instructions.

Q5) A 4 stage pipeline has the stage delays as 150 ns, 120 ns, 160 ns and 140 ns respectively. Registers that are used between the stages have a delay of 5 ns each. Assuming constant clocking rate, the total time taken to process 1000 data items on this pipeline will be ____ ns?

Ans Given:-



Total time taken to process 1000 data items = ?

Total cycle time = $(k+n-1)t_p$

where k = no. of stages = 5

n = no. of data = 1000

t_p = (max of all stage delay) + (Reg. delay)

$t_p = (160 + 5)\,ns$

$= 165\,ns$

∴ Total cycle time $= (4 + 1000 - 1) \times 165$

$= 1003 \times 165\,ns$

$= 16995\,ns$

$= 16.995\,\mu s$

Q6 Consider the following processes. Assume that the pipeline register have zero latency.

P1: Four stage with latencies 1ns, 2ns, 2ns, 1ns

P2: —11— —1— 1ns, 1.5ns, 1.5ns, 1ns

P3: 5 stage pipeline with latencies 0.5ns, 1ns, 1ns, 0.6ns, 1ns

P4: 5 stage pipeline with latencies 0.5ns, 0.5ns, 1ns, 1ns, 1.1ns.

Which processor has the highest peak clock frequency? [Note: frequency $\propto \frac{1}{\text{cycle time}}$]

Ans Lets assume there are n task for each process :-

Cycle time for P1:-

Given : $k = 4$

$t_p = 2\,ns$

∴ Cycle time $= (k+n-1)\,t_p$

$= (4+n-1)\,t_p$

$= (3+n)\,2\,ns$ //

Cycle time for P2 :-

Given: K = 4
tp = 1.5 ns

∴ Cycle time = (k+n-1) tp
= (3+n) 1.5 ns

Cycle time for P3 :-

Given: K = 5
tp = 1 ns

∴ Cycle time = (k+n-1) tp
= (5+n-1) · 1 ns
= (4+n) ns

Cycle time for P4 :-

Given K = 1
tp = 1.1 ns

∴ Cycle time = (k+n-1) tp
= (4+n) 1.1 ns

∴ By comparing the cycle time of P1, P2, P3, P4

we get that the cycle time of P3 is least.

& given that frequency ∝ 1/cycle time

∴ Process P3 will have the highest peak clock frequency

Q7) Explain all the reasons of Stall cycle?

Ans)

Q8) What is data dependency & possible solution for data depency?

Ans) When in an instruction the data is modified in different stages of a pipeline with the

help of instructions that data dependency, in this case, the data hazard will occur.

→ When instructions are read/write the registers that are used by some other instructions, in this case, the instruct hazard will occur resulting in a delay in the pipeline.

Different types of Data Hazards are:-

i) RAW (Read After Write)
ii) WAR (Write after Read)
iii) WAW (Write after Write)

The WAR is very uncommon or impossible

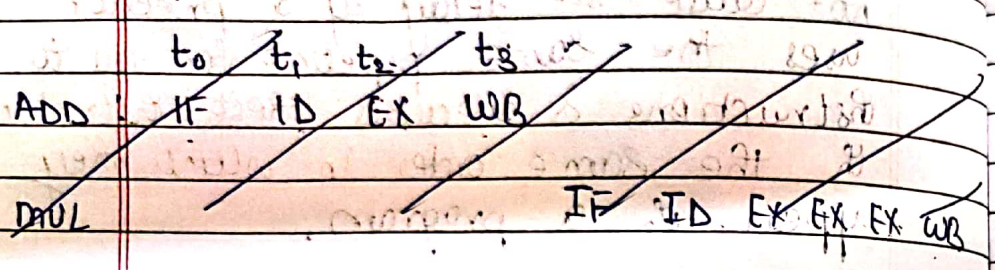
∴ To avoid RAW & WAW the processor should use different register to generate the output of each instruction if it contains an number of register

→ The WAR & WAW hazard will not cause the delay if a processor uses the same pipeline for all the instructions & executes these instructions in the same order in which they appear in a program.

Q9 Consider a pipelined processor with the following 4-stages, IF, ID, EX, WB The IF, ID, WB stage take one clock cycle to comple the operation The num of cycle for the EX stage depends on the instruction ADD & SUB need 1 clock cycle and MUL need 3 clock cycle. Pipeline do not use any operand forwarding. What is the number of cycle take to comple the following sequence of instruction

ADD, R_2, R_1, R_0
MUL R_4, R_3, R_2
SUB, R_6, R_5, R_4

Ans



∴ Total clock cycle required is 8.

Q10) What CPI? Consider a 5-stage pipeline which is executing a program of 100 instructions. Among all instructions 30 cause 3 stall cycles each.

- Calculate CPI of Pipeline
- If pipeline cycle time is 3ns then what is avg. Ins. execution time
- Cal. CPI of pipeline in ideal cond^n with hazard?
- If pipeline cycle time is 3ns then what is avg Inst. execution time in ideal cond^n.

Ans CPI stands for Cycle per Instruction

Given: $k = 5$ $n = 100$ $x = 30 (10 * 3)$

i) $CPI = \frac{k + n - 1}{n} = \frac{5 + 100 - 1}{100} = \frac{104}{100} = 1.04$

ii) Given: $t_p = 3ns$

$CPI_{avg} = \frac{k + n - 1 + x}{n} = \frac{5 + 100 - 1 + 30}{100}$

$= \frac{134}{100} = 1.34$

Avg Exe. tym $= CPI_{avg} * t_p = 1.34 * 3$

$= 4.02$ ns

Ans) $CPI_{Ideal} = \frac{m+n}{n} = \frac{100+30}{100} = \frac{130}{100}$ s/i

Ans) Avg Ins. tym in ideal cond^n $= CPI_{Ideal} * tp$

$= 1.3 * 3$ ns

$= 3.9$ ns

Q.11) What is non-linear pipeline, reservation table, permissible latency, forbidden latency, collision vector, state diagram, simple cycle & greedy cycle?

Ans) Non linear Pipeline :-

→ Non linear Pipeline are dynamic pipeline because they can be reconfigured to perform valuable func^n at diff times.

→ Non linear Pipeline allows read forward & read back connections in add^n to stream line connections.

Reservation Table :-

→ It displays the time space flow of data through the pipeline for one func^n evaluation.

→ Check mark in each row indicates stage repition.

→ Check mark in each column indicates multiple stage usage at a same cycle time.

Permissible latency :-
Latency that will not cause collision.

Forbidden latency :
Latency that will cause collision.

Collission vector :- It is the representation of forbidden latency & permissible latency together.

Simple cycle : A simple cycle is a latency cycle in which each state appears only once.

Greed cycle :- A simple cycle whose edges are all made with min. latencies from their respective starting states.

Q12) Consider the following reservation table:

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

Now calculate permissible latency, forbidden latency, collision vector, state diagram, simple cycle & greedy cycle.

Ans Permissible latency = 1, 3, 6, 8
forbidden latency = 2, 4, 5, 7

Collision vector C_i = 0 1 0 1 1 0 1 0
[forbidden = 1, permissible = 0]

State diagram :- (0 1 0 1 1 0 1 0)
$C_8 C_7 C_6 C_5 C_4 C_3 C_2 C_1$

i) C_1 : 0 1 0 1 1 0 1 0 } OR
shift right by 1) 0 0 1 0 1 1 0 }
0 1 1 1 1 1 1 1 → diff. : its a state

ii) C_3 : Here we need to shift right 3 times as subscript is 3.
C_i
↳ 0 1 0 1 1 0 1 0
shifted 3 time → 0 0 0 0 1 0 1 1
0 1 0 1 1 0 1 1 } → new state

Scanned with CamScanner

(iii) C_6 : we need to shift three right 6 times

C^0 0 1 0 1 1 0 1 0

0 0 0 0 0 0 0 1 } or

0 1 0 1 1 0 1 1 → Same as C_3

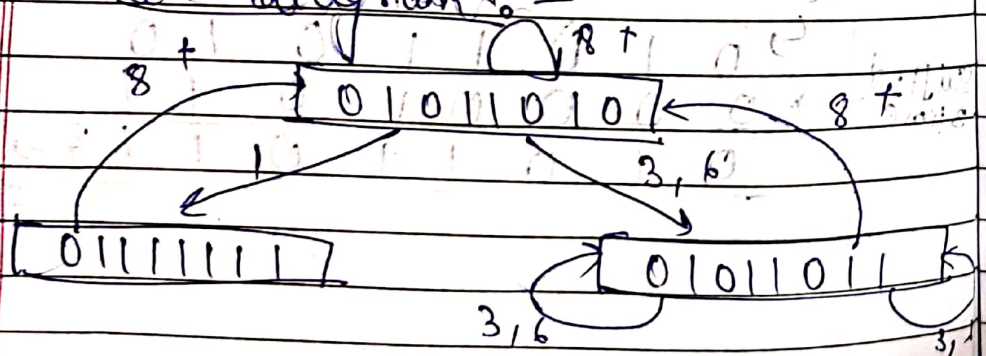
(iv) C_8 : 8 times right shift :

C^0 : 0 1 0 1 1 0 1 0

0 0 0 0 0 0 0 0

0 1 0 1 1 0 1 0 ≡ C^0

∴ State diagram :-



Simple cycle → {1, 8}, {8}, {3, 8}, {6, 8}, {3}, {8}

Greedy cycle → {3}, {1, 8}.