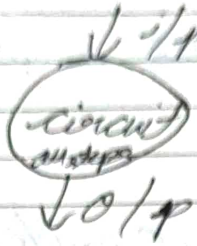
Date
Page No.

⇒ Pipeline :-



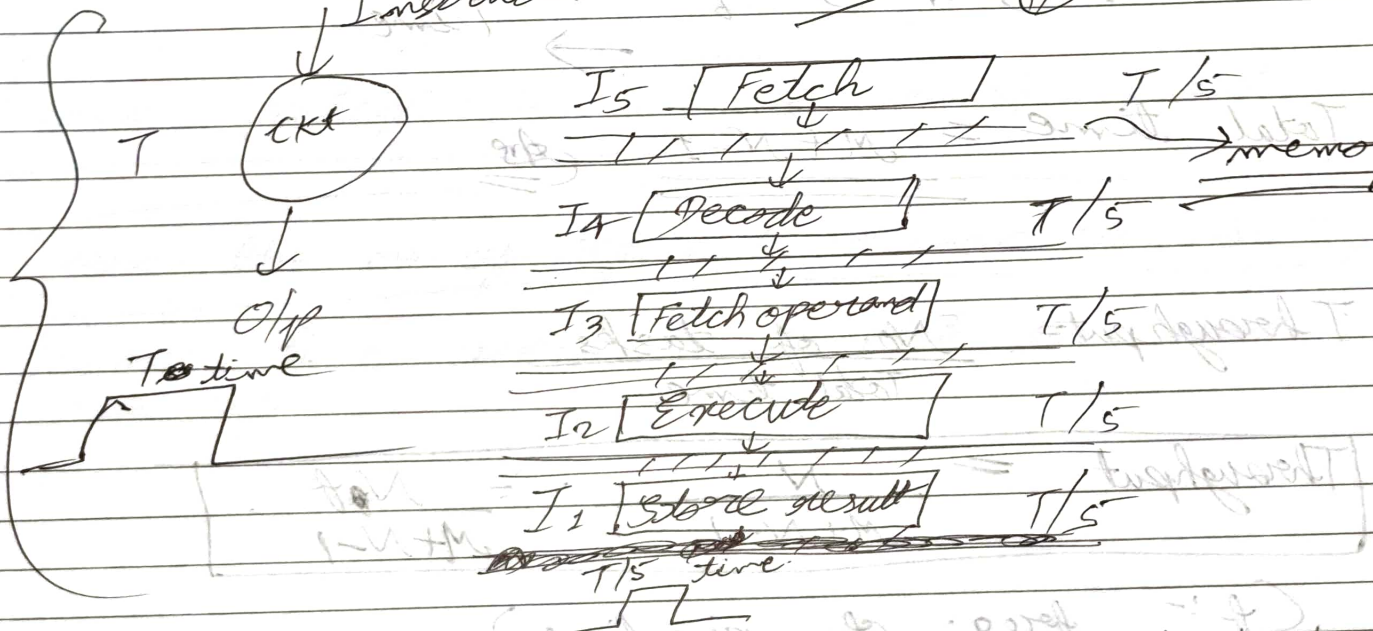
Problem

Data + Data Manipulation

DS + Algorithms → steps of manipulation

~~Code~~ language

Hardware



Date 22/08/22
Page No.

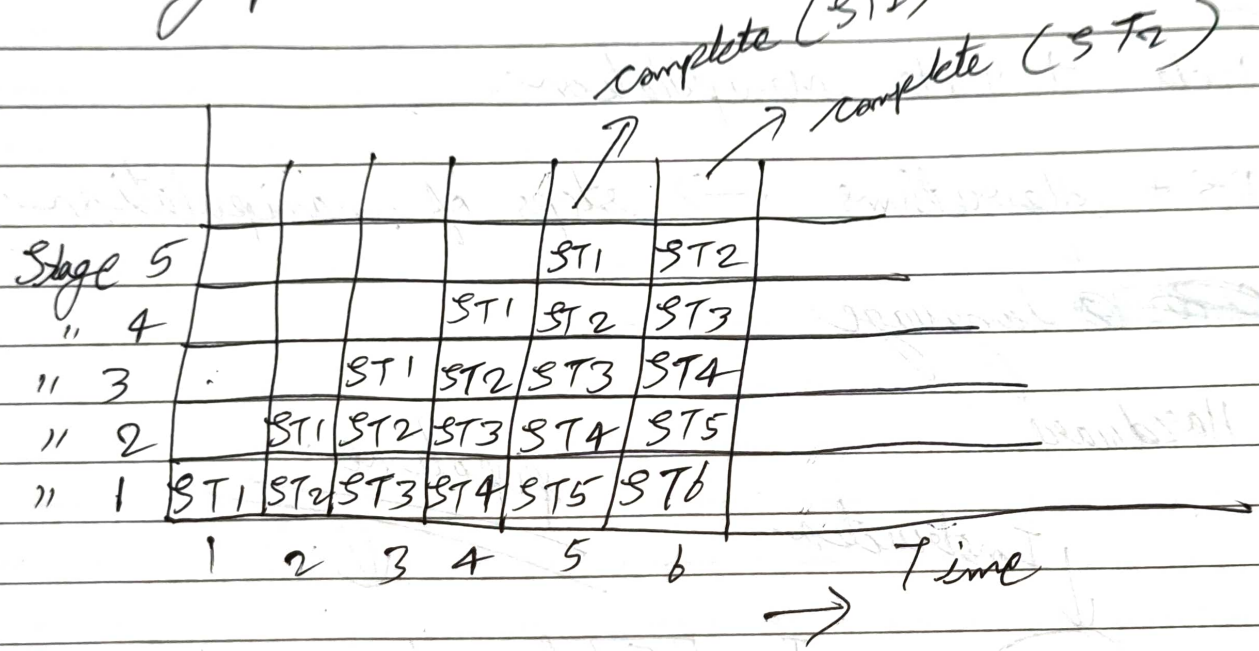
⇒ Pipeline

No. of segments / stages = M

" " tasks = N

Total time taken to complete tasks = 2

Throughput = 2



- Total time = $M+N-1$ Ans

- Throughput = $\frac{\text{No. of tasks}}{\text{Total time}}$

$$\boxed{\text{Throughput} = \frac{N}{M+N-1} \cdot = \frac{N \cdot f}{M+N-1}}$$

(f :- freq. of pipeline)

Date
Page No.

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

Max. Throughput = f ⇒ frequency.

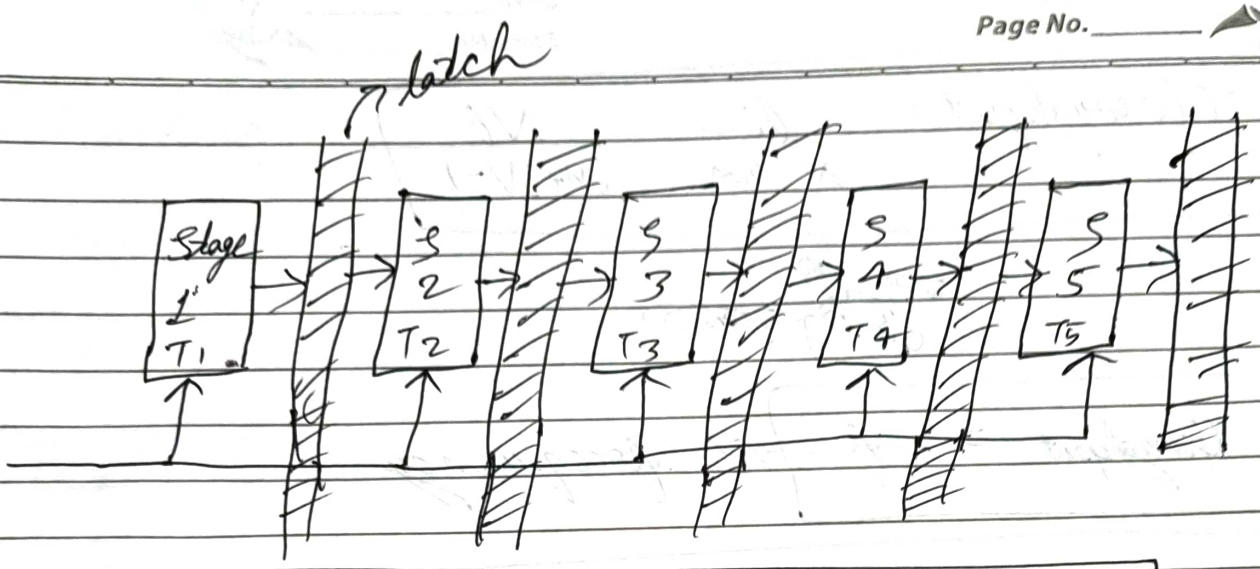
Speed up = $\frac{\text{Time without pipeline}}{\text{Time with pipeline}}$

Speed up = $\frac{MN \text{ clock cycles}}{M+N-1}$

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

Max. Speedup = M ⇒ No. of segment

Max. the no. of segments → max. the freq. because time taken to complete one stage decreases.



$$f = \frac{1}{\text{latch time} + \max(T_1, T_2, \cdots T_5)}$$

time to put & fetch data from latch (memory)

$$\Rightarrow \boxed{PCR = \text{Performance} \cdot \cancel{\text{Rate}} \text{ Cost Rates} = \frac{\text{Performance}}{\text{Cost}}}$$

Date 23/08/22
Page No.

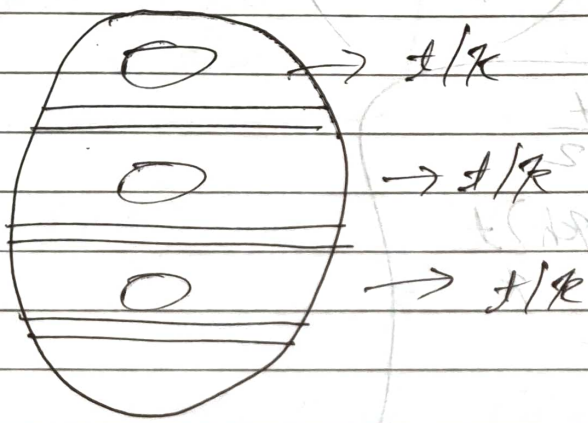
c = cost of digital gates

h = latch cost

k = no. of stages

$$PCR = \frac{f}{c + kh} = \frac{1}{(c+kh)(t/k+d)}$$

t = time taken by H/w without pipeline



d = latch time

$$\text{Time Period} = \frac{t}{k} + d$$

$$\text{freq} = \frac{1}{d + (t/k)}$$

$$PCR = \frac{1}{(c+kh)(t/k+d)}$$

Date
Page No.

$$PCR = \frac{1}{(c+kh)(t/k+d)}$$

$$\frac{d}{dk}(PCR) = 0$$

$$\frac{(-1)(h)}{(t/k+d)(c+kh)^2} = 0$$

$$k_0 = \sqrt{\frac{tc}{dh}} \qquad k_0^2 = \frac{tc}{dh}$$

$$\frac{-1}{(c+kh)^2(t/k+d)^2} \times \left[(c+kh)\left(-\frac{t}{k^2}\right) + (t/k+d)(h)\right] = 0$$

$$\left(\frac{t}{k}+d\right)h = (c+kh)\frac{t}{k^2}$$

$$(t+dk)h = (c+kh)\frac{t}{k}$$

$$k(t+dk)h = (c+kh)t$$

$$\frac{k(t+dk)}{(c+kh)} = t$$

$$kt + dk^2h = ct + kht$$

$$dk^2h + (t-ht)k - ct = 0$$

$$k = \frac{-(t-ht) \pm \sqrt{(t-ht)^2 + 4dhct}}{2dh}$$

Date
Page No.

$$\Rightarrow \frac{-1}{(c+kh)^2(t/k+d)^2}\left[(c+kh)\left(\frac{-t}{k^2}\right)+(t/k+d)(h)\right]=0$$

$$\Rightarrow \left(\frac{t}{k}+d\right)h = \frac{t}{k^2}(c+kh)$$

$$\frac{th}{k}+dh = \frac{tc}{k^2}+\frac{th}{k}$$

$$k^2 = \frac{tc}{dh} \quad \ldots\ldots$$

$$\boxed{k=\sqrt{\frac{tc}{dh}}} \quad \ldots$$

$$\frac{d^2}{dk^2}(PCR) = \frac{d^2}{dk^2}\left[\left(\frac{t}{k}+d\right)h - \frac{t}{k^2}(c+kh)\right]$$

$$= \left(\frac{-t}{k^2}\right)h - \left[(c+kh)\left(\frac{-2t}{k^3}\right)+\frac{t}{k^2}(h)\right]$$

$$(+ve) \quad (-ve) = -ve.$$

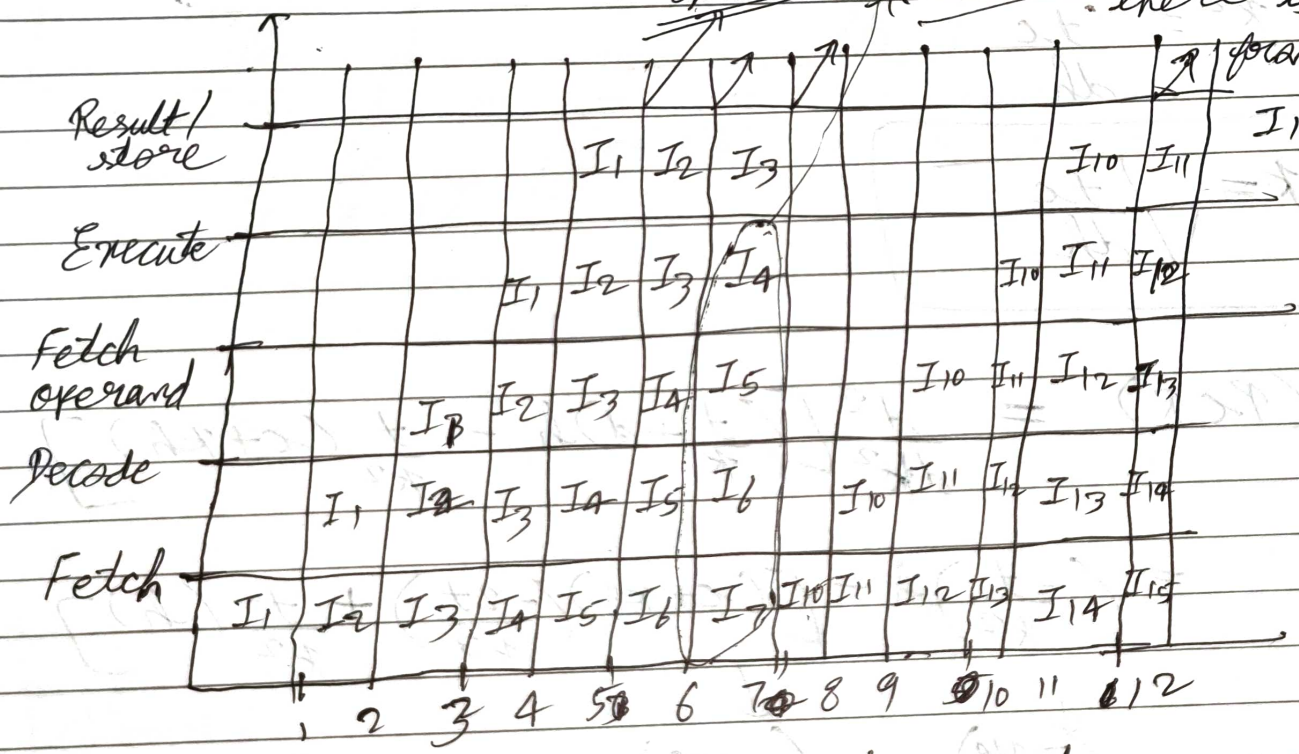
maximum.

Date 24/08/22
Page No.

⇒ Instruction Pipeline :-

No. of instructions = N
" " segment = M

Probability that a given instruction will cause branching = s.



Date
Page No.

- No of extra clock due to 1 branch construction = $(M-1)$
- Avg. no. of branching = $\sum_{i=1}^{N} 1 \cdot s + 0 \cdot (1-s)$

$= Ns$

$$\boxed{\text{Time} = M+N-1 + Ns(m-1)} \text{ clock cycles.}$$

$$\boxed{\text{Speed up} = \frac{MN}{M+N-1+Ns(m-1)}}$$

$$\boxed{\text{Throughput} = \frac{Nf}{M+N-1+Ns(m-1)}}$$

$$\text{Max. speed up} = \lim_{N \to \infty} \frac{MN}{M+N-1+Ns(M-1)}$$

$$\boxed{\text{Max. speed up} = \frac{M}{1+s(M-1)}}$$

$$\boxed{\text{Max. Throughput} = \frac{f}{1+s(M-1)}}$$

DELTA Notebook

Date
Page No.

Q:- $M = 5$, $f = 10\ MHz$

s	max speed up	Throughput
0	5	10×10^6
0.1	3.5	7.14×10^6
0.2	2.8	5.55×10^6
0.3	2.27	4.54×10^6

⇒ 1 Instruction Cycle (IC) = M clock cycles.

Avg. no. of Instructions executed per IC $= \dfrac{N}{\frac{M+N-1+NS(M-1)}{M}}$

$= \dfrac{NM}{M+N-1+NS(M-1)}$

Date 6/09/22
Page No.

⇒ N = No. of instructions

Probability that a given inst^{n} is unconditional Branch inst^{n} = P

" " " " " " conditional " " = Q

" " a conditional branch will cause branching = R

No. of segment of pipeline = M

Frequency of pipeline = f

i) Total time taken to execute N instruction = ?

ii) Throughput = ?

iii) Speed up = ?

iv) Avg. No. of inst^{n} executed per I-C = ?

Avg. No. of branching = NP + NQR

Total time = $M + N - 1 + (NP + NQR)(M-1)$ clock cy

$$\text{Throughput} = \frac{N \cdot f}{M + N - 1 + N(P + QR)(M-1)}$$

$$\text{Max. throughput} = \frac{f}{1 + (P + QR)(M-1)}$$

DELTA Notebook

Date
Page No.

$$\text{Speedup} = \frac{NM}{(M+N-1) + N(P+QR)(M-1)}$$

$$\text{Avg. No. of inst}^{m} \text{ exec. per TC} = \frac{NM}{(M+N-1) + N(P+QR)(M-1)}$$

$$\text{Max. speed up} = \frac{M}{1 + (P+QR)(M-1)}$$

⇒ ~~Branch~~

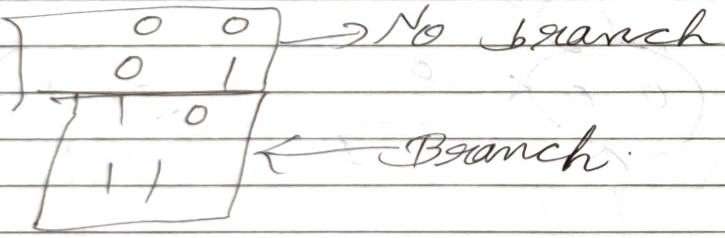
Date
Page No.

⇒ Branch Prediction :-

- Branch History Table :- (BHT) :-

1-bit Branch predictor

- 2- Bit Branch :-



- m- bit :-

$0 - 2^{m-1} - 1$ → No branching

$2^{m-1} - 2^m - 1$ → Branching

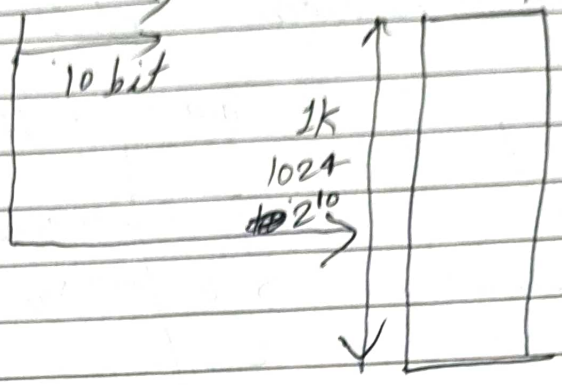
Date 7/9/22
Page No.

Address ← Instruction → (8k-32k)

PC ← → BHT

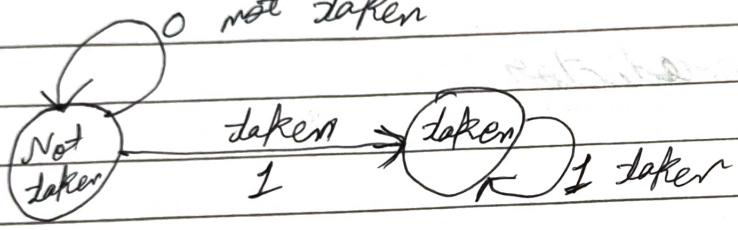
CS:IP

↳ PC in 8086

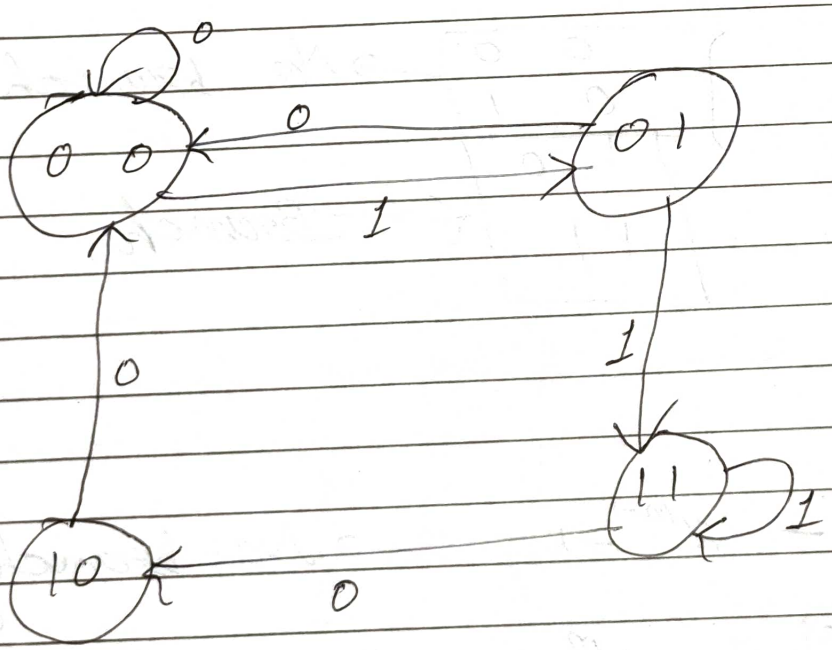


8 GB
2^{33}

1-bit :-



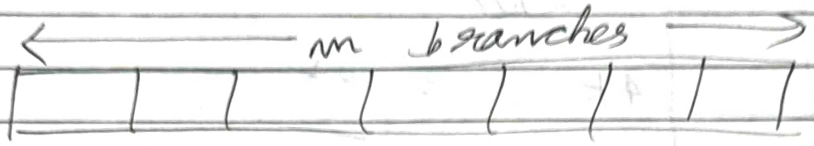
2-bit :-



Date
Page No.

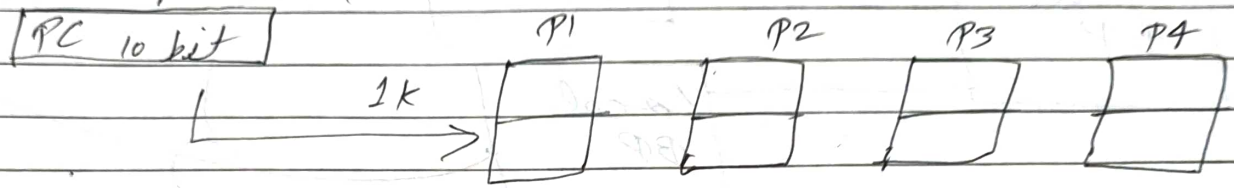
→ correlating branch predictor

⇒ (m, n) branch predictor :-



GHT

(2,2) branch predictor



GHT

(2^2 → predictors)

Q1:- How many bits are in (0, 2) branch predictor with 4k entries.

Q2:- How many entries are in (2, 2) branch predictor with same no. of bits.

Ans 1:- 2^0 ⇒ 1 predictor

~~4k = $2^2 \times 2^0$ = 12 bits.~~

bit in 1 predictor = 2

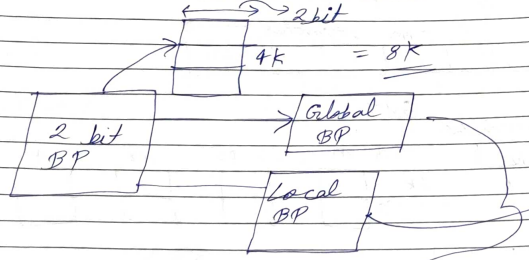
$2 \times 4k = 8k$ bits. Ans.

No. of entries $\times 2^2 \times 2 = 8k$

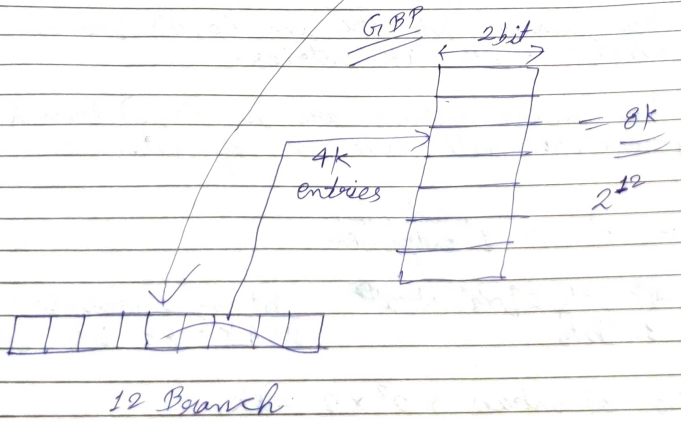
No. of entries = $\frac{8k}{2^3} = 1k$. Ans.

Date 12/09/22
Page No.

⇒ Tournament Branch Predictor :-

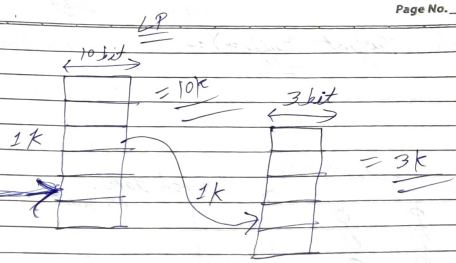


* Alpha 21264



Date
Page No.

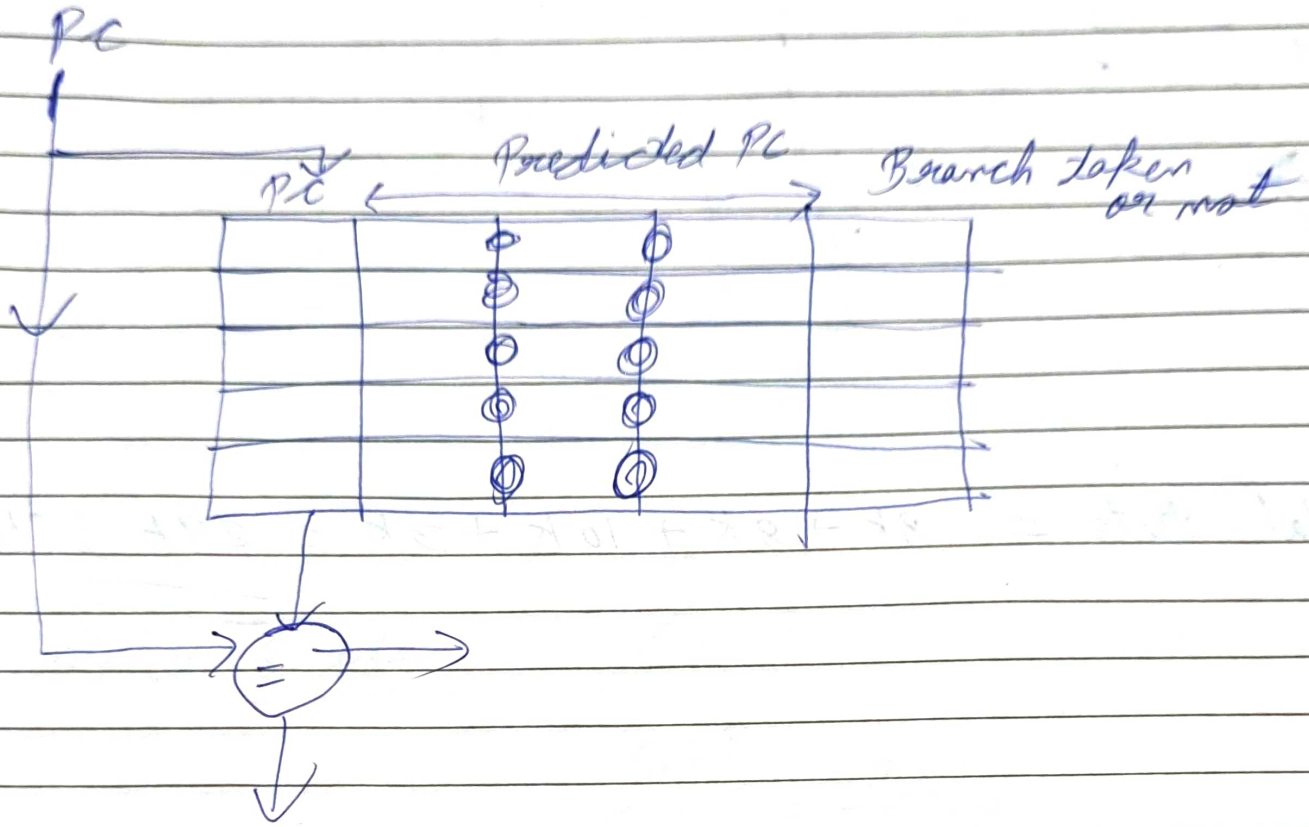
LP



Total Bits = 8k + 8k + 10k + 3k = 29k bits

Date
Page No.

⇒ BTB (Branch Target Buffer) :-



Ques :- Determine the total branch penalty for a branch ~~buffer~~ target buffer assuming the penalty cycle for individual misprediction from the following table.

0.18
0.2
0.38

Date
Page No.

Ans:-

Instruction in Buffer	Prediction	Actual prediction	Penalty cycle.
Yes	Taken	Taken	0
Yes	Taken	Not Taken	2
No		Taken	2
No		Not taken	0

- Prediction Accuracy = 90% (for instruction in buffer).

Hit rate = 90% (for branches predicted taken) in buffer

Ans:- Branch Penalty = (BTB hit but has wrong prediction) X 2

+ (Miss in BTB but actually taken) X2

= 0.90 X 0.1 X 2 + 0.1 X 2

= 0.38 Ans.

Date
Page No.

⇒ Collision force scheduling for y :-

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

⇒ Forbidden latencies, Permissible latency

Forbidden latencies = 2, 4

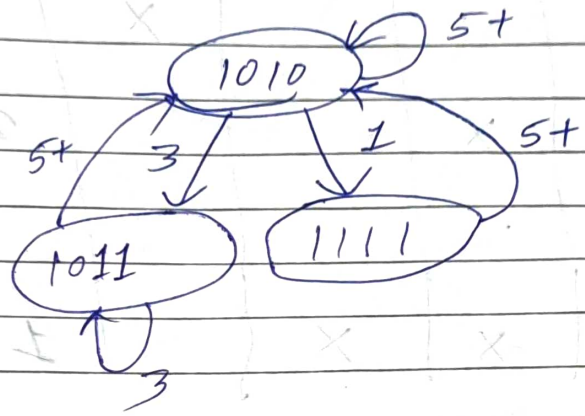
Permissible latencies = 1, 3, 5, 6+

⇒ Initial collision vector :-

1010

Date
Page No.

⇒ State Transition Diagram :-



1010
+ 0001
1011

1010
or 0101
1111

⇒ Simple cycles :-

(5), (3,5), (3), (1,5)

⇒ Greedy cycle :- (cycle containing minimal outgoing edges from each state).

(3), (1,5).

⇒ MAL :- 3

Date
Page No.

2)	1	2	3	4	5	6	
S1	X					X	5
S2		X		X			2
S3			X				
S4				X	X		1

⇒ Forbidden latencies = (1, 2, 5)

⇒ Permissible latencies = (3, 4, 6+)

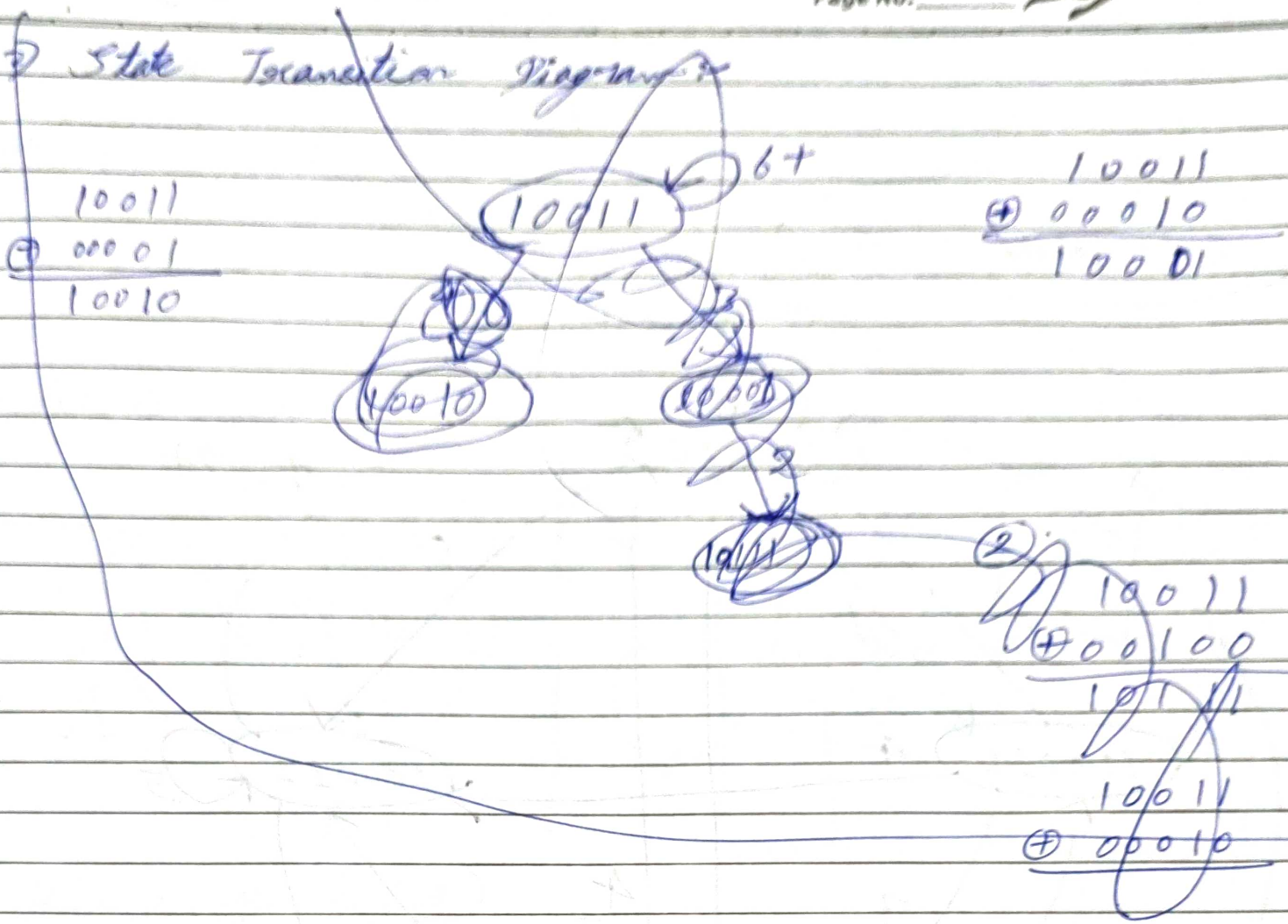
⇒ Initial collision vector :-

10011

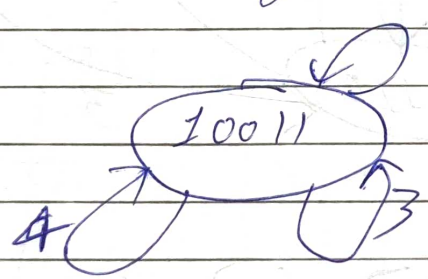
3)

Date
Page No.

⇒ State Transition Diagram :-



⇒ State Transition Diagram :-



10011
or 00010
10011

⇒ Simple Cycle :- (3), (4), (6)

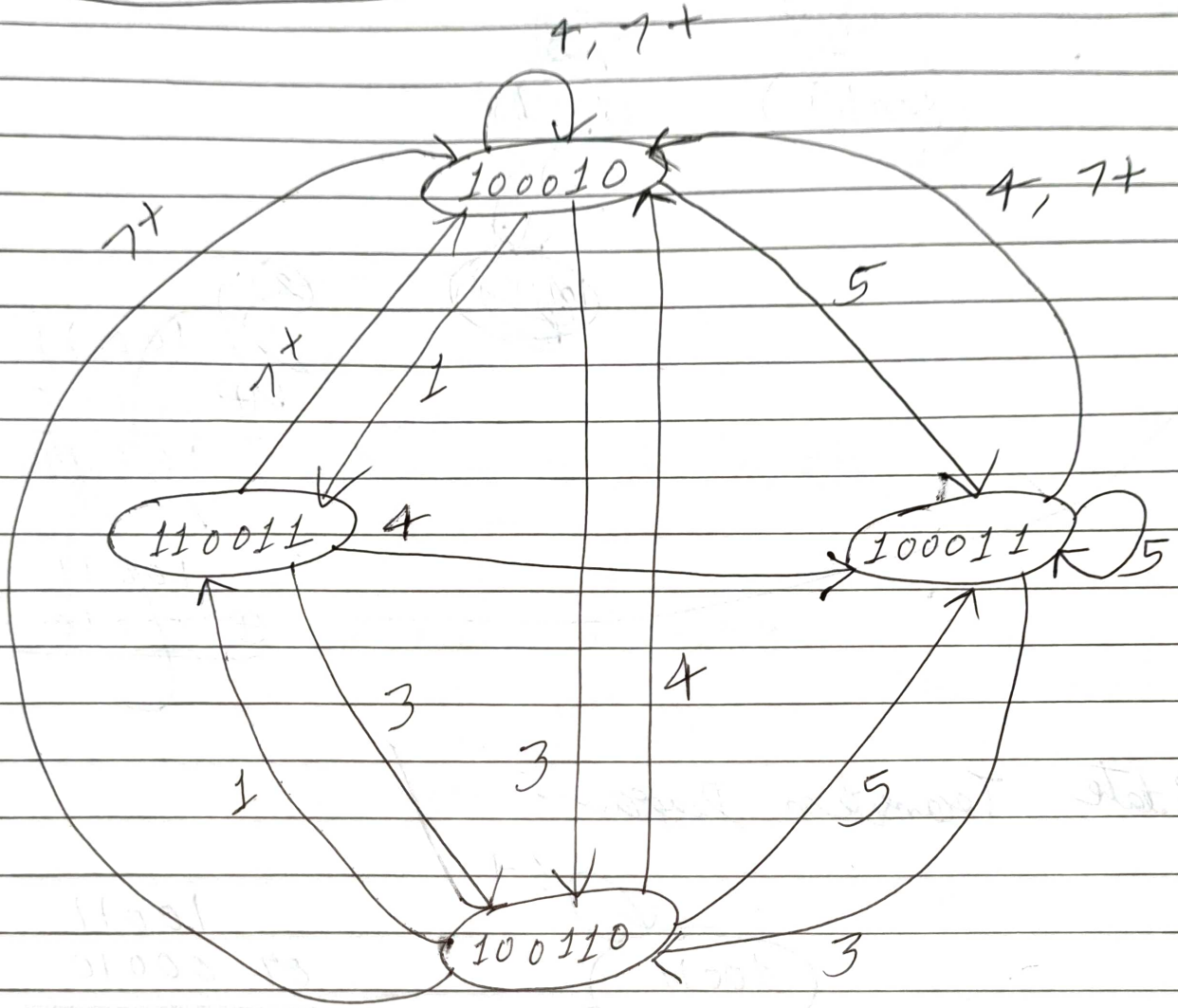
⇒ Greedy Cycle :- (3)

⇒ MAL :- (3) Ans

Date
Page No.

100010
000110
100110

2 <= MAL <= 4



⇒ Simple cycles = (4), (7), (4, 5), (5), (5, 7), (3, 4), (3, 5), (1, 3), (1, 7), (1, 3, 7), (1, 3, 5, 4)

⇒ Greedy cycle = (1, 3)

⇒ MAL = $\frac{1+3}{2}$ = 2. Ans

Date
Page No.

⇒ ~~Latency cycle (3)~~

⇒ Throughput = $\frac{1}{MAL}$

= $\frac{1}{2}$ = 0.5

⇒ latency cycle = 3

latency cycle (3)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S1	1			2		1	3	1	2	4	2	3	5	3	4	6	4	5		5	6
S2		1		1	2		2	3		3	4		4	5		5	6		6		
S3			1		1	2	1	2	3	2	3	4	3	4	5	4	5	6	5	6	6

efficiency = $\frac{8}{9}$ × 100 = 88.9 %. Ans

(8 → 0.88)

(1,8) cycle.

Date _____
Page No. _____

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S_1	1					1		1	2	3				2	3	2	3	4					4
S_2		1		1						2	3	2	3										
S_3			1		1		1				2	3	2	3	2	3							

Date 28/10/22
Page No.

⇒ Performance laws :-

Q:- A uni-processor computer can operate in either scaler or vector mode. In vector mode computation can be done 9 times faster than scaler mode. A certain benchmark program took time T to run on this computer. Further, it was found that 25% of T was attributed to vector mode. In the remaining time machine operated in scaler mode.

i) Calculate the effective speedup under the above cond^n as compared with cond^n when the vector mode is not used at all. Also calculate α (the % of code that has been vectorised) in the above program.

Ans i) speedup = $\frac{\text{time without enhancement}}{\text{time with enhancement}}$

$= \frac{0.75\,T + 0.25\,T \times 9}{T}$

$= 0.75 + 2.25$

$\boxed{\text{speedup} = 3.00}$ Ans

4

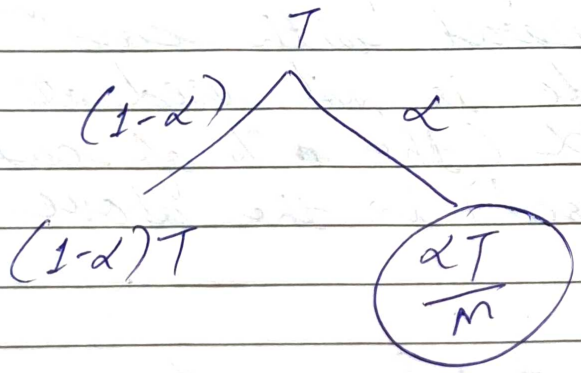
Date
Page No.

=> $\alpha = \frac{0.25T \times 9^3}{3T}$

$= 0.75$

$\boxed{\alpha = 75\%}$ Ans.

=> | 1-α Series | α Parallel |



$\boxed{\text{Time enhancement} = (1-\alpha)T + \frac{\alpha T}{n}}$

=> $\text{speedup} = \frac{T \text{ without enhancement}}{T \text{ with enhancement}}$

$\boxed{\text{speedup} = \frac{T}{(1-\alpha)T + \frac{\alpha T}{n}} = \frac{1}{(1-\alpha) + \frac{\alpha}{n}}}$

↳ Amdahl's law

Date
Page No.

$$\Rightarrow \quad 3 = \frac{1}{1-\alpha + \frac{\alpha}{9}}$$

$$3\left(1-\alpha+\frac{\alpha}{9}\right) = 1$$

$$3 - 3\alpha + \frac{\alpha}{3} = 1$$

$$\alpha = \frac{4}{3}\alpha$$

$$\boxed{\alpha = \frac{3}{4}}$$

enhancement 1	α_1	S_1
enhancement 2	α_2	S_2
⋮		
enhancement n	α_n	S_3

$$\text{overall speedup} = \frac{1}{1 - (\alpha_1 + \alpha_2 + \cdots \alpha_n) + \frac{\alpha_1}{S_1} + \frac{\alpha_2 + \cdots \alpha_n}{S_2 \quad S_n}}$$

↳ Generalized Amdahl's law.

Date
Page No.

reduced execution
⇒ Fraction of ^ time when no enhancement is used =

$$\frac{1-(\alpha_1+\alpha_2+\cdots\alpha_m)}{1-(\alpha_1+\alpha_2+\cdots\alpha_m)+\frac{\alpha_1}{s_1}+\frac{\alpha_2}{s_2}+\cdots+\frac{\alpha_m}{s_m}}$$

ii) Suppose we double the speed ratio b/w the vector mode and the scalar mode by hardware improvement. Cal the effective speedup that can be achieved.

iii) Suppose the same speedup obtained in part (ii) must be obtained by compiler improvement instead of hardware improvement. What could be new vectorization ratio α that should be supported by the vectorizing compiler to achieve the same enhancement.

ii) $$\text{speedup} = \frac{1}{1-\alpha+\frac{\alpha}{s}}$$

$$= \frac{1}{1-0.75+\frac{0.75}{18}}$$

speedup = 3.43

Date
Page No.

(iii) $3.43 = \frac{1}{1-\alpha+\frac{\alpha}{9}}$

$1-\alpha+\frac{\alpha}{9} = \frac{1}{3.43}$

$1-\frac{8\alpha}{9} = \frac{1}{3.43}$

$1-\frac{1}{3.43} = \frac{8\alpha}{9}$

$1-0.2915 = \frac{8\alpha}{9}$

$0.7084 = \frac{8\alpha}{9}$

$\boxed{\alpha = 0.79}$ Ans ⟹

Ques:- Let α be the fraction of program code that can be executed simultaneously by n processors in a computer system.

Assume that remaining code must be executed sequentially by a single processor.

Each processor has an execution rate of x MIPS and all processors are assumed to be equally capable.

Derive an expression for effective MIPS rate.

Date
Page No.

Ans:- let no. of instructions to be executed = k millions

	Parallel Mode	sequential Mode
MM →	αk	$(1-\alpha)k$

$$T = \frac{\alpha k}{mr} + \frac{(1-\alpha)k}{r}$$

$$\boxed{T = \frac{k(\alpha + (1-\alpha)m)}{mr}}$$

$$\Rightarrow \text{Effective MIPS} = \frac{k}{\frac{k[\alpha + (1-\alpha)m]}{mr}}$$

$$\boxed{\text{effective MIPS} = \frac{mr}{\alpha + (1-\alpha)m}}$$

Eg. 3.4

Date
Page No.

Ques 1. Consider a comp. which can execute a program in 2 ~~operational modes~~ mode regular mode and enhanced mode with probability distributions $\{\alpha, 1-\alpha\}$ respectively If α varies b/w a & b.

$$0 \leq a < b \leq 1$$

Derive an expression for average speedup factor using harmonic mean concept.

Date 7/11/22
Page No.

Q => Program :-

Unbalanced

PAR → parallel

```
for (L=1; L<=32; L++)
{
    for (i=(L-1)*32+1; i<=L*32; i++)
    {
        sum[i] = 0;
        for (j=1; j<=i; j++){
            sum[i] = sum[i] + i
        }
    }
}
```

Balanced :-

These 2 will run parallely.

```
PAR for (L=1; L<=32; L++){
    for (i=(L-1)*16+1; i<=L*16; i++){
        sum[i] = 0;
        for (j=1; j<=i; j++){
            sum[i] = sum[i]+1;
        }
    }
```

It'll run parallely with 1st box.

```
    for (i=(64-L)*16+1; i<=(64-L+1)*16; i++){
        sum[i] = 0;
        for (j=1; j<=i; j++){
            sum[i] = sum[i]+1;
        }
    }
}
```

Date
Page No.

⇒ MPI (Message Passing Interface):-

```
main (int argc, char *argv[]){
    int myrank, size;
    MPI_Init (&argc, &argv);
    .
    .
    .
    MPI_Finalize ();
}
```

~~MPI_Comm_rank (MPI_Comm ...~~

```
MPI_Comm_size (MPI_COMM_WORLD, &size);
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
```

MINHAAL

Date 9/11/22
Page No.

MPI_Send (buf, count, datatype, dest, tag, communicator);

MPI_Recv (buf, count, datatype, src, tag, communicator, status);

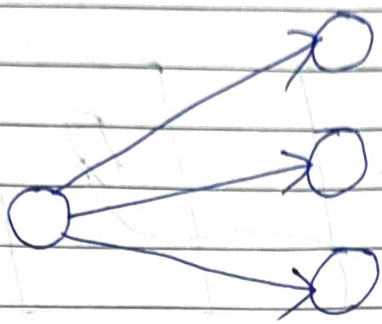
```
int x, y, myrank, msgtag = 1;
MPI_Status status;
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
if (myrank == 0){
    MPI_Send (&x, 1, MPI_INT, 1, msgtag,
              MPI_COMM_WORLD);
}
else if (myrank == 1){
    MPI_Recv (&y, 1, MPI_INT, 0, msgtag,
              MPI_COMM_WORLD, &status);
}
```

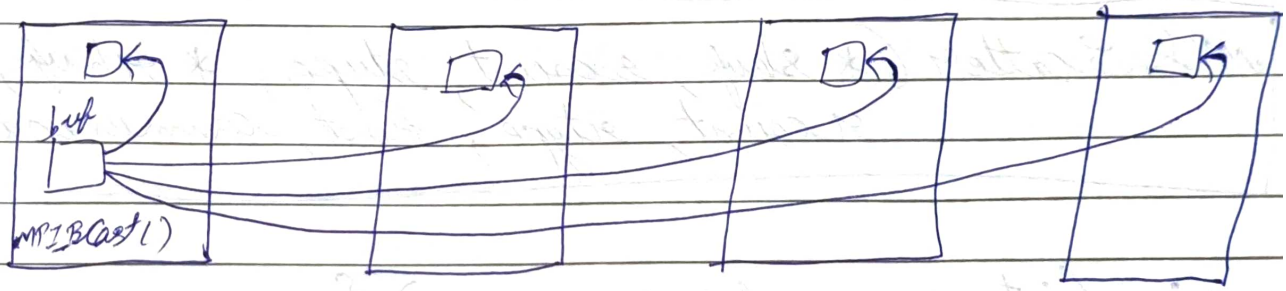
Date
Page No.

```
#include <mpi.h>
main(int argc, char *argv){
    char message[20];
    int i, rank, size, type = 99;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_Size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0){
        strcpy(message, "Hellow World");
        for(i=1; i< size; i++){
            MPI_Send(message, 13, MPI_CHAR, i,
                     type, MPI_COMM_WORLD);
    }
    else{
        MPI_Recv(message, 13, MPI_CHAR, 0, type,
                 MPI_COMM_WORLD, &status);
    }
    printf("Message from process = %d : %s \n", rank,
                                                message);
    MPI_Finalize();
}
```

Date 11/11/22
Page No.

⇒ MPI Broadcast :-



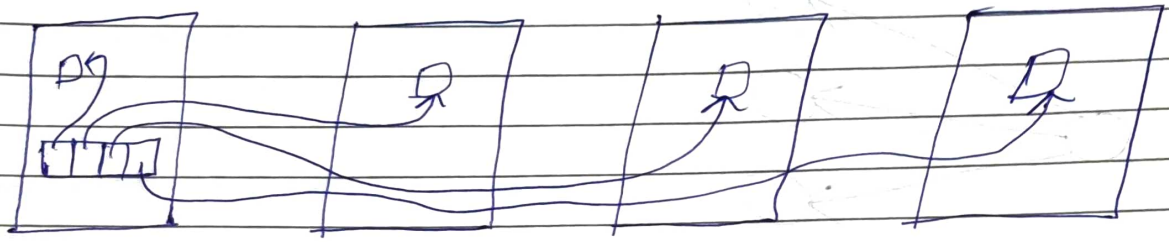



```
MPI_Bcast (*buf, count, datatype, root, communicator)
```

```
double A[N][N];
MPI_Bcast (A, NXN, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Date
Page No.

⇒ MPI_Scatter:-

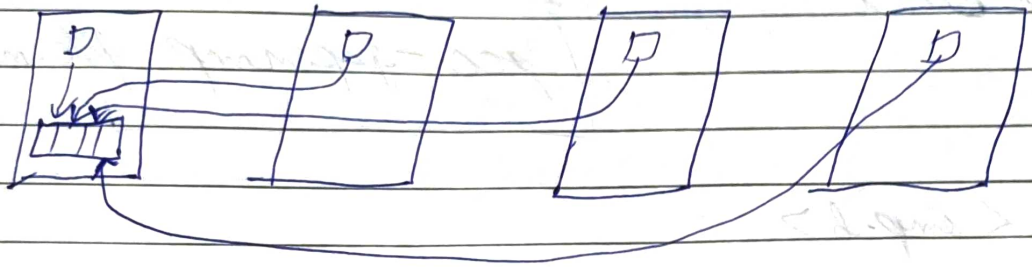


MPI_Scatter (*sbuf, scount, stype, *rbuf, rcount, rtype, root, communicator);

```
main (int argc, char *argv[]) {
    int size, *sendbuf, recvbuf[100];

    MPI_Init (&argc, &argv);

    MPI_Comm_size (MPI_COMM_WORLD, &size);
    sendbuf = (int*) malloc (size*100*sizeof(int));

    MPI_Scatter (sendbuf, 100, MPI_INT, recvbuf, 10,
                 MPI_INT, 0, MPI_COMM_WORLD);

    MPI_Finalize ();
}
```

Date
Page No.

⇒ MPI_Gather :-



MPI_Gather (*sbuf, scount, stype, *rbuf, rcount, rdtype, root, comm);

Program :- change MPI_Scatter to MPI_Gather in previous program.

⇒ MPI_Reduce :-

- MPI_Reduce (*sbuf, *recvbuf, count, datatype, op, root, communicator);

  ↳ MPI_MAX
  MPI_MIN
  MPI_SUM
  MPI_PROD

Date 14/11/22 MINHAAZ
Page No.

⇒ OPenMP

- Thread Based

↗ to compile

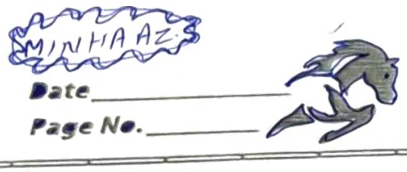
```
gcc -fopenmp file-name
```

```
#include <omp.h>
#pragma omp parallel
{

}
```

```
#include<omp.h>
#include<stdio.h>

int main (int argc, char *argv[]){
  #pragma omp parallel
  {
    printf("Hello Minhaaz from thread id %d of
        %d \n", omp_get_thread_num(),
            omp_get_num_threads());
  }
```

Imp:
• Program of π.

# threadid
it = thread




```
# pragma omp barrier
# pragma omp parallel for reduction (+:sum)
for (k=0; k<100; k++){
    sum = sum + funct(k);
}
```

→ to get final sum

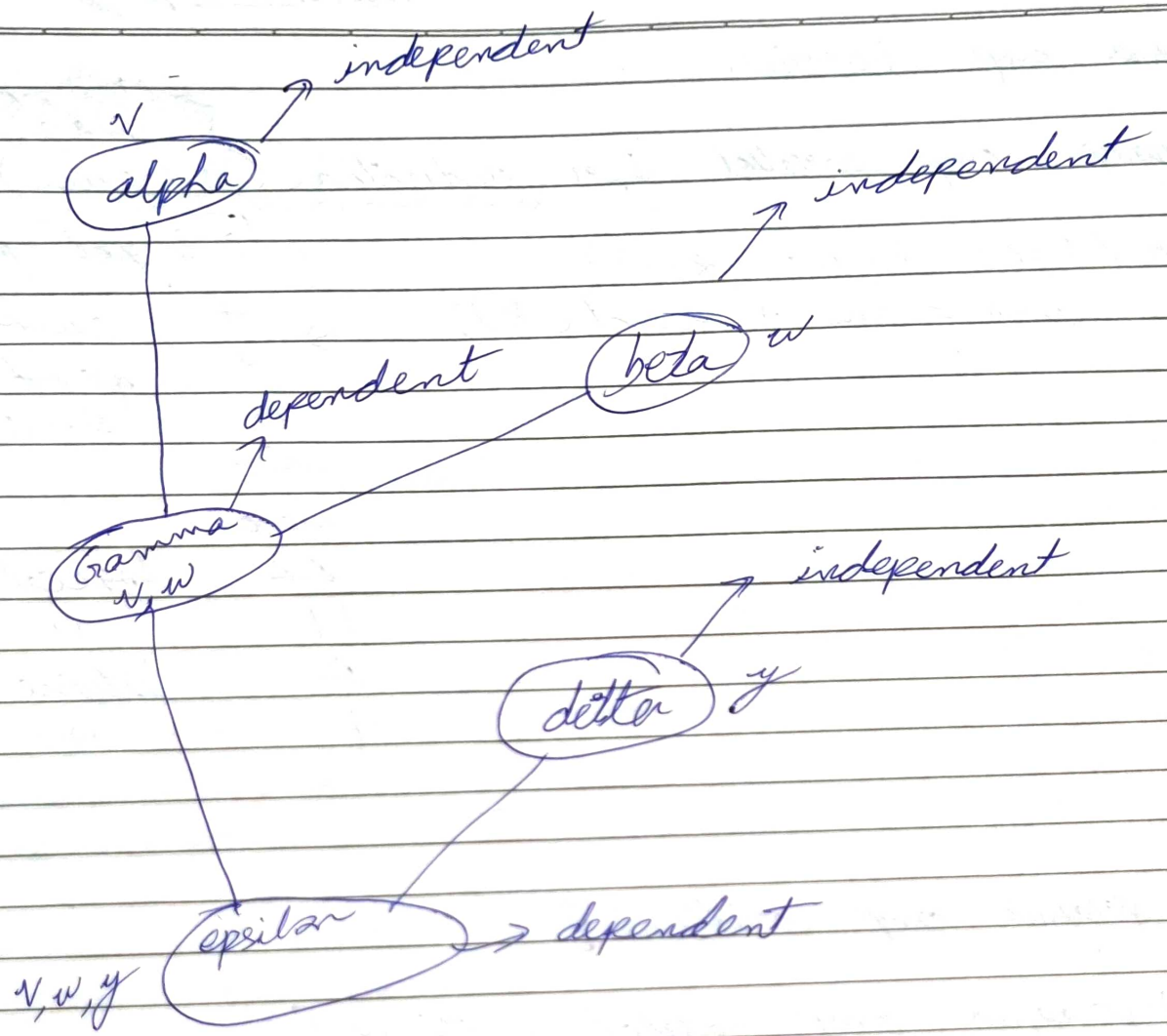
+	sum
*	prod
^	Ex-OR
min	
max	
&&	logical AND
||	" OR
&	bitwise AND
|	" OR

⇒ # pragma omp critical { }

⇒ # pragma omp parallel sections {

```
    # pragma omp section
        v = alpha();

    # pragma omp section
        w = beta();

    # pragma omp section
        y = delta();
}
```

Date _____
Page No. _____


independent
v
alpha
independent
beta
w
dependent
Gamma
v, w
independent
detter
y
epsilon
dependent
v, w, y

Date
Page No.

```
# pragma omp parallel {
    # pragma omp sections {
        # pragma omp section
            v = alpha ();
        # pragma omp section
            w = beta ();
    }
    # pragma omp sections {
        # pragma omp section
            x = gama (v, w)
        # pragma omp section
            y = delta ();
    }
```

2) int omp_get_num_procs (void); → gives no. of physical processors

void omp_set_num_threads (t);

Date
Page No.

⇒ Combination of MPI & openMP (Hybrid Programming):-