"Numericals"

(1) Relative Performance:

X is n times faster than Y.

$$n = \frac{\text{Exec. time of Y}}{\text{Exec. time of X}}$$

OR

$$n = \frac{\text{Performance of X}}{\text{Performance of Y}}$$

(2) Amdahl's law:

(i) $$\text{Speedup} = \frac{\text{Exec. time w/o enhancement}}{\text{" " with "}}$$

For Eg: Exec. time enhanced from 100 sec to 80 sec

Speedup = ?

$$S = 100/80 = 1.25$$

(ii) $$\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Frac}_{\text{enhan}}) + \text{Frac}/\text{speedup}_{\text{enhan}}}$$

Example 1: (pg 47)

Speedup = 10 times.

$Frac_{enhanced} = 40\% = 40/100 = 0.4$

$$Speedup_{overall} = \frac{1}{(1-0.4)+(0.4/10)}$$

Speedup = 1.56.
56%

Example 2: (pg 47)

(i) FP Inst. = 50% = 0.5

$Speedup_{FP}$ = 1.6 times.

(ii) FSQRT = 20% = 0.2

$Speedup_{FSQRT}$ = 10 times.

(i) $$Speedup_{overall} = \frac{1}{(1-0.5)+(0.5/1.6)}$$

S = 1.23. Isko select krna

(ii) $$Speedup_{overall} = \frac{1}{(1-0.2)+(0.2/10)}$$

S = 1.22.

→ Processor Performance Equations.

(3) clock time:

clock time = 1/clock rate.

Rate.

For Eg: Clock time = 1 GHz.

clock cycle time = $1/10^{-9}$ = 1ns

(4) CPU Time:

CPU Time = IC x clock cycle time x CPI.

OR.

$$\text{CPU Time} = \frac{\text{IC} \times \text{CPI}}{\text{clock rate}}$$

For Eg:

No. of Inst. = IC = 10,000

Clock Rate = 2 GHz.

CPI = 1.5.

$$\text{CPU Time} = \frac{10{,}000 \times 1.5}{2 \times 10^{9}}$$

CPU Time = 7.5×10^{-6} sec.

(5) CPI (Clock cycle per Instruction):

$$CPI = \sum_{i=0}^{n} (\text{clock cycles} * \text{Frequency})$$

For Eg:

Inst.	CPI	Freq
Branch	3	12%
Store	4	10%
Other	5	78%

$$CPI = (3 \times 0.12) + (0.1 \times 4) + (0.78 \times 5)$$

$$CPI = 4.66$$

Q: (Mid 1 of 2017) Q2a

$$\text{Speedup}_{\text{enhanced}} = 15$$

$$\text{Speedup}_{\text{overall}} = 2.$$

$$\text{Fraction}_{\text{enhanced}} = ?$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Frac}) + \text{Frac}/\text{Speedup}}$$

(5)

$2 = 1/(1-F) + (F/15)$

$15 - 15F + F = 0.5 \times 15$

$F = 0.5357$

Fraction enhanced = 53.57%.

(6) Pipelining:(

Non pipelined Architecture = IC = 1000.

Speedup = ?

Pipelined = $5 + 999 = 1004$ cycles.

Nonpipelined = $5 \times 1000 = 5000$ "

$$\text{Speedup} = \frac{5000}{1,004} \cong 5.$$

(7) Clock frequency (Pipelined):

IF = 30ns, ID = 25ns, EX = 25.5ns,

Mem = 30ns, WB = 20ns

$$F = \frac{1}{\text{clock cycle time}} = \frac{1}{30 \times 10^{-9}}$$

$F = 0.033$ GHz.

Q. (C-10) Example:

	No. of cycles	Rel. freq.
ALU	4	40%
Branch	4	20%
Mem Operation	5	40%

clock cycle = 1ns. (slowest bh yhi hugi)
Overhead = 0.2ns.
Speedup = ?

Speedup = Avg Inst. time unpipelined / " " " pipelined

Average Inst. unpipelined:
Inst. exec time = clock cycle x Avg CPI.
1ns x [((40% + 20%) x 4) + (40% + 5)]
= 4.4 ns.

Avg Inst. time pipelined:
Slowest speed + overhead
= 1ns + 0.2ns = 1.2ns.

Speedup = 4.4ns / 1.2ns
Speedup = 3.7 times.

(7)

Q: S1 = 25 ns 25+2 = 27
S2 = 15 ns 15+2 = 17
S3 = 30 ns 30+2 = 32
S4 = 15 ns 15+2 = 17
latch Delay = 2ns
clock frequency = ?

Frequency = 1/clock cycle.
= 1/32 ns
F = 0.03125 x 10^9 Hz

Pipeline cycle time = 32 ns (slowest)
Nonpipelined " = (25+15+30+15) = 85 ns.

Speedup = Non pipelined exec. / Pipelined exec.

Speedup = 85/32 = 2.65

(8)

Inst producing result	Inst. using result	latency
ALU	ALU	3
ALU	store	2
Load	ALU	1

(9) Loop Unrolling: Initiation interval = 1

loop:

		No. of cycles before unroll
fld	f0, 0(x1)	1 (2)
fadd.d	f4, f0, f2	3 → stalls
fsd	f4, 0(x1)	6 (4, 5)
addi	x1, x1, -8	7
bne	x1, x2, loop	8 (9)

Total cycles = 9

Unroll 4 times:

Jo loop me dependant nh hain unko aik dafa hi likhen gy.

loop:

```
fld    f0, 0(x1)      → double word ki waja se 8 dec hurha
fadd.d f4, f0, f2
fsd    f4, 0(x1)
fld    f0, -8(x1)
faddd  f4, f0, f2
fsd    f4, -8(x1)
fld    f0, -16(x1)
fadd.d f4, f0, f2
fsd    f4, -16(x1)
fld    f0, -24(x1)
fadd.d f4, f0, f2
fsd    f4, -24(x1)    → 4 x -8
addi   x1, x1, -32    (Independent hai ye)
bne    x1, x2, loop
```

4 times unroll.

(9)

Scheduling and Register renaming:
saasi aik jesi inst. aik sth likh dengy.
↑ aur alg alg register krdengy.

loop:	Fld	F0 , 0(x1)	1
	Fld	F5 , -8 (x1)	2
	Fld	F6 , -16(x1)	3
	Fld	F7 , -24(x1)	4
	Fadd.	F4 , F0 , F2	5
	Fadd	F8 , F5 , F2	6
	Fadd	F9 , F6 , F2	7
	Fadd	F10 , F7 , F2	8
	Fsd	F4 , 0(x1)	9
	Fsd	F8 , -8(x1)	10
	Fsd	F9 , -16(x1)	11
	Fsd	F10 , -24(x1)	12
	addi.	x1 , x1 , -32	13
	bne.	x1 , x2 , loop	14

Total cycles = 14

$\frac{14}{4}$ = 3.5 cycles.

4 times → (4)
unroll

Before Unrolling = 9.
After Unrolling = 3.5.

(10) Tomasulo's Algorithm:

Instructions:

1. ld F6, 32.(R2)
2. ld F2, 44(R3)
3. Mul.d F0, F2, F4 (S1, S2)
4. Sub.d F8, F2, F6
5. Div.d F10, F0, F6
6. Add.d F6, F8, F2.

Instruction status:

Inst.	Issue	Execute	Write R.
L1	✓	✓	✓
L2	✓	✓	
L3	✓		
L4	✓		
L5	✓		
L6	✓		

Jo write result tk complete hugi usko table me show nh krengy par wo reserve rhe gi.
Jese "load 1" ki jaga L1 ki Inst thi.

(11)

→ RS:

Name	Busy	Op (Opcode)	V_j (source reg 1)	V_k (source reg 2)	Q_j (For S1)	Q_k (For S2)	A
load1	No						
load2	Yes	load					44+ Reg[R3]
Add1	Yes	Sub		Mem[32+ Reg[R2]]	load2		
Add2							
Add3							
Mul1	Yes	Mul		Reg[F4]	load2		
Mul2	Yes	Div		Mem[32+ Reg[R2]]	Mul1		

(F2, load2 wali Inst ka dest. hai)

Agr wo source reg kisi uper wale inst ki destination hai tou Q_j aur Q_k me aegi warna V_j aur V_k mein.

Register status:

Field	F0	F2	F4	F6	F8	F10	...
Q_i	Mul1	load2		Add2	Add1	Mul2	

(F4 → Reg)

Jis Inst. F0 me ye destination Reg hai

(11) Scoreboard:

Functional Units:

Two integer units

Two Floating-point add/sub

One Floating " multiply

" " divide.

Instructions:

ld F0, 0(R1)

ld F2, 0(R2)

Mul.d F4, F0, F2.

Add.d F6, F4, F8

S.d F6, 8(R1)

L.d F10, 8(R2)

Mult.d F12, F0, F2.

(12) Direct Mapped Cache:

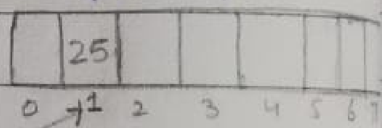
(Block No) MOD (No. of Blocks in Cache)

For E.g Block ko kahan place krengy:

Memory Address = 25.

Mem consists of 32 blocks. (0 - 31)

Cache " " 8 blocks (0 - 7)



25 Mod 8 = (1)

For Eg Memory = 32 blocks = 2^5

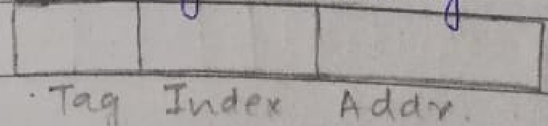
Cache = 8 blocks = 2^3

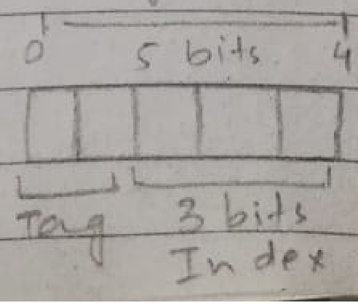
Memory access needs how many bits?

(Depends on block bits)

2^5 → (Need 5 bits)

For acessing Memory





→ ~~Depends krega~~ cache par $2^3 = 8$ Tou 3 bits hun gy

(15)

Directory

000	00
001	11
010	
011	
100	
101	
110	
111	

For Eg Address 25 is in cache or not?
cache hit or cache miss

Index bits

25 = 11001
Tag Index

Index 001 par 11 hai tw cache hit.

→ Valid bit in directory:

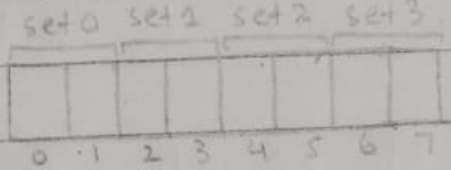
25 = 11001

Index	Tag	VB (Valid Bit)	
000			
001	11	1	→ hit
010			
011			
100			
101			
110			
111			

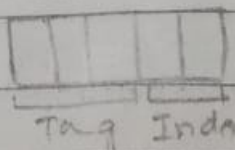
Agr VB 1 hai tw tag bit match krengy warna direct miss hai.

(13) Set Associative

(Block No) MOD (No. of sets in Cache)



2 blocks per set
and we have
8 blocks. (4 sets)
32 bits main Mem = 2^5 5 bits.
4 sets = 2^2 → Index bits



25 = 11001 → set 1 me block 2 aur 3 me se kahin rkhdeta.

4) Opteron Data Cache :-

Size of data Cache = 64 K bytes
block size = 64 byte
2 way Associative
48 bit Virtual Addr → 40 bit physical

Tag bits = ?
Index bits = ?
Block offset = ?

Physical Addr = Tag + Index + Block offset

Sol:

Cache size = 64 K bytes = $2^6 \cdot 2^{10} = 2^{16}$
Block Size = 64 = 2^6

No. of Blocks = $\frac{\text{Cache size}}{\text{block size}} = \frac{2^6 \cdot 2^{10}}{2^6} = 2^{10}$

No. of sets = $\frac{\text{No. of blocks}}{\text{blocks per set}} = \frac{2^{10}}{2} = 2^9$ (2 → Two way Associative)

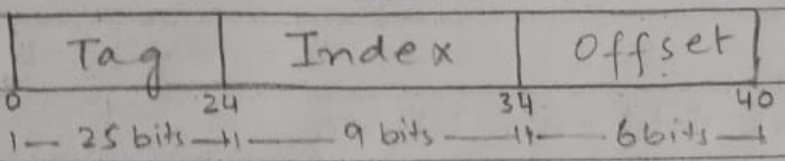
Block offset = block size = 2^6
6 bits. block offset.

(17)

Index bits = No. of sets = 2^9 = 9 bits

Tag bits = Physical Addr - Index - Block offset

40 - 6 - 9 = 25 bits.



Q: (Final paper 2017) Q5(b)

2 way Associative.

Tag	Index	offset
26 bits	12 bits	6 bits

(i) Size of Main Memory = Physical Addr

Tag + Index + offset = 26 + 12 + 6 = 44.

Size = 2^{44} bytes

Cache block size = 2^6 (offset wali)

No. of Blocks = No. of sets x blocks per set = $2^{12} \times 2$ (Index bits, 2 way Associative)

(ii) Size of Cache Memory = ?

No. of blocks = ?

Cache Size = No. of Blocks x block size.

$2^{13} \times 2^6$

Cache Size = 2^{19}.

(18)

(iii) Direct Mapped

25 bits	13 bits	6 bits
Tag	2 Way Associative ki wajah se ek bit add hogi	Offset

(iv) Block Address = 10,000

→ 2 Way Associative:

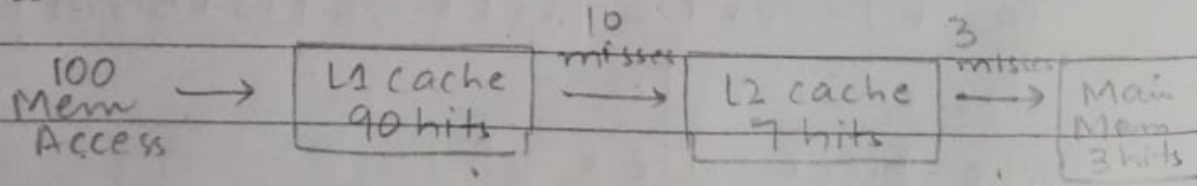
$10,000 \cdot \text{mod } 2^{12} = 1808$

→ Direct Mapped

$10,000 \text{ mod } 2^{13} = 1808$

(15) local and Global optimization:

For E.g



	local Miss rate	Global Miss rate
L1 cache	10/100	10/100
L2 cache	3/10	3/100
Main Mem	0/3	0/100