Division :- Restoring Division.

M → Divisor.
Q → Dividend.

(i) Load 2' Comp. of Divisor in M.
(ii) Load Dividend into AQ registers.
Dividend should be 2n bit +ve no.
(iii) Shift AQ left 1 bit Position.
(iv) Assign A ← A-M.
(v) a] If result is non-negative:-
$Q_0 \leftarrow 1$.
b] If result is negative:-
$Q_0 \leftarrow 0$. and restore previous values of A.
(vi) Repeat (ii) to (v) As many times as bits in Q.
(vii) Remainder will be in A, Quotient in Q.

eg.

$$\begin{array}{r} 10 \\ 11 \overline{)\,1000} \\ \underline{11} \\ 10 \end{array}$$

Here, we need to take 5-bit for A, as 4-bit magnitude + 1-bit

$7 \div 3$.

$$\begin{array}{r} 10 \\ 11 \overline{)\,111} \\ -110 \\ \hline 1 \end{array}$$

← D. (3-bits).

M.

3+1

Restoring Division:-

$3 \equiv 0011$

$-3 \equiv 1101$.

A	D.
0000	0111.

shift.

0000 1110

+ 1101

1 0 1 1

1 1

0000 1110 0

shift.

0001 1100

+ 1101

1 1 1 0

1 1

0001 1100 0

shift.

0011 1000

+ 1101

0 0 0 0

0000 1000 1

shift.

0001 0001

+ 1101

1 1 1 0

1 1

0001 0001 0

R Q.

$\therefore Q \equiv 0010$.

$R = 0001$.

5.

$$
\begin{array}{r|l}
101 & 10001 \\
\hline
 & 10.
\end{array}
\qquad 11
$$

A M. $M \equiv 00101$

$-M \equiv 11011$

000000 10001.

shift.

000000 10001.

+ 111011

111011

+ 000101

000000 00101.

shift.

000000 00101.

+ 111011

111101

+ 000101

000010 00101.

shift.

000010 00101.

+ 111011

111101

+ 000101

000100 01001.

shift.

000100 01001.

+ 111011

111111

+ 000101

000100 01001.

shift.

001000 10001.

+ 111011

000011

000011 10001.

000011 10001.

+ 111011

000010

000010 00011.

000010 00011.

R. Q.

$R \equiv 0010$

$Q \equiv 0011$

Non-restoring division:-

(i) If sign of A is 0, shift A and Q left one bit position and subtract M from A;
otherwise, shift A and Q left and add M to A.

(ii) Now if the sign of A is 0, set q_0 to 1;
otherwise set q_0 to 0.

- If sign of A is 1, add M to A.

$17 \div 5$:

$$101 \overline{)10001} \quad 11$$

$$10$$

A:

000 000 — 10001

shift. 000 001 — 0001 □
111011
1 11100 — 0001 ①

shift. 111000 — 0010 □
000101
1 11101 — 0010 ①

shift 111010 — 0100 □
000101
1 11111 — 0100 ①

5 ≡ 000101
−5 ≡ 111011

shift 111110 — 1000 □
000101
0 00011 — 1000 ①

shift 000111 — 0001 □
111011
0 00010 — 0001 1.

Q ≡ 00011.
R ≡ 00010.

If 1, add ⑤.

shift. 11110 — 1000 □
11011
11100 — 10000

$7 \div (-3)$

$$
\begin{array}{r}
-2 \\
1101 \overline{)\,0111} \\
\hline
1
\end{array}
$$

$M \equiv -3 \equiv 1101$

$-M \equiv 3 \equiv 0011$

	0000	0111
shift	0001	111 □
+	0011	
	0100	111 1
shift	1001	11 □
+	1101	
	0110	11 1
shift	1101	1 □
+	1101	
	1010	11 0

Add M.

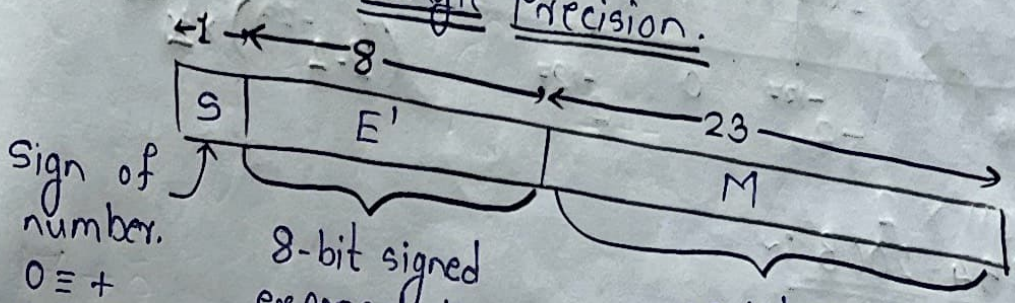
$$
\begin{array}{r}
1010 \\
1101 \\
\hline
111
\end{array}
$$

	0000	0111
shift	0000	111 □
+	0011	
	0011	111 1
shift	0111	111 □
+	0011	
	1010	1110

Floating Point Representation:-

- **Fixed Point Notation:-**

Single Precision.

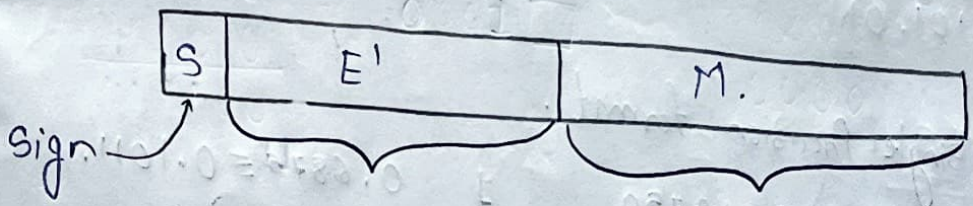


$$\pm 1.M \times 2^{E'-127}$$

For binary 128 bit,

S	E'	M
1	15	112

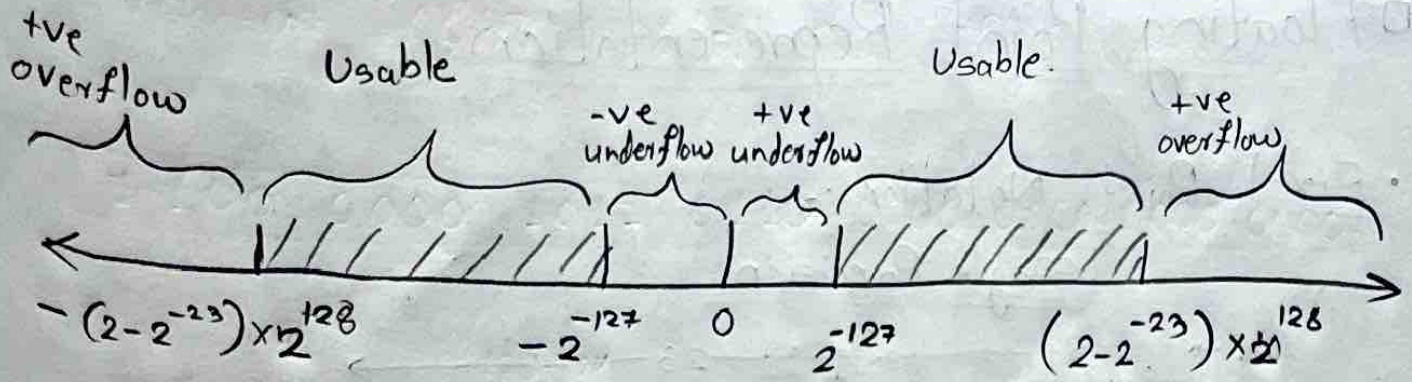
Double Precision.



$$\pm 1.M \times 10^{E'-1023}$$

- **Normal Number:-**

The most significant digit of the significand is non-zero.



□ IEEE 754-2008 :~

- Arithmetic format.
- Basic format.
- Interchange format.

• Additional Format:-

- Extended Precision Format.
- Extendable Precision Format.

eg. 5.6875

IEE754 single Precision format.

$5 \equiv 101.$

$0.6875 \times 2 = 1.3750$
$0.3750 \times 2 = 0.7500$
$0.75 \times 2 = 1.5$
$0.5 \times 2 = 1.0$

1 $0.6875 \equiv 0.1011$
0
1
1.

$5.6875 \equiv 101.1011.$

$\therefore 5.6875 = \underbrace{1.011011}_{M.} \times 2^{2.}$ ↘ E

$E' = 127 + 2 = 129.$

$= 10000001$

S	E'	M.
0	10000001	01101100000000000000000

Q. Represent 14.75 in IEEE single precision format.

→ 1110.1100

$0.75 \times 2 = 1.5$

$0.5 \times 2 = 1.0$

$0.0 \times 2 = 0$

$\equiv 1.11011 \times 2^3$

E E = 3 $E' = 127 + 3 = 130$.

S = 0.

S	E'	M.
0	10000010	11011000000000000000000

Q. Find out decimal equivalent of $(40B6\ 0000)_{Hex}$.

→ $6 \times 16^4 + 11 \times 16^5 + 4 \times 16^7 = 2.696 \times 10^{67}$

Q. 40B6 0000 is IEEE Representation. in Hex.

0100 0000 101 1 0110 0000 0000 0000 0000

$\approx 1.011011 \times 10^{129-127}$

≈ 101.1011

$\approx 5.$ ___ $\frac{1}{2} + \frac{1}{8} + \frac{1}{16} = 0.6875$

$= 5.6875.$

Q. Represent C0B6 0000 H in decimal.

1	100 0000	1	011 0110 0000 0000 0000 0000.

$= -1.011011 \times 2^{129-127}$

$= \underline{-5.6875}$

Q. C235 8000 H. in decimal.

1	100 0010	0	011 0101 1000 0000 0000 0000

$\equiv -1.01101011 \times 2^{5(132-127)}$

$\equiv -101101.011$

$\equiv \underline{-45.375.}$

Q. 41ED 8000 H. in decimal.

0	100 0001	1	110 1101 1000 0000 0000 0000

$\equiv +1.11011011 \times 2^{4}$

$\equiv 11101.1011$

$\equiv \underline{29.6875.}$

Considering the eqn $\pm 1.M \times 2^{E'-127}$.

M = Mantisa, s = sign bit.

E' = Excess-127 bias.

If E = 1111 1111, and M = 000...000, then the represented no. is infinity. (+ve or -ve, 's').

If E = All one's, and M ≠ all zero's, not a number. (NaN).

[Usually comes when we take square root of -ve no.]

If $1 \leq E \leq 254$, then value represented is $(-1)^s \times (1.M) \times 2^{(E-127)}$.

If E = 0000 0000, M = 000...000, represents a value zero.

If E = 0000 0000, M ≠ All zeros, represents the value closest to zero, i.e. some denormalized one.

- Precision:- 23+1 = 24 bit precision.

(i) Largest no. possible is M = all one's, and E = 254.

(ii) $\pm 1.11\ldots1 \times 2^{(254-127)} + 1 = \infty$.

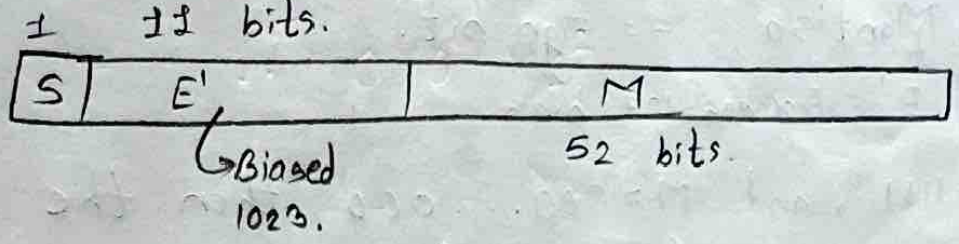
↑ add a little.

(ii) Min. possible no. M = all zero's and E = 1.

$\pm 1.00\ldots0 \times 2^{(-126)} - 1 = 0$. (De-normalized).

↑ subtract a little.

Double Precision:-



(i) If E = all ones (2047) and M = all zeros, it represents ∞ (infinity).

(ii) If E = All ones, M ≠ All zeros, it represents NaN (Not a number).

(iii) If E = All zeros, M = All zeros, it is a Zero.

(iv) If E = All zeros, M ≠ All zeros, it represents a value near to zero i.e. a denormailized no.

(v) If $1 \leq E \leq 2046$, it represents

$$(-1)^S \times (1.M) \times 2^{E-1023}$$

- Precision = 1 + 52 = 53 bits.

(i) Largest no. E = 2046. M = All ones.

$\pm 1.111\ldots111 \times 2^{1023}$ $\pm 1.8 \times 10^{308}$

(ii) Smallest no. E = 1, M = All zeros.

$\pm 1.00\ldots00 \times 2^{-1022}$ $\pm 2.2 \times 10^{-308}$

Q. Represent 29.6875 in IEE 754 Double Precision format.

11101.1011.

1.1101101×2^{4}.

$+1.1101101 \times 2^{1027-1023}$

$0.6875 \times 2 = 1.3750$

$0.375 \times 2 = 0.750$

$0.75 \times 2 = 1.50$

$0.5 \times 2 = 1.0$

S	E' (11 bits)	M (52 bits).
0	10000000011	1101101100...00

Q. C046 B00....00 H. to IEE 754 Double Precision Format. to Decimal.

1	00 0.000 0100	0110 1001 0000 0000

$-1.0110\,1001 \times 2^{5}$

$-101101.0001 = -45.3125$.

Q. A ≡ 414C 0000 H

B ≡ 40E8 0000 H.

Floating Pt. Add.

A ≡	0	100 0001 0	100 1100 0000 0000 0000 0000
B ≡	0	100 0000 1	110 1000 0000 0000 0000 0000.

$E_A = 128+2$

$E_B = 128+1$.

$E_A - E_B = 1$.

Shift M_B to Right by 1. OR Can also shift M_A to left, but it's better to loose LSB than to loose MSB rather

$A = +1.10011 \times 2^3$

$B = +1.11010 \times 2^2$

$= +0.11101 \times 2^3$

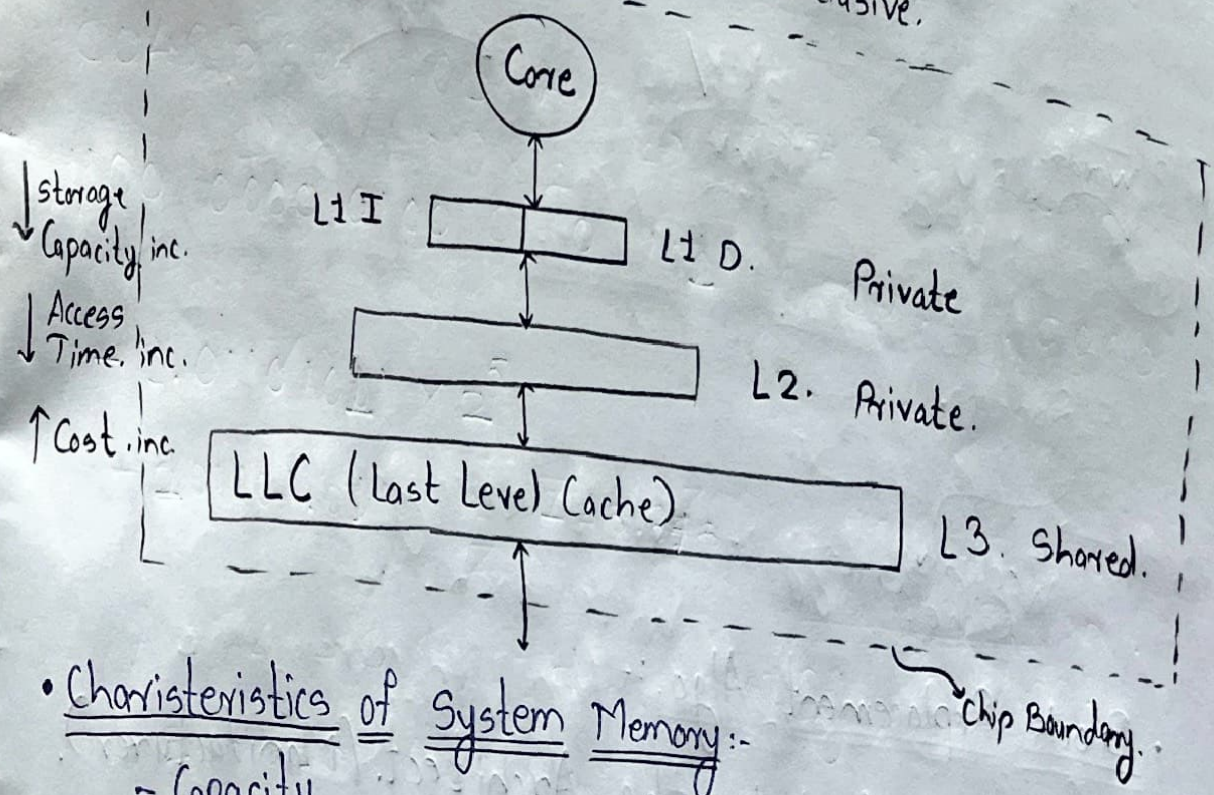
$A + B = 10.10000 \times 2^3$

$A + B = 1.01 \times 2^4 = (20)_{10}$

Memory Hierarchy :-

- Differentiate among different levels of Cache that are in Intel and AMD Ryzen Processors.

Points to be Considered :-
- Cache Size. (Capacity).
- Replacement Policy.
- Block size.
- Inclusive / Exclusive.



- Charisteristics of System Memory :-
  - Capacity
  - Transfer. Units.

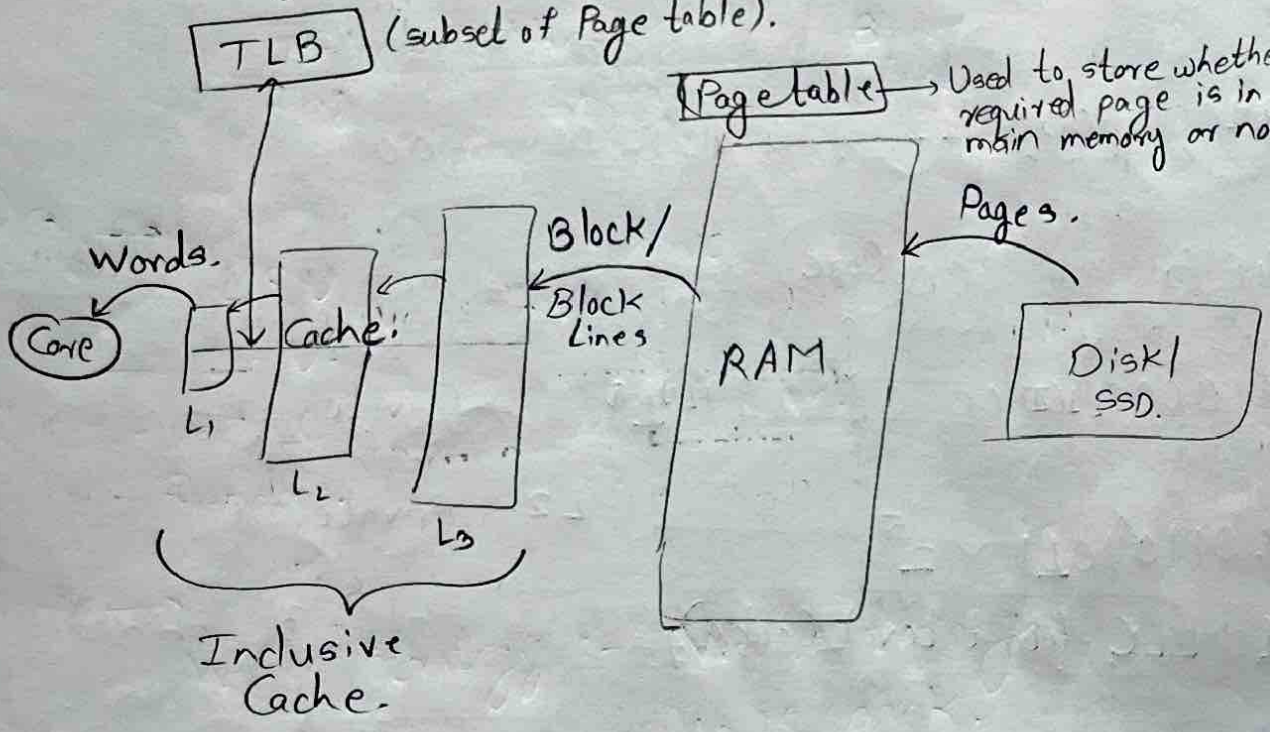
- Methods of Accessing Units :-
  - Sequential Access.
  - Direct Access
  - Random Access
  - Associative Access

- Performance Measures :-
  - Latency.
  - Memory cycle.
  - Transfer Rate.

- TLB is partial page table. (Transfer Lookaside Buffer).

If the entry is present in TLB, then only the page is in Memory. Now there's possibility that the data is in L2 or L3.

If entry is not in TLB, then page is not in memory, so obviously it will not be in Cache. Page swapping needed.



- Block Replacement Algo.:-

Block Size is fixed by the Architect (Manifacturer).

- Cache Addresses-
  - Physical
  - Logical.

Replace Algo.
- LRU
- LFU
- FIFO
- Random.

Write Policy.

Virtual Memory :-

- It allows programs to address memory from a logical p.o.v., regardless of physical capacity.
- Gives extended range of Addresses.
  - Physically Tagged Cache.
  - Logically Tagged Cache.

Q.1. Where can a block be placed in the upper level? (Block Placement)

Q.2. How is a block found if it is in the upper level? (Block Identification).

Q.3. Which block should be replaced on a miss? (Block Replacement).

Q.4. What happens on a write? (write strategy).

Patterson.

Cache Type :- (i) Direct Map.
(ii) Full Associative.
(iii) Set Associative. (Dominant).

Q.1.

→ (i) Direct Map.:

(Block Address) MOD (No. of blocks in Cache)

(ii) Full Associative.

Anywhere.

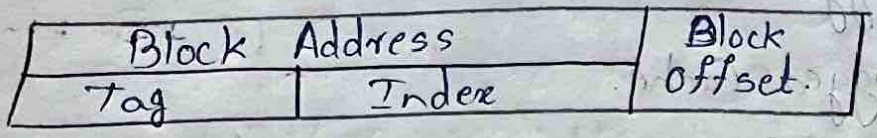
(iii) Set Associative.

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

If there are n-blocks in a set, Cache placement is called n-way set associative.

Q. 2. →

Address Tag, on each block gives the block address. With address tag, we provide a "valid bit". If the bit is not set, there cannot be a match.



Q. 3. →

- LRU
- LFU.
- FIFO.
- Random.

Q. 4. →

Write strategy. :- Write through.
Write back.

In order to pick data from Main memory, we apply Mapping.

- Direct Mapping.
- Fully Associative Maping.
- Set Associative Maping.

Direct Mapping

Q. Let, Main Memory = 16 kB. Cache Memory = 2 kB. Cache Memory block size is 32 bytes. Find no. of blocks.

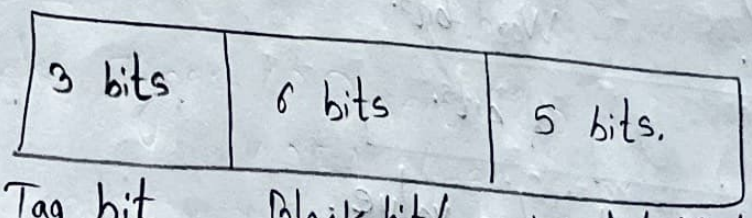
→ No. of blocks in MM = $\frac{16 \times 2^{10}}{2^5} = 2^9 = 512$.

No. of blocks in Cache Memory = $\frac{2 \times 2^{10}}{2^5} = 2^6 = 64$.

Q. 32 byte Block is loaded into cache memory. Processor will give ~~address~~ Memory address w.r.t. to MM block.

A block in Cache Memory will accomodate $2^9/2^6 = 2^3$ blocks.

8 blocks will be stored in ~~64~~ cache.



≡ 14 bit address by CPU.

Tag bit — Block bit/ Index. — Word/offset.

3-bits:- represents a typical block (whether the required block is in ~~memory~~ cache or not).

6-bits:- represents the block index.

5-bits:- represents the word out of one block.

If tag bits are matching, then Cache Hit.
else, Cache Miss.

If Miss happens, the required block is reloaded/swapped into the cache.

If present cache Memory block is modified, it must be kept in Main Memory first, and then put into the Cache. Then fetch.

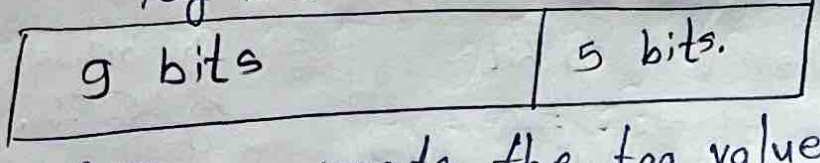
- Valid Bit. (If Data in main memory is loaded in Cache, then valid bit is 1) (else valid bit is 0).
- Dirty Bit.

[If data from the Cache block is modified, then the Dirty Bit is set to 1, else 0.
i.e. D = 1, means the block must be put in the Memory before replacement].

For each block we have
(i) Valid bit. (ii) Dirty bit. (iii) Tag bits.

Full Associative

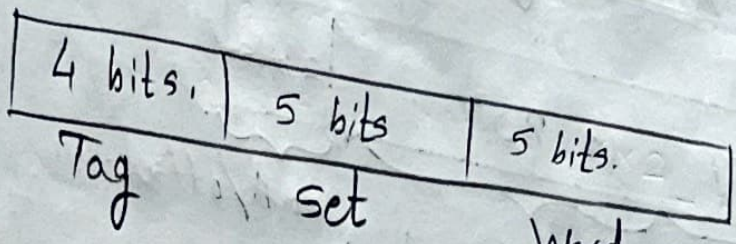
Ex.2.
Consider MM = 16 KB. Cache Memory = 2 KB.
Block size = 32 Bytes.



9-bits of Tag represents the tag value.
If matching, Cache Hit.
else, Cache Miss. (Block from MM is fetched).

Set Associative.

2 blocks per set.

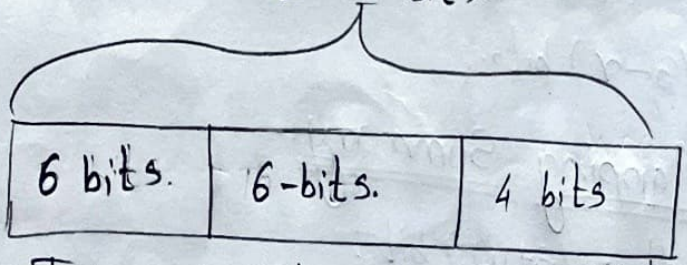


0-31 of MM can land on 0-31 of cache resp.
32-63 of MM can land on 0-31 set of cache resp.
Any set can be loaded with any of it's 16 blocks.

This is two way set associative mapping, as only two are present in a set.

Set Associativity ↑ as no. of cores ↑.

Q. Consider a Computing system, having 4 way set associative Cache of 4KB, MM = 64 KB. Block size = 16 bytes. Describe Cache Maping.



$2^{12}/2^4 = 2^8$

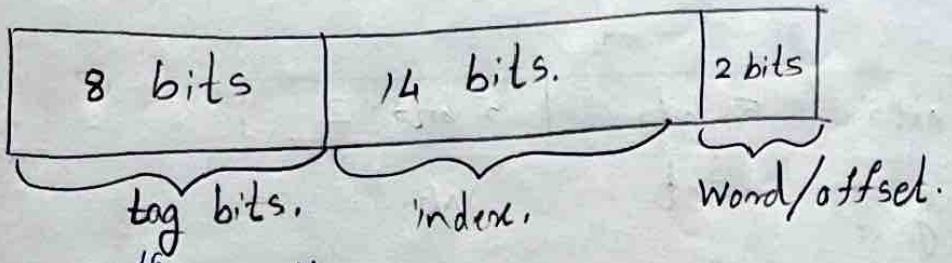
Tags = $2^{16}/2^{12} = 2^4$.
+2 for 4 sets.

$2^{12}/2^{4} = 2^8$ 4 sets.

2^6

Q. MM = 16 MB. Cache size = 64 kB. Block-size = 4 Bytes.

→ Direct Mapping:



$$\frac{64\ kB}{4\ bytes} = \frac{2^{16}}{2^2} = 2^{14}$$

$$\frac{16\ Mb}{2^{16}} = \frac{2^{24}}{2^{16}} = 2^8$$

Direct Mapping Summary:-

Address length = (s + w) bits.

No. of Addressable units = $2^{(s+w)}$ words/bytes.

Block size = line size = 2^w.

No. of blocks in memory = $2^{(s+w)}/2^w = 2^s$.

No. of lines in Cache = $m = 2^r$.

Size of tag = (s − r) bits.

Associative Mapping Summary:-

Address length = (s + w) bits.

No. of Addressable units = $2^{(s+w)}$ words/bytes.

Block size = line size = 2^w.

No. of blocks in memory = $2^{(s+w)}/2^w = 2^s$.

No. of lines in Cache = Undefined.

Size of tag = s bits.

Set Associative Mapping Summary:-

Address Length = $(s+w)$ bits.

No. of addressable units = $2^{(s+w)}$ words/bytes.

Block size = line size = 2^w words/bytes.

No. of blocks in memory = $2^{s+w} / 2^w = 2^s$.

No. of lines in set = k.

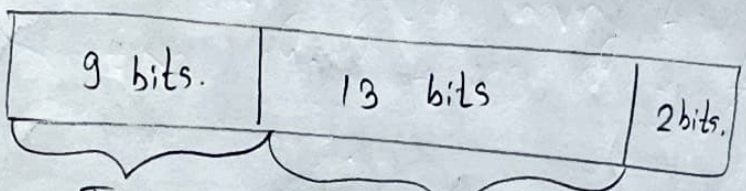
No. of sets = $v = 2^d$. (in cache)

No. of lines in cache = $m = kv = k \times 2^d$.

Size of Cache = $k \times 2^{(d+w)}$

Size of tags = $(s-d)$ bits.

MM = 16 MB. Cache size = 64 Cache. Block size = 4 bytes.



$$\frac{2^{24}}{2^{16}} = 2^8 \times \underset{\uparrow}{2} = 2^9.$$

k

6

1

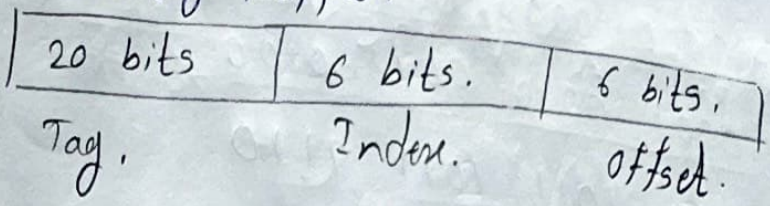
0001 1010 0001 1010

is in cache.

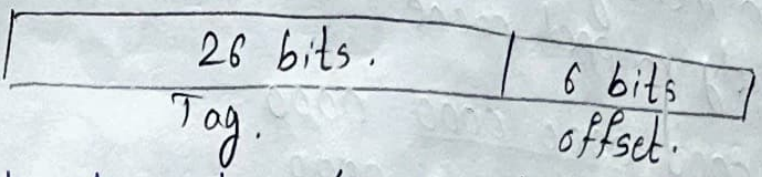
Q. 32 bits address. 64 bytes block size. Cache. Directly Mapped.
~~Block size = 2^6~~. no. of blocks in cache = 2^6.

→ Index = 6 bits. offset = 6 bits.

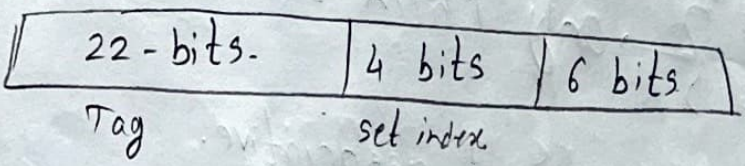
Directly Mapped.



Fully Associative.

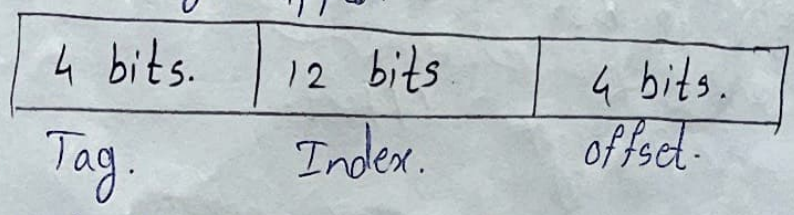


set Associative. (4 way set associative)

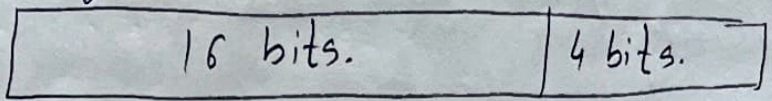


Q. MM = 1 mB. Word size = 1 byte. Block size = 16 bytes.
Cache = 64 KB.

- Directly Mapped.



- Fully Associative.



Q. Find given position

F0010, 01234, CABBE

F0010 = 1111 0000 0000 0001 0000

01234 = 0000 0001 0010 0011 0100

CABBE = 1100 1010 1011 1011 1110

Give two main memory addresses that map to same cache for directly mapped cache

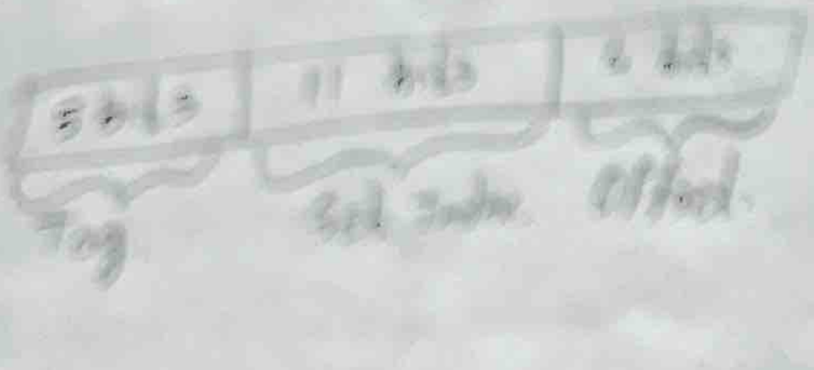
→ eg. 0000 0000 0000 0000 0000

1111 0000 0000 0000 0000

Also, 0FFFF } → Only match the Index

0FFFF

~~Tag~~ 8 = Two way set associative



Online lec

Q. From prev. question.

F0010, 01234, CABBE.

F0010 ≡ 1111 0000 0000 0001 0000

01234 ≡ 0000 0001 0010 0011 0100

CABBE ≡ 1100 1010 1011 1011 1110

Give two Main memory addresses that maps to same cache for directly mapped cache.

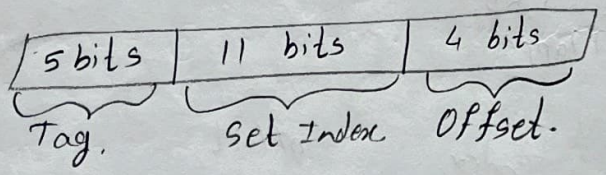
→ eg. 0000 0000 0000 0000 0000.

1111 0000 0000 0000 0000

Also, 0FFFF } → Only Match the Index.

3FFF1. }

~~Tway S~~ - Two way set associative.



Online Lec.

Paging :-

No. of page tables = Size available × $\frac{\text{PT entry}}{\text{size of PT}}$ × $\frac{1}{\text{PT enteries per process}}$

examples from William stalling.

Q. Consider a magnetic disk drive with 8 surfaces, 512 tracks, 64 sectors per track, and each sector with size 1 KB, Avg. seek time is 8 ms.

Track to track shifting time is 1.5 ms.

Drive rotates @ 3600 rpm. Successive tracks in cylinder can be read without head movement.

a] What is disk capacity?

b] What is the average access time? Assume file is stored in successive sectors and successive tracks, starting at sector 0, track 0 of cylinder 'i'.

c] Estimate the time required to transfer 5 mB file.

d] What is the burst transfer rate?

→ a] Capacity = $8 \times 512 \times 64 \times 1\,KB = 2^3 \times 2^9 \times 2^6 \times 1\,KB$

$= 2^{18}\,KB = 2^8\,MB. = 256\,MB.$

b] Avg. Access time = seek time + rotational delay.

Rotational delay = $\frac{\text{Rotational time}}{2} = \frac{60}{3600 \text{ rpm} \times 2} = \frac{1}{120}.$ (2 → Two Surfaces)

$= 8.333$ ms.

Access time = $8 + 8.33$ ms

$= 16.33$ ms.

c]. Time required to transfer 5×2^{20} Bytes.

Track size = $\frac{2^8\,MB}{8 \times 512} = 2^{-4}\,MB = 2^6\,KB. = 2^{16}$ Bytes. $\times 2^3$

We can access 8 tracks (cylinder) simultaneously.

∴ No. of cylinders required = $\frac{5 \times 2^{20}}{2^{19}} = 5 \times 2^1 = 10$ tracks-cylinders.

10 cylinders are required.

seek time = find cylinder.

rotational delay = find sector '0'.

To read 8 different tracks of 1 cylinder = $8 \times \frac{60}{3600} = 133.33$ ms.

Total Access time = $8 + 9(8.3 + 133.3 + 1.5) + 8.3 + 133.33$

$= 1296.17$ ms $+ 8.3 + 133.33$

$= 1437.8$ ms.

a] Burst ~~state~~ rate = Revolutions per sec × sectors per revolution × bytes per sector

$= \frac{60}{60} \times 64 \times 1\,KB$

$= 60 \times 64 \times 1\,KB$

$= 3840\ KB/s = 3.75\ MB/s$

Q. Consider a single platter disk.
Rotation speed is 7200 rpm. No. of tracks on side of platter is 30,000.
No. of sectors per track = 600.
seek time = 1 ms for every ~~every~~ 100 tracks traverse.
Let, the disk receive a request to access a random ~~seq~~ sector on ~~and~~ a random track. Assume disk head starts at track 0.

a] What is average seek time?
b] What is average rotational latency?
c] Transfer time for sector.
d] Total average time to satisfy a request.

→ a] Head start at track 0. The moment seek time is 0.
If requested track is 29999th track (last track).

Avg. seek time $= \frac{29999}{2 \times 100} = \frac{14999.5}{100} = 149.995$ ms.

(Average, $= \frac{0 + 29999}{2}$ tracks) 29999/100 ms will require for last track, and all in bet^n.

b] Rotational delay $= \frac{60}{\frac{7200}{120}} = 8.33$ ms.

For average $= \frac{8.33}{2} = 4.167$ ms.

c] Transfer time for sector $= \frac{8.33}{\text{no. of sectors} \rightarrow 600} = 0.0138$ ms.

d] Average time to satisfy req = Avg. seek time + Avg. rotational delay
$= 149.995 + 4.167$
(Sir said, also add the third one) $= 154.162$ ms.

Error Correcting Code:-

$2^K - 1 \geq M + K$.

eg. Data word: 1 1 0 1 (D_3 ... D_0)

Put, P_i at Power of 2.

$P_1 \rightarrow 2^0$ having 1.
$P_2 \rightarrow 2^1$ having 1.
0010.
$P_3 = 2^2$ having 1.
(All having 1 at 3rd pos. from right).

$D_3\ D_2\ D_1\ P_3\ D_0\ P_2\ P_1$
1 1 0 P_3 1 P_2 P_1

$P_1 \rightarrow$ Pickup 1, skip 1. (start from same). $D_3 D_1 D_0 P_1$
All those having 2^0 as 1

OR, write

C_1 C_2 C_4 C_8.

(All LSBs 1). (All 1's on second pos from LSB) ... so on...

XOR $C_8\ C_4\ C_2\ C_1$

$C_8'\ C_4'\ C_2'\ C_1$

Gives pos. of error.

Single Error Correcting. SEC

Double error detecting. DED.

CRC :-

eg. $P(x) = x^4 + x^3 + 1$

$= 11001$. Data: 110011.

Append 4 zeros, as degree = 4.

Sender side.

```
      100001
1001 ) 1100110000
       11001↓↓↓↓↓
       ----------
       0000010000
           11001
           -----
           01001.
```

→ To send:

1100111001.

Receiver side.

```
      100001
1001 ) 1100111001
       11001↓↓↓↓↓
       ----------
       0000011001
           11001
           -----
           00000
```

↙ Correct code. No errors.

eg. Original Msg. 1011011. Generator: 1101.

Append 3 zeros

```
      1100100
1101 ) 1011011000
       1101↓
       -----
       01100
        1101↓
        ----
        00011 10
           1101↓
           -----
           001100
            1101
            ----
            0001
```

To send:-

1011011001.

At Receiver:

```
      1100101
1101 ) 1011011001
       1101↓
       -----
       01100
        1101↓
        ----
        00011 10
           1101↓
           -----
           001101
            1101
            ----
            0000
```

↙ No error.

Generator Polynomial selection Criteria:-

- Polynomial should not be divisible by x.
- Should be divisible by $x+1$.

Dependencies:-

• Data Dependence in Programs:-

Consider following code fragment :-

S1:	Load R1, A	/R1 ← Memory(A)/
S2:	Add R2, R1	/R2 ← (R1)+(R2)/
S3:	Move R1, R3	/R1 ← (R3)/
S4:	store B, R1.	/Memory(B) ← (R1)/.

S2 is flow-dependent on S1.
S3 is anti-dependent on S2.
S3 is output dependent on S1.
S2 and S4 are totally independent.

• I/O Operations:

S1:	Read (4), A(I)
S2:	Process
S3:	Write (4), B(I)
S4:	Close (4).

S1 and S3 are I/O dependent.

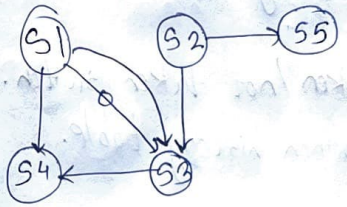
ex:

S1:	Load R1, 1024	/R1 ← 1024/.
S2:	Load R2, M(10)	/R2 ← Memory(10)/
S3:	Add R1, R2	/R1 ← (R1)+(R2)/
S4:	store M(1024), R1	/Memory(1024) ← R1/
S5:	store M((R2)), 1024.	/Memory(64) ← 1024/

where (Ri) means the content of register Ri and memory(10) contains 64 initially.

Draw dependence graph.

→ S3 is flow dependent on S1 and S2.
S4 is flow dependent on S3.
S5 is flow dependent on S2.
S3 is output dependent on S1.



S_4 and S_5 ~~are~~ need to use the same storage, thus they are potentially resource dependent if there is only 1 resource executor.

Types of Data Hazard :-

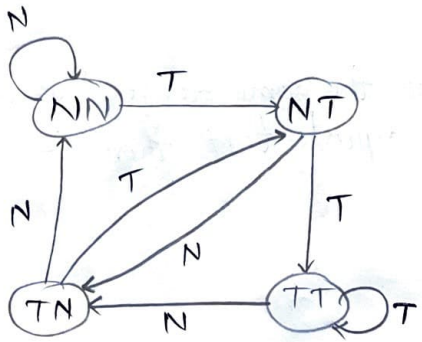
- RAW (Read after write) / True dependency.
- WAR (Write after Read) / Anti-dependent.
- WAW (write after write) / Output-dependent.

Control Hazards :- / Branch Hazards :-

Branch Prediction :-

Static Approaches :- (i) Predict never taken.
(ii) Predict always taken.
(iii) Predict by opcode.

Dynamic Approaches :- (i) Taken / not taken switch.
(ii) Branch history table.



Kai Hwang
IMP

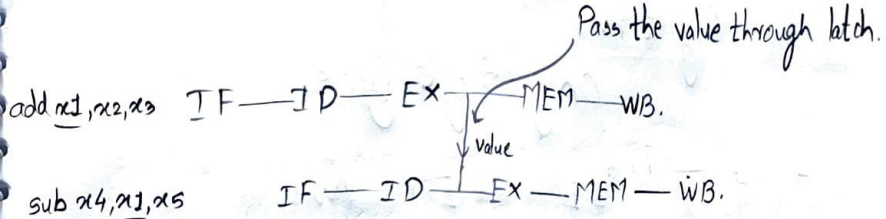
Pipelining and ISA Design.

Chapter-4 (34)
William stalling.

Stalling causes the pipeling bubble.

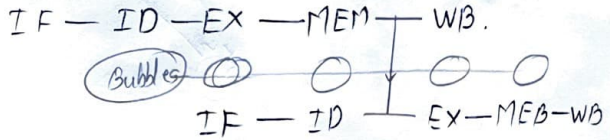
Remedies :-

- Forwarding (aka Bypassing)



Don't wait for the data to be stored in register, create extra datapath.

Load use Hazard :
If value not computed when needed.



- Code Scheduling to Avoid stalls :-

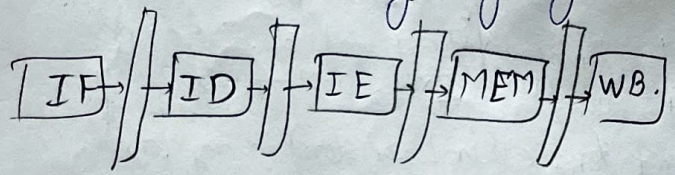
Change the sequence of instruction code, to avoid scheduling.

Control Hazards Remedies:-

- Wait/stall on Branch.
- Branch Prediction.
  - ~ Static Branch Prediction.
  - ~ Dynamic Branch Prediction.

- Pipeline Registers:-

Only Single Cycle.



RISC and CISC Characteristics:-

IMP

Data Bus:-

This is not mentioned in the reading sequence.

- Data lines that provide a path for moving data among system components/devices.
- No. of lines = Width of Data Bus.

Address Bus Control Bus.

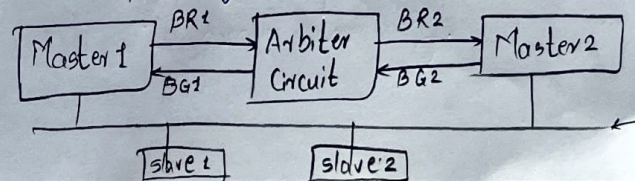
- Point-to-Point Interconnect:- Ch. 3 William Stallings.
- Quick Path Interconnect:- (QPI). Ch. 7. Zaky Hamature.
- Peripheral Component Interconnect:- (PCI): (Pg. 230)

- Synchronous Bus. (Based on Clock).
- Asynchronous Bus. (Based on Handshake) [Similar to MPT].

- Arbitration:- (IMP)

- Two devices want to access single resource in computer system.

Based on Priority.



Page 238

- Keyboard to Processor Interface:- Block Diagram.

- Universal standards:-

  - USB (Universal Serial Bus):-

  - Serial Bus tree structure:-

  - FireWire :- (developed by Apple).

  - PCI :-

  - SCSI Bus:-

  - SATA:-

  - SAS:-

  - PCI Express:-

Data Transfer Related Things.

Co-processor + GPGPU Not there in syallbus.