# Computer Organization

Ref book:

COA by Morris Mano

COA by Paul Chowdary

Computer Architecture is about
(i) Instruction
(ii) Addressing modes
(iii) Data Format

* Capacity of instruction

- RISC (Reduced Instruction Set Computer)
- CISC (Complex Instruction Set Computer)

RISC: Necessary condition for an instruction to be in RISC is

CPI = 1

↓

Clocks Per Instruction

Since every instruction must be finished in single clock cycle,

RISC has ~~beloo Disadvantag:~~

* Limited capacity
* Fixed instruction size
* Limited addressing Mode (IO devices can be addressed only with direct addressing mode)

Advantages:

→ Due to fixed size instruction,

* It is simple to design compiler
* They are more compatible with instruction pipeline

→ RISC processors have more CPU registers

∴ Efficient parameter parsing (This because of the fact that more registers provides register window) ~~and that is~~

→ RISC systems are used in Hardware Control Unit design

Programmer overhead is more in RISC Processor.

CISC:

→ CPI ≥ 1

→ Has More Capacity

→ Instruction size is not fixed (i.e. variable length)

→ More addressing modes are available

Disadvantage:

* Complexity is more while designing a compiler
* Less compatible for instruction pipeline.
* Less efficient parameter passing (∵ No of CPU registers is less and hence no register window)
→ Most of the memory (registers) is devoted for special purposes. Hence register window is not available.

Advantages:

* Programmer overhead is less

→ Let programmar overhead in RISC & CISC be denoted by P, & and Compiler design complexity in RISC & CISC be denoted by R, S. The we have below relations

$P > Q$, $R < S$,
$P > R$, $Q < S$

## Addressing Modes:

It is about how we access operands or how operand reference is given in the instruction.

Flexibility ∝ No of different addressing modes.

∴ CISC is more flexible than RISC



## Data Format:

It is about how we interpret binary strings

ASCII — uses 7 bits for Alpha Numeric
EBCDIC — uses 8 bits for Alpha Numeric
↓
Extended Binary Coded Decimal ~~Interchange~~ for Information Interchange

→ ASCII uses 8th bit for parity through it can do self error checking (checking if the coord is valid or not)

→ In ASCII
USA
$2^7 = 128$

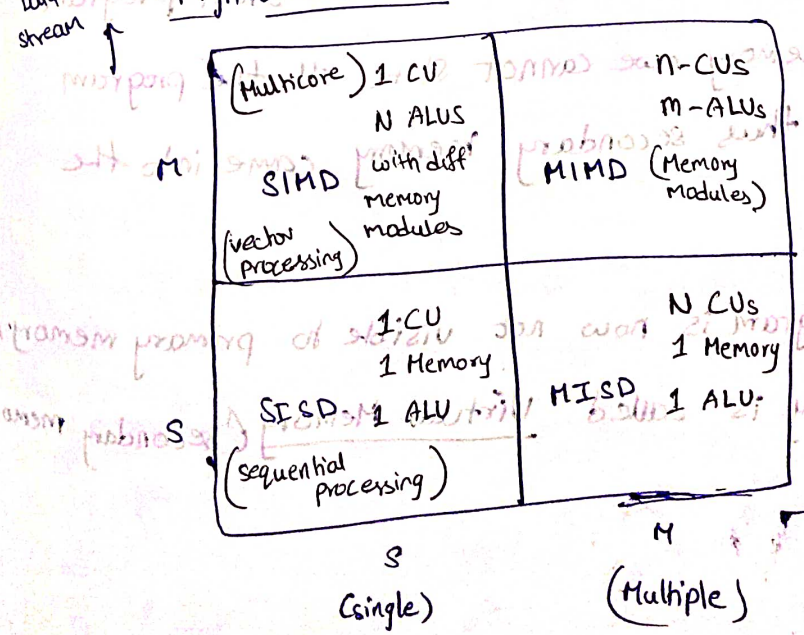
Still we have left over combinations which can be left for future use.

## Classification of Computer Architecture

Instructions & Memory organization

Memory store for program & Data

### Flyn Classification




(MISD is more of theoritical but not of practical)

→ Instruction is ~~denoted~~ issued by CU ∴ multiple instructions ⇒ multiple CUs

→ Multiple data ⇒ more ALUs are required.

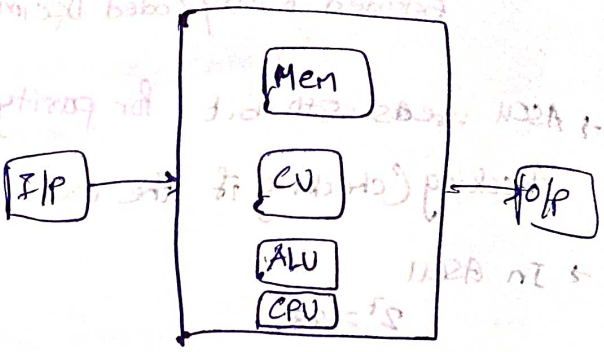
## Von-Neuman Architecture:

Here program and Data are stored in same memory. This is called stored program concept.

i.e., program & Data has to be stored before any processing.

Infact it is the first-memory based system.

→ This stored program concept supports inplace modification i.e., modifying the program or data itself ~~while~~ while processing




## Harvard Architecture:

→ It is a memory based architecture in which program and data are stored ~~sp~~ separately.

This seperation produces parallel access to b/w code and data and hence more efficient than Von-Neuman architecture.

**Note:**

The disadvantage with von-neuman architecture is when program size is larger than memory (we cannot store all the program in the memory and thus secondary memory came into the picture.

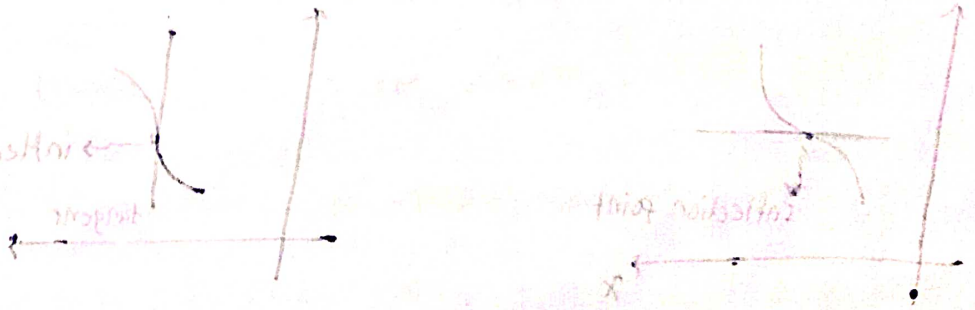
→ Since complete program is now not visible to primary memory, the added memory is called Virtual Memory (secondary memory)

## Computer Organization :

It deals with physical ~~deci~~ devices and their interconnections to improve performance.

→ performance : work done ~~per~~ per unit time

Million Instructions Per Second (~~MIP~~ MIPS) is a measure of performance.



# Computer Organization

(Q1) Performance gain of the machine $M_1$ over $M_2$ is 3. With respect to the following statements identify correct answer.

S1: $M_1$ takes 3 times the time that of $M_2$.

S2: $M_1$ increases performance by 200%.

S3: $M_1$ is 3 times faster than $M_2$.

which are true?

a) only S1, S2 b) S2, S3 c) S1, S3 d) All

(Q2) A system $M_1$ when operated with 1GHz clock takes 5 clocks per instruction on average. It runs the program of 1000 instructions.

i) what is total time to run the program.

CPI = 5

1000 instructions ⇒ 5000 clocks

Time period = 1 ns

⇒ total time = 5000 ns

= 5 μs

ii) what is its performance?

performance = no. of instructions per second

5000 ns ⟶ 1000 instruction

1 s ($10^9$ ns) ⟶ (Performance) ?

$$\text{performance} = \frac{1000}{5000} \times 10^9$$

$$= \frac{10^9}{5} = 200 \times 10^6 \text{ Inst/sec}$$

= 200 MIPS

(iii) Its improved version $M_2$ uses 20% faster clock, what is performance of this new machine.

Sol:

performance of $M_2$ = Performance of $M_1$ + 20% (performance of $M_1$)

= 200 + 20% (200)

= 240 MIPS

(iv) Performance gain of $M_2$ over $M_1$ is _____

$$= \frac{\text{performance of } M_2}{\text{performance of } M_1} = \frac{240}{200} = 1.2$$

i.e., performance of $M_2$ is 1.2 times that of $M_1$

(b) Note:

* $t = n * CPI * T$

$t \rightarrow$ execution time of program

$n \rightarrow$ no of instructions

$CPI \rightarrow$ clocks per instruction

$T \rightarrow$ Time period (per one clock)

* performance = Instructions per second

Q3 A processor $P_1$ takes 2 clocks per instruction and runs at 1GHz clock. Its improved version processor $P_2$ takes 2/3 of time of $P_1$ but increases CPI by 20%. what is clock rate of processor $P_2$?

Sol:

$CPI_1 = 2$ $\quad CPI_2 = 2 + 0.2 \times 2 = 2.4$

$f_1 = 1GHz$ $\quad f_2 = ?$

Performance of $P_2$ over $P_1$ = $\frac{\text{Time of } P_1}{\text{Time of } P_2} = \frac{\text{throughput of } P_2}{\text{throughput of } P_1}$

Let a program be with n instructions then

$\frac{2}{3}$ (time by $P_1$) = time by $P_2$

$\frac{2}{3} \times n \times CPI_1 \times T_1 = n \times CPI_2 \times T_2$

$\frac{2}{3} \times \frac{CPI_1}{f_1} = 1.2 \times \frac{CPI_1}{f_2}$

$f_2 = \frac{3}{2} \times 1.2 \times f_1$

$f_2 = 0.8 \times 1$

$f_2 = 0.8\,GHz$ $\Rightarrow f_2 = 1.8\,GHz$

(Q4) A dual core ~~processor~~ machine runs the program containing 60% parallel portion. what is the performance gain of dual core processor compare to single core processor

Sol:

for single core processor

100% time

for dual core processor

$40\% + \frac{60\%}{2}$

$40\% + 30\% = 70\%$

Performance of dual core over single core = $\frac{100}{70} = \frac{10}{7}$

$\cong 1.4285$

i.e., performance is increased by 42.85 %

(Q5) A novice processor takes twice the time to implement floating point operation compared to integer operation. However its improved new version takes only 2/3 of time for floating operation. If the program is the mix of 3:4 integer and floating point operations respectively what is the seepup of new processor compared novice one?

Sol:

Let $t_1$ be time taken by 1st processor

let n be no of instructions


int → x time

float → 2x time

$t_1 = \frac{3}{7}x + \frac{4}{7}(2x) = \frac{11}{7}x$

Let $t_2$ be time taken by 2nd processor

int → x time

float → $\frac{2}{3}(2x) = \frac{4}{3}x$

$t_2 = \frac{3}{7}x + \frac{4}{7}\cdot\frac{4}{3}x = \frac{9}{21}x + \frac{16}{21}x = \frac{25}{21}x$

gain of $P_2$ over $P_1$ =

$= \frac{t_1}{t_2} = \frac{\frac{11}{7}x}{\frac{25}{21}x} = \frac{33}{25} = 1.32$

# Memory Organization:

## Words & Bits:

→ Memory is accesed with address bus & databus.

→ words are addressed using address bus.

→ The unit of addressing in memory is word which does not need to be 8-bit unit.

→ The memory in which successive addresses refer successive bytes is called byte addressable memory.

→ If no of address bits to words are m, then no of address bits required is $\log_2 m$.

Eg:

256 word memory need address size of 8 bits.

→ when memory word is addressed, then all bits of word are accessed in parallel using databus. (Normally data bus size is of the word size)

→ Address bus is always unidirectional (address is given to memory but memory never gives any address)

→ Data bus can be unidirectional or bidirectional

(Read only) → ROM (Read & write) → RAM

→ Control bus dictates whether to perform read operation or write operation on addressed memory word.

→ Address bus, Data bus & control bus together are called system bus.

Note: 🙂

Consider 64 MB memory. It is can be designed in following ways:

1. 64M X 8 bits (64 million words and 8-bits/word)

64M words $\longrightarrow 2^6 \times 2^{20}$ words $= 2^{26}$ words

$\therefore$ address bus is of $2^{26}$ bit size

8 bits/word → data bits are 8 or data bus' size = 8

2. 32M × 16 bits (32 Million words and 16 bits/word)

no of words = $32 \times 2^{20} = 2^{25}$ words

∴ size of address bus = 25

size of data bus = 16

3. 16M × 32 bits (16 Million words and 32 bits/word)

$16 = 2^4 \times 2^{20}$

no of words = $16 \times 2^{20} = 2^{24}$ words

∴ size of address bus = $2^{24}$

size of data bus = 32

4. 8M × 64 bits (8 Million words and 64 bits/word)

5. 4M × 128 bits (4 Million words and 128 bits/word)

(66) In a system design 4GB main memory is organized as 64 bit words (eight 8 bit words). What will be the number of address bits required to access the memory.

Sol:

4GB memory ⇒ $4 \times 2^{30} \times 8$ bits

each word is 64 bits = $2^6$

∴ no of words = $\frac{4 \times 2^{30} \times 8}{2^6} = 2^2 \times 2^{24} \times 2^3 = 2^{29}$ words

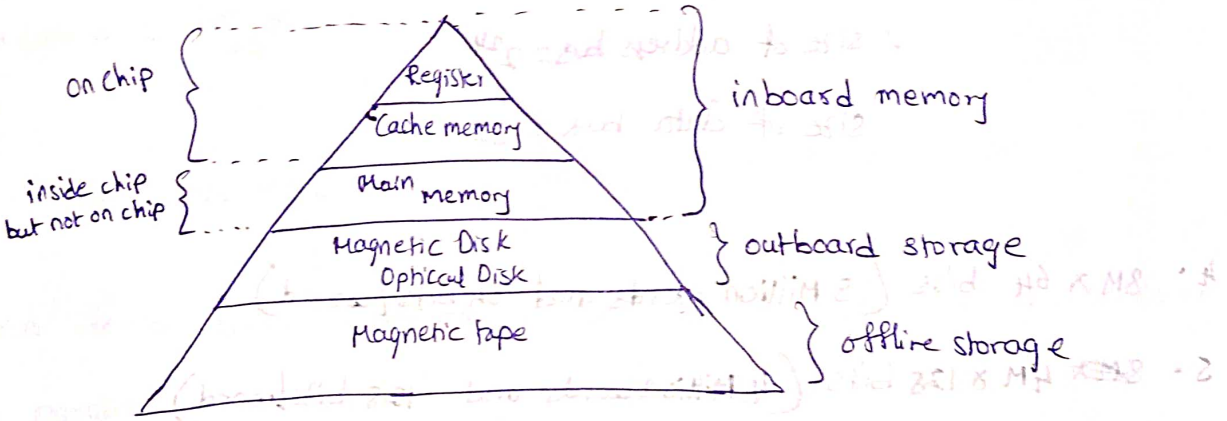
∴ no of address bits = 29

# Memory Hierarchy

→ Memory hierarchy is arrangement of different types of memories of different sizes with the objective of reducing average access time.

→ It ensures faster memories are accessed more frequently than slower memories.

→ Hierarchy is logical arrangement (not physical) so the reference for accessing memory is given by word address.




16/07/20

(87) Given the following design statements identify the most performing design

S1: Machine with 1GHz clock and CPI=5

S2: Machine with 2GHz clock and CPI=7

S3: Machine with 5GHz clock and CPI=10

a) S1 b) S2 c) S3 d) All S1, S2, S3 have same

Sol:

$t_1 = n \times 5 \times \frac{1}{10^9} = 5 \times 10^{-9}$

$t_2 = n \times 7 \times \frac{1}{2 \times 10^9} = 3.5 \times 10^{-9}$

$t_3 = n \times 10 \times \frac{1}{5 \times 10^9} = 2 \times 10^{-9}$

$t_3 < t_2 < t_1$ ∴ S3 has highest performance

90-10 to 10-10:

→ It says that to mostly 10% of the program uses 90% of the resources used by the program.

→ This 10% is called locality of reference.

→ The locality of reference is generally maintained in cache.

(88) In a system design 20 bit address bus is used and 8 bit bidirectional data bus is maintained. Identify the correct among the following options.

a) System uses 1GB RAM
b) System uses 1MB RAM
c) System uses 1GB ROM
d) System uses 1MB ROM

Sol:

given bidirectional data bus ⇒ RAM but not ROM

20 bit address bus ⇒ $2^{20}$ words

memory = $2^{20} \times 8$ bits

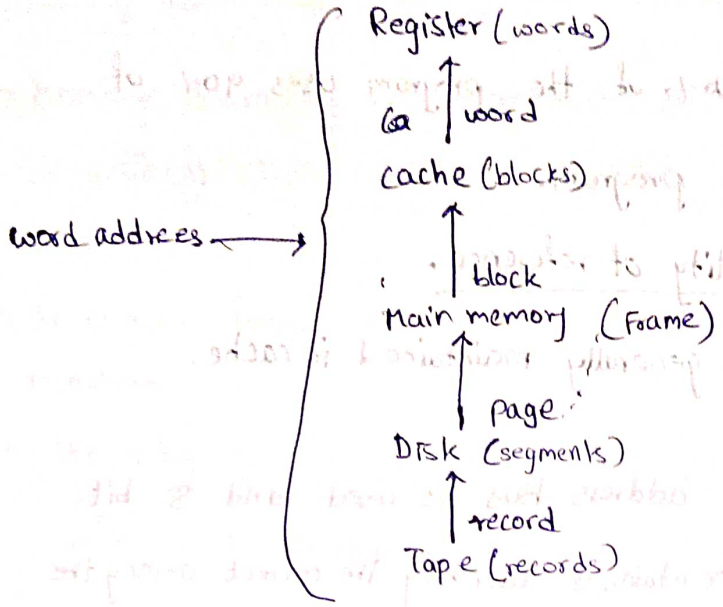
$= 2^{20}$ bytes = 1 MB RAM

∴ opt (b)

→ Based on how close the memory is to the processor the memories are assigned levels

register memory — level 0
Cache memory — level 1
Main memory — level 2
Disk memory — level 3
Tape memory — level 4

## Readjustment in memory Hierarchy:




Behaviour:

bipolar technology (BJT) { Register, Cache (SRAM) } — Constructed using Semi-conductors (∴ volatile)

uni-polar CMOS — Main memory (DRAM)

Disk — Magnetic / optical

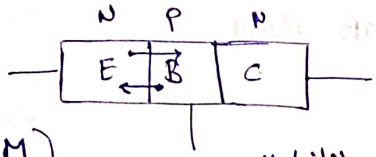
Tape — Magnetic

} non-volatile

CMOS - Complementary Metal Oxide Semiconductor set

BJT - Bi-Junction Transistor

## Bi Junction Transistor:

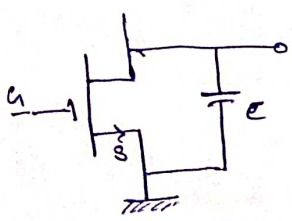



(SRAM)

(Static RAM)

Mobility is in two directions and hence memory is faster

## CMOS




Only one types of carries are moving

Hence mobility will be less.

The charge across the capacitor can't stay infinitely.

∴ It is called Dynamic RAM (DRAM)

Before the content is lost the capacitor needs to be charged again. This process is called Refreshing.

| SRAM | DRAM |
|---|---|
| (i) Faster due to bipolar technology | (i) Slow due to unipolar technology |
| (ii) Volatile | (ii) Volatile |
| (iii) Less package density (∵ we need two carriers for each bit) | (iii) More Package Density (∵ we need one carrier for each bit) |
| (iv) Refreshing is not required | (iv) Refreshing is required regularly |

Package density: no of bits that can be accomodated over given area

Properties of Memory Hierarchy:

1. In the hierarchy level 'i' memory is placed on top of level 'i+1' memory.
2. These levels are arranged such that

$T_1 < T_2$ ; $S_1 < S_2$ & $C_1 > C_2$ ; $f_1 > f_2$

(time) (size) (cost/byte) (frequency of accessing)

3. Containment (or) Inclusion i.e., Information in level 'i' ⊂ Information in level 'i+1'
4. Performance of memory hierarchy is given by hit ratio, h (i.e., availability of refered information at refered level)

$h \propto$ size

$h_i \propto \frac{1}{T_{avg}}$ ($T_{avg}$ - avg access time)

5) The drawback is · data inconsistency

i.e., same information will have different values at different memories

+ proper ~~wite~~ write operation would reduce this ~~prob~~ problem.

Note:

~~If 'h' is hit ratio then~~

If $h_i$ is hit ratio of level i then

$$h_0 < h_1 < h_2 < h_3 < h_4 \quad \& \quad h_4 = 1$$

→ Consider a two level memory system with

level $L_1$: $T_1$, $S_1$, $C_1$

level $L_2$: $T_2$, $S_2$, $C_2$

$S_T$ = total size of memory = $S_1 + S_2$

$C_{avg}$ = Avg of cost of memory system = $\frac{\text{total cost}}{\text{total size (in bytes)}}$

$$= \frac{S_1 C_1 + S_2 C_2}{S_1 + S_2}$$

$$= \frac{\sum S_i C_i}{\sum S_i}$$

→ Consider ~~two~~ 2-level memory system with

$T_1 = 10$ ns $\quad T_2 = 100$ ns

$h_1 = 0.8 \quad h_2 = 1$

Find avg access time.

Sol:

$T_{avg} = 0.8 \times 10\text{ns} + (1 - 0.08) \times 100\text{ns}$

$= 8\text{ns} + 20\text{ns} = \underline{28\text{ns}}$

17/07/20

# Computer Organization

(Q9) Given the following design statements identify most performing design

S1: 256MB cache, 8GB RAM (MM), 1 TB HDD

S2: 512MB cache, 4GB RAM (MM), 500GB HDD

S3: 1GB cache, 8GB RAM (MM), 500GB HDD

a) S1 b) S2 c) S3 d) All have same performance.

Sol:

for we need to check which system has more cache, then we check main memory and then HDD

∴ S3 gives better performance

∴ opt (c)

(Q10) In a three level memory hierarchy the hit ratios are $h_1, h_2, h_3$ identify most appropriate one among the following.

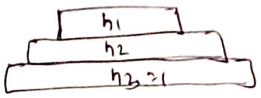
a) $h_1 > h_2 > h_3$ and $h_3 = 1$

b) $h_1 > h_2 < h_3$ and $h_3 = 1$

c) $h_1 < h_2 < h_3$ and $h_3 = 1$

d) $h_1 < h_2 < h_3$

Sol:




∴ $h_1 < h_2 < h_3$ & $h_3 = 1$

## Average access time Calculation:

### Case 1: (2-Level)

Processor is having bus for cache memory and main memory (Processor is having access with both level)

forming

, then we

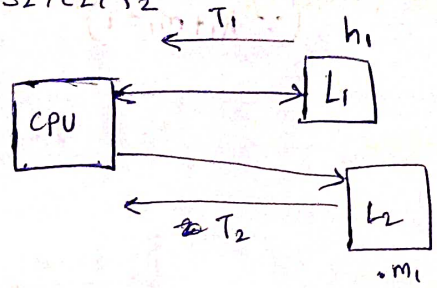
re $h_1, h_2, h_3$

memory



Let L1: $S_1, C_1, T_1$

L2: $S_2, C_2, T_2$




$m_1$ is miss ratio of level 1

$$T_{avg} = h_1 T_1 + m_1 T_2$$

$$\Rightarrow T_{avg} = h_1 T_1 + (1 - h_1) T_2$$

$T_{avg}$ is always for one word (i.e., time/word)

$$\therefore \text{Data transfer rate/throughput} = \frac{1}{T_{avg}} \text{ words/sec}$$

Case 2: (Exploring Locality of reference) → while solving problem if case is not mentioned we need to check for both the cases. However mostly in question case (ii) will be given

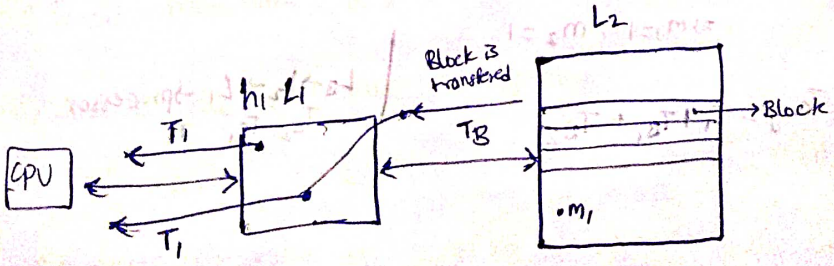
(i) Block size = one word

(ii) Block size > one word

→ Processor is having access with faster memory.

→ If there is a miss at faster memory, first a block of words are moved from slower memory to faster memory and addressed word is given to processor from faster memory

* word is always supplied from faster memory to CPU (it is never sent from slower memory)




$$T_{avg} = h_1 T_1 + m_1 \left[ T_B + T_1 \right]$$

$T_B$ → time taken to move block from memory to cache

$= h_1 T_1 + m_1 T_B + m_1 T_1$

$= (h_1 + m_1) T_1 + m_1 T_B \quad (\because h_1 + m_1 = 1)$

$= T_1 + m_1 T_B$

$\therefore$ **$T_{avg} = T_1 + m_1 T_B$**

Block size can be one word or n words

if block size = 1 word

$\Rightarrow$ **$T_B = T_2$**

if block size = n words

$\Rightarrow$ **$T_B = n * T_2$**

Extension for multiple level

**$T_{avg} = T_1 + m_1 T_{B_1} + m_1 m_2 T_{B_2} + m_1 m_2 m_3 T_{B_4} + \ldots$**

for 3 levels

$T_{avg} = T_1 + m_1 T_{B_1} + m_1 m_2 T_{B_2}$

(i) hit a level 1; $h_1 = 1$ & $m_1 = 0$

$T_{avg} = T_1$ (Level 1 gives the word)

(ii) miss at level 1 & hit at level 2

$\Rightarrow m_1 = 1, h_2 = 1 \Rightarrow m_2 = 0$

$T_{avg} = T_1 + T_{B_1}$

$L_2 \rightarrow L_1 \rightarrow$ processor
$T_{B_1} \quad T_1$

(iii) miss at level 1 & miss at level 2

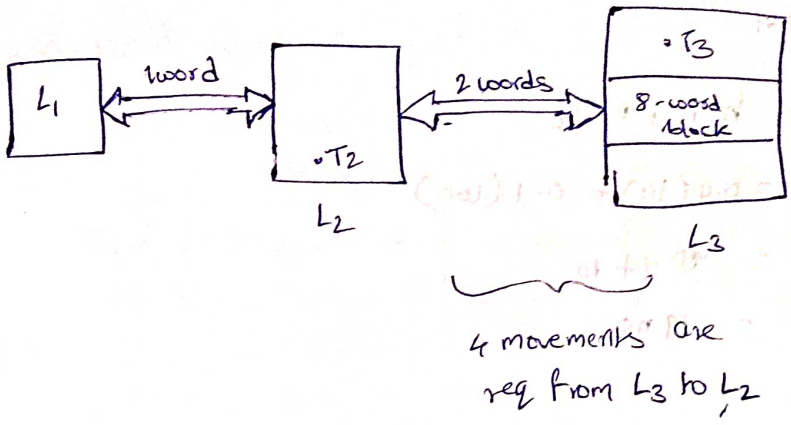
$\Rightarrow m_1 = 1, m_2 = 1$

$T_{avg} = T_1 + T_{B_1} + T_{B_2}$

$L_3 \rightarrow L_2 \rightarrow L_1 \rightarrow$ processor
$T_{B_2} \quad T_{B_1} \quad T_1$

→ Locality of reference will reduce penality (extra time)
But it reduces access time for future references (not for current ref)

## Placement time Consideration:




This is done as follows:

1st one word is moved from $L_3$ to $L_2$

once the previous word is made available in L2 then only start next two word unit.

Placing a word from data bus to memory is called placement time. In the previous two cases we didn't consider the placement time.

$T_3$ – Read from $L_3$ → Now data is in the bus (not in $L_2$)

$T_2$ – Then place the ~~place~~ word in the bus in $L_2$

∴ Time taken for one word unit to move from $L_3$ to $L_2$ is $T_3 + T_2$

(Q11) In a 2-level system access time of level 1 is 10 times faster than L2. Level 1 maintains 90% hit and its access time is 10ns

(i) what is the ~~ave~~ average access time

(ii) If average access time is allowed atmost 40ns, what is the minimum hit ratio at level 1?

Sol:

$$T_1 = \frac{T_2}{10}$$

$T_1 = 10ns \Rightarrow T_2 = 100ns$

$h_1 = 0.9$

$m_1 = 0.1$

$$T_{avg} = h_1T_1 + m_1T_2$$
$$= 0.9(10) + 0.1(100)$$
$$= 9 + 10$$
$$= 19\ ns$$

Case(ii) If the processor has access only to faster memory then

$$T_{avg} = h_1T_1 + m_1(T_1+T_2)$$
$$= 0.9(10) + 0.1(10+100)$$
$$= 9 + 11$$
$$= 20ns$$

(ii) $T_{avg} \leq 40$

i.e., $h_1T_1 + m_1T_2 \leq 40$

$h_1(10) + (1-h_1)(100) \leq 40$

$10h_1 + 100 - 100h_1 \leq 40$

$-90h_1 \leq -60$

$90h_1 \geq 60$

$$h_1 \geq \frac{2}{3}$$

Case(iii):

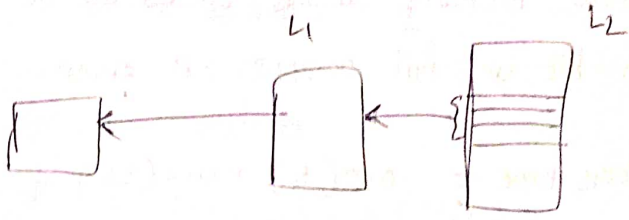
$T_{avg} \leq 40$

$$T_{avg} = h_1T_1 + m_1(T_1+T_2) \leq 40$$
$$\Rightarrow 10h_1 + (1-h_1)(110) \leq 40$$
$$10h_1 + 110 - 110h_1 \leq 40$$
$$-100h_1 \leq -70 \Rightarrow \boxed{h_1 \geq 0.7}$$

27/08/20

(Q12) In a 2-level memory level1 access time is 20ns and level2 access time is 200 ns. 90% of references present in L1. If there is a miss a 4-word block must be moved into L1 and addressed word is given to CPU later. what is avg access time

Sol:




avg access time = $0.9\,(20) + 0.1\,(4*200 + 20)$

(4*200: Block transfer to L1)

$= 18 + 0.1(820)$

$= 18 + 82 = 100\text{ns}$

Data transfer rate $= \frac{1}{\text{avg access time}}$ words/sec

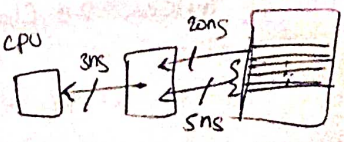
$= \frac{1}{100\text{ ns}} = \frac{10^9}{100} = 10^7$ words/sec

(Q13) A 2-level memory will have 3ns when hit occurs at level1. 94% of references are found in level1. If there is a miss at level1 the system takes 20ns to bring the first word from L2 to L1. Afterwards it takes 5 ns for each of the remaining word. Let block size is 32 words. ~~FAQ~~ Find avg access time.

Sol:

$0.94\,(3) + 0.06\,(20 + 31*5 + 3)$

$2.82 + 0.06\,(178) = 2.82 + 10.68 = 13.5\text{ ns}$



(Q14) In a memory system 50ns time is taken if there is a miss at faster memory and 5 ns for a hit with faster memory. If 80% references are present in faster memory, find the avg access time.

Sol:

Here individual memory access times are not given. Only times when hit or fail occurs, is given.

$$\therefore \text{avg access time} = 0.8(5) + 0.2(56)$$
$$= 4 + 10$$
$$= 14 \text{ ns}$$

Also performance or Data transfer rate

$$= \frac{1}{14 \text{ ns}}$$
$$= \frac{100}{14} \times 10^7 \text{ words/sec}$$
$$= 7.14 \times 10^7 \text{ words/sec}$$

(Q15) In above question, if word size is 8 bits what is the minimum bandwidth required for the bus.

Sol:

$$\text{Bandwidth} = (7.14 \times 10^7) \times 8 \text{ bits/sec}$$
$$= 57.12 \text{ MbPS}$$

(Q16) A memory system takes 2 clocks for cache hit to a read operation and 10 clocks for miss at cache for the same operation. For the write hit it takes 5 clocks and for miss it takes 20 clocks. If there are 60% read operations what is average access time? (take hit ratio as 80%)

Sol :

avg access time for read = 0.8 (2) + 0.2 (16)

= 1.6 + 2

= 3.6 clocks

avg access time for write = 0.8 (5) + 0.2 (20)

= 4 + 4

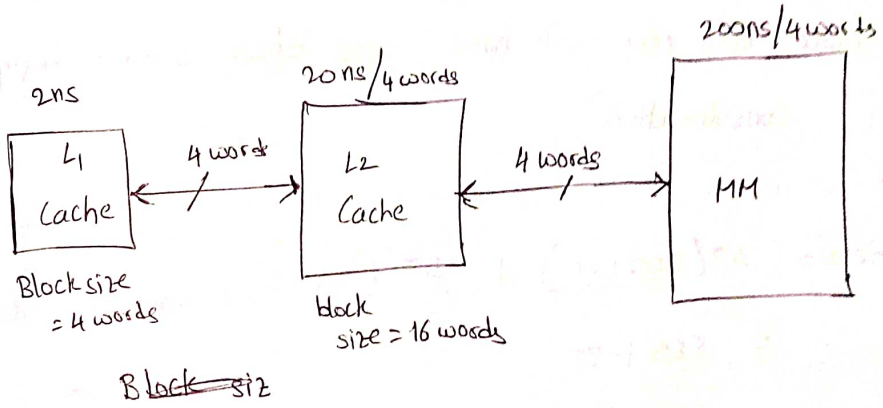
= 8 clocks

∴ total avg access time = 0.6 (3.6) + 0.4 (8)

= 2.16 + 3.2

= 5.36 clocks

Q17)




Block size

When there is a miss at L1 cache & hit at L2 cache a block is transfered from L2 to L1. What is the time taken to to transfer.

a) 2ns b) 20 ns c) 22 ns d) 88 ns

Sol:

Since L2 is sending the block

block size = 16 words

4 words are sent at a time.

∴ we need to move 4 moves.

Here we need to consider placement time also

i.e., 20 ns to put it on the bus and 2 ns for placement time.

∴ total time = 22 * 4 = 88 ns

Q18) In above question, if there is a miss at both $L_1$ & $L_2$ caches the block must be moved from MM to $L_2$ cache and the concerned block must be moved from $L_2$ cache to $L_1$ cache. What is the total block transfer time.

Sol:

a) 222 ns  b) 888 ns  c) 902 ns  d) 968 ns

Sol:

total block transfer time

= block transfer time from MM to $L_2$ cache

+

block tranfer time from $L_2$ cache to $L_1$ cache.

The cache closer to MM will have same block sizes (design consideration)

$\therefore$ req time = $4*(200+20) + 4*(20+2)$

$= 880 + 88$

$= 968$

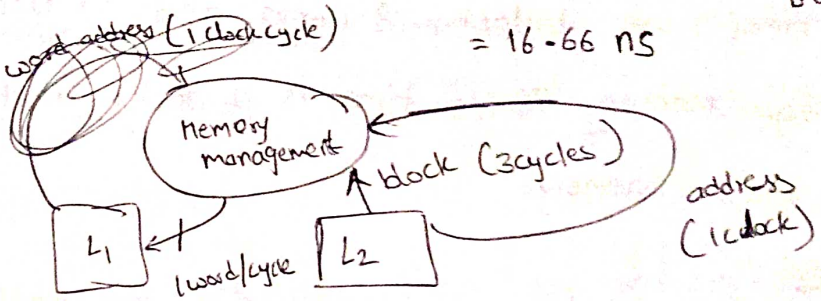
Q19) Certain processor operation with 60MHz clock and using 2-level memory. The block size is 8 words and each word contains 4 bytes. When there is a miss at level1, the internal management system takes 1 clock cycle to accept first word address of the block and pull all remaining words of the block in 3 clock cycles from level2, later it handover all the words to level1 at the rate of 1 word per cycle. How many total clocks are needed to serve miss request by this processor and how much time it takes?

Sol:

freq = 60MHz

time period of clock = $\frac{1}{60 \times 10^6} = \frac{1}{6 \times 10^7} = \frac{100}{6} \times 10^{-9}$ (or) $\frac{1}{60}\ \mu s$

$= 16.66\ ns$




$\therefore$ total no of clocks = 1 (address) + 3 + (8 * 1)

$= 12$ clocks

time req = $12 + \frac{1}{60}\ \mu s = \frac{1}{5}\ \mu s$

$= 0.2\ \mu s$

(Q20) In above question, the minimum bandwidth required for the bus connecting $L_2$ and $L_1$ is _____ $10^6$ bytes/sec

Sol:

to transfer one block (8 words = 32 bytes) = $0.2\ \mu s$

32 bytes $\longrightarrow 0.2\ \mu s$

160 bytes $\longleftarrow 1\ \mu s$

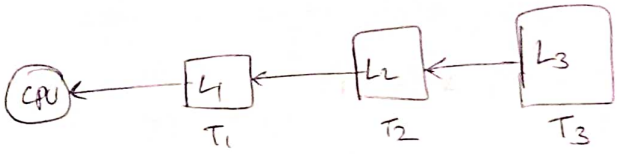
$\Rightarrow$ 1 s $\longrightarrow 160 \times 10^6$ bytes

$\therefore 160 \times 10^6$ bytes/sec

$\therefore$ 160 MBPS

(Q21) A processor design uses 3 level memory. The miss rate of level1 memory is twice to that of level2 memory. The access times are 1 clock, 8 clocks and 18 clocks respectively. If average memory access time is 2 clocks what are miss rates at L1 and L2 memory.

Sol:




$t = T_1 + m_1 T_2 + m_1 m_2 T_3$

$t = T_1 + (2m) T_2 + (2m)(m) T_3$

$2 = 1 + 2m(8) + 2m^2(18)$

$\Rightarrow 36m^2 + 16m - 1 = 0$

$36m^2 + 18m - 2m - 1 = 0$

$18m(2m+1) - 1(2m+1) = 0$

$\Rightarrow m = -1/2$ & $m = 1/18$

$\therefore 2m, m$

$\frac{2}{18}, \frac{1}{18}$

i.e., $\frac{1}{9}, \frac{1}{18}$

$\therefore 0.111$ & $0.056$

28/08/20

# Cache Memory

→ Cache is the smallest & fastest component in memory hierarchy It bridges speed miss match blw fastest processor to slowest memory.

→ It maintains locality of reference there by reduces average access time.

→ Cache and Main memory are logically divided into fixed size blocks.

→ the total no of blocks in MM are more compared to that of no of blocks in the cache.

→ Hence it is required to know which main memory block is to be placed and where in the cache.

for this we use Address mapping techniques.

→ Direct mapping is the simplest but results conflict problems.

→ Associative mapping resolves conflict problems but require more hardware.

→ Set associate mapping is the combination of direct mapping & associate mapping. (∵ it requires less hardware and less conflict problems)

→ Tag bits: These bits are used for cache memory blocks. They denote which MM block among the possible ones, for that cache position present at that time.

→ Write operation results cache coherence problem (i.e., same variable has different value in cache from memory)

→ proper updation techniques will reduce / resolve the cache coherence problem.

→ Block replacement techniques are aimed to minimize the no of block movement blw cache & MM.

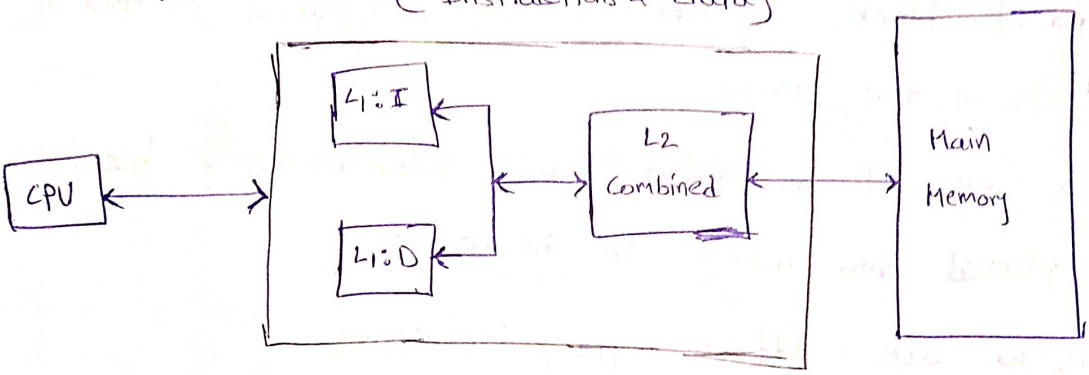
(Eg: FIFO, LRU, Direct mapped cache)

## Cache Design

① $L_1$ - Instruction cache (Holds only instructions)

$L_1$ - Data cache (Holds only data)

$L_2$ - Combined (Instructions & data)




* If CPU needs an instruction first $L_1$:I is checked, then $L_2$ and later main memory

Silly if CPU needs data then

$L_1$:D ⟷ $L_2$ ⟷ MM
Combined

So while accessing instruction, data cache (D-cache) is not involved. Silly while accessing data, instruction cache is not involved.

② Physically indexed cache (while solving ques, this is taken by default)

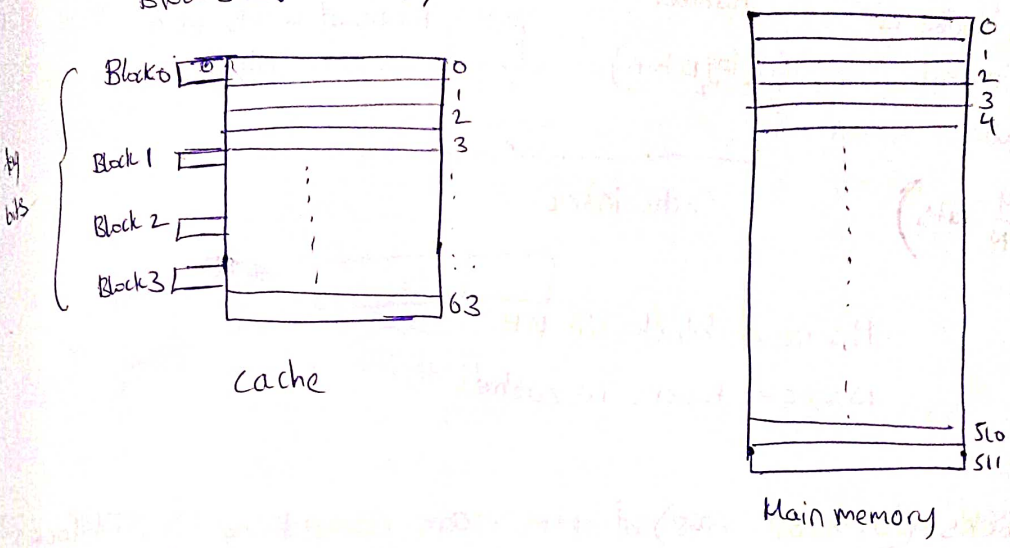
→ Main memory address is used in address mapping. (Physical address)

→ Cache is refered using physical address.

3) Logical Indexed cache / Virtually Indexed Cache:

* This is TLB (Translation look ahead Buffer)
* Virtual address is used to refer the cache.

Eg: Cache size: 64 words ; Main memory size: 512 words;
Block size: 16 words;




cache Main memory

∴ no of blocks in main memory,

$$M = \frac{512}{16} = 32 \text{ blocks}$$

no of blocks in cache memory, $N = \frac{64}{16} = 4$ blocks

Every block has tag bits.

Here we have four tags such tags

Direct Mapping:

→ Block 'k' of MM has to be placed in (k mod N) cache block

N → no of blocks in the cache

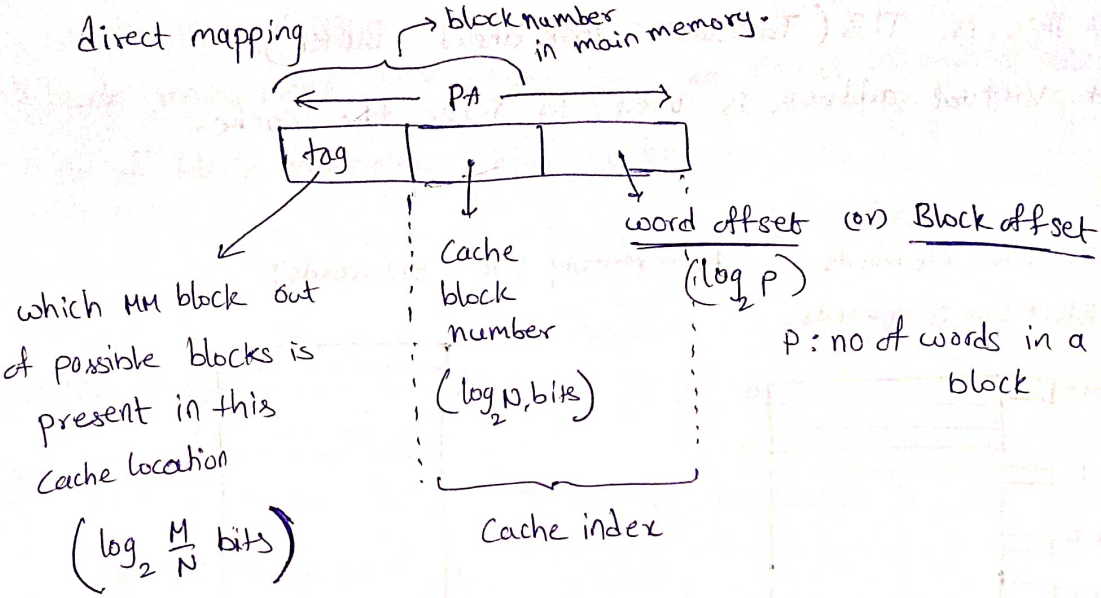
Eg: M=32, N=4;

Now consider 7th MM block

Now 7 mod 4 = 3

∴ 7th blocke of MM has to be placed in 3rd block of cache.

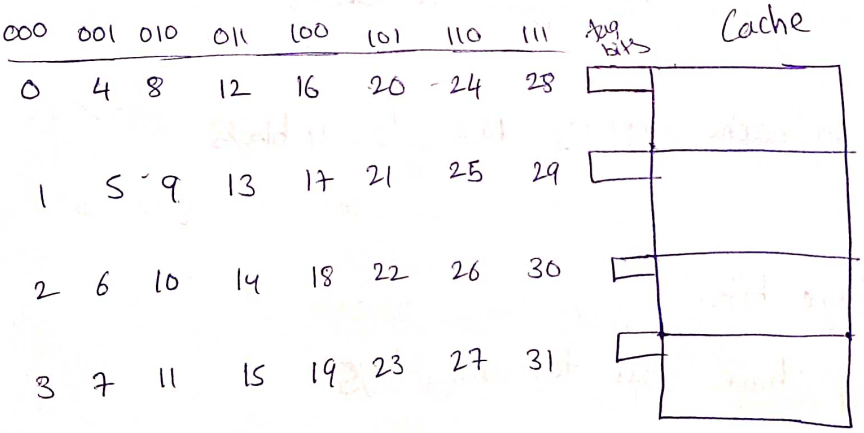
→ Here, physical address is partitioned into 3 fields with direct mapping




M: no of blocks in MM

N: no of blocks in cache

→ The blocks (0, 4, 8, ... 28) of MM are competing for block 0 of the cache.




* 8 blocks of MM are competing for each block of cache

∴ Size of tag bits = $\log_2 8 = 3$ bits

Eg: If the content in tages is $B_1$ $B_2$ $B_3$ $B_4$ 000, 100, 001, 100 find the blocks of MM that are present in the cache.

Sol:

$B_1$ — 000 ⟹ 0th block

$B_2$ — 100

The most significant 5 bits will be

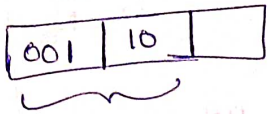
100 01

tag ↓ ∴ B1 => remainder = 1

$\therefore (10001)_2 = 17$

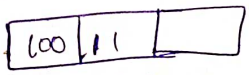
$B_2 - 001$

⇒




6th block

$B_3 - 100$




19th block

(Q22) In a direct mapped cache, the no of tag bits for a cache block is 5. what does it denote?

Sol:

→ tag bits denote no of MM block competing for a cache block.

→ since we have 5 bits in a tag

there must be $2^5$ MM blocks competing for a cache block.

It means for every block of cache we have 32 MM blocks.

i.e., If M is no of MM blocks and N is no of cache memory blocks then

$M/N = 32$

Note:

→ The MM blocks having same (K Mod N) value can not be placed together in cache inspite of cache blocks being free (this is known as conflict problem)

→ Direct mapping requires least number of TAG bits and require only one tag comparator.

(tag comparator is used to check whether the addressed word is present in the cache or not)

$$\text{no of tag bits} = \log_2 \frac{\text{MM capacity}}{\text{Cache capacity}} = \log_2 \frac{M}{N}$$

M → no of blocks in MM

N → no of blocks in cache

Q23 Main memory size = 8 MB

Cache memory size = 16 kB

Block size = 128 bytes

Direct mapping is used. Find no of tag bits.

Sol:

$$\log_2 \frac{8\,MB}{16\,kB} = \log_2 \frac{2^{23}}{2^{14}} = \log_2 2^9 = 9 \text{ tag bits.}$$

Q24 MM size : 4 GB

Block size : 32 bytes

word size : 16 bits = 2 bytes

no of lines in cache = 512 lines

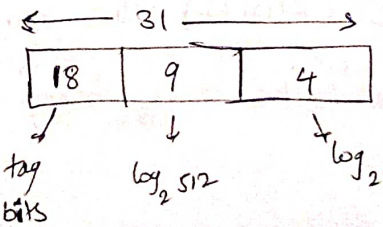
Assuming direct mapped cache

Find size of physical address & size of cache memory.

Sol:

Size of cache memory = $512 * 32 = 2^{14} = 16$ kB

PA:




block size = 16 words

MM size = $2^{32}$ bytes

$= 2^{31}$ words

(Q25) In a system design, direct mapped cache is having $2^n$ words and main memory is having 'm' bit physical address. The block contain $2^p$ words and each word contain $2^q$ bytes. The no of tag bits are

Sol:

$$\text{no of tage bit} = \log_2 \frac{\text{MM capacity}}{\text{Cache capacity}}$$

$$= \log_2 \frac{2^M}{2^n}$$

$\therefore$ m-n tag bits

(Q26) byte addressable MM $- 2^{20}$ bytes

Direct mapped cache $- 2^{12}$ blocks

blocken size $-$ 16 bytes

find tage, cache line address for the hexa decimal address

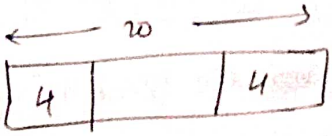
"E201F"

Sol:

$$\text{no of tage bits} = \log \frac{2^{20}}{2^{12} * 2^4} = 4$$

PA : 20 bits

$\Phi$ block offset : 4 bits




$\therefore$ no of bits for cache line = 20 - 4 - 4 = 12

$\therefore$ In hexa decimal

PA: | 1 | 3 | 1 |

$\therefore$ E, 201

# Cache Controller (or) Cache directory controller (or) tag Controller

→ The physical address is first sent to cache controller. It will check whether the addressed word is present in cache or not.

→ If present, the word is accessed from cache.
This is called cache hit.

→ If not present, the block is moved from MM to cache.
This is called cache miss.

→ Cache controller maintains one word for each cache block. Each of its word contain tag information of that cache block and status bits of the block.

→ Status bits indicate the status of the block in cache

Eg: valid bit
use bit/s (recently used or not)
Modified

→ No of words in cache controller = no of blocks in the cache

## Status bits:

### valid bit:
whether that block in cache has scope or not (i.e., live/dead)

### Used bit/s:
It denotes that the block is most recently used or not. (used in LRU replacement technique).

### Modified bit:
It denotes whether the block is modified or not. (Dirty or clean). This information is used with write Back updation.

(21) A direct mapped 32 kB cache maintain 32 byte blocks. physical address is 32-bits. Cache controller maintains one valid bits, 2 use bits and 1 modified bit along with the tag information for each cache block. what is the size of cace controller (in bits)

Sol:

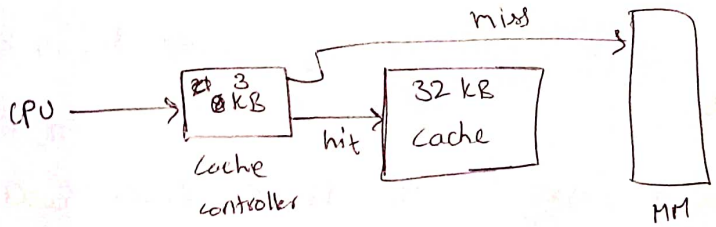
$$\text{no of tag bits} = \log \frac{2^{32}}{2^{15}} = 17$$

$$\text{no of blocks in cache} = \frac{2^{15}}{2^5} = 2^{10}$$

size of a word in cache controller = 17 + 4 = 21

$\therefore$ size of cache controller = $2^{10} \times 21$ bits

= 21 k bits ≅ 3 kB

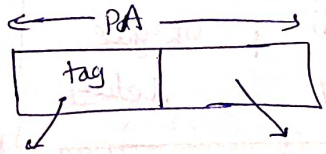



## 2. Associate Mapping:

→ Any block of the MM can be placed in any block of the cache.

→ Thus there is no conflict miss problem with associative mapping.

→ Physical address will be made two partitions.



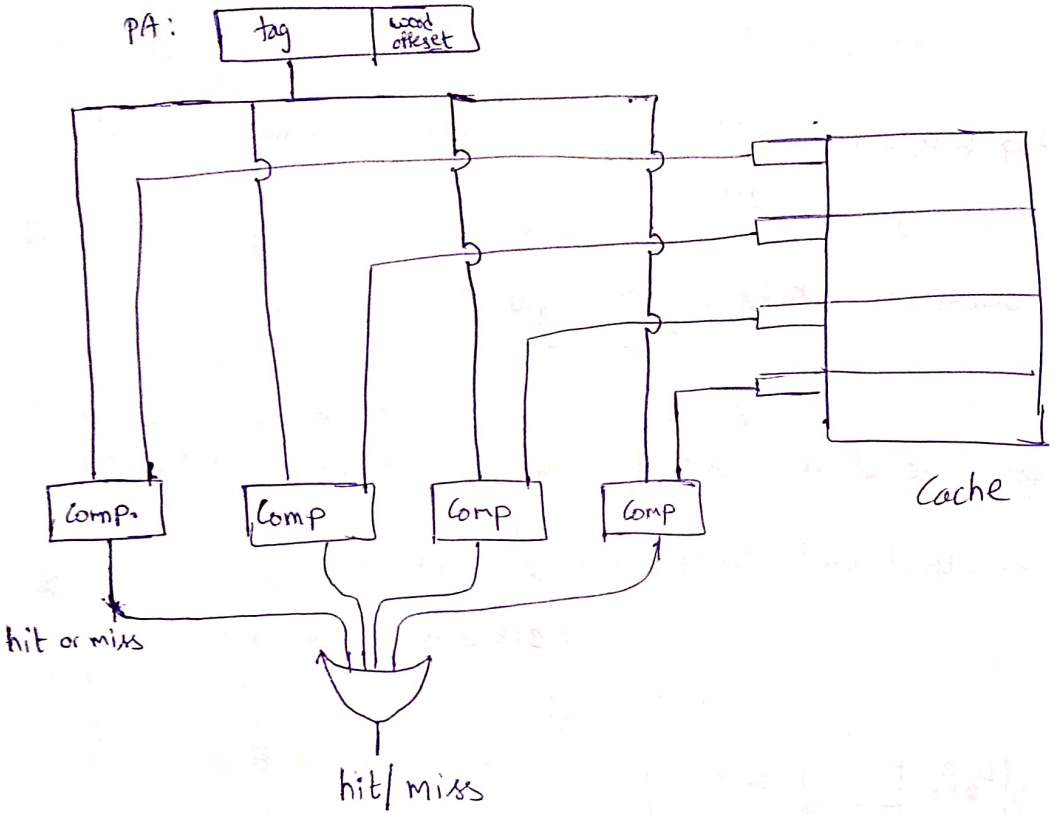

$\log_2 M$ — word offset (or) block offset

M: no of blocks in main memory.

→ Thus, associative mapping has more tag bits and requires N-tag comparators.

(i.e., Each cache blocks has to be checked to know if a hit or a miss has occured)

→ Thus, size of cache controller is also more (∵ more tag bits)




Hit latency:

→ the amount of time required, after providing address, to known ~~when~~ whether a cache hit or miss has occured is called hit latency.

∴ In ~~set~~ associative mapping,

hit latency = comparator delay + OR gate delay

→ Associative mapping requires more hardware.

## Set Associative Mapping

→ Here cache blocks are divided into sets (logically)

Eg: 2-way set associative ⟶ 2 cache blocks/set

k-way set associative ⟶ k cache blocks/set

Thus we will have N/k sets

→ Now,

k=1 ⟹ direct mapping (1-way set associative)

k=N ⟹ Associate mapping (N-way " " )

→ set associate mapping follow direct mapping as well as associate mapping.

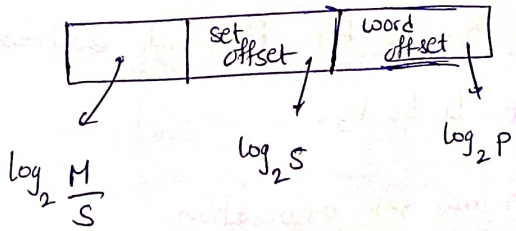
→ let no of sets, $S = \frac{N}{k}$

→ Now block 'x' of MM is placed is in

(x mod S) set of cache

→ within the set the block may be place anywhere.

→ Physical address:




M: no of blocks in MM

S: no of sets

P: word offset or block offset.

→

tag bits in k-way set associative = $\log_2 M/S$

= tag bits in direct mapping + $\log_2 k$

Proof:

no of tag bits = $\log_2 M/S$ $\qquad S = N/k$

$= \log_2 M/(\frac{N}{k})$

$= \log_2 \frac{Mk}{N}$

$= \log_2 \frac{M}{N} + \log_2 k$

= no of tag bits in direct mapping $+ \log_2 k$

→ To know hit/miss for a main memory reference only one cache set has to be searched. However all blocks of that set have to be searched.

∴ no of tag comparators = $k$ $\qquad (1 \le k \le N)$

(k-way set association)

→ Although conflict problem is present in set associative it is less. Also the hardware is of medium cost.

30/08/20

(Q27) Given the following designs for 32-bit physical address and 32kB cache with respective tag bits $t_1, t_2, t_3$.

S1: 32 byte blocks with 2 way set association.

S2: 16 byte blocks with 4 way set association

S3: 8 byte blocks with 8 way set association

Find value of expression $2^* t_2 - t_3 - t_1$

Sol:

$t_1 = \log \frac{2^{32}}{2^{15}} + \log_2 2 = 17 + 1 = 18$

$t_2 = \log_2 \frac{2^{32}}{2^{15}} + \log_2 4$

$= 17 + 2 = 19$

$t_3 = 17 + 3 = 20$

$\therefore 2(19) - 18 - 28$

$= 0$

(48) 1MB cache with 64 byte blocks is organized with 16-way set association. The virtual memory is having 8KB pages. Minimum no of page colors required such that synonym pages mapped to same cache sets will have same color.

a) 1 b) 2 c) 4 d) 8

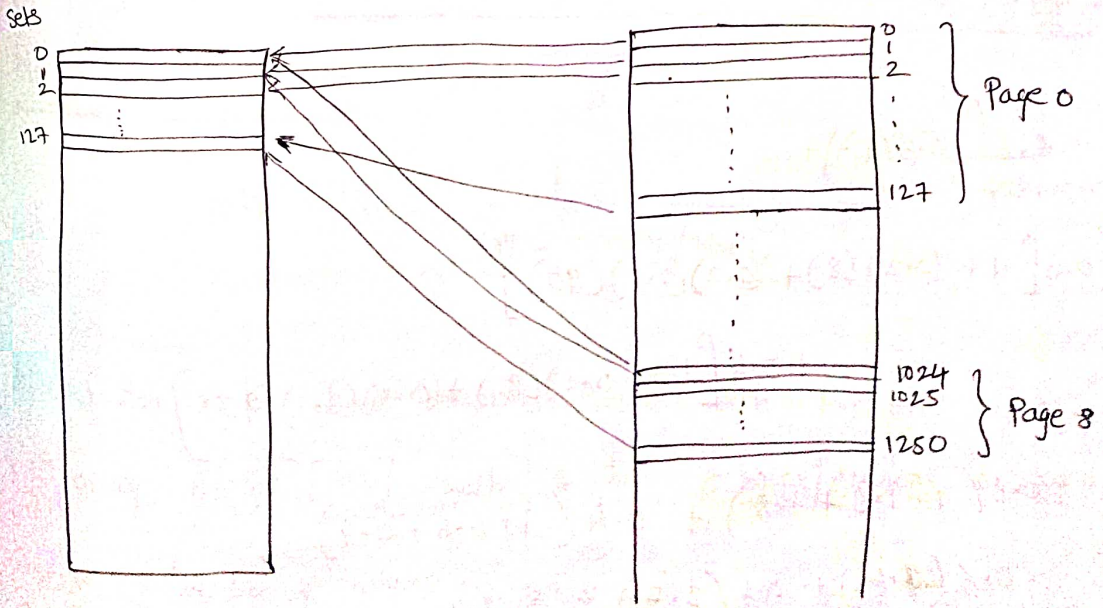
Sol:

no of cache blocks $= \frac{2^{20}}{2^6} = 2^{14}$

no of sets $= \frac{2^{14}}{16} = 2^{10}$

size of each set $= \frac{2^{20}}{2^{10}} = 2^{10} = 1KB$

no of blocks per page $= \frac{2^{13}}{2^6} = 2^7$ blocks




$\therefore$ 8 colors

## Types of misses in cache:

**Conflict miss:**

Miss that occurs even after having free blocks.

Cache association ↑ ⇒ conflict miss ↓

**Capacity miss:**

Miss that occurs when all blocks are full.

Cache size ↑ ⇒ capacity miss ↓

**Compulsory miss:**

Miss that occurs for the first time when the block is referred.

Cache Block size ↑ ⇒ compulsary miss ↓

(Q29) In a memory hierarchy the hit ratios and access times are given below

| Memory | Access time | Hit |
|---|---|---|
| L1 - Instruction cache | 1ns | 0.8 |
| L1 - Data cache | 1ns | 0.9 |
| L2 - Combined cache | 8ns | 0.9 |
| MM | 90ns | 1.0 |

If there are 60% instruction fetch operations and 40% data fetch operations what is the avg access time

Sol:

~~0.6[ 0.8(1)~~ +

$$0.6\left[1+(0.2)(8)+(0.2)(0.1)(90)\right]$$

$$+0.4\left[1+(0.1)(8)+(0.1)(0.1)(90)\right]$$

$$=0.6(1+1.6+1.8)+0.4(1+0.8+0.9)$$

$$=0.6(4.4)+0.4(2.7)=3.72$$

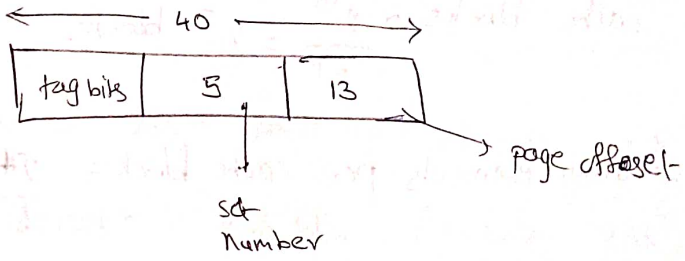
(20) 40 bit virtual address system of 8 KB pages maintains 128 word entry TLB. The TLB is having 32 sets and uses 4-way set associate mapping. Number of TAG bits of TLB is ____

Sol:

VAS = $2^{40}$ words

PS = 8KB = $2^{13}$ bytes

VA:




tag bits = 40 - 5 - 13 = 22

## Array:

* Initially cache contains no data relate to array and array is present in main memory.

* If array is stored in row major order and accessed in row major order then the performance is optimal. Sly column major order.

* If array contains fairly large no of blocks compared to no of blocks in the cache, then

$$\text{miss ratio} = \frac{1}{(\text{no of elements / block})}$$

(irrespective of the mapping technique)

Eg: A CPU has 32 KB direct mapped cache with 128 byte blocks. A 2-D array A[512][512] with 8 bytes/element is stored in MM. Consider following program

```
for(i=0; i<512; i++)
    for(j=0; j<512; j++)
        x+= A[i][j];
```

Assume i, j, x are present in processor registers.

i) what is the miss ratio?

Sol:

no of cache blocks $= \frac{2^{15}}{2^7} = 2^8$ blocks

no of array elements per cache block $= \frac{2^7}{8} = 2^4 =$ 16 elements

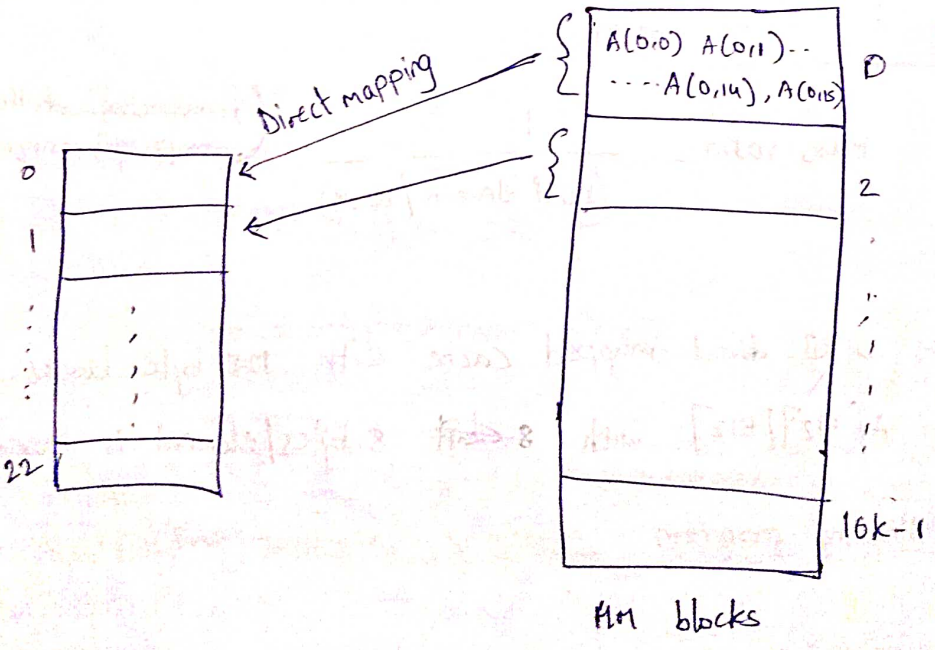
size of array $= 512 * 512 * 8 = 2^{21}$

no of array blocks in MM $= \frac{2^{21}}{2^7} = 2^{14} = 16k$ blocks

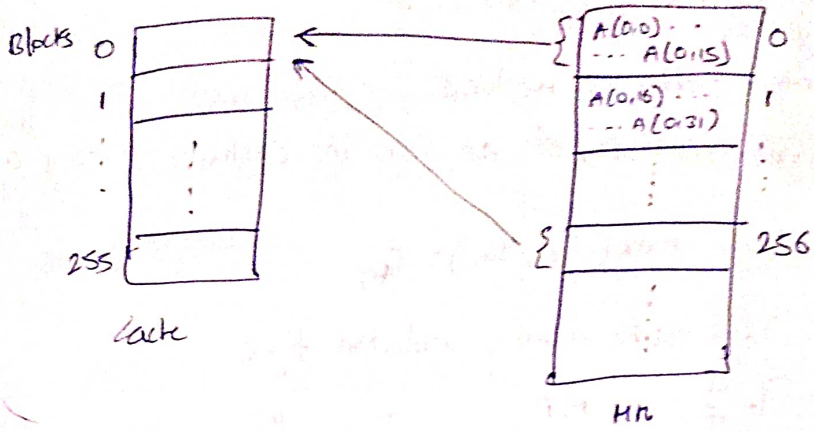
Since we have 16 elements per block and the element are in row major order, the 1st element of the block generates a miss and the references of subsequent 15 elements will be a hit.

$\therefore$ miss ratio $= 1/16$

$\Rightarrow$ hit ratio $= 15/16$



(ii) which array element replaces A(0,0) in cache memory?




the first element is 256th ~~block~~ block of array replaces A(0,0)

no of element in 256 blocks (0,1,2 ... 255)

$256 \times 16$
↳ Ele/block

$= 2^{12}$ element

$\therefore 2^{12}$th element

row number $= \frac{2^{12}}{512} = 8$

each row has 512 elee

one row $\rightarrow \frac{512 * 8}{27} = 2^5$ blocks

2nd row ends in $\rightarrow 2^6$ blocks

4th row ends in $\rightarrow 2^7$ blocks

8th $\rightarrow 2^8$ blocks

$\therefore$ A(8,0)

$\therefore$ In general for this problem

A(i,j) is replaced by A(i+8,j)

## Cache Updations

* write operations ~~are~~ result in cache coherance problem.
* So proper updation techniques are needed.
* we consider this problem in
  - (i) uni-processor system
    - write through
    - write back
  - (ii) Multiprocessor system
    - Cache snooping protocols
    - Cache directory based protocols.

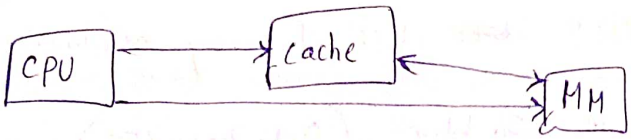
## Write through updation (uniprocessor system)

* cache & MM are updated simultaneously
* Here cache coherance problem is resolved
  i.e., all the time contents of an item in cache & memory are same.

$\therefore$ time for updation = $\max(T_c, T_m) = T_m$

$T_c$ - cache memory updation time

$T_M$ - MM " "




Advantage:

* For less no of write-, it gives better performance

Disadvantage:

* worst performance for more no of writes.

## Write-Back Updation:

→ Here MM is updated only when the concerned block is taken out of the cache.

→ As long as the block is present in the cache, the updation is not visible to MM.

→ Hence cache coherence problem is still present but with less magnitude

→ For each cache block a modified bit is used to know whether the block is updated or not.

time for updation = $h * T_c + m\left[\underbrace{(T_{Bnew} + T_c)}_{\text{clean slot (modified bit = 0) or empty block}} P + \underbrace{(1-P)(T_{Bold} + T_{Bnew} + T_c)}_{\text{Dirty slot (modified bit = 1)}}\right]$

h: hit m: miss ratio
$T_{Bnew}$: moving block from MM to cache
$T_{Bold}$: " " " cache to MM
$T_c$: write operation
P: probability of clean slot