MEMORY ORGANIZATION Unit-4

Topics will covered :

1. Basic concepts, memory hierarchy and it's need.
2. Parameters used to measure the performance
3. Types of memory components, semiconductor RAM memories.
4. Memory module design
5. ROM
6. Cache memories
7. Mapping functions
8. Replacement algorithms
9. Memory interleaving
10. Memory performance consideration.
11. virtual Memory organization

→ Both program and data are present in the memory
→ The execution speed of programs is highly dependent on the speed with which instructions and data can be transferred between the processor and the memory.
→ The memory would be fast, large, and inexpensive However it is impossible to meet all three requirements simultaneously.
→ Increased speed and size are achieved at increased cost.

Basic concept :-

→ Maximum memory size of a computer can be determined by the addressing scheme.

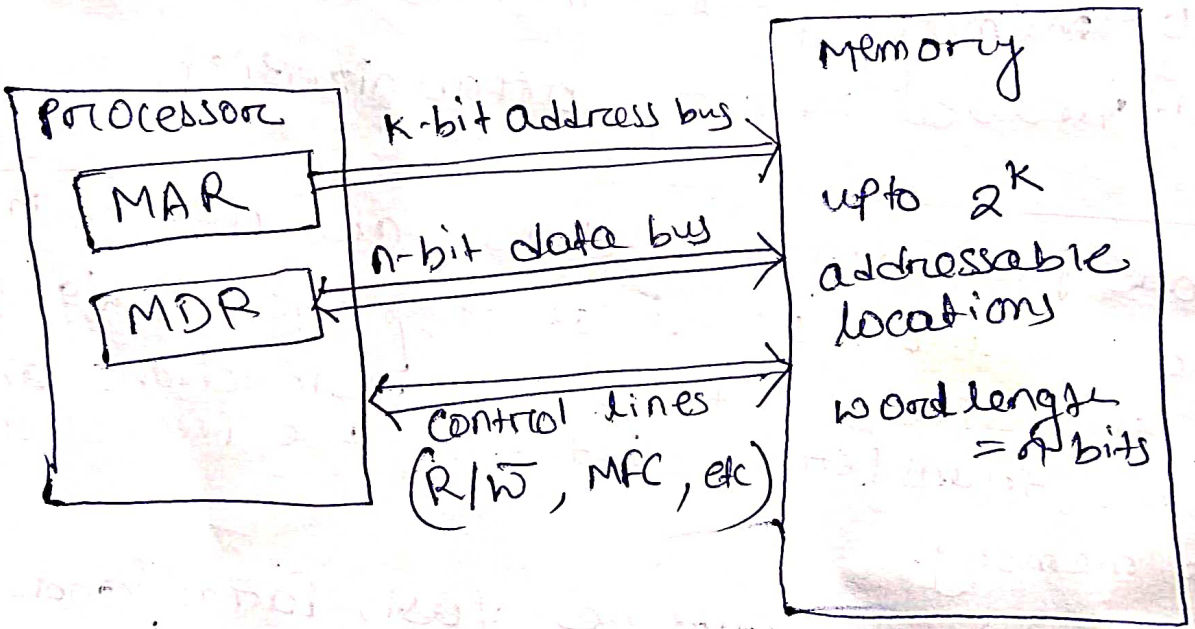
Ex :- A 16-bit computer generates 16 bit addresses is capable of addressing upto $2^{16} = 64K$ memory location.

A 32-bit computer → $2^{32} = 4G$ memory location.

A 40-bit computer → $2^{40} = 1T$ memory location.

→ The no. of locations represents size of address space of the computer.

→



Ex :- A 32-bit computer

Address bits = 32 bits.

High-order 30 bits determine which word to be accessed.

Low-order 2 bits of the address specify ~~the~~ which byte location is involved.

If MAR is k bits long and MDR is n bits long, the memory unit may contain up to 2^k addressable locations. - During a memory cycle, n bits of data are transferred between the memory and the processor.

→ The control lines Read / write (R/$\overline{W}$) and memory function complete (MFC) for coordinating data transfers.

→ The processor reads data from memory by loading the address of the required memory location into the MAR register and setting R/$\overline{W}$ line to 1. The memory responds by placing the data from the addressed location onto the data lines and confirms this action by asserting the MFC signal. After receipt of the MFC signal, the processor loads the data on the data lines into the MDR register.

→ The processor writes data into a memory location by loading the address of this location into MAR and loading the data into MDR. It indicates that a write operation is involved by setting R/$\overline{W}$ line to 0.

→ Block transfer operation → n consecutive memory address location involves in Read / write operation.

→ Memory access time :- Time elapses between the initiation of an operation and the completion of that operation. Ex: time between Read and MFC signal.

→ Memory cycle time :- minimum time delay between the initiation of two successive memory operation.

Random access memory :- if any location can be accessed for a Read or write operation in some fixed amount of time.

Cache memory :-

→ to reduce the memory access time

→ It is a fast, small memory that is inserted between the larger, slower main memory and the Processon.

→ It holds the currently active segments of a Program and their data.

Virtual Memory :-

→ is used to increase the apparent size of the physical memory.

→ CPU generates virtual or logical address (Processon)

→ Memory Management Unit : is a mapping function which maps logical address to physical address.

Internal organization of memory chips

16 x 8 organization.

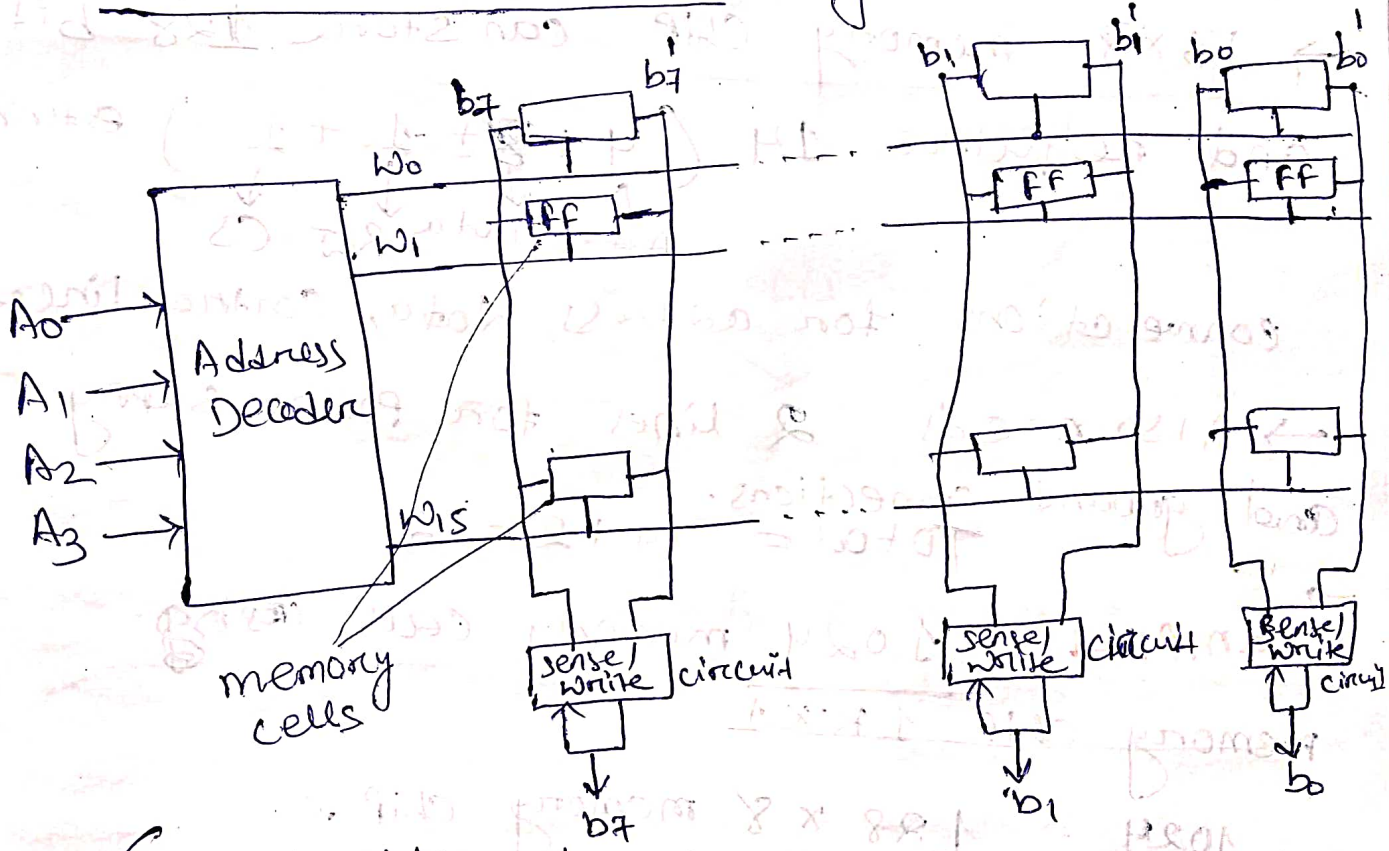
↓ ↳ each word 8 bits.

16 No. of words

→ Memory cells are usually organized in the form of an array, in which each cell is capable of storing one bit of info.

→ Each row of cells constitutes a memory word and all cells of a row are connected to a common line referred to as the word line

→ The cells in each column are connected to a sense/write circuit by two bit lines.



(Organization of bit cells in a memory chip)

→ The sense/write circuits are connected to the data input/output lines of the chip.

→ During a Read operation, these circuits sense or read, the information stored in the cells selected by a word line and transmit this information to the output data lines.

→ During a write operation, the sense/write circuits receive input information and store it in the cells of the selected word.

Example:- 16 x 8 memory chip organization.

→ Two control lines, R/$\overline{W}$ and CS are provided in addition to address and data.

→ The R/$\overline{W}$ (Read/$\overline{Write}$) input specifies the required operation.

→ CS (Chip select) input selects a given chip in a multichip memory system.

∴ → 16 x 8 memory chip can store 128 bits → 128 memory cells and require 14 (4 + 8 + 1 + 1) external connections for address, data, control lines.

(4 → Address, 8 → Data, 1 → R/$\overline{W}$, 1 → CS)

→ Also needs 2 lines for power supply and ground connections.

→ Total = 14 + 2 = 16

Example:- 1024 memory cells

~~Memory chip 1K x 1~~

1024 = 128 x 8 memory chip.

External connection = 7 + 8 + 2 + 2 = 19.

1024 = 1024 x 1

= 1K x 1 memory chip.

External connection = 10 + 1 + 4 = 15.

1K x 1, 128 x 8, 64 x 16

1024, $2^7 * 2^3$ = 1024, $2^6 * 2^4$ = 1024

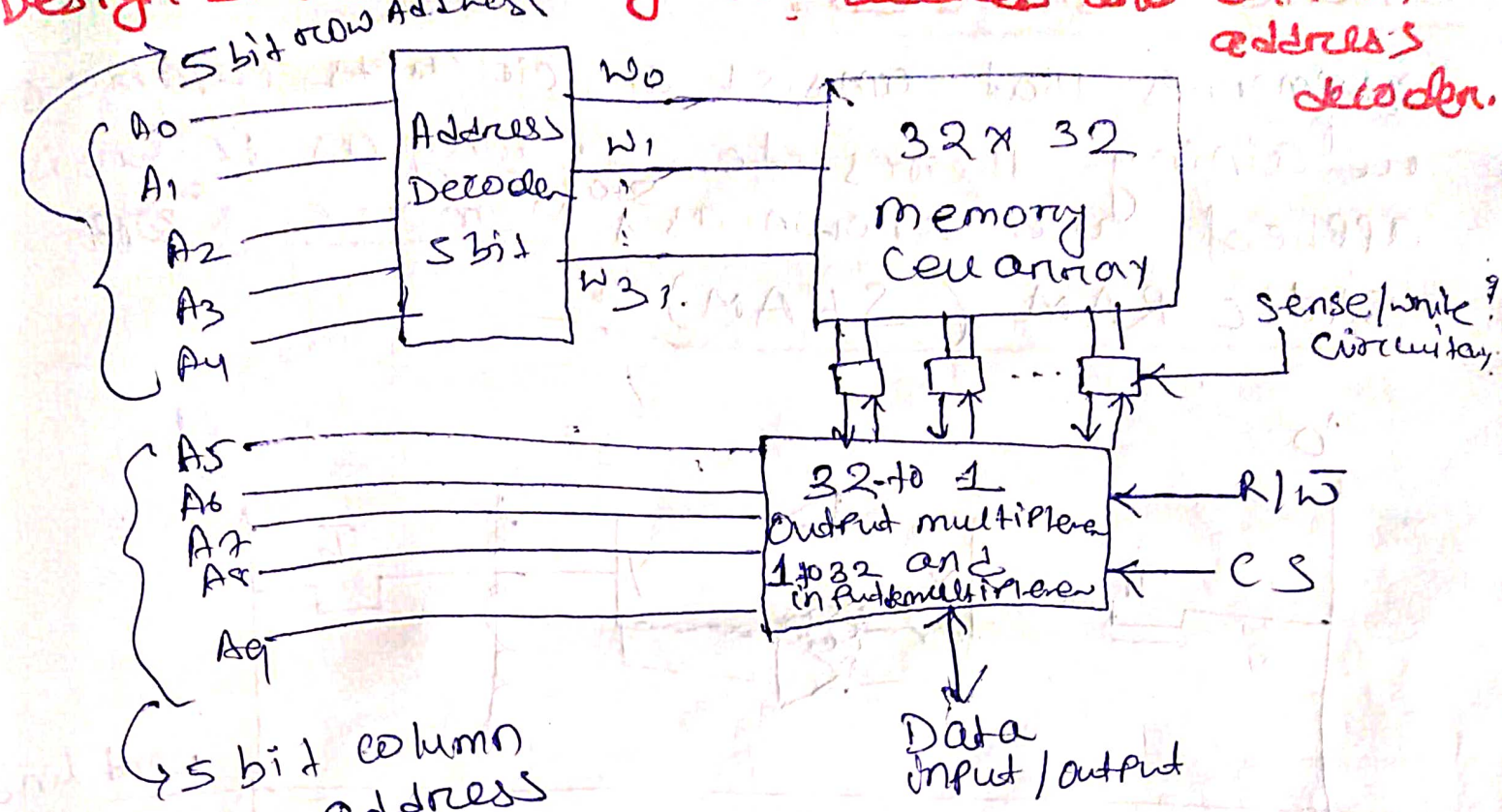
1024 memory cell ←

1K x 1 → 15

128 x 8 → 19

64 x 16 → 6 + 16 + 4 = 26

Design 1K x 1 by using row address and column address decoder.



$1024 = 2^{10}$

10 bit → 5 bit (row), 5 bit (column)

→ A row address selects a row of 32 cells, all of which are accessed in parallel. However according to the column address, only one of these cells is connected to the external data line by the output multiplexer and input demultiplexer.

Ex :- 4M bit chip

$= 4 * 2^{20} = 2^{22}$

$= 512K \times 8$

$= 2^{19} * 2^3 = 2^{22}$

→ 4M bit chip using 512K x 8 organization.

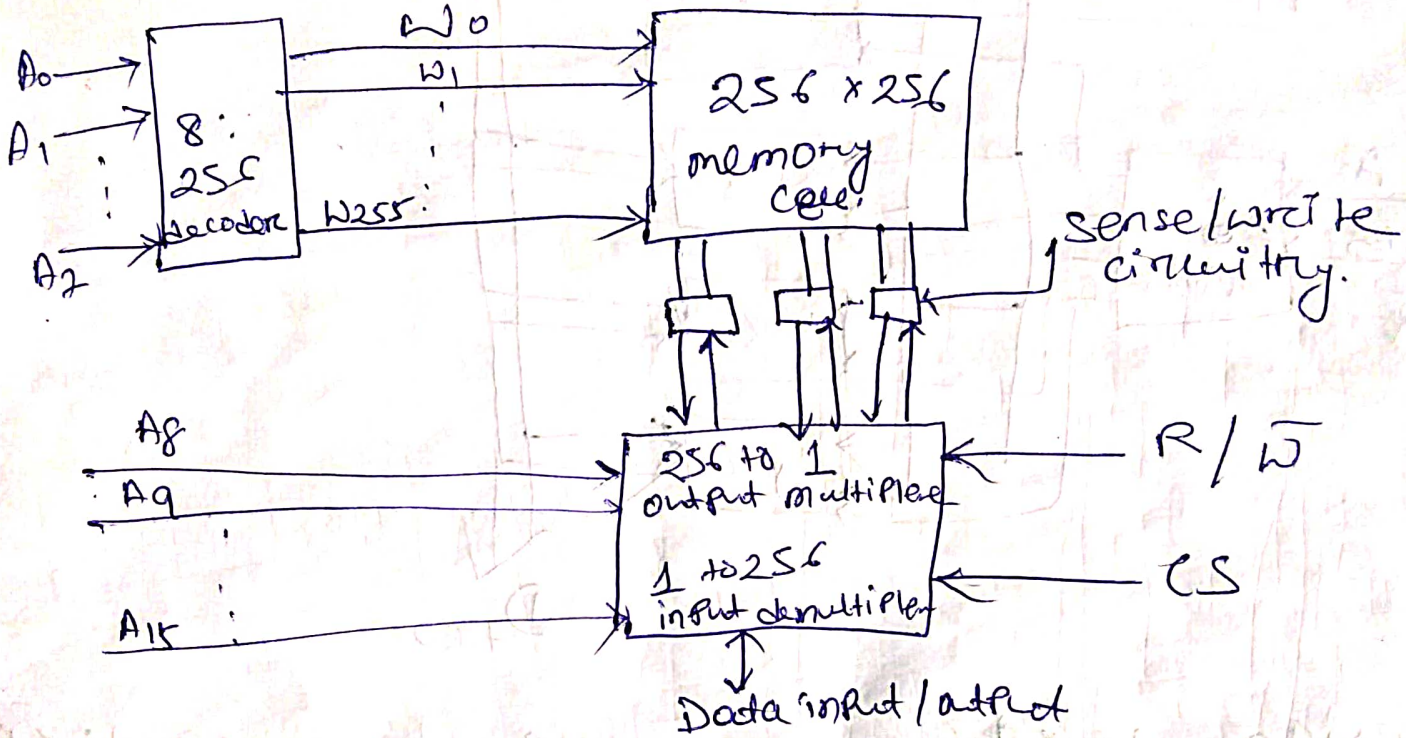
Total External connection = 19 + 8 + 4 = 31

Address bus = 19, Data bus = 8.

Design 64K×1 using Row and column address decoder.

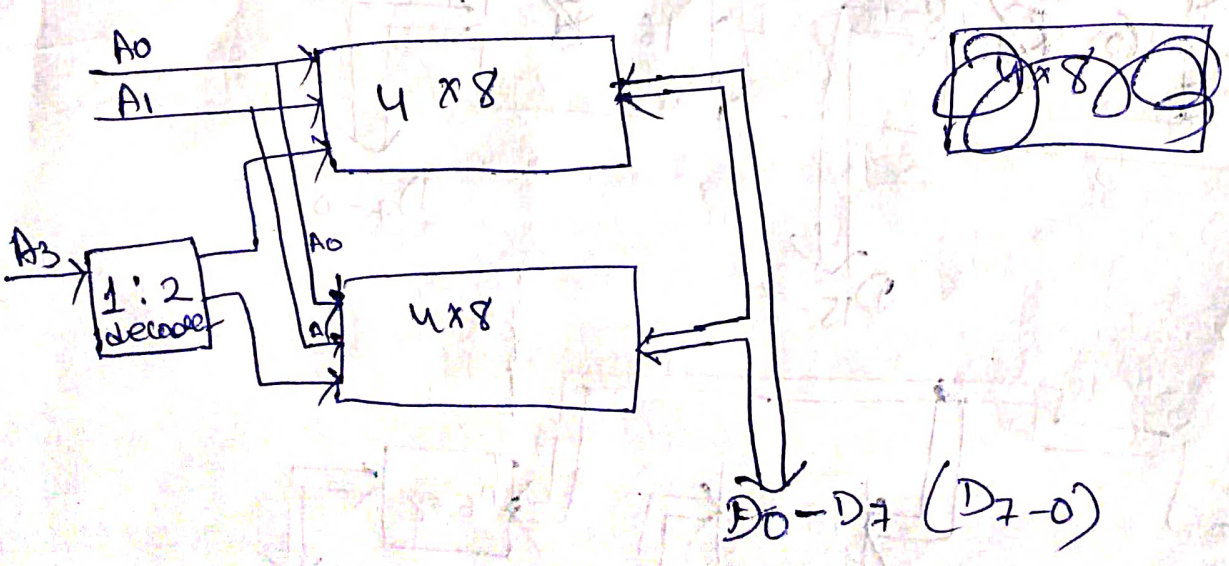
Address line = 16

8 Row address, 8 column address



Design 8×8 memory chip by using 4×8 memory chip.

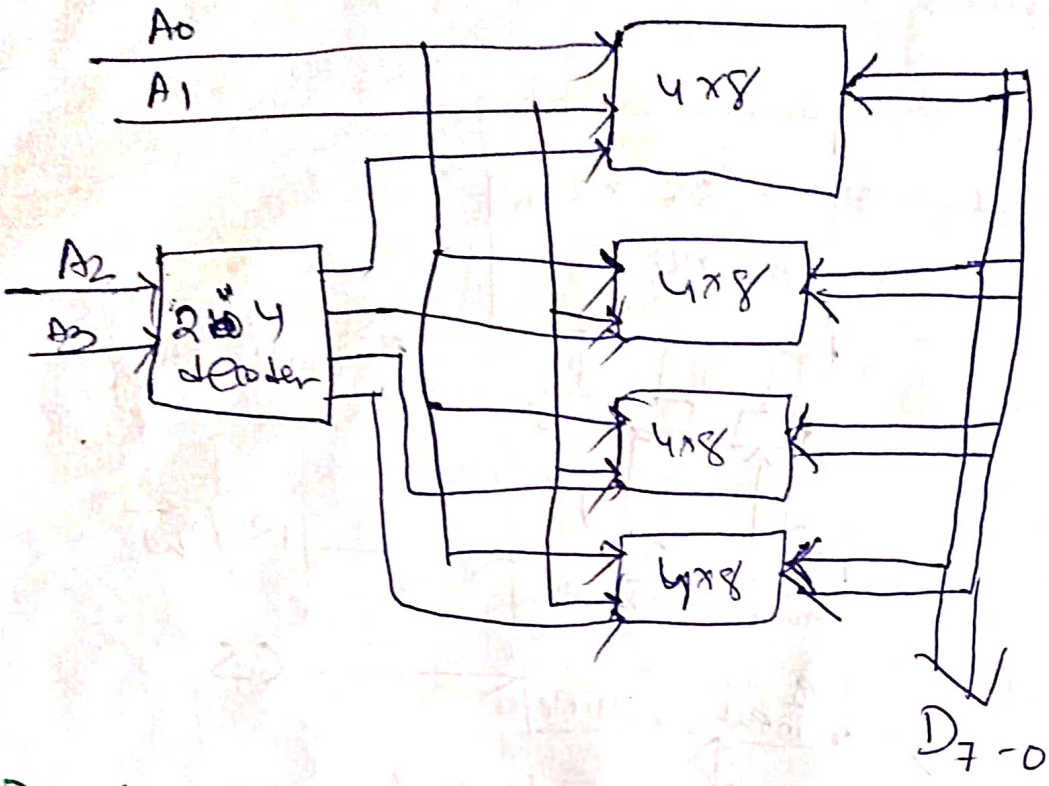
No. of Chip = $\frac{8 \times 8}{4 \times 8} = 2$



→ Ans:-

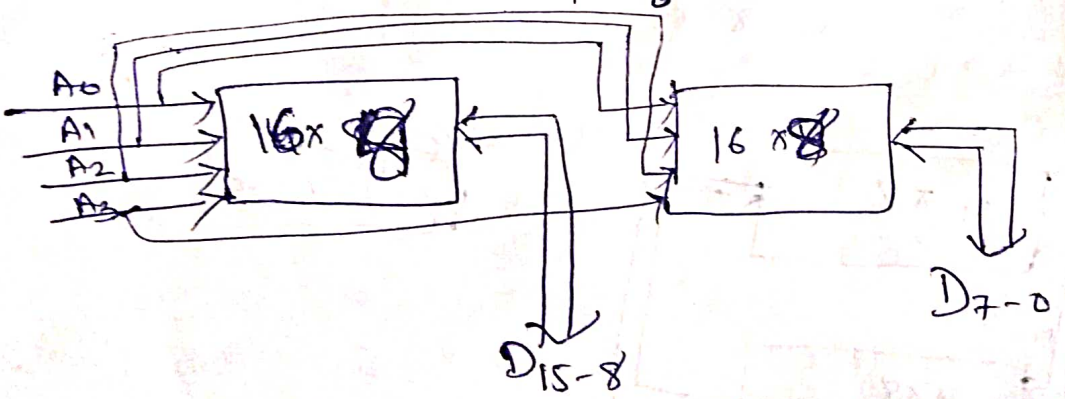
Design 16x8 memory chip by using 4x8 memory chip

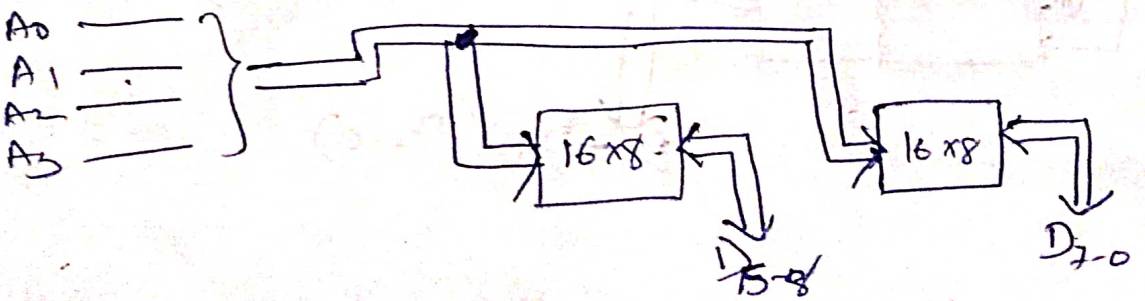
$$\text{No. of Chip} = \frac{16 \times 8}{4 \times 8} = 4$$



Design 16x16 memory chip by using 16x8 memory chip

$$\text{No. of Chip} = \frac{16 \times 16}{16 \times 8} = 2$$

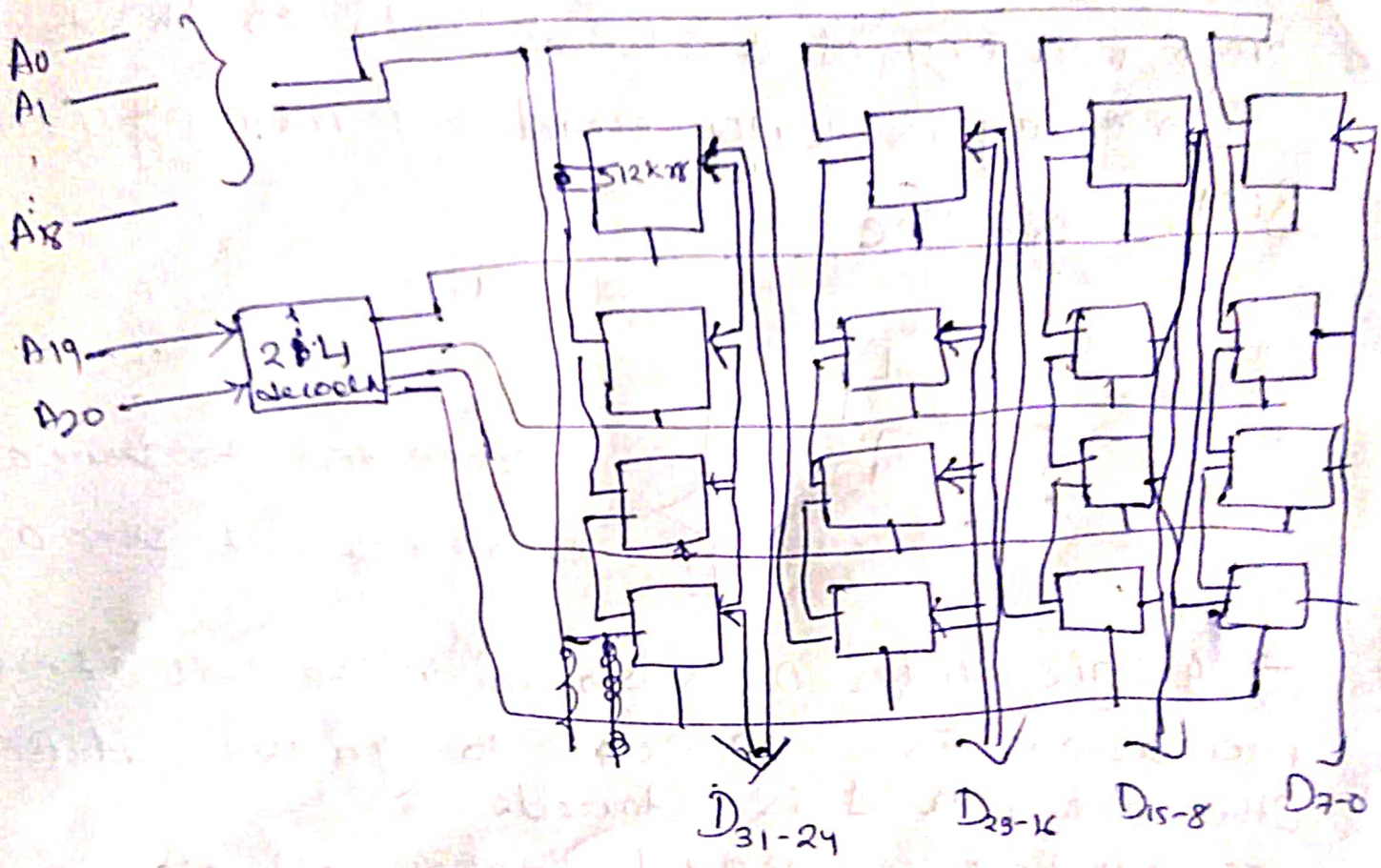




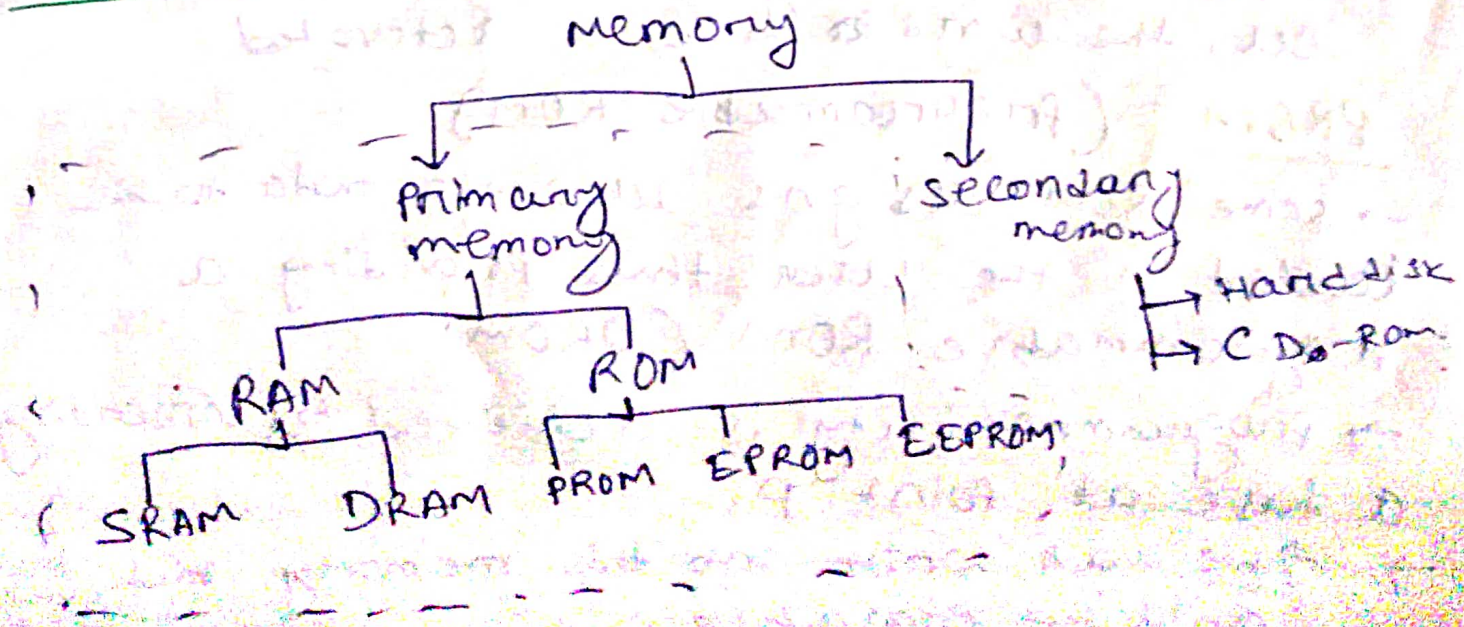
Design a 2M × 32 memory chip using 512K×8 memory chip

$$\text{No. of Chips} = \frac{2M \times 32}{512K \times 8} = \frac{2^{21} \times 2^5}{2^{19} \times 2^3}$$

$$= \underbrace{2^2}_{\text{in column}} \times \underbrace{2^2}_{\text{in a row}} =$$

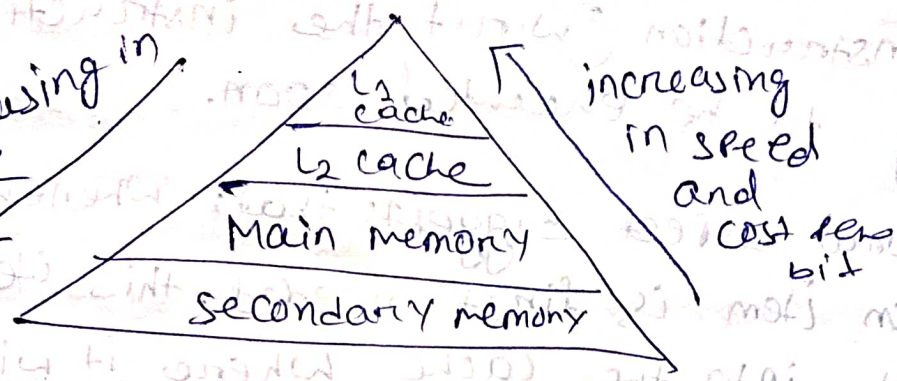


Classification of memory :—



CACHE MEMORY :-

Memory Hierarchy :-



Why cache ?

→ The processor spends more time to access data and instruction from main memory. A cache memory is introduced in between CPU and main memory to improve average access time. It is a small and fast storage unit

Locality of reference :-

→ Analysis of program shows that most of their execution time is spent on routines in which instructions are executed repeatedly. (Simple loop, nested loop).

→ Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently. This is referred to as locality of reference.

→ It manifests itself in two ways : temporal and spatial.

Temporal :

→ A recently executed instruction is likely to be executed again very soon.

Spatial :-

→ Instructions in close proximity to a recently executed instruction (w.r.t the instruction address) are likely to be executed soon.

→ The temporal aspect suggests that whenever an information item is first needed, this item should be brought into the cache where it will hopefully remain until it is needed again.

→ The spatial aspect suggests that instead of fetching just one item from the main memory to cache memory, it is useful to fetch several items that reside at adjacent address as well.

→ BLOCK refers to a set of contiguous address locations of some size. A cache block is also called cache line

Read Request and Write Request :-

→ When a read request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time.

→ Subsequently, when the program references any of the locations in this block, the desired contents are read directly from the cache.

→ The cache memory can contain less no. of blocks compared to the total no. of blocks in Main memory.

→ A mapping function specifies the correspondence between the main memory blocks and cache memory blocks.

→ When a cache is full and the requested memory word is not present in the cache, the cache control hardware must decide which block should be removed to create space for the new block. This decision is taking by replacement algorithm.

→ The cache control circuit determines whether the requested word currently exists in the cache. If it does, the read or write operation is performed on the appropriate cache location. In this case, read or write hit is occured.

→ In read operation, the main memory is not involved.

→ In write operation, the system can proceed in two ways ← Write through protocol, Write Back protocol } Cache validation technique.

→ In write through protocol, the cache location and main memory location are updated simultaneously.

→ In write back protocol, the cache location is updated only and marked it as updated with an associated flag bit, called dirty or modified bit.

- The main memory location is updated later, when that block is selected for replacement. This protocol is also called copy-back protocol.

→ Write through protocol is simpler but it results in unnecessary write operations in the main memory when a given cache word is updated several times during its cache residency.

→ Write back protocol also results unnecessary write operation because when a cache block is written back to the memory all words of the block are written back, even if only a single word has been changed while the block is in cache.

→ When the addressed word in a Read operation is not in the cache, a read miss occurs. The block of words that contains the requested word is copied from the MM into the cache. After the entire block is loaded in to the cache, the particular word requested is forwarded into the processor.

→ Alternatively, this word may sent to the processor as soon as it is read from MM. This approach is called load-through or early restard, reduces the processor's waiting period.

→ During write operation, if the desired address is not present in the cache, a write miss occurs. In write through protocol, directly written into the MM and in write back, 1st bring into cache then update it

Hit rate and Miss penalty :—

→ A successful access to data in a cache is called a cache hit.

→ An unsuccessful access to the data in a cache is called cache miss.

→ The no. of cache hit stated as a fraction of all attempted accesses is called the hit rate.

→ The no. of cache miss stated as a fraction of all attempted accesses is called the miss rate.

→ When the cache miss occured, the information is not present in cache. So it take extra time to bring the data from main memory to cache memory is called miss penalty.

$$\text{Hit rate} = \frac{\text{No. of successful access}}{\text{Total no. of attempted access}}$$

$$\text{Miss rate} = \frac{\text{No. of unsuccessful access}}{\text{Total no. of attempted access.}}$$

Average access time (Tavg)

$$\boxed{T_{avg} = h * C + (1-h) M}$$ or $h * T_C + (1-h) * T_m$

where h = hit rate

C = cache hit time

M = Miss penalty

$$\text{Improve in performance} = \frac{\text{Time without cache}}{\text{Time with cache.}}$$

Questions

① What is the hit ratio if MM access time is 150 nsec and cache memory access time is 30 nsec and average access time is 42 nsec.

Ans:- $h = ?$ $T_{avg} = h * C + (1-h) * m$

$\Rightarrow 42 = h * 30 + (1-h) * 150$

$\Rightarrow 42 = 30h + 150 - 150h$

$\Rightarrow 120h = 150 - 42$

$\Rightarrow h = \frac{108}{120} = .9$

$= \frac{108}{120} = .9$

② Hit rate of instruction is 95%, Hit rate of data is 90%. Assume that from 130 fetch, 100 are instr fetch and 30 are data fetch. Find the improvement in performance if without cache is 10 cycle needed for read access and with cache 1 cycle for read access in case of hit and 17 cycles for read access in case of miss.

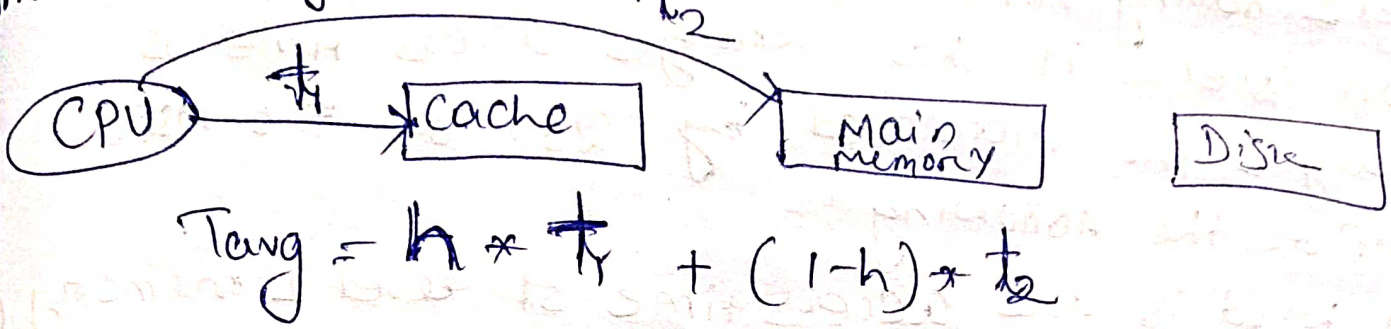
Ans

$$\text{performance} = \frac{130 * 10}{\text{Time with cache}} = 5.038$$

T_{avg} → T_{avg} for data + T_{avg} for instr

$= 30 * (.9 * 1 + .1 * 17) + (.95 * 1 + .05 * 17) * 100$

Case-1: Processor having access to both the levels simultaneously.



$$T_{avg} = h * t_1 + (1-h) * t_2$$

Case-2:- Processor is having access to only level 1.

$$T_{avg} = h * t_1 + (1-h)(t_1 + t_2)$$

Case-3:- ~~process~~ When cache miss, a block of data is transferred.

$$T_{avg} = h * t_1 + (1-h)(t_1 + t_B)$$

t_B : time for block of data

Multi level cache:-

$$T_{avg} = h_1 * t_1 + (1-h_1) h_2 * t_2 + (1-h_1)(1-h_2) * m$$

where h_1 : hit rate in L_1 cache

h_2 : " " L_2 "

t_1 : access time of L_1

t_2 : access time of L_2

m : access time of main memory.

Q. Consider a two-level memory system such that level 1 having hit ratio 0.8 and 5 times faster than level 2 if the average access time is changed or increased by 20% of 50 nsec. Compute the following :-

(i) What is the access time of level 1 memory (t_1)

(ii) What is the new hit ratio (hnew)

(iii) What is the % in change in hit ratio.

Ans :- $h_1 = 0.8$, $t_1 = \frac{t_2}{5}$

$T_{avg1} = 50$ nsec.

$T_{avg\,new} = 50 + \frac{20}{100} * 50 = 50 + 10 = 60$ nsec.

$T_{avg} = h_1 * t_1 + (1-h_1) t_2$

$50 = 0.8 * t_1 + (.2) * 5t_1$

$\Rightarrow 50 = 0.8 * t_1 + t_1$

$\Rightarrow t_1 (1.8) = 50 \Rightarrow t_1 = \frac{50}{1.8} = 27.77$ nsec (i)

$T_{avg\,new} = h_{new} * t_1 + (1-h_{new}) * 5t_1$

$\Rightarrow 60 = x * 27.77 + (1-x) * 5 * 27.72$

$\Rightarrow 60 = x * 27.77 + 138.85 - 138.85x$

$\Rightarrow (138.85 - 27.77)x = 138.85 - 60$

$\Rightarrow x = \frac{78.85}{111.08} = 0.709$

~~hnew = 70~~ $h_{new} = 0.709$ (ii)

(iii) $h \rightarrow h_{new}$

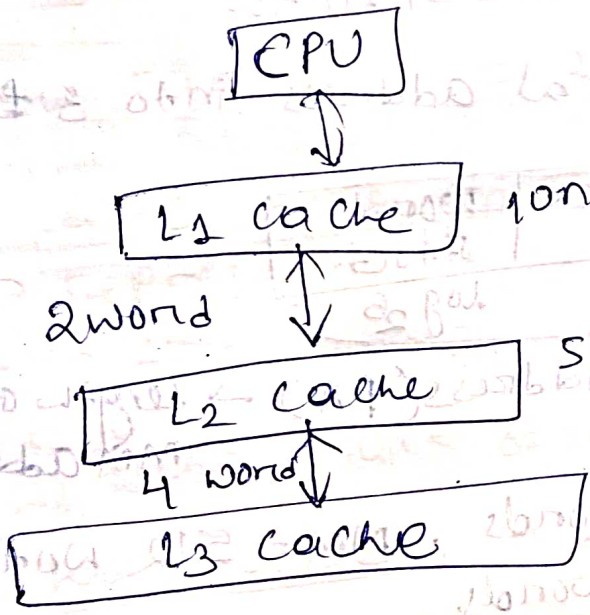
$0.8 \rightarrow 0.709$

0.71

~~0.709 = 0.8~~

$\frac{0.8 - 0.71}{0.8} = \frac{0.09}{0.8} * 100 = 11.25\%$

Q. Consider a 3-level memory system with access time on word is 10nsec, 50 nsec, 100 nsec. Hit ratios are 0.8, 0.9 and 1 if referred word is not available in L-1. Get the two word block from L-2 to L-1 and supply the desired word to the processor. if it is not available in L-2 then get a 4 word block from L_3 to L_2 and transfer the associate block to L_1 and handover the desired word to processor L_1. What is the average access time.



$t_{B_1} = 2 * 50$
$= 100$ nsec

$t_{B_2} = 4 * 100$
$= 400$ nsec

$$T_{avg} = h_1 * t_1 + (1-h_1) * h_2 (t_1 + t_{B_1}) + (1-h_1)(1-h_2)(t_1 + t_{B_1} + t_{B_2})$$

$$= 0.8 * 10 + (0.2) * 0.9 * (10 + 100) + 0.2 * 0.1 * (10 + 100 + 400)$$

$$= 8 + 0.18 * 110 + 0.02 * 510$$

$$= 8 + 19.8 + 10.2 = 38 \text{ nsec.}$$

Mapping function or Address Mapping

→ It is a technique to place main memory block into cache memory.

→ 3 type of mapping functions.

* Direct mapping
* Associative "
* set-associative "

Direct Mapping :-

→ selects $(K \bmod N)^{th}$ cache position for accomodating k^{th} main memory block and N is the no. of cache blocks / lines.

→ It divide the physical address into 3 fields

Tag	cache block offset	word offset.
$\log_2(\frac{M}{N})$	$\log_2 N$	$\log_2 B$

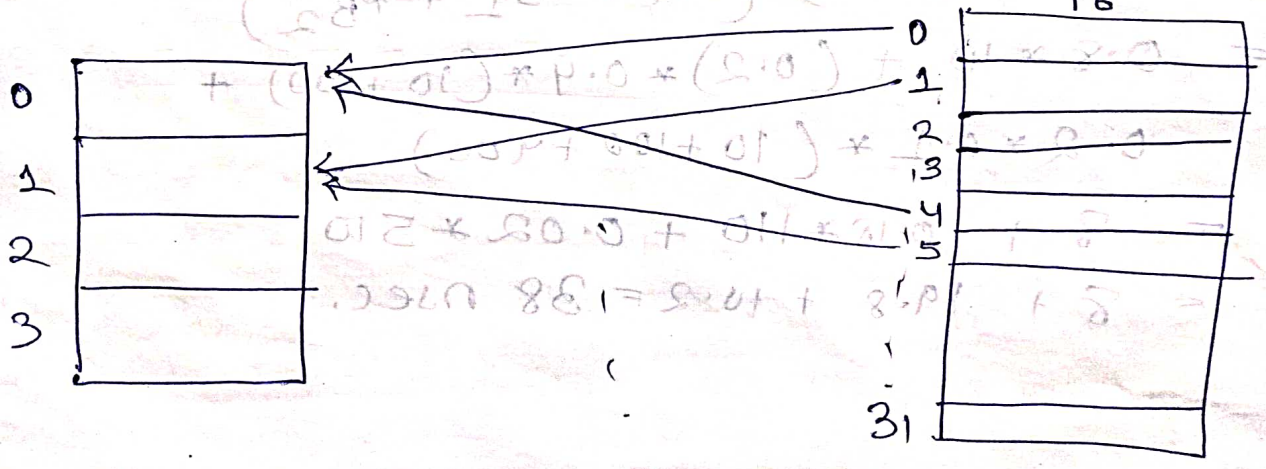
Physical Address (PA) → length of MM address

Example :- CM = 64 words, MM = 512 words.

Block size = 16 words.

No. of cache block = $\frac{64}{16} = 4$

No. of main memory blocks = $\frac{512}{16} = 32$



MM blocks maps to cache block 0 → {0, 4, 8, 12, 16, 20, 24, 28}

000 001 010 011 100 101 110 111

Total 8 MM blocks maps to one C.B = 3 bit require

1, 5, 9, 13, 17, 21, 25, 29

2, 6, 10, 14, 18, 22, ~~26~~, 30

3, 7, 11, 15, 19, 23, 27, 31

→ Tag specifies how many compitator for one cache block.

$$\frac{\text{No. of M.M block}}{\text{No. of C.M block}} = \frac{32}{4} = 8$$

⇒ 8 compitators means 3 bits require

so Tag bits = 3.

→ cache block offset is the no. of bits required to represent cache block.

In this example, 4 cache blocks means cache block offset is 2 bit.

→ word offset is the no. of bits required to represent the block size.

In this example, block size = 16 words.

so word offset is 4 bit.

3	2	4
Tag	C.B offset	Word offset

→ It consumes only one tag comparator and requires less tag memory.

Tag memory = N * Tag bits.

↳ No. of cache blocks.

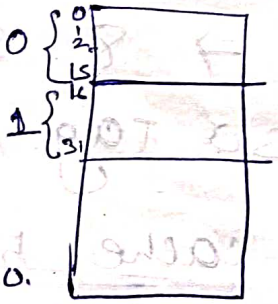
→ The conflict penalities makes more blocks movement and hence the direct mapping is the slowest mapping technique.

Q. Check whether the word 313 of main memory is present in cache or not.
(same data of previous example)

Block size = 16 words.

M.M size = 512 words

C.M size = 64 words



313 word belongs to 19th block no.

$\frac{313}{16} = 19.56$

M.M block 19 maps to $19 \% 4 = 3$

↓ No. of C.B

Cache blocks no.

Tag bit will be 100

↑ 19

(see the previous page, 19 comes under 100)

Or

$(313)_{10}$ = | 100 | 11 | 1001 |

Tag | C.B address | Word offset

$(313)_{10} = 100\,11\,1001$

Divide the bits into 3 fields (into present in previous page)

If tag bits of cache block no. 3 is 100 then 313 word of M.M is present in cache memory.

Check whether 170 word of MM is present in cache or not.

$(170)_{10}$ = | 010 | 10 | 1010 |

Tag of word of line

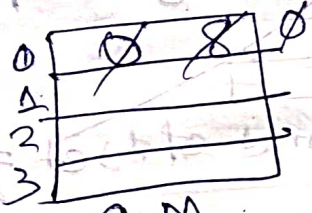
If Cache block No. 2 is 010 then 170 word of M.M is present in cache

Limitation of Direct mapping :-

→ Due to $(K \bmod N)^{th}$ rule, the direct mapping may results conflict penalities

→ Consider a cache with 4 blocks, which is initially empty has to support the main memory block references.

MM ref. : 0, 8, 0, 4, 8, 0, 8, 4, 12, 4, 12



0 4 8 . . .

Hit = 0.

Other blocks are empty still can't put block

→ It is called conflict penality and the movement of block is increase

→ TO overcome this problem, a technique is suggested called Associative mapping.

Associative mapping :-

→ Avoids the conflict penalty by relaxing the mapping rule.

→ Any block of main memory can be placed anywhere in the cache.

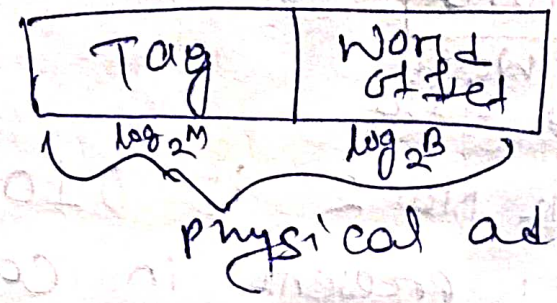
→ With the previous reference

0, 8, 0, 4, 8, 0, 8, 4, 12, 4, 12

M M H M H H H H M H H

No. of miss = 4, No. of hit = 7

→ Associative mapping divide the physical address into 2 partition.



Tag bits = $\log_2 M$ M : No. of MM blocks

word offset = $\log_2 B$ B : Block size.

Example :- C.M size = 64 words.

M.M size = 512 words.

Block size = 16 words

Determine tag bits and word offset.

word offset = $\log_2 16 = 4$ M = 512/16 = 32

Tag bits = $\log_2 512 = 9$

Tag = $\log_2 32 = 5$ word offset = $\log_2 16 = 4$

→ since the no. of tag bits are more, the associate memory required more tag memory.

→ since all the cache blocks are fetched for hit or miss, therefore Tag

No. tag comparator are required.

↳ No. of cache blocks.

Set-Associative mapping:-

→ contains the advantages of Direct mapping and associative mapping.

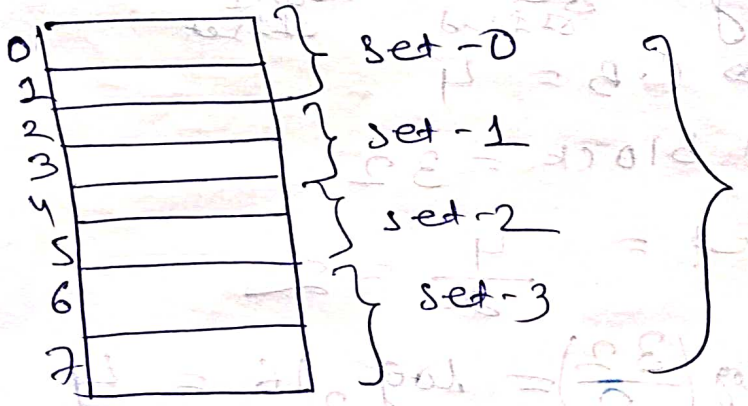
→ cache is divided into sets, and the no. of blocks per set is depends on the association used.

S = no. of set in a cache block.

$S = \frac{N}{P}$ where N: No. of cache block
P-way set associative

Ex:- 8 no. of cache block,
and 2-way set association.

No. of set = $\frac{8}{2} = 4$



→ K^th block of MM is mapped to (K mod S) set. in cache.
Within the set, it can be placed anywhere.

→ The physical address is divided into 3 partitions.

Tag	set offset	word offset

$\log_2(\frac{M}{S})$ $\log S$ $\log \frac{B}{2}$

Physical address.

→ The no. of tag comparator in set-associative mapping is the no. of set way.

if 2-way then 2 no. of tag comparator

if 4-way the 4 no. of tag comparator

Example -

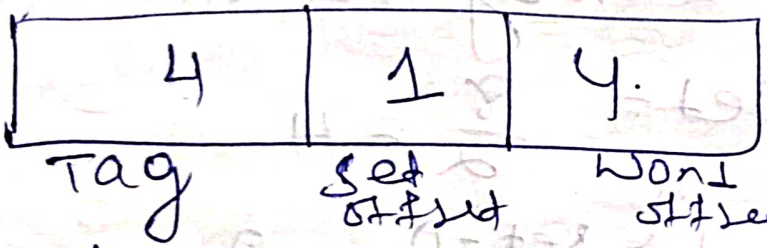
M.M = 512 word

C.M = 64 word

BS = 16 word

2-way set associative.

calculate, tag, set offset and word offset



NO of C.B = 4

No. of MM block = 32

NO. of set = $\frac{4}{2} = 2$

$$Tag = \log_2\left(\frac{32}{2}\right) = \log_2 16 = 4$$

Questions on mapping functions:

(1) A cache consists of 128 blocks of 16 words each, for a total of 2048 (2K) words. Main memory is addressable by a 16-bit address. The main memory has 64K words. Obtain Tag bits in different mapping technique.

Ans:- M.M size = 64K word, Block size = 16 words

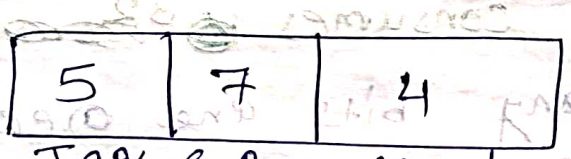
No. of MM blocks = $\frac{64K}{16} = \frac{2^{16}}{2^4} = 2^{12}$

$= 4K$

C.M size = 2048 words = 2K

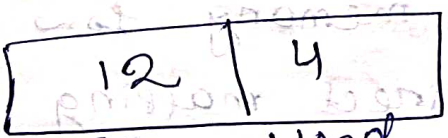
No. of cache blocks = $\frac{2048}{16} = \frac{2^{11}}{2^4} = 2^7 = 128$

Direct mapping



$\frac{4K}{128} = \frac{2^{12}}{2^7} = 2^5$

Associative mapping:-

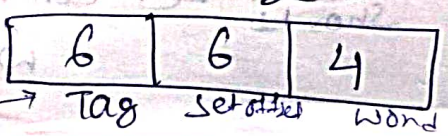


$4K = 2^{12}$

Set-associative mapping: 2-way set associative

No. of set = $\frac{128}{2} = 64$

$\frac{4K}{64} = \frac{2^{12}}{2^6} = 2^6$



Ex 2.20

(2) A cache consist of 64 blocks and MM consist of 4K blocks, each block consists of 128 words.

(a) find size of MM.

(b) find size of CM

(c) find length of main memory address

(d) how many bits are there in each of the tag in case of direct mapping.

(e) " in case of associative mapping

(f) " " " " " 4-way set "

(3) Consider 2GB main memory and 1MB cache partitioned into 256 word blocks. Each word consumes 32 bits.

(i) how many bits are needs to physical address.

(ii) how many blocks are present in MM and CM.

(iii) Identity no. of tag bit, tag comparator and tag memory for the following

(i) Direct mapping

(ii) Associative mapping

(iii) 4-way set associative mapping.

Soln :- Hit (memory should be convert into words.

Performance Consideration

Objective: Best possible performance at the lowest cost.

→ to improve performance without increasing the cost.

→ Measure is the price or performance ratio.

→ To achieve high performance, Parallelism is used in slower unit (memory). This Parallelism can be ~~achieved~~ organized by memory interleaving.

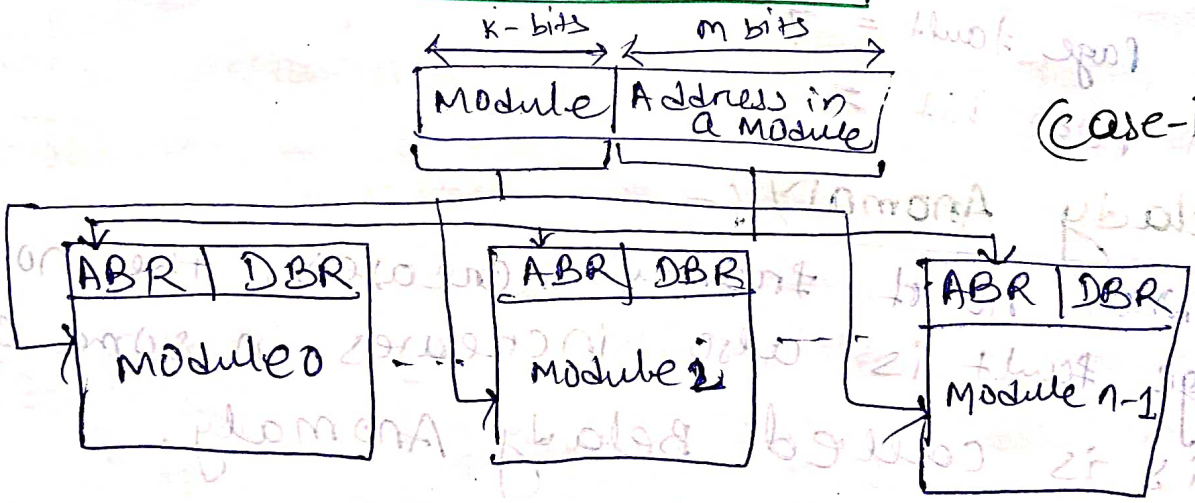
Memory interleaving :-

→ There are two methods of address layout.

(i) Consecutive words in a module.

(ii) Consecutive words in consecutive modules.

Consecutive words in a module :-



What is ABR and DBR :-

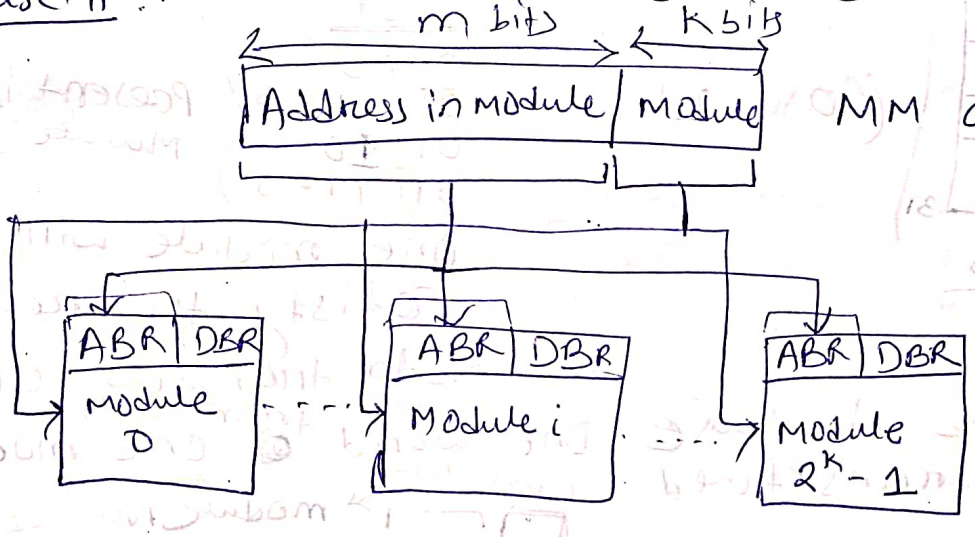
If the main memory of a computer is structured as a collection of physically separate modules, each with its own address buffer register (ABR) and data buffer register (DBR), memory access operation may proceed in more than one module at the same time.

thus, the aggregate rate of transmission of words to and from the main memory can be increased.

Case-i :- consecutive words in a module

→ The high order K bits points the n modules.
→ The low order m bits points to a particular word in that modules.
→ When consecutive locations are accessed, a block of data is transferred to a cache, only one module is involve.
→ At the same time, devices with direct memory access (DMA) ability may be accessing information in other memory address.

Case-ii :- consecutive words in consecutive module



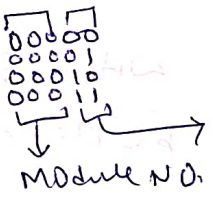
→ This way of memory addressing is called memory interleaving.
→ The low order K bits of the memory address select a module.
→ The high order m bits points to a location within that module.
→ In this way, the consecutive addresses are located in successive modules.
→ Any components of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time.

→ This results in both faster access to a block of data and higher average utilization of the memory system as a whole.

→ To implement the interleaved structure, there must be 2^k modules, otherwise there will be gaps of nonexistent location in the memory address space.

Examples of case-i and case ii

Case-1



Block size = 4 words
M.M size = 32 words
No. of module = 32/4 = 8.

(Case-1)

Suppose Request

01100 = 12
01101 = 13
01110 = 14
01111 = 15
} present in module 3 (011)

One module will satisfy the request But data transfer takes more time because one word from one module can be transferred.

particular word → module No.

00000
00001

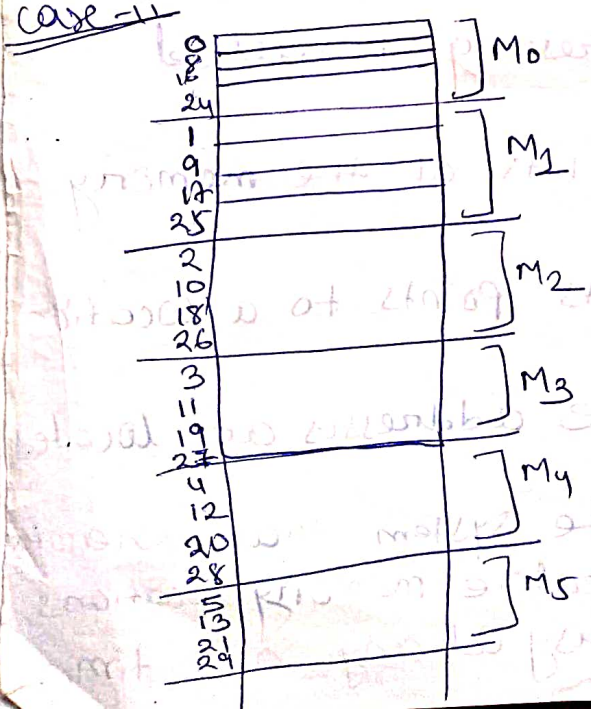
in Module 0

00 000 = 0
01 000 = 8
10 000 = 16
11 000 = 24

in Module 1

00 001 = 1
01 001 = 9
10 001 = 17
11 001 = 25

case-ii



Suppose request

12, 13, 14, 15

01100 (M4) 01101 (M5) 01110 (M6) 01111 (M7)

present in M_4, M_5, M_6, M_7. memory module

At same time these data can be transfer because present in different module.

Problem-1 :-

Let a cache with 8-word block. A read miss occurs so that entire block is to be loaded. Find the time required to access the block from memory without Interleaved and with Interleaved (4 module).

If the hardware properdies are:

(i) it takes one cycle to send address to MM.

(ii) first word of the block will be transferred by 8 clock cycles. and other words will take 4 clock cycle using fast page mode.

(iii) 1 clock cycle is needed to send one word to the cache memory.

Ans :

Without Interleaved :-

Send address to MM = 1 Clk cycle.

1st word access to DBR = 8 Clk.

Rest 7 word access to DBR and simultaneously 1st 7 words from DBR to cache = 7*4 = 28 Clock cycle

Load word after access transfer from DBR to cache = 1 Clk

38 Clk cycles

Interleaved :-

1st word address send to 4 module of MM = 1 Clk

1st word access simultaneously from 4-module to corresponding DBR = 8 Clk cycles.

2nd word access simultaneously from 4 module to corresponding DBR = 4 Clk cycles.

From 4 DBR one by one to cache memory

transfer = 4 Clk

Total = 17 Clock cycles

(reduces the block transfer time by more than a factor of 2)

The time required to load the desired block into cache is

Problems

① With interleaving:

1 clk cycle → to send address to all 4 modules in M.M

8 clk cycle → to send data from modules to DBR. 1st word all 4

→ These words are transferred to the cache one word at a time, during the next 4 clock cycles.

→ During this time, the next word in each module is accessed and store into DBR.

→ Then it takes another 4 cycles to transfer these words to the cache.

→ Therefore the total time required

1 + 8 + 4 + 4 = 17 clock cycle

→ Thus interleaving reduces the block transfer time by more than a factor of 2.

Without interleaving

1st clk: Send address to MM.

After 9th clk: 1st word data present in the DBR.

After 10th clk: Data placed into the cache.

After 13th clk: 2nd word data present in the DBR.

After 14th clk: Data placed into the cache.

After 18 → 3rd complete place in cache

22 → 4th " "

26 → 5th

30 → 6th

34 → 7th

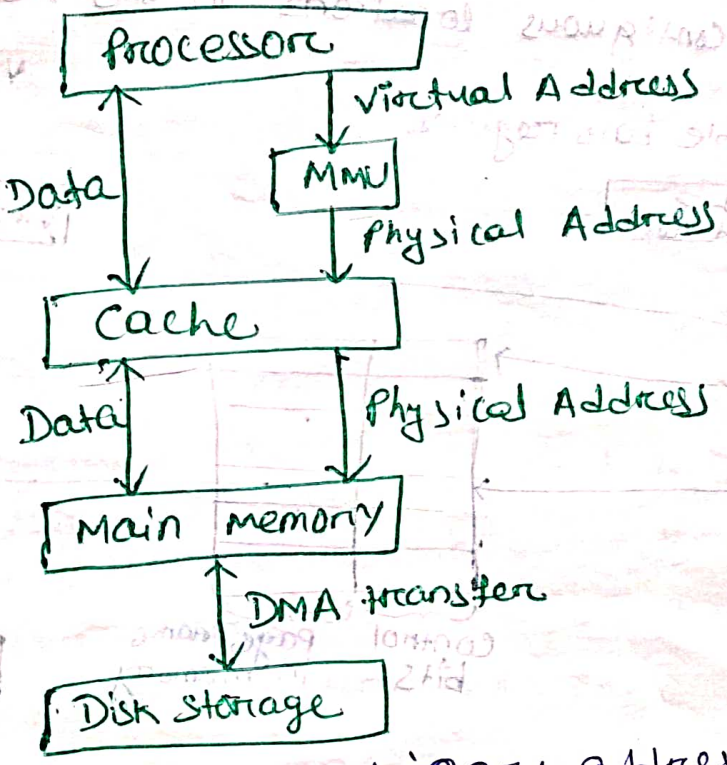
38 → 8th " " "

Virtual Memories :—

→ It is a technique that allows the execution of program that may not be completely in memory i.e program size can be greater than the physical memory size.

Advantage:

→ program size can be larger than the Physical memory.



→ The processor issues the binary address for the instruction and data is called virtual address. This address is the independent to the available physical main memory space.

→ virtual address is an imaginary address generate by the processor.

→ A memory management unit (MMU) is a special hardware unit, translates virtual address into physical address.

→ The desired data are in the main memory, these data are fetched.

→ If the data is not present in the main memory, the MMU causes the operating system to bring the data into the memory from disk.

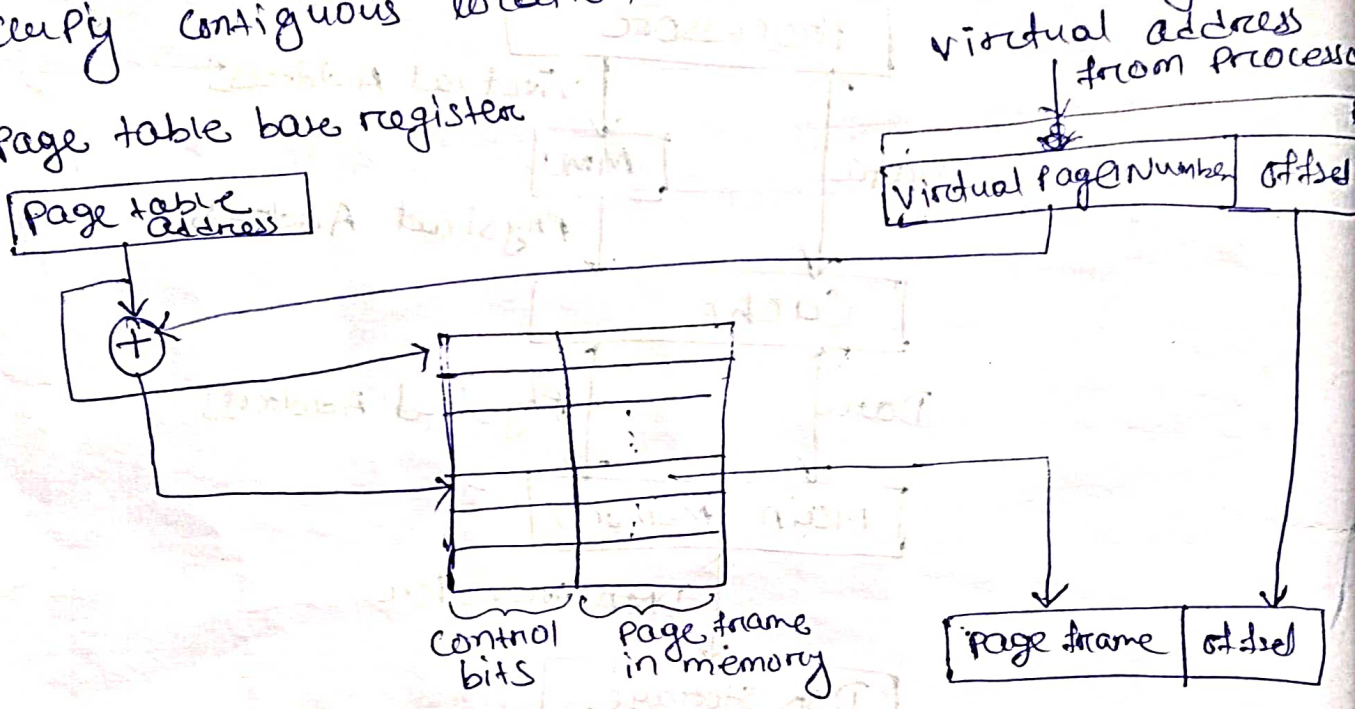
→ Transfer of data between the disk and the main memory is performed using the DMA scheme

Address translation :— (Paging)

→ translation of virtual addresses into physical address.

→ Page : All program and data are composed of fixed length units is called pages.

→ Page is consists of a block of words that occupy contiguous locations in the main memory.



→ The cache bridges the speed gap between the processor and the main memory, and is implemented in hardware

→ The virtual memory mechanism bridges the size and speed gaps between the main memory and secondary storage and is implemented in part by software techniques.

→ Processor generates virtual address, a virtual page number (high-order bits) followed by an offset (low-order bits) that specifies the location of a particular byte (or word) within a page.

→ Page table: Contain the information about the main memory locations of each page. This information includes the main memory address where the page is stored and the current status of the page.

→ Page frame: is a main memory location that can hold one page.

→ Page table base register: The starting address of the page table is kept in a page table base register (PTBR).

→ The corresponding entry in the page table is obtained by adding the virtual page number to the contents of PTBR.

→ The contents of this location gives the starting address of the page if that page currently resides in the main memory.

→ valid bit indicates the validity of the page that is whether the page is actually loaded in the main memory.

→ This bit allows the OS to invalidate the page without actually removing it.

→ Dirty bit indicates whether the page has been modified during its residency in the memory.

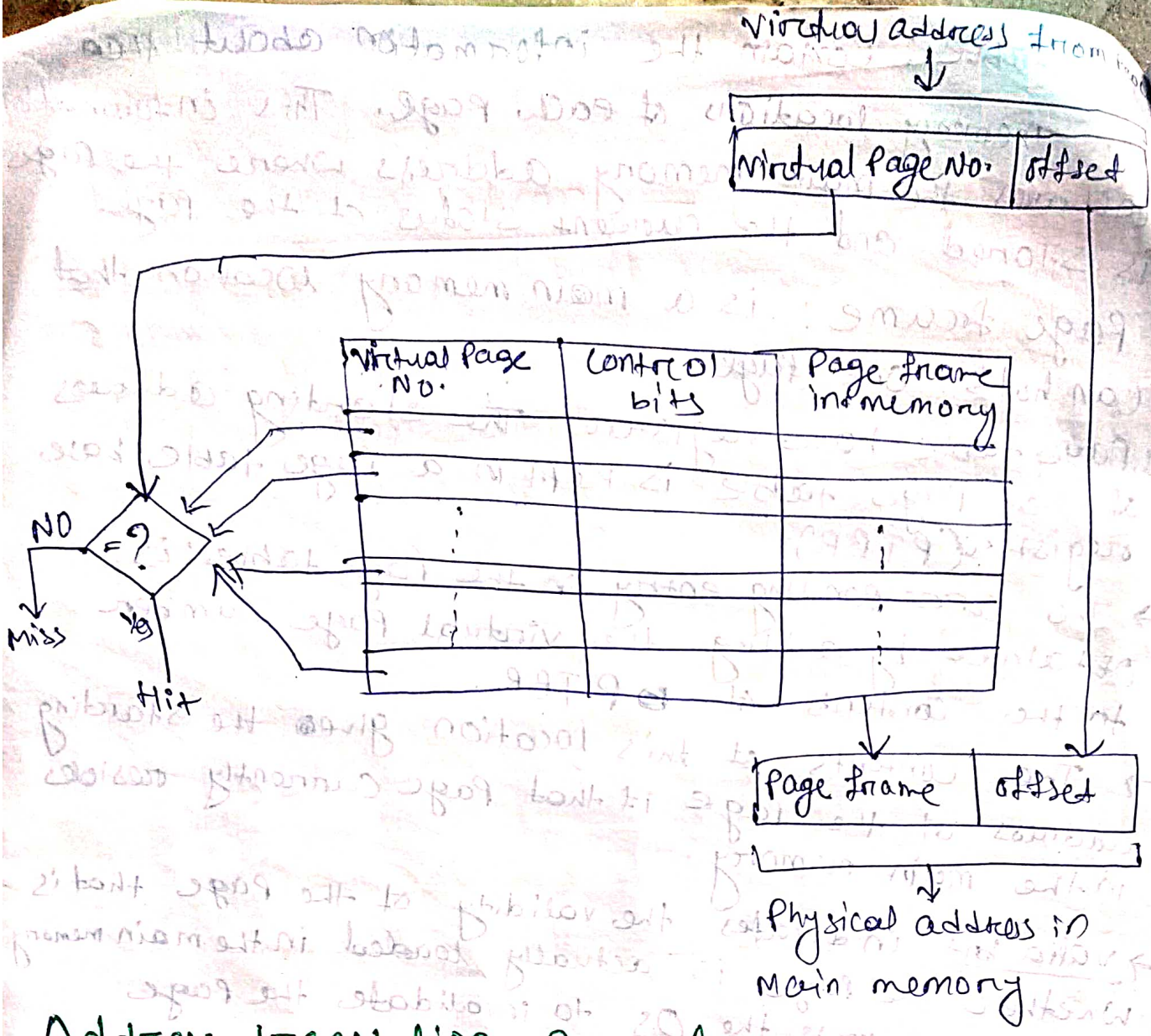
→ This information is needed to determine whether the page should be written back to the disk before it is removed from the main memory to make room for another page.

→ Page table is kept in the main memory.

→ A small cache, called the Translation look aside buffer (TLB) is incorporated into the MMU.

It contains the page table entries that correspond to the most recently accessed pages.

→ Associative mapping technique is used in TLB.



Address translation procedure :-

(i) Given a virtual address, the MMU looks in the TLB for the referenced page.

(ii) If the page table entry for this page found in the TLB, the physical address is obtained immediately. (TLB hit).

(iii) If there is Miss in TLB, then the required entry is obtained from the page table in the main memory and TLB is updated. (TLB miss then searching in Page table)

(iv) If the requested page is not in the main memory, a page fault is ~~said to have~~ occured

(v) The whole page must be brought from the disk into the memory before access can proceed

(v) When a page fault detects, the MMU asks the OS to intervene by raising an exception (interrupt).

(vi) The processing of active task is interrupted and control is transferred to the OS. The OS copies the requested page from the disk into the main memory and returns control to the interrupted task.

(vii) The execution of interrupted instruction must continue from the point of instruction or the instruction must be restarted.

Replacement Algorithm :-

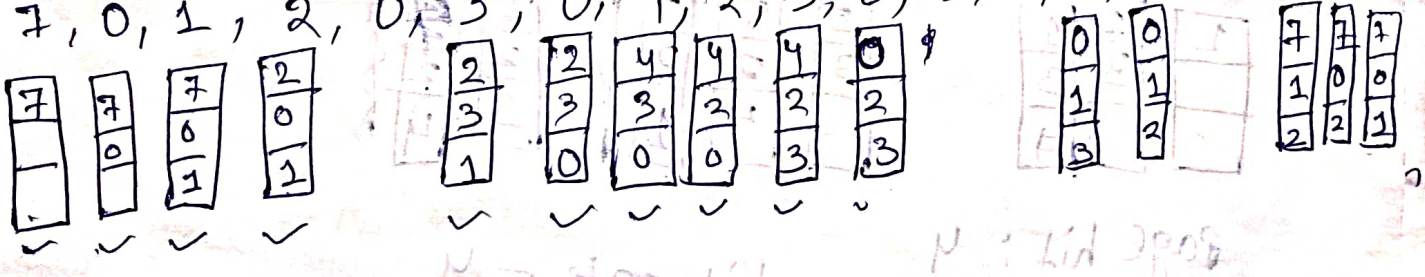
→ If a new page is brought from the disk and the main memory is full, the replacement algorithm selects a victim page in the main memory and replace the victim page with the new page from the disk.

(i) FIFO page replacement algorithm :-

→ The page that will come first will be replaced first.

Ex: Find the no. of page fault for the following reference string with frame size = 3

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



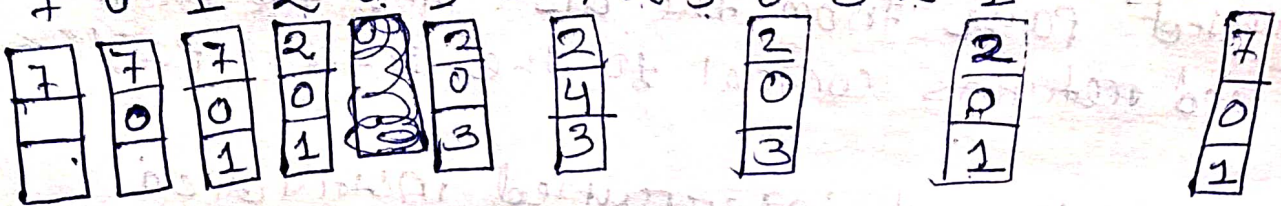
page fault = 15.

page hit = 5 hit rate = $\frac{5}{20}$

(ii) Optimal Page replacement algorithm :-

→ Replace the page that will not be used for the longest period of time. (future will check).

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



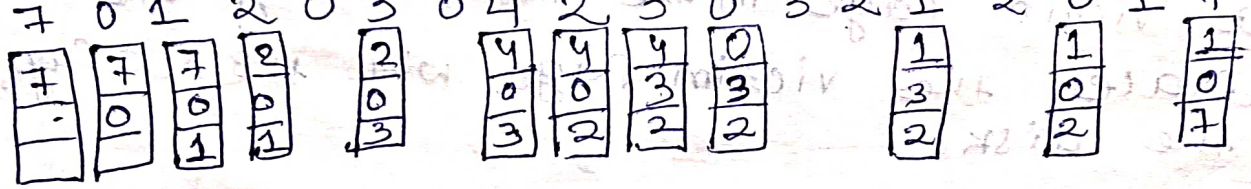
No. of Page fault = 9

∴ " Page hit = 11 hit rate = 11/20

(iii) Least Recently Used (LRU) Algorithm :-

→ Replace the page that has not been used for the longest period of time.

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



No. of Page fault = 12 hit rate = 8/20.

∴ " Page hit = 8

(iv) Most frequently used (MFU) Algorithm :-

→ The block which was used maximum no. of times that block will replace first.

Ex: 1 3 4 3 1 5 2 1 1 3 4 5

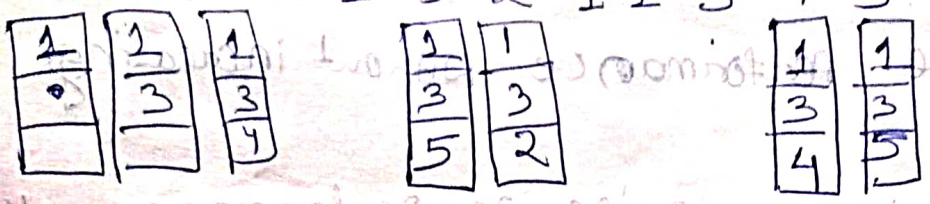
Page hit : 4 Hit rate = $\frac{4}{12}$

Page fault : 8

Least Frequently Used (LFU):-

The block which was used less no. of times will be replaced first.

Ex: 1 3 4 3 1 5 2 1 1 3 4 5



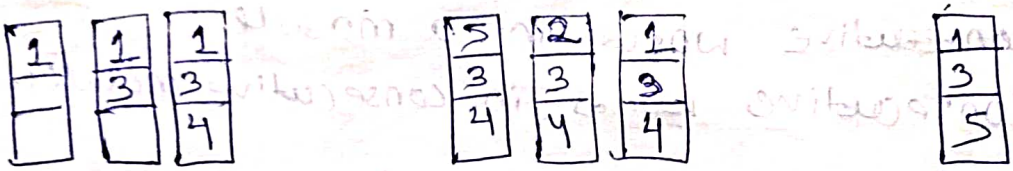
Page hit = 5

Page fault = 7

(vi) Most Recently Used (MRU):-

The block which was used in near past must be replaced first.

1 3 4 3 1 5 2 1 1 3 4 5



Page fault = 7

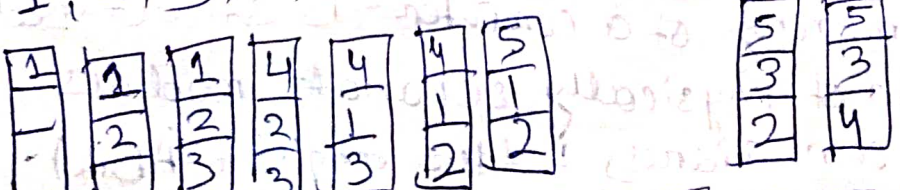
Page hit = 5

Belady Anomaly:-

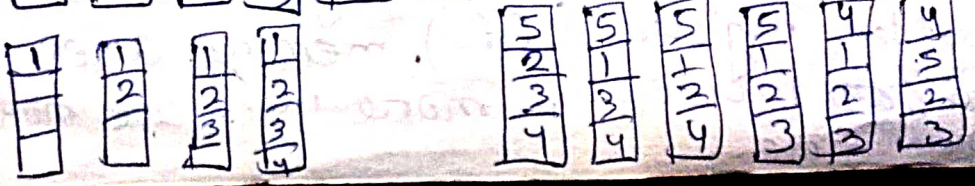
When no. of frames increases, the no. of page fault is also increases in some case. This is called Belady Anomaly.

Example: In FIFO it occurs.

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



Pf = 9



Pf = 10