PRIYANKA GOEL
ASST. PROF.
DEPTT OF CSE

UNIT-4 MEMORY

* The memory unit is an essential component in any digital computer since it is needed for storing programs and data.

* The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

e.g. a 16-bit computer that generates 16-bit addresses is capable of addressing upto $2^{16} = 64K$ memory locations.

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

Characteristics of Memory System

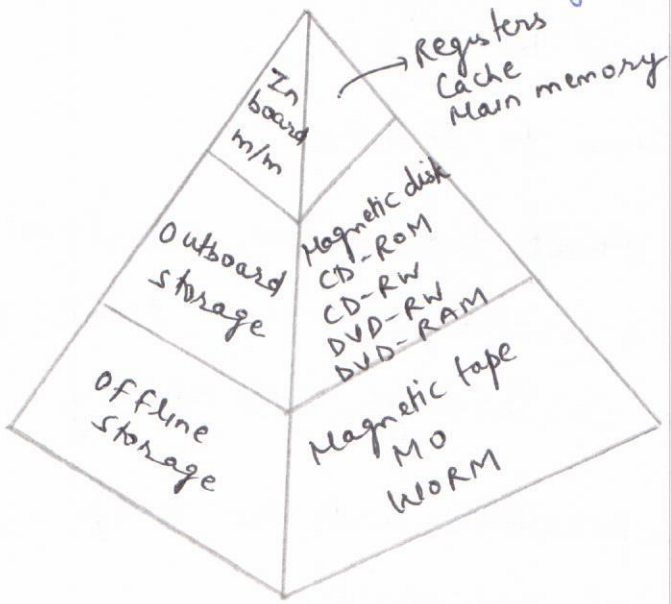
1- Location
* Processor
* Internal
* External

2- Capacity
* Word size
* No of words

3- Unit of transfer
* Word :- no. of bits used to represent a number & instruction
* Block :- In case of external memory, data is transferred in larger units than a word. These are referred as blocks.

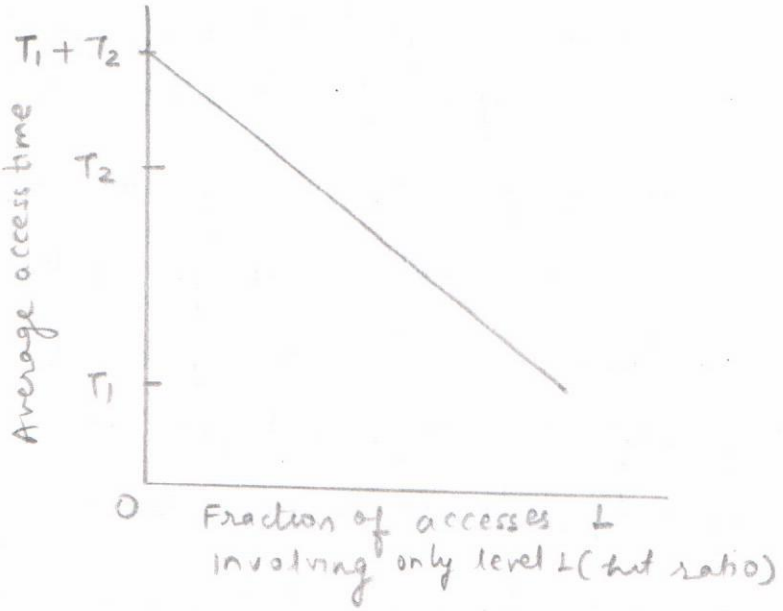
Memory Hierarchy

* There is a trade-off among the three key characteristics of memory :- cost, capacity and access time.
* Following relationship hold among various technologies that are used to implement memory systems :-
  (a) Faster access time, greater cost per bit
  (b) Greater capacity, smaller cost per bit
  (c) Greater capacity, slower access time.
* The solution to all dilemma related to meeting memory requirements is not to rely on single memory component or technology, but to employ memory hierarchy.
* A typical memory hierarchy is given as :-



* As one goes down the hierarchy, following occur :-
  a) Decreasing cost per bit
  b) Increasing capacity
  c) Increasing access time
  d) Decreasing frequency of access of the memory by processor

* Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories.



* The above figure shows the average access time to a two-level memory as a function of hit ratio H where:

  H : fraction of all memory accesses that are found in faster memory (here level 1)

  T_1 : access time to level 1

  T_2 : access time to level 2

* The graph shows that the use of two levels of memory can reduce average access time, provided that the conditions (a) through (d) apply.

* This is generally possible with the help of a principle known as locality of reference.

* This principle states that memory references tend to cluster.

* Over a long period of time, clusters in use change, but over a short period of time, processor is primarily working with fixed clusters of memory references.

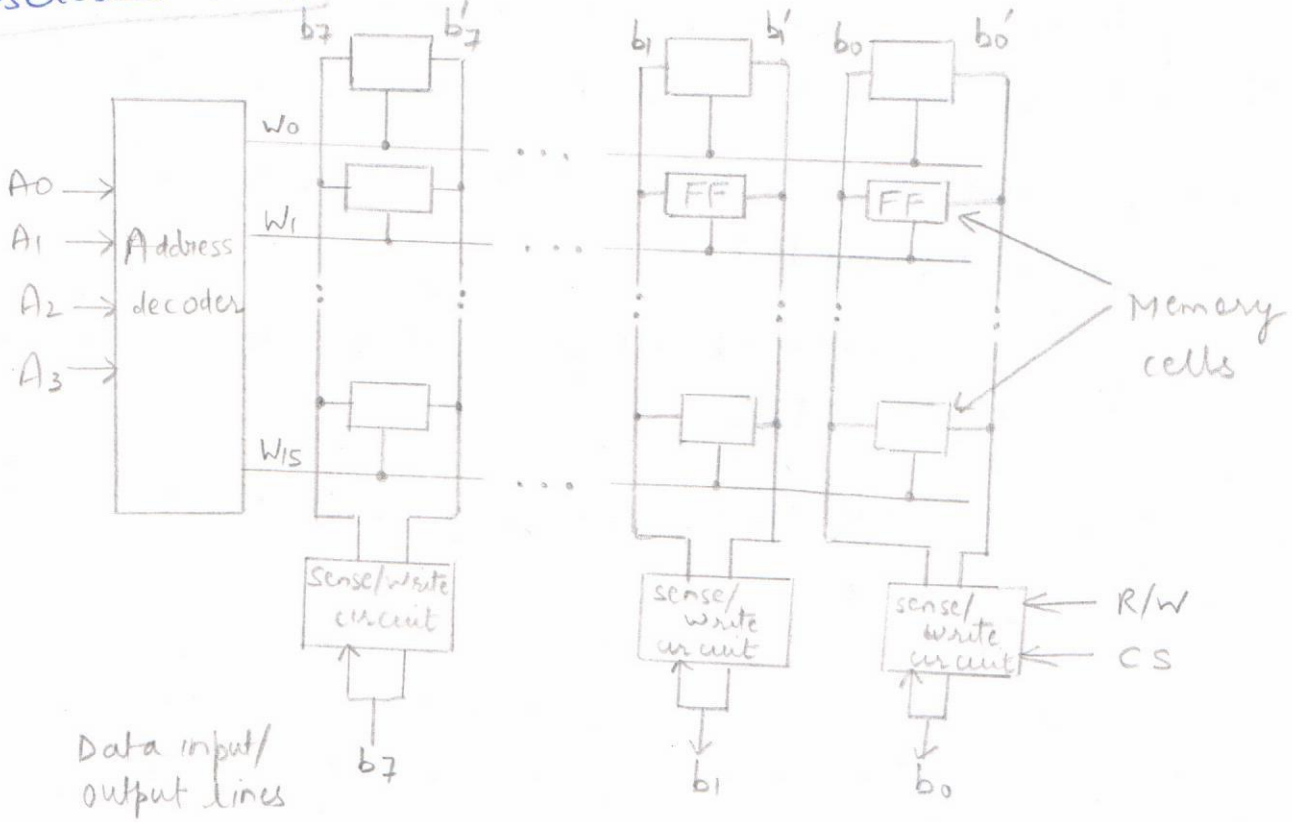
Semiconductor RAM Memories

* The cycle time of semiconductor memories range from 100ns. to less than 10 ns.
* Because of rapid advances in VLSI technology, the cost of semiconductor memories has dropped dramatically.

Internal Organization

* The basic element of semiconductor memory is the memory cell.
* Despite of various electronic technologies used, all semiconductor memory cells have following properties:-
  (a) They exhibit two stable states, which can be used to represent 1 and 0.
  (b) They are capable of being written into (atleast once) to set the state
  (c) They are capable of being read to sense the state.
* Memory cells are usually organized in the form of array in which each cell is capable of storing one bit of information.
* Each row of cells constitute a memory word and all cells of a row are connected to a common line referred to as word line, which is driven by address decoder on the chip.

* The cells in each column are connected to a sense/write circuit by two bit lines.
* 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 Write operation, the sense/write circuits receive input information and store it in the cells of the selected word.

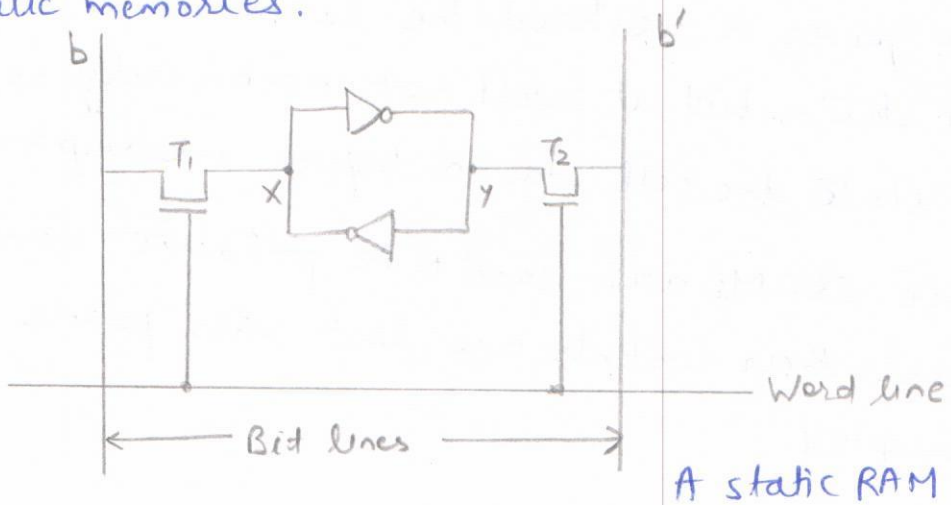


* Above figure is an example of very small memory chip consisting of 16 words of 8 bits each (referred to as 16 x 8 organization).

* The data input and data output of each sense/write circuit are connected to a single bidirectional data line that can be connected to the data bus of a computer.
* Two control lines $R/\overline{W}$ and CS are provided in addition to address and data lines.
* The $R/\overline{W}$ input specifies the required operation and the CS (Chip Select) input selects a given chip in a multichip memory system.

Static RAMs

* Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories.



A static RAM cell

* In above figure, two inverters are cross-connected to form a latch.
* The latch is connected to two bit lines by transistors T_1 and T_2.

* These transistors act as switches that can be opened or closed under control of word line.
* When the word line is at ground level, the transistors are turned off and latch retains its state
e.g. let us assume that the cell is in state 1 if the logic value at point X is 1 and at point Y is 0.
* This state is maintained as long as the signal on the word line is at ground level.

* It should be noted that continuous power is needed for the cell to retain its state.
* If power is interrupted, the cell's contents will be lost.
* When power is restored, the latch will settle into a stable state, but it will not necessarily be the same state the cell was in before interruption.
* Hence SRAMs are said to be volatile memories because their contents are lost when power is interrupted.

Advantage

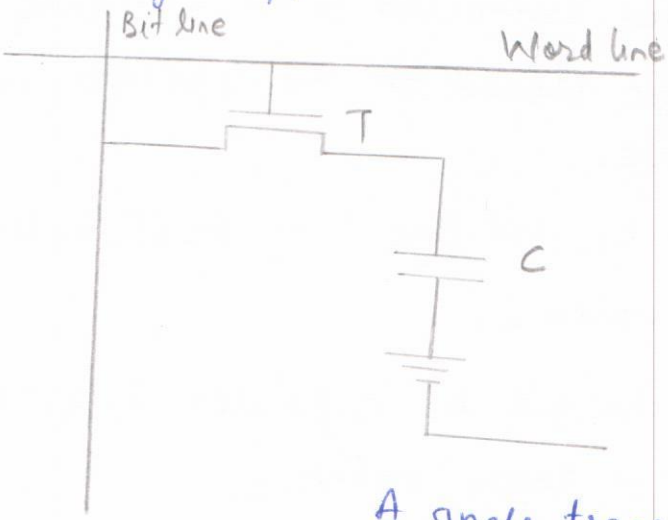
1. SRAMs consume low power because current flows in the cell only when the cell is being accessed.
2. Static RAMs can be accessed very quickly

Disadvantage

SRAMs come at a high cost because their cells require several transistors.

Dynamic RAMs

* A dynamic RAM is made with cells that store data as charge on capacitors.
* The presence or absence of charge on a capacitor is interpreted as a binary 1 or 0.
* Because capacitors have a natural tendency to discharge, dynamic RAMs require periodic charge refreshing to maintain data storage.
* The term 'dynamic' refers to the tendency of the stored charge to leak away, even when power is continuously supplied.



A single transistor dynamic memory cell

* In order to store information in the cell, transistor T is turned on and an appropriate voltage is applied to the bit line.
* This causes a known amount of charge to be stored in the capacitor.
* After the transistor is turned off, the capacitor begins to discharge.
* This is caused by capacitor's own leakage resistance &
* The transistor continues to conduct a tiny amount of current (generally in picoamperes) after it is turned off.
* Thus the information stored in the cell can be retrieved correctly only if it is read before the charge on the capacitor drops below threshold value.
* During a Read operation, the transistor in a selected cell is turned on.
* A sense amplifier connected to the bit line detects whether the charge stored on the capacitor is above the threshold value.
* If so, it drives the bit line to a full voltage that represents logic value 1.
* This voltage recharges the capacitor to the full charge that corresponds to logic value 1.
* If the sense amplifier detects that the charge on the capacitor is below the threshold value, it pulls the bit line to ground level, which ensures that the capacitor will have no charge, representing logic value 0.

* Thus reading the contents of the cell automatically refreshes its contents.
* All cells in a selected row are read at the same time which refreshes the contents of entire row.
* Applying a row address causes all cells on the corresponding row to be read and refreshed during both Read and Write operations.
* To ensure that the contents of a DRAM are maintained, each row of cells must be accessed periodically.
* Generally, a refresh circuit performs this function automatically.
* When the timing of the memory device is controlled asynchronously, such memories are called asynchronous DRAMs.
* In this case, a specialized memory controller circuit provides necessary control signals (for row & column selection) that govern the timing.
* In such cases, the processor must take into account the delay in the response of the memory.
* When the operation of DRAMs is directly synchronized with a clock signal, such memories are called synchronous DRAMs.

SRAMs Vs DRAMs

SRAMs	DRAMS
1. It is volatile ie. the power must be continously supplied to memory to preserve bit values.	It is also volatile
2. Less dense	More dense (ie. more cells per unit area)
3. Less simpler & less smaller as compared to DRAM cell.	DRAM cell is more simpler and smaller (thus more dense)
4. Does not require refreshing circuitry	requires supporting refresh circuitry.
5. faster than DRAMs	slower as compared to SRAMs
6. Due to its speed, it is generally used for cache memory (both on and off chip)	DRAM is used for main memory
7. SRAMs are digital in nature i.e. stores either 1 or 0.	DRAMs are analog in nature as capacitor can store any charge within a range; a threshold value determines whether the charge is interpreted as 1 or 0.

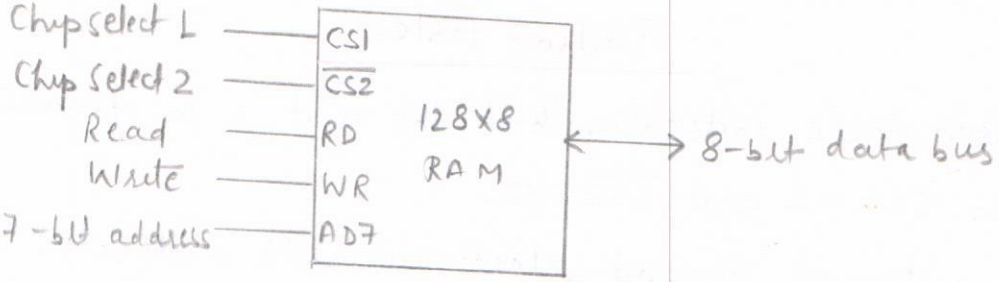
Main memory

* The main memory is the central storage unit in a computer system.
* It is a relatively large and fast memory used to store programs and data during computer operation.
* The main memory is made up of semiconductor integrated circuits.
* These integrated RAM chips are available in two possible operating modes — static and dynamic.
* The static RAM consists of internal flip flops that store the binary information.
* The stored information remains valid as long as power is applied to the unit.
* The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.
* These capacitors are provided inside the chip by MOS transistors.
* The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing dynamic memory.
* Refreshing is done by cycling through the words every few milliseconds to restore decaying charge.

* The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.
* The static RAM is easier to use and has shorter read and write cycles.
* Static RAM is used to implement cache memories while the dynamic RAMs are used to implement the main memory.
* Most of the main memory in a general purpose computer is made up of RAM integrated circuit chips but a portion of the memory may be constructed with ROM chips.
* RAM is used for storing the bulk of the programs and data that can be changed.
* ROM is used to store programs that are permanently resident in computer.
* ROM is also used to store an inital program called a bootstrap loader — whose function is to start the computer software operating when power is turned on.
* The startup of a computer consists of turning the power on and starting the execution of an inital program.
* Thus when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.
* The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system which prepares the computer for general use.

RAM and ROM chips

* If the memory needed for the computer is larger than the capacity of one chip, it is necessary to combine a number of chips to form the required memory size.



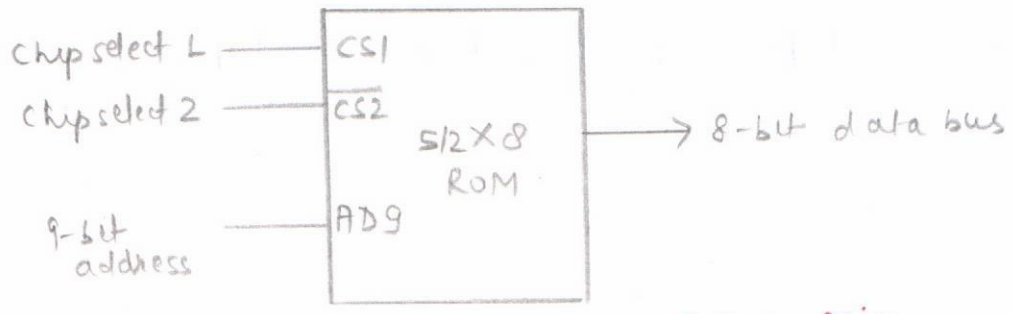
A 128x8 RAM Chip

* A bidirectional data bus is used to transfer data either from memory to CPU during read operation, or from CPU to memory during write operation.
* This bidirectional bus can be constructed with three-state buffers.
* In above example, capacity of the memory is 128 words of eight bits (1 byte) per word.
* This requires 7-bit address and an 8-bit bidirectional data bus.
* The read and write inputs specify the memory operation and the two chip select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor.

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	X	X	Inhibit	High-impedence
0	1	X	X	Inhibit	High-impedence
1	0	0	0	Inhibit	High-Impedence
1	0	0	1	Write	Input data to RAM
1	0	1	X	Read	Output data from RAM
1	1	X	X	Inhibit	High Impedence

Function table

* The function table indicates that the unit is in operation only when CS1 = 1 and $\overline{CS2} = 0$.
* The bar on top of second select variable indicates that this input is enabled when it is equal to 0.
* If the chip select inputs are not enabled, or if they are enabled but the read or write inputs are not enabled, the memory is inhibited and its data bus is in high impedence state.
* When CS1 = 1 and $\overline{CS2} = 0$, the memory can be placed in a write or read mode.
* When WR input is enabled, memory stores a byte from data bus into a location specified by the address input lines.
* When RD input is enabled, the contents of selected byte is placed into data bus.



A 512 X 8 ROM Chip

* Just like a RAM chip, ROM chip can also be organized in similar way.
* However, since ROM can only read, data bus can only be in an output mode.
* As shown in previous figure, for the same-size chip, it is possible to have more bits of ROM than of RAM because the internal binary cells in ROM occupy less space than in RAM.
* There is no need for a read or write control because the unit can only read.

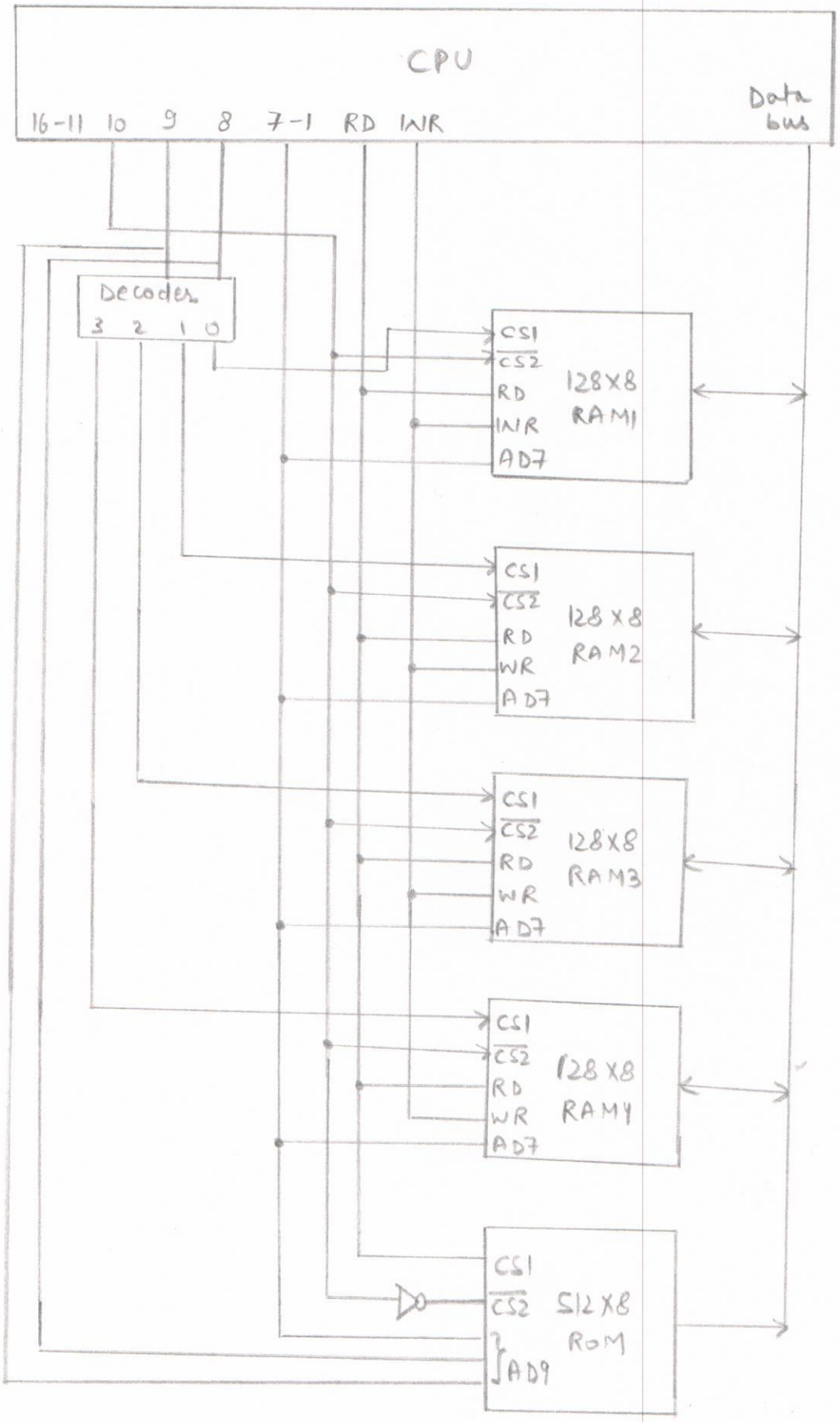
Memory Address Map

* Assuming that a computer system needs 512 bytes of RAM and 512 bytes of ROM, we need 4 128X8 RAM chips and 1 512X8 ROM chip.
* The addressing of memory can be established by means of a table that specifies memory address assigned to each chip.
* This table (known as memory address map) provides a pictorial representation of assigned address space for each chip in the system.

Component	Hexadecimal address	Address bus										Decimal addr
		10	9	8	7	6	5	4	3	2	1	
RAM 1	0000-007F	0	0	0	x	x	x	x	x	x	x	000-127
RAM 2	0080-00FF	0	0	1	x	x	x	x	x	x	x	128-255
RAM 3	0100-017F	0	1	0	x	x	x	x	x	x	x	256-383
RAM 4	0180-01FF	0	1	1	x	x	x	x	x	x	x	384-511
ROM	0200-03FF	1	x	x	x	x	x	x	x	x	x	512-1023

* In the table, the component column specifies whether a RAM or ROM chip is used.
* Although there are 16 address lines but the table shows only 10 lines because the other 6 are not used and are assumed to be zero.
* The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip
* The RAM chips have 128 bytes and so need 7 address lines.
* The ROM chip has 512 bytes and need 9 address lines.
* The x's are assigned to lower bus lines— lines 1 to 7 for the RAM and lines 1 through 9 for ROM.
* Now to distinguish between four RAM chips, we chose bus lines 8 and 9 to represent four distinct binary combinations.
* The distinction between RAM and ROM chip is done using bus line 10.

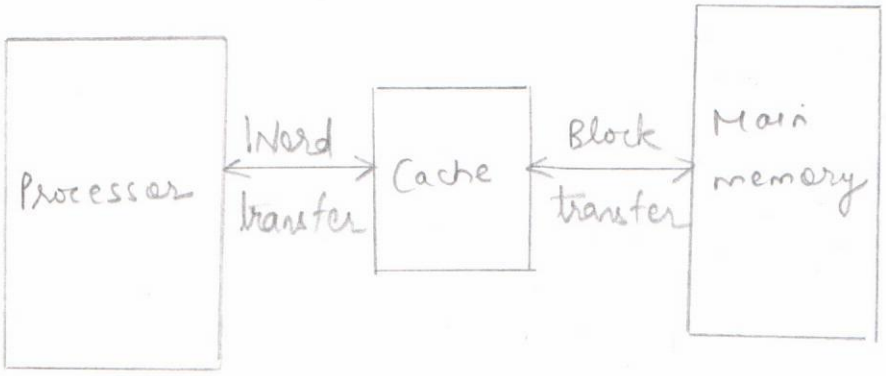
Memory Connection to CPU



Cache Memory

* The speed of main memory is very low in comparison with the speed of modern processors.
* For good performance, the processor cannot spend much of its time waiting to access instructions and data in main memory.
* Hence it is important to devise a scheme that reduces the time needed to access the necessary information.
* An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster than it really is.
* The effectiveness of the cache mechanism is based on the principle of locality of reference.
* This principle can be manifested in two ways :- temporal and spatial
* The temporal locality of reference means that a recently executed instruction is likely to be executed again very soon.
* The spatial locality of reference means that instructions in close proximity to a recently executed instruction (with respect to the instruction's addresses) are also likely to be executed soon.
* The memory control circuitry is designed to take advantage of the property of locality of reference.

* The temporal aspect of the locality of reference suggests that whenever an information element (instruction or data) is first needed, this element should be brought into cache where it will hopefully remain until it is needed again.
* The spatial aspect suggests that instead of fetching just one element from the main memory to the cache, it is useful to fetch several elements that reside at adjacent addresses as well.
* The term 'block' is used to refer to a set of contiguous address locations of some size.
* A cache block is also called cache line.



* When a Read request is received from the processor, the contents of a block of memory words containing the location specified are transferred into cache one word at a time.
* Thus, references to any of the locations in this block can be met by directly reading from cache.
* Generally, cache can store few number of blocks as compared to main memory.
* The correspondence between main memory blocks and those in the cache are is specified by mapping function.

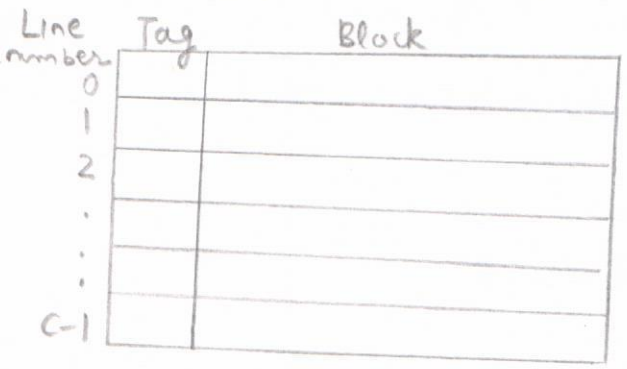
* When the cache is full and a memory word (instruction or data) that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contains the referenced word.
* The collection of rules for making this decision constitutes the replacement algorithm.

Cache hit :- If the required element is present in the cache, it is called cache hit.

Hit Latency :- The time taken to find data in cache whether required element is present in cache or not.

Cache Miss :- If the required element is not present in cache, it is called cache miss.

Miss latency :- The time taken to get required element (data or instruction) from main memory and place it on to cache is called miss latency.



(a) Cache

(b) Main memory

* This figure shows the cache/main memory structure.
* Main memory consists of upto 2^n addressable words, with each word having a unique n-bit address.
* This memory consists of a number of fixed length blocks of K words each.

Assuming 1 word = 1 byte

ie. Total number of words in main memory $= 2^n$

No. of bits required to represent each word = n

No of words per block in memory = K

No of blocks in main memory $= \frac{2^n}{K}$ blocks

Blocks in cache are called lines.

and, Block size of main memory = Line size of cache memory

No. of ~~lines~~ words in cache memory = $W = 2^m$

No. of lines in cache memory = $\frac{W = 2^m}{K}$

No. of bits required to represent each word in cache = m

It should be noted that $C << M$

* If a word in a block of memory is read, that block is transferred to one of the lines of the cache.

* Because there are more blocks than lines, an individual line cannot be uniquely and permanently dedicated to a particular block.

* Thus, each line includes a tag that identifies which particular block is currently being stored.

* This tag is usually a portion of main memory.

e.g. if main memory has 64 words, cache has 16 words and block size = 4 words

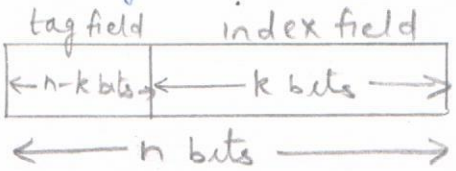
then, No. of blocks in main memory = $\frac{64}{4}$ = 16 blocks

No of blocks / lines in cache = $\frac{16}{4}$ = 4 lines.

Direct mapping

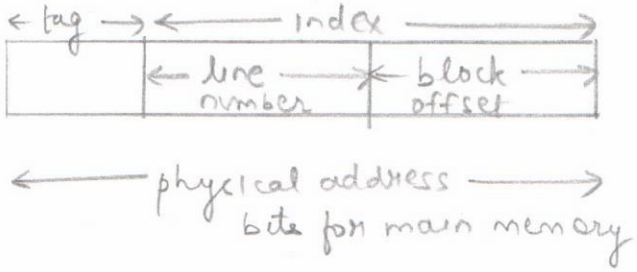
* There are 2^k words in cache memory and 2^n words in main memory.

* The n-bit memory address is divided into two fields :-



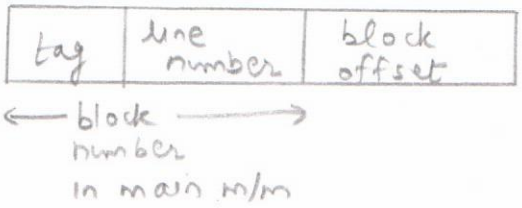
* The direct mapping cache organization uses n-bit address to access main memory and k-bit index to access cache.

* The ~~tag~~ index field can further be divided into :-



* The block offset field tells which block to be accessed
* The line number field tells which line in cache contains the block
* The tag field is used to differentiate between blocks present in one cache line.

Also,

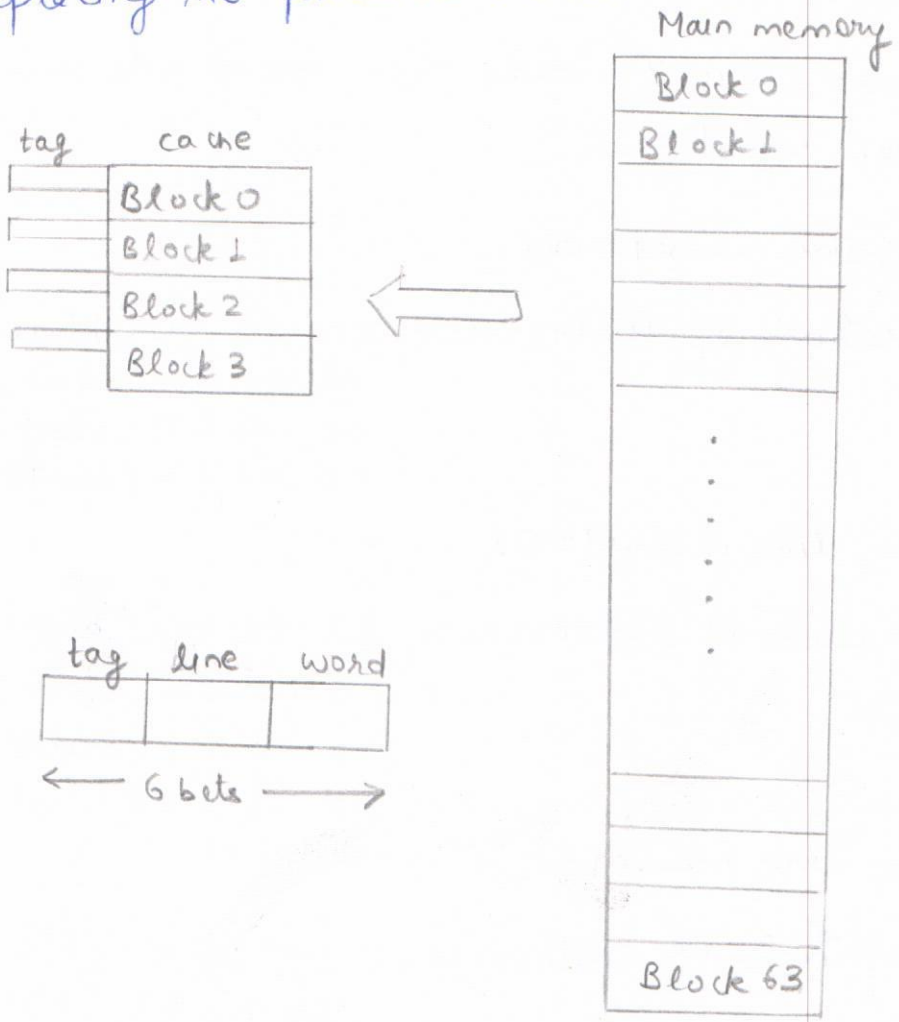


* Each word in cache consists of the data word and its associated tag.
* When a new word is first brought into the cache, the tag bits are stored alongside the data bits.

Working

* When the CPU generates a memory request, the index field is used for the address to access the cache.
* The tag field of the CPU address is compared with the tag in the word read from cache.
* If the two tags match, there is a hit and desired data word is in cache.

* If there is no match, there is a miss and the required word is read from main memory.
* It is then stored in the cache together with the new tag, replacing the previous value.



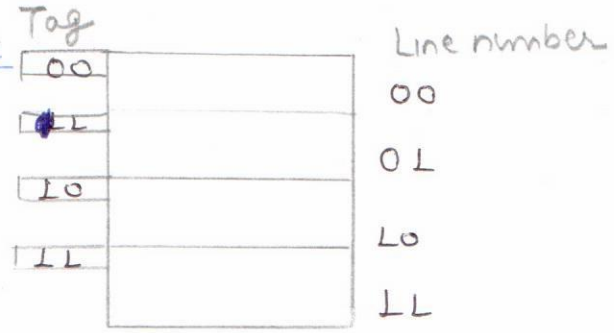
Here Main memory size = 64 words
Cache size = 16 words
Block size = 4 words

The blocks of main memory will be assigned in round robin way as :-

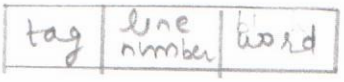
0	0, 4, 8, 12
1	1, 5, 9, 13
2	2, 6, 10, 14
3	3, 7, 11, 15

Reason is :- e.g

Block No.



If each line can store 4 words then, no. of bits used to represent each word = 2



When tag = 00, line number = 00

Possible combinations of Addresses :- 00 00 00 (block 0)
00 00 01 (block 1)
00 00 10 (block 2)
00 00 11 (block 3)

When tag = 11, line number = 01

Possible combinations of addresses :- 11 01 00 (block 52)
11 01 01 (block 53)
11 01 10 (block 54)
11 01 11 (block 55)

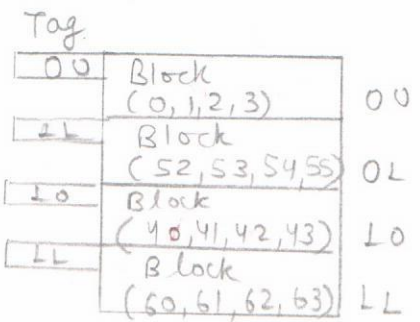
When tag = 10, line number = 10

Possible combinations of addresses :- 10 10 00 (Block 40)
10 10 01 (Block 41)
10 10 10 (Block 42)
10 10 11 (Block 43)

When tag = 11, line number = 11

Possible combinations of addresses :- 11 11 00 (Block 60)
11 11 01 (Block 61)
11 11 10 (Block 62)
11 11 11 (Block 63)

Thus,



Few examples on Direct mapping

e.g.1 :- Main memory size = 32 GB

$= 2^5 \times 2^{30}$ GB

$= 2^{35}$ B

↳ 35 bits

Find tag size & tag directory size.

Cache size = 32 KB

$= 2^5 \times 2^{10}$ B

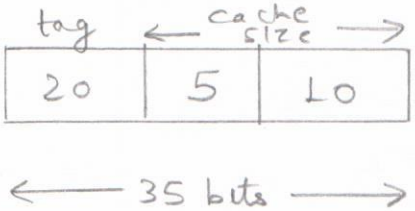
$= 2^{15}$ B

↳ 15 bits

Block size = 1 KB

$= 2^{10}$ B

↳ 10 bits



Thus, No of bits used for tag field = 20

Tag directory size = No of bits for tag field * No of lines in cache

$= 20 \times 2^5$

(Thus, it indicates that 2^{20} blocks of main memory can be mapped into one line of cache)

No of lines in cache = $\frac{\text{Cache size}}{\text{block size}} = \frac{2^{15}}{2^{10}}$

$= 2^5$

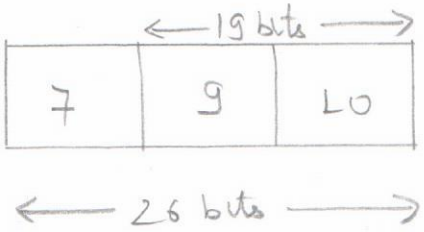
↳ 5 bits

e.g.2 :- Cache size = 512 KB

Block size = 1 KB

Tag bits in physical address = 7

Find Main memory size and tag directory size?



Cache size = 512 KB
$= 2^9 \times 2^{10}$ B
$= 2^{19}$ B
↳ 19 bits

Block size = 1 KB
$= 2^{10}$ B
↳ 10 bits

So, Main memory size
$= 2^{26}$ B
$= 2^6 \times 2^{20}$ B
= 64 MB

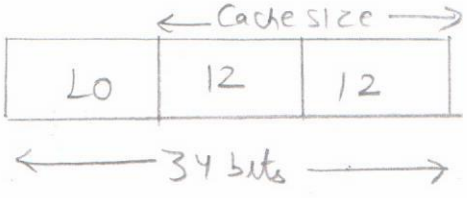
Tag directory size = 7×2^9 bits

eg.3 :- Main memory size = 16 GB
$= 2^4 \times 2^{30}$ B
$= 2^{34}$ B
↳ 34 bits

Block size = 4 KB
$= 2^2 \times 2^{10}$
$= 2^{12}$ B
↳ 12 bits

No. of tag bits = 10

Find cache size & tag directory size.



No. of bits for a line in cache = 12

So, Cache size = $2^{12} \times 2^{12}$
= 16 MB

Tag directory size = 10×2^{12} bits

e.g.4 :- Main memory size = 64 MB

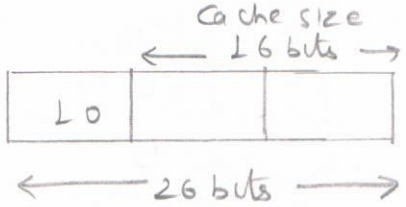
$= 2^6 \times 2^{20}$ B

$= 2^{26}$ B

↳ 26 bits

Tag bits in Physical address = LO

Find cache size, block size, tag directory size?



Cache size = 2^{16} bytes

But block size cant be guessed and similarly tag directory size cant be accessed.

e.g.5 :- Cache size = 512 KB

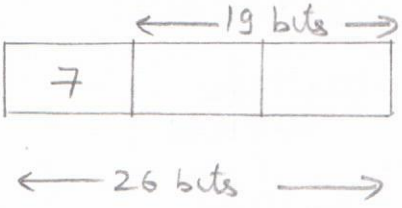
$= 2^9 \times 2^{10}$ B

$= 2^{19}$ B

↳ 19 bytes

Tag bits in Physical address = 7

Find Main memory size, tag directory size?



Main memory size = 2^{26} bytes

↓

26 bits

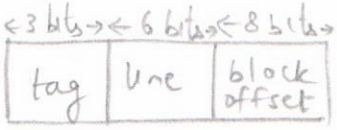
again block size cant be guessed & similarly tag directory size cant be guessed

e.g.1 :- Main memory size = 128 KB
$= 2^7 \times 2^{10}$
$= 2^{17}$ B
↳ 17 bits

Cache size = 16 KB
$= 2^4 \times 2^{10}$
$= 2^{14}$ B
↳ 14 bits

Block size = 256 B
$= 2^8$ B
↳ 8 bits

No of lines in cache
$= \frac{2^{14} B}{2^8 B}$
$= 2^6$ B
↳ 6 bits



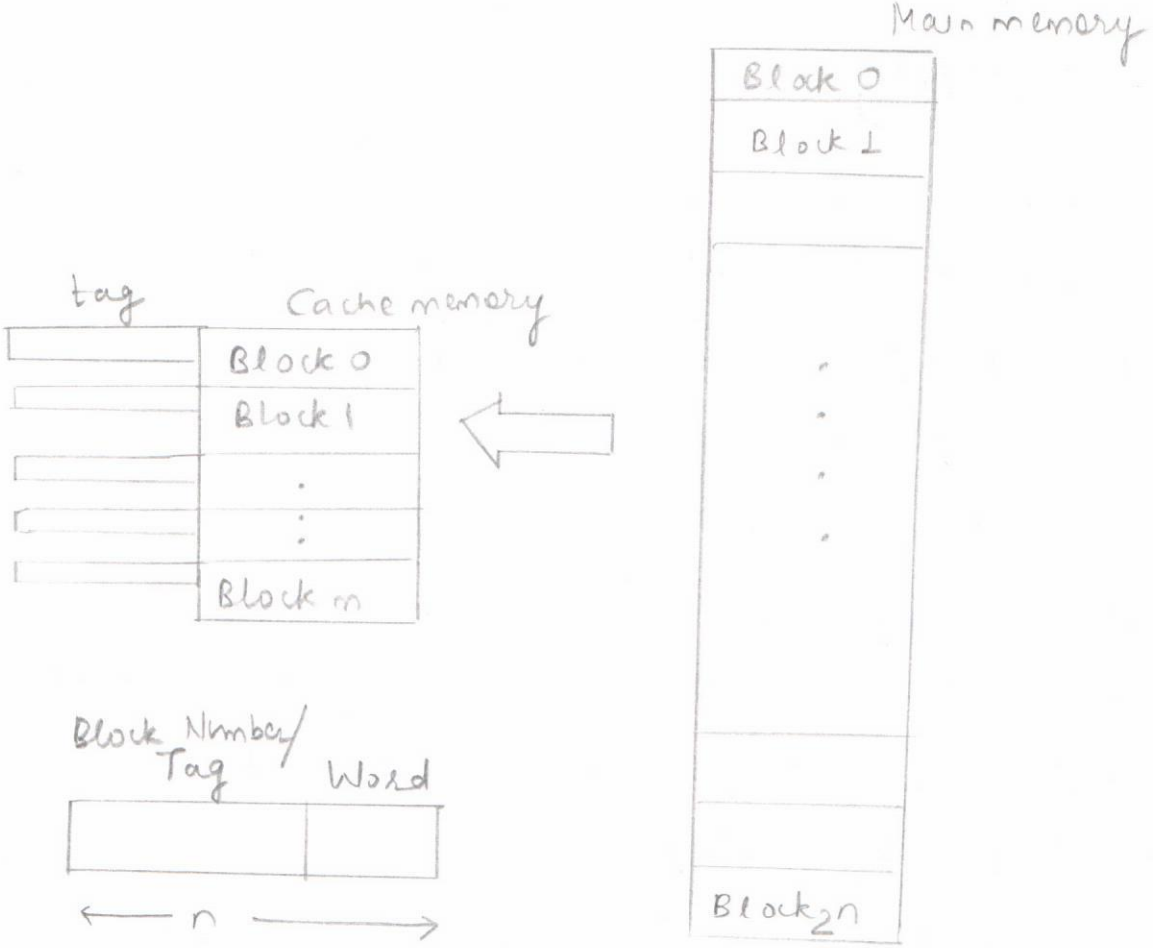
So, No of tag bits = 3 bits

Tag directory size = 3×2^6 bits

Main memory size	Cache size	Block size	Tag bits	Tag directory size
32 GB	32 KB	1 KB	20 bits	20×2^5
~~512~~ 64 MB	512 KB	1 KB	7 bits	7×2^9 bits
16 GB	16 MB	4 KB	10 bits	10×2^{12} bits
64 MB	64 KB	can't be guessed	10 bits	can't be guessed
64 MB	512 KB	can't be guessed	7 bits	can't be guessed
128 KB	16 KB	256 B	3 bits	3×2^6 bits

Associative mapping

* It is more flexible method in which a main memory block can be placed into any cache block position.
* The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present.
* This technique gives complete freedom in choosing the cache location in which to place the memory block.
* Thus the space in the cache can be used more efficiently. ⇒ Advantage
* A new block that has to be brought into the cache has to replace (eject) an existing block only if the cache is full.
* In this case, we need block replacement algorithm to select the block to be replaced, such as FIFO, LRU, MFU etc (similar to page replacement algorithm).
* The cost of an associative cache is higher than the cost of a direct-mapped cache because of the need to search all tag patterns to determine whether a given block is in cache.
* Such searching is called associative search and hence the name.



Disadvantage

* Tag size increases. Moreover if any word is present in cache, its exact position cant be find out easily.

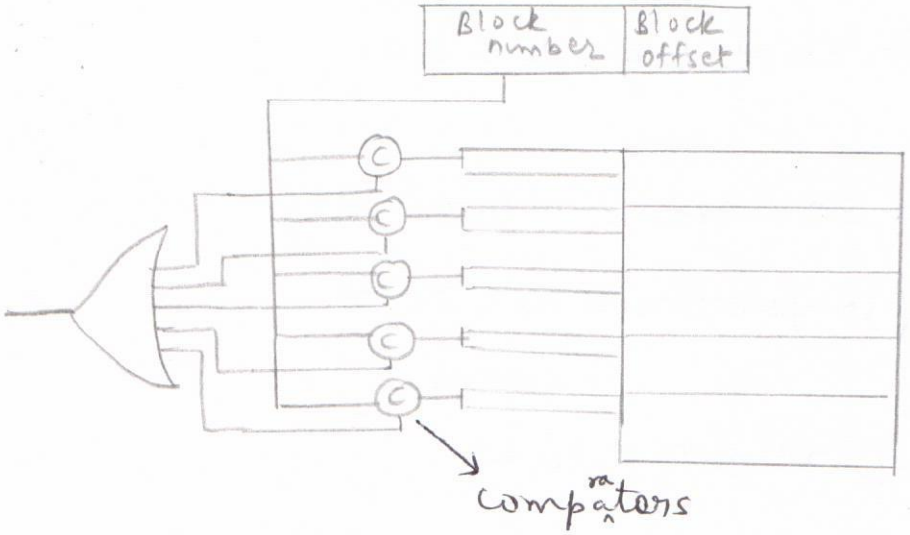
In direct mapping, only one tag is chosen using one comparator.

In associative mapping, all tag fields need to be compared with that of the word to be searched.

Thus leading to more access time.

↓ its solution is:

use all comparators parallely as shown in figure below-



If any of the tag field matches with the block number, it means hit in cache.
Or, the OR gate will produce 1 if any of tag field matches with block number.

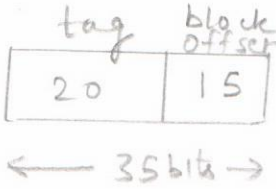
* In this case,
Hit latency = Propagation delay of comparators + Propagation delay of OR gate

The disadvantage of above scheme is that for n cache lines, we need n comparators ⇒ thus increase in hardware cost.

eg if main memory size = 32 GB
$= 2^5 \times 2^{30}$
$= 2^{35}$ B
↳ 35 bits

block size = 32 KB
$= 2^5 \times 2^{10}$
$= 2^{15}$ B
↳ 15 bits

Find tag bits and hit latency if propagation delay of 1 comparator is 10 ns & propagation delay of OR gate is 10 ns.



So, no. of tag bits = 20

No. of cache lines = 20 (equal to tag bits)

Propagation delay of comparators = 10 × 20 ns
= 200 ns

Propagation delay of OR gate = 10 ns

∴ Hit latency = 200 + 10 ns
= 210 ns

* It should be noted that in direct mapping, replacement of a word can only occur in a particular line. ~~Generally~~
* But in case of associative mapping, replacement algorithm is used.

e.g.:-

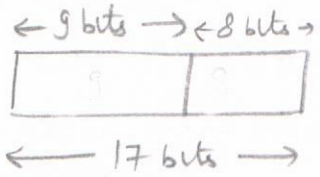
No. of comparators	Main memory size	cache size	block size	tag bits	tag directory size
$2^6 = 64$	128 KB	16 KB	256 B	9 bits	9×2^6 bits
$2^5 = 32$	32 GB	32 KB	1 KB	25 bits	25×2^5 bits
practically infeasible		512 KB	1 KB	7 bits	–
2^9	128 MB	512 KB	1 KB	17 bits	17×2^9 bits
can't be guessed	16 GB	can't be guessed	4 KB	10 bits ↳ not feasible	can't be guessed
can't be guessed	16 GB	can't be guessed	4 KB	22 bits	can't be guessed
can't be guessed	64 MB	can't be guessed	64 KB	10 bits	can't be guessed
can't be guessed		512 KB	can't be guessed	7 bits	can't be guessed

e.g.1 :- Main memory size = 128 KB
$= 2^7 \times 2^{10}$ B
$= 2^{17}$ B
↳ 17 bits

Cache size = 16 KB
$= 2^4 \times 2^{10}$ B
$= 2^{14}$ B
↳ 14 bits

Block size = 256 B
$= 2^8$
↳ 8 bits

No of lines in cache = $\frac{\text{Cache size}}{\text{block size}}$
$= \frac{2^{14}}{2^8}$
$= 2^6$



So, tag bits = 9
tag directory size = 9×2^6

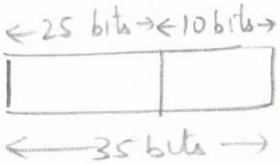
No of comparators = 2^6

e.g.2 :- Main memory size = 32 GB
$= 2^5 \times 2^{30}$ B
$= 2^{35}$ B
↳ 35 bits

Cache size = 32 KB
$= 2^5 \times 2^{10}$ B
$= 2^{15}$ B
↳ 15 bits

No of lines
$= \frac{2^{15}}{2^{10}}$
$= 2^5$

Block size = 1 KB
$= 2^{10}$ B
$= 10$ bits

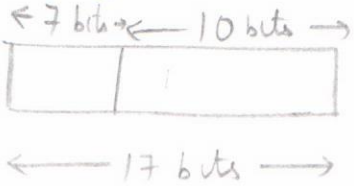


So, tag bits = 25
Tag directory size = 25×2^5 bits
No of comparators = 2^5

Q93 :- Cache size = 512 KB
$= 2^9 \times 2^{10}$ B
$= 2^{19}$ B
↓
19 bits

Block size = 1 KB
$= 2^{10}$ B
↳ 10 bits

Tag bits = 7 bits

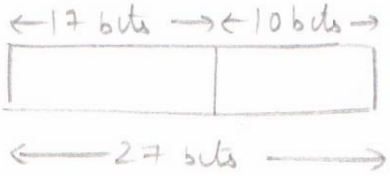


Thus, main memory size
$= 2^{17}$ B
= 128 KB
↳ which is practically impossible since Main m/m size > Cache size

Q94 :- Cache size = 512 KB
$= 2^9 \times 2^{10}$ B
$= 2^{19}$ B
↳ 19 bits

Block size = 1 KB
$= 2^{10}$ B
↓
10 bits

Tag bits = 17 bits



Thus, main memory size
$= 2^{27}$ B
= 128 MB

Tag directory size = 17×2^9 bits

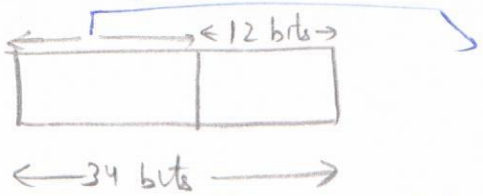
No of comparators = 2^9

No of lines
$= \frac{2^{19}}{2^{10}}$
$= 2^9$

eg5 :- Main memory size = 16 GB
$= 2^4 \times 2^{30}$ B
$= 2^{34}$ B
↳ 34 bits

Block size = 4 kB
$= 2^2 \times 2^{10}$ B
$= 2^{12}$ B
↳ 12 bits

Tag bits = 10 bits

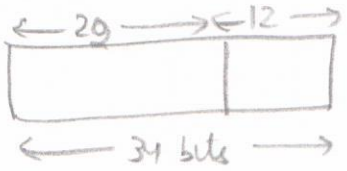


this must be 20 but in question, it is given 10 bits.

Also, tag bits do not contain any information about cache size so, we can't find cache size as well as tag directory size.

eg.6 :- Main memory size = 16 GB
$= 2^{34}$ B
↳ 34 bits

Block size = 4 kB
$= 2^{12}$ B
↳ 12 bits



Cache can be of any size so its size can't be guessed and Tag directory size can't be guessed too.

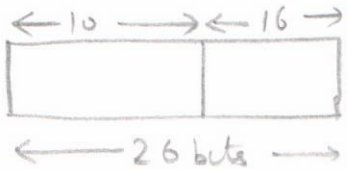
e.g.7 :- Main memory size = 64 MB

$= 2^6 \times 2^{20}$ B

$= 2^{26}$ B

↳ 26 bits

No. of tag bits = 10



So, block size = 2^{16} B

= 64 kB

Cache size & Tag directory size can't be guessed

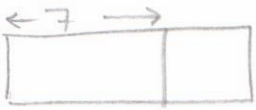
e.g 8 :- Cache size = 512 KB

$= 2^9 \times 2^{10}$ B

$= 2^{19}$ B

↳ 19 bits

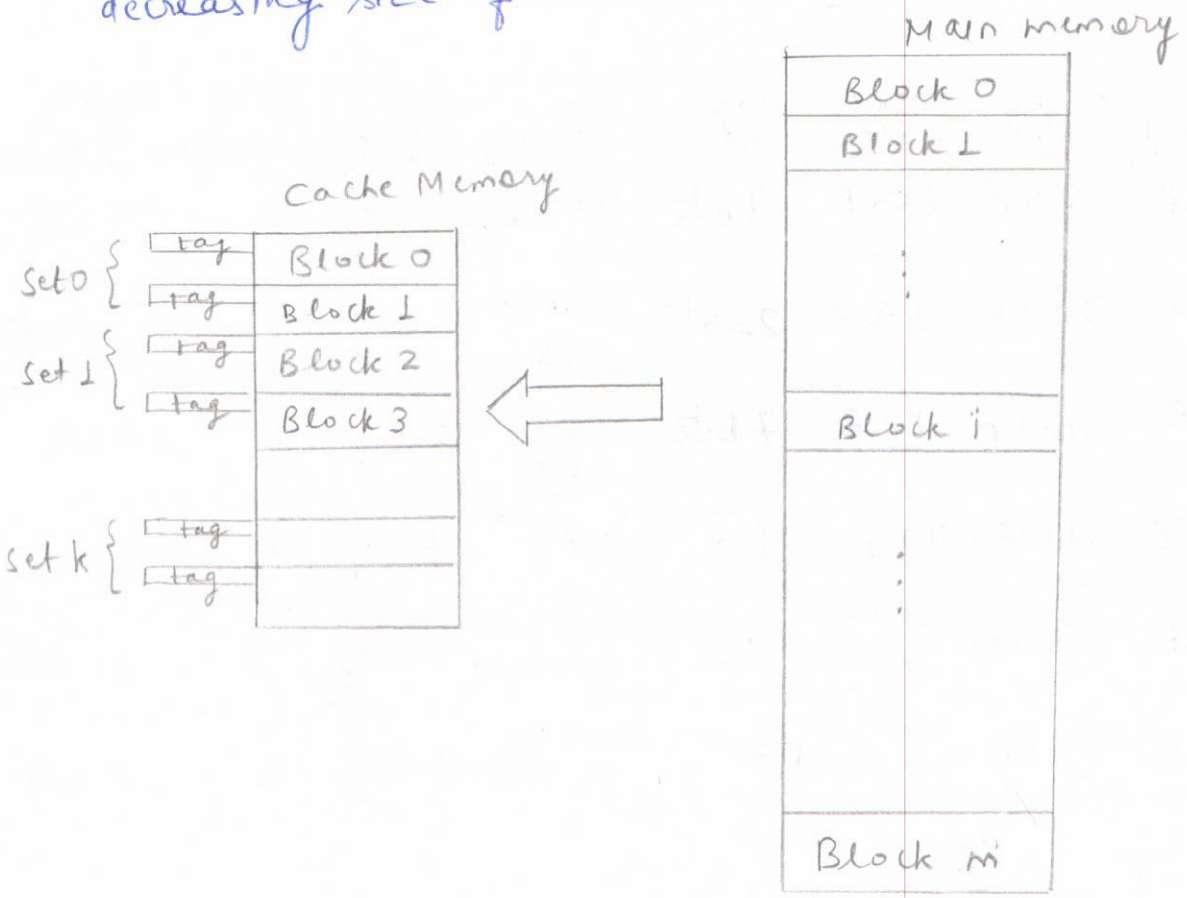
No. of tag bits = 7 bits



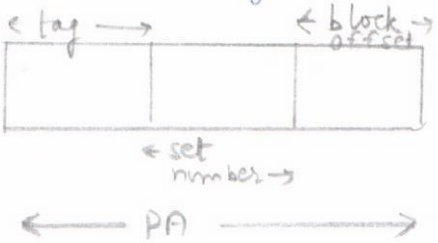
To find memory size, block size & tag directory size with given information is not possible.

Set Associative Mapping

* It is a combination of direct and associative mapping techniques.
* Blocks of the cache ~~ca~~ are grouped into sets and the mapping allows a block of the main memory to reside in any block of a specific set.
* Thus, the contention problem of direct mapping is eased by having few choices for block placement.
* At the same time, the hardware cost is reduced by decreasing size of associative search.



* The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer.
* A cache that has k blocks per set is referred to as a k-way set-associative cache.
* The physical address of a memory word in set-associative cache is given as —



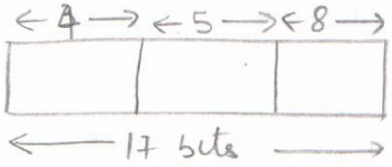
e.g.

Main memory size	Cache size	Block size	Tag bits	Tag directory size	Set associativity	No. of comparators	Size of each comparator
128 KB	16 KB	256 B	4 bits	$4*2^6$ bits	2-way	2	4 bits
32 GB	32 KB	1 KB	22 bits	$22*2^5$ bits	4-way	4	22 bits
8 MB	512 KB	1 KB	7 bits	$7*2^9$ bits	8-way	8	7 bits
16 GB	64 MB	4 KB	10	$10*2^{14}$	4-way	4	10 bits
64 MB	256 KB	can't be guessed	10	can't be guessed	4-way	4	10 bits
8 MB	512 KB	can't be guessed	7	can't be guessed	8-way	8	7 bits

eg1 :- Main memory size = 128 KB
$= 2^7 \times 2^{10}$ B
$= 2^{17}$ B
↳ 17 bits

Cache size = 16 KB
$= 2^4 \times 2^{10}$ B
$= 2^{14}$ B
↳ 14 bits

Block size = 256 B
$= 2^8$ B
↓
8 bits



Set associativity = 2
i.e. 2 lines per set

No of lines in cache $= \frac{2^{14}}{2^8} = 2^6$

No of ~~lines~~ sets $= \frac{2^6}{2} = 2^5$
↳ 5 bits

No. of tag bits = 4

Tag directory size = No. of tag bits * No of lines in cache
$= 4 * 2^6$

No of comparators = Set associativeness
= 2

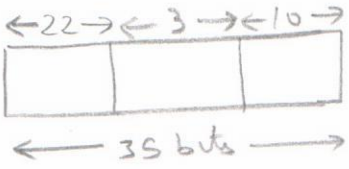
Size of each comparator = Size of tag bits
= 4 bits

eg2 :- Main memory size = 32 GB
$= 2^{35}$ B
↳ 35 bits

Cache size = 32 KB
$= 2^{15}$ B
↳ 15 bits

Block size = 1 KB
$= 2^{10}$ B
↳ 10 bits

Set associativity = 4
ie. 4 lines per set



$$\text{No of lines} = \frac{2^{15}}{2^{10}} = 2^5$$

$$\text{No. of sets} = \frac{2^5}{2^2} = 2^3 \hookrightarrow 3 \text{ bits}$$

So,
No. of tag bits = 22
Tag directory size = $22 * 2^5$ bits
No of comparators = 4
Size of each comparator = 22 bits

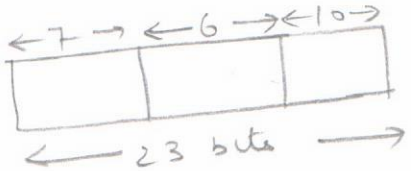
eg 3 :-

Cache size = 512 KB
$= 2^9 \times 2^{10}$ B
$= 2^{19}$ B
$\hookrightarrow$ 19 bits

Block size = 1 KB
$= 2^{10}$ B
$\downarrow$
10 bits

Set associativity = 8-way
$\downarrow$
8 lines per set

$$\text{No. of lines} = \frac{2^{19}}{2^{10}} = 2^9$$



$$\text{No. of sets} = \frac{2^9}{2^3} = 2^6$$

No. of tag bits = 7 bits

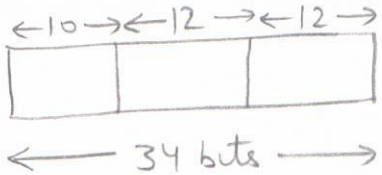
So, Main memory size = 2^{23}
= 8 MB

No. of comparators = 8
Size of each comparator = 22
Tag directory size = $7 * 2^9$

e.g 4 :- Main memory size = 16 GB
$= 2^{34}$ B
↳ 34 bits

Block size = 4 KB
$= 2^{12}$ B
↳ 12 bits

No. of tag bits = 10



Set associativity = 4-way
i.e. 4 lines per set

∴ No. of sets $= 2^{12}$ = No. of lines $= \frac{2^x}{2^{12}}$
$= 2^{x-12}$

$\Rightarrow 2^{12} = \frac{2^{x-12}}{2^2}$

$2^{12} = 2^{x-14}$

Solving :- $12 = x - 14$
$x = 26$

So, cache size $= 2^{26}$
$= 64$ MB

Cache size =
No. of set * No of lines per set * size of line

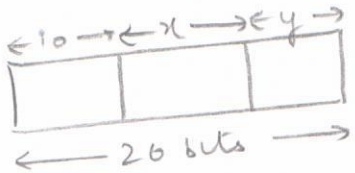
Tag directory size $= 10 * 2^{14}$

No of comparators = 4

Size of comparator = 10 bits

e.g 5 :- Main memory size = 64 MB
$= 2^{26}$ MB
↳ 26 bits

Tag bits = 10 bits



$\Rightarrow x + y + 10 = 26$
$x + y = 16$

(x be the bits required to represent set number & y be the bits required to represent block offset)

Set associativity = 4-way
i.e. 4 lines per set

Also,
No of sets = 2^x
No Block size = 2^y

So, cache size = No of sets * Size of each set * ~~No~~ Size of each lines ~~per set~~

$= 2^x * 4 * 2^y$

$= 2^{x+y+2}$

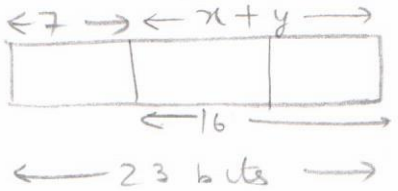
$= 2^{16+2} = 2^{18}$ B

$= 256$ KB

Block size, Tag directory size can't be guessed.
No. of comparators = 4, Size of comparator = 10 bits

e.g 6 :- Cache size = 512 KB
$= 2^9 \times 2^{10}$ B
$= 2^{19}$ B
↓
19 bits

Tag bits = 7 bits
Set associativity = 8-way
i.e. 8 lines per set



Cache size = No. of sets * Size of each set * Size of each line

$2^{19} = 2^x * 2^3 * 2^y$

$2^{19} = 2^{x+y+3}$

$x + y = 16$

So, Main memory size = 2^{23} B
$= 8$ MB

Block size & Tag directory size can't be guessed.
No. of comparators = 8, Size of each comparator = 7 bits

Comparisons between all mapping techniques

Consider an example :-

Main memory size = 4 GB
$= 2^{32}$ B
↳ 32 bits

Cache size = 4 MB
$= 2^{22}$ B
↳ 22 bits

Block size = 1 KB
$= 2^{10}$ B
↳ 10 bits

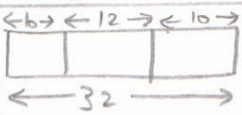
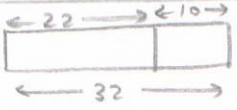
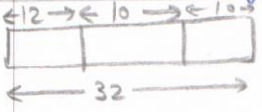
So, Physical address takes 32 bits

No of lines in cache $= \frac{CS}{BS} = \frac{2^{22}}{2^{10}} = 2^{12}$ lines

---

If we have 4-way set associative mapping, then

no. of sets $= \frac{2^{12}}{2^2} = 2^{10}$

	Direct	Associative	Set - associative (4-way)
Address format			
No of tag bits	10 bits	22 bits	12 bits
No of comparators	1	2^{12}	4
No. of lines	2^{12}	2^{12}	2^{12}
Size of comparator	10 bits	22 bits	12 bits
TOS	$10 * 2^{12}$ bits	$22 * 2^{12}$ bits	$12 * 2^{12}$ bits

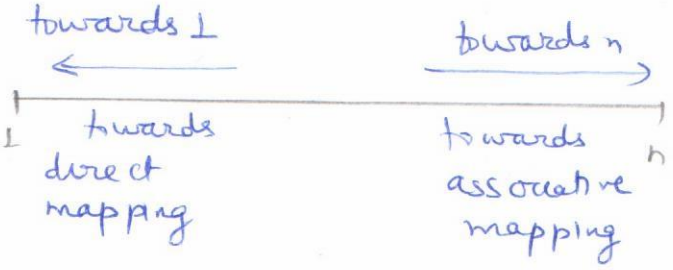
In k-way set associative mapping :-

If k=1, it becomes direct mapping

If k=n, it becomes associative mapping

where n is equal to no. of cache blocks.

Thus,



Cache Write Policy

* Before a block that is resident in the cache can be replaced, it is necessary to consider whether it has been altered in the cache but not in the main memory.
* If it has not, then the old block in the cache may be overwritten.
* If it has, that means that atleast one write operation has been performed on a word in that line of cache and main memory must be updated accordingly.
* There are two techniques for writing in cache :-

(a) Write through :- Using this technique, all write operations are made to main memory as well as to the cache, ensuring that main memory is always valid. The main disadvantage of this technique is that it generates lot of memory traffic and may create a bottleneck.

(b) Write back :- In this technique, updates are made only in cache. When an update occurs, an dirty or modified bit associated with the line is set. Thus when a block is replaced, it is written into main memory if and only if the dirty/modified bit is set. The advantage of write back policy is that it minimizes memory writes.

The disadvantage is that portions of main memory can be invalid and hence accesses by I/O modules can be allowed only through cache.

— x —

* In a bus organization in which more than one device (generally a processor) has a cache and main memory is shared, a new problem is introduced.
* If data in one cache are altered, this invalidates not only the corresponding word in main memory, but also that same word in other caches (if any other cache also contains that same word).
* A system that prevents this problem is said to maintain cache coherency.
* Possible approaches to cache coherency are —
  (a) Bus watching with write through (Snooping cache) ⇒ only possible in cache connecting to bus
  (b) Hardware transparency
  (c) Noncacheable memory
      ↳ In this case, request is not looked up in any cache. The requests are sent directly to memory.

Elements of cache Design

Although there are large number of cache implementations, there are a few basic design elements that serve to classify and differentiate cache architectures :

1. Cache Size :- The larger the cache, the larger the number of gates involved in addressing the cache. The result is that larger ch caches tend to be slightly slower than smaller ones — even when built with the same integrated circuits technology and put in the same place on chip and circuit board.

2. Mapping function :- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.
Such algorithms are of three types :-

a) Direct Mapping
b) Associative Mapping
c) Set associative mapping

3. Write policy :- As discussed earlier, there are two cache write policies —

a) Write-through policy
b) Write-back policy

4. Line size :- When a block of data is retrieved and placed in cache, not only the desired word but also some number of adjacent words are retrieved

As the block size increases from very small to larger sizes, the hit ratio will at first increase because of principle of locality but later it starts decreasing.

5. Number of caches :- There can be two categorizations :-

(a) Single or two-level

(b) Unified vs split caches

Multilevel cache

As logic density has increased, it has become possible to have a cache on the same chip as the processor : the on-chip cache. Compared with a cache reachable via an external bus, the on-chip cache reduces the processor's external bus activity and therefore speeds up execution times and increases overall system performance.

Besides this, we also need an off-chip or external cache. This results in an organization known as two-level cache with the internal cache designated as level 1 (L1) and the external cache designated as level 2 (L2).

If there is no L2 cache and the processor makes an access request for a memory location not in L1 cache, then the processor must access DRAM or ROM memory across the bus. Due to typically slower bus speed and slow memory access time, this results in poor performance.

On the other hand, if the L2 SRAM cache is used, then frequently the missing information can be quickly retrieved.

In general, the use of a second level cache improve performance. However the use of multilevel caches complicate all of the design issues related to caches including size, replacement algorithms & write policy.

Unified Vs split Cache

Unified cache refers to cache used to store references to both data and instructions.

Its advantages are—

a) For a given cache size, a unified cache has a higher hit rate than split caches because it balances the load between instruction and data fetches automatically.

b) Only one cache needs to be designed and implemented.

On the other hand, split cache refers to cache used for instructions and data separately. The split cache are used in superscalar machines which emphasize parallel instruction execution and prefetching of predicted future instructions.

Its advantages are—

a) It eliminates contention for cache between the instruction fetch/decode unit and the execution unit.

Operation of Two level Memory

Consider:

* The upper level memory (M1) is smaller, faster and more expensive (per bit) than the lower level memory (M2).
* M1 is used as a temporary store for part of the contents of the larger M2.
* When a memory reference is made, an attempt is made to access the item in M1.
* If it succeeds, then a quick access is made.
* If not, then a block of memory locations is copied from M2 to M1 and the access takes place via M1.
* Because of locality, once a block is brought into M1, there should be a number of accesses to locations in that block, resulting in fast overall service.

Operation of Two-level Memory

Consider : the upper level memory (M1) is smaller, faster and more expensive (per bit) than the lower level memory (M2). M1 is used as a temporary store for part of the contents of the larger M2. When a memory references is made, an attempt is made to access the item in M1. If this succeeds, then a quick access is made. If not, then a block of memory locations is copied from M2 to M1 and the access then takes place via M1.

Because of locality, once a block is brought into M1, there should be a number of accesses to locations in that block, resulting in fast overall service.

To, express the average time to access an item, we have :-

$$T_S = H \times T_1 + (1-H) \times (T_1 + T_2)$$
$$= T_1 + (1-H) \times T_2$$

where T_S : average access time
T_1 : access time of M1 (e.g. cache)
T_2 : access time of M2 (e.g. main memory)
H : hit ratio (fraction of time reference found in M1)

Performance of a two-level memory

Considering cost :-

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

where;

C_S : average cost per bit for combined two level memory

C_1 : average cost per bit of upper level memory M_1

C_2 : average cost per bit of lower level memory M_2

S_1 : size of M_1

S_2 : size of M_2

We would like $C_S \approx C_2$. Given that $C_1 \gg C_2$, this requires $S_1 << S_2$.

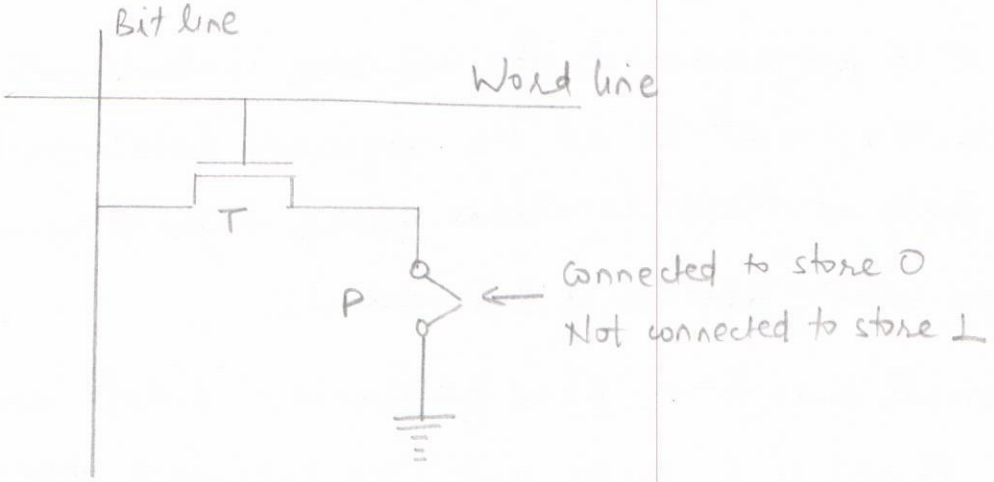
Now considering access time :-

For a two level memory to provide a significant performance improvement, we need to have T_S approximately equal to T_1 ($T_S \approx T_1$). Given that T_1 is much less than T_2 ($T_1 << T_2$), a hit ratio of close to 1 is needed.

ROM Memories

* It is non-volatile memory.
* It is used to store those instructions which must be stored permanently, like operating system software.
* Generally, the contents of such memory can be read as if they were SRAM or DRAM memories.

* But a special writing process is needed to place the information into this memory.
* Since its normal operation involves only reading of stored data, a memory this memory is called Read-only-Memory (ROM).



A ROM cell

* A logic value 0 is stored in the cell if the transistor is connected to ground at point P, otherwise a 1 is stored.
* The bit line is connected through a resistor to the power supply.
* To read the state of the cell, the word line is activated.
* Thus, the transistor switch is closed and the voltage on the bit line drops to near zero if there is a connection between transistor and ground.
* If there is no connection to ground, the bit line remains at the high voltage indicating a 1.
* A sense circuit at the end of the bit line generates the proper output value. Data are written into ROM when it is manufactured.

PROM

* Some ROM designs allow the data to be loaded by the user, thus providing a programmable ROM (PROM).
* Programmability is achieved by inserting a fuse at point P (shown in previous figure).
* Before it is programmed, the memory contains all 0s.
* The user can insert 1s at the required locations by burning out the fuses at these locations using high-current pulses.
* But, this whole process is irreversible.
* When small amount of fixed programs and data are needed to store, PROM is a faster and less expensive approach.

EPROM

* This type of ROM allows the stored data to be erased and new data to be loaded.
* Such an erasable, reprogrammable ROM is called EPROM.
* EPROMs are capable of retaining stored information for a long time, they can be used in place of ROMs while software is being developed.
* In this way, memory changes and updates can be easily made.
* The important advantage of EPROM chips is that their contents can be erased and reprogrammed.

* Erasure requires dissipating the charges trapped in the transistors of memory cells.
* This can be done by exposing the chip to ultraviolet light.
* For this reason, EPROM chips are mounted in packages that have transparent windows.

EEPROM

* One major disadvantage of EPROM is that a chip must be physically removed from the circuit for reprogramming and that its entire contents are erased by ultraviolet light.
* This drawback can be overcomed by using programmable and electrically erasable ROMs (called EEPROMs).
* These chips donot need to be removed from for erasure.
* Moreover, it is also possible to erase the cell contents selectively.
* The disadvantage of EEPROM is that different voltages are needed for erasing, writing and reading the stored data.

Do Yourself

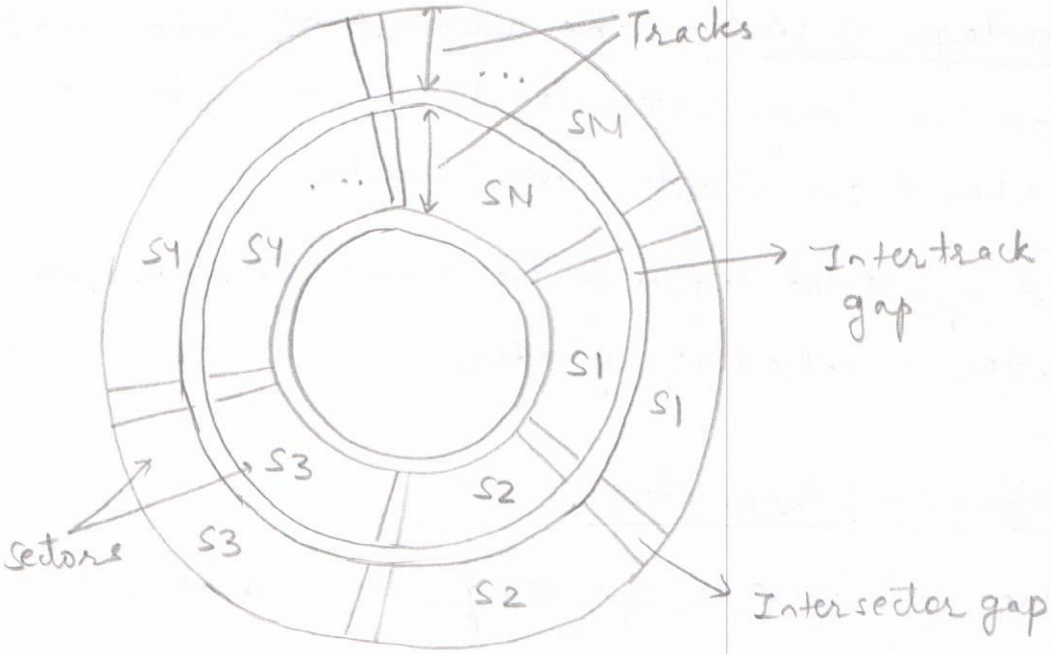
* Flash Memory

Secondary / Auxiliary storage

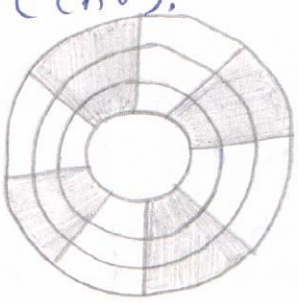
* Semiconductor memories like RAM cannot be used to provide all of the storage capacity needed in computers.
* Their main limitation is the cost per bit of stored information.
* Thus large storage requirements of most computer systems are fulfilled in the form of magnetic disks, optical disks and magnetic tapes, which are usually referred to as secondary storage devices.

Magnetic Disk

* A disk is a circular platter constructed of non-magnetic material, coated with a magnetizable material.
* Data are recorded on and later retrieved from the disk via a conducting coil named the head.
* The head is a relatively small device capable of reading from or writing to a portion of platter rotating beneath it.
* This gives rise to the organization of data on the platter in a concentric set of rings called tracks.
* There are thousand of tracks per surface.
* Adjacent tracks are separated by gaps.
* This prevents/minimizes errors due to misalignment of the head or interference of magnetic fields.



* Data are transferred to and from the disks in sectors.
* There are hundreds of sectors per track and they may be of either fixed or variable length.
* A bit near the centre of rotating disk travels past a fixed point (like read-write head) slower than a bit on the outside.
* Therefore some way must be found to compensate for the variation in speed so that the head can read all the bits at the same rate.
* This can be done by increasing the spacing between bits of information recorded in segments of disk.
* The information can then be scanned at the same rate by rotating the disk at a fixed speed known as constant angular velocity (CAV).



Disadvantage of CAV :- the amount of data that can be stored on the long outer tracks is the same as what can be stored on short inner tracks.

* The set of all the tracks in the same relative position on the platter is called as cylinder.

Disk Performance Parameters

* When the disk drive is operating, the disk is rotating at constant speed.
* To read or write, the head must be positioned at the desired track and at the beginning of desired sector on that track.
* Track selection involves moving the head in a movable head system or electronically selecting one head on a fixed-head system.

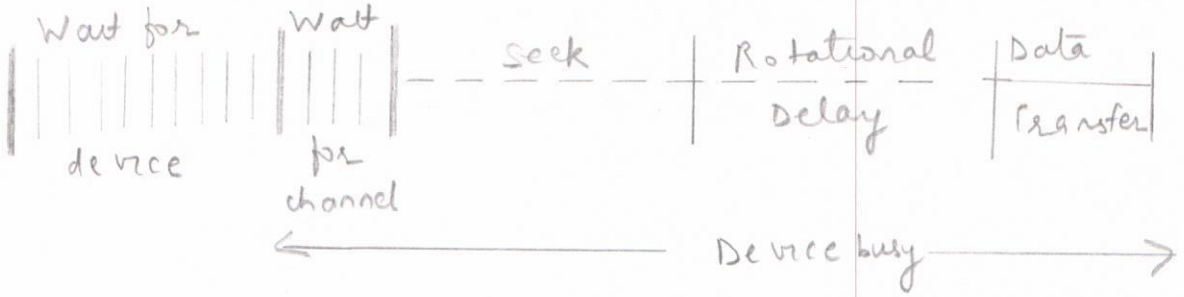
Seek time :- The time taken to position the head at the track, on a movable head system is called seek time.

Rotational Delay / Latency :- Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head. The time it takes for the beginning of the sector to reach the head is known as rotational latency.

Access time :- The time taken to get into position to read or write is called access time.

Access time = Seek time (if any) + Rotational delay

Transfer time :- Once the head is in position, the read or write operation is then performed as the sector moves under the head — the data transfer occurs.
The time required for the transfer is the transfer time.



- The transfer time to or from the disk depends on the rotation speed of the disk as :-

$$T = \frac{b}{rN}$$

where T : transfer time
b : no. of bytes to be transferred
N : no. of bytes on a track
r : rotation speed (in revolutions per sec)

- Thus, the total average access time is :-

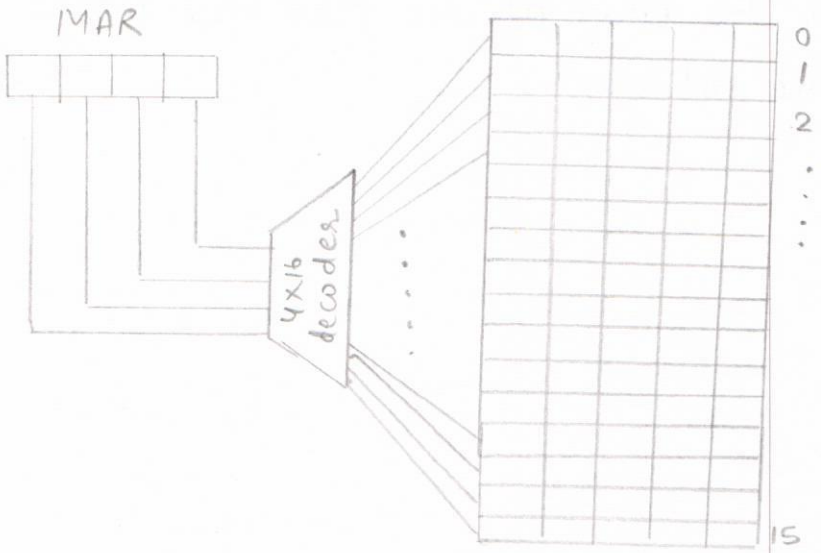
$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where T_s is the average seek time.

Q:- Explain optical storage devices & magnetic tapes.
Q:- Differentiate between CAV and CLV
Q:- Explain the concept of virtual memory.

2.5 D Memory Organization

Consider the 2D memory organization:



This 2D organization suffers from a problem of scaling. It works fine when the number of words in the memory is relatively small but creates problem when the memory is scaled up or increased in size.

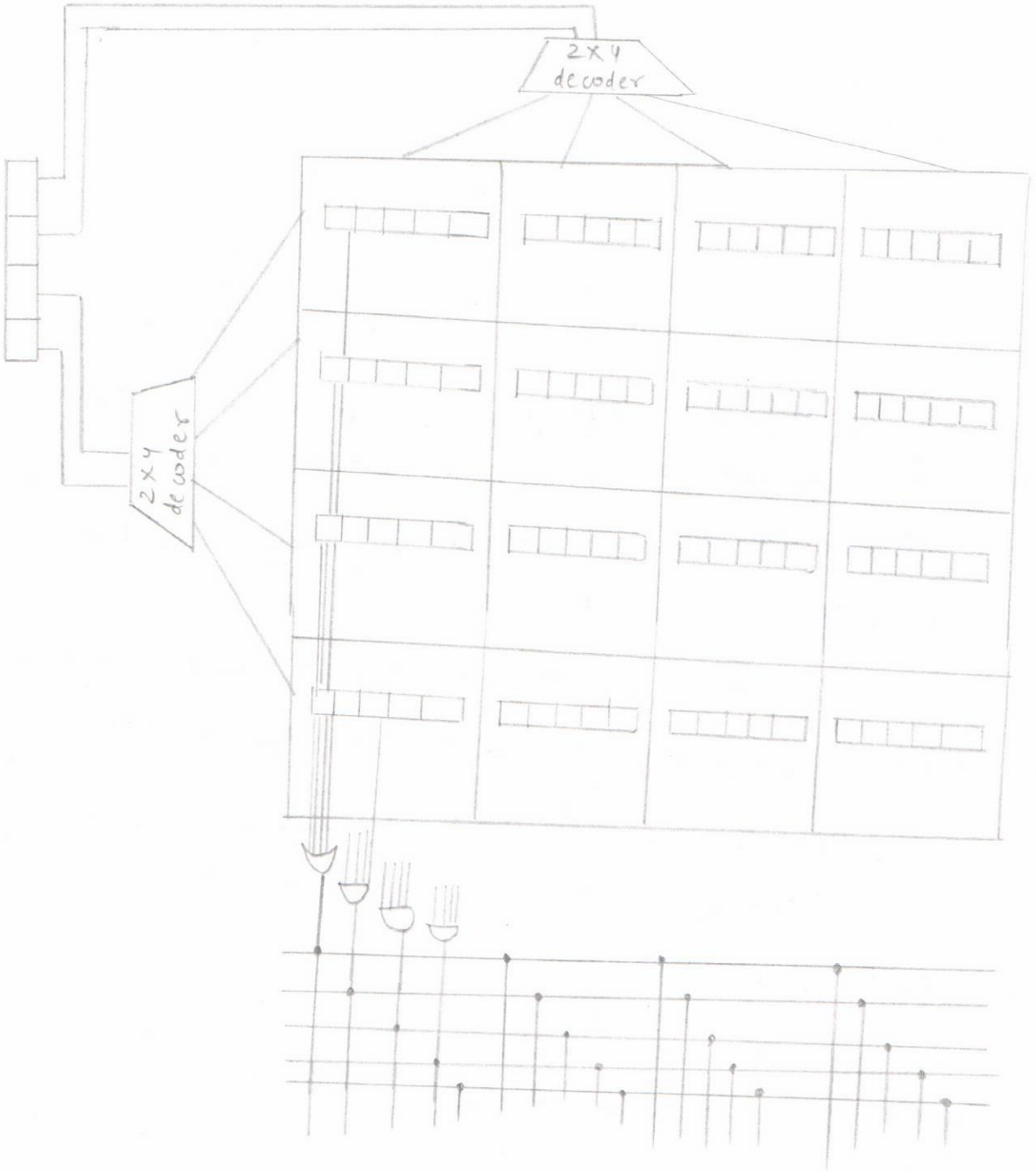
This happens because the number of word select wires is an exponential function of the size of the address.

e.g. If the MAR is 10 bits wide, it means there are 1024 words in the memory. In this case the decoder will need to output 1024 separate lines.

If MAR is 15 bits wide, there will be 32,708 wires and in case of 20 bits, this would be a million.

So, one way to tackle the exponential explosion of growth in the decoder and word select wires is to organize memory

cells into a two-dimension grid of words instead of a one dimensional arrangement. Then the MAR is broken into two halves which are fed separately into smaller decoders. One decoder addresses the rows of the grid and other decoder addresses the columns.

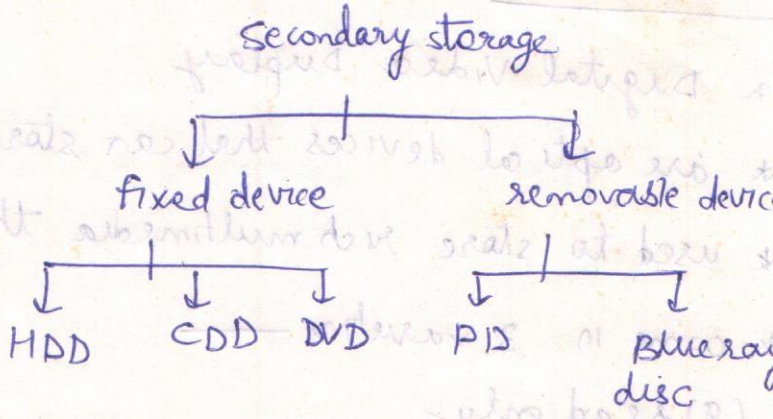


Each memory cell has an AND gate that represents the intersection of a vertical wire from one decoder and a horizontal wire from other. The output of this AND gate is the line select wire.

Thus in above figure, the total no. of word select lines goes down from 16 to 8.

If MAR had 10 bits, there would be 1024 word select wires in traditional organization, but only 64 in the 2.5D organization because each half of MAR contributes 5 address bits and $2^5 = 32$

Secondary storage



HDD

* circular disks called platters arranged one over other (almost ½" apart) around spindle
* made up of non magnetic material like Aluminium alloy, coated with 10-20 nm of magnetic material
* standard diameter is 14"
* rotate with speeds varying from 4200 rpm (PCs) to 15000 rpm (servers)
* Data stored by magnetizing or demagnetizing magnetic coating
* Magnetic reader arm is used to read data from and write data to disks

CD Drive

* circular disks that use optical rays to read & write
* Portable
* Three types —
  (a) CD-ROM (Read only M/m)
  (b) CD-R (Recordable) :- can be written; but can't be modified or deleted later
  (c) CD-RW (Re-writable) :- data can be written or deleted again & again

DVD Drive

- Digital Video Display
- are optical devices that can store 15 times data held by CDs.
- used to store rich multimedia that need high storage capacity
- come in 3 varieties ——
  (a) read only
  (b) recordable
  (c) re-writable

PDs (USB/key drive/ flash m/m)

* uses solid state m/m rather than magnetic or lasers to record data
* non-volatile

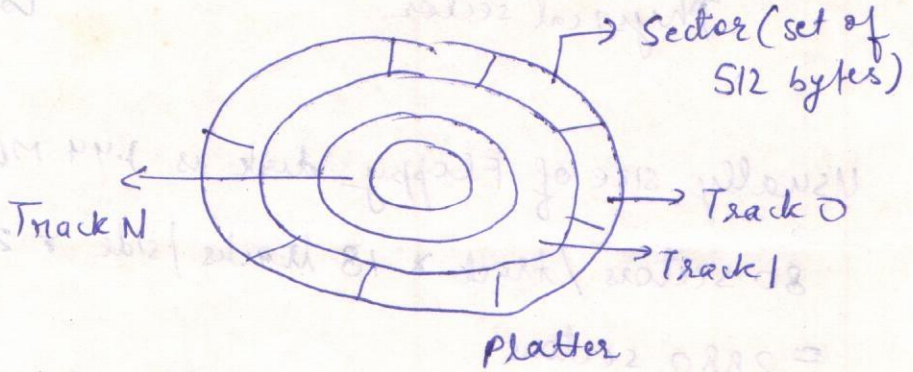
Blue-ray discs

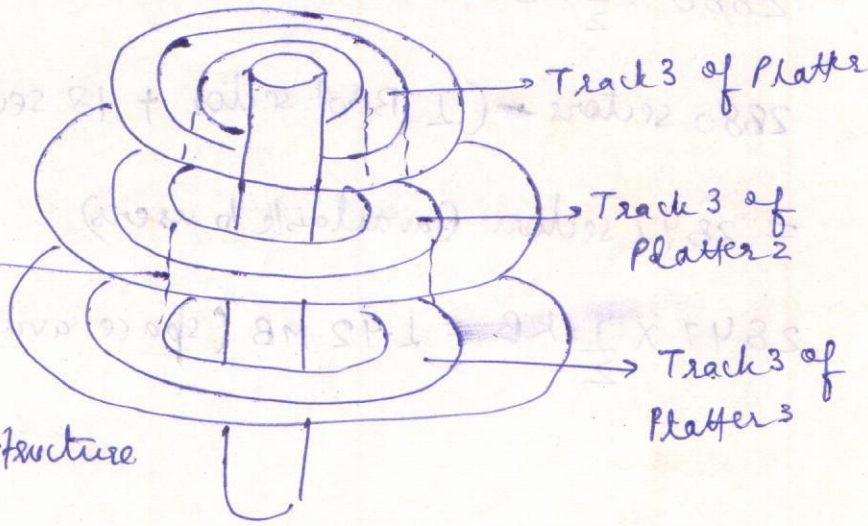
* used to store HD video & other multimedia files
* uses shorter wavelength laser as compared to CD/DVD
* enables writing arm to focus more tightly on disk & hence pack more data
* can store upto 128 GB data

Comparison b/w Primary & Secondary storage

	Primary	Secondary
Cost	Expensive	Less than Primary
Capacity	Limited	Nearly limited
Access Time	In billionth of second	In millionth of second
Processing	Directly Accessible	Routed through Primary storage

Cluster :- smallest storage unit onto a hard disk.
↳ can consist of 1 sector or group of sectors depending on file allocation s/s

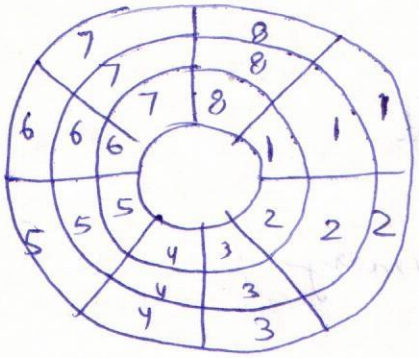




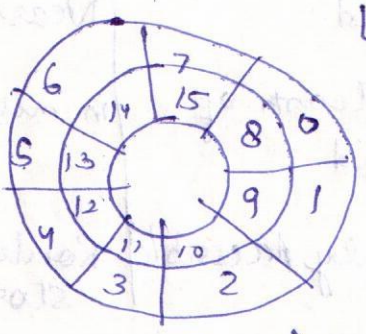
∴ Cylinder X where X is the track number

Advantage of using cylinder to read/store data is actuator need not to be moved when data is stored in same tracks of different platters

↳ results in faster access time



Physical sector



Logical sector

Logical sector 0
↳ boot sector which contains all I/f regarding disk medium characteristics

Usually size of Floppy disk is 1.44 MB —

80 sectors / track * 18 tracks / side * 2 sides

= 2880 sectors

$2880 \times \frac{1}{2} KB = 1.44 MB$

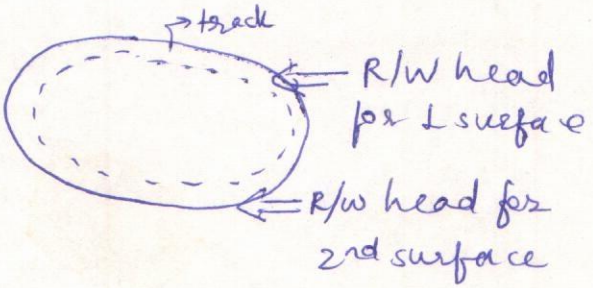
2880 sectors — (1 Boot sector + 18 sectors for 2 FAT copies + 14 sectors by root directory)

= 2847 sectors (available to users)

$2847 \times \frac{1}{2} KB = 1.42 MB$ (space available to users)

Q:- A hard disk has 63 sectors per track; 10 platters each with 2 recording surfaces & 1000 cylinders. The address of each sector is given as a triple <c,h,s> where c is cylinder no., h is surface number; s is sector number. Thus, the 0th sector is addressed as <0,0,0>, the 1st sector as <0,0,1> and so on.

The address <400,16,29> corresponds to sector number?



So, we have 20 surfaces in total

R/W heads for each track in a cylinder move in same position

↳ Low seek latency

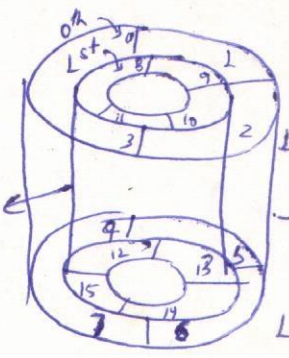
<0,0,0>

<1,0,0>

Capacity of each cylinder ⇒ 63 sectors per track * 20 surfaces

= 1260 sectors

Let



400 * 1260

= 504000 sectors numbered before 400th cylinder.

In 400th cylinder, ~~each surface~~ sectors

= 16 * 63

= 1008

504000
1008
505008
29
505037

Q.