The Memory System

* The memory unit is an essential component in any digital computer since it is needed for storing Programs and data.
* The memory unit that communicates directly with the CPU is called the main memory.
* A more economical and low-cost storage devices can be used to serve as a backup for storing the information that is not currently used by the CPU. Devices that Provide back-up storage are called Auxiliary memory.
* Only Programs and data currently needed by the Processor reside in main-memory. All other information is stored in Auxiliary memory and transferred to main memory when needed.
* A special very-high speed memory called a cache is used to increase the speed of Processing by making current Programs and data available to the CPU at a rapid rate.
* The cache is used for storing segments of Programs currently being executed in the CPU and temporary data frequently needed in the Present calculations.

SEMICONDUCTOR - RAM (Main) - Memories

* A memory unit is called Random-access memory RAM if any location can be accessed for a Read or Write operation in some fixed amount of time that is independent of the location's address.

* Access time on the devices such as magnetic disks and tapes depends on the address or position of the data.

* Memory access time is the time that elapses between the initiation of an operation and completion of that operation. For example the time between the Read and the MFC signals. This is also called speed of memory.

* Memory cycle time is the minimum time delay required between the initiation of two successive memory operations. For example, the time between two successive Read operation

* RAM was used to refer to a random-access memory, but now it is used to designate a read/write memory to distinguish it from a read-only memory, although ROM is also random access.

* RAM is used for storing of the Programs and data that are subject to change. ROM is used for storing Programs that are Permanent and that don't change in value once the production of the computer is complete

Latency and Bandwidth :-

* Transfers between the memory and the Processor involve single words of data or small blocks of words.
* Large blocks, constituting a page of data, are transferred between memory and the disks.

* The speed and efficiency of these transfers have a large impact on the performance of a computer system.
* A good indication of performance is given by two Parameters (i) Latency (ii) Bandwidth.

Latency: The amount of time it takes to transfer a word of data to or from the memory. In block transfers, Latency is the time it takes to transfer the first word of data.

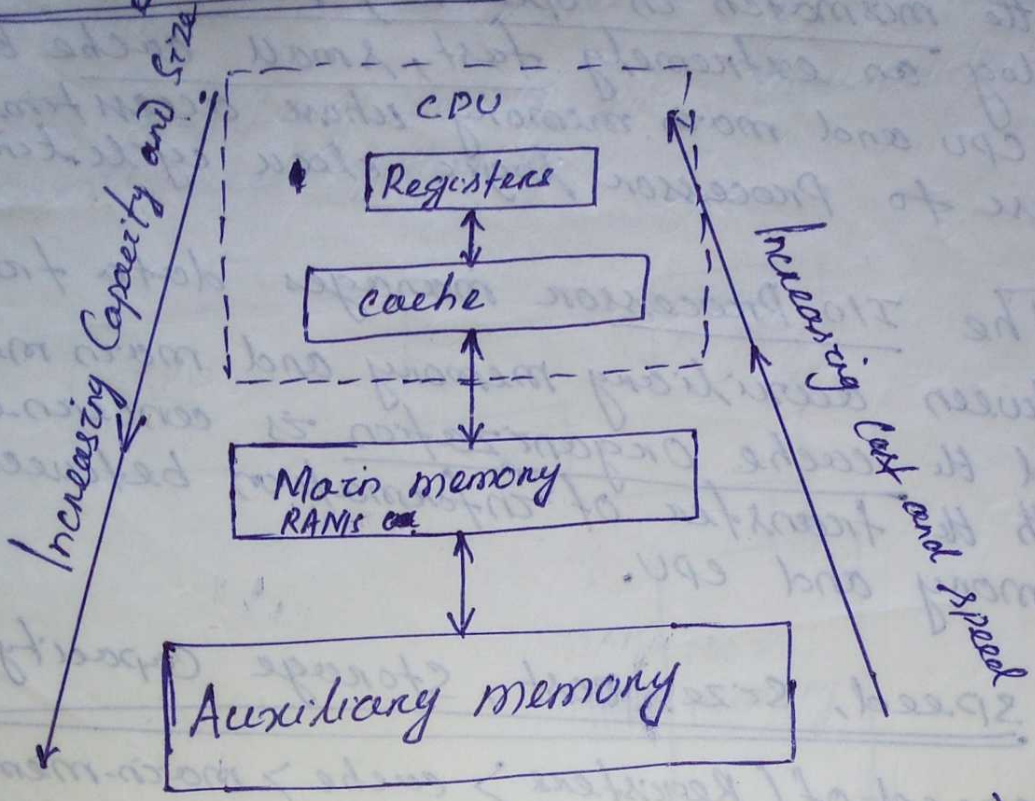
Bandwidth: The no of bits or bytes that can be transferred in one second.

The bandwidth of a memory unit depends on the speed of access to the stored data and on the number of bits that can be accessed in Parallel.

The bandwidth is the product of the rate at which data are transferred and the width of the data bus (the no of wires).

~~Memory Hierarchy~~:

Memory Hierarchy:



(Memory Hierarchy in a computer System)

* The main memory occupies a central role/position by being able to communicate directly with the CPU and with auxiliary memory.
* When Programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.
* Programs not currently needed in main-memory are transferred into Auxiliary memory to Provide space for currently used Programs and data.
* Processor logic is much faster than main memory access time, so the processing speed is limited Primarily by the speed of main memory.

* Therefore, a technique used to compensate for the mismatch in operating speeds is to employ an extremely fast, small cache between the CPU and main memory whose access time is close to Processor logic clock cycle time.

* The I/O Processor manages data transfer between auxiliary memory and main memory, and the cache organization is concerned with the transfer of information between main memory and CPU.

* speed, size, cost, storage capacity

(i) speed-of (Registers > cache > main-memory > Auxiliary memory).

(ii) Size-of (Registers < cache < main-memory < Auxiliary-memory)

(iii) cost$_{\text{Per bit}}$-of (Registers > cache > main-memory > Auxiliary-memory)

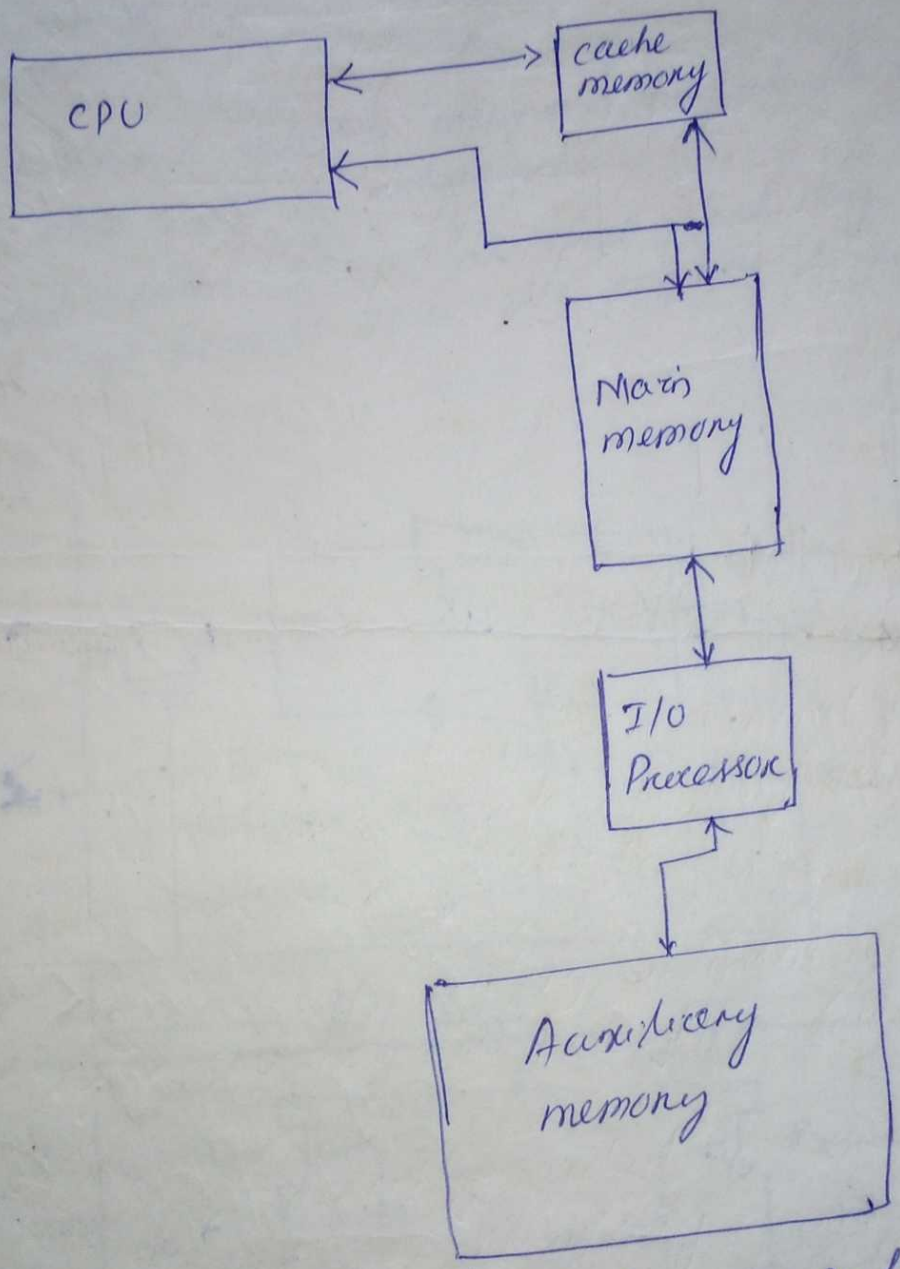
(iv) Storage-Capacity (Registers < cache < main memory < Auxiliary-memory.

* The goal of using a memory hierarchy is to obtain the ~~height~~ highest possible average access speed while minimizing the total cost of the entire memory system.

* The cache holds those parts of the Program and data that are currently being executed and most heavily used by the CPU.

* The main-memory holds only the Programs and data that are currently needed by the Processor.

* The Auxiliary memory holds those parts that aren't Presently used by the CPU.



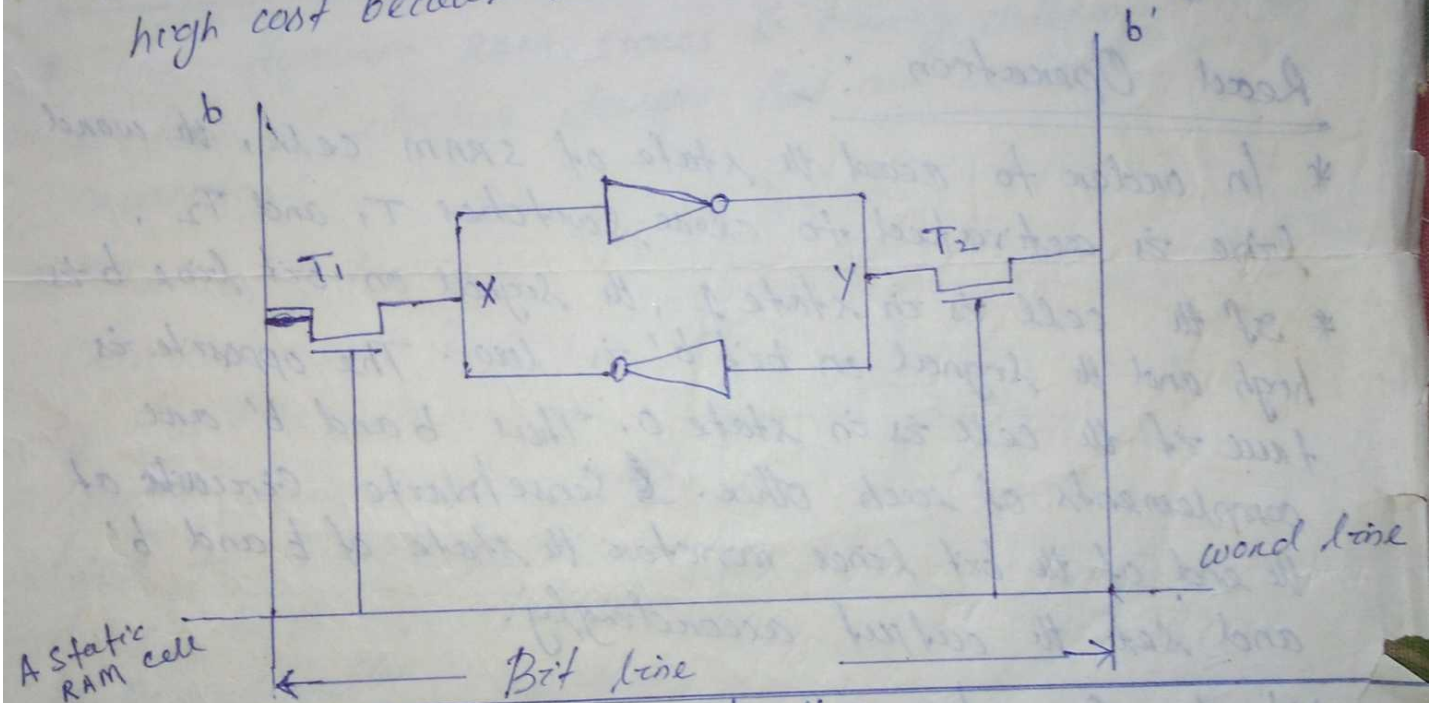
Memory Hierarchy in a computer System

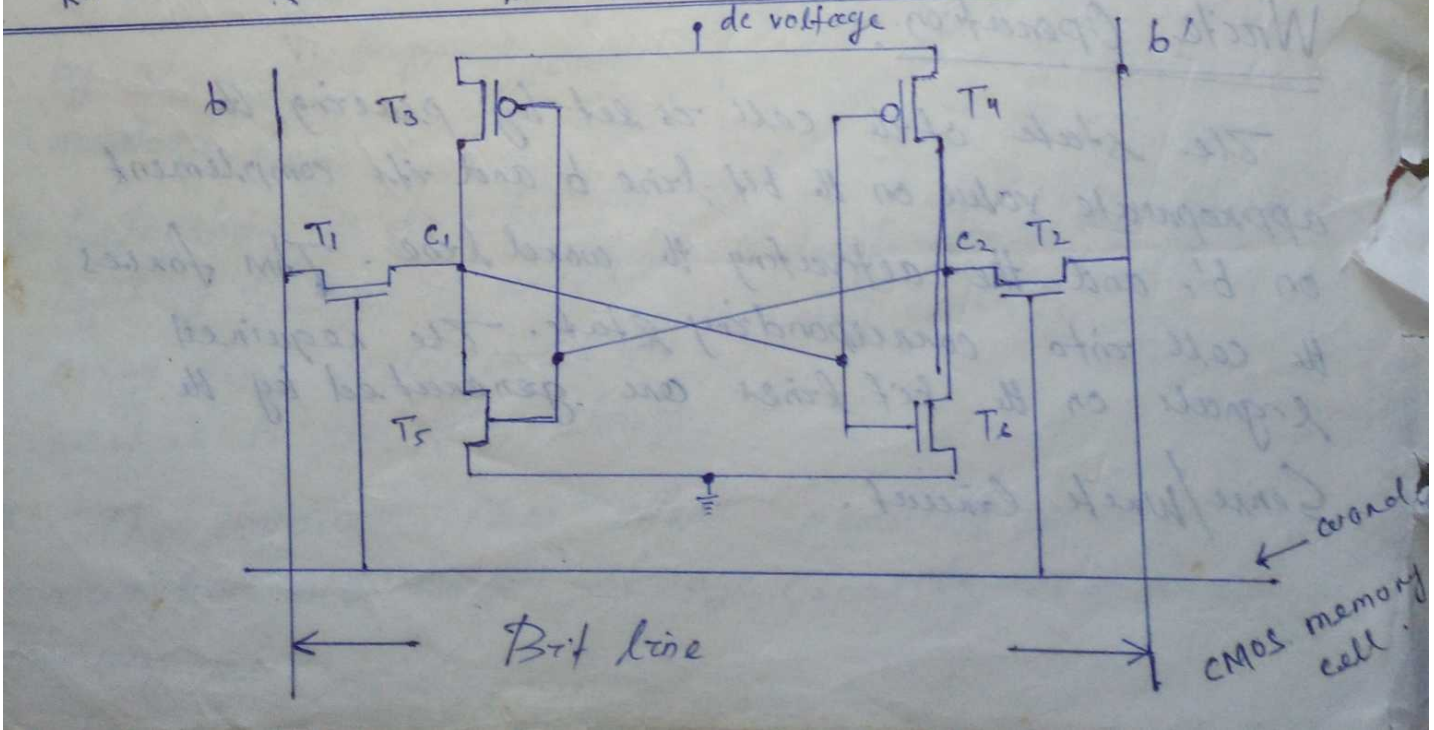
RAM CHIPS

Integrated circuit RAM chips are available in two possible operating modes.
(i) Static (ii) Dynamic

Static Memories: 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.

* SRAMs are volatile memories because their contents are lost when power is interrupted.
* Static RAMs (SRAMs) are fast, but they come at a high cost because their cells require several transistors.





* In the first fig, 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 the control of the word line.

* When the word line is at ground level, the transistors are turned off and the latch retains its state.

* For example, 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.

Read Operation :-

* In order to read the state of SRAM cell, the word line is activated to close switches T_1 and T_2.

* If the cell is in state 1, the signal on bit line b is high and the signal on bit b' is low. The opposite is true if the cell is in state 0. Thus b and b' are complements of each other. Sense/Write Circuits at the end of the bit lines monitor the state of b and b' and set the output accordingly.

Write Operation :

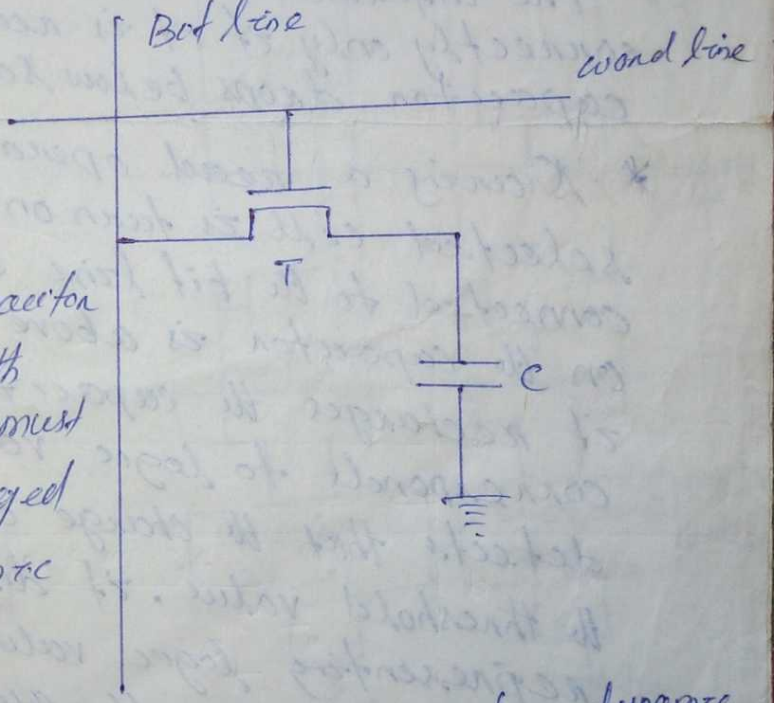
The state of the cell is set by placing the appropriate value on the bit line b and its complement on b', and then activating the word line. This forces the cell into corresponding state. The required signals on the bit lines are generated by the Sense/Write circuit.

CMOS cell: A cmos cell is given in the 2nd fig

* Transistor pairs (T_3, T_5) ad (T_4, T_6) form the inverters in the latch.
* The power supply voltage, V_{supply}, is 5V in older CMOS SRAMs or 3.3V in new low-voltage versions.
* A major advantage of CMOS SRAMs is their very low power consumption because current flows in the cell only when the cell is being accessed.
* Static RAMs are having fast access time.

Dynamic RAMS :

* The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.
* The capacitor are Provided inside the chip by MOS transistors. The stored charge on the capacitor tend to dis-charge with time, So the capacitors must be Periodically rechanged by refreshing the dynamic memory.
* Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge.



A single-transistor dynamic memory cell.

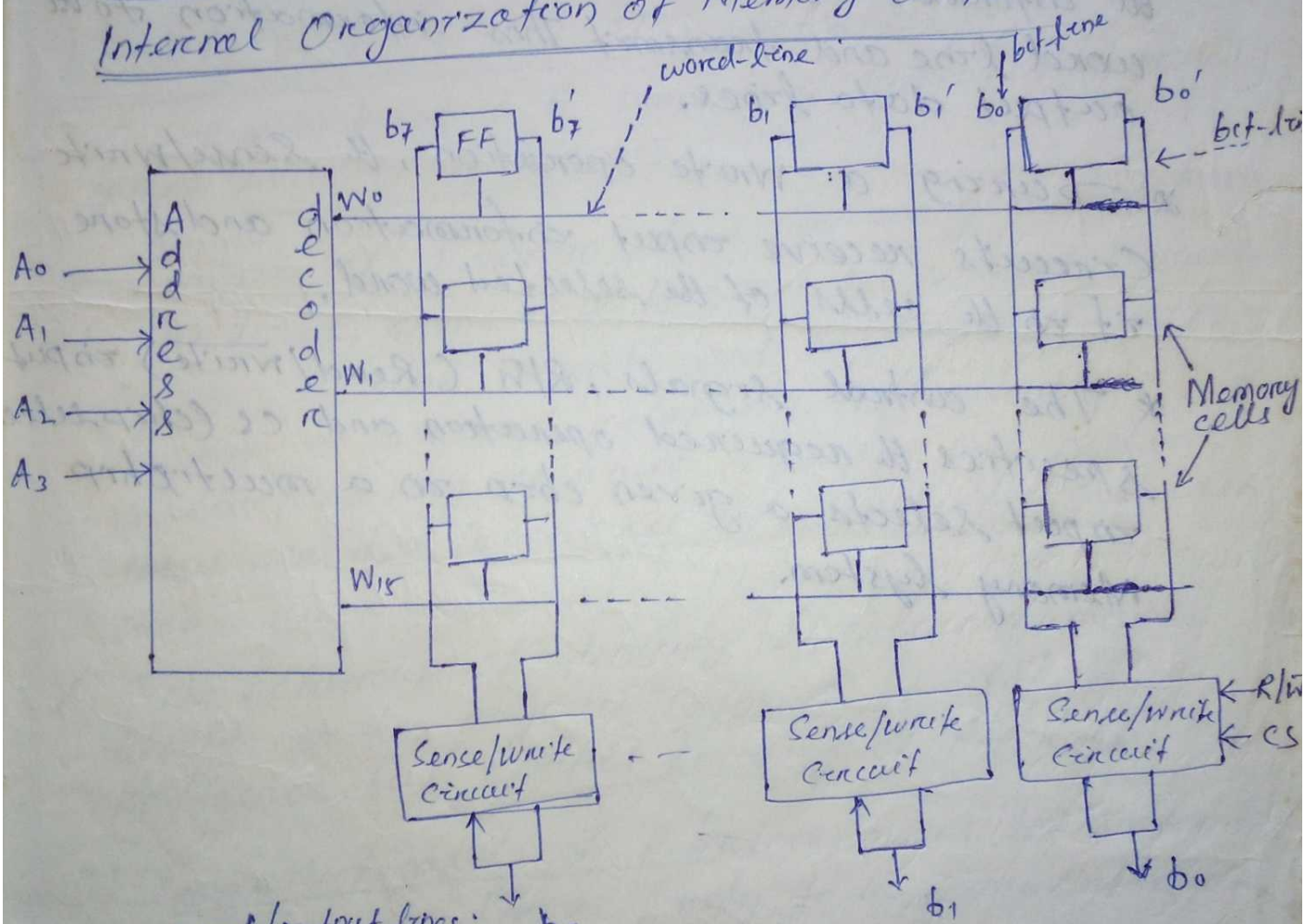
* The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.

* Information is stored in dynamic memory cell in the form of a charge on a capacitor, and this charge can be maintained for only tens of milliseconds.
* Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value.
* In order to store information in this cell, transistor T is turned on and an appropriate voltage is applied to the bit line. This causes desired amount of charge to be stored in the capacitor.
* After the transistor is turned off, the capacitor begins to discharge because of its own leakage resistance.
* The information stored in the cell can be refreshed correctly only if it is read before the charge on the capacitor drops below some threshold value.
* During a read operation, the transistor in the selected cell is turn on. A sense amplifier connected to the bit line detects whether the charge on the capacitor is above the threshold value. If it recharges the capacitor to the full charge that corresponds to logic value1. If the sense amplifier detects that the charge on the capacitor is below the threshold value, it discharges the capacitor representing logic value 0. So, reading the contents of the cell automatically refreshes the contents of the entire row.

* The select terminal, as the name suggests, selects a memory cell for a read or write operation.
* The control terminal indicates read or write.
* For writing, the data-in terminal provides an electrical signal that sets the state of the cell to 1 or 0.
* For reading, this terminal is used for output of the cell's state.

SEMICONDUCTOR RAM MEMORIES

Internal Organization of Memory chips



Data input/output lines: b_7

(Organization of bit cells in a memory chip.)

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

NOTE: The above memory circuit stores 128 bits (~~16 words x 8~~ 16 words x each word contains 8 bits = 128-bits) and requires (i.e 16 words x each word contains 8 bits = 128-bits) and requires 14 external connections ~~for data~~ for address (4) + data (8) ~~+~~ + control lines (2) = 14

* Each row of cells constitutes a memory word, and all cells of the row are connected to a common line referred to as the word-line, which is driven by the 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 a Write operation, the Sense/Write circuits receive input information and store it in the cells of the selected word.

* The control signals, R/$\overline{W}$ (Read/$\overline{Write}$) input specifies the required operation and CS (Chip Select) input selects a given chip in a multichip Memory System.

CACHE MEMORIES

Introduction

* The speed of main memories is very slow in comparison with the speed of the Processors.
* For good performance, the Processor can't spend much of its time waiting to access instructions and data in main memory.
* An efficient solution to use a fast cache memory between the Processor and the main memory, which makes the main memory appear to the Processor to be faster than it really is.

Locality of Reference: The effectiveness of the cache mechanism is based on the property of computer Programs called **Locality of Reference**.

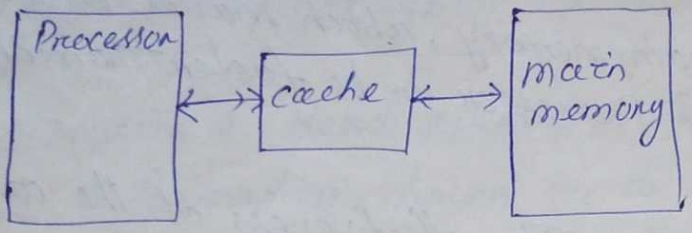
* Analysis of Program shows that many instructions in localized areas of the Program are executed repeatedly during some time period, and the remainder of the Program is accessed relatively infrequently. These instructions may constitute a simple loop, nested loops, or few procedures that repeatedly call each other.
* The behaviour of locality of reference makes itself in two ways (i) **Temporal** Locality of Reference (ii) **Spatial** Locality of Reference.

(i) **Temporal Locality of Reference** means that a recently executed instruction is likely to be executed again very soon. Examples are: Loops, Nested Loops, Procedures that repeatedly call each other.

(ii) The **spatial Locality of Reference** means that instructions in close proximity to a recently executed instruction are also likely to be executed soon. That is why, instead of fetching just one item from the main-memory to the cache,

* it is useful to fetch several items that reside at adjacent addresses as well, called blocks

* The term 'block' is used to refer to a set of contiguous address locations of some size. For a cache block size is called cache_line.



* When a read Request is received from the proces the contents of a block of memory words containing th location specified are transferred into the cache.

* Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory. The correspondence between the main memory blocks and those in the cache is specified by a mapping function.

* When the cache is full and a memory word that is 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.

* The Processor issues Read or write requests without knowing the existence of the cache. It simply generates addresses that refer to locations in the memory. If the requested word currently exists in the cache then it is said a read-hit or write-hit or simple hit Otherwise called a miss.

* For a write operation, the system can proceed in two ways (i) Write-through or (ii) Write-back protocol.

(i) Write-through Protocol: the cache location and the main-memory location are updated simultaneously.

(ii) Write-back Protocol:- In this technique, the cache-location is only updated and marked it with an associated flag-bit, called dirty or modified bit. The main memory location of the word is updated later. The main memory is updated when the block containing this marked word is removed from cache to make room for a new block.

* The write-through Protocol is simpler, but it results in unnecessary Write operation to the main memory when a given cache word is updated several times during its cache residency.

* The Write-back Protocol may also result in unnecessary write operations 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 was in the cache.

* During a read-miss, the block of words that contains the requested word is copied from the main-memory into the cache. After the entire block is loaded into the cache, the particular word requested is forwarded to the Processor. Alternatively, this word may be sent to the Processor as soon as it is read from the main-memory. This

approach is called load-through or early restart.

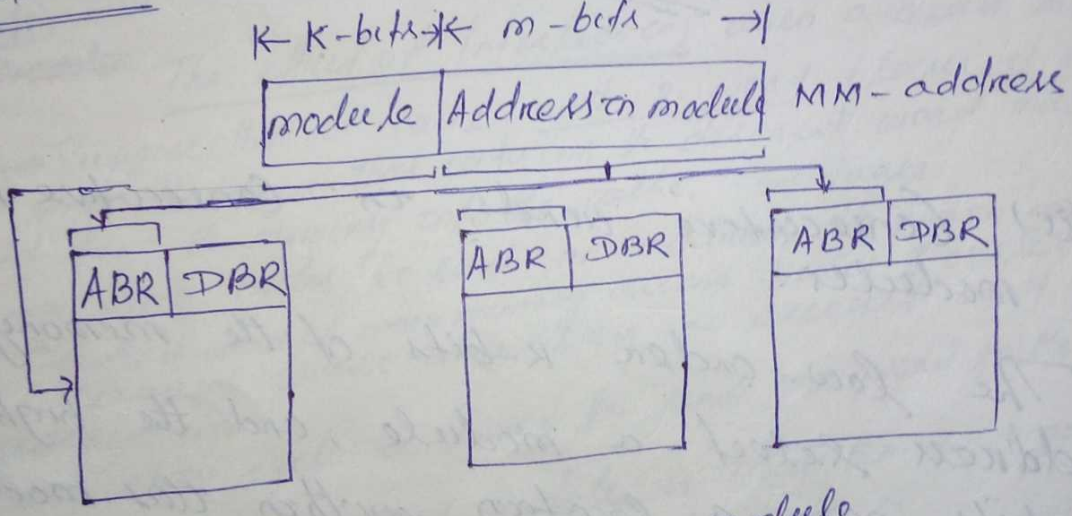
* During a Write-miss, if Write-through Protocol is used, the information is written directly into the main-memory. In the case of the Write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is over written with new information.

Memory Interleaving:

Main-memory of a computer can be structured as a collection of physically separate modules, each with its own address buffer register (ABR) and data buffer register (DBR).

* Memory access operations may proceed in more than one module at the same time, so that the aggregate rate of transmission of words to and from the main memory can be increased.
* Two methods of memory module addressing is given below.

method-1



(i) consecutive words in a module.

* In this above module addressing, the memory address generated by the processor (the decoded, the high order K-bits name one of n modules, and the low-order m-bits name a particular word in that module.

* In this case, when consecutive locations are accessed, only one module is involved. But at the same time, the devices with direct-memory-access (DMA) ability may be access information in other modules.

Second-Method:

This method is the most effective way to address the memory modules and this method is called memory interleaving.

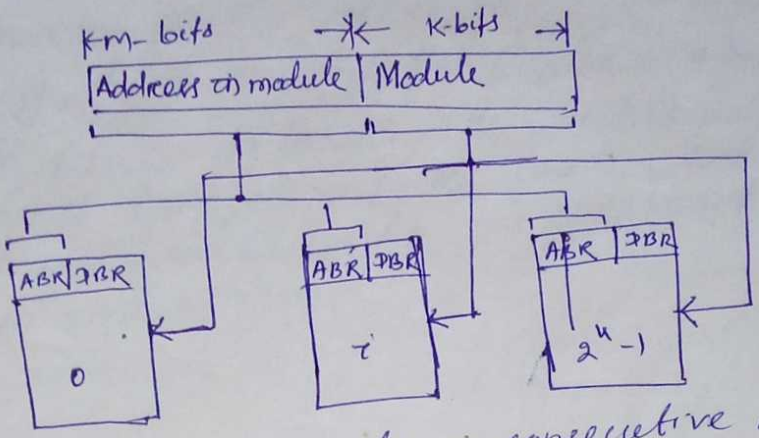
Diagram

(ii) Consecutive words in Consecutive memory modules.

* The low-order k-bits of the memory address select a module, and the high-order m-bits name a location within that module. In this way, consecutive addresses are located in successive modules.

* So, the computer system generates requests for access to consecutive memory locations can keep several modules busy at any one time resulting faster access to a block of data

* To implement the interleaved structure there must be 2^k modules, otherwise there will be gaps of non-existent locations in the memory address space.



Addressing multiple-module memory systems.

consecutive words in consecutive modules

Example: The effect of Interleaving when a read miss occurs

Suppose that a cache with 8-word blocks is used. on read miss, the block that contains the desired word must be copied from the memory into the cache. It takes

(i) one clock cycles to send an address to main memory.
(ii) The first word of main memory access takes 8 cycles
(iii) subsequent words of the block are accessed in 4 clock cycles per word.
(iv) one clock cycle is needed to send one word to the cache.

(a) If a single memory module is used, then the time needed to load the desired block into the cache is

$$1 + 8 + (7 \times 4) + 1 = 38 \text{ cycles.}$$

(b) If the memory is constructed as four interleaved modules.

* when the starting address of the block arrives at the memory all four modules begin accessing required data, using high-order bits of the address. After 8 clock cycles, each module has one word of data in its DBR. Then words are transferred to the cache, one word at a time, during next 4 clock cycles.

$1 + 8 + 4 + 4 = 17$ cycles.

5.11

* During a Write operation, if the addressed word is not in the cache, a Write miss occurs. Then, if the write-through Protocol is used, the information is written directly into the main memory. In the case of the write-back Protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

Mapping Function:

We will use a small example, to discuss possible methods for specifying where memory blocks are placed in the cache.

Consider a cache consisting of 128-blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by a 16-bit address. The main memory has 64K words, which is 4K blocks of 16-words each.
Assume that consecutive addresses refer to consecutive words.

Direct-mapping: In this technique, block j of the main memory maps onto block j modulo 128 of the cache.

* Whenever one of the main memory blocks 0, 128, 256 is loaded in the cache, it is stored in the cache block 0.
  $0 \% 128 = 0$, $128 \% 128 = 0$
  $256 \% 128 = 0$ etc.

* Similarly Blocks 1, 129, 257, ---- are stored in cache block 1 and so on.

$1 \% 128 = 1$, $129 \% 128 = 1$ ...

* placement of a block in the cache is determined from the memory address. The memory address can be divided into three fields.

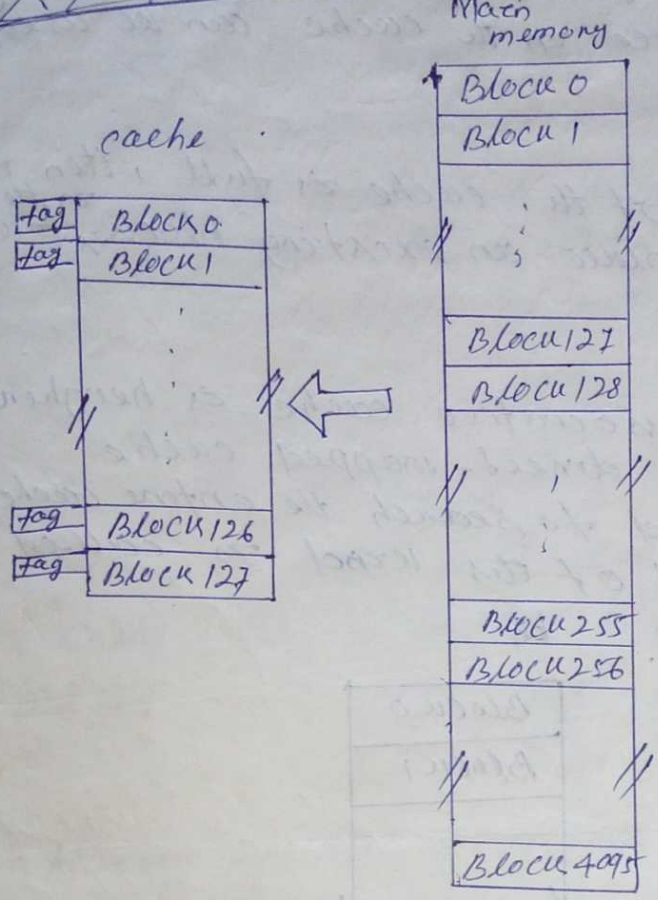
Tag	Block	Word
5	7	4

← main memory address.

(Direct-mapped cache)

* The lower-order 4-bits select one of 16-words in a block. i.e $= 2^4 = 16$ block
* The 7-bit cache field determines the cache position in which this block must be stored.
  i.e: $2^7 = 128$
* The high-order 5-bits of the memory address of the block are stored in 5-tag bits associated with its location in the cache. They identify which of 32 blocks that are mapped into this cache position are currently resident in the cache. i.e, $4095/128 = 2^{12}/2^7 = 2^5$
* If the high-order 5 tag bits and 7-bit each block field matches with each address generated by the Processor, then the desired word is in that block of the cache.

Associative Mapping :-



(Direct-mapped cache)

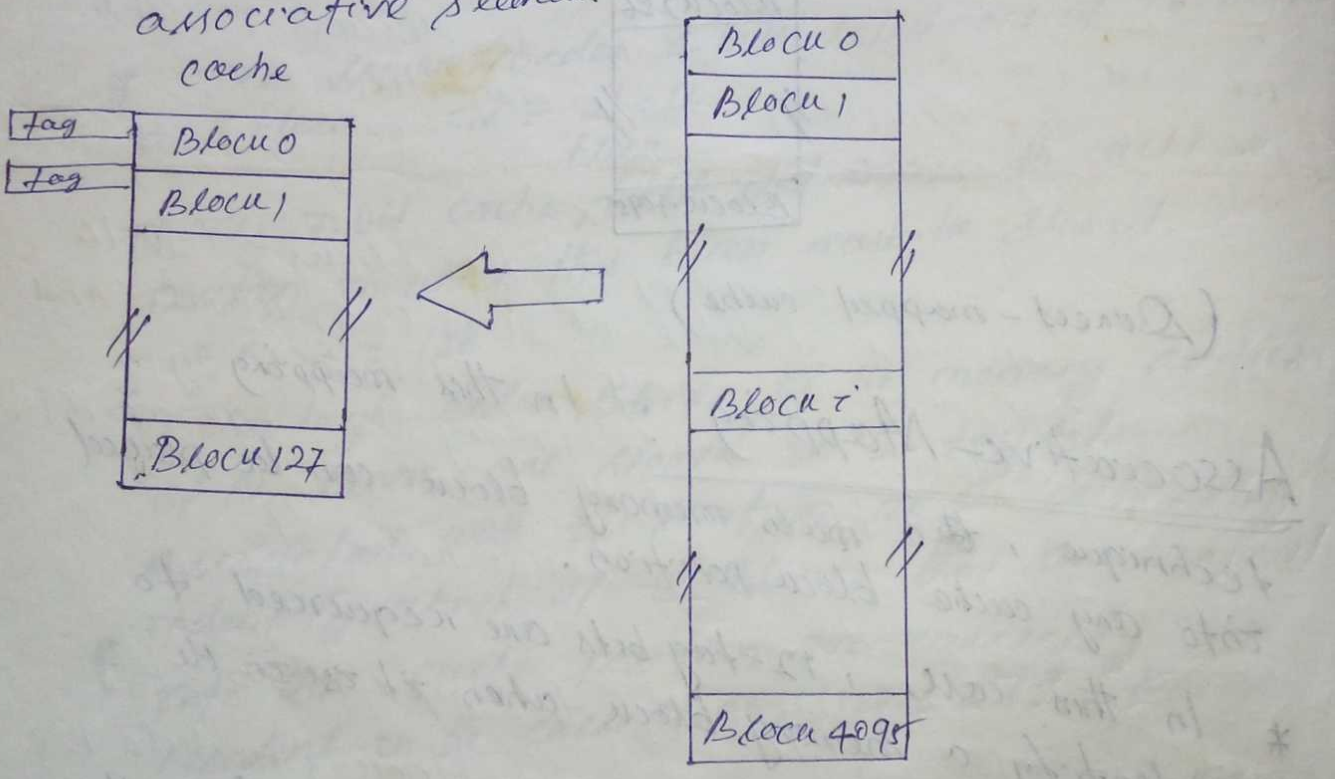
Associative-Mapping : In this mapping technique, a main memory block can be placed into any cache block position.

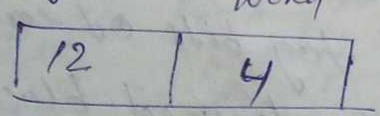
* In this case, 12-tag bits are required to identify a memory block when it is in the cache.
* The tag bits of an address received from the Processor are compared to the tag bits of each block of the cache to see if the desired block is present. This is called the associative-mapping technique.

* This gives the complete freedom in choosing the cache location in which to place the memory block. So, the space in the cache can be used more efficiently.

* ~~A new block that~~ If the cache is full, then it in the cache is necessary to replace an existing block for a new block.

* The cost of an associative cache is heighen than the cost of a direct-mapped cache because of the need to search the entire cache blocks. A search of this kind is called associative search.





← main memory address.

(Associative-mapped cache)

22.3

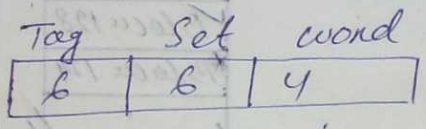
5.12

Set-Associative Mapping :

Set-Associative mapping is a combination of the direct- and associative-mapping techniques.

* Blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set.

* For example: a ~~2~~ cache with two blocks per set, In this case, memory blocks 0, 64, 128 - 4032 map into cache set 0, and they can occupy either of the two block positions within this set.

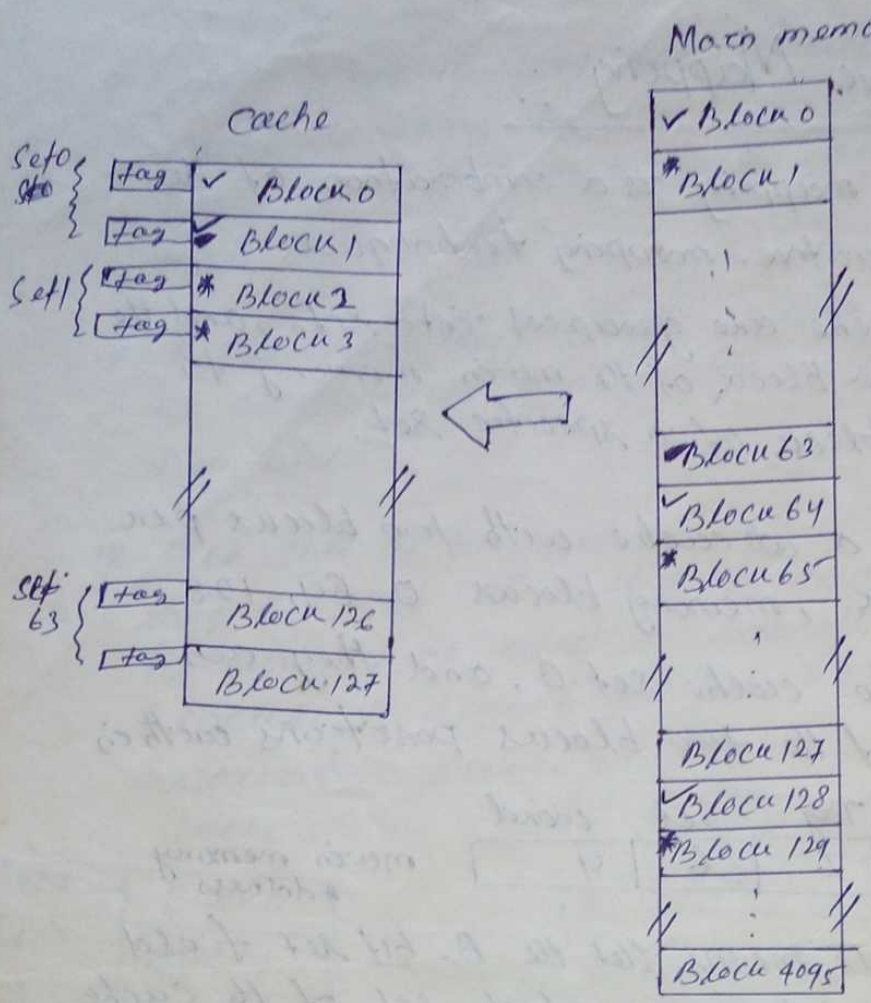


*

* Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block.

* The tag field of the address must be associatively compared to the tags of the two blocks of the set to check if the desired block is present.

* In Set-Associative Mapping, the contention problem of the direct method is ~~eased~~ eased by having a few choices for block placement ~~and~~ and of the same time, the hardware cost is reduced by decreasing the size of the associative Search.


Main memory
Cache
Set 0
tag
Block 0
tag
Block 1
Set 1
tag
Block 2
tag
Block 3
Set 63
tag
Block 126
tag
Block 127
Block 0
Block 1
Block 63
Block 64
Block 65
Block 127
Block 128
Block 129
Block 4095

5.13

Replacement Algm

* In direct mapped cache, the position of each block is predetermined, hence no replacement strategy exists
* In associative and set-associative, when the cache is full and a memory word (instructions and data) that isn't in the cache is referred, 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 Algm.
* Various page replacement algms are.

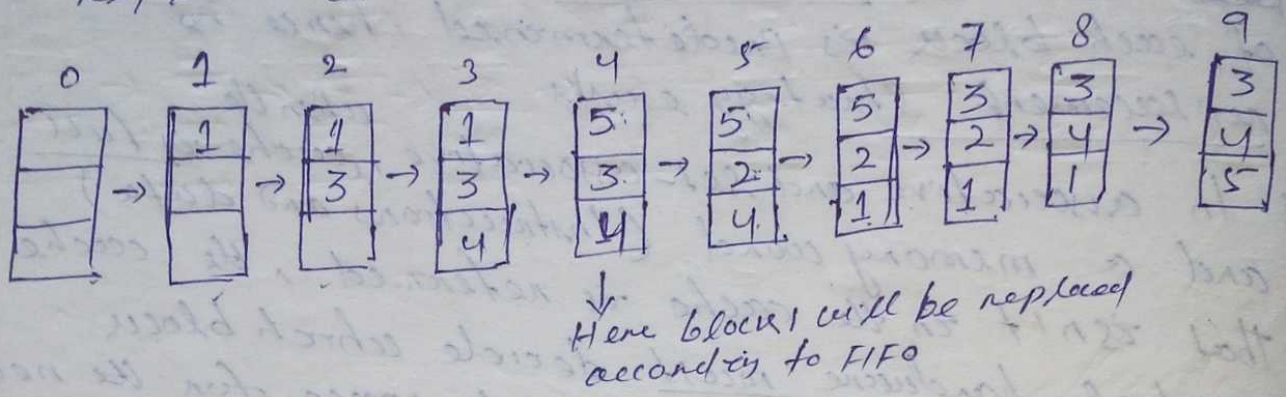
1. FIFO (First-in-First-out)
2. LIFO (Last-in-first-out)
3. MFU (Most-frequently used)
4. LFU (Least frequently used)
5. MRU (Most-recently used)
6. LRU (Least recently used)
7. ~~Optimal algm~~

1. FIFO

* Here the ~~page~~ block which was brought first must be replaced first.
* It is represented by using a queue.

Ex: Let is retrieve the blocks in the following order,
i.e, 1, 3, 4, 3, 1, 5, 2, 1, 1, 3, 4, 5

* Let the no of blocks in cache is 3 and initially the cache is empty. Then we can replace in the following way.

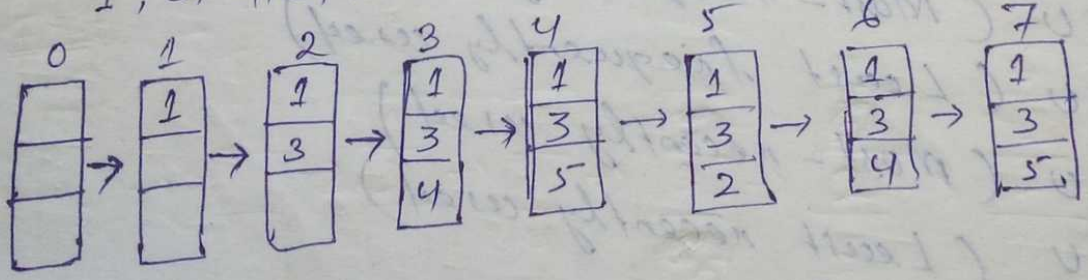


Total no of miss = 9

2. LIFO: * Here the block which was brought last must be replaced first.
* It is implemented by using a stack i.e top of the stack represents the block to be replaced

Let's consider the above example.

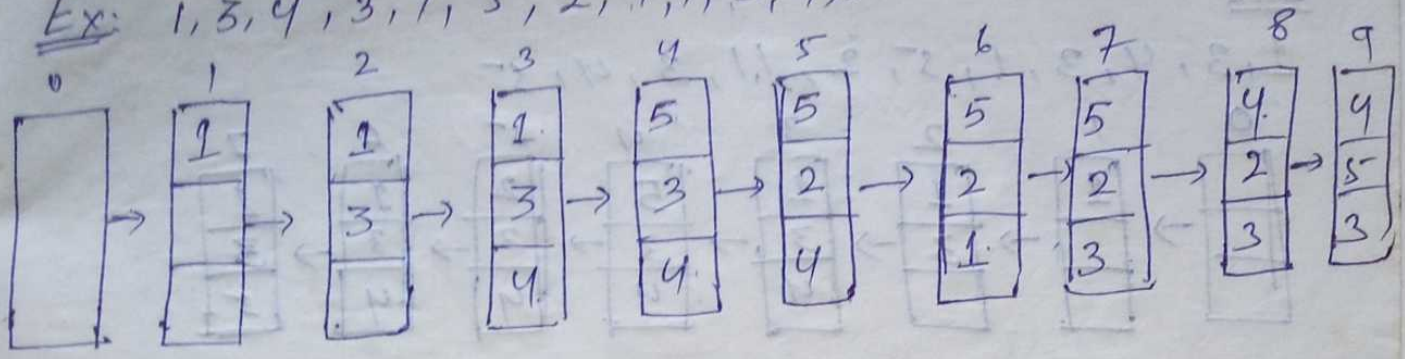
1, 3, 4, 3, 1, 5, 2, 1, 1, 3, 4, 5



Total miss = 7.

3. MFU:
* The block which was ~~used~~ reffered for maxm no. of time, that block will be replaced first.
* A counter can be used to know which block was reffered for maxm no. of time.

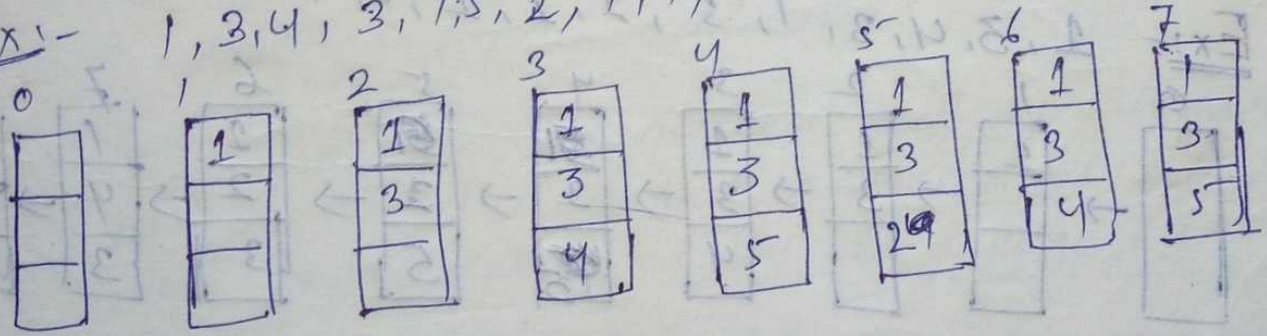
Ex: 1, 3, 4, 3, 1, 5, 2, 1, 1, 3, 4, 5



Total no of miss = 9

4. LFU : * The block that was reffered for less no. of times, that will be replaced first.
* A counter can be used to keep tracks of the block no, which was used for less no of time.

Ex:- 1, 3, 4, 3, 1, 5, 2, 1, 1, 3, 4, 5

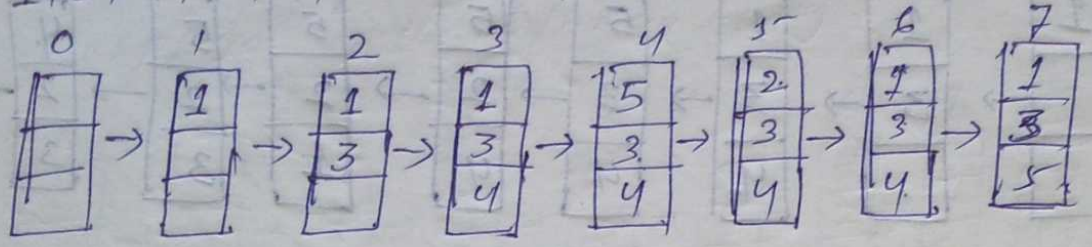


Total no of miss = 7

5. MRU : * Here the block which was reffered most recently, must be replaced first
* A timer can be used to know which block was reffered most recently.

Ex:-

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

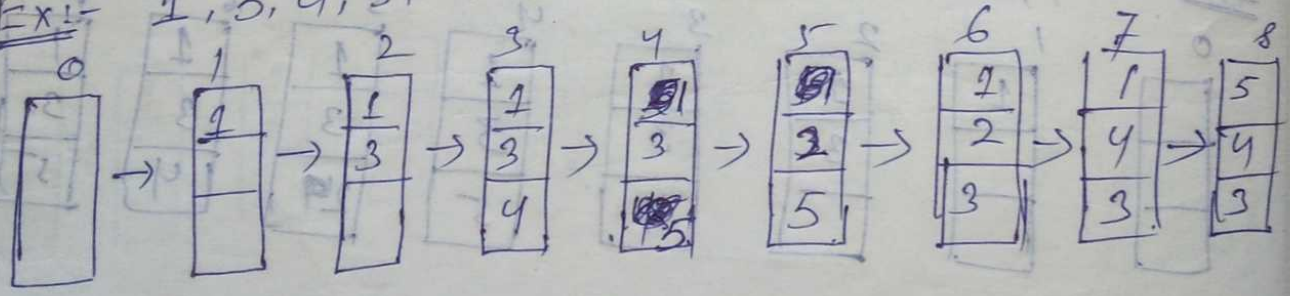


Total miss = 7

6. LRU: The block that was not reffered for the longest period of time, must be replaced first.

* A timer can be used to know the block which has not been reffered for longest period of time.

Ex:- 1, 3, 4, 3, 1, 5, 2, 1, 1, 3, 4, 5



Total no of miss = 8

Virtual Memories :- In a memory hierarchy system, Programs and data are first stored in auxiliary (secondary) memory. Portions of a Program or data are brought into main-memory as they are needed by the CPU.

Virtual memory is a concept that Provides an address space that is equal to the size of auxiliary memory.

For Example: A Processor that issues 32-bit addresses has an addressable space of 4G bytes (i.e 2^{32} bytes), but the size of main memory is of few hundred megabytes to 1G bytes.

So, it is possible to get an address space 4G by moving Programs and data between main-memory and secondary storage.

* A Virtual memory system Provides a mechanism for translating Program-generated addresses into correct main-memory locations.
* The binary addresses that the Processor issues for either instructions or data are called Virtual or logical addresses.

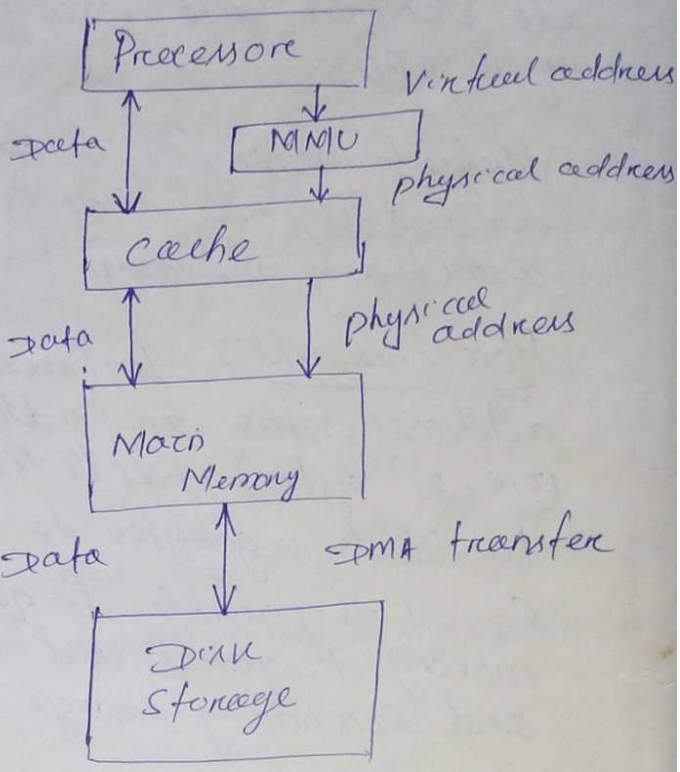
Address Space: An address used by a Programmer will be called a Virtual address and set of such addresses is called the Address space.

Memory space: An address in main-memory is called a location or physical address. The set of such locations (physical address) is called the memory space.

* A special hardware unit, called the memory Management unit (MMU) transfers Virtual addresses into physical addresses.

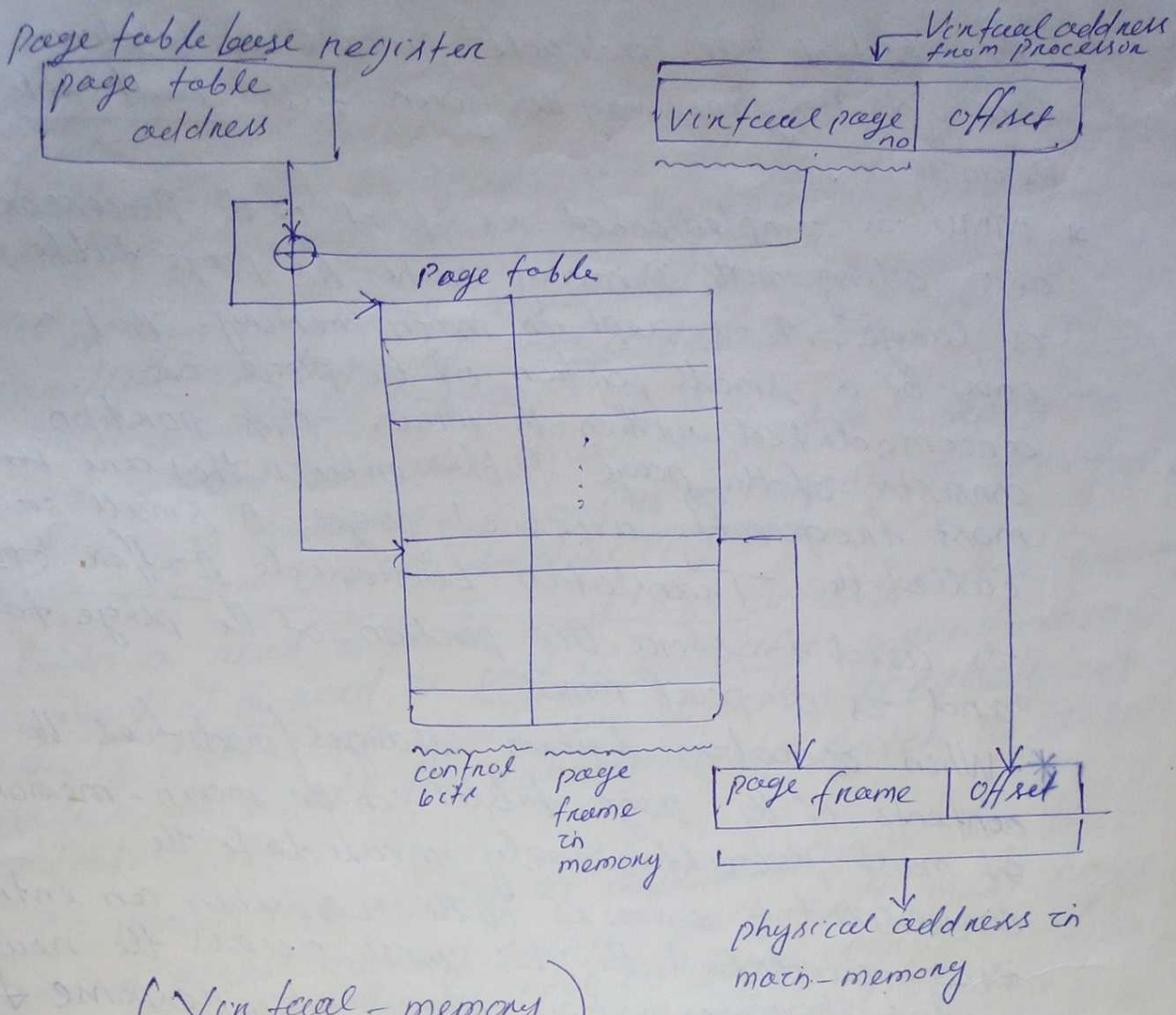
Address Translation :-

For translating Virtual addresses into physical addresses is to divide both the Virtual address space and physical address space (main-memory address space) into fixed-length units of equal size.



(Virtual memory Organization)

* In case of Virtual address this unit size are called pages, and in case of physical addresses (main-memory) this unit size are called blocks or page frame.
* pages or blocks are of size range from 2K to 16K bytes in length.
* Each Virtual address generated by the Processor is interpreted as Virtual Page number (high order bits) followed by an offset (low-order bits). The offset specifies the location of the word within a page.
* Information about the main-memory location of each page is kept in a page table.



(Virtual-memory Address Translation)

* page table contains information about main-memory address where the page is stored and the current states of the page (i.e, control bits)
* The starting address of the page table is kept in a page table base register.
* By adding the Virtual page no to the contents of page table base register, the address of the corresponding entry in the page table is obtained.
* The content of this location give the starting address of the page currently resides in main-memory.
* The control bits describes the status of the page while it is in the main-memory. one-bit indicates the validity of the page. Another bit indicates whether the page has been modified.

other control bits indicate various restrictions like, a Program may be given full read/write Permission.

* MMU is implemented as part of the Processor chip along with Primary cache. As page table is large, it is kept in main memory. But a copy of a small portion of the page is accomodated within the mmu. This portion consists of the page table entries that are most frequently accessed pages. A small cache called the Translation Lookaside Buffer (TLB) is used to store this portion of the page table and is a part mmu.
* When operating system changes/modifies the contents of the page tables in the main-memory, it must simultaneously invalidate the corresponding entries in the TLB, when an entry is invalidated, the TLB will access the new/updated information may be in response to access misses.

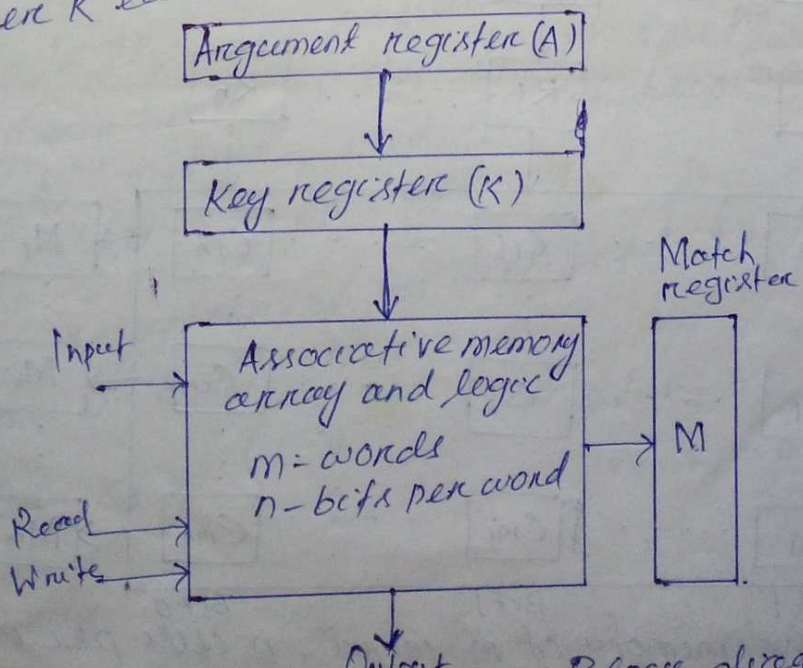
Associative memory

A memory unit accessed by content of the data itself rather than by an address is called an associative memory or content addressable memory.

* This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address location.
* when a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word.
* When a word is to be read from an associative memory, the content of the word, or part of the word is specified. The memory locates all words by searching parallelly and marks the words for reading which match the specified content.
* An associative memory is more expensive than a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external data.

Hardware Organization

It consists of a memory array and logic for m-words with n-bits per word. The argument register A and key register K each have n-bits, one for each bit of a word.



Block diagram of Associative memory.

* The match register M has m-bits, one for each memory word. Each word in memory is compared in parallel with the content of the argument register.

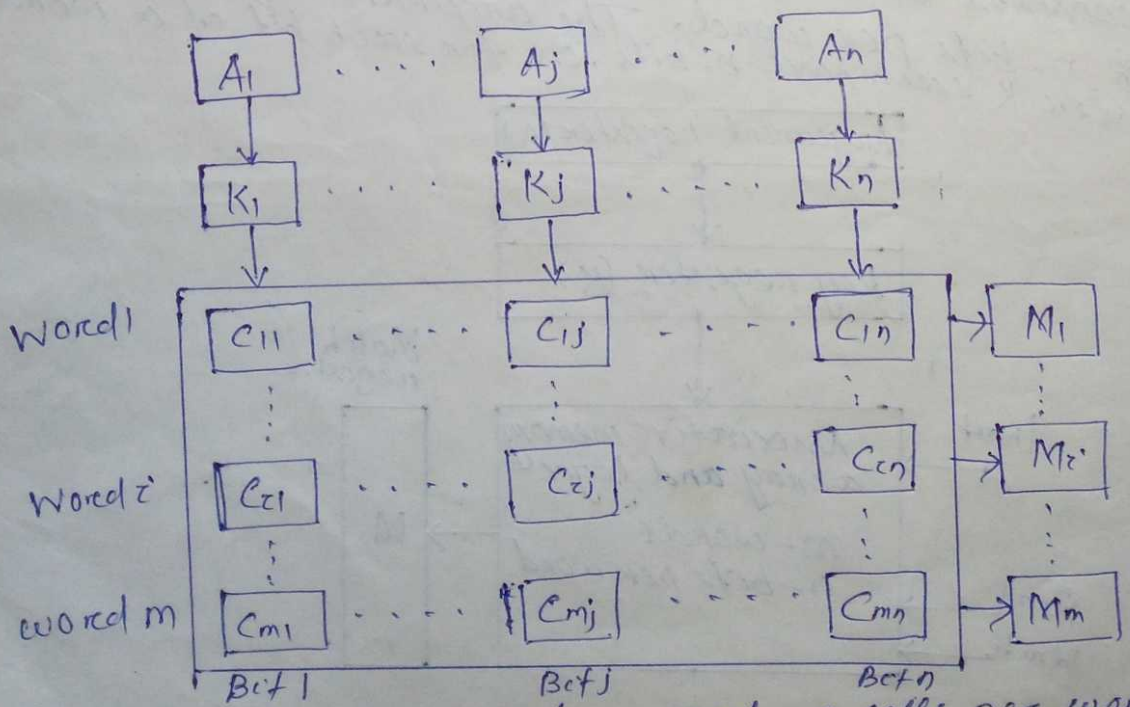
* The words that match the bits of the argument register set a corresponding bit in the match register.

* The key register provides a mask for choosing a particular field or key in the argument word. The entire argument is compared with each memory word if the key register contains all 1's. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.

Example

A	101	111100
K	111	000000
Word 1	100	111100 no match
Word 2	101	000001 match

Only the three left most bits of A are compared with memory words because K has 1's in these positions.



Associative memory of m word, n cells per word.

* The cells in the array are marked by the letter 'c' with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. Thus cell C_{ij} is the cell for bit j in word i. A bit A_j in the Argument register is compared with all the bits in column j of the array Provided that $K_j = 1$, for $j = 1, 2 \cdots n$. If a match occures with the word i, the corresponding bit M_i in the match register is set to 1, If doesn't match, M_i is cleared to ~~0~~ 0.

Match Logic: (Neglecting the key bits)

* Word i is equal to the argument in A if $A_j = C_{ij}$ for $j = 1, 2 \cdots n$. Two bits are equal if both A_j and C_{ij} are equal i.e 1 or both 0.

* The equality of two bits (A_j and C_{ij}) can be expressed logically by $x_j = A_j C_{ij} + A_j' C_{ij}$ [match if $x_j = 1$, mismatch if $x_j = 0$]

* Logic for setting the corresponding match bit M_i is $M_i = x_1 \cdot x_2 \cdots x_n$ i.e AND operation between all x_j's.

Match Logic (Including key bits)

* Let's now include the key bit K_j in the comparision (match) logic. The requirement is ~~achieved by ORing each term with~~ is that if $K_j = 0$, the corresponding bits of A_j and C_{ij} need no comparision. Only when $K_j = 1$ must they be compared.

Thus requirement is achieved by ORing each term with K_j' thus

$$x_j + K_j' = \begin{cases} x_j & \text{if } K_j = 1 \\ 1 & \text{if } K_j = 0 \end{cases}$$

bit serial bit parallel

Similarly for

$M_i = (x_1 + K_1')(x_2 + K_2')(x_3 + K_3') \cdots (x_n + K_n')$

The Performance of cache memory

The performance of cache is measured in terms of a quantity called hit ratio.

* When the CPU refers to memory and finds the word in cache, it is said to be a hit. If the word is not found in cache, it is in main memory and it counts as a miss.
* The ratio of the number of hits ~~divided by~~ the total CPU references to memory (hits plus misses) is the hit ratio.
* Due to cache miss, extra time needed to bring the desired information into the cache is called the miss penalty.

* Let h be the hit rate, M the miss Penalty, that is, the time to access information in the main memory & C the time to access information in the cache.

The average access time experienced by the processo is

$$t_{ave} = hC + (1-h)M$$

* In high-Performance Processors two levels of cache are normally used. The L1 cache is on the Processo chip. The L_2 cache, which is much larger, may be implemented on the Processor chip

* The average access time experienced by the Processor in a system with two levels of caches is

$$t_{ave} = h_1 c_1 + (1-h_1) h_2 c_2 + (1-h_1)(1-h_2) M$$

where

h_1 = is the hit ~~ratio~~ rate in the L1 cache

h_2 = is the hit rate " " L2 cache

c_1 = is the time to access information in the L1 cache.

c_2 = is the time to access information in the L2 cache

M = is the time to access information in the main memory.

Ex