Chapter_-4

Memory Organization

i) MEMORY HIERARCHY

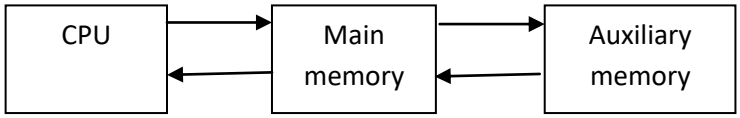
The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy. The memory unit is an essential component in any digital computer since it is needed for storing programs and data. At present the following three kinds of memory are commonly used in modern computers:

a) Semiconductor memory
b) Magnetic memory
c) Optical memory

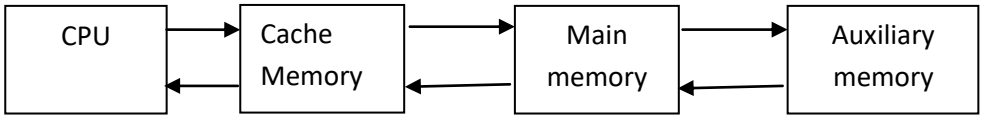
The memory unit that communicates directly with the CPU is called the Main Memory (or Primary memory).
Devices that provide backup storage are called auxiliary Memory (or Secondary).
Only programs and data currently needed by the processor reside in Main memory. All other information is stored in Auxiliary memory and transferred to main memory when needed.

The Memory hierarchy system consists of all storage devices employed in a computer system from the slow but high capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory accessible to the high speed processing logic.

a)



b)



Memory Hierarchies a) Without cache memory b) with cache 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.

The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic.
The cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations.
By making programs and data available at a rapid rate, it is possible to increase the performance rate of the computer. While the I/O processor manages data transfers between auxiliary memory and main memory, the cache organization is concerned with the transfer of information between main memory and CPU. Thus each is involved with a different level in the memory hierarchy is economics.

ii) MAIN MEMORY

It is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation. The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.

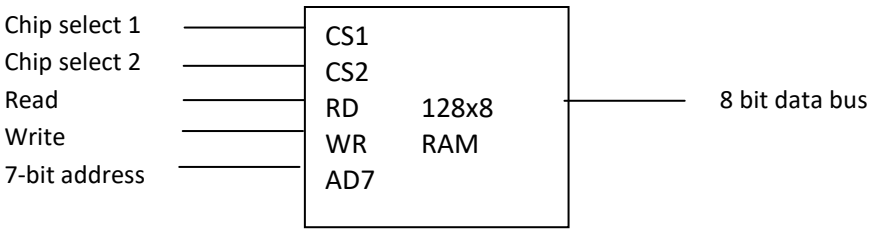
The static RAM (SRAM) consists essentially of internal flip flops that store the binary information. The stored information remains valid as long as power is applied to the unit.

The dynamic RAM (DRAM) stores the binary information in the form of electric charges that are applied to capacitors. The 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 the dynamic memory. Refreshing is done by cycling through the words every few milliseconds to restore the 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. 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 are subject to change. ROM is used for storing programs that are permanently resident in the computer and for tables of constants that do not change in value once the production of the computer is completed.

The ROM portion of main memory is needed for storing an initial program called a bootstrap loader. The bootstrap loader is a program whose function is to start the computer software operating when power is turned on. The contents of ROM remain unchanged after power is turned off and on again. The start-up of a computer consists of turning the power on and starting the execution of an initial 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 are available in a variety of sizes. 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.

To demonstrate the chip interconnection, we will show an example of a 1024 x 8 memory constructed with 128 x 8 RAM chips and 512 x 8 ROM chips.

RAM and ROM chips: A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed. Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation, or from CPU to memory during a write operation.



A bidirectional bus can be constructed with three state buffers. A three state buffer output can be placed in one of these possible states: a signal equivalent to logic 1, a signal equivalent to logic 0, or a high impedance state. The logic 1 and 0 are normal digital signals. The high impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance.

The capacity of the memory is 128 words of 8 bits per word. This requires a 7 bit address and an 8 bit bidirectional data bus. The read and write inputs specify the memory operation and the two chips select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor. The availability of more than one control input to select the chip facilitates the decoding of the address lines when multiple chips are used in the microcomputer. When the chip is selected, the two binary states in this line specify the two operations of read or write.

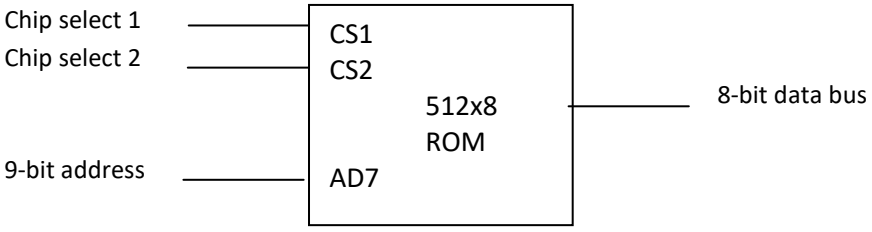
CS1	CS2	RD	WR	MEMORY FUNCTION	STATE OF DATA BUS
0	0	X	X	Inhibit	high impedance
0	1	X	X	Inhibit	high impedance
1	0	x	1	write	input data to RAM
1	0	1	X	Read	Output data from RAM
1	1	x	x	Inhibit	high impedance

The unit is in operation only when CS1=1 and CS2=0. The bar on top of the 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 it data bus is in a high impedance state. When CS1=1 and CS2=0, the memory can be placed in a write or read mode.

When the WR input is enabled, the memory stores a byte from the data bus into a location specified by the address input lines. When the RD input is enabled, the content of the selected byte is placed into the data bus. The RD and WR signals control the memory operation as well as the bus buffers associated with the bidirectional data bus.

A ROM chip is organized externally in a similar manner, however, since a ROM can only read, the data bus can only be in an output mode.



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 the RAM. For this reason, the diagram specifies a 512 byte ROM, while the RAM has only 128 bytes. The nine address lines in the ROM chip specify any one of the 512 bytes stored in it.

The two chip select inputs must be CS1=1 and CS2=0 for the unit to operate. Otherwise, the data bus is in a high impedance state. There is no need for a read or write control because the unit can only read. Thus when the chip is enabled by the two select inputs, the byte selected by the address lines appears on the data bus.

Memory address map:

The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM. The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available.

The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip. The table, called a memory address map, is a pictorial representation of assigned address space for each chip in the system.

To demonstrate with a particular example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.

Memory address map for Micro computer

Component	Hexadecimal Address										
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000-007F	0	0	0	X	X	X	X	X	X	X
RAM 2	0080-00FF	0	0	1	X	X	X	X	X	X	X
RAM 3	0100-017F	0	1	0	X	X	X	X	X	X	X
RAM 4	0180-01FF	0	1	1	X	X	X	X	X	X	X
ROM	0200-03FF	1	X	X	X	X	X	X	X	X	X

The component column specifies whether a RAM or a ROM chip is used. The hexadecimal address column assigns a range of hexadecimal equivalent address for each chip.

The address bus lines are listed in the third column. Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero. The small x's under the address lines designate those lines that must be connected to the address inputs in each chip.

The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines. The x's are always assigned to the low order bus lines: lines 1 through 7 for the RAM and lines 1 through 9 for the ROM. It is now necessary to distinguish between four RAM chips by assigning to each a different address.

For this particular example we choose bus lines 8 and 9 to represent four distinct binary combinations. The table clearly shows that the 9 low order bus lines constitute a memory space for RAM equal to 2^9 = 512 bytes. The distinction between a RAM and ROM address is done with another bus line. Here we choose line 10 for this purpose.
When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it select the ROM. The equivalent hexadecimal address for each chip is obtained from the information under the address bus assignment. The address bus lines are subdivided into groups of 4 bits each so that each group can be represented with a hexadecimal digit.

RAM and ROM chips are connected to a CPU through the data and address buses. The low order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs.
Thus, when address lines 8 and 9 are equal to 00, the firs RAM chip is selected. When 01, the second RAM chip is selected, and so on. The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip.

iii) AUXILIARY MEMORY

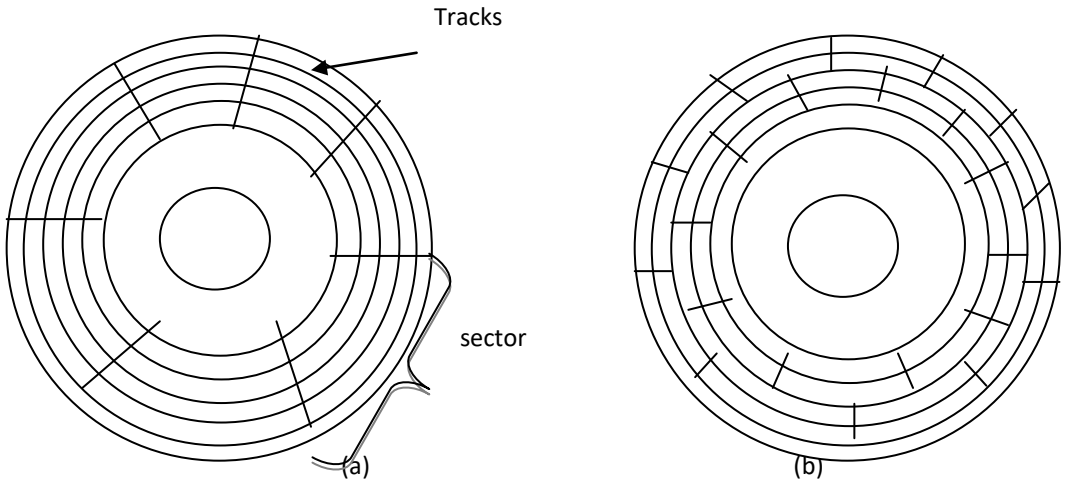
The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. The important characteristics of any device are its access mode, access time, transfer rate, capacity, and cot. The average time required to reach a storage location in memory and obtain its contents is called the access time. In electromechanical devices with moving parts such as disks and tapes, the access time consists of a seek time required to position the read-write head to a location and a transfer time required to transfer data to or from the device.

Auxiliary storage is organized in records or blocks. A record is a specified number of characters or words. Reading or writing is always done on entire records. Magnetic drums and disks are consisting of high speed rotating surfaces coated with a magnetic recording medium. The recording surface rates at uniform speed. Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a write head.

Stored bits are detected by a change in magnetic filed produced by a recorded spot on the surface as it passes through a read head. The amount of surface available for recording in a disk is greater than in a drum of equal physical size. Therefore, more information can be stored on a disk than on a drum of comparable size. For this reason, disks have replaced drums in more recent computers.

Magnetic disks:

Magnetic disk include hard disks and floppy disks. Working principle is same for both hard disks and floppy disks. A magnetic disk is a surface device. It stores data on its surface. Its surface is divided into circular concentric tracks, and each track is divided into sectors.



Tracks and sectors

a) All tracks have same number of sectors
b) Outer tracks have more sectors than inner tracks.

The number of bytes stored in each sector is kept same. All tracks store the same amount of data. This results higher bit density in inner tracks than that of the outer tracks. Since the same number of bytes is stored in each sector, the size of the inner sectors decides the storage capacity for all other sectors on the disk.
Due to this reason some storage space remains unused on the outer tracks. To utilize this unused space, the disk surface is divided into zones. In a zone the number of sectors in each track is same, but it is different than the number of sectors in a track of the other zone. In outer zones the number of sectors per track is more than that in inner zones. Thus storage capacity of the disk is increased by this technique. A track on a disk is selected in random fashion, but data is written to or read from a sector in serial fashion.
Disk controller: magnetic disk drives require controller. The controller converts instructions received from software to electrical signals to operate disks. The functions of a disk controllers are:

i) To interface a disk drive system to the CPU.
ii) Disk drive selection, because a computer uses more than one disk drive
iii) Track and sector selection
iv) To issue commands to the disk drive system to perform read/write operation
v) Data separation
vi) Serial to parallel and parallel to serial conversion
vii) Error detection, etc.

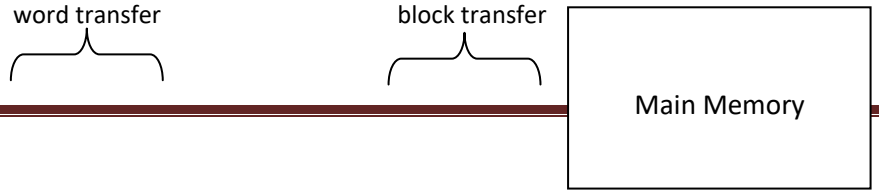
Hard disk controller and floppy disk controller are available in IC form.

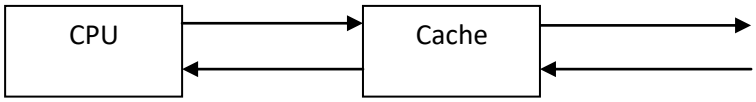
Hard disk: Hard disks are on line storage devices. i.e, permanently connected to the computer system. When computer is on, the device is available to store information or to give information. They store programs data, operating system, compilers, assemblers, application programs, database etc.,
A hard disk id made of aluminium (or other metal or metal alloy) with a thin coating of magnetic material (iron oxide)over it. Standard size of hard disk is 3.5 inch. Hard disks and read/write heads are kept in a sealed air filtered enclosure. This technique is known as Winchester technique. The disks are addressed by drive number, cylinder number (ie., track number), surface number and sector number.
Floppy disk: floppy disks are made of Mylar(a plastic material) coated with magnetic material (iron oxide or barium ferrite). The disk is not a hard plate, rather it is very thin piece of flexible plastic. They are removable disks. It is an inexpensive storage device and it is used as backup memory. The size of floppy disks is 3.5 inch diameter. A floppy disk rotates at 360 rpm. Its average access time is 150-250 ms.
Magnetic tapes: are used for backup memory. They are sequential access device whereas a disk drive is a direct is a direct access device. In disk drive system the head moves to the position of the desired record. But in the case of tap drive system, the head moves sequentially. It has to move through the adjacent records until it reaches the desired record. However, recently Exabyte company has developed a technology using which the heads can read data from the physical location on the tape, without having to follow tracks from beginning to end. The magnetic tape is made of flexible polyester coated with magnetisable material. Tape width varies from 3 mm to 12.7 mm. Most of the tapes are available in cartridge form. The capacity varies from a few hundred mb to a few hundred GBs.
Optical disks: optical disks are used for backup memory. Information is written to or read from an optical disk using laser beam. It has very high storing capacity as compared to magnetic floppy disks. It has very long life. The capacity of optical disks varies from 650 MB to 17 GB. DVDs of 15,25,30 and 50 GB capacity etc., an optical disk is a direct access device. As its read /write head does not touch the disk surface, there is no disk wear and problem of head crash. Elaborate error checking codes can be used as there is no problem of space because of its high storage capacity. The greatest drawback of an optical disk drive system is its large access time as compared to magnetic hard disk drive. An optical disk system the drive has to move on a sizable optical assembly across the disk surface. This results in an increased access time.
Types of optical disks: CD(compact disk), CD-R(recordable), CD-RW(read/write), DVD(digital versatile disk), DVD-R, DVD-RW..

The diameter of CD disk are 12 cms. DVDs come in 8 cm diameter as well as 12 cm diameter.

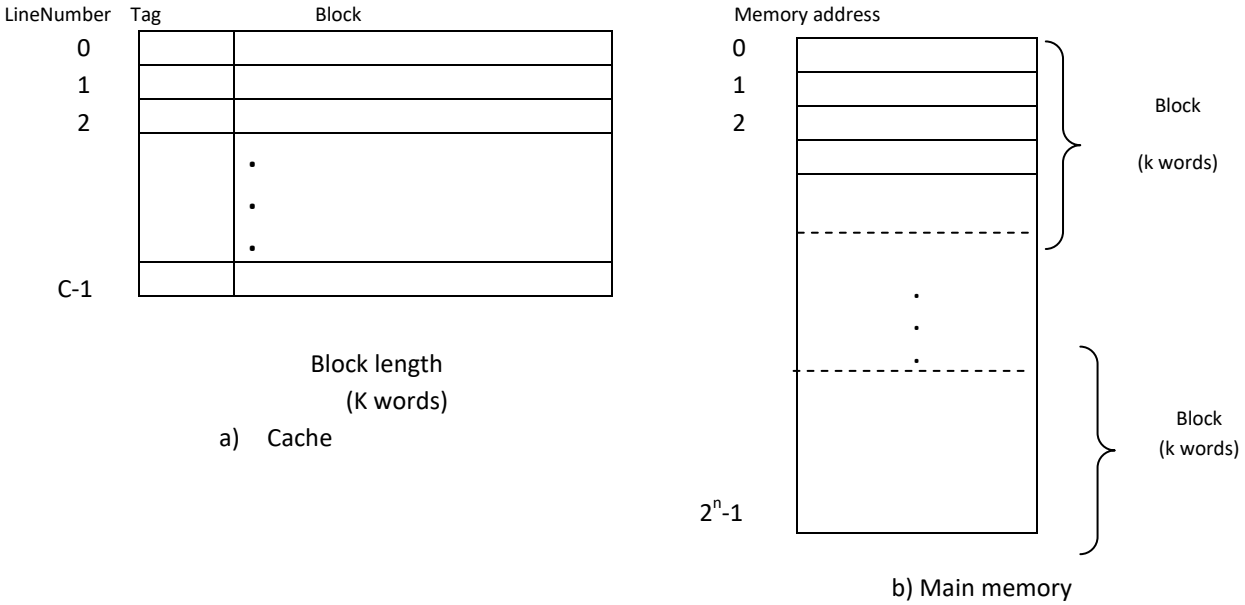
iv) CACHE MEMORY

Cache memory is intended to give memory speed approaching that of the fastest memories available, and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. The cache contains a copy of portions of main memory. When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.



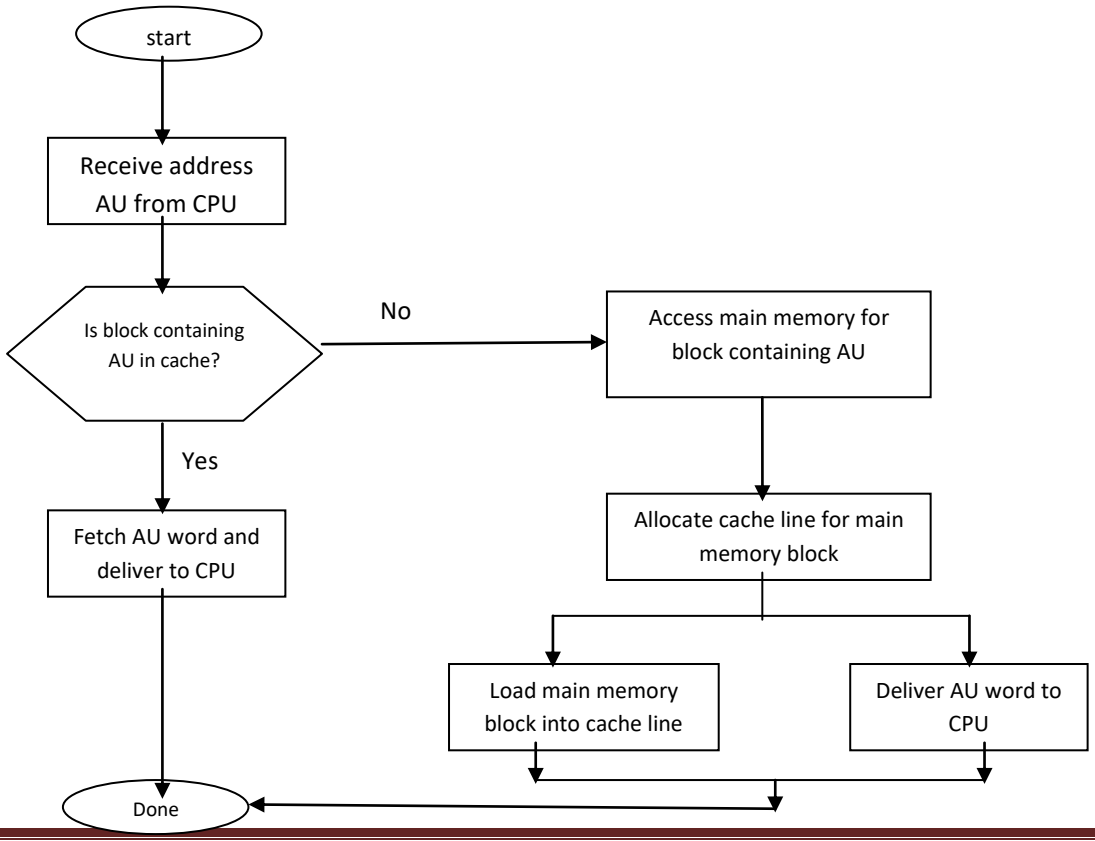


Main memory consists of up to 2^n addressable words, with each word having a unique n-bit address. The mapping purposes, this memory is considered to consist of a number of fixed length blocks of K words each. That is, there are $M=2^n/K$ blocks.



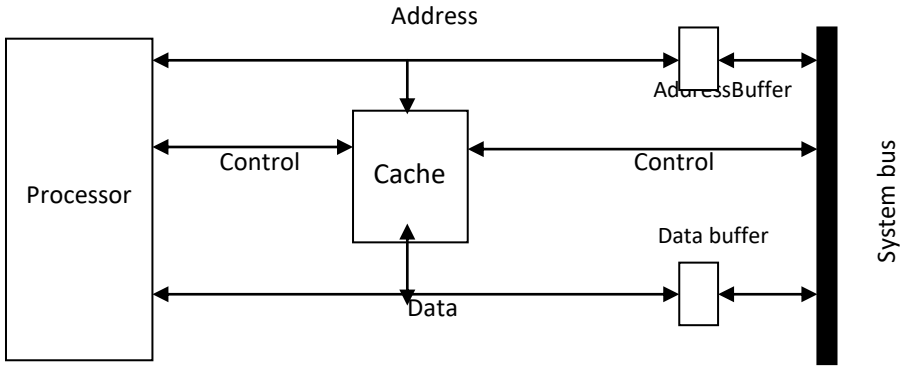
---Cache/Main memory structure---

The cache consists of C lines. Each line contains K words, plus a tag of a few bits; the number of words in the line is referred to as the line size. The number of lines is considerably less than the number of main memory blocks (C << M). At any time, some subset of the blocks of memory resides in lines in the cache. If a word in a block of memory is read, that block is transferred to one of 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. The tag is usually a portion of the main memory address.



The processor generates the address, AU, of a word to be read. If the word is considered in the cache, it is delivered to the processor. Otherwise, the block containing that word is loaded into the cache, and the word is delivered to the processor.

In typical of contemporary cache organization, the cache connects to the processor via data, control, and address lines. The data and address lines also attach to data and address buffers, which attach to a system bus from which main memory is reached.



--Typical cache organization--

When a cache hit occurs, the data and address buffers are disabled and communication is only between processor and cache, with no system bus traffic. When a cache miss occurs, the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.

Mapping Function:

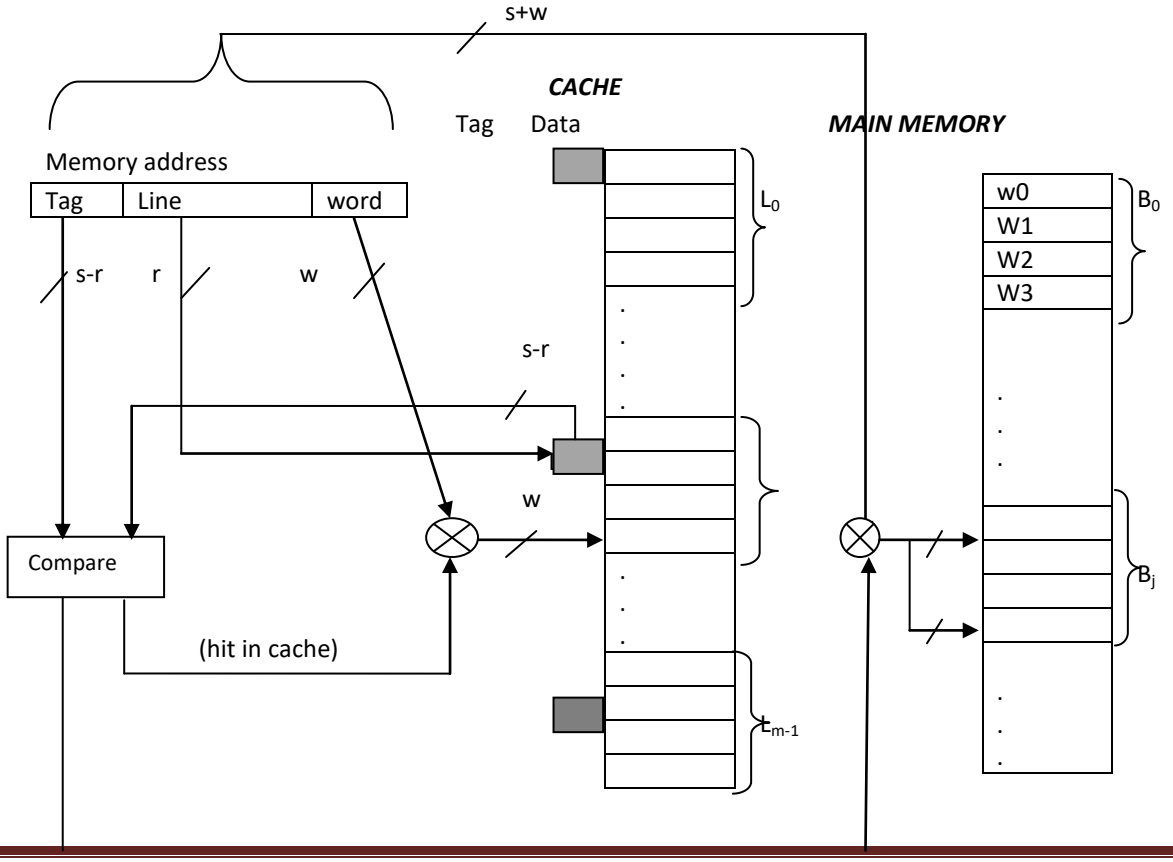
There are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. Further, a means is needed for determining which main memory block currently occupies a cache line. The choice of the mapping function dictates how the cache is organized. Three techniques can be used: direct, associative, and set associative.

Direct Mapping: the simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. The mapping is expressed as

$$i = j \text{ modulo } m$$

where

i= cache line number , j=main memory block number, m=number of lines in the cache



(miss in cache)

--Direct mapping cache organization--

The mapping function is easily implemented using the address. For purposes of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory; in most contemporary machine, the address is at the byte level. The remaining s bits specify one of the 2^s blocks of main memory. The cache logic interprets these s bits as a tag of s - r bits (most significant portion) and a line field of r bits. This latter field identifies one of the $m=2^r$ lines of the cache.

Address length = (s+w) bits

Number of addressable units = 2^{s+w} words or bytes

Block size = line size = 2^w words or bytes

Number of block in main memory = $2^{s+w} / 2^w = 2^s$

Number of lines in cache = $m= 2^r$

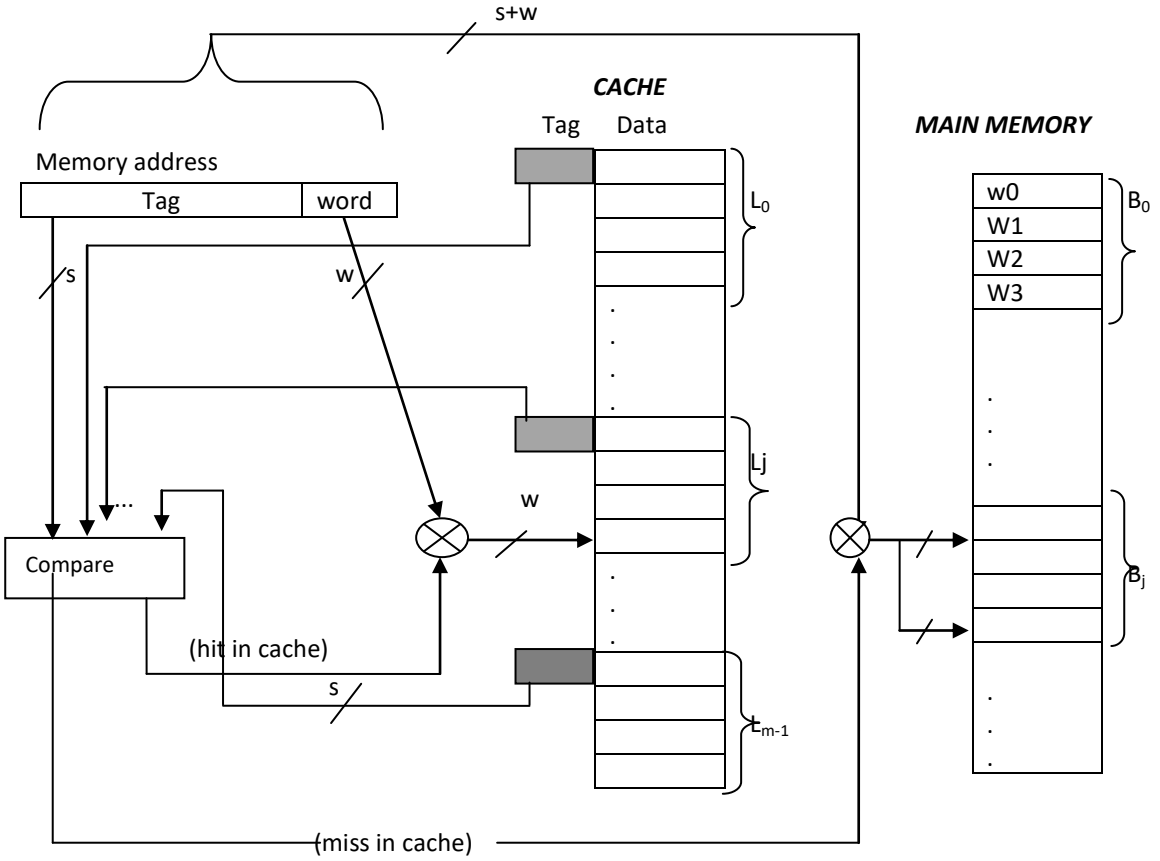
Size of tag= (s-r) bits

The use of a portion of the address as a line number provides a unique mapping of each block of main memory into the cache. When a block is actually read into its assigned line, it is necessary to tag the data to distinguish it from other blocks that can fit into that line. the most significant s – r bits serve this purpose.

The direct mapping technique is simple and inexpensive to implement. Its main disadvantage is that there is a fixed cache location for any given block. Thus, if a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low.

<u>Associative Mapping :</u>

It overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache. In this case, the cache control logic interprets a memory address simply as a tag and a word field. The tag field uniquely identifies a block of main memory. To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's tag for a match.



--Associative mapping cache organization--

Note that no field in the address corresponds to line number, so that the number of lines in the cache is not determined by the address format.

Address length = (s+w) bits
Number of addressable units = 2^{s+w} words or bytes
Block size = line size = 2^{w} words or bytes
Number of blocks in main memory = $2^{s+w} / 2^{w} = 2^{s}$
Number of lines in cache = undetermined
Size of tag=s bits

With associative mapping, there is flexibility as to which block to replace when a new block is read into the cache. Replacement algorithms are designed to maximize the hit ratio. The principle disadvantage of associative mapping is the complex circuitry required to examine the tags of all cache lines in parallel.

<u>Set associative mapping :</u>

It is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages. In this case, the cache is divided into v sets, each of which consists of k lines. The relationships are

M = v x k

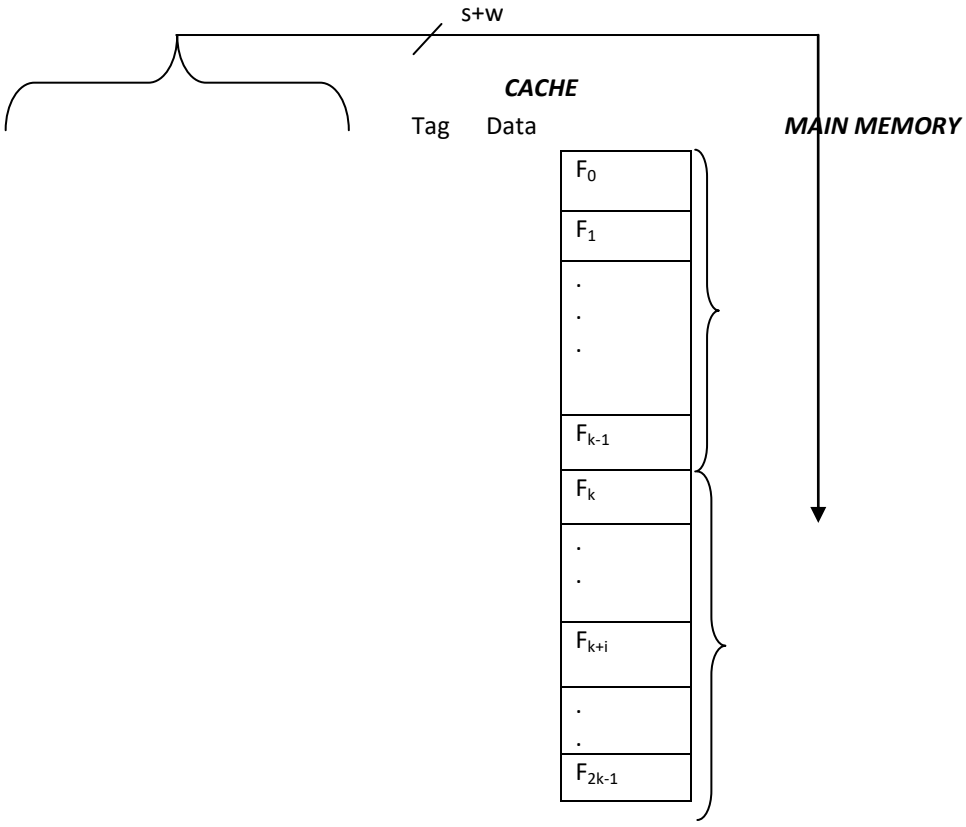
i = j modulo v

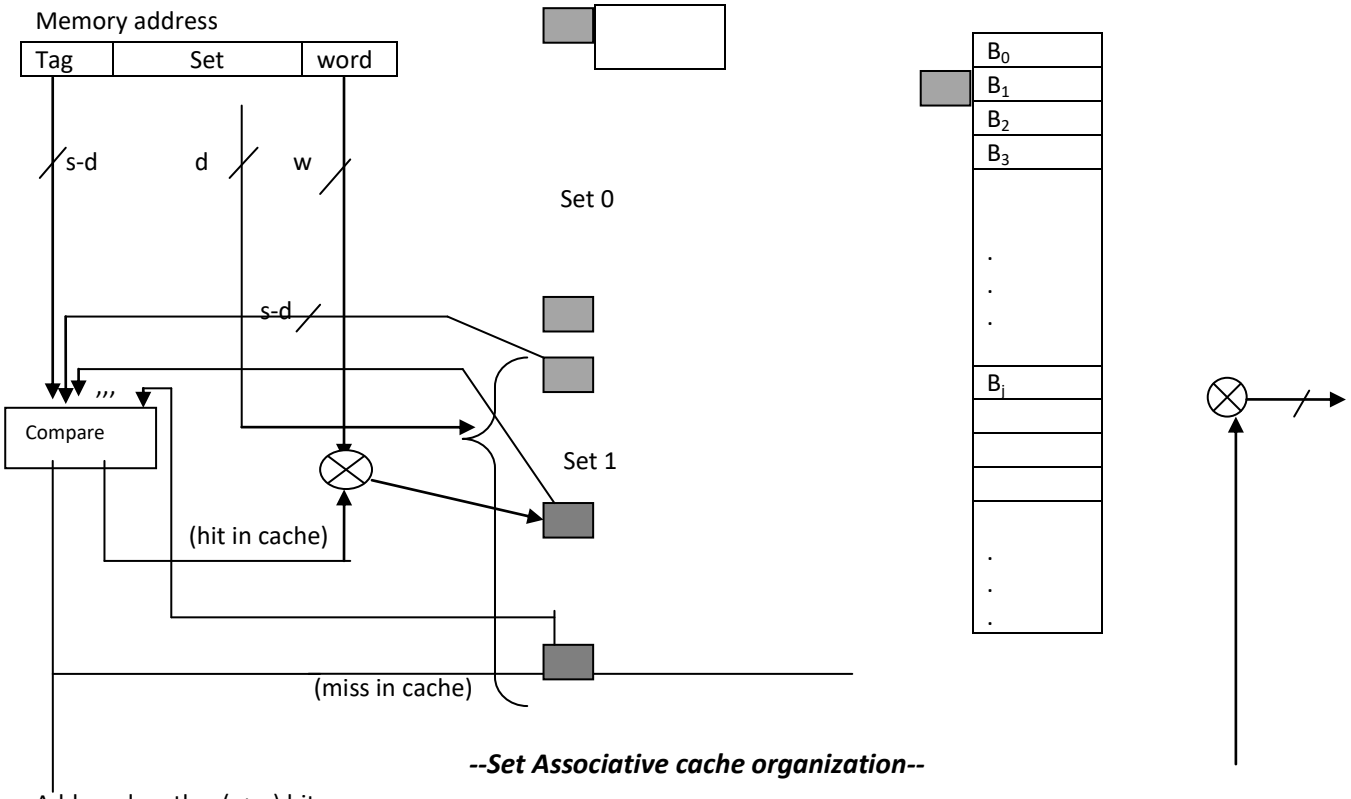
where

i = cache set number, j = main memory block number, m=number of lines in the cache

The is referred to as k-way set associative mapping. With set associative mapping, block B_j can be mapped into any of the lines of set i. In this case, the cache control logic interprets a memory address simply as three fields: tag, set and word. The d set bits specify one of v = 2^{d} sets. The s bits of the tag and set fields specify one of the 2^{s} blocks of main memory.

With fully associative mapping the tag in a memory address is quite large and must be compared to the tag of every line in the cache. With k-way set associative mapping, the tag in a memory address is much smaller and is only compared to the k tags within a single set.





--Set Associative cache organization--

Address length = (s+w) bits
Number of addressable units = 2^{s+w} words or bytes
Block size = line size = 2^w words or bytes
Number of blocks in main memory $2^{s+w} / 2^w = 2s$
Number of lines in set = k
Number of sets = v = 2^d
Number of lines in cache = kv = k x 2^d
Size of tag = (s – d) bits

v) VIRTUAL MEMORY

In a memory hierarchy system, programs and data are first stored in auxiliary memory. Portions of a program or data are brought into main memory as they are needed by the CPU. Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory.

Each address that is referenced by the CPU goes through an address mapping from the virtual address to a physical address in main memory.

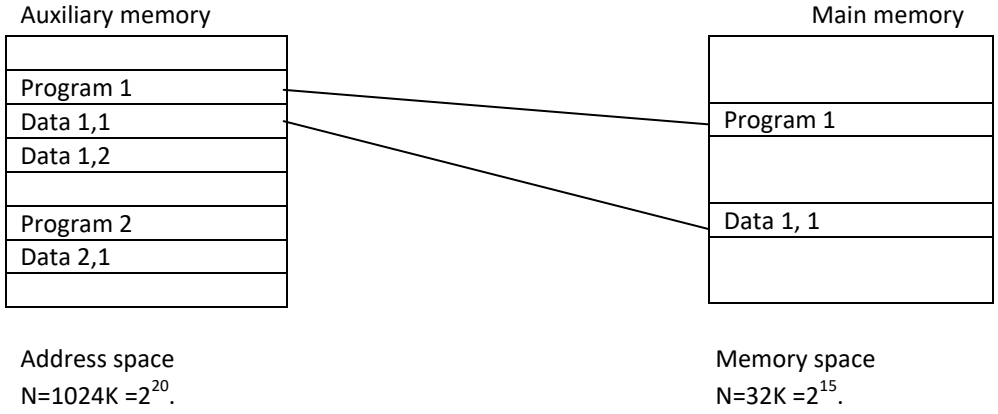
Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

A virtual memory system provides a mechanism for translating program generated addresses into correct main memory locations. This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table.

Address space and Memory space: An address used by a programmer will be called a virtual address, and the set of such addresses is called the address space An address in main memory is called a location or physical address. The set of such locations is called the memory space. The most computers the address and memory spaces are identical. The address space is allowed to be larger than the memory space in computers with virtual memory.

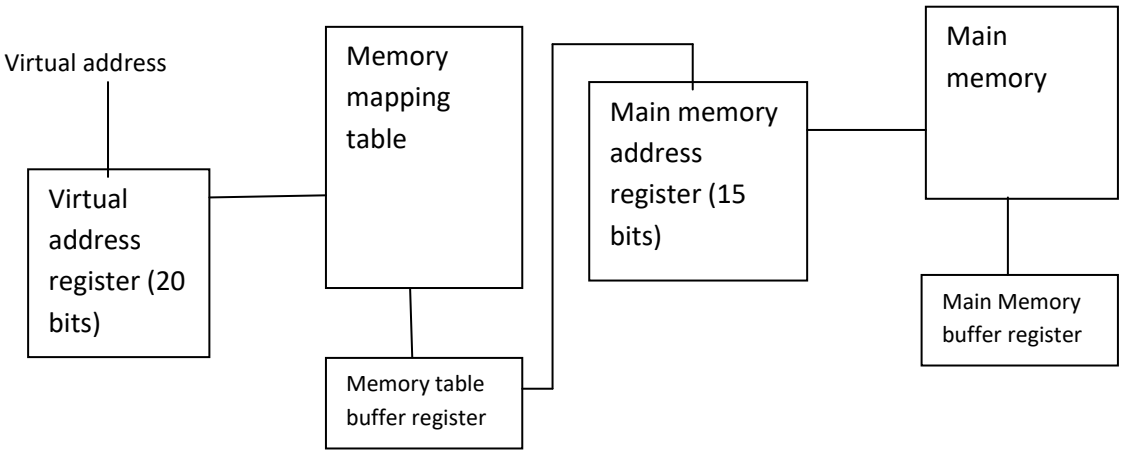
As an illustration, consider a computer with a main memory capacity of 32K words (K-1024). Fifteen bits are needed to specify a physical address in memory since 32K=2^{15}. Suppose that the computer has available auxiliary memory for storing 2^{20}=1024K words. Thus auxiliary memory has a capacity for storing information equivalent to the capacity of 32 main memories. Denoting the address space by N and the memory space by M, we then have for this example N = 1024K and M=32K. In a multiprogramming computer system, programs and data are transferred to and from auxiliary memory and main memory based on demands imposed by the CPU.

Suppose that program 1 is currently being executed in the CPU. Program 1 and a portion of its associated data are moved from auxiliary memory into main memory. Portions of programs and data need not be in contiguous locations in memory since information is being moved in and out, and empty spaces may be available in scattered locations in memory. In a virtual memory system, programmers are told that they have the total address space at their disposal. Moreover, the address field of the instruction code has a sufficient number of bits to specify all virtual addresses.



In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits. Thus CPU will reference instructions and data with a 20 bit address, but the information at this address must be taken from physical memory because access to auxiliary storage for individual words will be prohibitively long.

A table is then needed to map a virtual address of 20 bits to a physical address of 15 bits. The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU. The mapping table may be stored in a separate memory or in main memory.



Address mapping using pages: The table implementation of the address mapping is simplified if the information in the address space and the memory space are each divided into groups of fixed size.

For example, if a page or block consists of 1K words, then, using the previous example, address space is divided into 1024 pages and main memory is divided into 32 blocks. Although both a page and a block are split into groups of 1K words, a page refers to the organization of address space while a block refers to the organization of memory space. The programs are also considered to be split into pages.

Page 0
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7

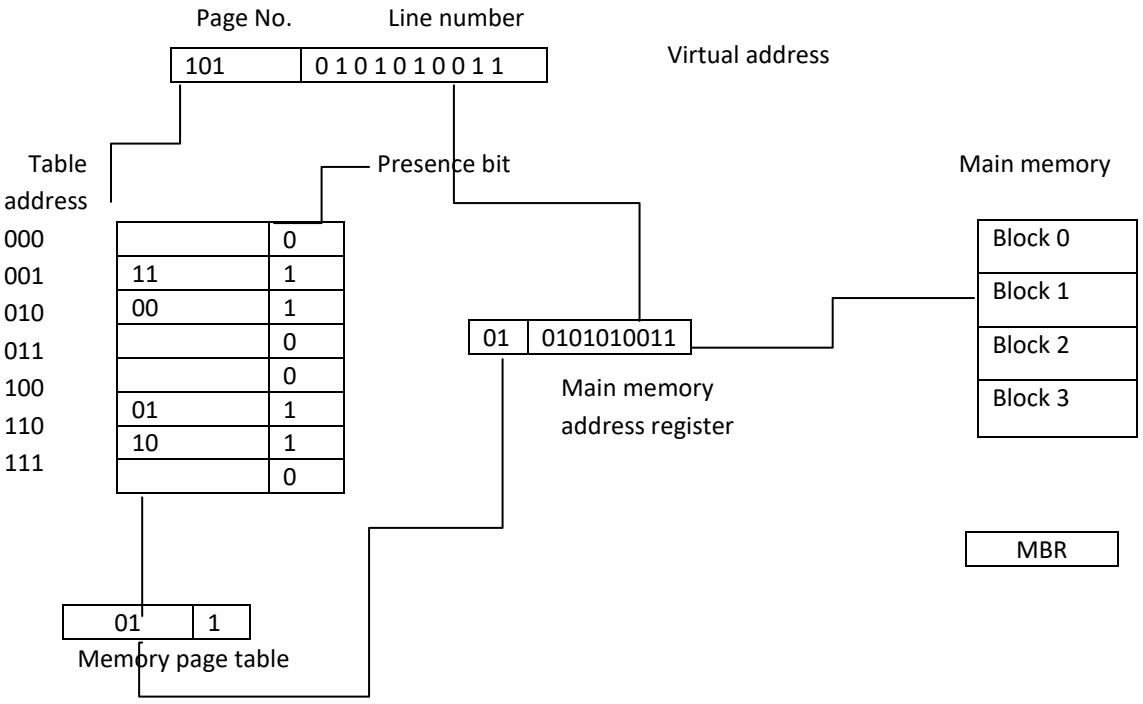
Address space
$N=8K=2^{15}$.

Block 0
Block 1
Block 2
Block 3

Memory space
$N=4K=2^{12}$.

Portions of programs are moved from auxiliary memory to main memory in records equal to the size of a page. The term "page frame" is sometimes used to denote a block.
Consider a computer with an address space of 8K and a memory space of 4K. If we split each into groups of 1K words we obtain eight pages and 4 blocks. At any given time, upto four pages of address space may reside in main memory in any one of the four blocks. The mapping from address space to memory space is facilitated if each virtual address is considered to be represented by two numbers: a page number address and a line within the page.

In the example, a virtual address has 13 bits. Since each page consists of 2^{10}=1024 words, the high order three bits of a virtual address will specify one of the eight pages and the low order 10 bits give the line address within the page.
Note that the line address in address space and memory space is the same; the only mapping required is from a page number to a block number.



The memory page table consists of eight words, one for each page. The address in the page table denotes the page number and the content of the word gives the block number where that page is stored in main memory.

The table shows that pages 1,2,5,6 are now available in main memory in blocks 3,0,1,2 respectively. A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory. A 0 in the presence bit indicates that this page is not available in main memory. The CPU references a word in memory with a virtual address of 13 bits. The 3 high order bits of the virtual address specify a page number and also an address for the memory page table.

The content of the word in the memory page table at the page number address is read out into the memory table buffer register. If the presence bit is a 1, the block number thus read is transferred to the two high order bits of the main memory address register. The line number from the virtual address is transferred into the 10 low order bits of the memory address register. A read signal to main memory transfers the content of the word to the main memory buffer register ready to be used by the CPU.

If the presence bit in the word read from the page table is 0, it signifies that the content of the word referenced by the virtual address does not reside in main memory. A call to the operating system is then generated to fetch the required page from auxiliary memory and place it into main memory before resuming computation.

Associative Memory Page Table: A more efficient way to organize the page table would be to construct it with a number of words equal to the number of blocks in main memory. In this way the size of the memory is reduced and each location is fully utilized. This method can be implemented by means of an associative memory with each word in memory containing a page number together with its corresponding block number. The page field in each word is compared with the page number in the virtual address. If a match occurs, the word is read from memory and it corresponding block number is extracted.

Replacement policies: A virtual memory system is a combination of hardware and software technique. The memory management software system handles all the software operations for the efficient utilization of memory space. It must decide (1) which page in main memory ought to be removed to make room for a new page, (2) when a new page is to be transferred from auxiliary memory to main memory, and (3) where the page is to be placed in main memory. The hardware mapping mechanism and the memory management software together constitute the architecture of a virtual memory.

When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called page fault. When page fault occurs, the execution of the present program is suspended until the required, page is brought into main memory. Since loading a page from auxiliary memory to main memory is basically an I/O operation, the operating system assigns this task to the I/O processor. In the meantime, control is transferred to the next program in memory that is waiting to be processed in the CPU. Later, when the memory block has been assigned and the transfer completed, the original program can resume its operation.

When a page fault occurs in a virtual memory system, it signifies that the page referenced by the CPU is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, it would be necessary to remove a page from a memory block to make room for the new page. The policy for choosing pages to remove is determined from the replacement algorithm that is used. Two of the most common replacement algorithms used are the first-in, first-out (FIFO) and the least recently used (LRU). The FIFO algorithm selects for replacement the page that has been in memory the longest time. Each time a page is loaded into memory, its identification number is pushed into a FIFO stack.

FIFO will be full whenever memory has no more empty blocks. When a new page must be loaded, the page least recently brought in is removed. The page to be removed is easily determined because its identification number is at the top of the FIFO stack. The FIFO replacement policy has the advantage of being easy to implement. It has the disadvantage that under certain circumstances pages are removed and loaded from memory too frequently.

The LRU policy is more difficult to implement but has been more attractive on the assumption that the least recently used page is a better candidate for removal than the least recently loaded page as in FIFO. The LRU algorithm can be implemented by associating a counter with every page that is in main memory. At fixed intervals of time, the counters associated with all pages presently in memory are incremented by 1. The least recently used page is the page with the highest count. The counters are often called aging registers, as their count indicates their age, that is, how long ago their associated pages have been referenced.

Memory management related hardware: A memory management system is a collection of hardware and software procedures for managing the various programs residing in memory. The memory management software is part of an overall operating system available in many computers.

Here we are concerned with the hardware unit associated with the memory management system. The basic components of a memory management unit are:

1. A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
2. A provision for sharing common programs stored in memory by different users.
3. Protection of information against unauthorized access between users and preventing users from changing operating system functions.

The dynamic storage relocation hardware is a mapping process similar to the paging system. The fixed page size used in the virtual memory system causes certain difficulties with respect to program size the logical structure of programs. It is more convenient to divide programs and data into logical parts called segments.

A segment is a set of logically related instructions or data elements associated with a given name.

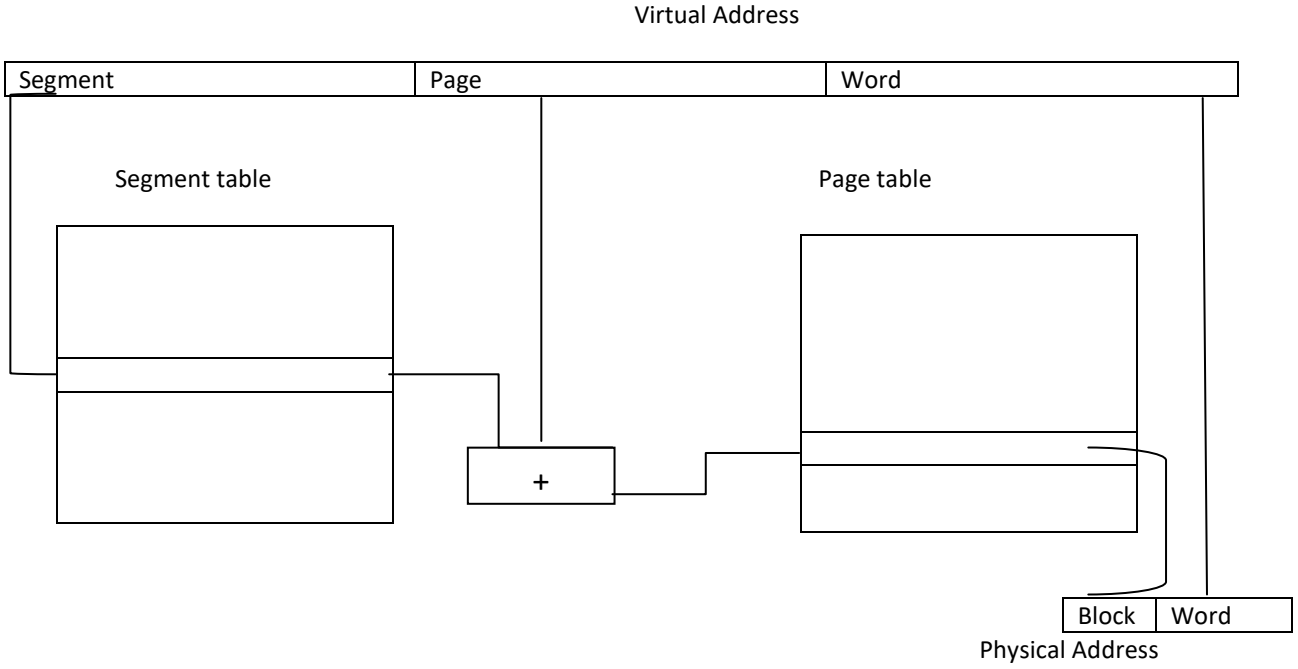
Segments may be generated by the programmer or by the operating system. Examples of segments are a subroutine, an array of data, a table of symbols, or a user's program. The sharing of common programs is an integral part of a multiprogramming system. For example, several users wishing to compile their C programs should be able to share a single copy of the compiler rather than each user having a separate copy in memory. Other system programs residing in memory are also shared by all users in a multiprogramming system without having to produce multiple copies.

The third issue in multiprogramming is protecting one program from unwanted interaction with another. An example of unwanted interaction is one user's unauthorized copying of another user's program. The address generated by a segmented program is called a logical address.

This is similar to a virtual address except that logical address space is associated with variable length segments rather than fixed length pages. The logical address may be larger than the physical memory address as in virtual memory, but it may also be equal, and sometimes even smaller than the length of the physical memory address. In addition to relocation information, each segment has protection information associated with it. Shared programs are placed in a unique segment in each user's logical address space so that a single physical copy can be shared. The function of the memory management unit is to map logical addresses into physical addresses similar to the virtual memory mapping concept.

vi) PAGING CONCEPT OF MEMORY COMPACTION

The property of logical space is that it uses variable length segments. The length of each segment is allowed to grow and contract according to the needs of the program being executed.



-Logical to physical address mapping-

The logical address is partitioned into 3 fields. The segments field specifies a segment number. The page field specifies the page within the segment and the word field gives the specific word within the page. A page field of k bits can specify up to 2^k pages. A segment number may be associated with just one page or with as many as 2^k pages. Thus the length of a segment would vary according to the number of pages that are assigned to it.

The mapping of the logical address into a physical address is done by means of two tables. The segment number of the logical address specifies the address for the segment table. The entry in the segment table is a pointer address for a page table base.

The page table base is added to the page number given in the logical address. The sum produces a pointer address to an entry in the page table. The value found in the page table provides the block number in physical memory. The concatenation of the block field with the word field produces the final physical mapped address. The two mapping tables may be stored in two separate small memories or in main memory.
In either case, a memory reference from the CPU will require three accesses to memory: one from the segment table, one from the page table, and the third from main memory. This would slow the system significantly when compared to a conventional system that required only one reference to memory.
To avoid this speed penalty, a fast associative memory is used to hold the most recently referenced table entries. This type of memory is sometimes called a **Translation Lookaside Buffer (TLB**). The first time a given block is referenced, it s value together with the corresponding segment and page numbers are entered into the associative memory. Thus the mapping process is first attempted by associative search with the given segment and page numbers. If it succeeds, the mapping delay is only that of the associative memory. If no match occurs, the result transformed into the associative memory for future reference using mapping.

vii) MEMORY SEGMENTATION

Memory protection can be assigned to the physical address or the logical address. The protection of memory through the physical address can be done by assigning to each block in memory a number of protection bits that indicate the type of access allowed to its corresponding block. Every time a page is moved from one block to another it would be necessary to update the block protection bits.

A much better place to apply protection is in logical address space rather than the physical address space. This can be done by including protection information within the segment table or segment register of the memory management hardware. The content of each entry in the segment table or a segment register is called a descriptor.

Base Address	Length	Protection

-Format of a typical segment descriptor-

A typical descriptor would contain, in addition to a base address field, one or two additional fields for protection purposes. The base address field gives the base of the page table address in a segmented-page organization or the block base address in a segment register organization. The is the address used in mapping from a logical to the physical address. The length field gives the segment size by specifying the maximum number of pages assigned to the segment.

The protection field in a segment descriptor specifies the access rights available to the particular segment. In a segmented-page organization, each entry in the page table may have its own protection field to describe the access rights of each page.

The protection information is set into the descriptor by the master control program of the operating system. Some of the access rights of interest that are used for protecting the programs residing in memory are:

1. Full read and write privileges
2. Read Only (write protection)
3. Execute only (program protection)
4. System only (operating system protection)

Full read and write privileges are given to a program when it is executing its own instructions. Write protection is useful for sharing system programs such as utility programs and other library routines. These system programs are stored in an area of memory where they can be shared by many users. They can be read by all programs, but no writing is allowed.

This protects them from being changed by other programs. It restricts the segment to be referenced only during the instruction fetch phase but not during the execute phase. Thus it allows the users to execute the segment program instructions but prevents them from reading the instructions as data for the purpose of copying their content. Portions of the operating system will reside in memory at any given time. These system programs must be protected by making them inaccessible to unauthorized users. The OS protection condition is placed in the descriptors of all operating system programs to prevent the occasional user from accessing operating system segments.