UNIT V: Memory organization: memory hierarchy – main memory – auxiliary memory – associative memory – cache memory – virtual memory.

Memory organization:

Memory hierarchy

The memory unit is an essential component in any digital computer since It Is needed for storing programs and data. They are used for storing system programs, large data, and other backup information. Only programs and data currently needed by the processor reside in main memory.

Cache memory

A special very-high-speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic. CPU logic is usually faster than main memory access time, with the result that processing speed is limited primarily by the speed of main memory. 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 cache is used for storing segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations.

Memory hierarchy

Auxiliary memory

Magnetic tapes

Magnetic disks

I/O processor

Main memory

CPU

Cache memory
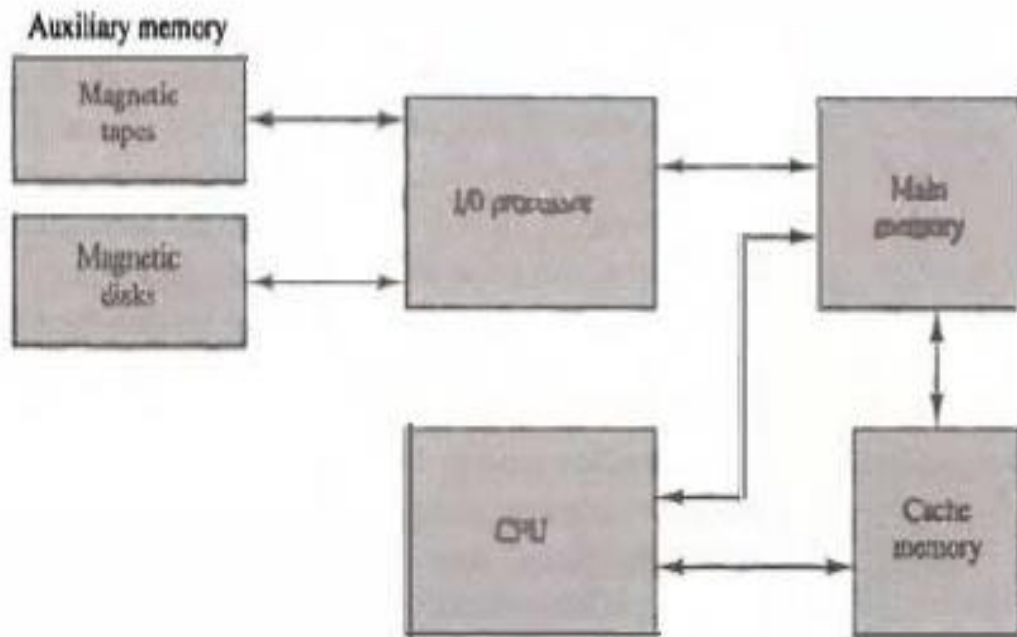
Main Memory

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 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 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 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. Originally, 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 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.

Among other things, 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.
Since RAM is volatile, its contents are destroyed when power is turned off.
The contents of ROM remain unchanged after power is turned off and on again. The startup 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 three 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.
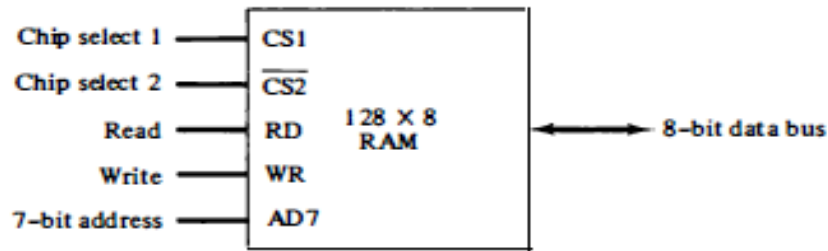
(a) Block diagram

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

(b) Function table
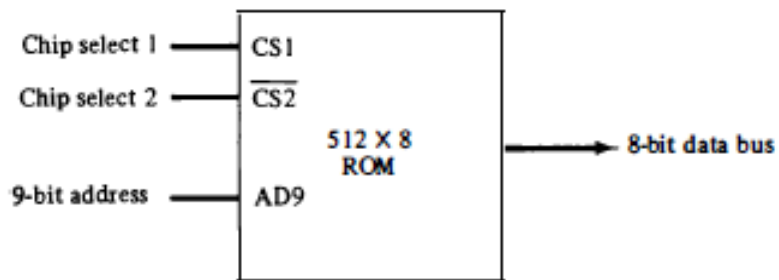
ROM CHIP

Auxiliary Memory

The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. Other components used, but not as frequently, are magnetic drums, magnetic bubble memory, and optical disks.

To understand fully the physical mechanism of auxiliary memory devices one must have a knowledge of magnetics, electronics, and electromechanical systems.
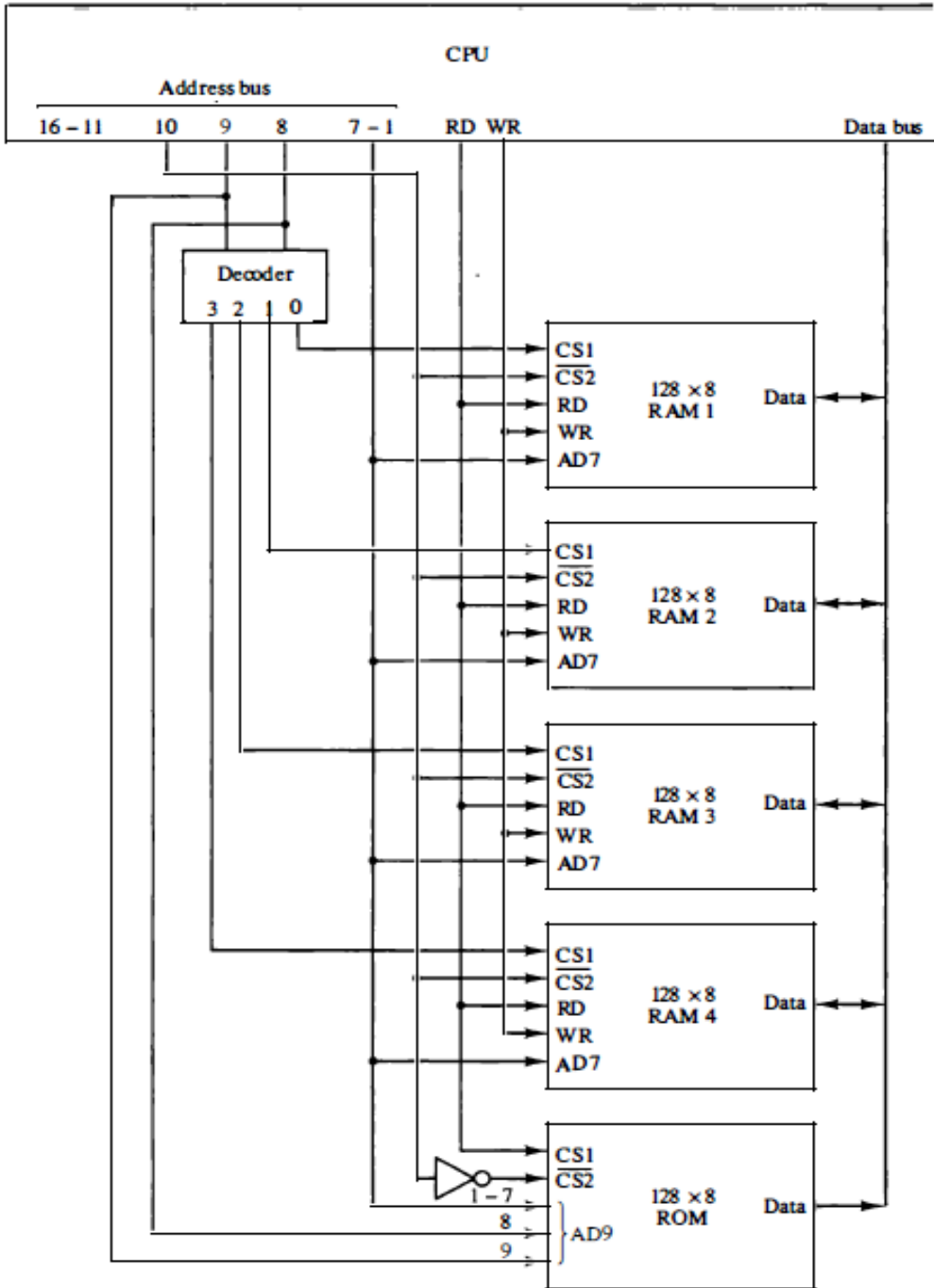
MEMORY CONNECTION TO CPU

Magnetic Disks

   A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material. Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.

All disks rotate together at high speed and are not stopped or started for access purposes. Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. In most systems, the minimum quantity of information which can be transferred is a sector. The subdivision of one disk surface into tracks and sectors is shown in Fig.
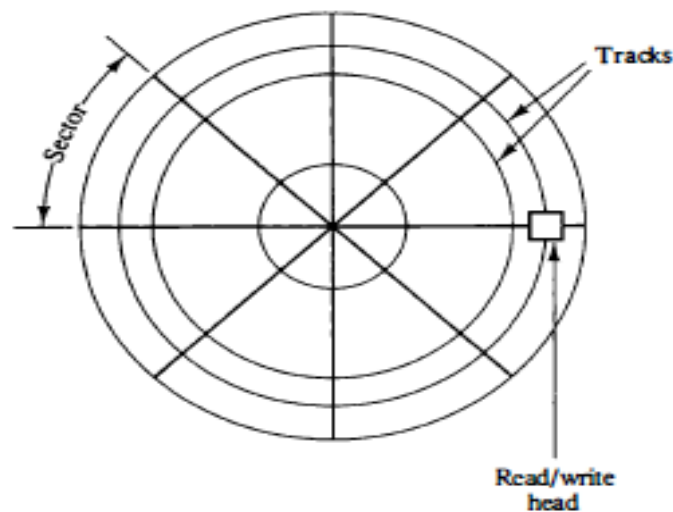
Magnetic Tape

A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit. The tape itself is a strip of plastic coated with a magnetic recording medium. Bits are recorded as magnetic spots on the tape along several tracks.

Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit. Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters.

Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks referred to as records. Gaps of unrecorded tape are inserted between records where the tape can be stopped. The tape starts moving while in a gap and attains its constant speed by the time it reaches the next record. Each record on tape has an identification bit pattern at the beginning and end. By reading the bit pattern at the beginning, the tape control identifies the record number. By reading the bit pattern at the end of the record, the control recognizes the beginning of a gap. A tape unit is addressed by specifying the record

number and the number of characters in the record. Records may be of fixed or variable length.

Associative Memory

    Many data-processing applications require the search of items in a table stored in memory. An assembler program searches the symbol address table in order to extract the symbol's binary equivalent. An account number may be searched in a file to determine the holder's name and account status. The established way to search a table is to store all items where they can be addressed in sequence. The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing the information read with the item being searched until a match occurs. The number of accesses to memory depends on the location of the item and the efficiency of the search algorithm. Many search algorithms have been developed to minimize the number of accesses while searching for an item in a random or sequential access memory.

The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed

by content is called an associative memory or content addressable memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or 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 which match the specified content and marks them for reading.

Because of its organization, the associative memory is uniquely suited to do parallel searches by data association. Moreover, searches can be done on an entire word or on a specific field within a word. 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 argument. For this reason, associative memories are used in applications where the search time is very critical and must be very short.

Hardware Organization

The block diagram of an associative memory is shown in Fig. 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. 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.

After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.

Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

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 I' s. Otherwise, only those bits in the argument that have I's in their corresponding position of the key register are compared.
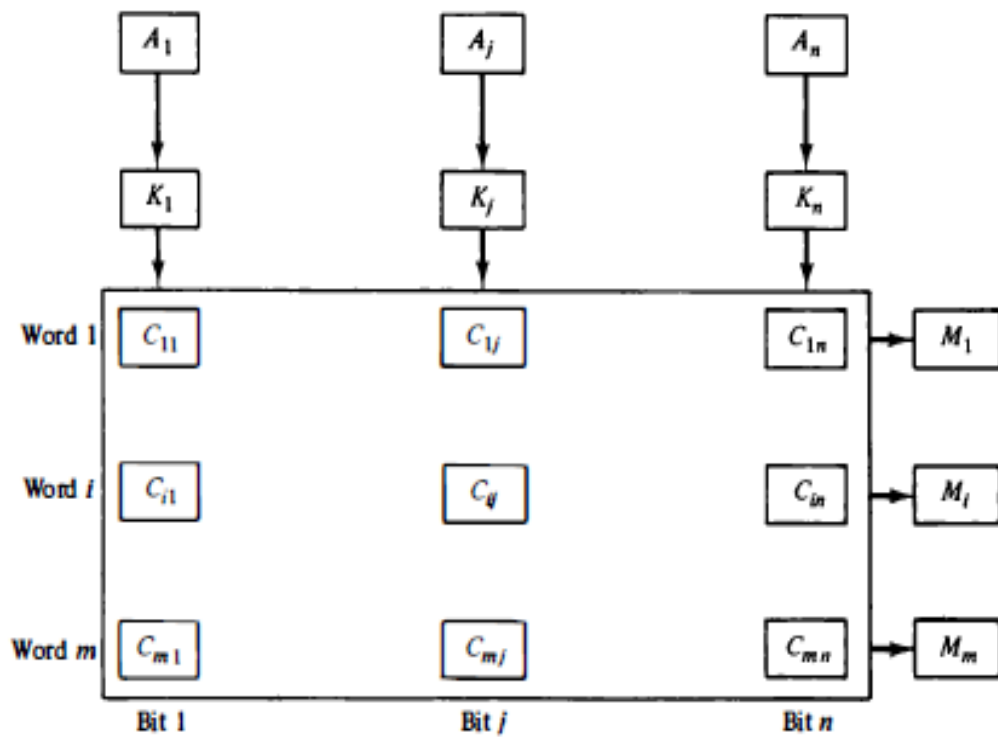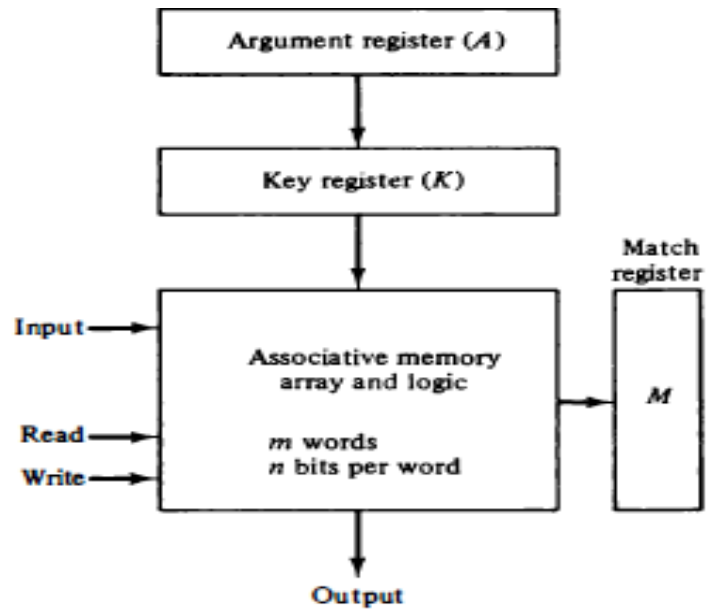
ASSOCIATIVE MEMORY

Match Logic

The match logic for each word can be derived from the comparison algorithm for two binary numbers. First, we neglect the key bits and compare the argument in A with the bits stored in the cells of the words. Word i is equal to the argument in A if A_i = $F_{,i}$ for j = 1, 2, . . . , n . Two bits are equal if they are both 1 or both 0. The equality of two bits can be expressed logically by the Boolean function where x_i = 1 if the pair of bits in position j are equal; otherwise, x_i = 0.

For a word i to be equal to the argument in A we must have all x_i variables equal to 1. This is the condition for setting the corresponding match bit M, to 1. The Boolean function for this condition is and constitutes the AND operation of all pairs of matched bits in a word.

Read Operation

If more than one word in memory matches the unmasked argument field, all the matched words will have 1's in the corresponding bit position of the match register. It is then necessary to scan the bits of the match register one at a time.

The matched words are read in sequence by applying a read signal to each word line whose corresponding M, bit is a 1 .

Write Operation

An associative memory must have a write capability for storing the information to be searched. Writing in an associative memory can take different forms, depending on the application. If the entire memory is loaded with new information at once prior to a search operation then the writing can be done by addressing each location in sequence. This will make the device a random access memory for writing and a content addressable memory for reading. The advantage here is that the address for input can be decoded as in a random access memory. Thus instead of having m address lines, one for each word in memory, the number of address lines can be reduced by the decoder to d lines, where m = 2'.

Cache Memory

Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference locality of reference . The reason for this property may be

understood considering that a typical computer program flows in a straight-line fashion with program loops and subroutine calls encountered frequently. When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop. Every time a given subroutine is called, its set of instructions are fetched from memory. Thus loops and subroutines tend to localize the references to memory for fetching instructions. To a lesser degree, memory references to data also tend to be localized. Table-lookup procedures repeatedly refer to that portion in memory where the table is stored. Iterative procedures refer to common memory locations and array of numbers are confined within a local portion of memory. The result of all these observations is the locality of reference property, which states that over a short interval of time, the addresses generated by a typical program refer to a few localized areas of memory repeatedly, while the remainder of memory is accessed relatively infrequently.

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory. It is placed between the CPU and main memory.

The basic operation of the cache is as follows. When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed. In this manner, some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

The performance of cache memory is frequently 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 produce 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.

The hit ratio is best measured experimentally by running representative programs in the computer and measuring the number of hits and misses during a given interval of time. Hit ratios of 0.9 and higher have been reported. This high ratio verifies the validity of the locality of reference property.

The average memory access time of a computer system can be improved considerably by use of a cache. If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of the fast cache memory.

For example, a computer with cache access time of 100 ns, a main memory access time of 1000 ns, and a hit ratio of 0.9 produces an average access time of 200 ns. This is a considerable improvement over a similar computer without a cache memory, whose access time is 1000 ns .The basic characteristic of cache memory is its fast access time.

Therefore, very little or no time must be wasted when searching for words in the cache.

The transformation of data from main memory to Cache memory is referred to as a mapping process. Three types of mapping procedures are of practical interest when considering the organization of cache memory:

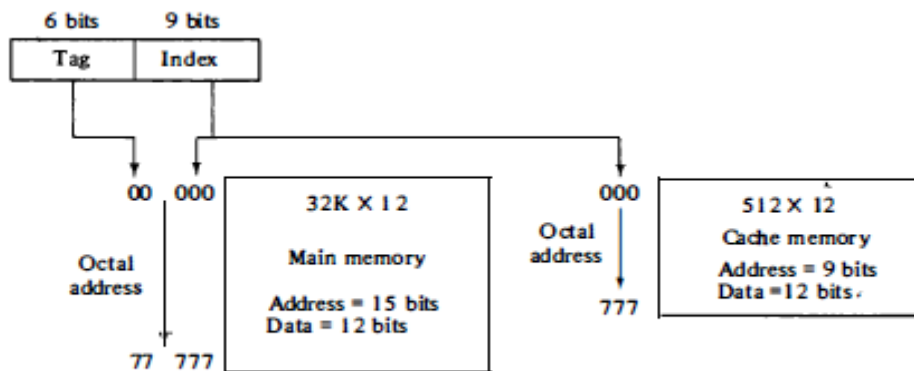1. Associative mapping
2. Direct mapping
3. Set-associative mapping
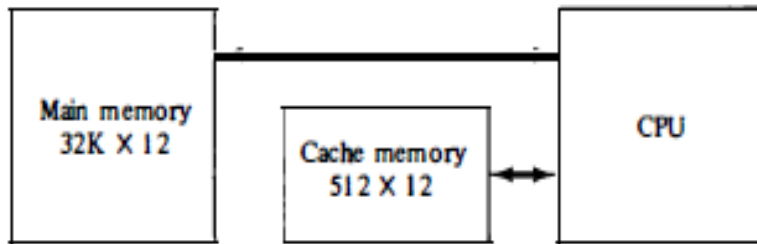
Associative Mapping
The fastest and most flexible cache organization uses an associative memory
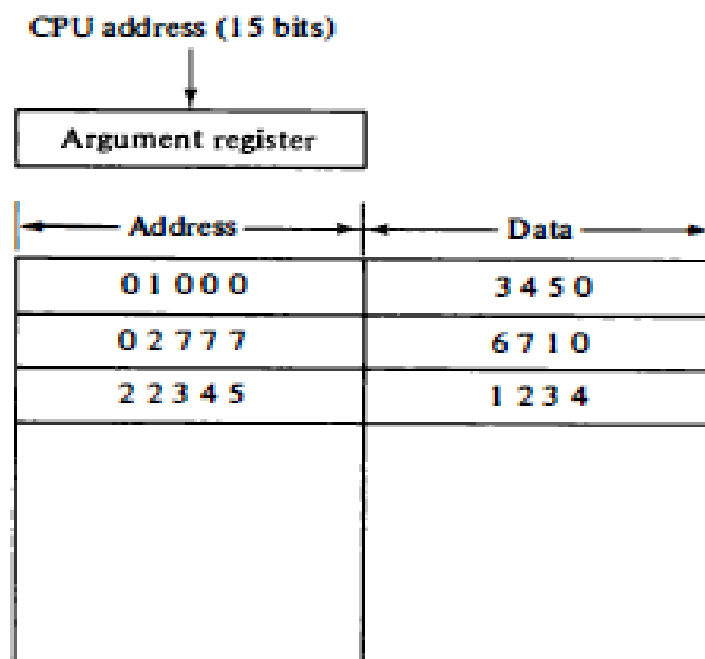
Direct Mapping

Associative memories are expensive compared to random-access memories because of the added logic associated with each cell

	6 bits	9 bits
	Tag	Index

00 000

32K X 12

Main memory

Address = 15 bits
Data = 12 bits

Octal address

77 777

000

Octal address

777

512 X 12

Cache memory

Address = 9 bits
Data = 12 bits.

Memory address	Memory data
00000	1 2 2 0
00777	2 3 4 0
01000	3 4 5 0
01777	4 5 6 0
02000	5 6 7 0
02777	6 7 1 0

(a) Main memory

Index address	Tag	Data
000	0 0	1 2 2 0
777	0 2	6 7 1 0

(b) Cache memory

Set-Associative Mapping

It was mentioned previously that the disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time. A third type of cache organization,called set-associative mapping, is an improvement over the directmapping organization in that each word of cache can store two or more words of memory under the same index address. Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set. An example of a set-associative cache organization for a set size of two is shown in Fig.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

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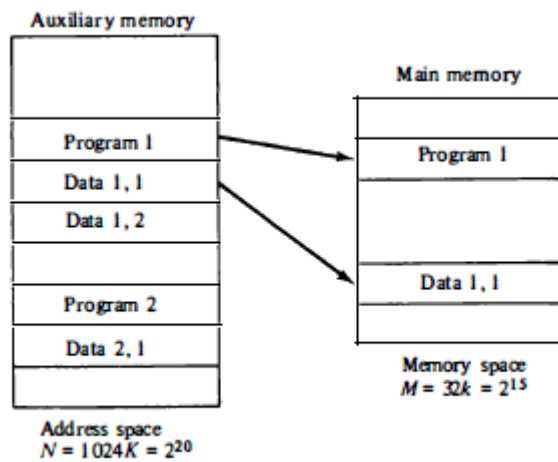
address space . An address in main memory is called a location or physical address . The set of such locations is called the memory space .

Thus the address space is the set of addresses generated by programs as they reference instructions and data; the memory space consists of the actual main memory locations directly addressable for processing. In 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.
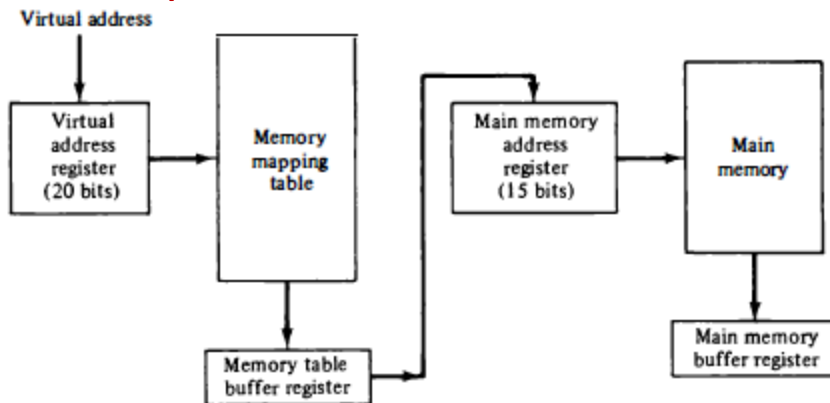
As an illustration, consider a computer with a main-memory capacity of 32K words (K = 1024). Fifteen bits are needed to specify a physical address in · memory since 32K = 215• Suppose that the computer has available auxiliary memory for storing 220 = 1024K words. Thus auxiliary memory has a capacity 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 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.

Auxiliary memory

| Program 1 |
| Data 1, 1 |
| Data 1, 2 |

| |
| Program 2 |
| Data 2, 1 |

Address space
$N = 1024 K = 2^{20}$

Main memory

| Program 1 |

| Data 1, 1 |

Memory space
$M = 32k = 2^{15}$

Relation between address and memory space in virtual memory

Virtual address

| Virtual address register (20 bits) | → | Memory mapping table | → | Main memory address register (15 bits) | → | Main memory |

Memory table buffer register

Main memory buffer register

Memory table for mapping virtual address

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. The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each. The term page refers to groups of address space of the same 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.

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.
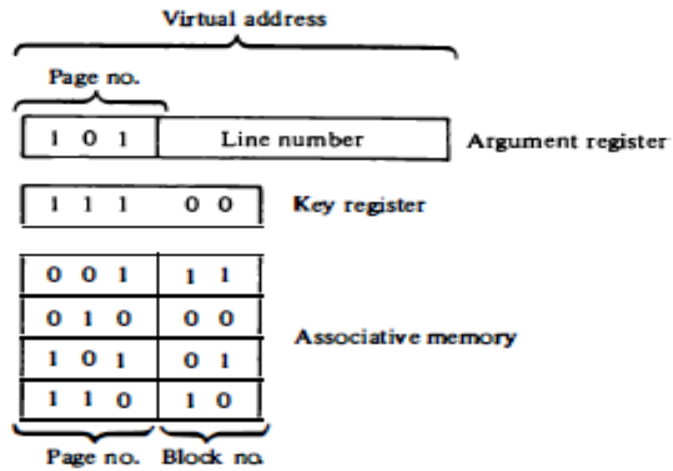
Associative Memory Page Table

A random-access memory page table is inefficient with respect to storage utilization. In the example of Fig. we observe that eight words of memory are needed, one for each page, but at least four words will always be marked empty because

main memory cannot accommodate more than four blocks.

In general, a system with n pages and m blocks would require a memory-page table of n locations of which up to m blocks will be marked with block numbers and all others will be empty. As a second numerical example, consider an address space of 1024K words and memory space of 32K words. If each page or block contains 1K words, the number of pages is 1024 and the number of blocks 32. The capacity of the memory-page table must be 1024 words and only 32 locations may have a presence bit equal to 1. At any given time, at least 992 locations will be empty and not in use.

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 its corresponding block number is extracted.

Virtual address

Page no.

| 1 0 1 | Line number | Argument register |

| 1 1 1 | 0 0 | Key register |

0 0 1	1 1
0 1 0	0 0
1 0 1	0 1
1 1 0	1 0

Associative memory

Page no.   Block no.

Associative memory page table