Chapter 10
Memory Organization

10.1 Memory Hierarchy

→ 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***. Devices that provide backup 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.

→ The memory hierarchy 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.

→ The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor,

→ Access time of CPU and main memory are different. So, to co-ordinate the speed between these, a fast memory is needed called as cache memory.

→ While the I/O processor manages data transfer between auxiliary memory and main memory, the cache organization is concerned with the transfer of information between main memory and CPU.

→ As the storage capacity of the memory increases, the cost per bit for storing binary information decreases and the access time of the memory becomes longer.

→ The auxiliary memory has a large storage capacity, is relatively inexpensive, but has low access speed compared to main memory.

→ The cache memory is very small, relatively very small, relatively expensive, and has very high access speed.

→ CPU has direct access to both cache and main memory but not to auxiliary memory. The transfer from auxiliary to main memory is usually done by means of direct memory access of large blocks of data.

→ Many OS are designed to enable the CPU to process a number of independent programs concurrently. This concept, called ***multiprogramming***, refers to the existence of two or more programs in different parts of memory hierarchy at the same time. In this way, it is possible to keep all parts of the computer busy by working with several programs in sequence.

→ The part of the computer system that supervises the flow of information between auxiliary memory and main memory is called ***memory management system***.

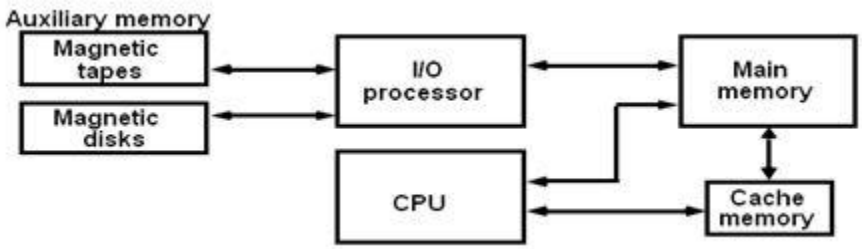
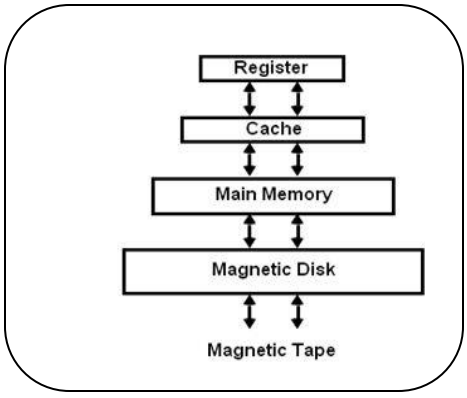


Fig: Memory hierarchy in a computer system



10.2 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.

→ Integrated circuit RAM chips are available in two possible modes: ***static*** and ***dynamic***.

→ The static RAM consists essentially of internal flip-flops. 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

capacitor tends to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.

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

→ 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 the 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 as ***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.

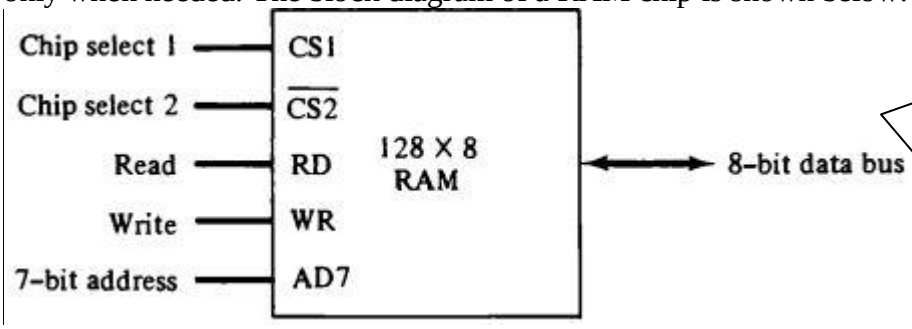
→ When power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader which loads a portion of the OS from disk to main memory and control is then transferred to the OS, which prepares the computer for general use.

RAM and ROM chips

RAM and ROM chips are available in a variety of sizes. If we need larger memory for the system, it is necessary to combine a number of chips to form the required memory size.

RAM Chips

A RAM chip is better suited to communicate with CPU if it has one or more control inputs that select the chip only when needed. The block diagram of a RAM chip is shown below:



→ Requires 7-bit address and an 8-bit bidirectional data bus

→ Chip select (CS) control inputs are for enabling the chip only when it is selected by CPU.

Fig: Typical RAM chip (128 words of eight bits each)

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

→ The unit is in operation only when CS1=1 and (CS2)'=0.

→ High impedance state indicates open circuit i.e. output does not carry a signal and has no logic significance.

Fig: Function table for RAM chip

ROM Chips

Since a ROM chip can only read, data bus is unidirectional (output mode only).

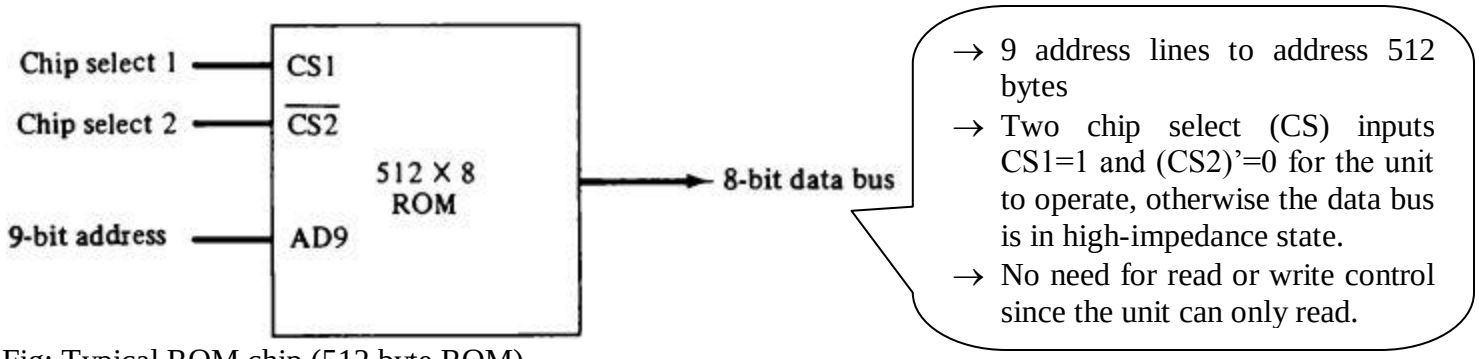


Fig: Typical ROM chip (512 byte ROM)

Memory Address Map

The addressing of memory can be established by means of a table that specifies the memory address assigned to each RAM or ROM chip. This table is called memory address map and is a pictorial representation of assigned address space for particular chip.

Example: Suppose computer system needs 512 bytes of RAM and 512 bytes of ROM.

Component	Hexadecimal address	Address bus 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

→ Component column specifies RAM or ROM chip. We use four 128 words RAM to make 512 byte size.
→ Hexadecimal address column assigns a range of addresses for each chip.
→ 10 lines in address bus column: lines 1 through 7 for RAM and 1 through 9 for ROM. Distinction between RAM and ROM chip is made by line 10. When line 10 is 1, it selects ROM and when it is 0, CPU selects RAM.
→ X's represents a binary number ranging from all-0's to all-1's.

10.3 Associative Memory

→ Also called as Content-addressable memory (CAM), associative storage, or associative array
→ Content-addressed or associative memory refers to a memory organization in which the memory is accessed by its content (as opposed to an explicit address).
→ It is a special type of computer memory used in certain very high speed searching applications.
→ In standard computer memory (random access memory or RAM), the user supplies a memory address and the RAM returns the data word stored at that address.
→ In CAM, the user supplies a data word and then CAM searches its entire memory to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found.
→ CAM is designed to search its entire memory in a single operation.
→ It is much faster than RAM in virtually all search applications.
→ An associative memory is more expensive than RAM, as each cell must have storage capability as well as logic circuits for matching its content with an external argument.
→ Associative memories are used in applications where the search time is very critical and short.

→ Associative memories are expensive compared to RAMs because of the add logic associated with each cell.

Hardware Organization

→ Associative memory consists of a memory array and logic for m words and 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 words. 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 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 1's. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus, the key provides a mask or identifying piece of information which specifies how the reference to memory is made.

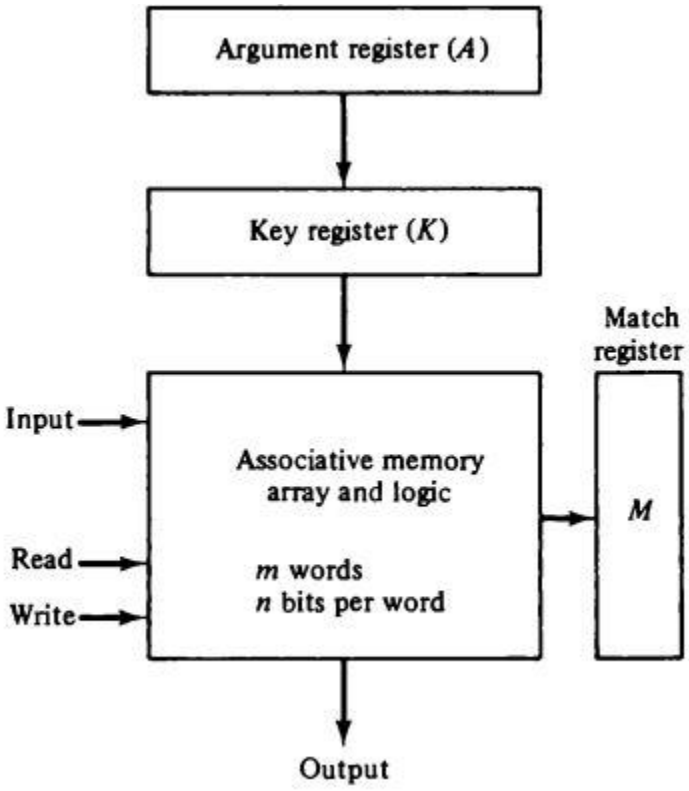


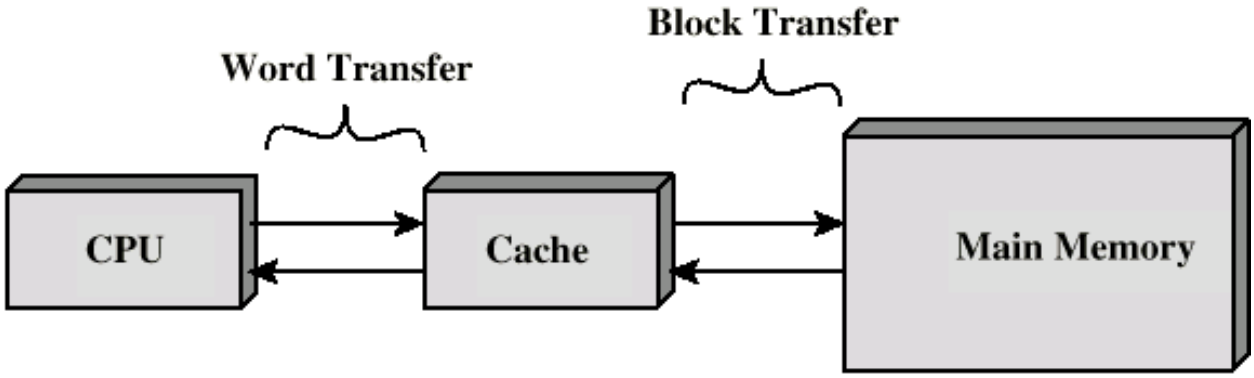
Fig: Block Diagram of Associative Memory

E.g. A=10111100
K=111000000

Word1	100111100	no match
Word2	101000001	match

10.4 Cache Memory: cache mapping techniques

Cache (pronounced cash) memory is extremely fast memory that is built into a computer's central processing unit (CPU), or located next to it on a separate chip. The CPU uses cache memory to store instructions that are repeatedly required to run programs, improving overall system speed. The advantage of cache memory is that the CPU does not have to use the motherboard's system bus for data transfer. Whenever data must be passed through the system bus, the data transfer speed slows to the motherboard's capability. The CPU can process data much faster by avoiding the bottleneck created by the system bus.



→ A cache is a small amount of very fast associative memory.
→ It sits between normal main memory and CPU.
→ When CPU needs to access contents of memory location, then cache is examined for this data.
  - If present, get from cache (fast).
  - If not present, read required block from main memory to cache, then deliver from cache to CPU.
→ Cache includes tags to identify which block of main memory is in each cache slot.
→ The performance of cache memory is frequently measured in terms of a quantity called **hit ratio.** When CPU refers to memory and finds the word in cache, then it is said to produce hit. If word is not found in cache, it is in main memory and it counts as a miss. The ratio of the number of hits (success in finding the words in cache) to the total CPU references to memory (hits + misses) is known as hit ratio.
→ The basic characteristic of cache memory is its fast access time.

Cache Mapping

The process of transferring the data from main memory to cache is known as mapping process. There are three types of cache mapping techniques:

- ❖ Associative mapping
- ❖ Direct mapping
- ❖ Set-associative mapping

→ The main memory can store 32K words of 12 bits each. The cache is capable of storing 512 of these words at any given time. For every word stored in cache, there is a duplicate copy in main memory. The CPU communicates with both memories. It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

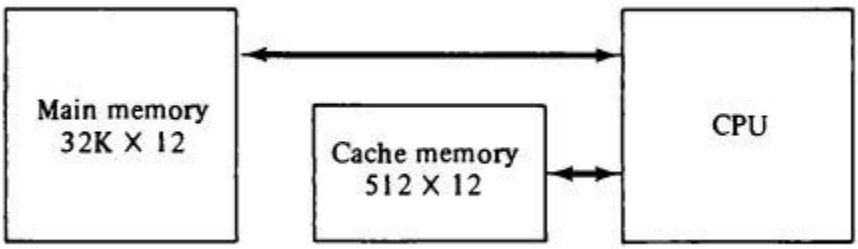


Fig: Example of Cache memory

Associative Mapping

The fastest and most flexible cache organization uses an associative memory. This organization is illustrated in Fig. The associative memory stores both the address and content (data) of the memory word. This permits any location in cache to store any word from main memory. The diagram shows three words presently stored in the cache. The address value of 15 bits is shown as a five digit octal number and its corresponding 12 bit word is shown as a four digit octal number. A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address. If the address is found, the corresponding 12 bit data is read and sent to the CPU. If no match occurs, the main memory is accessed for the word. The address-data pair

is then transferred to the associative cache memory. If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache. The decision as to what pair is replaced is determined from the replacement algorithm that the designer chooses for the cache. A simple procedure is to replace cells of the cache in round robin order whenever a new word is requested from main memory. This constitutes a first in first out (FIFO) replacement policy.

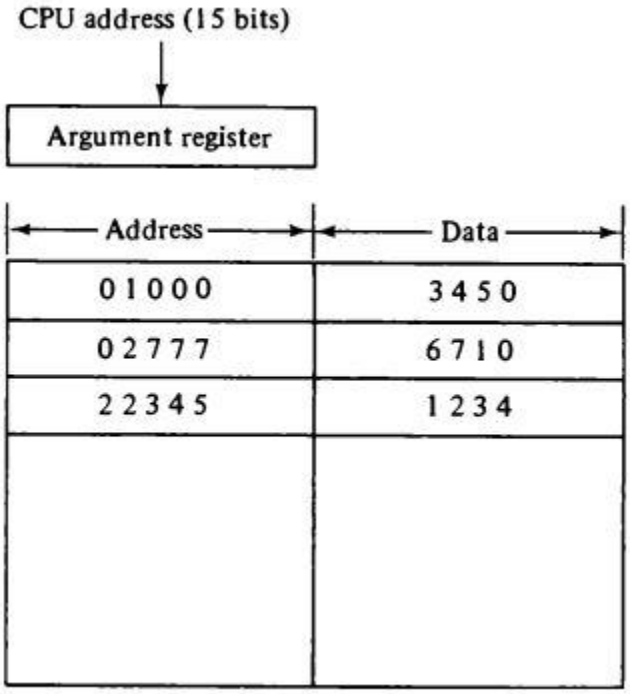


Fig: Associative mapping cache (all numbers in octal)

Direct Mapping

→ The CPU address of 15 bits is divided into two fields. The nine least significant bits constitute the *index field* and the remaining six bits form the *tag field*.

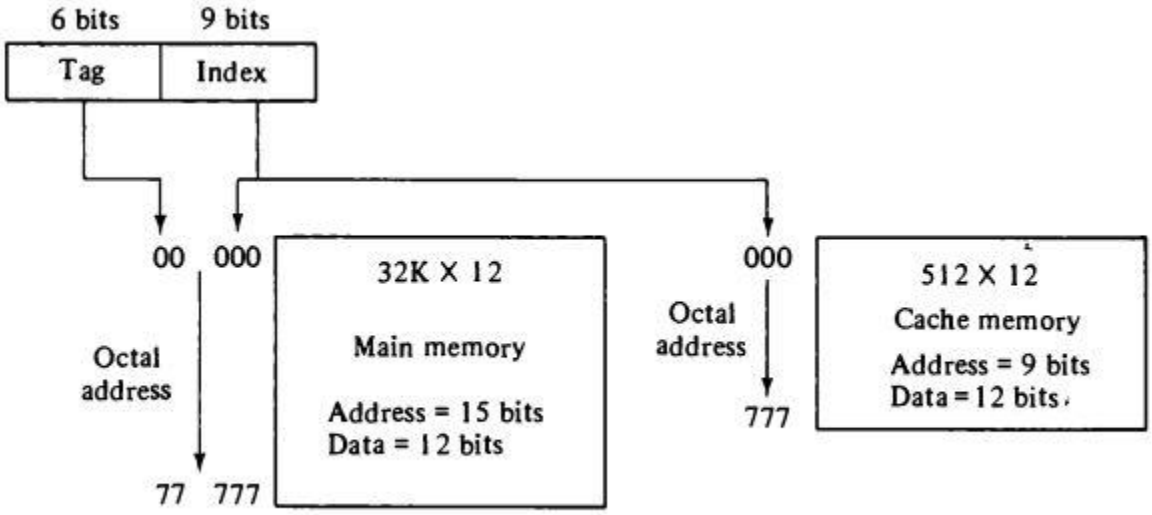


Fig: Addressing relationships between main and cache memories.

→ The figure shows that main memory needs an address that includes both the tag and the index bits. The number of bits in the index field is equal to the number of address bits required to access the cache memory.

→ The n bit memory address is divided into two fields: k bits for the index field and n-k bits for the tag field. The direct mapping cache organization uses the n bit address to access the main memory and the k bit index to access the cache.

→ Each word in cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access the cache.

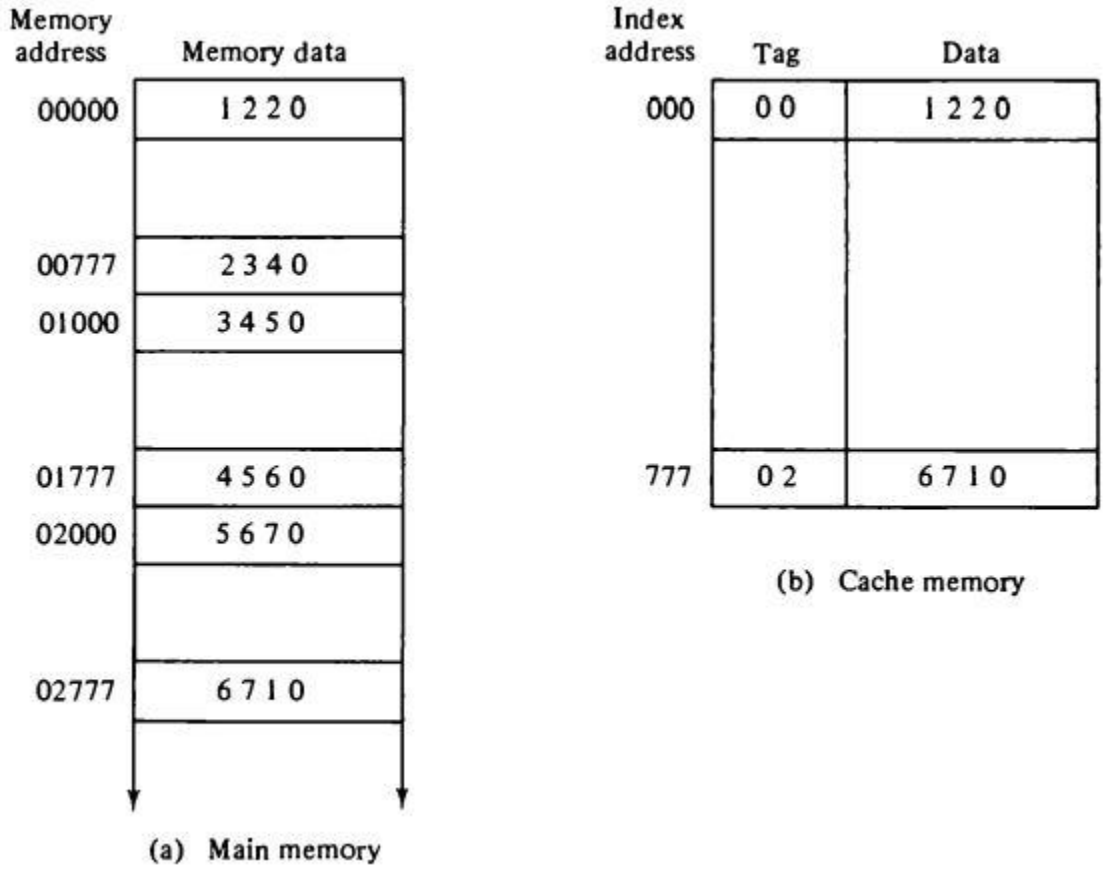


Fig: Direct mapping cache organization.

→ The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value.

→ 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.

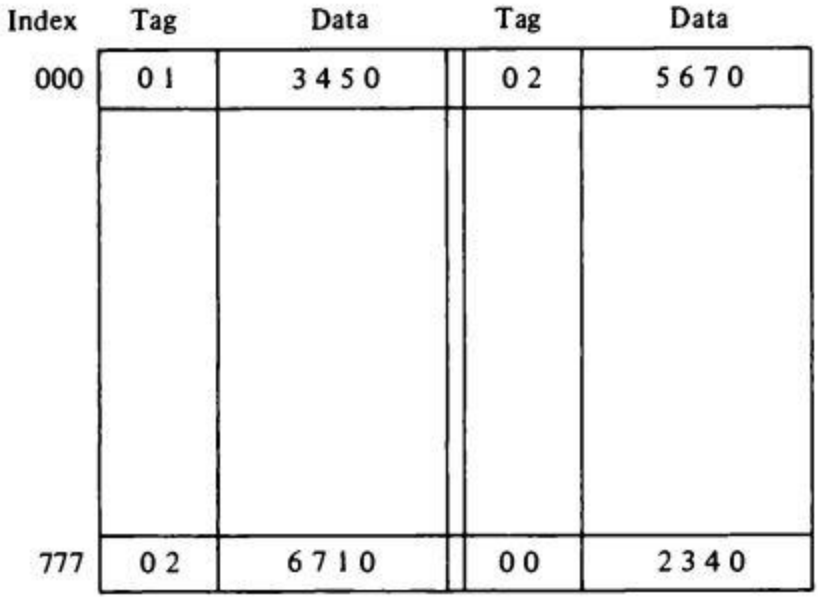
Set-Associative Mapping

→ It is an improvement over the direct mapping 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.

→ Each index address refers to two data words and their associated tags. Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits.

→ An index address of nine bits can accommodate 512 words. Thus, the size of cache memory is 512*36. It can accommodate 1024 words of main memory since each word of cache contains two data words.



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

Fig: Two-way set associative mapping cache.

→ The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777.

→ When a miss occurs in a set associative cache and the set is full, it is necessary to replace one of the tag data items with a new value.

→ The most common replacement algorithms used are: *random replacement*, *first in first out (FIFO)*, and *least recently used (LRU).*

10.5 Virtual Memory

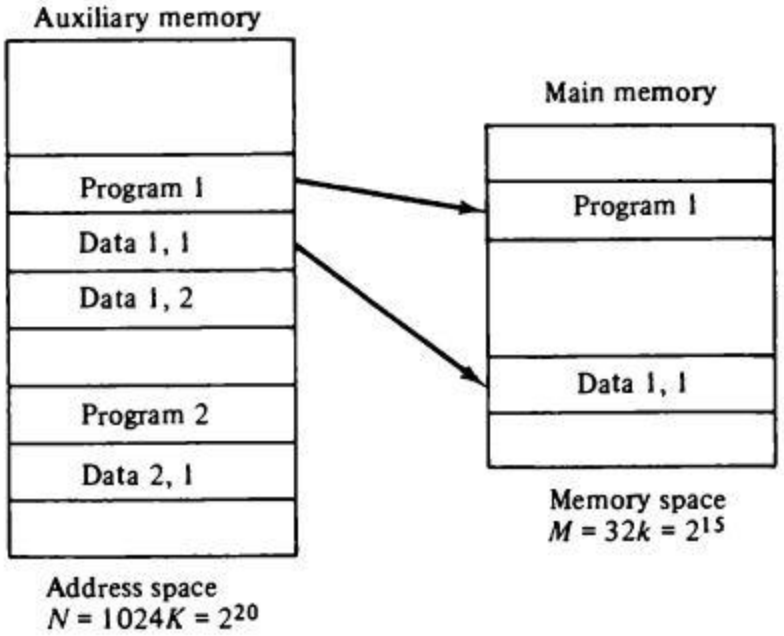
→ A virtual memory system attempts to optimize the use of the main memory (the higher speed portion) with the hard disk (the lower speed portion). In effect, **virtual memory** is a technique for using the secondary storage to extend the apparent limited size of the physical memory beyond its actual physical size. It is usually the case that the available physical memory space will not be enough to host all the parts of a given active program.

→ Virtual memory gives programmers the illusion that they have a very large memory and provides mechanism for dynamically translating program-generated addresses into correct main memory locations. 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 the programmer is a virtual address (virtual memory addresses) and the set of such addresses is 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 the programs as they reference instructions and data; the memory space consists of actual main memory locations directly addressable for processing. Generally, the address space is larger than the memory space.

Example: consider main memory: 32K words (K = 1024) = 2^{15} and auxiliary memory 1024K words = 2^{20}. Thus we need 15 bits to address physical memory and 20 bits for virtual memory (virtual memory can be as large as we have auxiliary storage).



→ Here auxiliary memory has the capacity of storing information equivalent to 32 main memories.
→ Address space N = 1024K
→ Memory space M = 32K
→ In multiprogram computer system, programs and data are transferred to and from auxiliary memory and main memory based on the demands imposed by the CPU.

Fig: Relation between address and memory space in a virtual memory system

In virtual memory system, address field of an instruction code has a sufficient number of bits to specify all virtual addresses. In our example above we have 20-bit address of an instruction (to refer 20-bit virtual address) but physical memory addresses are specified with 15-bits. So a table is needed to map a virtual address of 20-bits to a physical address of 15-bits. Mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.

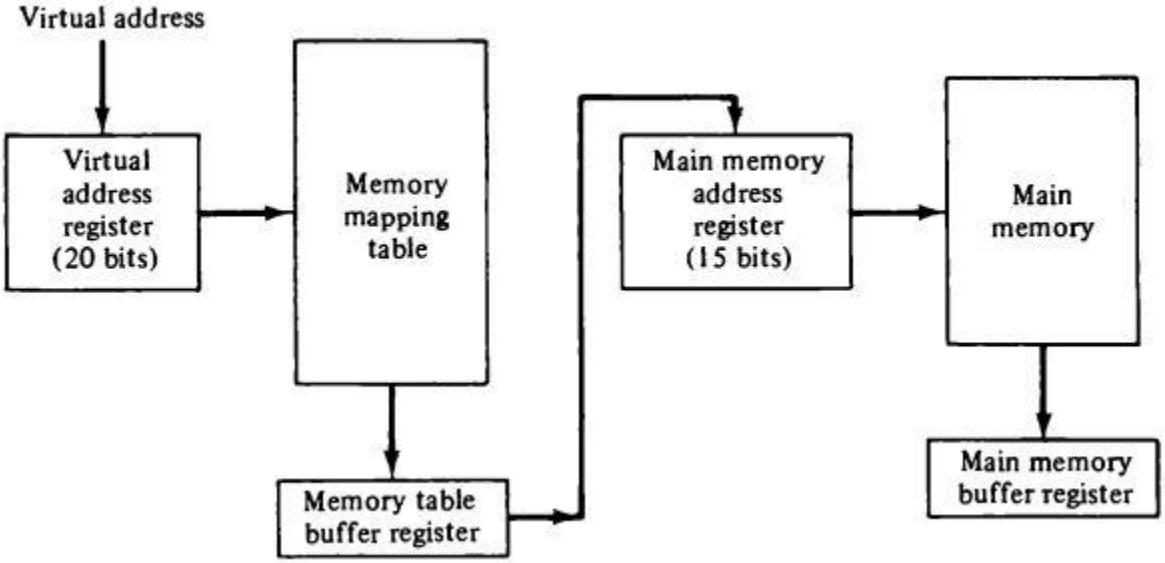


Fig: Memory table for mapping a virtual address

10.6 Memory Management Hardware

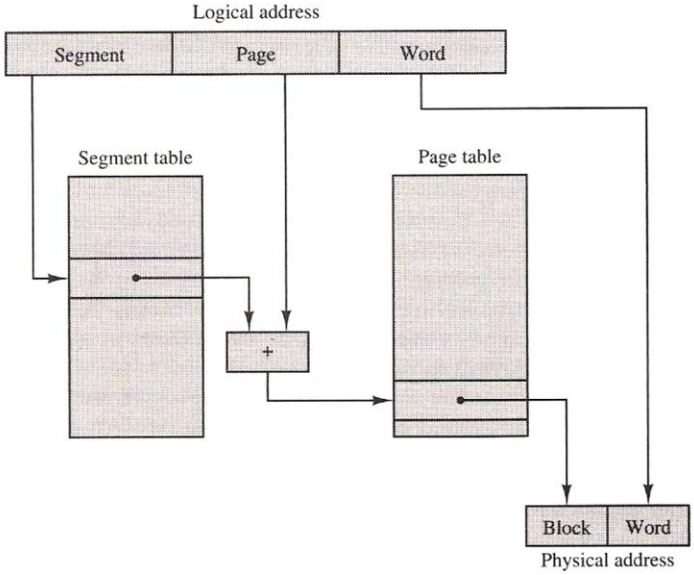
A memory management system is a collection of hardware and software procedures for managing various programs (effect of multiprogramming support) residing in memory. Basic components of memory management unit (MMU) are:

→ A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
→ A provision for sharing common programs by multiple users.
→ Protection of information against unauthorized access.

The dynamic storage relocation hardware is a mapping process similar to paging system.
Segment: It is more convenient to divide programs and data into logical parts called segments despite of fixed-size pages. A **segment** is a set of logically related instructions or data elements. Segments may be generated by the programmer or by OS. Examples are: a subroutine, an array of data, a table of symbols or user's program.

Logical address: The address generated by the segmented program is called a *logical address*. This is similar to virtual address except that logical address space is associated with variable-length segments rather than fixed-length pages.

Segmented-Page Mapping
The length of each segment is allowed to grow and contract according to the needs of the program being executed. One way of specifying the length of a segment is by associating with it a number of equal-sized pages.
Consider diagram below:
Logical address = Segment + Page + Word
Where **segment** specifies segment number, **page** field specifies page within the segment and **word** field specifies specific word within the page.



→ Mapping of logical address to physical address is done by using two tables: segment and page table.
→ The entry in the segment table is a pointer address for the page table base, which is then added to page number (given in logical address). The sum point to some entry in page table and content of that page is the address of physical block. The concatenation of block field with the word field produces final **physical mapped address**.

Fig: Logical to Physical address mapping

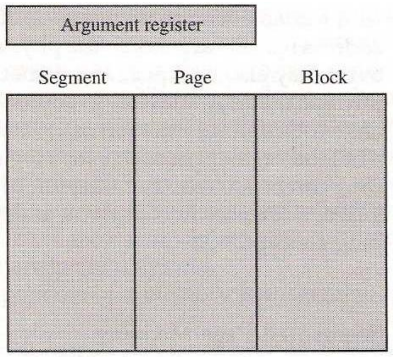


Fig: Associative Memory: Translation Lookaside Buffer (TLB)

→ This is a fast associative memory (TLB) and holds most recently referenced entries.
(Alternatively we could store above two tables: segment table and page table, in two separate small memories which really increases the CPU access time).

H\W, *See Numerical example to clear the concept of MMU (Computer System Architecture, 3rd edition, page no. 481, Morris Mano)*