Unit - 04

Storage
&
Memory Hierarchy

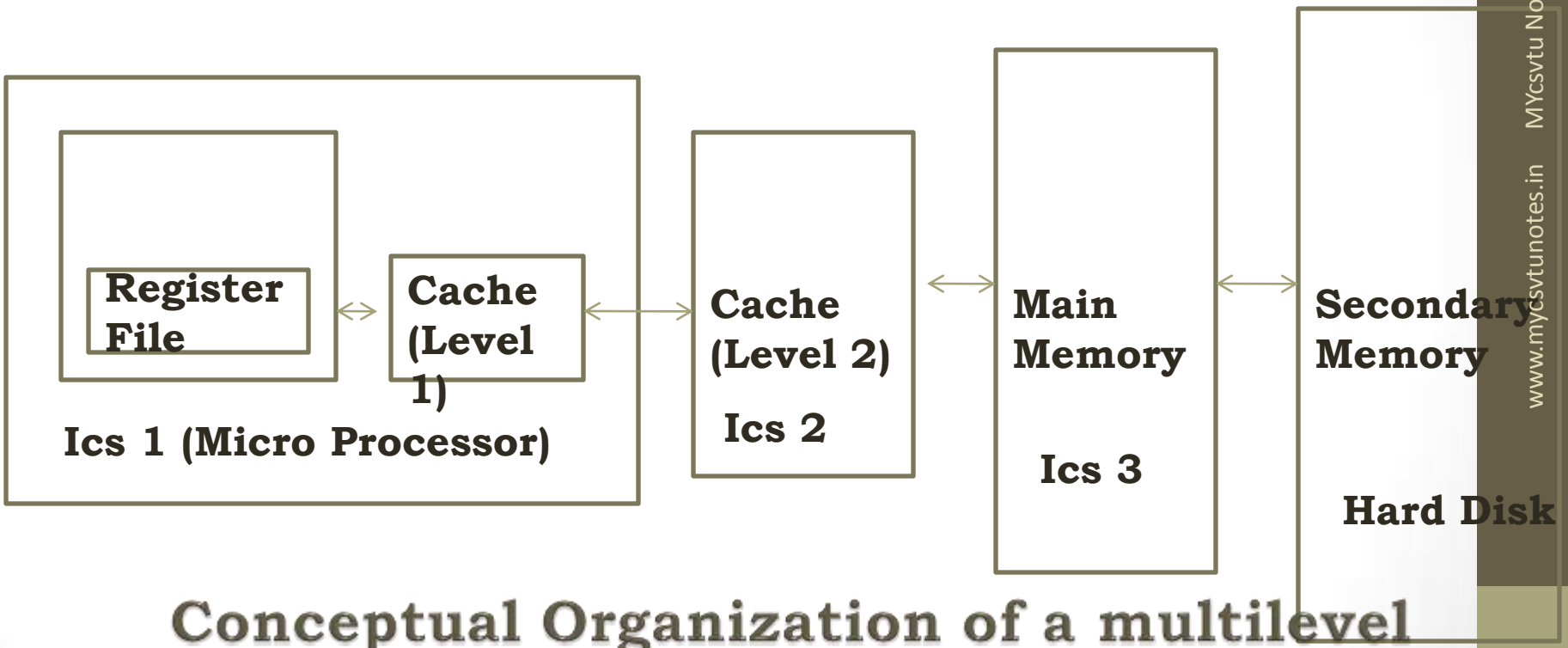
Deepak Bhalla (Asst. Professor (MCA & IT Dept.))

Memory

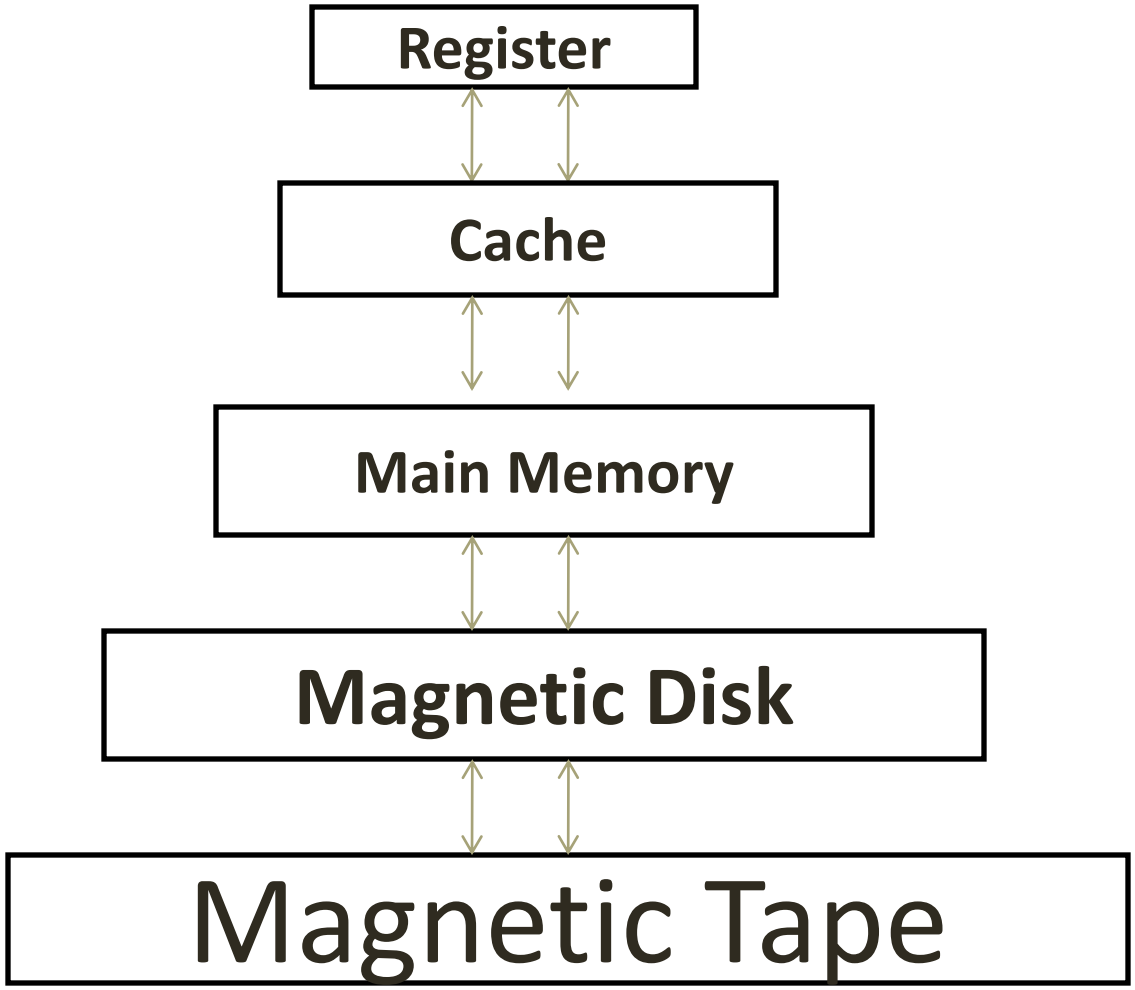
- **Memory is an integral part of a computer system.**
- **Its primary function is to store all information required by the system.**
- **Typically a memory unit holds programs and data.**
- **The system performance is largely dependent on the organization, storage capacity, and speed of operation of the memory system.**

Memory Types

- **The information storage components of a computer can be placed in 4 groups as shown in the figure:**

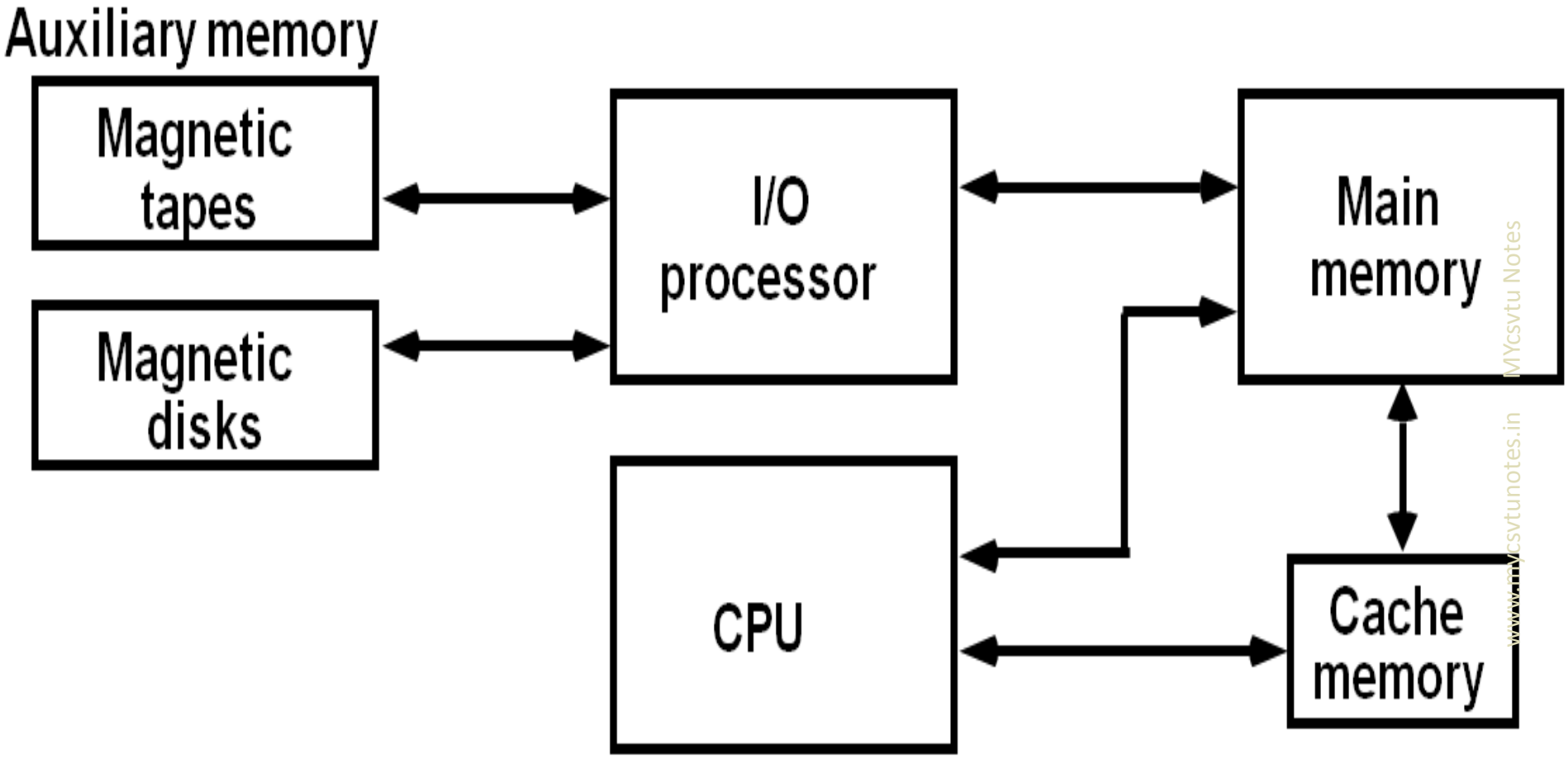


Conceptual Organization of a multilevel Memory System in Computer



Memory Hierarchy

- **The capacity of total memory can be visualized as being a hierarchy of components.**
- **The memory hierarchy system includes all the storage devices employed in a computer system.**
- **It ranges from slow but high capacity auxiliary memory to relatively faster main memory, to an even smaller and faster cache memory accessible to the high speed processing logic.**
- **Figure shows the components in a typical memory hierarchy.**



Memory Hierarchy

Deepak Bhalla (Asst. Professor (MCA & IT Dept.))

- **At the bottom of the hierarchy , there are relatively slow magnetic tapes used to store removable files.**
- **Magnetic disks used as backup storage.**
- **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.**
- **When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.**
- **A special very high speed memory called cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.**

- **This memory is generally employed in a computer system 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 speed is to employ an extremely fast, small cache between the CPU and the main memory whose access time is close to processor logic clock cycle time.**

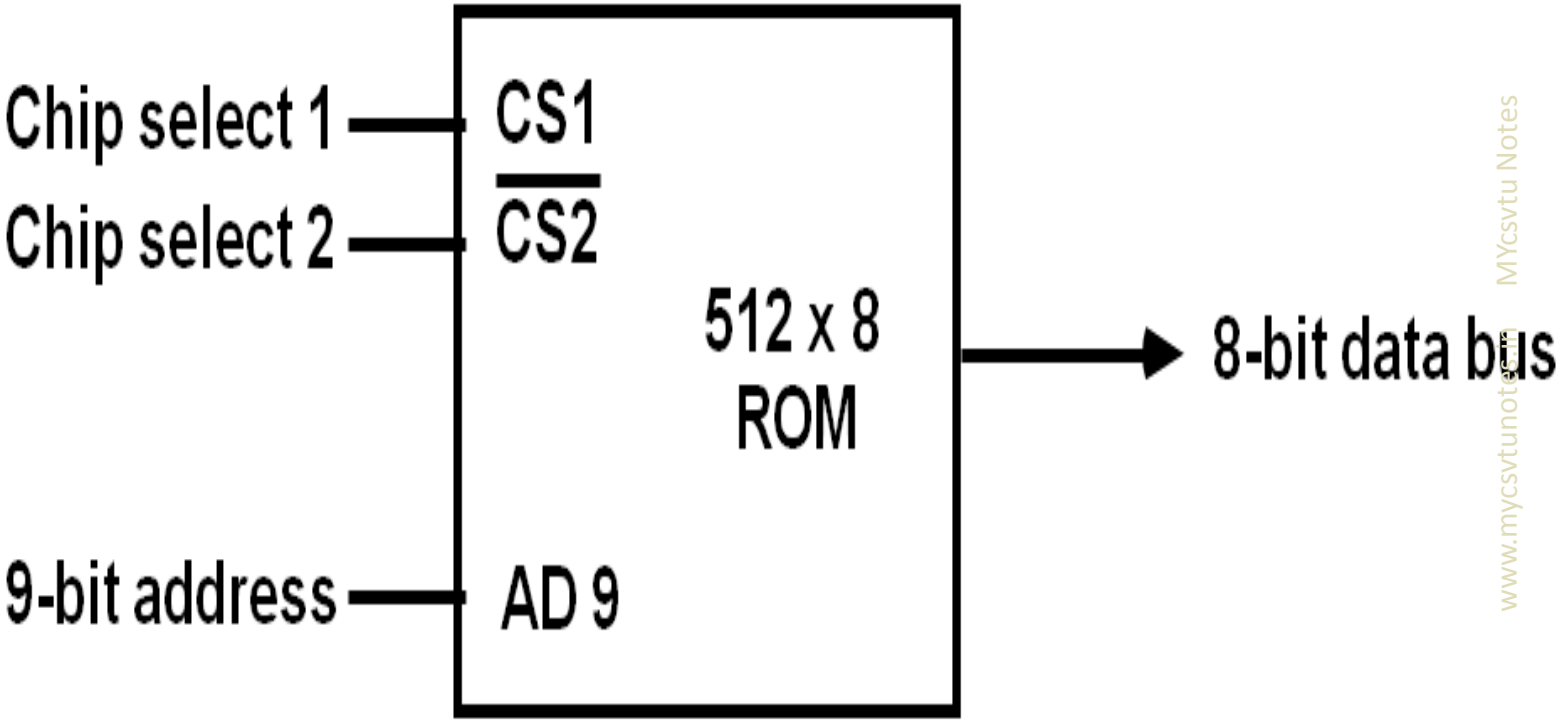
- **The cache stores segment of programs currently being executed in the CPU and temporary data frequently needed in the present calculations.**
- **Thus by making programs and data available at rapid rate, it is possible to increase the performance rate of computer.**

Memory Address Map for RAM and ROM

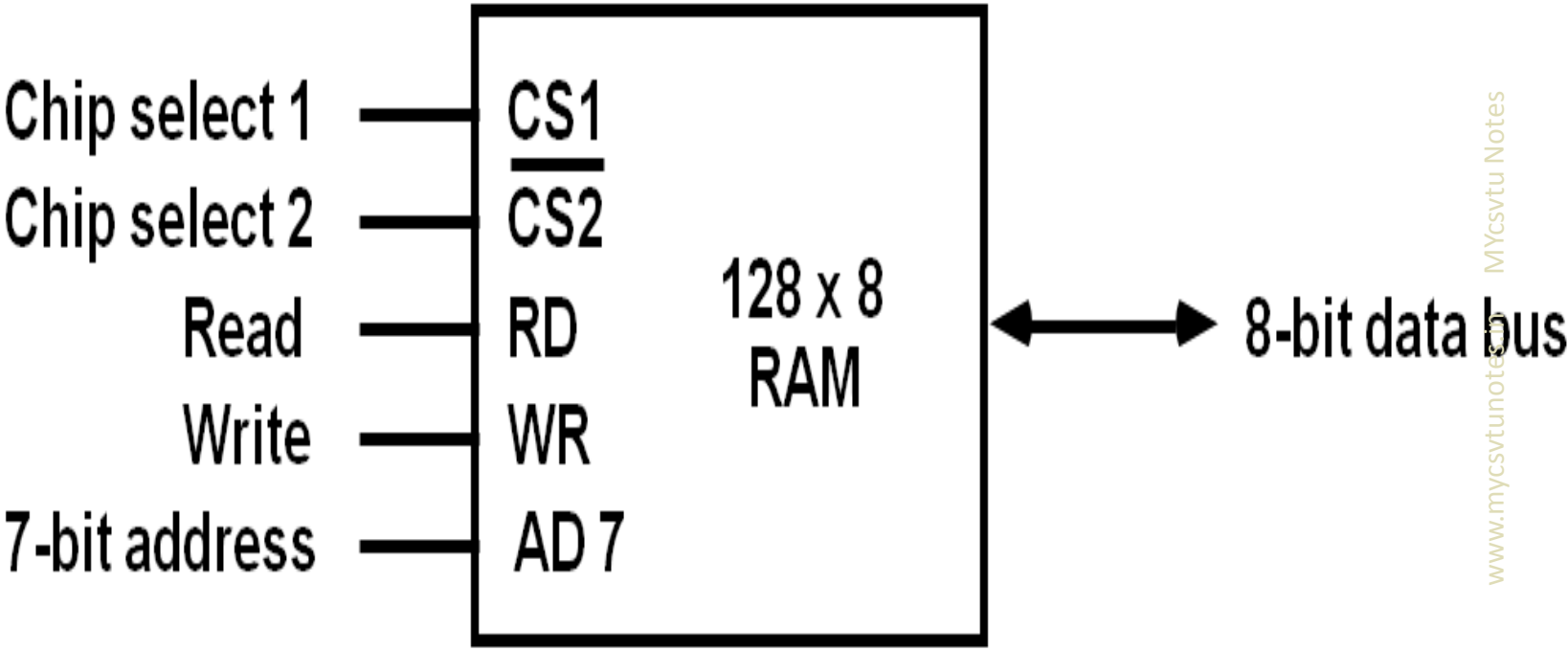
- **A computer designer must calculate the amount of memory required for any particular application and assign it to either RAM or ROM.**
- **After that the interconnection between memory and processor is established from knowledge of the size of memory needed and the type of RAM and ROM chips available.**

Deepak Bhalla (Asst. Professor (MCA & IT Dept.))

Typical ROM chip



RAM chip



- **The addressing of memory can be established by using fig. 1 that specifies the memory address assigned to all chips. This is called Memory Address Map.**

Component	Hexa 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

Fig 1 Memory Address Map for Microcomputer

- **For example, assume that a computer system needs 128 bytes of RAM and 512 bytes of ROM.**
- **Explanation of Memory Address Map Figure:**
- **The first column specifies whether RAM or ROM is used.**
- **Hexadecimal address column gives a range of hexadecimal equivalent address for each chip.**
- **Address bus lines are listed in the address bus column.**
- **In our example only 10 lines are used, thus , third column only have 10 address lines out of 16 lines.**

- **The small x's under the address bus lines designate those lines that must be connected to the address input in each chip.**
- **The RAM chip have 128 bytes and need seven address lines.**
- **The ROM has 512 bytes and need 9 address lines.**
- **x's are assigned to the low order bus lines.**
- **It is necessary for assigning the different address to each different RAM.**
- **In fig. 1 the line 8 and 9 represent the 4 different combination for RAM.**

- **The distinction between RAM and ROM address is done with the another bus lines.**
- **The line 10 is used for this purpose.**
- **When line 10 is 0, CPU selects RAM, and if line 10 is 1 then it selects to ROM.**

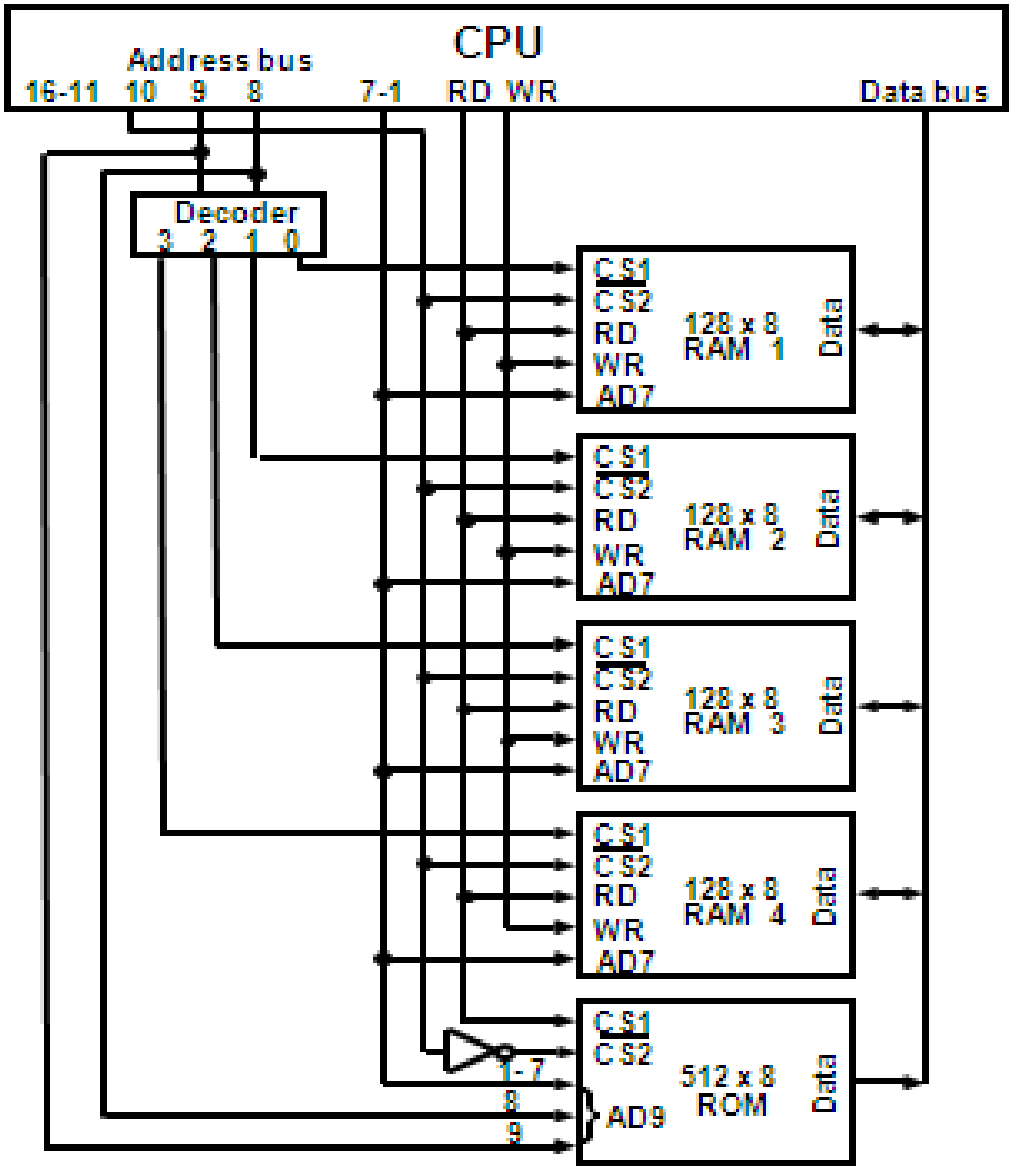
How Memory is connected to CPU

- **RAM and ROM chips are connected to the CPU through the data and address buses.**
- **The low order lines in the address bus are used to select the bytes within the chips and remaining other lines are used to select a particular chip through its chip select inputs.**
- **Figure in the next slides shows the connection of memory chips to the CPU.**
- **Here 512 bytes of RAM and 512 bytes of ROM are used.**

- **Table given in the figure 1 shows the memory map.**

Component	Hexa 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

CONNECTION OF MEMORY TO CPU



- **Each RAM receives the 7 low order bits of address bus to select one of 128 possible bytes.**
- **From line 8 and 9 in the address bus, particular RAM chip selected is determined.**
- **This is done through 2 x 4 decoder whose outputs go to the CS1 inputs in each RAM chip.**
- **When address lines 8 and 9 are equal to 00,the first 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.**

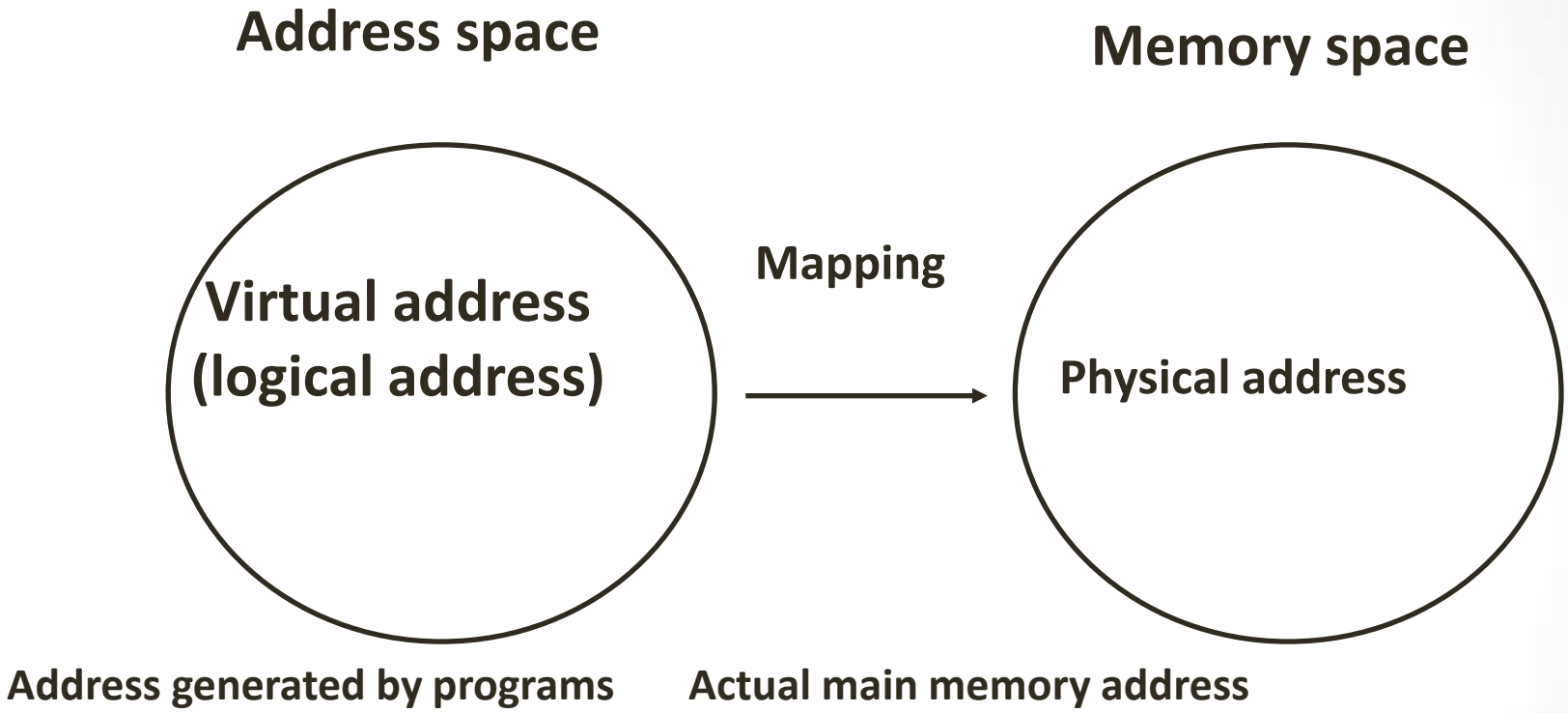
- **The line 10 is used to select between RAM and ROM chip**
- **When line 10 has bit 0, RAMs are selected and when the bit is 1, ROM is selected.**
- **The other chip select input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a READ operation.**
- **Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder.**
- **This assigns addresses 0 to 511 to RAM and 512 to 1023 to ROM.**

- **The data bus of the ROM has only an output capability.**
- **In contrast, the data bus connected to the RAM's can transfer information in both directions**

Use of virtual Memory in Computer System

- **Give the programmer the illusion that the system has a very large memory, even though the computer actually has a relatively small main memory.**
- **Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though as large memory space were available, equal to the totality of auxiliary memory.**

Address Space(Logical) and Memory Space(Physical)



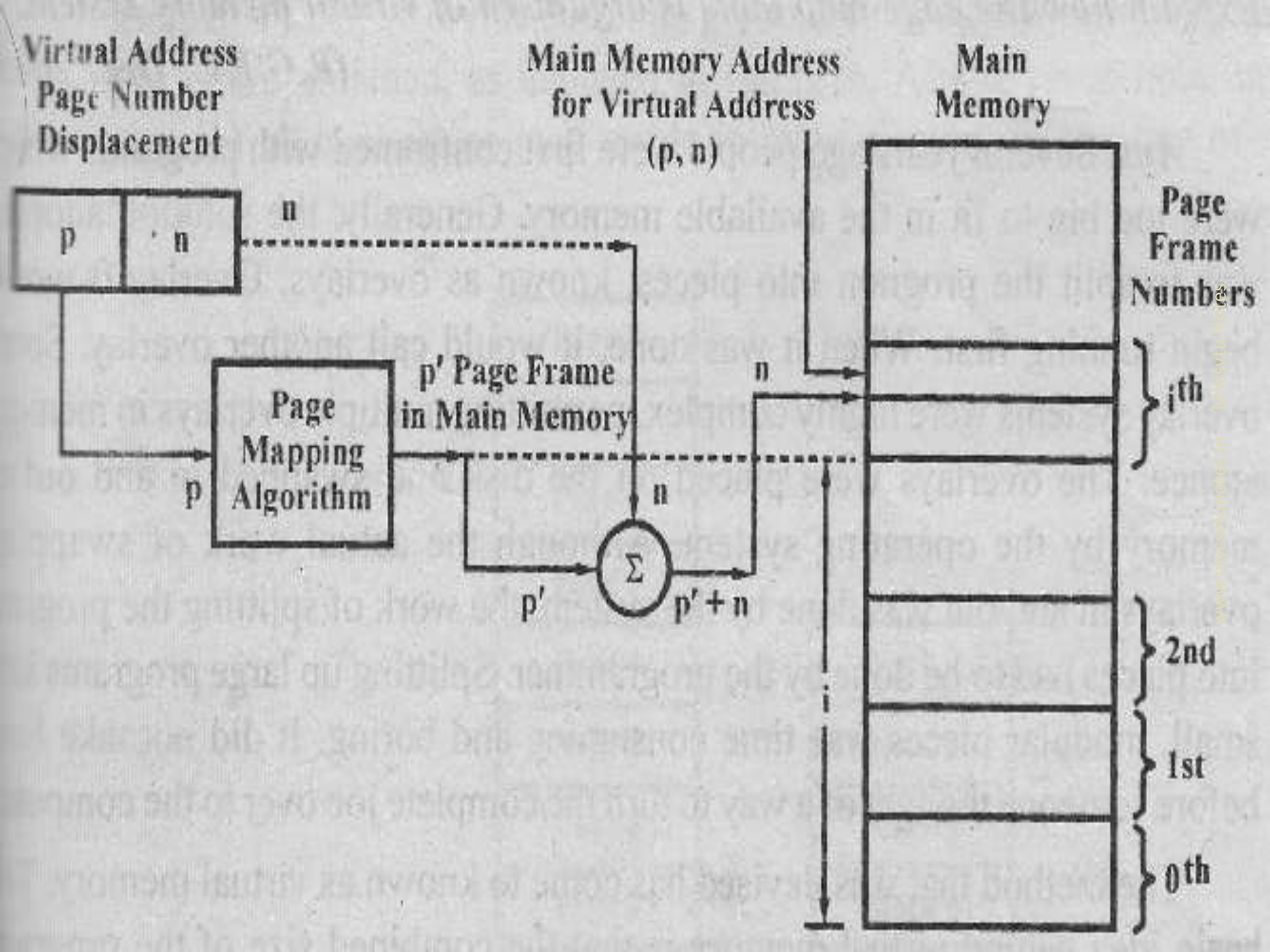
- **Traditionally virtual memory is used for following 3 purposes-**

1. **To free user programs from the need to carry out storage allocation and to allow efficient sharing of the available memory space among different users.**
2. **To make programs independent of the configuration and capacity of the physical memory present for their execution; for example, to permit seamless overflow into secondary memory when the capacity of main memory is exceeded.**

3. **To obtain the very low access time and cost per bit that are possible with a memory hierarchy.**

- **Virtual memory can be implemented as an extension of pages or segmented memory management or as a combination of both.**
- **Accordingly, address translation is performed by means of page-map tables, segment descriptor tables or both.**
- **In a paging system, the virtual address space is divided into equal-size blocks known as pages.**
- **Likewise, the physical memory is also divided into equal size blocks called frames.**

- **Size of page is same as the size of frame.**
- **Size of page may be 512, 1024 or 2048 words.**
- **Each virtual address may be regarded as an ordered pair <p,n> in a virtual system, where p is the page number and n is the word number within the page p.**
- **Some times the quantity n is referred as to as the displacement or offset.**
- **A user program may be regarded as a sequence of pages and a complete copy of the program is always held in a backup store such as a drum or a disk.**


Virtual Address
Page Number
Displacement
Main Memory Address
for Virtual Address
(p, n)
Main
Memory
p
n
n
Page
Frame
Numbers
Page
Mapping
Algorithm
p
p' Page Frame
in Main Memory
n
n
Σ
p'
p' + n
i
i th
2nd
1st
0 th

- **A page p of the user program can be placed in any available page frame p' of the main memory.**
- **If the page is in main memory, a program may access the page.**
- **In a paging scheme, pages are brought from secondary memory and are stored in main memory in a dynamic manner.**
- **All virtual addresses produced by a user program must be translated into physical memory addresses.**
- **The above process is called dynamic address translation and is depicted in the fig 1.**

- **When a running program accesses a virtual memory location v=<p,n>, the mapping algorithm finds that the virtual page p is mapped to the physical page p'.**
- **Then physical address is determined by appending p' to n.**
- **This dynamic address translator can be implemented using a page table.**
- **This table is maintained in the main memory in most systems.**

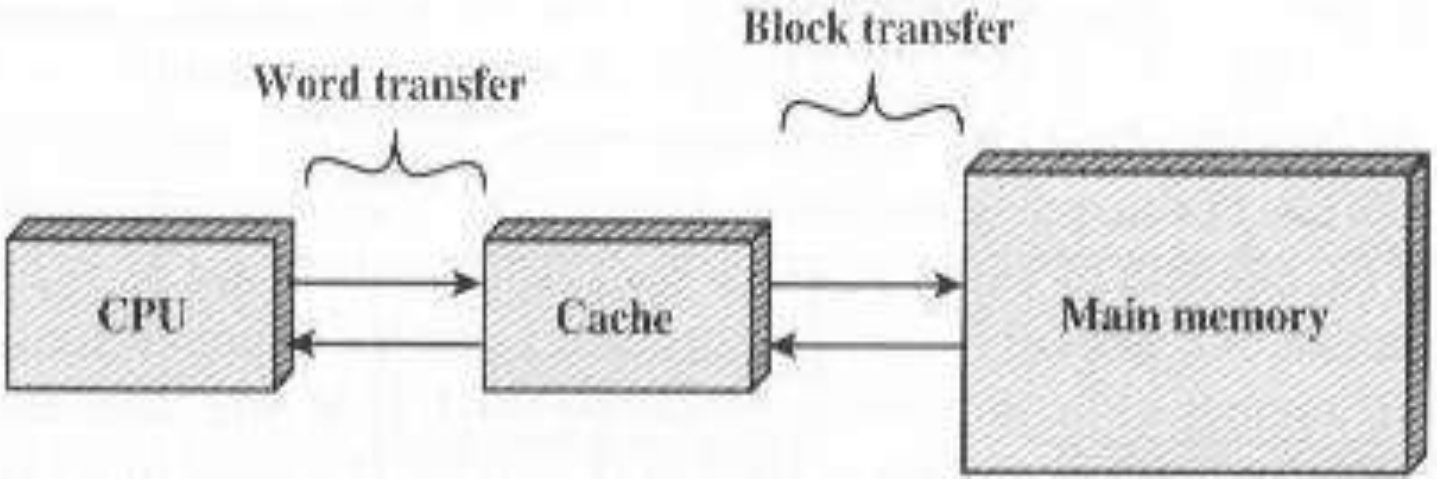
Cache Memory

Ans. A very high speed memory called a cache which is sometimes used to increase the processing speed by making current programs and data available to the CPU at a rapid rate. It is small but fast memory is installed to keep the most frequently needed information, and the CPU is instructed to access this fast memory, which is opposed to the main memory. Thus, the efficiency of program execution can then be significantly improved by using ***cache memory***. CPU logic 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 a extremely fast, small cache between CPU and main memory, which has a access time close to processor logic clock cycle time.

The basic idea of the cache organization is that by keeping the frequently accessed instruction and data in the fast cache memory. Although cache memory is the small fraction of memory, a large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

Fig. 5.12 shows the memory organization of computer system with cache memory.

Cache Memory



Deepak Bhalla (Asst. Professor (MCA & IT Dept.))

The fundamental operation of cache is as follows. When the CPU needs to access memory, it first examines in the cache. If the word is found in the cache, it is read from cache memory. If the word is not found in cache, the main memory is accessed to read the word. Also, a block of words containing the one just accessed is then copied to the cache memory. The block size may vary from one word to about 16 words adjacent to the one just accessed. Thus, the future references to memory found the required words in the fast cache memory.

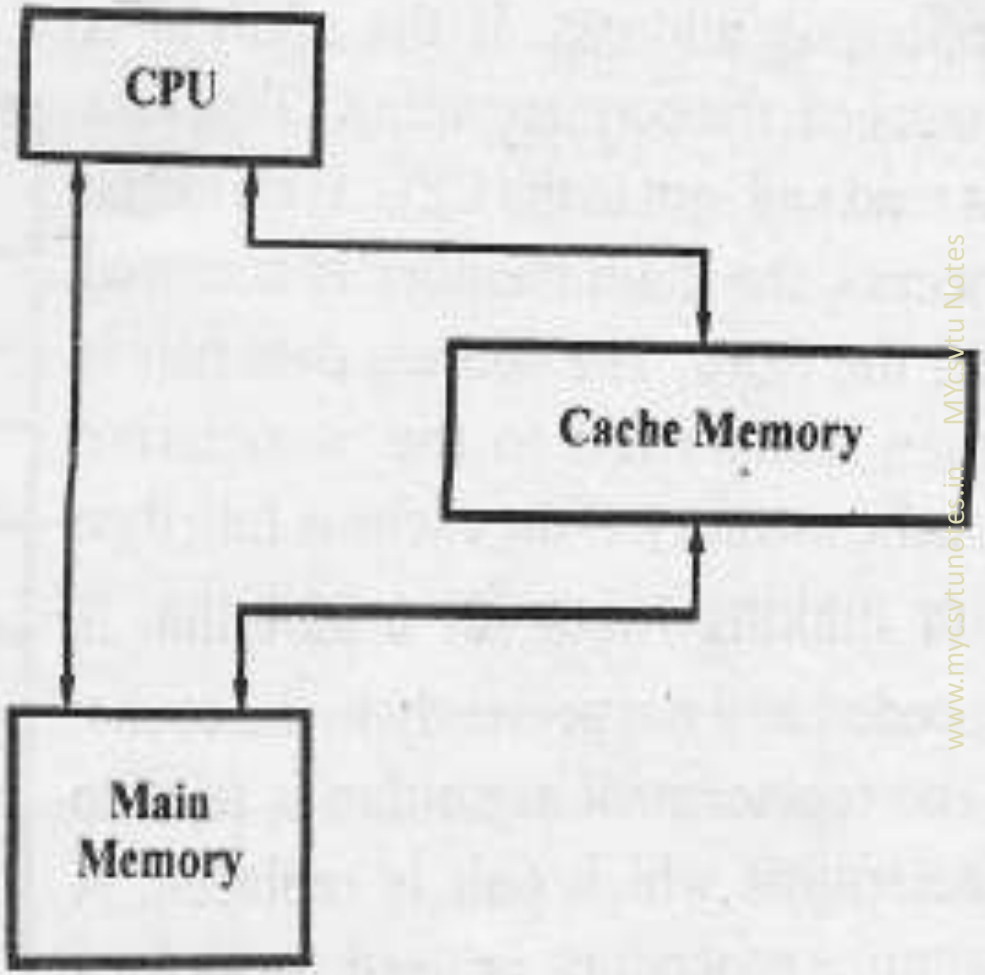
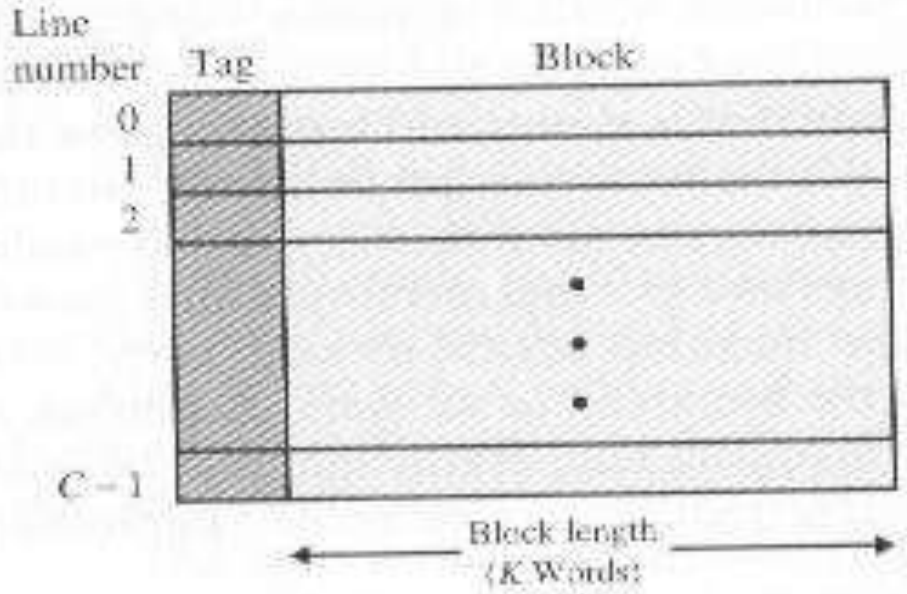
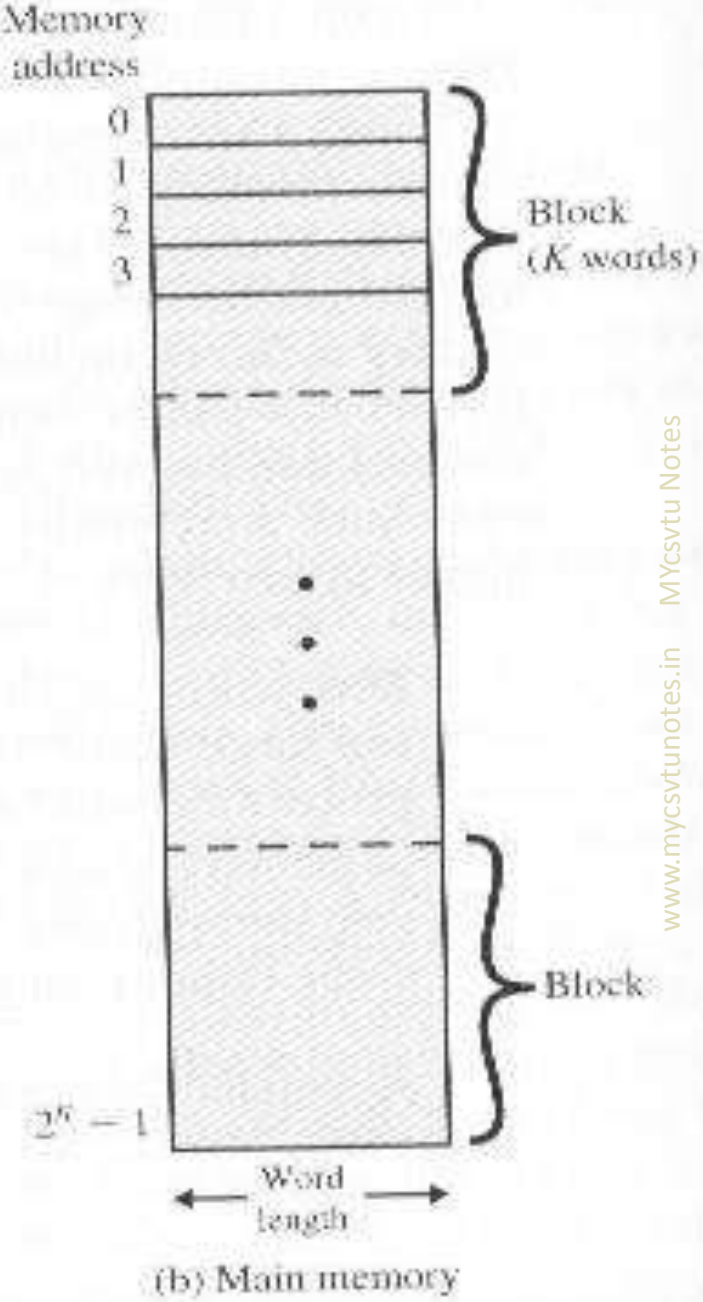


Fig. 5.12 Memory Organization with Cache Memory

When the CPU refers to memory and finds the word in cache, it is said to produce *hit*. If the word is not found in cache, it counts as *miss*. The performance of cache memory is frequently measured in terms of a quantity called *hit ratio*. Hit ratio is the ratio of the numbers of hits divided by the total CPU references to memory (hits + misses). Hit ratios of 0.9 and higher have been reported. This high ratio verifies the validity of the locality of reference property.



(a) Cache



(b) Main memory

Figure 4.4 Cache/Main Memory Structure

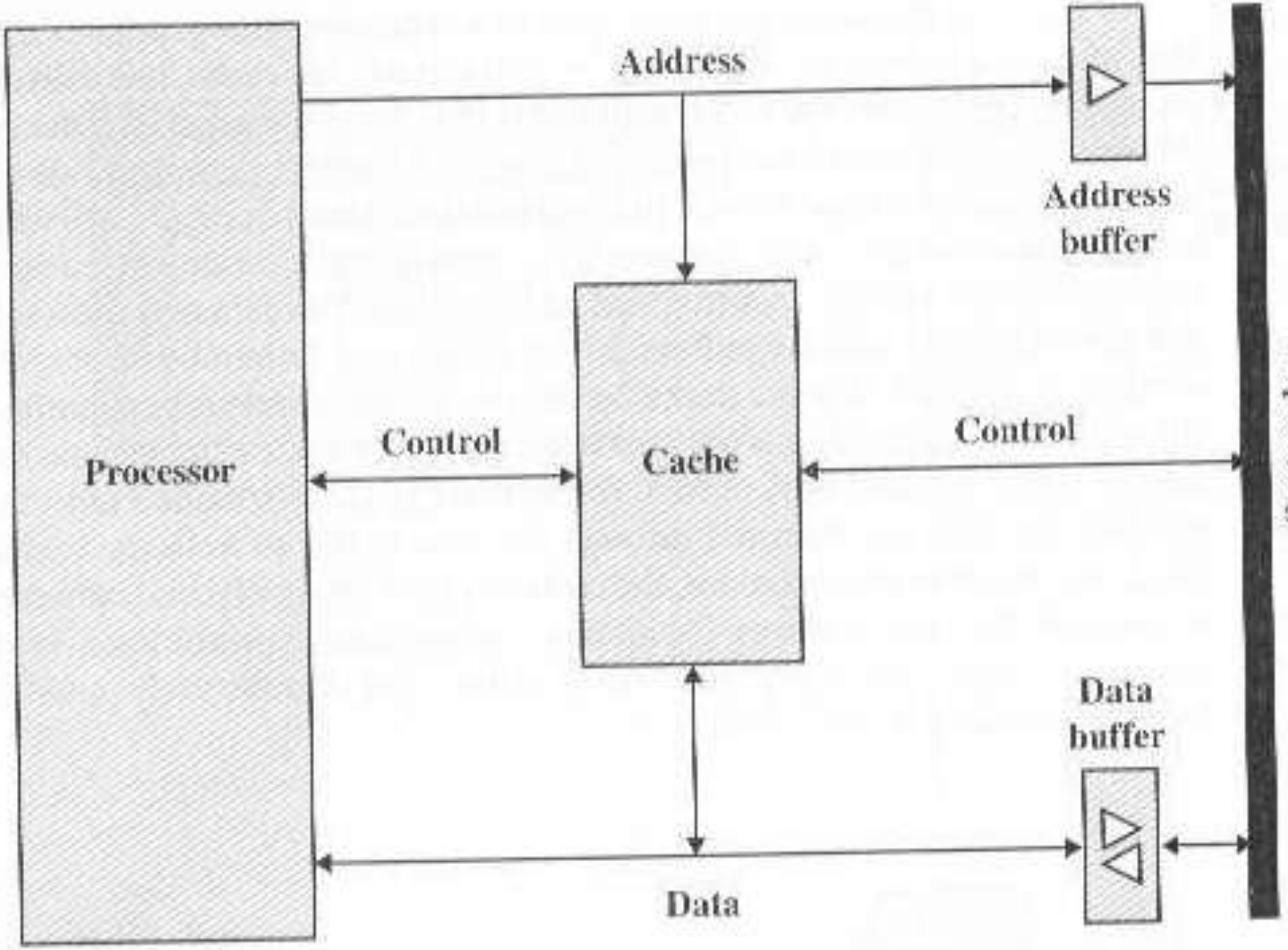


Figure 4.6 Typical Cache Organization

Various Techniques for Mapping Data from the Main Memory

- The characteristics of cache memory is the fast access time.
- Therefore, a very little time is waster in searching of any words in cache.
- The transformation of data from one main memory to cache memory is referred to as a mapping process.
- There are 3 types of memory procedures in cache memory organization

- Associative Mapping
- Direct Mapping
- Set Associative Mapping

Associative Memory

(i) Associative Mapping – The cache organization for associative mapping is illustrated in fig. 5.13. The associative memory stores both the address and content of memory word. This permits any location in cache to store any word from main memory. Fig. 5.13 shows that the three word 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 15 bit CPU address is placed in the argument register and the associative memory is searched for a matching address. If the address is matched, 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, then for making room for a pair that is needed and not presently in the cache. The replacement algorithm is used to determine which pair is replaced. A simple procedure is used to replace cells of the cache in round-robin order whenever a new word is requested from main memory.

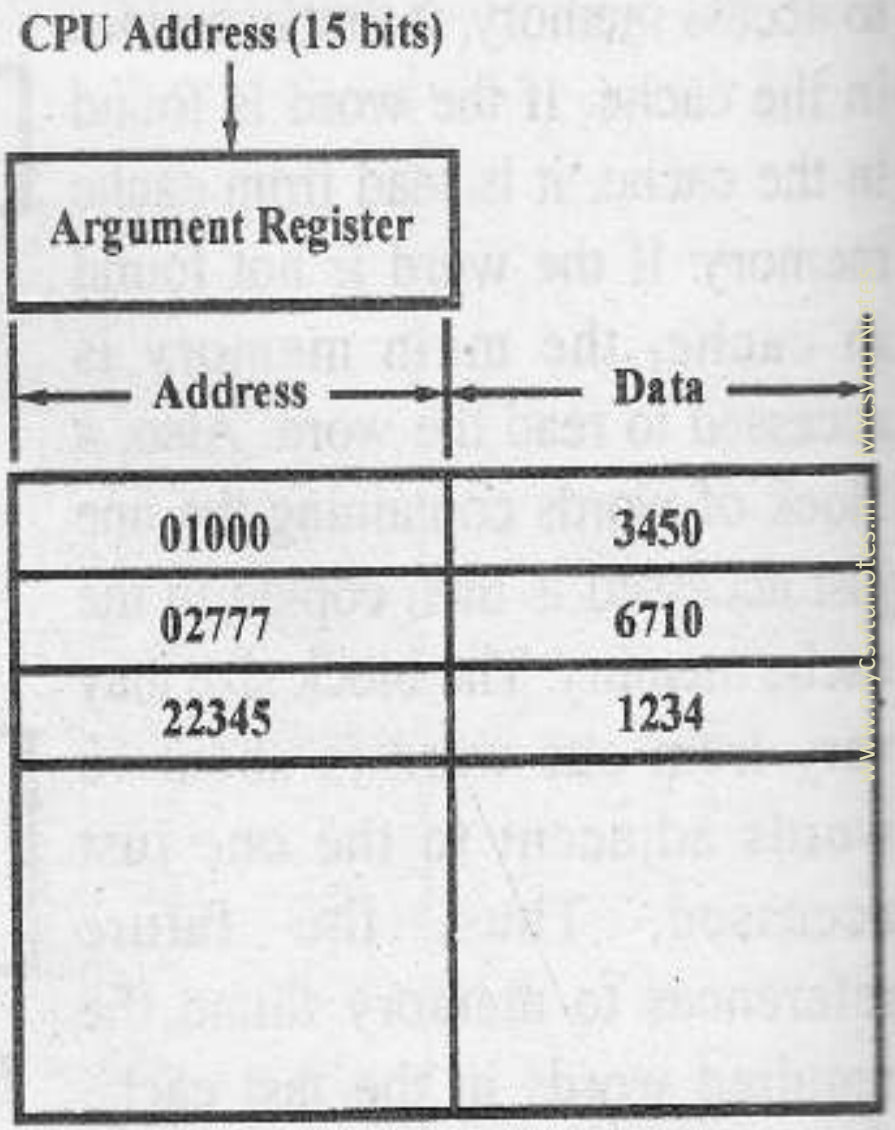


Fig. 5.13 Associative Mapping Cache

Direct Mapping

(ii) Direct Mapping – Associative memories are expensive compared to random-access memories because of the added logic associated with each cell. Fig. 5.14 illustrates the possibility of using random-access memory for the cache. A 15 bit CPU address is divided into two fields. The six most significant bits constitute the tag field and remaining 9 bits form a index field. Main memory needs an address that includes both the tag and index bits. The number of bits in the index field is equal to the number of address bit required to access the cache memory.

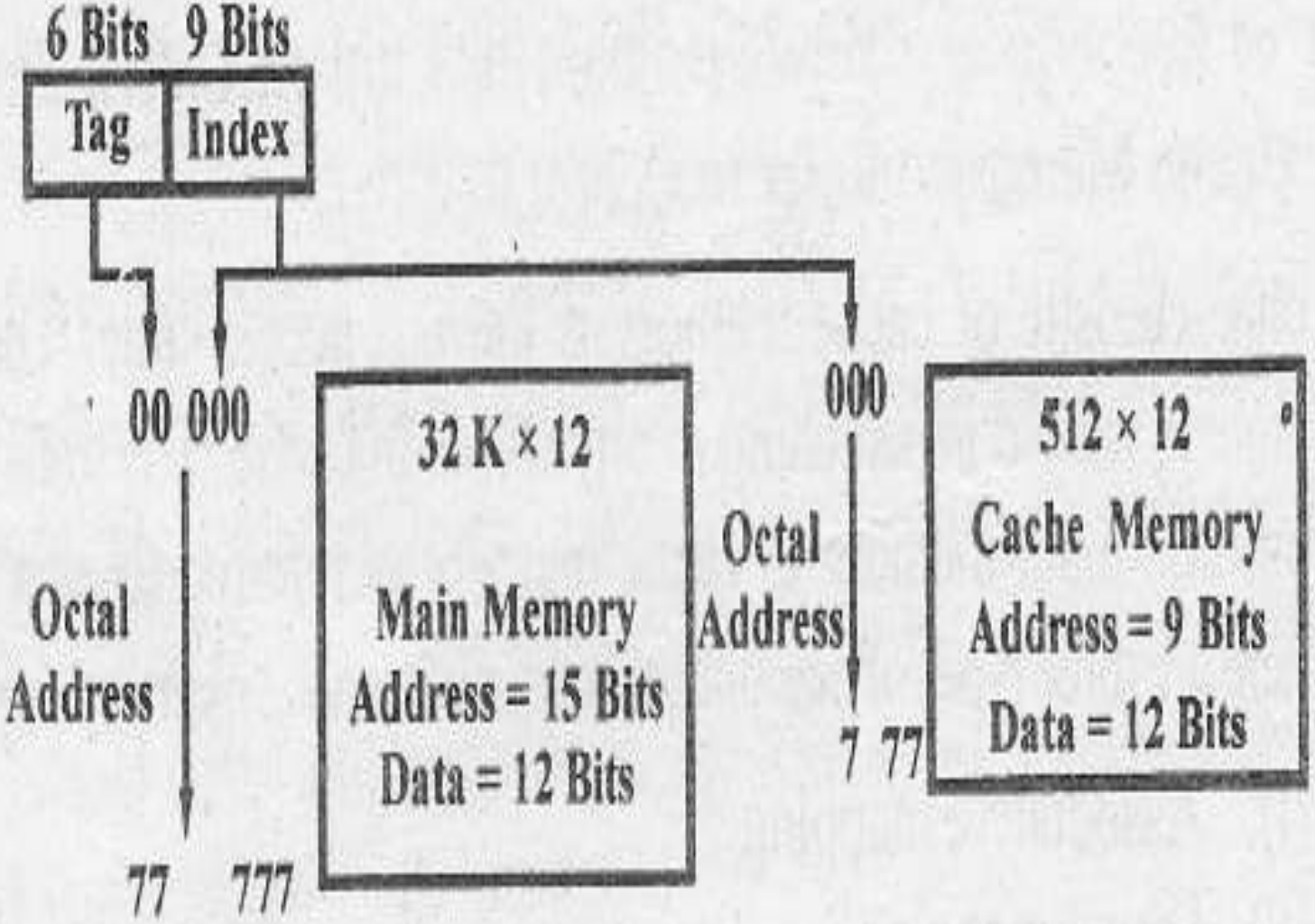
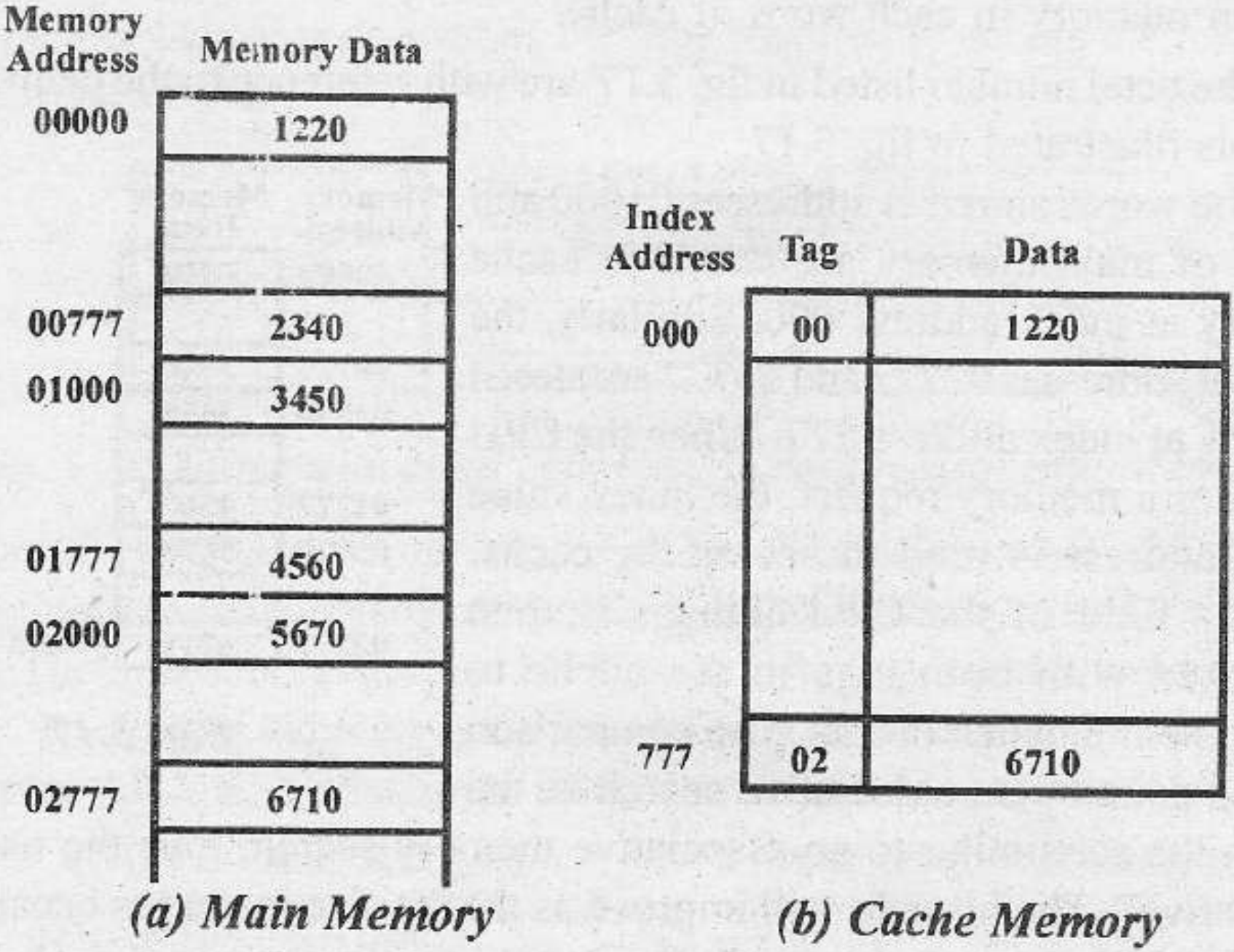


Fig. 5.14 Addressing Relationships Between Main and Cache Memories

Generally, there are 2^K words in cache memory and 2^n words in main memory. The n-bit memory address is divided into two fields – k bits for the index field and n – k bits for the tag field. This organization uses the n-bit address to access the main memory and the k-bit index to access the cache. 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. 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 not, there is a miss and required data is in the main memory. Fig. 5.15 shows direct mapping cache organisation.



(a) Main Memory *(b) Cache Memory*

Fig. 5.15 Direct Mapping Cache Organization

(iii) Set-Associative Mapping – The disadvantage of direct mapping is that two words with the same index in their address 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 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. Fig. 5.16 shows an example of a set associative cache organization for a set size of two.

Each index address refers to two data words and their associated tags.

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

Fig. 5.16 Two-way Set-associative Mapping Cache

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 stores 1024 words of main memory since each word of cache contains two data words.

Generally, a set-associative cache of set size K will accommodate K words of main memory in each word of cache.

The octal number listed in fig. 5.17 are with reference to the main memory contents illustrated in fig. 5.17.

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 the CPU generates a memory request, the index value of the address is used to access the cache. The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs. The comparison logic is done by an associative search as the tags in the set similar to an associative memory search, thus the name "set-associative". The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache. However an increase in the set-size increases the number of bits in words of cache and require more complex comparison logic.

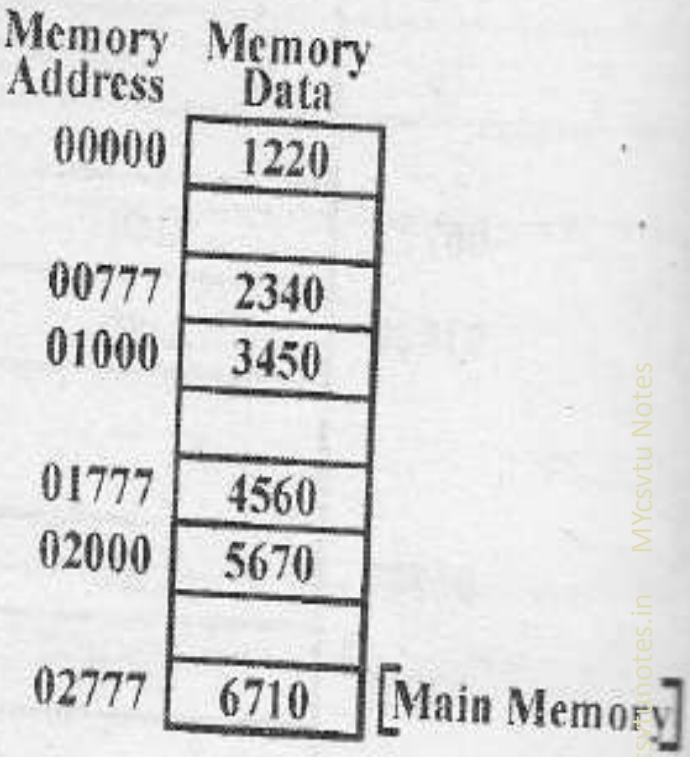


Fig. 5.17

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). With the random replacement policy the control chooses one tag-data item for replacement at random. The FIFO procedure selects for replacement the item that has been in the set the longest. The LRU algorithm selects for replacement the item that has been least recently used by CPU. Both FIFO and LRU can be implemented by adding a few extra bits in each word of cache.