SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

**SCHOOL OF ELECTRICAL AND ELECTRONICS**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# UNIT – III – Memory Organization – SCSA1402

# UNIT III Memory Organization

Memory Hierarchy - Main memory - auxiliary Memory - Associative Memory –Cache Memory - Virtual memory

## Memory Hierarchy

The memory unit is an essential component in any digital computer since It Is needed for storing programs and data. The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and other backup information. 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. Figure 1 illustrates the components in a typical memory hierarchy

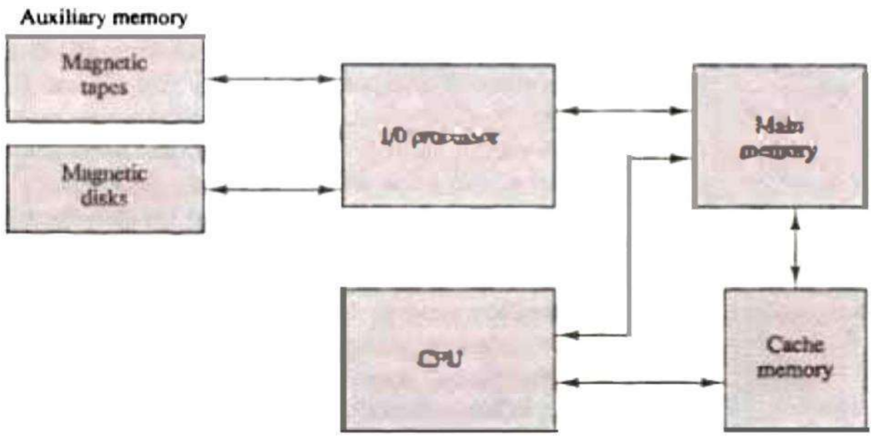


**Fig.1 Memory hierarchy in a typical Computer system**

## Main memory

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 semi conductor integrated circuits. Integrated circuits RAM chips are available in two possible operating modes, static and dynamic.

• Static RAM – Consists of internal flip flops that store the binary information.

• Dynamic RAM – Stores the binary information in the form of electric charges that are applied to capacitors.

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.

• Read Only Memory –Store 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.

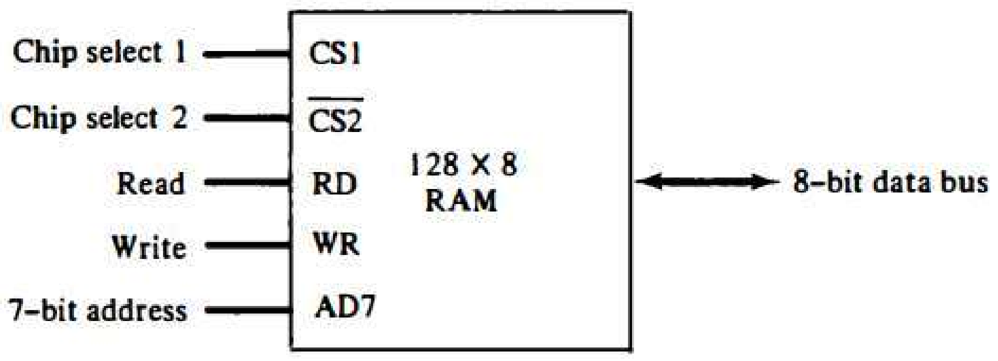
The ROM portion of main memory is needed for storing an initial program called a Bootstrap loader.

• Boot strap loader –function is start the computer software operating when power is turned on.

• Boot strap program loads a portion of operating system from disc to main memory and control is then transferred to operating system.

RAM and ROM CHIP

• RAM chip –utilizes bidirectional data bus with three state buffers to perform communication with CPU



(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

**Fig.2: Block Diagram of a RAM Chip**

The block diagram of a RAM Chip is shown in Fig.2. The capacity of memory is 128 words of eight bits (one byte) 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 enabling the chip only when it is selected by the microprocessor. The read and write inputs are sometimes combined into one line labelled R/W. The function table listed in Fig.12-2(b) specifies the operation of the RAM chip. 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 its 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.

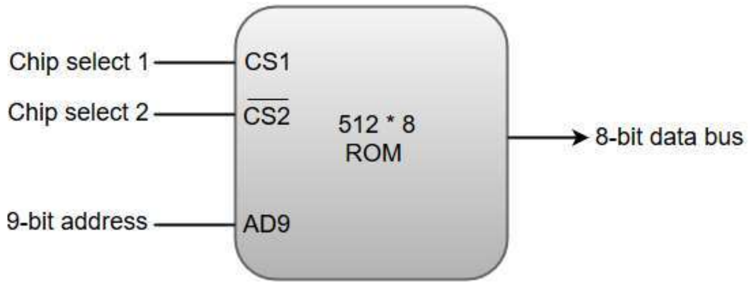


**Fig.3: Typical ROM Chip**

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. The block diagram of a ROM chip is shown in fig.3. The nine address lines in the ROM chip specify any one of the512 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.

## Memory Address Map

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 specify the memory address assigned to each chip. The table called Memory address map, is a pictorial representation of assigned address space for each chip in the system.

**Table 1: Memory address map for microcomputer**

| 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 |

The memory address map for this configuration is shown in table. The component column specifies whether a RAM or a ROM chip is used. The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip. The address bus lines are listed in the third column. The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines.

**Memory Connection to CPU**

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.

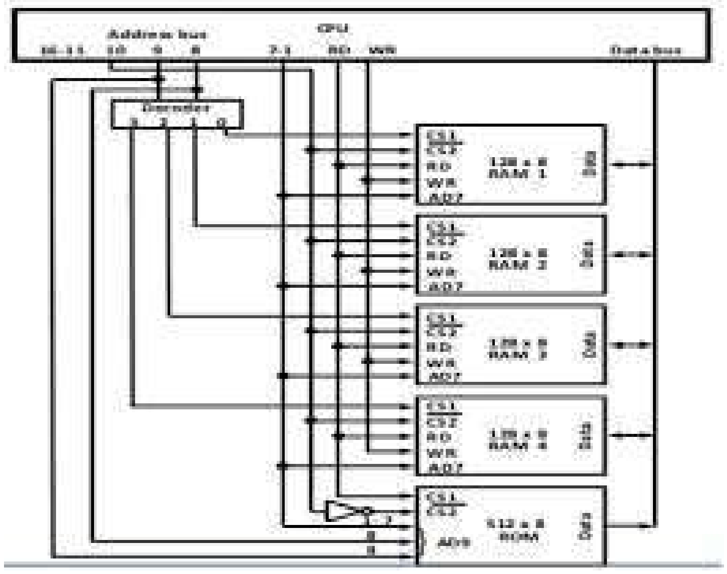


**Fig.4: Memory connection to the CPU**

The connection of memory chips to the CPU is shown in the above figure. This configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM. Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus. This is done through a 2 X 4 decoder whose outputs go to the CS1 inputs in each RAM chip. Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is select, and so on. The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip. The selection between RAM and ROM is achieved through bus line 10. The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1. Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder. The data bus of the ROM has only an output capability, whereas the data bus connected to the RAMs can transfer information in both directions.

**Auxiliary 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).

• CAM is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location

• Associative memory is more expensive than a RAM because each cell must have storage capability as well as logic circuits

• Argument register –holds an external argument for content matching

• Key register –mask for choosing a particular field or key in the argument word

**Hardware Organization**

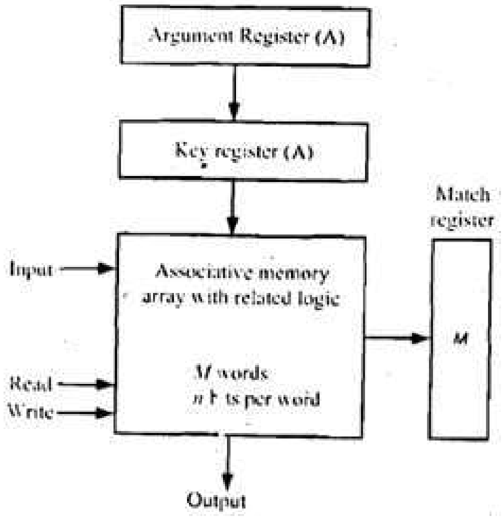


**Fig.5: Block Diagram of associative memory**

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 b 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

**Read and Write operation 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

• In read operation all matched words are read inλ sequence by applying a read signal to each word line whose corresponding Mi bit is a logic 1

• In applications where no two identical items are stored in the memory , only one word may match , in which case we can use Mi output directly as a read signal for the corresponding word

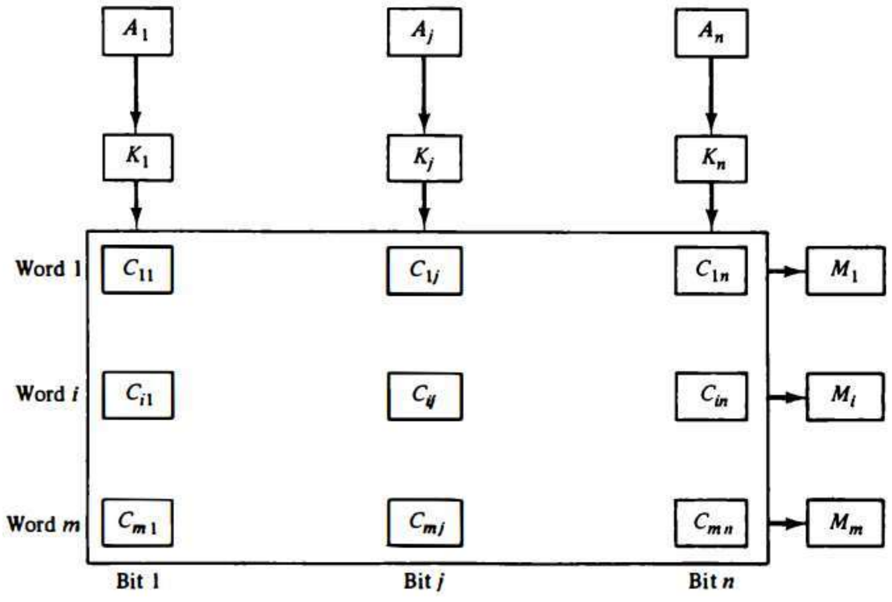


**Fig.6:Associative memory of m word, n cells per word**

The relation between the memory array and external registers in an associative memory is shown in Fig.6. The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and second specifies the bit position in the word. Thus cell Cij is the cell for bit j in word i. A bit Aj in the argument register is compared with all the bits in column j of the array provided that kj =1.This is done for all columns j=1,2,….n. If a match occurs between all the unmasked bits of the argument and the bits in word I, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.

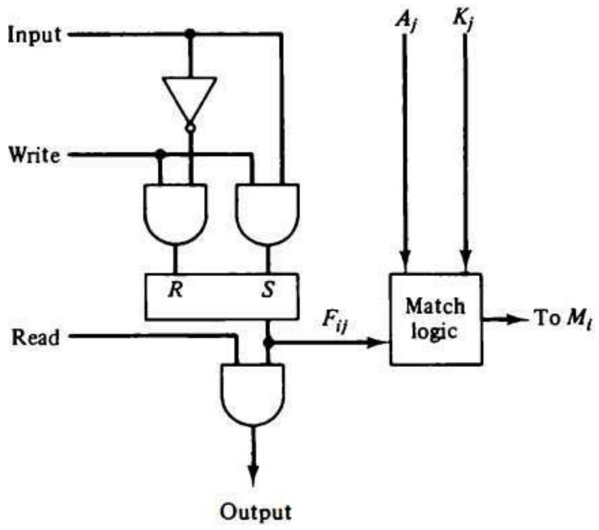


**Fig.7: One cell of associative memory**

It consists of flip-flop storage element Fij and the circuits for reading, writing, and matching the cell. The input bit is transferred into the storage cell during a write operation. The bit stored is read out during a read operation. The match logic compares the content of the storage cell with corresponding unmasked bit of the argument and provides an output for the decision logic that sets the bit in Mi.

**Read and Write operation**

**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

• In read operation all matched words are read in λ sequence by applying a read signal to each word line whose corresponding Mi bit is a logic 1

• In applications where no two identical items are stored in the memory , only one word may match , in which case we can use Mi output directly as a read signal for the corresponding word

**Write Operation**

Can take two different forms

1. Entire memory may be loaded with new information

2.Unwanted words to be deleted and new words to be inserted

1.Entire memory : writing can be done by addressing each location in sequence – This makes it random access memory for writing and content addressable memory for reading – number of lines needed for decoding is d Where m = 2 d , m is number of words.

2.Unwanted words to be deleted and newλ words to be inserted :

• Tag register is used which has as many bits as there are words in memory

• For every active ( valid ) word in memory , the corresponding bit in tag register is set to 1

• When word is deleted the corresponding tag bit is reset to 0

• The word is stored in the memory by scanning the tag register until the first 0 bit is encountered After storing the word the bit is set to 1.

**Cache Memory**

Effectiveness of cache mechanism is based on a property of computer programs called "locality of reference"

• The references to memory at any given time interval tend to beλ confined within a localized areas

• Analysis of programs shows that most of their execution time is spent on routines in which instructions are executed repeatedly These instructions may be – loops, nested loops , or few procedures that call each other

• Many instructions in localized areas of program are executed

• Repeatedly during some time period and reminder of the program is accessed infrequently

This property is called "Locality of Reference".

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 dose to processor logic dock 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.

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. 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. 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

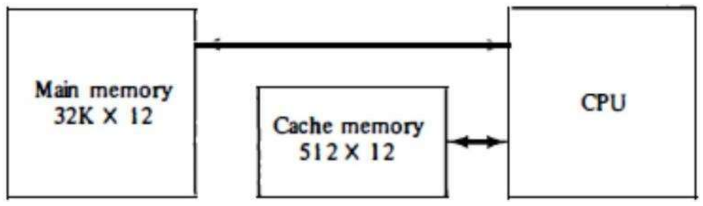


**Fig.8:Example of cache memory**

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 , 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.

• When a read request is received from CPU, contents of a block of memory words containing the location specified are transferred in to cache

• When the program references any of the locations in this block , the contents are read from the cache Number of blocks in cache is smaller than number of blocks in main memory

• Correspondence between main memory blocks and those in the cache is specified by a mapping function

• Assume cache is full and memory word not in cache is referenced

• Control hardware decides which block from cache is to be removed to create space for new block containing referenced word from memory

• Collection of rules for making this decision is called "Replacement algorithm "

**Read/ Write operations on cache**

**• Cache Hit Operation**

• CPU issues Read/Write requests using addresses thatλ refer to locations in main memory

• Cache control circuitry determines whether requested word currently exists in cache

• If it does, Read/Write operation is performed on the appropriate location in cache (Read/Write Hit )

**Read/Write operations on cache in case of Hit**

• In Read operation main memory is not involved.

• In Write operation two things can happen.

1.Cache and main memory locations are updated λ simultaneously (" Write Through ") OR

2. Update only cache location and mark it as " Dirty or λ Modified Bit " and update main memory location at the time of cache block removal (" Write Back " or " Copy Back ") .

**Read/Write operations on cache in case of Miss Read Operation**

• When addressed word is not in cache Read Miss occurs there are two ways this can be dealt with

1.Entire block of words that contain the requested word is copied from main memory to cache and the particular word requested is forwarded to CPU from the cache ( Load Through ) (OR)

2.The requested word from memory is sent to CPU first and then the cache is updated ( Early Restart )

**Write Operation**

• If addressed word is not in cache Write Miss occurs

• If write through protocol is used information is directly written in to main memory

• In write back protocol , block containing the word is first brought in to cache , the desired word is then overwritten.

**Mapping Functions**

• Correspondence between main memory blocks and those in the cache is specified by a memory mapping function

• There are three techniques in memory mapping

1. Direct Mapping

2. Associative Mapping

3. Set Associative Mapping

**Direct mapping:**

A particular block of main memory can be brought to a particular block of cache memory. So, it is not flexible.

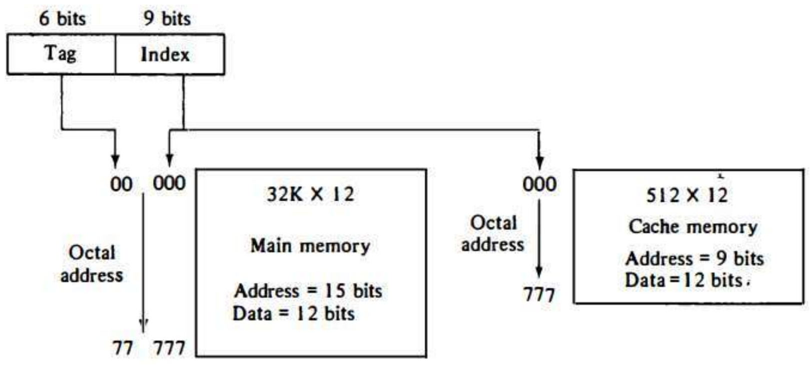


**Fig.9:Addressing relationship between main and cache memories**

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

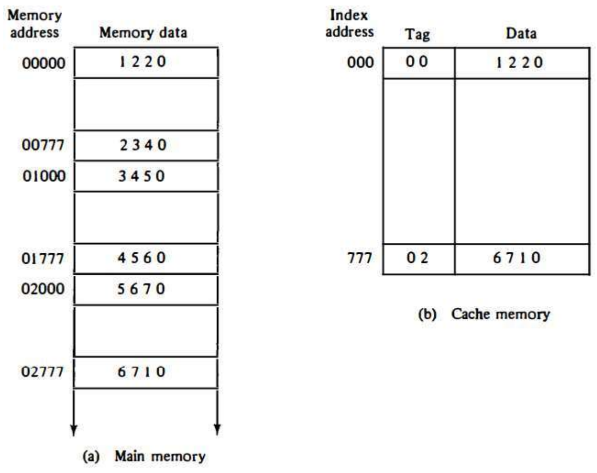


**Fig.10:Direct mapping cache organization**

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 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 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 there is no match, there is a miss and the required word is read from main memory.

**Associative mapping**

In this mapping function, any block of Main memory can potentially reside in any cache block position. This is much more flexible mapping method.

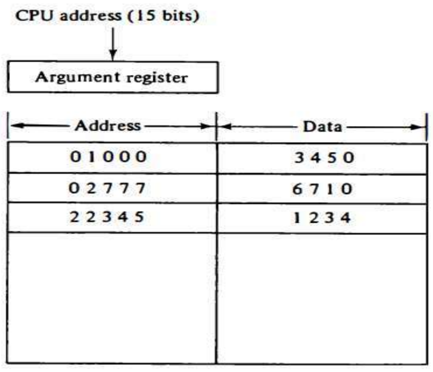


**Fig.11:Associative mapping cache (all numbers in octal)**

The associative memory stores both 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 ctal 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 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.

**Set-associative mapping**

In this method, blocks of cache are grouped into sets, and the mapping allows a block of main memory to reside in any block of a specific set. From the flexibility point of view, it is in between to the other two methods.

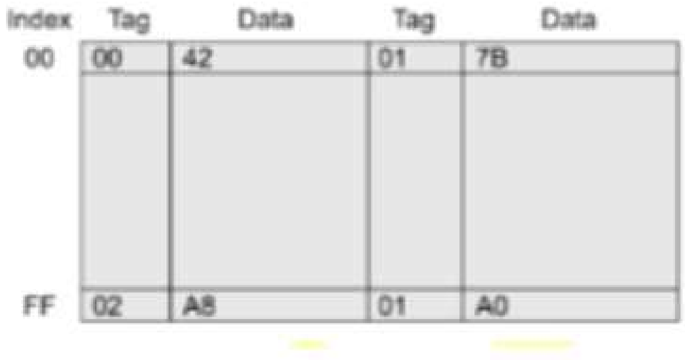


**Fig.12:2 way set associative cache memory**

The octal numbers listed in Fig.12-15 are with reference to the main memory contents. When the CPU generates a memory request, the index values 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 dine by an associative search of the tags in the set similar to an associative memory search thus the name “Set Associative″

**Replacement Policies**

• When the cache is full and there is necessity to bring new data to cache , then a decision must be made as to which data from cache is to be removed

• The guideline for taking a decision about which data is to be removed is called replacement policy Replacement policy depends on mapping

• There is no specific policy in case of Direct mapping as we have no choice of block placement in cache Replacement Policies In case of associative mapping

• A simple procedure is to replace cells of the cache in round robin order whenever a new word is requested from memory

• This constitutes a First-in First-out (FIFO) replacement policy

**In case of set associative mapping**

• Random replacement

• First-in First-out (FIFO) ( item chosen is the item that has been in the set longest)

• Least Recently Used (LRU) ( item chosen is the item λ that has been least recently used by CPU)

**Virtual Memory**

Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program generated addresses are translated automatically to the corresponding machine addresses. The size of virtual storage is limited by the addressing scheme of the computer system and amount of secondary memory is available not by the actual number of the main storage locations.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.

A process may be broken into number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time address translation and use of page or segment table permits this

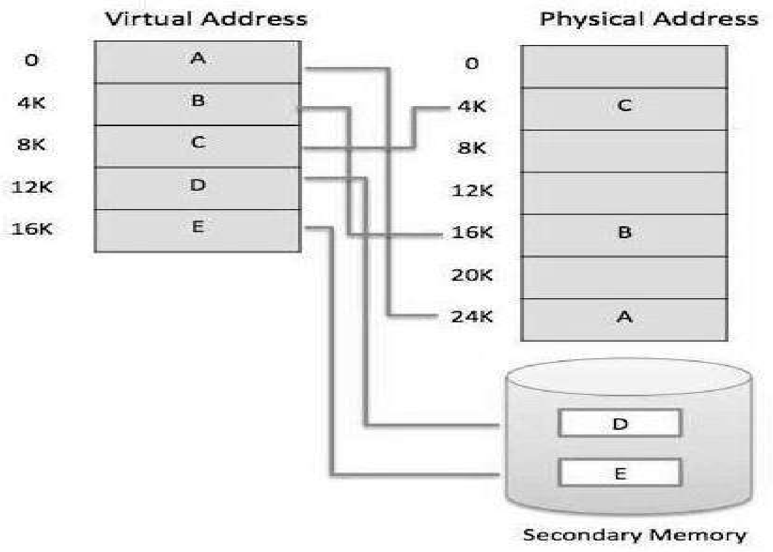


**Fig.13: virtual memory**

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory