Memory Organization

Contents

- **9.1 Introduction:** Memory Hierarchy, Main Memory, RAM and ROM Chips, Memory address Map, Memory Connection to CPU, Auxiliary Memory (*Review of* magnetic Disk, Magnetic Tape)
- **9.2 Associative Memory**: Hardware Organization, Match Logic, Read Operation, Write Operation
- **9.3 Cache Memory**: Locality of Reference, Hit & Miss Ratio, Mapping, Write Policies

Memory hierarchy

Memory hierarchy in a computer system :

- 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.
  - **Main Memory**: memory unit that communicates directly with the CPU (RAM)
  - **Auxiliary Memory**: device that provide backup storage (Disk Drives)
  - **Cache Memory**: special very-high-speed memory to increase the processing speed (Cache RAM

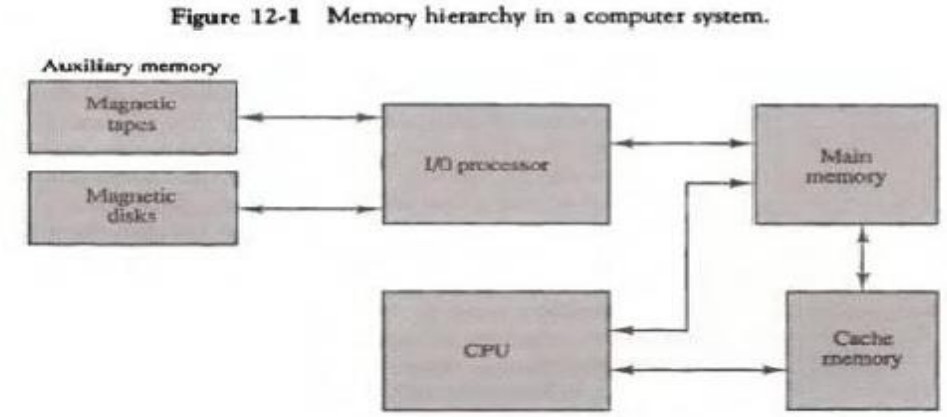


Figure 12-1 Memory hierarchy in a computer system.

- Figure illustrates the components in a typical memory hierarchy.
- At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files.
- The Magnetic disks used as backup storage.
- The main memory occupies a central position by being able to communicate directly with CPU and with auxiliary memory devices through an I/O process.
- Program not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.
- The cache memory is used for storing segments of programs currently being executed in the CPU. The I/O processor manages data transfer between auxiliary memory and main memory. 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 expensive, and has very high access speed. The CPU has direct access to both cache and main memory but not to auxiliary memory.

- **Multiprogramming:** Many operating systems are designed to enable the CPU to process a number of independent programs concurrently.
- Multiprogramming refers to the existence of 2 or more programs in different parts of the memory hierarchy at the same time.
- **Memory management System:** The part of the computer system that supervises the flow of information between auxiliary memory and main memory.

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.

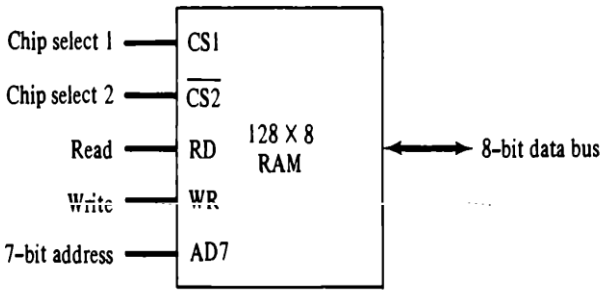
- Boot strap loader –function is start the computer software operating when power is turned on.
- Boot strap program loads a portion of operating system from disc to main memory and control is then transferred to operating system.

RAM and ROM CHIP

- RAM chip –utilizes bidirectional data bus with three state buffers to perform communication with CPUs

- Figure (a). 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.

Figure 12-2 Typical RAM chip.



(a) Block diagram

CS1	CS2	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

(b) Function table

- The function table listed in Figure (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.

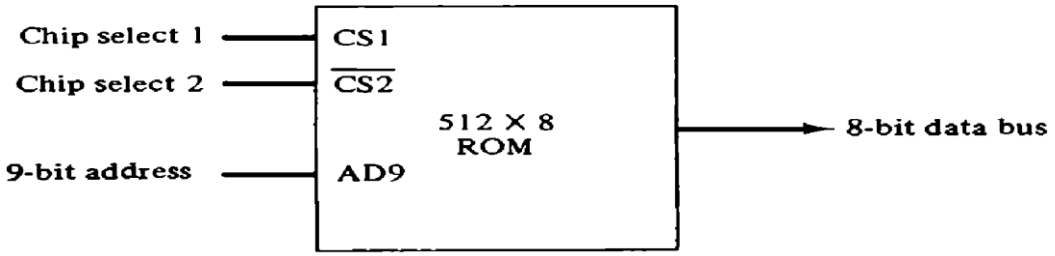


Figure 12-3 Typical ROM chip.

- 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.12-3. 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 bus data 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 12-1 Memory Address Map for Microprocomputer

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 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

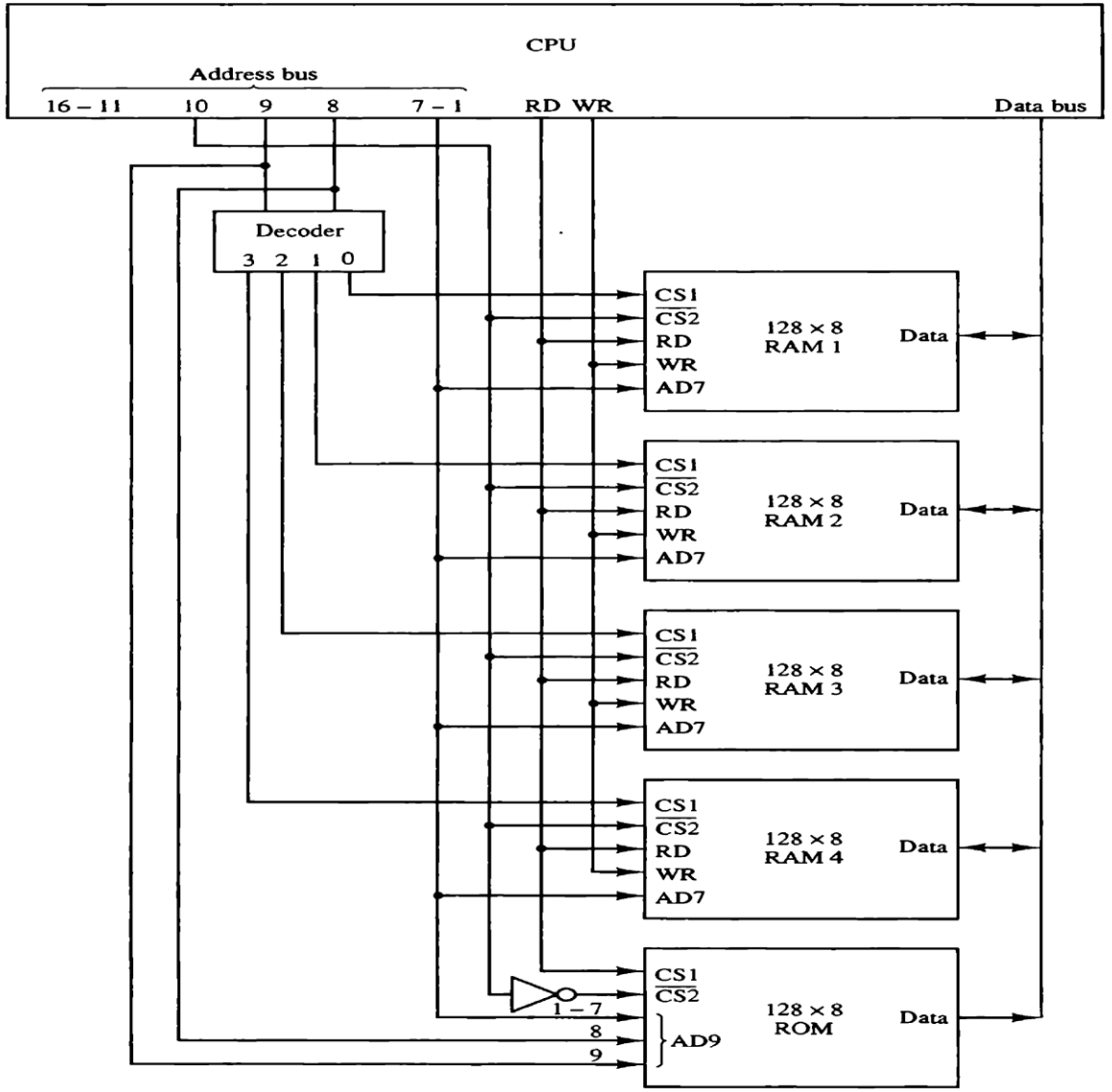


Figure 12-4 Memory connection to the CPU.

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

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

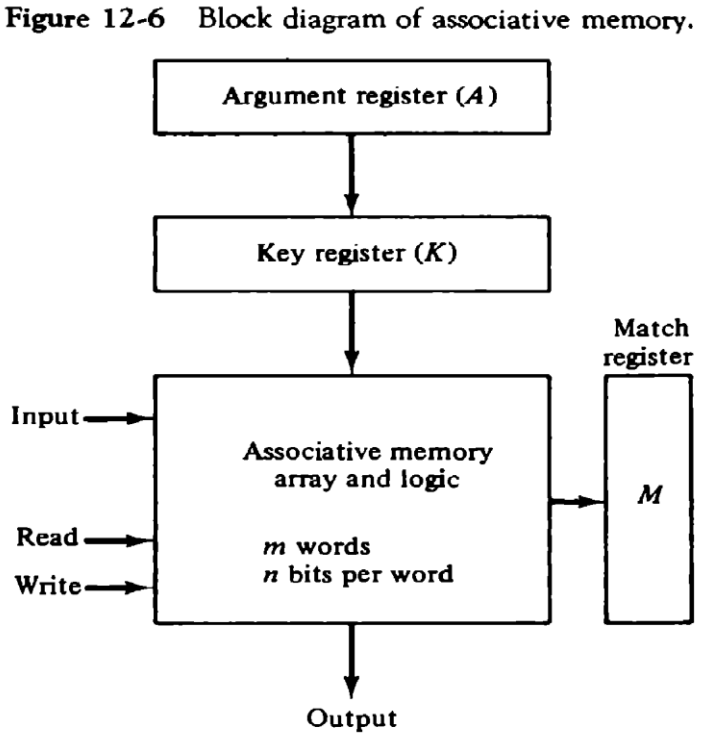
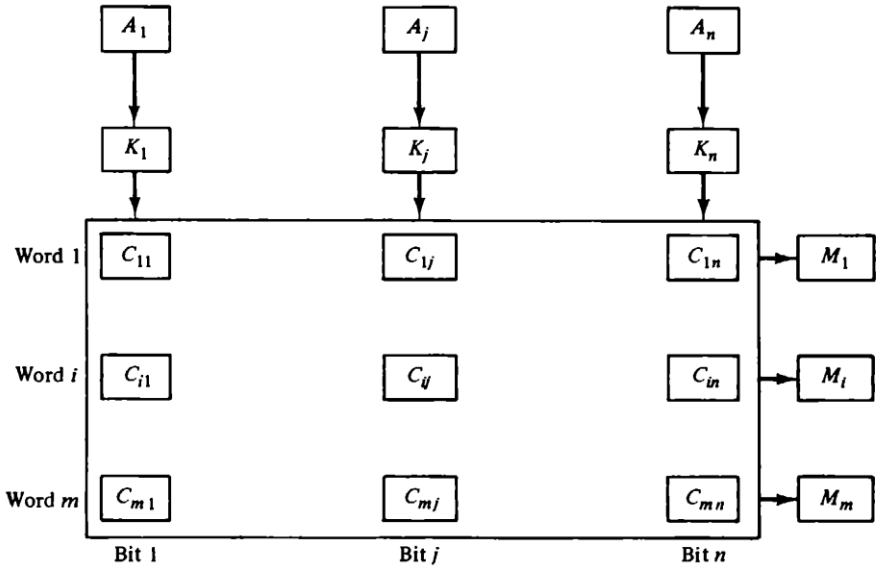


Figure 12-6 Block diagram of associative memory.

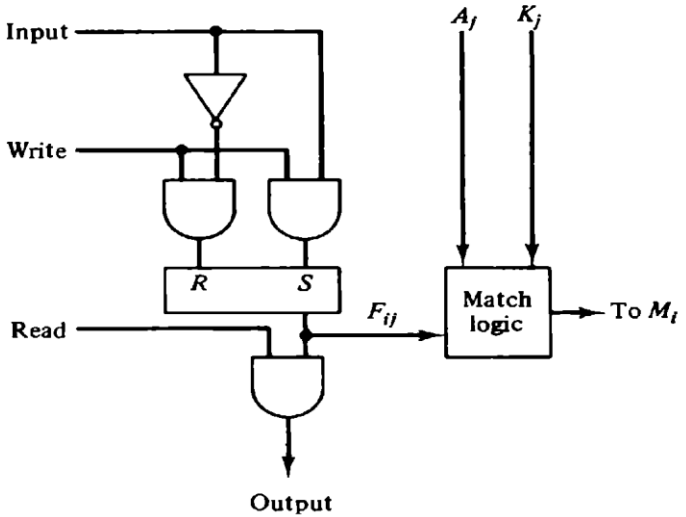
- The relation between the memory array and external registers in an associative memory is shown in Fig.12-7.
- The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and second specifies the bit position in the word. Thus cell C_{ij} is the cell for bit j in word i. A bit A_j in the argument register is compared with all the bits in column j of the array provided that k_j=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 M_i in the match register is set to 1.
- If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

Figure 12-7 Associative memory of m word, n cells per word.



- It consists of flip-flop storage element F_{ij} 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 M_i.

Figure 12-8 One cell of associative memory.



Match Logic

- The match logic for each word can be derived from the comparison algorithm for two binary numbers. First, 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 $A_j = F_{ij}$ 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

$$x_j = A_j F_{ij} + A_j \text{ ‘}F_{ij}\text{ ‘}$$

- where $x_j = 1$ if the pair of bits in position j are equal;otherwise , $x_j = 0$. For a word i is equal to the argument in A we must have all x_j variables equal to 1. This is the condition for setting the corresponding match bit M_i to 1. The Boolean function for this condition is

$$M_i = x_1 x_2 x_3 \ldots\ldots x_n$$

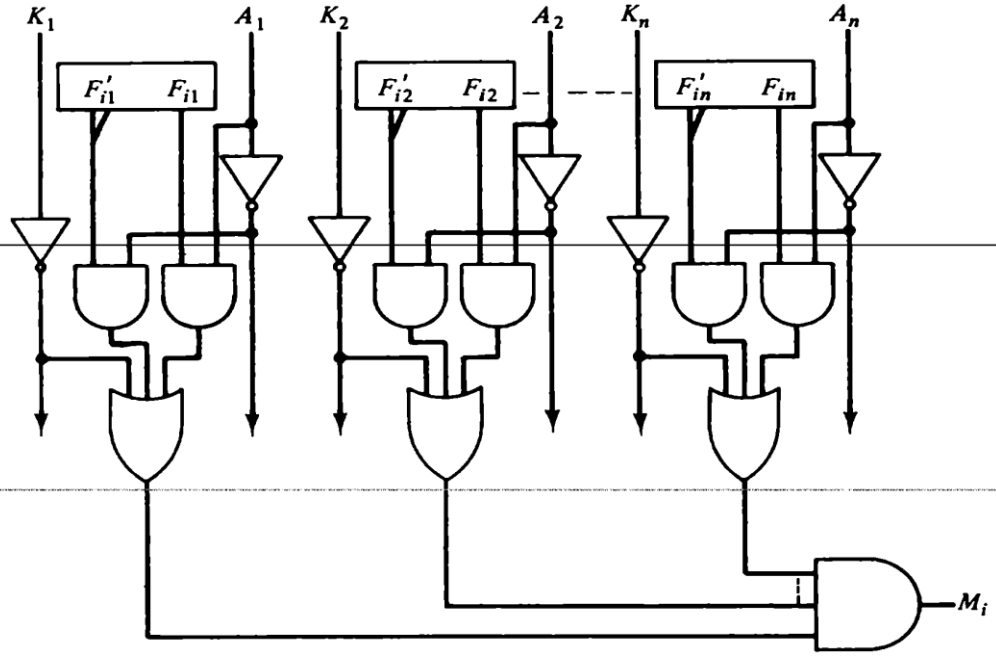


Figure 12-9 Match logic for one word of associative memory.

- Each cell requires two AND gate and one OR gate. The inverters for A and K are needed once for each column and are used for all bits in the column. The output of all OR gates in the cells of the same word go to the input of a common AND gate to generate the match signal for M_i. M_i will be logic 1 if a match occurs and 0 if no match occurs.

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 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 line Where m = 2^d , m is number of words.

2.Unwanted words to be deleted and new 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 interval of time tend to be confined within a few localized areas in memory
- 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”.

Locality of Reference

Locality of reference is manifested in two ways :

1. Temporal- means that a recently executed instruction is likely to be executed again very soon.
    - The information which will be used in near future is likely to be in use already(e.g. reuse of information in loops)
2. Spatial- means that instructions in close proximity to a recently executed instruction are also likely to be executed soon
    - If a word is accessed, adjacent (near) words are likely to be accessed soon (e.g. related data items (arrays) are usually stored together; instructions are executed sequentially
3. If active segments of a program can be placed in a fast (cache) memory , then total execution time can be reduced significantly
4. Temporal Locality of Reference suggests whenever an information (instruction or data) is needed first , this item should be brought in to cache
5. Spatial aspect of Locality of Reference suggests that instead of bringing just one item from the main memory to the cache ,it is wise to bring several items that reside at adjacent addresses as well (ie a block of information)

Principles of cache

- 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 cahache. 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 "**

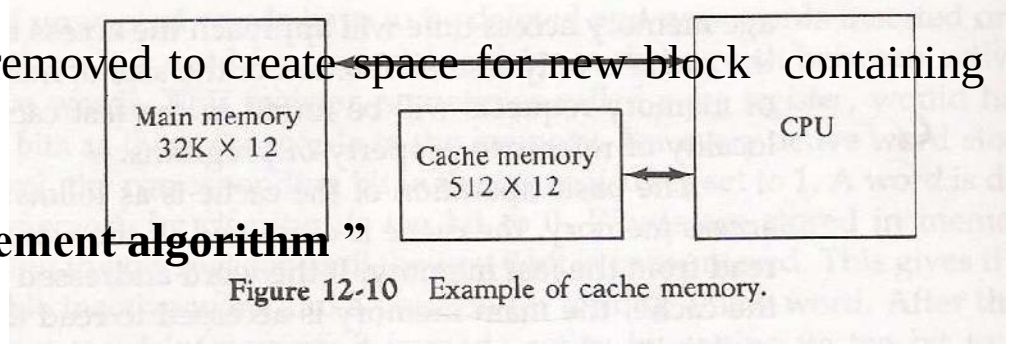


Figure 12-10 Example of cache memory.

Read/ Write operations on cache

Cache Hit Operation

- CPU issues Read/Write requests using addresses that 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 simultaneously **(“ Write Through ”)** OR
2. Update only cache location and mark it as “ Dirty or 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.

- In figure, 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.

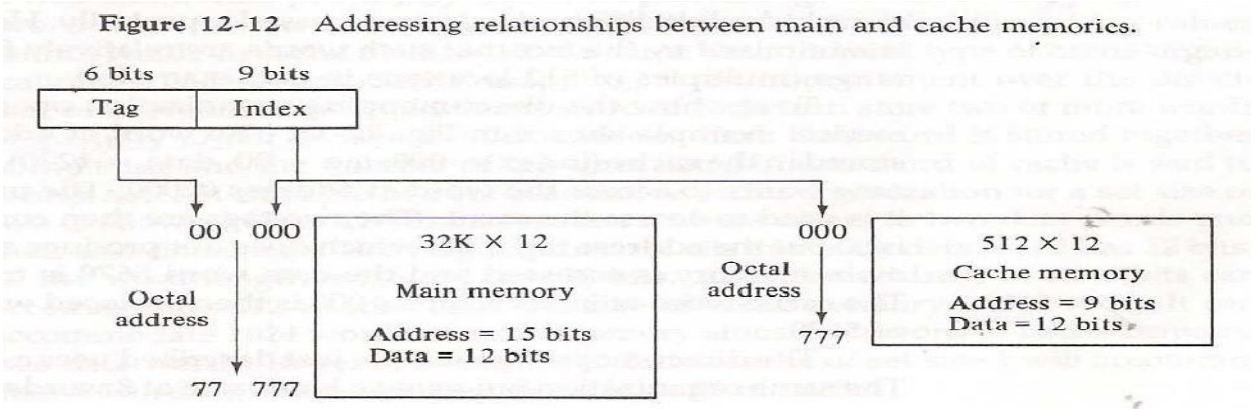


Figure 12-12 Addressing relationships between main and cache memories.

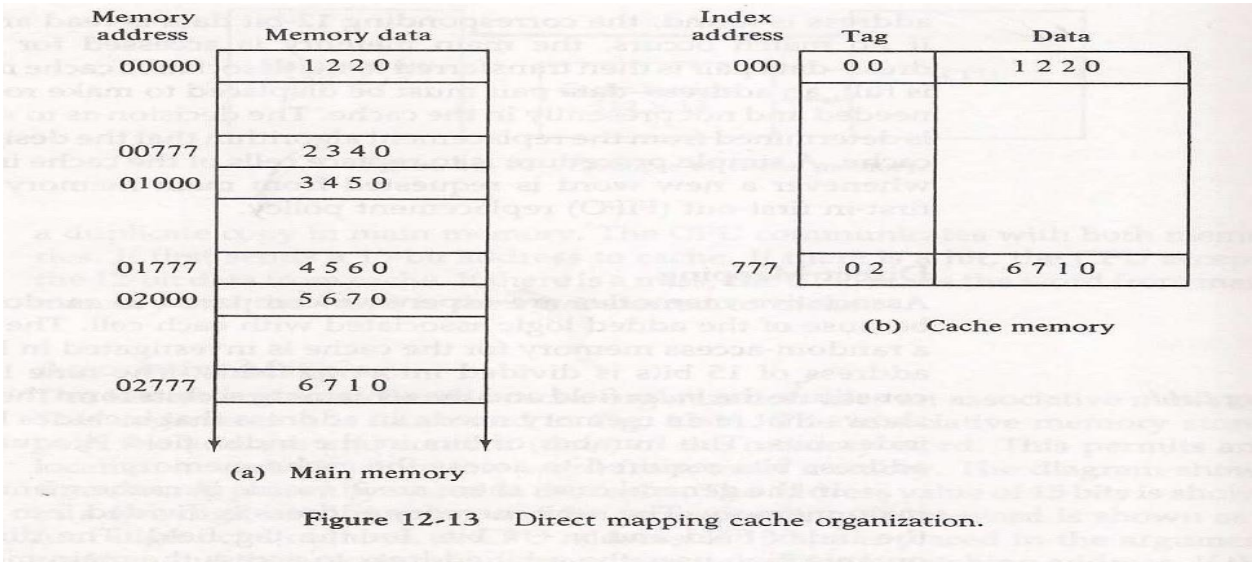


Figure 12-13 Direct mapping cache organization.

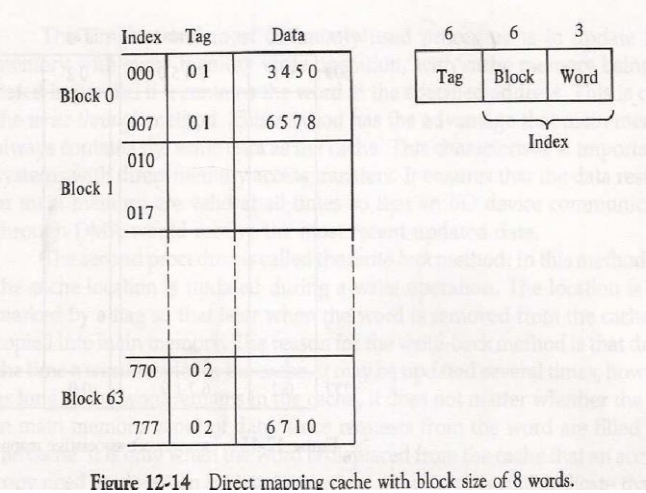


Figure 12-14 Direct mapping cache with block size of 8 words.

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.
- In figure, 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 octal number and its corressponding 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.

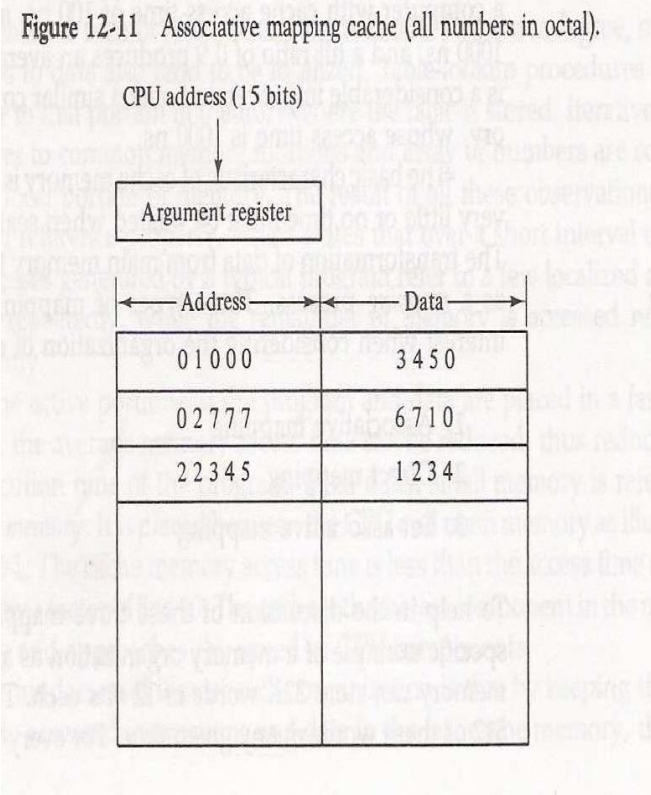
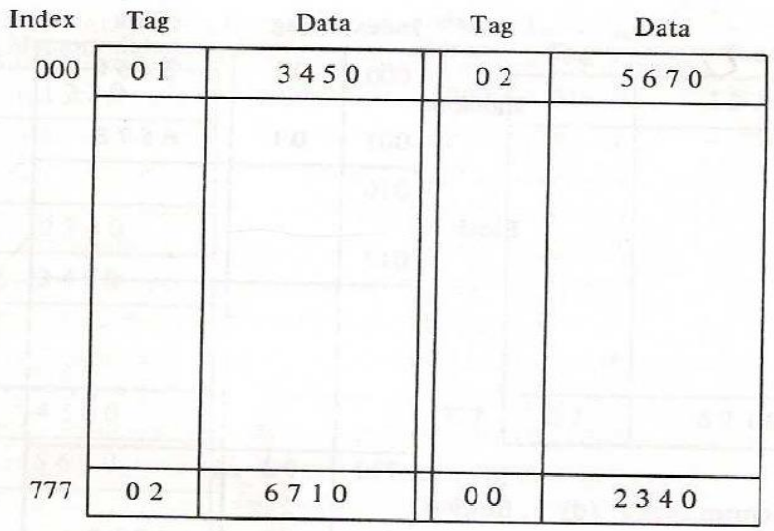


Figure 12-11 Associative mapping cache (all numbers in octal).

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.
- 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”.



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

Figure 12-15 Two-way set-associative mapping cache.

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 that has been least recently used by CPU)