Chapter 4

MEMORY ORGANIZATION

Overview

- Integrated Circuit
- Transistors
- Metal-Oxide Semiconductor
- Classifications of Computer Memory
- Main Memory
- RAM and ROM
- Read and Write Operations
- Internal Organization of Memory Chip
- Auxiliary Memories
- Cache Memory and Cache Memory Mappings
- Replacement Algorithms
- Virtual Memory

4.1 Integrated Circuits

Integrated Circuit or **IC** is a small silicon semiconductor crystal called a ***chip***. Digital systems are constructed with multiple ICs. In a digital system board there are many integrated circuit chips. Inside the chip, there are various digital gates interconnected to form the required circuit i.e. each IC chip has number of small circuits incorporated in them.

Classification of IC packages

Based on the number of gates that can be put in a single IC, chip packages are classified as:

(a) Small Scale integration (SSI) devices:
Here in a single package, there are usually less than 10 gates.

(b) Medium-scale integration (MSI) devices:
Here in a single package there are approximately 10-200 gates.

Example:
Decoders, Adders, Registers, etc.

(c) Large scale integration (LSI) devices:
Here in a single package there are 200- few thousand gates.

Example:
Processors, Memory Chips etc.

(d) Very-large-scale integration (VLSI) devices: Here in a single package there are thousands of gates.

Example:
Large memory arrays & complex microcomputer chips.

Technologies for Implementing ICs:
ICs are implemented using some special technologies like:

- ***(a) TTL:*** Transistor–Transistor Logic
- ***(b) ECL:*** Emitter–Coupled Logic
- ***(c) MOS:*** Metal-Oxide Semiconductor
- ***(d) CMOS:*** Complementary MOS.

4.2 Transistors

Transistors are basically miniature electronic switches. They can also be described as solid state semiconductor devices. Transistors are the building blocks of microprocessors. They are turned on and off by electrical signals. The on/off (binary) switching of transistors actually facilitates the work performed by the microprocessors.

Transistor Types

Transistors can be of either unipolar types or of bipolar types depending on the flow of carrier types.

a) Unipolar

Transistors of ***unipolar*** types depend on the flow of one type of carrier, either electrons (n-channel) or holes (p-channel).

b) Bipolar

In bipolar type of transistors, both electrons and holes move simultaneously.

4.3 Metal-Oxide Semiconductor

Metal-Oxide Semiconductor ***(MOS)*** is a unipolar transistor. Transistors of MOS technology are known as ***field-effect transistors*** or **MOSFETs**.

Types of MOS

MOS can be of the following 3 types:

(a) PMOS: The p-channel (+ve) MOS which depends on the flow of holes.

(b) NMOS: The n-channel (-ve) MOS which depends on the flow of electrons. These are commonly used in circuits with only one type of MOS transistor.

(c) CMOS: The complementary MOS (CMOS) technology uses both PMOS and NMOS transistors connected in a complementary fashion in all circuits.

4.4 Computer Memory

Memory is the storage unit of a computer i.e. the purpose of memory is to store both instructions & data. It is also called the ***Random-Access Memory*** **(RAM)** because the CPU can access any memory location at random.

Memory unit is made up of number of small memory cells, each of which is a ***flip-flop***. Memory cells are usually organized in the form of an array. Each cell is capable of storing one bit of information. As memory uses semiconductor chips, it is known as semiconductor memory.

Memory Cell:

Memory cells are actually flip-flops. Each cell is capable of storing one unit or bit of information (either a 0 or a 1). Each memory cell has the following ***properties:***

(a) A cell exhibits 2 states → binary 0 and binary 1.

(b) It is possible to write into the cell (at least once) to set the state of the cell.

(c) It is possible to read or sense the state (i.e. content) of the cell.

Functions of the Different Terminals of a Memory Cell:

Each memory cell has 3 functional terminals: select, control, data-in / sense. ***Each terminal carries an electric signal.***

(a) Select terminal:

This terminal selects a particular memory cell, out of many memory cells in a chip, to perform read / write operations.

(b) Control terminals:

Such terminals indicate whether the operations to be performed in a memory cell, is read or write.

(c) Data-In Terminal and Sense Terminal:

For writing, the electric signal given at the ***data-in terminal*** sets the state of the cell to either 0 or 1 and for reading the ***sense terminal*** is used to output the cell's state i.e. to know whether the cell's content is 1 or 0. So for reading/writing the cell's content, a single terminal (either data-in / sense) is provided i.e. the same terminal acts either as data-in or as sense.

Suppose to write data, the terminal is enabled or made high to act as data-in and to read data, the terminal is made disabled or made low to act as sense terminal (or vice versa).

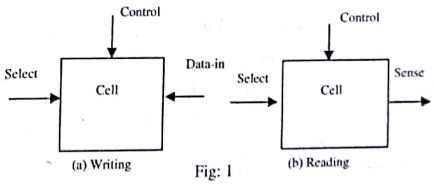


Fig: 1

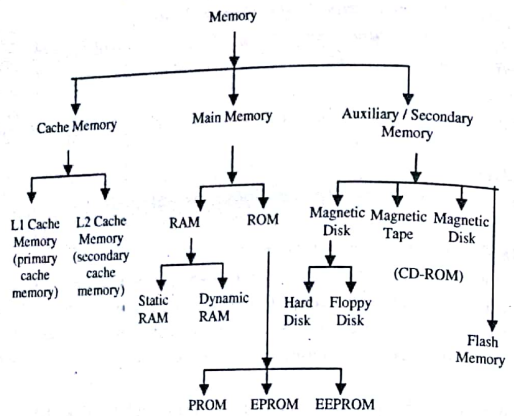
4.5 Memory Classifications and Hierarchy

Memory can be broadly classified into three main parts: the cache memory, the main memory and the auxiliary memory. The cache memory lies between the CPU and the main memory and is the fastest, smallest and the most expensive of all the memory units. The auxiliary or the secondary memory unit is the slowest, largest and the least expensive of all the memory units. The main memory lies in between the two.

Memory Classification:

Memory can be classified accordingly:

Chart:



Memory Hierarchy:

Block Diagram:

The block diagram of memory hierarchy is as follows:

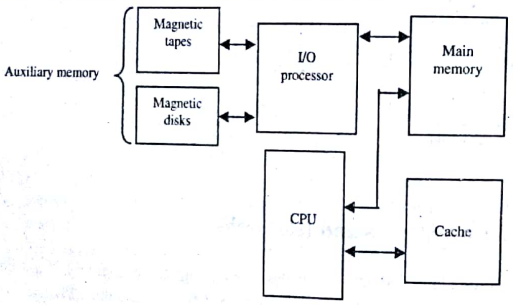


Fig: 2 Memory hierarchy in a computer system

Explanation:

The total memory capacity of a computer can be visualized as being a hierarchy of components. The fig 2, illustrates the components in a typical memory hierarchy.

(a) The ***magnetic tape*** being the slowest component in the hierarchy lies at the bottom. These tapes are used to store removable files.

(b) Next lies the ***magnetic disks*** used as backup storage.

(c) In the central position of the hierarchy, lies the ***main memory***. It can communicate directly with the CPU and the auxiliary memory devices through I/O (input-output) processor. Main memory generally stores the programs that are currently needed by the CPU.

(d) In between the main memory and the CPU lies the ***cache memory***. It lies at the top (not considering the registers) of the memory hierarchy.
Cache memory is the fastest, smallest but the most expensive among the all memory devices. It stores the program segments and data that are needed frequently by CPU in current executions.

(e) The ***I/O processor*** manages data transfer between auxiliary and main memory.

Basic Objectives of Memory Hierarchy:

The basic objective of using a memory hierarchy is to obtain the highest-possible access and transfer speed, maximum storage capacity, while minimizing the total cost of the entire memory system.

Factors on which Memory Hierarchy depends:

There are various factors on which the basic hierarchy of memory depends. These are cost, storage capacity (size), and speed and access time.

Diagram:

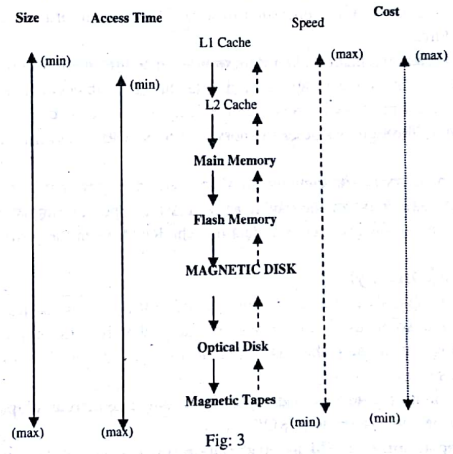


Fig: 3

L1 cache memory is the smallest, fastest and most expensive among all the memories. It has also got the minimum access time. Whereas the magnetic tapes are the slowest, least

expensive but largest among the memory devices. These memories have got the maximum access time.

Hence the cache memories are the fastest, costliest but smallest devices among the memory units whereas the auxiliary memories are the slowest, cheapest but largest devices among the memory units. In between these two lie the main memory devices.

4.6 Main Memory

The main memory is the central storage unit in a computer system. The principal technology used for the main memory is based on semiconductor integrated circuits.

RAM and ROM:

RAM and ROM are integrated circuit chips that make the main memory of a computer. While RAM (random-access memory) is the memory unit in which any location can be accessed for a Read or Write operation in some fixed amount of time independent of the location's address, ROM (read-only memory) is used for permanent storage of information and it also possesses random access property.

(a) RAM (Random-Access Memory):

This is the read and write memory (R/W memory) of a computer, in which any location can be accessed for a Read or Write operation in some fixed amount of time independent of the location's address.

The users can both read information from it as well as write information into it. RAM is used for storing the bulk of the programs and data that are subject to change. RAM possesses random access property i.e. any memory location can be accessed in a random manner without going through any other memory locations. The access time is same for each memory location.

RAM is a volatile memory i.e. its contents are destroyed when power is turned off. The information written into it is retained in it as long as the power supply is on. The programmer has to reload his program and data into the RAM when the power supply is resumed.

(b) ROM (Read-Only Memory):

This is nonvolatile in nature (i.e. the information stored in it is not lost even if the power supply goes off) and so is used for storing programs that remain in the computer permanently and also information that are not subject to any change. It is used only to read the data stored in it.

Though most of the main memory is made up of RAM integrated circuit chips, a portion of the memory may be constructed with ROM chips.

Since the normal operation of ROM involves only reading of stored data, this type of memory is thus called read-only memory. The contents of ROM are permanently stored only at the time of manufacture.

4.7 Typical RAM and ROM Chips

RAM and ROM chips consist of many RAM and ROM cells respectively. While a RAM cell has both data-in (write) and sense (read) lines, a ROM cell has only sense (read) line. RAM and ROM chips are available in a variety of sizes depending on the on the size of the memory system. If the memory needed for the computer is larger than the capacity of one chip, it is necessary to combine a number of chips to form the required memory size. i.e. say, for example, a memory size of 1024 bytes can be formed by combining four 128 byte RAM chips and one 512 byte ROM chip.

A typical RAM chip:

A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip (from among other chips on the board) only when needed.

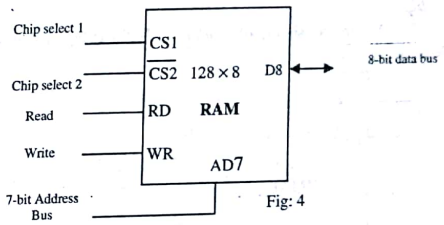


Fig: 4

CS1 & $\overline{\textbf{CS2}}$ are the chip selects (control inputs) that select a particular chip, from among many chips, in a board.

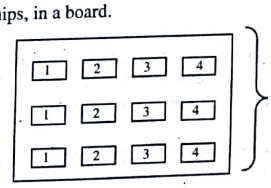


Fig: 5

Say, in the memory system board, there are 3 rows of chips, each row having 4 chips. To select, say, the 2nd chip in the 2nd row out of all the other chips, the 'chip select' lines of that particular chip is to be enabled.

The ***data bus*** transfers data in and out of the chip depending on the memory operations specified by the Write and Read lines. So when ***Write (WR)*** input is enabled, the memory stores a byte from the data bus into the specific chip location and when ***Read (RD)*** input is enabled, the memory places the content of the selected byte into the data bus. Sometimes the 'read' & 'write' lines are combined into one line labeled as ***R/W***.

The ***address bus*** specifies the address of the location in the memory chip from which the data is to be read or to which the data is to be written.

Now, the RAM chip in the figure is a 128 × 8 RAM chip. This means that the storing capacity of this RAM chip is 128 words & each word has 8 bits in it. So, this RAM chip has 7-bit (since $2^7 = 128$) address bus (this can specify any one of the 128 words in the chip) and 8-bit (since 8 bits / word) data bus.

All RAM chips has bi-directional data bus (constructed with tri-state buffers) that does the following 2 functions:

(a) Transfer data from memory to CPU during memory read operation (in this case the Read line is enabled and data is transferred via the bus from memory to CPU).

(b) Transfer data from CPU to memory during memory write operation (in this case the Write lines is enabled and data in transferred via the bus from CPU to memory).

A typical ROM chip:

Organization of a ROM chip (consisting of number of ROM cells) is almost the same as that of the RAM chip.

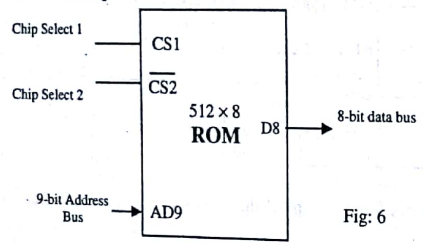


Fig: 6

The ROM chip in the fig 6, is a 512 × 8 ROM chip. This means that the storing capacity of this chip is 512 words & each word has 8 bits or cells in it. So, this ROM chip has 9-bit (since $2^9 = 512$) address bus (this can specify any one of the 512 words in the chip) and 8-bit (since 8 bits / word) data bus.

Since a ROM can only read, the data bus can only be in an output mode and there is no need for read / write line as the unit can only read.

The functions of the data bus and address bus are the same as in the RAM chip (only that the data bus operates in the read mode and hence it is unidirectional incase of ROM chip and address bus carries the address of the location to be read).

4.8 Working of RAM and ROM Chips

A RAM chip supports both reading and writing of data whereas a ROM chip supports only reading of data.

Working of a RAM Chip:

A ***RAM chip*** works as follows (consider fig 4):

Writing or storing data in to the chip:

Steps:

(a) Control or selection inputs (CS1 and $\overline{CS2}$) select the particular chip out of many other chips on the board (i.e. of all the chips of the memory system).

(b) If it is a memory 'write' operation, hence the 'write' line (WR) is to be enabled.

(c) The address of the specified memory location (on the selected chip) where data is to be written is sent to the memory from the CPU via the address bus (AD7).

(d) Data to be stored (i.e. to be written) is sent to the memory from the CPU via the data bus (D8).

(e) That data is then stored on the specified memory location as per the address given.

Reading data from the chip:

Steps:

(a) Control or selection inputs select the particular chip out of many other chips on the board (i.e. of all the chips of the memory system).

(b) As it is a memory 'read' operation, hence the 'read' line is to be enabled.

(c) The address of the specified memory location (on the selected chip) from where the data is to be read is sent to the memory from the CPU via the address bus.

(d) Data is read (from the specified memory location) and sent to the CPU from the memory via the data bus.

Working of a ROM Chip:

A ***ROM chip*** (consider fig 6) works in a similar way as the RAM chip only that in a ROM chip data can only be read from the memory chip (data cannot be stored and hence no specific read or write lines are provided).

Reading data from the ROM chip:

Steps:

(a) Control or selection inputs (CS1 and $\overline{CS2}$) select the particular chip out of many other chips on the board (i.e. of all the chips of the memory system).

(b) The address of the specified memory location (on the selected chip) from where the data is to be read is sent to the memory from the CPU via the address bus (AD9).

(c) Data is read (from the specified memory location) and sent to the CPU from the memory via the data bus (D8).

4.9 Internal Organization of Bit Cells in a Memory Chip

The following diagram shows the internal organization of RAM / internal organization of main memory chip consisting of multiple bit cells.

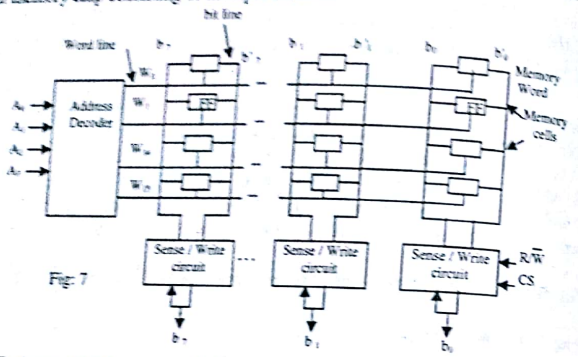


Fig: 7

Figure 7, shows 16 * 8 memory chip i.e. the chip has 16 words of 8-bits (cells) per word i.e. the chip has 16 rows and 8 columns of RAM cells (i.e. total of 128 cells). Hence it can store 128 bits where each cell is a flip-flop storing 1 bit.

Each row of cells constitute a memory word i.e. all cells in row 1, constitute a memory word, all cells in row 2 constitutes another memory word & so on.

All cells of a row are connected to a common 'word line', which in turn is connected to the address decoder on the chip. So the decoder has 16 output lines (since 16 words or 16 rows). Hence, the decoder is having 4 input lines.

The address decoder takes as input, the required word address sent by CPU and selects (enables) the particular word line (i.e. the particular word, as each word has a separate word line) out of the many words in that chip.

In the fig 7, as there are 16 words in the chip, there are 16 word lines also. The cells in each column are connected to a sense / write circuit by two bit lines ('b_i' & 'b_i''). These circuits are connected to the data input / output lines of the chip.

There is a Read / Write line (R / $\overline{W}$) connected to all sense / write circuits. During a read operation, these circuits sense, or read, the information (i.e. places the memory operation in the 'read' mode) stored in the cells selected by a word line and via the data input / output lines, these data are transmitted to the data bus.

During a write operation, the sense / write circuit places the memory in the 'write' mode such that information are received from the data input / output lines (via the data bus) and get stored in the cells of the selected word.

The CS (chip select) input selects a given chip in a multi-chip memory system.

4.10 Memory Read and Write Operations

Memory Read Operation:

Steps:

(a) CPU places the address of the desired memory location from which the data is to be read on the address bus.

(b) CPU then places the 'read' signal on the control bus.

(c) A main memory system can have multiple memory chips. There is the external address decoder in the main memory system that decodes the CPU address to find out the particular memory chip that has the desired word in it. Then that chip is selected by the chip select (CS) lines of that chip, based on the decoded address.

(d) After the particular chip is selected, the required CPU address is then decoded by the internal address decoder of that particular chip to select the required word line (row of cells) that has the required address.

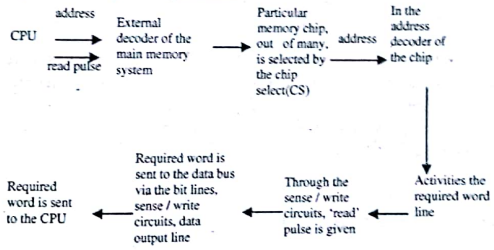
(e) Meanwhile, the Sense / Write circuits of the cell-columns (in the particular chip), sense the 'read' pulse from the CPU.

(f) As it is a 'read' memory operation hence the 'Sense' (R) line gets enabled.

(g) So, data from the selected cells (i.e. the selected word) on the activated word line comes to the bit lines and then via the sense / write circuit goes to the data output lines.

(h) Through the data output lines, the information is passed to the data bus and then to the CPU.

Flow chart of the memory read operation:



Memory Write operation:

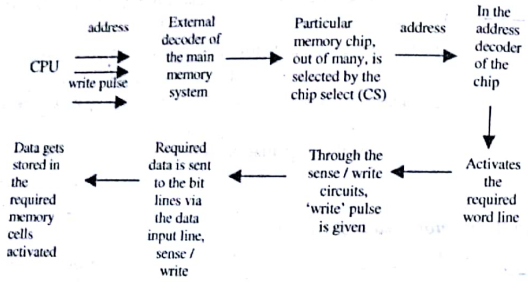
Steps:

(a) CPU places the address of the desired main memory location where the data is to be written on the address bus.

(b) CPU then places the 'write' signal on the control bus.

(c) Then the data to be written is placed by the CPU on the data bus.
(d) Same as (c) memory read operation.
(e) Same as (d) in memory read operation.
(f) Meanwhile, the Sense / Write circuits of the cell-columns (in the particular chip), sense the 'write' pulse from the CPU.
(g) As it is a 'write' memory operation hence the 'Write' (W) line gets enabled.
(h) Through the 'data input' lines the data is placed on the respective sense / write circuits (i.e. if say 01110011 is to be written on this 16 * 8 chip, then the first data input line 'b_0' will place '0' in the first sense / write circuit; the second data input line 'b_1' will place '1' on the second sense / write circuit and so on. The 8th data input line 'b_7' will place last '1' in the 8th sense / write circuit).
(i) Through the bit lines, the bits of the required word will be stored in the respective cells of the activated word line.

Flow chart of the memory write operation:



[*Note:* Here only the 'read' and 'write' operation of the main memory is discussed. CPU only knows about the main memory system. Cache & Auxiliary memories are not known to the CPU. Accessing of cache and auxiliary memories (as well as main memory) is controlled fully by the memory management unit. CPU has no idea about what is happening inside the entire memory system]

4.11 Static and Dynamic RAMs

RAM chips are available in 2 possible operating modes:
(i) Static RAM:
It consists of internal flip-flops that store the binary information. Stored information remains valid as long as power is applied to the unit.

Static memories, consisting of Static RAM chips are memories consisting of circuits capable of retaining their state as long as power is on.

(ii) Dynamic RAM:
A dynamic RAM loses its stored information in a very short time (a few milliseconds) even though the power supply is on.
Information is stored in a dynamic memory cell in the form of charge on a capacitor and this charge can be maintained only for a very few milliseconds. As the charge on the capacitors leak away as a result of normal leakage, the capacitor gets turned off after the few milliseconds. So, to retain the cell information for a much longer time, the cell's content must be periodically refreshed to restore the capacitor charge to its full value.

Working Principle of a Static RAM Cell:
Diagram of a static RAM cell:

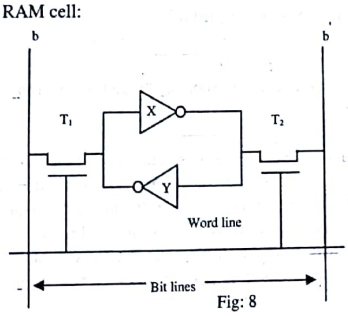


Fig: 8

In the figure 8, as can be seen, two inverters are cross-connected to form a latch. b and b' are the 2 bit lines. T_1 and T_2 are the 2 transistors connecting the latch to the bit lines. The transistors can be opened or closed under the control of the word line and hence acts as a switch.

Write operation:
Suppose in a static RAM organization, the nth cell (a memory organization consists of multiple memory chips and each chip consists of multiple memory cells) is to be written. Let the nth cell be located in the i^{th} row and in the j^{th} column of the specific memory chip.
Steps:
(a) CPU sends the desired memory location's address to be written through the address bus. 'Write' signal is then placed in the control bus. Decoding the address firstly by the external decoder of the memory organization (to select the desired chip) and then

by the internal decoder, the desired memory location is selected (i.e. the i^{th} word line is enabled).

(b) CPU places the data to be written in the data bus. Via the data input/output lines, the data gets transmitted to the respective sense/write circuits.

(c) Now, the i^{th} word line is to be enabled.

(d) The contents of the b^{th} and b'^{th} bit lines come from the sense/write circuit. Suppose '1' is to be stored in the cell. Hence b^{th} bit line must contain '1' and so content of b'^{th} bit line will be 0. That is, what is to be stored, must be placed in the b^{th} line and the b'^{th} line will just hold the compliment of that.

(e) Enabling the i^{th} word line will make the transistors T_1 and T_2 to get charged and they will act as closed switches (closed circuits).

(f) So, the contents of b^{th} bit line (i.e. 1) and that of the b'^{th} bit line (i.e. 0) will come to X and Y respectively.

(g) Due to the cross–connected inverters the contents of X (i.e. 1) and Y (i.e. 0) will be retained as it is until the power is turned off.

Read operation:

This is the same as the write operation. Suppose to read the data in the n^{th} cell located in the i^{th} row and j^{th} column.

Steps:

(a) CPU sends the desired memory location's address to be read through the address bus. 'Read' signal is then placed in the control bus. Decoding the address firstly by the external decoder of the memory organization (to select the desired chip) and then by the internal decoder, the desired memory location is selected.

(b) So, let the i^{th} row (i^{th} word) be selected by enabling the i^{th} word line.

(c) After enabling the i^{th} word line, the j^{th} column has to be selected by enabling the bit lines.

(d) Enabling the i^{th} word line, the two transistors T_1 and T_2 get charged up and act as a closed switch. Hence the circuit now is a closed one i.e. information can be transmitted.

(e) The contents of X and Y thus get transmitted to the bit lines. Via the bit lines, through the sense/write circuit (the sense line is now enabled as it's a 'read' memory operation) and then through the data input/output lines the contents get transmitted to the bus and then to the CPU.

Working Principle of a Dynamic RAM Cell:

Diagram of a single-transistor dynamic memory-cell:

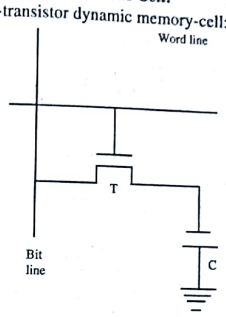


Fig: 9

A typical dynamic ROM cell consists of a capacitor, transistor and a sense amplifier. The sense amplifier senses (or read) the charge content in a static RAM cell.

Write operation:

To store or write information on the nth cell in the i^{th} row:

Steps:

(How the desired memory location's address, data to be written and 'write' signal comes from the CPU to the memory system, is the same as has been discussed in case of static RAM)

(a) The i^{th} word line is to be enabled.

(b) So, the transistor 'T' is turned on and it acts as a closed switch (a closed circuit).

(c) This applies the appropriate voltage to the bit line.

(d) Hence the capacitor (C) gets charged i.e. charge from the respective bit line gets stored in the capacitor.
The presence of charge in the capacitor i.e. a charged capacitor represents '1' or it can be said that '1' is stored in the capacitor. Absence of charge represents a '0' i.e. a discharged capacitor represents that no charge (i.e. 0) is stored.

Read operation:

To read the content of the nth cell in the i^{th} word line:

Steps:

(a) The i^{th} word line is enabled.

(b) Hence T gets turned on, and acts as a closed switch.

(c) Capacitor charge comes to the bit line.

(d) There is a sense amplifier. It senses or detects the charge from the capacitor, whether a '1' or a '0'.

The charge on the capacitor can be retrieved correctly i.e. it can be said that the charge stored is '1', if there is a specific quantity of charge stored in the capacitor. If the charge goes below that specific value then it indicates that a '0' is stored.

So, if the sense amplifier senses that the charge is above the threshold value, it indicates that a '1' is stored else '0' is stored.

Refreshing the cell's content:

If the sense amplifier senses that the capacitor's charge is above the threshold value, it enables the bit line to a full voltage which in turn recharges the capacitor to the full charge that corresponds to logic value 1 i.e. recharging the capacitor to the full charge means that again '1' is restored in the cell.

If the capacitor's charge is below threshold value, it means that capacitor has no charge and thus logic value '0' is stored in the cell.

So, in case of a dynamic RAM, reading the contents of the cell automatically refreshes its contents. All cells in a selected row are read at the same time, which refreshes the contents of the entire row.

Differences between Static RAM and Dynamic RAM:

The differences between static and dynamic RAMs are as follows:

Static RAM	Dynamic RAM
1. Such kind of RAM retains the stored information as long as the power supply is on.	1. Loses its stored information in a very short time (a few milliseconds) even though the power supply is on.
2. Circuit consists of multiple transistors (generally 6) and two cross-connected inverters (latch).	2. Circuit consists of lesser number of (generally 1) transistors and a capacitor.
3. Retaining of information depends on the power supply. It holds information in flip-flops.	3. Depends on how long can the capacitor retain its charge. Holds information as charge on a capacitor.
4. Costlier due to the requirement of multiple transistors.	4. Comparatively Cheaper.
5. Less packing density	5. High packing density.
6. Faster	6. Moderate speeds but slower than static RAMs.
7. Power consumption is high.	7. Consume less power.
8. Don't need refreshing of the circuitry.	8. Needs refreshing of the circuitry.
9. Storage capacity in a single memory chip is less.	9. Storage capacity in a single memory chip is more.

Static RAM	Dynamic RAM
10. Has a shorter read and write cycle.	10. Has a larger read and write cycle.
11. More user-friendly.	11. Less user-friendly.
12. Denoted as SRAM	12. Denoted as DRAM.

Uses of Static and Dynamic RAMs:

Static and Dynamic RAMs have the following uses:

Uses of Static RAM:

Being faster, static RAMs are used in Cache memory and in other applications where speed is of critical concern.

Uses of Dynamic RAM:

As dynamic RAMs are less expensive and posses high density, they are widely used in the main memory units of computers.

4.12 Working Principle of a Typical ROM Cell

Diagram of a typical ROM cell:

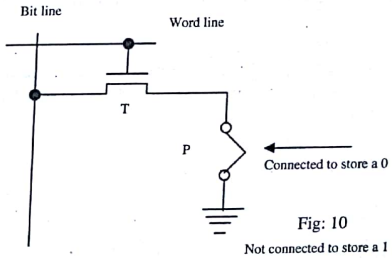


Fig: 10

At point P there is a transistor. The contents of a ROM cell are stored permanently at the time of its manufacture.

If the transistor is connected to ground at point P (i.e. if P is a closed path), the cell stores a logic value 0, else stores 1.

Reading information from a ROM Cell:

Steps:

(How the desired memory location's address and the 'read' signal come from the CPU to the memory system, is the same as has been discussed in case of static RAM).

(a) Word line of the respective cell to be read, is activated.

(b) Transistor gets charged up & gets closed.
(c) If the path at P is closed i.e. if there is a connection between ground & T then the bit line reads as 0, else reads as 1.
(d) These values (whether 0 or 1) are generated by a sense circuit at the end of the bit line.

Example of a typical ROM cell:
Toshiba Mask ROM etc.

4.15 Different Types of ROM

ROM may be of the following types:

(a) PROM (Programmable ROM):
PROM's contents are decided by the user i.e. data can be loaded by the user.
Diagram of a typical PROM cell is the same as a typical ROM cell only that at the point P (figure 10) in PROMs, all cells contain a fuse instead of a transistor. Now before programming, all cells contain 0s. Only, users using high current pulses can burn the fuses of those cells to be programmed. Burning of fuses at point P means that a '1' is stored at the cell.
This process however is irreversible.
(***Note:*** Initially if 0 is stored, burning will store a 1 or initially if 1 is stored, burning will store a 0. This depends on the architecture of the PROM and varies depending on the different manufacturing companies).
Uses:
For research and development purposes.
Example:
74S287 chip.

(b) EPROM (Erasable PROM):
Here the ROM chip allows the stored data to be erased & new data to be loaded.
In EPROM's, a special transistor (instead of fuse in PROM) is used at point P (figure 10). By passing electric current i.e. activating the word line, this transistor at point P can be enabled or disabled as needed. So, when enabled, it stores logic '1' and when disabled stores logic '0'. Thus this special transistor at P acts as a switch.
Dissipating the charges in the special transistor at P, by exposing the chip to ultraviolet (UV) light can erase the contents of EPROM chips. After erasing, the chips can again be reprogrammed as needed.

Advantages:
Contents of EPROM chips can be erased & reprogrammed.
Disadvantages:
The EPROM chip:
(a) must be physically removed from the circuit for erasing & reprogramming.
(b) Chip's contents are erased by UV light.
Uses:
(a) Used for software developing purposes.
(b) Used for research & development purposes.
Example:
Intel's 87C257, 256K (32K x 8) CHMOS EPROM etc.

(c) EEPROM (Electrically Erasable PROM):
Also known as EAPROM (Electrically Alterable PROM).
EEPROM chips do not have to be removed from the circuit board for erasure. The contents of EEPROM can be both erased & programmed electrically. It is possible to erase the cell contents selectively on a byte-by-byte basis. In EEPROM, it is possible to read & write the contents of a single cell.
Advantages:
(a) Chip can be erased & programmed on the board itself.
(b) Both erasing & Programming can be done electrically.
(c) Cell contents can be both erased & written selectively on a byte (word) by byte (word) basis.
Disadvantages:
Different voltages are needed for reading, writing & erasing the stored data.
Uses:
For research & development purposes.
Example:
Intel 2816A (a 16K i.e. 2K x 8) chip.

(d) Flash Memory:
It is electrically erasable and programmable permanent type memory. It uses the same approach as EEPROM. Just like an EEPROM cell, it is also based on a single transistor that is controlled by trapped charge. Also just like on EEPROM, entire contents of a flash memory can be erased in one operation. The name 'Flash memory' has been given due to very fast reprogramming capability.
Advantages:
Have greater density, higher capacity & low cost / bit, less power consumption & require a single power supply.

Disadvantages:
Here though it is possible to read the contents of a single cell, it is only possible to write an entire block of cells & not a single one.

Uses:
For its low power consumption, flash memories are used in battery-driven portable equipment like hand-held computers, cell phones, digital cameras etc.

4.14 Differences

Differences between ROM and RAM:
The differences between ROM and RAM are as follows:

ROM	RAM
i) Read only memory.	i) Both read & write memory.
ii) Nonvolatile i.e. its contents are not lost even if the power supply goes off.	ii) Volatile i.e. retains its contents as long as the power supply is there.
iii) Used for permanent storage of information.	iii) Used for temporary storage.
iv) Cheaper when produced in large volumes.	iv) Costlier.
v) Stored information can only be read at the time of operation.	v) Can be read when needed.
vi) Can be of PROM, EPROM, EEPROM, Flash memory types.	vi) Can be of static & dynamic types.

Differences between ROM and PROM:
The differences between ROM and PROM are as follows:

ROM	PROM
i) Non-programmable.	i) Programmable.
ii) not flexible (because data cannot be changed).	ii) flexible (Field-programmable i.e. can be programmed in any place of work. On the other hand, if the programming can be done only at the factories or by masking, then it is known as factory-programmable).
iii) slower.	iii) faster.
iv) economical only when produced in large volumes.	iv) economical even when produced in small volumes.
v) used in PC's.	v) used mainly for research & development purposes.
vi) writing of cell contents mainly done by 'masking' mechanisms.	vi) mainly done electrically.

Differences between EPROM and EEPROM:
The differences between EPROM and EEPROM are as follows:

EPROM	EEPROM
i) erasable PROM.	i) electrically erasable PROM.
ii) contents of EPROM chips can be erased by UV light.	ii) contents of EEPROM chips can be both erased & programmed electrically.
iii) to erase & reprogram a chip must be physically removed from the circuit.	iii) such chips do not have to be removed from the circuit.
iv) an entire chip content has to be erased at a time.	iv) possible to erase the cell contents selectively byte by byte.

Differences between EEPROM and Flash Memory:
The differences between EEPROM and Flash Memory are as follows:

EEPROM	Flash Memory
i) possible to read & write the contents of cells byte by byte	i) possible to read the contents of a byte of cells but only possible to write an entire block of cells.
ii) prior to writing, the previous contents of cell-bytes are erased.	ii) prior to writing the previous contents of the block of cells are erased.
iii) less densely packed cells.	iii) more densely packed cells.
iv) lesser capacity.	iv) higher capacity.
v) more cost per bit.	v) less cost per bit.
vi) reprogramming capability is comparatively slower.	vi) comparatively faster.
vii) such chips are suitable for storing & updating parameters.	vii) such chips are suitable for storing & updating firmware (codes).
viii) power consumption is comparatively high.	viii) comparatively low.
ix) used mainly in battery driven portable equipment like cell phones, digital cameras etc.	ix) used mainly for research & development purposes.

4.15 Auxiliary Memories

Auxiliary or secondary memories are low-cost, very large storage devices serving as a backup for storing information that are not currently used by the CPU. Such devices are used for storing system programs, large data files and other backup information. Most common auxiliary memory devices used in computer systems are magnetic disks and tapes.

Need for Auxiliary Memories:
The capacity of the main memory is very limited. It is not possible for computers with limited main memory capacity to store large applications (i.e. large number of programs

and data). In such cases additional storage is required. Hence auxiliary memories of very large storage capacity provide the extra storage and are used to accommodate all information that is not currently used by the CPU.

Explanation of How Information is accessed from Auxiliary Memory Devices:

Auxiliary memory devices serve as a backup for storing information not currently used by the CPU. Only programs and data currently needed by the processor reside in the main memory. All other information is stored in the auxiliary memory and is brought into the main memory as and when needed.

When the CPU requests for some data, that request comes to the main memory via the bus. If the data is not found in the main memory, it is brought from the auxiliary memory. Now information in auxiliary memory is always accessed in form of 'blocks' (i.e. block of data). So the entire block of information is to be transferred to the main memory.

Magnetic Disk:

Magnetic disk is the primary medium for long-term storage of data. Disk storage survives power failures and system crashes.

Structure:

A magnetic disk is actually a circular plate, called a ***platter***, constructed of metal or plastic coated with magnetized material. Several disks may be stacked on one ***spindle***. ***Read / write heads*** (information recorded on the disk surface may be read and information may be stored on the disk surface with the help of this read / write head) may be there on each surface of the disks and all read/write heads are mounted on a single assembly called a ***disk arm***.

Information (in form of bits) is recorded on the magnetized surface in spots along concentric circles called ***tracks***. The tracks (generally there are 20 to 1500 tracks per disk surface) are commonly divided into sections called ***sectors***. So a sector is the smallest unit of information that can be read from or written to the disk. Though sectors, usually, are of 512 bytes but they may vary from 32 to 4096 bytes. Generally there are 4 to 32 sectors per track. The read / write heads move across the platter to access different tracks. Figures 11 & 12 show the magnetic disk structure.

A ***disk controller*** interfaces between the computer system and the actual hardware of the disk drive i.e. the disk controller controls all the disk activities. Address bits that specify the disk number, the disk surface, the sector number and the track within the sector address a disk system. A track in a given sector near the circumference is longer than a track near the center of the disk.

Diagram:

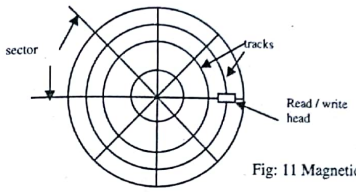


Fig: 11 Magnetic disk – top view

Working Principle of Magnetic Disk:

In a disk the read / write heads are used for reading information from the disk and storing or writing information on the disk. All disks rotate together at high speed. The read / write head must be positioned in the specified track before reading or writing. After the read / write head is positioned in the specified track, the system has to wait until the rotating disk reaches the specified under the read / write head. Once the beginning of a sector is reached, information transfer occurs at a very high rate. So, while reading or writing, the minimum quantity of information that can be transferred is a sector.

The time required for the access of the disk is called the access time. So ***access time*** (ranges from 10 to 40 milliseconds) is the time in between from when a request for read or write is issued to when the actual data transfer begins. This time is divided into:

(a) ***Seek Time*** (ranges from 2 to 30 milliseconds) that is the time taken by the read / write head to get positioned over the correct track (containing the required sector to be read).

(b) ***Latency Time*** (ranges from 60 to 120 rotations per second) that is the time taken by the read / write head to get positioned over the required sector to be accessed.

Finally the rate at which the actual data transfer, from or to the disk, occurs is called the ***data transfer rate*** (ranges from 1 to 5 megabytes per second). The reliability of the disk is measured by '***mean time to failure***' or ***MTF*** concept. It is the amount of time that, on average, a system can be expected to run continuously without failure. It ranges generally from 30,000 to 800,000 hours. Figure below shows the structure of a magnetic disk.

Diagram:

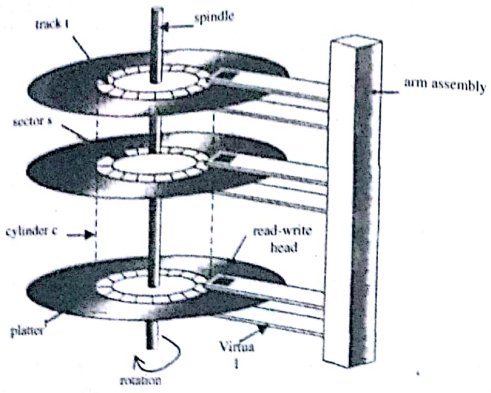


Fig: 12 Structure of magnetic disk

Explanation of how information is read from or written to the magnetic disk:

On each magnetic disk surfaces, a thin magnetic film is deposited. The disks are placed on the rotary drive (spindle) so that the magnetized surfaces move in close proximity to read / write heads. Each head consists of a magnetic yoke and magnetizing coil. Figure 13 shows the mechanical structure of magnetic disks.

(a) Storing or writing information on the disk:

Digital information is stored on the magnetic film by applying current pulses of suitable polarity to the magnetizing coil (thus the yoke act as a magnet). This causes the magnetization of the film in the area immediately underneath the head (i.e. the area gets magnetized and turns to very small magnets and thus bits are stored). Whether a 0 is stored or a 1 is stored in the disk depends on the polarity of the current pulses applied to the coil. Suppose when the current flows from the +ve direction to the –ve direction the direction of magnetic lines of force may be from North to South. This may indicate that a 1 is stored in the disk. Reversing the current direction may reverse the flowing of magnetic lines of force and thus a 0 may be stored (and vice versa).

(b) Reading information from the disk:

In this case the reverse thing happens. As soon as the tiny magnets on the disk surface come under the read/write head of the yoke, magnetic lines of force flows through the coil and a voltage gets induced in the coil, which then serves as a sense coil. Monitoring the polarity of this voltage by the control circuitry, the state of magnetization of the film (i.e.

the tiny magnet on the disk surface) can be determined (i.e. whether a 0 is stored or a 1 is stored). Only changes in the magnetic field under the head can be sensed during the read operation.

Diagram:

Figure 14 shows the read/write head detail.

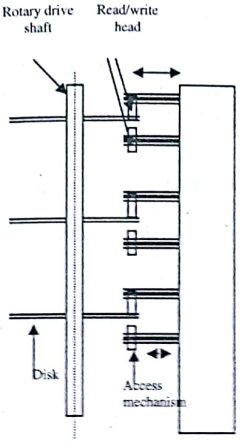


Fig 13: Mechanical structure

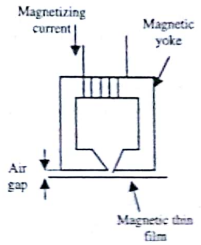


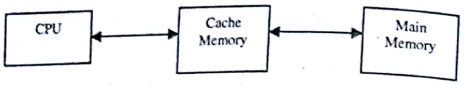
Fig 14: Read / write head detail

Storage structure and working principle of magnetic tape:

A ***magnetic tape*** is a strip of plastic coated with magnetic material. Here also the tape surfaces are divided into number of tracks and tracks are further subdivided into sectors. Bits are recorded as magnetic spots on the tape along the tracks. Read / write heads are mounted one in each head so that the data can be recorded and read as a sequence of characters. Usually seven or nine bits are recorded simultaneously to form a character along with a parity bit. Magnetic tape can be stopped, started to move forward, or inverse or can even be rewind.

4.16 Cache Memory

Cache memory is a very small but very fast memory. It is the smallest but fastest among all other memory units. A very-high-speed memory, it is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate. It lies between the CPU and the main memory unit.



Need for Cache Memory:

The speed of CPU is faster than that of main memory unit. So while accessing the main memory the CPU has to wait for the main memory to complete its job i.e. the processing speed of CPU is limited primarily by the speed of main memory. This leads to considerate loss of CPU's time. Hence cache memory is employed in computer system to compensate for the speed differences between main memory access time and the processor logic.

Structure of a Cache Memory:

The main structural difference between a cache and other memories is that caches contain hardware to track the memory addresses that are contained in the cache and move data in and out of the cache as necessary. The different parts of a cache organization are:

Tag Array: It contains the addresses of the data contained in the cache.

Data Array: It contains the data itself.

Hit/Miss Logic: This determines whether a cache hit or a cache miss has occurred.

Dividing the cache into separate tag and data arrays reduces the access time of the cache, because the tag array typically contains many fewer bits than the data array and can therefore be accessed more quickly than either the data array or a single combined tag / data array. Once a tag array has been accessed, its output must be compared to the address of the memory reference to determine if a hit has occurred. This separation of cache into tag and data arrays reduces the overall access time as hit / miss determination can be done parallely.

Diagram:

Figure 15 shows the cache block diagram.

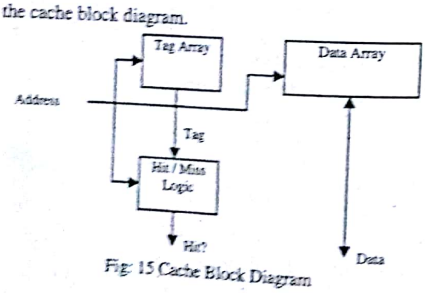


Fig: 15 Cache Block Diagram

Types of Cache Memories:

In some high-performance processors, two levels of caches may be used, L1 and L2.

L1 and L2 Cache Memories:

L1 (level 1) cache memory lies on the same chip (on-chip cache) with the processor whereas the L2 (level 2) cache memory lies external to the processor chip i.e. L2 cache lies in-between the main memory and the processor chip. L2 cache is generally implemented using Static RAMs.

Comparison between L1 and L2 cache memories:

	L1 Cache	*L2 Cache*
1.	Lies on the same chip as the processor.	Lies external to the processor chip in-between the CPU and the main memory.
2.	Smaller in size.	Larger in size.
3.	Faster.	Comparatively slower.
4.	Lower hit rate as size is small.	Higher hit rate as size is comparatively large.
5.	Number of misses is more.	Number of misses is less.

Functions of Cache Memory:

The cache memory lies in-between the slower main memory and the relatively faster CPU. The access time of cache is close to processor logic clock cycle time. The cache is used to store segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations i.e. it stores those programs and data that are executed or needed by the CPU presently or frequently. So by making programs and data available at a rapid rate, it is possible to increase the performance rate of the computer.

Expression to calculate the average access time of a processor in presence of a cache memory:

The ***average access time*** experienced by a processor is: $t_{ave} = hc + (1 - h)\ M$ where h is the hit rate, M is the miss penalty, and C is the time to access information in the cache.

Locality of Reference:

Locality of reference is the property that shows for large number of programs, references to memory at any given interval of time tend to be confined within a few localized areas in memory.

A computer program is accessed in a straight-line fashion (i.e. instructions in a program are executed sequentially) with program loops and subroutine calls encountered frequently. Now in the memory, instructions for a program are stored sequentially

generally in consecutive memory locations. When a program loop is encountered, the CPU repeatedly refers to the set of instructions in memory that constitute the loop (the instructions in a loop are stored sequentially in consecutive memory locations and hence only those locations are repeatedly accessed by the CPU). Every time a given subroutine is called, its set of instructions is fetched from memory. Thus loops and subroutines tend to localize the references to memory for fetching instructions (i.e. only those locations containing the loop instructions are repeatedly accessed and program control remains confined to those locations only).

So according to the locality of reference property, over a short interval of time, the addresses generated by a typical program refer to few localized areas of memory repeatedly, while the remainder of memory is accessed relatively infrequently.

Explanation of how the 'locality of reference' property helps to reduce the average memory access time and thus the total execution time of the program:

'Locality of reference' property refers to those few localized memory areas that are frequently accessed. So to access these locations, each time the CPU must refer to these locations and then the control must be placed to these locations and then they can be accessed. If these locations are to be accessed 'n' (say) number of times, then the same process must be repeated 'n' number of times. This is very time consuming.

Hence to avoid this, the active portions of the programs and data (i.e. the frequently accessed instructions and data) are placed in the cache memory, thus reducing the average memory access time and hence also the total execution times of the program. The cache access time is less than the main memory access time by a factor of 5 to 10. So the locality of reference property suggests that whenever an information item (instruction or data) is first needed, this item should be brought into the cache and will remain there until it is needed again and so instead of fetching just one item from the cache, several items (called block of data) residing at adjacent addresses are fetched as well.

Cache Hit:

Cache Hit (i.e. Read Or Write Hit): When The CPU Refers To Memory And Finds The Word (i.e. The Desired Address Location) In The Cache Itself, Is It Said To Produce A Cache Hit. So This Means That The Required Location Is Found In The Cache.

Cache Miss:

Cache Miss (i.e. Read or Write Miss): If the word is not found in cache, it means that it may be in main memory (or in auxiliary memory) and is known as a cache miss.

Hit Ratio:

It is the ratio of the number of cache hits divided by the total CPU references to memory (i.e. number of hits plus number of misses). Hit ratio measures the performance of the cache memory. Hit ratios generally range from 0.9 and above.

$$\text{Hit Ratio} = \frac{\text{Number of cache hits}}{\text{Number of cache hits + number of cache misses}}$$

Hit Rate:

Hit Rate is the number of hits stated as a fraction of all attempted accesses to the cache i.e. it is the rate of successful accesses to data in a cache.

Miss Rate:

Miss Rate is the number of misses stated as a fraction of attempted accesses to the cache i.e. it is the rate of unsuccessful accesses to data in a cache.

Miss Penalty:

Miss Penalty is the extra time needed to bring the desired information into the cache from the main memory i.e. on occurrence of a miss the desired data must be brought to the cache and an extra time (miss penalty) is needed for that.

Possible Measures to Improve the Hit Rate:

The possible measures to ***improve the hit rate*** are:

(a) The cache size can be made larger.

(b) The block size to be brought into the cache may be increased while keeping the total cache size constant as if all items in a larger block are needed in a certain computation. It is better to load these items all at a time as a consequence of a single miss (rather than loading several smaller blocks separately as a result of several misses).

(c) But the most practical and efficient approach is to use block sizes that are neither very small nor very large.

Possible Measures to Reduce Miss Penalty:

The possible measure to **reduce the miss penalty** is to use the load-through approach when loading new blocks into the cache. This is because instead of waiting for the desired block to get loaded in the cache, it is directly sent to the processor.

Write-Through and Write-Back:

CPU can either send a request to read information from the memory system or to write (store) information on the memory system (i.e. in the main memory). While reading information, if the required content is found in the cache memory then main memory is not involved in the transfer. But while writing information then always the cache memory is involved along with the main memory.

There are two ways to do this:

(a) Write-Through:

While updating the main memory with every memory write operation, the cache memory is also updated parallely (at a time) provided that the cache contains the required word at the specified address. So the main memory always contains the same data as the cache memory.

(b) Write-Back or Copy-Back:

In this method during a write operation only the cache memory location is updated. The main memory content remains unchanged. The location in the cache memory (i.e. the updated location) is marked by a flag. Afterwards when this specific marked word (or marked location's content) is not needed further in the cache, then it is copied to the main memory for future use.

The reason is as follows. While a word remains in the cache, it is updated several times. It does not matter whether the copy of the word in the main memory is updated parallely or not as requests (for the word) are filled from the cache itself. It is only when the word is removed from the cache that an accurate copy needs to be rewritten into main memory for further use. So in this method, both the main memory and the cache memory copies of the same word need not be updated every time simultaneously.

4.17 Types of Cache Memory Mapping Techniques

The transformation of data from the main memory to the cache memory is known as a ***mapping*** process. So mapping process deals with the various approaches of transferring the main memory information into the cache memory.

While transferring, information from main memory is transferred in ***blocks*** (i.e. multiple memory locations constitute a block) and also the cache memory locations are considered as blocks. As the size of main memory is much larger than cache memory, number of blocks in main memory is much more compared to that in cache. So in a fewer number of cache blocks, it is somehow needed to place more number of main memory blocks. This procedure of placing main memory blocks into cache blocks is mapping.

Types of Mapping Procedures:

Consider a main memory with the capacity of storing 64K words or 65,536 words. There are 4K or 4096 blocks in the main memory with 16 words in each of the blocks. A cache memory of capacity of storing 2K or 2048 words is there. Number of cache blocks is 128 with 16 words in each of the cache blocks. It is needed to map the main memory blocks, as per the need, in the cache memory blocks. Three types of mapping procedures are there. They are:

(a) Associative Mapping:

This is a very flexible mapping method. Here a main memory block can be placed into any cache memory block position. Each of the cache blocks has a *'tag'* field in them containing the address of the data. The tag field is required to identify a memory block when it is resident in the cache i.e. which of the main memory block is at currently present in a cache block. In the associative mapping, 12 tag bits are required to identify a memory block when it is resident in the cache.

As shown in the figure 16, out of the 16-bit main memory address, 4 bits are kept for the memory words i.e. with these 4 bits 16 (as $2^4 = 16$) different memory locations within a memory block can be identified. The remaining 12 bits (i.e. 12 tag bits) are used to identify any one of the 4096 main memory blocks (as $2^{12} = 4096$), placed in the 128 cache blocks. This is because at a time, a cache block can hold any one of the 4096 main memory blocks.

When an ***address*** is to be ***transferred*** from the main memory, a match between the tag fields of the CPU address and the list of the tag fields maintained by the associative cache memory are made. The working principle is simple. Firstly the tags are matched and then the particular cache block address (i.e. cache block numbers) is obtained. Hence, the search is made as per the contents (tags) and then the required address (block) is found out. The required main memory data is placed on the matched cache block.

When information is to be searched in the cache, all the tag fields of the 128 cache blocks are ***compared*** parallely with the given CPU address (i.e. tag portion of the CPU address) to see if the desired block is present. Once the desired block in the cache is found (i.e. the cache block whose tag field matched the CPU address's tag field), the required location (out of the 16 locations) can be found depending on the 4-bit word field.

Advantages of associative mapping:

(i) In this technique there is no restriction on where a main memory block can be placed in the cache. So, as a main memory block can be placed into any cache block position, the space in the cache can be used very efficiently.

(ii) Very fast method as all the tags are compared with the given CPU address (tag portion) parallely using a parallel comparator logic circuitry.

Disadvantages of associative mapping:

(i) Very complex circuitry is needed to compare all the tags parallely.

(ii) Due to more complex circuitry, the cost is also very high.

Diagram:

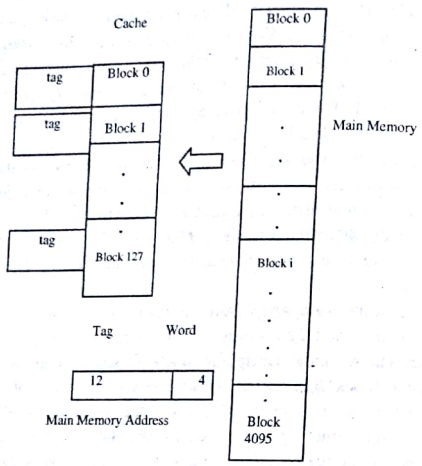


Fig: 16 Associative-mapped cache

(b) Direct Mapping:

This is the simplest way to determine cache locations in which to store memory blocks. Here also each of the cache blocks has a 'tag' field in them containing the address of the data. In this technique, block j of the main memory maps onto block j modulo 128 of the cache. In direct mapping, the placement of main memory blocks into the cache is done as per the above condition. So a cache block can hold any one of the 32 main memory blocks (as 4096 / 128 = 32). Thus, at a time, any one of the main memory blocks 0, 128, 256 etc. can be stored only in cache clock 0 and any one of the main memory blocks 1, 129, 257 etc. can be stored only in cache block 1 and so on.

So in this technique, the main memory address is considered to have three fields: the 4 bit word field with which 16 different memory locations within a memory block can be identified; the 7 bit block field which determines the cache position where to place (map) the main memory block (i.e. out of 128 cache block, which block to contain the required main memory block) and the 5 bit tag field which identifies which of the 32 blocks mapped into the cache are currently there in the cache.

[***Explanation:*** The 4096 main memory blocks are to be mapped in the 128 cache blocks. So each cache block may hold any one of the 32 (as 4096 / 128 = 32) main memory blocks at a time. So the 7-bit block field is used to select one cache block out of the 128 cache blocks that ***may hold*** the particular main memory location required by the CPU and 5-bit

tag field is used to select which of the main memory block (out of the 32 main memory blocks) is currently present in the particular cache block.]

So while storing a main memory block onto the cache, firstly based on the equation (block j modulo 128), the required cache block (out of 128 cache blocks) is selected. Then based on the tag field, which of the main memory block, out of 32 main memory blocks per cache block, will be there in a cache block at a time, is selected. And then the mapping is done.

Similarly while reading information from a cache, firstly the tag fields and block fields of each cache block are compared with the CPU address to get the required main memory block in a particular cache block. Then similarly using the 4-bit word field, the particular location (out of the 16 different locations in a block) is found out.

Diagram same as figure 16.

Diagram:

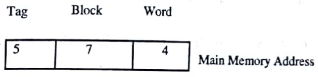


Fig: 17 Direct-mapped cache

Advantages of direct mapping:

(i) Implementation of this technique is very easy, as less complex circuitry is needed.
(ii) Cost of hardware is much less.

Disadvantages of direct mapping:

(i) This technique is not very flexible, as in it the placement of a particular main memory block is restricted to a particular cache memory block.
(ii) Slower compared to Associative mapping.
(iii) Cost of hardware is much less compared to that in Associative mapping.

(c) Set-Associative Mapping:

This is a combination of the direct and associative mapping techniques. This actually tries to solve the problems of direct and associative mapping. Here cache blocks are grouped into sets, and the mapping allows a block of main memory to reside in any block of a specific set.

Consider figure 18 below. In each set there are two cache blocks. As there are 128 blocks in the cache, hence there will be 64 sets in the cache. In this technique, block j of the main memory maps onto block j modulo 64 (number of sets) of the cache. Main memory blocks 0, 64, 128,...., 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Hence each cache block can hold any one of the 64 main memory blocks (as 4096 / 64). So the 6 bit set field is needed to determine the required cache set (out of the 64 sets) that might contain the desired block. Once the set field is matched, it means that any one of the two blocks might contain the desired main memory

location. Now, each of the blocks can hold any one of the 64 main memory blocks. So with the help of the 6 bits block field it can be found out that which of the particular main memory block (out of the 64 main memory blocks in each) is currently located in each of the two cache memory blocks in a set. So, the memory location to be mapped and to be searched, can be determined by associatively comparing the CPU address's tag field to the tags of the two blocks of the cache set (the 6 bit tag field can identify any one of the 64 locations) to check if the desired block is present.

Advantages of set-associative mapping:

(i) Reduces the contention problem of the direct method by giving a few choices for block placement as this technique allows a main memory block to reside in any block of a specific cache set (i.e. here 32 main memory blocks can be places in any of the two cache blocks unlike in direct mapping method where 32 main memory blocks can be placed in only one cache block).

(ii) Decreases the hardware cost by reducing the size of associative search and thus by reducing the complexity of the needed circuitry.

Diagram:

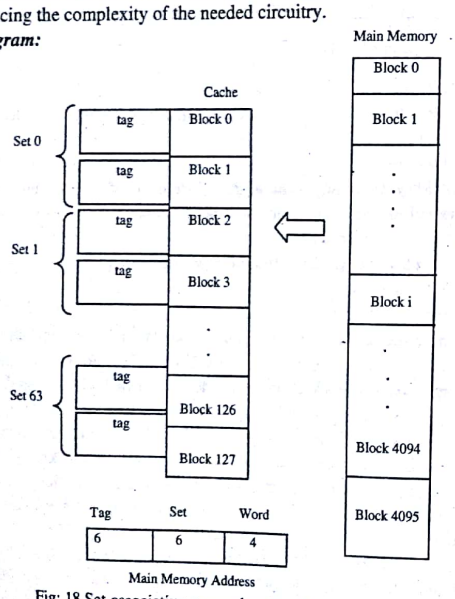


Fig: 18 Set-associative-mapped cache with two blocks per set

4.18 Replacement Algorithms

Sometimes during cache mapping, it may happen that more than one main memory block is mapped onto a given cache block position. In that case contention (conflicts) may arise for that position if the cache is full or even if it is not full. Suppose that block 1 and block 129 (say) may be needed to get transferred to the same cache block 1 (fig 17). To resolve this contention the new block may be allowed to overwrite the currently resident block using a replacement algorithm.

It may also happen that to bring a new block into the cache that is full, an existing block must be replaced. Replacement algorithms are needed in such cases to select the block to be replaced. This decision of which block to replace is taken by the cache controller and the memory management unit (MMU).

Explanation of the Replacement Algorithms:

The decision of the right block getting replaced is an important system performance factor. The objective is to keep the blocks in the cache that are likely to be referenced (needed by the CPU) in the near future. As per the property of 'locality of reference' it can be said that the blocks that have been referenced recently will be referenced again soon. So when a block is to be overwritten, it is needed to overwrite the one that has gone the longest time without being referenced. This block is called the ***least recently used (LRU) block*** and the technique is called the ***LRU replacement algorithm***.

While using the ***LRU algorithm***, the cache controller must track references to all blocks as computation proceeds. A counter is used to track the references to each block. Depending on the number of references tracked by each counter (i.e. by the number of hits and misses) it is decided which block is used the longest time back (i.e. the least recently used block). And that block is then replaced. Though very effective LRU algorithms can lead to poor performance when accesses are made to sequential elements of an array that is slightly too large to fit into the cache.

Other replacement algorithms are however not as effective as LRU. However using an algorithm that randomly chooses the blocks to be replaced is also getting effective.

4.19 Virtual Memory

Virtual memory is a technique that allows the execution of processes that may not be completely in main memory. This concept used in some large computer systems permit the user to construct programs as though a large memory space is available. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

When a program does not fit completely into the main memory, parts of it not currently being executed are stored in the auxiliary memory (say, the disks) and are brought into main memory automatically, as needed, to get executed. This automatic movement of programs and data into the physical main memory when they are requested for execution is the virtual memory technique.

Diagram:

The figure 19 below shows the organization that implements virtual memory.

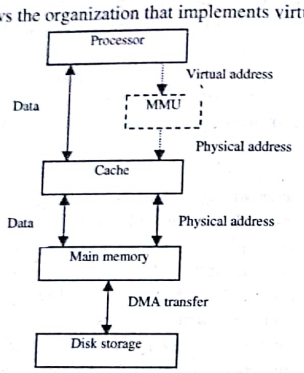


Fig: 19 Virtual memory organization

Advantages of Virtual memory:

The advantages are as follows:

(a) Programs can be larger than physical memory size and users would be able to write programs for an extremely large virtual address space, simplifying the programming task.

(b) This technique separates the logical memory as seen by the user from physical memory and thus memory storage limitations do not arise.

(c) Because each user program could take less physical memory, more programs could be run at the same time.

(d) Less input-output (I/O) activities would be needed to load each user program into the main memory, so each user program would run faster.

Address Space:

Address space is the set of addresses (i.e. set of virtual or logical addresses) used by a programmer. The address used by a programmer is called ***virtual*** or ***logical address***. Hence the binary addresses generated by the CPU for either instructions or data are the

logical addresses. An address space may also be termed as ***virtual*** or ***logical address space***.

Memory Space:

Memory space is the set of main memory addresses (i.e. set of main memory locations). An address in main memory (that is, the one loaded into the memory address register (MAR) of the main memory) is called ***location*** or ***physical address***.
However in most computers the address and memory spaces are identical.

Relationship between Address and Memory Space in a Virtual memory System:

In virtual memory technique, programs larger than the size of the physical memory may be executed. Here users can construct programs as though a very large memory space was available, equal to the totality of all the memory units in the memory system.
The address generated or issued by the CPU for instructions or data is brought from the virtual memory to the main memory and are then executed. So programs and data are transferred to and from virtual memory and main memory based on demands imposed by the CPU.
The figure 20 below shows the relationship between the two. To execute a program (instructions and data) it must be brought to the main memory first. If the CPU is currently executing program 1, then it should be brought (along with the associated data 1, 1) to the main memory. A mapping process does this transfer of instructions and data to the main memory from the virtual memory. The virtual memory address space is much larger compared to the physical memory space.

Diagram:

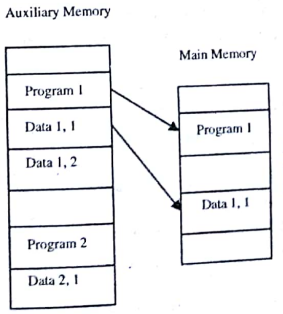


Fig: 20 Relation between address and memory space in a virtual memory system

'Mapping' of Virtual Address to a Physical Address:

Each address referenced by a CPU goes through an address mapping from the virtual address to a physical address in the main memory. A virtual memory system provides a mechanism for translating program-generated (virtual) addresses into correct main memory locations. This translating procedure, done dynamically, is called ***mapping*** of addresses and is carried out by a combination of hardware and software components. So mapping gives an idea of how to bring the required data to the main memory (from the disk) and where to place the required data in the main memory.

Need for Mapping of Addresses:

To execute programs (instructions and data), they must be physically present in the main memory. If a virtual address issued by the CPU refers to a part of the program or data space that is currently in the physical memory, then those contents are accessed immediately in the main memory. Else, if the referenced address is not in the main memory, its contents must be brought into a suitable location in the main memory before they can be used.

Pages and Frames:

The physical (main) memory is broken down (i.e. memory space is divided) into groups of equal size called blocks or ***frames*** while logical (virtual) memory is broken down (i.e. address space is divided) into groups of equal sizes called ***pages***. Pages and frames must be of equal sizes.

Need for Pages and Frames:

Such divisions are done to simplify the implementation of memory table for address mapping. Programs are considered to be split into pages. When a program is to be executed, its pages from the auxiliary memory are loaded (mapped or translated) into any available memory frames.

Mapping a Virtual Address to a Physical Address:

In the main memory, portions of programs and data may not be in contiguous locations and empty spaces (to bring programs from the disk) may be available in scattered locations in memory. With help of mapping it is decided exactly where to place the new data to be brought in from the disk. The mapping hardware organization consists of a virtual address register that contains the virtual address provided, a memory mapping table that will keep track of the available frames and map them as needed to the different pages, a memory table buffer register to hold the frame number where the page is to be mapped, a main memory address register that holds the physical address and the main memory. The figure 21 below shows the hardware organization.

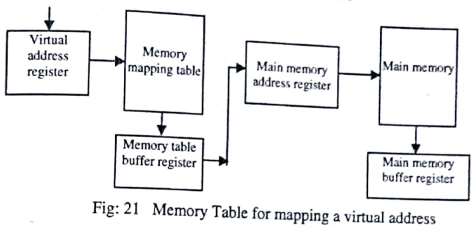


Fig: 21 Memory Table for mapping a virtual address

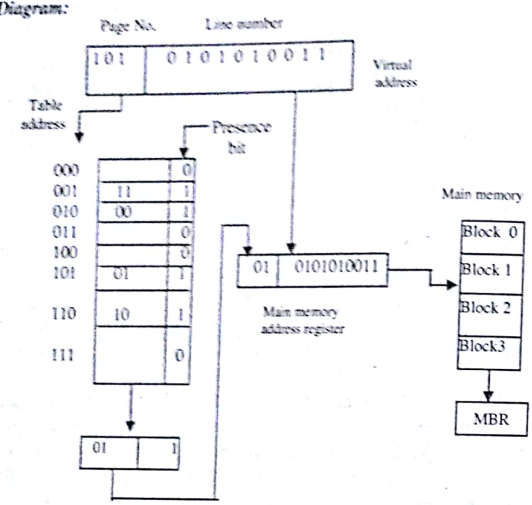
The mapping from address space to memory space is facilitated if each virtual address is considered to be represented by two numbers: a ***page number address*** and a ***line within the page***. Suppose the address space capacity is 8K and the memory space capacity is 4K. Splitting each into groups of 1K words, 8 pages and 4 blocks/frames are obtained respectively. Now $8K = 2^{13}$ and $1K = 2^{10} = 1024$ words. So a virtual address has 13 bits (as $8K = 2^{13}$) and each page has 1024 words.

Let the ***high-order three bits*** of the virtual address specify one of the 8 pages and the ***low-order 10 bits*** give the line address within the page. The page number to block number mapping is required. As shown in figure 22, the memory-page table consists of 8 words, one for each page. The virtual address has got two parts, the page number and the line number. The page table address denotes the page number and the content of the word gives the block number where that page is stored in main memory.

In the table pages 1, 2, 5, and 6 are shown to be available in main memory in blocks 3, 0, 1, and 2 respectively. The ***presence bit*** in each location, if 1, indicates that particular page has been transferred from auxiliary memory to main memory and, if 0, indicates that particular page is not available in main memory. The content of the word in the memory page table at the page number address is read out into the memory table buffer register. If the presence bit is a 1, the block number thus read is transferred to the two high-order bits of the main memory address register. The line number from the virtual address is transferred into the 10 low-order bits of the memory address register (main memory address space has $4K = 2^{12}$ words i.e. 2 bit block number and 10 bit line number). A 'read' signal to main memory then transfers the content of the word to the main memory buffer register ready to be used by the CPU.

However, if the presence bit in the word read from the page table is 0, it signifies that the content of the word referenced by the virtual address does not reside in main memory. Then a '***page fault***' occurs if that page is needed in the main memory and it is needed to fetch that page from the disk to the main memory to resume further computation.

Diagram:



Memory page table

Fig: 22 Memory table in paged system

Related Questions & Answers

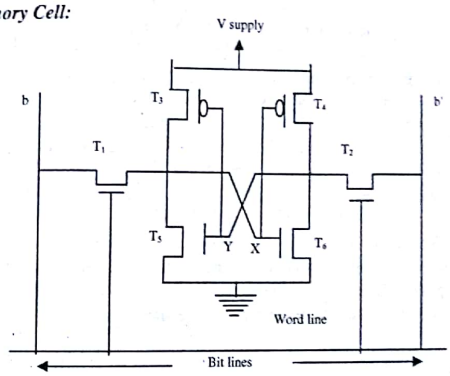
Question 1

What is a bipolar storage cell. With the help of a diagram explain the working principle of a CMOS memory cell.

Answer:

In bipolar technology, both the electrons and holes are the carriers simultaneously. Here both electrons and holes move simultaneously.

CMOS Memory Cell:



As shown in the figure, transistors pairs (T_3, T_5) and (T_4, T_6) form the inverters in the latch. Here the information stored in X (i.e. if either of 1 or 0 is stored in X then it will remain as it is) and Y is retained as it is, due to the latch.

To read: To read the content of the 'nth' cell:

Steps:

(a) The required word line is enabled.

(b) This turns the transistor T_1 on.
So the information in X comes to bit line b.

(c) Similarly information in Y comes to b'

(d) Information from the respective bit line goes to the sense / write circuit.

To write:

Steps:

(a) The required word line of the cell to be written is enabled.

(b) Information to be stored comes to the cell from the sense / write circuit via the bit lines.
If '1' is to be stored, then bit line 'b' must contain 1 (the b' will hold a 0 then). If '0' is to be stored, the bit line b must contain 0 (b' will then hold 1).

(c) So, T_1 and T_2 will get enabled (on enabling the word line) and closed paths are formed. Contents of b and b' will thus go to X & Y respectively.

To retain the stored information in X & Y:

Suppose X has got '1' in it i.e. X is high and Y has got '0' in it i.e. Y is low.

(a) Consider the case when X = 1 (i.e. content of X is 1):

T_3 will be enabled and will be high (will form a closed path). So direct connection between Y and the ground is made. Hence, Y becomes low i.e. it holds 0 (but Y already holds a 0 in it). So it can be said that Y retains its value/state).

Also as X = 1, T_4 gets low or disabled and acts as an open path.

Hence there is no connection between V_{supply} (i.e. high voltage) and Y. So Y remains at 0 (i.e. Y retains its state).

(b) Consider the case when Y = 0:

T_3 gets enabled (low state) and acts as an open circuit. So X remains high (as no contact occurs between X and ground).

Also, as Y is low (0), T_3 gets enabled (1), and so direct contact between X and V_{supply} occurs (i.e. X remains high).

So, this is how X and Y retain their states due to the latch.

Question 2

How to perform read / write operations in a memory chip on a particular row of cells?

Answer:

Suppose in a memory chip there are numerous memory cells arranged in the form of an array. Say there are 'n' rows of cells and in each row there are 'm' cells. So the n * m cells are arranged in n rows (rows are numbered 1 through N) and m columns (columns are numbered 1 through M). Now the CPU issues a request to read the content of, say, the cells in the 1st row. The steps to ***read*** the cell's content are the following:

(a) CPU issues the request to read the content of the cells in the 1st row of the memory chip.

(b) Request reaches the memory chip via the memory bus. The address (i.e. the number) of the desired row of cells is given in the request.

(c) The address from the CPU is then decoded by the chip's address decoder to enable the desired row of cells out of the 'n' rows of cells.

(d) The contents of the enabled cells in the particular row are then 'read' by giving a 'read' signal via the 'control' terminal.

(e) The contents of the cells then are transferred to the data bus via the 'sense' terminal and thus go to the CPU.

The steps to ***write*** data (i.e. 0 or 1) in a cell are also the same only that the control terminal will issue a write signal and the data-in (write) terminal will be enabled to write the data in the specific cell.

Question 3

Explain the memory hierarchy pyramid, showing both primary and secondary memory in the diagram and also explain the relationship of cost, speed and capacity.
[WBUT 2004, 2005, 2009]

Answer:

Based on the need of the hierarchical memory organization, the different memories are arranged in the form of a pyramid which gives a clear vision of the existing relationships between the different factors (like cost, speed, storage capacity, access time etc.) among the different memories in the hierarchical pyramid. The memory hierarchy pyramid (figure 23), on top of which are the registers while at the bottom lie the magnetic tapes, is as shown below:

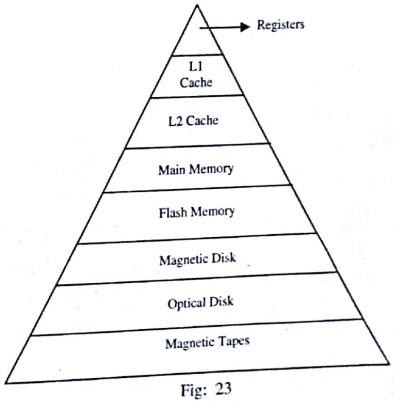


Fig: 23

Question 4

Why is the 'bootstrap loader' program stored in ROM and not in RAM?

Answer:

The ROM portion of main memory is needed for storing the initial 'bootstrap loader' program Since RAM is volatile, its contents are destroyed when power is turned off and it is not feasible to write the contents in the RAM each time after the power is turned on. Hence ROM being a nonvolatile memory, (i.e. its content remain unchanged even after power is turned off) it retains its content.

The startup of a computer consists of turning the power on and starting the execution of an initial program. So when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader. This bootstrap loader then gets the rest of the operating system from the disk to main memory and thus prepares the computer for use.

Hence this entire process is possible only if the bootstrap loader program is kept in ROM (as if it would have been kept in RAM then it was needed to first load the bootstrap program itself in the RAM each time the computer is turned on).

Question 5

Show the operation of a RAM chip with the help of a function table.

Answer:

Function Table for the operation of the RAM chip:

CS 1	$\overline{CS\ 2}$	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	0	Read	Output data from RAM
1	1	X	X	Inhibit	High-impedance

Fig: 24

A particular *RAM* chip, in a board, can operate *only* when

$CS1 = 1$ & $\overline{CS2} = 0$. For all other CS1 & $\overline{CS2}$ conditions, the RAM chip will not operate (i.e. memory function will be *inhibit*) & so no signal will flow through the data bus & it will remain in 'high impedance' (open-circuit) state.

In case of ***write operation:***

$CS1 = 1$ & $\overline{CS2} = 0$ and WR = 1 i.e. information can be written on the chip if the two control lines are 1 and 0 respectively and the write line is enabled.

In case of ***read operation:***

$CS1 = 1$ & $\overline{CS2} = 0$ and RD = 1 i.e. information can be read from the chip if the two control lines are 1 and 0 respectively and the read line is enabled.

In all other cases, the chip will not function.

Question 6

What are the uses of the control or chip select lines?

Answer:

A main memory system may consist of multiple memory chips (either ROM or RAM). Each chip has multiple memory cells. The memory chip select lines, select the particular memory chip (out of multiple memory chips on a memory system board) that contains the specific row of memory cells (locations) whose contents are to be read or to be written.

Question 7

What does address decoding mean?

Answer:

Address decoding means finding out the specific (desired) memory locations to be accessed based on the address sent by the CPU.

It's a two-phase process. Firstly, CPU places the address of the specific memory location or cells to be read or to be written on the address bus. The external address decoder of the memory system (board of memory chips) decodes the CPU address to find out the particular chip (out of the many memory chips on the board) that holds the desired address needed by the CPU.

Then, once the particular memory chip is selected, the CPU address is again decoded by the internal address decoder (on the specific memory chip) to find out the particular row of memory cells (location or word), on the chip, to be accessed.

The decoding is done with the help of decoders. To select a chip, its particular chip select line(s) is enabled by the output of the address decoder i.e. based on the output of the address decoder the 'CS' line(s) of the particular chip is enabled.

Question 8

Explain the concept of memory map.

Answer:

A memory system is made up of multiple RAM and ROM chips. The entire memory capacity is divided and assigned (by the designer), as required, among the different RAM and ROM chips. The designer of the computer system calculates the amount of memory required for the particular application and assigns it to the different RAM and (or) ROM chips. Then the interconnection between memory and processor is established. A table is maintained that specifies the memory address assigned to each chip i.e. if, suppose, a total of 40 GB of memory is assigned for a particular application, then the designer may divide this capacity as 30 GB to RAM chips and remaining 10 GB to ROM chips (even the entire information may be stored in the RAM chip if the ROM is full). This table, called a ***memory address map***, is a pictorial representation of assigned address space for each chip in the system i.e. what are the different address locations and how many address locations that each chip may hold.

Example:

Suppose a total of 1024 bytes of memory are needed for a particular application. The designer assigns 512 bytes to the RAM and remaining 512 bytes to the ROM. Explain the memory address mapping for the particular application (desired address range to be mapped, is from 0000 to 03FF) provided there are four 128 x 8 RAM chips and one 512 x 8 ROM chip.

Solution:

The basic aim of the memory mapping is to divide or assign the given address range (i.e. 0000 to 03FF) of the memory system among the five chips.

There are four 128 bytes (i.e. 128 x 4 = 512 bytes) RAM chips and one 512 bytes of ROM chip. So each of the RAM chips will have 7 address lines (as $2^7 = 128$) and the ROM chip will have 9 address lines (as $2^9 = 512$). So in all there will be 16 lines in the address bus.

Also as the addresses assigned are in hexadecimal and each address is of four bits, hence to convert them in binary, 16 (4 * 4) different binary bits (i.e. 16 different address lines) are required.

Always, a designer tries to utilize a single line to do multiple specific jobs if possible i.e. aim is to utilize minimum number of lines to do maximum jobs. So, out of the required 16 address lines (each line will carry one bit), lines 1 – 7 will carry RAM addresses, lines

1 – 9 will carry ROM addresses. Hence, lines 8 and 9 carry ROM addresses as well as selects among the four RAM chips. Again line 10 selects between the ROM and the RAM chips. Lines 11 – 16, though, are not used for any specific purpose but will have to be considered for the memory mapping.

The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip. Now, as lines 11 – 16 are not utilized to do any specific purpose hence they are marked as 0 in the map. Line 10 when 0 will indicate a RAM chip and when 1 will indicate a ROM chip. Lines 9 and 8 will give any one out of the four different combinations to select a particular RAM chip at a time out of the four different RAM chips. Also lines 9 and 8 will be marked as 'x' (don't-care) to indicate that they are also carrying ROM addresses. Lines 1 – 7 will be marked as 'x' to indicate that they are carrying, both, RAM and ROM addresses and they can signify any different combinations. The mapping is shown below:

		16			13										
components	Hexadecimal addresses	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM 1	0000–007F	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	X
RAM 2	0080–00FF	0	0	0	0	0	0	0	0	1	x	x	x	x	x	x	X
RAM 3	0100–017F	0	0	0	0	0	0	0	1	0	x	x	x	x	x	x	X
RAM 4	0180–01FF	0	0	0	0	0	0	0	1	1	x	x	x	x	x	x	X
ROM	0200–03FF	0	0	0	0	0	0	1	x	x	x	x	x	x	x	x	X

So the addresses assigned in the respective chips are given in their hexadecimal forms in the memory address map. The address bus lines are subdivided into groups of four bits so that each group can be represented with a hexadecimal digit i.e. in each case group of four lines are taken to represent one hexadecimal digit.

Hence, the initial address or the address of the first location in the first RAM chip is 0000 and the address of the final location in the first RAM chip is 007F. Similarly address range of the second RAM chip is from 0080 to 00FF. Similar is in the case of other chips also.

Question 9

What is the use of the latch in the static RAM cell?

Answer:

It is due to the latch (i.e. 2 cross-connected inverters) that the information stored at X and at Y are retained as it is till the power supply goes off.

This is because, content of X (say 1), if inverted by the inverter 1, will give '0' at Y. Again when this 0 will get inverted (by inverter 2), the same '1' will be there at X. Same in the case of Y also i.e. same values at X and Y will be retained owing to the latch.

Question 10

What are the methods to improve the performance of a memory system?

Answer:

Some of the very common methods that designers follow to improve the performance of the memory systems are:

(a) Pipelining:

Here multiple memory systems can be arranged in a pipeline to overlap execution which in turn improves the execution speed and thus throughput.

(b) Parallelism:

Here memory systems are so designed to support multiple memory references in parallel.

(c) Precharging:

This method requires the memory circuitry to get prepared or precharged for the next memory access. This is done in the idle time between two memory accesses. Precharging the memory system at the end of each memory access operation improves the memory system performance.

Question 11

What is refresh time of a DRAM chip?

Answer:

DRAMs or dynamic RAM chips need to get recharged or the charge in such chips need to get refreshed periodically because in such chips the stored charge gets leaked. Refresh time in DRAM chips indicates the time period that a row of cells in the chip can maintain its charge or the time period the row of cells can remain without getting refreshed before it is in danger or losing its contents.

Question 12

What are synchronous and asynchronous DRAMs?

Answer:

Synchronous DRAM:

Those dynamic RAM systems the operations of which are directly synchronized with a clock signal are known as synchronous DRAMs or SDRAMs.

Asynchronous DRAM:

These are the memory devices whose timings are asynchronously controlled. In such DRAMs, the timing is governed by specialized memory controller circuit that provides the needed control signals, Row Address Strobe (RAS) and Column Address Strobe (CAS).

Question 13

Explain why every computer system is associated with a set of general purpose registers. **[WBUT 2003]**

Answer:

The general-purpose registers (R1, R2, …., Rn) are used for storing any kind of data temporarily during any processing. So computers must give a set of such registers to store any kind of general-purpose (not specific) information.

Question 14

Explain why a given (Infix) arithmetic expression needs to be converted to Reverse Polish Notation (RPN) for effective use of a stack organization. **[WBUT 2003]**

Answer:

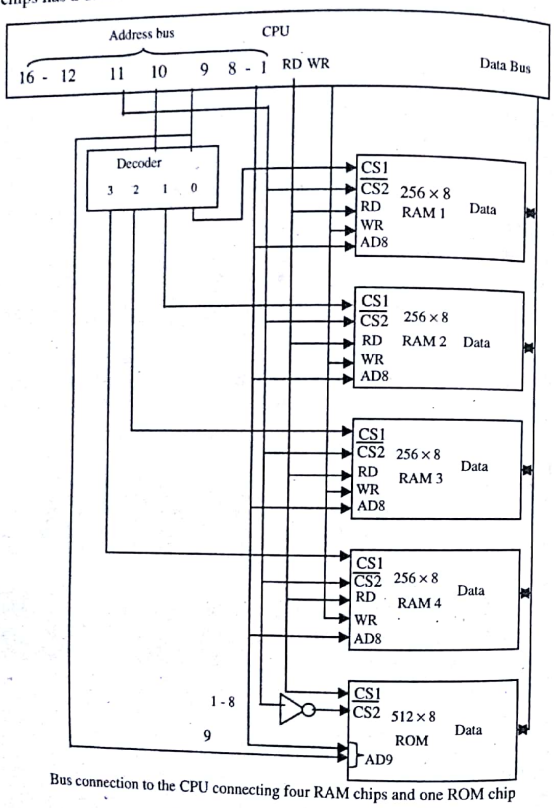
Reverse Polish Notation (RPN), combined with a stack arrangement of registers, provides the most suitable way for evaluating arithmetic expressions. This is because if the expressions are given in infix notation, then the computer needs to scan the given expression multiple of times according to operator priority. This is a time consuming process. To avoid this RPN is used, where only a single scan is needed as the expressions are already arranged as per the operator priority.

Question 15

Show the bus connections with a CPU to connect four RAM chips of size 256 X 8 bit each and a ROM chip of 512 X 8 bit size. Assume the CPU has 8 bit data bus and 16 bit address bus. Clearly specify generation of chip select signals. **[WBUT 2006, 2010]**

Answer:

The figure 25 below shows the required bus connection between CPU and four 256 X 8 bit RAM chips and one 512 X 8 bit ROM chip. The decoder has four bits. The RAM chips have 8 bit address bus each whereas the ROM chip has 9 bit address bus. Each of the five chips has a data bus of 8 bits. CS1 and $\overline{CS2}$ are select lines.



Bus connection to the CPU connecting four RAM chips and one ROM chip

Fig: 25

Question 16

Consider a situation where there are two ADD instructions. One is part of a stack-based instruction set architecture and the other is part of a general-purpose instruction set architecture. Which one do you think is smaller and why?

Answer:

The stack-based ADD instruction is smaller than the other one as it requires fewer bits to encode in comparison to the other one. This is because, in case of a stack-based instruction set architecture, the ADD instruction does not include any explicit operands and hence this does not require the machine code to have any bit representing which operands to add.

On the other hand, a general-purpose instruction set architecture requires that all the register operands should be specified in each instruction. For example, if there are sixteen possible registers, to specify each register, an ADD instruction would require four bits.

Question 17

How will you design a memory system with Small memory units and using horizontal and vertical expansion schemes? Explain with suitable examples.

Answer:

A small memory unit is a small array of memory cells organized as an array of N words of b-bit per word (N x b array). A number, of, say M = KL, such small memory units may be interconnected to form a larger array, say, of KN x Lb, of memory cells. K small, memory units should be placed in each column and L small memory units should be placed in each row.

Question 18

Convert the given arithmetic expression to RPN
Notation: A*B+A*(B*D+C*E) **[WBUT 2003]**

Answer:

The solution is: AB * ABD * CE * + * +

Question 19

Explain the basic objective of a memory hierarchy in a Computer system. **[WBUT 2003]**

Answer: Refer to section 4.5.

Question 20

With a suitable diagram explain the working structure of a RAM cell. What is the working principle of writing and erasing some contents from a memory cell?

Answer:
As it is not clearly mentioned whether the RAM cell is of static or dynamic type hence either of the two will do. So refer to the working of the two discussed above.

Question 21

Why a DRAM cell needs refreshing? [WBUT 2004, 2006]

Answer:
A dynamic RAM cell loses its stored information in a very short time (a few milliseconds) even though the power supply is on.
Information is stored in a dynamic memory cell in the form of charge on a capacitor and this charge can be maintained only for a very few milliseconds. As the charge on the capacitors leak away as a result of normal leakage, the capacitor gets turned off after the few milliseconds. So, to retain the cell information for a much longer time, the cell's content must be periodically refreshed to restore the capacitor charge to its full value.

Question 22

Some of the input wires to a Main Memory (or "RAM") are called address wires. Explain the purpose of the address wires and how they are used.

Answer:
Main Memory consists of a large number of "locations." Each location is a memory circuit that can hold a binary number. These locations are numbered 0, 1, 2, 3,..., and the number of a location is called its "address." At a given time, only one location in memory can be used for storing and reading data. The purpose of the address wires is to tell main memory which location it should make available. The pattern of ON/OFF values of the address wires can be interpreted as a binary number. This number is the address of the desired location. The contents of that location are visible on the memory's data-out wires. If a value is loaded into memory, the value goes into the location whose address is given by the number on the address wires.

Question 23

The terms location and address are related to a computer's main memory (or RAM). Explain these two terms. What is the difference between them?

Answer:
The RAM consists of a sequence of locations. Each location is a memory unit that can hold a binary number. The locations are numbered sequentially, and the number that corresponds to a particular location is called the address of that location. Thus, locations are actual, physical spots in memory, and addresses are numbers that are used to pick particular locations out of all the possible locations.

Question 24

What are the main characteristics of a RAM?

Answer:
The main characteristics of a RAM are:
a) The RAM is divided into fixed-lengths entries, which are also called cells.
b) All the cells are uniquely addressable with each having their own unique numeric identification number.
c) For all the cells in the entire RAM chip, the access time is the same.

Question 25

What is Manchester / Phase encoding?

Answer:
Phase encoding is a simple scheme of transforming data bits into electrical signal where the signal during the recording interval for each data bit has a transition (either +ve to –ve or vice-versa) in the middle. This helps in generating the clock frequency that was used in recording.
So in this scheme, changes in magnetization occur for each data bit at the midpoint of each bit period, thus providing the clocking information.

Diagram:

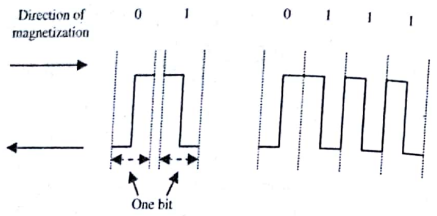


Fig: 26 Bit representation by phase encoding

Question 26

What are the advantages of Winchester Technology?

Answer:

These are the following:

(a) Winchester disks have a larger storage capacity for a given physical size compared to unsealed units as dust particles are absent here and hence more data can be stored.

(b) Data integrity tends to be greater in sealed units where the storage medium is not exposed to contaminating elements.

Question 27

Describe the storage structure of a Bipolar storage cell & explain the reading & writing operations on the cell. Give a suitable diagram. **[WBUT 2003]**

OR,

Explain the reading and writing operations of a basic Static MOS cell. **[WBUT 2004, 2007]**

OR,

Explain the working principle of a static MOS cell. **[WBUT 2005]**

Answer: Refer to section 4.11 and question 1.

Question 28

What is a 'disk controller'?

Answer:

Disk controller is the electronic circuitry that controls the operating of the entire disk system. Either the disk controller is implemented as separate module, or it may be

incorporated into the enclosure that contains the entire disk system. It also provides an interface between the disk drive (the electromechanical mechanism that spins the disk and moves the read/write heads) and the system bus (i.e. the bus that connects it to the rest of the computer system).

Question 29

Explain with the help of a diagram, what happens when CPU wants to access data from the memory system.

Answer:

The entire memory access procedure is as follows:

Steps:

(a) CPU places the address of a particular memory location to be accessed in the address bus. CPU, however, does not know about any other memory units except the main memory. So it generally gives a main memory location address.

(b) In the memory system, firstly the cache memory is searched for that particular address location. If it is found in the cache, then the location contents are directly sent to the CPU via the data bus.

(c) If the particular location is not found in the cache, then the main memory is searched for that address. If found, its contents are sent to the CPU via the data bus. Then a copy of the contents (and copy of the contents of other related locations i.e. block of contiguous related address location contents is sent to the cache for future use.

(d) If the required address location is not found even in the main memory, then the auxiliary memory locations are searched for the address. Finding it, the contents are send to the main memory (copy is stored in the cache for future use) and then to the CPU.

(e) The ***memory-management unit (MMU)*** controls this entire procedure inside the memory system.

Diagram:

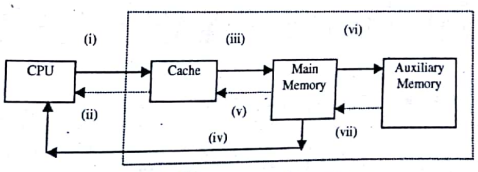


Fig: 27

Explanation of the Diagram:

(i) CPU placing the required address on the address bus.
(ii) If required location is in cache, then its content is directly sent to CPU.
(iii) Else, control passes to main memory and it is searched for the address.
(iv) If found, then content is sent to the CPU.
(v) A copy of the content (and its related contents) is stored in cache for future use.
(vi) Else, control passes to the auxiliary memory and it is searched.
(vii) Finding the content, it is sent to the main memory and then to the CPU. Again a copy is sent to the cache.

Question 30

What is Cache memory? Explain how cache memory increases the performance of a computer system. [WBUT 2003]

OR,

What is cache memory? How does it increase the performance of a computer? What is hit ratio? [WBUT 2008]

OR,

What is locality of reference? Explain the concept of cache memory with it. [WBUT 2009]

Answer: Refer to section 4.16.

Question 31

Why hierarchical memory organization is needed? [WBUT 2004]

Answer: Refer to section 4.5.

Question 32

How information is stored and received from cache memory?

Answer:

Procedure of storing and reading of information on and from the cache is the same as that in the main memory. Cache has also got address register, data register and memory locations just like the main memory and they function in the same way as in the main memory while reading or writing information against a CPU request. Only difference is that the memory cells used in cache are ***static*** in nature and also the number of locations or number of cells in each location is much less compared to that in the main memory.

But a mapping of information must be there between the main memory and the cache memory. Before information are accessed by the CPU directly from the cache, they must be mapped (transferred) in to the cache from the main memory.

Question 33

Given the following determine the size of the sub fields (in bits) in the address for Direct Mapping, associative and set associative mapping cache schemes: We have 256 MB main memory and 1 MB cache memory.

a) The address space of this processor is 256 MB.
b) The block size is 128 bytes.
c) There are 8 blocks in a cache set. [WBUT 2004, 2007]

Answer: Refer to Example 24 in Work-Out Example.

Question 34

What are the differences between real and virtual memory?

Answer:

Differences between real and virtual memory:

Sl. No.	Real Memory	Sl. No.	Virtual Memory
1.	Real or physical in existence.	1.	Virtual or conceptual in existence.
2.	Size limitation is there.	2.	No such size limitation.
3.	Generally the main memory constitutes the real memory.	3.	It is assumed to be constituted by the auxiliary memory, cache memory and also main memory.
4.	Speed is much faster.	4.	Speed is much slower.
5.	Storage capacity is limited.	5.	Unlimited storage capacity.
6.	While mapping, the real memory space is divided into frames.	6.	Virtual memory space is divided into pages.

Question 35

Briefly explain the two 'write' policies: write through and write back for cache design. What are the advantages and disadvantages of both the methods? [WBUT 2004, 2006, 2007, 2009, 2010]

Answer:

CPU can either send a request to read information from the memory system or to write (store) information on the memory system (i.e. in the main memory). While reading

information, if the required content is found in the cache memory then main memory is not involved in the transfer. But while writing information the cache memory is always involved along with the main memory.

There are two ways to do this i.e. there are two 'write' policies:

(a) Write-Through:

While updating the main memory with every memory write operation, the cache memory is also updated parallely (at a time) provided that the cache contains the required word at the specified address. So the main memory always contains the same data as the cache memory.

(b) Write-Back or Copy-Back:

In this method during a write operation only the cache memory location is updated. The main memory content remains unchanged. The location in the cache memory (i.e. the updated location) is marked by a flag. Afterwards when this specific marked word (or marked location's content) is not needed further in the cache, then it is copied to the main memory for future use.

The reason is as follows: While a word remains in the cache, it is updated several times. It does not matter whether the copy of the word in the main memory is updated parallely or not as requests (for the word) are filled from the cache itself. It is only when the word is removed from the cache then an accurate copy needs to be rewritten into main memory for further use. So in this method, both the main memory and the cache memory copies of the same word need not be updated every time simultaneously.

Both the policies have their own advantages and disadvantages, which are as follows:

Advantages of write through policy:

1. Simple and easy to implement and hence is the most commonly used cache-write method.
2. Main memory and cache memory always contain the same data in them.
3. Effective in DMA transfers as the I/O device communicating with main memory always receive the most recent updated data from the main memory.

Disadvantages of write through policy:

1. Slow (time consuming), as always two memories (cache and main memory) need to get updated simultaneously.
2. This policy will not work if the specified address location in the cache memory does not hold the required word to be updated.

Advantages of write back policy:

1. Faster than the previous policy as cache and main memory locations do not get updated simultaneously with every write operation with only the cache memory

getting regularly updated and the final copy of the updated word gets stored at the main memory (finally).

Disadvantages of write back policy:

1. In a write back policy, data (modified or not) is written to the main memory finally. Now suppose if the data is not modified at all, then the same data (unmodified) will be again written to the main memory i.e. same data will get overwritten in the main memory. But this is time consuming and hence acts as an overhead.

Question 36

Give two differences between tape drive and magnetic disk.

Answer:

The two differences between tape drive and magnetic disk are as follows:

	Magnetic Disk	Tape Drive
1.	Circular plate constructed of metal.	This is a metallic strip of plastic.
2.	Supports access mechanism of any type.	Supports sequential access mechanism.

Question 37

Explain the difference between full associative and direct mapped cache mapping approaches. **[WBUT 2004, 2007, 2009]**

Answer:

Differences between full associative (i.e. set-associative) and direct mapped cache mapping approaches are as follows:

	Direct mapping	Full-Associative mapping
1.	Suffers from contention problem as provides few choice of block replacement.	Choice of block replacement is more and hence suffers much less from contention problem.
2.	Slow process.	Much faster compared to direct mapping technique.
3.	Less expensive (hardware).	Much less expensive than direct mapping.

Question 38

What is a translation look-aside buffer (TLB)? **[WBUT 2004]**

Answer:

This is a special, small, fast associative memory used in segmented-page mapping. This buffer holds the most recently referenced page table entries. In general, while mapping a

logical address to physical address, the two mapping tables remain stored in two separate small memories or in main memory. So this increases the number of memory references while accessing data and hence time taken is more. This time penalty may be avoided by using TLB which holds the value of the given block along with its segment and page numbers. So any references to that block can be directly taken from the TLB itself.

Question 39

Explain the terms location and address. What is the difference between them?

Answer:
The main memory (generally RAM) consists of a sequence of locations. Each location is a memory unit holding a binary number. The locations are numbered sequentially, and the number that corresponds to a particular location is called the address of that location. Thus, locations are actual, physical spots in memory, and addresses are numbers that are used to pick particular locations out of all the possible locations.

Question 40

Explain Dynamic Memory Allocation.

Answer:
Dynamic memory allocation is the allocation of memory storage for use in a computer program during the runtime of that program. It is a way of distributing ownership of limited memory resources among many pieces of data and code. A dynamically allocated object remains allocated until it is deallocated explicitly, either by the programmer or by a garbage collector.
The main problem for most dynamic memory allocation algorithms is to avoid both internal and external fragmentation while handling both allocation and deallocation efficiently.

Question 41

Explain the concept of virtual memory. [WBUT 2004, 2007]

Answer: Refer to section 4.19.

Question 42

Explain Garbage Collection. What are its advantages?

Answer:
Garbage collection is a form of automatic memory management. The ***garbage collector*** attempts to reclaim the memory used by objects that will never be accessed again by the application. The basic principle of how a garbage collector works is:

1. Determine what data objects in a program will not be accessed in the future
2. Reclaim the storage used by those objects

Garbage Collector finds out that if there are no recent references to an object in the system i.e. an object is not accessed or required by any program in the recent times then the garbage collector takes it for granted that the object can never be referenced again. The particular object is thus freed from any further memory allocation.

Advantages:
Programs written in languages (like Java, C#, C++) that support garbage collectors are normally simpler, significantly shorter and more reliable. Performance of such programs are nearly perfect.

Question 43

What do you understand by page fault? [WBUT 2004]

Answer:
In the virtual memory organization if the content of a particular word referenced by the virtual address does not reside in main memory then a '**page fault**' occurs to signify that the particular page, though needed in the main memory, is not there in main memory and hence it is needed to fetch that page from the disk to the main memory to resume further computation.

Question 44

Write short note on ROM architecture. [WBUT 2004, 2005]

Answer: Refer to section 4.7.

Question 45

What is 'load-through' or 'early restart'?

Answer:
During a memory read operation, if the desired word is not found directly in the cache then it should be copied from the main memory to the cache and a copy of it should be sent to the processor. However in the '***load-through***' or '***early restart***' approach, the

required word may be sent to the processor directly from the main memory without waiting for the cache transfer. Though this approach reduces the processor's waiting period but the circuitry becomes more complex.

Question 46

What is 'capacity' of a cache memory? What is 'line length' of a cache?

Answer:

Capacity of cache is simply the amount of data that can be stored in the cache. For example, a cache with a capacity of 32 KB can store 32 kilobytes of data.

The ***line length*** of a cache is the cache's block size. When there is a cache miss, the required data must be brought into the cache. If the cache is full then some data must be thrown out of the cache to make room for the new data. So the size of the chunks of data that is brought into the cache and thrown out of the cache in response to a cache miss is the cache's line length.

For example, when a cache with 32-byte cache lines has a cache miss, it brings a 32-byte block of data (as the cache line is of 32-bytes) containing the required address from the main memory into the cache. Also, if necessary, a 32-byte block of data may be thrown out of the cache to make room for the new data.

Question 47

a) What are the widths of data bus and address bus for (4096×8) memory?
b) What do you mean by program status word?
c) Define content addressable memory? **[WBUT 2005]**

Answer:

a) Data Bus width is 8 and address bus width is 12.

b) The Program Status Word (PSW) is an area of memory (a hardware register), that contains information about the state of the program used by the operating system and the underlying hardware. It contains an error status field and condition codes such as the interrupt enable/disable bit and a user mode bit.

c) Content Addressable Memory (CAM) or Associative Memory is a special type of memory unit accessed by contents (i.e. contents of memory location) and not by location addresses (like other memory units). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.

Question 48

Can a Read Only Memory be also a Random Access Memory? Justify your answer. Discuss the concept of associative memory unit using suitable example. **[WBUT 2005]**

OR,

Can a ROM be also a RAM? Justify your answer. **[WBUT 2009]**

Answer:

1st part: Refer to section 4.8.

2nd part: Refer to section 4.17.

Question 49

Describe the different types of memory access schemes as found in a digital computer. What is the relationship between data transfer rate and memory access time?

Answer:

The different types of memory access schemes are:

(a) Serial Access scheme:

Such type of access scheme is used in magnetic tapes. Here the information is accessed in a byte-by-byte fashion one by one serially.

(b) Semi-Random access scheme:

This type of scheme is used in magnetic disks. Here firstly the correct track, in which the required information is there, is to be chosen randomly out of the many tracks. Then, in that track, the sectors are accessed serially one by one for the required information. As in magnetic disks information is accessed both randomly and serially, hence such access schemes are semi-random in nature.

(c) Random Access scheme:

Such schemes are common for main memories. Here the locations are accessed completely randomly depending on the address send by the CPU

Data transfer rate is reciprocal to the ***memory access time*** irrespective of the memory size i.e. as the data transfer rate increases, the memory access time decreases as the access will be faster then and vice versa.

Question 50

Explain the different cache design parameters that are taken care for cache implementation. Explain the set associative mapping technique used for cache mapping.

Answer:

The different cache design parameters are as follows:

(a) Capacity: The capacity of a cache is simply the amount of data that can be stored in the cache.

(b) Line Length: This is the cache's block size.

(c) Associativity: The associativity of a cache determines how many locations within the cache may contain a given memory address.

(d) Replacement Policy: This policy determines which block of data is to be thrown out of the cache to make room for incoming data (either because the cache is full or because of conflicts for a set).

(e) Writing Policy: This policy determines whether the cache is write-back or write-through in nature.

Question 51

How do the following influence the performance of a virtual memory system?
i) Size of a page ii) Replacement policy. **[WBUT 2007]**

Answer:

(i) Size of a page: If the size of a page is large then less number of page faults will occur. But this will lead to increase in the page transfer time.

(b) Replacement policy: The least recently used frame (i.e. the main memory block) or the least frequently referenced frame must be replaced.

Because if a frame that is the most recently used or the most frequently used, is replaced then it may lead to more faults.

Question 52

Give examples of non-destructive read out memory and destructive read out memory. **[WBUT 2005]**

OR,

Explain destructive read out & non-destructive read out of memory system. **[WBUT 2010]**

Answer:

Example of non-destructive read-out memory: Static Semiconductor memories like Static Random Access Memory (SRAM).

Example of destructive read-out memory: These are magnetic-core memories or ferrite-core memories.

Question 53

Could it be logical for a system to have only paging and not segmentation and vice versa or even both?

Answer:

Only Paging: No, it is not logical. This is because paging does not provide any information regarding the actual contents in a page whether code or data or both. Segmentation provides that.

Only Segmentation: Possible, but number of external fragmentation will be much more.

Having Both: Possible. In this scheme, user processes will directly access the segmented memory and a mapping of the segmented memory into the pages memory will be done after that, which will reduce the external fragmentations.

Question 54

Discuss with suitable logic diagram the operation of an SRAM cell. **[WBUT 2006]**

Answer: Refer to section 4.11.

Question 55

Explain cache replacement policies. **[WBUT 2004, 2008]**

Answer:

Cache replacement policies:

The decision of the right block getting replaced is an important system performance factor. The objective is to keep the blocks in the cache that are likely to be referenced (needed by the CPU) in the near future. As per the property of 'locality of reference' it can be said that the blocks that have been referenced recently will be referenced again soon. So when a block is to be overwritten, it is needed to overwrite the one that has gone the longest time without being referenced. This block is called the ***least recently used (LRU) block*** and the technique is called the ***LRU replacement algorithm*** or **policy**.

While using the ***LRU algorithm***, the cache controller must track references to all blocks as computation proceeds. A counter is used to track the references to each block. Depending on the number of references tracked by each counter (i.e. by the number of hits and misses) it is decided which block is used the longest time back (i.e. the least recently used block). And that block is then replaced. Though very effective LRU algorithms can lead to poor performance when accesses are made to sequential elements of an array that is slightly too large to fit into the cache.

Other replacement algorithms are however not as effective as LRU. Another algorithm that replaces the particular block in the cache set that has been in the cache for the longest period of time is called the ***first-in-first-out (FIFO)*** algorithm. The third algorithm that replaces the particular block in the cache set that has experienced the fewest references is called the ***least-frequently used (LFU)*** algorithm. Apart from these, using an algorithm that **randomly** chooses the blocks to be replaced is also getting effective.

Question 56

What are the different types of ROMs? Explain their principles. **[WBUT 2005]**

Answer: Refer to section 4.13.

Question 57

Classify memory system in a digital computer according to their use. **[WBUT 2007]**

Answer: Refer to section 4.5 and question 3.

Question 58

Differentiate between cache memory and the registers. Could it be possible to use only the registers in place of a cache memory?

Answer:

Differences between cache memory and registers:

No.	Cache Memory	Registers
1.	Cache memory is not visible to the users i.e. the assembly programmers.	Registers are visible to the users i.e. assembly programmers.
2.	Explicitly addressing the cache memory locations are not possible	Explicitly addressing the registers are possible.
3.	Cache memory locations cannot be allocated or used explicitly by any user program.	Possible in registers.

No, it not possible to use only the registers and other memories by passing the cache memory. This is because, the registers are basically small but very fast storage mechanisms, which cannot store large bulk data. For that the systems need to have the main memories and then the secondary memories. Now, a mechanism to form a bridge between the registers and the main memories are required. The cache memories play this role. So, it is required to have the cache memories in the system as well.

Question 59

Draw the internal cell diagram of PROM and explain its functionality. **[WBUT 2008]**

Answer: Refer to section 4.13.

Question 60

Explain Cache Policy. What are the different cache behavior aspects that may be affected by cache policy?

Answer:

A cache policy defines rules that are used to determine whether a request can be satisfied using a cached copy of the requested resource i.e. if the requested resources are already there in the cache then the request can be served directly from the cache else the requested resources are to be fetched from the main or secondary memories.

Cache policy may be implemented in hardware, software, or a combination of both. Some systems allow programs to influence cache policy, by giving hints or directions about future use of data.

There are ***three main aspects of cache behavior*** which the cache policy can affect:

(a) Fetch policy:

This determines which data is fetched into the cache, usually as a result of receiving a request for data that isn't cached.

(b) Eviction policy:

This determines which data is discarded from the cache to provide space for newly fetched data.

(c) Write policy:

This determines how and when modifications to cached data are synchronized with the underlying storage.

Question 61

Differentiate between spatial locality and temporal locality in a cache memory.

Answer:

Spatial and temporal localities refer to program properties. In spatial locality, instructions stored in a certain area or space in the memory is accessed in one go. That is once a certain instruction in a memory location is accessed, it is very likely that the neighboring instructions (in the same area) will be accessed soon. So, when accessed the entire

memory block is brought into the cache, so that the neighboring instructions can also be accessed easily.

In case of temporal locality, instructions once accessed are likely to be accessed soon. So, it is likely that recently used data are likely to be found in the cache memory because it has been brought recently.

Question 62

What do you mean by Winchester Technology?

Answer:

The modern approach of placing the disks and read / write heads in a sealed, air-filtered enclosure is known as ***Winchester Technology***. In unsealed disk units, dust particles may creep in to the track surfaces and hence the data storing capacity of the tracks get reduced. This problem is solved in Winchester technology.

In such units, the read/write heads can operate closer to the magnetized tracks, as dust particles are absent here. The closer the heads are to a track surface, the more densely the data can be packed along the track, and the closer the tracks can be to each other.

Question 63

A hierarchical cache-main memory subsystem has the following specification:
i) Cache access item of 160ns
ii) Main memory access time 960n
iii) Hit ratio of cache memory is 0.9
Calculate the following:
a) Average access time of the memory system
b) Efficiency of the memory system. **[WBUT 2009]**

Answer:

(a) Given: Hit ratio (h) = 0.9, cache memory access time (t_{cache}) = 160 ns, main memory access time (t_{main}) = 960 ns

Now, to access a word, the average required access time ($t_{average}$) is given by:

$t_{average} = h.t_{cache} + (1 - h)\, t_{main}$

So, $t_{average} = 0.9 \times 160 + 0.1 \times 960 = 240$ ns.

(b) Avg access time presence of cache memory 240 ns.

Now if a cache memory is not there,

Then h = 0,

$$\therefore t_{average} = 0 \times t_{cache} + (1-h) \times t_{main}$$
$$= t_{main}$$
$$= 960\text{ns}$$

$$\therefore \text{ system effieiency } = \left(\frac{960-240}{960} \times 100\right)\%$$
$$= \left(\frac{720}{960} \times 100\right)\% \quad = 75\%$$

$\therefore$ System efficiency = 75%

Question 64

Differentiate between paging and segmentation.

Answer:

	Paging	Segmentation
1.	Fixed page sizes	Variable segment sizes
2.	Pages are generally not visible to the users	Segments are visible to the assembly programmers
3.	Length of page table is large	Length of segment table is small
4.	Implementation of a paging scheme is more expensive	Implementation of a segmentation scheme is less expensive
5.	Paging does not give the memory management unit any information about the contents in a page i.e. whether data or code or both	Segmentation provides the memory management unit with the actual logical contents i.e. whether the page contains code or data or both
6.	Paging does not cause external fragmentation	Segmentation causes external fragmentation

Question 65

Explain why some computers may need an associative memory in spite of having a page table.

Answer:

This is because, the accessing time of page tables in a computer memory is much more than that in case of associative memories. Associative memories are fast memories used for speedy access of data. So, using associative memories in spite of having the page table eventually enhances the performance of the computer by reducing the overall data accessing time.

Question 66

Explain the type of fragmentation that may occur in a simple paging system.

Answer:

Internal fragmentation occurs in simple paging systems having fixed paged size. This is because, is such pages, occurrence of holes or unused memory slots are common and for an allocated memory location, these holes add up together to increase the extent of internal fragmentation in the simple paging systems.

Question 67

How many 256 × 4 RAM chip are needed to provide a memory capacity of 2048 bytes? [WBUT 2009]

Answer:

The number of RAM chips needed are: 2048 bytes / 256 x 4
= 16 chips.

Question 68

What is the cause of thrashing? How does a system detect thrashing? What can the system do to get rid of this problem?

Answer:

Thrashing, a very common problem / concept in the memory system (in the virtual memory paging procedures), occurs owing to allocation of less number of pages to a process than it actually requires. This under allocation of the number of pages leads to continuous page faults.

Thrashing can be detected by the system by evaluating the level of CPU utilization as compared to the multiprogramming level in the system. The level of multiprogramming increases in this case.

In order to reduce the problem of thrashing, the system needs to reduce the level of multiprogramming.

Question 69

Explain Belady's anomaly in context of page faults. [WBUT 2010]

Answer:

Belady's anomaly, introduced in 1969, is a very common concept discussed in context of page faults. It proved that while dealing with page faults, it is possible to have more page faults when increasing the number of page frames if a first-in first-out (FIFO) method of frame management is used.

The explanation is as follows:

In a computer memory, information is loaded in the main memory in the form of pages, which are specific sized storage chunks. It is possible to load only a limited number of pages at a time in the memory. For each page to be loaded, an equal sized frame is need in the main memory. If a required page is not found in the main memory, a page fault occurs and that page is brought from the disk or secondary memory. It might happen that there is no free or empty frame in the main memory at the time of occurrence of the page fault to accommodate the new page in the main memory. In such instances, it is required to free a frame to accommodate the new page. Before the introduction of Belady's anomaly, it was acceptable that the common page replacement algorithm producing acceptable results was the FIFO one. But, the anomaly proved that wrong.

Question 70

Rank the following page-replacement algorithms according to their page-fault rates. Also, indicate whether they suffer from Belady's anomaly.
LRU, FIFO, second-chance and optimal.

Answer:

Rank	*Algorithms*	*Belady's anomaly*
1	Optimal algorithm	Does not suffer
2	LRU algorithm	Does not suffer
3	Second-chance algorithm	Suffers
4	FIFO algorithm	Suffers

Question 71

Explain paging in Memory with suitable examples.

Answer:

Paging is a concept of transfer of pages between main memory and an auxiliary store, like the hard disk. So, in paging, relatively inactive pages are removed from physical memory to make places for pages, which are needed by the memory for execution of an instruction. For example, nowadays all Windows O.Ss come with built-in paging files. Page files are in megabytes, created during the Windows XP installation and reside on the hard drive. The actual size of the page file is based on how much RAM is installed in

the computer. By default, XP creates a page file that is 1.5 times the amount of installed RAM and places it on the hard drive where XP is installed.

Question 72

Where does DMA mode of data transfer find its use?

Answer:

DMA is needed to transfer of block of data between the fast magnetic disk and memory. If the transfer is done via the CPU (speed of magnetic disk almost same as of CPU) then time required to transfer the entire block would have been much more. So, when data transfer must be done in blocks and when communicating devices are very fast then DMA is preferable. Generally in the computer, DMA mode of data transfer occurs.

Question 73

Explain a stack organization. **[WBUT 2007]**

Answer:

A stack is actually a storage device that stores information. In a stack, information are stored in such a manner that the last item stored in a stack is the first one to be retrieved from the stack. In a stack, only one information can be accessed at a time. A set of registers constitutes a stack. A stack is called a LIFO or last-in, first-out list (also known as a pushdown list), as the last item stored in the stack is the first item to be retrieved. Stack can be organized with multiple contiguous memory registers and it is then known as memory stack or can be organized using CPU registers, then known as register stack.

Stack Pointer (SP) is a register whose value always points at the top item in stack i.e. SP always holds the address for the stack. The two basic operations that can be performed on a stack are the insertion and deletion of items. In the insertion operation, which is also known as push, information is stored in the top of stack (TOS) position in the stack and in the deletion operation, known as pop, information from the TOS is retrieved.

Figure below shows a 64-word stack organization. The stack grows upward from location 0 to location 63, which is the final TOS. On addition of every new element, the TOS value is incremented by one until the stack is full, marked by FULL ← 1 (FULL is a flip-flop which is 1 when the stack is full and EMTY is a flip-flop which is 1 when the stack is empty). The first element is pushed in the stack at the SP ← 1 location i.e. the first location. So when SP ← 0, it means that the stack is full and hence FULL is 1. So EMTY is 0 i.e. stack not empty.

The steps for the PUSH operation are as follows:

SP ← SP + 1 ⇒ Stack pointer is incremented.

M[SP] ← DR ⇒ Item, from data register, is stored on the TOS.

If (SP = 0) then (FULL ← 1) ⇒ To check if stack is full.

EMTY ← 0 ⇒ Means that the stack is not empty.

While popping elements from the stack, if SP ← 0, it means that the first item pushed at the initial position is popped out. Hence the stack is now empty and so EMTY ← 1. So, FULL is 0.

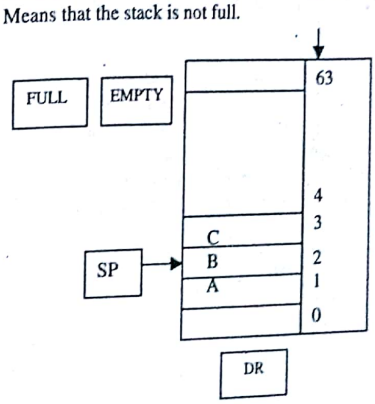
The steps of POP are as follows:

DR ← M[SP] ⇒ Item, from TOS, is popped and stored in the data register.

SP ← SP – 1 ⇒ Stack pointer is decremented.

If (SP = 0) then (EMTY ← 1) ⇒ To check if stack is empty.

FULL ← 0 ⇒ Means that the stack is not full.



Question 74

Explain a "dumb" memory. **[WBUT 2006]**

Answer:

During a memory read or write operation, the CPU plays the key role. It is the CPU, which is aware of the endianness and the alignment of words. Memory has got absolutely no idea of how things work. It does not have any idea about how a byte that is asked to be read is actually retrieved or how the storage mechanism (in form of bytes) works. Memory is totally 'dumb' in this context. Unlike the CPU, memory does not place meaning to the bytes.

Question 75

Explain the stack organization of a computer system & write all the sequence of Micro-operations for push & pop operations. [WBUT 2003]

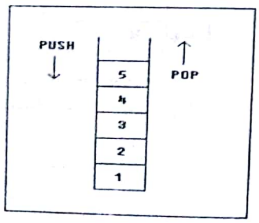
Answer: Refer to question 73.

Question 76

Explain a stack memory.

Answer:

The Stack is an area of memory for keeping temporary data. So, it is basically a Last In First Out (LIFO) memory. PUSH and POP instructions are used to insert and remove data from the stack. Two registers, namely the Stack Pointer (SP) and the Stack Segment (SS) register, maintain the stack memory. When a word of data is PUSHED onto the stack the High order 8-bit Byte is placed in location SP-1 and the Low 8-bit Byte is placed in location SP-2. The SP is then decremented by 2. Figure below shows a simple stack.



Question 77

What is the difference between associative and set-associative mappings?

Answer:

(i) In associative mapping technique there is no restriction on where a main memory block can be placed in the cache. So, as a main memory block can be placed into any cache block position, the space in the cache can be used very efficiently.

On the other hand, set-associative mapping technique reduces the contention problem by giving a few choices for block placement as this technique allows a main memory block to reside in any block of a specific cache set.

(ii) In associative mapping technique, a very complex circuitry is needed to compare all the tags parallel and thus owing to the complex circuitry, the implementation cost is also very high.

On the other hand, set-associative mapping technique reduces the hardware cost by reducing the size of associative search and reducing the complexity of the needed circuitry.

Question 78

How can locality of reference reduce page faults?

Answer:

Page faults can be significantly reduced if the required pages can be pre-fetched in to the memory (from the secondary storage) before they are actually needed. So, as soon as a program core is loaded in the memory, execution starts and the O.S starts pre-fetching other required pages into the memory, anticipating their requirements as execution proceeds. This approach allows a program to load and execute faster. A pre-fetching is marked as an effective one, if the O.S. ably pre-fetches those pages that are most relevant to the execution of the program. In this context, pre-fetching pages based on the 'locality of reference' concept is an able approach. With the help of this concept, those pages, that are in close proximity of the page currently being used by the processor, gets loaded in the memory. Both temporal and spatial locality concepts are taken into account while choosing the required pages. This idea is based on the fact that program execution is usually local and thus page loads that are local will be most beneficial to execution speed.

Question 79

What are the differences between primary and secondary storage devices?

Answer:

Differences between primary and secondary storage devices:

1. In primary devices the storage capacity is limited, whereas in secondary storages the capacity is much larger.
2. Primary storage devices may have volatile memory, which is not the case for secondary storage devices.
3. Primary storage devices may be expensive as compared to the secondary storage devices.
4. Primary storage devices are faster than the secondary storage devices.
5. RAM, ROM etc are examples of primary storage devices, whereas hard drives and floppy disks are typical examples of secondary storage devices.

Question 89

What is meant by random access and sequential access of memory devices?

Answer:
A concept like random access of memory devices, takes the form of ICs in which the stored data can be accessed in any order or randomly. Any data can be accessed in a constant time irrespective of its storage location.
In contrast, in the concept of sequential access of memory devices, the data is accessed in sequence one after another as in any magnetic memory.

Worked-Out Examples

Example 1

How many 8K-byte RAM chips are needed to build a 512K word memory, each word being 32 bits wide?

Solution:
512 K word, each of 32 bits (or 4 bytes) is equivalent to 512 K x 4 = 2 M-bytes.
Hence the number of RAM chips required is: 2 M bytes / 8 K bytes = $2 \times 2^{20} / 8 \times 2^{10} = 2^{11} / 2^3 = 2^8 = 256$.

Example 2

Explain how many address lines are present in a 4K byte RAM.

Solution:
$4K = 4 \times 1024$ bytes $= 2^2 \times 2^{10} = 2^{12}$ bytes.
Now, since each memory location must have a unique address, 2^{12} unique addresses are required. Hence 12 address lines are present in a 4K byte RAM.

Example 3

(a) How many address lines are present in a 256K x 8 RAM?
(b) How many such RAMs will be required to construct 1M x 32-memory bank?
(c) How many such RAMs will be required to construct 512K x 32-memory bank?

Solution:
(a) 256K x 8 RAM $\Rightarrow$ Memory has 256K words of 8 bit each.
As, $256K = 2^8 \times 2^{10} = 2^{18}$ bytes and as memory location need a unique address, hence 2^{18} unique addresses are required. So, 18 address lines are required.
(b) $1M \times 32 = 2^{10} \times 2^{10} \times 2^5$ and $256K \times 8 = 2^8 \times 2^{10} \times 2^3$.
So, $2^{10} \times 2^{10} \times 2^5 / 2^8 \times 2^{10} \times 2^3 = 2^4 = 16$ such RAM chips are required to construct the required memory bank.
(c) $512K \times 32 = 2^9 \times 2^{10} \times 2^5$ and $256K \times 8$ RAM $= 2^8 \times 2^{10} \times 2^3$.
So, $2^9 \times 2^{10} \times 2^5 / 2^8 \times 2^{10} \times 2^3 = 2^3 = 8$ such RAM chips are required to construct the required memory bank.

Example 4

A computer uses RAM chips of 1024 x 1 capacity. How many chips are needed to provide a memory capacity of 16K bytes?

Solution:
RAM chip capacity = 1024 x 1 $\Rightarrow 2^{10}$ words are there.
Now, 16K bytes = $2^4 \times 2^{10} \times 2^3 = 2^{17}$.
Hence $2^{17} / 2^{10} = 2^7$ RAM chips are needed.

Example 5

Assuming that a processor has 1M-byte memory addressing capability, determine the address bits that must be externally decoded to select a 16K- byte RAM chip.

Solution:
As the processor has 1M-byte memory addressing capability, so the memory address bus is 20-bit (as $2^{20} = 1$ M) wide. Hence, the address lines are A_{19} (MSB) through A_0 (LSB).
Now, the RAM chip is 16K-byte i.e. it has 14 address (as $2^4 \times 2^{10} = 2^{14} = 16$K). Hence, these 14 address lines, A_{13} through A_0 and the chip should be selected by an external 6-bit (as 20 – 14 = 6) decoder that will decode the address bits A_{19} through A_{14}.

Example 6

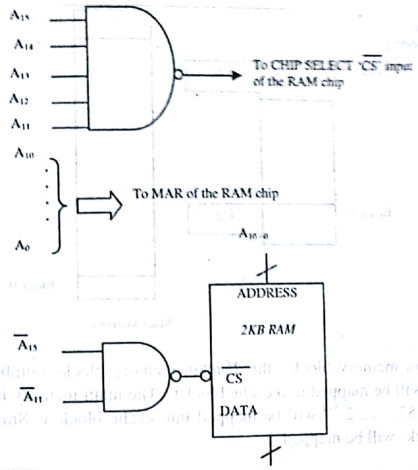
Classify the memory system in a digital computer according to their use. A random access memory module of capacity 2048 bytes is to be used in a computer and mapped between the address $(2000)_H$ and $(27FF)_H$. Explain with the help of a block diagram the address-decoding scheme assuming a 16-bit address tree.

[WBUT 2007]

Solution:
RAM capacity = 2048 bytes = 2^{11} bytes.
Thus the RAM has an 11-bit internal decoder which decodes the RAM address 0 through $2^{11} - 1$ (i.e. 2047).
It is desired to select this RAM chip (using its CS input) with an external decoder such that the RAM occupies the 2^{11} addresses, 2000H through 27FFH.

Clearly, the highest 5 bits of the address, namely, A_{15} A_{14} A_{13} A_{12} A_{11} should be chosen equal to the base address 00100B, so that the 2K byte RAM chip occupies the desired addresses (i.e. 2000H through 27FFH) in the 64 bytes.

Diagram:

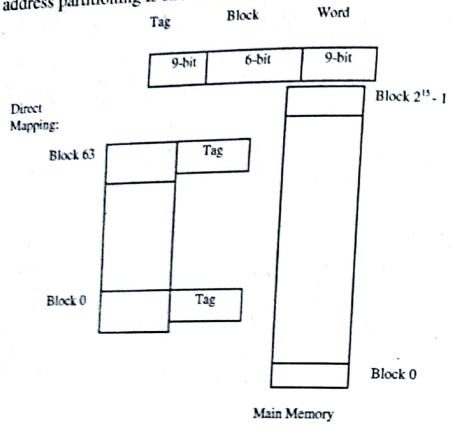


Example 7

A computer has main memory capacity of 16M bytes and a cache memory of 32K bytes. Each block of 51 bytes in main memory is mapped onto a similar size block in cache using direct mapping technique. Show the partitioning of the cache address register into tag field, sector field and block field. Explain how cache will be accessed. Discuss how hit ratio can be improved.

Solution:
Main memory capacity = 16 M bytes = 2^{24} bytes i.e. main memory address is 24-bit wide.
Cache memory capacity = 32 K bytes = 2^{15} bytes i.e. the cache memory address is 15-bit wide.
The block size is 512 bytes = 2^9 bytes.
So the number of cache blocks is $2^{15} / 2^9 = 2^6 = 64$.
The number of main memory blocks is $2^{24} / 2^9 = 2^{15} = 32$K.

Diagram:
The memory address partitioning is shown below:



Explanation:
Out of the 2^{15} main memory blocks, the 2^9 main memory blocks numbered 0, 64, 128, 192, 256, ..., 2^{15} will be mapped into cache block 0. The main memory blocks numbered 1, 65, 129, 193, 257, ..., 2^{15-1} will be mapped into cache block 1. Similarly, the other main memory blocks will be mapped.

Cache Access Mechanism:
When the CPU will send a 24-bit main memory address A_{14-0}, the cache management unit will extract the most significant 9 bits (A_{23-15}) of this address, i.e. the TAG field of the desired address. It will also extract the next most significant 6 bits of the address, i.e. the BLOCK field of the desired address. Now it will compare the TAG field of the issued address with the TAG field associated with block (the address of the selected block address is the 6-bit block field extracted from the issued address). If the comparison shows equality, then the desired location corresponding to the issued address is resident in selected BLOCK of the cache and its content can be read out and sent to the CPU. Otherwise the selected block will be replaced by the main memory block whose block address as well as the tag address matches with the corresponding values in the desired (issued) address.

Hit Ratio Improvement:
The hit ratio can be improved by having more number of blocks (i.e. having 128, 256, 512, etc. blocks) or by increasing the size of each blocks (say, 1K, 2K, 4K etc.).

Example 8

A computer has direct mapped cache with 16 one-word blocks. The cache is initially empty. What is the observed hit ratio when the CPU generates the following word address sequence?
1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9, 17

Solution:
There are 16 blocks each of size 1 word. Since the cache is initially empty, the hit / miss is given below against each word address in the sequence:

1	→ miss	Block 1 contains (1)	→	load
4	→ miss	Block 4 contains (4)	→	load
8	→ miss	Block 8 contains (8)	→	load
5	→ miss	Block 5 contains (5)	→	load
20	→ miss	Block 4 contains (20)	→	Replace
17	→ miss	Block 1 contains (17)	→	Replace
19	→ miss	Block 3 contains (19)	→	Replace
56	→ miss	Block 8 contains (56)	→	Replace
9	→ miss	Block 9 contains (9)	→	load
11	→ miss	Block11contains (11)	→	load
4	→ miss	Block 4 contains (4)	→	Replace
43	→ miss	Block11contains (43)	→	load
5	→ hit	Block 5 contains (5)	→	Read/Write
6	→ miss	Block 6 contains (6)	→	Load
9	→ hit	Block 9 contains (9)	→	Read/Write
17	→ hit	Block 1 contains (17)	→	Read/Write

Hence, hit ratio = 3 / 16.

Example 9

A computer has cache access time of 100 nanosecs, a main memory access time of 1000 nanosecs, and a hit ratio of 0.9. Find the average access time f the memory system.

Solution:
Hit ratio= 0.9 i.e. out of 100 words, 90 words are in cache.

Hence in cache, these 90 words can be accessed in 90 x 100nanosecs = 9000 nanosecs.
It is needed to get the remaining 10 words from the main memory to cache to execute them and time needed for that = 10 x 1000 nanosecs = 10000 nanosecs.
Once these 10 words are in cache, they can be accessed in 10 x 100 = 1000 nanosecs in the cache.
Total access time for these 10 words (i.e. access time in main memory + access time in cache) = 10000 nanosecs + 1000 nanosecs = 11000 nanosecs.
Hence, overall access time for these 100 words = access time for the 90 words in cache + access time for the remaining 10 words = 11000 nanosecs + 9000 nanosecs
= 20000 nanosecs.
So average access time for these 100 words
= 20000 / 100 nanosecs = 200 nanosecs.

Example 10

Consider the previous problem. Suppose that in the computer there is no cache memory. Then find the average access time, when the main memory access time is 1000 nanosecs. Compare the two access times.

Solution:
If there is no cache memory in the computer, then all the 100 words would have been in the main memory itself.
So access time for these 100 words in main memory = 100 x 1000 nanosecs = 100000 nanosecs.
Hence the average access time = 100000 / 100 nanosecs = 1000 nanosecs.
It can be said that the access time decreases by 5 times in presence of the cache memory.

Example 11

If numbers of hits are equal to 10 and number of misses are equal to 4 then find the hit ratio.

Solution:
Hit ratio = 10 / (10 + 4) = 10 / 14 = 5 / 7.

Example 12

Suppose that a main memory is to have 65536 locations. How many address wires will that main memory need? Explain carefully your answer.

Solution:
The main memory circuit would need 16 address wires. The address wires are used to pick out one particular memory location. Every possible *combination* of values for the address wires can specify a different memory location. Since there are two possible values (ON and OFF) for each wire, then with 16 wires, there will be a total of 2^{16} combinations.

Example 13

A virtual memory system is based on 32 bit virtual addresses. Each page is 2k bytes. A Translation Look-aside Buffer is used to cache page table entries, the TLB is direct mapped and holds a total of 2k page table entries.
(a) How many of the bits from a virtual address are used to determine where in the page table to look for the physical page number?
(b) How many (potential) entries are there in each pagetable (each process has it's own pagetable)?
(c) Assume the TLB is currently full. How many unique bytes of memory (how many different byte addresses) can be accessed without needing to go to memory for the address translation?

Solution:
[Hint: 232 bytes / 211 bytes/page implies 221 pages, each has a unique page address, so the page addreses are 21 bits long.
Each pagetable could have 221 page table entries (2M page table entries).
TLB holds 211 (2K) page table entries, each entry is the translation for one page (211 (2K) bytes). Total memory addressable without going to the page table is:
211 * 211 = 222 (4M) bytes].

Example 14

How many bytes can be addressed on a computer with 26 bit addresses?

Solution:
[Hint: 2^{26} = 64M bytes can be addressed. It should be converted to bytes for the answer].

Example 15

What is the average rotational latency of a disk that rotates at 3600 rpm?

Solution:
The disk rotates at 3600 rpm, which means that 1/60 seconds is the time for each revolution. Now, on an average, distance between the head and a sector will be half a revolution. So, the average rotational latency will be 1/2 * 1/60 = 8 ms.

Example 16

If hit ratio of a cache memory system is 0.9, access times for the cache and the main memory are respectively 20 ns and 80 ns, then find out the overall system speed up when such a cache memory is used.

Solution:
Given: Hit ratio (h) = 0.9, cache memory access time (t_{cache}) = 20 ns, main memory access time (t_{main}) = 80 ns
Now, to access a word, the average required access time ($t_{average}$) is given by:

$$t_{average} = h.t_{cache} + (1 - h)\, t_{main}$$

So, $t_{average} = 0.9 \times 20 + 0.1 \times 80 = 26$ ns.
Now, if a cache memory is not there then h = 0 and $t_{average} = t_{main} = 80$ ns. Hence, in presence of the cache memory, the speed up factor is 80/26 = 3.1.

Example 17

In a cache with 64-byte cache lines, how many bits are used to determine which byte within a cache line an address points to?

Solution:
$\text{Log}_2 64 = 6$. Hence the low 6 bits of the address determine an address's byte within a cache line.

Example 18

If a cache has 64-byte cache lines, how long does it take to fetch a cache line if the main memory takes 20 cycles to respond to each memory request and returns 2 bytes of data in response to each request?

Solution:
As the cache has 64-byte cache line, hence if the main memory returns 2 bytes of data in response to each request, 64 / 2 = 32 memory requests are required to fetch the line. So at 20 cycles per request, fetching a cache line will take 32 x 20 = 640 cycles.

Example 19

Main memory size is 64K bytes. Cache memory size is 1K bytes. Block size is 64 bytes. Block-set associative mapping with 4 blocks per set is used.
How many bits are there in different fields of the address generated by CPU?

Solution:
Total main memory address size is $64K = 2^6 \times 2^{10} = 16$ bits.
Size of each block (both main memory and cache memory) is 64 bytes. Hence number of cache memory blocks = 1K bytes / 64 bytes = $2^{10} / 2^6 = 2^4 = 16$ (where cache memory size is 1K bytes).
There are 4 blocks per sets. Hence there are total of 16 / 4 = 4 sets. So 2 bits are required to identify any one of these 4 sets. Hence the set field is of 2 bits.
As each block size is 64 bytes = 2^6 bytes, hence to identify any of the words in the block, 6 bits word field is required.
There are total of 64K / 64 bytes = $2^6 \times 2^{10} / 2^6 = 2^{10} = 1024$ main memory blocks. As there are 4 sets in the cache memory, hence, at a time, each cache block can hold any one of the 1024 / 4 = 256 = 2^8 main memory blocks. So 8 bits tag field is required. So the main memory address will be like:

Tag	Set	Word
8-bit	2-bit	6-bit

Example 20

In a direct-mapped cache with a capacity of 16 KB and a block size of 32 bytes, how many bits are used to determine the byte that a memory operation references within a block, and how many bits are used to select the cache block that may contain the data?

Solution:
As the cache block size is 32 bytes, hence 5-bit (as $2^5 = 32$) word field is required to determine which byte within a cache block the CPU references.
As the cache capacity is 16 KB, hence number of cache blocks = 16KB / 32 bytes = $2^4 \times 2^{10} / 2^5 = 2^9 = 512$. Hence 9-bits are required in the block field to select the particular block that may contain the desired main memory address.

Example 21

How many sets are there in a set-associative cache with 32KB capacity and 64 bytes block size. How many bits of the address are used to select a set in this cache?

Solution:
As the cache capacity is 32 KB and size of each cache block is 64 bytes, hence number of cache blocks = $2^5 \times 2^{10} / 2^6 = 2^9 = 512$.
Assuming there are two cache blocks per set, number of cache sets = 512 / 2 = 256.
Hence number of bits needed to select a set out of the 256 sets = 8 (as $2^8 = 256$).

Example 22

In a memory system, the logical address space consists of 64 pages with 2048 words in each page. These are mapped into the physical address space of 32 frames. Calculate the number of bits in the logical and physical addresses.

Solution:
Number of bits constituting the 64 pages is 6 (as $2^6 = 64$) and number of bit constituting the words in each page is 11 (as $2^{11} = 2048$). Also, number of bits in the 32 frames is 5 (as $2^5 = 32$). Therefore, the number of bits in the logical address is 6 + 11 = 17 (i.e. number of bits constituting the pages and words in each page). Also, the number of bits in the physical address is 5 + 11 = 16 (i.e. number of bits in frames + those in the words).

Example 23

If a cache memory is 10 times faster than the main memory and the cache memory can be used 90% of the time, then calculate the total speedup that can be obtained by using the cache.

Solution:
Speed up is given by the following expression:
1 / [(1 - % of time that the cache is used) + (% of time that the cache is used / speed of cache)]
Therefore, 1 / [(1 – 0.9) + (0.9 / 10)] = 1 / 0.19 = 5.3.
Hence, speed up obtained by using the cache is 5.3 times.

Example 24

According to the following information, determine the size of the subfields (in bits) in the address for Direct Mapping and Set Associative Mapping cache schemes:

- **We have 256 MB main memory and 1 MB cache memory**
- **The address space of the processor is 256 MB**
- **The block size is 128 bytes**
- **There are 8 blocks in a cache set.**

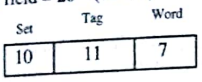
Solution:
As the size of the main memory is 256 MB hence there are 28 bits (as $256 = 2^8$ and 1 MB = 2^{20} bytes and hence 256 MB = $2^8 \times 2^{20} = 2^{28}$) in the main memory address or the address size of main memory is 28 bits.
Size of the sub-fields for direct mapping cache schemes:
Now size of cache memory is 1 MB = 2^{20} bytes. Hence number of cache memory blocks = $2^{20} / 2^7 = 2^{13}$. So number of bits in the block field = 13. Now out of the total main memory address size of 28 bits, world field contains 7 bits and the block field contains 13 bits. Hence, the number of bits in the tag field is 28 – (13 + 7) = 8.

Block	Tag	Word
13	8	7

Main Memory Address

Size of the sub-fields for set-associative mapping cache schemes:
In this case, there are 8 blocks per cache set and the total numbers of cache blocks are 2^{13}. So number of sets in the cache memory are $2^{13} / 8 = 2^{13} / 2^3 = 2^{10}$.

Hence, number of bits in the set field is 10 and that in the word field is 7. So number of bits in the tag field = 28 – (10 + 7) = 11.



Main Memory Address

Example 25

How many 128 x 16 RAM chips are needed to construct a memory capacity of 4096 words (16 bit is one word)? How many lines of the address bus must be used to access a memory of 4096 words? For chip select, how many lines must be decoded?

Solution:

The number of RAM chips needed are: 4096 words / 128 x 16
= 4096 x 16 bits / 128 x 16 = 2^{12} x 2^4 / 2^7 x 2^4 = 2^5 = 32.

As, 4096 = 2^{12}, hence 12 lines address bus should be used.

For chip select, 6 lines in the horizontal and 6 lines in the vertical direction should be decoded.

Example 26

Given the following, determine the size of the sub-fields in the address for direct mapping, associative mapping and set-associative mapping cache schemes:

Main memory size	**512 MB**
Cache memory size	**1 MB**
Address space of processor	**512 MB**
Block size	**128 B**
8 blocks in cache set.	

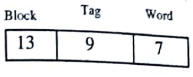
Solution:

As the size of the main memory is 512 MB hence there are 29 bits (as 512 = 2^9 and 1 MB = 2^{20} bits; hence 512 MB = 2^9 X 2^{20} = 2^{29}) in the main memory address i.e. the address size of main memory is 29 bits.

Size of the sub-fields for direct mapping scheme:

Now size of cache memory is 1 MB = 2^{20} bytes. Hence number of cache memory blocks = 2^{20} / 2^7 = 2^{13}. So number of bits in the block field = 13.

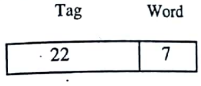
Now out of the total main memory address size of 29 bits, word field contains 7 bits (as block size is 128B) and the block field contains 13 bits. Hence, the number of bits in the tag field is 29 – (13 + 7) = 9.



Main Memory Address

Size of the sub-fields for associative mapping scheme:

Each block size is 128 bytes or 2^7 bytes. Hence number of main memory blocks = 2^{29} / 2^7 = 2^{22}. So number of bits in the tag field is 22 and that in the word field is 7 (as block size is 128 bytes).

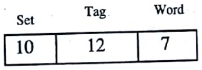


Main Memory Address

Size of the sub-fields for set-associative mapping scheme:

In this case, there are 8 blocks per cache set and the total numbers of cache blocks are 2^{13}. So number of sets in the cache memory is 2^{13} / 8 = 2^{13} / 2^3 = 2^{10}.

Hence, number of bits in the set field is 10 and that in the word field is 7. So number of bits in the tag field = 29 – (10 + 7) = 12.



Main Memory Address

Example 27

A disk pack has 20 recording surfaces and has a total 4000 cylinders. There is an average of 300 sectors per track. Each sector contains 512 bytes of data.

i) What is the maximum number of bytes can be stored in this pack?

ii) What is the data transfer rate in bytes per second at a rotational speed of 3600 rpm?

Solution:

Storage capacity of the entire disk drive = number of surfaces in the drive * number of tracks / surface * number of sectors / track * number of bytes / sector = 20 * 400 * 20 * 4000 bytes = 256 MB

Number of bytes transferred from each surface during one revolution of the disk = number of bytes / track = 20 * 4000 bytes = 80, 000 bytes.

Time per revolution = 60 / 3600 sec = 1 / 60 sec

So, data transfer rate from each surface of the disk drive = 60 * 80 kb = 4.8 mbytes / sec.

Example 28

It is desired to have 5kB of memory in a computer. 4kB should be ROM and 1kB should be RAM. You have to design the chip-select signals of Memory-chips in such a way that first 4k addresses should select ROM and next higher 1k addresses should select RAM. You have been given a number of ROM chips (namely 2716 which is 2k × 8) and RAM chips (namely 2142 which is 1k × 4).

Solution:

We are given 2716 (2k x 8) ROM chips and 2142(1k x 4) RAM chips. We note that two 2716 ROM chips and two 2142 RAM chips will be needed for building (4k x 8) ROM and (1k x 8) RAM, respectively. We assume that the computer has a 16-bit memory address bus, (A15-0) and we note that the least significant 11 bits (A10-0) of the ROM chips, and the least significant 10 bits (A9-0) of the RAM chips, are decoded internally while the remaining 5 and 6 bits, respectively, are to be decoded by external decoders.

Thus, the schemes for chip selection are as follows:

(a) ROM: Since the two ROM chips should occupy the address blocks, 0-2k (0000h-07FF h) and 2k-4k (0800h-0FFFh), respectively, they should be selected by decoding (ANDing) the address bits (A15=A14=A13=A12=0, A11=0) and (A15=A14=A13=A12=0,A11=1),respectively.

(b) RAM: Since the two RAM chips should occupy the same address block 4k-5k (1000h-13FFh), both should be selected by decoding the address bits (A15=A14=A13=A12=A11=A10=0).

Example 29

Explain how a RAM of capacity 2 k bytes can be mapped into address space (1000) H to (17 FF) H of CPU having a 16-bit address lines. Show how the address lines are decoded to generate the chip select condition for the RAM.

Solution:

RAM capacity = 2Kbytes = 2048 bytes = 2^{11} bytes.

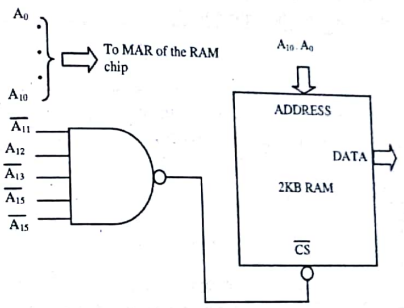
Thus the RAM has a 11-bit internal decoder which decodes the RAM addresses 0 through $2^{11}-1$ (i.e. 2047).

It is desired to select this RAM chip (using CS input), such that the RAM occupies the 2^{11} address, $(1000)_H$ through $(17FF)_H$. This can be done using a (16 – 11)-bits i.e. a 5-bit external decoder.

So, the highest five bits of the address namely A_{15} to A_{11} should be chosen equal to the base addresses 00010_H such that the 2Kbyte RAM chip occupies the desired address space [i.e. $(1000)_H$ through $(17FF)_H$].

The decoding needs ANDing of the bits (literals) $\overline{A_{15}}$, $\overline{A_{14}}$, $\overline{A_{13}}$, A_{12} and $\overline{A_{11}}$ (this is because the base address is 00010_H).

Diagram:



Example 30

A CPU has 32-bit memory address and a 256 kB cache memory. The cache is organized as a 4-way set associative cache with block size of 16 bytes.
i) What is the number of sets in the cache?
ii) What is the size (in bits) of the tag field per cache block?
iii) What is the number and size of comparators required for tag matching?
iv) How many address bits are required to find the byte offset within a cache block?

Solution:

32-bit memory address implies that the main memory size is (2^{32}) bytes = 4 GB.

(i). Since the size of the cache is 256 KB = 2^{18} B and the size of each block is 2^4B, the number of blocks in the cache is $2^{18}/2^4=2^{14}$. The cache being 4-way set-associative, the number of sets in the cache is $2^{14}/2^2 = 2^{12}$ = 4K.

(ii). If the cache were direct-mapped, the size in bits of the tag field would have been 14 bits since $2^{32}/2^{18}=2^{14}$. However, because the cache is 4-way set-associative and $4=2^2$, the tag field should have a size of 14+2=16 bits.

(iii) Since the cache is 4-way set-associative and the tag field is 16 bits, 4 comparators, each of size 16 bits, are required.
(iv) Since the size of a cache block is 16B = (2^4)B, it is obvious that 4 bits are required to find the byte offset within a cache block.

Example 31

What is the bandwidth of a memory system that transfers 128-bit data per reference having a speed of 20 nano sec per operation?

Solution:
Each memory reference (for a read or write operation) takes 20 nano seconds and transfers 128 bits of data between the CPU and the memory. Therefore, the bandwidth of the memory system is 128 bits / 20 nano seconds = 6400 Mbits per second, i.e., 800 MB/sec.

Example 32

A disk drive has 20 sectors / track, 4000 bytes / sector, 8 surfaces all together. Outer diameter of the disk is 12 cm and inner diameter is 4 cm. Inter-track space is 0.1mm. What is the no. of tracks, storage capacity of the disk drive and data transfer will be there from each surface? The disk rotates at 3600 rpm.

Solution:
Radial distance covered by all the tracks lying on a surface = 12 – 4/2 = 4 cms.
So, number of tracks/surface = 4 * 10 mm/.1 mm = 400.
Storage capacity of the entire disk drive = number of surfaces in the drive * number of tracks / surface * number of sectors / track * number of bytes / sector = 8 * 400 * 20 * 4000 bytes = 256 MB
Number of bytes transferred from each surface during one revolution of the disk = number of bytes / track = 20 * 4000 bytes = 80, 000 bytes.
Time per revolution = 60 / 3600 sec = 1 / 60 sec
So, data transfer rate from each surface of the disk drive = 60 * 80 kb = 4.8 mbytes / sec.

Example 33

A disk pack has 19 surfaces. Storage area on each surface has an inner diameter of 22 cm and outer diameter of 33 cm. Maximum storage density on each track is 2000 bits/cm and minimum spacing between tracks is 0.25 mm.
i) What is the storage capacity of the pack?
ii) What is the data transfer rate in bytes per second at a rotational speed of 3600?

Solution:
We shall assume a cylinder-based design of the disk pack so that each track has the same number of bits stored in it. This implies that the recording density is maximum in the innermost track and minimum in the outermost track. Additionally, we shall assume a shared head disk surface which is more common than a costly 'head per track' disk surface.
(i) Number of tracks in each surface = track recording space / inter track spacing = (33 cm – 22 cm) / 0.25 mm = 440.
Number of bits in each track = (circumference of the innermost track in cm) * 2000 bits / cm = pi * D min for innermost track * 2000 bits = (3.14) * (22) * (2000)= 138160 bits
Therefore, storage capacity of the disk pack = 19 surfaces * 440 tracks per surface * 138160 bits = 1,135,017,600 bits.
(ii) During one rotation of the disk pack, the entire volume of data stored in the selected track on the selected surface can be transferred.
Now, at 3600 rpm rate, the number of rotations per sec is 3600 / 60 = 60.
So, data transfer rate = (60) * (138160) / 8 byte per sec=17270 bytes per sec. [Note: 138160 bits = 8 bytes]

Example 34

Suppose a DRAM memory has 4k rows in its array of bit cells. Its refreshing period is 64 ms. 4 clock cycles are required to access each row.
i) What is the time needed to refresh the memory if the clock rate is 133 MHz?
ii) What fraction of the memory's time is spent for performing refreshes?

Solution:
Period of 133 MHz clock = 1 / [(133) * (10^6)] sec = 7.52 nano sec.
Number of clock cycles needed for refreshing 4k rows = 4 clock cycles per row * 4096 rows = 16,384 clock cycles [Note: 4k rows i.e. 4 * 1024 = 4096 rows].

So, time required to refresh the chip = 7.52 * 16, 384 = 123207.68 nano sec = 0.123 msec.
Therefore, fraction of the DRAM's time spent for performing refreshes = 0.123 / 64 = 0.002

Example 35

A hierarchical cache-main memory sub-system has the following specifications: Cache access time : 50 ns, Main memory access time : 500 ns, 80% of memory request for read, hit ratio : 0.9 for read access and write-through the usage.

i) Calculate the average access time of the memory system considering memory read cycle.

ii) Calculate the average access time of the memory system both for retrieve & write.

Solution:
(i) considering only read operations, average access time = (0.9) * (50) + (1 – 0.9) * (500) = 45 + 50 = 95 nano sec.
(ii) considering 80% read and 20% write operations, average access time = (0.8) * (95) + (0.2) * (500) nano sec = 76 + 100 = 176 nano sec.

Short Questions & Answers

1. What is a bipolar storage cell?
Answer:
In bipolar technology, both the electrons and holes are the carriers simultaneously. Here both electrons and holes move simultaneously.

2. What is a replicated memory?
Answer:
Replicated memory system supports parallel memory requests by providing multiple copies of the entire memory where each copy has the capability to handle any memory requests. Thus the overall memory system performance gets improved a lot.

3. Explain bandwidth in reference to memory systems.
Answer:
Bandwidth gives the total rate of data movement between the memory system and the processor. It is basically the product of the amount of data referenced by each memory operation and the throughput.

4. What is a scratch pad of a computer?
Answer:
Cache memory is known as the scratch pad of a computer

5. What is a stack?
Answer:
Stack is basically a portion of the main memory unit, or specifically, a portion of the RAM in which the contents of the program counter and the general purpose registers are stored.

6. Why can't ROM be used as a stack?
Answer:
This is because it is difficult to write in a ROM and it is non-volatile in nature.

7. What is a shadow RAM?
Answer:
Shadow RAMs are basically non-volatile read-write memories. For example, flash memory.

8. Name the two main components in a DRAM bit.

Answer:
These are the transistor and the capacitor.

9. What is meant by memory management?
Answer:
This is the act of managing computer memory. In its simpler forms this involves providing ways to allocate portions of memory to programs at their request and free it back to the system for reuse when no longer needed.

10. What is a look-ahead-cache?
Answer:
A "look-ahead cache" attempts to store information that will be (or may be) asked soon.

11. What is Memory Management Unit (MMU)?
Answer:
MMU is a special hardware unit that translates virtual addresses into physical addresses. If the required data to be executed is not present in the main memory, MMU causes the operating system to bring the data into the memory from the disk.

12. Where are modular memories used?
Answer:
Modular memories are useful in systems dealing with pipeline and vector processing.

13. What are the advantages and disadvantages of implementing a computer system having only cache memory and no other memory?
Answer:
In this case the cache memory will perform as the main memory of the system. The ***advantage*** is that the system would be very fast but the ***disadvantage*** would be the very high cost of the system for a specific memory capacity.

14. What is relocatable code?
Answer:
These codes can reside anywhere in the memory. This implies that in such codes, the memory location's address is not hard-coded or fixed but are relative i.e. it depends on other fixed locations, say, the beginning address of the main program. During its execution, it is the operating system, which decides the location in the memory where the code will run then.

15. Why is formatting of disk necessary?

Answer:
Formatting of disk is necessary to keep the recording format of bits in the disk fixed i.e. to keep the number of bits per sector per track fixed and the frequency of application of clock pulses fixed. Otherwise errors may creep in.

16. What is encoding?
Answer:
Encoding is the concept of conversion of input binary numbers into electric signals that can be recorded on the disks.

17. What is the advantage of phase encoding?
Answer:
Phase encoding provides a clock signal because of the mid-interval transition in every bit signal.

18. When do you think is larger block sizes advantageous?
Answer:
This is advantageous when a sequence of instructions are executed or in an array looping occurs through every element. In such cases having smaller block size means frequent page faults, which ultimately degrades performance.

19. What does the bar over the chip select pin in a memory chip imply?
Answer:
The bar implies that the particular chip is selected by keeping this pin low. It is like a Boolean variable. Different selections are made by keeping chip select pin at high or at low.

20. What is demand paging?
Answer:
Demand Paging: In virtual memory paging scheme, a page is only brought in to the memory system only when it is needed or to be more precise, only when that page is in demand. This concept is called demand paging.

21. What is tertiary memory?
Answer:
These are the memories used to store removable files and bulk data. For example, the magnetic tapes are tertiary memory. Such memories are very slow.