PGCA/Sc IT/PGDCA (S2) 01
Exam. Code: COA

Computer Organization and Architecture

SEMESTER II

MASTER OF COMPUTER APPLICATION / MASTER OF SCIENCE IN INFORMATION TECHNOLOGY / POST GRADUATE DIPLOMA IN COMPUTER APPLICATION

Block 2



KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakati Mahanta, Gauhati University
Prof. (Retd.) Pranhari Talukdar, Gauhati University
Dr. Jyotiprokash Goswami, Assam Engineering College
Prof. Sikhar Kumar Sarma, Cotton University

Course Co-ordinators

Dr. Tapashi Kashyap Das, KKHSOU
Ms. Sruti Sruba Bharali, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
8, 9, & 11	Ms. Jonalee Barman Kakati, NERIM
10	Dr. Kshirod Sarmah, Pandit Deendayal Upadhyaya Adarsha Mahavidyalaya
12 & 15	Dr. Pranab Das, Assam Don Bosco University
13 & 14	Ms. Shobhanjana Kalita, Tezpur University

Editorial Team

Content : Dr. Jyotiprokash Goswami, Assam Engineering College

Structure, Format & Graphics: Dr. Tapashi Kashyap Das, KKHSOU

April, 2019 ISBN: 978-93-89123-01-2

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State University is made available under a Creative Commons Attribution-Non Commercial-ShareAlike4.0 License (International): http.//creativecommons.org/licenses/by-nc-sa/4.0.

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters : Patgaon, Rani Gate, Guwahati-781 017

City Office : Housefed Complex, Dispur, Guwahati-781 006; Web: www.kkhsou.in

The university acknowledges with thanks the financial support provided by the **Distance Education Bureau, UGC** for the preparation of this study material.

MASTER OF COMPUTER APPLICATION / MASTER OF SCIENCE IN INFORMATION TECHNOLOGY / POST GRADUATE DIPLOMA IN COMPUTER APPLICATION

Computer Organization and Architecture

Block-2

DETAILED SYLLABUS

		Pages
UNIT 8:	**Introduction to Cache and Virtual Memory** Memory Hierarchy, Semiconductor Memories, Internal Organization of Typical RAM and ROM Memory, Switches, Cache Memory, Cache Memory Access Techniques; Mapping Functions, Virtual Memory, Locality of Reference, Paging, Cache Coherence Problem	**147-190**
UNIT 9:	**Memory and I/O Access** Memory Read Write Operations, Concept of Handshaking, Polling Techniques (Serial and Hub Polling) and Interrupt Driven I/O, Priority and Daisy Chaining Technique, Introduction to Memory Mapping	**191-203**
UNIT 10:	**Memory and I/O Interfacing** I/O Processor, Priority Encoder, Device Scheduler, Interfacing with the I/O Devices, Keyboard, Printer and Display Interfaces	**204-218**
UNIT 11:	**External Memory** Magnetic Disk, Magnetic Read Write Operation, Sector, Track, Inter Track and Inter Sector Gap, Cylinder, Fixed Head Disk, Seek Time, Access Time, Transfer Time, Rotational Delay, RAID, Optical Memory, Magnetic Tap	**219-241**
UNIT 12:	**Processor Structure and Function** Processor Organization: Fetch Instruction, Interpret Instruction, Fetch Data, Process Data, Write Data, Registrar Organization, Control and Status Registrar, Instruction Cycle, Instruction Pipelining (Definition only)	**242-257**
UNIT 13:	**Introduction to Parallel Processing** Introduction to Pipelining and Basics of Parallel Processing, Scalable Architecture	**258-268**

UNIT 14: CISC and RISC Architecture **269-276**

CISC (Complex Instruction Set Computers), RISC (Reduced Instruction Set Computers), Examples of CICS and RISC

UNIT 15: Parallel Architectures **277-296**

SISD, SIMD, MISD, MIMD, Scalar, Flynn's Classification of Computer Architecture, Vector, Superscalar and Pipelined Processor, Pipelining,Vector Computing, Arithmetic Pipeline, Vector and Scalar Register, Chaining, Scatter Gather Operations, Vector-Register Processor, Memory-Memory Vector Processor. Array Processor

BLOCK INTRODUCTION

This is the second block of the course ***Computer Organization and Architecture***. This block contains eight units.

Unit 8 mainly discusses cache and virtual memory. Various concepts such as mapping functions, paging etc.are described in this unit

Unit 9 is on memory and I/O access. This unit discusses handshaking concept, various polling techniques, interrupt driven I/O, Priority and Daisy Chaining techniques etc.

Unit 10 describes memory and I/O Interfacing. Various components like I/O processor, priority encoder, device scheduler etc. and their functions are discussed in this unit

Unit 11 descusses various external memories like magnetic disk, optical memory etc.along with their working priciples.

Unit 12 descibes processor structure and function. Processor organization, registar organization, instruction cycle, instruction pipelining are discussed in this unit.

Unit 13 deals with parallel processing. Pipelining and its different hazards are described in this unit.

Unit 14 deals with CISC and RISC architectures. The CISC and RISC architectures are discussed in detail along with examples in this unit.

Unit 15 is the last unit of this block. This unit describes parallel architecture. Flynn's classification of computer architecture, Pipelining, vector computing etc.are discussed in this unit.

Each unit of these blocks includes some along-side boxes to help you know some of the difficult, unseen terms. Some "EXERCISES" have been included to help you apply your own thoughts. You may find some boxes marked with: "LET US KNOW". These boxes will provide you with some additional interesting and relevant information. Again, you will get "CHECK YOUR PROGRESS" questions. These have been designed to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with "ANSWERS TO CHECK YOUR PROGRESS" given at the end of each unit.

UNIT 8: INTRODUCTION TO CACHE AND VIRTUAL MEMORY

UNIT STRUCTURE

8.1 Learning Objectives

8.2 Introduction

8.3 Memory Hierarchy

8.4 Main Memory

8.5 Semiconductor RAM

- 8.5.1 Static and Dynamic RAM
- 8.5.2 Internal Organization of Memory Chips

8.6 ROM

- 8.6.1 Types of ROM

8.7 Locality of Reference

8.8 Cache Memory

- 8.8.1 Cache Operation– an overview

8.9 Mapping Functions

- 8.9.1 Direct Mapping
- 8.9.2 Associative Mapping
- 8.9.3 Set-Associative Mapping

8.10 Replacement Algorithm

8.11 Virtual Memory

- 8.11.1 Paging

8.12 Magnetic Disk

- 8.12.1 Data Organization in Magnetic Disk
- 8.12.2 Disk Access Time

813 RAID

8.14 Optical Memory

- 8.14.1 CD-ROM
- 8.14.2 DVD Disks

8.15 Magnetic Tape

8.16 Let Us Sum Up

8.17 Further Reading

8.18 Answers to Check Your Progress

8.19 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about main memory
- describe the role of RAM, ROM and their types
- learn the concept of locality of reference
- learn about cache and virtual memory
- describe different types of mapping techniques
- decribe the mechanism of paging
- illustrate the organization of data in magnetic disks as well as and magnetic tapes
- learn about RAID technology
- describe the technology of optical memory.

8.2 INTRODUCTION

In the previous unit we learn about various techniques with the help of which data transfers between the main memory and the input-output devices takes place.

In this unit we shall discuss about various types of memory associated with a computer system including main memory, cache and virtual memory and various technology associated with these memory units. Finally, we conclude the unit discussing the concept of secondary memory along with their types.

8.3 MEMORY HIERARCHY

The computer stores the programs and the data in its memory unit. The CPU fetches the instructions out of the memory unit to execute and process them.

Memory can be primary (or main) memory and secondary (or auxiliary) memory. Main memory stores programs and data currently executed by the

CPU of a computer. Auxiliary memory provides backup storage of information. Data as well as instructions are transferred from the secondary memory to the main memory whenever it is needed by the CPU.

The capacity of the memory is typically expressed in terms of bytes or words (1 byte = 8 bits). Word lengths are commonly 8 bits, 16 bits and 32 bits. The size of a word is generally the number of bits that are transferred at a time between the main memory and the CPU.

Memory has different locations, which are called its addresses, to store the data. There are different methods for accessing those address locations such as ***sequential access***, direct access and ***random access***.

- In ***Sequential access*** method, the records or the data are accessed in a linear fashion, from its current location in the memory to the desired location moving through each and every record in the memory unit. For example , in case of the magnetic tapes this method is used.
- In ***Direct Access***, each record has different addresses based on the physical location of the memory and the shared Read/Write head moves directly to the desired record. This method is used in magnetic disks.
- In ***Random access*** each location can be randomly selected and accessed directly. Main memory can be randomly accessed.

Memory has two basic operations: *Read* and *Write* operations. In Read operation, the processor reads data from a particular memory location and transmits that data to the requesting device via bus. On the other hand, a memory Write operation causes the memory to accept data from a bus and to write that particular information in a memory location.

Regarding the speed of the memory, there are two useful measures of the speed of memory units: *Memory Access Time* and *Memory Cycle Time*.

Memory Access Time is the time between the initiation of a memory operation and the completion of that operation. Memory Cycle Time is the minimum time delay that is required between the initiation of two successive memory operations, for example between two successive memory read operations.

The computer system has a memory hierarchy consisting of the storage devices in it. A typical memory hierarchy is illustrated in the figure below:

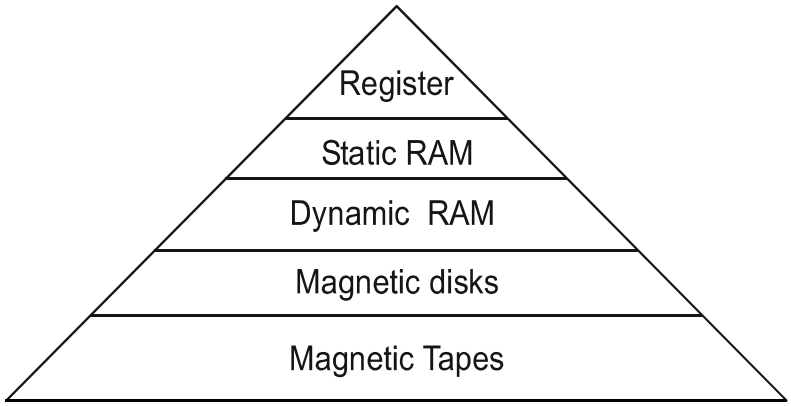


Figure 8.1: Memory Hierarchy

There are three key characteristics of the memory. They are *cost*, *capacity* and *access time*. On moving down the memory hierarchy, it is found that the cost of the storage devices decreases but their storage capacity as well as the memory access time increases. In other words, the smaller memories are more expensive and much faster. These are supplemented by the larger, cheaper and slower storage devices.

Thus from the above figure it can be seen that the registers are at the top of the hierarchy and so provides the fastest, the smallest and the most expensive type of memory device for the CPU to access data. Registers are actually small amount of storage available on the CPU and their contents can be accessed more quickly than any other available storage. They may be 8–bit registers or 32-bit registers according to the number of bits that they can hold in them.

Magnetic disks and Magnetic Tapes are the secondary storage mediums whose data holding capacities are much larger than the Processor Registers and the semiconductor memories which cannot hold all the data to be stored in the computer system. The magnetic tapes are more suited for the off-line storage of the large amounts of the computer data. The data are kept as records which are again separated by gaps.

3.4 MAIN MEMORY

The main memory is the central storage unit of the computer system. Main memory refers to the physical memory which is internal to a computer.

The word “Memory” when used usually refers to the Main Memory of the computer system. The computer can process only those data which are inside the main memory. For execution, the programs and the data must be first brought into the main memory from the storage device where they are stored. Computer memory has a crucial role in the performance, reliability and the stability of the system. It is also an important factor in the software support of the system. More number of software can be used with more memory than with lesser ones. There are two types of main memory :

- ***RAM*** (Random Access Memory) and
- ***ROM*** (Read Only Memory).

- **RAM:** In RAM, it is possible to both read and write data from and to the memory in a fixed amount of time independent of the memory location or address. RAM is also a volatile memory which means it stores the data in it as long as power is switched on. Once the power goes off, all the data stored in it is also lost. Therefore, a RAM cell must be provided a constant power supply.
- **ROM:** ROM is a non-volatile semiconductor memory; that is, it doesn’t lose its contents even when the power is switched off. ROM is not re-writable once it has been written or manufactured. ROM is used for programs such as bootstrap program that starts a computer and load its operating system.

8.5 SEMICONDUCTOR RAM

The basic building block of the semiconductor memories is the RAM chip. RAM is actually made up of a number of RAM chips. They contain a number of memory cells, which are electronic circuits having two stable states: 0 and 1. The binary information are stored in the form of arrays having rows and columns in the memories. With the advent and advances of VLSI (Very Large Scale Integration) circuits, thousands of memory cells can be placed in one chip. As a result, the cost of the semiconductor memories has dropped dramatically.

8.5.1 Static and Dynamic RAM

There are two main types of semiconductor RAM Memories: Static RAM (SRAM) and Dynamic RAM (DRAM) and also their variations.

Static RAM consists of internal flip-flops to store the binary information. Each flip-flop has four to six transistors in them. SRAM can hold its data as long as the power is supplied to the circuit. Also SRAM cells can keep the data intact without any external refresh circuitry. This fact makes SRAM simple and contrasted to Dynamic RAM, which needs to be refreshed many times per second in order to hold its data contents.

The figure below shows the implementation of a SRAM cell. A latch is formed by cross-connecting two inverters. The latch is then connected to two bit lines by the transistors T1 and T2. T1 and T2 are controlled by a word line. They are in the off-state when the word line is at the ground level.

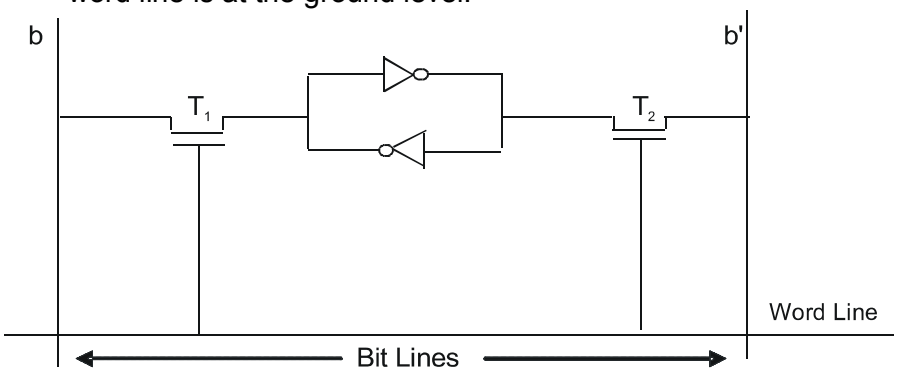


Figure 8.2: Implementation of SRAM Cell

Static RAM is so called as they can retain their state as long as the power is applied. As SRAM never has to be refreshed, it is very fast. However, because it requires several transistors, it takes up more space on a RAM chip than the Dynamic RAM cells. Thus, there is less memory per chip which makes it lot more expensive. The size of SRAM is so larger comparatively than DRAM. In other words performance-wise SRAM is superior to the DRAM. But because of the size and cost of the SRAMs, DRAM is used for the system

memory or main memory instead and SRAM are used for Cache memory as cache memory needs to be more faster and small. D-type and RS-type flip-flops are generally used for SRAM.

Dynamic RAM are so named because their cells do not retain their states indefinitely. DRAM stores the information in the form of a charge on capacitors. The capacitor holds a charge if the bit is a "1" and holds no charge if the bit is a "0". Unlike SRAM, DRAM uses only one transistor to read the contents of the capacitor. The capacitors are very tiny and can hold a charge only for a short period of time, after which it starts fading away. Therefore a refresh circuitry is required in case of DRAM cells to read the contents of the cell and to refresh them with a fresh charge before the contents are lost. This refreshing of the cells are done hundreds of time in every second irrespective of whether the computer is using the DRAM memory at that time or not. So the DRAM is slower than the SRAM just because of the refresh circuitry overhead.

DRAMs are used for the computer's main memory system as they are cheaper and take up much less space than the SRAM. Even though there is the overhead of the refresh circuitry, it is but possible to use a large amount of inexpensive main memory. The figure given below is a DRAM cell consisting of a capacitor C and a transistor T.

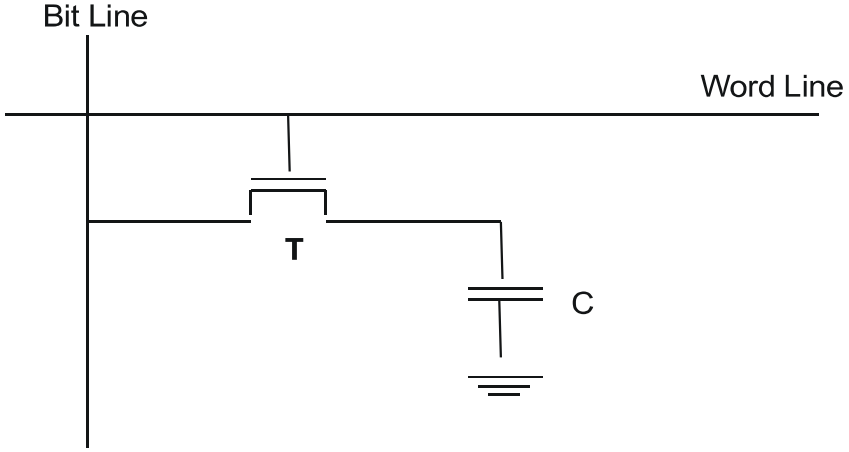


Figure 8.3: A DRAM Cell

To store the binary information in this cell, the transistor T is first turned on and voltage is applied to the bit line. This causes the capacitor to get some amount of charge. After the transistor is turned off, the capacitor begins to discharge. Therefore, the cell contents can be got correctly only if it is read before the capacitor's charge falls below some threshold value. During a Read operation, the transistor in a DRAM cell is first turned on. To check whether the amount of charge on the capacitor is above the threshold value, there is a sense amplifier connected to the bit line. If so, the amplifier pulls the bit line to full voltage to represent logical 1. This voltage then recharges the capacitor to its full voltage.

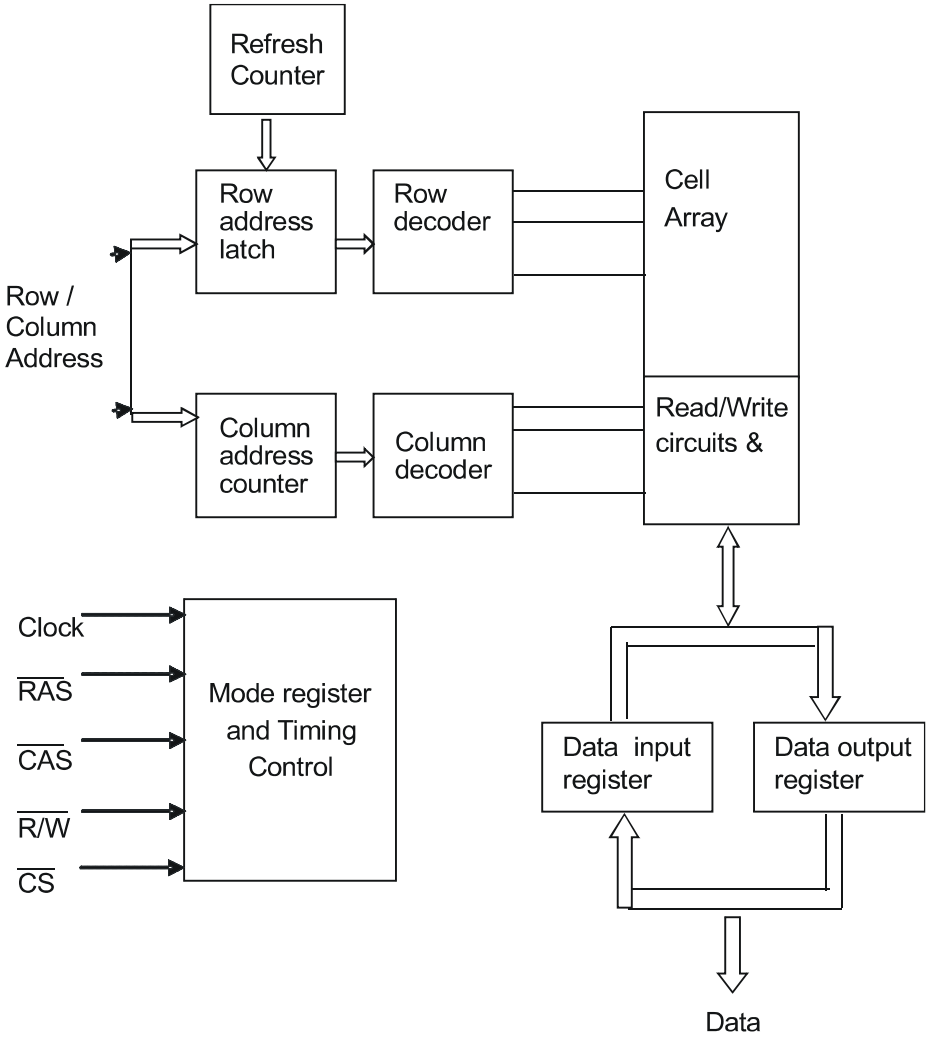


Figure 8.4: Synchronous DRAM

On the other hand, if it was detected that the amount of charge on the capacitor is below the threshold value, then the bit line is pulled down to the ground level so that the capacitor now has no charge at all, that is it will represent logical 0. Thus, reading the cell contents automatically refreshes its cell contents. The time taken in doing all this is very short and is expressed in nanoseconds.

DRAM can again be Synchronous DRAM and Asynchronous DRAM. The DRAM discussed above is Asynchronous DRAM, that is, the memory is not synchronized to the system clock. The memory signals are not at all coordinated with the system clock.

The Synchronous DRAM or SDRAM is synchronized with the system clock; that is, it is synchronized with the clock speed of the microprocessor. All signals are according to the clocks so the timings are controlled and tight. The figure below shows the structure of SDRAM.

The speed of the DRAM has become more critical as the electronic systems that utilizes these devices are now operating at increasing speeds. Thus SDRAM will soon be replacing the conventional DRAM as it is designed to work with higher operating bus speeds in the computer systems.

8.5.2 Internal Organization of Memory Chips

Internally, the memory in the computer system is organized in the form of array of rows and columns. Each cell in the array can hold one bit of information. Each row in the array forms a memory word. The cells in the column are all connected to a Read/Write circuit. The figure below presents a possible organization of the memory cells.

Let us consider a simple example of the organization of a 64-bit memory. If the memory is organized into 16 groups or words, then it can be arranged as 16 x 4 memory, that is, it contains 16 memory words and each of the word is 4 bits long. There are also other ways of organizing the same memory, like for example, it can be arranged as 64 x 1, 32 x 2 or 8 x 8.

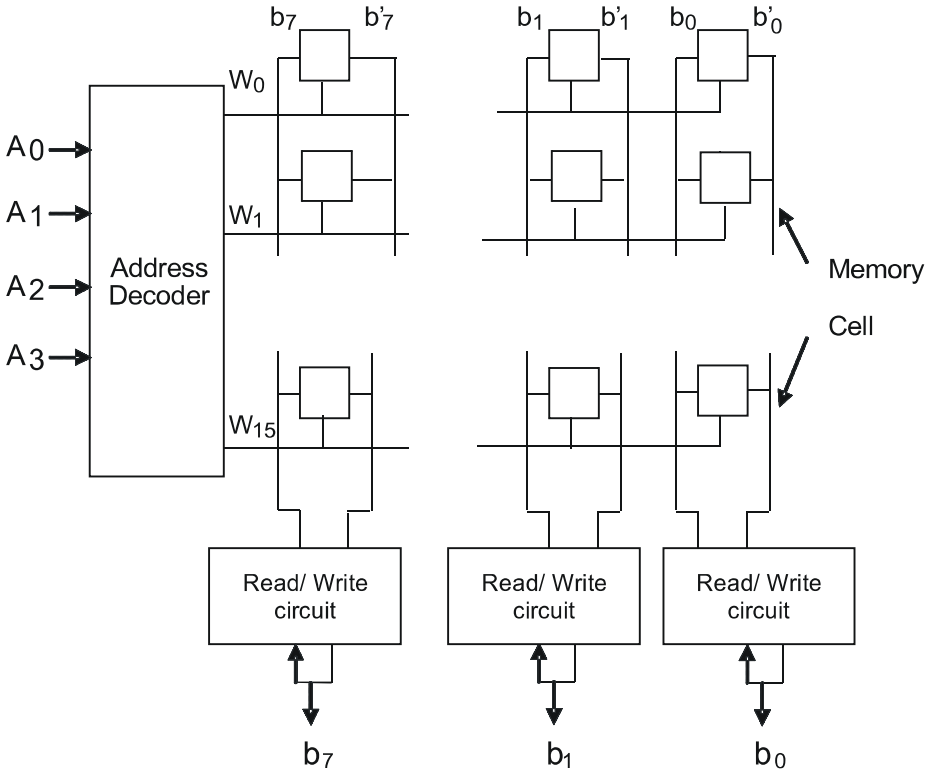


Figure 8.5: Organization of Cells in a Memory Chip



CHECK YOUR PROGRESS

Q.1: Compare the characteristics of SRAM and DRAM.

.....

.....

.....

.....

.....

Q.2: Fill in the blanks:

a) Data and ..... are transferred from the secondary memory to the ..... whenever it is needed by the CPU.

b) The records or the data are accessed in a linear fashion, from its current location in the memory in ..... access.

c) Memory ..... time is the time between the initiation of a memory operation and the completion of that operation.

d) Memory ..... time is the minimum time delay that is required between the initiation of two successive memory operations.

e) Memory access time is ..... in Magnetic disks than in magnetic tapes.

f) Registers are small storage inside the .....

g) RAM can access data in a fixed amount of time ..... of the memory location or address.

h) RAM is a ..... memory and ROM is a ..... memory.

i) Registers are measured by the number of ..... that they can hold.

j) The magnetic tapes are more suited for the ..... of the large amounts of the computer data.

8.6 ROM

ROM (Read Only Memory) is another type of main memory that can only be read. Each memory cell in ROM is hardware preprogrammed during the IC (Integrated Circuit) fabrication process. That is the code or the data in ROM is programmed in it at the time of its manufacture. The data stored in ROM is not lost even if the power is switched off. For this reason, it is called a non-volatile storage. It is used to store programs that are permanently stored and are not subject to change. The system BIOS program is stored in ROM so that the computer can use it to boot the system when the computer is switched on.

The figure below shows a possible configuration for a ROM memory cell.

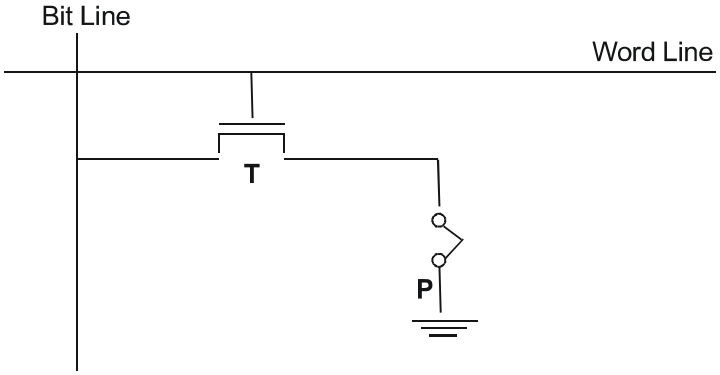


Figure 8.6: A ROM Memory Cell

If the point at P is connected to ground, then a logic value 0 is stored in the cell and if it is not connected then a logic value 1 is stored in the cell. To read the cell contents, the word line is activated. A sense circuit connected at the end of the bit line generates the output value.

ROM can also be randomly accessed as RAM. Only unlike RAM it cannot be written.

8.6.1 Types of ROM

There are five types of ROM:

1) ROM (Read Only Memory)
2) PROM (Programmable Read Only Memory)
3) EPROM (Erasable Programmable Read Only Memory)
4) EEPROM (Electrically Erasable Programmable Read Only Memory)
5) Flash EEPROM Memory

1) ROM (Read Only Memory): The contents of ROM are permanent and is programmed at the factory. It is designed to perform a specific function and cannot be changed. It is reliable.

2) PROM (Programmable Read Only Memory): As the name indicates, this type of ROM chips allows the user to code data into it. They were created as making ROM chips from scratch. It is time-consuming and also expensive in creating small numbers of them. PROM chips are like ROM chips but the difference is that PROM chips are created by inserting a fuse at the point P

(in the above diagram). Before programming, all the memory cells in PROM contains 0. The user then can insert 1's wherever needed by burning out the fuses at those locations by sending high current pulses. However, PROMs can be programmed only once. They are more fragile than ROMs.

3) **EPROM (Erasable Programmable Read Only Memory):** An EPROM is another type of ROM chip that allows data to be erased and reprogrammed. They can be rewritten many times. It is also similar to a ROM chip. However, an EPROM cell has two transistors. One of the transistors is known as the floating gate and the other is known as the control gate. In an EPROM cell, the connection to ground is always made at point P. The erasing of the contents is done by exposing the chip to ultraviolet light. EPROM chips are mounted in packages that has a small glass window through which the UV light is sent into it.

4) **EEPROM (Electrically Erasable Programmable Read Only Memory):** The drawbacks of EPROMs are that they must be physically removed to be rewritten and also the entire chip has to be completely erased to just change a particular portion of it. EEPROM was introduced to remove these drawbacks of EPROM. EEPROM chips can be both programmed and the contents can be erased electrically. They are versatile but slow as they are changed one byte at a time.

5) **Flash EEPROM Memory:** As EEPROM is slow to be used in products that have to make quick changes to the data on the chip, so the Flash EEPROM devices were developed. Although the flash memory devices are similar to EEPROM, there are also differences between them. In EEPROM, a single cell contents can be read and written while in flash memory single cell contents can be read but it is possible to write an entire block of cells. Also, before writing, the previous contents of the block are erased.

The advantage is that they work faster and the power consumption is low.



CHECK YOUR PROGRESS

Q.3: Write True or False:

a) SRAM and DRAM are the two types of semiconductor memories.

b) SRAM stores the data in flip-flops and DRAM stores the data in capacitors.

c) Main memory is actually the Static RAM.

d) DRAM is more expensive as compared to SRAM.

e) The capacitors have their own tendency to leak their charge.

f) Conventional DRAM is the Synchronous DRAM.

Q.4: Fill in the blanks:

a) A refresh circuitry is required in case of .....

b) SRAM are used for ..... memory and DRAM is used for ..... memory

c) DRAM uses one transistor to read the contents of the .....

d) The type of RAM that can hold its data without external refresh for as long as power is supplied to the circuit is called .....

e) SDRAM is synchronized to the .....

f) ..... is rapidly becoming the new memory standard for modern PC's.

8.7 LOCALITY OF REFERENCE

During program execution, memory access time is of utmost importance. It has been observed that data and instructions which are executed repeatedly are located near to each other. Many instructions in localized areas of the program are executed repeatedly during some time period and the other remaining instructions of the program are accessed relatively infrequently. This characteristic of the program is referred to as

the ***locality of reference***. There are two types of locality of references. These are: *temporal locality* and *spatial locality*.

- ***Temporal locality of reference*** means that a recently executed instruction is likely to be executed again very soon. This is so because when a program loop is executed the same set of instructions are referenced and fetched repeatedly. For example, loop indices, single data element etc.
- ***Spatial locality of reference*** means that data and instructions which are close to each other are likely to be executed soon. This is so because, in a program, the instructions are stored in consecutive memory locations. For example, sequential code, array processing, code within a loop etc.

8.8 CACHE MEMORY

To make use of the locality of reference principle, a small high-speed memory can be used to hold just the active portions of code or data. This memory is termed as ***cache memory***. The word *cache* is pronounced as *cash*. It stores data and instructions which are to be immediately executed. Two types of caching are commonly found in computers. These are : *memory* caching and *disk caching.*

- **Memory Caching:** In case of memory caching, the cache is made up of high-speed static RAM (SRAM). Static RAM is made up of transistors that do not need to be constantly refreshed. It is much faster than the dynamic RAM; access time is about 10 nanoseconds(ns). The main memory is usually made up of dynamic RAM(DRAM) chip. It is directly addressed by the CPU and its access time is about 50 ns. Data and instructions stored in cache memory are transferred to the CPU many times faster as compared to main memory. By using an intelligent algorithm, a cache contains the data that is accessed most often between a slower peripheral device and the faster processor. SRAM chips are much more expensive as compared to DRAM chips. If the whole main memory is made using SRAM chips, then the need of cache memory will be eliminated.

  Some memory caches are built into the architecture of micro-processors. These are called ***internal cache***. For example, the *Intel 80486*

microprocessor contains a 8K memory cache and the *Pentium* has a 16K cache. Such internal caches are often called **Level 1(L1)** caches. Cache outside the microprocessor i.e., on the motherboard is called ***external cache*** *or* **Level 2(L2)** cache. External caches are found in almost all modern personal computers. These caches are placed between the CPU and the DRAM. Like L1 caches, L2 caches are composed of SRAM but they are much larger.

- **Disk Caching:** Disk caching works under the same principle as memory caching, but instead of using high-speed static RAM, it uses dynamic RAM. The most recently accessed data from disk is stored in the main memory which is made up of DRAM. When a program needs to access data from the disk, it first checks the disk cache to see if the data is there. Disk caching can improve the performance of applications, because accessing a byte of data in DRAM can be thousands times faster than accessing a byte on a hard disk.

8.8.1 Cache Operation– an Overview

When the CPU sends an address of instruction code or data, the *cache controller* examines whether the content of the specified address is present in the cache memory. If the requested data or instruction code is found in the cache, the cache controller enables the cache memory to send the specified data/instruction to the CPU. This is known as a '**hit**'. If it is not found in the cache memory, then it is said that a '**miss**' has occured and the cache controller enables the controller of the main memory to send the specified code or data from the main memory. The performance of cache memory is measured in terms of ***hit ratio***. It is the ratio of number of hits divided by the total number of requests made. By adding number of hits and number of misses, the total request is calculated.

It is not necessary for the processor to know about the existence of cache. Processor simply issues READ and WRITE requests using addresses that refer to locations in the memory. Both the main memory and the cache are divided into equal-size units

called *blocks*. The term *block* is usually used to refer to a set of contiguous address locations of some size. In a **READ** operation, the main memory is not involved. When a READ request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache, one word at a time. Subsequently, when the program references any of the locations in this block, the desired contents are read directly from the cache. Usually, the cache memory can store a reasonable number of blocks at any given time.The correspondence between the main memory blocks and those in the cache is specified by a *mapping function*. When the cache is full, it becomes necessary to implement a *replacement algorithm*. The replacement algorithm decides which block should be moved out of the cache to make room for the new block. Normally, the block that will be replaced is the one that will not be accessed or needed again for the longest time. Cache provides a timer to control this situation.

When the memory operation is a **WRITE**, there are two ways to proceed: *write-through method* and *write-back method*.

- **Write-through method:** In this method, the cache location and the main memory location are updated simultaneously.
- **Write-back method:** This method is to update only the cache location and to mark it as updated with an associated *flag bit* also known as *dirty bit* . Here, the update of main memory occurs only when the block containing this marked word is to be removed from the cache to make room for a new block.



CHECK YOUR PROGRESS

Q.5: Choose the appropriate option:

i) The cache is made up of high-speed ..... in case of memory caching.

a) EPROM b) SRAM

c) ROM d) none of these

ii) Cache memory is used in computer sytem to:
   a) Replace static RAM b) Replace ROM
   c) Ensure first booting d) Speed-up memory access

iii) Compared with primary storage, secondary storage is:
   a) fast and inexpensive b) fast and expensive
   b) slow and inexpensive d) slow and expensive

Q.6: Fill in the blanks:

i) ..... occurs when the requested word is not present in the cache memory.

ii) ..... states that data and instruction which are close to each other are likely to be accessed in near future.

iii) Level 2 cache are also termed as ..... cache.

iv) The performance of cache memory is measured in terms of .....

8.9 MAPPING FUNCTIONS

To search for a word in the cache memory, cache is mapped to the main memory. There are three different ways that this mapping can generally be done. These are:

- *Direct Mapping*
- *Associative Mapping*
- *Set-Associative Mapping*

Mapping functions are used as a way to decide which main memory block occupies which line of cache. As there are less lines (or block) of cache than main memory blocks, an algorithm is needed to decide this. Let us take an example, a system with a cache of 2048 (2K) words and 64K (65536) words of the main memory. Each block of the cache memory is of size 16 words. Thus, there will be 128 such blocks (i.e., 16*128 = 2048). Let the main memory is addressable by 16 bit address (i.e., 2^{16} = 65536 = 64*1024).

8.9.1 Direct Mapping

The simplest way for determining the cache location for placement of a main memory block is the *direct mapping* technique.

Here, the ***block i*** of the main memory maps onto ***block i modulo 128*** of the cache memory. For example,

Block 0 of main memory = 0 %128 = 0, i.e., Block 0 of cache memory

Block 128 of main memory = 128%128 = 0, Block 0

Block 256 of main memory = 256%120 = 0, i.e., Block 0

Block 1 of main memory = 1%128 = 1, i.e., Block 1

Block 129 of main memory = 129%128 = 1, i.e., Block 1

Block 257 of main memory = 257%128 = 1, i.e., Block 1

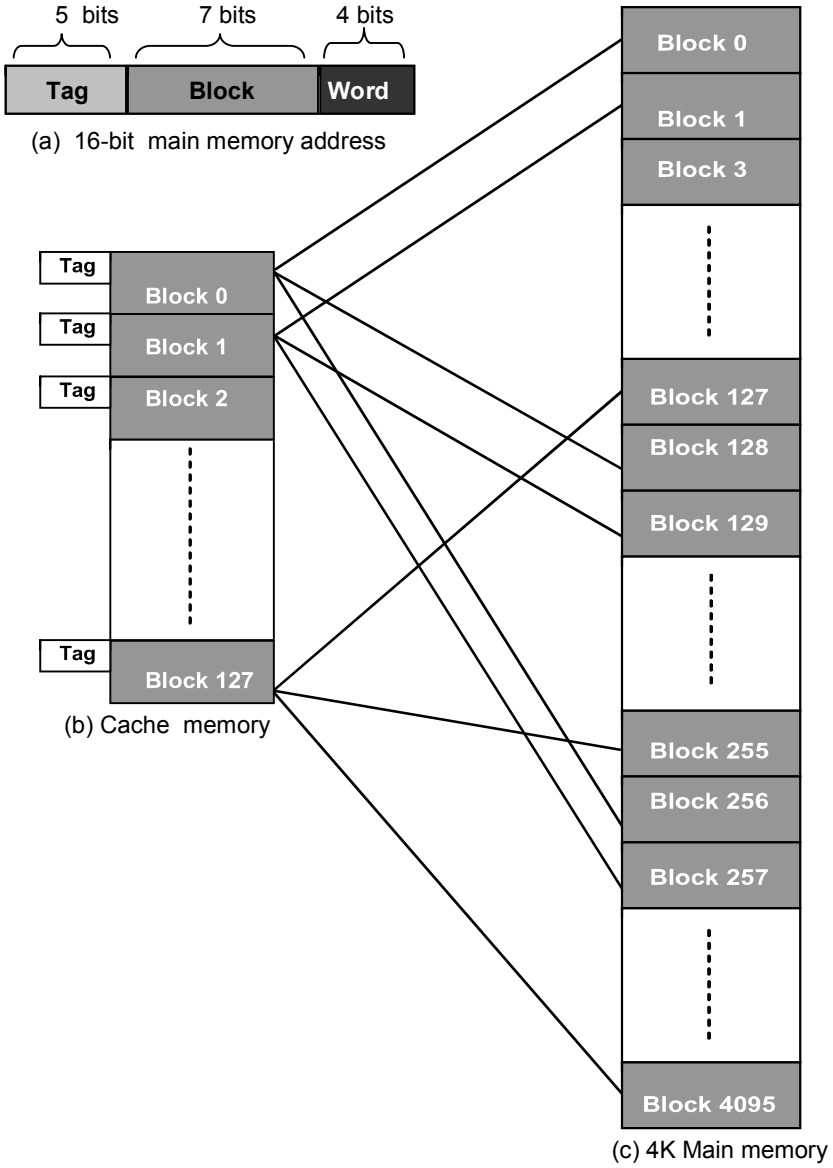


Figure 8.7: Direct Mapping Technique



Cache line (*or* ***Cache block):*** It is the smallest unit of memory that can be transferred between the main memory and the cache. The cache line is generally fixed in size, typically ranging from 16 to 256 bytes.

Thus, whenever one of the main memory *blocks 0, 128, 256, ...* is loaded in the cache, it is stored in cache *block 0*. *Block 1, 129, 257*, ... are stored in cache *block 1*, and so on. Since more than one main memory block is mapped onto a particular cache block, contention may arise for that position even when the cache is not full. Currently resident block in the cache is overwritten by new block. The detailed operation of the direct mapping technique is figure 8.7.

In the figure 8.7(c), the 64K words main memory is viewed as 4K (4096) blocks of 16 words each. In direct mapping technique, the 16 bit address sent by CPU is interpreted a tag, a block and a word field as shown in figure 3.7(a). The low-order 4 bits are used to select one of 16 words in a block. When a new block enters the cache, the 7 bits block field determines the cache position in which this new block can be stored. The high-order 5 bits of the memory address of the block are stored in 5 tag bits associated with its location in the cache. These bits identify which of the 32 blocks that are mapped into this cache block field of each address generated by the processor points to a particular block location in the cache. The tag field of that block is compared to the tag field of the address. If they match, then the desired word specified by the low-order 4 bits of the address is in that block of the cache. If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache.

The advantage of direct mapping is that it is simple and inexpensive. The main disadvantage of this mapping is that there is a fixed cache location for any given block in main memory. If a program accesses two blocks that map to the same cache line repeatedly, then cache misses are very high.

8.9.2 Associative Mapping

This type of mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of cache. To do so, the cache conroller interprets a memory

address as a *tag* and a *word* field. The tag uniquely identifies a block in main memory. For a 16 bit memory address, 12 bits are used as tag field and 4 bits are used as word field as shown in figure 8.8(a).

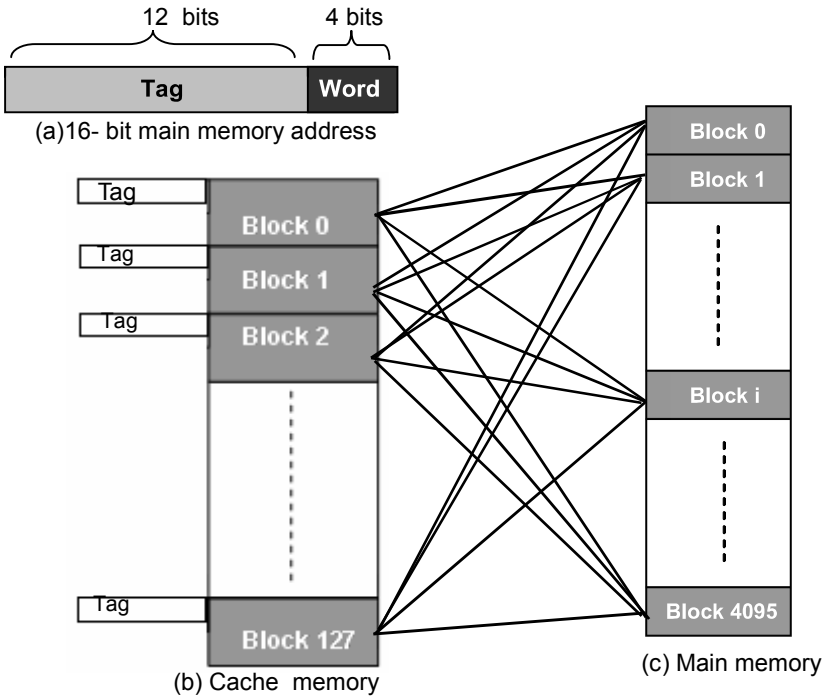


Figure 8.8: Associative Mapping Technique

With this mapping, the space in the cache can be used more efficiently.The primary disadvantage of this method is that to find out whether a particular block is in cache, all cache lines would have to be examined. Using this method, replacement algorithms are required to maximize its potential.

8.9.3 Set-Associative Mapping

Set-associative mapping combines the best of *direct* and *associative* cache mapping techniques. In this mapping, cache memory blocks are grouped into ***sets***. It allows a block of the main memory to reside in any block of a specific set. Thus, the contention problem which usually arise in direct mapping can be avoided by having a few choices for placement of block. In the figure 8.9, set-associative mapping technique is shown. Here, the cache is divided into sets where each set contains 2 cache blocks. Thus, there will

be 64 sets in the cache of 2048(2K) words. Each memory address is assigned a set, and can be cached in any one of those 2 locations within the set that it is assigned to. In other words, within each set the cache is associative, and thus the name set associative mapping.

The 6 bits **set** field of the address determines which set of the cache might contain the desired block. The **tag** field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present. For the main memory and cache sizes shown in figure 3.9, four blocks per set can be accommodated by a *5 bits set* field, eight blocks per set by a *4 bits set* field, and so on. If there is 128 block per set, then it requires no set bits and it becomes a fully associative technique with 12 tag bits. The other extreme condition of one block per set is the direct-mapping method. A cache that has ***N*** blocks per set is referred to as ***N-way set-associative cache***. Figure 3.9(b) is a *2-way set-associative* cache.

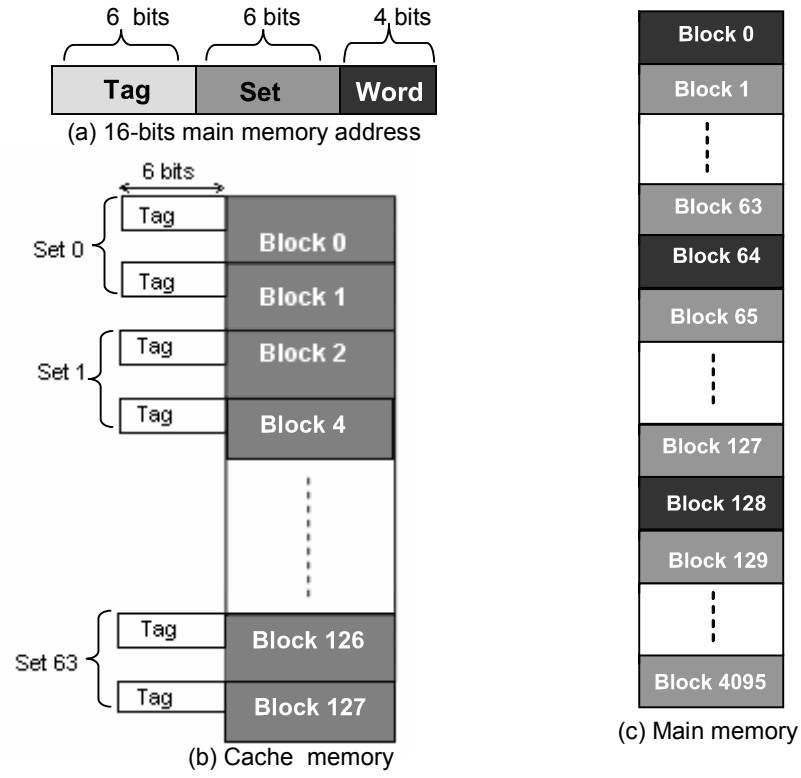


Figure 8.9: Set-Associative Mapping Technique



CHECK YOUR PROGRESS

Q.7: Fill in the blanks:

i) Main memory block can reside in any cache position in case of ..... mapping.

ii) In Set-associative mapping blocks of the cache are grouped into .....

iii) In ..... mapping, main memory address consists of tag, block and word fields.

iv) A cache that has 4 blocks per set is referred to as .....

8.10 REPLACEMENT ALGORITHMS

For direct mapping where there is only one possible line for a block of memory, no replacement algorithm is required. For associative and set associative mapping, however, an algorithm is needed. At this point, we will describe the one most common replacement algorithm, called *LRU algorithm.* This algorithm is easy to understand and provides a good background to understand the more advanced replacement algorithms. Several other replacement algorithms are also used in practice such as: *first in first out* replacement algorithm, *random replacement* algorithm etc.

Least Recently Used (LRU): Due to locality of reference, programs usually stay in localized areas for reasonable period of time. So there is a high probability that the blocks that have been referenced recently will be referenced again soon. Therefore, it overwrites block in cache memory that has been there the longest with no reference to it. That block is known as the least recently used block and the technique is known as ***least recently used*** (***LRU***) replacement algorithm. In this method, a counter is associted with each page in the main memory and it is incremented by 1 at fixed intervals of time. When a page is referenced, its counter is set to 0. Thus, counter gives the age of a page. When a page needs to be removed, the page with the highest counter is removed.

8.11 VIRTUAL MEMORY

Before coming to the point *virtual memory,* let us review some basics: *physical memory(RAM)* versus *secondary memory(disk space)*.

Computers these days typically have somewhere between 128 megabytes (128 million bytes) and 4 gigabytes (4 billion bytes) of main memory (RAM). What is important is that when we turn the computer off or if it crashes– anything stored in RAM is gone. That is why when we are editing a document it is a better to save to disk often. When we talk about *disks*, we are talking about the hard disk address of main memory. In such a case, an address generated by the CPU is called ***virtual address*** or *logical address*. A set of such addresses is called the *address space*. Virtual addresses might be the same as physical addresses. ***Physical address*** refers to the address of a location of the main(physical) memory. A set of physical addresses is termed as *memory space*. If virtual and physical addresses are different, then virtual addresses must be mapped into physical addresses and this mapping is done by Memory Management Unit (MMU).

8.11.1 Paging

Paging is a method for achieving virtual memory. It is the most common memory management technique. Here, the virtual address space and memory space are broken up into several equal sized groups. To facilitate copying the virtual memory into the main memory, the operating system divides *virtual address space* into fixed size ***pages***. Physical address space(memory space) is also broken up into fixed size ***page frames***. *Page* in virtual address space fits into *frame* in physical memory. Each page is stored on a secondary storage (hard disk) until it is needed. When the page is needed, the operating system copies it from disk to main memory(RAM), translating the virtual addresses into physical addresses. This process of copying the virtual pages from disk to main memory is known as ***paging***.

To illustrate these, let us consider a computer with 14 bit address field in its instruction and 4096(4K) words of memory (RAM).

A program on this computer can address 2^{14} = 16384 = 16 K words of memory. Thus the virtual address space is from 0 to 16383. If the address space and the memory space is divided into groups having 1K = 1024 words each, then 16K virtual address space will consist of 16 pages [figure 8.10(a)] and 4K memory space will consist of 4 page frames [figure 8.10(b)].

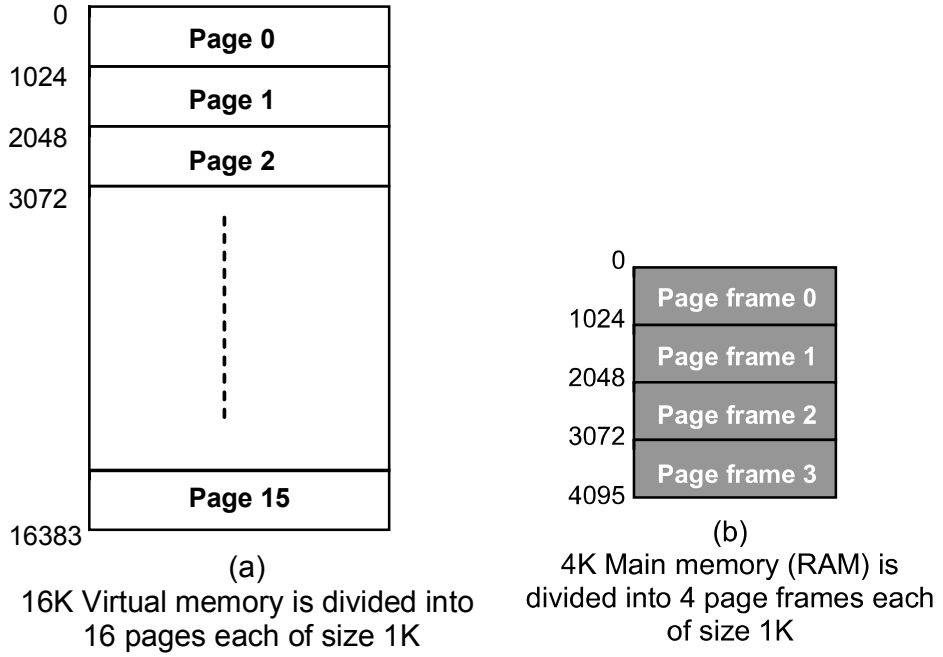


Figure 8.10: A Possible Way to Divide up 16K Address Apace and 4K Memory Space

Out of these 16 pages, only 4 pages can accommodate in the main memory at a time. The physical address must be specified with 12 bits (as 2^{12}= 4096 = 4K). At any instant of time 4096 words of memory can be directly accessible but they need not correspond to address 0 to 4095. For the execution of the program, the 14 bit virtual address needs to be mapped into 12 bit physical address.

When the program refers to memory in fetching data or instruction or to store data, it would first generate a 14 bit address corresponding to a virtual address between 0 and 16383. This 14 bit addresses are interpreted as follows (figure 8.11):

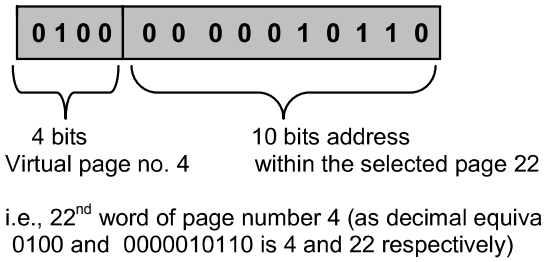


Figure 8.11

Here, 4 bits are shown as virtual page number and 12 bits as address within the selected page. The decimal equivalent of 14 bit address 01000000010110 is 4118, which is interpreted as address 22 of page 4. Having discovered that virtual page number 4 is needed, the operating system must find out where virtual page 4 is located. The page 4 may reside in any of the four page frame in the main memory or somewhere in the secondary memory. To find out where it is, the operating system searches it in the *page table* which is in the main memory. A ***page table*** is a table stored in main memory or in some fast memory which has one entry for each of the virtual pages. The *page table* contains the list of all the 16 pages of virtual memory and the page frame number where the page is stored in the main memory. A ***presence bit*** is also there to indicate whether the page is present in the main memory or not. If the page is present in the main memory then the presence bit will be 1 otherwise it will be 0. For example, if the pages 2, 4, 5, 8 are present in the main memory, the the content of memory page table will be as shown in the figure 8.12.

Let us assume that the virtual page 4 is in main memory. The first 4 bits of a virtual address will specify the *page number* where the word is stored. Similary, the first 2 bits of a physical address will specify the page frame number of the memory where the word is stored. The 4 bit page number is taken from the virtual address and it is compared with the memory page table entry. If the presence bit against this page number is 1, then the block number (2 bits) is taken and is written in the place of page number in the address (figure 8.12). Thus, a 14 bit virtual address is mapped into a 12 bit

physical address. The block number is searched in the main memory and then the word located at that address is fetched.

When a reference is made to an address on a page not present in main memory, it is called a ***page fault***. After a page fault has occured it is necessary for the operating system to read in the required page from the secondary memory, enter its new physical location in the page table and then repeat the instruction that cause page fault.

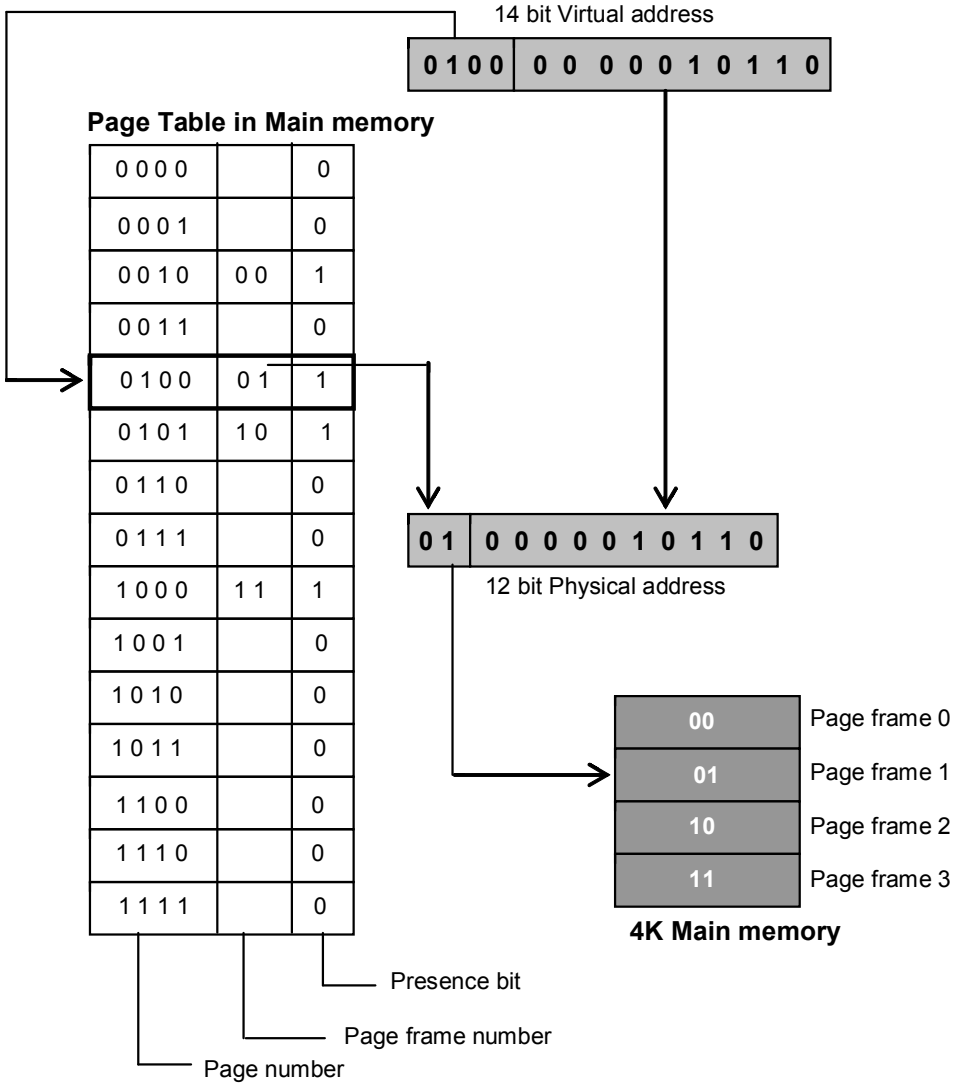


Figure 8.12: Contents of *memory page table* when pages 2, 4, 5 and 8 are present in the main memory and formulation of main memory address from virtual address



CHECK YOUR PROGRESS

Q.8: Fill in the blanks:

i) A page replacement policy is not necessary in ..... mapped cache.

ii) Paging is a method of achieving .....

iii) Virtual address is also known as ..... address.

iv) In paging, memory space is broken up into equal size ..... and address space is broken up into equal size .....

v) Set of physical address is known as .....

vi) Set of virtual address is known as .....

vii) ..... in virtual address space fits into ..... in physical memory

viii) Paging is the most common ..... technique.

8.12 MAGNETIC DISKS

Magnetic disks are the widely used popular secondary storage medium. The magnetic disks are used for storing large amounts of data and programs. A magnetic disk is a circular plate which is constructed of metal or plastic coated with a magnetizable material on which electronic data are stored. Data can be stored on both sides of the disk. Several disks can be stacked on top of the other on a spindle. These disks are actually rotating platters with a mechanical arm that moves a read/write head between the inner and outer edges of the disks surface. A read/write head is available on each of the disk surface. A magnetic disk works on the principle of magnetic charge.

The disks rotates at very high speeds. During a read/write operation, only the disk rotates and the head is always stationary.

8.12.1 Data Organization in Magnetic Disk

Data on the magnetic disks is organized in a concentric set of circles, called tracks. The bits of data are stored as magnetic spots in these tracks. A gap exist between each track to prevent errors due to interference of magnetic fields. The number of bits stored in each track is equal. Therefore, the data density is more in the inner tracks than the outer ones. The more the number of tracks, the more is the storage capacity of the disk.

Tracks are again divided into sections called sectors. A small gap also exists between two sectors so as to distinguish between them. Each sector usually contains 512 bytes of data.

The set of corresponding tracks on all surfaces of a stack of disks forms a **logical cylinder**. The disk platters rotates together at a very high speed.

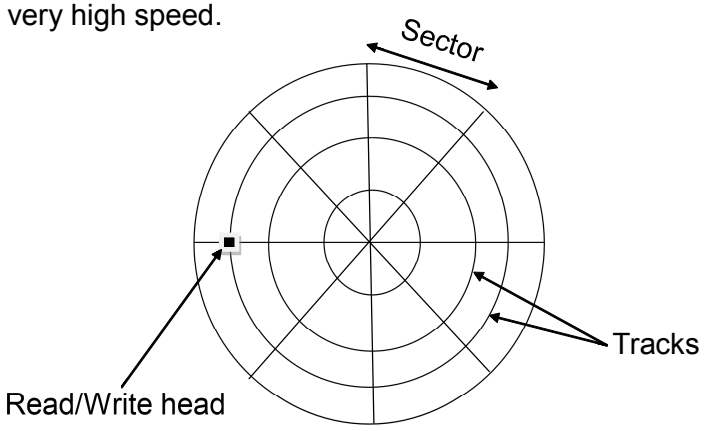


Figure 8.13: Organization of Data on Magnetic Disk

A write head is used to record the data bits as magnetic spots on the tracks and these recorded bits are detected by a change in the magnetic field produced by a recorded magnetic spot, on the disk surface, as it passes through a read head. The data on the disk surfaces are accessed by specifying the surface number, track number and sector number.

Some magnetic disk uses a single read/write head for each disk surface while others uses separate read/write heads for each track on the disk surface. Accordingly, the read/write head may be

movable or fixed. If the magnetic disk uses a single head for each disk surface then the read/write head must be able to be positioned above any track in the surface. Therefore, the head has to be a movable head. In such case, the head is mounted on an arm and the arm can be extended or retracted to position the head on any track.

In a fixed-head disk, there is one read/write head per track in the surface. All the heads are mounted on a rigid arm that extends across all tracks.

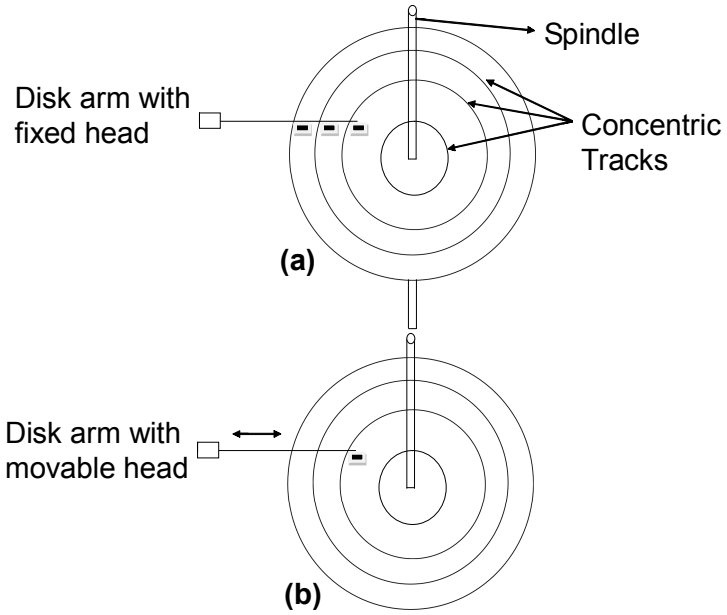


Figure 8.14: (a) Fixed Read/Write Head, (b) Movable Read/Write Head

The read and write operations starts at the sector boundaries. The bits of data are recorded serially on each track. To read or write data, the head must be first positioned to that particular track.

The heads should be always be at very small distance from the moving disk surfaces so that the high bit densities and also due to this more reliable read/write operations can be performed.

Now-a-days, Winchester technology is used where both the disks and the read/write heads are placed in sealed enclosures. This technology has two advantages. First, in such disk units, the heads operates closer to the tracks as there are no dust particles, as in disk units which are not sealed. As such data can be more densely stored along the tracks and the tracks can also be closer to

one another. These Winchester disk units has a larger capacity for storing data.

Secondly, in these disk units, the data integrity is more as they are not exposed to contaminating elements.

The disk system consists of three key parts. First is the stack of disk platters, usually referred to as the **disk.**

Second is the electromechanical mechanism that rotates the disk and moves the read/write heads, which is called the **disk drive.**

Third is the **disk controller**, which controls the disk system operation. It also provides an interface between the disk drive and the bus that connects it to the computer system.

The disks which are attached to the computer units and cannot be removed by the occasional users are called the hard disk. Those which can be inserted and removed from the system easily by the users are called floppy disks.

8.12.2 Disk Access Time

To perform a read or a write operation, the read/write head is first positioned on the desired track and sector. In fixed-head systems the head over the desired track is selected electronically. In a movable head system, the head is positioned on that particular track. In such systems, the time that is taken to move the head to the proper track is called the ***seek time***. It depends on the initial position of the head relative to the track specified in the address.

In either case, after the read/write head is positioned over the track, the system waits until the appropriate sector passes under the read/write head. This delay in time is called the ***rotational delay or latency time***.

The sum of these two delays, that is the seek time and the latency time is called the *disk access time*.

The storage capacity of a disk is a multiple of the number of recording surfaces, number of tracks per surface, number of sector per track, and the number of bytes per sector. That is,

Storage capacity of a disk system = Number of recording surfaces
x Number of tracks per surfaces
x Number of sector per track
x Number of bytes per sector

The following figure shows a typical disk pack. The upper surface of the top plate, and the lower surface of the bottom plate are not used for information storing.

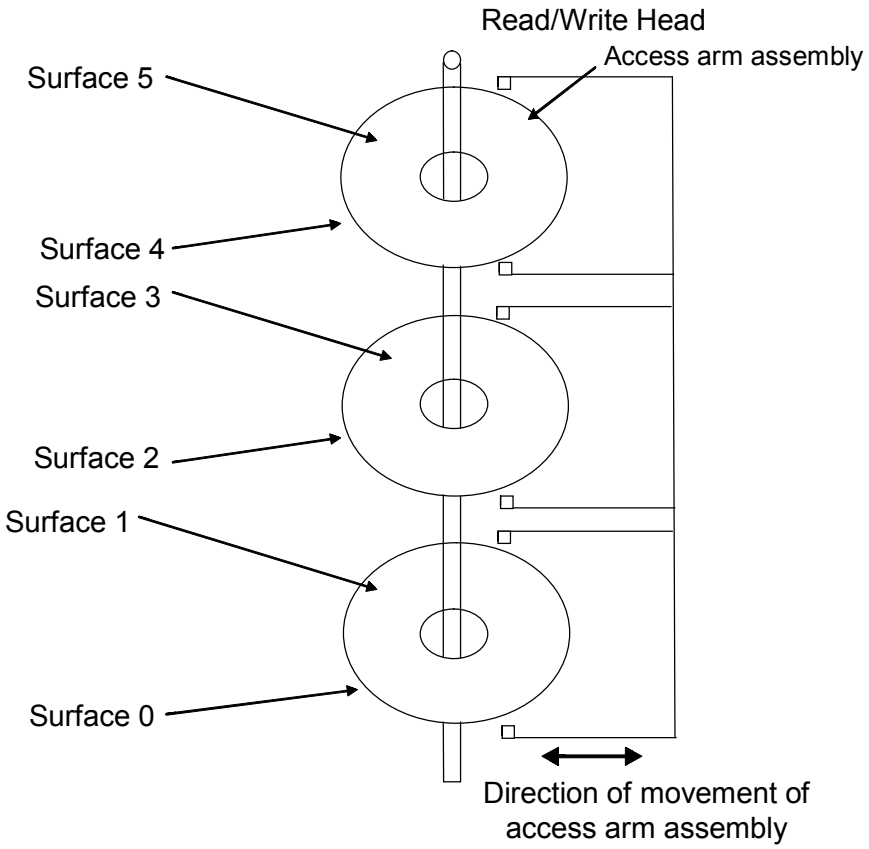


Figure 8.15: Disk Pack



CHECK YOUR PROGRESS

Q.9: What are tracks, sectors and cylinders?

.....

.....

.....

.....

.....

Q.10: Fill in the blanks :

a) The density of data is ..... in the inner tracks and ..... in the outer tracks.

b) The inter-track gap prevents .....

c) Data is recorded and accessed in magnetic disks with the help of a ..... mechanism.

d) Read/write heads may be ..... or .....

e) Data is stored as _____ in the magnetic disks.

f) In case of Winchester disks, the disks and the heads are .....

g) The three vital parts of a disk system are ....., ... and .....

h) ..... is the time required to position the head to the desired track.

i) The seek time and the rotational delay together is called the ..... time.

8.13 RAID

The rate in the increase in the speeds of the processor and the main memory is much more than that in the improvement of the secondary storage devices. Of course there has been an increase in the storage capacities of these devices.

However, it was been recognized that high performance could be achieved at a reasonable cost by using multiple low-cost devices to operate in parallel. This lead to the use of arrays of disks that operate independently and in parallel.

In 1988, researchers at the University of California-Berkeley proposed a multiple disk storage system, called ***RAID (Redundant Array of Inexpensive Disks)***. There were six different configurations, which were known as thr RAID levels– **RAID 0, RAID 1, RAID 2, RAID 3, RAID 4** and **RAID 5**. All of these six levels share three common characteristics. They are–

1) RAID is a set of physical disk drives which operates as one logical drive.
2) The data is distributed across the physical drives of an array.
3) In case of a disk failure, the data can be recovered as identical data is stored on more than one disk in the set of the disks.

In RAID 0, data stripping is used. This means a single file is stored in several separate disks by breaking the file into a number of smaller pieces. Whenever a file is read, all the disks delivers their stored portion of the file in parallel. So the total transfer time of the file is equal to the transfer time that would be required in case of a single disk system divided by the number of disks used in the array.

The disk is divided into strips, which may be physical blocks or some sectors.

In the RAID 1 scheme identical copies of the same data are stored on two disks. Data stripping is also used here but the strip is mapped to both the physical disks. The two disks are mirrors of each other. If a disk failure occurs, then all the operations on its data can be done on its mirror disk. The disadvantage of RAID 1 scheme is the cost involved with it to improve the reliability.

RAID 2, RAID 3, RAID 4 and RAID 5 in all these schemes some parity calculations are done so as to achieve reliability and recover the errors. The data are not fully duplicated RAID 2 requires fewer disks but is still costly. RAID 5 distributes the parity strips across all disks.

RAID offers excellent performance and are generally used in high performance computers or in systems where higher degree of data reliability is required.



CHECK YOUR PROGRESS

Q.11: Find true or false:

a) RAID combines two or more physical hard disks into a single logical unit using special hardware or software.

b) The full form of RAID is Redundant Array of Inexpensive Disks.

c) The levels of the RAID are hierarchy of one another.

d) All the disks in RAID behave as one logical disk drive.

e) In case of a disk failure, the data cannot be recovered in RAID.

f) RAID1 level uses parity calculations for error-recovery.

Q.12: What are the three key concepts in RAID?

.....

.....

.....

.....

8.14 OPTICAL MEMORY

In optical memory, the data is stored on an optical medium like CD-ROM. The stored data can then be read with the help of a laser beam. Large amounts of data can be stored in optical memory at a very reasonable cost. The audio CD (Compact Disk) were the first application of such technology.

In the mid 1980's Sony and Philips companies developed the first generation of CDs. The CDs are non-volatile and they could not be erased.

Optical storage systems consist of a drive unit and a storage medium in a rotating disk form. In general the disks are pre-formatted using grooves and lands (tracks) to enable the positioning of an optical pick-up and recording head to access the information on the disk. Under the influence of a focused laser beam emanating from the optical head, information is recorded on the media as a change in the material characteristics, often using a thermally induced effect. To record a bit, a small spot is generated on the media modulating the phase, intensity, polarization, or reflectivity of a readout optical beam which is subsequently detected by a detector in the optical head. The disk media and the pick-up head are rotated and positioned through drive motors and servo systems controlling the position of the head with respect to data tracks on the disk. Additional peripheral electronics are used for control and data acquisition and encoding/decoding.

In the past few years a variety of optical disk systems have been introduced:

CD (Compact Disk)

CD-ROM (Compact Disk Read Only Memory)

WORM (Write Once Read Many)

Erasable Optical Disk, etc.

8.14.1 CD-ROM

CD-ROM disk is a shiny, silver color metal disk usually of 51/4-inch (12cm) diameter. It is made of polycarbonate plastic and thin layer of pure aluminum is applied to make the surface reflective. For some good quality disks, gold layer is used. A thin layer of lacquer protects it. The surface of an optical disk consists of *pits* and *land*. The information is read from pits and lands, like 1s and 0s. It is changed into binary so computers can read it. An optical reader reads the patterns of pits that stands for bytes. One CD can hold 650MB of data or 300,000 pages of text. Most CDs are read only, which means you cannot save data to the disk. This device is usually not used as a primary storage device for data.

An optical disk is mounted on an optical disk drive for reading/ writing of information on it. An optical disk drive contains all the mechanical, electrical and electronic components for holding an optical disk and for reading/writing of information on it. That is, it contains the tray on which the disk is kept, read/write laser beams assembly, and the motor to rotate the disk. Access time for optical disks are in the range of 100 to 300 milliseconds.

Advantages of CD-ROM are–

a) Storage capacity is much more in optical disks.
b) Multiple copies of the disk along with its contents can be made inexpensively.
c) They are transportable from one computer to another very easily.

Disadvantages are–

a) They are read-only and contents cannot be changed or updated.

b) The access time in optical disks is more than in case of magnetic disks.

8.14.2 DVD Disks

The success of CD technology and the continuing quest for greater storage capability has led to the development of DVD. **DVD**, also known as **Digital Versatile Disk** or **Digital Video Disk**, is an optical disk storage media format, and was invented and developed by Sony, and Philips in 1995. Its main uses are video and data storage. DVDs are of the same dimensions as compact disks (CDs), but store more than six times as much data.

Variations of the term *DVD* often indicate the way data is stored on the disks: DVD-ROM (read only memory) has data that can only be read and not written; DVD-R and DVD+R (recordable) can record data only once, and then function as a DVD-ROM; DVD-RW (re-writable), DVD+RW, and DVD-RAM (random access memory) can all record and erase data multiple times. The wavelength used by standard DVD lasers is 650 nm thus, the light has a red color.

A DVD disk consists of two substrates (0.6mm thick) bonded together. Each side can contain two layers called 'Layer 0' and 'Layer 1' (the outermost layer). The physical format of a DVD determines the capacity of the DVD disk. DVD capacity is determined by pit size, track pit spacing and the number of layers the disk contains. In the following some list of disks (current) and their capacities are given:

Physical Format	Capacity	Layers	Side(s)
DVD – 5	4.7GB	1	1
DVD – 9	8.54GB	2	1
DVD – 10	9.4 GB	1	2
DVD – 14	13.24GB	2	2
DVD – 18	17.08GB	2	2
DVD – R	4.7GB	1	1
DVD – R	9.4GB	1	2
DVD – RW	4.7GB	1	1
DVD – RW	9.4GB	1	2

8.15 MAGNETIC TAPE

The magnetic tape is mostly used for off-line storage of large amount of data. They are the cheapest and the slowest methods for data storage.

The magnetic tape is a strip of plastic coated with a magnetic film. The tape is very small and generally is of 0.5 or 0.25 inch wide. The data recording on the magnetic tape is same as in the case of magnetic disks. There are a number of parallel tracks on the tape. Seven or nine bits corresponding to one character are recorded simultaneously with a parity bit.

Earlier the tapes had nine tracks each but the newer tape systems use 18 or 36 tracks, corresponding to a word or a double word. A separate read/write head is mounted one in each track so that data can be recorded and read in a sequential manner.

Data are organized in the form of records and these records are separated by gaps referred to as inter-record gaps.

The data on a magnetic tape is recorded and accessed in a sequential manner. Whenever the tape head reach a record gap, the tape motion can be stopped. Each record has an identification pattern both at the beginning and at the end. The starting bit pattern gives the record number and when the tape head reach the bit pattern at the record end, it comes to know that there is a gap after it.

A file is a collection of some related records. The starting of a file is always marked by a file mark as shown in the figure above. The gap after the file mark can be used as a header or identifier for the file. There are gaps after each record to distinguish between them.

In addition to the read and write commands, there are a number of other control commands executed by the tape drive, which includes the following operations:

a) Rewind tape
b) Erase tape
c) Forward space one record
d) Backspace one record
e) Forward space one file
f) Backspace one file

The end of the tape is marked by EOT (End of Tape). The records in a tape may be of fixed or variable length.



CHECK YOUR PROGRESS

Q.13: Fill in the blanks:

a) The width of a magnetic tape is generally ..... or ..... inch.

b) To access the data on a tape there is a ..... in each track.

c) ..... gaps are in between two records to distinguish one from another.

d) Magnetic tape is a ..... access device storage.

e) Tape motion stops when it reaches a .....

f) Data on a tape are organized in the form of .....

g) A tape is written from the ..... to the end.

Q.14: Find True or False:

i) Data storage in optical memory is very costly.

ii) CDs are non-erasable.

iii) WORM is an optical disk product.

iv) Recorded data can be read with the help of a laser beam.



8.16 LET US SUM UP

- The memory hierarchy in the computer system consists of the storage devices employed in the system with the faster but smaller devices at the top and the slow but high capacity and larger memory devices at the bottom.
- Main memory holds the programs and data currently executed by the processor.
- Main memory is of two types: Random Access Memory (RAM) and Read Only Memory (ROM).

- RAM is a volatile memory; that is, the data and programs in RAM are lost when the power is switched off whereas ROM is non-volatile memory; so it doesn't lose its contents when power is turned off.
- RAM may be Static RAM (SRAM) and Dynamic RAM (DRAM). SRAM is simple and has more speed than DRAM. But it is larger and costlier than DRAM.
- Main memory is actually DRAM. SRAM is used in Cache memory.
- ROM is a non-volatile memory that can only be read and not written. ROM is used in certain functions in the computer because of its data permanency and data security as the data cannot be modified easily. There are 5 types of ROM such as ROM, PROM, EPROM, EEPROM and Flash EEPROM.
- The *cache* is a small amount of high-speed memory. Compared to the size of main memory, cache is relatively small. It operates at or near the speed of the processor. Cache memory contains *copies* of sections of the main the memory and is very expensive compared to the main memory.
- Caches exploit the property of locality which states that applications tend to re-use data which they have recently used. This is seen in two forms: *temporal locality* and *spatial locality*.
- Mapping is a technique for determining where main memory blocks can be placed in the cache. There are three different mapping techniques: *direct mapping*, *associative mapping* and *set-associative mapping*.
- In direct mapping, one block from main memory maps into only one possible line of cache memory. As there are more blocks of main memory than there are lines of cache, many blocks in main memory can map to the same line in cache memory.
- Associative cache mapping is the most complex, but it is most flexible with regards to where data-can reside. A newly read block of main memory can be placed anywhere in an associative cache. If the cache is full, a replacement algorithm is used to determine which block in the cache gets replaced by the new data.

- Set-associative mapping is designed to utilize the strengths of the previous two mappings. As with a direct mapped cache, blocks of main memory data will still map into as specific set, but they can now be in any N-cache block frames within each set. Here, The cache is divided into a number of sets containing an equal number of lines. Each block in the main memory maps into one set in cache memory similar to that of direct mapping. Within the set, the cache acts as associative mapping where a block can occupy any line within that set. Replacement algorithms may be used within the set.
- *Virtual Memory* is simply the operating system using some amount of disk space as if it were real memory.
- *Paging* is a method of achieving virtual memory. Virtual space is broken up into equal size *pages* and memory space is broken up into equal size *page frames*.
- The magnetic disk is divided into tracks, sectors and cylinders where the data are recorded.
- There may be a single read/write head for each disk surface or separate heads for each track in the surfaces. Accordingly, heads may be movable or fixed.
- Magnetic disk systems consists of three vital parts- the disk, disk drive and the disk controller.
- RAID combines two or more physical disk units and behave as a single disk unit.
- The three important concepts in RAID are - mirroring, striping and parity.
- Optical memory stores data on an optical medium and the data is read using laser beams and photodetector.



8.17 FURTHER READING

1) Hamacher, V. Carl, Vranesic, Zvonko G., Zaky, Safwat G. *Computer Organization*. McGraw-Hill International Edition.
2) Mano, M. M. *Digital Logic and Computer Design*. PHI.

3) Mano, M. M. *Computer System Architecture*. PHI.
4) Stallings, William. *Computer Organization and Architecture*. PHI.
5) Talukdar, Dr. P. H. *Digital Techniques*. N. L. Publications.



8.18 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: The characteristics of SRAM and DRAM are as follows:

- Simplicity :: SRAM are more simple as no refresh circuitry are needed to keep their data intact.
- Speed :: SRAM is faster than DRAM
- Cost :: SRAM is costlier than DRAM
- Size :: SRAM is larger compared to DRAM.

Ans. to Q. No. 2: a) instructions, main memory; b) sequential; c) Access; d) Cycle; e) less; f) processor; g) independent; h) volatile, non-volatile; i) bits; j) off-line storage

Ans. to Q. No. 3: a) True, b) True, c) False, d) False, e) True, f) False

Ans. to Q. No. 4: a) Dynamic RAM; b) Cache, Main; c) capacitor; d) Static RAM; e) System Clock; f) SDRAM

Ans. to Q. No. 5: i) (b) SRAM, ii) (d) Speed-up memory access; iii) (b) slow and inexpensive

Ans. to Q. No. 6: i) Miss, ii) Spatial locality, iii) external, iv) hit ratio

Ans. to Q. No. 7: i) direct, ii) sets, iii) direct, iv) 4-way set-associative cache

Ans. to Q. No. 8: i) direct; ii) virtual memory; iii) logical; iv) page frames, pages; v) memory space; vi) address space; vii) page, page frame; viii) memory management

Ans. to Q. No. 9: Tracks are the concentric set of rings along which the data is stored in magnetic disks.

The tracks are divided into sections called sectors.

A set of corresponding track on all the surfaces of a disk pack forms a logical cylinder.

Ans. to Q. No. 10: a) more, less; b) interference of magnetic fields; c) read/write head; d) fixed, movable; e) magnetic spots; f) sealed; g) disk, disk drive, disk controller; h) seek time; i) disk access time

Ans. to Q. No. 11: a) True, b) True, c) False, d) True, e) False, f) False

Ans. to Q. No. 12: The three key concepts in RAID are–

i) mirroring– writing identical data to more than one disk

ii) striping– splitting of data across more than one disk.

iii) error-correction– uses parity for error detection and recovery.

Ans. to Q. No. 13: a) 0.5, 0.25; b) read/write head; c) inter-record, d) sequential; e) record gap; f) records; g) beginning

Ans. to Q. No. 14: i) False, ii) True, iii) True, iv) True



8.19 MODEL QUESTIONS

Q.1: What is memory hierarchy? Discuss.

Q.2: Briefly describe Main Memory.

Q.3: Differentiate between Static RAM and Dynamic RAM.

Q.4: Discuss synchronous DRAM.

Q.5: Explain ROM and its different types.

Q.6: What is the function of cache memory? Explain the term *cache hit* and *cache miss*.

Q.7: Discuss physical and virtual memory? What is logical and physical address?

Q.8: What is memory and disk caching? What do mean by *level 1* and *level 2* cache?

Q.9: What do you mean by paging? What is page table and where is it located?

Q.10: Write short notes on any one of the following:

a) Paging

b) Direct Mapping

c) Associative mapping

d) Set-Associative mapping

Q.11: What do you mean by locality of reference? What are its types?

Q.12: What is Winchester disks? Briefly discuss its advantages.

Q.13: Differentiate between fixed-head systems and movable head-systems.

Q.14: Explain optical memory with example.

Q.15: Describe in brief the magnetic tape systems.

Q.16: Differentiate between CD-ROM and DVD.

Q.17: List the types of optical storage.

Q.18: List the major types of magnetic storage.

Q.19: Write short notes on the following:

a) Hard Disk
b) RAID
c) Optical Memory
d) CD-ROM
e) DVD
f) Magnetic Tapes

*** ***** ***

UNIT 9: MEMORY AND I/O ACCESS

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Memory Read Write Operations

9.4 Concept of Handshaking

9.5 Interrupt Driven I/O

9.6 Priority and Daisy Chaining Technique

9.7 Let Us Sum up

9.8 Further Reading

9.9 Answers to Check Your Progress

9.10 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about memory Read-Write operations
- describe the concept of handshaking
- explain polling techniques
- learn about interrupt-driven I/O
- define priority and daisy chaining technique
- learn how memory mapping takes place.

9.2 INTRODUCTION

In the previous unit we have learnt about memory hierarchy. Different semiconductor memories along with their internal structures are discussed in the unit. At the end we have acquainted with the concept of cache and virtual memory.

The speed and capacity of the memory impacts the performance of the computer. The speed of the memory has to keep pace with the speed of the CPU. The speed gap between the CPU and the main memory has been tried to be compensated by the use of the cache memory. The I/O

syatem can be defined as a subsystem of components that moves the data between the external devices and the CPU and main memory. Signals are exchanged between the sender and the receiver in a computer.

In this unit, we will learn about Handshaking, interrupts and polling techniques of data transfer.

9.3 MEMORY READ WRITE OPERATIONS

The memory can be defined as an array of a millions of adjacent cells each of which is capable of storing a binary digit. The basic operations of the memory are the Read and the Write operations. The read operation reads the stored data and the write operation stores new data. The block diagram of the memory is shown in figure 9.1.

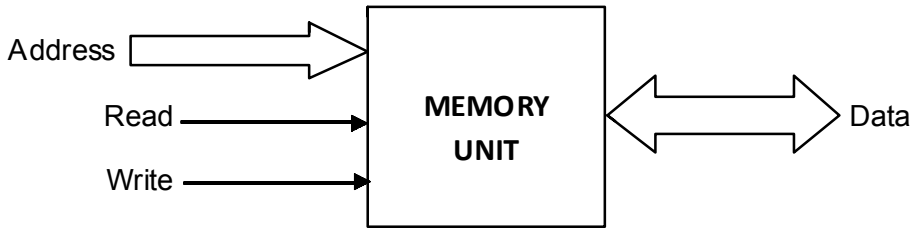


Figure 9.1: Block Diagram of Memory Unit

The read and the write operations require memory address. The address and data of the memory are connected to the address and data buses of the system. There are two metrics that are used to characterize the memory, namely memory access time and memory cycle time. Access time is the time required by the memory to retrieve the data from a memory location. The minimum time between two successive memory operations is called memory cycle time.

The following are the steps that occur in a memory read operation:

1) The address of the location to be read is placed on the address bus.
2) The memory read signal is activated in the control bus.
3) The data is retrieved from the addressed memory locations and placed on the data bus and wait to complete this operation.
4) The data is read from the data bus.
5) The memory read signal is deactivated.

The following are the steps that occur in a memory write operation:

1) The address where the data is o be written is placed on the address bus.
2) The data to be written is placed on the data bus.
3) The memory write signal on the control bus is activated.
4) Wait for the memory to store the data at the addressed location.
5) The write signal is deactivated.

9.4 CONCEPT OF HANDSHAKING

Handshaking is a technique in which the I/O devices communicate with the CPU in transferring data.. It occurs before the transfer of data and just after the establishment of the physical channel between the two entities. Handshaking helps in checking the speed of the transmission. Even when the sender and the receiver are operating at the same speed, the sender may still want to know whether the receiver has indeed received the information. Handshaking provides a mechanism for addressing this issue.

When the speed of the I/O devices do not match with that of the microprocessor and the timing characteristics of the devices are unpredictable, asynchronous data transfer method is used. Such data transfer uses two types of techniques based on the signals. They are Strobe Control and Handshaking methods.

The strobe control method may be activated either by the source or the destination.

Strobe Control Data Transfer Initiated by Source Unit:

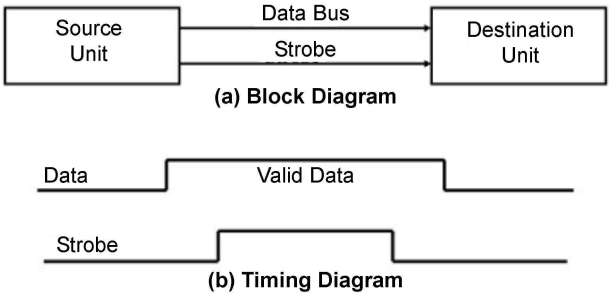


Figure 9.2: Source-Initiated Strobe for Data Transfer

In the above diagram (a), the data bus carries the binary data from the source to destination. The bus has multiple lines to transfer an entire

byte or word. The strobe is a single line that informs the destination when a valid data is available.

In the timing diagram (b), the source first places the data on the data bus. The information on the bus and the strobe signal remains active to allow the destination unit to receive the data.

Strobe Control Data Transfer Initiated by Destination Unit:

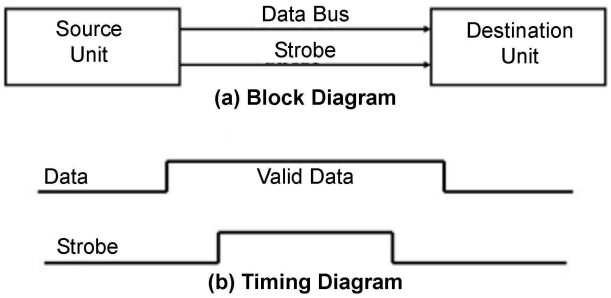


Figure 9.3: Destination-Initiated Strobe for Data Transfer

The destination unit activates the strobe signal to inform the source to provide the data. The source responds by placing the data on the data bus. When the data is accepted, the destination unit disables the strobe and the source unit removes the data from the data bus.

In this data transfer, the disadvantage of the strobe signal is that the source unit that initiated the transfer has no way of knowing whether the destination unit has actually received the data. Similarly, a destination unit that initited the transfer also has no way of knowing whether the source has actually placed the data on the bus. Handshaking method solves this problem by introducing a second control signal that gives a reply to the unit that initiated the transfer.

Handshaking method usually uses two additional hardware lines, one is called "*strobe*" and the other is called "*acknowledge*". The sender provides the signal to the strobe line and the receiver provides the signal to the acknowledge line. In it, one control line is in the same direction as the data flow in the bus from the source to destination. It is used by source unit to inform the destination unit whether there are valid data in the bus. The other control line is in the other direction from destination to the source. It is used by the destination unit to inform the source whether it can accept data. The sequence of control depends on the unit that initiates the transfer.

This means that the sequence of control depends whether transfer is initiated by source and destination.

Handshaking can be used in both parallel data transfer and serial data transfer. It can be understood by citing an example.

Let us suppose that a printer is connected to the computer. The printer can print 100 characters/second. However as the speed of the processor is higher than that of the printer (an I/O device), it can send more information to the printer at the same time. So, when the printer gets enough data to print,

it places a logic 1 signal at its Busy pin, indicating that it is busy in printing. The CPU now tests the busy bit to decide if the printer is busy or not. When the printer will become free it will change the busy bit and the CPU will again send enough amounts of data to be printed. This process of interrogating the printer is called handshaking.

Source Initiated Handshaking Method: The source initiates the transfer by placing the data on the data bus and then enables the data valid signal. The destination activates the data accepted signal after accepting the data. Then the source unit disables the data valid signal.

The block diagram and the sequence of events are shown in the figure below.

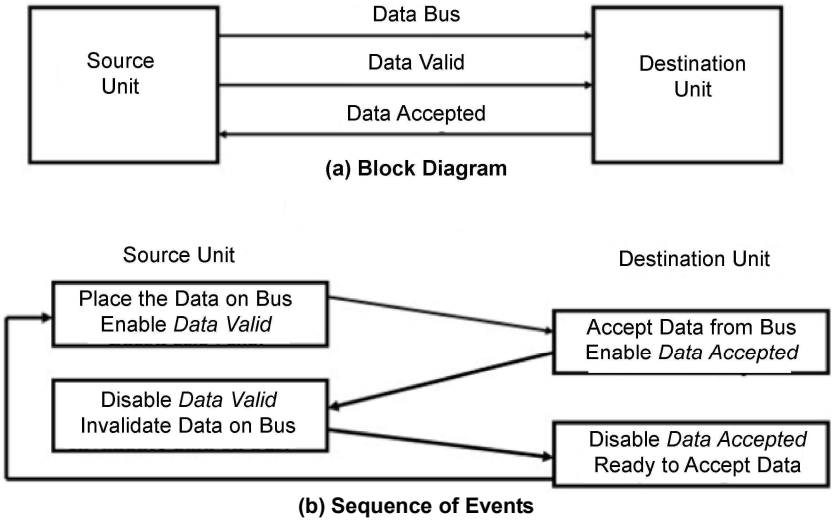


Figure 9.4

Destination Initiated Handshaking Method: When the destination unit is ready to accept data, it activates the Ready for Data signal. The

source places the data on the bus only after it receives the Ready for Data signal from the destination.

The block diagram and the sequence of events are shown in the figures (a) and (b) below.

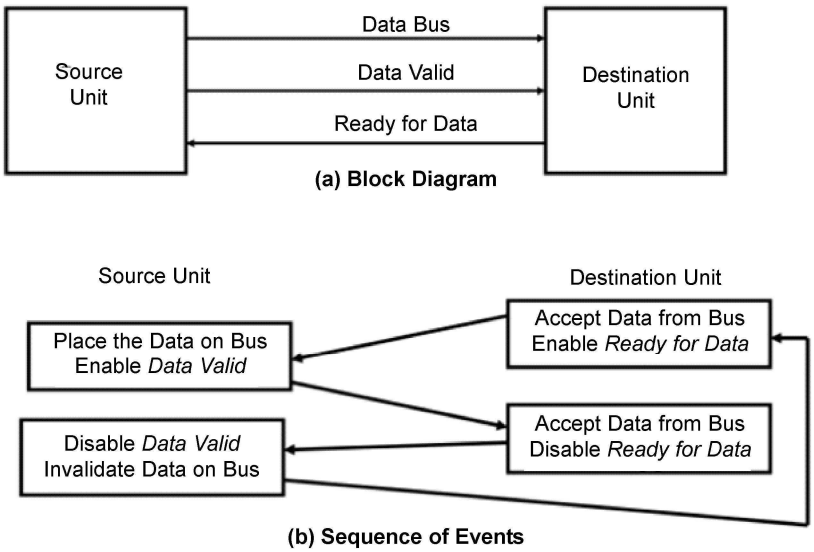


Figure 9.5

The advantage of the handshaking method is that it provides a reliable method of transfer because the successful completion relies on the active participation of both the units. If there is fault then the data transfer will not be a successful one.



CHECK YOUR PROGRESS

Q.1: Fill In the blanks:

i) The speed gap between the CPU and the main memory is compensated by the .....

ii) Signals are exchanged between ..... and the ..... in a computer.

iii) The two metrics that are used to characterize the memory, are ..... and .....

iv) ..... occurs before the transfer of data and just after the establishment of the physical channel between the two entities.

v) Handshaking method uses two additional hardware lines, ... and *acknowledge*.

vi) Handshaking can be used in both ..... data transfer and ..... data transfer.

vii) The process by which the computer waits for an input/ output device to check for its readiness is called .....

9.5 INTERRUPT DRIVEN I/O

The term Interrupt refers to the exceptional event which causes the processor to temporarily suspend the current programs being executed. In the programmed I/O method the CPU stays in the program loop until the I/ O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.

This problem can be overcome by using **interrupt driven I/O**. In both programmed and interrupt I/O, the processor is responsible for extracting data from main memory.

For example, an Ethernet device driver would interrupt whenever it receives an Ethernet packet from the network. The Linux kernel needs to be able to deliver the interrupt from the hardware device to the correct device driver.

An interrupt is generated by the I/O device so as to draw the CPU's attention. The situations in which an interrupt is generated may be as follows:

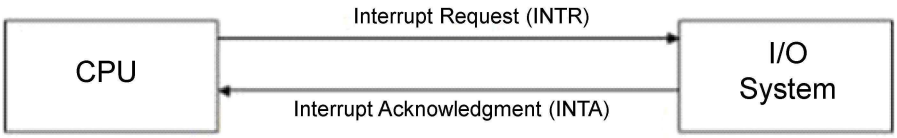
a) The I/O device is ready to transfer data and so request the CPU to initiate a new I/O operation

b) To signal the completion of an I/O operation initiated earlier.

c) To signal unusual condition in the device.

This technique is used to overcome the limitation of programmed I/O. In interrupt driven I/O, instead of making the processor to verify the status of I/O module the I/O module intimate the processor by interrupt signal. An interrupt or exception causes CPU to transfer the control temporarily from its current program to another program i.e., interrupt handler.

Block Diagram for Interrupt Driven I/O:



Transfer control from main program to Interrupt Handle.

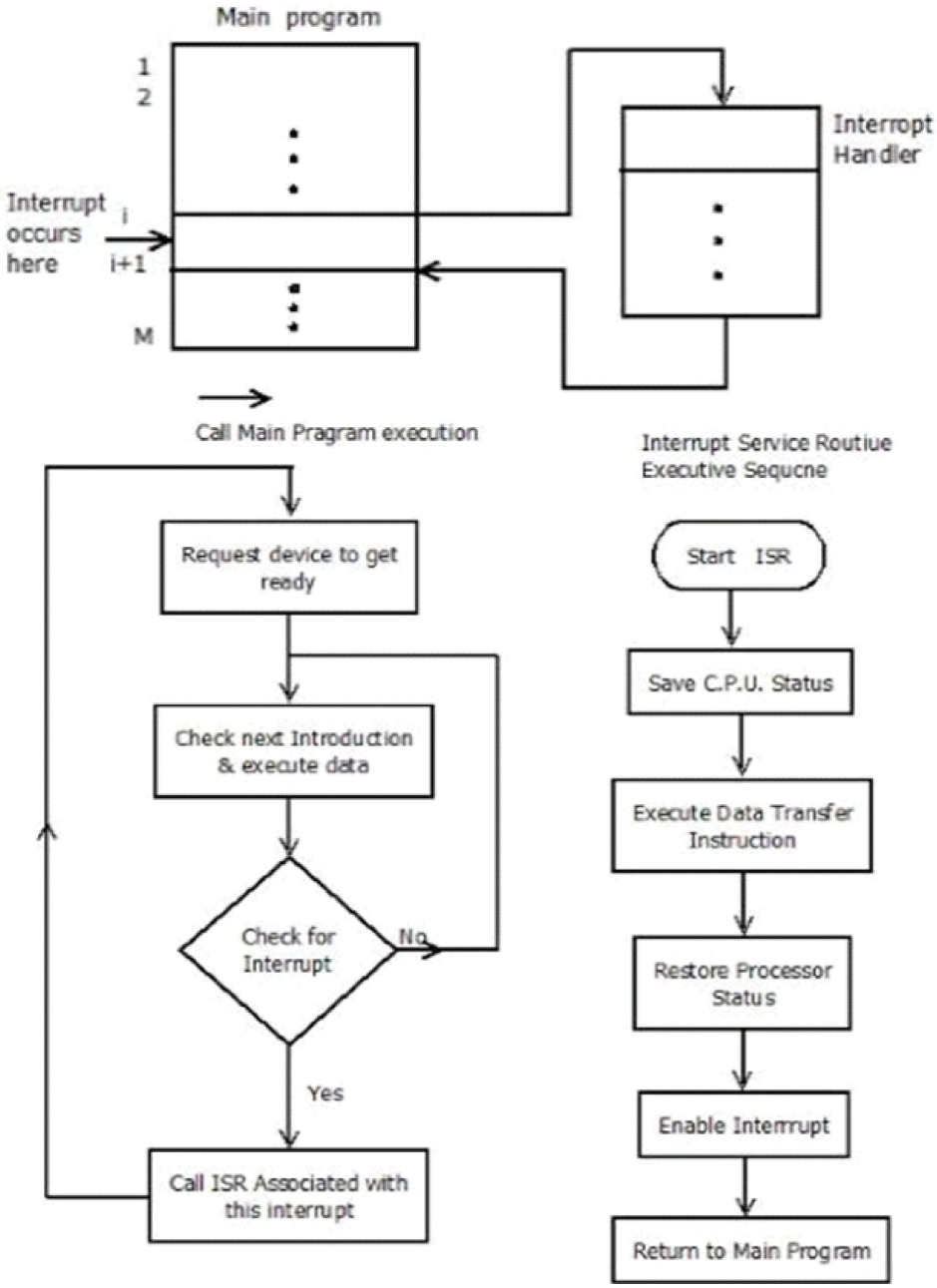


Figure 9.6

By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available

from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

9.6 PRIORITY AND DAISY CHAINING TECHNIQUE

When the I/O devices are ready for data transfer they generate an interrupt signal to the computer. The CPU receives this signal, suspends the current instructions it is executing and then moves forward to service that transfer request. Sometimes multiple devices generate interrupts simultaneously. In such a case, there has to be a way to decide which interrupt to service first. Thus, a priority is to be set among all the devices for systemic interrupt servicing. Devices with high speed transfer such as magnetic disks are given high priority and slow devices such as keyboards are given low priority. When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first. ***Polling*** is a software technique and ***daisy-chaining*** is a hardware technique for determining priority of the devices.

- ***Polling*** is a method by which a process continuously checks the status of an object. The process by which the computer checks the input/ output devices one after another for its readiness is called polling. The computer thus waits until the device is ready. It repeatedly checks for the readiness of the device and if not ready, it does a different task. Although it is not as wasteful as the CPU busy waiting but it is also not efficient as interrupt driven I/O. In other words, polling the device usually means reading its status register every so often until the device's status changes to indicate that it has completed the request. Polling technique is used to determine the priority of the I/O devices i.e., which device will be served by the CPU, when interrupt request comes from more than one device simultaneously.

- The ***daisy-chaining*** method involves connecting all the devices that can request an interrupt in a serial manner. The device with the highest priority is placed at the first position followed by lower priority devices and the device which has lowest priority among all is placed at the last in the chain.

All the devices in the daisy chaining technique are connected in a serial manner. The figure given below depicts the arrangement.

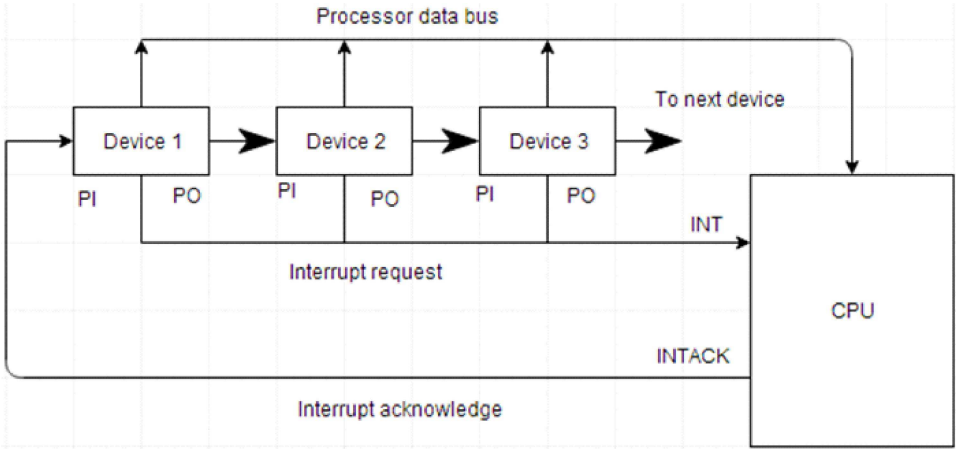


Figure 9.7

In the above figure it can be seen that the interrupt request line is common to all devices. When there are no interrupts the line is in the High state. When a request is there by any device, the interrupt request line goes to the Low state. The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.

The signal is received at the **PI** (Priority in) input of device 1. If the device has not requested the interrupt, it passes this signal to the next device through its **PO** (priority out) output. On the other hand, if the device had requested the interrupt, the device consumes the acknowledge signal and block its further use by placing 0 at its PO. The device then places the interrupt vector address (VAD) into the data bus of the processor. When a device gets a 0 at its PI, it generates a 0 at the PO so as to tell the other devices that the acknowledge signal has been blocked. Thus, the daisy chain arrangement ensures that the highest priority interrupt gets serviced first.



CHECK YOUR PROGRESS

Q.2: Fill In the blanks:

i) ..... refers to the exceptional event which causes the processor to temporarily suspend the current programs being executed.

ii) ..... technique is used to overcome the limitation of programmed I/O.

iii) When the CPU receives an ..... signal, it suspends the current instructions it is executing and then moves forward to service that transfer request.

iv) A ..... is to be set among all the devices for systemic interrupt servicing when multiple devices generate interrupts simultaneously.

v) When two or more devices interrupt the computer simultaneously, the computer services the device with the ..... first.

vi) The devices in ..... technique are connected in a serial manner.



9.7 LET US SUM UP

- The computer memory is an array of a millions of adjacent cells each of which is capable of storing a binary digit.
- The basic operations of the memory are the Read and the Write operations.
- Access time is the time required by the memory to retrieve the data from a memory location
- The minimum time between two successive memory operations is called memory cycle time.
- Handshaking is a technique in which the I/O devices communicate with the microprocessor.

- Handshaking helps in checking the speed of the transmission.
- Handshaking method usually uses two additional hardware lines, one is called "*strobe*" and the other is called "*acknowledge*".
- The sender provides the signal to the strobe line and the receiver provides the signal to the acknowledge line.
- Handshaking can be used in both parallel data transfer and serial data transfer.
- Polling is a method by which a process continuously checks the status of an object
- It repeatedly checks for the readiness of the device and if not ready, it does a different task.
- Interrupt refers to the exceptional event which causes the processor to temporarily suspend the current programs being executed.
- An interrupt is generated by the I/O device so as to draw the CPU's attention.
- When the I/O devices are ready for data transfer they generate an interrupt signal to the computer.
- When multiple devices generates interrupts at the same time, a priority is to be set among all the devices for systemic interrupt servicing
- The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner.
- The device with the highest priority is placed at the first position followed by lower priority devices and the device which has lowest priority among all is placed at the last in the chain.



9.8 FURTHER READING

1) Chaudhuri, P. Pal. *Computer Organization and Design*. PHI.
2) Godse, A. P., Godse, D. A. *Computer Architecture*. Technical Publications Pune.
3) Stallings, William. *Computer Organization and Architecture: Designing for Performance*. PHI.



9.9 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q. No. 1: i) cache memory; ii) Sender, receiver; iii) memory access time, memory cycle time; iv) Handshaking; v) Strobe; vi) Parallel, serial; vii) Polling

Ans to Q. No. 2: i) Interrupt; ii) Interrupt driven I/O; iii) interrupt; iv) priority; v) higher priority; vi) daisy chaining



9.10 MODEL QUESTIONS

Q.1: Explain the handshaking technique.

Q.2: What is polling? Explain the interrupt driven I/O technique along with a block diagram.

Q.3: Why are the priority and Daisy Chaining techniques used? Explain.

*** ***** ***

UNIT 10: MEMORY AND I/O INTERFACING

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.3 I/O Interface

- 10.3.1 Functions of I/O Interfaces
- 10.3.2 Connection of I/O Buses to I/O Devices

10.4 I/O Processor

- 10.4.1 Block Diagram of I/O Processor
- 10.4.2 CPU- I/O Processor Communication

10.5 Priority Encoder

- 10.5.1 Definition of Encoder
- 10.5.2 Priority Encoder and its features
- 10.5.3 Applications of Priority Encoder

10.6 I/O Scheduling

- 10.6.1 Techniques of I/O Operations
- 10.6.2 Types of I/O Scheduling Techniques

10.7 Let Us Sum Up

10.8 Further Reading

10.9 Answers to Check Your Progress

10.10 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about I/O interface and its different functions
- learn how I/O buses are connected to I/O devices
- learn about I/O Processor and its functinality with CPU
- learn about Priority encoder and its various features and applications
- learn about different types of I/O operations
- learn about I/O scheduling techniques and its different types.

10.2 INTRODUCTION

We know that input-output subsystem of a computer system, referred to as I/O, provides an efficient mode of communication between the central system of computer and the outside environment. Programs and data must be entered first into computer memory for processing and results obtained from computations must be recorded or displayed for outer world. Keyboard is the most common input device for entering information into a computer system that allows a user to enter alphanumeric information directly. On the other hand the CPU is an extremely fast device capable of performing operations at very high speed. I/O devices attached to the computer system are also known as peripherals. There are three types of peripherals such as input, output and input-output peripherals. These I/O devices may be analog or digital, serial or parallel. These peripherals are electromagnetic and electromechanical devices of some complexity. Video monitors are the most common used peripherals. Printers provide a permanent record on paper of computer output data or text. Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.

Some important characteristics of I/O devices are given below:

- They are shared by multiple programs.
- They are interrupt driven and low-level control is complex.

10.3 I/O INTERFACE

Input-output interface provides a method for transferring information between internal storage and external I/O devices. I/O devices connected to a computer system need special communication links for interfacing them with the CPU. The purpose of the communication link is to resolve the differences that exist between the CPU and each I/O device. The major issues are given below:

a) Peripherals are electromagnetic and electromechanical devices and their manner of operation is different from the operation of the CPU and memory which are electronic devices. Therefore, a conversion of signal values may be required.

b) The data transfer rate of I/O devices is usually slower than the transfer rate of the CPU and as a result, a synchronization mechanism may be needed.

c) Data codes and format in peripherals differ from the word format in the CPU and memory.

d) The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other I/O devices connected to the CPU.

To resolve these differences, the computer system includes special hardware components between the CPU and I/O devices to supervise and synchronize all I/O transfers. These components are called interface units because they interface between the (CPU) processor bus and the I/O devices. In digital computer system the interface is referred as a complementary set of signal connection points between two parts of a system. Basically, there are two types of interfaces which are (i) CPU interface that correspond to the system bus and (ii) I/O interface that depends on the nature of I/O device.

10.3.1 Functions of I/O Interfaces

The main functions of I/O interfaces are given below:

a) Data conversion: Data conversion refers to conversion between digital and analog signals.

b) Synchronization: It refers to matching operation speed of CPU and other I/O devices.

c) Device selection: It refers to the selection I/O device by CPU in a queue manner. In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.

10.3.2 Connection of I/O Bus to I/O Devices

I/O bus consists of data lines, address lines and control lines. Each I/O device has associated with it an interface unit. Each interface decodes the address and control received from the I/O bus and

interprets them for the I/O devices as well as provides signals for the I/O controller. It also synchronizes the data flow and supervises the transfer between I/O devices and CPU. Each I/O device has its own controller that operates the particular electromechanical device.

The CPU is attached to all peripheral interfaces through the I/O bus. To communicate with a particular device, the CPU places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address then it activates the path between the bus lines and the device that it controls. All I/O devices whose address does not correspond to the address in the bus are disabled by their interface.

At the same time that the address is made available in the address lines, the processor provides a function code in the control lines. The interface selects respond to the function code and proceeds to execute it. The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and its attached I/O units. The interpretation of the command depends on the I/O device that the CPU is addressing. There are four type of commands that an interface may receive, which are– (i) control command, (ii) status command, (iii) data output command and (vi) data input command.

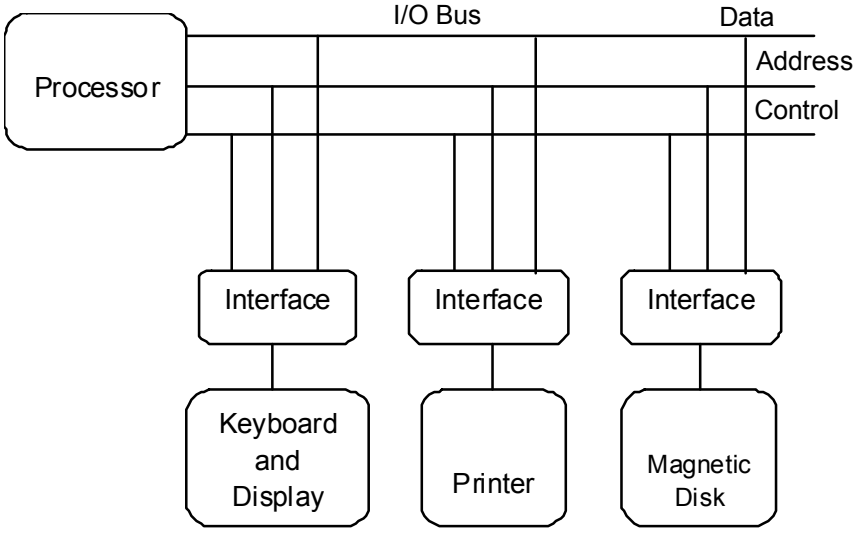


Figure 10.1: Connection of I/O Bus to I/O Devices through Interfaces

10.4 I/O PROCESSOR

An input-output processor (IOP) is a processor with direct memory access capability. In this case, the computer system is divided into a memory unit and number of processors. Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

10.4.1 Block Diagram of I/O Processor

Here the block diagram of a computer along with various I/O Processors is given below. The memory unit occupies the central position and can communicate with each processor. The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program.

The IOP operates independent from CPU and transfer data between peripherals and memory.

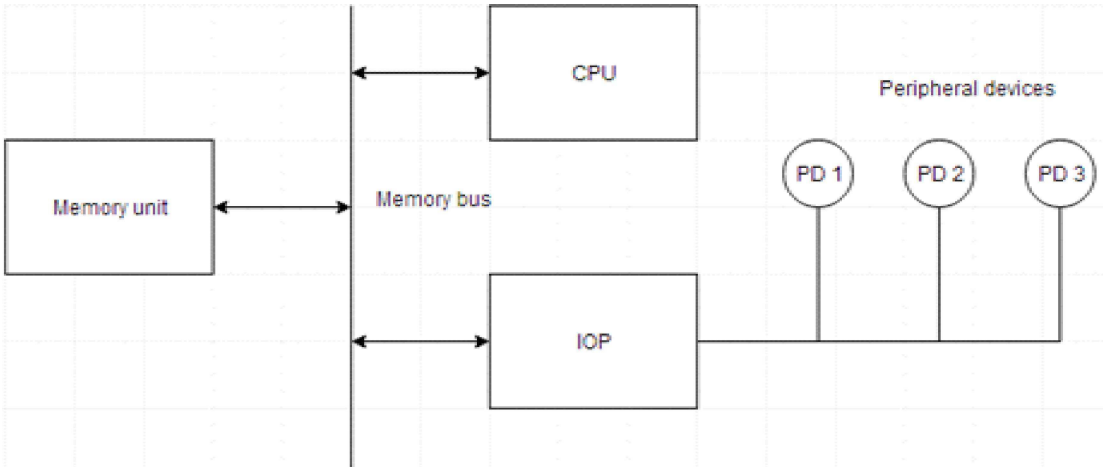


Figure 10.2: Block Diagram of I/O Processor

The communication between the IOP and the devices is similar to the program control method of transfer. And the communication with the memory is similar to the direct memory access method.

In large scale computers, each processor is independent of other processors and any processor can initiate the operation. The

CPU can act as master and the IOP act as slave processor. The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU. CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt. Instructions that are read from memory by an IOP are also called *commands* to distinguish them from instructions that are read by CPU. Commands are prepared by programmers and are stored in memory. Command words make the program for IOP. CPU informs the IOP where to find the commands in memory as well as when it is time to execute the I/O program.

10.4.2 CPU–I/O Processor Communication

The communication between CPU and I/O processor may take various forms, depending on the particular computer considered. In most cases the memory unit acts as a message center where each processor leaves information for the other. First CPU sends an instruction to test the I/O processor path. The I/O processor responds by inserting a status word in memory for the CPU to check. The bits of the status word indicate the condition of the I/O processor and I/O devices. If all is in order, then CPU sends the instruction to start I/O transfer. CPU can also continue with another program while the I/O processor is busy with I/O program. When I/O processor terminates the execution of its program, it sends an interrupt request to the CPU. The CPU responds to the interrupt by issuing an instruction to read the status from I/O processor. Then the I/O processor responds by placing the contents of its status report into specified memory location.

The I/O processor takes care of all data transfer between different I/O units and the memory while CPU is processing another program. The I/O processor and CPU are competing for the use of memory so that the number of devices can be in operation is limited by the access time of the memory.

10.5 PRIORITY ENCODER

The Priority encoder is a circuit that implements the priority function. The logic of the priority encoder is such that if two or more inputs arrive at the same time, the input having high priority will take precedence.

10.5.1 Definition of Encoder

The general definition of an encoder is “An encoder is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another, for the purposes of standardization, speed or compression.” Now, talking about a digital encoder, the Digital Encoder is a combinational circuit that generates a specific code at its outputs such as binary or BCD in response to one or more active inputs.

Digital encoders produce outputs depending on the number of data input lines. An “n-bit” binary encoder has 2^n input lines and n-bit output lines with common types that include 4-to-2, 8-to-3 and 16-to-4 encoders.

10.5.2 Priority Encoder and Its Features

The Priority Encoder allocates a priority level to each input. The priority encoders output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored. Priority encoder is a circuit that converts multiple binary inputs into binary representation of the index of active input bit with the highest priority. Each of input has assigned priority. The least significant bit has the highest priority and the most significant bit the lowest. If more than one input is active at the same time the input having highest priority will take precedence. The priority encoder component has an additional output indicating that output signal is valid when at least one input is active. Priority encoders are often used when multiple components are to share common resource or in interrupt controllers.

Using a parallel prefix computation (PPC) speeds up calculation and improves the circuit timing.

We may also define in another terms as the Priority encoder is a circuit that converts multiple binary inputs into a binary representation of the index of active input bit with the highest priority. Each of input has assigned priority where the least significant bit has the highest priority and the most significant bit the lowest. If more than one input is active at the same time the input having highest priority will take precedence. Additional output signal indicates that output grant is valid.

A priority encoder provide n bits of binary coded output representing the position of the highest order active input of 2^n inputs. If two or more inputs are high at the same time, the input having the highest priority will take precedence.

Some important features of Priority Encoder is given below:

- Parameterized priority encoding capable of both small and large (up to 8k) input width.
- Valid output indicating the output is valid.
- Large encoders built as a structure of smaller encoders.
- Simple or fast parallel implementation using parallel prefix computation.
- Optional pipeline registers on the output of each stage of hierarchy.
- Small building block width selected independently from total input width controlling the number of hierarchy stages and latency of pipeline versions.
- Clock enable inputs for pipeline stages.

10.5.3 Priority Encoder and Its Applications

Some important applications of priority encoder are given below:

- Priority encoders can be used to reduce the number of wires needed in a particular circuits or application that have multiple inputs. For example, we may assume that a microcomputer needs

to read the 104 keys of a standard QWERTY keyboard where only one key would be pressed either “HIGH” or “LOW” at any one time.

- Another more common application of priority encoder is in magnetic positional control as used on ships navigation or for robotic arm positioning etc. For example, the angular or rotary position of a compass is converted into a digital code by a 74LS148 8-to-3 line priority encoder and input to the systems computer to provide navigational data. Magnets and reed switches could be used at each compass point to indicate the needles angular position.
- Other applications especially for Priority Encoders may include detecting interrupts in microprocessor applications. In this case the microprocessor uses interrupts to allow peripheral devices such as the disk drive, scanner, mouse, or printer etc, to communicate with it, but the microprocessor can only “talk” to one peripheral device at a time so needs some way of knowing when a particular peripheral device wants to communicate with it.

The processor does this by using “Interrupt Requests” or “IRQ” signals to assign priority to all the peripheral devices to ensure that the most important peripheral device is serviced first. The order of importance of the devices will depend upon their connection to the priority encoder.

10.6 I/O SCHEDULING

Input/output (I/O) scheduling is the method that computer’s operating systems use to decide in which order the block I/O operations will be submitted to memory or storage volumes. I/O scheduling is sometimes called disk scheduling. I/O scheduling usually has to work with hard disk drives that have long access times for requests placed far away from the current position of the disk head.

I/O schedulers can have many purposes depending on the goals. Some important common purposes are given below:

- To minimize time wasted by hard disk seeks.
- To guarantee that certain requests will be issued before a particular deadline occurs.
- To prioritize a certain processes' I/O requests.
- To give a share of the disk bandwidth to each running process.
- To use hardware efficiently.

Suppose the I/O request issues a system call to the OS. ‰If desired disk drive or controller is available, request is served immediately. ‰If busy, new request for service will be placed in the queue of pending requests. When one request is completed, the OS has to choose which pending request to service next.

10.6.1 Techniques of I/O Operations

There are three techniques for performing I/O operation which are:

i) Programmed I/O,
ii) Interrupt-driven I/O and
iii) Direct Memory Access (DMA)

i) Programmed I/O: In this case, the processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding.

ii) Interrupt-driven I/O: In this case of I/O operations, the processor issues an I/O command on behalf of a process in two cases: *non-blocking* and *blocking*.

In the case of non-blocking, the processor continues to execute instructions from the process that issued the I/O command. But for blocking case, the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process.

iii) Direct Memory Access (DMA): In this case of I/O operation, a DMA module controls the exchange of data between main memory and an I/O module without hampering the working load of a

processor. There are mainly three steps in DMA transfers, which are given below:

- Processor sets up the DMA transfer by supplying identity of device, operation to perform, starting memory address that is source or destination of data, number of bytes to be transferred.
- DMA controller starts the operation (arbitrates for the bus, supplies address, reads or writes data), until the entire block is transferred.
- DMA controller interrupts the processor, which then takes the necessary actions.

10.6.2 Types of I/O Scheduling Techniques

Although there are lots of other algorithms that reduce the seek time of all requests, here we will only concentrate on the following I/O scheduling algorithms.

i) First Come-First Serve (FCFS)
ii) Shortest Seek Time First (SSTF)
iii) Elevator (SCAN)
iv) Circular SCAN (C-SCAN)

i) First Come-First Serve (FCFS): All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next.

ii) Shortest Seek Time First (SSTF): In this case, request is serviced according to next shortest distance. The process would continue until all the process is taken care of. There is a great chance that starvation would take place. The reason for this is if there were a lot of requests close to each other the other requests will never be handled since the distance will always be greater.

iii) Elevator (SCAN): This approach works like an elevator does. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up.

iv) Circular SCAN (C-SCAN): In this Circular scanning technique it works just like the elevator to some extent. It begins its scan toward the nearest end and works it way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction. Keep in mind that the huge jump doesn't count as a head movement.



CHECK YOUR PROGRESS

Q.1: State whether the following statements are true (T) or false (F):

a) I/O devices attached to the computer system are also known as peripherals.

b) I/O devices are not electromagnetic and electromechanical devices of some complexity.

c) The data transfer rate of I/O devices is usually slower than the transfer rate of the CPU.

d) Device selection: It refers to the selection I/O device by CPU in a stack manner.

e) Data codes and format in peripherals differ from the word format in the CPU and memory.

f) The computer system includes special hardware components between the CPU and I/O devices to supervise and synchronize all I/O transfers known as interface.

g) I/O bus consists of only data lines and address lines.

h) The Priority encoder is a circuit that implements the priority function.

i) Input/output (I/O) scheduling is the method that computer's operating systems use to decide in which order the block I/O operations.

j) A DMA module controls the exchange of data between main memory and an I/O module hampering the working load of a processor.



10.7 LET US SUM UP

- We know that input-output subsystem of a computer system, referred to as I/O, provides an efficient mode of communication between the central system of computer and the outside environment.
- I/O devices attached to the computer system are also known as peripherals. There are three types of peripherals such as input, output and input-output peripherals.
- Input-output interface provides a method for transferring information between internal storage and external I/O devices. I/O devices connected to a computer system need special communication links for interfacing them with the CPU.
- Basically, there are two types of interfaces which are (i) CPU interface that correspond to the system bus and (ii) I/O interface that depends on the nature of I/O device.
- An input-output processor (IOP) is a processor with direct memory access capability. Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions.
- The I/O processor (IOP) takes care of all data transfer between different I/O units and the memory while CPU is processing another program.
- The Priority encoder is a circuit that implements the priority function. The logic of the priority encoder is such that if two or more inputs arrive at the same time, the input having high priority will take precedence.

- We may also define in another terms as the Priority encoder is a circuit that converts multiple binary inputs into a binary representation of the index of active input bit with the highest priority.
- A priority encoder provide n bits of binary coded output representing the position of the highest order active input of 2^n inputs.
- Input/output (I/O) scheduling is the method that computer's operating systems use to decide in which order the block I/O operations will be submitted to memory or storage volumes. I/O scheduling is sometimes called disk scheduling.
- There are three techniques for performing I/O operation which are: (i) Programmed I/O, (ii) Interrupt-driven I/O and (iii) Direct Memory Access (DMA).
- In the programmed I/O, the processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding.
- In interrupt-driven I/O, for the I/O operations, the processor issues an I/O command on behalf of a process in two cases: (i) non-blocking and (ii) blocking.
- DMA module controls the exchange of data between main memory and an I/O module without hampering the working load of a processor.
- Although there are lots of other algorithms that reduce the seek time of all requests, Some I/O scheduling algorithms are:
  a) First Come-First Serve (FCFS)
  b) Shortest Seek Time First (SSTF)
  c) Elevator (SCAN)
  d) Circular SCAN (C-SCAN)



10.8 FURTHER READING

1) Chaudhuri, P. Pal. (2nd Edition, 2003). *Compter Organization and Design*. PHI.
2) Hammacher, Carl. (Fifth Edition, 2002). *Computer Organization*. McGraw Hill International Edition.

3) Rajaraman, V. (Fouth Edition, 2008). *Fundamentals of Computers*.

4) Stallings, William. (2004). *Computer Organization and Architecture: Designing for Performance*. PHI.



10.9 ANSWERS CHECK YOUR PROGRESS

Ans. to Q. No. 1: a) True, b) False, c) True, d) False, e) True, f) True, g) False, h) True, i) True, j) False



10.10 MODEL QUESTIONS

Q.1: What do you mean by I/O devices? Explain different I/O devices.

Q.2: What is I/O interface? Describe different issues and functions of I/O interfaces in the computer system.

Q.3: Explain with block diagram how does the connection of I/O bus to I/O devices work.

Q.4: What is I/O processor? Explain its working principle.

Q.5: What do you mean by priority encoder? Briefly discuss the Significance of priority encoder in digital computer system.

Q.6: Explain different techniques of I/O operations.

Q.7: What is I/O scheduling technique? Explain its different types.

*** ***** ***

UNIT 11: EXTERNAL MEMORY

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Magnetic Disk

11.4 Magnetic Read-Write Operation

11.5 Track

11.6 Sector

11.7 Inter Track and Inter Sector Gap

11.8 Cylinder

11.9 Fixed Head Disk

11.10 Seek Time

11.11 Access Time

11.12 Transfer Time

11.13 Rotational Delay

11.14 RAID

11.15 Optical Memory

11.16 Magnetic Tap

11.17 Let Us Sum Up

11.18 Further Reading

11.19 Answers to Check Your Progress

11.20 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about different external memory
- explain read-write operation
- describe different parts associated with external memory like track, sector, cylinder, etc.
- learn about seek time, access time and rotational delay
- learn about RAID, optical memory etc.

11.2 INTRODUCTION

In the previous unit we have learnt about the I/O processor and the interfacing of the I/O processors with the different devices. Also we have gone through the input and output devices.

In this unit, we will describe secondary storage devices, magnetic disk, optical memory, magnetic tape and various terms associated with these memories.

The speed and capacity of the memory impacts the performance of the computer. Magnetic disk is a random access medium. That is the data on the disk can be found almost immediately rather than sequentially going through the data. The data on the disk can be erased and reused virtually indefinitely. It is designed to reside in a protective case or cartridge to protect it from dust and other external interference.

The magnetic tape is another important storage device that is a sequential one. Also, the optical storage technology is important in all types of computer systems. The CD and the DVDs have also become increasingly important.

11.3 MAGNETIC DISK

Magnetic disks are the widely used as secondary storage device. They are mainly used for bulk storage. Unlike the magnetic tapes, the data on the magnetic disks can be read randomly. A magnetic disk is a thin, circular metal plate the surface is coated with a magnetic oxide layer usually on both the sides. These have a high storage capacity, reliable and have capability to access the stored data directly. The disks are mounted on a rotating shaft. Each magnetic disk has a head which consists of a magnetizing coil and a magnetic yoke. Information is stored on the disks which are represented as magnetic spots. Presence of a magnetic spot represents a 1 bit and the absence of a magnetic spot represents a 0 bit.

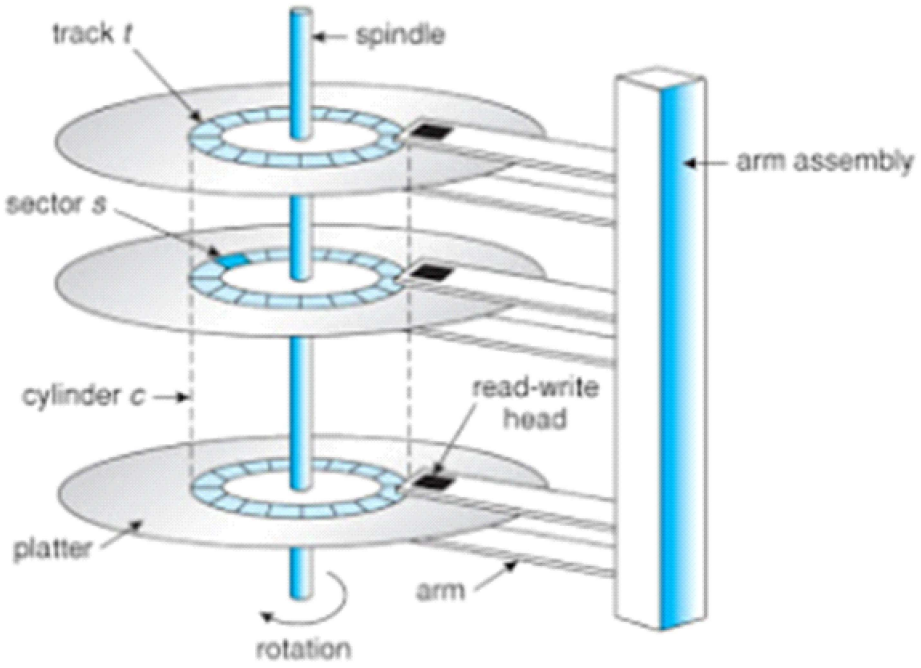


Figure 11.1

The individual disks are spaced on the shaft to allow room for a read/write mechanism to move between them.

Information is stored on the disk by magnetizing the surface in a particular direction as shown in the figure below.

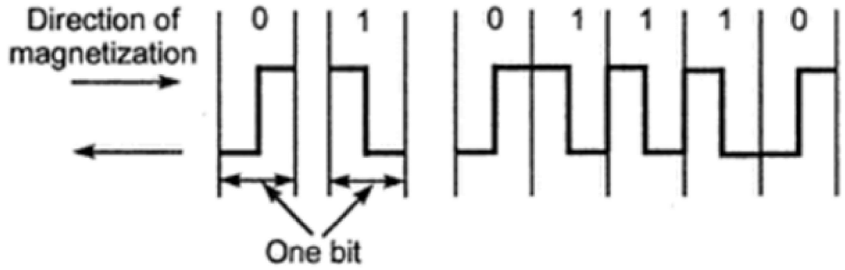


Figure 11.2: Bit Representation of Magnetic Disk

In most disk storage devices, several disks are assembled to form a disk pack as shown below:

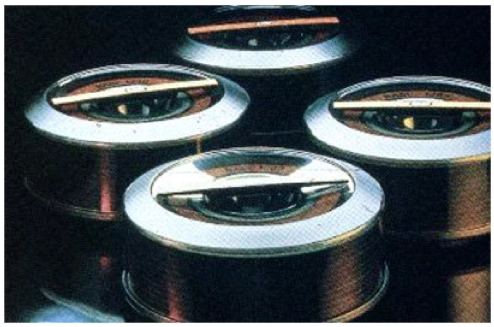


Figure 11.3

The top and bottom surfaces are not used for storing data because they are likely to become scratched or nicked. A disk pack may contain anywhere from five to one hundred disks.

The advantage of the magnetic disk over the magnetic tapes is that the access time is much faster.

11.4 MAGNETIC READ WRITE OPERATIONS

The data on a disk is read or written by the read/write heads located between the disks. Most disk units have one read/write head for each disk recording surface. All the heads are permanently connected to an access mechanism.

In some systems, there are two heads– a read head and a write head. During a Read/Write operation, the head is stationary and the platter rotates beneath it.

A read/write mechanism is used to achieve the access to read or write on the magnetic disk. When the device is used to read from a particular magnetic medium, it detects the individual magnetized areas and translates them into electrical pulses. In performing the write operations, the device magnetizes the appropriate areas on the magnetic medium thereby erasing the previously stored data, if any. Also, the electrical impulses from the

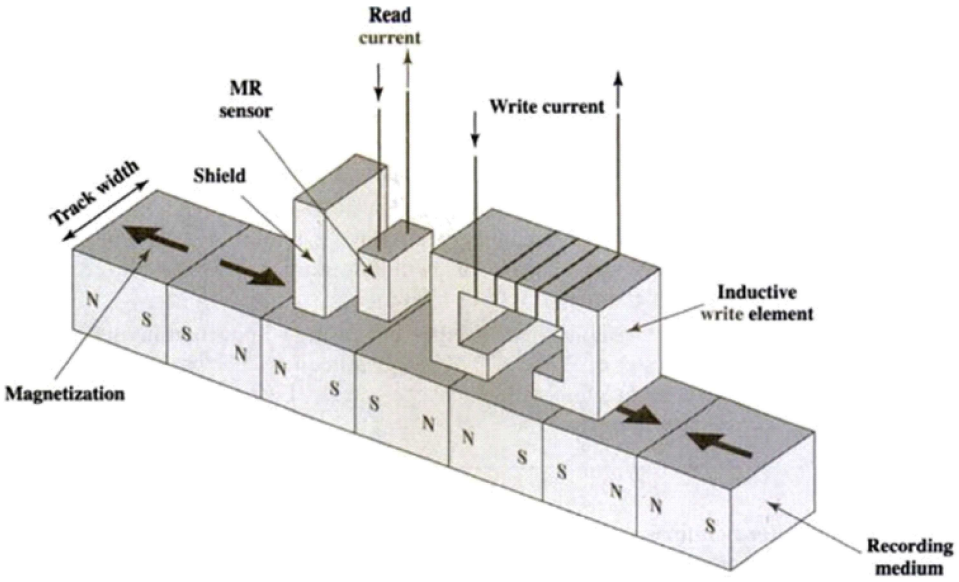


Figure 11.4

processor are converted into magnetic spots on the disks for writing to it. When electric pulses are passed through the write head, a magnetic field is produced. These magnetic patterns are recorded on the surface of the disk below creating different for the positive and the negative currents. If the direction of the current is reversed, the direction of the magnetization of the recording medium is also reversed.

The figure above shows the inductive Read/Write head. During a read operation, a magnetic field moving relative to the coil produces an electrical current in the coil. When the surface of the disk passes under the head, it produces the same current that was recorded.

The head consists of a partially shielded magnetoresistive (MR) sensor. By passing a current through the MR sensor, the resistive changes are detected as voltage signals.



Figure 11.5

11.5 TRACK

The data on the magnetic disk is organized in a concentric set of circles called track. There can be from a few hundred to a few thousand tracks on a single disk surface. Each of the tracks has the same width as the head and the adjacent tracks are separated by gaps. Data is stored serially on each track. The tracks are numbered from the outermost to the innermost circle starting with zero.

Each track on a disk can store the same amount of data even though the tracks get smaller toward the center of the disk. This results higher bit density in inner tracks than that of the outer tracks. Consider a disk pack with 4,000 usable tracks (20 surfaces x 200 tracks per surface) on which 7,294 characters can be stored on each track. The disk pack could conceivably store 29,176,000 characters of data (4,000 tracks x 7,294 characters per track).

11.6 SECTOR

Each track is divided into sectors. The sectors refers to the number of fixed size areas (imaginary pie slices) that can be accessed by one of the read/write heads, in one rotation of the disk, without the head having to change its position. The number of bytes stored in each sector is same. Thus, the size of the inner sectors decides the storage capacity for all other sectors on the disk. Sectors are typically 512 bytes in size, although there is an initiative to increase the sector size to 4096 bytes.

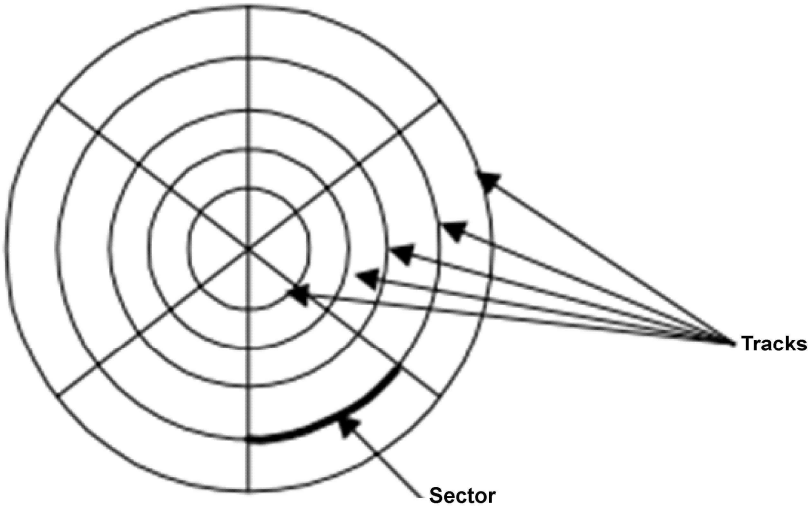


Figure 11.6

11.7 INTER TRACK AND INTER SECTOR GAP

The gap in between the tracks in a magnetic disk is called the Inter-track gap and the gap between the sectors in the disks is called inter-sector gap. The inter-track gaps help protects data integrity. To avoid, imposition

unreasonable precision requirements on the system, adjacent sectors are separated by inter-track (inter-sector) gaps.

Some means are needed to locate sector positions within a track. Clearly there must be some starting points on the track and a way of identifying the start and end of each sector.These requirements are handled by means of a control data recorded on the disk. Thus, the disk is formatted with some extra data used only by the disk drive and not accessible to the user.

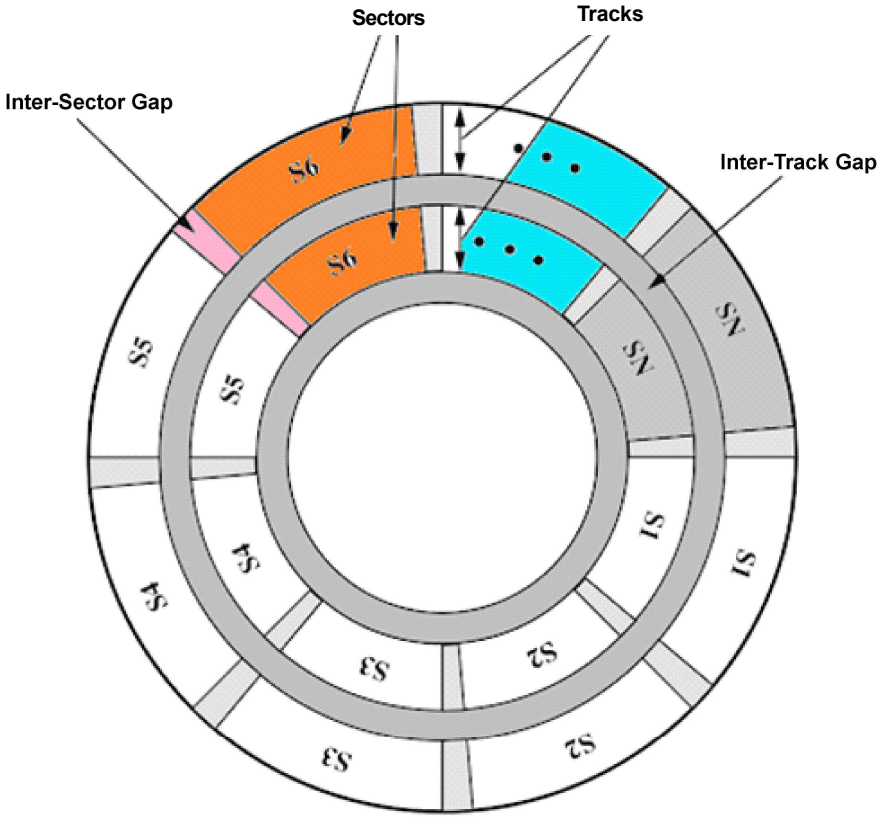


Figure 11.7

11.8 CYLINDER

To have a large storage system in the computer, multiple magnetic disks are maintained and used together. The disks or the platters are stacked on top of each other with a common spindle. The spindle turns or rotates the disks of the pack.

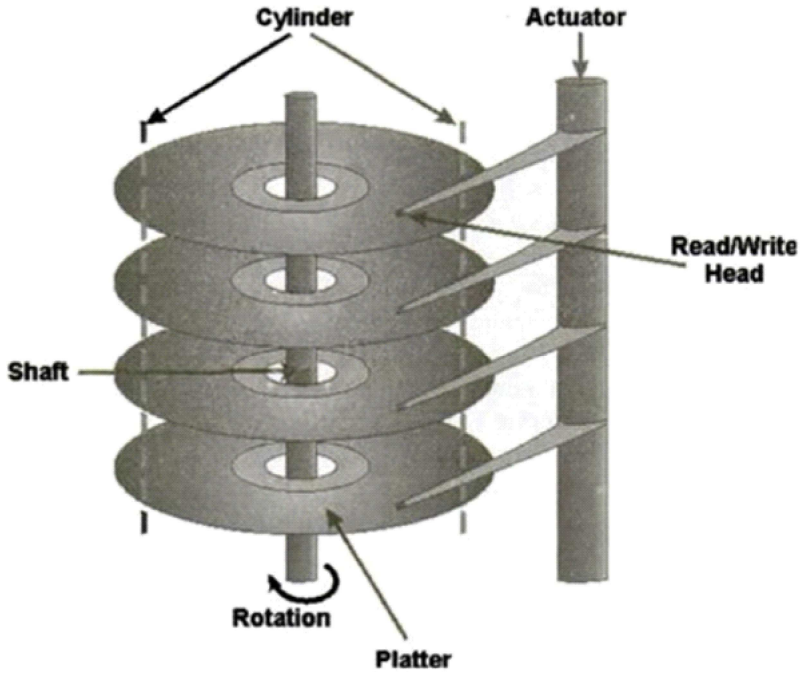


Figure 11.8

There is a gap between the platters so that there is space for the magnetic read/write head. For each side of each platter, there is a magnetic read/write head. The head is mounted on arms which can move the heads towards the central spindle or towards the edge. This concept of stacking the disks is called cylinder. A cylinder is made up of all the tracks of the same number from all the metal disks. Thus cylinder is an imaginary view of all the matching tracks on top of each other as shown in the figure above. Thus the storage capacity of a magnetic disk can be defined as the following:

Storage capacity of a magnetic disk = Number of recording surfaces
x Number of tracks per surface
x Number of sectors per track
x Number of bytes per sector

11.9 FIXED HEAD DISK

There are two types of disks depending upon the two methods of accessing data. One is moving head and the other is fixed head.

A fixed head disk has many read/write heads on each disk surface as the number of tracks which are mounted on the access arm as shown in the figure below.

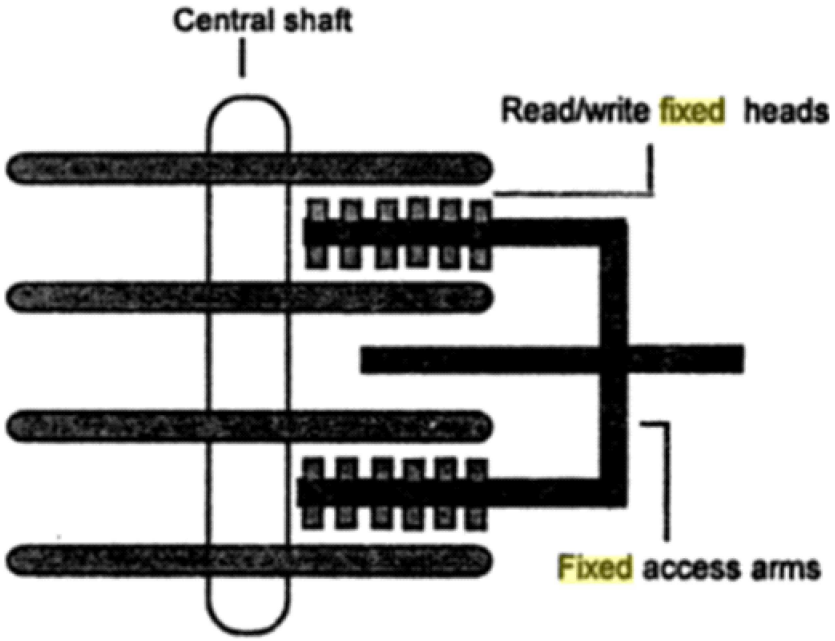


Figure 11.9

The head does not move as each track has a separate read/write head. The moving head disk systems on the other hand have one read/ write head for each disk surface which is mounted on the access arm. The head in such systems moves horizontally on the disk surface to read or write data.

The access speed of the fixed head disks are more as compared to the moving head disks. In a fixed head disk, the track or the cylinder is selected by electronically switching to the appropriate read/write head rather than by mechanically moving it. This makes it faster.

However, as the cost of the additional read/write heads is quite high, so the fixed head disks are not commonly used.



CHECK YOUR PROGRESS

Q.1: Fill In the blanks:

i) Magnetic disk has a head which consists of a ..... and a .....

ii) The data on a disk is read or written by the ..... heads located between the disks.

iii) During a read operation, a magnetic field moving relative to the coil produces an ..... in the coil.

iv) The head consists of a partially shielded ..... sensor.

v) The data on the magnetic disk is organized in a concentric set of circles called .....

vi) Each track on a disk can store the same amount of data even though the tracks get ..... toward the center of the disk.

vii) Tracks are divided into .....

viii) The gap in between the tracks in a magnetic disk is called the ..... and the gap between the sectors in the disks is called .....

11.10 SEEK TIME

The time required to position the read/write head from the current track or cylinder to the desired track or cylinder is called seek time. The seek time varies depending upon the position of the head when a data request is made. It is zero when the head is already positioned on the specified cylinder.

The seek time is maximum when the read/write has to move from the outermost track to the innermost track or vice-versa.

11.11 ACCESS TIME

Access time is the time taken to read a record from the magnetic disk. In other words, it is the time elapsed between the initiation of a particular request for data and the receipt of the data.

The sum of the seek time and the rotational latency is the access time. The average disk access time is equal to seek time + average rotational delay+ transfer time + controller overhead.

11.12 TRANSFER TIME

Once the read/write head is in position, the read or write operation can be performed as the sector moves under the head. This is the data

transfer operation of the disk. The transfer time is the time taken to transfer a block of bits. It is a function of the sector size, rotation speed, and the recording density of a track.

The transfer time to or from the disk depends on the rotation speed of the disk. The transfer time, T can be calculated as

$$T = \frac{b}{rN}$$

Where, T = transfer time

b = number of bytes to be transferred

N = number of bytes on a track

r = rotation speed

Thus, the Total average access time can be expressed as

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

here, T_S is the average seek time.

11.11 ROTATIONAL DELAY

Seek time as we have seen is the time taken to position the head at the track. After the track is selected the disk controller waits until the appropriate sector rotates to line up with the head. This time which is taken for the beginning of the sector to reach the head is called rotational delay or rotational latency.

11.12 RAID

RAID is ***Redundant Array of Independent Disks***. Originally it was known as Redundant Array of Inexpensive Disks. It is a way of storing data in different places on multiple disks to protect the data in case of a drive failure. It is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both. The data recording is distributed on the different drives which appear as a single drive to the application program.

The access speed of the magnetic disks is considerably slow as compared to the speed of the processor and the main memory. By improving

the disk performance the overall performance of the computer can be increased. This led to the development of arrays of disks that operate independently and in parallel. The same data is kept in multiple disks. With the multiple disks, separate I/O requests can be handled in parallel provided the data that is required is on the separate disks. Also, a single I/O request can be handled in parallel as long as the data to be accessed is in the multiple disks.

In a RAID, a single large file is stored in several different disks by breaking up the file into a number of smaller pieces and storing the pieces on the separate disks. This is called ***data stripping.*** Thus the file modules can be accessed simultaneously. This increases the access speed and the data transfer speed. Thus the transfer time would be the time that would be required in a single disk divided by the number of disks used in the array. A data stripe can be any size between 512 bytes up to several megabytes.

There are a number of ways in which the data can be organized with the use of multiple disks. These are called as the RAID levels. These levels do not imply a hierarchical relationship. They are different design architectures. The RAID levels use these stripes and the multiple disks in different ways.

RAID 0: The RAID level 0 uses the data stripping as shown in the figure above. A number of disks are connected together to form a single disk.

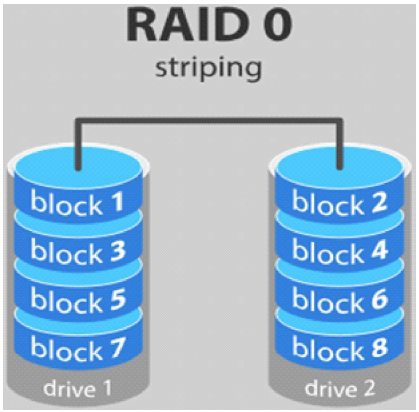


Figure 11.10

In this level, the data are split up into blocks that are written across all the drives in the array. The multiple disks here offer better I/O performance and also a disk system with higher capacity is realized.

RAID 0 does not provide fault handling capacity because if one disk fails, the entire disk system fails.

RAID 1: RAID 1 uses mirroring. It uses pairs of stripped disks and the data is written to both the disk in the pair at once.

If one disk controller is used for both the disk in each pair, the process is called ***mirroring***, and if a separate disk controller is used for each disk in the pair it is called ***duplexing***.

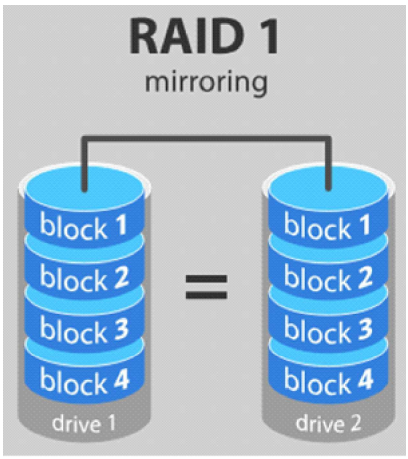


Figure 11.11

RAID 1 is a highly reliable system. If one of the drive fails, then the controller uses either the data drive or the mirror drive for data recovery.

RAID 2: In RAID 2, the data striping is used as in RAID 0 and RAID 1. The data is stripped across two or more disks and one or more extra disk are used to hold error correcting code for the stored data. Thus, if one of the disk crashes, the data of that disk can be regenerated using the appropriate error correcting code held on another disk.

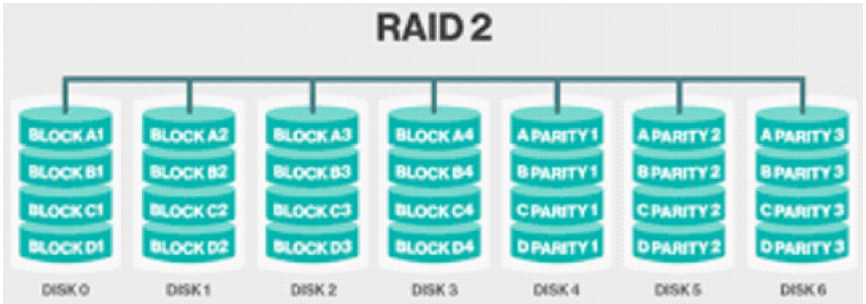


Figure 11.12

These systems are expensive mainly because of the error correcting code that has to be stored. They are rarely used.

RAID 3: This is similar to RAID 2. The data is stripped across a number of disks and an extra disk is used to hold the parity information. Instead of an error correcting code, a simple parity bit is computed for the set of individual bits in the same position. In the event of a disk failure, the data on the failed disk can be revived from the surviving data and parity information.

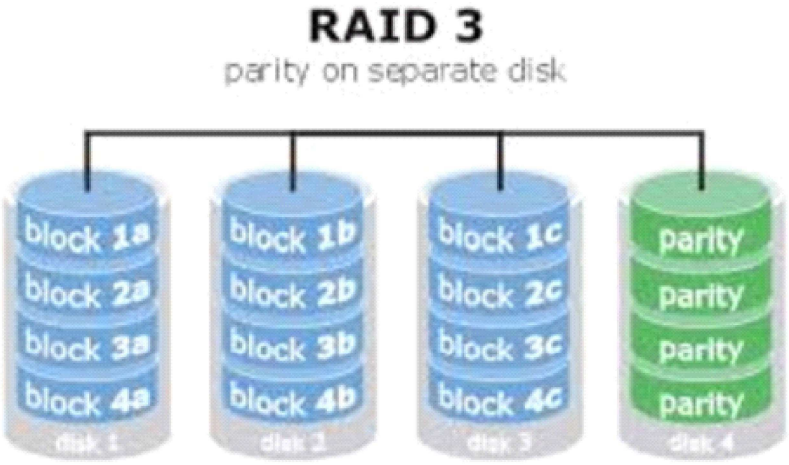


Figure 11.13

RAID 4: RAID 4 consists of a data disks and a parity disk. This level is similar to RAID level 3 because similar stripping method is used. But it employs stripped data in much larger blocks or segments.

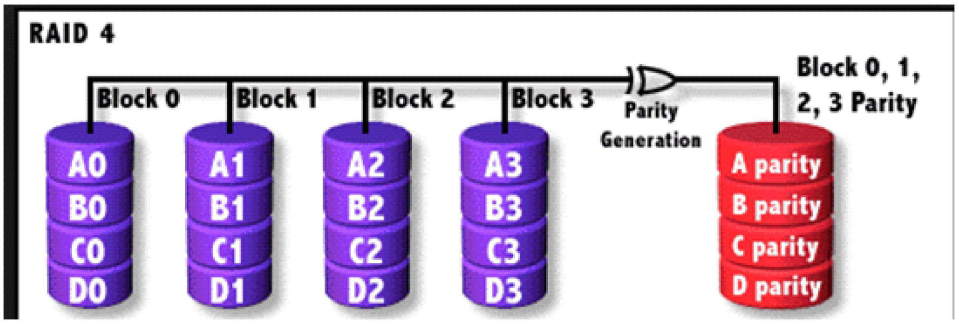


Figure 11.14

As all the parity data is written in to one single disk, so RAID level 4 suffers from a write bottleneck and is not eneally used.

RAID 5: RAID 5 also uses disk stripping with parity. In this configuration, the data and the parity are stripped evenly across all of the disks. So no single disk is a bottleneck. Striping also allows users to reconstruct data in case of a disk failure.

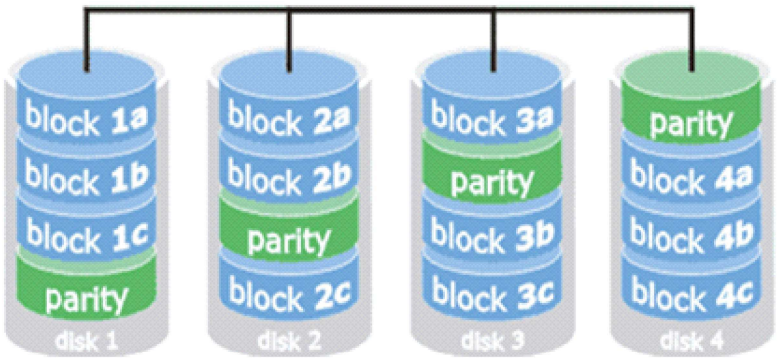


Figure 11.15

RAID 5 used parity instead of mirroring for data redundancy. When data is written to a RAID 5 drive, the system calculates parity and writes that parity into the drive. While mirroring maintains multiple copies of data in each volume to use in case of failure, RAID 5 can rebuild a failed drive using the parity data, which is not kept on a fixed single drive.

RAID 6: As like RAID 5, the RAID level 6 does block level striping. However, it uses dual parity. It creates two parity blocks for each data block. With RAID 6, the RAID array will even survive that second failure.

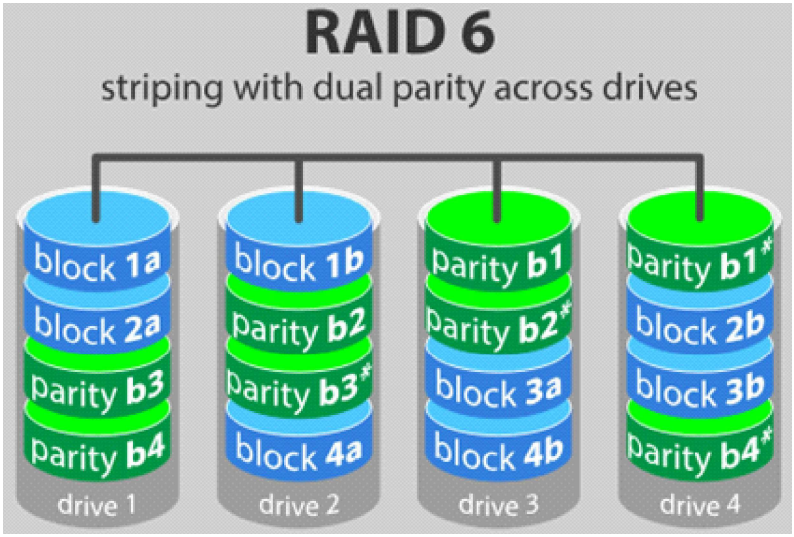


Figure 11.16

If two drives fail, the data can still be accessed, even while the failed drives are being replaced. So RAID 6 is more secure than RAID 5. RAID 6 is a good all-round system that combines efficient storage with excellent security and decent performance. It is preferable over RAID 5 in file and application servers that use many large drives for data storage.

11.11 OPTICAL MEMORY

In 1983, the first optical memory device CD (Compact Disc) was successfully launched. The huge success of CD encouraged the development of optical storage technology.

CD (Compact Disk) and DVD (Digital Versatile Disk) are common examples of optical memory.

Compact Disk: Information is digitally recorded on the compact disk. The recording is done in the form of microscopic dots on the surface of the polycarbonate. It is done by focusing a laser beam on the surface of the disk. The pitted surface is then coated with a highly reflexive surface usually aluminum or gold. The shiny surface protects against dust and scratches by a top coat of clear acrylic.

To record data in a compact disk, the high-intensity laser beam is turned On and Off according to a digital signal. This causes tiny holes or pits are burnt into the disk. This is shown in the figure below:

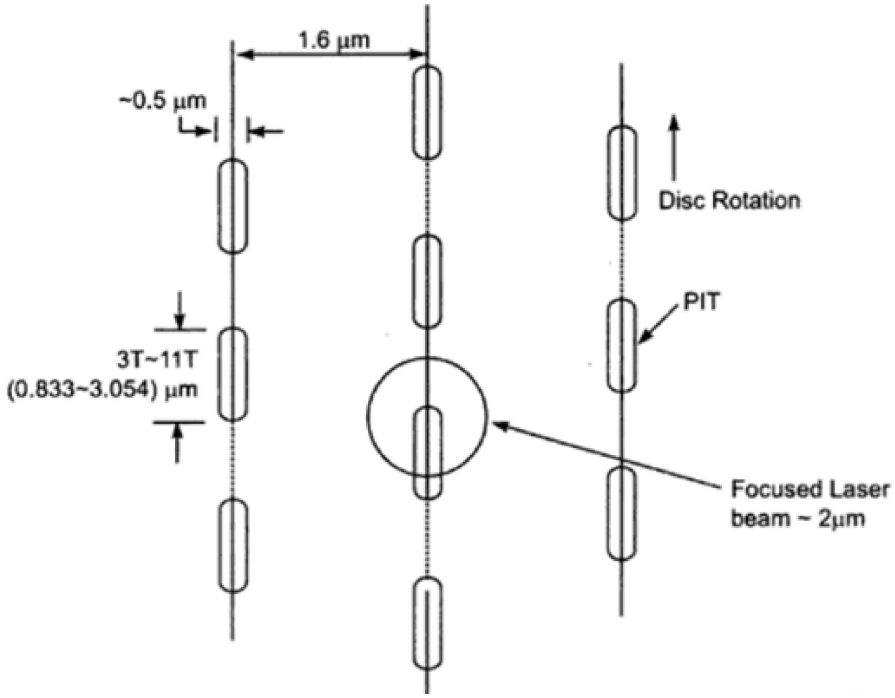


Figure 11.17: Recorded Portion of CD Showing Pits along its Tracks

The recorded data is recovered from the optical disk with an optical pickup, which moves across the surface of the rotating disk. The pickup system is shown in the figure below:

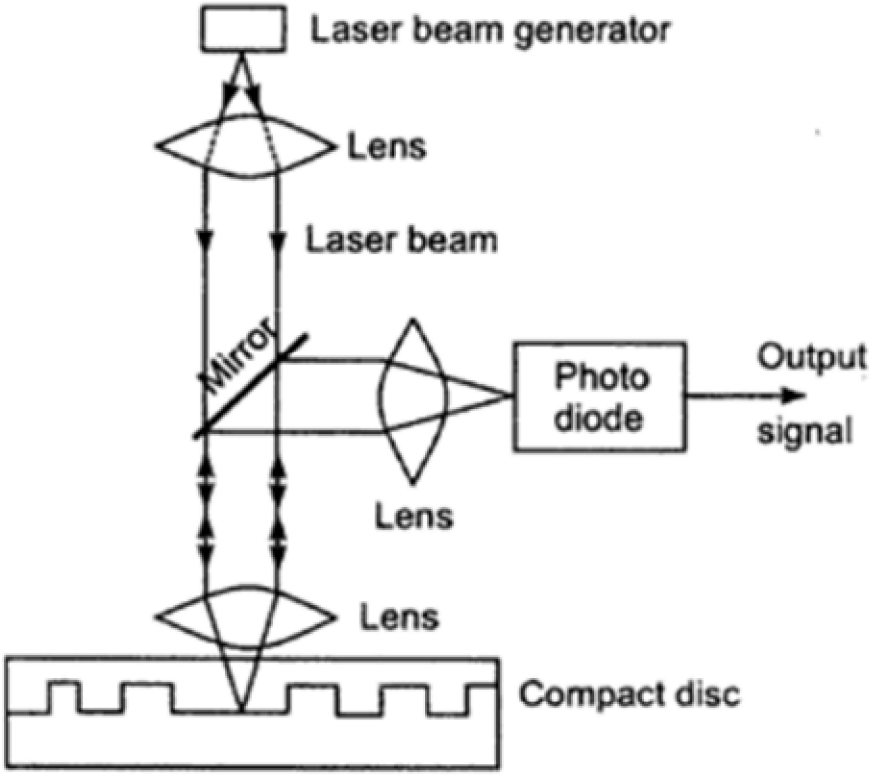


Figure 11.18: Optical Pickup System of a Compact Disk

The area in between the pits is called ***lands***. A land is a smooth surface that reflects back at a higher intensity. A photosensor detects the change between the change between the pits and the lands and converts them to a digital signal.

The information on the compact disks is not stored in tracks. Rather, the disk contains a single spiral track, beginning near the center and spiraling out to the outer edge of the disk. The sectors inside and the outside of the spirals are of the same length. So the information is evenly spread across the disk. The magnetic spots or pits are read by the laser at a *Constant Linear Velocity*. The disk rotates more slowly to access the data near the outer edge.

When data is read from a CD, a laser is directed to the surface of the disk. The laser follows the stream of pits and lands from the inside of the disk to the outer in a spiral direction. As the laser shines on the data track, it reflects one pattern of light off of a pit and a different pattern off of a land area. The resulting reflections equate to a series of ones and zeros by a photo sensor. It is shown in the figure below.

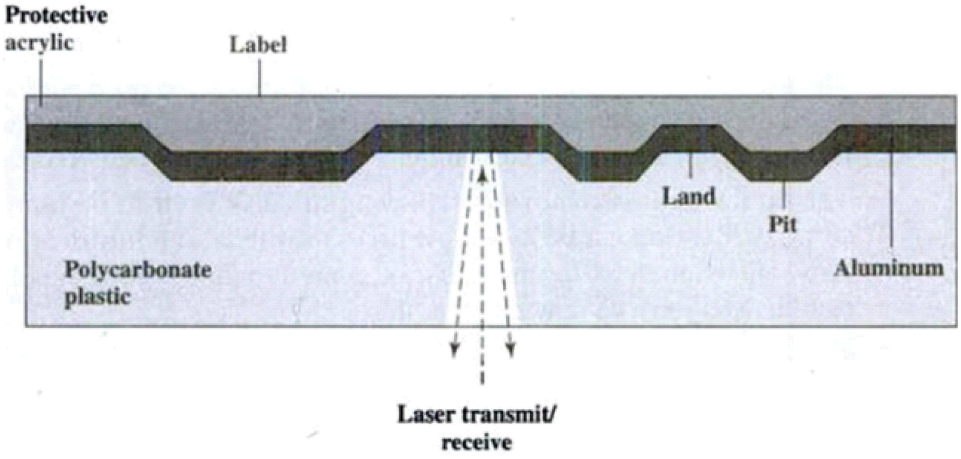


Figure 11.19: CD Operations

Data in CD ROM is stored as a sequence of blocks. It consists of the following fields.

Sync– it identifies the beginning of a block. It has a byte of all 0s, 10 bytes of all 1s and then a byte of all 0s.

Header– it contains the block address and mode byte.

Mode 0 = Blank data field

Mode 1 = 2048 data bytes + error correcting code

Mode 2 = 2336 data bytes

The block format is shown in the figure below.

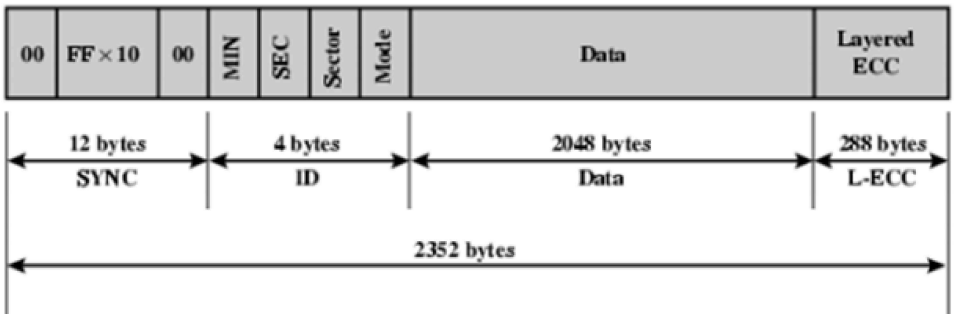


Figure 11.20

The advantages of a compact disk are mainly two. These are

1) The disk together with its information can be replicated inexpensively.
2) It is removable. Thus, the disk can be used for archival storage.

The disadvantages are:

1) The CDs are Read only and cannot be updated.
2) The access time is more than that of a magnetic disk.

DVD (Digital Versatile Disk): The DVDs took the video to the digital age. It is based on the CD technology with increased storage. It has much larger storage capacity. They can be single or double sided. A two layered DVD can hold up to 17 GB of data, video or audio or other information. The DVD has a multiple layers of plastic pieces with a total thickness of 1.2 millimeters. Polycarbonate plastic is basically used for making these layers.

The greater capacity of the DVDs than that of the CDs are due to three reasons as given in the figure below.

1) The bits in a DVD are more closely packed as that of a CD.
2) The DVD has a second layer of pits and lands on top of the first layer. A dual layer DVD has a semi-reflexive layer on top of the reflexive layer. Each layer of the DVD can be read by the lasers.
3) DVDs can be two-sided, whereas the data can be recorded on only one side of a CD. This increases the capacity of the DVD.

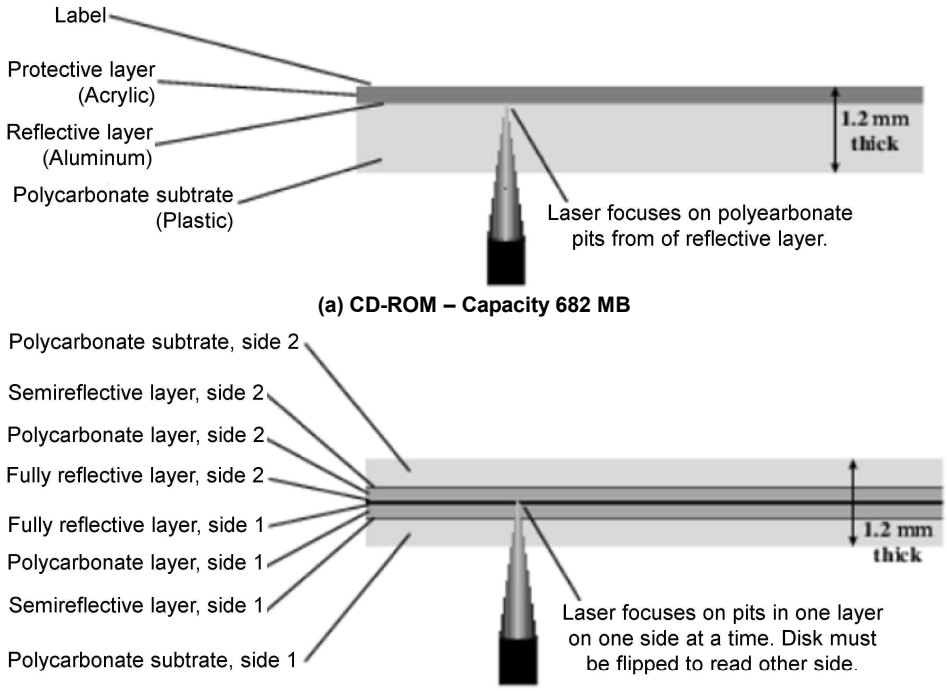


Figure 11.21

11.16 MAGNETIC TAP

Magnetic tape is one of the most popular storage medium for large data that are sequentially accessed and processed. It is a flexible polyester

tape coated with a magnetizable material. Data in the magnetic tape is stored in the form of magnetized and de-magnetized portion on the surface of the tape. The magnetized portion on the surface refers to '1' bit value and the demagnetized portion refers to '0' bit value. The amount of data that can be stored on a linear inch of tape is known as the tape's recording density.

The data in the magnetic tapes are stored in the tracks that run parallel lengthwise. The recording of the data in the parallel tracks is called parallel recording. If the data is laid out as a sequence of bits along each track, the recording is called serial recording. Data are written in blocks which are called the physical records on the tape. The blocks are separated by gaps called the inter-record gaps.

The magnetic tape drive is an example of a sequential storage device. The tape must be rewind or fast forwarded to get the correct block under the read/write head. The magnetic disks, on the other hand, are direct access devices.

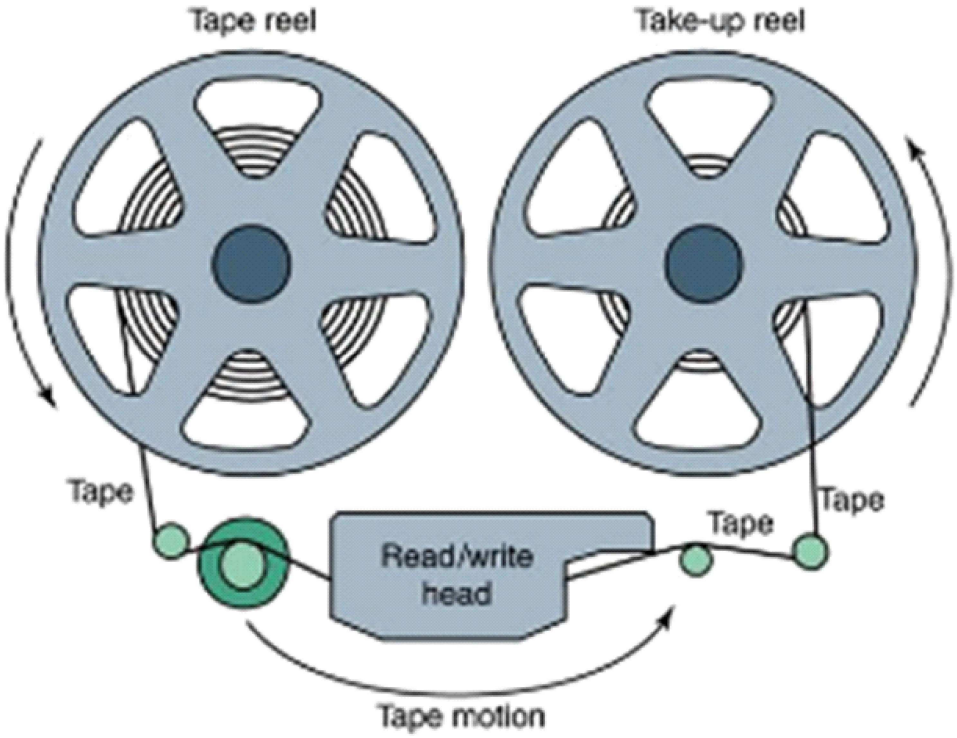


Figure 11.22: A Magnetic Tape Storage Medium



CHECK YOUR PROGRESS

Q.2: Fill In the blanks:

i) The time required to position the read/write head from the current track or cylinder to the desired track or cylinder is called .....

ii) ..... is the time elapsed between the initiation of a particular request for data and the receipt of the data.

iii) The transfer time to or from the disk depends on the ..... of the disk.

iv) The time which is taken for the beginning of the sector to reach the head is called .....

v) RAID is .....

vi) The access speed of the magnetic disks is considerably ..... as compared to the speed of the processor and the main memory.

vii) ..... is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both.

viii) Examples of optical memory are ..... and .....

ix) A ..... detects the change between the change between the pits and the lands and converts them to a digital signal.

x) The magnetic spots or pits are read by the laser at a .....



11.11 LET US SUM UP

- A magnetic disk is a thin, circular metal plate the surface is coated with a magnetic oxide layer usually on both the sides.

- Information is stored on the disks which are represented as magnetic spots. Presence of a magnetic spot represents a 1 bit and the absence of a magnetic spot represents a 0 bit.
- A disk pack may contain anywhere from five to one hundred disks.
- A read/write mechanism is used to achieve the access to read or write on the magnetic disk.
- The data on the magnetic disk is organized in a concentric set of circles called track
- Each of the tracks has the same width as the head and the adjacent tracks are separated by gaps.
- The tracks are numbered from the outermost to the innermost circle starting with zero.
- Each track is divided into sectors.
- The gap in between the tracks in a magnetic disk is called the Inter-track gap and the gap between the sectors in the disks is called inter-sector gap.
- A cylinder is made up of all the tracks of the same number from all the metal disks. Thus cylinder is an imaginary view of all the matching tracks on top of each other.
- There are two types of disks depending upon the two methods of accessing data. One is moving head and the other is fixed head.
- A fixed head disk has many read/write heads on each disk surface as the number of tracks which are mounted on the access arm
- The moving head disk systems have one read/write head for each disk surface which is mounted on the access arm.
- The time required to position the read/write head from the current track or cylinder to the desired track or cylinder is called seek time.
- Access time is the time taken to read a record from the magnetic disk.
- The transfer time is the time taken to transfer a block of bits.
- After the track is selected the disk controller waits until the appropriate sector rotates to line up with the head. This time which is taken for the beginning of the sector to reach the head is called rotational delay or rotational latency.

- RAID is a way of storing data in different places on multiple disks to protect the data in case of a drive failure
- In a RAID, a single large file is stored in several different disks by breaking up the file into a number of smaller pieces and storing the pieces on the separate disks. This is called ***data stripping.***
- CD (Compact Disk) and DVD (Digital Versatile Disk) are common examples of optical memory.



11.18 FURTHER READING

1) Chaudhuri, P. Pal. *Computer Organization and Design*. PHI.
2) Godse, A. P., Godse, D. A. *Computer Architecture*. Technical Publications Pune.
3) Stallings, William. *Computer Organization and Architecture: Designing for Performance*. PHI.



11.19 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: i) Magnetizing coil, magnetic yoke; ii) read/write; iii) electrical current; iv) magnetoresistive; v) tracks; vi) smaller; vii) sectors; viii) Inter-track gap, inter-sector gap

Ans. to Q. No. 2: i) seek time; ii) access time; iii) rotation speed; iv) rotational delay; v) Redundant Array of Independent Disks; vi) Slow; vii) RAID; viii) CD, DVD; ix) Photosensor; x) Constant Linear Velocity



11.1 MODEL QUESTIONS

Q.1: Explain the organization of the magnetic disk.

Q.2: Compare magnetic disk with that of magnetic tape

Q.3: What is RAID technology? Explain the different levels of RAID.

Q.4: What is optical memory? Explain with examples.

*** ***** ***

UNIT 12: PROCESSOR STRUCTURE AND FUNCTION

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Processor Organization

12.4 Registrar Organization

12.4.1 User Visible Registers

12.4.2 Control and Status Registrar

12.5 Steps for Executing a Program

12.6 Instruction Cycle

12.7 Instruction Pipelining

12.8 Let Us Sum Up

12.9 Further Reading

12.10 Answers to Check Your Progress

12.11 Model Question

12.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about processor organization and functioning
- learn about types of user visible registers and their roles
- describe role of control and status registers in processor operations
- describe various steps of instruction cycle
- learn about basics of instruction pipeline.

12.2 INTRODUCTION

This unit discusses internal structure of processor organization. Register which forms the internal memory of a processor are analyzed in this unit. Various functions of registers including user visible register and control and status register are discussed in detail. Instruction cycle and the phases of instruction cycle are also described. The concept of instruction

pipeline which increases the performance of a system are also discussed at the end of this unit.

12.3 PROCESSOR ORGANIZATION

A processor consists of three major components namely Arithmetic Logic Unit (ALU), Control Unit (CU) and Registers. The figure 12.1 gives a simplified view of a processor. All the components within the processor are interconnected through buses. There are two types of buses: internal bus and external bus. The internal bus is responsible for transferring information within the components in the processor. On the other hand the external bus is responsible for transferring data to and from the processor to the memory unit and the input-output devices. The processors internal data bus is connected to external data bus by logic called bi-directional bus or transreceiver.

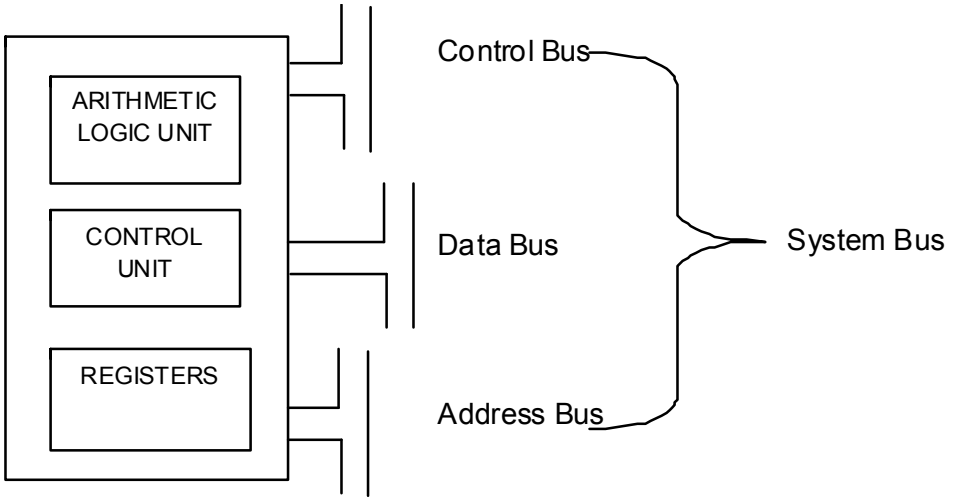


Figure 12.1: Simplified View of Processor Organization

The detailed diagram of a general processor organization is given in figure 12.2. The processor needs a small temporary memory to store data and instruction. It reads the instruction from memory, decodes it, fetches data, executes instruction and finally writes the result back to memory. For doing so it utilizes a set of registers. The ALU does the main processing of the data which are stored in the registers. The Control unit's responsibility is to control the operations of the ALU. The Control unit also checks the movement of data in and out of the ALU.

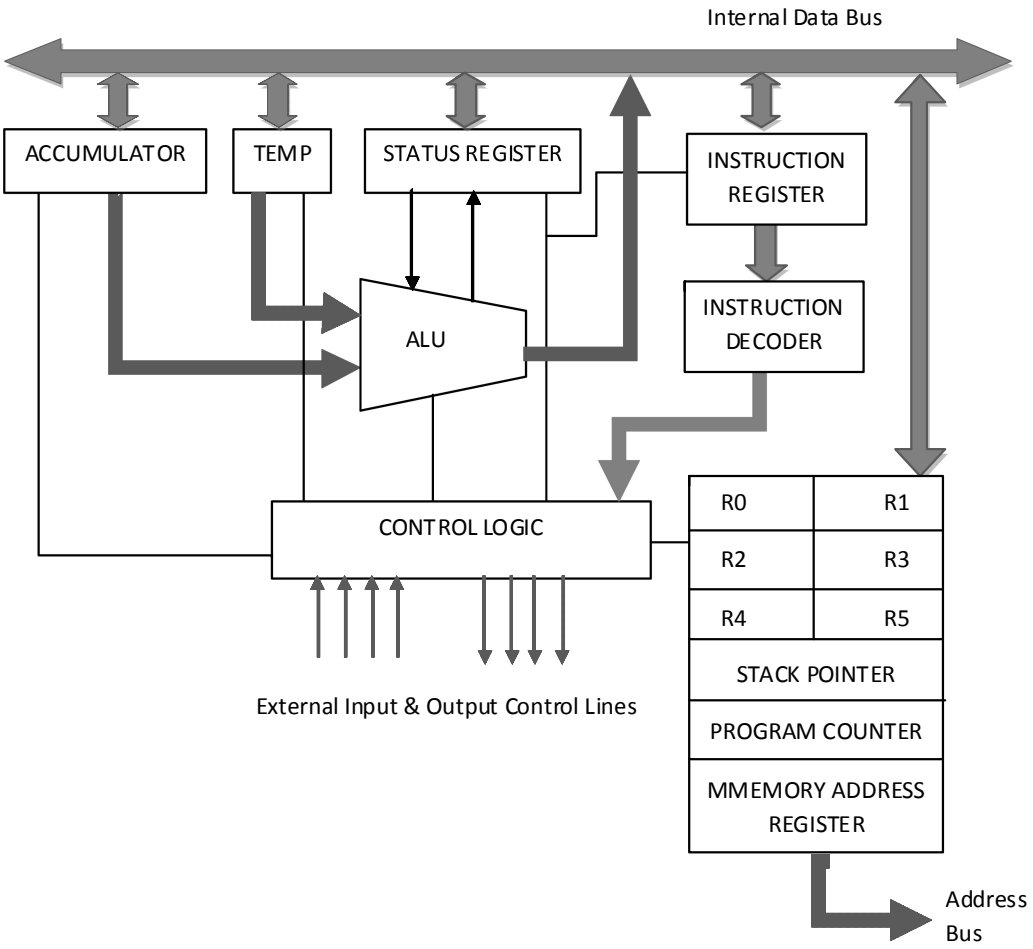


Figure 12.2: General Processor Organization

12.4 REGISTRAR ORGANIZATION

The processor consists of various registers and each register is used for different purposes. Following are the two categories of registers and their functionality.

12.4.1 User Visible Registers

These are the set of registers available to the assembly language programmers so that the main memory references can be minimized. Programmers can make the optimal use of the registers while writing a program. Following are the categories into which user visible register can be divided:

- General Purpose registers
- Accumulator
- Segment Register
- Index Register
- Stack pointer

General Purpose Registers: These are the set of registers ranging from R0 to R(n–1) that are available to the programmers. The programmers can directly make use of these registers in their programs to store the operands and the results. Use of the general purpose registers help minimize memory references therefore making the program execution faster.

Figure 12.3 show general register organization of seven processor registers. It shows how registers are selected by the bit combination of the three select A and select B lines and data is transferred from registers to ALU. Also how the registers are selected using the 3×8 decoders to store the final result. The process involves connecting output of each register to the multiplexer A and multiplexer B. The selection line A & B selects the register outputs and passes them to bus A & B. The two buses form the input to the ALU. The type of ALU operation to be performed on the input is selected by the select operation lines. Finally the result is stored in one of the register. The register selected for storing the result is done using 3×8 decoder.

For example the operation: R1 ← R2 + R3 are executed as

- SELECT A places the content of R2 into bus A
- SELECT B places the content of R3 into bus B
- SELECT OPERATION does the arithmetic addition of the contents of the bus A & B
- SELECT REGISTER is used to select register R1 and transfer the content of the output bus into R1.

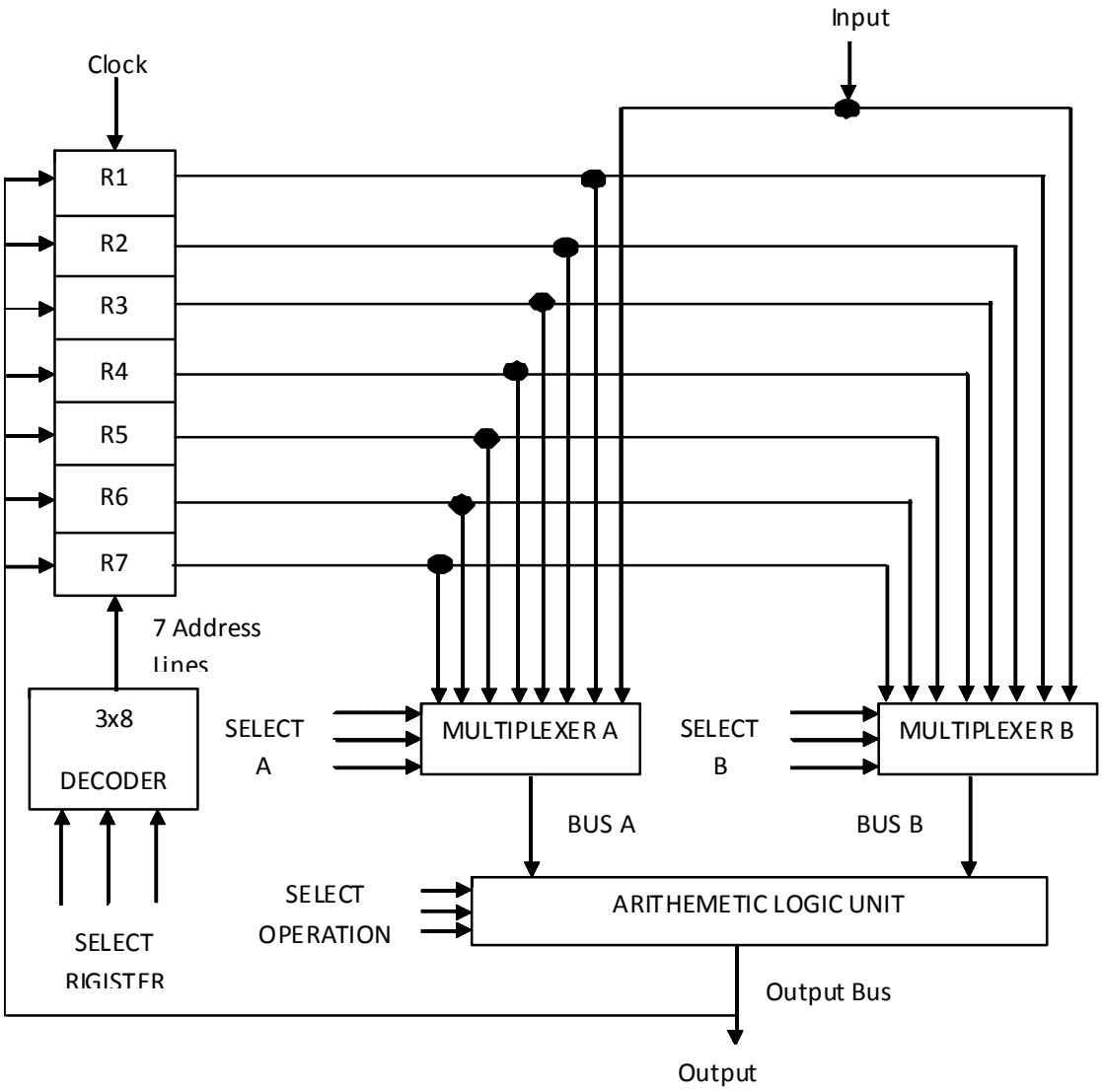


Figure 12.3: General Register Organization

The binary code for selecting register can be understood as:

Binary Code	SELECT A	SELECT B	SELECT REGISTER
000	Input	Input	–
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Accumulator: It is one of the major working register of a processor. For any operation involving two words in ALU, the accumulator holds one of the operand while the other operand is in another register or in memory. For certain operations like complement, the operand must be stored in the accumulator for complementing. Most of the time the result of operation is stored in accumulator which is later transferred to any other register or memory. Also the transfer of data between processor-memory and processor-input output device involves accumulator. It is user accessible and represent by 'A'.

Segment Register: A segment register is generally found in machine that supports segmented addressing. In such cases, the segment register is used to hold the starting address of the segment. There are three main segments namely code segment that contains all instructions of a program to be executed, data segment contains data and constants to be used by the program in the code segment and lastly the stack segment which stores the return address of procedures or functions. In addition to this there is one more segment called extra segment to store data.

Index Register: The machine that supports indexed addressing mode uses index register. For example SI is a source index register for string operations and DI is a destination index register for string operations.

Stack pointer: It is a user visible stack addressing register used to store the address of the top of the stack. It allows implicit stack addressing as the operations PUSH and POP does not need any explicit stack operand.

12.4.2 Control and Status Registers

These are the set of registers that are not visible to users but are required to control the operations in the processor. Following are the categories of these registers:

i) Program Counter

ii) Instruction Register

iii) Memory Address Register

iv) Memory Buffer Register also called Memory Data Register

v) Program Status Word or Status register or condition code register

i) Program Counter (PC): It is an important register used to inform the CPU the address of the next instruction to be executed. A program is a set of instructions which are executed by the CPU in a sequential order. Therefore the program counter helps the CPU by providing the address of the current instruction to be executed and also the address of the instruction to be executed next. The program counter is increment automatically to the next address once the CPU reads the content of program counter. The length of the program counter gives the maximum length of the program in bytes as for example a 16 bit program counter can address to the maximum 2^{16} bytes of memory.

ii) Instruction Register (IR): This register is used to store the instruction fetched from memory specified by the program counter, i.e., the instruction is available in the IR after it is fetched from memory.

iii) Memory Address Register (MAR): This register is used to store the address of the memory location. In a system the MAR connects directly to the address bus. Data are transferred with the help of MAR and MBR. The memory address from which or to which data are to be transferred is specified by the content of the MAR.

iv) Memory Buffer Register (MBR): Contains the data word to be written to memory or read from memory. It works along with MAR. It is also called memory data register. The data is available in the MBR or MDR after reading is completed.

v) Program Status Word (PSW): PSW register contains condition code or status information. When some operation is performed in the ALU, some of the status bits/flags in the PSW is either set/reset. Based on the flags the program can take some logical decisions, for example jump to a specified address when the zero flag is set. Following are the common PSW flags /bits:

Carry: Set to 1 if an operation results in a carry (addition) or borrow (subtraction) out of high order bit.

Zero: Set to 1 when result of an operation is zero.

Sign: Set to 1 if result of the last operation is negative.

Auxiliary Carry: Is set to 1 when addition of first four bits results in carry in the 5^{th} bit.

Equal: Set to 1 if logical comparison operation results in equality.

Overflow: Set to 1 if the result of addition of two signed numbers overflows into sign bit.

Parity: Set to 1 if the operation results in byte that contains even number of one's.

12.5 STEPS FOR EXECUTING PROGRAM

To execute a program/instruction stored in memory the processor goes through the following steps:

- **Fetch Instruction:** The task of fetching the instruction one at a time is achieved with the help of two registers namely Program counter and Instruction Register. The program counter keeps track of which instruction is being executed and what instruction will be executed next. The instruction register is used to store the instruction currently being executed. In addition to this two more registers MAR and MDR are used to transfer data between memory and processor.
- **Interpret Instruction:** After fetching the instruction, the next step is to decode or interpret the instruction so as to know what action needs to be taken. This step also identifies how many bytes the instruction is? This helps to identify how many numbers of bytes of data to read from memory.
- **Fetch Data:** At this step the bytes of data is read from memory or I/O device depending upon the type of instruction; one byte, two byte or three byte. If the instruction is of one byte then data is assumed to be in special register called accumulator. If instruction is of two bytes, first byte represent instruction and second byte represent data. If instruction is of three bytes, first byte represents instruction and second and third

byte represents data. So data of one or two bytes is fetched from memory and stored in processor registers at this step.

- **Process Data:** Once the data and the action to be taken on the data are available, the processor executes the instruction requiring some arithmetic or logical operations.
- **Write Data:** The result obtained from previous step will require the data to be written or stored in memory or I/O device.

12.6 INSTRUCTION CYCLE

It is a process by which a computer goes through a cycle of steps to read instruction and data from memory, execute the instruction and store the result. The figure 12.4 shows an instruction cycle. The instruction cycle is consists of the following four cycles:

- Fetch Cycle
- Indirect Cycle
- Execute Cycle
- Interrupt Cycle

The process starts with fetch cycle. The Interrupt Cycle is always followed by the Fetch Cycle Then Indirect Cycle is followed by the Execute Cycle. In case of fetch and execute cycles, the next cycle is dependent on the current state of the system.

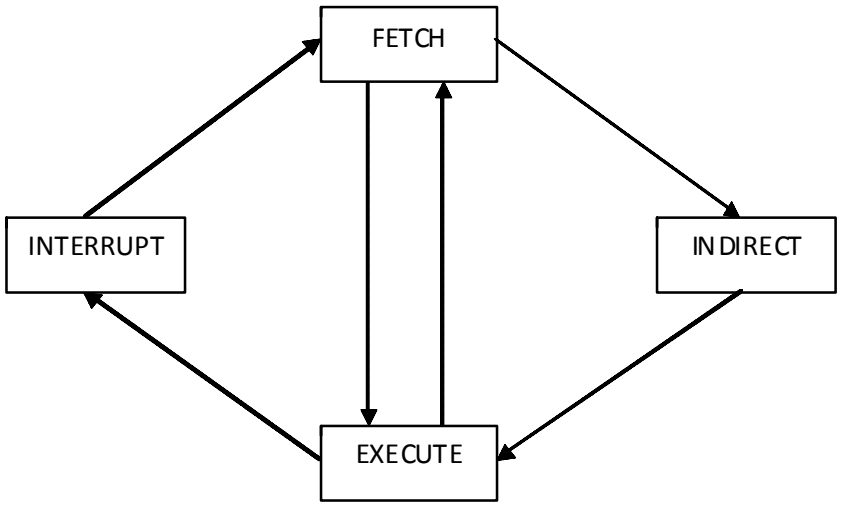


Figure 12.4: Instruction Cycle

The instruction cycle involves the following registers in the process of instruction execution.

- Memory Address Register (MAR)
- Memory Buffer Register (MBR)
- Program Counter (PC)
- Instruction Register (IR)

Fetch cycle: In the beginning of any cycle, the instruction to be executed next is available in the Program Counter (PC). And following steps are taken:

- The content (address) of program counter is placed in the memory address register (MAR). The MAR is connected to the address bus of the system bus.
- The control unit then issues a READ command and the data from the memory specified by MAR is copied to data bus.
- Next the data bus content is copied to memory buffer register (MBR) and program counter (PC) is incremented by one, so that address of the next instruction is ready.
- Finally the MBR content is moved to instruction register (IR).

Indirect cycle: After fetching the instruction and storing it in instruction register, the next step is to fetch the operands by indirect addressing. If the operands are in register then no need to fetch the operands. Otherwise following steps are taken:

- Rightmost N bits of the Memory buffer register (MBR) containing the reference operand address is transferred to the MAR.
- Control unit issues a memory read instruction, and the desired operand address is moved to the MBR.

Execute Cycle: This cycle depends on the type of instruction in the IR. The instruction may be a data transfer instruction, a branch instruction, a memory read or write instruction, and an I/O read or write instruction or some arithmetic or logical operation in the ALU.

Interrupt Cycle: On completion of the execute cycle the processor checks if the any interrupt has occurred or not. If the interrupt is enabled and any interrupt occurs then the interrupt cycle takes place.

- Contents of program counter are transferred to memory buffer register and the contents are saved so that on completion of the interrupt cycle it can return to the point where it transferred the control to interrupt.
- Contents of the PC are saved to the address specified by MAR.
- Next PC is loaded with the starting address of the interrupt service routine.

On completion of the interrupt cycle, previously saved return address is used to fetch the next instruction.

12.7 INSTRUCTION PIPELINING

A pipeline is similar to an assembly line in a production factory. A product has to go through multiple stages in the assembly line before the final product is manufactured. At a time all the stages work simultaneously but on different stages of the product. This process is referred to as pipelining. In a computer system the technique of executing multiple instructions in an overlapped fashion is known as ***pipelining***. A pipeline consists of many stages and these stages are connected to one another in a pipe like structure. An instruction enters one end of the pipeline, goes through several stages before exiting from another end. Pipelining improves the overall throughput of the system.

In a pipeline system, each stage uses register to hold the output of that stage. Output of one stage is applied as input to the next stage.



Figure 12.5: Five Stage Instruction Pipeline

Figure 12.5 shows an example of five stage instruction pipeline consisting of fetch, decode, operand fetch, execute and write stages. Here streams of instructions are executed in overlapped fashion thereby increasing the throughput of the computer system.

Figure 12.6 shows the timing diagram of an instruction pipeline. While the instruction pipeline reads one instruction from the memory, previous

instructions is executed in other stage of the pipeline. Thus multiple instructions are executed simultaneously. From the figure 12.6 it can be observed that while the first instruction started at time period one, the second instruction started at time period two and so on. Up to time period four not all stages were working simultaneously, but from time period five onwards all the five stages are working simultaneously. Therefore from instruction number five onwards each stage is working on a different instruction as:

Instruction 1: Write

Instruction 2: Execute

Instruction 3: Operand Fetch

Instruction 4: Decode

Instruction 5: Fetch

Time →

	1	2	3	4	5	6	7	8	9	10	11
Instruction 1	F	D	OF	E	W						
Instruction 2		F	D	OF	E	W					
Instruction 3			F	D	OF	E	W				
Instruction 4				F	D	OF	E	W			
Instruction 5					F	D	OF	E	W		
Instruction 6						F	D	OF	E	W	
Instruction 7							F	D	OF	E	W

Figure 12.6: Timing Diagram for Instruction Pipeline Operation

If there are ***k*** number of stages and ***n*** number of instructions, then total time ***T*** taken to execute n instructions can be given as:

$$T = k + (n–1)$$

The pipeline performs to its maximum if all the stages take same amount of time to process and there is no branch instruction in the program. In case of a branch instruction, figure 12.7 shows the timing diagram of the branch instruction pipeline assuming instruction four is a branch instruction. Here branch is not taken until the end of time period 7. At this point the pipeline clears all those instruction which are not useful i.e. instruction 5, 6 and 7. Also no instruction complete in time period 9, 10 and 11. This is known

as branch penalty as the processor could not anticipate the branch. So the instruction 17 assumed to be the instruction to be executed on the branch starts at time period eight.

Time → |←Branch Penalty→|

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	F	D	OF	E	W									
Instruction 2		F	D	OF	E	W								
Instruction 3			F	D	OF	E	W							
Instruction 4				F	D	OF	E	W						
Instruction 5					F	D	OF							
Instruction 6						F	D							
Instruction 7							F							
Instruction 17								F	D	OF	E	W		
Instruction 18									F	D	OF	E	W	

Figure 12.7: Timing Diagram for Instruction Pipeline Operation



CHECK YOUR PROGRESS

Q.1: Choose the correct option:

i) A processor consists of–

a) Registers b) ALU, Control Unit

c) ALU, Control Unit, Registers

d) ALU, Control Unit, Registers, Memory

ii) Which of the following is not a user visible register–

a) Accumulator b) Segment Register

c) Index Register d) Program counter

iii) A program consists of the following segment registers–

a) Code, Data b) Code, Data, Stack

c) Code. Data, Stack, Extra

d) Code. Data, Stack, Segment

iv) The register that points to the top of the stack is called–

a) Stack register b) Stack pointer

c) Push Pop pointer d) All of the above

v) The register that is used to keep track of next instruction to be executed is–

a) Program Counter b) Instruction Register

c) Program Status Word d) Stack Pointer

vi) Consider subtraction of two floating point numbers 10101010 and 10101011. Which bit will be set?

a) Sign b) Carry

c) Overflow d) Both (a) and (b)

vii) What will be the auxiliary carry and parity bit if the number 10101011 is complemented

a) 1, 0 b) 0, 1 c) 1, 1 d) 0, 0

viii) Which of the following is not a valid flag?

a) Borrow b) Direction

c) Index d) Overflow

ix) The order in which phases of instruction cycle are executed is

a) Fetch cycle, Indirect cycle, Execute cycle, Interrupt cycle

b) Fetch cycle, Execute cycle, Indirect cycle, Interrupt cycle

c) Fetch cycle, Interrupt cycle, Execute cycle, Indirect cycle

d) Interrupt cycle, Fetch cycle, Execute cycle, Indirect cycle

x) Time taken to execute 10 instruction in a five stage pipeline is–

a) 12 b) 13 c) 14 d) 15



12.8 LET US SUM UP

- A processor consists of three major components namely Arithmetic Logic Unit (ALU), Control Unit (CU) and Registers.

- User Visible Registers are the set of registers available to the assembly language programmers so that the main memory references can be minimized.
- It is one of the major working register in the processor. For any operation in ALU involving two words, the accumulator holds one of the operand while the other operand is in another register or in memory.
- A segment register is generally found in machine that supports segmented addressing. In such cases, the segment register is used to hold the starting address of the segment.
- It is a user visible stack addressing register used to store the address of the top of the stack.
- It is an important register used to inform the CPU the address of the next instruction to be executed.
- Instruction Register is used to store the instruction fetched from memory specified by the program counter.
- Memory Address Register is used to store the address of the memory location.
- Memory Buffer Register Contains the data word to be written to memory or read from memory. It works along with MAR.
- Program Status Word register contains condition code or status information. When some operation is performed in the ALU, some of the status bits/flags in the PSW is either set/reset.
- To execute a program/instruction stored in memory the processor goes through following: fetch instruction, interpret instruction, fetch data, process data, and write data.
- It is a process by which a computer goes through a cycle of steps to read instruction and data from memory, execute the instruction and store the result.
- In a computer system the technique of executing multiple instructions in an overlapped fashion is known pipelining.



12.9 FURTHER READING

1) Hamacher, Carl. (5th Edition). *Computer Organization*. McGraw Hill.
2) Mano, M. Morris; Mall, Rajib. (3rd Edition). *Computer System Architecture*. Pearson.
3) Stallings, William. (10th Edition). *Computer Organization and Architecture: Designing for Performance*. Pearson.



12.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: i) (c), ii) (d), iii) (c), iv) (b), v) (a), vi) (d), vii) (d), viii) (c), ix) (a), x) (c)



12.11 MODEL QUESTION

Q.1: Explain the roles of user visible processor registers?

Q.2: Discuss segment registers role in a user program.

Q.3: What is a program status word? How different flag bits contribute in an ALU operation?

Q.4: What are the various phases of instruction cycle?

Q.5: Explain instruction pipeline using a suitable example.

Q.6: Briefly explain how instruction pipeline can contribute to conditional branch instruction.

Q.7: Explain how memory address register and memory buffer register help in data transfer operation.

Q.8: Using an example show the difference between Carry flag and Overflow flag.

Q.9: Consider addition of two 8-bit numbers 00110111 and 11100010. What will be the value of following flags: (a) Carry, (b) Auxiliary Carry, (c) Sign, (d) Zero, (e) Parity

*** ***** ***

UNIT 13: INTRODUCTION TO PARALLEL PROCESSING

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Pipelining

- 13.3.1 Hazards of Pipelining
- 13.3.2 Structural Hazard
- 13.3.3 Data Hazard
- 13.3.4 Control Hazard

13.4 Parallel Processing

13.5 Let Us Sum Up

13.6 Answers to Check Your Progress

13.7 Further Reading

13.8 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn the basic concepts of pipelining and the hazards associated with it
- describe the concepts of superscalar and parallel processing.

13.2 INTRODUCTION

We have already come to know the basic organization of a computer and how a processor executes a program. To execute a sequence of instructions, the processor uses a cycle of fetch-decode-execute operations: an instruction is fetched from memory, it is then decoded and executed; the next instruction in the sequence is fetched, decoded and executed and the cycle is repeated until the end of the program (or sequence of instruction) is reached.

In this unit, we shall consider only the three stages viz. Fetch, Decode and Execute and discuss how the simple fetch- decode- execute cycle can be optimized so that performance of the system is improved. Pipelined execution of instructions is based on the fact that the different phases of the fetch-decode-execute cycle require completely different hardware units. Pipelining involves reordering instructions so that all hardware units are used optimally to achieve better performance. Parallel processing further involves executing several independent instructions in the program at the same time, to achieve better performance.

13.3 PIPELINING

The basic idea of pipelining is similar to the assembly line operation in manufacturing plants. For example, in a car manufacturing plant, there is a sequence of stations each of which perform different stages of the assembling the car. The first station prepares the chassis, the next station adds the body to the chassis, the next station installs the engine and so on. While one station is preparing the chassis of one car, the second station is adding the body to the chassis of another car, the third station is installing the engine in yet another car and so on. This way, all stations are fully utilized at all times so that an optimal number of cars can be manufactured. Similarly, in pipelining, the execution of a single instruction has at least three different stages (fetch, decode and execute) that are performed by different hardware units. While the system bus is used to fetch an instruction from the memory, the decoder unit of CPU can be used to decode a different instruction and the ALU can be used to execute yet another instruction. Compared to sequential execution of instruction, pipelined execution leads to considerably better performance.

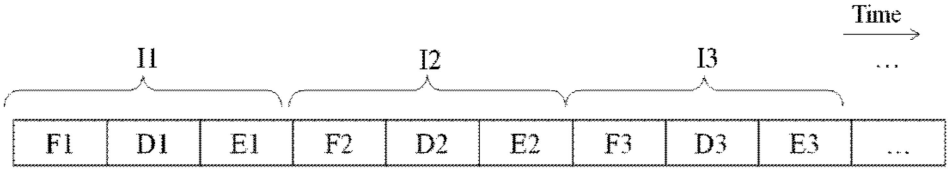


Figure 13.1: (a) Sequential Execution of Instructions I1, I2, I3, where F1, D1, E1 means fetch, decode and execution phase of instruction 1.

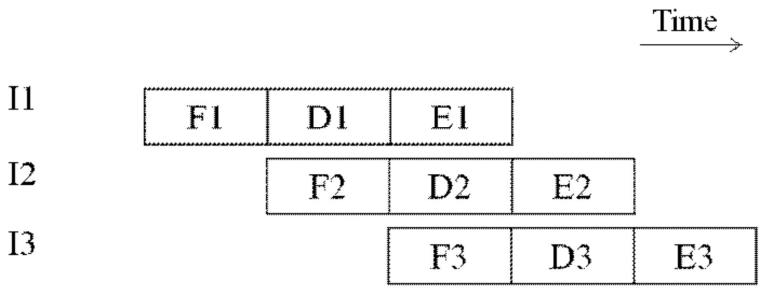


Figure 13.1: (b) Pipelined Execution of Instructions I1, I2, I3

Pipelined execution of sequence of instructions is shown in figure 13.1(b). The time taken to complete any of the fetch, decode or execute phases of an instruction is one clock cycle. While sequential execution (Figure 13.1(a)) completes execution of instructions I1, I2 and I3 in nine clock cycles, pipelined execution completes the same in just five clock cycles. An even greater difference in the number of clock cycles is seen if the total number of instructions to be executed is more. The number of pipeline stages may vary depending on the number of separate hardware units available in the system. The increase in performance, as a result of pipelining, is proportional to the total number of pipeline stages.

13.3.1 Hazards of Pipelining

The increase in performance, due to pipelining, is proportional to the number of pipelining stages only if the pipelined executions (as shown in Figure 13.1 (b)) can be continued throughout the entire program. However, during pipelined execution of a program, there are many situations when one or more stages of executions may be halted or ***stalled*** temporarily. As a result, it is often not possible to achieve an uninterrupted pipelined execution of instructions.

The pipelined execution of instructions may be interrupted due to various reasons– for example, the execution phase of some instruction requires more than one clock cycle. Figure 13.2 shows a situation where the execution phase of I1 (E1) requires two clock cycles instead of one. The execution phase of I2 (E2) has to be ***stalled*** for one clock cycle– instead of being completed in clock

cycle 4, it is completed in clock cycle 5. This is because the execution unit had not completed E1 and can handle execution of only instruction at a time. Any condition that causes the pipeline to stall is called a ***hazard***.

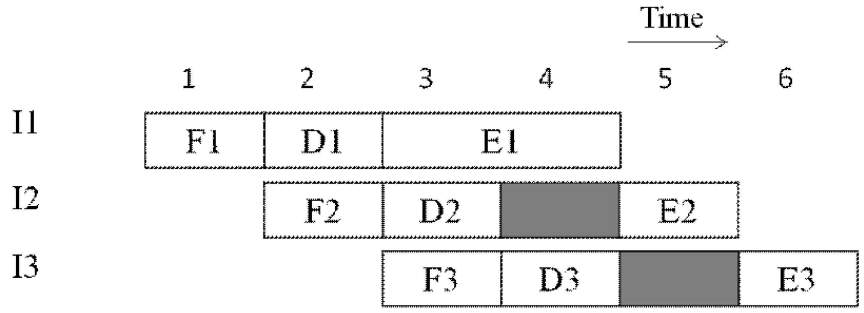


Figure 13.2: Pipelined execution of instructions when interrupted or *stalled*

Based on the type of condition that has caused the pipeline to stall, pipelining hazards can be classified as:

- Structural hazards
- Data hazards
- Control hazards

13.3.2 Structural Hazard

Structural hazard arises when two instructions in the pipeline require the use of the same hardware resource at the same time. One of the most common hardware resources that can lead to structural hazard is memory. Say, there is a sequence of instructions as follows:

I1: Add A, 5

I2: Add B, 5

I3: Load R1, C

Here, A, B, C are references to memory locations and R1 is a register. Each of the arithmetic instructions (I1 and I2) can be split into stages to: fetch instruction (F), decode instruction (D), fetch operand (M), execute arithmetic operation (E) and write result (W). Out of these, F, M and W require memory access. Similarly,

instruction I3 can be split into stages to: fetch instruction (F), decode instruction (D), and to execute the load instruction a memory access operation is require (M). The pipelined execution of these three instructions is shown in Figure 13.3 during clock cycle 5, both instruction I1 and I3 require memory access. As a result, I3 has to be stalled for clock cycle to allow I1 to complete. During clock cycle 6, again I2 and I3 both require memory access. Once again I3 is stalled to allow I2 to complete first.

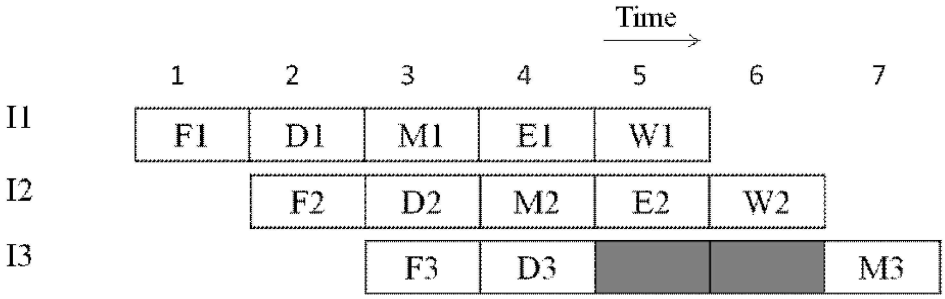


Figure 13.3: Structural Hazard caused by I1 and I3 Requiring Memory Access at the same time

To handle structural hazards, in general the number of hardware resources may be increased.

13.3.3 Data Hazard

Data hazards occur when an instruction operates on a data that is to be produced by some other instruction in the pipeline. Say there is sequence of instructions as follows:

I1: Add A, 5

I2: Load R1, A

I3: Add R2, R1

The pipelined execution of these instructions is shown in figure 13.4. In this sequence, immediately after adding 5 to the memory location A, the value of A has to be loaded to register R1. However, in the pipelined execution, even before the result of the addition operation is written into memory location A, instruction I2 attempts to access the value in A. If the initial value of A is 4, then after execution of I1 the value changes to 9 and the value loaded into

register R1 should be 9. However, due to pipelining, even before the value 9 is stored in memory location A, I2 is executed and the value loaded into register R1 is 4. Such an incorrect execution will affect the execution of instruction I3 and the end result.

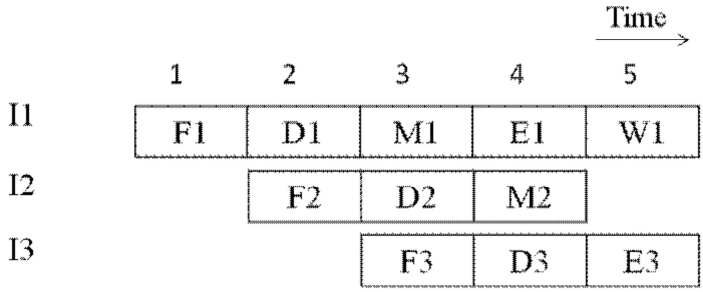


Figure 13.4: Effect of Data Hazard if not handled properly

It is difficult to deal with data hazards because such hazards are not detected easily. In order for the above example to be correctly executed, the system will have to detect that both I1 and I2 are working on the same data. This can be done only at the software level during compilation. Once detected, the execution of the second instruction can be stalled until the previous instruction has completed execution. Another way to handle data hazard would be to reorder the complete sequence of instructions such that by the time I2 is executed, the write results phase of I1 will be completed.

13.3.5 Control Hazards

In pipelining, the instruction fetch unit fetches subsequent instructions in advance so that the decode/execute unit does not have to wait. As a result, the execute unit is used optimally because it always has a steady stream of instructions that are ready for execution. However, this is only true if the sequence of instructions that is to be executed next is well-defined. In case of branching instructions (conditional and unconditional JUMP statements) the sequence of instructions to be executed next depends on the result of the current instruction in the pipeline. Such hazards caused by transfer of control instructions are called control hazards or instruction hazards. Say there is a sequence of instructions as follows:

I1: CMP A, 10

I2: JG label

I3: ADD R1, R2

I4: SUB B, 1

I5: label: SUB A, B

I6: ADD B, 1

The pipelined execution of this sequence of instructions is shown in figure 13.5. Here, whether instruction I3 is executed or I5 is executed depends on the result of the instruction I1 – if A is greater than 10 then I5 is executed, otherwise I3 is executed. However, as shown in the figure, the result of I1 is not obtained until clock cycle 4. But pipelining requires instructions I3 and I4 to be fetched even before that. Therefore in cases where A is greater than 10, redundant work (fetching of I3 and I4) is done by pipelining. This redundant work has to be discarded before the proper execution of the program can be continued.

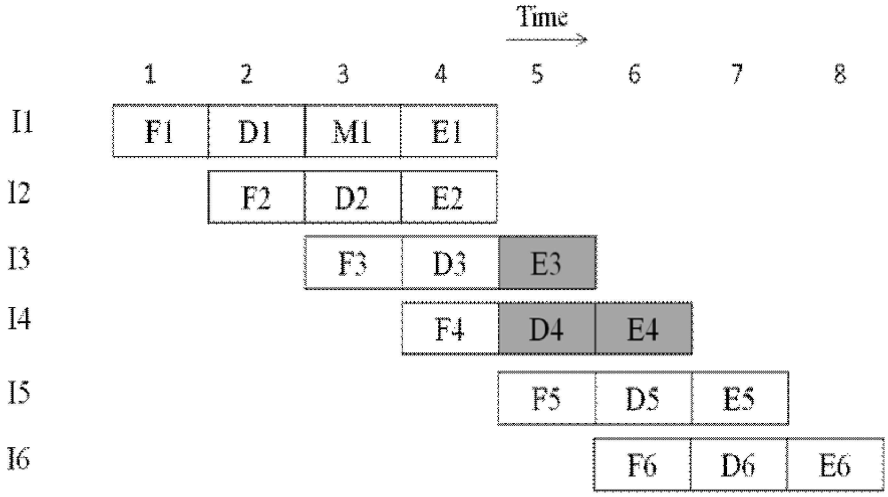


Figure 13.5: Effect of Control Hazard if not handled properly

Similar to data hazards, control hazards can be detected at the software level during compilation. To reduce the amount of redundant work done due to control hazards, the sequence of instructions can be stalled until execution of I1 is complete. Further, to efficiently utilize all clock cycles the sequence of instructions can be reordered. The reordering is done such that the execution of I1 is completed before fetching I3 or I4. It has to be ensured that the

instructions that are fetched immediately after I1 are instructions that have to be executed regardless of the outcome of the comparison operation. If such a reordering of execution is not possible then at the compilation stage itself a prediction is made as to which branch will be taken and the instructions are fetched accordingly. However, there is still a possibility that the branch is incorrectly predicted and any changes made due to the pipelined execution of the incorrect branch may have to be discarded later.



CHECK YOUR PROGRESS

Q.1: What are the different categories of pipelining hazards?

.....

.....

Q.2: When do data hazards occur?

.....

.....

Q.3: Which pipelining hazards may occur in the following sequence of instructions?

```
ADD     A, AH
SUB     AH, 30h
ADD     CH, B
CMP     CH, 10
JG      label1
ADD     AH, B
JMP     label2
label1: SUB     AH, B
label2: MOV     BH, B
```

13.4 PARALLEL PROCESSING

Traditionally, the execution of instructions in a computer is assumed to be strictly sequential in nature. However, with pipelined execution of

instructions, the sequential view of program execution is somewhat inaccurate. During pipelined execution, at any given clock cycle, there are several instructions in the pipeline that are in different stages of their execution. However, only a single instruction is in its execution stage at any given clock cycle. This is taken a step further with ***superscalar processors*** that allow multiple instructions to be executed at the same time.

Superscalar Processors: Superscalar organizations have several execution units within a single processor. This allows multiple instructions from the same program to be executed in parallel. As a result the number instructions executed per cycle is more than one. This is also known as ***instruction-level parallelism***. Figure 9.6 shows an example of instruction level parallelism in a superscalar processor with two execution units.

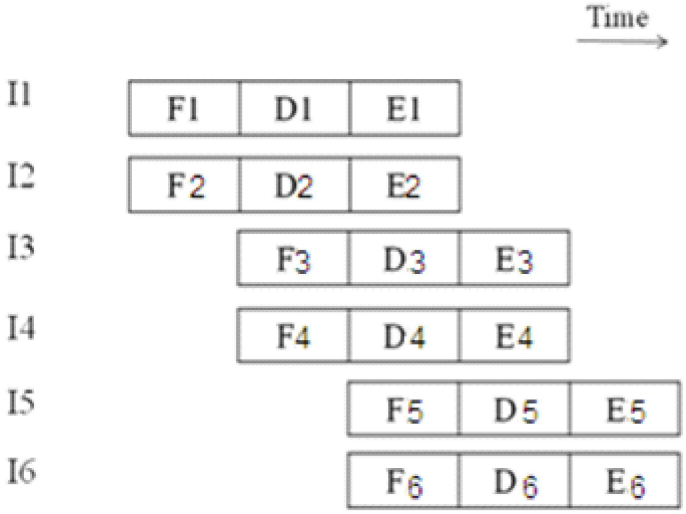


Figure 13.6: Instruction Level Parallelism in a Superscalar Processor with Two Execution Units

However, such an aggressive approach to increase performance of the system also has several issues. The various hazards encountered in pipelining are encountered in superscalar processors as well and are seen to be even more prominent. Further, most programs are written assuming a sequential execution of the instructions, with several dependencies amongst the instructions. In superscalar processors, multiple instructions are executed at the same time. If the dependencies amongst the instructions being executed parallel are not handled, then it may result in incorrect executions. Thus in such systems, in order to achieve high performance

and correct executions, it is important for the compiler to be able to deal with the hazards and the dependencies effectively by proper selection and reordering of the instructions.



13.5 LET US SUM UP

- Pipelining is based on dividing the work done in the instruction execution cycle into stages such that each stage can be executed by completely different hardware units.
- Pipelining involves concurrent execution of multiple instructions even though the instructions in the pipeline are in different stages of their execution. Therefore, pipelining does not follow a strictly sequential execution of instruction.
- Pipelining hazards occur because most programs are written for a sequential execution. Instructions in a program have several dependencies amongst themselves which are properly executed only if executed sequentially.
- There are three most common types of pipelining hazards: data hazards, control hazards, structural hazards.
- Pipelining aims to achieve a performance of one instruction per cycle.
- Superscalar processors have more than one execution units within the same processor. When combined with pipelining the performance of superscalar processors is expected to be more than one instruction per cycle.
- Superscalar processors encounter more aggressive forms of pipelining hazards.



13.6 FURTHER READING

1) Hamacher, V. C., Vranesic, Z. G., Zaky, S. G., Vransic, Z. & Zakay, S. (1984). *Computer Organization* (Vol. 3). New York et al.: McGraw-Hill.

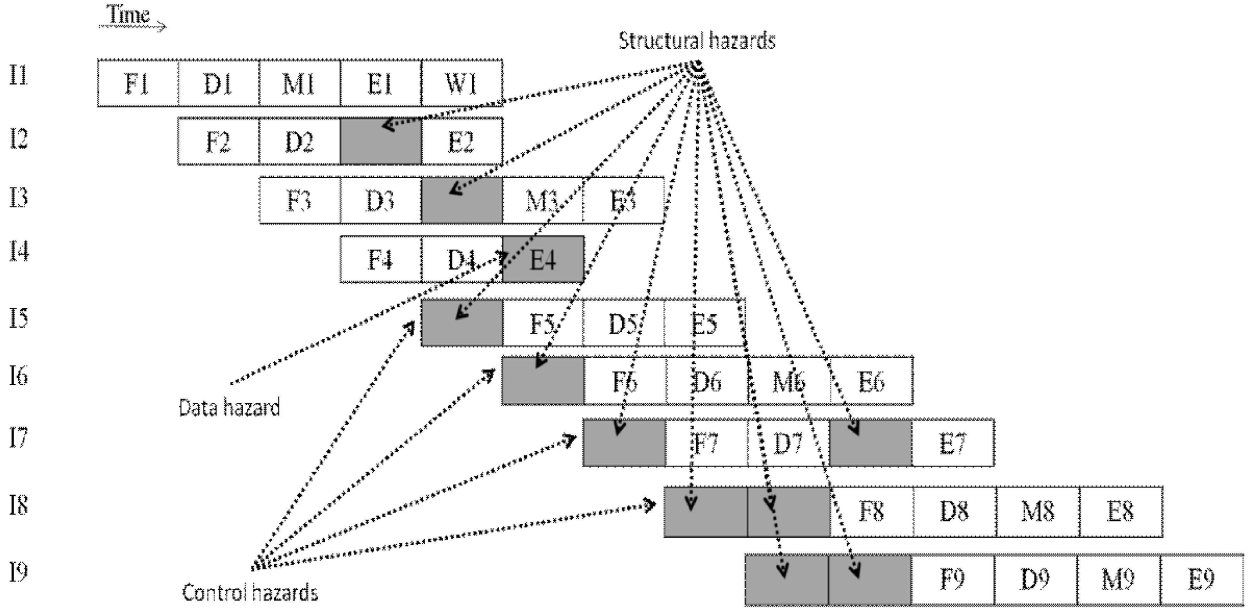
2) Stallings, William. (2006). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.

13.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: The different categories of pipelining hazards are: Structural hazards, Data Hazards and Control hazards.

Ans. to Q. No. 2: Data hazards occur when an instruction operates on a data that is also being operated on some by some other instruction in the pipeline.

Ans. to Q. No. 3:



13.8 MODEL QUESTIONS

Q.1: What is pipelining? Explain the pipelining hazards with example.

Q.2: What are superscalar processors?

Q.3: Differentiate between pipelining and superscalar processors.

Q.4: What is instruction level parallelism?

*** ***** ***

UNIT 14: CISC AND RISC ARCHITECTURE

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Complex Instruction Set Computers (CISC)

14.3.1 Performance Analysis of CISC

14.4 Reduced Instruction Set Computers (RISC)

14.4.1 Performance Analysis of RISC

14.5 Let Us Sum Up

14.6 Further Reading

14.7 Answers to Check Your Progress

14.8 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- learn the basic concepts CISC architecture and its performance issues
- describe the basic concepts RISC architecture and its performance issues
- differentiate between CISC and RISC architecture.

14.2 INTRODUCTION

With the development and evolution of high level languages has led to a semantic gap, between the kind of operations allowed by these high level languages and the kind of operations allowed by the computer architecture. To meet the requirements of complex high level languages, complex architectures were designed that offered larger number of basic operations. On the other hand a reduced instruction set computer focuses on a simpler set of instructions and fewer basic operations that can be executed in one clock cycle.

In this unit, we shall discuss the problems associated with complex architectures and how reducing the complexity of the architectures may lead to improved performances.

14.3 COMPLEX INSTRUCTION SET COMPUTERS (CISC)

High level languages (HLL) often include complex operations to make it easier for the programmer to express the algorithms. There is a semantic gap between the complex operations provided by the HLLs and the operations provided by the architecture. The translation of these complex operations into the basic operations that the architecture provides is left to the compiler. As a result the programs generated by the compilers are large and not efficient enough. To close this semantic gap, complex architectures with large instructions sets, several addressing modes, and more number of basic operations were designed to meet the requirements of the high level languages. Computers with such architectures are called ***Complex Instruction Set Computers*** (CISC).

Such complex instruction sets are expected to reduce the complexity of compilers and improve the efficiency of the code generated by the compilers. CISC architectures aim to complete any task in as few number of machine instructions as possible. To accomplish that, a large set of complex instructions are directly incorporated into hardware. Such a complex architecture also provides support for even more sophisticated HLLs.

The compilers in CISC architectures are much simpler and the machine code generated is also much smaller in size. As a result, another advantage to CISC architecture is that programs require lesser memory. Further, smaller programs have the advantage that the number of *instruction fetch* operations during the execution of the program will be much smaller. This is expected to reduce execution time of the program.

14.3.1 Performance Analysis of CISC

CISC architectures were originally designed to simplify compilers for high level language and enable compilers to generate

more concise machine code. In an attempt to do so, complex operations are incorporated into the hardware. However, it has been noted that it is often difficult to exploit such a complex set of machine instructions. In order to use the complex instruction set properly the compiler has to look for the machine language translation that exactly fits a HLL instruction. This becomes difficult and time consuming when the number of choices that the compiler has for translating some HLL instruction is very large.

Further, CISC architecture compilers were expected to reduce the size of machine code generated. However, with the advances in technology, memory has become much cheaper and therefore is no longer a constraint. Additionally, it is not always true that CISC architectures generate smaller size machine code because the length of a single machine instruction is much larger. As the number of opcodes increase with the number basic machine operations available, the number of bits required to specify the opcodes and the various addressing modes also increase.

In CISC architectures instruction execution was expected to be faster. This is because in order to execute a single HLL instruction much fewer number of machine instructions need to be fetched and executed. However, in practice this is not always so. To allow for such a sophisticated set of instructions, either the control unit needs to be much more complex or in case of micro programmed control unit, the number of stored micro programs will be much larger. In both of these cases executing a single machine instruction would require more number of clock cycles. This is true not only for the complex operations but also for the simpler operations that are more frequently used during program execution.

14.4 REDUCED INSTRUCTION SET COMPUTERS (RISC)

While CISC architectures were initially designed to reduce memory requirements, complexity of compilers and memory requirements,

researchers have later recognized there are several disadvantages to the approach. Because of these disadvantages, several groups have decided to pursue a completely opposite approach to CISC architecture. The Reduced Instruction Set Computers (RISC) have characteristics that are completely opposite of CISCs. Instead of having complex hardware to simplify the software (viz. compilers), RISC architecture aims to simplify the hardware at the cost of complex software. At the hardware level only the most basic and most frequently used operations are incorporated, in order to keep the number of basic operations as small as possible. As a result of this, even though the compilers are more complex and the number of machine instructions generated for a HLL program is much larger, the time required to execute a single machine instruction is constant. This enables the processor to execute programs at the rate of *one instruction per cycle.* With such a simple set of machine instructions, it becomes possible to have simple hardwired control unit. As a result, a single machine instruction can be executed much faster.

RISC architectures usually ensure faster execution of operations by ensuring that instructions are executed on operands that are stored in registers. Registers are much faster storage devices and therefore if frequently used operands remain on registers, the execution time becomes much faster. Further, this ensures that control unit is even simpler because the only operations that require memory access are simple LOAD and STORE operations. This characteristic of RISCs necessitates a large number of registers available for the processor in order to keep the operands on the registers for a longer duration and further reduce the number of memory access operations.

Since most instructions in RISC architectures allow only register-register operations, the number of addressing modes required is also significantly reduced. This further simplifies the instruction set and the control unit. A simple instruction set in turn leads to simple instruction formats. Instruction length in RISC architectures are fixed and are usually aligned to word boundaries. The number of bits required for opcode is fixed and relatively small because of the small number of available basic operations.

The number of bits required for specifying addressing of operands is also fixed and small, because all operands are in registers and registers have shorter address lengths. The simple instruction format requires simple hardware for decoding in the control unit and a uniform number of instruction fetch operations. All characteristics of RISC architectures are aimed at simplifying the control unit to improve efficiency of instruction execution.

14.4.1 Performance Analysis of RISC

The characteristics with which RISC architectures were designed allow for several performance benefits. While the complexity of compilers is increased with RISC architecture, it is also true that a lot of power to generate more efficient machine code rests with the compiler. The compilers are better enabled to optimize the machine code because the available machine instructions are more primitive. Further, the compiler is not required to find an exact match of machine instruction for every HLL instruction. Instead, depending on how well the compiler is designed, it may be possible to generate a more effective and efficient machine code for a given HLL program.

Most instructions in any HLL program involve only simple operations. As a result the machine code generated by the compiler would involve a lot of simple operations that can be executed much faster due to the simple control unit.

The instruction format in RISC architecture is relatively simple and uniform. As a result the various stages of execution for all instructions are also more or less uniform. This uniformity allows for pipelining to be implemented more efficiently than compared to CISC architectures.

However, most of the perceived advantages of RISC over CISC are based on a lot circumstantial evidence. It is worth noting that there are certain features of CISC architectures that can be incorporated in a RISC design and vice versa for a more well-rounded system. Table 10.1 lists some of the key differences between RISC and CISC architectures.

Table 14.1: Differences between RISC and CISC

CISC	RISC
➢ Emphasis on hardware	➢ Emphasis on software
➢ Includes multi-clock complex instructions	➢ Single-clock, reduced instruction only
➢ Small code sizes, high cycles per second	➢ Low cycles per second, large code sizes
➢ Transistors used for storing complex instructions	➢ Spends more transistors on storage registers



CHECK YOUR PROGRESS

Q.1: Give the full form of CISC and RISC.

.....

.....

.....

Q.2: Give two differences between CISC and RISC.

.....

.....

.....

14.5 LET US SUM UP

- CISC architectures involve incorporating complex operations at the hardware level to enable simple translation of high level language to machine code.
- CISC architectures aim towards simplifying compiler design, support for very high level languages, and reduction of memory requirements for program storage.
- CISC machine instructions allow a large number of basic operations and addressing modes. This increases the number of bits required to specify a single instruction.

- CISC architectures have complex hardware and require a large number of microprograms. This increases the execution time for a single instruction to several clock cycles.
- RISC architectures are aimed at simplifying hardware at the cost of complex compilers. Compilers in RISC architectures are expected to be able to effectively translate HLL programs.
- RISC architectures incorporate only the most basic and frequently used instructions at the hardware level. This significantly reduces the hardware complexity.
- RISC architecture allows execution of a single machine instruction in one clock cycle.
- RISC architectures have fewer operations and addressing modes. This reduces the length of a single instruction and allows for uniformity in the instruction format.
- RISC architectures incorporate a large number of registers as most operations are performed only on register operands. This increases cost due to the number of registers but significantly reduces execution time.
- RISC architectures allow for a more effective implementation of pipelining.
- An architecture that incorporates characteristics of both RISC and CISC would have a more balanced effective approach towards reducing complexity and execution time.



14.6 FURTHER READING

1) Hamacher, V. C., Vranesic, Z. G., Zaky, S. G., Vransic, Z. & Zakay, S. (1984). Computer organization (Vol. 3). New York et al.: McGraw-Hill.
2) Stallings, William. (2000). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.



14.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: The full forms are:

CISC: Complex Instruction Set Computers

RISC: Reduced Instruction Set Computers

Ans. to Q. No. 2: Two differences between CISC and RISC are:

CISC	RISC
➢ Emphasis on hardware	➢ Emphasis on software
➢ Includes multi-clock complex instructions	➢ Single-clock, reduced instruction only



14.8 MODEL QUESTIONS

Q.1: What do you mean by CISC architectures? What are the reasons behind the development of CISC architectures?

Q.2: What are the basic characteristics of CISC architectures?

Q.3: What are the disadvantages of CISC architectures?

Q.4: What do you mean by RISC architectures?

Q.5: What are the basic characteristics of RISC architectures?

Q.6: What are the advantages of RISC over CISC?

Q.7: What are the differences between RISC and CISC?

*** ***** ***

UNIT 15: PARALLEL ARCHITECTURES

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Flynn's Classification of Computer Architecture

- 15.3.1 SISD
- 15.3.2 SIMD
- 15.3.3 MISD
- 15.3.4 MIMD

15.4 Pipelining

- 15.4.1 Arithmetic Pipeline

15.5 Type of Processors

- 15.5.1 Scalar Processor
- 15.5.2 Superscalar Processor and Pipelined Processor
- 15.5.3 Array Processor

15.6 Vector Processor

- 15.6.1 Vector-Register Processor
- 15.6.2 Memory-Memory Vector Processor
- 15.6.3 Vector and Scalar Register
- 15.6.4 Chaining
- 15.6.5 Scatter Gather Operations

15.7 Vector Computing

15.8 Let Us Sum Up

15.9 Further Reading

15.10 Answers to Check Your Progress

15.11 Model Question

15.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about classification of computer architecture based on the notion of instruction and data stream

- describe pipelining technique used in systems to improve the overall performance of the system
- learn about different types of processors and their functioning
- define vector processor and describe its types and instructions
- learn about process of vector computation.

15.2 INTRODUCTION

Previous unit discusses the main functions, architectures and difference between CISC and RISC. This unit reviews architectural evolution of computers starting from uniprocessor systems to multiprocessor system though Flynn's classification of computer architecture. Various parallel computer architectures like pipelined processors, array processors and vector processors were discussed. Finally the unit discusses various techniques in vector processing like chaining, scatter, and gather instruction which aid in improving the performance of the system.

15.3 FLYNN'S CLASSIFICATION OF COMPUTER ARCHITECTURE

With the increase of multiple processing units where a job is broken into multiple segments and each segment is executed concurrently resulted in classification system. Flynn's classification or Flynn's taxonomy of computer architectures is proposed by Michael J. Flynn in the year 1972. The classification is based on the notion of instruction stream and data stream. A stream refers to sequence of instruction or data operated by the computer system. The flow of instruction from memory to processor is called instruction stream and the flow of data between processor and memory is called data stream.

The figure 15.1 shows four categories in which Flynn has classified computer architecture based on instruction and data stream. A conventional uniprocessor system is called SISD (Single instruction stream Single data stream) computers. A vector /array of processors is called SIMD (Single instruction stream Multiple data stream) computers. In MISD (Multiple instruction stream Single data stream) computers, different instructions work

on the same data. Finally in MIMD (Multiple instruction stream Multiple data stream) computers multiple processors each working on different data increases the overall performance of the system.

Data Stream	Instruction Stream: Single	Instruction Stream: Multiple
Multiple	SISD Traditional Von Neumann Single Processor Architecture	MISD Systolic Arrays
Single	SIMD Vector processor/Array processor fine grained parallel computer	MIMD Multiprocessor Systems, Multi Core systems

Figure 15.1: Flynn's Classification of Computer Architecture

15.3.1 SISD

A Single instruction stream single data stream (SISD) system as shown in figure 15.2 is a uniprocessor system. Such systems works on a single instruction and single data stream at a time. Any instructions in SISD systems are processed in a sequential order and are therefore known as sequential computers. Conventional systems were having SISD architecture. Processing in SISD systems involve storing both instructions and data in primary memory. Processing speed of such systems depends on internal data transfer rate. However performance of such systems can be improved with the help of multiple functional units or pipelining. Example of SISD systems includes CDC 6600, IBM PC.

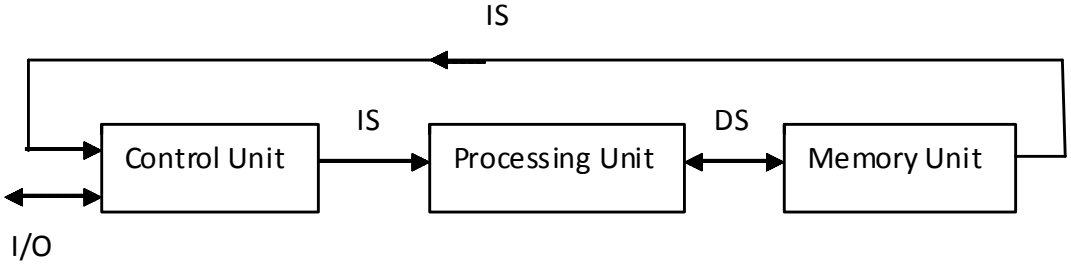


Figure 15.2: SISD Architecture

15.3.2 SIMD

A single instruction stream multiple data stream (SIMD) system as shown in figure 15.3 are multiprocessor systems capable of working on different data streams through a single instruction stream. SIMD systems are used in scientific computation involving vector operations. They are also known as vector processors or array processors. There are n number of processing units each having its own memory and stream of data. All the n processing units receive the same instruction from the control unit. Example of SISD systems includes CRAY vector computers.

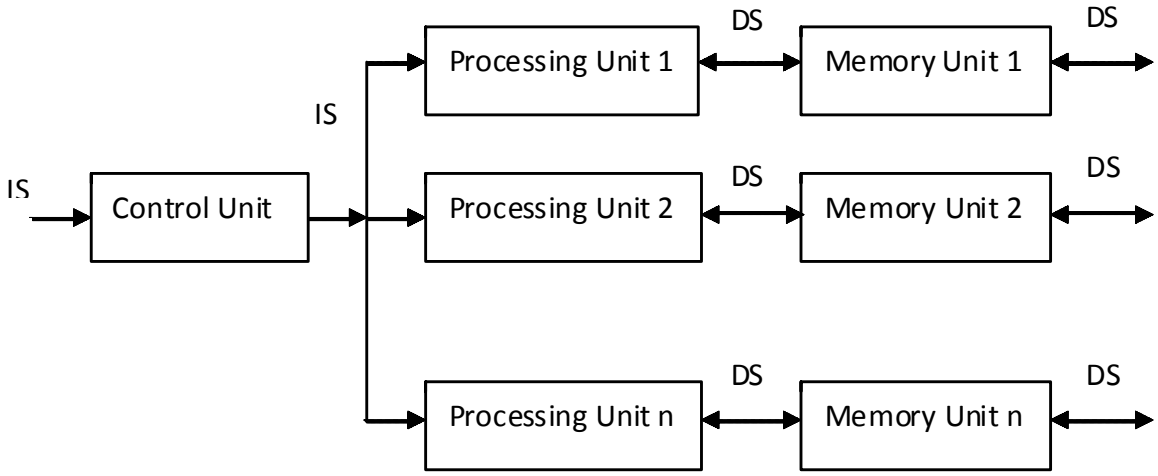


Figure 15.3: SIMD Architecture

15.3.3 MISD

A multiple instruction stream single data stream (MISD) system as shown in figure 15.4 is a multiprocessor system which executes different instructions on different processing unit but data set is same for all the instructions. MISD systems are not practical in the majority of the application and therefore are not available commercially. One such example of MISD system is *systolic array.*

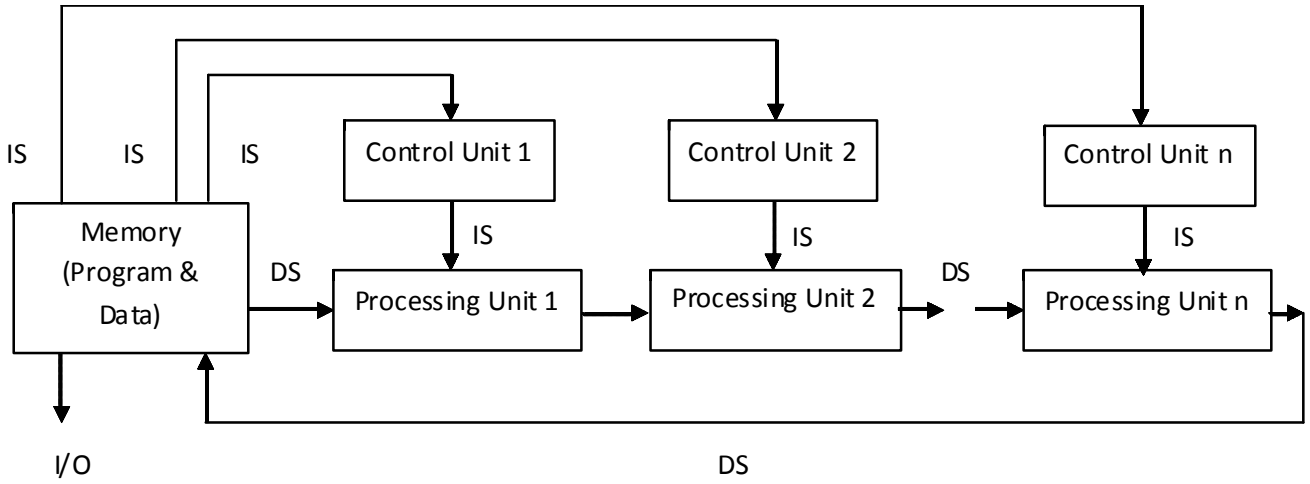


Figure 15.4: SIMD Architecture

15.3.4 MIMD

A multiple instruction stream multiple data stream (MIMD) system is a multiprocessor system capable of executing different sets of instructions each working on a different set of data simultaneously. The figure 15.5 shows MIMD architecture. Multiple SIMD systems connected together can be viewed as a MIMD system. MIMD systems can be classified into shared-memory MIMD and distributed-memory MIMD based on processing unit-main memory connections.

The shared memory MIMD system also known as tightly coupled multiprocessor system, are the one where all the processing units are connected to a single shared memory. Any form of communication between processing unit takes place with the help of shared memory. Changes done to data in shared memory by one processing unit is visible to all other processing units. In distributed memory MIMD systems or loosely coupled multiprocessor systems all processing units have their own local memory. The communication between processing units takes place through the interconnection.

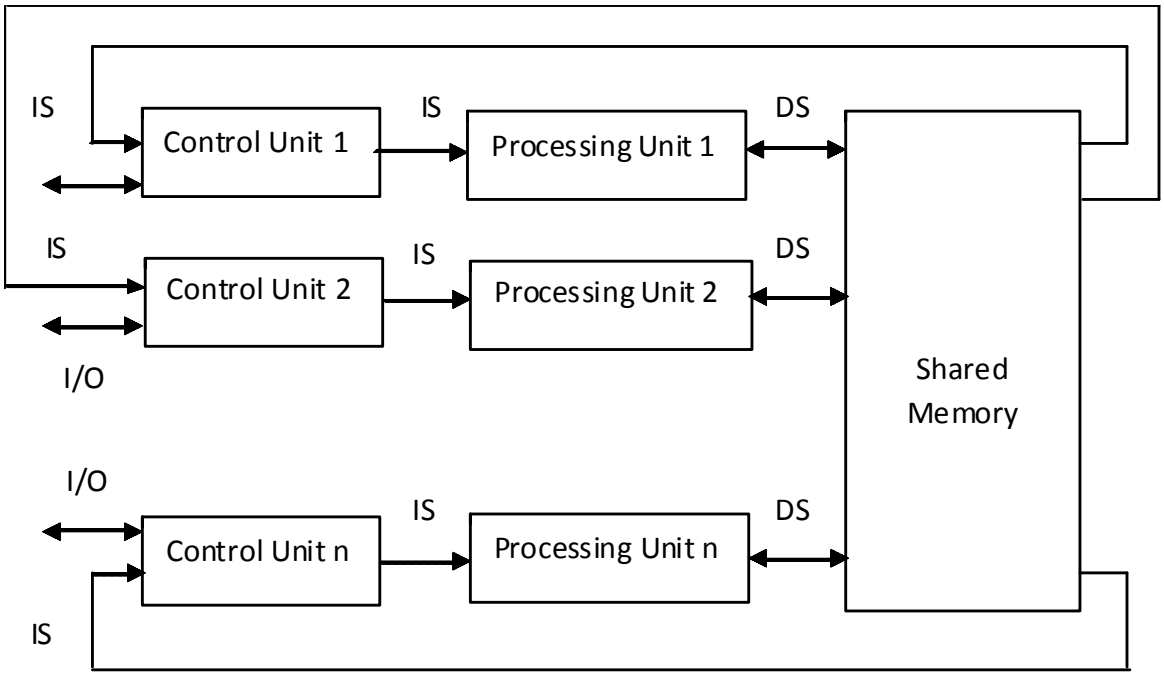


Figure 15.5: MIMD Architecture

15.4 PIPELINING

The technique of executing multiple instructions in an overlapped manner is called pipelining. In pipelining an instruction passes through several stages before completing execution. Each stage is connected to the next stage and so on to form a pipeline structure. An instruction enters from one end of the pipeline and goes through several stages before exiting from other end of the pipeline.

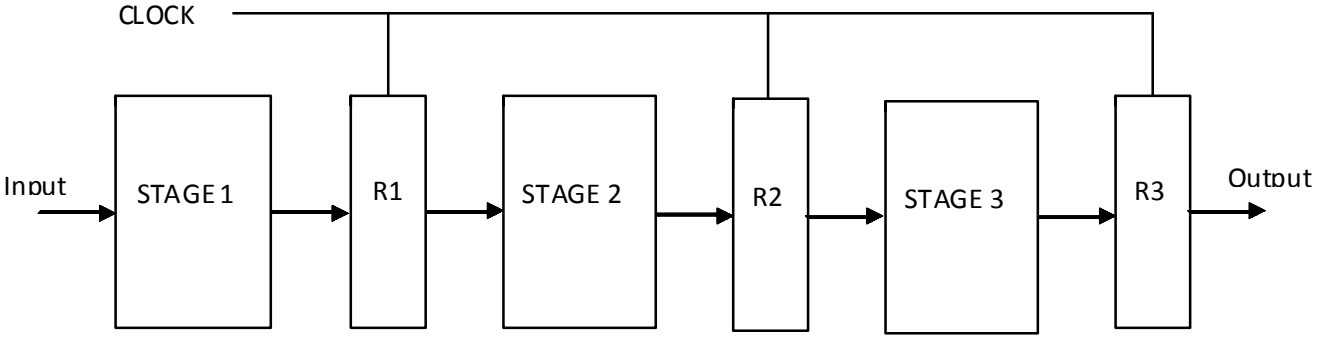


Figure 15.6: Pipeline Structure

Figure 15.6 shows a pipeline system with 3 stages and 3 registers. Each stage consists of an input register and a combinational circuit. The

register is used to hold the data and the combinational circuit to operate on the data. The output of each stage is stored in the register and with each clock pulse the output is move to the next stage. Overall the throughput of the instruction is improved by pipelining.

15.4.1 Arithmetic Pipeline

Arithmetic pipelines are generally found in the majority of the computers. They are used for operations involving floating point and fixed point numbers like addition, subtraction multiplication etc. For example figure 15.7 shows an arithmetic pipeline for floating point addition and subtraction.

The floating point addition and subtraction are completed in four stages:

- Compare exponents.
- Choose exponent and align mantissa
- Add or subtract mantissas
- Adjust exponent and normalize the result.

Registers are used for storing the intermediate results between stages.

In the first stage the exponents x and y are subtracted and the difference is used to determine how many times the mantissa of the smaller exponent must be shifted. In the second stage the larger exponent is chosen as the final result exponent and the mantissa is aligned. In the third stage the two aligned mantissa are added or subtracted. Finally in the fourth stage the result is normalized i.e. in case of overflow the result is shifted right and exponent incremented by number of right shifts, in case of underflow result is shifted left and exponent is decremented by number of left shifts.

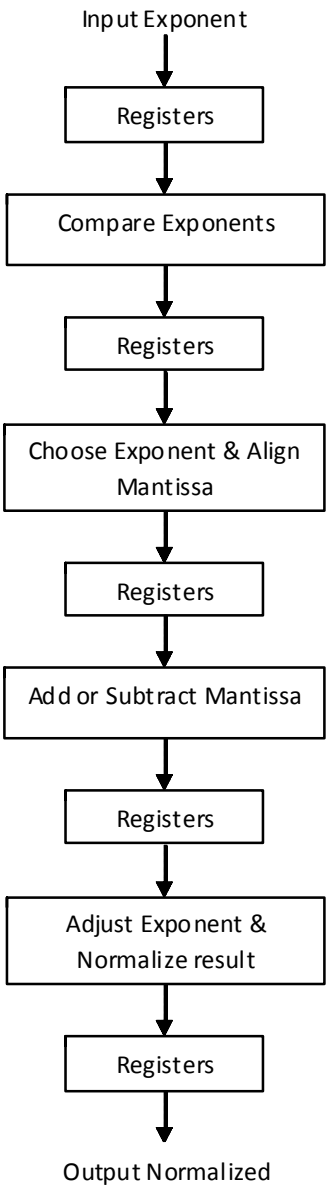


Figure 15.7: Arithmetic Pipeline for Floating Point Addition and Subtraction

The example below gives a clear idea of the process:

$P = 0.1235 \times 10^5$

$Q = 0.432 \times 10^3$

Stage 1: Exponent subtraction: 5 – 3 = 2

Stage 2: $P = 0.1235 \times 10^5$

$Q = 0.00432 \times 10^5$

Stage 3: $R = 0.1235 + 0.00432 \times 10^5$

Stage 4: $Z = 0.12782 \times 10^5$

15.5 TYPE OF PROCESSORS

15.5.1 Scalar Processor

A scalar processor also known as SISD (Single Instruction Stream Single Data Stream) can process a single data item at a time. Scalar processor can process either integer or floating point operands. The simplest scalar processor uses floating point unit to process integer operands. However scalar processor may have separate integer and floating point units for handling integer and floating point operands. AMD 2900, Motorola 68040, Intel 386, Intel 486, M88100 are some examples of scalar processor.

15.5.2 Superscalar Processor and Pipelined Processor

Superscalar Processor: Superscalar processors are found in parallel computing architecture to improve the performance of the system by executing multiple instructions in parallel. A superscalar processor manages multiple instruction pipelines to execute multiple instructions concurrently in a clock cycle. The performance of superscalar processor is highly dependent on the instruction dependency quotient. If the instructions to be executed are independent, then high performance is achieved.

The figure 15.8 shows a superscalar pipeline of degree 2 (i.e., Two instructions can be executed in parallel). There are five stages in the pipeline namely fetch, decode, operand fetch, execute and write. It can be observed from the figure that the superscalar pipeline has two units each of fetch, decode, operand fetch, execute and write, therefore two instructions can be simultaneously executed. In the first clock cycle two instructions are fetched, in the second clock cycle next two instructions are fetched and the process continues. Pentium, DEC Alpha, PowerPC are some of the example of superscalar processor computers.

	1	2	3	4	5	6	7	8
Instruction 1	F	D	OF	E	W			
Instruction 2	F	D	OF	E	W			
Instruction 3		F	D	OF	E	W		
Instruction 4		F	D	OF	E	W		
Instruction 5			F	D	OF	E	W	
Instruction 6			F	D	OF	E	W	
Instruction 7				F	D	OF	E	W
Instruction 8				F	D	OF	E	W

Figure 15.8: A superscalar Pipeline of Degree 2

Pipelined Processor: There are four types of pipelined processors depending upon the following criterions. It is assumed that all the pipelined processors are of k stages.

- Machine Pipeline Cycle (MPC): Time taken by each stage to process an instruction.
- Instruction Issue Rate (ISR): Number of instruction that can be issued simultaneously.
- Instruction Issue Latency (ISL): Time interval between issue of two instructions.
- Instruction Level Parallelism (ILP): Number of instructions that can be executed simultaneously in the pipeline.

Machine Type	Scalar Pipeline	Superscalar Pipeline	Superpipeline	Superpipeline Superscalar
MPC	1	1	1/n	1/n
ISR	1	m	1	m
ISL	1	1	1/n	1/n
ILP	1	m	n	mn

Figure 15.9: Parameters of Pipelined Processor

15.5.3 Array Processor

An array processor or vector processor are multiprocessor systems that performs computation on large arrays of data. A scalar

processor operates on one data item at a time whereas an array processor operates on multiple sets of data item at a time. The instruction however remains same for all the processors executing different data sets.

As for example if a system wants to execute hundred instructions each associated with thousand data items then in order to perform a task a scalar processor executes one by one all the hundred instruction. Whereas an array processors take the first instruction (from hundred) and executes all thousand data items and then progress to the next instruction.

There are mainly two types of array processors:

- Attached Array Processors
- SIMD Array Processors

Attached Array Processors: An attached array processor shown in figure 15.10 is used to improve the overall performance of a general purpose computer. It is designed as a peripheral device for supporting a general purpose computer in solving complex applications by means of vector processing with multiple functional units. It consists of more than one functional unit like floating point adders, multipliers etc. Such processor can also be programmed by the user to support different types of complex applications.

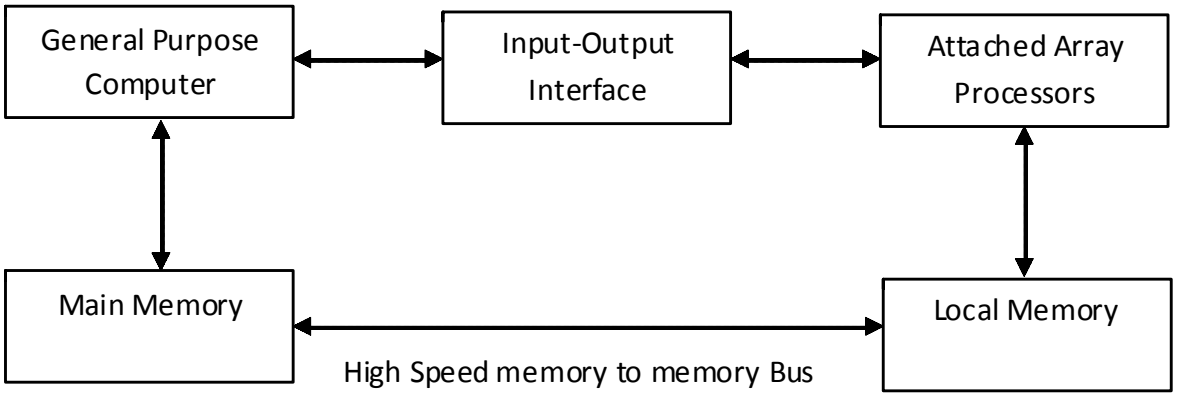


Figure 15.10: Attached Array Processor

SIMD Array Processors: Refer to section 15.3.2

15.6 VECTOR PROCESSOR

Vector processors are found mainly in supercomputers combining pipelining and interleaved memory unit. It is used mainly in scientific and multimedia applications involving processing of huge volume of data. It is capable of processing entire vector in single instruction. The operands in the instructions are vectors instead of a single element. One of the advantages of vector processors is less number of fetch and decode instructions.

Vector processor uses many optimization schemes to improve performance of the system such as use of memory banks to reduce load/ store latency, use of strip mining technique to adjust the size mismatch between vector operands and vector registers, vector chaining to resolve data dependency between vector instructions etc.

Advantages of Vector processing:

- Programs are smaller in size as the number of instructions is quite less.
- As each data in registers is actually used by the vector processor therefore wastage in memory access is significantly less compared to cache memory.
- Requirement of power is limited to only functional unit and register buses during vector operation.

Based on how operands are fetched in vector processors is categorized into two types:

- Vector-Register Processor
- Memory-Memory Vector Processor

15.6.1 Vector-Register Processor

It requires that all the operations performed in the vector processor use the source operands and destination operands as vector registers. However there is a small disadvantage initially that vector data in memory is must be divided into fixed length segments so that can be placed in vector register. But once the pipelining starts this disadvantage is nullified.

15.6.2 Memory-Memory Vector Processor

Such processors allow source operand and destination operand to be routed directly to the arithmetic logic unit (ALU). Once the processing is completed in the ALU, the result is routed back to memory. However due to memory latency the time between initializing the first instruction and the getting the first output from the pipeline is quite large.

15.6.3 Vector and Scalar Register

A vector consists of fixed length set of data words and a set of vector registers to hold the data. These are specialized registers used in vector processing to hold vector data (multiple data) in a single register. A vector registers is loaded with input vector data from memory and are like FIFO queues that can hold multiple data items at a time. In a vector pipeline, vector data elements are sent one at a time from vector register and the output of the pipeline is sent to the vector register one at a time. There are several vector registers in a vector processor and also have specific instruction to load, perform operation and store register contents to memory. In supercomputers like CRAY, vector registers are of fixed length 64 bits. In Fujitsu VP2000 vector registers are reconfigurable to match the length of the vector operands.

Scalar registers unlike vector register is used to hold single/ scalar data item. There are separate set of instructions that work on scalar register.

15.6.4 Chaining

The concept of chaining or vector chaining in vector processor involves forwarding the result of a pipeline vector operation to the next pipeline without storing the result in vector register on completion. The vector chaining technique was first used in Cray 1 supercomputer in 1976. Chaining technique enables in reducing

the startup time i.e. time to initialize the pipeline with first instruction and obtaining the first result compared to memory–memory systems. It is extension to internal data forwarding technique bypassing paths between functional units. The concept of internal data forwarding is common in microprocessors but chaining extends the concept with vectors of larger length. From a compilers point of view chaining allows vectorization of dependent instructions.

15.6.5 Scatter Gather Operations

In vector processing there are two instructions Scatter and Gather which are useful in handling sparse matrices or sparse vectors. In gather instruction non zero elements of sparse matrix are stored in vector register using indices which are itself indexed. The figure 15.11 shows a clear picture of Gather Operation.

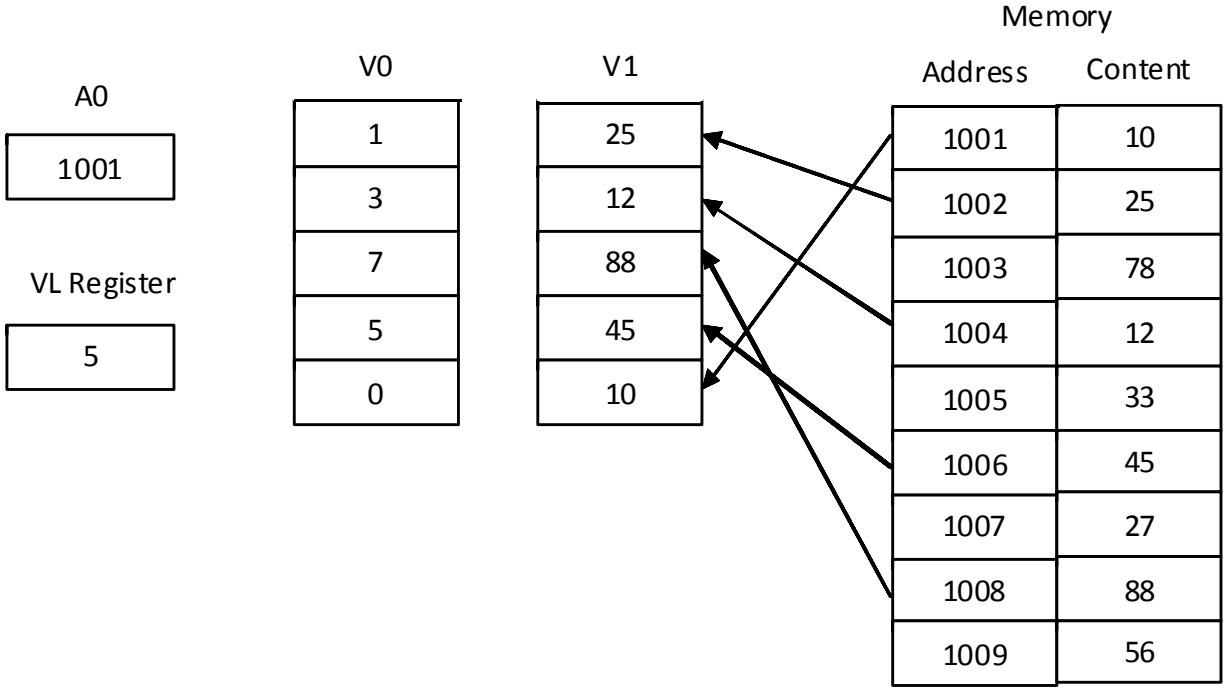


Figure 15.11: Gather Operation

Here the gather instruction the moves non sequential memory contents 25, 12, 88, 45, 10 to register V1 of length (length=5) specified by register VL. The contents are moved based on the index specified in the V0 register. The register A0 is used as the base register.

The scatter instruction on the other hand stores the elements of the vector register to memory specified by the index available in another vector register. It does the opposite mapping operation as to gather. The figure 15.12 shows a scatter operation. Here contents of vector register V1, 59, 62, 08, 29, 17 are stored in non sequential memory location specified by the index value in register V0. The register A0 is used as the base register.

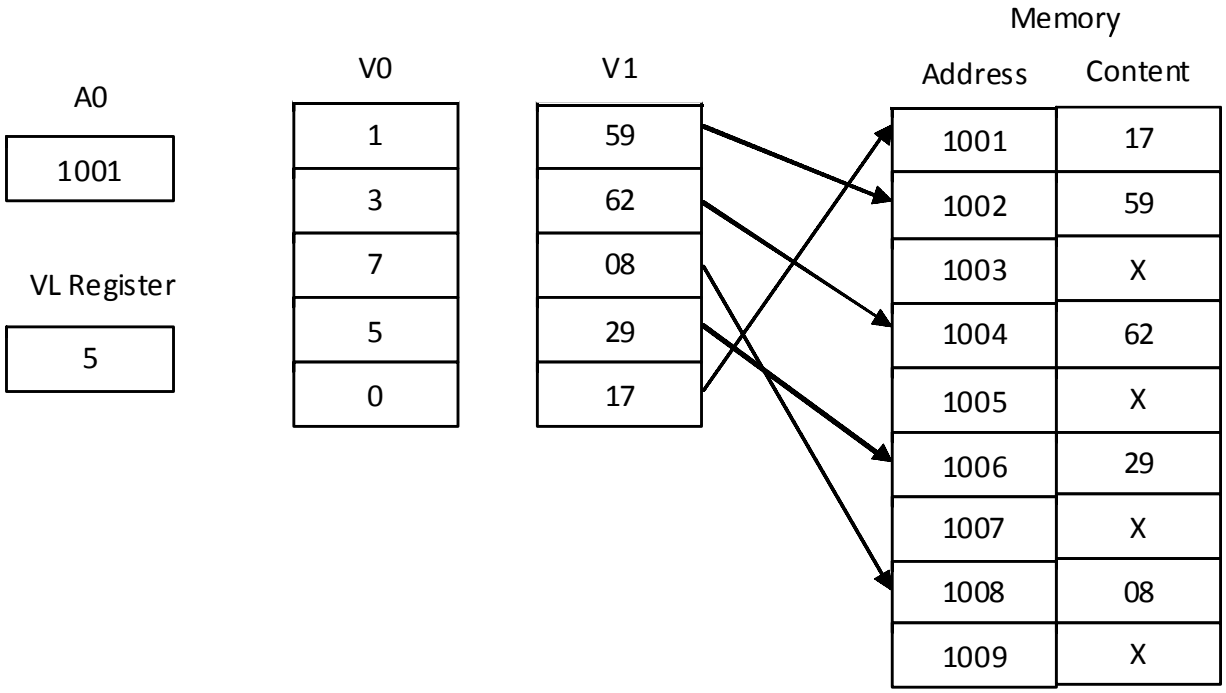


Figure 15.12: Scatter Operation

15.7 VECTOR COMPUTING

The main task in design of a supercomputer is to perform arithmetic operations on vector of floating point numbers. In case of general purpose computer there will be n number of iterations required for a vector of length n to perform any arithmetic operation. Considering multiplication of two vectors A, B of size N X N and storing the result back to vector C, a vector computation will help in parallelizing the entire operation.

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \, X \, b_{k,j}$$

Where $a_{i,j}$, $b_{i,j}$, $c_{i,j}$ are respectively elements of the vector A, B, C.

In scalar processing the code can be written as:

```
for (i = 1; i < N; i++)
    {
        for (j = 1; j < N; j++)
        {
            c[i][j] = 0;
            for (k = 1; k < N; k++)
            c[i][j] = c[i][j] + a[i][k]*b[k][j];
        }
    }
```

In vector processing the code can be written as:

```
for (i = 1; i < N; i++)
{
    c[i][j] = 0;  (for all j=1,N)
    for (k = 1; k < N; k++)
    c[i][j] = c[i][j] + a[i][k]*b[k][j]; (for all j=1,N)
}
```

The vector computation specifies that the statement (for all j=1,N) will be carried out as a single operation on all indices of j. And all the elements will be computed in parallel for each index of i. The code requires N^2 vector multiplication compared to N^3 scalar multiplications.



CHECK YOUR PROGRESS

Q.1: CRAY systems are an example of .....

a) SISD b) SIMD

c) MISD d) MIMD

Q.2: In an arithmetic pipeline addition of two numbers 0.31511×10^4 and 0.01972×10^3 will result in .....

a) 3.17082×10^4 b) 3.17082×10^3

c) 3.17082×10^4 d) 0.317082×10^4

Q.3: Registers are used in a pipeline to .....

a) Read data b) Write data

c) Hold data d) Process data

Q.4: Motorola 68040 is an example of .....

a) Scalar processor b) Superscalar processor

c) Superpipeline processor d) Pipelined processor

Q.5: A superscalar pipeline (5 stages) of degree 3 will need ..... cycles to complete 9 instructions.

a) 6 b) 7 c) 8 d) 9

Q.6: Instruction Issue Latency (ISL) in a pipelined processor means–

a) Time interval between issuing of first and last instruction.

b) Time interval between completion of first and second instruction.

c) Time taken to complete execution of first instruction.

d) Time interval between issuing of two instructions.

Q.7: Instruction Level Parallelism in a pipelined processor means–

a) Number of instructions in the pipeline.

b) Number of instructions that can be completed simultaneously in the pipeline.

c) Number of instructions that can be executed simultaneously in the pipeline.

d) None of the above

Q.8: Array processors are ..... systems

a) SISD b) MISD c) SIMD d) MIMD

Q.9: Attached array processor is a .....

a) Pipelined processor b) Powerful processor

c) Peripheral device d) All of the above

Q.10: Advantages of Vector processing is .....

a) Very fast processing

b) Parallel processing

c) Number of instructions is quite less

d) All of the above

Q.11: Vector register processor uses source and destination operands as–

a) Scalar b) Vector

c) Scalar and vector d) Any type

Q.12: The concept of internal data forwarding in vector processing is known as–

a) Chaining b) Scatter c) Gather d) Pipeline

Q.13: To handle sparse matrix ..... instruction is used.

a) Scatter b) Gather

c) Both (a) and (b) d) Sparse



15.8 LET US SUM UP

- Flynn's taxonomy of computer architectures is proposed by Michael J. Flynn in the year 1972. The classification is based on the notion of instruction and data stream.
- A Single instruction stream single data stream (SISD) system is a uniprocessor system.
- A single instruction stream multiple data stream (SIMD) system are multiprocessor systems capable of working on different data streams through a single instruction stream.
- A multiple instruction stream single data stream (MISD) system is a multiprocessor system which executes different instructions on different processing unit but data set is same for all the instructions.
- A multiple instruction stream multiple data stream (MIMD) system is a multiprocessor system capable of executing different sets of instructions each working on a different set of data simultaneously.
- The technique of executing multiple instructions in an overlapped manner is called pipelining.
- Arithmetic pipelines are used for operations involving floating point and fixed point numbers like addition, subtraction multiplication etc.

- Superscalar processors are found in parallel computing architecture to improve the performance of the system by executing multiple instructions in parallel.
- There are four types of pipelined processors namely scalar, superscalar, superpipeline and superscalar superpipeline.
- An array processor or vector processor are multiprocessor systems that performs computation on large arrays of data.
- An attached array processor is designed as a peripheral device for supporting a general purpose computer in solving complex applications by means of vector processing with multiple functional units.
- Vector processors are found mainly in supercomputers combining pipelining and interleaved memory unit.
- The concept of chaining or vector chaining in vector processor involves forwarding the result of a pipeline vector operation to the next pipeline without storing the result in vector register on completion.
- In vector processing there are two instructions Scatter and Gather which are useful in handling sparse matrices or sparse vectors.



15.9 FURTHER READING

1) Hamacher, Carl. (5th Edition). *Computer Organization*. McGraw Hill.
2) Mano, M. Morris; Mall, Rajib. (3rd Edition). *Computer System Architecture*. Pearson
3) Stallings, William. (2000). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.



15.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (b) **Ans. to Q. No. 2:** (b)

Ans. to Q. No. 3: (c) **Ans. to Q. No. 4:** (a)

Ans. to Q. No. 5: (b) **Ans. to Q. No. 6:** (d)

Ans. to Q. No. 7: (c) **Ans. to Q. No. 8:** (c)

Ans. to Q. No. 9: (c) **Ans. to Q. No. 10:** (d)

Ans. to Q. No. 11: (c) **Ans. to Q. No. 12:** (a)

Ans. to Q. No. 13: (c)



15.11 MODEL QUESTION

Q.1: Discuss Flynn's classification of computer architecture.

Q.2: Differentiate between shared memory and distributed memory MIMD systems.

Q.3: Explain how pipelining can increase the performance of a system compared to a single processor system.

Q.4: Consider the two floating point numbers given below; discuss in detail the steps to be taken by an arithmetic pipeline to add these numbers.

$P = 0.3521 \times 10^4$

$Q = 0.01920 \times 10^2$

Q.5: Differentiate between superscalar processor and superpipeline processor.

Q.6: Briefly describe the parameters on which different pipelined processors are measured in terms of their performance.

Q.7: How an attached array processor helps in improving the overall performance of a general purpose computer?

Q.8: Differentiate between vector register processor and memory-memory vector processor.

Q.9: The concept of internal data forwarding in case of vector processing is chaining. Explain?

Q.10: Explain how sparse matrix is handled by vector processor?

Q.11: Time complexity of a program can be reduced by vector computing. How?

*** ***** ***



REFERENCES

1) Chaudhuri, P. Pal. (2nd Edition, 2003). *Compter Organization and Design*. PHI Learning Pvt. Ltd.
2) Floyd, Thomas L. *Digital Fundamentals*. Pearson Education.
3) Gill, D. N. S. & Dixit, J. B. (2008). *Digital Design and Computer Organisation*. Firewall Media.
4) Godse, A. P., Godse, D. A. *Computer Architecture*. Technical Publications Pune.
5) Leach, D. P. & Malvino, A. P. (1994). *Digital Principles and Applications*. Glencoe/McGraw-Hill.
6) Hammacher, Carl. (5th Edition) (International Edition, 2002). *Computer Organization*. McGrawHill.
7) Hamacher, V. Carl, Vranesic, Zvonko G., Zaky, Safwat G. *Computer Organization*. McGraw-Hill International Edition.
8) Lee, S. C. (1976). Digital Circuits and Logic Design.
9) Mano, M. M. (2006). *Computer Systems Architecture*. Pearson Education.
10) Mano, M. (1979). *Digital Logic, Computer Design*. Englewood Cliffs Prentice-Hall.
11) Mano, M. Morris; Mall, Rajib. (3rd Edition). *Computer System Architecture*. Pearson.
12) Null, L. & Lobur, J. (2014). *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Publishers.
13) Rajaraman, V. (4th Edition, 2008). *Fundamentals of Computers*. PHI Publication.
14) Stallings, William. (2004). *Computer Organization and Architecture Designing for Performance*. Pearson Education India.
15) Talukdar, P. H. *Digital Techniques*. N. L. Publications.
16) Zaky, S. G., Hamacher, V. C. & Vranesic, Z. G. (1996). *Computer Organization*. New York: McGraw-Hill.

Name of the Paper: .....
Course Code: .....

Centre for Internal Quality Assurance (CIQA)
Krishna Kanta Handiqui State Open University
City Office: Housefed Complex, Guwahati-781006

Learner's Feedback on Course

Dear Learner,

Regarding the course as mentioned above, we would like to know your opinions and comments so as to improve the quality of self learning materials in future. Please respond to the following statements by ticking the number you feel most reflect your opinion. After completion of the additional comments, please detach the page and send/mail the same to us at the address given below.

The Director, Centre for Internal Quality Assurance, KKHandiqui State Open University
Housefed Complex, Dispur, Guwahati-781006
(E-mail id: ciqa@kkhsou.in)

1) Approximately how many hours did you spend for studying the units in the course? ☐☐☐

2) Please give your opinions (by ✓ mark) to the following items based on your reading of the block:

Sl. No.	Statements	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
I)	The SLMs of the course fulfil the learning objectives set out in the beginning of each unit					
II)	The units of the SLM could satisfy my academic needs and expectations					
III)	The Unit writers have excellent knowledge about the course contents					
IV)	Language and contents of the units were presented at a level which I could readily understand					
V)	Ample opportunity for participation in the activities provided in the units					
VI)	Used enough Illustrations (Diagrams, tables etc.) for conceptual clarity					
VII)	Quality of content is engaging, relevant, and up-to-date					
VIII)	The self check questions are very helpful					
IX)	The Possible/model questions and the answers to check my progress have benefited me a lot					

Additional Comments: *(Please feel free to provide your open comments)*

1) Which aspects of the SLM, according to you, worked well?

   .....

   .....

2) What sort of changes/improvements do you feel KKHSOU could implement to improve the overall quality of the SLM?

   .....

   .....

Thank you for taking the time to complete this form.