UNIT I

1 Fundamentals of Computer Organization

Syllabus

Basic Organization of Computers, Classification Micro, Mini, Mainframe and Super Computer. System Bus and Interconnection, PCI, Computer Function, I-Cycle, Interrupt and Class of Interrupts.

Contents

1.1 Basic Organization of Computers

- A digital computer or simply computer in its simplest form is a fast electronic calculating machine that accepts digitized information from the user, processes it according to a sequence of instructions stored in the internal storage, and provides the processed information to the user.
- The sequence of instructions stored in the internal storage is called **computer program** and internal storage is called **computer memory**.
- The computer consists of five functionally independent units :
  - Input
  - Memory
  - Arithmetic and logic
  - Output and
  - Control units.
- The Fig. 1.1.1 (a) and (b) show these five functional units of a computer and its physical locations in the computer.

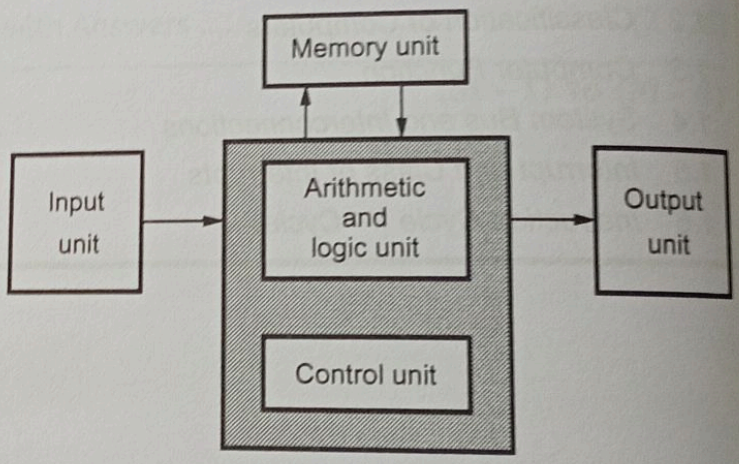


Fig. 1.1.1 (a)

- The input unit accepts the digital information from user with the help of input devices such as keyboard, mouse, microphone etc.
- The information received from the input unit is either stored in the memory for later use or immediately used by the arithmetic and logic unit to perform the desired operations.
- The program stored in the memory decides the processing steps and the processed output is sent to the user with the help of output devices or it is stored in the memory for later reference.
- All the above mentioned activities are co-ordinated and controlled by the control unit.
- The arithmetic and logic unit in conjunction with control unit is commonly called **Central Processing Unit (CPU)**.

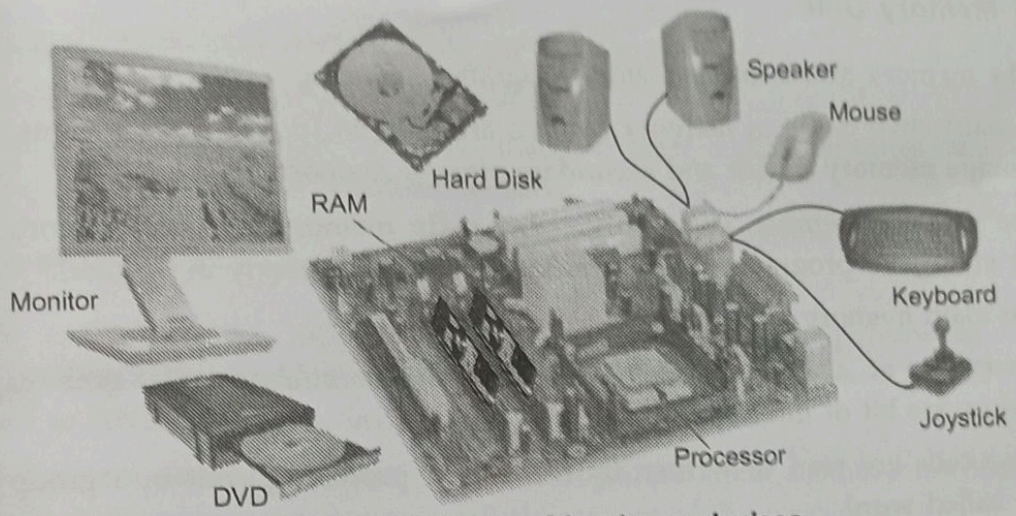
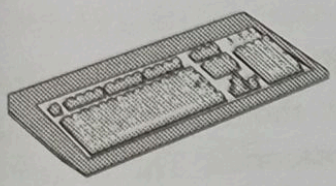


Fig. 1.1.1 (b) Types of hardware devices

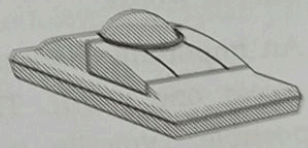
1.1.1 Input Unit

- A computer accepts a digitally coded information through input unit using input devices.
- The most commonly used input devices are keyboard and mouse.
- The keyboard is used for entering text and numeric information.
- Mouse is used to position the screen cursor and thereby enter the information by selecting option.
- Apart from keyboard and mouse there are many other input devices are available, which include joysticks, trackball, spaceball, digitizers and scanners.

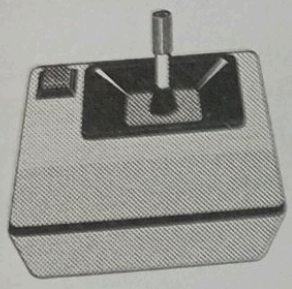


(a) Keyboard

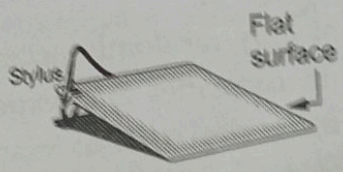
(b) Mouse



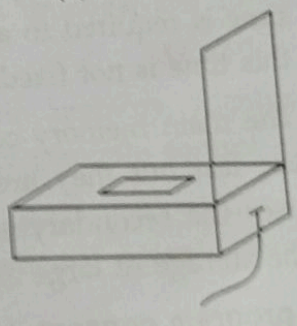
(c) Track ball



(d) Joystick



(e) Tablet or digitizer



(f) Scanner

Fig. 1.1.2 Input devices

1.1.2 Memory Unit

- The memory unit is used to store programs and data.
- Usually, two types of memory devices are used to form a memory unit : **Primary storage memory** device and **secondary storage memory** device.
- The primary memory, commonly called **main memory** is a fast memory used for the storage of programs and active data (the data currently in process).
- The main memory is a semiconductor memory.
- It consists of a large number of semiconductor storage cells, each capable of storing one bit of information.
- These cells are read or written by the central processing unit in a group of fixed size called **word**.
- The main memory is organized such that the contents of one word, containing n bits, can be stored or retrieved in one write or read operation, respectively.
- To access data from a particular word from main memory each word in the main memory has a **distinct address**. This allows to access any word from the main memory by specifying corresponding address.
- The number of bits in each word is referred to as the **word length** of the computer. Typically, the word length varies from 8 to 64 bits.
- The number of such words in the main memory decides the **size of memory** or **capacity of the memory**.
- The size of computer main memory varies from few million words to tens of million words.
- An important characteristics of a memory is an **access time** (the time required to access one word). The access time for main memory should be as small as possible. Typically, it is of the order of 10 to 100 nanoseconds. The access time depends on the type of memory. In Randomly Accessed Memories (RAMs), fixed time is required to access any word in the memory. In sequential access memories this time is not fixed.
- The main memory consists of only randomly accessed memories. These memories are fast but they are small in capacities and expensive. Therefore, the computer uses the secondary storage memories such as magnetic tapes, magnetic disks for the storage of large amount of data.

Stored program concept

- Today's computer are built on two key principles
  1. Instructions are represented as numbers.
  2. Programs can be stored in memory to be read or written just like numbers.

- These principles lead to the stored-program concept.
- According to stored-program concept, memory can contain the program (source code), the corresponding compiled machine code, editor program and even the compiler that generated the machine code.

1.1.3 Arithmetic and Logic Unit

- The arithmetic and logic unit (ALU) is responsible for performing arithmetic operations such as add, subtract, division and multiplication and logical operations such as ANDing, ORing, Inverting etc.
- To perform these operations, operands from the main memory are brought into the high speed storage elements called **registers** of the processor.
- Each register can store one word of data and they are used to store frequently used operands.
- After performing operation, the result is either stored in the register or memory location.

1.1.4 Output Unit

- The output unit sends the processed results to the user using output devices such as video monitor, printer, plotter, etc.
- The video monitors display the output on the CRT screen whereas printers and plotters give the hard-copy output.
- Printers are classified according to their printing methodology : **Impact printers** and **non-impact printers**.

1.1.5 Control Unit

- The control unit co-ordinates and controls the activities amongst the functional units.
- Control unit fetches the instructions stored in the main memory, identify the operations and the devices involved in it and accordingly generate control signals to execute the desired operations.
- It uses control signals or timing signals to determine when a given action is to take place.
- It controls input and output operations, data transfers between the processor, memory and input/output devices using timing signals.

- The control and the arithmetic and logic units of a computer are usually many times faster than other devices connected to a computer system. This enables them to control a number of external input/output devices.

Review Questions

1. *Explain different functional units of a digital computer.*
2. *What is a stored program concept ? Explain the functional units of a stored program digital computer, along with a block diagram.*

1.2 Classification of Computers

- According to size, cost computational power and application computers are classified as :
  - Microcomputers
  - Minicomputers
  - Desktop computers
  - Personal computers
  - Portable notebook computers
  - Workstations
  - Mainframes or enterprise systems
  - Servers
  - Super computers.

Microcomputers :

- As the name implies micro-computers are smaller computers.
- They contain only one central processing unit.
- One distinguishing feature of a microcomputer is that the CPU is usually a single integrated circuit called a **microprocessor**.
- Microcomputer is the integration of microprocessor and supporting peripherals (memory and I/O devices).
- The word length depends on the microprocessor used and is in the range of 8 bits to 32 bits.
- These type of computers are used for small industrial control, process control and where storage and speed requirements are moderate.

Minicomputers :

- Minicomputers are the scaled up version of the microcomputers with the moderate speed and storage capacity.
- These are designed to process smaller data words, typically 32-bit words.
- This type of computers are used for scientific calculations, research, data processing application and many other.

Desktop Computers :

- The desktop computer are the computers which are usually found on a home or office desk.
- They consist of processing unit, storage unit, visual display and audio as output units and keyboard and mouse as input units.
- Usually storage unit of such computer consists of hard disks, CD-ROMs and diskettes.

Personal Computers :

- The personal computers are the most common form of desktop computers.
- They found wide use in homes, schools and business offices.

Portable Notebook Computers :

- Portable notebook computers are the compact version of personal computers.
- The laptop computers are the good example of portable notebook computer.

Workstations :

- Workstations have higher computation power than personal computers.
- They have high resolution graphics terminals and improved input/output capabilities.
- Workstations are used in engineering applications and in interactive graphics applications.

Mainframes or Enterprise Systems :

- Mainframe computers are implemented using two or more Central Processing Units (CPU).
- These are designed to work at very high speeds with large data word lengths, typically 64 bits or greater.
- The data storage capacity of these computers is very high.
- This type of computers are used for complex scientific calculations, large data processing applications, Military defense control and for complex graphics applications (e.g. : For creating walkthroughs with the help of animation software).

Servers :

- These computers have large storage unit and faster communication links.
- The large storage unit allows to store sizable database and fast communication links allow faster communication of data blocks with computers connected in the network.
- These computers serve major role in Internet communication.

Supercomputers :

- These computers are basically multiprocessor computers used for the large-scale numerical calculations required in applications such as weather forecasting, robotic engineering, aircraft design and simulation.

Review Question

1. *Explain the various types of computers and their applications.*

1.3 Computer Function

- The basic function of computer is to execute program, sequence of instructions.
- Instructions are stored in the computer memory.
- Instructions are executed to process data which is loaded into the computer memory through input unit.
- After processing the data, the result is either stored back into the computer memory for further reference or it is sent to the outside world through the output port.
- All functional units of the computer contribute to execute a program.
- The functions of different computer units are,
  - The input unit accepts data and instructions from the outside world to machine. It is operated by control unit.
  - The memory unit stores both, data and instructions.
  - The arithmetic-logic unit performs arithmetic and logical operations.
  - The control unit fetches and interprets the instructions in memory and causes them to be executed.
  - The output unit transmits final results and messages to the outside world.
- To perform execution of instruction, the processor contains a number of registers used for temporary storage of data and some special function registers, as shown in Fig. 1.3.1.

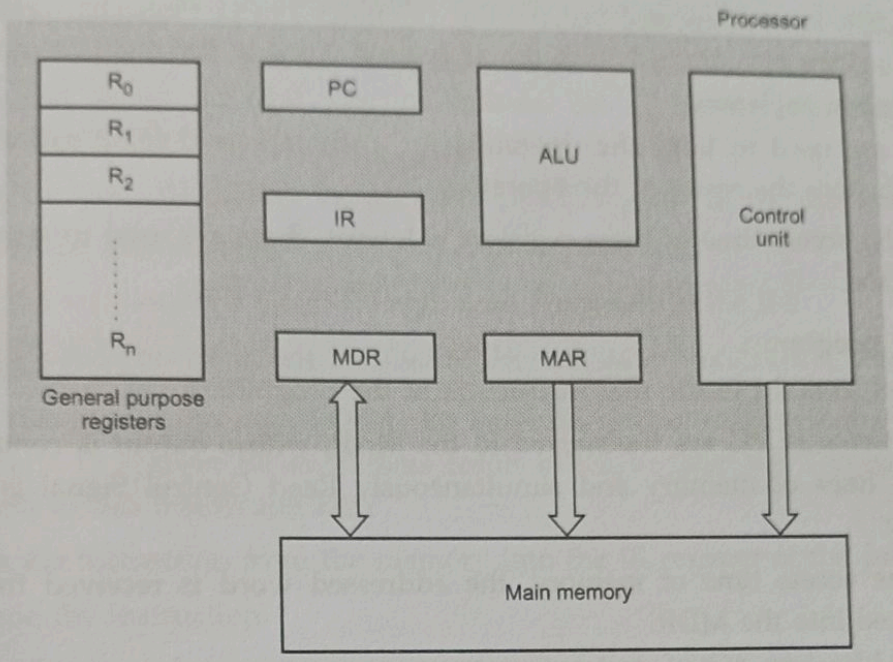


Fig. 1.3.1 Connections between the processor and the main memory

- The special function registers include Program Counter (PC), Instruction Register (IR), Memory Address Register (MAR) and Memory Data Register (MDR).

Program Counter (PC) :

- A program is a series of instructions stored in the memory. These instructions tell the CPU exactly how to get the desired result.
- It is important that these instructions must be executed in a proper order to get the correct result.
- The sequence of instruction execution is monitored by the program counter.
- It keeps track of which instruction is being executed and what the next instruction will be.

Instruction Register (IR) :

- It is used to hold the instruction that is currently being executed.
- The contents of IR are available to the control unit, which generate the timing signals that control the various processing elements involved in executing the instruction.

Memory Address Register (MAR) and Memory Data Register (MDR) :

- These registers are used to handle the data transfer between the main memory and the processor.
- The MAR holds the address of the main memory to or from which data is to be transferred.

- The MDR sometimes also called MBR (Memory Buffer Register) contains the data to be written into or read from the addressed word of the main memory.

General purpose registers :

- These are used to **hold the operands** for arithmetic and logic operations and/or used to store the result of the operation.
- Since the access time of these registers is lowest, these are used to **store frequently used data.**

Execution of program :

- PC is set to point to the first instruction of the program.
- The contents of PC are transferred to the MAR, whose output is connected to the address lines of memory and simultaneously Read Control Signal is sent to the memory.
- After the access time of memory, the addressed word is received from memory and stored into the MDR.
- The received instruction code, i.e. the contents of MDR are transferred to IR and now instruction is ready to be decoded and executed.
- If the instruction involved an operation to be performed by the ALU and operand is available in general-purpose registers the operation is performed by the ALU.
- If operand is not available in the general-purpose registers, it is reside in the memory. It has to be fetched by sending its address to the MAR and initiating a Read cycle.
- When the operand is read from the memory into the MDR, it is transferred from the MDR to the ALU.
- ALU then performs the desired operation and stores the result either in the general purpose registers or in the memory.
- If the result of operation is to be stored in the memory, then the result is sent to the MDR. The address of the memory location where the result is to be stored is sent to the MAR, and a write cycle is initiated.
- During the execution of current instruction, the contents of PC are incremented so that they points to the next instruction to be executed.
- As soon as current instruction is executed, next instruction is fetched and cycle is repeated.
- Like memory, computer also communicates with input/output devices. Thus, computer also has some instruction to handle I/O data transfer.

Example 1.3.1 *State the operations involved in the execution of ADD R1, R0 instruction.*

Solution : The instruction Add R1, R0 adds the operand in R1 register to the operand in R0 register and stores the sum into R0 register. Let us see the steps involved in the execution of this instruction.

1. Fetch the instruction from the memory into IR register of the processor.
2. Decode the instruction.
3. Add the contents of R1 and R0 and store the result in the R0.

Example 1.3.2 *State the operations involved in the execution of Add LOCA, R0.*

Solution : The instruction Add LOCA, R0 adds the operand at memory location LOCA to the operand in register R0 and stores result in the register R0. The steps involved in the execution of this instruction are :

1. Fetch the instruction from the memory into the IR register of the processor.
2. Decode the instruction.
3. Fetch the second operand from memory location LOCA and add the contents of LOCA and the contents of register R0.
4. Store the result in the R0.

Review Questions

1. *Explain different functional units of a computer. Mention the function of the processor registers i) PC ii) MAR iii) IR.*
2. *Draw and explain the connections between the processor and the main memory.*

1.4 System Bus and Interconnections

- The central processing unit, memory unit and I/O unit are the hardware components/modules of the computer. They work together with communicating each other and have paths for connecting the modules together. The collection of paths connecting the various modules is called the **interconnection structure**.
- The design of this interconnection structure will depend on the exchanges that must be made between modules.
- A group of wires, called **bus** is used to provide necessary signals for communication between modules.
- A bus that connects major computer components/modules (CPU, memory, I/O) is called a **system bus**. The system bus is a set of conductors that connects the CPU, memory and I/O modules.

- Usually, the system bus is separated into three functional groups :
  - Data Bus
  - Address Bus
  - Control Bus

1. Data Bus :

- The data bus consists of 8, 16, 32 or more parallel signal lines. These lines are used to send data to memory and output ports, and to receive data from memory and input port. Therefore, data bus lines are **bi-directional.**
- This means that CPU can read data on these lines from memory or from a port, as well as send data out on these lines to a memory location or to a port.
- The data bus is connected in parallel to all peripherals.
- The communication between peripheral and CPU is activated by giving output enable pulse to the peripheral.
- Outputs of peripherals are floated when they are not in use.

2. Address Bus :

- It is an **unidirectional** bus.
- The address bus consists of 16, 20, 24 or more parallel signal lines.
- On these lines the CPU sends out the address of the memory location or I/O port that is to be written to or read from.
- Here, the communication is one way, the address is send from CPU to memory and I/O port and hence these lines are unidirectional.

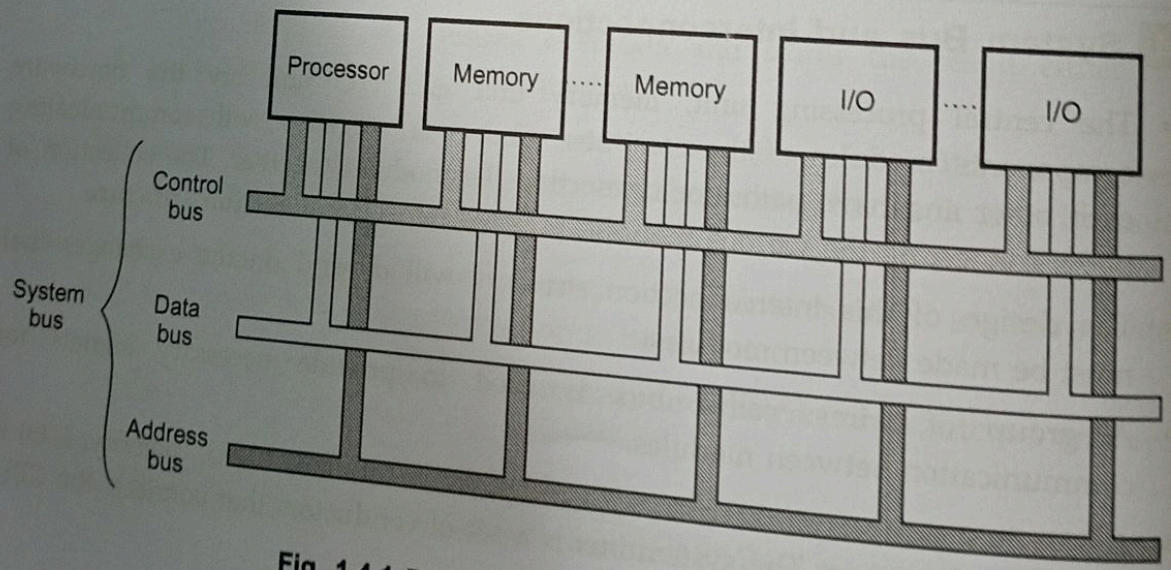


Fig. 1.4.1 Bus interconnection scheme

3. Control Bus :

- The control lines regulate the activity on the bus.
- The CPU sends signals on the control bus to enable the outputs of addressed memory devices or port devices.
- Fig. 1.4.1 shows bus interconnection scheme.

1.4.1 Single Bus Structure

- Another way to represent the same bus connection scheme is shown in Fig. 1.4.2.

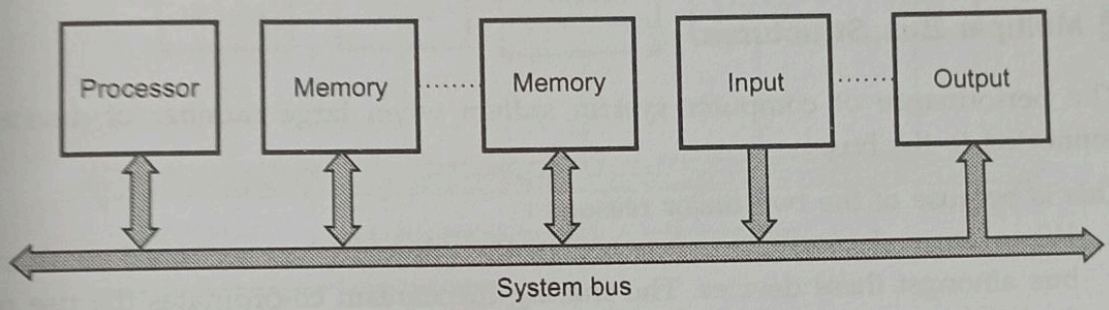


Fig. 1.4.2 Single bus structure

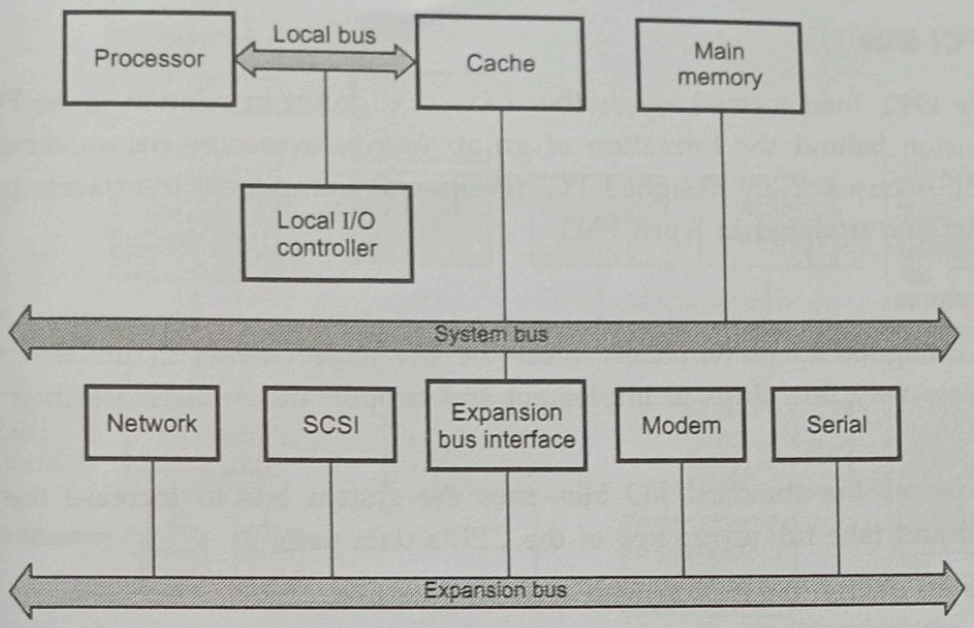
- Here, address bus, data bus and control bus are shown by single bus called **system bus**. Hence such interconnection bus structure is called **single bus structure**.
- In a single bus structure all units are connected to **common bus** called **system bus.**
- However, with single bus only two units can communicate with each other at a time.
- The bus control lines are used to arbitrate multiple requests for use of the bus.
- The main advantage of single bus structure is its low cost and its flexibility for attaching peripheral devices.
- The complexity of bus control logic depends on the amount of translation needed between the system bus and CPU, the timing requirements, whether or not interrupt management is included and the size of the overall system.
- For a small system, control signals of the CPU could be used directly to reduce handshaking logic. Drivers and receivers would not be needed for the data and address lines.
- But large systems with several interfaces would need bus driver and receiver circuits connected to the bus in order to maintain adequate signal quality.
- In most of the processors, multiplexed address and data buses are used to reduce the number of pins. During first part of bus cycle, address is present on this bus.

Afterwards, the same bus is used for data transfer purpose. So **latches** are required to hold the address sent by the CPU initially.

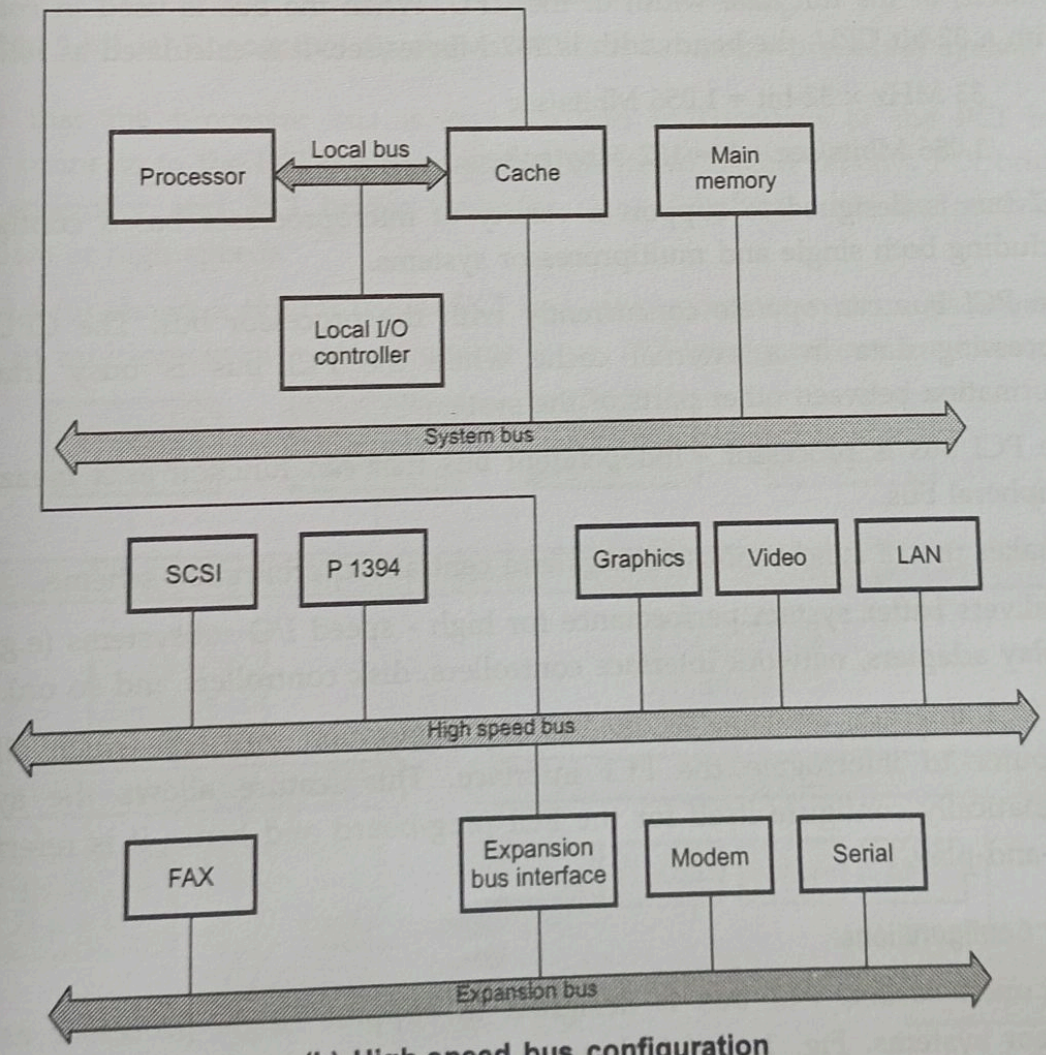
- Interrupt priority management is optional in a system. It is not required in systems which use software priority management.
- The complex system includes hardware for managing the I/O interrupts to increase efficiency of a system.
- Many manufacturers have made priority management devices. Programmable Interrupt Controller (PIC) is the IC designed to fulfil the same task.

1.4.2 Multiple Bus Structures

- The performance of computer system suffers when large number of devices are connected to the bus.
- This is because of the two major reasons :
  1. When more devices are connected to the common bus we need to share the bus amongst these devices. The sharing mechanism co-ordinates the use of bus to different devices. This co-ordination requires finite time called **propagation delay**. When control of the bus passes from one device to another frequently, these propagation delays are noticeable and affect the performance of computer system.
  2. When the aggregate data transfer demand approaches the capacity of the bus, the bus may become a bottleneck. In such situations we have to increase the data rate of the bus or we have to use wider bus.
- Now-a-days the data transfer rates for video controllers and network interfaces are growing rapidly.
- The need of high speed shared bus is impractical to satisfy with a single bus. Thus, most computer systems use the multiple buses.
- These buses have the hierarchical structure.
- Fig. 1.4.3 shows two bus configurations. The traditional bus connection uses three buses : Local bus, system bus and expanded bus.
- The high speed bus configuration uses high-speed bus along with the three buses used in the traditional bus connection.
- Here, cache controller is connected to high-speed bus.
- This bus supports connection to high-speed LANs, such as Fiber Distributed Data Interface (FDDI), video and graphics workstation controllers, as well as interface controllers to local peripheral buses including **Small Computer System Interface SCSI** and P1394.



(a) Traditional bus configuration



(b) High-speed bus configuration

Fig. 1.4.3

1.4.3 PCI Bus

In early 1992, Intel formed an another industrial group in relation to be PC bus. The main intension behind the formation of group was to overcome the weaknesses in the ISA and EISA buses. They designed PCI (Peripheral Component Interface) specifications in June 1992 and updated in April 1993.

1.4.3.1 Features

1. It is designed to economically meet the I/O requirements of modern systems. It requires very few chips to implement and support other buses attached to the PCI bus.
2. It bypasses the standard I/O bus, uses the system bus to increase the bus clock speed and take full advantage of the CPU's data path.
3. It has an ability to function with a 64-bit data bus.
4. It has high bandwidth. The information is transferred across the PCI bus at 33 MHz, at the full data width of the CPU. When the bus is used in conjunction with a 32-bit CPU, the bandwidth is 132 Mbytes/sec. It is calculated as follows :

   33 MHz × 32-bit = 1,056 Mbits/sec

   1,056 Mbits/sec ÷ 8 = 132 Mbytes/sec
5. PCI bus is designed to support a variety of microprocessor based configurations including both single and multiprocessor systems.
6. The PCI bus can operate concurrently with the processor bus. The CPU can be processing data in a external cache while the PCI bus is busy transferring information between other parts of the system.
7. The PCI bus is processor - independent bus that can function as a mezzanine or peripheral bus.
8. It makes use of synchronous timings and centralized arbitration scheme.
9. It delivers better system performance for high - speed I/O subsystems (e.g. graphic display adapters, network interface controllers, disk controllers and so on).
10. The PCI interface contains a 256 bytes configuration memory which allows the computer to interrogate the PCI interface. This feature allows the system to automatically configure itself for the PCI plug-board and hence it is referred to as plug-and-play.

1.4.3.2 PCI Configurations

As mentioned earlier, PCI bus is designed to support single processor as well as multiprocessor systems. Fig. 1.4.4 (a) shows a typical use of PCI in a single processor system.

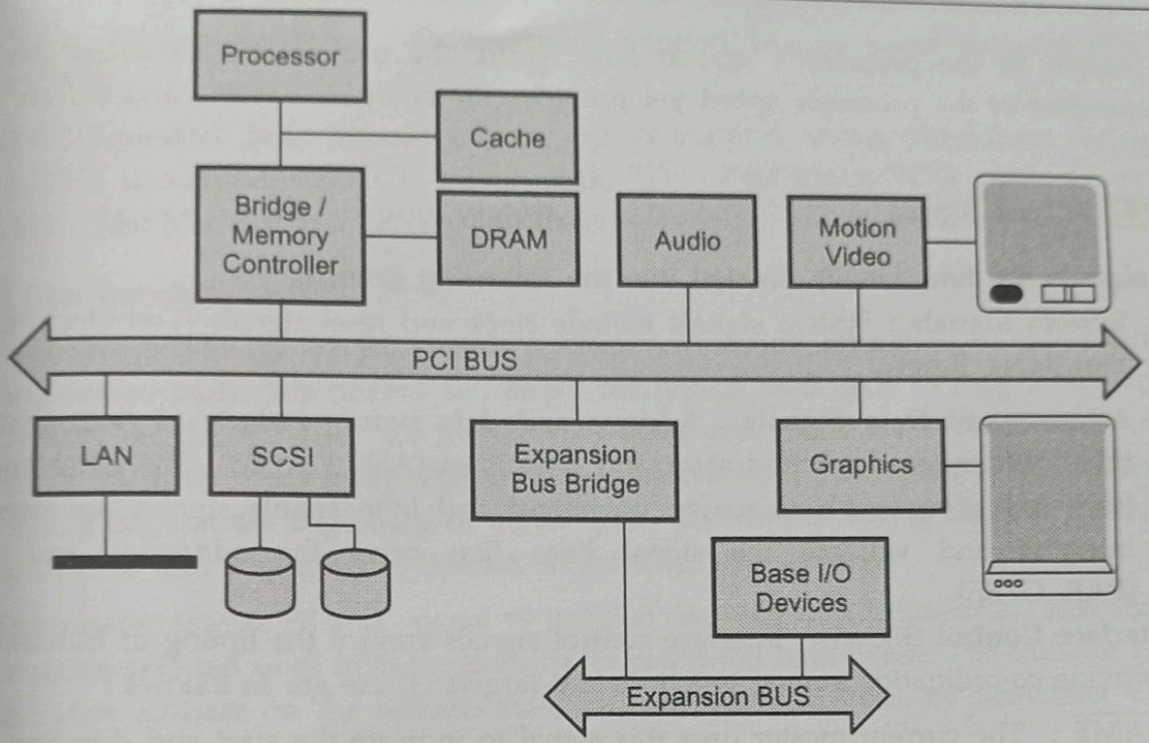


Fig. 1.4.4 (a) Conceptual diagram of PCI bus for single processor system

Notice that the processor bus is separate and independent of the PCI bus. The processor connects to the PCI bus through an integrated circuit called a PCI bridge. The memory controller and PCI bridge provides tight coupling with the processor and delivers data at high speeds.

Fig. 1.4.4 (b) shows a typical use of PCI in a multiprocessor system. As shown in the Fig. 1.4.4 in multiprocessor systems one or more PCI configurations may be connected

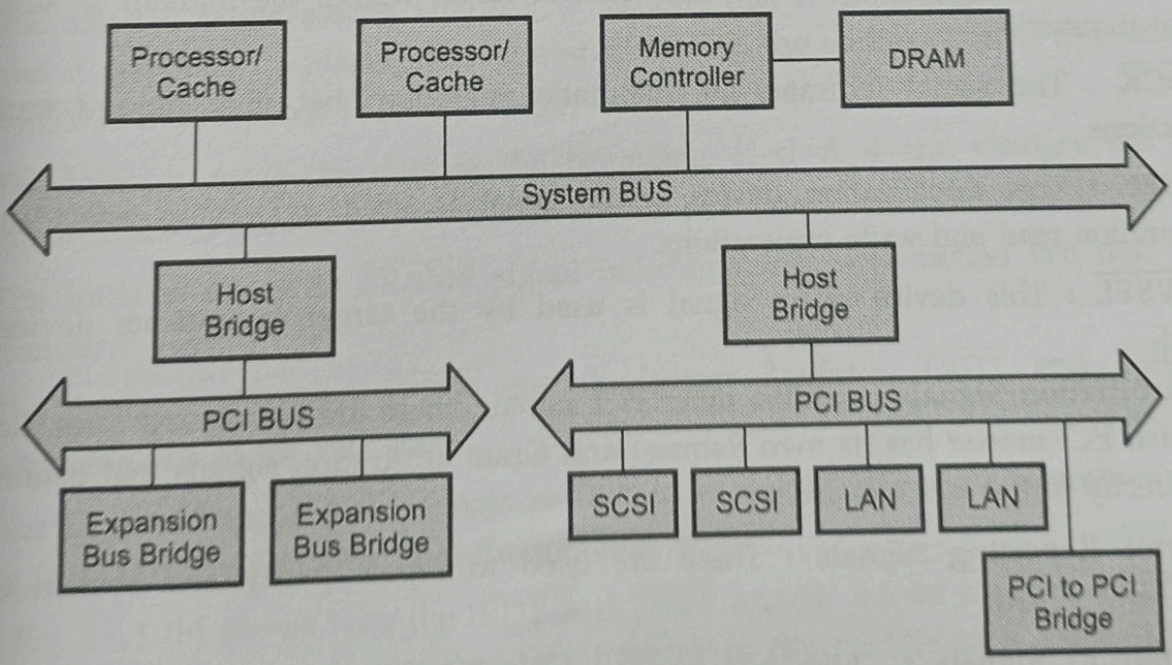


Fig. 1.4.4 (b) Conceptual diagram of PCI bus for multiprocessor system

by bridges to the processor's system bus. Again, the use of bridges keeps the PCI independent of the processor speed yet provides the ability to receive and deliver data rapidly.

1.4.3.3 PCI Bus Signals

PCI signals are functionally divided into the following groups :

- System Signals : System signals include clock and reset signals (Test clock, clock, Test Reset, Reset).
- **Address and Data Signals :** Address and data signals include 64 lines that are time - multiplexed for addresses and data lines (AD0 - AD63). The other signals from this group such as parity, command and byte enable signals are used to interpret and validate the signal lines that carry the addresses and data (PAR, C/BE).

Interface Control Signals : Interface control signals control the timing of transactions and provide co-ordination among initiators and targets. These are as follows :

$\overline{\text{FRAME}}$: The current master uses this signal to indicate the start and duration of a transaction. The signal is asserted at the start and deasserted when the initiator is ready to begin the final data phase.

$\overline{\text{IRDY}}$: The current master (initiator) uses this signal to indicate that it is ready to read or write valid data.

$\overline{\text{TRDY}}$: The target device (selected device) uses this signal to indicate that it is ready to read or write valid data.

$\overline{\text{STOP}}$: This signal indicates that the current target wishes the initiator to stop the current transaction.

$\overline{\text{LOCK}}$: The signal indicates an automatic operation that may require multiple transactions.

IDSEL : This initialization device select signal is used as a chip select during configuration read and write transactions.

$\overline{\text{DEVSEL}}$: This device select signal is used by the target to indicate device has selected.

- **Arbitration Signals :** Unlike other PCI signals, these are not shared lines. Rather, each PCI master has its own Request and Grant arbitration signals that connect it directly to the PCI bus arbiter.
- **Error Reporting Signals :** These are used to report parity and system errors ($\overline{\text{PERR}}$, $\overline{\text{SERR}}$).
- **Interrupt Signals :** Like bus arbitration signals these are not shared signals. Rather, each PCI device has its own interrupt signals ($\overline{\text{INTA}}$, $\overline{\text{INTB}}$, $\overline{\text{INTC}}$, $\overline{\text{INTD}}$).

- **Cache Support Signals :** These signals support snoopy cache protocol ($\overline{SBO}$, SDONE).
- **JTAG/Boundary Scan Signals :** These signals support testing procedures defined in IEEE standard 149.1 (TCK - Test clock, TDI - Test Input, TDO - Test Output, TMS - Test Mode Select, TRST - Test Reset).

1.4.3.4 Data Transfer

Every data transfer on the PCI bus is a single transaction consisting of one address phase and one or more data phases. Let us see the typical read cycle to read 32-bit word from memory on the PCI bus. All the events during read cycle are synchronized with the falling edge of the clock cycle. The devices connected to the bus sample the bus lines on the rising edge at the beginning of a bus cycle. The events during the bus cycle are explained below and they are also labeled on the diagram.

1. The master asserts $\overline{FRAME}$ signal to indicate the start of a transaction. This signal remains asserted until initiator is ready to begin the data phase. The initiator puts the start address on the address bus, and initiates read cycle by activating C/$\overline{BE}$ lines.
2. At the beginning of clock2, the target device recognizes its address on the address bus.
3. The initiator ceases the address lines and it changes the information on the C/$\overline{BE}$ lines to designate which address lines are to be used for transfer for the currently addressed data from 1 to 4 bytes. The initiator also asserts IRDY to indicate that it is ready for the first data byte.
4. The selected target asserts $\overline{DEVSEL}$ to indicate that it has recognized its address and it responds by placing the requested data on the address bus. It then asserts $\overline{TRDY}$ to indicate that valid data is present on the bus.
5. The initiator reads the data at the beginning of clock 4 and changes the byte enable lines as needed in preparation for the next read.
6. The initiator deasserts $\overline{FRAME}$ signal indicating that it is second last data byte transfer.
7. The initiator deasserts $\overline{IRDY}$ and the target deasserts $\overline{TRDY}$ and $\overline{DEVSEL}$, returning bus to the idle state.

Similar kind of handshaking signals are activated during PCI write operation. Here, the data from initiator is copied into the targeted devices.

The Fig. 1.4.5 (b) shows how the $\overline{IRDY}$ and $\overline{TRDY}$ signals can be used by the initiator and target, respectively to pause the data transfer in the middle of a transaction. As shown in the Fig. 1.4.5 (b) the read cycle operation is same upto two data bytes transfer.

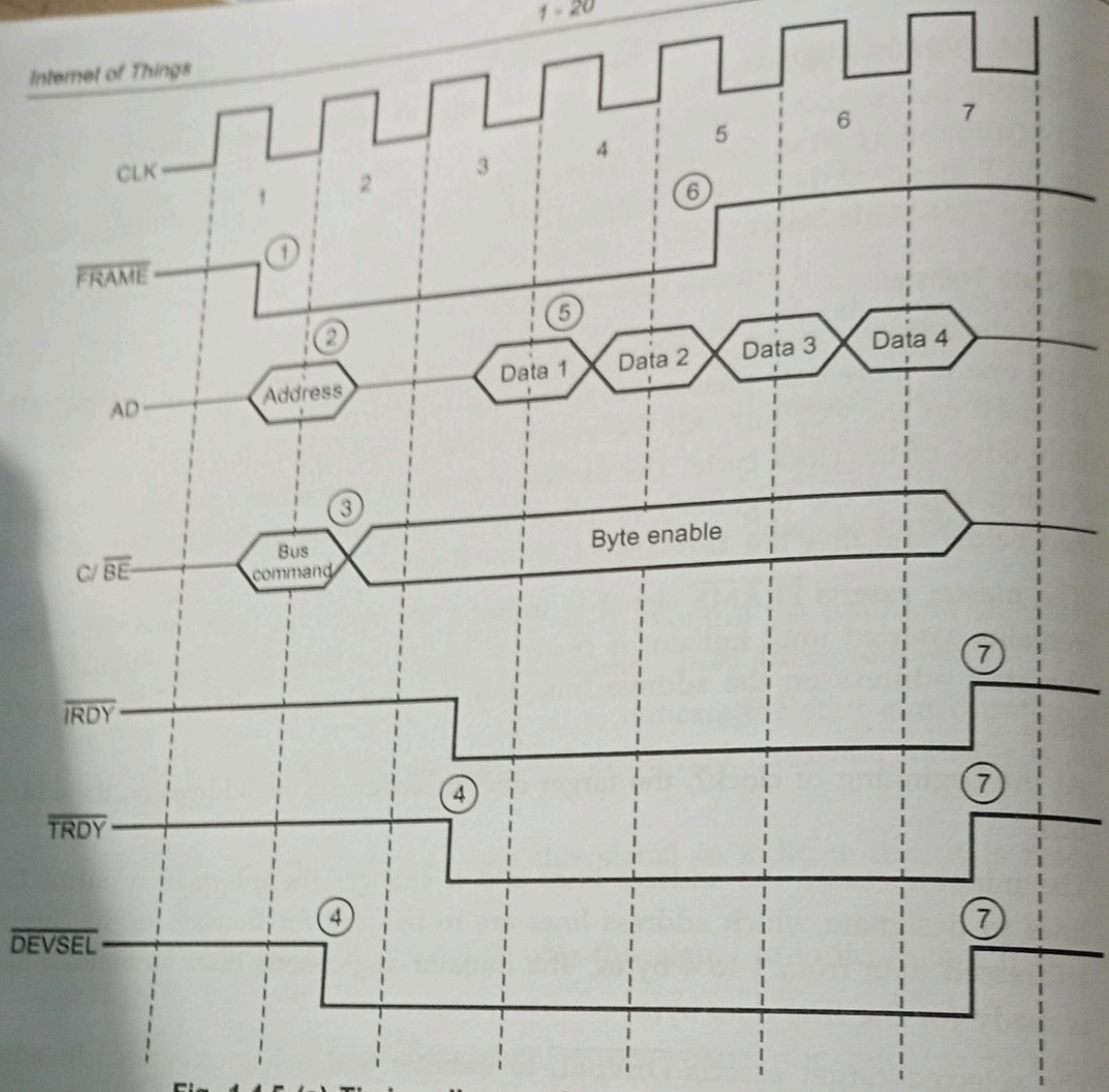


Fig. 1.4.5 (a) Timing diagram for PCI read operation

The target sends the third word in cycle 5. In Fig. 1.4.5 (b), it is assumed that the initiator is not able to receive it. Hence, it deasserts $\overline{\text{IRDY}}$. In response, the target holds the third data byte on the AD lines until $\overline{\text{IRDY}}$ is asserted again.

In cycle 6, it is assumed that the initiator is ready to receive next data bytes thus is asserts $\overline{\text{IRDY}}$ signal and reads third data byte. At this point, it is assumed that the target is not ready to transfer the fourth word. The target informs this by deasserting $\overline{\text{TRDY}}$ line. In cycle 8, target sends the fourth word and asserts $\overline{\text{TRDY}}$. Since $\overline{\text{FRAME}}$ signal is deasserted during the third data byte transfer, the transaction ends after the fourth data byte transfer.

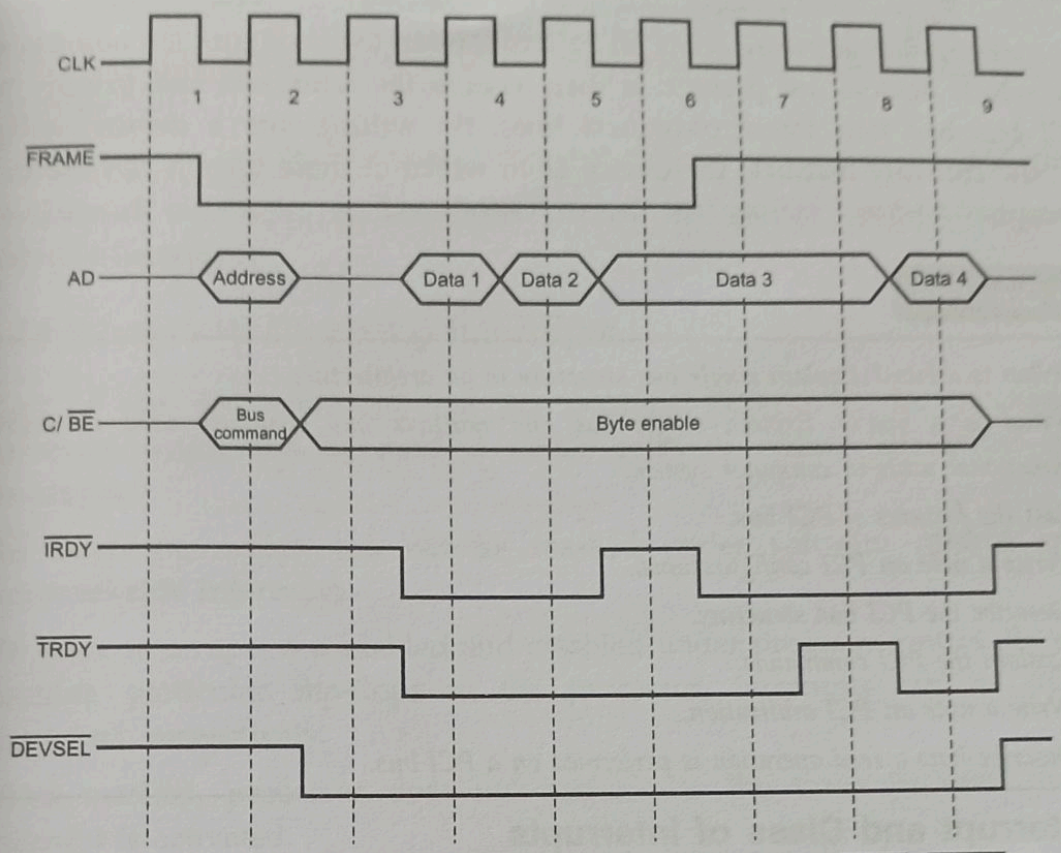


Fig. 1.4.5 (b) PCI read operation showing the role of $\overline{\text{IRDY}}$ / $\overline{\text{TRDY}}$

Device Configuration

When an I/O device is connected to a computer, it is necessary to configure both the device and the driver software for that device. To configure the device we may need to set switches and jumpers. On the other hand, to configure software we need to know the device's address, characteristics and so on. The PCI simplifies the process of configuration of devices by incorporating a small configuration ROM memory in each I/O device interface. This memory stores information about that device. The configuration ROMs of all devices are accessible in the configuration address space. When power up or reset, PCI initialization software reads these ROMs, determines the type of the devices and assigns addresses to them. During bus configuration operation, addresses are not yet assigned to the devices and hence they cannot accessed based on their addresses.

During a configuration operation, $\overline{\text{IDSEL}}$ (Initialization device select) signal is used to select the device rather than the device addresses. About of 32 address lines, AD11 to AD31, i.e. 21 address lines are used to connect $\overline{\text{IDSEL}}$ pin of each device. This restricts the number of I/O devices that can be connected to 21. The lower 11 address lines AD0 to AD10 are used to specify the type of operation and to access the contents of the device configuration ROM.

The configuration software scans all 21 locations in the configuration address space to identify which devices are present. It then assigns the addresses and priority to them. The PCI bus has four interrupt-request lines. By writing into a device configuration register, the software instructs the device as to which of these lines it can use to request an interrupt.

Review Questions

1. *What is a bus ? Explain single bus structure in an architecture.*
2. *What is a bus ? Explain single bus and multiple bus structure used to interconnect functional units in computer system.*
3. *List the features of PCI bus.*
4. *Write a note on PCI configurations.*
5. *Describe the PCI bus structure.*
6. *Explain the PCI commands.*
7. *Write a note on PCI arbitration.*
8. *Describe how a read operation is performed on a PCI bus.*

1.5 Interrupt and Class of Interrupts

- Sometimes it is necessary to have the computer automatically execute one of a collection of special routines whenever certain conditions exists within a program or the computer system e.g. It is necessary that computer system should give response to devices such as keyboard, sensor and other components when they request for service.
- This method provides an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine (Interrupt Service Routine) that will service the requesting device. Once this servicing is completed, the processor would resume exactly where it left off. The event that causes the interruption is called **interrupt** and the special routine executed to service the interrupt is called **Interrupt Service Routine (ISR).**

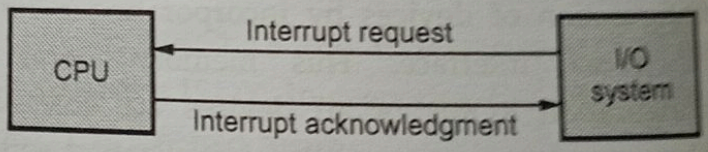


Fig. 1.5.1

- The interrupt service routine is different from subroutine because the address of ISR is predefined or it is available in Interrupt Vector Table (IVT), whereas subroutine address is necessarily to be given in subroutine CALL instruction. IRET instruction is used to return from the ISR whereas RET instruction is used to return from subroutine. IRET instruction restores flag contents along with CS and

IP in the IA-32 architecture; however RET instruction only restores CS and IP contents.

- An interrupt caused by an external signal is referred as a **hardware interrupt**.
- Conditional interrupts or interrupts caused by special instructions are called **software interrupts**.

1.5.1 Enabling and Disabling Interrupts

- Most of the processors provide the masking facility. In the processor those interrupts which can be masked under software control are called **maskable interrupts**.
- The interrupts which can not be masked under software control are called **non-maskable interrupts**.
- Maskable interrupts are enabled and disabled under program control. By setting or resetting particular flip-flops in the processor, interrupts can be masked or unmasked, respectively.
- When masked, processor does not respond to the interrupt even though the interrupt is activated.

1.5.2 Vector Interrupts

- When the external device interrupts the processor (interrupt request), processor has to execute interrupt service routine for servicing that interrupt. If the internal control circuit of the processor produces a CALL to a predetermined memory location which is the starting address of interrupt service routine, then that address is called **vector address** and such interrupts are called **vector interrupts**. For vector interrupts fastest and most flexible response is obtained since such an interrupt causes a direct hardware-implemented transition to the correct interrupt-handling program. This technique is called **vectoring**. When processor is interrupted, it reads the vector address and loads it into the PC.

1.5.3 Interrupt Nesting

- For some devices, a long delay in responding to an interrupt request may cause error in the operation of computer. Such interrupts are acknowledged and serviced eventhough processor is executing an interrupt service routine for another device.
- A system of interrupts that allows an interrupt service routine to be interrupted is known as **nested interrupts**.

1.5.4 Interrupt Priority

- When interrupt requests arrive from two or more devices simultaneously, the processor has to decide which request should be serviced first and which one should be delayed. The processor takes the decision with the help of interrupt priorities.
- It accepts the request having the highest priority.

1.5.5 Recognition of Interrupt and Response to Interrupt

- The CPU recognizes the interrupt when the external asynchronous input (interrupt input) is asserted (a signal is sent to the interrupt input) by an I/O device.
- In response to an interrupt a special sequence of actions are performed. These are as follows :

- When a processor is interrupted, it stops executing its current program and calls a special routine which "services" the interrupt. The event that causes the interruption is called **interrupt** and the special routine which is executed is called **interrupt service routine.**

1. The processor completes its current instruction. No instruction is cut-off in the middle of its execution.
2. The program counter's current contents are stored on the stack. Remember, during the execution of an

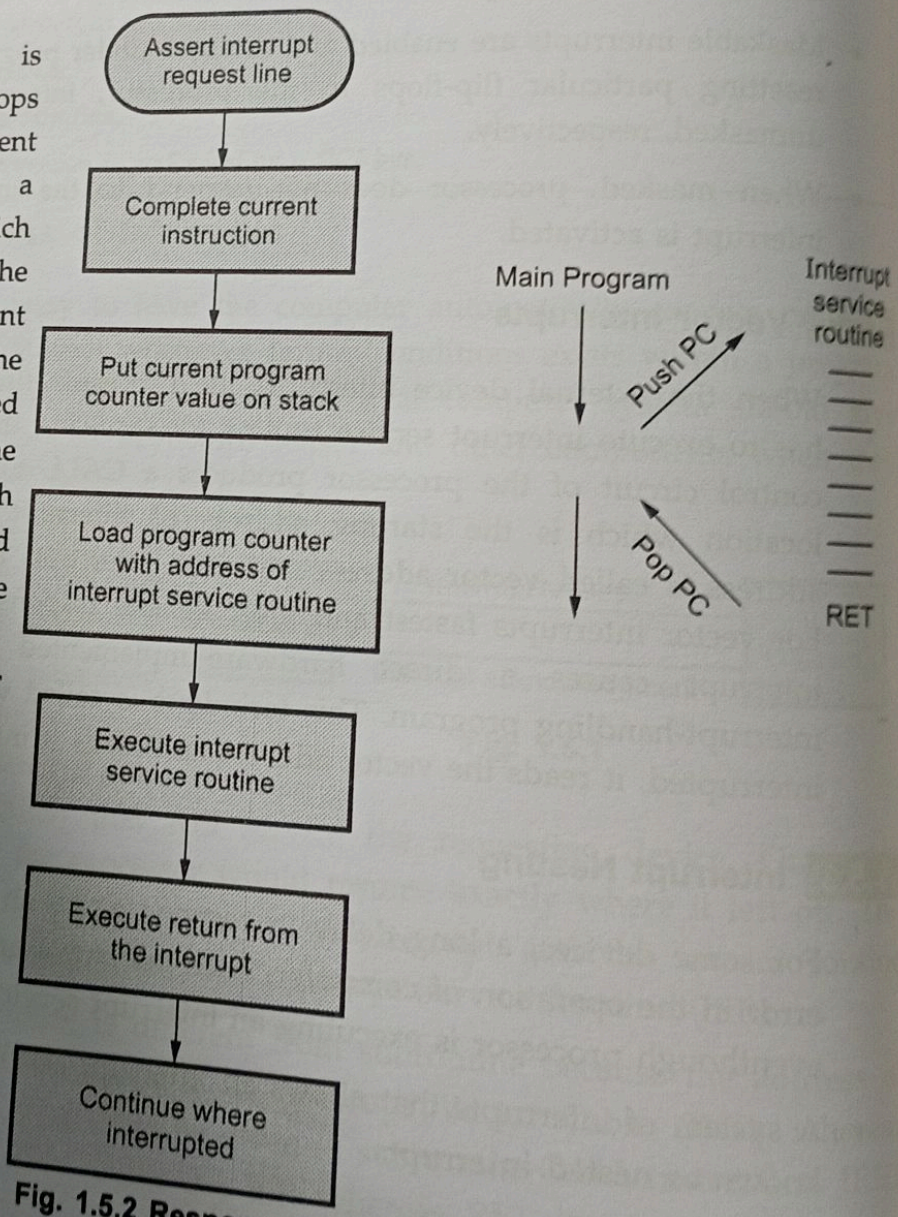


Fig. 1.5.2 Response to an interrupt with the flowchart and diagram

instruction the program counter is pointing to the memory location for the next instruction.

3. The program counter is loaded with the address of an interrupt service routine.
4. Program execution continues with the instruction taken from the memory location pointed by the new program counter contents.
5. The interrupt program continues to execute until a return instruction is executed.
6. After execution of the RET instruction processor gets the old address (the address of the next instruction from where the interrupt service routine was called.) of the program counter form the stack and puts it back into the program counter. This allows the interrupted program to continue executing at the instruction following the one where it was interrupted. Fig. 1.5.2 shows the response to an interrupt with the flowchart and diagram.

Review Questions

1. *What is interrupt ?*
2. *What is a non-maskable interrupt ? What is the action performed on receipt of a NMI ?*
3. *What are vectored interrupts ?*
4. *What do you mean by interrupt nesting ?*
5. *What is priority interrupt ?*

1.6 Instruction Cycle (I - Cycle)

- An instruction cycle involves three subcycles,
  - **Fetch :** The fetch phase reads the next instruction from memory into the CPU.
  - **Decode :** The decode phase interprets the opcode by decoding it.
  - **Execute :** The execute phase performs the indicated operation.
- Fig. 1.6.1 shows the basic instruction cycle.
- Actually, processor checks for valid interrupt request after each instruction cycle. If any valid interrupt request is present, processor saves the current process state and services the interrupt.
- If the operands on which the instruction works are present within the processor-registers, a memory access is not required. But if the execution of an instruction involves one or more operands in memory, each requires a memory access.
- For fetching the indirect addresses, one or more instruction sub cycles are required.

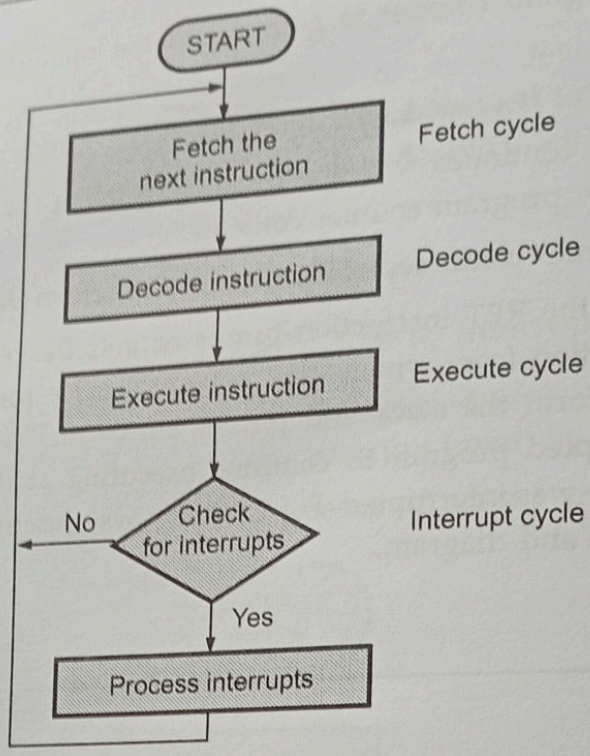


Fig. 1.6.1 Basic instruction cycle with interrupt cycle

- After fetching the instruction, it is decoded and if any indirect addressing is involved, the required operands are fetched using indirect addressing.
- Also, after performing the operation on the operands according to the opcode, a similar process may be needed to store the result in memory.
- Following execution, interrupt processing may be required before fetchingthe next instruction. The same process can be viewed as shown in Fig. 1.6.2.

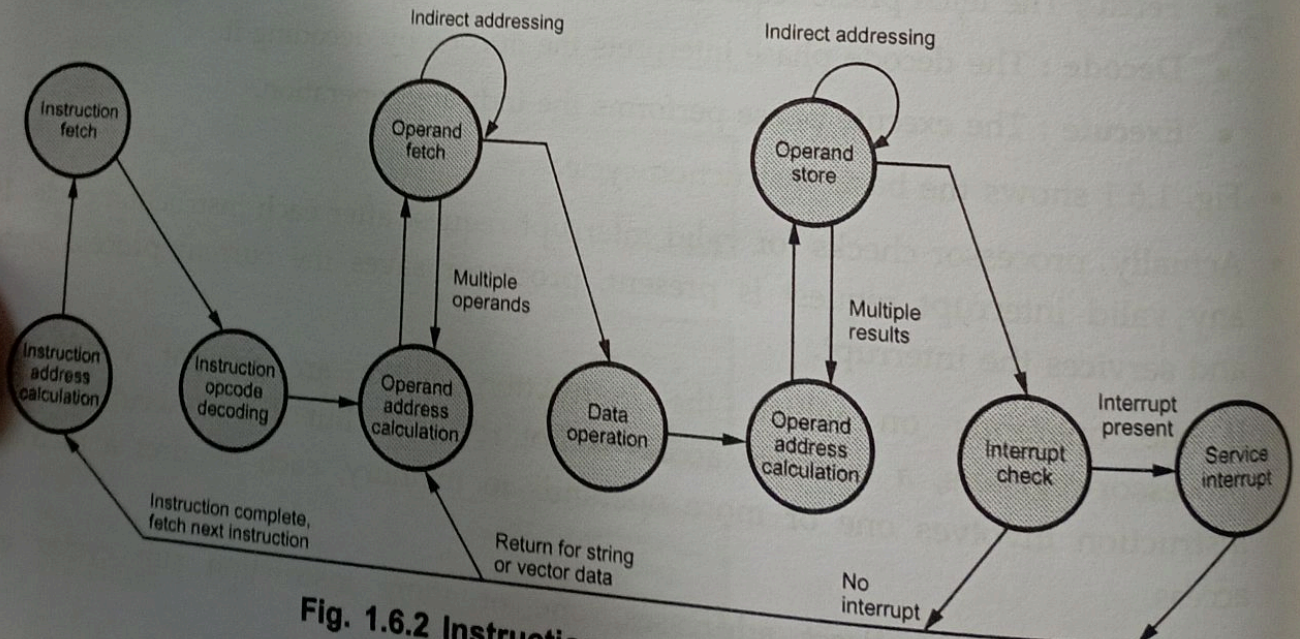


Fig. 1.6.2 Instruction cycle state diagram

1.6.1 Fetch Cycle

- During the **fetch cycle** an instruction opcode is read from memory. The following sequence of operations take place during this cycle.
  - The address of an instruction to be fetched from memory is in PC. This address is moved to AR.
  - The contents of AR (i.e. address) is placed on the address bus.
  - The control unit generates the control signal to read the contents of the memory.
- The contents of the addressed memory location are placed on the data bus and copied into the DR.
- The data in DR is moved to IR.
- Meanwhile, the PC is incremented by one for pointing the next data in memory.

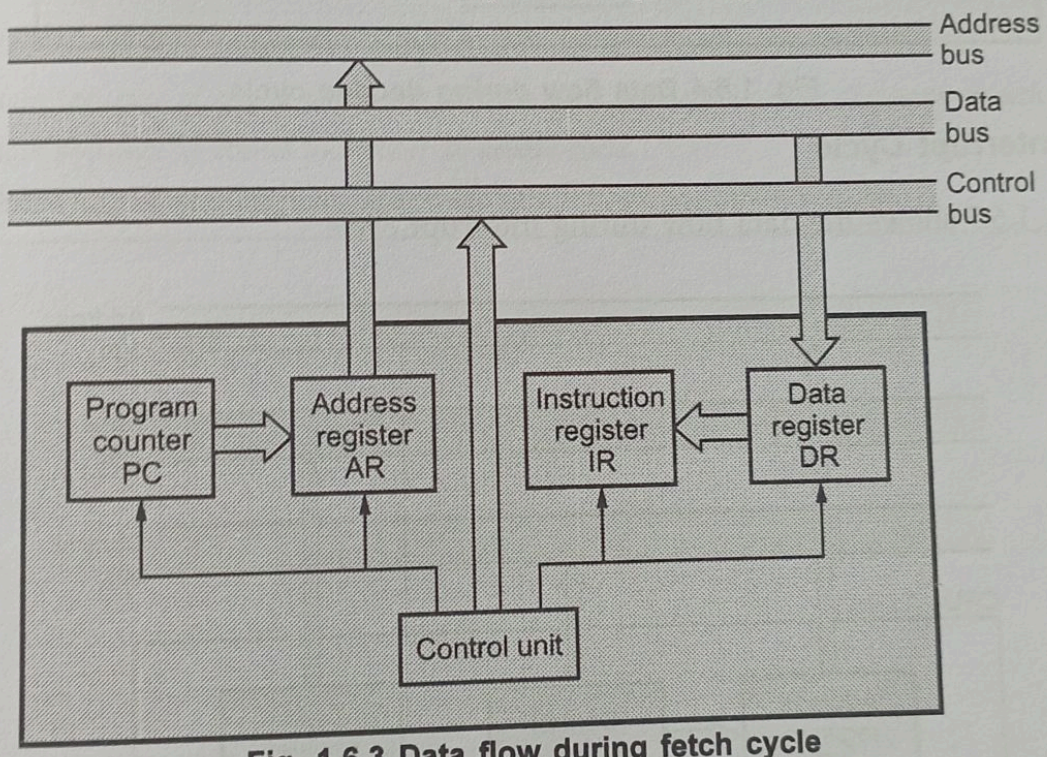


Fig. 1.6.3 Data flow during fetch cycle

1.6.2 Decode Cycle

During **decode cycle**, the IR contents (i.e. opcode) are decoded by ID. During decoding operation, it is determined that the operands are present within registers of CPU or indirect addressing is required. For indirect addressing, indirect cycle is performed. The DR contains the address reference. This is transferred to the AR. The control unit generates the control signal to read the desired address of the operand into DR as shown in Fig. 1.6.4.

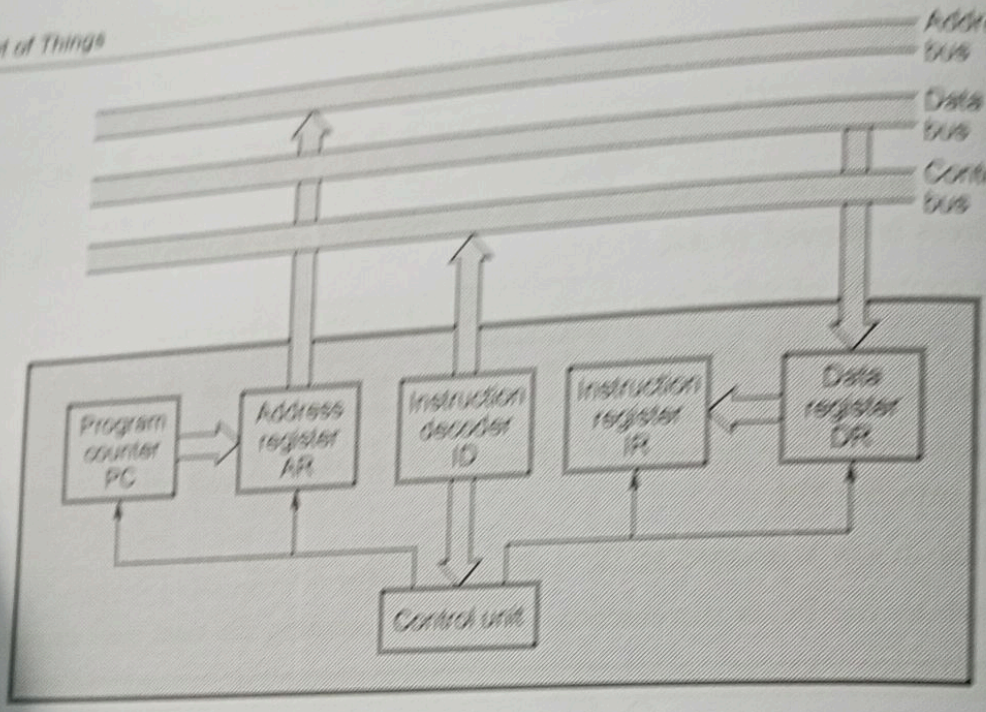


Fig. 1.6.4 Data flow during decode cycle

1.6.3 Interrupt Cycle

- Fig. 1.6.5 shows the data flow during interrupt cycle.

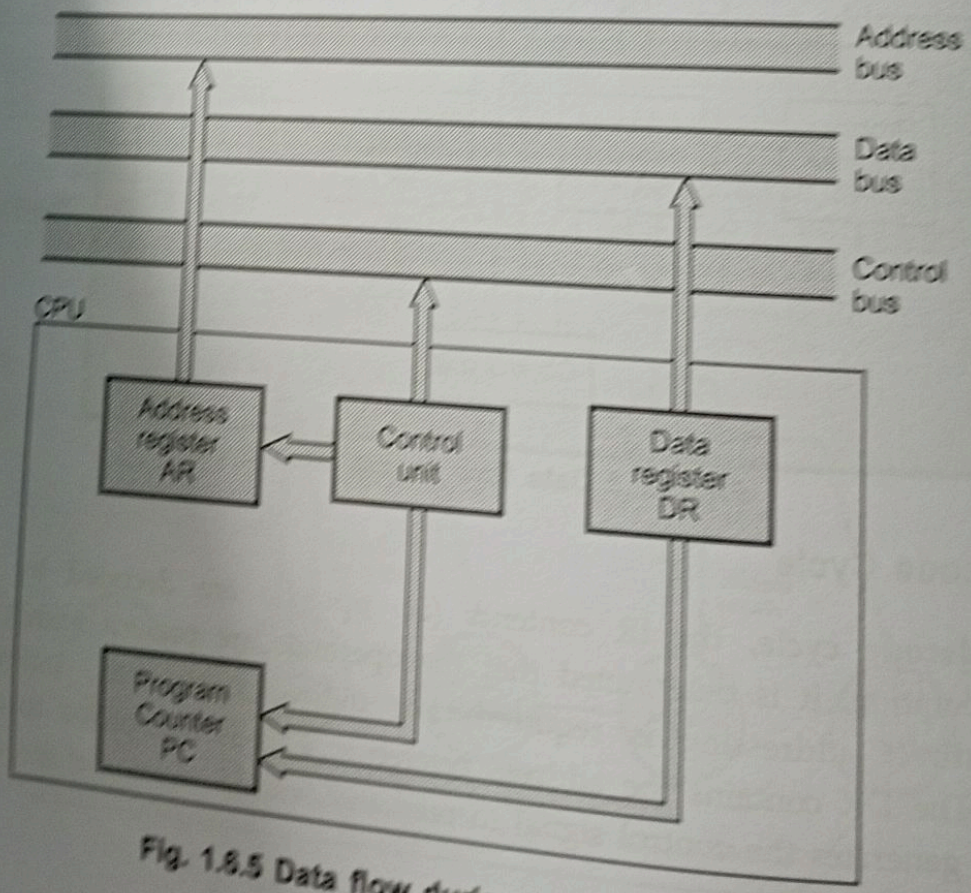
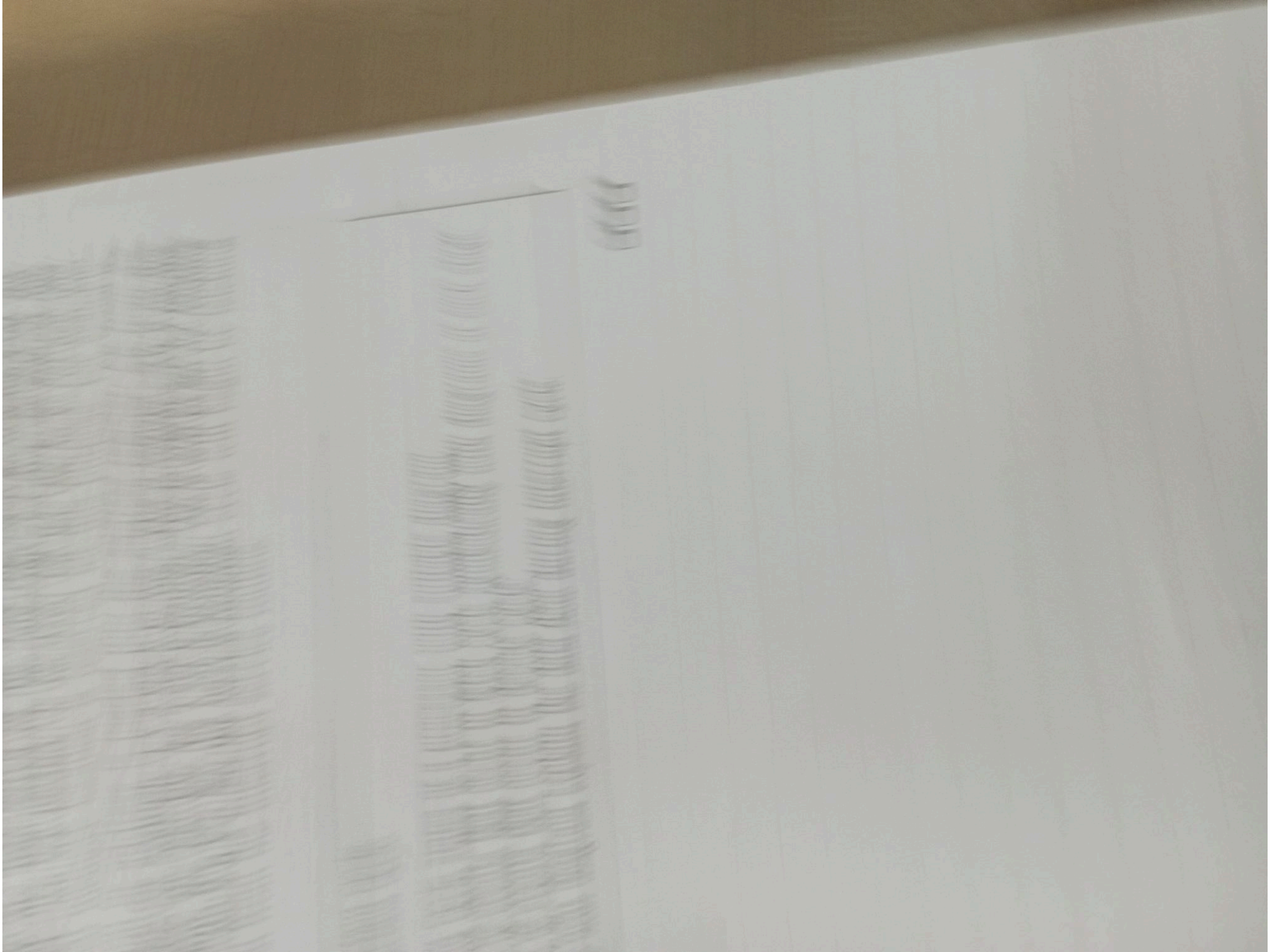


Fig. 1.6.5 Data flow during interrupt cycle



UNIT I

2 Number Systems

Syllabus

Number systems, Decimal Number system, Binary number system, Octal & Hexadecimal number system, 1's & 2's complement, Binary Fixed Point Representation.

Contents

2.1 Introduction

- Number system is a basis for counting various items.
- The decimal number system has 10 digits : 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
- Modern computers communicate and operate with binary numbers which use only the digits 0 and 1.
- When decimal quantities are represented in the binary form, they take more digits.
- For large decimal numbers people have to deal with very large binary strings and therefore, they do not like working with binary numbers. This fact gave rise to three new number systems : **Octal**, **Hexadecimal** and **Binary Coded Decimal (BCD)**.

Review Questions

1. *Explain various number systems.*
2. *Name the number system used in computers.*

2.2 Representation of Numbers in Different Radix

2.2.1 Decimal Number System

- In decimal number system we can express any decimal number in units, tens, hundreds, thousands and so on.
- When we write a decimal number say, 5678.9, we know it can be represented as $5000 + 600 + 70 + 8 + 0.9 = 5678.9$
- The decimal number 5678.9 can also be written as 5678.9_{10}, where the 10 subscript indicates the **radix or base.**
- The position of a digit with reference to the decimal point determines its value/weight. The sum of all the digits multiplied by their weights gives the total number being represented.
- The leftmost digit, which has the greatest weight is called the **most significant digit** and the rightmost digit, which has the least weight, is called the **least significant digit**.
- Fig. 2.2.1 shows decimal digit and its weights expressed as a power of 10.

	10^3	10^2	10^1	10^0		10^{-1}
	5	6	7	8	.	9
In power of 10	5×10^3	6×10^2	7×10^1	8×10^0	.	9×10^{-1}
	MSD					LSD

Fig. 2.2.1 Representation of a decimal number

2.2.2 Binary Number System

- Binary system with its two digits is a **base-two system.**
- The two binary digits (bits) are 1 and 0.
- In binary system, weight is expressed as a power of 2.
- The Fig. 2.2.2 (a) shows representation of binary number 1101.101 in power of 2.
- By adding each digit of a binary number in a power of 2 we can find the decimal equivalent of the given binary number.

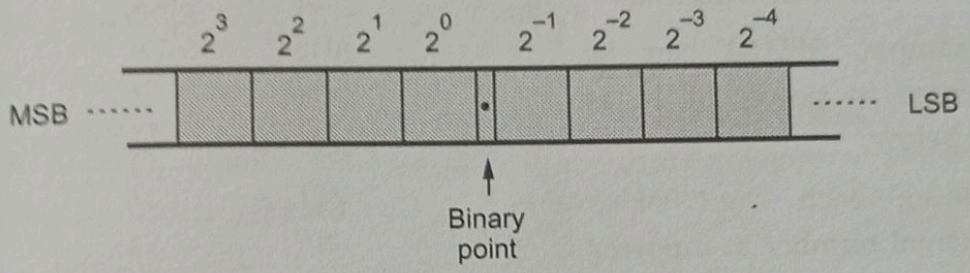


Fig. 2.2.2 Binary position values as a power of 2

	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
N =	1	1	0	1	.	1	0	1
	1×2^3	1×2^2	0×2^1	1×2^0	.	1×2^{-1}	0×2^{-2}	1×2^{-3}

$$N = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = (13.625)_{10}$$

Fig. 2.2.2 (a)

2.2.3 Octal Number System

- The octal number system uses first eight digits of decimal number system : 0, 1, 2, 3, 4, 5, 6 and 7. As it uses 8 digits, its base is 8.
- For example, the octal number 5632.471 can be represented in power of 8 as shown in Fig. 2.2.2 (b).

	8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
N =	5	6	3	2	.	4	7	1
	5×8^3	6×8^2	3×8^1	2×8^0	.	4×8^{-1}	7×8^{-2}	1×8^{-3}

$$N = 5\times8^3 + 6\times8^2 + 3\times8^1 + 2\times8^0 + 4\times8^{-1} + 7\times8^{-2} + 1\times8^{-3} = (2970.611328)_{10}$$

Fig. 2.2.2 (b)

- By adding each digit of an octal number in a power of 8 we can find the decimal equivalent of the given octal number.

2.2.4 Hexadecimal Number System

- The hexadecimal number system has a base of 16 having 16 characters : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.
- Table 2.2.1 shows the relationship between decimal, binary, octal and hexadecimal.
- For example, 3FD.84 can be represented in power of 16 as shown below.
- By adding each digit of a hexadecimal number in a power of 16 we can find decimal equivalent of the given hexadecimal number.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 2.2.1 Relation between decimal, binary, octal and hexadecimal numbers

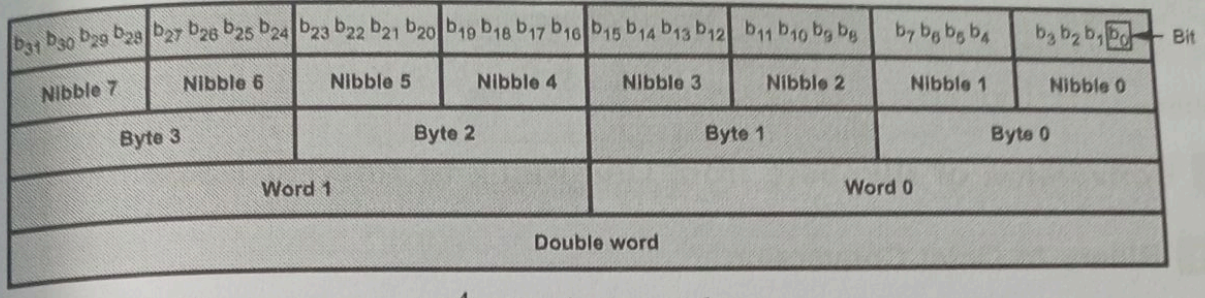
	16^2	16^1	16^0		16^{-1}	16^{-2}
N =	3	F	D	.	8	4
	3×16^2	$F \times 16^1$	$D \times 16^0$	.	8×16^{-1}	4×16^{-2}

$$N = 3 \times 16^2 + F \times 16^1 + D \times 16^0 + 8 \times 16^{-1} + 4 \times 16^{-2}$$

$$= 3 \times 16^2 + 15 \times 16^1 + 13 \times 16^0 + 8 \times 16^{-1} + 4 \times 16^{-2} = (1021.515625)_{10}$$

2.2.5 Format of a Binary Number

- A single digit in the binary number is called **bit.**
- The following figure shows the format of binary number. Four binary digits form a **nibble**, eight binary digits form a **byte**, sixteen binary digits form a word and thirty-two binary digits form a double-word.



Nibble : 4-bits can represent $2^4 = 16$ distinct values

Byte : 8-bits can represent $2^8 = 256$ distinct values

Word : 16-bits can represent $2^{16} = 65536$ distinct values

Double word : 32-bits can represent $2^{32} = 4294967296$ distinct values

2.2.6 Counting in Radix (Base) r

- Each number system has r set of characters. For example, in decimal number system r equals to 10 has 10 characters from 0 to 9, in binary number system r equals to 2 has 2 characters 0 and 1 and so on.
- In general we can say that, a number represented in radix r, has r characters in its set and r can be any value. This is illustrated in Table 2.2.2.

Radix (Base) r	Characters in set
2 (Binary)	0, 1
3	0, 1, 2
4	0, 1, 2, 3
5	0, 1, 2, 3, 4
6	0, 1, 2, 3, 4, 5
7	0, 1, 2, 3, 4, 5, 6
8 (Octal)	0, 1, 2, 3, 4, 5, 6, 7
9	0, 1, 2, 3, 4, 5, 6, 7, 8
10 (Decimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
:	
:	
16 (Hexadecimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Table 2.2.2 Radix and character set

Example 2.2.1 *Find the decimal equivalent of $(231.23)_4$.*

Solution :

$$N = 2 \times 4^2 + 3 \times 4^1 + 1 \times 4^0 + 2 \times 4^{-1} + 3 \times 4^{-2}$$

$$= 32 + 12 + 1 + 0.5 + 0.1875$$

$$= \mathbf{45.6875_{10}}$$

Example 2.2.2 *Count from 0 to 9 in radix 5.*

Solution : The Table 2.2.2 indicates that radix 5 has 5 characters. A count sequence from 0 decimal to 9 decimal is

00, 01, 02, 03, 04, 10, 11, 12, 13, 14

Example 2.2.3 *What is the largest binary number that can be expressed with 12-bits ? What is the equivalent decimal and hexadecimal ?*

Solution : $(1111\ 1111\ 1111)_2$, $(4095)_{10}$, $(FFF)_H$.

2.3 Conversion of Numbers from One Radix to Another Radix

2.3.1 Binary to Octal Conversion

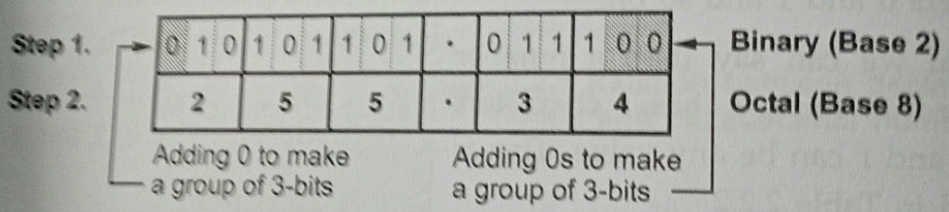
- The base for octal number is 8 and the base for binary number is 2.
- The base for octal number is the third power of the base for binary numbers. Therefore, by grouping 3 digits of binary numbers and then converting each group digit to its octal equivalent we can convert binary number to its octal equivalent.

Example 2.3.1 *Convert 10101101.0111 to octal equivalent.*

Solution :

Step 1 : Make group of 3-bits starting from LSB for integer part and MSB for fractional part, by adding 0s at the end, if required.

Step 2 : Write equivalent octal number for each group of 3-bits.



$\therefore \quad (10101101.0111)_2 = (255.34)_8$

2.3.2 Octal to Binary Conversion

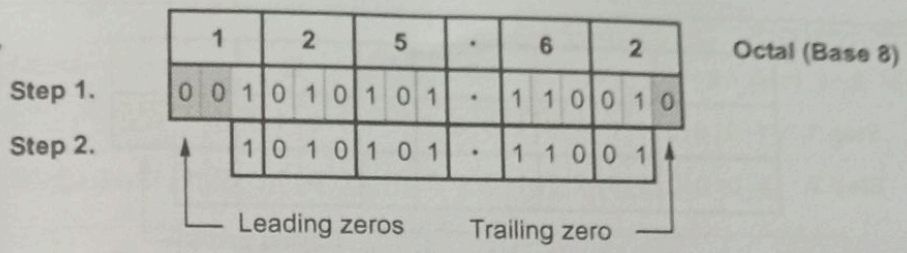
- Conversion from octal to binary is a reversal of the process explained in the previous section. Each digit of the octal number is individually converted to its binary equivalent to get octal to binary conversion of the number.

Example 2.3.2 *Convert $(125.62)_8$ to binary.*

Solution :

Step 1 : Write equivalent 3-bit binary number for each octal digit.

Step 2 : Remove any leading or trailing zeros.



$\therefore \quad (125.62)_8 = (1010101.11001)_2$

2.3.3 Binary to Hexadecimal Conversion

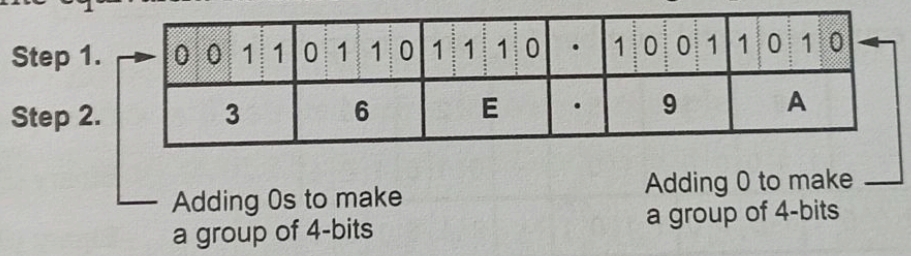
- The base for hexadecimal numbers is 16 and the base for binary numbers is 2.
- The base for hexadecimal number is the fourth power of the base for binary numbers. Therefore, by grouping 4 digits of binary numbers and then converting each group digit to its hexadecimal equivalent we can convert binary number to its hexadecimal equivalent.

Example 2.3.3 *Convert 1101101110 · 1001101 to hexadecimal equivalent.*

Solution :

Step 1 : Make group of 4-bits starting from LSB for integer part and MSB for fractional part, by adding 0s at the end, if required.

Step 2 : Write equivalent hexadecimal number for each group of 4-bits.



$\therefore \quad (1101101110.1001101)_2 = (36E.9A)_{16}$

2.3.4 Hexadecimal to Binary Conversion

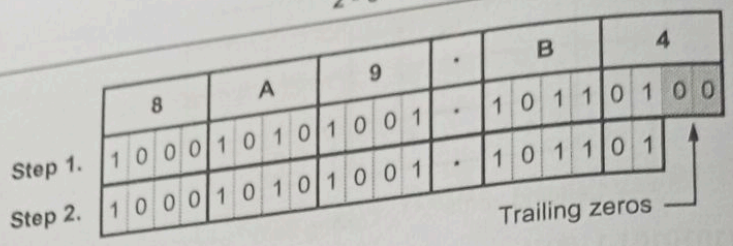
- Conversion from hexadecimal to binary is a reversal of the process explained in the previous section. Each digit of the hexadecimal number is individually converted to its binary equivalent to get hexadecimal to binary conversion of the number.

Example 2.3.4 *Convert $(8A9.B4)_{16}$ to binary.*

Solution :

Step 1 : Write equivalent 4-bit binary number of each hexadecimal digit.

Step 2 : Remove any leading or trailing zeros.



$\therefore \quad (8A9.B4)_{16} = (1000\ 1010\ 1001.101101)_2$

2.3.5 Octal to Hexadecimal Conversion

The easiest way to convert octal number to hexadecimal number is given below :

1. Convert octal number to its binary equivalent.
2. Convert binary number to its hexadecimal equivalent.

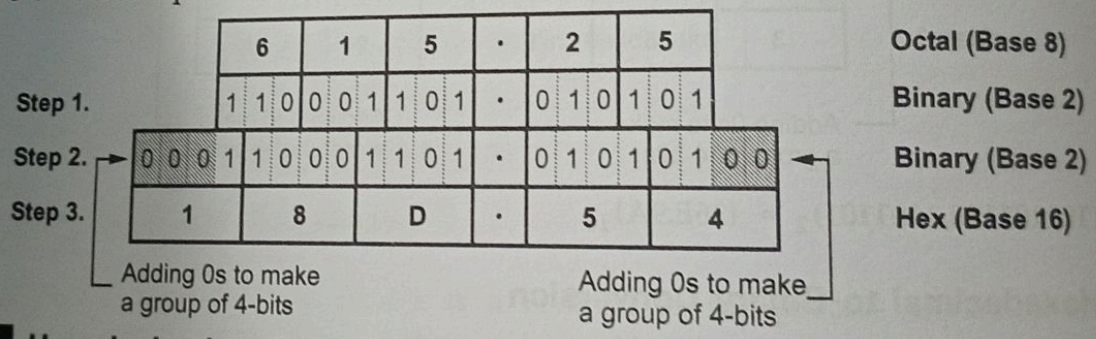
Example 2.3.5 *Convert $(615.25)_8$ to its hexadecimal equivalent.*

Solution :

Step 1 : Write equivalent 3-bit binary number for each octal digit.

Step 2 : Make group of 4-bits starting from LSB for integer part and MSB for fractional part by adding 0s at the end, if required.

Step 3 : Write equivalent octal number for each group to 4-bits.



2.3.6 Hexadecimal to Octal Conversion

The easiest way to convert hexadecimal number to octal number is given below :

1. Convert hexadecimal number to its binary equivalent.
2. Convert binary number to its octal equivalent.

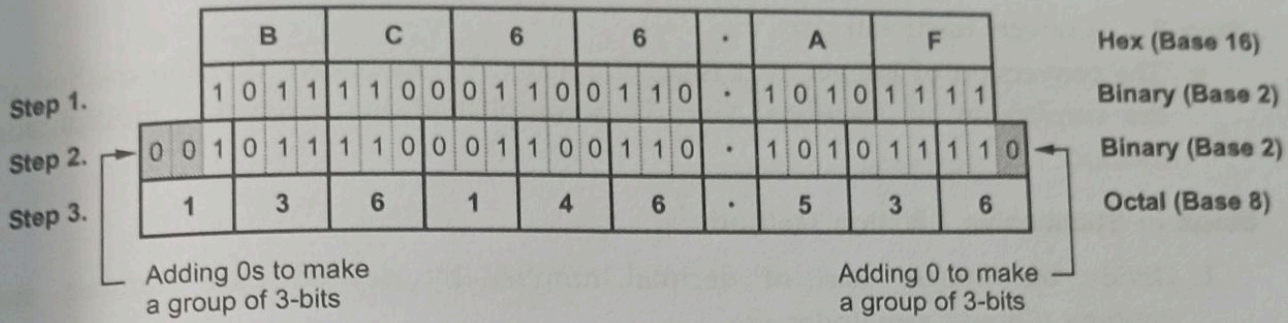
Example 2.3.6 *Convert $(BC66.AF)_{16}$ to its octal equivalent.*

Solution :

Step 1 : Write equivalent 4-bit binary number for each hexadecimal digit.

Step 2 : Make group of 3-bits starting from LSB for integer part and MSB for fractional part by adding 0s at the end, if required.

Step 3 : Write equivalent octal number for each group of 3-bits.



2.3.7 Converting Any Radix to Decimal

- In general, numbers can be represented as,

$$N = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \ldots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \ldots C_{-m}r^{-m}$$

where N = Number in decimal
A = Digit
r = Radix or base of a number system
n = The number of digits in the integer portion of number
m = The number of digits in the fractional portion of number

- From this general equation we can convert number with any radix into its decimal equivalent. This is illustrated using following example.

Example 2.3.7 *Convert $(3102.12)_4$ to its decimal equivalent.*

Solution : $N = 3\times4^3 + 1\times4^2 + 0\times4^1 + 2\times4^0 + 1\times4^{-1} + 2\times4^{-2}$

$= 192 + 16 + 0 + 2 + 0.25 + 0.125 = \mathbf{210.375_{10}}$

Example 2.3.8 *Determine the value of base x, if : $(193)_x = (623)_8$*

Solution : $(193)_x = (623)_8$

Converting octal into decimal : $6\times8^2 + 2\times8 + 3 = (403)_{10} = (623)_8$

$\therefore \quad (193)_x = 1\times x^2 + 9\times x + 3\times x^0 = (403)_{10}$

$\therefore \quad x^2 + 9x + 3 = 403$

$\therefore \quad x = 16 \text{ or } x = -25$

$\therefore \quad x = 16$

Negative is not applicable

$\therefore \quad (193)_{16} = \mathbf{(623)_8}$

2.3.8 Conversion of Decimal Number to Any Radix Number

Step 1 : Convert integer part.

Step 2 : Convert fractional part.

- The conversion of integer part is accomplished by successive division method and the conversion of fractional part is accomplished by successive multiplication method.

Steps in Successive Division Method

1. Divide the integer part of decimal number by desired base number, store quotient (Q) and remainder (R).
2. Consider quotient as a new decimal number and repeat step 1 until quotient becomes 0.
3. List the remainders in the reverse order.

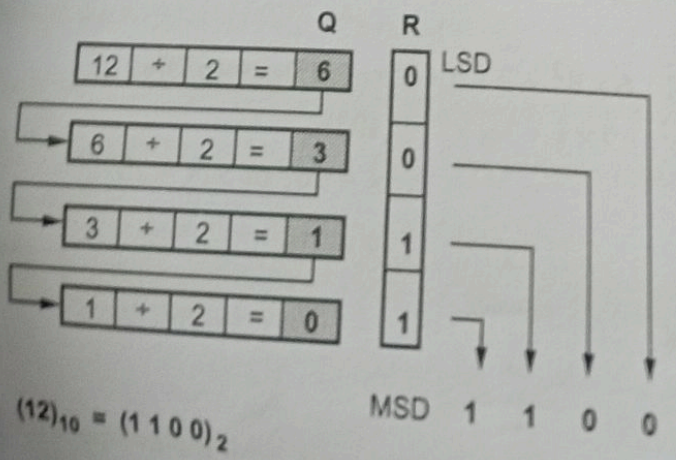
Steps in Successive Multiplication Method

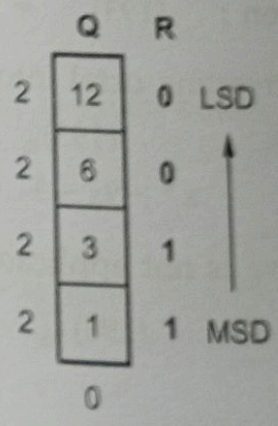
1. Multiply the fractional part of decimal number by desired base number.
2. Record the integer part of product as carry and fractional part as new fractional part.
3. Repeat steps 1 and 2 until fractional part of product becomes 0 or until you have many digits as necessary for your application.
4. Read carries downwords to get desired base number.

Example 2.3.9 *Convert 12.125 decimal into binary.*

Solution :

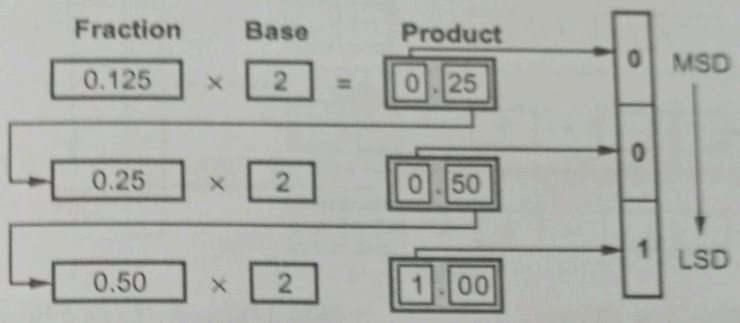
Integer part : Conversion of integer part by successive division method.





$(12)_{10} = (1\,1\,0\,0)_2$

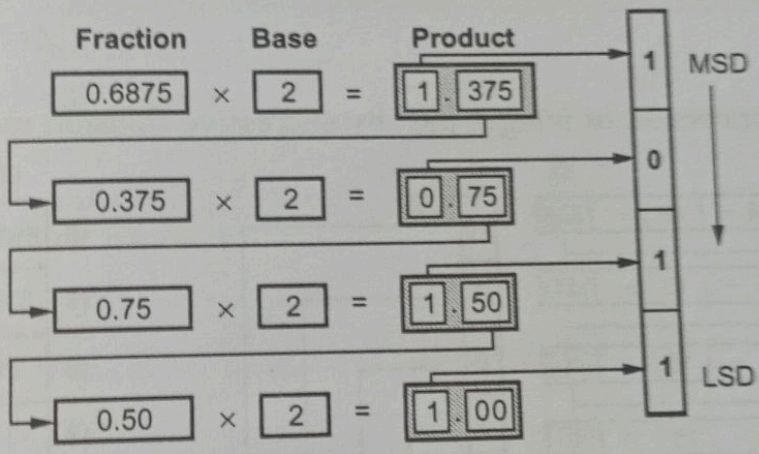
Fractional part : Conversion of fractional part by successive multiplication method.



$(0.125)_{10} = (0.001)_2$ $(12.125)_{10} = (1100.001)_2$

Example 2.3.10 *Convert $(0.6875)_{10}$ to binary.*

Solution :

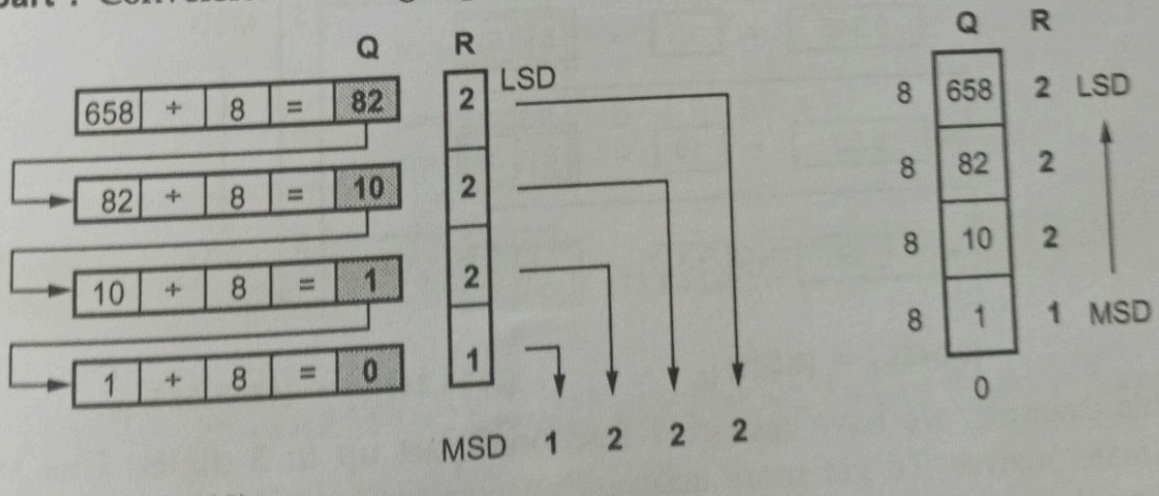


$\therefore (0.6875)_{10} = (0.1011)_2$

Example 2.3.11 *Convert 658.825 decimal into octal.*

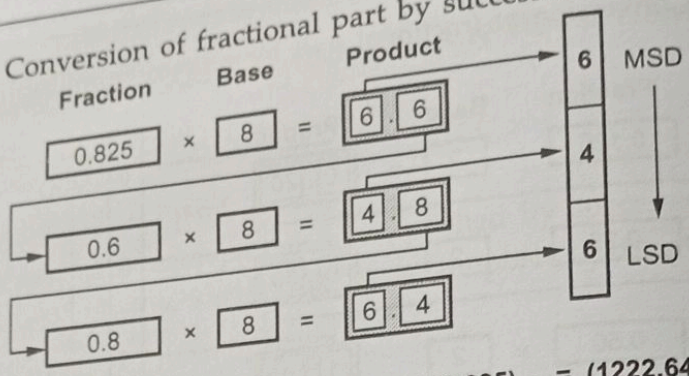
Solution :

Integer part : Conversion of integer part by successive division method.



$(658)_{10} = (1222)_8$

Fractional part : Conversion of fractional part by successive multiplication method.



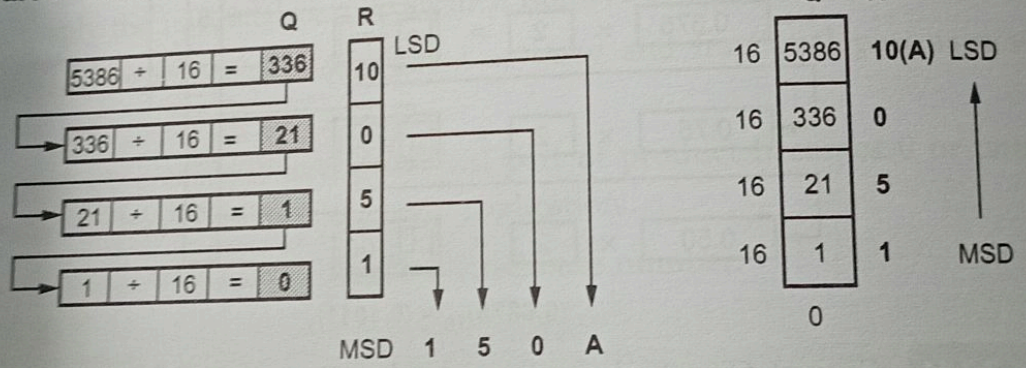
$(0.825)_{10} = (0.646)_8$ $\therefore (658.825)_{10} = (1222.646)_8$

In this example, we have restricted fractional part up to 3 digits. This answer is an approximate answer. To get more accurate answer we have to continue multiplying by 8 until we have as many digits as necessary for our application.

Example 2.3.12 *Convert 5386.345 decimal into hexadecimal.*

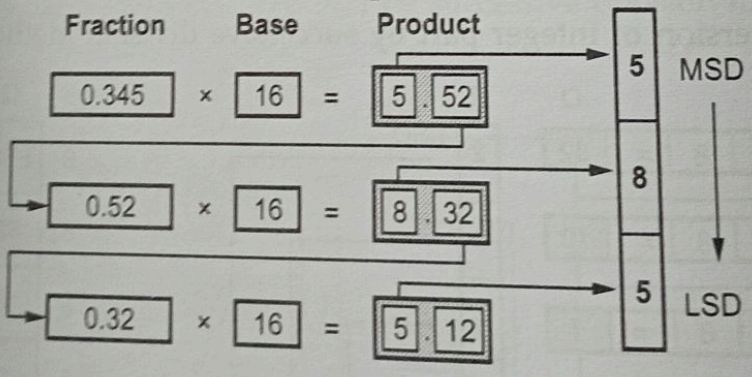
Solution :

Integer part : Conversion of integer part by successive division method.



$(5386)_{10} = (150A)_{16}$

Fractional part : Conversion of fractional part by successive multiplication method.



$(0.345)_{10} = (0.585)_{16}$ $\therefore (5386.345)_{10} = (150A.585)_{16}$

In this example, we have restricted fractional part up to 3 digits. This answer is an approximate answer. To get more accurate answer we have to continue multiplying by 16 until we have as many digits as necessary for our application.

Example 2.3.13 *The solution to the quadratic equation $x^2 - 11x + 22 = 0$ is $x = 3$ and $x = 6$. What is the base of the numbers ?*

Solution : Given quadratic equation is $x^2 - 11x + 22 = 0$ and it is given that $x = 3$ and $x = 6$

$$\therefore \quad (x - 3)(x - 6) = x^2 - 11x + 22 \qquad ...(1)$$

The numbers 3 and 6 are same in any base, whose value is more than 6.

i.e. $(3)_b = (3)_{10}$

$(6)_b = (6)_{10}$ where b = Base

Solving equation (1) we have,

$$(x^2 - 9x + 18)_{10} = (x^2 - 11x + 22)_b$$

Comparing coefficients of x we have

$$(-9)_{10} = (-11)_b$$

$$\therefore \quad (9)_{10} = (11)_b$$

$$\therefore \quad 9 = b^1 + b^0 = b + 1$$

$$\therefore \quad b = 8$$

or comparing constants we have,

$$(18)_{10} = (22)_b$$

$$\therefore \quad 18 = 2b + 2$$

$$\therefore \quad b = 8$$

Example 2.3.14 *Determine the value of b for the following :*

i) $(292)_{10} = (1204)_b$ ii) $(16)_{10} = (100)_b$

Solution : **i)** $(292)_{10} = 1 \times b^3 + 2 \times b^2 + 0 \times b^1 + 4 \times b^0$

$$292 = b^3 + 2b^2 + 4$$

$$\therefore \quad b = 6$$

ii) $(16)_{10} = 1 \times b^2 + 0 \times b^1 + 0 \times b^0 = b^2$

$$\therefore \quad b = 4$$

Example 2.3.15 *Convert $(101101.1101)_2$ to decimal and hexadecimal form.*

Solution :

$$D = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$

$$= 32 + 8 + 4 + 1 + 0.5 + 0.25 + 0.0625 = \mathbf{45.8125}$$

0	0	1	0	1	1	0	1	.	1	1	0	1	Binary number
2			D					.	D				Hex number

$$(101101.1101)_2 = (45.8125)_{10} = (2D.D)_{16}$$

Examples for Practice

Example 2.3.16 *Find the octal equivalent of the hexadecimal number DC.BA.* **[Ans. : $(334.564)_8$]**

Example 2.3.17 *Convert $(126)_{10}$ to octal number and binary number.* **[Ans. : $(176)_8$, $(1111110)_2$]**

Example 2.3.18 *Convert the $(153.513)_{10}$ to octal.* **[Ans. : $(231.4065)_8$]**

Example 2.3.19 *Perform the following code conversions :*
$(1010.10)_{16} \rightarrow (?)_2 \rightarrow (?)_8 \rightarrow (?)_{10}$
[Ans. : $(1000000010000.0001)_2$, $(10020.04)_8$, $(4112.0625)_{10}$]

Example 2.3.20 *Convert $(1947)_{10}$ into its equivalent octal and hexadecimal representations.* **[Ans. : $(1947)_{10} = (79B)_{16} = (3633)_8$]**

2.4 Complements

2.4.1 1's Complement Representation

- The 1's complement of a binary number is the number that results when we change all 1's to zeros and the zeros to ones.

Example 2.4.1 *Find 1's complement of $(11010100)_2$.*

Solution :

1	1	0	1	0	1	0	0	Number
▽	▽	▽	▽	▽	▽	▽	▽	NOT operation
0	0	1	0	1	0	1	1	1's complement of number

2.4.2 2's Complement Representation

- The 2's complement is the binary number that results when we add 1 to the 1's complement. It is given as,

2's complement = 1's complement + 1

Example 2.4.2 *Find 2's complement of* $(11000100)_2$.

Solution :

	1	1	0	0	0	1	0	0	Number
						1	1		Carry
	0	0	1	1	1	0	1	1	1's complement of number
+								1	Add 1
	0	0	1	1	1	1	0	0	2's complement of number

Example 2.4.3 *Represent the decimal number – 200 and 200 using 2's complement binary form.*

Solution : To represent 200 and – 200 we need 9-bit number system.

+200 = 0 1 1 0 0 1 0 0 0

–200 =	1	0	0	1	1	0	1	1	1	1's complement
+									1	Add 1
	1	0	0	1	1	1	0	0	0	2's complement

Fig. 2.4.1

2.5 Binary Fixed Point Representation

- In real life, we deal with real numbers - Numbers with fractional part. Most modern computer have native (hardware) support for **floating point** numbers. However, the use of floating point is not necessarily the only way to represent fractional numbers.
- The use of fixed point data type is used widely in Digital Signal Processing (DSP) and game applications, where performance is sometimes more important than precision.
- The key to represent fractional numbers in fixed point representation, is the concept of **binary point.** A binary point is like the decimal point in a decimal system. It acts as a divider between the integer and the fractional part of a number.
- In a decimal system, a decimal point denotes the position in a numeral that the coefficient should multiply by $10^0 = 1$. For example, in the numeral 36.5, the coefficient 6 has a weight of $10^0 = 1$. But what happen to the 5 to the right of decimal point ? We know that it carries a weight of 10^{-1}. We know the numeral "36.5" represents the value "thirty six and a half" because,

  $3*10^1 + 6*10^0 + 5*10^{-1} = 36.5$

- The very same concept of decimal point can be applied to our binary representation, making a **"binary point".** As in the decimal system, a binary point represents the coefficient of the term $2^0 = 1$. All digits (or bits) to the left of the

binary point carries a weight of $2^0, 2^1, 2^2$ and so on. Digits (or bits) on the right of binary point carries a weight of $2^{-1}, 2^{-2}, 2^{-3}$ and so on.

- For example, the number : 100100.1_2 represents the value :

2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	0	0	1	0	0	1	0	...

$$= 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 0*2^0 + 1*2^{-1}$$

$$= 32 + 4 + 0.5$$

$$= 36.5$$

- Consider the binary representation of number 73. It is 1001001. We can realize that the bit pattern of 73 and 36.5 is exactly the same. The **only difference**, is the position of binary point. In the case of 73_{10}, there is no binary point. Alternatively, we can say the binary point is located at the far right, at position 0. (Think in decimal 73 and 73.0 represents the same number.)
- In the case of 36.5_{10}, binary point is located one position to the left of 73_{10} :

2^6	2^5	2^4	2^3	2^2	2^1	2^0	DP	2^{-1}	
1	0	0	1	0	0	1	0	0	= 73

2^5	2^4	2^3	2^2	2^1	2^0	DP	2^{-1}	
1	0	0	1	0	0	.	1	= 73/2 = 36.5

- Thus, we can say that shifting binary number to the right by 1 bit position is equivalent to dividing the number by 2.
- In general, mathematically, given a fixed binary point position, shifting the bit pattern of a number to the right by 1 bit divide the number by 2. Similarly, shifting a number to the left by 1 bit multiplies the number by 2.
- To define a fixed point type conceptually, all we need are two parameters :
  - Width of the number representation and
  - Binary point position within the number.
- We will use the notation fixed<w,b>, where w denotes the number of bits used as a whole (the width of a number) and b denotes the position of binary point counting from the least significant bit (counting from 0).

- For example, fixed<8, 3> denotes a 8-bit fixed point number, of which 3 right most bits are fractional.
- Therefore, the bit pattern :

0	0	0	1	0	1	1	0

represents a real number : 00010.110_2

$$= 1*2^1 + 1*2^{-1} + 1*2^{-1}$$

$$= 2 + 0.5 + 0.25$$

$$= 2.75$$

- The fixed point numbers are indeed a close relative to integer representation. The two only differs in the position of binary point.
- In fact, you might even consider integer representation as a "special case" of fixed point numbers, where the binary point is at position 0. All the arithmetic operations a computer can operate on integer can therefore be applied to fixed point number as well.
- The fixed point arithmetic is much faster than floating point arithmetic.
- The disadvantage of fixed point number, is that of course the loss of range and precision when compare with floating point number representations.

Review Question

1. *Write a note on binary fixed point representation.*

Unit I
Multiple Choice Questions

Q.1 Which of the following is an output device ?

a Keyboard
b Mouse
c Light pen
d VDU

Q.2 Which of the following is an input device ?

a Plotter
b Printer
c VDU
d Mouse

Q.3 The central processing unit consists of _____.

a ALU
b ALU and control unit
c ALU and memory unit
d Memory unit and control unit

Q.4 _____ memory is a semiconductor memory.

a Hard disk
b Secondary memory
c Main memory
d Floppy disk

Q.5 In which of the following memory access time is not fixed ?

a ROM
b RAM
c Main memory
d Serial access memory

Q.6 _____ are used for solving complex application such as global weather forecasting.

a Super computers
b Portable notebook computers
c Mini computers
d Desktop computers

Q.7 They can operate on batteries and hence are very popular with travelers _____.

a mainframes
b laptops
c super computers
d desktop computers

Q.8 _____ **computers have large storage unit and faster communication links.**

a Mainframes
b Servers
c Super computers
d Desktop computers

Q.9 **Program Counter (PC) contains** _____.

a address of an instruction to be fetched
b instructions most recently fetched
c data to be written into memory
d data to be read from memory

Q.10 _____ **registers used to handle data transfer between processor and main memory.**

a MDR and MAR
b PC and MDR
c IR and MAR
d All of the above

Q.11 **The main virtue for using single bus structure is** _____.

a fast data transfers
b cost effective connectivity and speed
c cost effective connectivity and ease of attaching peripheral devices
d none of the mentioned

Q.12 **To extend the connectivity of the processor bus we use** _____.

a PCI bus
b SCSI bus
c controllers
d multiple bus

Q.13 **The components of the computer communicated with each other using** _____.

a memory
b CPU
c system bus
d data bus

Q.14 **PCI stands for** _____.

a Peripheral Component Interconnect
b Peripheral Computer Internet
c Processor Computer Interconnect
d Processor Cable Interconnect

Q.15 The current master uses _____ signal to indicate the start and duration of a transaction.

a FRAME
b LOCK
c IDSEL
d DEVSEL

Q.16 _____ signal is used as a chip select during configuration read and write transactions.

a FRAME
b LOCK
c IDSEL
d DEVSEL

Q.17 While CPU is executing a program, an interrupt exists then it _____.

a follows the next instruction in the program
b jumps to instruction in other registers
c breaks the normal sequence of execution of instructions
d stops executing the program

Q.18 If any interrupt request given to an input pin cannot be disabled by any means then the input pin is called ____.

a maskable interrupt
b nonmaskable interrupt
c vector interrupt
d none of the above

Q.19 The interrupt-request line is a part of the _____.

a data line
b control line
c address line
d none of the above

Q.20 The time between the receiver of an interrupt and its service is _____.

a interrupt delay
b interrupt latency
c cycle time
d switching time

Q.21 When an instruction is read from the memory, it is called ___.

a memory read cycle
b fetch cycle
c instruction cycle
d memory write cycle

Q.22 The number of bits in a nibble is _____.

a 16 b 5

c 4 d 8

Q.23 The largest value of a digit in any number system _____.

a base value b base value - 1

c base value + 1 d base value / 2

Q.24 What is the base of duodecimal number ?

a 8 b 10

c 12 d 16

Q.25 The number of bits required to encode 90 pieces of information is_____.

a 4 b 5

c 6 d 7

Q.26 The decimal number 127 can be represented by_____.

a 1111 1111B b 1000 0000B

c EEH d 0111 1111

Q.27 The hexadecimal equivalent of $(2768)_{10}$ is _____.

a $(AD0)_{16}$ b $(0DA)_{16}$

c $(10131)_{16}$ d $(11310)_{16}$

Q.28 If $(12A8C)_{16} = (X)_8$ then value of X is_____.

a 225214 b 125214

c 225114 d 325214

Q.29 A 0 in the sign bit represents a _____.

a positive number b negative number

c unsigned number d none of these

Q.30 Two's complement representation of - 8 is_____.

a 0111 b 1000

c 01000 d 10100

Q.31 What will be the 1's complement of binary number $(01110001)_2$?

a $(10001110)_2$ b $(10000111)_2$

c $(10001101)_2$ d $(10001111)_2$

Answer Keys for Multiple Choice Questions :

Q.1	d	Q.2	d	Q.3	b	Q.4	c
Q.5	d	Q.6	a	Q.7	b	Q.8	b
Q.9	a	Q.10	a	Q.11	c	Q.12	a
Q.13	c	Q.14	a	Q.15	a	Q.16	c
Q.17	c	Q.18	c	Q.19	b	Q.20	b
Q.21	b	Q.22	c	Q.23	b	Q.24	c
Q.25	d	Q.26	d	Q.27	a	Q.28	a
Q.29	a	Q.30	b	Q.31	a		

❑❑❑