CHAPTER ONE

Introduction to computer

What is computer?

A **computer** is an electronic machine, operating under the control of instructions stored in its own memory that can accept data, manipulate the data according to specified rules, produce results, and store the results for future use. Computers **process** ***data*** to create information. **Data** is a collection of raw or unprocessed facts, figures, and symbols. **Information** is data that is organized, meaningful, and useful. To process data into information, a computer uses hardware and software. **Hardware** is the electric, electronic, and mechanical equipment that makes up a computer. **Software** is the series of instructions that tells the hardware how to perform tasks.

Characteristics of computer

The characteristics of computers that have made them so powerful and universally useful are speed, accuracy, diligence, versatility and storage capacity. Let us discuss them briefly.

1. **Speed**: - Computers work at an incredible speed. A powerful computer is capable of performing about 3-4 million simple instructions per second. Their speed is measured by the amount of time it took to perform or carry out a basic operation. Computer speed measured in terms of microsecond (10^{-6} one millionths), nanosecond (10^{-9} one billionths), and Pico second (10^{-12} one trillionths).
2. **Accuracy**: - ***Nowadays*** computers are being used for surgical purposes, which need almost a hundred percent accuracy. From this we can understand that computers are accurate and consistent. Unless there is an error in the input data or unreliable program the computer processes with a very high accuracy.
3. **Diligence:-** Unlike human beings, computers are highly consistent. They do not suffer from human traits of boredom and tiredness resulting in lack of concentration. Computers, therefore, are better than human beings in performing voluminous and repetitive jobs.
4. **Versatility:-** Computers are versatile machines and are capable of performing any task as long as it can be broken down into a series of logical steps. The presence of computers can be seen in almost every sphere – Railway/Air reservation, Banks, Hotels, Weather forecasting and many more.
5. **Automatic:-** Once necessary information and program is fed, the computer performs processing without human intervention.
6. **Storage capacity:-** Today's computers can store large volumes of data. A piece of information once recorded (or stored) in the computer, can never be forgotten and can be retrieved almost instantaneously. And the time it took to retrieve or process single information is not more than a micro or nanoseconds.

In general a computer has a capacity to store a very large amount of information in organized manner so that accessing information is very fast.

Types of Computers

Computers can be classified into different categories based on different characteristics.

- **Based on type of data they process**

  Based on the type of data they process computers can be classified as:

Analog

Analog computers operate by measuring physical properties. They deal with continuous variables; they don't compute directly with numbers, rather, they operate by measuring physical magnitude such as pressure, temperature, voltage, current etc.

Examples: Thermometer, Voltmeter, Speedometer

Digital

Digital computers deal with discrete variables; they operate by counting rather than measuring. They operate directly up on numbers (or digits) that represent numbers, letters, or other special symbols.

Examples: Abacus, Desk & pocket calculators, general purpose computers

Hybrid

Hybrid computers inherit the best features of both analog and digital computers. Usually the Input is continuous data (analog). Since Digital Processing is more accurate, processing takes place digitally. The processed information – the output – could be either digital or analog, depending on the user preference or the type of application.

Examples: digital camera, health monitoring machines in some hospitals,

- **Based on Size, Capacity and price**

Size and capacity are also the other characteristics of computers that can be used to categorize computers. Based on these characteristics computers can be classified as:

Super computer

The term supercomputer has been coined to describe a category of extremely powerful computer designed for high-speed processing. A supercomputer is generally characterized as being the fastest, most powerful, and most expensive computer.

Generally, Supercomputers are:

- The largest and the most efficient computers
- Very expensive
- very fast and
- Supports hundreds of users at different locations

Mainframe computer

Mainframe computers are large, powerful computers that are physically larger than micros and minis and usually have processors with faster instruction processing speeds. For example, they may be able to process from 10 to 200 million instructions per second (MIPS). Mainframe computers also support multiple users and are expensive.

Mini computer

Minicomputers are midrange computers that are larger and more powerful than most microcomputers but are smaller and less powerful than mainframe computer systems. Minicomputers are being used for a large number of business and scientific applications. They are popularly used in scientific laboratories, research centers, universities and colleges, engineering firms, industrial process monitoring and control, etc.

Micro computers

The smallest computers ever produced in the history of computers are microcomputers. Since they are designed to be used by a single user, they have the least capacity as compared to the other types of computers. They are also the least expensive of all types. There two different types of microcomputers are desktop computers and portable computers (laptops, notebook computers and palmtops)

Organization of a computer system

Generally, a computer system is composed of two main components:

1. Computer hardware and
2. Computer software

Computer Hardware

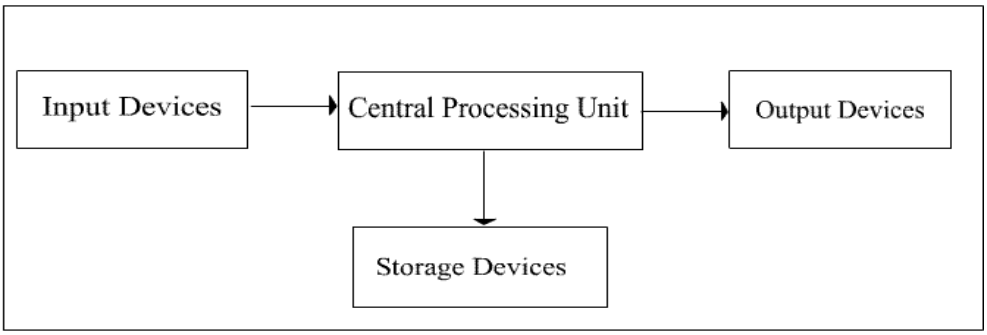
Computer hardware is the physical part of the computer system that can be seen and felt. The hardware part of a computer system is composed of a number of interacted physical parts. E.g. keyboard, mouse, CPU

1.1 Types of Computer Hardware

The hardware part of a computer system is composed of a number of interacting physical parts based on the need of the information flow. Information flows in the computer hardware. There are several criteria by which computer hardware can be categorized.

Based on **information processing**, we can divide **computer hardware** into **four**:

i) Input Devices
ii) Storage Devices
iii) Output Devices
iv) Central Processing Unit (CPU)



i) Input Devices

Input devices are used to **enter information** into **computer**. They **convert** the **data** we give them into the form that can be **manipulated** in the **computer** (**electronic format**).

Some examples of input devices are Keyboard, mouse, scanner**,** Bar Code Reader, mice

ii) Central Processing Unit

CPU executes instructions and performs the computer's processing activities. It is also known as processor or microprocessor. It functions the same purpose as the human brain for human being. It is called the brain of the computer.

One of the basic features of a computer that affects its entire performance is the CPU speed. CPU speed is measured in Hertz (Hz). Hertz is the number of **cycles per second**. 1Hz=1cycle per second. Larger units are KHz (Kilo Hertz), MHz (Mega Hertz), GHz (Giga Hertz), etc.

1 KHz = 1000 Hz

1 MHz = 1000 KHz

1 GHz = 1000MHZ

Current CPUs are as fast as 2-3GHz (2-3 billion cycles per second)

CPU has three sub-components:

- ✓ Control Unit (CU)
- ✓ Arithmetic Logic Unit (ALU)

Control Unit

As human brain controls the body, control unit controls the computer hardware. Control Unit does not execute instruction by itself, i.e. does not carry out instruction processing, but it directs other processing elements to execute instructions.

Arithmetic Logic Unit (ALU)

The purpose of ALU is to execute instruction. It performs two operations:

- ✓ Arithmetic operation
- ✓ Logic Operation

Arithmetic operation*:* this includes mathematical operations like addition, subtraction, multiplication, division, etc. If you give your computer the instruction *2+3*, this will be included in arithmetic operation and it is executed by Arithmetic Unit.

Logical Operation*:* this is concerned with the comparison of data and it is called logical operation. It includes operators like less than, greater than, equal to, less or equal to, greater or equal to, different from, etc. e.g. if mark>80, grade is 'A'.

iii) Output Devices

Output devices are used to get data out of a computer so that it can be examined, analyzed or distributed to others. It converts information from machine-understandable form to a human understandable form. The outputs are of two types: *Softcopy*: displayed on monitor, projector, or similar devices and *Hardcopy*: printed on paper

Examples

- ✓ The Visual Display Unit (VDU) or monitor or screen
- ✓ Printers (dot matrix, daisy wheel, laser printers)
- ✓ Plotters
- ✓ Voice (audio) response unit
- ✓ Disk drives

iv) Storage Devices

One of the unique features of computers is storage. Data can be stored on different storage media temporarily or permanently. Storage devices can be categorized into to as:

1. Primary storage device

2. Secondary storage device

1. Primary Memory / Main Memory

Primary memory, also called *Main memory*, refers to integrated circuit that stores program instructions and data. The CPU closely works with the main memory to perform its activities. Memory stores three things:

- ✓ Operation system software instructions
- ✓ Application software instruction
- ✓ Data that is being processed

Depending on the type of information they store and the technology used, the primary memory can be categorized into three:

- ✓ RAM (Random Access Memory)
- ✓ ROM (Read Only Memory)

RAM is temporary storage i.e. the data is lost when the computer is off unlike secondary storage. Because of this it is called volatile memory.

Why is it volatile? It uses electric power to store data. When you write anything on your computer, first it is stored on RAM. When you save the file, it is transferred into secondary storage.

RAM has differing capacity, the common ones being 128, 256, and 512.It is directly accessible by CPU. It is called RAM because each memory location can be accessed randomly using memory address. Each unit in RAM has memory address by which it can be easily accessed/referenced.

ROM stores data and programs that are permanently required by the computer. They have programs built into them at the factory and that program could not be changed or erased by the user, but read. It is non-volatile, read-only (not changeable) memory. Read-only means data can't be altered or erased but read.

2. Secondary Storage

Secondary storage (also called auxiliary storage) supplements the primary memory. It takes many forms. It includes punched cards, punched paper tape, magnetic tape, magnetic disk and optical disk.

Computer Software

Computer hardware is directed by a set of instructions. Without these instructions, computers can do nothing. These set of instructions are called software (also called programs). One or more programs are termed as **software**. We use programming languages to write these instructions. Examples of programming languages include C, C++, Java, etc.

Software is categorized into two:

- System Software
- Application Software

- **System Software**

  System software consists of programs that are related to controlling the actual operations of the computer equipment/resource.

There are three types of system software:

- ✓ Operating System
- ✓ Utility Software
- ✓ Language translators

Operating system manage resources, provides a user interface, and run application software. It organizes resources such as keyboard, mouse, printer, monitor, etc. It also presents GUI (Graphical User Interface) to the user for easy use of computer. It makes complex hardware more users friendly i.e. it acts between the user and hardware. Operating system coordinates the activity between the user and the computer

Utilities software is programs that make computing easier. They perform specific tasks related to managing computer resources or files. There are different utility programs:

i) **Troubleshooting programs**: enable us to recognize and correct computer problems before they become serious.

ii) **Anti-virus programs**: they protect your computer against viruses or other malicious programs that damage computer.

iii) **File compression programs**: are used to reduce the size of files or data so that it takes less storage space or network band. E.g WinZip, WinRAR, etc.

iv) **Uninstall programs**: this software enable us to safely and completely remove unneeded programs/software from your computer.

v) **Back up software**: with the help of this software, we can make copies of files to be used in case of the original data is lost/damaged. This copy is called back up**.**

vi) **Screen savers**: helps to prevent your work from being seen by others if you leave your computer idle for some time.

Language translators are used to **convert the programming instruction written by users into binary code** that the computer can understand. They are **written for specific programming languages** and **computer system**.

Application Software

Application software performs **useful work** for the **user**. These useful works could be:

- ✓ Word processing-document creation
- ✓ Spreadsheet-electronic calculation
- ✓ Data base system
- ✓ Email/communicating-email sending and reading

Users use this software to perform different activities like calculation, video editing, word processing, presentation, etc.

DATA REPRESENTATION IN COMPUTER SYSTEM

Data Representation

Every computer stores numbers, letters, & other special characters in a coded form. But a computer stores those data's in a binary number system. There are two main code forms that every number, letter and special character is represented in a computer system. Before discussing those codes let have a look at binary number system.

The Binary Number System

Computers are electronic devices that use electrical patterns to represent numbers. Modern digital computers recognize only two electrical states—ON and OFF— but their memories contain millions of transistors that can be either on or off. Working with numbers in computers is like making numbers out of a row of lights that can be switched on and off independently. Because there are only two ways to represent a number, computers use the binary number system (base 2 or radix 2).

A relationship among Binary & other number system

Decimal	Hexadecimal	Octal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000

9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111
16	10	20	0001 0000

1.1.1 Why Binary?

Why do we go for binary numbers instead of decimal numbers?' The reasons are as follows:

1. The 1st reason is that the **electronic & electrical components, by their very nature, operate in a binary mode.** Information is handled in the computer by electronic/electrical components such as transistors, semiconductors, wires, etc all of which can only indicate 2 states or conditions – on(1) or off(0).
   Transistors are either conducting (1) or non conducting (0); a voltage is present (1) or absent (0) in wire. The binary number system, which has only two digits (0&1), is most suitable for expressing the two possible states

2. The second reason is that the **computer circuits only have to handle two binary digits** rather than ten decimal digits. This greatly simplifies the internal circuit design of computers, resulting in less expensive & more reliable circuits.

3. Finally, the binary system is used because everything that can be done in decimal number system (**addition, subtraction, division & multiplication**) can also be done in binary number system.

1.2 Units of Data Representation

When data is stored, processed, or communicated within the computer system, it is "packed" in units. Arranged from the smallest to the largest, the units are called bits, bytes, & words.

Bits, Bytes and Words

Bits

A ***bit*** is a single binary digit. A bit may have a value of 0 or 1. In a computer, a switch or transistor that is **off** represents a **0**, and a switch or transistor that is **on** represents a **1**. A bit is represented by the numbers 1, & 0, which correspond to the states **on & off, true & false, or yes & no.**

Bytes

Most computers work with groups of **8 bits**, which is called **a *byte*.** To make it easier to read, the 8 binary digits in a byte are divided into two groups of four, called *nibbles*, when they are written.

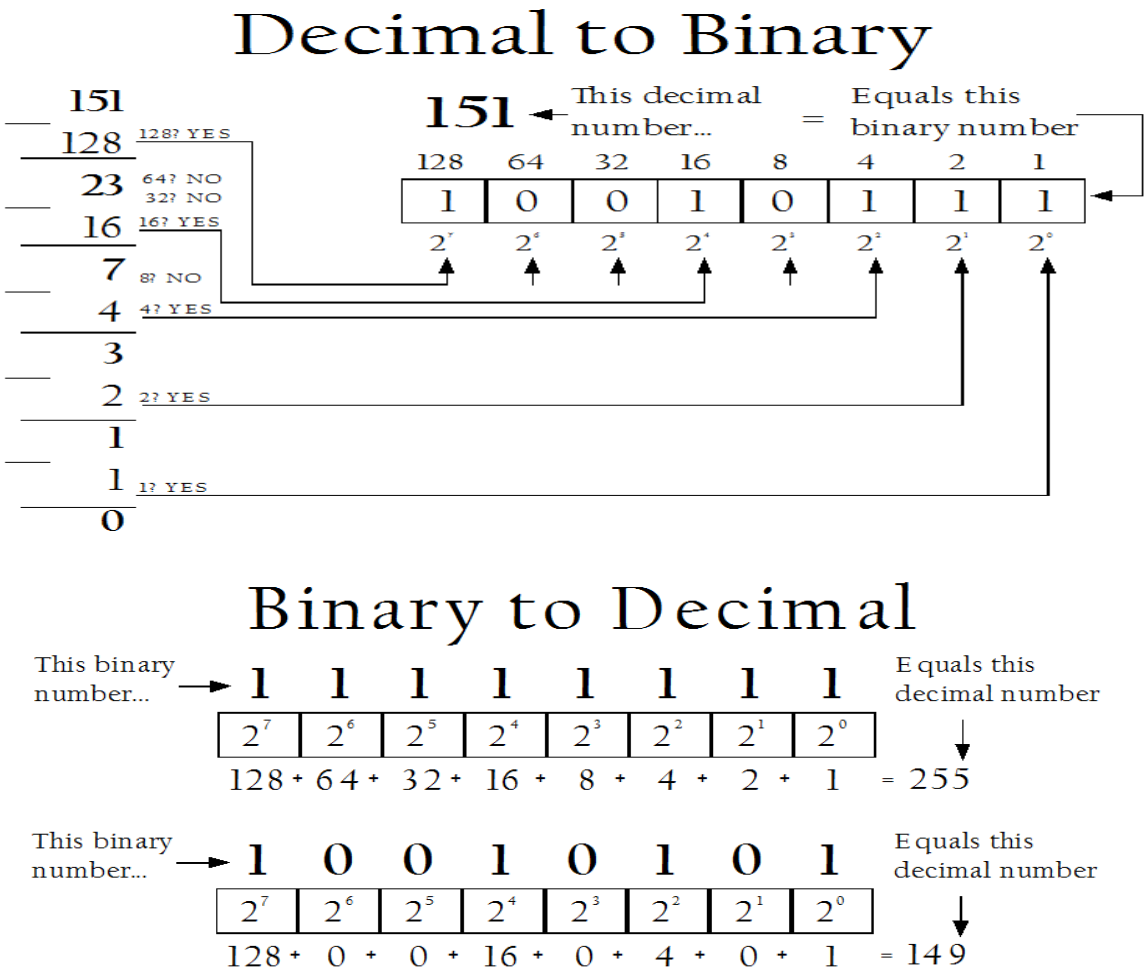
One byte may hold binary numbers ranging in value from 0000 0000 (base 2) to 1111 1111 (base 2), or from 0 (base 10) to 255 (base 10). Counting 0 as a value, one byte can contain 256 values. For many computer variables, the maximum value is 255 because the computer wants to store the value in a single byte, or you are limited to 256 choices.

Words

- Bytes are combined into groups of 1 to 8 bytes called words.
- Words refer to the number of bits that a computer process at once.
- Typically word lengths are 8 bits, 16 bits, 32 bits & 64 bits.

Computer Number systems

A decimal integer is converted to any other base, by using the division operation.



Hexadecimal number system

- Computers use hexadecimal because it represents binary values in a compact form.
- Base 16 number system
- *Hex* means six, *Deci* means ten
- 16 symbols are used - 0 thru 9, A, B, C, D, E, F
    - 0-9 = 0-9
    - A = 10 D = 13
    - B = 11 E = 14
    - C = 12 F = 15

Conversion of decimal to hexadecimal

- Reverse the process
- 59 is 3B in hexadecimal
- 16 goes into 59 three times
- It will take "11" or "B" to get to 59 (59 = 3*16+11) =3 * B (11)

Example: Convert the decimal number 90 into hexadecimal number

Solution:

90 = (5 * 16) +10 this implies 5A because the number 10 is represented by A in hexadecimal.

90 dividing by 16 and collecting the remainder of the digit we can verify it.

☞ A binary number can be converted into octal or hexadecimal number using a shortcut method. The shortcut method is based on the following information:-

- An octal digit from 0 to 7 can be represented as a combination ***of 3 bits***, since $2^3 = 8$.
- A hexadecimal digit from 0 to 15 can be represented as a combination ***of 4 bits,*** since $2^4 = 16$.

1.3 Decimal to Other Base System

- **Steps 1** - Divide the decimal number to be converted by the value of the new base.
- **Step 2** - Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
- **Step 3** - Divide the quotient of the previous divide by the new base.
- **Step 4** - Record the remainder from Step 3 as the next digit (to the left) of the new base number.

Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.

The last remainder thus obtained will be the most significant digit (MSD) of the new base number.

1.3.1 Example

Decimal Number: 29_{10}

Calculating Binary Equivalent:

Step	Operation	Result	Remainder
Step 1	29 / 2	14	1
Step 2	14 / 2	7	0
Step 3	7 / 2	3	1
Step 4	3 / 2	1	1
Step 5	1 / 2	0	1

As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the least significant digit (LSD) and the last remainder becomes the most significant digit (MSD).

Decimal Number: 29_{10} = Binary Number: 11101_2.

1.4 Other base system to Decimal System

Steps

- **Steps 1** - Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
- **Step 2** - Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
- **Step 3** - Sum the products calculated in Step 2. The total is the equivalent value in decimal.

1.4.1 Example

Binary Number: 11101_2

Calculating Decimal Equivalent:

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	11101_2	$(16 + 8 + 4 + 0 + 1)_{10}$
Step 3	11101_2	29_{10}

Binary Number: 11101_2 = Decimal Number: 29_{10}

1.5 Other Base System to Non-Decimal System

Steps

- **Steps 1** - Convert the original number to a decimal number (base 10).
- **Step 2** - Convert the decimal number so obtained to the new base number.

1.5.1 Example

Octal Number: 25_8

Calculating Binary Equivalent:

1.5.2 Step 1: Convert to Decimal

Step	Octal Number	Decimal Number
Step 1	25_8	$((2 \times 8^1) + (5 \times 8^0))_{10}$
Step 2	25_8	$(16 + 5)_{10}$
Step 3	25_8	21_{10}

Octal Number: 25_8 = Decimal Number: 21_{10}

1.5.3 Step 2: Convert Decimal to Binary

Step	Operation	Result	Remainder
Step 1	21 / 2	10	1

Step 2	10 / 2	5	0
Step 3	5 / 2	2	1
Step 4	2 / 2	1	0
Step 5	1 / 2	0	1

Decimal Number: 21_{10} = Binary Number: 10101_2

Octal Number: 25_8 = Binary Number: 10101_2

- **Shortcut method - Binary to Octal**

Steps

- **Step 1** - Divide the binary digits into groups *of three* (starting from the right).
- **Step 2** - Convert each group of three binary digits to one octal digit.

1.5.4 Example

Convert Binary Number: 10101_2 ***to Octal Or***

Calculating ***Octal Equivalent:***

Step	Binary Number	Octal Number
Step 1	10101_2	010 101
Step 2	10101_2	2_8 5_8
Step 3	10101_2	25_8

Binary Number: 10101_2 = Octal Number: 25_8

1.6 Binary to Octal

An easy way to convert from binary to octal is to group binary digits into sets of three, starting with the least significant (rightmost) digits.

Binary: 11100101 = 11 100 101

011 100 101 Pad the most significant digits with zeros if necessary to complete a group of three.

Then, look up each group in a table:

Binary: 000 001 010 011 100 101 110 111

Octal: 0 1 2 3 4 5 6 7

Binary = 011 100 101

Octal = 3 4 5 = 345 oct

- **Shortcut method - Octal to Binary**

Steps

- **Steps 1** - Convert each octal digit to a 3-digit binary number (the octal digits may be treated as decimal for this conversion).
- **Step 2** - Combine all the resulting binary groups (of 3 digits each) into a single binary number.

1.6.1 Example

Octal Number: 25_8

Calculating Binary Equivalent:

Step	Octal Number	Binary Number
Step 1	25_8	$2_{10}\ 5_{10}$
Step 2	25_8	$010_2\ 101_2$
Step 3	25_8	010101_2

Octal Number: 25_8 = Binary Number: 10101_2

Converting from octal to binary is as easy as converting from binary to octal. Simply look up each octal digit to obtain the equivalent group ***of three binary digits***.

Octal: 0 1 2 3 4 5 6 7

Binary: 000 001 010 011 100 101 110 111

Octal = 3 4 5

Binary = 011 100 101 = 011100101 binary

- **Shortcut method - Binary to Hexadecimal**

Steps

- **Step 1** - Divide the binary digits into groups of four (starting from the right).
- **Step 2** - Convert each group of four binary digits to one hexadecimal symbol.

1.6.2 Example

Binary Number: 10101_2

Calculating hexadecimal Equivalent:

Step	Binary Number	Hexadecimal Number
Step 1	10101_2	0001 0101
Step 2	10101_2	1_{10} 5_{10}
Step 3	10101_2	15_{16}

Binary Number: 10101_2 = Hexadecimal Number: 15_{16}

1.7 Binary to Hexadecimal

An equally easy way to convert from binary to hexadecimal is to group binary digits into sets of four, starting with the least significant (rightmost) digits.

Binary: 11100101 = 1110 0101

Then, look up each group in a table:

Binary: 0000 0001 0010 0011 0100 0101 0110 0111

Hexadecimal: 0 1 2 3 4 5 6 7

Binary: 1000 1001 1010 1011 1100 1101 1110 1111

Hexadecimal: 8 9 A B C D E F

Binary = 1110 0101

Hexadecimal = E 5 = E5 hex

Example

Convert the following binary numbers into decimal numbers.

* Conversion of binary number 1010101000010111to hexadecimal number is - $AA17_{16}$

1.8 Shortcut method - Hexadecimal to Binary

Steps

- **Steps 1** - Convert each hexadecimal digit to a 4-digit binary number (the hexadecimal digits may be treated as decimal for this conversion).
- **Step 2** - Combine all the resulting binary groups (of 4 digits each) into a single binary number.

1.8.1 Example

Hexadecimal Number: 15_{16}

Calculating Binary Equivalent:

Step	Hexadecimal Number	Binary Number
Step 1	15_{16}	1_{10} 5_{10}
Step 2	15_{16}	0001_2 0101_2
Step 3	15_{16}	00010101_2

Hexadecimal Number: 15_{16} = Binary Number: 10101_2

Converting from hexadecimal to binary is as easy as converting from binary to hexadecimal. Simply look up each hexadecimal digit to obtain the equivalent group of four binary digits.

Hexadecimal: 0 1 2 3 4 5 6 7

Binary: 0000 0001 0010 0011 0100 0101 0110 0111

Hexadecimal: 8 9 A B C D E F

Binary: 1000 1001 1010 1011 1100 1101 1110 1111

Hexadecimal = A 2 D E

Binary = 1010 0010 1101 1110 = 1010001011011110 binary

1.9 Octal to Hexadecimal

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal. For example, to convert 345 octal into hex:

(From the previous example)

Octal = 3 4 5

Binary = 011 100 101 = 011100101 binary

Drop any leading zeros or pad with leading zeros to get groups of four binary digits (bits): Binary 011100101 = 1110 0101

Then, look up the groups in a table to convert to hexadecimal digits.

Binary: 0000 0001 0010 0011 0100 0101 0110 0111

Hexadecimal: 0 1 2 3 4 5 6 7

Binary: 1000 1001 1010 1011 1100 1101 1110 1111

Hexadecimal: 8 9 A B C D E F

Binary = 1110 0101

Hexadecimal = E 5 = E5 hex

Therefore, through a two-step conversion process, octal 345 equals binary 011100101 equals hexadecimal E5.

1.10 Hexadecimal to Octal

When converting from hexadecimal to octal, it is often easier to first convert the hexadecimal number into binary and then from binary into octal. For example, to convert A2DE hex into octal:

(From the previous example)

Hexadecimal = A 2 D E

Binary = 1010 0010 1101 1110 = 1010001011011110 binary

Add leading zeros or remove leading zeros to group into sets of three binary digits.

Binary: 1010001011011110 = 001 010 001 011 011 110

Then, look up each group in a table:

Binary: 000 001 010 011 100 101 110 111

Octal: 0 1 2 3 4 5 6 7

Binary = 001 010 001 011 011 110

Octal = 1 2 1 3 3 6 = 121336 octal

Therefore, through a two-step conversion process, hexadecimal A2DE equals binary 1010001011011110 equals octal 121336.

1.11 Hexadecimal to Decimal

Converting hexadecimal to decimal can be performed in the conventional mathematical way, by showing each digit place as an increasing power of 16. Of course, hexadecimal letter values need to be converted to decimal values before performing the math.

Hexadecimal: 0 1 2 3 4 5 6 7

Decimal: 0 1 2 3 4 5 6 7

Hexadecimal: 8 9 A B C D E F

Decimal: 8 9 10 11 12 13 14 15

= ((A) * 16^3) + (2 * 16^2) + ((D) * 16^1) + ((E) * 16^0)
= (10 * 16^3) + (2 * 16^2) + (13 * 16^1) + (14 * 16^0)
= (10 * 4096) + (2 * 256) + (13 * 16) + (14 * 1)
= 40960 + 512 + 208 + 14
= 41694 decimal

1.12 Decimal to Hexadecimal

Here is an example of using repeated division to convert 1792 decimal to hexadecimal:

Decimal Number	Operation	Quotient	Remainder	Hexadecimal Result
1792	÷ 16 =	112	0	0
112	÷ 16 =	7	0	00
7	÷ 16 =	0	7	700
0	Done.			

The only addition to the algorithm when converting from decimal to hexadecimal is that a table must be used to obtain the hexadecimal digit if the remainder is greater than decimal l 9.

Decimal: 0 1 2 3 4 5 6 7

Hexadecimal: 0 1 2 3 4 5 6 7

Decimal: 8 9 10 11 12 13 14 15

Hexadecimal: 8 9 A B C D E F

Example:

Understanding Kilobytes, Megabytes, and Gigabytes

A kilobyte is often referred to as 1,000 bytes, but this is not totally accurate. In computer terms, a *kilobyte* is 1,024 bytes. K is used as shorthand for 2^{10}, which equals 1,024.

A *megabyte* is not exactly 1,000,000 bytes, but 2^{20} bytes, or 1,024 KB, or 1,048,576 bytes. A 200 MB drive can store 204,800 KB or 209,715,200 bytes.

A *gigabyte* is not 1 billion bytes, but 2^{30} bytes or 1,073,741,824 bytes. 3 GB is the same as 3,221,225,472 bytes.

For Example

2^{10} is 1024 ≈1000(or 1KB)

- 1KB (Kilo bytes)-1024 bytes-can store 1,000 characters.
- 1MB = 2^{20} bytes ≈10^6 bytes- can store 1 Million characters
- 1GB = 2^{30} bytes ≈ 10^9 bytes – can store 1 Billion characters.
- 1TB= 2^{40} bytes ≈ 10^{12}bytes- can store 1 Trillion characters.

* A computer having 256MB of memory is capable of storing 256 * 1048576(2^{20}Mega) bytes.
* A floppy disk holds 1.44 MB of data ≈ 1, 400, 000 characters.

Programming and Problem Solving

1.1. Programming Languages

Programming Language is a set of rules, symbols, and special words used to construct a computer program. Programming language rules consists of:

- ✓ Rules of Syntax which specify how valid instructions are written in the language.
- ✓Rules of Semantics which determine the meaning of the instructions (what the computer will do).

Programming languages are categorized into *two main types*, these are:

1) Low level languages which in turn include
   I. Machine languages and
   II. Assembly languages
2) High level languages

Machine Language: is a low level language in which its instructions are string of binary representation that a computer's hardware can understand and perform.

e.g. 100100 010001

Assembly Language: it is also a low-level programming language in which a mnemonic or a symbol is used to represent each of the machine language instructions for a specific computer. Assembly language programs also allow the user to use text names for data rather than having to remember their memory addresses. *The Assembly language is translated into the machine language through an **Assembler***.

Table 1.1: E

Machine Language	Assembly Language
100101	ADD
011001	SUB
001101	MPY
100111	CMP
100011	JMP
110011	JNZ

High Level Language: The codes in high-level languages are similar to everyday English, closer to standard notations and it uses ordinary mathematical notations. e.g. C++, Basic, Fortran, Cobol. The High Level Language is translated to Machine language through Compilers.

Assembler Language	C++
LDA X ADD Y SUB Z STO A	a = x + y - z;

Table 1.2 Examples of Assembly language and High level language

1.2. Program Development

Program development involves **creating** and **executing** a program. To create and execute a program, three environments need to be invoked:

- ✓ The Editor environment, to create the program source code
- ✓ The Compilation environment, to convert a source program into a machine language
- ✓ The Execution environment, to run the program

A program consists of a set of instructions written by the programmer. Normally a high level language (such as C, C++, Pascal, FORTRAN, …) is used to create a plain-text source code.

1.3. Problem Solving Approach

Programming is a process of problem solving that consists of several steps, which include design, creation, testing and debugging activities. Below are listed the steps recommended for the process of writing a program.

1. Clearly define the problem.
2. Analyze the problem and formulate a method to solve it
3. Describe the solution in the form of an algorithm.
4. Draw a flowchart or a pseudocode for the algorithm.
5. Write the computer program.
6. Compile and run the program (debugging).
7. Test the program (debugging)
8. Interpret the results.

1.4. Verification and Validation

Every program should be tested to make sure that it does what the programmer intends it to do and that it actually provides a valid solution to the problem. These tests are commonly divided as:

Verification: verify that program does what you intended it to do; steps 7(8) above attempt to do this.

Validation: answers the question "does the program actually solve the original problem?" i.e. is it valid? This goes back to steps 1 and 2 - if you get these steps wrong then your program is not a valid solution.

1.5. Algorithm

An algorithm is a procedure for solving a problem in terms of the actions to be executed and the order in which those actions are to be executed. An algorithm is merely the sequence of steps taken to solve a problem. There are two commonly used tools to help to document an algorithm. These are flowcharts and Pseudocode. We will use both methods here. Generally, flowcharts work well for small problems but Pseudocode is used for larger problems.

1. Pseudocode

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is similar to natural language, yet close enough to programming language that it can be easily converted later into program source code. By writing the algorithm in pseudocode first, the programmer can focus on just the logical steps the program must perform, without having to worry yet about syntax or about details such as how output will be displayed.

Example 1: Write an algorithm, in pseudocode, for a program that take two numbers and output the sum of the two numbers.

Algorithm

Step 1: Input two numbers say N_1 and N_2
Step 2: Calculate the sum $N_1 + N_2$
Step 3: out put the sum

Pseudocode

Input N_1 and N_2
Add N_1 and N_2
Output N_1 and N_2

Example 2: Write an algorithm to determine a student's final mark and indicate whether it is passing or failing. The final mark is calculated as the average of six marks

Algorithm

Step 1: Input the six marks M_1, M_2, M_3, M_4, M_5, M_6
Step 2: Calculate the average of the six marks
Average = (M_1, M_2, M_3, M_4, M_5, M_6)/ 6
Step 3: Check if the Average is greater than 50.
Step 4: if the average is greater than 50 then print (Output) Pass else i.e. if it is below 50 then print Fail.

Pseudocode

Input M_1, M_2, M_3, M_4, M_5, M_6
Calculate Average= (M_1, M_2, M_3, M_4, M_5, M_6)/ 6
If average is below 50
    Print "FAIL"
Else
    Print "PASS"

Exercise: Write an algorithm and pseudocode for the following problems

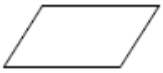
a. Take a number and calculate the factorial of that number
b. Take a number and prints "odd" if the number is odd otherwise prints "even"
c. Calculate the CGPA of a student. (The student should enter grade for the five subjects along with the respective credit hour).

2. Flowcharts

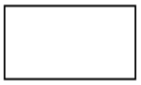
A *flowchart* is a diagram that shows the logical flow of a program. It is a useful tool for planning each operation a program must perform, and the order in which the operations are to occur. The logical flow of an algorithm can be seen by tracing through the flowchart. Some standard symbols used in the formation of flow charts are given below.

Flow Chart Symbols

An oval is used to indicate the beginning and end of an algorithm



A parallelogram indicates the input or output of information.



A rectangle indicates a computation, with the result of the computation assigned to a variable.

A diamond indicates a point where a decision is made.

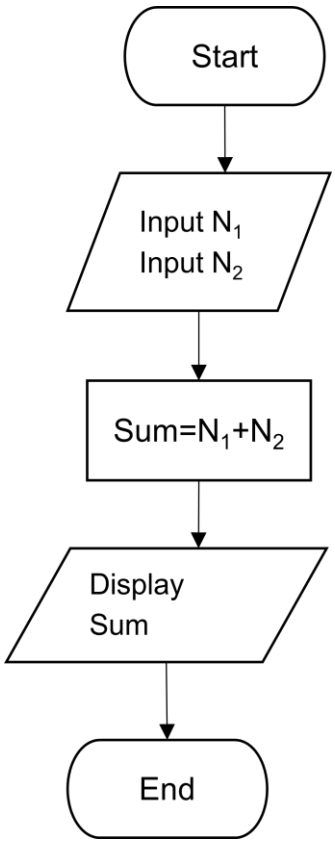
An arrow indicates the direction of flow of the algorithm. Circles with arrows connect the flowchart.

Example 1: Write an algorithm, in flow chart form that take two numbers and output the sum of the two numbers.

Algorithm

Step 1: Input two numbers say N_1 and N_2
Step 2: Calculate the sum $N_1 + N_2$
Step 3: out put the sum

Flow chart

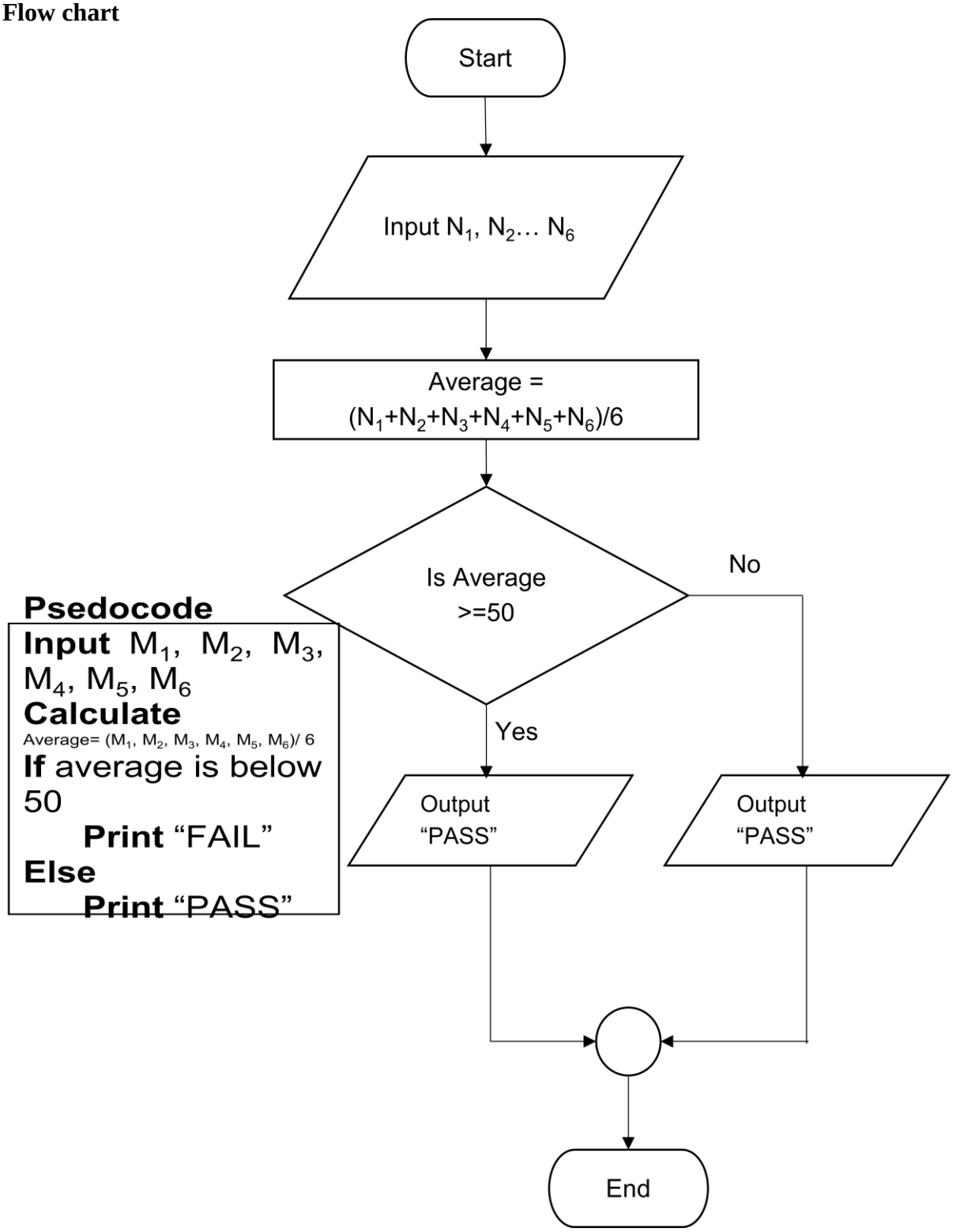


Psedocode

Input N_1 and N_2
Add N_1 and N_2
Output N_1 and N_2

Example 2: Write an algorithm in flow chart form to determine a student's final mark and indicate whether it is passing or failing. The final mark is calculated as the average of six marks.

Flow chart



Psedocode

Input M_1, M_2, M_3, M_4, M_5, M_6
Calculate
Average= (M_1, M_2, M_3, M_4, M_5, M_6)/ 6
If average is below 50
    Print "FAIL"
Else
    Print "PASS"

Example 3: Draw the flow chart for Software development Cycle.

The flowchart shows the software development cycle. In software development cycle, you basically design the system, code it and test it. When an error is found, it must be determined whether the error is a design error or not. Coding errors can quickly be fixed, but design errors may take longer.