Lecture 1, First Class
Production Engineering Division, Production Eng. & Metallurgy Dep., University of Technology

Dr. Laith Abdullah Mohammed

► Requirements and Grading:
First term Exam: 10%, Second term Exam: 10%
Final Exam: 60%
Homeworks, Quizess and Class attendance: 10%
Laboratory: 10%

► Course Materials:
Suggested references that covers Computer Science & Programming (any textbook). Also, some useful web sites:

http://computer.howstuffworks.com/
http://www.cgw.com/
http://www.computerworld.com/
http://www.computerandvideogames.com
http://www.apple.com/mac/
http://www.pcmag.com/
http://www.pcworld.co.uk/martprd/editorial/software_homepage
http://www.bbc.co.uk/computertutor/computertutorone/index.shtml
**Introduction:**
A **computer** is a machine that manipulates data according to a set of instructions. A computer is an **electronic machine** that can be programmed to **accept data (input)**, **process** it into **useful information (output)**, and **store** it in a **storage** media for future use.

**Computer Data Processing could include:**
- Arithmetic Operations.
- Logical Comparision.
- Transmitting Info.
- Receiving Info.
- Storing Info.
Every computer today is based on the von Neumann Model. It is based on 3 ideas:

- Four subsystems.
  1. **Memory** - the storage area of programs and data.
  2. **ALU** - arithmetic/logic operations take place
  3. **Control Unit** - control Memory, ALU, and I/O
  4. **I/O** - accept input data/send output data

- **Stored Program Concept.**

- **Sequential Execution of Instructions.**
  One or more pieces of data are *read* from memory (one at a time), The data is processed in the CPU, The results are *written* back into memory (one at a time)
**Computer Components**

Hardware: Electronic devices and Mechanical parts.
Software: Instructions.

## Hardware

consists of three groups according to functionalities:
- Central Processing Unit (CPU).
- Memory Unit.
- Input/Output Unit (I/O)

A general purpose computer has four main components: the arithmetic logic unit (ALU), the control unit, the memory, and the input and output devices (I/O). These parts are interconnected by busses, often made of groups of wires. Inside each of these parts are thousands to trillions of small electrical circuits which can be turned off or on by means of an electronic switch. Each circuit represents a bit (binary digit) of information so that when the circuit is on it represents a "1", and when off it represents a "0" (in positive logic representation). The circuits are arranged in logic gates so that one or more of the circuits may control the state of one or more of the other circuits. The control unit, ALU, registers, and basic I/O are collectively known as a central processing unit (CPU).

**Control Unit:** The control unit manages the computer's various components; it reads and interprets (decodes) the program instructions, transforming them into a series of control signals which activate other parts of the computer.

**Arithmetic Logic Unit:** The ALU is capable of performing two classes of operations: arithmetic and logic.

<table>
<thead>
<tr>
<th>Arithmetic Operations</th>
<th>Logical Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Add</td>
<td>=, ≠ equal to, not equal to</td>
</tr>
<tr>
<td>− Subtract</td>
<td>&gt;, &gt; greater than, not greater than</td>
</tr>
<tr>
<td>× Multiply</td>
<td>&lt;, &lt; less than, not less than</td>
</tr>
<tr>
<td>÷ Divide</td>
<td>≥, ≥ greater than or equal to, not greater than or equal to</td>
</tr>
<tr>
<td>^ Raise by a power</td>
<td>≤, ≤ less than or equal to, not less than or equal to</td>
</tr>
</tbody>
</table>
Central Processing Unit (CPU)
It is the *brain* of the computer, responsible for execute given instructions, and for arithmetic & logical processing, and control other hardware. **CPU examples:** Intel ® Core™, Intel Pentium, AMD Athlon™.

<table>
<thead>
<tr>
<th>CPU Chip Type</th>
<th>386</th>
<th>486</th>
<th>P1</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.Transistors</td>
<td>275,000</td>
<td>1,200,000</td>
<td>3,100,000</td>
<td>50,000,000</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>16-14</td>
<td>25-100</td>
<td>60-120</td>
<td>1000-3000</td>
</tr>
<tr>
<td>MIPS</td>
<td>11.4</td>
<td>54</td>
<td>112</td>
<td>9000</td>
</tr>
</tbody>
</table>

MIPS: millions of instructions per second

**Some facts about CPU:**
► The CPU speed is not controlled by the microprocessor itself, but by an external clock located on the motherboard. The speed of the processor is determined by the frequency of the clock signal. It is typically expressed in megahertz (MHz), and the higher the number, the faster the processor.
The information stored in memory may represent practically anything. Letters, numbers, even computer instructions can be placed into memory with equal ease. Since the CPU does not differentiate between different types of information, it is the software's responsibility to give significance to what the memory sees as nothing but a series of numbers.

In almost all modern computers, each memory cell is set up to store binary numbers in groups of eight bits (called a byte). Each byte is able to represent 256 different numbers ($2^8 = 256$); either from 0 to 255 or -128 to +127. To store larger numbers, several consecutive bytes may be used (typically, two, four or eight). A computer can store any kind of information in memory if it can be represented numerically.

Modern computers have billions or even trillions of bytes of memory. The CPU contains a special set of memory cells called registers that can be read and written to much more rapidly than the main memory area. Registers are used for the most frequently needed data items to avoid having to access main memory every time data is needed. As data is constantly being worked on, reducing the need to access main memory (which is often slow compared to the ALU and control units) greatly increases the computer's speed.

There are two types of memory:

- **Main memory:**
  Main memory is temporary storage (e.g. current program), faster, more expensive.
  Two types of memory are available: RAM and ROM.
1. Random-Access Memory (RAM):
RAM provides the bulk of the memory in a computer, can be read from and written to by the user. The information (program or data) is lost after the system is powered off. The more RAM a computer has, the more capacity the computer has to hold and process large programs and files. RAM can be installed on the motherboard, either as a permanent fixture, or in the form of small chips, referred to as Single Inline Memory Modules (SIMMs) or Dual Inline Memory Modules (DIMMs).

2. Read-Only Memory (ROM):
The contents of ROM come from the manufacturer; users are only allowed to read from it, but not write to it. A good example is the ROM-BIOS chip, which contains read only software. Often network cards and video cards also contain ROM chips. ROM is typically used to store the computer's initial start-up instructions.

In general, the contents of RAM are erased when the power to the computer is turned off, but ROM retains its data indefinitely. In a PC, the ROM contains a specialized program called the BIOS that loading the computer's Operating System from the hard disk drive into RAM whenever the computer is turned on or reset. In Embedded computers, which frequently do not have disk drives, all of the required software may be stored in ROM. Software stored in ROM is often called Firmware, because it is notionally more like hardware than software.
Flash memory blurs the distinction between ROM and RAM, as it retains its data when turned off but is also rewritable. It is typically much slower than conventional ROM and RAM however, so its use is restricted to applications where high speed is unnecessary.
Secondary memory:
Store data, slower, cheaper. Secondary memory like: Flash Memory, Floopy Disk, Hard Disk, Compact Disk (CD), Digital Video Disk (DVD). It hold data when the power is off.

Hard Disk Drive (HDD):
The HDD has a much larger storage capacity than the floppy drive for long-term storage. It store programs and files, as well as the operating system. Typically, the HDD is an internal drive that cannot be removed from the computer.
Knowledge of the **motherboard**, also called the system board or main board, is crucial because it is the nerve center of the computer system. Everything else in the system plugs into it, is controlled by it, and depends on it to communicate with other devices on the system.

It generally houses the CPU, the controller circuitry, the bus, RAM, expansion slots for additional boards, and ports for external devices. In addition, it contains the CMOS and other ROM BIOS and support chips providing varied functionality.
**Input/Output Unit**

- **Input devices** take data into computer system like (Mouse, Keyboard, Joystick, Scanner, Web camera, Graphics Tablet, Microphone).
  - **Scanner**: allow to scan printed material and convert it into a file format that may be used within the PC.
  - **Light Pens**: Used to allow users to point to areas on a screen.
  - **Touch Pads**: A device that lays on the desktop and responds to pressure.
  - **Joysticks**: Many games require a joystick for the proper playing of the game.
  - **Tracker Balls**: an alternative to the traditional mouse and often used by graphic designers.

- **Output devices** provide data to outside of the systems (Printer, Monitor, Plotter, Speakers).
  - **Printers**: There are many different types of printers. In large organizations laser printers are most commonly used due to the fact that they can print very fast and give a very high quality output.
  - **Plotters**: A plotter is an output device similar to a printer, but normally allows you to print larger images and maps.
  - **Speakers**: Enhances the value of educational and presentation products.
  - **Speech synthesizers**: Give the ability to not only to display text on a monitor but also to read the text to the user.
Monitors

Computers are usually connected to a display, also called a monitor. Some key monitor-related terms are: pixels, refresh rate, resolution, and size.

**Pixels:** Are picture elements. The screen image is made of pixels (tiny dots), which are arranged in rows across the screen. Each pixel consists of three colors: red, green, and blue (RGB).

**Dot pitch:** A measurement of how close together the phosphor dots are on the screen. The finer the dot pitch, the better image quality (measured in millimeters).

**Resolution:** Varies based on the number of pixels. The more pixels in the screen, the better the resolution. Better resolution means a sharper image. The lowest screen resolution on modern PCs is 640 x 480 pixels, which is called Video Graphics Array (VGA). (This is the default resolution after installing a video card using Windows.)

**Monitor screen sizes:** Measured in inches. The most common sizes are 14", 15", 17", 19", and 21" screens, measured diagonally.

**Video Cards:** The video card or video adapter is the interface between the computer and monitor. The video card tells the monitor which pixels to light up, what color the pixels should be and the intensity of the color. The display capabilities of a computer depend on both the video adapter and the monitor.
Input / Output (I/O) Ports

All peripheral devices that connect to the computer such as printers, scanners, and so on, use connectors on the back of the computer known as ports. There are different types of ports on the computer that serve different purposes. An I/O port is a pathway into and out of the computer.

1. Serial Ports
A serial port can be used to connect devices that use a serial interface such as a modem, scanner, mouse, etc. Generally, a PC can identify up to four serial ports, but the typical computer contains only two, referred to as COM1 and COM2. A serial port transmits data bits one after the other (serially) over a single line.

2. Parallel Ports
A parallel port is a socket on the computer that is used to connect a printer or other peripheral device such as a portable hard disk, tape backup, scanner, or a CD-ROM. The parallel port contains eight lines for transmitting an entire byte (8 bits) across the eight data lines simultaneously. Parallel ports can be configured as LPT1, LPT2, or LPT3.
Connecting CPU and Memory

A data bus, an address bus, and a control bus connect the CPU and the memory. **Bus**: a group of wires, each carrying 1 bit at a time.

Number of wires of each bus:
- Size of the word.
- Address space of memory.
- Total number of control commands.

Connecting I/O devices to the buses

A controller handles the I/O operations between the CPU/memory and the much slower I/O devices. SCSI, FireWire, and USB are common controllers.
1. The Small Computer Systems Interface (SCSI) controller

It has a parallel interface with 8, 16, or 32 wires. Provide a daisy chained connection. Each device must have a unique address.

Transfer rate for SCSI:
- SCSI-1: 5MB/sec
- Ultra-320: 320MB/sec
2. Universal Serial Bus (USB) controller:
USB is a serial controller used to connect slower devices to a computer. It has four wire bus, two carry power to the device. USB is an external port that allows the user to connect up to 127 external PC peripherals. External peripherals include the following: USB keyboards, Mice, Printers, Modems, Scanners, Digital cameras, Digital video cameras, External disk drives.
Transfer Rate for USB:
- Low Speed: 192 KB/sec.
- Full Speed: 1.5 MB/sec.
- Hi-Speed: 60 MB/sec.
3. FireWire controller

- High speed serial interface that transfers data in packets, also known as IEEE 1394.
  - FireWire 400 (IEEE 1394a): 50 MB/sec.
  - FireWire 800 (IEEE 1394b): 100 MB/sec.
- Connect up to 63 devices in a daisy chain or a tree connection.
Before the 1500s, in Europe, calculations were made with an **abacus**, Invented around 500BC, in Babylonia.

300-600 AD, The first use of the **number 0**, and **negative numbers** (first appeared in India).

In 1642, Pascal (French mathematician) invented a **mechanical calculator** called the Pascaline (The Arithmetic Machine could only add and subtract, while multiplication and division operations were implemented by performing a series of additions or subtractions), Initially used to help Pascal’s father with Tax computations!

In 1671, Gottfried von Leibniz (German mathematician) extended the Pascaline to do multiplications, divisions, square roots.

None of these machines had memory, and they required human intervention at each step.
In 1822 Charles Babbage (English mathematician), sometimes called the “father of computing” built the Difference Engine, a machine designed to automate the computation of polynomial functions. It implements some storage, all internal and temporary; the user doesn’t store anything.

The **Difference Engine**, consisted of 4,000 components, weighed three tons, and was 10 feet wide and 6½ feet tall. In Babbage's time, mathematical tables, such as logarithmic and trigonometric functions, were generated by teams of mathematicians working day and night on primitive calculators. Due to the fact that these people performed computations they were referred to as *"computers"*. In fact, the term "computer" was used as a job description (rather than referring to the machines themselves) well into the 1940s. This term later became associated with machines that could perform the computations on their own.

In 1833 Babbage designed the **Analytical Engine**, but he died before he could build it. It was built after his death, powered by steam! It was much more general than the difference engine and could in theory perform “any” mathematical operation. This is really the first machine that somewhat resembles our computers:

- An arithmetic processing unit (the mill).
- A memory (the store).
- Input/output devices (punched metal cards).
In 1842, Ada Lovelace (English mathematician, daughter of Lord Byron) wrote instructions for the Analytical Engine to compute the Bernoulli numbers: the first computer program. A programming language is named after her.

**Generation 1: Vacuum Tubes**

The vacuum tube is the first known device to amplify, switch, or modify a signal (by controlling the movements of electrons).

Memory was made up of hundreds of vacuum tubes. These tubes produce so much heat, high energy consumption, large. Input and output media were punched cards and magnetic tapes.
The ENIAC (Electronic Numerical Integrator and Calculator) was unveiled in 1946 University of Pennsylvania: the first all-electronic, general-purpose digital computer. (17,468 vacuum tubes, 1,800 ft², 30 tons, 174 KW of power, 1,000-bit memory, Punched card)

(By comparison, Modern Laptop computer may use around 30W, nearly six thousand times less. ENIAC able to process between 5 and 100 operations per second. A modern microprocessor (as of 2007) can process billions of operations per second).

It could do nuclear physics calculations (in two hours) which it would have taken 100 engineers a year to do by hand.

The use of binary: In the 30s Claude Shannon (the father of “information theory”) had proposed that the use of binary arithmetic and boolean logic should be used with electronic circuits.

The Von-Neumann architecture: In 1944, John von Neumann wrote a memo about computer architecture,
► A memory containing both data and instructions. Also to allow both data and instruction memory locations to be read from, and written to, in any desired order.
► A calculating unit capable of performing both arithmetic and logical operations on the data.
► A control unit, which could interpret an instruction retrieved from the memory and select alternative courses of action based on the results of previous operations. This is called von Neumann Machine, and virtually all digital computers from that time forward have been based on this architecture.
In 1948, AT&T Bell Laboratories scientists invent the "transfer resistor"; later labeled the transistor. An electronic switch that alternately allow or disallow electronic signal to pass, replaces vacuum tubes. These transistors were made of solid material, some of which is silicon, therefore they were very cheap to produce. Much smaller than vacuum tubes, draw less power, and generate less heat, conduct electricity faster.

- Evolution from Machine Language to **Assembly Language** (which were also called Symbolic Languages).

- Estimates say that there are around 100 computers in the world in 1953.

- 1959 It was the beginning of big computer vendors (**IBM**).

- Computers were still bulky and expensive, and so there were only in universities, government agencies, and large businesses.
Integrated Circuit developed in 1958 by Jack Kilby an engineer with Texas Instruments. An electronic circuit that packages transistors and other electronic components into one small silicon chip called semiconductor. Computers became ever smaller as more components were squeezed onto the chip.

The use of an operating system that allowed machines to run many different programs at once with a central program that monitored and coordinated the computer's memory.

Keyboards and monitors were used. Magnetic disks were used widely as secondary storage.

IBM System/360 were the first computers to be built entirely with ICs.

In 1965 BASIC (Beginners All Purpose Symbolic Instruction Code) was developed at Dartmouth College.

In 1965, the first mouse was invented by Douglas Englebart, but did not become popular until the mid 80’s by Apple.

Intel was founded in 1968. Also in 1970, the UNIX operating system was started.

In 1969 ARPANET was started by the US Dept. of Defense for research into networking. It was the original basis for what we now call the internet.
Generation 4: Microprocessor

- A silicon chip on which transistors are integrated onto it. Microprocessor can do all the processing of a full scale computer (smaller in size, faster in speed). These circuit integrations are known as Large-scale integrated (LSI) and Very Large-scale integrated (VLSI) circuits. Computers built after 1972 considered fourth generation computers.

Improvements to IC technology made it possible to integrate more and more transistors in a single chip:

- **SSI (Small Scale Integration):** 10-100
- **MSI (Medium Scale Integration):** 100-1,000
- **LSI (Large Scale Integration):** 1,000-10,000
- **VLSI (Very Large Scale Integration):** >10,000

- Intel produced the first **RAM chip** in 1970, capacity of 1 K-bit, 1024 bits.

- In 1971 the first **microprocessor**, the 4004 (4-bit, 108KHz RAM chip), was developed for Intel.

- In 1972 Nolan Bushnell founded Atari and Pong, Pong is widely recognized as the first popular arcade **video game**.

- **C programming language** developed. C++, which allowed for Object-Orientated Programming, was introduced in early 1980s.

- **First scientific calculator** developed by Hewlett-Packard.

- **Microsoft** is founded by Bill Gates and Paul Allen (1975).
The Cray 1 is the first commercially developed Supercomputer.

Introduction of the 8086 by Intel (1978), the first commercially successful 16 bit processor.

IBM started to develop their own PC. On August 12, 1981 IBM announced its own personal computer (which is used in home, office, school). Using the 16 bit Intel 8088 microprocessor, allowed for increased speed and huge amounts of memory.

1981 Microsoft provides the Disk Operating System (DOS) for the IBM Personal Computer.

TCP/IP Protocol established.

Domain Name Server (DNS) introduced to the Internet.


Microsoft Windows is launched (1985).


Netscape 1.0 written (1994).

Windows '95 was launched by Microsoft. This is an entire operating system and does not rely on MS-DOS.
Generation 5: The Future

Future developments could be in the following fields:
► Voice recognition.
► Artificial intelligence and Expert systems.
► Quantum computing
► Bio computing and parallel processing (many CPUs works together).
► Nano technology and Superconductor technology (which allows the flow of electricity with little or no resistance, greatly improving the speed of information flow).
► Learning.
► Natural languages.

Why are Computers so useful?

Storage — Reliability — Speed — Accuracy — Communication

Types of Computer Users:
1. End Users: Individuals who uses the product after it has been fully developed and marketed.
2. Small Business Users: Small companies.

Directions of Computer Development:

↓ Size: Everything has become smaller.

▲ Power: Miniaturization allowed computer makers to cram more power into their machines, providing faster processing speeds and more data storage capacity.

↓ Expensive: The price of the hardware is getting cheaper.
How computers change our life?

- Communication (Long distance phone v.s. Skype).
- Gathering after school (icq, msn).
- Entertainment (cinema, DVD/youtube, media).
- Project/Reports (paper-based, doc/presentations).
- Activities in daily life: News, Banking, Shopping.
Some Forms of Computers

Desktop Computer  Notebook Computer  Tablet

Game Console  PDA

Embedded Computer  CNC Milling machine
Supercomputers: Fastest, most powerful, most expensive among the categories. Suitable for intensive calculations & processing, Example Application: weather maps, construction of atom bombs, finding oil, earthquake prediction, etc.

Mainframes support more simultaneous programs. Allows hundreds of people to have simultaneous computer usage. Used in large business environment (e.g. bank).
**Workstation:**

- Powerful desktop computers.
- Used by engineers and scientists for engineering applications, software development, applications that require a high amount of computing power.

**Microcomputer:**

Microcomputers are designed to be used by individuals, whether in the form of PCs (Mini-tower, Desktop), workstations or notebook computers.

A microcomputer contains: a CPU on a microchip, a memory system (typically ROM and RAM), a bus system and I/O ports, typically housed in a motherboard.

The most common for home users, computers that can fit on a desktop or in one's briefcase. Can perform all of its input, processing, output and storage activities by itself.
**Tablet computer**
Specialized notebook equipped with touch-screen / tablet input. Instead of using keyboard, users could also input via handwriting recognition / virtual keyboard.

**Notebooks:**
with dimension from 9 inches to 14 inches.
Microcontroller:

Personal Digital Assistant (PDA)
Perform simple tasks, Small screen, Input and output. Like: Hand-held Personal Digital Assistant.

DSP: Digital Signal Processor
Computers in the future - PC in year 2015

MORE CONVENIENT
E-Book with Dual monitor?!!

PC anywhere?

More environment-friendly: Solar notebook?
**Software:** a set of instructions that used for controlling computer hardware. Software is written in a computer programming language such as Pascal, COBOL, C++, Java, BASIC etc.

- **Application software:** is a program that helps users in accomplishing specific tasks productively.
  Like (Word processors, spreadsheets, Photo Editing, Creating Web pages, Games, Instant messenger, Design, Accounting, Factory automation,.....).

- **System software:** is a program that is not directly intended to help accomplishing the user's task but supports other application software. Helps the computer perform essential operating tasks and enables the application software to run.
  Like:
  1. Operating systems (e.g. DOS, Windows, Unix, Mac, Linux, solaris),
  2. System tools (e.g. Antivirus, archiving tools, disk defragmenter),
  3. Software development tools (e.g. Compiler, debugger, integrated development environment suite, Programming Languages, Database systems).
  4. Drivers: Specific software programs that operates a specific piece of hardware.
### Useful Software Categories

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>Unix and BSD</th>
<th>Unix system V, solaris, IRIX, BSD, HP-UX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>DOS</td>
<td>QDOS, PC-DOS, MS-DOS, Free DOS</td>
<td></td>
</tr>
<tr>
<td>Mac OS</td>
<td>Mac OS Classic, Mac OS X</td>
<td></td>
</tr>
<tr>
<td>Experimental</td>
<td>Amoeba, Oberon/Bluebottle, Plan 9 from Bell Labs</td>
<td></td>
</tr>
<tr>
<td>Library</td>
<td>Multimedia</td>
<td>Direct X, open GL, Open AL</td>
</tr>
<tr>
<td></td>
<td>Programming Library</td>
<td>C Standard Library, Standard Template Library</td>
</tr>
<tr>
<td>Data</td>
<td>Protocol</td>
<td>TCP/IP, Kermit, FTP, HTTP, SMTP</td>
</tr>
<tr>
<td></td>
<td>File Format</td>
<td>HTML, XML, JPEG, MPEG, DXF, MP3, DWG</td>
</tr>
<tr>
<td>User Interface</td>
<td>Graphical User Interface</td>
<td>Microsoft Windows, GNOME, KDE, QNX Photon, CDE, GEM</td>
</tr>
<tr>
<td></td>
<td>Text-based user interface</td>
<td>Command-line interface, Text user interface</td>
</tr>
<tr>
<td>Application</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Office suite</td>
<td>Word processing (Microsoft Word), Desktop publishing, Presentation program (Microsoft PowerPoint), Database management (Microsoft Access), Spreadsheet (Microsoft Excel), Accounting, Scheduling &amp; time management (Visio), Statistics (SPSS, NCSS).</td>
<td></td>
</tr>
<tr>
<td>Internet Access</td>
<td>Browser (IE, FireFox), E-mail client (Outlook, GMail), Web server, Instant messaging (Yahoo, MSN, Skype)</td>
<td></td>
</tr>
<tr>
<td>Design &amp; Manufacturing</td>
<td>Computer Aided Design (AutoCAD, SolidWorks), Computer aided Manufacturing (MasterCAM, ArtCAM, ESPRIT), Planet management (Microstation), Robotic Manufacturing, Supply chain management</td>
<td></td>
</tr>
<tr>
<td>Graphics</td>
<td>Raster graphics editor (PhotoExpress), Vector graphics editor (Corel Draw), 3D modeler, Animation editor (Flash, Shockwave), Video editing (Prmiere, Ulead Media Studio), Image processing and Editor (Paint, Adobe Photoshop), 3D computer graphics (3D Studio MAX, POSER, Bryce, Carrara).</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>Digital audio editor, Audio playback, Mixing, Computer Music, Audio synthesis.</td>
<td></td>
</tr>
<tr>
<td>Software Engineering</td>
<td>Compiler, Assembler, Interpreter, Debugger, Text editor, Software configuration Management.</td>
<td></td>
</tr>
<tr>
<td>Educational</td>
<td>Flight simulator, Educational game, Interactive Training, Language learning, Home Design,</td>
<td></td>
</tr>
<tr>
<td>Games</td>
<td>Strategy, Arcade, Puzzle, Simulation, Interactive fiction</td>
<td></td>
</tr>
<tr>
<td>Misc</td>
<td>Artificial Intelligence, Antivirus (Norton, McAfee), File manager, Mathematical (Maple, Mathematica, Matlab, MathCAD), FEA (Ansys, Nastran)</td>
<td></td>
</tr>
</tbody>
</table>
Why we need Software?

We known that the computer is composed of various components of hardware, When the user wants to perform certain action, he/she gives instructions to computer Hardware through computer Software. Generally, program (software) are written in high level languages easily understood by humans and translated into machine language, The machine language is easily readable by computer microprocessors.

Program:
A sequence of instructions to be performed by a computer.

Scripts are text files that tell programs what to do (like in Matlab).

Computer programming: The process of planning a sequence of steps for a computer to follow.

Programming languages:
A set of rules, symbols, and special words used to construct a computer program. There are rules for syntax (grammar) and semantics (meaning). They are generally either translated into machine code by a compiler or an assembler before being run, or translated directly at run time by an interpreter. Sometimes programs are executed by a hybrid method of the two techniques. There are thousands of different programming languages, some intended to be general purpose, others useful only for highly specialized applications.

<table>
<thead>
<tr>
<th>Commonly used languages</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly languages</td>
<td>ARM, MIPS, x86</td>
</tr>
<tr>
<td>Scripting languages</td>
<td>Bourne script, JavaScript, Python, Ruby, PHP, Perl</td>
</tr>
<tr>
<td>High-level programming languages</td>
<td>Ada, BASIC, C, C++, COBOL, Fortran, Java, Lisp, Pascal, Object Pascal, Matlab</td>
</tr>
</tbody>
</table>
| **Machine language** | - Binary-coded instructions.  
- Closely coupled with design of computer hardware. |
| **Assembly language** | - Low-level programming language in which a mnemonic is used to represent each of the machine language instructions.  
- We need an **assembler** which translate an assembly language program into machine code. |
| **High-level language** | - Closer to English and other natural languages.  
- We need a **compiler** which translate a high-level language program into machine code.  
- Source program: Program written in a high-level language.  
- Object program: Machine language version of a source program.  
- Difference between compilation and execution of a program.  
- Some programming languages are translated by an **interpreter** (some versions of BASIC). Interpreter translates and executes each instruction in the source program. Java uses both a compiler and interpreter. |
Some examples of Programming Languages....

Fortran code:
if (N .eq. M) then
  N = M + 1
endif

C code:
if (N == M) {
  N = M + 1;
}

Matlab:
if (N == M)
  N = M + 1;
end

An assembly language:
LDA N ;get n
SUB M ;subtract m
JNZ NEXT ;test N=M?
ADD C1 ;yes-add one
STA N ;store result

Machine code - just bits.....
000010100
000100100
000001110
000010100
000100100
000001110

Why Study Programming ?

☑ Computers are fast and accurate in computation.
☑ Virtually all engineers use computers in some way.
☑ Engineers are responsible for the input and output of the programs they use.
☑ It is likely that you will have to write some programs some time.
☑ Programming skills are needed in other courses.
☑ Being able to program is a valuable job skill.
☑ Programming/problem solving skills from this course can be applied to other areas.
☑ It is fun !
<table>
<thead>
<tr>
<th>Types of Software Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commercial Software</strong></td>
</tr>
<tr>
<td>Created by a software house (Microsoft, Borland, Adobe, Mac OS), typically by large teams of programmers.</td>
</tr>
<tr>
<td>The user will pay the licensing fee to the maker.</td>
</tr>
<tr>
<td>The user is legally prohibited from making copies of the licensed software.</td>
</tr>
<tr>
<td>Some commercial software is bundled with hardware.</td>
</tr>
<tr>
<td>Evaluation versions may be free, low cost, or with limited functionality.</td>
</tr>
<tr>
<td><strong>Freeware</strong></td>
</tr>
<tr>
<td>Software that can be copied and distributed at no cost.</td>
</tr>
<tr>
<td>Author still retains the copyright.</td>
</tr>
<tr>
<td>Free Software Downloadable websites like <a href="http://www.download.com">www.download.com</a></td>
</tr>
<tr>
<td>Need to be very careful about security when downloading.</td>
</tr>
<tr>
<td><strong>Shareware</strong></td>
</tr>
<tr>
<td>Software that can be used in trial period for free, then the author expects a small payment.</td>
</tr>
<tr>
<td>Author still retains the copyright.</td>
</tr>
</tbody>
</table>

**Types of Agreement in Software Licensing**

There are four types of agreement which specifies the number of people that may use the software and the number of computers on which software may be installed:

1. Single-User License
2. Multi-User License
3. Concurrent-User License (e.g. Database Software)
4. Site License (e.g. University Users)
Programming Process

1. Problem-solving phase:
   - Analysis and specification (understand and define problem, and what is expected of solution)
   - General solution (algorithm: a logical sequence of steps that solves the problem)
   - Verification (Follow steps to make sure solution solves the problem)

2. Implementation phase:
   - Concrete solution (Program in a Programming language)
   - Testing (make sure the program produces the desired results)

3. Maintenance phase:
   - Use Program
   - Maintain Program (meet changing requirements)
Basic operations that computers can perform are very limited. These operations are:

1. Read data.
2. Write data.
3. Make comparison
   (= Equal to, > Greater than, < Less than, ≤ Less than or equal to, ≥ Greater than or equal to, <> Not Equal to)

Relational Operators:
- **OR** Joins two conditions and gives true if either of the condition is true or both of them are true. In all other cases it gives false.
- **AND** Joins two conditions and gives false if either of the condition is false or both of them are false. In all other cases it gives true.
- **NOT** If the condition is true, it gives false. If it is false, it gives true.

4. Create variables and assign values to them.
5. Do processing: Computers can do calculations
   (+ Addition, - Subtraction, ^ Raised to the power of, / Division, * Multiplication, () Brackets)
6. Loop formation: Computer can process a single or more steps repeatedly.
**Presentation by students**

Each student prepares a presentation on one of the following topics.
No. of Slides: Less than 10 (using Microsoft PowerPoint)
Duration: 2 weeks.

<table>
<thead>
<tr>
<th></th>
<th>Topic</th>
<th></th>
<th>Topic</th>
<th></th>
<th>Topic</th>
<th></th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCMCIA bus</td>
<td>11</td>
<td>3D Video Chipset</td>
<td>21</td>
<td>optical character recognition (OCR)</td>
<td>31</td>
<td>Wi-Fi</td>
</tr>
<tr>
<td>2</td>
<td>Infrared Interface</td>
<td>12</td>
<td>3D audio Processing</td>
<td>22</td>
<td>Raster Graphics</td>
<td>32</td>
<td>World Wide Web (WWW)</td>
</tr>
<tr>
<td>3</td>
<td>Bluetooth Radio Interface</td>
<td>13</td>
<td>ASCII code</td>
<td>23</td>
<td>Registry Files</td>
<td>33</td>
<td>FTP</td>
</tr>
<tr>
<td>4</td>
<td>Xeon processors</td>
<td>14</td>
<td>computer-based training (CBT)</td>
<td>24</td>
<td>Image Rendering</td>
<td>34</td>
<td>ETHERNET</td>
</tr>
<tr>
<td>5</td>
<td>AMD Athlon</td>
<td>15</td>
<td>Dynamic Link Library (DLL)</td>
<td>25</td>
<td>Network Router</td>
<td>35</td>
<td>HTML</td>
</tr>
<tr>
<td>6</td>
<td>Dual-Core Processors</td>
<td>16</td>
<td>Joint Photographic Experts Group (JPEG)</td>
<td>26</td>
<td>TCP/IP</td>
<td>36</td>
<td>Puzzle Games</td>
</tr>
<tr>
<td>7</td>
<td>CMOS</td>
<td>17</td>
<td>LAN</td>
<td>27</td>
<td>Unicode</td>
<td>37</td>
<td>OpenGL,OpenAL</td>
</tr>
<tr>
<td>8</td>
<td>BIOS</td>
<td>18</td>
<td>LCD</td>
<td>28</td>
<td>UPC</td>
<td>38</td>
<td>Linux</td>
</tr>
<tr>
<td>9</td>
<td>MultiMediaCard (MMC)</td>
<td>19</td>
<td>Light pen</td>
<td>29</td>
<td>URL</td>
<td>39</td>
<td>Mac OS X</td>
</tr>
<tr>
<td>10</td>
<td>DVD Multi</td>
<td>20</td>
<td>MPEG</td>
<td>30</td>
<td>virus</td>
<td>40</td>
<td>Animation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Vector Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Perl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Nano engineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Information technology (IT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>W3C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>DNA computing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Chemical Computer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Optical Computer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Quantum Computer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Algorithm:

**CHARACTERISTICS OF AN ALGORITHM:**

1. Algorithms always have a definite starting point and an end point. These points are generally marked with the words like Start, Begin, End, Stop etc.
2. They consist of finite number of steps.
3. They always relate to a specific problem or you can say that they are written for a given problem.
4. They serve as foundation stone for programming.
5. They are written in easy language.

---

**Example 1:** Algorithm for converting from Celsius to Fahrenheit can be represented as:

1. \(F = \frac{9}{5} C + 32\) (algebraic equation)
2. “Multiply the temperature reading in Celsius by 9/5 and then add 32 to the product”.

---

**Example 2:** Find minimum in a set of numbers

1. Take a look at the first number; memorize it
2. Take a look at the next number
3. Is it less than the number we just memorized?
   - **Yes:** Memorize this one instead
   - **No:** Ignore this number
4. Are more numbers are left?
   - **Yes:** Goto step 2
   - **No:** Read out the memorized number. That’s the required minimum.
**Example 3.**
Write algorithm for swapping contents of two variables.

**Algorithm:**

1. Start
2. Read value of A
3. Read value of B
4. C=A
5. A=B
6. B=C
7. Print A
8. Print B
9. End

This algorithm makes use of intermediate variable C. It holds value of A in it, temporarily and later assigns it to B variable. Thus values of variable A and B get interchanged. There is another way of interchanging the values i.e. without making use of intermediate variable. Algorithm for it is given below.

**Algorithm:**

1. Start
2. Read value of A and B
3. A=A+B
4. B=A-B
5. A=A-B
6. Print A, B
7. End
Problem 4.
Write an algorithm to find the largest of given three numbers.

Algorithm:

1. Start
2. Read three numbers A, B and C
3. Let Big=0
4. IF A>B Then Big=A Else Big=B
5. IF C>Big Then Big=C
6. Print Big
7. End

Problem 5.
Write an algorithm for calculating and printing factorial (!) of a given number.

Algorithm:

1. Start
2. Read a number, A
3. Let I=1
4. Let Sum=1
5. Sum=Sum*I
6. I=I+1
7. If I is not > A perform step 5, 6 and 7
8. Print Sum
9. End
Problem 6.
Write an algorithm for solving a given quadratic equation, \( ax^2 + bx + c = 0 \). Note that roots are determined by following formula:

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

Algorithm:

1. Start
2. Read value of a, b and c
3. If a=0 Stop
4. Calculate values of discriminant \( D = b^2 - 4ac \)
5. If \( D = 0 \) then there is one root \( p = \frac{-b}{2a} \)
6. If \( D > 0 \) then there are two real roots
7. If \( D < 0 \) then there are two complex roots

\[
p = \frac{b + \sqrt{D}}{2a} \quad \text{and} \quad q = \frac{-b - \sqrt{D}}{2a}
\]

8. Print p and q
9. Stop

PROBLEMS WITH ALGORITHMS:

As mentioned earlier, algorithms are written in English like sentences. Sentences are always subject to misinterpretation. If the sentence is complex, different readers may interpret it differently. This definitely leads to problems at the time of coding. To overcome this problem, often the solution of the problem is provided in pictorial form. Pictures carry more meaning and don't lead to ambiguity. Such type of step-by-step solution, provided in pictorial form is called flowchart.
FLOW CHARTS

When a step-by-step solution of a given problem is illustrated in the form of graphical chart that chart is called **flowchart**.

**FLOWCHARTS SYMBOLS**

Flowchart is a universal tool. A flowchart consists of special geometric symbols connected by arrows. Within each symbol is a phrase presenting the activity at that step. The shape of the symbol indicates the type of operation that is to occur. For instance, the parallelogram denotes input or output. The arrows connecting the symbols, called **flowlines**, show the progression in which the steps take place. Flowcharts should “flow” from the top of the page to the bottom.

A table of the flowchart symbols adopted by the American National Standards Institute (ANSI) follows.

<table>
<thead>
<tr>
<th>Symbol, Name,</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flowline</td>
<td>Used to connect symbols and indicate the flow of logic.</td>
</tr>
<tr>
<td>Terminal</td>
<td>Used to represent the beginning (Start) or the end (End) of a task.</td>
</tr>
<tr>
<td>Input/Output</td>
<td>Used for input and output operations, such as reading and printing. The data to be read or printed are described inside.</td>
</tr>
<tr>
<td>Processing</td>
<td>Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.</td>
</tr>
<tr>
<td>Decision</td>
<td>Used for any logic or comparison operations. Unlike the input/output and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is “yes” or “no.”</td>
</tr>
<tr>
<td>Connector</td>
<td>Used to join different flowlines.</td>
</tr>
<tr>
<td>Offpage Connector</td>
<td>Used to indicate that the flowchart continues to a second page.</td>
</tr>
<tr>
<td>Predefined Process</td>
<td>Used to represent a group of statements that perform one processing task.</td>
</tr>
<tr>
<td>Annotation</td>
<td>Used to provide additional information about another flowchart symbol.</td>
</tr>
</tbody>
</table>

The main **advantage** of using a flowchart to plan a task is that it provides a pictorial representation of the task, which makes the logic easier to follow. We can clearly see every step and how each step is connected to the next.

The major **disadvantage** with flowcharts is that when a program is very large, the flowcharts may continue for many pages, making them difficult to follow and modify.
**Condition Checking / Decision**

It is used for depicting comparison of two values or condition checking or decision-making etc. The condition that is to be checked is written within the figure. It is quite obvious that whenever a condition will be checked there will be two outcomes i.e. either the condition will be true (condition holds good) or it will be false (condition doesn't hold good). These two outcomes are shown as two branches coming out of the symbol. To clearly illustrate, which branch relates to true and which branch relates to false, the words Yes (or Y or True or T) and No (or N or False or F) are written near the branches.

![Condition Checking](image)

**Processing**

Processing or calculation activities are depicted, using a rectangle. To make the process more illustrative, calculations that are being done are written within the rectangle.

```
Area=L*W
```

```
A=10
B=20
C=A+B
```
**Start/End**
Like algorithm, flowchart must have a definite starting/Ending point. Starting/Ending point of the flowchart is depicted through a flat oval shape symbol. To make the symbol more illustrative, the word (Start, End) is written within the symbol.

![Start/End symbol](image)

**Flow Lines**
Joining many symbols together makes a flowchart. Each symbol of the chart represents an activity. Which activity will be conducted after which activity, is depicted with help of flow lines. A flow line is a simple line with an arrow at its front end. The head of the arrow depicts the direction of flow.
An up arrow is used for depicting a loop in the flowcharts.

![Flow lines](image)

**Connectors**
Some of the flowcharts may turnout to be quite long. They may extend over many pages. Now the question is, how do you connect the processes, which are either far apart or are off the pages? For such type of requirements, two types of connectors are used. One is called same page connector and another is called "Off Page Connector".

![Connectors](image)
Predefined Process

If the problem to be solved is long then by showing all the steps in the same flowchart may make the chart complex. To avoid the complexity and keep the flowchart simple, you have to decompose the problem. In decomposition, you identify the group of mundane routines, such as initialization of variables etc. and the routines, which are repeatedly required. For example, printing the header lines, calculating a value based on certain formula etc. are the tasks that have to be performed repeatedly. You draw a flowchart for these processes separately and declare them as predefined processes and assign them name for reference. In the main flowchart, instead of writing all the steps, you simply include the symbol of predefined process along with the name. Whenever one wants to go into the details of predefined process, he refers corresponding chart separately.

Note that each process has a unique name. With the help of name you identify the process and refer it in the flowchart. In figure shown two predefined processes with the name “Ph” and “IV”.

From the description of flowchart, given above, it is quite clear that flowchart is nothing but a graphical representation of the solution for a given problem, in which you make use of standard graphical symbols and within the symbol, you write the details of the operation. For communicating the details of the activities of a particular step, you have to include some text within the symbol. For this, you have to often make use of different types of operators.
Problem (1):
Draw a flowchart for swapping the contents of two variables.
Flowchart for this problem is shown below fig (a).
Note that this flowchart makes use of intermediate variable, C to interchange the values of two variables A and B.
This flowchart can be redrawn to depict the method, which doesn't make use of intermediate variable. Shown in fig (b).

Problem (2)
Draw a flowchart for solving a given quadratic equation $ax^2+bx+c=0$. 
**PROBLEM (3)**
Draw a flowchart to find the larger of the three given numbers.

---

**PROBLEM (4)**
Draw a flowchart for calculating and printing factorial of a given number.
**Problem (5):**
Calculate and report the grade-point average for a class.

The average grade equals the sum of all grades divided by the number of students. We need a loop to read and then add (accumulate) the grades for each student in the class. Inside the loop, we also need to total (count) the number of students in the class.

**Input:** Student grades

**Processing:** Find the sum of the grades; count the number of students; calculate average grade = sum of grades / number of students.

**Output:** Average grade
The number system that we use in day-to-day life is called **Decimal Number System**. It makes use of 10 fundamental digits i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Computer doesn't make use of decimal number system. Internally it makes use of another number system, called **Binary Number System**. Binary number system is the fundamental base of computers. It makes use of two digits only i.e. 0 and 1. All other numbers in binary number system are formed using these digits.

### Conversion Of Decimal Number Into Binary Number:

A given decimal number can be converted into equivalent binary number by following method:

1. Since binary number system makes use of only two digits hence divide the decimal number by 2. Write the quotient below the number and remainder on the right hand side.
2. Now divide the quotient by 2 and repeat above-mentioned process till the time quotient becomes 0.
3. Now arrange the remainders \((x_1, x_2, x_3, \ldots \text{ etc.})\) in reverse order \((\ldots x_3, x_2, x_1)\)

The number thus obtained will be binary equivalent of given decimal number. To understand the procedure of conversion, consider following examples:

**Example -1**

Convert decimal number \((109)_{10}\) to equivalent binary number.

\[
\begin{array}{c|c}
2 & 109 \\
2 & 54 & 1 \\
2 & 27 & 0 \\
2 & 13 & 1 \\
2 & 6 & 1 \\
2 & 3 & 0 \\
2 & 1 & 1 \\
\hline
0 & 1 \\
\end{array}
\]

Number = \((1101101)_2\)

Thus above mentioned process yields: \((109)_{10} = (1101101)_2\)
How do you convert fractional numbers (that have decimal point in them) into equivalent binary numbers?

1. Since binary number system consists of only two digits hence multiply fractional decimal part by 2. When you do so, integer part obtained on the left hand side of the decimal number will either be 1 or 0. Whatever you get, write it separately (say you write x1. Here x can either be 1 or 0).

2. Now take the resultant fractional part and multiply it by 2 again. Repeat the process, as many times as number of digits are required on the right hand side of the binary point (thus obtaining x2, x3, x4 etc.). For example, if 4 digits are required on the right hand side of the binary point, repeat above-mentioned process 4 times.

3. Arrange the digits that were written separately in the sequence, in which they were obtained (e.g. x1, x2, x3, and x4). The number thus obtained (x1x2x3x4) will be binary equivalent of given fractional decimal number.

Example (1):
Convert decimal number (0.862)10 to equivalent binary number.

\[
\begin{align*}
0.862 \times 2 &= 1.724 & 1 \\
.724 \times 2 &= 1.448 & 1 \\
.448 \times 2 &= 0.896 & 0 \\
.896 \times 2 &= 1.792 & 1 \\
.792 \times 2 &= 1.584 & 1 \\
.584 \times 2 &= 1.168 & 1 \\
\end{align*}
\]

Thus above mentioned process yields: (0.862)_{10} = (0.110111)_2

Example (2)
Convert decimal number (122.486)10 to equivalent binary number.

First Step: Integer part of the number is taken:

\[
\begin{array}{c|c}
2 & 122 \\
2 & 61 \\
2 & 30 \\
2 & 15 \\
2 & 7 \\
2 & 3 \\
2 & 1 \\
2 & 0 \\
\hline
\end{array}
\]

Thus (122)_{10} = (111010)_2

Second Step: In second step, fractional part of the number is taken:

\[
\begin{align*}
.486 \times 2 &= 0.972 & 0 \\
.972 \times 2 &= 1.944 & 1 \\
.944 \times 2 &= 1.888 & 1 \\
.888 \times 2 &= 1.776 & 1 \\
\end{align*}
\]

Number = (0.0111)_2

Third Step: Joining both the parts together will result in following number:

Thus (122.486)_{10} = (111010.0111)_2
Converting Binary Numbers into Equivalent Decimal Numbers

Positional values for different positions in binary number are illustrated below:

<table>
<thead>
<tr>
<th></th>
<th>$2^5$</th>
<th>$2^4$</th>
<th>$2^3$</th>
<th>$2^2$</th>
<th>$2^1$</th>
<th>$2^0$</th>
<th>$2^{-1}$</th>
<th>$2^{-2}$</th>
<th>$2^{-3}$</th>
<th>$2^{-4}$</th>
<th>$2^{-5}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Point</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Perform following steps to convert a binary number into equivalent decimal number:
1. Multiply each digit of the number, by its positional value.
2. Now add all the products to get the sum.
The sum, thus obtained will be the equivalent decimal number of given binary number.

**Example:**
Convert $(1100110)_2$ into equivalent decimal number.

Solution:

$$(1100110)_2 = (1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)_{10}$$

$$= (64 + 32 + 4 + 2)_{10} = (102)_{10}$$

**ONE'S COMPLEMENT:**

In computers, one's complement is used for representing negative numbers.
When 0's in a binary number are replaced by 1s and 1s are replaced by 0s then the resultant number is said to be 1's complement of the number.

**For example:**
1's complement of 100110 will be 011001.
Similarly 01011100's one's complement will be 10100011.
**BIT:**

Computer stores all its data in the form of 0s and 1s. Digit 0 and 1 and called **Bits**. That means 0 is one bit and 1 is another bit. The data 1100 comprises of 4 bits. You can say that bit is the *smallest storage unit of computer*.

**BYTE:**

8 bit put together form 1 **byte**. For example, 10101011 will occupy one byte space.

**WORD:**

Number of bits that are used for representing a character within the computer are called **word**. Note that in most of the computers 8 bit are used to represent a character. Thus in those computers words comprises of one byte.

---

**MS-DOS Operating System**

A computer system is basically combination of hardware and software. For its functioning it requires different types of hardware devices, electronic components and various types of software. Operating system is one of the software, which computer uses for its internal functioning.

Operating System is essential software that is required for a computer to become operational. It provides functionality to computer hardware so that electro-mechanical components of it perform read, write and processing functions as human beings do. Without operating system, computer cannot work. Any instruction given by the user to the computer to perform a function is actually carried out by operating system. Operating system is essential software, purpose of which is to activate the computer and:

1. **perform internal management functions**
2. **provide services**.

Internal management functions are the functions that have to be essentially performed to make the computer work. For example, managing the processor, memory, devices, input /output functions, data etc. Services are bunch of commands and utilities that operating system provides to its users to have better control over computer.
MS-DOS is one of the most popular, powerful and useful operating system. It was designed and developed in the initial days of Personal Computers (PC) by Microsoft Corporation of USA. Due to its versatility and ease of operations, it became quite popular, within short span of time.

MS-DOS is Character User Interface (CUI) based operating system. To execute any command in MS-DOS, you need to know the command and its format. Any mistake in its spelling or format leads to error.

MS-DOS not only activates computer resources and controls them but it also provides many commands for performing day-to-day tasks.
CONSTITUTION OF MS-DOS
MS-DOS is a modular operating system and comprises of many files. It utilizes these files as and when required. Following files put together constitute MS-DOS operating system:

1. **IO.SYS**: It gets loaded into computer's memory at the time of booting the system. It primarily activates basic input and output devices like keyboard, VDU etc. and makes them functional.

2. **MSDOS.SYS**: It automatically gets loaded at the time of booting the operating system. It controls internal resources like memory, ALU, Control Unit etc. of computer.

3. **COMMAND.COM**: It comprises of few frequently used MS-DOS commands. For example, DIR, COPY, TYPE, REN etc. are internal commands. To execute an internal command, you need to type the command from the keyboard in its recommended format. When you do so, it is directly read from the memory and executed.

4. **External Command Files**: Each command exists in the form of executable file and resides on the disk. For example, FORMAT, XCOPY etc. They exist in the form of FORMAT.COM, XCOPY.EXE files respectively. To execute an external command, you need to type the name of the command from the keyboard and press Enter key. When you do so, computer reads the file from the disk, loads it into memory and executes it. When its execution is complete, it is removed from the memory.

FILES AND FILE TYPES
Whatever computer has to store on media like floppy hard disk or CD, it stores in the form of files. Whenever it has to make use of the contents, stored in the file, it accesses the file and reads them from it. In other words you can say that all read/write operations, in computer are done through files. Computer files can be broadly classified into two categories:

1. **Executable files**: are basically command files, which when executed perform specific task. For example, DISKCOPY.COM is a command file, which copies the contents of a floppy on another floppy.

2. **Data files**: are the files, which contain data, program or some information in them.
CONCEPT OF DIRECTORY:

Directory can be conceptualized as special file, which can hold files and directories in it. From this figure, it is quite clear that directories can be utilized for classified storage of files on the disk.

MS-DOS FILE SYSTEM

The mechanism of arranging the files and directories on the disk is called file system. In MS-DOS, file system looks like an inverted tree. Root directory appears at the top of the tree. Other directories branch off from there and files act as leaves. Such a file arrangement is often referred to as Hierarchical File System. A hierarchical file system consisting of few files and directories is illustrated in figure beside.

CONCEPT OF PATH

Path of a file or directory is the list of directory names in descending sequence, starting from root and each directory name separated by a back slash (\), following which you reach to the desired file or directory. For example, refer figure above. The path for file F1.1 will be \D1\F1.1. Similarly the path name for F2.2 file will be \D2\F2.2. Note that first backslash in the path name denotes root directory, while other backslashes serve the purpose of separators.
When you switch on the computer, it performs self-test. If self-test passes through correctly, operating system is read from the disk and loaded in computer's memory. Immediately after that following sign appears on VDU screen:

C:\>

Above mentioned sign is called system prompt. Different characters in system prompt signify different things. C: indicates that the system was booted from C drive. Character \ indicates that root is the current directory. The > sign is a terminator. Any thing written after this sign is treated as command. The dash sign is called cursor. It keeps blinking. It acts like tip of the pen. Whatever you type from the keyboard, gets typed at current cursor location and it shifts towards right.

INTERNAL COMMANDS

All those commands, which are part of COMMAND.COM file and remain resident in memory from the time of booting to the time of shutdown, are called internal commands. For example, DIR, COPY, EDIT, CLS, PROMPT, REN, DEL etc.

DIR Command
DIR command is used for displaying the names of all the files residing on media like floppy hard disk, CD etc. In its simplest form, it can be executed in following format:
C:\> DIR [Enter]

EDIT Command
EDIT command is used to create a text file and type text in it. It is executed in the following format:
C:\> EDIT [file name] [Enter]

TYPE Command
TYPE command is used for displaying the contents of a file on the screen. It is executed in the following format:
C:\> TYPE <file name.> [Enter]
COPY Command
COPY command is used to make a duplicate copy of a given file. It is executed in the following format:

C:\> COPY <Source file name> <Destination file name> [Enter]

REN Command
REN command is used for renaming an existing file. General syntax for REN command is as follows:

C:\> REN <Existing file name> <New file name> [Enter]

DEL Command
DEL command is used for removing a file from the disk. In its simplest form, it can be executed as follows:

C:\> DEL <File name> [Enter]

CD Command
CD is short form of Change Directory. It is used for moving from one directory to another directory:

C:\> CD <Directory name> [Enter]

MD Command
MD is short form of Make Directory. This command is used for creating a new directory:

C:\> MD <Directory name> [Enter]

RD Command
RD is short form of Remove Directory. This command is used for removing the directory from the disk, provided it is blank. It is used in the following format:

C:\> RD <Directory name> [Enter]
EXTERNAL COMMAND

All those commands of MS-DOS, which are not part of COMMAND.COM file and reside on the disk, in the form of executable files, fall into the category of external commands. For example, FORMAT, MOVE, MORE, TREE, DISKCOPY etc. are external commands.

**FORMAT Command**

Format command makes internal logical arrangement on the media like floppy, hard disk etc. to store the data. This arrangement is made by dividing the surface of the media into concentric circles and concentric circles into small segments, called *sectors*. Internal logical format of a typical floppy is illustrated in figure below. FORMAT command, in its simplest form can be executed as follows:

C:\> FORMAT [Drive name]

**MOVE Command**

MOVE command, physically moves files and directories from one place to another. It is used in the following format:

C:\> MOVE <Source file name> <Destination directory>

**TREE Command**

TREE command displays the file system, present on the disk, in graphical form. In its simplest form, it is executed in the following format:

C:\> TREE [Enter]
QBasic stands for **Beginner’s All-Purpose Symbolic Instruction**. It is a programming language written for computers back in 1975, by Bill Gates & Paul Allen. It is ease of use, its English-like commands and its power.
• The Character set:
   1. The Letters: its from A to Z (small or capital).
   2. The Digits: its from 0 to 9
   3. Special symbols: like (+,-,^,?,!,#, etc.)

• The Constants:
   1. Numeric constants: include all numbers (real, not real, integer, ……).
      Example: 25, -230, 0, 16.44, 0.88
   2. Character constants: include all characters sets (letters, digits, symbols) between two “ “. Example: “ BASIC “ , “ The width is = 83 “, “ Telephone Number 07901 “.

• The Variables:
A variable is a name which can contain a value. The variables must include the conditions below:
1. From A to Z (include all letters).
2. Not contained symbols except dot (.).
3. Maximum length of variable is 40.
4. Must not represent any word which is defined as a special word in QBASIC.
5. Must be start by letters.

A. **Numeric Variables:**
Like: M, A2, WE,……etc
In Numeric variables the symbol (%) (Integer) mean make the numeric variable as real number.
Like: A%=6.2 its mean A%=6
     A%=6.6 its mean A%=7
     A%=6.5 its mean A%=7
     A%=-6.5 its mean A%=-6
Using symbol (&) (Long) with numeric variable make it long variable.
Using symbol (!) (Single) with numeric variable mean the length of variable equal to 7 digits or less.
Using symbol (#) (Double) with numeric variable mean the length of variable more than 7 digits.

B. **Character (String) Variables:** If the variable holds symbols or text, it may be a character variable or a string variable. like: M$, A2$, WE$,……etc
Single-precision variables (floating-point variables): these types of variables are used to store numbers that contain a decimal value such as 1.89 or 3.141593.

**INTEGER**: A non-floating point variable (no decimal value) that can store integers between -32,768 and 32,767

**LONG**: same as INTEGER, but can contain numbers between -2,147,483,648 and 2,147,483,647

**DOUBLE**: same as SINGLE, but can have twice as many digits. (like: 983288.18)

**SINGLE**: single precision variables. (like: 39.2932)

To define a variable's type, use **DIM** with the **AS** attribute.

```vbnet
DIM var1 AS INTEGER
DIM var2 AS LONG
DIM var3 AS DOUBLE
var1=15.28
var2=-2000000000
var3=12345678.12345678
PRINT var1
PRINT var2
PRINT var3
```

**Example:**

```vbnet
B=26
A$=" I Like to Learn QBASIC "
PRINT B
PRINT A$
```

**Output:**

```
15
-2000000000
12345678.12345678
```
**Strings variables:**

String variables are ones that can hold all ASCII characters (Letters, Numbers, Symbols). They can not be used in math problems. When asking questions about them, or changing their content, the expressions must be in quotes. String variables are letters and numbers followed by a dollar sign ($). String names must have a letter as the first character, but everything else is up to you.

*Examples:*
"0123456789"
"abc123"
"1+1=2"
"!@&%§?><°^"
"Hi"

**Expressions:**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Function</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Add</td>
<td>8+2</td>
<td>10</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
<td>8*2</td>
<td>16</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
<td>8/2</td>
<td>4</td>
</tr>
<tr>
<td>-</td>
<td>Subtract</td>
<td>8-2</td>
<td>6</td>
</tr>
<tr>
<td>^</td>
<td>Exponentiation</td>
<td>8^2</td>
<td>64</td>
</tr>
</tbody>
</table>

*Example:*
Z = 500 + (10*7)
rate = 50
time = 2
distance = rate * time
PRINT Z
PRINT distance

*Output:*
570
100
Example: Write the following expression using BASIC format. And define the priority of the calculation for the expression

\[ x = \frac{a^3 - 7b^2}{3b + \sqrt{d}} \]

Solution:

\[ X = (A^3 - 7 \times B^2) / (3 \times B + D^{(1/2)}) \]

Example: Based on the priority of the calculation calculate the following:

1. \(2 + 3/4 \times 5\) \(\Rightarrow\) \(= 5.75\)
2. \(2 + 3^2\) \(\Rightarrow\) \(2 + 9 = 11\)
3. \(3 \times 4/5\) \(\Rightarrow\) \(12/5 = 2.4\)
4. \(3 + 4 - 5\) \(\Rightarrow\) \(7 - 5 = 2\)
Comparison Operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
<td>IF a=15 THEN...</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
<td>IF a&lt;&gt;15 THEN...</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>IF a&lt;15 THEN...</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less or equal to</td>
<td>IF a&lt;=15 THEN...</td>
</tr>
<tr>
<td>&gt;</td>
<td>More than</td>
<td>IF a&gt;15 THEN...</td>
</tr>
<tr>
<td>&gt;=</td>
<td>More or equal to</td>
<td>IF a&gt;=15 THEN...</td>
</tr>
</tbody>
</table>

Logical Operators:

1. **OR operator**: the result of this operator is true if one of the statements is true.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A OR B</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>IF 10&lt;14 OR 3^2=9 THEN...</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>IF 10&lt;14 OR 3^2=5 THEN...</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>IF 10&lt;2 OR 3^2=9 THEN...</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>IF 10&lt;2 OR 3^2=5 THEN...</td>
</tr>
</tbody>
</table>
2. **AND operator**: the result of this operator is true when both statements are true.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A AND B</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>IF 10&lt;14 AND 3^2=9 THEN...</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>IF 10&lt;14 AND 3^2=5 THEN...</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>IF 10&lt;2 AND 3^2=9 THEN...</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>IF 10&lt;2 AND 3^2=5 THEN...</td>
</tr>
</tbody>
</table>

3. **NOT operator**:

*Example:*
- NOT 15 > 10            False
- NOT 15 = 8             True
- NOT (5 >= 5 AND 6/2= 4) True

**Note**: ♦ The priority in Logical operations are:
1. The parenthesis ( ).
2. NOT operator.
3. AND operator.
4. OR operator.

♦ The priority between Mathematical Expression and Logical operations are:
1. Mathematical Expressions.
2. Logical Operations.
Example: Decide which statement is True and which is False.

A=3 , B=10 , C=100

According to Priority Law,

C=100

C<100

A=3 OR C>100 AND B<10

(A=3 OR C>100) AND B<10

B<=10 AND C=10 AND A*B=6 OR A<>13

B<=10 AND C=10 AND (A*B=6 OR A<>13)
Adding Documentation to the program:

Documenting your program allows you to remind yourself about something in your program. Plus, if your program is seen by other people, documenting can help them understand your code. The **REM** (remark) command enables you to add comments to your program without the text being treated like an instruction.

**General Form:**

```
REM or ' followed by comments.
```

**Example:**

```
CLS ' This command clears the screen
REM Program to compute area
```

**LET command:**

**General Form:**

```
LET variable=expression [Note: the word LET is optional].
```

**Purpose:**

Algebraic or string depending upon the variable – uses + - ^ \ MOD.

**Example:**

```
LET X= A+B*C
LET J=J+1
LET C=SQRT (A*A+B^2)
```
PRINT command:

PRINT statement is the statement which causes output to be displayed on the screen. If one wants words or letters printed they must be put between quotation marks ("). The words or letters between the quotation marks are called a string. To print numbers one does not require quotation marks.

Notes:
- Variables can be PRINTed by adding the variables title after the PRINT command.
- Math problems can also be PRINTed.
- More than one thing can be PRINTed on a line by adding semicolons between separate things you want PRINTed.
- You must PRINT spaces between string variables and other things you want PRINTed.
- If a semicolon follows a PRINT statement, the next thing PRINTed will be on the same line.
- If you type PRINT without any variables or text after it, it will PRINT a blank line.

Examples:
PRINT "Hello " [ENTER then F5]
Hello

PRINT " 512+456 " [ENTER then F5]
512+456

PRINT 523 [ENTER then F5]
523
INPUT command:

INPUT is a command that takes information from the user of the program. The person types in information, and it is assigned to a variable. You INPUT like this:

```
INPUT [Variable name].
```

This prints a question mark (?) and puts a cursor on the screen where you should begin to type. After all desired information is inputted, press enter, and it is saved in the array specified by the INPUT command.

String and numeric variables can be written with this command, but if you type characters other than numbers for numeric variables, the computer will not accept the information, and will write "Redo from start", and you will be given a second chance to INPUT the information.

(input number or string without a message)

```
INPUT w  (for a number)
INPUT w$  (for a string)
```

(input with a message)

```
INPUT "prompt ", w
INPUT "prompt ", w$
```
Example:

PRINT "What is your name ? "
INPUT name$
PRINT " Hello "; name$; " nice to meet you ! "
PRINT "How old are you ? 
INPUT age
PRINT " You are "; age; " years old "; name$
END

PRINT “This Program computes the perimeter of a rectangle.”
PRINT “Please enter the length and width of the rectangle, ”
PRINT “separated by commas. ”
INPUT L,W or INPUT “Length =“, L
INPUT “Width =“, W
PRINT “The Perimeter is”; 2*L+2*W
END
## Non-Trigonometric Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Exponentiation</td>
<td>PRINT $7^3$</td>
<td>343</td>
</tr>
<tr>
<td>ABS (x)</td>
<td>Absolute Value (converts a negative number to a positive number)</td>
<td>PRINT ABS (16.2) PRINT ABS (-27.3)</td>
<td>16.2 27.3</td>
</tr>
<tr>
<td>EXP (x)</td>
<td>Natural Exponent (e*)</td>
<td>PRINT EXP (4)</td>
<td></td>
</tr>
<tr>
<td>FIX (x)</td>
<td>Integer Truncation (chops off anything after the decimal point)</td>
<td>PRINT FIX (4.728)</td>
<td>4</td>
</tr>
<tr>
<td>INT (x)</td>
<td>Largest Integer Truncation</td>
<td>PRINT FIX (-4.728)</td>
<td>4</td>
</tr>
<tr>
<td>LOG (x)</td>
<td>Natural Logarithm (inverse of Natural Exponent)</td>
<td>LOG(10)</td>
<td>1</td>
</tr>
<tr>
<td>RND (x)</td>
<td>Random Number, produce a random number from 0 to 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGN (x)</td>
<td>Sign Determination</td>
<td>PRINT SGN (14) PRINT SGN (0) PRINT SGN (-244)</td>
<td>1 0 -1</td>
</tr>
<tr>
<td>SQR (x)</td>
<td>Square Root</td>
<td>PRINT SQR (625)</td>
<td>25</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Example</td>
<td>Result</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| COS (x)  | Cosine                          | CONST PI=3.141593
          | PRINT COS (PI / 4)           | 0.7071067    |
| SIN (x)  | Sine                            | PRINT SIN (PI / 3)          | 0.8660254    |
| TAN (x)  | Tangent                         | PRINT TAN (-PI / 2)         | 6.137956     |
| ATN (x)  | Arctangent, inverse of TAN      | PRINT ATN (TAN (-PI / 2))   | 1.570796     |
General Form
IF condition THEN
  statements
END IF

IF condition THEN
  statements
ELSE
  statements
END IF

Purpose: One-way selection
Purpose: Two-way selection

Used to compare an expression and then perform some task based on that expression. ELSE: using this command enable the program to perform a different action if the statement is false. END IF: allows to have multiple commands after the IF...THEN statement, but they must start on the line after the IF statement. END IF should appear right after the list of commands. ELSEIF: this command allows to perform a secondary action if the first expression was false. Unlike ELSE, this task is only performed if a specified statement is true.

This command checks if an argument involving a variable is true. An argument may look like this: IF b=18 THEN...

If the argument is true (b=18), then QBasic executes the command after the IF...THEN.
IF...THEN asks only yes or no questions. It's either true or false.
- If the statement is true, the command(s) after the THEN are executed.
- If the statement is false, the command(s) after the THEN command are skipped.
- **AND** and **OR** can make IF...THEN commands more powerful by making the command ask more questions.
- Answers to questions asked about string variables must be in quotes. ("")
- The **ELSE** command follows the THEN command, and is executed if the IF...THEN question is false.

**Example:**

IF b=18 THEN PRINT "OK 
If the argument is not true (if b is not equal to 18), QBASIC bypasses this line and goes to the next. In some cases, ELSE command could be used which tells QBASIC exactly what to do if the argument is not true.

IF b=18 THEN
    PRINT "OK "
ELSE
    PRINT "It is not 18 "
END IF

**Example:**

X=16
IF (X=5) THEN
    INPUT a$
    PRINT a$
ELSE
    PRINT X*2
END IF

Output:
32
Example:

1 CLS
score=0
PRINT “How many days are there in a week?”
INPUT a
IF a=7 THEN
   GOTO 2
ELSE
   PRINT “Wrong answer!”
   PRINT “To try again – press y”
   INPUT a$
   IF a$= “y” THEN
      PRINT “OK, let’s try again.”
      GOTO 1
   ELSE
      END
   END

2 score =10
PRINT “It’s the right answer!”
PRINT “Your score is now”; score; “!”
PRINT “Thanks for playing.”
END

Example:
CLS
X=5
IF (X >= 5) THEN PRINT “X is greater than or equal to 5”
IF (X <= 5) THEN PRINT “X is less than or equal to 5”
IF (X <> 5) THEN PRINT “X does not equal to 5”

Output:
X is greater than or equal to 5
X is less than or equal to 5

Example:
G=8
IF (G=5) THEN
   PRINT “Statement 1 is true “
ELSEIF (G=6) THEN
   PRINT “Statement 2 is true “
ELSEIF (G=7) THEN
   PRINT “Statement 3 is true “
ELSE
   PRINT “No above statements are true “
END IF

Output:
No above statements are true
IF...THEN [Multiple expressions]:

You can have more than one expression in IF...THEN by using either the OR operator or the AND operator. The OR operator only requires one expression to be true in order to print “Yes” in the following program:

K=20
IF (K=5 OR K=20) THEN PRINT “Yes”

Output:
Yes

The AND operator requires both expressions to be true.

Example:
K=7
IF (K>5 AND K<20) THEN PRINT “True”

Output:
True

Example:
X=16
Y=3
IF ((X>5 AND X<10) OR Y=3) THEN PRINT “Correct”

Output:
Correct
Using Strings in IF...THEN:

Example:

M$="Hello “
IF (M$= "Hello “ OR M$= “World “) THEN PRINT M$

Output:
Hello

Also, You can compare two variable strings.

Example:

M$="Hello “
Z$= “World “
IF (M$<>Z$) THEN PRINT M$;“ “;Z$

Output:
Hello World
Read / Data Statements
READ / DATA statements are another way of assigning values to variables. When the computer comes to a READ statement it looks for the DATA statement.

Example:

REM Read and Data statements
READ x, y, z
PRINT "x="; x
PRINT "y="; y
PRINT "z="; z
PRINT "sum= "; x + y + z
DATA 10,20,30
END

REM Words as Data
READ A$, B$, C$
PRINT A$, B$, C$
DATA "Material","Density"
DATA "Hardness"
END

Read numeric data
Read string data

Loading and processing Data
Lecturer: Dr. Laith Abdullah Mohammed
**Example:** All the following programs processing just one set of data, it computes the perimeter of only one rectangle.

DATA 6.5, 2.3
READ L, W
LET P = 2*L+2*W
PRINT “Perimeter =”, P
END

READ L, W
DATA 6.5, 2.3
LET P = 2*L+2*W
PRINT “Perimeter =”, P
END

READ L, W
DATA 6.5, 2.3
LET P = 2*L+2*W
DATA 6.5, 2.3
PRINT “Perimeter =”, P
END

The following program compute the perimeter for three rectangles.

TOP:
    READ L, W
    LET P = 2*L+2*W
    PRINT “Perimeter =”, P
    GOTO TOP
DATA 6.5, 2.3, 7.86, 6.03, 21, 17
END
**Program: Find the Largest Number:**

MORE:

    READ X
    IF X=-1 THEN GOTO FINISH
    IF X>LARGEST THEN GOTO UPDATE
GOTO MORE
UPDATE:

    LARGEST = X
GOTO MORE
FINISH:

    PRINT “Largest is ”; LARGEST
END

**Program: Find Consecutive Products.**

MORE:

    READ N
    O=N+1
    P=N+2
    Q=N+3
    PRINT N*O*P*Q
GOTO MORE
END

**Program: Find the average of two numbers.**

MORE:

    READ A,B
    LET SUM=A+B
    PRINT SUM/2
GOTO MORE
DATA 1,3,2,4,57,122,6,-9
END
## Exercises:

### REM Reading Data (Numeric)
READ A, B
PRINT A; " x "; B; "=": A * B
DATA 5,4
END

### REM Reading Data (Text)
READ Name$, Age
PRINT Name$; " is": Age
DATA "Mohammed",4
END

### REM area of rectangle
READ length, wide
area = length * wide
PRINT "Length= "; length
PRINT " Width= "; wide
PRINT " Area= "; area
DATA 12,6
END

### REM Average Calculation
REM variables used
REM avg=Average , sum=sum
REM n1,n2,...,n5 the scores
READ n1, n2, n3, n4, n5
sum = n1 + n2 + n3 + n4 + n5
avg = sum / 5
PRINT "Average of":
PRINT n1; n2; n3; n4; n5
PRINT "equals": avg
DATA 3,4,5,6,7
END
**GOTO:**

The Syntax is:  **GOTO** [line label or number]

GOTO is a command that tells the computer to go to another place in the program, and continue executing the statements. GOTO tells the computer to find a line number or label, and start reading from there.

**Example:**

Top:

```plaintext
CLS
b=12
h=8
PRINT "Calculate the area for two geometries"
55 INPUT "1. Parallelogram  2. Triangle "; Choice
IF Choice = 1 THEN GOTO Parallel
IF Choice = 2 THEN GOTO Tri
GOTO What
```

Parallel:  PRINT "Area of Parallelogram :"; b*h
END

Tri :  PRINT "Area of Triangle :"; 0.5*b*h
END

What:  PRINT "Try again….to calculate"
GOTO 55
Example 1:

Again:

PRINT “Calculate 1/n for any number (n)”
PRINT “(type a 0 to end)”
INPUT “Please Enter n:”, n%
IF n% = 0 THEN GOTO nowend
Answer# = 1# / n%
PRINT “The answer is:”; Answer#
PRINT
PRINT “Do you want to enter another n?”

TypeAgain:

INPUT “(Y/N):”, yesorno$
IF yesorno$ = “Y” THEN
GOTO Again
ELSEIF yesorno$ = “N” THEN
GOTO nowend
ELSE
GOTO TypeAgain
END IF

nowend:

END
Example 2:

CLS
start:
PRINT "Guess a number between 1 and 10: ";
INPUT num
IF (num < 1 OR num > 10) THEN
PRINT "That is not between 1 and 10"
GOTO start
END IF
IF (num = 6) THEN
PRINT "Correct!!!"
ELSE
PRINT "Try again"
PRINT
GOTO start
END IF

Output:
Guess a number between 1 and 10: ? 2
Try again
Guess a number between 1 and 10: ? 7
Try again
Guess a number between 1 and 10: ? 6
Correct!!!
The syntax is: **ON**  index  **GOTO**  N1,N2,N3,…….,Ni

- **index**: is an integer number
- N1, N2, N3,….., Ni: The Label number of step.

**Example:**
Write a program to calculate the square, cube and square root of a number.

**Solution:**
INPUT “Input the number =“; x
INPUT “Press 1 for square, 2 for cubic, 3 for square root”; k
ON k GOTO 10,20,30
10   Y=x^2
    GOTO 100
20   Y=x^3
    GOTO 100
30   Y=SQR(x)
    GOTO 100
100  PRINT “The result number is=“;Y
END
Example: Write a program to calculate y from the equations below:
\[ y = x^2 - x + 10 \] when \( x = 1 \) or \( x = 5 \)
\[ y = 20x + 12 \] when \( x = 2 \) or \( x = 4 \)
\[ y = 1 - x - x^3 \] when \( x = 3 \)
\[ y = 127 \] when \( x = 6 \)

Solution:
5   CLS
6   INPUT “Enter the value of x =”; x
ON x GOTO 10,20,30,40,50,60
PRINT “Invalid value of x….Try again”
GOTO 6
10   y=(x^2)-x+10
GOTO 70
20   y=20*x+12
GOTO 70
30   y=1-x-(x^3)
GOTO 70
40   GOTO 20
50   GOTO 10
60   y=127
70   PRINT “X= “;x,”Y= “;y
END
DECISION MAKING / SELECT CASE

It provides a better program structure for multiple decisions/alternatives.

The syntax is:

```
SELECT CASE varname
CASE option [,option,]
CASE ELSE
END SELECT
```

- **SELECT CASE** varname
  - Defines the beginning of a Select-part. Varname is the variable which has to be checked using this Select structure.

- **CASE option [,option,]**
  - Case defines a code part for if Varname = one of the options specified.

- **CASE ELSE**
  - The code Case Else will only be executed if no other specified cases are true. Also, the code that will be executed when a Case is true, will be everything until a next Case or an End Select is reached.

- **END SELECT**
  - Denotes the end of a select structure.

Example:

```
PRINT “Please enter a number”
INPUT “Number:”, number%
SELECT CASE number%
  CASE 1
    PRINT “The number is 1”
  CASE 2, 3
    PRINT “The number is either 2 or 3”
  CASE IS > 3
    PRINT “The number is greater than 3”
  CASE IS < 0
    PRINT “The number is smaller than 0”
  CASE ELSE
    PRINT “unknown number”
END SELECT
END
```

- **CASE 1**: Single constant (numeric or string)
- **CASE 2,3**: Multi constants (numeric or string)
- **CASE IS >3**: using IS with comparison operators
Example:
INPUT "Type Y for yes or N for no"; choice$
SELECT CASE choice$
CASE IS = "Y"
    PRINT "Your choice is YES"
CASE IS = "N"
    PRINT "Your choice is NO"
END SELECT
END

Example: Write a program to assign the following grades to students examination results.

CLS
INPUT "Enter the score : "; Score
SELECT CASE Score
CASE IS >= 90
    Grade$ = "A"
CASE IS >= 80
    Grade$ = "B"
CASE IS >= 70
    Grade$ = "C"
CASE IS >= 60
    Grade$ = "D"
CASE ELSE
    Grade$ = "F"
END SELECT
PRINT "The Grade is = "; Grade$
END

<table>
<thead>
<tr>
<th>Score</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater or equal to 90</td>
<td>A</td>
</tr>
<tr>
<td>Greater or equal to 80</td>
<td>B</td>
</tr>
<tr>
<td>Greater or equal to 70</td>
<td>C</td>
</tr>
<tr>
<td>Greater or equal to 60</td>
<td>D</td>
</tr>
<tr>
<td>Otherwise</td>
<td>F</td>
</tr>
</tbody>
</table>
Example:
Write a program to make a decision of weather condition according to the month.

<table>
<thead>
<tr>
<th>Month</th>
<th>Weather condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>November, December, January, or February</td>
<td>Cool season</td>
</tr>
<tr>
<td>March, April, or May</td>
<td>Hot season</td>
</tr>
<tr>
<td>June to October</td>
<td>Wet season</td>
</tr>
</tbody>
</table>

INPUT “Enter the number of the month :”; Month
SELECT CASE Month
CASE 11, 12, 1, 2
    PRINT "Cool season"
CASE 3 TO 5
    PRINT "Hot season"
CASE 6 TO 10
    PRINT "Wet season"
END SELECT
END
Example: Write a program to solve the following set of equations, using SELECT CASE statement.

\[
y = \begin{cases} 
    x_1 + x_2 & x_1 < x_2 \\
    x_1 \cdot x_2 & x_1 = x_2 \\
    x_1 - x_2 & x_1 > x_2 
\end{cases}
\]

The Program:

INPUT "X1 and X2", x1, x2

IF x1< x2 THEN
    U = 1
END IF

IF x1= x2 THEN
    U = 2
END IF

IF x1> x2 THEN
    U = 3
END IF

SELECT CASE U
    CASE IS = 1
        y = x1 + x2
    CASE IS = 2
        y = x1 * x2
    CASE IS = 3
        y = x1 - x2
END SELECT

PRINT “The Result : “; y
It used to execute a part of a program a certain number of times.

```
FOR variable=start TO end, step-size
  statements
NEXT variable

Example 1:
FOR NUMBER = 1 TO 3
  PRINT “This is loop number “; NUMBER
NEXT NUMBER
END

The output will be:
This is loop number 1
This is loop number 2
This is loop number 3
```

```
Example 2:
FOR j = 0 TO 6 STEP 1.5
  PRINT j
NEXT j
END

The output will be:
0
1.5
3
4.5
6
```

```
Example 3:
FOR wg = 4 TO 1 STEP -1
  PRINT wg
NEXT wg
END

The output will be:
4
3
2
1
```
Example 4:

CLS
REM for/Next statements
PRINT "Number", "Square"
FOR number = 1 TO 5
PRINT number, number ^ 2
NEXT
END

The output will be:

<table>
<thead>
<tr>
<th>Number</th>
<th>Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

Example 5:

This example illustrates using a negative step to assign decreasing values to a variable.

CLS
REM FOR/NEXT with negative STEP
REM count=counter
FOR count = 10 TO 5 STEP -1
PRINT count
NEXT
END

The output will be:

10
9
8
7
6
5
The following program uses a READ/DATA statement within a loop. Line numbers are listed to identify individual lines.

5 CLS
10 REM READ/DATA within a Loop
20 PRINT "Numbers:"
30 FOR k = 1 TO 5
40    READ n
50    PRINT n;
60    s = s + n
70 NEXT k
80 PRINT
90 PRINT "Average: "; s / 5
100 DATA 7,10,12,15,6,9,.5
END

The Output will be:
Numbers: 7 10 12 15 6
Average: 10

In this example the following points should be noted:
• Line 60 has the effect of saying "Let the new value of S be the old value of S plus N"
• Not all of the data in line 100 is used. The loop is only executed 5 times so only the first five values from the data statement are read.
• Line 80 is necessary to stop the output from line 90 appearing on the same line as the numbers
• The variable S was not given an initial value so the computer gives S the initial value of 0 (zero).
The following example illustrates the use of an input statement within a loop.

REM Input within a Loop
REM Averaging three Numbers
FOR counter = 1 TO 3
   INPUT "Enter a number: "; num
   sum = sum + num
NEXT counter
PRINT
PRINT "Average of the three numbers= "; sum / 3
END

This program could be modified to let the person running the program choose the number of numbers to be averaged while the program was running. See below:

REM Input within a Loop
REM Averaging three Numbers
INPUT "How many Numbers to Enter:"; k
FOR counter = 1 TO k
   INPUT "Enter a number: "; num
   sum = sum + num
NEXT counter
PRINT
PRINT "Average of the "; k; "numbers= "; sum / k
END

The output will be:
Enter a number: 4
Enter a number: 5
Enter a number: 3
Average of the three numbers= 4

The output will be:
How many Numbers to Enter: 3
Enter a number: 4
Enter a number: 5
Enter a number: 3
Average of the 3 numbers= 4
Exercises:

```
READ A, B, C
FOR N = A TO B STEP C
P = N * B
PRINT N; "Times"; B; "="; P
NEXT N
DATA 1,10,3
END

FOR N = 1 TO 5
READ A
PRINT A
NEXT N
DATA 2,4,6,8,10,12,14
END

PRINT "N", "N^2", "SQR(N)"
FOR N = 1 TO 4
PRINT N, N ^ 2, SQR(N)
NEXT N
END
```

The output will be:

```
1 Times 10 = 10
4 Times 10 = 40
7 Times 10 = 70
10 Times 10 = 100
```

```
2
4
6
8
10
```

```
N | N^2 | SQR(N)
---|-----|-----
1  | 1   | 1
2  | 4   | 1.414214
3  | 9   | 1.732051
4  | 16  | 2
```
CLS
a = 4
B = 10
C = 2
FOR N = a TO B STEP C
FOR W = 1 TO 4
  P = N * W
  PRINT N; "Times"; W; "="; P
NEXT W
PRINT "---------------------------------
NEXT N
END

The output will be:

4 Times 1 = 4
4 Times 2 = 8
4 Times 3 = 12
4 Times 4 = 16
---------------------------------
6 Times 1 = 6
6 Times 2 = 12
6 Times 3 = 18
6 Times 4 = 24
---------------------------------
8 Times 1 = 8
8 Times 2 = 16
8 Times 3 = 24
8 Times 4 = 32
---------------------------------
10 Times 1 = 10
10 Times 2 = 20
10 Times 3 = 30
10 Times 4 = 40
---------------------------------
S = 0
FOR N = 1 TO 6
    READ a
    PRINT a;
    S = S + a
NEXT N
PRINT
PRINT "Total"; S
DATA 5,9,-3,7,12,8
END

Write a program to print even numbers up to 20 and find the total?

CLS
total=0
FOR i=0 TO 20 STEP 2
    total = total + i
PRINT i ;
NEXT i
PRINT
PRINT "total is "; total
END

The output will be:
5 9 -3 7 12 8
Total 38

The output will be:
0 2 4 6 8 10 12 14 16 18 20
Total is 110
Example: Use FOR…NEXT loop in a program to calculate the sum of the first $n$ terms of the series, then execute the program for $n=4$ and $n=20$:

$$\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$$

Solution:

INPUT "Enter the number of the terms ="; n
S=0
FOR k=1 TO n
S=S+((-1)^k)*k/(2^k)
NEXT k
PRINT "The summation of the series is ="; S

The output:
Enter the number of the terms = 4
The summation of the series is = -0.125
Enter the number of the terms = 20
The summation of the series is = -0.222