Data Representation

- Numbers are represented in binary form (i.e. 0's & 1's).
- Individual objects can be assigned some unique integer index. (e.g., each English character is assigned an unique integer index.)
- Multimedia information

  - Text and numbers
  - Graphics
  - Audio
  - Images
  - Video

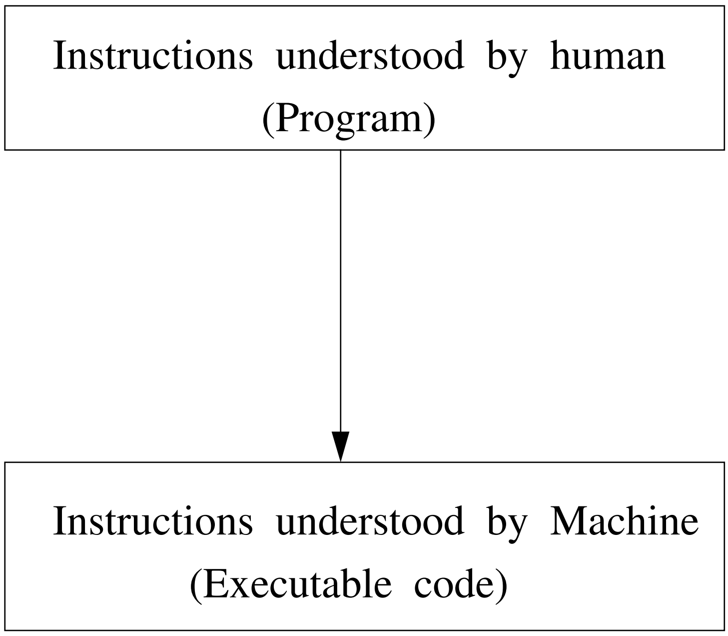
NUMBER SYSTEMS

Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

DATA REPRESENTATION

- Bit (or binary digit):
  - Basic unit for storing data in computer memory ("0" for OFF, "1" for ON)
- Byte:
  - 8 bits
  - Each byte usually represents one character of data (number, letter, special character)
  - Unit to express capacity of memory:
    - 1 kilobyte (KB) = 1024 bytes
    - 1 megabyte (MB) = 1024 x 1024 bytes
    - 1 gigabyte (GB) = 1024 x 1024 x 1024 bytes
- Word:
  - Number of bits that constitute a common unit of data in a computer system
  - 1 computer word = 16 bits for 16-bit machines, = 32 bits for 32-bit machines, and so on

Compiling


Instructions understood by human
(Program)
Instructions understood by Machine
(Executable code)

Machine Languages

- All programming languages must be translated into the computer's machine language, which is the only language that can be executed directly.

- An instruction is coded as a sequence of 0's and 1's.

- Machine language programs are cumbersome and difficult to write.

- Example of a machine language instruction:

01101	01100101000	01100101010	01100101111
ADD operation	Address of 1st operand	Address of 2nd operand	Address to store result

Assembly Languages

- An assembly language is the symbolic version of the computer's machine language.

- An assembler translates an assembly language to the corresponding machine language.

- Assembly languages use easily recognized symbols to represent instructions. (e.g. MOV for move operation, MUL for multiplication.)

- Example of an assembly language instruction:

ADD	OP1	OP2	SUM

High-level Programming Languages

- Closer to natural languages than machine/assembly languages.

- Translated into executable form using compilers or interpreters.

- A compiler translates the entire source program written in a high-level language into machine-oriented instructions for later execution.

- An interpreter translates and executes each source program instruction before translating and executing the next.

- Interpreters facilitate program development, although interpreted code usually executes more slowly than compiled code.

COMPUTER GENERATIONS

- First generation (1945 - 1954):
  - First commercial computer (using vacuum tubes) in 1951
  - Machine languages
  - Single-user machines
- Second generation (1955 - 1964):
  - Transistors
  - Assembly (or symbolic) languages
  - Single-user machines
- Third generation (1965 - 1974):
  - Integrated circuits (ICs)
  - Problem- and procedure-oriented languages
  - Time-sharing and multiprogramming to support multiple users

COMPUTER GENERATIONS (cont'd)

- Fourth generation (1975 - 1990):
  - Large scale integrated (LSI) and very large scale integrated (VLSI) circuits --> microcomputers
  - Compact-disc read-only-memory (CD-ROM)
  - Data communications
  - Packaged software
  - Graphical user interface
- Fifth generation (1991 - present):
  - Ultra large scale integrated (ULSI) circuits
  - Massively parallel processing (MPP)
  - Scalable architectures
  - Optical technologies
  - Heterogeneous processing over networked computers
  - Large-scale applications

Software

Compilers
Pascal, Fortran, C++

Applicaton Programs
Editor, Calculator

Operating System
DOS, Unix

What can we do at the end of the course?

Be able to program a computer - A program is a set of instructions that direct the actions of a computer.

1. Program a simple calculator.
2. Program a computer to play simple games.
3. Write a program to manipulate polynomials and matrices.
4. Program a simple text editor.
5. Program a small inventory system for a store.

A Programmer's Tasks

1. WRITE THE PROGRAM (SOFTWARE)

   Use the Editor to prepare a program file

   (source code, source file, etc)

2. COMPILE THE PROGRAM

   Use the C++ compiler (a piece of software) to

   generate executable code (stored in a different file)

3. RUN THE PROGRAM

   Usually just type the name of the file containing

   the executable code.

NOTE : Borland C++ offers an integrated environment

to perform the above tasks.

Programming as a Problem Solving Process

- Define and analyze the problem.
- Develop an algorithm.
- Implement a program.
- Compile, test, and debug the program.
- Document and maintain the program.