Lecture 1: An Overview of Computers and Programming Languages

Part I: An Overview of Computers
1. What is a computer?
   Computer - device that solves problems by:
   1) accepting data. e.g. numbers to add
   2) performs prescribed operations on data e.g. actual adding
   3) supplies the results of these operations

2. History of computers.
   1) 1950s: very few computers;
   2) 1960s: very large, expensive computers;
   3) 1970s: cheaper, smaller computers.
   4) Today: computers are affordable and faster.

3. Types of computers:
   mainframe, midsize, and micro computers

4. Elements of a computer system.
   1) Software
      The computer instructions that form a computer program.
      A. System programs control the computer
         e.g. operating system: software that enables us to use the computer hardware.
      B. Application programs perform specific tasks.
   2) Hardware
      Actual physical devices that make up the computer.

![Diagram of computer hardware components]

Figure 1-1  Hardware components of a computer
A. central processing unit (CPU): brain of the computer
   - control unit (CU)—fetches and decodes instructions, controls the flow of
     information in and out of main memory, and controls the operation of the CPU’s
     internal components
   - arithmetic logic unit (ALU)—carries out arithmetic and logical operations
     - Arithmetic operations: Add, Subtraction, Division, Multiplication
     - Logical operations: yields a true or false answer. e.g. 2 is larger than 4
   - program counter (PC)—points to the next instruction
   - instruction register (IR)—holds the instruction that is being executed
B. main memory: All instructions and data must be loaded into main memory each time a
   program is run. Main memory is called volatile because when the computer is turned
   off, the information in memory is lost. Each location in memory is identified by its
   address much like a group of post office boxes.
C. secondary storage: Since main memory is only so large, and usually expensive, extra
   information needed to be processed in the near or distant future resides on peripherals
   called secondary memory. These include tape drives, floppy disk drives, hard disk
   drives, CD ROM, etc
D. input/output devices: allow for the user to communicate with the computer. e.g.
   keyboard, punched cards, scanner(input), monitor, printer (output). These devices are
   sometimes called peripherals.

Part II: An Overview of Programming Languages
1. The language of a computer
   In modern computers, all data is stored and manipulated as a sequence of 0s and 1s; each called a
   bit. A sequence of eight bits is called a byte.
   We use the decimal system for counting and the computer uses the binary system.
2. Evolution of programming languages
   1) Low Level Languages - machine dependent. Based on the computer hardware or
      architecture.
      - machine language - sequences of 0's and 1's. The only language that computers can
        understand.
      - assembly language - used mnemonics to represent sequences of 0's and 1's. Programs
        called assemblers translated these mnemonics into machine language. Assembly
        language is also low level.
   2) High Level languages - more "English" like. These languages are machine independent.
      The program code written in these languages is known as source code or source programs
      - e.g. COBOL, Pascal, FORTRAN, Logo, C, C++, BASIC, ADA, LISP, JAVA, PERL.
      - The computer cannot directly execute instructions written in a high-level language.
      - To be executed by a computer, high level languages must be translated. The program
        that does the translating is machine dependent.
 Compiled language - The language is translated into machine code by a program called a **compiler**. All source code is translated, saved, and then executed. *e.g.* FORTRAN, C++, COBOL

 Interpreted language - when each program statement is translated and then immediately executed. Program that does this is called an Interpreter. *Ex.* BASIC, LISP

 High level languages can be divided into two catalogs:

 Procedural languages - programs are written as a set of functions or procedures that accomplish a specific task. Each procedure/function, accepts data and manipulates the data in some way to produce a desired result. *Ex.* Pascal, Cobol

 Object-Oriented languages (OOP) - Programs are made up of objects that fully describe the data and functions associated with it. *Ex.* C++ (not just a OOP), JAVA, Smalltalk

3. Steps for creating and executing a high level program.

1) Use an editor, such as notepad or one that comes with the compiler, to type in the high level language instructions. This becomes the source code or source program.

2) Use the compiler to check for syntax errors. Syntax errors occur when you "break the rules" of the programming language.

3) If there are no syntax errors, the compiler translates the source code into machine language. The resulting machine code is called an object program.

4) Many tasks that need to be done by a computer program are already written and do not need to be recoded by the programmer. An example of this is the code that displays the results of a program onto the computer screen. The codes for these tasks reside in libraries supplied by the compiler. The linker bundles the object code and the library code into an executable file.

5) The loader then places the executable program into main memory where it can be executed.

![Figure 1-3 Processing a high-level language program](image-url)