

# Some Fundamentals (Review)

## Programming

- ◆ Computers are really very dumb machines -- they only do what they are told to do.
- ◆ Most computers perform their operations on a very primitive level.
- ◆ The basic operations of a computer system is called the computer's **instruction set**.
- ◆ In order to solve a problem using a computer, we must express the solution to that problem in a language that the computer can understand - through the instruction set.

- ◆ A computer **program** is just a collection of the instructions necessary to solve a specific problem.
- ◆ The approach or method that we use to solve the problem is called an **algorithm**.
- ◆ To develop a program to solve a particular problem, we first express the solution to the problem in terms of an algorithm.
- ◆ With the algorithm in hand, we can then write the instructions necessary to implement the algorithm on a particular computer system.

"A computer is like a violin. You can imagine a novice trying first a phonograph and then a violin. The latter, he says, sounds terrible. That is the argument we have heard from our humanists and most of our computer scientists. Computer programs are good, they say, for particular purposes, but they aren't flexible. Neither is a violin, or a typewriter, until you learn how to use it."

— Marvin Minsky, "Why Programming Is a Good Medium for Expressing Poorly-Understood and Sloppily-Formulated Ideas"

## Example - Algorithms

**Problem:** Develop a program that tests if a number is even or odd.

**Algorithm #1:** Divide the number by two.

If the remainder is zero, then the number is even.

Otherwise (the remainder is one), the number is odd.

**Algorithm #2:** Test the least significant bit of the number.

If the bit is one, then the number is odd.

Otherwise (the bit is zero), the number is even.

---

# Higher-Level Languages

- ◆ When computers were first developed, the only way they could be programmed was with binary numbers that corresponded directly to the machine instructions and locations in the computer's memory. (**Machine language**).
- ◆ **Assembly language** enabled the programmer to work with the machine on a slightly higher level.
- ◆ A special program called an **assembler** translates the assembly language programs from its symbolic format into machine language.
- ◆ Because a one-to-one correspondence exists between assembly language instructions and machine language instructions, assembly language is called a **low-level language**.

- ◆ The programmer must still learn the instruction set of the particular computer system in order to write a program.
  - ◆ Assembly language programs are not **portable** -- they will not run on a different type of computer without being rewritten (**machine-dependent**).
  - ◆ Operations of a **higher-level language** are much more sophisticated -- one **statement** would result in many different machine instructions being executed.
  - ◆ Standardization of the syntax of a higher-level language mean that a program could be written to be **machine independent** -- a program could run on any machine that supported the language with few or no changes.
  - ◆ A **compiler** is a special program that translates the statements of a program developed in a higher-level language into machine language.
-

# Operating Systems

- ◆ An **operating system** is a program that controls the entire operation of a computer system.
- ◆ All **input/output (I/O)** operations are channeled through the operating system.
- ◆ The operating system must also manage the computer's resources and must handle the execution of programs.

# Compiling Programs

- ◆ A compiler is a program that analyzes a program developed in a particular computer language and then translates it into a form that is suitable for execution on your particular computer system.
- ◆ The program that is to be compiled is first typed into a **file** on the computer.
- ◆ A text editor must usually be used to enter a program into a file.
- ◆ The program that is entered into the file is known as the **source program** or **source code**.
- ◆ Once the source program has been entered into a file, you can then have it **compiled**.

- ◆ The compilation process is initiated by typing a special command. When this command is entered, the name of the file that contains the source program must also be specified.
- ◆ Step 1: the compiler examines each statement in the source program and checks for **syntax** and **semantic** errors.
- ◆ Step 2: the compiler translates each statement to a “lower” form -- usually to assembly language, or directly to machine language. This translation is called **object code**.
- ◆ Step 3: After the program has been translated into object code, it is ready to be **linked** -- the program is joined with other programs that have been previously written and programs from the operating system’s **library**.
- ◆ Step 4: The final linked file is called an **executable** file, and is stored in another file on the system -- ready to run or be **executed**.

- ◆ To execute (run) the program, all you have to do is type in the name of the executable file. This **loads** the program into the computer's memory and starts execution.
- ◆ When the program is executed, each of the statements of the program is **sequentially** executed.
- ◆ If the program requests any data from the user, called **input**, the program will temporarily suspend its execution and wait for the user to enter the data.
- ◆ Results that are displayed by the program are called **output**. (Normally displayed on the screen or terminal).
- ◆ If all goes well (and it probably won't the first time the program is executed), the program will do what it is supposed to do.

- ◆ If things go wrong, it will be necessary to go back and re-analyze the program's logic. This is called **debugging** -- you try to remove all the **bugs** (features?) from the program.
- ◆ In order to **debug** a program, the source code usually has to be changed. Then the entire process of compiling, linking, and executing the program is repeated...

"99 bugs in the code,  
99 little bugs  
Fix one bug, recompile  
100 bugs in the code."

: -)