Chapter - 2
The Basics of Programming

What is a program?

A program is a set of instructions that a computer or someone else follows.

We have programs in English. But English is a lousy language when it comes to writing exact instructions. The language is full of ambiguity and imprecision. Grace Hopper, the grand old lady of computing, once commented on the instructions she found on a bottle of shampoo:

> Wash, Rinse, Repeat

She tried to follow the directions, but she ran out of shampoo. (Wash-Rinse-Repeat. Wash-Rinse-Repeat. Wash-Rinse-Repeat...)

Machine and Assembly Language

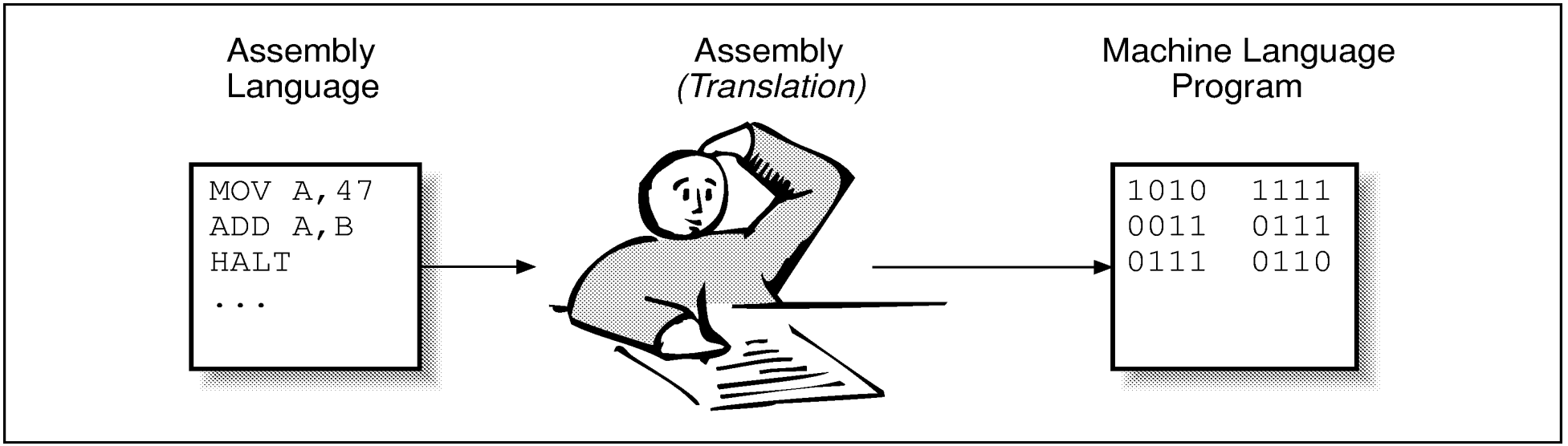
When computers cost millions and programmers cost $15,000 a year, people programed in *machine languag*e:

```
1010 1111
0011 0111
0111 0110
.. and so for several hundred instructions
```

Later they devised a translator called an assembler so they could program in *assembly language.*

```
Program       Translation
MOV A,47   1010 1111
ADD A,B 0011 0111
HALT    0111 0110
.. and so for several hundred instructions
```

What an assembler does



Note: The first programmer who wrote a program to assemble code was chewed out because, "How dare you even think of using such an expensive machine for a mere 'clerical' task."

High Level Language

High level languages were developed to:

- Make programming easier. A single high level language could generate many assembly instructions.

  For example:

  ```
  area = width * height
  ```

  generates:

  ```
  MOVE D0, HEIGHT
  MOVE D1, WIDTH
  MUL D0,D1
  MOVE AREA,D0
  ```

- Make programming *machine independent*. The idea was to hide the details of the machine from the programmer.
- Allow the programmer to program in a language that was more natural to him.

High Level Languages

Some of the high level languages developed are:

FORTRAN

(FORmula TRANslator) designed to perform scientific calculations.

COBOL Useful for writing business reports.

PASCAL A language for teaching students.

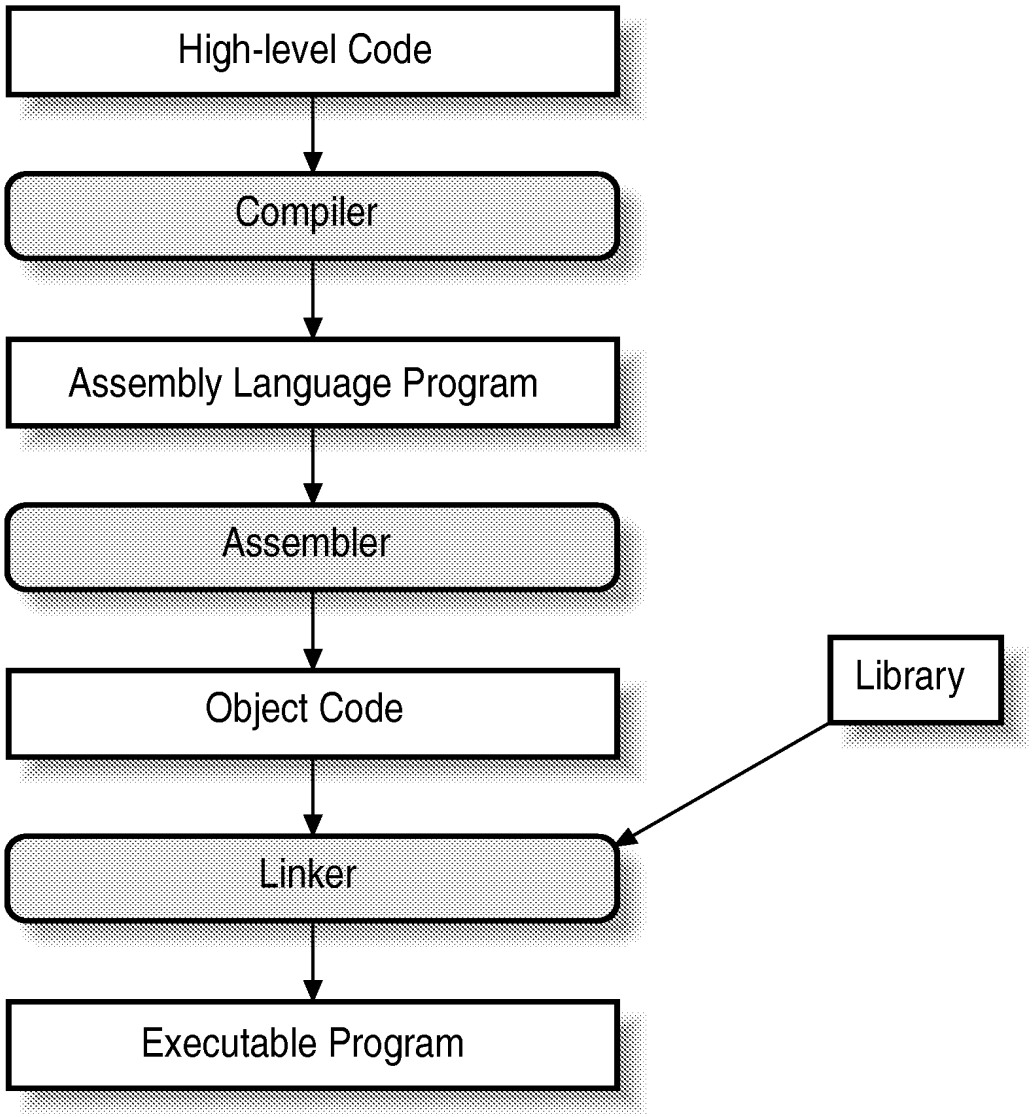
C Designed by hackers for hackers to hack with (to write operating systems)

Development of C++

In 1980 Bjarne Stroustrup started working on a new language, called “C with Classes.” This language improved on C by adding a number of new features, the most important of which was classes. This language was improved, augmented, and finally became C++.

C++ owes it success to the fact that it allows the programmer to organize and process information more effectively than most other languages. Also, it builds on the work already done with the C language. In fact, most C programs can be transformed into C++ programs with little trouble. True they don’t use all the new features of C++, but they do work. So C++ allows people to build on a existing base of C code.

Construction of a Program



Construction Tools

- **Text Editor**

  This is used to create the program in C++ form. Since this is the start or source of the other forms this is called a *source file*. (Source files end with `.cpp.` -- also used `C` and `.cc`.)

```
#include <iostream>
int main()
{
    std::cout << "Hello World\n";
    return (0);
}
```

- **Compiler**

  This translates the *source file* into a *machine dependent* file called an *object file*. The object file contains the instructions in a way that the machine can understand.

  The source file is in the C++ language (high level code) while the object file is in machine language (low level code.)

More Construction Tools

- **Library**

  One of the goals of a high level language is to create resusable code modules. The most useful of these have been put in a standard library for your use.

- **Linker**

  The linker takes the object file produced by the compiler, combines it with entries from the library (linking) and produces a executable program.

  On MS-DOS/Windows executable programs end with `.EXE`. On UNIX, they end with nothing.

- **Make utility**

  The *make* utility is designed to be the programmer's assistant. It helps him keep track of what compiler commands to use in the construction of a program.

Debuggers

- **Debugger**

  The debugger allows a programmer to examine his program while it is running. Some of the features of a debugger include:

  - **Breakpoints**. They allow the programmer to stop the program on any line.
  - **Single step**. Executes the program one line at a time.
  - **Display**. Allows the programmer to print the value of variables and expressions at any time.

- **Browser**

  Allows the programmer to display relevant portions of the source file.

Wrappers

Wrappers or Integrated Development Environments try to combine all the utilities needed to create a program under one roof.

For example, under UNIX the CC command is a wrapper that will call the compiler, linker and other programs automatically.

Under MS-DOS/Windows both Borland and Microsoft have integrated development environments that combine editor, compiler, linker, and debugger all under one GUI.

Hello World

```
#include <iostream>
int main()
{
    std::cout << "Hello World\n";
    return (0);
}
```

Compiling the program using the UNIX CC compiler (GENERIC UNIX)

Use the command:

```
% CC -g -ohello hello.cpp
```

Details

`CC`	The compiler
`-g`	Turns on debugging
`-ohello`	Tells C++ that the name of the output (the program) is hello
`hello.ccp`	The source file to compile

Compiling the program using the Free Software Foundation's *g*++ compiler.

Use the command:

```
% g++ -g -Wall -ohello hello.cpp
```

Details

`g++` The command to invoke the compiler

`-g` Turns on debugging

`-Wall` Turns on all warnings

`-ohello` Tell the compile that the name of the output file (the program) is hello.

`hello.cpp` The source file containing the program.

Compiling the program using Borland-C++ under Microsoft Windows

Use the following command to compile the program:

```
C:> bcc32  -v -N -w -tWC -ehello hello.cpp
```

Details

`bcc32` **The name of the compiler**

`-v` **Turns on debugging**

`-N` **Turns on stack checking**

`-tWC` **Indicates that this program is a console application (runs in a command prompt window).**

`-ehello` **Tells the compiler that the program name (executable) is hello.**

`hello.cpp`

The source file containing the program

Compiling the program using Microsoft Visual C++ .NET

C:> **`cl /FeHELLO /GZ /RTCsuc /Zi /Wall hello.cpp`**

Details

`cl`	The name of the compiler.
`/FeHELLO`	The executable is named HELLO.EXE
`/GZ /RTCsuc`	Turn on runtime checking
`/Zi`	Turns on debugging
`/Wall`	Turns on all warnings
`hello.cpp`	The name of the source file

Running the program

On UNIX:

```
% hello
```

and the message

```
Hello World
```

appears.

On MS-DOS/Windows:

```
C:> hello
```

and the message

```
Hello World
```

appears.

Getting Help

On UNIX

```
% man subject
```

and

```
% man -k keyword
```

On **MS-DOS/Windows**, use the Help menu item in the integrated development environment.

Programming Exercises

Exercise 2- 1: On your computer, type in the "hello" program and execute it.

Exercise 2-2: Take several programming examples from any source, enter them into the computer and run them.