

System Programming

Second Class

مدرس مساعد: منال مطلوب

References:

- 1) "System software" I.A.Dhotre, A.A. Puntambekar (first edition 2008).
- 2) "The 8086 book" Russell Rector, George Alexy (1980).
- 3) "Operating Systems" I.A.Dhotre (seventh revised edition 2009).
- 4) Robert M.Graham, "Principles of systems programming". 1975.

CHAPTER ONE (General Introduction)

1.1 What is System programming?

A computer system is a set of tools that can people in performing data processing tasks and in solving computational problems. Some of these tools consist of mechanical hardware and electronic circuits. These are the computer itself and its peripheral equipment such as terminal. Printers, disk storage devices, and communications equipment. Other tools are software the systems programs.

System programs have two purposes:

- 1) They make a computer system easy to use.
- 2) They make it possible for the resources of the system to be used efficiently.

System programs are software that is normally included with a computer system when it is purchased. These programs are typically provided by the manufacturer or by specialized vendors. Some examples are:

Input/output subroutine packages.

Monitors.

Operating system.

Assemblers.

Microprocessors.

Interpreters.

Compilers.

Linking Loaders.

Editors.

Debuggers.

Data Base Managers.

Communications Software.

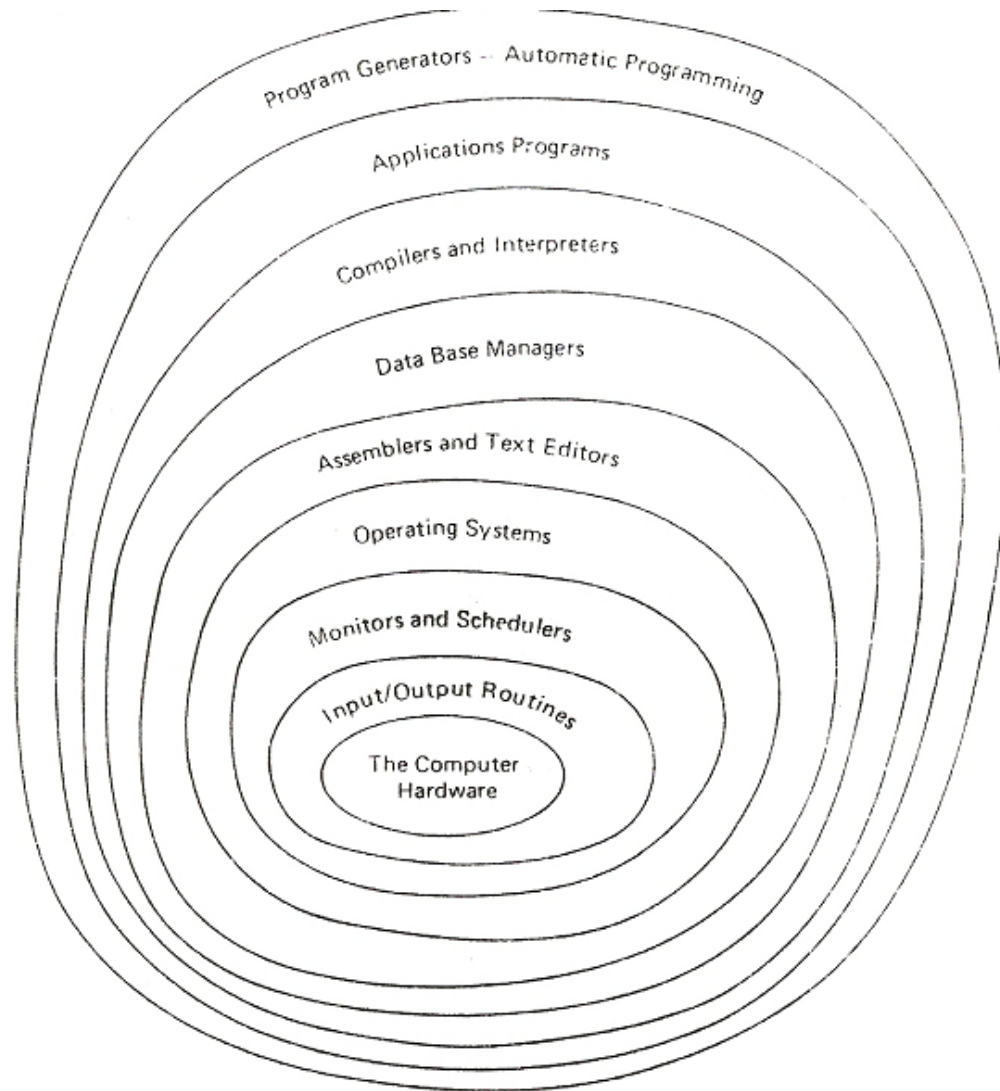


Figure 1-1: Hierarchies of Systems Software. The user is separated from the machine by many intervening layers of systems software. Lucky for him!

Assembler - used to translate instructions from a human understandable format to a machine understandable format.

Linker - used to combine one or more object modules to create one binary executable file.

Loader - used to load executable machine code into memory and begin the execution at a specified starting address.

Typical Computer Structure

Main components:

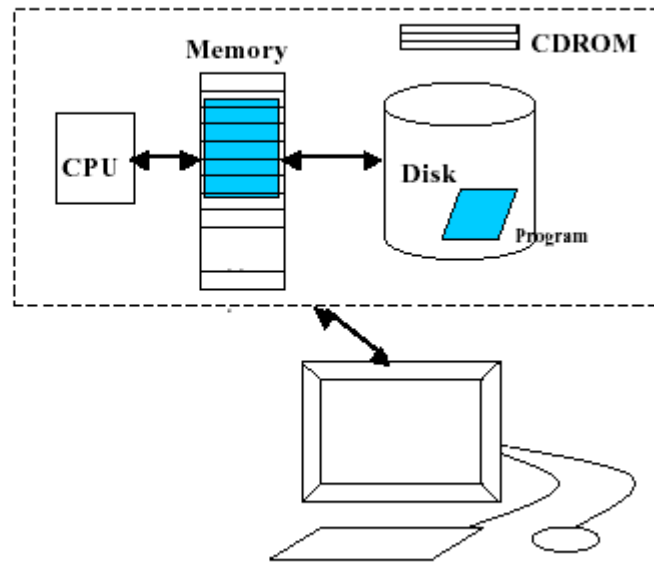
- CPU
- Main Memory
- Secondary: disk
- IO devices:

Input:

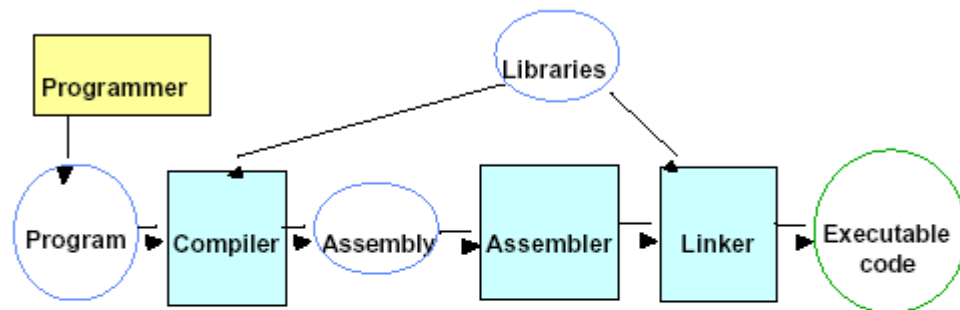
- Keyboard
- Mouse
- CDROM

Output:

- Display
- Printer



“Building” a Program



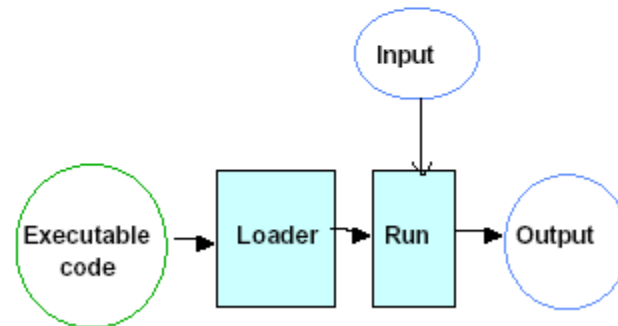
`z = x + y;`

```
load  r1, x
load  r2, y
add   r3, r1, r2
store r3, z
```

```
1100 0110
1010 1111
0101 1000
1010 1111
0101 1000
0000 1001
1100 0110
0000 1001
```

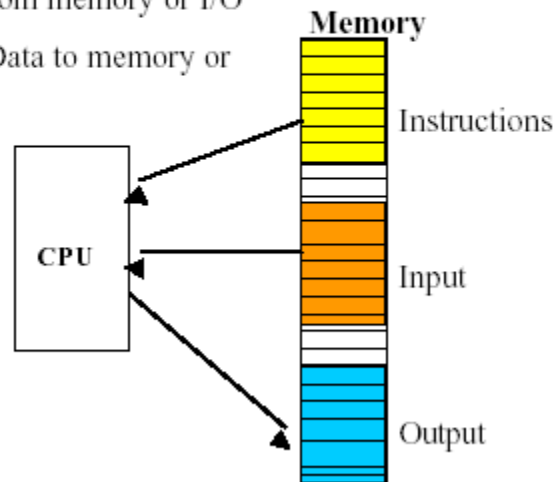
Running the Program

- ❑ *Loader* puts the program into the computer memory
- ❑ Running the program is done by an Operating System command
- ❑ Input are read from the I/O devices and from Memory
- ❑ Output is written to I/O devices and to Memory

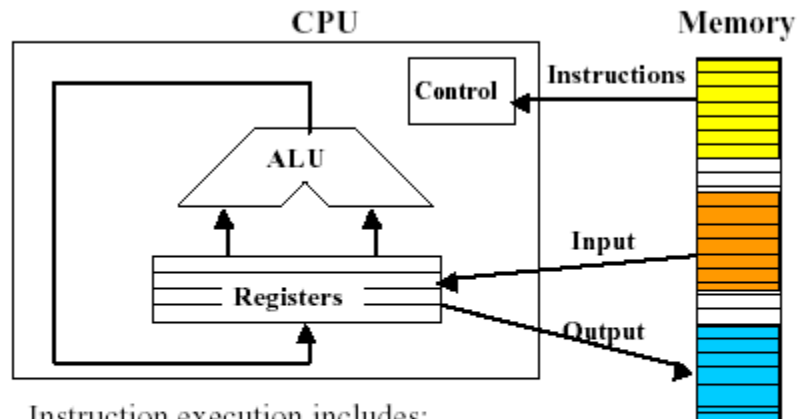


Execution of a Program

- CPU executes the Instructions
- CPU reads Input Data from memory or I/O
- CPU stores the Output Data to memory or to I/O



Execution of the Instructions



Instruction execution includes:

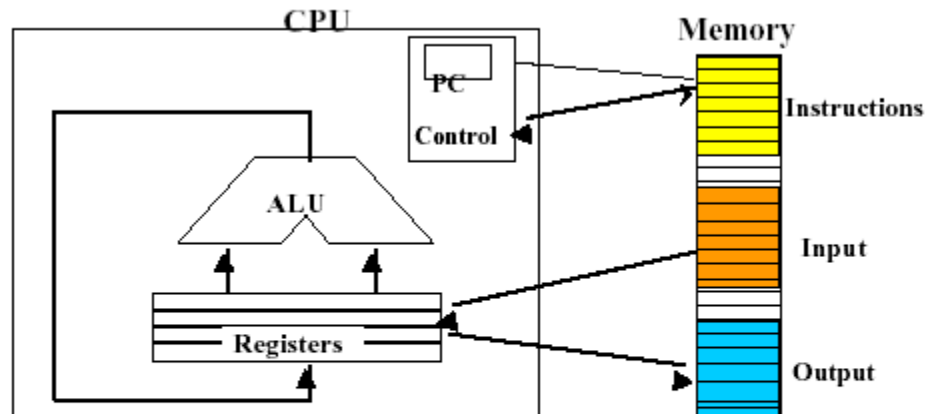
1. Load instruction from memory
2. Decode instruction
3. Load data from memory/registers
4. Execute the operation
5. Store result in register/memory

$z = x + y;$

```

load  r1, x
load  r2, y
add   r3, r1, r2
store r3, z
    
```

Loading Instruction



To load an **instruction**:

- Need an “instruction address”
- Pointed by *PC* (Program Counter)

Next instruction is usually in $PC + 4$, except for jumps