Chapter 1
Introduction to Operating Systems

1.1 General Definition

An OS is a program which acts as an *interface* between computer system users and the computer hardware. It provides a user-friendly environment in which a user may easily develop and execute programs. Otherwise, hardware knowledge would be mandatory for computer programming. So, it can be said that an OS hides the complexity of hardware from uninterested users.

In general, a computer system has some resources which may be utilized to solve a problem. They are

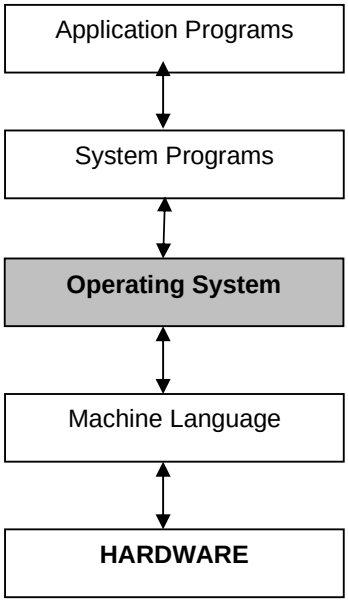
- Memory
- Processor(s)
- I/O
- File System

  etc.

The OS manages these resources and allocates them to specific programs and users. With the management of the OS, a programmer is rid of difficult hardware considerations. An OS provides services for

- Processor Management
- Memory Management
- File Management
- Device Management
- Concurrency Control

Another aspect for the usage of OS is that; it is used as a *predefined library* for hardware-software interaction and this is why, system programs apply to the installed OS since they cannot reach hardware directly.



Since we have an already written library, namely the OS, to add two numbers we simply write the following line to our program:

c = a + b ;

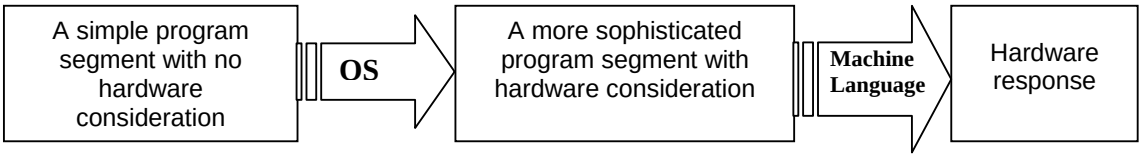
whereas in a system where there is no OS installed, we should consider some hardware work as:

(Assuming an MC 6800 computer hardware)

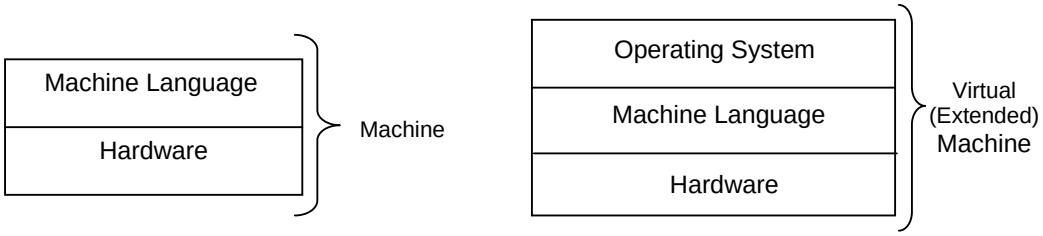
LDAA $80 → Loading the number at memory location 80
LDAB $81 → Loading the number at memory location 81
ADDB → Adding these two numbers
STAA $55 → Storing the sum to memory location 55

As seen, we considered memory locations and used our hardware knowledge of the system.

In an OS installed machine, since we have an intermediate layer, our programs obtain *some advantage of mobility* by not dealing with hardware. For example, the above program segment would not work for an 8086 machine, where as the "c = a + b ;" syntax will be suitable for both.



With the advantage of easier programming provided by the OS, the hardware, its machine language and the OS constitutes a new combination called as a **virtual (extended) machine**.



In a more simplistic approach, in fact, OS itself is a program. But it has a priority which application programs don't have. OS uses the **kernel mode** of the microprocessor, whereas other programs use the **user mode**. The difference between two is that; all hardware instructions are valid in kernel mode, where some of them cannot be used in the user mode.

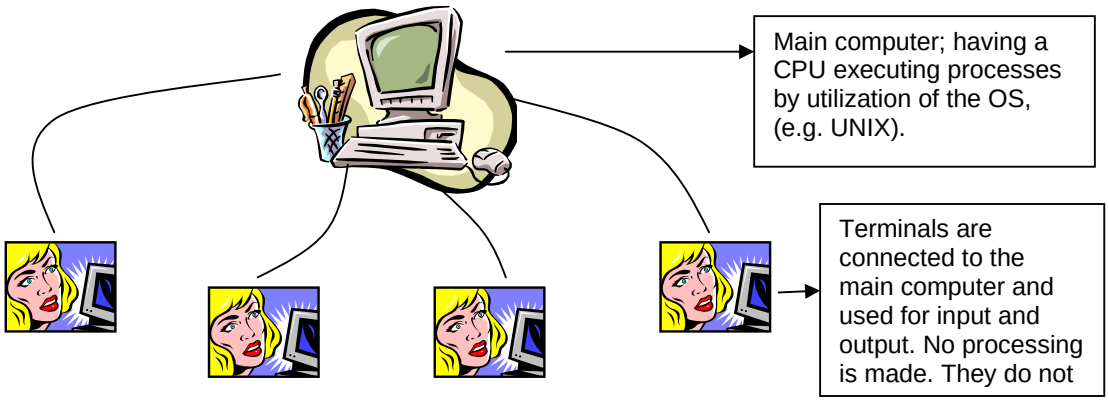
1.2 History of Operating Systems

It all started with computer hardware in about 1945s. Computers were using vacuum tube technology. Programs were loaded into memory manually using switches, punched cards, or paper tapes.

As time went on, card readers, printers, and magnetic tape units were developed as additional hardware elements. Assemblers, loaders and simple utility libraries were developed as software tools. Later, off-line spooling and channel program methods were developed sequentially.

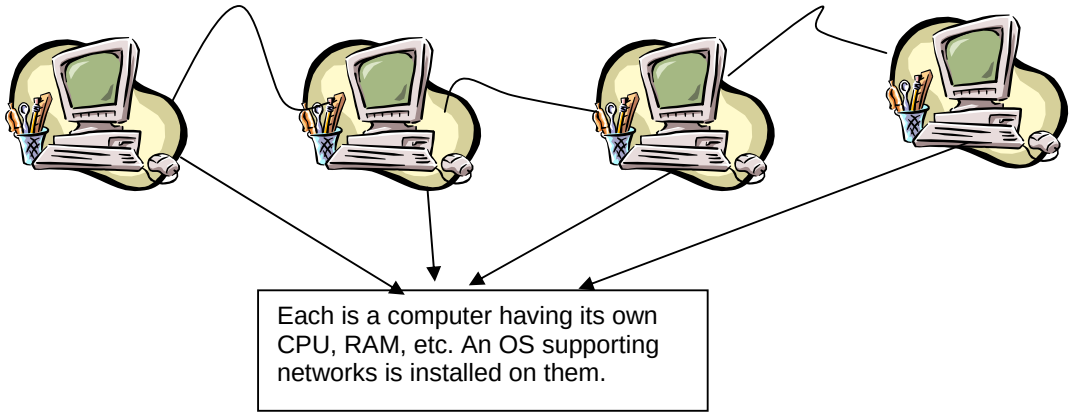
Finally, the idea of **multiprogramming** came. Multiprogramming means sharing of resources between more than one processes. Now the CPU time was not wasted. Because, while one process moves on some I/O work, the OS picks another process to execute till the current one passes to I/O operation.

With the development of interactive computation in 1970s, **time-sharing systems** emerged. In these systems, multiple users have *terminals* (not computers) connected to a *main computer* and execute her task in the main computer.



Another computer system is the **multiprocessor system** having multiple processors sharing memory and peripheral devices. With this configuration, they have greater computing power and higher reliability. Multiprocessor systems are classified into two as tightly-coupled and loosely-coupled (distributed). In the former one, each processor is assigned a specific duty but processors work in close association, possibly sharing the memory. In the latter one, each processor has its own memory and copy of the OS.

Use of the networks required OSs appropriate for them. In **network systems**, each process runs in its own machine. The OS can access to other machines. By this way, file sharing, messaging, etc. became possible.In networks, users are aware of the fact that s/he is working in a network and when information is exchanged. The user explicitly handles the transfer of information.



Distributed systems are similar to networks. However in such systems, there is no need to exchange information explicitly, it is handled by the OS itself whenever necessary.

With continuing innovations, new architectures and compatible OSs are developed. But their details are not in the scope of this text since the objective here is to give only a general view about developments in OS concept.