INTRODUCTION TO OPERATING SYSTEMS

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.

General Definition

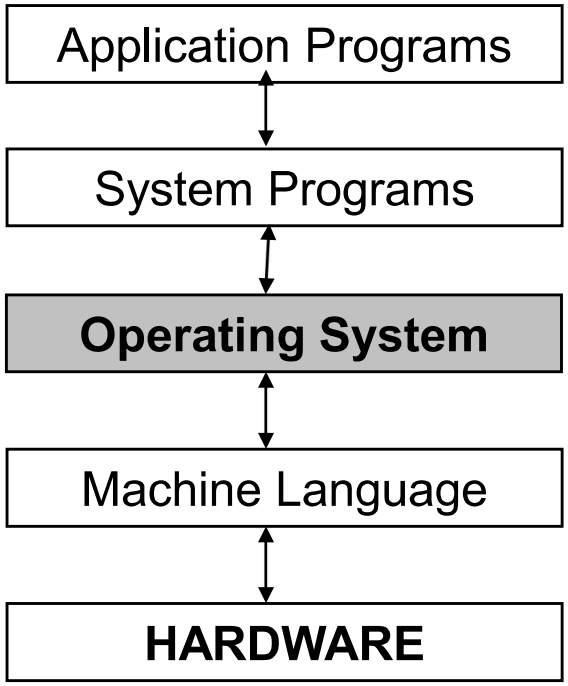
- In general, a computer system has some resources which may be utilized to solve a problem. They are
  - Memory
  - Processor(s)
  - I/O
  - File System
  - etc.

General Definition

- 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

General Definition

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



General Definition

- 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 ;
```

General Definition

- 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.

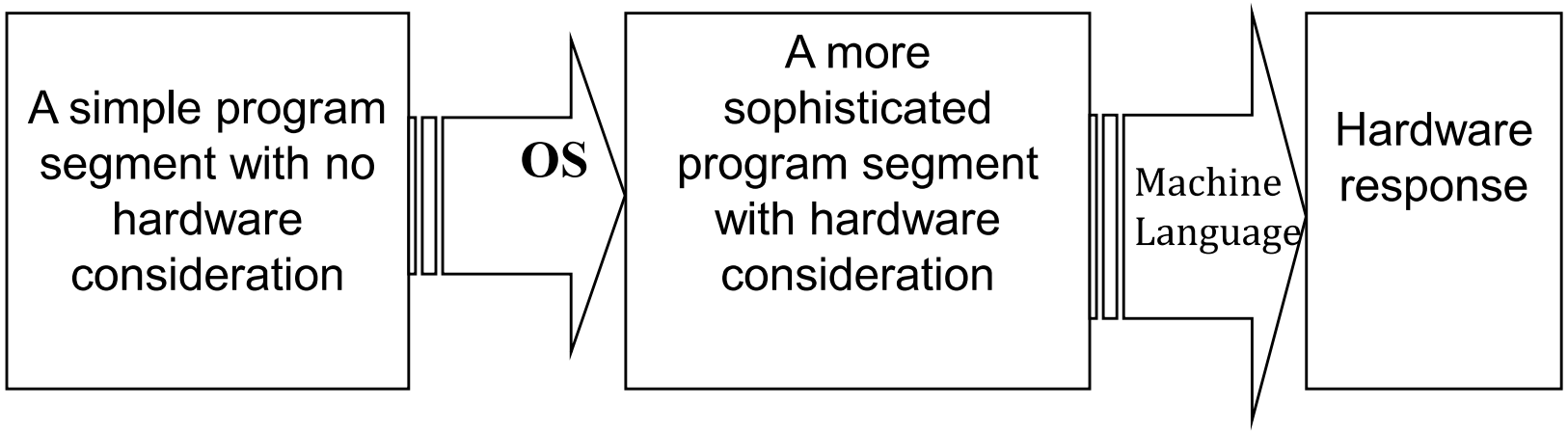
General Definition

- 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 ;"

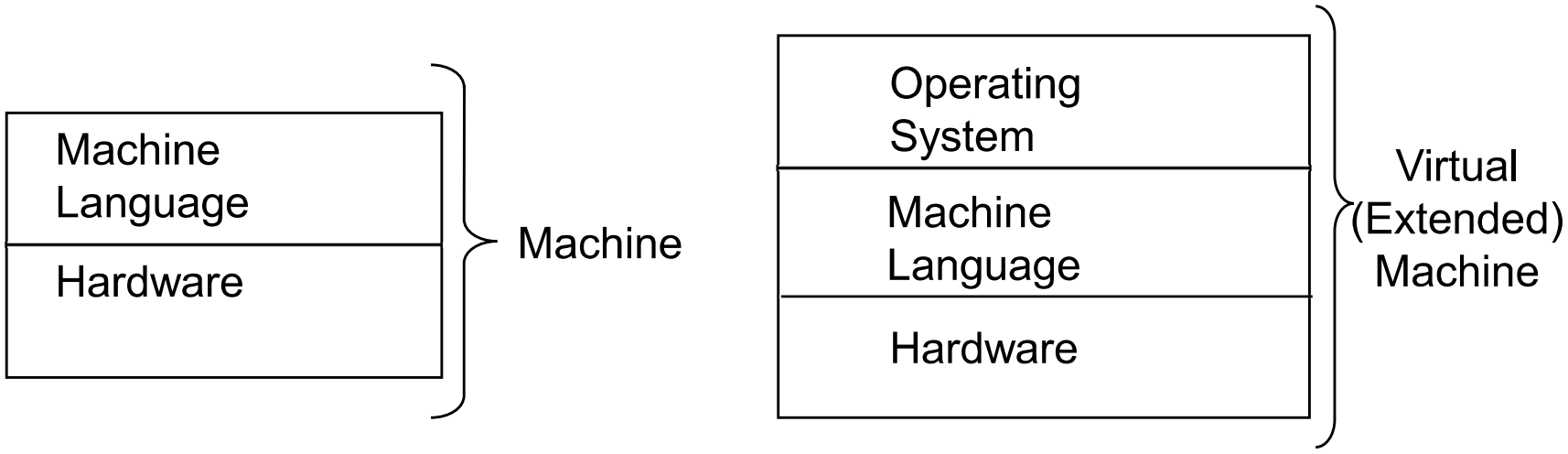
  syntax will be suitable for both.

General Definition



General Definition

- 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.**



General Definition

- 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.

History of Operating Systems

- **Single user** (no OS).
  - The only “operating system” was a person. All machine operation was “hands on”.

History of Operating Systems

- **Batch, uniprogrammed, run to completion**.
  - A computer operator takes care of the system administration.
  - The operator collects a “batch” of programs from several programmers, feeds the programs into the computer, and hands out the printed results back to the programmers.
  - The OS now must be protected from the user program so that it can start (and assisting) the next program in the batch.

History of Operating Systems

- Finally, the idea of **multiprogramming** came.
- Multiprogramming means sharing of resources between more than one processes.
- By multiprogramming the CPU time is 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.

History of Operating Systems

- **IBM OS/MFT (Multiprogramming with a Fixed number of Tasks)**

  - The (real) memory is partitioned and a batch is assigned to a fixed partition.

  - The memory assigned to a partition does not change

IBM OS/MVT (Multiprogramming with a Variable number of Tasks)

- Each job gets just the amount of memory it needs. That is, the partitioning of memory changes as jobs enter and leave

- MVT is a more ``efficient'' user of resources but is more difficult.

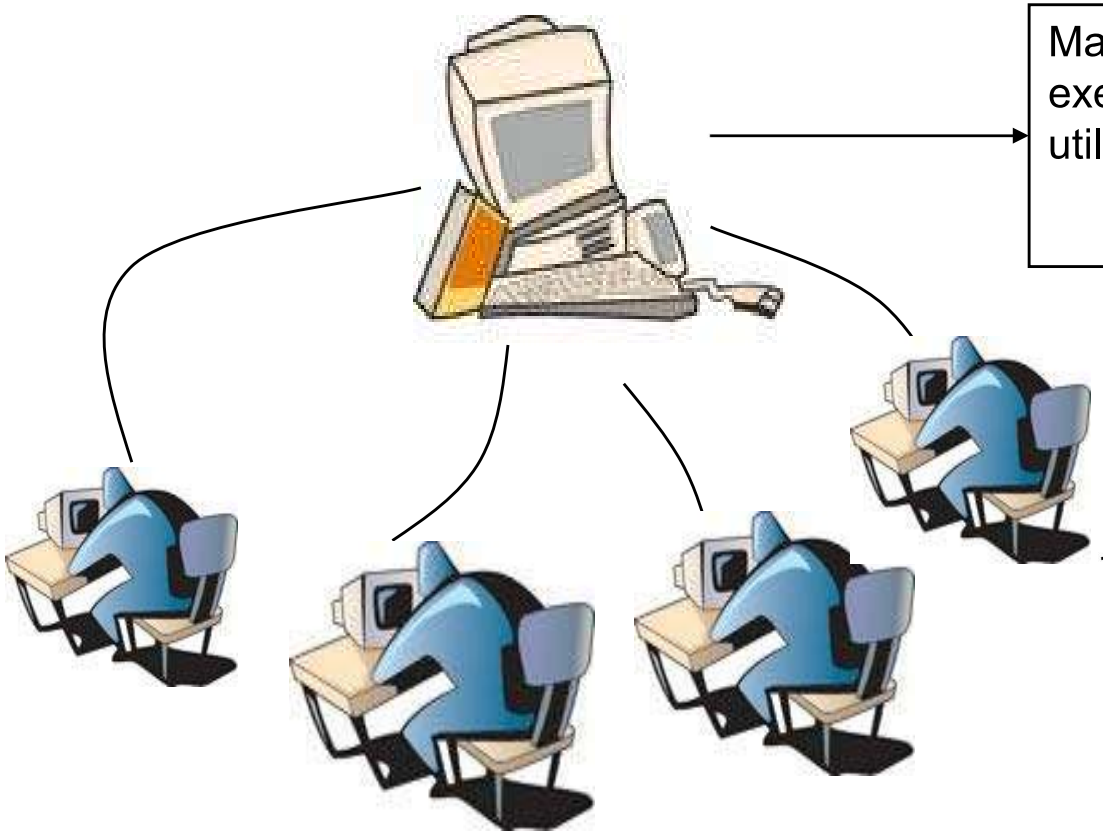
History of Operating Systems

- With the development of interactive computation in 1970s, **time-sharing systems** emerged.

- This is multiprogramming with rapid switching between jobs (processes). Deciding when to switch and which process to switch to is called **scheduling**.

- In these systems, multiple users have *terminals* (not computers) connected to a *main computer* and execute her task in the main computer.

History of Operating Systems



Main computer; having a CPU executing processes by utilization of the OS, (e.g. UNIX).

Terminals are connected to the main computer and used for input and output. No processing is made. They do not have CPUs.

History of Operating Systems

- 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.

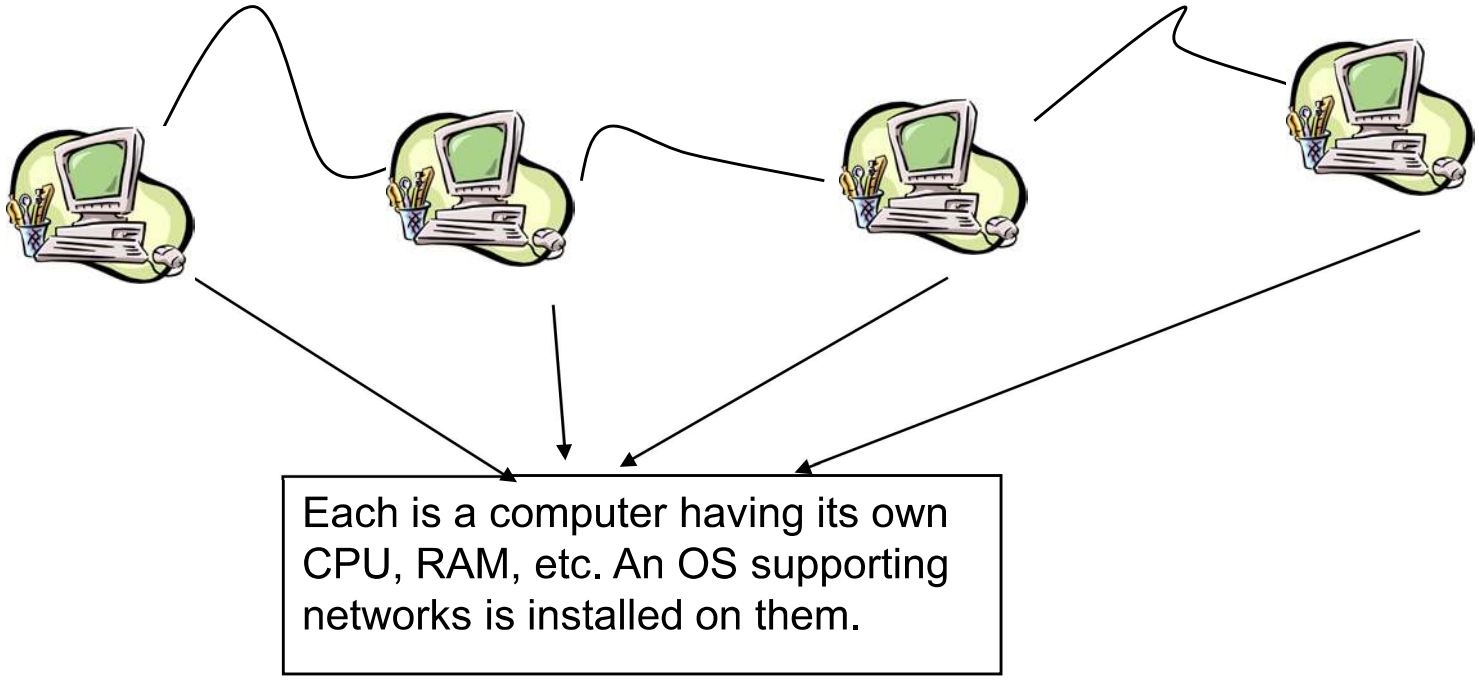
History of Operating Systems

- Multiprocessor systems are classified into two as tightly-coupled and loosely-coupled (distributed).
- In the tightly-coupled one, each processor is assigned a specific duty but processors work in close association, possibly sharing the same memory.
- In the loosely coupled one, each processor has its own memory and copy of the OS.

History of Operating Systems

- Use of the networks required OSs appropriate for them.
- In **network systems,** each process runs in its own machine but the OS have 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.

History of Operating Systems



History of Operating Systems

- **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.

- **Real time systems:** Often used in embedded systems.

  Soft vs hard real time. In the latter missin a deadline is a fatal error.