CHAPTER 1

COMPUTER SYSTEM ARCHITECTURE

Dr. Soha S. Zaghloul

LAYOUT

- Types of Processors
  - General-Purpose Processors
  - Special-Purpose Processors
- Computer Paradigms
  - Single Processor Systems
  - Multiprocessor Systems
    - Shared Memory Processors (SMP)
    - Message Passing Processors (MPP)
    - Clusters (Hybrid)
- Multiprocessors Organization
  - Separate Kernel Systems
  - Master-Slave Systems

Types of Processors

Processors may be broadly categorized into:

- General-purpose processors
- Special-purpose processors

General-purpose Processors

A general-purpose CPU is capable of executing a general-purpose instructions set, including instructions from user processes.

General-purpose instructions set are specified by the processor designer.

General-purpose instructions set perform basic data manipulation that programmers commonly use to write applications and system software.

General-purpose instructions set use memory addresses and the registers provided by the processor, such as Intel for example.

General-purpose instructions set may be classified as follows:

- Data transfer instructions: move data between memory addresses and registers.
- Binary arithmetic operations: perform arithmetic operations between binary numbers. These include add, subtract, etc…
- Decimal arithmetic operations: perform arithmetic operations on decimal operands. These include add, subtract, etc…
- Logical instructions: perform logical operations on operands. These include and, or, xor, etc…
- Control transfer instructions: such as branch and jump. Such instructions alter the normal sequence of program execution.
- Other instructions: such as shift, rotate, and string operations.

Special-purpose Processors

Sometimes, only a small subset of the general-purpose instruction is needed. In this case, it is less expensive to design a special-purpose processor.

Special-purpose processors DO NOT run users' processes.

Examples of a special-purpose processor include:

- Digital Signal Processor (DSP): specially used for signal processing.
- Graphical Processing Unit (GPU): designed for real-time 3D image rendering.
- Device-specific processors (device controllers): designed to control devices. Examples include:
  - **Keyboard microprocessor:** converts the keystrokes into binary codes to be sent and manipulated by the CPU.
  - **Disk-controller microprocessors:** receives a sequence of requests from the CPU; and implements its own disk queue and scheduling algorithm.

The OS cannot communicate with special-purpose processors provided with the main CPU. Instead, they do their jobs autonomously.

Such hierarchical arrangement relieves the OS of detailed device-specific tasks.

The OS only learns the statuses of the devices through their controllers.

COMPUTER PARADIGMS

Computer systems may be built as:

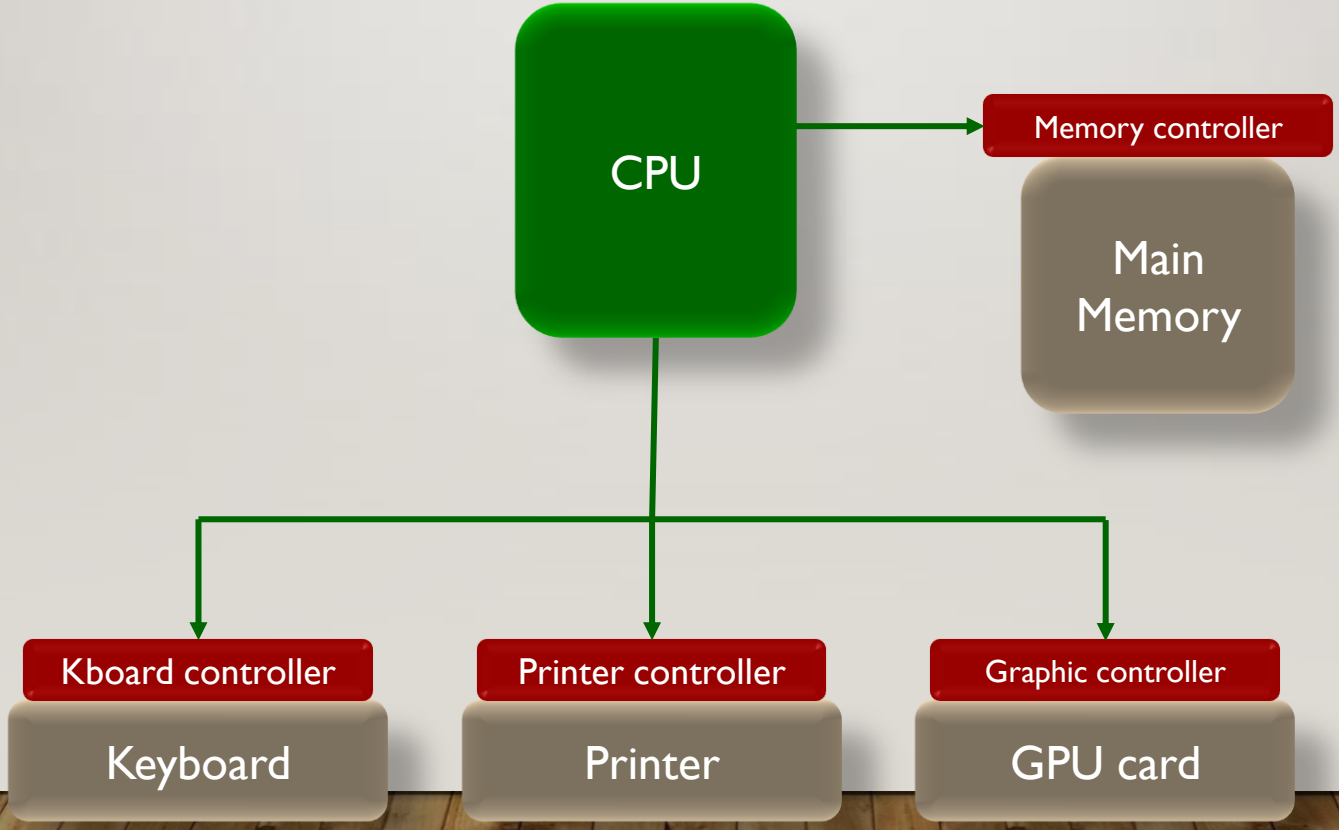
- Single-processor systems
- Multiprocessor systems

SINGLE-PROCESSORS

A single-processor system is provided with only one general-purpose CPU.

One or more special-purpose processors may be provided in a single-processor system.

The following figure represents a potential architecture of a single-processor computer system:



MULTIPROCESSORS (I) – INTRODUCTION

Multiprocessor systems have two or more general-purpose CPUs.

These processors work in close communication with each other.

Multiprocessor systems have three main advantages:

- Increased throughput: more instructions are executed in less time.
- Economy of scale: if several programs work on the same set of data, it is cheaper to store these data on one disk, and share them among all processors.
- Increased reliability: the failure of one processor does not halt the whole system. However, it may slow the execution down. This is known as fault tolerance.

Note that if we have two processors, the execution time of a program is NOT half its running time on a single processor. This is because multiprocessors incur some overhead arisen from:

- Synchronization: the OS should ensure that all parts are working together correctly.
- Resource conflict: the system components contend on the available resources. Resolving such conflict consumes additional time.

Multiprocessor systems are provided with fault tolerant algorithms to detect a failure, diagnose it, correct it, or overcome it.

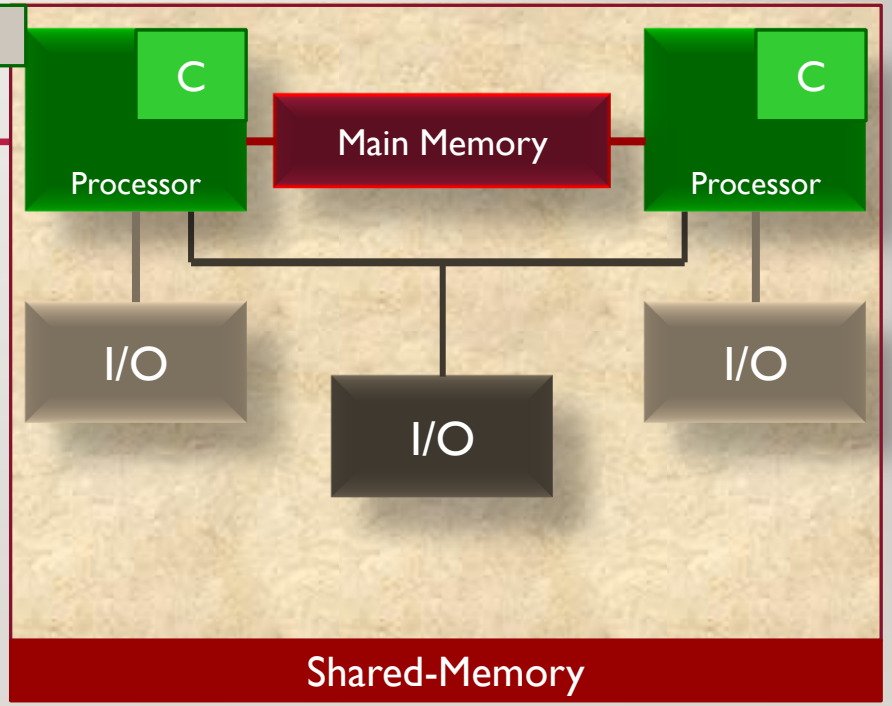
MULTIPROCESSORS (2) – PARADIGMS

Multiprocessor systems are found in two main different architectures:

- Shared Multi-Processors (SMP)
- Message Passing Processors (MPP)
- Clusters (Hybrid)

MULTIPROCESSORS (3) – SHARED-MEMORY PROCESSORS (SMP)

The following figure illustrates the SMP architecture

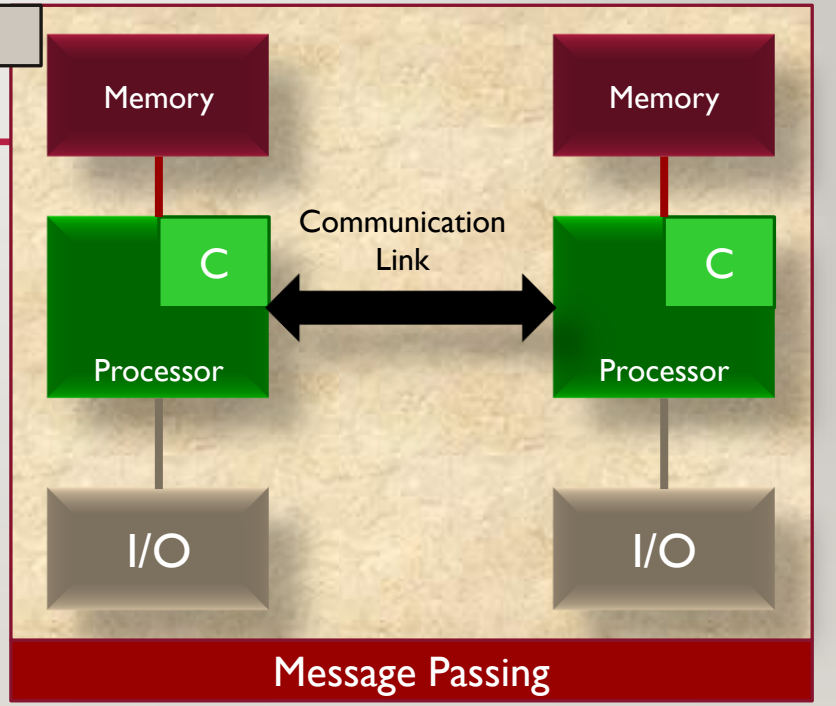


SMP has the following characteristics:

- Each processor has its own local cache.
- All processors share the same main memory.
- I/O devices may be shared or independent.
- The OS is centralized: all processors work under the same OS. So, they are known to be tightly coupled.
- They are also known to be symmetric since all processors have the same architecture.
- Communication between processors takes place through the shared memory.
- When a processor *Pi* needs a variable:
  - *Pi* searches for the variable in its own cache.
  - If not found, *Pi* fetches the variable from the main memory.

MULTIPROCESSORS (4) – MESSAGE PASSING PROCESSORS (MPP)

The following figure illustrates the MPP architecture



MPPs have the following characteristics:

- Each processor has its own local cache.
- Each processor has its own local memory.
- Each processor has its own I/O devices.
- The OS is distributed: processors may have different operating systems. So, they are known to be loosely coupled.
- They are also known to be asymmetric since processors may have different architectures.
- Communication between processors takes place through the communication link.
- When a processor *Pi* needs a variable:
  - *Pi* searches for the variable in its own cache.
  - If not found, *Pi* fetches the variable from its local memory.
  - If not found, *Pi* communicates with the other processors by passing messages through the communication link searching for the variable.

MULTIPROCESSORS (5) – SMPs VS. MPPs

The following emphasizes the differences between SMPs and MPPs:

- **Access time:** MPPs are less efficient than SMPs. This is because the speed of the communication links is slower than that of the main memory.
- **Flexibility:** MPPs are more flexible than SMPs. Different computer systems, that are distinct in both hardware and OS, can be interconnected to build a multiprocessor system.
- **Scalability:** MPPs are more scalable than SMPs. SMPs are limited by:
  - The size of the shared main memory.
  - The number of cores per processor *(to be explained later in this course)*.
- **Fault tolerance:** MPPs are more fault tolerant than SMPs. Think what would happen if the OS crashes in both systems? What would happen if a processor fails?

MULTIPROCESSORS (6) – CLUSTERS (1)

Clusters consist of two or more nodes interconnected together.

A node may be a single processor or a multicore computer.

Clustered computers share storage and are closely linked via a local area network (LAN) or a faster interconnection network.

In other words, clusters are a hybrid construction of SMP together with MPP.

Clustered computers provide high availability. In other words, they are fault tolerant.

MULTIPROCESSORS (7) – CLUSTERS (2)

Clusters may be structured in one of two ways:

- Symmetric
- Asymmetric

In symmetric clustering:

- Two or more machines are running applications and monitoring each other.
- When a machine fails, the running applications are migrated to another one.

In asymmetric clustering:

- One machine is in hot-standby mode, while the others are running the applications.
- What is a hot-standby machine?
  - Does nothing but monitors the active server.
  - If the active server fails, it becomes the active server.
  - The newly assigned active server selects a new hot-standby machine.

MULTIPROCESSORS (8) – CLUSTERS (3)

The following table summarizes the differences between the two clusters' schemes:

ASYMMETRIC

- ~ Monitor: One machine (the hot-standby)
- ~ Running: The other machines
- ~ Server failure:
  1) the hot-standby machine becomes the active server.
  2) Another machine is assigned to be the hot-standby one.
- ~ Resources Utilization: Does not use all available H/W ➔ low resources utilization

SYMMETRIC

- ~ Monitor: All machines
- ~ Running: All machines
- ~ Machine failure:
  All running processes are migrated to another machine.
- ~ Resources Utilization: Uses all available H/W ➔ better resources utilization ➔ more efficient
- However, it requires more than one application to be available to run to achieve efficient resources utilization.

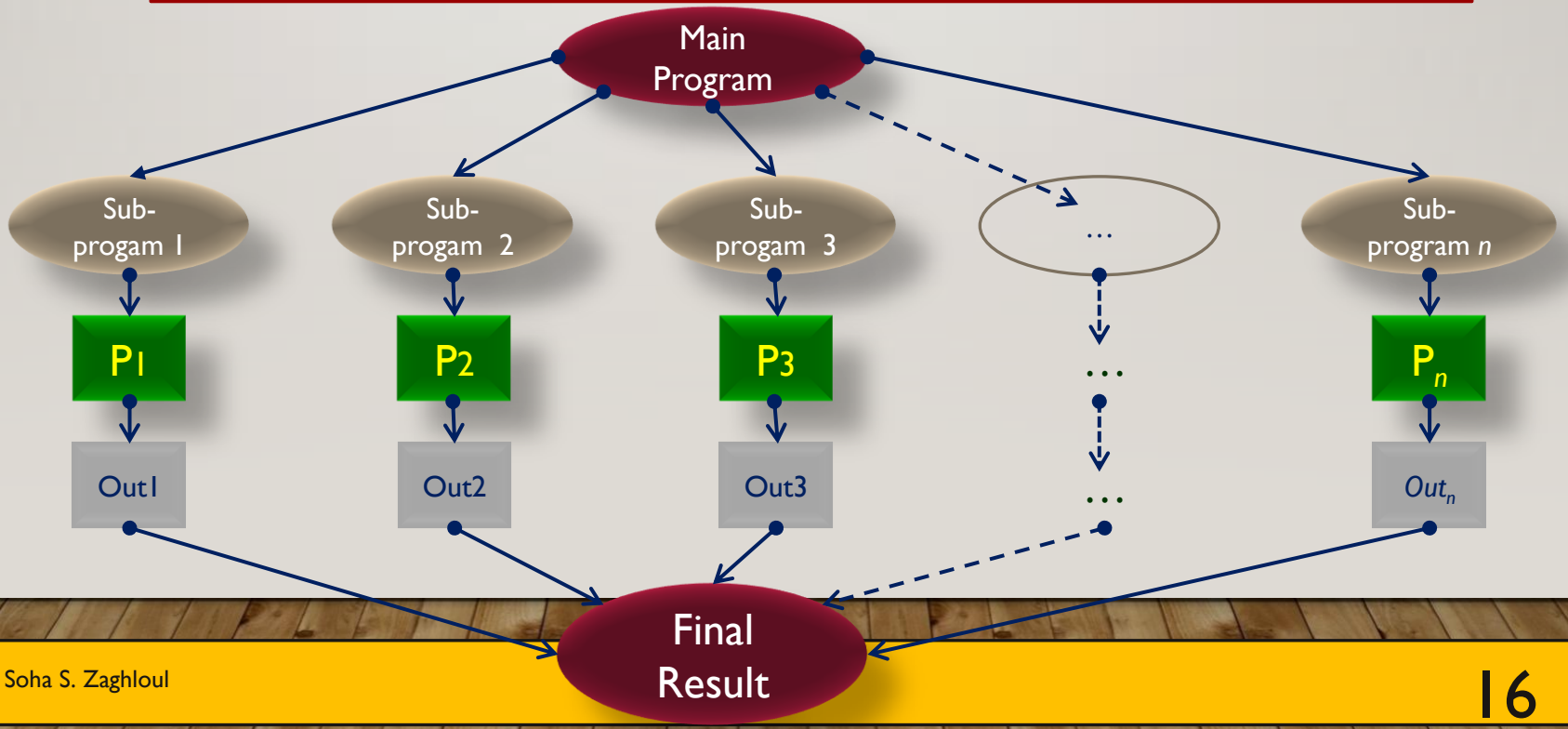
MULTIPROCESSORS (9) – CLUSTERS (4)

Clusters are used to provide High Performance Computing (HPC).

HPC systems supply significantly a greater computational power.

The programs that run on HPC must be parallelized:

- The program is divided into separate independent subprograms.
- The OS assigns each subprogram to a distinct processor.
- Results from all nodes are combined into a final solution.



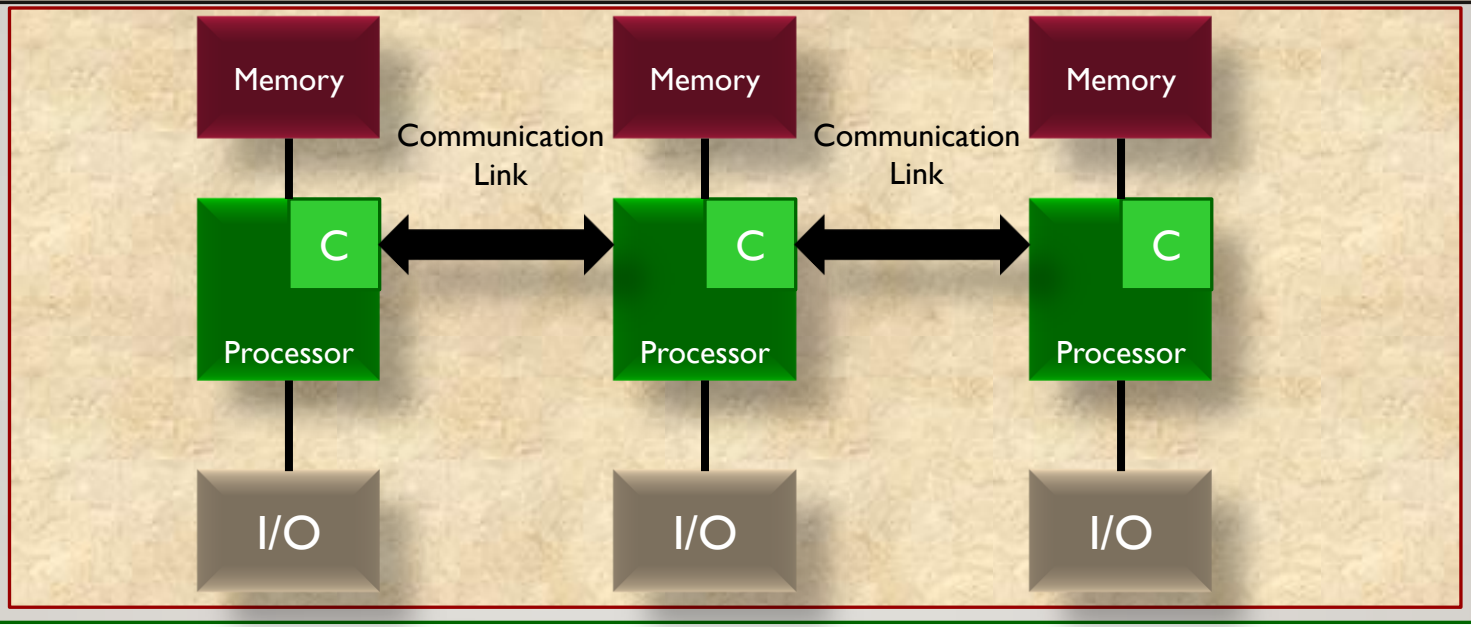
MULTIPROCESSOR ORGANIZATION (1)

On the other hand, multiprocessor systems may be also classified as:

- Separate-kernel organization
- Master-slave organization

Such organizations DO NOT apply on Shared-Memory Multiprocessors.

MULTIPROCESSOR ORGANIZATION (2) – SEPARATE-KERNEL ORGANIZATION



Each processor executes its own OS.

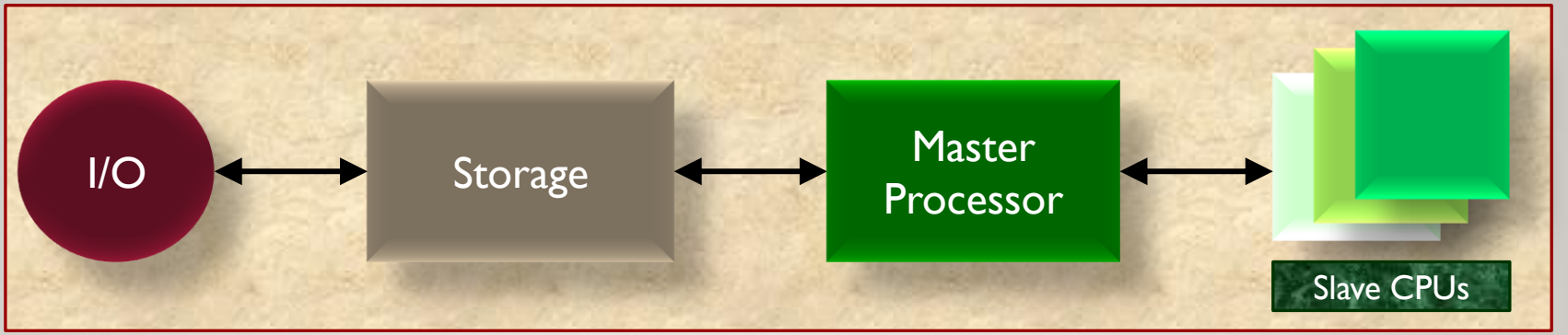
Each processor responds to interrupts from user-mode running on that processor.

This organization is fault tolerant. The failure of a processor does not entail the failure of the system. However, the processes that were executing on the failed processor should restart on another processor.

Note that in this organization, the processors do not cooperate to execute an individual process.

Therefore, some processors may remain idle while others are overloaded ➔ load unbalanced

MULTIPROCESSOR ORGANIZATION (3) – MASTER/SLAVE ORGANIZATION



One processor is designated as the master and all other processors are slaves.

Only the master executes the OS code, input/output and computations.

Slaves execute user programs only. Note that:

- (1) processor-bound jobs are performed effectively.
- (2) I/O-bound jobs running on the slave cause frequent calls to the master.

When a slave needs to perform an I/O operation:

- (1) the slave sends an interrupt to the master.
- (2) the slave waits for the master to handle the interrupt by invoking the OS

Note that the failure of the master halts the system. The failure of a slave only slows down the system.