Applied Operting Systems

Introduction

Hikmat Farhat

hfarhat@ndu.edu.lb

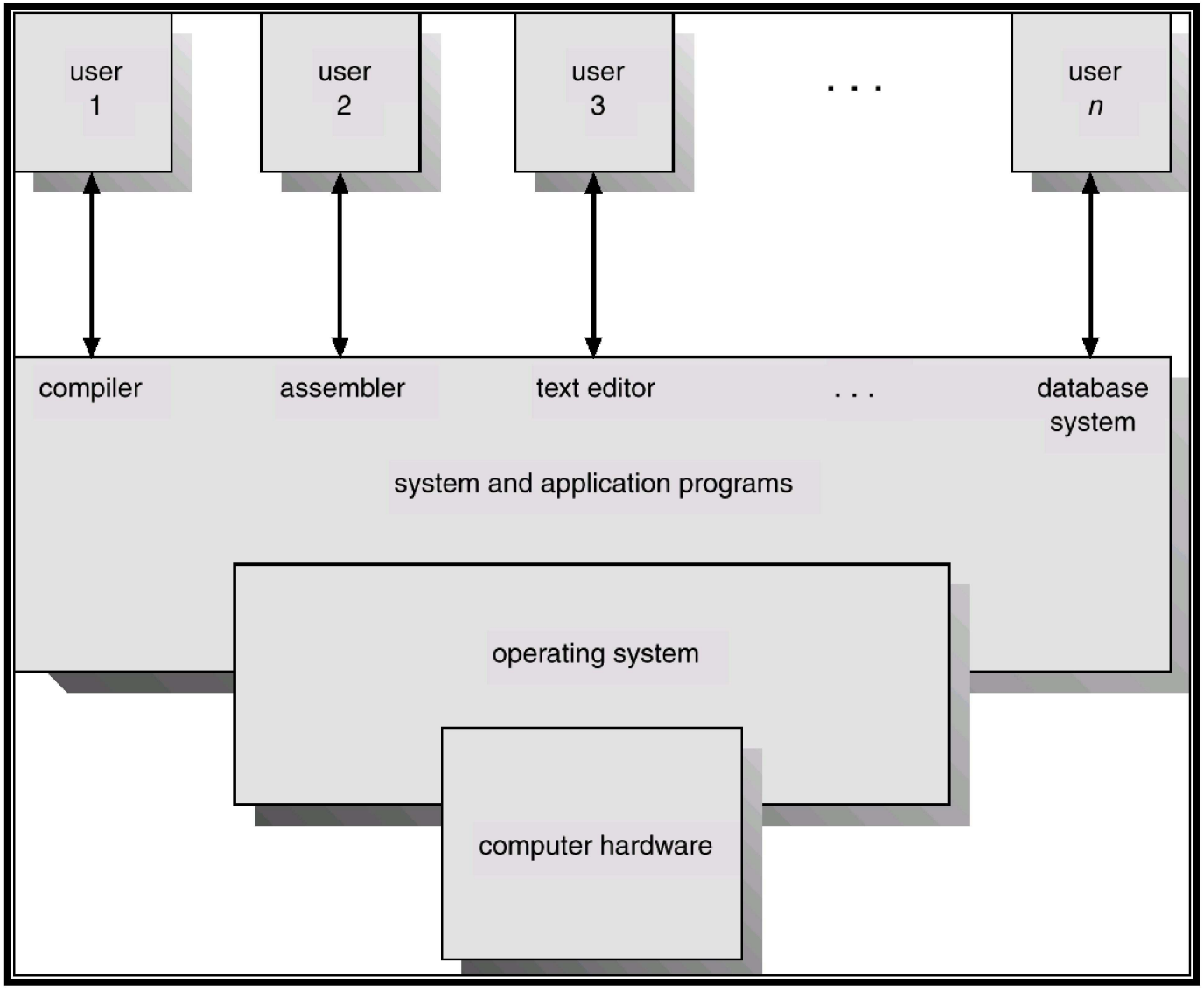
Notre Dame University

What is an Operating System

The functions of an operating system can be divided into to roles

- Presents an abstract machine for user programs.
- Resource management.

Computer System



OS as an abstract machine

- Hide the hardware detail from the user and present user programs with a much easier interface.
- Reading data from a floppy disk would require knowledge about tracks, sectors, disk rotation...
- The OS presents user programs with a simple open/read/write/close primitives.
- This concept is extended to almost all resources of a computer system.
- The OS provides a variety of services for programs through the **system call** interface.

OS as a resource manager

- Modern Computers consist of processors, memories and other devices.
- The job of the OS is to provide controlled access to these devices.
- When the computer has multiple users the need for managing and protecting the resources is even higher.
- Resource management includes multiplexing resources in two ways: in time and in space.

- When a resource is multiplexed in time, different programs take turn using it
- An example is CPU sharing, the OS allocates the CPU to some program for a period of time then to another.
- In space multiplexing each program gets a portion of the resource.
- An example is memory where each program gets a portion of the total memory of the system

Operating System Concepts

All operating systems have certain basic concepts

- Processes
- Deadlocks
- Memory Management
- Input/Output

Processes

- A key concept in all modern operating systems is the **process**.
- A process is much more than a program. It is an **executing**(active) program.
- Associated with a process is an **address space** which contains the executable (the program), its data and stack.
- It also includes a set of registers that represent the status of the process at a given instant in time.
- In addition to all the above it should include additional information to support multitasking.

- When a process is stopped temporarily it must be later restarted at exactly the same state it was in.
- Suppose that a process has a file open when it was stopped.
- When it is resumed, the file should be open and the file pointer at exactly the same place as it was.
- In almost all OS's all relevant information about a process is stored in a table, an array of structure, each entry representing the state of one process

Memory Management

- A running program usually resides in memory (Von Newman machine).
- Modern OS's allow multiple programs to reside in memory at the same time.
- The operating system needs a protection mechanism to keep programs from interfering with each other and with it.
- The protection is provided by the hardware but usually managed by the OS.

- Another memory related issue is the process address space.
- In a 32-bit computer there are 2^{32} addressable bytes. What happens if a process has an address space larger than physical memory and tries to use it all?
- In modern operating systems the technique of virtual memory is used to provide processes with memory as big as the largest address space

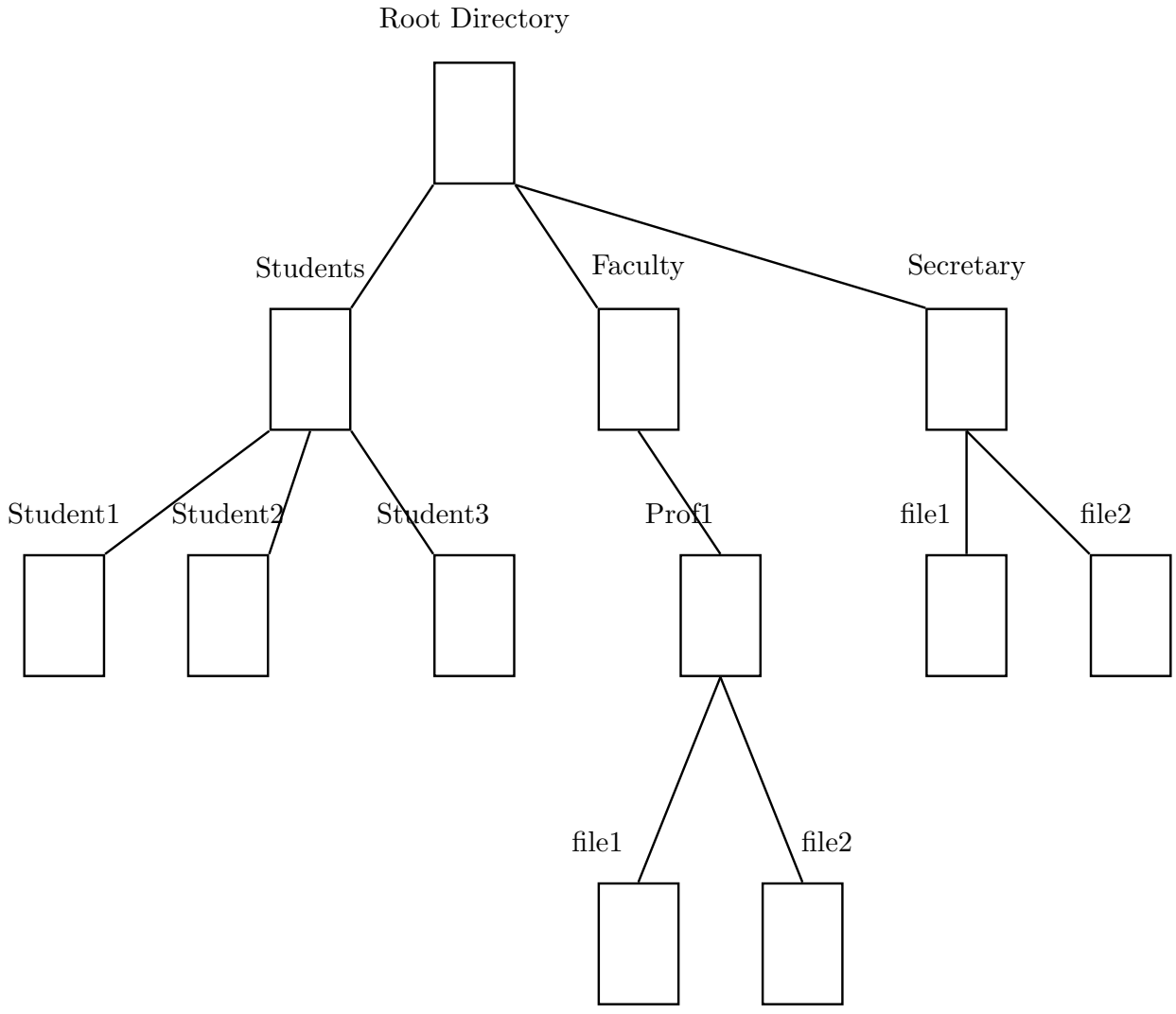
Device Management

- All computers have physical devices for input and output: keyboard, mice, network interface...
- OS manage the devices (resource management).
- OS provides a uniform and/or relatively easy interface to access all devices ().
- Usually the OS part responsible for device management is split into two parts: a device independent part and a device dependent part.

- User programs interact with the device independent part which provides a uniform abstract interface to all devices.
- The device dependent parts (device drivers) interact with the device from one side and the device independent part from the other.

Files

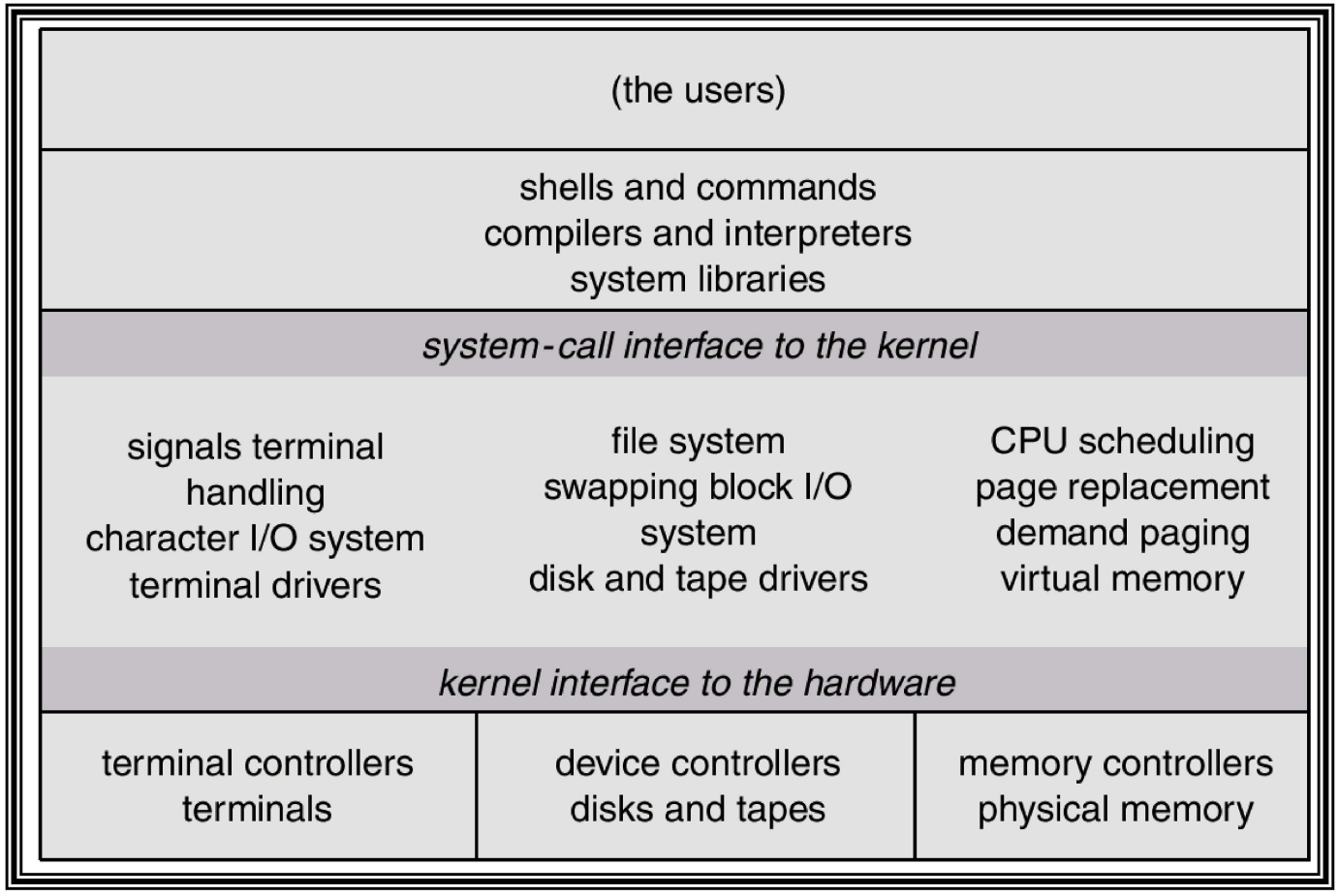
- A key concept supported by all operating systems is the file system.
- A file system provides a convenient way to store and retrieve data.
- To provide a way to organize files most operating systems have the concept of a directory.
- The easiest and most powerful way is to organize files in a hierarchy.


Root Directory
Students
Faculty
Secretary
Student1
Student2
Student3
Prof1
file1
file2
file1
file2

System Calls

- System calls provide an interface between user programs and the operating system.
- User programs request OS services through the system call interface.
- Usually library functions (C, C++) are provided as wrappers for system calls (assembly).
- A system call performs a privileged operation on behalf of the user (unprivileged) program.

UNIX Structure



Multiprogramming

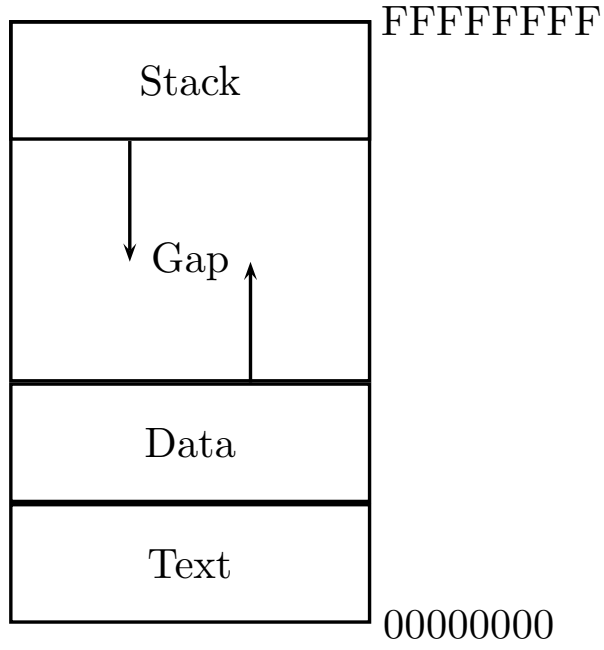
- Multiprogramming is the ability to run multiple processes at the same time.
- Note that in uniprocessor systems only one process can be executing at a given moment.
- By switching quickly between processes the "illusion" of running multiple processes can be maintained.
- Multiprogramming also requires that multiple programs are resident in memory at the same time.

Dual-Mode Operation

- OS acts as a policeman.
- No process can access the memory space allocated to others.
- User programs cannot access hardware directly nor issue privileged instructions.
- Hardware usually differentiates between at least two modes of operation: **user mode** and **monitor**(or kernel) mode.
- If the system is in monitor mode it can switch to user mode but **not** vice versa.

Process Address Space

- Each process is allocated a portion of the total memory.
- A typical process needs memory for text, data and stack



Interrupts

- An operating system is interrupt driven.
- An interrupt transfers control to the appropriate interrupt service routine (function).
- Interrupt service routines (ISR) are part of the operating system.
- Different types of interrupts exist and each has its own routine.
- An interrupt is distinguished by its number which is used as an index into a table.

How Are Interrupts Generated?

Interrupts can be generated by

- Hardware devices: when a device requires the attention of the CPU it raises the interrupt signal.
- Errors or exceptions: division by zero, invalid memory reference.
- Software: a user program can use a specific instruction to cause an interrupt. This is usually used for system calls.

Interrupts And Multiprogramming

- When a timer expires it raises an interrupt.
- The ISR for the timer is a part of the OS.
- When the timer ISR is called the system switches to monitor mode.
- Being in kernel mode the current process can be suspended and another run in its place.
- The decision which process to run after the timer interrupt is up to the **scheduler**.
- A process is selected from the **ready queue** according to some policy.

Interrupts And System Calls

- By definition a system call is a service provided by the OS to user processes, hence it runs in kernel mode.
- How is this done since a process cannot switch from user mode to monitor mode?
- The answer is the ability of user mode processes to generate software interrupts.
- Almost all processors have an **int** instruction which generates an interrupt.
- Since the processor switches automatically to kernel mode to handle interrupts this provides a convenient way to implement system calls.

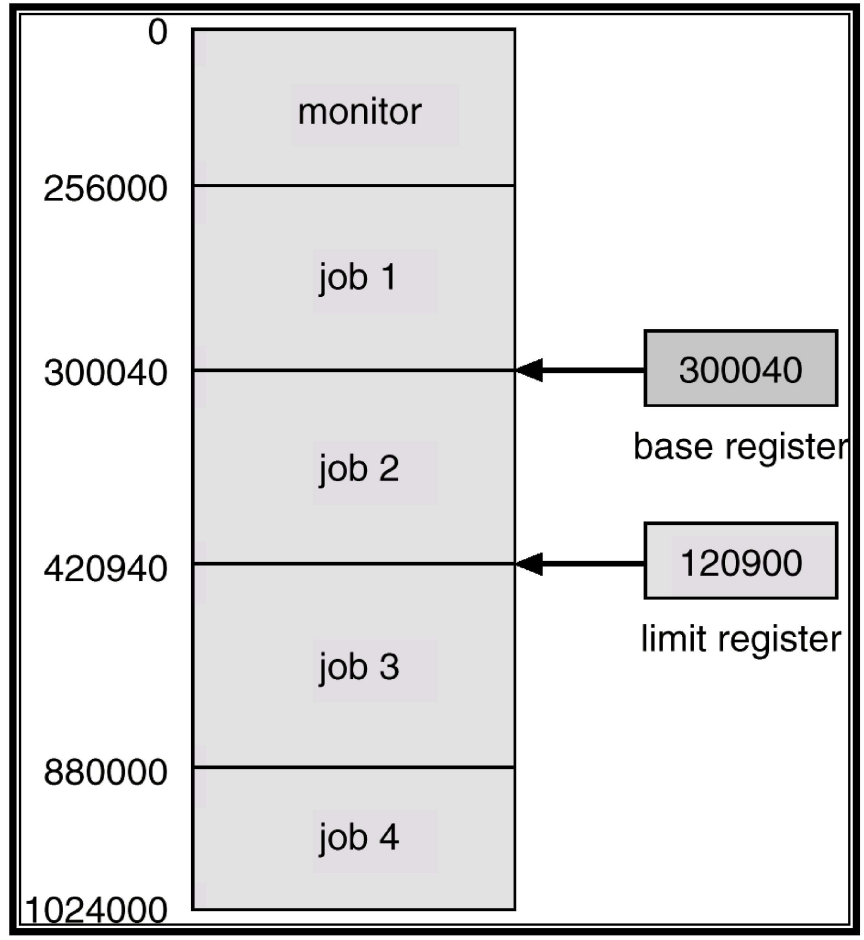
- After each instruction the CPU checks the interrupt line.
- If there is a signal on the line the CPU checks the interrupt number (more on that later).
- The current process is suspended.
- According to the interrupt number the CPU invokes the corresponding interrupt service routine (ISR).
- When the ISR is done control is transferred to another process.

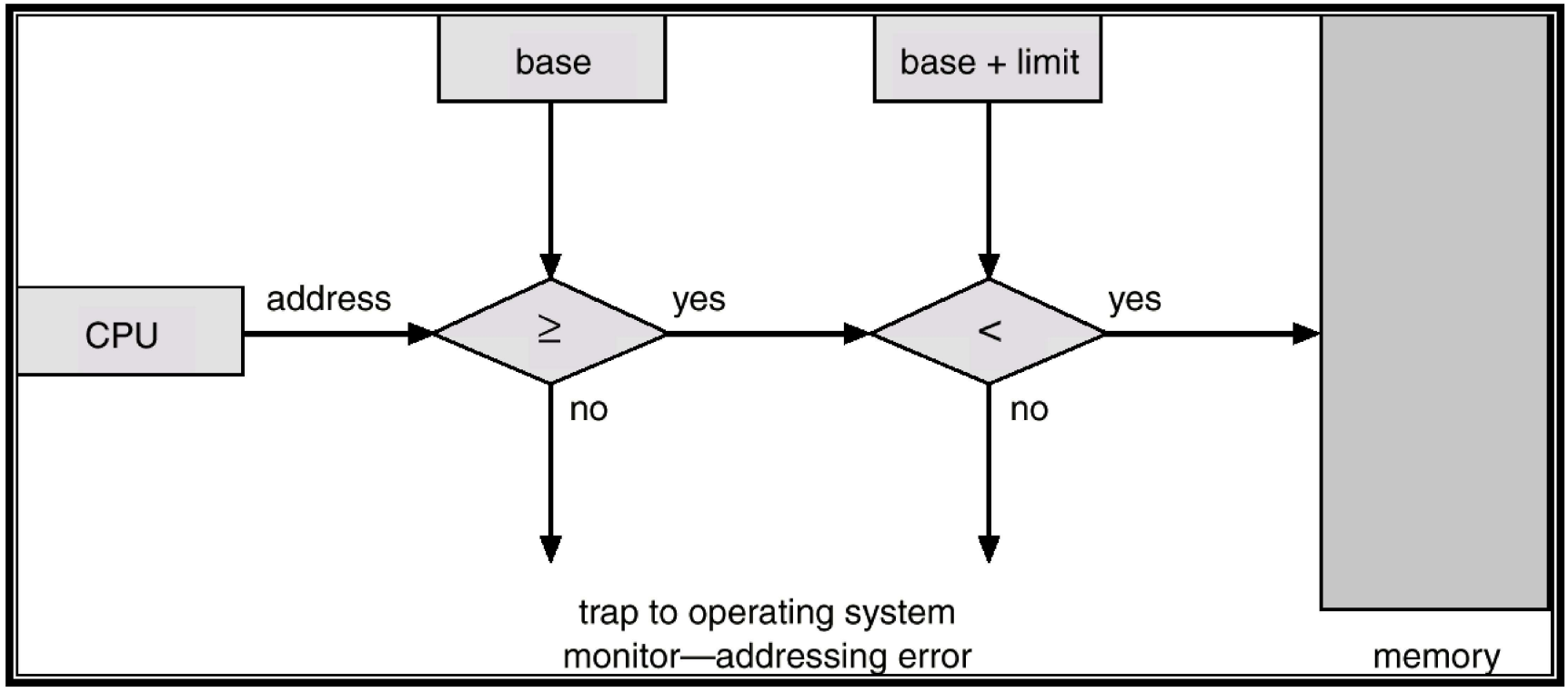
Processors

- The basic cycle of a CPU is to fetch an instruction from memory, decode it, then execute it. A program is just a series of such cycles.
- The CPU maintains several registers. Some registers are general purpose and some are for specific purpose like
  - Program counter,**PC**, which holds the address of the instruction to fetch.
  - The stack pointer, **SP**, which holds the address of the top of the stack.
  - The Program status word, which holds many control bits like condition codes and processor mode.

Memory

- Several programs can be in main memory at once.
- Two problems must be solved
  1. How to protect programs.
  2. How to handle relocation
- Simplest solution: use **base register** and **limit register**.
- This is a simple example of mapping from **virtual memory** to **physical memory**.
- This translation is done by the **Memory management unit**


0
monitor
256000
job 1
300040
300040
base register
job 2
420940
120900
limit register
job 3
880000
job 4
1024000


base
base + limit
CPU
address
≥
yes
<
yes
no
no
trap to operating system
monitor—addressing error
memory

CPU Timer

- The timer can be set by software.
- It is decremented every clock tick.
- When the timer reaches 0 it generates an interrupt.
- This way an OS can regain control and implement time sharing.
- Setting the timer is a privileged instruction, it can be executed in monitor mode only.