Operating System and Computer Organization Background

CS 502

Fall 98

Waltham Campus

Operating System Components

- Process Management
- Memory Management
- Persistent Storage Management
- File Management
- I/O System Management
- Distributed Systems and Networks
- Interface to the User
  - Shells
  - Graphical User Interface

Process Management

- A process is a program in execution. It is equivalent to a Virtual Computer. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management:
  - Process creation and deletion
  - Process suspension and resumption
  - Provision of mechanisms for
    - Process synchronization
    - Process communication

Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses is contents in the case of a system failure.
- The operating system is responsible for the following activities in connection with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which process to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Persistent Storage Management

- Since main memory is volatile and too small to accommodate all data and programs permanently, the computer system must provide persistent storage to back up main memory.
- Disks are the principle on-line storage medium.
- The operating system is responsible for the following activities in conjunction with disk management:
  - Free-space management
  - Storage allocation
  - Disk scheduling

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connection with file management:
  - File creation and deletion
  - Directory creation and deletion
  - Support of primitives for manipulating files and directories
  - Mapping files onto persistent storage
  - File backup on off-line media

I/O System Management

- The I/O system consists of:
  - A buffer caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Distributed Systems and Networks

- A distributed system is a collection of processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows”
  - Computation speed-up
  - Potential for increased data availability
  - Potential for enhanced reliability

User Interface

- A command shell or a window-based GUI provide the ability to executed application programs and convenient access to invoking systems programs. These allow for:
  - Process creation and management.
  - I/O handling, monitoring, and management
  - Secondary storage management
  - Main memory monitoring and management
  - File-system access
  - Protection
  - Networking
  - Accounting
  - Programming Language support

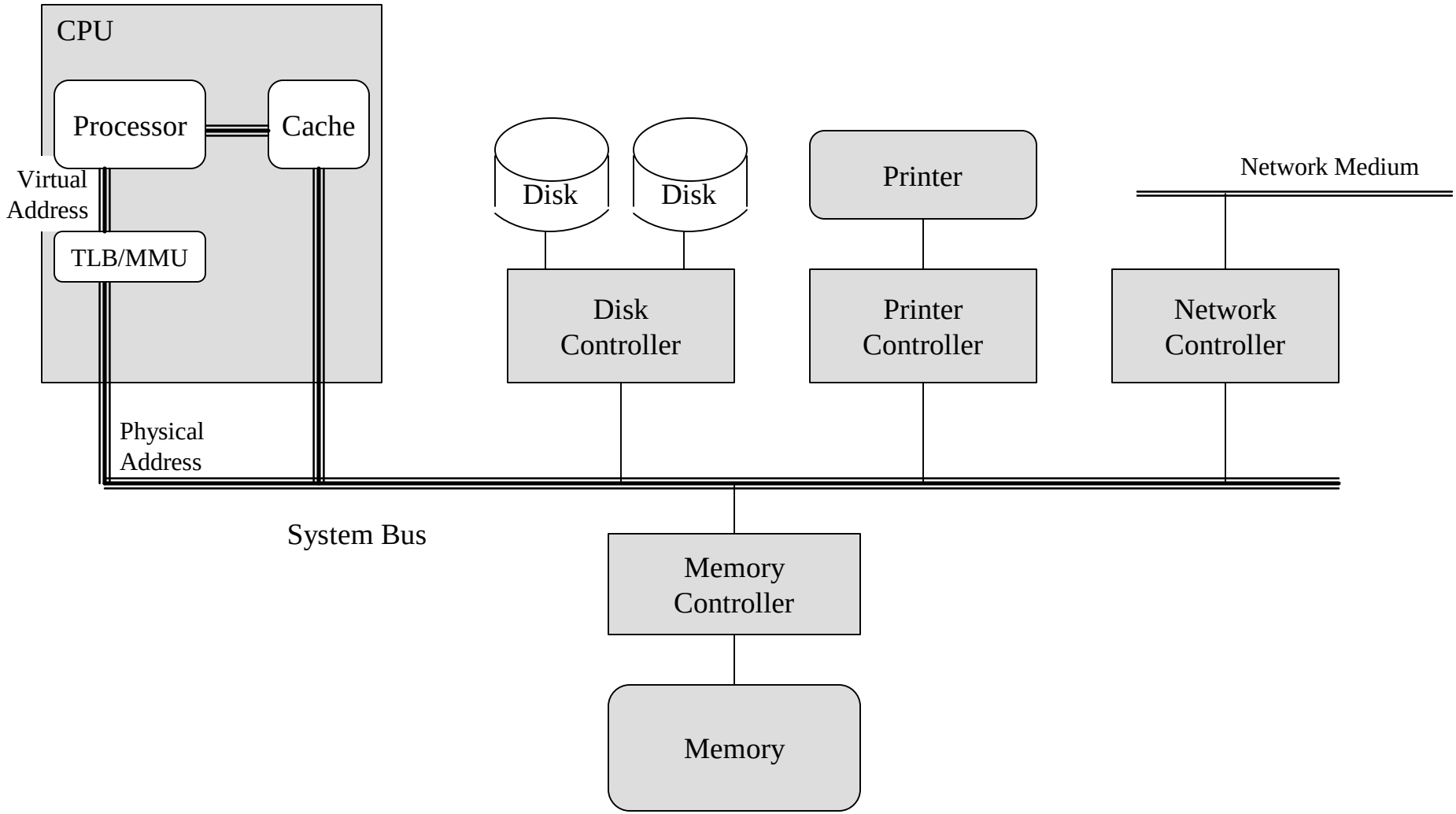
OS Components Common Requirements

- Concurrency
- Resource Management
- Protection
- Exceptions
- Asynchronous Operation and I/O
- Scheduling
- Synchronization

OS Motivated Architectural Features

OS Functionality	Hardware Support
Protection	Kernel/User Mode Protected Instructions Base and Limit Registers
Exceptions and Asynchronous Operation	Interrupt and Trap Vectors
I/O Control	Memory-Mapping or I/O Instructions DMA
Scheduling and Time-Multiplexing	Timer
Synchronization	Atomic Instructions

Computer System Architecture

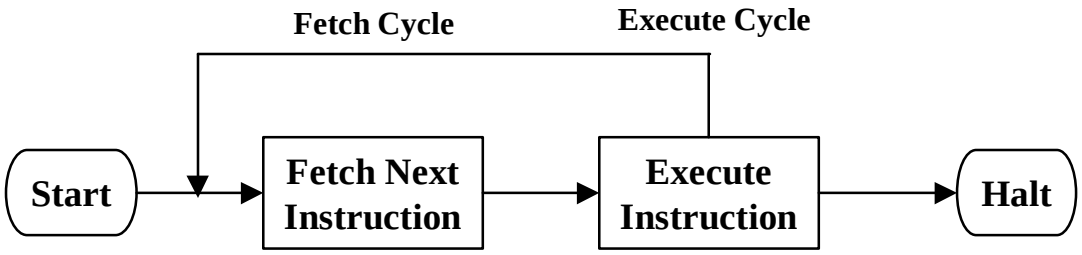


Computer System Operation

- I/O Devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from the local buffers.
- I/O is from the device to the local buffer of the controller.
- Device controller informs CPU that it has finished an operation or needs attention by causing an interrupt.

Central Processing Unit

- Processor Registers
  - User-visible registers
    - Data Registers
    - Address Registers
  - Control and Status registers
    - IA: Instruction Address
    - IR: Instruction Register
    - PSW: Program Status Word
- Instruction Execution



Processor (cont.)

- Instruction Classes
  - Non-privileged
    - Processor-memory
    - Data Processing
      - Arithmetic
      - Shift Operations
    - Flow control
      - Branch
      - Subroutine linkage
  - Privileged
    - Processor-I/O
    - Control

Architectural Features

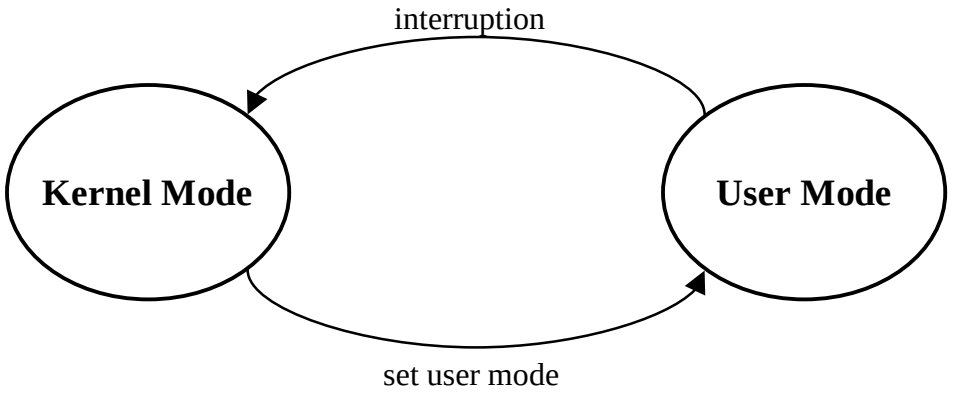
- Kernel/User Mode
- Protected Instructions
- Address Translation
- Exception and Trap Vectors
- Interrupts
- Communication with I/O Controllers
- Timer
- Atomic Instructions

Dual Mode Architecture

- To protect the system from aberrant users, some instructions are restricted to use only by the operating system. Users (I.e. in *user mode*) may not:
  - Address I/O directly
  - Use instructions that manipulate the state of memory (page table pointers, TLB load, etc.)
  - Set the mode bits that determine user or kernel mode
  - Halt the machine
- In *kernel mode*, the OS can do all these things.

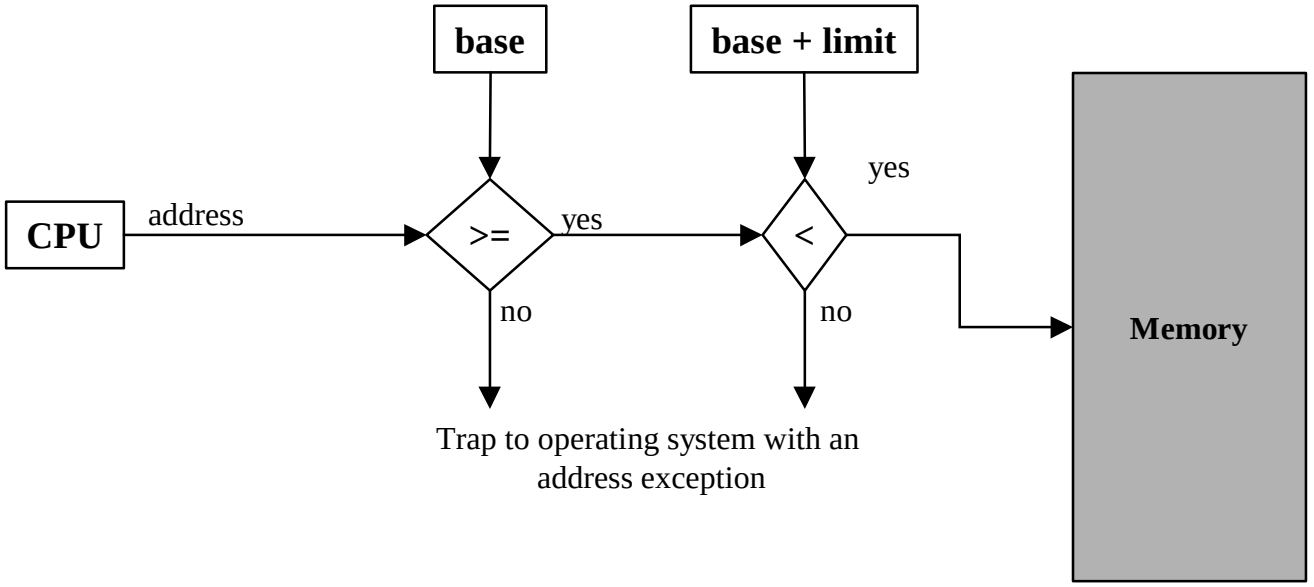
Dual Mode (Cont.)

- Mode bit added to computer hardware to indicate the current mode: kernel (0) or user (1)
- When an interruption occurs (exception or interrupt), hardware switches to kernel mode.



- Further discussion of crossing protection boundaries is deferred until after the discussion on interruptions.

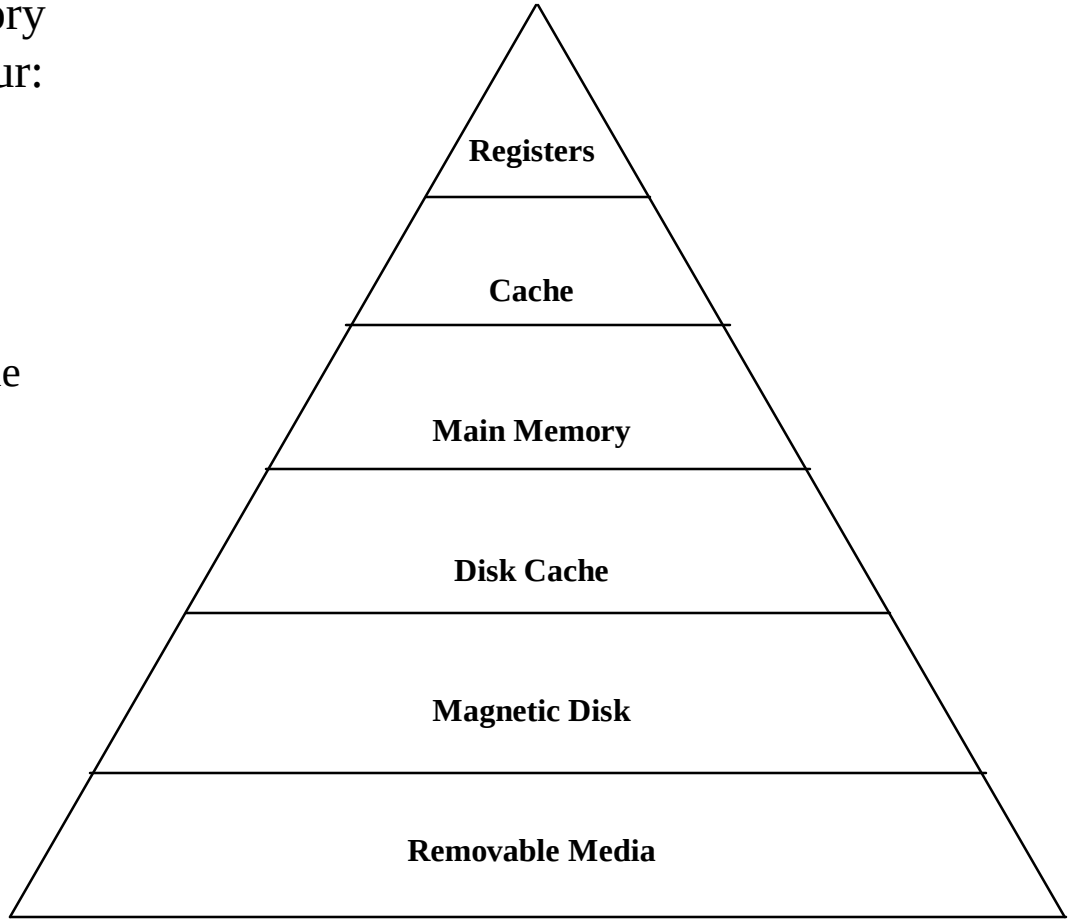
Address Translation



- When executing in kernel mode, the operating system has unrestricted access to both the OS memory and the users' memory
- The load instructions for the base and limit registers must be privileged instructions

The Memory Hierarchy

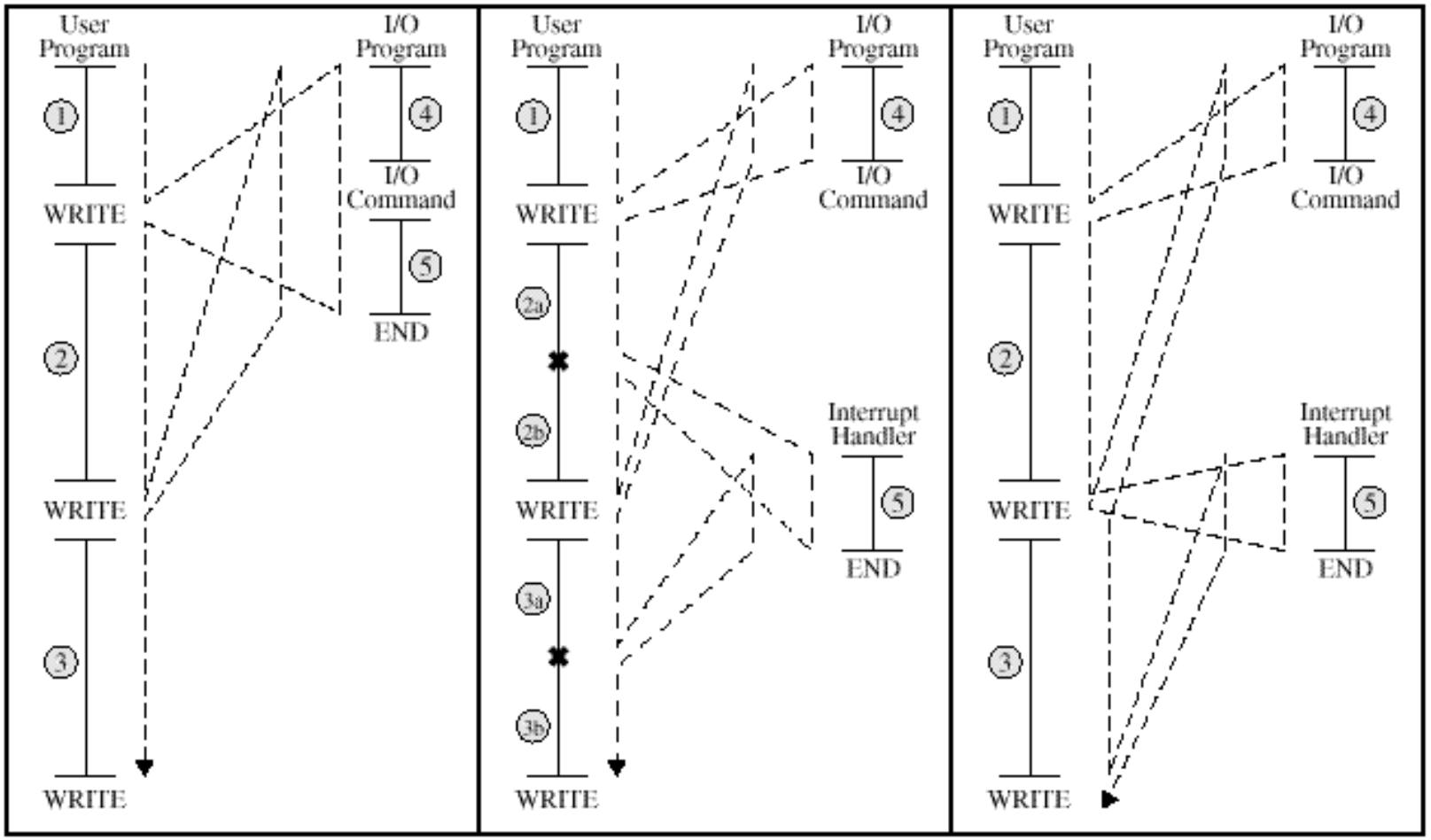
- As one goes down the memory hierarchy, the following occur:
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access of the memory by the processor



Interruptions

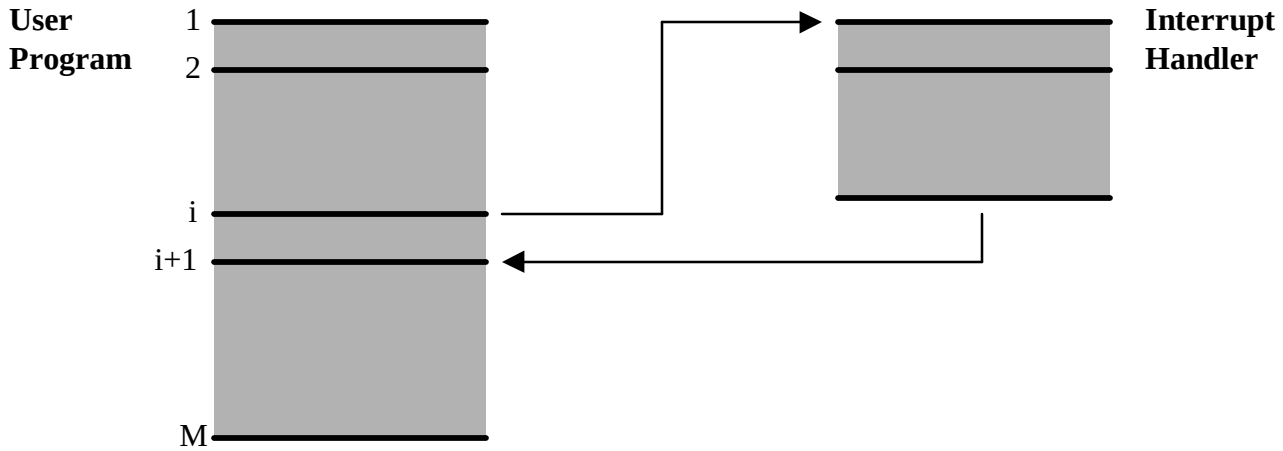
- Interruption – an event in a computer system disrupting the normal instruction execution flow.
- Interruption Classes
  - Program, called *Exception* or *Trap*
    - Synchronous with the instruction stream
    - Arithmetic overflow, divide by zero, illegal instruction, illegal memory reference, etc.
  - Timer, called *Interrupt*
    - Asynchronous
    - Interval timer
  - I/O, called *Interrupt*
    - Asynchronous
    - I/O normal completion or error condition (abnormal completion)
  - Hardware Failure

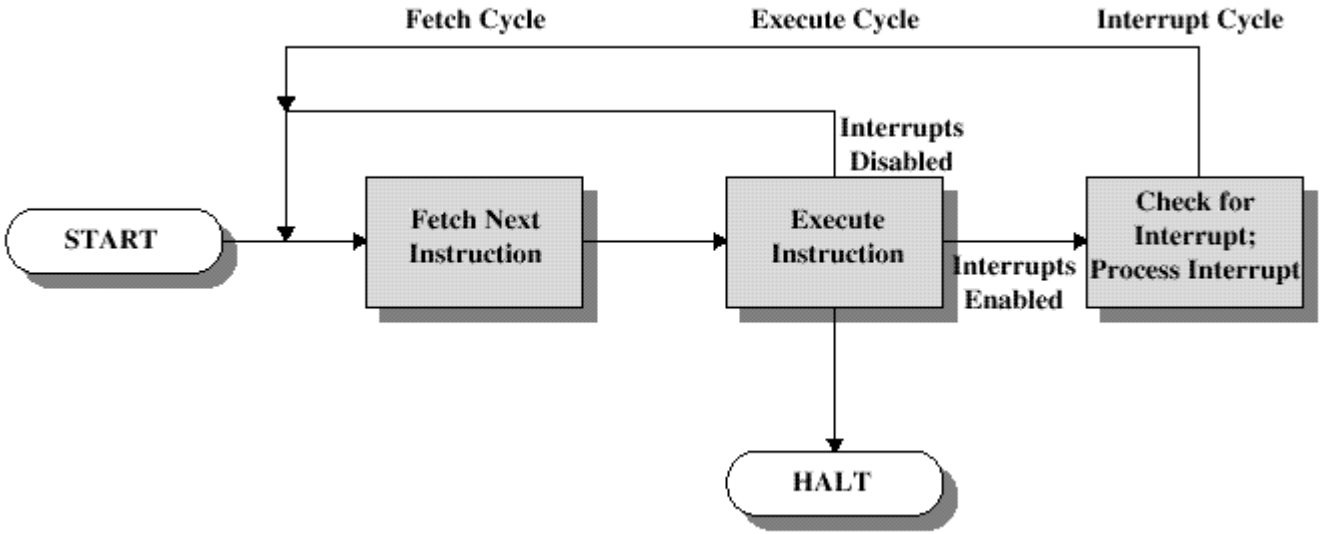
Program Flow



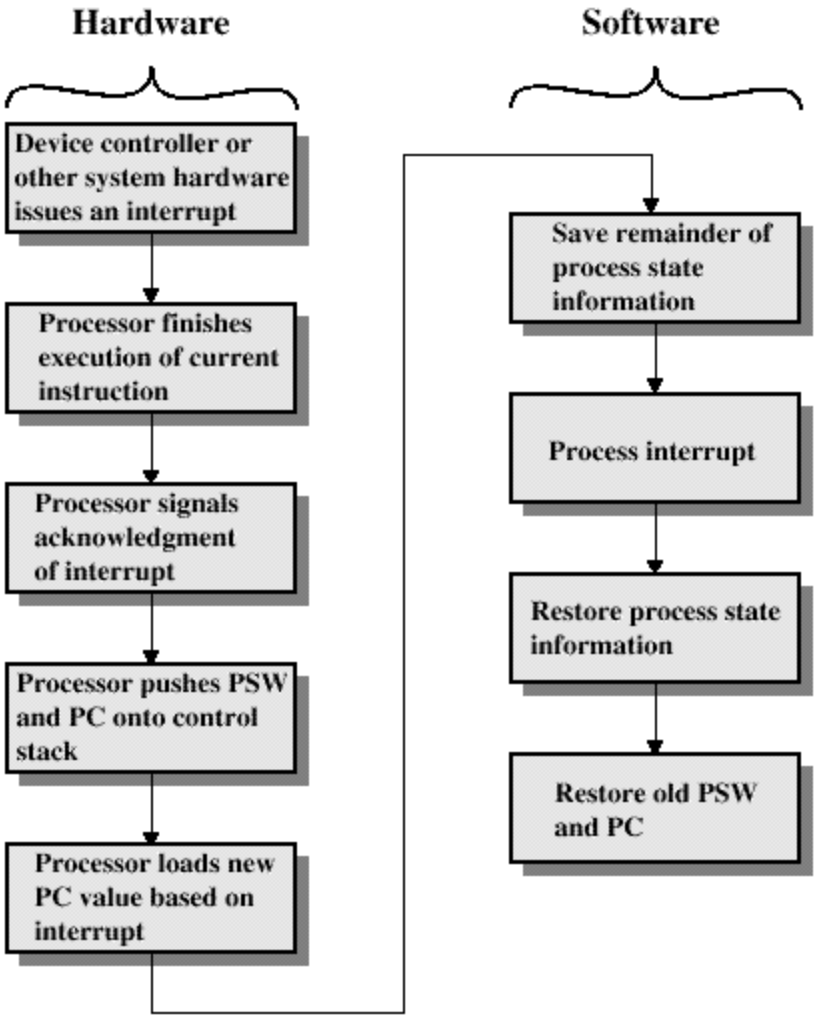
(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

Instruction Control Flow and Instruction Cycle with Interrupts

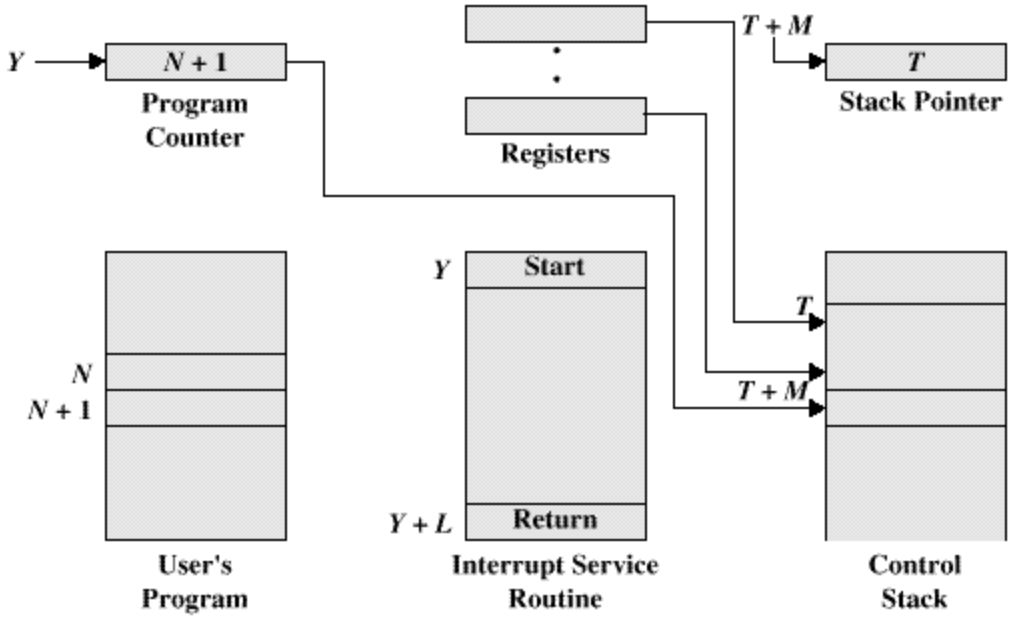




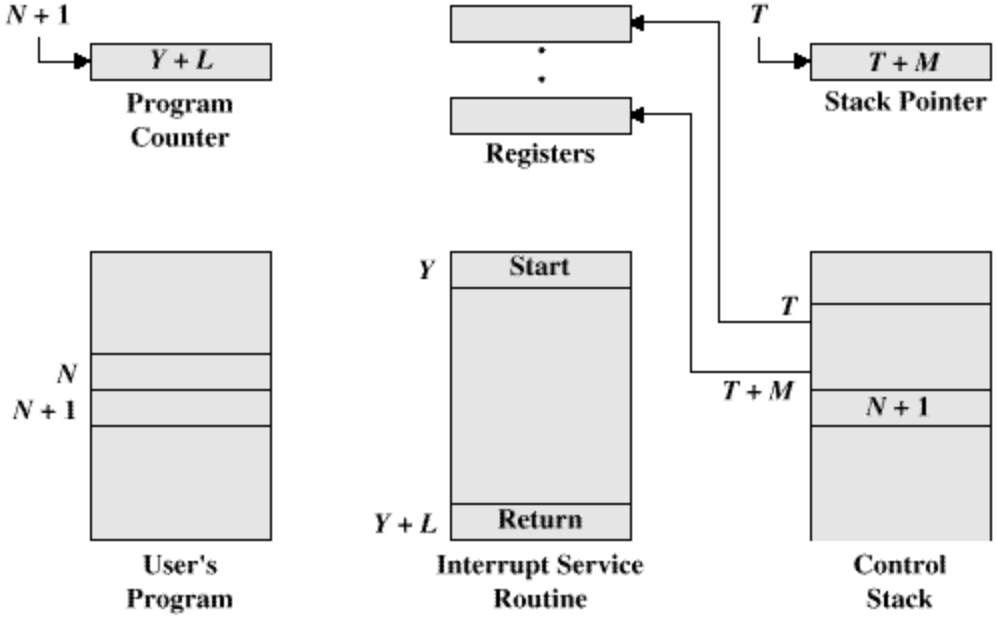
Simple Interrupt Processing



Interrupt Example



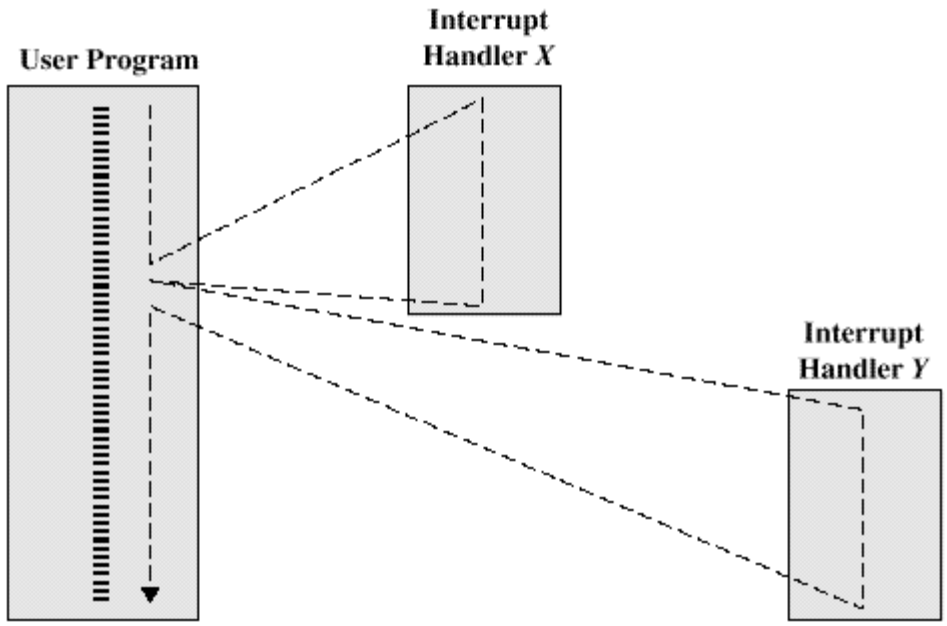
(a) Interrupt occurs after instruction at location *N*



(b) Return from interrupt

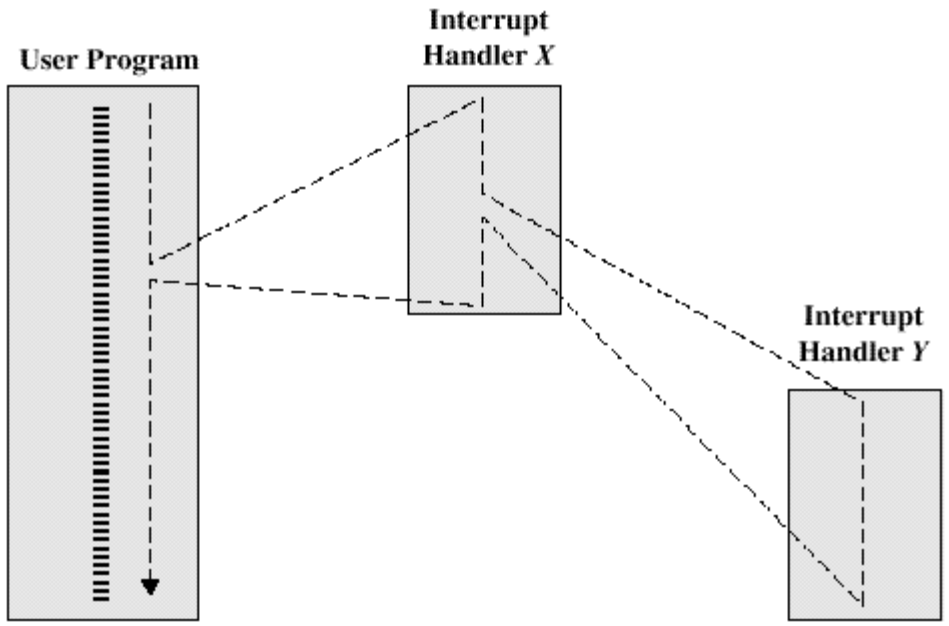
Multiple Interrupts

- Two approaches
  - Disable interrupts while an interrupt is being processed.
  - Sequential interrupt processing



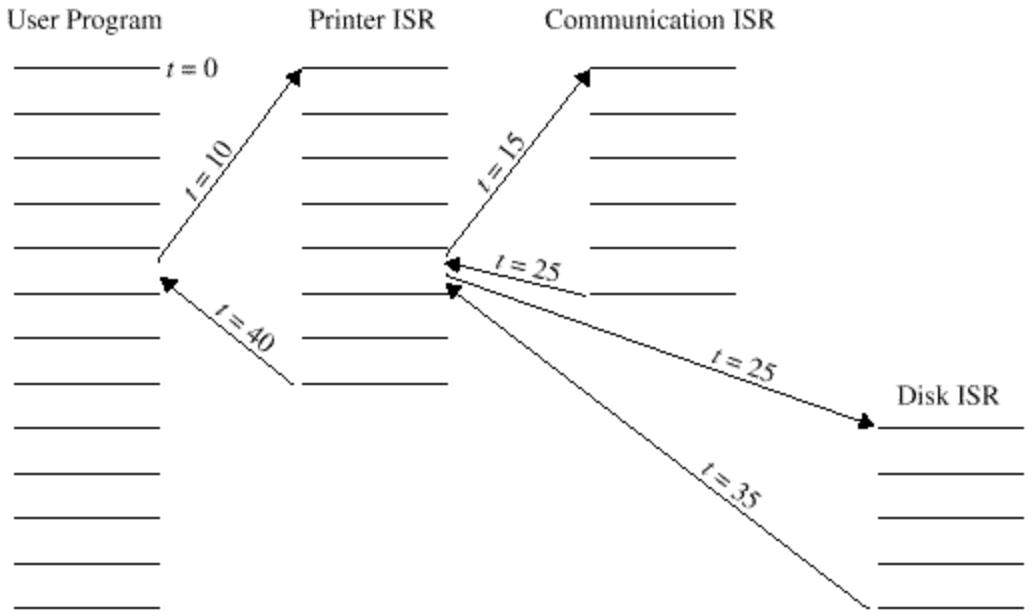
Nested Interrupts

- Define priorities for interrupts and allow an interrupt of higher priority to cause a lower priority interrupt handler to be interrupted.



Nested Interrupt Example

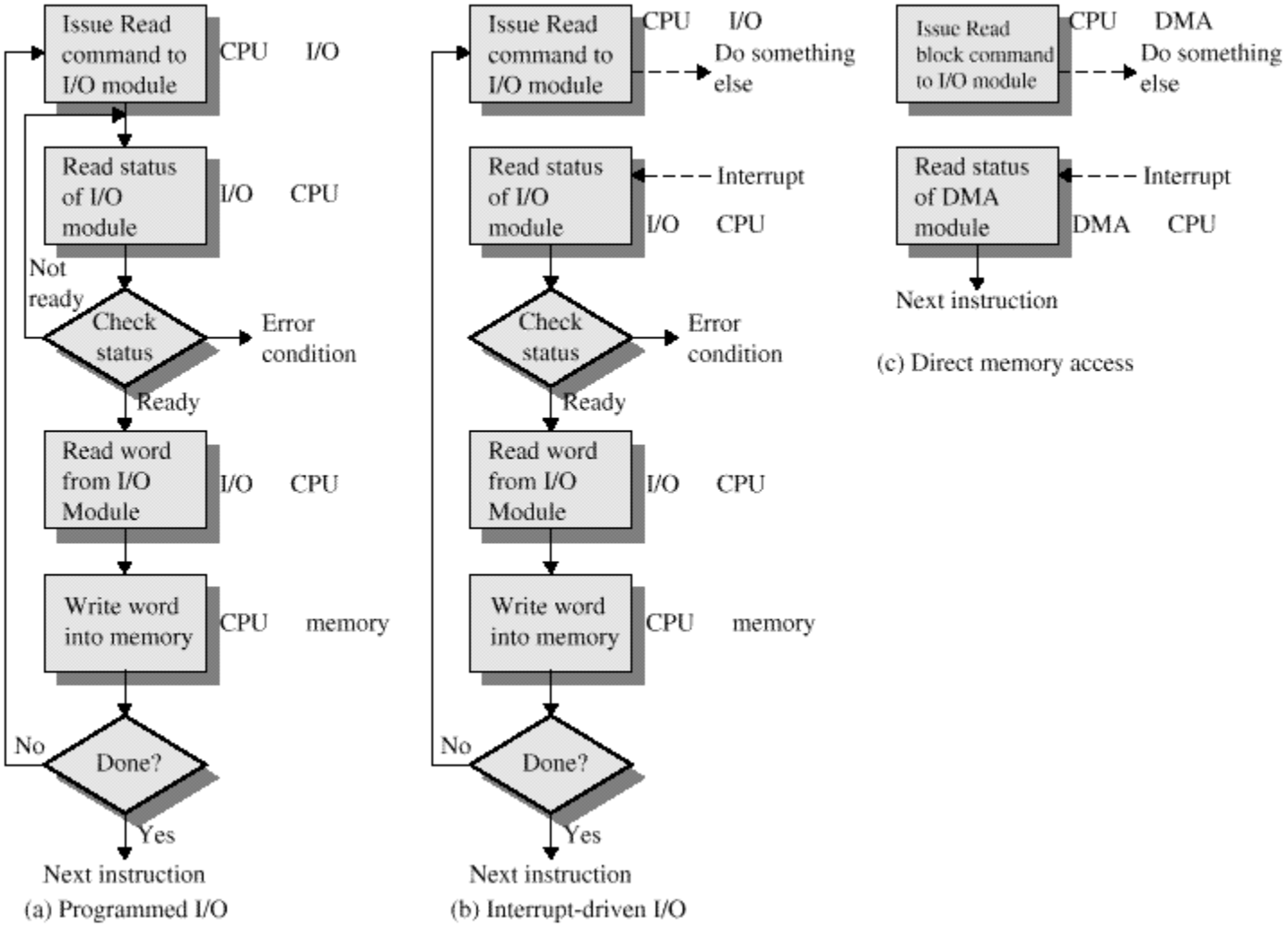
– Assume a printer, a disk, and a comm line, with priorities of 2, 4, and 5 respectively. (Bigger number is more important).



I/O Communication

- Two techniques for interface:
  - I/O Instructions
  - "Memory-mapped I/O"
- Three techniques for controlling the operation of the I/O from the CPU:
  - Programmed I/O
    - I/O instruction categories
      - Control
      - Test
      - Read, write
  - Interrupt-driven I/O
  - Direct memory access (DMA)

Block Input Example



CPU Protection – The Interval Timer

- Interval Timer – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented on every clock tick
  - When timer reaches the value 0, an interrupt occurs
- Timer is commonly used to implement time sharing
- Timer is also used to compute the current time.
- Load timer must be a privileged instruction.

A Hypothetical HW Interface

- Consider a hypothetical system consisting of:
  - A CPU with an interval timer
  - Memory
  - A Disk Controller

General-Purpose Registers

- r0: always loads 0, store is a no-op
- r1: return value from procedures
- r8: first parameter to a function call (or system call)
- r9 to r11: second, third, and fourth parameters to a function call (or system call)
- r29: the frame pointer
- r30: the stack pointer
- r31: the return address from a procedure call

Control Registers

- ia: the instruction address register contains the address of the next instruction
- psw: the program status word. Bit 0 is processor mode, Bit 1 is interrupt enable
- base: the memory base register is added to all addresses when the system is in user mode
- bound: the memory bound register is the address limit (user mode).
- iia: the interrupt instruction address register stores the value of the ia register before an exception.
- ipsw: the interrupt program status word
- ip: the interrupt parameter
- iva: the interrupt vector address register
- timer: the interval timer register

Instruction Set

- load
- store
- loadAll
- storeAll
- move
- syscall
- rti

Interrupt Example

System Call Example