UTN FRD – Sistemas Operativos
Clase I

Basic Elements

- Processor
  - Two internal registers
    - Memory address resister (MAR)
      - Specifies the address for the next read or write
    - Memory buffer register (MBR)
      - Contains data written into memory or receives data read from memory

Basic Elements

- Processor
  - I/O address register
  - I/O buffer register

Basic Elements

- Main Memory
  - Volatile
  - Referred to as real memory or primary memory

Basic Elements

- I/O Modules
  - Secondary Memory Devices
  - Communications equipment
  - Terminals
- System bus
  - Communication among processors, main memory, and I/O modules

Computer Components: Top-Level View

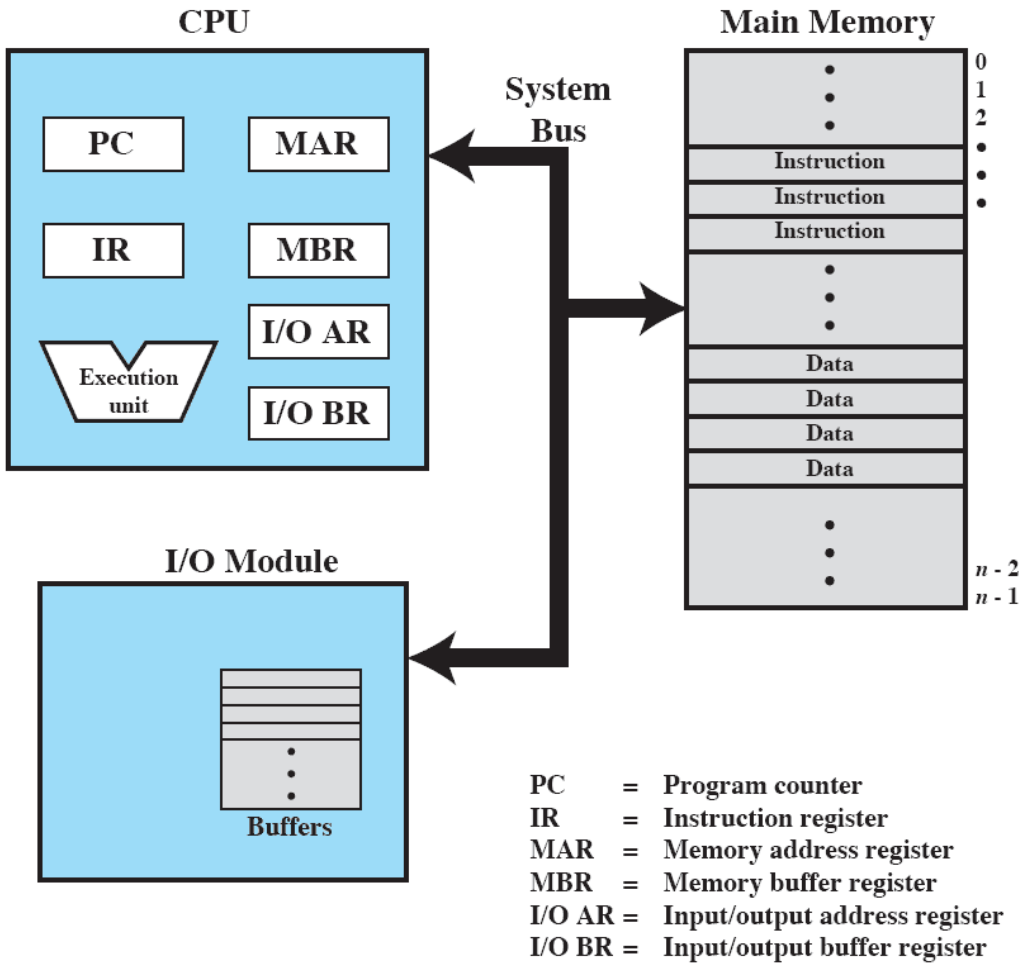
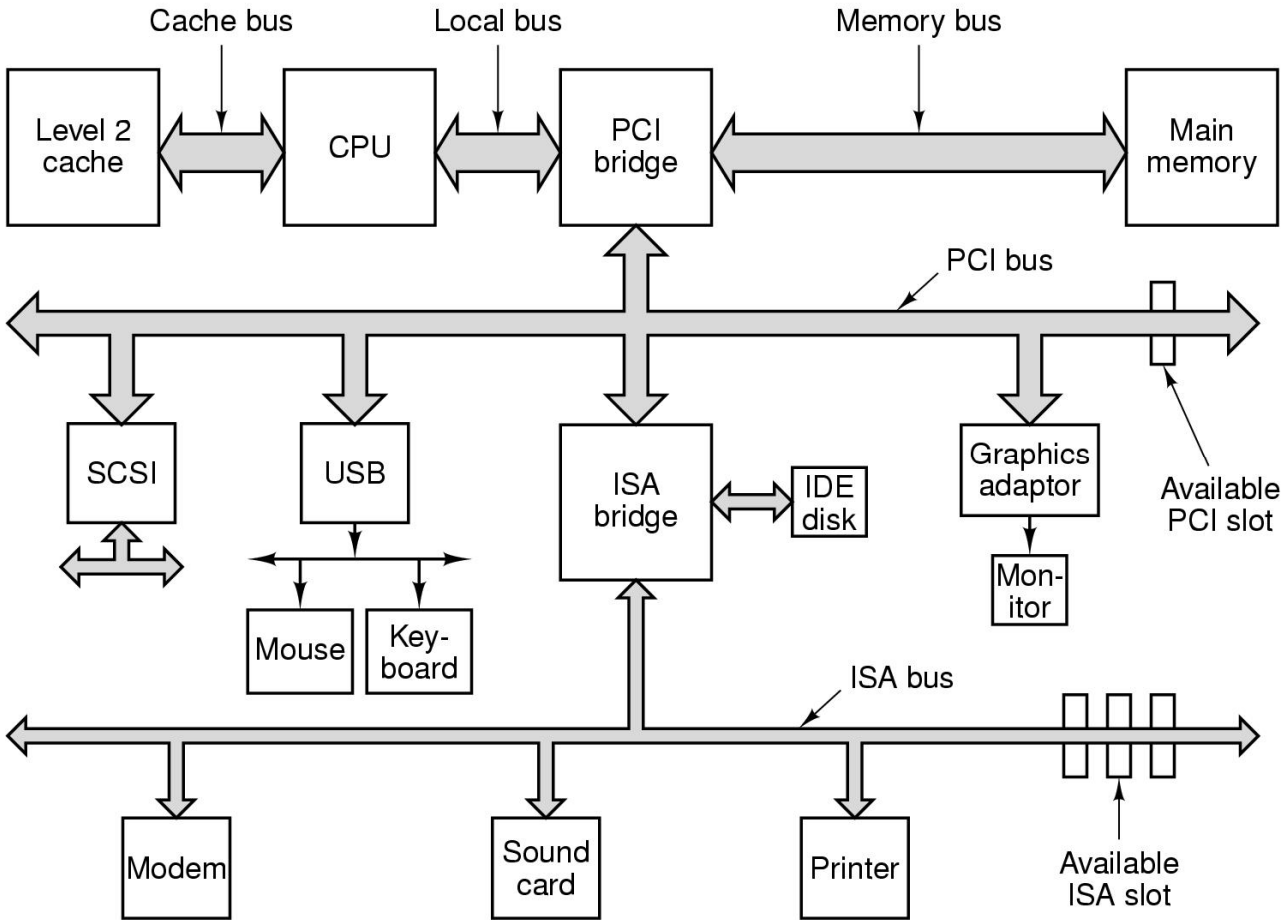


Figure 1.1 Computer Components: Top-Level View

Computer Hardware Review



Structure of a large Pentium system

Processor Registers

- User-visible registers
  - Enable programmer to minimize main memory references by optimizing register use
- Control and status registers
  - Used by processor to control operating of the processor
  - Used by privileged OS routines to control the execution of programs

User-Visible Registers

- May be referenced by machine language
- Available to all programs – application programs and system programs

User-Visible Registers

- Data
- Address
  - Index register: Adding an index to a base value to get the effective address
  - Segment pointer: When memory is divided into segments, memory is referenced by a segment and an offset
  - Stack pointer: Points to top of stack

Control and Status Registers

- Program counter (PC)
  - Contains the address of an instruction to be fetched
- Instruction register (IR)
  - Contains the instruction most recently fetched
- Program status word (PSW)
  - Contains status information

Control and Status Registers

- Condition codes or flags
  - Bits set by processor hardware as a result of operations
  - Example
    - Positive, negative, zero, or overflow result

Instruction Execution

- Two steps
  - Processor reads (fetches) instructions from memory
  - Processor executes each instruction

Basic Instruction Cycle

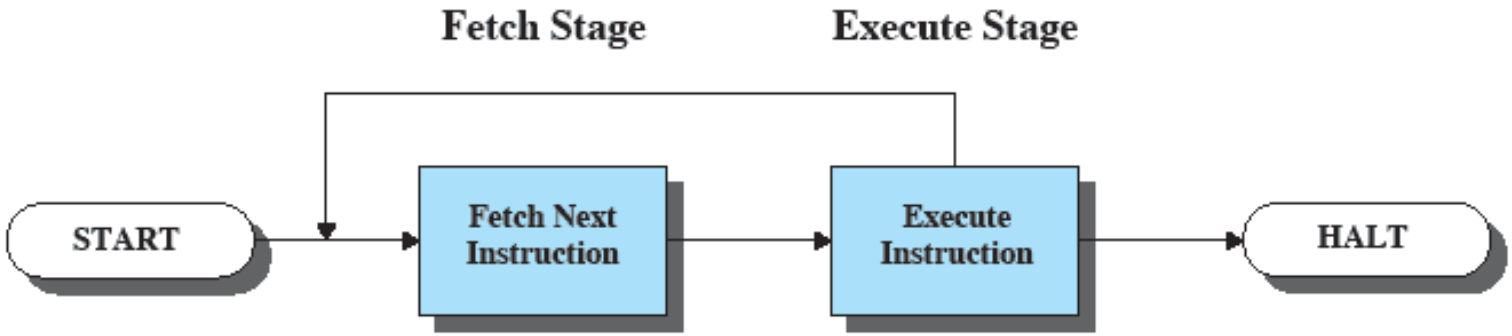


Figure 1.2 Basic Instruction Cycle

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
- PC is incremented after each fetch

Instruction Register

- Fetched instruction loaded into instruction register
- Categories
  - Processor-memory, processor-I/O, data processing, control

Example of Program Execution

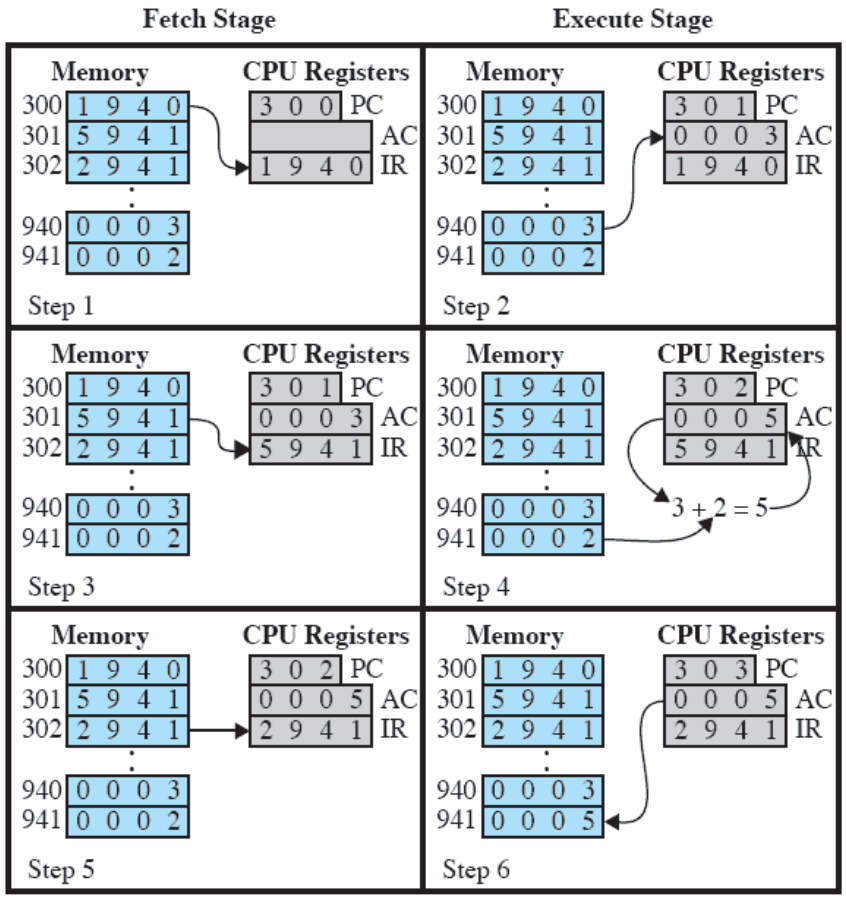


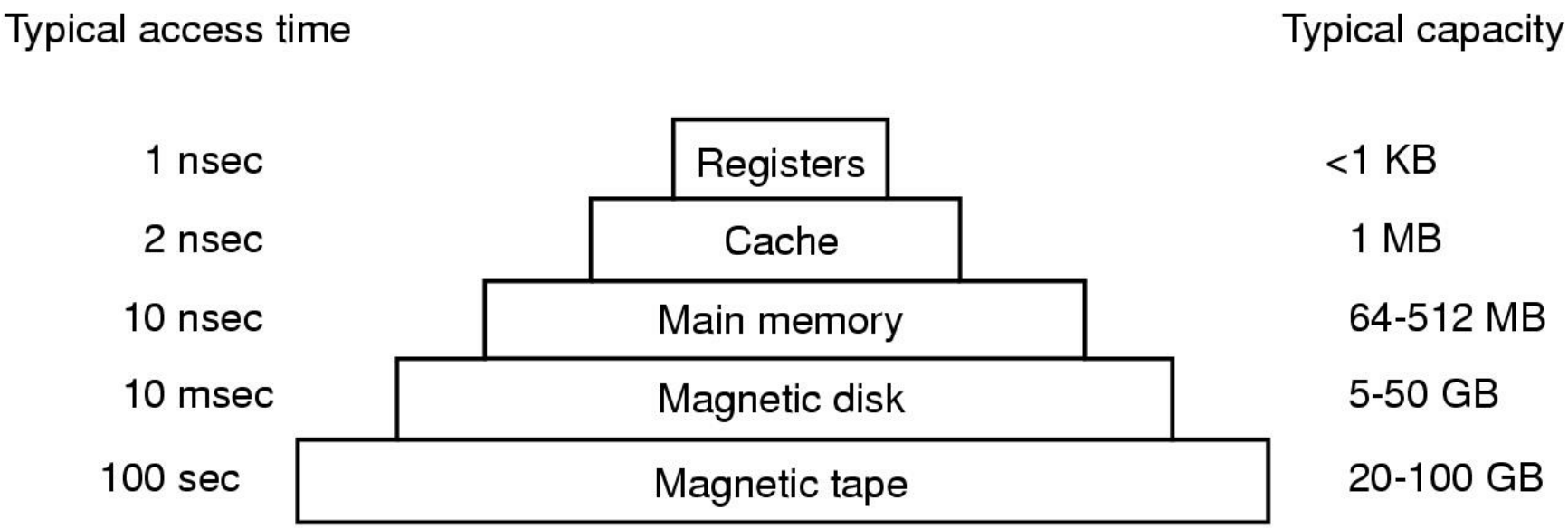
Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

Metric Units

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

The metric prefixes

Computer Hardware Review



- Typical memory hierarchy
  - numbers shown are rough approximations

Interrupts

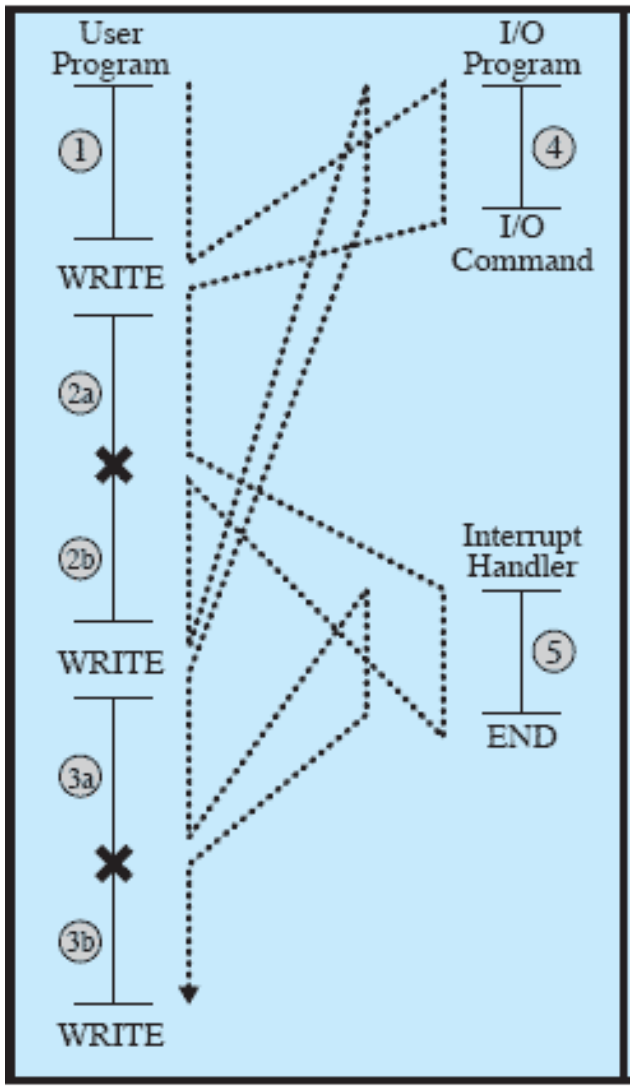
- Interrupt the normal sequencing of the processor
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device

Classes of Interrupts

Table 1.1 Classes of Interrupts

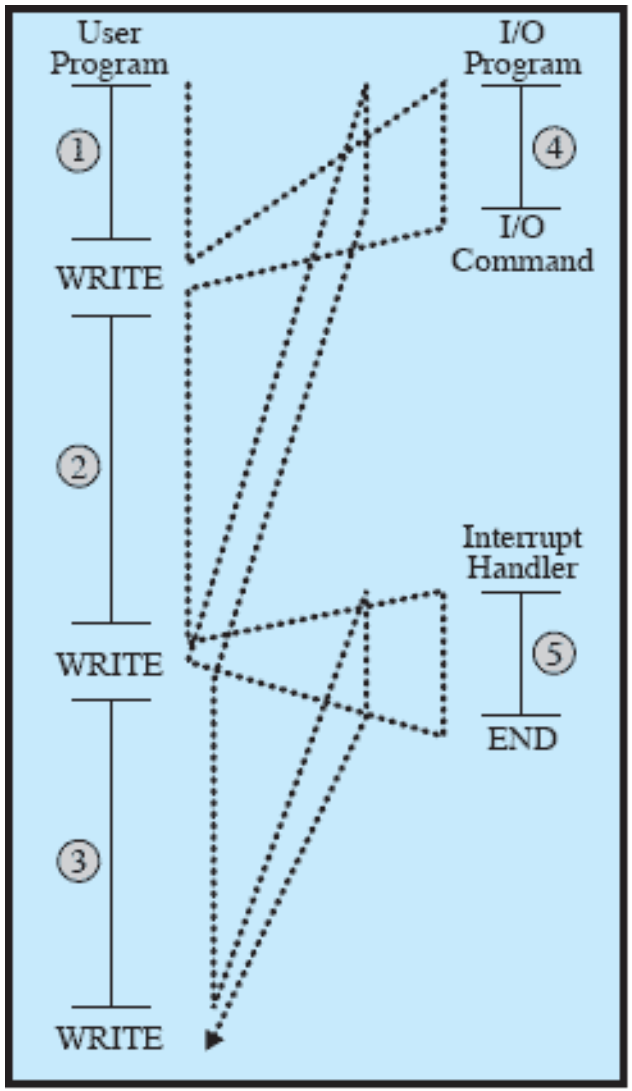
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Program Flow of Control



(b) Interrupts; short I/O wait

Program Flow of Control



(c) Interrupts; long I/O wait

Interrupt Stage

- Processor checks for interrupts
- If interrupt
  - Suspend execution of program
  - Execute interrupt-handler routine

Transfer of Control via Interrupts

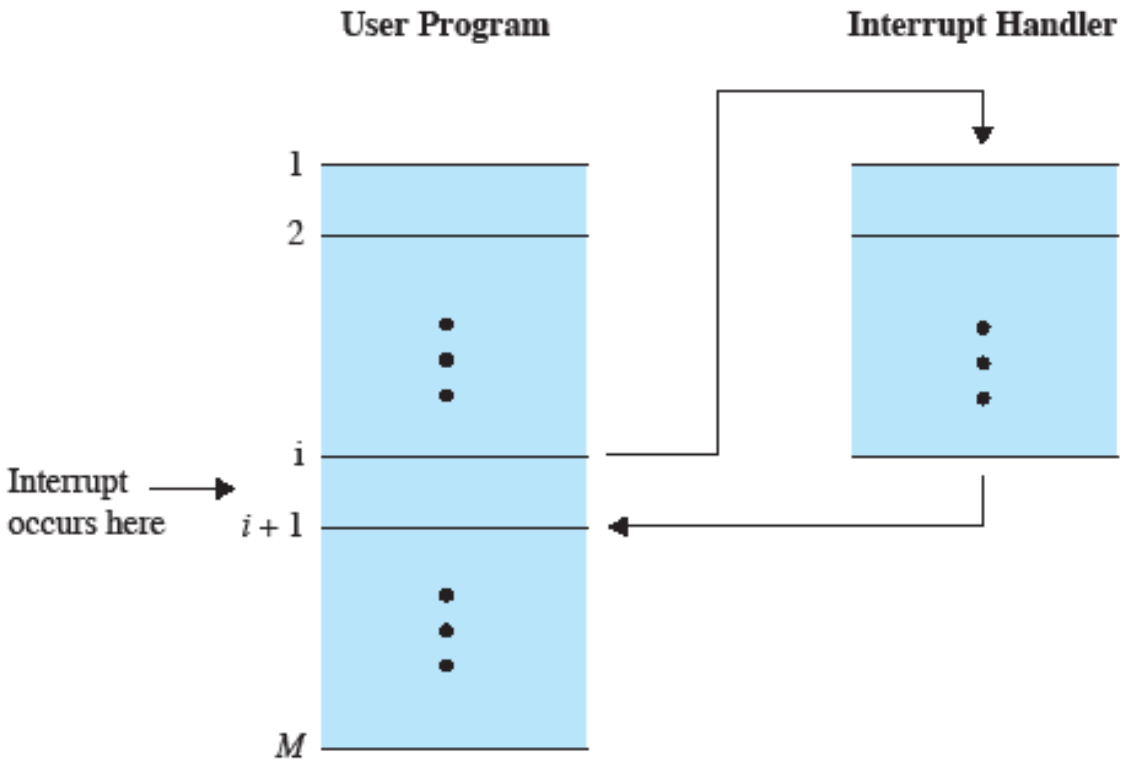


Figure 1.6 Transfer of Control via Interrupts

Instruction Cycle with Interrupts

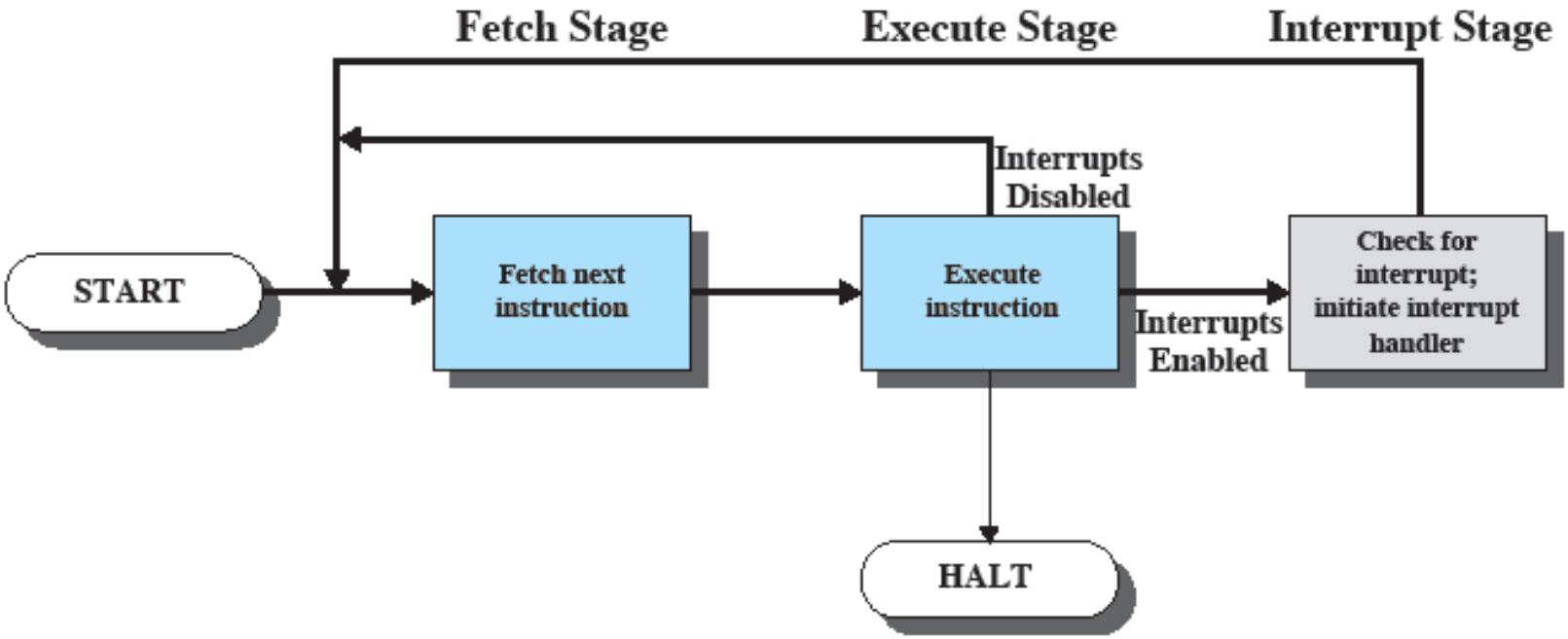
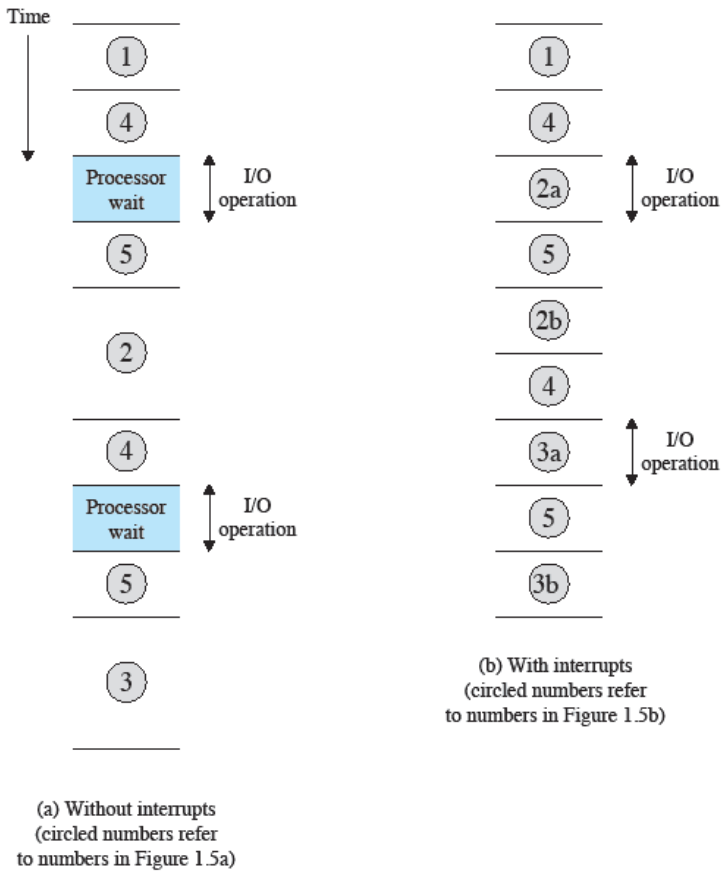


Figure 1.7 Instruction Cycle with Interrupts

Program Timing: Short I/O Wait



(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

(b) With interrupts (circled numbers refer to numbers in Figure 1.5b)

Figure 1.8 Program Timing: Short I/O Wait

Program Timing: Long I/O Wait

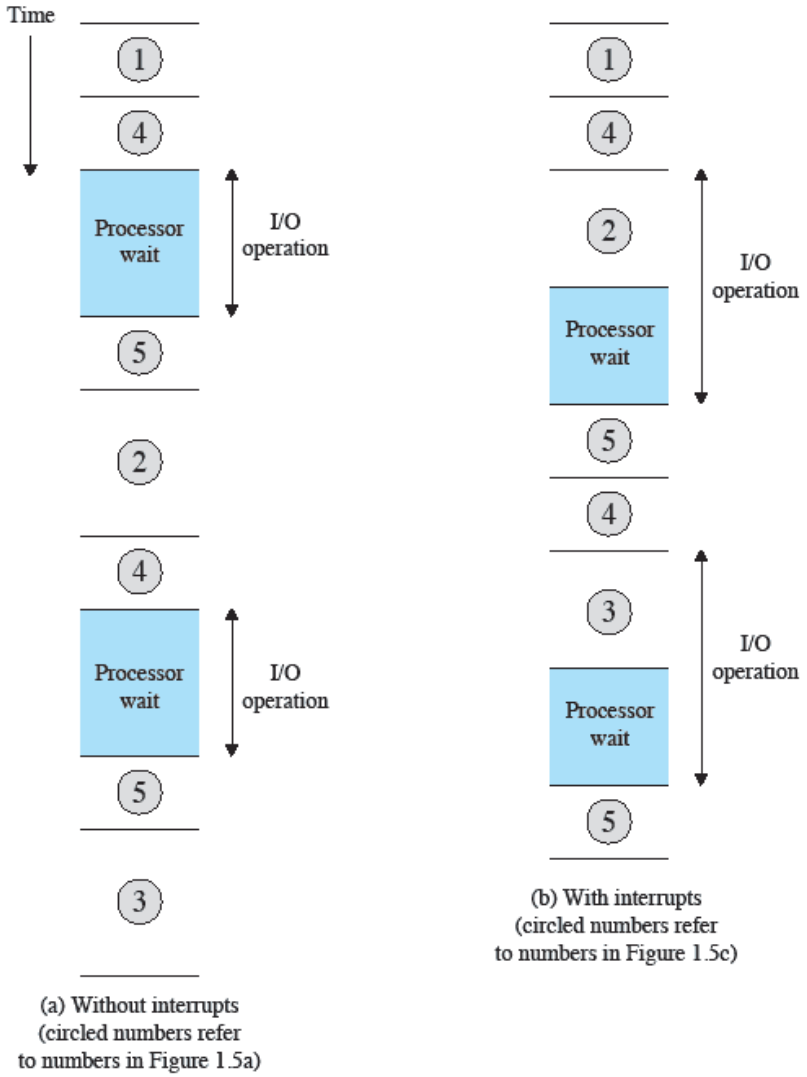


Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

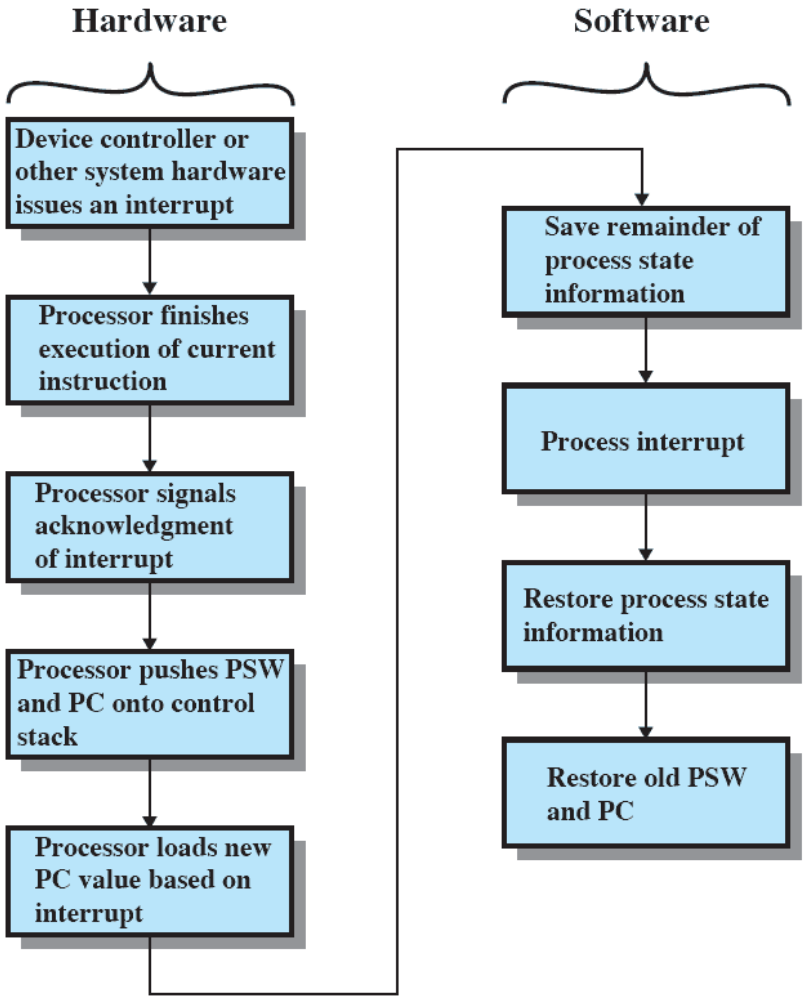
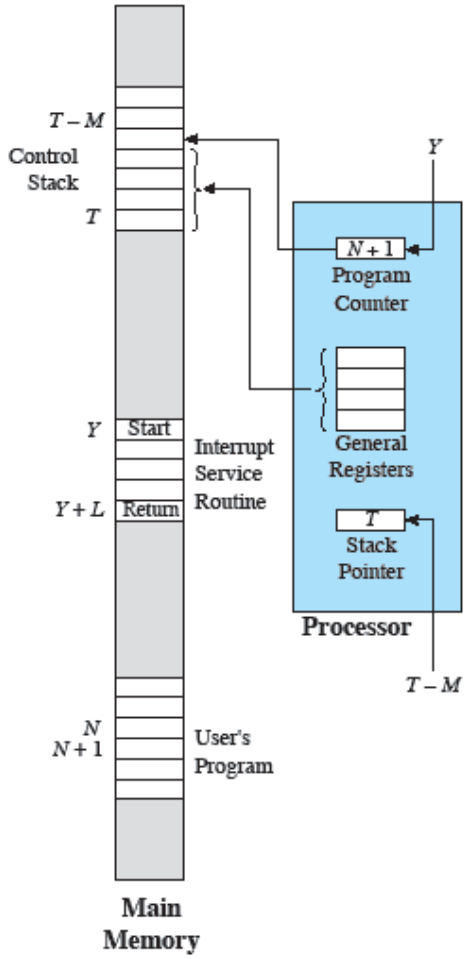


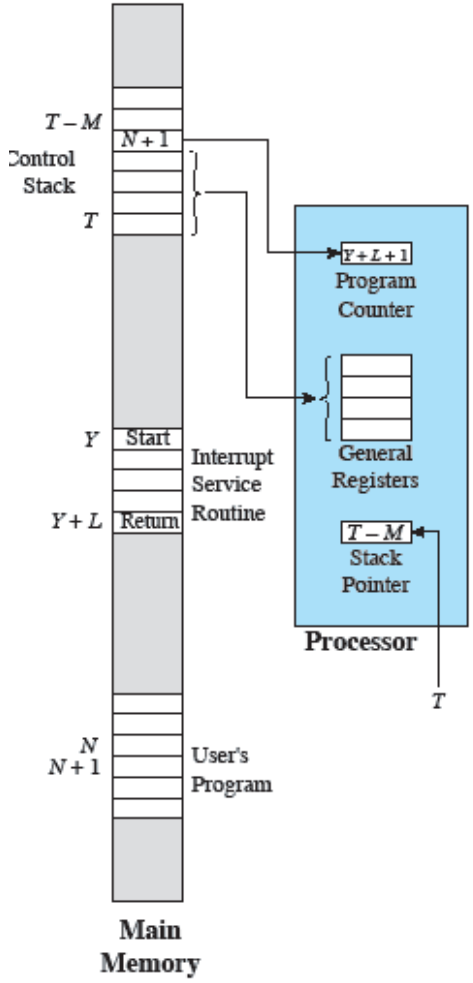
Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt



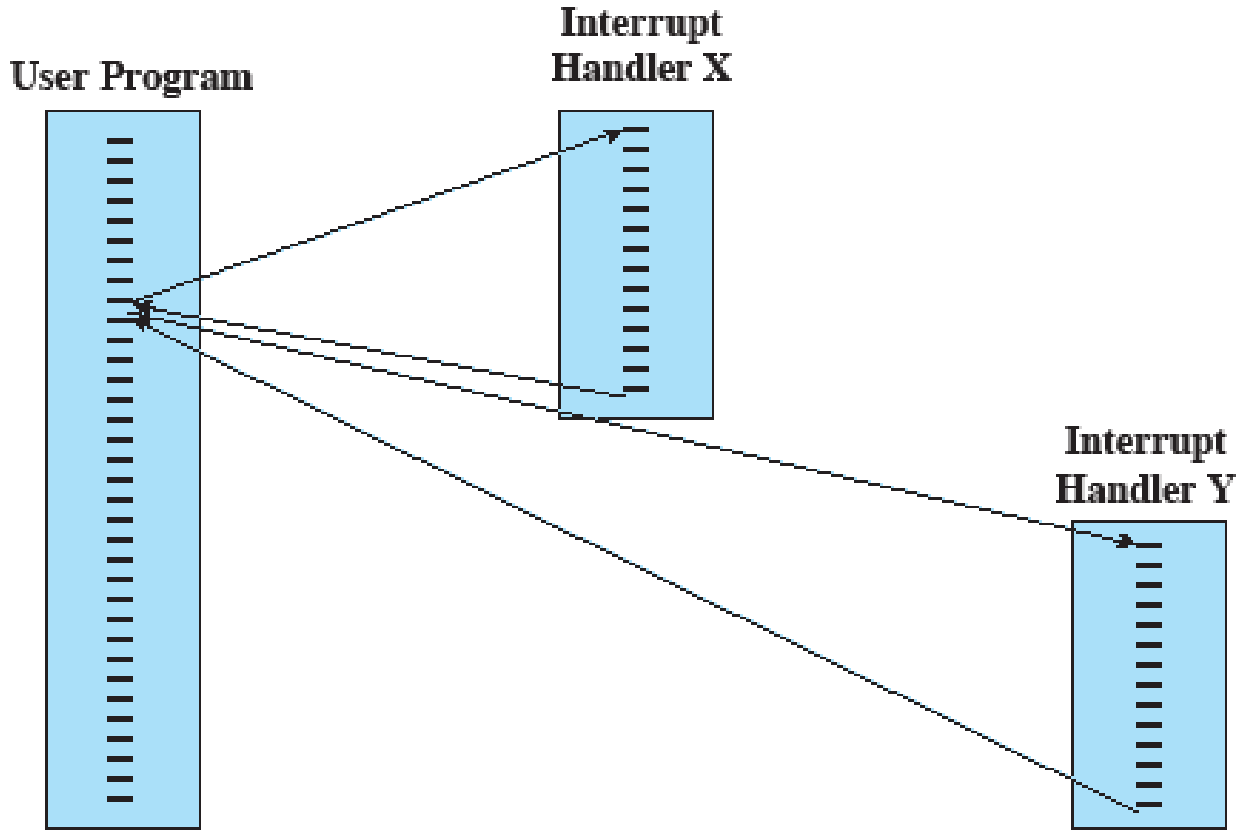
(a) Interrupt occurs after instruction at location N

Changes in Memory and Registers for an Interrupt



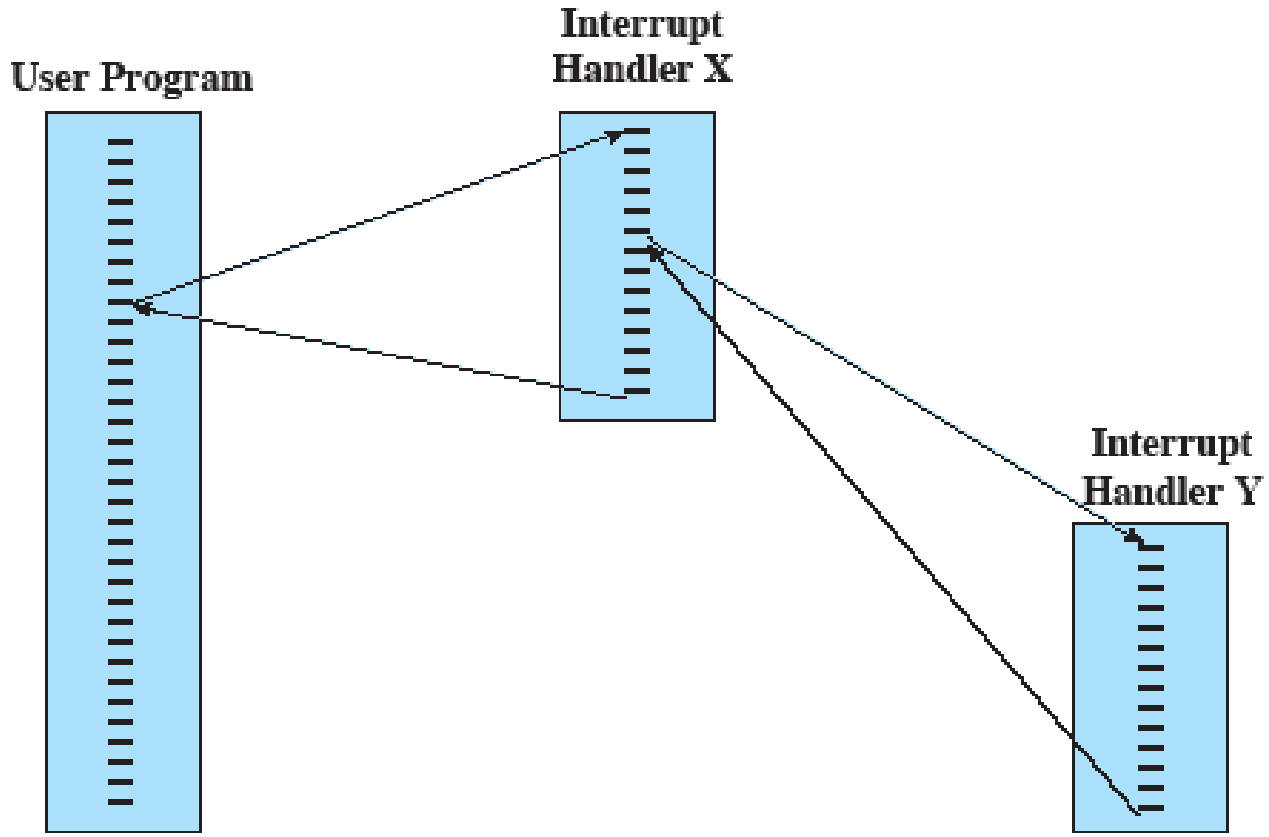
(b) Return from interrupt

Sequential Interrupt Processing



(a) Sequential interrupt processing

Nested Interrupt Processing



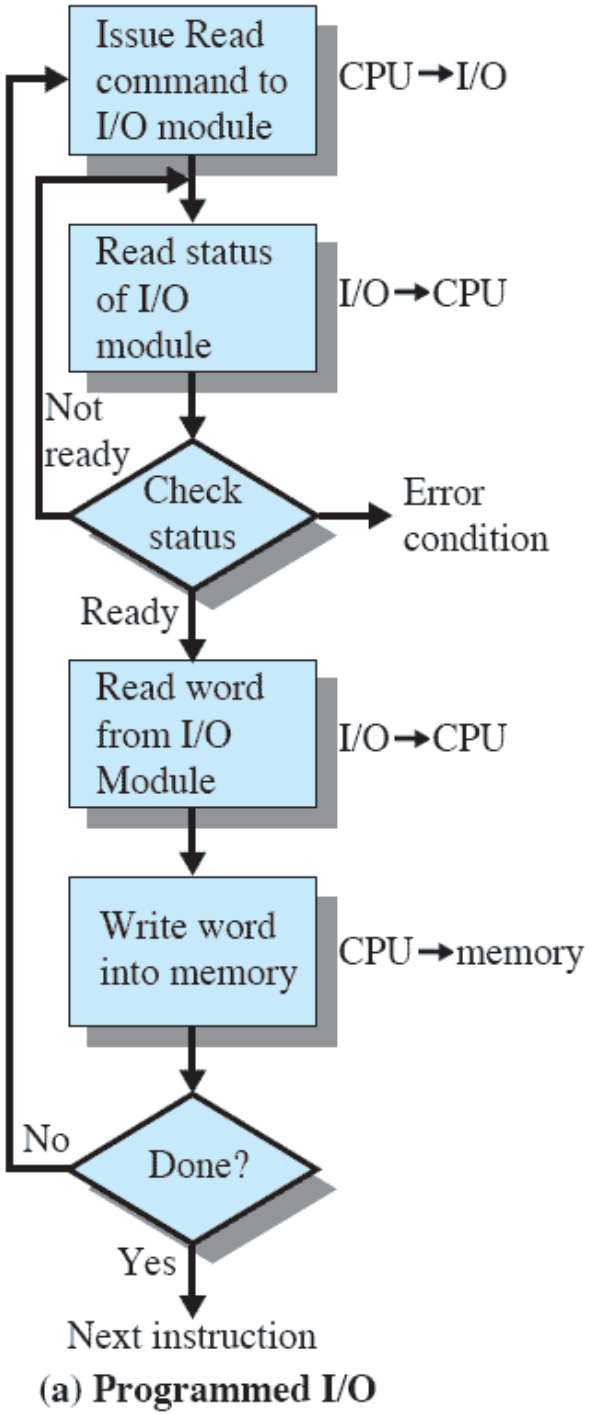
(b) Nested interrupt processing

Multiprogramming

- Processor has more than one program to execute
- The sequence in which programs are executed depend on their relative priority and whether they are waiting for I/O
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt

Programmed I/O

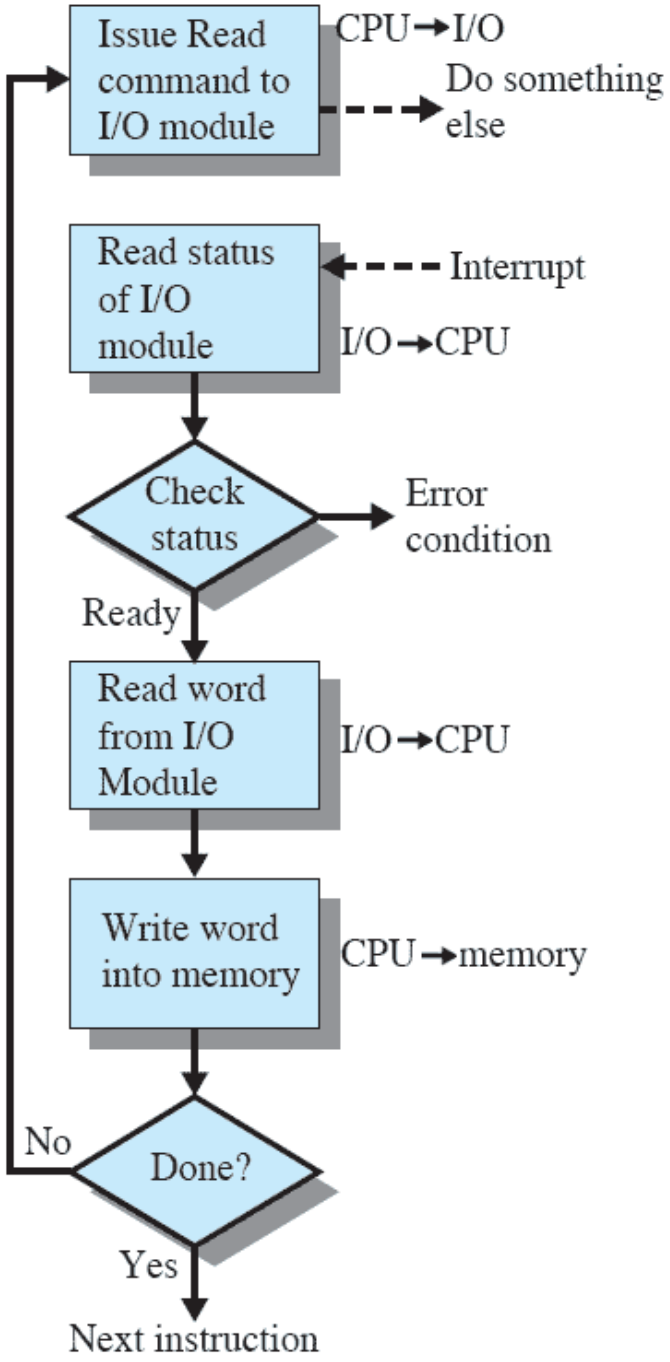
- I/O module performs the action, not the processor
- Sets the appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete



(a) Programmed I/O

Interrupt-Driven I/O

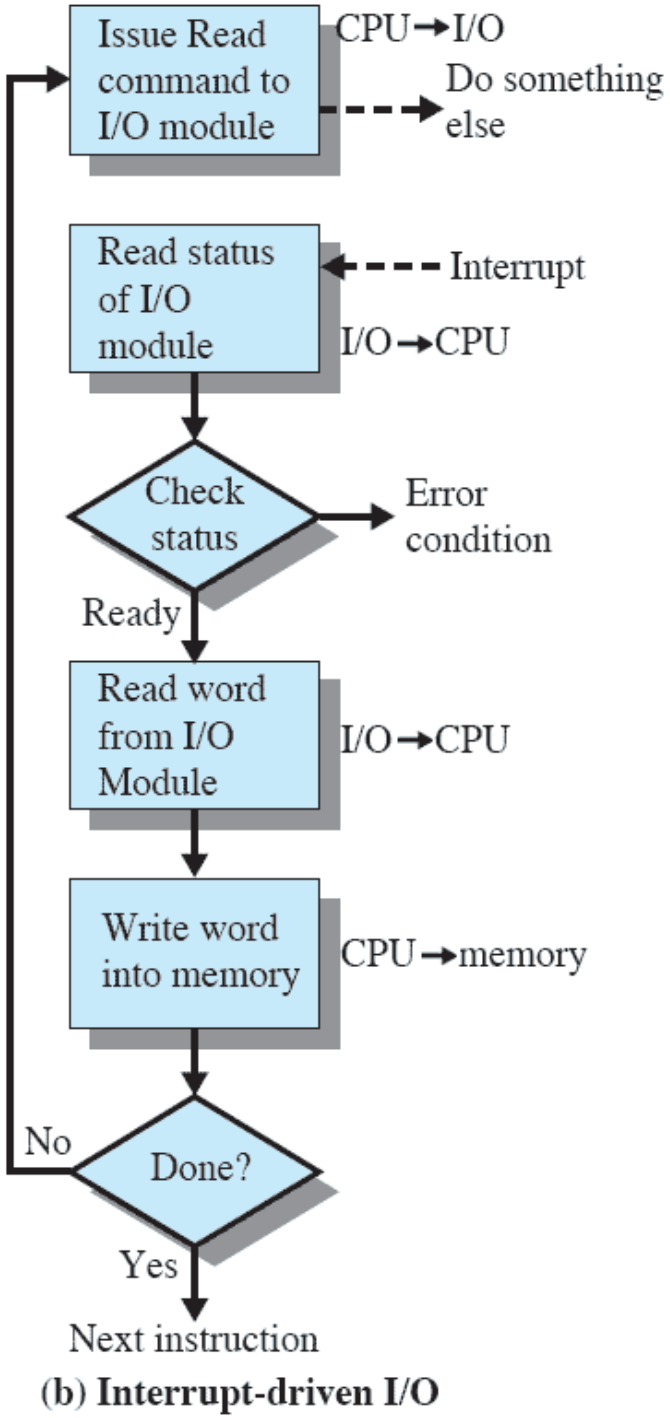
- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler



(b) Interrupt-driven I/O

Interrupt-Driven I/O

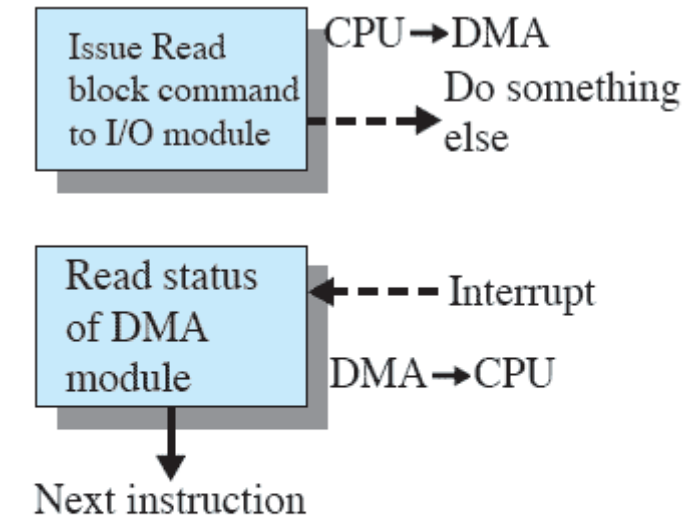
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- More efficient



(c) Direct memory access

* Revision Ejercicios Clase I
Clase II

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware

Layers and Views

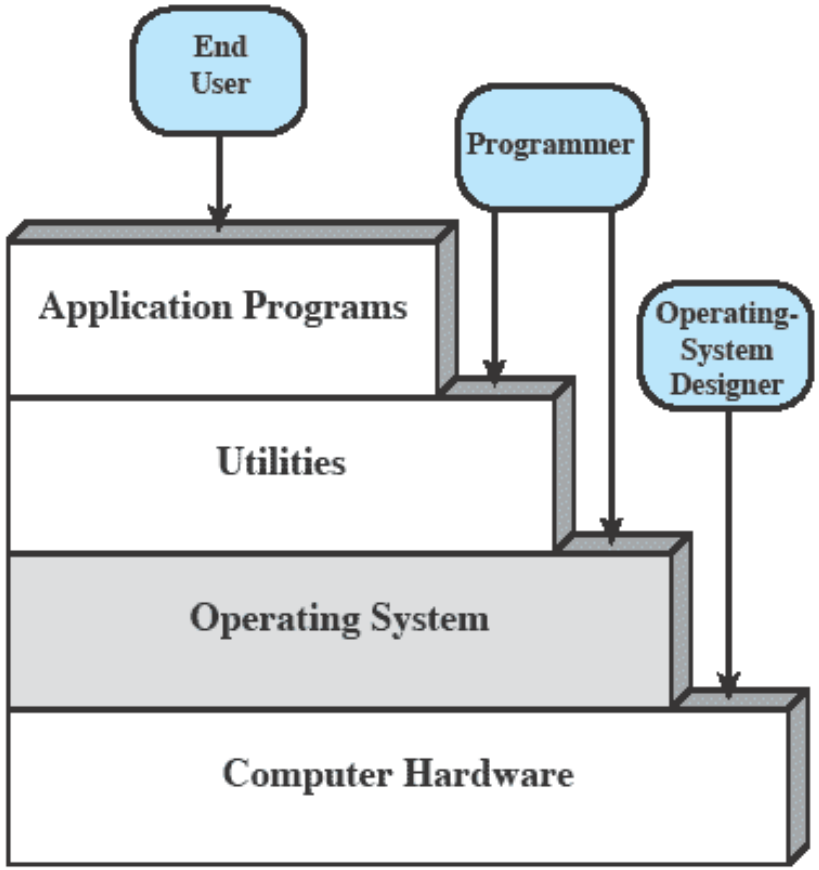


Figure 2.1 Layers and Views of a Computer System

Services Provided by the OS

- Program development
  - Editors and debuggers
- Program execution
- Access I/O devices

Services Provided by the OS

- Controlled access to files
- System access

Services Provided by the OS

- Error detection and response
  - Internal and external hardware errors
  - Software errors
  - Operating system cannot grant request of application

Services Provided by the OS

- Accounting
  - Collect usage statistics
  - Monitor performance
  - Used to anticipate future enhancements
  - Used for billing purposes

Operating System

- Responsible for managing resources
- Functions same way as ordinary computer software
  - It is a program that is executed
- Operating system relinquishes control of the processor

OS as Resource Manager

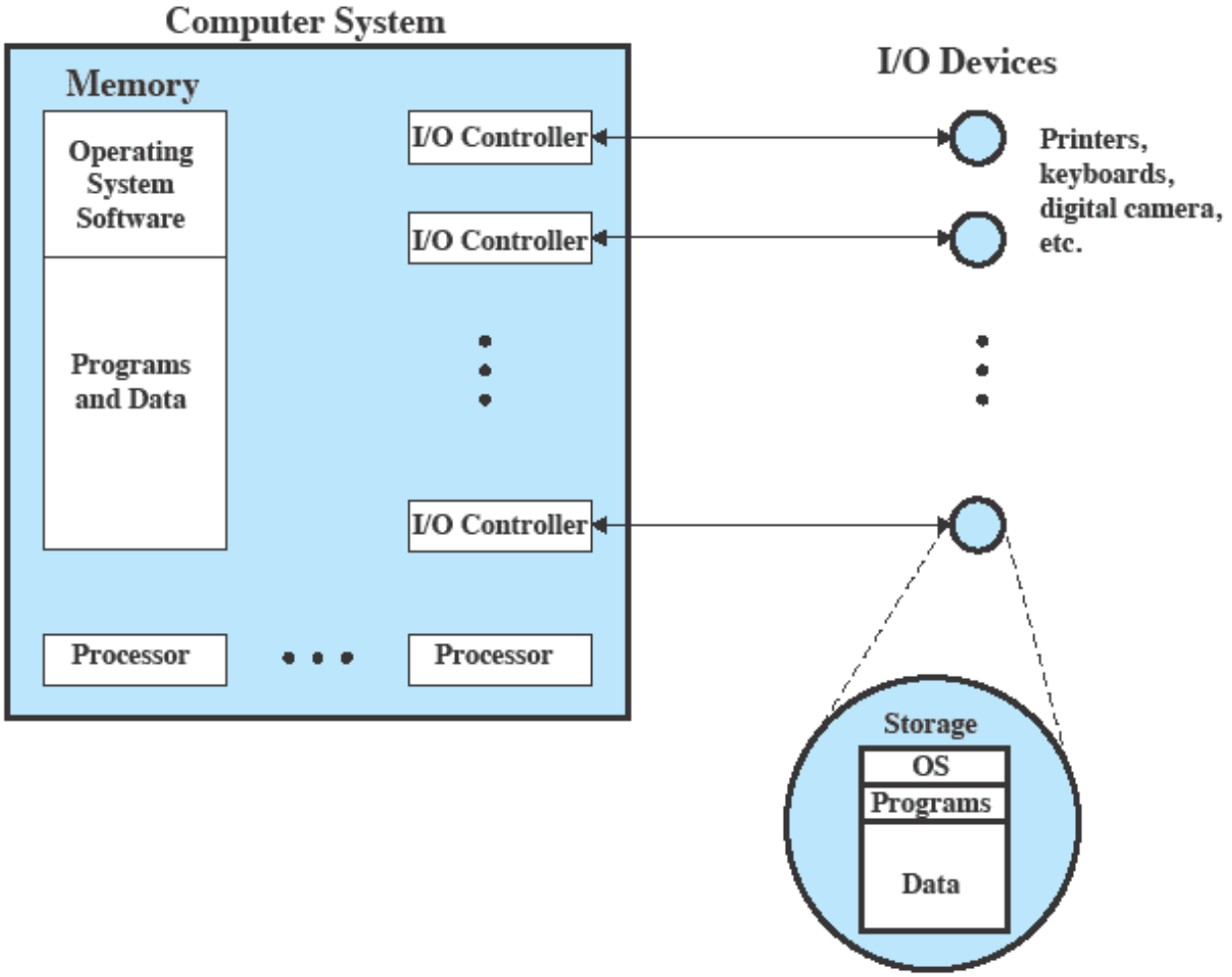


Figure 2.2 The Operating System as Resource Manager

Kernel

- Portion of operating system that is in main memory
- Contains most frequently used functions
- Also called the nucleus

Evolution of Operating Systems

- Hardware upgrades plus new types of hardware
- New services
- Fixes

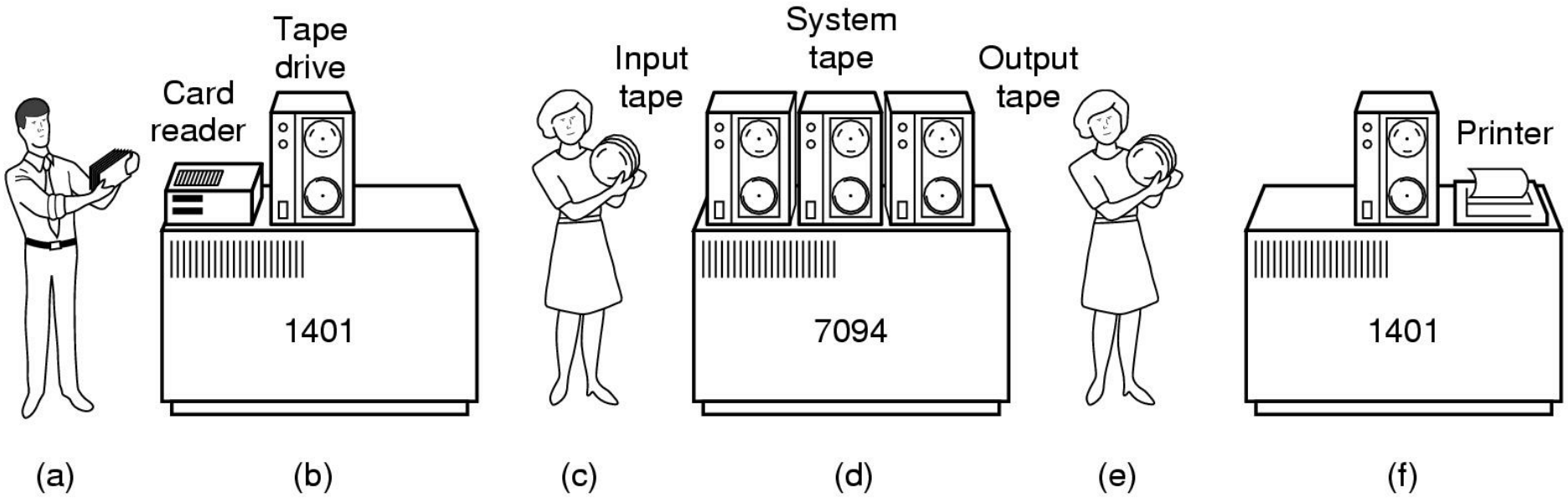
Evolution of Operating Systems

- Serial processing
  - No operating system
  - Machines run from a console with display lights, toggle switches, input device, and printer

Evolution of Operating Systems

- Serial processing
  - Schedule time
  - Setup included loading the compiler, source program, saving compiled program, and loading and linking

Serial Processing



Steps:

a) bring cards to 1401

b) read cards to tape

c, d) put tape on 7094 which does computing

e, f) put tape on 1401 which prints output

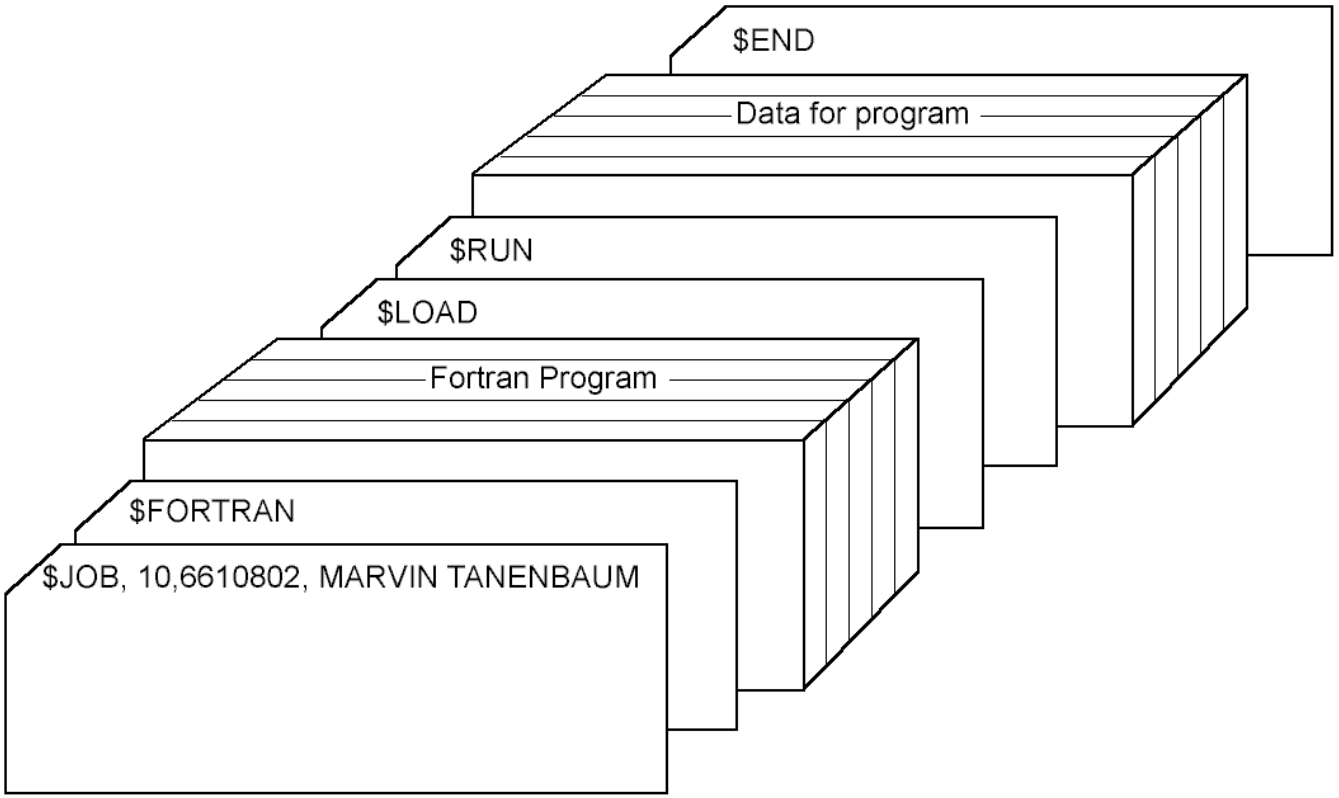
Evolution of Operating Systems

- Simple batch system
  - Monitor
    - Software that controls the sequence of events
    - Batch jobs together
    - Program returns control to monitor when finished

Job Control Language

- Special type of programming language
- Provides instruction to the monitor
  - What compiler to use
  - What data to use

Job Control Language Example



Hardware Features

- Memory protection
  - Does not allow the memory area containing the monitor to be altered
- Timer
  - Prevents a job from monopolizing the system

Hardware Features

- Privileged instructions
  - Certain machine level instructions can only be executed by the monitor
- Interrupts
  - Early computer models did not have this capability

Memory Protection

- User program executes in user mode
  - Certain instructions may not be executed

Memory Protection

- Monitor executes in system mode
  - Kernel mode
  - Privileged instructions are executed
  - Protected areas of memory may be accessed

System Utilization Example

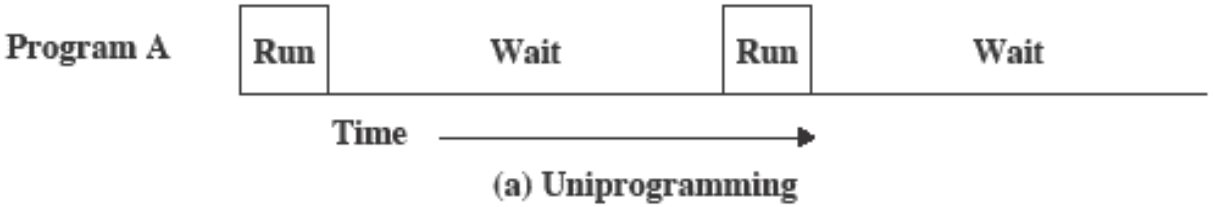
Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

Percent CPU Utilization $= \frac{1}{31} = 0.032 = 3.2\%$

Figure 2.4 System Utilization Example

Uniprogramming

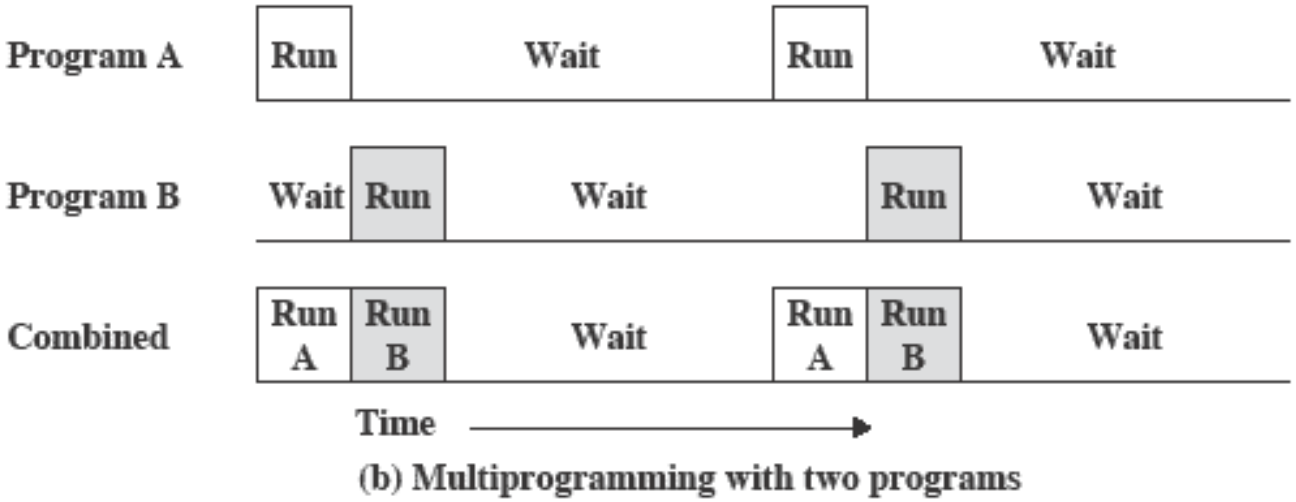
- Processor must wait for I/O instruction to complete before proceeding



(a) Uniprogramming

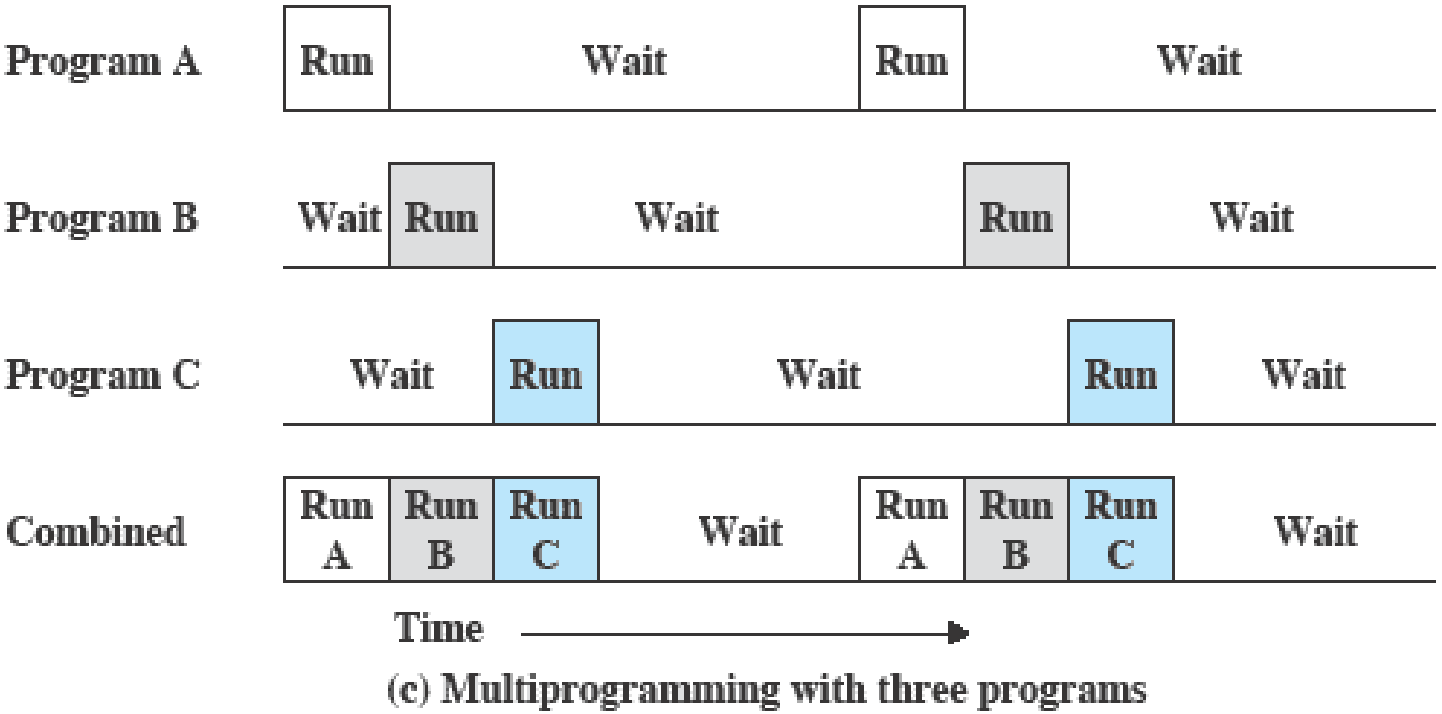
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



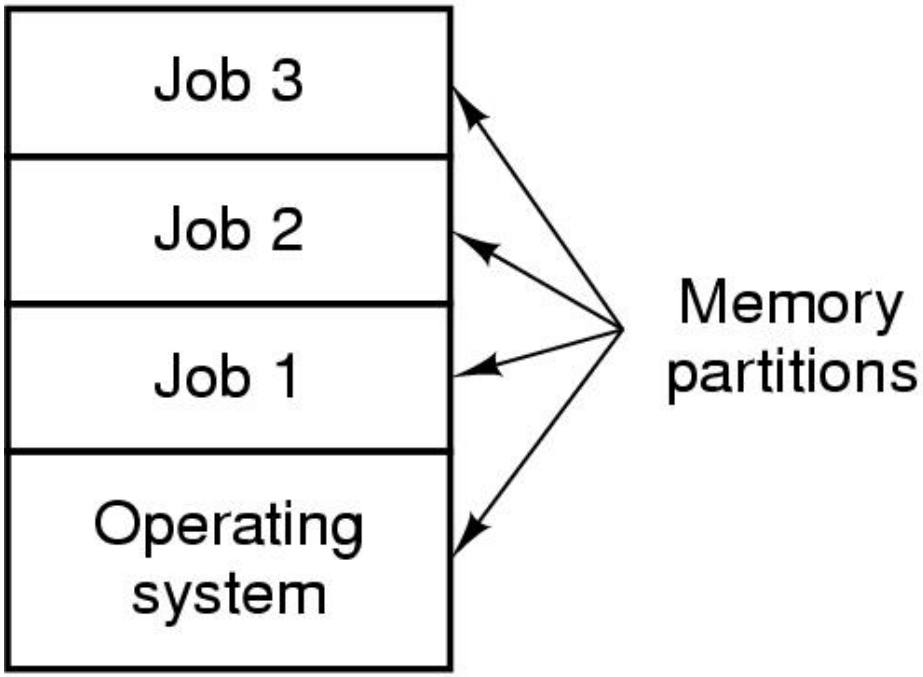
(b) Multiprogramming with two programs

Multiprogramming



(c) Multiprogramming with three programs

Multiprogramming system



Three jobs in memory

Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals

Batch Multiprogramming versus Time Sharing

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

* Revision Ejercicios Clase II
Clase III

Process

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor

Process

- A unit of activity characterized by
  - A single sequential thread of execution
  - A current state
  - An associated set of system resources

Difficulties with Designing System Software

- Improper synchronization
- Failed mutual exclusion
- Nondeterminate program operation
- Deadlocks

Process

- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

Process

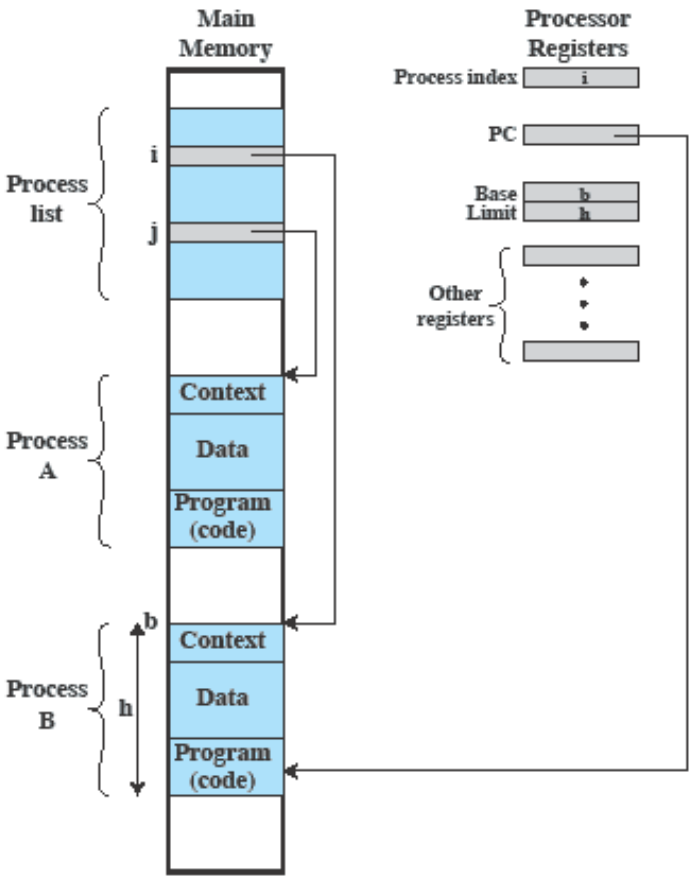


Figure 2.8 Typical Process Implementation

Memory Management

- Process isolation
- Automatic allocation and management
- Support of modular programming
- Protection and access control
- Long-term storage

Virtual Memory

- Implements long-term store
- Information stored in named objects called files
- Allows programmers to address memory from a logical point of view

Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized

Information Protection and Security

- Data integrity
  - Protection of data from unauthorized modification
- Authenticity
  - Concerned with the proper verification of the identity of users and the validity of messages or data

Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - Discriminate among different classes of jobs

Scheduling and Resource Management

- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible

Key Elements of an Operating System

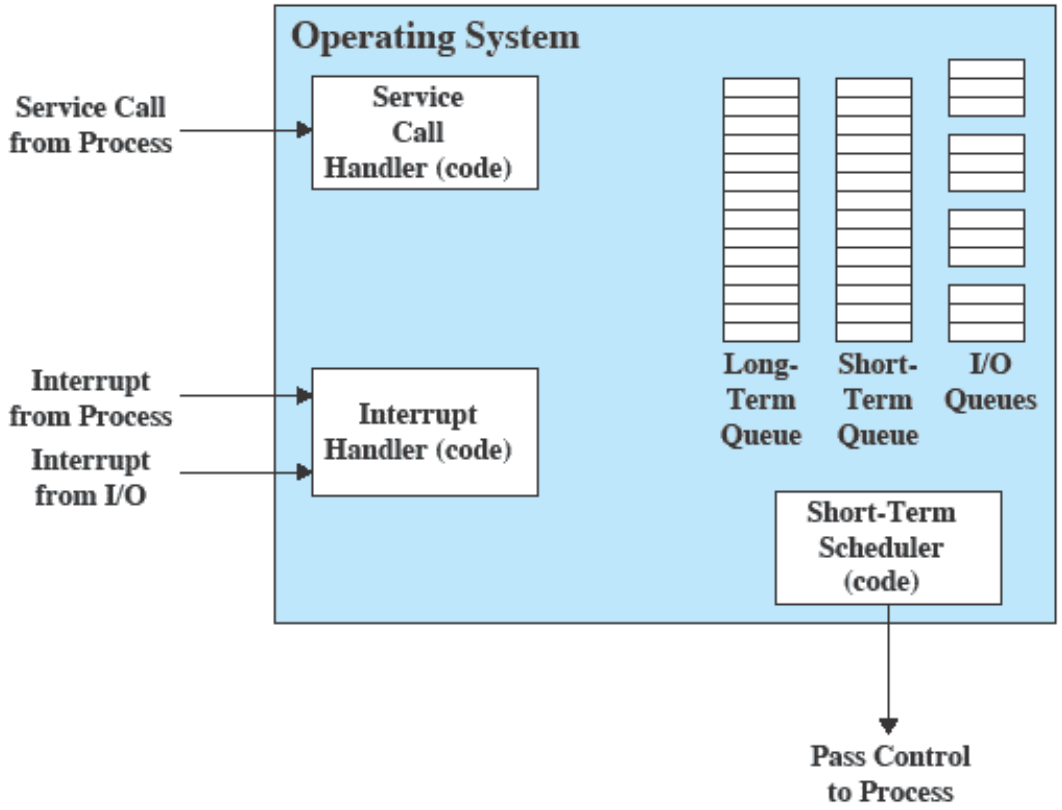


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Modern Operating Systems

- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces
    - Interprocess communication (IPC)
    - Basic scheduling

Modern Operating Systems

- Multithreading
  - Process is divided into threads that can run concurrently
    - Thread
      - Dispatchable unit of work
      - executes sequentially and is interruptable
    - Process is a collection of one or more threads

Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions

Multiprogramming and Multiprocessing

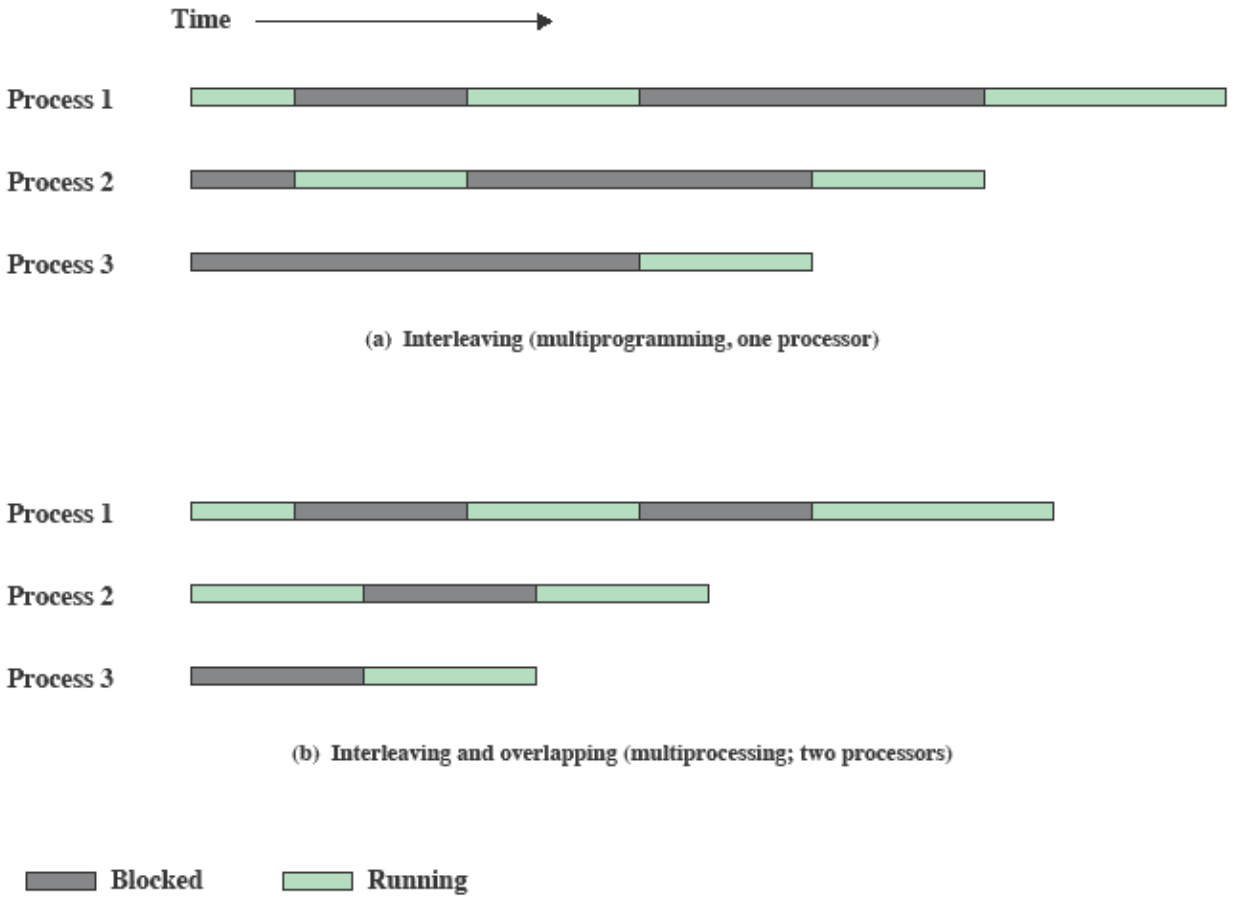


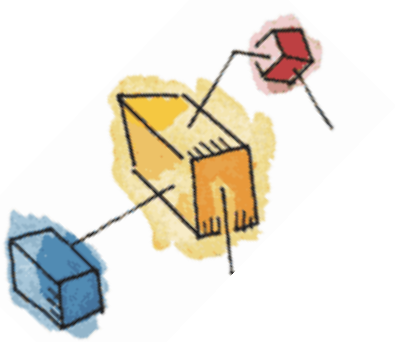
Figure 2.12 Multiprogramming and Multiprocessing

Modern Operating Systems

- Distributed operating systems
  - Provides the illusion of a single main memory space and single secondary memory space

Modern Operating Systems

- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity



UNIX

- Hardware is surrounded by the operating system software
- Comes with a number of user services and interfaces
  - Shell
  - Components of the C compiler

General UNIX Architecture

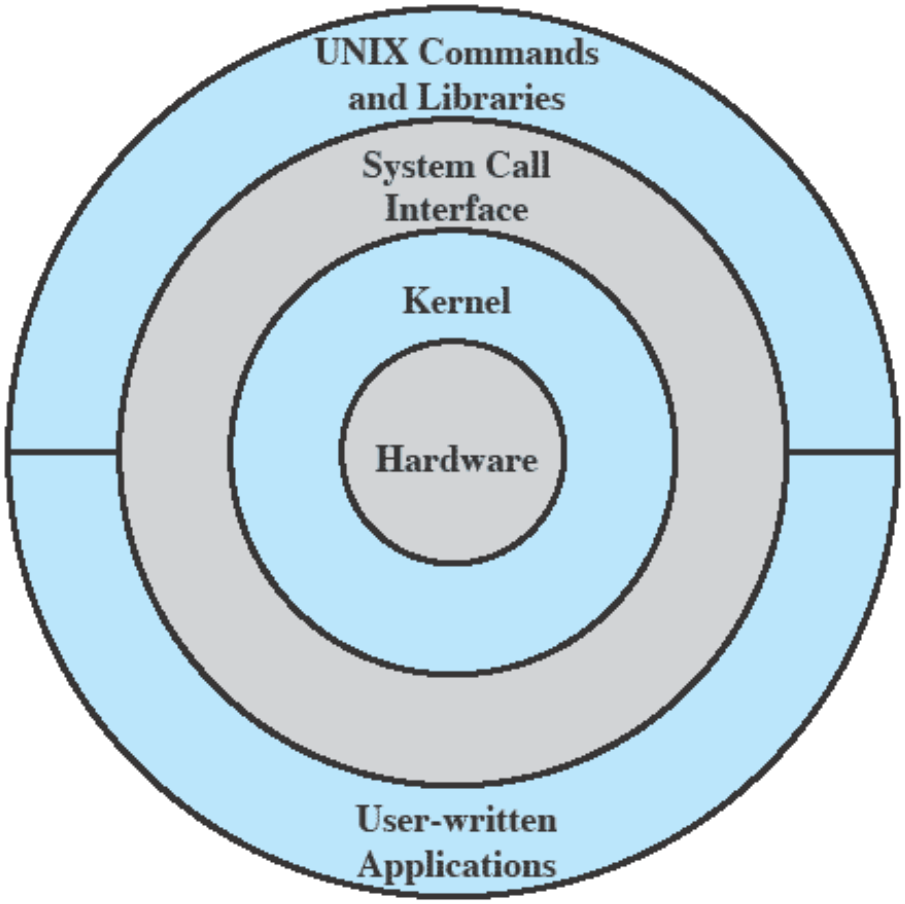


Figure 2.14 General UNIX Architecture

Traditional UNIX Kernel

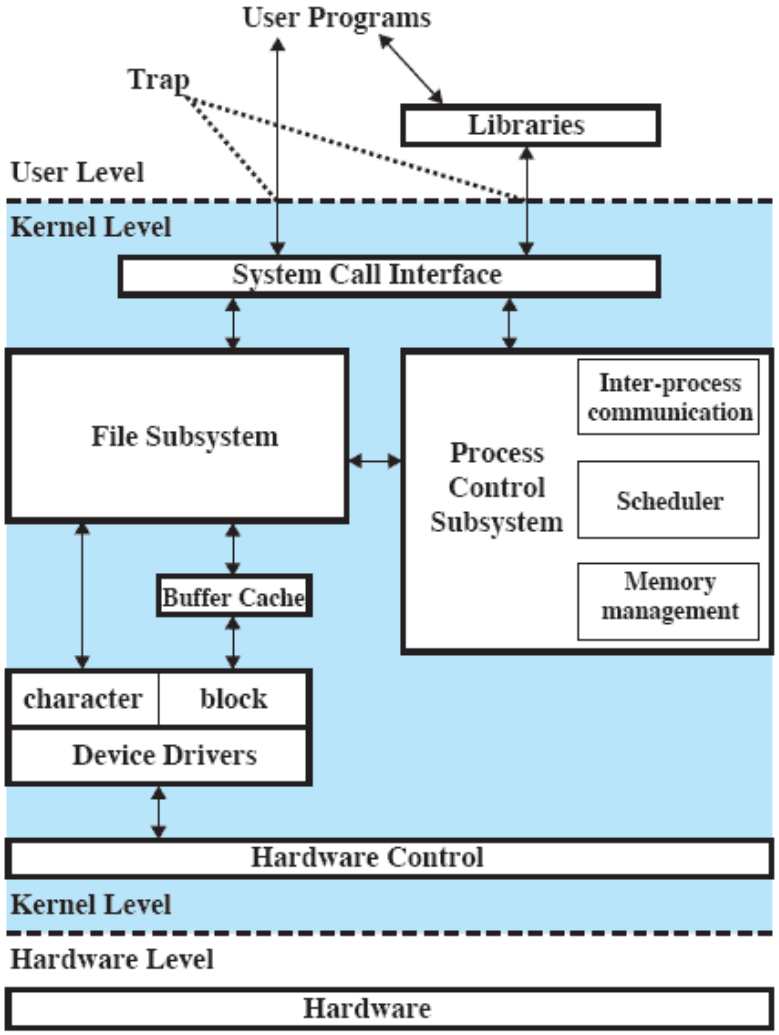
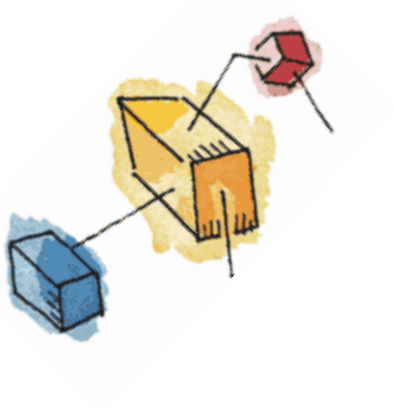


Figure 2.15 Traditional UNIX Kernel



Modern UNIX Kernel

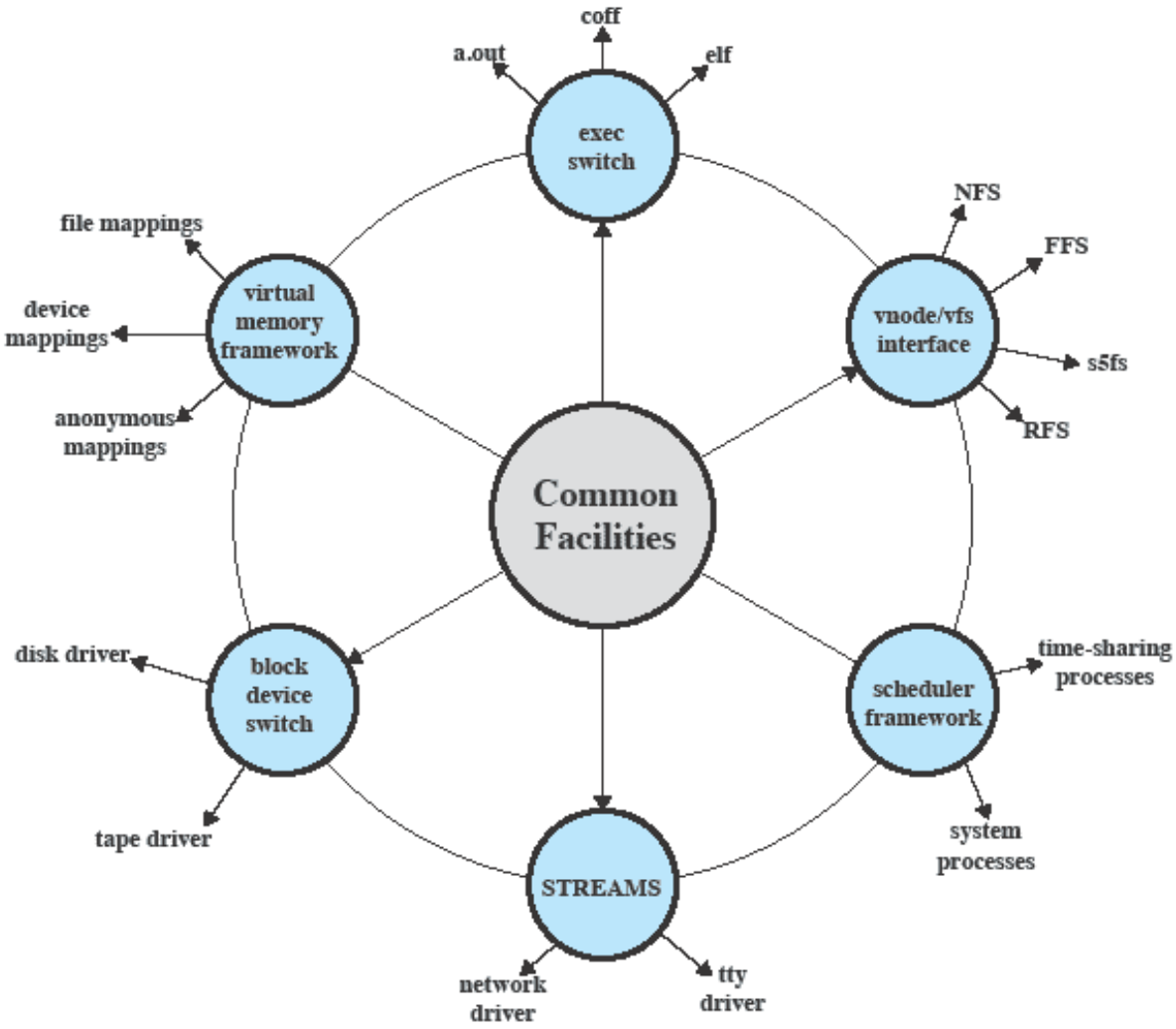
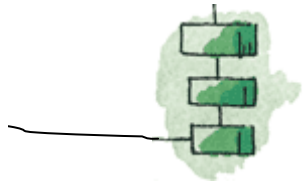
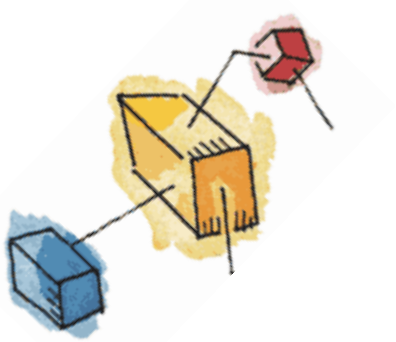


Figure 2.16 Modern UNIX Kernel [VAHA96]



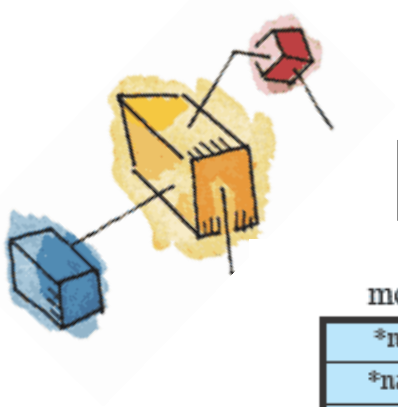
Modern UNIX Systems

- System V Release 4 (SVR4)
- BSD
- Solaris 10



Linux

- Does not use a microkernel approach
- Collection of loadable modules
  - Dynamic linking
  - Stackable modules



Linux Kernel Modules

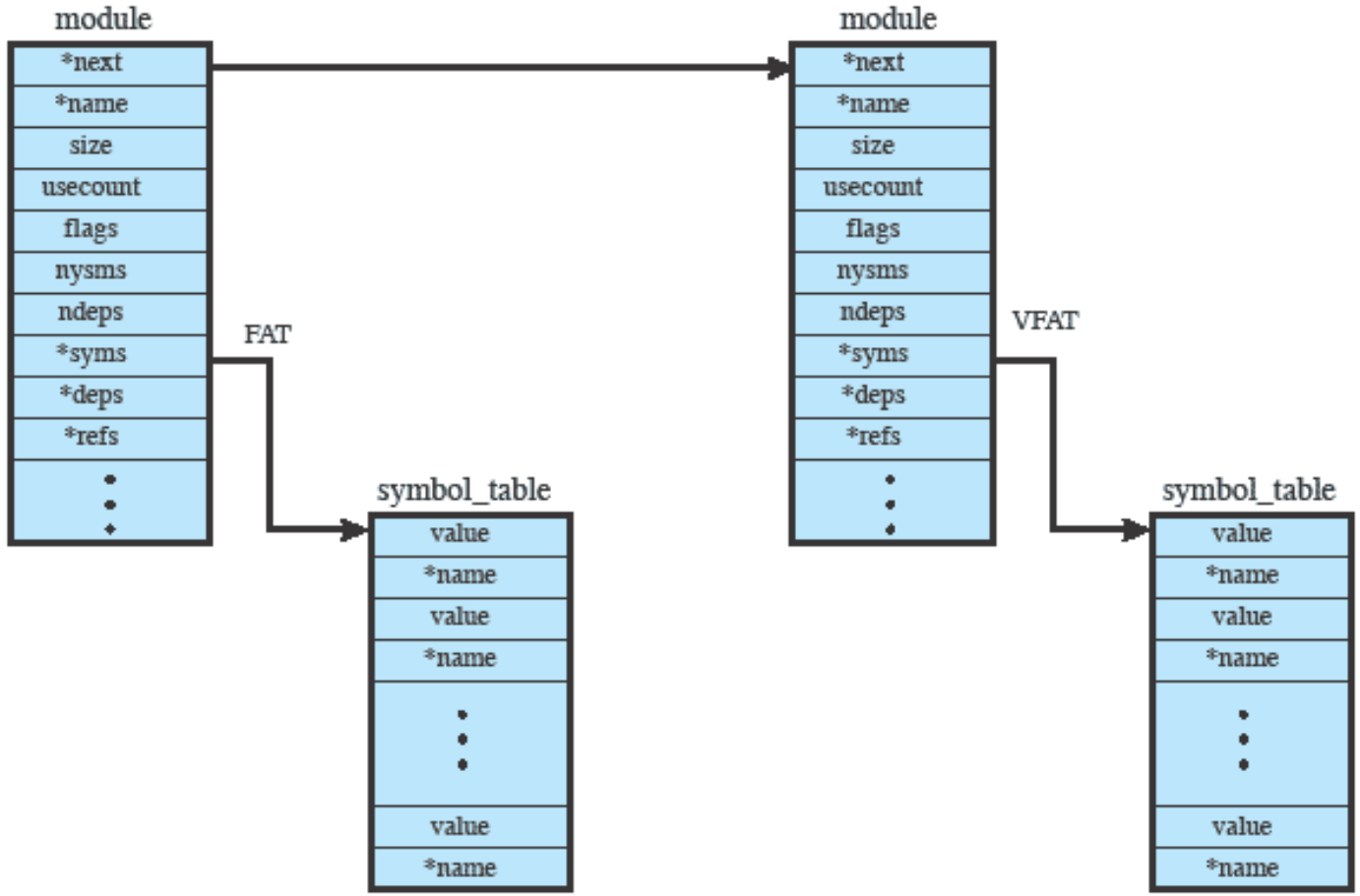


Figure 2.17 Example List of Linux Kernel Modules

Linux Kernel Components

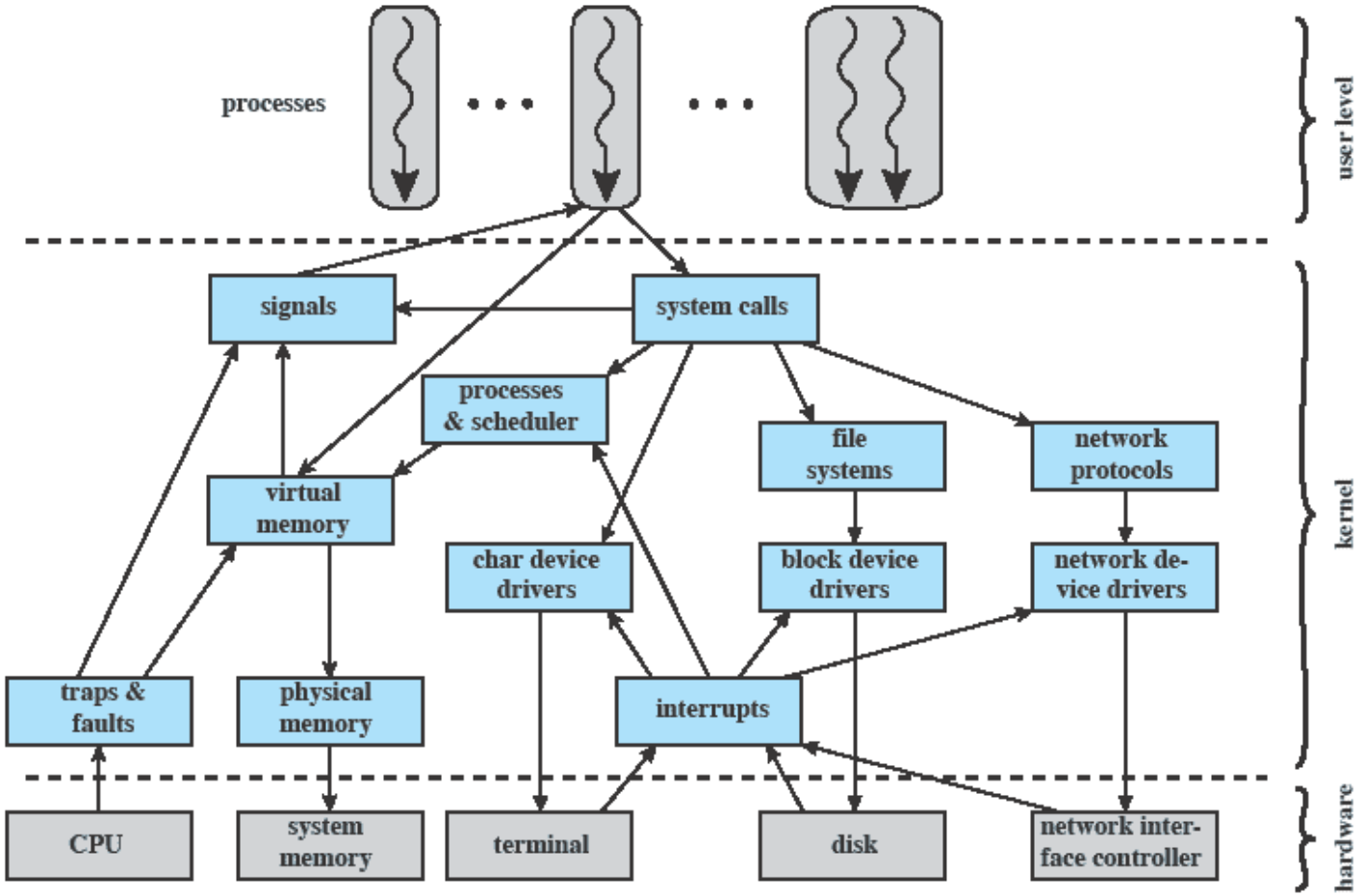


Figure 2.18 Linux Kernel Components