Unit - 1

Computer System Overview

1.Basic Elements

2.Instruction Execution

3.Interrupts

4.Memory Hierarchy

5. Cache Memory

Operating system overview

Objectives and functions

Evolution of Operating System.

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.
- Exploits the hardware resources of one or more processors
- Provides a set of services to system users
- Manages secondary memory and I/O devices

1.Basic Elements

- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques

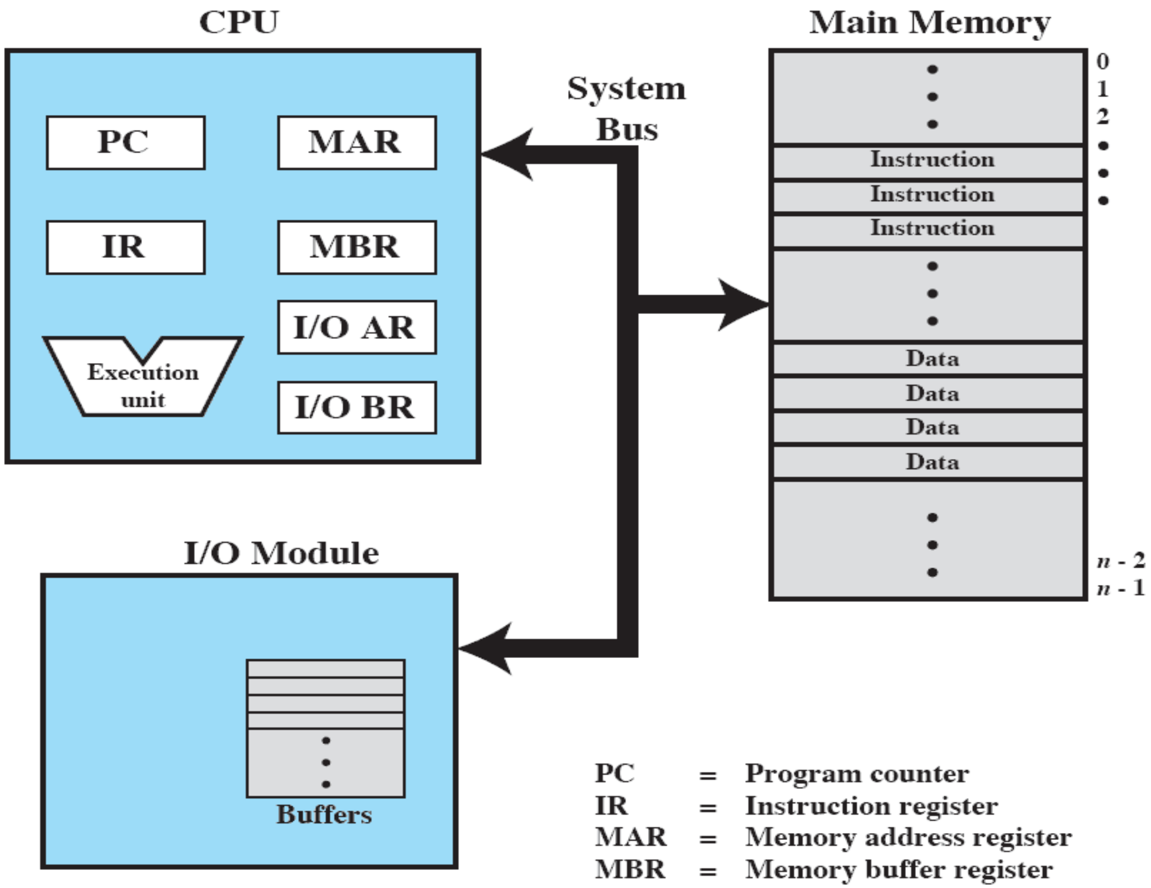
- Processor
- Main Memory
- I/O Modules
- System Bus

- Controls operation, performs data processing
- Two internal registers
  - Memory address resister (MAR)
  - Memory buffer register (MBR)
- I/O address register
- I/O buffer register

- Volatile
    - Data is typically lost when power is removed
- Referred to as real memory or primary memory
- Consists of a set of locations defined by sequentially numbers addresses
    - Containing either data or instructions

- Moves data between the computer and the external environment such as:
  - – Storage (e.g. hard drive)
  - – Communications equipment
  - – Terminals
- Specified by an I/O Address Register
  - – (I/OAR)

- Communication among processors, main memory, and I/O modules



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Figure 1.1 Computer Components: Top-Level View

- Faster and smaller than main memory
- 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

- May be referenced by machine language
  - Available to all programs – application programs and system programs
- Types of registers typically available are:
  - data,
  - address,
  - condition code 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..contains conditional codes

- Data
  - Often general purpose
  - But some restrictions may apply
- Address
  - Index Register:- adding a index to a base value to get EA
  - Segment pointer-Memory is divided into segments, reference to a particular segment
  - Stack pointer-register that points to the top of stack.

- Usually part of the control register
  - Also called *flags*
- Bits set by processor hardware as a result of operations
  - Read only, intended for feedback regarding the results of instruction execution.

2.Instruction Execution

- • A program consists of a set of instructions stored in memory
- • Two steps –
- Processor reads (fetches) instructions from memory –
- Processor executes each instruction

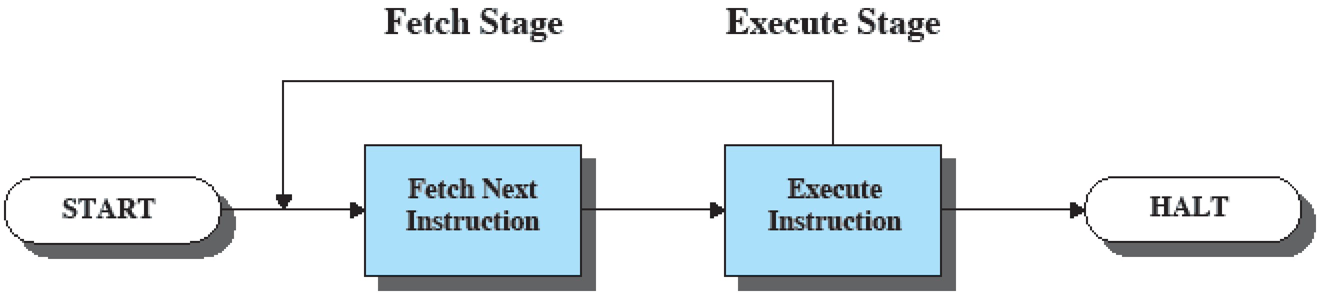
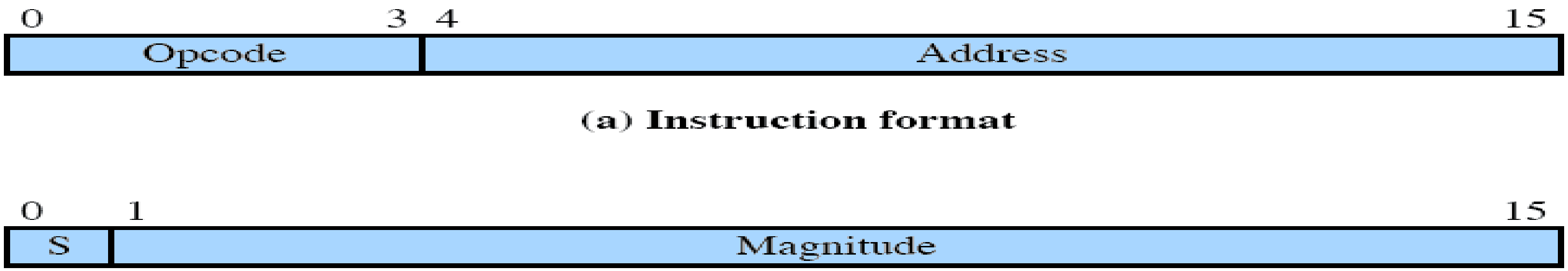


Figure 1.2 Basic Instruction Cycle

- 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

- Fetched instruction loaded into instruction register
- Categories
  - Processor-memory, -data from processor to memory
  - processor-I/O,
  - Data processing,
  - Control



(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

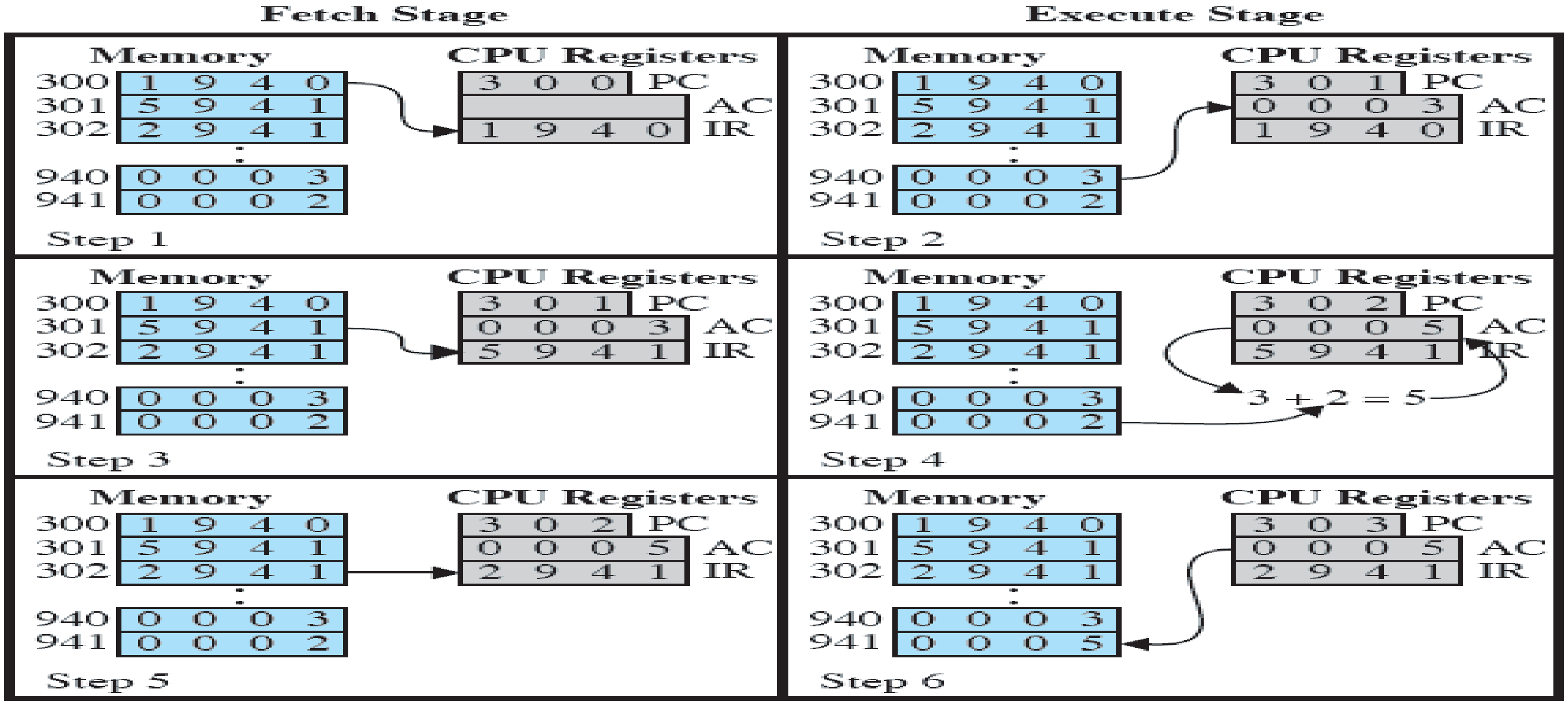


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

- Io module
- Can exchange data directly with the processor
- DMA-Exchange data without going to the processor.

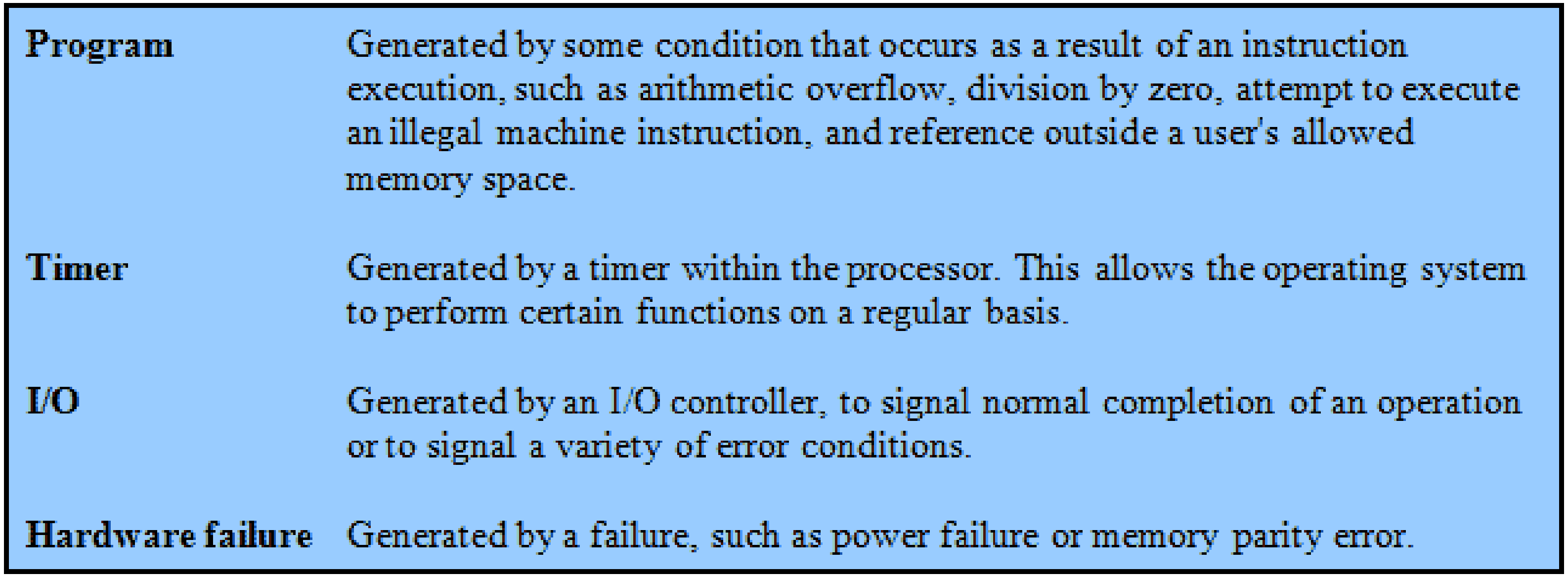
3.Interrupts

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

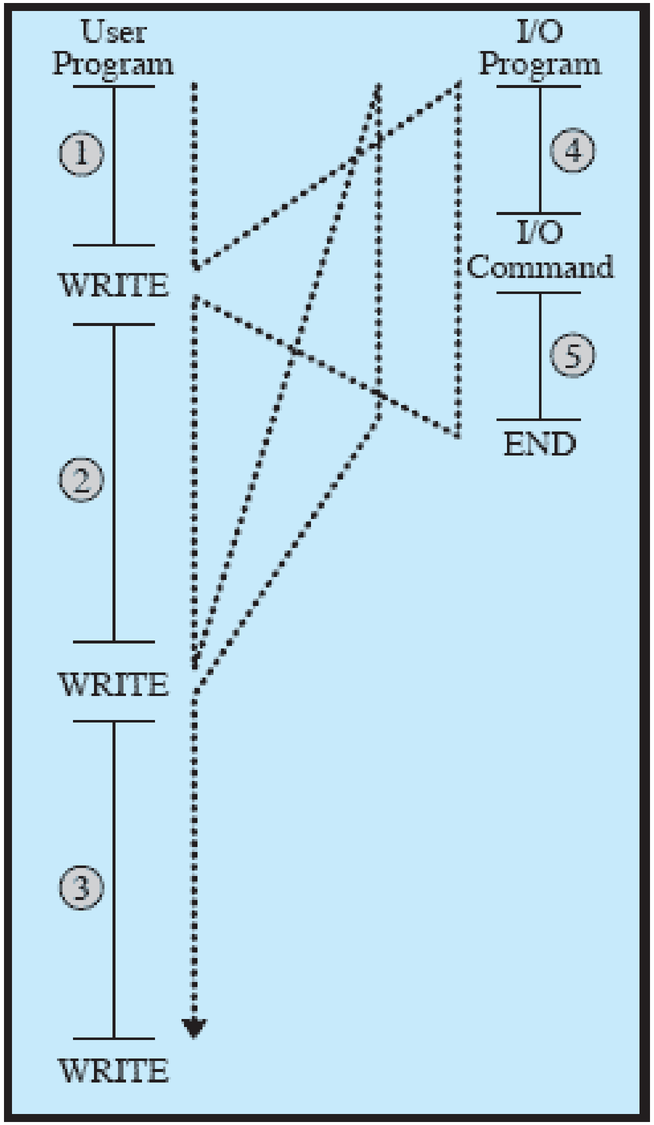
Interrupts

- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
    - most I/O devices are slower than the processor
    - processor must pause to wait for device
    - wasteful use of the processor

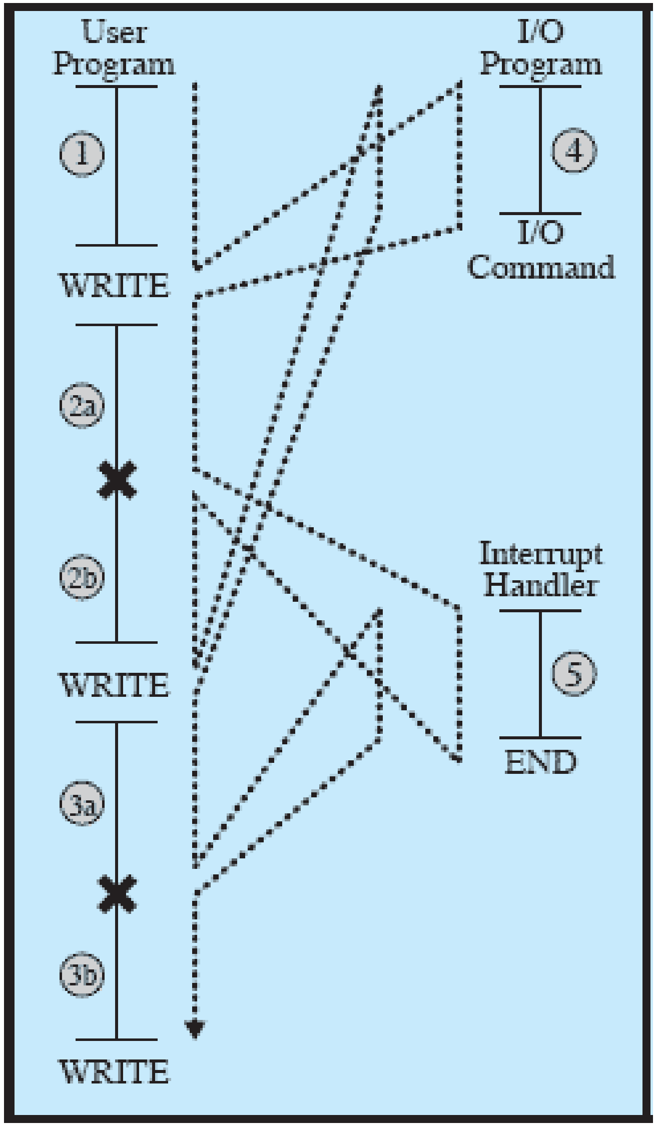
Table 1.1 Classes of Interrupts



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.



(a) No interrupts



(b) Interrupts; short I/O wait

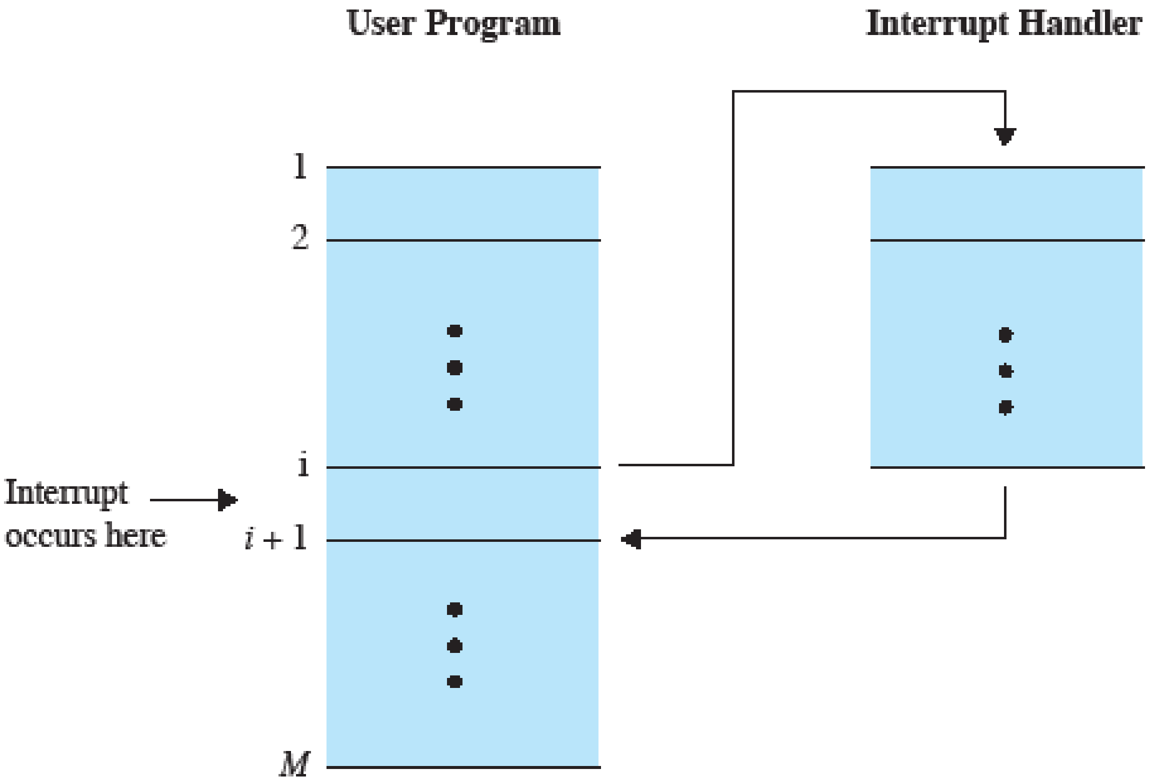
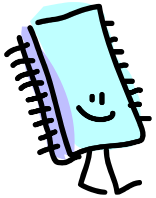


Figure 1.6 Transfer of Control via Interrupts



Instruction Cycle With Interrupts

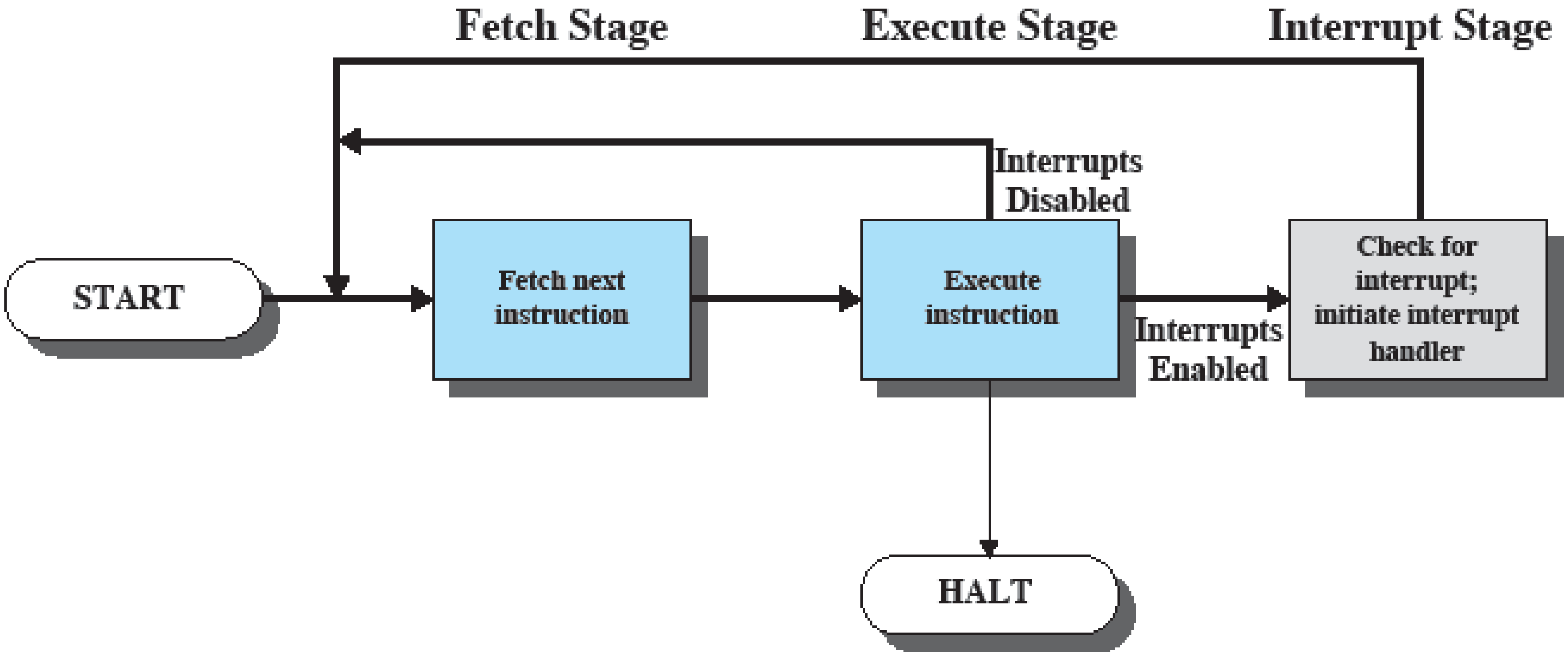
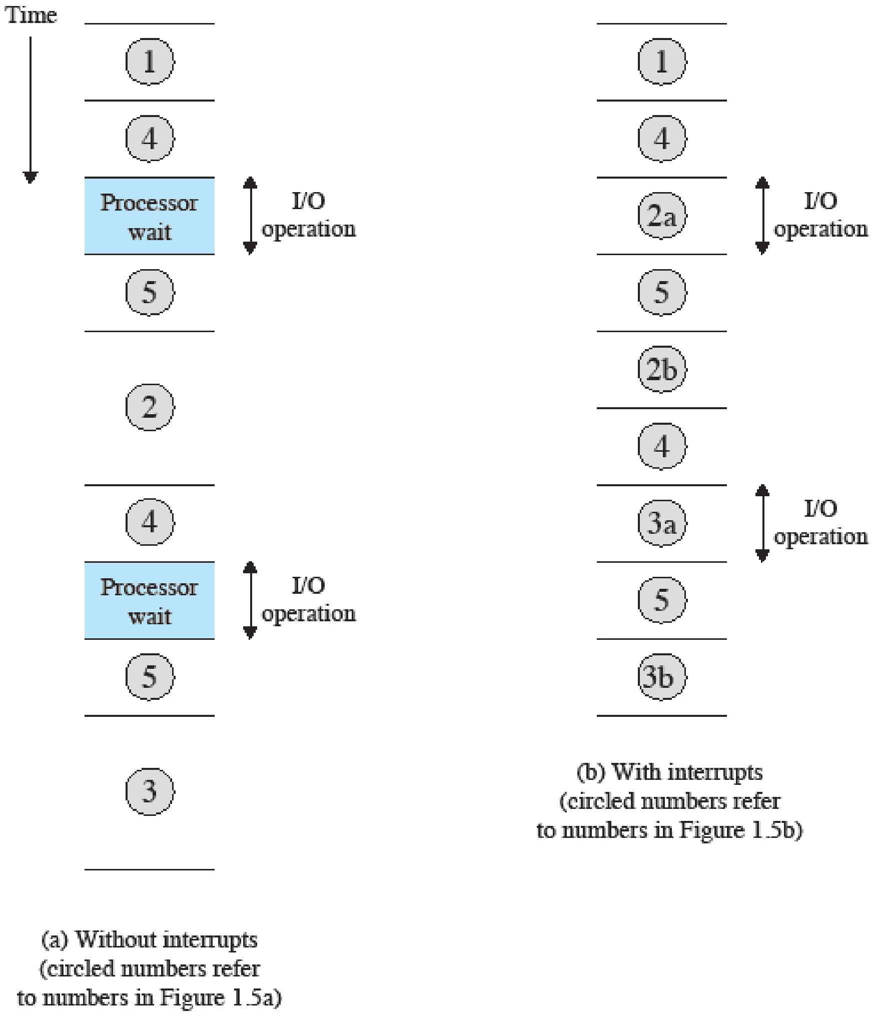


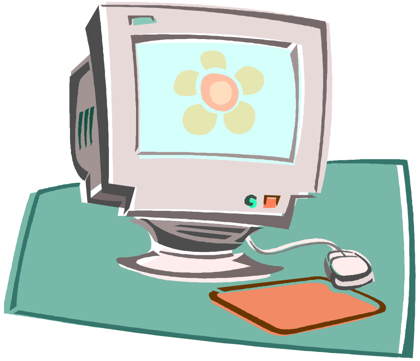
Figure 1.7 Instruction Cycle with Interrupts

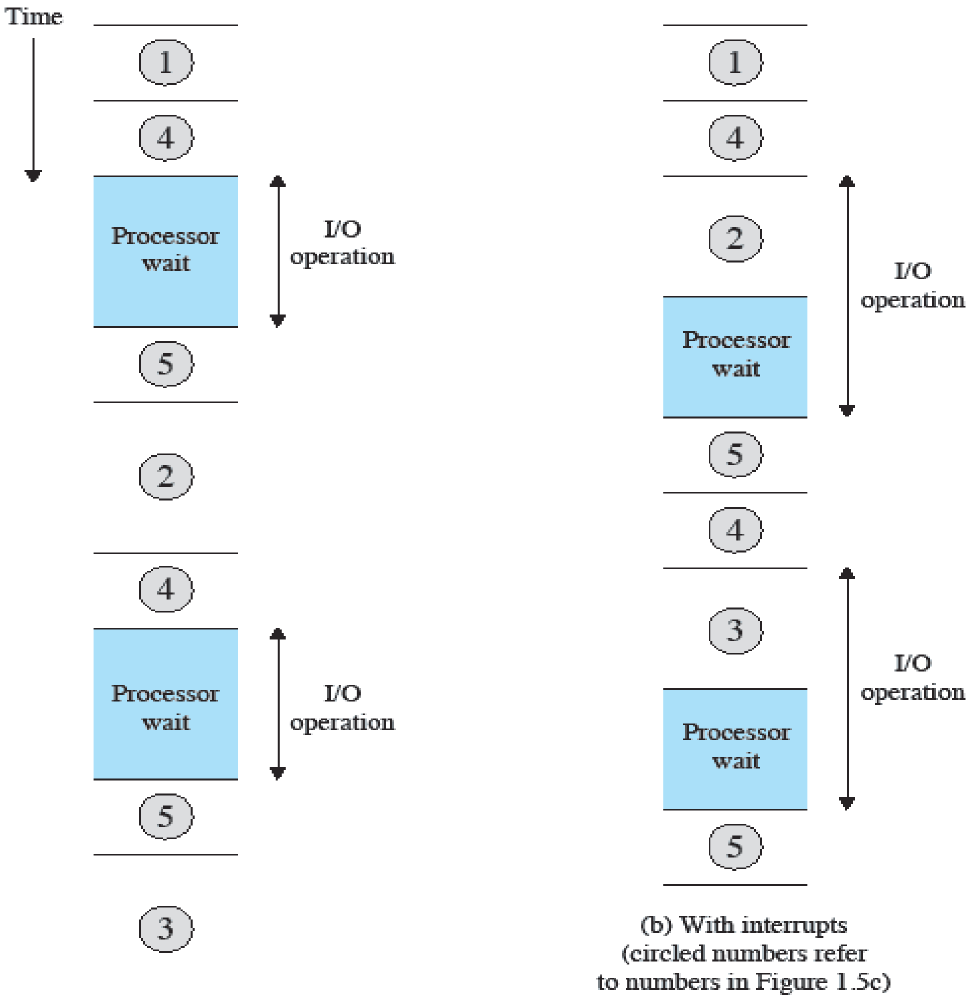


(a) Without interrupts
(circled numbers refer
to numbers in Figure 1.5a)

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

Figure 1.8 Program Timing: Short I/O Wait

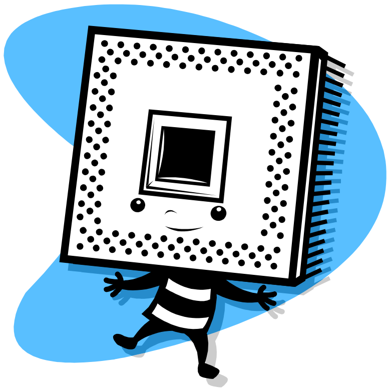


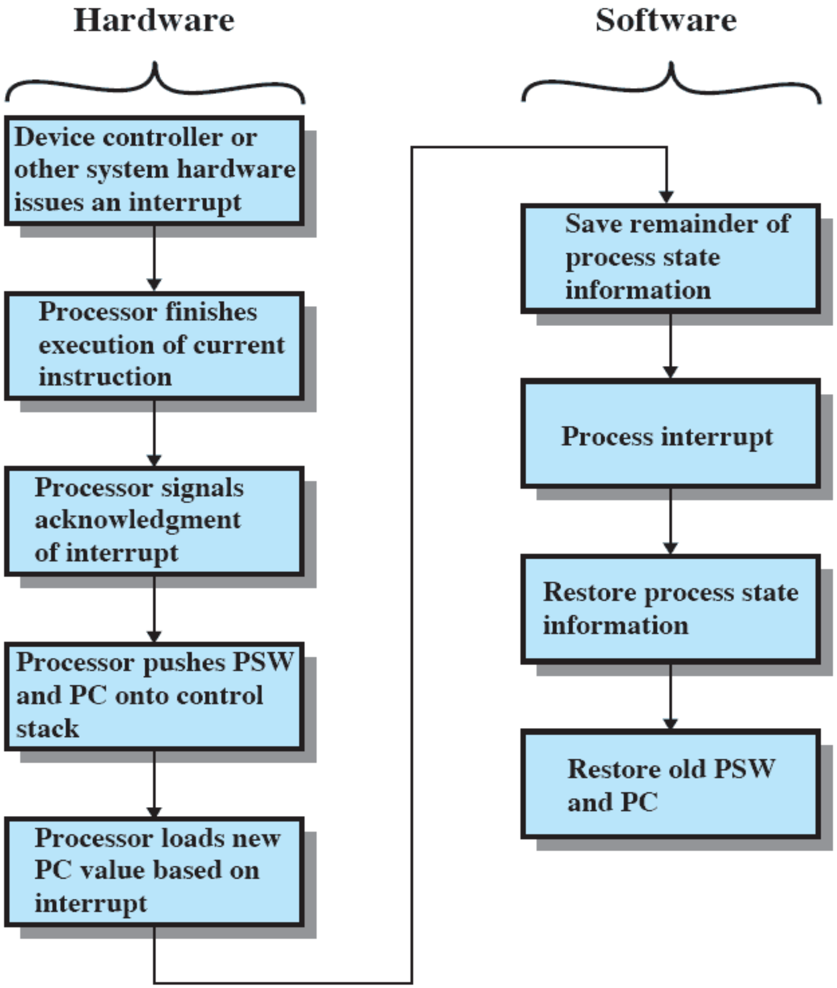


(a) Without interrupts
(circled numbers refer
to numbers in Figure 1.5a)

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

Figure 1.9 Program Timing: Long I/O Wait





Hardware

Software

Device controller or other system hardware issues an interrupt

Processor finishes execution of current instruction

Processor signals acknowledgment of interrupt

Processor pushes PSW and PC onto control stack

Processor loads new PC value based on interrupt

Save remainder of process state information

Process interrupt

Restore process state information

Restore old PSW and PC

Figure 1.10 Simple Interrupt Processing

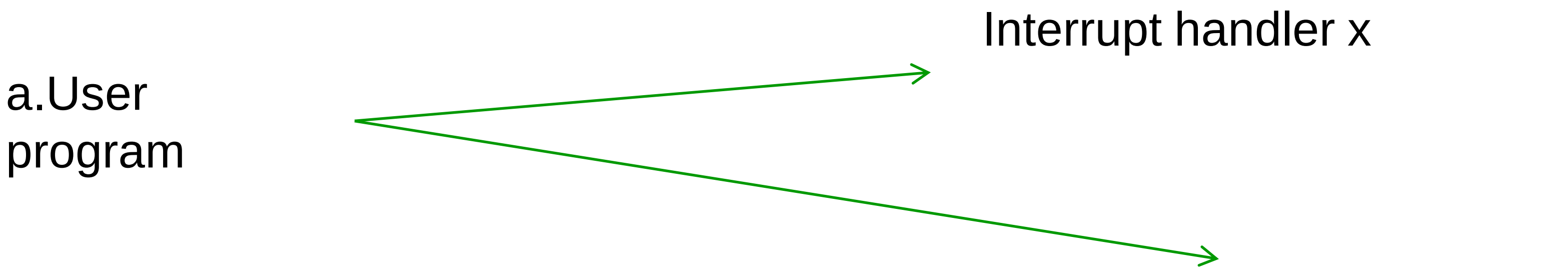
Multiple Interrupts

An interrupt occurs while another interrupt is being processed

- e.g. receiving data from a communications line and printing results at the same time

Two approaches:

- disable interrupts while an interrupt is being processed
- use a priority scheme

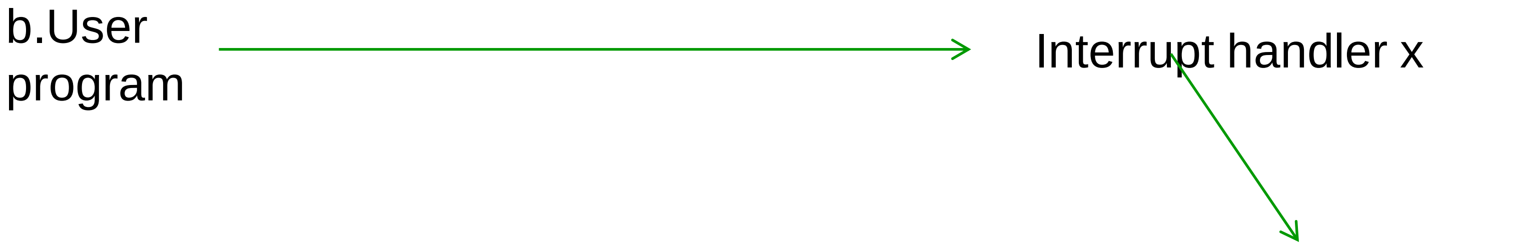


a.User program

Interrupt handler x

Interrupt handler y

a.Sequential interrupt processing



b.User program

Interrupt handler x

Interrupt handler y

b.Nested interrupts

- Processor has more than one program to execute
- The sequence the 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

4.Memory Hierarchy

- Three characteristics of memory
- Cost
- Capacity
- Access time

- Major constraints in memory
  - Amount
  - Speed
  - Expense
- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

- Going down the hierarchy
  - – Decreasing cost per bit
  - – Increasing capacity
  - – Increasing access time
  - – Decreasing frequency of access to the memory by the processor

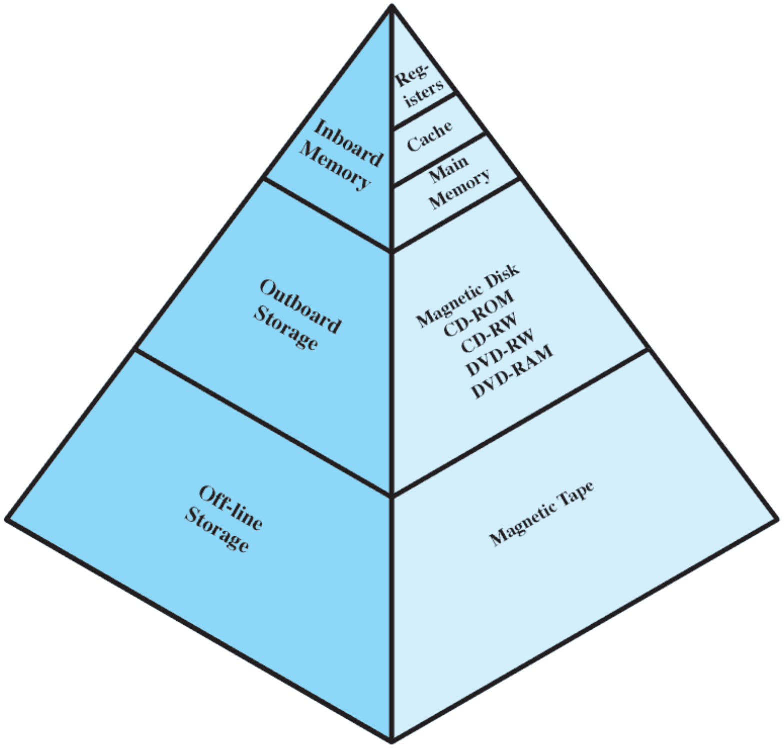
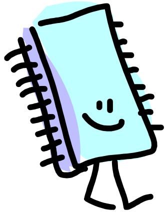


Figure 1.14 **The Memory Hierarchy**

- H is defined as the fraction of all memory accesses that are found in the faster memory.(cache)

- Principle (Locality of reference)
- Decreasing the frequency of access of the memory by the processor.

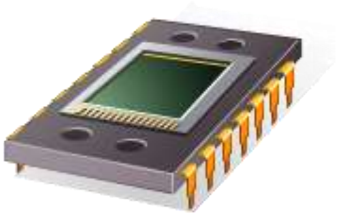
Secondary Memory



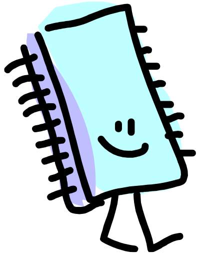
Also referred to as auxiliary memory

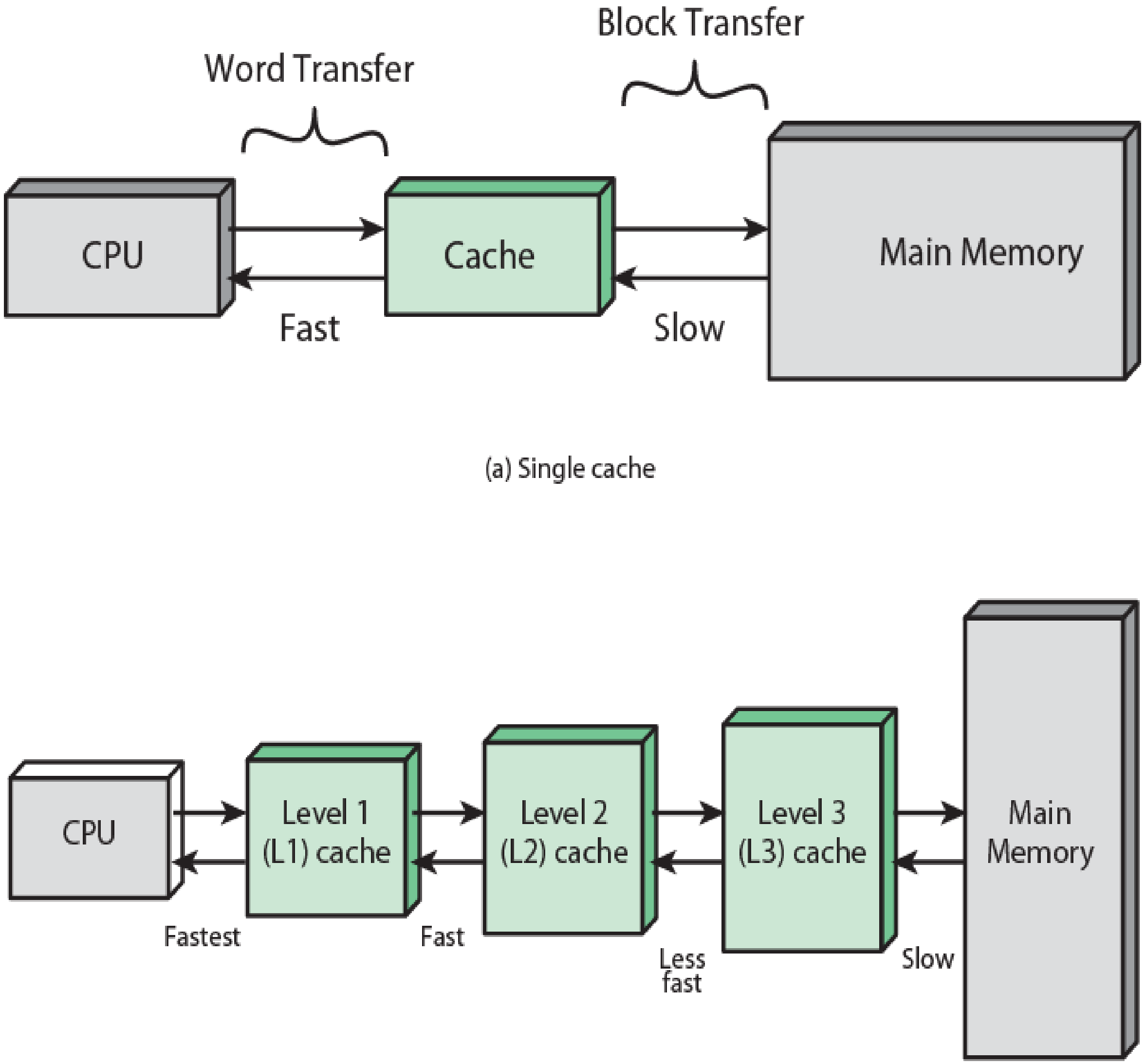
- External
- Nonvolatile
- Used to store program and data files

5.Cache Memory



Cache Principles





(a) Single cache

(b) Three-level cache organization

Cache/Main-Memory Structure

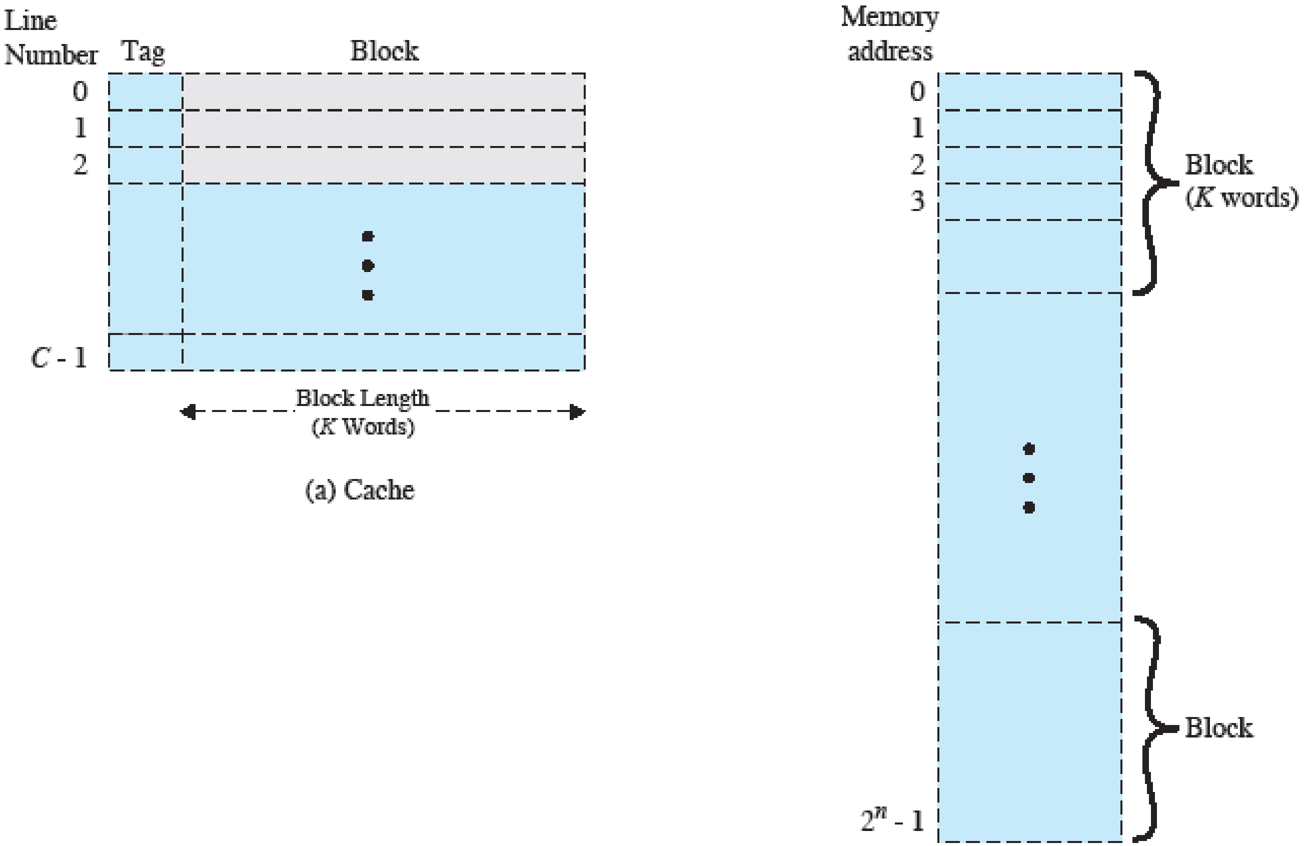
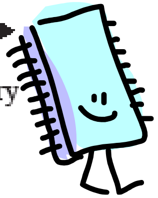


Figure 1.17 Cache/Main-Memory Structure



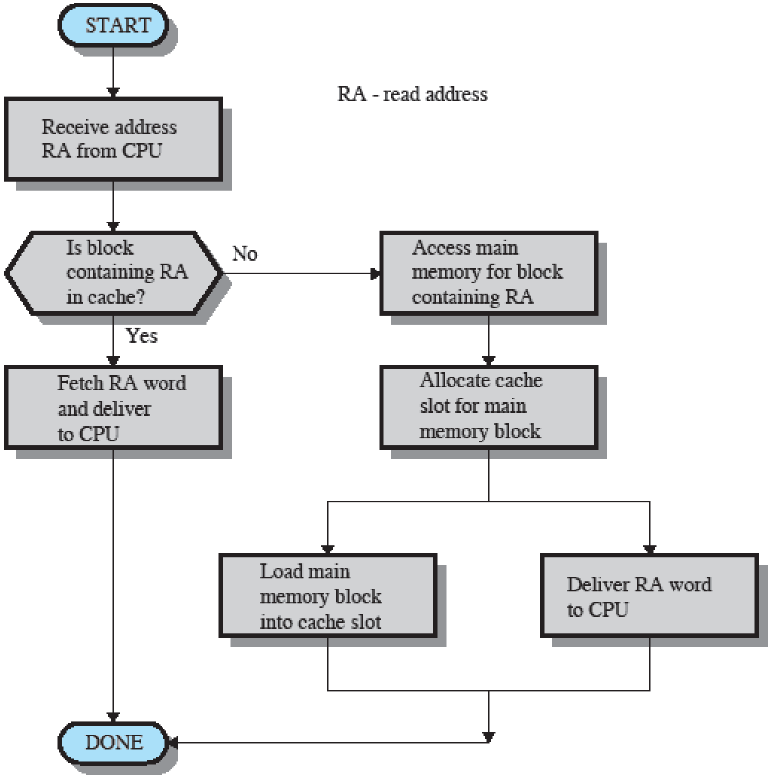

START
RA - read address
Receive address RA from CPU
Is block containing RA in cache?
No
Access main memory for block containing RA
Yes
Fetch RA word and deliver to CPU
Allocate cache slot for main memory block
Load main memory block into cache slot
Deliver RA word to CPU
DONE

Figure 1.18 Cache Read Operation

- Main categories are:
  - Cache size
  - Block size
  - Mapping function
  - Replacement algorithm
  - Write policy

- Cache size
  - Small caches have significant impact on performance
- Block size
  - The unit of data exchanged between cache and main memory
  - Larger block size means more hits
  - But too large reduces chance of reuse.

- Determines which cache location the block will occupy
- Two constraints:
  - When one block read in, another may need replaced
  - Complexity of mapping function increases circuitry costs for searching.

- Chooses which block to replace when a new block is to be loaded into the cache.
- Ideally replacing a block that isn’t likely to be needed again
  - Impossible to guarantee
- Effective strategy is to replace a block that has been used less than others
  - Least Recently Used (LRU)

- Dictates when the memory write operation takes place
- Can occur every time the block is updated
- Can occur when the block is replaced
  - Minimize write operations
  - Leave main memory in an obsolete state

Operating System

Main objectives of an OS:

- Convenience
- Efficiency
- Ability to evolve

The OS as a User/Computer Interface

- Computer Hardware-Software Structure
  - Layered organization
- OS services to users

Computer Hardware and Software Infrastructure

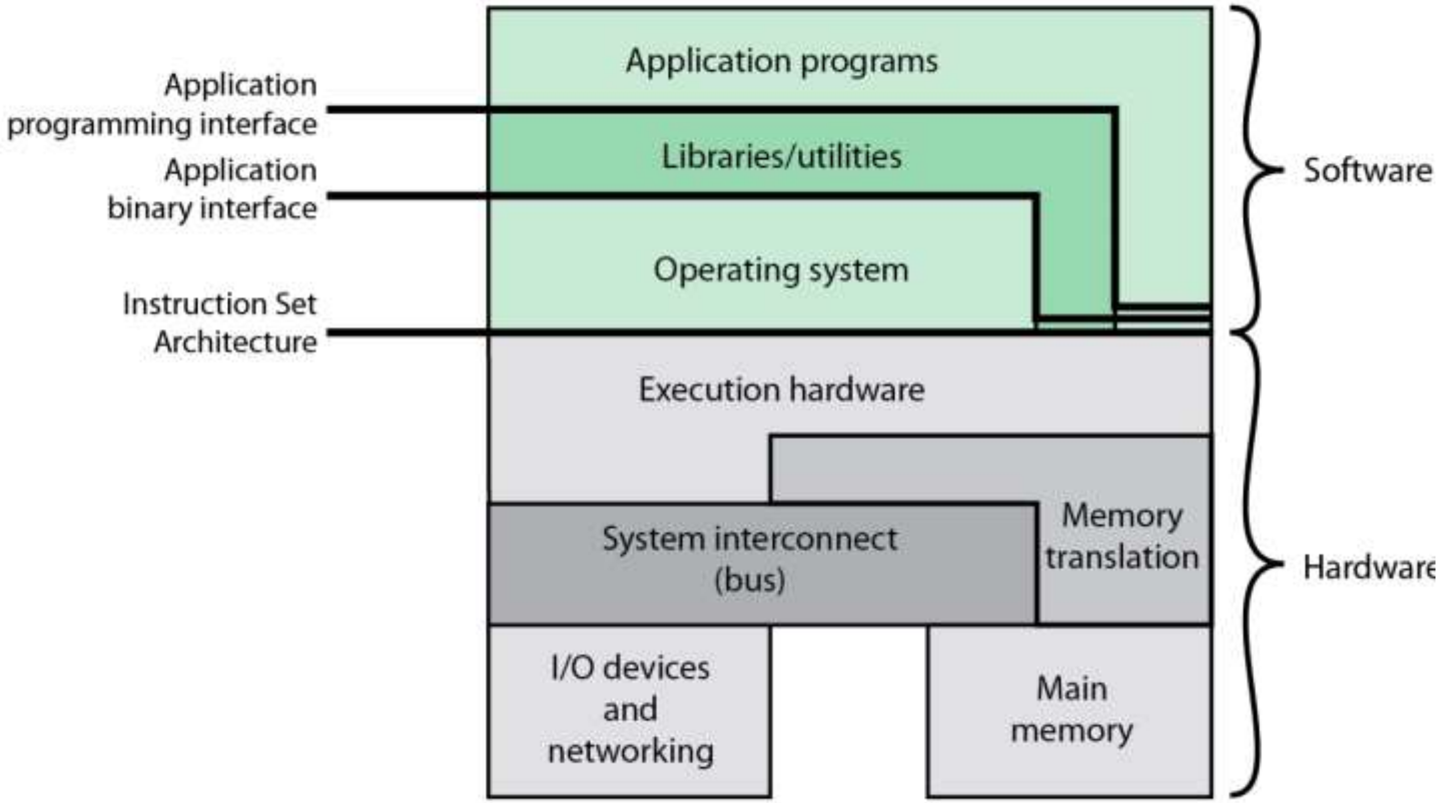

Application programming interface
Application binary interface
Instruction Set Architecture
Application programs
Libraries/utilities
Operating system
Execution hardware
Memory translation
System interconnect (bus)
I/O devices and networking
Main memory
Software
Hardware

Figure 2.1 Computer Hardware and Software Infrastructure

Operating System Services

Key Interfaces

- Instruction set architecture (ISA)
- Application binary interface (ABI)
- Application programming interface (API)

The Operating System as a Resource Manager

- A computer is a set of resources for moving, storing, & processing data
- The OS is responsible for managing these resources
- The OS exercises its control through software

Operating System as Software

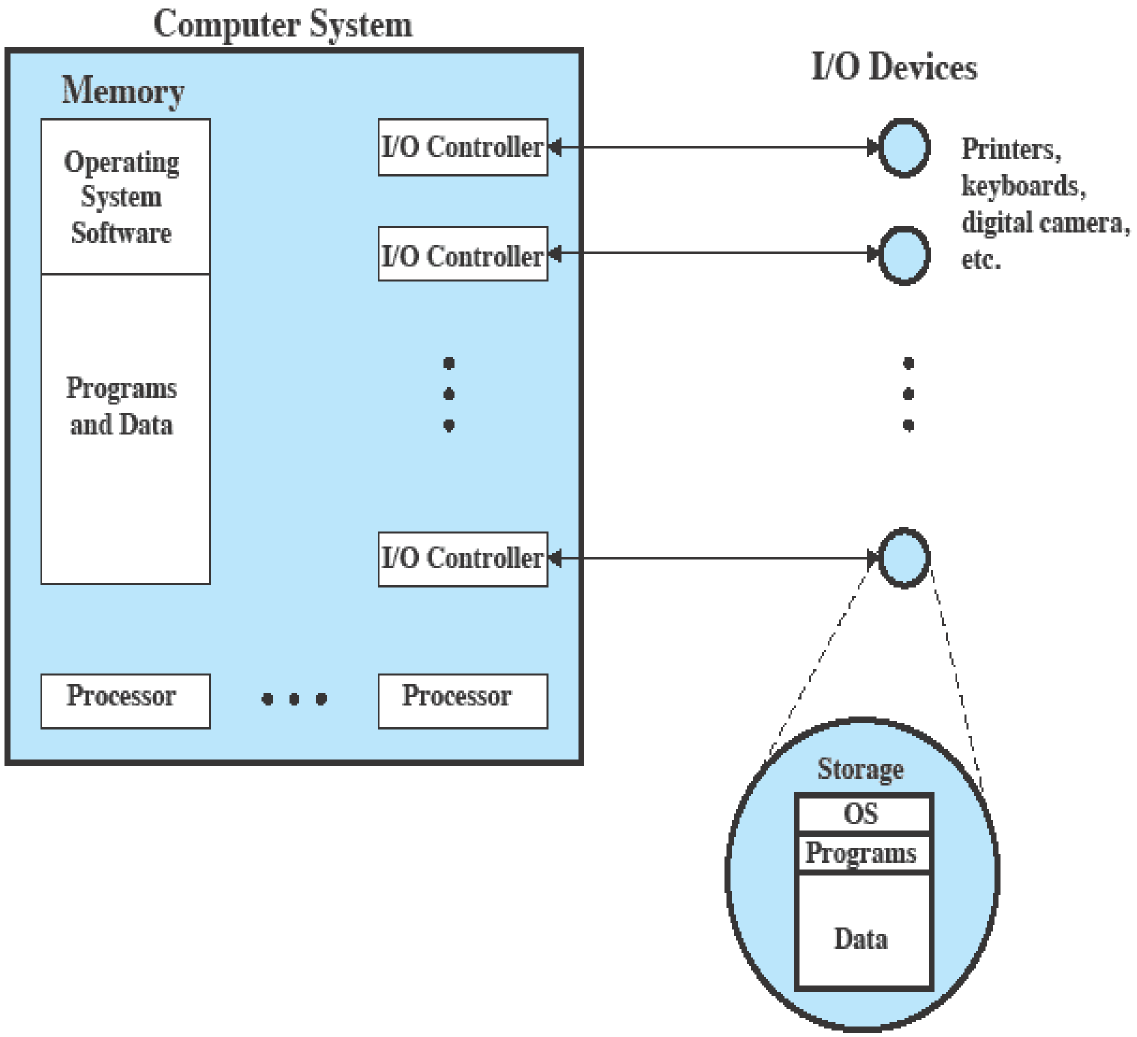


Figure 2.2 The Operating System as Resource Manager

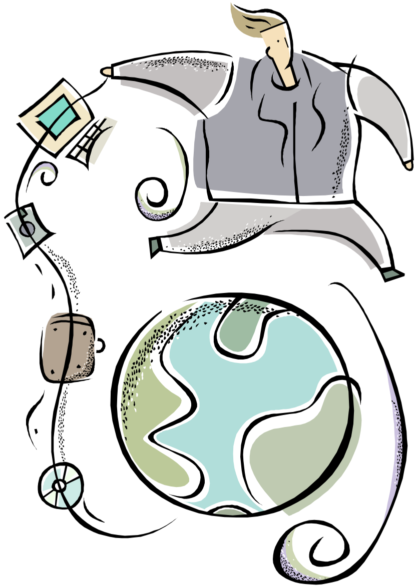
Evolution of Operating Systems

Hardware upgrades

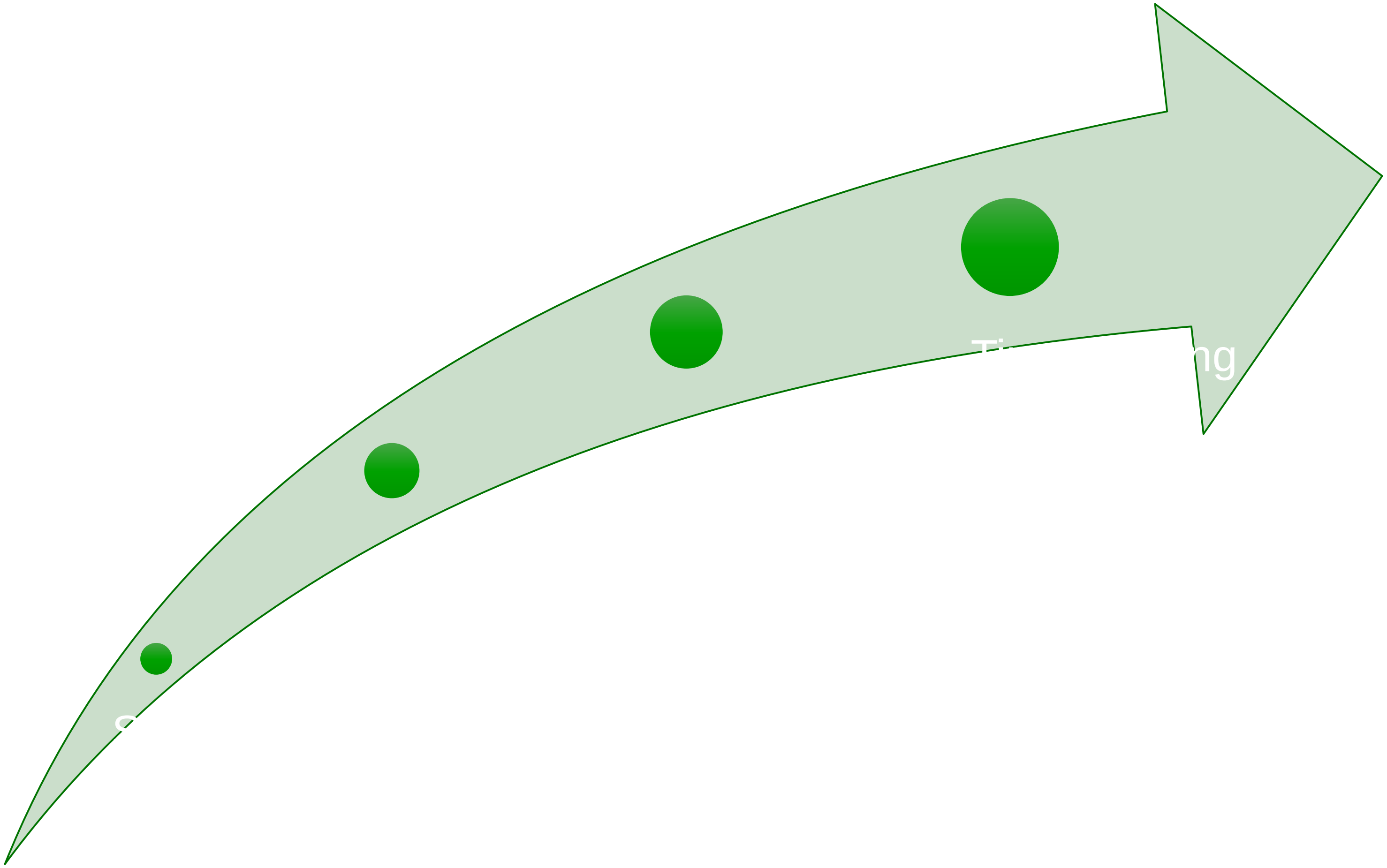
New types of hardware

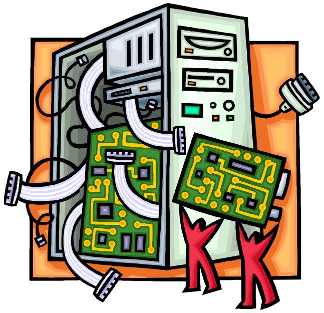
New services

Fixes



Evolution of Operating Systems





Processing

Earliest Computers:

- No operating system
  - programmers interacted directly with the computer hardware
- Computers ran from a console with display lights, toggle switches, some form of input device, and

Problems:

- Scheduling:
  - most installations used a hardcopy sign-up sheet to reserve computer time
    - time allocations could run short or long, resulting in wasted computer time
  - Setup time

Simple Batch Systems

Monitor Point of View

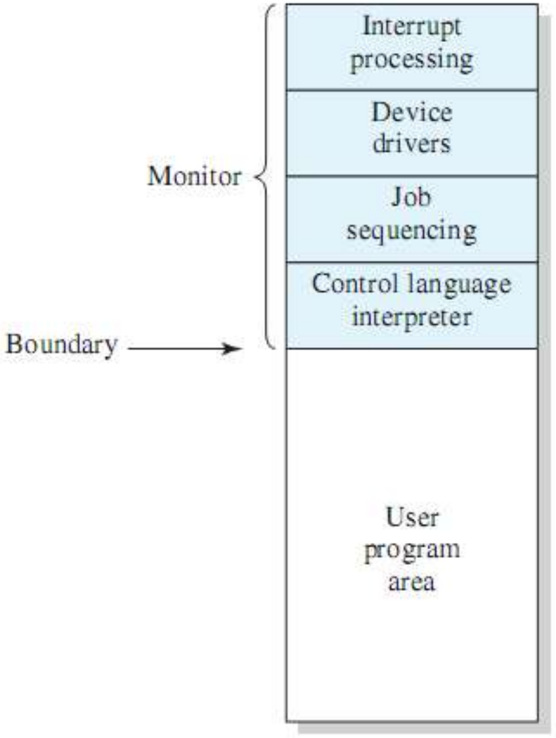
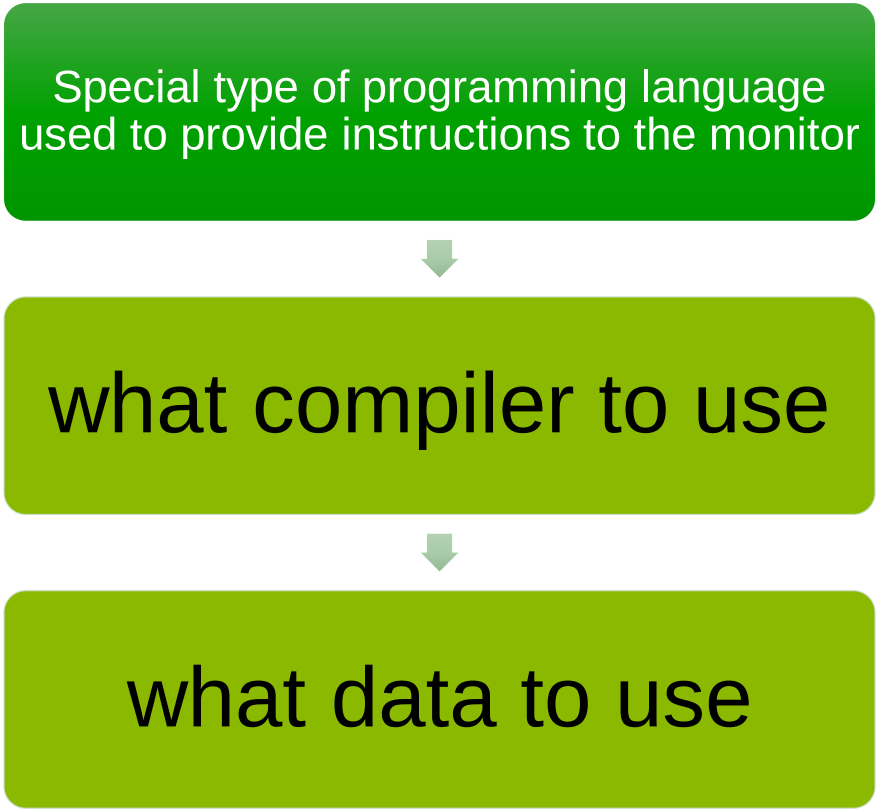


Figure 2.3 **Memory Layout for a Resident Monitor**

Processor Point of View

- Processor executes instruction from the memory containing the monitor
- Executes the instructions in the user program until it encounters an ending or error condition
- “*control is passed to a job*” means processor is fetching and executing instructions in a user program
- “*control is returned to the monitor*” means that the processor is fetching and executing instructions from the monitor program

Job Control Language (JCL)



Special type of programming language used to provide instructions to the monitor

what compiler to use

what data to use

Desirable Hardware Features

Memory protection for monitor

- while the user program is executing, it must not alter the memory area containing the monitor

Timer

- prevents a job from monopolizing the system

Privileged instructions

- can only be executed by the monitor

Interrupts

- gives OS more flexibility in controlling user programs

Modes of Operation

User Mode

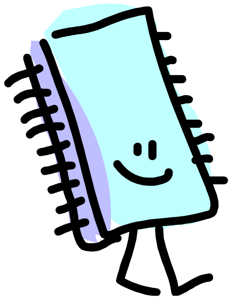
- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

Kernel Mode

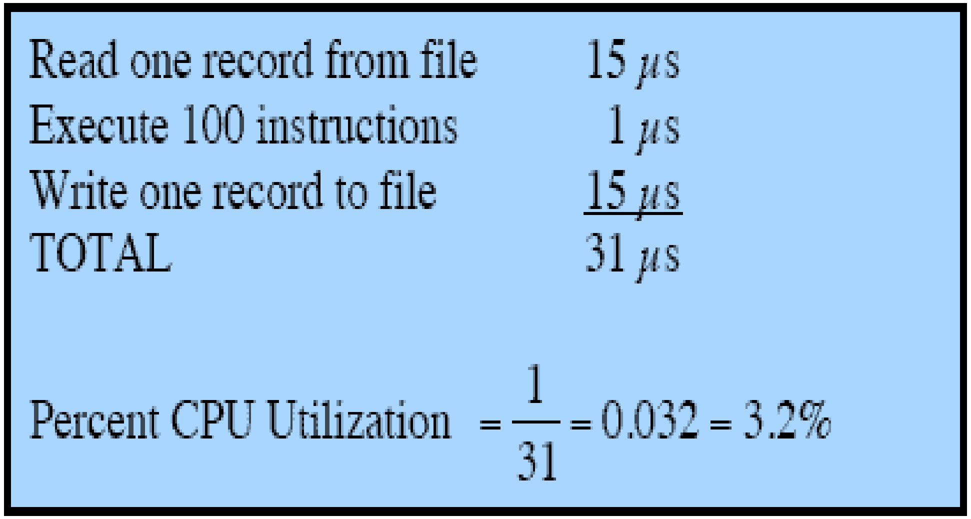
- monitor executes in kernel mode
- privileged instructions may be executed
- protected areas of memory may be accessed

Simple Batch System Overhead

- Processor time alternates between execution of user programs and execution of the monitor
- Sacrifices:
  - some main memory is now given over to the monitor
  - some processor time is consumed by the monitor
- Despite overhead, the simple batch system improves utilization of the computer. (How?)



Multiprogrammed Batch Systems

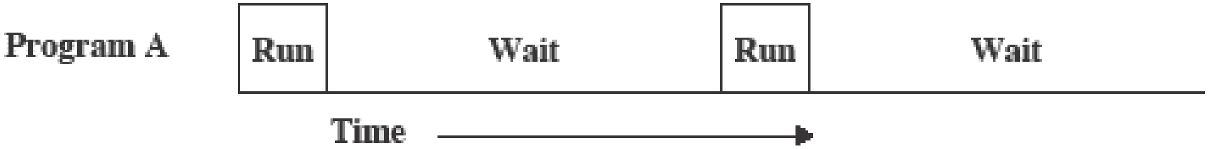


Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	$\underline{15\ \mu\text{s}}$
TOTAL	31 μs

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

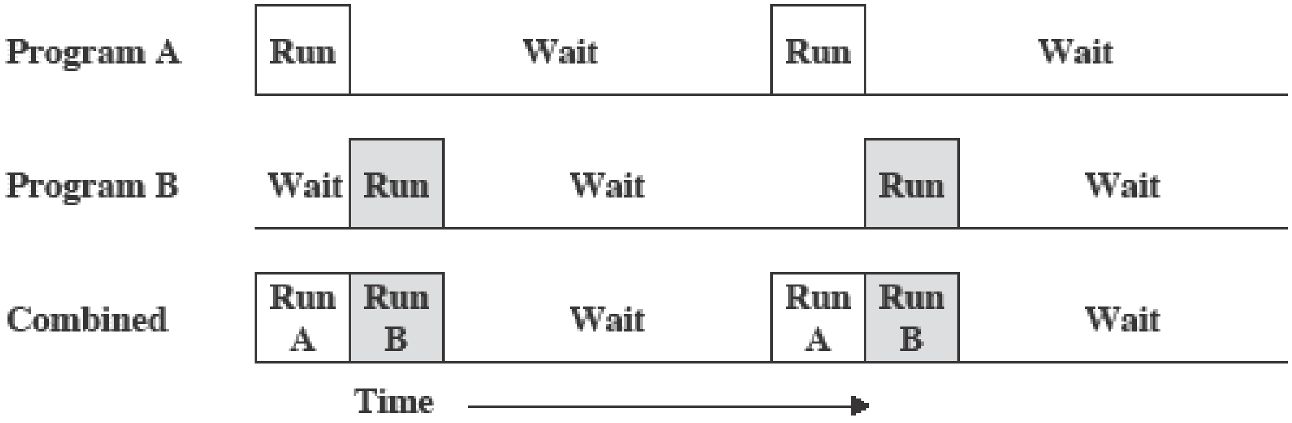
Figure 2.4 System Utilization Example

Uniprogramming



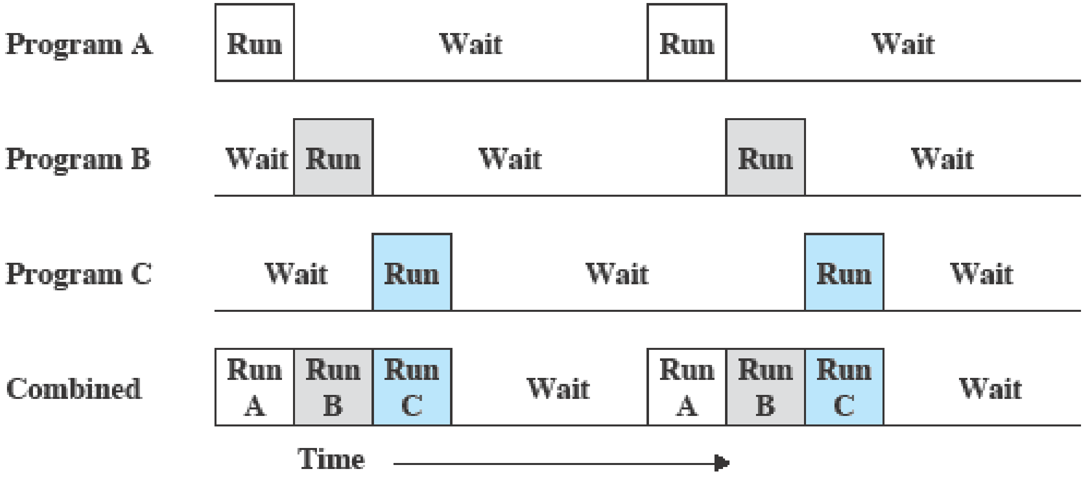
(a) Uniprogramming

Multiprogramming



(b) Multiprogramming with two programs

Multiprogramming



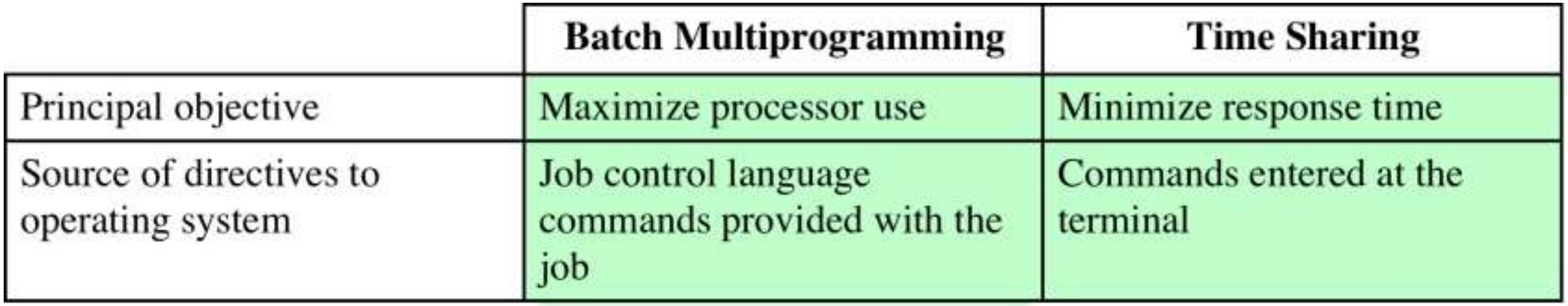
(c) Multiprogramming with three programs

- Multiprogramming
  - also known as multitasking
  - memory is expanded to hold three, four, or more programs and switch among all of them

Time-Sharing Systems

- Can be used to handle multiple interactive jobs
- Processor time is shared among multiple users
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation

Batch Multiprogramming vs. 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

Table 2.3 Batch Multiprogramming versus Time Sharing

Compatible Time-Sharing Systems

CTSS

- One of the first time-sharing operating systems

– Developed at MIT by a group known as Project MAC

– Ran on a computer with 32,000 *36*-bit words of main memory, with the resident monitor consuming 5000 of that

– To simplify both the monitor and memory management a program was always loaded to start at the location of the 5000th word

Time Slicing

– System clock generates interrupts at a rate of approximately one every 0.2 seconds

– At each interrupt OS regained control and could assign processor to another user

– At regular time intervals the current user would be preempted and another user loaded in

– Old user programs and data were written out to disk

– Old user program code and data were restored in main memory when that program was next given a turn