Operating Systems and Networks
AE4B33OSS

Introduction

Operating Systems and Networks

Operating system – Lecture 1-8

- Silberschatz A., Galvin P. B., Gagne G.: Operating System Concepts
  http://codex.cs.yale.edu/avi/os-book/OS7/os7c/index.html
- Tanenbaum A. S.: Modern Operating Systems
  http://www.cs.vu.nl/~ast/books/mos2/
- YouTube lectures :
  CS 162 – UC Berkeley
  OS-SP06 – Surendar Chandra – UC Berkeley
  MIT 6.004

Networking – Lecture 9-12/13

- Comer D. E.: Internetworking With TCP/IP Volume 1: Principles Protocols, and Architecture,
  http://www.cs.purdue.edu/homes/dec/vol1/vol1_presentation.pdf
- YouTube lectures :
  CS 162 – UC Berkeley

Operating System

Motto: Everybody is using OS but only few people know OS

Goal of course:

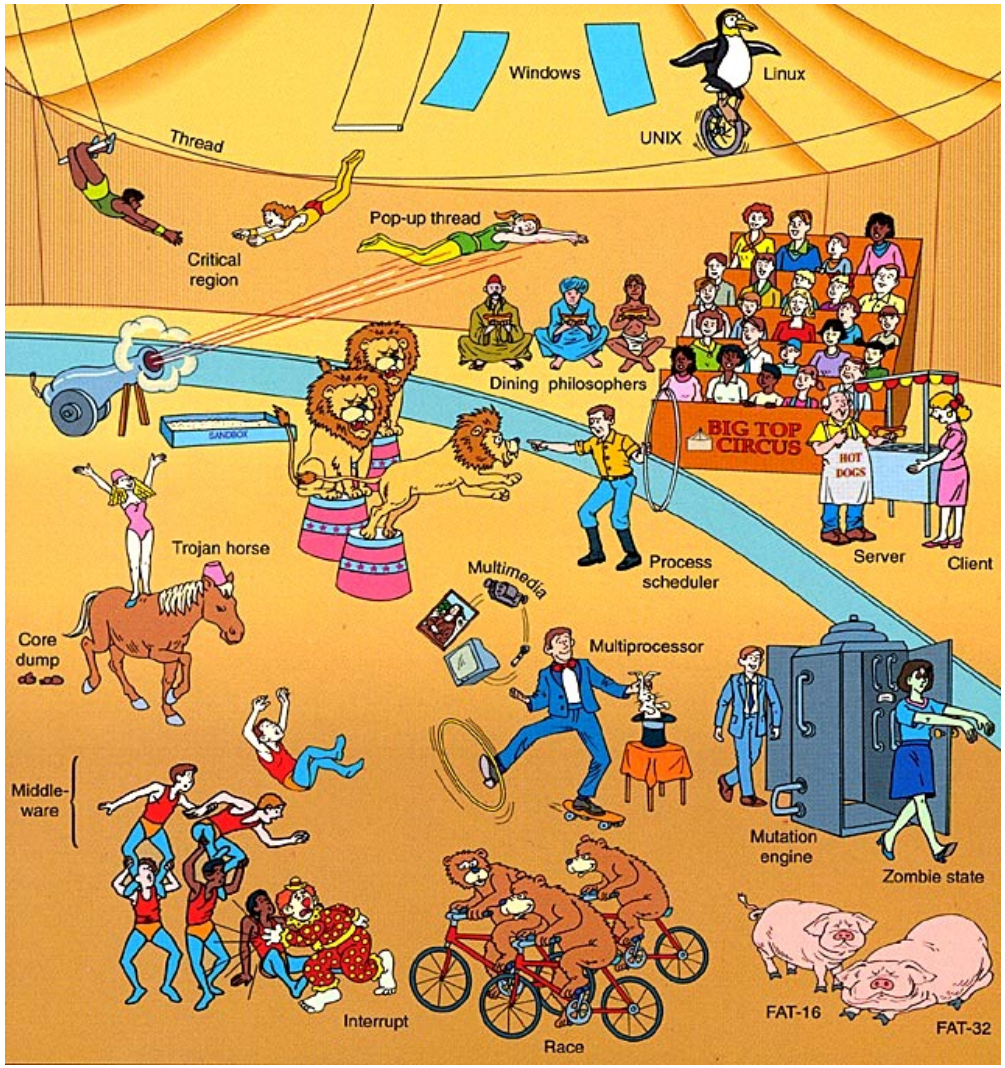
- To learn what is OS
- To learn principles of OS design
- To learn algorithms and known solution for complicated problems
- To learn how to use efficiently OS

Material:

- http://labe.felk.cvut.cz/courses/AE4B33OSS/2014
- Book: Silberschatz A., Galvin P.B., Gange G.: Operating Systems Concepts – http://codex.cs.yale.edu/avi/os-book/OS7/os7c/index.html

What is operating system?

Can you use computer without operating systems?



What is 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 memory storage and I/O devices

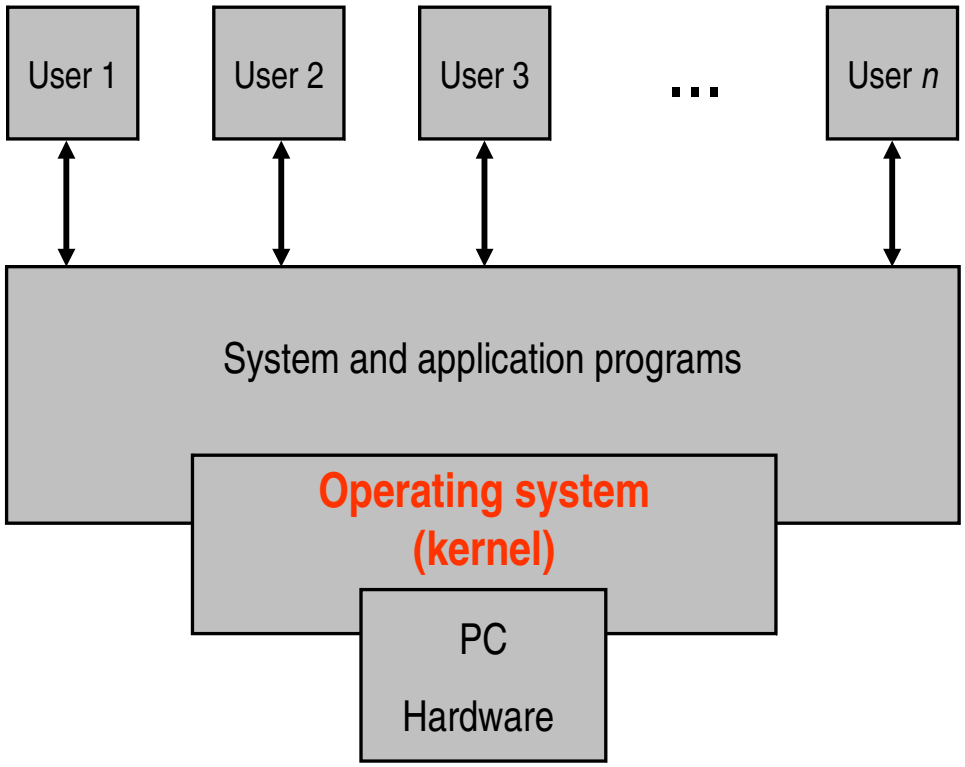
What Operating System needs?

- Operating system runs on a computer
- Operating system strongly depends on computer architecture
  - On CPU – type, number, instruction set
  - On bus – connection of components
  - On devices – drivers (programs that control the device)
- In this lesson we will suppose “general” computer
- Some approaches will be documented on OS Linux, Windows, MacOS for PC computer

What is operating system for this lecture?

In this lecture we will use operating system as oprating system kernel.

- other (so called system) programs are OS enlargement
- GUI – Windows is only graphical interface for users

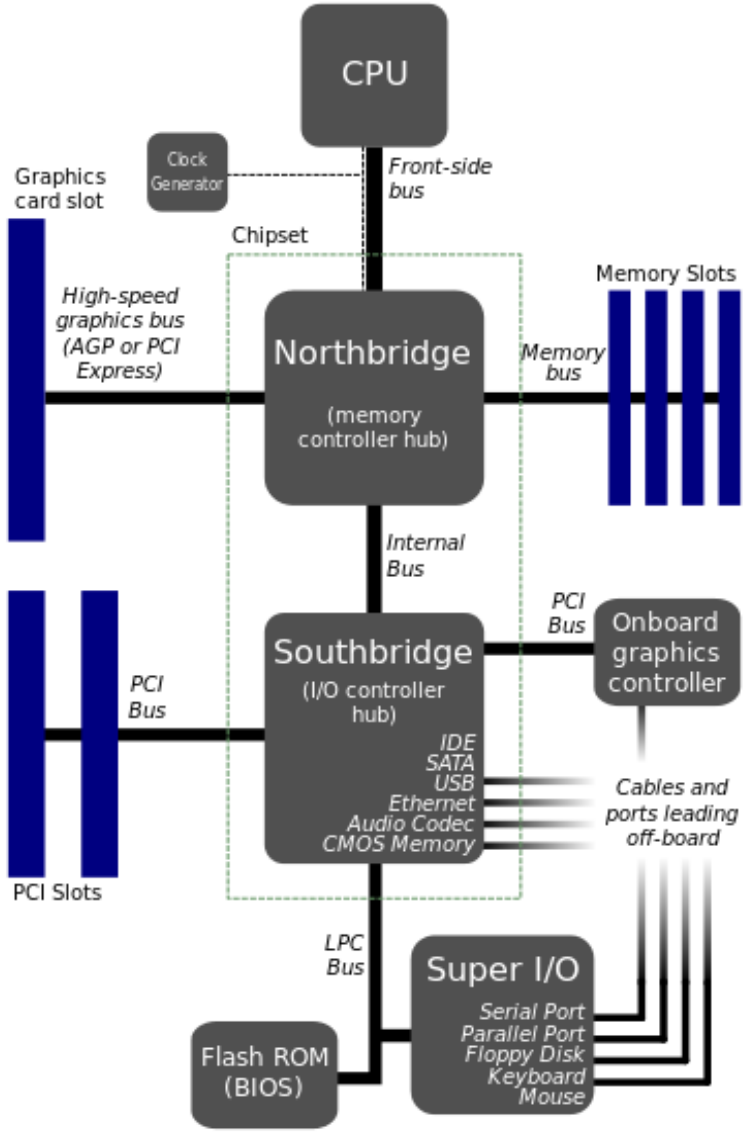


OS diversity

- OS for supercomputer (3,120,000 cores, 54,902TFlops, 17,8MW power)
- OS for data and networks server
- OS for personal computer
- OS for real-time application – control of air planes, rockets,...
- OS for mobile applications – telephone, tablet
- Embedded OS (printer, washing machine, ...)
- Smart card OS
- ... and other specialized OS

Elements of General - Personal Computer

- Processor (one or more)
- Main Memory
  - Volatile, real memory or primary memory
- System bus
  - Communication among processors, memory, and I/O modules
- I/O modules
  - Secondary memory – hard disk
  - Communications devices
  - Terminals
  - Printers …



Processor - CPU

- General Processor execute instructions only from memory
- Usually two modes of processor
  - System mode - privileged
    - Processor can do everything what is possible
  - User mode - restricted
    - Special operations are disabled
    - Disabled operation – operations that can modify state of the whole system
      - Halt, reset, Interrupt enable/disable, modification of processor mode, modification of MMU
      - Instruction for input and output (in / out)

Instruction Execution

- CPU works in steps – instruction cycles defined by clock signal
- Two steps
  - Processor reads instructions from memory
    - Fetches
  - Processor executes each instruction
- The processor fetches the instruction only from main memory
- Program counter (PC) holds address of the instruction to be fetched next
- Program counter is incremented after each fetch

Instruction Cycle

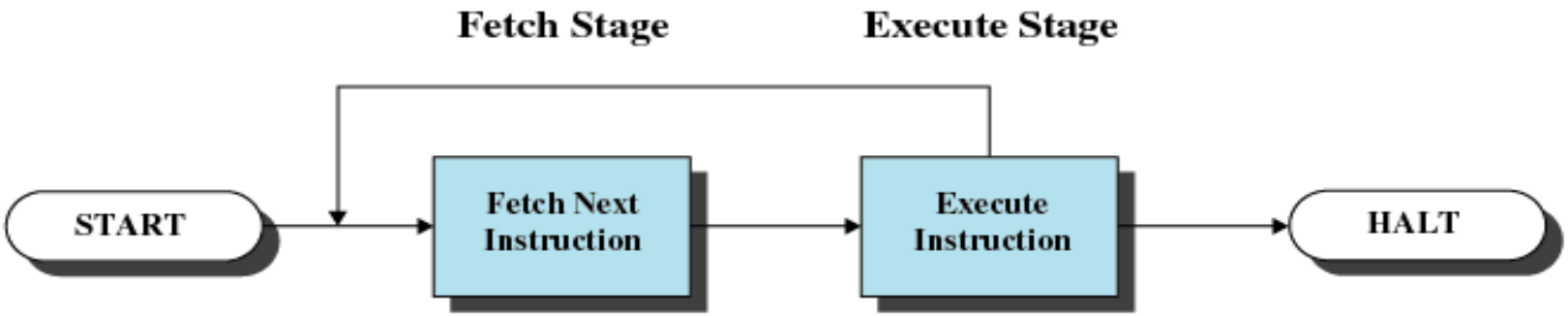
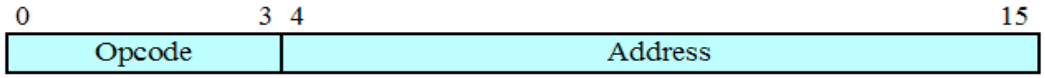
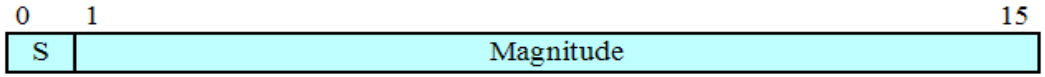


Figure 1.2 Basic Instruction Cycle

Characteristics of a Hypothetical Machine



(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

Example of Program Execution

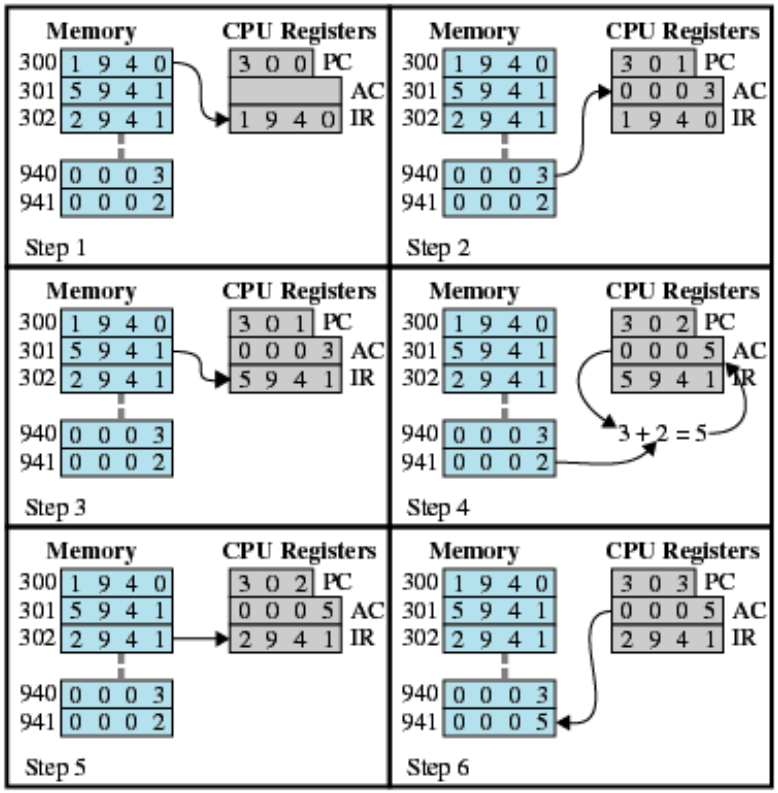


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

Top-Level Components

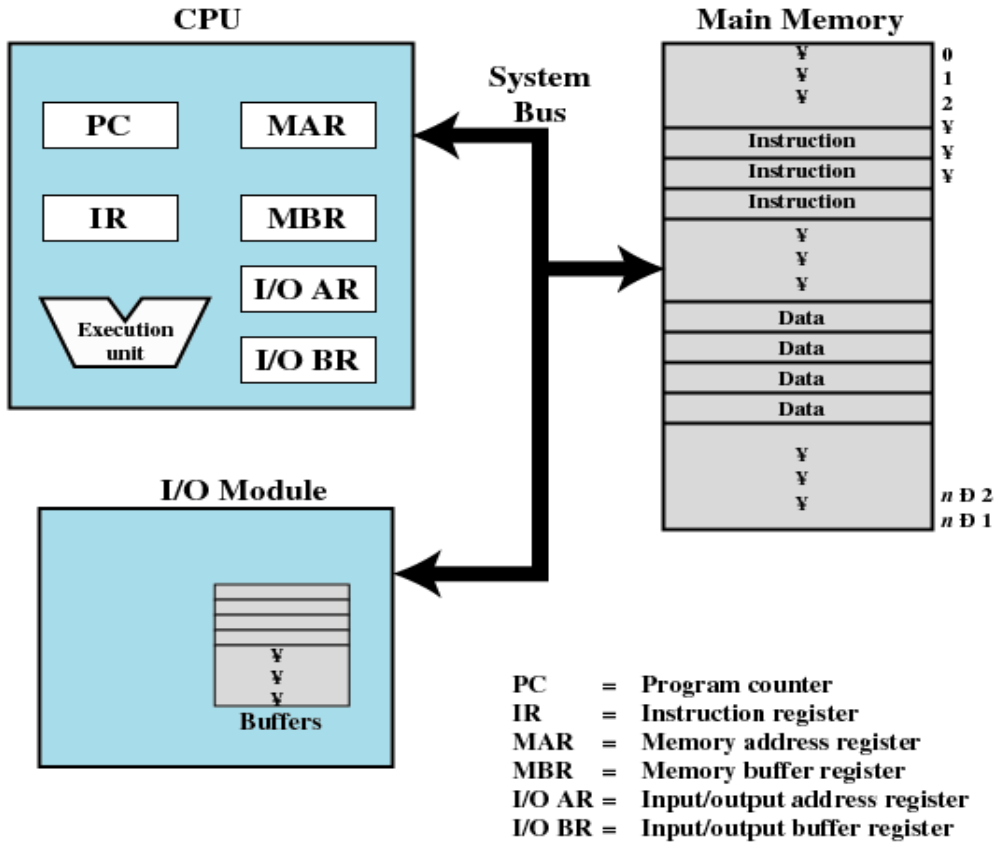


Figure 1.1 Computer Components: Top-Level View

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)
  - Condition codes
  - Interrupt enable/disable
  - Supervisor/user mode
  - Used by privileged operating-system routines to control the execution of programs

Control and Status Registers

- Condition Codes or Flags
  - Bits set by the processor hardware as a result of operations
  - Examples
    - Positive result
    - Negative result
    - Zero
    - Overflow

Processor

- Two internal registers
  - Memory address register (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
  - I/O address register
  - I/O buffer register
- User-visible registers
  - Enable programmer to minimize main-memory references by optimizing register use

User-Visible Registers

- May be referenced by machine language
- Available to all programs - application programs and system programs
- Types of registers
  - Data
  - Address
    - Index
    - Segment pointer
    - Stack pointer

User-Visible Registers

- Address Registers
  - Index
    - Involves adding an index to a base value to get an 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

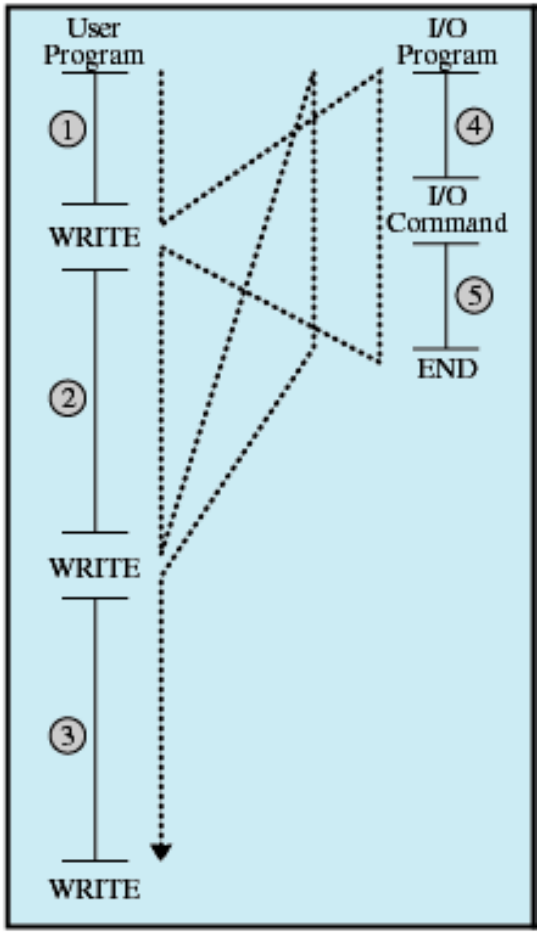
Interrupts

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

Table 1.1 Classes of Interrupts

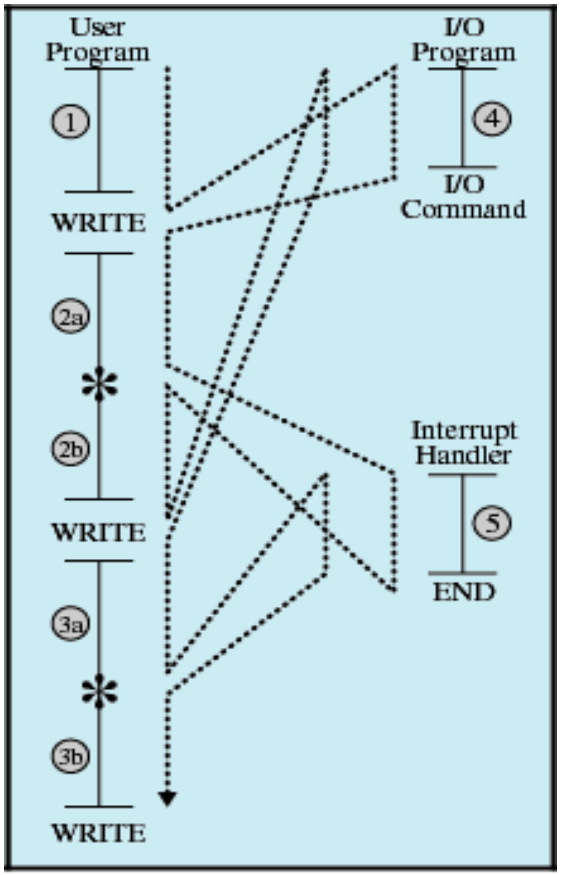
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows 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 Without Interrupts



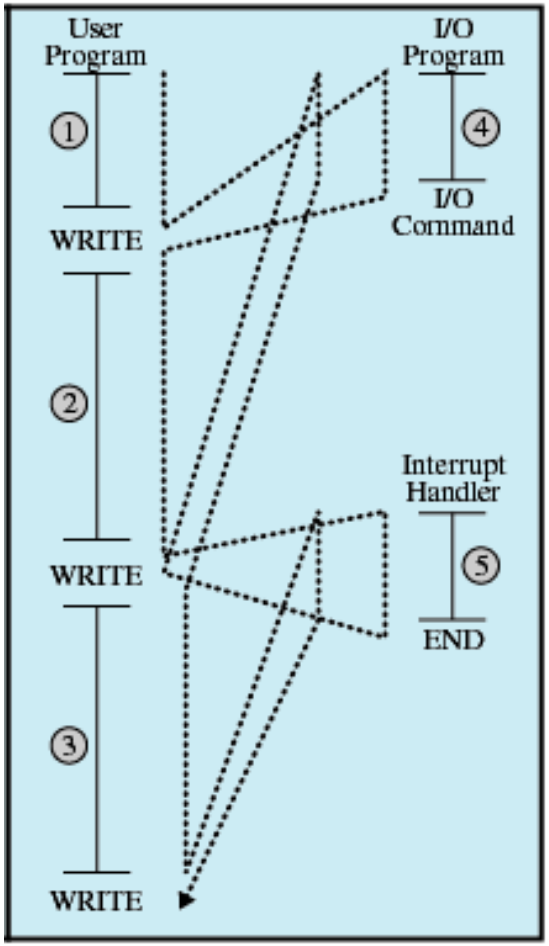
(a) No interrupts

Program Flow of Control With Interrupts, Short I/O Wait



(b) Interrupts; short I/O wait

Program Flow of Control With Interrupts; Long I/O Wait



(c) Interrupts; long I/O wait

Interrupts

- Interrupt handler:
  - Program to service a particular I/O device
  - Generally part of the operating system
  - Suspends the normal sequence of execution

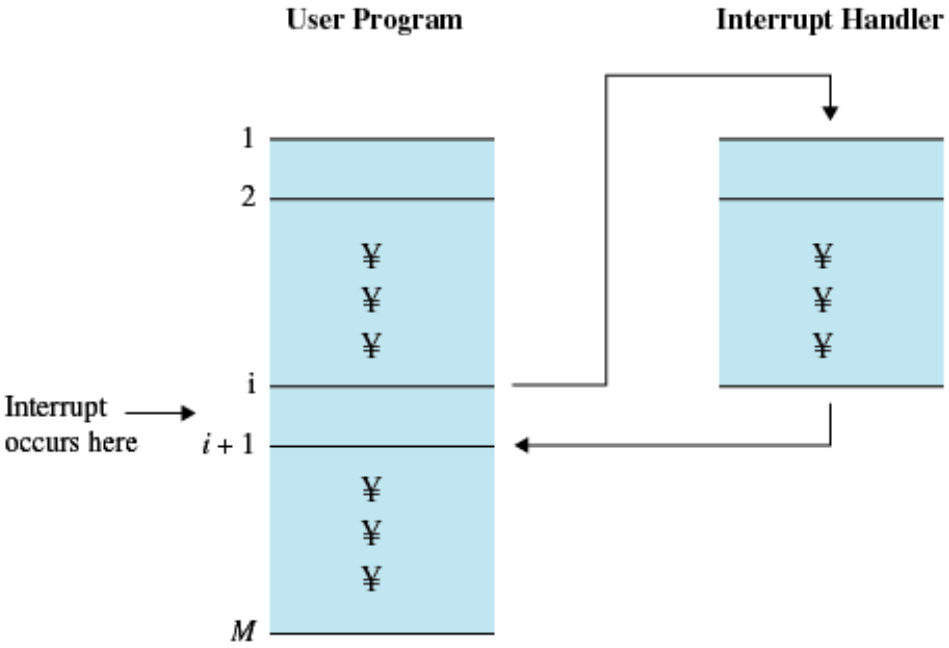


Figure 1.6 Transfer of Control via Interrupts

Interrupt Cycle

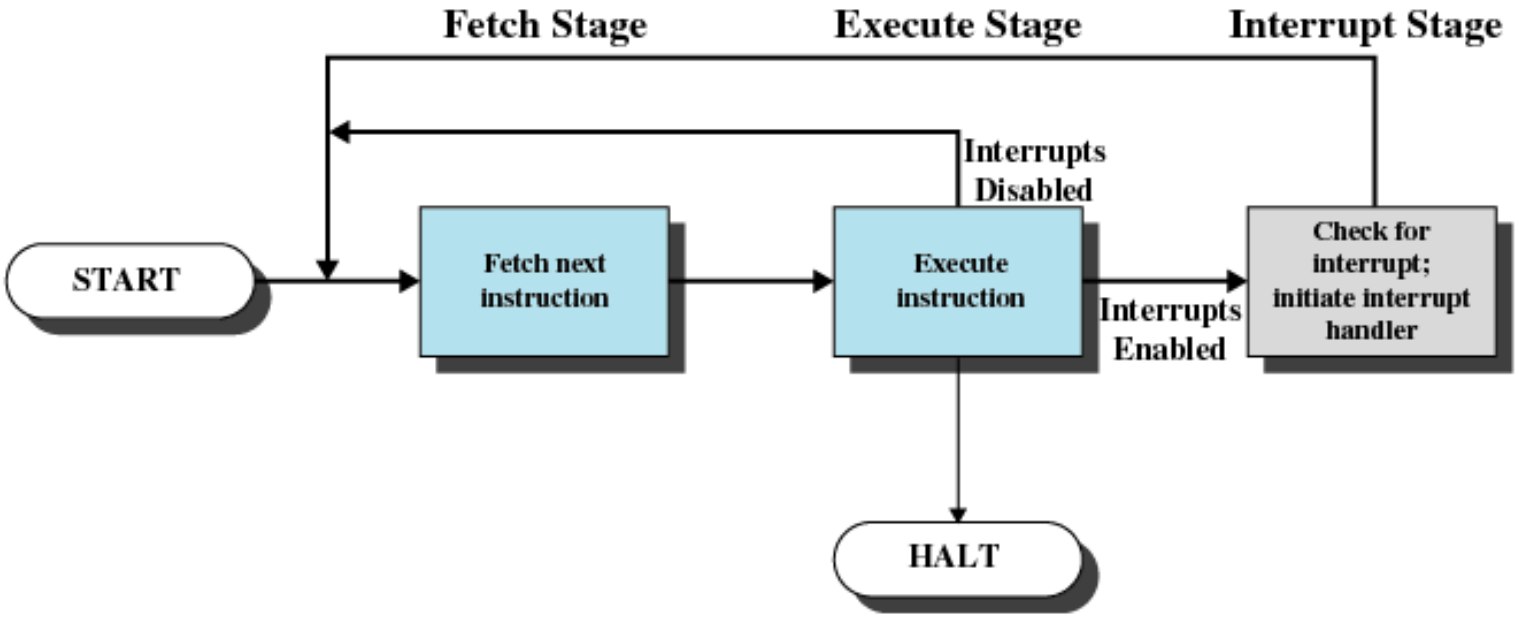


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Cycle

- Processor checks for interrupts
- If there is no interrupt then fetch the next instruction for the current program
- If an interrupt is pending (waiting), suspend execution of the current program, and execute the interrupt-handler routine

Simple Interrupt Processing

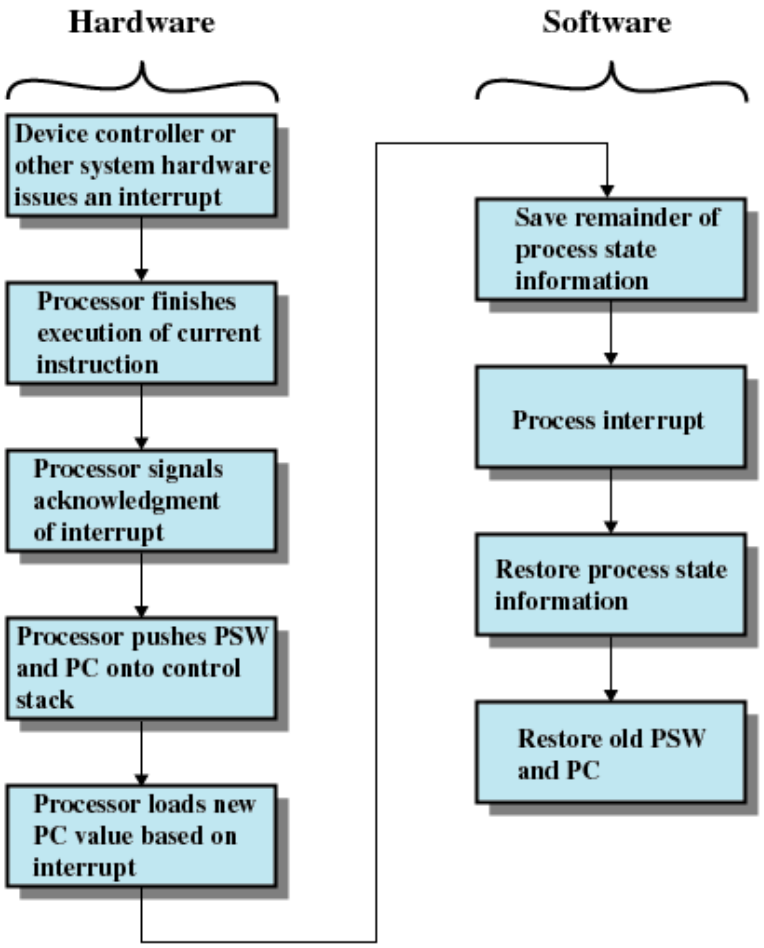
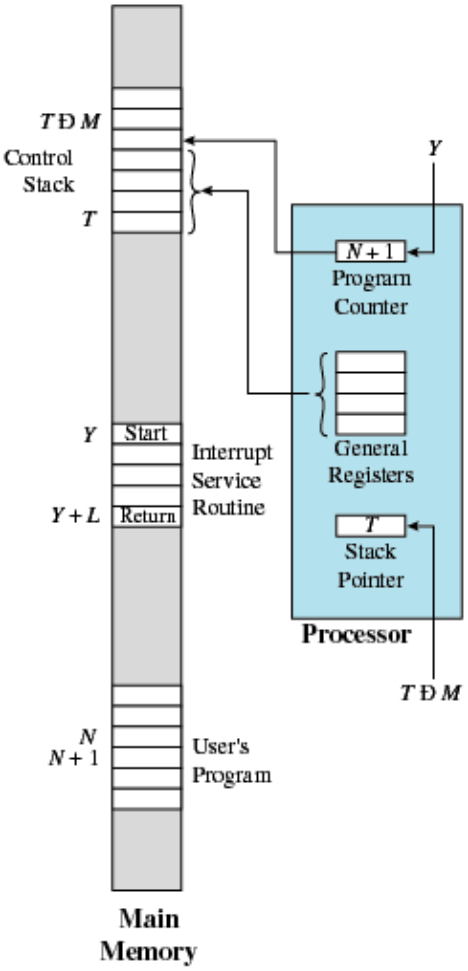


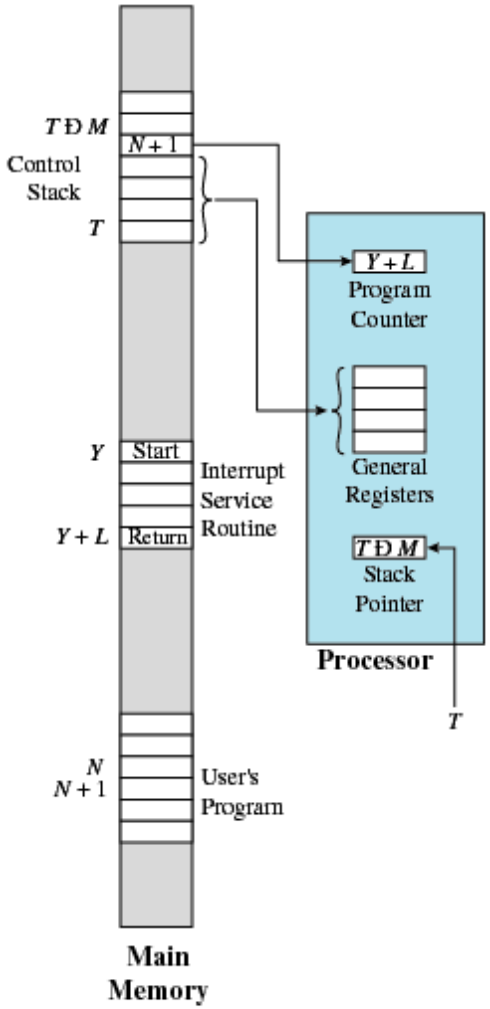
Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

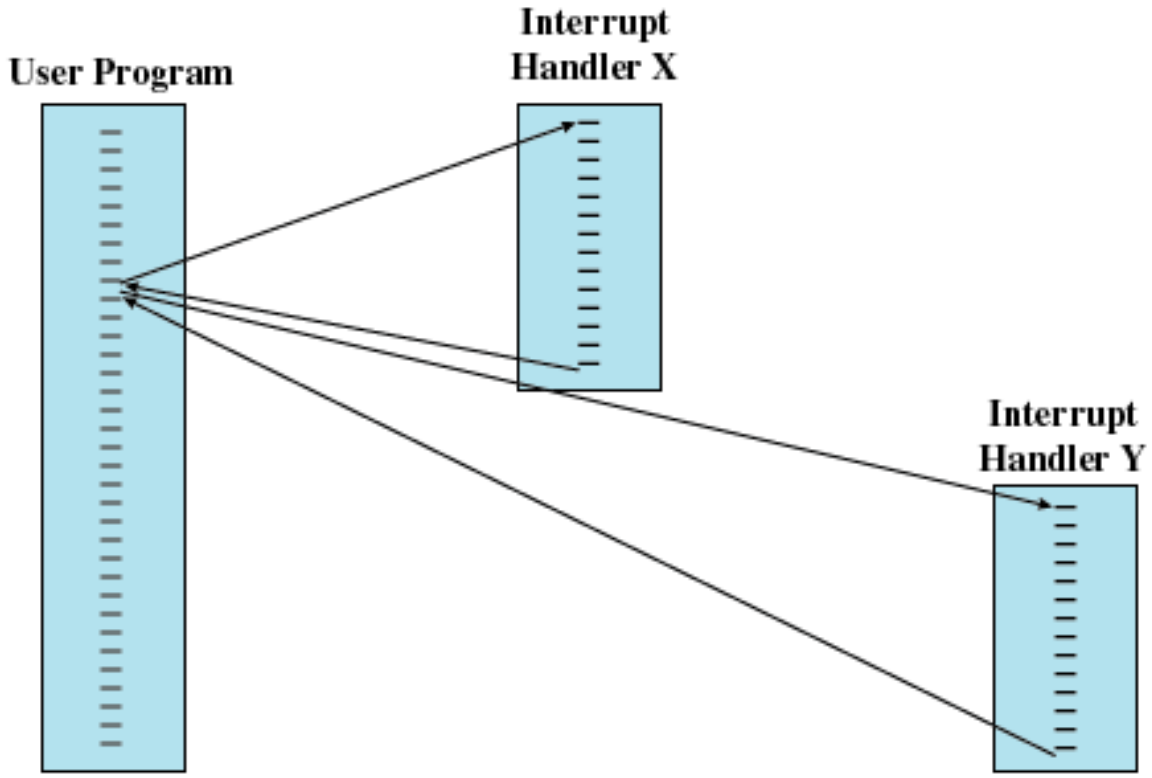
Changes in Memory and Registers for an Interrupt



(b) Return from interrupt

Multiple Interrupts

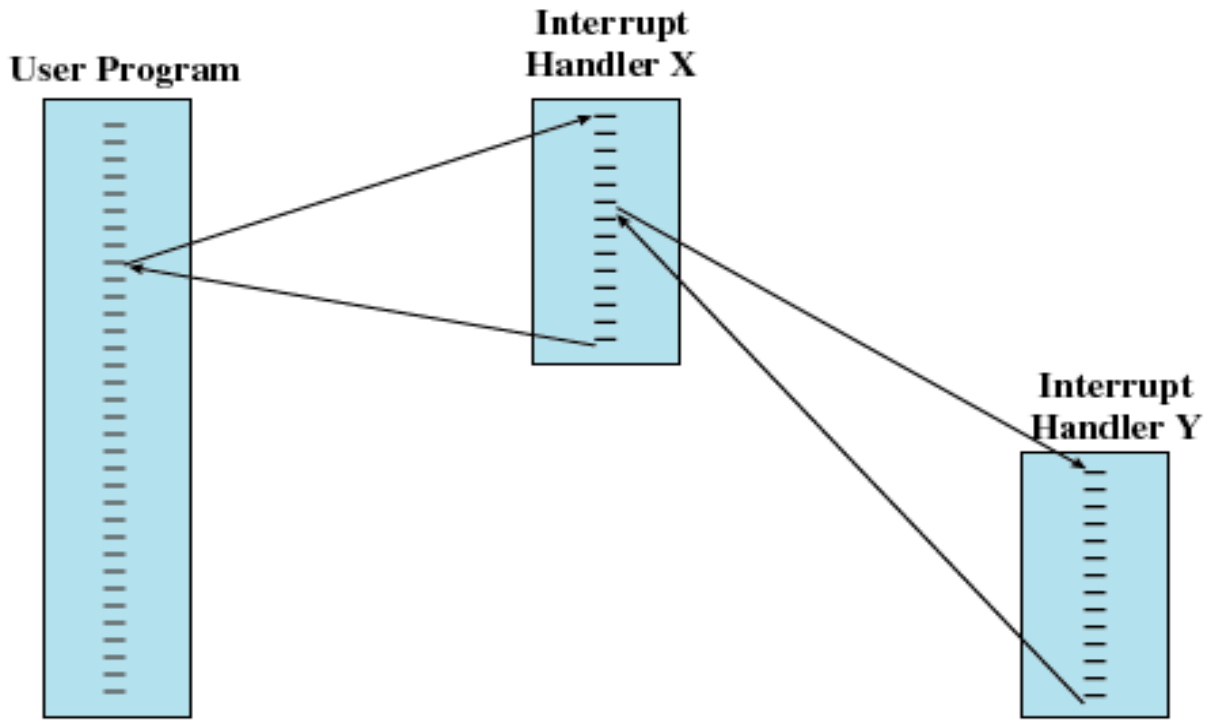
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupts



(b) Nested interrupt processing

Multiple Interrupts

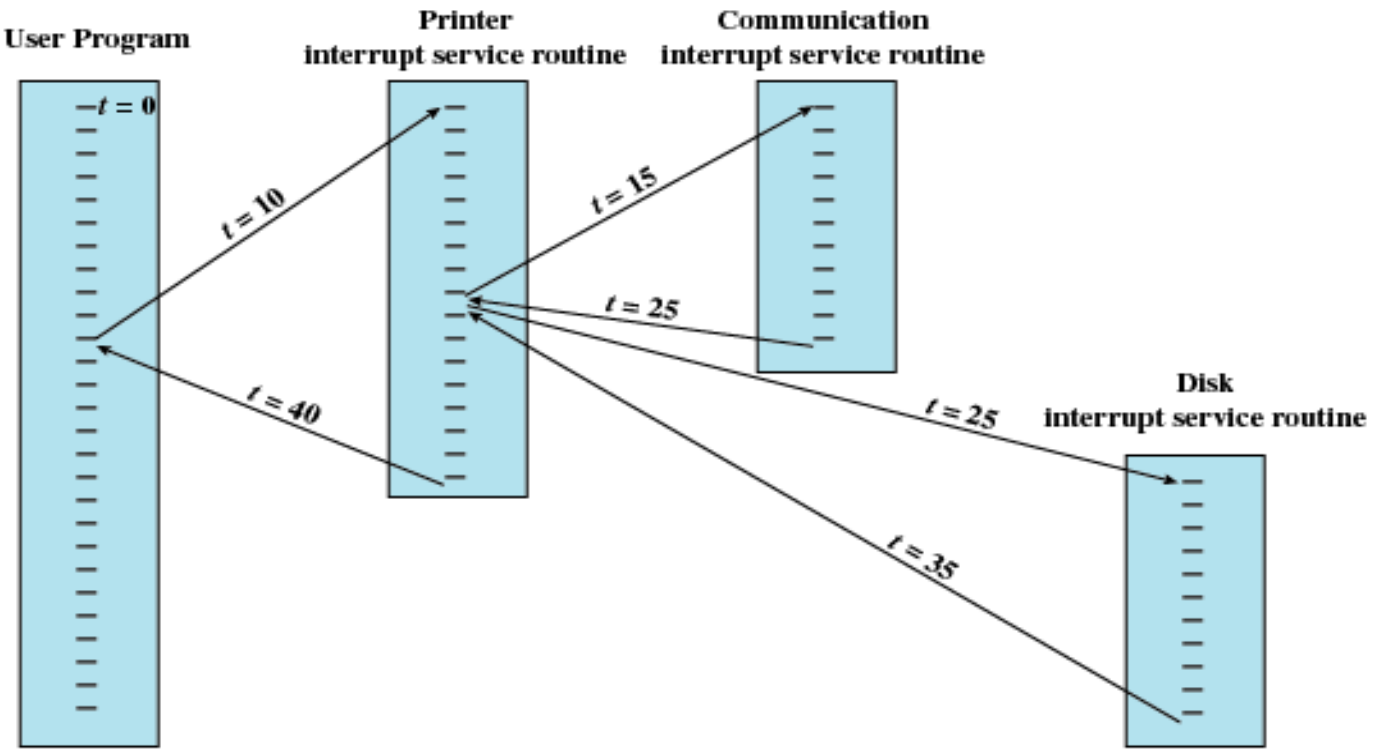


Figure 1.13 Example Time Sequence of Multiple Interrupts

Multiprogramming

- 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
- Multiprogramming depends on timer interrupt
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt. Interrupt makes context switch, store old process status into memory and load status of another process

Memory Hierarchy

- Faster access time, greater cost per bit
  - Cache memory is fast but it is small because it is expensive
- Greater capacity, smaller cost per bit & slower access speed
  - DVD memory is cheap but the CPU need first to read data into main memory – it is slow

Memory Hierarchy

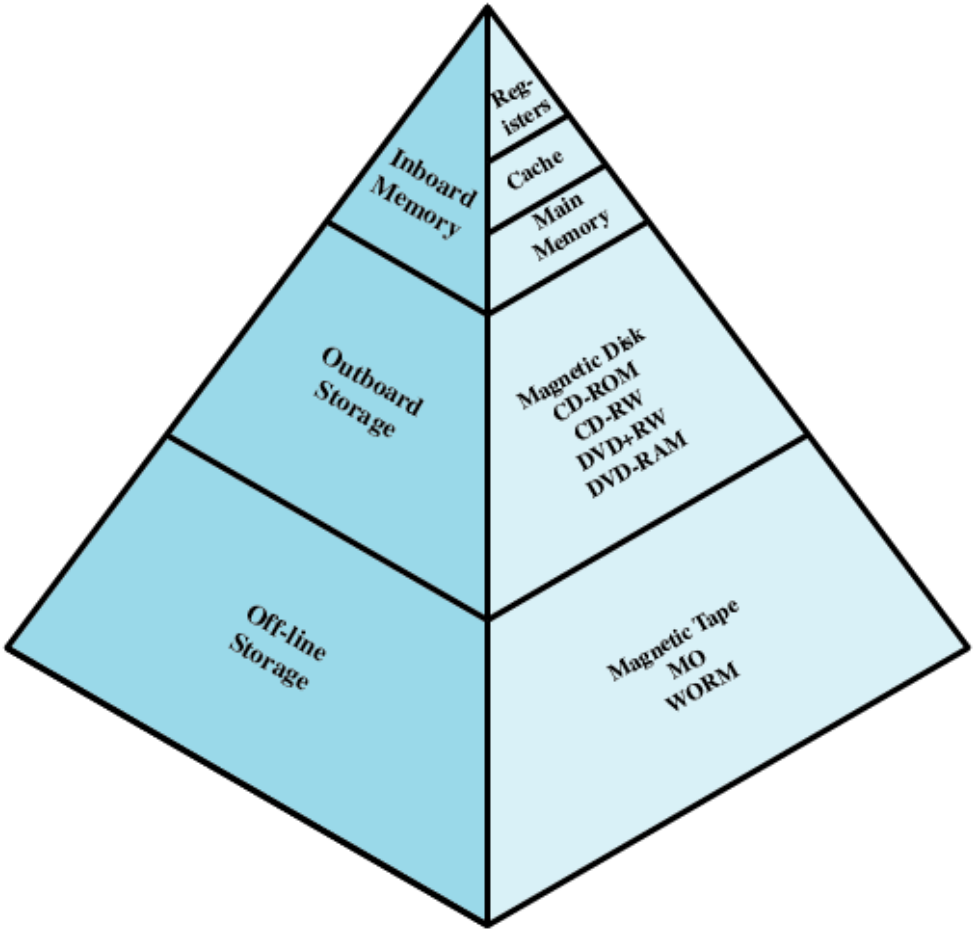


Figure 1.14 The Memory Hierarchy

Going Down the Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
  - Locality of reference

Cache Memory

- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality

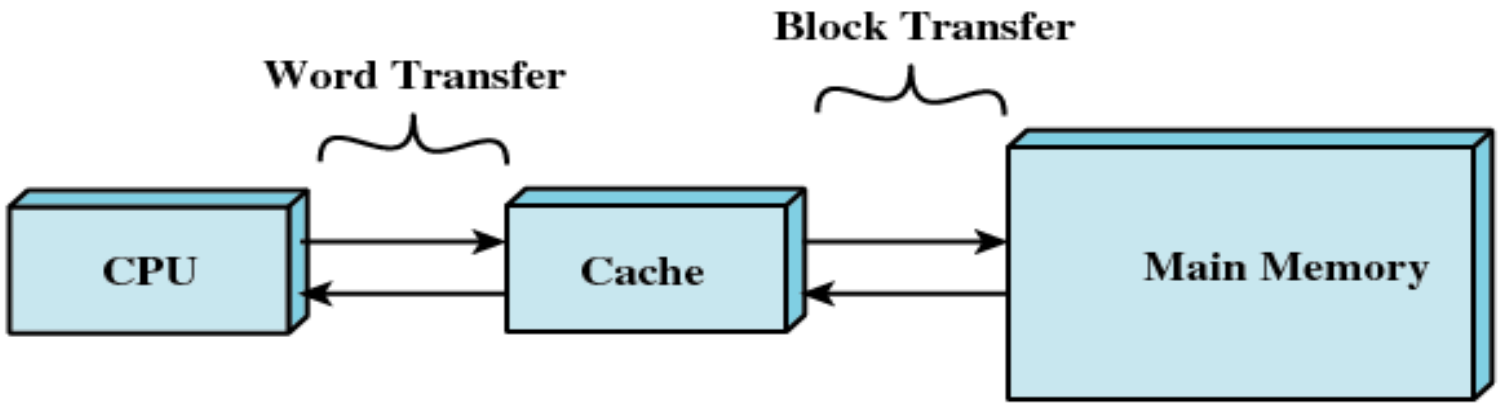


Figure 1.16 Cache and Main Memory

Cache Memory

- Contains a copy of a portion of main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor

Cache/Main Memory System

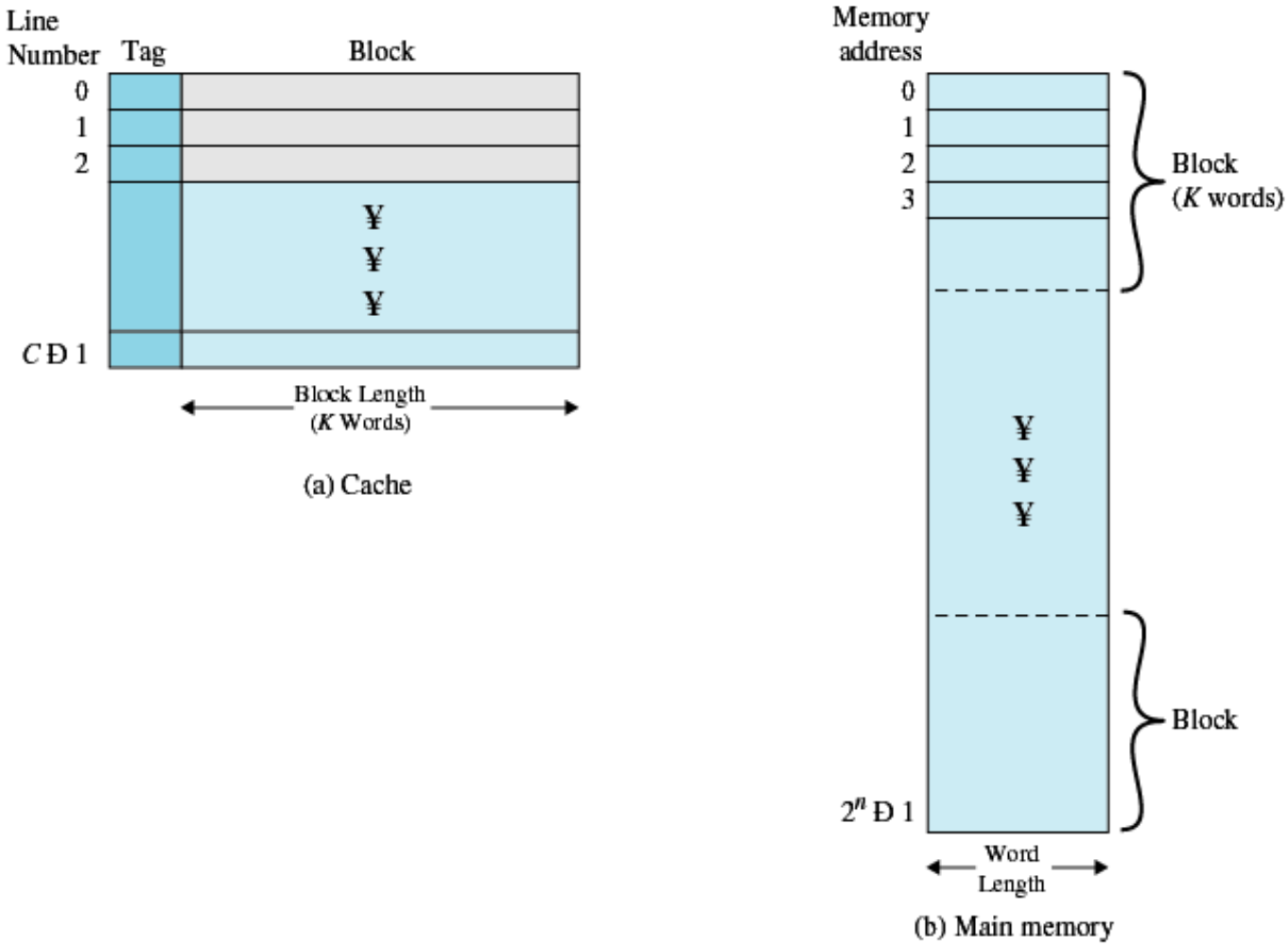


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation

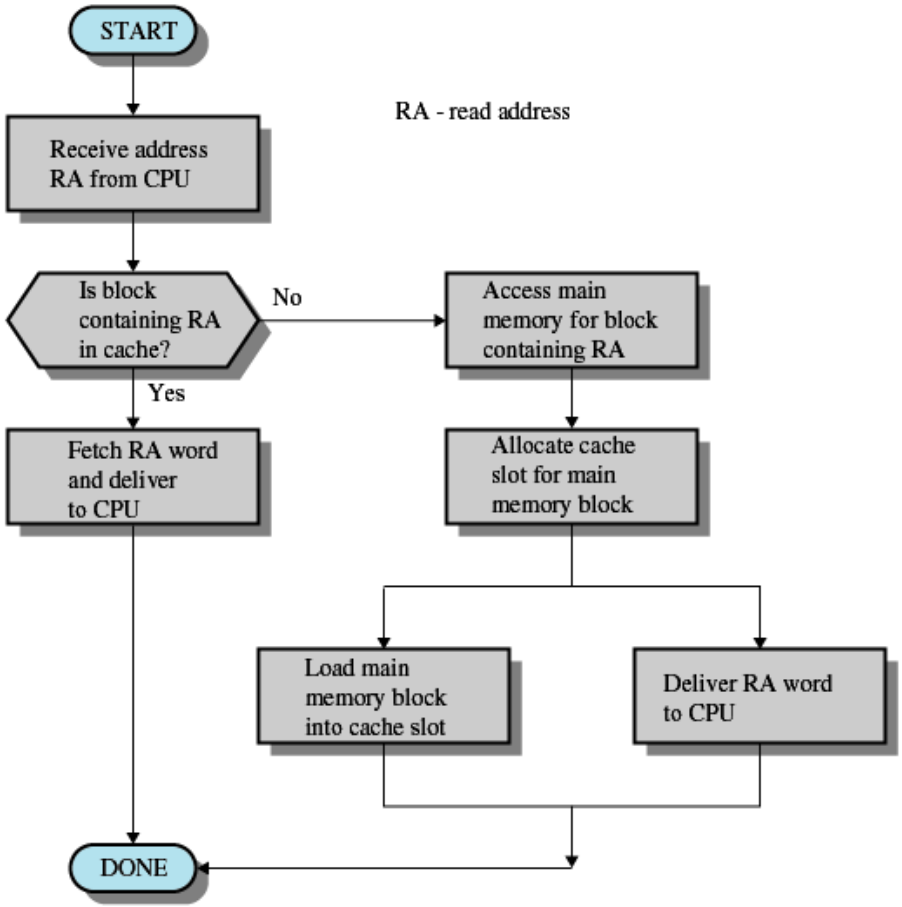


Figure 1.18 Cache Read Operation

Cache Design

- Cache size
  - Small caches have a significant impact on performance
- Block size
  - The unit of data exchanged between cache and main memory
  - Larger block size more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache

Cache Design

- Mapping function
  - Determines which cache location the block will occupy
- Replacement algorithm
  - Determines which block to replace
  - Least-Recently-Used (LRU) algorithm

Cache Design

- Write policy
  - When the memory write operation takes place
  - Can occur every time block is updated
  - Can occur only when block is replaced
    - Minimizes memory write operations
    - Leaves main memory in an obsolete state

Secondary Memory

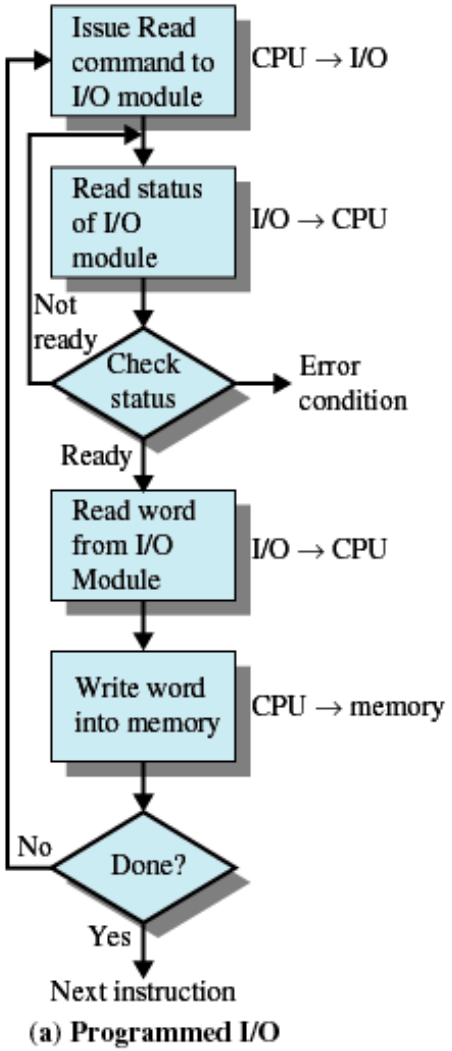
- Nonvolatile
- Auxiliary memory
- Used to store program and data files
- Hard-drive
- SSD
- CD, DVD, Blue-Ray

Disk Cache

- A portion of main memory used as a buffer to temporarily to hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk

Programmed I/O

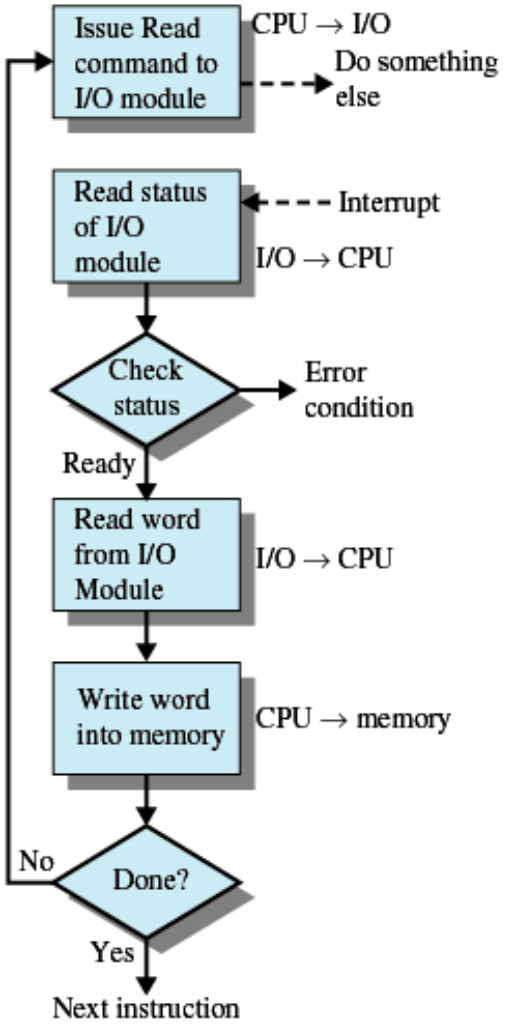
- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete



(a) Programmed I/O

Interrupt-Driven I/O

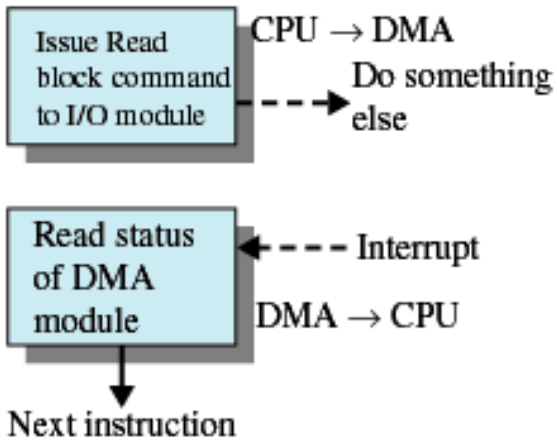
- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(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
- Processor continues with other work



(c) Direct memory access

Direct Memory Access (DMA)

- I/O exchanges occur directly with memory
- Processor grants I/O module authority to read from or write to memory
- Relieves the processor responsibility for the exchange