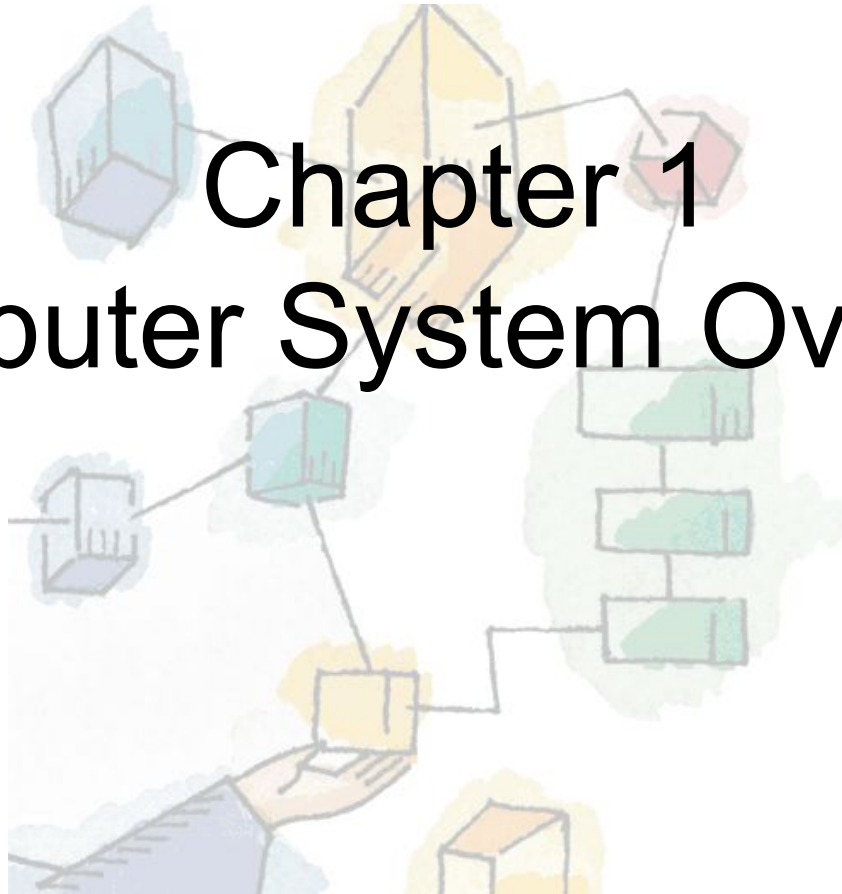


*Operating Systems:  
Internals and Design Principles, 6/E*  
William Stallings

# Chapter 1

## Computer System Overview





# Roadmap



## Basic Elements

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





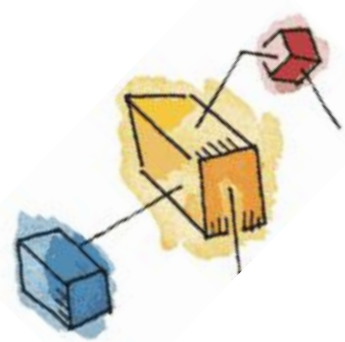
# Operating System

- Exploits the hardware resources of one or more processors
- Provides a set of services to system users
- Manages secondary memory and I/O devices



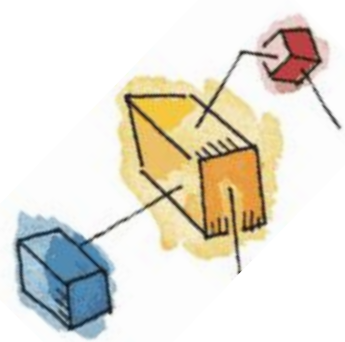
# A Computer's Basic Elements

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



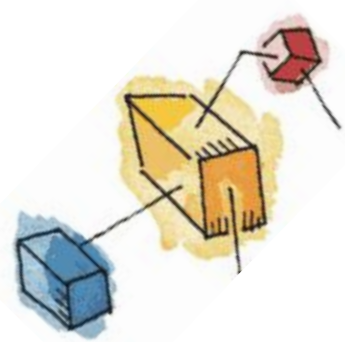
# Processor

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



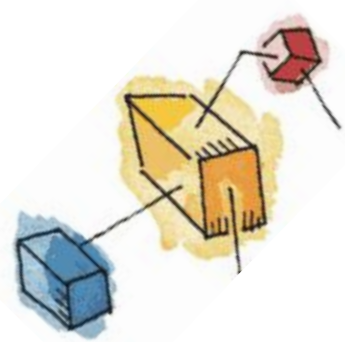
# Main Memory

- 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



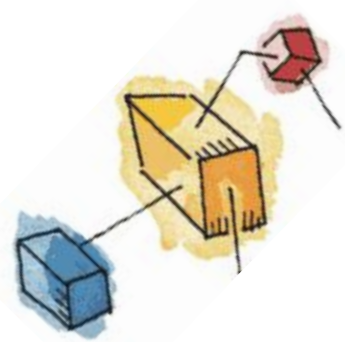
# I/O Modules

- 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)



# System Bus

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





# Top-Level View

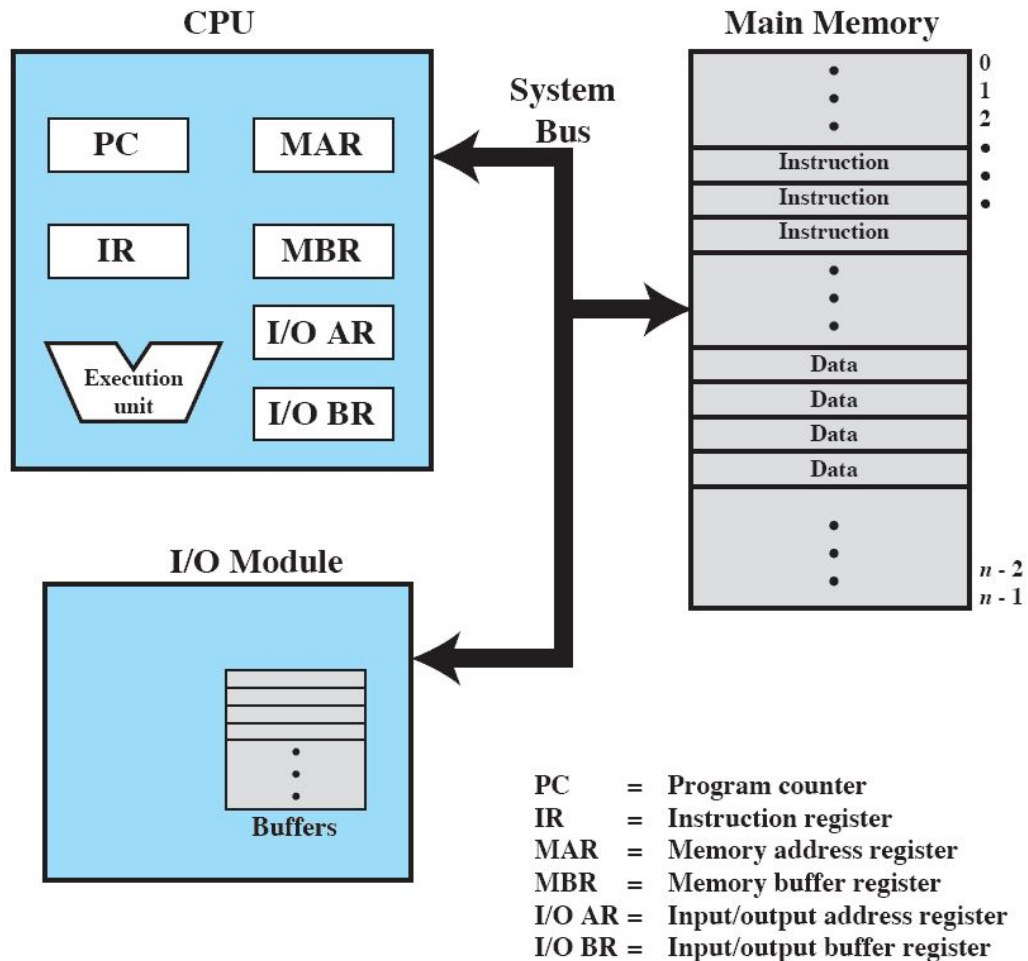


Figure 1.1 Computer Components: Top-Level View



# Roadmap

- Basic Elements

 Processor Registers

- Instruction Execution

- Interrupts

- The Memory Hierarchy

- Cache Memory

- I/O Communication Techniques





# Processor Registers

- 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





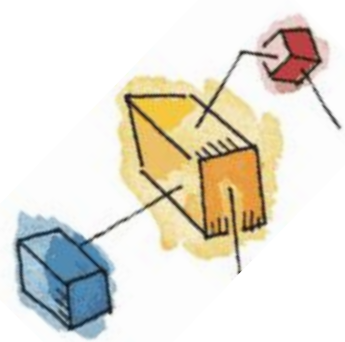
# User-Visible Registers

- 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.



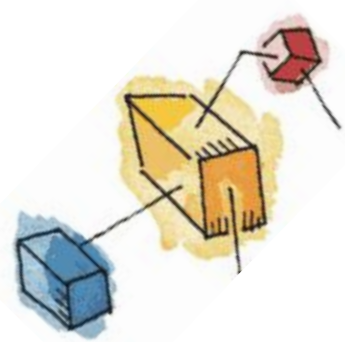
# Data and Address Registers

- Data
  - Often general purpose
  - But some restrictions may apply
- Address
  - Index Register
  - Segment pointer
  - Stack pointer



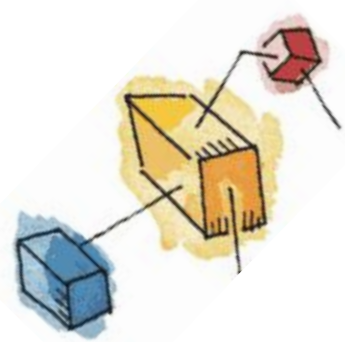
# Control and Status Registers

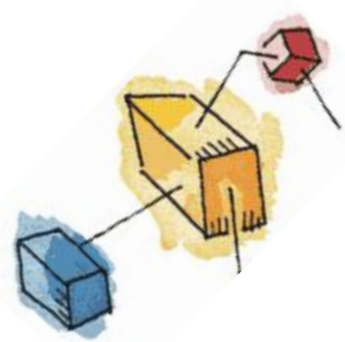
- Program counter (PC)
  - Contains the address of an instruction to be fetched
- Instruction register (IR)
  - Contains the instruction most recently fetched
- Program status word (PSW)
  - Contains status information



# Roadmap

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





# 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





# Basic Instruction Cycle

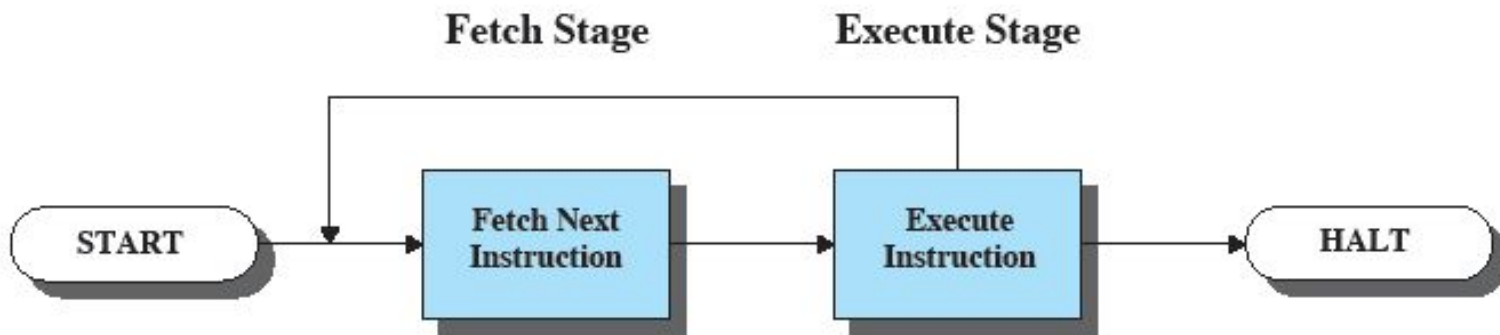
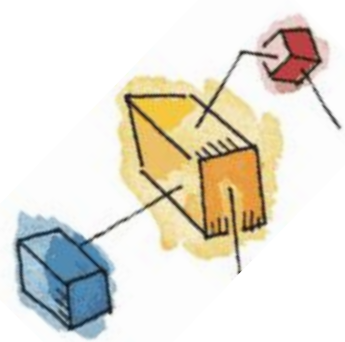


Figure 1.2 Basic Instruction Cycle



# 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

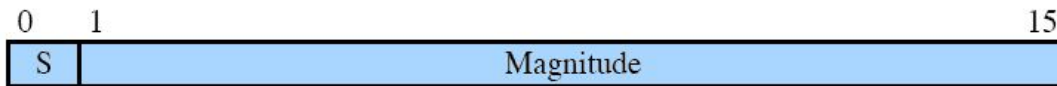
- Fetched instruction loaded into instruction register
- Categories
  - Processor-memory,
  - processor-I/O,
  - Data processing,
  - Control



# 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

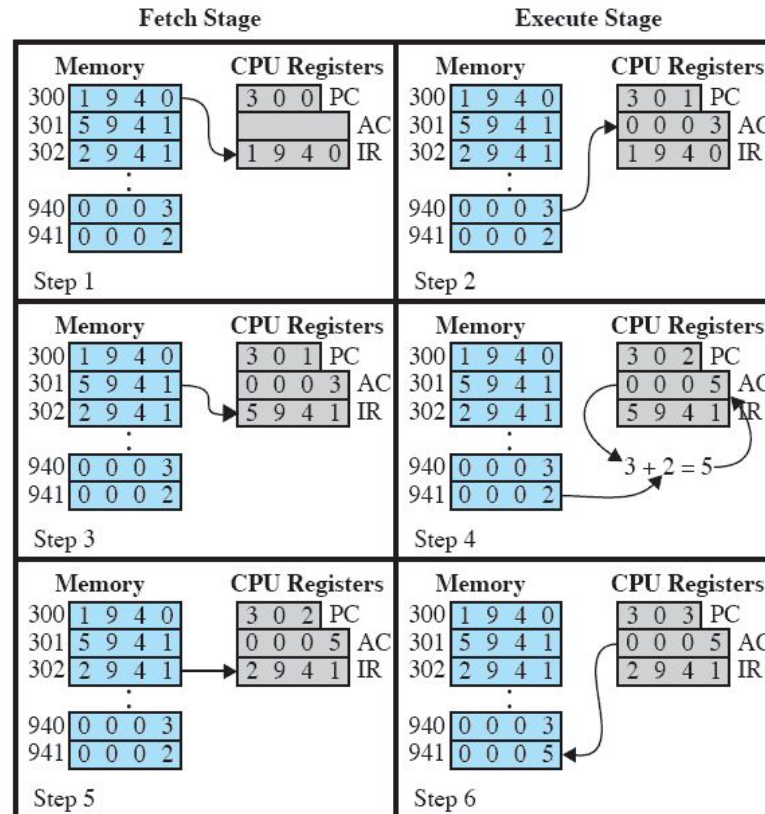


Figure 1.4 Example of Program Execution  
(contents of memory and registers in hexadecimal)

# Roadmap

- Basic Elements
- Processor Registers
- Instruction Execution

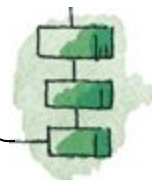
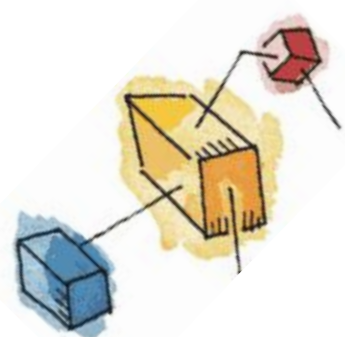
## → Interrupts

- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques



# 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



# Common Classes of Interrupts

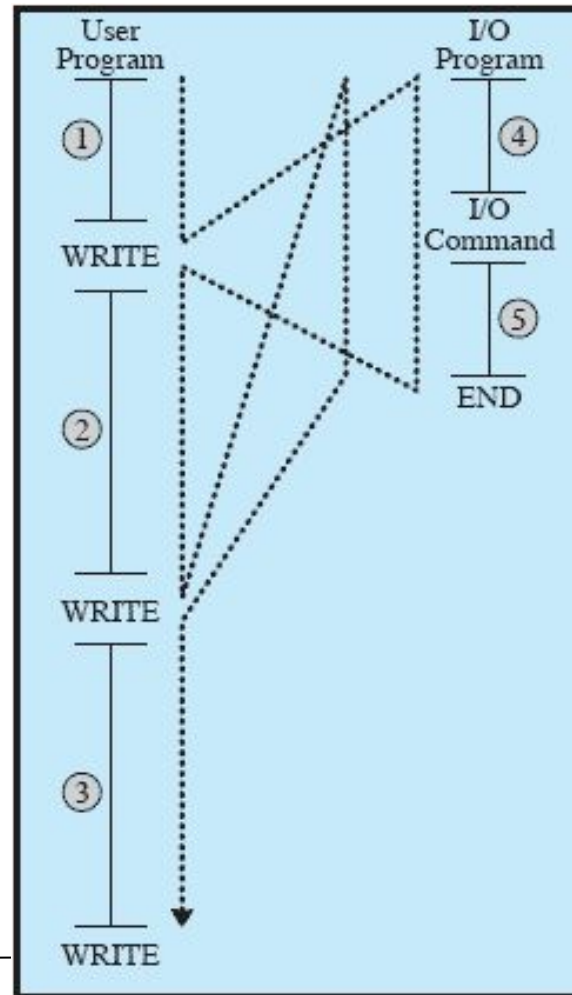
**Table 1.1** Classes of Interrupts

<b>Program</b>	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.
<b>Timer</b>	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
<b>I/O</b>	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
<b>Hardware failure</b>	Generated by a failure, such as power failure or memory parity error.



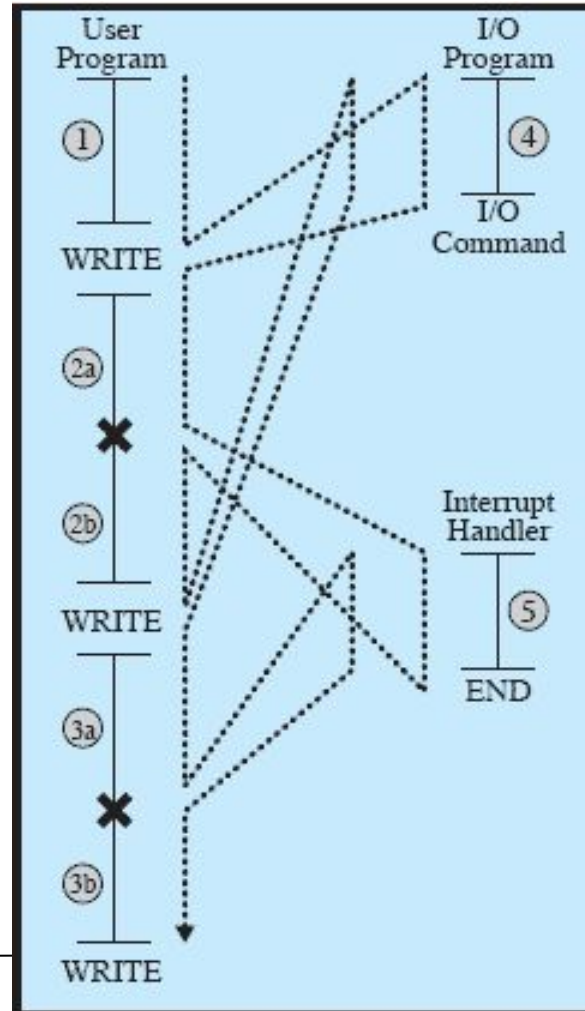


# Flow of Control without Interrupts



(a) No interrupts

# Interrupts and the Instruction Cycle



(b) Interrupts; short I/O wait

# Transfer of Control via Interrupts

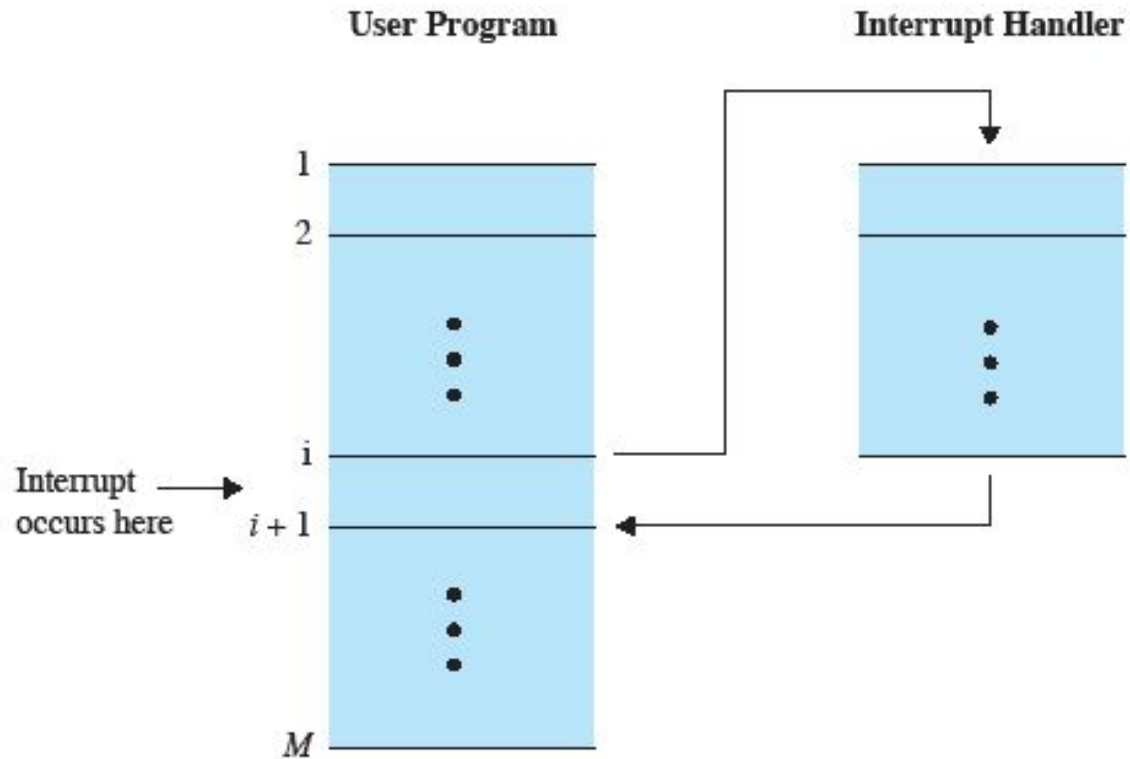


Figure 1.6 Transfer of Control via Interrupts

# Instruction Cycle with Interrupts

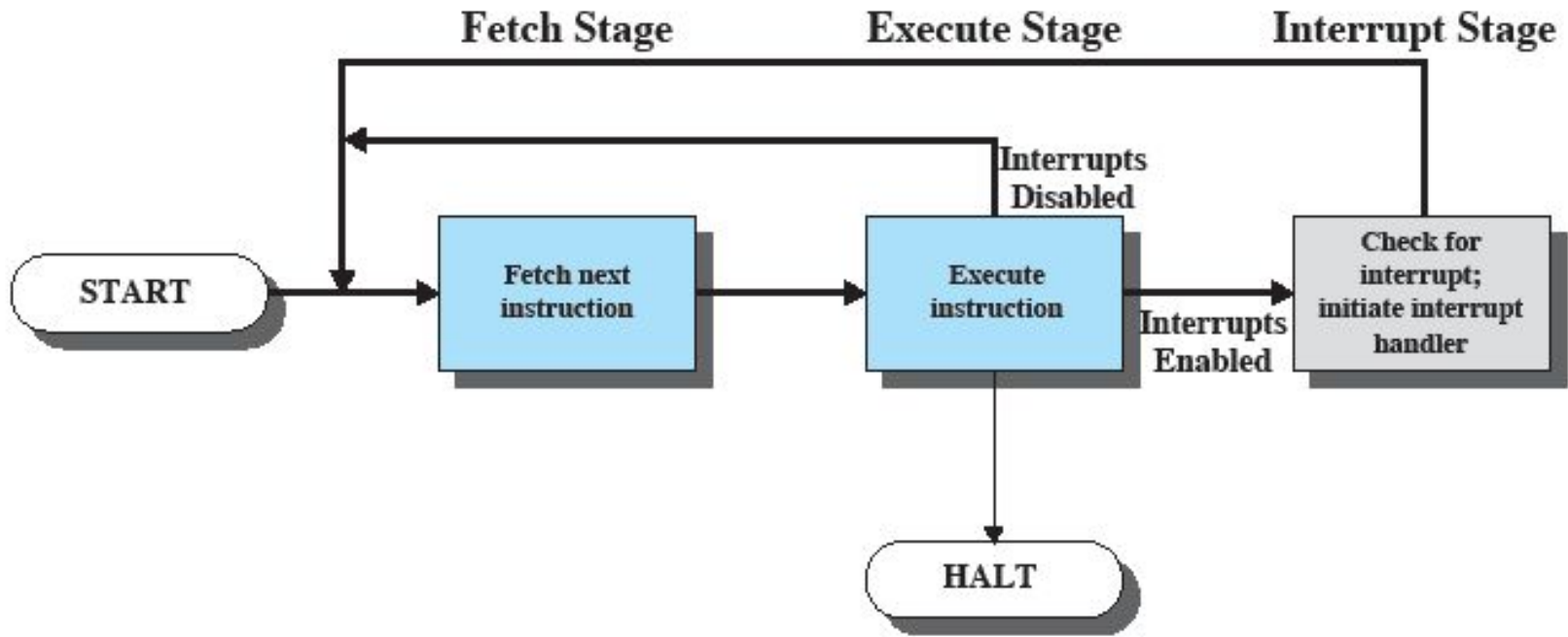
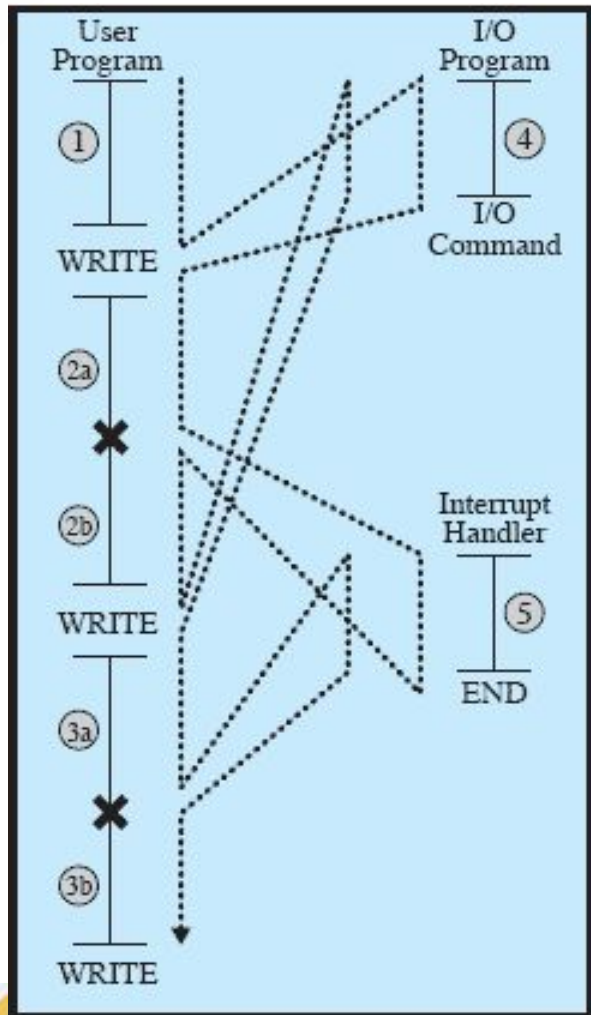
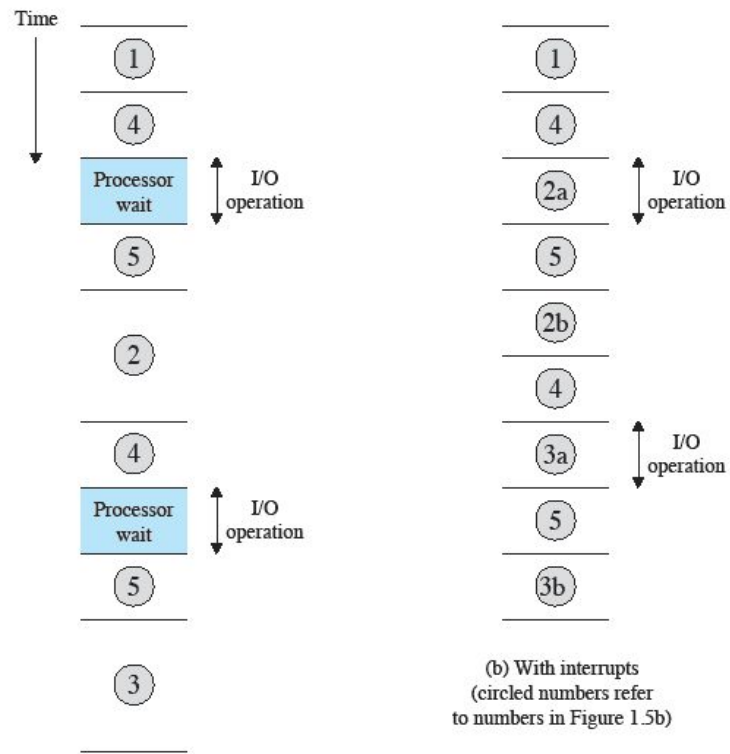


Figure 1.7 Instruction Cycle with Interrupts

# Short I/O Wait



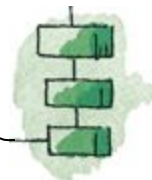
(b) Interrupts; short I/O wait



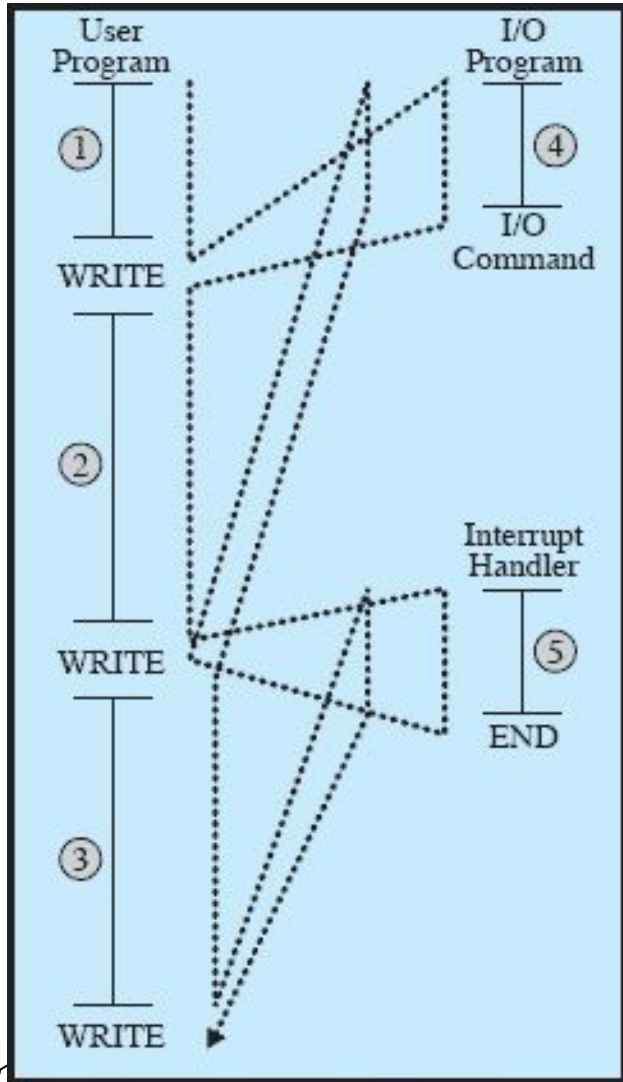
(a) Without interrupts  
(circled numbers refer to numbers in Figure 1.5a)

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

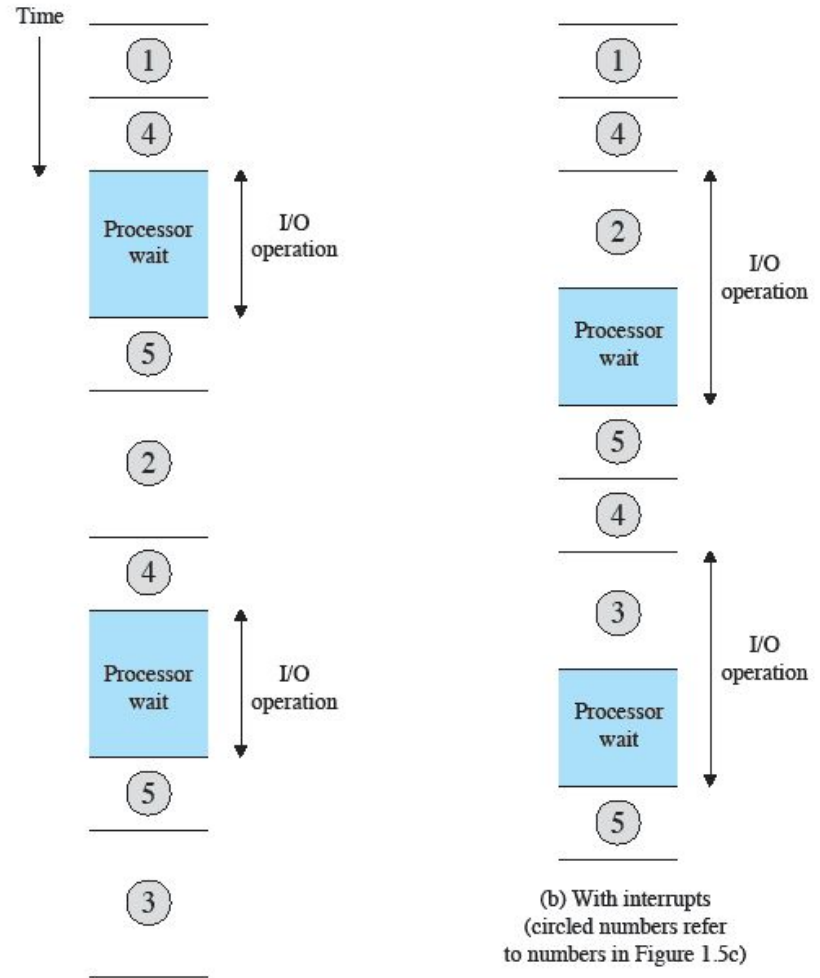
Figure 1.8 Program Timing: Short I/O Wait



# Long I/O wait



(c) Interrupts; long 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

# Simple Interrupt Processing

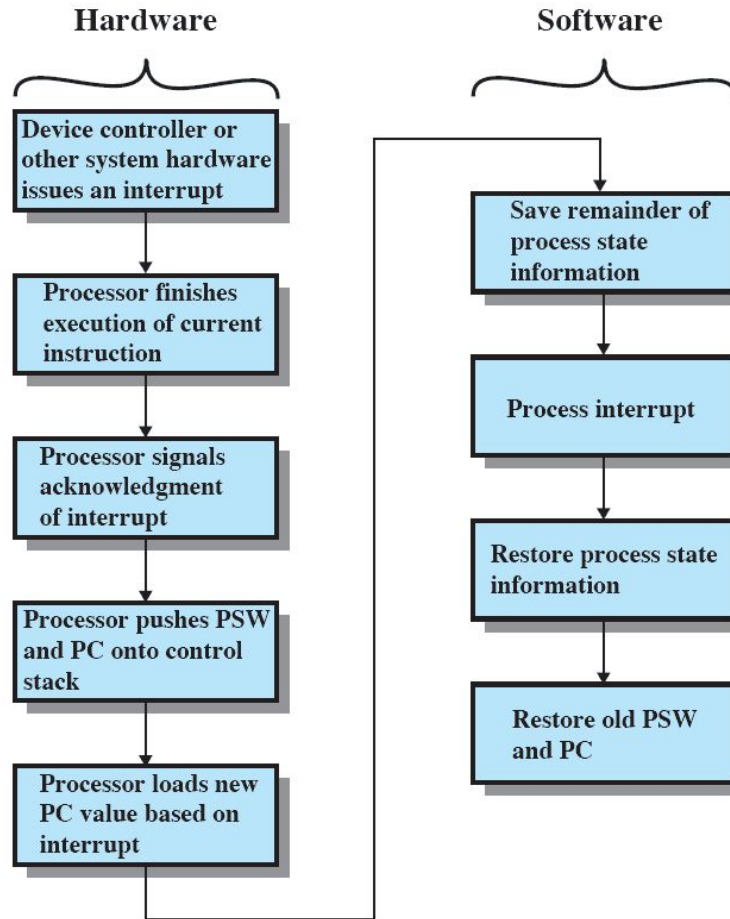
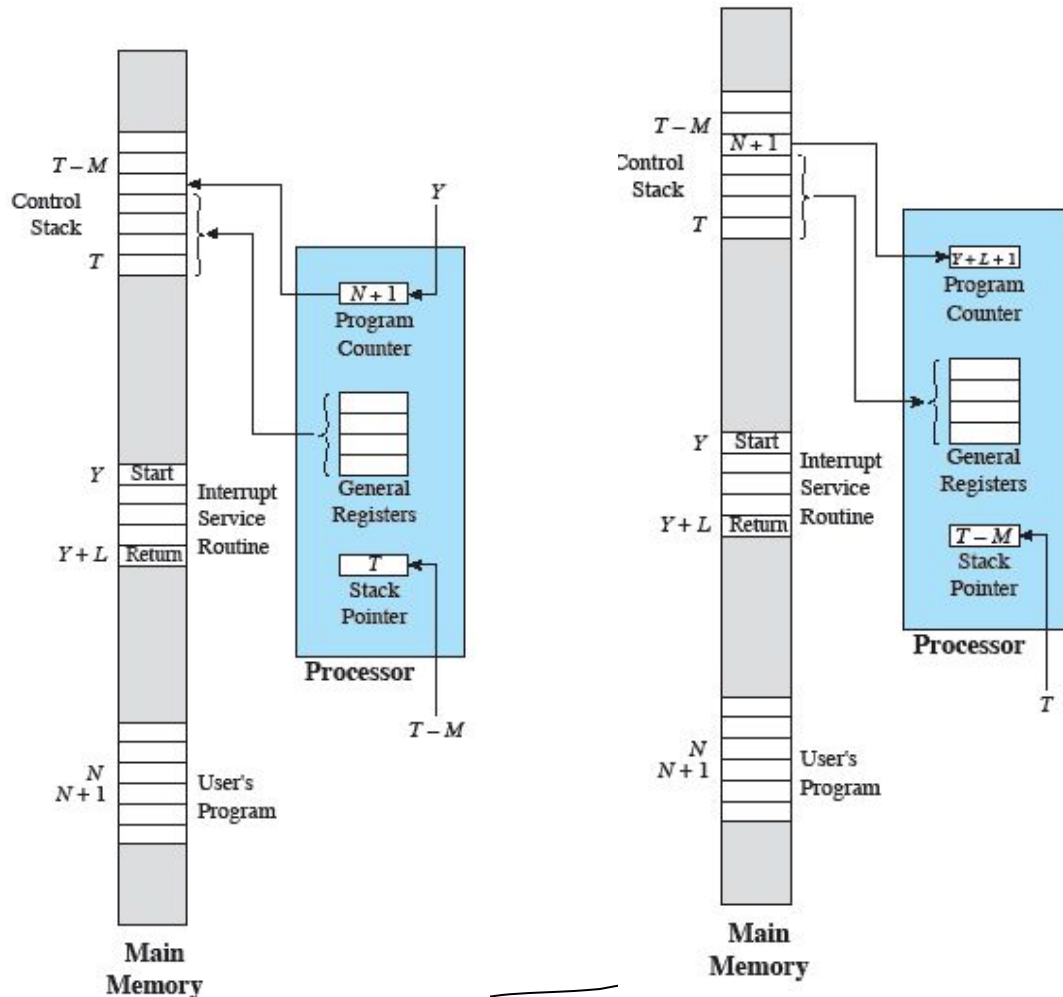


Figure 1.10 Simple Interrupt Processing

# Changes in Memory and Registers for an Interrupt



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

(b) Return from interrupt



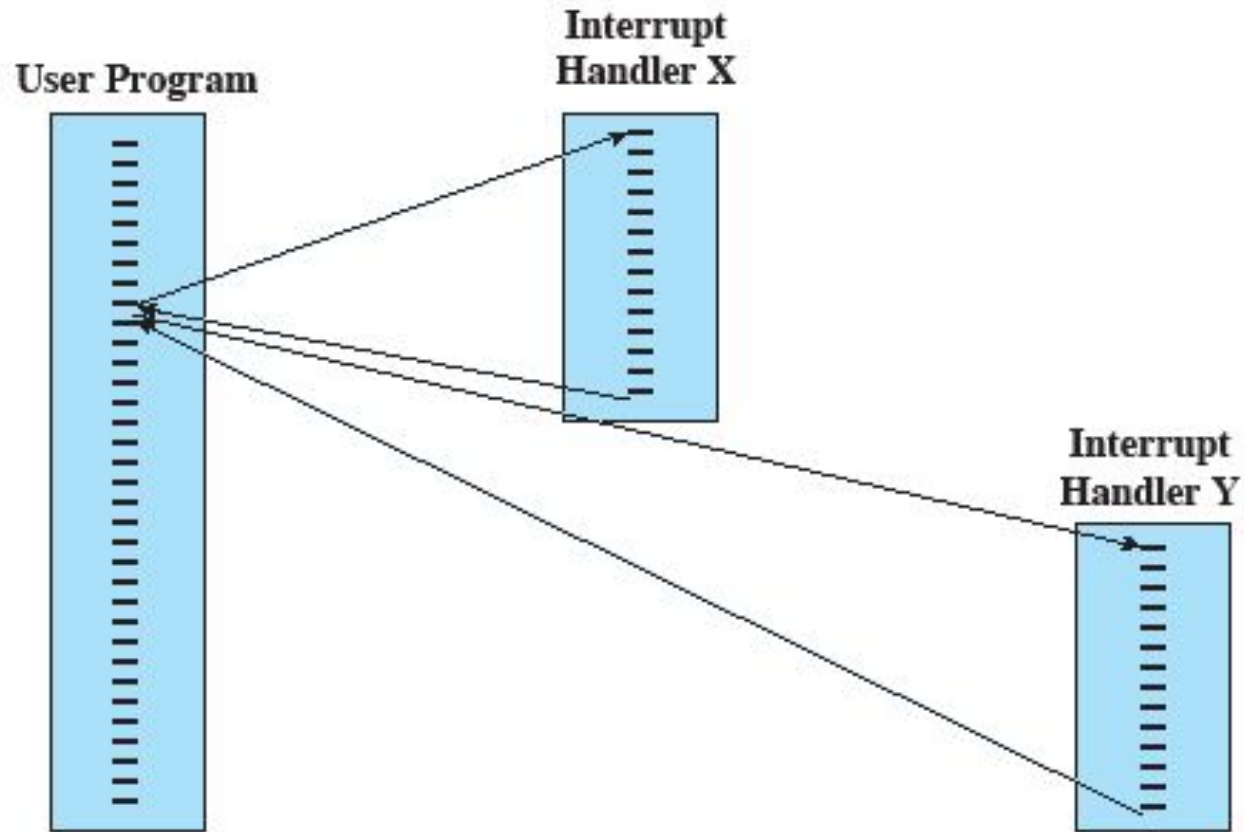


# Multiple Interrupts

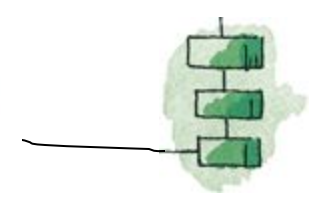
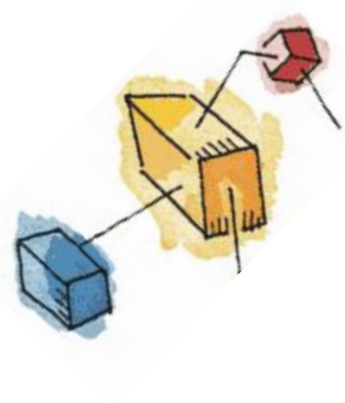
- Suppose an interrupt occurs while another interrupt is being processed.
  - E.g. printing data being received via communications line.
- Two approaches:
  - Disable interrupts during interrupt processing
  - Use a priority scheme.



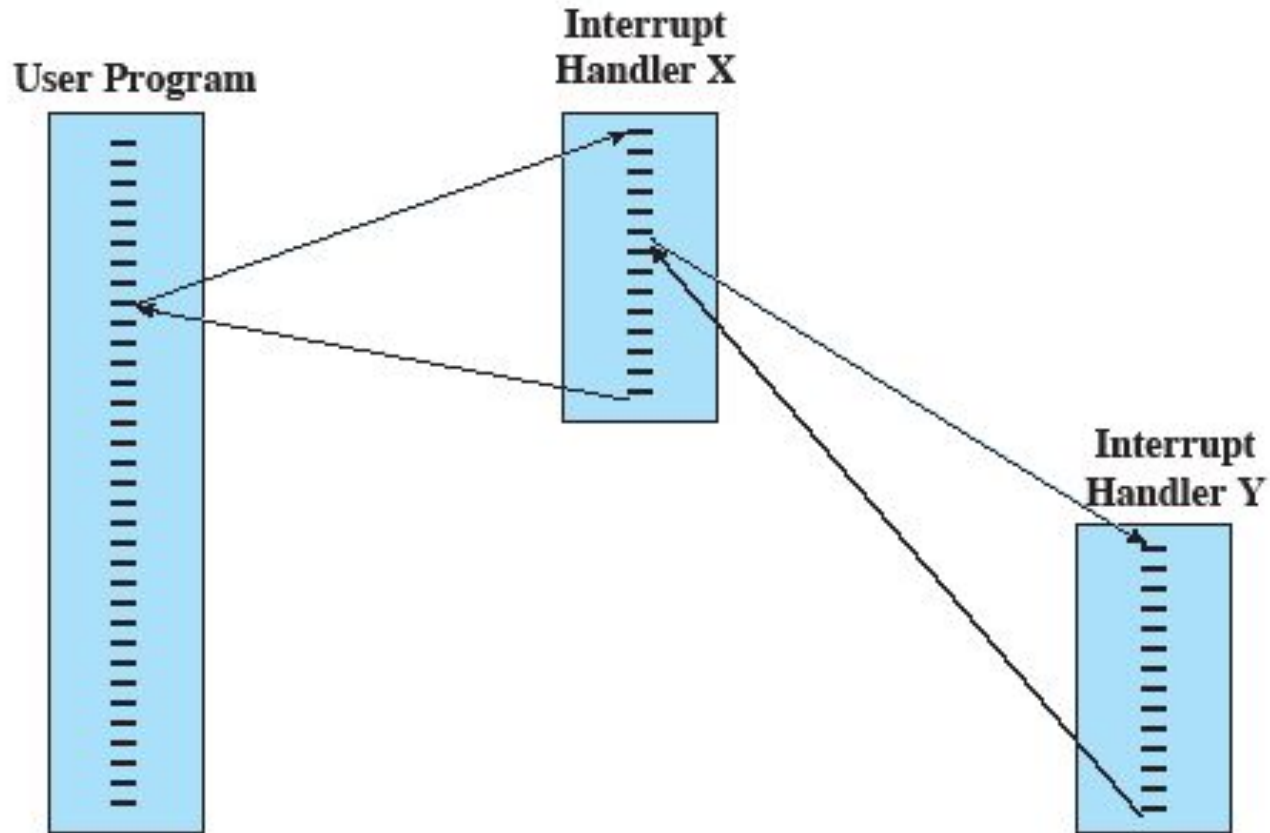
# Sequential Interrupt Processing



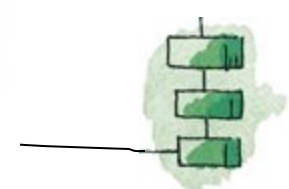
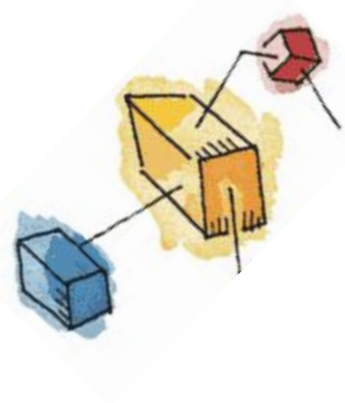
(a) Sequential interrupt processing



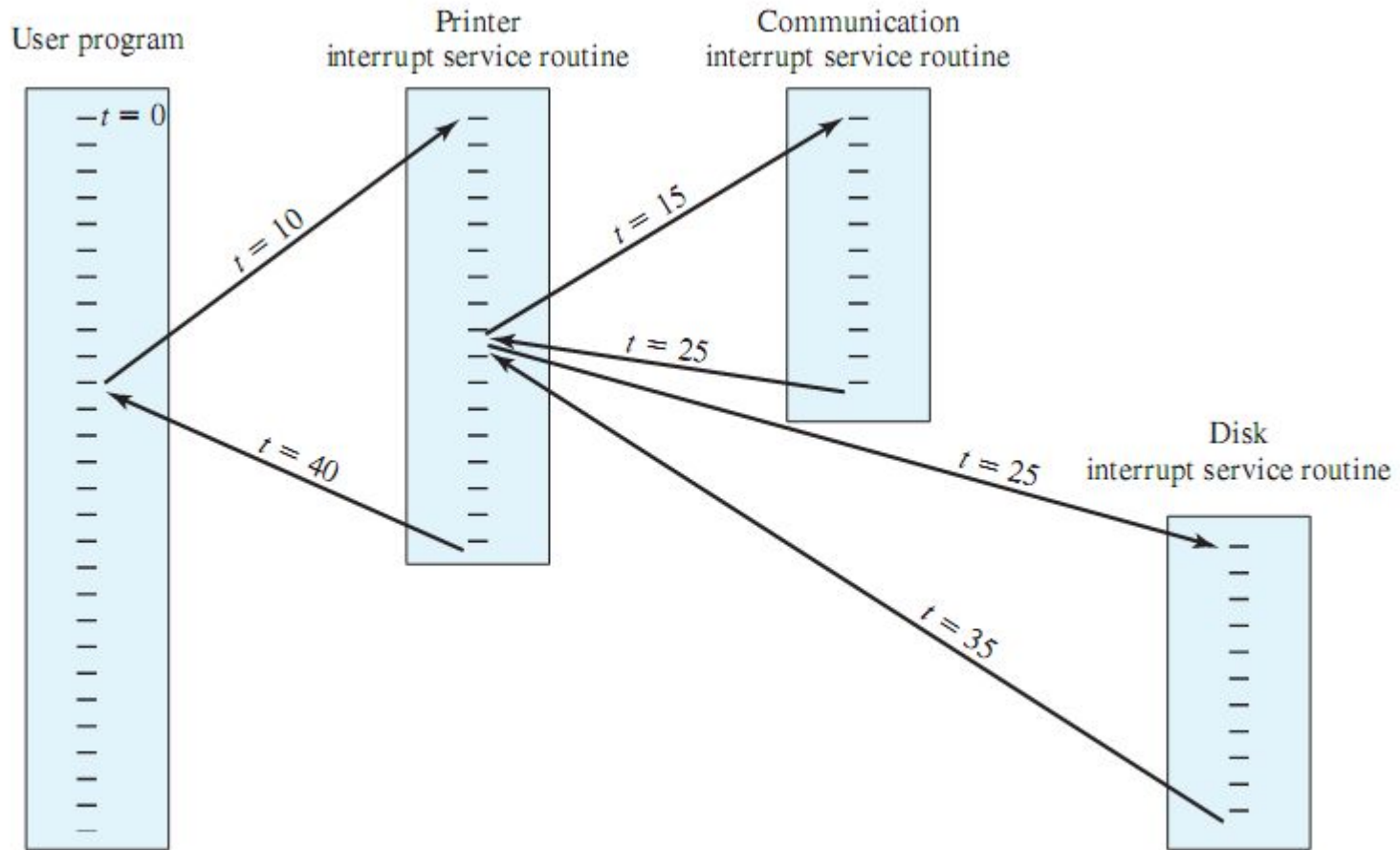
# Nested Interrupt Processing



(b) Nested interrupt processing



# Example of Nested Interrupts



**Figure 1.13** Example Time Sequence of Multiple Interrupts



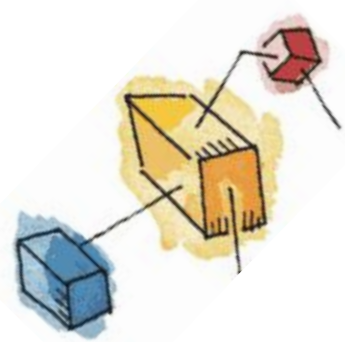
# Multiprogramming

- 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
- After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt



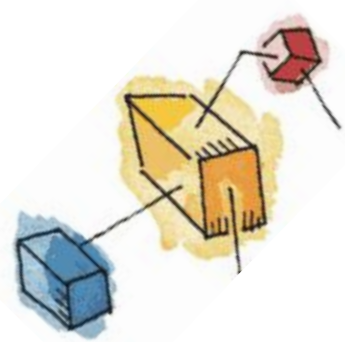
# Roadmap

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



# Memory Hierarchy

- 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



# The Memory Hierarchy

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

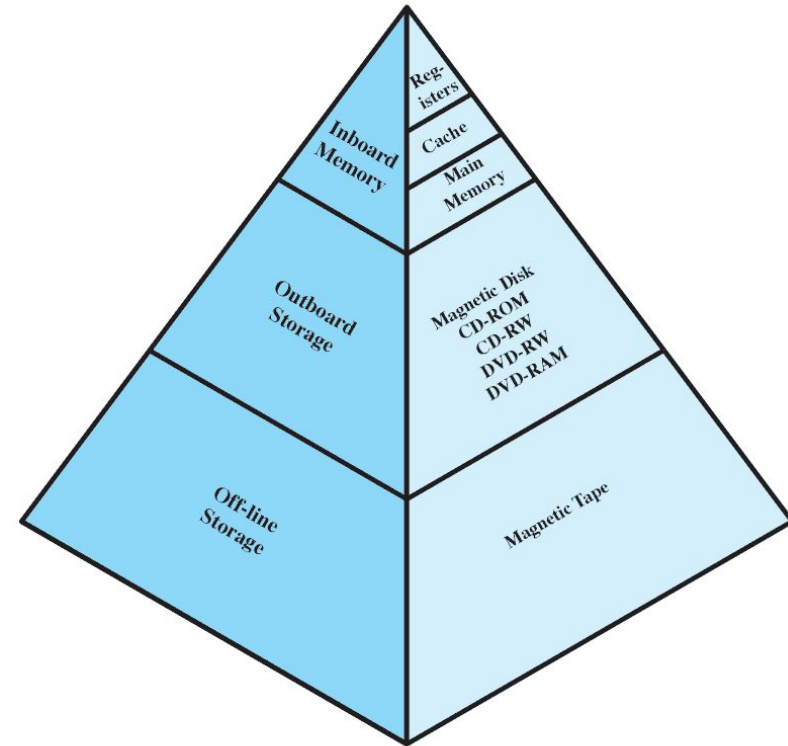


Figure 1.14 The Memory Hierarchy





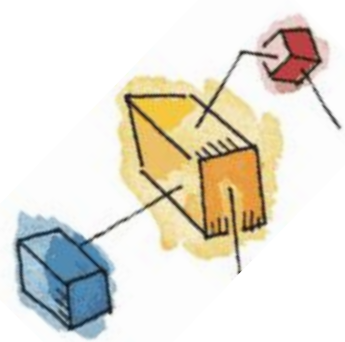
# Secondary Memory

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



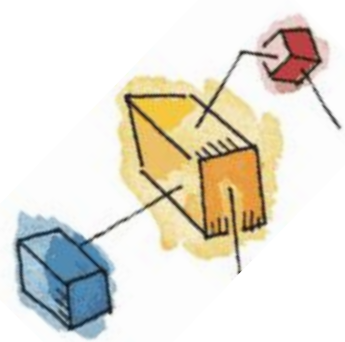
# Roadmap

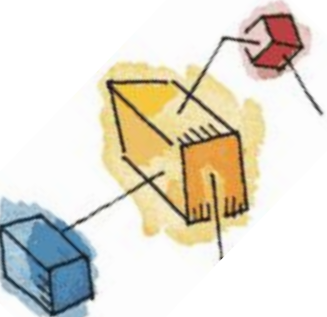
- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- **Cache Memory**
- I/O Communication Techniques



# Cache Memory

- Invisible to the OS
  - Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
  - Processor speed faster than memory access speed
- Exploit the principle of locality with a small fast memory





# Principal of Locality

- More details later but in short ...
- Data which is required soon is often close to the current data
  - If data is referenced, then it's neighbour might be needed soon.



# Cache and Main Memory

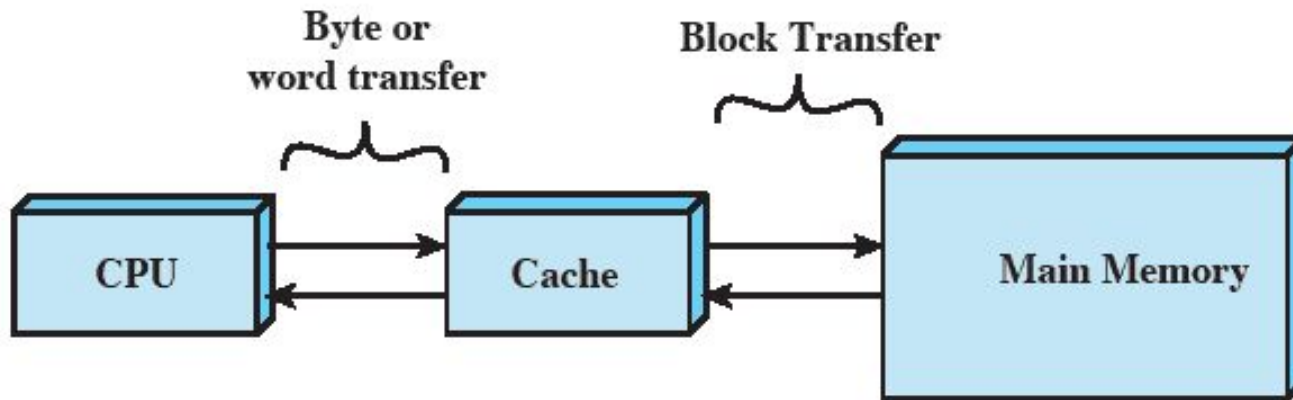
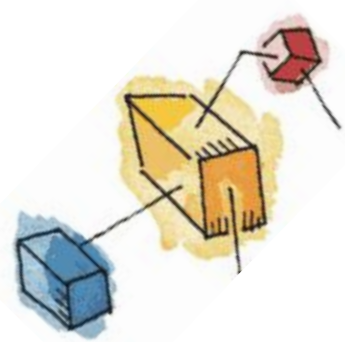


Figure 1.16 Cache and Main Memory

# Cache Principles

- Contains copy of a portion of main memory
- Processor first checks cache
  - If not found, block of memory read into cache
- Because of locality of reference, likely future memory references are in that block



# Cache/Main-Memory Structure

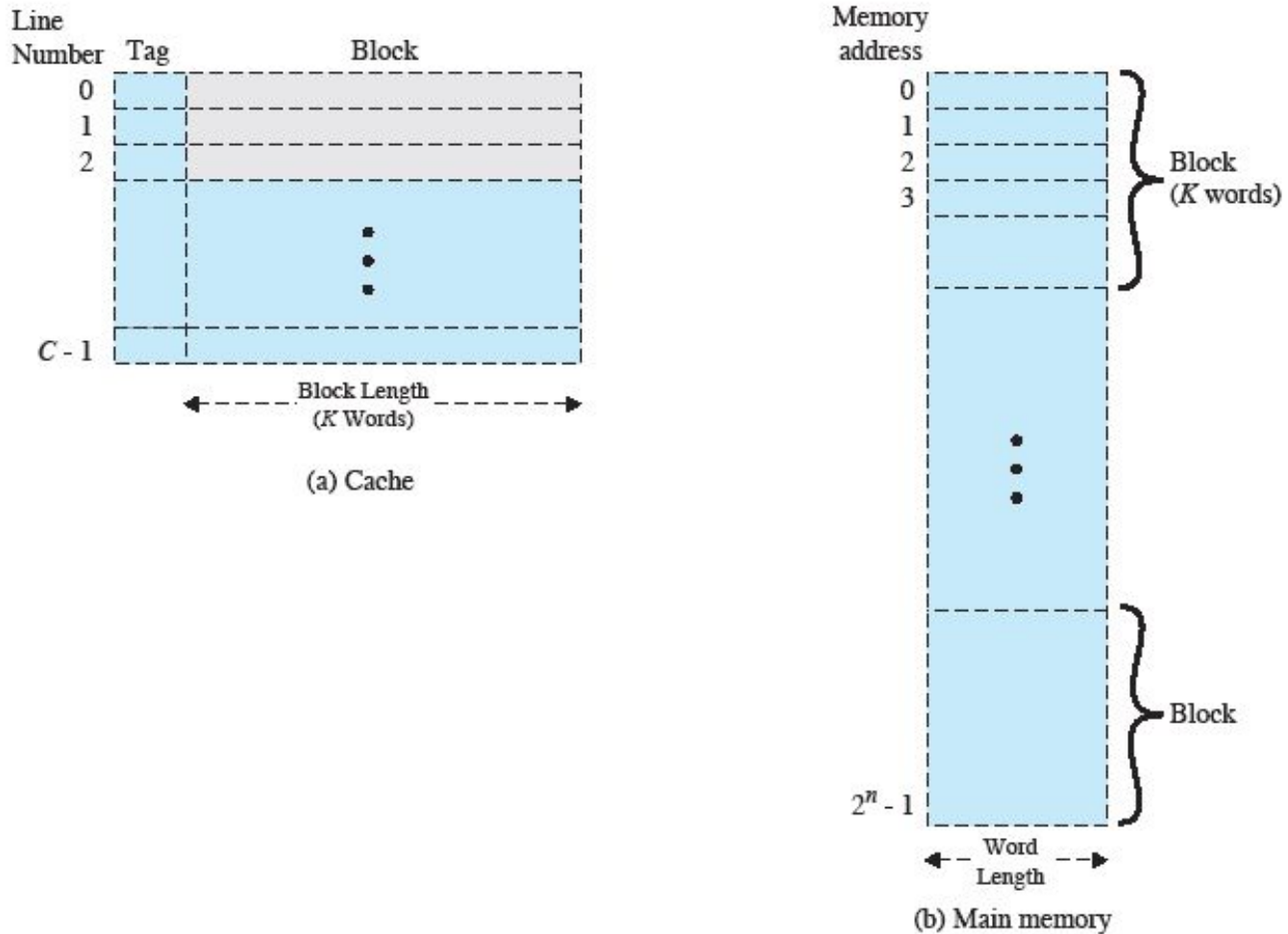


Figure 1.17 Cache/Main-Memory Structure

# Cache Read Operation

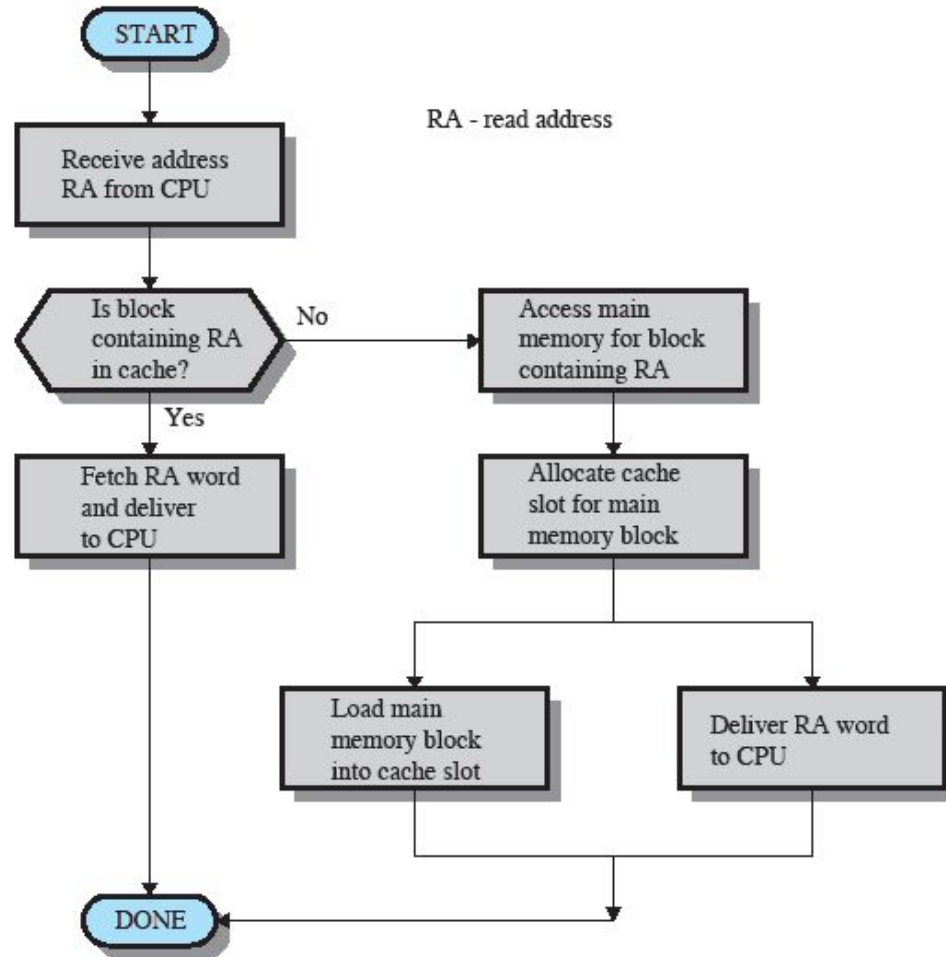
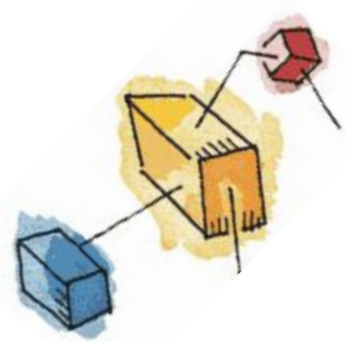


Figure 1.18 Cache Read Operation





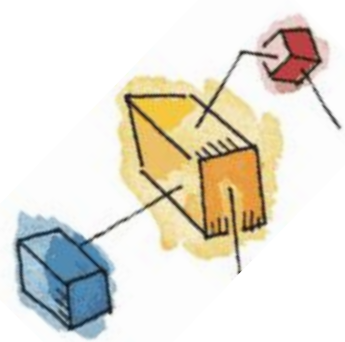
# Cache Design Issues

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



# Size issues

- 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.

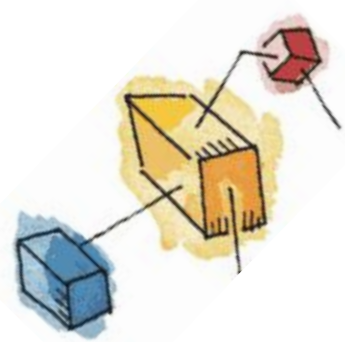




# Mapping function

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





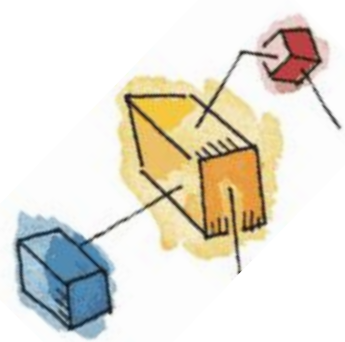
# Replacement Algorithm

- 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)



# Write policy

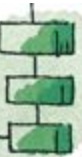
- Dictates when the memory write operation takes place
- Can occur every time the block is updated
- Can occur when the block is replaced
  - Minimize write operations
  - Leave main memory in an obsolete state





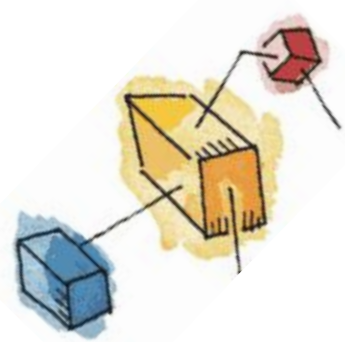
# Roadmap

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



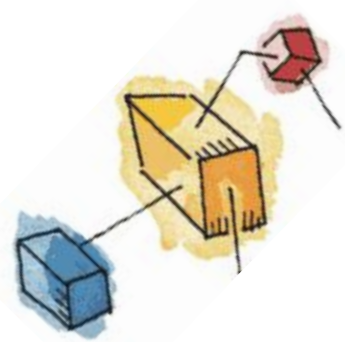
# I/O Techniques

- When the processor encounters an instruction relating to I/O,
  - it executes that instruction by issuing a command to the appropriate I/O module.
- Three techniques are possible for I/O operations:
  - Programmed I/O
  - Interrupt-driven I/O
  - Direct memory access (DMA)



# Programmed I/O

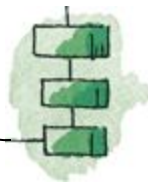
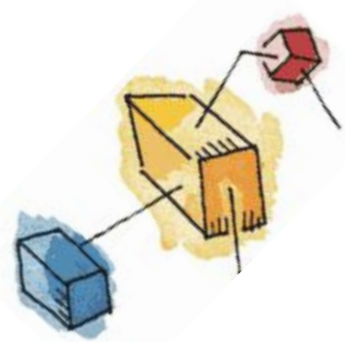
- The I/O module performs the requested action
  - then sets the appropriate bits in the I/O status register
  - but takes no further action to alert the processor.
- As there are no interrupts, the processor must determine when the instruction is complete

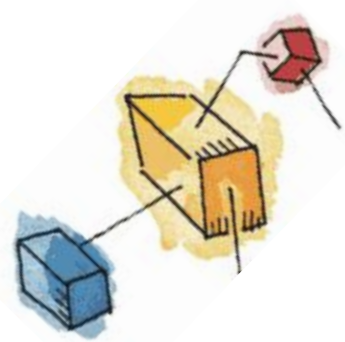




# Programmed I/O Instruction Set

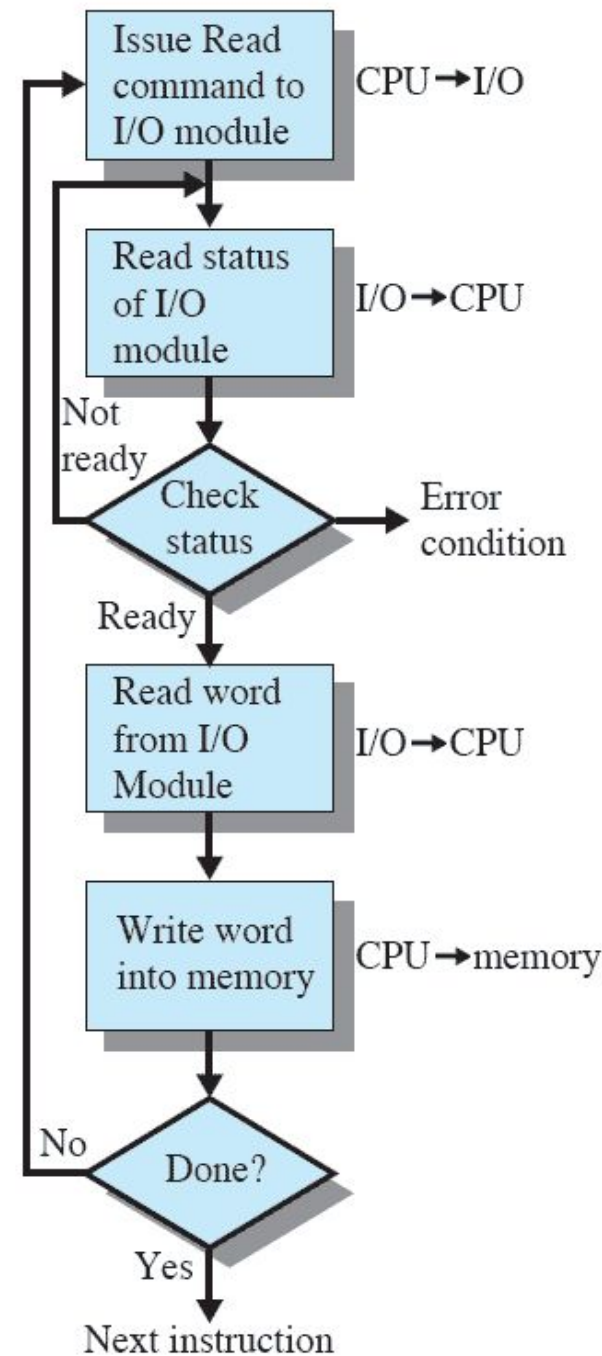
- Control
  - Used to activate and instruct device
- Status
  - Tests status conditions
- Transfer
  - Read/write between process register and device





# Programmed I/O Example

- Data read in a word at a time
  - Processor remains in status-checking look while reading



(a) Programmed I/O

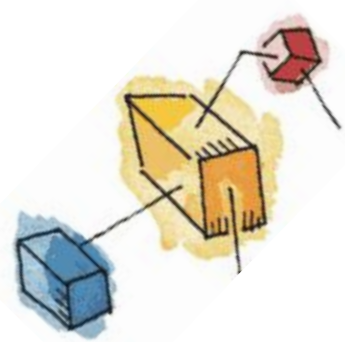




# Interrupt-Driven I/O

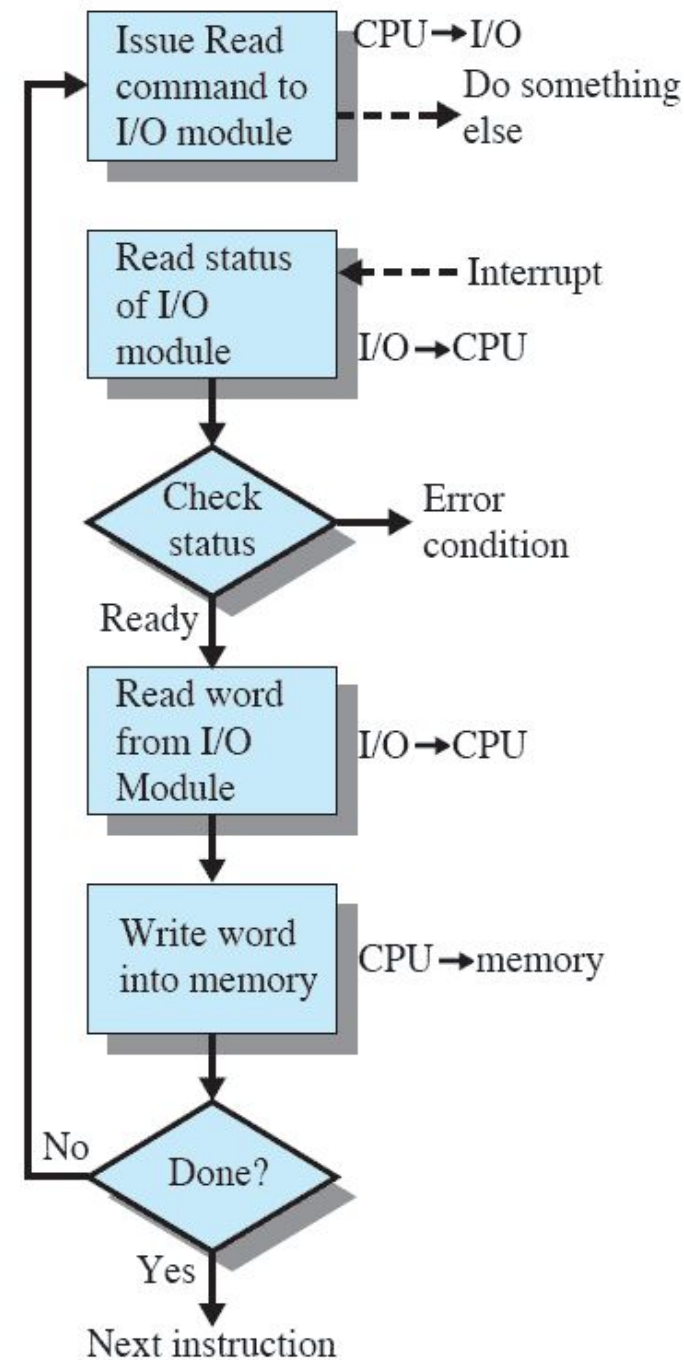
- Processor issues an I/O command to a module
  - and then goes on to do some other useful work.
- The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor.





# Interrupt-Driven I/O

- Eliminates needless waiting
  - But everything passes through processor.



(b) Interrupt-driven I/O





# Direct Memory Access

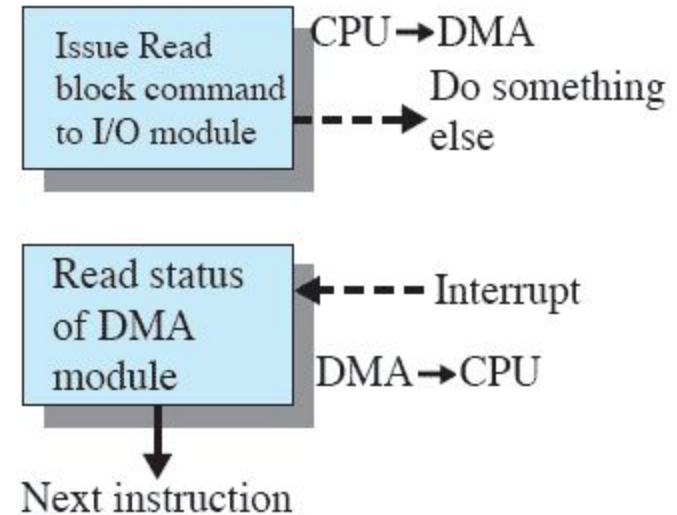
- Performed by a separate module on the system
- When needing to read/write processor issues a command to DMA module with:
  - Whether a read or write is requested
  - The address of the I/O device involved
  - The starting location in memory to read/write
  - The number of words to be read/written





# Direct Memory Access

- I/O operation delegated to DMA module
- Processor only involved when beginning and ending transfer.
- Much more efficient.



(c) Direct memory access

