Computer System Overview

Chapter 1

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

Basic Elements

- Processor
- Main Memory
  - volatile
  - referred to as real memory or primary memory
- I/O modules
  - secondary memory devices
  - communications equipment
  - terminals
- System bus
  - communication among processors, memory, and I/O modules

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
- Two I/O registers (peripherials)
  - I/O address register
  - I/O buffer register

Top-Level Components

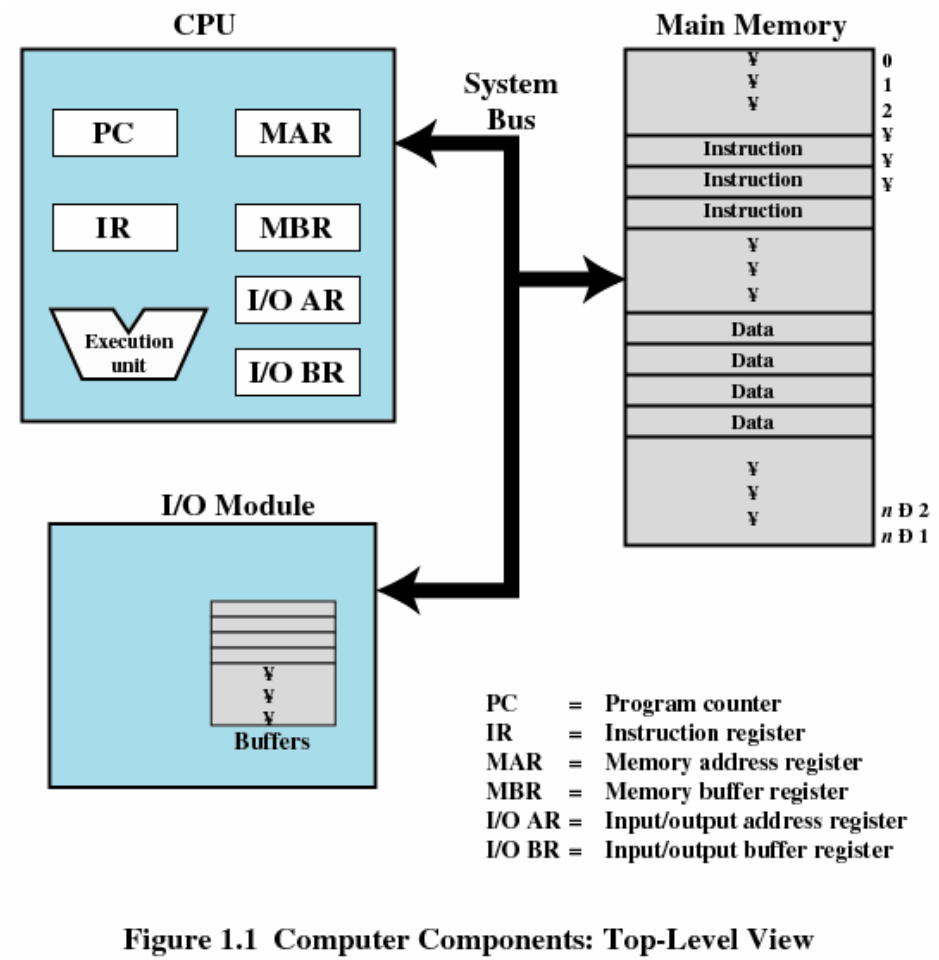


Figure 1.1 Computer Components: Top-Level View

General & Special Purpose 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 operating-system 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
  - 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

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

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

Instruction Execution

- Two steps
  - Processor reads instructions from memory
    - Fetches
  - Processor executes each instruction

Instruction Cycle

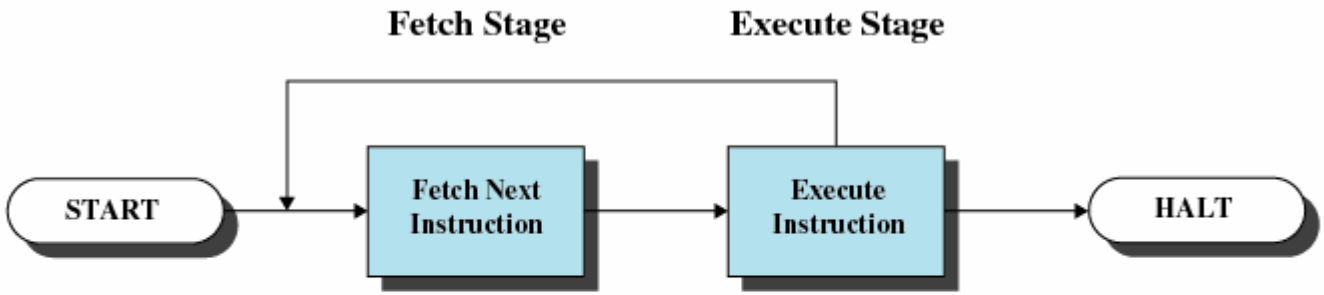


Figure 1.2 Basic Instruction Cycle

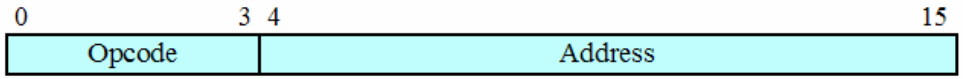
Instruction Fetch and Execute

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

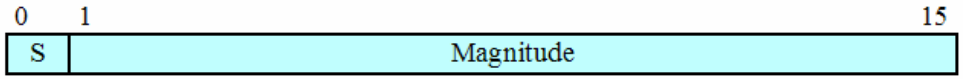
Instruction Register

- Fetched instruction is placed in the instruction register
- Instruction types (categories)
  - Processor-memory
    - Transfer data between processor and memory
  - Processor-I/O
    - Data transferred to or from a peripheral device
  - Data processing
    - Arithmetic or logic operation on data
  - Control
    - Alter sequence of execution

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

1940:
Acc <- Mem [940]

5941:
Acc <- Acc + Mem [941]

2941:
Mem [941] <- Acc

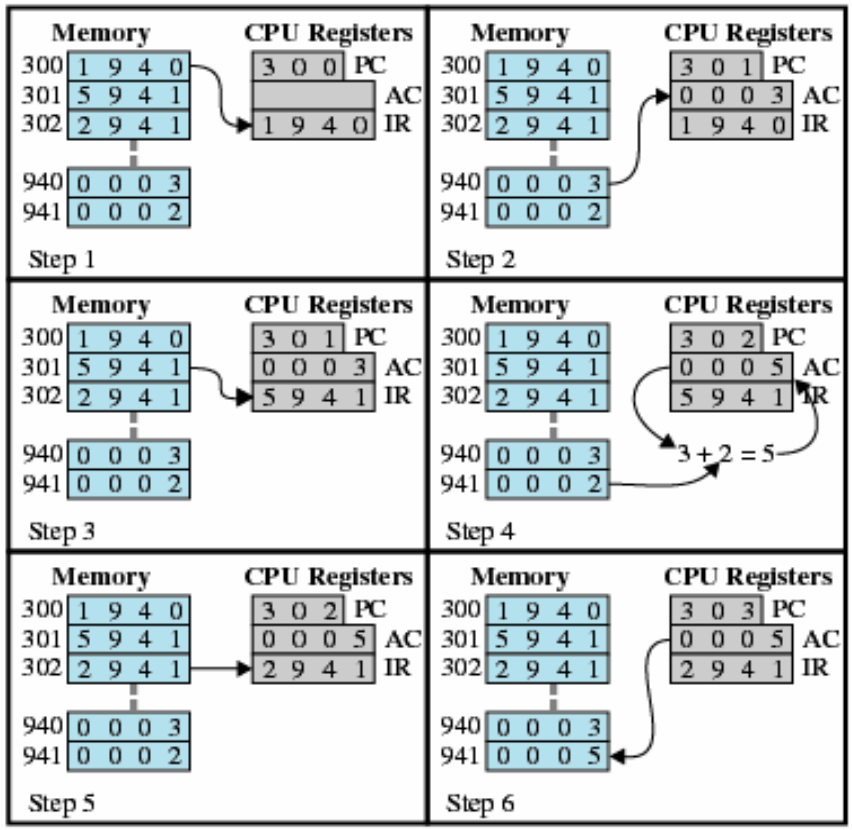


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

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

Interrupts

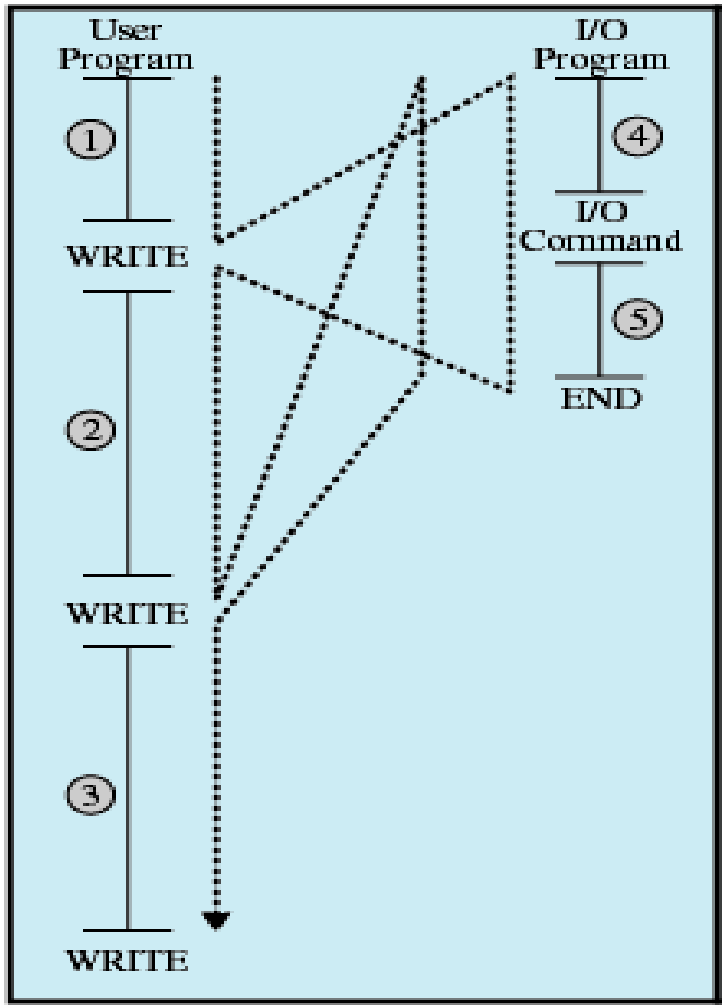
- Interrupt the normal sequencing of the processor
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device

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 Without Interrupts

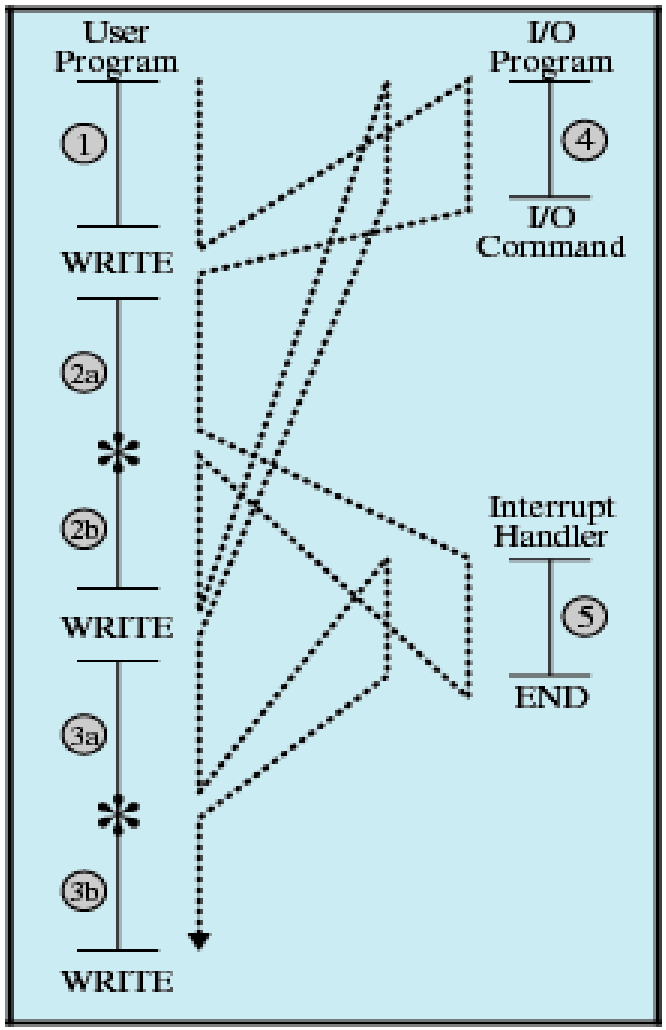


(a) No interrupts

Program waits "busy waits" for Data to transfer (between 4 & 5)

1 – 4 –*– 5 – 2 – 4 –*– 5 – 3

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

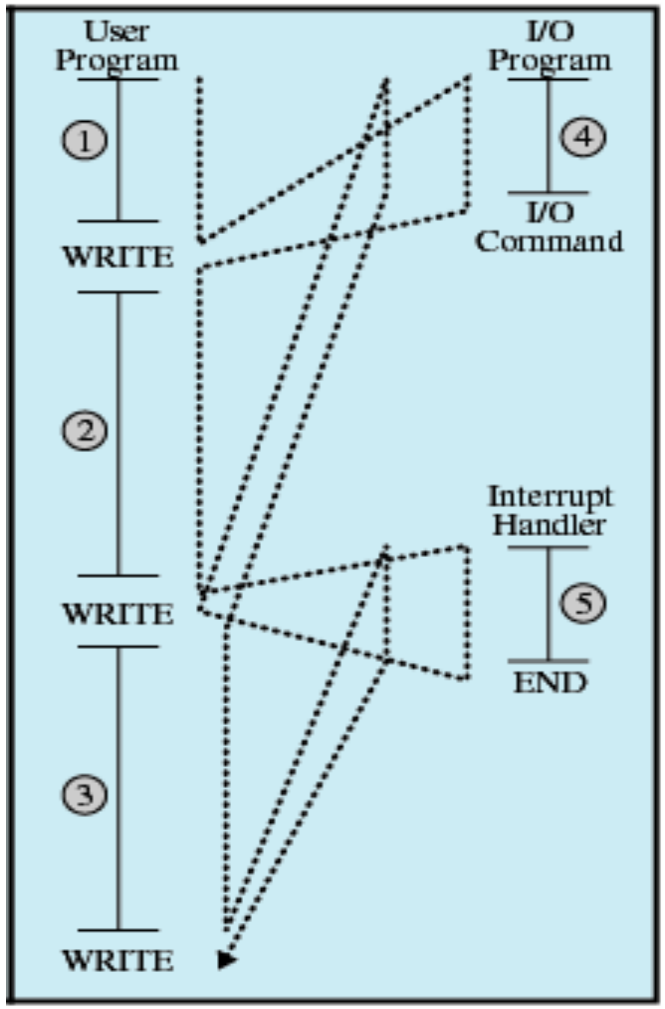


(b) Interrupts; short I/O wait

1 – 4 – 2a – 5 – 2b – 4 – 3a – 5 – 3b

Program execution and data transfer overlap at 2a and 3a

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



(c) Interrupts; long I/O wait

Program execution blocks at 2nd Write because I/O controller is busy

1 – 4 – 2 – 5 – 4 – 3 – 5

Program execution overlaps with data transfer at 2 and 3

Interrupt Handler

- Program to service a particular I/O device
- Generally part of the operating system

Interrupts

- Suspends the normal sequence of execution

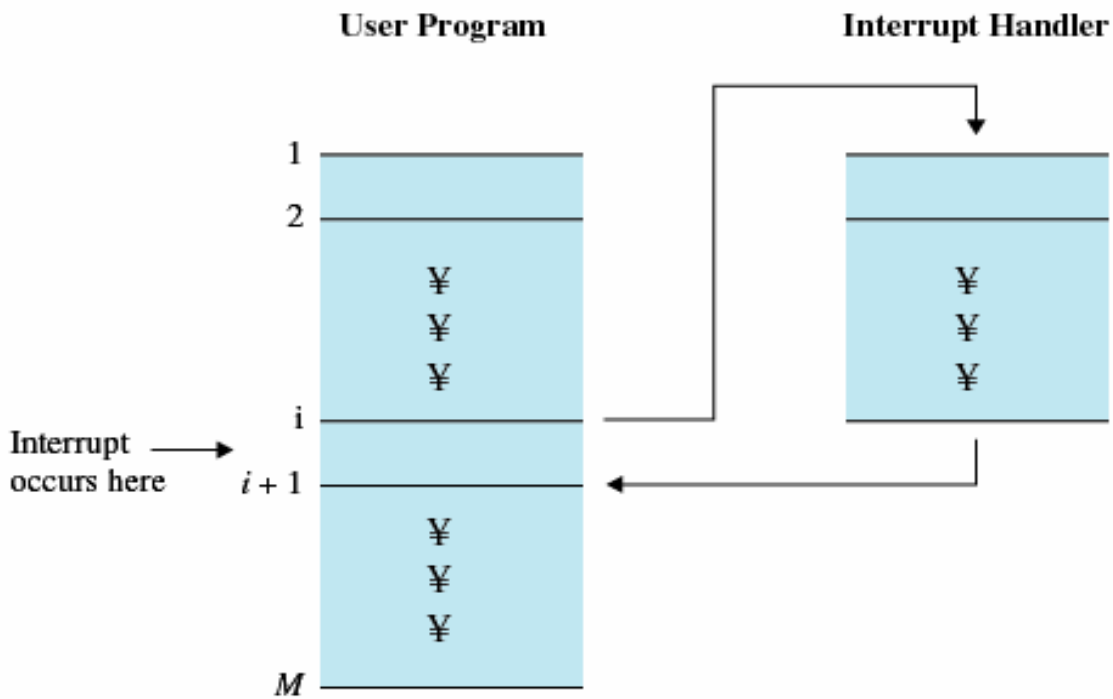


Figure 1.6 Transfer of Control via Interrupts

Interrupt Cycle

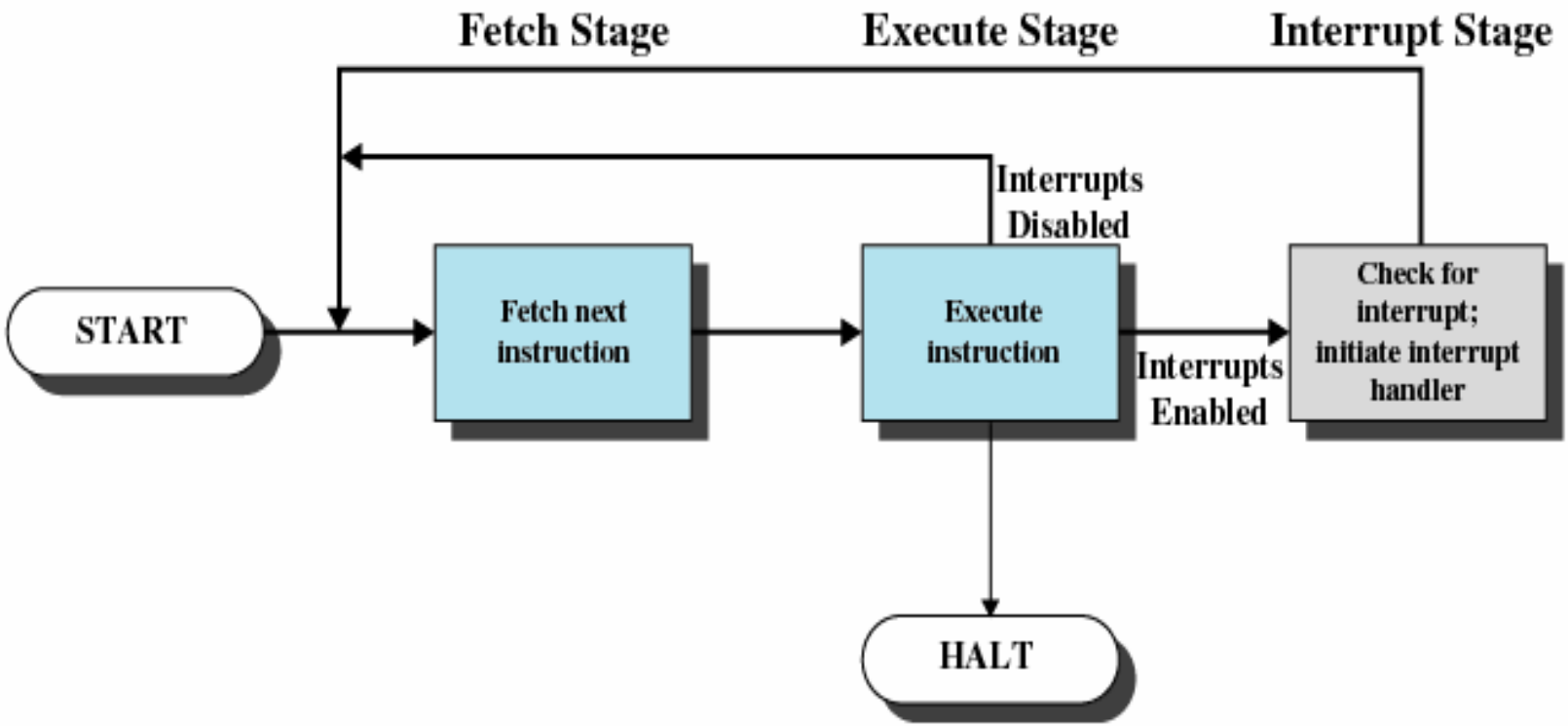
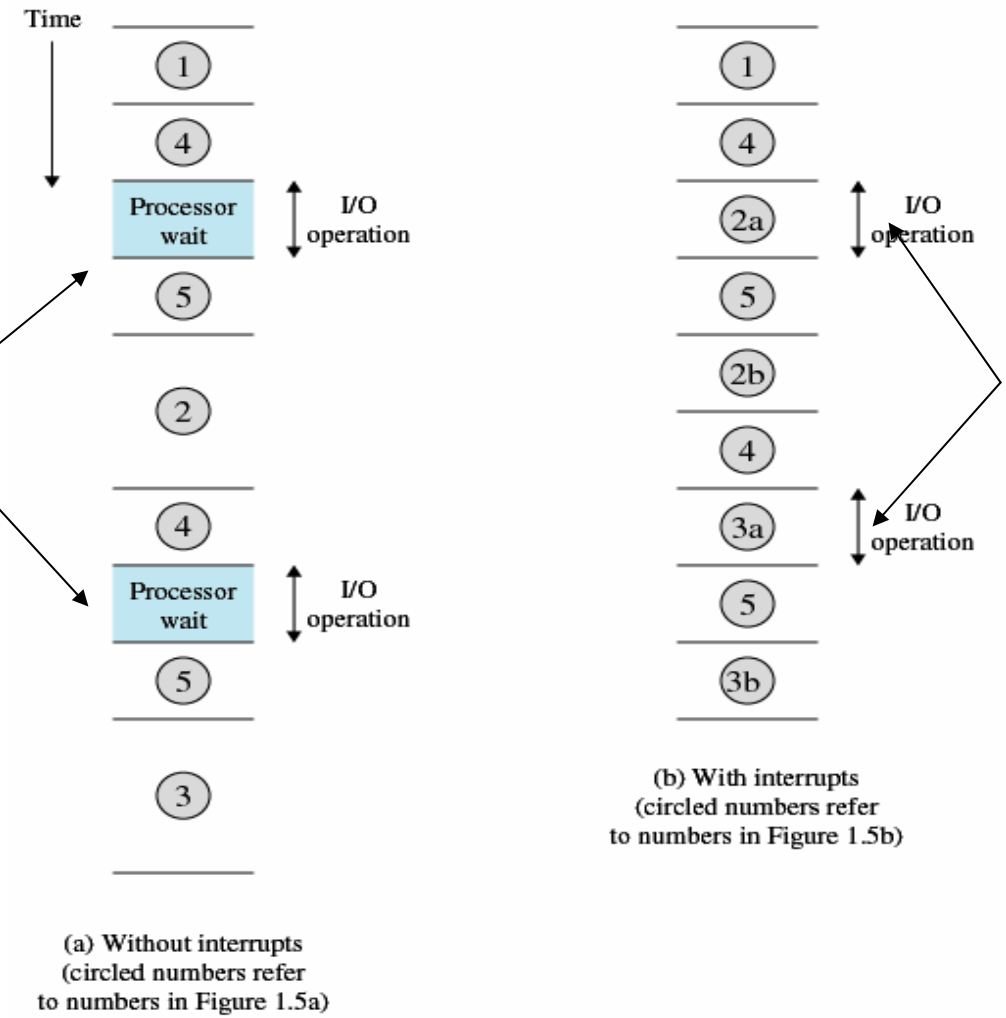


Figure 1.7 Instruction Cycle with Interrupts

Interrupt Cycle

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

Timing Diagram Based on Short I/O Wait



Program
"busy wait"

Concurrency:
Data transfer and
Program execution

Figure 1.8 Program Timing: Short I/O Wait

Timing Diagram Based on Long I/O Wait

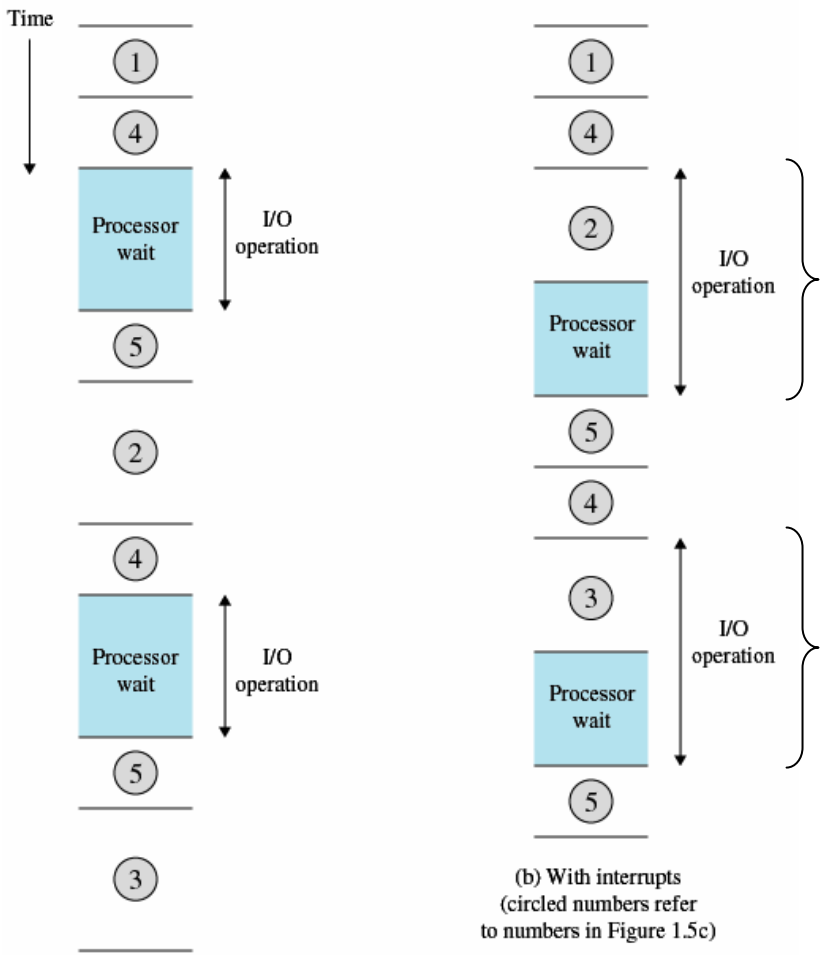


Figure 1.9 Program Timing: Long I/O Wait

Assumption:

Cannot issue a 2nd "write" until 1st "write" finishes

Data transfer time

Pgm segment 2 completes before Data transfer completes

Data transfer time

Pgm segment 3 completes before Data transfer completes

Simple Interrupt Processing

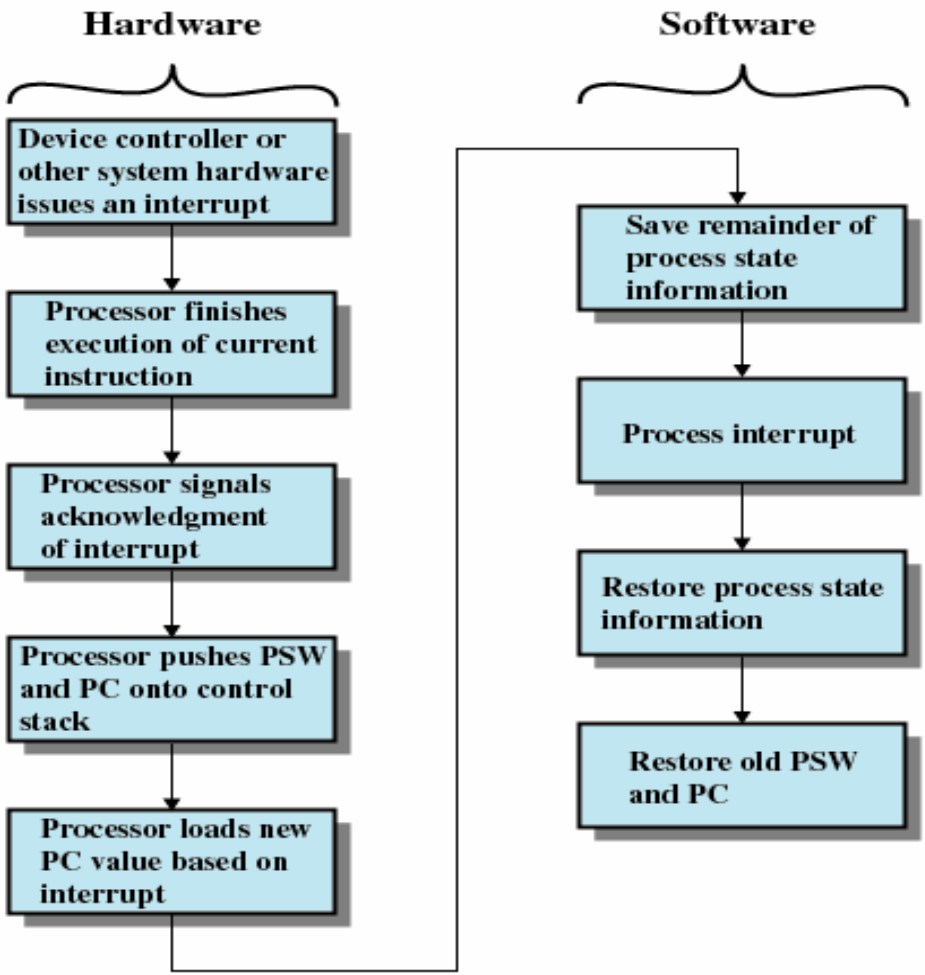
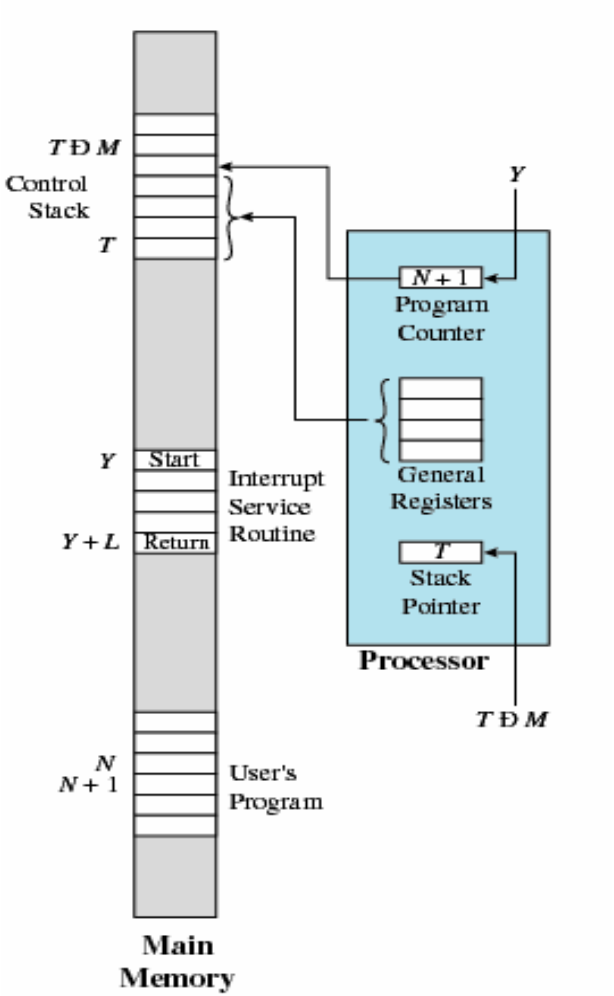


Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt

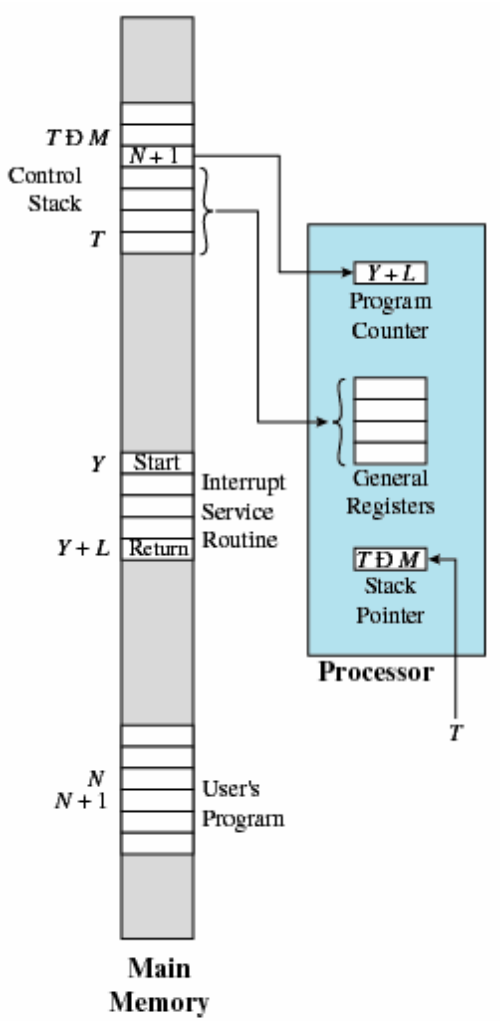


(a) Interrupt occurs after instruction at location *N*

Program execution environment (PC, Gen Regs, Stack Ptr) is saved on control stack

Processor set up to execute Interrupt Service Routine
PC <- Y, SP <- TDM…
(and other things)

Changes in Memory and Registers for an Interrupt



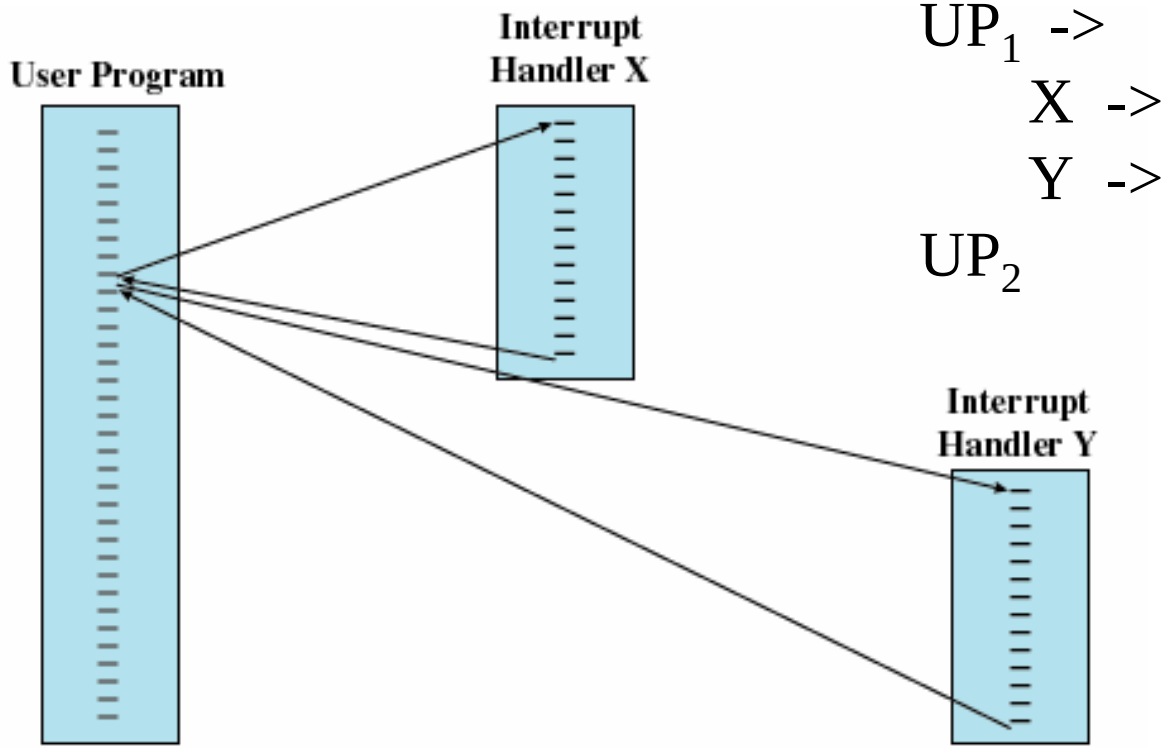
(b) Return from interrupt

After Interrupt Handler finishes
Restart user program execution:

PC <- N+1
General Regs restored
SP <- T

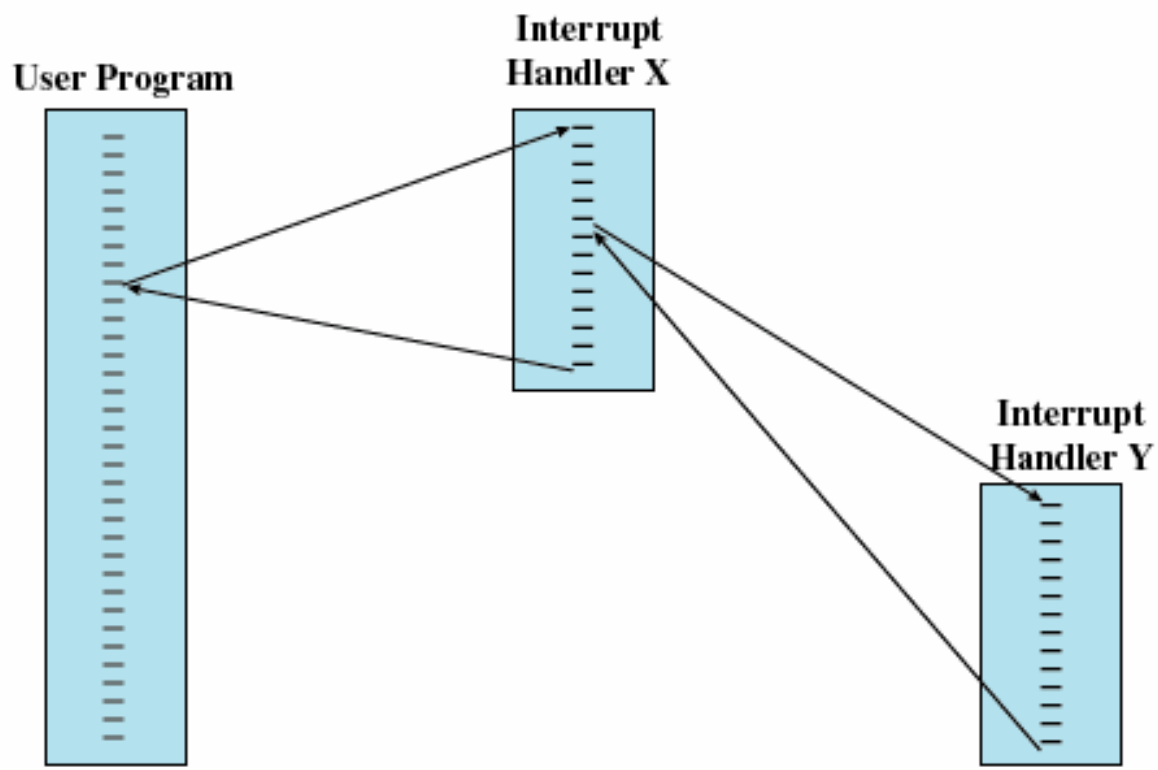
Handling Multiple Interrupts: Approach 1

- Disable interrupts while an interrupt is being processed



Handling Multiple Interrupts Approach 2

- Define priorities for interrupts



(b) Nested interrupt processing

Up_1

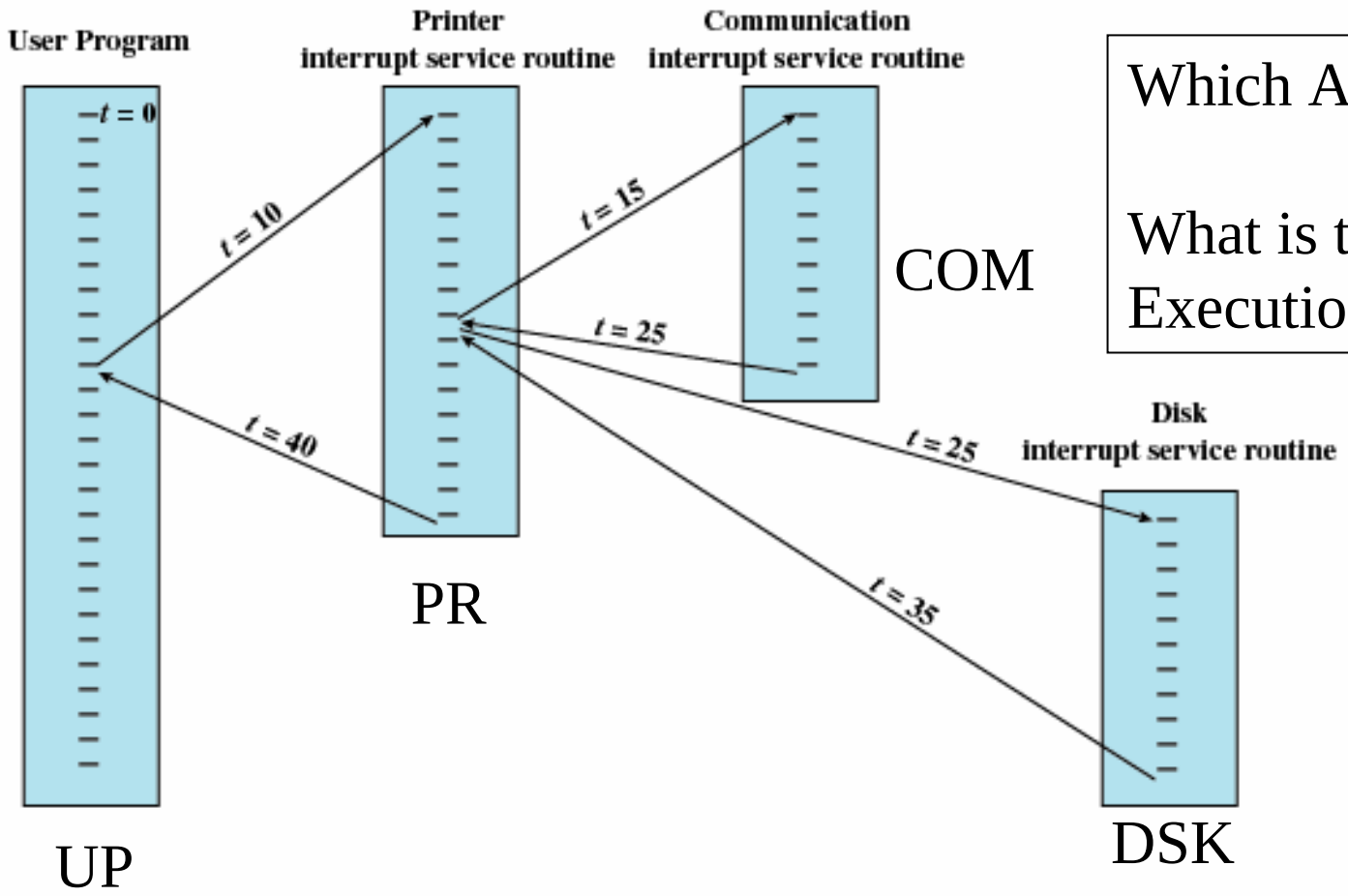
X_1

Y

X_2

Up_2

Multiple Interrupts



Which Approach?

What is the Execution Sequence?

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

Memory Hierarchy

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed

Memory Hierarchy

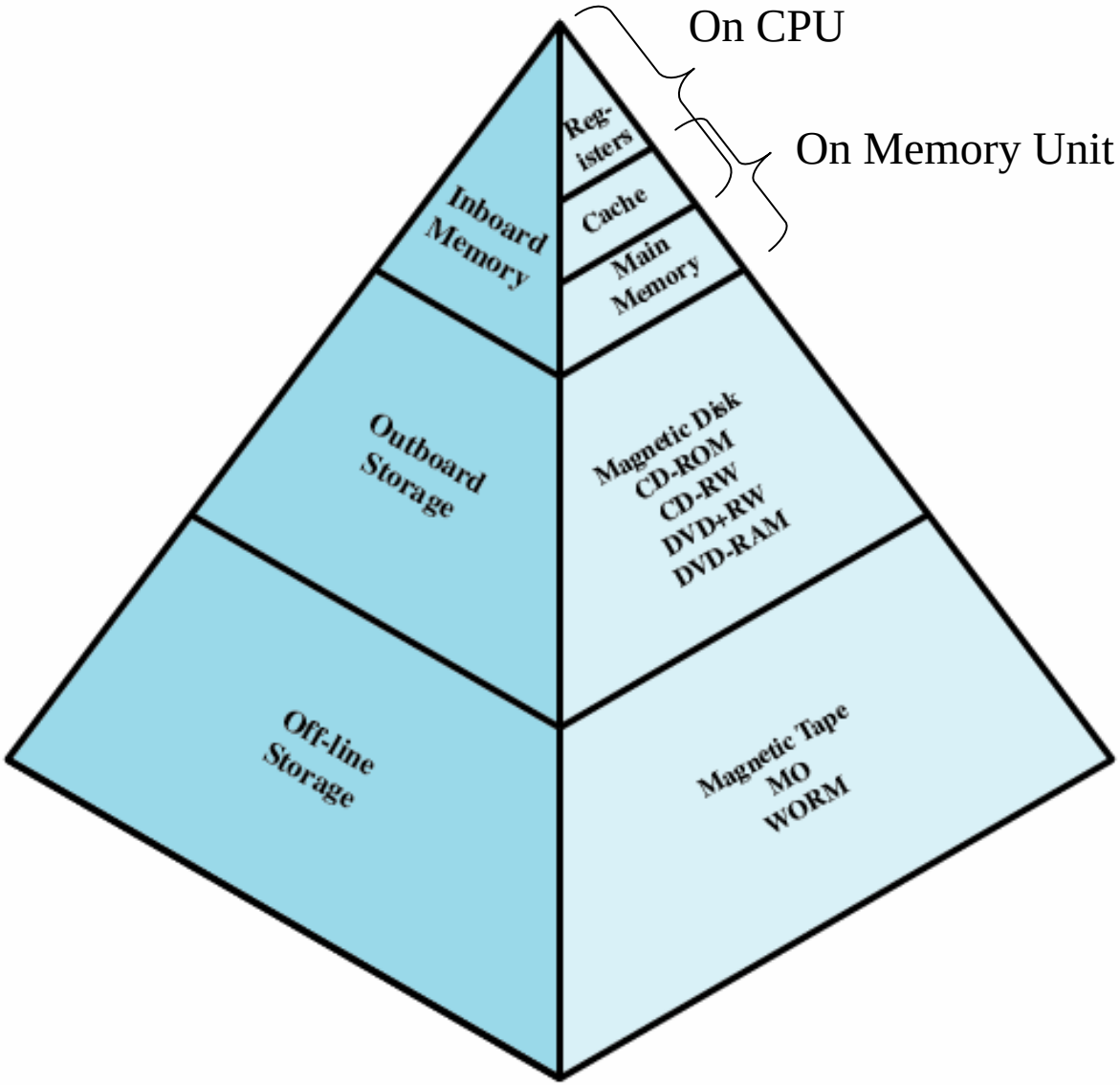


Figure 1.14 The Memory Hierarchy

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

Secondary Memory

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

Disk Cache

- A portion of main memory used as a buffer to temporarily to hold data for the disk
- Disk read/writes exhibit address clustering
  - Successive/multiple accesses to same data structure or set of instructions (locality)
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk

Memory Cache

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

Cache Memory

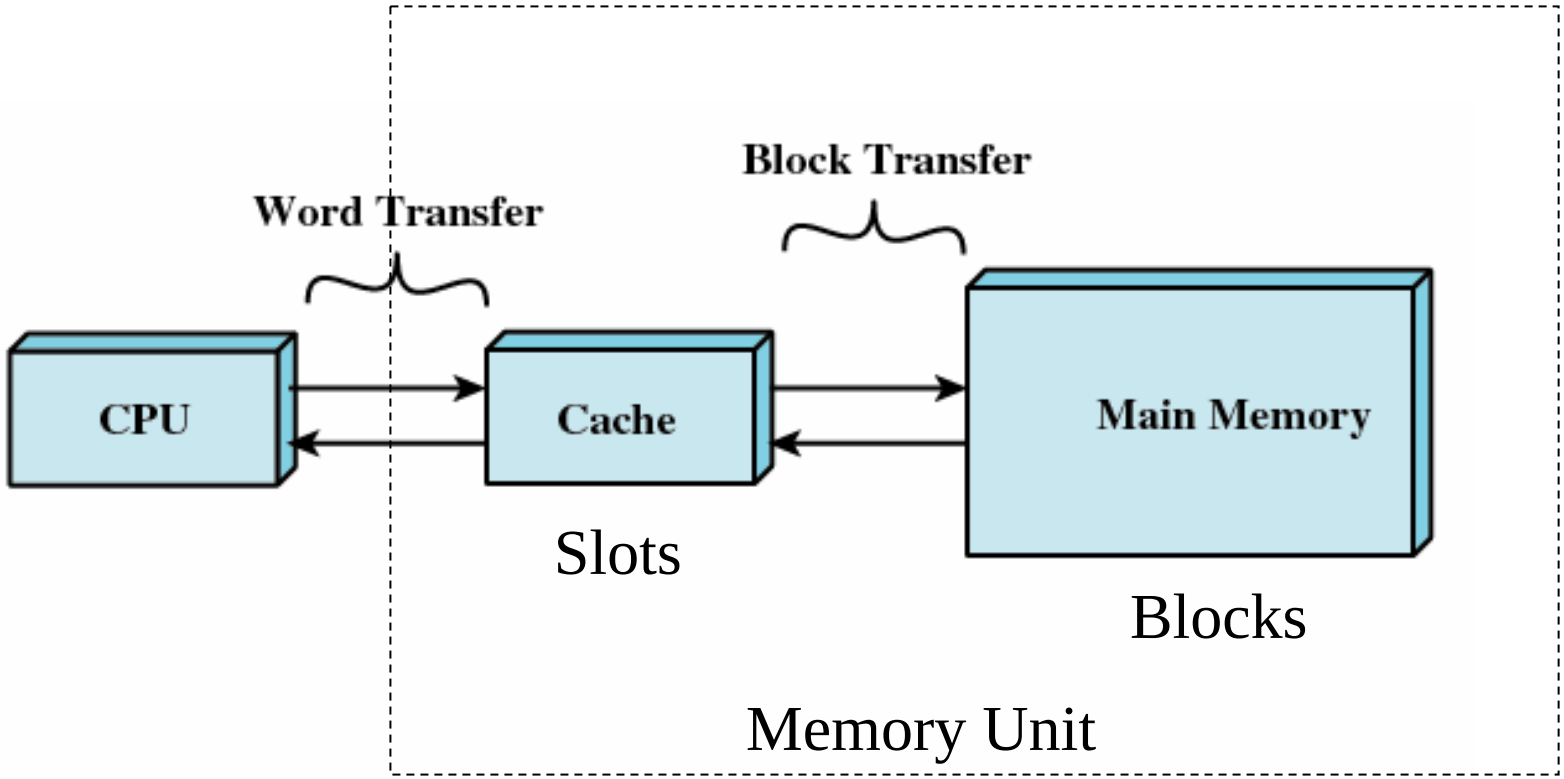


Figure 1.16 Cache and Main Memory

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

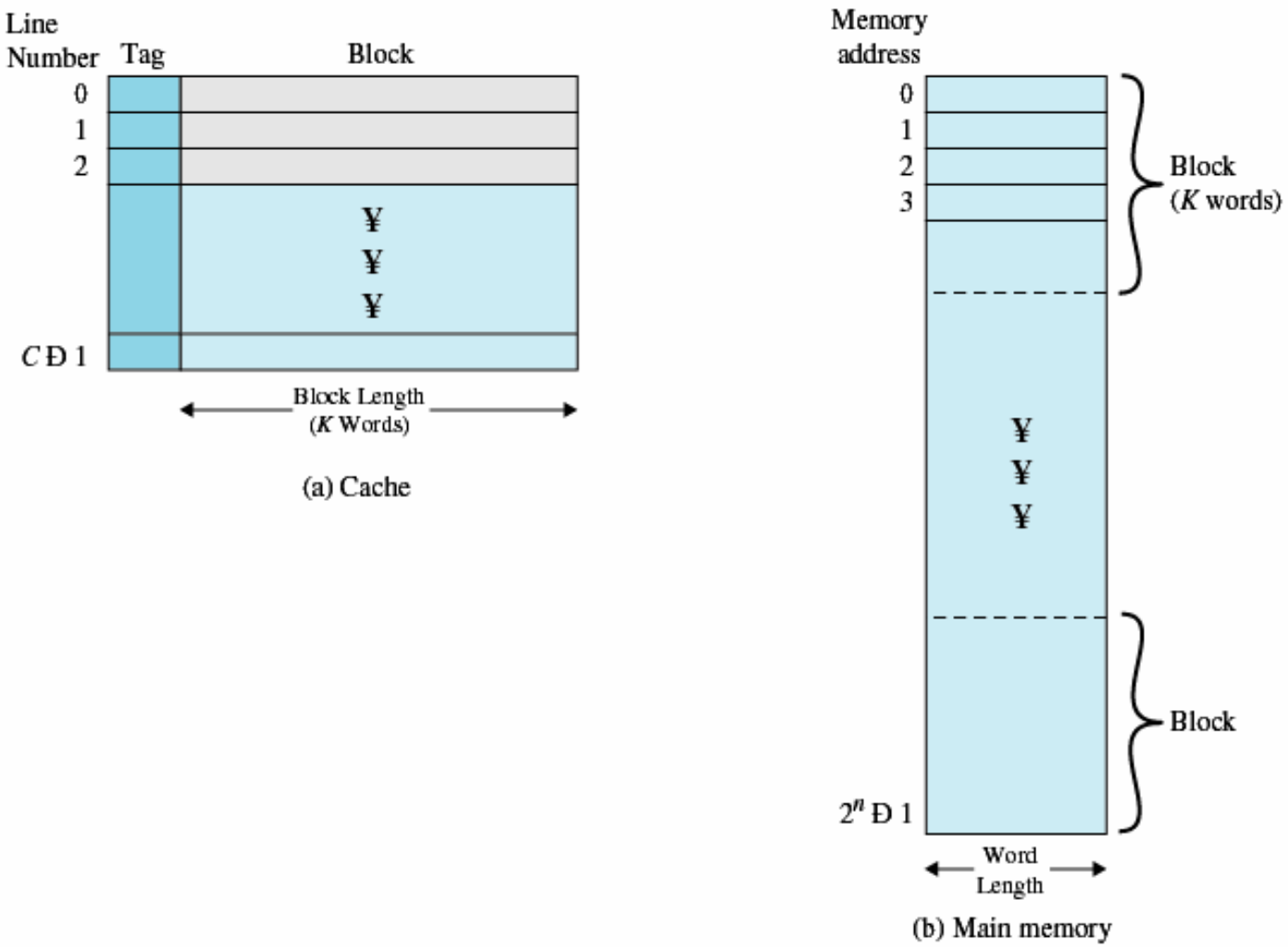


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation

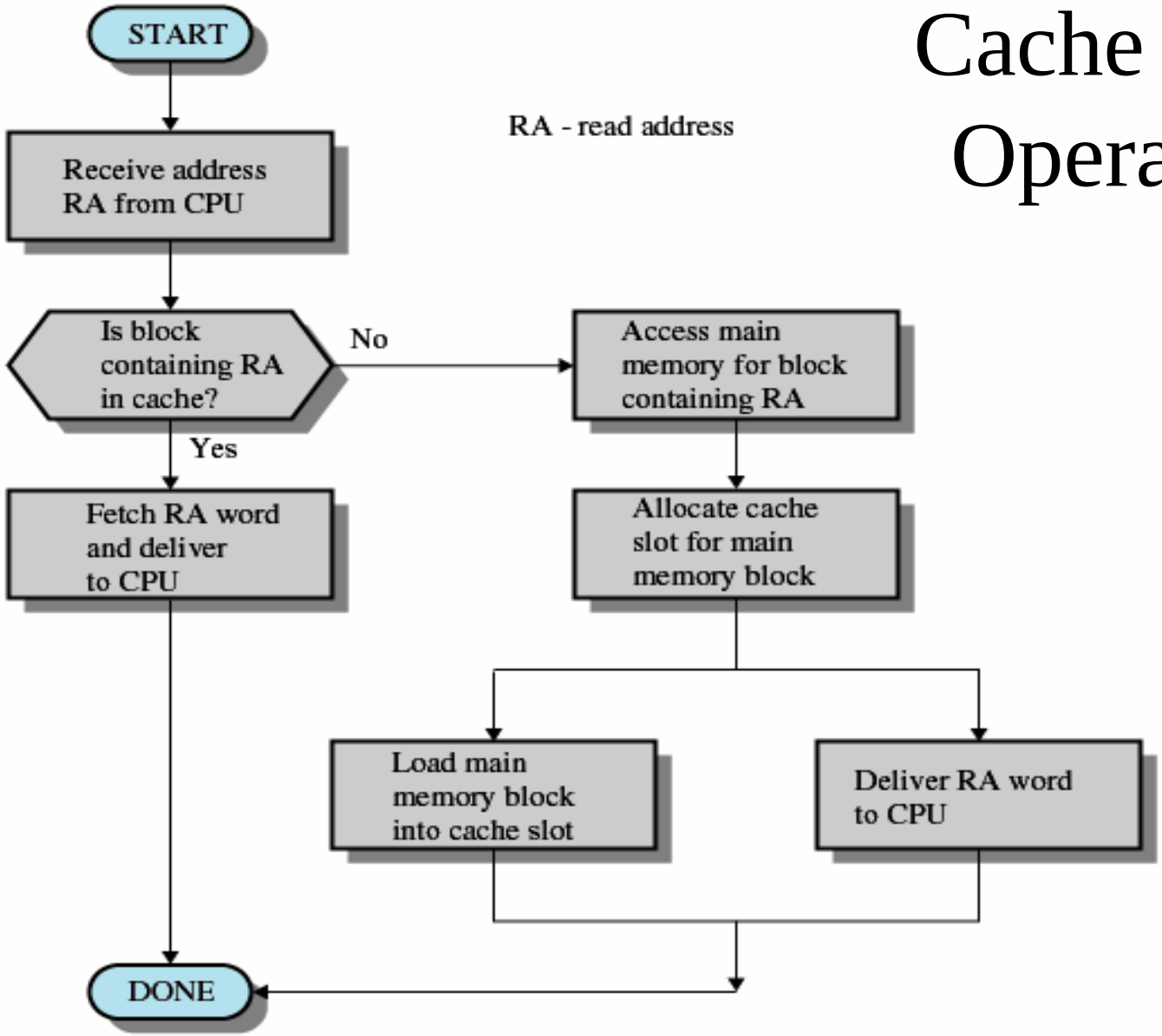


Figure 1.18 Cache Read Operation

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

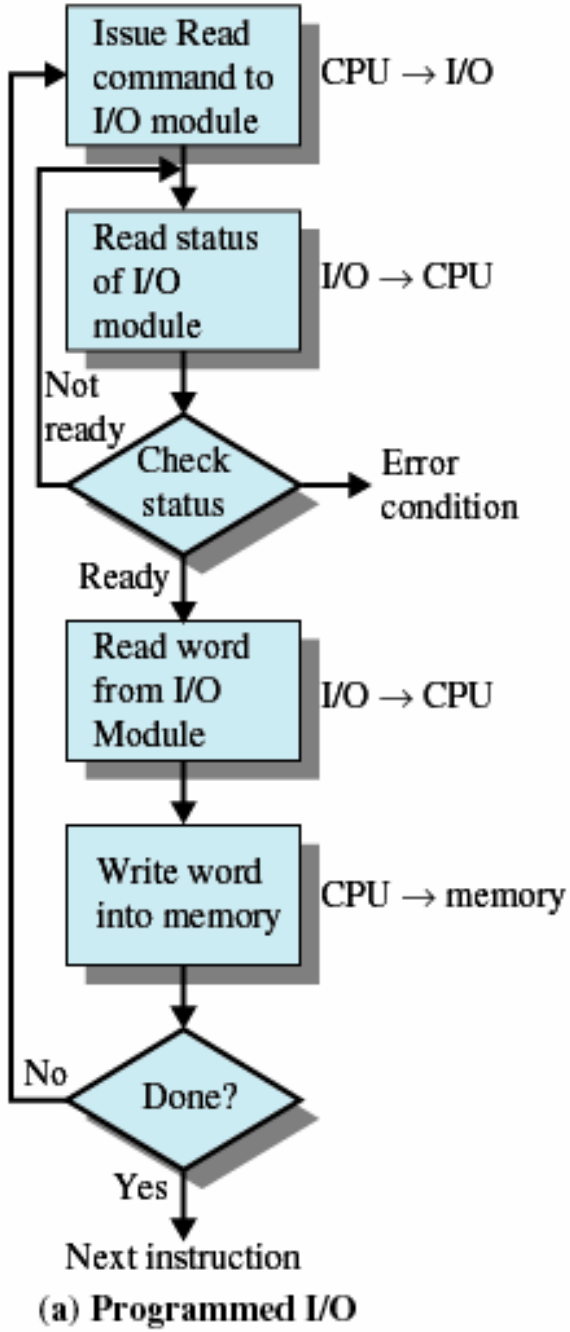
- Write policy needed
  - When a memory write operation takes place
    => Inconsistency between cache and main memory
  - Need to synchronize cache contents with memory
  - Can occur every time block is updated
  - Can occur only when block is replaced
    - Minimizes memory write operations
    - BUT, leaves main memory in an obsolete state

Approached to Handling I/O (Data Transfer)

- Programmed I/O
  - I/O Module performs minimal actions, relies on processor to recognize when I/O complete
- Interrupt driven I/O
  - I/O Module sets interrupt buit
  - Overlapping of Pgm execution and data transfer
- Direct Memory Access (DMA)
  - I/O Module talks directly to Memory Unit

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 continually checks status until operation is complete

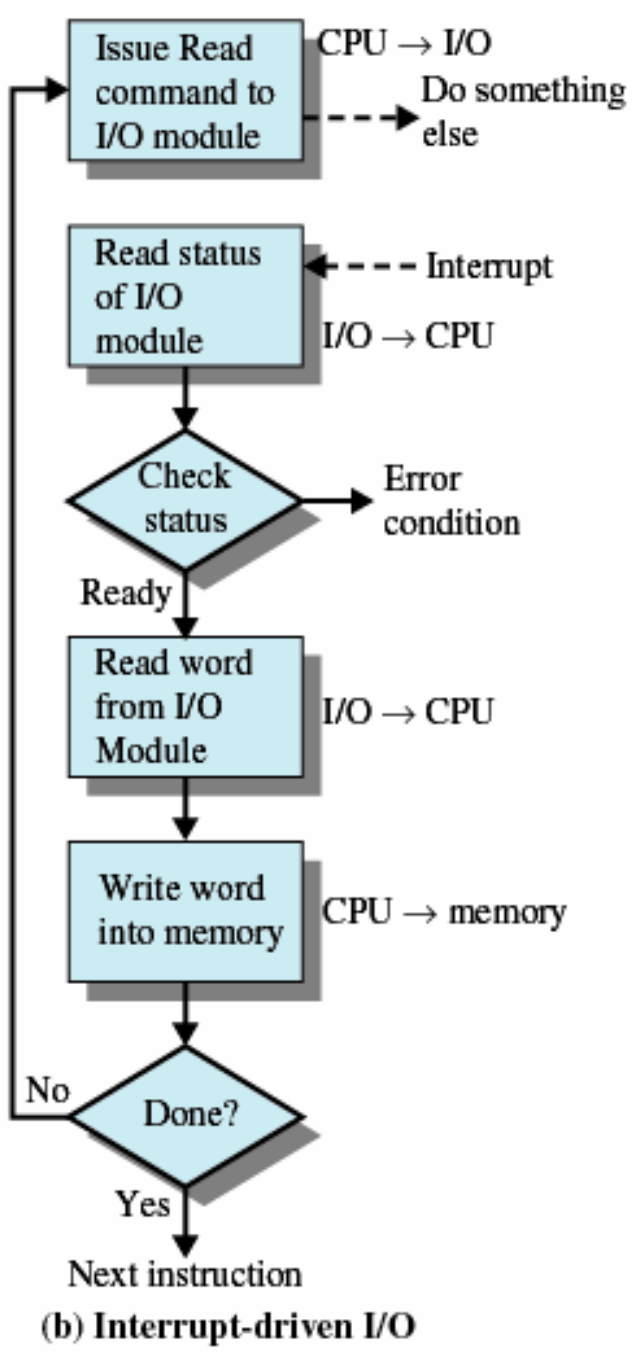


(a) Programmed I/O

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

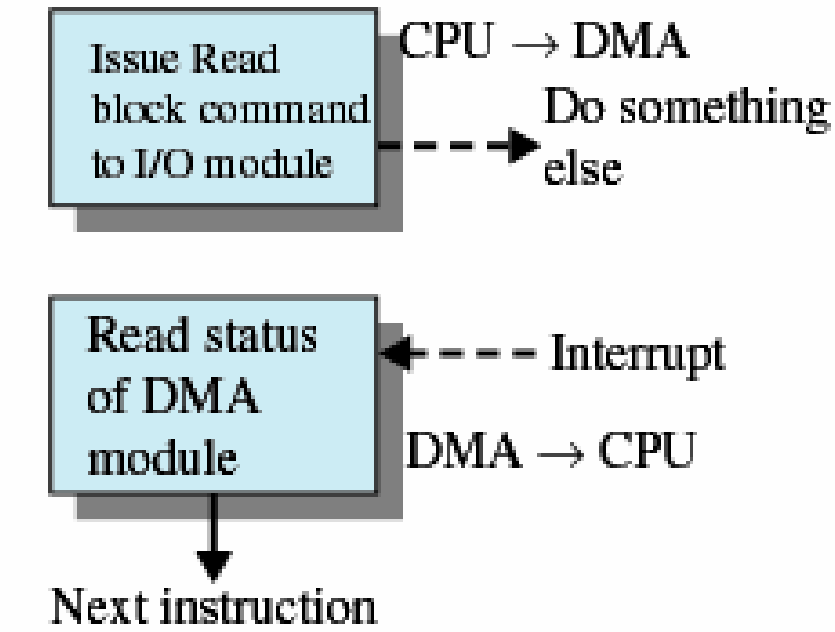
- Consumes a lot of processor time because every word read or written passes through the processor



(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