



Operating Systems

“Computer System Overview”

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
 - 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

Basic Elements

- 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, main memory, and I/O modules

Computer Components: Top-Level View

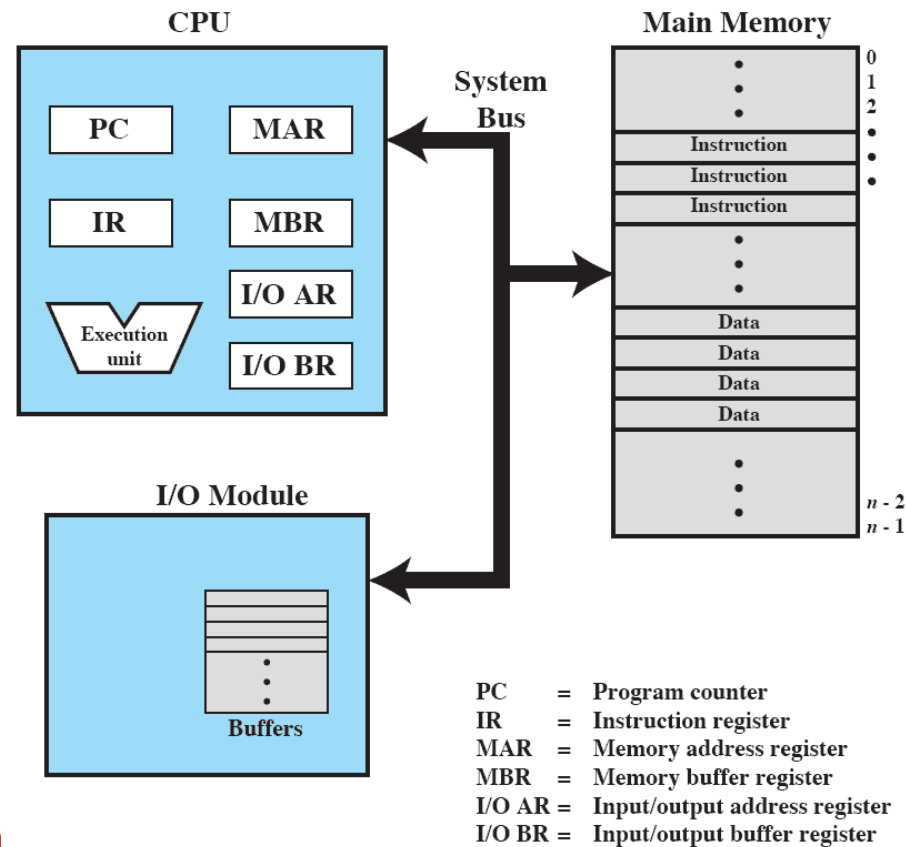


Figure 1.1 Computer Components: Top-Level View

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
- 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
- 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

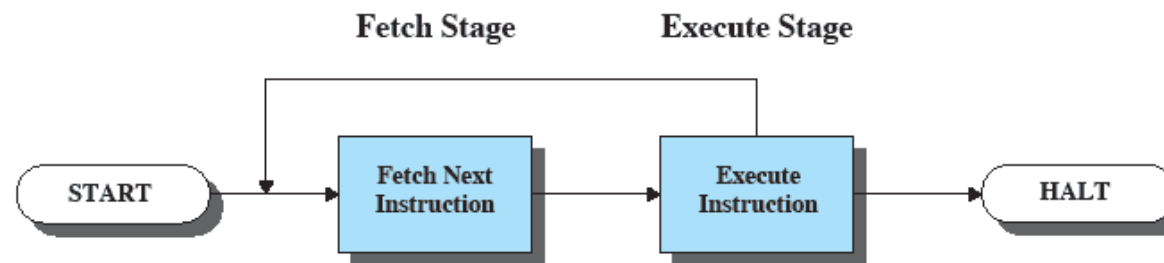


Figure 1.2 Basic Instruction Cycle

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

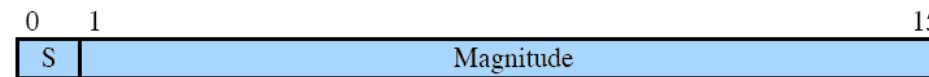
Instruction Register

- Fetched instruction loaded into instruction register
- Categories of the instructions
 - 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

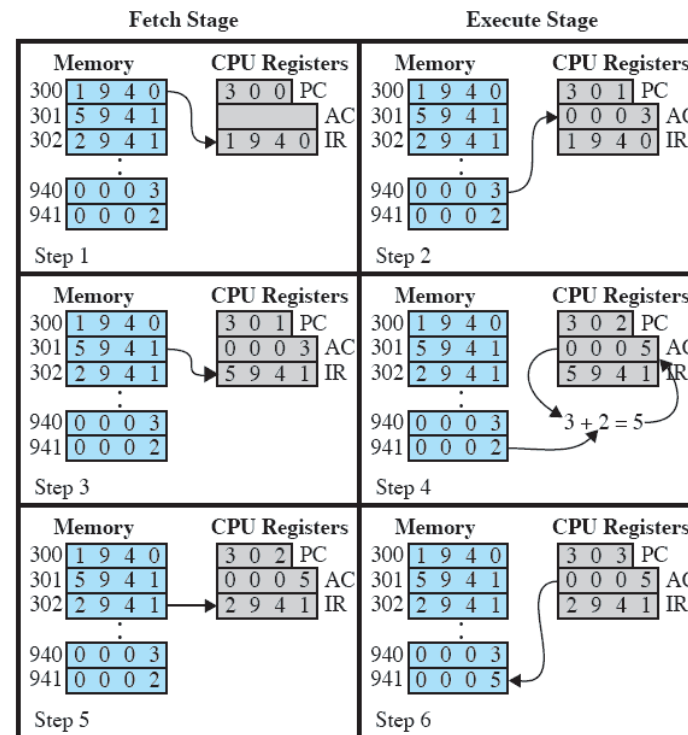


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

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.

Interrupt Stage

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

Transfer of Control via Interrupts

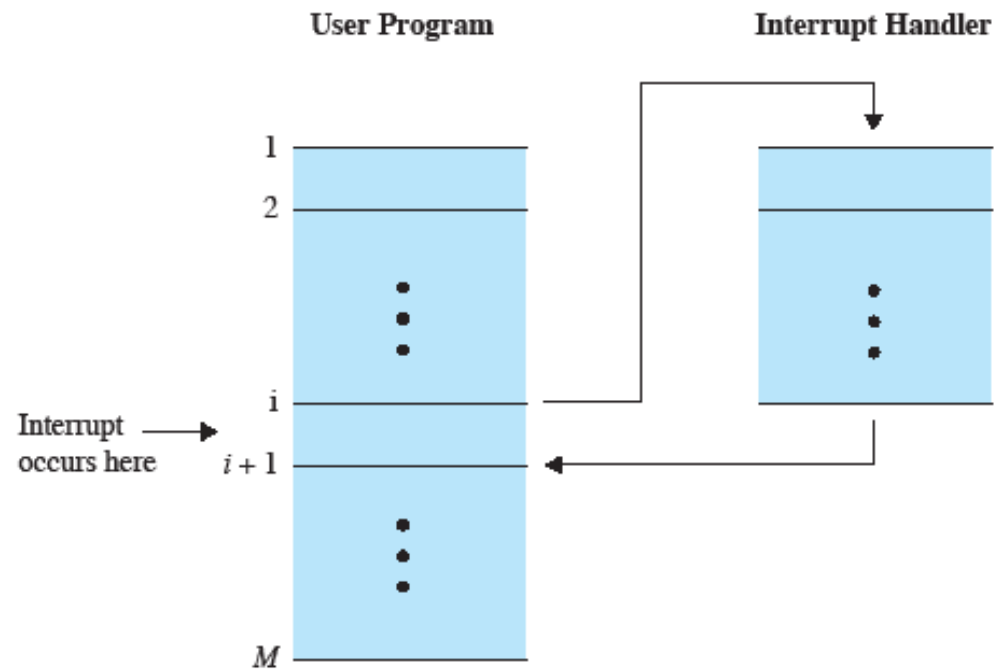
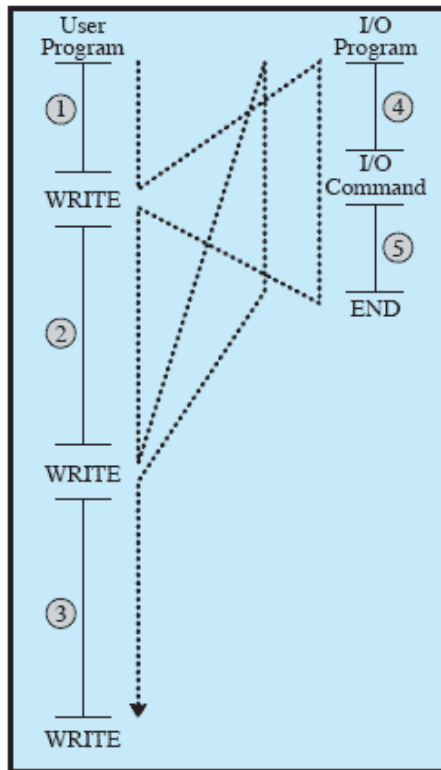
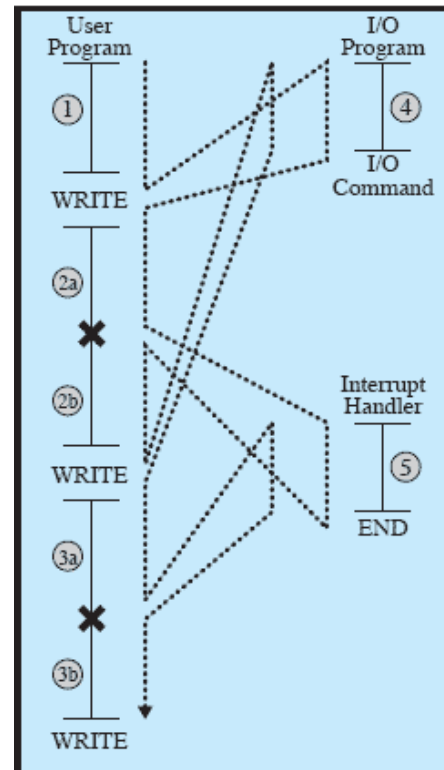


Figure 1.6 Transfer of Control via Interrupts

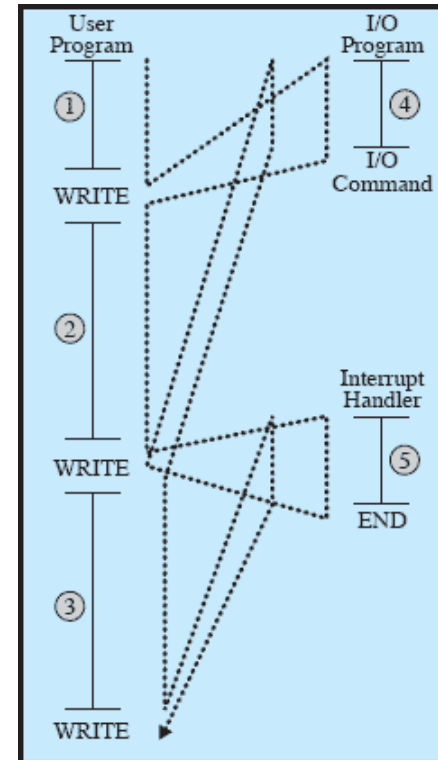
Program Flow of Control



(a) No interrupts



(b) Interrupts; short I/O wait



(c) Interrupts; long I/O wait

Instruction Cycle with Interrupts

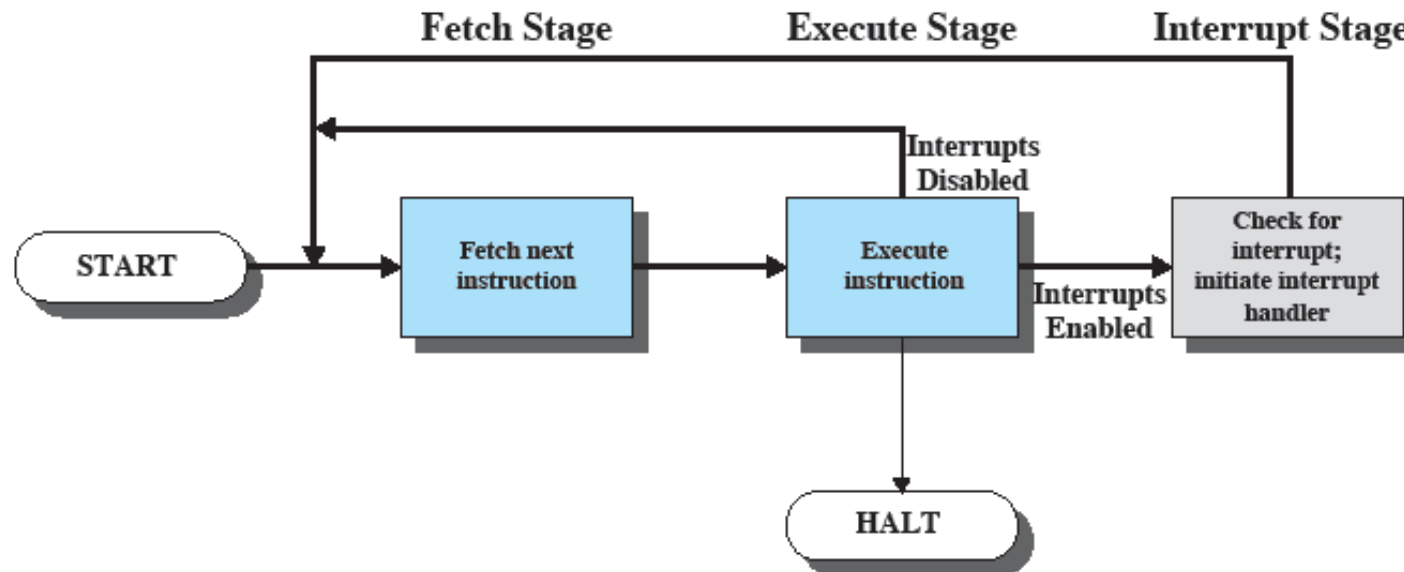


Figure 1.7 Instruction Cycle with Interrupts

Program Timing

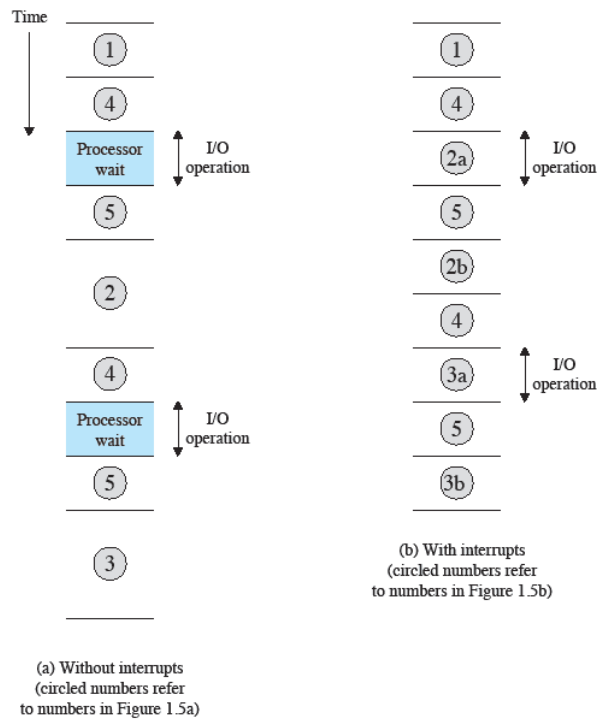


Figure 1.8 Program Timing: Short I/O Wait

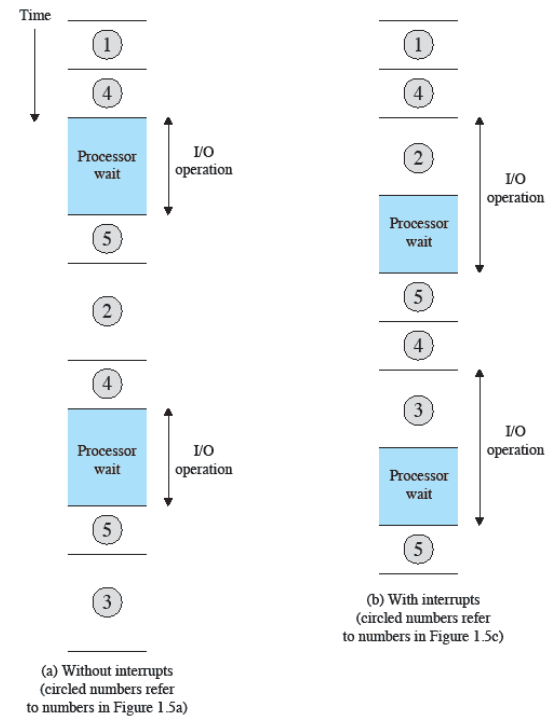


Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

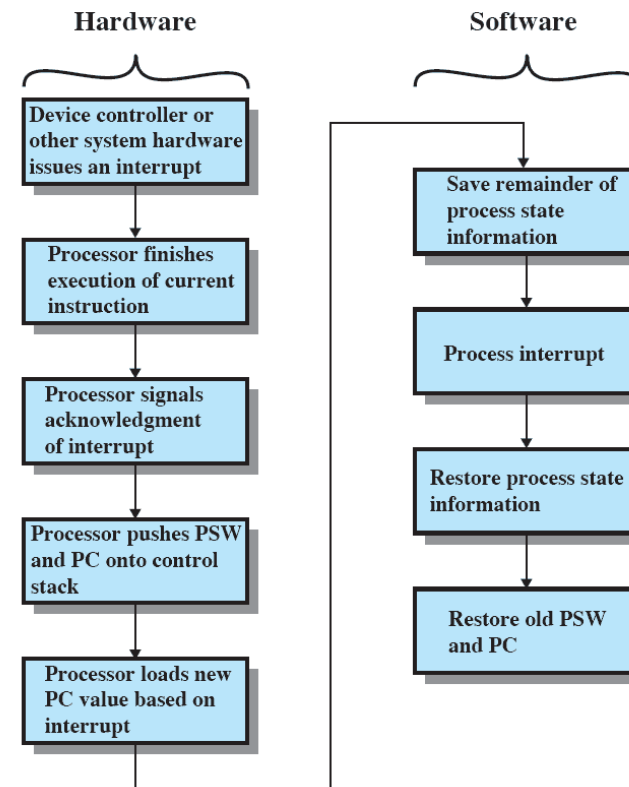
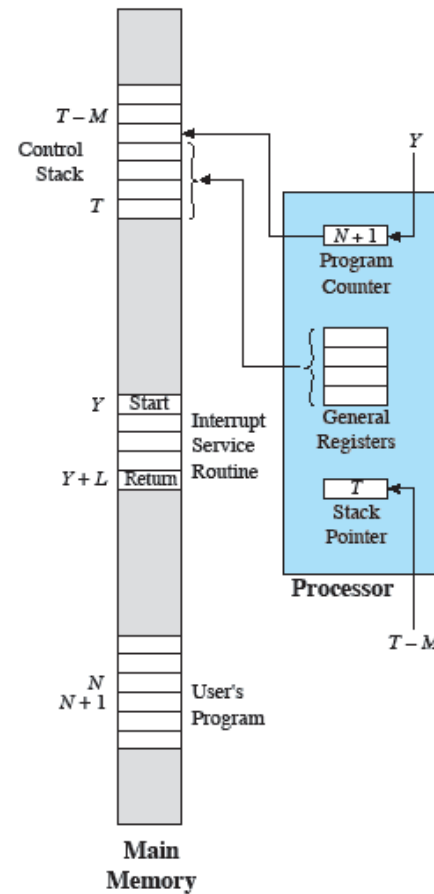
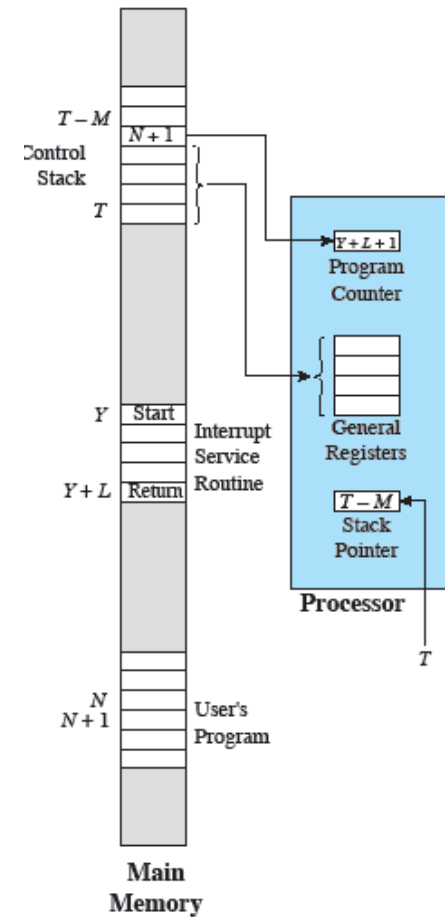


Figure 1.10 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt

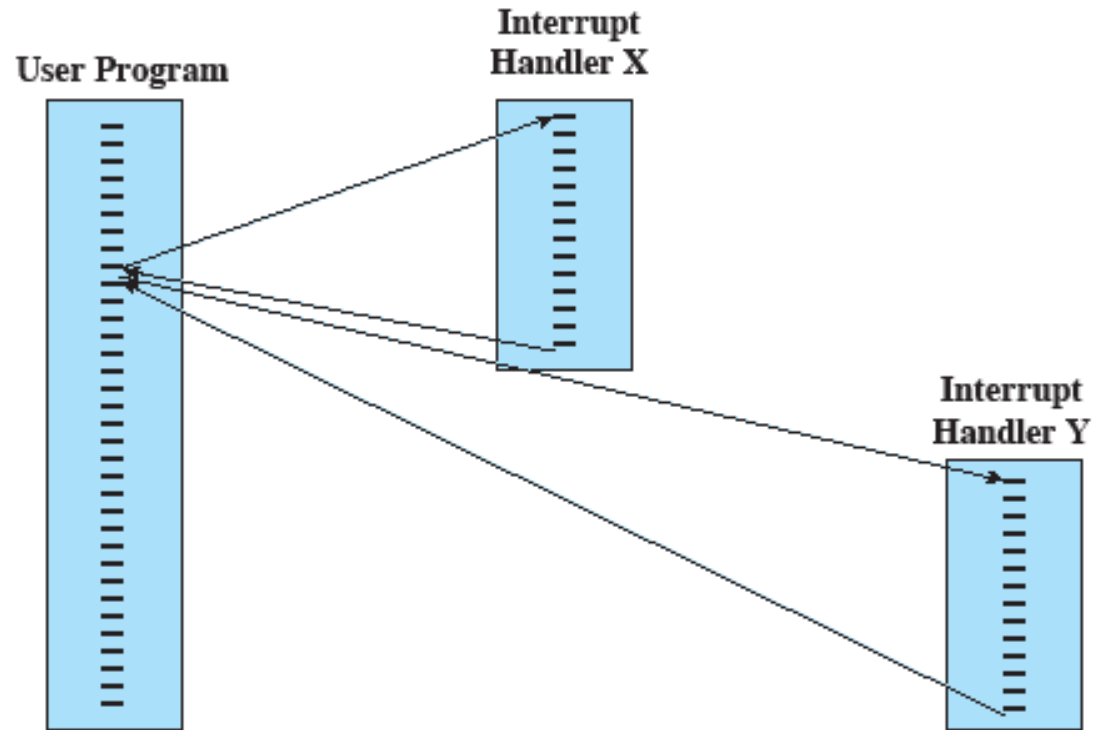


(a) Interrupt occurs after instruction at location N



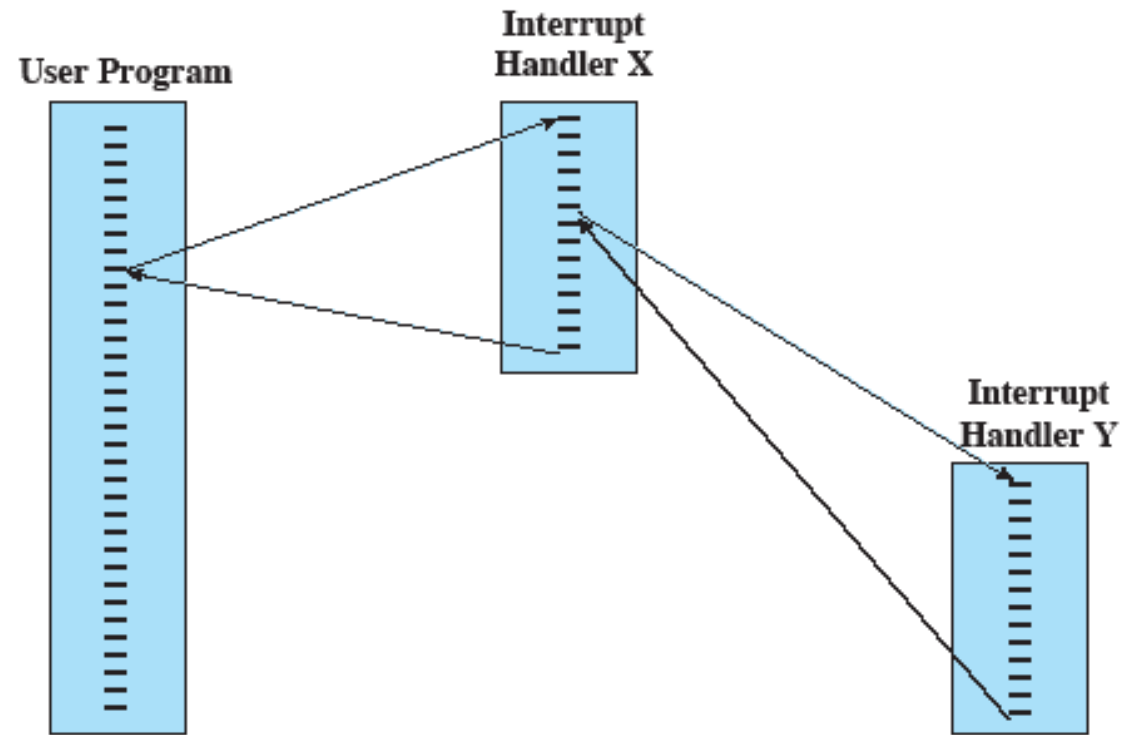
(b) Return from interrupt

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

Memory Hierarchy

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

The Memory Hierarchy

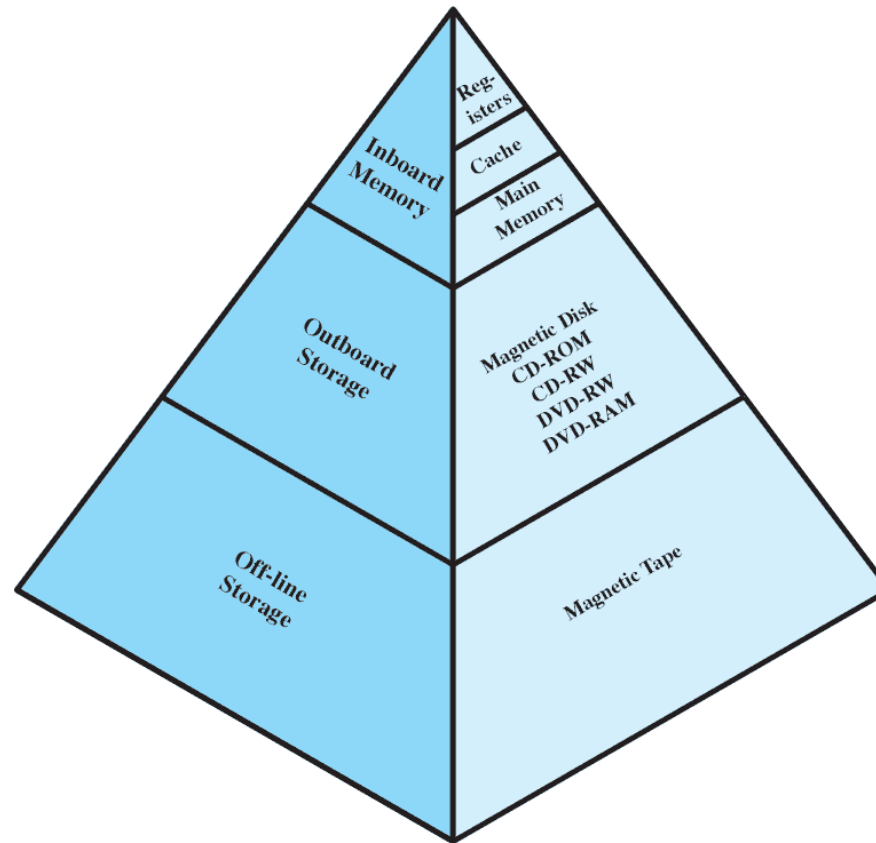


Figure 1.14 The Memory Hierarchy

Going Down the Hierarchy

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

Secondary Memory

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

Cache Memory

- Processor speed faster than memory access speed
- Exploit the principle of locality with a small fast memory

Cache and Main Memory

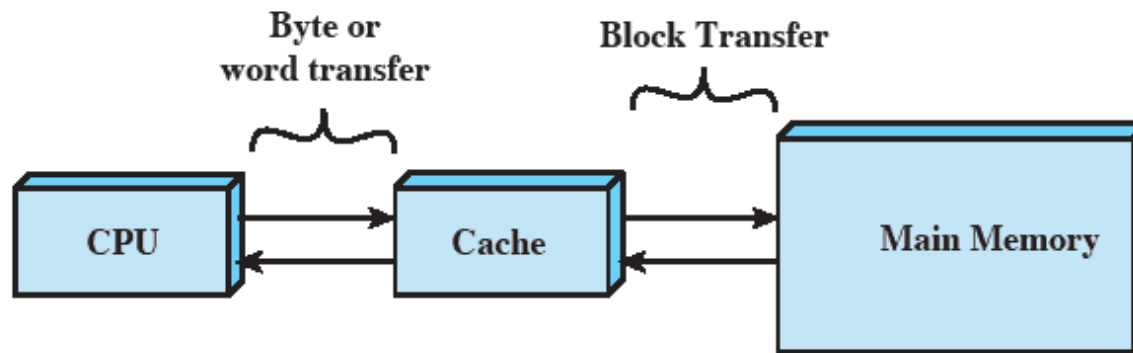


Figure 1.16 Cache and Main Memory

Cache Principles

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

Cache/Main-Memory Structure

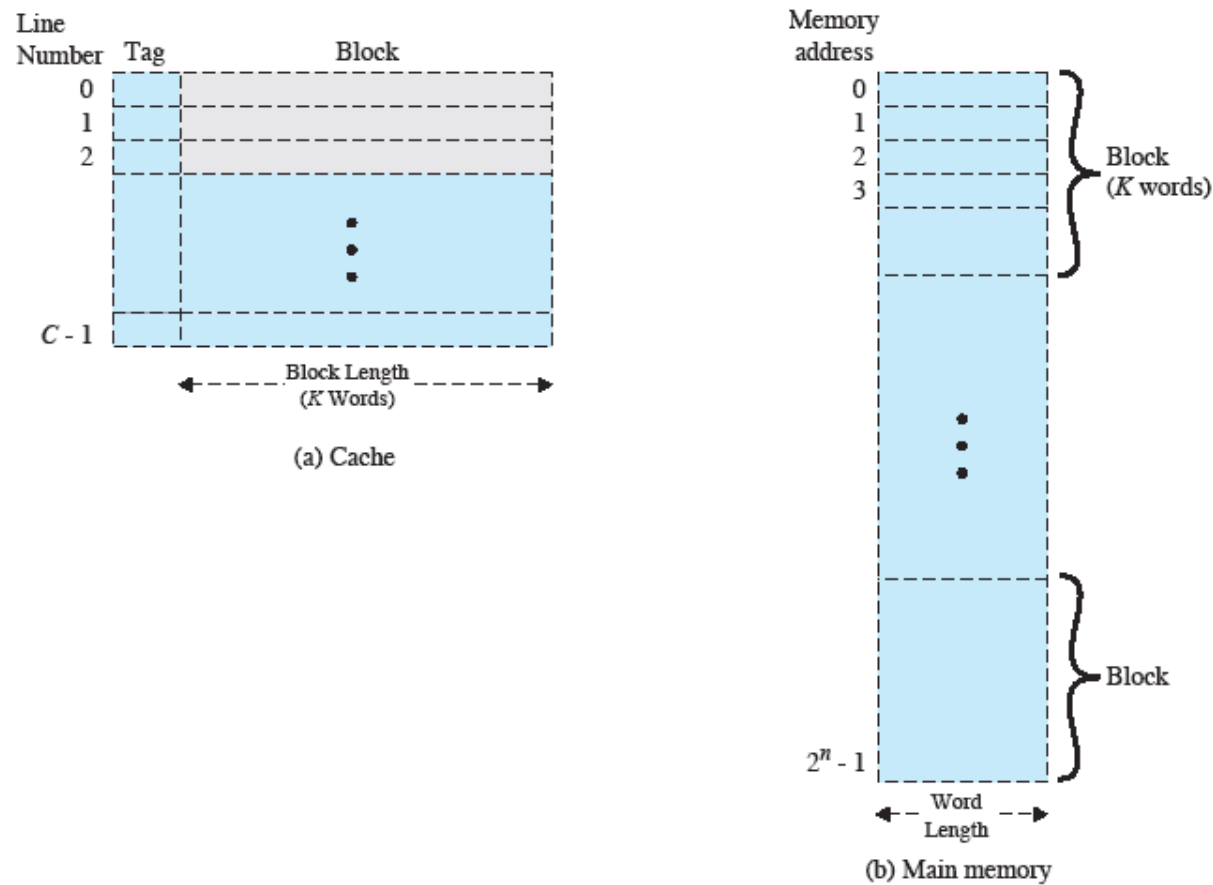


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation

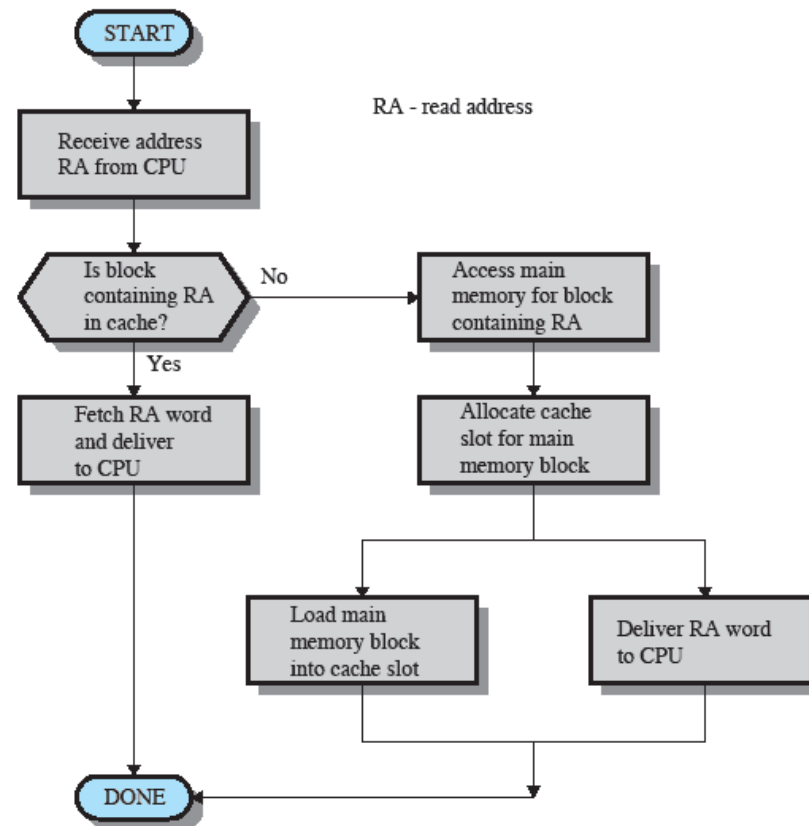


Figure 1.18 Cache Read Operation

Cache Principles

- Cache size
 - Even small caches have significant impact on performance
- Block size
 - The unit of data exchanged between cache and main memory
 - Larger block size yields 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 Principles

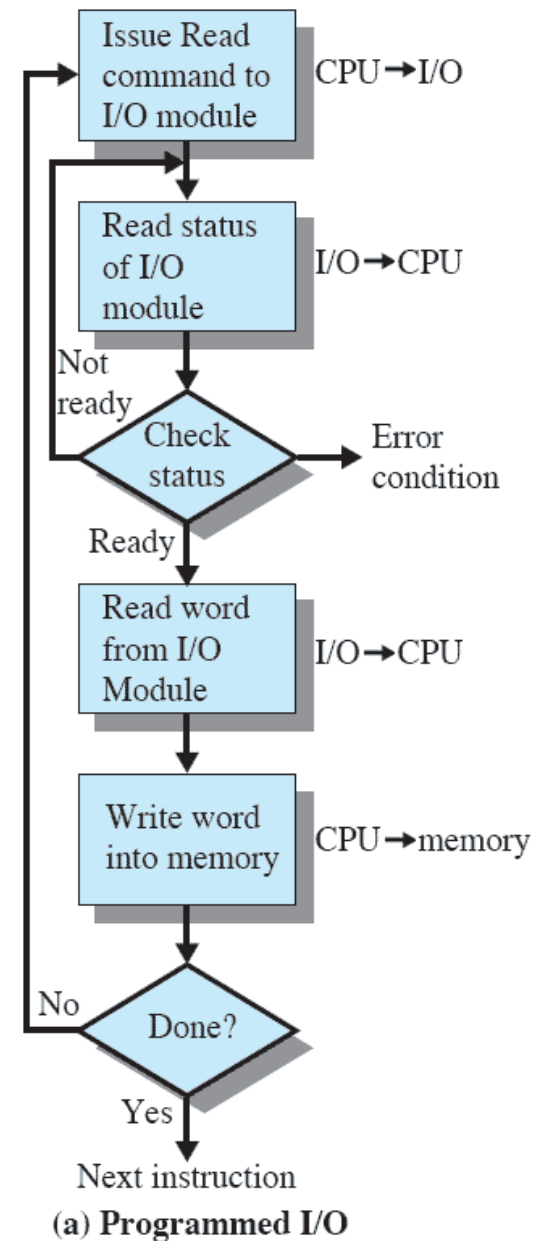
- Mapping function
 - Determines which cache location the block will occupy
- Replacement algorithm
 - Chooses which block to replace
 - Least-recently-used (LRU) algorithm
 - First in First Out (FIFO) algorithm
 - Least Frequently Used (LFU) algorithm

Cache Principles

- 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 which are in an obsolete state

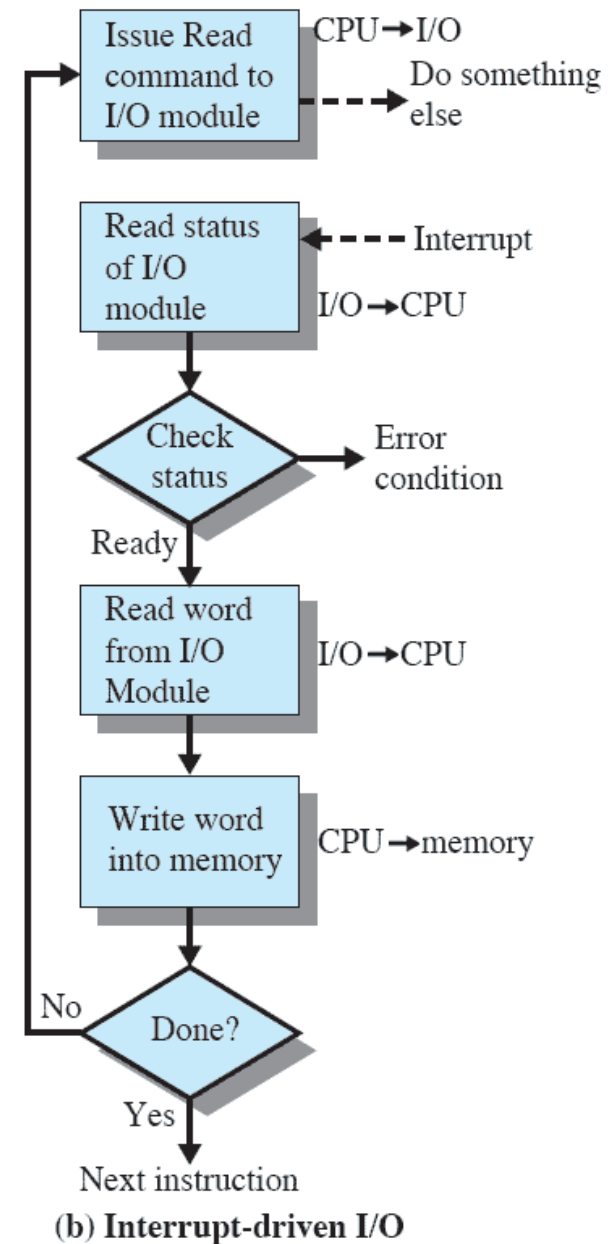
Programmed I/O

- 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



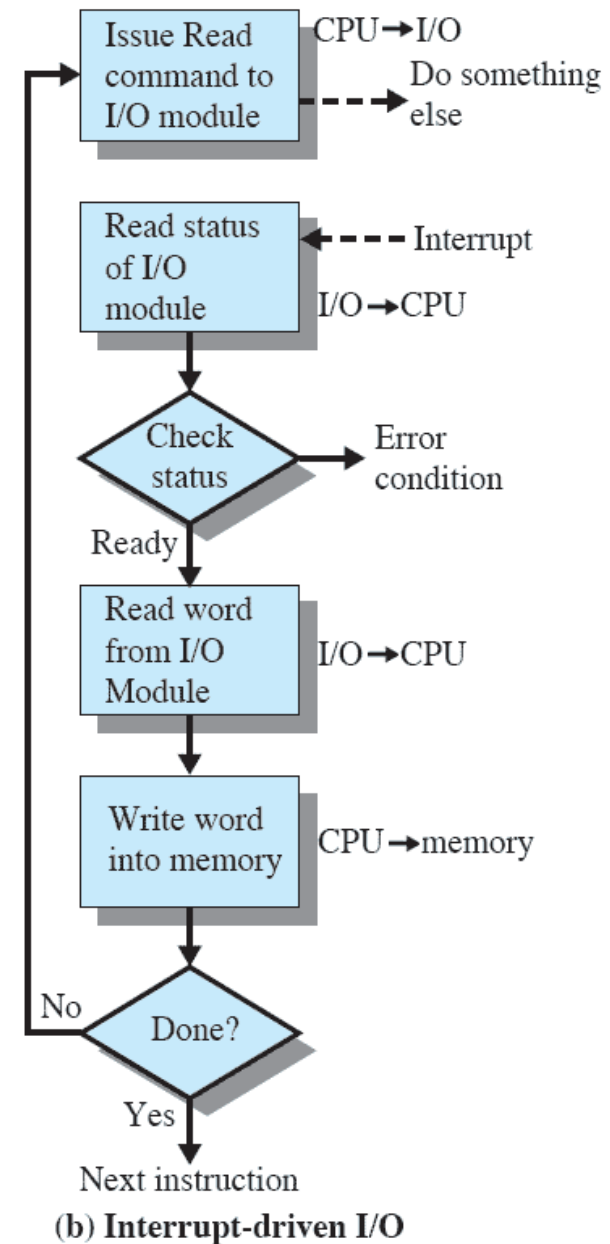
Interrupt-Driven I/O

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



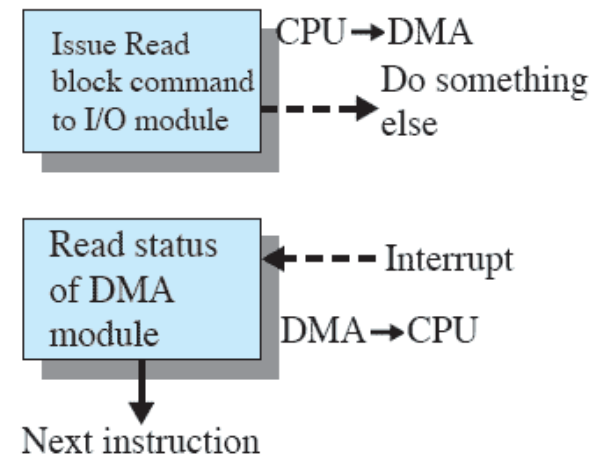
Interrupt-Driven I/O

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



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

Thank You