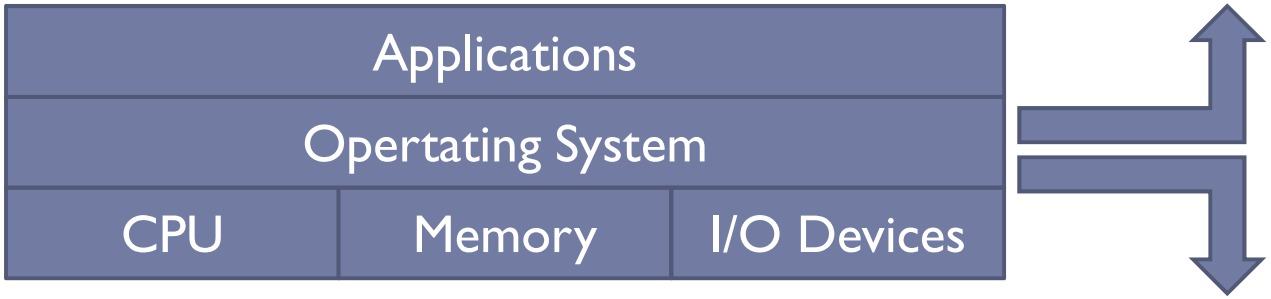
Computer System Overview
计算机系统概述

Chapter 1

Why should we understand the hardware?

Provides a set of services to applications and system users



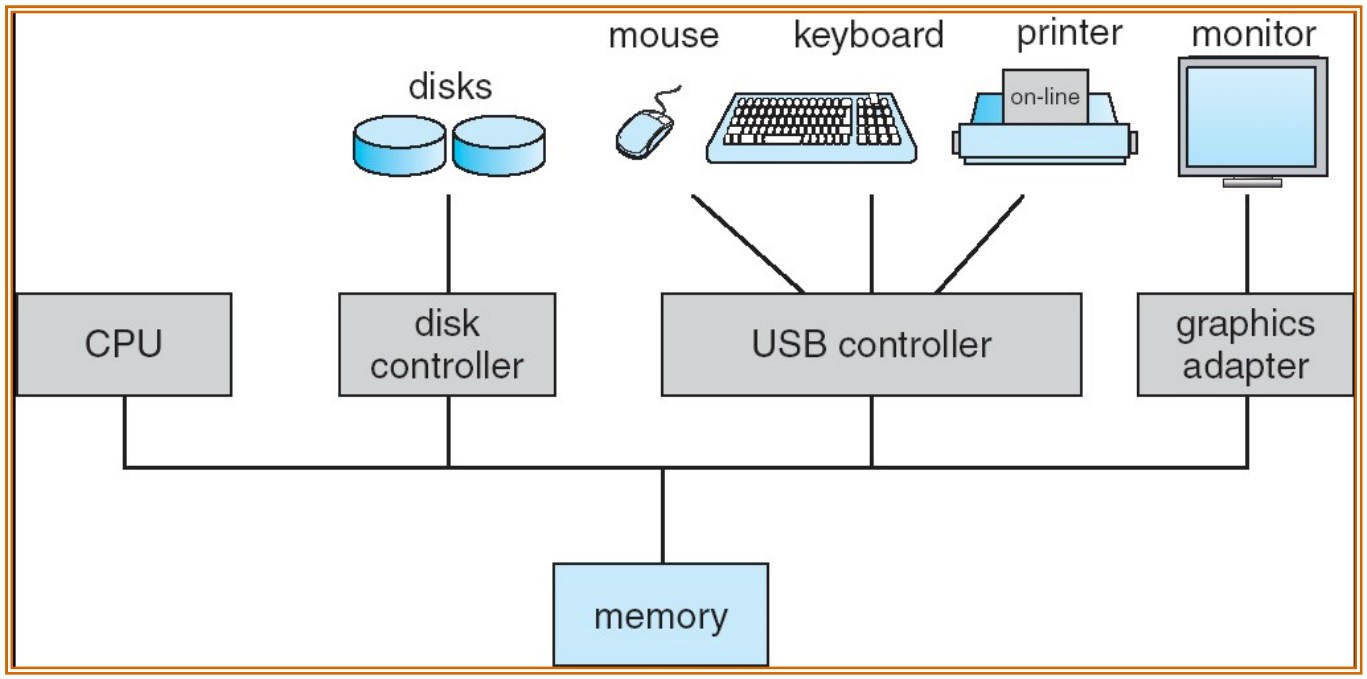
Manages the hardware resources of a computer system, inclusive of CPUs, memory, and I/O devices

Main Contents

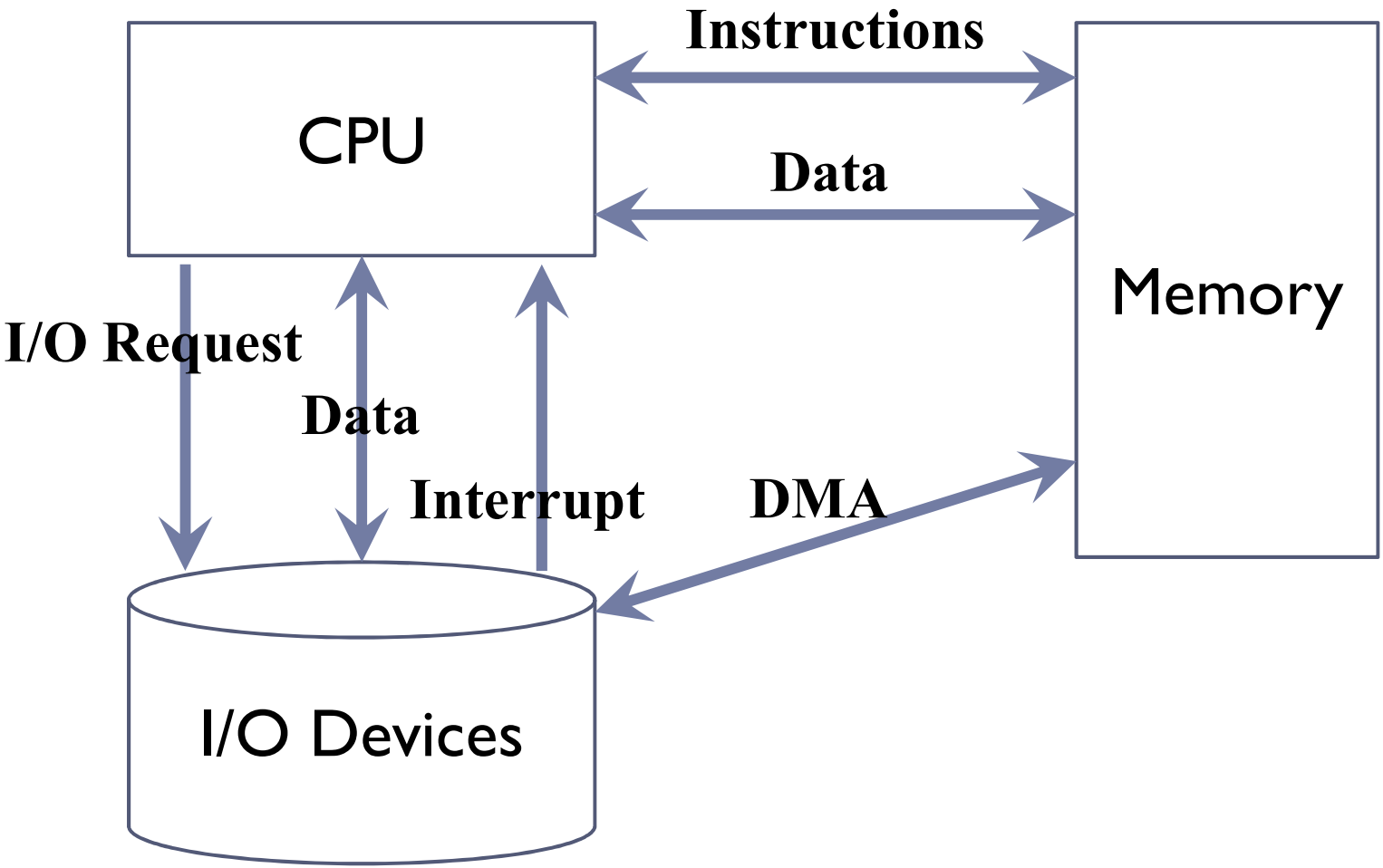
- ***Basic Elements of a Computer System***
- ***Processor***
  - *Processor Registers*
  - *Instruction Execution*
  - *Interrupts*
- ***Memory***
  - *Memory Hierarchy*
  - *Cache Memory (hit ratio)*
- ***I/O Device***
  - *I/O Communication Techniques*
- ***Computer Booting***

Basic Elements of a Computer System

A Simple Computer



The CPU, memory, and I/O devices are connected by a system bus, and communicate with one another over it.


Instructions
CPU
Data
Memory
I/O Request
Data
Interrupt
DMA
I/O Devices

Processors

Processors

- *The CPU is the "brain" of the computer*
- *The basic cycle of each CPU:*
  - *(Fetch Phase) fetch an instruction from memory*
  - *(Execute Phase) decode it and execute it.*
- *This cycle is repeated until the program finishes.*

Processor Registers (1)
处理器寄存器

- ***A processor provides a set of registers***
  - ***It provides a level of memory 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, operating-system routines to control the execution of programs

Processor Registers (2)
User-Visible Registers

- Available to all programs - application programs and system programs
- Types of registers
  - Data Registers
    - They can be used with any machine instruction that performs operations on data
    - Restrictions: some dedicated registers may be used for floating-point operations and others for integer operations
  - Address Registers: Contain main memory addresses of data and instructions, or they contain a portion of the address that is used in the calculation of the complete address.

Processor Registers (3)
User-Visible Registers

- **Address Registers**
  - Index (索引寄存器)
    - involves adding an index to a base value to get an address
  - Segment pointer (段指针)
    - when memory is divided into segments, memory is referenced by a segment and an offset
  - Stack pointer (栈指针)
    - points to top of stack

Processor Registers (4)
Control and Status Registers

- **Program Counter (PC) 程序计数器**
  - Contains the address of an instruction to be fetched
- **Instruction Register (IR) 指令寄存器**
  - Contains the instruction most recently fetched
- **Program Status Word (PSW) 程序状态字**
  - condition code bits
  - Interrupt enable/disable
  - Supervisor/user (or kernel/user) mode

Instruction Execution (1)
Basic Instruction Cycle (基本指令周期)

In its simplest form, instruction processing consists of two steps:

The processor reads (fetches) instruction from memory one at a time and executes each instruction

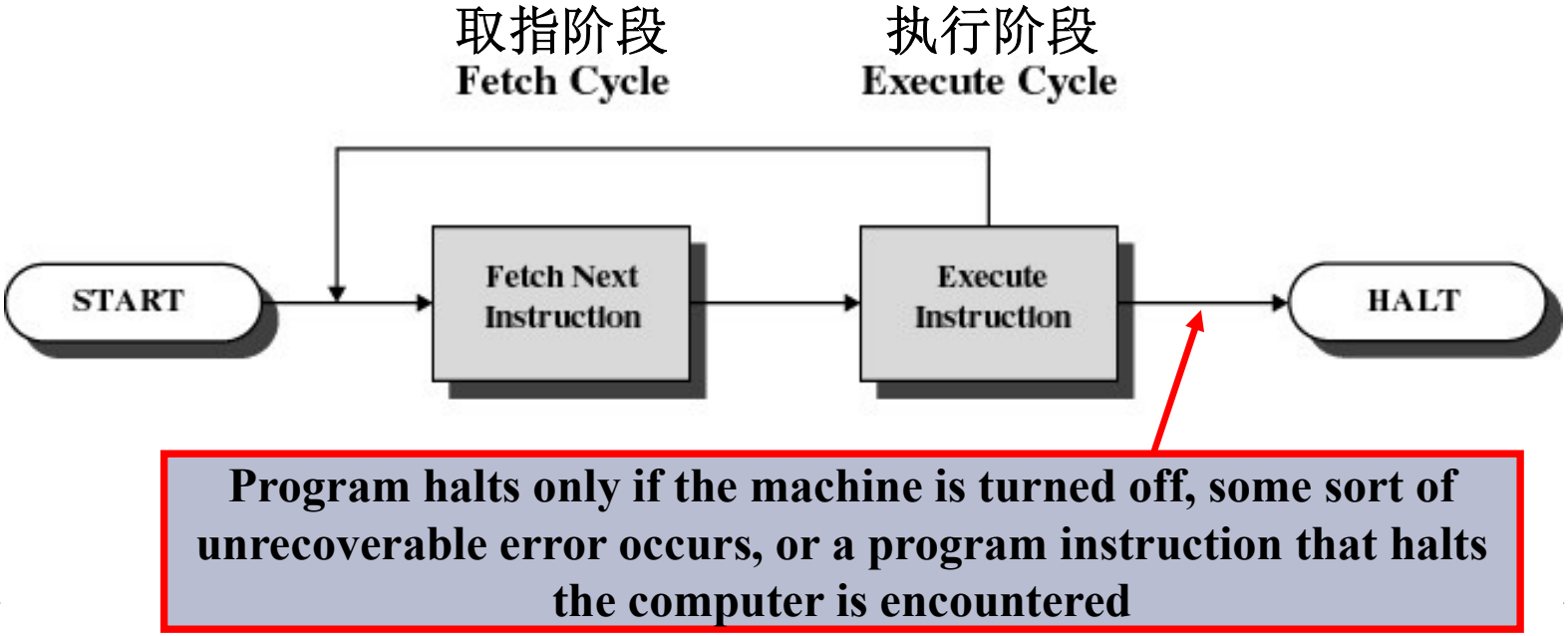


Figure 1.2 Basic Instruction Cycle

Instruction Execution (2)
Instruction Fetch

- ***The processor fetches the instruction from memory***
  - *Program counter* (PC程序计数器) *is used to get the address of the instruction to be fetched next*
  - *Fetched instruction is placed in the instruction register* (IR 指令寄存器)

Instruction Execution (3)
Types of Instructions

- ***The actions required by instructions fall into four 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*
- ***An instruction's execution may involve a combination of these actions***

Execution Modes

- **Most CPUs have two modes: kernel mode and user mode.**
  - The operating system runs in kernel mode, the CPU can execute every instruction in its instruction set and use every feature of the hardware
  - User programs runs in user mode, which permits only a subset of the instructions and a subset of features to be accessed.
- **Generally, all instructions involving I/O and memory protection are disallowed in user mode.**

Interrupts
中断

Interrupt Mechanism

- *Interrupt mechanism is provided to allow other modules (I/O, memory) to interrupt the normal execution sequence of the processor.*
- *The goal: to improve processing efficiency*
  - *Allows the processor to execute other instructions while an I/O operation is in progress*
- *A suspension of a process caused by an event external to that process and performed in such a way that the process can be resumed*

Classes of Interrupts

- **Program**
  - arithmetic overflow算术溢出
  - division by zero 除数为零
  - execute illegal instruction 执行非法的机器指令
  - reference outside user's memory space
- **Timer**
- **I/O**
- **Hardware failure**

Interrupt Handler

- A program that determines nature of the interrupt and performs whatever actions are needed
- Control is transferred to this program
- Generally part of the operating system

Instruction Cycle with Interrupts
具有中断的指令周期

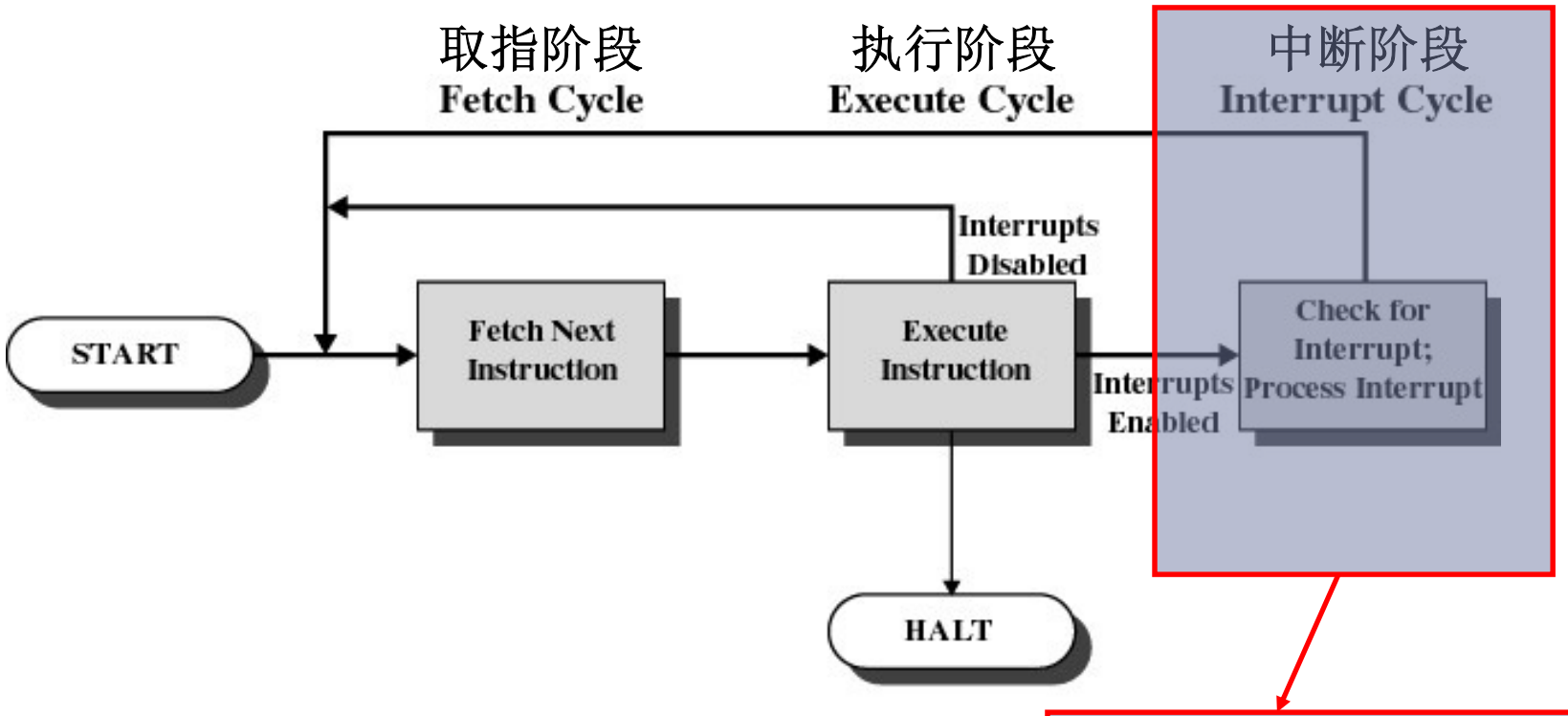
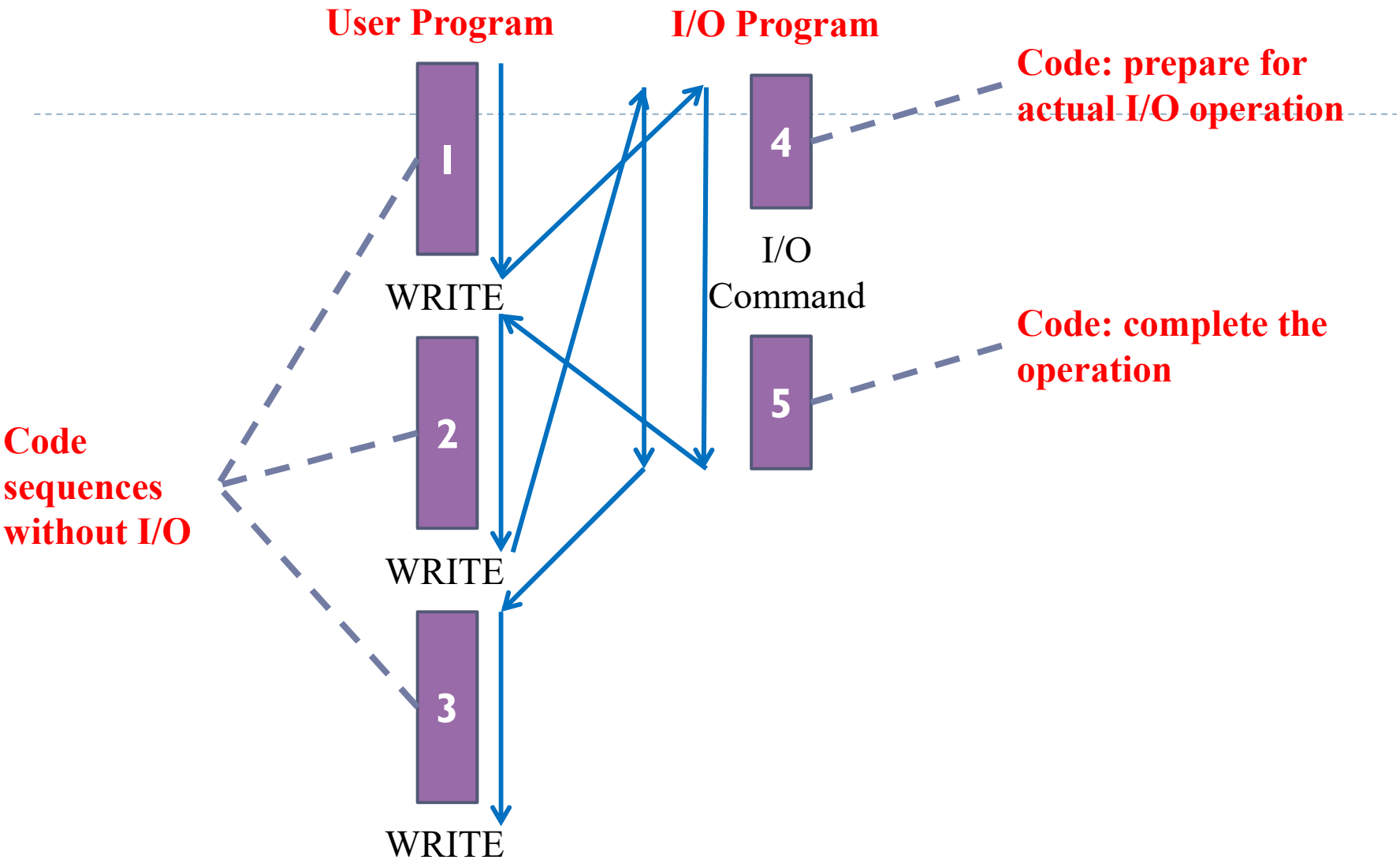


Figure 1.7 Instruction Cycle with Interrupts

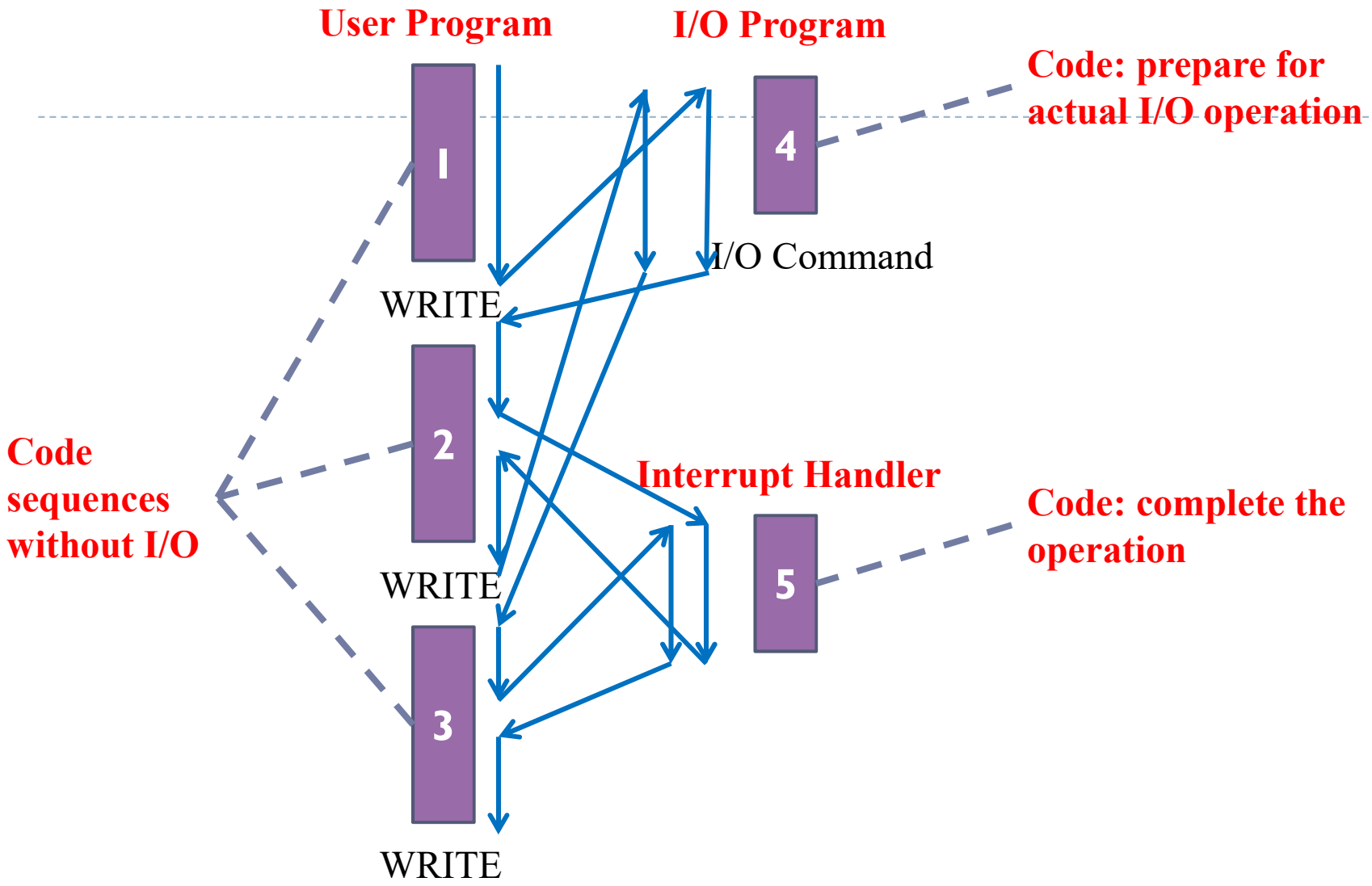
An interrupt cycle is added to the instruction cycle

Instruction Cycle with Interrupts
Interrupt Cycle(中断阶段)

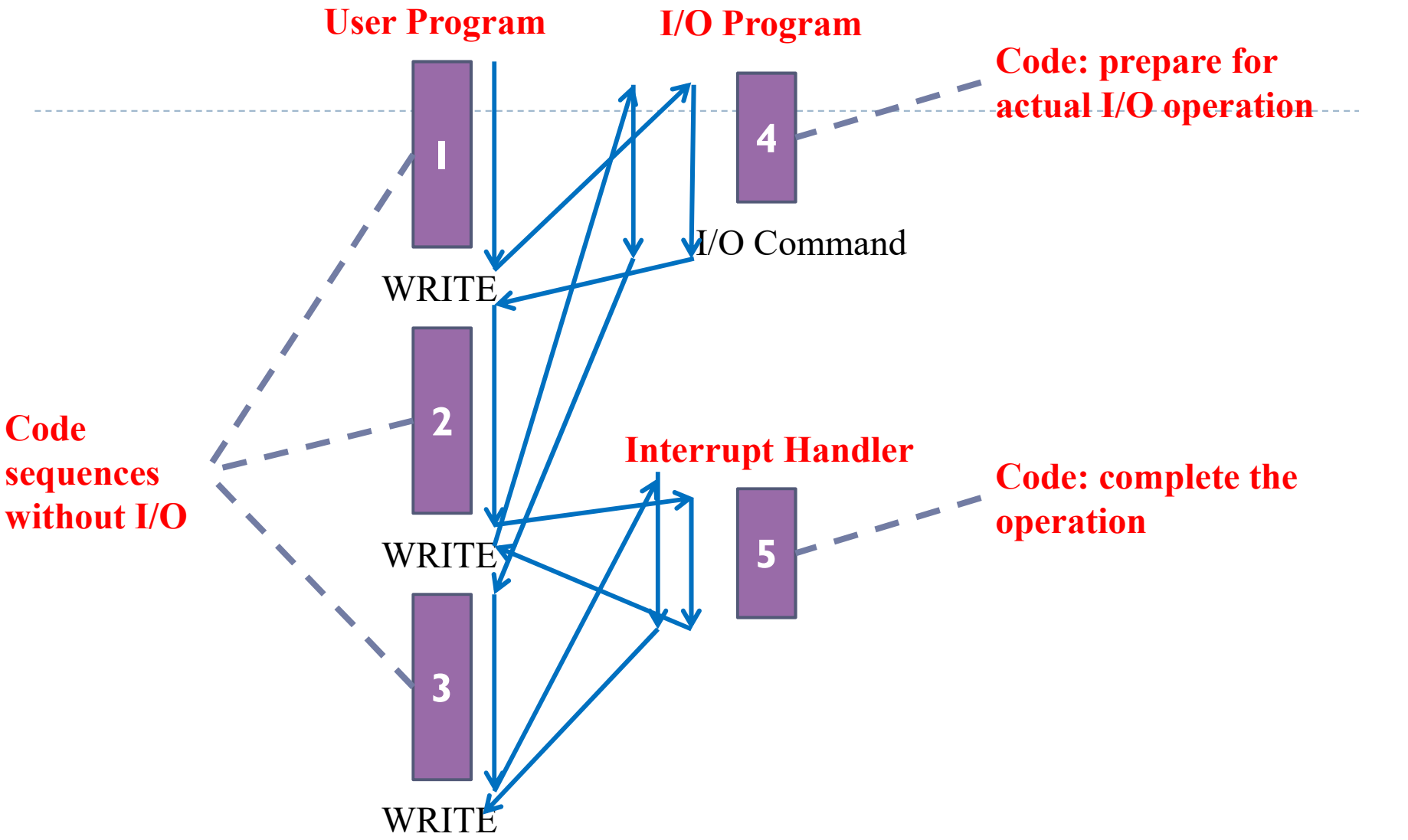
- **Processor checks for interrupts**
  - **If no interrupts fetch the next instruction for the current program**
  - **If an interrupt is pending, suspend execution of the current program, and execute the interrupt handler**



No Interrupts



Interrupts: Short I/O Wait



Interrupts: Long I/O Wait

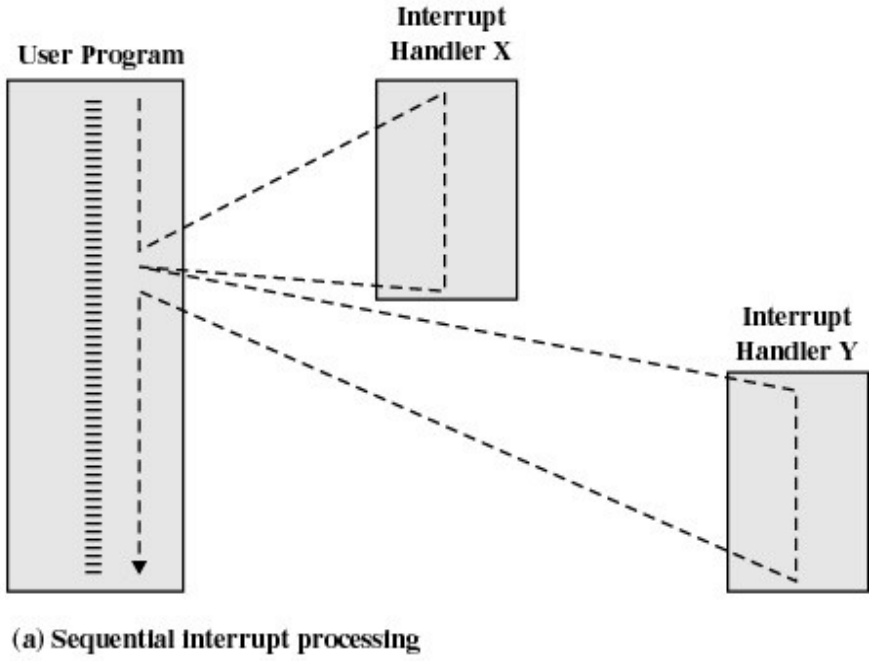
Multiple Interrupts (1)

- *There are multiple I/O devices in the system*
  - *Multiple interrupts can occur*
- *Example: A program may be receiving data from a communications line and printing results.*
  - *It is possible for a communications interrupt to occur while a printer interrupt is being processed*
  - *How to deal with this situation?*

Multiple Interrupts (2)
Two approaches

- ***One approach: To disable interrupts while an interrupt is being processed***
  - *Drawback: It does not take into consideration **relative priority** or **time-critical needs***
- ***The other approach: To define priorities for interrupts***
  - *To allow an interrupt of higher priority to interrupt an interrupt handler of lower priority.*

- *Disable interrupts so processor can complete the interrupt-handler*
- *Interrupts remain pending until the processor enables interrupts*
- *After interrupt handler routine completes, the processor checks for additional interrupts*



- *A higher priority interrupt can cause a lower-priority interrupt handler to be interrupted*
- *Example: when input arrives from communication line, it needs to be absorbed quickly to make room for more input*

Figure 1.12 Transfer of Control with 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

The Memory Hierarchy
存储器的层次结构

Memory
存储器

- Memory is the second major component in any computer
- Ideally, a memory should be
  - ***Speed***: Extremely fast (速度)
  - ***Capacity***: Abundantly large (容量)
  - ***Price***: Dirt Cheap (价格)
- It is hard to satisfy all these goals!! How to do with it?

Memory Hierarchy

Increasing capacity
Decreasing access speed
Decreasing cost per bit
Decreasing access frequency by the processor

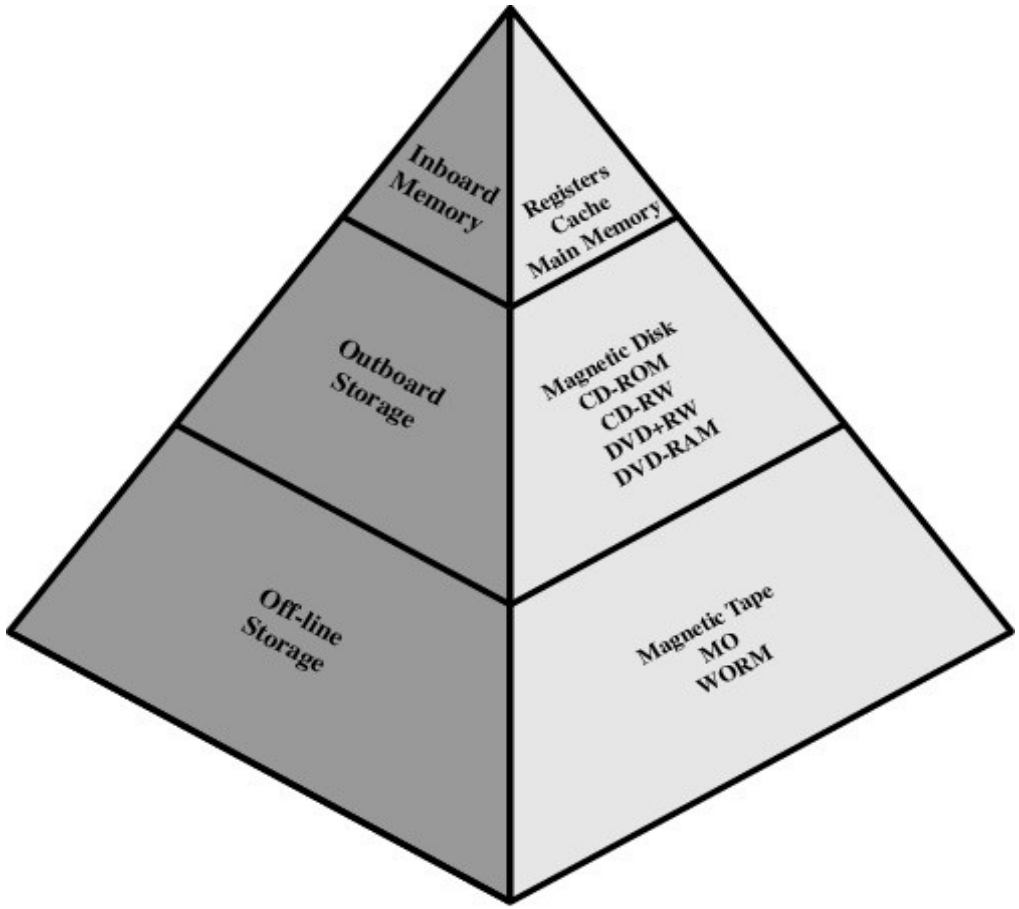


Figure 1.14 The Memory Hierarchy

Registers

- **They are internal to the CPU**
  - **They are as fast as CPU**
  - **Capacity: typically 32*32-bits on a 32-bit CPU and 64*64-bits on a 64-bit CPU.**
  - **Programs must manage the registers (i.e. decide what to keep in them) themselves, in software.**

Cache Memory

- **Mostly controlled by the hardware**
- **Cache Line**
  - **Main Memory is divided into cache lines (typically 64 bytes), with addresses 0 to 63 in cache line 0, addresses 64 to 127 in cache line 1, and so on.**
  - **The most heavily used cache lines are kept in a high-speed cache located inside or very close to the CPU**
- **Cache Hit:**
  - **when the program needs to read a memory word, the cache hardware finds out that the line is in the cache**
- **Capacity is limited, due to its high cost**

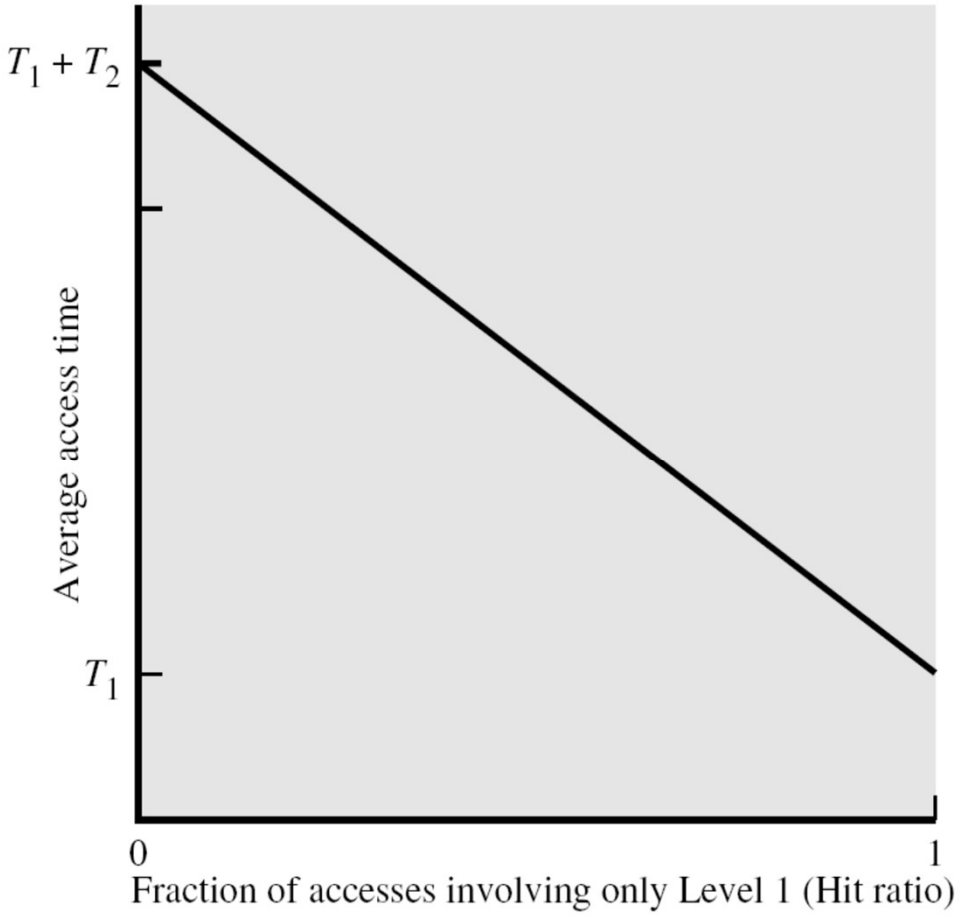
Main Memory

- Main memory is often called RAM (Random Access Memory)
- All CPU requests that can not be satisfied out of the cache go to main memory
- The key to the success of cache memory is "decreasing frequency of memory access"

Hit Ratio of Cache

- Benefits of cache memory
  - Invisible to operating system
  - Increase the speed of memory
- Hit Ratio
  - Hit ratio is defined as the fraction of all memory accesses that are found in the faster memory (cache memory)

Performance of a Simple Two-Level Memory



- T_1→the access time to level 1
- T_2→the access time to level 2

If hit ratio is 95%, T_1=0.1μs, and T_2=1μs, then the average time to access a word can be expressed as

0.95*0.1+0.05*(0.1+1)=0.15μs

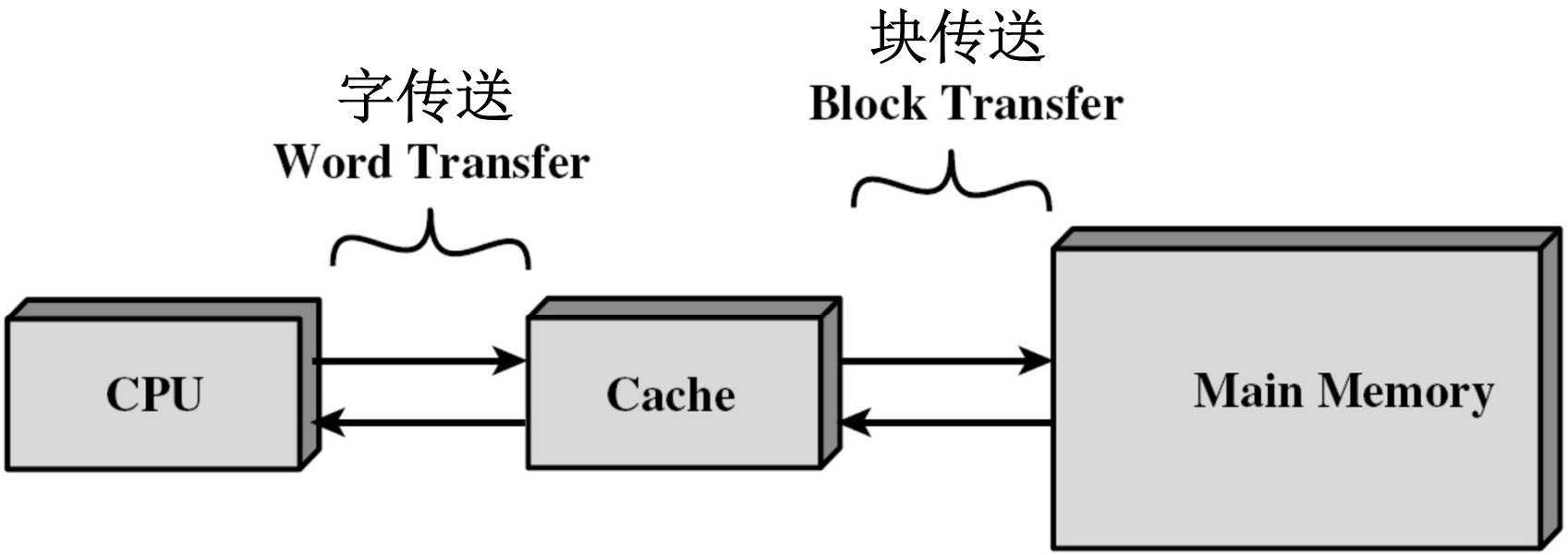
The basis for the success

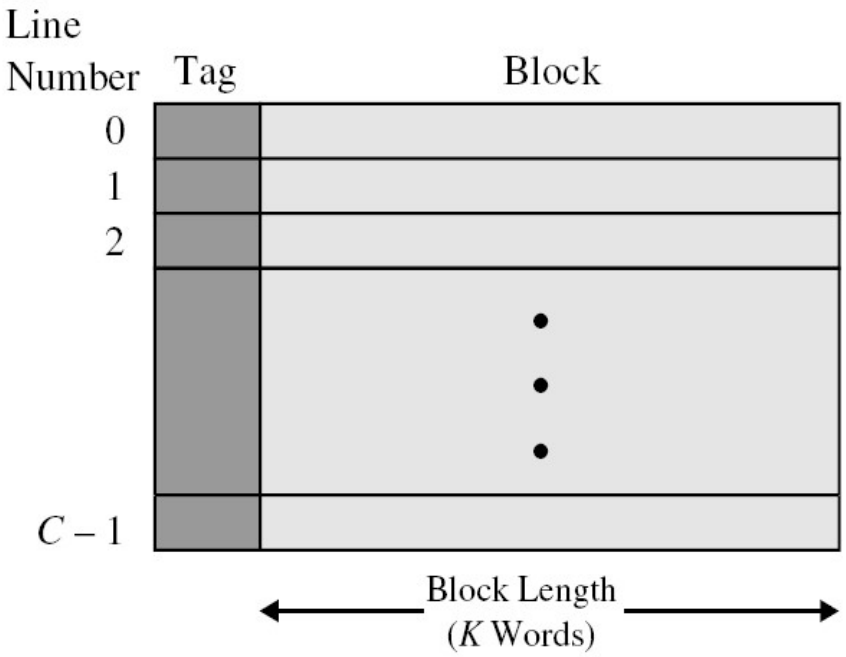
- Principle: Locality of reference
  - During the course of execution of a program, memory references by the processor tend to cluster.
- This principle can be applied across more than two levels of memory.
  - Exercise 1.13→Assignment!!

Cache Memory
高速缓冲存储器

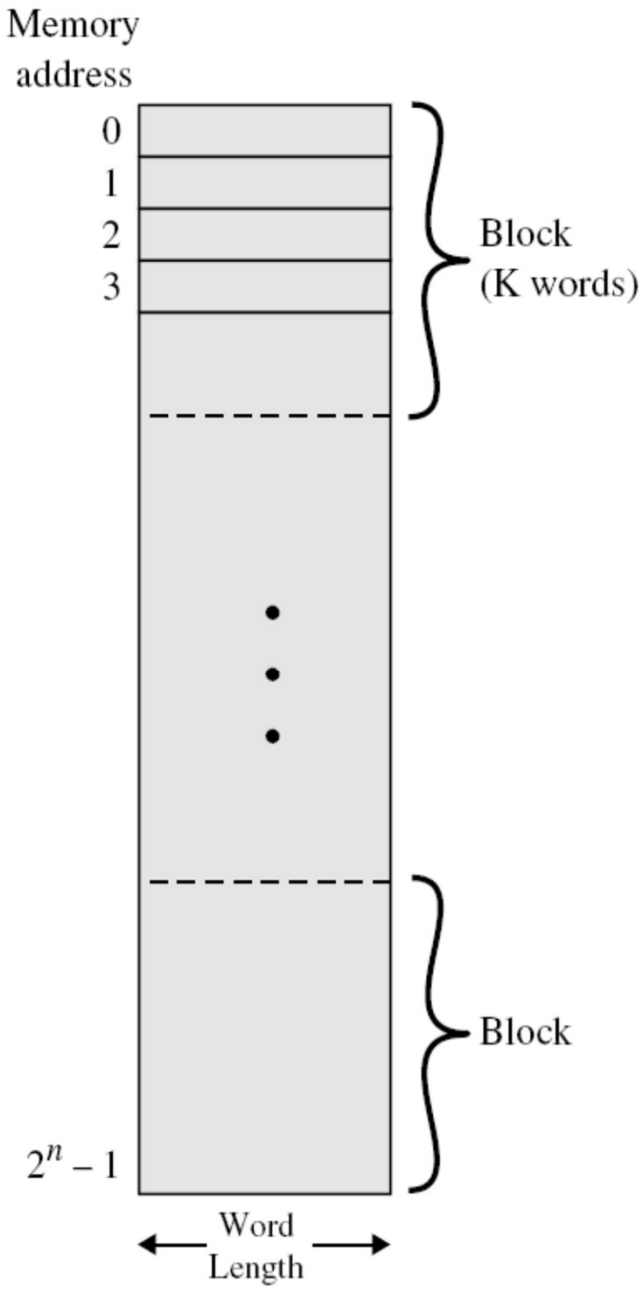
Motivation
动机

- **On all instruction cycles, the processor has to access memory**
  - **at least once: to fetch the instruction**
  - **often one or more additional times: to fetch operands and/or store results**
- **Persistent mismatch between processor and main memory speeds!!**
  - **Solution: Providing a small, fast memory (cache) between the processor and main memory**


字传送
Word Transfer
块传送
Block Transfer
CPU
Cache
Main Memory



(a) Cache



(b) Main memory

Design of Cache Memory (1)
高速缓冲存储器设计

- **Design issues include:**
  - **Cache size (高速缓冲存储器大小)**
  - **Block size (块大小)**
  - **Mapping function (映射函数)**
  - **Replacement algorithm (替换算法)**
  - **Write policy (写策略)**

I/O Devices

A computer is of no use without some means of getting data into it and get information out of it.

I/O Devices

- I/O devices generally consist of two parts:
  - A *controller*
  - The device itself
- The job of the controller is to present a *simpler interface* to the OS
  - The actual device interface is hidden behind the controller

Device Drivers

- ***Each type of controller is different, so different software is needed to control each one.***
  - *Device driver is the software that talks to a controller, giving it commands and accepting responses.*
- ***Each controller manufacturer has to supply a driver for each operating system it supports***

Device Registers

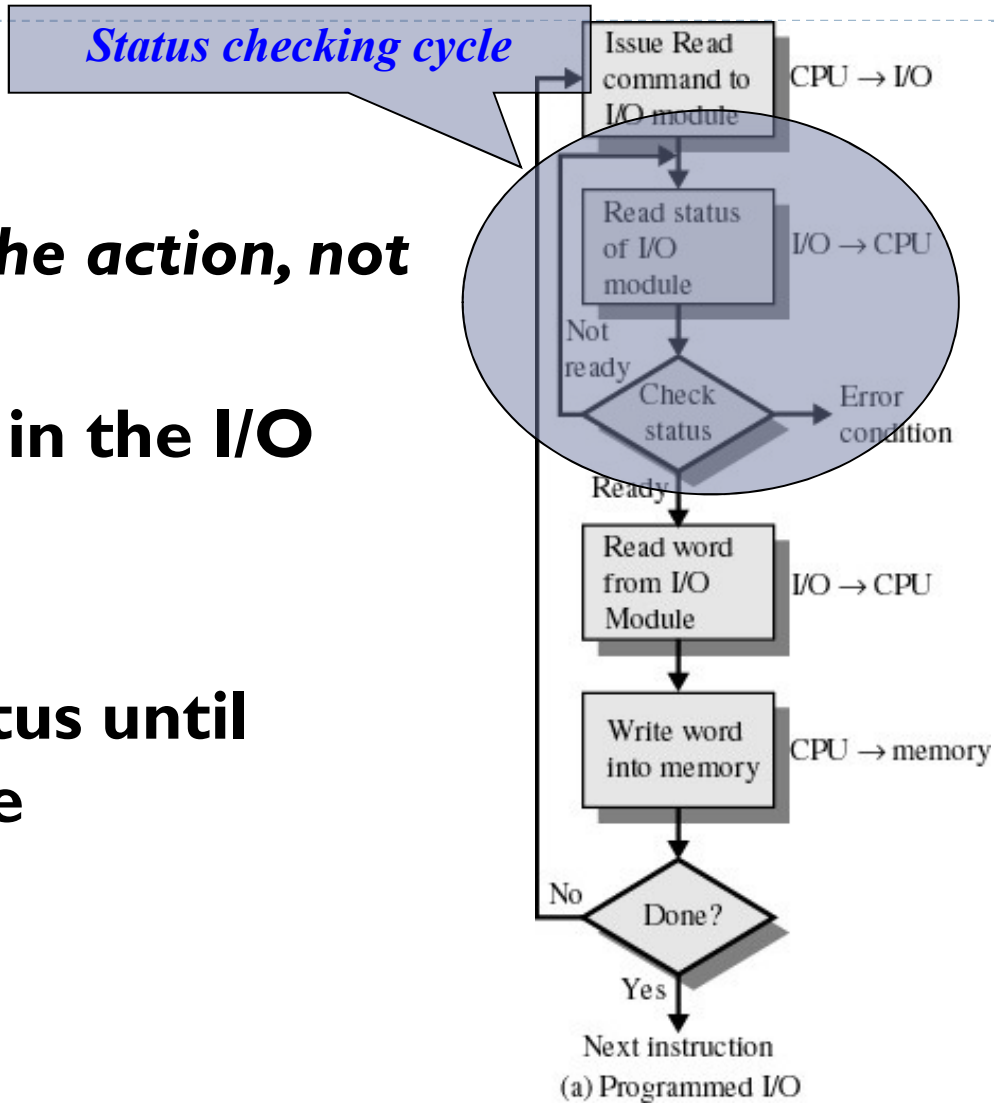
- **Every controller has a small number of registers for communication with Processors**
- **I/O addressing**
  - **Special instructions: Special IN and OUT instructions are available in kernel mode to allow drivers to read and write the registers**
  - **Memory-mapped I/O: The device registers are mapped into the operating system's address space**

I/O Communication Techniques

- *Programmed I/O*
  可编程I/O
- *Interrupt-Driven I/O*
  中断驱动I/O
- *Direct Memory Access (DMA)*
  直接内存存取

Programmed I/O

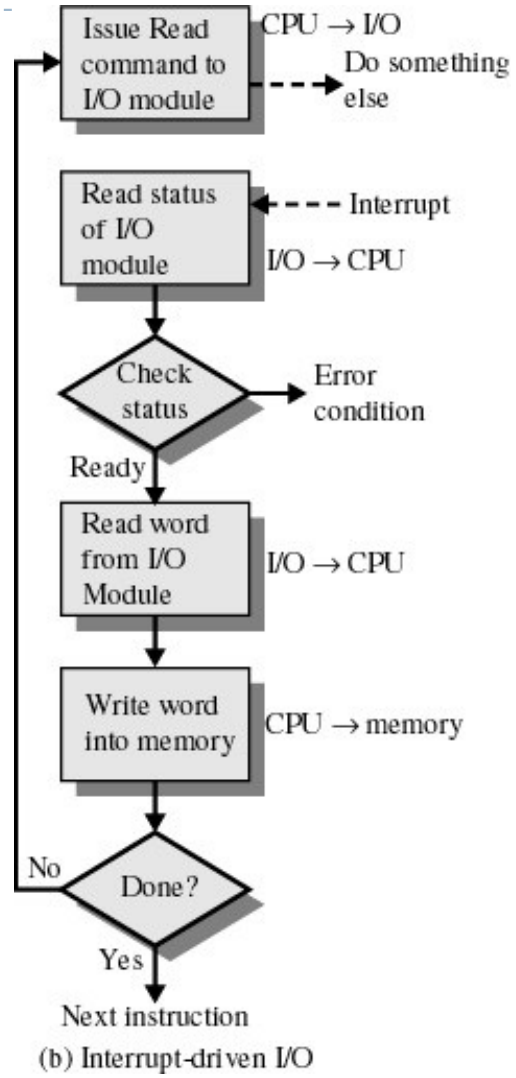
- *I/O module performs the action, not the processor*
- **Sets appropriate bits in the I/O status register**
- **No interrupts occur**
- **Processor checks status until operation is complete**



- With programmed I/O, the problem is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of the I/O module.

Interrupt-Driven I/O

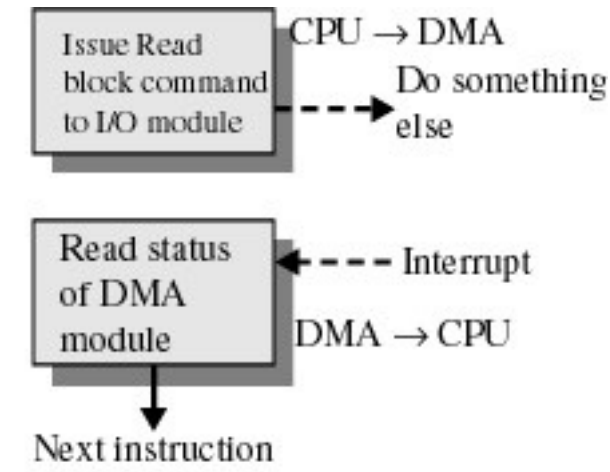
- Processor is interrupted when I/O module ready to exchange data
- Processor is free to do other work
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

Direct Memory Access (1)

- DMA chip transfers a ***block of data*** between memory and some device controller
  - Without constant CPU intervention
- An ***interrupt*** is sent when the task is complete
- The CPU is only involved at the beginning and end of the transfer



(c) Direct memory access

Direct Memory Access (2)

- **The information that should be sent to the DMA module**
  - **Whether a read or write is requested**
  - **The address of the I/O device involved**
  - **The starting location in memory to read from or write to**
  - **The number of words to be read or written**