北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Computer Organization and Architecture

Chapter 7

Input & Output

School of Computer Science（National Pilot Software Engineering School）
AO XIONG (熊翱)
xiongao@bupt.edu.cn

Preface

We have learned:

- Basic Concepts and Computer Evolution 基本概念和计算机发展历史
- Performance Issues 性能问题
- Top level view of computer function and interconnection 计算机功能和互联结构顶层视图
- Cache Memory cache存储器
- Internal Memory 内部存储器
- External Memory 外部存储器
  - Magnetic Disk 磁盘
  - RAID RAID技术
  - SSD 固态硬盘
  - Optical Memory 光存储
  - Magnetic Tape 磁带存储

Preface

We will focus the following contents today:

- Input & Output 输入和输出
  - What is the operation principle of I / O system? I / O 系统的操作原理是什么?
  - How do programming I/O and interrupt-drived I/O work? 编程式I/O和中断式I/O如何工作?
  - What is the DMA? 什么是DMA?
  - How to improve the performance of I/O system? 如何提高? I / O 系统的性能?

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

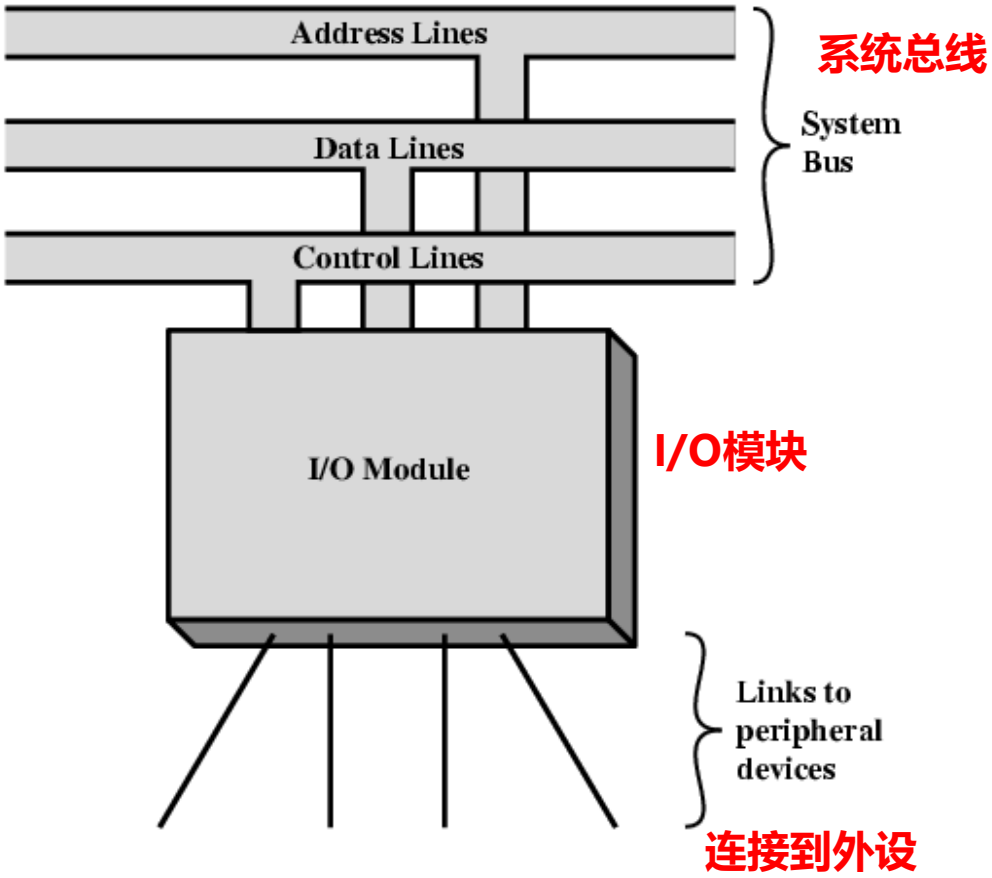
Input/Output Problems I/O的问题

- Peripherals are used for information exchange between computer and environment 外设用于计算机和环境的信息交互
- Wide variety of peripherals 外围设备范围很广
  - Delivering different amounts of data 传送不同数量的数据
  - In different formats 不同的格式
  - At different speeds 不同的速率
- Almost all slower than CPU and RAM 几乎所有的比CPU和RAM都慢
- Need I/O modules 需要I/O模块
  - Connection to processor and memory via system bus or central exchanger 通过系统总线与处理器和存储器进行连接
  - Connect with one or more peripherals through a dedicated data line 通过专用的数据线与一个或多个外设连接

Function of I/O modules

- Connecting peripherals with system bus 将外设和系统总线互连
- Not only a connector, but also the communication logic between peripheral devices and system bus 不只是一个连接器，还需要完成外设和系统总线之间的通信逻辑
  - many types of peripherals 外设种类多
  - amount of data transmitted varies greatly 数据量差别很大
  - speed varies greatly 速度也相差很大
- I/O modules are required to resolve the differences between peripherals and processors 需要I/O模块来解决外设和处理器之间的差异

Generic Model of I/O Module 通用的IO模块模型



- I/O模块的基本模型
- 通过系统总线和处理器、存储器进行连接
- I/O模块向下连接到具体的外设。可能会连接若干个外设

Types of peripherals 外设类型

- Human readable 人机交互
  - Screen, printer, keyboard 屏幕，打印机，键盘
- Machine readable 机器间交互
  - Monitoring and control 监视和控制
- Communication 通信
  - Modem 调制解调器
  - Network Interface Card (NIC) 网络接口卡
- Can also be categorized as 也能按下面进行分类
  - Input devices 输入设备
  - Output devices 输出设备
  - Input/Output devices 输入/输出设备

Interface of peripherals 外设接口

SCSI接口，硬盘、光驱和系统的接口

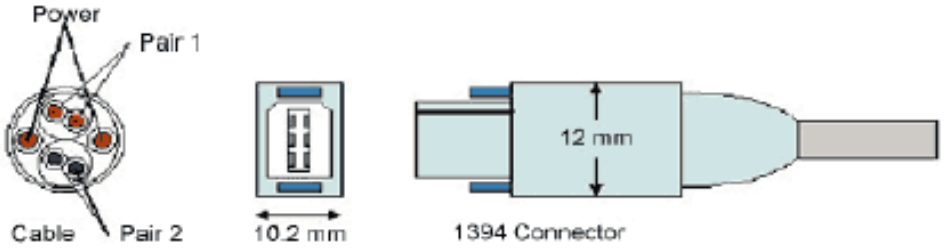
PS/2，老式键盘的接口

15针显示器接口

USB

并口，老式打印机接口

串口，低速数据传输，一般用于命令行的输入



1394接口，也称为**Firewire**，是一个高速串口，适合传输数字图像

USB（Universal Serial Bus ）interface USB接口

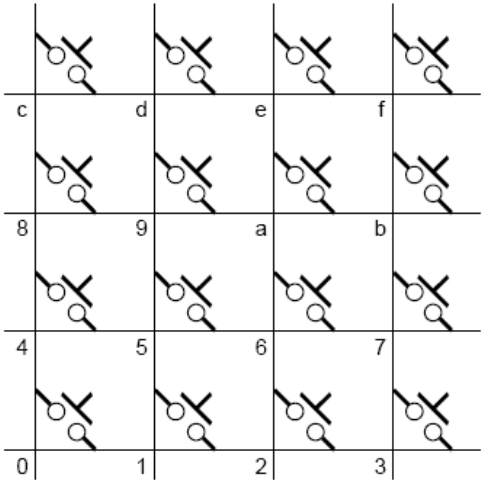
USB type-c

- 从标准上分为USB1.0、USB2.0和USB3.0
  - USB1.0：1.5Mbps
  - USB2.0：480Mbps
  - USB3.0：5.0Gbps
- 硬件接口类型上分为标准A型、标准B型和标准C型
  - A型：普通型，发展出了Mini-A、Mini-B、Micro-A、Micro-B
  - B型：打印机、扫描仪、USB HUB等外部设备
  - C型：也称为type-C接口，视频、USB、供电等共用一个接口

Chrematistics of USB interface USB接口的特点

- Hot plug：Plug and Play 热插拔，即插即用
- Portability：Small equipment, easy to carry 便携性，设备小便于携带
- Standard unification：Peripherals can be connected to personal computers with the same interface 标准统一，外设用同样的接口与个人电脑连接
- Multiple devices connectivity：More devices can be connected through USB-HUB expansion 多设备连接，通过USB-HUB扩展，可以接入更多的设备

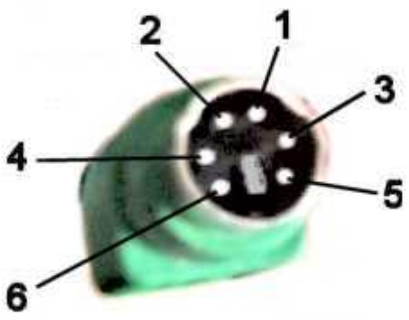
Keyboard – input device 键盘-输入设备



- 早期的键盘接口
- 采用圆形的6针
  - Pin1：数据
  - Pin5：时钟
- 大部分键盘都不用这个接口
  - USB
  - 蓝牙

PS/2 keyboard connector (MINI-DIN6)

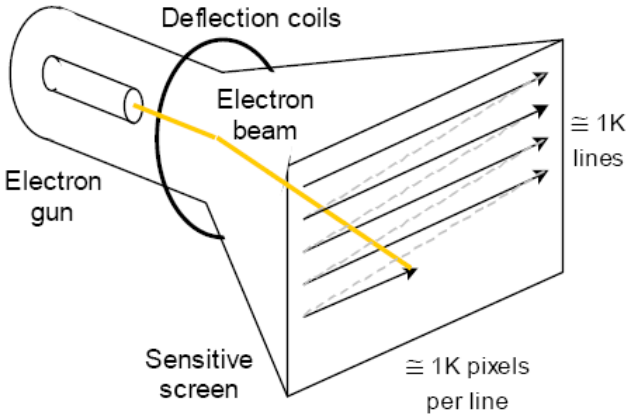
Connector Pin #	Purpose
Pin 1	KBDAT (data)
Pin 2	not used
Pin 3	GND
Pin 4	VCC (+5V)
Pin 5	KBDCLK (clock)
Pin 6	not used

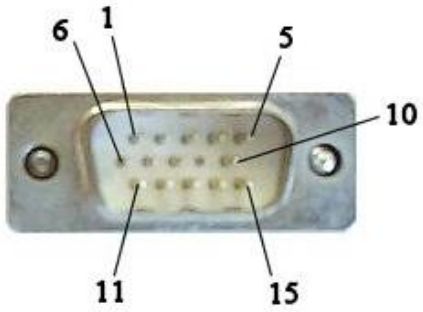


Monitor – output device 监视器-输出设备

Pin	Name	Dir	Description
1	RED	→	Red Video (75 ohm, 0.7 V p-p)
2	GREEN	→	Green Video (75 ohm, 0.7 V p-p)
3	BLUE	→	Blue Video (75 ohm, 0.7 V p-p)
4	ID2	←	Monitor ID Bit 2
5	GND	—	Ground
6	RGND	—	Red Ground
7	GGND	—	Green Ground
8	BGND	—	Blue Ground
9	KEY	-	Key (No pin)
10	SGND	—	Sync Ground
11	ID0	←	Monitor ID Bit 0
12	ID1 or SDA	←	Monitor ID Bit 1
13	HSYNC or CSYNC	→	Horizontal Sync (or Composite Sync)
14	VSYNC	→	Vertical Sync
15	ID3 or SCL	←	Monitor ID Bit 3

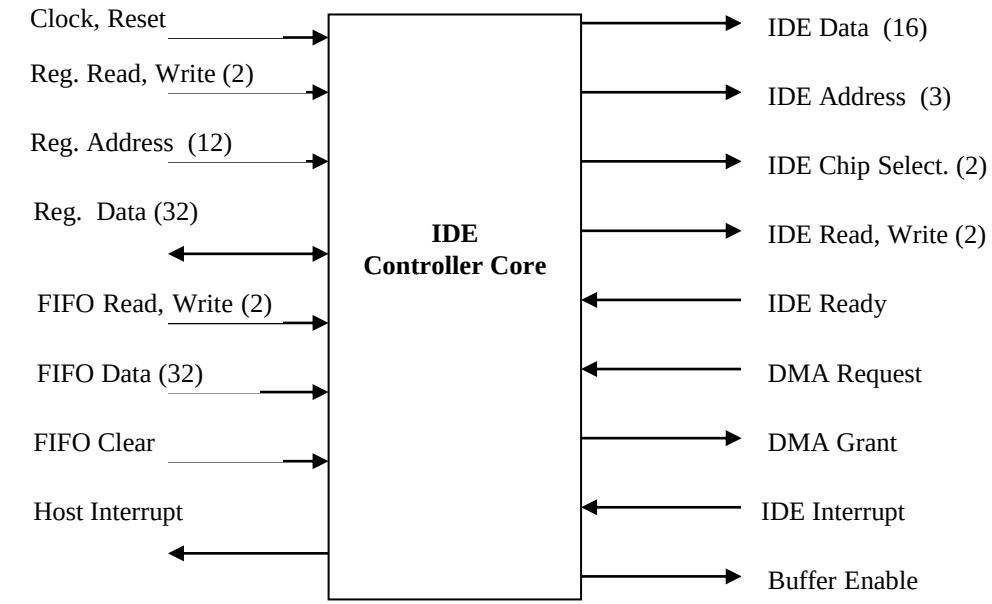
- 早期的显示接口
- 采用梯形15针
- 有些显示器还有这个接口
- 大部分都用HDMI接口

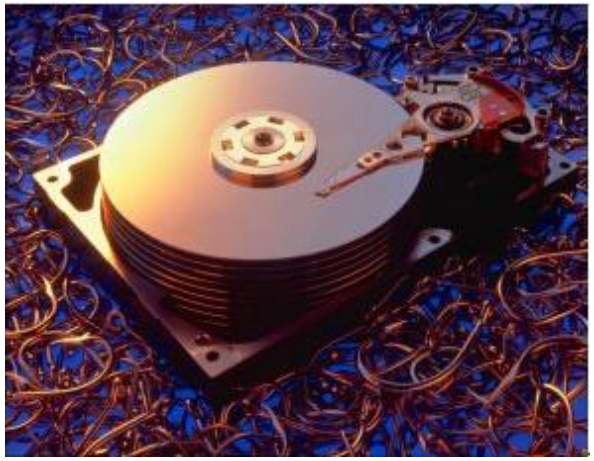




Disk drive – input/output device 磁盘驱动器-输入输出

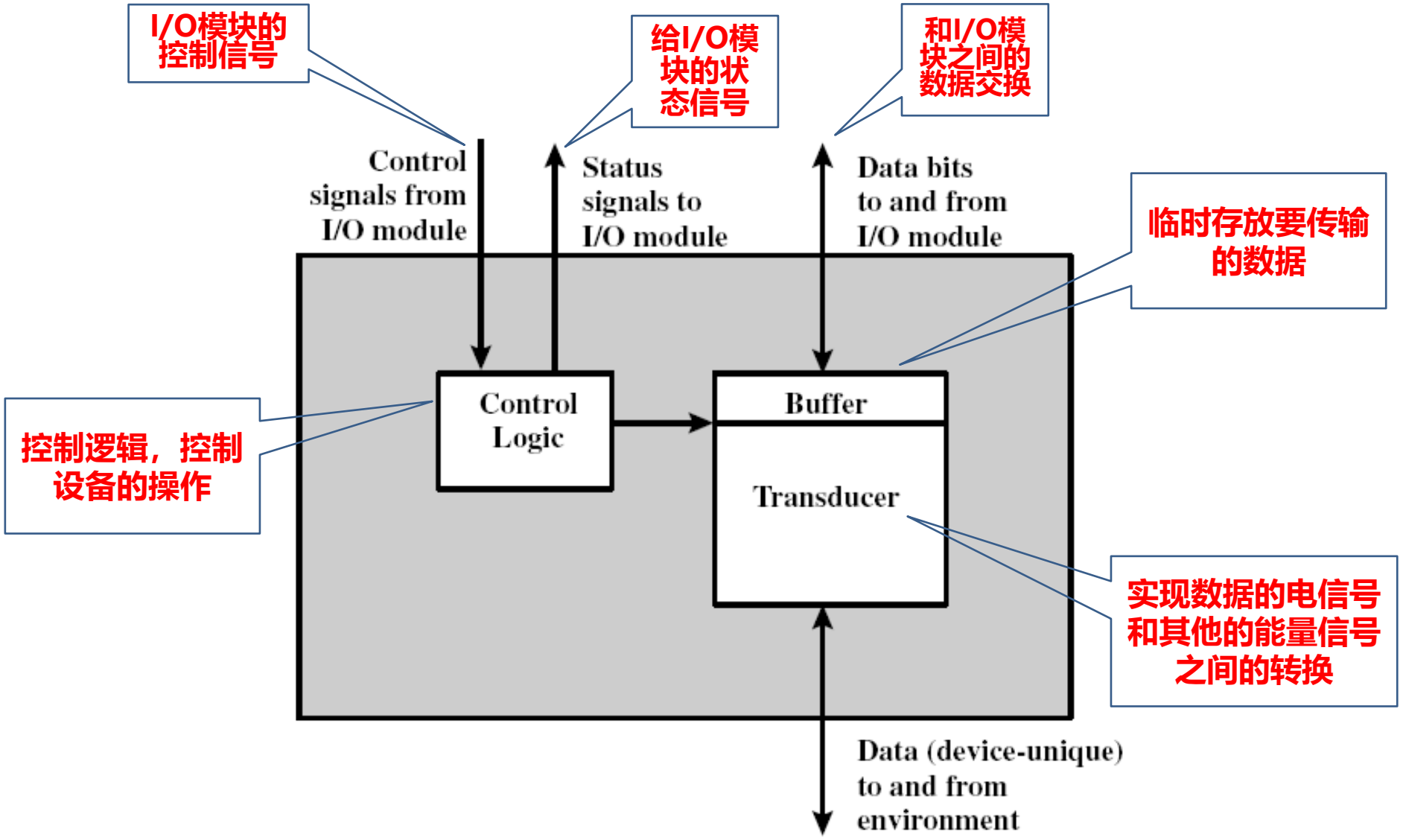
主机侧信号 IDE侧信号





- IDE（Integrated Drive Electronics）硬盘接口
- 双向接口
- 具有一般的数据传输和DMA功能

External device structure 外设结构



Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- I/O Channels and Processors I/O通道和处理器
- The External Interface 外部接口

Function of I/O module I/O模块的功能1

- Control & Timing 控制和定时
  - Control the operation of peripherals 控制外设操作
  - Sequential control 时序控制
- CPU Communication CPU通信
  - communication between the processor and peripherals 处理器和外设之间的通信
  - Command decoding, data transmission, status report, address identification 命令解码，数据传输，状态报告，地址识别

Function of I/O module I/O模块的功能2

- Device Communication 设备通信
  - communication with device 和外设的通信
  - Command, data and status 命令、数据和状态
- Data Buffering 数据缓存
  - Speed of different peripherals varies greatly 不同外设的速度差异很大
  - Conversion of transmission rate 速度匹配
- Error Detection 检错

Step of data transfer 数据传送步骤

How to transfer data from device to CPU?

1. CPU checks I/O module device status CPU检测I/O模块的状态
2. I/O module returns status I/O模块返回状态
3. If ready, CPU requests data transfer 如果就绪，CPU请求数据
4. I/O module gets data from device I/O模块从设备中获取数据
5. I/O module transfers data to CPU I/O模块传输数据给CPU

Variations for output, DMA 对于数据输出和DMA，步骤不一样

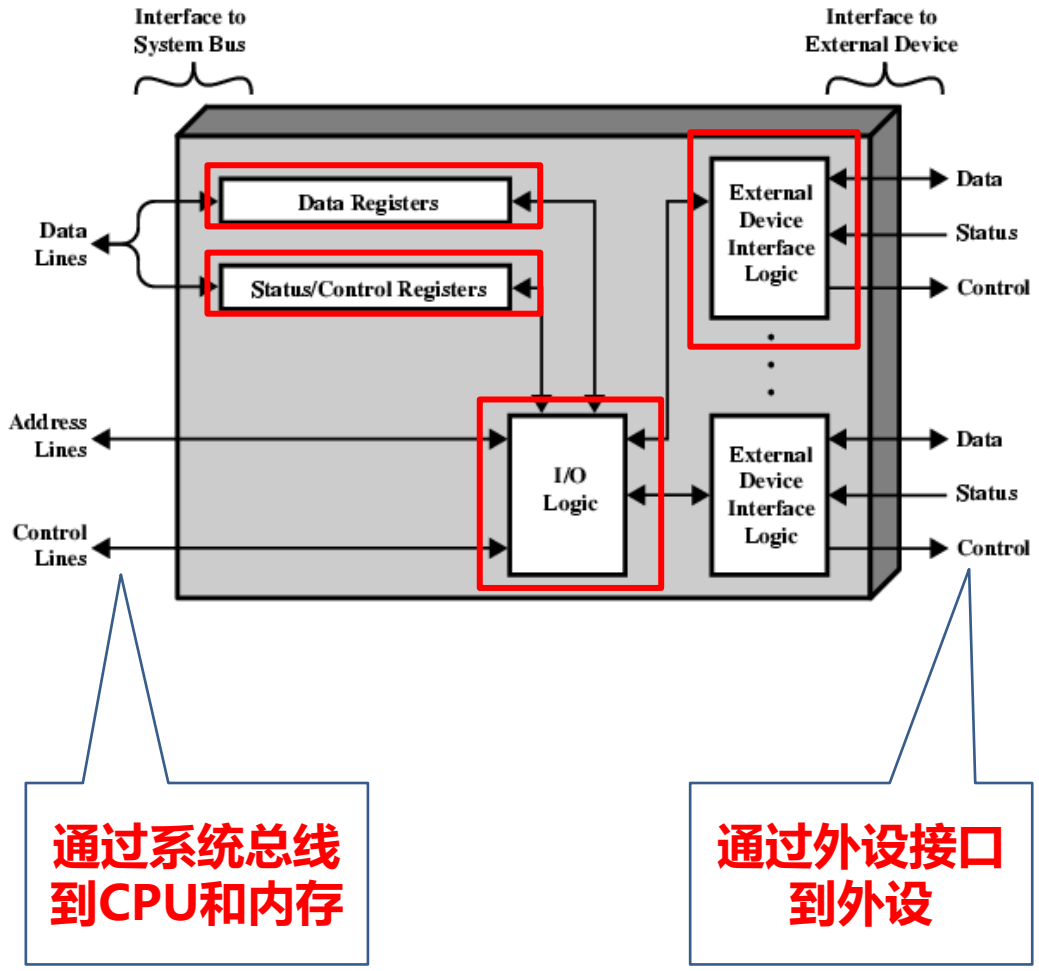
Processor communication 处理器通信1

- Command decoding 命令解码
  - processor sends instructions to the I/O module 处理器发指令给I/O模块
  - I/O module decodes the instructions to determine the operations to be completed I/O模块对指令解码，确定需要完成的操作
- Address recognition 地址识别
  - Each peripheral has an address, similar to a memory unit 每个外设都有一个地址，类似内存单元
  - Operating instructions include peripheral address处理器的操作指令包括外设地址
  - I/O module determines which device to operate on according to the address I/O模块根据地址，确定对哪个外设进行操作

Processor communication 处理器通信2

- Data transfer 数据传送
  - Data transmission is bidirectional 地址传送是双向
  - Complete through data bus 通过数据总线来完成
- Status reporting 状态报告
  - Report the status to the CPU to determine whether the current I/O operation is executed 向CPU报告状态，以决定当前的I/O操作是否进行
  - Report various error messages 报告各种错误信息

I/O module diagram IO模块框图



- I/O模块屏蔽外设的细节，简化了处理器的处理
- I/O模块连接处理器、内存和外设，对外包括和系统总线的接口以及和外设的接口
- I/O模块包括数据寄存器、状态/控制寄存器、I/O逻辑、外设接口、总线接口

I/O module structure I/O模块结构1

All I/O modules have similar structure that consists of I/O模块具有相似的结构，包括如下部分

- Data registers：data buffer to the I/O device 数据寄存器：到I/O设备的数据缓存
- Status/Control registers 状态/控制寄存器
  - Current status 当前状态信息
  - or receive control from the processor. 或者接收来自处理器的控制信息
  - Connecting to data 和数据总线相连
- Bus interface：data, address, control 总线接口，包括数据地址和控制

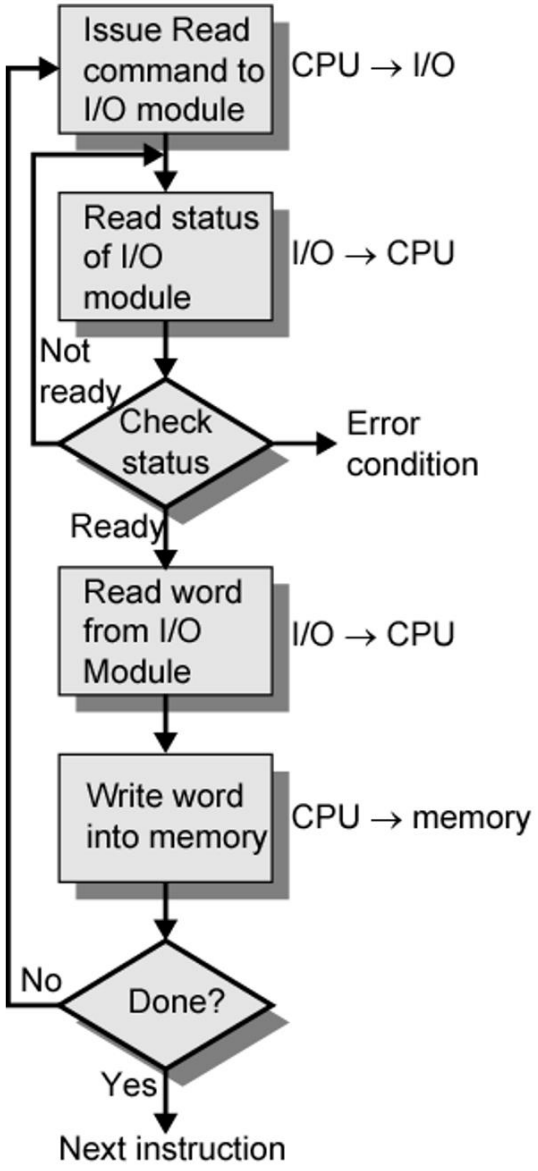
I/O module structure I/O模块结构2

- I/O logic I/O逻辑
  - Receive the command sent by the processor 接收处理器发送的操作命令
  - Receive address 接收I/O地址信息
  - Control peripherals for operation 控制设备进行操作
- External device interface 外部设备接口
  - Interact with peripherals 和外设进行交互
  - Including data, status and control 包括数据、状态和控制

Three I/O techniques 三种I/O技术

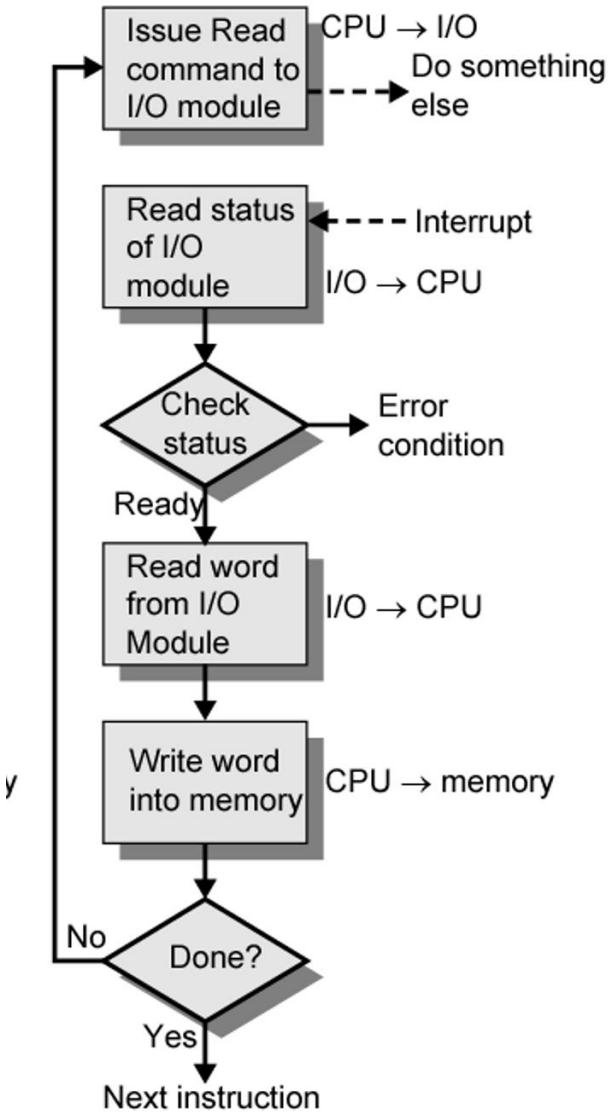
- Programmed I/O 编程式I/O
  - The processor executes a program to control the I/O operation 处理器执行程序来控制I/O操作
  - The processor needs to wait while I/O processing the command 当I/O处理命令的时候，处理器需要等待
- Interrupt-driven I/O 中断式I/O
  - Processor issues an I/O command, and continues to execute other instructions until the interrupt occurs 处理器发送I/O命令，然后继续执行指令，直到中断发生
- Direct memory access (DMA) 直接存储器访问
  - The data transfer without processor involvement 数据传输不需要处理器参与

Programmed I/O



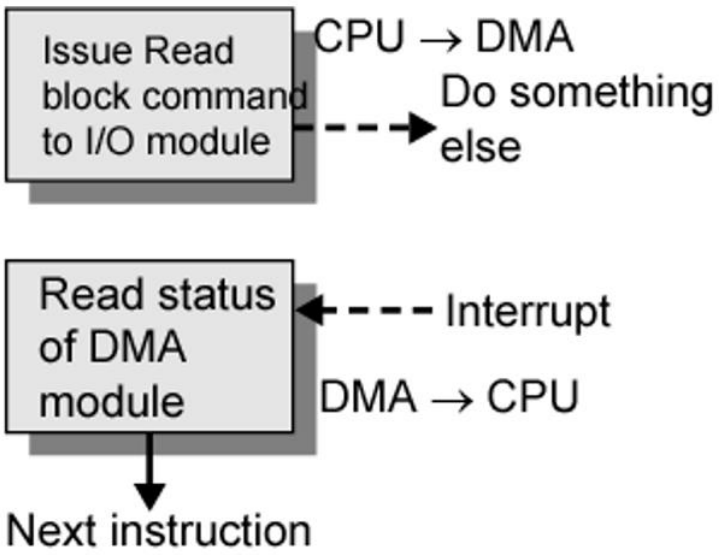
- CPU先发一个读命令给I/O模块，然后读I/O状态
- 如果I/O就绪，就等待I/O准备数据
- I/O准备好数据后，CPU从I/O模块中读取数据，然后写入存储器
- 在I/O模块准备的时候，CPU需要等待

Interrupt-driven I/O



- CPU发送读命令之后，继续做其他事情
- I/O模块准备好之后，发一个中断给CPU
- CPU收到中断后，检查状态，然后从I/O模块中读取数据，并写入存储器
- 在I/O模块准备的时候，CPU可以执行其他操作，不需要等待

Direct memory access (DMA)



- CPU发送一个指令给DMA控制器，然后继续做其他指令
- DMA模块来完成数据从外设到存储器的传输
- DMA完成之后，发送一个中断给CPU，并将总线控制权还给CPU

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

Programmed I/O 编程式IO

- CPU has direct control over I/O CPU对I/O有直接的控制权
  - Sensing status 设置状态
  - Read/write commands 读写命令
  - Transferring data 传输数据
- CPU waits for I/O module to complete operation CPU需要等待I/O模块完成操作
- Wastes CPU time 浪费CPU时间

Detail

- CPU requests I/O operation CPU请求I/O操作
  - I/O module performs operation I/O模块执行操作
  - I/O module sets status bits I/O模块设置状态
- CPU checks status bits periodically CPU周期性检查状态
  - I/O module does not inform CPU directly I/O模块不直接通知CPU
  - I/O module does not interrupt CPU I/O模块不会中断CPU
- CPU may wait or come back later CPU需要等待或者一会儿再处理

I/O commands

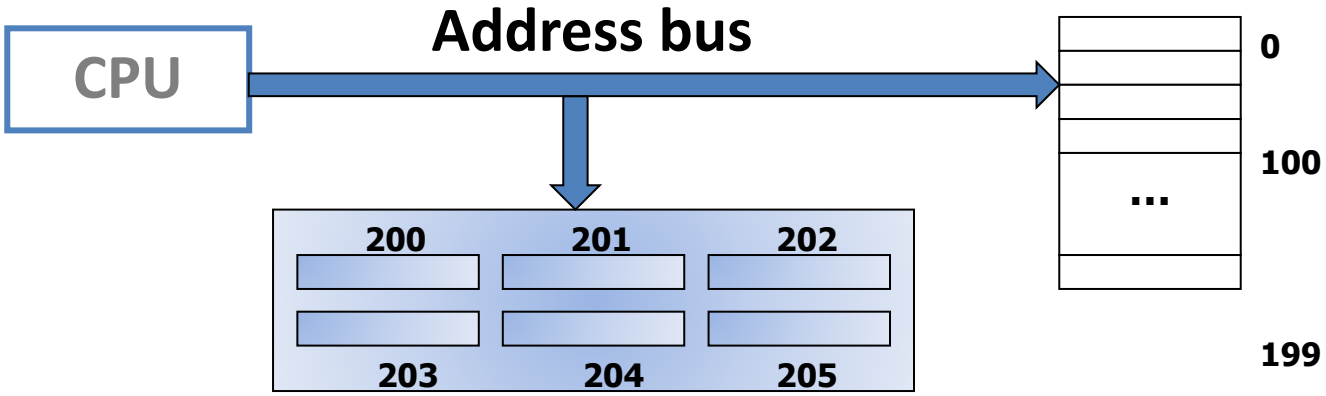
- CPU sends out address CPU发送地址
  - Identifies module (& device if >1 per module) 确定模块。模块中的设备数量大于1的时候，根据地址确定设备
- CPU sends out command CPU发送指令
  - Control - telling module what to do 控制指令-告诉I/O模块做什么
  - Test - check status 测试指令-检测状态
  - Read/Write —transfers data via buffer from/to device读写指令；通过缓冲区传送数据

I/O addressing

- Typically, there might be many I/O devices connected through I/O modules 一般情况下，有多个设备通过I/O模块连接到计算机
- Each device is given a unique address 每个都有唯一的地址
- The I/O instruction contains the address of the desired device I/O指令中会包含需要操作的设备的地址
- Two types of addressing mode are used: 两种类型的地址模式
  - Memory-mapped I/O 内存映射I/O地址
  - Isolated I/O 独立的I/O编址

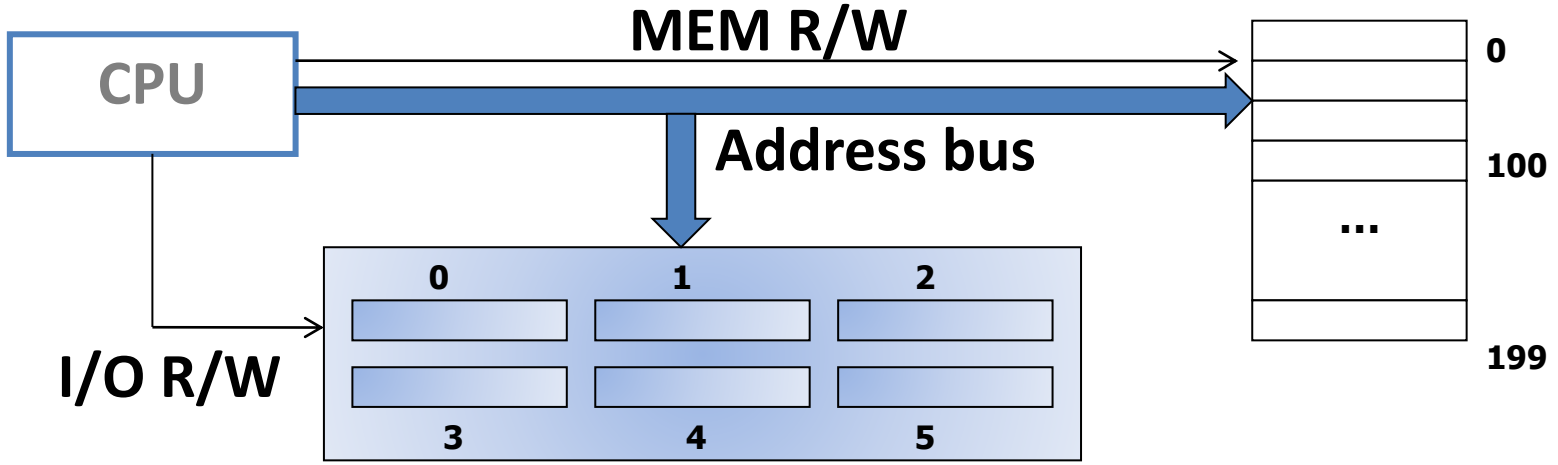
Memory mapped I/O 内存映射I/O

- The total addressable space is divided into two portions: one for memory and one for I/O 整个地址空间分为2个部分：一部分给内存，一部分给I/O
- Use same instructions to transfer data 能够使用相同的指令去传输数据
- Need not extra control signals and instructions 不需要额外的信号和指令

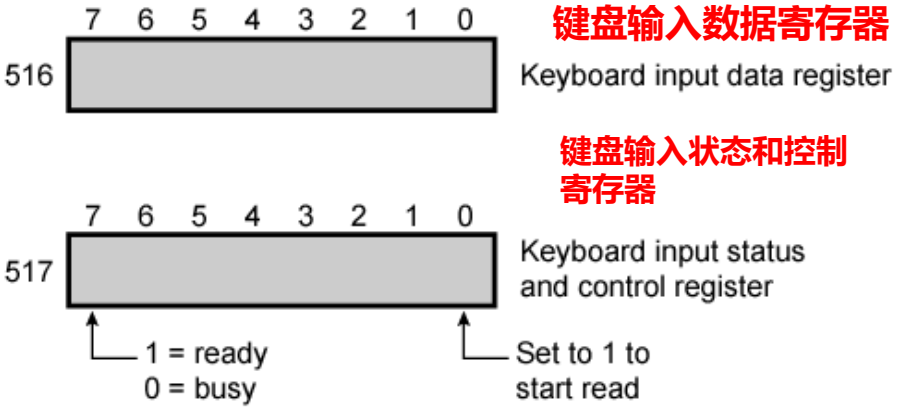


I/O mapping - isolated I/O 独立映射I/O

- Separate address spaces 独立地址空间
- Need I/O or memory select lines (M/IO#) 需要I/O或者内存选择线
- Special commands for I/O are required 对于I/O需要特殊的指令



Memory mapped and isolated I/O



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

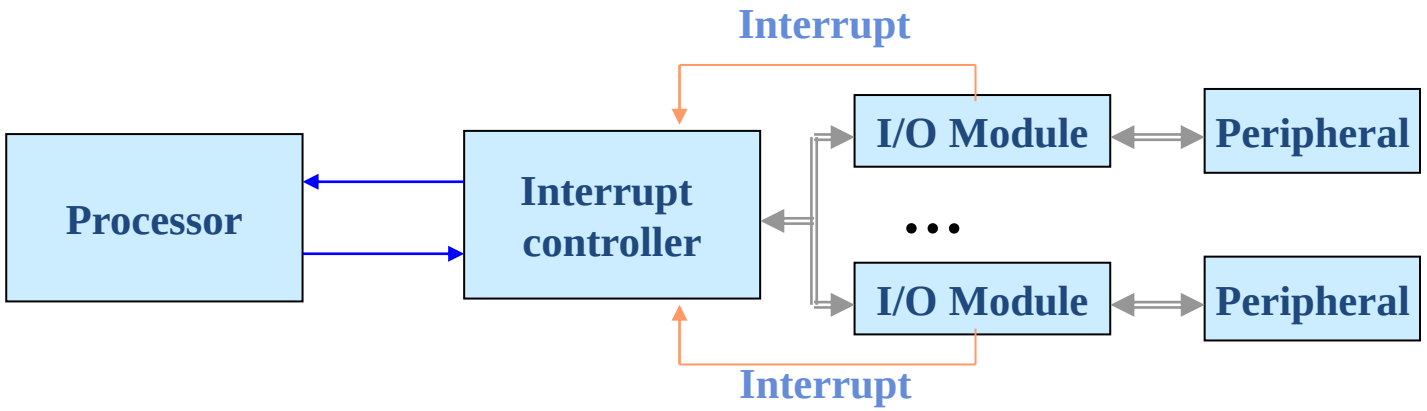
- （a）中，I/O和内存统一编址。0~511为内存单元地址，512~1023为I/O地址。两个占用同一个地址空间，因此可以用同一个指令进行内存和I/O操作
- （b）中，存储器和I/O独立编址，所以需要有明确的访问I/O的指令，Load I/O等

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

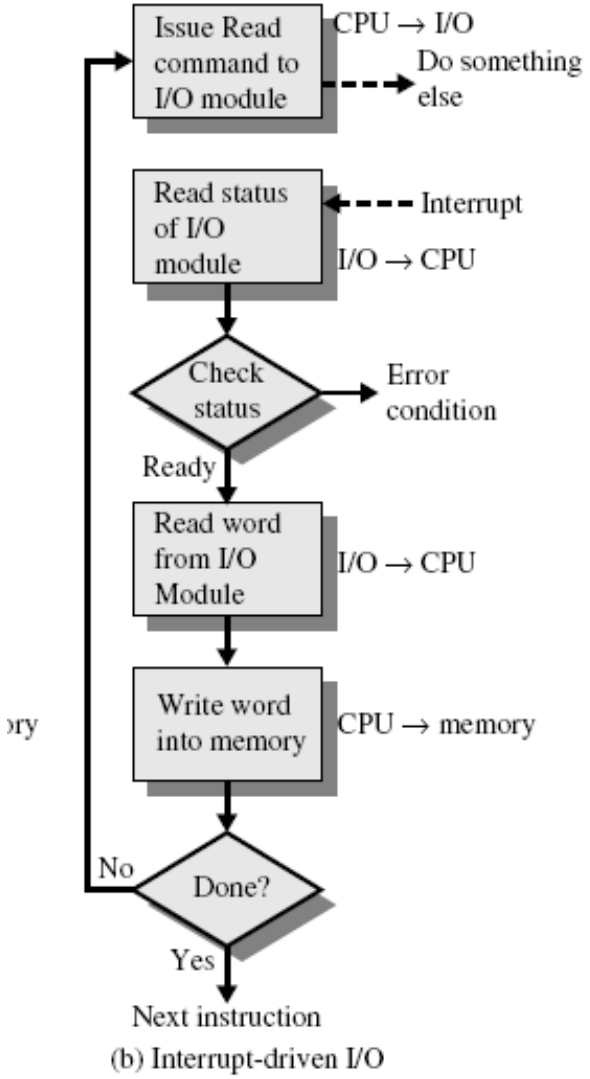
Interrupt driven I/O 中断驱动I/O

- Programmed I/O reduces CPU efficiency 编程式I/O效率低
- Interrupt driven I/O overcomes CPU waiting 中断式I/O克服了CPU等待的问题
- No repeated CPU checking of device CPU不需要重复检测设备
- I/O module interrupts when ready I/O模块就绪时产生中断



Interrupt processing -1 中断处理过程1

- CPU: issues I/O command CPU发出I/O指令
- CPU: continues with other tasks CPU继续工作
- The module: receives the command and works on the task I/O模块：接收指令，处理工作
- When finished, signals CPU an interrupt 处理完后，给CPU发一个中断

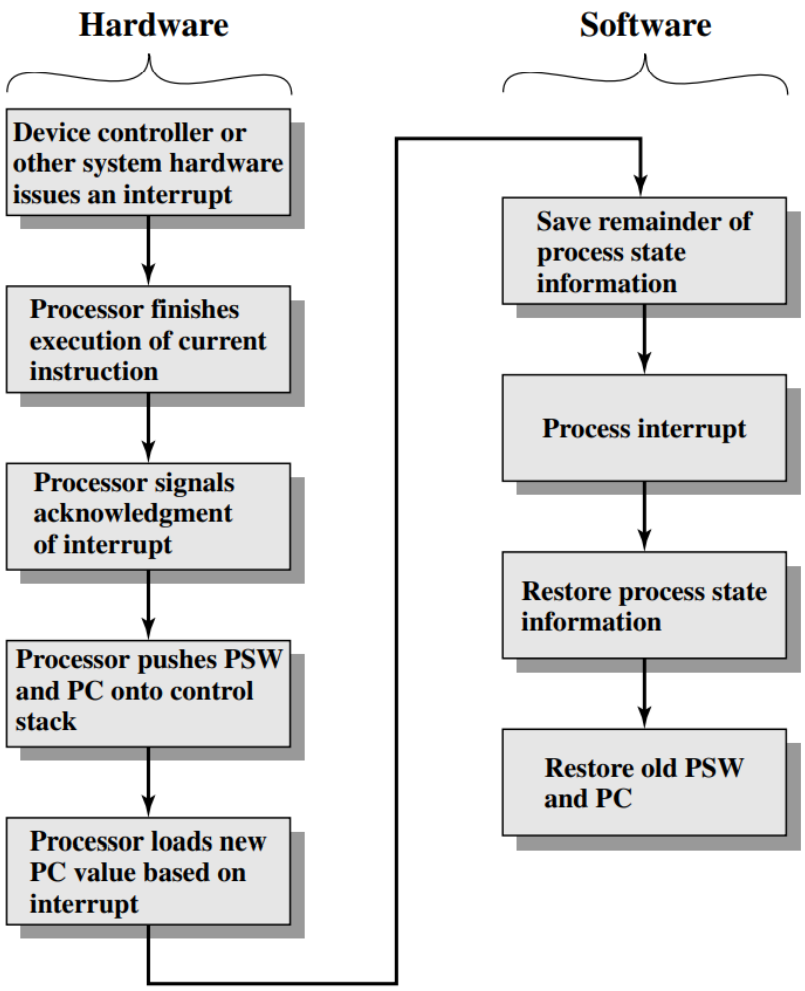


(b) Interrupt-driven I/O

Interrupt processing -2 中断处理过程2

- CPU: checks the interrupt at the end of each instruction cycle
  CPU在每个指令结束的时候进行中断检测
- CPU: if interrupt occurs 如果有中断
  - CPU saves the context CPU保存当前指令的上下文
  - Executes interrupt service routine 执行中断服务例程
  - E.g. read the data from I/O and stores in memory) 比如从I/O模块中读取数据并保存到内存中
- CPU: restores the context 恢复上下文
- CPU: continues on its primary task 继续执行主任务

Interrupt processing diagram 中断处理流程

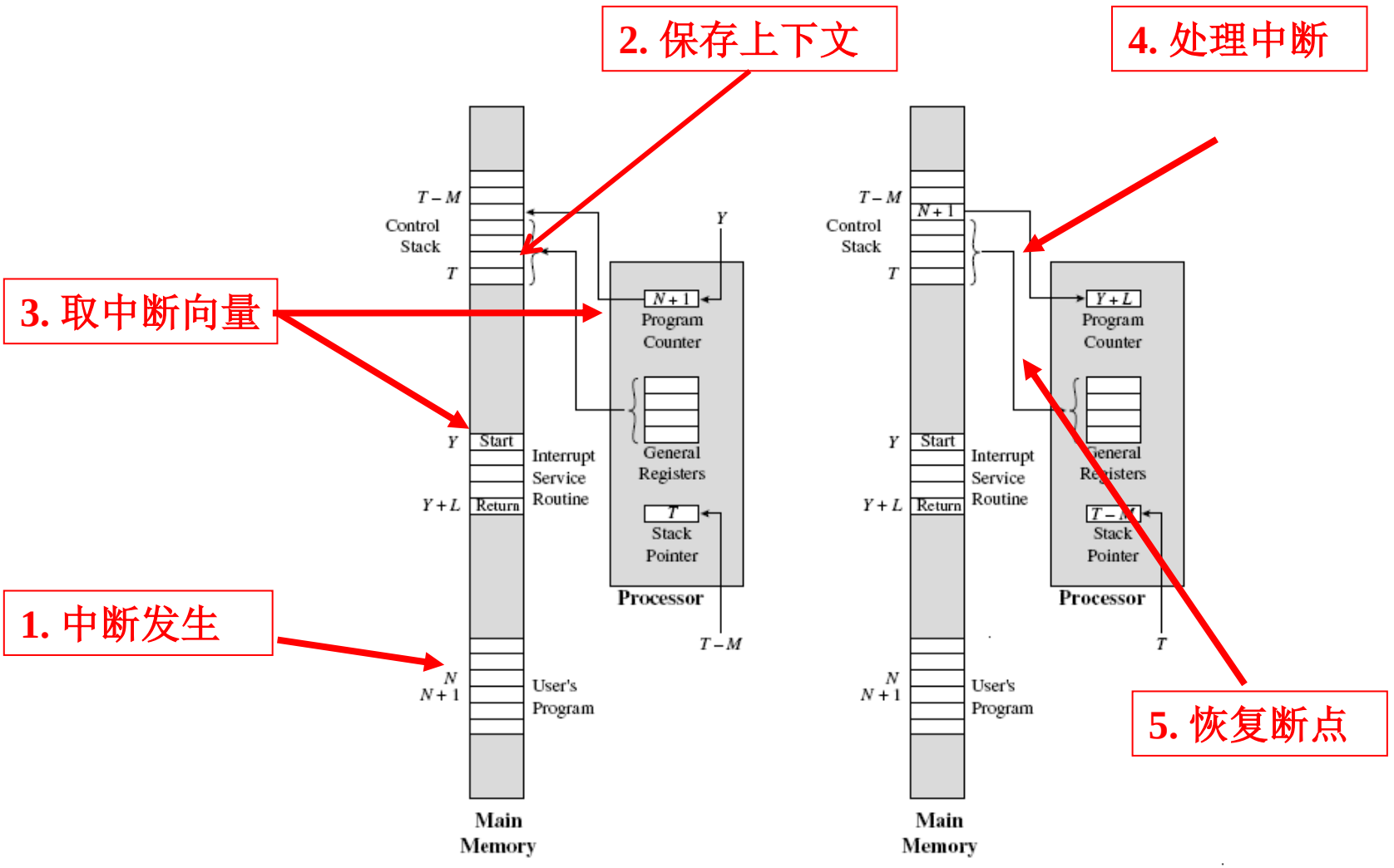


- 设备控制器或其他系统硬件发出中断
- 处理器处理完当前的指令后，进行中断检测。如果发现有中断，就开始处理中断
  - 程序状态字PSW和程序计数器PC压栈，并取得中断对应的指令地址，存入PC
  - 保存其他的处理器状态信息
  - 中断处理
- 中断处理完成后，恢复处理器状态信息，从栈中恢复PSW和PC，继续执行中断前的指令

CPU viewpoint CPU视角

- Issue read command 发出读指令
- Do other work 做其他工作
- Check for interrupt at end of each instruction cycle 每个指令结束后检查中断
- If interrupted 如果有中断
  - Save context (registers) 保存上下文信息，主要是寄存器
  - Process interrupt 处理中断
    - Fetch data & store 取数、保存
- Restore PSW and PC 恢复PSW 和 PC

Interrupt process – CPU viewpoint CPU视角的中断处理



Design issues 设计问题

- Two main design issues
- How do you identify the module issuing the interrupt? 如何确定是哪个I/O模块发出的中断？
- How do you deal with multiple interrupts? 如何处理多个中断？
  - i.e. an interrupt handler being interrupted 也就是说，一个中断被中断后如何处理？

Types of device identification 辨别中断设备的方法

- For device identification, there are four general categories of techniques are in use 对于设备辨别，有四种技术
  - Multiple interrupt line – each device has a IRQ 多个中断线，每个设备有独立的的中断请求
  - Software poll 软件轮询
  - Hardware poll (Vectored interrupt) 硬件轮询，属于向量中断
  - Bus arbitration (Vectored interrupt) 总线仲裁，也是向量中断

Software poll 软件轮询

- An interrupt service routine polls each I/O module to determine which module caused the interrupt 中断服务程序轮询每个I/O模块，以确定是哪个模块导致了中断
  - TEST I/O command TEST I/O命令
  - Status register：After an interrupt is issued by the I/O module, a status register is written. The processor polls this status register 状态寄存器：I/O模块产生中断后，写入状态寄存器。处理器查询状态寄存器，确定中断的I/O模块
- Once the correct module is identified, the processor branches to a device-service routine specific to that device 一旦识别出正确的模块，处理器将分支到该设备的设备服务例程
- Disadvantage: time consuming 缺点：费时

Hardware poll 硬件轮询

- All I/O modules share a common interrupt request line 所有I/O模块共享中断请求线
- When the processor senses an interrupt, it sends out an interrupt acknowledge 处理器发现中断，发出一个中断应答信号
  - This signal propagates through a series of I/O modules until it gets to a requesting module 信号通过I/O模块传播，直到到达请求的模块
  - The requesting module typically responds by placing a word (vector) that identifies the device 请求模块通过设置一个向量来响应
  - The processor uses the vector as a pointer to the appropriate device-service routine 处理器用这个向量作为指针指向设备服务程序
- Called vectored interrupt 称为向量中断

Bus arbitration 总线仲裁

- An I/O module must first gain control of the bus before it can raise the interrupt request line I/O模块必须首先获得总线控制权，然后才能触发中断请求线
- When the processor detects the interrupt, it responds on the interrupt acknowledge line 当处理器检测到中断时，它会在中断确认线上做出响应
- The requesting module then places its vector on the data line 请求模块将其向量放置在数据线上

Multiple interrupts 多重中断

- FIFO，no priority 先来先处理，没有优先级
- Multiple interrupt line ： each interrupt line has a priority 多个中断线：每个中断线有一个优先级
- Hardware poll or software poll: order of polling determines the priority 硬件或软件轮询：轮询顺序决定了优先级
- Bus arbitration: Arbitration can be conducted in priority mode 总线仲裁：仲裁方式决定了优先级模式

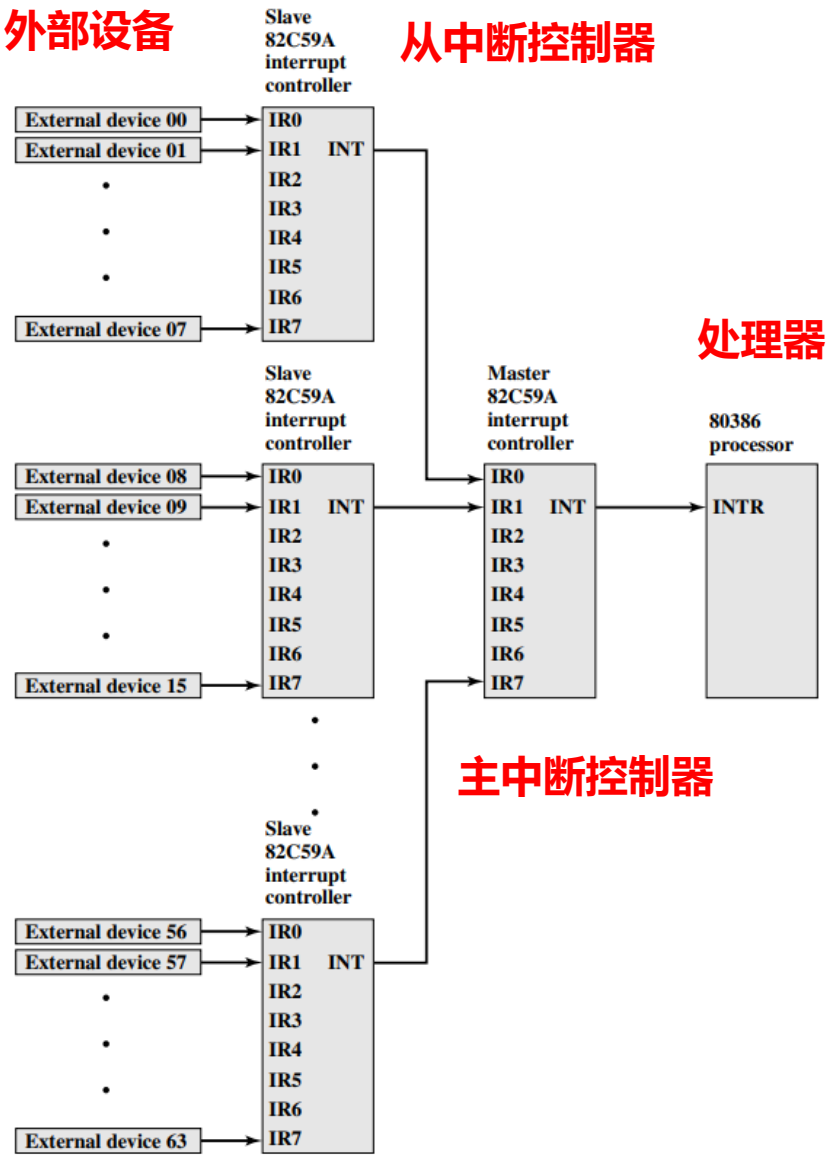
Example

Intel interrupt controller 82C59A Intel中断控制器

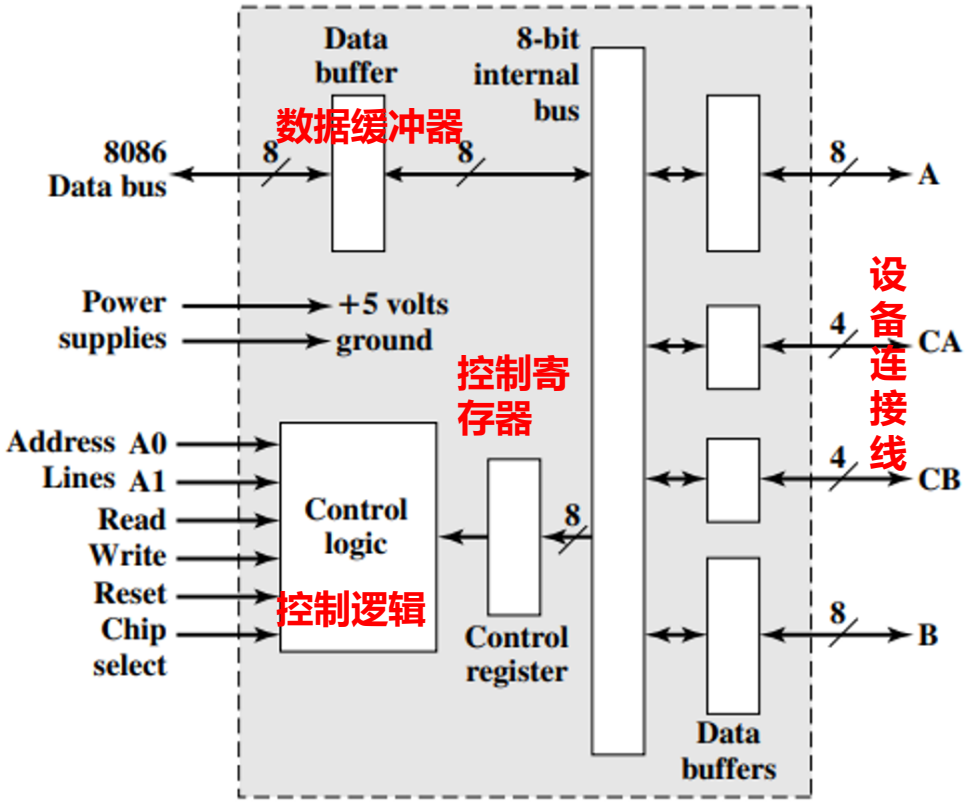
- 80386 has one interrupt request (INTR）and one interrupt answer (INTA） 80386只有一个中断请求和一个中断应答引脚
- Use one Intel 82C59A interrupt controller 配置了一个Intel 82C59A的中断控制器
  - Intel 82C59A has 8 interrupt lines 有8个中断线
  - Interrupt lines can be extended by cascading 中断线能级联扩展
- 82C59 is responsible for managing interruptions 管理中断

82C59A interrupt controller diagram

- 通过级联方式进行扩展
- 配置了8个82C59A从中断控制器，和一个主控制器
- 8个从中断控制器连接到1个主中断控制器，主中断控制器再连接到386处理器
- 通过级联后，80386可以连接最多64个外部设备

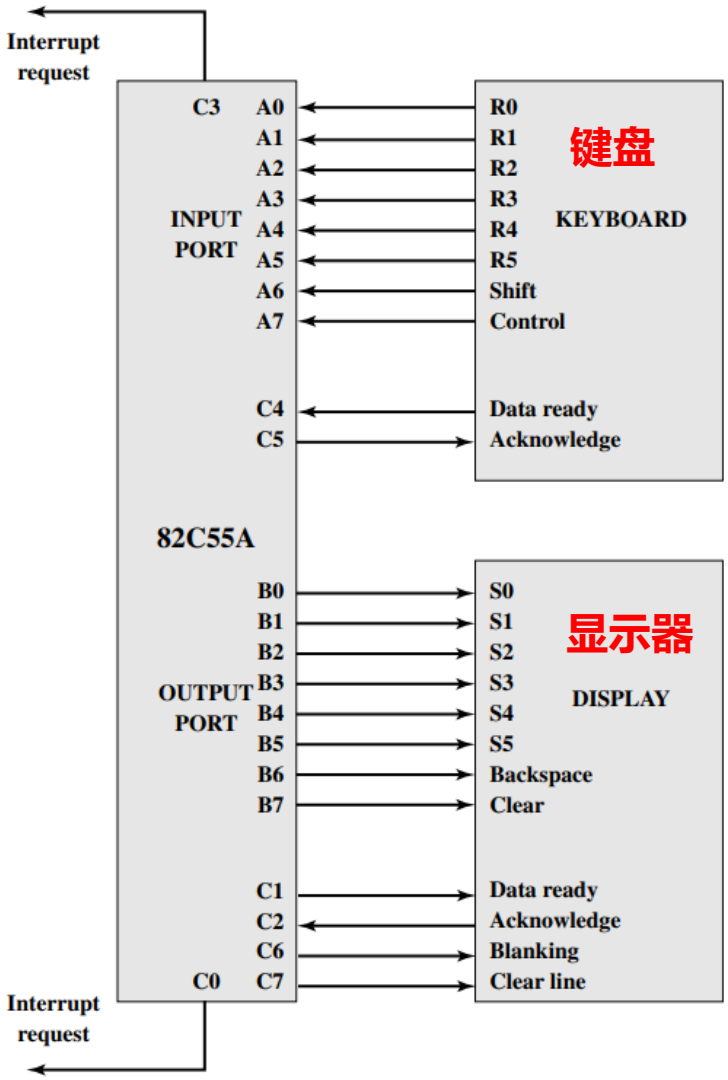


Intel 82C55A programmable peripheral interface



- 单芯片通用可编程I/O模块
- 包括了数据缓冲区，控制逻辑，控制寄存器
- 三组共24根线，用于和设备相连
- 不同的模式下，C组线可以用于不同的目的，或者用于数据，或者用于控制
- 控制寄存器可以通过处理器进行编程，实现不同的操作方式和配置

Keyboard/Display Interfaces to 82C55A



- 通过82C55A来实现键盘和显示器的接入
- 键盘使用了82C55A 的A组I/O线
- 显示器使用了82C55A 的B组I/O线
- C组线用于进行键盘控制或显示器控制
- C组线的功能由控制寄存器来确定

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

Why use DMA – 1

- Interrupt driven and programmed I/O require active CPU intervention 中断驱动和编程I/O均需要CPU干预
  - I/O->processor->memory I/O->处理器>存储器
  - Transfer rate is limited 传输速率受限
  - CPU is tied up CPU很忙
- DMA is the answer DMA是一种方案
- DMA：Direct Memory Access 直接存储器存取
  - a module on system bus 系统总线上的模块
  - take over the system control work from the processor 接管系统控制工作
  - transfers data between memory and I/O module 在内存和I/O之间直接传输数据

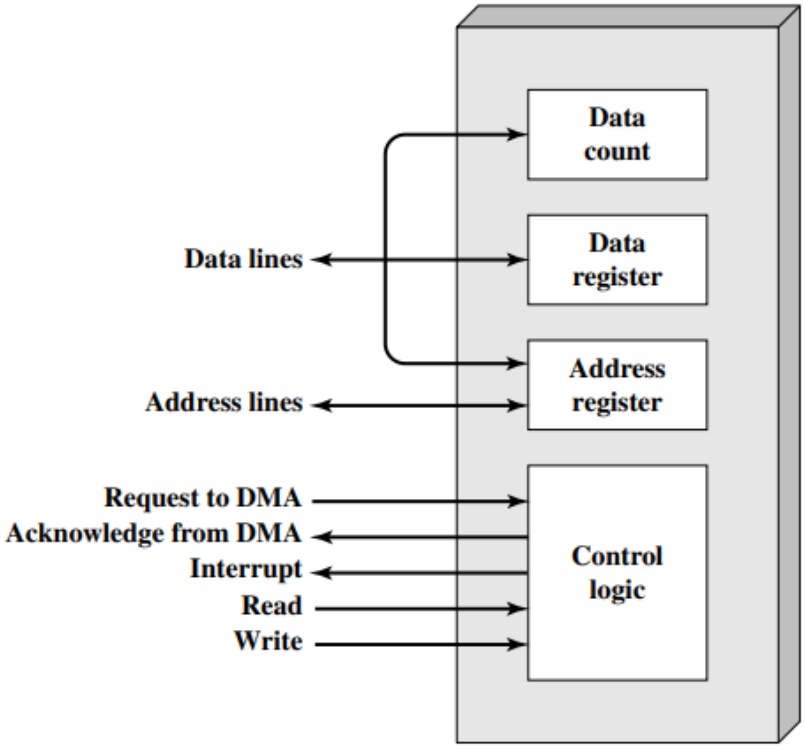
Why use DMA – 2

- Both interrupt driven and programmed I/O require the continued involvement of the CPU in the I/O operation I/O操作中，中断驱动和编程I/O都需要CPU持续的参与
  - I/O device → CPU → Memory
  - I/O device ← CPU ← Memory
- DMA takes the CPU out of the task DMA代替CPU来接管这个任务
  - I/O device → Memory
  - I/O device ← Memory

DMA function

- Additional Module (hardware) on bus 总线上附加的硬件模块
- DMA controller takes over from CPU for I/O DMA控制器代替CPU完成I/O操作
  - DMA need system bus to complete data transmission DMA需要系统总线传输数据
  - It must use the bus only when the processor does not need it 它只能在处理器不需要总线的时候才能用总线
  - Or it must force the CPU to suspend operation temporarily – referred to as cycle stealing 或者强迫CPU临时挂起，这个称谓周期窃取

Typical DMA module diagram 典型的DMA模块图



- DMA包括四个部分：数据计数，数据寄存器，地址寄存器和控制逻辑
- 控制逻辑主要实现和处理器、I/O模块的交互
- 地址寄存器和数据寄存器用于地址和数据的临时存放
- 数据计数用于进行传输数据计数

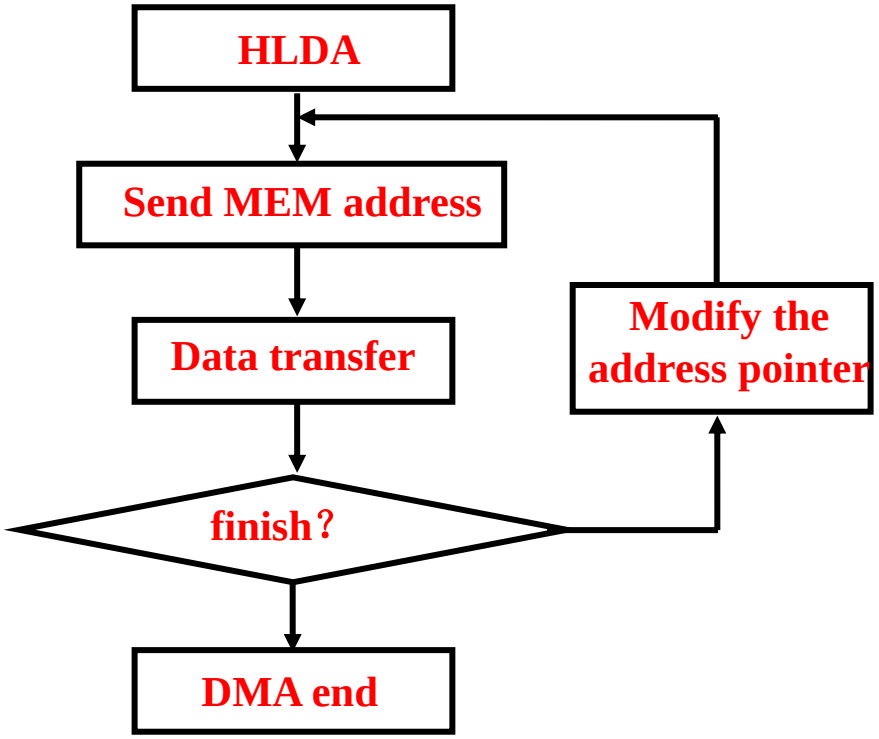
DMA operation – 1 DMA操作1

- CPU tells DMA controller CPU告诉DMA控制器
  - Read/Write 读/写指令
  - Device address 设备地址
  - Starting address of memory block for data 内存块的开始地址
  - Amount of data to be transferred 需要传输的数据数量
- CPU carries on with other work CPU继续做其他工作
- DMA controller deals with transfer DMA控制器处理传输
- DMA controller sends interrupt when finished DMA控制器完成后发出中断

DMA operation – 2 DMA操作2

- CPU, DMA controller, I/O device exchange the handshake signals CPU、DMA控制器、I/O设备交换握手信号
- DMA controller deals with the data transfer DMA控制器处理数据传输
  - I/O device to memory, memory to I/O device I/O设备到内存，或者内存到I/O设备
  - Multiple data may be transmitted at one time 一次可以传输多个数据
  - Automatic address increasing, counter updating 地址自动增加，计数器更新
- CPU only involved at the beginning and end of the transfer CPU只在传输的开始和结束时介入

DMA operation diagram DMA操作框图

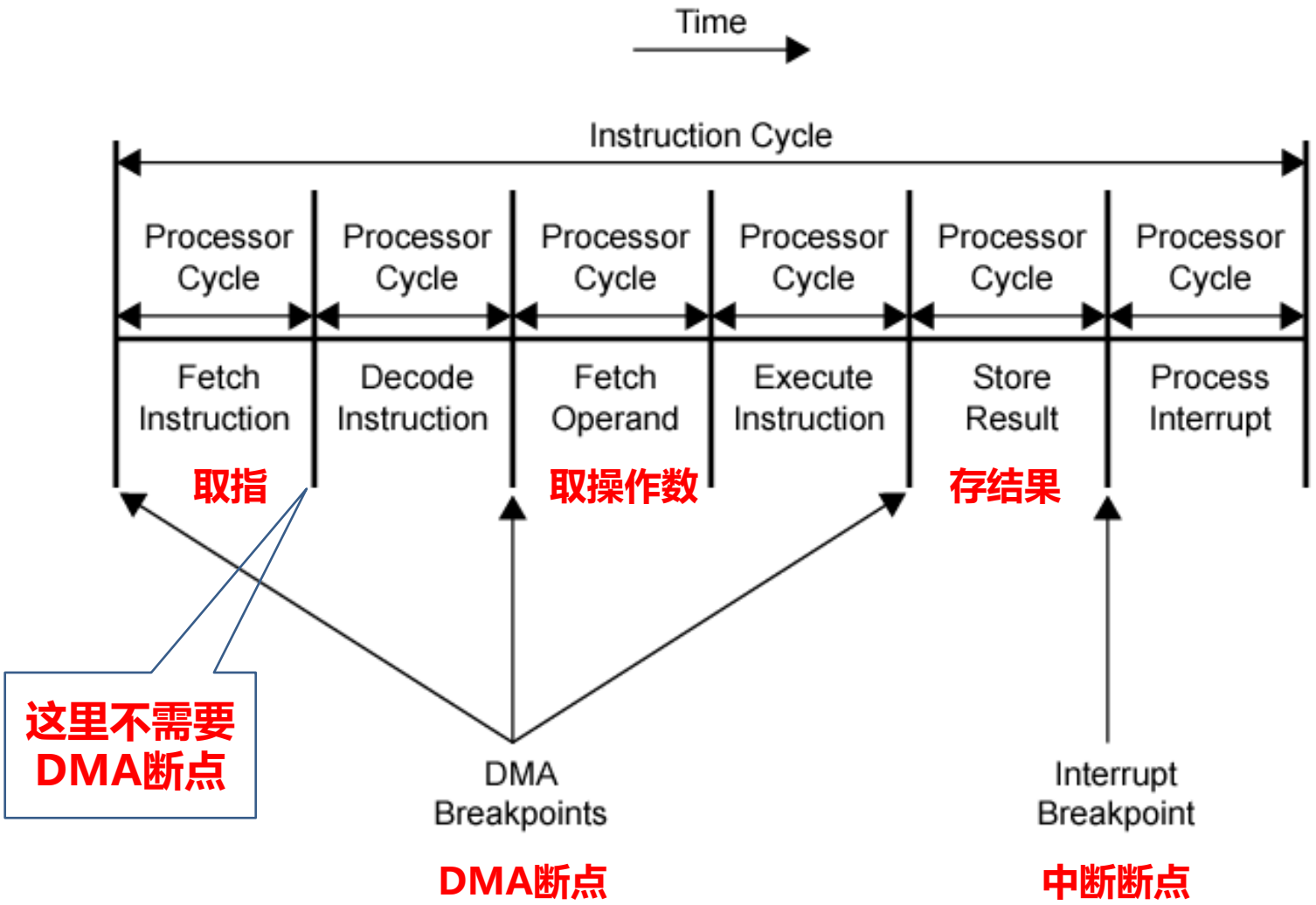


- DMA模块收到总线响应信号HLDA，Hold Acknowledgment后，发送内存地址，然后开始进行传输。
- 如果数据传输没有结束，那么就继续传输数据，并同时更改地址指针，使指针指向下一个需要传输数据的内存地址。
- 如果完成了传输，就结束本次操作。

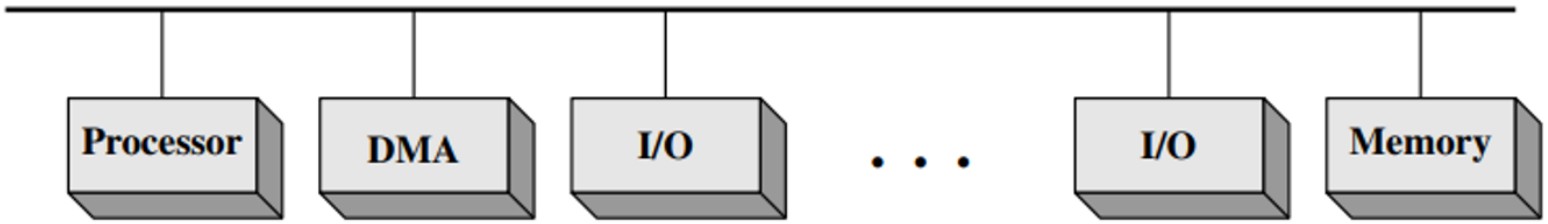
DMA transfer cycle-stealing 周期窃取

- Cycle-stealing：DMA controller takes over bus for a cycle 周期窃取：DMA控制器接管总线一个周期
  - Transfer of one word of data in this cycle 周期内传输一个字
  - Cycle-stealing is not an interrupt 不是中断
  - CPU does not switch context CPU不需要交换上下文
- CPU suspended just before it accesses bus CPU在访问总线前挂起
  - i.e. before an operand or data fetch or a data write 在取操作数或存操作数之前
- Slows down CPU but not as much as CPU doing transfer 减慢CPU，但比CPU传输要好很多

DMA breakpoint DMA断点



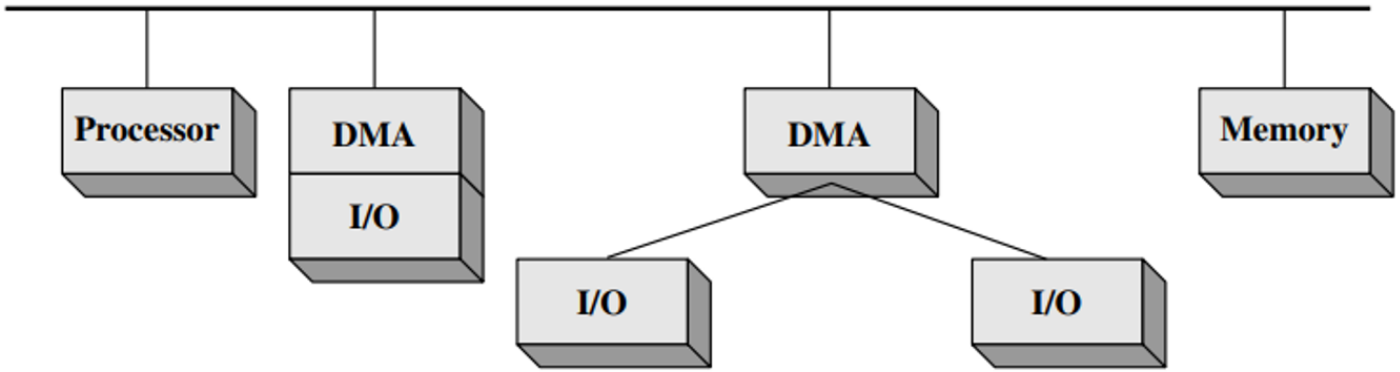
Alternative DMA configurations DMA配置1



(a) Single-bus, detached DMA

- 单总线分离的DMA，DMA、I/O模块和内存都挂在总线上
- DMA是CPU的代理，采用类似于编程式I/O的方式，在内存和I/O之间传送数据。
- 每次传送需要消耗2个总线时钟周期。一个周期用于和I/O交换数据，一个周期用于和存储器交换数据
- 配置简单，效率比较低

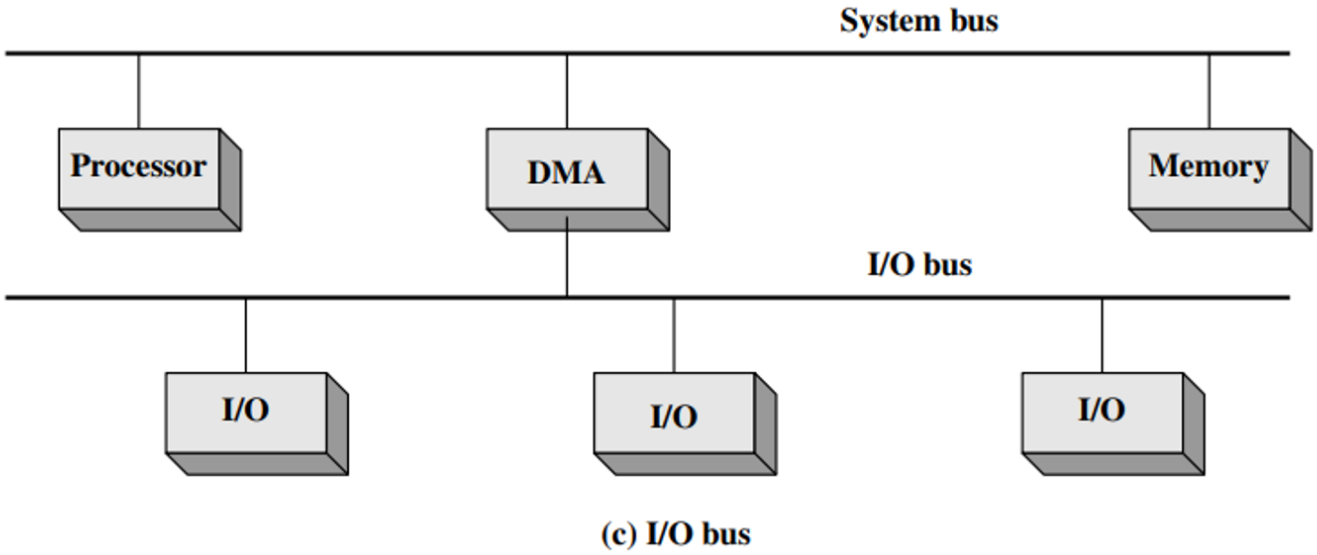
Alternative DMA configurations DMA配置2



(b) Single-bus, integrated DMA-I/O

- 单总线集成式DMA-I/O结构，I/O模块通过DMA和总线互连
- DMA可能是I/O的一部分，也可能一个DMA控制一个或几个I/O模块
- DMA与I/O之间的数据交换不需要系统总线，只有DMA和存储器交换数据到时候才会用到系统总线，减少了系统总线的开销

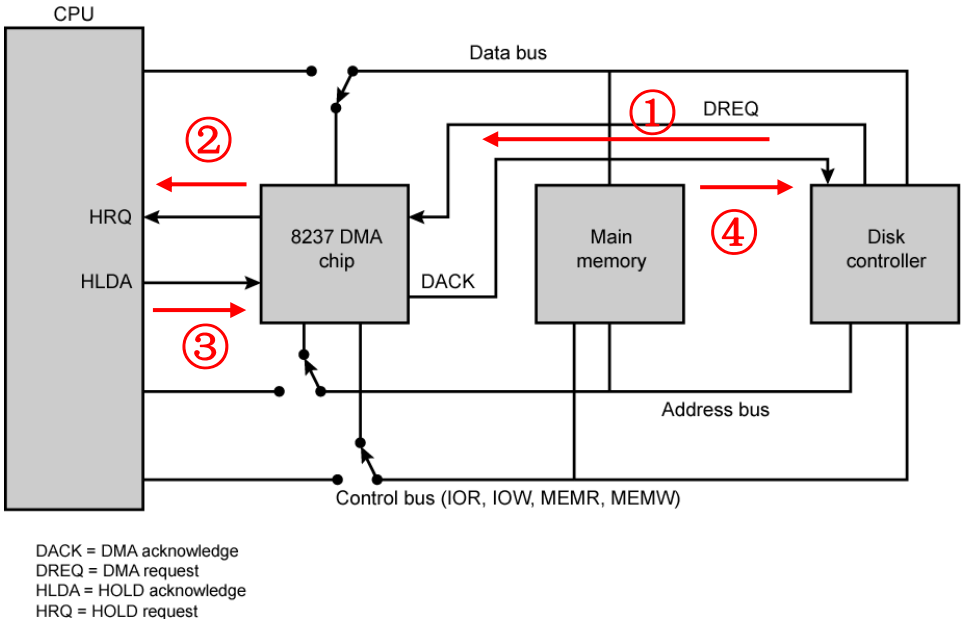
Alternative DMA configurations DMA配置3



(c) I/O bus

- I/O总线方式，从第二种方式扩充而来
- 在DMA和I/O设备之间配置一条专门的I/O总线
- 可以减少DMA模块的接口数量
- DMA与存储器交换数据的时候才会用到系统总线

8237 DMA controller



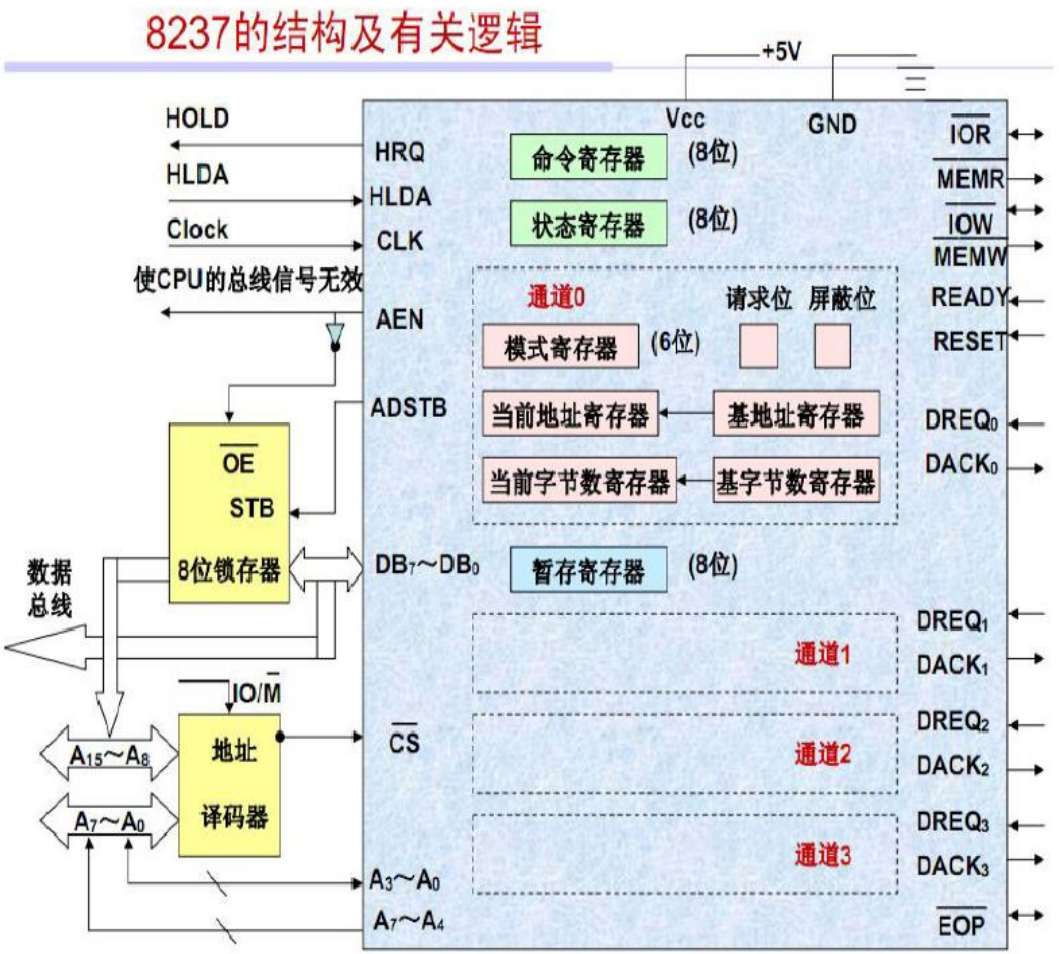
- x86处理器和DRAM之间的接口
- 当DMA模块需要总线的时候，发出HRQ信号给处理器
- CPU发出HLDA信号请求
- DMA模块收到信号后，可以使用总线

- 数据从内存到磁盘
  - 设备请求DMA服务，设置DREQ为高电平，DMA设置HRQ为高电平
  - CPU完成当期总线周期，设置HDLA，允许DMA使用总线
  - DMA激活DACK，告诉设备开始传输数据
  - DMA开始传输数据，直到完成。DMA完成后，恢复HRQ

Fly-by mode飞越模式

- DMA and CPU use the bus alternately DMA和CPU交替使用总线
- Data does not pass through and is not stored in DMA chip 数据既不通过也不保存在DMA芯片
  - 8237 DMA is known as fly-by DMA controller 称为飞越式DMA控制器
  - DMA only between I/O port and memory DMA只能完成I/O端口和内存之间的数据传输
  - Not between two I/O ports or two memory locations 不能完成I/O端口之间或内存单元之间的数据传输
- Can perform memory to memory transfer via register 通过寄存器可以完成内存间的传输

8237



- 可编程DMA控制器
- 4个独立的DMA通道
- 存储器到存储器传送：通道0为源，通道1为目的
- 每传送一个字节需8个时钟周期，前4个时钟周期用于从源区将数据读入8237A的临时寄存器，后4个时钟周期把临时寄存器中的数据写入目的区
- 通过寄存器，可以执行内存-内存的数据传输

Summary 小结

	Programmed I/O	Interrupt driven I/O	DMA
Operation	CPU checks I/O status repeatedly	I/O device sends request when ready	Data transfer without CPU
Advantage	Simple Polling sequence can be changed by program	Save time	More faster
Disadvantage	Time consuming	Complex	More complex

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

Limitations of DMA DMA的局限性

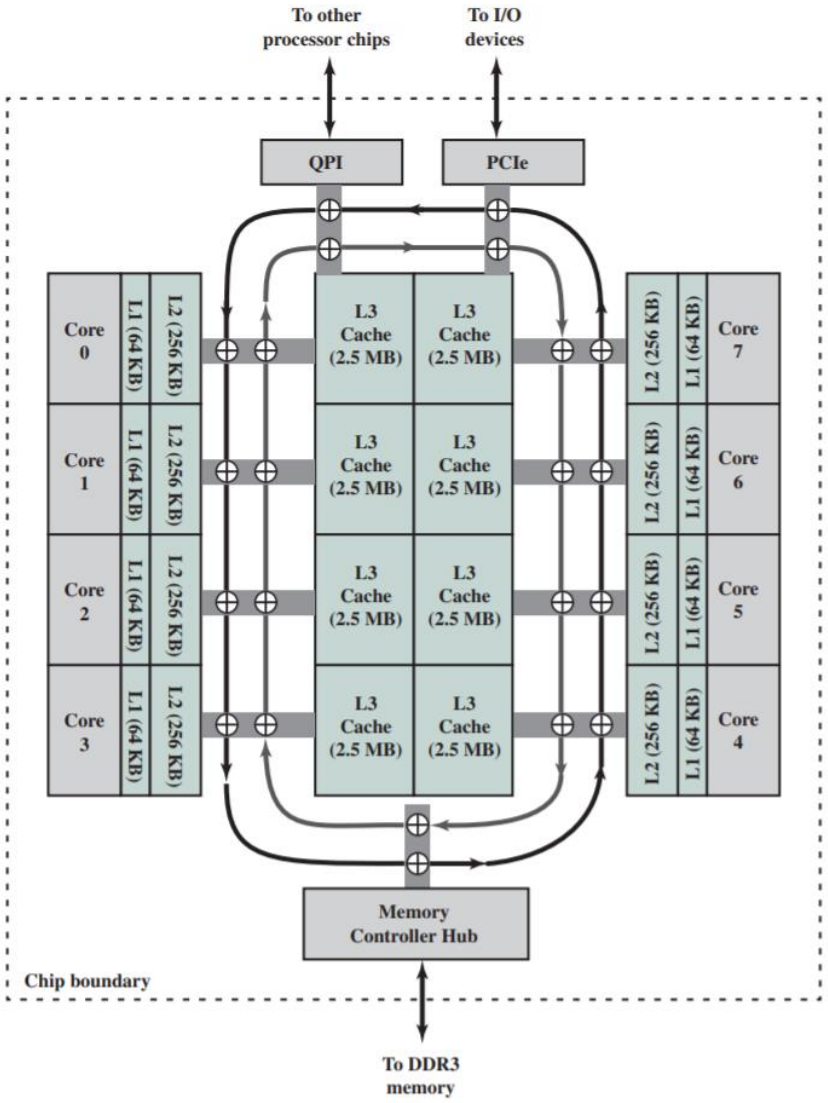
- DMA can effectively improve the access speed of I/O devices

  DMA能够有效提高I/O设备的访问速度

- With the rapid improvement of I/O data rates, such as high-speed Ethernet and DMA methods, they cannot meet the growing needs 随着I/O数据速率的快速提高，例如高速以太网，DMA方式无法满足不断增长的需要

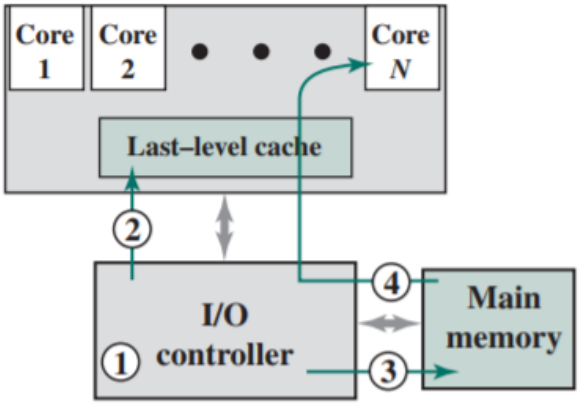
- Directly accessing Cache through I/O devices to improve performance is called Direct Cache Access, that is DCA 通过I/O设备直接访问Cache来提高性能，称为Direct Cache Access，也就是DCA

Architecture of Multi-Core 多核结构

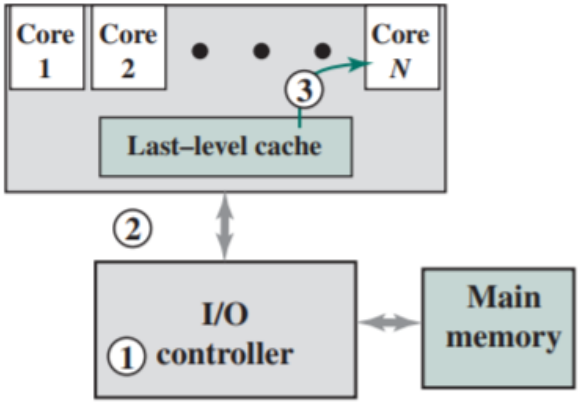


- 三级缓存的参与后，网络I/O涉及到对高速缓存和主存的多次访问
- 数据还需要在系统缓冲区和应用程序缓冲区之间的移动
- 内存的大量参与成为性能的瓶颈

Direct Cache Access-Packet Input 直接缓存访问-包输入



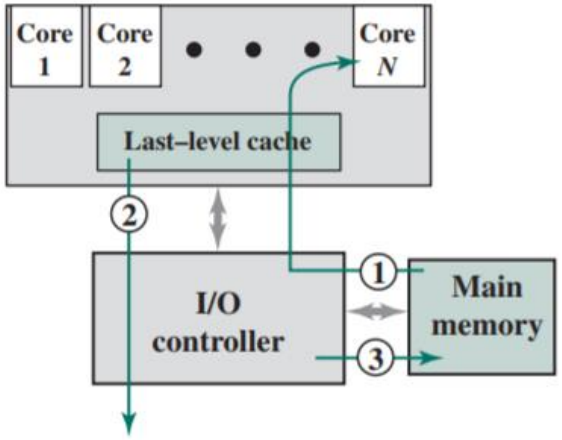
(a) Normal DMA transfer to memory



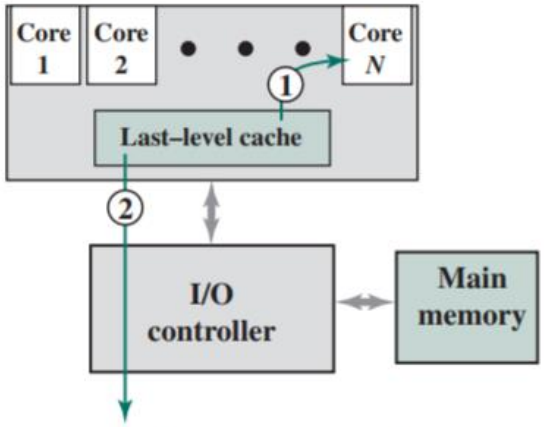
(b) DDIO transfer to cache

- 如果高速缓存命中，则更新高速缓存，不更新主存
- 如果高速缓存不命中，则对高速缓存进行写操作，并不会写回内存（对网络协议应用有效，数据处理完后直接发出去，不保留在内存中）

Direct Cache Access-Packet Output 直接缓存访问-包输出



(c) Normal DMA transfer to I/O



(d) DDIO transfer to I/O

- 协议处理程序创建要传输的包
- 包存储在高速缓存中，并不存储到内存中
- NIC发起的读操作由高速缓存来提供，不会访问内存

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

Evolution of I/O function -1 I/O功能的演进1

Peripherals are more and more complex, and I/O functions are also developing 外设越来越复杂，I/O功能也不断发展

1. CPU directly controls a peripheral device CPU直接控制外设
2. Programmed I/O： a controller or I/O module is added. The CPU uses programmed I/O without interrupts 编程式I/O：增加了控制器或I/O模块。CPU使用编程式I/O，没有中断
3. Interrupt I/O： same configuration as in step 2 is used, but now interrupts are employed 中断式I/O和上面一样，但是使用了中断

Evolution of I/O function -2 I/O功能的演进2

4. DMA: I/O module is given direct access to memory via DMA

I/O模块通过DMA可以直接访问存储器

5. I/O channel
   - I/O module is enhanced to become a processor（no memory）
     I/O模块增强成为一个处理器（没有存储器）
   - with a specialized instruction set tailored for I/O 具有定制的I/O指令集
   - The CPU instructs the I/O processor to execute I/O programs in memory CPU指示I/O处理器在内存中执行I/O程序
   - Interrupt will not occur until I/O program execution is completed
     I/O程序执行完成之后才会产生中断

Evolution of I/O function -2 I/O功能的演进3

6. I/O processor
   - has a local memory of its own and is, in fact, a computer in its own right I/O模块有自己的存储器，本身就是一个处理器
   - Control a large number of I/O devices with minimal CPU participation 控制大量的I/O设备，CPU参与最少
   - Controls communication with interactive terminals. The I/O processor handles most of the tasks 控制与交互式终端的通信。I/O处理器负责处理大部分任务

Generally, I/O channel and I/O processor are not distinguished. Called as I/O channel I/O通道和I/O处理器不做区分，都称为I/O通道

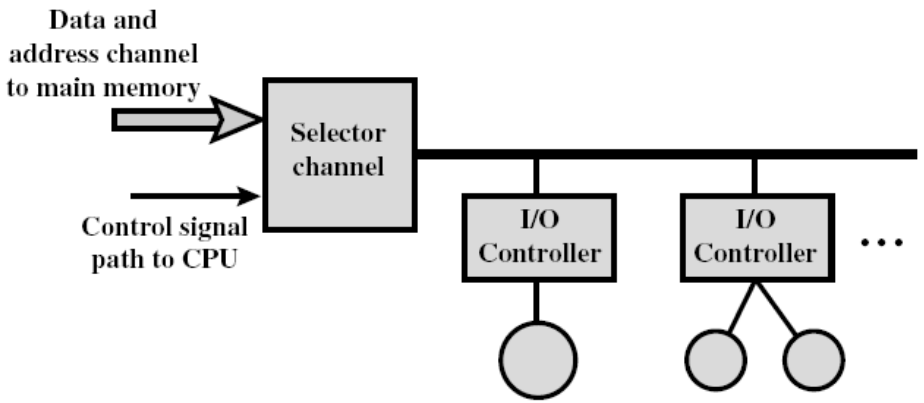
I/O channel

- I/O devices getting more sophisticated I/O设备越来越复杂
  - e.g. 3D graphics cards 比如3D图形卡
- CPU instructs I/O controller to do transfer CPU指示I/O控制器去进行数据传输
- I/O controller does entire transfer I/O控制器完成全部传输
- Improves speed 提高了速度
  - Takes load off CPU 减轻了CPU的负荷
  - Dedicated processor is faster 专用处理器会更快

I/O channels operation I/O通道的操作

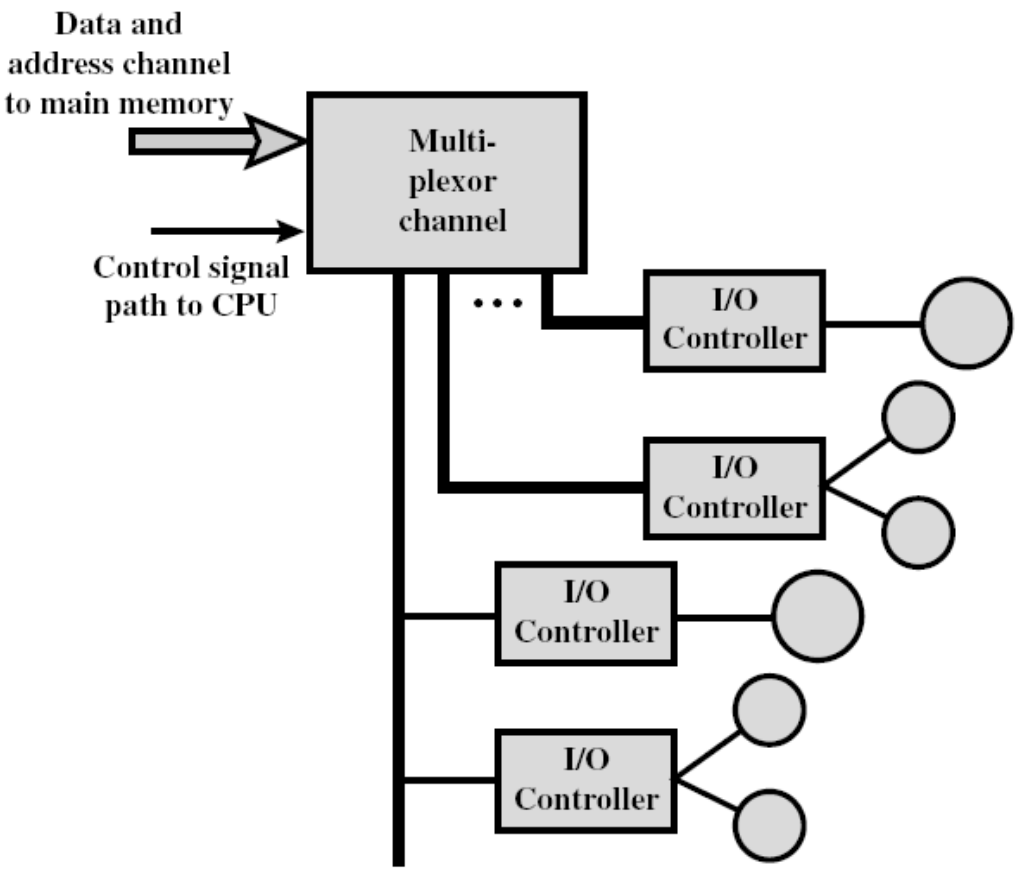
- CPU initiates an I/O instruction that instructing the I/O channel to execute a program in memory CPU初始化I/O指令，指示I/O通道去执行内存中的程序
- The program will specify the device, the area of memory for storage, priority, error handling 程序指定设备，内存存储的区域，优先级和错误处理方法
- The I/O channel follows these instructions and controls the data transfer I/O通道按照指令来控制数据传输
- I/O channel has two types 两种类型
  - Selector channel 选择通道
  - Multiplexor 多路复用

Selector channel 选择通道



- 选择通道控制多个高速设备
- 每次只与一个设备进行数据传输
- 每个设备或一组设备由一个I/O模块控制
- I/O通道代替CPU对I/O模块进行控制

Multiplexor 多路复用



- 能够同时处理多个设备的数据传输
- 多路复用器尽可能快地为多个设备传输数据
- 数据顺序可能是交叉的

三个设备的数据流

$A_1A_2A_3A_4$,

$B_1B_2B_3B_4$,

$C_1C_2C_3C_4$

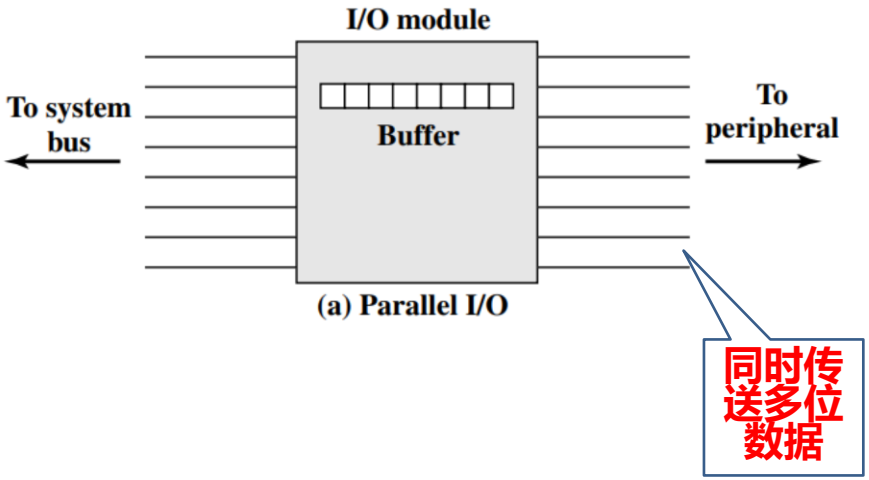
可能是如下顺序

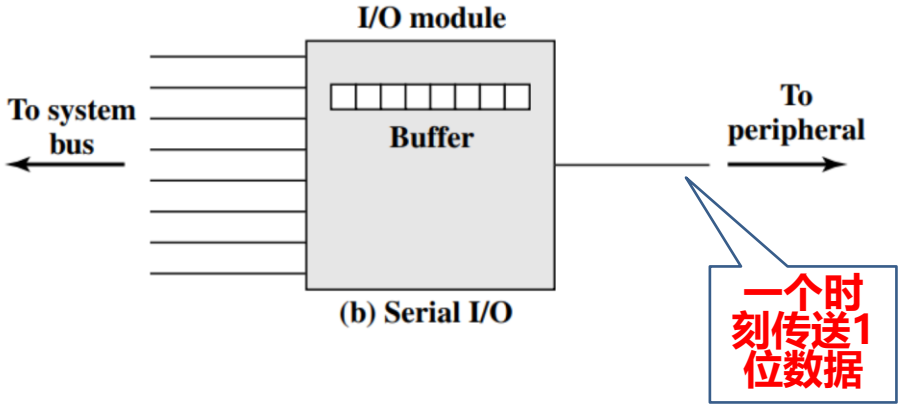
$A_1B_1C_1A_2C_2A_3B_2C_3A_4$

Outline

- External Devices 外部设备
- I/O Modules I/O模块
- Programmed I/O 编程式I/O
- Interrupt-Driven I/O 中断式I/O
- Direct Memory Access 直接存储器访问
- Direct Cache Access 直接缓存访问
- I/O Channels and Processors I/O通道和处理器
- The External Interconnection 外部互联

Types of interface





- 并行接口有多个数据线和外设相连接，同时可以发送多位数据给外设，一般用于高速外设
- 串行接口只有一个数据线
- 每次只能发送一位数据。主要用于低速外设，比如打印机、终端
- 随着技术的发展，高速串行接口出来之后，并行接口用的越来越少

Interaction between I/O module & peripherals I/O模块和外设的交互

- I / O module sends a control signal and requests to send data

  I/O模块发送控制信号，请求发送数据

- Peripheral responds to the request 外设响应请求
- I / O module sends data I/O模块发送数据
- Peripherals acknowledge receipt of data 外设确认收到数据
- Connection type 连接方式
  - one to one
  - one to many：I/O bus 一对多，I/O总线
- Two typical I/O buses：1394，InfiniBand

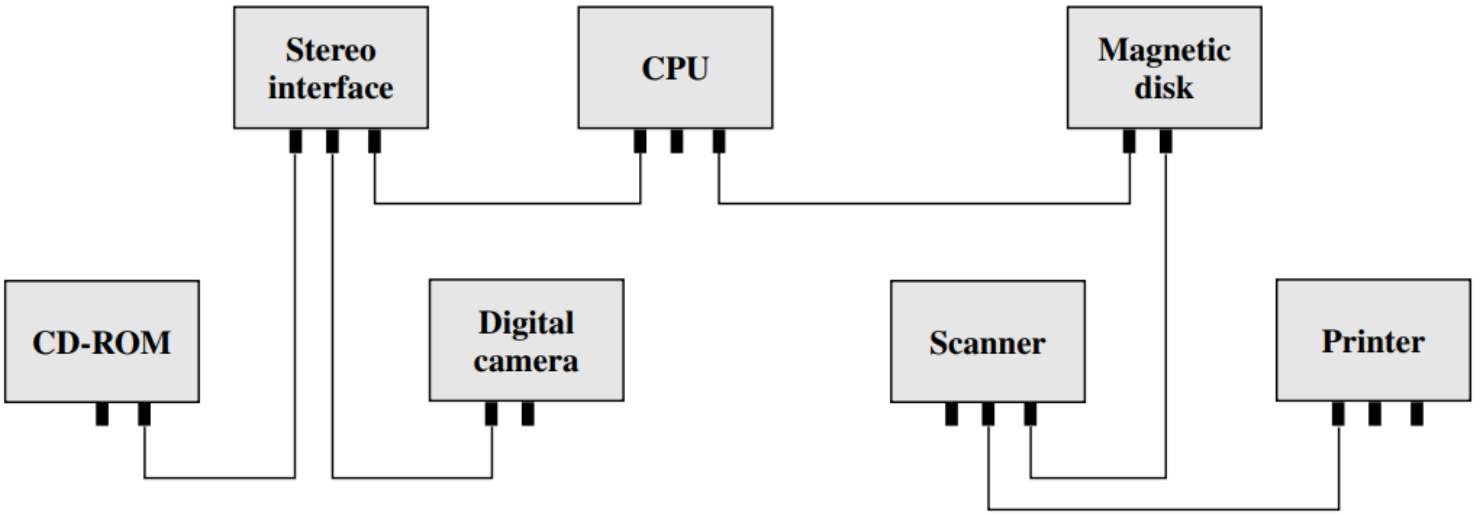
IEEE 1394 FireWire IEEE 1394 火线

- In 1995, it was designated as IEEE1394-1995 technical specification by IEEE
  IEEE 1995年定为1394标准
  - Proposed by Apple in 1986，called FireWire 苹果公司1986年提出，称为Firewire
  - Sony also proposed a similar technology, called i.Link SONY公司也提出类似的技术，称为i.link
  - Texas Instruments （TI）called it Lynx TI称为Lynx
- High performance serial bus 高性能串行总线
- Fast 快
- Low cost 便宜
- Easy to implement 容易实现
- Also being used in digital cameras, VCD and TV 也用于数码相机，视频播放机和电视

FireWire configuration

- Daisy chain 菊花链
- Up to 63 devices on single port 单个端口最多63个设备
  - Really 64 of which one is the interface itself 接口本身占了1个，总共64个
- Up to 1022 buses can be connected with bridges 通过桥接的方式最多能有1022个总线
- Hot plug，automatic configuration 热插拔，自动配置
- No bus terminators 没有总线终结器
- May be tree structure 可以配置为树形结构

Simple FireWire configuration

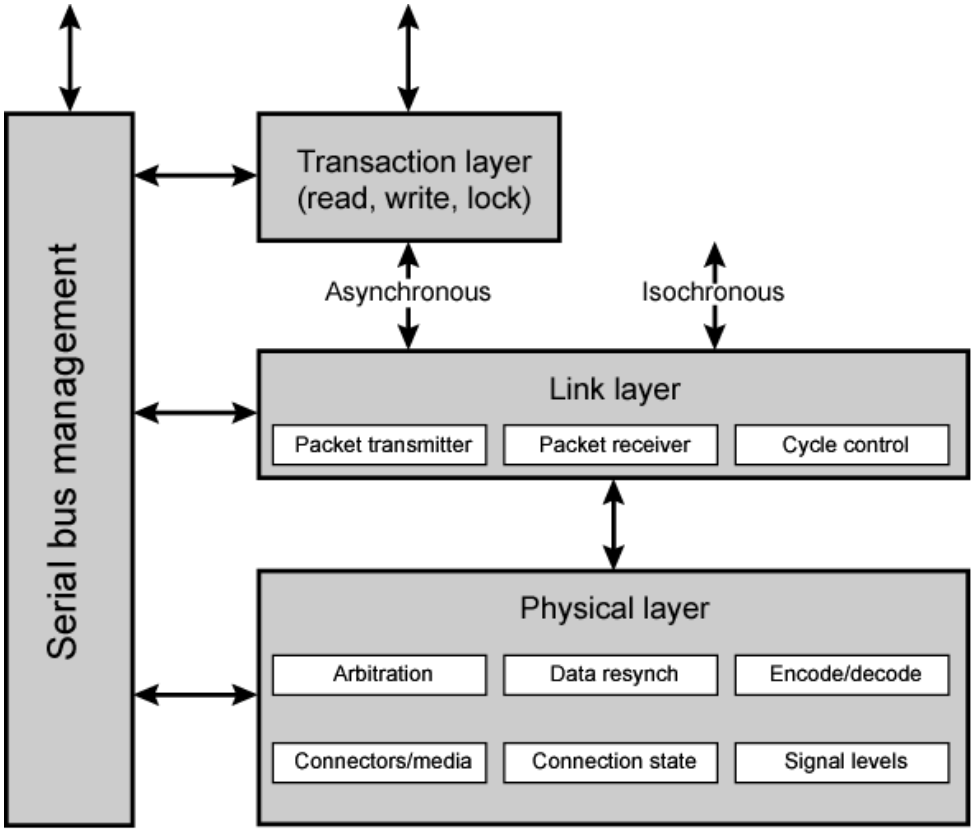


- 左边的CD-ROM和数码相机并接到立体声接口
- 而右边的磁盘、扫描仪和打印机都是串接的
- 1394接口非常灵活

FireWire 3 layer stack FireWire 3层协议栈

- Physical 物理层
  - Transmission medium, electrical and signaling characteristics 传输媒介，电气和信号特征
- Link 链路层
  - Transmission of data in packets 数据包的传输
- Transaction 业务层
  - Request-response protocol 请求-响应协议

FireWire protocol stack FireWire协议栈



- 最底层是物理层，指定可选的传输媒介，以及连接器。物理层将二进制数据转化成各种物理媒体的电信号，并提供仲裁保证每次只有1个设备发送数据
- 第二层是链路层，负责数据包的发送和接收
- 第三层是业务层，负责数据的读写等功能

FireWire – physical layer 物理层

- Data rates from 25 to 3200Mbps 数据速率从25到3200Mbps
- The physical layer needs to provide an arbitration mechanism to ensure that only one device sends data each time 物理层需要提供仲裁机制，确保每次只有一个设备进行数据的发送
- Priority based arbitration 基于优先级的仲裁
  - Based on tree structure 基于树结构
  - Root acts as arbiter 根节点作为仲裁器
  - node closer to the root has higher priority 离根越近的节点优先级越高
  - First come first served 先来先服务
  - When requesting at the same time, the higher priority should be served first 同时请求时，优先级高的先服务

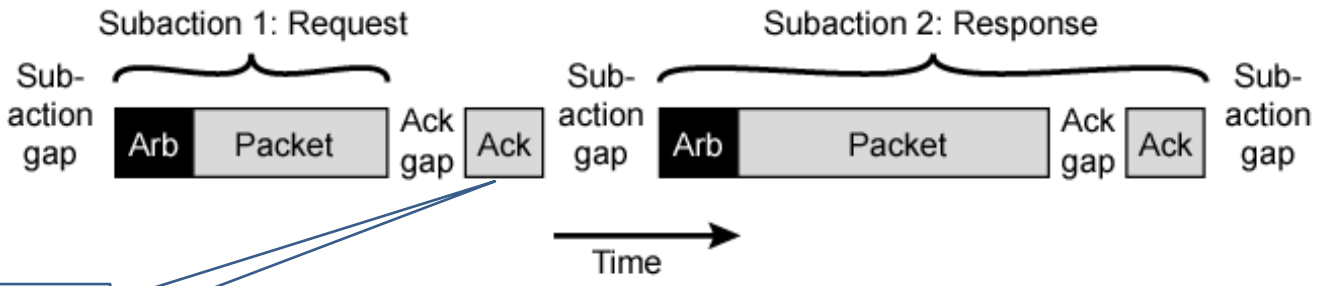
FireWire – physical layer 物理层

- Fair arbitration 公平仲裁
  - Bus time is divided into equal time intervals 总线时间划分为相等的间隔
  - During the time interval, each node can request the bus 在时间间隔内，每个节点都可以请求总线
  - After obtaining bus access, it is not allowed to request the bus again 获得总线访问后，不允许再次请求总线
  - Avoid high priority device exclusive bus 避免高优先级的设备独占总线
- Urgent arbitration 紧急仲裁
  - Specific equipment has emergency priority 特定设备具有紧急优先级
  - Bus control can be obtained multiple times in the interval 可以在间隔期内多次获得总线控制权

FireWire – link layer 链路层

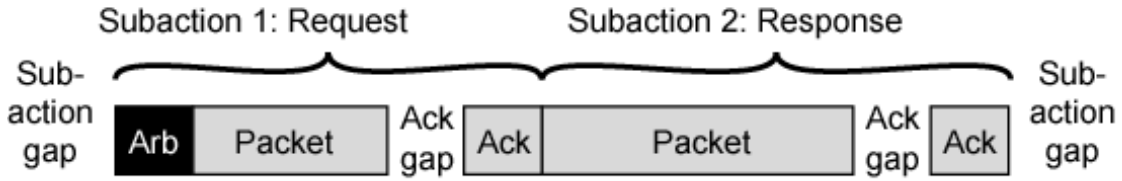
- Two transmission types 两种传输类型
  - Asynchronous 异步
    - Variable amount of data transferred as a packet 包里的数据不等长
    - To explicit address 到显式地址
    - Acknowledgement returned 需要返回确认信号
  - Isochronous 同步
    - Variable amount of data in sequence of fixed size packets at regular intervals 规定间隔发送固定大小的包
    - Simplified addressing 简单寻址
    - No acknowledgement 不需要确认信号

FireWire sub-actions FireWire 子操作

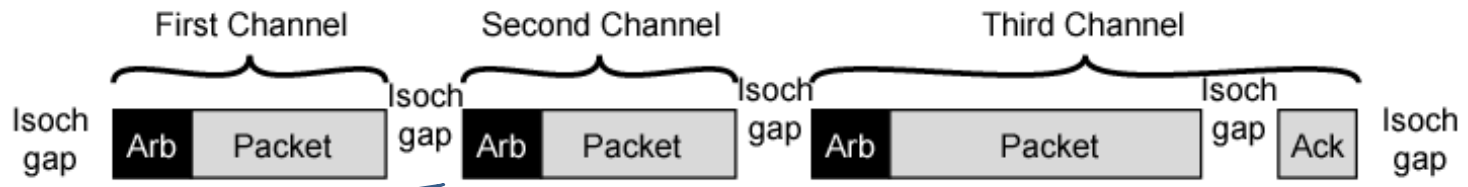


等待ACK

(a) Example asynchronous subaction 异步



(b) Concatenated asynchronous subactions 连续异步



不需要ACK

(c) Example isochronous subactions 同步

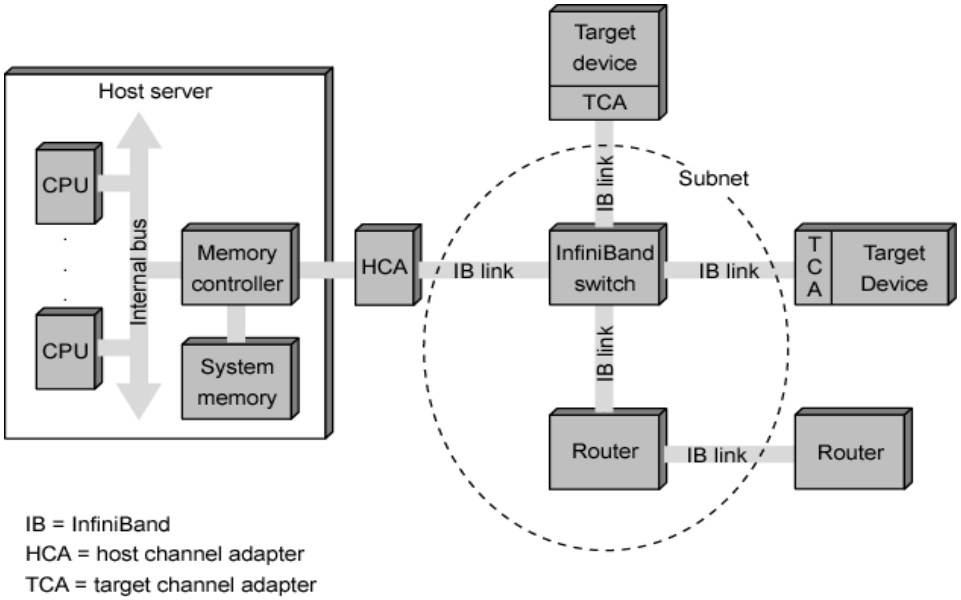
InfiniBand

- I/O specification aimed at high end servers 针对高端服务器的I/O规范
  - Merger of Future I/O (Cisco, HP, Compaq, IBM) and Next Generation I/O (Intel) 由思科、惠普、康柏、IBM等提出的未来I/O和Intel提出的下一代I/O融合而形成
  - Version 1 released early 2001 2001年发布V1版本
- Architecture and spec. for data flow between processor and intelligent I/O devices 处理器和智能I/O之间的数据流的架构和规范
- Intended to replace PCI in servers 代替PCI
- Increased capacity, expandability, flexibility 提升性能，扩展性和灵活性

InfiniBand architecture InfiniBand体系结构

- Attach servers, remote storage, network devices to central fabric of switches and links 连接服务器，远程存储和网络设备到由交换机和链路组成的中央网带
  - Greater server density 服务器密度大
  - Scalable data centre 规模可扩展
  - Independent nodes added as required 根据需要增加独立节点
- Different transmission media can reach different transmission distances 不同的传输媒介，可以达到不同的传输距离
  - 17m using copper 铜线17m
  - 300m multimode fibre optic 多模光纤300m
  - 10km single mode fibre 单模光纤10km
- Up to 30Gbps 最高速度30Gbps

InfiniBand Switch Fabric 交换机网带



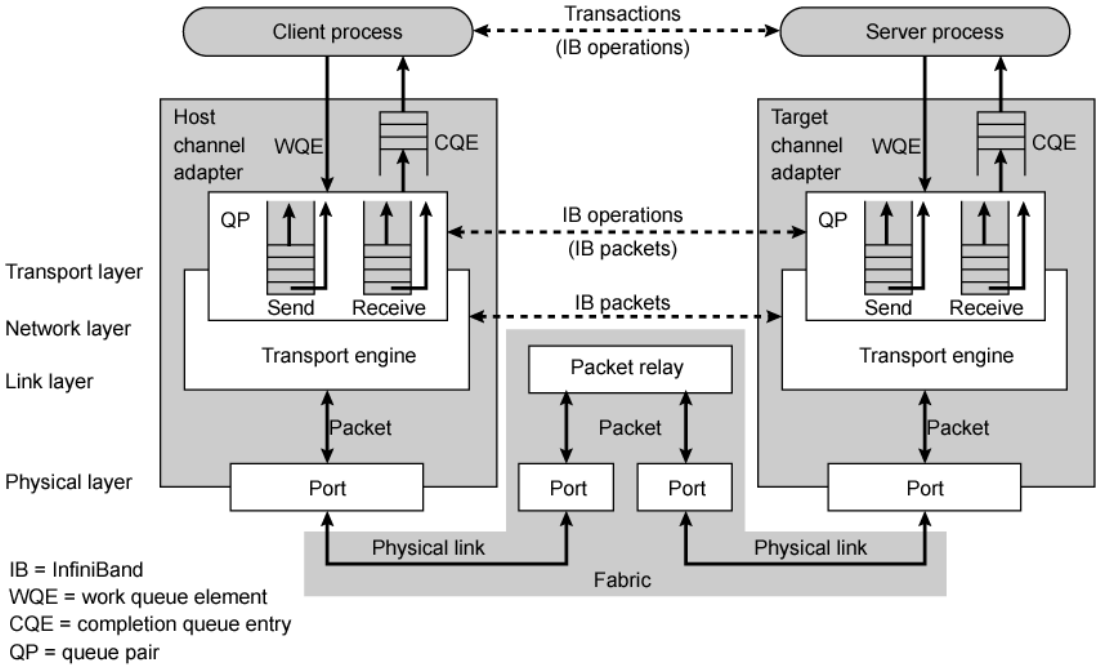
- 主机通道适配器，HCA，将主机和交换器相连。HCA通过存储控制器连接到系统总线，采用DMA的方式。
- 目标通道适配器，TCA，将远程存储系统、路由器和其他外围设备连接到交换机
- 链路：各个部件之间的数据通道
- 子网：一个或多个交换器、以及和它连接的设备组成，内部可以实现广播和多播服务
- 路由器：子网间或者子网和局域网、广域网之间的连接设备

InfiniBand operation InfiniBand 操作

- 16 logical channels (virtual lanes) per physical link 每个物理连接有16个逻辑通道，也称为虚拟通路
- One lane for management, rest for data 一个虚拟通路用作管理，其他的用作数据
- Data in stream of packets 数据包以流的形式传输
- Virtual lane dedicated temporarily to end to end transfer 端到端传输的过程中，虚拟通道是作为专用通道
- Switch maps traffic from incoming to outgoing lane，realize data exchange function 从输入到输出映射流量，实现数据交换功能

InfiniBand protocol stack InfiniBand协议栈

- 物理层：定义了物理媒介，以及传输速度。传输速度为1X，4X和12X，分别是2.5G，10G和30G
- 链路层：定义了数据传输的包格式，包括每个设备的唯一链路地址的寻址方式，以及用于提高可靠性的检错码
- 网络层：提供不同InfiniBand子网之间的路由
- 传输层：为端到端的数据包的传输提供可靠性机制



Limitations of DMA 其他互联方式

- USB 通用串行总线
- SCSI 小型计算机系统接口
- Thunderbolt 闪电接口
- PCIe PCI express
- SATA 串行高级技术附件
- Ethernet 以太网
- Wi-Fi 无线互联网

Key Terms

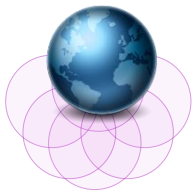
Cycle stealing	Interrupt-driven I/O	Isolated I/O	Selector channel
DMA	I/O Channel	Memory-mapped I/O	Serial I/O
FireWire	I/O command	Multiplexor channel	
InfiniBand	I/O module	Peripheral device	
interrupt	I/O processor	Programmed I/O	

Summary and Question

- 小结
  - 这节课我们对外设的工作原理做了详细的分析，对三种I/O方式做了详细的讨论，并对外部接口做了简要描述。
- 问题
  - 问题1：I/O模块在输入输出中起到什么作用？I/O模块有哪些功能？
  - 问题2：DMA中的时钟窃取是什么意思？

Assignments

- Review questions: 7.1, 7.3~7.7
- Problems: 7.1, 7.3, 7.12, 7.13



谢谢大家！