



计算机组成原理

外设，输入输出 (RV \$1.4 \$6.10, 唐第5章)

llxx@ustc.edu.cn



内容提要

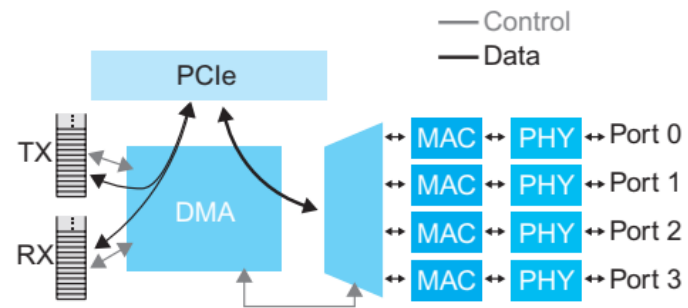
- 唐第5章、§8.4
- RV §1.4: 外设 (显示器, 触摸屏, 网络)
- RV §6.10: 总线, I/O系统
 - 示例: NetFPGA 10G Ethernet NIC
 - an FPGA-based open platform for network research and classroom experimentation

– 总线

- PCI: 并行总线
- PCIe总线: 串行总线

– I/O系统

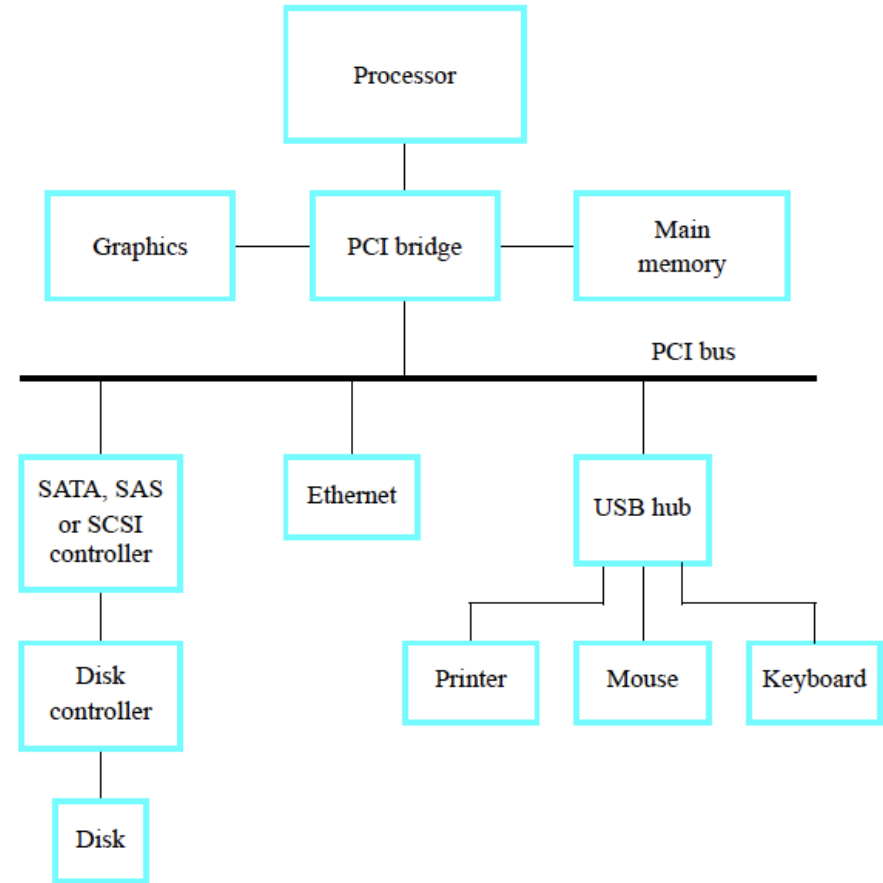
- MMIO
- I/O控制方式: Polling, Interrupt-driven I/O, DMA
- 设备驱动程序



I/O: CPU寄存器或内存与外设间数据交换



- 现代计算机组成设备
 - Von Neumann机组成
 - Peripheral device
 - 输入设备: keyboard、mouse、touchscreen、scanner、digital camera、microphone、sensor
 - 输出设备: displayer、printer、acuator
 - Connectivity: Bus, Network
 - data rate
 - keyboard: 100 bytes/sec
 - disk: 30 MB/s
 - network: 1 Mb/s - 1 Gb/s
- 如何访问I/O设备: 软硬协同
 - 接口组成 (软件、硬件、接口)
 - 过程控制 (查询, 中断, DMA)



I/O: printf(), getchar(), putchar()?



```
#include <stdio.h>
int main(void)
{ int ch;
```

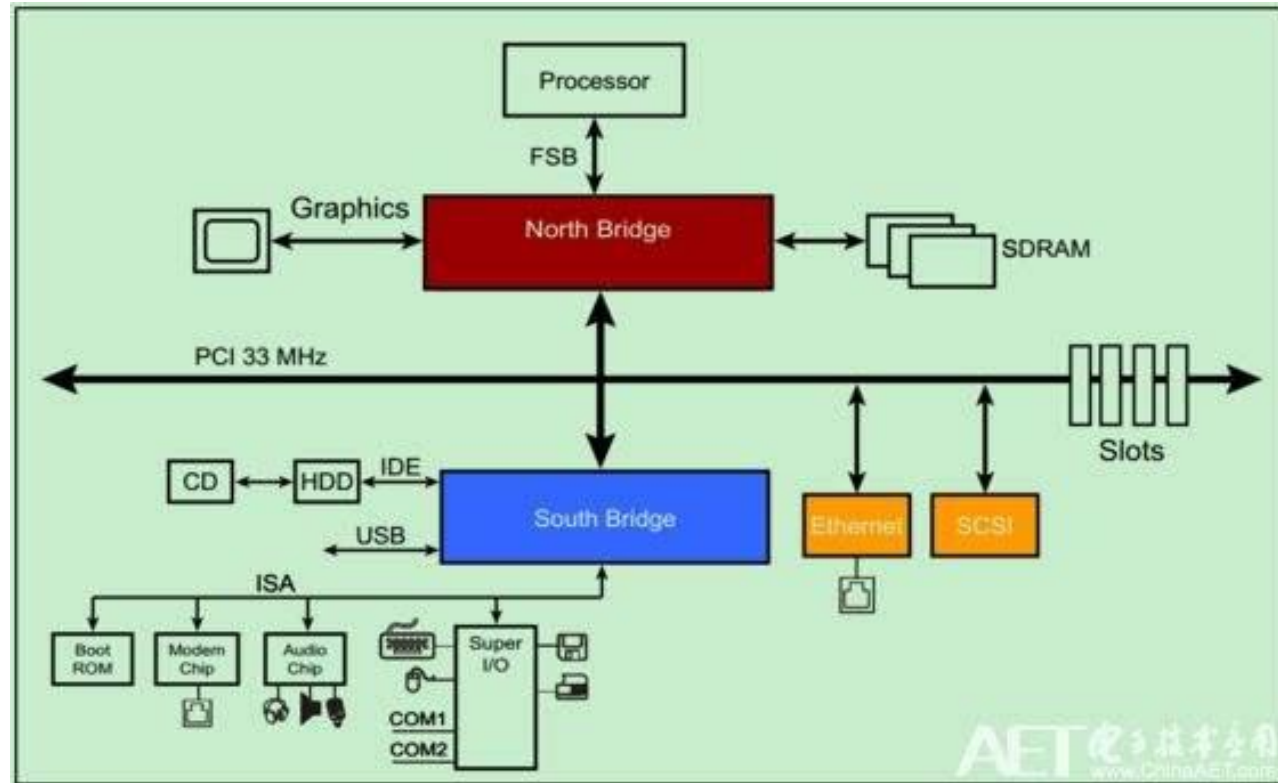
```
printf("Input a
character:");
```

```
/* read a
character from
the standard input
stream */
```

```
ch = getchar();
putchar(ch);
```

```
return 0;
```

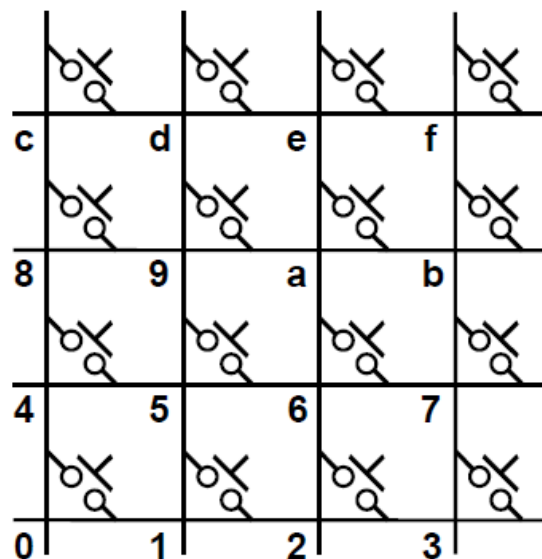
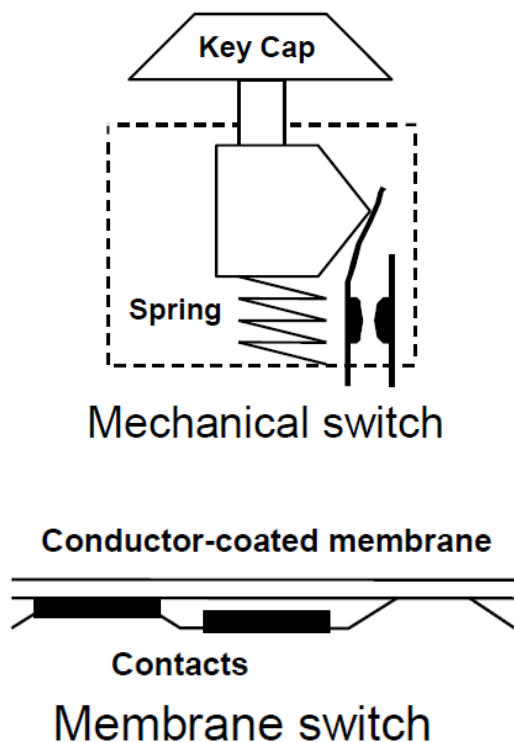
```
}
```



从哪儿输入，向哪儿输出？
何时输入，何时输出？
并发访问：单任务，多任务？



示例：KB，位置码

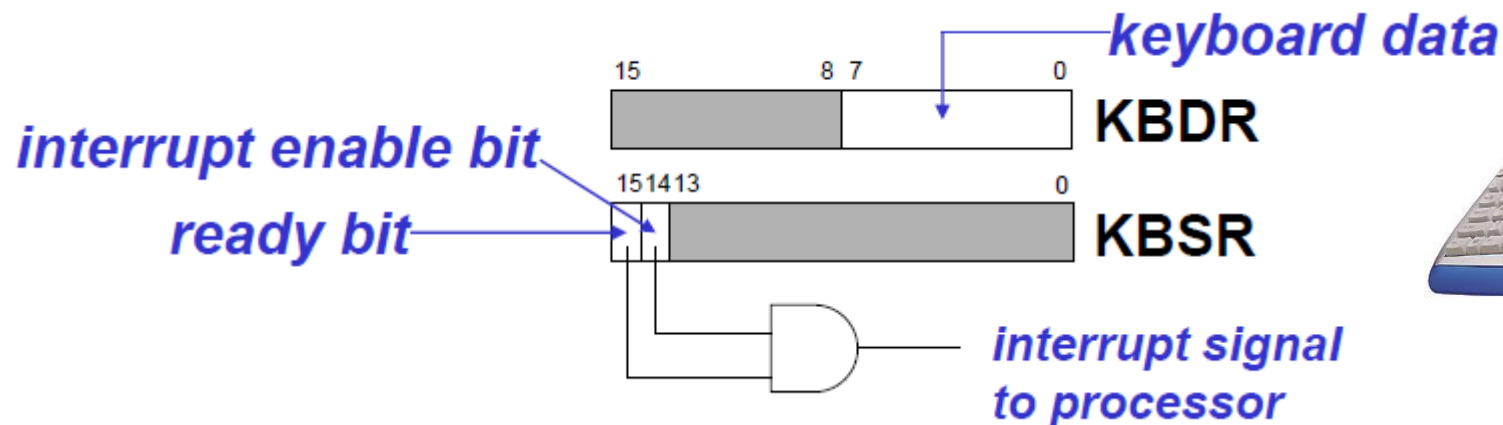
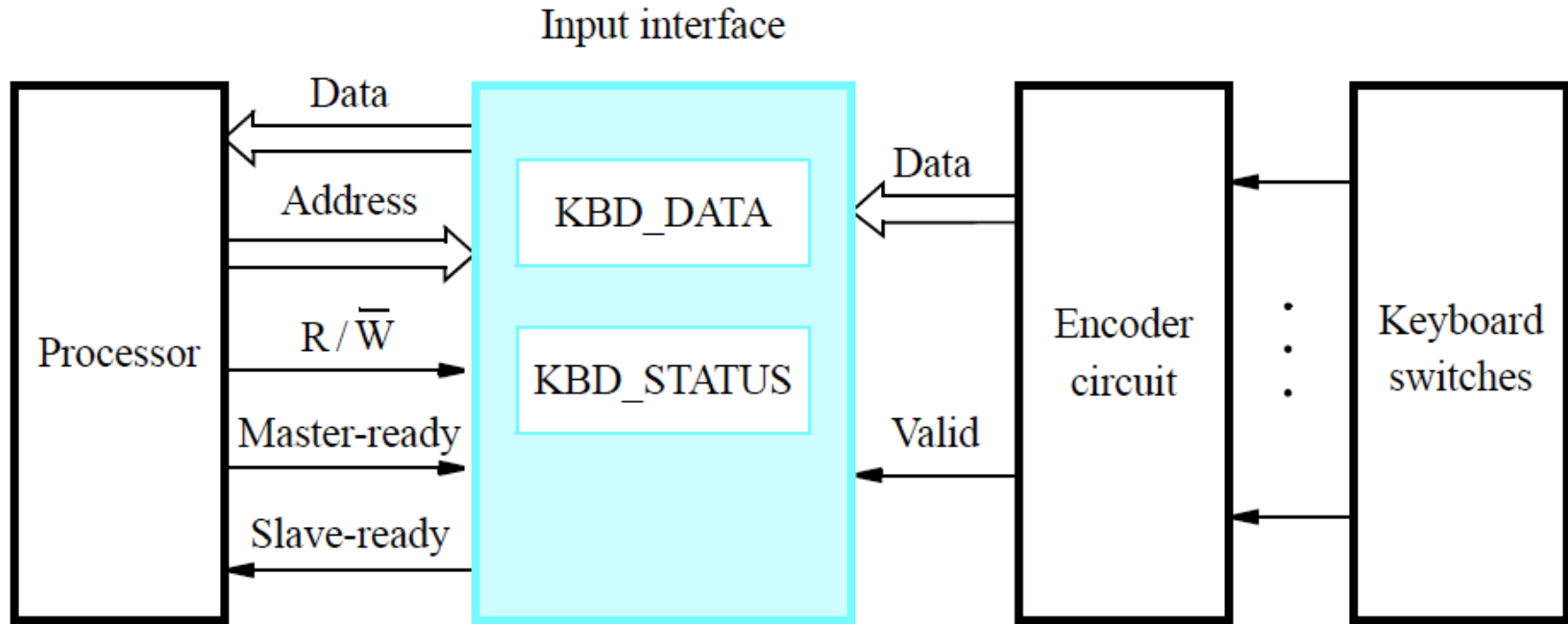


Logical arrangement of keys

编码键盘：字符码（ASCII）
非编码键盘：位置码（POS机）



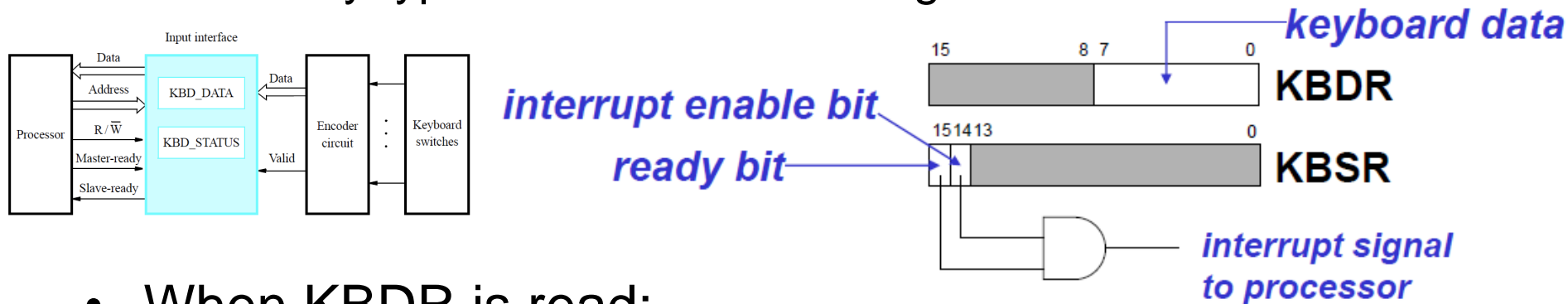
I/F: Keyboard-to-processor connection



Input from Keyboard: 步骤



- When a character is typed:
 - KB Data: its ASCII code is placed in bits [7:0] of KBDR
 - (bits [15:8] are always **zero**)
 - the “ready bit” (KBSR[15]) is set to 1
 - keyboard is **disabled**, ——不接收新的键入
 - any typed characters will be ignored



- When KBDR is read:
 - KBSR[15] is set to 0, ——空, 数据无效
 - keyboard is enabled, ——可接受新的键入
- I/O过程: 程序轮询polling或INTR driven
- 多任务共享?



I/O系统：唐\$5.2

- I/O系统组成原理

- I/O系统的构成

- 接口，端口，编址

- 数据传输过程

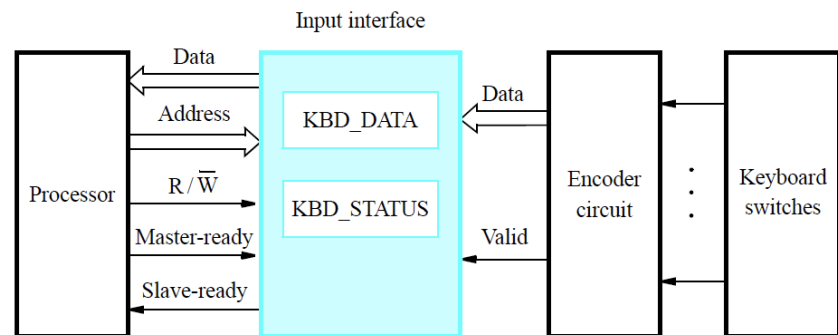
- 总线通信协议：物理层、数据链路层

- 数据传送方式：串/并，单字节/突发

- 收发同步方式

- 应用层：数据传输控制方式（编程模型）

- 程序查询、中断、DMA、通道





I/O系统组成

- **软件：**

- **软件的主要任务：**

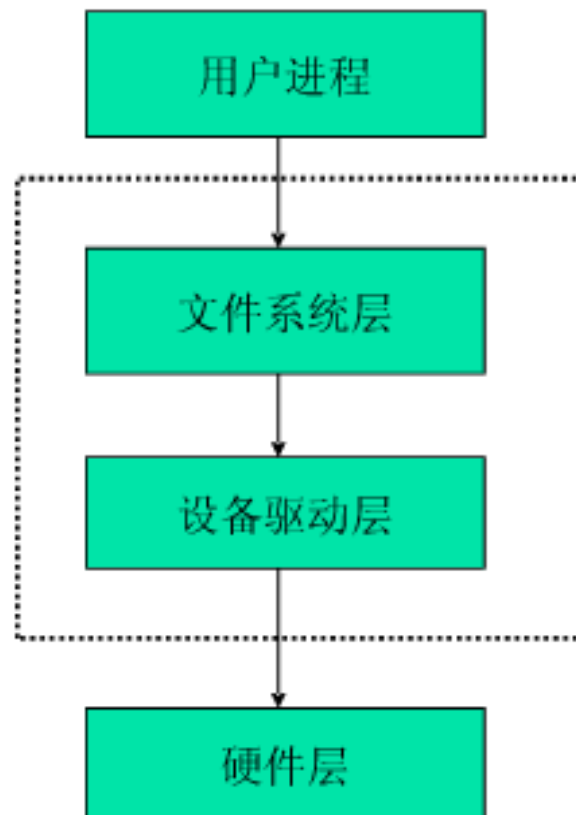
- 将数据输入至主机
 - 将运算结果输出给用户
 - 实现I/O系统和主机协同工作
 - I/O过程控制：多个总线周期

- **应用软件：APP**

- **操作系统：API**

- **设备驱动程序**

- **硬件：设备，接口**



I/O接口



- 接口：部件之间的交接部分

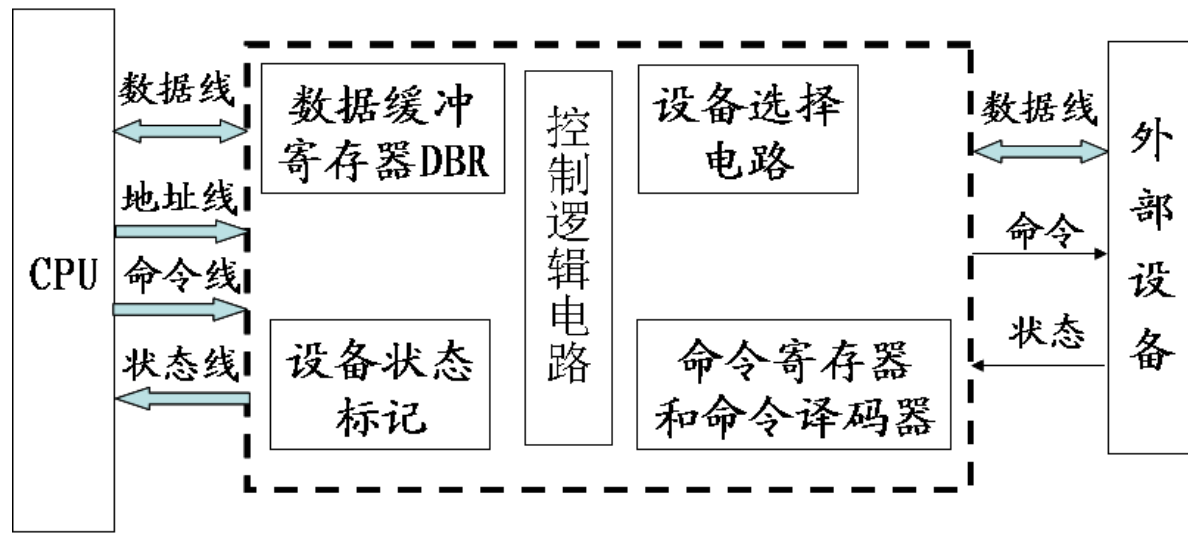
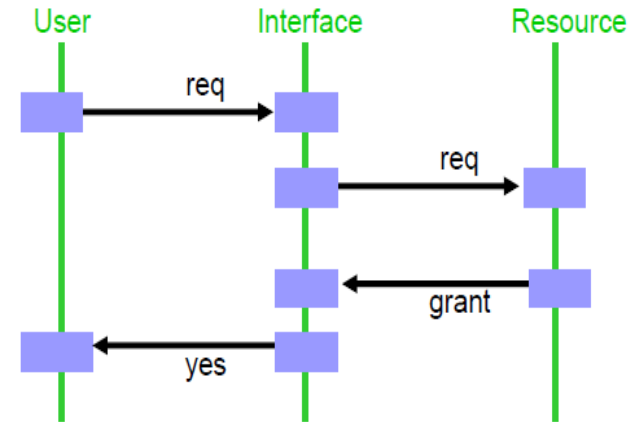
- 硬件接口：连接电路
- 软件接口：逻辑边界（数据结构）

- I/O接口

- 指主机与外设之间设置的**硬件电路**及相应的**软件控制**

- I/O接口的功能：模块化、标准化

1. CPU和外设命令转换
2. 电平转换
3. 设备选择
4. 数据缓冲
5. 设备状态
6. 错误处理





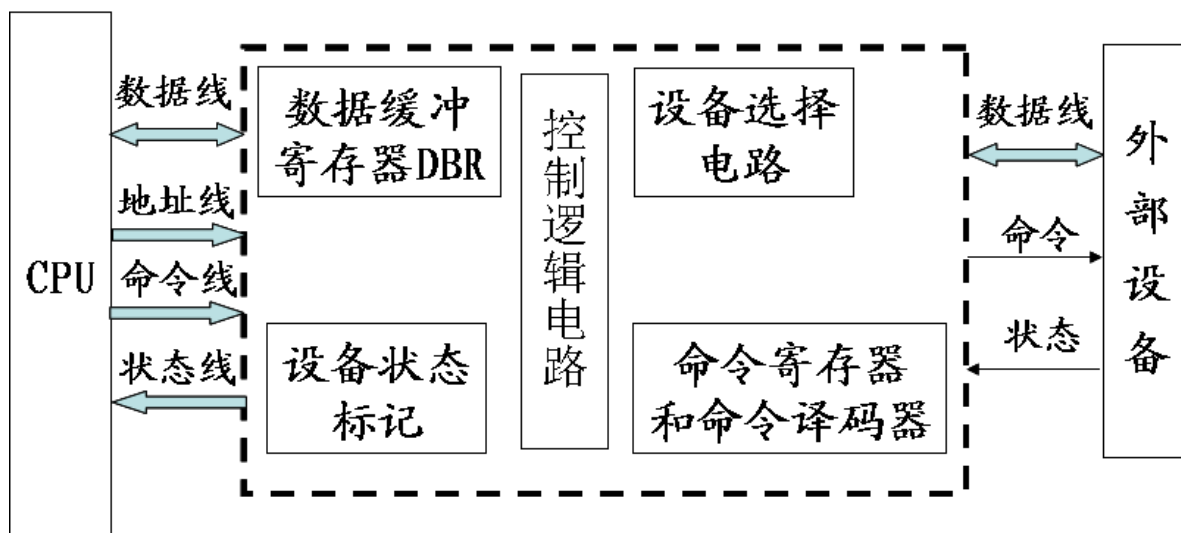
接口的类型

- 按数据传送方式分，有**并行**接口和**串行**接口。
 - 并行接口：一个字节或一个字的所有位同时传送，如Intel 8255。
 - 串行接口：一位一位传送，如Intel 8251。
- 按功能选择的灵活性分，有**可编程**接口和不可编程接口。
 - 可编程接口：可用程序来改变或选择接口的功能和操作方式(如Intel 8255、Intel 8251)。
 - 不可编程接口：不能用程序来改变其功能，但可通过硬连线路逻辑来实现不同的功能(如并行接口芯片Intel 8212)
- 按通用性分类，有**通用**接口和专用接口。
 - 通用接口：可供多种外设使用，如Intel 8255、8212。
 - 专用接口：为某类外设或某种用途专门设计的，如Intel 8279可编程键盘/显示器接口；Intel 8275可编程CRT控制器接口等。
- 按数据传送的**控制方式**分类，有程序型接口和DMA式接口
 - 程序型接口：用于连接速度较慢的设备，如键盘、打印机等，如Intel 8259。
 - DMA式接口：用于连接高速I/O设备，如磁盘，常用Intel 8237。

接口 (interface) 、端口 (port)



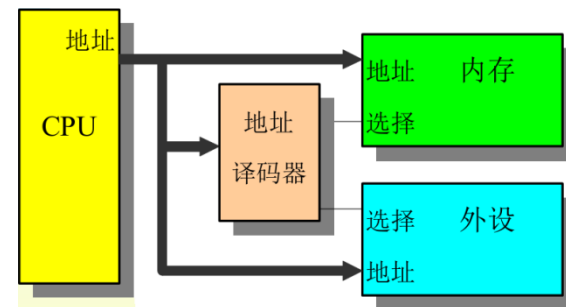
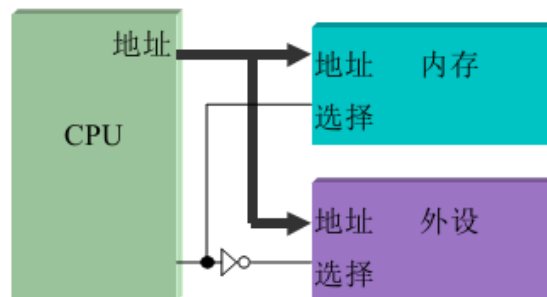
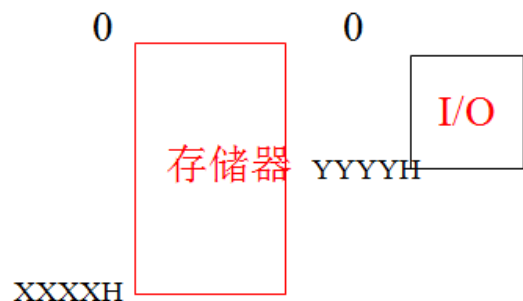
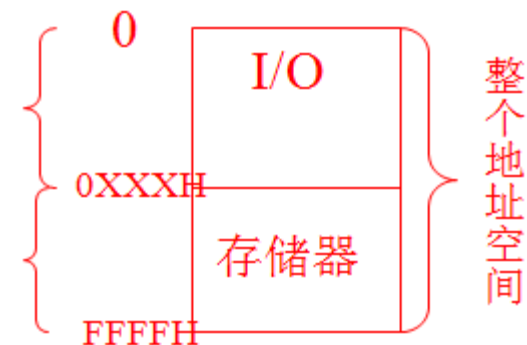
- 端口 = 接口寄存器 (数据、控制、状态...)
- 接口 = 模块 = N个端口 + 控制逻辑



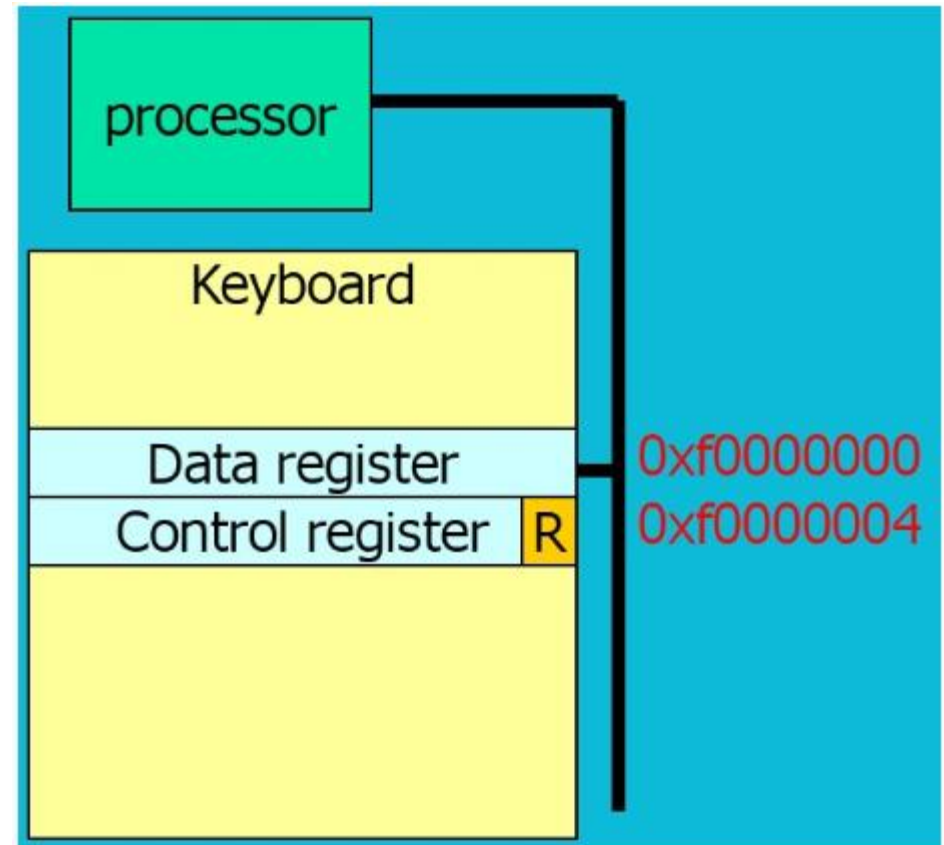
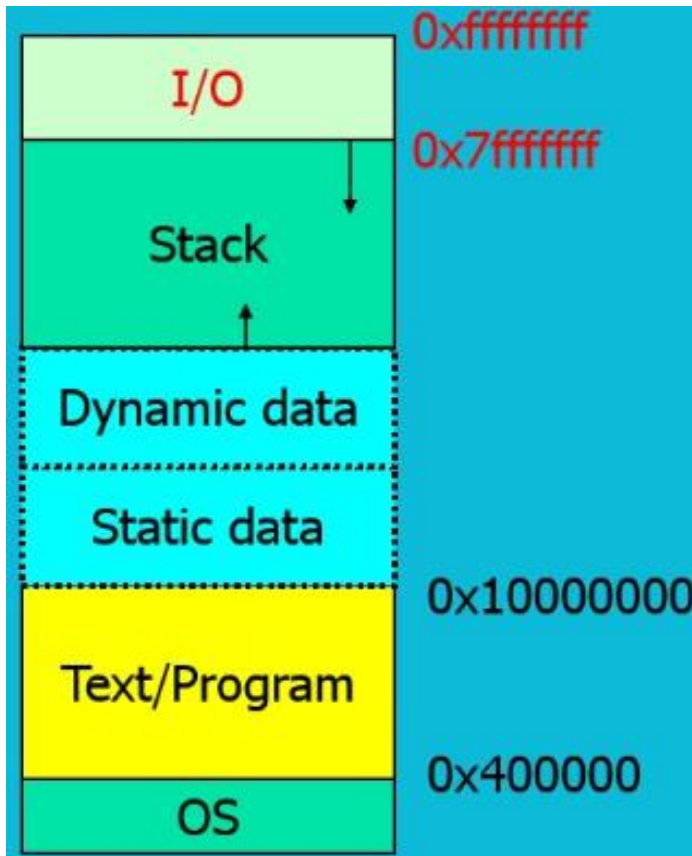


I/O端口编址方式

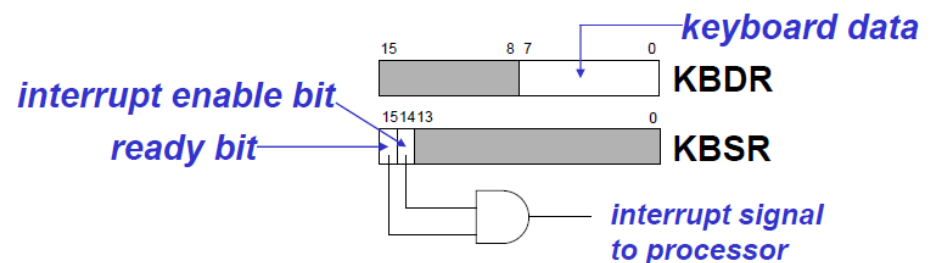
- 存储器映射MMIO：统一编址
 - 在主存储器的地址空间中划出某一区域专门作为外设地址区使用
 - 使用通用的MOV或访存指令（load/store）也可以访问I/O接口
 - Intel MCS-51、MIPS、ARM、RV等采用
 - **Protection via page table【IOMMU】**
- 端口映射PMIO：独立编址
 - I/O端口和存储器独立编址
 - ISA分别设立访存指令和I/O指令（in/out）
 - 由“选择”指示地址类型：M/I/O
 - x86采用



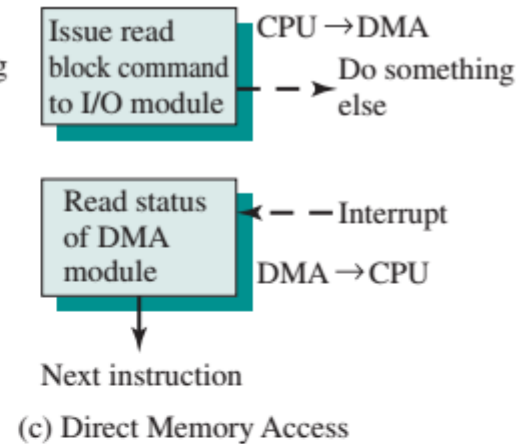
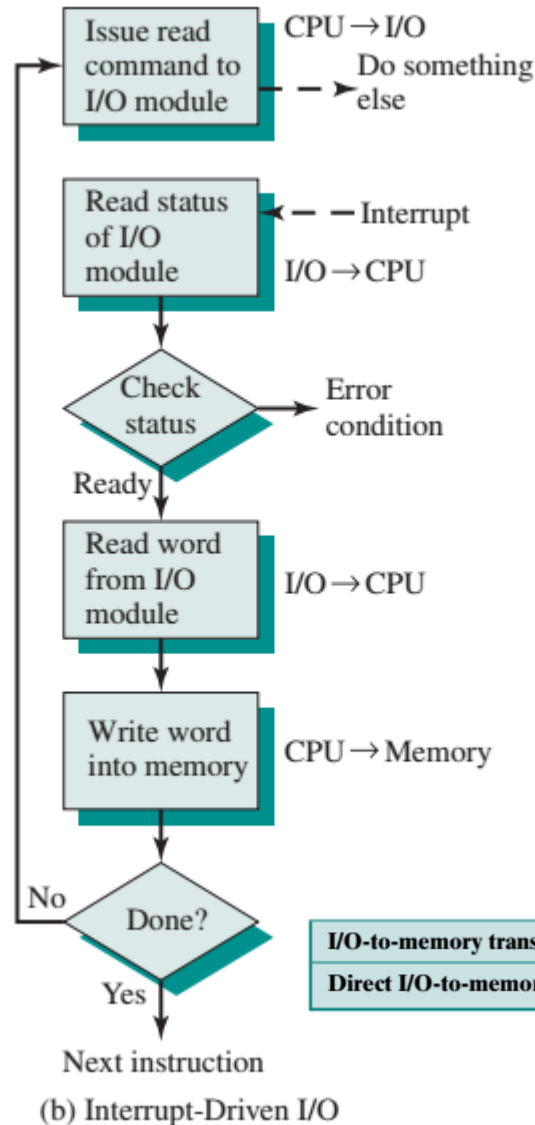
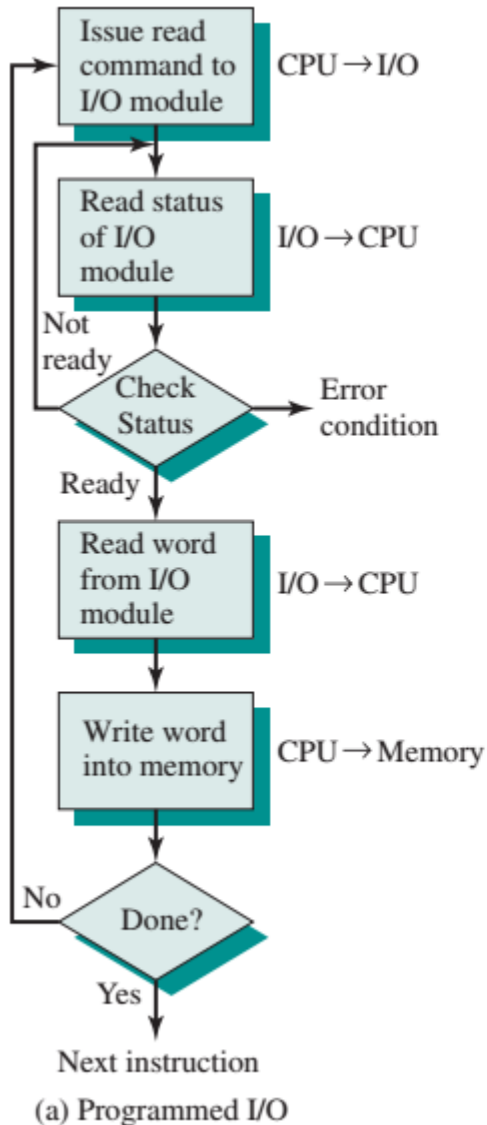
MMIO例: Keyboard example



getchar():
reads an ASCII byte from keyboard and deposits it in \$v0. $\$s0 \leftarrow 0xf0000000$



Three Techniques for Input of a Block of Data



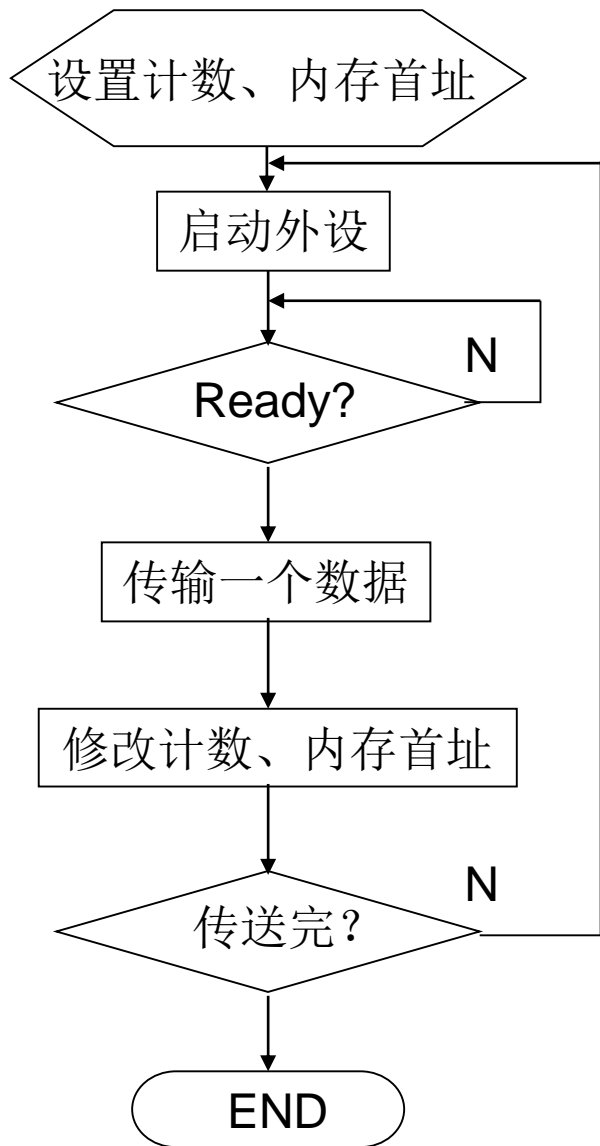
- I/O控制方式
 - 通过CPU
 - 不通过CPU

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

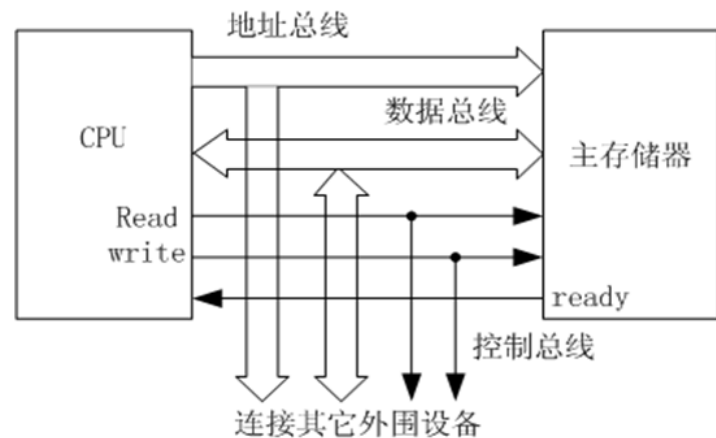


唐5.4 程序查询方式 programmed I/O

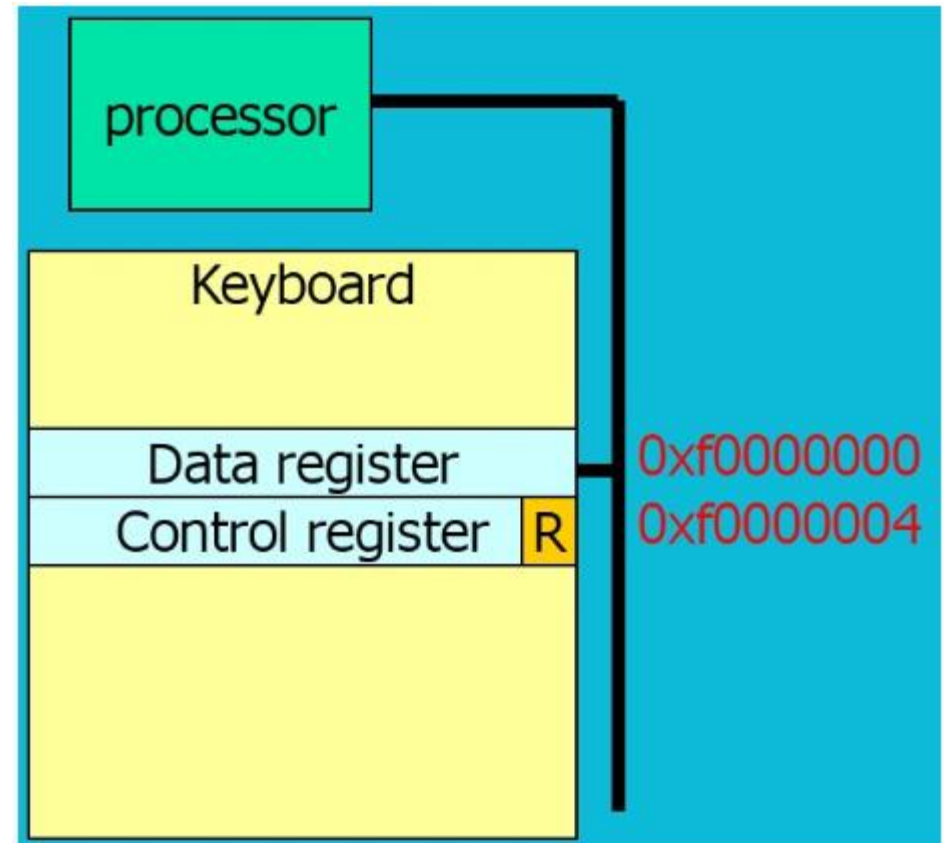
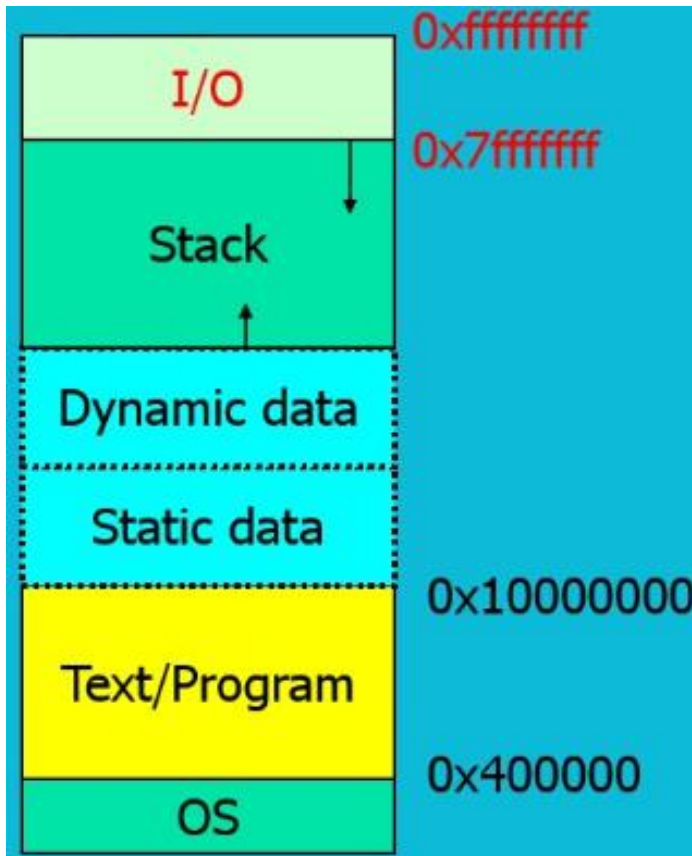
程序查询方式：轮询，同步



- 由CPU控制数据传输过程
 - 单字，多字
 - 等待，低效（多次访问总线）
 - Mouse至少polling 30次/s
 - Polling开销：~600cycle/次
- 多设备轮询



程序查询：Keyboard example



getchar(): **\$s0:** 基址
reads an ASCII byte from keyboard and deposits it in \$v0. **\$s0** ← 0xf0000000

```
poll: lw $t0, 4($s0)
      andi $t0, $t0, 0x1
      beq $t0, $zero, poll
getchar: move $v0, 0($s0) ?
```



唐5.5 程序中断方式

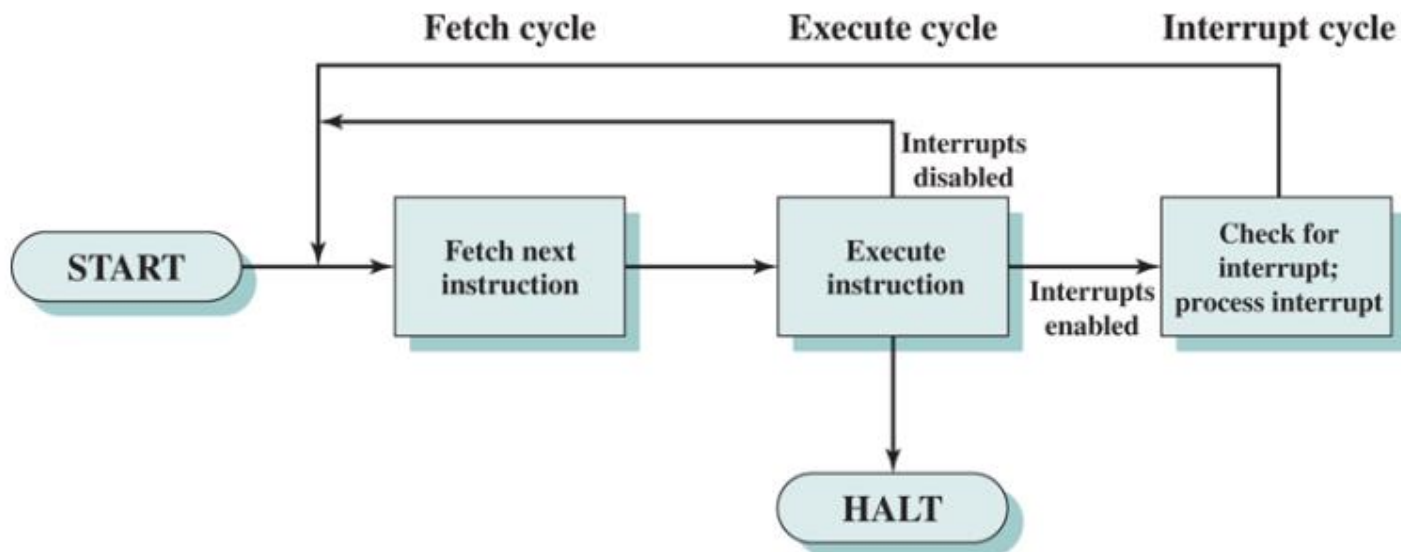
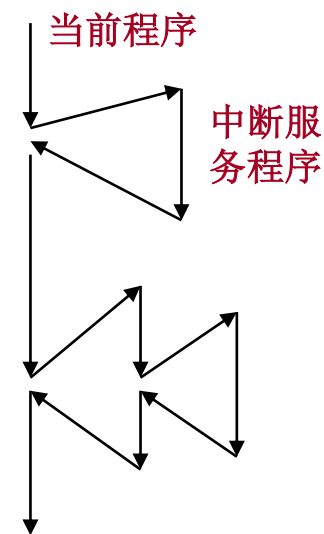
中断驱动I/O

(§5.5、 §8.4)



中断概念

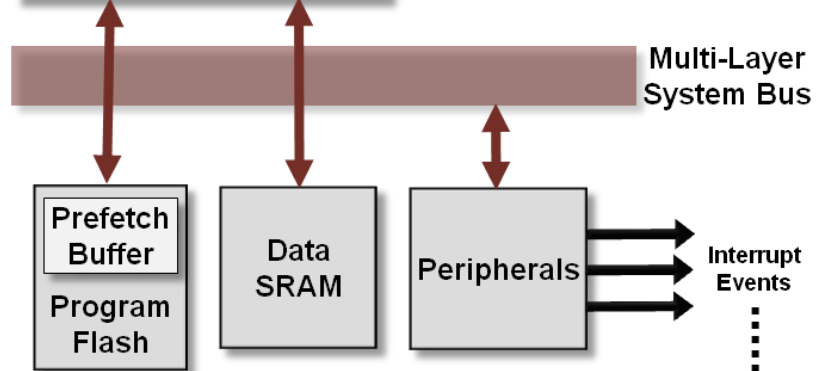
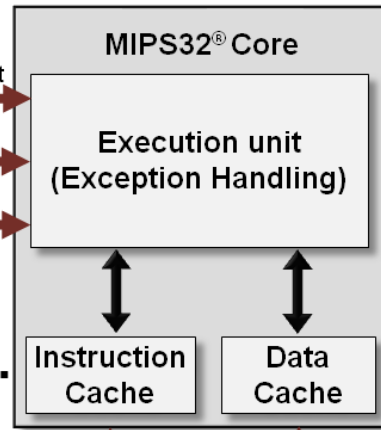
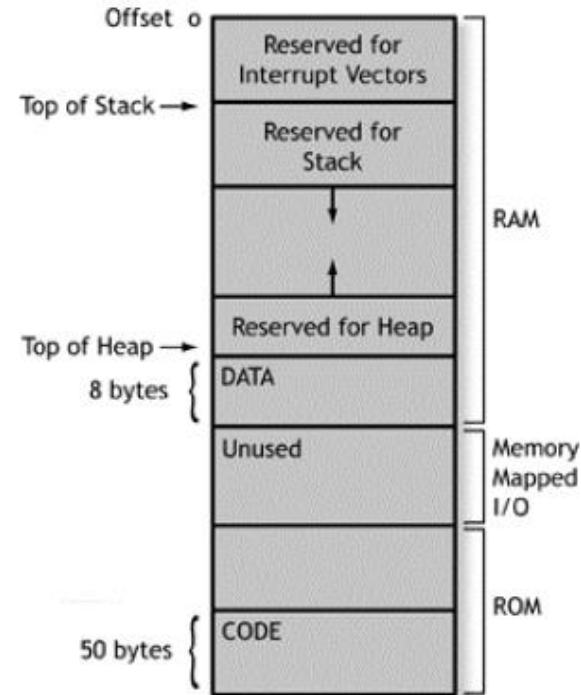
- 中断的概念
 - 暂停当前程序的执行，转而执行其他程序，在它们执行完成后再恢复被中断程序
 - 允许一个处理器“同时”（并发）执行多个任务
 - 响应外部事件（I/O）
- 中断系统
 - CPU (CSR)：中断允许（开关中断指令），中断嵌套
 - 中断控制器IC：判优，屏蔽，发中断请求INTR，中断源
 - ISR：中断服务例程
- 中断周期
 - IC发请求，CPU中断查询INTA，识别中断源，转ISR



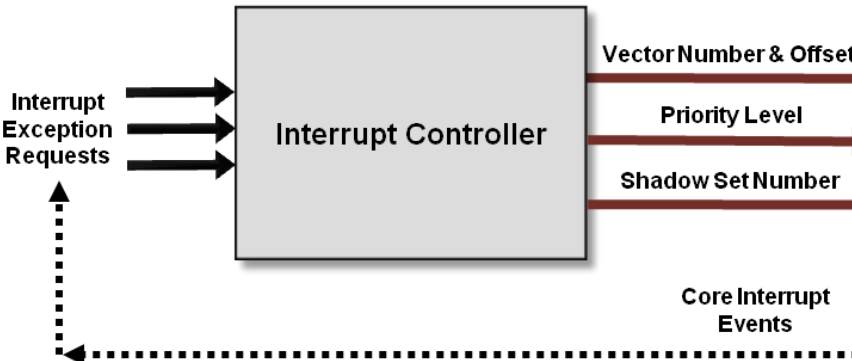
外部中断：中断源，类型，处理机构



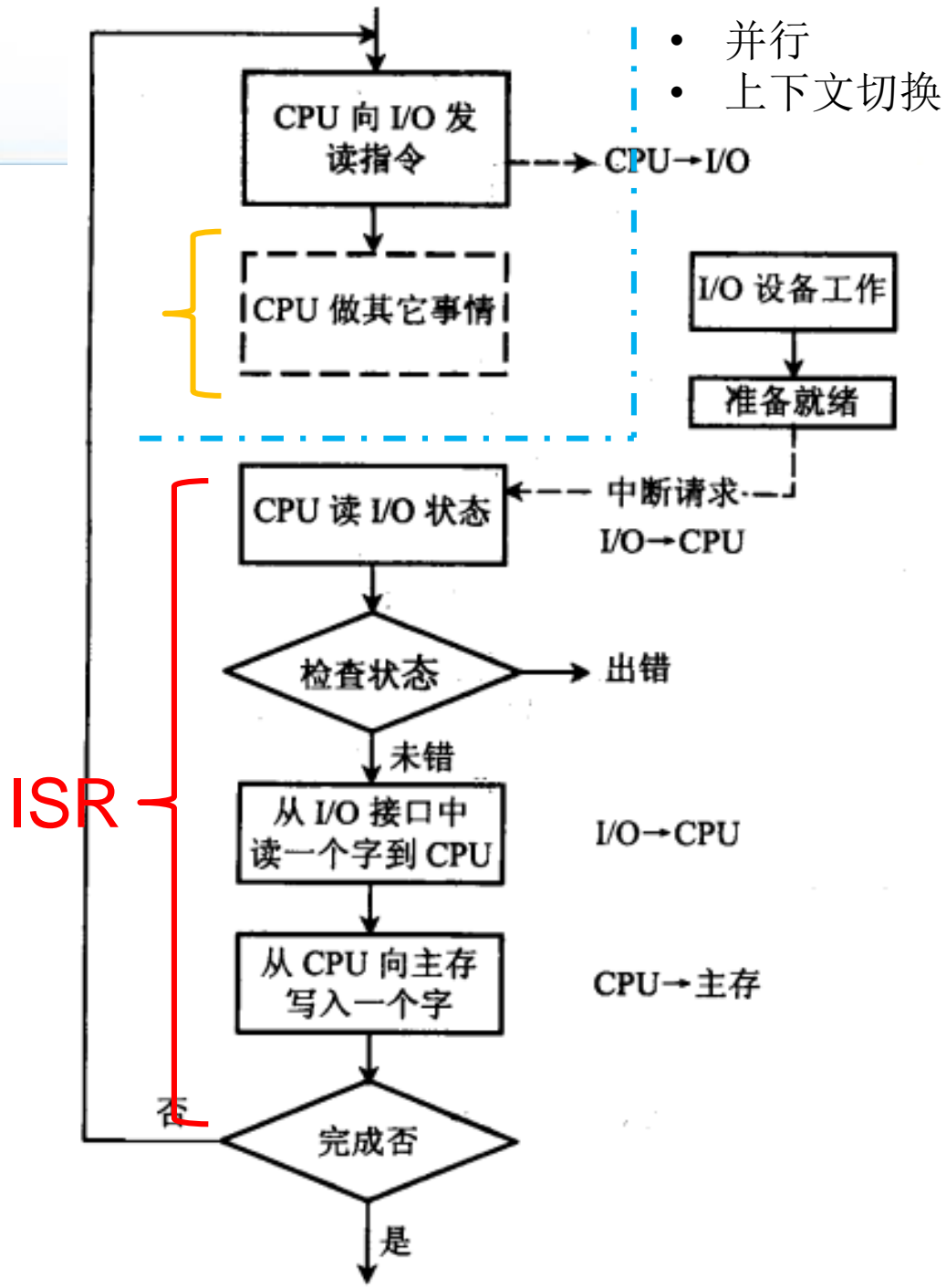
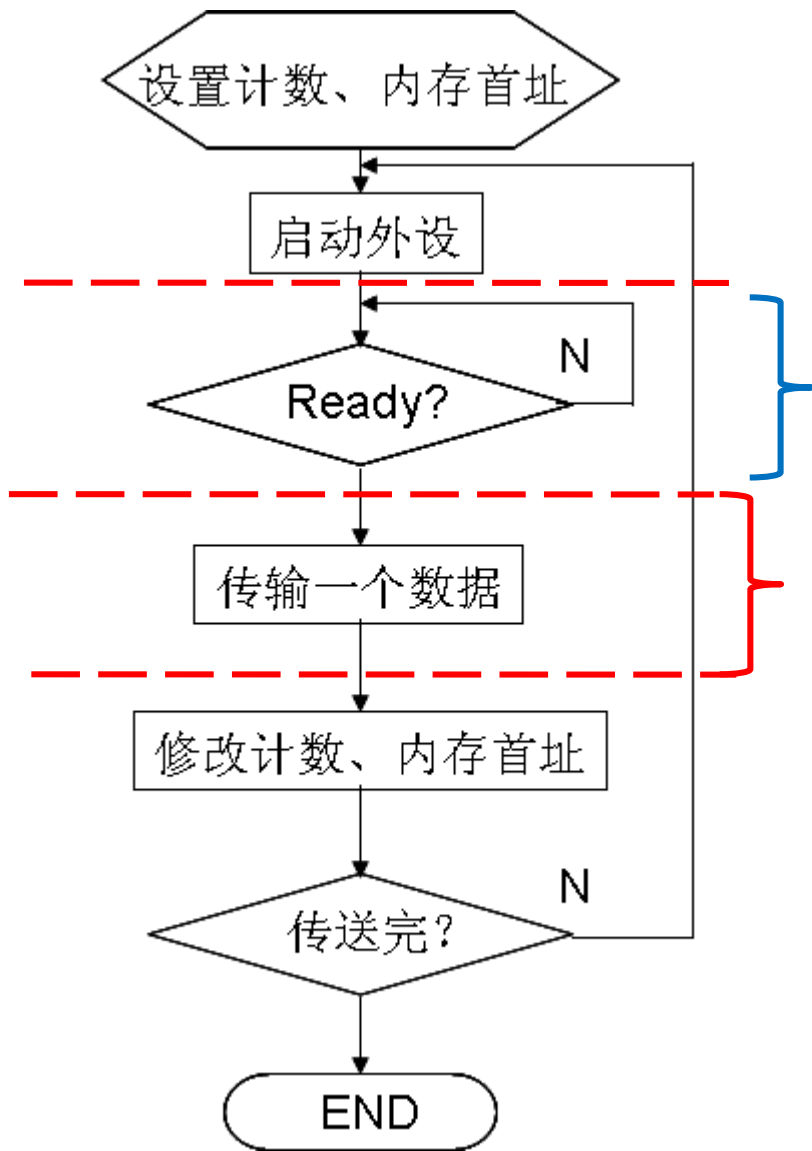
- 处理器异常处理
 - EPC, CAUSE
 - CSR: 中断禁止【】
 - Stack: 抢占与嵌套



- 外部中断源：外设，核间通信
 - 可屏蔽
 - 时钟、键盘、鼠标...
 - 不可屏蔽
 - 访存错、设备错、电源、...
- 中断控制器：中断源管理
 - 判优，屏蔽、挂起pending



中断I/O过程

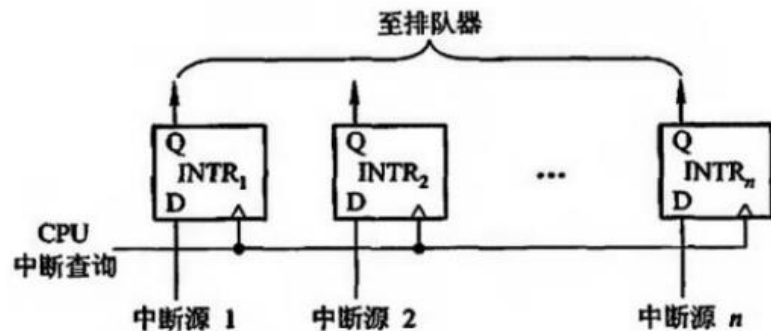
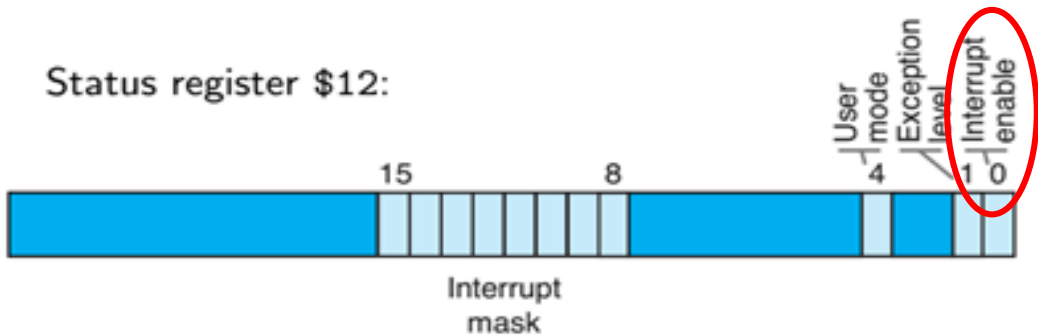


关中断， 中断查询， 中断判优， 中断屏蔽

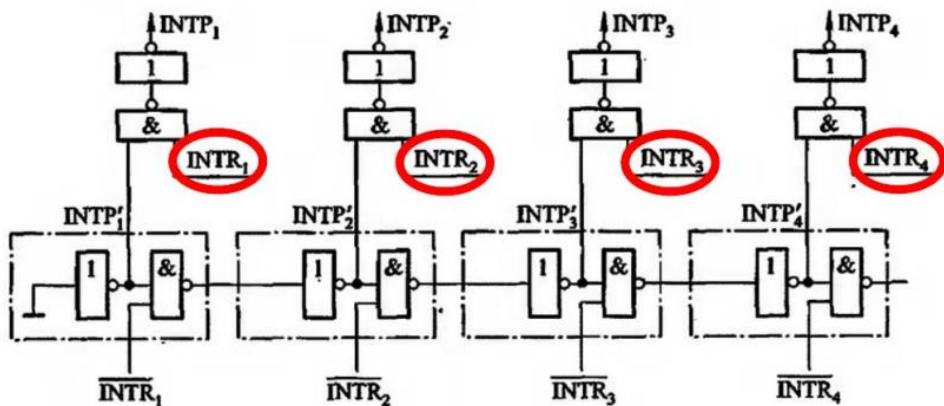


CSR: 开关中断

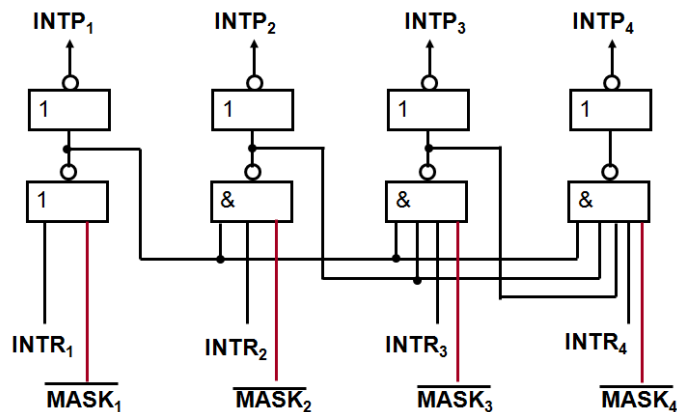
Status register \$12:



唐图8.29 CPU中断查询INTA

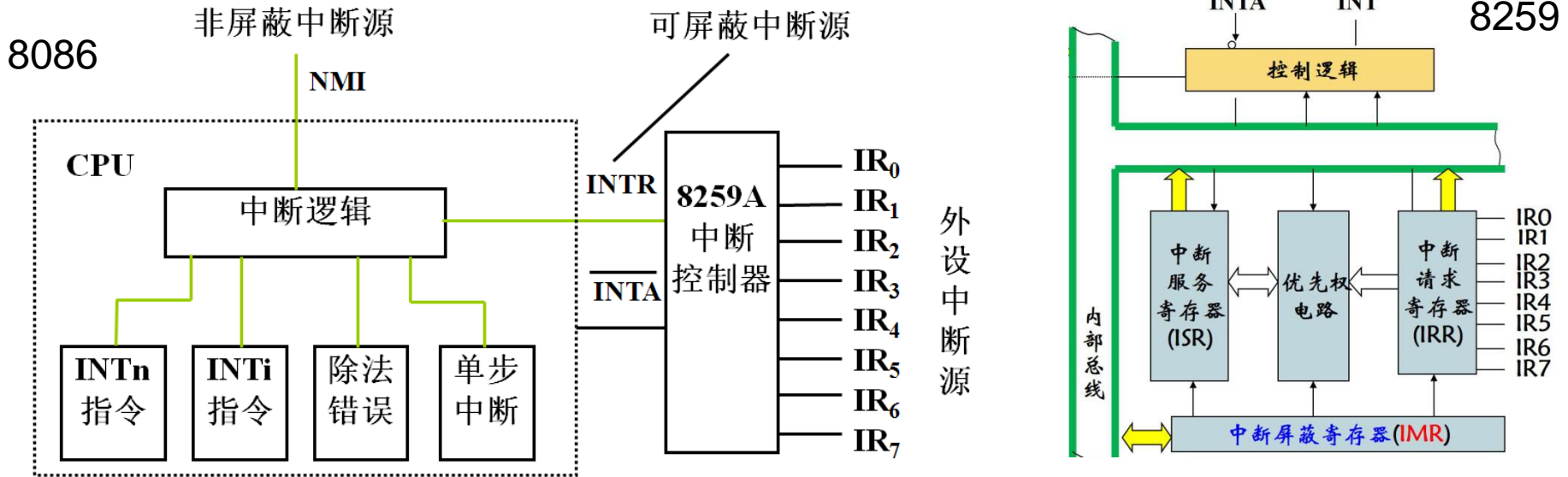


唐图5.38 链式排队【类似“链式仲裁”总线】



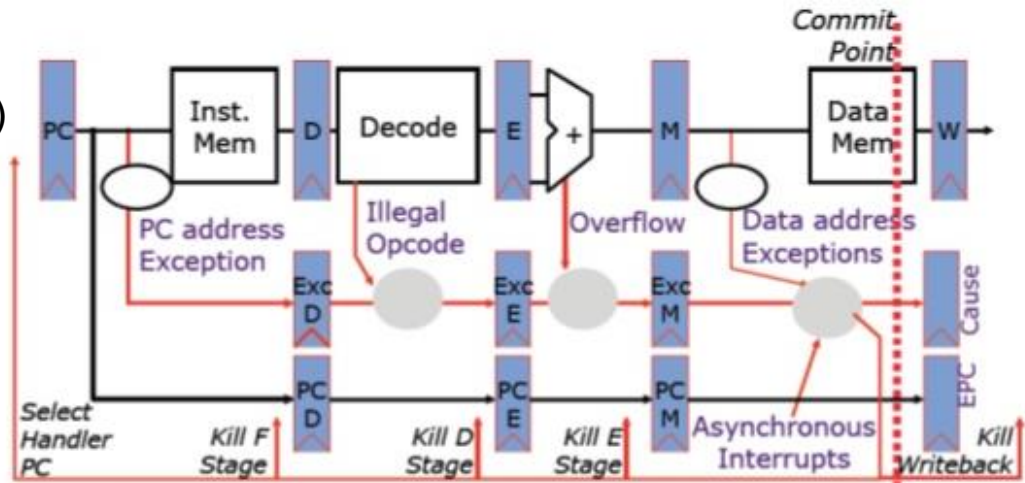
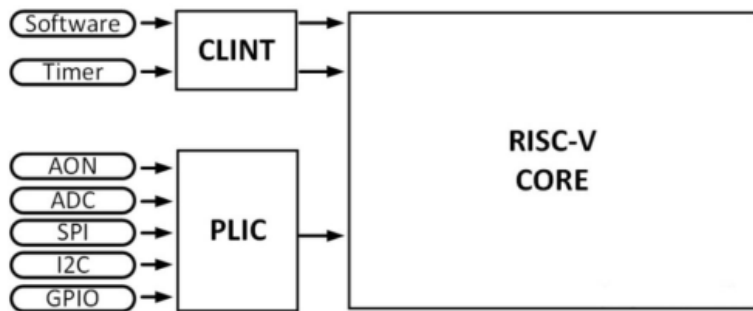
唐图8.25 集中判优与屏蔽

中断请求与响应：中断周期

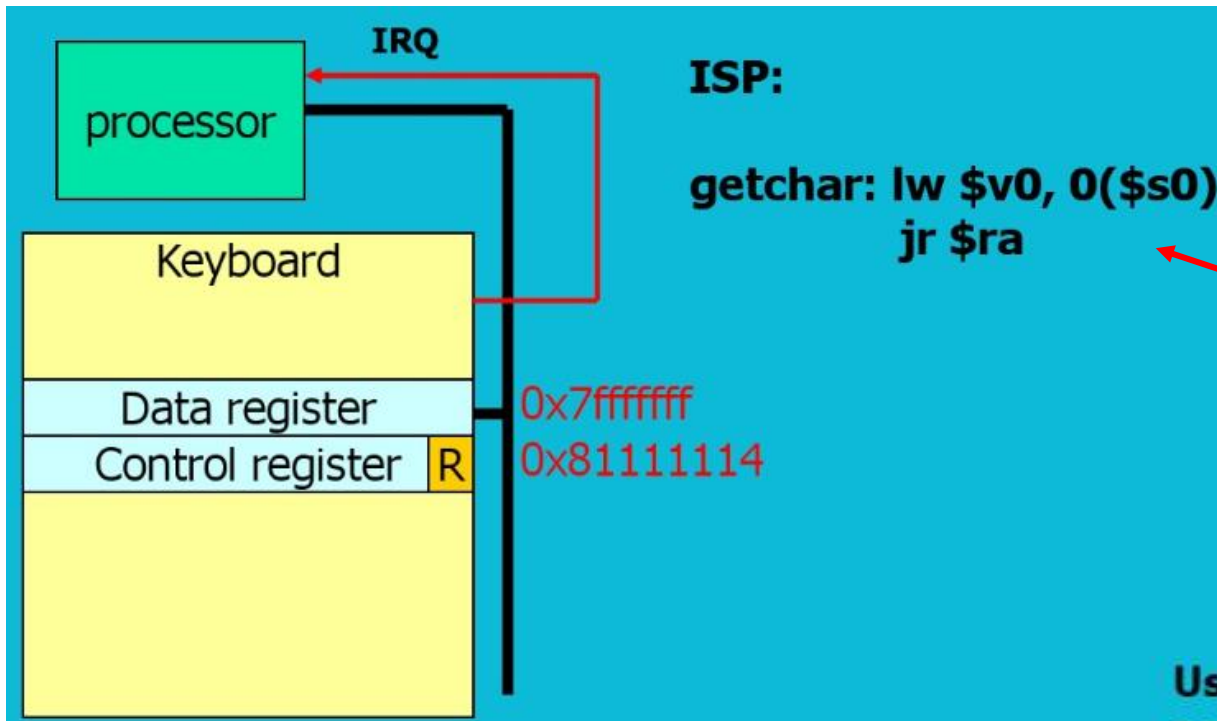


RV:

Core Local Interruptor (CLINT)
Platform-Level Interrupt Controller (PLIC)



Interrupt driven keyboard

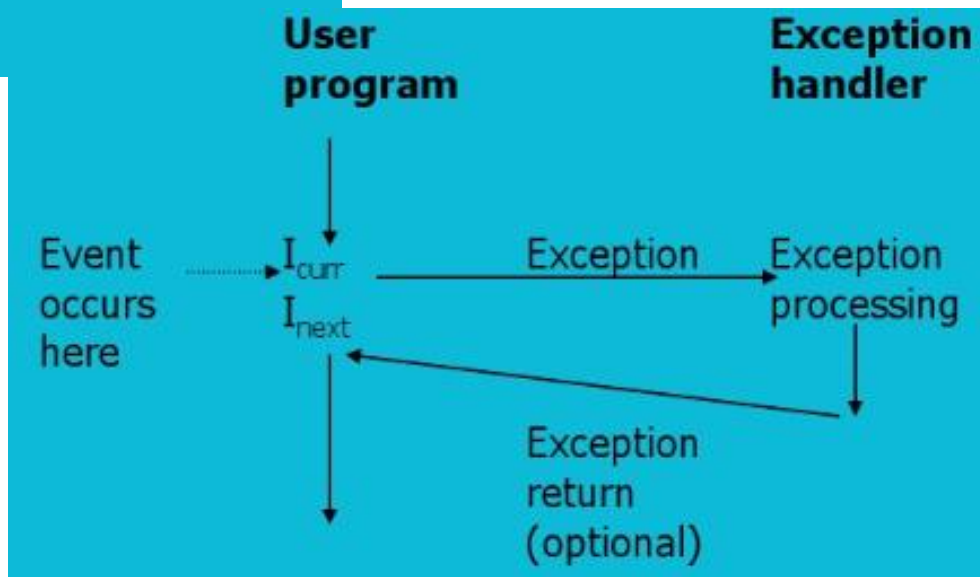
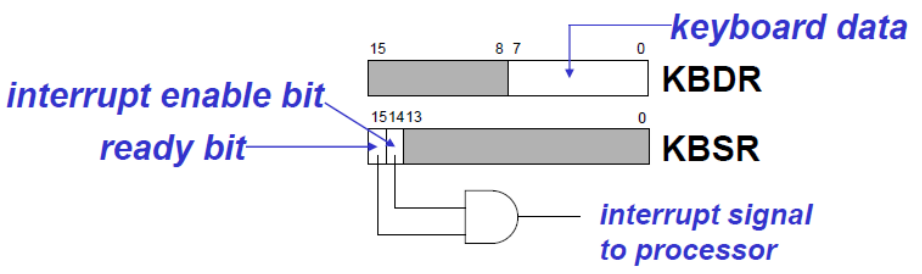


\$s0: 基址

注意: ISP中不能等待!

```

poll: lw $t0, 4($s0)
      andi $t0, $t0, 0x1
      beq $t0, $zero, poll
getchar: move $v0, 0($s0) ?
    
```



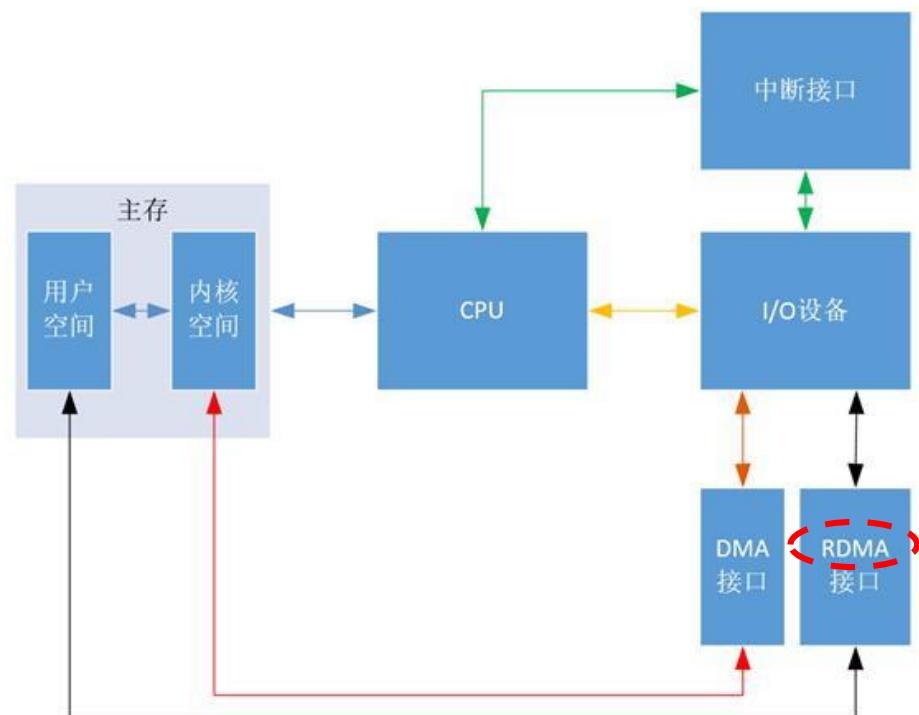
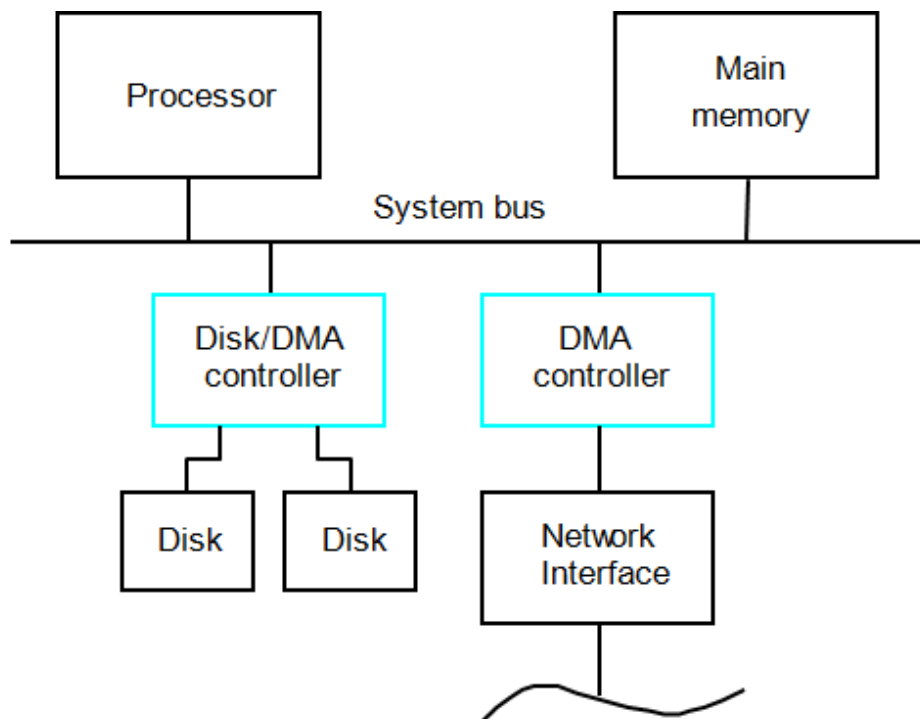


唐5.6 DMA方式



唐5.6 DMA方式

- 注意： **本节系统组成的假设**
 - 8086处理器（兼总线仲裁器），单总线（系统总线）
 - 多个DMAC
 - 一个DMAC可管理多个外设（硬盘）





DMA方式：I/O并行化

- 高速、**批量**数据传输

- 硬件提升I/O性能

- I/O任务与计算任务并发

- 批量：页（扇区），数据包

- DMA访问方式：总线仲裁，**单总线/单CPU**

- 处理器暂停：如8086架构（无Cache）

- COA称为“周期窃取”，“常用”！

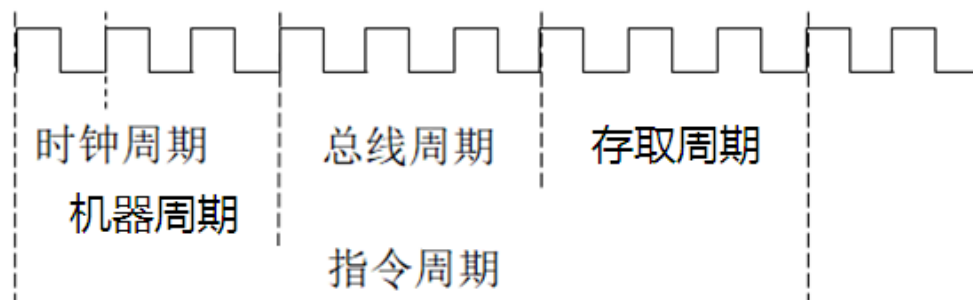
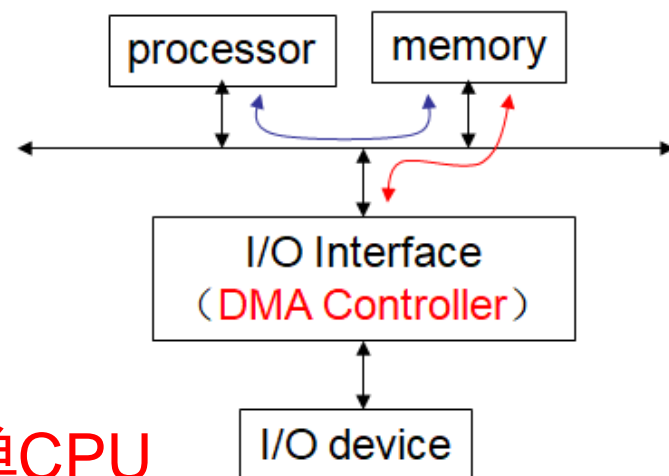
- 周期挪用（周期窃取）：处理器不用时用

- 唐本白本称“广泛采用”？

- **交替访问**

- DMAC：过程控制

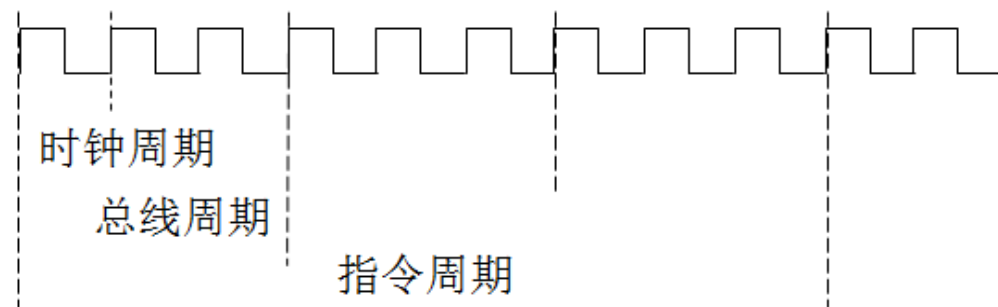
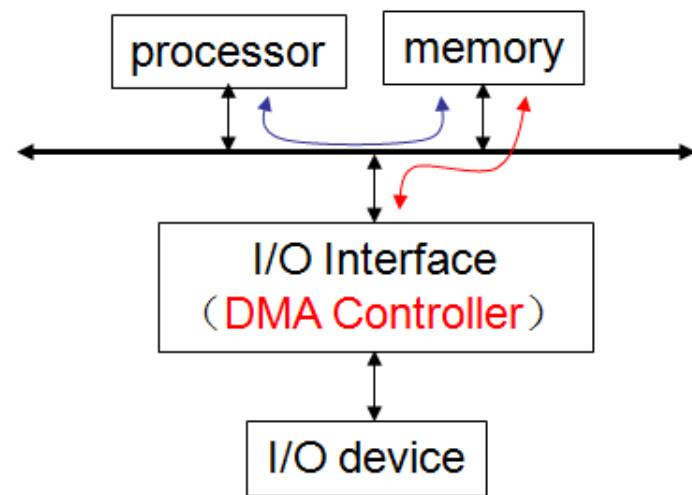
- **协处理器**





处理器暂停方式

- 处理器兼任总线仲裁器：8086
- CPU暂停方式过程
 - DMA向CPU申请总线
 - CPU暂停
 - DMA传输
 - DMA释放总线
 - CPU继续
- 并未真正实现并发

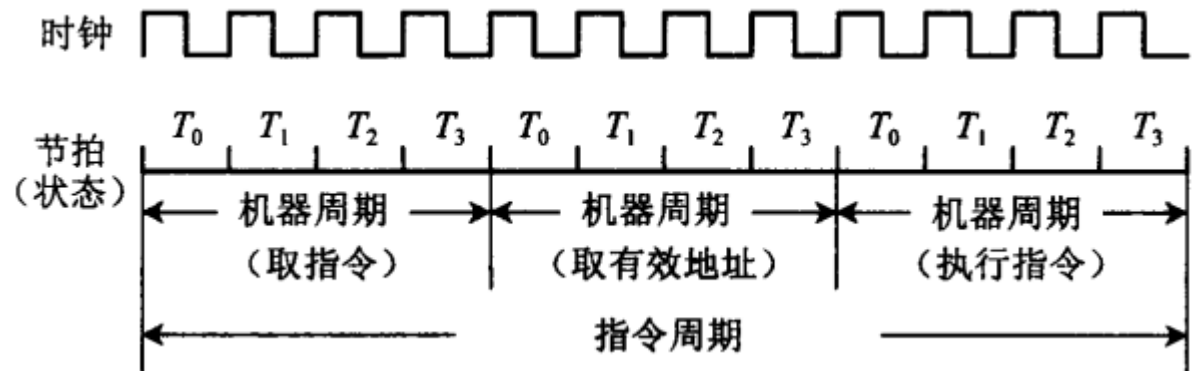
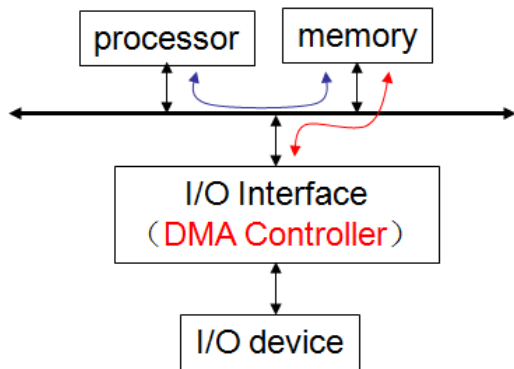


- 回顾：总线主设备 (Master)、从设备 (Slaver)
 - 总线仲裁



周期窃取 (cycle stealing)

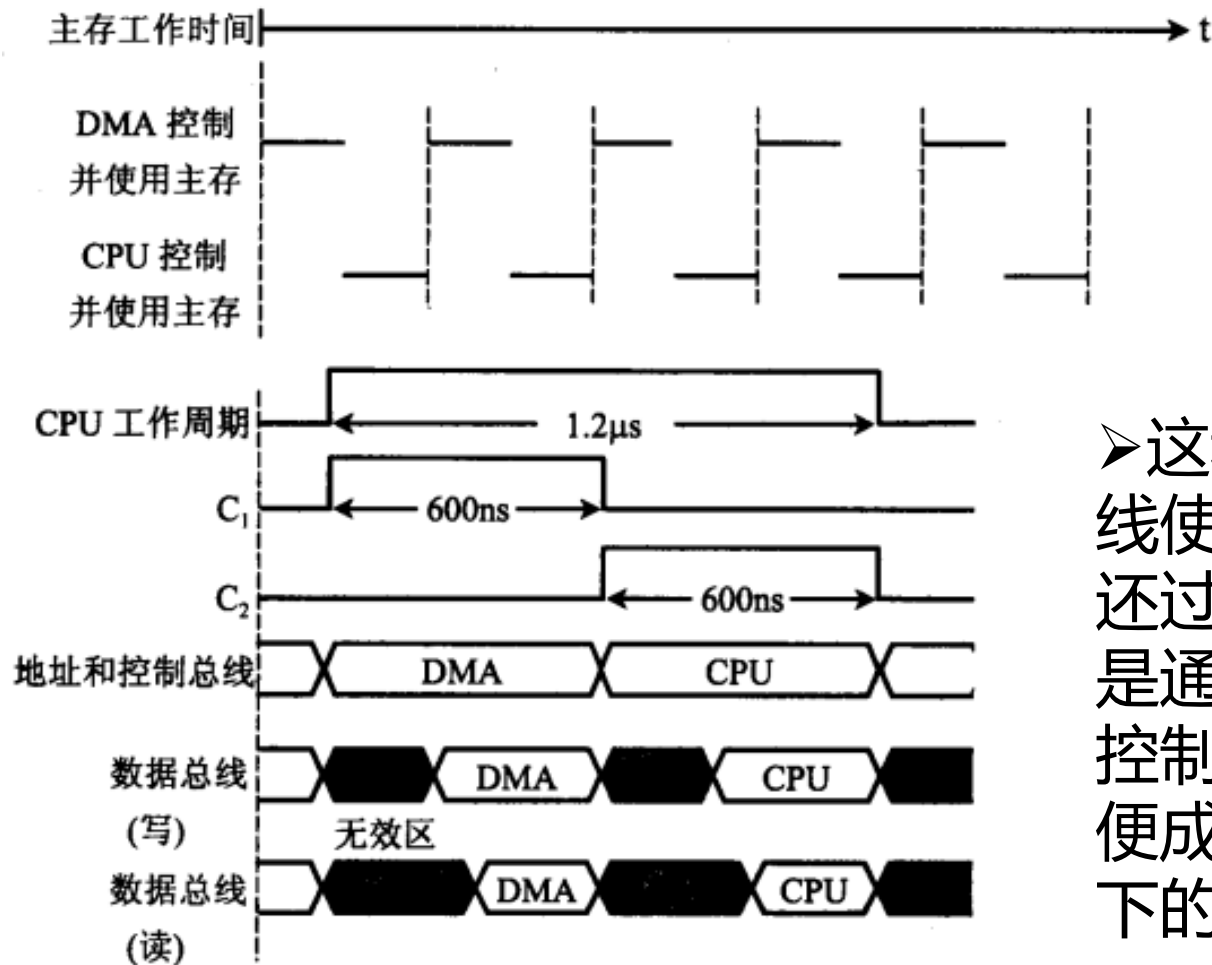
- 当I/O设备发出DMA请求时，I/O设备便挪用或窃取总线占用权一个或几个主存周期。
- 有三种情况
 - CPU不访存（复杂多周期指令mul，Cache hit）：DMA使用
 - CPU正在访存：DMA等待，然后获得总线使用权
 - CPU与DMA同时发生：**DMA优先**，窃取一到二个**存取周期**（否则数据丢失）





交替访问——周期扩展

- 将CPU工作周期延长，分成两段。



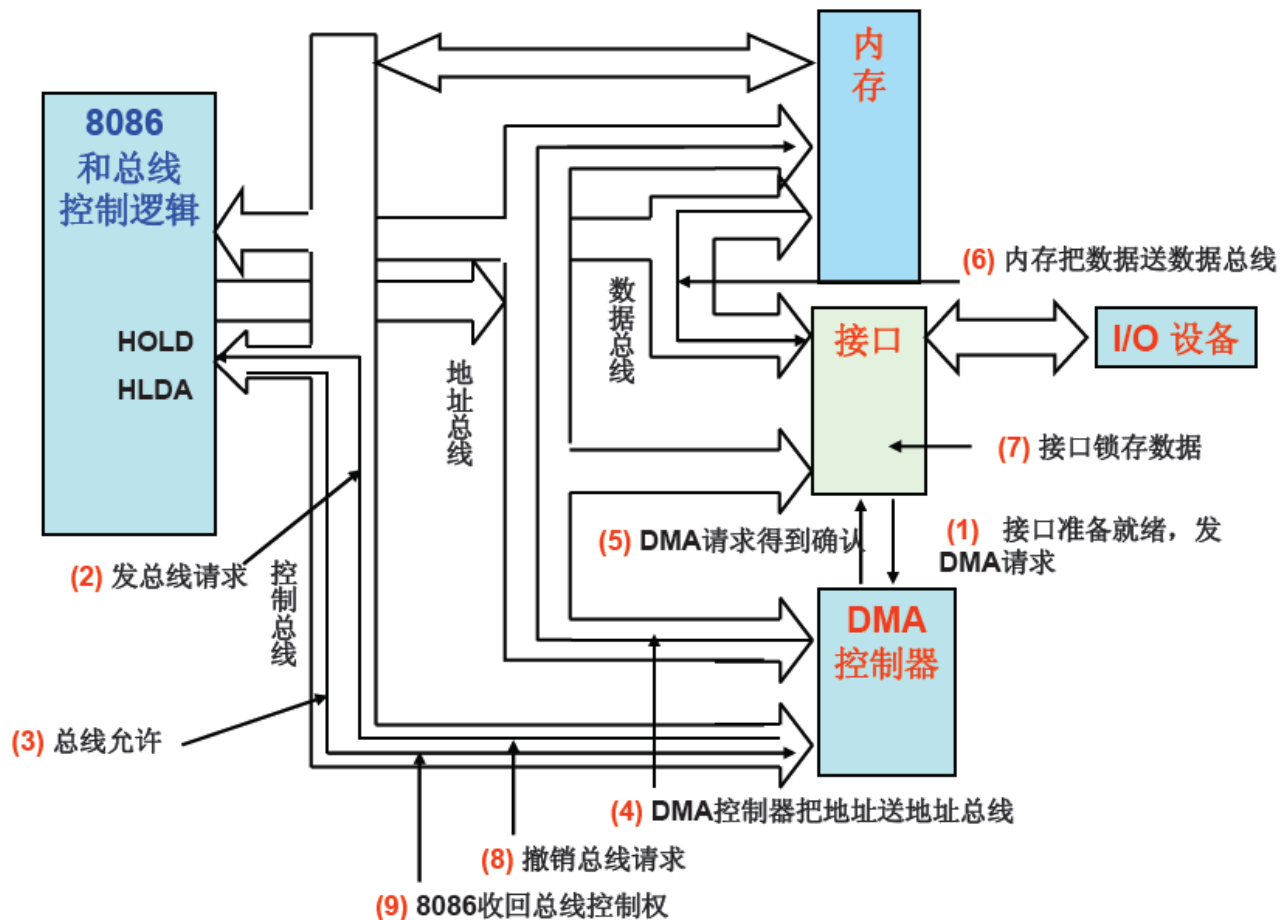
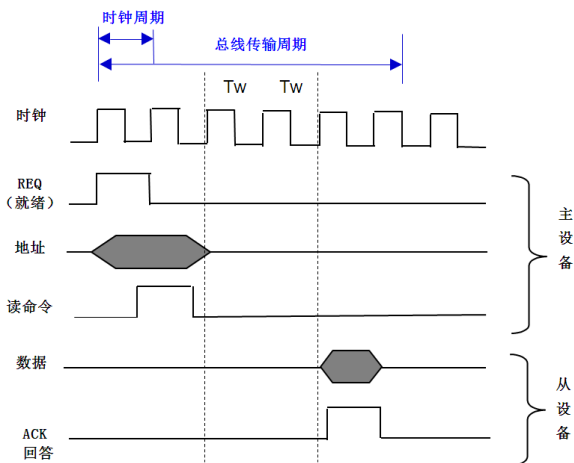
➤这种方式**不需要**总线使用权的建立和归还过程，总线使用权是通过C₁和C₂分别控制的。实际上总线便成了C₁和C₂控制下的多路转换器。

(c) DMA 与 CPU 交替访问

DMA控制I/O：CPU暂停

I/O类型

- 内存与外设
- 内存与内存
- 外设与外设



例：内存->I/O接口->设备

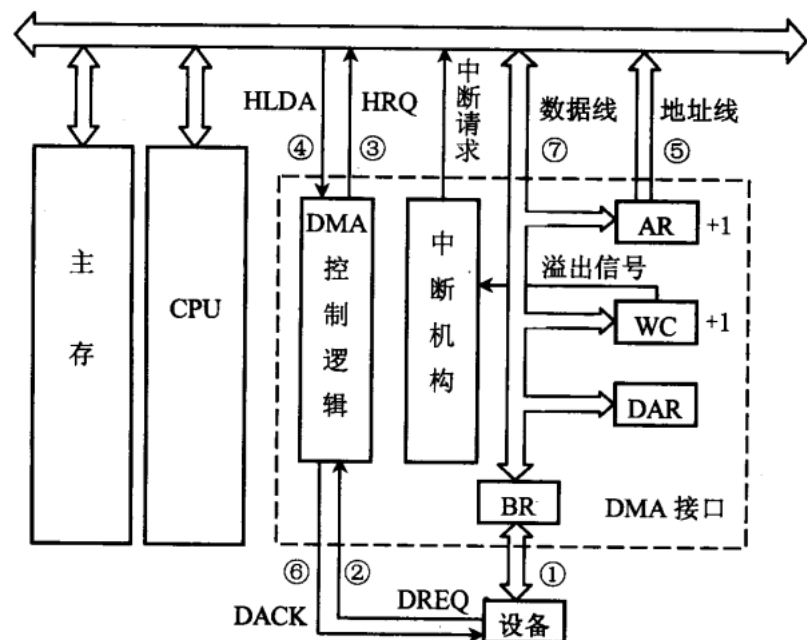
- 传输类型：单字节，块传输，请求传输
- 三个阶段：预处理（CPU），数据传输（DMAC），后处理（ISR）
 - CPU（8086）的总线控制器负责总线仲裁

DMAC的基本组成

➤ DMA控制器(DMAC)也称作DMA接口

- 主存地址寄存器 (Address Register)
 - 每传输一个数据 (字节、字), 地址加1
- 设备地址寄存器 (DAR)
 - 外设中数据块地址或当前设备的设备号
- 字计数器 (word counter)
- 数据缓存寄存器 (Buffer Reg)
 - 完成数据格式转换等
 - 数据传输不一定经过DMAC
- DMA控制逻辑
 - DREQ-DACK、HRQ-HLDA
 - 多个外设: 优先级控制
- 中断机构
 - DMA结束: WC溢出, 进行“后处理”
 - DMA出错

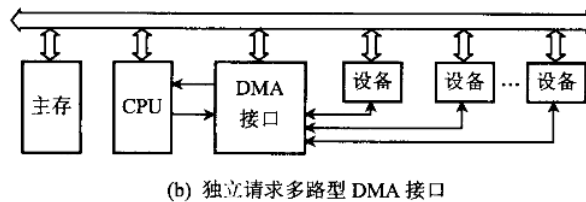
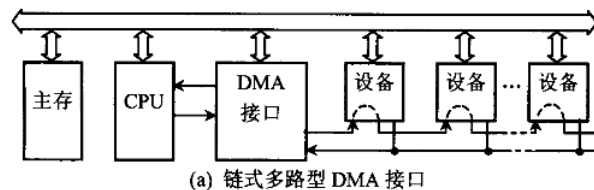
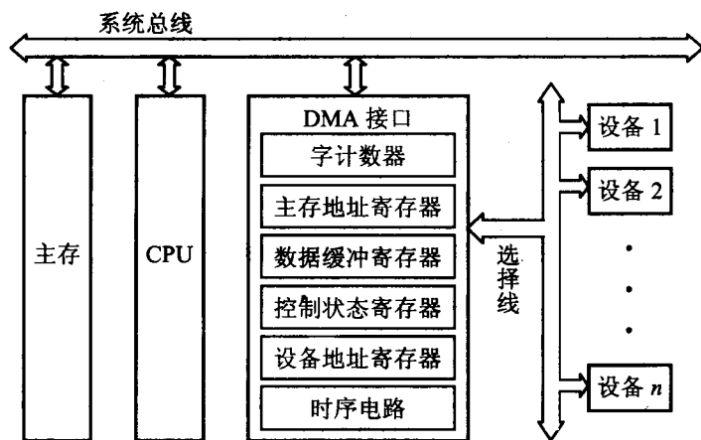
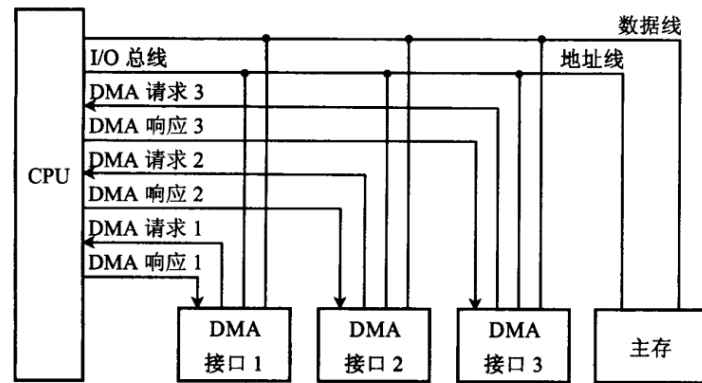
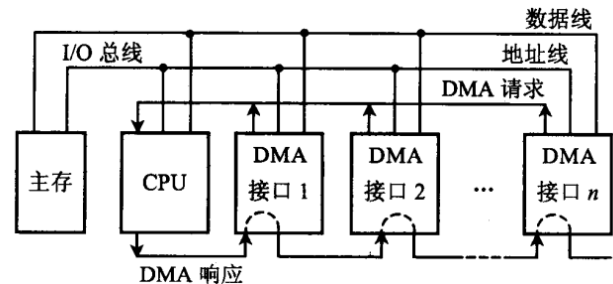
Address	Function
0x00 <small>地址</small>	1: start, 0: stop <small>指令作用</small>
0x04	Status. 0: ready; 1: busy
0x08	Source address
0x0C	Destination address
0x10	Size



唐图5.47 DMAC

DMA系统连接方式：CPU-DMAC-I/O

- 系统中存在多个DMA通道和多个I/O设备，如何连接？
 - 参考集中式总线仲裁方式
- $\text{DMAC} \leftrightarrow \text{CPU}$ ：多个DMAC
 - 公共请求方式：链式/级联
 - 独立请求方式：星型连接
- $\text{DMAC} \leftrightarrow \text{设备}$ ：单DMAC多个设备
 - 选择型：“软”选择响应设备
 - “计数式”？
 - 多路型：“硬”选择响应设备
 - 链式多路型
 - 独立请求多路型



唐图5.49

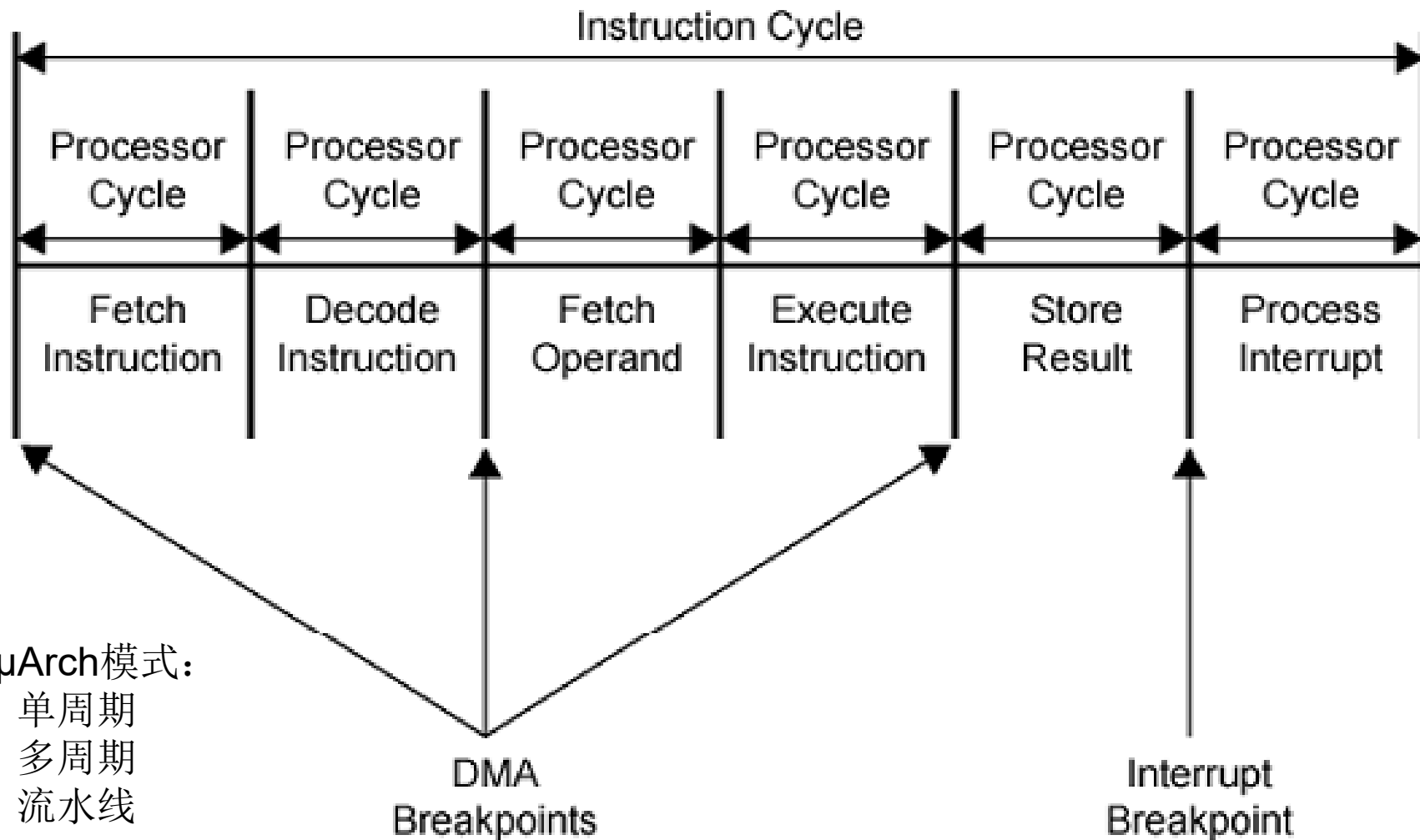
唐图5.50

唐图5.51

DMA and Interrupt Breakpoints



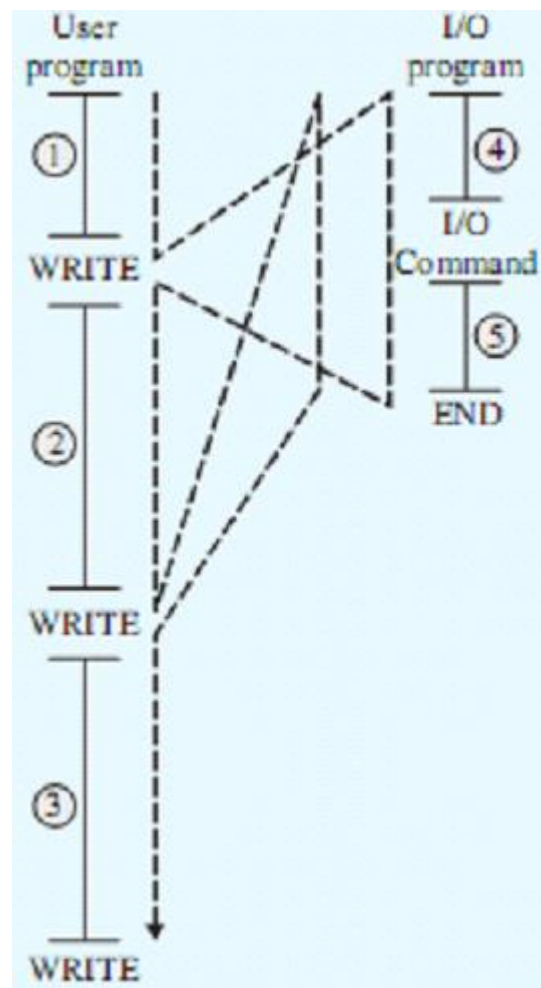
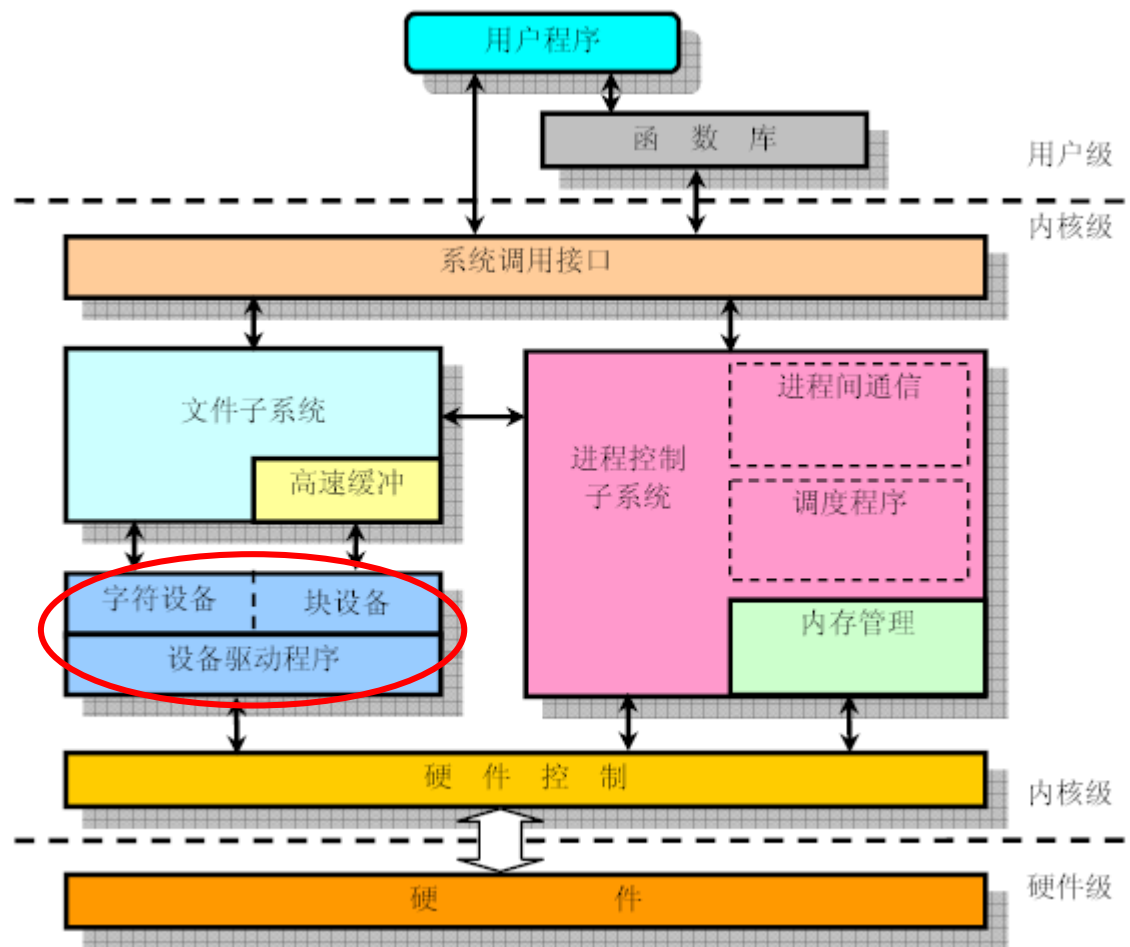
Time →



μArch模式:
单周期
多周期
流水线

软硬件接口：设备驱动程序，API库

- 操作特定设备的程序：字符设备、块设备
- 将所有设备映射成“文件”，【临界区？】



device driver的功能



- **设备管理**

- **Hardware Startup**, initialization of the hardware upon power-on or reset.
- **Hardware Shutdown**, configuring hardware into its power-off state.
- **Hardware Install**, allowing other software to install new hardware on-the-fly.
- **Hardware Uninstall**, allowing other software to remove installed hardware on-the-fly.
- **Hardware Disable**, allowing other software to disable hardware on-the-fly.
- **Hardware Enable**, allowing other software to enable hardware on-the-fly.

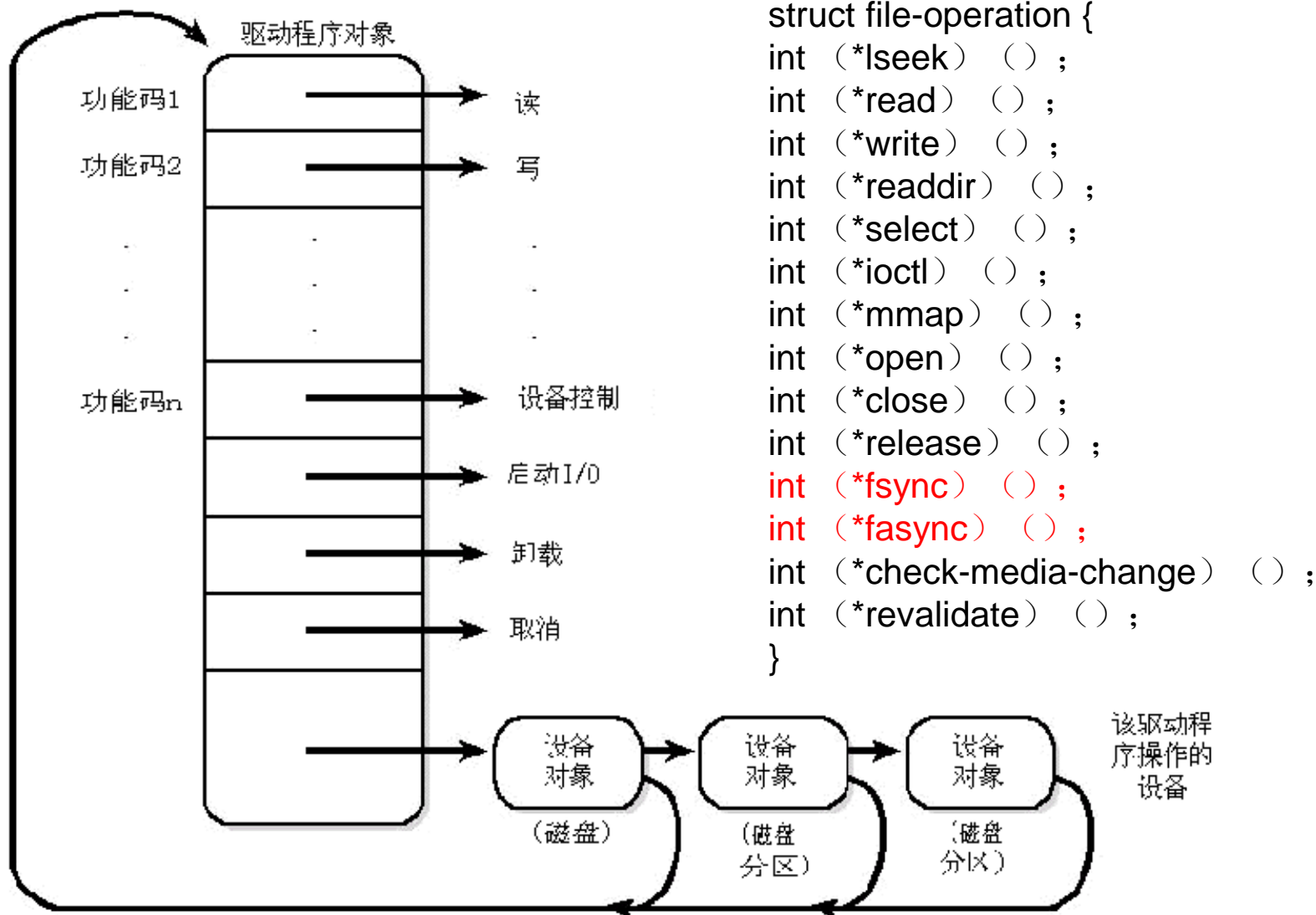
- **读写操作**

- **Hardware Read**, allowing other software to read data from hardware.
- **Hardware Write**, allowing other software to write data to hardware.

- **并发控制**

- **Hardware Acquire**, allowing other software to gain singular (locking) access to hardware.
- **Hardware Release**, allowing other software to free (unlock) hardware.

驱动程序和设备

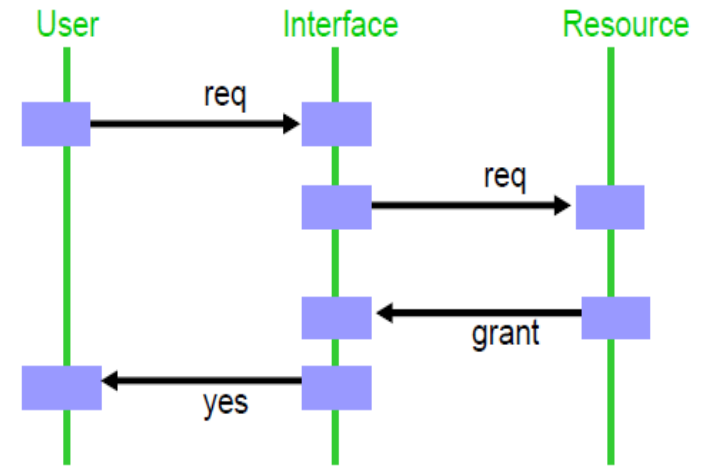




例：LED应用程序

```
int main(void)
{  int fd;
   char led_on = 0x01; //待显示的数据

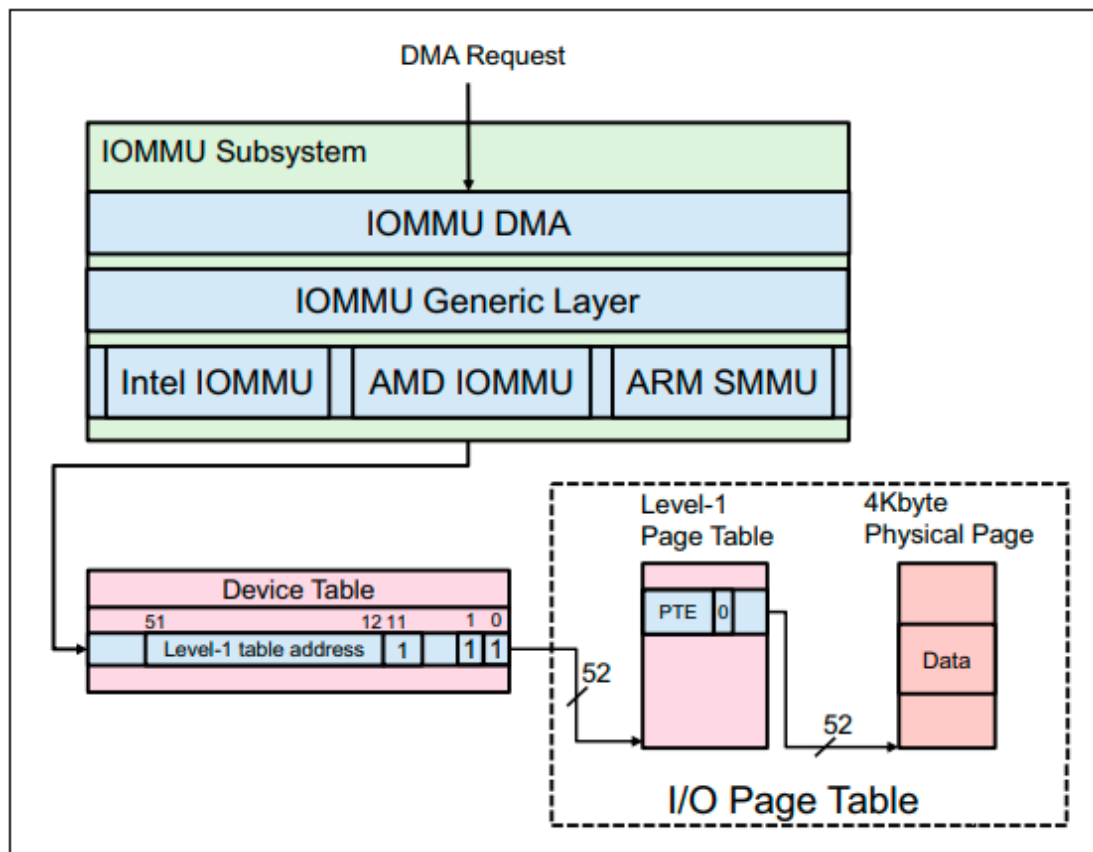
   fd = open("/dev/led/0", O_RDWR); //打开led设备驱动
   if(fd== -1) {
       printf("can not open device\n");
       exit(1);
   }
   write(fd, &led_on, 1); //LED开
   close(fd); //关闭设备文件
   return 0;
}
```





IOMMU

- 将**设备**发出的虚拟地址IOVA通过页表翻译为物理地址
 - DMA Remapping
 - Interrupt Remapping: 中断请求传递到目标虚拟CPU, **虚拟化**



小结

- 内容：工作原理、流程、结构

- 外设原理

- I/O接口的基本工作机制

- I/O系统组成（接口，HW/SW）
 - 编址方式：MMIO，独立编址
 - 控制方式：查询，中断，DMA

- 中断机构组成

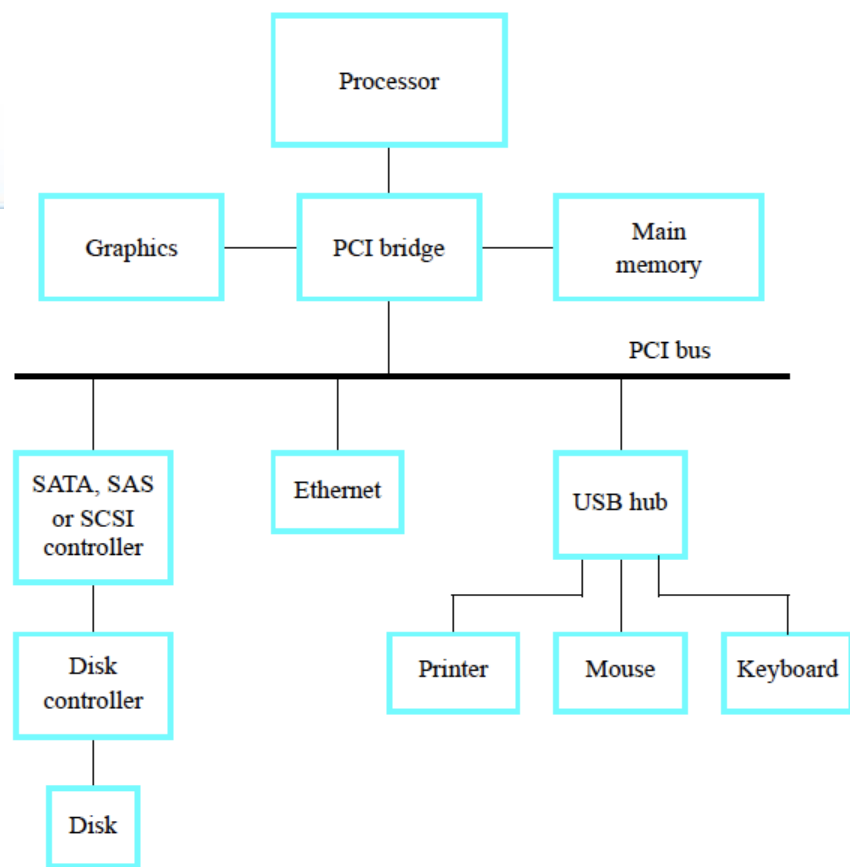
- DMA控制器

- 作业

- 唐：5.4、5.11、8.24

- 实验2（可选）：基于Xilinx ISE，设计一个TinyComputer系统。

- 系统组成：MicroBlaze CPU、on_chip_ram和JTAG UART；
 - 编写并运行一个简单C程序 “hello_world_11xx”；





休息是为了走更远的路!