《计算机组成原理》

（第九讲）

厦门大学信息学院软件工程系 曾文华

2023年6月5日

目录

第9章　输入输出系统

- 9.1　输入输出设备与特性
- 9.2　I/O接口
- 9.3　数据传输控制方式
- 9.4　程序控制方式
- 6.7　异常与中断处理
- 9.5　程序中断控制方式
- 9.6　DMA方式
- 9.7　通道方式
- 9.8　常见I/O设备

9.1 输入输出设备与特性

- 输入输出系统主要用于实现CPU与外部设备、外部设备与主存之间的信息交换。

- 输入输出系统是典型的软硬件协同系统，既包括I/O设备、I/O接口、总线、I/O管理部件等I/O硬件系统，也包括驱动程序、软件访问接口、用户程序等I/O软件系统。

- 输入输出设备（外部设备）：
  ① 输入设备：键盘、鼠标、扫描仪、摄像头等。
  ② 输出设备：显示器、打印机等。
  ③ 输入输出设备：磁盘、网卡等。

- 输入输出设备的特性：
  ① 异步性：CPU与外部设备速度相差巨大，两者之间必须采用异步的方式进行数据交换。
  ② 实时性：不论是慢速的设备，还是高速的设备，设备准备好数据后，CPU都应及时处理；如键盘、鼠标按下后，CPU应及时响应；高速设备的数据如果得不到及时处理就会丢失；现场测试和实时控制设备的信息如果不能及时处理，可能会导致严重的灾难。
  ③ 独立性：外部设备应采用标准的总线接口与CPU进行连接，使输入输出与具体的设备类型无关，这便是输入输出的独立性。

9.2 I/O接口

• 9.2.1 I/O接口的功能

– 计算机中的所有I/O设备均使用I/O接口（总线接口）与总线相连，CPU使用设备地址经总线与I/O接口通信来访问I/O设备。

– I/O接口应具有以下的功能：

① 设备寻址：接收来自总线的地址信息，经译码电路，选择对应外部设备中的寄存器或存储器。

② 数据交互：实现外部设备、主存与CPU之间的数据交换，这也是接口最基本的功能。

③ 设备控制：传送CPU命令。

④ 状态检测：反映外部设备的工作状态。

⑤ 数据缓冲：匹配CPU与外部设备的速度差距。

⑥ 格式转换：实现数据格式转换或逻辑电平信号转换。

– 此外，接口还应具有中断、时序控制和数据检错、纠错等功能。

• 9.2.2 I/O接口的结构

– I/O接口内部主要包括总线接口和内部接口两部分（图9.1）：

- 总线接口：连接总线的总线接口必须按总线标准进行设计，这部分逻辑为接口的标准部分。
- 内部接口：连接设备的内部接口逻辑因设备而异，是非标准的。

– I/O接口应包括以下基本的功能部件：

① 数据缓冲寄存器（DBR：Data Buffer Register）

② 设备状态寄存器（DSR：Device Status Register）

③ 设备命令寄存器（DCR：Device Command Register）

④ 设备存储器：例如显卡中的显存。

⑤ 地址译码器

⑥ 数据格式转换逻辑：进行串并或并串传送的转换。

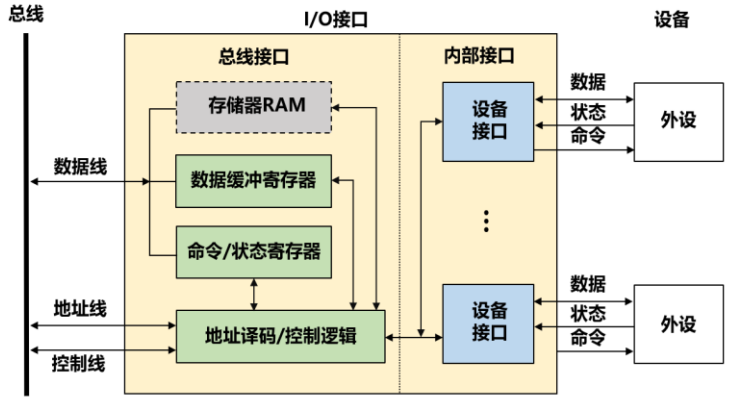


图9.1 I/O接口通用结构

• 9.2.3 I/O接口的编址

– I/O接口中的命令寄存器、状态寄存器、数据缓冲寄存器、设备存储器，都由CPU进行统一的设备地址分配，并通过对应的设备地址访问。

– 通常有两种编址方法：统一编址、独立编址。

– 1、统一编址

- 也称内存映射编址，Memory-mapped，外部设备与内存地址统一编址，两者在逻辑上处于同一个地址空间，通过不同的地址区域来区分是访问内存，还是访问外部设备；图9.2。
- 统一编址不需要设置专用的I/O指令，访存指令就可以访问外部设备。
- 由于I/O接口中的数据是动态变化的，因此统一编址中的I/O地址空间不能使用cache进行缓存，否则CPU无法了解设备状态的实时变化；在C语言中接口变量应该声明为volatile型，表明该变量是会经常自动变化的。

统一编址CPU访问存储器和I/O的代码（MIPS汇编语言）

```
lw  $t0,0x00000004      #从ROM读取一个字
sw  $t0,0x00000004      #写ROM（会产生总线错误）
lbu $t0,0x00010001      #从内存读取一个字
sb  $t0,0xff000002      #写一个字到内存
lbu  $t0,0xffff0000     #从I/O设备读一个字
sb  $t0,0xffff0004      #写一个字到I/O设备
```

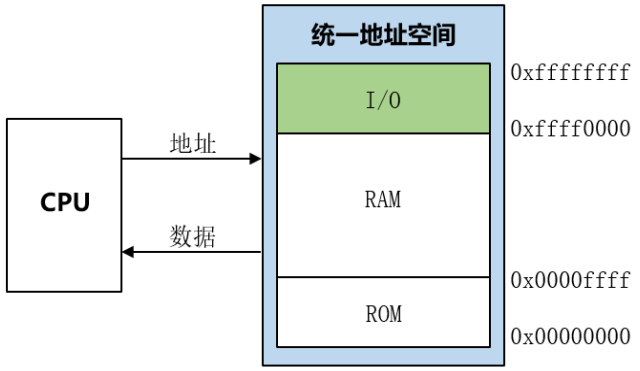


图9.2 I/O设备统一编址地址空间

– 2、独立编址

- 也称端口映射编址，Port-mapped，I/O地址空间与主存地址空间相互独立；独立编址需要使用特殊的I/O指令访问外部设备。
- 例如：80386处理器的主存空间为00000000H～FFFFFFFFH，4GB；而I/O地址空间是0000H～FFFFH，共64KB；用MOV指令访问内存，用IN/OUT指令访问外部设备。
- 表9.1为个人计算机中常见的I/O端口；图9.3为Windows系统（设备管理器）中的I/O编址。

表9.1　个人计算机中常见I/O端口

I/O地址范围（十六进制）	IRQ中断号	设备
0000～000F	00000004	DMA控制器
0020～0021	无	中断控制器
0040～0043	00000004	计时器
0060～0064		PS/2鼠标、键盘
0070～0071	00000008	CMOS实时时钟
02FB～02FF	00000004	串口
F060～F07F	FFFFFFFE	SATA控制器
03D0～03DF	FFFFFFFB	显卡控制器

图9.3　Windows系统中的I/O编址

独立编址CPU访问存储器和I/O的代码（x86汇编语言）

```
OUT DX, AL          #I/O写（输出）
IN AL, DX           #I/O读（输入）
MOV [BX], AL        #存储器写
```

• 9.2.4 I/O接口的软件

– 现代计算机中，用户并不能直接访问设备，必须通过操作系统间接访问设备。

– 操作系统中的I/O软件主要包括3个层次（图9.4）：

① 与操作系统无关的I/O库：如C语言中的标准I/O库stdio.h，包括printf()、scanf()、getchar()、putchar()、fopen()、fseek()、fread()、fwrite()、fclose()等函数；用户程序通过调用I/O库中的函数来访问设备，这些函数与操作系统无关，I/O库通常工作在用户态下。

② 与设备无关的操作系统调用库：如UNIX操作系统中的open()、read()、write()、seek()、ioctl()、close()等函数；这些函数与设备无关，屏蔽了设备的具体访问细节，向用户提供了统一的I/O调用接口；系统调用工作在内核态。

③ 独立的设备驱动程序：设备驱动程序通过具体的I/O指令或访存指令，访问I/O接口中的数据缓冲寄存器DBR（Data Buffer Register）、命令寄存器DCR（Device Command Register）、状态寄存器DSR（Device Status Register），与具体设备进行数据和命令交互；设备驱动工作在内核态。

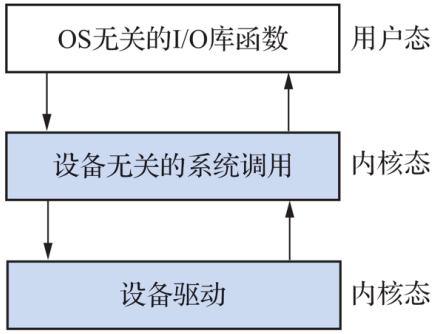


图9.4 操作系统中的I/O软件层次

• 9.2.5 I/O接口的分类

① 按数据传送方式分为：并行接口、串行接口；SCSI、IDE等属于并行接口，SAS、SATA、USB等属于串行接口。

② 按接口的灵活性分为：可编程接口、不可编程接口。

③ 按通用性分为：通用接口、专用接口；USB属于通用接口，SATA属于专用接口。

④ 按总线传输的通信方式分为：同步接口、异步接口。

⑤ 按访问外部设备的方式分为：直接传送方式接口、程序控制方式接口、程序中断方式接口、DMA接口、通道处理机接口。

SCSI：Small Computer System Interface，小型计算机系统接口

IDE：Integrated Drive Electronics，电子集成驱动器

SAS：Serial Attached SCSI，串行SCSI

SATA：Serial Advanced Technology Attachment，串行ATA（高技术配置）

USB：Universal Serial Bus，通用串行总线

9.3 数据传输控制方式

- 1、程序控制方式
  - 程序控制方式：首先，通过设置I/O接口的命令寄存器启动设备；设备准备的过程中，CPU通过读取I/O接口中的状态寄存器，查询设备是否已就绪，根据查询结果决定下一步操作究竟是进行数据传送，还是等待。
  - 程序控制方式也称为程序查询方式。
  - 程序控制方式的I/O接口设计简单，但是CPU与外部设备只能串行工作，CPU浪费大量时间进行查询和等待，系统效率较低。
- 2、程序中断控制方式
  - 程序中断控制方式：CPU启动外部设备后，不再查询外部设备的状态，当外部设备准备好后，主动向CPU发出中断请求；CPU响应中断请求，暂停正在执行的程序，并调用相应的中断服务程序，完成CPU与外部设备的一次信息传输。
  - 程序中断控制方式中，CPU与外部设备是并行工作，CPU利用率高。

- **3、直接存储器访问方式**
  - 直接存储器访问方式：DMA方式，Direct Memory Access。
  - DMA方式中，由DMA控制器（DMAC，DMA Controller）临时代替CPU控制总线，控制外部设备与内存之间进行直接的数据交换，信息传送不再经过CPU寄存器中转；DMA方式主要用于存储器与外部设备（如磁盘）之间的大量数据传送。
- **4、通道方式**
  - 为进一步减少CPU被I/O操作中断的次数，提高CPU效率，出现了通道技术（通道方法）；由通道分担CPU的I/O管理，能有效提高系统效率。
  - 通道拥有独立的通道指令系统，可以通过执行通道程序来完成CPU指定的I/O任务。
- **5、外围处理器方式**
  - 外围处理机方式（PPU，Peripheral Processor Unit）是通道方式的进一步发展，通常用于大中型计算机系统中；各种I/O控制方式如图9.5所示。

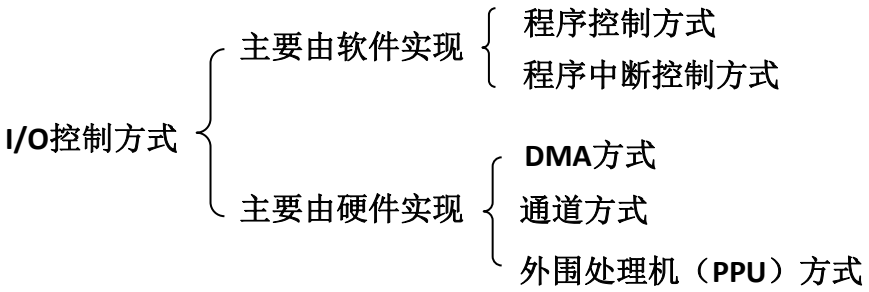


图9.5　I/O控制方式

9.4 程序控制方式

• 9.4.1 简单设备程序查询流程

– 程序控制方式（Programed I/O）是最原始的、最简单的方式。

– 程序控制方式又分为程序查询方式（也称为轮询方式，Polling）和直接传送方式两种；程序查询方式属于有条件传送方式，直接传送方式属于无条件传送方式。

– 表9.2为键盘和字符显示等简单设备的I/O地址。

表9.2 设备I/O地址

设备	寄存器（8位）	内存映射地址	端口映射地址	备注
键盘设备	数据缓冲寄存器DBR	0xFFFF0004	0x0004	
键盘设备	设备状态寄存器DSR	0xFFFF0000	0x0000	最低位为Ready位
字符显示设备	数据缓冲寄存器DBR	0xFFFF000C	0x000C	
字符显示设备	设备状态寄存器DSR	0xFFFF0008	0x0008	最低位为Ready位

– 键盘设备的程序查询流程如图9.6a所示，其MIPS和x86查询程序代码如下：

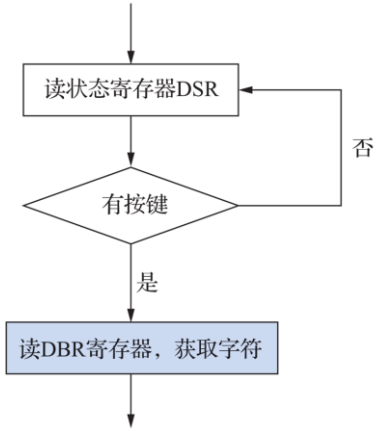


图9.6a　键盘程序查询流程

MIPS代码

```
keyPoll:  lui  $t0,0xffff0000   #载入键盘DSR内存映射地址至$t0
          lbu  $t2,($t0)        #载入键盘DSR的值至$t2
          andi $t2,$t2,1        #获取最低位Ready位
          beq $t2,$0,keyPoll    #如果Ready==0，则继续查询
          lui $t0, 0xffff0004   #载入键盘DBR内存映射地址至$t0
          lbu $s0,($t0)         #载入键盘DBR中的字符数据至$s0
```

x86代码

```
keyPoll:  MOV DX,0x0000         #载入键盘DSR端口地址至DX
          IN AL,DX              #载入键盘DSR的值至AL
          TEST AL,1             #测试就绪位Ready
          JZ keyPoll            #如果Ready==0，则继续查询
          MOV DX,0x0004         #载入键盘DBR端口地址至DX
          IN AL,X               #载入键盘DBR中的字符数据至AL
```

– 字符终端（显示）设备的程序查询流程如图9.6b所示，其MIPS和x86查询程序代码如下：

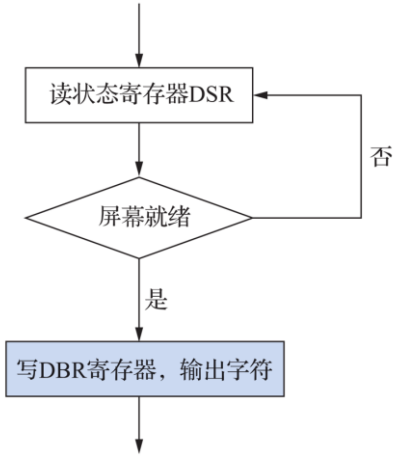


图9.6b　字符终端程序查询流程

MIPS代码

```
ttyPoll:  lui  $t0,0xffff0000   #载入字符终端DSR内存映射地址至$t0
          lbu  $t2,($t0)        #载入字符终端DSR的值至$t2
          andi $t2,$t2,1        #获取最低位Ready位
          beq $t2,$0,ttyPoll    #如果Ready==0，则继续查询
          lui $t0, 0xffff0004   #载入字符终端DBR内存映射地址至$t0
          sb $s0,($t0)          #将$s0中的字符数据输出至DBR
```

x86代码

```
ttyPoll:  MOV DX,0x0000         #载入字符终端DSR端口地址至DX
          IN AL,DX              #载入字符终端DSR的值至AL
          TEST AL,1             #测试就绪位Ready
          JZ ttyPoll            #如果Ready==0，则继续查询
          MOV DX,0x0004         #载入字符终端DBR端口地址至DX
          OUT DX,AL             #将AL中的字符数据输出至DBR
```

• 9.4.2 复杂设备程序查询流程

– 图9.7为复杂设备的程序查询流程。

– CPU首先查询设备的状态（①），如果设备就绪（②），就通过总线向I/O接口发送命令与参数，启动设备（③）。

– 设备收到命令后，即刻去准备或处理；设备准备好后，会将状态寄存器中的相关位置位，表示命令执行完毕或准备就绪。

– 而CPU在启动设备后，就开始不断查询设备状态（④），当设备就绪（⑤），即可进行实际的数据传输（⑥）。

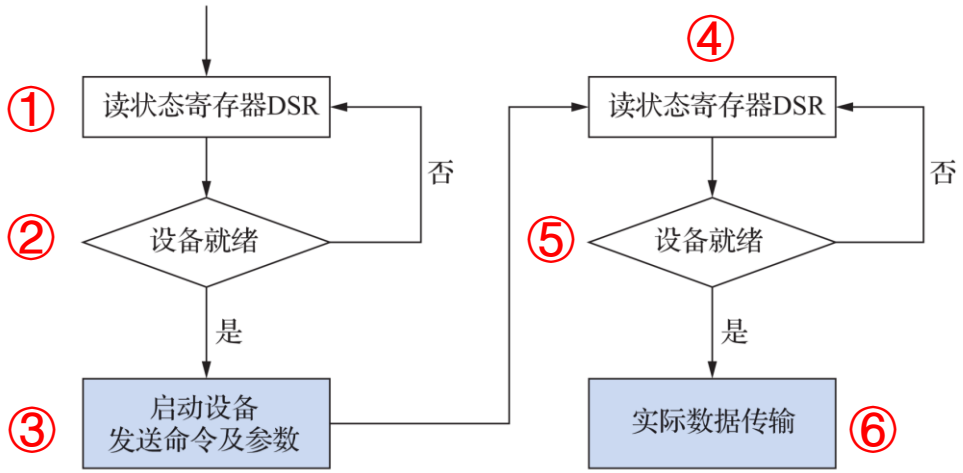


图9.7　复杂设备的程序查询流程

• 9.4.3 程序查询特点

– 程序查询方式中，CPU要不断查询I/O接口中的状态寄存器DSR（Device Status Register），当设备就绪时才能进入下一操作，否则继续查询。

– 程序查询（也称为轮询）有两种策略：忙等待和定时轮询。

– 1、忙等待（Busy-waiting）

- 忙等待方式也称为独占式查询，程序运行轨迹如图9.8a所示；在设备准备数据阶段，CPU不能执行其他任务，称为忙等待状态（轮询等待，busy-waiting）。

- 忙等待方式的缺点：CPU浪费了大量的时间进行轮询操作。

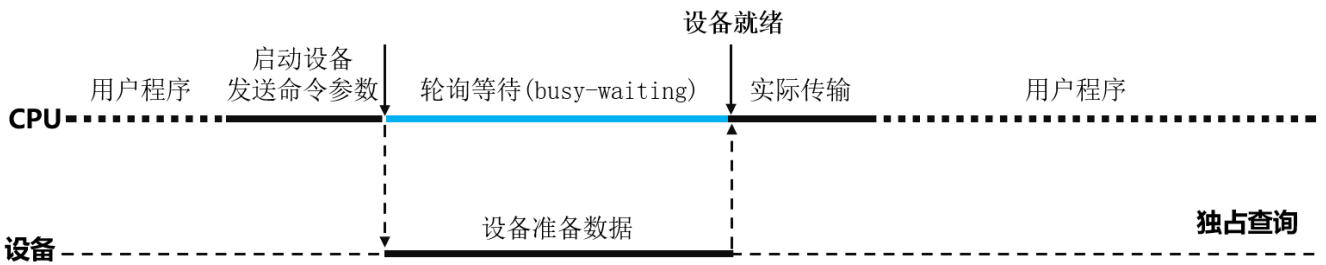


图9.8a　独占式查询方式的程序运行轨迹

– 2、定时轮询（Polling）

- 定时轮询不需要反复查询，程序运行轨迹如图9.8b。

- CPU启动设备后，会启动一个定时中断（①）；然后挂起当前用户进程P1，并放入I/O等待队列，调度用户进程P2运行（②）；定时时间到后（③），CPU会执行定时中断服务程序，查询设备状态，如果设备准备好，则唤醒等待进程P1（④），否则将继续定时查询。

- 定时轮询方式的优点：CPU可以执行其他任务，避免轮询等待CPU时间的浪费，有效节约了CPU时间。

- 定时轮询中的定时时间间隔比较关键；时间间隔过短，则用于定时查询的中断服务开销浪费较多；时间间隔过长，外部设备数据可能得不到及时处理。

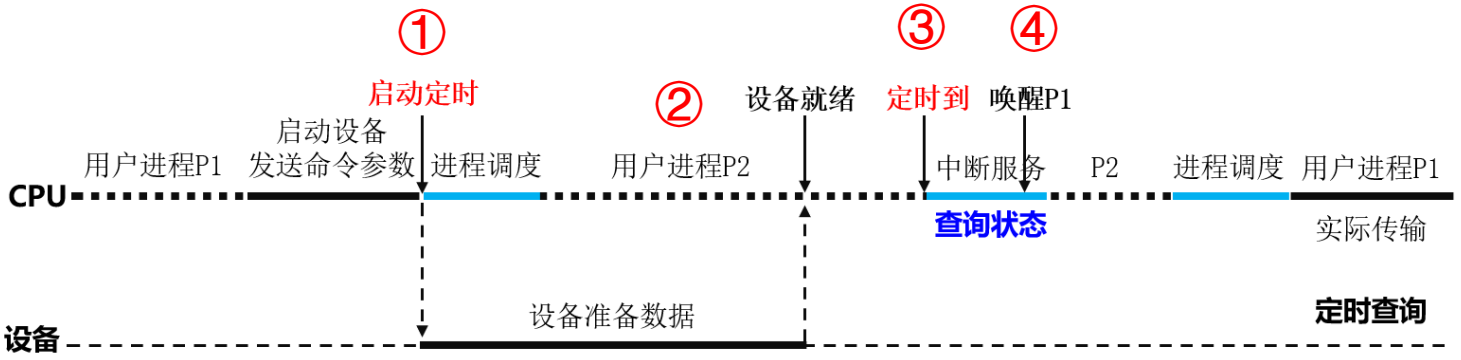


图9.8b　定时轮询方式的程序运行轨迹

- 例9.1：假设某程序查询方式的输入输出系统采用定时轮询方式，每次定时中断服务开销需要400个时钟周期，CPU的时钟频率为200MHz，包括鼠标和硬盘两个外部设备，求两种不同外部设备进行I/O操作时的CPU时间占用率。
- （1）鼠标以字节为单位进行数据传输，假设每秒必须进行50次轮询才能保证不会错过任何鼠标操作，每次轮询成功后的实际数据传输需要13个时钟周期。
- （2）硬盘以512字节的扇区为单位传输数据，启动阶段发送命令和参数需要90个时钟周期，实际传输阶段需要1555个时钟周期，CPU访问硬盘的速率为20MB/s。
- （3）如果硬盘以字节为单位进行数据传输，其他参数不变，会发生什么情况？

- 解：
- （1）
  - 时钟周期T = 1/200MHz
  - CPU每秒用于鼠标I/O操作的时间 = (400+13) x 50 x T = 20650T = 0.00010325s
  - CPU时间占用率= 0.00010325s / 1s = 0.010325 %
  - 结论：对鼠标进行定时轮询操作基本不影响CPU的性能。

- 例9.1：假设某程序查询方式的输入输出系统采用定时轮询方式，每次定时中断服务开销需要400个时钟周期，CPU的时钟频率为200MHz，包括鼠标和硬盘两个外部设备，求两种不同外部设备进行I/O操作时的CPU时间占用率。
- （1）鼠标以字节为单位进行数据传输，假设每秒必须进行50次轮询才能保证不会错过如何鼠标操作，每次轮询成功后的实际数据传输需要13个时钟周期。
- （2）硬盘以512字节的扇区为单位传输数据，启动阶段发送命令和参数需要90个时钟周期，实际传输阶段需要1555个时钟周期，CPU访问硬盘的速率为20MB/s。
- （3）如果硬盘以字节为单位进行数据传输，其他参数不变，会发生什么情况？

- 解：
- （2）
  - 硬盘每次传输512个字节，要达到CPU访问硬盘的速率20MB/s，则每秒传输的次数 = 20MB/512B = $20x10^6/512$ = 39062.5次/s
  - 每次传输的开销（每次传输512个字节）= (90+400+1555) x T = 2045 x (1/200MHz) = 0.000010225s
  - CPU时间占用率= 0.000010225s x 39062.5 / 1s = 39.94%
  - 结论：采用定时轮询方式进行硬盘的数据传输，会影响CPU的性能。

- 例9.1：假设某程序查询方式的输入输出系统采用定时轮询方式，每次定时中断服务开销需要400个时钟周期，CPU的时钟频率为200MHz，包括鼠标和硬盘两个外部设备，求两种不同外部设备进行I/O操作时的CPU时间占用率。
- （1）鼠标以字节为单位进行数据传输，假设每秒必须进行50次轮询才能保证不会错过如何鼠标操作，每次轮询成功后的实际数据传输需要13个时钟周期。
- （2）硬盘以512字节的扇区为单位传输数据，启动阶段发送命令和参数需要90个时钟周期，实际传输阶段需要1555个时钟周期，CPU访问硬盘的速率为20MB/s。
- （3）如果硬盘以字节为单位进行数据传输，其他参数不变，会发生什么情况？

- 解：
- （3）
  - 如果硬盘以字节为单位进行数据传输，假设从接口读出一个字节转存至内存的开销是15个时钟周期，则512个字节的传输开销 = (90+400+15) x 512 x T = 258560T = 0.0012928s
  - 硬盘每次传输512个字节，要达到CPU访问硬盘的速率20MB/s，则每秒传输的次数 =20MB/512B=20x10^6/512= 39062.5次/s
  - CPU时间占用率 = 0.0012928s x 39062.5 / 1s = 5005%
  - 结论：如果硬盘以字节为单位进行数据传输，即使CPU所有的时间都花在硬盘上，也达不到20MB/s的速率；最高速率 = 512B/0.0012928s = 396039.6B/s = 0.396MB/s

6.7 异常与中断处理

• 6.7.1 异常与中断的基本概念

– 异常（Exception）：通常是指CPU内部引起的异常事件，也称为内部中断、软件中断；异常可以进一步分为：故障、自陷、终止。

① 故障（Fault）：通常是由指令执行引起的异常，如未定义指令、越权指令、段故障、缺页故障、存储保护违例、数据未对齐、除数为零、浮点溢出、整数溢出等；故障又可分为：可恢复故障（如数据缺页）和不可恢复故障（如未定义指令、越权指令等）。

② 自陷（Trap）：是一种事先安排的“异常”事件；如系统调用指令、条件陷阱指令；例如x86的INT 21H指令，MIPS的syscall指令。

③ 终止（Abort）：是指随机出现的使得CPU无法继续执行的硬件故障，和具体指令无关；如机器校验错、总线错误、异常处理中再次异常的双错等。

– 中断（外部中断，Interrupt）：是指由外部设备向CPU发出的中断请求（如鼠标点击、按键动作等），要求CPU暂停当前正在执行的程序，转去执行为某个外部设备事件服务的中断服务程序，处理完毕后，再返回断点继续执行。

– MIPS处理器将异常和中断都称为“异常”。

– x86处理器将异常和中断都称为“中断”，来自CPU内部的中断称为“内部中断（软件中断）”，来自CPU外部的中断称为“外部中断（硬件中断）”。

• 6.7.2 异常与中断处理过程

- 当发生中断（或异常）事件时，CPU接收到中断请求，在指令执行结束时CPU要进入中断响应周期进行响应处理（例外情况：例如产生故障异常的指令并没有执行完毕，但必须立即进入中断响应）。

- 中断响应周期的主要任务是：关中断、保存断点、中断识别。

- （1）关中断：关中断的目的是临时禁止新的中断请求，是为了在中断响应周期以及中断服务程序中保护现场操作的完整性，只有这样才能保证中断服务程序执行完成后，能返回断点正确执行。

- （2）保存断点：保存断点就是保存将来返回被中断程序的位置；对于已经执行完毕的指令，断点是下一条指令的位置；对于缺页故障、段错等执行指令引起的故障异常，由于指令并没有执行，断点就是异常指令的PC值。

- （2）中断识别：中断识别的主要任务就是根据当前的中断请求识别中断来源，将中断服务程序入口地址送入程序计数器PC。

- 中断响应周期内的操作是由硬件实现的，整个中断响应周期是不可被打断的；中断响应周期结束后，CPU将执行中断服务程序，直至中断返回，这部分由软件实现（中断服务程序）；因此，整个中断处理过程是软硬件协同实现的。

• 6.7.3 支持中断的CPU设计

– 1、数据通路升级

- 图6.64为支持中断的单总线结构的计算机框图（图中没有画出ALU和寄存器堆）；与图6.4相比，增加了：

① EPC：Exception Program Counter，异常程序计数器，用于保存断点；有2个控制信号：EPC_{in}、EPC_{out}。

② IE：Interrupt Enable，中断使能位（中断使能触发器/寄存器），有2个控制信号：开中断STI、关中断CLI。

③ 中断控制（中断控制逻辑）：用于进行中断优先级排队；假设有4个外部中断请求信号A、B、C、D，中断控制逻辑对4个中断请求进行优先级排队，输出优先级最高的中断号，并发出中断请求信号，控制信号ClrInt用于清除当前正在响应的中断请求。

④ 多路选择器MUX：根据中断号，将对应的中断服务程序入口地址（A中断入口、B中断入口、C中断入口、D中断入口）通过内总线送程序计数器PC，控制信号为$IntA_{out}$。

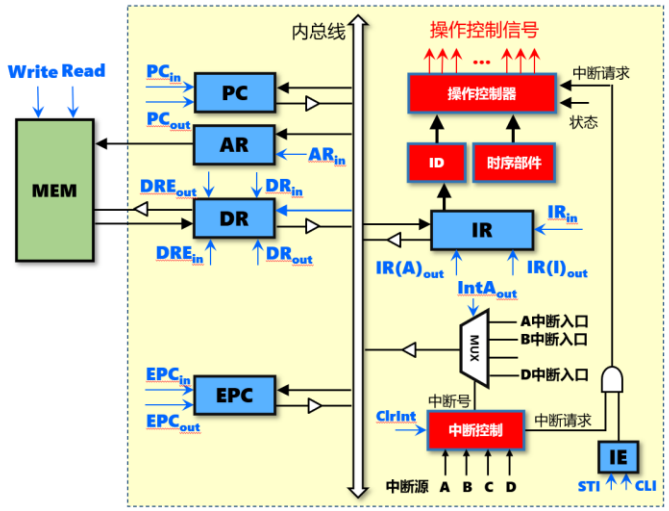


图6.64 支持中断的单总线结构计算机框图

表6.18 中断控制信号及其作用

序号	控制信号	作用说明
1	EPC_{in}	控制EPC接收来自内总线的数据，需配合时钟控制
2	EPC_{out}	控制EPC向内总线输出数据
3	STI	开中断，将中断使能寄存器置1，操作控制器可以接收中断请求
4	CLI	关中断，将中断使能寄存器置0，操作控制器可以接收不到任何中断请求
5	$IntA_{out}$	将中断服务程序入口地址输出到内总线
6	CtrlInt	清除当前正在响应的中断请求

– 2、操作控制器升级

• （1）三级时序硬布线控制器升级

– 对图6.41的单总线结构计算机的三级时序状态机（变长指令周期）进行升级，使其能支持中断，升级后的状态机如图6.65所示。

– 图6.65中增加了1个中断响应周期M_{int}，包含2个状态S_9和S_{10}，如表6.19所示：

① 状态S_9（T1）：进行关中断和保存断点的操作，将PC值送入EPC寄存器中保存，发出控制信号：CLI=1、PC_{out}=1、EPC_{in}=1。

② 状态S_{10}（T2）：完成中断识别的操作，将要响应的中断服务程序入口地址送PC，发出控制信号：$IntA_{out}$=1、PC_{in}=1 。

– 此外，还有增加1条中断返回指令eret；该指令的主要功能是将EPC中保存的断点送回PC，同时开中断；该指令执行周期的操作及控制信号如表6.20所示。

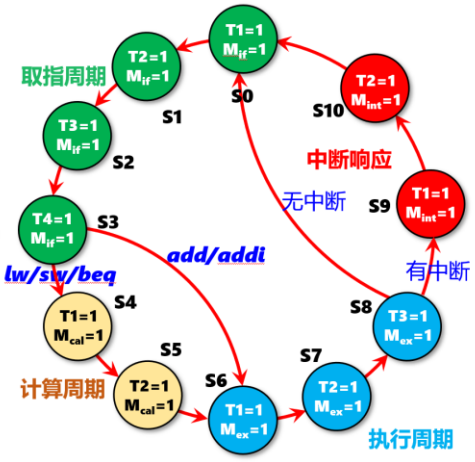


图6.65　支持中断的三级时序状态机（变长指令周期）

表6.19　中断响应周期的操作及控制信号

周期	节拍	操作	功能说明	控制信号
中断 M_{int}	T1	0->IE　PC->EPC	关中断，同时将PC值送入EPC寄存器中保存	$CLI=PC_{out}=EPC_{in}=1$
	T2	中断程序入口->PC	将要响应的中断对应的中断程序入口地址送PC	$IntA_{out}=PC_{in}=1$

表6.20　eret指令执行周期的操作及控制信号

周期	节拍	操作	功能说明	控制信号
中断 M_{int}	T3	EPC->PC　1->IE	恢复断点，将EPC送PC，同时开中断	$EPC_{out}=PC_{in}$=STI=ClrInt=1

- （2）现代时序硬布线控制器升级
  - 同样可以对图6.45的单总线结构计算机的现代时序状态机进行升级，使其能支持中断，升级后的状态机如图6.66所示。
  - 图6.66中增加了2个状态S_{26}和S_{27}，用于中断响应。
  - 此外，也要增加1条中断返回指令eret，该指令的执行只需要1个时钟周期S_{25}。

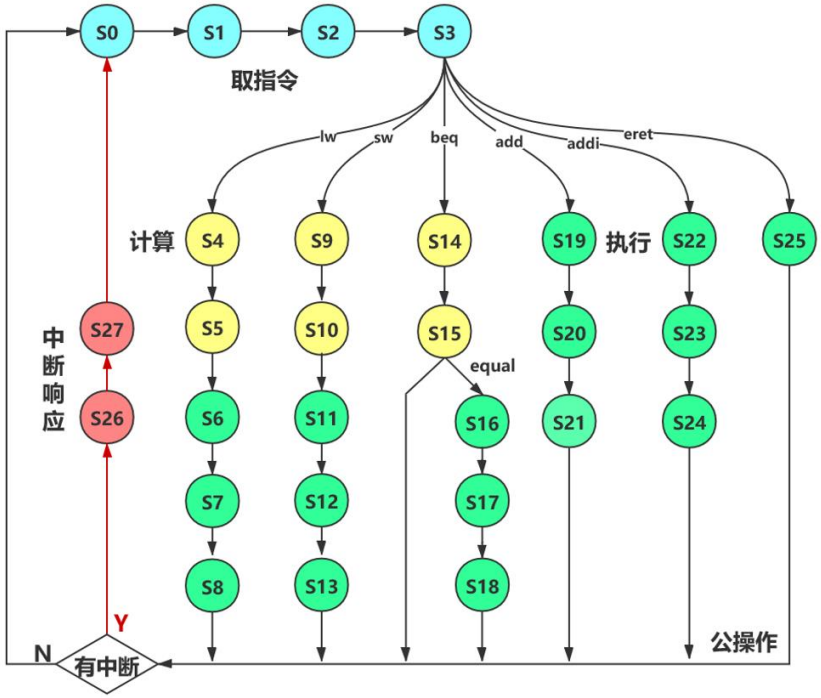


图6.66　支持中断的现代时序状态机

- （3）微程序控制器升级
  - 可以用微程序设计方法实现图6.66的支持中断的现代时序状态机：
  - 首先，要修改微指令格式，增加6个中断控制信号（表6.18）；图6.67为支持中断的微指令格式（计数器法）。
  - 其次，要增加2条中断响应微指令（对应图6.66中的S_{26}和S_{27}）；增加1条中断返回指令eret的微指令（对应图6.66中的S_{25}）。
  - 此外，指令执行完毕要进行中断判断，不管是下址字段法还是计数器法，都要有P_{end}测试位，并根据中断测试条件进行分支（如果有中断，则转中期响应周期；如果没有中断则转取指微程序入口）。
  - 图6.68为支持中断的微程序控制器原理图（采用下址字段法）：
    ① 第1种情况：P_{end}=1、中断请求IntR=1、P_{equal}&equal=0；进入中断响应周期
    ② 第2种情况：P_{end}=1、IntR=0；进入S0
    ③ 第3种情况：beq指令，P_{end}=1、IntR=0、P_{equal}&equal=0；进入S0
    ④ 第4种情况：beq指令，P_{end}=1、IntR=0、P_{equal}&equal=1；进入S16

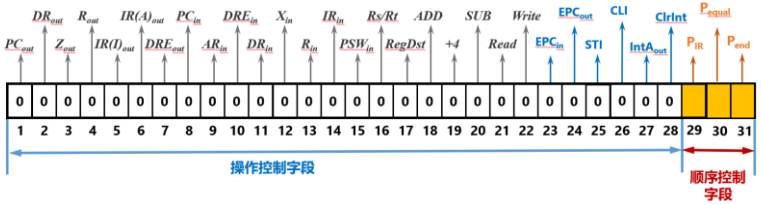


图6.67　支持中断的微指令格式（计数器法）

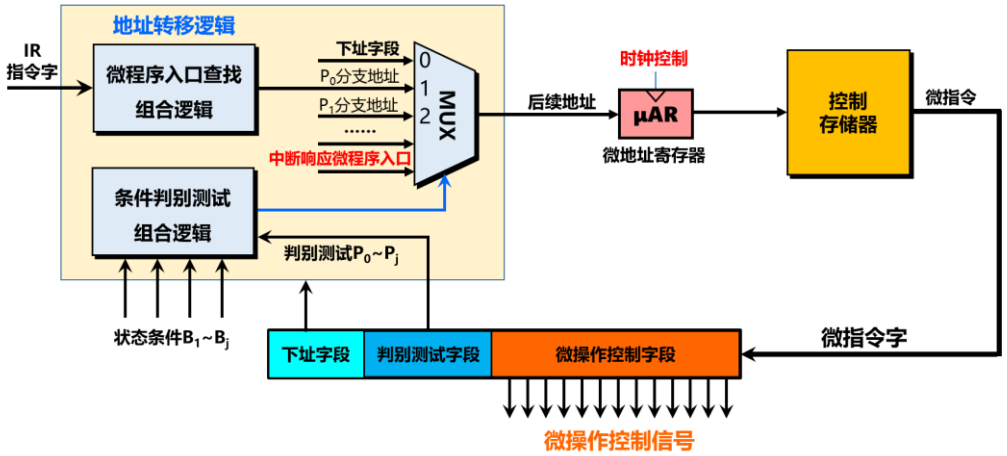


图6.68　支持中断的微程序控制器原理图（下址字段法）

– 3、中断服务程序

- 中断机制是软硬件协同的机制，中断响应周期内的操作（关中断、保存断点、中断识别）都是由硬件实现的，中断服务程序则是由软件实现的。

- 中断服务程序主要包括4个步骤：

① 保护现场：将中断服务程序中会被改写的寄存器，通过压栈的方式保存到内存堆栈中，以保证被中断的程序在执行完中断服务程序后，还能正确执行。

```
» PUSH AX
» PUSH BX
» PUSH CX
```

② 中断服务：完成中断处理，这是中断服务程序的核心内容。

③ 恢复现场：通过出栈的方式，将保存在内存堆栈中的内容送回寄存器。

```
» POP CX
» POP BX
» POP AX
```

④ 中断返回：执行中断返回指令eret，返回到断点处。

9.5 程序中断控制方式

- 为了解决程序查询方式CPU反复不停地查询设备状态、浪费大量CPU时间的缺点，产生了程序中断控制（Interrupt-driven I/O）的输入输出方式；程序中断控制方式的执行流程如图9.9所示。
  - （1）用户系统调用：用户进程P1通过系统调用read()函数使操作系统进入内核态（①）。
  - （2）驱动启动设备：设备驱动程序发送命令参数，启动设备（②）。
  - （3）操作系统进程调度：操作系统将用户进程P1放入I/O等待队列，调度用户进程P2运行（③）。
  - （4）设备中断请求：当设备准备好数据后，立即主动向CPU发出中断请求（④）。
  - （5）CPU中断服务：CPU暂停当前进程的执行，转去执行中断服务程序，唤醒用户进程P1，完成实际的数据传输（⑤）。
  - （6）CPU恢复运行：中断服务程序执行完毕后，返回到断点继续执行用户进程P2（⑥）。

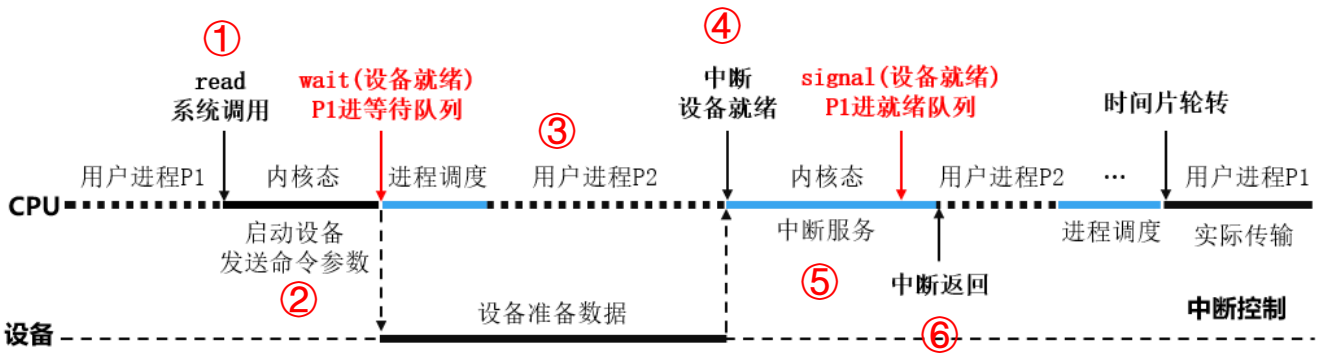


表9.9 中断控制方式的程序运行轨迹

- 中断控制方式也是有时间开销的，相对于程序查询方式，它的额外开销是用于进程调度的两次上下文切换时间，以及中断服务程序本身的开销。

- 对于极高速设备，设备准备数据的时间很短（比两次上下文切换以及中断服务程序的开销还短），使用中断控制方式的效率反而比忙等待轮询方式的效率更低，中断就会成为瓶颈。

- 如最新的NVMe SSD硬盘在处理同步I/O时，就不采用中断控制方式，而是采用程序查询方式进行数据交互。

- 例9.2：某外部设备传送信息的最高频率为$40x10^3$次/s，而相应的中断处理程序的执行时间为40μs，问：该外部设备是否可以采用中断控制方式工作？为什么？

- 解：
  - 该外部设备传送一次数据的时间为：$1/(40x10^3)=25\mu s$ 。
  - 中断处理程序的执行时间为40μs。
  - 因此该外部设备不能采用中断控制方式工作，可以考虑使用程序查询方式。

• 9.5.1 中断的基本概念

– 中断（Interrupt）是指由外部设备向CPU发出的中断请求（如鼠标点击、按键动作等），要求CPU暂停当前正在执行的程序，转去执行为某个外部设备事件服务的中断服务程序，处理完毕后，再返回断点继续执行。

– 中断技术将有序的程序运行和无序的随机中断事件统一起来，大大增强了系统的处理能力和灵活性。

– 1、中断的作用

① 提升并行性：可以实现CPU和外部设备的并行工作。

② 程序调试：方便在程序中设置断点来观察程序执行的中间结果。

③ 故障处理：方便及时处理各种随机出现的软、硬件故障和异常。

④ 实时处理：中断技术能确保实时数据被及时处理。

⑤ 人机交互：键盘、鼠标等都是通过中断控制方式实现人机对话的。

⑥ 实现多任务：进程时间片轮转必须借助定时中断技术实现。

⑦ 多处理器交互：可以通过中断控制方式实现多处理器之间的信息交换和任务切换。

– 2、中断的基本类型

- （1）内部异常和外部中断
  - 根据中断来源的不同，可以分为内部异常和外部中断（图9.10），具体见6.7.1小节。

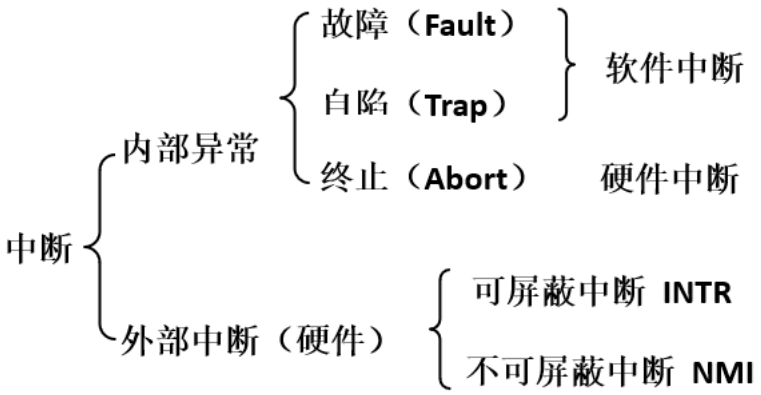


图9.10　根据中断来源的分类

- （2）硬件中断和软件中断
  - 外部中断和内部异常的终止都属于硬件中断，内部异常的故障和自陷属于软件中断。

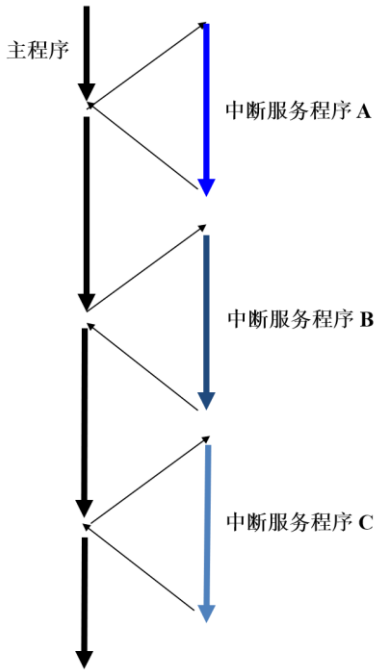
- （3）自愿中断和强迫中断
  - 自愿中断是在程序中事先安排好的，如内部异常的自陷、x86的int n指令、MIPS的syscall指令（系统调用指令）等；强迫中断是随机产生的。

- （4）可屏蔽中断和不可屏蔽中断
  - 可屏蔽中断的请求信号INTR将与中断允许位IE/IF进行逻辑与后，再送到CPU，CPU的关中断指令使IE/IF=0，从而使INTR无法送到CPU（屏蔽掉），其中断优先级最低。
  - 不可屏蔽中断的请求信号NMI直接送到CPU，无法屏蔽，其中断优先级最高，常用于定时器中断、断电中断等情况。

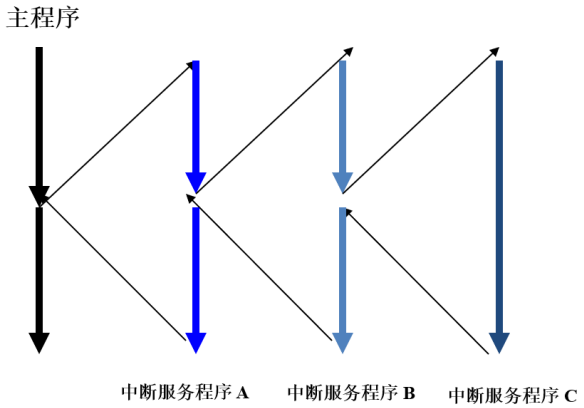
- （5）向量中断和非向量中断
  - 向量中断的中断服务程序入口地址（也称为向量地址）是由硬件提供的；非向量中断的中断服务程序入口地址不能直接由硬件得到。

- （6）单级中断和多重中断

  – 单级中断执行中断服务程序时，不再响应其他中断请求；多重中断（也称为嵌套中断、中断嵌套）的中断服务程序可以被更高优先级的中断请求中断；图9.11，A优先级最低、C优先级最高。



（a）单级中断



（b）多重中断

图9.11　单级中断和多重中断的处理示意图

– 3、中断优先级与中断屏蔽

- 中断优先级就是指CPU响应并处理中断的先后次序；中断优先级包含两层含义：响应优先级和处理优先级。

- 响应优先级是指CPU对各设备中断请求进行响应的先后次序，其在硬件电路上是固定的，不便于变动；中断响应优先级的一般划分规律如下：

  ① 不可屏蔽中断 > 内部异常 > 可屏蔽中断

  ② 内部异常中：硬件终止（Abort）> 指令异常或自陷等程序故障（Fault、Trap）

  ③ DMA中断 > I/O中断

  ④ I/O中断中：高速设备 > 低速设备，输入设备 > 输出设备，实时控制 > 普通设备

- 处理优先级是指中断嵌套的实际优先级处理次序，通常可以利用中断屏蔽技术动态调整，从而使低优先级的中断也可以中断高优先级的中断服务程序，使中断处理更加灵活。

- 如果不使用中断屏蔽技术，处理优先级与响应优先级相同。

- 中断屏蔽技术：在中断控制器中设置中断屏蔽寄存器（Interrupt Mask Register，IMR），IMR的值称为中断屏蔽字，为“1”时表示屏蔽，为“0”时表示允许（不屏蔽）。

- 中断屏蔽字（图9.12）：8个设备，其中4个被屏蔽（A、C、D、G）。

- 中断屏蔽技术原理（图9.13）：
  - 假设4个设备的中断优先级是：IR0最高、IR3最低。
  - 如果没有中断屏蔽技术，3个设备（IR1、IR2、IR3）同时发出中断请求，将先处理IR1的中断请求。
  - 如果采用中断屏蔽技术，中断屏蔽寄存器IMR设置为“0 1 0 0”，即屏蔽IR1；3个设备（IR1、IR2、IR3）同时发出中断请求，将先处理IR2的中断请求。

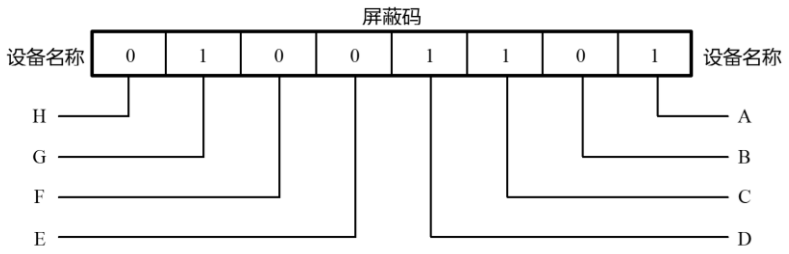


图9.12　中断屏蔽字

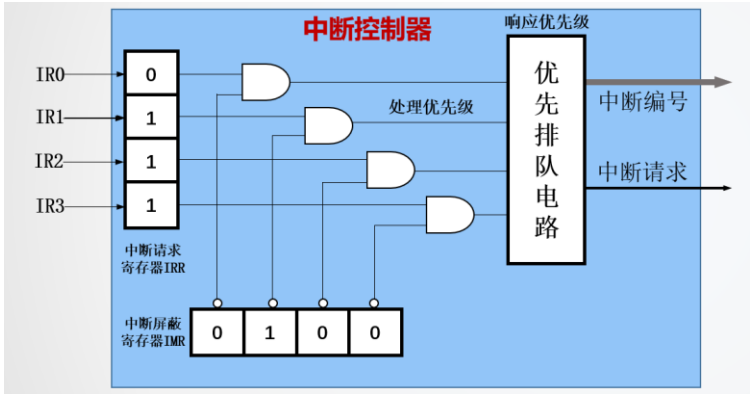


图9.13　中断屏蔽技术原理

- 例9.3：某计算机系统有A、B、C三个外部中断源，中断优先级排队电路中的响应优先级次序为：A > B > C，支持多重嵌套中断。假设CPU在执行主程序时A、B、C三个设备的中断请求在同一时刻产生，请分别给出表9.3所示的左右两种屏蔽字的情况下，CPU运行的轨迹图。

表9.3　各设备中断屏蔽字

设备名	原始屏蔽字		
	A	B	C
A	1	1	1
B	0	1	1
C	0	0	1

设备名	修改后的屏蔽字		
	A	B	C
A	1	0	0
B	1	1	0
C	1	1	1

- 解：
- （1）表9.3左边的屏蔽字（原始屏蔽字）：其处理优先级与响应优先级相同，即：A > B > C；相当于没有屏蔽；其CPU运行轨迹如图9.14a所示。

- 图中的两个箭头表示：CPU执行完中断服务程序后，返回到主程序执行完一条指令后，才能转去执行另一个中断服务程序。

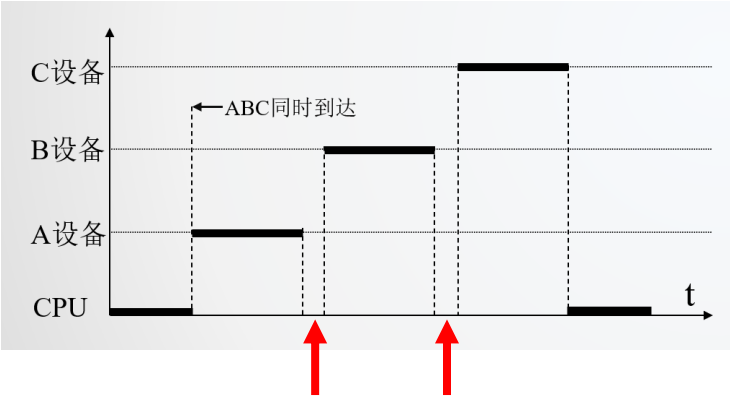


图9.14a　无屏蔽时CPU运行轨迹

- 例9.3：某计算机系统有A、B、C三个外部中断源，中断优先级排队电路中的响应优先级次序为：A > B > C，支持多重嵌套中断。假设CPU在执行主程序时A、B、C三个设备的中断请求在同一时刻产生，请分别给出表9.3所示的左右两种屏蔽字的情况下，CPU运行的轨迹图。

表9.3　各设备中断屏蔽字

设备名	原始屏蔽字		
	A	B	C
A	1	1	1
B	0	1	1
C	0	0	1

设备名	修改后的屏蔽字		
	A	B	C
A	1	0	0
B	1	1	0
C	1	1	1

- 解：
- （2）表9.3右边的屏蔽字（修改后的屏蔽字）：其处理优先级为：C > B > A；响应优先级为：A > B > C；其CPU运行轨迹如图9.14b所示。

- 图中的第一个箭头表示：CPU响应A设备的中断请求（因为A设备的响应优先级最高）；图中的第二个箭头表示：CPU响应B设备的中断请求（因为B设备的响应优先级比C设备高）。

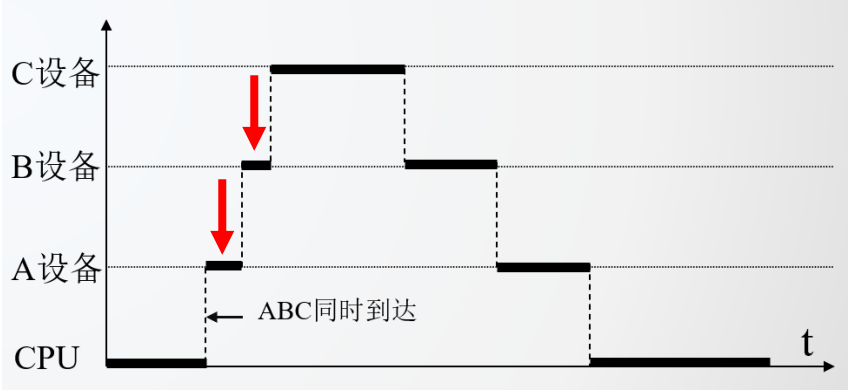


图9.14b　中断屏蔽时CPU运行轨迹

• 9.5.2 中断请求

– 1、中断请求信号的传送

- 多个中断源的中断请求信号与CPU的连接方式（传输方式）有4种（图9.15）：
- （1）独立请求方式：每个中断源有单独的中断请求线（INTR，Interrupt Request）和中断应答线（INTA，Interrupt Acknowledge）。
- （2）链式请求方式：多个中断源共用一根公共的中断请求线INTR和中断应答线INTA；又分为硬件查询法和软件查询法两种方式。
- （3）中断控制器方式：所有中断源通过独立请求方式与中断控制器连接，中断控制器与CPU连接，中断控制器分担了CPU的中断控制功能，负责实现外部中断的优先级仲裁逻辑与中断识别；如x86中的Intel 8259芯片就是中断控制器。
- （4）分组链式：是独立请求方式与链式请求方式的折中；对于要求快速响应的中断请求采用独立请求方式，对于不要求快速响应的中断请求则采用链式请求方式。

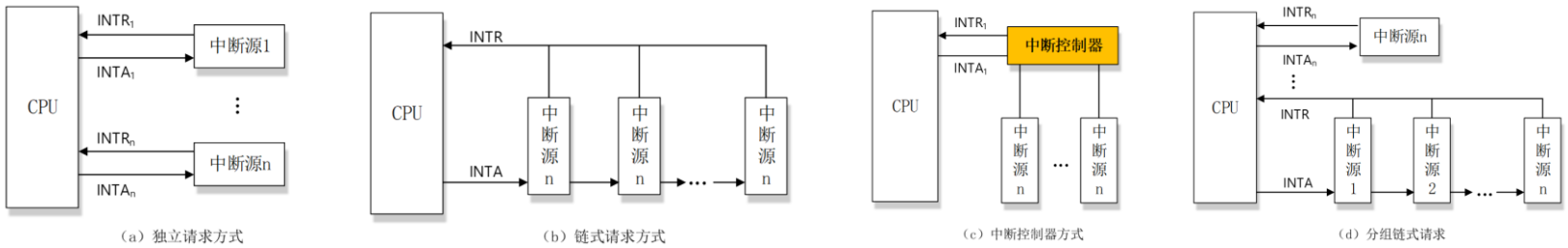


图9.15　中断请求信号的传输方式

– 2、中断请求的硬件支撑（图9.16）

- （1）中断请求寄存器：IRR，Interrupt Request Register，IRR中的内容称为中断字。
- （2）中断屏蔽寄存器：IMR，Interrupt Mask Register，“1”表示屏蔽，“0”时表示允许（不屏蔽）。
- （3）中断服务寄存器：ISR，Interrupt Service Register，在多重中断中用于存放正在被服务的中断请求。
- （4）中断优先级排队电路：PR，Priority Resolver，用于多个外部可屏蔽中断请求的优先级排队。
- （5）中断允许触发器：IE/IF，Interrupt Enable/Interrupt Flag，也称中断使能位、中断标志位，用于开、关中断的控制，“1”表示开中断，“0”表示关中断；IE/IF通常设置在CPU内部。

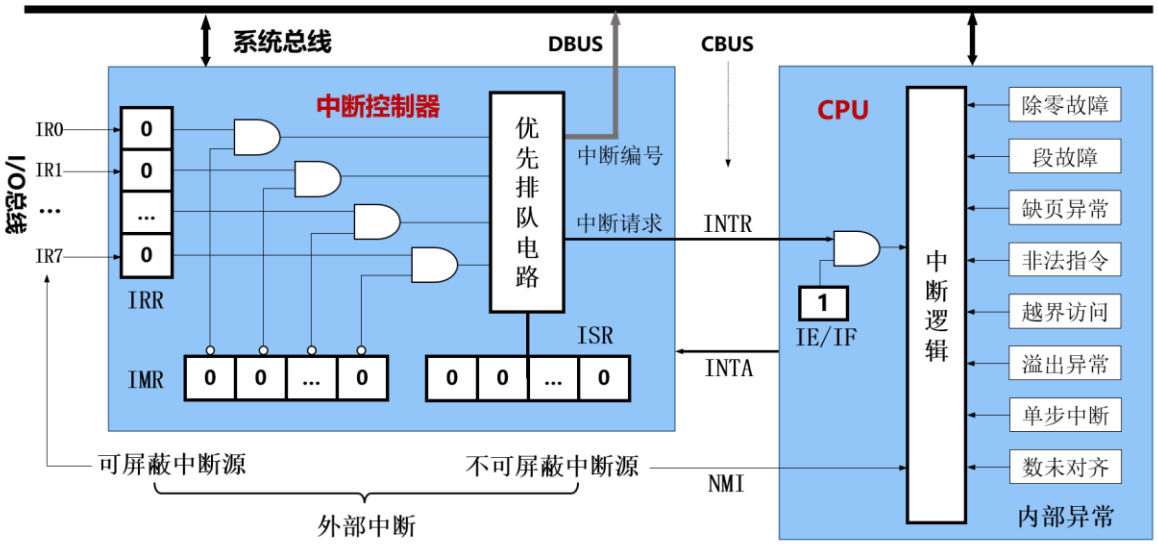


图9.16　中断请求的硬件支持

- 中断控制器的工作流程，图9.17。

① IR_i有中断请求，则使IRR[i]=1

② 如果IMR[i]=0，且i大于ISR中的最高优先级，则使INTR=1

③ 如果IE/IF=1，则CPU进入中断响应周期；否则CPU继续执行主程序（取指令、执行指令）

④ 中断响应周期：先是关中断、保存断点，然后发出2个INTA脉冲

⑤ 第1个INTA脉冲：使IRR[i]=0、IRR[i].EN=0，表示不再接收IR_i的新的中断请求；并使ISR[i]=1，表示IR_i的中断请求正在服务

⑥ 第2个INTA脉冲：请求中断控制器将中断编号通过数据总线（DBUS）发送给CPU

⑦ CPU收到中断编号（中断服务程序入口地址）后，执行中断服务程序，中断服务程序执行完毕后，发送中断服务结束命令EOI（End Of Interrupt）

⑧ 中断控制器接收到EOI命令后，使ISR[i]=0，表示IR_i的中断服务处理完毕，同时使IRR[i].EN=1，表示可以接收新的IR_i的中断请求

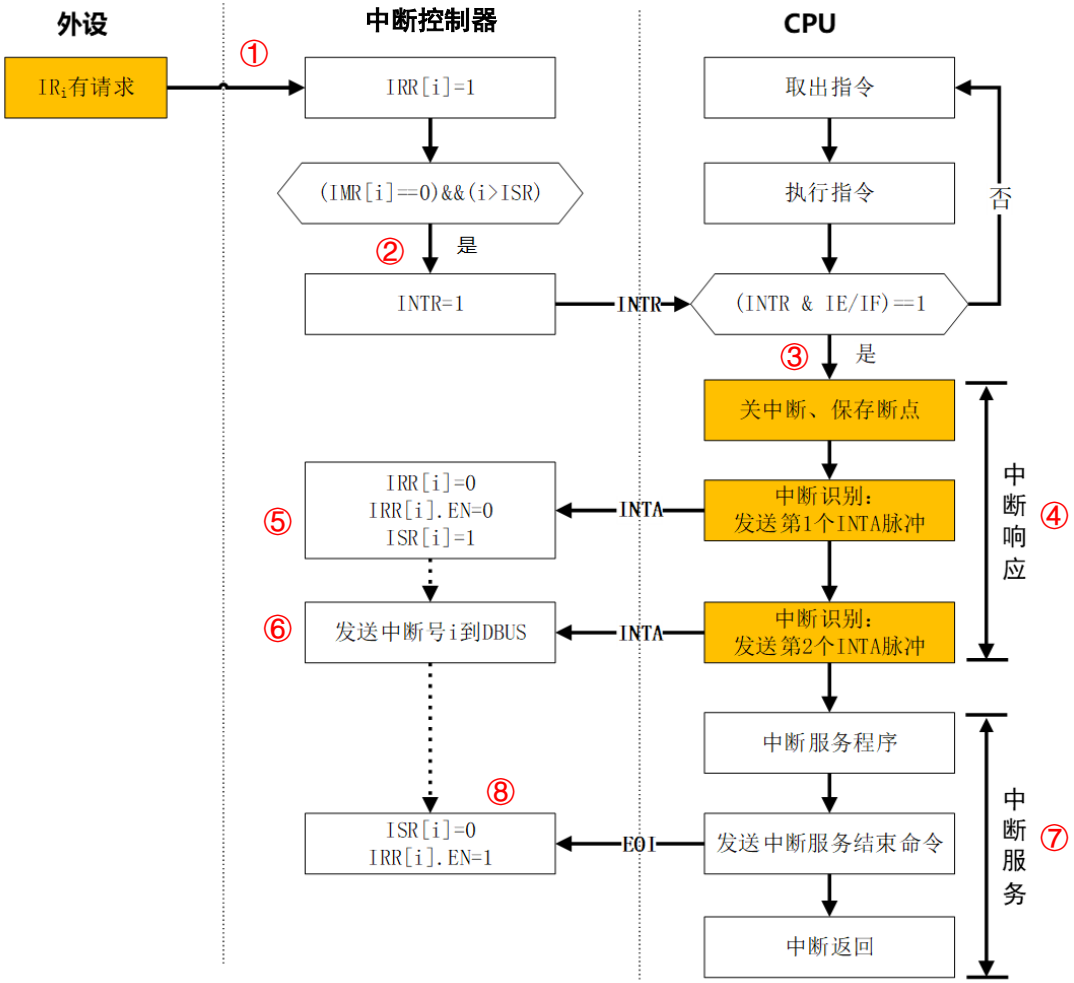


图9.17　中断控制器工作流程

• 9.5.3 中断响应

– 1、中断的响应条件

- CPU响应中断的条件：
  - ① 对应的中断请求未被屏蔽：IMR[i]=0。
  - ② 当前没有更高优先级的其他中断请求：i > ISR。
  - ③ 如果CPU正在执行中断服务，则中断请求应符合嵌套条件。
  - ④ 中断使能位处于使能状态，也就是开中断状态：IE/IF=1；内部异常和不可屏蔽中断不受此限制。
  - ⑤ CPU已经执行完一条指令的最后一个状态周期（中断时机）；内部异常指令因为无法执行完毕，因此其中断时机不受此限制。

– 2、中断响应过程

- CPU的中断响应周期要完成以下工作（见6.7.2小节）：
  - ① 关中断
  - ② 保存断点
  - ③ 中断识别
- 由于中断响应过程中CPU不能执行其他任务，因此中断响应过程也可以看作是由CPU执行中断隐指令完成的。

• 9.5.4 中断识别

– 中断识别的任务是确定中断是由哪个中断源发出的，并获取中断服务程序入口地址；中断识别分为非向量中断和向量中断两种。

– 非向量中断属于共享中断请求信号的软件查询法；CPU响应中断后，跳转到固定地址去执行中断查询程序，由中断查询程序确定是哪一个设备申请中断，并获取中断服务程序入口地址。

– 1、中断号

- 向量中断中每一个设备的中断源都有一个唯一的中断编号，称为中断号，CPU根据中断号快速查找中断服务程序入口地址。

- 中断号是由计算机系统统一分配，通常是固定不变的；x86计算机共有256个中断号（0～255），表9.4为8086中断号分配。

表9.4　8086中断号分配

中断号	中断功能	中断号	中断功能	中断号	中断功能
00	除零异常	08	定时器	10～1F	BIOS中断调用
01	单步中断	09	键盘	10	视频显示I/O调用
02	不可屏蔽中断	0A	保留	11	设备配置检查调用
03	断点中断	0B	串口2	12	存储器容量检查
04	溢出异常	0C	串口1	13	磁盘I/O调用
05	打印屏幕	0D	硬盘	1B	Ctrl+Break控制
06～07	保留	0E	软盘	28～3F	DOS保留
08～0F	可屏蔽外部中断	0F	打印机	60～67	用户软中断保留

– 2、获得中断服务程序入口地址

- 根据中断号，在中断向量表中，可以查找得到中断服务程序入口地址。
- 中断向量表通常存放在内存中，中断向量表中存放的内容称为中断向量，即中断服务程序入口地址；有些计算机将中断服务程序入口地址和程序状态字合在一起称为中断向量。
- 中断向量在中断向量表中的地址称为向量地址。
- 图9.18为中断向量表、中断向量、向量地址之间的关系；其中PCi和PSWi为第i个中断源的中断服务程序入口地址和程序状态字。
- 图9.19为8086计算机的中断向量表，每个中断向量占4个字节（段地址CS=16位、2个字节，段内偏移地址IP=16位、2个字节），256个中断号，共256x4=1024=1K字节，存放在主存地址为0000H～03FFH单元中；假设中断号为n，则向量地址 = nx4。

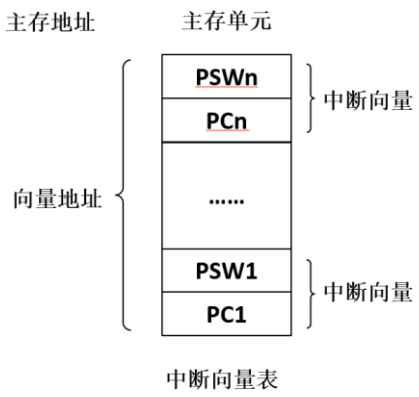


图9.18　中断向量表、中断向量、向量地址之间的关系

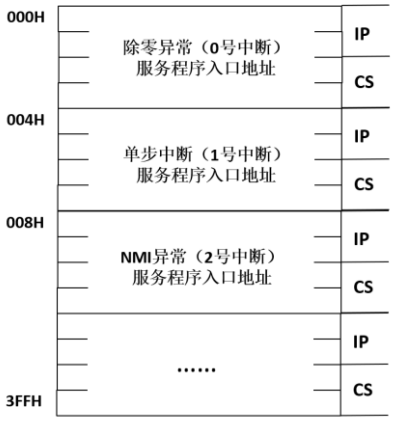


图9.19　8086中断向量表

• 9.5.5 中断处理

– 中断处理的过程就是执行中断服务程序的过程。

– 1、中断服务程序与子函数的差异

① 调用方式不一样：子函数（子程序）在程序中显式调用（x86为call指令，MIPS为jal指令），而中断服务程序大多是随机调用的。

② 保存并修改PC的方式不同：中断服务程序保存并修改PC是由中断隐指令完成的，而子函数保存并修改PC是由调用指令（call指令，jal指令）实现的；中断服务程序入口地址是由中断识别给出的，而子函数入口地址是由调用指令给出的。

③ 现场的内容不同：因为中断服务程序没有调用者，所以需要保存ABI（Application Binary Interface）中约定的由调用者函数保存的寄存器，因此中断服务程序的现场包含的寄存器更多；而子函数只需要保存可能被改写的被调用者保存的寄存器。

④ 返回主程序的方式不同：x86中，子函数返回指令为ret，而中断返回指令为iret；MIPS中子函数返回指令为jr $ra，而中断返回指令为eret。

– 2、单级中断处理流程

- 单级中断处理流程如图9.20a所示；CPU在指令执行阶段的最后，检查是否有中断请求，如果有中断请求，则进入中断响应周期。

① 中断响应：由硬件自动完成，即所谓的中断隐指令；包括关中断、保存断点、中断识别等3个工作（见6.7.2小节和9.5.3小节）。

② 保护现场：主要是保存中断服务程序中用到的（会被修改的）寄存器，通常用压栈指令（push）实现。

③ 中断服务：进行实际的中断事务处理，如数据传输、唤醒等待进程等操作。

④ 恢复现场：就是将保护现场中保存的寄存器释放出来，通常用出栈指令（pop）实现，需要注意的是出栈指令的顺序必须与压栈指令的顺序相反（先进后出）。

⑤ 开中断：通过执行开中断指令（如x86的sti指令），将中断允许触发器IE/IF设置为“1”，使CPU可以接收新的中断请求。

⑥ 中断返回：中断服务程序的最后一条指令是中断返回指令（x86的iret，MIPS的eret），将程序返回到主程序的断点处继续执行。

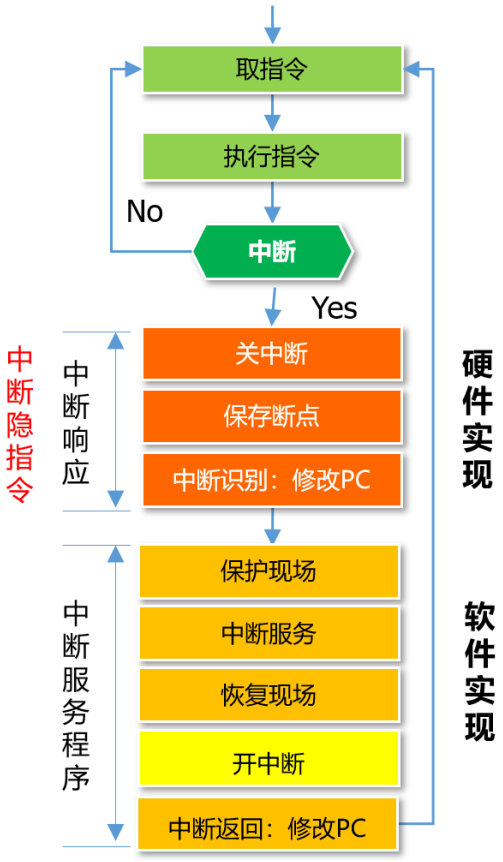


图9.20a　单级中断处理流程

– 3、多重中断处理流程

- 多重中断处理流程如图9.20b所示，与单级中断相比，中断服务程序有几处变化。

① 中断响应：与单级中断相同。

② 保护现场、设置屏蔽字：这里除了要保护现场（寄存器）外，还要保护中断屏蔽字；并要设置新的屏蔽字，从而改变中断处理的优先级，实现多重中断（中断嵌套）。

③ 开中断：目的是使优先级更高的中断服务程序，可以中断当前的中断服务程序。

④ 中断服务：与单级中断相同。

⑤ 关中断：利用关中断指令（如x86的cli指令）关中断，保证恢复现场任务的完整性。

⑥ 恢复现场和屏蔽字：这里除了恢复现场（寄存器）外，还要恢复中断屏蔽字。

⑦ 发送中断结束命令：CPU向中断控制器发送中断结束命令EOI（End Of Interrupt），通知中断控制器当前中断处理完毕，请求中断控制器清除中断服务寄存器ISR中最高优先级的中断位（图9.16）。

⑧ 开中断：与单级中断相同。

⑨ 中断返回：与单级中断相同。

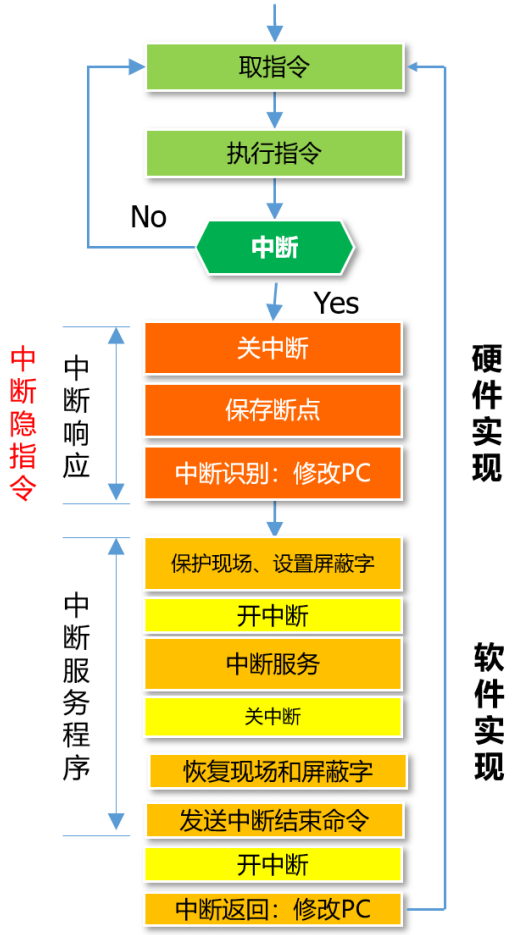


图9.20b 多重中断处理流程

– 4、多重中断服务程序实例

- 以下为80x86计算机中，多重中断嵌套的中断服务程序实例：

	#代码	#功能说明
保护现场	PUSH AX …… PUSH DX	#保护现场 #保护现场
保护屏蔽字	IN AL, 8259_IMR_PORT PUSH AX	#访问8259（中断控制器）的中断屏蔽寄存器IMR #保护中断屏蔽寄存器的内容
设置新的屏蔽字	MOV AL, Curr_Dev_IMR OUT 8259_IMR_PORT, AL	#设置新的中断屏蔽字 #输出新的中断屏蔽字到8259的IMR
开中断	STI	#开中断
中断服务	……	#中断服务
关中断	CLI	#关中断
恢复屏蔽字	POP AX OUT 8259_IMR_PORT, AL	#恢复中断屏蔽字 #恢复中断屏蔽字
恢复现场	POP DX …… POP AX	#恢复现场 #恢复现场
发送EOI	MOV AL, 20H OUT 8259_CMD_PORT, AL	#发送EOI命令 #发送EOI命令到8259的命令口
开中断	STI	#开中断
中断返回	IRET	#中断返回

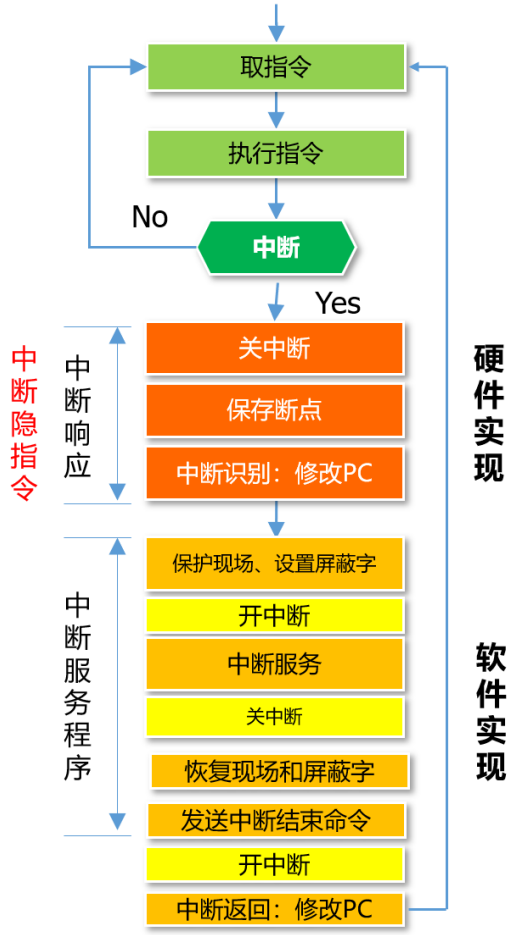


图9.20b　多重中断处理流程

- 例9.4：假设例9.1的计算机系统采用中断驱动方式进行输入输出，CPU的时钟频率为200MHz，硬盘以512字节大小的扇区为单位传输数据，启动阶段发送命令和参数需要90个时钟周期，每次中断服务的开销为400个时钟周期（包括中断响应、中断处理，不包括数据传输），实际传输阶段需要1555个时钟周期，CPU访问硬盘的速率为20MB/s。
- （1）求中断驱动I/O方式中的CPU占用率。
- （2）如果硬盘速率提高到60MB/s，会发生什么情况？

- 解：
- （1）
  - 时钟周期T=1/200MHz；硬盘每个扇区（512个字节）传输的开销为：
    - 启动开销=90T
    - 中断服务开销=400T
    - 数据传输开销=1555T
    - 合计：2045T
  - CPU访问硬盘的速率为20MB/s，则每秒传输的次数为（每次传输512字节）：20MB/512B = 39062.5次/s
  - 中断方式硬盘传输数据占用CPU的时间比例（CPU占用率）为：2045T x 39062.5 / 1s = 2045 x (1/200) x 10^{-6} x 39062.5 = 39.94 %

- （2）
  - 如果硬盘速率提高到60MB/s，中断方式硬盘传输数据占用CPU的时间比例（CPU占用率）为：(60/20) x 39.94 % = 119.82 %
  - 这意味着，即使CPU完全服务于硬盘，硬盘的速率也达不到60MB/s，最高只能达到：(100/119.82) x 60MB/s = 50.07 MB/s

  - 如果采用忙等待的程序查询方式，可以省去中断服务的开销，此时，CPU的占用率 = (90T+1555T) x (60MB/512B) / 1s = 96.38%
  - 勉强可以实现

9.6 DMA方式

• 9.6.1 DMA的基本概念

– DMA：Direct Memory Access，直接内存访问（直接存储器访问）。

– DMA是为了减少I/O过程中CPU用于实际传输的开销而引入的，由DMAC（DMA控制器）临时接管总线，代替CPU控制外部设备和内存之间的批量数据交换，数据直接在外部设备和内存之间进行交换。

– DMA传输前：CPU需要访问DMAC，设置DMA传输参数（包括主存起始地址、数据块长度、传输方向等），并启动设备。

– DMA传输过程中：CPU可以执行其他程序，由DMAC向CPU申请总线控制权用于数据传输。

– DMA传输结束后：DMAC通过中断方式请求CPU对数据缓冲区和DMAC进行后处理。

- DMA方式主要用于高速设备与内存之间的块数据传输，如磁盘、显卡、网卡、声卡等均支持DMA访问。

- DMA方式也可以用于内存数据的内部搬迁（内存数据块的复制或移动）。

- DMA方式与中断方式的区别：

  ① 两者均采用了“请求-应答”机制；中断方式请求的是CPU时间，响应时机是指令周期结束时刻；DMA方式请求的是总线控制权，响应时机是任何一个机器周期结束的时刻。

  ② 中断方式通过CPU执行程序进行实际的数据传送，存在程序执行现场的保护和恢复问题；DMA方式依靠额外的硬件来实现数据传输，其不改变CPU现场，不影响系统性能。

  ③ DMA方式仅仅用于数据的传输；中断方式不仅可以实现数据传输，还可以用于处理各种随机事件，提高计算机的灵活性。

• 9.6.2 内存争用问题

– DMA方式中，DMAC接管总线控制权后，将完成外部设备与内存的数据传输。

– 此时，CPU仍可执行主程序；因此，可能存在DMAC与CPU争用内存的问题（内存争用问题）。

– 为避免由此引起的内存访问冲突，通常有3种解决方法，如图9.21所示：

（a）停止CPU访问内存　　（b）DMAC与CPU交替访问内存　　（c）周期挪用

图9.21　DMA传送方式

– 1、停止CPU访问内存

- 在整个DMA传送过程中（DMAC访问内存期间），停止CPU对内存的访问。

- 优点：控制简单；缺点：CPU可能较长时间不能访问内存，由于外部设备速度较慢，如软盘读一个字节需要32μs，而RAM的存取周期仅为1μs，内存将有31个存储周期空闲，利用率只有1/32=3.125%。

– 2、DMAC与CPU交替访问内存

- 将内存的存取周期分成两段，一段专用于DMAC访问内存，另一段专用于CPU访问内存。

- 优点：不需要总线使用权的申请、建立和交还过程，总线使用权是分时控制的；缺点：会增加内存存储周期，且由于CPU及外部设备的速度与内存不匹配，可能有多个供DMA使用的内存时间片被浪费掉。

– 3、周期挪用

- 也称为周期窃取；只有当DMAC需要访问内存时，CPU才暂停一个存储周期供DMAC访问内存。

- 优点：如果挪用周期期间CPU并不访问内存，则周期挪用方法对CPU的执行没有性能影响；如果挪用周期期间CPU也要访问内存，则DMA优先访问内存；周期挪用方法是DMA传送的主要方法。

• 9.6.3 DMA控制器

– 图9.22为DMA控制器（DMAC）的基本结构；主要包括：

① 地址计数器：初始值为数据块的起始地址，每传送1个字或字节，自动加1。
② 字计数器：初始值为数据块长度，每传送1个字或字节，自动减1。
③ 数据缓冲寄存器（DBR）：DMAC作为从设备时，用来接收CPU传输的数据；DMAC作为主设备时，用来暂存数据。
④ 命令寄存器（DCR）：用来接收CPU的控制命令。
⑤ 状态寄存器（DSR）：负责向CPU反馈DMAC的状态。
⑥ DMA控制逻辑：包括控制和时序电路，以及状态标志，支持中断机制。

– DMA传输控制信号有4个：

① DREQ：DMA请求，由外部设备发给DMAC。
② HOLD：总线请求，由DMAC发给CPU。
③ HLDA：总线响应，由CPU发给DMAC。
④ DACK：DMA响应，由DMAC发给外部设备。

– 在DMA传输前和传输结束后，DMAC是从设备；在DMA传输过程中，DMAC是主设备。

– 图9.22的DMAC称为第三方DMA；现代总线技术普遍支持总线主控技术，将DMAC集成到I/O接口中，称为第一方DMA。

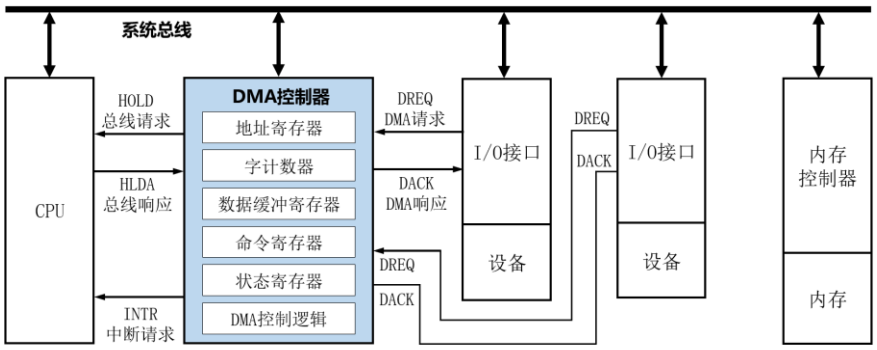


图9.22 DMA控制器基本结构

• 9.6.4 DMA传输流程

– 图9.23为DMA读操作详细流程（周期挪用方式），包括：

① **DMA准备阶段**（预处理阶段）：包括初始化DMA、启动设备、其他进程运行（当前进程阻塞）等。

② **数据传输阶段**：不需要CPU参与，由DMAC负责实现设备与内存的数据交互，包括设备准备数据、设备发送DMA请求、DMAC申请总线、总线仲裁和授权、DMA数据传输、传输控制等。

③ **DMA结束阶段**（后处理阶段）：由CPU执行中断服务程序实现。

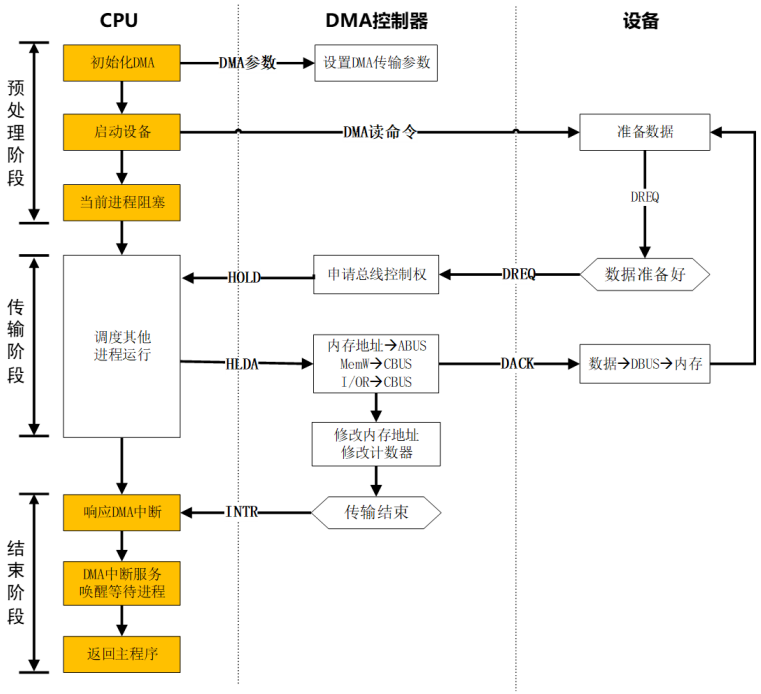


图9.23　DMA读操作详细流程（周期挪用方式）

- 例9.5：某磁盘采用DMA方式与CPU交换信息，其传输速率为20MB/s。若DMA的预处理阶段需要200个时钟周期，DMA完成传输后的中断处理阶段需要400个时钟周期。如果DMA平均传输的数据块长度为512B，请问：磁盘工作时，200MHz的处理器进行DMA传输时的CPU占用率是多少？如果采用4KB的传输单位呢？（不考虑DMAC与CPU争用内存对CPU性能的影响。）

- 解：
  - 时钟周期T=1/200MHz
  - 要达到20MB/s的传输速率，则每秒传输的次数为（每次传输512字节）：20MB/512B = 39062.5次/s
  - DMA处理的3个阶段，只有第1阶段（预处理阶段）和第3阶段（后处理阶段）需要CPU参与，相应的时间为：200T + 400T = 600T
  - DMA方式磁盘传输数据的CPU占用率 = 600T x 39062.5 / 1s = 600 x (1/200) x 10^{-6} x 39062.5 = 11.72 %
  - 如果采用4KB的传输单位，则CPU占用率 = 600T x (20MB/4KB) / 1s = 600 x (1/200) x 10^{-6} x 4882.8125 = 1.46 %
  - 例9.4，中断方式磁盘传输数据的CPU占用率（传输速率为20MB/s） = 39.94 %
  - 例9.1，查询方式磁盘传输数据的CPU占用率（传输速率为20MB/s） = 5005%

9.7 通道方式

• 9.7.1 通道的基本概念

– 为了进一步减少I/O操作中的中断次数和CPU占用时间，通常把对外部设备的管理、操作控制以及数据传输从CPU中分离出来，交由专门的I/O处理器（IOP）负责，这种I/O处理器称为通道控制器。

– 通道有自己独立的指令系统，可以代替CPU独立地执行一系列的I/O操作指令。

– 通道具体以下的功能：

① 根据CPU要求，组织设备与系统连接。
② 通过设备控制器向设备发出操作命令。
③ 指出数据在设备中的位置和在内存缓冲区内的位置。
④ 检查设备和设备控制器的工作状态。
⑤ 向CPU反映设备、设备控制器及通道本身的状态信息。
⑥ 进行必要的信息格式变换。

– 设备控制器：介于通道和设备之间，是通道对外部设备实行具体控制的部件。

• 9.7.2 通道的类型

– 根据设备共享通道的情况，以及信息传送速度的要求，通道分为：字节多路通道、选择通道、成组多路通道等3类。

– 1、字节多路通道

• 与通道连接的各个设备以字节为单位交叉使用通道，也称为字节交叉方式，图9.24。

• IBM370系列计算机的字节多路通道有256个子通道，可支持256路信息的并行传送；字节多路通道通常用于连接低速设备，如打印机等。

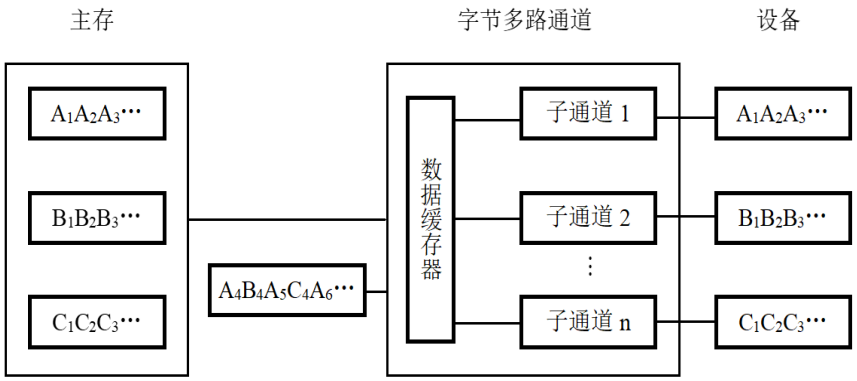


图9.24　字节多路通道传送

– 2、选择通道

- 选择通道中设备以成组数据连接传送方式占用通道，直到指定数量的数据全部传送完毕，通道才转为其他设备服务。
- 图9.25为选择通道组织框图，图9.26为选择通道数据传送示意图。
- 选择通道在物理上可以连接多个设备，但这些设备不能同时工作。
- 选择通道适用于大批量数据的高速传送；早期的IDE硬盘接口就采用选择通道模式。

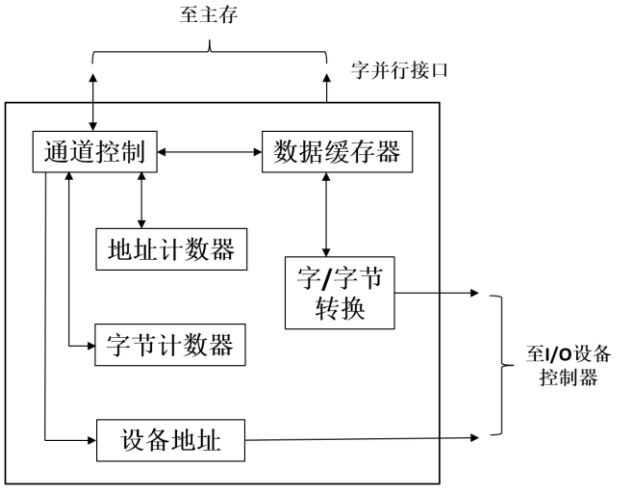


图9.25　选择通道组织框图

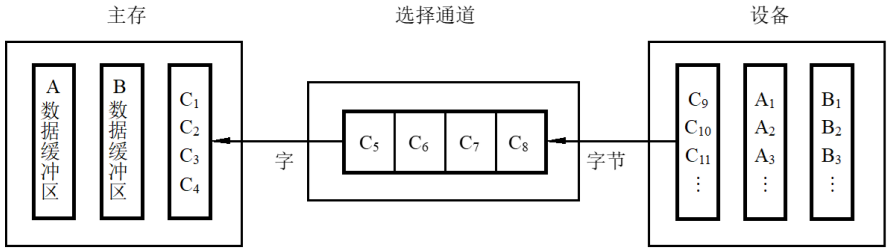


图9.26　选择通道数据传送示意图

– 3、成组多路通道

- 成组多路通道是字节多路通道和选择通道的结合。
- 成组多路通道中多个设备以数据组（块）为单位交叉使用通道。
- 当某设备占用通道时，连续传送一组数据（选择通道），然后将通道让给其他设备（字节多路通道）。
- 在一批数据传送过程中，要多次与设备断开和连接，通道的硬件结构比较复杂。
- 成组多路通道适用于中高速设备，如磁盘、磁带等；早期的SCSI总线就采用成组多路通道。

• 9.7.3 CPU对通道的控制

– 1、执行I/O指令

- CPU通过执行特权I/O指令“STRAT I/O”来启动通道。
- CPU启动通道后，通道和外部设备将独立工作。

– 2、处理来自通道的中断请求

- 当通道和外部设备发生异常或通道处理结束时，CPU接收来自通道的“中断”请求，对故障进行测试和处理，以及对通道传输进行后期处理。
- 收集外部设备和通道自身的状态信息，并送入内存固定单元中存放，供CPU测试外部设备和通道的状态时使用。

• 9.7.4 通道结构的发展

– 随着通道结构的进一步发展，出现了两种计算机I/O系统结构：

– 1、通道结构的I/O处理器，也称为输入输出处理器（IOP）。

- IOP（I/O Processor）可以与CPU并行工作，提供高速的DMA处理能力，实现数据的高速传送。
- IOP广泛应用于中小型及微型计算机中。

– 2、外围处理器（PPU）

- PPU（Peripheral Processor Unit）独立于CPU工作，它有自己的指令系统，能完成算术与逻辑运算、存储器的读写、与外设交换信息等操作。
- PPU一般用于大型、高效率的计算机系统中。

9.8 常见I/O设备

• 9.8.1 键盘

– 键盘是最常用的输入设备，主要由按键开关、盘架、编码器和接口电路组成。

– 1、按键开关

- 触点式按键开关、无触点式按键开关。
- 薄膜键盘、机械键盘、静电电容键盘。

– 2、编码器

- 全编码键盘、非编码键盘。

标准键盘

小键盘

• 9.8.2 鼠标

– 鼠标也是最常用的输入设备，主要有机械式鼠标和光电式鼠标两类。

– 1、机械式鼠标

  • 机械式鼠标底部有一个可自由滚动的球。

– 2、光电式鼠标

  • 早期光电式鼠标需要在特制的鼠标板上使用；目前市场上主流的光电式鼠标，采用“光眼（Optical Sensor）”技术，可以在任何非反光的表面使用。

机械式鼠标

光电式鼠标

无线鼠标

• 9.8.3 打印机

– 打印机是最常用的输出设备，可分为击打式打印机（如针式打印机）和非击打式打印机（如喷墨打印机、激光打印机）两大类。

– 1、针式打印机

- 由字车传动机构、打印针控制机构、色带驱动机构、走纸机构、打印机状态传感器等组成。

– 2、喷墨打印机

- 根据喷墨方式分为连续式喷墨打印机和随机式喷墨打印机。

– 3、激光打印机

- 由激光扫描系统、电子成像部分、字符发生器和控制电路等组成；激光打印的过程：充电、曝光、显影、转印、定影、消电、清洁。

– 4、打印机语言

- 计算机通过打印机语言控制打印机，打印机语言主要有：页描述语言（Page Description Language，PDL）、嵌入式语言（Escape Code Language，ECL）；HP公司的PCL（Printer Command Language）语言和Adobe公司的PostScript语言都属于页描述语言。

针式打印机

喷墨打印机

激光打印机

• 9.8.4 显示器

– 显示器也是最常用的输出设备，显示器的主要性能指标包括：

① 分辨率：1024x768、1280x1024、1920x1080、4096x2160（4K）、7680x4320（8K）等。

② 长宽比：4:3、16:9、21:9等。

③ 屏幕尺寸：21英寸、23英寸、27英寸、32英寸等。

④ 灰度级：黑白显示器中的灰度级表示像素点的亮度差别，彩色显示器中的灰度级表示颜色的差别；8位灰度表示256级灰度或颜色，现代彩色显示器的颜色位数可达32位（2^{32}=4,294,967,296级颜色）。

⑤ 刷新：单位时间的刷新次数称为刷新频率（帧频、帧率），刷新频率过低人眼会感到闪烁，主流液晶显示器的刷新频率为60Hz～240Hz。

⑥ 显存：也称刷新存储器，用于暂存当前屏幕显示信息和不断刷新屏幕显示；显存容量=分配率x灰度级位数；显存带宽=分辨率x灰度级位数x刷新频率。

– 根据显示信息内容的不同，显示器可分为：字符显示器、图形显示器、图像显示器。

– 根据显示器件的不同，显示器可分为：阴极射线管显示器（CRT显示器，Cathode Ray Tube）、液晶显示器（LCD显示器，Liquid Crystal Display）、发光二极管显示器（LED显示器，Light-emitting Diode）等。

CRT显示器

LCD显示器

LED显示器

- 例9.6：若显示器的工作方式为：分辨率=1024x768，24位真彩色，刷新频率（刷新速度）=72Hz，请回答下列问题：
- （1）显存容量至少需用多少KB？
- （2）若保留50%带宽用于其他非刷新功能，则显存的总带宽应为多少？
- （3）为了提高显存的带宽，应采取哪些技术措施？

- 解：
- （1）
  - 显存容量=分配率x灰度级位数=1024x768x24/8=2304KB

- （2）
  - 刷新屏幕所需的带宽=分辨率x灰度级位数x刷新频率=2304KBx72Hz=165888KB/s≈170MB/s

  - 保留50%带宽用于其他非刷新功能，因此显存带宽=170x2=340MB/s

- （3）
  - 提高显存带宽的技术措施：
    ① 使用高速度的DRAM芯片，组成刷新存储器。

    ② 刷新存储器采用多体交叉结构。

    ③ 提高刷新存储器内显示控制器的内部总线宽度。

    ④ 刷新存储器采用双端口存储器结构，将刷新端口与更新端口分开。

• 9.8.5 硬盘存储器

– 硬盘存储器是磁表面存储器的一种，磁鼓、软盘、磁带都属于磁表面存储器。

– 1、读写原理

- 磁表面存储器利用磁性材料剩磁的两种磁化方向来记录信息：例如，S → N表示1，N → S表示0。
- 写入原理：电流转换为磁场。
- 读出原理：磁场转换为电流。

– 2、记录方式

- 归零制、不归零制、调相制、调频制、见“1”就翻转不归零制、改进调频制等。

– 3、磁盘存储器

- （1）磁盘存储器内部结构
  - 包括磁记录介质、磁盘控制器（软盘称为FDC，硬盘称为HDC）、磁盘驱动器（软盘称为FDD，硬盘称为HDD）。

- （2）磁盘数据信息编址和记录格式
  - 包括：记录面、磁道、扇区。
  - 将半径相等的磁道集合称为圆柱面。
- （3）磁盘存储器的技术指标
  - 存储密度：包括道密度（TPI，Track Per Inch）和位密度（BPI，Bit Per Inch）。
  - 存储容量：磁盘存储容量（双面盘）= 盘片数 x 2 x 磁道数 x 扇区数 x 扇区容量。
  - 平均定位时间：包括寻道时间（找磁道的时间，通常为4～10ms）和等待时间（找扇区的时间，取决于磁盘的转速，转速为5400转/分钟、7200转/分钟、10000转/分钟的等待时间分别为5.56ms、4.16ms、3ms）。
  - 数据传输速率：假设位密度为M（bit/英寸）、转速为V（英寸/s），则数据传输速率 = M x V（bit/s）。

软盘

软盘驱动器

硬盘

移动硬盘

• 9.8.6 磁盘阵列

– RAID：廉价冗余磁盘阵列/独立冗余磁盘阵列，简称磁盘阵列，Redundant Arrays of Inexpensive Disks，Redundant Arrays of Independent Disks；是由美国加州大学伯克利分校的D.A.Patterson教授提出的一种多磁盘存储系统。

– RAID将多个磁盘按照一定的方式进行组织和管理，构成一个大容量、高性能、高容错的存储系统；RAID具有以下的特征：

① 在操作系统或硬件的支持下，多个磁盘构成一个更大的逻辑存储空间，从而扩充存储系统容量。

② 连续数据被分割成相同大小的数据块，相邻的数据块分布在不同的磁盘上，在进行数据访问时，多个磁盘并发工作，从而提升存储系统访问性能。

③ 采用校验编码提高多磁盘系统可靠性，在某个磁盘出现故障后，磁盘阵列仍可正常工作。

– 根据不同的数据组织与管理形式，RAID分为多个级别：

① RAID 0：主要应用于对访问性能要求高，但对数据的可靠性要求不高的场合。

② RAID 1：主要应用于对数据的可用性要求高，且读操作所占比例较高的场合。

③ RAID 2：成本较高，目前很少被使用。

④ RAID 3：与RAID 3的性能特点类似。

⑤ RAID 4：不适合应用于有大量写操作的场合。

⑥ RAID 5：对大、小数据的读与写都具有较好的性能，是应用最为广泛的阵列级别。

⑦ RAID 6：RAID 5只能支持一块磁盘故障，RAID 6可支持两块磁盘同时出现故障。

磁盘阵列

磁盘阵列

• 9.8.7 光盘存储器

– 1、光盘存储器的类型

- 按照访问模式分为：
  - ① 只读型光盘（ROM，也称CD-ROM）
  - ② 一次写入型光盘（WORM）
  - ③ 可擦重写型光盘（ReWrite）：又分为相变光盘和磁光盘（MO）两种。
  - ④ 直接重写型光盘（OverWrite）

- 按照容量的不同分为：
  - ① CD（也称CD-ROM）
  - ② 数字化视频光盘（DVD）
  - ③ 蓝光光盘（BD）

- 按照光盘盘片直径的大小分为：14英寸、12英寸、8英寸、5.25英寸、3.5英寸、2.5英寸、1.8英寸等。

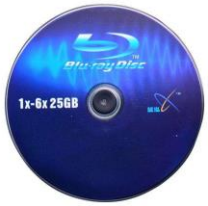
– 2、光盘存储器的记录原理

① CD-ROM
② CD-WORM
③ CD-RW


光驱

CD-R

CD-RW

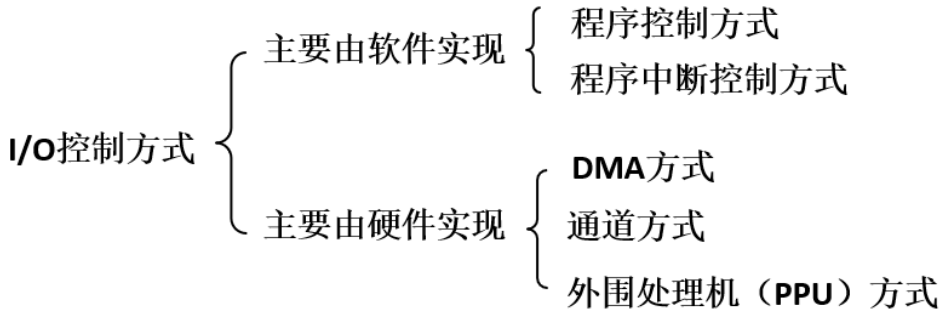


蓝光光盘

本章小结

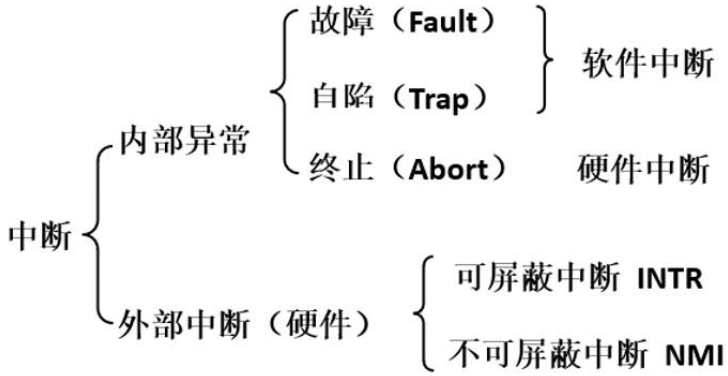
- 输入输出系统是典型的软硬件协同系统（I/O硬件、I/O软件）
- 输入输出设备的特性：异步性、实时性、独立性
- I/O接口的功能：设备寻址、数据交互、设备控制、状态检测、数据缓冲、格式转换；以及中断、时序控制和数据检错、纠错等功能
- I/O接口的功能部件：数据缓冲寄存器（DBR）、设备状态寄存器（DSR）、设备命令寄存器（DCR）、设备存储器、地址译码器、数据格式转换逻辑
- I/O接口的编址方法：统一编址、独立编址
- 现代计算机中，用户并不能直接访问设备，必须通过操作系统间接访问设备
- 操作系统中的I/O软件层次（3个层次）：与操作系统无关的I/O库、与设备无关的操作系统调用库、独立的设备驱动程序

- I/O接口的分类
- 数据传输控制方式（I/O控制方式）：



- 程序控制方式：包括程序查询方式（轮询方式，Polling）和直接传送方式
- 简单设备查询流程、复杂设备查询流程
- 程序查询（也称为轮询）的两种策略：忙等待（独占式查询）和定时轮询

- 中断的分类：



- 中断优先级：
  - 响应优先级（固定不变）
  - 处理优先级（可以改变）
- 中断屏蔽技术：可以改变处理优先级
- 中断屏蔽字
- CPU运行轨迹图

- 中断请求的硬件支持：
  ① 中断请求寄存器
  ② 中断屏蔽寄存器
  ③ 中断服务寄存器
  ④ 中断优先级排队电路
  ⑤ 中断允许触发器

- CPU响应中断的条件：
  ① 对应的中断请求未被屏蔽：IMR[i]=0
  ② 当前没有更高优先级的其他中断请求：i > ISR
  ③ 如果CPU正在执行中断服务，则中断请求应符合嵌套条件
  ④ 中断使能位处于使能状态，也就是开中断状态：IE/IF=1；内部异常和不可屏蔽中断不受此限制
  ⑤ CPU已经执行完一条指令的最后一个状态周期（中断时机）；内部异常指令因为无法执行完毕，因此其中断时机不受此限制

- CPU中断响应周期要完成的工作（中断隐指令）：
  ① 关中断
  ② 保存断点
  ③ 中断识别

- 中断号、中断向量、向量地址、中断向量表

- 中断服务程序与子函数的差异

- 单级中断处理流程、多重中断（中断嵌套、嵌套中断）处理流程

- DMA：Direct Memory Access，直接内存访问（直接存储器访问）

- DMA解决内存争用的3种方式：
  ① 停止CPU访问内存
  ② DMAC与CPU交替访问内存
  ③ 周期挪用

- DMAC：DMA控制器

- DMA传输控制信号（4个）：
  ① DREQ：DMA请求，由外部设备发给DMAC。
  ② HOLD：总线请求，由DMAC发给CPU。
  ③ HLDA：总线响应，由CPU发给DMAC。
  ④ DACK：DMA响应，由DMAC发给外部设备。

- DMA传输流程（3个阶段）：
  ① DMA准备阶段（预处理阶段）
  ② 数据传输阶段
  ③ DMA结束阶段（后处理阶段）

- 通道：为了进一步减少I/O操作中的中断次数和CPU占用时间，通常把对外部设备的管理、操作控制以及数据传输从CPU中分离出来，交由专门的I/O处理器（IOP）负责，这种I/O处理器称为通道控制器

- 通道有自己独立的指令系统，可以代替CPU独立地执行一系列的I/O操作指令

- 通道的类型：字节多路通道、选择通道、成组多路通道等3类

- CPU对通道的控制：执行I/O指令、处理来自通道的中断请求

- 通道结构的发展：外围处理器（PPU）

- 常见I/O设备：
  ① 键盘
  ② 鼠标
  ③ 打印机
  ④ 显示器
  ⑤ 硬盘存储器
  ⑥ 磁盘阵列
  ⑦ 光盘存储器

习题（P368-371）

- 9.2
- 9.3
- 9.4
- 9.5
- 9.6
- 9.8

关于作业提交

- 1周内必须提交（上传到学院的FTP服务器上），否则认为是迟交作业；如果期末仍然没有提交，则认为是未提交作业
  - 作业完成情况成绩=第1次作业提交情况*第1次作业评分+第2次作业提交情况*第2次作业评分+……+第N次作业提交情况*第N次作业评分
  - 作业评分：A（好）、B（中）、C（差）三挡
  - 作业提交情况：按时提交（1.0）、迟交（0.5）、未提交（0.0）

- 请采用电子版的格式（PPT文档）上传到FTP服务器上，文件名取“学号+姓名+第X次作业.ppt”
  - 例如：3062019220384O+孙明策+第9次作业.ppt

- 第9次作业提交的截止日期为：2023年6月18日晚上24点。

实践训练（实验6）

- 在MARS仿真器中利用虚拟仿真键盘设备和字符终端显示设备分别实现程序查询方式和程序中断方式输入输出，比较二者的不同

- 在Logisim中将单周期MIPS处理器或单总线MIPS处理器增加中断机制，使其能接收外部按键中断

Thanks