西安科技大学
XI' AN UNIVERSITY OF SCIENCE TECHNOLOGY

第二章 操作系统的硬件基础

刘晓建

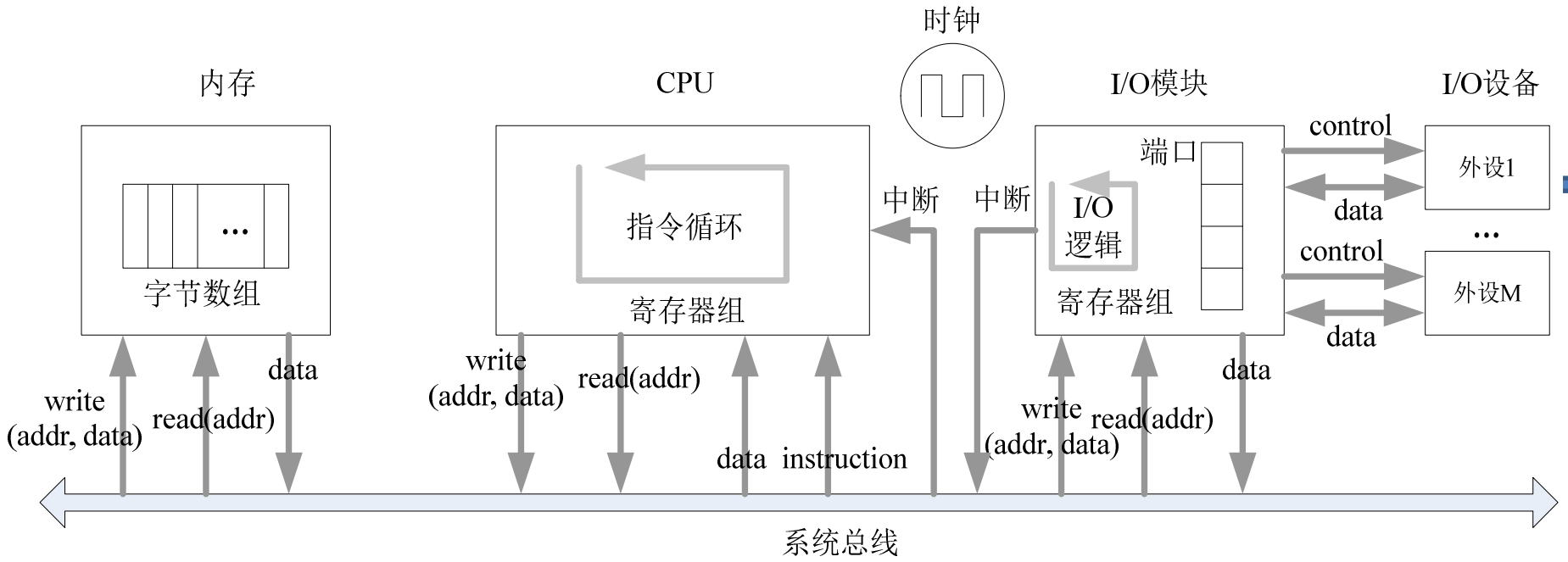
2017年8月30日

操作系统是直接工作在硬件资源上的软件系统

要了解操作系统的工作原理，首先必须了解硬件的基本特性和工作原理，以及硬件为计算（或程序执行）所提供的基本支撑。

本章主要学习：

- 计算机的构成部分以及各部分之间的互连关系
- 内存、CPU、I/O模块的基本特性和工作原理
- 指令循环、异常和异常处理的概念，明确它们在计算机程序执行过程中的基础性作用。
- 处理器的两种运行模式和模式切换，明确它们在保障计算机安全、提高计算机的可用性方面的重要作用。



◆处理器是唯一具有处理功能的单元

◆内存模块充当外部存储设备与处理器之间交换数据的高速缓存（Cache），通常用来存放用户程序和数据，并驻留操作系统内核；

◆I/O模块用来管理和控制外部设备，并充当外部设备与处理器之间交互的中介；

◆系统总线是这三类模块的连接方式，实现它们之间控制、状态和数据信息的交换；

◆时钟模块用来产生时钟脉冲，同步和协调这四类模块之间的交互。

内存

内存是易失性存储介质，其中存储的数据随着计算机的掉电而消失。

◆从用户角度看，内存中存储操作系统内核以及应用程序的指令和数据。

◆从系统角度看，内存实际上充当高速的处理器与相对低速的外部设备之间的缓存（Cache），以提高系统整体运行效率。

如何访问字节中的位呢?

内存的抽象模型

◆大多数计算机使用**字节**（Bytes）（8位）作为内存的最小访问单元

◆每个字节有一个唯一的**物理地址**。

◆内存可以被抽象为字节的一维数组，首字节的地址从0开始，顺序编址。

◆内存中所有M个字节的物理地址构成了一个集合{0,1,2,…,M-1}，称为**“物理地址空间”**。

字（word）

由若干个确定数目的连续字节所构成的存储区块（Chunk）称为“字”（Word）。

字的大小通常用它包含的字节数或位数来表示。常见的字大小为4字节（32位）或8字节（64位），它是计算机系统的一个重要参数。通常，一个整数值或指针值用一个字的大小来表示，因此如果一个计算机系统的字大小是32位，那么一个整数值或指针值（即地址值）也就用32位（即4个字节）来表示。

C语言中不同类型的数据占据的字（或字节）大小不同。

字的大端（Big endian）和小端（Little endian）

多字节数据对象都用相邻的字节来存储。这样就带来两个问题：

◆一是多字节数据对象的地址怎样确定？通常用字节的最小地址作为多字节数据对象的地址。

◆二是多个字节的顺序如何规定？

有一个32位整数：

0x12345678

要把它存放在地址分别为0x100，0x101，0x102和0x103的四个字节中，如何在内存中保存它呢？如何确定这个整数的地址呢？

处理器（CPU）：唯一具有处理能力的单元

◆算术逻辑单元ALU

◆控制单元

◆寄存器组

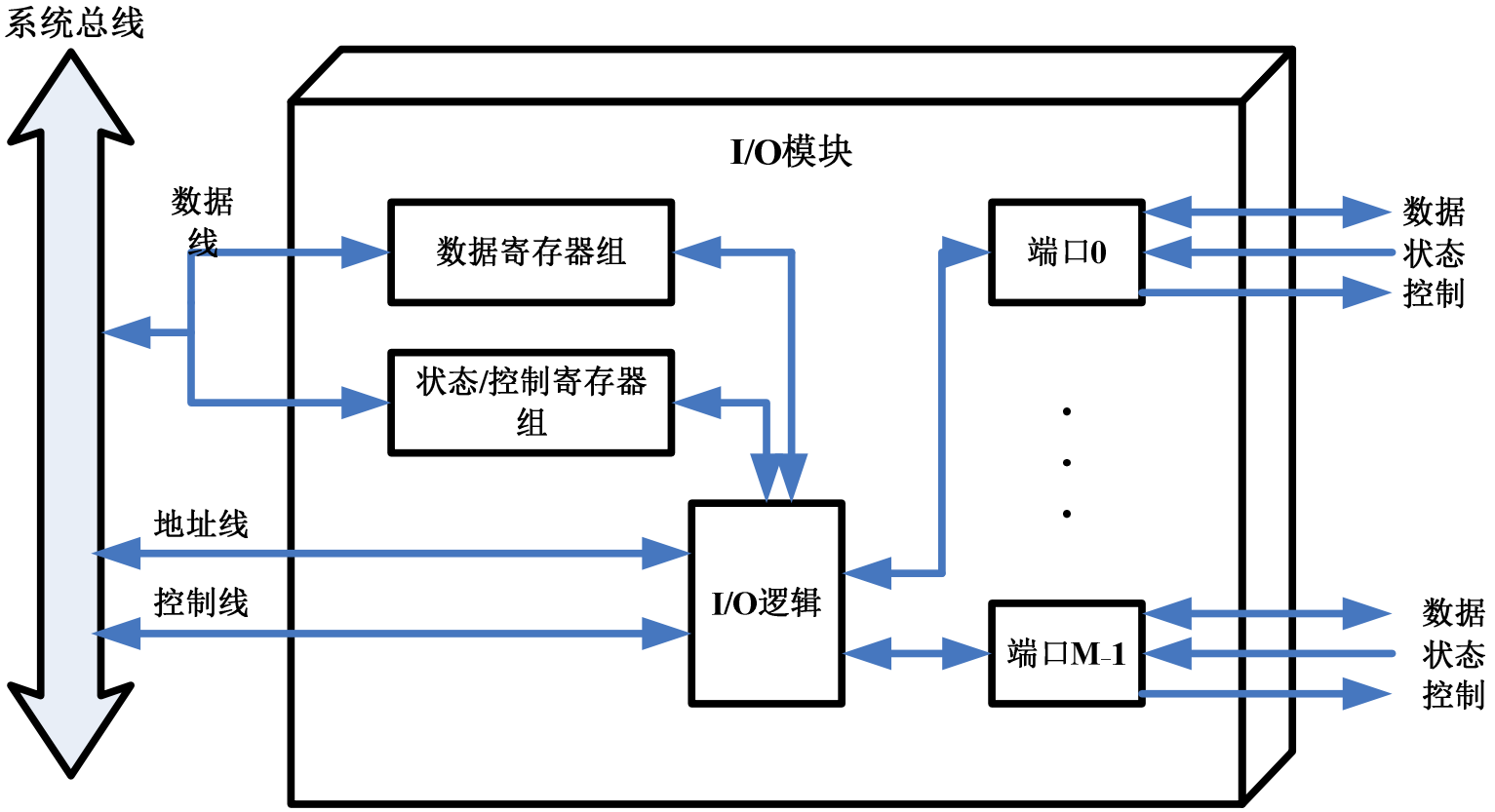
◆总线接口

ALU、控制单元和总线接口等这些逻辑，在CPU设计时就已经确定下来，程序员是无法改动的。只有部分寄存是向程序员开放的接口，换句话说，程序员可以通过一些寄存器读取CPU的内部状态，通过一些寄存器为CPU提供数据和地址。

根据寄存器对程序员的可见性，可以把寄存器组分为：

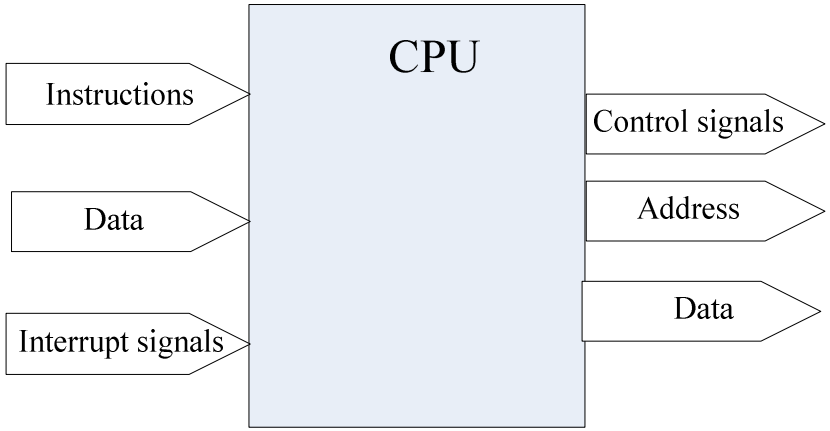
◆用户可见（可读或可读写）的寄存器

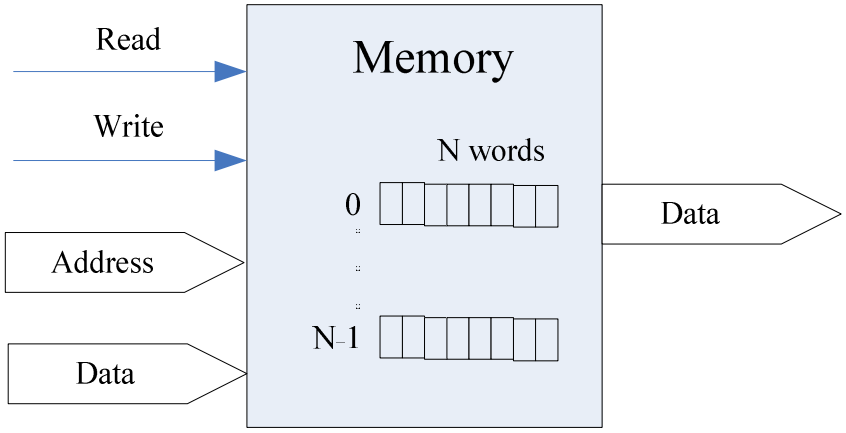
◆控制和状态寄存器（通常不可访问，或仅可读取，不可改写）

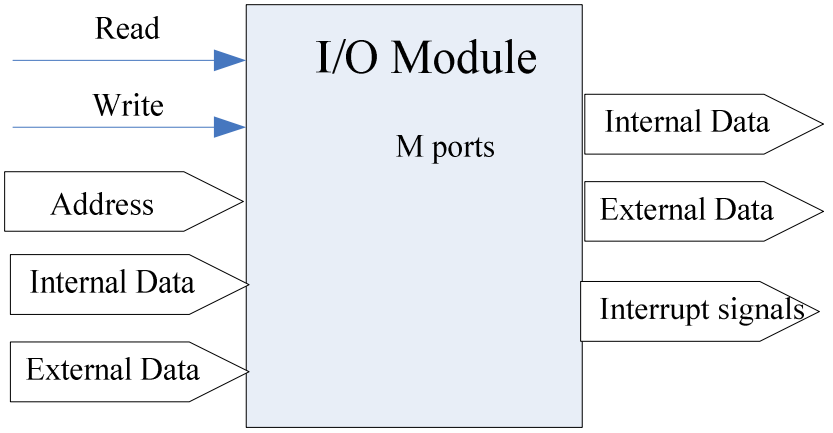


I/O模块屏蔽了不同外部设备的细节，使得处理器能够以一种简单、一致的读、写命令的方式访问外部设备。同时，I/O模块也为处理器对外部设备的控制留有足够的细节，如磁带机的回卷操作等。

系统总线： 把其它部件连接起来的一组物理导线以及通信协议。







系统总线通常包括50到上百条导线，每条线都具有特定的含义和功能。尽管有多种不同的总线设计，我们总可以把总线中的导线按照功能分为**数据线、地址线和控制线**。除此之外，还有为总线上的模块供电的导线。

◆数据线提供系统模块间交换数据的路径，通常包括32，64，128条或更多数目的导线，这些线的数目称为数据总线的“**带宽**”

◆地址总线用来指示数据总线上数据的源地址或目标地址。

◆控制线用来控制访问和使用数据线和地址线的方式

- ◆内存读写
- ◆I/O读写
- ◆中断请求/应答
- ◆总线仲裁、时钟、重置等

【例2-1】假设有一个微处理器产生一个16位的地址（例如，假设程序计数器和地址寄存器都是16位）并且具有一个16位的数据总线。

（1）如果连接到一个16位存储器上，处理器能够直接访问的最大存储器地址空间为多少?

（2）如果连接到一个8位存储器上，处理器能够直接访问的最大存储器地址空间为多少?

（3）如果输入指令和输出指令可以表示8位端口号，这个微处理器可以支持多少8位I/O端口?

（4）如果存储器的地址是32位的，那么如何用16位的地址总线传输32位的地址呢?

（5）如果某个数据是32位的，那么如果用16位的数据总线来传输32位的数据呢?

2.2 指令

指令集是由处理器为程序员提供的一组基本指令的集合，处理器能够**识别并执行**指令集中的每一条指令。任何高级语言程序都必须被翻译成基本指令的序列，才能被处理器最终执行。

每一条指令由操作码（opcode）和一个或多个操作数（operand）组成，写作

op R1,R2,…

其中op为操作码，R1，R2，…为操作数。一条指令用二进制（或16进制）串来编码。

有些指令集的所有指令编码的长度都是固定的，如ARM指令集，每条指令用4个字节来编码；而有些指令集的指令编码是变长的，如IA32指令编码的长度是1~15字节长度。

```
mov    $100,  %eax
mov    %eax,  %ebx
mov   0x100,  %eax
mov   (%eax) , %ebx
mov   4(%ebp), %eax
mov (%edi, %ecx), %eax
```

立即数寻址

寄存器寻址

直接内存寻址

间接内存寻址

基址偏移量内存寻址

学习指令时要注意：

◆操作数的寻址方式

◆操作数的大小（字节数）

◆指令的编码大小（字节数）

◆指令的编码方式

比如，

（1）mov 0x0405, %eax，把地址0x0405上多少个字节的数据移动到%eax中呢?

（2）如何编码mov $0x0405, %eax和mov 0x0405, %eax呢?

【例2-7】数据移动指令的格式为rmmove rA, D(rB)，将寄存器rA中的值move到内存位置D(rB)。D(rB)表示“基址+偏移量”式的内存寻址方式，基址存放在寄存器rB中，D是一个立即数，表示偏移量。该指令编码使用6个字节，其中操作码占用一个字节，三个操作数占用剩下的5个字节。halt指令编码只占一个字节，编码为0x00，没有操作数。

字节	操作码		1		2	3	4	5
rmmov rA, D(rB)	4	0	**rA**	**rB**	D			
halt	0	0						

如何编码rmmove %eax, 4(%ebx)???

【例2-8】设当前PC=0x100，试解释内存中二进制指令序列

0x100: 40630008000000

的含义。假定寄存器%esi和%ebx的编码分别为0x6和0x3，内存采用小端存储多字节数据。

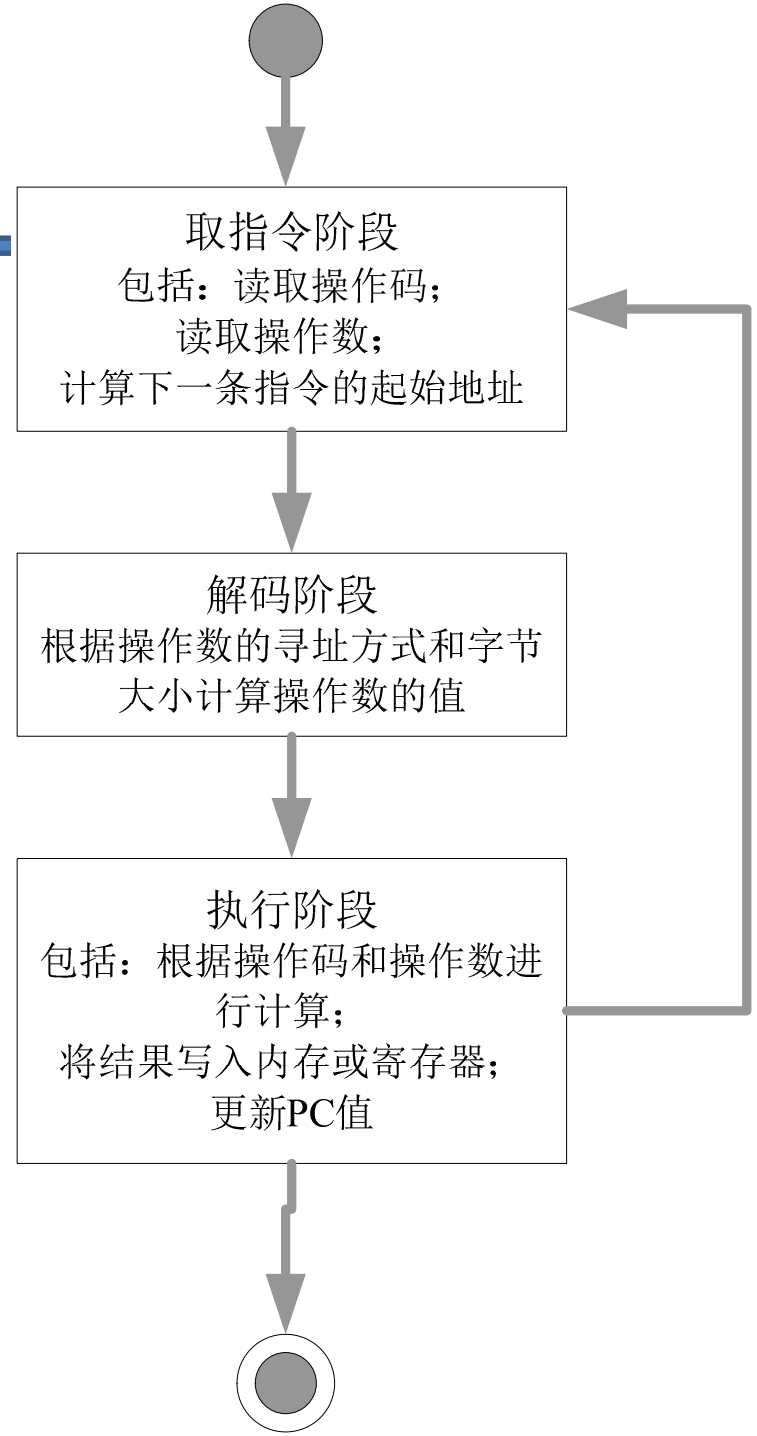
2.3 指令循环和异常处理

处理器执行一条机器指令的过程可以用指令循环来描述。一个指令循环分为取指阶段、解码阶段和执行阶段。

◆**取指阶段**。处理器以PC（程序计数寄存器）中的值所指示的内存地址为起始地址，取出指令编码。如果指令编码为固定长度，则取出这个长度的指令编码；如果指令编码为可变长度，则先取出操作码，然后根据操作码判断操作数的个数、字节大小以及指令长度，并依次取出操作数。然后计算下一条指令的起始地址。

◆**解码阶段**。根据指令操作数的寻址方式和字节大小，计算操作数的值。

◆**执行阶段**。执行该指令，并根据指令的后效将计算结果写入寄存器或内存位置，然后将PC设置为下一条指令的起始地址。



【例2-8】设当前PC=0x100，试解释内存中二进制指令序列

0x100: 40630008000000

0x100: rmmov %esi, 0x800(%ebx)

0x106：halt

异常和异常处理

上面的指令循环没有考虑到异常（Exception）的发生。异常是一个事件（Event），而一个事件是处理器或其它设备的状态的一种变化。

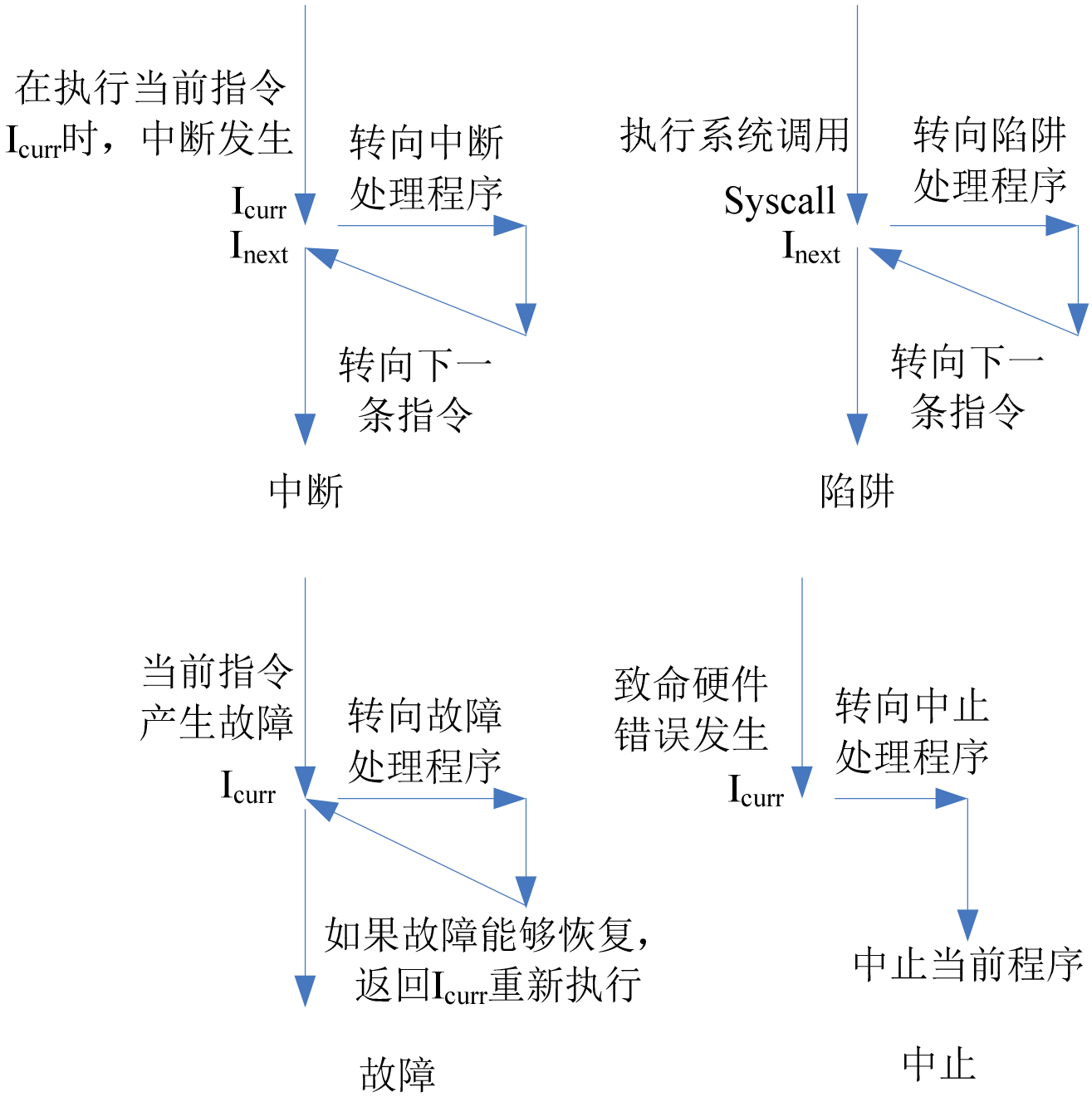
◆外部异常：由时钟或I/O设备等外部设备触发

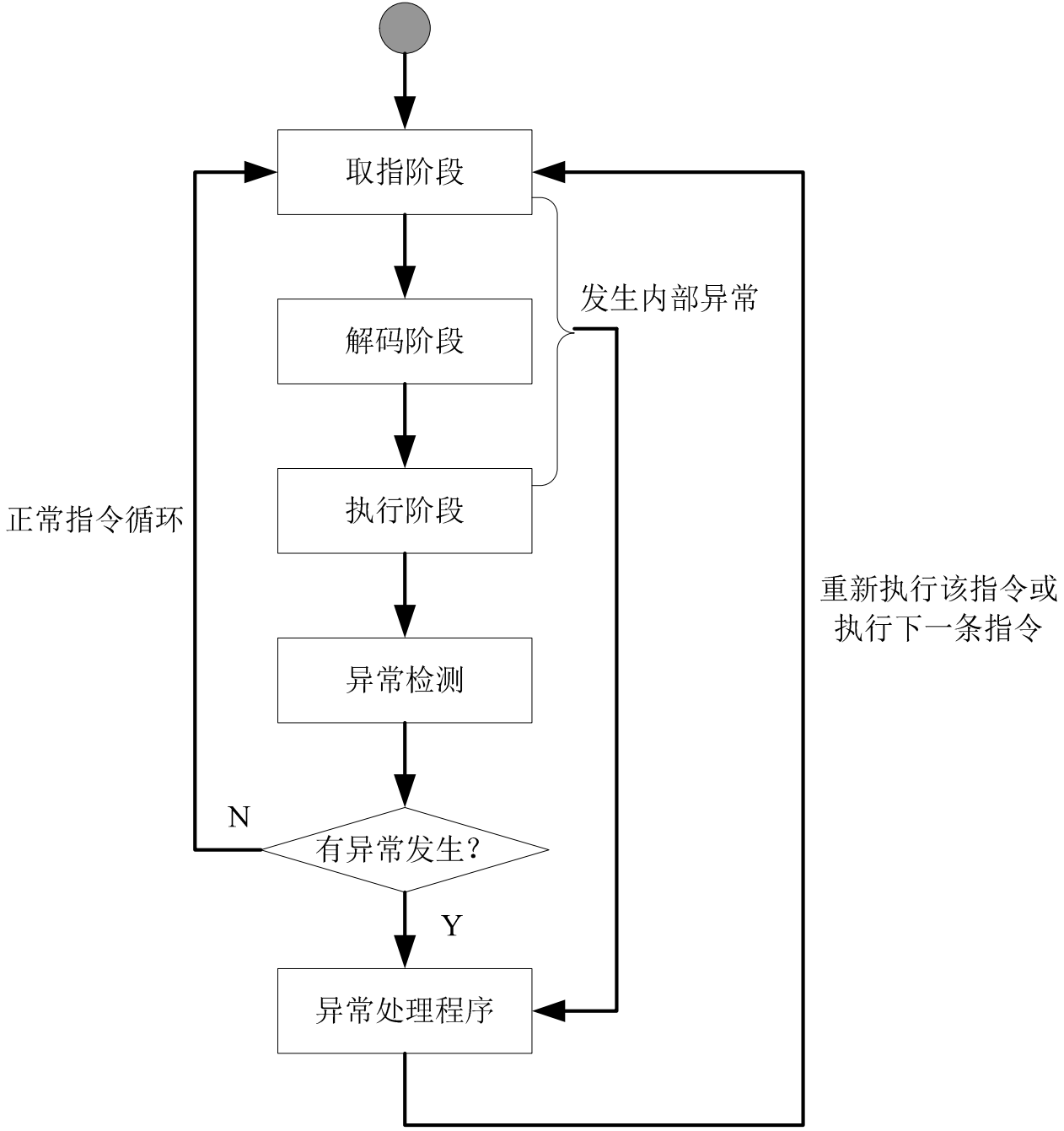
◆内部异常：正在处理的指令触发，比如算术运算指令出现浮点溢出，操作数地址越界等。

◆异常通常具有较高优先级，应当优先得到处理。当异常发生时，处理器正常的指令循环就要被打断，转而先执行异常处理流程，处理完毕后，有可能再返回到原来的程序继续执行。

表2-7 异常的分类

类别	发生的原因	同步/异步	返回行为	举例
中断	由时钟或I/O设备发出的信号触发，表明这些设备的状态发生了某种改变	异步	总是返回到下一条指令继续执行	打印机准备就绪；打印操作完成等
陷阱	程序指令主动发起的异常	同步	总是返回到下一条指令继续执行	系统调用；断点(Breakpoint)
故障	由正在执行的指令产生的各种错误引发。这些错误可能是可恢复的，也可能是不可恢复的	同步	可能返回到当前指令重新执行；也可能终止当前程序	算术溢出；除零错误；缺页故障；保护性故障等。如程序引用一个未定义的内存区域，或程序企图向只读文本段写入数据
致命故障	由不可恢复的致命错误引发，通常由硬件错误引发。	同步	终止当前程序	硬件错误，如DRAM或SRAM极性错误等


在执行当前指令
I_{curr}时，中断发生
I_{curr}
I_{next}
转向中断
处理程序
转向下一
条指令
中断
执行系统调用
Syscall
I_{next}
转向陷阱
处理程序
转向下一
条指令
陷阱
当前指令
产生故障
I_{curr}
转向故障
处理程序
如果故障能够恢复，
返回I_{curr}重新执行
故障
致命硬件
错误发生
I_{curr}
转向中止
处理程序
中止当前程序
中止


取指阶段
解码阶段
执行阶段
异常检测
有异常发生?
N
Y
异常处理程序
发生内部异常
正常指令循环
重新执行该指令或
执行下一条指令

2.4 处理器的运行模式和模式切换

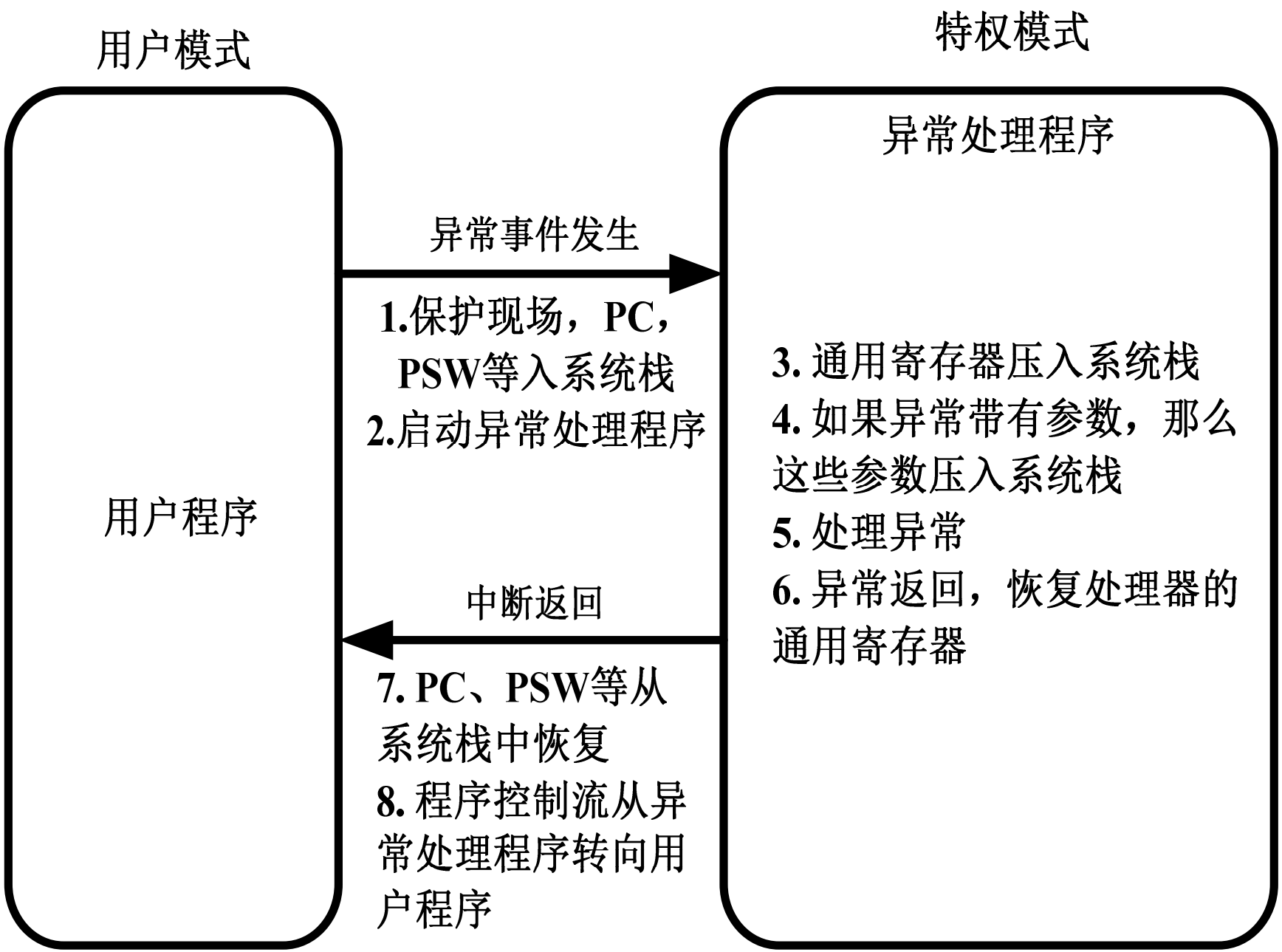
异常处理过程不仅涉及到控制流的转换，而且还伴随着处理器运行模式的切换。处理器一般具有两种宏状态或运行模式：

◆**用户模式**（User mode）。用户程序通常运行在用户模式下，不能直接访问受保护的系统资源，不能执行特权指令，也不能改变处理器的运行模式。

◆**特权模式**（Privileged mode）。系统软件（如操作系统内核、一些具有特权的操作系统任务等）、异常处理程序等运行在特权模式下，可以访问系统的所有资源，而且能够自由的切换系统的运行模式。

◆Intel X86体系结构定义了Ring0~Ring3四个级别的模式，但只用到两个模式。Ring0是特权模式，Ring3是用户模式。Arm定义了6个模式。

处理器为什么要区分运行模式呢?


用户模式
特权模式
异常处理程序
异常事件发生
1.保护现场，PC，PSW等入系统栈
2.启动异常处理程序
3. 通用寄存器压入系统栈
4. 如果异常带有参数，那么这些参数压入系统栈
5. 处理异常
6. 异常返回，恢复处理器的通用寄存器
用户程序
中断返回
7. PC、PSW等从系统栈中恢复
8. 程序控制流从异常处理程序转向用户程序

- 作业

P48　3、5、8

1、一个程序P在操作系统OS上运行，OS是否需要读取P的代码，然后替P去执行?

2、多道程序批处理系统和分时系统都使用了_____基本原理，造成多道程序或多个用户同时使用计算机的假象。

3、在微内核结构中，应用程序和系统服务程序使用_____机制进行交互，优点是降低了应用程序与系统服务之间的_____，缺点是_____下降了。

4、虚拟机可以直接对_____层进行虚拟化，也可以对_____层进行虚拟化。JVM（Java VM）属于_____类型的虚拟机。

5、内存地址0x100存放了一个4字节十六进制整数，即0x100: 01030507。已知整数是按小端方式存储的，请将这个十六进制整数转化为十进制整数。如果这4个字节表示一条4字节指令，请写出这条指令的二进制编码。

6、当应用程序正在执行时，发生了一个异常，那么控制流要从这个应用程序转移到_____，而且处理器的模式要从_____模式转换到_____模式。

7、如果地址总线是16位的，那么处理器能够直接访问的地址空间是_____；如果数据总线也是16位的，那么要传输一个4字节数据，需要_____个总线周期。