计算机组成原理

江南大学《计算机组成原理》

第四章

- 输入输出系统

更多考试真题
请扫码获取

主要内容

外围设备的定时方式与信息交换方式

程序查询方式

程序中断方式

DMA方式

通道方式

一、外围设备的定时方式

输入/输出设备同CPU交换数据的过程：

输入过程：

(1) CPU把一个地址值放在地址总线上，这一步将选择某一输入设备；

(2) CPU等候输入设备的**数据成为有效**；

(3) CPU从数据总线读入数据，并放在一个相应的寄存器中。

输出过程：

(1) CPU把一个地址值放在地址总线上，选择输出设备；

(2) CPU把数据放在数据总线上；

(3) 输出设备认为**数据有效**，从而把数据取走。

问题的关键在于： 究竟什么时候数据才成为有效?

首先解决主机与外围设备在时间上的同步问题。

由于输入/输出设备本身的速度差异很大，因此，对于不同速度的外围设备，需要有不同的定时方式，总的说来，CPU与外围设备之间的定时，有以下三种情况。

(1)速度极慢或简单的外围设备

CPU认为数据一直有效，CPU只要接收或发送数据就可以了。

例如： 对机械开关来讲，CPU可以认为输入的数据一直有效，因为机械开关的动作相对CPU的速度来讲是非常慢的；

对显示二极管来讲，CPU可以认为输出一定准备就绪，因为只要给出数据，显示二极管就能进行显示。

(2)慢速或中速的外围设备

由于这类设备的速度和CPU的速度并不在一个数量级，或者由于设备(如键盘)本身是在不规则时间间隔下操作的。因此，CPU与这类设备之间的数据交换通常采用异步定时方式。

接收： 如果CPU需要从外设接收一个字，则它：

- 首先询问外设的状态，如果该外设的状态标志表明设备已 “准备就绪”，那么CPU就从总线上接收数据；
- CPU在接收数据以后，发出输入响应信号，告诉外设已经把数据总线上的数据取走；
- 然后，外设把“准备就绪”的状态标志复位，并准备下一个字的交换。

发送: 如果CPU需要向外设发送一个字，则它：

- CPU询问外设是否准备就绪。如果外设已准备就绪， CPU便并送出数据。
- 外设接收数据以后，将向CPU发出“数据已经取走”的通知。

如果CPU询问外设时，外设没有“准备就绪”，那么它就发出表示外设“忙”的标志。于是，CPU将进入一个循环程序中等待，并在每次循环中询问外设的状态，一直到外设发出“准备就绪”信号以后，才从外设接收数据。

这种在CPU和外设间用问答信号进行定时的方式叫做应答式数据交换 。

(3)高速的外围设备

由于这类外设是以相等的时间间隔操作的，而CPU也是以等间隔的速率执行输入/输出指令的，因此，这种方式叫做**同步定时方式**。一旦CPU和外设发生同步，它们之间的数据交换便靠时钟脉冲控制来进行。

更快的同步传送要采用**直接内存访问(DMA)**方式。

二、外设的识别与端口寻址

为了能在众多的外设中寻找或挑选出要与主机进行信息交换的外设，就必须对**外设进行编址**。外设识别是通过地址总线和接口电路中的外设识别电路来实现的，I/O**端口地址**就是主机与外设直接通信的地址，CPU可以通过端口发送命令、读取状态和传送数据。

1、端口地址编址方式

I/O端口编址方式有两种：一种是**I/O映射方式**，即把I/O端口地址与存储器地址分别进行独立的编址；另一种是**存储器映射方式**，即把端口地址与存储器地址统一编址。

（1）独立编址

在这种编址方式中，主存地址空间和I/O端口地址空间是相对独立的，分别单独编址。CPU访问主存时，由主存读写控制线控制；访问外设时，由I/O读写控制线控制。

（2）统一编址

在这种编址方式中，I/O端口地址和主存单元的地址是统一编址的，把I/O接口中的端口作为主存单元一样进行访问，不设置专门的I/O指令。

2、独立编址方式的端口访问

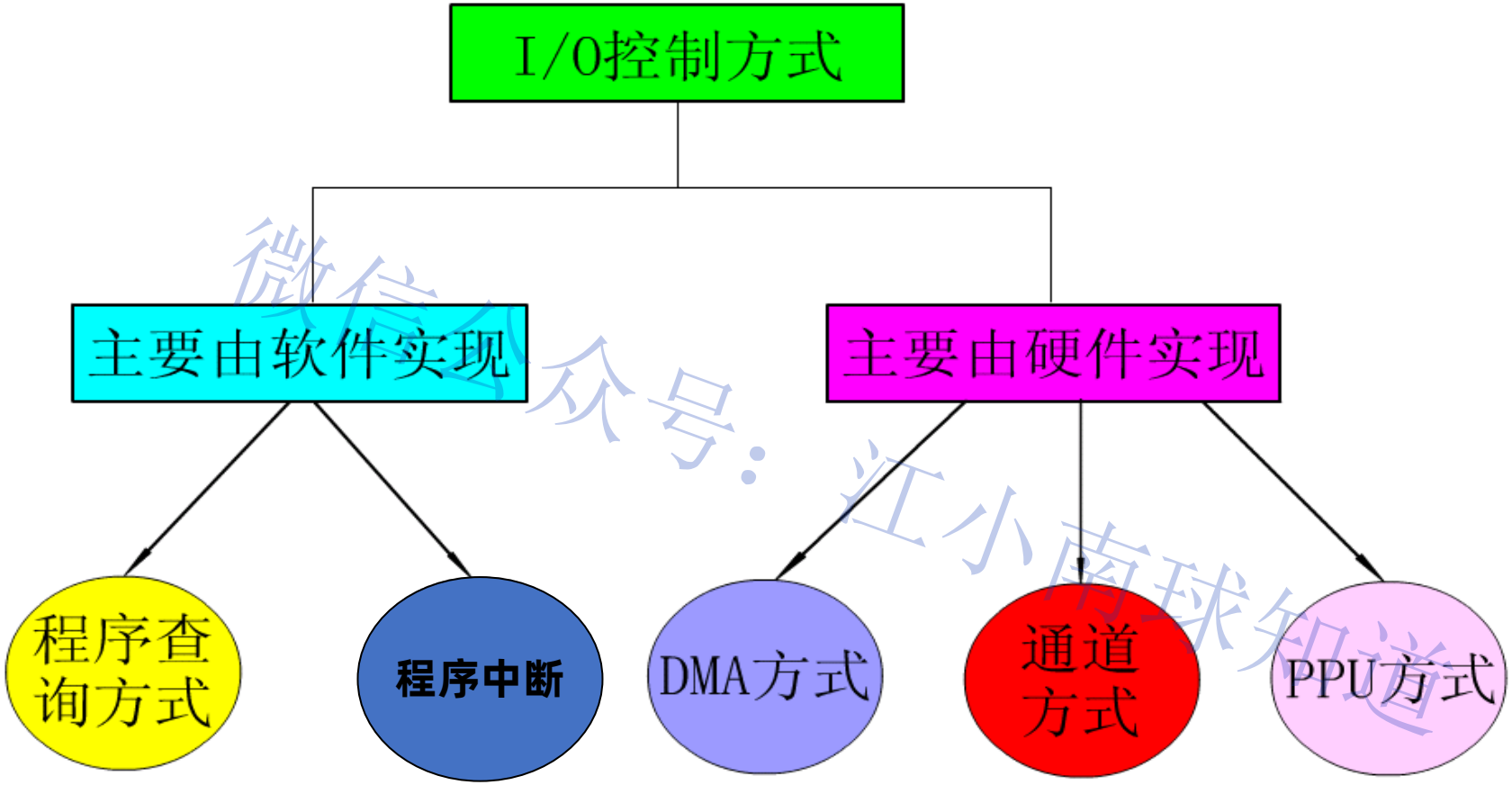
独立编址方式在Intel系列、Z80系列微机及大型计算机中得到广泛应用，Intel 80x86的I/O地址空间由2^{16}（64K）个独立编址的8位端口组成。两个连续的8位端口可作为16位端口处理；四个连续的8位端口可作为32位端口处理。因此，I/O地址空间最多能提供64K个8位端口、32K个16位端口、16K个32位端口或总容量不超过64KB的不同端口的组合。

80x86的专用I/O指令IN和OUT有直接寻址和间接寻址两种类型。直接寻址I/O端口的寻址范围为0000～00FFH，至多为256个端口地址。

间接寻址由DX寄存器间接给出I/O端口地址。DX寄存器长16位，所以最多可寻址2^{16}=64K个端口地址。

CPU一次可实现字节（8位）、字（16位）或双字（32位）的数据传送。32位端口应对准可被4整除的偶地址；16位端口应对准偶地址；8位端口可定位在偶地址，也可定位在奇地址。

三、信息交换方式



程序查询方式

无条件传送方

条件传送方式

1、直接程序控制方式

程序查询方式是主机与外设间进行信息交换的**最简单**方式，程序查询方式的核心问题在于需要不断地查询I/O设备是否准备就绪。

1. 程序查询的基本思想

由CPU执行一段输入输出程序来实现主机与外设之间数据传送的方式叫做程序直接控制方式。根据外设的不同性质，这种传送方式又可分为**无条件传送**和**程序查询**方式两种。

为了保证数据传送的正确进行，就要求CPU在程序中查询外设的工作状态。如果外设尚未准备就绪，CPU就循环等待，只有当外设已作好准备，CPU才能执行I/O指令进行数据传送，这就是程序查询方式。

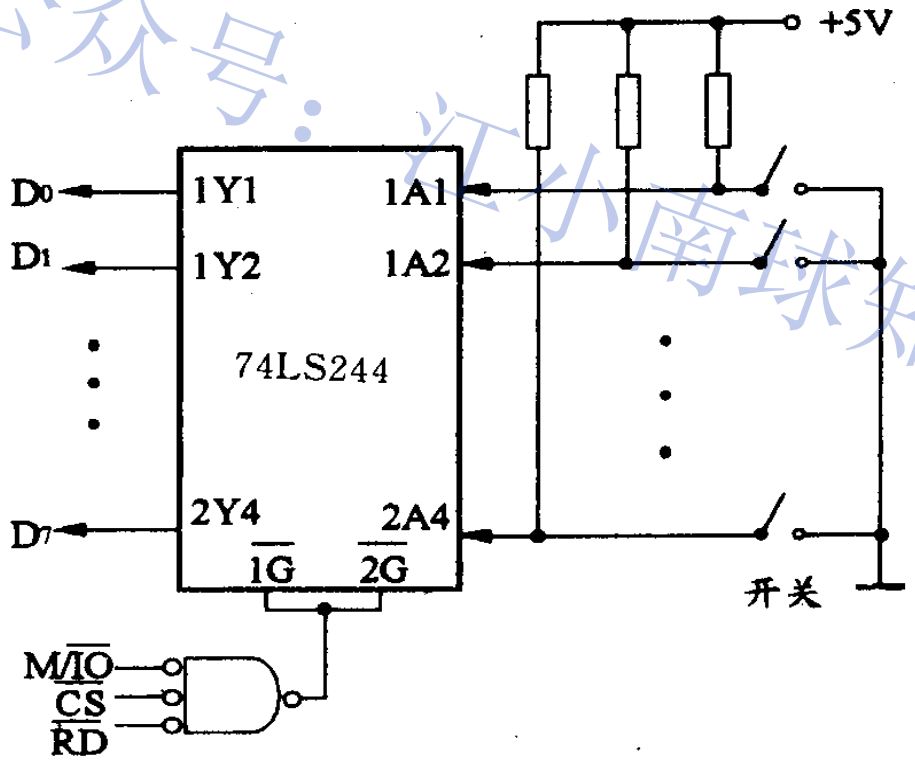
程序查询方式——无条件传送

外设总是准备好

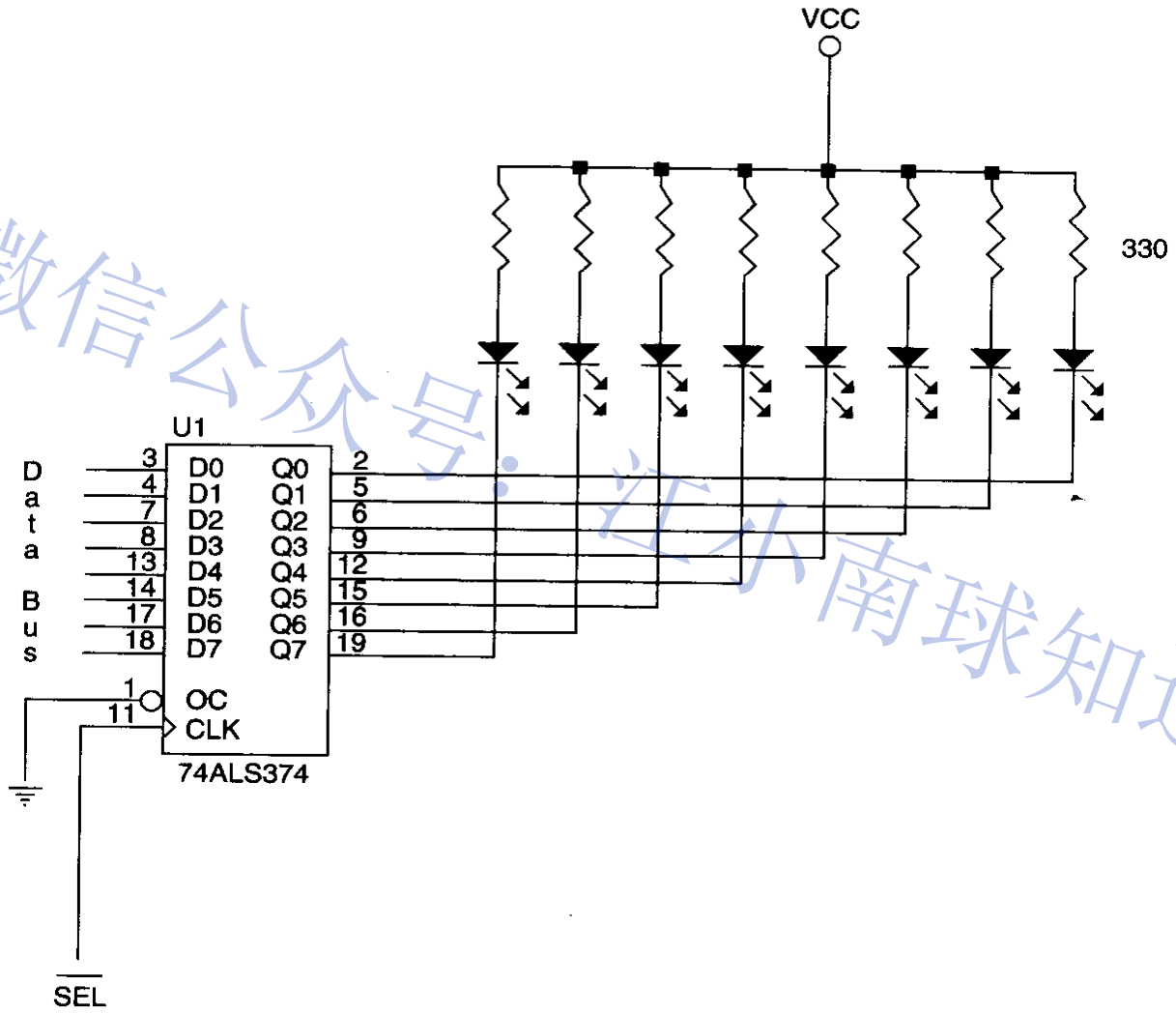
- 输入——数据已经准备好
- 输出——外设已准备好接收

只有数据，没有状态，同步方式

不需要过多的程序处理，在需要与外设交换信息时，随时访问I/O端口



程序查询方式——无条件传送



2. 程序查询方式的工作流程

①**预置传送参数**。在传送数据之前，由CPU执行一段初始化程序，预置传送参数。传送参数包括存取数据的主存缓冲区首地址和传送数据的个数。

②**向外设接口发出命令字**。当CPU选中某台外设时，执行输出指令向外设接口发出命令字启动外设，为接收数据或发送数据做应有的操作准备。

③**从外设接口取回状态字**。CPU执行输入指令，从外设接口中取回状态字并进行测试，判断数据传送是否可以进行。

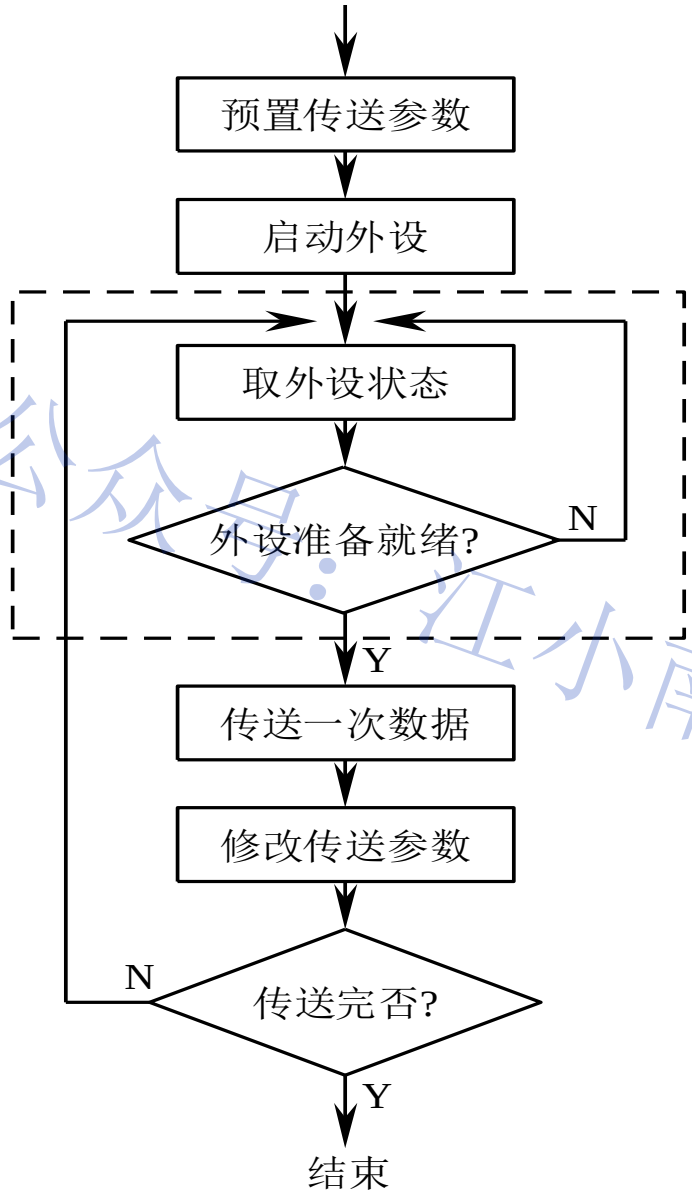
④**查询外设标志**。CPU不断查询状态标志。如果外设没有准备就绪，CPU就踏步进行等待，一直到这个外设准备就绪，并发出“外设准备就绪”信号为止。

⑤**传送数据**。只有外设准备好，才能实现主机与外设间的一次数据传送。输入时，CPU执行输入指令，从外设接口的数据缓冲寄存器中接收数据；输出时，CPU执行输出指令，将数据写入外设接口的数据缓冲寄存器中。

⑥**修改传送参数**。每进行一次数据传送之后必须要修改传送参数，其中包括主存缓冲区地址加1，传送个数计数器减1。

⑦**判断传送是否结束**。如果传送个数计数器不为0，则转第③步，继续传送，直到传送个数计数器为0，表示传送结束。

程序查询方式流程：



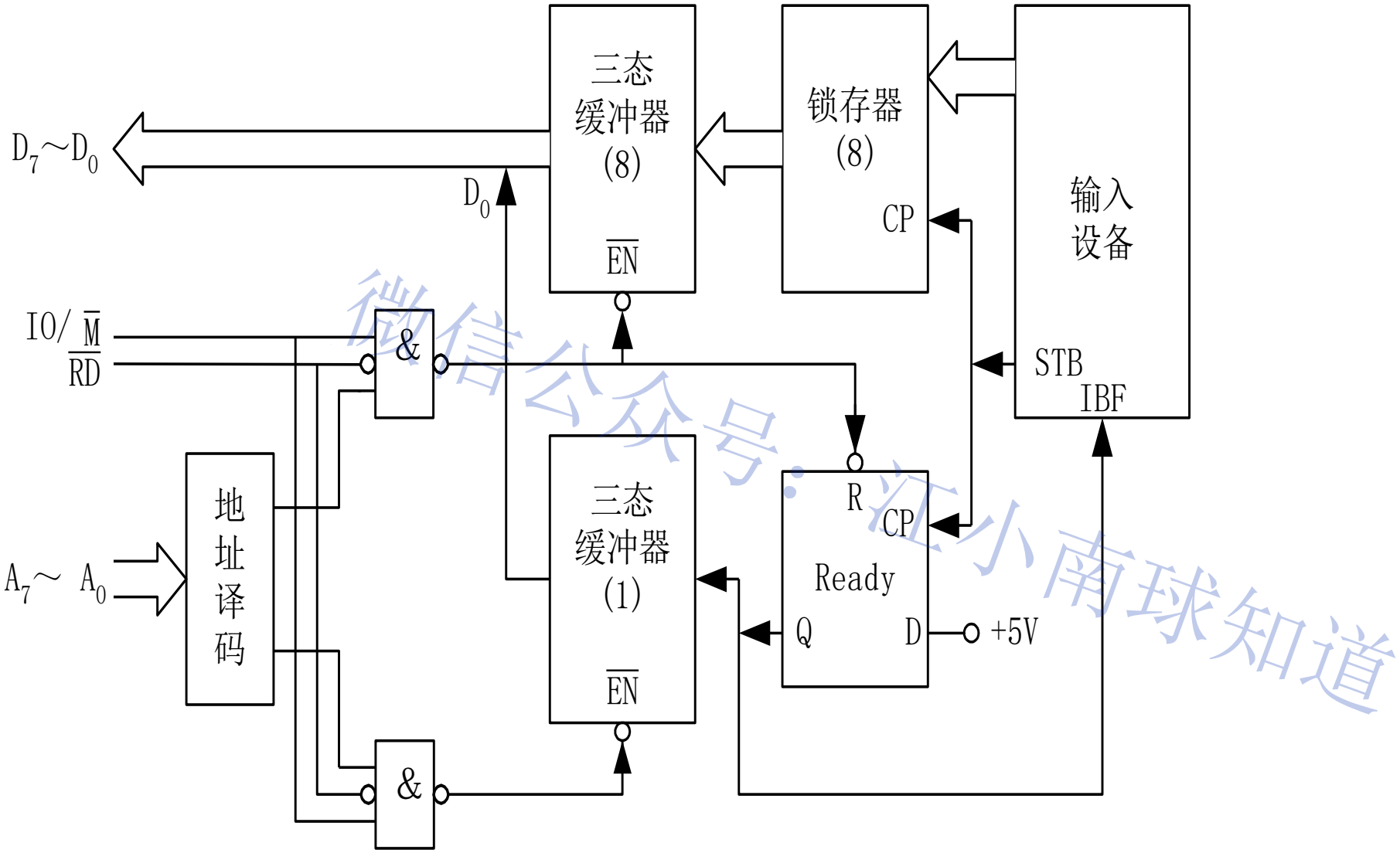
3. 程序查询方式接口

程序查询方式是最简单、经济的I/O方式，只需很少的硬件。通常接口中至少有两个寄存器，一个是数据缓冲寄存器，即数据端口，用来存放与CPU进行传送的数据信息；另一个是供CPU查询的设备状态寄存器，即状态端口，这个寄存器由多个标志位组成，其中最重要的是“外设准备就绪”标志。当CPU得到这位标志后就进行判断，以决定下一步是继续循环等待还是进行I/O传送。也有些计算机仅设置状态标志触发器，其作用与设备状态寄存器相同。

（1）输入接口

图为查询式输入接口电路，图中Ready为准备好触发器，它对应于设备状态寄存器的D_0位。

在输入设备准备好数据时，发出一个选通信号（STB），一方面将数据送入锁存器，同时将Ready触发器置“1”，以表示接口电路中已有数据（即准备就绪）。CPU要从外设输入数据时，先执行输入指令读取状态字，如Ready=1，再执行输入指令从锁存器中读取数据，同时把Ready触发器清“0”，以准备从外设接收下一个数据；如Ready=0，则踏步等待，继续读取状态字，直至Ready=1为止。

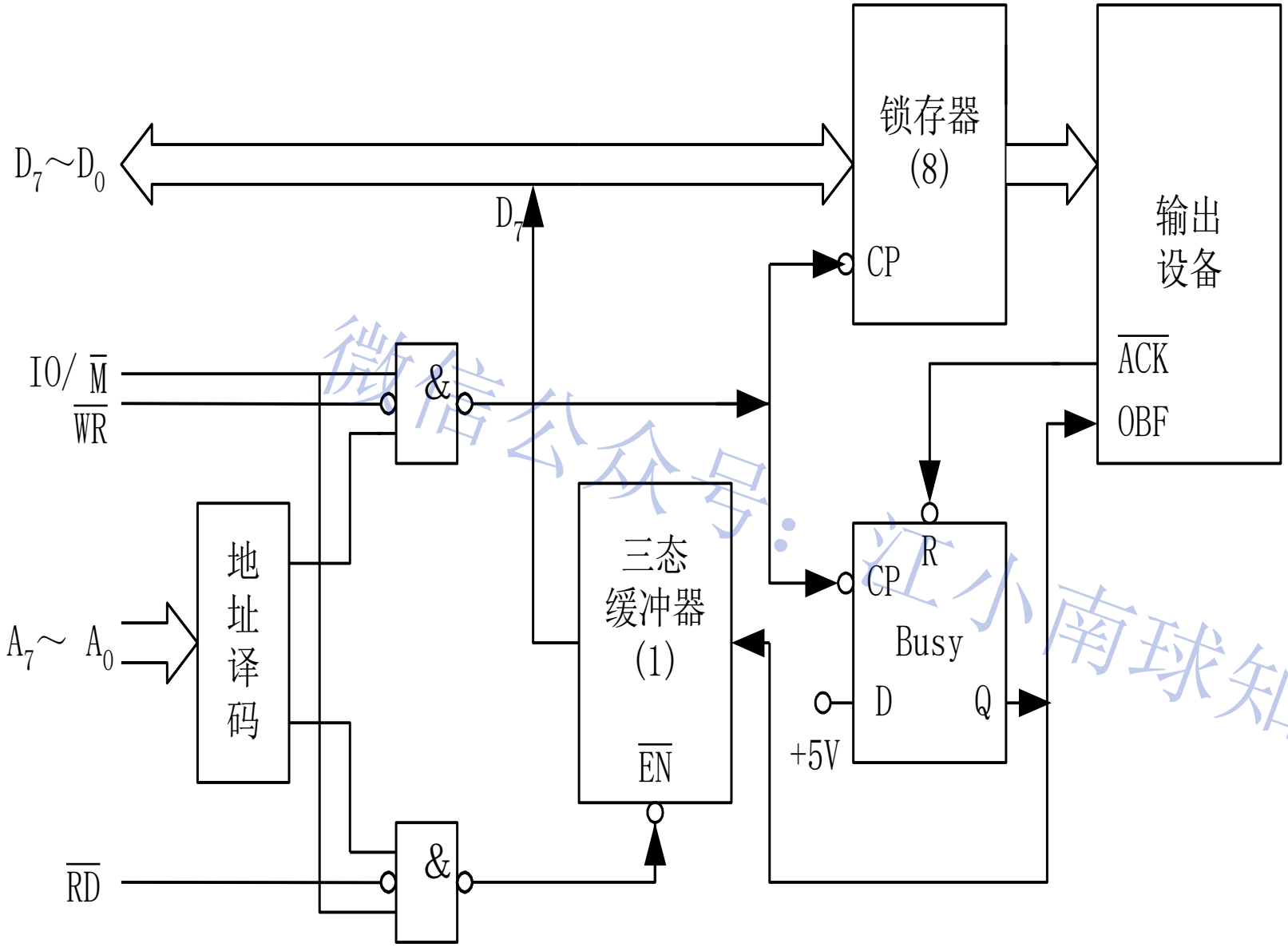


查询式输入接口电路

（2）输出接口

图为查询式**输出接口**电路，图中Busy为忙触发器，对应于设备状态寄存器的D_7位。

输出时，CPU首先执行输入指令读取状态字，如Busy=1，表示接口的输出锁存器是满的，CPU只能踏步等待，继续读取状态字，直至Busy=0为止；如Busy=0，表示接口的输出锁存器是空的，允许CPU向外设发送数据。此时，CPU执行输出指令，将数据送入锁存器，并将Busy触发器置“1”。当输出设备把CPU送来的数据真正输出之后，将发出一个 $\overline{\text{ACK}}$ 信号，使Busy触发器置“0”，以准备下一次传送。



查询式输出接口电路

程序查询方式特点：

1、何时对何设备输入/输出操作完全由CPU控制

2、外设与CPU处于异步工作方式

3、数据的输入/输出要经过CPU，至少要几条指令

4、CPU利用率低，但控制简单

5、实时性？

2、程序中断方式

中断的基本概念

中断是现代计算机有效合理地发挥效能和提高效率的一个十分重要的功能。CPU中通常设有处理中断的机构——中断系统，以解决各种中断的共性问题。本节主要分析中断系统的功能，特别强调I/O中断。

1. 中断的提出

程序查询方式虽然简单，但却存在着下列明显的**缺点**：

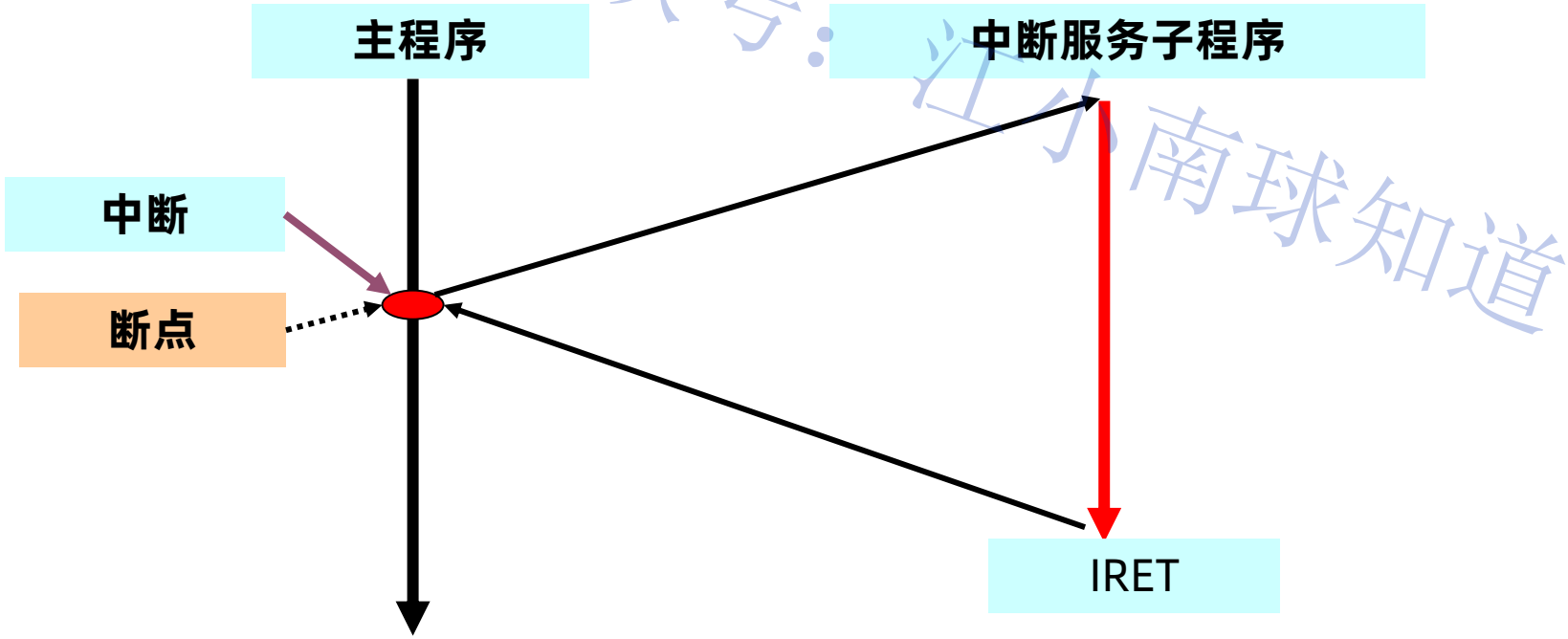
(1) 在查询过程中，CPU长期处于踏步等待状态，使系统效率大大降低；

(2) CPU在一段时间内只能和一台外设交换信息，其他设备不能同时工作；

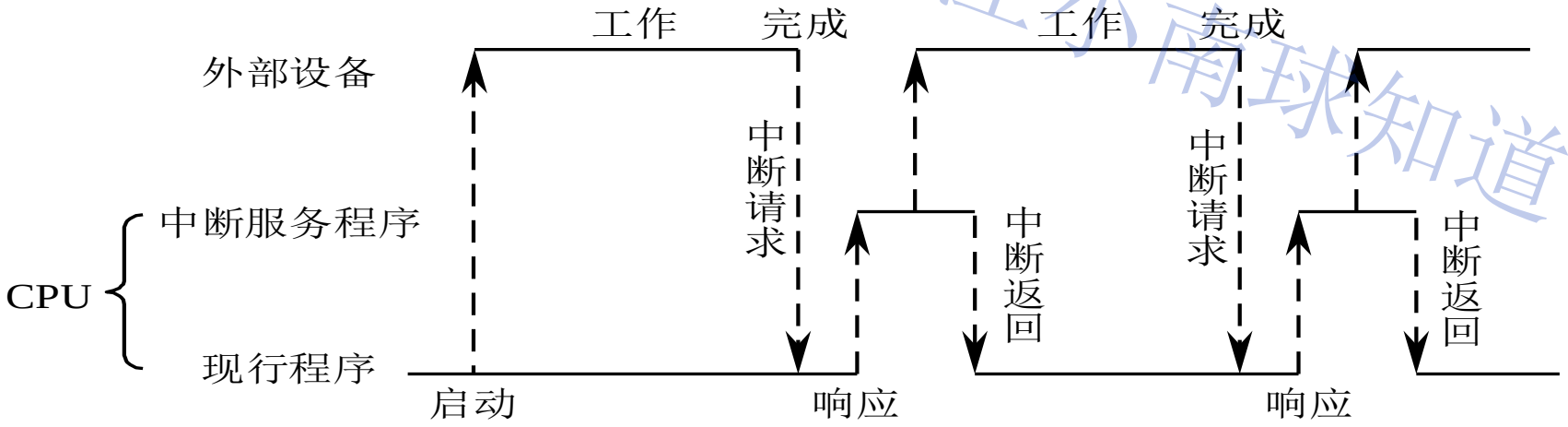
(3) 不能发现和处理预先无法估计的错误和异常情况。

2.程序中断

当CPU正常运行程序时，由于内部事件或外设请求(随机的)，引起CPU暂时中止正在运行的程序，转去执行发出请求的外设（或内部事件）的服务子程序，待该服务程序执行完毕，再返回被中止的程序，这一过程称为中断。



程序中断方式的思想是：CPU在程序中安排好在某一时刻启动某一台外设，然后CPU继续执行原来程序，不需要象查询方式那样一直等待外设的准备就绪状态。一旦外设完成数据传送的准备工作时，便主动向CPU发出一个**中断请求**，请求CPU为自己服务。在可以响应中断的条件下，CPU**暂时中止**正在执行的程序，转去执行中断服务程序为中断请求者服务，在中断服务程序中完成一次主机与外设之间的数据传送，传送完成后，CPU仍返回原来的程序，从断点处继续执行。



程序中断方式示意图

中断的处理过程实际上是程序的切换过程，即从现行程序切换到中断服务程序，再从中断服务程序返回到现行程序。CPU每次执行中断服务程序前总要保护断点、保护现场，执行完中断服务程序返回现行程序之前又要恢复现场、恢复断点。这些中断的辅助操作都将会限制数据传送的速度。

中断系统是计算机实现中断功能的软、硬件总称。一般在CPU中配置中断机构，在外设接口中配置中断控制器，在软件上设计相应的中断服务程序。

程序中断是指：计算机执行现行程序的过程中，出现某些急需处理的异常情况和特殊请求，CPU暂时中止现行程序，而转去对随机发生的更紧迫的事件进行处理，在处理完毕后，CPU将自动返回原来的程序继续执行。

3.中断的作用

“中断”是由I/O设备或其他非预期的急需处理的事件引起的，它使CPU暂时中断现在正在执行的程序，而转至另一服务程序去处理这些事件，处理完后再返回原程序。

中断有下列一些作用：

(1) CPU与I/O设备并行工作。

例CPU与打印机并行工作的时间安排。

(2)硬件故障处理。计算机运行时，如硬件出现某些故障，机器中断系统发出中断请求，CPU响应中断后自动进行处理。

(3)实现人机联系。在计算机工作过程中，如果用户要干预机器，如抽查计算中间结果，了解机器的工作状态，给机器下达临时性的命令等。在没有中断系统的机器里这些功能几乎是无法实现的。

(4)实现多道程序和分时操作。计算机实现多道程序运行是提高机器效率的有效手段。多道程序的切换运行需借助于中断系统。在一道程序的运行中，由I/O中断系统切换到另外一道程序运行。也可以通过分时分配每道程序一个固定时间片，利用时钟定时发中断进行程序切换。

(5)实现实时处理。在某个计算机过程控制系统中，当出现压力过大，温度过高等情况时，必须及时输入到计算机进行处理。这些事件出现的时刻是随机的，而不是程序本身所能预见的，因此要求计算机中断正在执行的程序，转而去执行中断服务程序

(6)实现应用程序和操作系统(管态程序)的联系。可以在用户程序中安排一条“Trap”指令进入操作系统，称之为“软中断”。其中断处理过程与其他中断类似。

(7)多处理机系统中各处理机间的联系。在多处理机系统中，处理机和处理机之间的信息交流和任务切换可以通过中断来实现。

4.中断与调用子程序的区别

从表面上看起来，计算机的中断处理过程有点类似于调用子程序的过程，这里现行程序相当于主程序，中断服务程序相当于子程序。

但有本质上的区别：

(1)子程序的执行是由程序员事先安排好的，而中断服务程序的执行则是由随机的中断事件引起的。

(2)子程序的执行受到主程序或上层子程序的控制，而中断服务程序一般与被中断的现行程序毫无关系。

(3)不存在同时调用多个子程序的情况，而可能发生多个外设同时请求CPU为自己服务的情况。

特点：CPU的利用率高

5. 中断的基本类型

（1）自愿中断和强迫中断

自愿中断又称程序自中断，它不是随机产生的中断，而是在程序中安排的有关指令，这些指令可以使机器进入中断处理的过程，如：80x86指令系统中的软中断指令INT n。

强迫中断是随机产生的中断，不是程序中事先安排好的。当这种中断产生后，由中断系统强迫计算机中止现行程序并转入中断服务程序。

（2）内中断和外中断

内中断是指由于CPU内部硬件或软件原因引起的中断，如单步中断、溢出中断等。

外中断是指CPU以外的部件引起的中断。通常，外中断又可以分为**不可屏蔽中断和可屏蔽中断**两种。不可屏蔽中断优先级别较高，常用于应急处理，如掉电、主存读写校验错等；而可屏蔽中断级别较低，常用于一般I/O设备的数据传送。

（3）向量中断和非向量中断

向量中断是指那些中断服务程序的入口地址是由中断事件自己提供的中断。中断事件在提出中断请求的同时，通过硬件向主机提供中断服务程序入口地址，即向量地址。

非向量中断的中断事件不能直接提供中断服务程序的入口地址。

（4）单重中断和多重中断

单重中断在CPU执行中断服务程序的过程中不能被再打断。

多重中断在执行某个中断服务程序的过程中，CPU可去响应级别更高的中断请求，又称为**中断嵌套**。多重中断表征计算机中断功能的强弱，有的计算机能实现8级以上的多重中断。

6.中断请求和中断判优

（1）中断源和中断请求信号

中断源是指中断请求的来源，即引起计算机中断的事件。通常，一台计算机允许有多个中断源。由于每个中断源向CPU发出中断请求的时间是随机的，为了记录中断事件并区分不同的中断源，可采用具有存储功能的触发器来记录中断源，这个触发器称为**中断请求触发器**（INTR）。当某一个中断源有中断请求时，其相应的中断请求触发器置成“1”状态，表示该中断源向CPU提出中断请求。

（2）中断请求信号的传送

① 独立请求线

每个中断源单独设置中断请求线，将中断请求信号直接送往CPU。

② 公共请求线

多个中断源共有一根公共请求线。

③ 二维结构

将中断请求线连成二维结构，同一优先级别的中断源，采用一根公共的请求线；不同请求线上的中断源优先级别不同。

（3）中断优先级与判优方法

通常，把全部中断源按中断的性质和处理的轻重缓急安排优先级，并进行排队。

确定中断优先级的原则是：对那些提出中断请求后需要立刻处理，否则就会造成严重后果的中断源规定最高的优先级；而对那些可以延迟响应和处理的中断源规定较低的优先级。如故障中断一般优先级较高，其次是简单中断，接着才是I/O设备中断。

① 软件判优法

所谓软件判优法，就是用程序来判别优先级，这是最简单的中断判优方法。当CPU接到中断请求信号后，就执行查询程序，逐个检测中断请求寄存器的各位状态。检测顺序是按优先级的大小排列的，最先检测的中断源具有最高的优先级，其次检测的中断源具有次高优先级，如此下去，最后检测的中断源具有最低的优先级。

软件判优方法简单，可以灵活地修改中断源的优先级别；但查询、判优完全是靠程序实现的，不但占用CPU时间，而且判优速度慢。

② 硬件判优电路

采用硬件判优电路实现中断优先级的判定可节省CPU时间，而且速度快，但是成本较高。

根据中断请求信号的传送方式不同，有不同的优先排队电路，常见的方案有：独立请求线的优先排队电路、公共请求线的优先排队电路等。这些排队电路的共同特点是：优先级别高的中断请求将自动封锁优先级别低的中断请求的处理。硬件排队电路一旦设计连接好之后，将无法改变其优先级别。

7、中断响应和中断处理

1. CPU响应中断的条件

(1) CPU接收到中断请求信号

首先中断源要发出中断请求，同时CPU还要接收到这个中断请求信号。

(2) CPU允许中断

CPU允许中断，即开中断。CPU内部有一个中断允许触发器（EINT），只有当EINT=1时，CPU才可以响应中断源的中断请求；如EINT=0，CPU处于不允许中断状态，即使中断源有中断请求，CPU也不响应。

通常，中断允许触发器由开中断指令来置位，由关中断指令或硬件自动使其复位。

(3) 一条指令执行完毕

这是CPU响应中断请求的时间限制条件。一般情况下，CPU在一条指令执行完毕且没有更紧迫的任务时才能响应中断请求。

2. 中断隐指令

CPU响应中断之后，经过某些操作，转去执行中断服务程序。这些操作是由硬件直接实现的，我们把它称为中断隐指令。**中断隐指令并不是指令系统中的一条真正的指令，它没有操作码，所以中断隐指令是一种不允许、也不可能为用户使用的特殊指令**。

中断隐指令完成的操作主要有：

(1)保存断点

为了保证在中断服务程序执行完毕能正确返回原来的程序，必须将原来程序的断点（即程序计数器PC的内容）保存起来。断点可以压入堆栈，也可以存入主存的特定单元中。

(2)暂不允许中断

暂不允许中断即关中断。在中断服务程序中，为了保护中断现场（即CPU的主要寄存器状态）期间不被新的中断所打断，必须要关中断，从而保证被中断的程序在中断服务程序执行完毕之后能接着正确地执行下去。

(3)引出中断服务程序

引出中断服务程序实质是取出**中断服务程序的入口地址**送程序计数器PC。对于向量中断和非向量中断，引出中断服务程序的方法是不相同的。

3. 中断周期

以上几个基本操作在不同的计算机系统中的处理方法是各异的。通常，在**组合逻辑控制**的计算机中，专门设置了一个**中断周期**来完成中断隐指令的任务。在**微程序控制**的计算机中，则专门安排有**一段微程序**来完成中断隐指令的这些操作。

4. 进入中断服务程序

软件的方法由中断隐指令控制进入一个中断总服务程序，在那里判优、寻找中断源并且转入相应的中断服务程序。这种方法方便、灵活，硬件极简单，但效率是比较低的。

对于**硬件向量中断法**，当CPU响应某一中断请求时，硬件能自动形成并找出与该中断源对应的中断服务程序的入口地址。

当中断源向CPU发出中断请求信号之后，CPU进行一定的判优处理。若决定响应这个中断请求，则向中断源发出中断响应信号INTA。中断源接到INTA信号后就通过自己的向量地址形成部件向CPU发送向量地址，CPU接收该向量地址之后就可转入相应的中断服务程序。

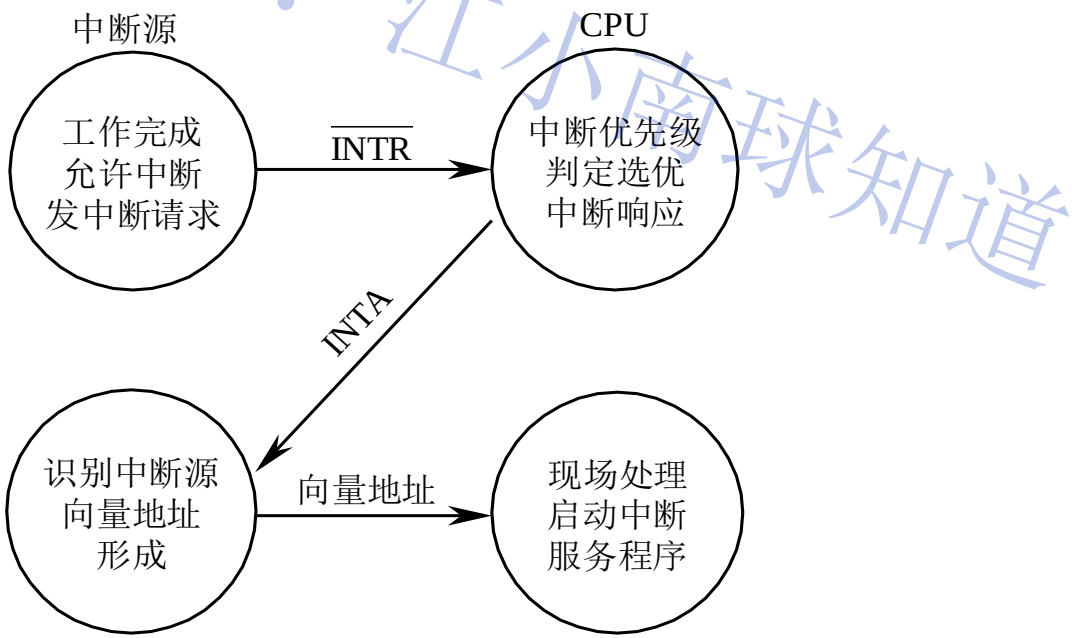
向量地址通常有两种情况：

(1)向量地址是中断服务程序的入口地址

如果向量地址就是中断服务程序的入口地址，则CPU不需要再经过处理就可以进入相应的中断服务程序。

(2)向量地址是中断向量表的指针

如果向量地址是中断向量表的指针，则向量地址指向一个中断向量表，从中断向量表的相应单元中再取出中断服务程序的入口地址，此时中断源给出的向量地址是中断服务程序入口地址的地址。



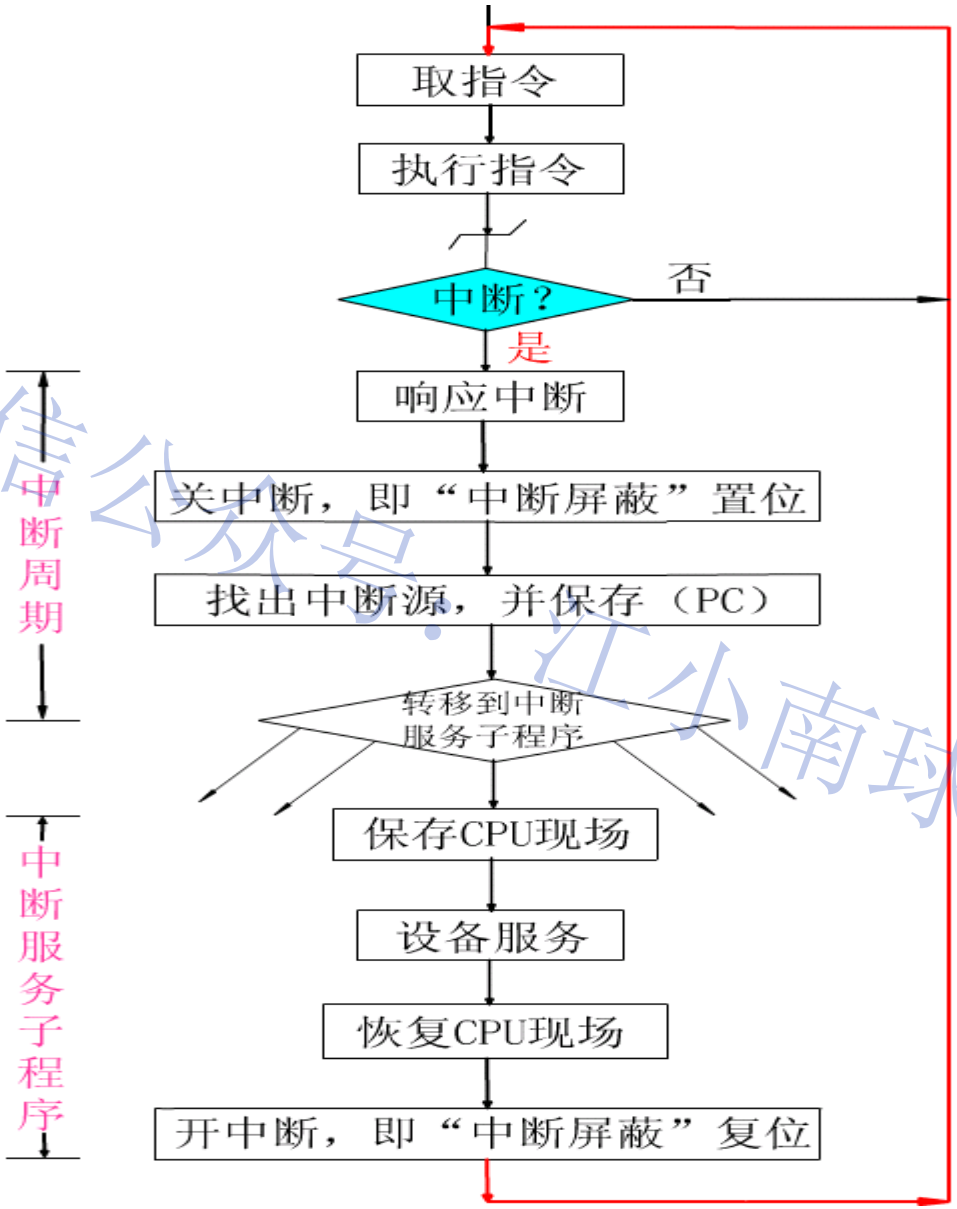
5. 中断现场的保护和恢复

中断现场指的是发生中断时CPU的主要状态，其中最重要的是**断点**，另外还有一些**通用寄存器**的状态。之所以需要保护和恢复现场的原因是因为CPU要先后执行两个完全不同的程序，必须进行两种程序运行状态的转换。一般来说，在中断隐指令中，CPU硬件将自动保存断点，有些计算机还自动保存程序状态寄存器的内容。但是，在许多应用中，要保证中断返回后原来的程序能正确地继续运行，仅保存这一、二个寄存器的内容是不够的。

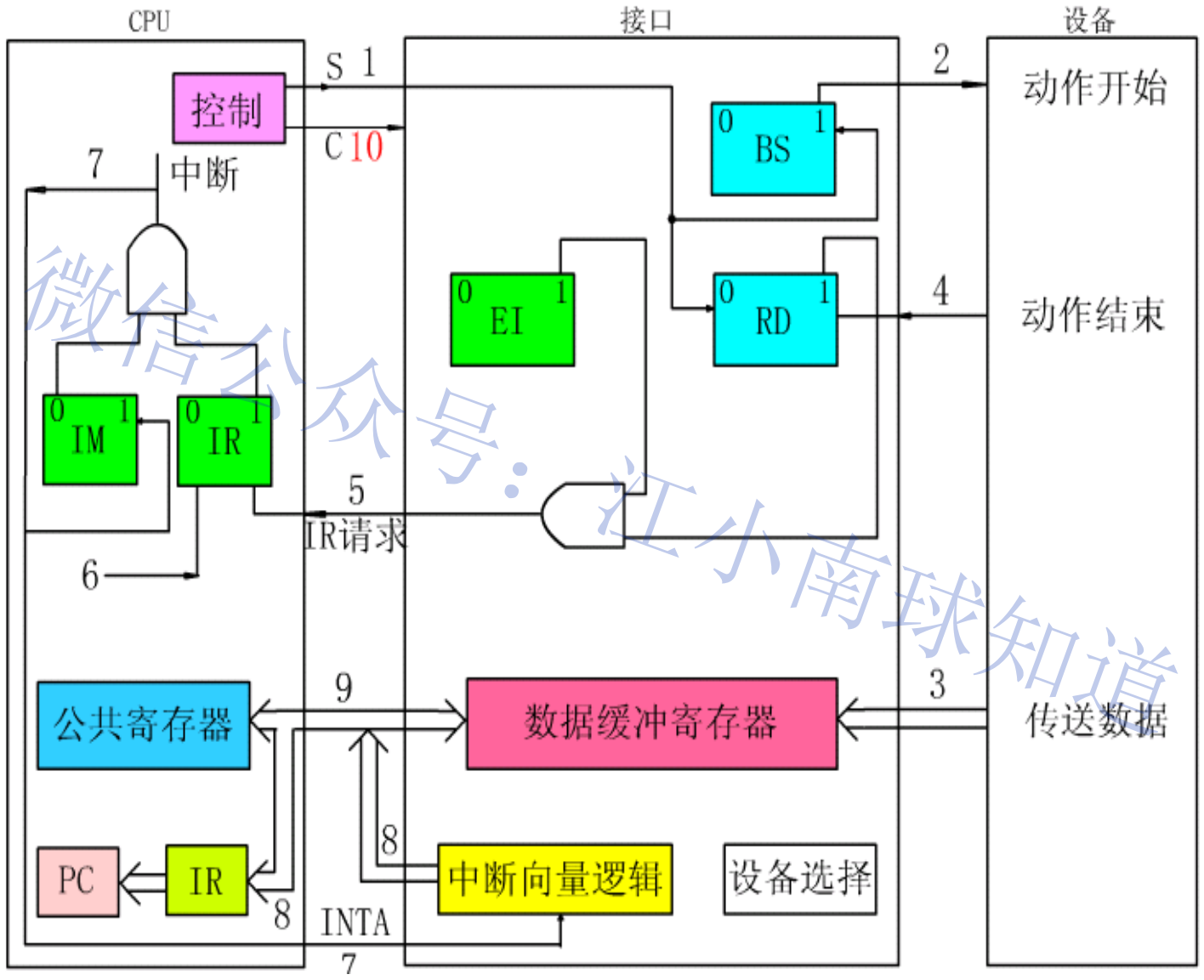
在中断服务程序开始时，应由软件去保存那些硬件没有保存，而在中断服务程序中又可能用到的寄存器的内容，在中断返回之前，这些内容还应该被恢复。

软、硬件保护现场往往是和向量中断结合在一起使用的。先把断点和程序状态字自动压入堆栈，这就是保护旧现场；接着根据中断源送来的向量地址自动取出中断服务程序入口地址和新的程序状态字，这就是建立新现场；最后由一些指令实现对必要的通用寄存器的保护。

中断处理过程流程图



程序中断方式的基本接口



程序中断由外设接口的状态和CPU两方面来控制：

在接口方面，有决定是否向CPU发出中断请求的机构，主要是接口中的“准备就绪”标志(RD)和“允许中断”标志(EI)两个触发器；

在CPU方面，有决定是否受理中断请求的机构，主要是“中断请求”标志(IR)和“中断屏蔽”标志(IM)两个触发器。

上述四个标志触发器的具体功能如下：

准备就绪的标志(RD) 一旦设备做好一次数据的接收或发送，便发出一个设备动作完毕信号，使RD标志置“1”。在中断方式中，该标志用作为中断源触发器，简称中断触发器。

允许中断触发器(EI) 可以用程序指令来置位。EI为“1”时，某设备可以向CPU发出中断请求；EI为“0”时，不能向CPU发出中断请求，这意味着某中断源的中断请求被禁止。设置EI标志的目的，就是通过软件来控制是否允许某设备发出中断请求。

中断请求触发器(IR) 它暂存中断请求线上由设备发出的中断请求信号。当IR标志为“1”时，表示设备发出了中断请求。

中断屏蔽触发器(IM) 是CPU是否受理中断或批准中断的标志。IM标志为“0”时，CPU可以受理外界的中断请求，反之，IM标志为“1”时，CPU不受理外界的中断。

6.单级中断和多级中断

1.单级中断的概念

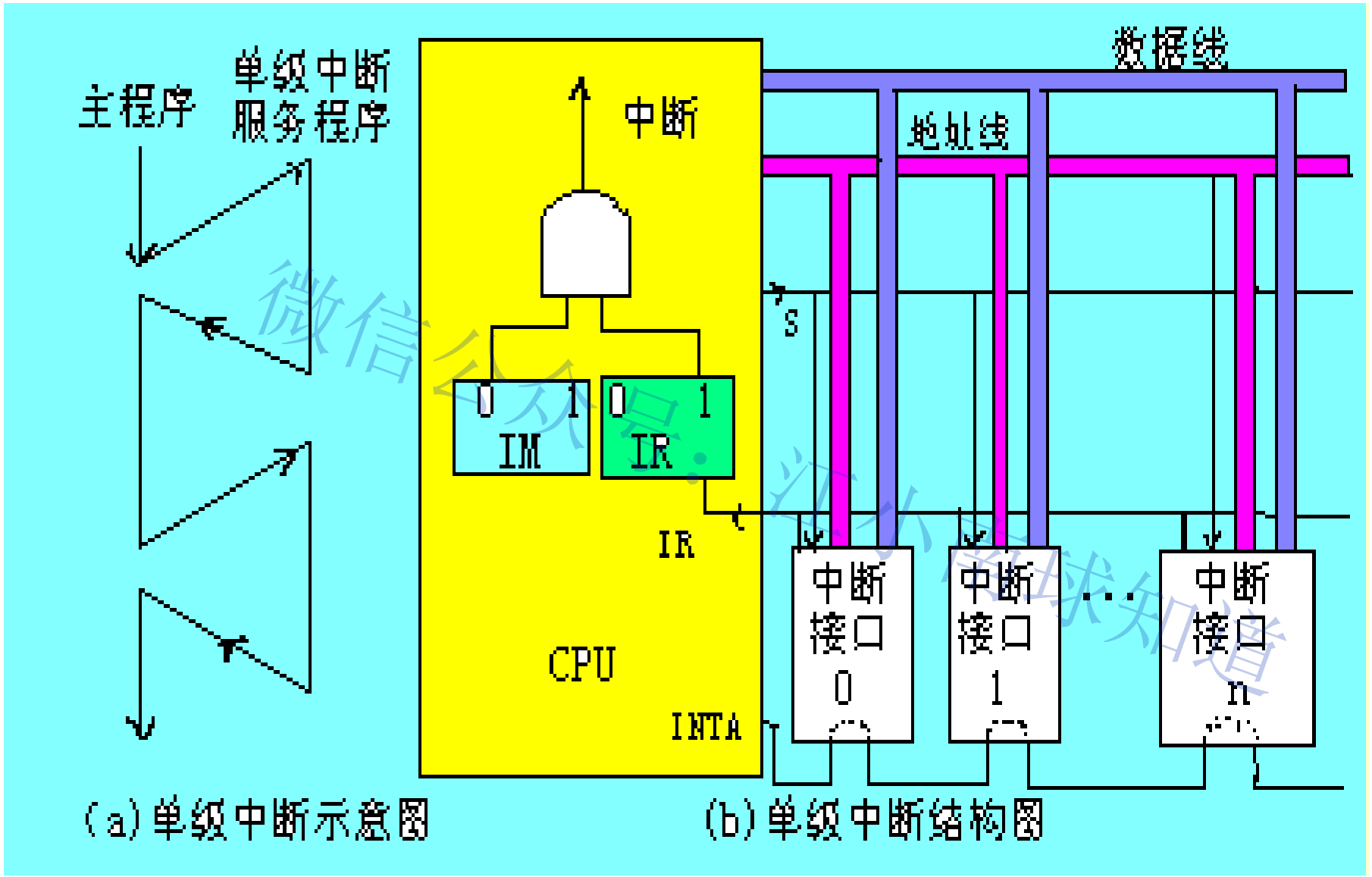
根据计算机系统对中断处理的策略不同，可分为**单级中断系统**和**多级中断系统**。

单级中断系统是中断结构中最基本的形式。在单级中断系统中，所有的**中断源都属于同一级**，所有中断源触发器排成一行，其优先次序是离CPU近的优先权高。当**响应某一中断请求时，执行该中断源的中断服务程序。**在此过程中，不允许其他中断源再打断中断服务程序，即使优先权比它高的中断源也不能再打断。参照后面的单级中断示意图和系统结构图。

2.单级中断源的识别

如何确定中断源，并转入被响应的中断服务程序入口地址，是中断处理首先要解决的问题。

单级中断中，采用**串行排队链法**来实现具有公共请求线的中断源判优识别。


主程序
单级中断
服务程序
中断
数据线
地址线
S
0 1
IM
0 1
IR
IR
CPU
INTA
中断
接口
0
中断
接口
1
中断
接口
n
(a)单级中断示意图
(b)单级中断结构图

3.中断向量的产生

开关理论中把若干个布尔量排成的序列定义为**布尔向量**。由于存储器的地址码是一串布尔量的序列，因此常常把地址码称为**向量地址**。

当CPU响应中断时，由硬件直接产生一个固定的地址(即向量地址)，由向量地址指出每个中断源设备的中断服务程序入口，这种方法通常称为**向量中断**。显然，每个中断源分别有一个中断服务程序，而每个中断服务程序又有自己的向量地址。当CPU识别出某中断源时，由硬件直接产生一个与该中断源对应的向量地址，很快便引入中断服务程序。向量中断要求在硬件设计时考虑所有中断源的向量地址，而实际中断时只能产生一个向量地址。

有些计算机中**由硬件产生的向量地址不是直接地址，而是一个“位移量”**，这个位移量加上CPU某寄存器里存放的基地址，最后得到中断处理程序的入口地址。

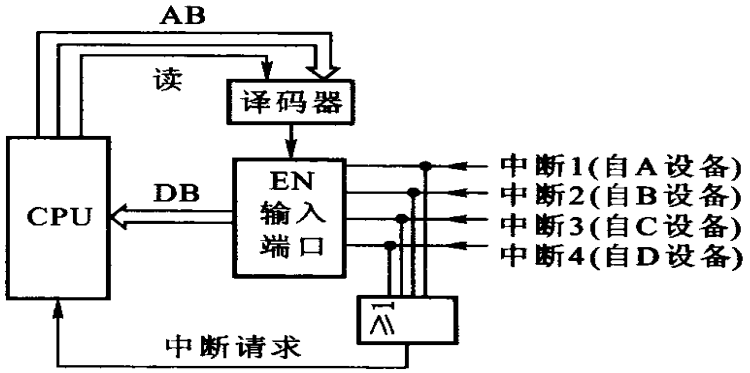
还有一种采用**向量地址转移**的方法。假设有8个中断源，由优先级编码电路产生8个对应的固定地址码(例如0，1，2，...，7)，这8个单元中存放的是转移指令，通过转移指令可转入设备各自的中断服务程序入口。这种方法允许中断处理程序放在内存中任何地方，非常灵活。

识别中断源

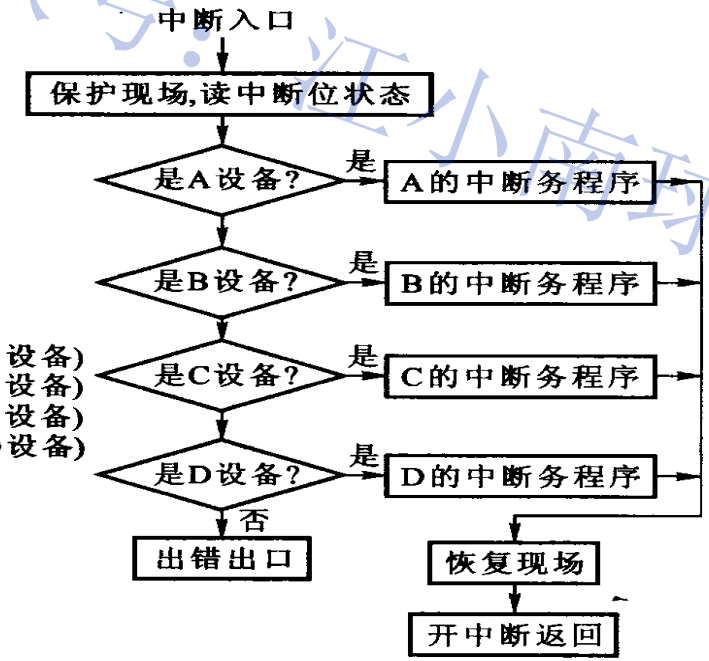
(1)查询法 由测试程序按一定优先排队次序检查各个设备的“中断触发器”(或称为中断标志)，当遇到第一个“1”标志时，即找到了优先进行处理的中断源，通常取出其设备码，根据设备码转入相应的中断服务程序。

中断优先级的判断

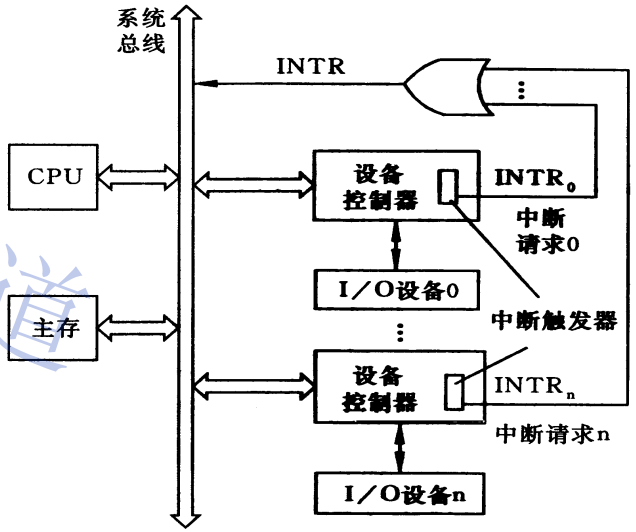
软件查询——依优先次序查询



(a)



(b)



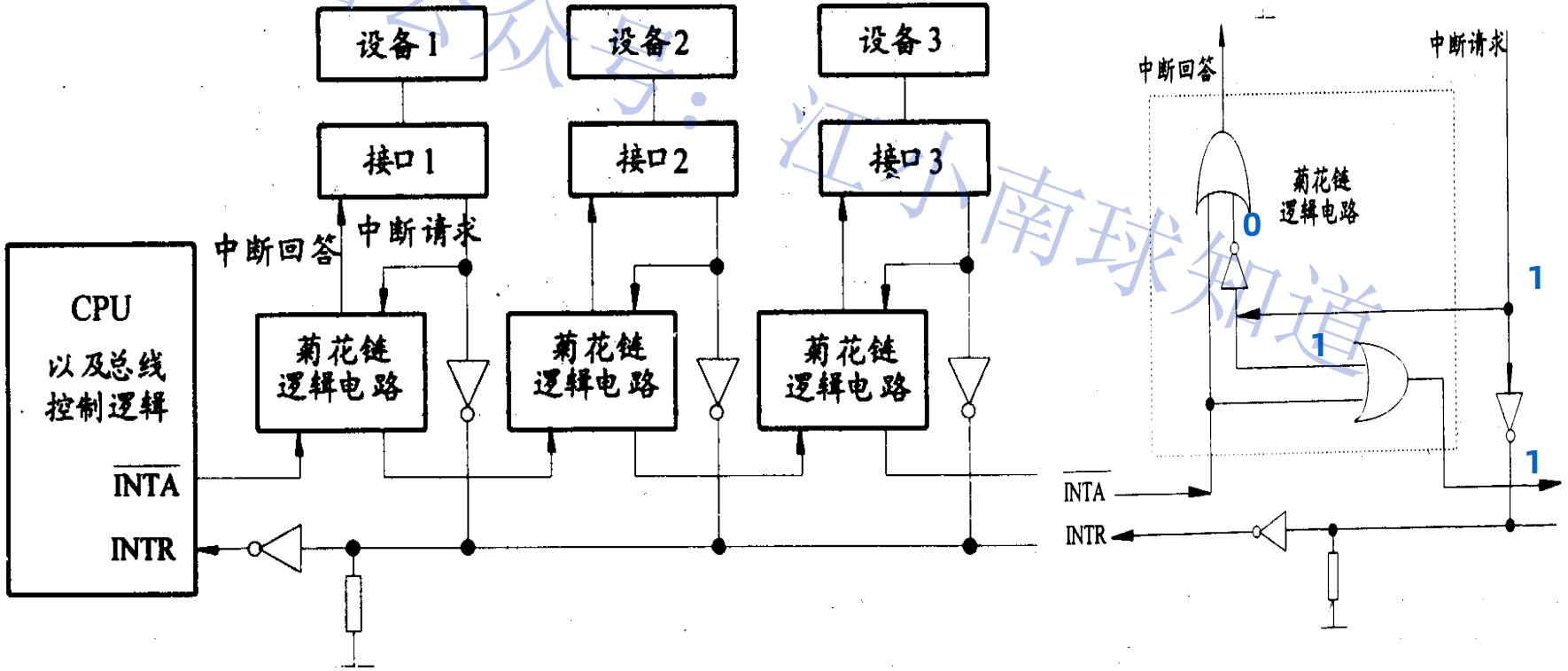
(a) 中断请求逻辑

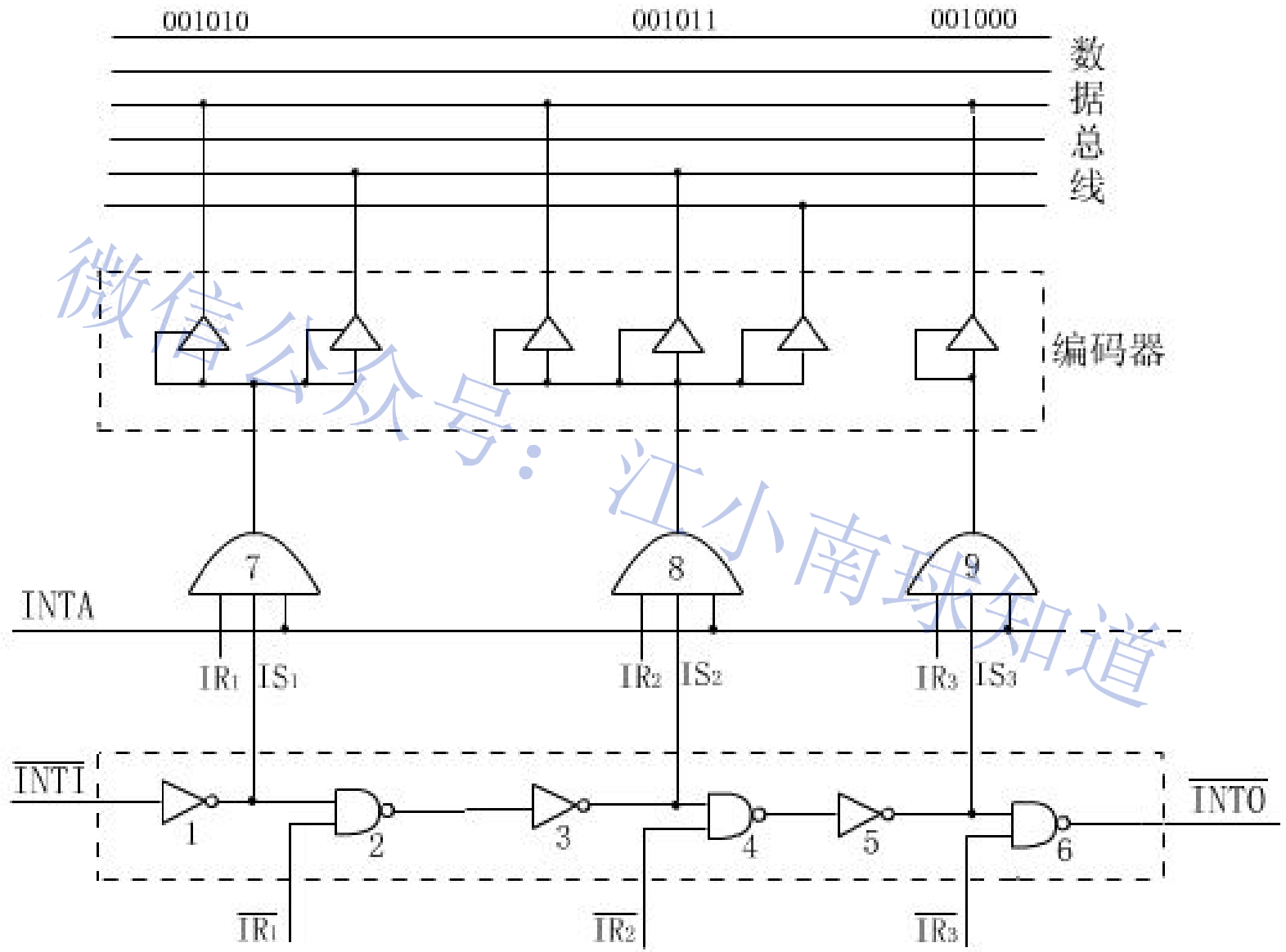
(2)串行排队链法　由硬件确定中断源，当任一设备的中断触发器为“1”时，向CPU发出中断请求信号INTR

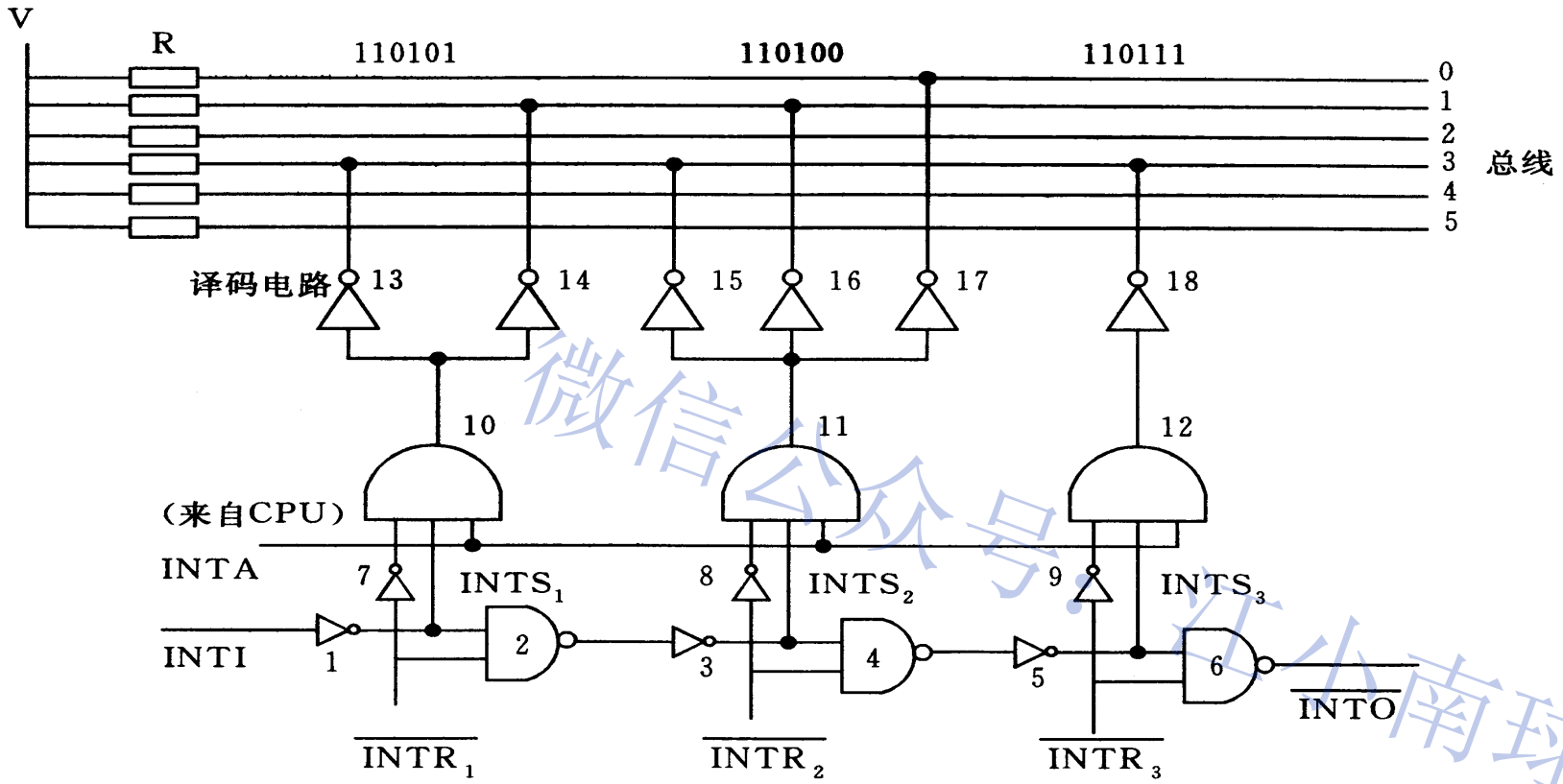
中断优先级的判断

硬件方式——如：菊花链

中断请求=1







(b) 串行排队逻辑

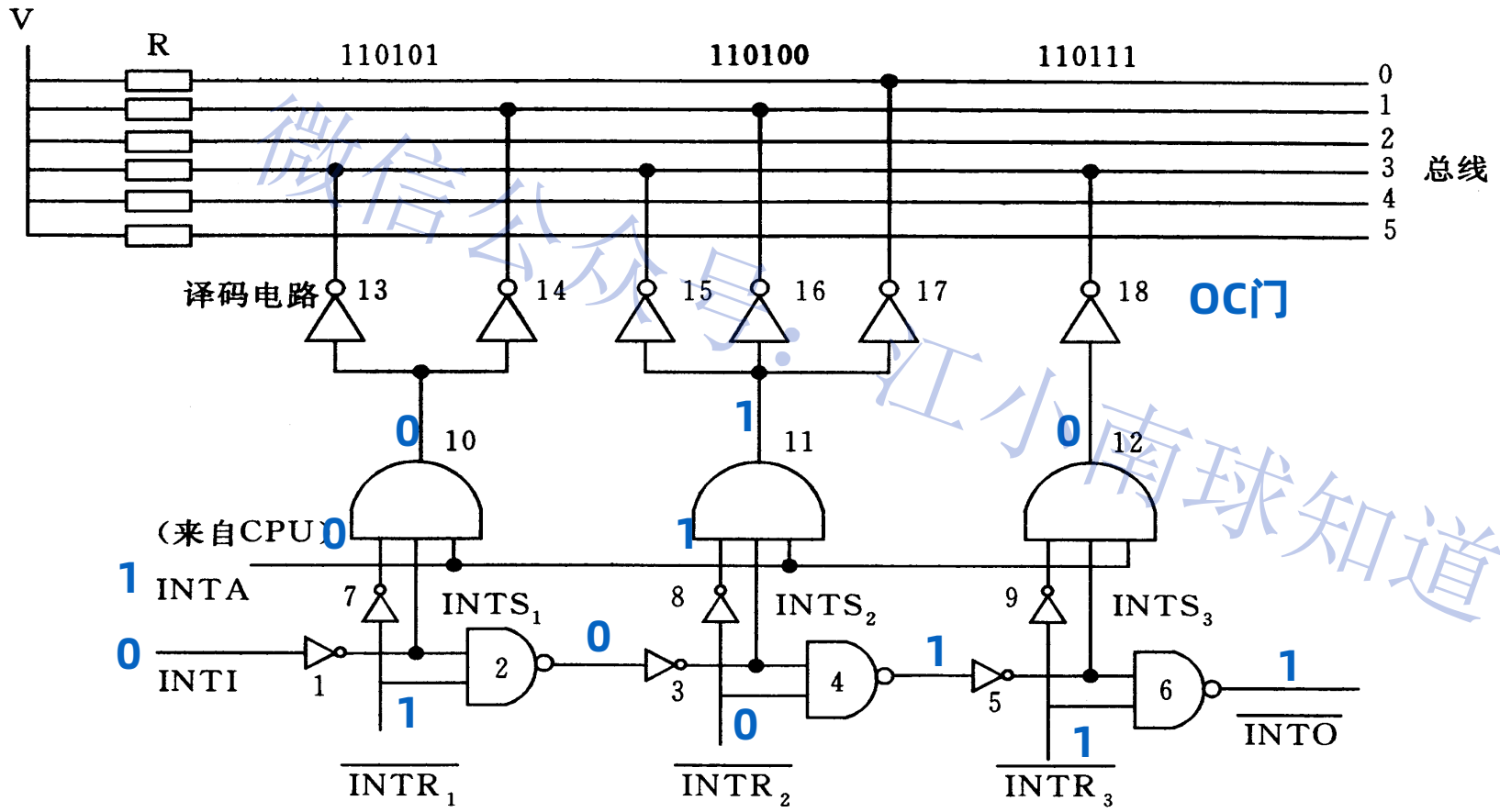
INTA是由CPU~~送来~~的取中断设备码信号，高有效。

由CPU送来的INTI为中断排队输入，低有效。

~~从各~~设备来的中断请求信号，优先顺序从高到低，依次是$\overline{INTR_1}$，$\overline{INTR_2}$，$\overline{INTR_3}$，低有效。

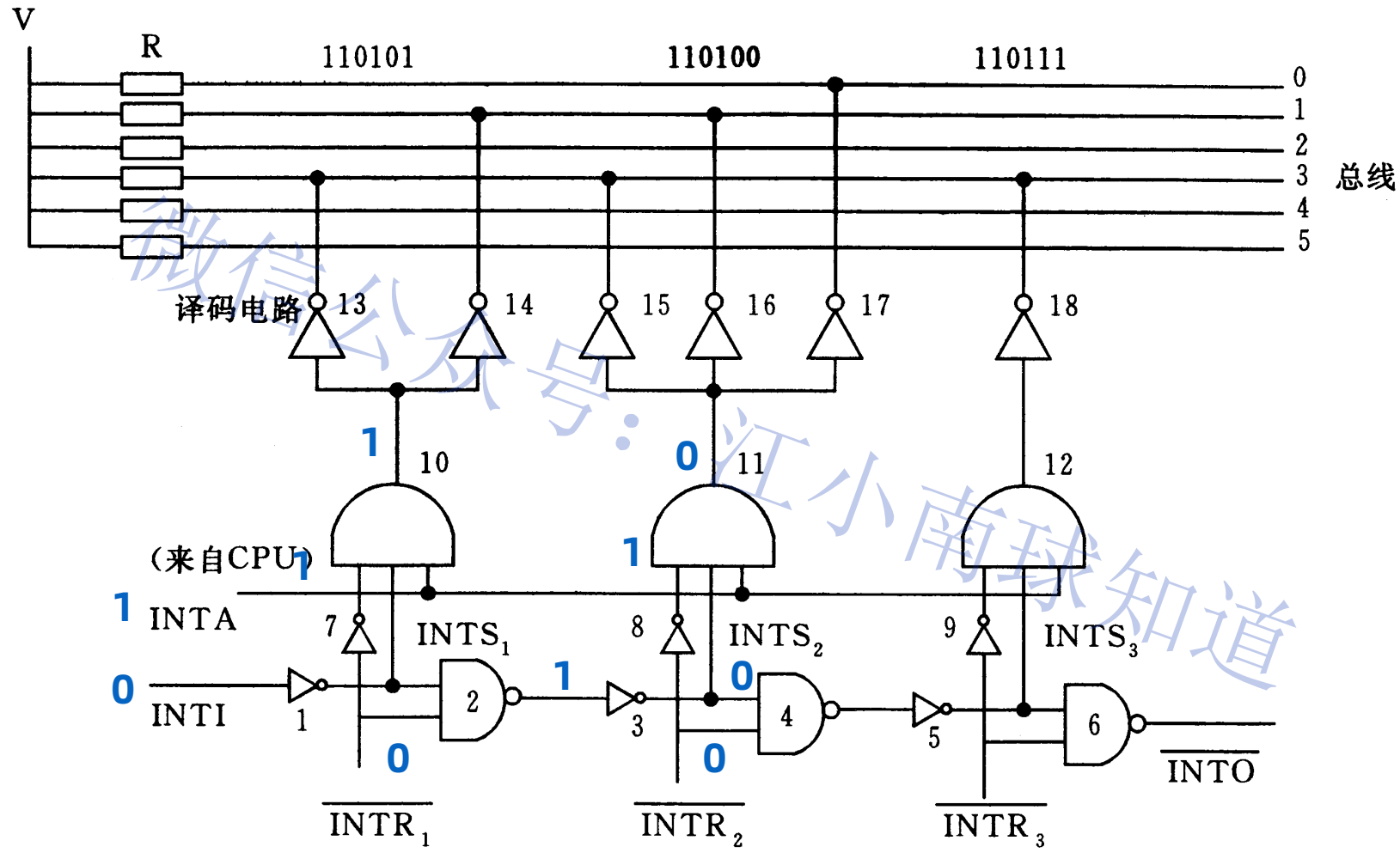
（1）仅 $\overline{INTR_2}$低有效，即设备2有中断请求。

上半部分是一个编码电路，它将产生请求中断的设备中优先权最高的设备码经总线送往CPU。



（b）串行排队逻辑

（2） $\overline{INTR_1}$ $\overline{INTR_2}$都为低，即设备1、设备2同时有中断请求。

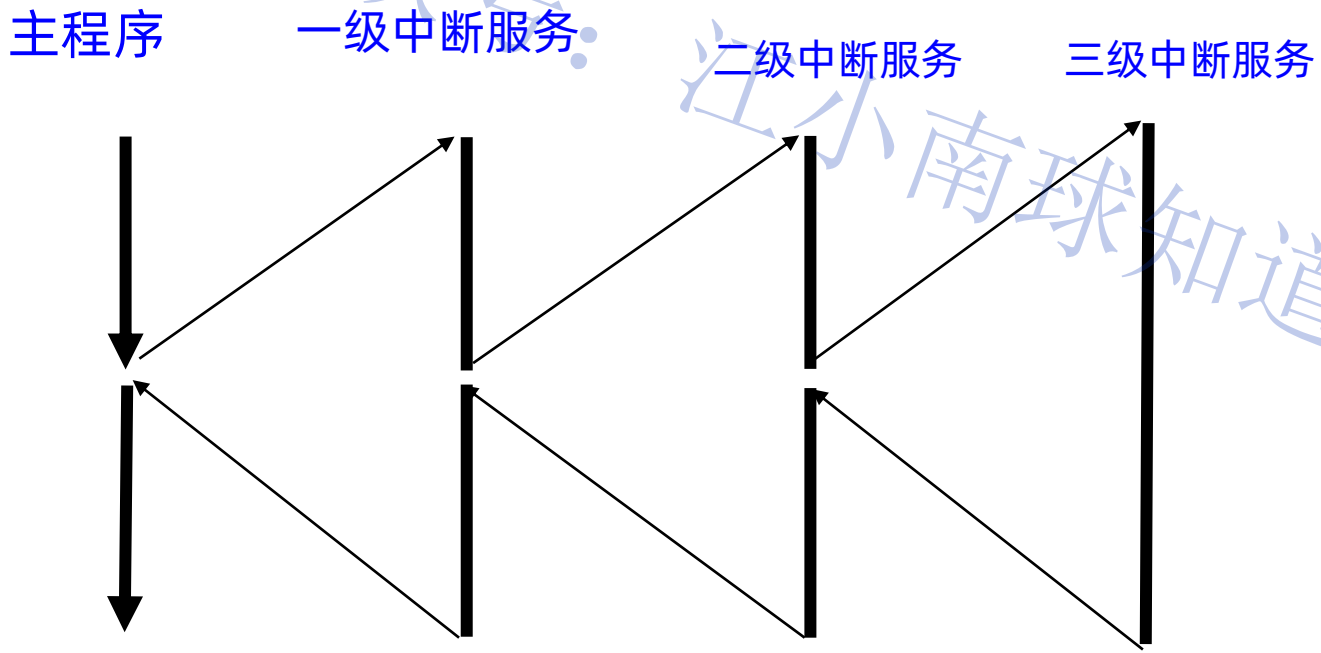


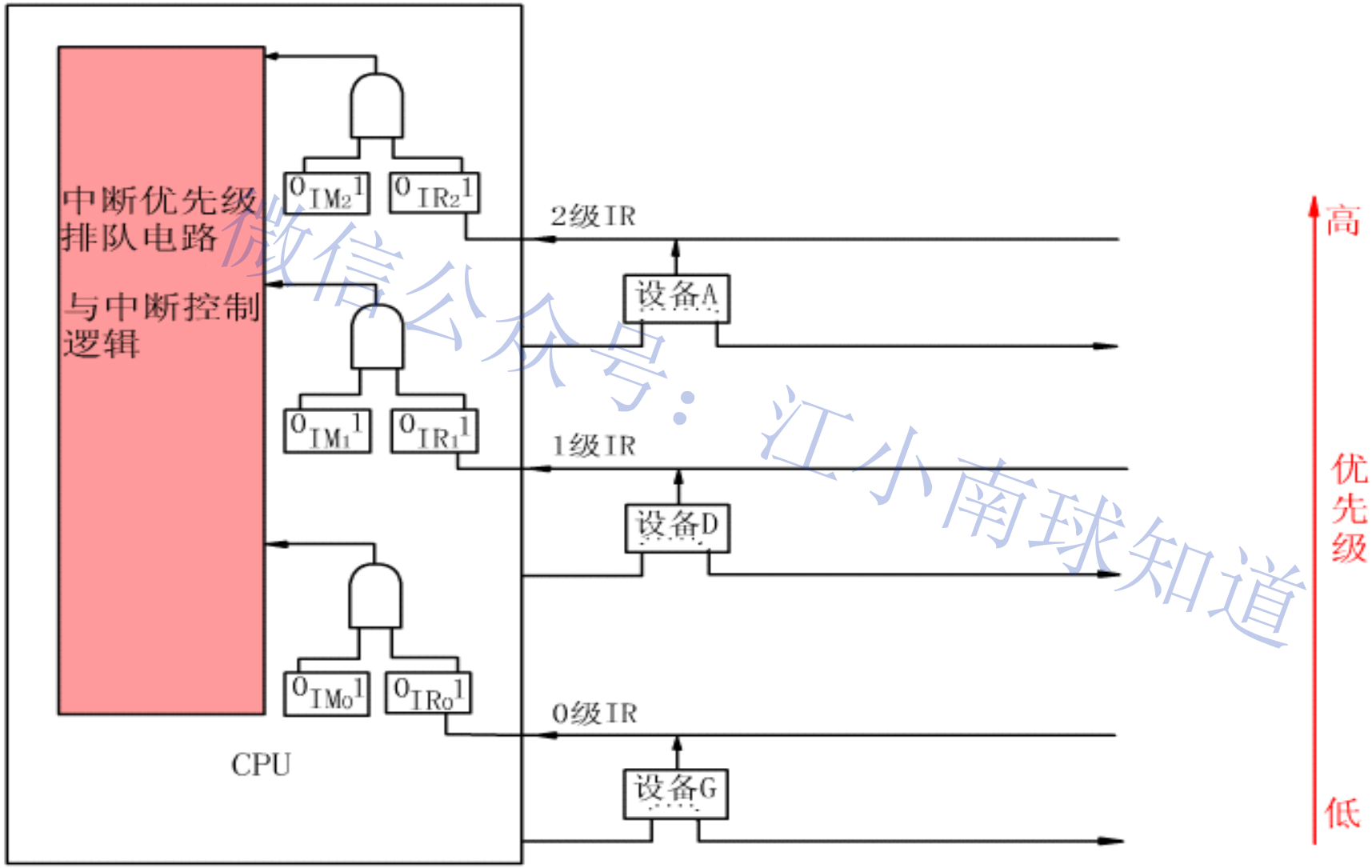
（b）串行排队逻辑

4.多级中断

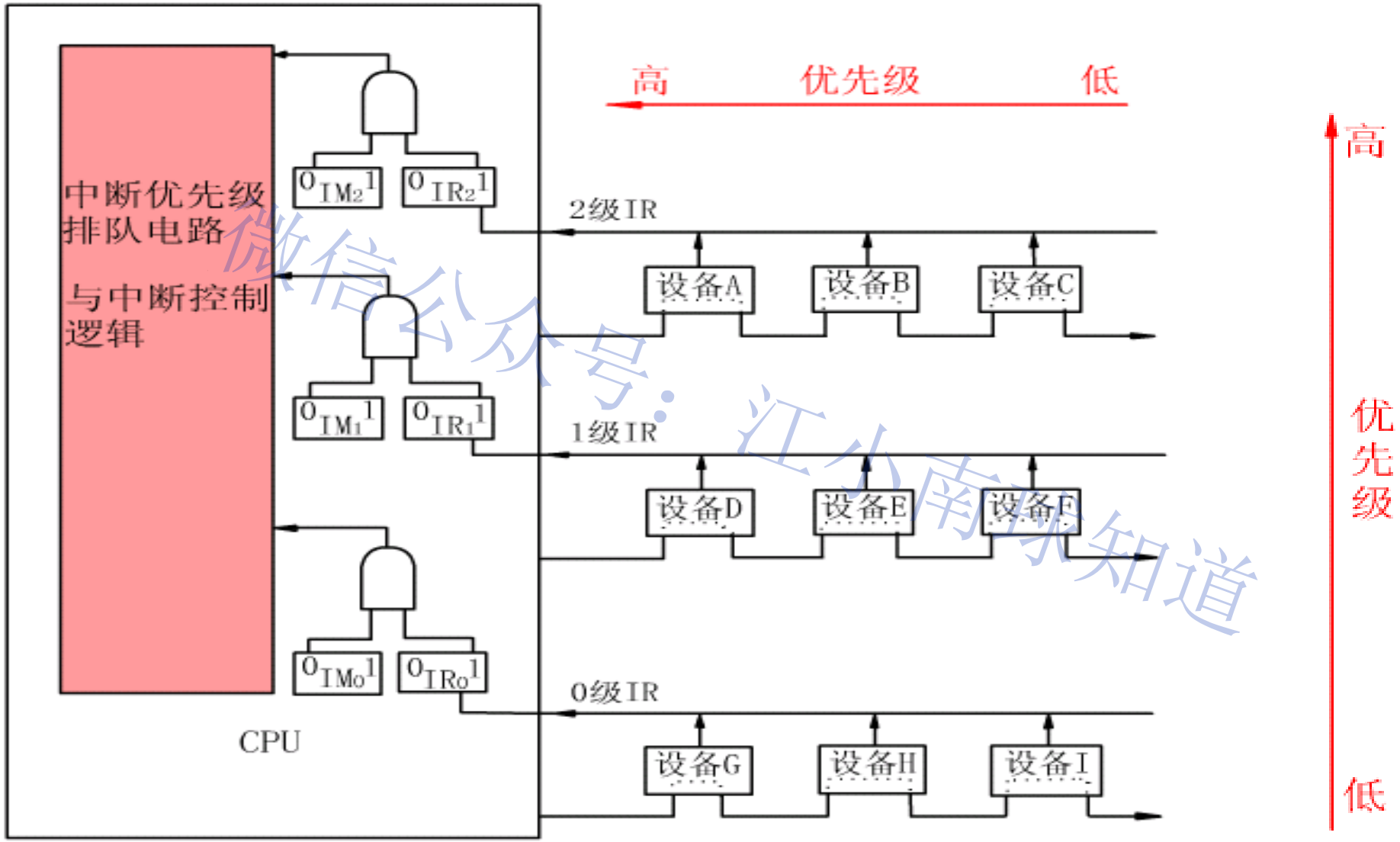
（1）多级中断的概念

多级中断系统是指计算机系统中有相当多的中断源，根据各中断事件的轻重缓急程度不同而分成若干级别，每一中断级分配给一个优先权。优先权高的中断级可以打断优先权低的中断服务程序，以程序嵌套方式进行工作，。





一维多级中断结构



二维多级中断结构

一维多级中断是指每一级中断里只有一个中断源，而**二维多级中断**是指每一级中断里又有多个中断源。

对多级中断，我们着重说明如下几点：

① 一个系统若有n级中断，在CPU中就有n个中断请求触发器，总称为**中断请求寄存器**；与之对应的有n个中断屏蔽触发器，总称为**中断屏蔽寄存器**。与单级中断不同，在多级中断中，中断屏蔽寄存器的内容是一个很重要的程序现场，因此在响应中断时，需要把中断屏蔽寄存器的内容保存起来，并设置新的中断屏蔽状态。一般在某一级中断被响应后，要置“1”(关闭)本级和优先权低于本级的中断屏蔽触发器，置“0”(开放)更高级的中断屏蔽触发器，以此来实现正常的中断嵌套。

② 多级中断中的每一级可以只有一个中断源，也可以有多个中断源。在多级中断之间可以实现中断嵌套，但是同一级内有不同中断源的中断是不能嵌套的，必须是处理完一个中断后再响应和处理同一级内其他中断源。

③ 设置多级中断的系统一般都希望有较快的中断响应时间，因此首先响应哪一级中断和哪一个中断源，都是由硬件逻辑实现，而不是用程序实现。另外，在二维中断结构中，除了有中断优先级排队电路确定优先响应中断级外，还要确定优先响应的中断源，一般通过链式查询的硬件逻辑来实现。显然，这里采用了独立请求方式与链式查询方式相结合的方法决定首先响应哪个中断源。

④ 和单级中断情况类似，在多级中断中也使用堆栈保存现场信息。使用堆栈保存现场的好处是：

a.控制逻辑简单，保存和恢复现场的过程按先进后出顺序进行。

b.每一级中断不必单独设置现场保护区，各级中断现场可按其顺序放在同一个栈里。

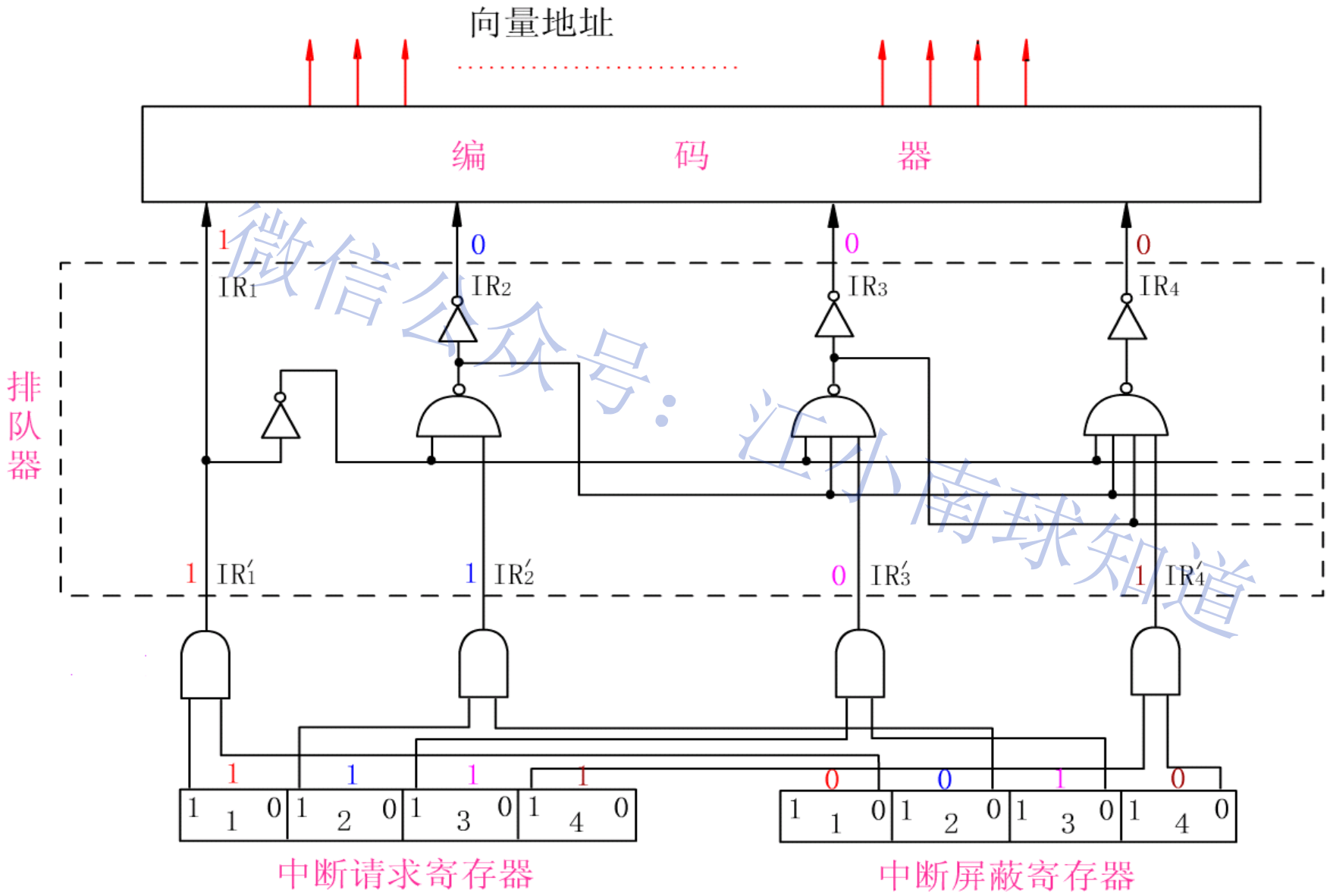
（2）多级中断源的识别

在多级中断中，每一级均有一根中断请求线送往CPU的**中断优先级排队电路**，对每一级赋予了不同的优先级。显然这种结构就是**独立请求方式**的逻辑结构。

每个中断请求信号保存在“中断请求”触发器中，经“中断屏蔽”触发器控制后，可能有若干个中断请求信号IR_i'进入虚线框所示的排队电路。排队电路在若干中断源中决定首先响应哪个中断，并在其对应的输出线IR_i上给出“1”信号，而其他各线为“0”信号(IR_1—IR_4中有一个信号有效)。之后，编码电路根据排上队的中断源输出信号IR_i，产生一个预定的地址码，转向中断服务程序入口地址。

在多级中断中，如果每一级请求线上还连接有多个中断源设备，那么在识别中断源时，还需要进一步用**串行链式方式**查询。这意味着要用**二维方式**来设计中断排队逻辑。

独立请求方式的中断优先级排队电路与中断向量产生的逻辑结构



处理一个中断的过程，就是妥善处理以下一些基本问题的过程：

1）何时检查中断输入信号及其处理办法。
2）如何把控制转给中断服务程序。
3）如何保护和恢复中断的现场。
4）如何识别中断源。
5）如何识别优先级较高的中断。
6）如何开放和关闭中断。

中断方式的特点：

1、CPU与外设能并行工作
2、能处理异常事件
3、I/O操作仍然经过CPU，在程序控制下完成I/O
4、CPU利用率仍然不太好
5、实时性好

7.中断响应次序与处理次序

1)中断响应次序

同时发生多个中断请求时，由中断响应硬件的排队器所决定的响应次序，次序是固定的。

2)中断处理次序

一个中断处理程序执行前再有其它中断产生时,中断处理完成的次序，可以不同于响应次序。

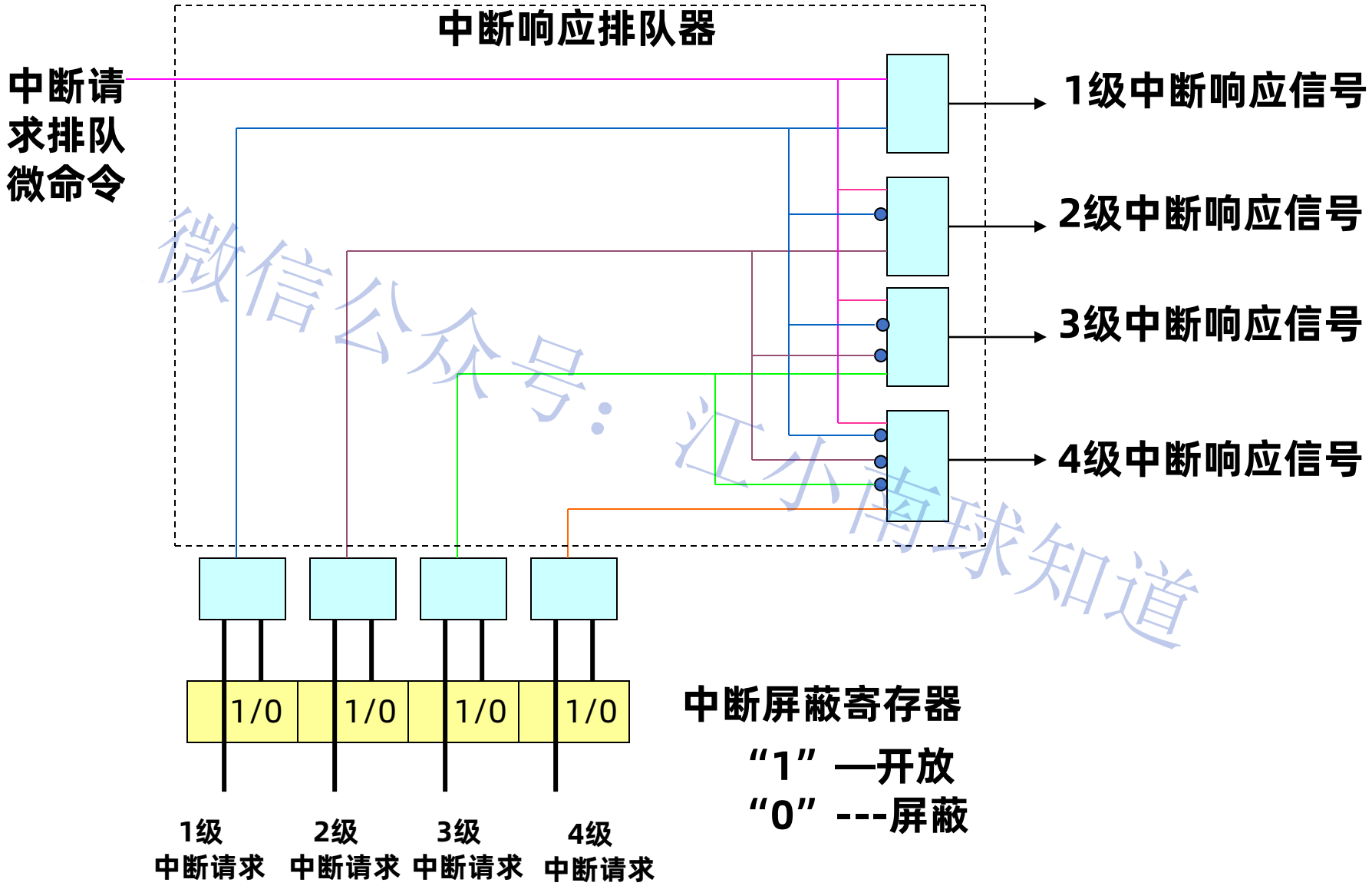
3)处理原则

在处理某级中断时,只有更高级的请求到来才转去响应和处理，完成后返回原中断继续处理。

中断处理次序改变方法

1)设置中断级屏蔽寄存器硬件以决定是否让某级中断请求进入中断响应排队器，进入排队器的中断请求，级别高的优先得到响应。

2)OS对每类中断处理程序的现行PSW中的中断级屏蔽位进行设置，可以实现希望的处理次序。

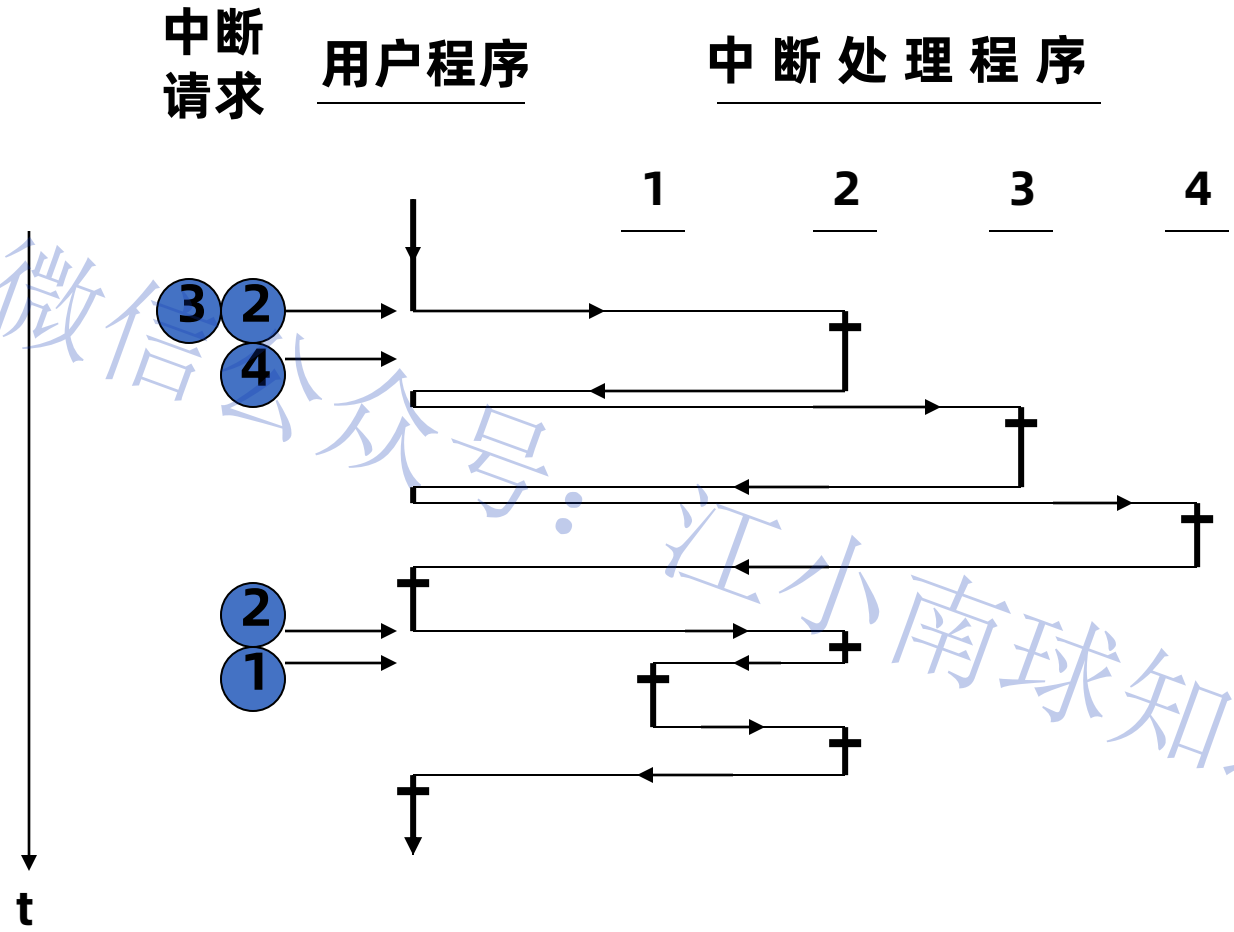


例:系统有4个中断级,每级现行PSW有4位屏蔽位“1”表示对该级的请求都开放,允许其进入排队器。“0”表示屏蔽各个请求,不允许进入排队，现要求各级**中断处理次序**和**响应次序**都是1->2->3->4，请设计屏蔽位状态。

中断级屏蔽位举例1(1->2->3->4)

中断处理程序级别	中断级屏蔽位			
	1级	2级	3级	4级
第1级	0	0	0	0
第2级	1	0	0	0
第3级	1	1	0	0
第4级	1	1	1	0

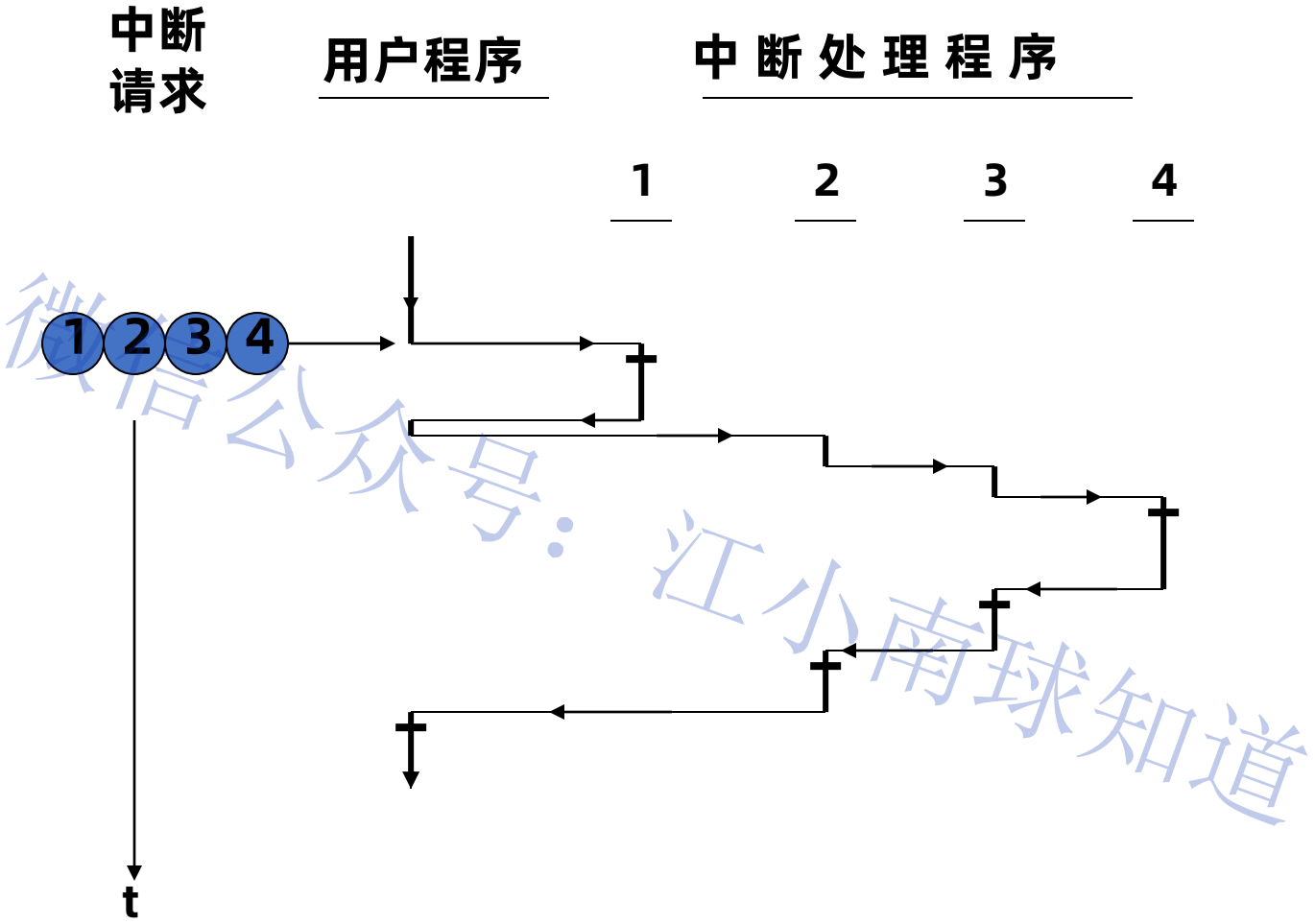
不高于本级的屏蔽掉即“0”



中断处理次序为1→2 3→4的例子

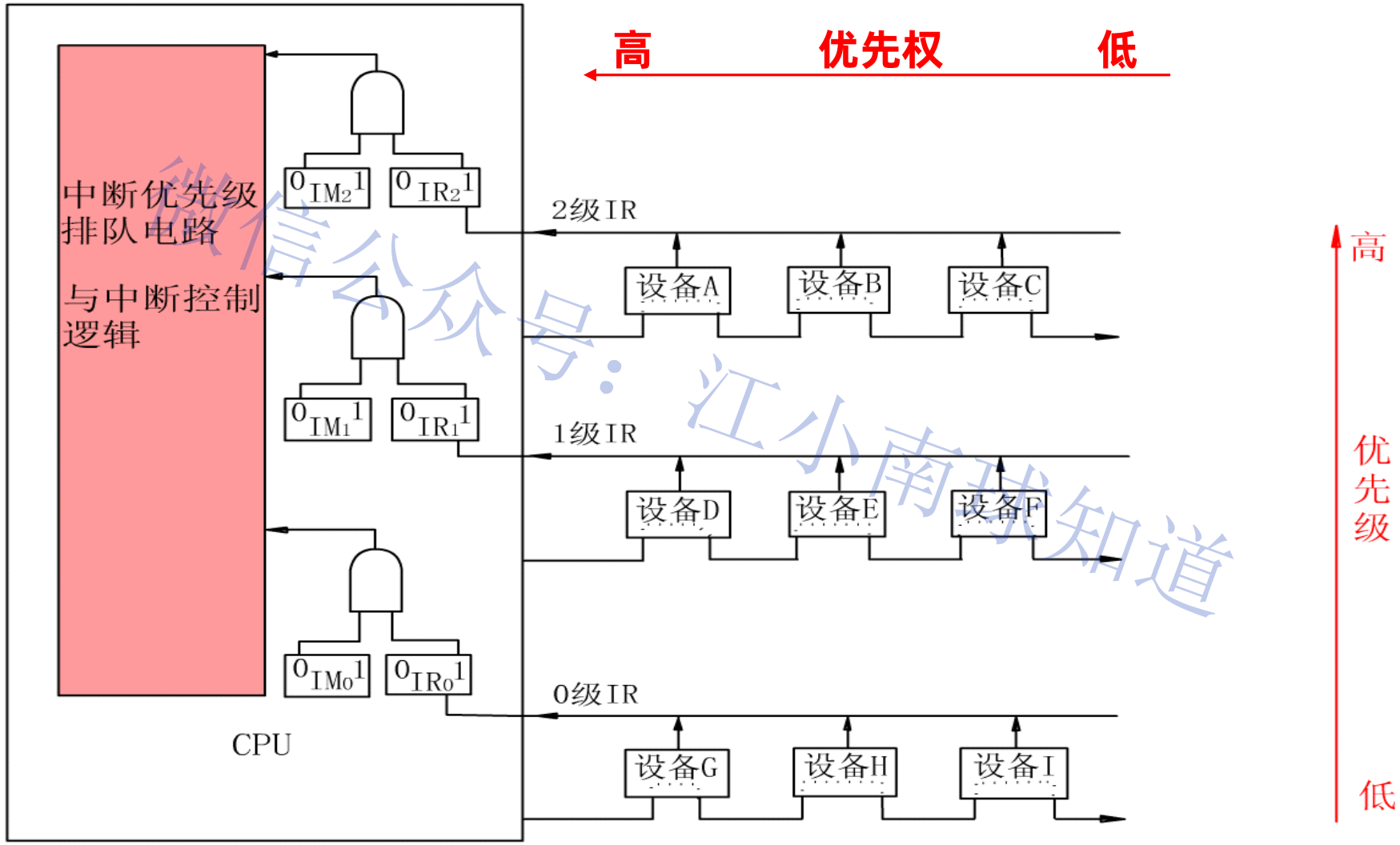
中断级屏蔽位举例2(1→4→3→2)→

中断处理程序级别	中断级屏蔽位			
	1级	2级	3级	4级
第1级	0	0	0	0
第2级	1	0	1	1
第3级	1	0	0	1
第4级	1	0	0	0



中断处理次序为1 → 4 → 3 → 2的例子

例



请问：

(1)在中断情况下，CPU和设备的优先级如何考虑?请按降序排列各设备的中断优先级。

(2)若CPU现执行设备B的中断服务程序，IM_2，IM_1，IM_0的状态是什么?如果CPU执行设备D的中断服务程序，IM_2，IM_1，IM_0的状态又是什么?

(3)每一级的IM能否对某个优先级的个别设备单独进行屏蔽?如果不能，采取什么办法可达到目的?

(4)假如设备C一提出中断请求，CPU立即响应，如何调整才能满足此要求?

解：

(1)在中断情况下，CPU的优先级最低。各设备的优先次序是：

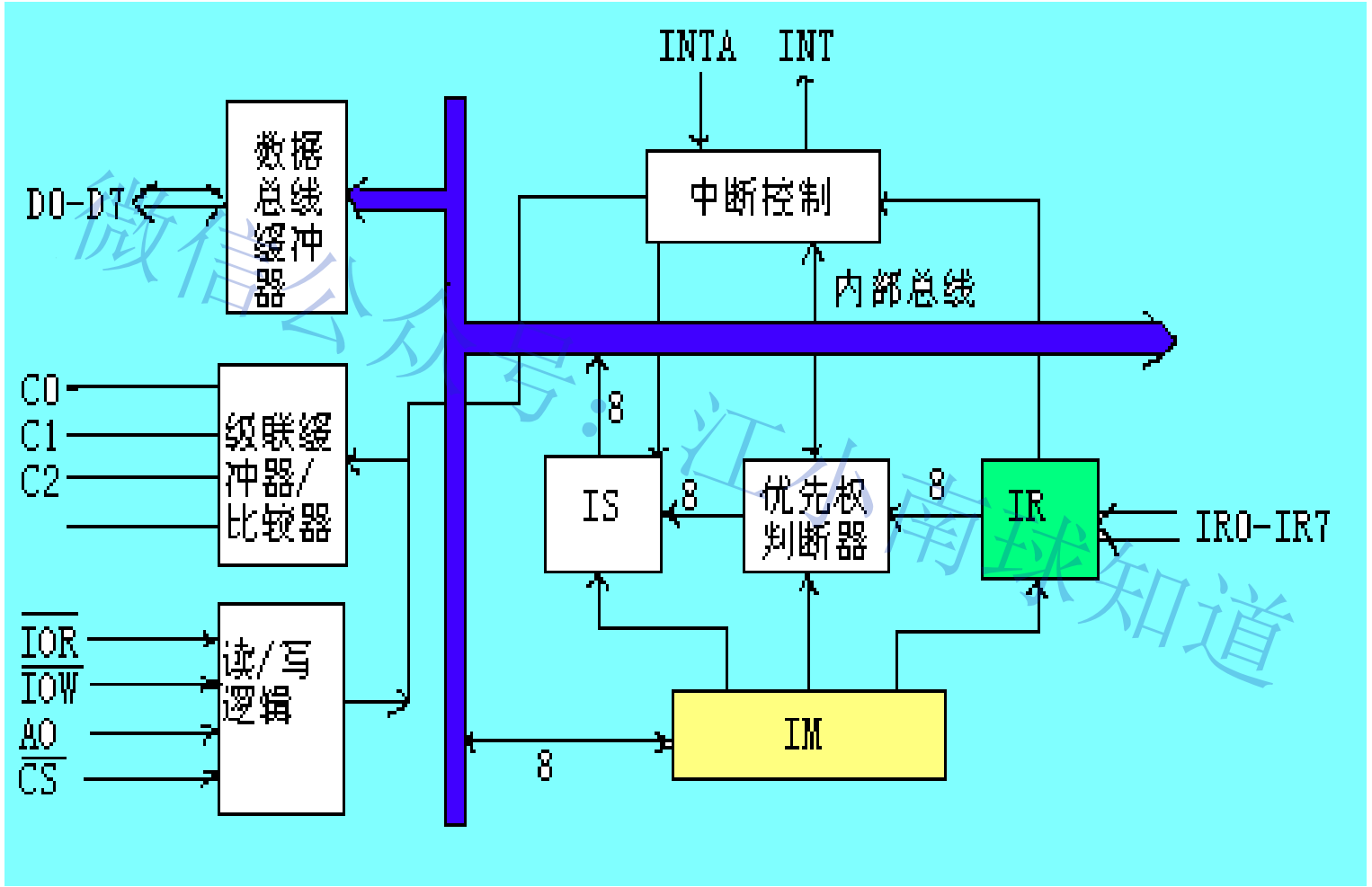
A→B→C→ D→E→F→G→H→I→CPU。

(2)执行设备B的中断服务程序时$IM_2IM_1IM_0$=111；执行设备D的中断服务程序时，$IM_2IM_1IM_0$=011。

(3)每一级的IM标志不能对某个优先级的个别设备进行单独屏蔽。可将接口中的EI(中断允许)标志清“0”，它禁止设备发出中断请求。

(4)要使设备C的中断请求及时得到响应，可将设备C从第2级取出来，单独放在第3级上，使第3级的优先级最高，即令IM_3=0即可。

8.中断控制器(8259)



8259的中断优先级选择方式有四种：

(1)完全嵌套方式：是一种固定优先级方式，连至IR0设备优先级最高，IR7的优先级最低。这种固定优先级方式对级别低的中断不利，在有些情况下最低级别的中断请求可能一直不能被处理。

(2)轮换优先级方式A：每个级别的中断保证有机会被处理，将给定的中断级别处理完后，立即把它放到最低级别的位置上去。

(3) 轮换优先级方式B：要求CPU可在任何时间规定最优优先级，然后顺序地规定其他IR线上的优先级。

(4)查询方式：由CPU访问8259的中断状态寄存器，一个状态字能表示出正在请求中断的最高优先级IR线，并能表示出中断请求是否有效。

8259提供了两种屏蔽方式：

(1)简单屏蔽方式：提供8位屏蔽字，每位对应着各自的IR线。被置位的任一位则禁止了对应IR线上的中断。

(2)特殊屏蔽方式：允许CPU让来自低优先级的外设中断请求去中断高优先级的服务程序。当8位屏蔽位的某位置“0”时，例如屏蔽字为11001111，说明IR4和IR5线上的中断请求可中断任何高级别的中断服务程序。

8259中断控制器的不同工作方式是通过编程来实现的。CPU送出一系列的初始化控制字和操作控制字来执行选定的操作。

9.奔腾中断机制

(1)中断类型

中断 通常称为外部中断，是由CPU的外部硬件信号引发的。有两种情况：

①可屏蔽中断：

CPU的INTR引脚收到中断请求信号，如果CPU中标志寄存器IF=1时，可引发中断；IF=0时，中断请求信号在CPU内部被禁止。

②非屏蔽中断：

CPU的NMI引脚收到的中断请求信号而引发中断，这类中断不能被禁止。

异常 通常称为异常中断，它是由指令执行引发的。有两种情况：

①执行异常：

CPU执行一条指令过程中出现错误、故障等不正常条件引发的中断；

②执行软件中断指令：

如执行INT 0，INT 3，INT n等指令，执行时产生异常中断。

Pentium共有256种中断和异常。每种中断给予一个编号，称为**中断向量号**(0—255)，以便发生中断时，程序转向相应的中断服务子程序入口地址。

当有一个以上的异常或中断发生时，CPU以一个预先确定的优先顺序为它们先后进行服务。

(2)中断服务子程序进入过程

中断服务子程序的入口地址信息存于中断向量号检索表内。**实模式为中断向量表IVT，保护模式为中断描述符表IDT**。

CPU识别中断类型取得中断向量号的途径有三种：

①指令给出：如软件中断指令INT n 中的n即为中断向量号。

②外部提供：可屏蔽中断是在CPU接收到INTR信号时产生一个中断识别周期，接收外部中断控制器由数据总线送来的中断向量号；非屏蔽中断是在接收到NMI信号时中断向量号固定为2。

③CPU识别错误、故障现象，根据异常和中断产生的条件自动指定向量号。

(3)中断服务子程序进入过程

中断服务子程序的入口地址信息存于中断向量号检索表内。**实模式为中断向量表IVT，保护模式为中断描述符表IDT**。

CPU识别中断类型取得中断向量号的途径有三种：

①指令给出：如软件中断指令INT n 中的n即为中断向量号。

②外部提供：可屏蔽中断是在CPU接收到INTR信号时产生一个中断识别周期，接收外部中断控制器由数据总线送来的中断向量号；非屏蔽中断是在接收到NMI信号时中断向量号固定为2。

③CPU识别错误、故障现象，根据异常和中断产生的条件自动指定向量号。

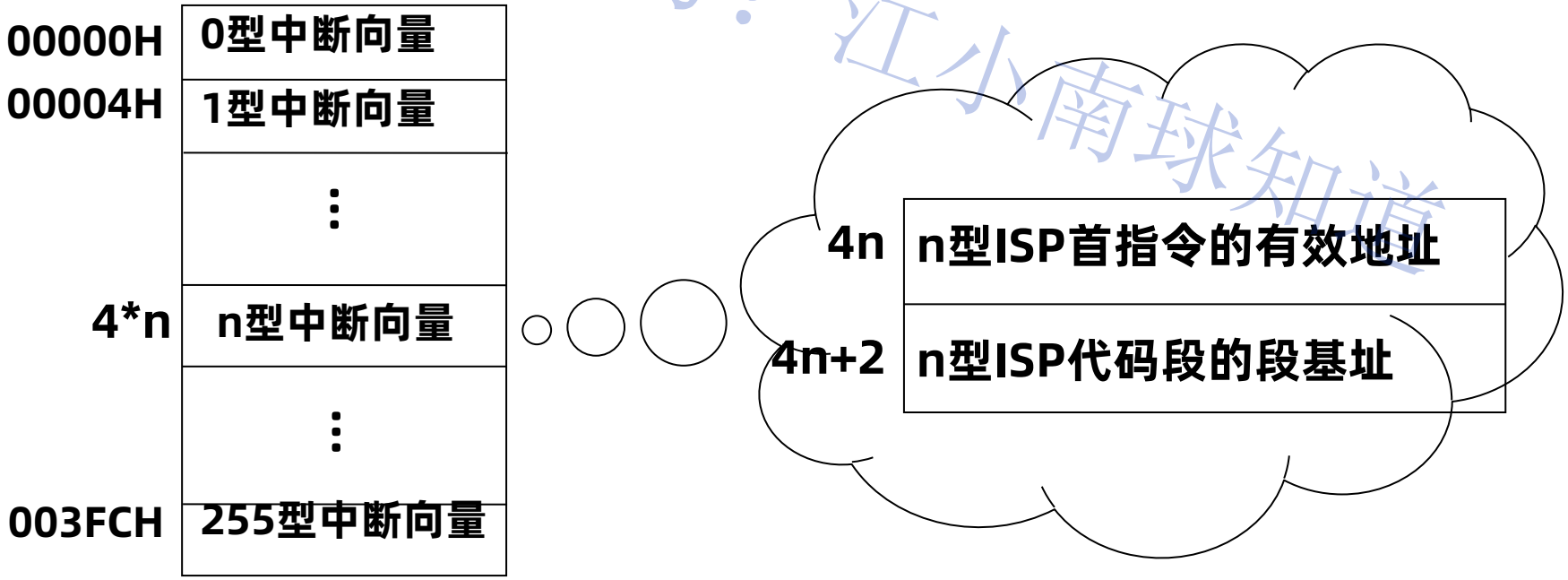
CPU依据中断向量号获取中断服务子程序入口地址，但在实模式下和保护模式下采用不同的途径:

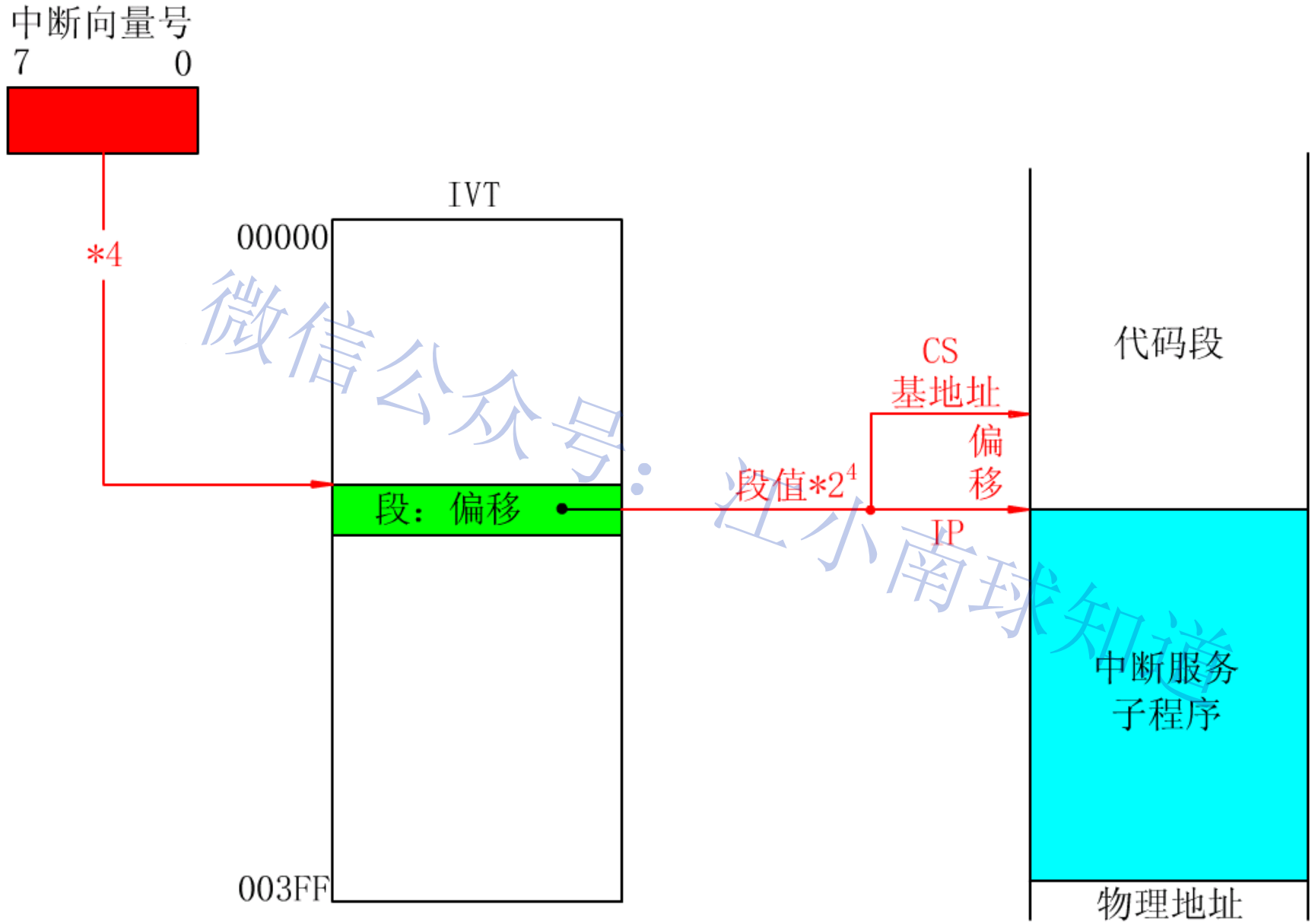
实模式下:

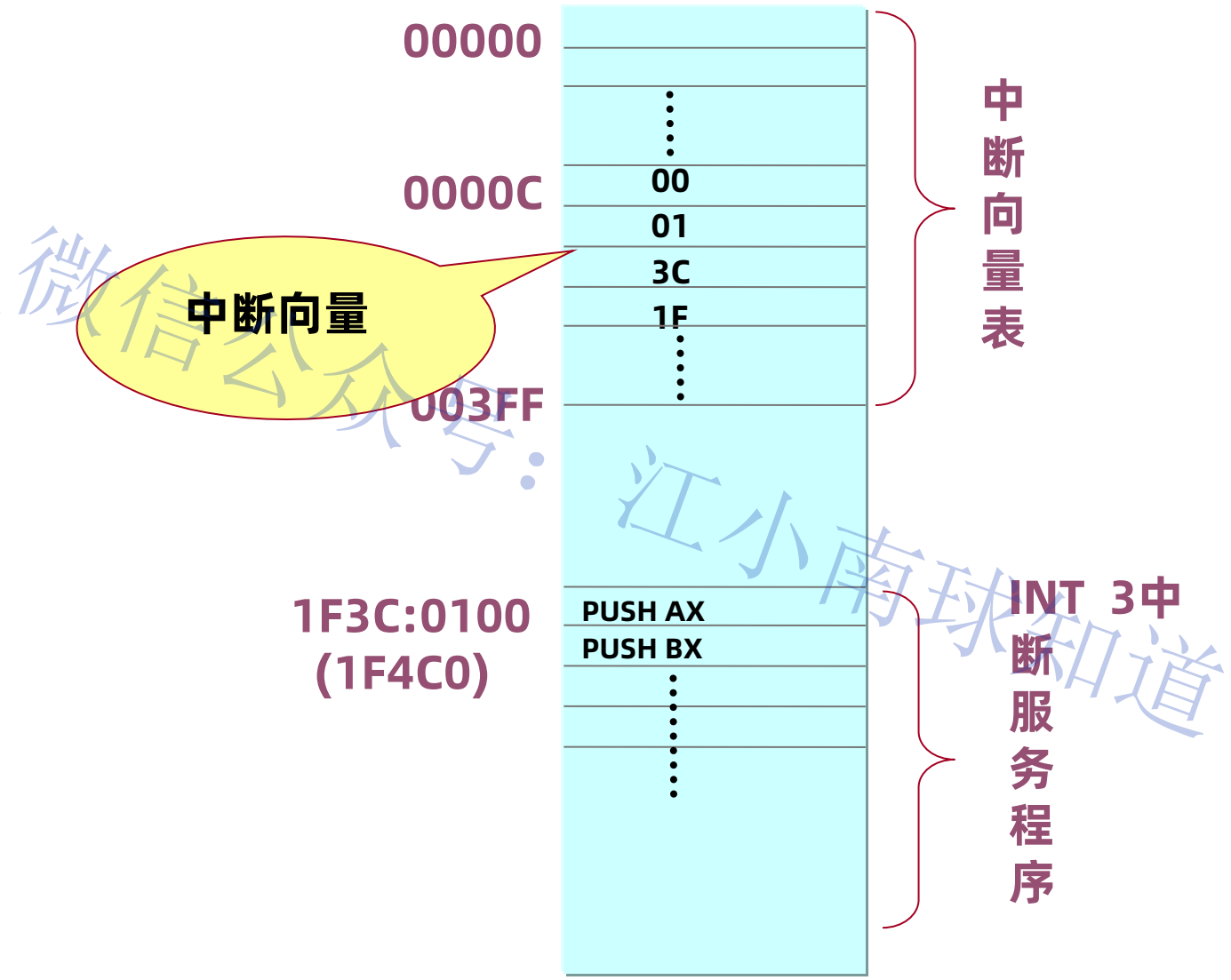
使用中断向量表

中断向量表IVT位于内存地址0开始的1KB空间。

实模式是16位寻址，中断服务子程序入口地址(段，偏移)的段寄存器和段内偏移量各为16位。







保护模式下 使用中断描述符表

概述

- 保护模式下采用中断描述符表IDT管理各级中断；
- IDT中最多可以有256个描述符，对应于256个中断/异常源；
- IDT表中的描述符包括了中断服务程序的入口地址信息；
- 这些门描述符为8字节长，对应256个中断向量号，IDT表长为2KB。

IDT可置于内存的任意区域，其起始地址由中断描述符表寄存器(IDTR)设置；

- 中断门和陷阱门必须设在IDT表中，中断门对应外部硬件中断，陷阱门对应内部软件中断或异常。

中断门、陷阱门描述符

访问权限字节

P位： P=0，该段不在内存中

P=1，该段在内存中

DPL： 取值0~3，确定段的特权级

00为0级；

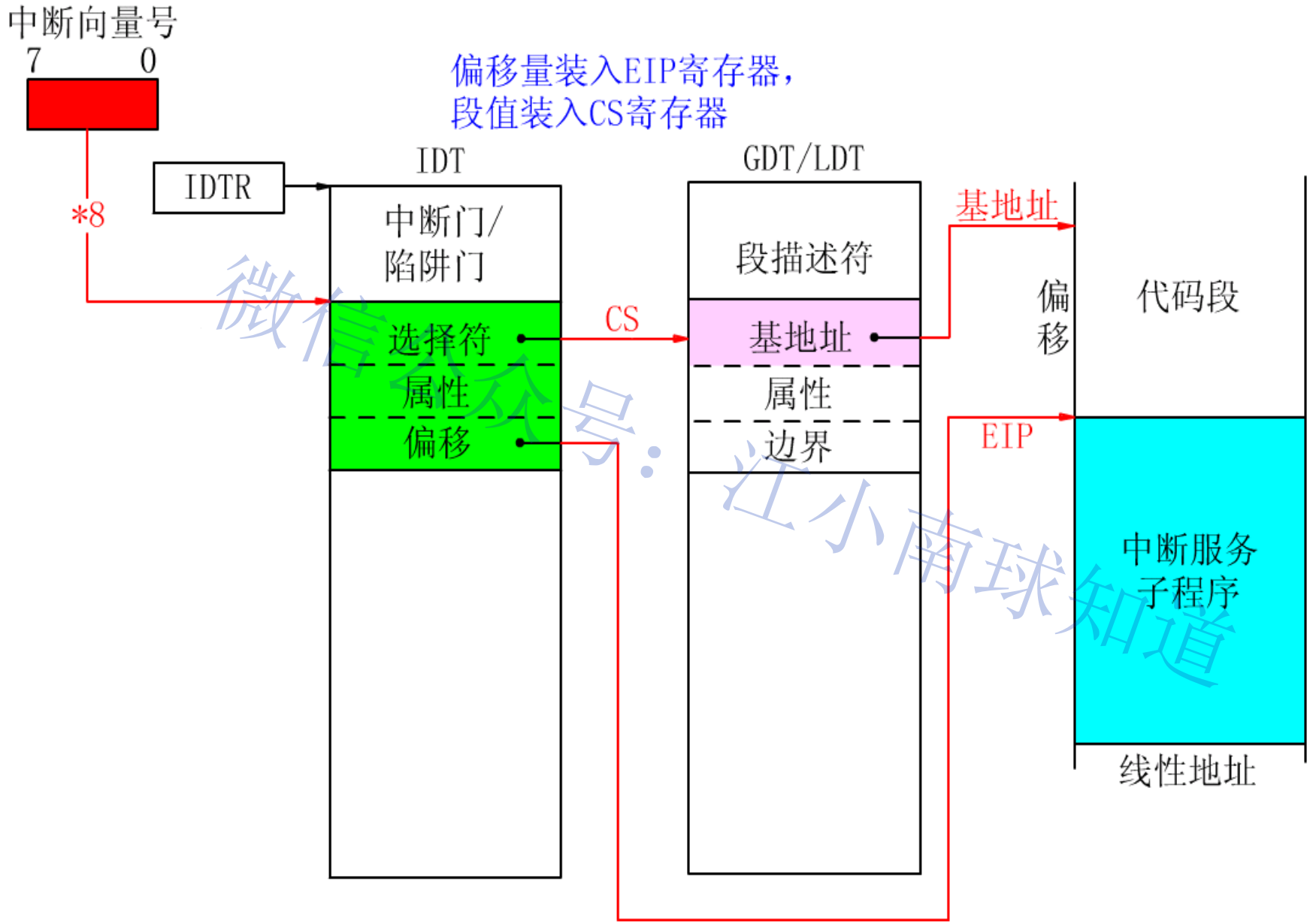
01为1级；

10为2级；

11位3级。

Type：1110 —中断门；

1111— 陷阱门。



中断处理过程

① 当中断处理的CPU控制权转移涉及到特权级改变时，必须把当前的SS和ESP两个寄存器的内容压入系统堆栈予以保存。
② 标志寄存器EFLAGS的内容也压入堆栈。
③ 清除标志触发器TF和IF。
④ 当前的代码段寄存器CS和指令指针EIP也压入此堆栈。
⑤ 如果中断发生伴随有错误码，则错误码也压入此堆栈。
⑥ 完成上述中断现场保护后，从中断向量号获取的中断服务子程序入口地址(段，偏移)分别装入CS和EIP，开始执行中断服务子程序。
⑦ 中断服务子程序最后的IRET指令使中断返回。保存在堆栈中的中断现场信息被恢复，并由中断点继续执行原程序。

一、DMA方式的基本概念

DMA方式是为了在主存与外设之间实现高速、批量数据交换而设置的。DMA方式的数据传送直接依靠硬件（DMA控制器）来实现，不需要执行任何程序。

1.DMA方式

直接内存访问(DMA)是一种完全由硬件执行I/O交换的工作方式。在这种方式中，**DMA控制器**从CPU完全接管对总线的控制，数据交换不经过CPU，而直接在内存和I/O设备之间进行 。DMA方式一般用于**高速传送成组数据**。DMA控制器将向内存发出地址和控制信号，修改地址，对传送的字的个数计数，并且以中断方式向CPU报告传送操作的结束。

2.DMA方式的主要优点

速度快。由于CPU根本不参加传送操作，因此就省去了CPU取指令、取数、送数等操作。在数据**传送过程中**，没有保存现场、恢复现场之类的工作。内存地址修改、传送字个数的计数等等，也不是由软件实现，而是用硬件线路直接实现的。所以DMA方式能满足高速I/O设备的要求，也有利于CPU效率的发挥。

3.DMA方式具有下列特点

(1)它使主存与CPU的固定联系脱钩，主存既可被CPU访问，又可被外设访问；

(2)在数据块传送时，主存地址的确定、传送数据的计数等硬件电路直接实现；

(3)主存中要开辟专用缓冲区，及时供给和接收外设的数据；

(4)DMA传送速度快，CPU和外设并行工作，提高了系统的效率。

(5)DMA在传送前要通过程序进行预处理，结束后要通过中断方式进行后处理。

二、基本的DMA控制器

1、DMA控制器的基本组成

DMA控制器，实际上是采用DMA方式的外围设备与系统总线之间的接口电路。这个接口电路是在中断接口的基础上再加DMA机构组成。一个最简单的DMA控制器由以下逻辑部件组成：

（1）内存地址计数器 用于存放内存中要交换的数据的地址。在DMA传送前，须通过程序将数据在内存中的起始位置(首地址)送到内存地址计数器。而当DMA传送时，每交换一次数据，将地址计数器加“1”，从而以增量方式给出内存中要交换的一批数据的地址。

（2）字计数器 用于记录传送数据块的长度(多少字数)。其内容也是在数据传送之前由程序预置，交换的字数通常以补码形式表示。在DMA传送时，每传送一个字，字计数器就加“1”，当计数器溢出即最高位产生进位时，表示这批数据传送完毕，于是引起DMA控制器向CPU发中断信号。

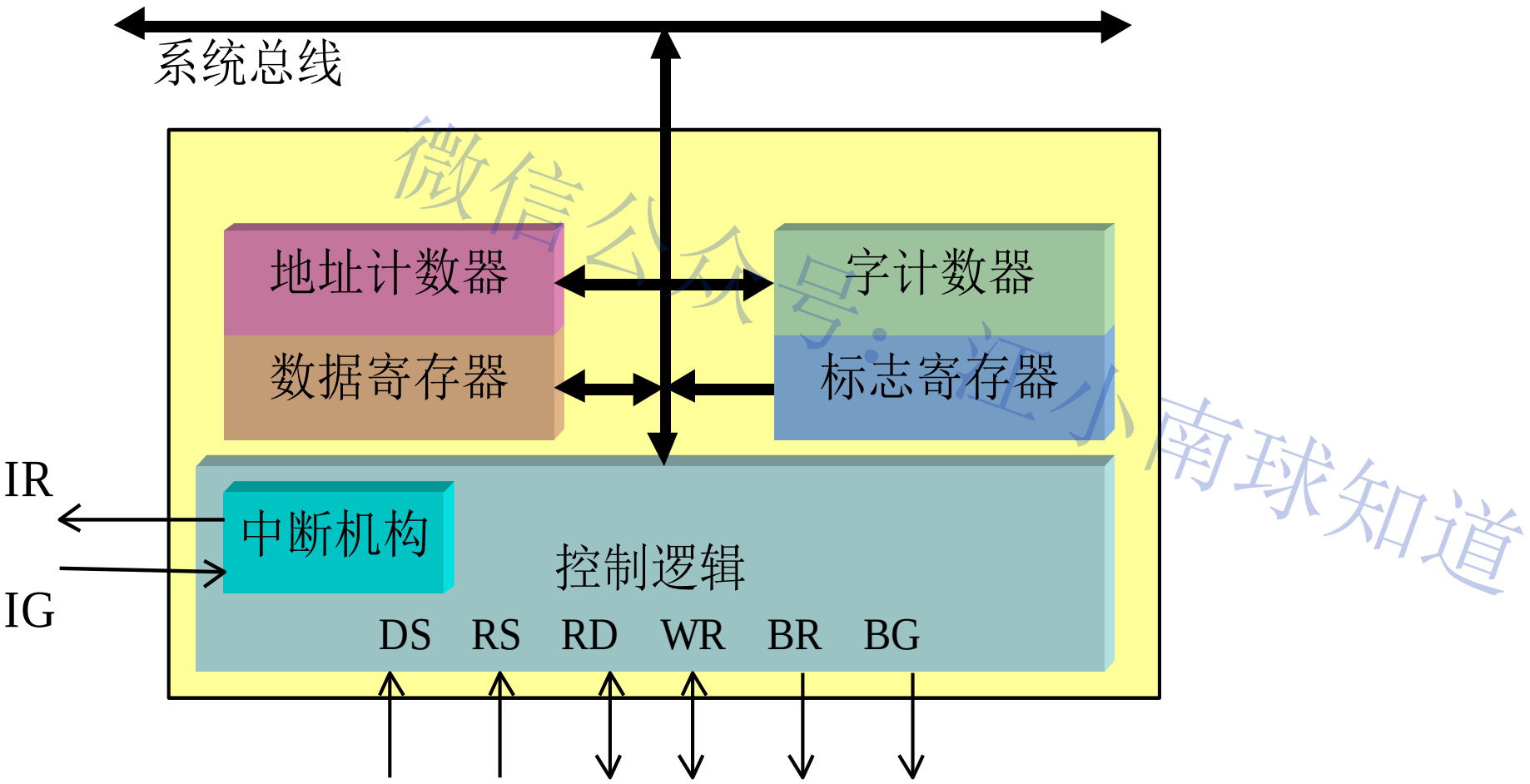
（3）数据缓冲寄存器 用于暂存每次传送的数据(一个字)。当输入时，由设备(如磁盘)送往数据缓冲寄存器，再由缓冲寄存器通过数据总线送到内存。反之，输出时，由内存通过数据总线送到数据缓冲寄存器，然后再送到设备。

（4）“DMA请求”标志 每当设备准备好一个数据字后给出一个控制信号，使“DMA请求” 标志置“1”。该标志置位后向“控制/状态”逻辑发出DMA请求，后者又向CPU发出总线使用权的请求(HOLD)，CPU响应此请求后发回响应信号HLDA，“控制/状态”逻辑接收此信号后向CPU发总线请求，在得到CPU的响应信号，使“DMA请求”标志复位，为交换下一个字做好准备。

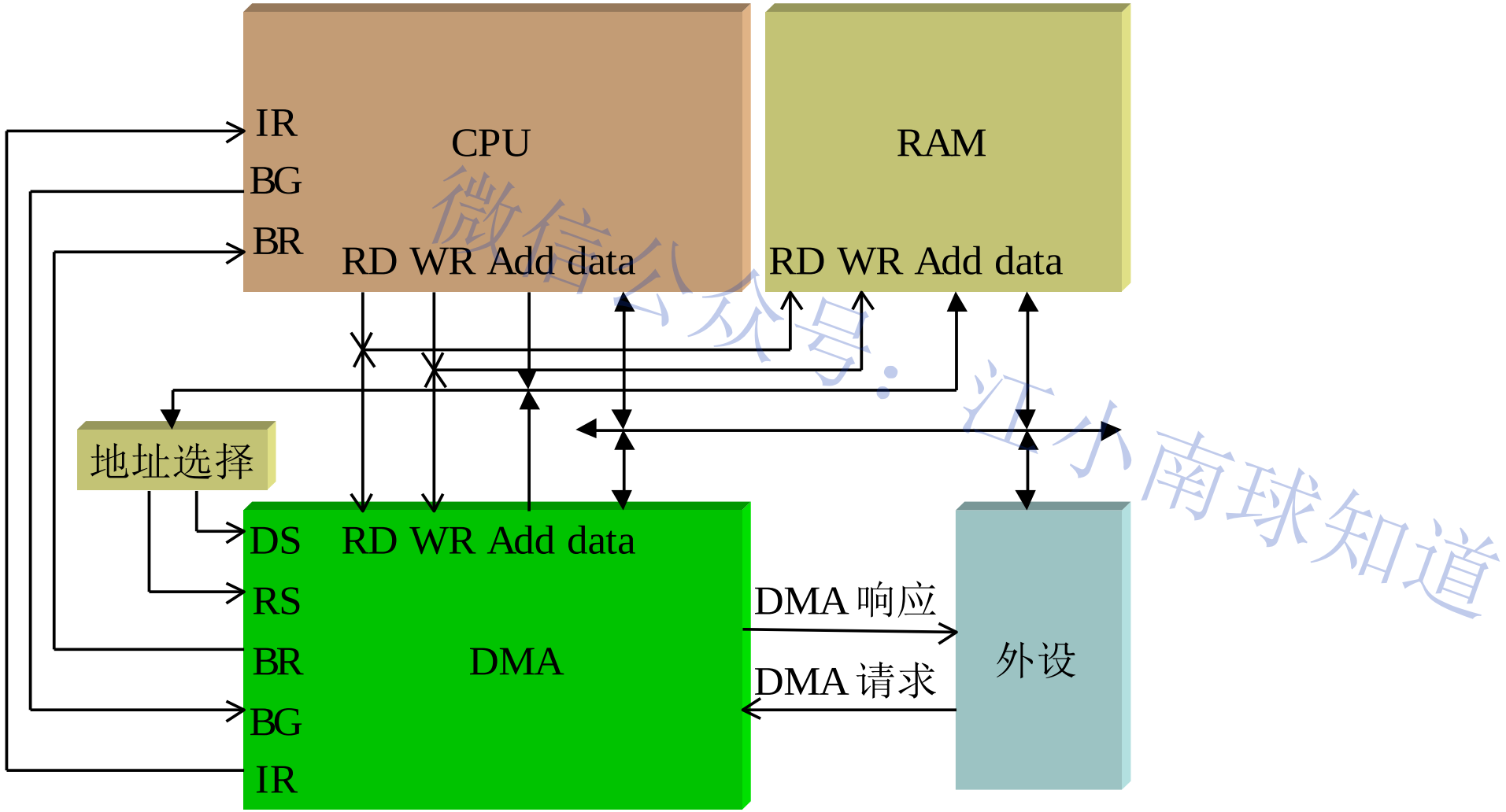
（5）“控制/状态”逻辑 由控制和时序电路以及状态标志等组成，用于**修改内存地址计数器和字计数器**，指定传送类型(输入或输出)，并对“DMA请求”信号和CPU响应信号进行协调和同步。

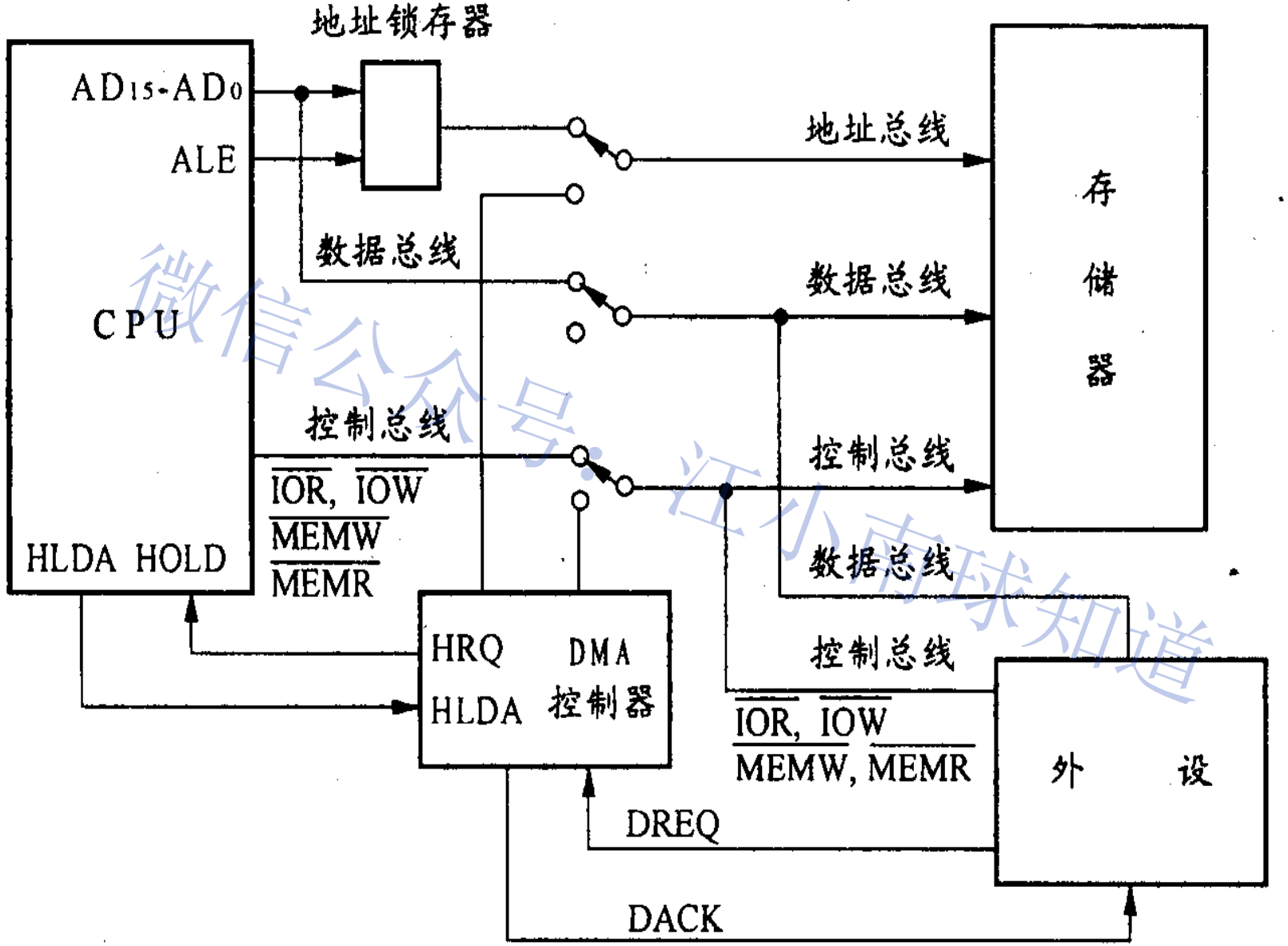
（6）中断机构 当字计数器溢出时(全0)，意味着一组数据交换完毕，由溢出信号触发中断机构，向CPU提出中断报告。这里的中断与上一节介绍的I/O中断所采用的技术相同，但中断的目的不同，前面是为了数据的输入或输出，而这里是为了报告一组数据传送结束。因此它们是I/O系统中不同的中断事件。

DMA控制器结构



DMA与CPU的连接





当某个外围设备请求DMA服务时，操作过程如下：

1.DMA控制器接到设备发出的DMA请求时，将请求转送到CPU。

2.CPU在适当的时刻响应DMA请求。若CPU不需要占用总线则继续执行指令；若CPU需要占用总线，则CPU进入等待状态。

3.DMA控制器接到CPU的响应信号后，进行以下工作：

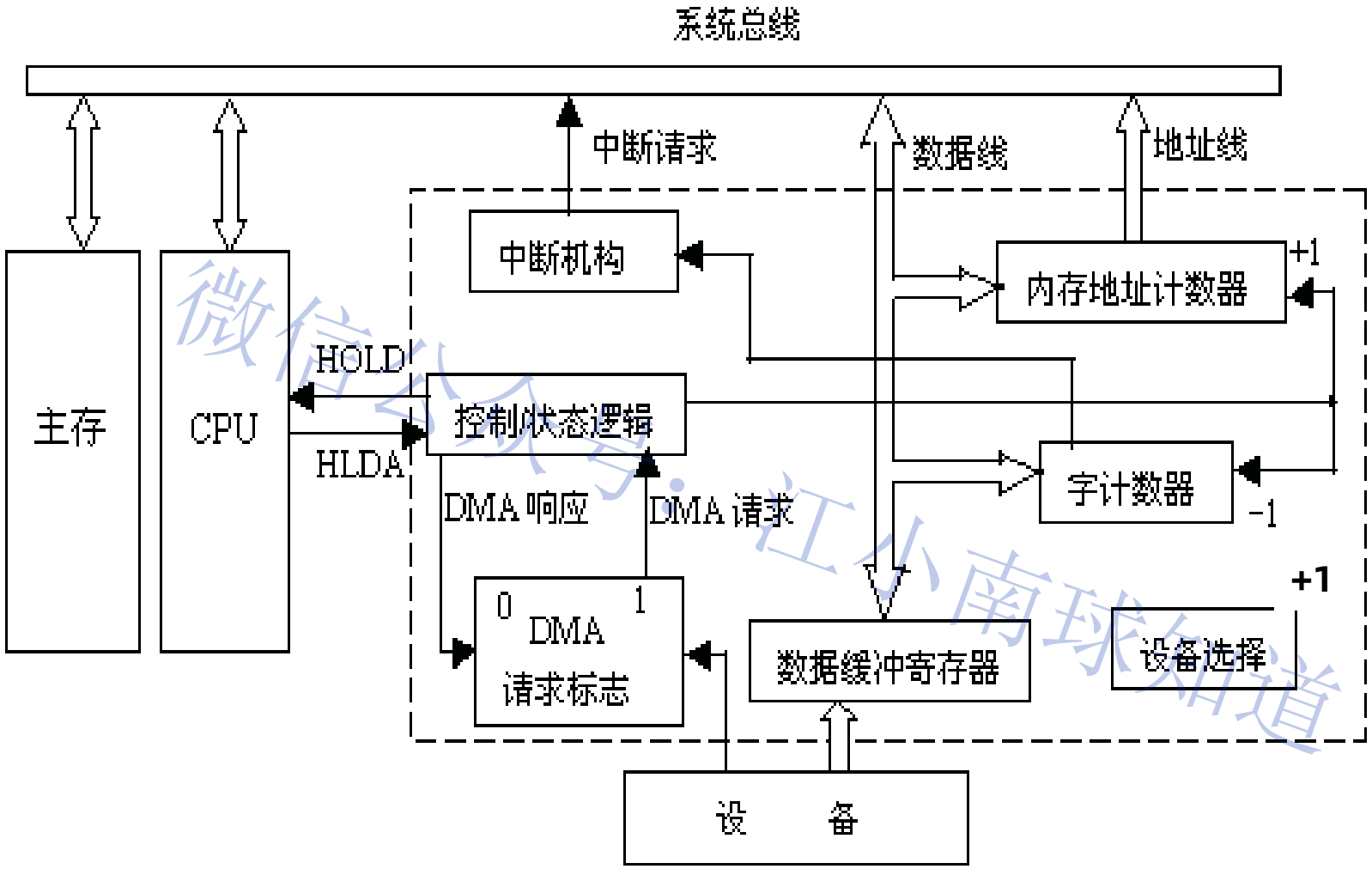
① 对现有DMA请求中优先权最高的请求给予DMA响应；

② 选择相应的地址寄存器的内容驱动地址总线；

③ 根据所选设备操作寄存器的内容，向总线发读、写信号；

④ 外围设备向数据总线传送数据，或从数据总线接收数据；

⑤ 每个字节传送完毕后，DMA控制器使相应的地址寄存器和长度寄存器加“1”或减“1”。



2、DMA控制器的功能

在DMA传送过程中，DMA控制器将接管CPU的地址总线、数据总线和控制总线，CPU的主存控制信号被禁止使用。而当DMA传送结束后，将恢复CPU的一切权利并开始执行其它操作。由此可见，DMA控制器必须具有**控制系统总线的能力**，即能够像CPU一样输出地址信号，接收或发出控制信号，输入或输出数据信号。

DMA控制器在外设与主存之间直接传送数据期间，完全代替CPU进行工作，**主要功能有**：

（1）接受外设发出的DMA请求，并向CPU发出总线请求；

（2）CPU响应此总线请求，发出总线响应信号后，接管对总线的控制，进入DMA操作周期；

（3）确定传送数据的主存单元地址及传送长度，并能自动修改主存地址计数值和传送长度计数值；

（4）规定数据在主存与外设之间的传送方向，发出读写或其他控制信号，并执行数据传送的操作。

（5）向CPU报告DMA操作的结束。

3、DMA数据传送过程

DMA的数据块传送过程可分为三个阶段：**传送前预处理；正式传送；传送后处理**。

预处理 由CPU执行几条输入输出指令，测试设备状态，向DMA控制器的设备地址寄存器中送入设备号并启动设备，向内存地址计数器中送入起始地址，向字计数器中送入交换的数据字个数。在这些工作完成后，CPU继续执行原来的主程序。

正式传送 当外设准备好发送数据(输入)或接受数据(输出)时，它发出DMA请求，由DMA控制器向CPU发出总线使用权的请求(HOLD)。下图示出了停止CPU访内方式的DMA传送数据的流程图。

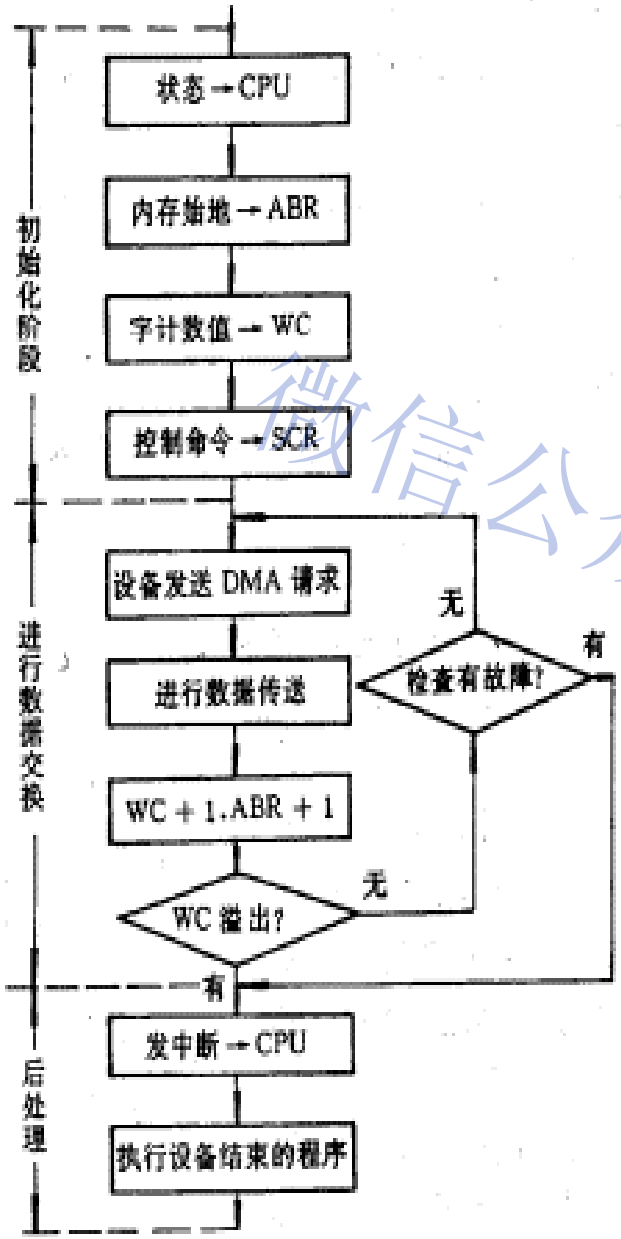
DMA的数据传送是以**数据块**为基本单位进行的，因此，每次DMA控制器占用总线后，无论是数据输入操作，还是输出操作，都是通过循环来实现的。当进行输入操作时，外围设备的数据(一次一个字或一个字节)传向内存；当进行输出操作时，内存的数据传向外围设备。

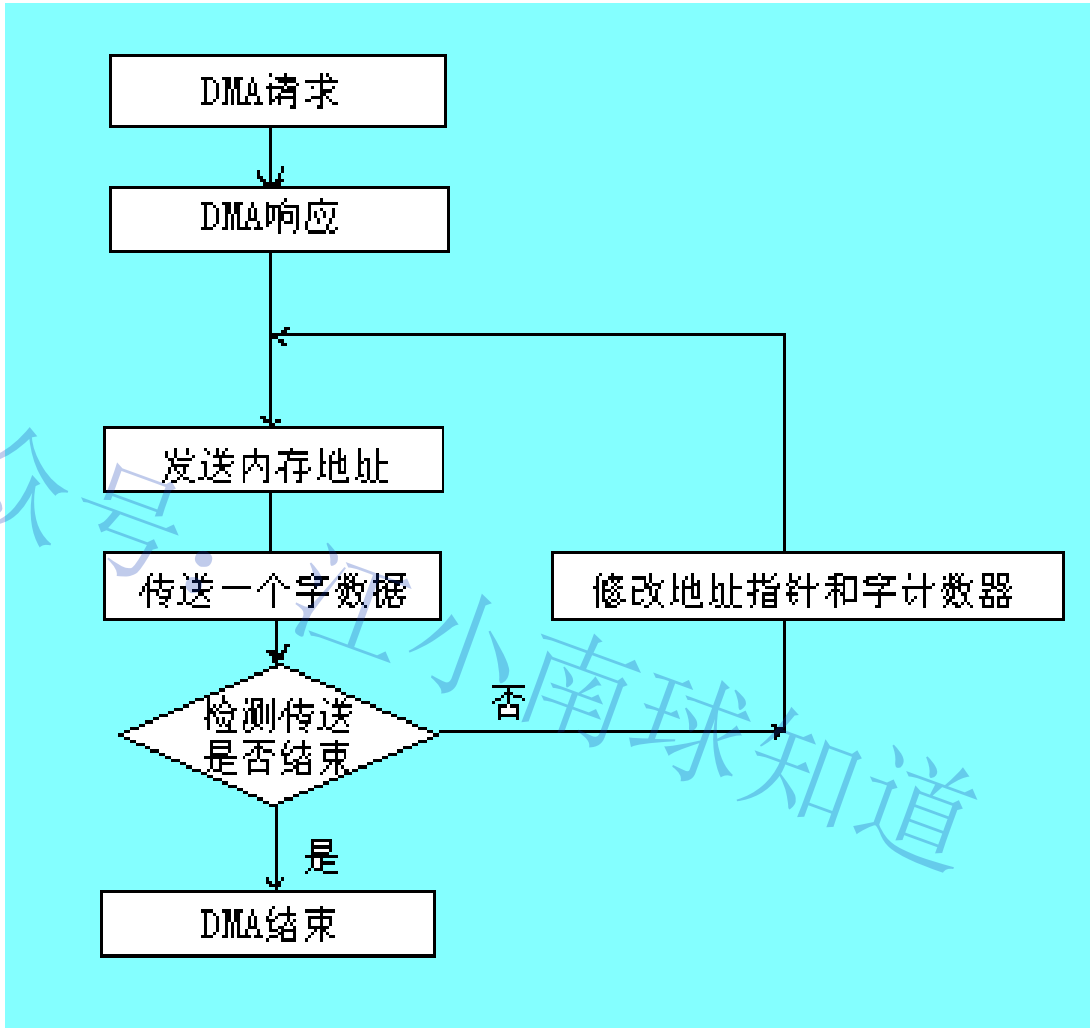
后处理 一旦DMA的中断请求得到响应，CPU停止主程序的执行，转去执行中断服务程序做一些DMA的结束处理工作。这些工作包括校验送入内存的数据是否正确；决定继续用DMA方式传送下去，还是结束传送；测试在传送过程中是否发生了错误等等。

基本DMA控制器与系统的连接方式：

（1）公用的DMA请求方式；

（2）独立的DMA请求方式，这与中断方式类似。





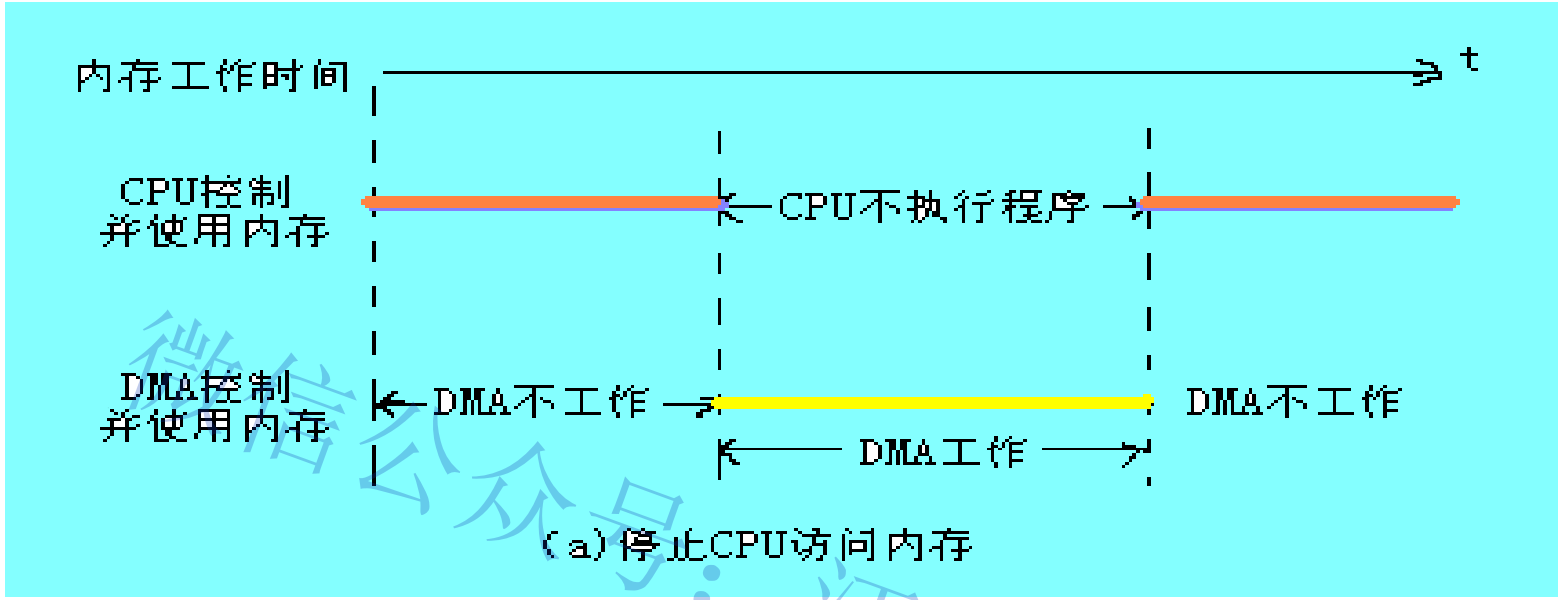
三、DMA传送方式

DMA技术的出现，使得外围设备可以通过DMA控制器直接访问内存，与此同时，CPU可以继续执行程序。DMA控制器与CPU分时使用内存通常采用以下三种方法：

1、停止CPU访问内存

当外围设备要求传送一批数据时，由DMA控制器发一个停止信号给CPU，要求CPU放弃对地址总线、数据总线和有关控制总线的使用权。DMA控制器获得总线控制权以后，开始进行数据传送。在一批数据传送完毕后，DMA控制器通知CPU可以使用内存，并把总线控制权交还给CPU。在这种DMA传送过程中，CPU基本处于不工作状态或者说保持状态。

这种传送方式的时间图如下：



优点：控制简单，它适用于数据传输率很高的设备进行成组传送。

缺点：在DMA控制器访内阶段，**内存的效能**没有充分发挥，相当一部分内存工作周期是空闲的。这是因为，外围设备传送两个数据之间的间隔一般总是大于内存存储周期，即使高速I/O设备也是如此。

2、周期挪用

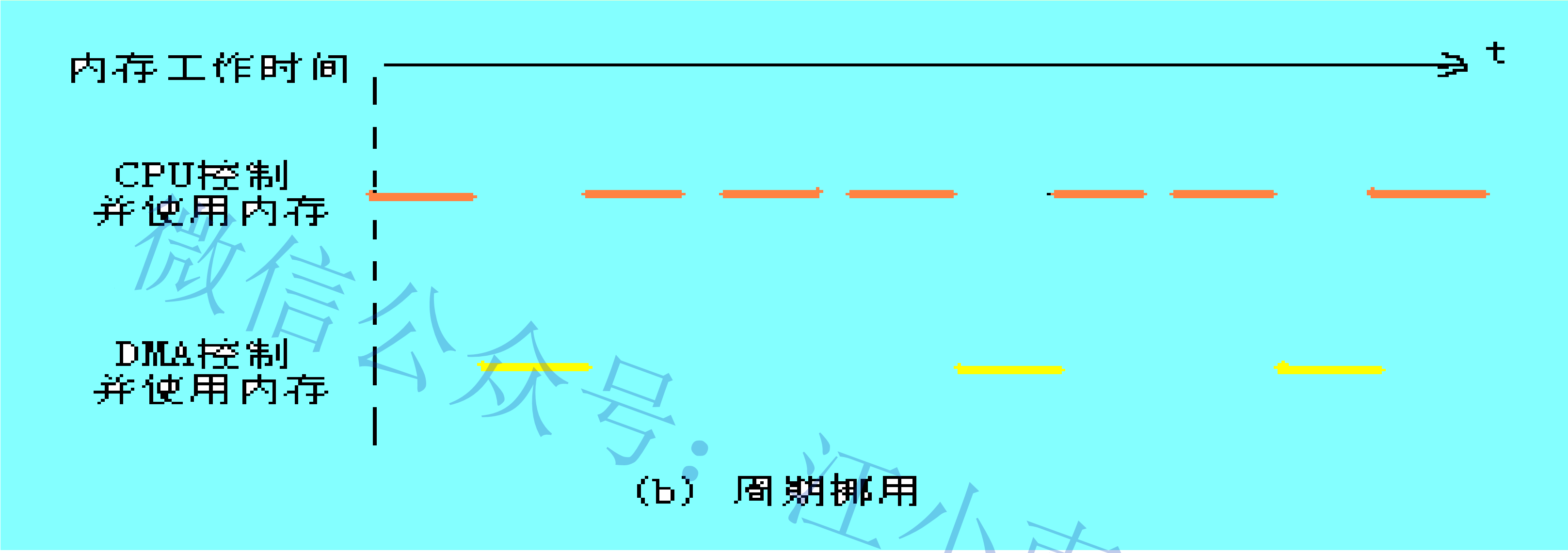
当I/O设备没有DMA请求时，CPU按程序要求访问内存；一旦I/O设备有DMA请求，则由I/O设备挪用一个或几个内存周期。

I/O设备要求DMA传送时可能遇到两种情况：

（1）此时CPU不需要访内，如CPU正在执行乘法指令。由于乘法指令执行时间较长，此时I/O访内与CPU访内没有冲突，即I/O设备挪用一二个内存周期对CPU执行程序没有任何影响。

（2）I/O设备要求访内时CPU也要求访内，这就产生了访内冲突，在这种情况下I/O设备访内优先，因为I/O访内有时间要求，前一个I/O数据必须在下一个访内请求到来之前存取完毕。显然，在这种情况下I/O 设备挪用一二个内存周期，意味着CPU延缓了对指令的执行，或者更明确地说，在CPU执行访内指令的过程中插入DMA请求，挪用了一二个内存周期。

传送方式的时间如下：

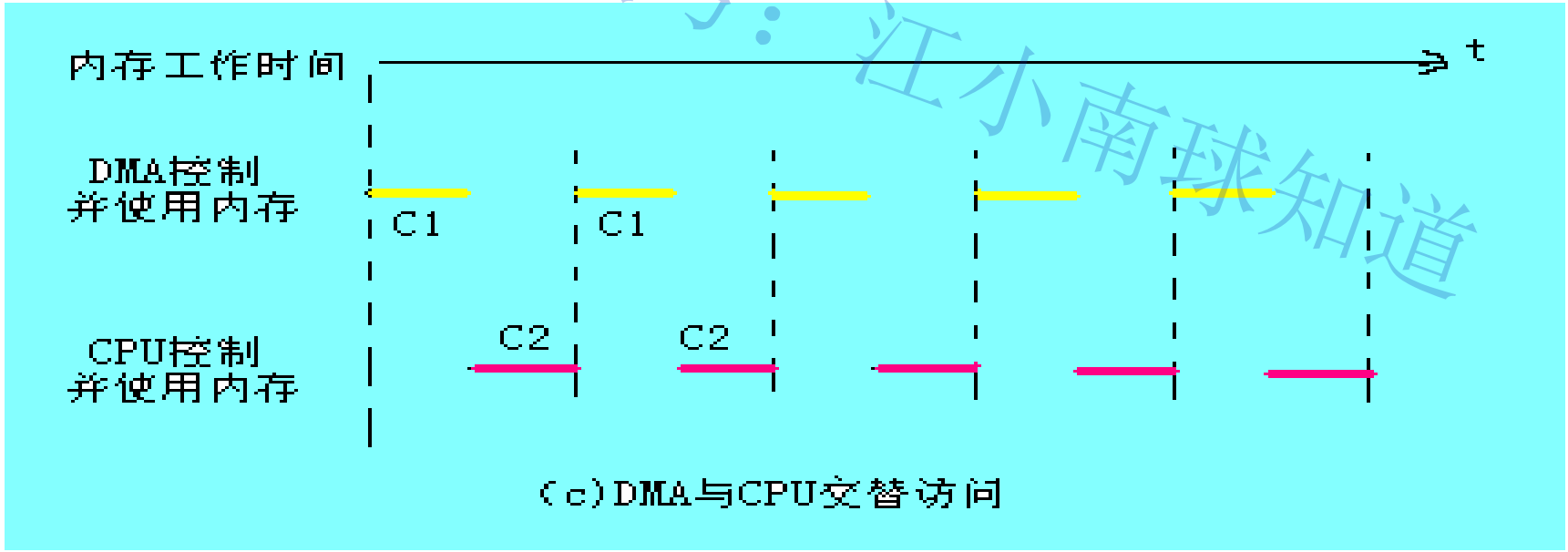


与停止CPU访内的DMA方法比较，周期挪用的方法既实现了I/O传送，又较好地发挥了内存和CPU的效率，是一种广泛采用的方法。但是I/O设备每一次周期挪用都有申请总线控制权、建立总线控制权和归还总线控制权的过程，所以传送一个字对内存来说要占用一个周期，但对DMA控制器来说一般要2—5个内存周期(视逻辑线路的延迟而定)。因此，周期挪用的方法适用于I/O设备读写周期大于内存存储周期的情况。

3、DMA与CPU交替访内

如果CPU的工作周期比内存存取周期长很多，此时采用交替访内的方法可以使DMA传送和CPU同时发挥最高的效率。假设CPU工作周期为 1.2μs，内存存取周期小于0.6μs，那么一个CPU周期可分为C1和C2两个分周期，其中C1供DMA控制器访内，C2专供CPU访内。

这种传送方式的时间图如下：



(c)DMA与CPU交替访问

这种方式不需要总线使用权的申请、建立和归还过程，总线使用权是通过C1和C2分时进行的。CPU和DMA控制器各自有自己的访内地址寄存器、数据寄存器和读/写信号等控制寄存器。在C1周期中，如果DMA控制器有访内请求，可将地址、数据等信号送到总线上。在C2周期中，如CPU有访内请求，同样传送地址、数据等信号。事实上，对于总线，这是用C1，C2控制的一个多路转换器，这种总线控制权的转移几乎不需要什么时间，所以对DMA传送来讲效率是很高的。

这种传送方式又称为“**透明的DMA**”方式，其来由是这种DMA传送对CPU来说，如同透明的玻璃一般，没有任何感觉或影响。在透明的DMA方式下工作，CPU既不停止主程序的运行，也不进入等待状态，是一种高效率的工作方式。当然，相应的硬件逻辑也就更加复杂。

四、选择型和多路型DMA控制器

1.选择型DMA控制器

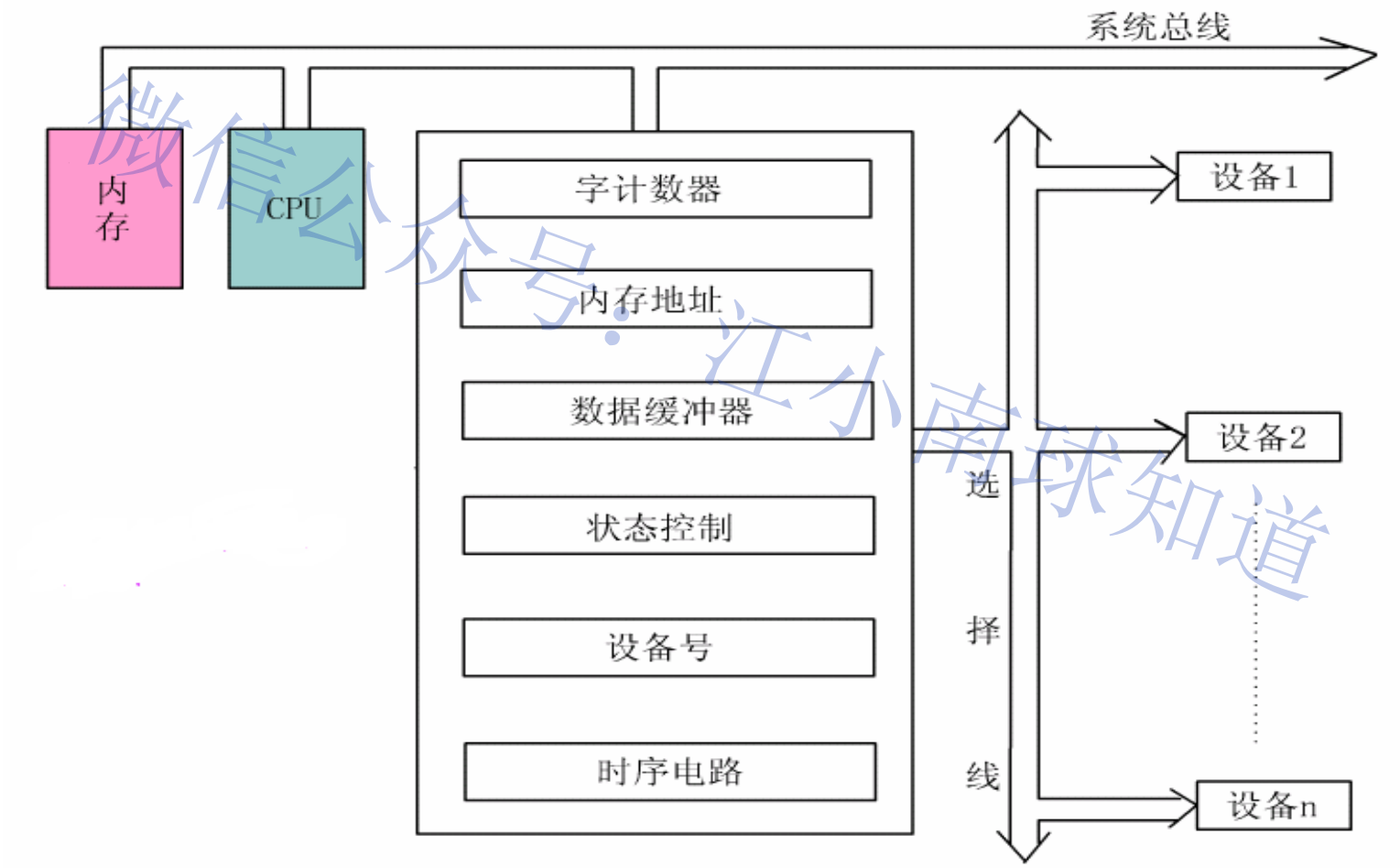
选择型DMA控制器在物理上可以连接多个设备，而在逻辑上只允许连接一个设备。换句话说，在某一段时间内只能为一个设备服务。

选择型DMA控制器的逻辑框图请见CAI演示。

选择型DMA控制器的工作原理：

与前面的简单DMA控制器基本相同。除了前面讲到的基本逻辑部件外，还有一个设备号寄存器。数据传送是以数据块为单位进行的，在每个数据块传送之前的预置阶段，除了用程序中I/O指令给出数据块的传送个数、起始地址、操作命令外，还要给出所选择的设备号。从预置开始，一直到这个数据块传送结束，DMA控制器只为所选设备服务。下一次预置再根据I/O指令指出的设备号，为另一选择的设备服务。显然，选择型DMA控制器相当于一个逻辑开关，根据I/O指令来控制此开关与某个设备连接。

选择型DMA控制器只增加少量硬件达到了为多个外围设备服务的目的，它特别适合数据传输率很高以至接近内存存取速度的设备。在很快地传送完一个数据块后，控制器又可为其他设备服务。

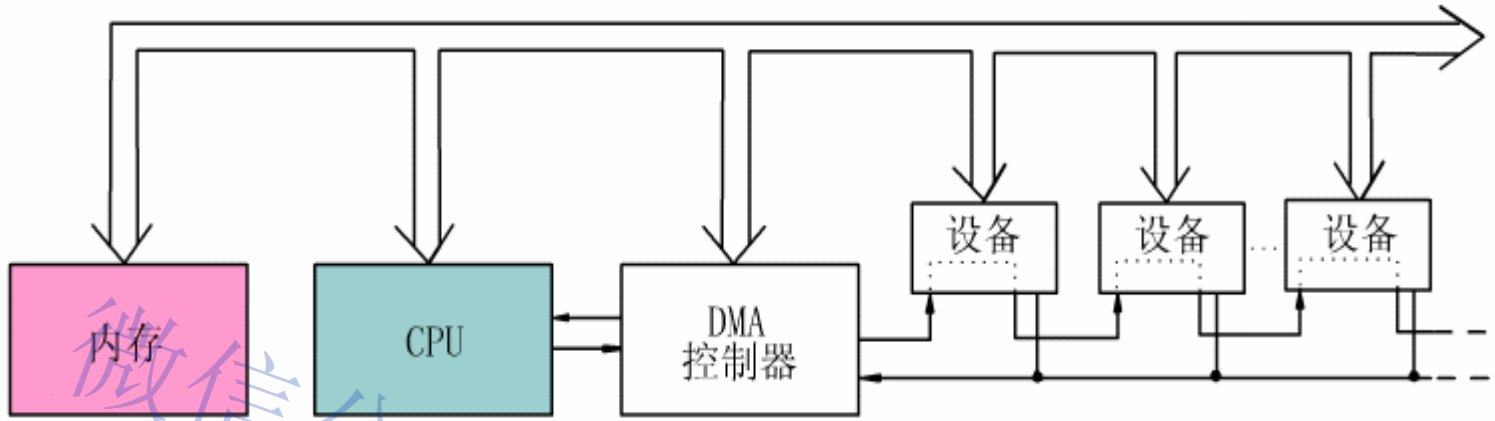


2.多路型DMA控制器

选择型DMA控制器不适用于慢速设备。但多路型DMA控制器适合于同时为多个慢速外围设备服务。

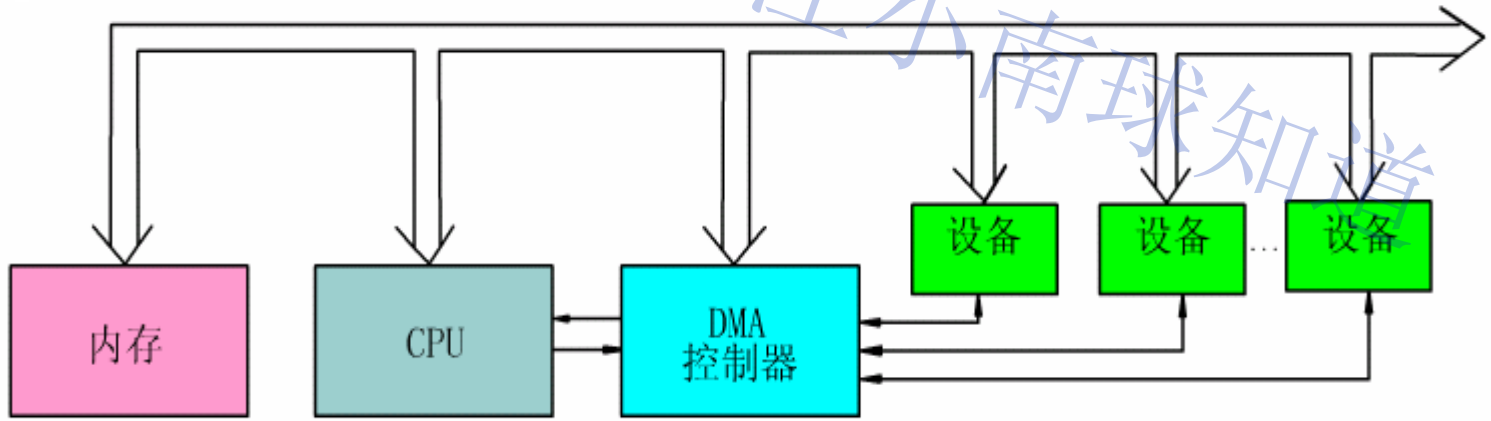
链式多路型DMA控制器和独立请求方式多路型DMA控制器请见CAI演示。

多路型DMA不仅在物理上可以连接多个外围设备，而且在逻辑上也允许这些外围设备同时工作，各设备以字节交叉方式通过DMA控制器进行数据传送。



链式多路型DMA

多路型DMA控制器



独立请求多路型DMA

五、DMA和中断的区别

(1) 中断方式是程序切换，需要保护和恢复现场；而DMA方式除了开始和结尾时，不占用CPU的任何资源。
(2) 对中断请求的响应时间只能发生在每条指令执行完毕时；而对DMA请求的响应时间可以发生在每个机器周期结束时。

(3) 中断传送过程需要CPU的干预；而DMA传送过程不需要CPU的干预，故数据传输速率非常高，适合于高速外设的成组数据传送。

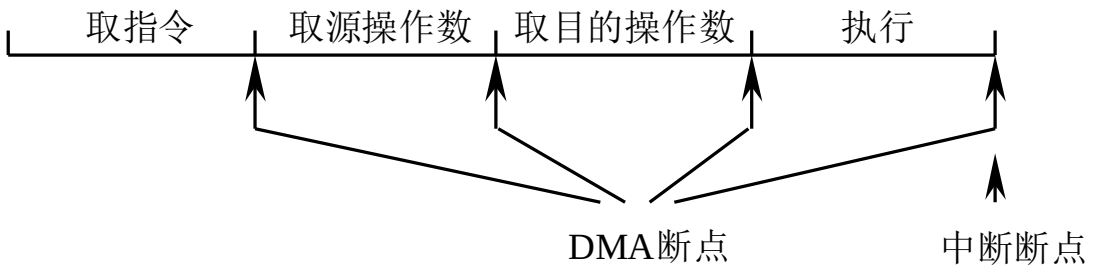
(4)二者优先权不同：DMA请求的优先级高于中断请求。

(5)应用不同：中断方式具有对异常事件的处理能力，而DMA方式仅局限于完成传送数据块的I/O操作。

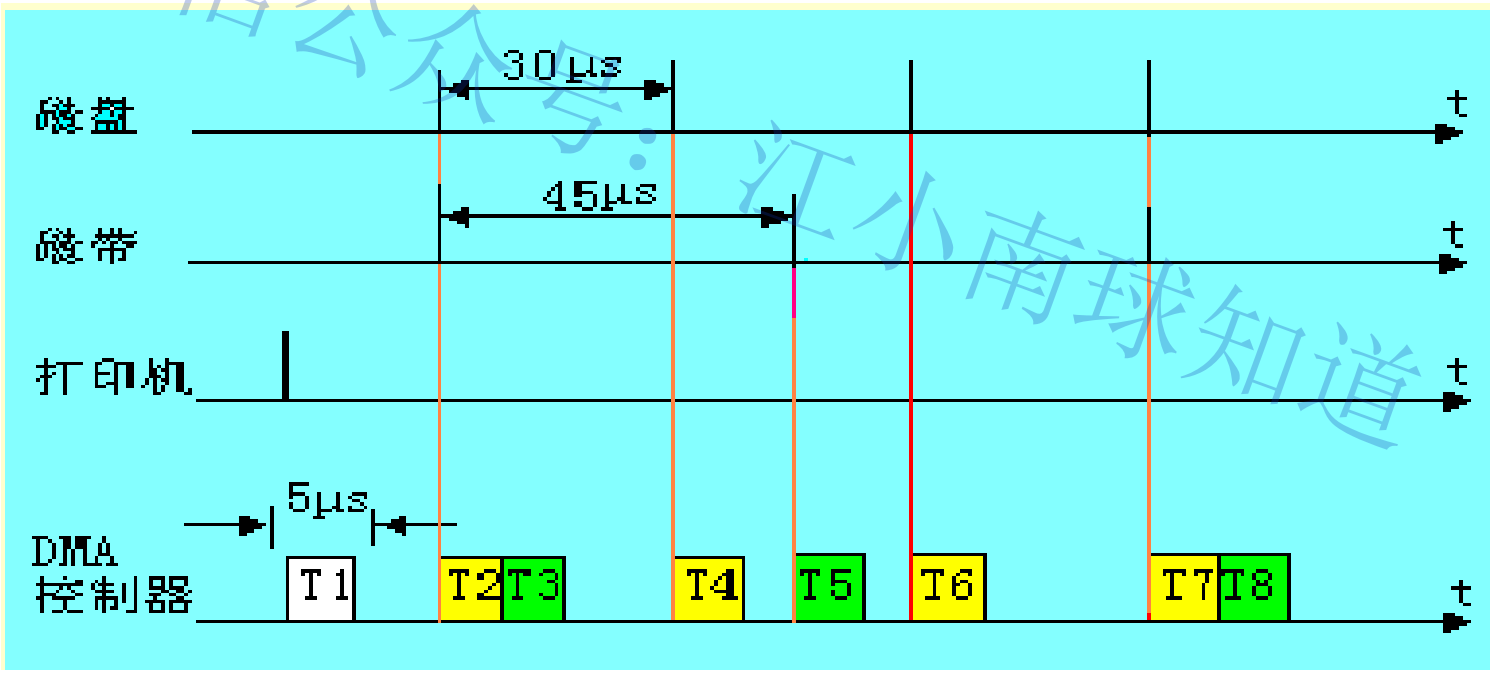
(6)实现方式不同：

中断主动通知，硬件、软件结合，信息交换在软件控制下完成

DMA中CPU不介入交换过程，不影响CPU的状态



例： 下图中假设有磁盘、磁带、打印机三个设备同时工作。磁盘以30μs的间隔向控制器发DMA请求，磁带以45μs的间隔发DMA请求，打印机以150μs间隔发DMA请求。根据传输速率，磁盘优先权最高，磁带次之，打印机最低，图中假设DMA控制器每完成一次DMA传送所需的时间是5μs。若采用多路型DMA控制器，请画出DMA控制器服务三个设备的工作时间图。



解：由图看出，T1间隔中控制器首先为打印机服务，因为此时只有打印机有请求。T2间隔前沿磁盘、磁带同时有请求，首先为优先权高的磁盘服务，然后为磁带服务，每次服务传送一个字节。在150μs时间阶段中，为打印机服务只有一次(T1)，为磁盘服务四次(T2，T4，T6，T7)，为磁带服务三次(T3，T5，T8)。从图上看到，在这种情况下DMA尚有空闲时间，说明控制器还可以容纳更多设备。

由于多路型DMA同时要为多个设备服务，因此对应多少个DMA通路(设备)，在控制器内部就有多少组寄存器用于存放各自的传送参数。

多路型DMA控制器的逻辑结构请见文字教材图8.17。通过配合使用I/O通用接口片子，多路型DMA控制器可以对8个独立的DMA通路(CH)进行控制，使外围设备以周期挪用方式对内存进行存取。

8条独立的DMA请求线或响应线能在外围设备与DMA控制器之间进行双向通信。一条线上进行双向通信是通过分时和脉冲编码技术实现的。也可以分别设立DMA请求线和响应线实现双向通信。每条DMA线在优先权结构中具有固定位置，一般DMA0线具有最高优先权，DMA7线具有最低优先权。

控制器中有8个8位的控制传送长度的寄存器，8个16位的地址寄存器。每个长度寄存器和地址寄存器对应一个设备。每个寄存器都可以用程序中的I/O指令从CPU送入控制数据。每一寄 存器组各有一个计数器，用于修改内存地址和传送长度。

当某个外围设备请求DMA服务时，操作过程如下：

（1）DMA控制器接到设备发出的DMA请求时，将请求转送到CPU。

（2）CPU在适当的时刻响应DMA请求。若CPU不需要占用总线则继续执行指令；若CPU需要占用 总线，则CPU进入等待状态。

（3）DMA控制器接到CPU的响应信号后，进行以下工作：① 对现有DMA请求中优先权最高的请求给予DMA响应；② 选择相应的地址寄存器的内容驱动地址总线；③ 根据所选设备操作寄存器的内容，向总线发读、写信号；④ 外围设备向数据总线传送数据，或从数据总线接收数据；⑤ 每个字节传送完毕后，DMA控制器使相应的地址寄存器和长度寄存器加“1”或减“1”。

以上是一个DMA请求的过程，在一批数据传送过程中，要多次重复上述过程，直到外围设备表示一个数据块已传送完毕，或该设备的长度控制器判定传送长度已满。

通道方式

1、通道的基本概念

什么是通道处理机？

“通道”是计算机系统中代替CPU管理控制外设的独立部件，是一种能执行有限I／O指令，并且能被多台外设共享的DMA专用处理机

为什么需要通道处理机？

进一步提高CPU的利用率,提高附加硬件的利用率

1)为了I/O与CPU、主存并行操作，以及让多用户或多道程序共同运行。

2)防止用户自行输入而破坏其他用户程序或系统程序及用户窃取系统不该让其读出的内容。

通道的功能

执行通道指令，组织外围设备和内存进行数据传输，按I/O指令要求启动外围设备，向CPU报告中断等，具体有以下五项任务：

(1)接受CPU的I/O指令，按指令要求与指定的外围设备进行通信。

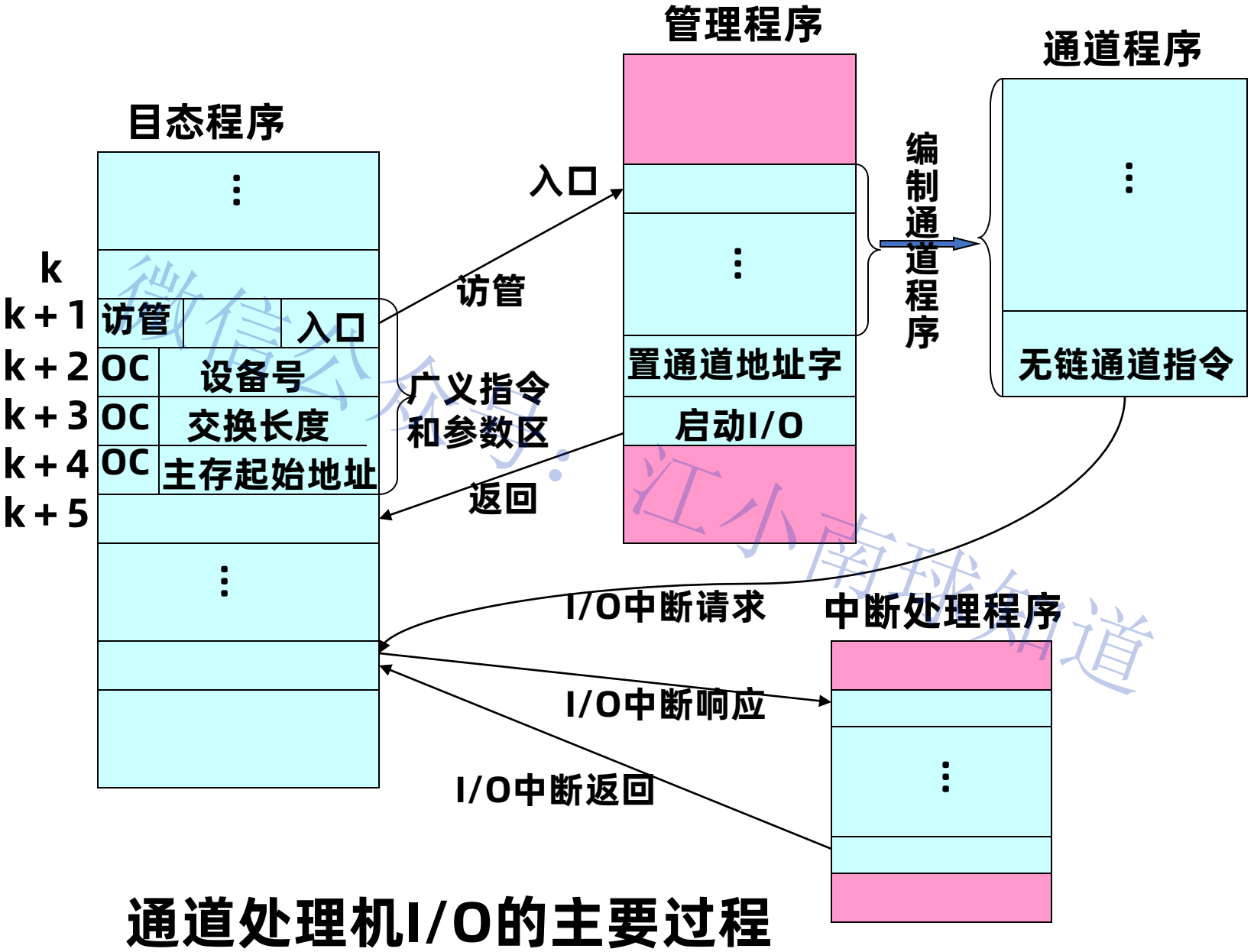
(2)从内存选取属于该通道程序的通道指令，经译码后向设备控制器和设备发送各种命令。

(3)组织外围设备和内存之间进行数据传送，并根据需要提供数据缓存的空间，以及提供数据存入内存的地址和传送的数据量。

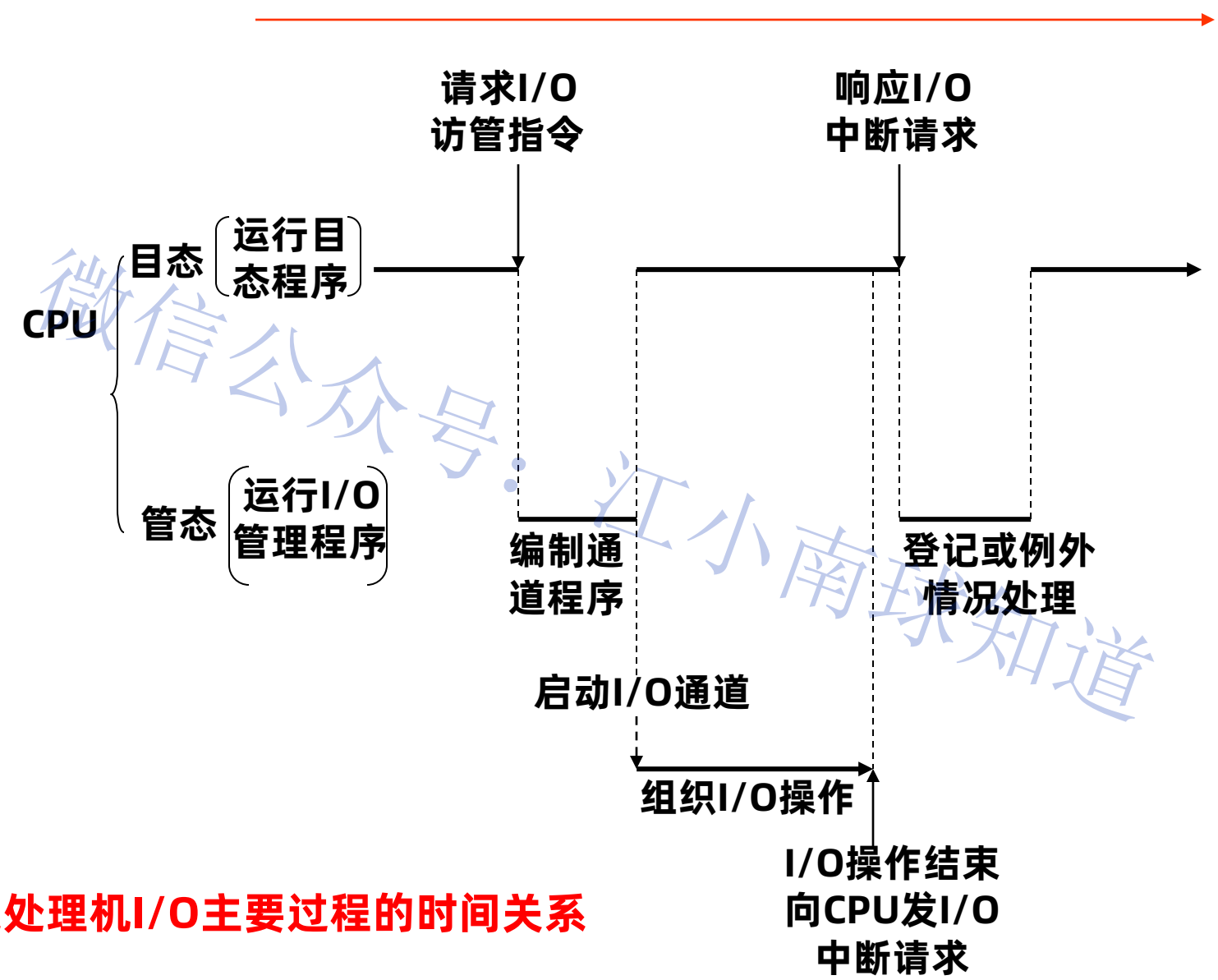
(4)从外围设备得到设备的状态信息，形成并保存通道本身的状态信息，根据要求将这些状态信息送到内存的指定单元，供CPU使用。

(5)将外围设备的中断请求和通道本身的中断请求，按次序及时报告CPU。

2.工作过程



通道处理机I/O的主要过程



通道处理机I/O主要过程的时间关系

3.优点：

1)完成一次I/O两次访管，减少对目态程序的干扰，提高了CPU运算和外设操作的重叠度。

2)各个通道可以有自己的通道程序，使多种、多台外设可以充分并行工作。

4.类型：

1)字节多路通道

适用于连接大量字符低速设备,传送一个字符或字节占用时间短,但等待时间长。数据通路宽度为单字节,采用字节交叉方式提高效率,或多个子通道独立并行工作。

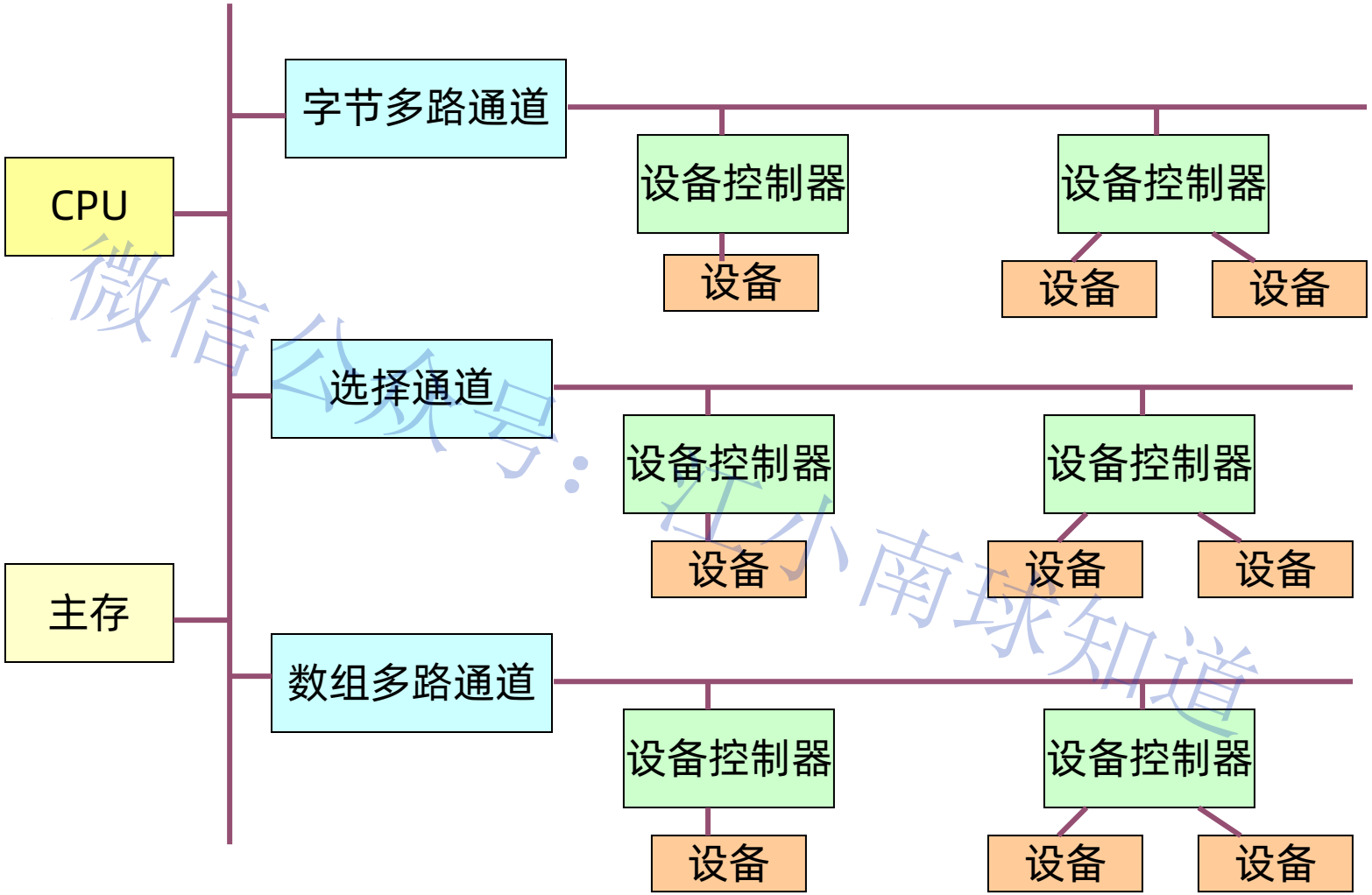
2)数组多路通道

适合于磁盘等高速设备,传送速率高,但传送前 辅助操作时间长。数据宽度为定长块,传送K个字节后重选设备进行下K个字节的传送。多个子通道分时共享I/O通路,成组交叉并行传送。

3)选择通道

适合于优先级高的高速设备，独占通道，只能执行一道通道程序。数据宽度为可变长块，一次将N个字节全部传送完毕，传送期内只选一次设备。

通道类型



5、通道流量分析

1.通道流量

通道在数据传送期内，单位时间所传送的字节数。它所能达到的最大流量称为通道极限流量。

2.影响极限流量的因素

1)工作方式

2)数据传送期内选择一次设备的时间T_S

3)传送一个字节的时间T_D

3. 极限流量

1)字节多路通道:每选一台设备传送一个字节。

$$f_{max.byte}=1/(T_S+T_D)$$

2)数组多路通道:每选一台设备传送K个字节。

$$f_{max.block}=k/(T_S+kT_D)=1/(T_S/k+T_D)$$

3)选择通道:每选一台设备就把N个字节传送完。

$$f_{max.select}=N/(T_S+NT_D)=1/(T_S/N+T_D)$$

若T_S,T_D一定，N>k，则：

$$f_{max.select} > f_{max.block} > f_{max.byte}$$

4.实际最大流量

1)字节多路通道:

$$f_{byte.j}=\Sigma f_{i.j}$$

2)数组多路通道:

$$f_{block.j}=\max f_{i.j}$$

3)选择通道:

$$f_{select.j}=\max f_{i.j}$$

$\sum_{i=1}^{p_j}$ → 工作于字节交叉方式，子通道独立各设备的字节传送率之和

$\max_{i=1}^{p_j}$ → 所接设备的字节传送率最大的那个

j——通道的号

$f_{i.j}$——通道上设备的字节传送率

p_j——通道上所接的总设备数

5.设计原则

1)极限流量大于等于实际最大流量

2)极限流量与实际最大流量的差值越小越好

$$f_{max.byte.j} >= f_{byte.j}$$
$$f_{max.block.j} >= f_{block.j}$$
$$f_{max.select.j} >= f_{select.j}$$

上述两个基本原则只是保证宏观上不丢失设备信息，并不能从微观上保证每一个局部时刻都不丢失信息。因为当设备要求通道的实际最大流量非常接近于通道设计所能达到的极限流量时，速率高的设备频繁发出请求而优先得到响应和处理，速率低的设备会因得不到通道而丢失信息。为此可在设备或设备控制器中设置一定容量的缓冲器以缓冲来不及处理的信息，或可动态改变设备响应优先级，使得低速设备也有机会得到通道而保证微观上不丢失信息。当然基本原则是一定要满足的，否则无论采用什么办法，设备总是要丢失信息的。

6.缺点：

1)并非独立的处理机，指令简单，无大容量存贮器。

2)I/O过程中需要CPU承担很多工作。

3)流水等组成技术因I/O中断而不能发挥作用，CPU速度严重下降。

4)访管中断转入I/O管理程序妨碍CPU资源的合理利用。

外围处理机(PPU)

为了克服通道处理机的缺点，希望CPU进一步摆脱数I/O操作的控制，以便更好地集中精力专注于自己的事情而发展了外围处理机(PPU)。

1.外围处理机的优点

1)更接近于一般的处理机，指令丰富，功能强。

2)独立于主处理机异步工作。

3)可以与主处理机共享或不共享主存。

4)可以自由选择通道和设备进行灵活通信。

2.缺点：

就硬件利用率和成本来讲不如通道处理机好，但随着器件技术不断提高，成本在逐渐降低。

例：一个字节多路通道连接D1、D2、D3、D4、D5共5台设备，这些设备分别每10微秒、30微秒、30微秒、50微秒和75微秒向通道发出一次请求。问

1、通道的实际流量和工作周期

2、设通道的实际流量等于通道的极限流量，数据传输率高的设备，优先级高。在时刻0同时第一次向通道提出请求，画出时间关系图

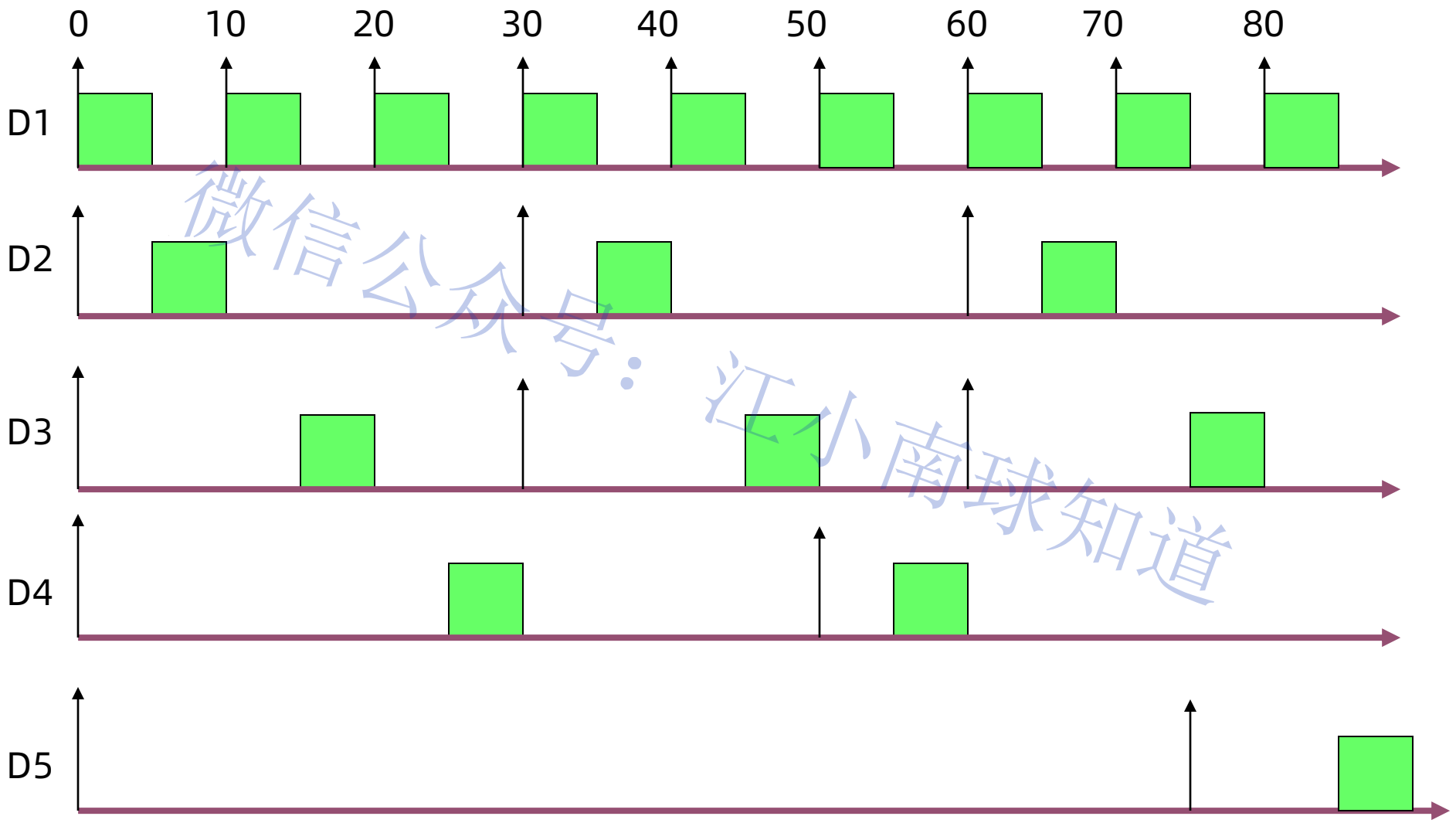
3、时间关系图上发现什么问题？如何解决？

解：

实际流量　　fbyte=(1/10+1/30+1/30+1/50+/1/75)=0.2MB/s

工作周期　　t=1/fbyte =5微秒/字节

2、时间关系图



3、时间关系图上发现什么问题？

D5的第一次请求未能得到通道的响应如何解决？

1）增加通道的最大流量。

2）动态改变设备的优先级。

3）增加一定数量的数据缓冲寄存器。

(1)程序查询方式

程序查询方式是早期计算机中使用的一种方式。数据在CPU和外围设备之间的传送完全靠计算机程序控制。

(2)程序中断方式

中断是外围设备“主动”通知CPU，准备送出输入数据或接收输出数据的一种方法。

(3)直接内存访问(DMA)方式

直接内存访问(DMA)方式是一种完全由硬件执行I/O交换的工作方式。

(4)通道方式

DMA方式的出现已经减轻了CPU对I/O操作的控制，使得CPU的效率有显著的提高，而通道的出现则进一步提高了CPU的效率。这是因为，CPU将部分权力下放给通道。

通道是一个具有特殊功能的处理器，某些应用中称为**输入输出处理器(IOP)**，它可以实现对外围设备的统一管理和外围设备与内存之间的数据传送。

这种提高CPU效率的办法是以花费更多硬件为代价的。

(5)外围处理机方式

外围处理机(PPU)方式是通道方式的进一步发展。由于PPU基本上独立于主机工作，它的结构更接近一般处理机，甚至就是微小型计算机。在一些系统中，设置了多台PPU，分别承担I/O控制、通信、维护诊断等任务。

从某种意义上说，这种系统已变成分布式的多机系统。

(6) 各种方法比较：

- 程序查询方式和程序中断方式适用于数据传输率比较低的 外围设备，
- DMA方式、通道方式 和PPU方式适用于数据传输率比较高的设备。

目前，单片机和微型机中多采用程序查询方式、程序中断方式和DMA方式。通道方式和PPU方式大都用在中、大型计算机中。