第 9 章 I/O系统组织

本章学习内容

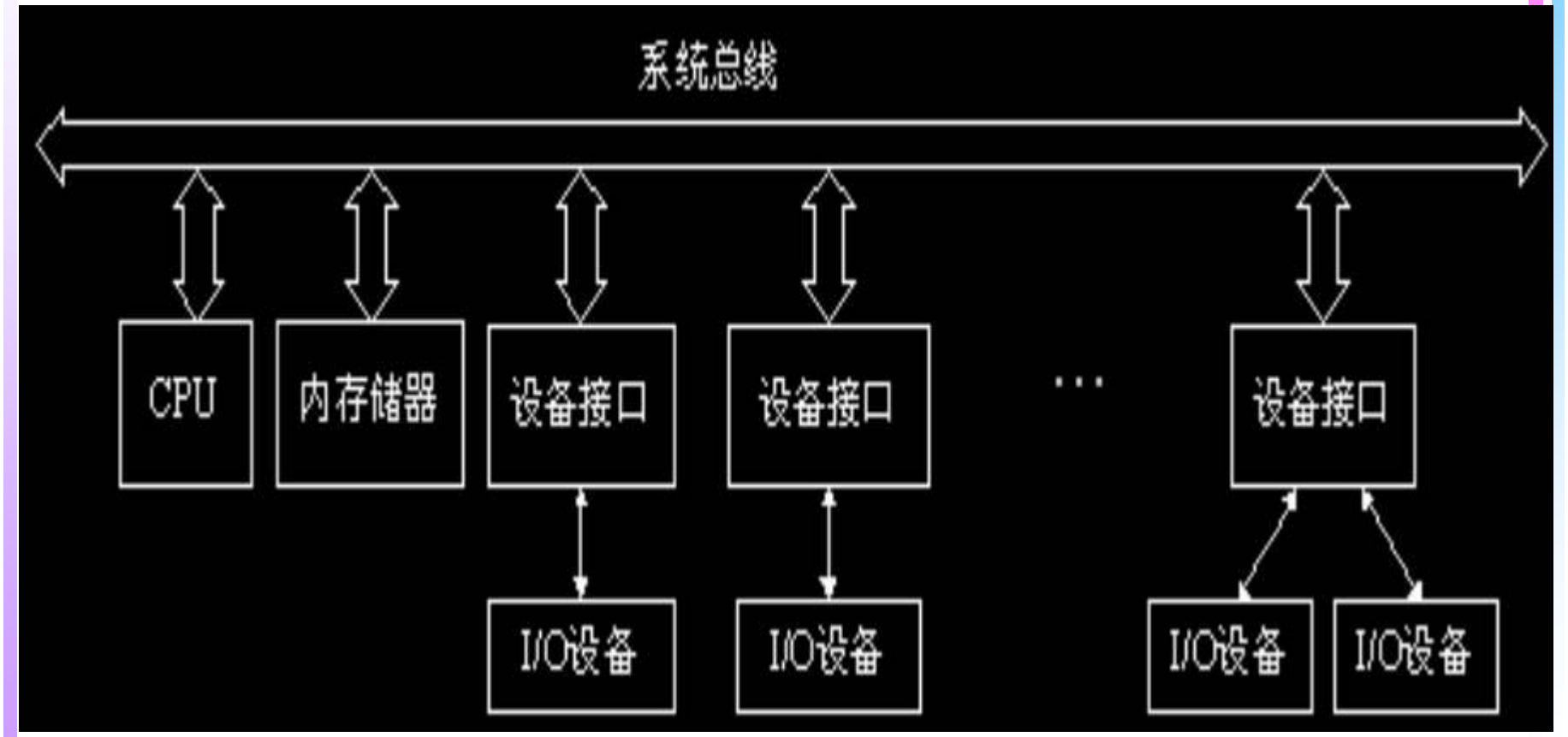
- I/O系统的功能
- 接口的功能
- 中断的基本概念
- DMA的基本概念
- 通道的基本概念

9.1 I/O系统概述

- 9.1.1 需解决的主要问题
- 计算机系统中的I/O系统，主要用于解决主机与外部设备间的信息通讯，使外围设备与主机能够协调一致地工作。
- I/O系统的基本功能：
- ① 选择输入/输出设备
- ② 信息交换。

9.1.2 I/O系统的组成

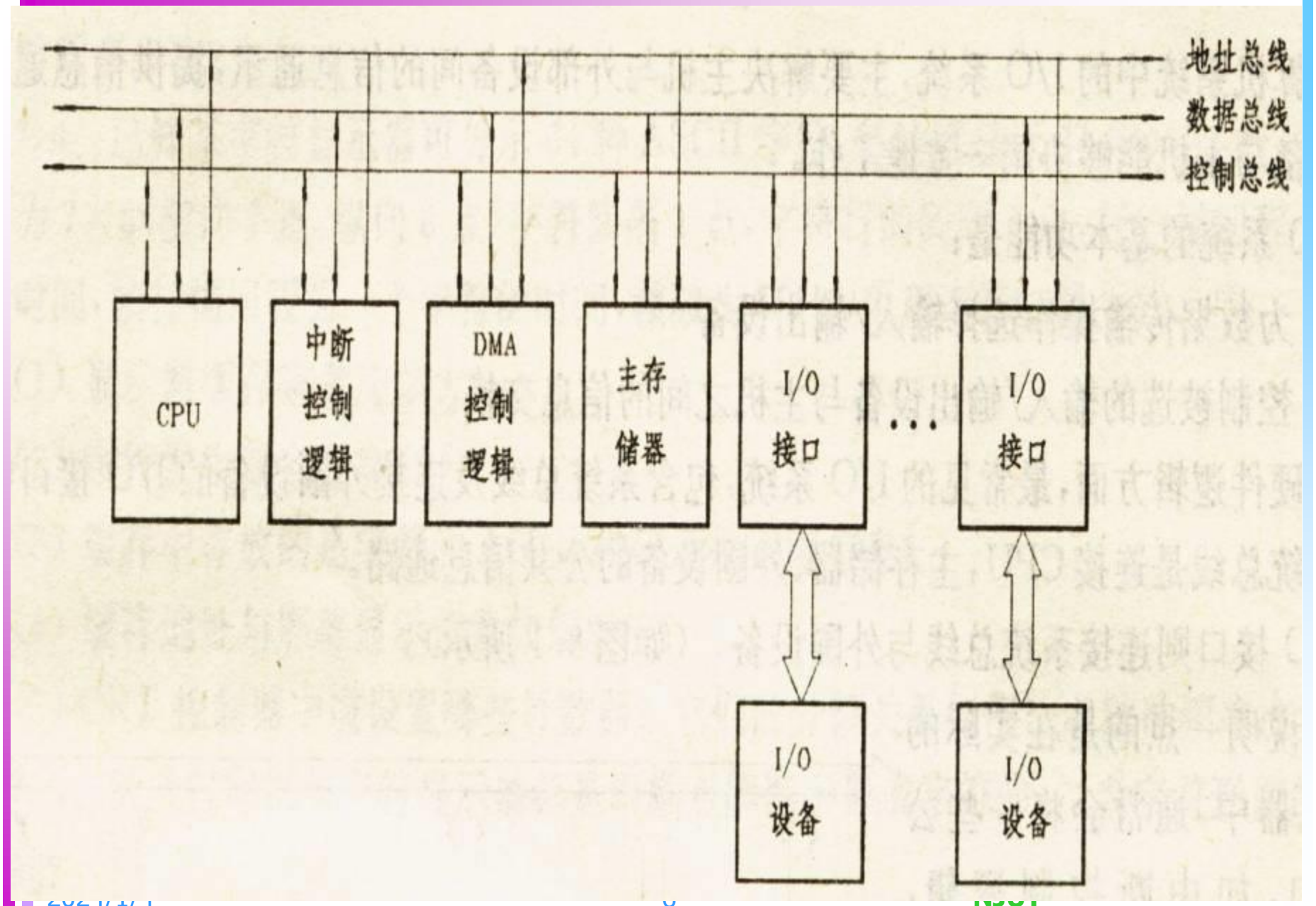
包括:系统总线、I/O接口、I/O设备等



9.1.3 主机与外设间的连接模式与组织管理

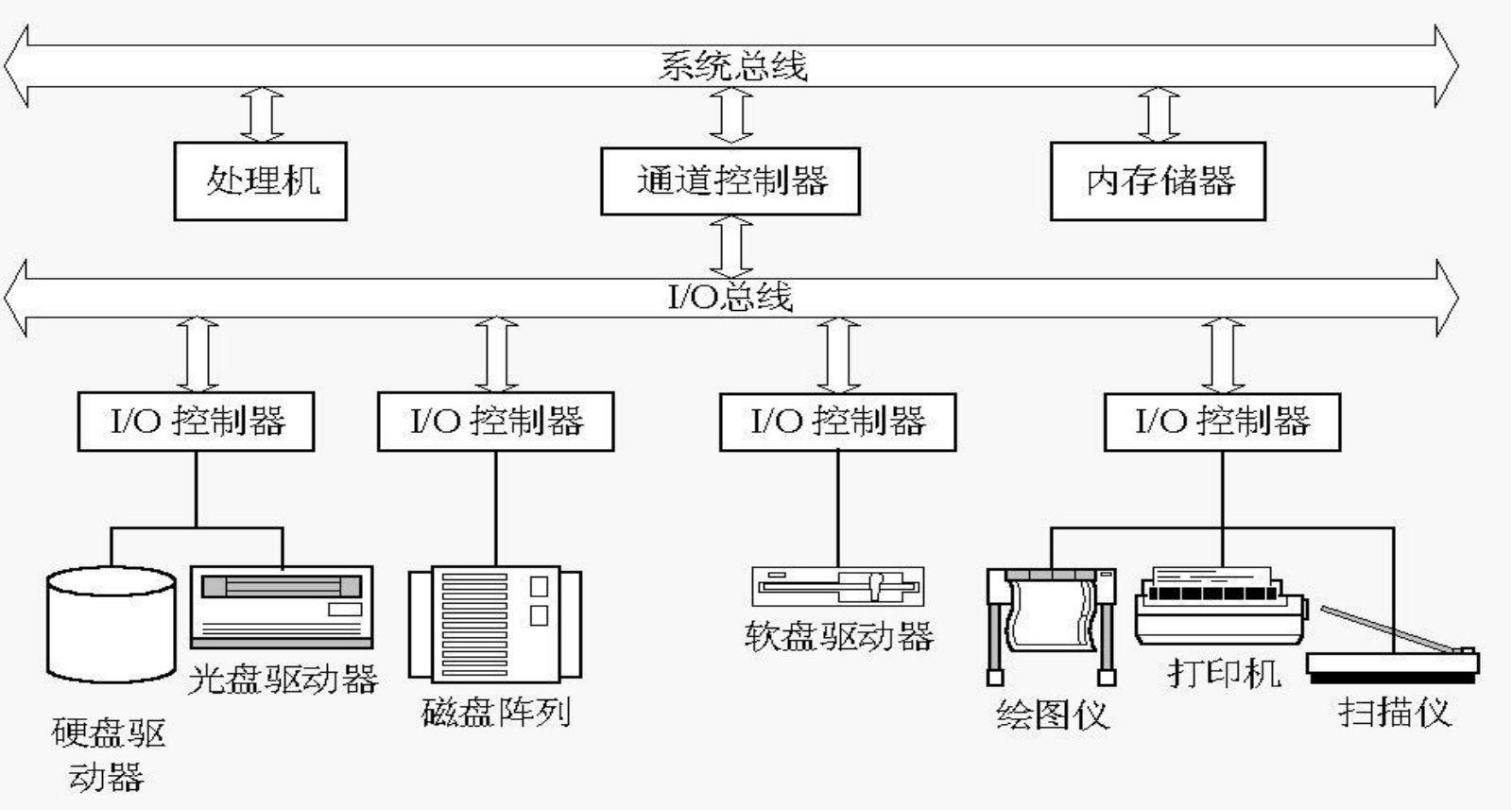
- 1．总线型连接方式
- CPU通过系统总线与主存储器、I/O接口电路相连接，通过I/O接口电路实现对外设的控制。

- 优点：系统模块化程度较高，I/O接口扩充方便。
- 缺点：系统中部件之间的信息交换，均依赖于总线，总线成为系统中的速度瓶颈，因而不适于系统需配备有大量外设的场合。


地址总线
数据总线
控制总线
CPU
中断控制逻辑
DMA控制逻辑
主存储器
I/O接口
I/O接口
I/O设备
I/O设备

2. 通道控制连接方式

- 主要用于大型主机系统中，一般所连接外设数量多、类型多以及速度差异大。



- 通道控制器是一种专门负责I/O操作控制的控制器，它通过执行由专门的通道指令编制的并存放在内存之中的通道程序实现对外设的控制。在这种I/O控制方式下，由通道控制器控制实现主存储器与外部设备之间的直接数据交换，CPU不再负责具体的I/O控制，实现了处理机与通道控制器和外设的并行工作。

3. I/O处理机控制连接方式

- **I/O**处理机与通道相比，有更强的独立性，可视为一种专门的**CPU**。它适应性强，通用性好。**I/O**处理机可大可小，可以是一台计算机，也可以是微处理器，如**Intel8089**。

9.2 I/O接口

- 接口：通常指设备(硬件)之间的界面。
- I/O接口:主机(系统总线)与外部设备或其它外部系统之间的接口逻辑。

9.2.1 接口的基本功能

- (1) 识别设备地址，选择指定的设备
- (2) 实现数据的传送与缓冲
- (3) 控制主机与外设之间的通信联络，实现控制命令和状态信息的交换，保证时序协调。
- (4) 实现信号形式和数据格式转换

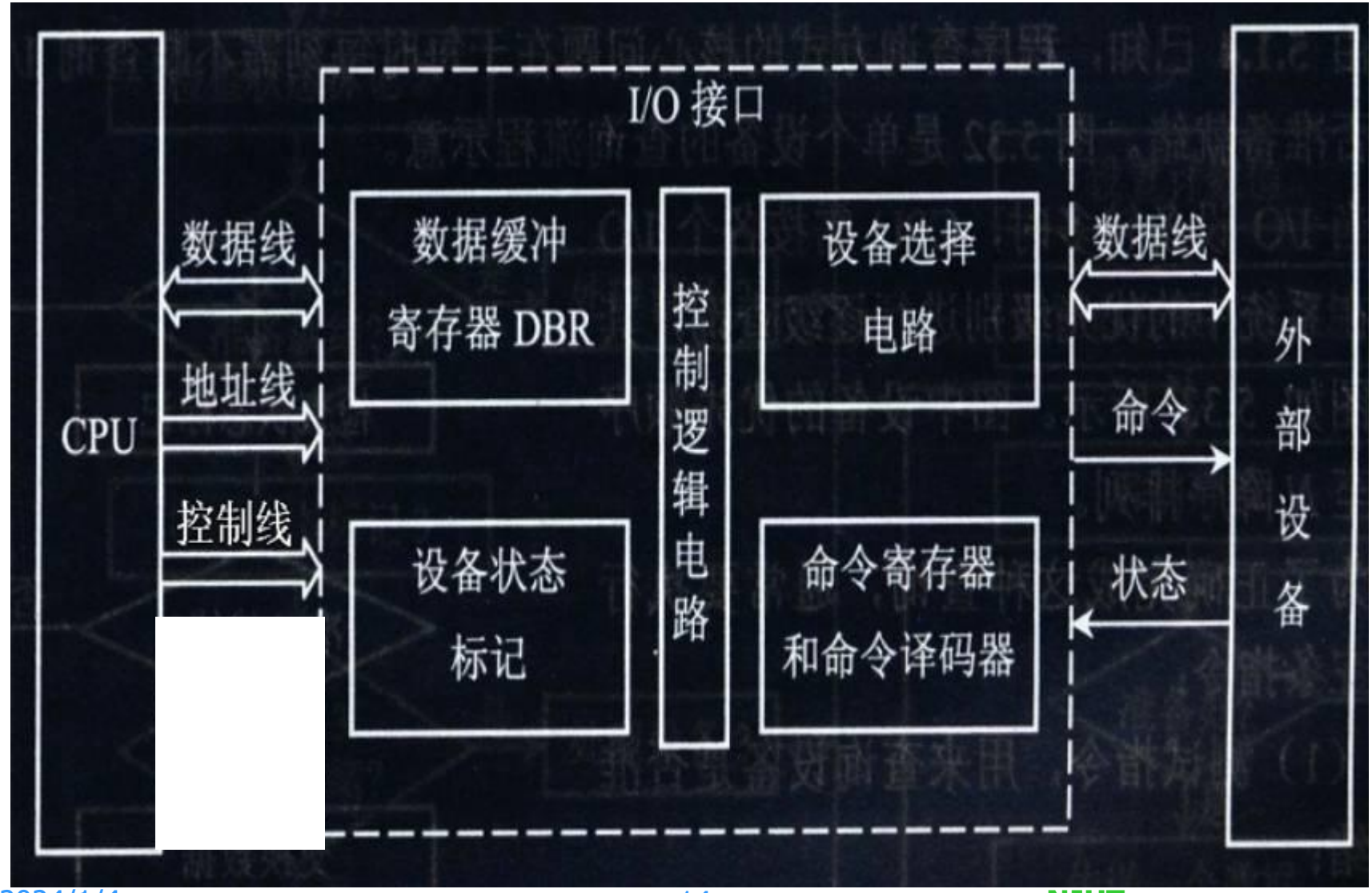
CPU和外设之间传送的信息

- (1) 数据信息
- ① 数字量
- 数字量是指用二进制码形式提供的信息，如以ASCII码表示的字符。通常有8位、16位等。
- ② 模拟量
- 模拟量是指连续变化的物理量，如温度、湿度、压力等。计算机无法直接处理模拟量，要经过A/D变换变成数字量，才能送入计算机。
- ③ 开关量
- 开关量有两个状态：“0”和“1”，可以用1位二进制数表示。具有两种状态的量，如开关的闭合和断开可用开关量表示。

- (2) 状态信息
- 状态信息反映当前外设所处的工作状态。
- 在与外设进行数据信息交换时，CPU需要通过状态信息了解外设的工作状态。通常外设用准备好（READY）信号来表明是否准备就绪；用忙（BUSY）信号表示是否处于空闲状态。外设的状态信息通过接口送往CPU。
- (3) 控制信息
- 在外设的工作过程中，CPU需要通过控制信息控制外设的工作，如对外设的启动和停止等。CPU需要通过接口将控制信息传送给外设。

- 状态信息、控制信息也通过数据总线来传送。为了区别这三种信息，在接口线路中将它们分别送入不同的寄存器。CPU同外设之间的信息传送实质上是对相应的寄存器进行“读”或“写”操作。接口中这些可以由CPU进行读或写的寄存器被称为“端口”（Port）或I/O端口。

接口的基本组成


I/O 接口
CPU
数据线
地址线
控制线
数据缓冲
寄存器 DBR
设备状态
标记
控制逻辑电路
设备选择
电路
命令寄存器
和命令译码器
数据线
命令
状态
外部设备

- (1) 设备选择电路
- 用于接收总线传来的地址信息，经译码后，决定选择哪个设备或I/O接口内部的部件。
- (2) 数据缓冲寄存器（数据端口）
- 用于存放主机与外设之间要传递的数据信息。
- (3) 命令寄存器（控制端口）
- 用于存放主机向外设发送的控制命令。
- (4) 状态寄存器（状态端口）
- 用于存放外设或接口的工作状态。

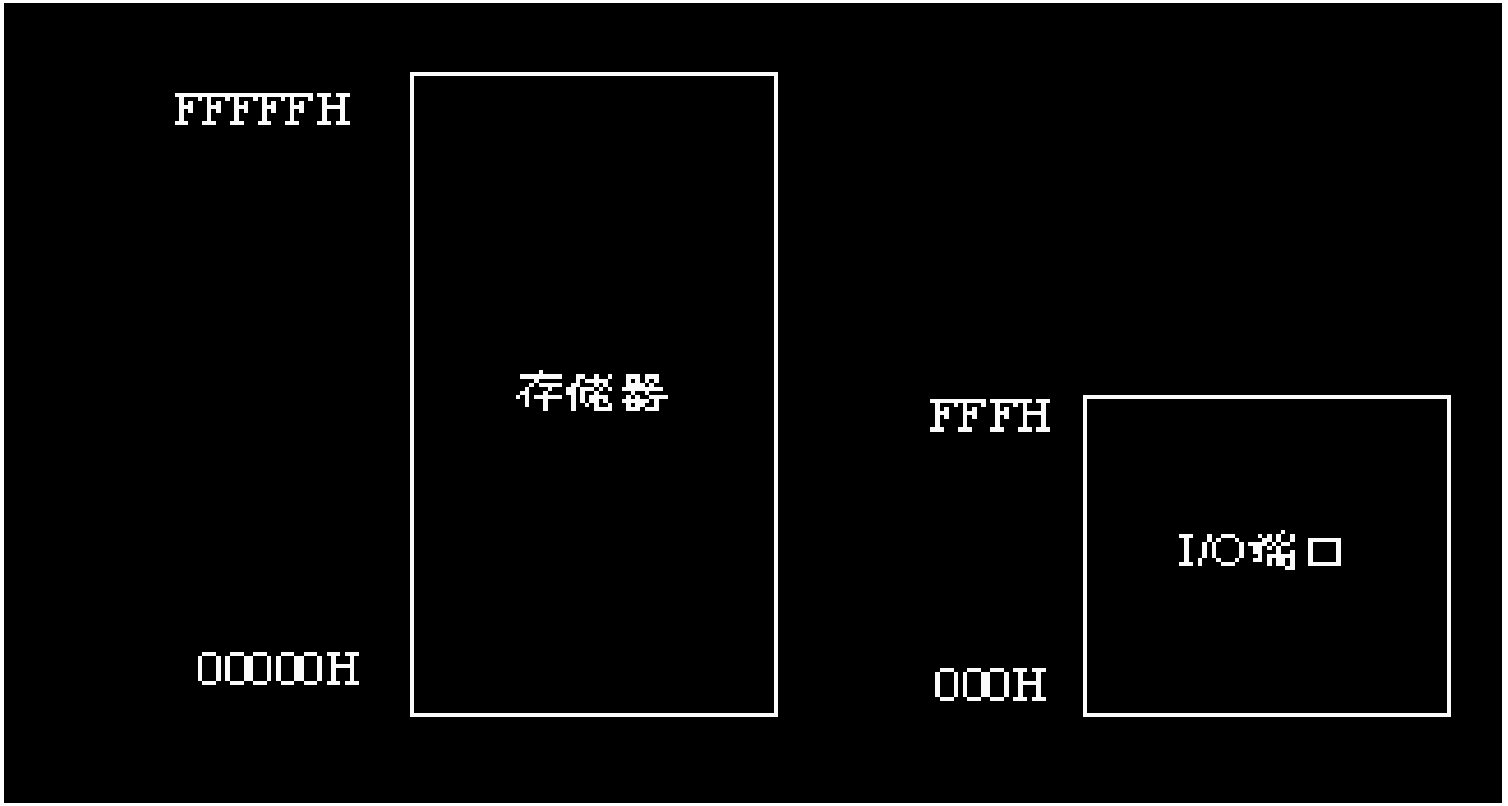
I/O端口的寻址方式

- 对I/O设备的寻址实质上就是对I/O端口的寻址。
- (1) I/O端口与主存统一编址
- 将一个I/O端口作为存储器中的一个单元对待，每一个I/O端口占用一个存储器单元地址。编址时将I/O端口与存储器单元一起进行编址。
- 由于编址时I/O端口与存储器单元不加区分，因此CPU可以用所有访问存储器的方法访问I/O端口，即所有访问存储器的指令均可用来访问I/O端口。
- 这种方式又称存储器映象编址方式。

- I/O端口与主存统一编址方式的优点：
- ① CPU可使用所有存储器操作指令对I/O端口中数据进行操作，十分灵活和方便。
- ② 不需要用专门的指令及控制信号区分是存储器还是I/O操作。使得系统相对简单。
- 这种方式存在的问题是：
- ① I/O端口占用了内存单元的部分地址空间，使内存容量减小。
- ② 由于在程序中不易分清指令访问的是存储器还是I/O端口，所以采用这种方式编制的程序不易阅读。

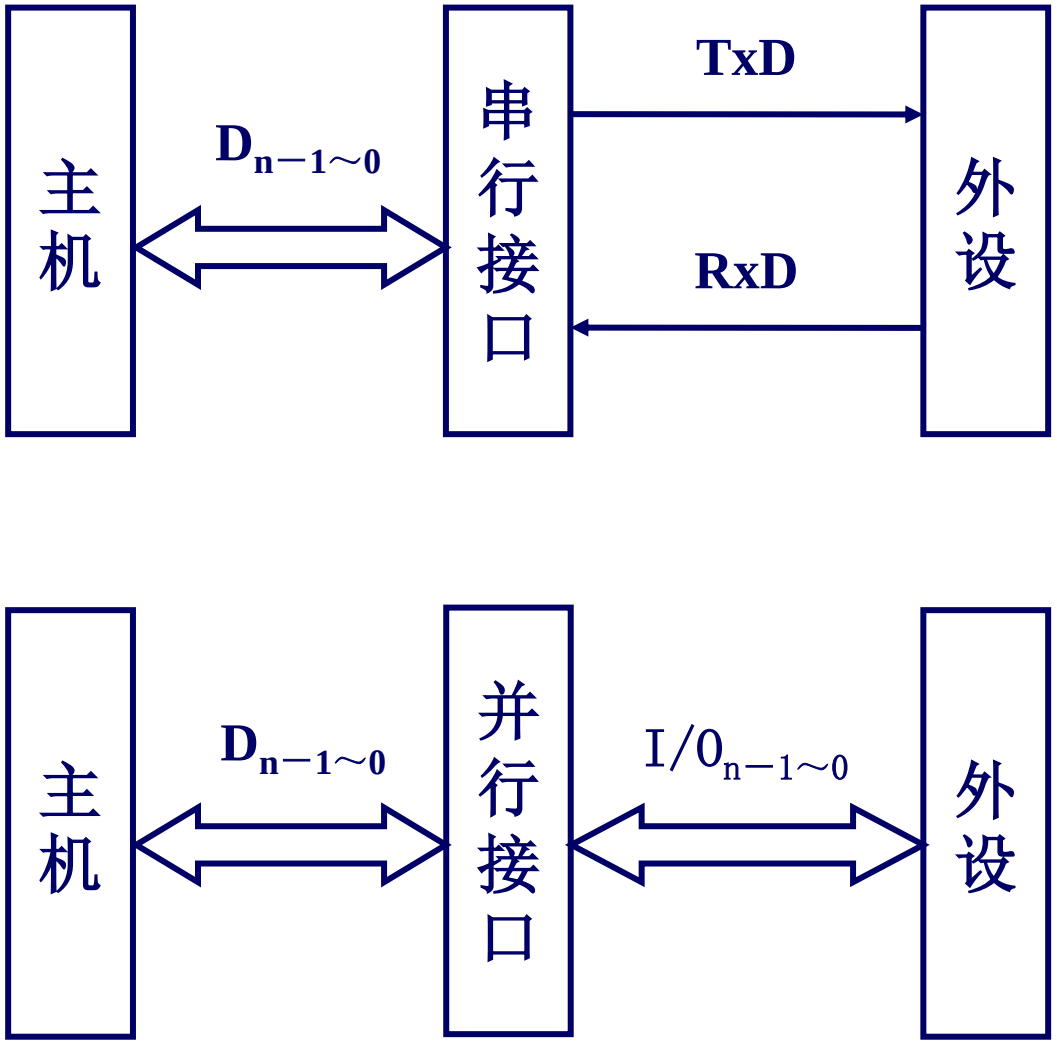
- (2) I/O端口独立编址
- 将I/O端口与存储器单元分别独立进行编址，CPU访问外设时，需使用专门的I/O指令，并需要有与接口电路联系的单独的控制信号。也称为I/O端口寻址输入输出方式。
- I/O端口独立编址方式的优点：
- ① I/O端口具有独立的地址空间，不占用内存空间。
- ② I/O指令中的地址字段的长度较短，可以节省指令存储空间和指令执行时间。
- ③ 由于访问存储器和访问I/O端口使用不同的指令，因此编制的程序比较清晰易读。

- I/O端口独立编址方式的缺点：
- I/O操作指令的种类通常没有存储器操作指令丰富，设计程序时不够方便。



9.2.2 I/O 接口的分类

- (1)按数据传送格式分
- 串行接口：接口与设备之间的信息传送是逐位串行进行的。8251
- 并行接口：接口与设备之间的信息传送是将一个字或一个字节的所有位同时并行地进行传送的。8255


主机
$D_{n-1\sim0}$
串行接口
TxD
RxD
外设
主机
$D_{n-1\sim0}$
并行接口
$I/O_{n-1\sim0}$
外设

- （2）按时序的控制方式分
- 同步接口：一般与同步总线相连，接口与总线采用统一时钟，CPU与I/O设备，存储器与I/O设备交换信息，都与总线同步时钟脉冲同步。同步接口的控制简单，但要求在速度上匹配。
- 异步接口：与异步总线相连，接口与系统总线之间采用异步应答方式。
- 通常主设备提出 “请求”信号，经总线和接口传递到从设备，从设备完成主设备指定的操作后，向主设备发出“回答”信号。
- 交换过程是 “请求”、“回答”地进行着，而非系统定时节拍的硬性规定。

- 三种异步应答方式：
- **(a)**不互锁
- 　请求与回答信号都有一定的时间宽度（主从设备在发送信号无需等待对方确认，当自己认为对方收到信号后便开始发送下一次信号。这个过程中，主设备不论从设备是否接受到了请求信号，都会撤销请求信号，从设备不论主设备是否接受到了应答信号，都会撤销应答信号。）

- **(b)** 半互锁
- 请求信号接到回答信号后再撤消（主设备发送请求信号后必须收到从从设备的应答才可以撤销请求信号，存在互锁关系。而从设备发出应答信号后无需等待主设备确认即刻撤销其回答信号，不存在互锁关系。）

- **(c)** 全互锁
- 请求信号接到回答信号后撤消，请求的撤消又导致回答信号的撤消（当主设备发送请求信号后必须等待从设备发送应答信号后才能撤销信号，存在互锁关系。当从设备发送应答信号后必须等待主设备确认，当收到确认信号后才能撤销信号，也存在互锁关系。双方都存在互锁关系，故称为全互锁关系。双向奔赴，有应答时才会做出反应）

(3)按交换控制方式来分（5种）

- 直接程序控制方式
- (程序)中断传送方式
- 直接存储器存储方式(DMA方式)
- 通道方式
- I/O处理机方式

9.3 程序控制方式

- I/O信息交换方式就是数据传送控制方式，通常分为五种方式。

- 1直接程序控制方式：完全由软件实现
- 2程序中断方式：软硬结合，以软件为主。

适用于慢速外设

- 3DMA方式：
- 4通道方式
- 5I/O处理机方式

采用辅助硬件实现

适合快速外设

快慢设备均适用

适合于大、中型机

适合于小微型机

9.3.1 直接程序控制方式

- 直接程序控制方式：指CPU直接利用I/O指令编程，实现数据输入输出。
- 直接程序控制方式可分为两种传送方式。

1. 直接传送方式

- 直接传送方式：CPU在控制与外设之间的数据传送之前，不需了解外设的工作状态，即可直接执行I/O指令，实现数据传送。
- 直接传送方式无需查询设备的任何状态，又称为无条件传送方式。多用于I/O操作时间固定且已知的情况下。

2. 程序查询方式

- 程序查询方式：CPU在进行输入/输出操作之前，先查询外设的状态，只有当外设准备就绪时，才进行数据传送。又称为条件传送方式。
- 当有关操作的时间未知或不定时，往往采用程序查询方式进行同步控制。

- (1)程序查询的基本流程(步骤)
- ①向I/O设备发启动命令;
- ②查询设备状态;
- ③I/O设备未好, 回②;
- ④I/O准备好, 执行I/O, 传送数据;
- ⑤结束否, 若未结束, 则转②;
- ⑥流程结束.
- (见下一页图)

程序查询方式I/O程序的操作步骤

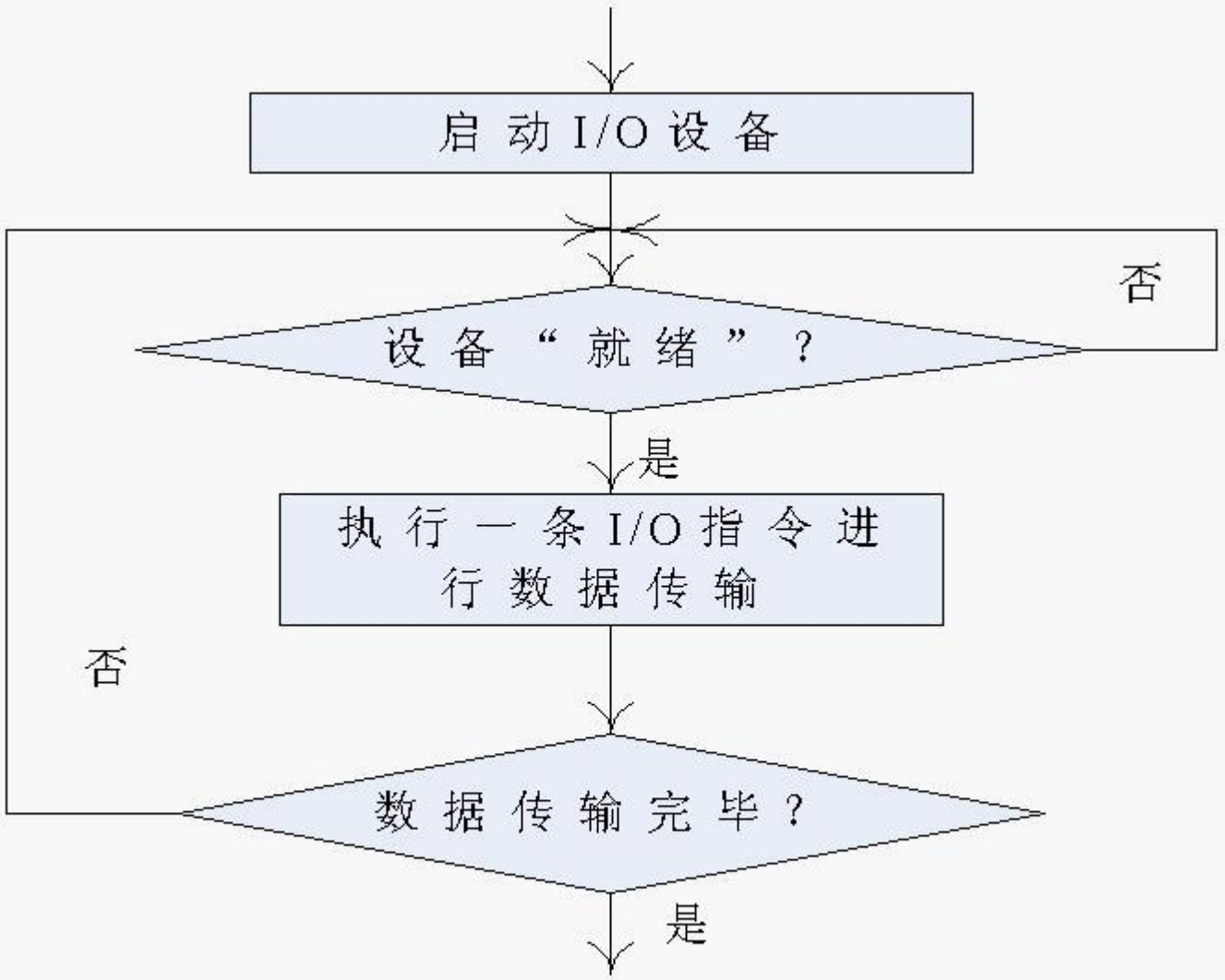
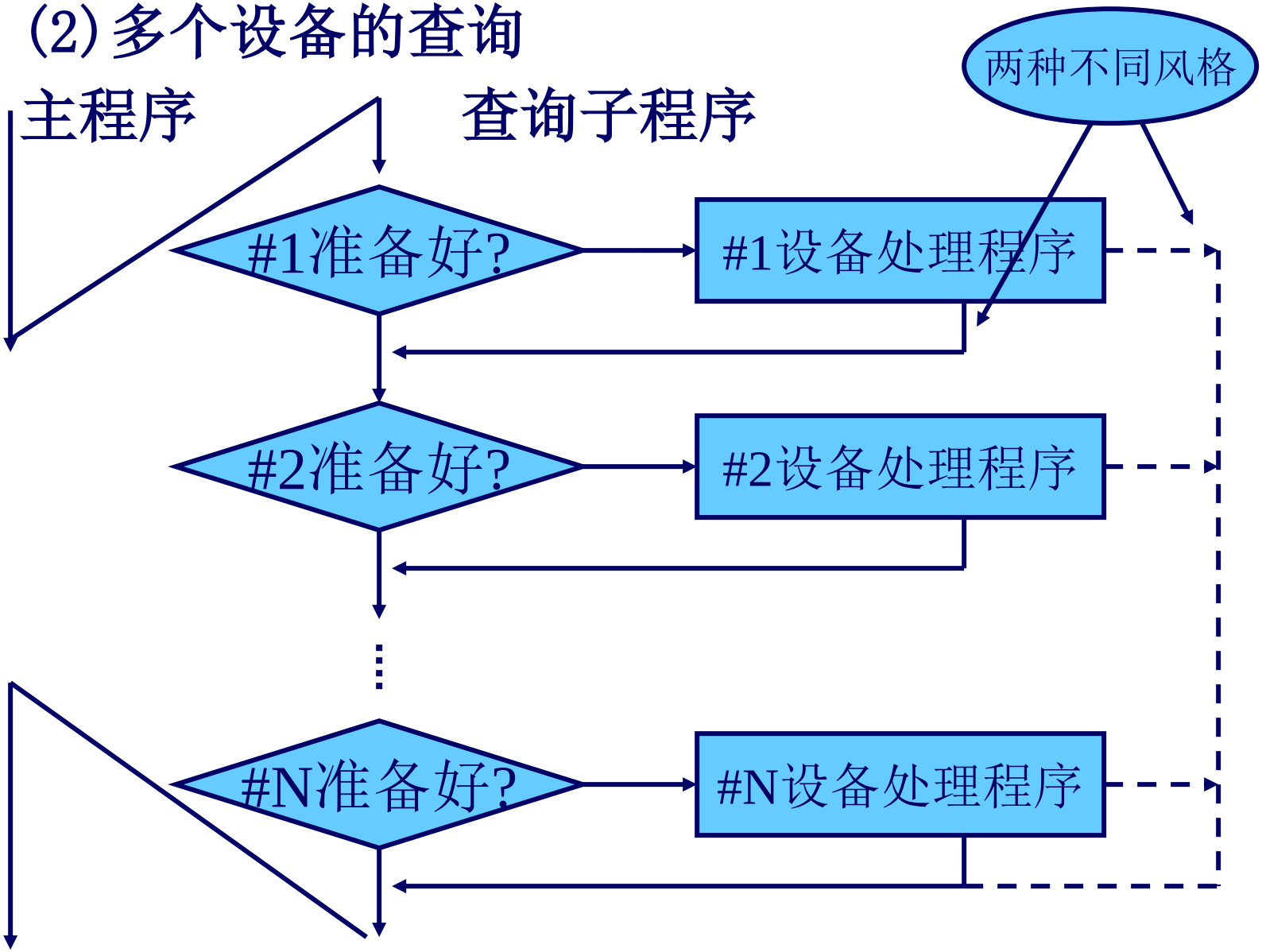


图 9-6 程 序 查 询 方 式

- (2) 多个设备的查询
- 主程序　　查询子程序



- 这种方式的优点：接口简单
- 这种方式的缺点：
- ① CPU与外围设备无法并行工作，CPU效率很低。
- ② 无法发现和处理异常情况，不能响应来自外部的随机请求。

- 解决方法：中断方式

1. 已知 x=+0.0101100 (8 位二进制真值)，试求$[x]_{补}$、$[-x]_{补}$、$[x/2]_{补}$、$[2x]_{补}$、$[-2x]_{补}$和$[4x]_{补}$。

（或 X 为分数：X=+11/32）　　$[x]_{原}$、$[-x]_{原}$

2. 根据下表中给定的机器数（整数），分别写出把它们看作原码、反码、补码表示形式时所对应的十进制真值。（无符号数？）

表示形式 / 机器数	原码表示	反码表示	补码表示
01111110			
11111111			

1 答： $[x]_{补} = 0.0101100$　　$[-x]_{补} = 1.1010100$　　$[x/2]_{补} = 0.0010110$

$[2x]_{补} = 0.1011000$　　$[-2x]_{补} = 1.0101000$　　$[4x]_{补}$ 溢出

2 答：

表示形式 / 机器数	原码表示	反码表示	补码表示
01111110	+126	+126	+126
11111111	-127	-0	-1

5-1．某磁盘组有 5 片磁盘，每片有两个记录面，最上最下两个面不用。每面有 256 个磁道，每个磁道分为 32 个扇区，每个扇区包括 512 字节，已知磁盘内磁道直径为 10 英寸，外磁道直径为 16 英寸，转速为 50 转/秒，试计算下列参数。

（1）该磁盘组最大存储容量（MB）

（2）该磁盘组最大位密度(bpi，只取整数)

（3）该磁盘的数据传输率(KB/s)

（4）该磁盘组共有多少个柱面？如果某文件长度超过一个磁道的容量，应将它记录在同一个存储面上，还是记录在同一个柱面上?

5-1 答：（1）该磁盘组最大存储容量是：C＝(2×5-2)×256×32×512B＝2^{25}B＝32MB

（2）最大位密度 32×512×8/10π＝4174 位/英寸＝4174bpi

（3）数据传输率：Dr＝32×512×8×50＝6553600bit/s＝819200B/s＝800KB/s

(4) 该磁盘组共有 256 个柱面， 如果某文件长度超过一个磁道的容量，应将它记录在同一个柱面上。

5-3. 某磁盘组有 7 片磁盘，每片有两个记录面（最上下两个面不可用），存储区域内直径为 20cm，外直径为 30cm，道密度为 4tpmm（道/毫米），内层位密度为 30pmm（位/毫米），转速为 40r/s(转/秒)，问：

(1) 该磁盘组共有多少存储面可用？共有多少柱面？

(2) 该磁盘组总存储容量是多少？（MB）

(3) 该磁盘的数据传输率是多少？(KB/s)

(4) 该磁盘组最小位密度(bpmm)

5-3 答：

(1) 该磁盘组共有 7*2-2=14 个存储面可用

有效存储区域为（15-10）cm=5cm=50mm，因为道密度=4tpmm，故共有 4tpmm×50mm=200 道，即共有 200 个圆柱面。

(2)

内层磁道周长为 2×3.14×10cm=62.8cm=628mm

每道信息量=30bpmm×628mm=18840b =2355B

每面的信息量=18840b×200=3768000b =471000B

该磁盘组总容量=471000B×12=5652000B=5519.5KB=5.39MB

(3) 每条磁道容量 N=2355B，转速 r=40r/s

该磁盘组数据传输率 C=Nr=2355B×40r/s=94200B/s =92KB/s

(4) 该磁盘组最小位密度=(20×3.14×30bpmm)/(30×3.14)=20bpmm(位/毫米)

9.3.2 程序中断传送方式（重点）

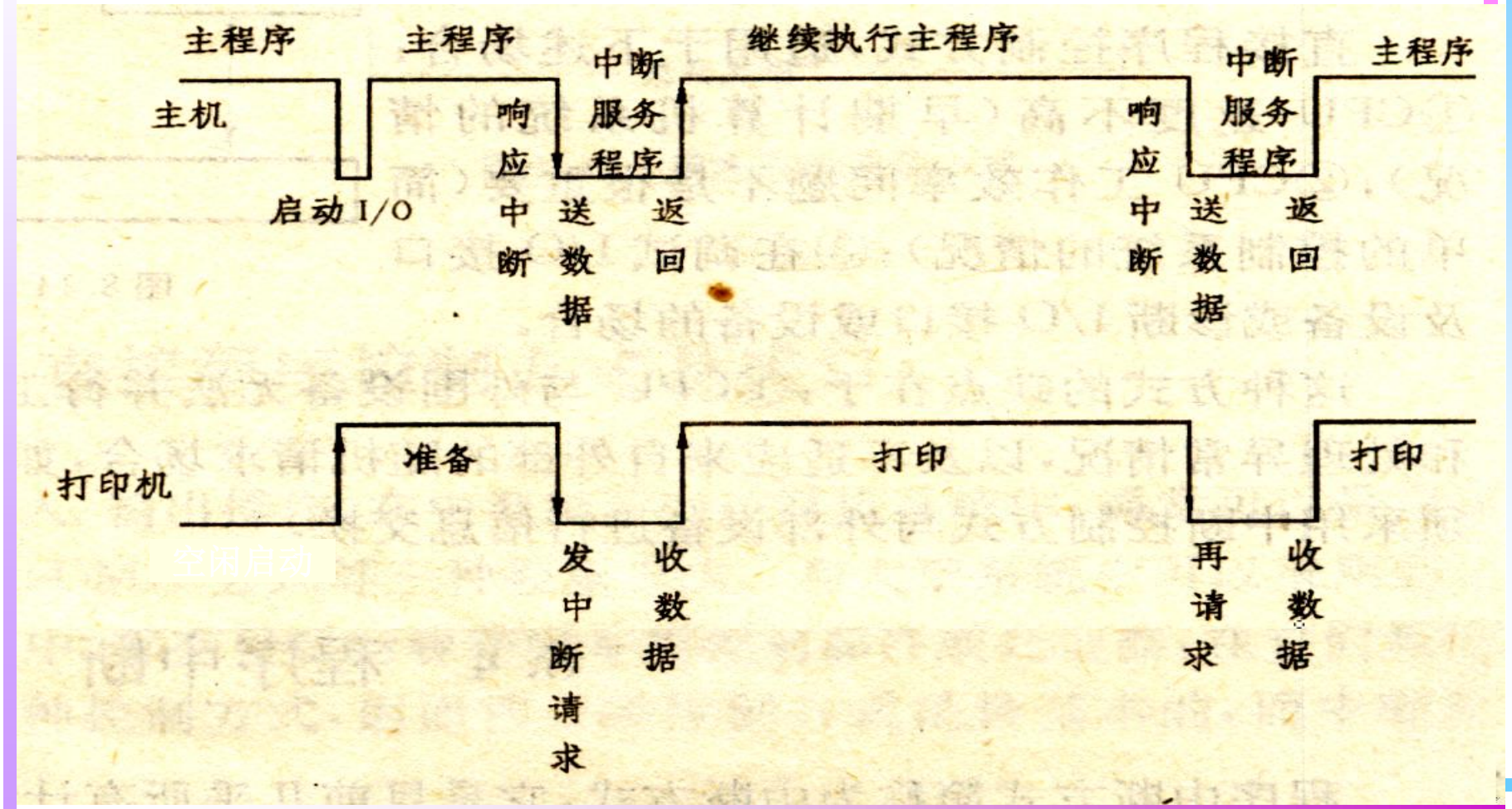
- 程序中断方式简称为中断方式，它是目前几乎所有计算机系统都具备的一种重要工作机制。
- 中断不仅用在输入输出过程控制中，而且在多道程序、分时操作、实时处理、人机联系、故障处理、程序的监视与跟踪等方面都起着十分重要的作用。

1 中断的基本概念（概念很多）

- 1）中断
- 中断是指处理机暂时中止执行现行程序而转去执行处理更加紧迫事件的服务程序，待处理完毕后，再自动返回执行原来的程序的过程。
- 2）实现中断应考虑的问题
- (1) 保存现场
- 中断过程实质上是一种程序切换过程，因此必须处理好保存旧现场、建立新现场的问题。
- (2) 及时获得中断请求信号
- 中断具有随机性(程序自愿中断除外)。因此必须及时检测中断请求信号，以便及时处理。

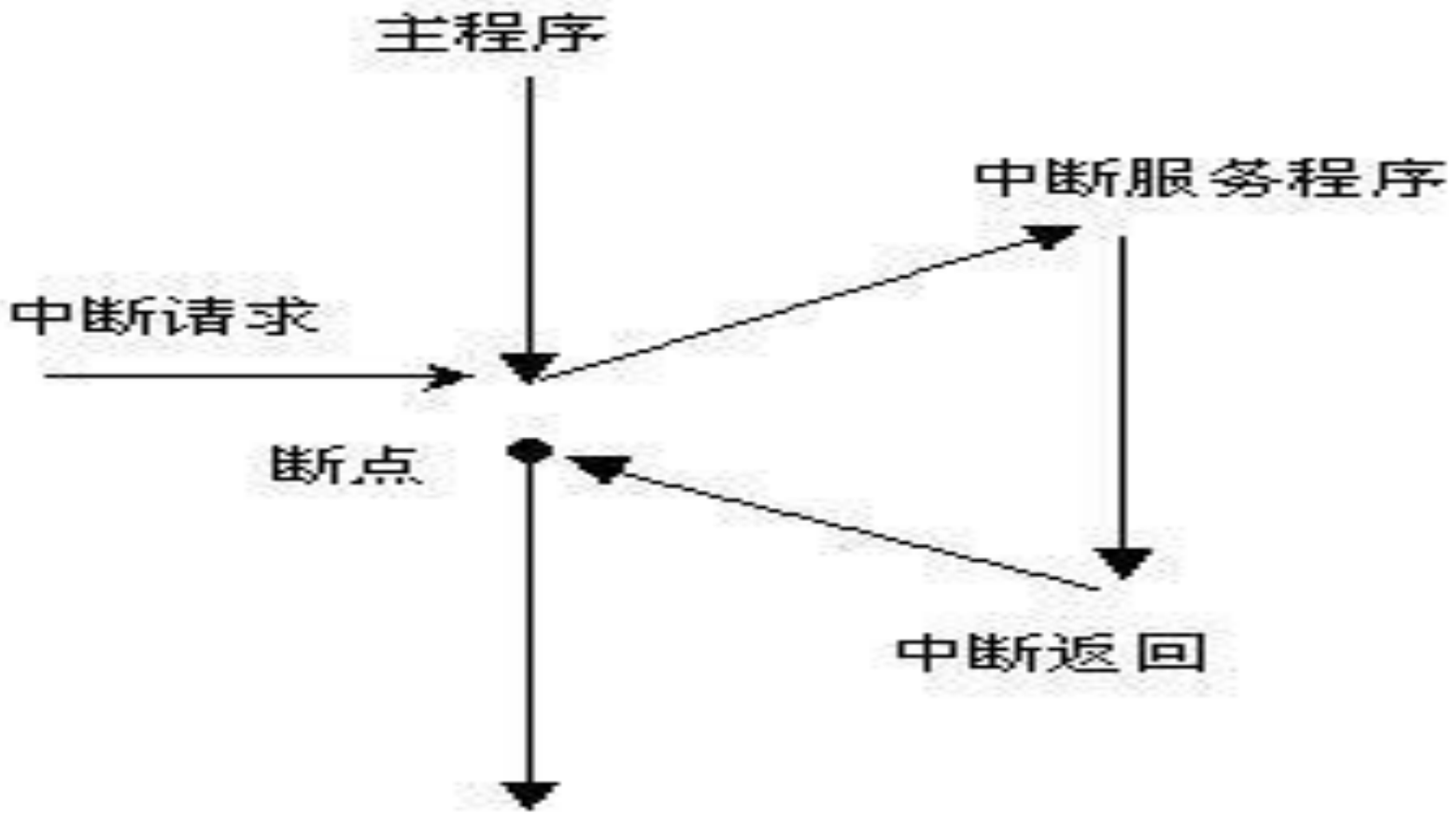
3）中断的作用

- （1）解决主机与外设的速度匹配问题，实现CPU与I/O设备并行工作。



- (2) 及时处理异常情况，提高机器的可靠性。
- (3) 便于实现人—机联系
- (4) 便于实现多台设备并行工作
- (5) 便于实现实时控制

4）中断处理的过程



中断过程中程序的执行情况

(1) 中断请求

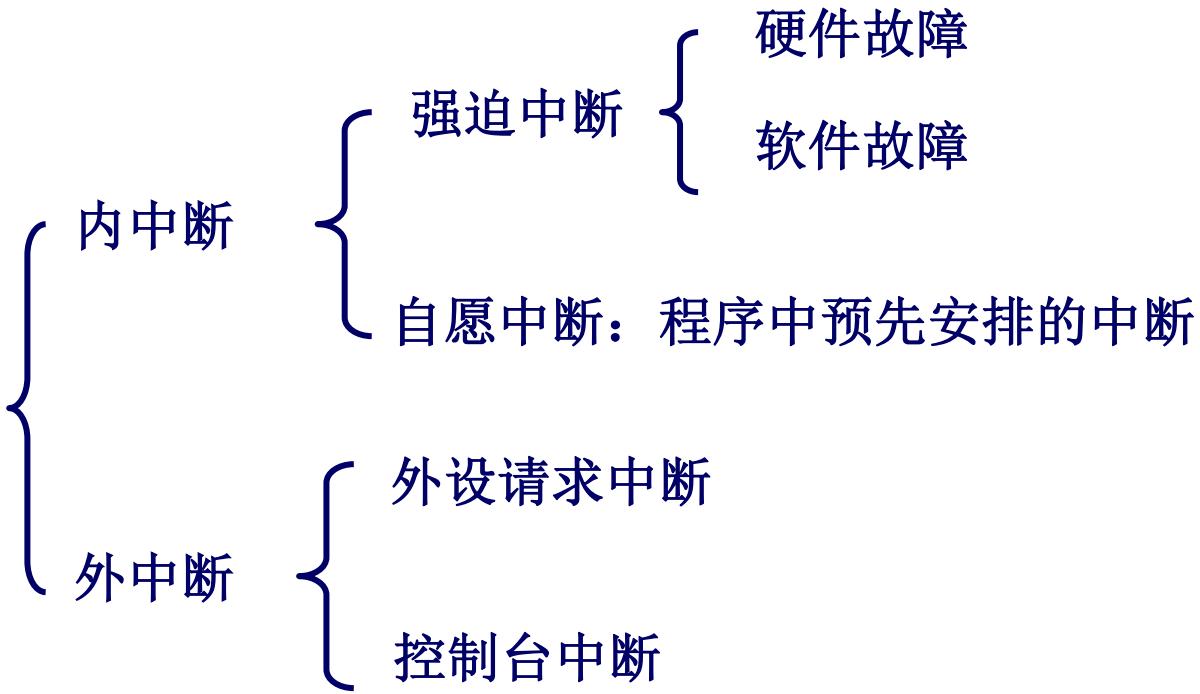
- 中断源以硬件信号形式通过中断控制线路向CPU提出中断请求。
- 中断源：引起中断的事件或原因。
- 中断源可以是外部的硬件设备，如键盘、打印机等输入/输出设备和各种控制设备；也可以是软件指令，如中断指令；还可能是由各种故障和出错引起的中断，如计算溢出等。

- (2) **中断判优及响应**
- 根据中断优先权进行判断，择优予以响应。
- (3) **保护现场**
- 保护主程序的运行现状，如PC值、PSW、寄存器和内存中的重要数据。
- (4) **中断服务**
- 按中断源的工作要求，进行各类特定的数据传送或控制处理。
- 中断服务程序：完成中断服务的程序

- **(5) 恢复现场**
- 为了正确返回原程序，需要进行恢复现场的工作，即将前面保存的寄存器的内容送回原寄存器。
- **(6) 中断返回**
- 返回被中断的程序，继续执行。

5）中断的分类

- (1) 按中断来源分



- (2) 按中断服务程序入口的获取方式分

向量中断：由中断系统硬件，直接向主机提供被响应中断的中断向量地址。

非向量中断：通过软件查询方式识别中断源，转入相应的中断服务程序入口地址。

- (3) 按是否可屏蔽分

可屏蔽中断：CPU可以禁止响应的外部中断。

不可屏蔽中断：CPU必须响应的外部中断。

6）中断系统的设计要求

- (1) 高级中断应能中断低级中断的处理，即要允许中断嵌套。
- (2) 能够方便、灵活地设置中断优先级。
- (3)保证中断请求信号的建立及保持的准确性，保证中断在未被响应时，中断请求信号不能随便被丢失。
- (4)保证各类中断都能及时得到响应，不应出现某些中断由于某种原因长时间得不到响应的情况。

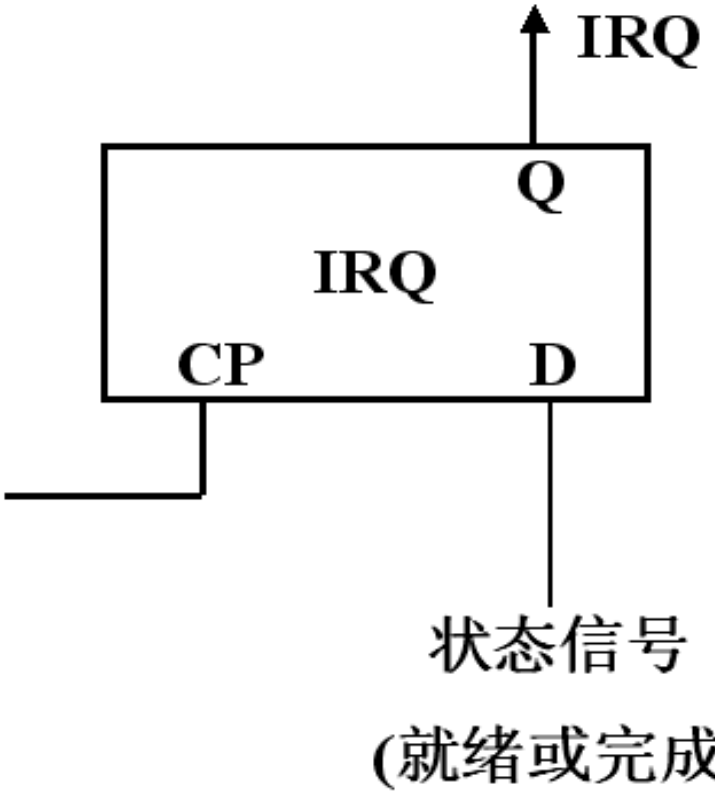
2. 中断请求信号的建立与传送

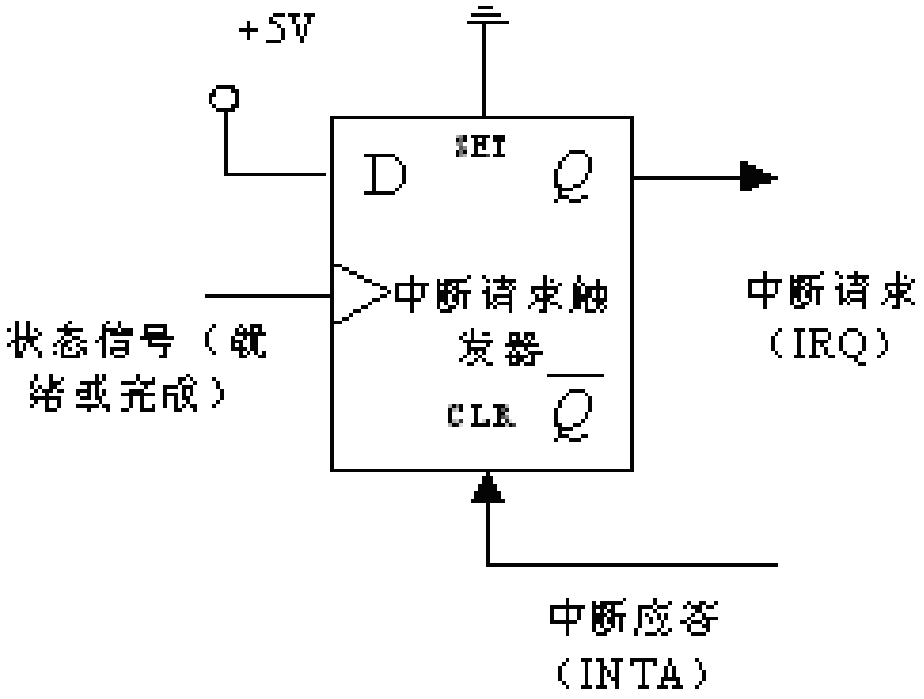
1）中断请求信号的建立与中断屏蔽

- 中断请求信号的建立，基于中断源有请求中断的需要。
- 例如，当外设已“准备就绪”或“完成一次操作”，可以用这类状态信号作为中断请求信号，使中断请求触发器的状态置“1”，表明已有中断请求。

外设发出请求中断信号的条件

- ① 外设准备就绪（Ready＝1）
- ② 外设的中断请求没有被屏蔽



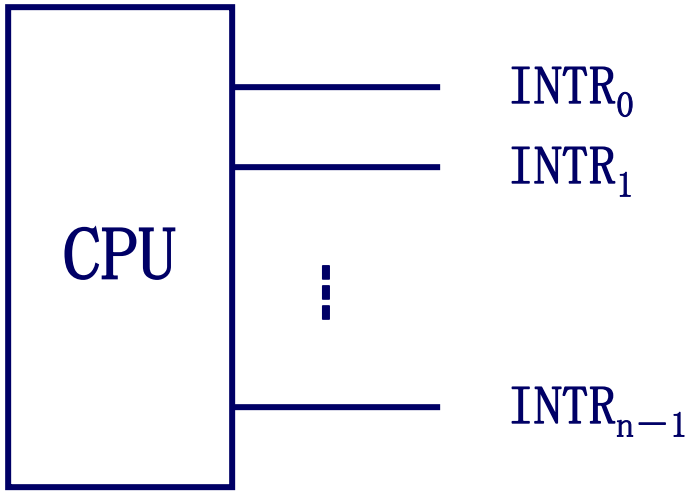


- 中断屏蔽：中断源的中断请求不能向CPU发出。
- 中断请求信号是否能够传送给CPU，要看当时占有CPU进程的程序的优先级。
- 如程序的优先级高于或等于当前中断请求的优先级，则不应将中断请求信号传给CPU，即需进行中断屏蔽。
- 如占有CPU进程的优先级低于请求中断的优先级，则不应屏蔽这个中断，而使CPU能够响应这个中断。

2) 中断请求信号的传送

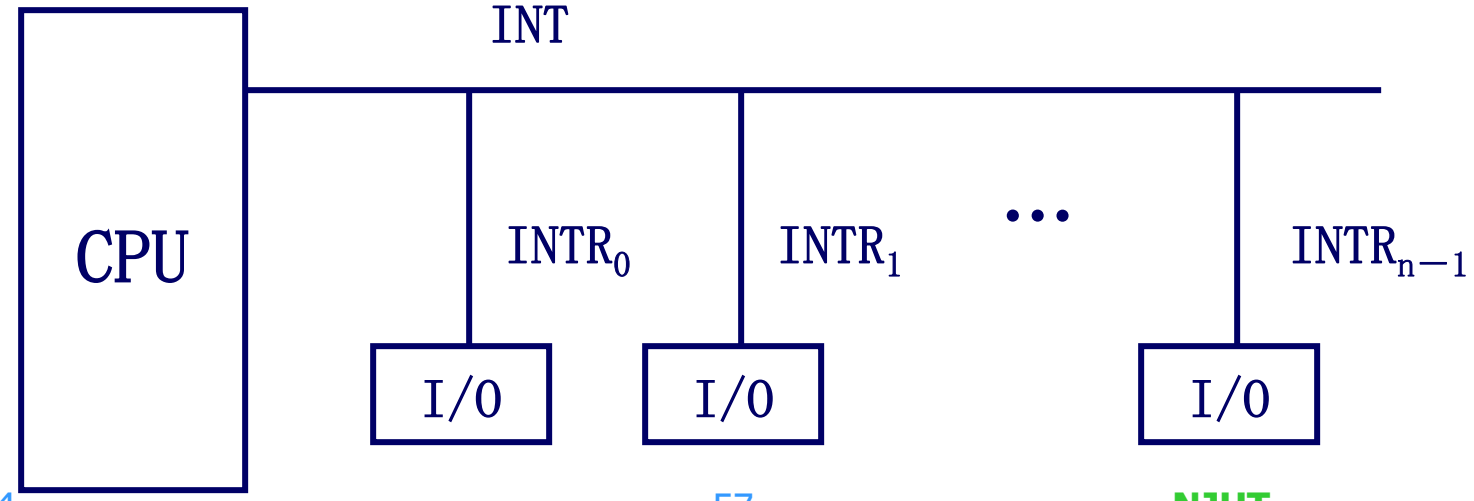
- 一台计算机系统中有多个中断源，可能产生多个中断请求信号，因此需要解决多个中断请求信号如何传送给CPU的问题。
- **(1) 独立请求线方式（多线单级结构）**
- 各中断源单独设置自己的中断请求线，多根请求线直接送往CPU。当CPU接到中断请求信号后，立即知道请求源是谁，并予以相应的处理。
- 这种方法有利于实现向量中断，因为可以通过编码电路形成向量地址。但因为CPU所能连接的中断请求线数目有限，所以中断源数目难以扩充。

独立请求线方式



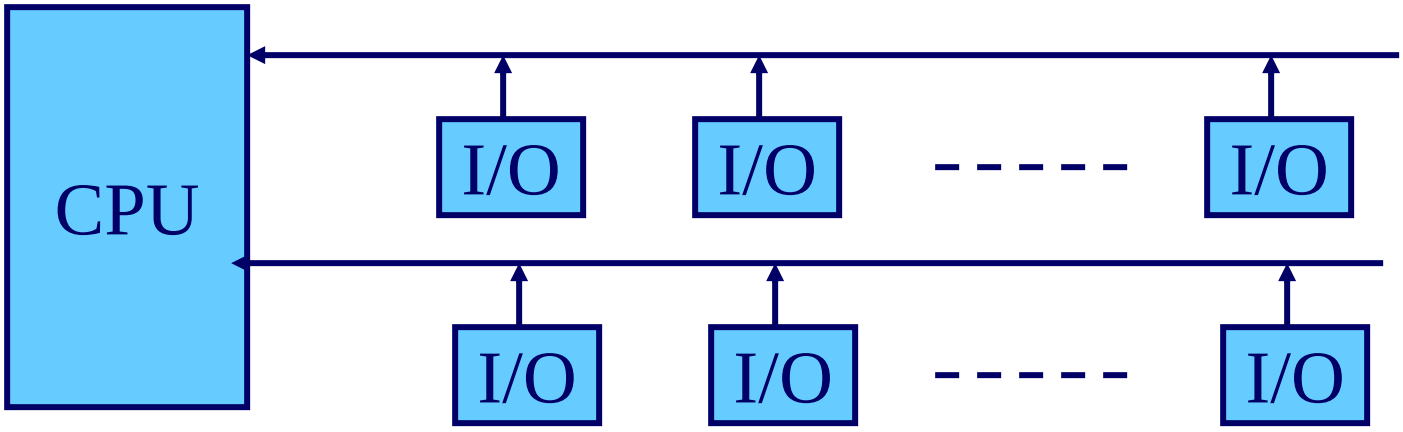
(2) 公共请求线方式（单线多级结构）

- 各中断源的请求信号通过三态门汇集到一根公共请求线，CPU只需接收一根中断请求线的请求信号。
- 这种方法节省引脚，但CPU响应中断后，还需要通过一定逻辑来识别是哪个中断源发出的中断请求，所以响应速度慢一些。



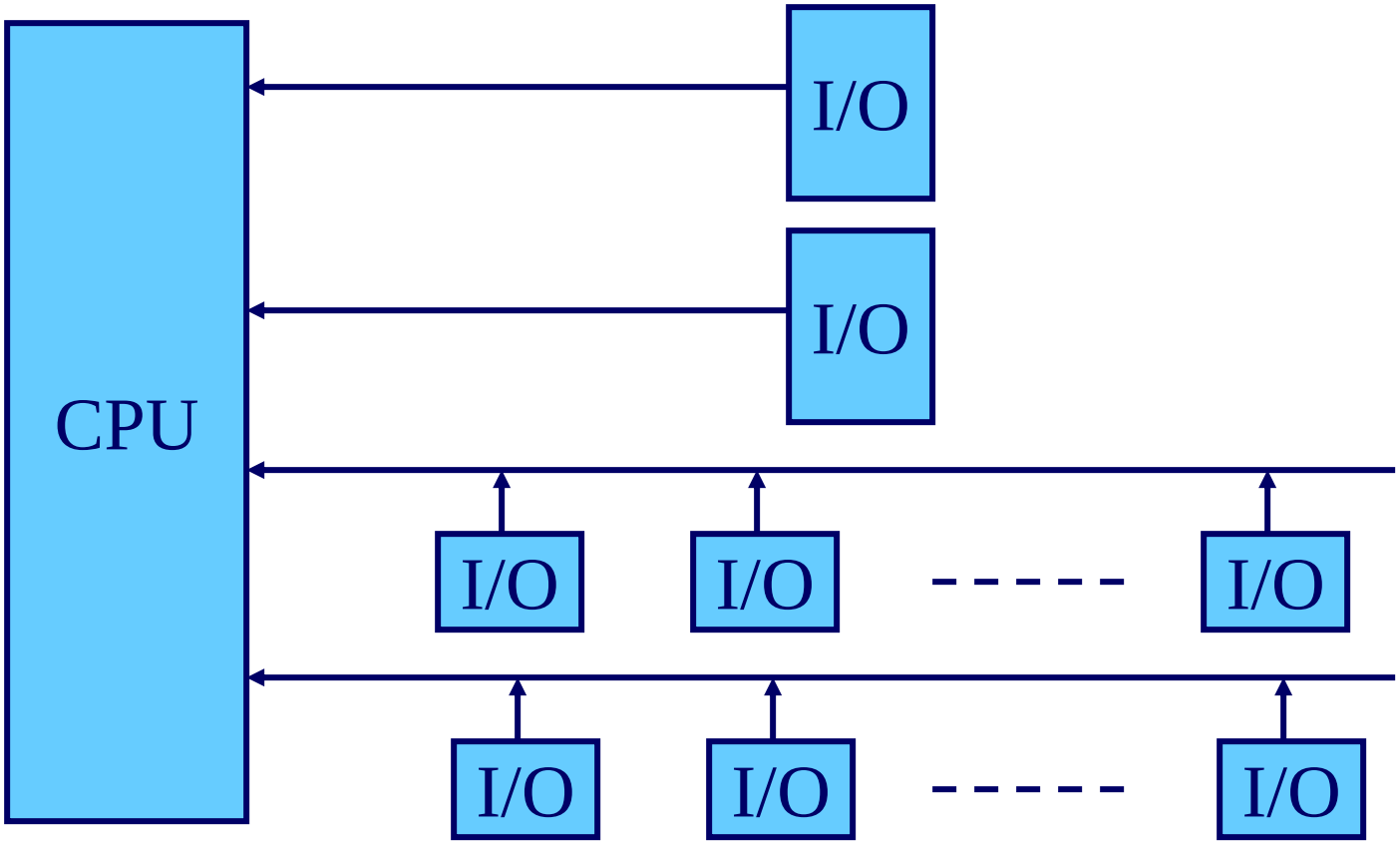
（3）采用二维结构

- ①CPU设置若干根中断请求输入线，体现不同的优先级；（主优先级）
- ②每根线挂接若干个I/O设备，以中断方式向CPU发请求。（此时，该线为公共请求线，又存在优先级问题，称为次优先级）



（4）兼有公共请求线与独立请求线

- 有些微机中采用此模式



3. 中断排队与判优

1）有关概念

- 中断排队：中断系统设计人员对中断请求的响应次序作出安排。
- 中断判优：系统运行过程中，当有多个中断源同时请求中断时，根据中断排队事先规定的次序判断中断请求的响应优先次序。
- 中断优先级（中断优先权）：根据中断源中断任务的紧迫程度，给各中断请求安排的响应次序。

中断排队的原则

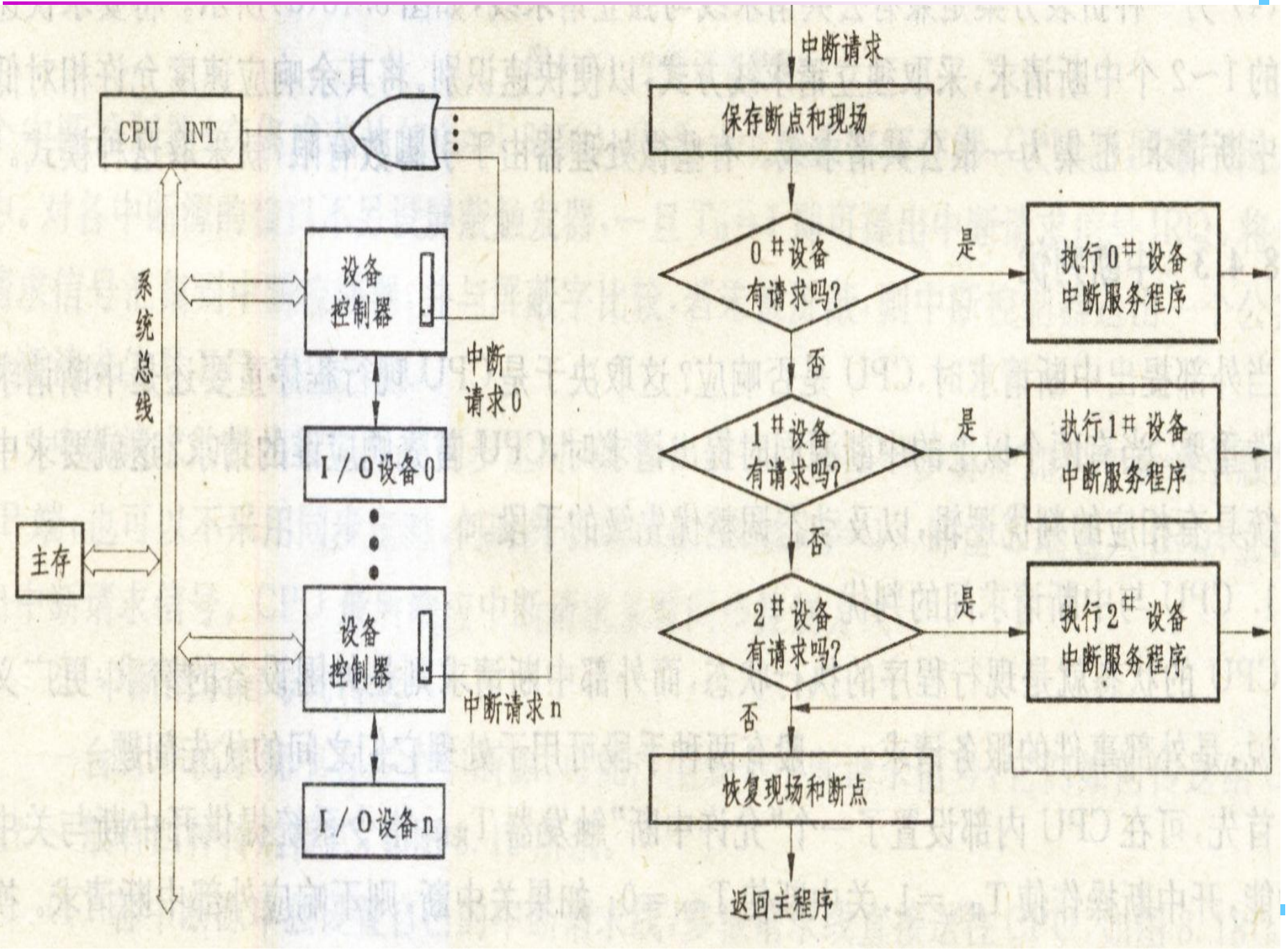
- ① 内部中断优先于外部中断
- ② 故障中断优先于设备请求中断
- ③ 非屏蔽中断优先于可屏蔽中断
- ④ 输入操作的中断请求优先于输出操作的中断请求
- ⑤ 数据有效时间短的中断优先于数据有效时间长的中断
- 具体设计时，中断优先级可以是固定的，也可以是动态变化的。可以采用硬件或软件进行中断排队和判优。

2) CPU与中断请求间的判优

- CPU是否响应中断请求，要看当时占有CPU进程的程序的优先级。
- 如程序的优先级高于或等于当前中断请求的优先级，则CPU可以不响应这个中断，或说CPU不允许被中断、中断被禁止。
- 如占有CPU进程的优先级低于请求中断的优先级，则不应禁止这个中断，而使CPU能够响应这个中断，或说CPU允许中断。
- 中断禁止：在一定条件下，CPU不允许响应中断。
- 中断允许：在一定条件下，CPU允许响应中断。

3) 中断请求之间的排队与判优

- **(1) 软件查询**
- 响应中断请求后，先转入中断查询程序，按优先顺序依次询问各中断源是否提出请求。如果是，则转入相应的服务处理程序。如果没有，则继续往下查询。查询的顺序体现了优先级别的高低，改变查询顺序也就修改了优先级。


CPU INT
系统总线
主存
设备控制器
中断请求0
I/O设备0
设备控制器
中断请求n
I/O设备n
中断请求
保存断点和现场
0#设备有请求吗?
是
执行0#设备中断服务程序
否
1#设备有请求吗?
是
执行1#设备中断服务程序
否
2#设备有请求吗?
是
执行2#设备中断服务程序
否
恢复现场和断点
返回主程序

- 软件查询方法适用于低速和中速设备，多用于公共请求线方式。
- 优点：中断条件标志的优先级可用程序任意改变，灵活性好。
- 缺点：设备多时，响应速度太慢。

(2) 并行优先排队逻辑

- 采用硬件并行优先排队逻辑对具有独立中断请求线的中断请求进行判优。
- 并行排优逻辑适于具有多请求线的系统，速度较快，硬件代价较高。

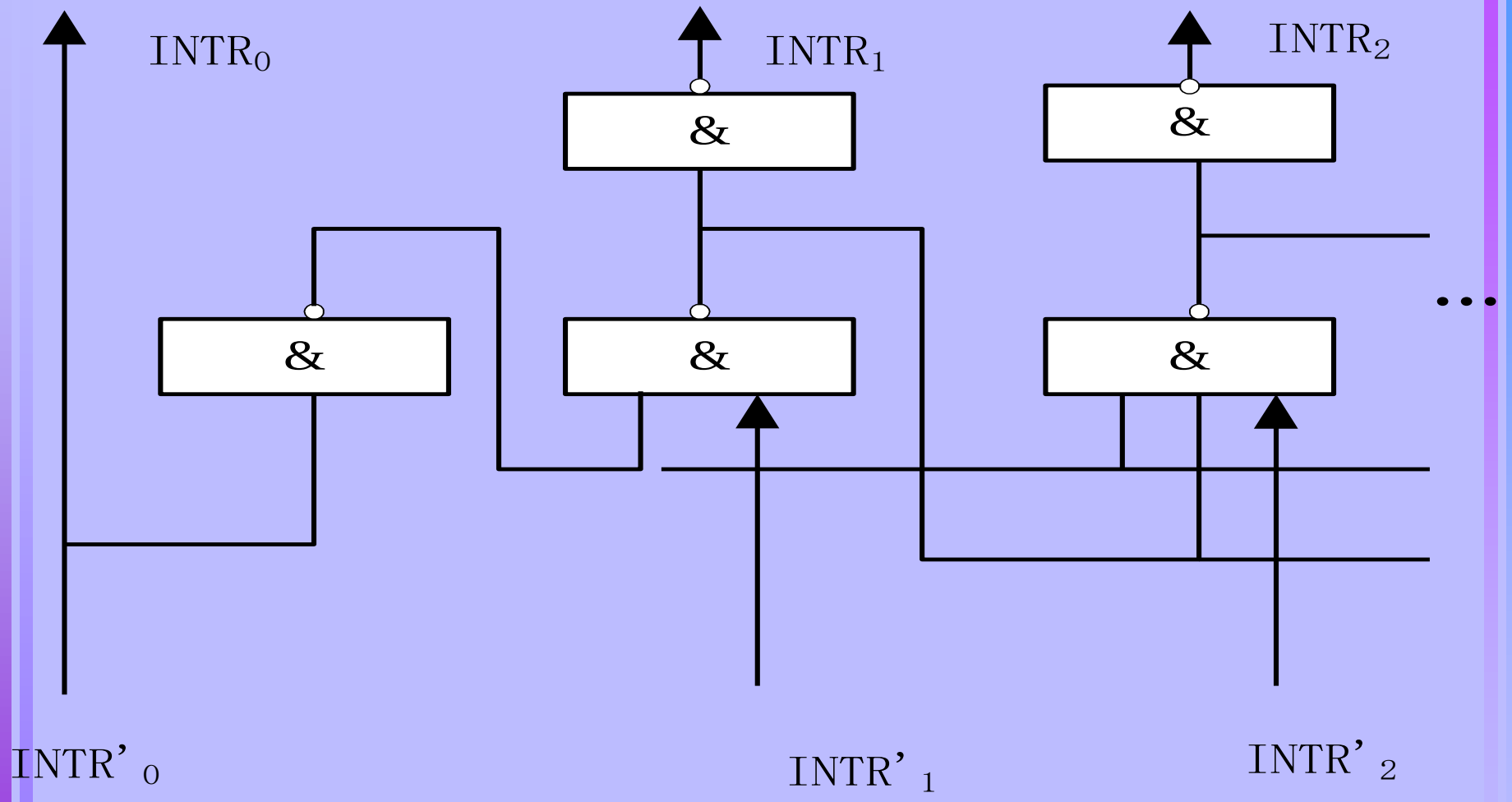
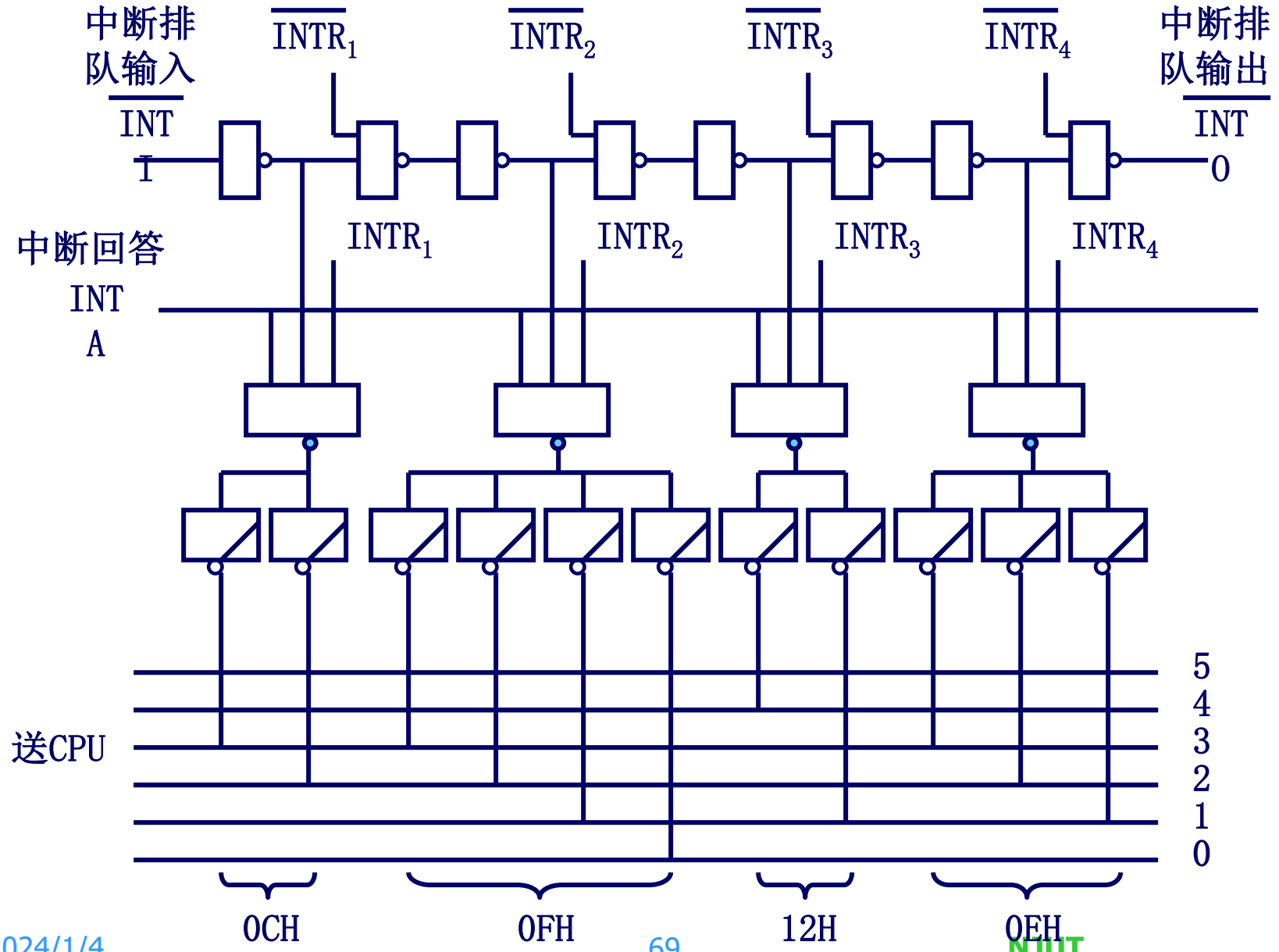


图9-12 具有独立请求线的并行优先级排队电路

(3) 链式优先排队逻辑

- 采用硬件优先链电路判断中断优先级，判优结果可用不同的设备码或用中断源类型码来表示。
- 用于采用公共请求线的系统。
- 串行方式

  （ 图9-14）


中断排
队输入
INT
I
INTR1
INTR2
INTR3
INTR4
中断排
队输出
INT
O
中断回答
INT
A
INTR1
INTR2
INTR3
INTR4
送CPU
5
4
3
2
1
0
0CH
0FH
12H
0EH

（4）二维结构的优先排队

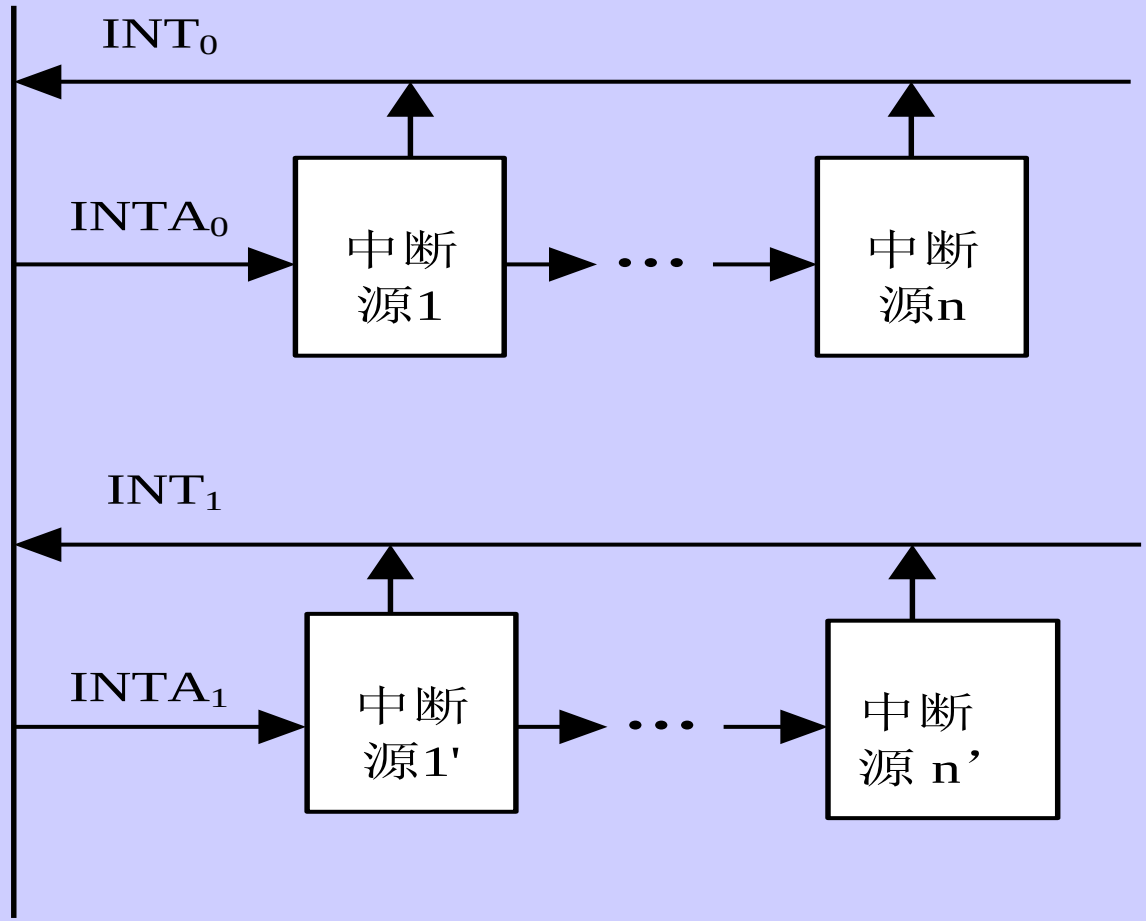
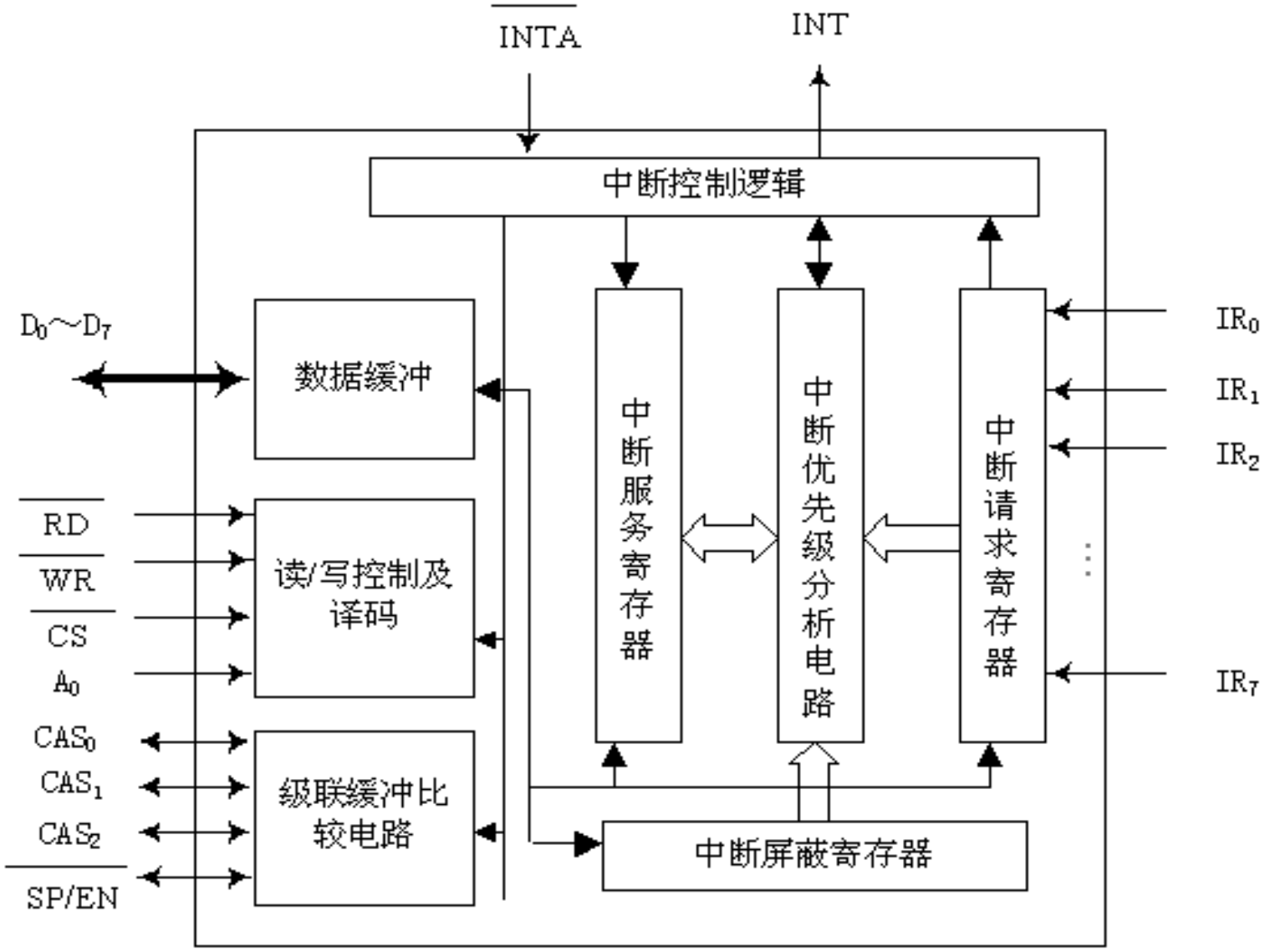


图9-15 采用二维结构的优先链排队逻辑

(5)采用中断控制器集成芯片的优先逻辑

- 在微型计算机中，广泛使用中断控制器集成芯片，如Intel 8259A。中断控制器将中断请求信号的寄存、汇集、屏蔽、排优、编码等逻辑，集成在一块芯片之中。(见下图)

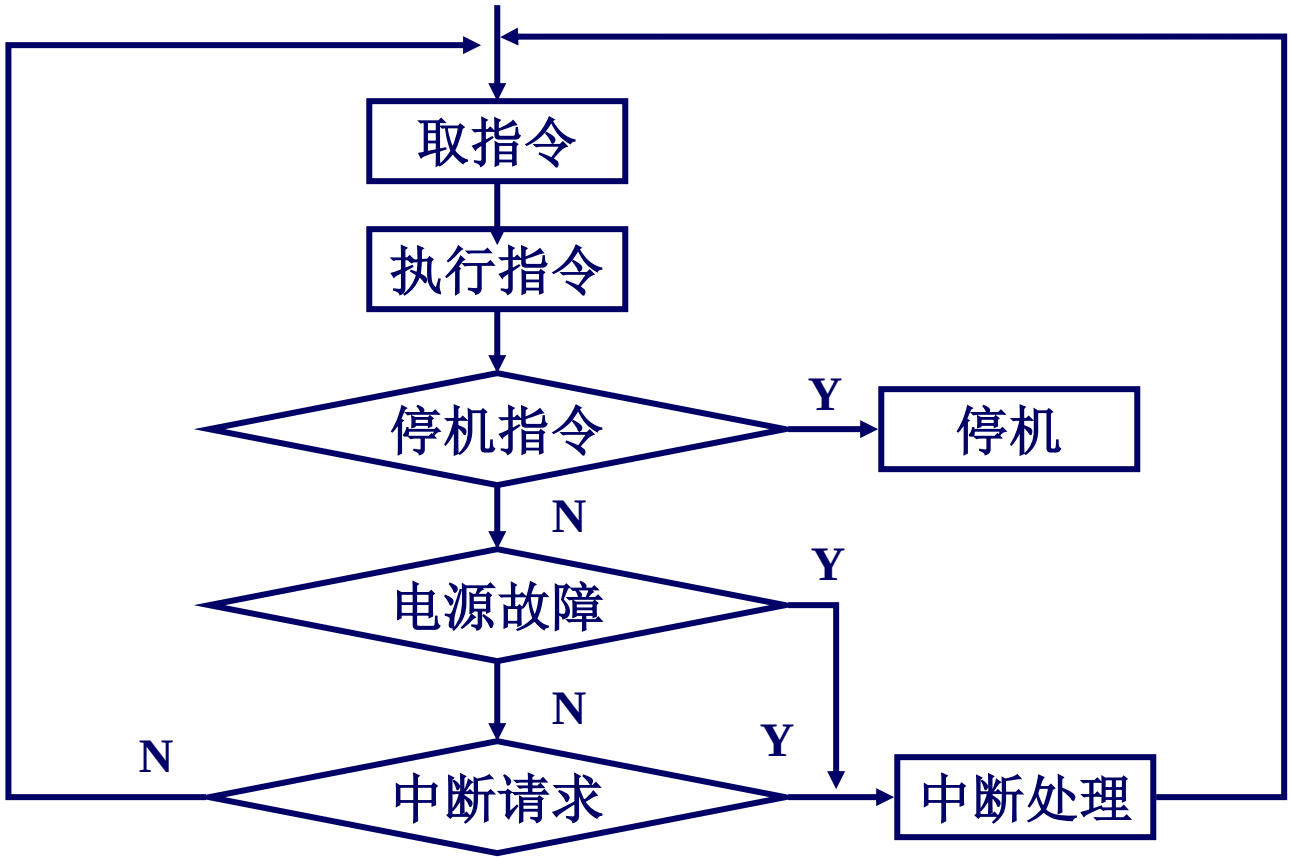
- Intel 8259A 可以多个级连，用于更多的中断请求信号。


INTA
INT
中断控制逻辑
D0～D7
数据缓冲
中断服务寄存器
中断优先级分析电路
中断请求寄存器
IR0
IR1
IR2
IR7
RD
WR
CS
A0
读/写控制及译码
CAS0
CAS1
CAS2
SP/EN
级联缓冲比较电路
中断屏蔽寄存器

4. 中断响应

- CPU接到中断请求信号后，若满足响应中断的条件，即暂停现行程序的执行，而转入中断处理，将这一过程称为中断响应。
- 1）CPU响应中断应具备的条件：
- (1) 有中断源请求中断。
- (2) CPU允许响应中断，即处于开中断状态。
- (3) 现行指令不是停机指令
- (4) 一条指令执行结束。
- （另，没有被屏蔽，优先级高等）

中断响应流程



2）中断响应过程中应完成的操作

- (1)关中断
- 以便在保存现场过程中不允许响应新的中断请求，确保现场保存的正确性；
- (2)保存断点地址-PC和程序状态字PSW；
- (3)转入中断服务程序入口，以便执行相应的中断服务程序，完成中断处理任务。
- 中断响应周期的操作流程中的操作不是在程序中安排的，而是直接由硬件完成的。通常把这种操作称为执行中断隐指令。（中断周期的任务）

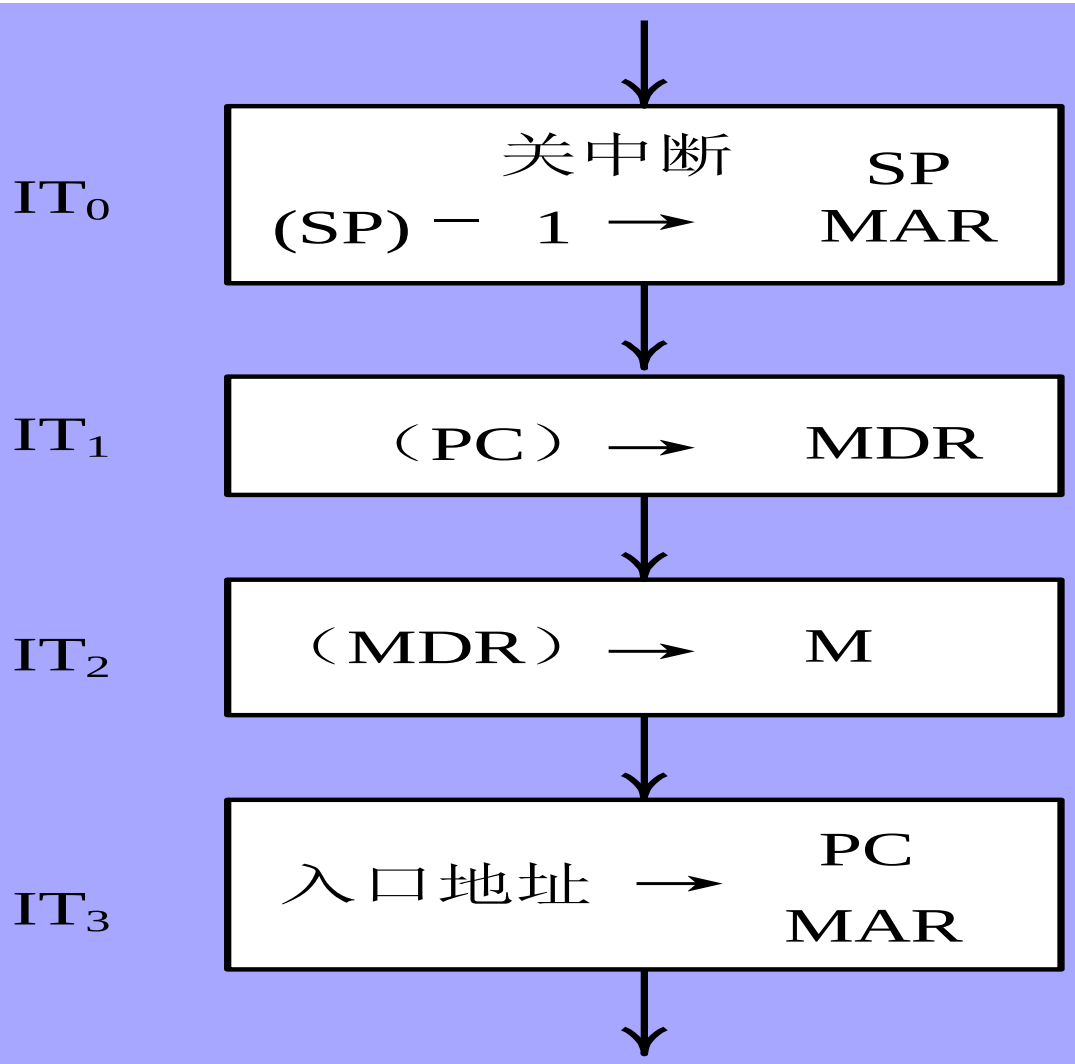


图9-17 中断响应隐指令操作流程

硬件完成

5. 中断服务程序入口地址的获取方式

1）软件查询方法

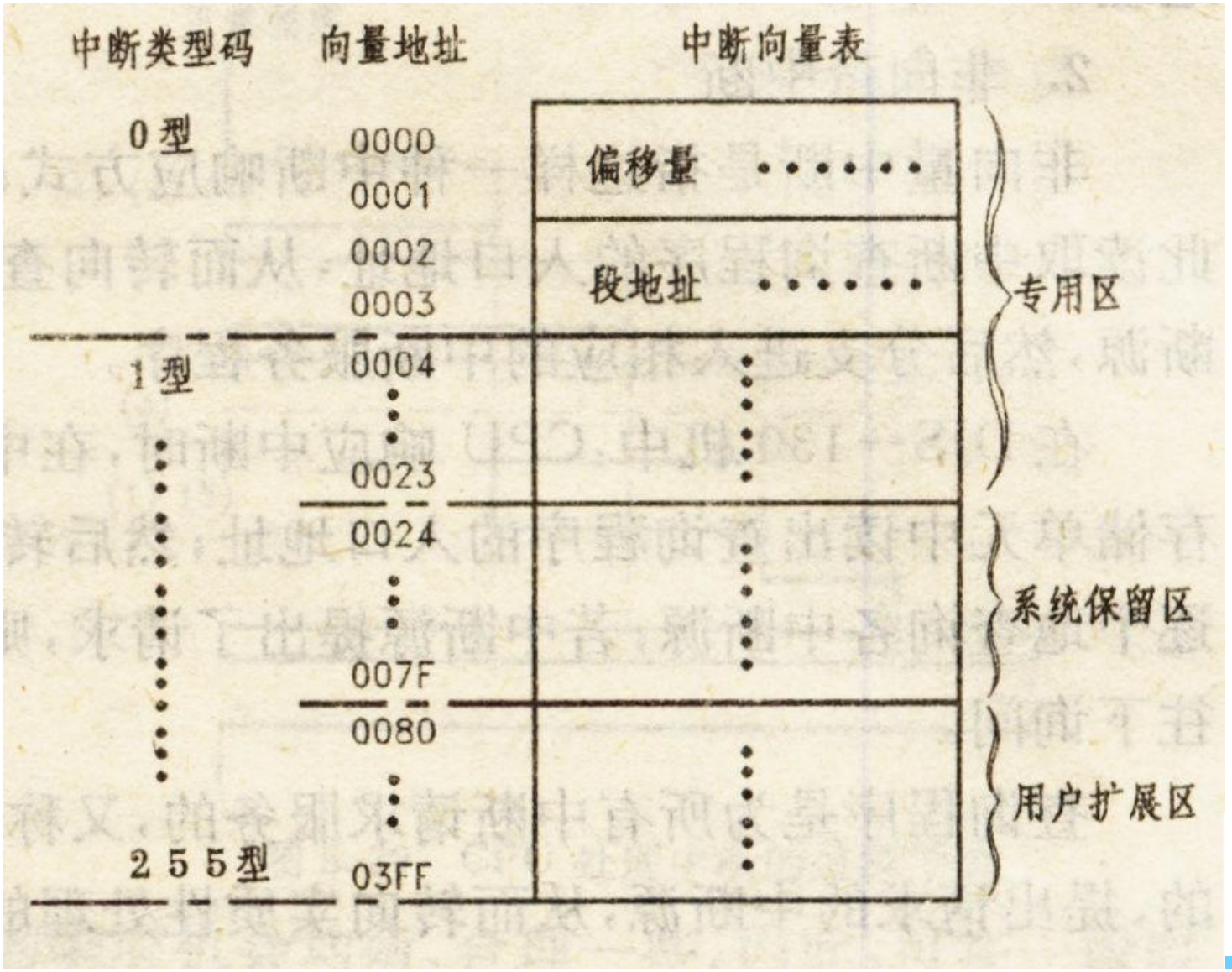
CPU响应中断请求时，产生1固定地址，先转入中断查询程序，按优先顺序依次识别中断源，并转入相应的中断服务程序入口

也称为非向量中断

2）向量中断

- 中断向量：中断源对应的中断服务程序的入口地址。
- 中断向量表：存放中断向量的表。
- 所有的中断服务程序入口地址(组织成一个一维的表格，存放在一段连续的存储区中。
- 中断向量地址：访问中断向量表的地址码，即读取中断向量所需的地址，也称为中断指针。

IBM PC机的中断向量表



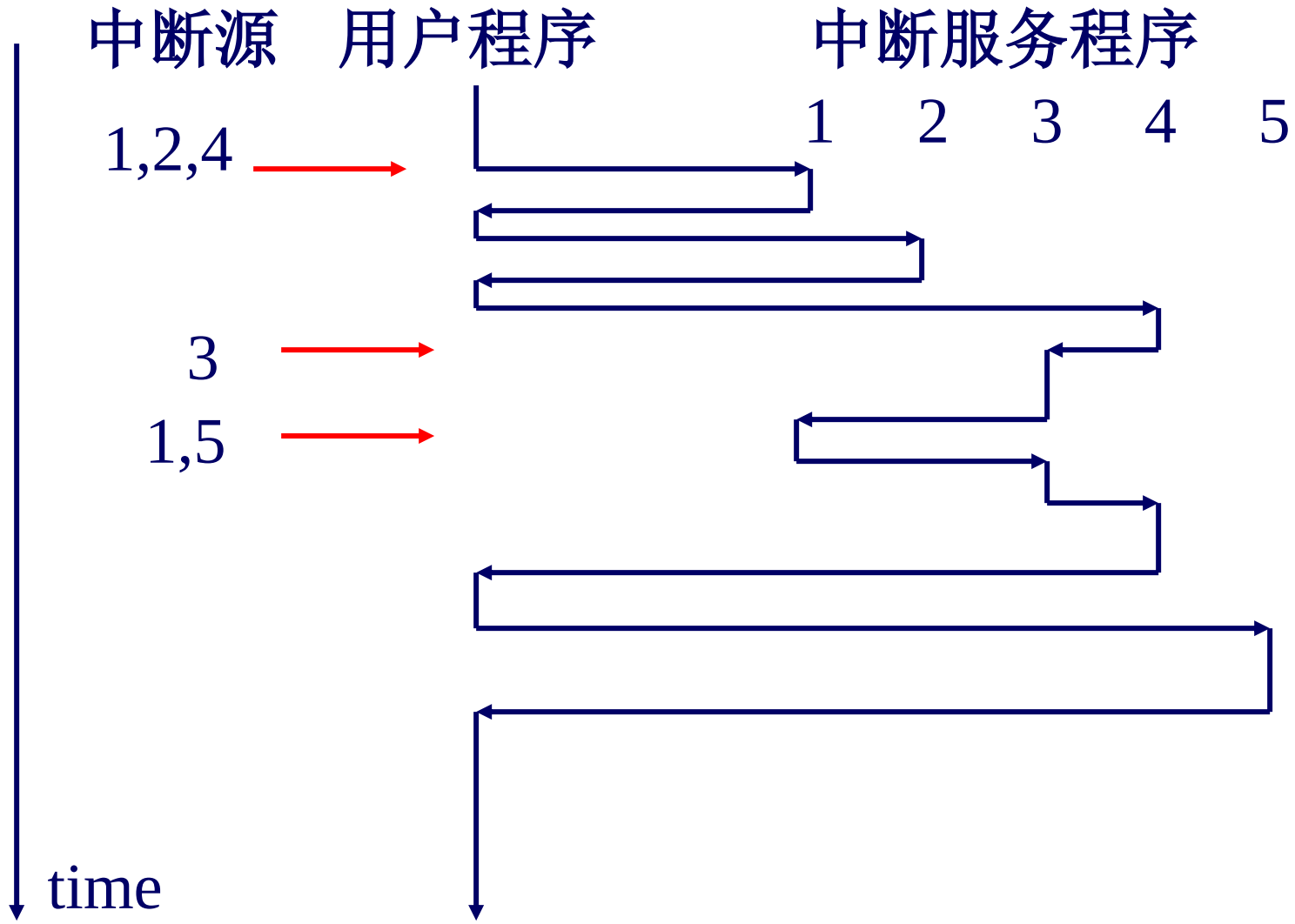
- 向量中断：将各个中断服务程序的入口地址组织成中断向量表；响应中断时，由硬件直接产生对应于中断源的向量地址；由此转向中断服务程序。
- 向量中断的响应工作一般在中断周期中由硬件直接实现。
- 向量中断的特点是能够根据中断请求信号快速、直接地转向对应的中断服务程序。因此现代计算机基本上都具有向量中断功能。

6. 多重中断与中断屏蔽

- 多重中断→中断嵌套
- 一. 多重中断原则
- 1) 若新产生的中断源高于现中断的优先级,则予以响应;
- 2) 若新产生的中断源同于或低于现中断的优先级,则不予以响应. 等现中断处理结束后,再处理新中断.
- 中断嵌套需要堆栈支持

举例

- 某计算机的中断系统有五级中断,优先次序为
- **1→2→3→4→5**
- 若**CPU**在执行正常程序时,有下列事件发生:
- **(1)** 中断**1,2,4** 提出请求;
- **(2)** 在处理中断**4**过程中,又有中断**3**提出请求;
- **(3)** 在处理中断**3**时,又出现**1,5**中断请求.
- 请画出**CPU**对所有事件的处理过程图.


中断源
用户程序
中断服务程序
1 2 3 4 5
1,2,4
3
1,5
time

二. 中断屏蔽技术

- 1) 作用: 封锁部分中断;
- 中断升级(高优先的被屏蔽,低优先
- 此时“最优”)
- 2) 例子
- 在上例中,我们可以得到如下所示的中断屏蔽码. (1→屏蔽; 0→开放)

- 注意:本级不允许相互干扰

级别	屏　蔽　码				
	1	2	3	4	5
1	1	1	1	1	1
2	0	1	1	1	1
3	0	0	1	1	1
4	0	0	0	1	1
5	0	0	0	0	1

- 若改成 **1→4→3→2→5**次序, 则屏蔽码如下:

级别	屏蔽码				
	1	2	3	4	5
1	1	1	1	1	1
2	0	1	0	0	1
3	0	1	1	0	1
4	0	1	1	1	1
5	0	0	0	0	1

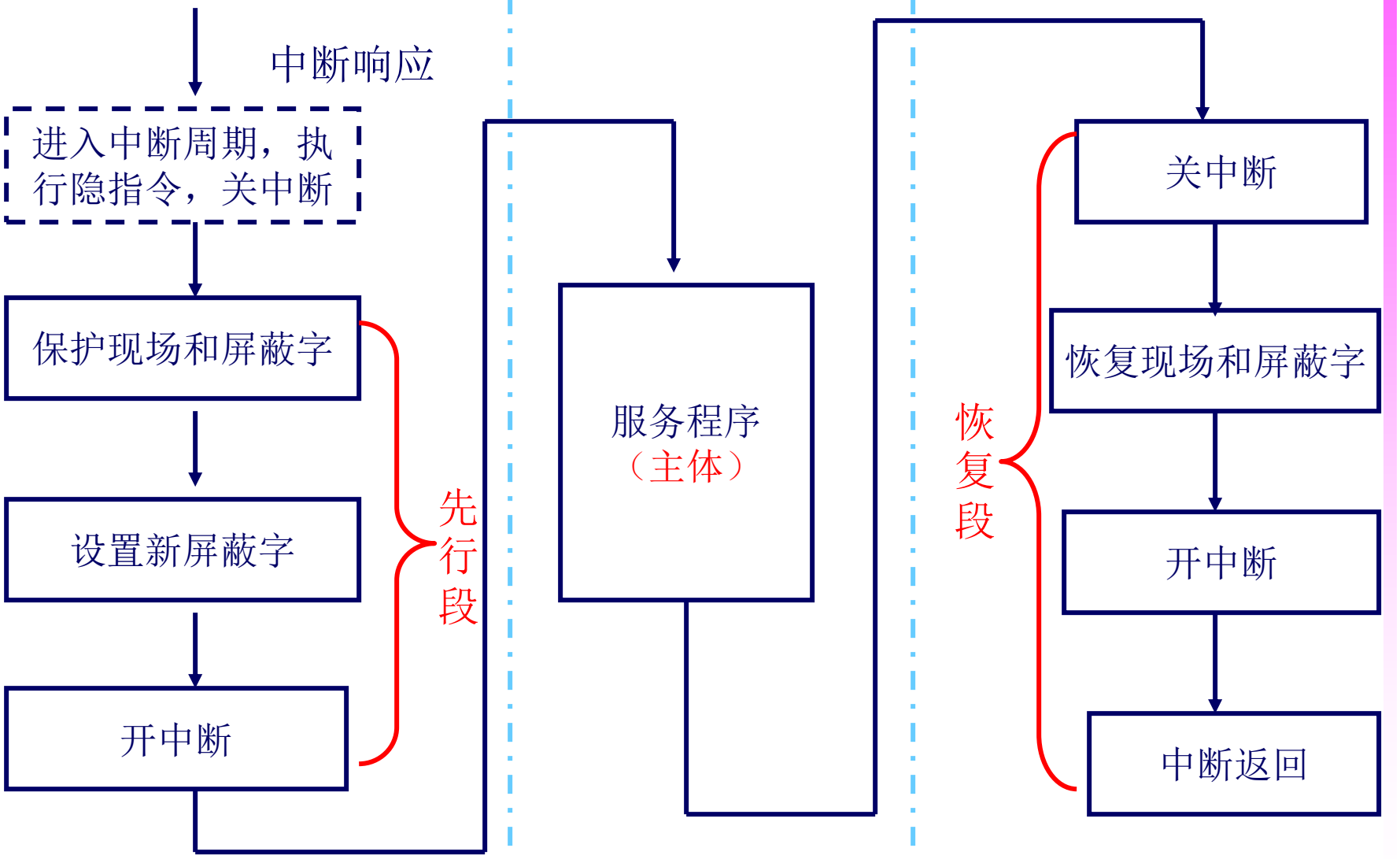
7. 中断服务(中断处理)

- 取得中断服务程序的入口地址后，CPU开始执行中断服务程序，完成规定的中断处理任务。
- 中断服务程序一般由起始、主体、结尾三部分组成。
- 1）起始部分（先行段）
- ① 判明中断原因，识别中断源，对于不同中断源转入不同的服务程序。
- 对于向量中断，直接由硬件判明中断源并给出中断向量地址，转入相应中断服务程序。
- 对于非向量中断，需通过执行一段程序判明中断源，转入相应中断服务程序。

- ② 设置屏蔽字，封锁同级与低级中断。
- ③ 保存中断现场。除了PSW外，还要保存一些在执行中断服务程序过程中可能被改变的寄存器的内容。
- ④ 开中断，以便在本次中断处理过程中，允许响应更高级的中断请求。
- 这是因为在中断响应时，为避免影响保护现场，进行了关中断操作。
- 2）主体部分
- 执行具体的为中断源服务的程序

- 3）结尾部分（恢复段）
- ①关中断，以便在恢复现场过程中不允许响应新的中断。
- ②恢复中断现场，将原来保存的寄存器内容送回原寄存器。
- ③清中断请求信号，表示本次中断处理结束。
- ④清屏蔽字，开放同级与低级中断。
- ⑤开中断，以便响应新的中断请求。
- ⑥恢复PSW、PC，返回被中断的程序。

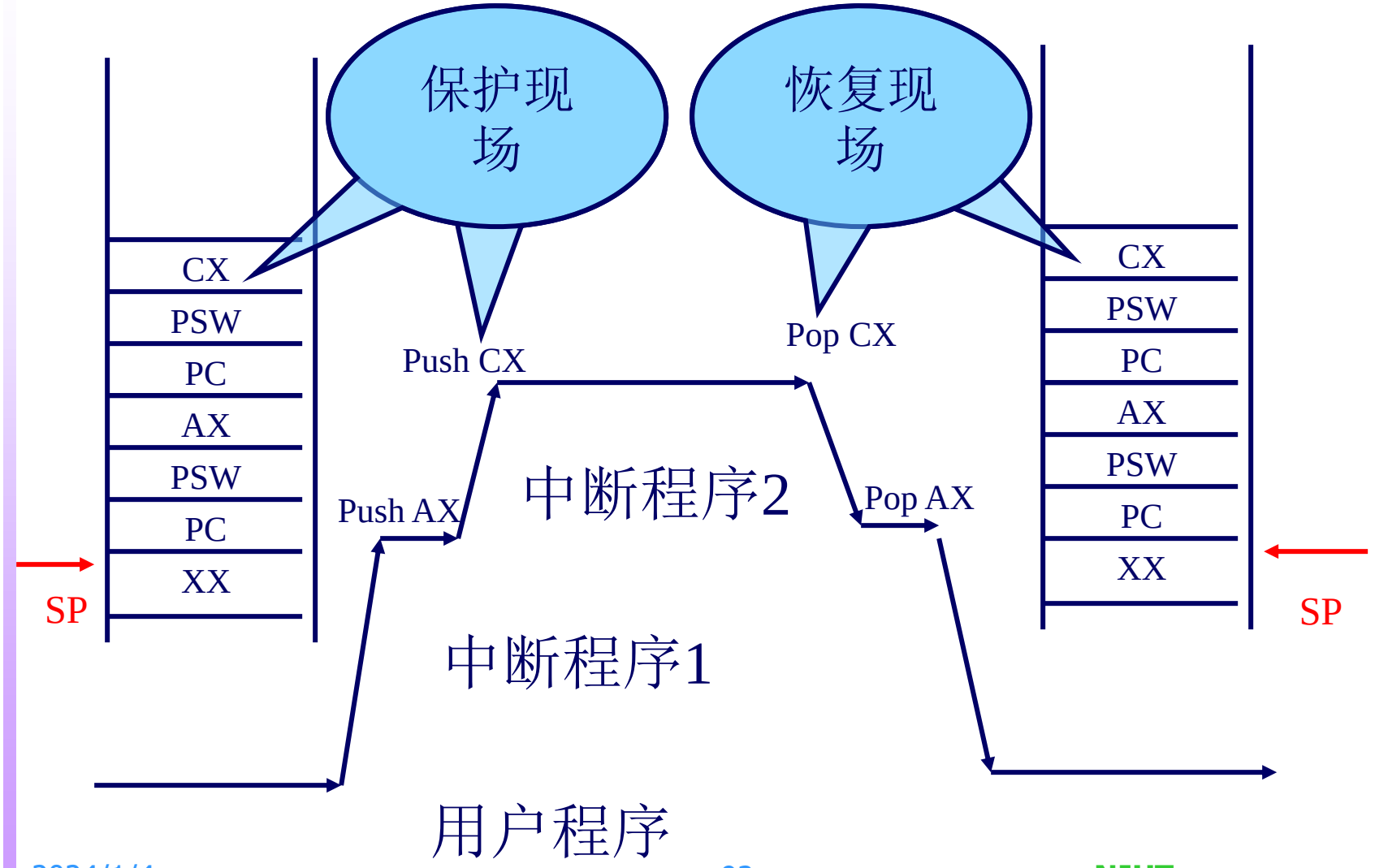
中断服务子程序


中断响应
进入中断周期，执行隐指令，关中断
保护现场和屏蔽字
设置新屏蔽字
开中断
先行段
服务程序（主体）
关中断
恢复现场和屏蔽字
开中断
中断返回
恢复段

中断处理的全过程（6步）

- ① 中断请求；
- ② 择优响应；
- ③ 保护现场；
- ④ 中断服务；
- ⑤ 恢复现场；
- ⑥ 中断返回；
- 在中断处理的过程中，有些是由硬件完成的，有些是由软件完成的，因此中断是一种软、硬件结合的技术手段。不同的机器，软、硬件功能分配的比例有所不同。

中断堆栈的变化


保护现场
恢复现场
CX
PSW
PC
AX
PSW
PC
XX
SP
Push CX
Pop CX
Push AX
Pop AX
中断程序2
中断程序1
用户程序
CX
PSW
PC
AX
PSW
PC
XX
SP

9.4 直接存储器存取(DMA)

- 直接存储器存取方式(Direct Memory Access)简称DMA方式。
- DMA方式：以主存为中心，采用硬件手段在主存与I/O设备之间建立直接的数据传送通路，由DMA控制器（DMAC）取得总线控制权，控制主存与I/O设备之间的数据传送，在传送过程中不需要CPU的程序干预的数据传送控制方式。
- DMA方式主要用于高速外设按照连续地址直接访问存储器。

9.4.1 DMA方式的特点与应用场合

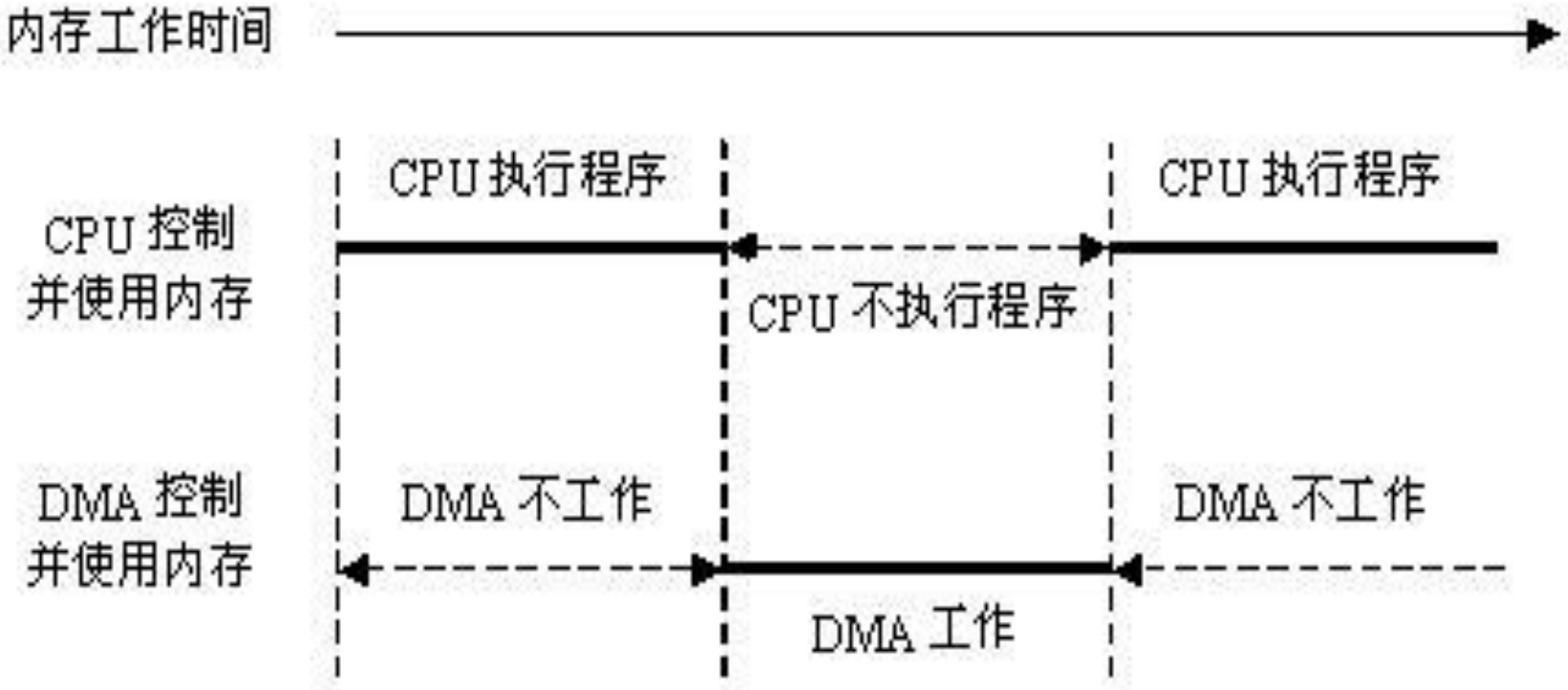
- 1. DMA方式的特点
- (1) 以响应随机请求的方式，实现主存与I/O设备间的快速数据传送。
- (2) 采用DMA方式控制数据传送时，仅需占用系统总线，不切换程序，不存在保存断点、保护现场、恢复现场、恢复断点等操作。因此DMA传送的插入不影响CPU的程序执行状态，除了访问主存的冲突外，CPU可以继续执行自己的程序，提高了CPU的利用率。
- (3) DMA方式只能处理简单的数据传送，难以识别与处理复杂的情况。

- 2. DMA方式的应用
- DMA方式一般应用于主存与高速I/O设备间的简单数据传送(高速I/O设备如磁盘、磁带、光盘等外存储器)，以及其它带有局部存储器的外围设备、通信设备等。如：
- (1) 磁盘与主存的成块数据传送
- (2) 通信设备的批量数据传送
- (3) 动态存储器的刷新

- DMA传送是直接依靠硬件实现的，可用于快速的数据直传。但DMA方式本身不能处理复杂事态。因此，在某些场合常综合应用DMA方式与程序中断方式，二者互为补充。
- 典型的例子是磁盘调用，磁盘读写采用DMA方式进行数据传送，而对寻道是否正确的判别处理、批量传送结束后的善后处理，则采用程序中断方式。

9.4.2 DMA的传送方式

- 1. CPU停机方式
- 用CPU停机方式实现DMA传送时，CPU停止工作，让出对总线的控制权，而由DMAC接管总线，进行数据传送。数据传送结束后，再将总线交还给CPU。
- CPU停机方式控制简单，比较容易实现，是最常用、最简单的一种DMA实现方式，大部分DMAC都采用这种方式。
- 但在采用这种方式进行的DMA传送期间，由于使CPU处于空闲等待状态，降低了CPU的利用率，并且可能会影响到某些实时性很强的操作，如中断响应。



(a) CPU 停机方式

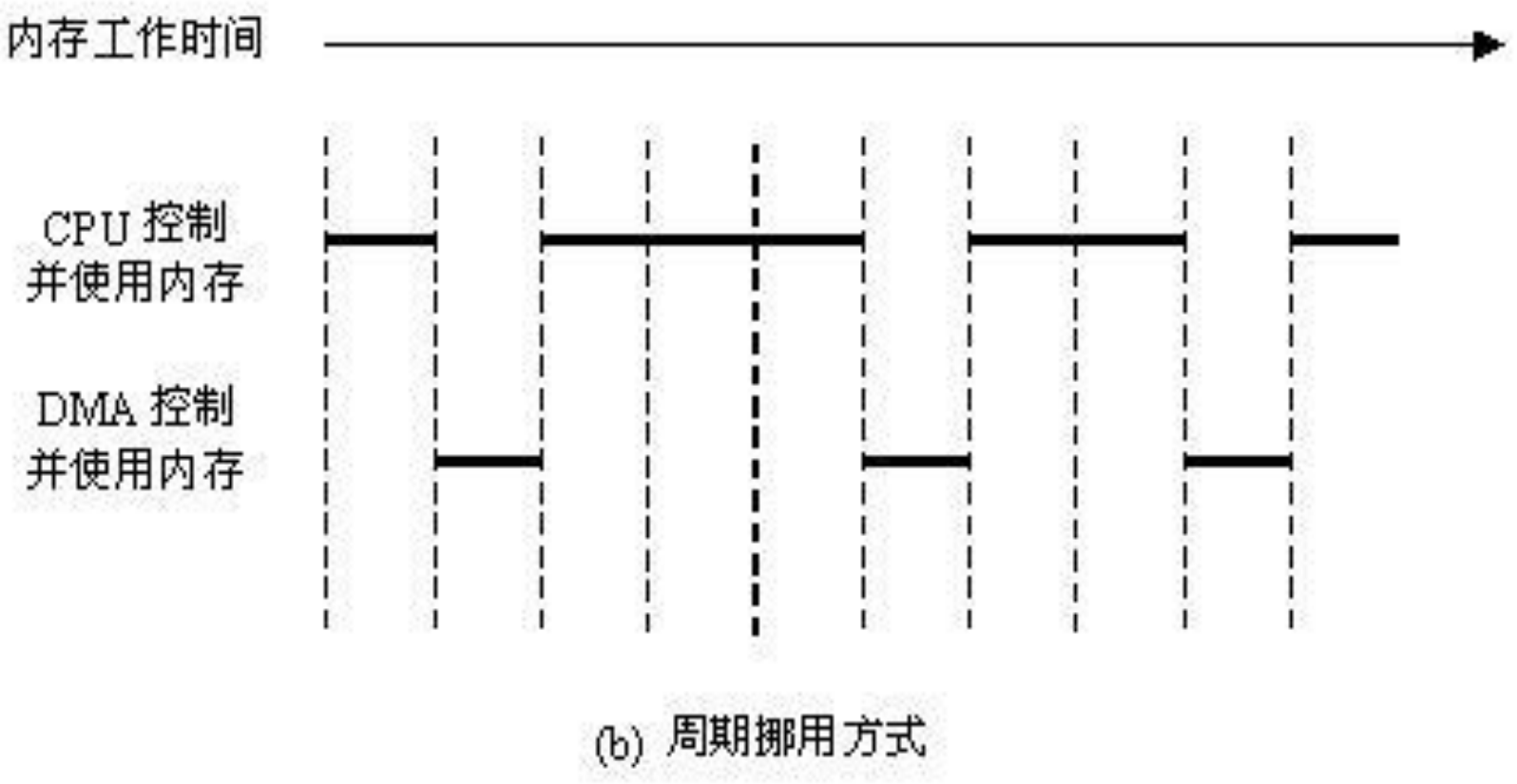
申请一次传送一批（快速设备成批传送）

2. 周期挪用(周期窃取)方式

- 周期挪用：每当外设发出DMA请求时，DMAC便挪用或窃取总线控制权一个或几个主存周期，而外设不发出DMA请求时，CPU仍继续访问主存。

- 外设要求DMA传送时，CPU正在访存，此时必须等存取周期结束后，CPU才能让出总线控制权。
- 外设要求访存时，CPU也要求访存，这就出现了访存冲突。此时要求外设访存优先于CPU访存。因为外设不立即访存就可能丢失数据，这时DMAC要窃取一、二个存取周期，使CPU延缓一、二个存取周期再访存。
- 与CPU暂停访存的方式相比，周期挪用方式既实现了I/O传送，又较好地发挥了主存与CPU的效率，是一种广泛采用的方法。

周期挪用

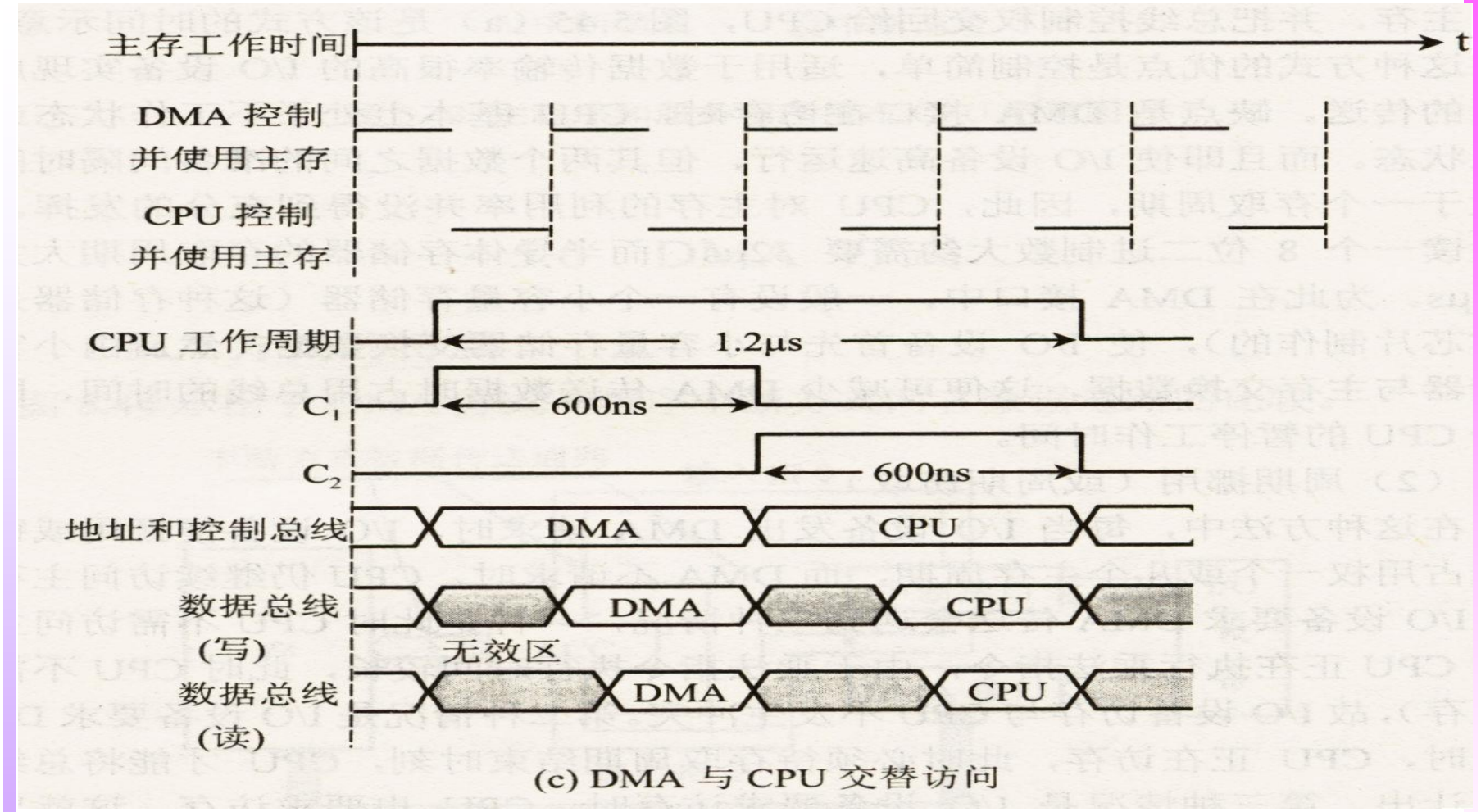


(b) 周期挪用方式

申请一次传送一字节（慢速设备传送）

3. DMA与CPU交替访问

- 将一个CPU周期分为两个分周期，与DMA分别使用。其中一个专供DMA访存，另一个专供CPU访存。
- 这种方式不需要总线使用权的申请建立和归还过程，总线使用权是通过不同的周期分别控制的。
- 在这种工作方式下，CPU既不停止主程序的运行也不进入等待状态，在CPU不知不觉中完成了DMA的数据传送，故又称“透明的DMA”方式。



(c) DMA 与 CPU 交替访问

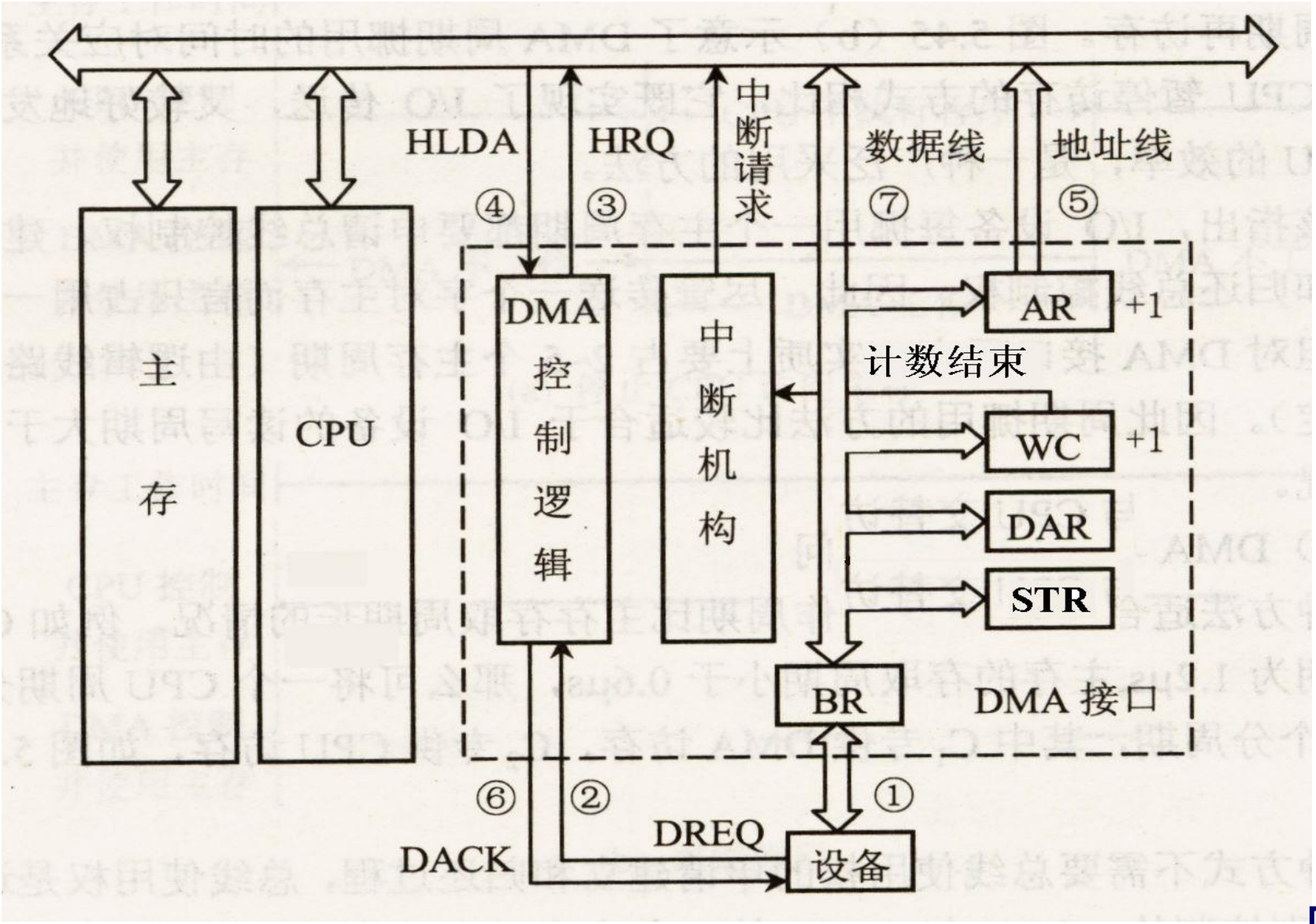
不需要申请建立和归还总线过程, 不停止主程序（折中方案）

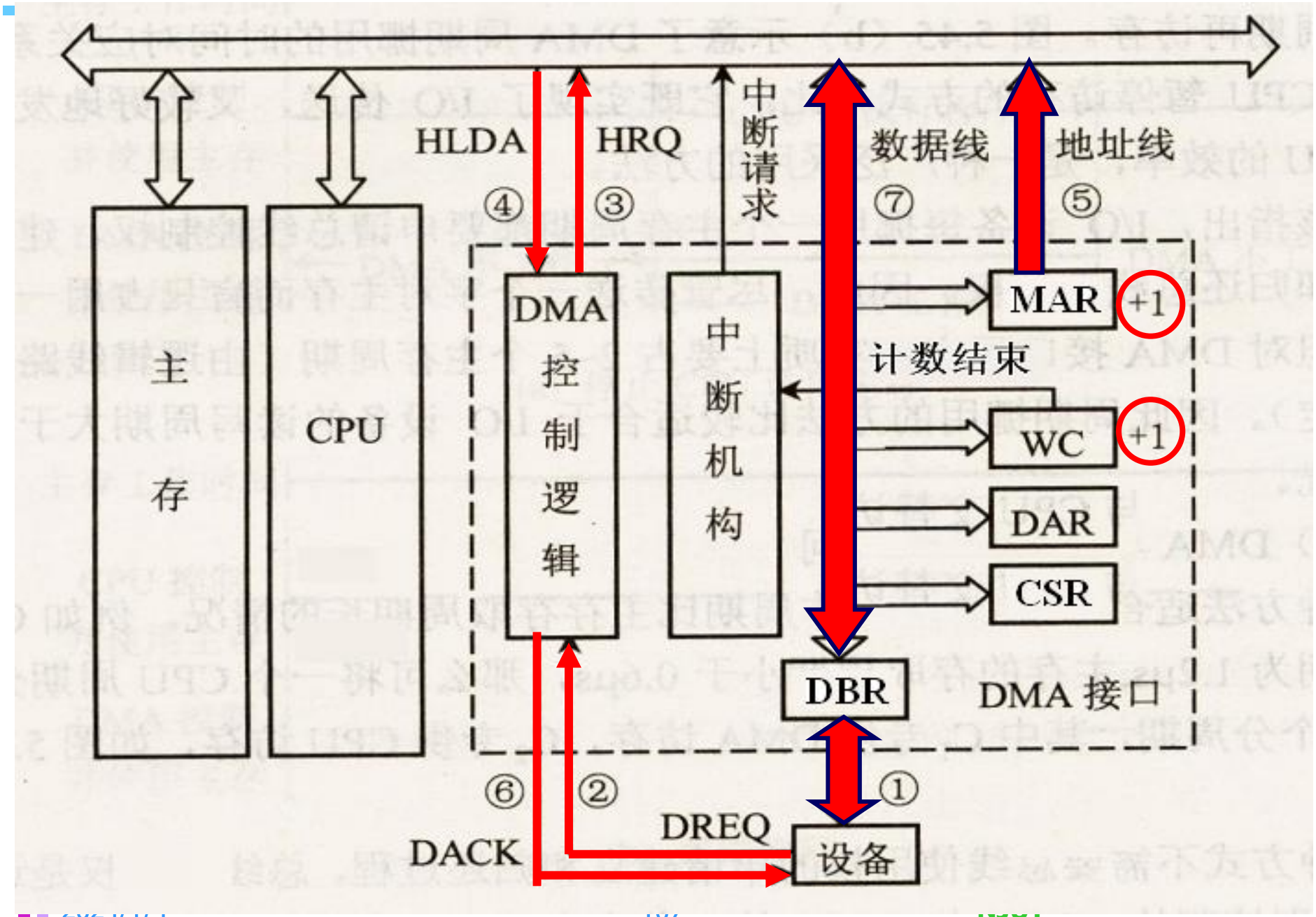
9.4.3 DMA控制器（DMAC）的组成

- 1. DMAC的功能
- ① 接收外设的DMA请求，向CPU发出总线请求信号。请求CPU让出总线。
- ② 当CPU发出DMA响应信号之后，接管对总线的控制，进入DMA方式。
- ③ 对存储器寻址，输出和修改地址信息。
- ④ 向存储器和外设发出相应的读/写控制信号。
- ⑤ 控制传送的字节数，判断DMA传送是否结束。
- ⑥ 在DMA传送结束以后，向CPU发出结束DMA请求信号，释放总线，使CPU恢复对总线的控制，继续正常工作。

2．DMAC的基本组成

- 为了实现DMAC的功能，DMAC内部除需要有接受和发送DMA请求和响应信号的能力外，还应具有地址寄存和计数功能，以便控制对存储器的寻址；具有字节计数器，能够对传送的数据个数进行计数。


主存
CPU
HLDA
④
HRQ
③
中断请求
数据线
⑦
地址线
⑤
DMA控制逻辑
中断机构
AR
+1
计数结束
WC
+1
DAR
STR
BR
DMA 接口
⑥
②
①
DREQ
DACK
设备


主存
CPU
HLDA
HRQ
④
③
中断请求
数据线
地址线
⑦
⑤
DMA控制逻辑
中断机构
MAR
+1
计数结束
WC
+1
DAR
CSR
DBR
DMA 接口
⑥
②
①
DREQ
DACK
设备

- (1)主存地址寄存器AR
- 用于存放主存中需要交换数据的地址。
- 在DMA传送前，须通过程序将数据在主存中的首地址送到主存地址寄存器。在DMA传送过程中，每交换一次数据，将地址寄存器内容加/减1，指向下一单元，直到一批数据传送完毕为止。
- (2)字计数器 WC
- 用于记录传送数据的总字数。
- 在DMA传送过程中，每传送一个字，字计数器减1，直到计数器为0，即最高位产生进位时，表示该批数据传送完毕，DMAC发出DMA传送结束信号。

- (3)数据缓冲寄存器BR
- 用于暂存每次传送的数据。
- 通常DMA接口与主存之间采用字传送，而DMA与设备之间可能是字节或位传送。因此DMA接口中还可能包括有装配或拆卸字信息的硬件逻辑，如数据移位缓冲寄存器、字节计数器等。
- (4)设备地址寄存器DAR
- 存放I/O设备的设备码或表示设备信息存储区的寻址信息。如磁盘数据所在的区号、盘面号和柱面号。具体内容取决于设备的数据格式和地址的编址方式。

- (5)DMA控制逻辑
- 用于负责管理DMA的传送过程，由控制电路、时序电路及命令状态控制寄存器等组成。
- 每当设备准备好一个数据字(或一个字传送结束)，就向DMA接口提出申请(DREQ)，DMA控制逻辑便向CPU请求DMA服务，发出总线使用权的请求信号(HRQ)。待收到CPU发出的响应信号HLDA后，DMA控制逻辑便开始负责管理DMA传送的全过程，包括对主存地址寄存器和字计数器的修改、识别总线地址、指定传送类型(输入或输出)以及通知设备DMA请求已经被响应(DACK)等。

- (6)中断机构
- 当字计数器计数到全“0”时，表示一批数据交换完毕，由计数结束信号通过中断机构向CPU提出中断请求，请求CPU作DMA操作的后处理。
- 必须注意，这里的中断与前面介绍的I/O中断的技术相同，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批数据传送结束。它们是I/O系统中不同的中断事件。
- (7) 控制/状态寄存器 STR
- 存放有关控制和状态信息，如传送方式、读/写状态、传送完毕与否等。

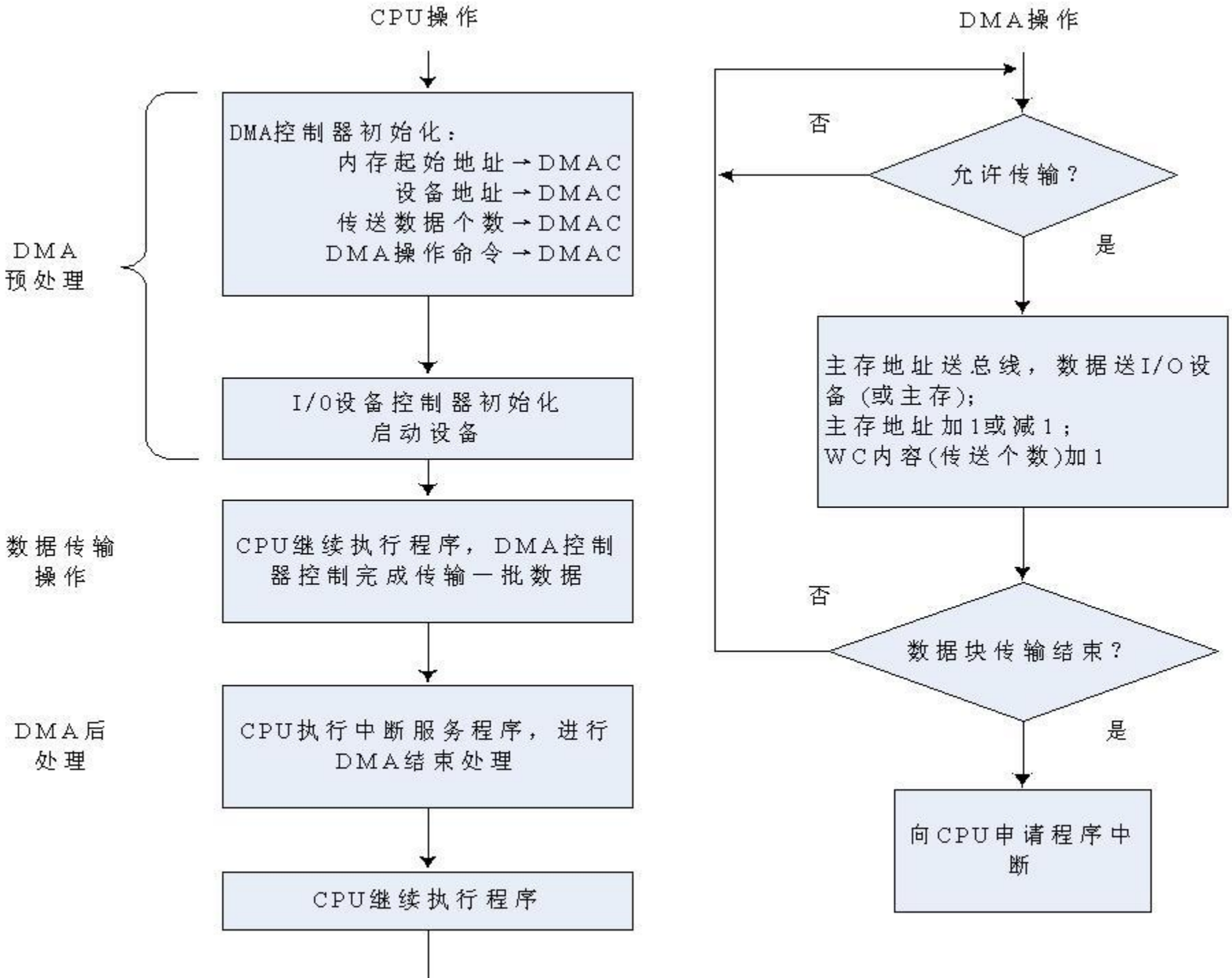
9.4.5 DMA数据传输过程

- 一. 过程
- 1. 预处理
- 在DMAC开始工作之前，CPU必须给它预置如下信息：
- ① 给DMA控制逻辑指明数据传送方向是输入(主存写)还是输出(主存读)。
- ② 向DMA设备地址寄存器送设备号，并启动设备。
- ③ 向DMA主存地址寄存器送入交换数据的主存起始地址。

- ④ 向字计数器送入交换数据的个数。
- 上述工作由CPU执行几条输入输出指令完成，即程序的初始化阶段。这些工作完成后，CPU继续执行原来的程序。
- 当外部设备准备好发送的数据(输入)或上次接受的数据已经处理完毕(输出)时，它便通过DMA接口向CPU提出占用总线的申请，若有多个DMA同时申请，则按轻重缓急由硬件排队判优逻辑决定优先等级。

- 待设备得到主存总线的控制权后，数据的传送便由该DMAC进行管理。
- 2. 数据传送
- DMAC获得总线后，即可按规定的传送方式，进行数据的输入或输出操作，直到将所有数据传输完毕，DMAC将总线交还给CPU。需要时还向CPU发出中断请求。

- 3. DMA后处理
- CPU响应中断后，为DMA传送作结束处理工作。
- ① 校验送入主存的数据是否正确
- ② 决定是否继续用DMA方式传送，还是结束传送
- ③ 测试在传送过程中是否发生了错误
- ④ 判断是否正常结束


CPU操作
DMA操作
DMA控制器初始化：
内存起始地址→DMAC
设备地址→DMAC
传送数据个数→DMAC
DMA操作命令→DMAC
DMA
预处理
I/O设备控制器初始化
启动设备
数据传输
操作
CPU继续执行程序，DMA控制器控制完成传输一批数据
DMA后
处理
CPU执行中断服务程序，进行DMA结束处理
CPU继续执行程序
否
允许传输？
是
主存地址送总线，数据送I/O设备（或主存）；
主存地址加1或减1；
WC内容(传送个数)加1
否
数据块传输结束？
是
向CPU申请程序中断

DMA方式与程序中断的比较

程序中断	DMA方式
以CPU为中心，采用软硬结合，以软件为主的方式，控制设备与主机之间的数据传送。	以主存为中心，采用硬件手段，控制设备与主存间直接进行数据传送。
因为需要程序切换，所以需要保护与恢复现场。	由DMA控制器直接控制数据传送。在数据传送期间，不需要CPU干预，不需保护与恢复现场。
适合于慢速外设。	适合于快速外设。
必须在一条指令执行结束后才能响应。	在一个访存周期结束后即可响应。
可实现多种处理功能	仅用于数据传送

总结

- 直接程序控制（查询式传送）：外设与主机工作完全串行。
- 程序中断：外设与主机工作大部分并行。
- DMA方式：外设与主机工作几乎完全并行，系统开销很小。
- 随着I/O系统的日益复杂，仅靠中断和DMA方式已不能满足要求。因为程序中断的系统开销较大；DMAC可带的设备少，功能简单，所以又出现了通道和I/O处理机方式，从而可以更好地控制I/O系统的工作。

9.5 I/O通道方式

- 1. 概述
- I/O通道是一种能够执行有限I/O指令，并且能够被多台外围设备共享的小型DMA专用处理机。
- 在计算机系统中，通道作为一个独立的I/O控制部件，能执行有限的I/O通道指令，代替CPU管理和控制外设。通道使主机与I/O设备之间能够达到更高的并行程度。

I/O通道的特点

- (1) I/O通道有自己的指令系统，能够独立执行用通道命令编写的输入输出控制程序，产生相应的控制信号控制设备的工作。
- (2) I/O通道可根据需要控制多种不同的设备。
- (3) 每个I/O通道可以连接多个外部设备，每个外设对应一个子通道。

I/O通道与DMA方式的异同

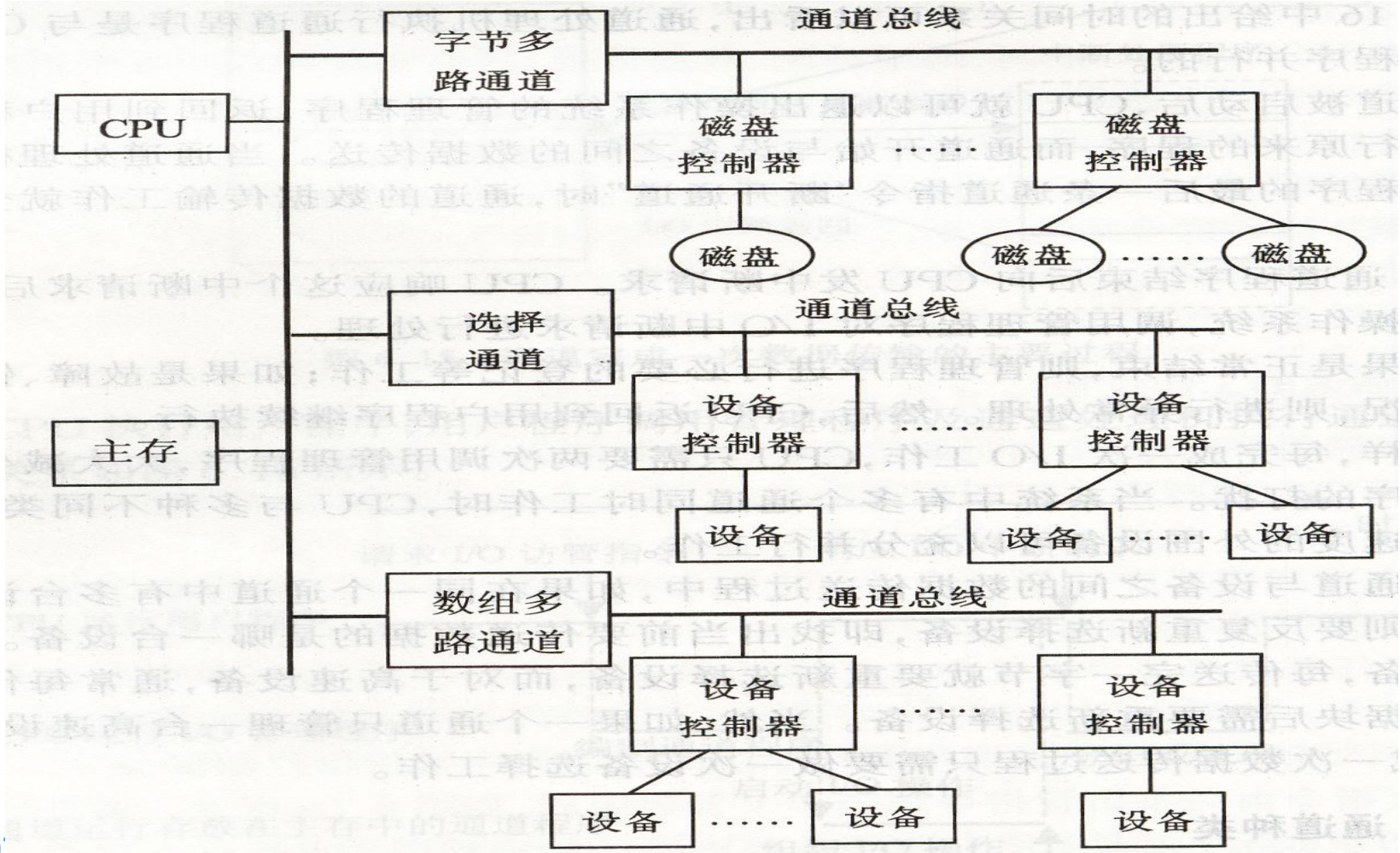
- (1) 相同点
- I/O通道与DMA方式都是在主存与I/O设备之间建立数据通路，用控制器控制数据的直接传送。
- (2) 不同点
- DMA方式通过硬件控制主存与设备之间的信息传送。 I/O通道通过执行通道程序控制主存与设备之间的信息传送。
- DMA方式只能控制少量的同类设备，只能传送数据。 I/O通道可控制多种不同的设备，除传送数据外，还可以接口的初始化、故障诊断与处理等工作。

- 不同点

DMA方式	通道
DMA控制器利用硬件控制主存与设备之间的信息传送。	通道处理机执行通道程序控制主存与设备之间的信息传送。
只能控制少量的同类设备，只能传送数据。	可控制多种不同的设备，除可传送数据外，还可以接口的初始化、故障诊断与处理等工作。

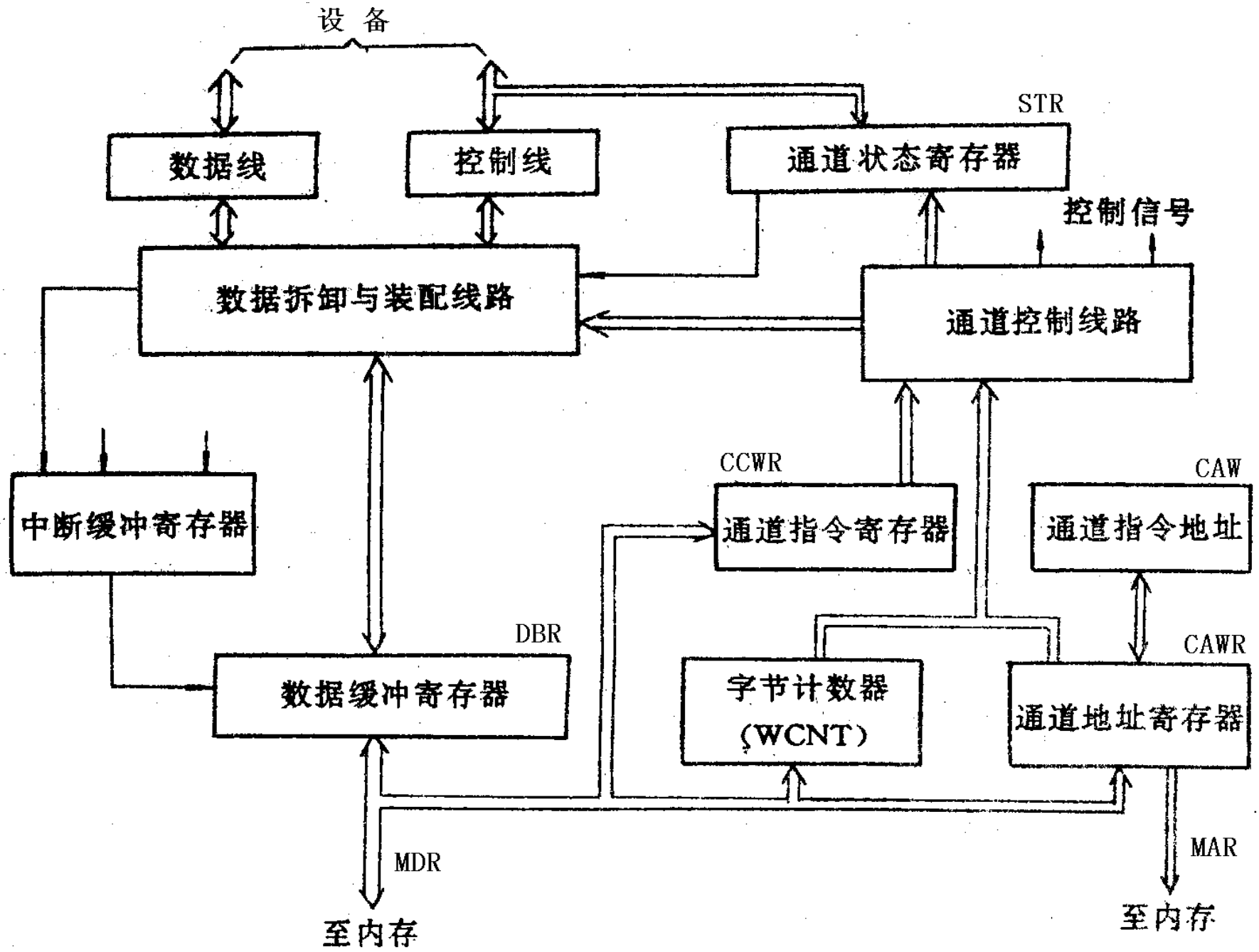
2.带有I/O通道的I/O系统结构

- 主机—通道—设备控制器—设备四级连接方式。



9.5.2 通道的基本结构

- 通道的主要硬件包括寄存器部分和控制部分。
- (1) 寄存器部分
- 数据缓冲寄存器(DBR)、主存地址计数器、传输字节数计数器(WCNT)、通道命令字寄存器(CCWR)、通道地址寄存器(CAR)，通道状态字寄存器(STR) 。
- (2) 控制部分
- 包括数据装配和拆卸控制、数据传送控制、地址分配控制、分时控制等控制逻辑。


设 备
数据线
控制线
STR
通道状态寄存器
控制信号
数据拆卸与装配线路
通道控制线路
中断缓冲寄存器
CCWR
通道指令寄存器
CAW
通道指令地址
DBR
数据缓冲寄存器
字节计数器
(WCNT)
CAWR
通道地址寄存器
MDR
MAR
至内存
至内存

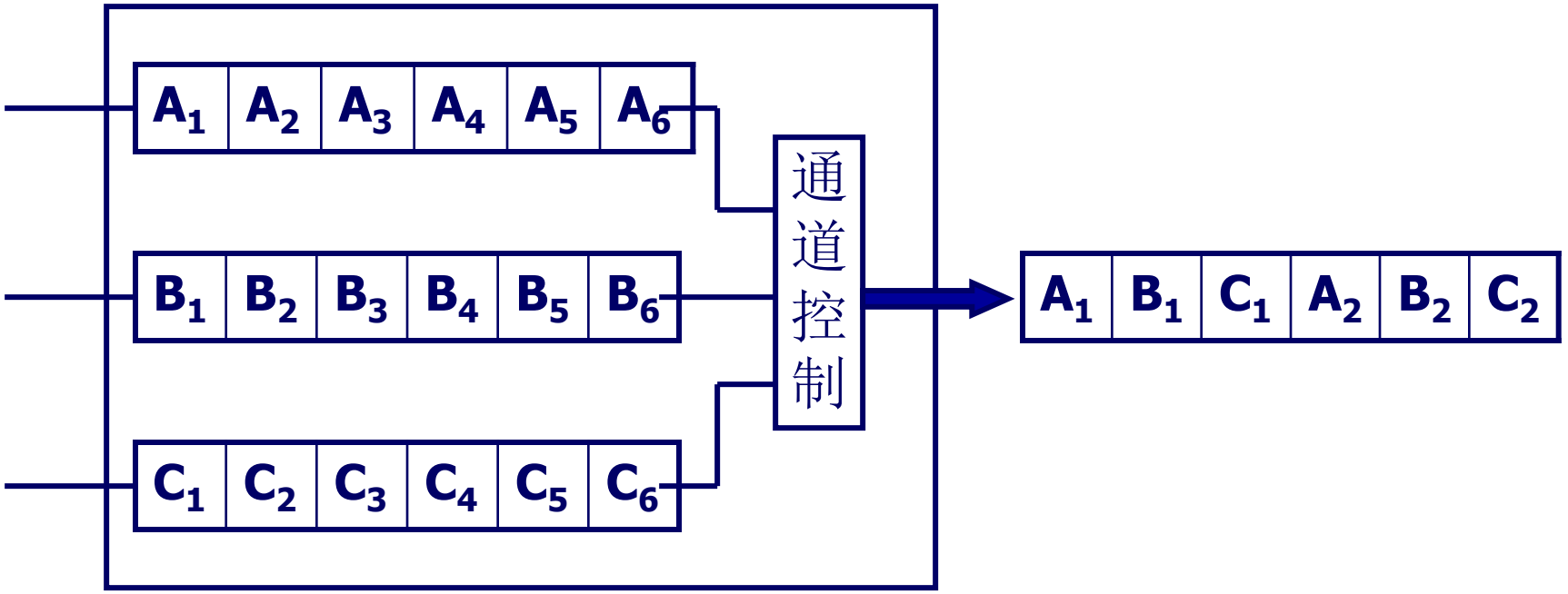
9.5.3 通道的工作过程

- 在具有通道的计算机中，用户程序通常通过调用通道程序来完成数据输入输出的过程。其中**CPU**执行用户程序和管理程序，通道处理机执行通道程序。
- **1.** 在用户程序中使用广义指令进入管理程序，由**CPU**通过管理程序组织 一个通道程序，并启动通道。
- **2.** 通道进行设备选择
- **3.** 通道执行通道程序，控制主存—通道—设备之间的信息传送，直至完成指定的**I/O**工作。
- **4.** 通道信息传送结束

9.5.4 通道类型

- 根据多台外围设备共享通道的不同情况，可将通道分为三种类型。
- **1**．字节多路通道
- 字节多路通道是一种简单的共享通道，可以依靠通道与**CPU**之间的高速数据通路分时地为多台设备服务。
- 连接在通道上的各个设备轮流占用一个很短的时间片传输一个字节。

字节多路通道的信息传送



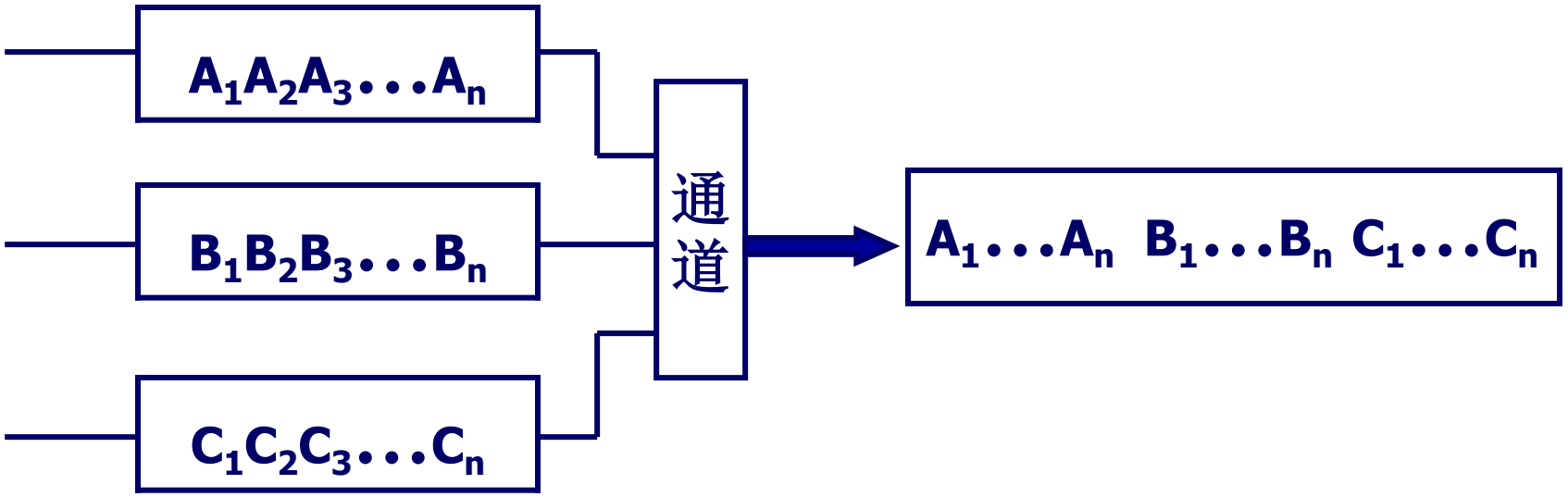
Ai、Bi、Ci 分别为传送的字节信息

慢速设备（分时占用）

2．选择通道

- 选择通道只有一套完整的硬件，以独占的方式工作，逐个轮流地为物理上连接的几台高速外设服务。
- 选择通道在一段时间内单独为一台外设服务，但在不同的时间内可以选择不同的设备。
- 选择通道一旦选中某一设备，通道就进入“忙”状态直到该设备的数据传输工作全部结束为止。
- 选择通道传送的数据宽度是可变的，它为一台外设传送完数据后才转去处理其他外设。

选择通道的信息传送



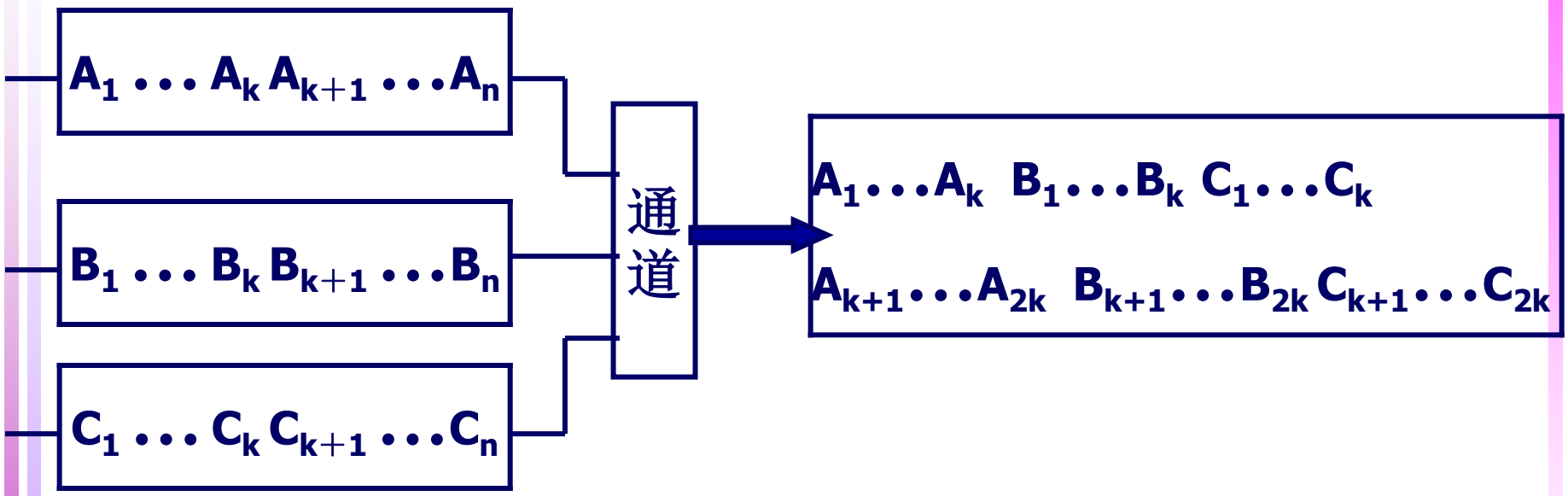
$A_1\cdots A_n$、$B_1\cdots B_n$、$C_1\cdots C_n$ 分别为成组数据

快速设备（独占）

3．数组多路通道

- 数组多路通道将字节多路通道和选择通道的特性结合起来。一个通道可带有多个子通道，各子通道以成组交叉模式轮流使用通道。
- 成组交叉模式：利用通道传送完一组数据（数据块）后让出通道。
- 数组多路通道适用于以数组为单位的高速外设。
- 数组多路通道选择一个高速设备后，先向其发出一个寻找的命令，然后在这个设备寻找期间可以为其他设备服务。在设备寻找完成后再与其真正建立数据连接，并一直维持到一个数据块传输完毕。

数组多路通道的信息传送



$A_1 \cdots A_k$、$B_1 \cdots B_k$、$C_1 \cdots C_k$ 分别为数据块

快速设备（如磁头移动时，换子通道）

9.5.5 I/O指令、I/O通道指令与I/O通道程序

- 1. I/O指令
- I/O指令是计算机系统给用户使用的指令系统的一部分。由CPU负责解释执行。
- 采用通道控制器后，I/O指令不再直接控制I/O数据的具体传送，一般只用于负责启、停I/O通道，查询通道及I/O设备状态，控制I/O通道进行某些操作等。

- 2. I/O通道指令
- I/O通道指令又称为I/O通道控制字（CCW）。CCW是用于编制I/O通道程序的指令，专供I/O通道解释执行，以实现I/O数据传输等I/O操作。
- 3. I/O通道程序
- 用CCW编制成有关的I/O通道程序，在CPU的命令下启动通道程序实现有关I/O操作，完成I/O通道对I/O设备的具体控制。

- 在早期的I/O通道实现中，I/O通道程序存放在主机的主存中，即I/O通道与CPU共用主存。
- 目前，计算机系统中通常为I/O通道配置了局部存储器。这样可以减少CPU与I/O通道之间的冲突，进一步提高CPU与I/O通道工作的并行度。