第三章 存储、中断、总线与I/O系统

Dr. Feng Li

fli@sdu.edu.cn

https://funglee.github.io

目录

- 存储系统的基本要求和并行主存系统
- 中断系统
- 总线系统
- 输入/输出系统

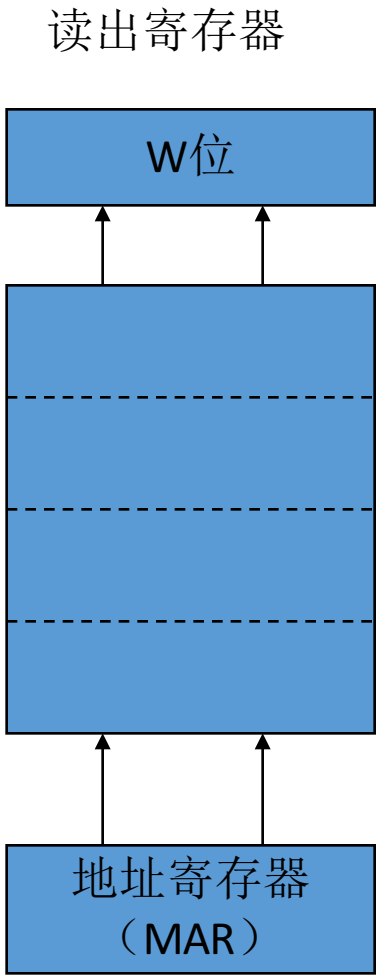
存储系统的基本要求

- 大容量、高速度、低价格
- 存储器容量 $S_M = W \times l \times m$
  - W为存储器字长，l为存储器字数，m为并行存储器体数
- 存储器速度
  - 访问时间T_A：存储器从接收访存读申请至信息被读到数据总线上的时间
  - 存储周期T_M：连续启动一个存储器所需要的间隔时间（$T_M > T_A$）
  - 频宽（带宽）B_M：存储器可提供的数据传送速率
    - 最大频宽：单体$B_M = W/T_M$，m个存储体并行的最大频宽$B_M = W \times m/T_M$
    - 实际频宽
- 存储器价格
  - 存储体及外围电路的价格 $C = cS_M$（c 表示每一位的价格）

- 存储器速度需要与CPU速度相匹配，但采用单一工艺的存储器难以同时满足容量、价格和速度的要求
- 采用由多种工艺存储器组成的存储器系统，使所有信息以不同的方式分布于不同的存储器上
  - 例如，主存＋辅存的存储器系统。辅存中的较大的程序分成有重叠的块，根据算题的需要将当前用到的块调入主存中，覆盖或替换掉那些在主存中已不用的内容
  - 该方法虽然可以使程序员在小容量主存中运行较大的程序，但程序块在主存和辅存之间的调入／调出延长了程序的运行时间，并增加了编程的难度
- 为弥补CPU和存储器在速度上的差异，需要引入并行和重叠技术，构成并行主存系统，提高主存频宽

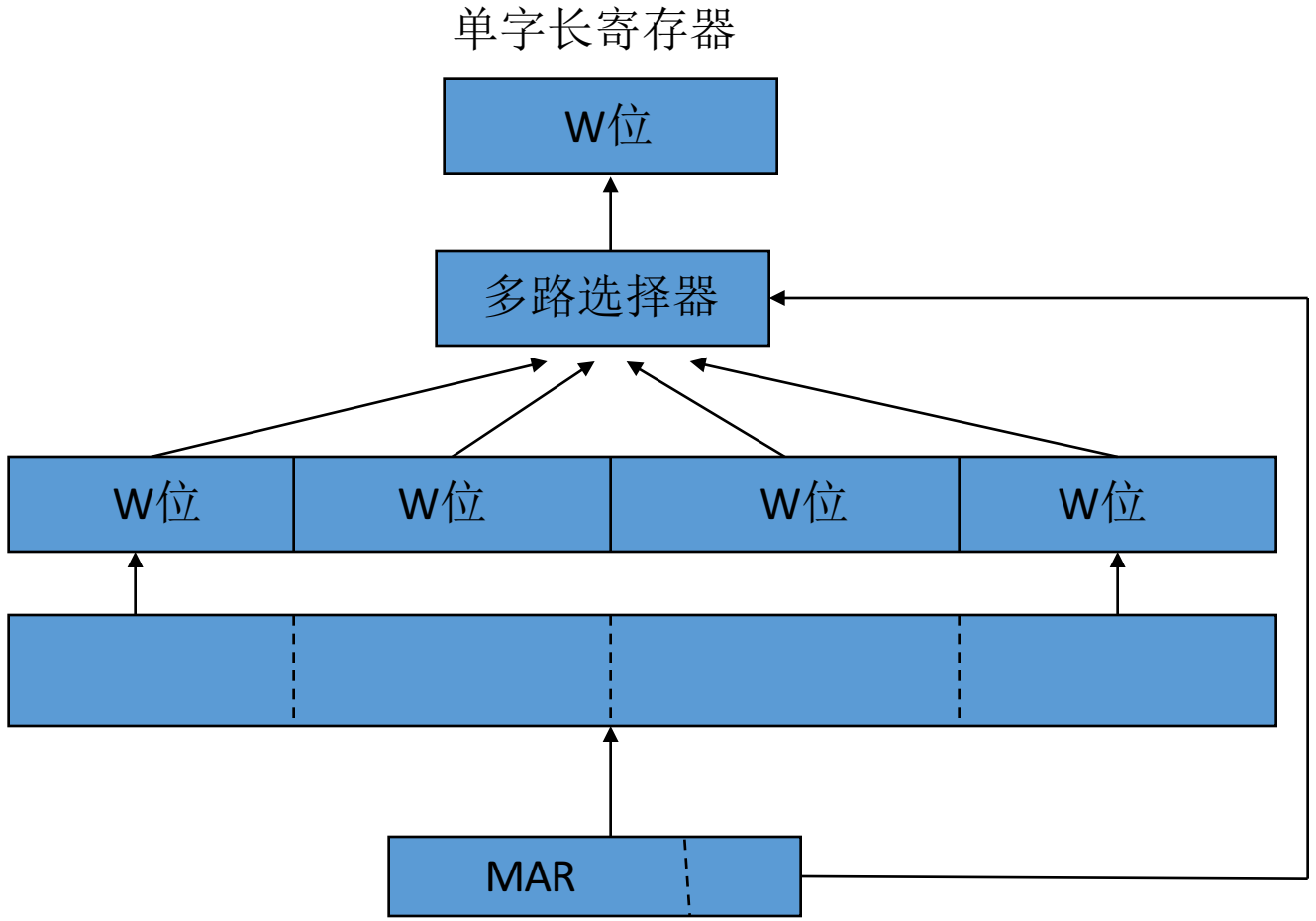
单体单字存储器

- 当并行的存储器共用一套地址寄存器和地址译码电路时称为单体方式
- 每次访问一个存储器的一个字
- 最大频宽$B_M = W/T_M$



并行主存系统

- 单体多字
  - 把存储器字长增加n倍，为保持总容量不变，把存储器字数（地址数）相应减少n倍
  - 在一个存储周期内访问n个数据
  - 最大频宽$B_M = W \times n / T_M$

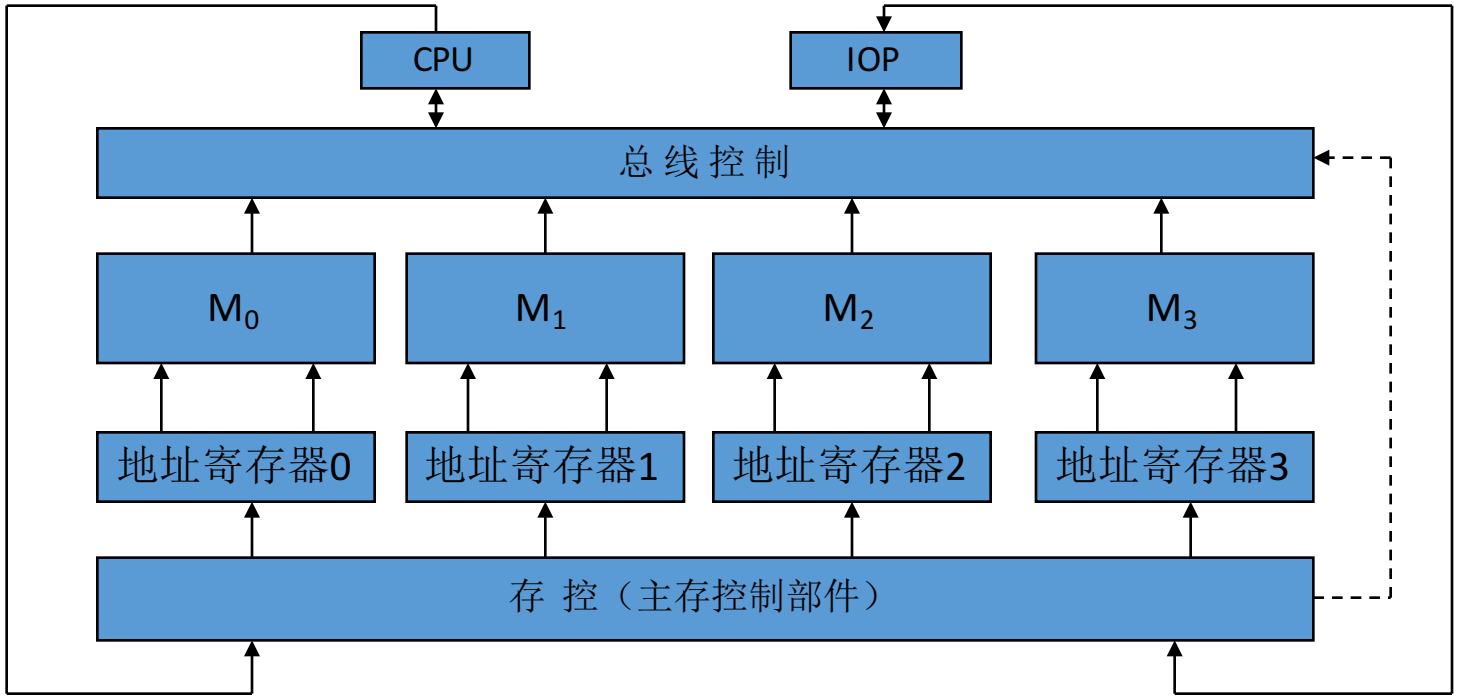


单体多字

- 取指令冲突：在遇到程序转移而且转移成功时，一个存储周期中读出的n条指令中，后面的指令将无用
- 读操作数冲突：一次同时读出的n个操作数，不一定都有用
- 写数据冲突：必须凑齐n个数之后才能一起写入存储器
- 读写冲突：当要读出的一个字和要写入存储器的一个字处在同一个存储字内时，无法在一个存储周期内完成

并行主存系统

- 多体单字
  - 由多组容量小、字长短的存储器片子组成，每个存储片子都有自己的地址译码、读 / 写驱动等外围电路
  - 每个存储片子都有独立的地址译码、读 / 写驱动等外围电路

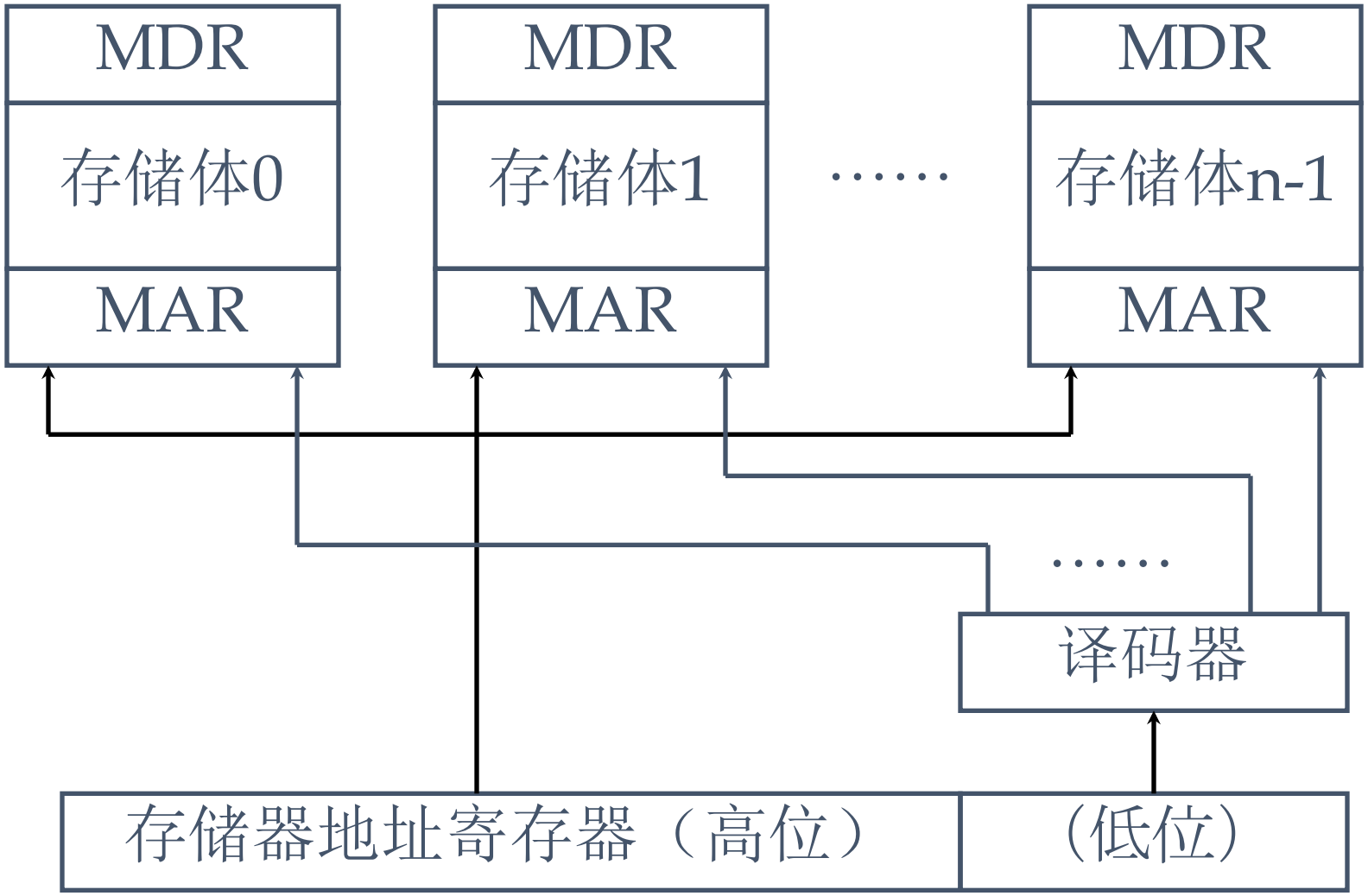


模m低位交叉编址

- CPU字在主存中按模m低位交叉编址
  - 单体容量为l的m个分体，其M_j体的编址模式为$m\times i+j$，其中$i=0,1,2,\cdots,l-1$，$j=0,1,2,\cdots,m-1$
- 寻址规则
  - 体地址 $j=A\ mod\ m$
  - 体内地址 $i=A/m$
    - $M_0: 0, m, 2m, \cdots, m(l-1)$+0
    - $M_i: i, m+i, 2m+i, \cdots, m(l-1)+i$
- 适合于单处理机内的数据存取和带Cache的主存

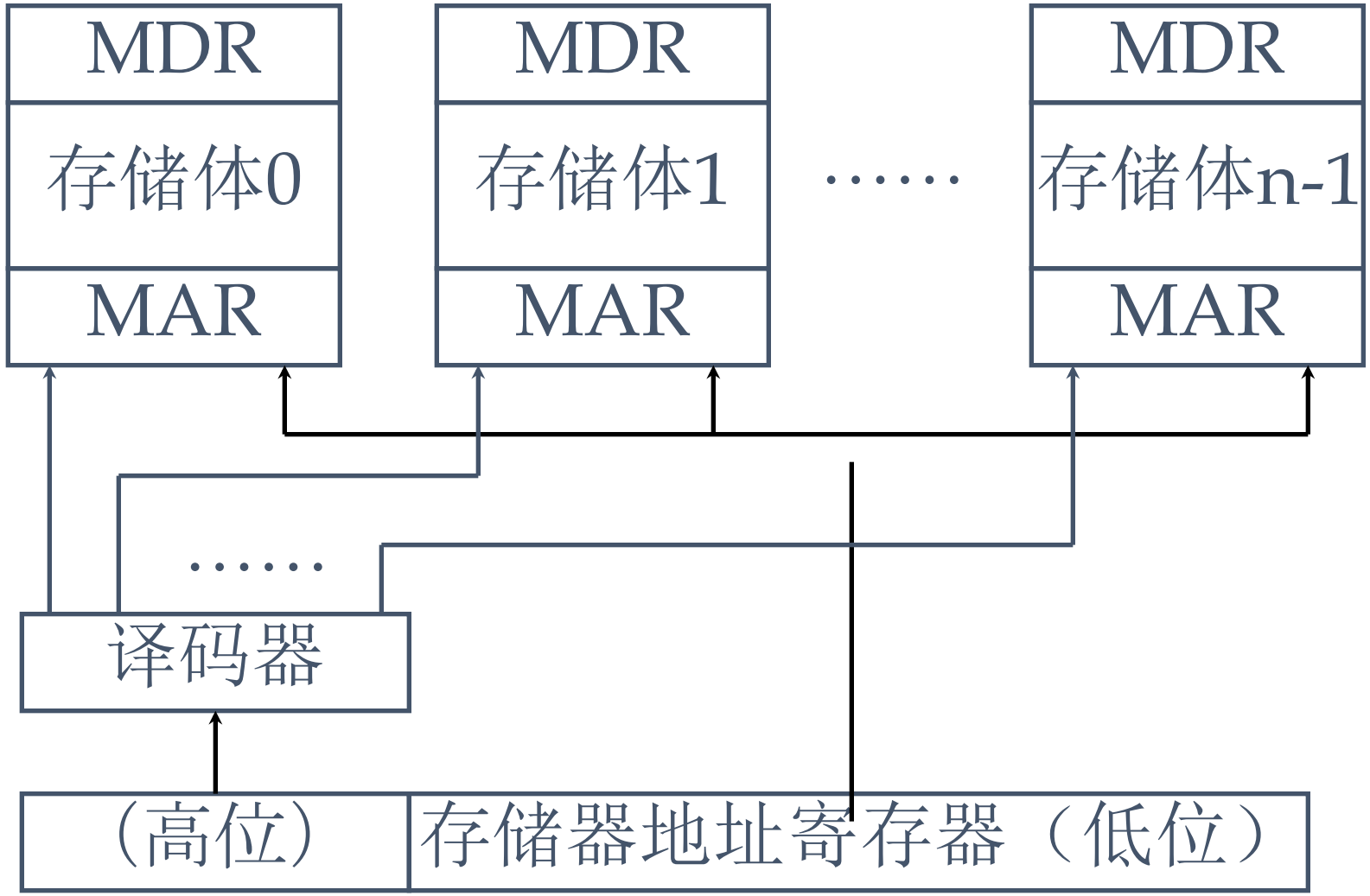
模4低位交叉编址

模体	地址编址序列	对应二进制地址码最末二位状态
M_0	0,4,8,12,…,4i+0,…	00
M_1	1,5,9,13,…,4i+1,…	01
M_2	2,6,10,14,…,4i+2,…	10
M_3	3,7,11,15,…,4i+3,…	11


MDR
MDR
MDR
存储体0
存储体1
……
存储体n-1
MAR
MAR
MAR
……
译码器
存储器地址寄存器（高位）
（低位）

模m高位交叉编址

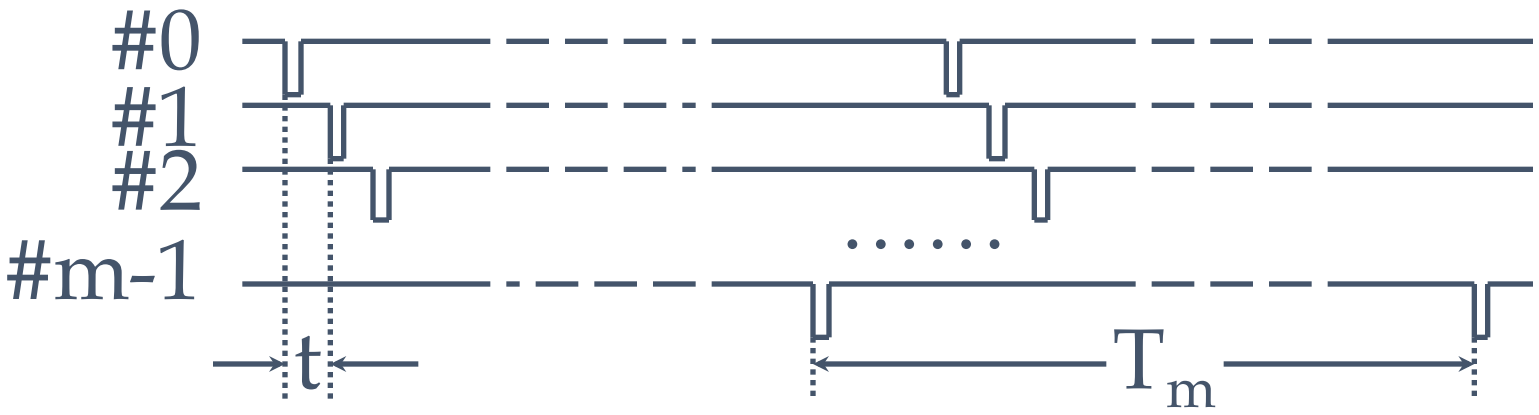
- CPU字在主存中按模m高位交叉编址
  - 单体容量为l的m个分体，其M_j体的编址模式为$m\times j+i$，其中$i=0,1,2,\cdots,l-1$，$j=0,1,2,\cdots,m-1$
- 寻址规则
  - 体地址 $j=A\ /\ m$
  - 体内地址 $i=A\ mod\ m$
    - $M_0: 0,1,2,\cdots,l-1$
    - $M_i: il, il+1,\cdots,(i+1)l-1$
- 适合于共享存储器的多机系统，适用于指令和数据分别存于不同分体中


MDR
MDR
MDR
存储体0
存储体1
……
存储体n-1
MAR
MAR
MAR
……
译码器
（高位）
存储器地址寄存器（低位）

存储体分时启动

- 实际上是一种采用流水线方式工作的并行存储器
- 理论上，存储器的速度可望提高m倍 （ m 是存储器个数）

$$t = \left\lfloor \frac{T_m}{m} \right\rfloor$$



多体单字 Vs 单体多字

- 总价格相近
- 多体单字的频宽要高得多
  - 多体单字系统只要求m个地址不发生分体冲突（即没有发生两个以上地址同属于一个分体），即使地址之间可以不是顺序的，也可并行读出
  - 单体多字系统要求可并行读取的地址是顺序的，且处于同一主存单元
- 可以将多体并行存取与单体多字相结合，构成多体多字交叉存储器
- 能并行读出多个CPU字的单体多字、多体单字和多体多字的交叉访问主存系统，称为并行主存系统

- 提高m的值并不能线性提高主存的实际频宽
  - 转移指令造成的系统效率的下降
  - m越高，存储器数据总线越长，总线上并联的负载越重，有时会造成门的级数的增加，提高了工程难度，增加了传输迟缓

定量分析

- 假设对于具有m路独立存储体的并行主存系统，CPU发出地址$A_1, A_2, A_3, \cdots, A_q$的访问申请队列，
  - 在每一个主存周期之间，扫描队列，截取

$$A_1, A_2, A_3, \cdots, A_k$$

  - $A_1, A_2, A_3, \cdots, A_k$ 中没有两个或者两个以上地址处于同一存储体中（ $k \leq m$ ）
  - 截取k个地址的队列，能同时访问k个存储体（ $k=1, 2, 3, \cdots, m$）

设$p(k)$表示申请长度为k的概率，显然k的平均值（即每个周期能访问的平均字数）

$$\mathrm{B} = \sum_{k=1}^{m} kp(k)$$

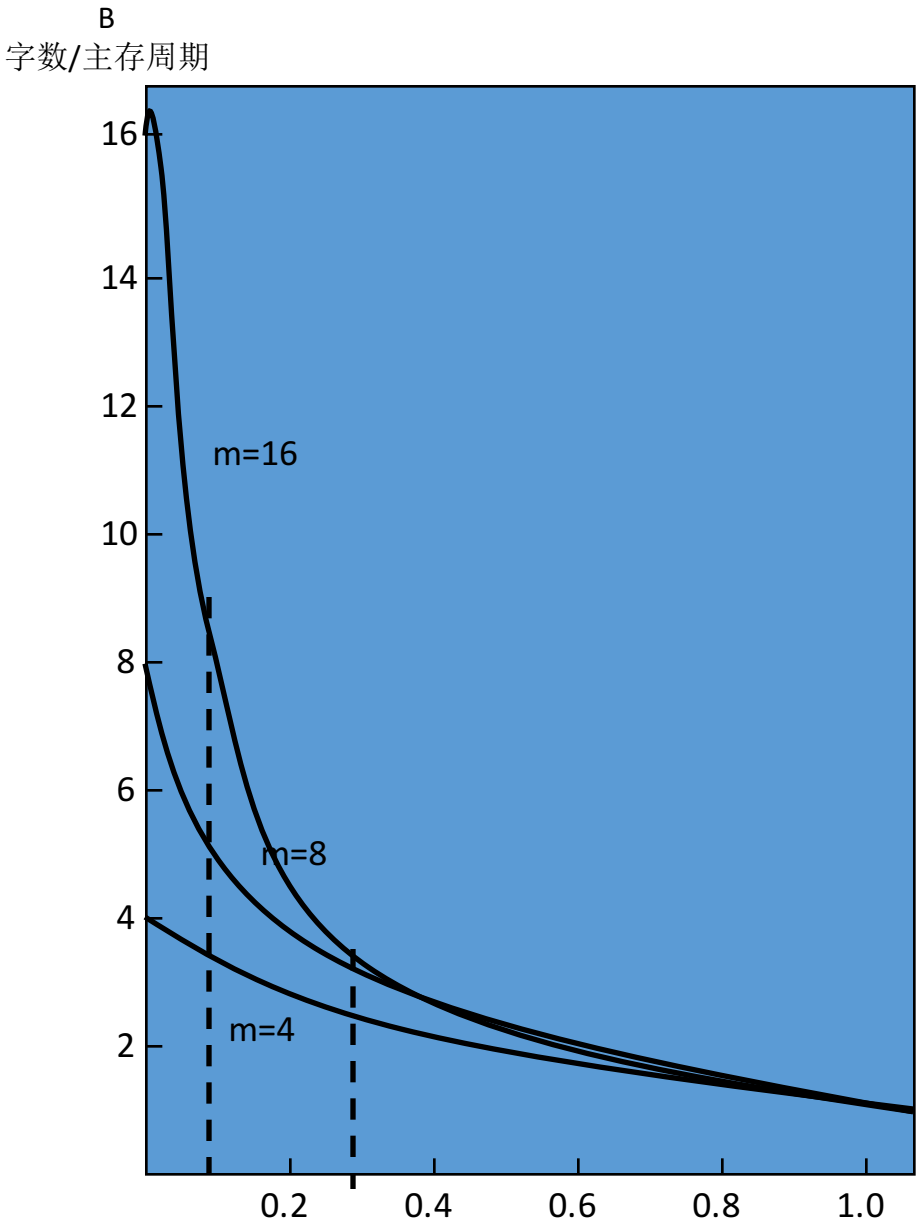
假设转移概率λ

$$p(1) = \lambda$$
$$p(2) = (1-\lambda)\lambda$$
$$p(3) = (1-\lambda)^2\lambda$$
$$p(k) = (1-\lambda)^{k-1}\,\lambda$$
$$p(m) = (1-\lambda)^{m-1}$$

因此

$$B = \sum_{i=0}^{m-1} (1-\lambda)^i = \frac{1-(1-\lambda)^m}{\lambda}$$

- λ=1时， B= 1
- λ=0时， B= m ,效率最高
- λ> 0.3时， m=4、8、16的B差别不大
- λ< 0.1时,m值的大小对B的改进会有显著影响
- 一般取m<=8 m = 2,4,8
- 对数据来讲，由于随机性大，因此靠加大m不一定满足要求。



举例

- 设访存申请队列的转移概率λ为25%，比较在模32和模16的多体单字交叉存储器中，每一个周期能访问到的平均字数。
- 每一个周期能访问到的平均字数为：$B = \dfrac{1 - (1-\lambda)^{m}}{\lambda}$
- 将λ=25%，m=32代入上式，可求得：$B = \dfrac{1 - 0.75^{32}}{32} = 4$
- 将λ=25%，m=16代入上式，可求得：$B = \dfrac{1 - 0.75^{16}}{32} = 3.96$
- 将λ=25%，m=8代入上式，可求得：$B = \dfrac{1 - 0.75^{8}}{32} \approx 3.6$

结论

- 从最坏情况考虑，设所有申请（包括指令和数据）都是全随机的，用单来单服务、先来先服务的排队论模型进行模拟，可得出随 m 的提高，主存频宽只是以近似$\sqrt{m}$的关系改善。
- 当然，指令流和数据流也不会是全随机的，因此，B的值总是会比$\sqrt{m}$的值要大
- 正是因为程序的转移概率不会很低，数据分布的离散性较大，所以单靠增大m来提高并行主存系统的频宽是有限的，而且性能价格比还会随m的增大而下降，就必须从系统结构上进行改进，采用存储体系

中断系统

- CPU中止正在执行的程序，转去处理随机提出的请求，待处理完毕后，在回到原先被中断的程序继续恢复执行的过程称为“中断”
- 响应和处理各种中断的软硬件总体称为中断系统
  - 内部中断：由CPU异常引起
  - 外部中断：由中断信号引起
    - 可屏蔽中断
    - 不可屏蔽中断
  - 软件中断：由自陷指令引起，用于操作系统服务

中断系统

- 中断源：引起中断的各种事件
- 中断请求：中断源向中断系统发出中断的申请
- 中断响应：就是允许其中断CPU现行程序的运行，转去对该请求进行预处理，包括保存好断点现场，调出有关处理该中断的中断服务程序
  - 在多数机器上用交换新旧程序状态字PSW（程序状态寄存器）来实现
  - 为了某种需要，中断系统也可对中断请求进行屏蔽，使之临时得不到响应

中断的分类

- 对中断进行分类，对每一类中断给定一个硬件的中断服务入口，再由软件分支转入相应的中断处理程序
- IBM 370
  - 机器校验中断（64位）：指明设备故障，如电源故障、运算电路误操作、主存错误等
  - 访管中断（8位）：申请操作系统介入
  - 程序性中断（16位）：指令和数据的格式错误、程序异常、事件检测
  - 外部中断（16位）：定时器中断、外部信号中断、中断键中断
  - I/O中断（16位）：I/O操作完成，或者I/O设备或通道故障
  - 重新启动中断：操作员或另一台CPU要启动一个程序

中断（Interruption）和异常（Exception）

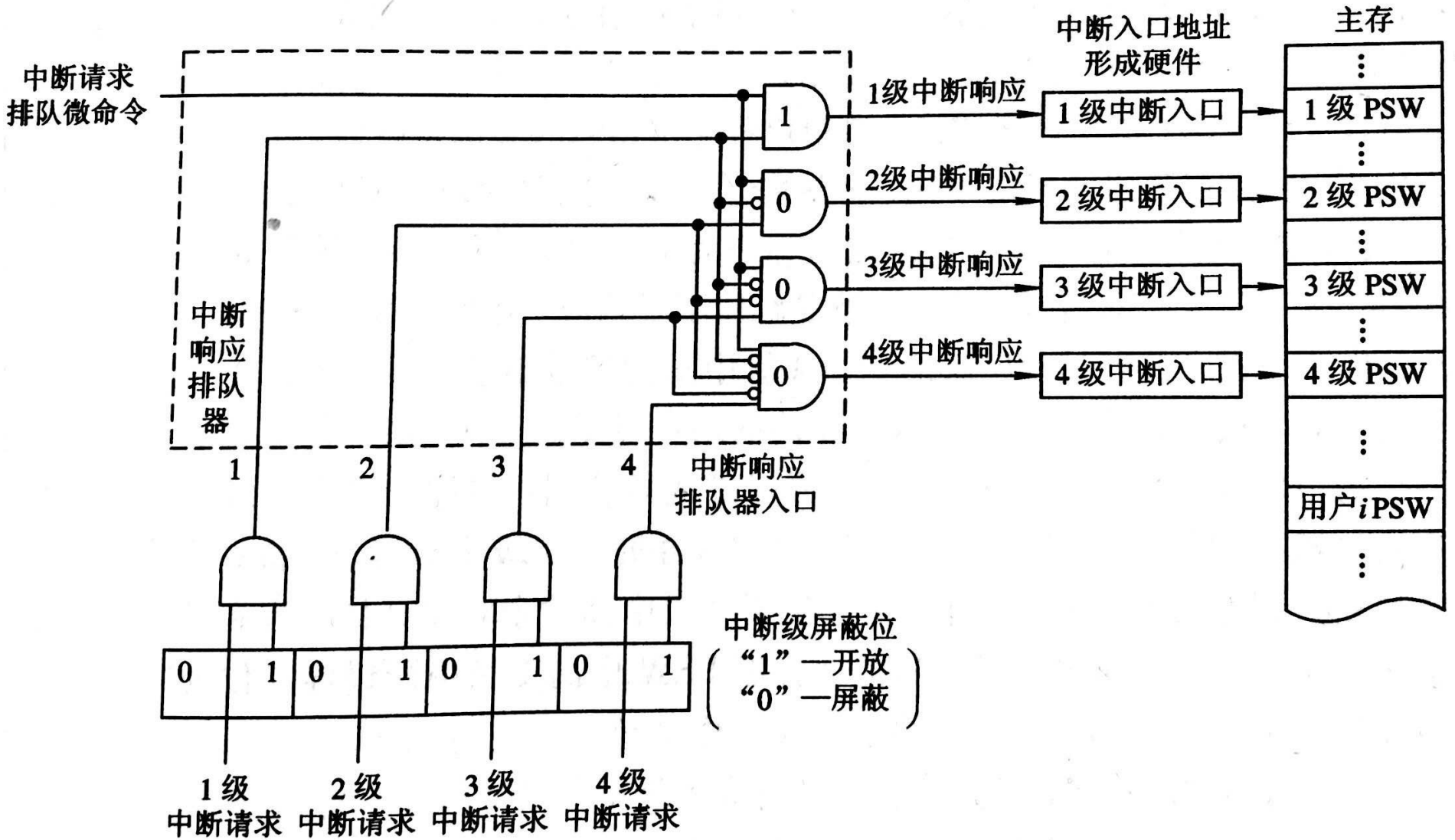
- 异常：由执行现行指令引起的临时停止事件，如运算结果溢出、页面失效等，一般不能屏蔽，应予立即响应和处理
- 中断：指那些与当前进程运行无关的请求临时停止的事件，如机器故障中断请求、外设中断请求、定时器中断请求等，可以屏蔽，未被响应的中断源保留在中断字寄存器中，直至屏蔽解除后可得到响应和处理

中断的分级

- 同一类中各个中断请求的响应和处理的优先次序，一般由软件或通道来管理；而不同类的中断要根据其性质、紧迫性、重要性以及软件处理的方便性分成不同级别
- IBM 370
  - 第一级：机器校验中断
  - 第二级：管理程序调用、程序性中断
  - 第三级：外部中断
  - 第四级：I/O中断
  - 最低级：重新启动中断
- 有的机器存在第0级中断，当机器因故障重叠发生或无法排除，完全不能正常工作时，告急。不是真正的中断级，不参加中断级排队，中断后也无法自行恢复

中断响应次序和处理次序

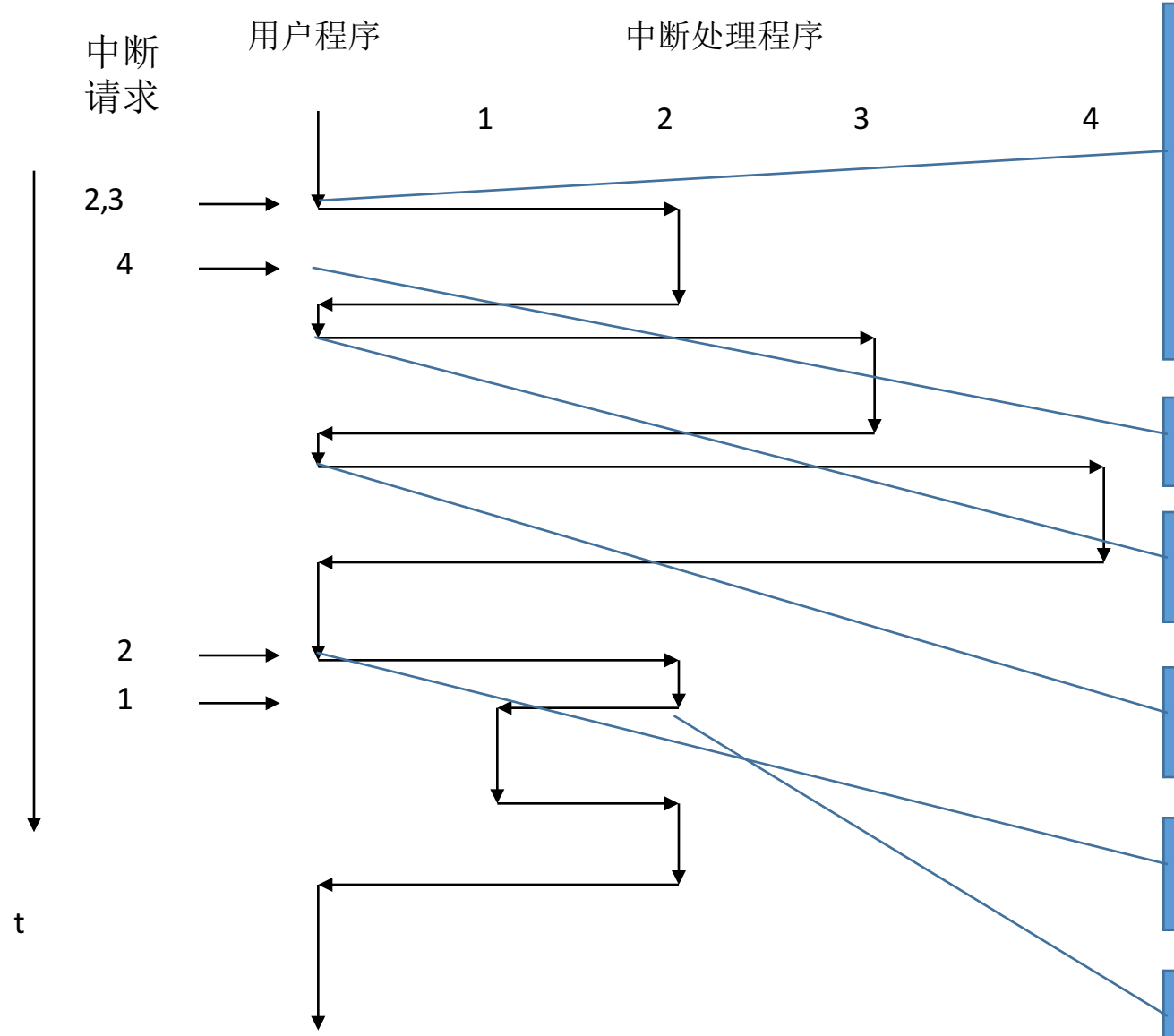
- 中断响应次序
  - 同时发生多个不同中断类的中断请求时，中断响应硬件中的排队器所决定的响应次序
  - 一般在处理某级中某个中断请求时，是不能被与它同级的或比它低一级的中断请求所中断，只有比它高一级的中断请求才能中断其处理，等响应和处理完后再继续处理原先的那个中断请求
- 中断处理次序
  - 中断处理完的次序
  - 由于中断处理程序也可能被中断，中断处理次序可能不同于中断响应次序


中断请求
排队微命令
中断
响应
排队
器
1
2
3
4
中断响应
排队器入口
1
0
0
0
1级中断响应
2级中断响应
3级中断响应
4级中断响应
中断入口地址
形成硬件
1 级中断入口
2 级中断入口
3 级中断入口
4 级中断入口
主存
1 级 PSW
2 级 PSW
3 级 PSW
4 级 PSW
用户iPSW
中断级屏蔽位
“1” —开放
“0” —屏蔽
0 1 0 1 0 1 0 1
1 级
中断请求
2 级
中断请求
3 级
中断请求
4 级
中断请求

举例

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

- 如果中断级屏蔽位为1，则表示对该级中断开放，允许其进入响应排队器；如果为0，则对该级中断屏蔽
- 正在执行某级中断处理程序时，现行PSW中应屏蔽同级和低级的中断请求
- 为保证中断嵌套时的处理，应设置一个返回地址堆栈，以实现正确返回
- 要让各级中断处理次序和各级中断响应次序都一样，都是1-2-3-4，就只需按上表设置好即可
- 用户程序（目态程序）不能屏蔽任何中断



2、3级中断均进入排队器，当中断请求排队微命令到来时，优先响应2级中断。

1) 中断当前用户程序
2) 将中断断点压入返回地址堆栈
3) 交换新旧PSW实现程序的切换
4) 设置二级中断处理程序的中断级屏蔽位1000，屏蔽3级中断

2级中断处理程序执行过程中，屏蔽4级中断

3、4级中断进入排队器，优先处理3级中断

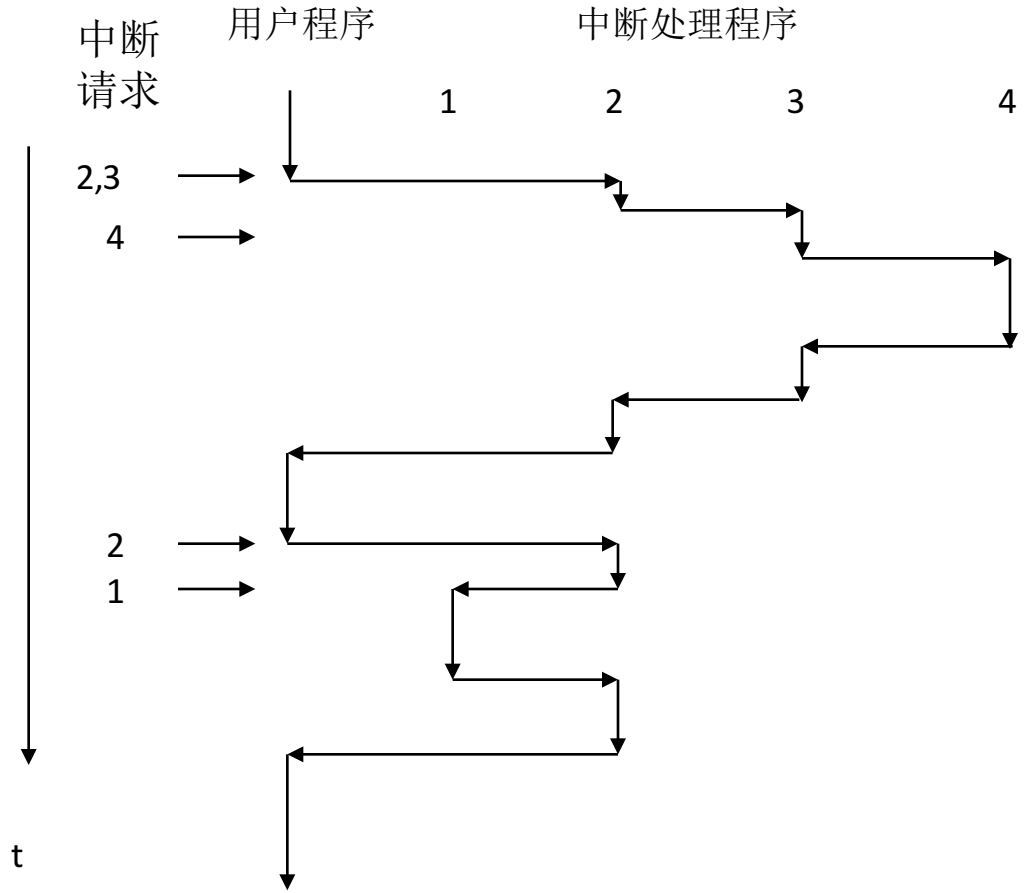
处理4级中断

处理2级中断

中断2级中断处理程序，优先处理1级中断

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



结论

- 只要操作系统根据需要软的方法，改变各级中断处理程序的中断级屏蔽位状态，就可以改变实际的中断处理。这就是中断系统采用软、硬件结合的好处
- 中断响应用排队器硬件实现可以加快响应和断点现场保护，中断处理采用软的技术可以提供很大的灵活性，因此，中断系统的软、硬件功能的实质是中断处理程序软件和中断响应硬件的功能分配
- 为了改善性能，用软件实现的功能，可以部分改用硬件来实现

中断系统的软、硬件功能分配

- 中断系统的功能
  - 中断请求的保存和清除
  - 优先级的确定
  - 中断断点及现场的保护
  - 对中断请求的分析和处理
  - 中断返回
- 高的响应速度，其次是中断处理的灵活性
- 中断系统的软、硬件功能分配实质上是中断处理程序软件和中断响应硬件的功能分配
- 随着硬件和器件的发展，其价格在不断下降，为了改善性能，加快中断处理，也可以在软件实现的功能上不断增加硬件的支持

- 早期，为了简化硬件、降低成本，大部分功能都是由软件完成，响应时间长
- 中断响应及其次序由程序查询软件的方法改为用中断响应排队器硬件实现，中断源的分析也由程序查询改为硬件编码，直接或经中断向量表间接形成各中断处理程序的入口地址
- 对每级中断经中断响应硬件形成该级中断程序状态字地址的入口，再把中断源的状况以中断码的形式经旧程序状态字告知中断处理程序

- 中断现场包括软件状态和硬件状态
- 软件状态在主存中，适宜用中断处理程序保存
- 硬件状态日益复杂组成程序状态字，中断响应硬件将程序状态字保存到指定主存单元（或区域）
- 但是硬件状态如果很多，完全利用中断响应硬件来保存会极大降低中断响应的速度。所以，要根据机器的规模和使用制定硬件状态的保存策略

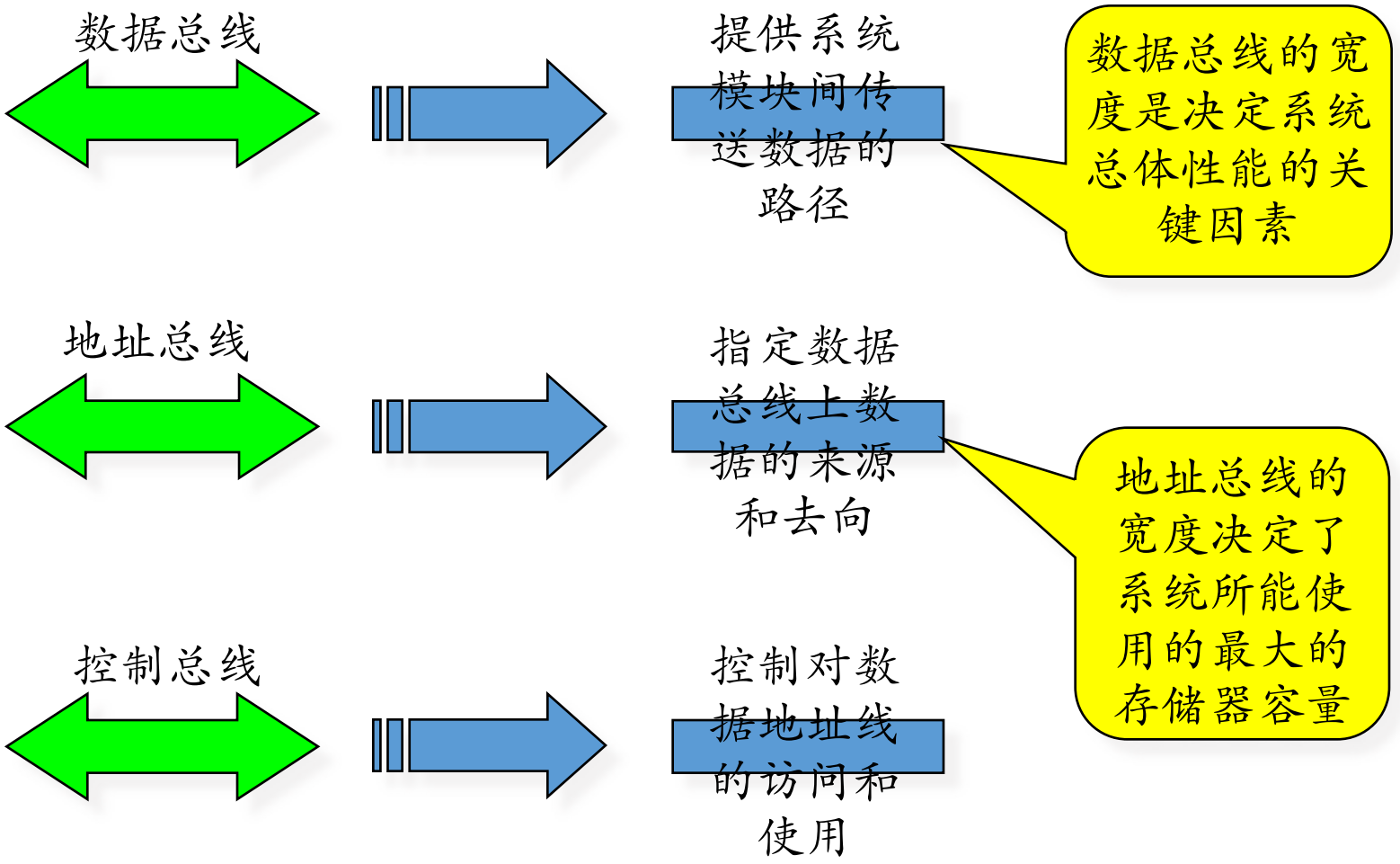
总线系统

- 总线是用于互连计算机、CPU、存储器、I/O端口及外部设备、远程通信设备间信息传送通路的集合
- 总线与其相配合的附属控制电路统称为总线系统
- 按照信息传送功能、性能的不同，总线系统包括数据线、地址线、时序和中断信号等控制 / 状态线、电源线、地线以及备用线等
  - 数据线的根数决定同时传送的数据位数，即数据通路宽度
  - 地址线的根数决定直接寻址的范围
  - 控制 / 状态线决定总线的功能和使用能力
  - 备用线用于系统功能的扩充

总线的分类

- 按系统中的位置：
  - 芯片级：CPU芯片内的总线；板级：连接插件版内的各组件，局部总线、内部总线；系统级：系统间或主机与I/O接口或设备之间的总线
- 按信息传送方向：
  - 单向传输
  - 双向传输
    - 半双向可沿相反方向传送，但同时只能向一个方向传送
    - 全双向：可以同时向两个方向传送，速度快、造价高、结构复杂
- 按使用方法：
  - 专用总线：只连接一对物理部件的总线
  - 非专用总线：可以被多种功能或多个部件分时共享，同一时间只有一对部件可使用总线进行通信

总线的类别


数据总线
提供系统模块间传送数据的路径
数据总线的宽度是决定系统总体性能的关键因素
地址总线
指定数据总线上数据的来源和去向
地址总线的宽度决定了系统所能使用的最大的存储器容量
控制总线
控制对数据地址线的访问和使用

专用总线

- 只连接一对物理部件的总线
- 优点：
  - 多个部件可以同时收发信息，不争用总线，系统流量高
  - 通信时不用指明源和目的，控制简单；
  - 任何总线的失效只会使连接该总线的两个部件不能直接通信，但它们仍可以通过其他部件间接通信，因而系统可靠
- 缺点：
  - 总线数多，总线数与部件数成平方倍关系增加
  - 难以小型化、集成电路化
  - 总线较长时，成本相当高
  - 利用率低
  - 不利于系统模块化
- 只适用于实现某个设备（部件）仅与另一个设备（部件）连接

非专用总线

- 可以被多种功能或多个部件所分时共享，同一时间只有一对部件可使用总线进行通信
  - 单总线：低性能微、小型机，既是主存总线又是I/O总线
  - 双总线或多总线：大型系统
  - 远距离通信总线：多个远程终端共享主机系统
  - 纵横交叉开关：多处理机系统互联
- 优点：
  - 总线数少、造价低
  - 总线接口标准化，模块性强
  - 可扩充能力强
  - 部件的增加不会使电缆、接口和驱动电路激增
  - 易用多重总线提高总线的带宽和可靠性，使故障弱化
- 缺点：
  - 系统流量小，经常会出现争用总线的情况，降低效率，共享总线失效会导致系统瘫痪

总线的控制方式

- 非专用总线上的多个设备或者部件同时请求使用总线，使得由总线控制机构按某种优先次序裁决，保证只有一个高优先级的申请者首先取得对总线的使用权
- 集中式控制：总线控制机构基本集中在一起，不论是在连接到总线的一个部件中，还是在单独的硬件中
- 分布式控制：总线控制逻辑分散在连到总线的各个部件间

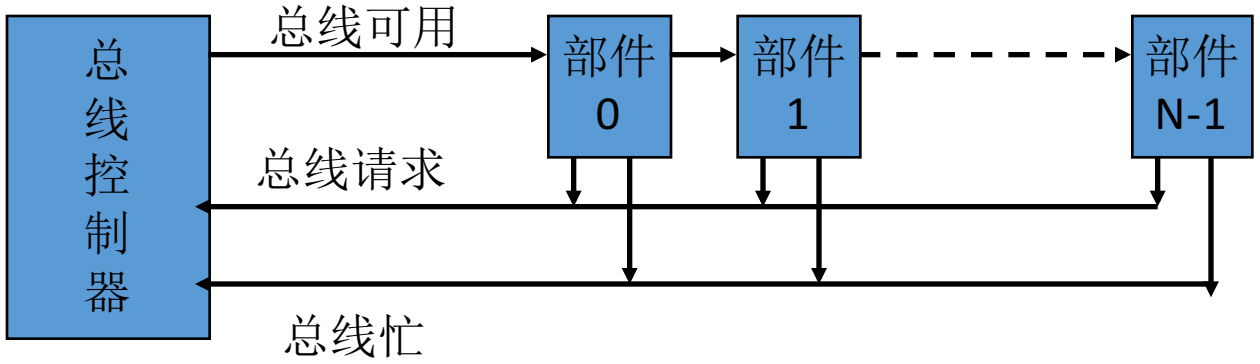
集中式的优先次序确定

- 串行连接
- 定时查询
- 独立请求

- 采用何种方式取决于控制线数目、总线分配速度、灵活性、可靠性等因素的综合权衡

集中式的串行链接

- 所有部件都经公共的“总线请求”线向总线控制器发出请求
- 当“总线忙”信号未建立，总线空闲，请求响应，送出“总线可用”信号，串行通过每个部件
- 如某个部件接收到“总线可用”信号但未发送请求，则送往下一个部件，如该部件发出过请求，则停止信号
- 建立“总线忙”，清除请求信号，准备数据传送
- 数据传送期间，“总线忙”维持“总线可用”的建立
- 传送完，去除“总线忙”和“总线可用”信号

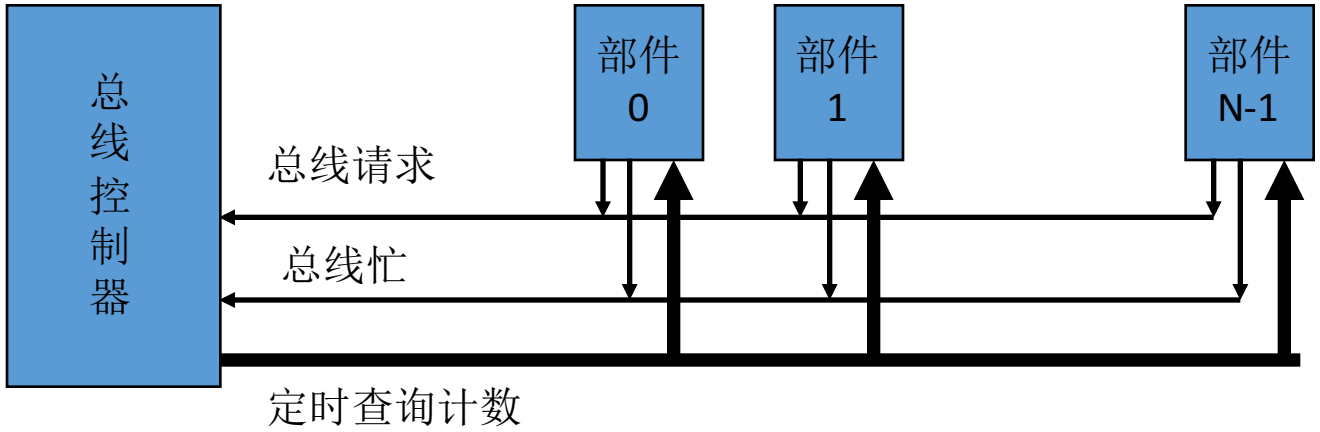


集中式的串行链接

- 优先次序是由“总线可用”线所接部件的物理位置决定的
- 优点：
  - 选择算法简单，控制总线少（3根）
  - 部件增减容易，可扩充性好
  - 逻辑简单容易通过重复设置提高可靠性
- 缺点：
  - 对“总线可用”线及其有关电路的失效很敏感
  - 优先级是线连固定，不能由程序改变，灵活性差增加
  - “总线可用”信号顺序脉动地通过各个部件，限制总线的分配速度
  - 受总线长度的限制，增减或移动设备受到限制

集中式定时查询

- 总线上的每个部件通过“总线请求”线发出请求
- 若总线处于空闲，“总线忙”未建立，总线控制器收到请求后，让计数器开始计数，定时查询各部件以确定是哪个部件发出请求
- 查询到请求发出部件，该部件建立“总线忙”，计数器停止计数，控制器停止查询
- 去除该部件的“总线请求”，让该部件获得总线使用权，传送完成后去除“总线忙”
- 如果总线分配前计数器清0，同串行链接；如果总线分配前不清0，则是循环优先级；如果总线分配前计数器设置某个初值，则指定这个部件为最高优先级；如果总线分配前将部件号重新设置一下，则可以指定各部件为任意所希望的优先级

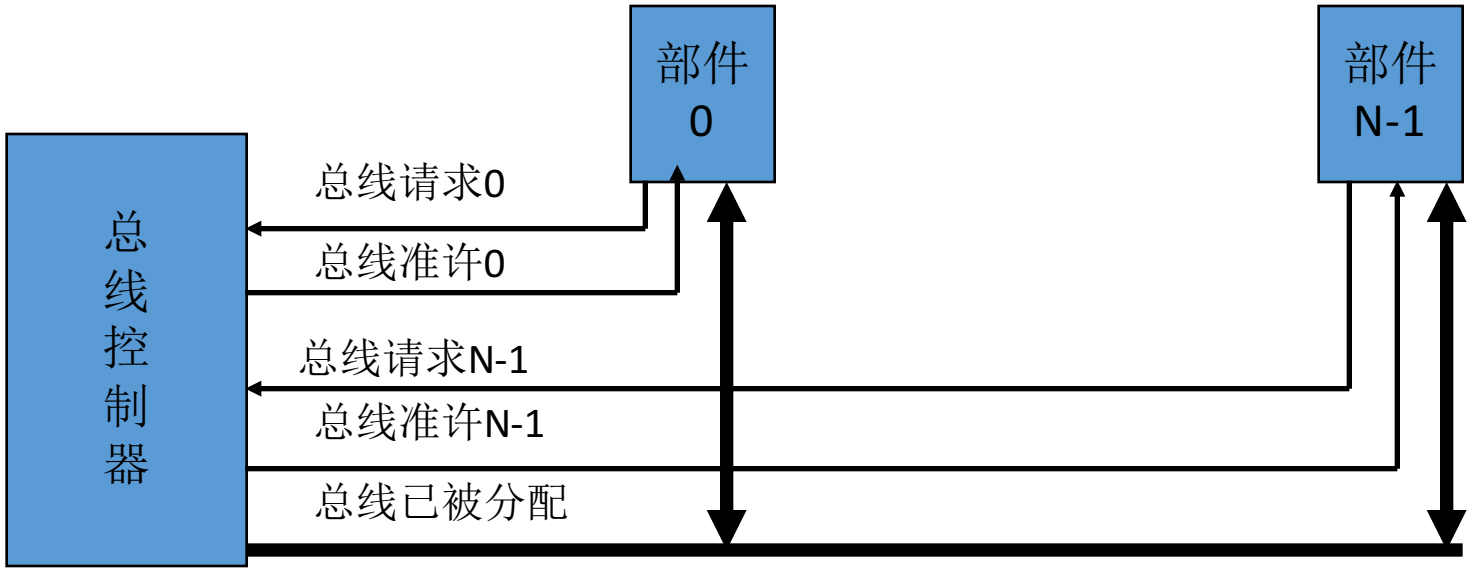


集中式定时查询

- 优点：
  - 因计数器初值，部件号均可由程序设定，优先次序可由程序控制，灵活性强
  - 可靠性高
- 缺点：
  - 控制线数较多，需$2+\log_2 N$,扩展性稍差
  - 控制较为复杂
  - 速度取决与计数信号的频率和部件数，不能很高

集中式独立请求

- 共享总线的每个部件各自都有一对“总线请求”和“总线准许”线
- 总线控制器根据某种算法对同时送来的多个请求进行仲裁



集中式独立请求

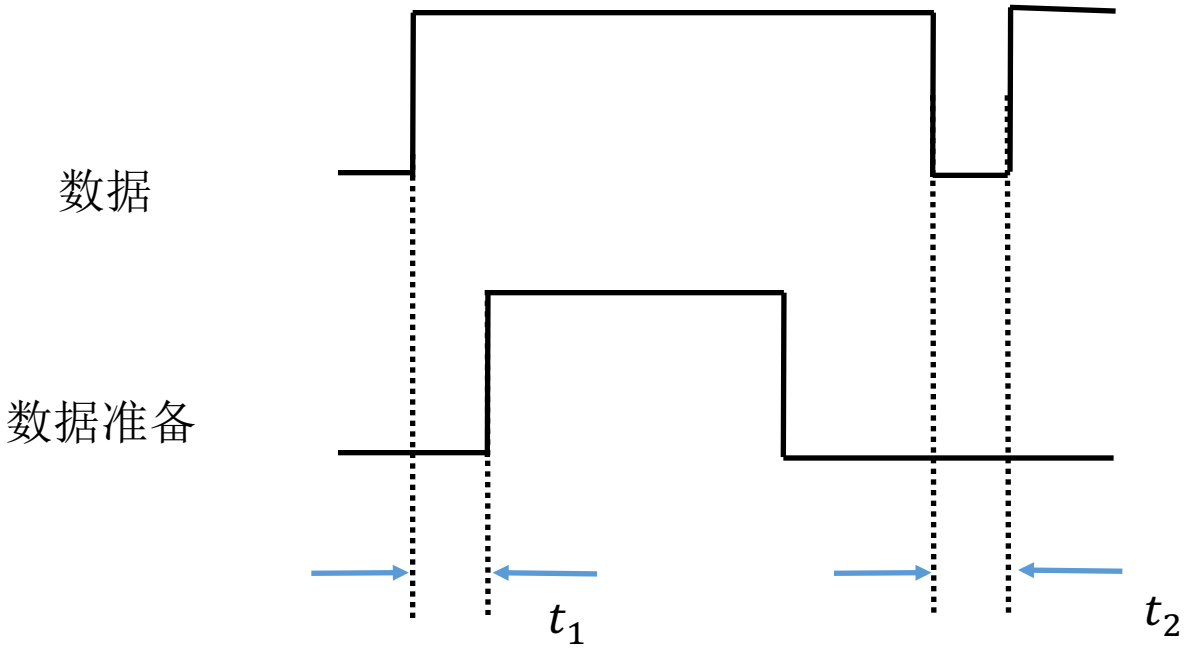
- 优点：
  - 总线分配速度快，不用查询
  - 可以使用程序可控的预订方式、自适应方式、循环方式或它们的混和方式灵活确定下一个使用总线的部件
  - 能方便地隔离失效部件的请求
- 缺点：
  - 控制线数量多，N个设备必须有2N+1根控制线
  - 总线控制器复杂

总线的通讯方式

- 同步通讯：两个部件的信息是通过定宽、定距的系统时标进行同步的。
  - 传送率高，受总线长度影响小
  - 时钟在总线上的时滞可能会造成同步误差，时钟线上的干扰信号容易引起误同步
  - 为提高可靠性，目的部件需要对数据是否正确接受予以回答：正常时不回答，出错时在同步时间片过去后向源部件发送出错信号，使之重发数据
- 异步通讯
  - 单向源（目）控制：通讯过程只由目的或源部件中的一个控制。
  - 双向控制：由源和目的双方控制。

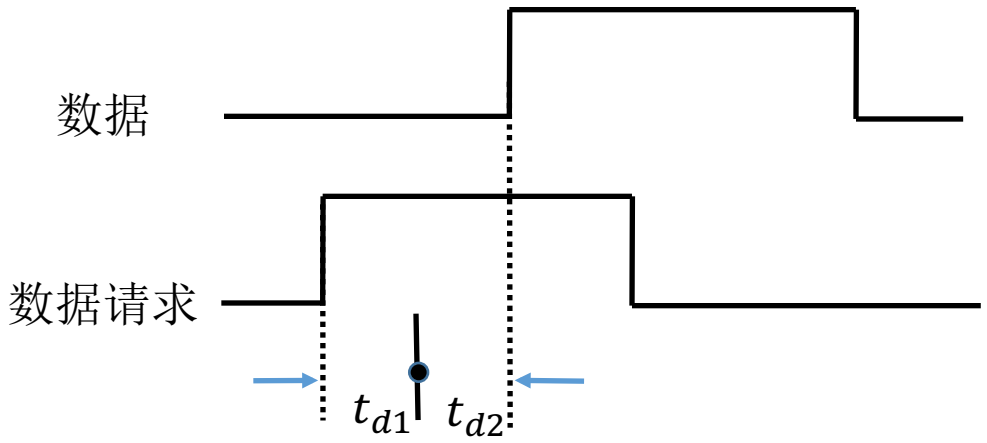
异步单向源控式通信

- 优点：简单、高速
- 缺点：没有来自目的部件的信息指明数据传送是否有效；不同速度之间的部件之间通信比较复杂；部件内需设置缓冲器以缓冲来不及处理的数据；效率低，高速部件难以发挥效能；要求“数据准备”信号干扰要小，否则易误认成有效信号



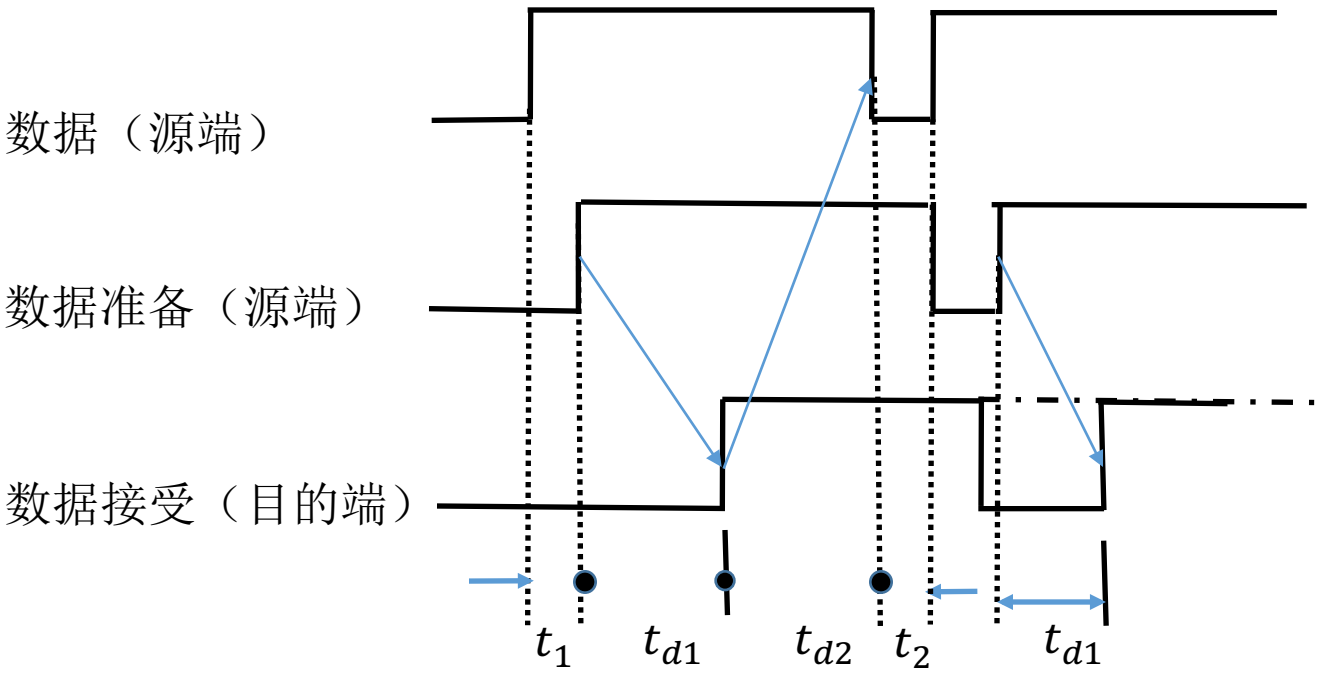
异步单向目控式通信

- 解决传送有效性检验的问题

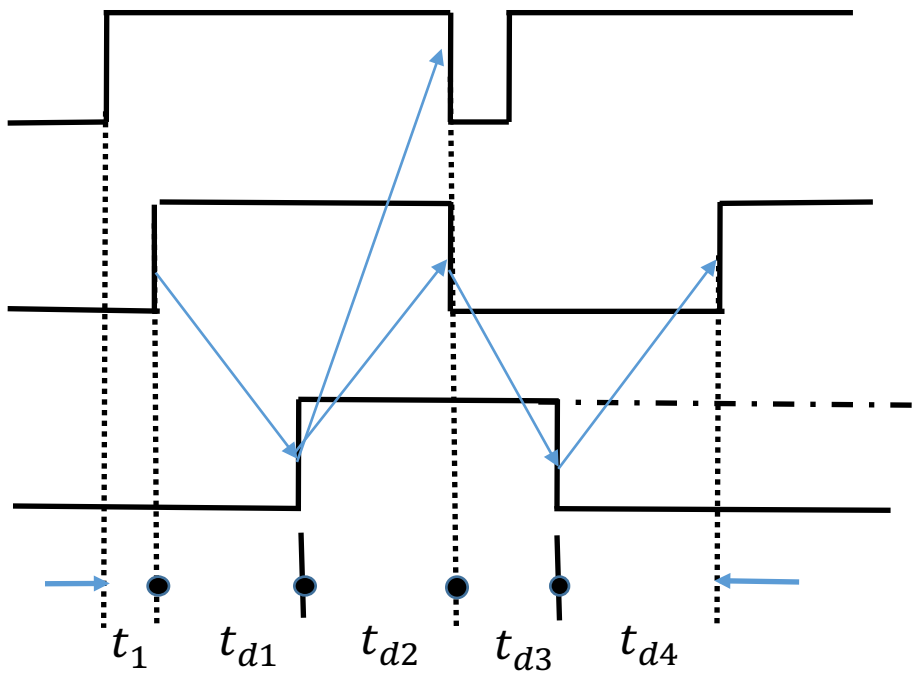


异步请求 / 回答式双向控制

- 单向控制的缺点是不能保证下一数据传送之前让所有数据线和控制线的电平信号恢复成初始状态，从而可能造成错误状态
- 为解决上述问题，可以采用异步请求 / 回答式双向控制



非互锁方式



互锁方式

- 异步双向互锁方式虽然增加了信号沿总线来回传送的次数，使控制硬件变得更加复杂
- 适应各种不同速度的I/O设备，保证数据传送的正确性，且有较高的数据传送速率（其速率为源部件和目的部件中相对较低的速率来通信，比同步方式总是以所有部件中最低的速率来通信的效率要高）

数据宽度

- 数据宽度是I/O设备取得I/O总线后所传送数据的总量，而数据通路宽度是数据总线的物理宽度（即一个时钟周期所传送的数据量）
- 一个数据宽度可能需要多个数据周期才能完成
- 数据宽度的类型
  - 单字（单字节）宽度适合输入机、打印机等低速设备
  - 定长块宽度适合于磁盘等高速设备
  - 可变长块宽度适合于高优先级的中高速磁带、磁盘等设备

总线的线数

- 总线线数越多，成本越高，干扰越大，可靠性越低，占用空间也越大，当然，传送速度和流量也越高
- 总线长度越长，成本越高，干扰越大，波形畸变越严重，可靠性越低
- 因此，总线增长，线数就应该越少
- 总线线数可以通过用线的组合、编码，以及并／串—串／并转换来实现，但一般会降低总线的流量

输入输出系统

- 输入输出系统包括输入输出设备、设备控制器及与输入输出操作有关的软硬件。
- 在计算机系统中，把处理机与主存储器之外的部分统称为输入输出系统
- 输入输出系统是处理机与外界进行数据交换的通道。
- 与处理机有关的、除人以外的各种设备称为输入输出设备（或外围设备）

输入输出系统的特点

- 异步性
  - 输入输出设备通常不使用统一的中央时钟，各个设备按照自己的时钟工作，但又要在某些时刻接受处理机的控制
  - 处理机与外围设备之间，外围设备与外围设备之间能够并行工作
- 实时性
  - 对于一般外部设备：可能丢失数据，或造成外围设备工作的错误
  - 对于实时控制计算机系统，如果处理机提供的服务不及时，可能造成巨大的损失
  - 对于处理机本身的硬件或软件错误：如电源故障、数据校验错、页面失效、非法指令、地址越界等，处理机须及时处理
  - 对不同类型的设备，必须具有与设备相配合的多种工作方式

输入输出系统的特点（续）

- 与设备无关性
  - 独立于具体设备的标准接口。例如，串行接口、并行接口、SCSI（Small Computer System Interface）接口等
  - 计算机系统的使用者，在需要更换外围设备时，各种不同型号，不同生产厂家的设备都可以直接通过标准接口与计算机系统连接
  - 处理机采用统一的硬件和软件对品种繁多的设备进行管理
  - 某些计算机系统已经实现了即插即用技术

输入输出系统的组织方式

- 针对异步性，采用自治控制的方法
  - 输入输出系统是一个独立于处理机之外的自治系统
  - 处理机与外围设备之间要有恰当的分工
- 针对实时性，采用层次结构的方法
  - 最靠近处理机的是输入输出处理机、输入输出通道等
  - 中间层是标准接口
  - 标准接口通过设备控制器与输入输出设备相连
  - 设备控制器控制外围设备工作
- 针对与设备无关性，采用分类处理方法
  - 为面向字符的设备（character-oriented device）；指工作速度比较低的机电类设备。例如，字符终端、打字机等
  - 面向数据块的设备主要指工作速度比较高的外围设备；例如，磁盘、磁带、光盘的辅助存储器，行式打印机等

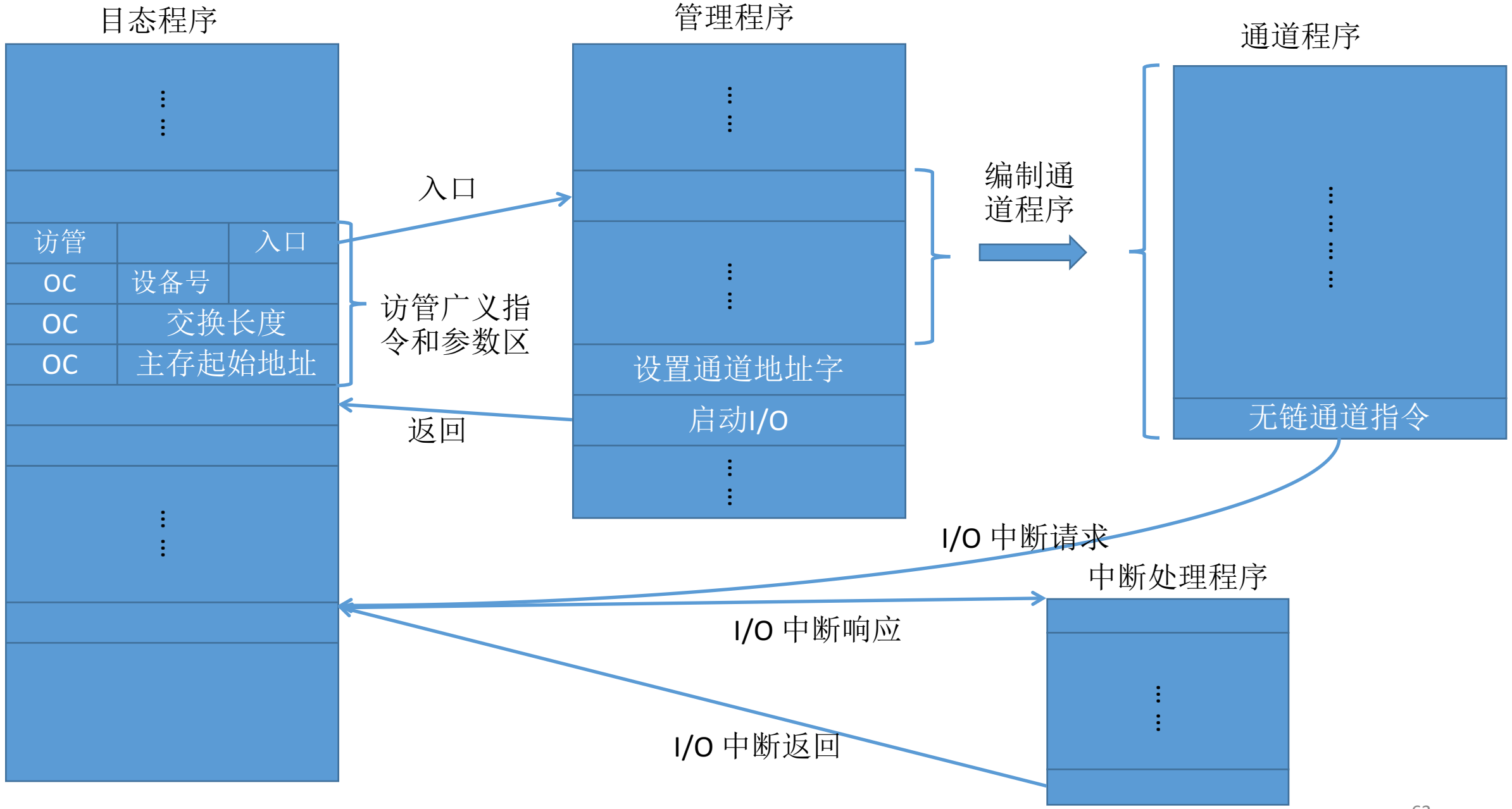
输入输出系统的发展阶段

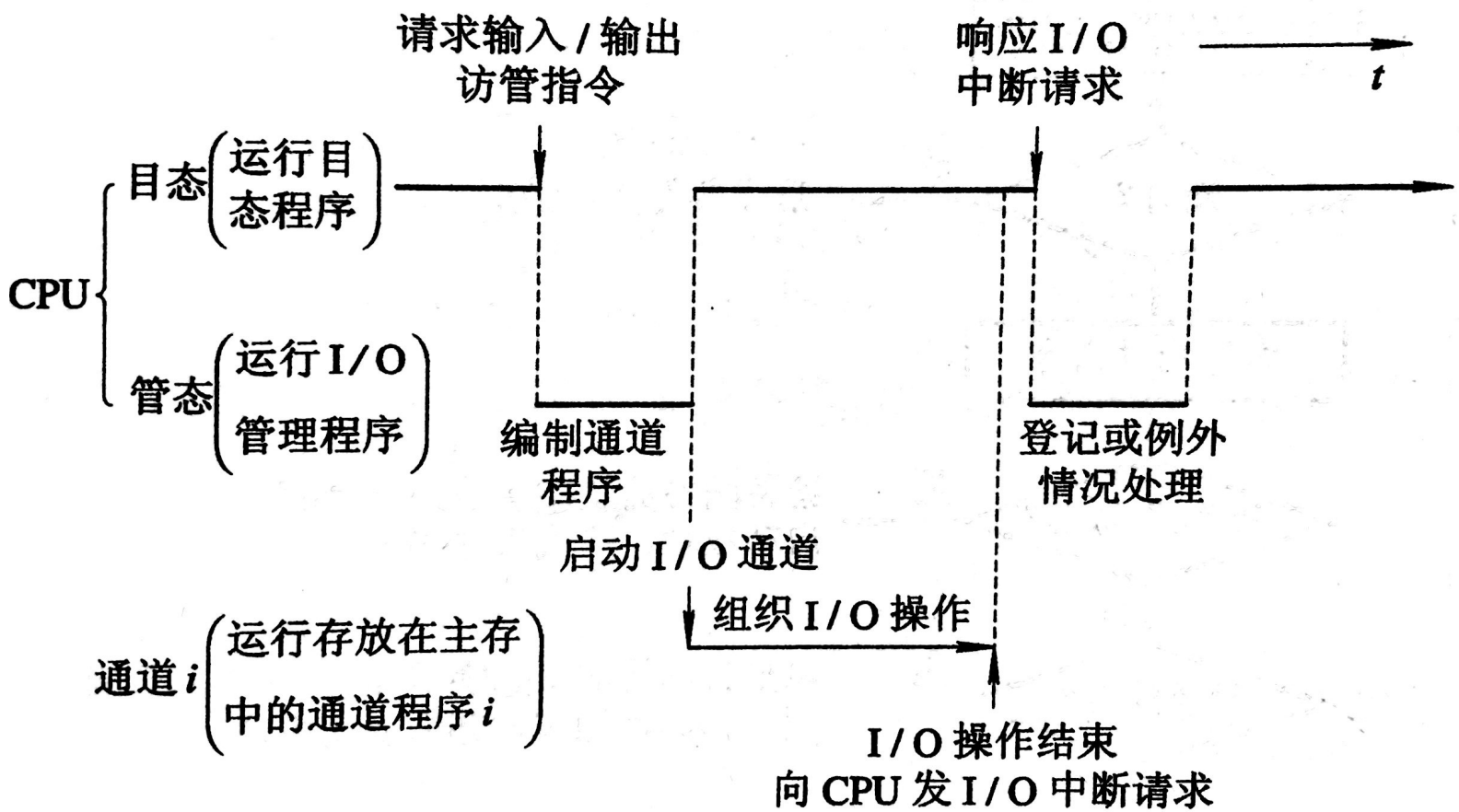
- 程序控制输入输出（包括全软的、程序查询状态驱动的、中断驱动的等）
- 直接存储器访问（DMA）
- I/O处理机
  - 通道方式（Channel）
  - 外围处理机方式（Peripheral Processor Unit， PPU）

- 通道也可以看做是“处理机”，有自己的指令系统（通道指令）和程序（通道程序）
- 每条通道指令为输入 / 输出规定一定的动作，对外设进行控制
- 通过链接标志将多条通道指令构成通道程序存放在主存对应通道的缓冲区中
- 通道通过执行通道程序来控制输入 / 输出，它可以与CPU并行工作
- 通道能代替CPU对多个设备的信息传输进行分时管理，在主存和外设信息交换过程中实现字与字节之间的装配和拆卸
- 通道还能向CPU报告设备和设备控制器的状态，处理输入 / 输出系统可能出现的情况
- 通道并不是严格意义上的“处理机”

通道处理机的工作原理

- 用户一般不能直接安排输入 / 输出操作，而通过程序状态的切换来实现输入输出，以保证系统的可靠性
- 中央处理机用来控制外部设备操作用的输入 / 输出指令称为管态指令
- 用户在目态程序中不能直接使用管态指令，只能使用要求输入 / 输出的广义指令，然后进入相应的管理程序执行输入 / 输出的管态指令。


目态程序
管理程序
通道程序
访管
入口
OC
设备号
OC
交换长度
OC
主存起始地址
入口
访管广义指令和参数区
返回
设置通道地址字
启动I/O
编制通道程序
无链通道指令
I/O 中断请求
中断处理程序
I/O 中断响应
I/O 中断返回


请求输入/输出
访管指令
响应I/O
中断请求
t
CPU
目态
运行目
态程序
管态
运行I/O
管理程序
编制通道
程序
登记或例外
情况处理
启动I/O通道
组织I/O操作
通道i
运行存放在主存
中的通道程序i
I/O操作结束
向CPU发I/O中断请求

启动I/O指令

- 选择通道
- 选取子通道
- 选取通道指令
- 选取控制器、设备
- 向设备发送启动命令
- 启动成功，则结束通道开始选择设备期，进入通道程序，并退出管态程序，返回目态程序

通道数据传送方式

- 字节多路：适合于连接像光电机的字符类低速设备
  - 传送一个字符（字节）的时间很短，但字符（字节）间的等待时间很长
  - 以字节交叉方式轮流为多台低速设备服务
  - 可有多个子通道，独立执行通道命令，并行操作
- 数组多路：适合于连接多台像磁盘的高速设备
  - 传送速率很高，但传送开始前的寻址辅助操作时间很长
  - 成组交叉方式，传送完K个字节数据后重新选择下一个设备
- 选择通道：适合于连接优先级高的磁盘等高速设备
  - 独占通道
  - 在数据传送期内只选择一次设备

通道流量的设计

- 通道流量是通道在数据传送期间内，单位时间传送的字节数
- 能达到的最大 流量称为通道的极限流量
  - 工作方式
  - 数据传送期间内选择一次设备的时间T_s
  - 数据传送期间内传送一个字节的时间T_d

字节多路的通道极限流量

每选择一台设备只传送一个字节

$$f_{max \cdot byte} = \frac{1}{T_S + T_D}$$

数组多路的通道极限流量

- 每选择一台设备就能传送完K个字节
- 如果要传送N个字节，就得分N/K次传送才行，
- 每次传送就要选一次设备

$$f_{max\cdot block} = \frac{K}{T_S + KT_D} = \frac{1}{\frac{T_S}{K} + T_D}$$

选择通道的通道极限流量

- 每选择一台设备就把N个字节全部传送完

$$f_{max\cdot select} = \frac{N}{T_S + NT_D} = \frac{1}{\frac{T_S}{N} + T_D}$$

- 如通道的T_S、T_D确定一定时，且$N > K$时

$$f_{max\cdot select} \geq f_{max\cdot block} \geq f_{max\cdot byte}$$

- 设备要求通道的**实际最大流量**与三种通道的工作方式有关
- 字节多路是该通道连接各设备的字节传送速率之和

$$f_{byte\cdot j}=\sum_{i=1}^{p_j} f_{i\cdot j}$$

- 数组多路和选择方式是所连接各设备的字节传送速率中之最大者

$$f_{block\cdot j}=\max_{i=1,\cdots,p_j} f_{i\cdot j}\,,\qquad f_{select\cdot j}=\max_{i=1,\cdots,p_j} f_{i\cdot j}$$

j：通道编号

$f_{i\cdot j}$：为第j号通道上所挂的第i台设备的字节传送速率

p_j：为第j号通道上所连接的设备的台数

- 为了保证第 j 号通道上所挂设备在满负荷的最坏情况下都不丢失信息，必须使设备要求通道的实际最大流量不超过通道的极限流量

$$f_{byte\cdot j} \le f_{max\cdot byte\cdot j}$$
$$f_{block\cdot j} \le f_{max\cdot block\cdot j}$$
$$f_{select\cdot j} \le f_{max\cdot select\cdot j}$$

- 如果I/O系统有m个通道，其中1至m_1为字节多路，m_1+1至m_2为数组多路，m_2+1至m为选择，则I/O系统的极限流量为：

$$f_{max} = \sum_{j=1}^{m_1} f_{max\cdot byte\cdot j} + \sum_{j=m_1+1}^{m_2} f_{max\cdot block\cdot j} + \sum_{j=m_2+1}^{m_3} f_{max\cdot select\cdot j}$$

$$f_{max} \ge \sum_{j=1}^{m_1}\sum_{i=1}^{p_j} f_{i\cdot j} + \sum_{j=m_1+1}^{m_2} \max_{i=1,\cdots,p_j} f_{i\cdot j} + \sum_{j=m_2+1}^{m_3} \max_{i=1,\cdots,p_j} f_{i\cdot j}$$

举例

- 假设通道在数据传送期，选择设备需要$9.8\mu s$，传送一个字节需要$0.2\mu s$。某低速设备每隔$500\mu s$发出一个字节数据请求，那么最多可以接几台这种设备？

由于低速设备周期性发出一个字节的数据请求，可以看出挂载低速设备的通道是按字节多路通道方式工作的，其极限流量是

$$f_{max\cdot byte} = \frac{1}{T_s + T_D}$$

其实际流量需要满足

$$f_{max\cdot byte} \geq f_{byte}$$

设挂载的设备数量是m，每台设备的速率$f_i = 1/500\mu s$。应满足

$$\frac{1}{T_s + T_D} \geq m f_i$$

因此

$$m \leq \frac{1}{(T_S + T_D) f_i} = \frac{500\mu s}{(9.8 + 0.2)\mu s} = 50$$

- 对于以下6种高速设备，采用选择通道，假设通道一次通信传送的字节数为1024个字节，则哪些设备可以挂，哪些不能。

设备	A	B	C	D	E	F
发送请求时间间隔（μs）	0.2	0.25	0.5	0.19	0.4	0.21

- 一次传输字节数n为2014个，使用选择通道，其极限流量为

$$f_{max\cdot select} = \frac{n}{T_S + nT_D} = \frac{1}{\frac{9.8}{n} + 0.2}$$

所以需要满足

$$f_i \leq \frac{n}{T_S + nT_D} = \frac{1}{\frac{9.8}{n} + 0.2}$$

- 设备的速率

设备	A	B	C	D	E	F
发送请求时间间隔（μs）	1/0.2	1/0.25	1/0.5	1/0.19	1/0.4	1/0.21

只能挂B, C, E, F四台设备

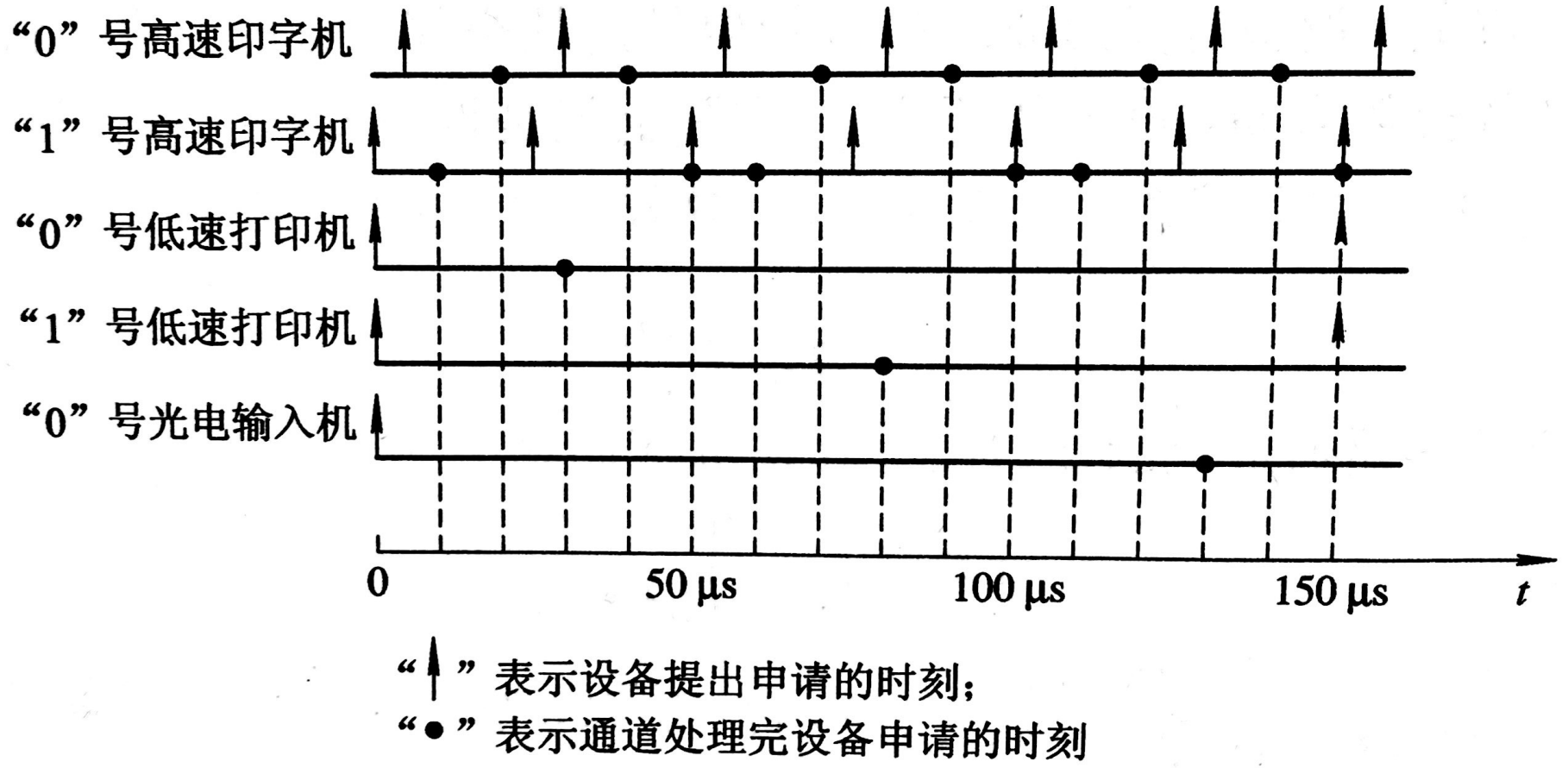
- 假设有一个字节多路通道，有3个子通道：“0”号、“1”号高速印字机各占一个通道；“0”号打印机、“1”号打印机和“0”号光电输入机合用一个子通道。假定数据传送期内高速印字机每隔25us发送一个字节请求，低速打印机每隔150us发送一个字节请求，光电输入机每隔800us发送一个字节请求，则这5台设备要求的通道流量为

$$f_{byte\cdot j}=\sum_{i=1}^{5}f_{i\cdot j}=\frac{1}{25}+\frac{1}{25}+\left(\frac{1}{150}+\frac{1}{150}+\frac{1}{800}\right)\approx 0.095\ MB/s$$

通道的极限流量可以设计成$0.1\ MB/s$，即所设计的通道工作周期为

$$T_D+T_S=10\ \mu s$$

- 设备优先级：“0”号高速印字机→“1”号高速印字机→“0”号低速打印机→“1”号低速打印机→“0”号光电输入机



通道的流量设计

- 上述流量设计的基本条件只保证了宏观上设备不丢失数据，并不能保证每一个局部时刻都不丢失信息
- 通常高速设备请求的响应优先级比较高，高速设备频繁发出请求并总是优先得到响应和处理，使得速率较低的设备长期得不到通道而丢失信息。
- 在设备或设备控制器中设置一定容量的缓冲器来缓冲得不到及时处理的数据，或是通过动态提高低速设备的响应优先级

外围处理机

- 通道处理机指令简单，只具有面向外设控制和数据传送的功能，无大容量存储器。因此，在输入和输出过程中还需要CPU承担大量工作
- 支持CPU高速运行所采用的流水技术，由于输入 / 输出过程中的中断，速度下降严重

外围处理机

- 发展外围处理机，让CPU进一步摆脱对输入 / 输出的控制，充分利用其高速性能
- 外围处理机很接近于一般处理机（甚至通用机），独立于主处理机异步工作，具有丰富的指令和功能，可以选择与主处理机共享主存
- 采用外围处理机，可以自由选择通道与设备进行通讯。
- 主存，PPU，通道和设备控制器相互独立，可以动态控制他们之间的连接，具有更强的灵活性