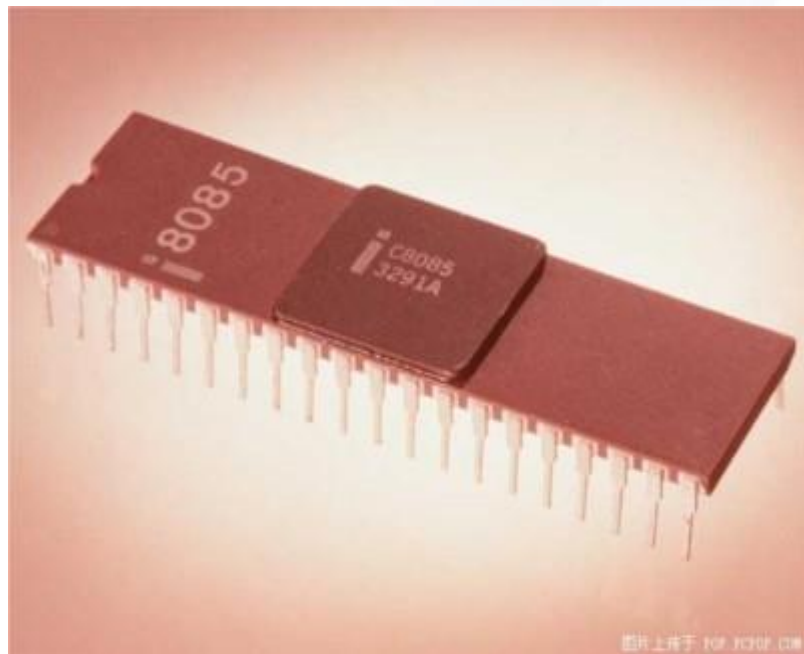
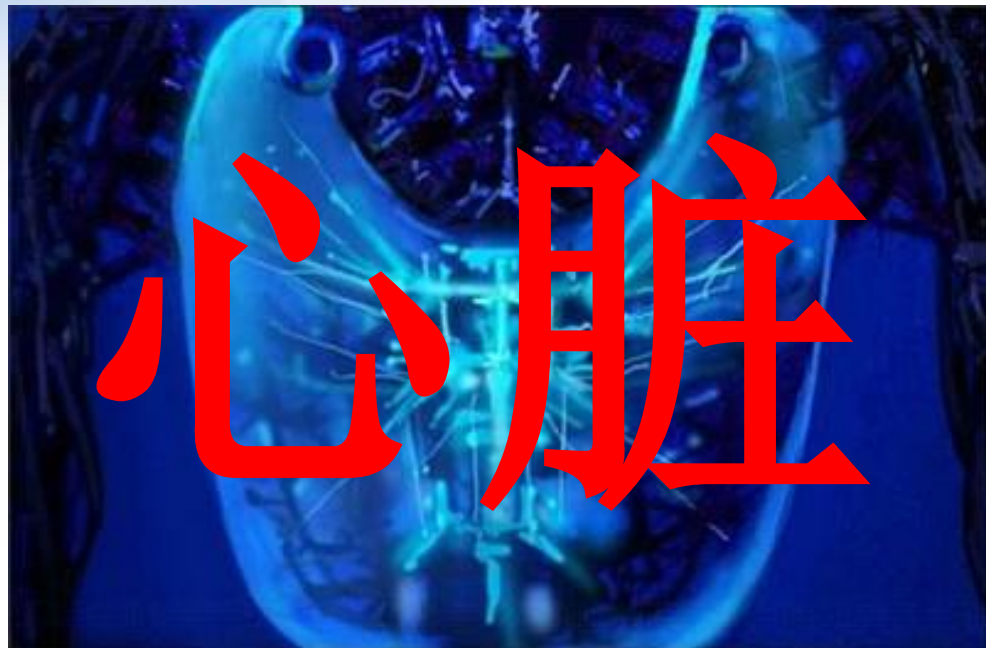


第8章---CPU的结构和功能

- § 8.1 CPU的结构
- § 8.2 指令周期
- § 8.3 指令流水
- § 8.4 中断系统

- **§ 8.1 CPU的结构**
- **§ 8.2 指令周期**
- **§ 8.3 指令流水**
- **§ 8.4 中断系统**

第8章---CPU的结构和功能



图片来源于 POF 网

结论： CPU作为一台电脑中的核心，它的作用是无法替代的。

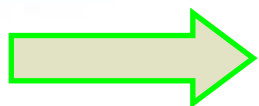
实质： 而CPU本身只是在块硅晶片上所集成的超大规模的集成电路，集成的晶体管数量可达到上亿个，是由非常先进复杂的制造工艺制造出来的，拥有相当高的科技含量。

第8章 CPU的结构和功能——8.1 CPU的结构

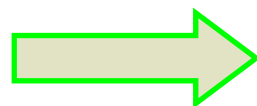
一、CPU 的功能

1.指令控制

取指令



分析指令



执行指令（发出各种操作命令）

程序是一个指令序列，这些指令的相互顺序不能任意颠倒，必须严格按照程序规定的顺序执行，故保证机器按照顺序执行程序时**CPU**的首要任务。

2.操作控制

CPU分析指令

产生



操作信号

控制



相应部件

3.时间控制

对各种操作顺序及每一步操作顺序都实施严格的控制。

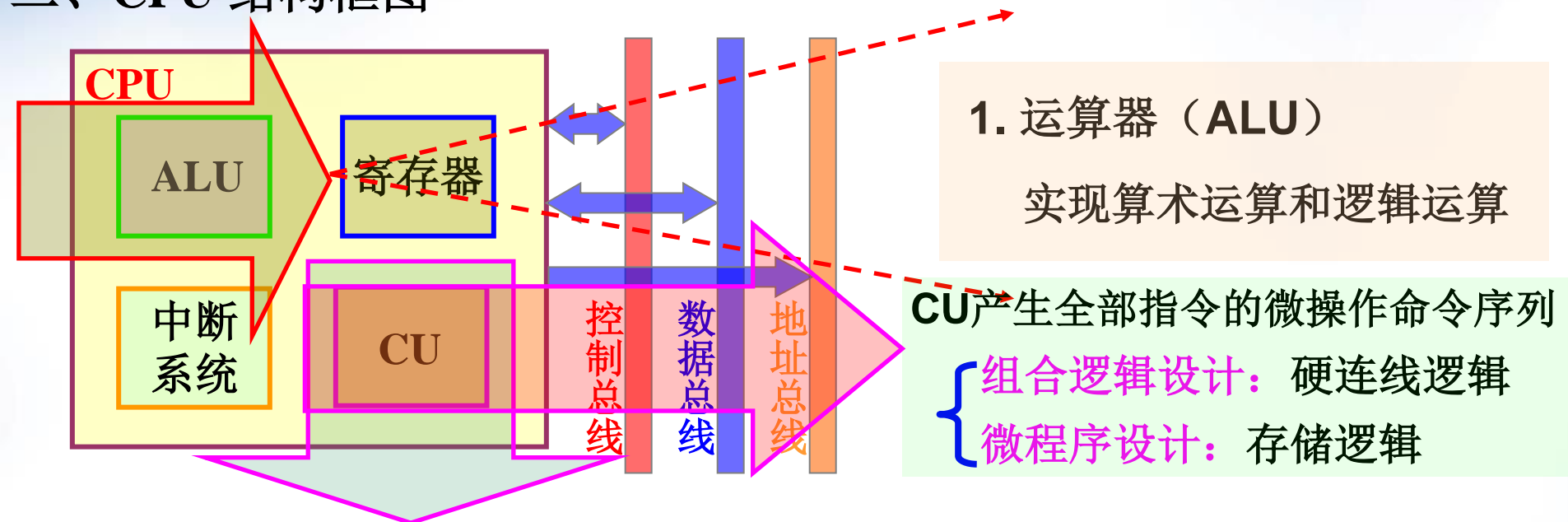
4.数据加工

完成数据的加工处理是**CPU**的根本任务。

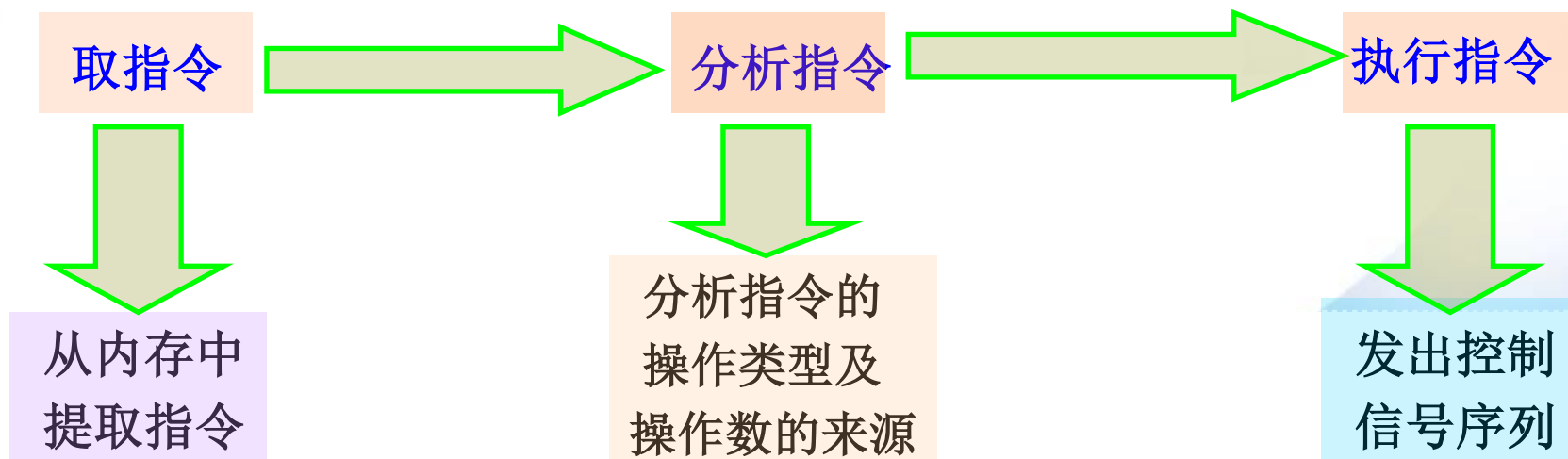
5.中断处理

第8章 CPU的结构和功能——8.1 CPU的结构

二、CPU 结构框图

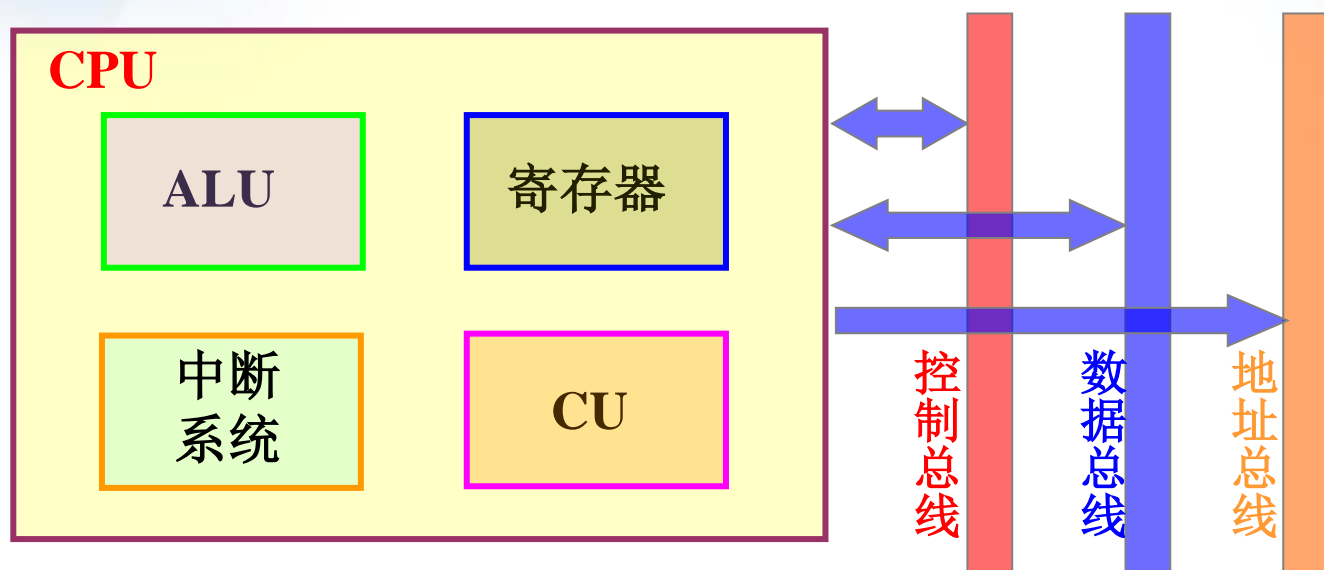


2. 控制器 (CU)(控制和协调各部件工作，具体功能如下)



第8章 CPU的结构和功能——8.1 CPU的结构

二、CPU 结构框图



3. 中断系统（用于处理种类中断）

常识： 中断的作用是为了响应和处理外部设备请求或异常事件。

功能

在CPU内部设置中断系统用以解决各种中断的共性问题；

处理与中断相关的

中断判优

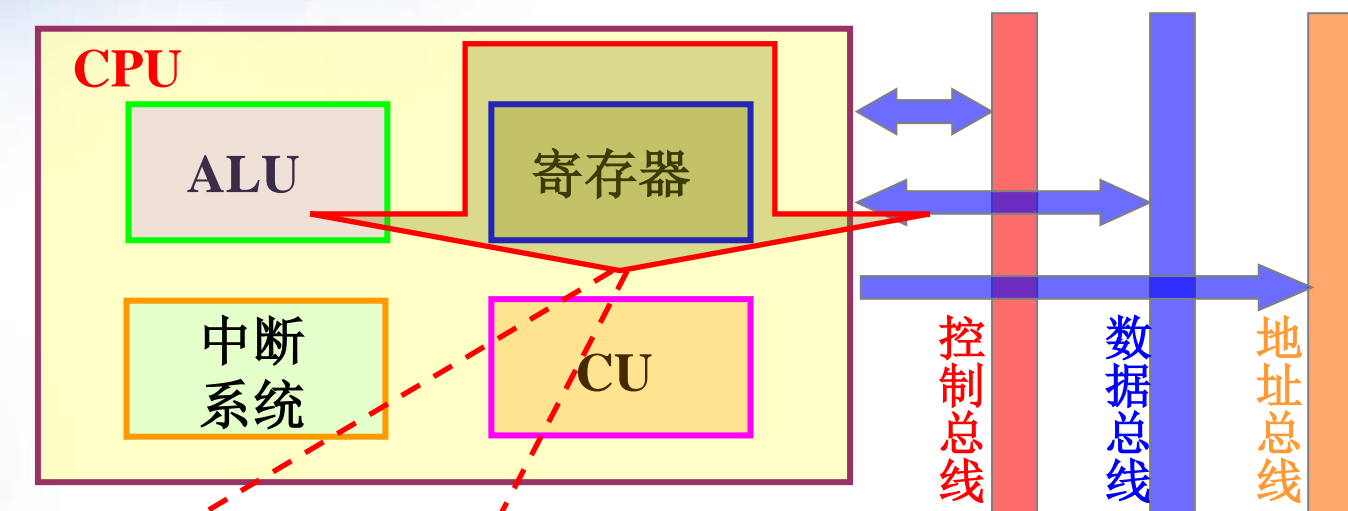
中断屏蔽

中断转换

等相关功能

第8章 CPU的结构和功能——8.1 CPU的结构

二、CPU 结构框图



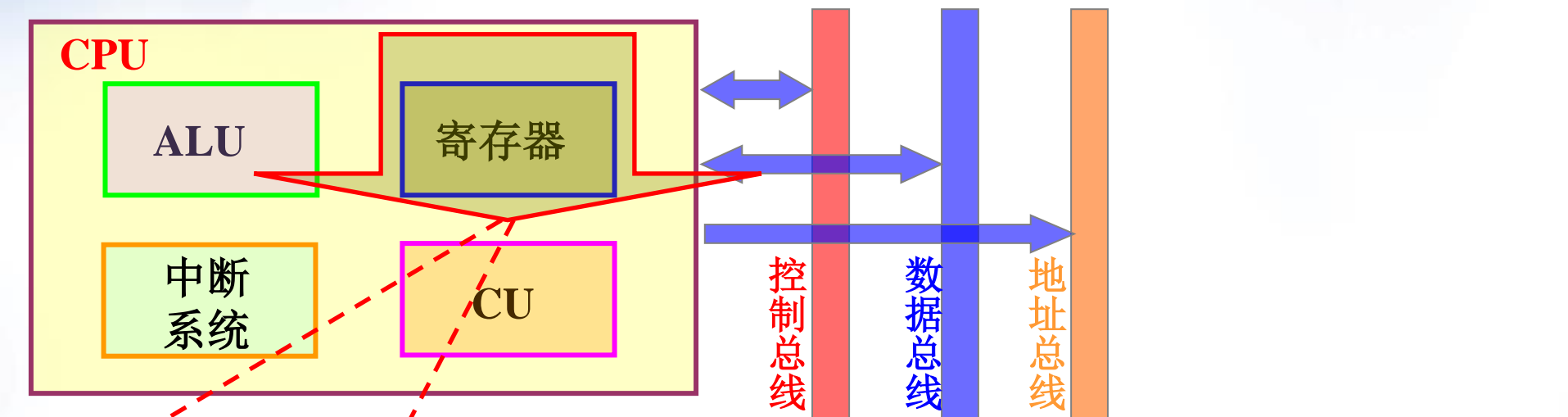
4.CPU 的寄存器

(1) 用户可见寄存器

- (1) 通用寄存器：存放操作数，可作某种寻址方式所需的专用寄存器。
- (2) 数据寄存器：存放操作数（满足各种数据类型）
- (3) 地址寄存器：存放地址，其位数应满足最大的地址范围。
用于特殊的寻址方式 段基值 栈指针
- (4) 条件码寄存器：存放条件码，可作程序分支的依据

第8章 CPU的结构和功能——8.1 CPU的结构

二、CPU 结构框图



4.CPU 的寄存器

(2) 控制和状态寄存器

① 控制寄存器 (CPU工作时所需的寄存器)
(控制 CPU 操作)

$PC \Rightarrow MAR \Rightarrow M \Rightarrow MDR \Rightarrow IR$

其中 { MAR MDR IR 用户不可见
 PC 用户可见

② 状态寄存器 (反映程序的运行状态)

{ 状态寄存器: 存放条件码

{ PSW 寄存器: 存放程序状态字

第8章 CPU的结构和功能 —— 8.2指令周期

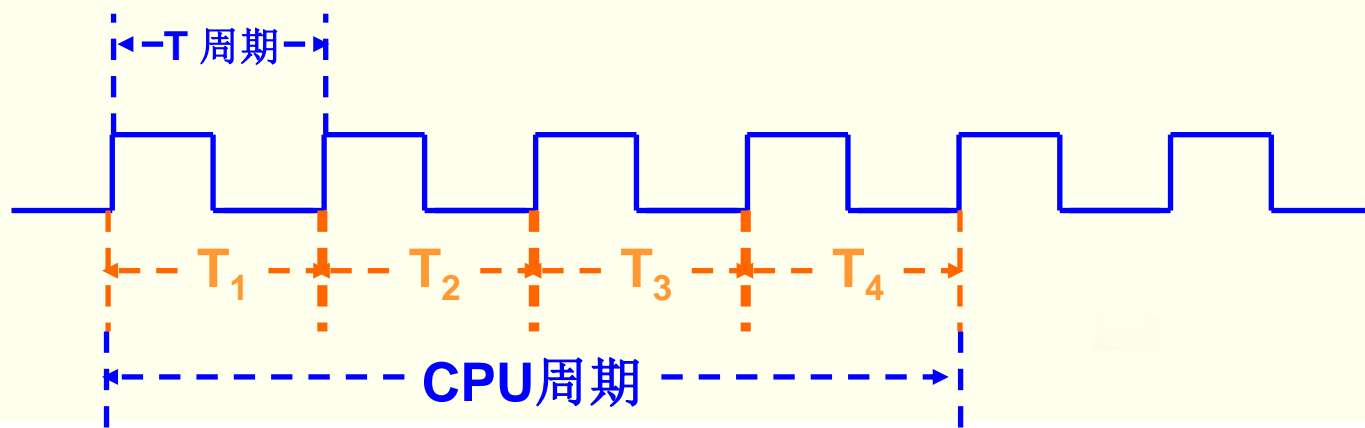
一、指令周期的基本概念

1. CPU周期: CPU从内存中读取一个指令字的最短时间。

机器周期

即一条指令的取出阶段被定义为一个CPU周期时间。

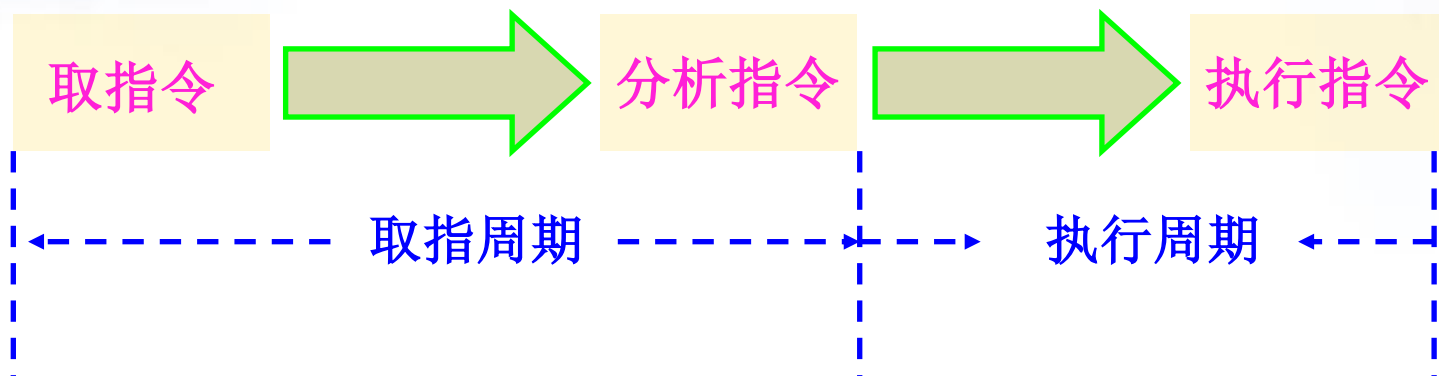
2. 时钟周期: CPU处理操作的最基本单位。一个CPU周期由若干个时钟周期构成。又被叫做 T 周期或节拍脉冲。



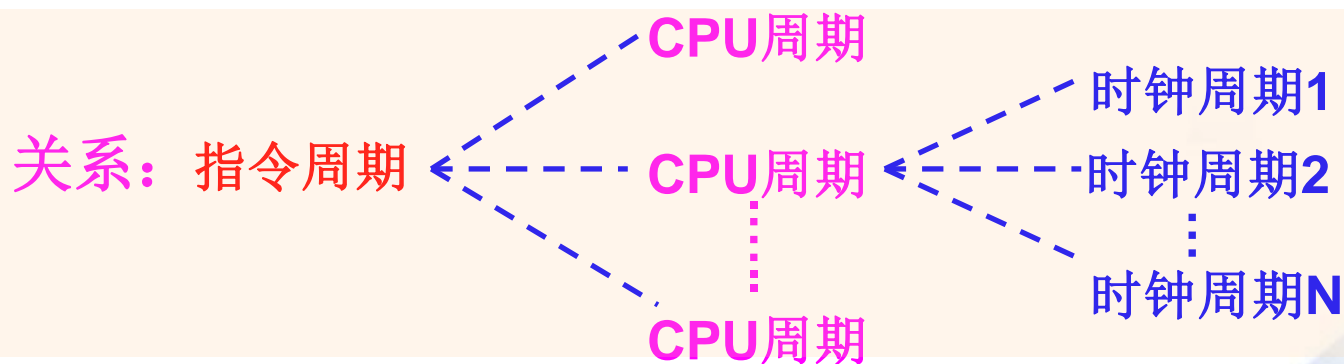
3. 指令周期: CPU取出并执行一条指令所需的全部时间。
包括取指令周期和执行指令周期。

第8章 CPU的结构和功能 —— 8.2指令周期

一、指令周期的基本概念



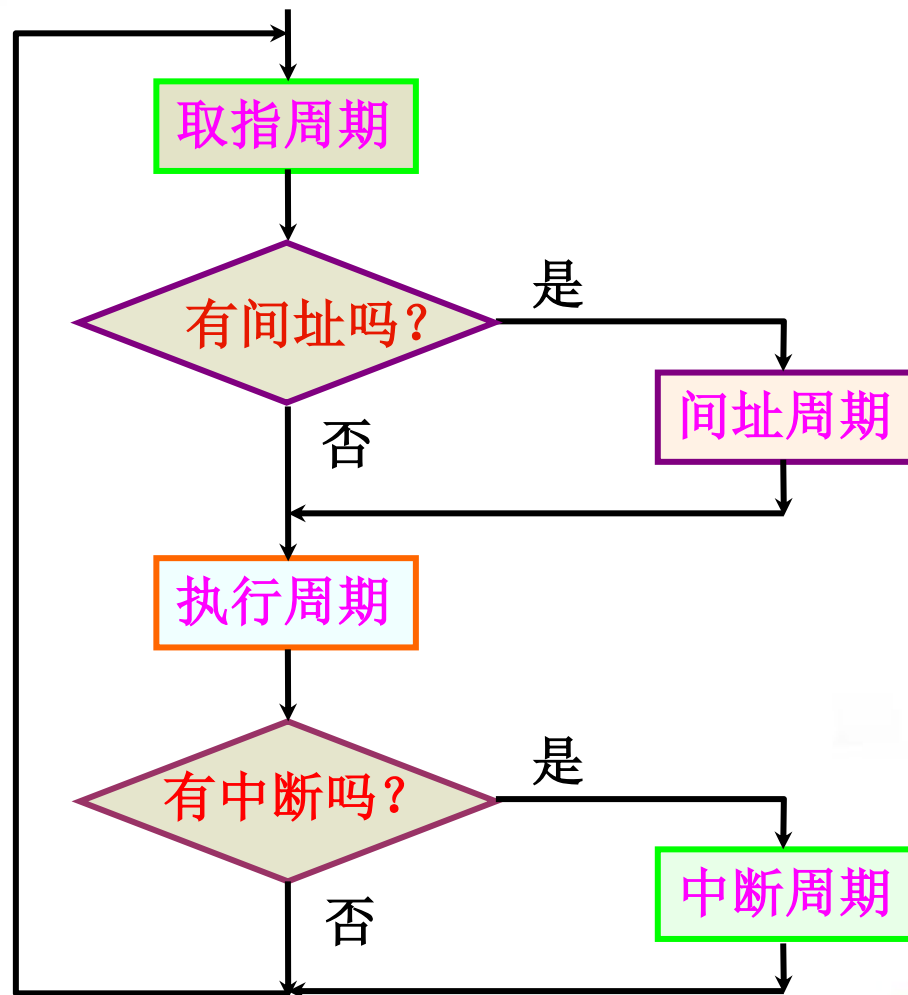
注：{ (1) 指令周期常用若干个**CPU**周期来表示。
(2) 不同的指令其指令周期由于功能不同，所以是不相同的。



第8章 CPU的结构和功能 —— 8.2指令周期

一、指令周期的基本概念

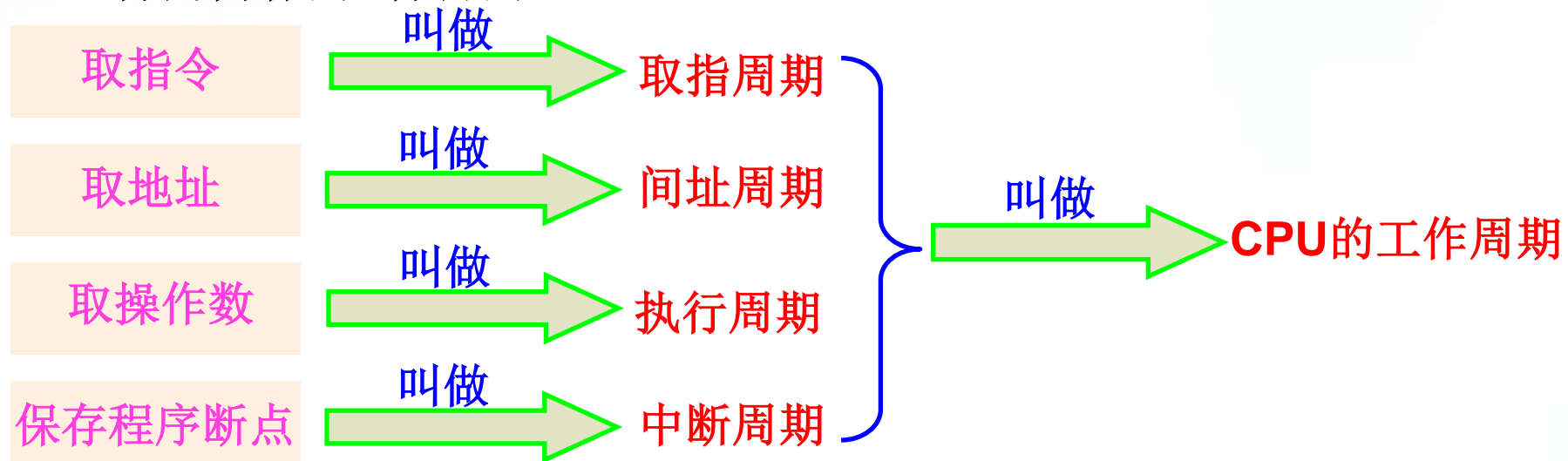
5. 指令周期流程



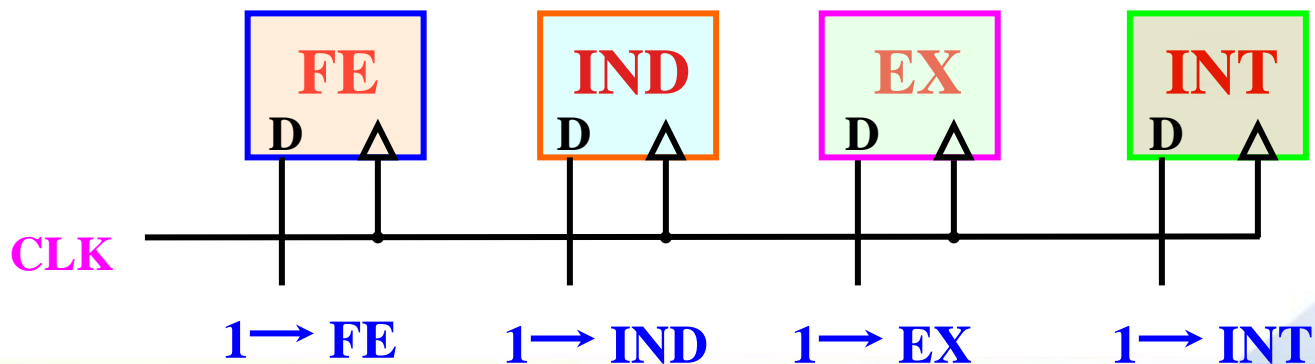
第8章 CPU的结构和功能 —— 8.2指令周期

6. CPU 访存类型

(1) CPU访问内存的4种目的

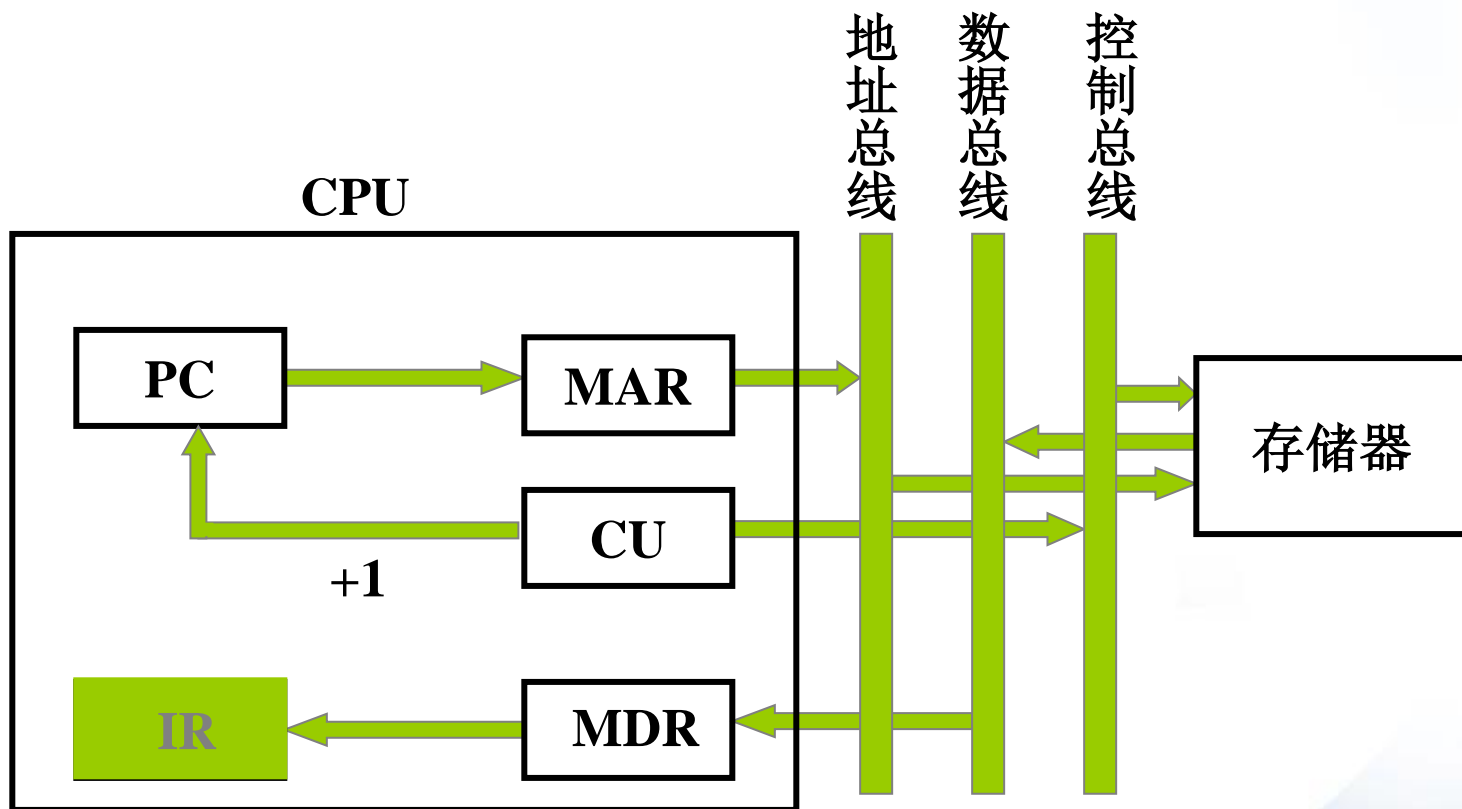


(2) CPU内的4个标志触发器



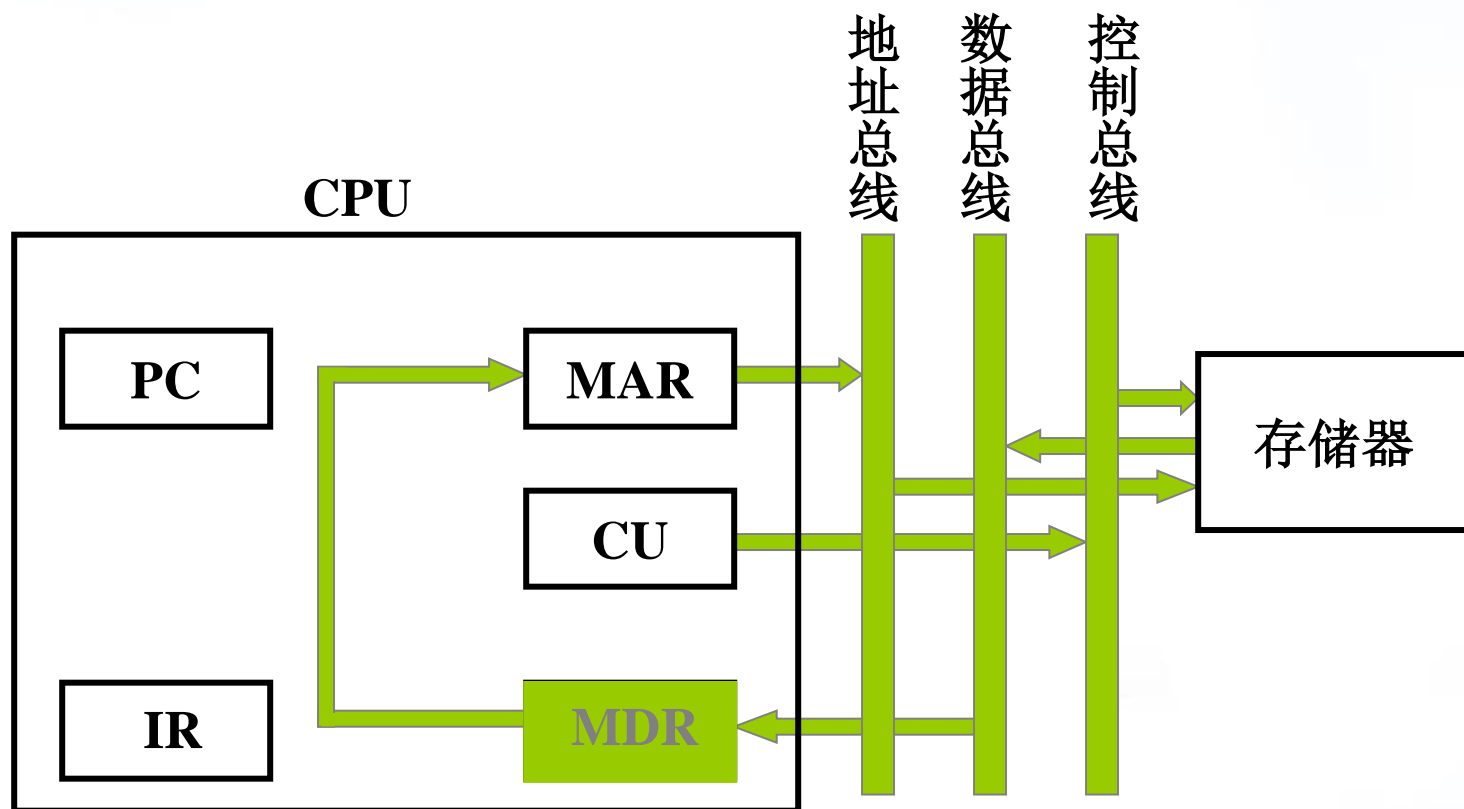
二、指令周期的数据流

1. 取指周期数据流



第8章 CPU的结构和功能 —— 8.2指令周期

2. 间址周期数据流

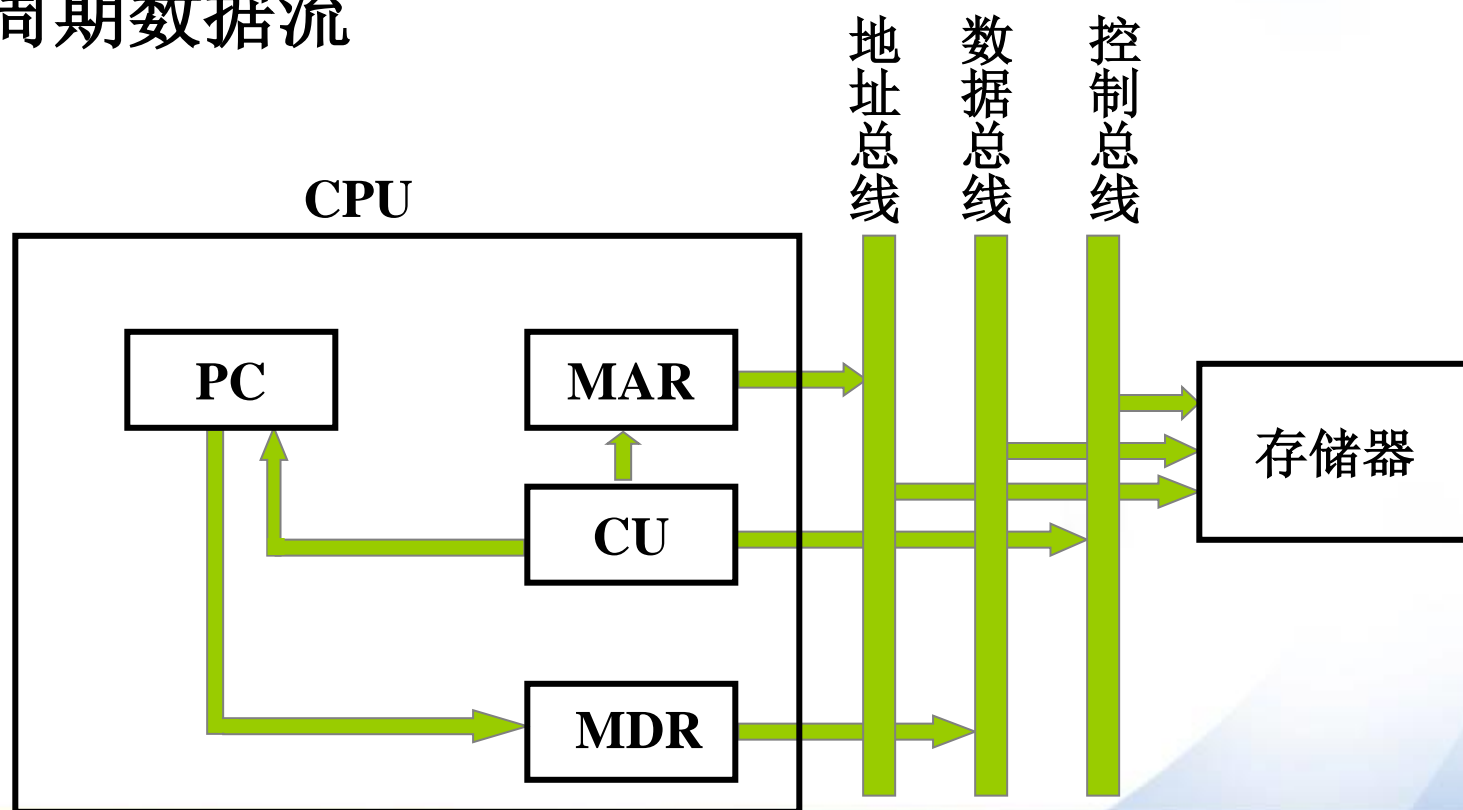


第8章 CPU的结构和功能 —— 8.2指令周期

3. 执行周期数据流

不同指令的执行周期数据流不同

4. 中断周期数据流



1. 设CPU 内有下列部件:PC、IR、SP、AC、MAR、MDR 和CU，要求:

- (1) 写出完成间接寻址的取数指令LDA @ x (将主存某单元的内容取至AC 中) 的信息流。
- (2) 画出中断周期的信息流，并简要说明。

(1) 完成间接寻址的取数指令包括取指、间址和执行三个阶段。

取指阶段的信息流：

PC→MAR→地址线

CU 发出读存储器命令

M→数据线→MDR→IR,至此指令读至 IR

OP(IR)→CU,指令操作码送 CU 分析

(PC)+1→PC,形成下一条指令地址

间址阶段的信息流：

MDR(或 IR)的地址码字段→MAR→地址线

CU 发出读存储器命令

M→数据线→MDR,至此有效地址读至 MDR

执行阶段的信息流：

MDR→MAR→地址线

CU 发出读存储器命令

M→数据线→MDR→AC,至此数据读至 AC 中

(2) 中断周期的信息流

在中断周期内需将程序断点(在 PC 中)保存起来,通常把断点存入堆栈。假设进栈操作是先修改堆栈指针,后存入数据,则中断周期的信息流如图 8.21 所示。具体可描述为:

CU 控制 $(SP) - 1 \rightarrow SP \rightarrow MAR \rightarrow$ 地址线

CU 发出写存储器命令

$PC \rightarrow MDR \rightarrow$ 数据线 \rightarrow 存储器

CU 将向量地址(硬件向量法)或中断识别程序入口地址(软件查询法) $\rightarrow PC$

第8章 CPU的结构和功能 —— 8.3指令流水

1、指令流水原理

(1) 顺序执行



精髓：当前指令的执行与后一条指令的取指在时间上重叠

(2) 二级流水

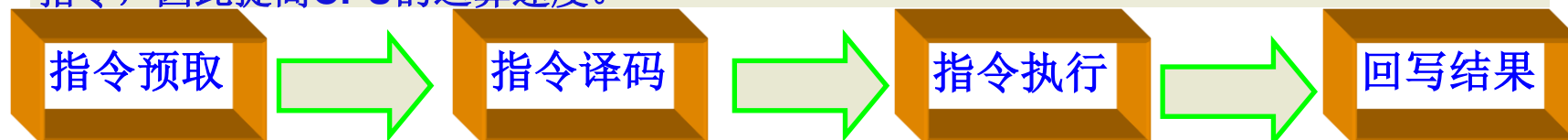


(3) 多级流水

将执行指令分为若干步，每步操作由不同部完成，即工序细化，形成多级流水。
(流水线是Intel首次在486芯片中开始使用的。)

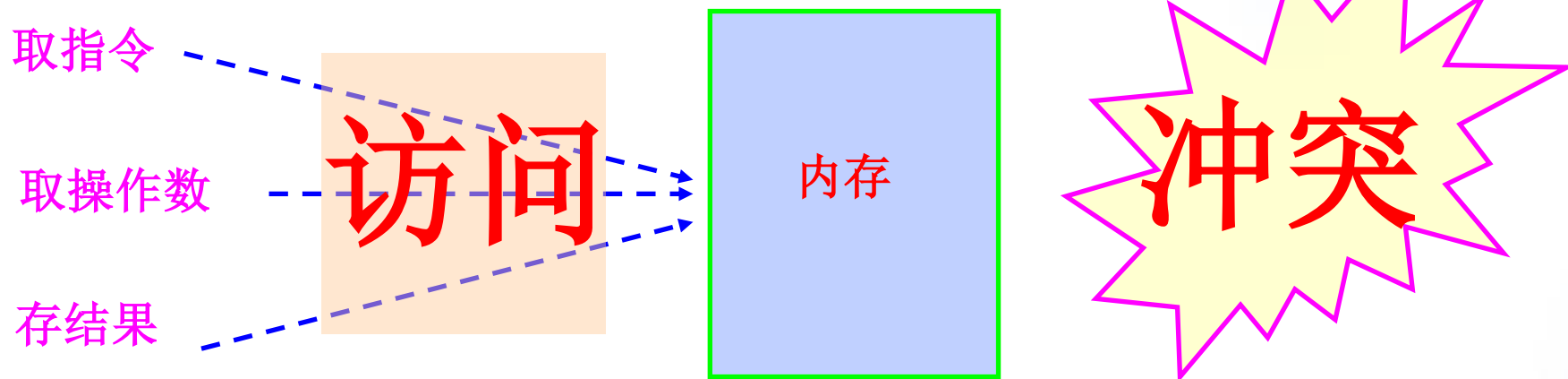
级数越多
速度越快

在CPU中由5—6个不同功能的电路单元组成一条指令处理流水线，然后将一条X86指令分成5—6步后再由这些电路单元分别执行，这样就能实现在一个CPU时钟周期完成一条指令，因此提高CPU的运算速度。



2、影响指令流水性能的因素

(1) 访存冲突（指令执行与取指访存的冲突）



解决方法

- ①将指令和数据分开存放，设置指令Cache和数据Cache。
- ②采用指令预取技术，设置指令队列，存放多条指令。

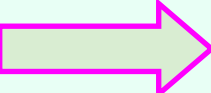
第8章 CPU的结构和功能 —— 8.3指令流水

2、影响指令流水性能的因素

(2) 相关问题


程序的相近指令之间出现**某种关联**使指令流水出现停顿，影响流水线效率。

①控制相关

当一条指令要等前一条（或几条）指令作出转移方向的决定后，才能进入流水线  便发生了控制相关

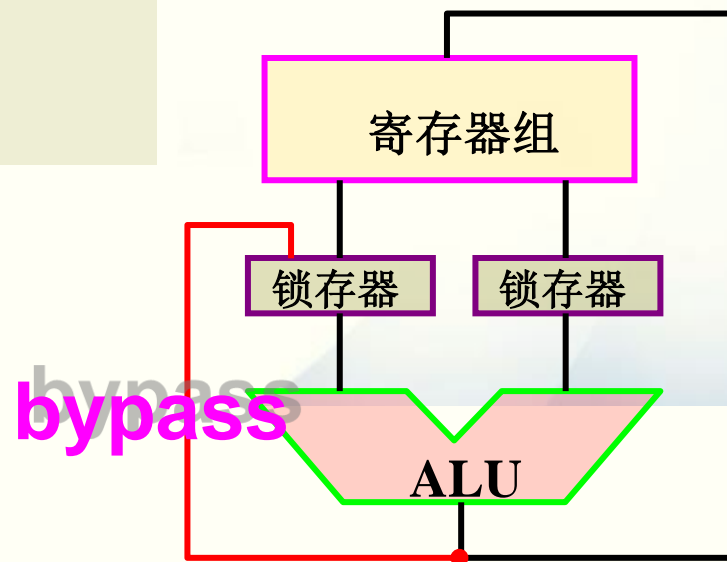
```
if( a > b )  
{  
    c = c + 2;  
}  
else  
{  
    c = c + 4;  
}
```

②数据相关

几条相近的指令间，共用 **同一存储单元** 或 **同一寄存器** 时  出现 **数据相关**。

如： **ADD R1, R2**
SUB R1, R3

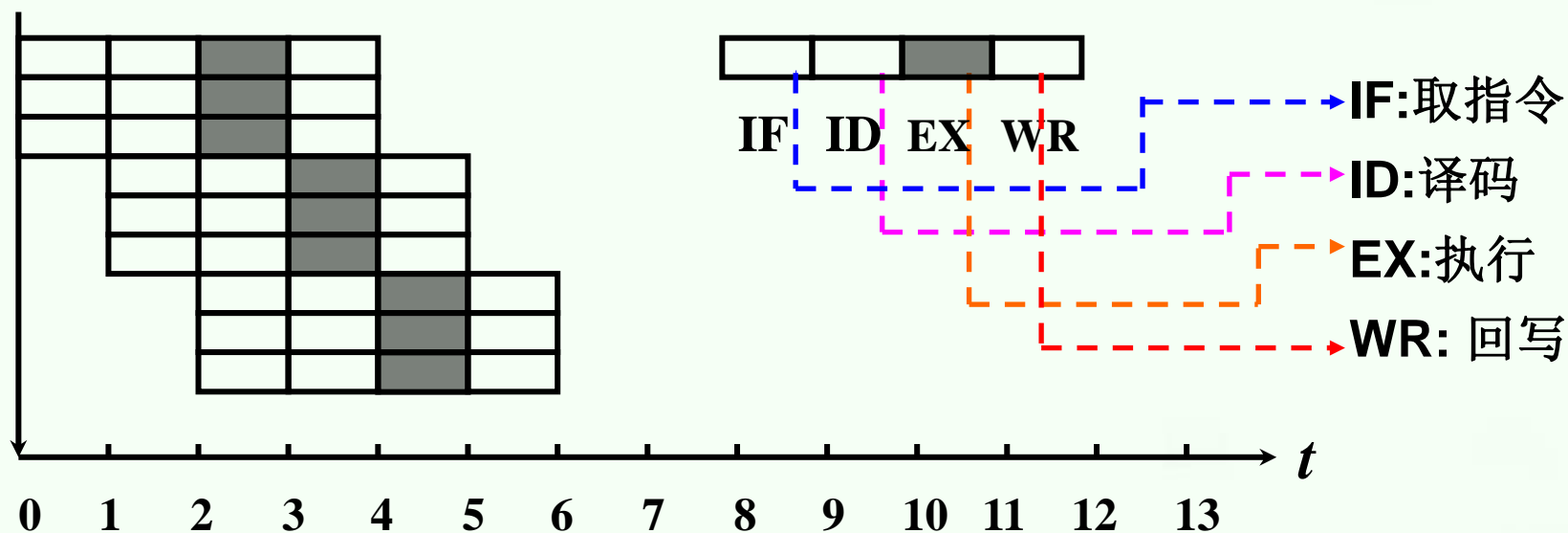
解决办法：
采用 **旁路技术**



3.指令流水中的多发技术

(1) 超标量技术 **精髓：** 多条流水线并行工作

每个时钟周期内可 **并发多条独立指令**（需要配置多个功能部件）



缺点： 不能调整 指令的 执行顺序

解决办法： 通过编译优化技术，把可并行执行的指令搭配起来

第8章 CPU的结构和功能 —— 8.3指令流水

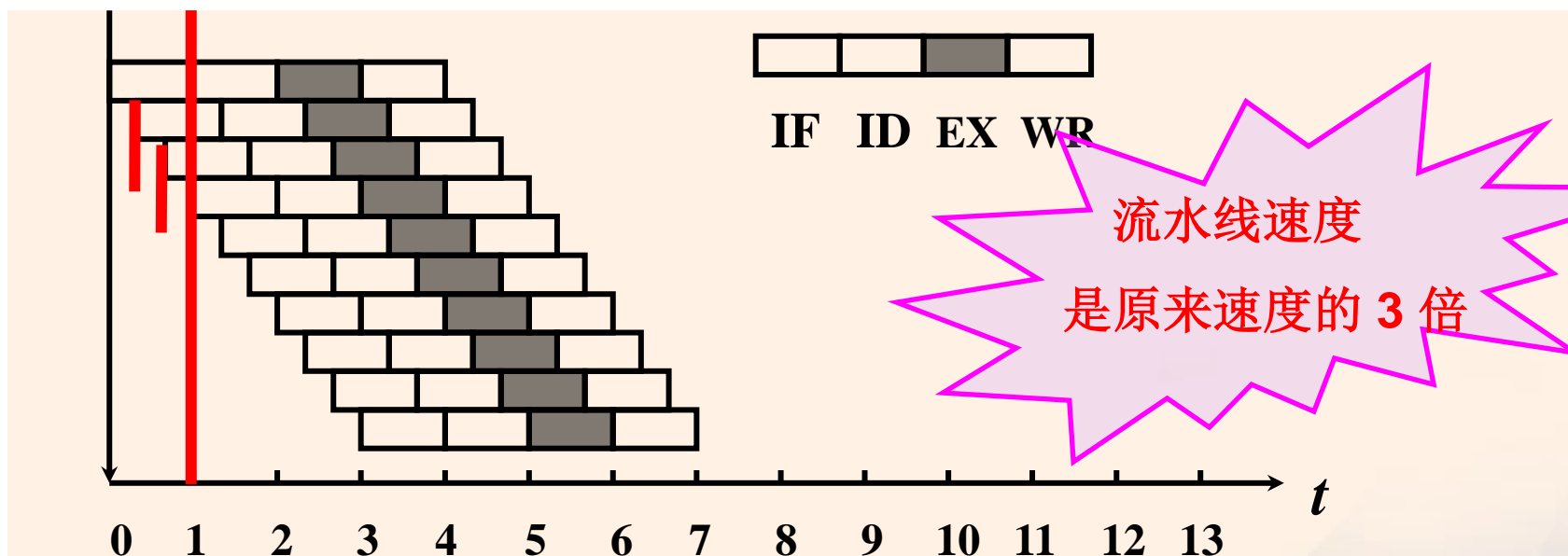
3.指令流水中的多发技术

(2) 超级流水线技术

在一个时钟周期内再分段（3段）

即在一个时钟周期内一个功能部件使用多次（3次）

精髓：一个时钟周期内并发多条指令，相当于时钟频率提高 n 倍。



缺点：不能调整指令的执行顺序

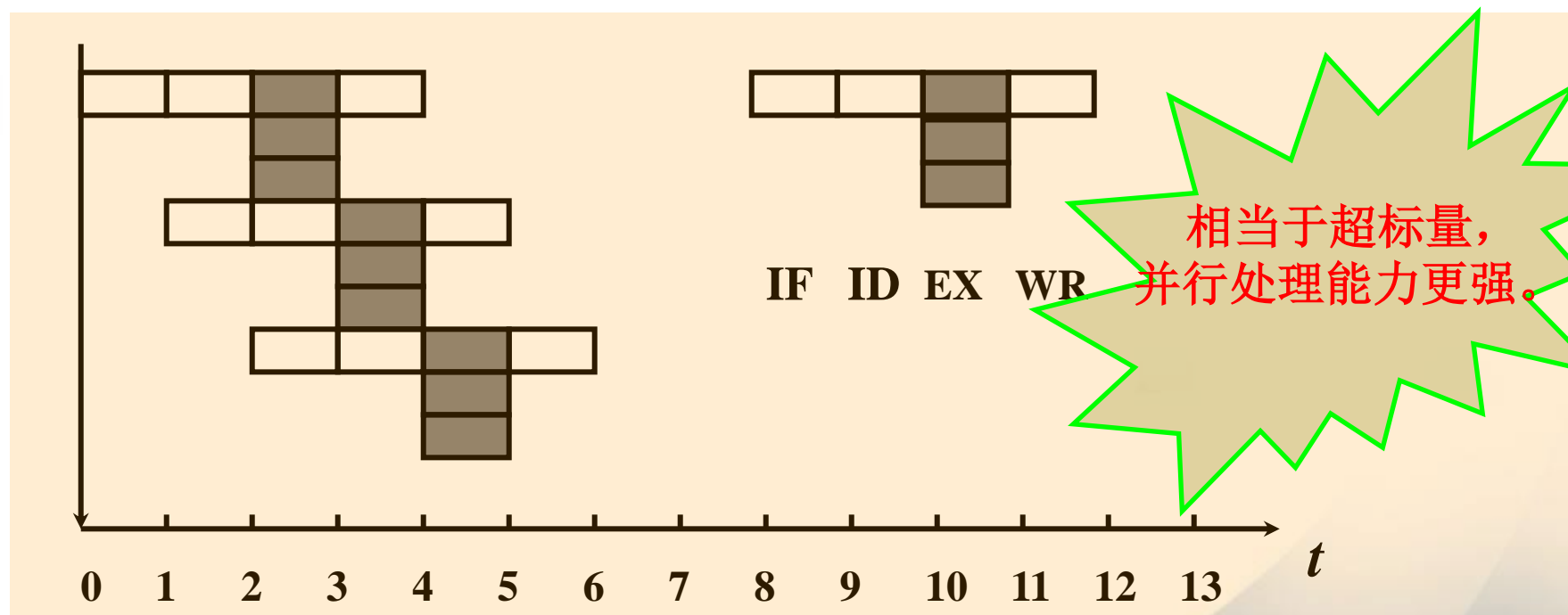
解决办法：靠编译程序解决优化问题

第8章 CPU的结构和功能 —— 8.3指令流水

3.指令流水中的多发技术

(3) 超长指令字技术

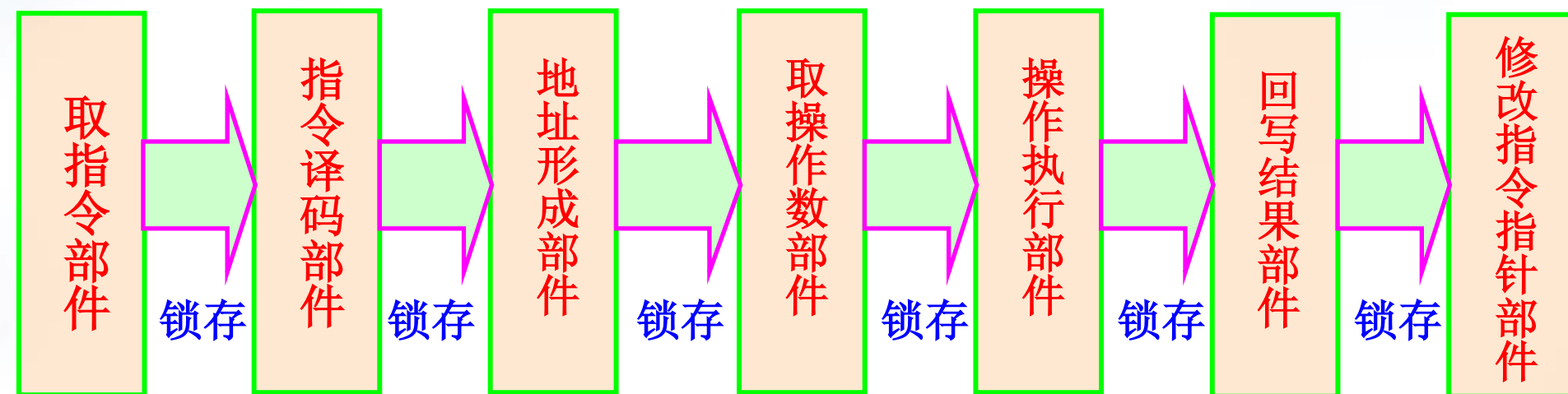
- ① 由编译程序挖掘出指令间潜在的并行性,将多条能并行操作的指令合并为一条指令。
- ② 有多个功能处理部件。



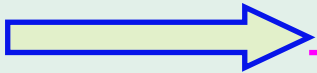
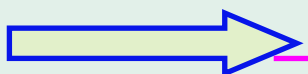
第8章 CPU的结构和功能 —— 8.3指令流水

三、流水线的基本结构

1. 指令流水线结构



若完成一条指令分 **7 段**，每段需一个时钟周期。

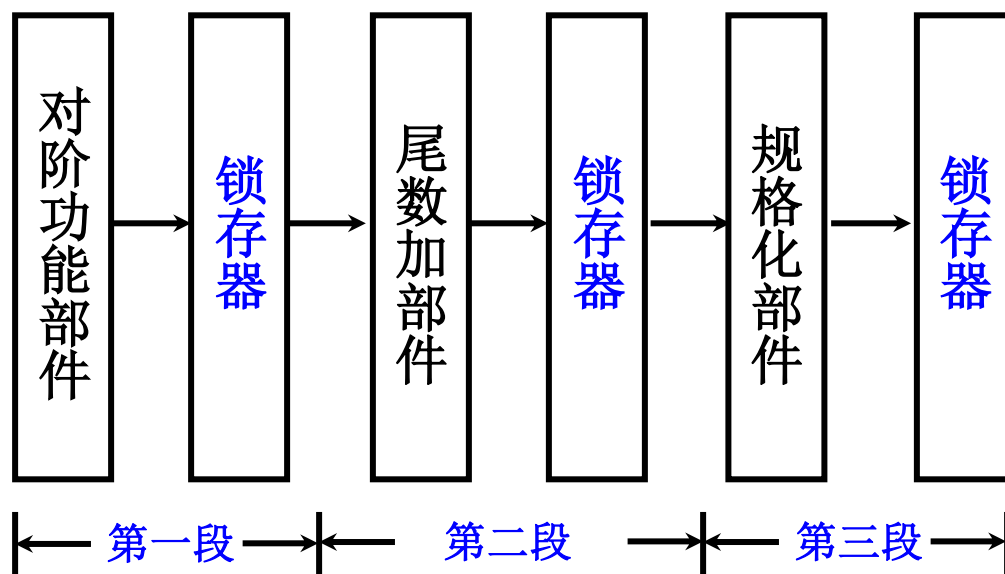
则 { 若流水线不出现断流，则  **1** 个时钟周期出 **1** 结果
若不采用流水技术，则  **7** 个时钟周期出 **1** 结果

结论：理想情况下 { **7 级流水** 的速度是不采用流水技术的 **7 倍**
n 级流水 的速度是不采用流水技术的 **n 倍**

2. 运算流水线

完成 浮点加减 运算 可分

对阶、尾数求和、规格化 三段



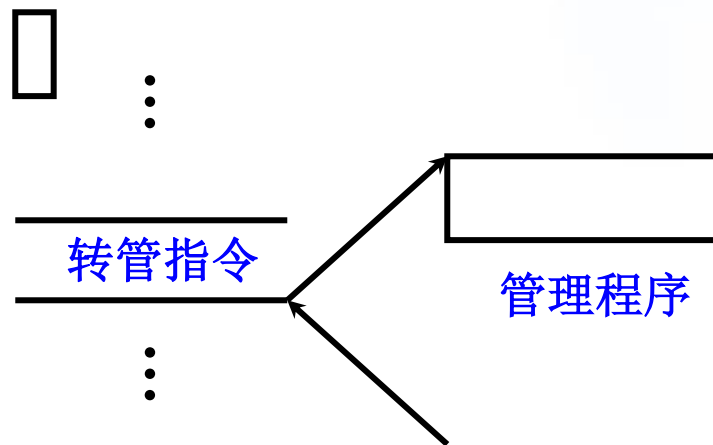
分段原则 每段 操作时间 尽量 一致

一、概述

1. 引起中断的各种因素

(1) 人为设置的中断

如 转管指令



(2) 程序性事故 溢出、操作码不能识别、除法非法

(3) 硬件故障

(4) I/O 设备

(5) 外部事件 用 键盘中断 运行程序

2. 中断系统需解决的问题

- (1) 各中断源 如何 向 CPU 提出请求？
- (2) 各中断源 同时 提出 请求 怎么办？
- (3) CPU 什么 条件、什么 时间、以什么 方式响应中断？
- (4) 如何 保护现场？
- (5) 如何 寻找入口地址？
- (6) 如何 恢复现场，如何 返回？
- (7) 处理中断的过程中又 出现新的中断 怎么办？

硬件 + 软件

二、中断请求标记和中断判优逻辑

1. 中断请求标记 **INTR**

一个请求源 一个 **INTR** 中断请求标记触发器

多个**INTR** 组成 中断请求标记寄存器

1	2	3	4	5			<i>n</i>
掉电	过热	主存读写校验错	阶上溢	非法除法		键盘输入	打印机输出

INTR 分散 在各个中断源的 接口电路中

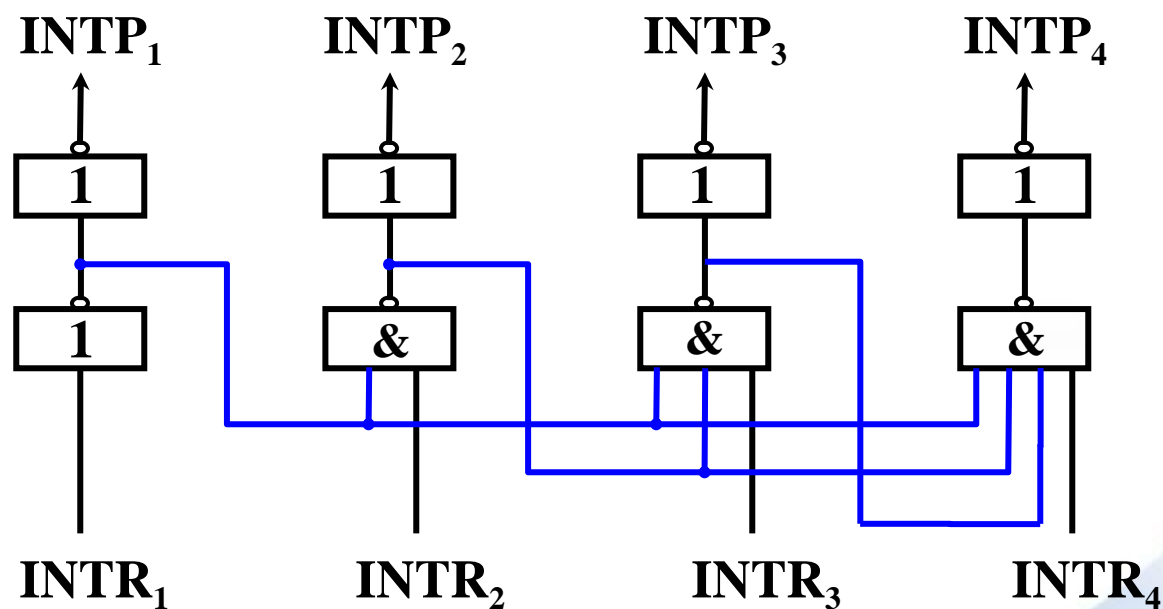
INTR 集中 在 **CPU** 的中断系统 内

2. 中断判优逻辑

(1) 硬件实现（排队器）

① 分散 在各个中断源的 接口电路中 链式排队器 参见 第五章

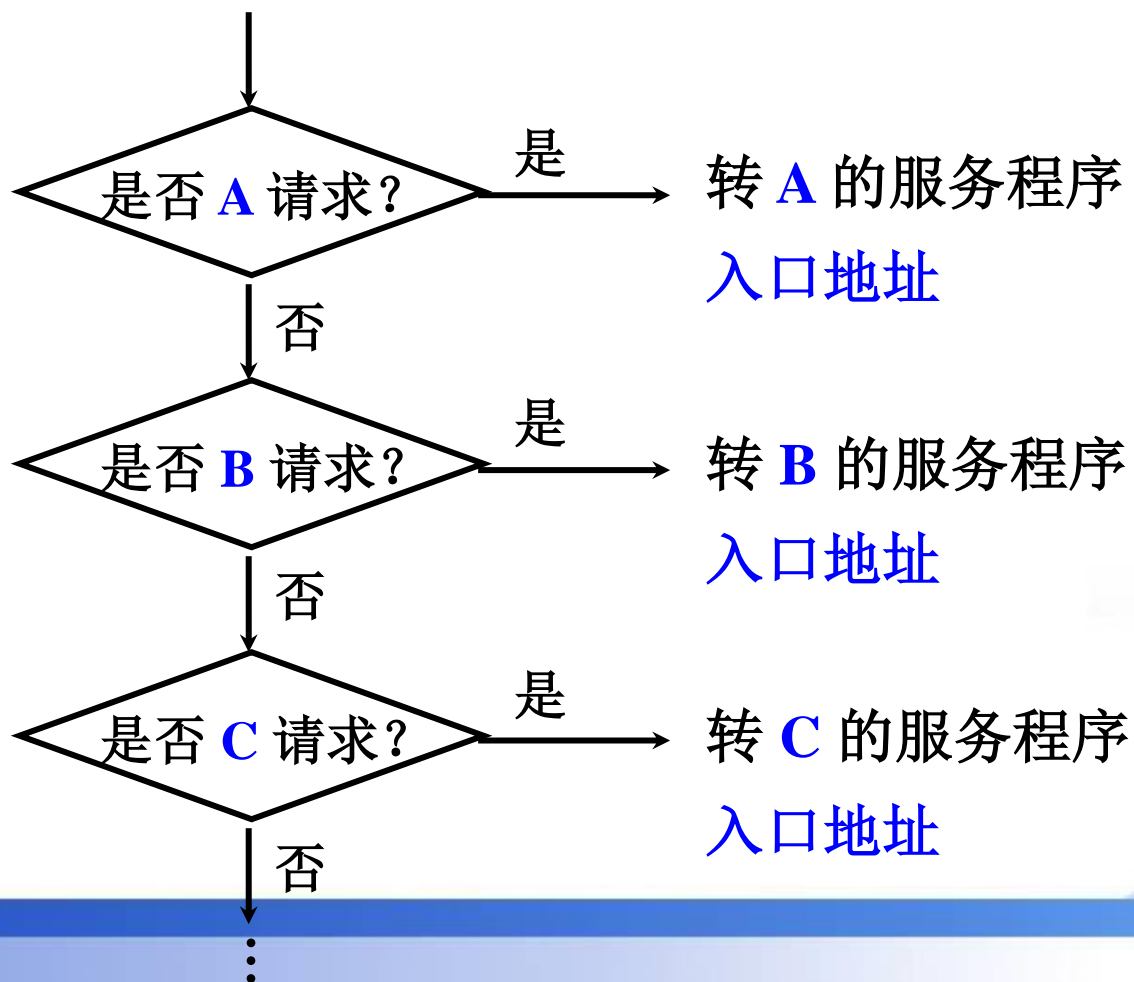
② 集中 在 CPU 内



$INTR_1$ 、 $INTR_2$ 、 $INTR_3$ 、 $INTR_4$ 优先级按降序排列

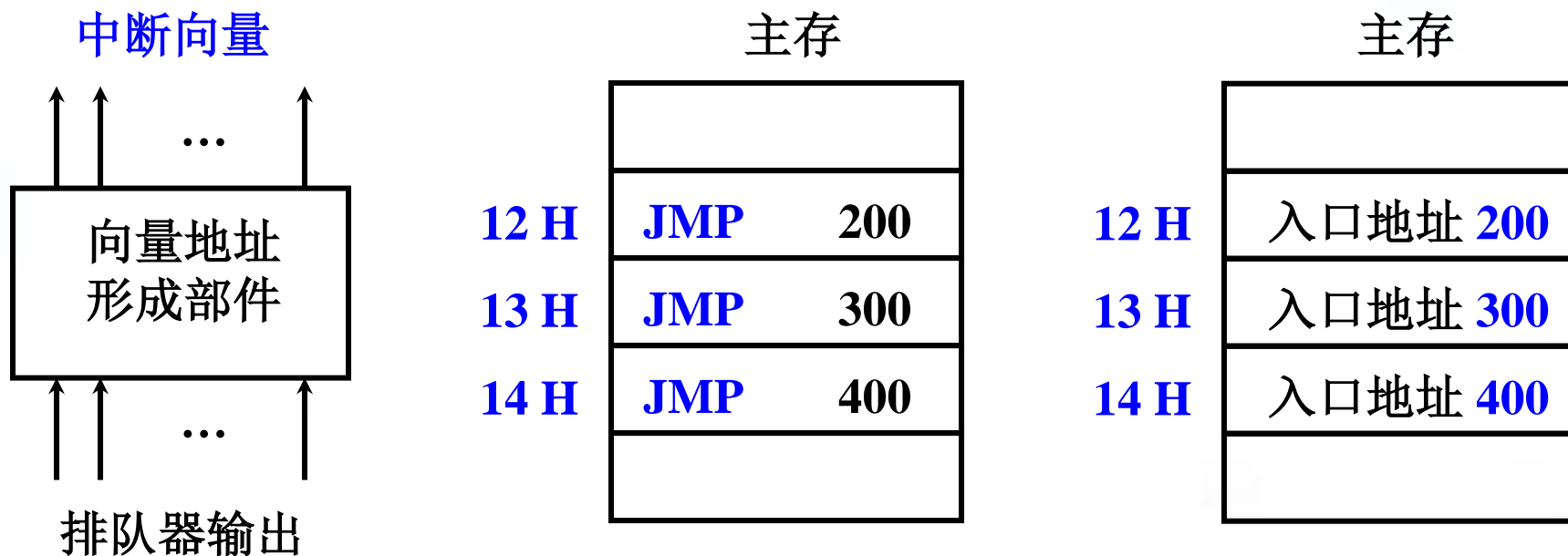
(2) 软件实现（程序查询）

A、B、C 优先级按 降序 排列



三、中断服务程序入口地址的寻找

1. 硬件向量法



向量地址 12H、13H、14H

入口地址 200、300、400

2. 软件查询法

八个中断源 1, 2, ... 8 按 **降序** 排列

中断识别程序 (入口地址 **M**)

地 址	指 令	说 明
M	SKP DZ 1 [#]	1 [#] D = 0 跳 (D为完成触发器)
	JMP 1 [#] SR	1 [#] D = 1 转1 [#] 服务程序
	SKP DZ 2 [#]	2 [#] D = 0 跳
	JMP 2 [#] SR	2 [#] D = 1 转2 [#] 服务程序
	⋮	
	SKP DZ 8 [#]	8 [#] D = 0 跳
	JMP 8 [#] SR	8 [#] D = 1 转8 [#] 服务程序

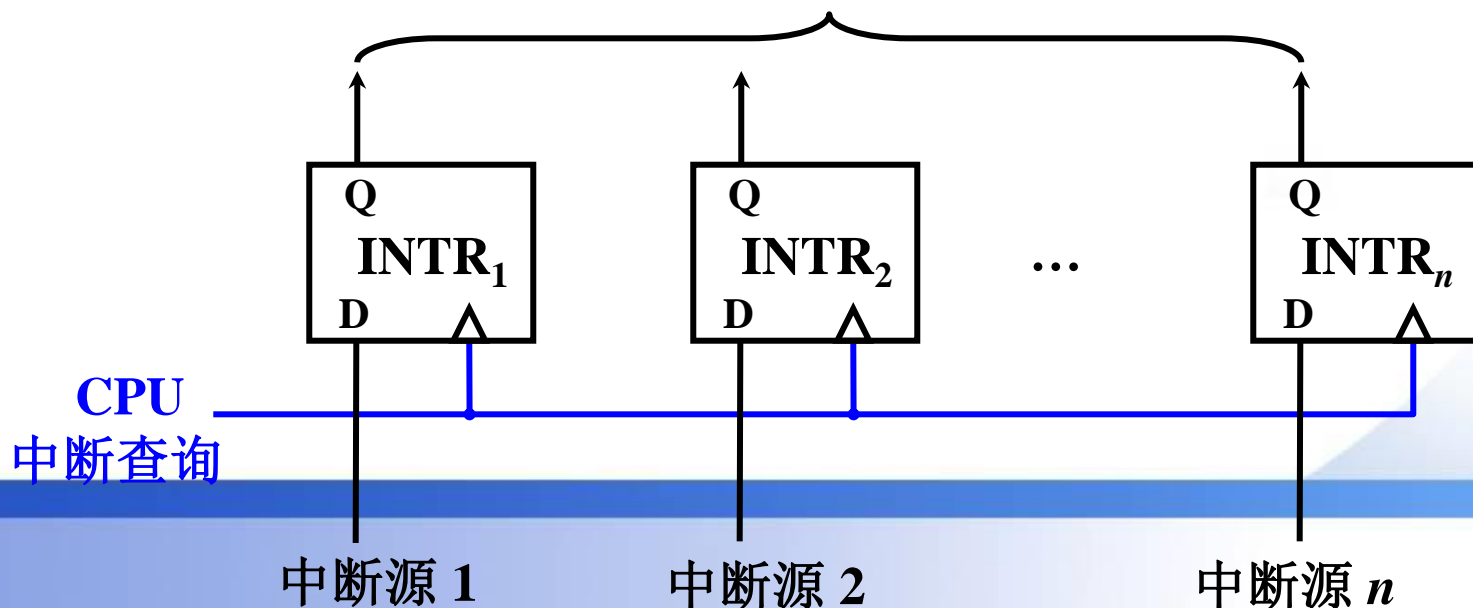
四、中断响应

1. 响应中断的条件

允许中断触发器 $EINT = 1$

2. 响应中断的时间

指令执行周期结束时刻由CPU发查询信号
至排队器



第8章 CPU的结构和功能 —— 8.4 中断系统

3. 中断隐指令

(1) 保护程序断点：断点存于 **特定地址**（**0 号地址**）内 断点 **进栈**

(2) 寻找服务程序入口地址

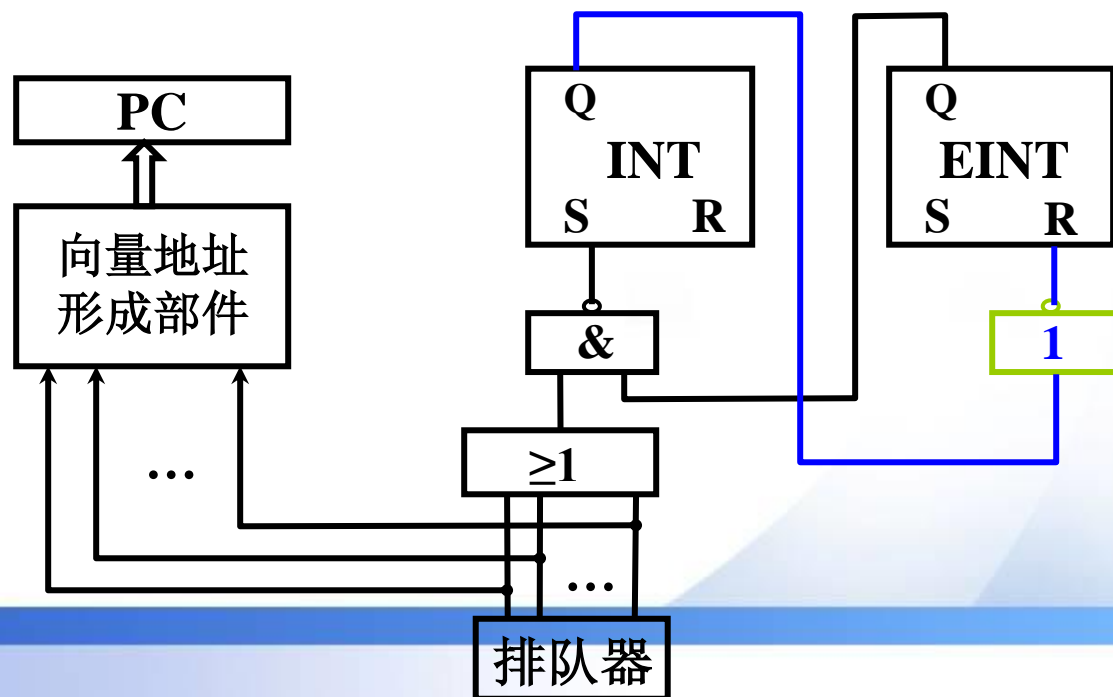
向量地址 \rightarrow **PC** （硬件向量法）

中断识别程序 入口地址 **M** \rightarrow **PC** （软件查询法）

(3) 硬件 **关中断**

INT **EINT**
中断 允许
标记 中断

R - S 触发器



五、保护现场和恢复现场

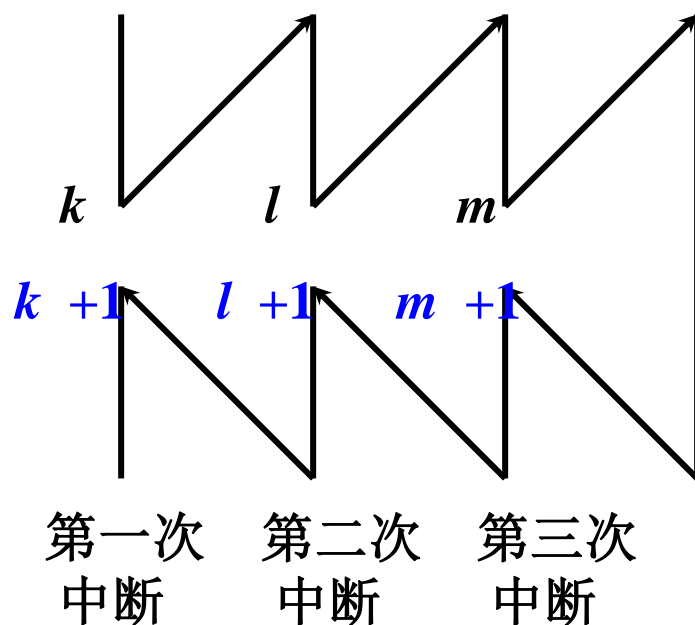
1. 保护现场 { 断点 中断隐指令 完成
 寄存器 内容 中断服务程序 完成

2. 恢复现场 中断服务程序 完成



六、中断屏蔽技术

1. 多重中断的概念



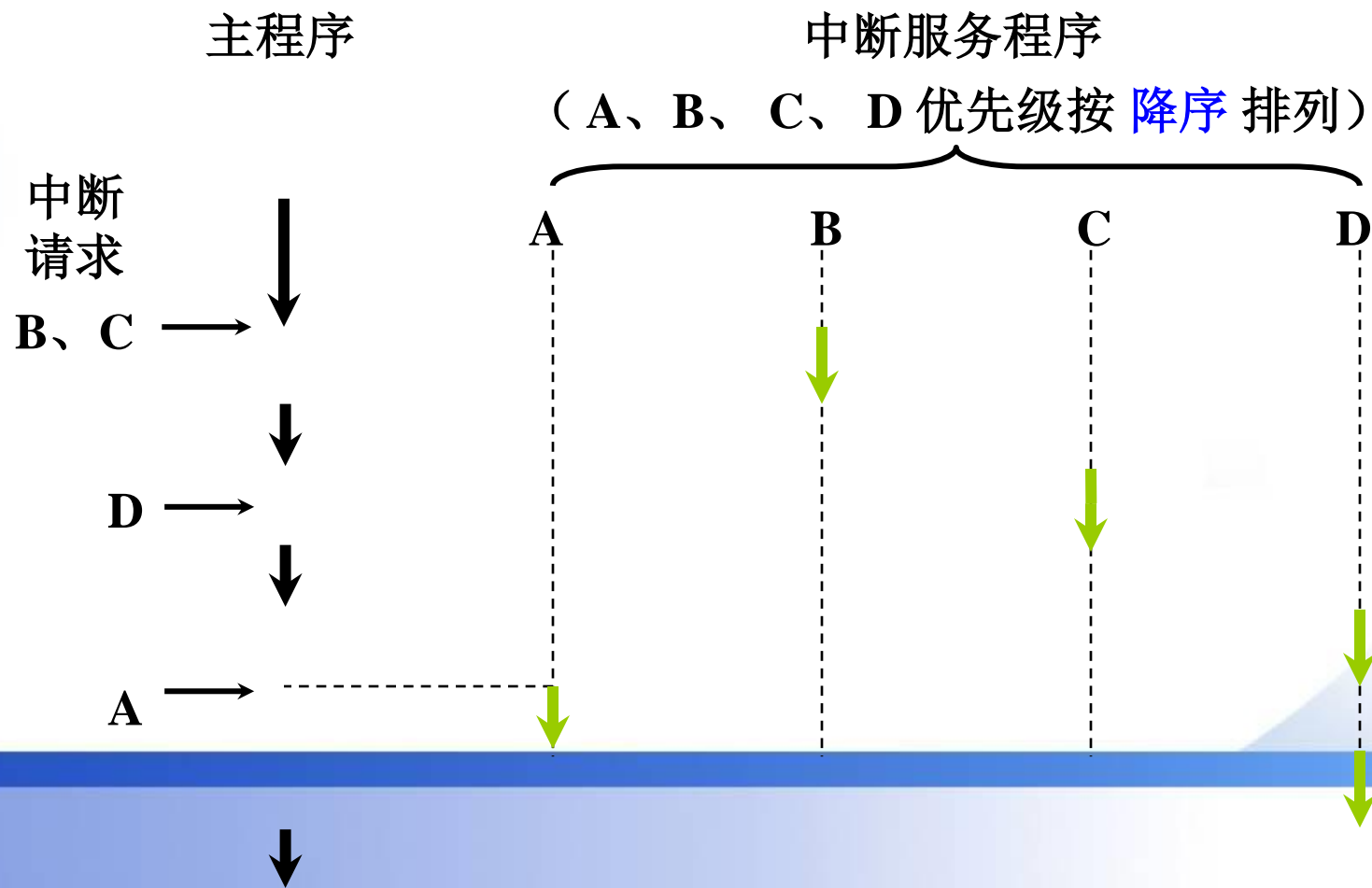
程序断点 $k+1$, $l+1$, $m+1$

第8章 CPU的结构和功能 —— 8.4 中断系统

2. 实现多重中断的条件

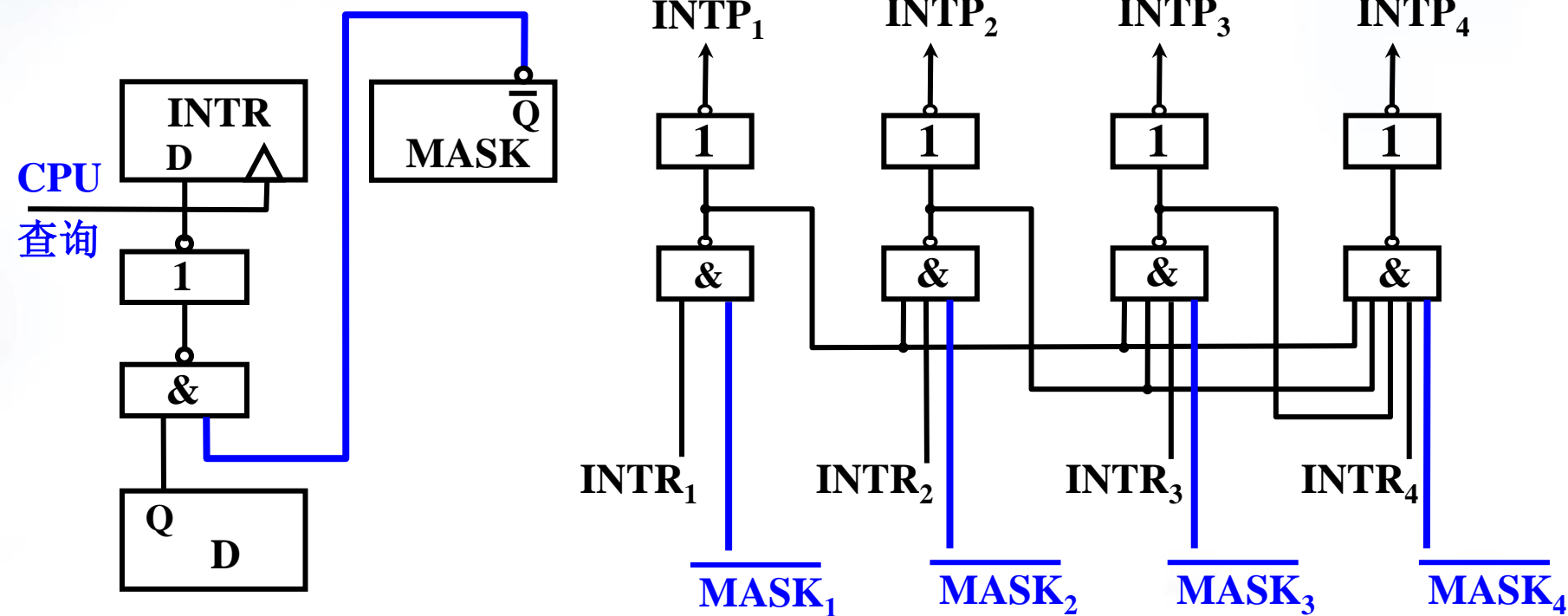
(1) 提前 设置 开中断 指令

(2) 优先级别高 的中断源 有权中断优先级别低 的中断源



3. 屏蔽技术

(1) 屏蔽触发器的作用



$MASK = 0$ (未屏蔽)

$MASK_i = 1$ (屏蔽)

INTR 能被置“1”

$INTP_i = 0$ (不能被排队选中)

(2) 屏蔽字

16个中断源 1, 2, 3, ..., 16 按 降序 排列

优先级	屏	蔽	字
1	1	1	1
2	0	1	1
3	0	0	1
4	0	0	0
5	0	0	0
6	0	0	0
⋮			
15	0	0	0
16	0	0	0

(3) 屏蔽技术可改变处理优先等级

响应优先级 不可改变

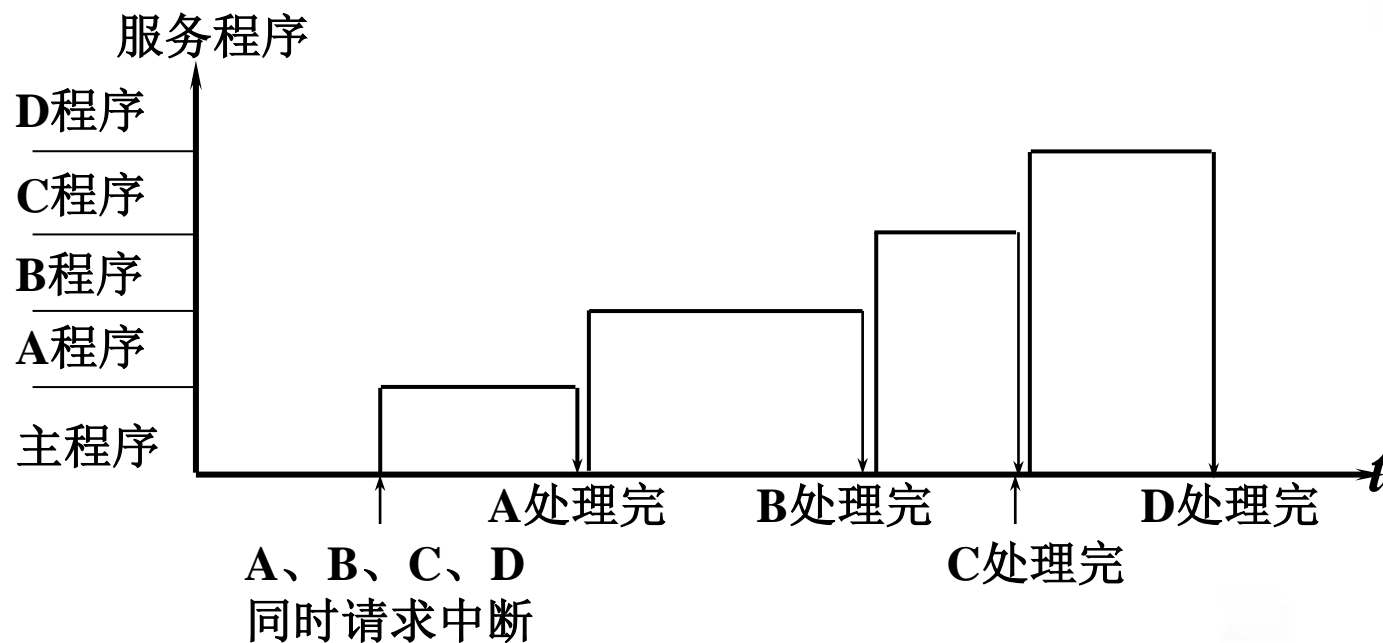
处理优先级 可改变（通过重新设置屏蔽字）

中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

响应优先级 **A→B→C→D** 降序排列

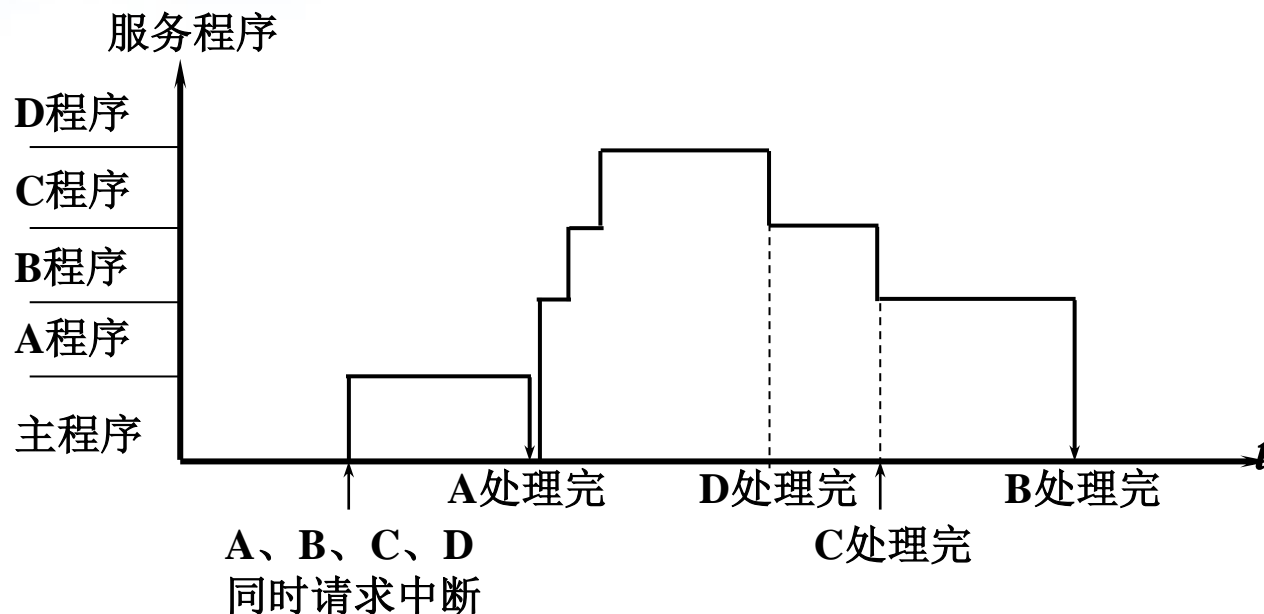
处理优先级 **A→D→C→B** 降序排列

(3) 屏蔽技术可改变处理优先等级



CPU 执行程序轨迹（原屏蔽字）

(3) 屏蔽技术可改变处理优先等级



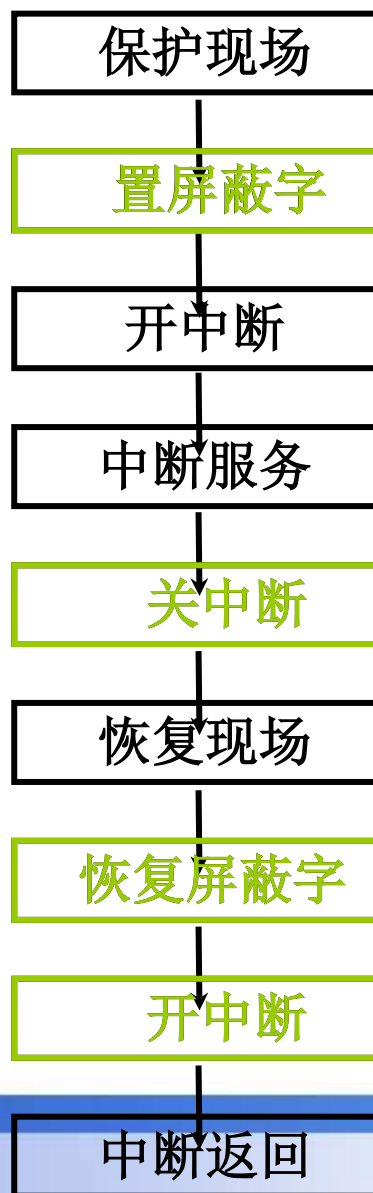
CPU 执行程序轨迹（新屏蔽字）

(4) 屏蔽技术的其他作用

可以 **人为地屏蔽** 某个中断源的请求

便于程序控制

(5) 新屏蔽字的设置



4. 多重中断的断点保护

(1) 断点进栈

中断隐指令 完成

(2) 断点存入“0”地址

中断隐指令 完成

中断周期

0 \longrightarrow **MAR**

命令存储器写

PC → MDR

断点 — MDR

(MDR) → 存入存储器

三次中断，三个断点都存入“0”地址

? 如何保证断点不丢失?

第8章 CPU的结构和功能 —— 8.4 中断系统

(3) 程序断点存入 “0” 地址的断点保护

地 址	内 容	说 明
0	××××	存程序断点
5	JMP SERVE	5 为向量地址
SERVE	STA SAVE	保护现场
	⋮	
置屏蔽字	LDA 0	} 0 地址内容转存
→	STA RETURN	
	ENI	开中断
	⋮	} 其他服务内容
	LDA SAVE	
	JMP @ RETURN	恢复现场
SAVE	××××	间址返回
RETURN	××××	存放 ACC 内容
		转存 0 地址内容

• **8.25.** 某机有五个中断源L0、L1、L2、L3、L4，按中断响应的优先次序由高向低排序为L0→L1→L2→L3→L4，根据下示格式，现要求中断处理次序改为L1→L4→L2→L0→L3，根据下面的格式，写出各中断源的屏蔽字。

- 解：各中断源屏蔽状态见下表：
表中：设屏蔽位=1，表示屏蔽；屏蔽位=0，表示中断开放。

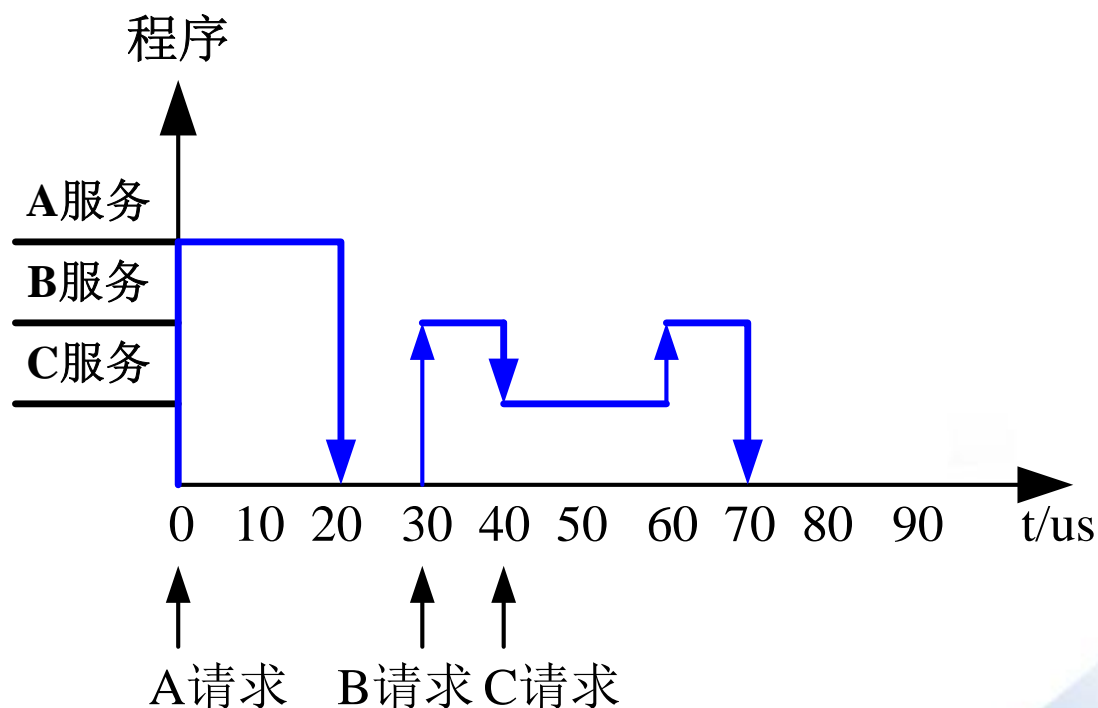
中断源	屏蔽字				
	0	1	2	3	4
L0	1	0	0	1	0
L1	1	1	1	1	1
L2	1	0	1	1	0
L3	0	0	0	1	0
L4	1	0	1	1	1

•8.26.设某机配有**A**、**B**、**C**三台设备，其优先顺序按**A→B→C**降序排列，为改变中断处理次序，它们的中断屏蔽字设置如下：

设备	屏蔽字
A	111
B	010
C	011

请按下图所示时间轴给出的设备请求中断的时刻，画出**CPU**执行程序的轨迹。设**A**、**B**、**C**中断服务程序的执行时间均为**20μ s**。

- 解：A、B、C设备的响应优先级为A最高、B次之、C最低，处理优先级为A最高、C次之、B最低。CPU执行程序的轨迹图如下：



练习题

1. **CPU**响应中断的时间是_____。

- A. 中断源提出请求;
- B. 取指周期结束;
- C. 执行周期结束;
- D. 间址周期结束。

2. 中断向量可提供_____。

- A.被选中设备的地址;
- B.传送数据的起始地址;
- C.中断服务程序入口地址;
- D.主程序的断点地址

3. **CPU**响应中断时要保护现场, 包括对_____和_____的保护, 前者通过_____实现, 后者可通过_____实现。