



3.2 CPU控制器

船说：计算机基础

3. 指令系统设计与CPU运行控制

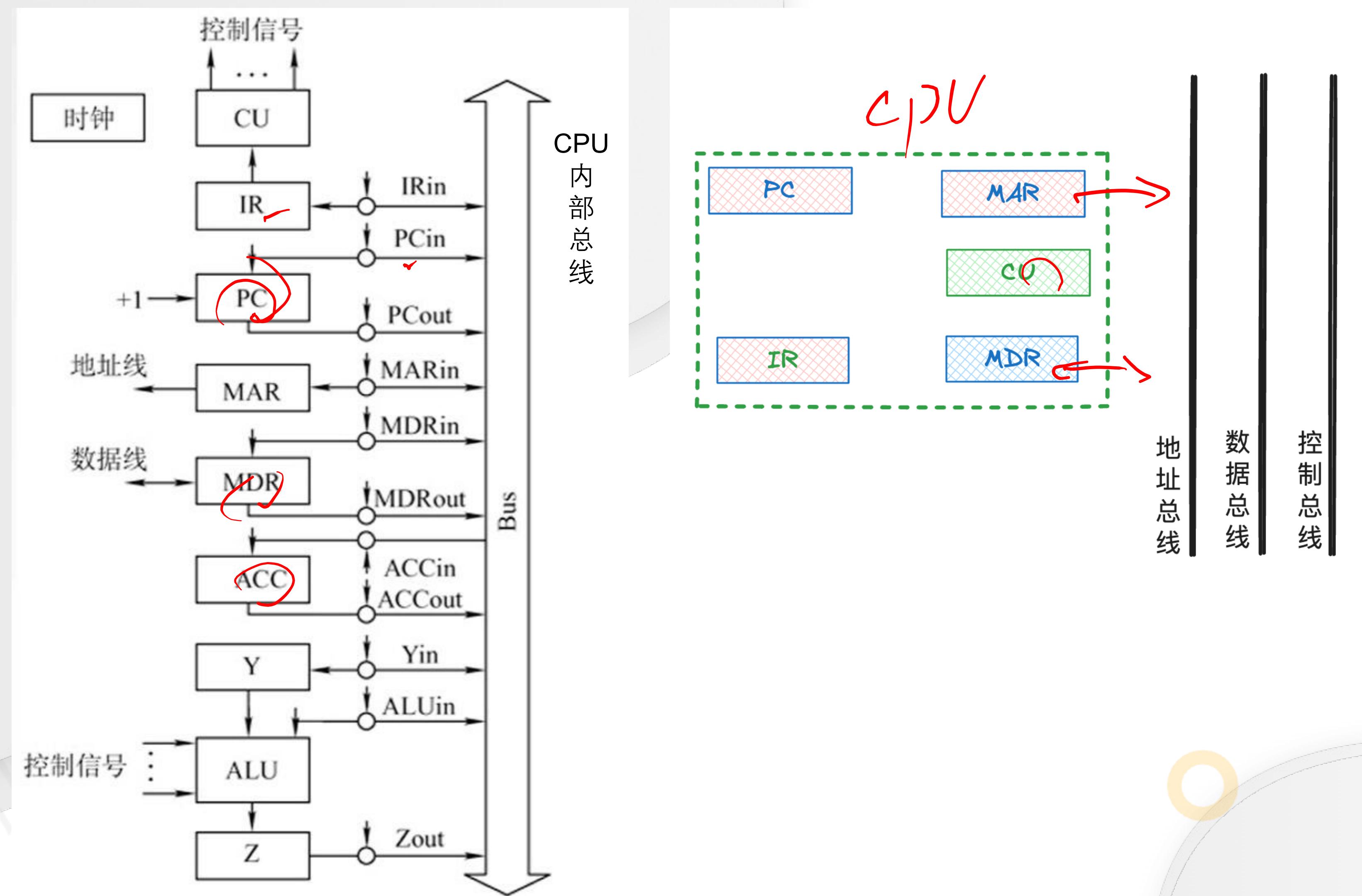
2 CPU控制器

1. 控制器的硬布线控制单元设计
2. 微程序控制器
3. 指令流水线
4. 异常和中断
5. 多处理器介绍



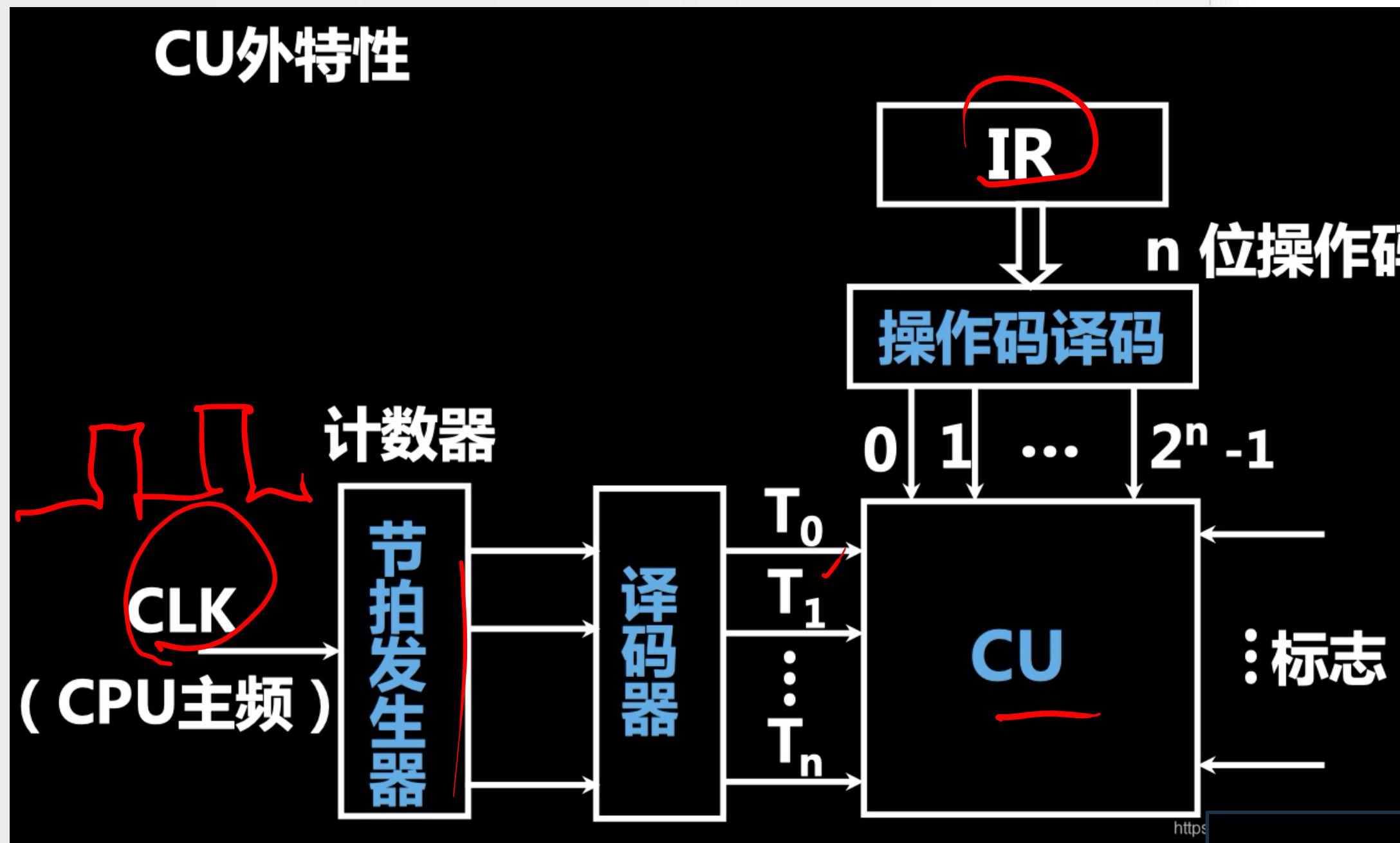
3. 指令系统设计与CPU运行控制

3.2.1 控制器就是指挥中心



3. 指令系统设计与CPU运行控制

3.2.1 控制单元CU的工作节拍



安排微操作时序的原则：

- ① 微操作的先后顺序一般不得随意更改。
- ② 被控对象不同的微操作尽量安排在一个节拍内完成。
- ③ 占用时间较短的微操作，尽量安排在一个节拍内完成，并允许有先后顺序。

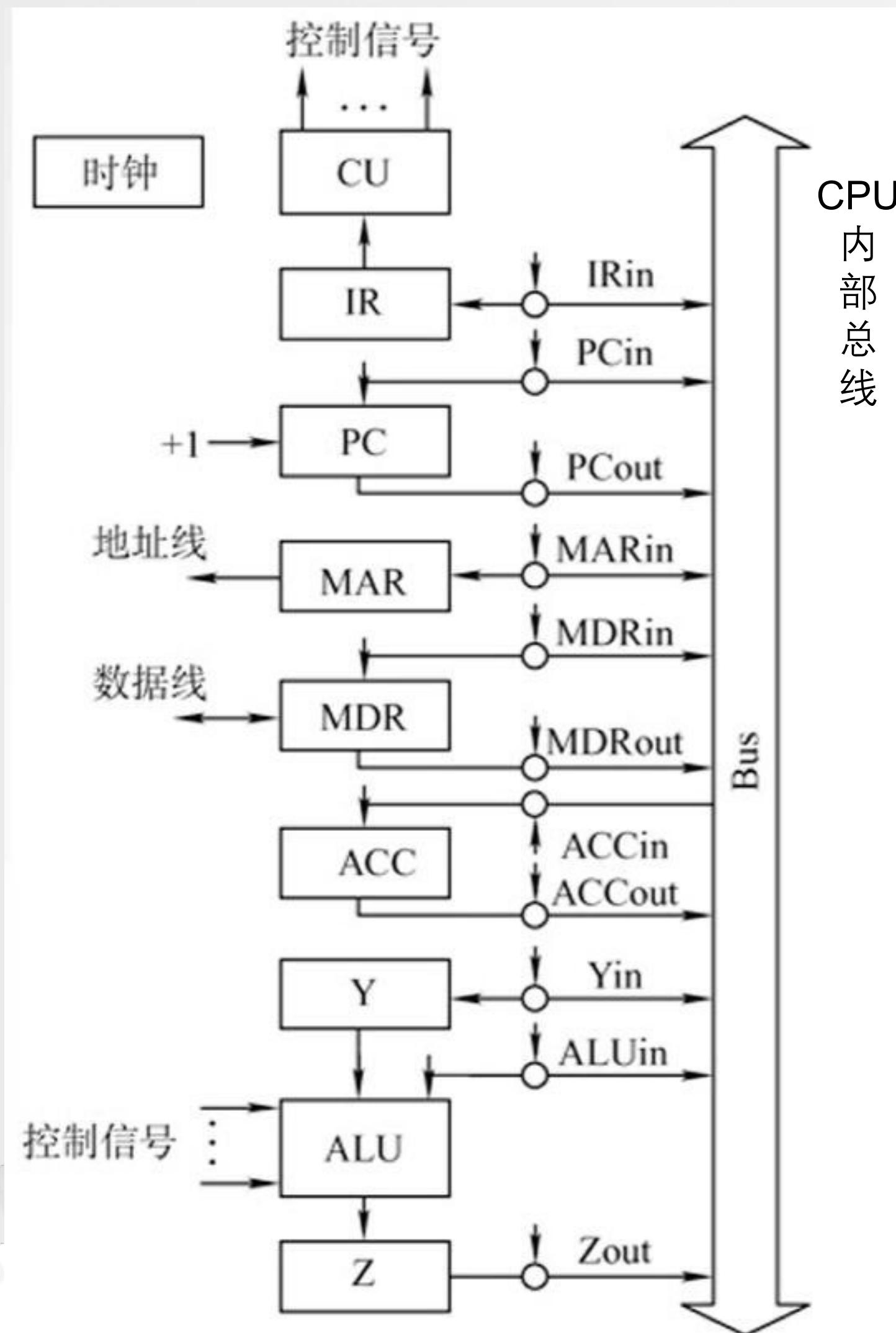
CU的微操作设计的前提假设：

- 采用同步控制方式（异步，联合）。
- 一个机器周期内有3个节拍。（时钟周期 T_0, T_1, T_2 ）
- CPU 内部结构采用非总线方式。



3. 指令系统设计与CPU运行控制

3.2.1 指令周期微操作节拍安排



1、取指周期：

T0: PC \rightarrow MAR, 1 \rightarrow R

T1: M (MAR) \rightarrow MDR, (PC) +1 \rightarrow PC

T2: MDR \rightarrow IR, OP (IR) \rightarrow ID

2、间址周期：

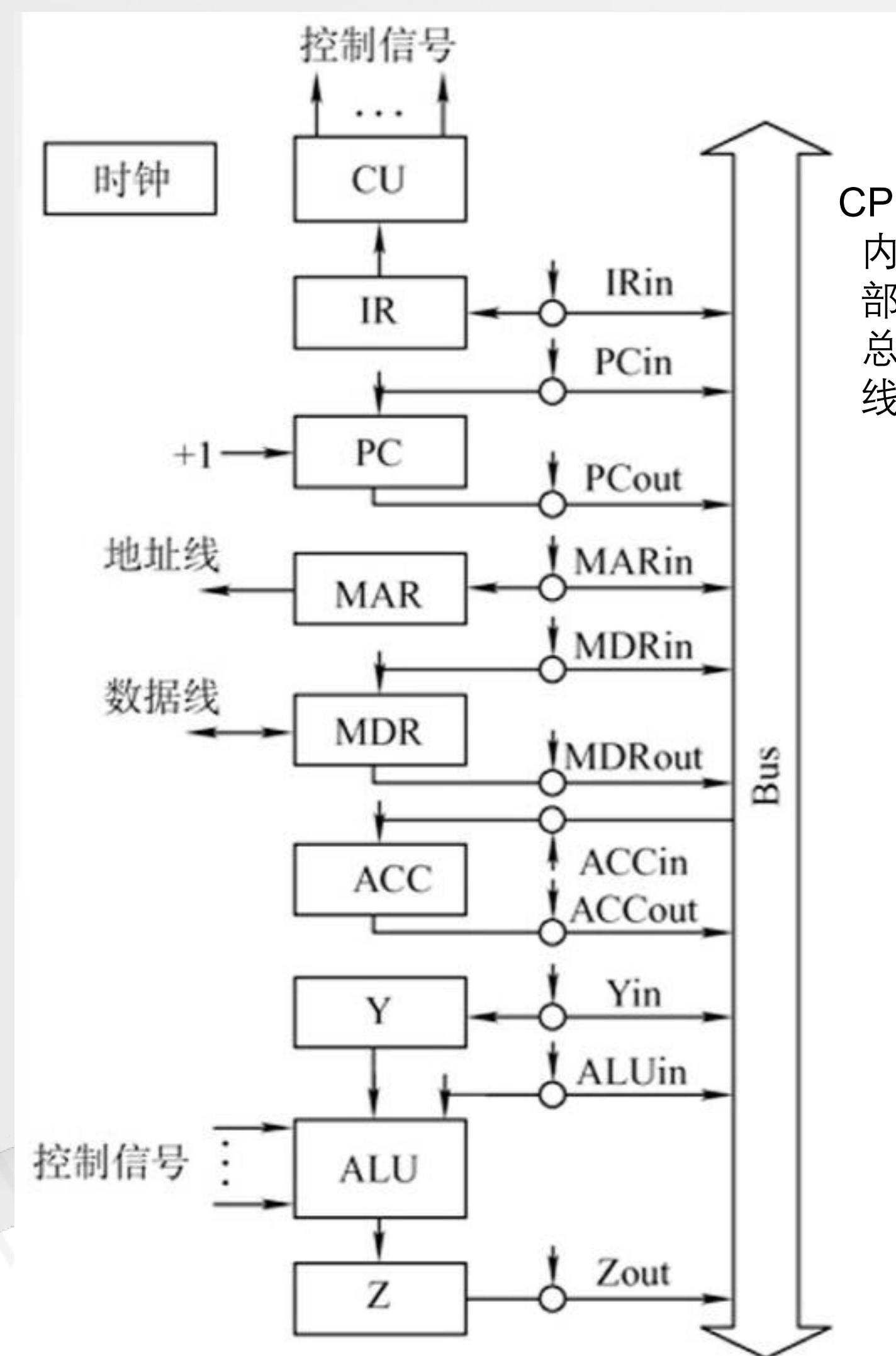
T0: Ad (IR) \rightarrow MAR, 1 \rightarrow R

T1: M (MAR) \rightarrow MDR

T2: MDR \rightarrow Ad (IR)

3. 指令系统设计与CPU运行控制

3.2.1 指令周期微操作节拍安排



3.1、CLA、COM、SHR、CSL、STP执行周期：

T0：空

T1：空

T2：执行

3.2、ADD X (ACC+X) 执行周期：

T0：Ad (IR) → MAR, 1 → R

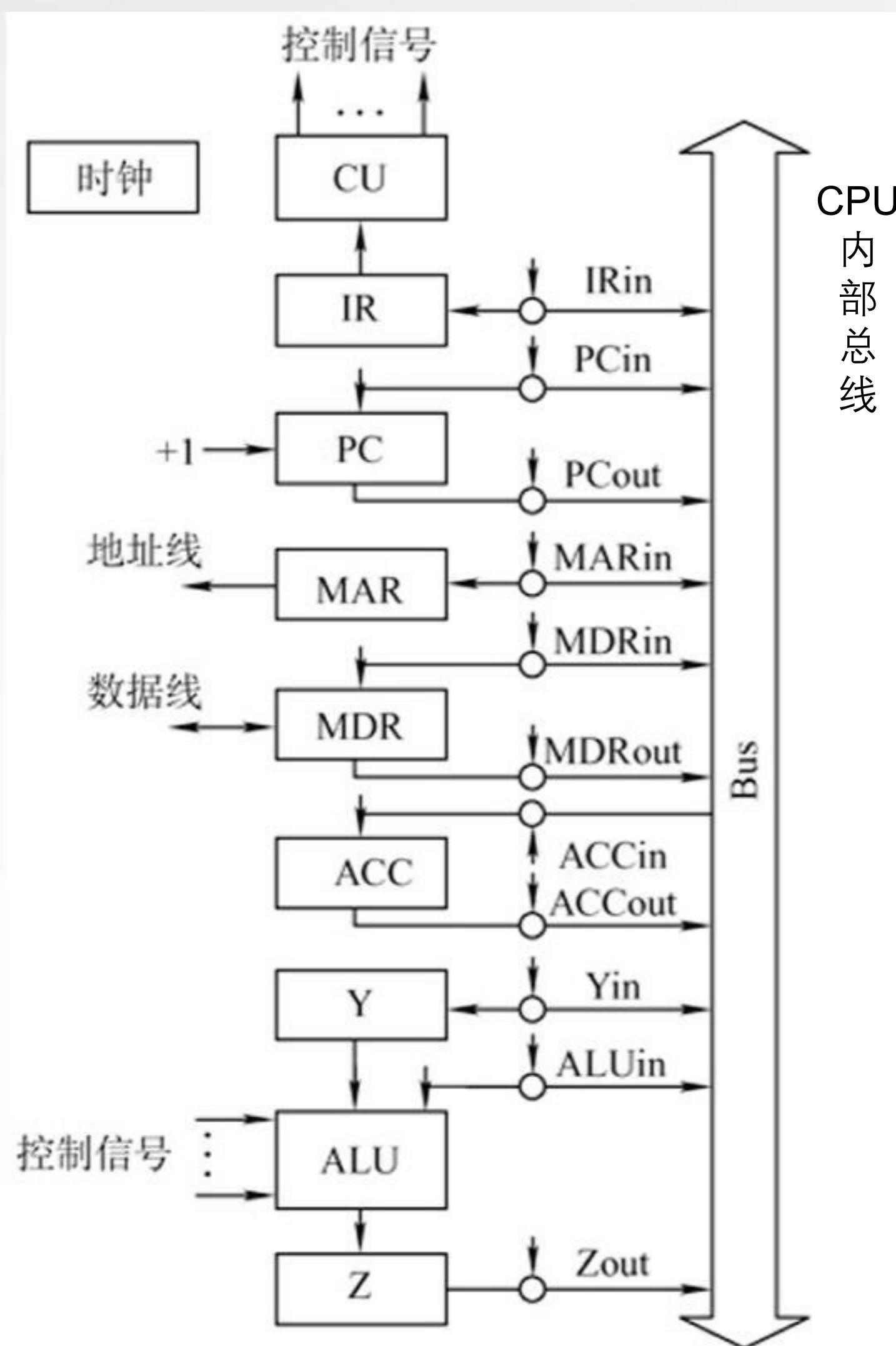
T1：M (MAR) → MDR

T2：(AC) + MDR → AC



3. 指令系统设计与CPU运行控制

3.2.1 指令周期微操作节拍安排



3.3、STAX (存数) 执行周期:

T0: Ad (IR) → MAR, 1 → W

T1: AC → MDR

T2: MDR → M (MAR)

3.4、LDA X (读数) 执行周期:

T0: Ad (IR) → MAR, 1 → R

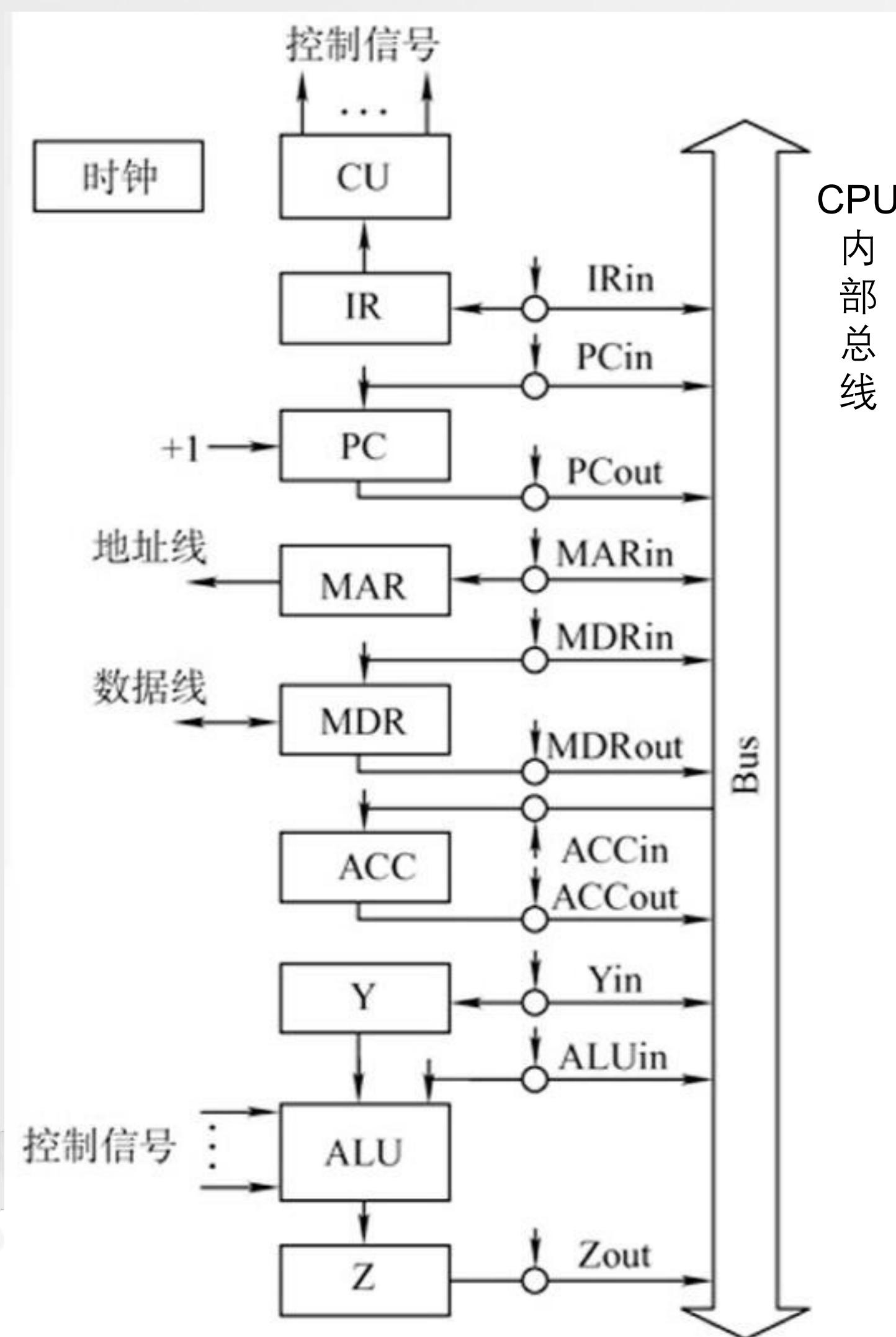
T1: M (MAR) → MDR

T2: MDR → AC



3. 指令系统设计与CPU运行控制

3.2.1 指令周期微操作节拍安排



3.5、JMP X (跳转) 执行周期:

T0: 空

T1: 空

T2: $Ad(IR) \rightarrow PC$

3.6、BAN X (条件跳转) 执行周期:

T0: 空

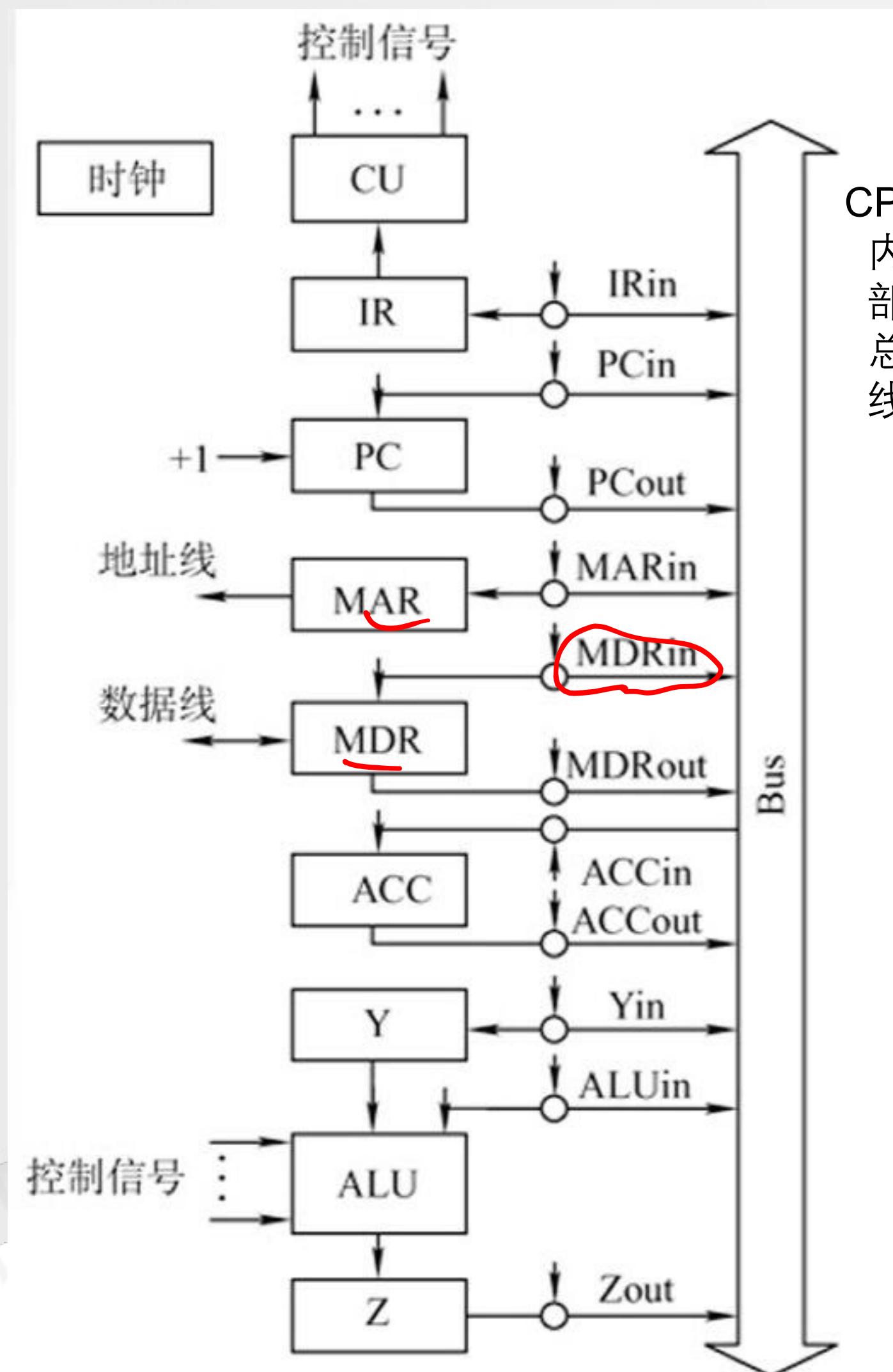
T1: 空

T2: $A0 * Ad(IR) + A0 (PC) \rightarrow PC$



3. 指令系统设计与CPU运行控制

3.2.1 指令周期微操作节拍安排



4、中断周期：（假设程序断点存入主存0号地址）
SP?

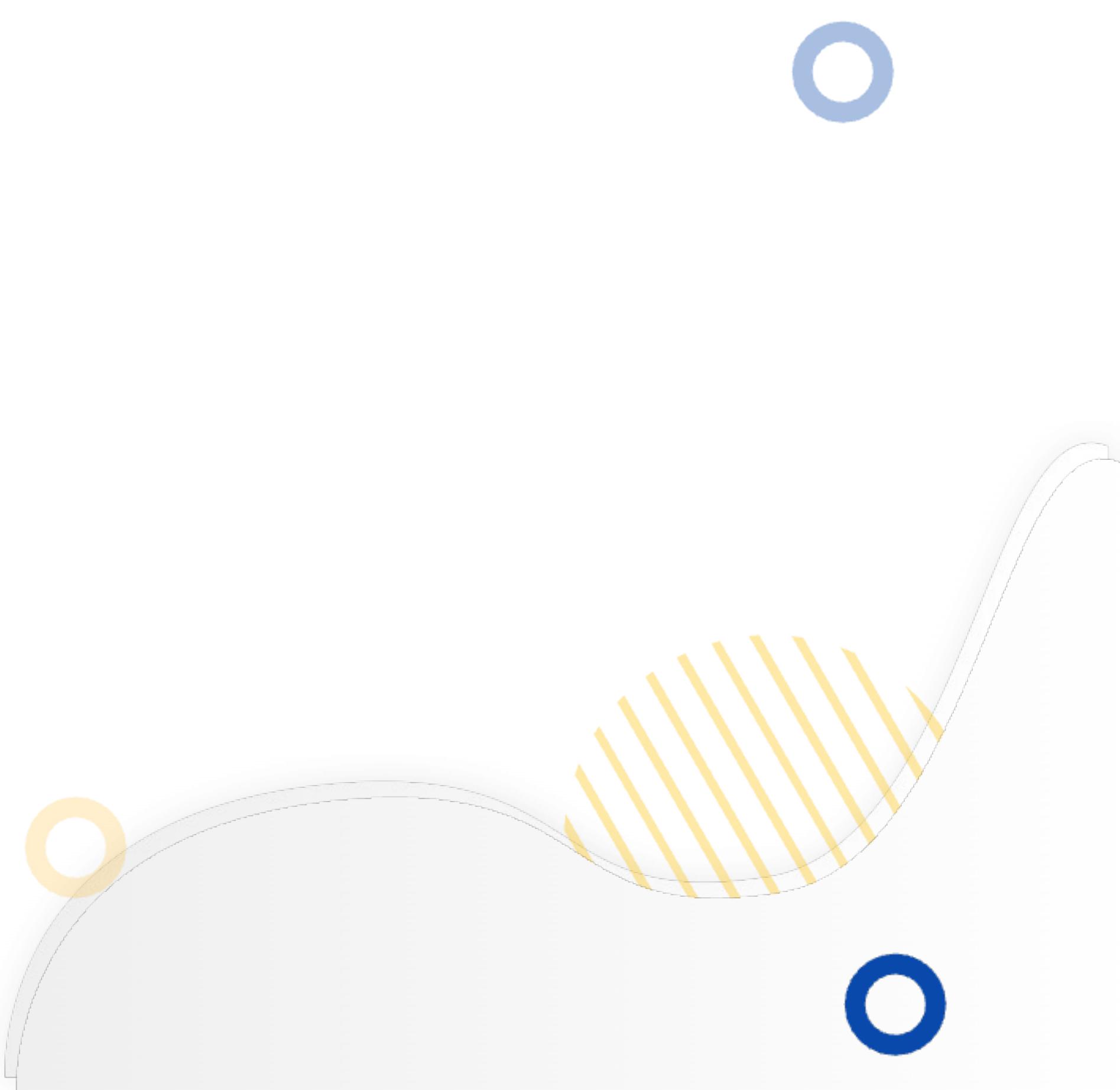
T0: 0 → MAR, 1 → W, 0 → EINT (关中断)

T1: PC → MDR

T2: MDR → M(MAR), 向量地址 → PC

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	SHR	CSL	STP	ADD	STA	LDA	JMP	BAN
FE (取指)	T_0		PC \rightarrow MAR	1	1	1	1	1	1	1	1	1	1
			1 \rightarrow R	1	1	1	1	1	1	1	1	1	1
	T_1		M(MAR) \rightarrow MDR	1	1	1	1	1	1	1	1	1	1
			(PC) + 1 \rightarrow PC	1	1	1	1	1	1	1	1	1	1
	T_2		MDR \rightarrow IR	1	1	1	1	1	1	1	1	1	1
			OP(IR) \rightarrow ID	1	1	1	1	1	1	1	1	1	1
IND (间接寻址)	T_0	I	I \rightarrow IND						1	1	1	1	1
			I \rightarrow EX	1	1	1	1	1	1	1	1	1	1
	T_1		Ad(IR) \rightarrow MAR						1	1	1	1	1
			I \rightarrow R						1	1	1	1	1
	T_2	IND	M(MAR) \rightarrow MDR						1	1	1	1	1
			MDR \rightarrow Ad(IR)						1	1	1	1	1
EX (执行)	T_0		I \rightarrow EX						1	1	1	1	1
			Ad(IR) \rightarrow MAR						1	1	1		
			I \rightarrow R						1	1	1		
	T_1		1 \rightarrow W							1			
			M(MAR) \rightarrow MDR						1	1			
			AC \rightarrow MDR							1			
	T_2		(AC) + (MDR) \rightarrow AC							1			
			MDR \rightarrow M(MAR)								1		
			MDR \rightarrow AC									1	
			0 \rightarrow AC	1									
			$\overline{AC} \rightarrow AC$		1								
			L(AC) \rightarrow R(AC), AC ₀ 不变			1							
			$\rho^{-1}(AC)$				1						
			Ad(IR) \rightarrow PC									1	
	A_0		Ad(IR) \rightarrow PC										1
			0 \rightarrow G					1					

节拍安排





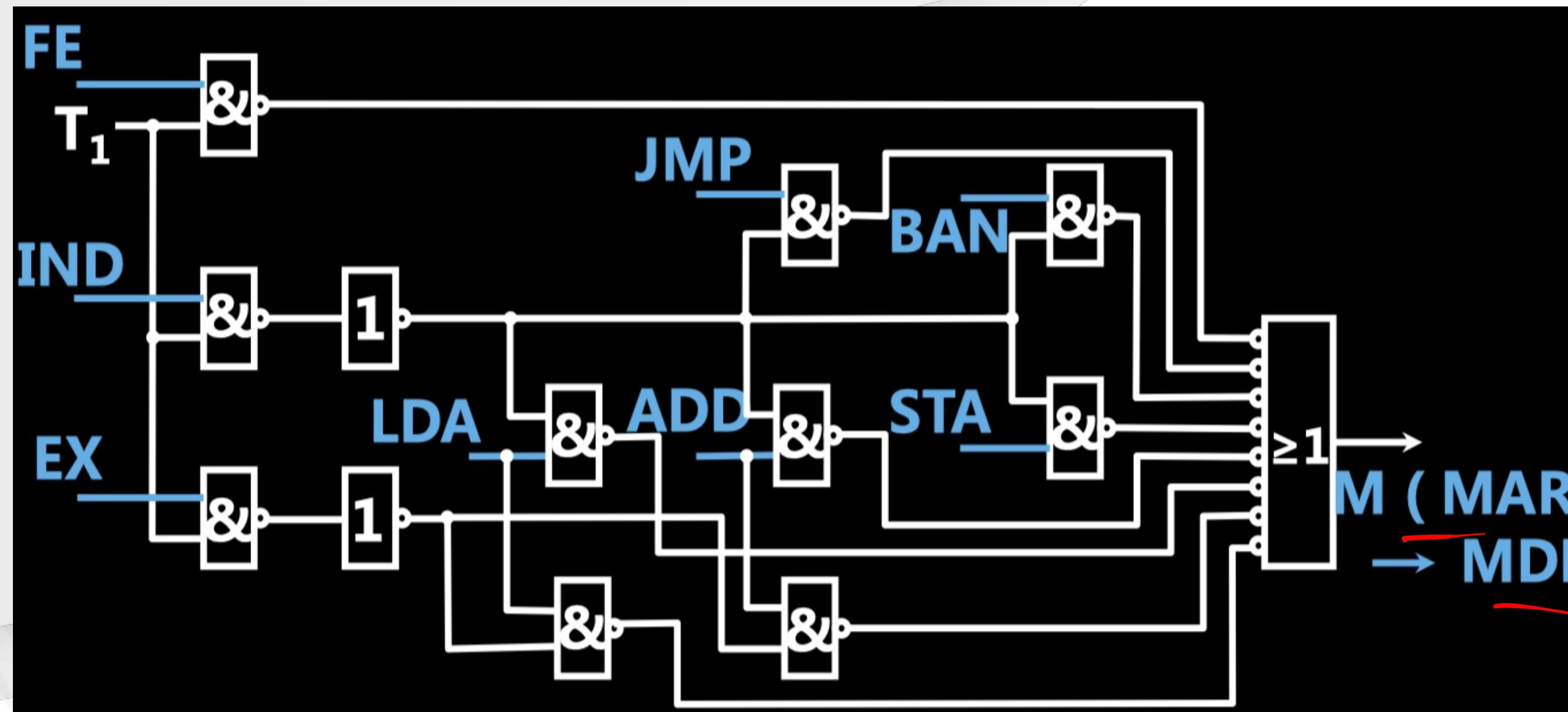
3. 指令系统设计与CPU运行控制

3.2.1 M (MAR) → MDR的硬布线控制器

$M(MAR) \rightarrow MDR$

$$= FE * T_1 + IND * T_1(ADD + STA + LDA + JMP + BAN) \\ + EX * T_1(ADD + LDA)$$

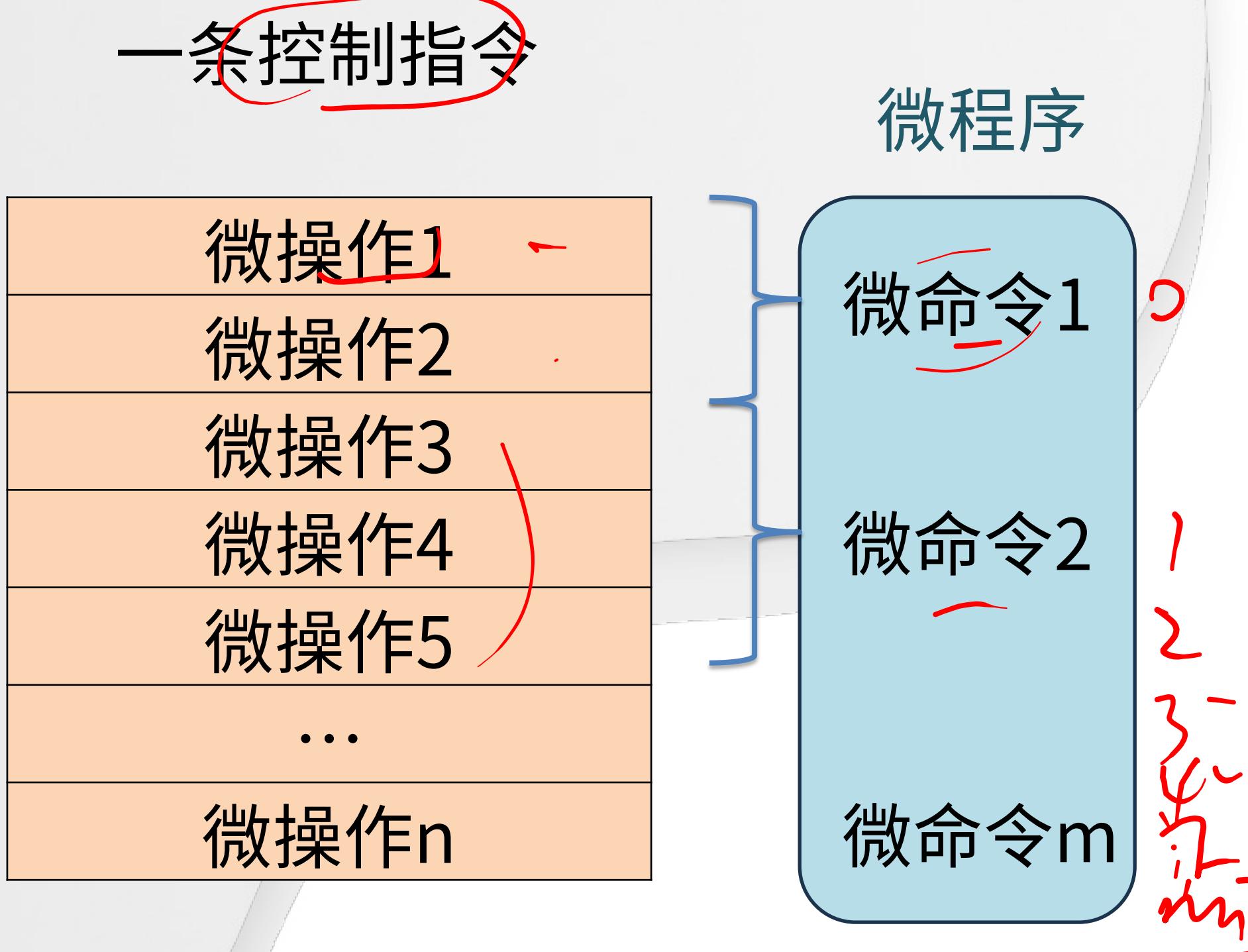
$$= T_1\{FE + IND(ADD + STA + LDA + JMP + BAN) + EX(ADD + LDA)\}$$





3. 指令系统设计与CPU运行控制

3.2.2 微程序控制器



微程序控制的概念：

- 1、一条微指令可分解为一系列的微操作
- 2、微操作是计算机中最基本的、不可分解的操作
- 3、微命令与微操作系列是一一对应
- 4、通过微命令发起微操作的执行过程
- 5、存放微程序的控制存储器的单元地址称微地址



3. 指令系统设计与CPU运行控制

3.2.2 微程序存储结构

一条控制指令

M	
M+1	
M+2	
P	
P+1	
P+2	
K	
K+1	
...	...

取指周期微程序

间址周期微程序

中断周期微程序

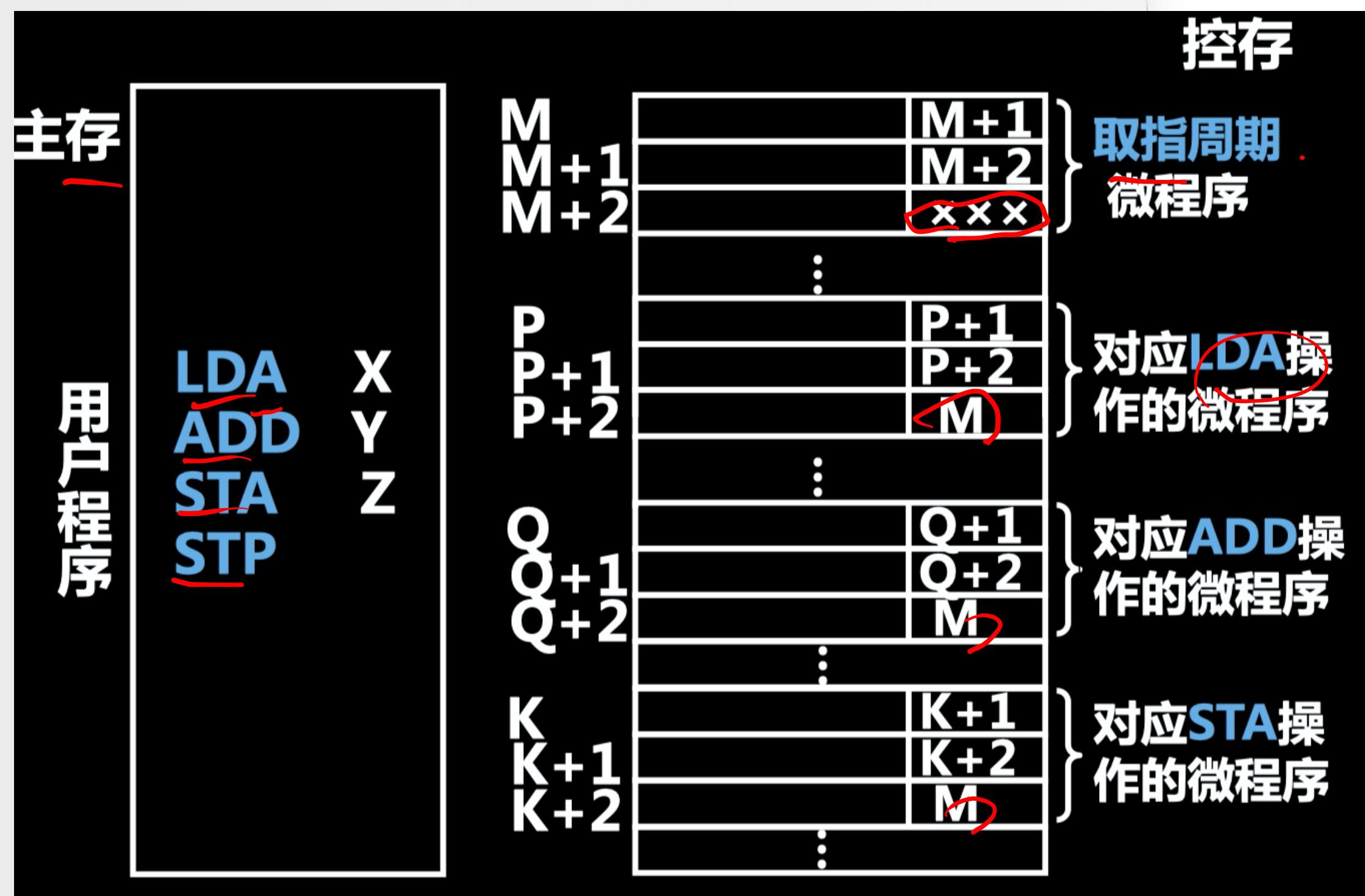
对应用LDA操作的微程序

CU



3. 指令系统设计与CPU运行控制

3.2.2 微程序存储结构



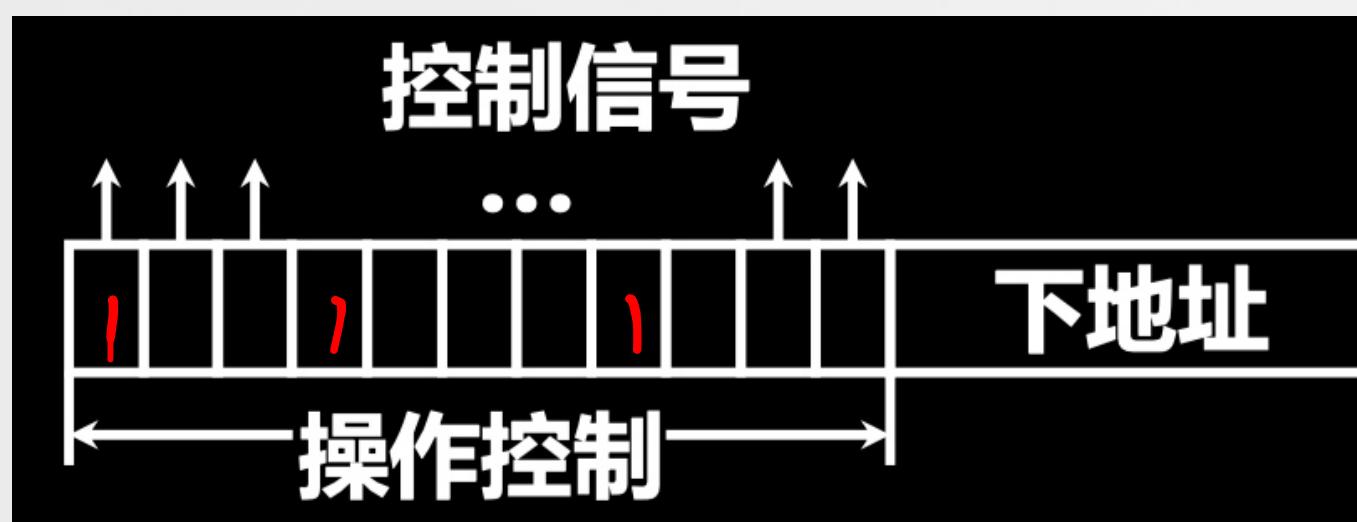


3. 指令系统设计与CPU运行控制

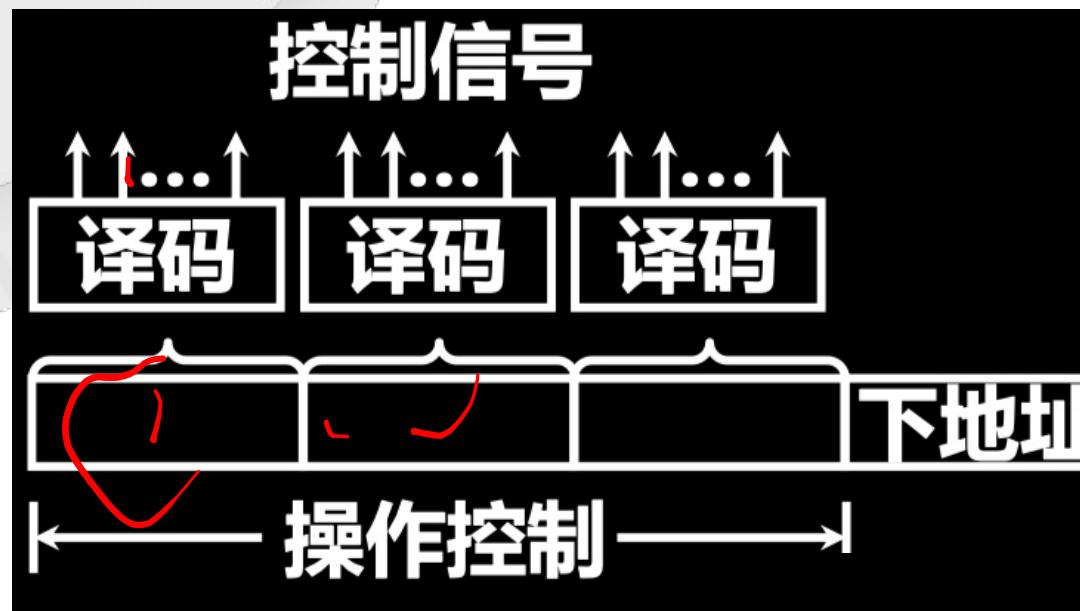
3.2.2 微指令编码方式



1、直接编码微指令：



2、字段直接编码微指令 ★：



- ① 互斥性微命令分在同一字段，相容性微命令在不同字段中。
- ② 每小段中信息不能多。
- ③ 每小段预留一个状态，表示不发出任何指令。



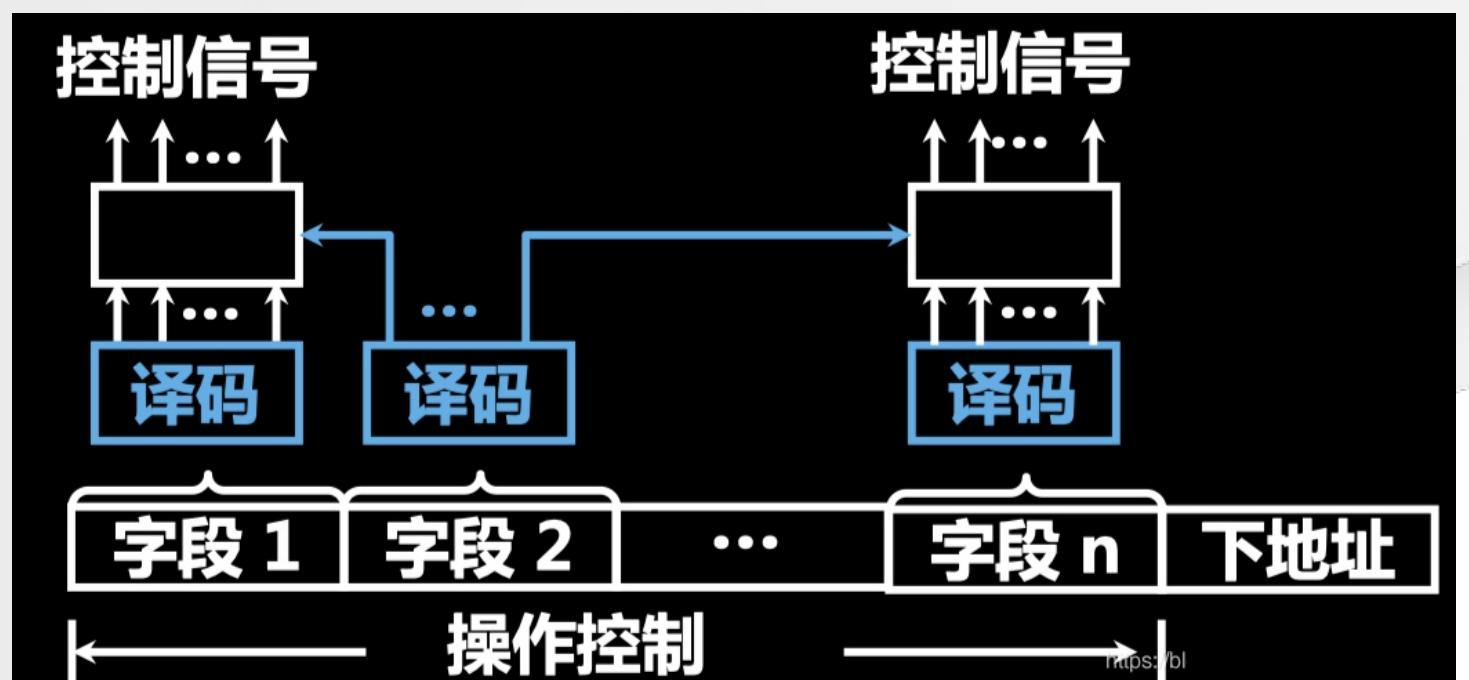
3. 指令系统设计与CPU运行控制

3.2.2 微指令编码方式

操作控制字段

顺序控制字段（下地址）

3、字段间接编码微指令：



4、混合编码微指令：



3. 指令系统设计与CPU运行控制

3.2.2 微指令格式

1、水平型：直接编码，字段直接编码，字段间接编码，混合编码

2、垂直型

微操作码	地址码		其他		微指令类别及功能		
	3~7	8~12	13~15				
0 1 2							
0 0 0	源寄存器	目的寄存器	其他控制	传送型微指令			
0 0 1	ALU 左输入	ALU 右输入	ALU	运算控制型微指令 按 ALU 字段所规定的功能执行,其结果送暂存器			
0 1 0	寄存器	移位次数	移位方式	移位控制型微指令 按移位方式对寄存器中的数据移位			
0 1 1	寄存器	存储器	读写	其他	访存微指令 完成存储器和寄存器之间的传送		
1 0 0	D		S	无条件转移微指令 D 为微指令的目的地址			
1 0 1	D	测试条件		条件转移微指令 最低 4 位为测试条件			
1 1 0				可定义 I/O 或其他操作			
1 1 1				第 3~15 位可根据需要定义各种微命令			

3、混合型



3. 指令系统设计与CPU运行控制

3.2.2 微指令格式比较

比较内容	水平型	垂直型
并行性	好	差
执行指令需要的 <u>微指令数</u>	少	多
和机器指令的相似度	差别大	相似
理解程度	无法直观看出	好理解

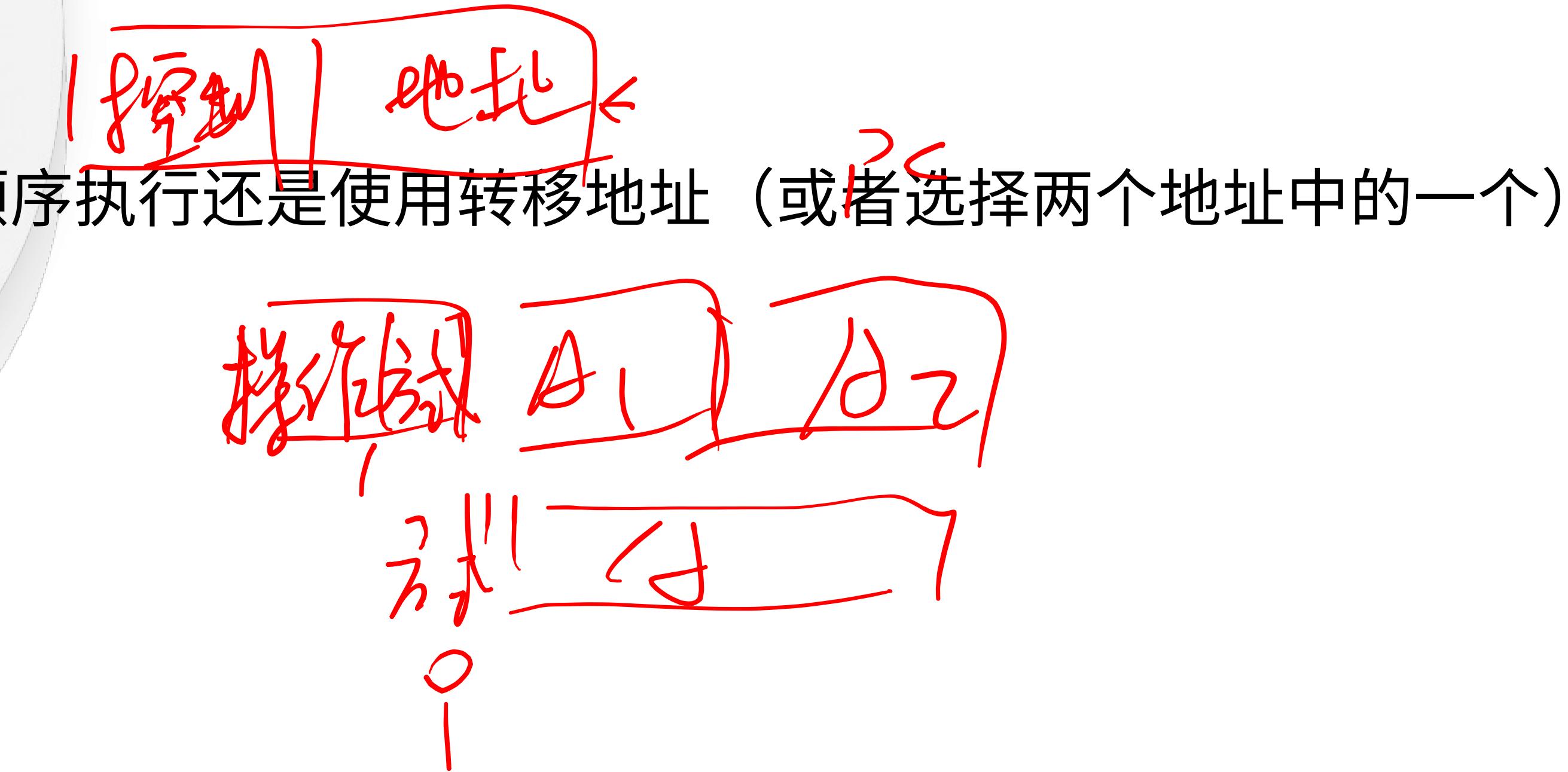


3. 指令系统设计与CPU运行控制

3.2.2 微指令序列地址的形成

1. 微指令的下地址字段指出
2. 增量计数器: $(CMAR) + 1 \rightarrow CMAR$
3. 分支转移: 根据转移方式判断是顺序执行还是使用转移地址 (或者选择两个地址中的一个)
4. 由硬件产生微程序的入口地址

操作控制字段	转移方式	转移地址
转移方式	指明判别条件	
转移地址	指明转移成功后的去向	





3. 指令系统设计与CPU运行控制

3.2.2 硬布线和微程序控制器比较

比较内容	微程序	硬布线
工作原理	微操作控制信号以 <u>微程序</u> 的形式存放在 <u>控制存储器</u> 中，执行指令时读出即可	微操作控制信号由组合逻辑电路根据当 <u>前的指令码、状态和时序</u> ，即时产生
执行速度	慢	快
规整性	很规整	烦琐、不规整
应用场合	CISC CPU	RISC CPU
易扩展性	易扩展	困难

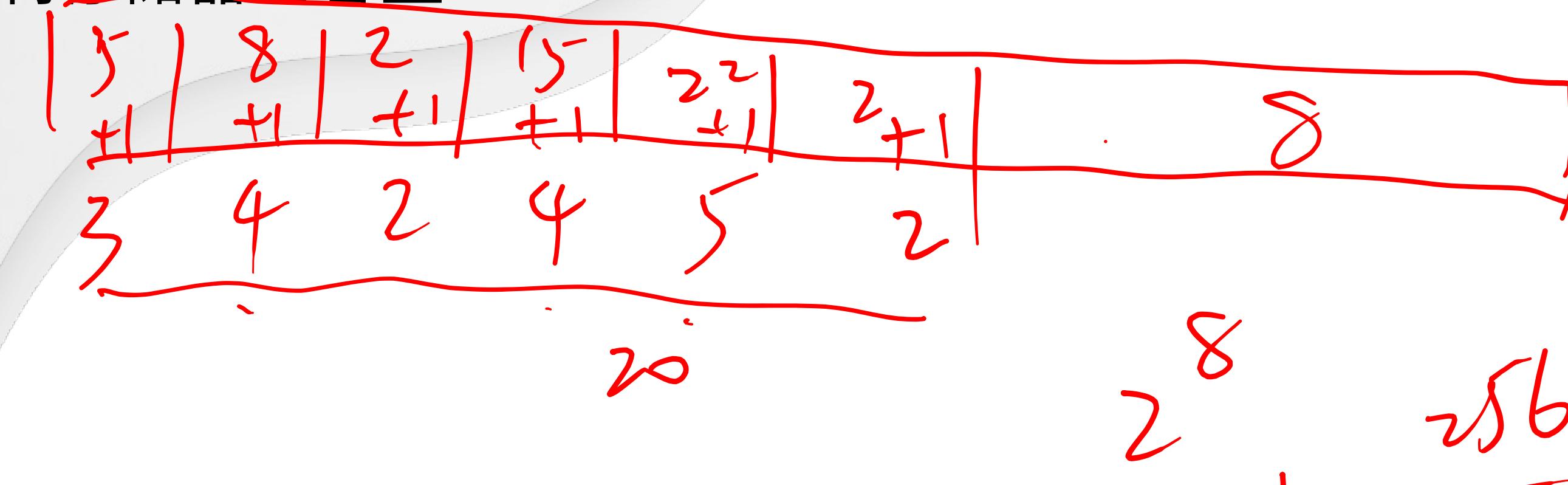


3. 指令系统设计与CPU运行控制

3.2.2 练习

例1：某计算机共有52个微操作控制信号，构成5个互斥类的微命令组，各组分别包含5, 8, 2, 15, 22个微命令。已知可判定的外部条件有两个，微指令字长为28位。

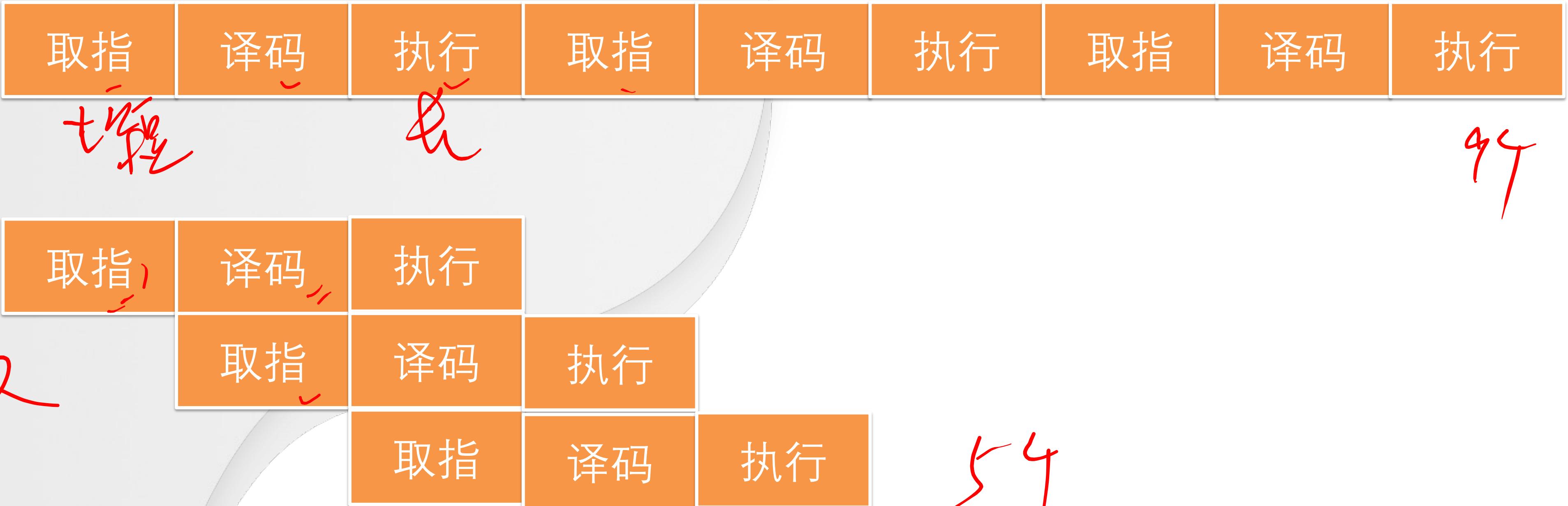
- 1) 按直接编码格式设计微指令，要求微指令的下地址字段直接给出后续微指令地址。
- 2) 指出控制存储器的容量





3. 指令系统设计与CPU运行控制

3.2.3 指令流水线



1、指令执行时间一般大于取指时间，因此取指可能要等待

2、当遇到条件转移指令，下一条指令要等执行完成后才知道

3、三级变四级更多级，速度会更快



3. 指令系统设计与CPU运行控制

3.2.3 六级流水线

取指FI 从存储器取出一条指令放入指令部件缓冲区

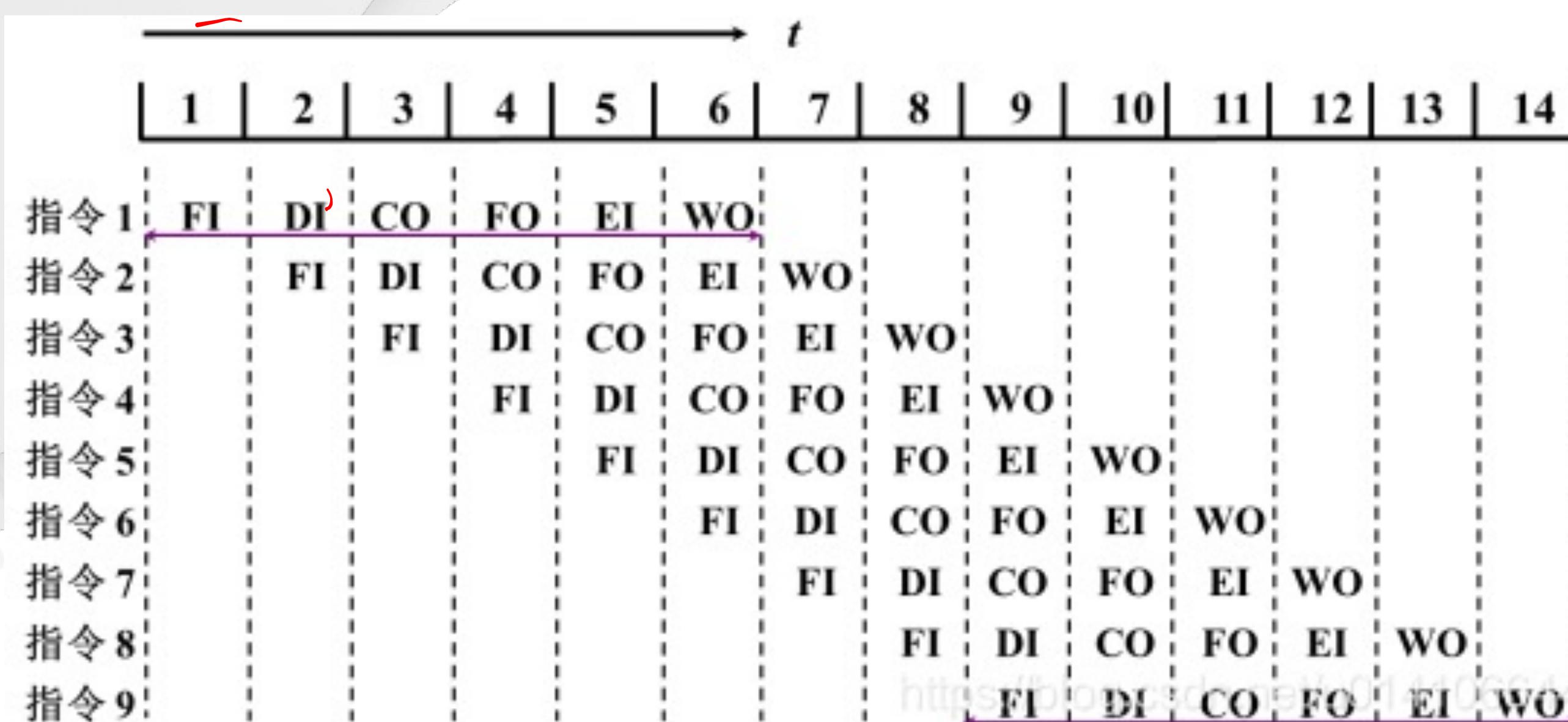
指令译码DI 确定操作性质和操作数地址形成方式

计算操作数地址CO 计算操作有效地址

取操作数FO 从存储器中取出操作数

执行指令EI 执行指令，将结果存入目的位置

写操作数WO 将结果存入寄存器



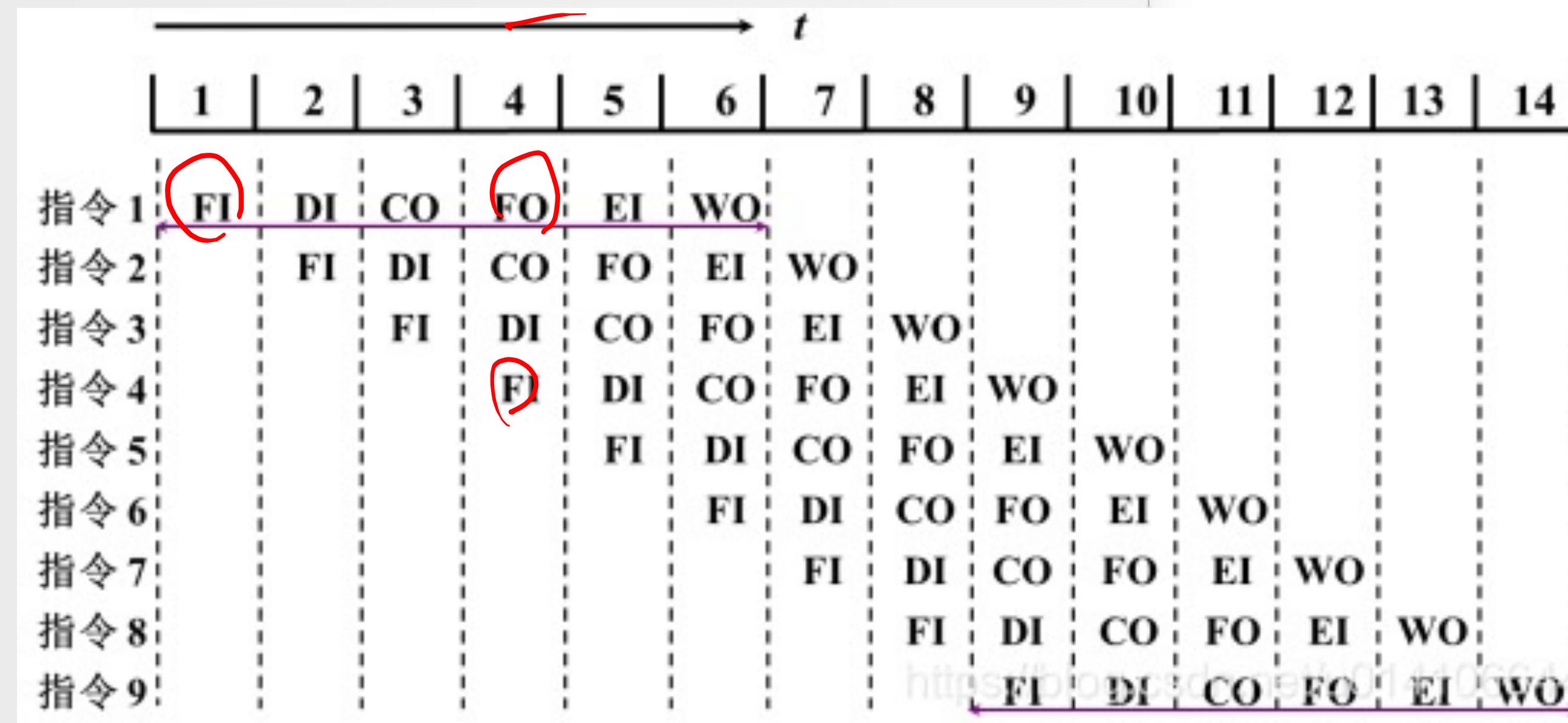
5F4



3. 指令系统设计与CPU运行控制

3.2.3 流水线执行的影响

1、结构相关（资源相关）



解决办法：延迟



3. 指令系统设计与CPU运行控制

3.2.3 流水线执行的影响

2、数据相关：~~④~~

写后读 (Read After Write, RAW)

读后写 (Write After Read, WAR)

写后写 (Write After Write, WAW)

解决办法：延迟，或使用数据旁路



3. 指令系统设计与CPU运行控制

3.2.3 练习

例2：判断下面3组指令各可能存在哪种类型的数据相关：

1) I₁ SUB R₁, R₂, R₃;

R₂+R₃ → R₁

I₂ ADD R₄, R₅, R₁;

R₁+R₁ → R₄

2) I₃ STA M, R₂;

Read

I₄ ADD R₂, R₄, R₅;

WAR

3) I₅ MUL R₃, R₂, R₁;

WAW

I₆ ADD R₃, R₄, R₅;



3. 指令系统设计与CPU运行控制

3.2.3 流水线执行的影响

3、控制相关：因为有转移指令

```
LDA #0
LDX #0
→ M ADD X, D      BNE 指令必须等
INX                  CPX 指令的结果
CPX #N              才能判断出
BNE M               是转移
DIV #N              还是顺序执行
STA ANS             https://blog.csdn.net/u014106644
```

解决办法：猜测法

尽早判别转移是否发生，尽早生成转移目标地址
预取转移成功和不成功两个方向的目标指令
加快和提前形成条件码



3. 指令系统设计与CPU运行控制



3.2.3 流水线性能

1、吞吐率

单位时间内流水线完成指令或输出结果数量

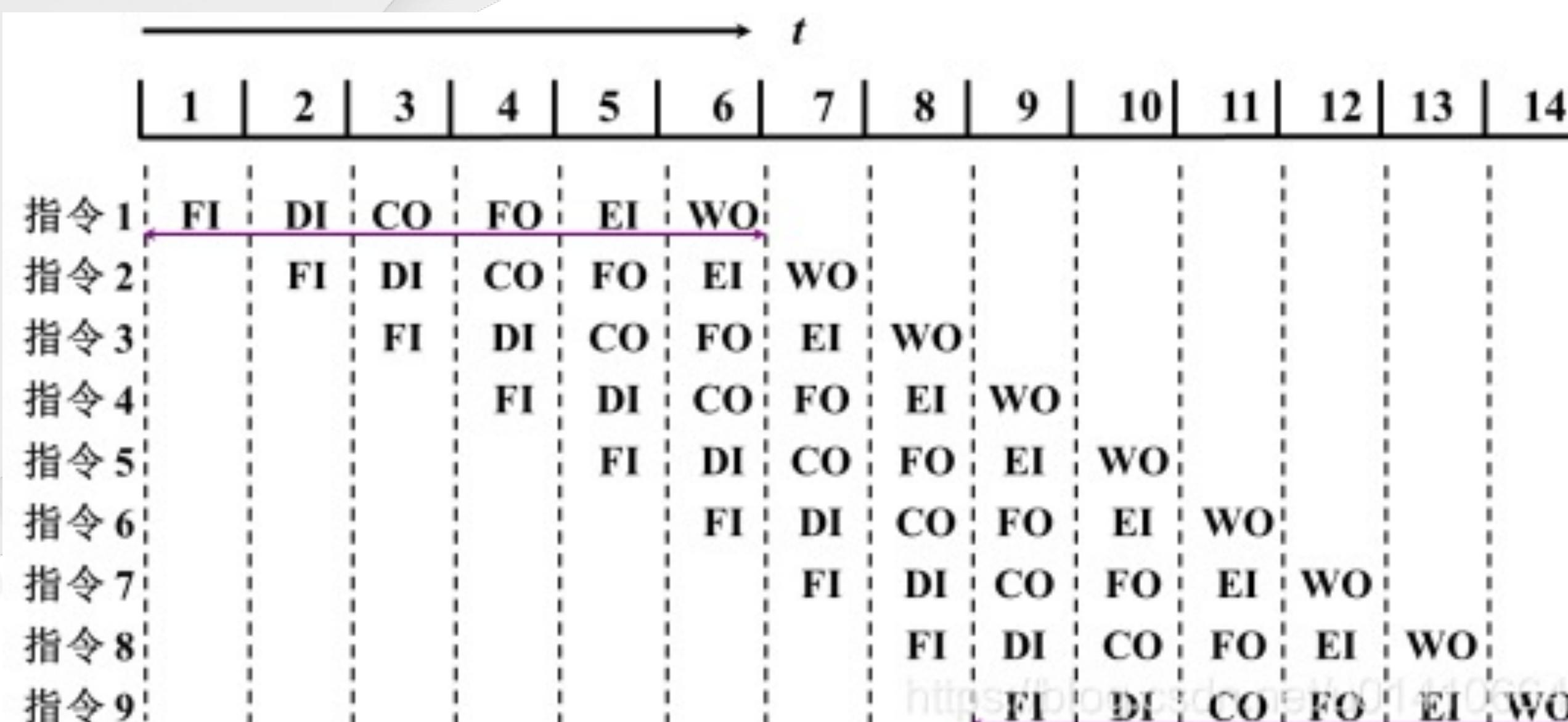
最大吞吐率：流水线在连续流动达到稳定状态后所得吞吐率

实际吞吐率：真实运行的吞吐率

$$6 + (9 - 1) = 14 \times$$

2、加速比

不使用流水线所用的时间与使用流水线所用的时间之比





3. 指令系统设计与CPU运行控制

3.2.3 练习



例3：假设指令流水线分取指、译码、执行、回写4个过程段，每个过程段都需要一个时钟周期来完成，现在需要执行10条指令，且连续输入此流水线。

- 1) 假设时钟周期为100ns，求流水线的实际吞吐率。
- 2) 求该流水线处理器的加速比。

$$\text{不申} \cdot 4 \times 10 = 40$$

(申)

13

$$40 / 13 \approx 3.08$$

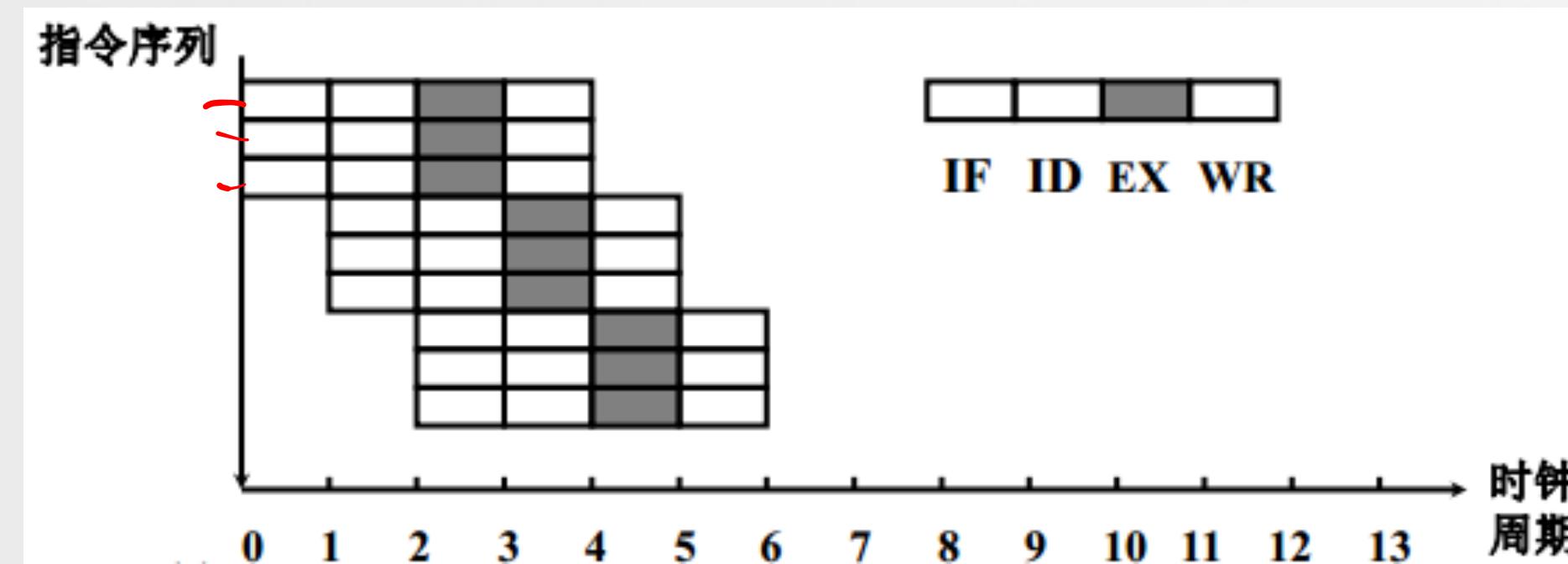
$$\frac{[4 + (10 - 1)] \times 100\text{ns}}{\text{约} 0.77 \times 10^7 \text{ 次}/\text{s}}$$



3. 指令系统设计与CPU运行控制

3.2.3 流水线多发技术

1、超标量技术



例如 三条指令是相互独立的，可以并行执行

MOV BL,8

ADD AX,1756H

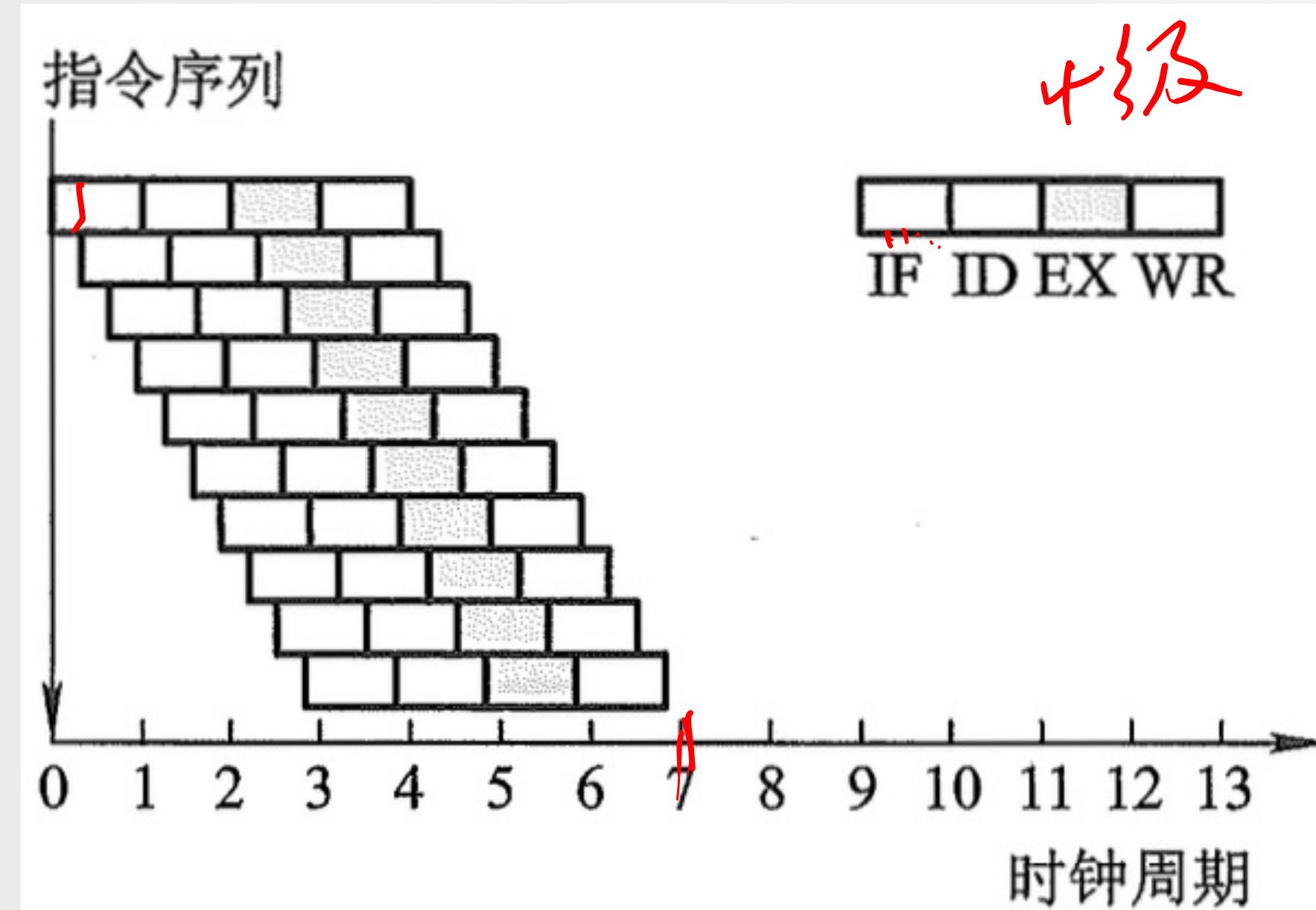
ADD CL,4EH



3. 指令系统设计与CPU运行控制

3.2.3 流水线多发技术

2、超流水线技术

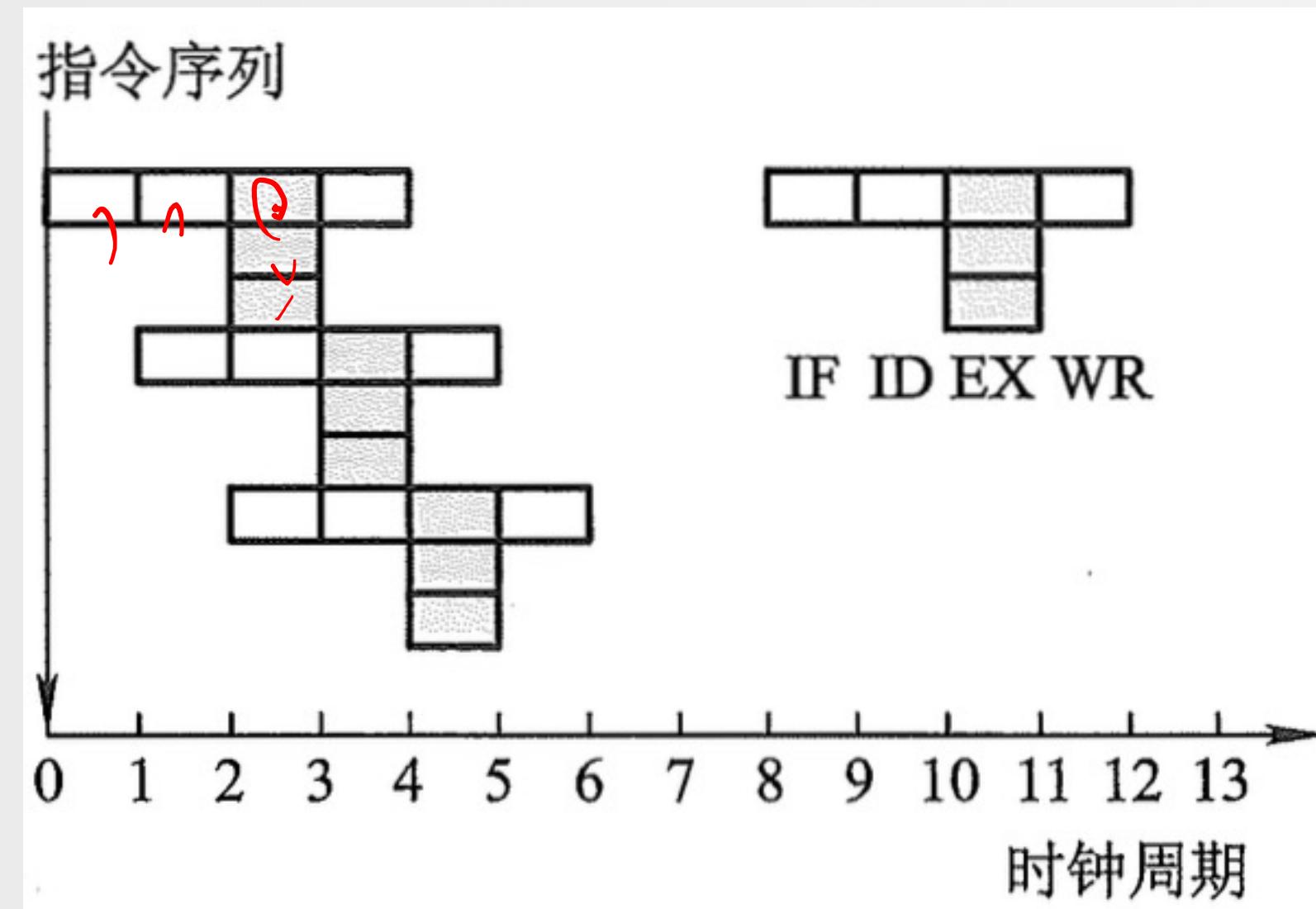




3. 指令系统设计与CPU运行控制

3.2.3 流水线多发技术

3、超长指令字技术





3. 指令系统设计与CPU运行控制

3.2.4 异常与中断

引起中断的中断源：

- 1、人为设置的中断：自愿中断、软中断
- 2、程序性事故：计算溢出，除0等
- 3、硬件故障中断
- 4、I/O设备中断请求：设备数据需要处理

x86, INT Type



3. 指令系统设计与CPU运行控制

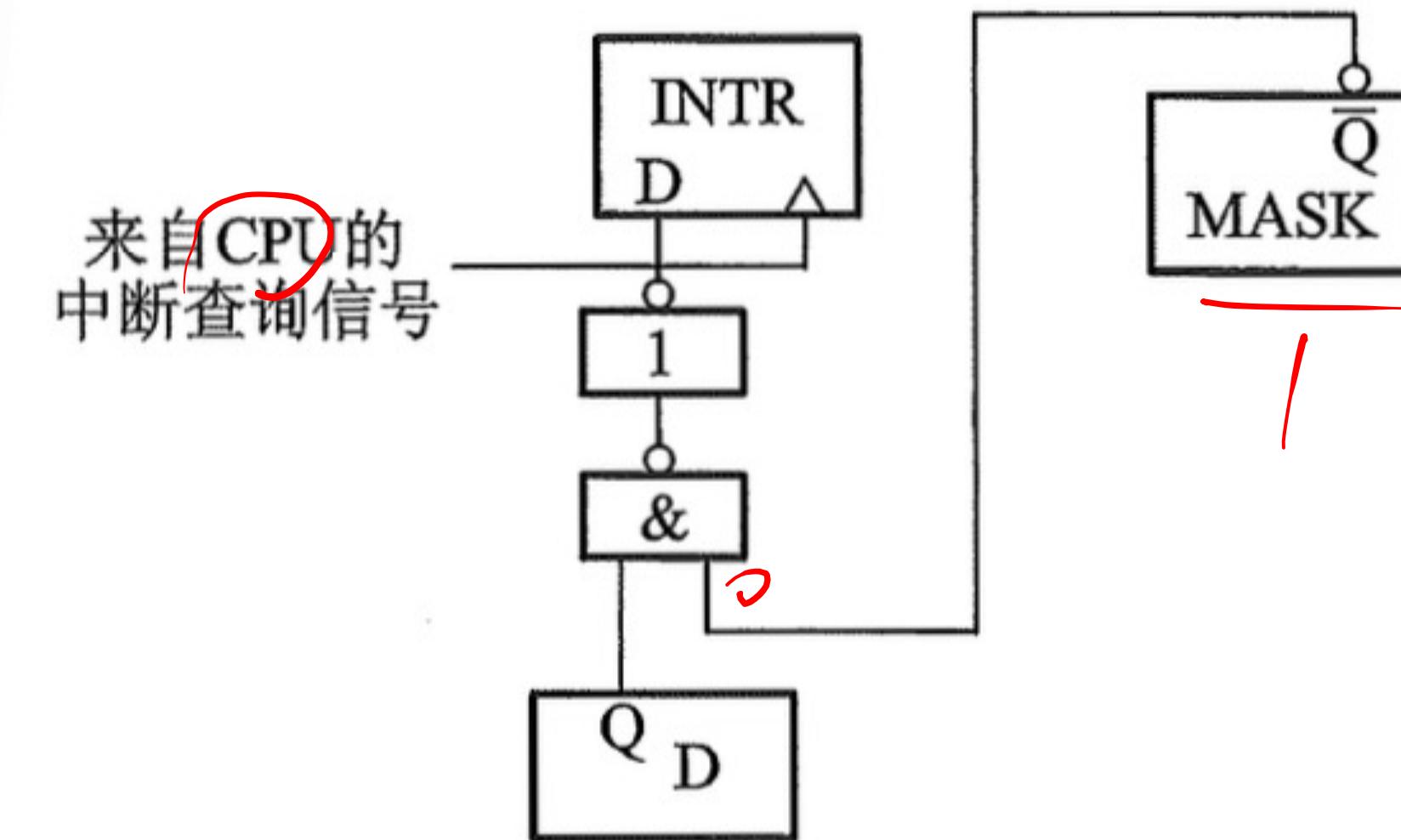
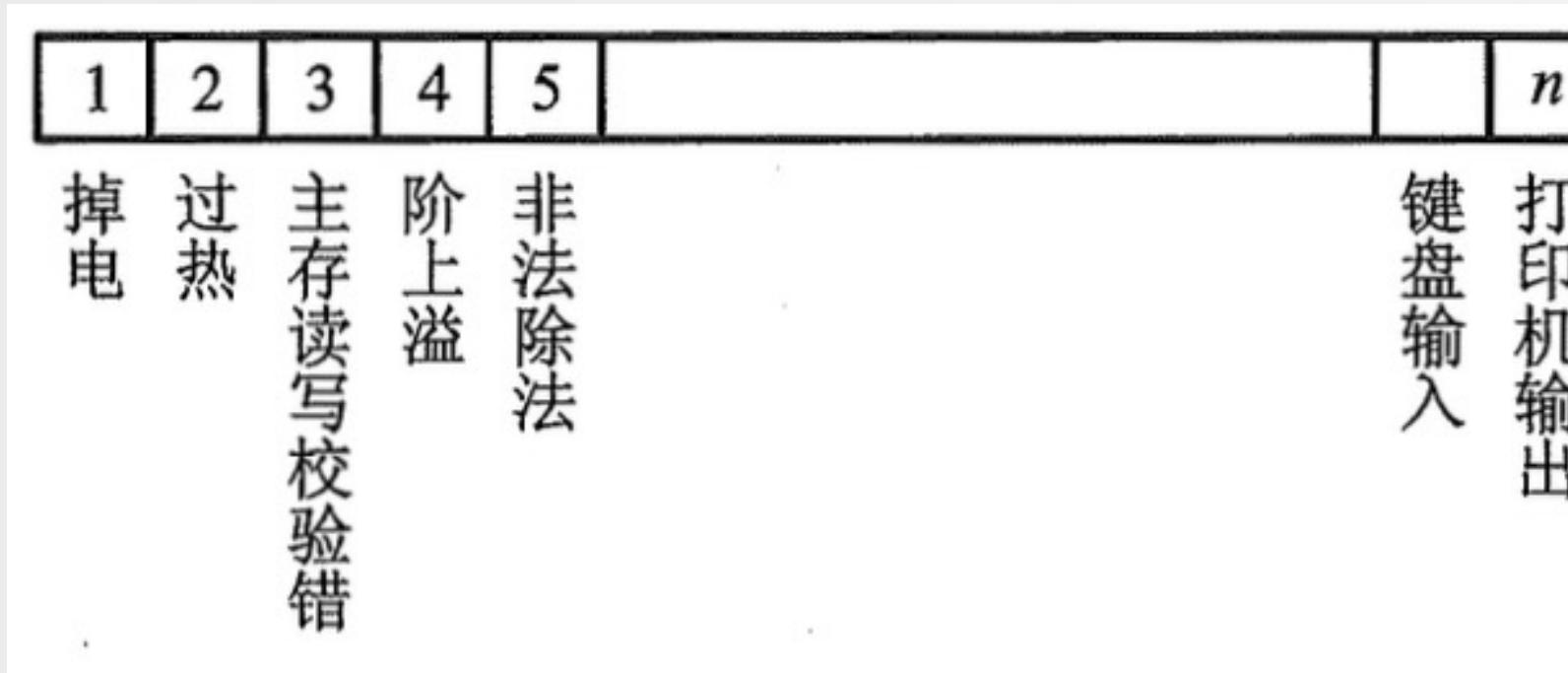
3.2.4 异常与中断

计算机中断处理必须解决的问题：

- 1、识别中断来源
- 2、多个中断同时请求时的处理优先级
- 3、CPU在什么条件、什么时间点、以什么方式响应
- 4、CPU响应中断后如何保护原程序的执行现场
- 5、CPU如何停止原程序的执行，而转入中断响应程序
- 6、中断处理结束后，如何恢复原程序的执行
- 7、在中断处理过程中又出现新的中断如何处理

3. 指令系统设计与CPU运行控制

3.2.4 识别中断来源

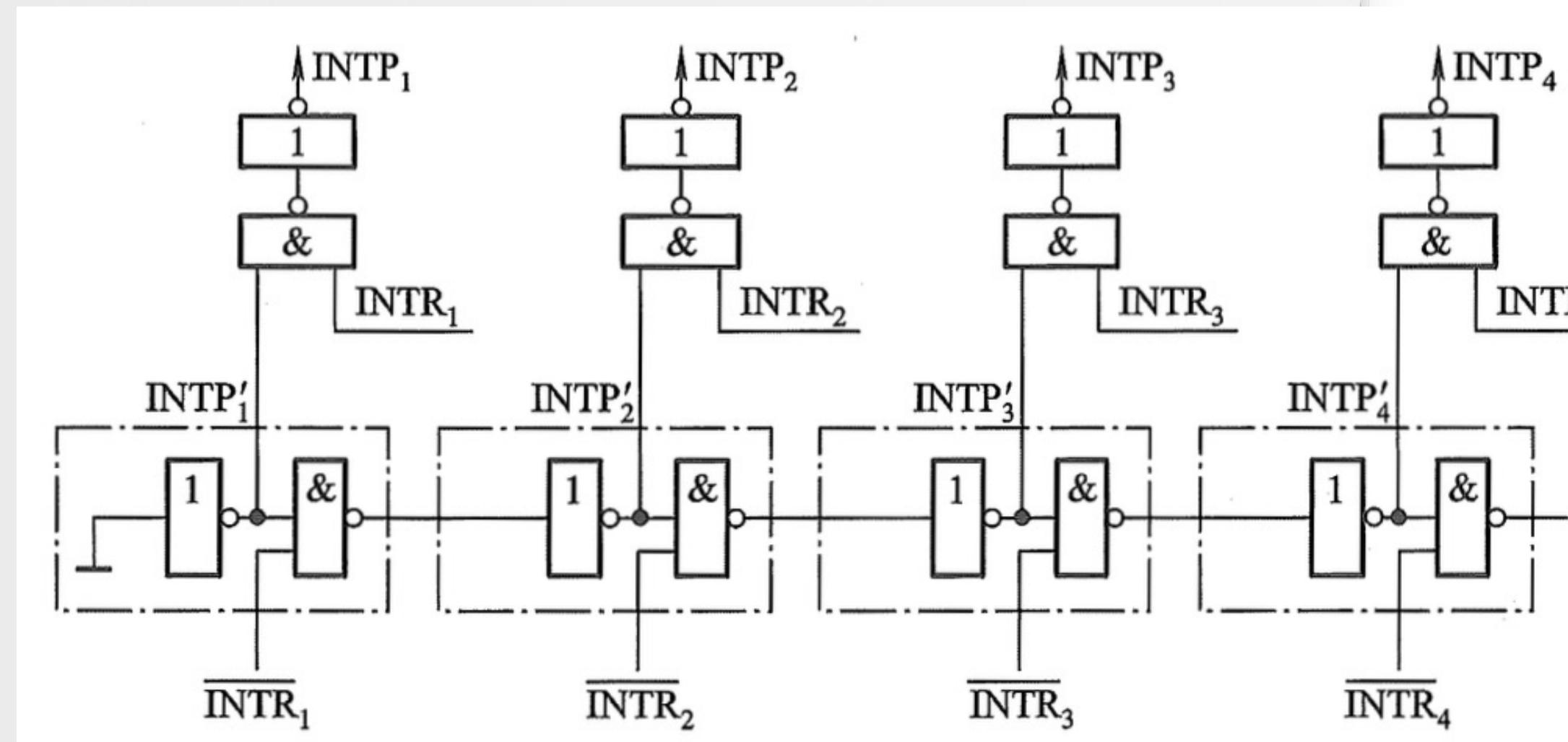




3. 指令系统设计与CPU运行控制

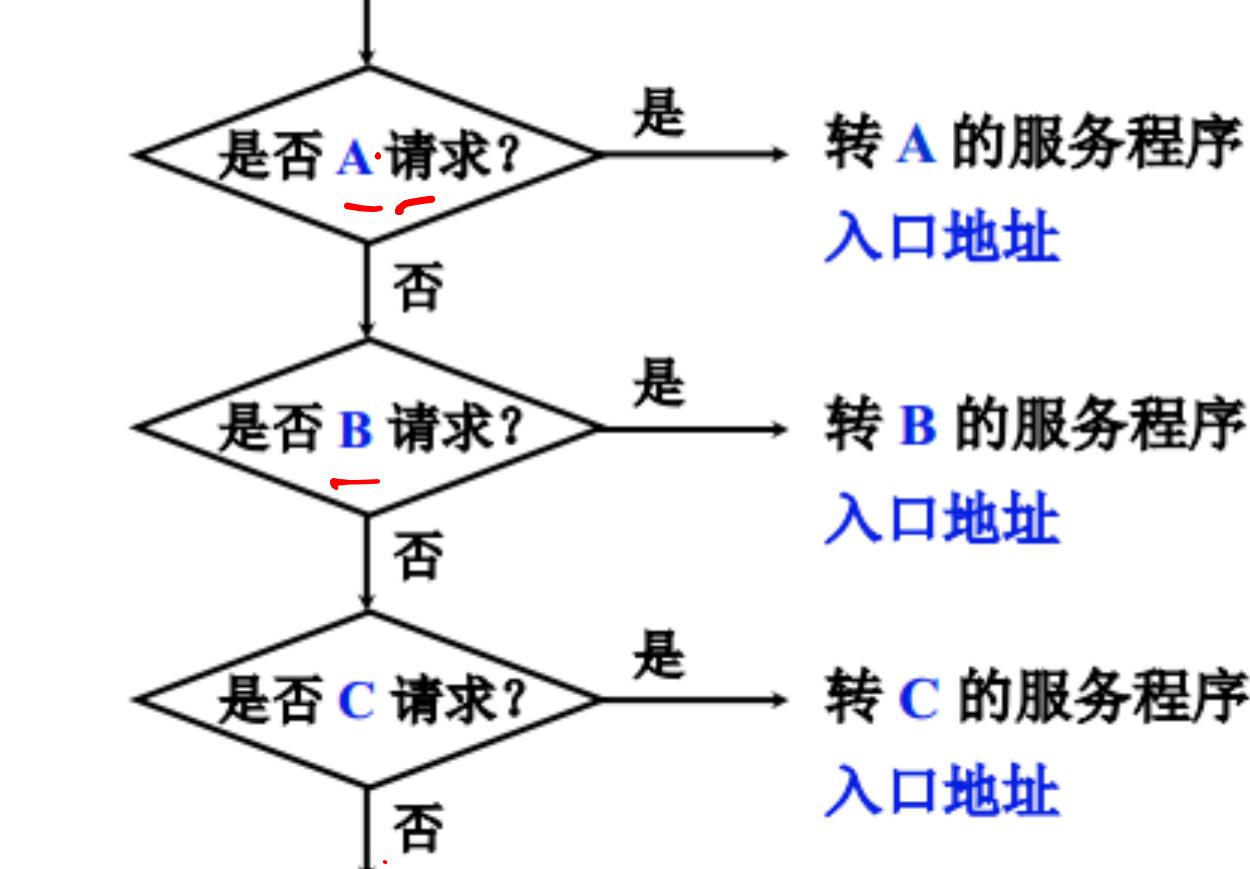
3.2.4 中断排队

硬件 (链式排队器)



软件 (查询与判断)

A、B、C 优先级按 降序 排列

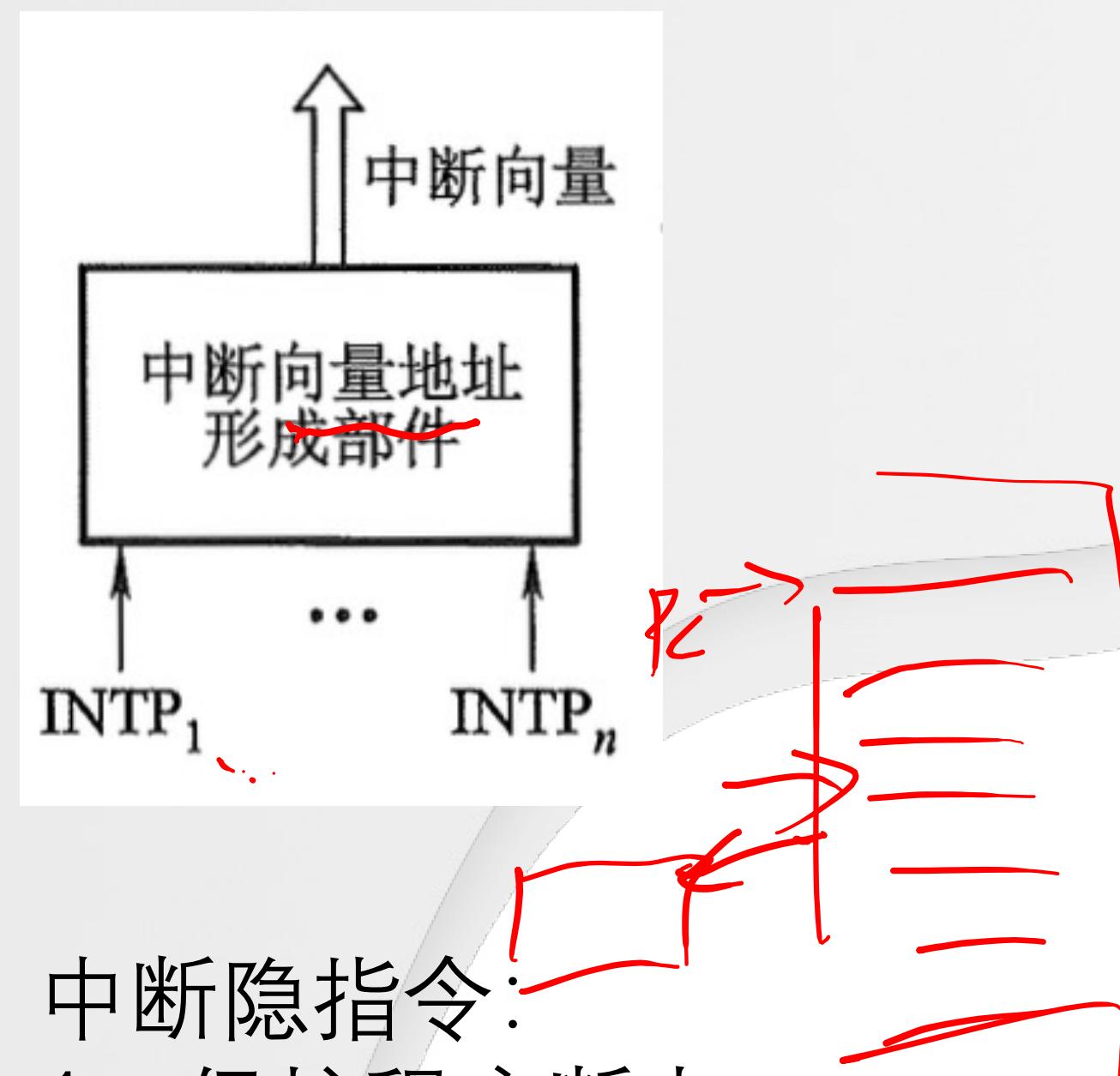




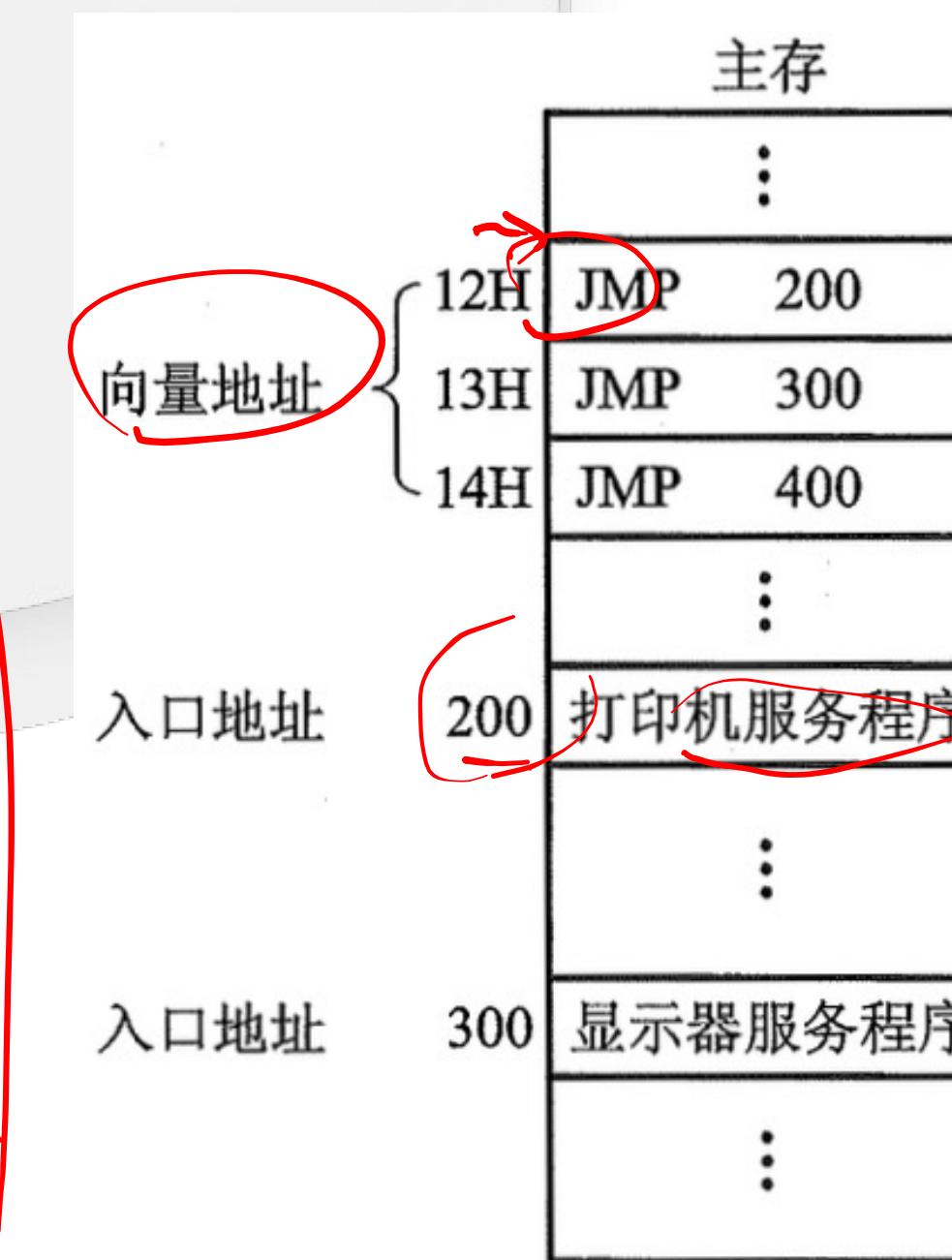
3. 指令系统设计与CPU运行控制

3.2.4 中断服务（隐指令）

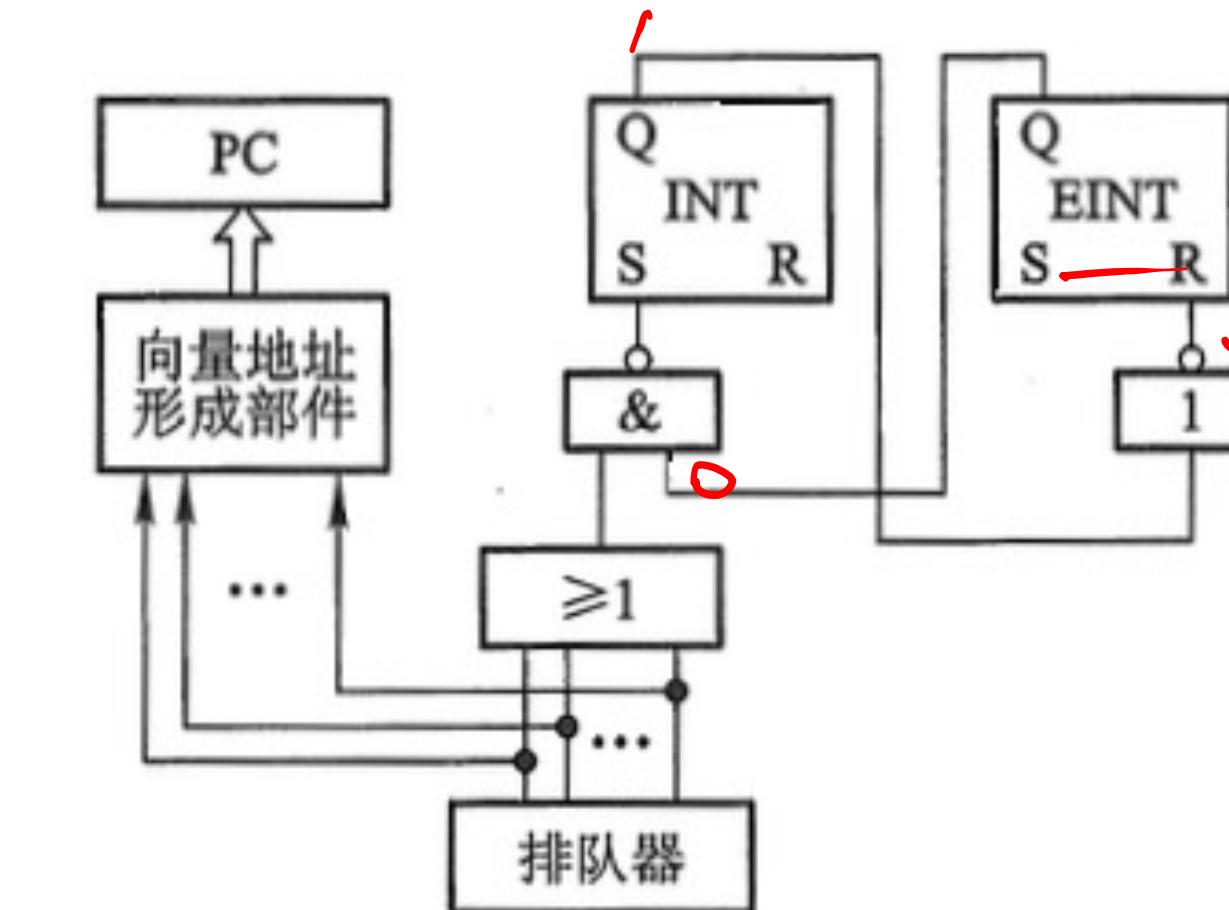
中断向量地址形成部件



向量地址



硬件关中断



中断隐指令：

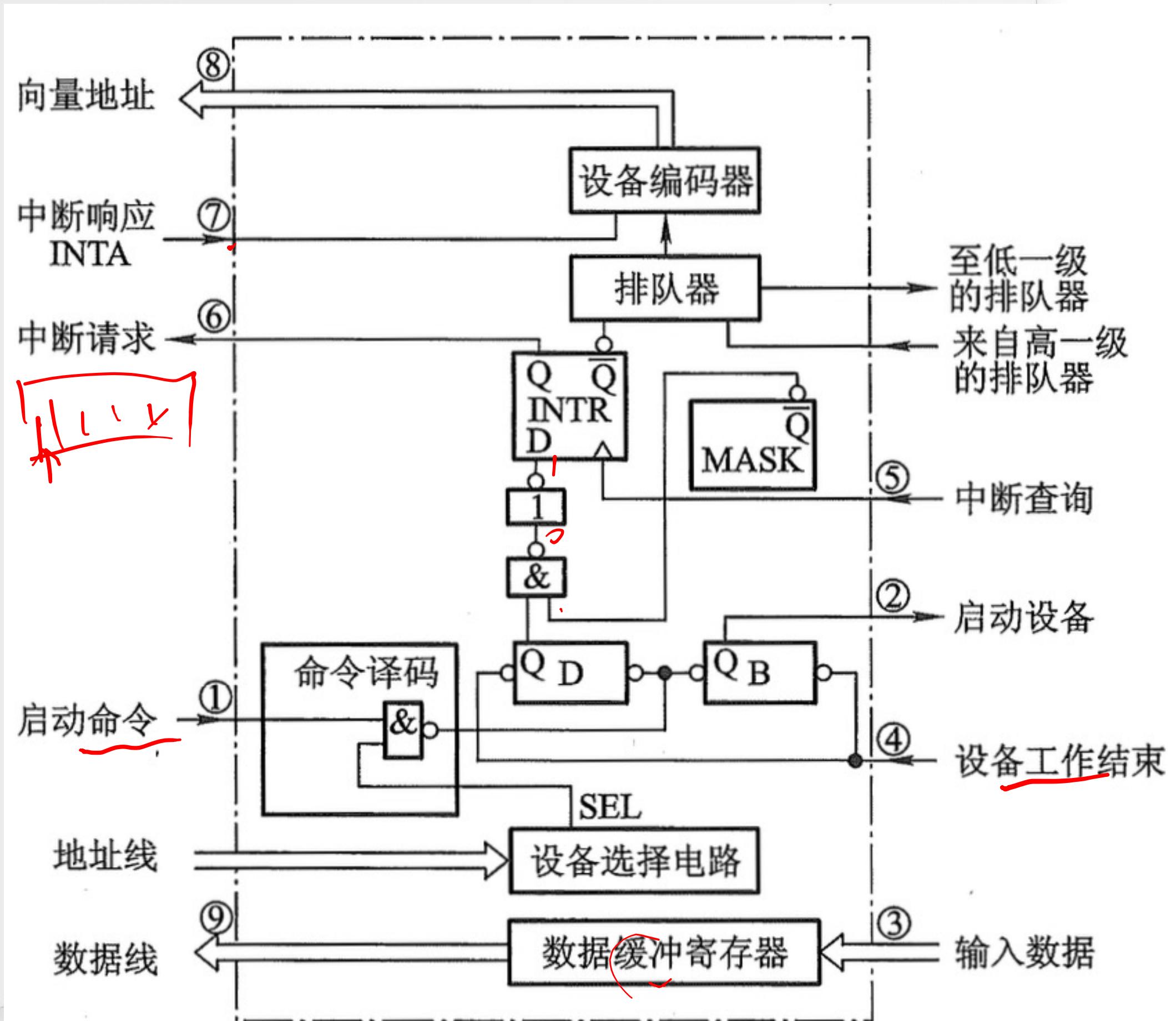
- 1、保护程序断点
- 2、寻找中断服务程序的入口地址
- 3、关中断

ARM、Un. 14.



3. 指令系统设计与CPU运行控制

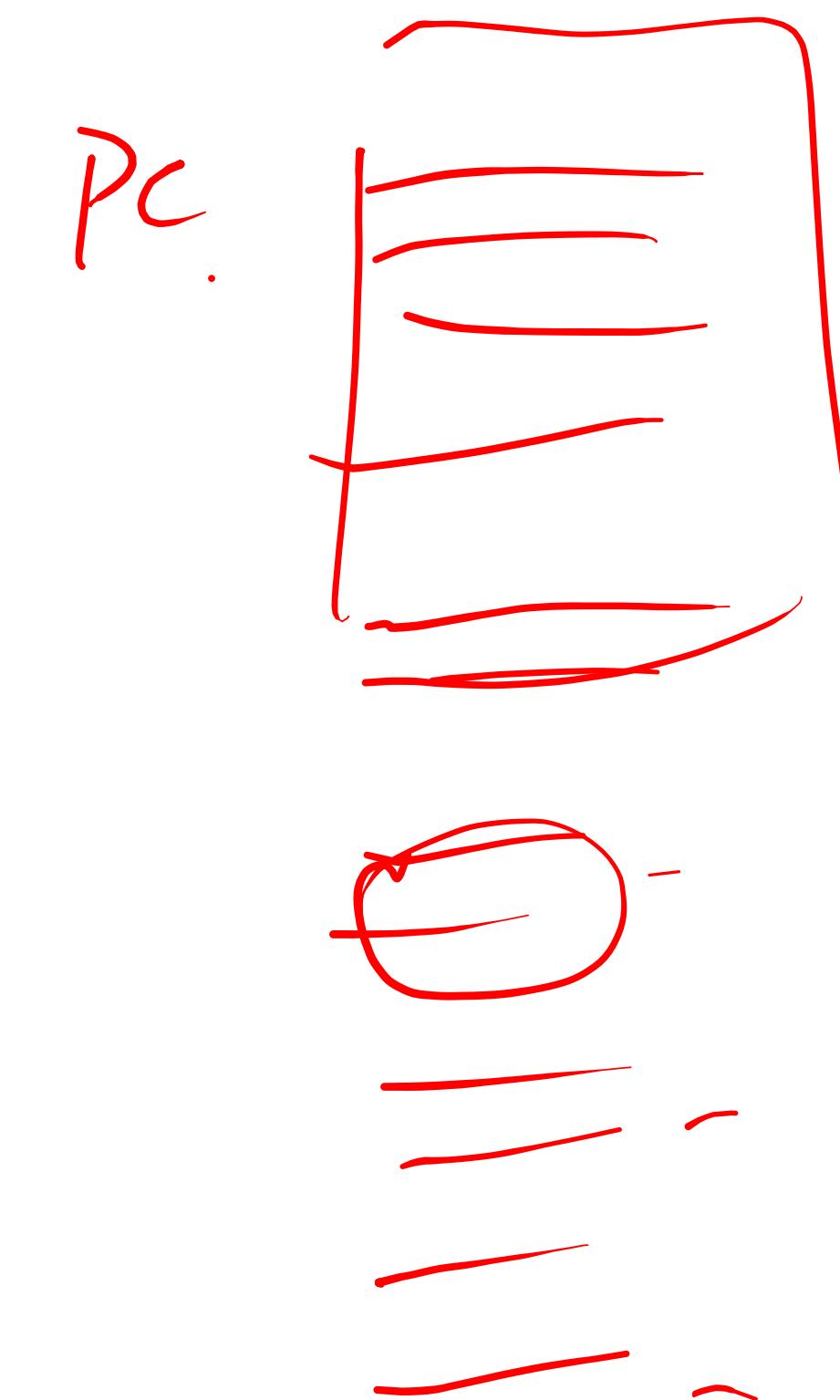
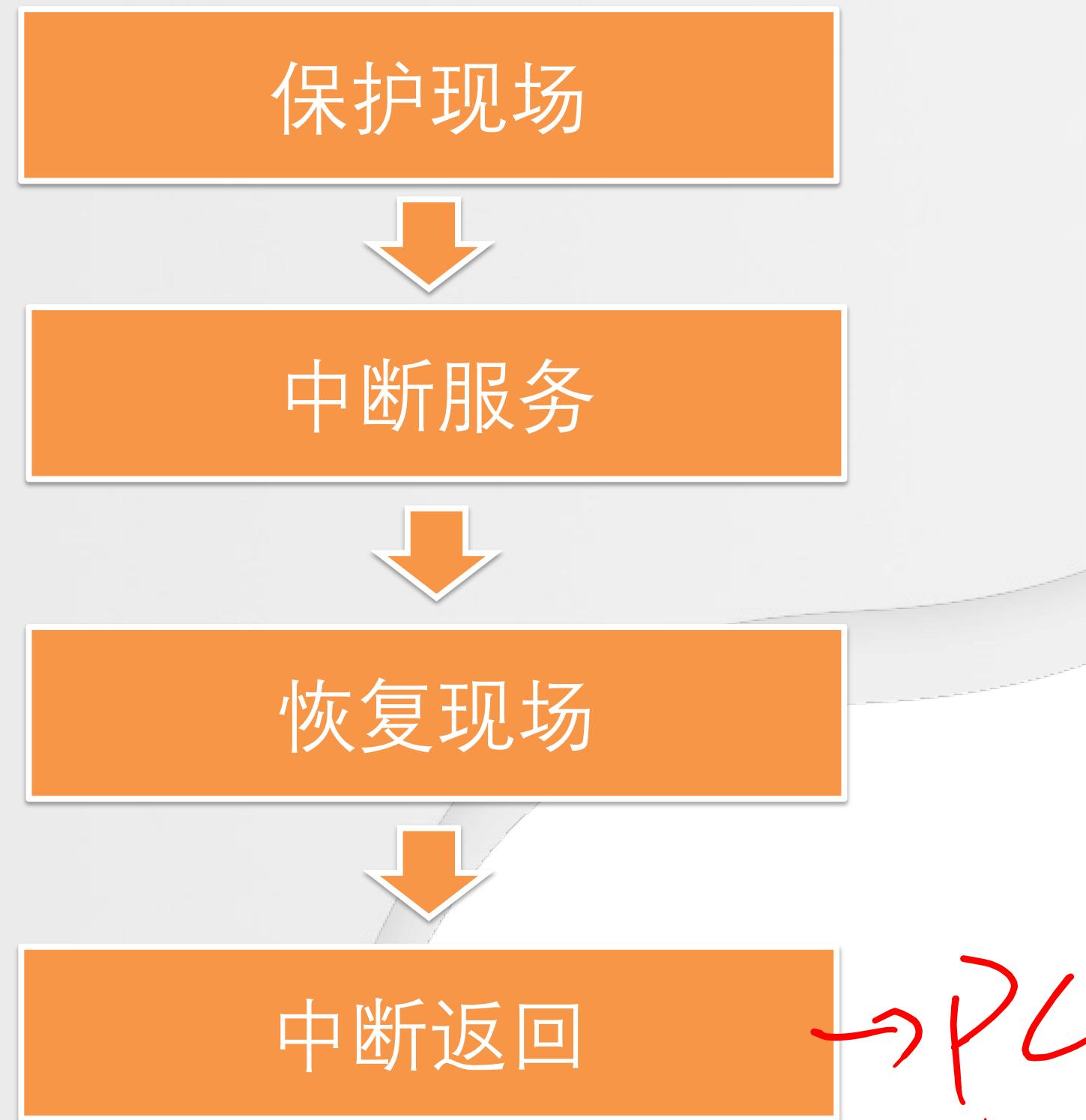
3.2.4 中断处理过程





3. 指令系统设计与CPU运行控制

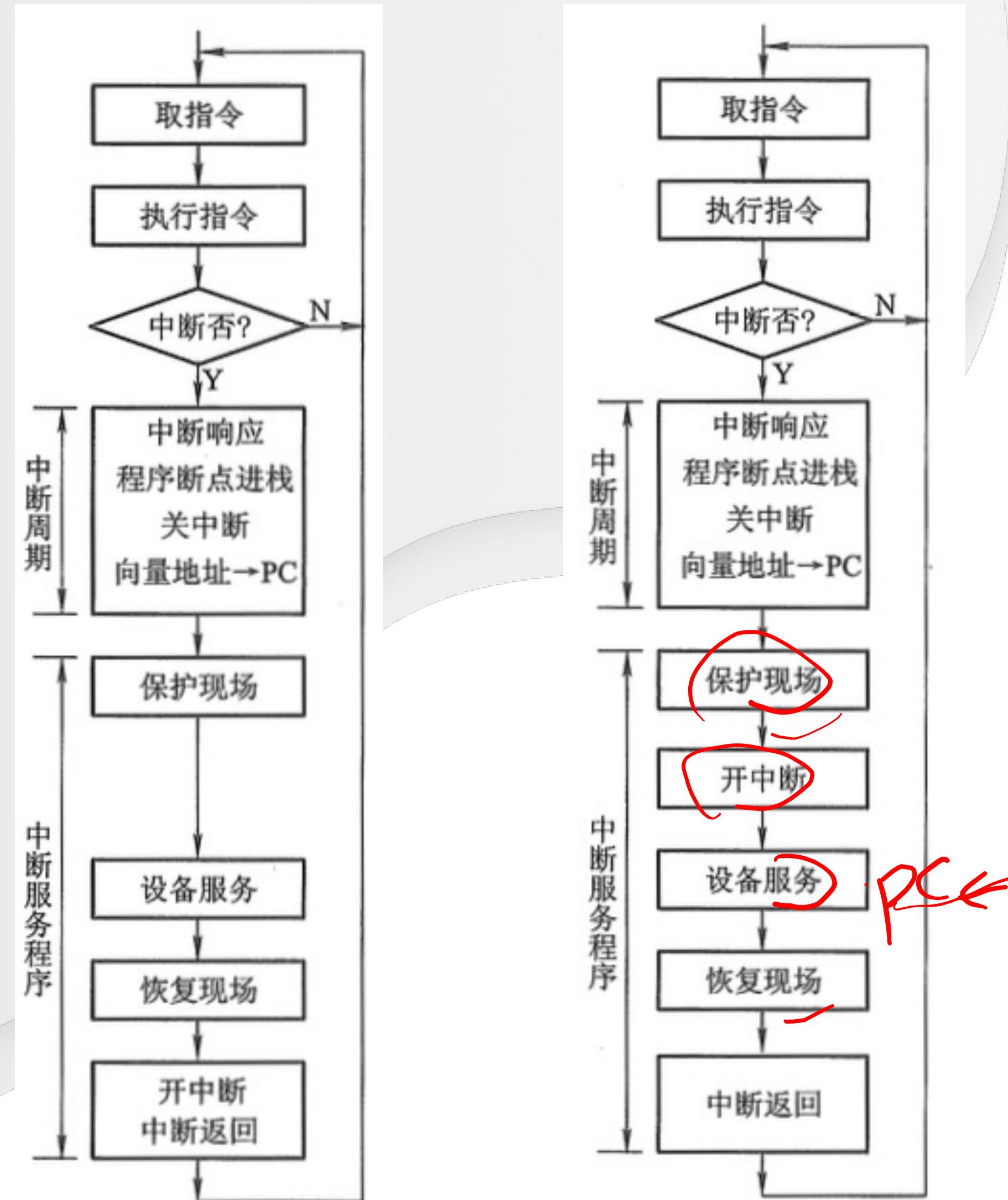
3.2.4 中断服务程序流程



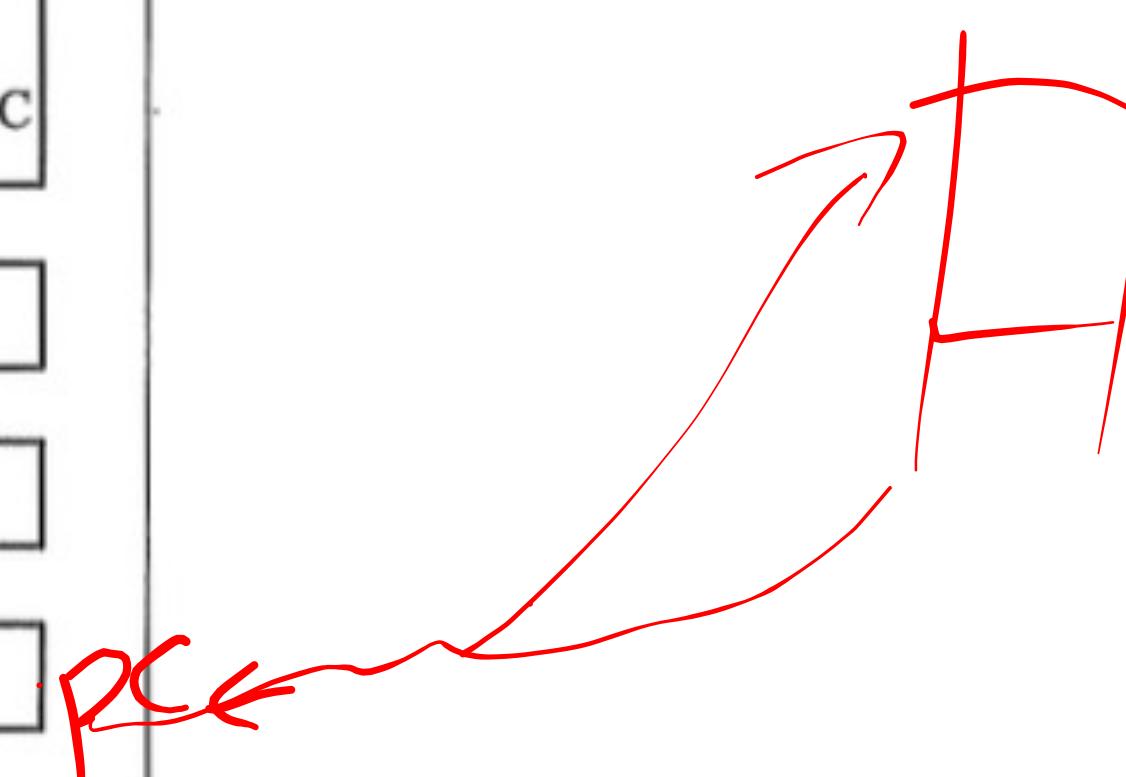


3. 指令系统设计与CPU运行控制

3.2.4 中断重入（嵌套）

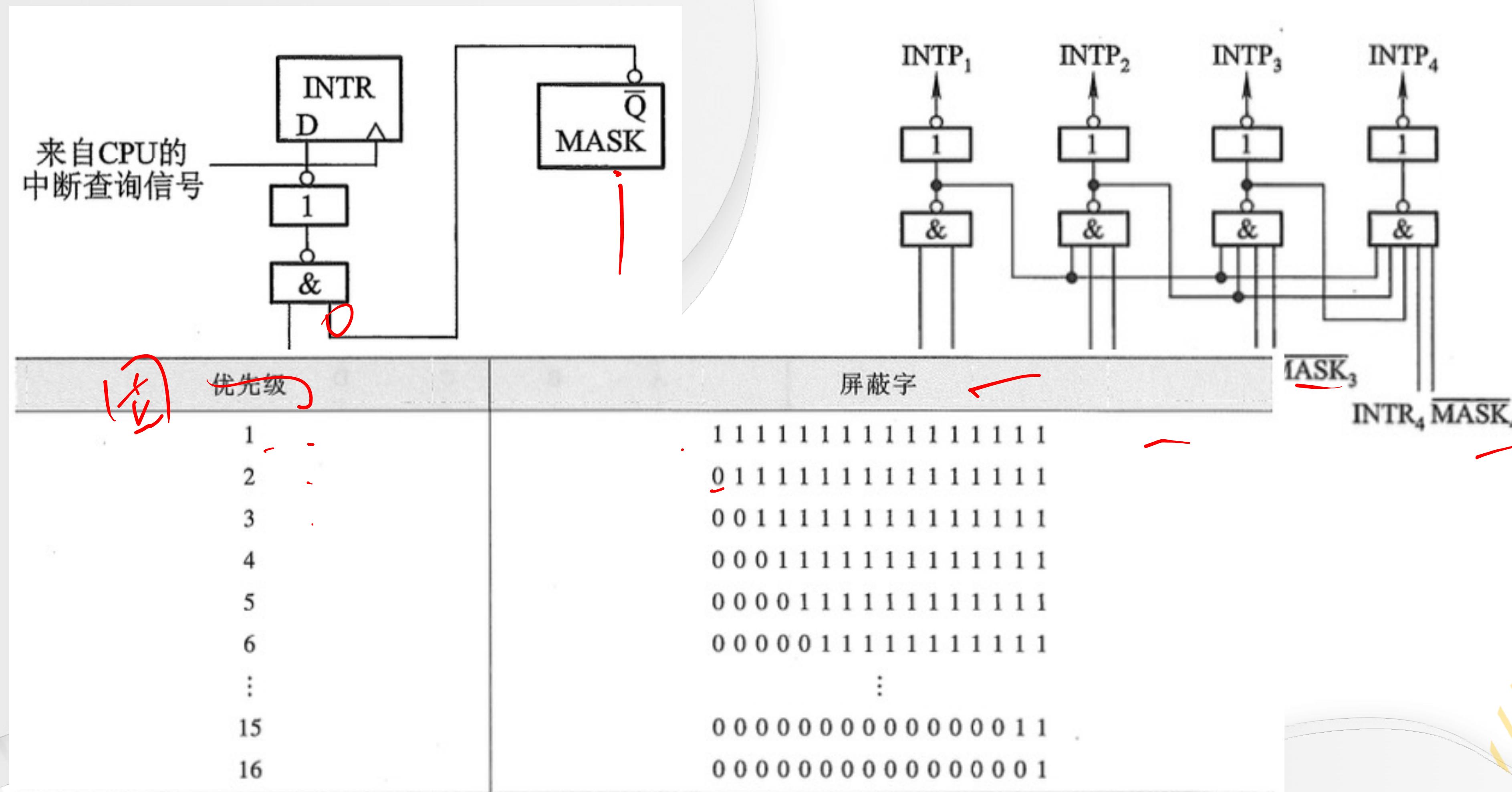


- 1、要产生重入，必须要在中断返回前开中断
- 2、更高优先级的中断产生请求



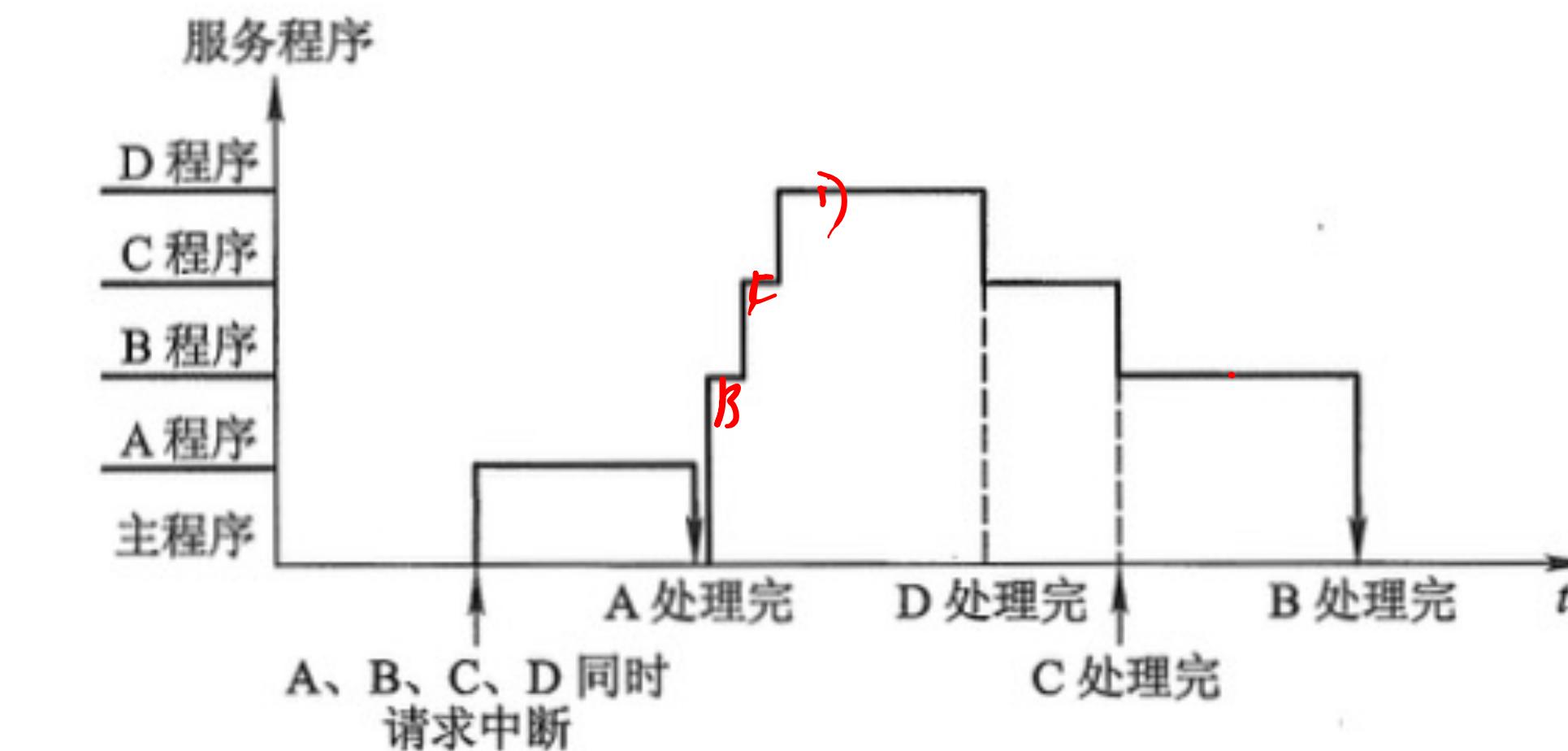
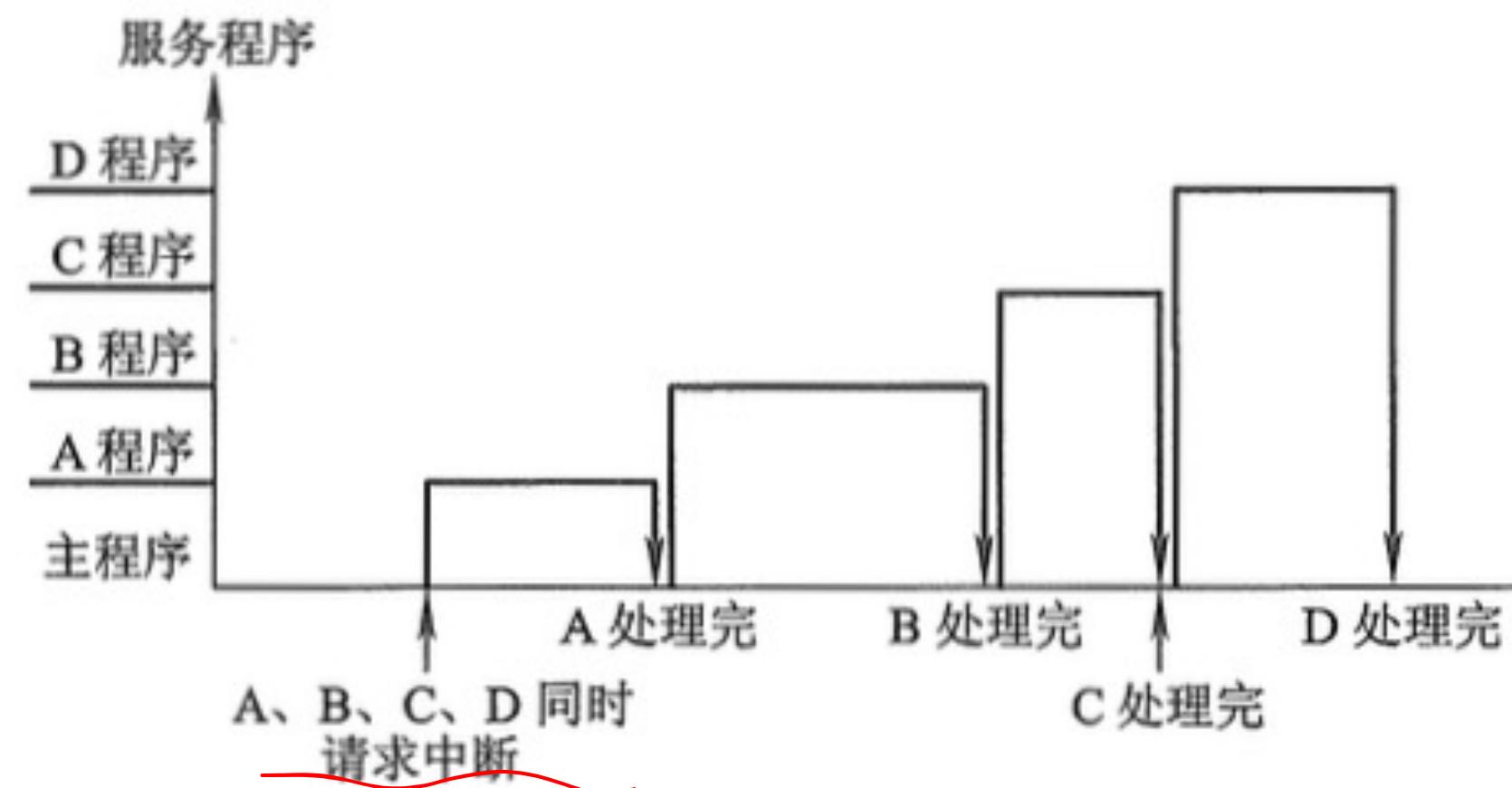
3. 指令系统设计与CPU运行控制

3.2.4 中断屏蔽



3. 指令系统设计与CPU运行控制

3.2.5 中断响应优先级与中断处理优先级

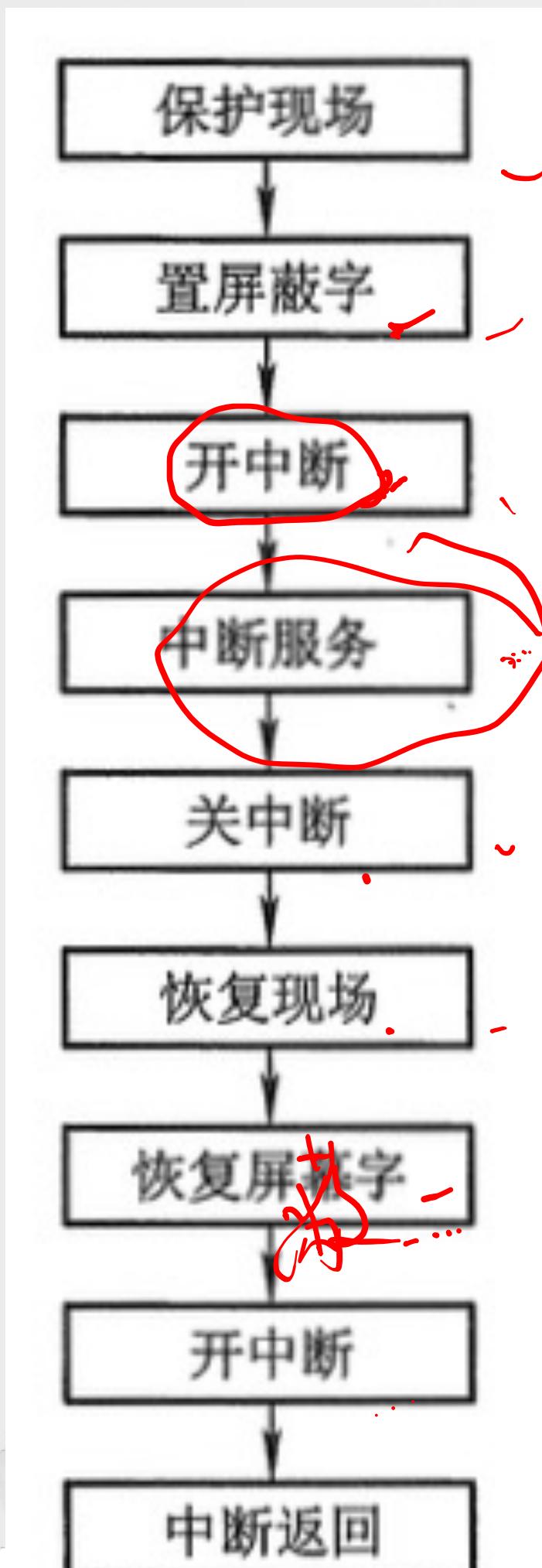


中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1



3. 指令系统设计与CPU运行控制

3.2.4 带屏蔽的中断服务程序





3. 指令系统设计与CPU运行控制

3.2.4 练习

例4 【2011年真题】：某计算机有5级中断L4~L0，中断屏蔽字为M4~M0，若中断响应优先级从高到低的顺序是L0→L1→L2→L3→L4，且要求中断处理优先级从高到低的顺序是L4→L0→L2→L1→L3，则L1的中断处理程序中设置的中断屏蔽字是（ ）。

A. 11110

B. 01101

C. 00011

D. 01010





3. 指令系统设计与CPU运行控制

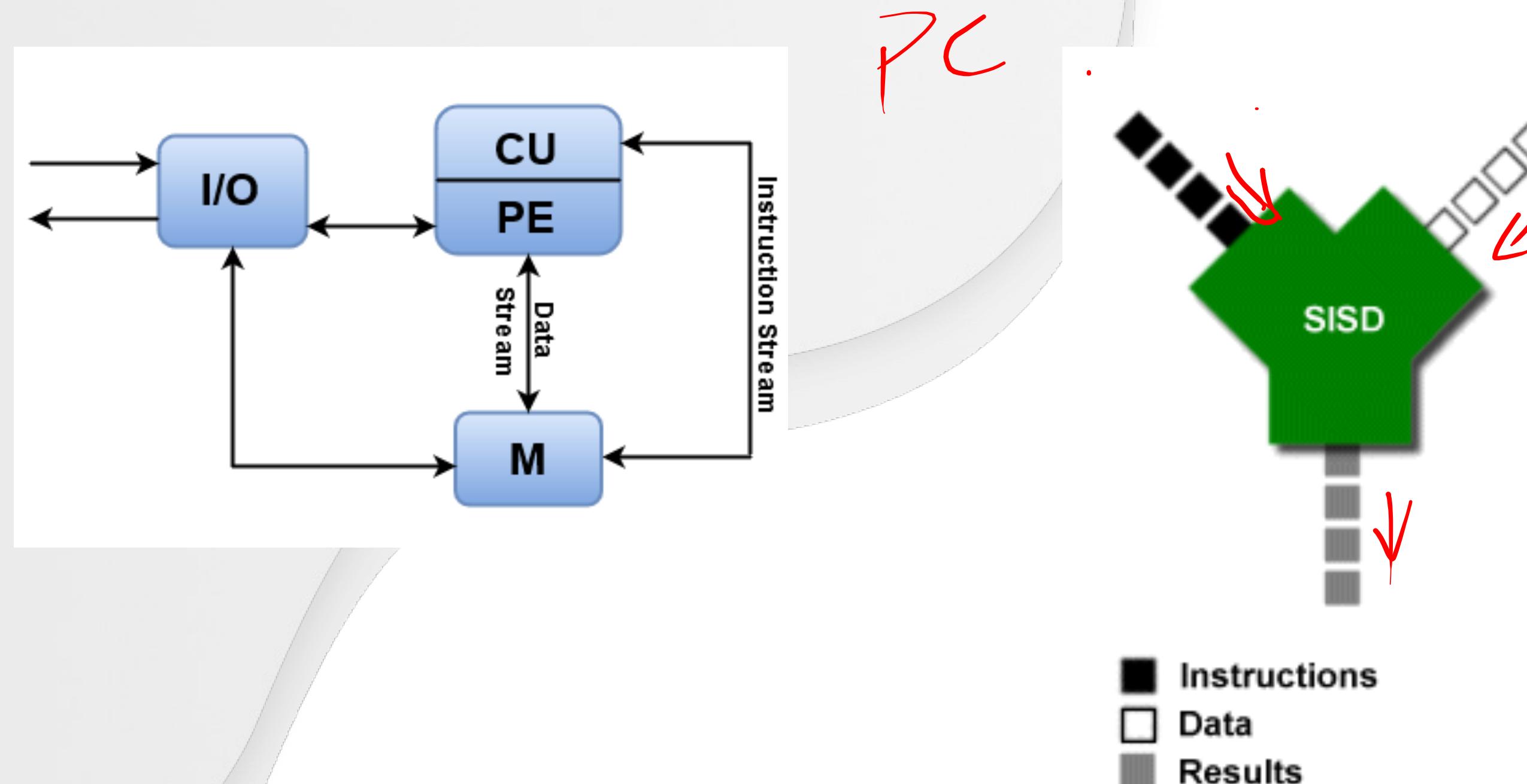
3.2.5 现在计算机中的多处理器

		指令流 (Instruction Stream)	
		单一(Single)	多个(Multi)
数据流(Data Stream)	单一 (Single)	SISD	<u>MISD</u>
	多个 (Multi)	<u>SIMD</u>	MIMD

3. 指令系统设计与CPU运行控制

3.2.5 现在计算机中的多处理器

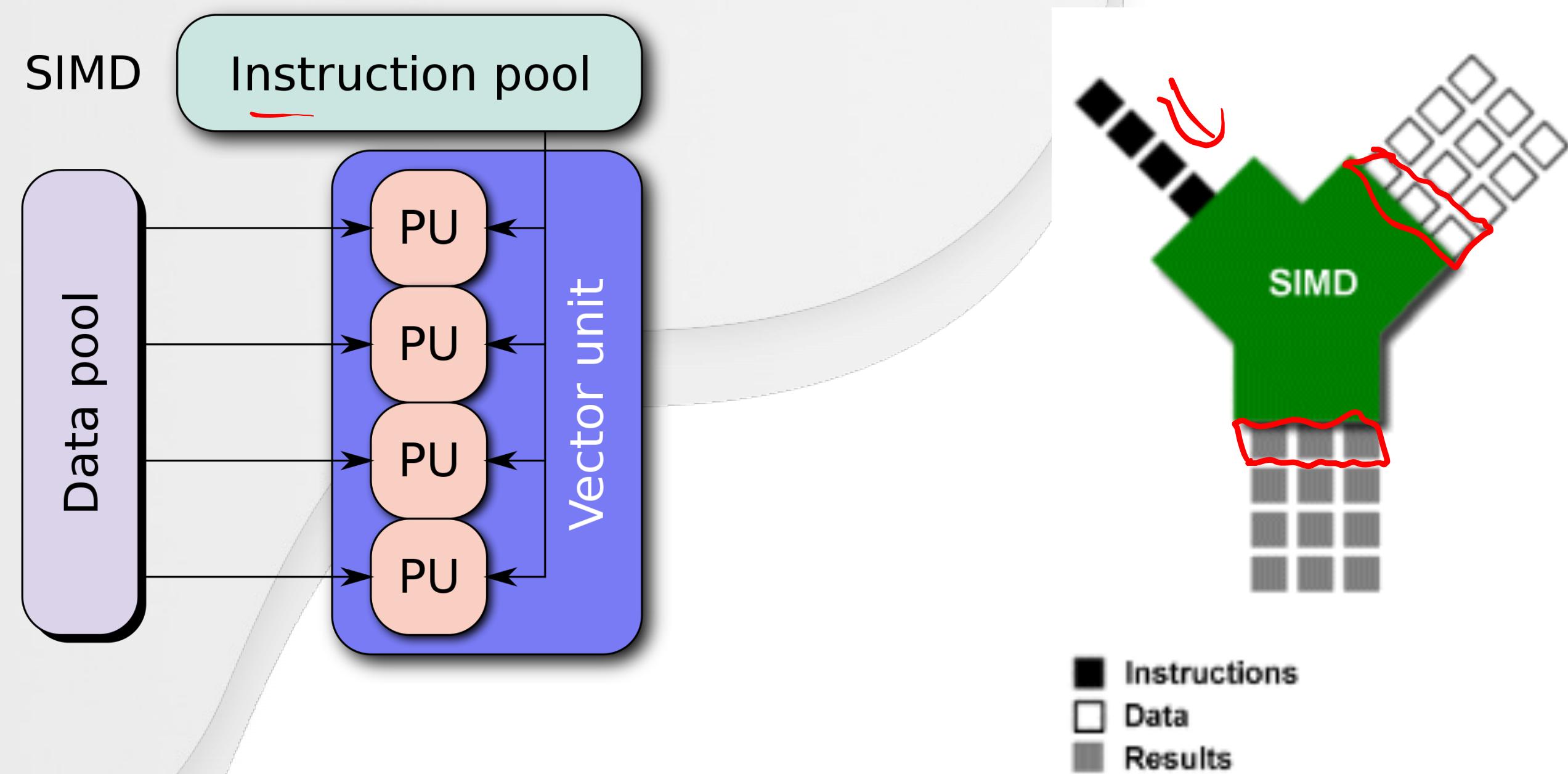
1、SISD(Single Instruction Single Data): 一指令一个数据 (Intel Pentium4)。



3. 指令系统设计与CPU运行控制

3.2.5 现在计算机中的多处理器

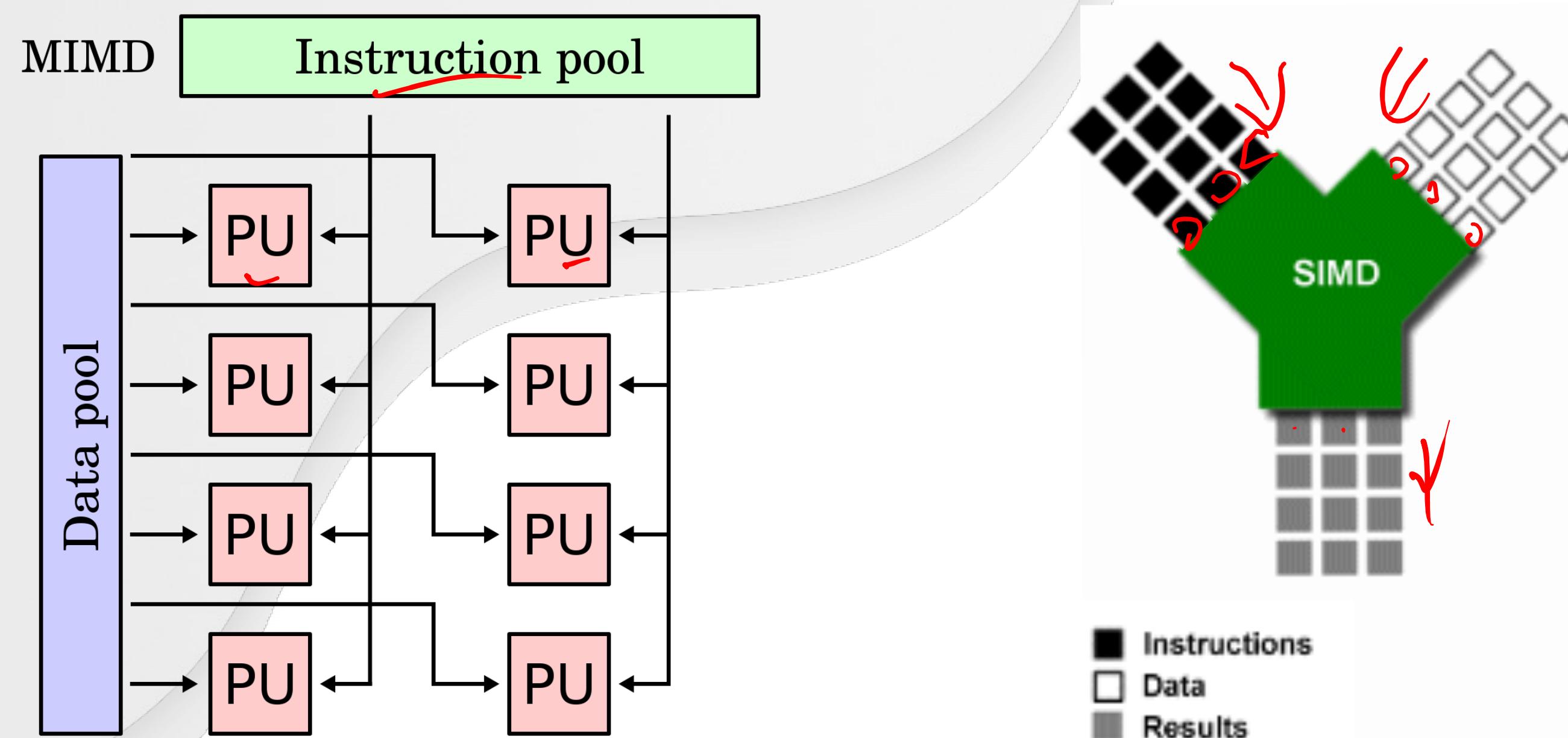
2、SIMD (Single Instruction Multiple Data): 一指令多数据 (X86的SSE指令，向量计算机)。



3. 指令系统设计与CPU运行控制

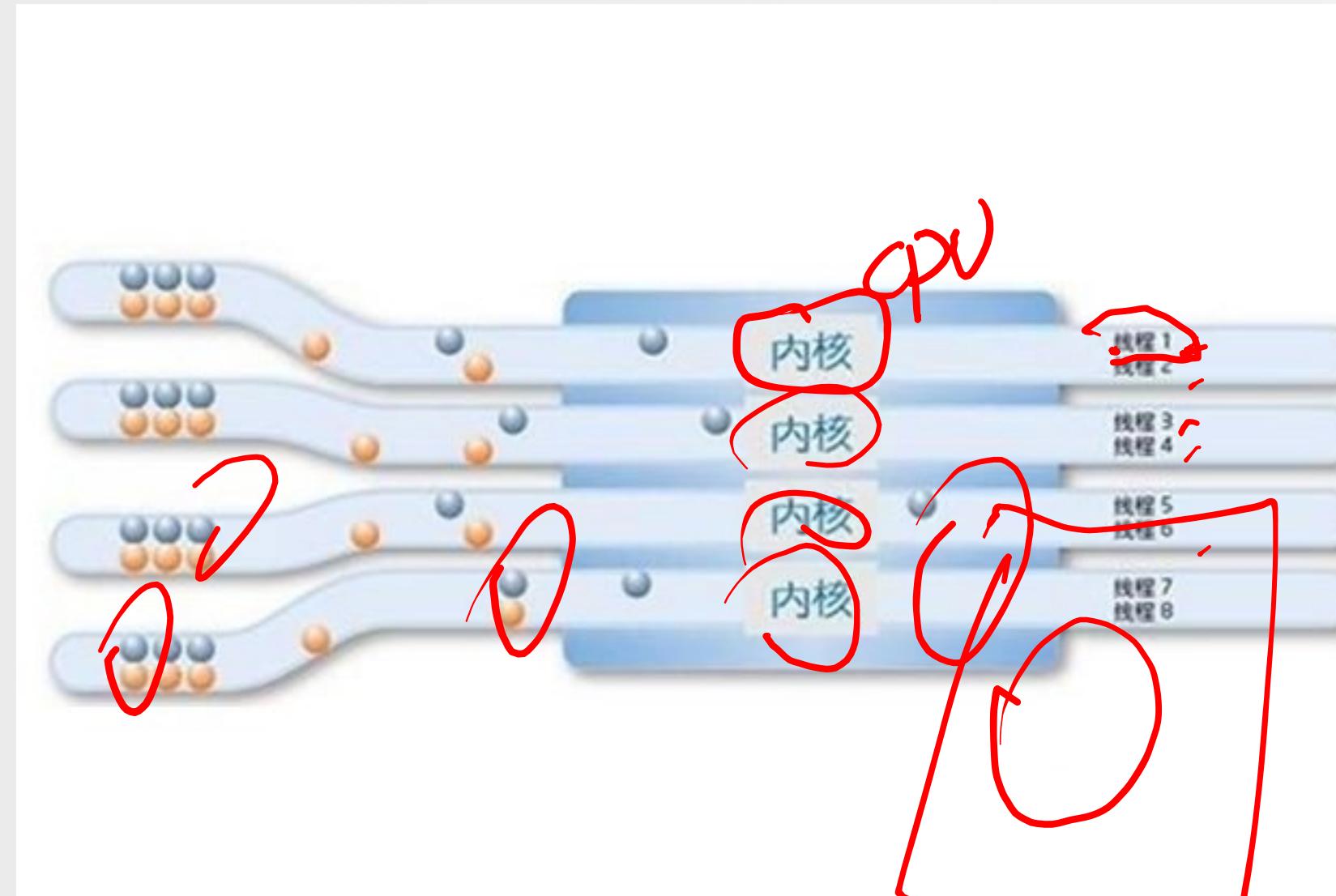
3.2.5 现在计算机中的多处理器

3、MIMD (Multiple Instruction Multiple Data): 多指令多数据，多计算机与多处理器
应用：Intel Xeon e5345）。





3.2.5 硬件线程、软件线程、超线程

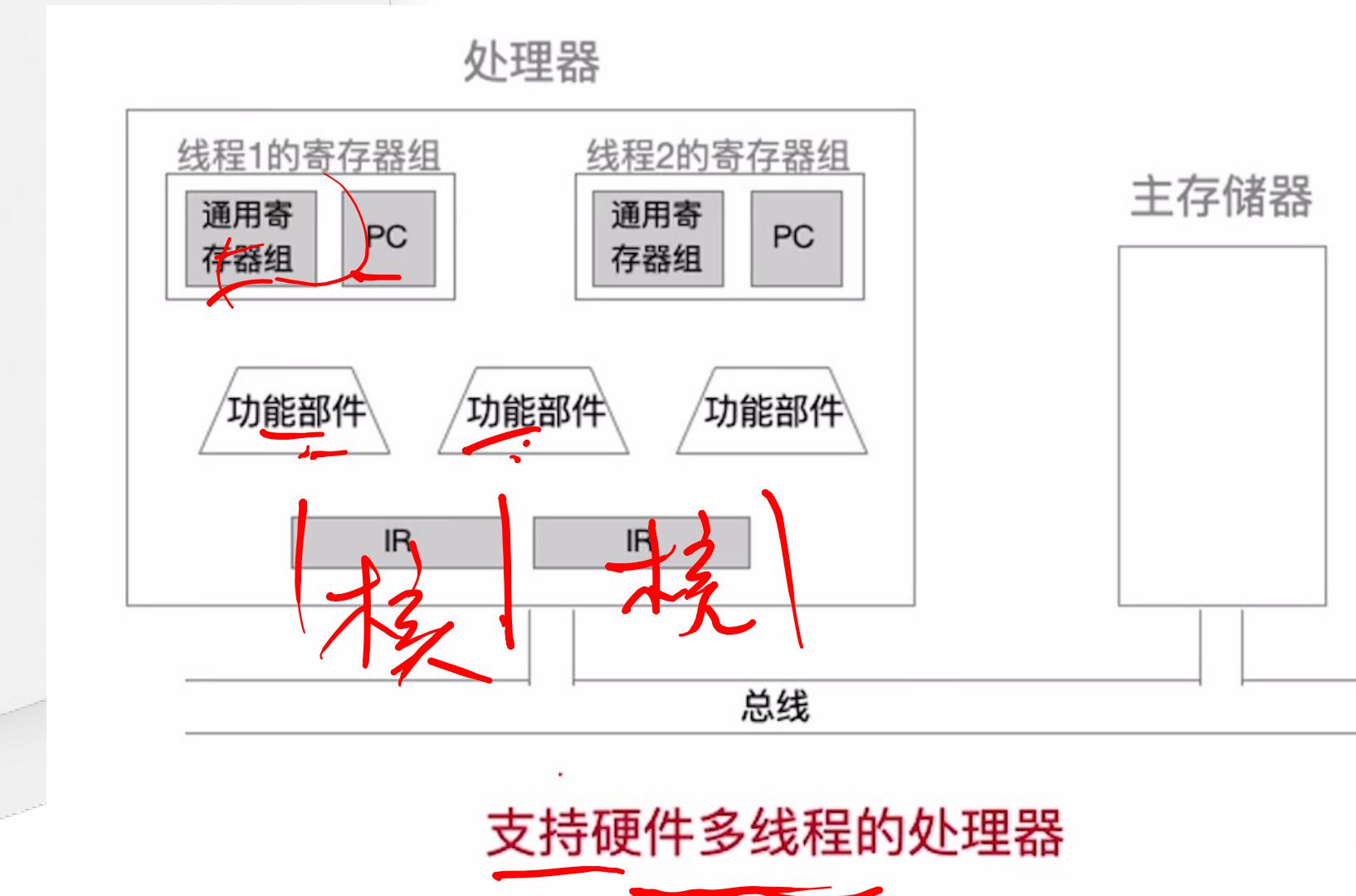
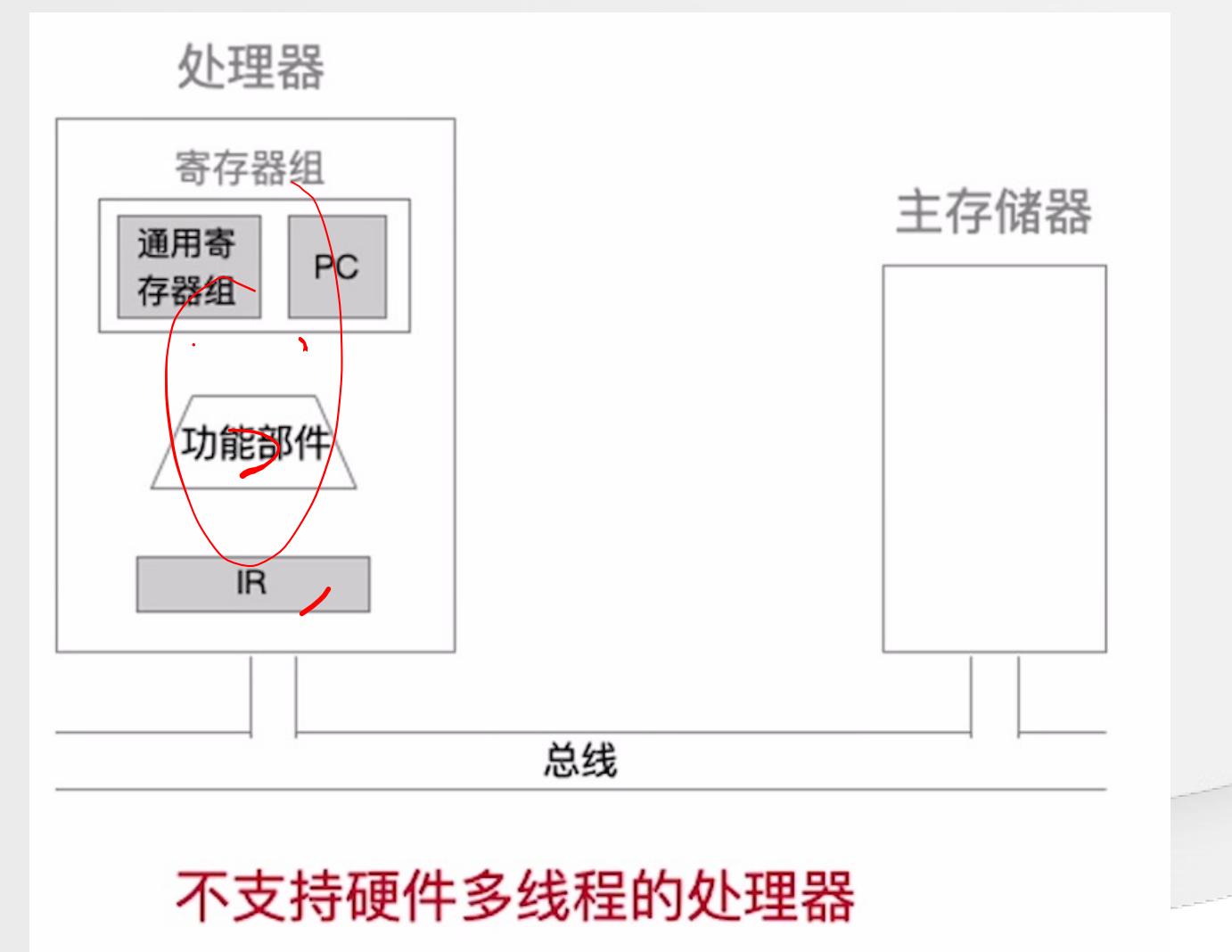


- 1、硬件线程是逻辑内核或逻辑处理器，数量是确定的
- 2、操作系统运行程序开启的是软件线程，数量可以无限增加
- 3、一个硬件线程对于操作系统的调度来说就是一个CPU，多个软件线程可以运行在同一个硬件线程中。
- 4、可以在操作系统中设置软件线程占用不同的逻辑处理器，来均衡工作负载。
- 5、超线程 (HT, Hyper-Thread) 是在一个单处理器或单个核中设置两套线程状态部件，共享cache和功能部件



3. 指令系统设计与CPU运行控制

3.2.5 硬件多线程的基本概念



三种执行方式：

- 1、细粒度多线程：多个线程间轮流交叉执行指令，各线程指令不相关
- 2、粗粒度多线程：一个线程碰到阻塞时切换线程
- 3、同时多线程 (SMT)：两种都使用



3. 指令系统设计与CPU运行控制

3.2.5 硬件多线程的执行比较

细粒度	
时钟	CPU
i	发射线程A的指令: j、j+1
i+1	发射线程B的指令: k、k+1
i+2	发射线程A的指令: j+2、j+3
i+3	发射线程B的指令: k+2、k+3

粗粒度	
时钟	CPU
i	发射线程A的指令: j、j+1
i+1	发射线程A的指令: j+2、j+3、发现Cache缺失
i+2	发射线程B的指令: k、k+1
i+3	发射线程B的指令: k+2、k+3

同时	
时钟	CPU
i	发射线程A的指令: j、j+1, 发射线程B的指令: k、k+1
i+1	发射线程A的指令: j+2, 发射线程B的指令: k+2、发射线程C的指令: m
i+2	发射线程A的指令: j+3, 发射线程C的指令: m+1、m+2

比较

特性	细粒度	粗粒度	同时
指令发射	轮流发射指令 (每个时钟周期发一个线程)	连续几个时钟周期发射同一线程指令序列, 流水线阻塞时, 切换另一线程	一个时钟周期内, 同时发射多个线程指令
线程切换频率	每个时钟周期切换一次	只有流水线阻塞时切换	NULL
线程切换代价	低	高, 要重载流水线	NULL
并行性	指令级并行, 线程间不并行	指令级并行, 线程间不并行	指令级并行, 线程间并行



3. 指令系统设计与CPU运行控制

3.2.5 性能核心与效率核心

The screenshot shows a woman sitting at a desk with a computer monitor displaying a game. The page title is 'How Intel® Core™ Processors Work'. A red circle highlights the text '3.5GHz'. Below the title, it says 'Next gen support. Enhanced overclocking. Revolutionary hybrid design. Learn what makes Intel® Core™ desktop processors tick.' A red box highlights a section about Intel® Core™ 14th Gen Processors having up to 24 cores and 32 threads. Handwritten red annotations include 'x + y = 24' and '2x + y = 32'.

Intel® Core™ 14th Gen Processors have arrived, and they're even more capable than before. An Intel® Core™ i9 Processor 14900K, for instance, has up to 24 cores, 32 threads, and a P-core Max Turbo Frequency of up to 5.6 GHz — ideal for high-FPS gaming and running multiple resource-consuming applications.

Along with higher clock speeds and more cores, Intel® Core™ desktop processors contain technologies that further heighten performance. Chief among them is Intel's latest in hybrid architecture design. Introduced in 12th Gen, this breakthrough technology increases core efficiency and delivers intelligent workload.

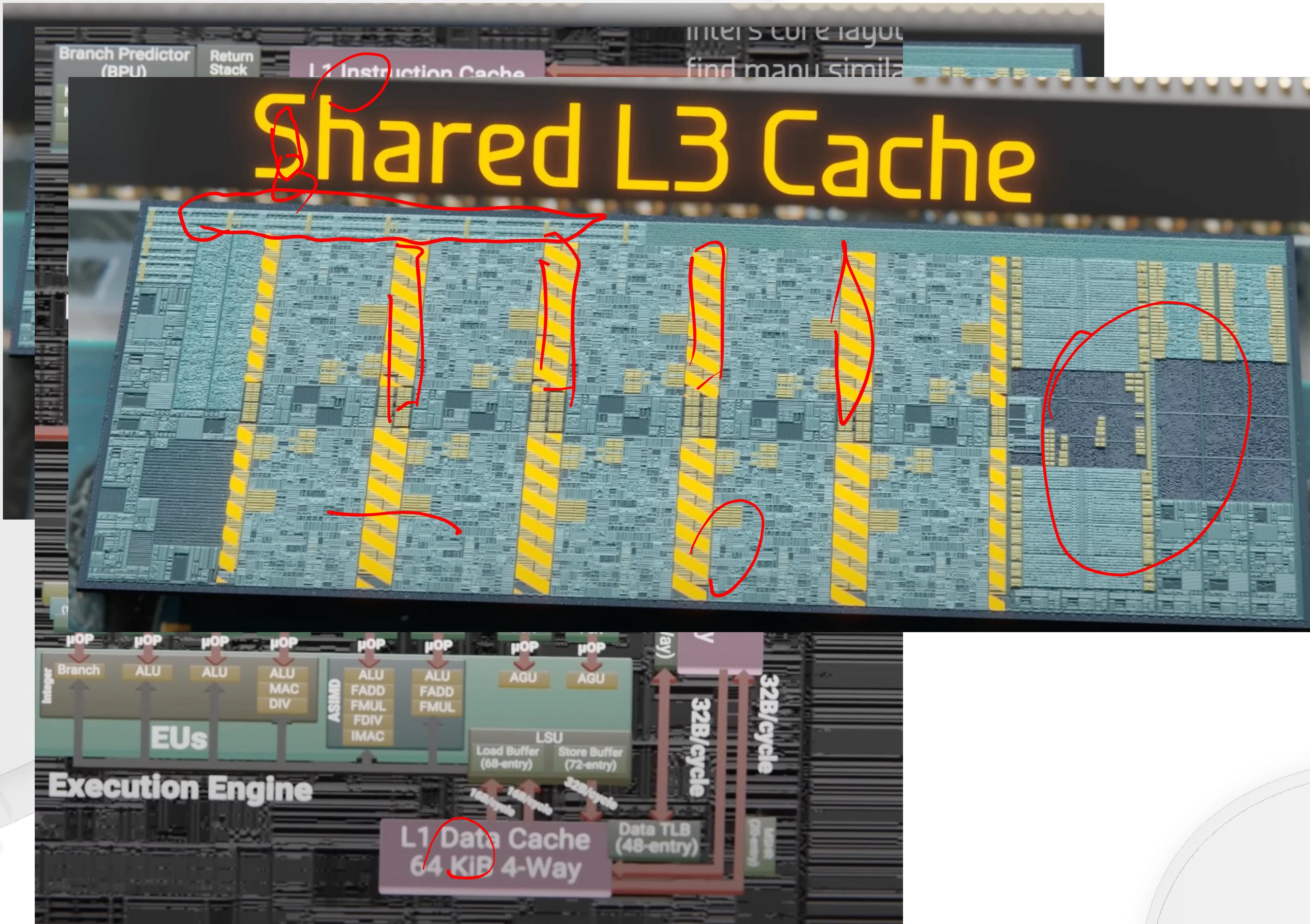
了性能核心与效率核心的混合架构。这种设计允许处理器根据需要动态调配资源，以便在执行重负载任务时提供最大的性能，同时在处理轻负载任务时优化能源使用效率。通过这种方式，处理器能够在保持高性能的同时，也提高了整体的能源效率，有助于提高便携设备如笔记本电脑的电池续航能力。

$$\begin{aligned}x + y &= 24 \\2x + y &= 32 \\x &= 8 \\y &= 16\end{aligned}$$



3. 指令系统设计与CPU运行控制

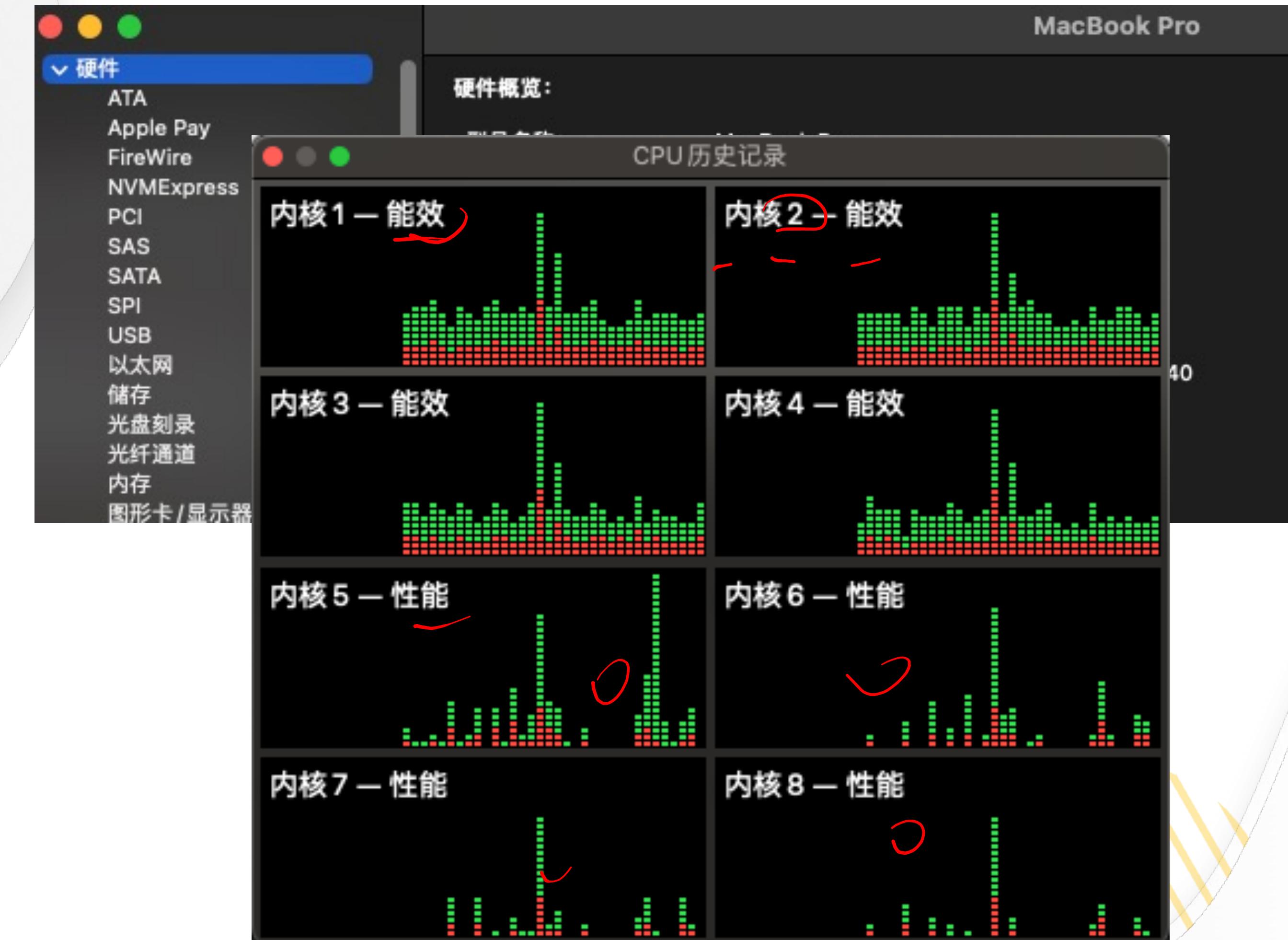
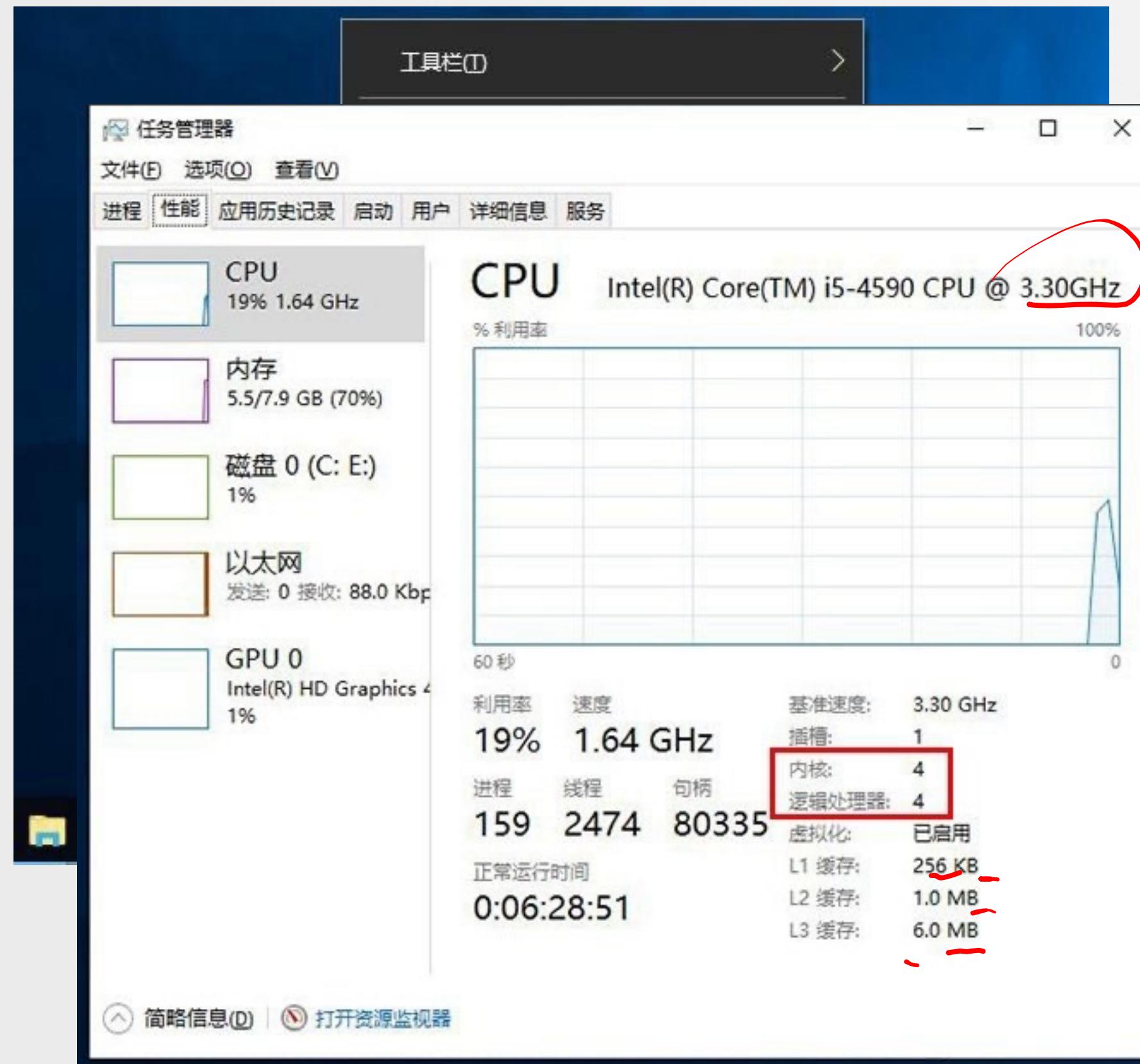
3.2.5 硬件线程与软件线程





3. 指令系统设计与CPU运行控制

3.2.5 硬件线程与软件线程

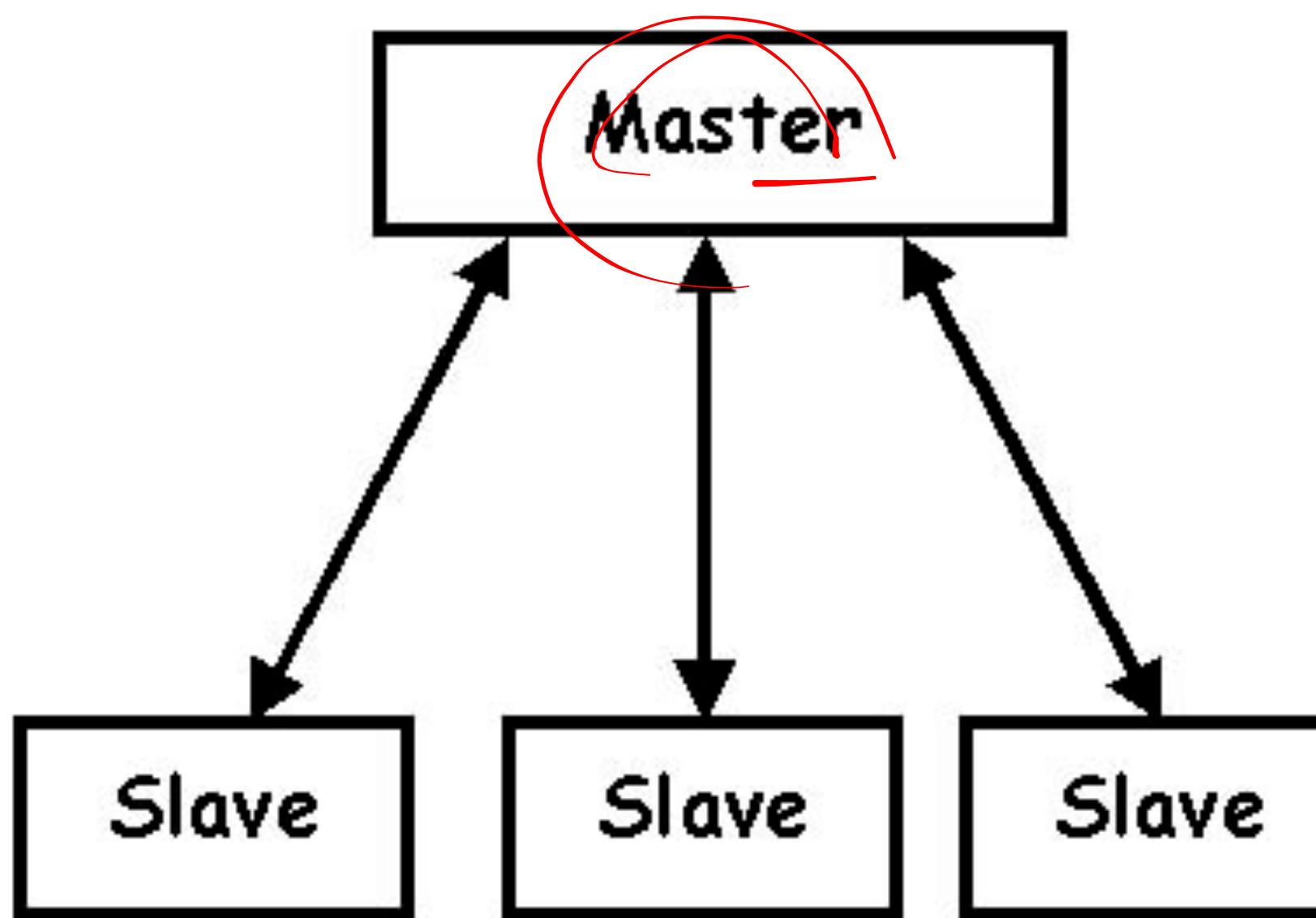


3. 指令系统设计与CPU运行控制

3.2.5 多核处理器

MIMD

1、主/从结构：主处理器负责操作系统的管理，进程或者线程需要使用系统的服务(如一次I/O调用)，则它必须给主处理器发送请求，并等待服务的处理



特点：

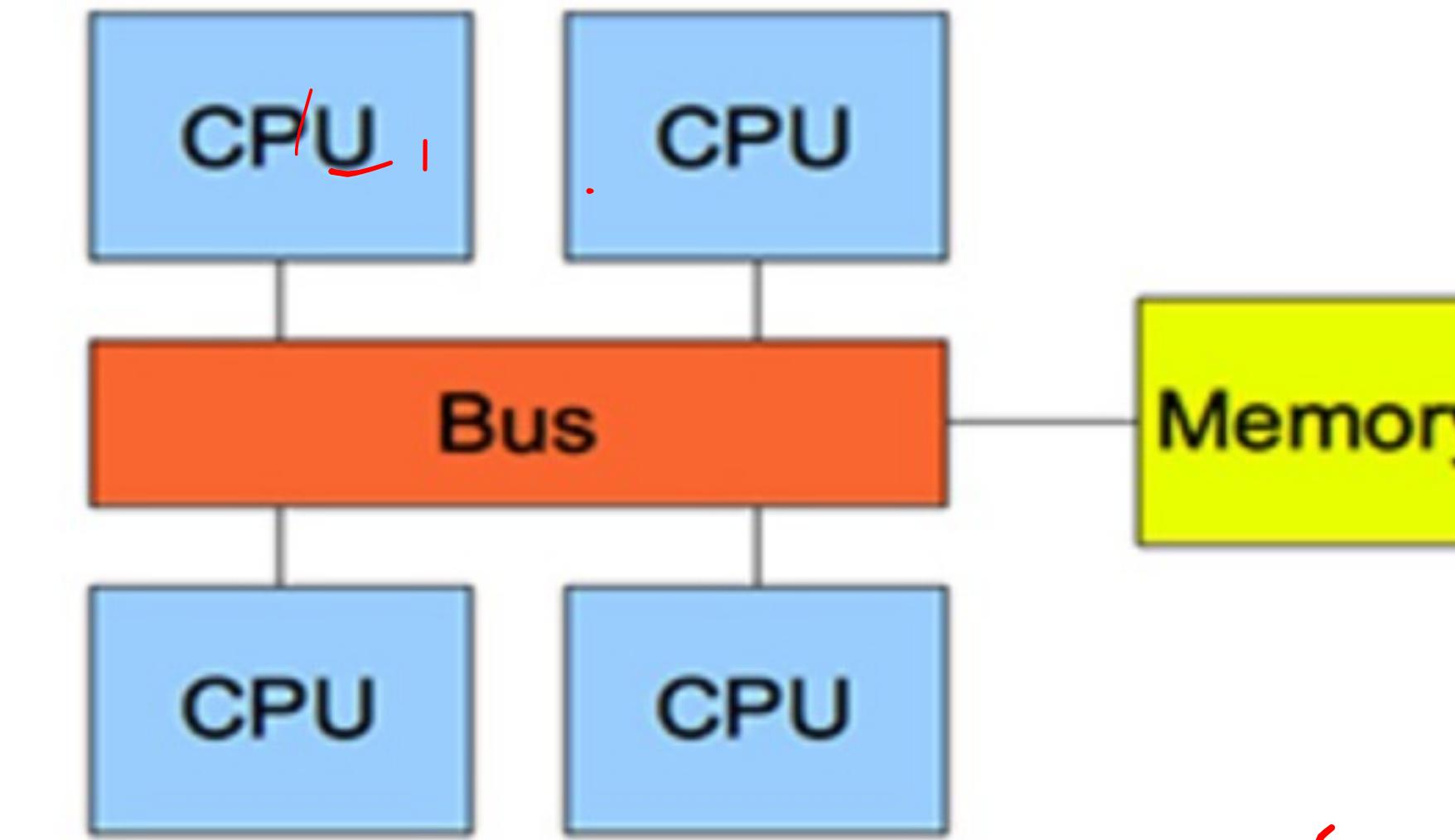
- 1)一个处理器控制了所有存储器和I/O资源，解决冲突很简单
- 2)主处理器的失败将导致整个系统失败
- 3)由于主处理器必须负责所有进程的调度和管理，因此可能称为性能瓶颈



3. 指令系统设计与CPU运行控制

3.2.5 共享存储器多处理器系统

2、对称结构(SMP): 内核可以在任何处理器上执行，所有的CPU共享全部资源，如总线，内存和I/O系统。



4~8个CPU

特点：

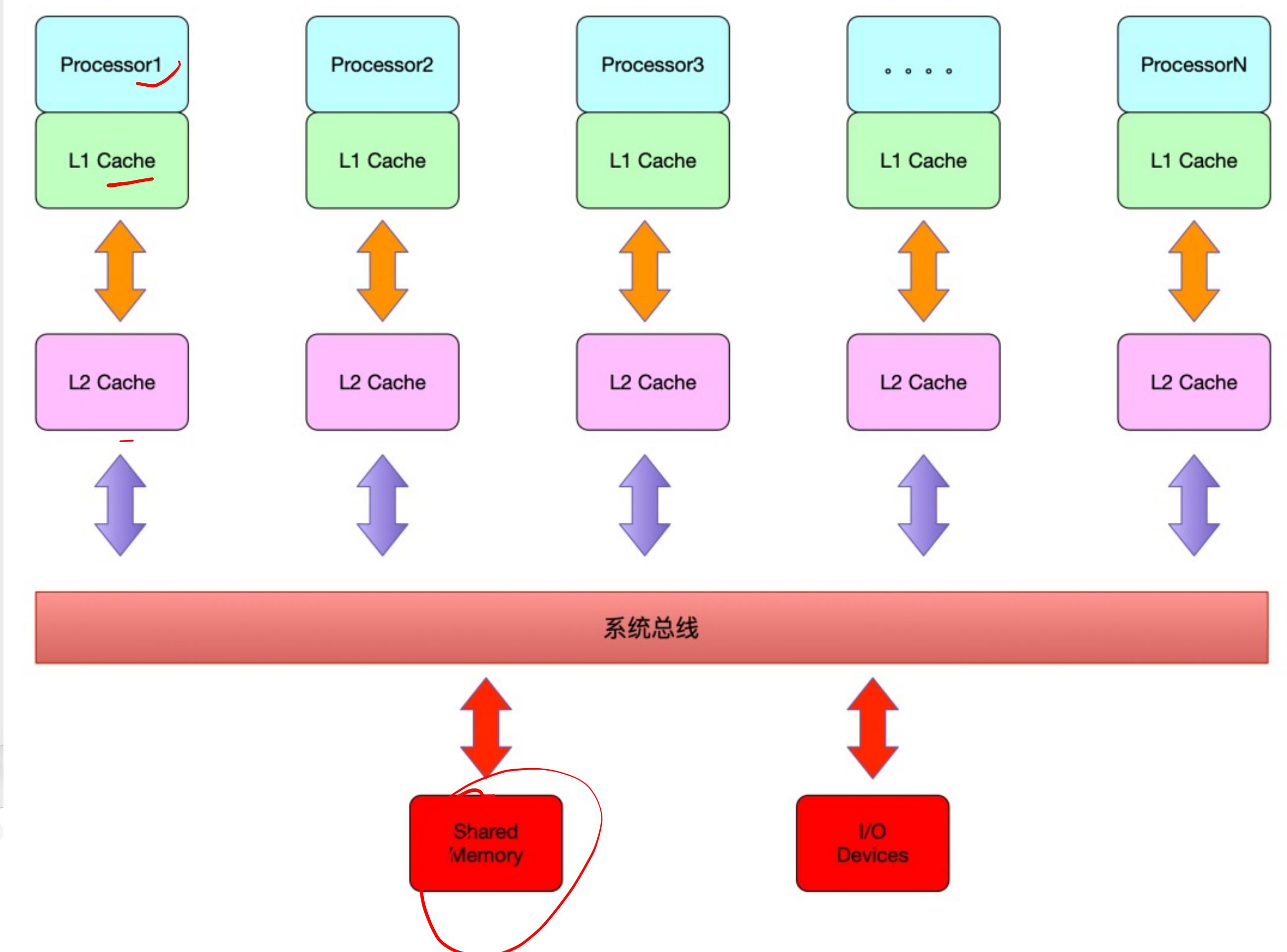
- 1、增加了操作系统的复杂度
- 2、必须确保二个处理器不会同时使用同一个资源
- 3、确保队列不会丢失，因此需要解决同步问题
- 4、扩展方式包括：增加内存、使用更快的CPU、增加CPU、扩充I/O(槽口数与总线数)以及添加更多的外部设备(通常是磁盘存储)



3. 指令系统设计与CPU运行控制

3.2.5 共享存储器多处理器系统

SMP也称为： UMA (Uniform Memory Access, 一致存储器访问结构)





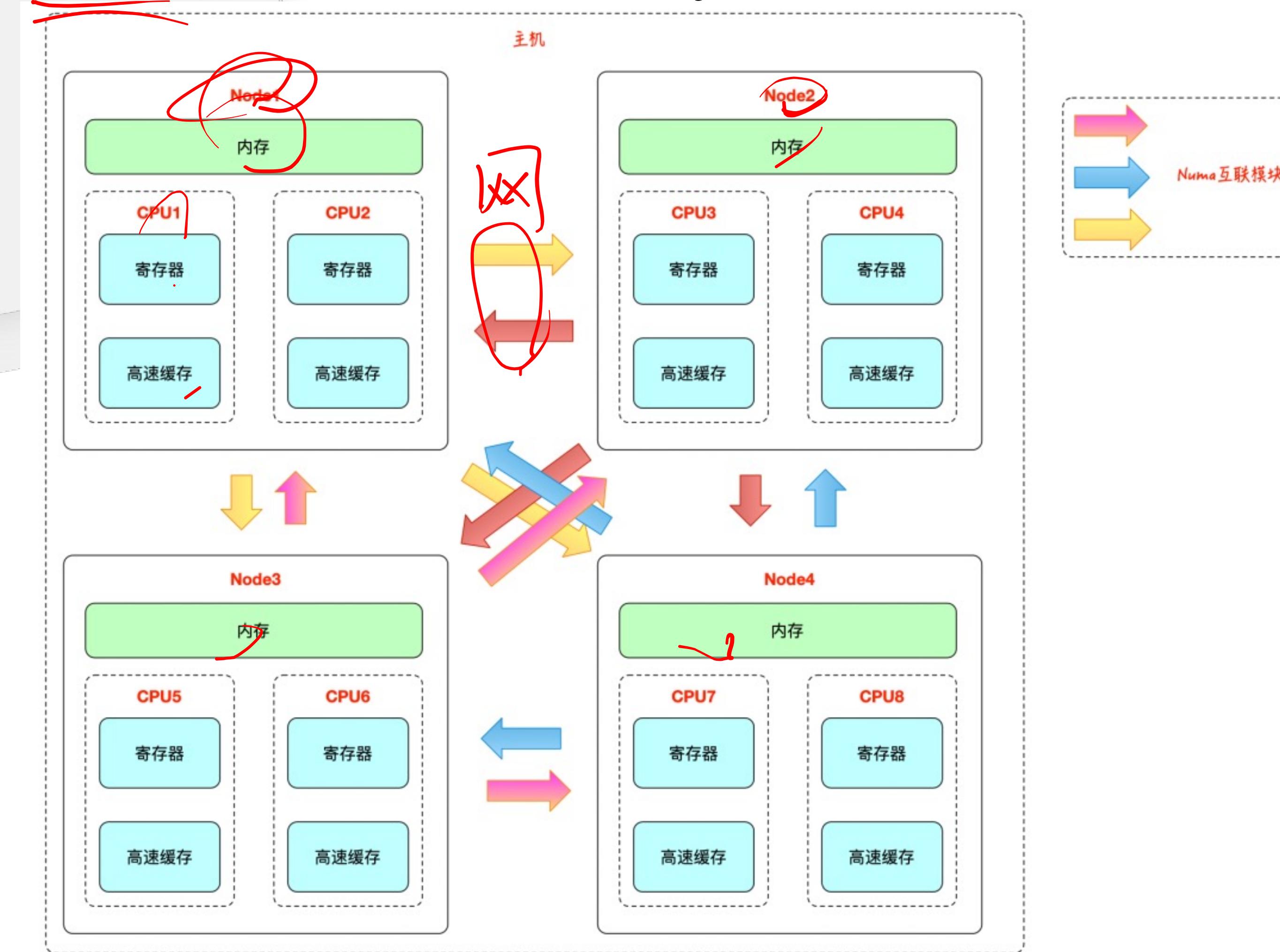
3. 指令系统设计与CPU运行控制

3.2.5 共享存储器多处理器系统

3、非一致性内存访问结构(NUMA: Non-Uniform Memory Access)

特点：

- 1、多个Node模块
- 2、每个Node有多个CPU
- 3、Node内有本地内存

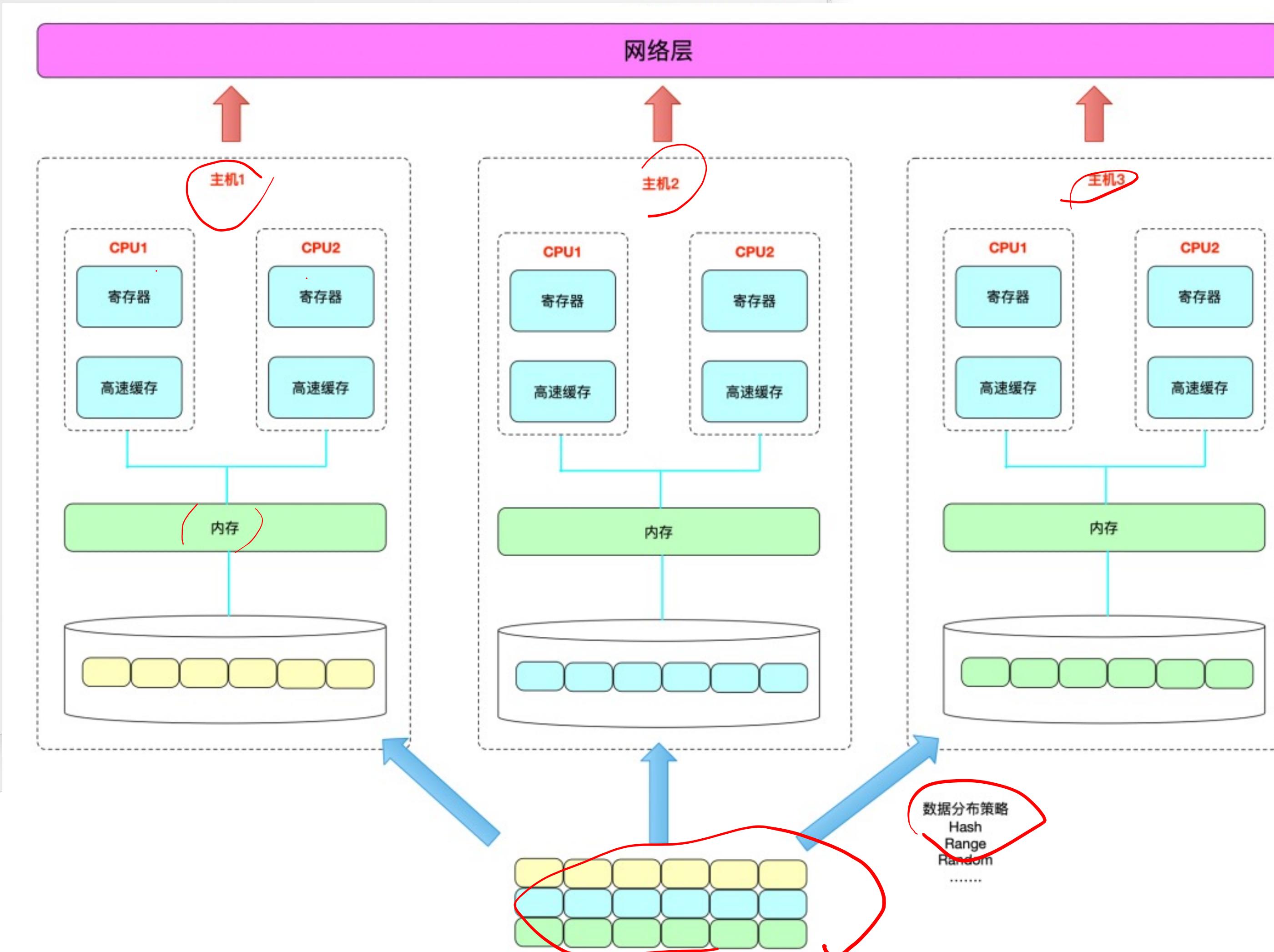




3. 指令系统设计与CPU运行控制

3.2.5 大规模并行处理模型

(MPP: Massively Parallel Processing)



特点：

- 1、多个Node模块
- 2、每个Node有多个CPU
- 3、Node内有本地内存



3. 指令系统设计与CPU运行控制

3.2 练习

例5 【2022真题】：下列关于并行处理技术中的叙述，不正确的是 ()。

- A. 多核处理器属于MIMD结构
- C. 硬件多线程技术只可用于多核处理器
- B. 向量处理器属于SIMD结构
- D. SMP中所有处理器共享单一物理地址空间



3. 指令系统设计与CPU运行控制

3.2 练习

例6：下列关于超线程（HT）技术的描述中，正确的是（C）。

- A. 超线程技术可以令四核的Intel i7处理器变成八核。
- B. 超线程是一项硬件技术，能使系统性能大幅提升，与操作系统和应用软件无关。
- C. 含有超线程技术的CPU需要芯片组的支持才能发挥技术优势。
- D. 超线程模拟出的每个CPU核都具有独立的资源，各自工作互不干扰。



支持硬件多线程的处理器



3. 指令系统设计与CPU运行控制

3.2 练习

例7：下列关于多核CPU和单核CPU的描述中，错误的是（D）。

- A. 双核的频率为2.4GHz，那么其中每个核心的频率也是2.4GHz。
- B. 采用双核CPU可以降低计算机的系统的功耗和体积。
- C. 多核CPU共用一组内存，数据共享。SMP
- D. 所有程序在多核CPU上运行速度都快。



3. 指令系统设计与CPU运行控制



3.2 本节总结

1. CPU中的CU控制器的设计有两种方式：硬布线和微程序控制
2. 为了进一步加快计算机的处理速度，对指令的执行可以采用流水线的方式执行，但是要注意流水线执行过程中的流水线中断引起的额外开销
3. 程序执行过程中如果有突发情况，CPU可以在执行完一条指令时去查看是否需要处理，如果需要，会先通过隐指令在内部先完成PC地址的保存、关中断、获取中断处理程序的入口地址。后面就把所有事务交给工程师处理。
4. 多处理器进一步提升计算性能，针对不同应用会使用不同的多处理器或是多计算机系统。

集脉 clast.

欢迎参与学习

WELCOME FOR YOUR JOINING

船说：计算机基础