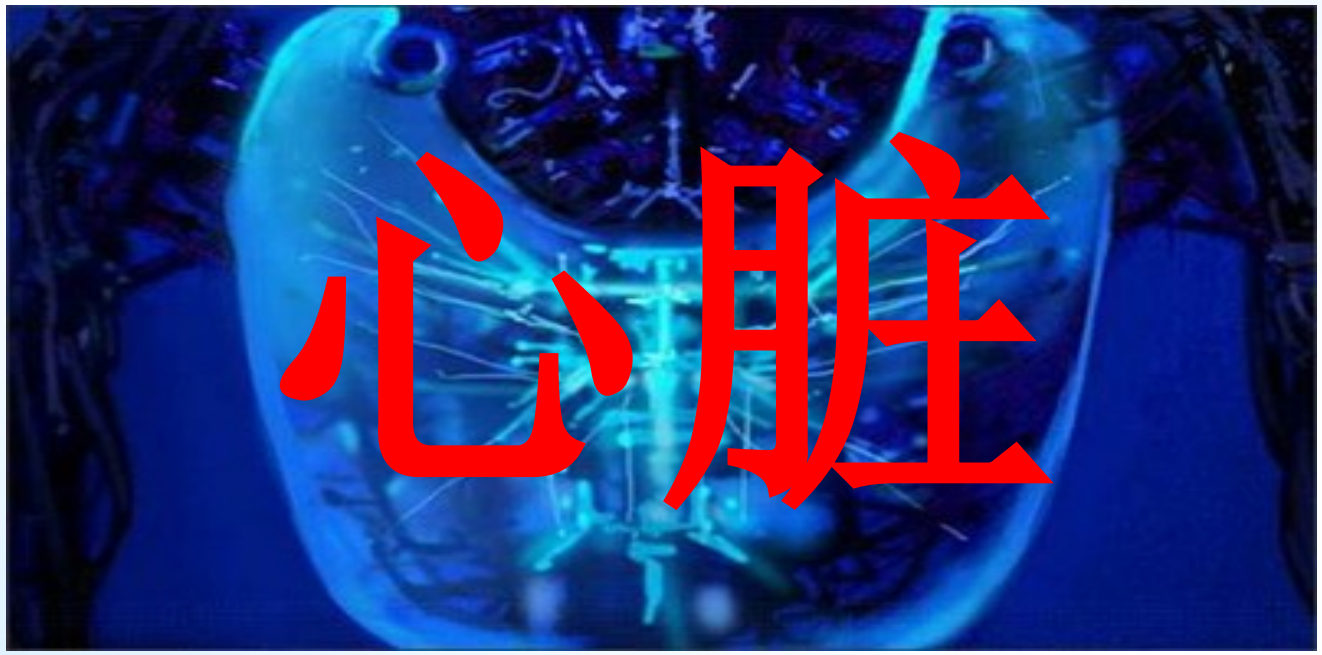
第5章
中央处理器

引入

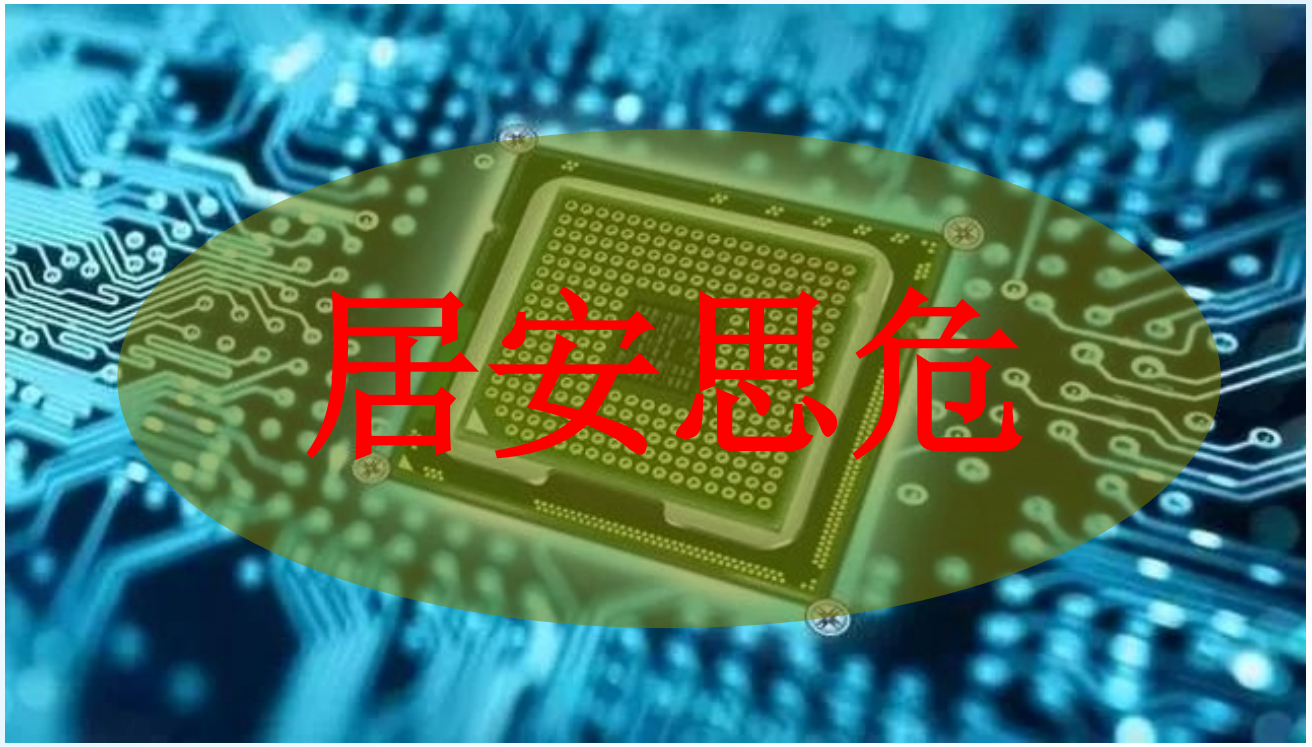


作用： CPU作为一台电脑中的核心，它的作用是无法替代的。

实质： CPU本身只是在硅晶上所集成的超大规模集成电路，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。

中央处理器

引入

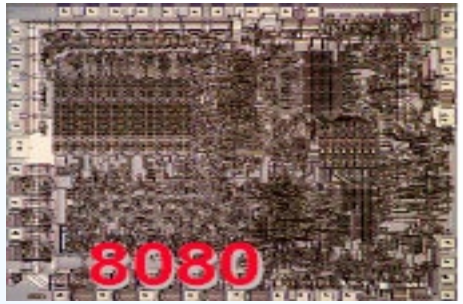


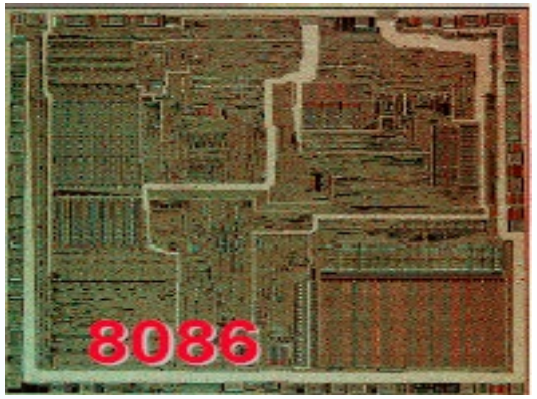
“无论是否愿意，只要民族国家还存在，科技企业和学者就必然有家国属性，不能忘记家国情怀。”

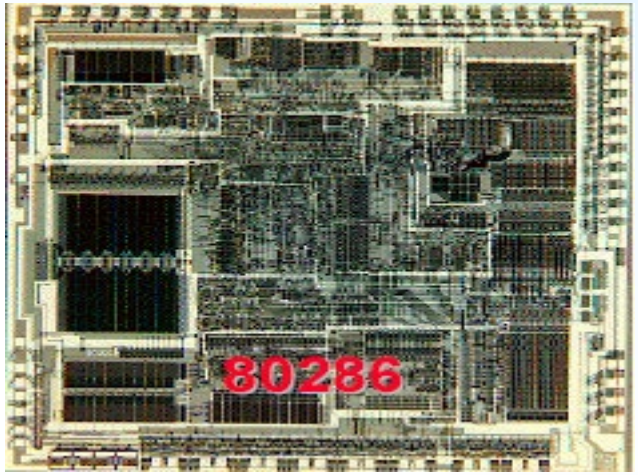
引入

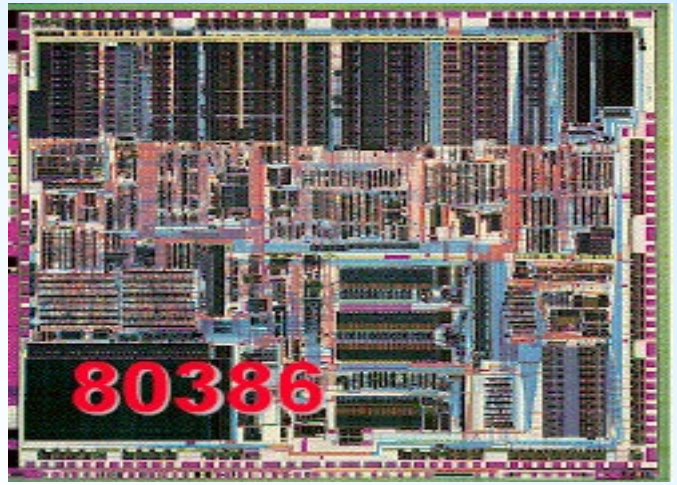
目前，最有机会的是人工智能行业，需要新的芯片和系统，打造一套全独立的生态系统。

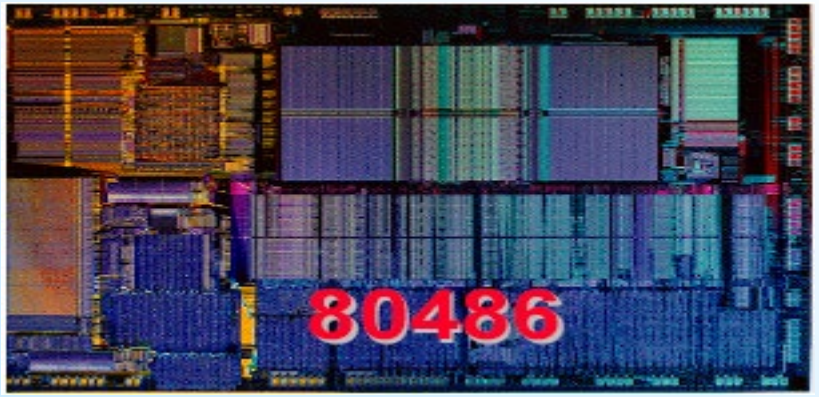
引入

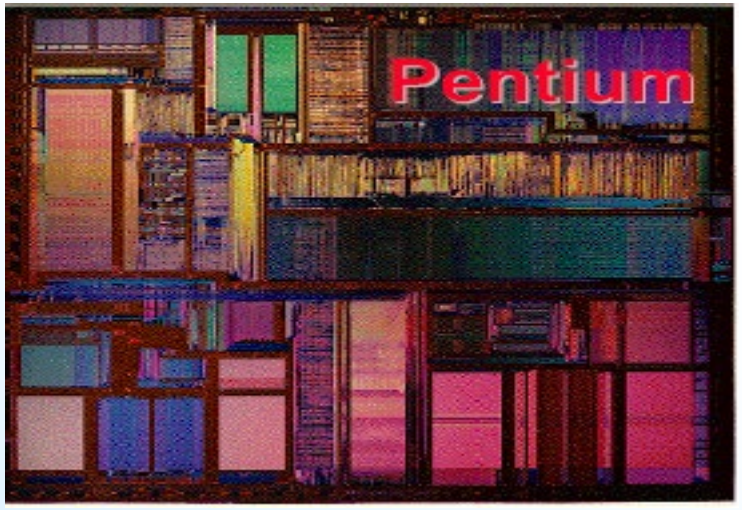


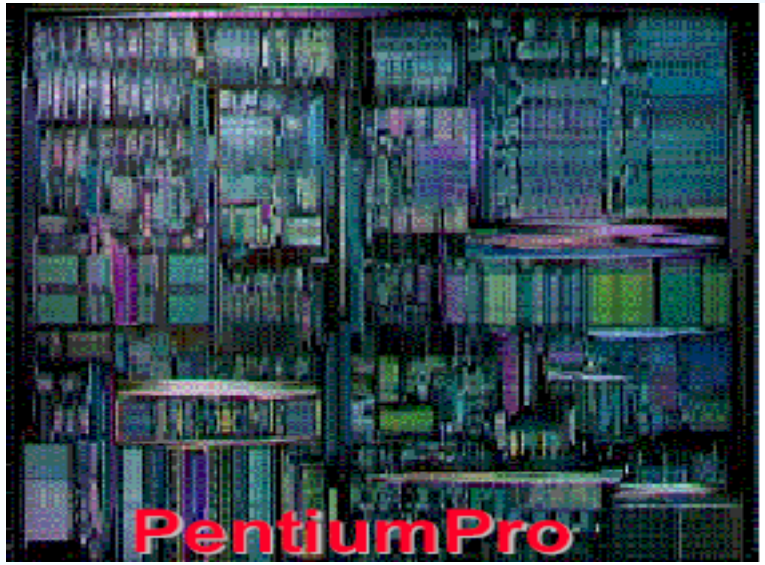










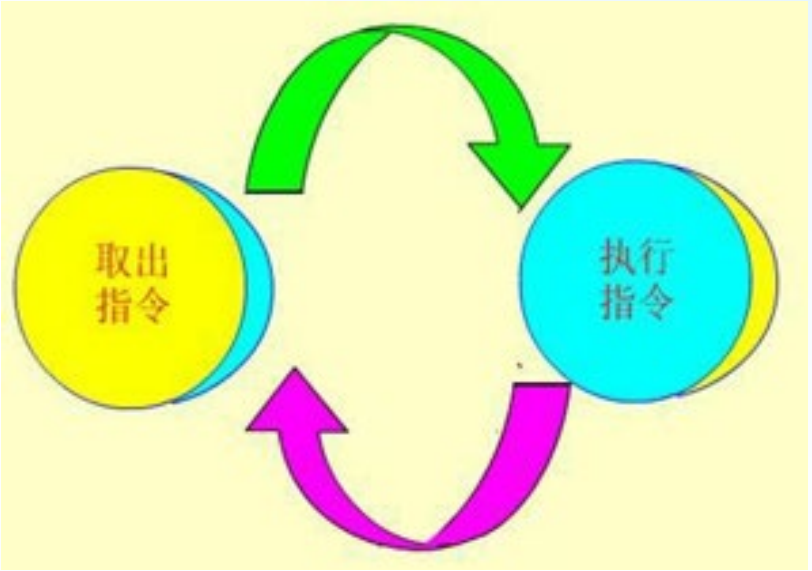


本章主要教学内容

- CPU的功能和组成（重点）
- 指令周期（重点）
- 时序产生器和控制方式
- 微程序控制器（重点难点）
- 微程序设计技术
- 硬布线控制器
- 流水线CPU

学习时，要注意。。。

- 要牢牢把握整机的概念
- 要抓住以下线索：
  - 计算机操作的逻辑依据
    - 时间条件
    - 控制逻辑



5.1 CPU的功能和组成

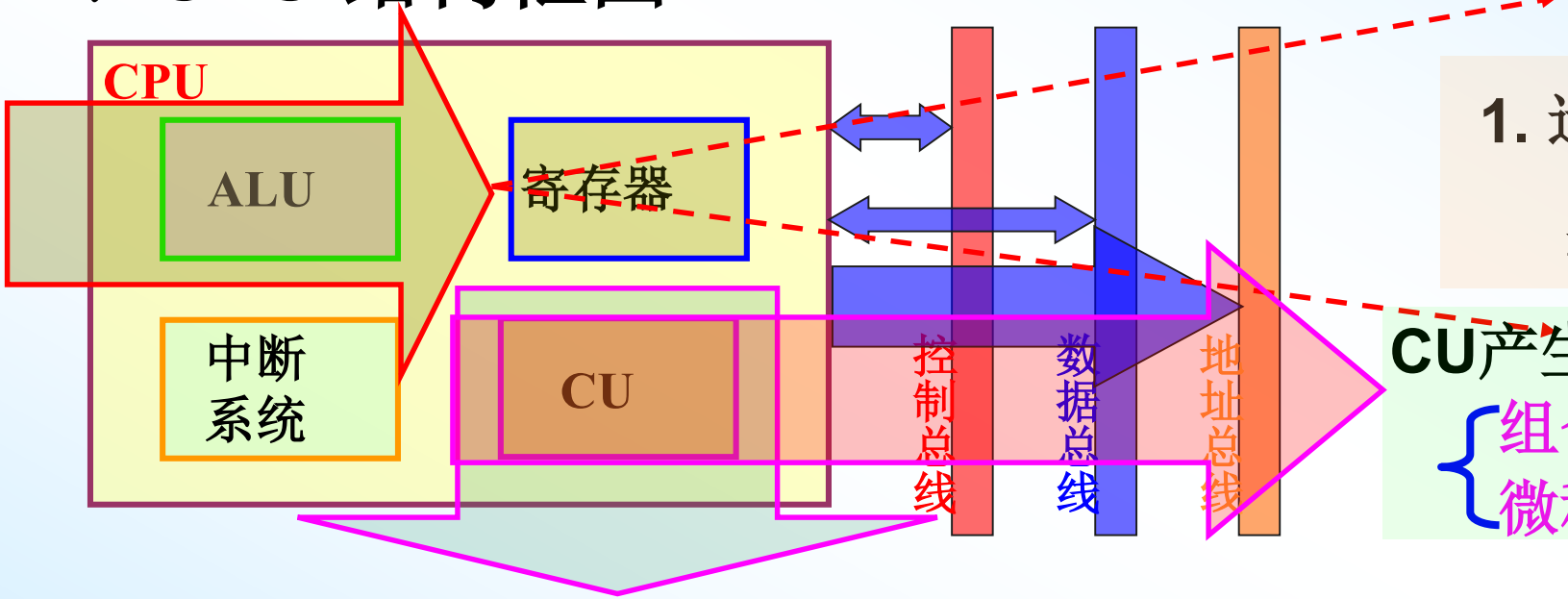
- 从冯诺依曼的程序存储思想说起...
- 一、CPU的功能

取指令 —操作控制、时间控制→ 执行指令

  - 指令控制（程序的顺序控制）
  - 操作控制（微操作控制信号处理指令）
  - 时间控制（微操作控制信号的定时）
  - 数据加工（算术运算和逻辑运算）

5.1 CPU的功能和组成

二、CPU 结构框图



1. 运算器（ALU）

实现算术运算和逻辑运算

CU产生全部指令的微操作命令序列

- 组合逻辑设计：硬连线逻辑
- 微程序设计：存储逻辑

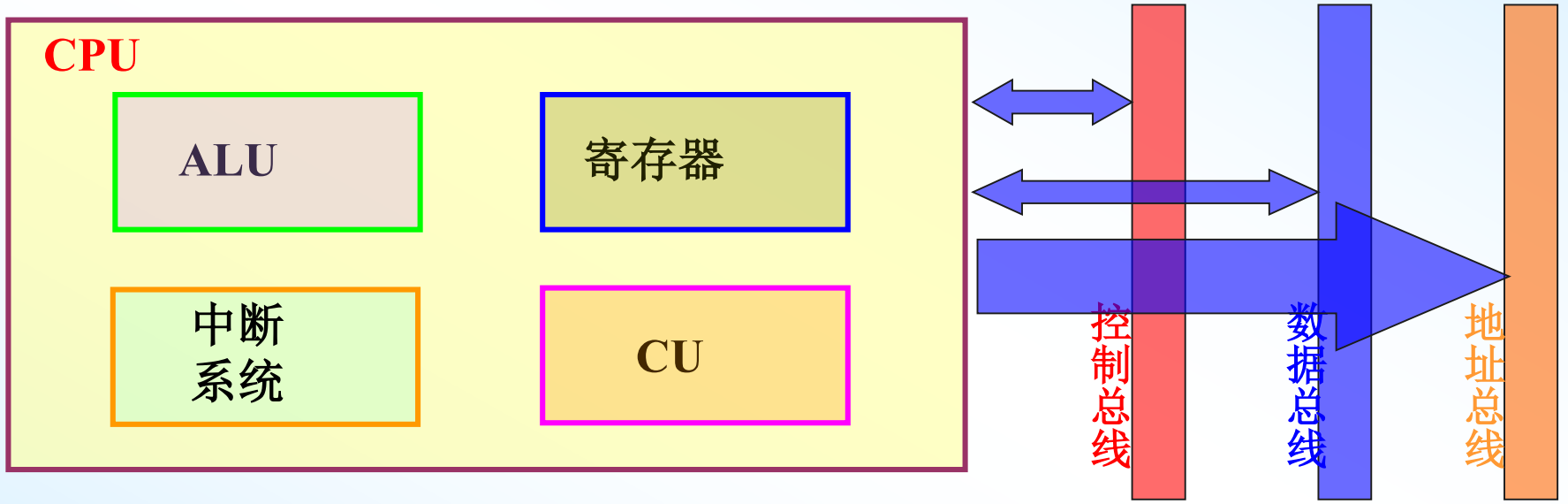
2. 控制器（CU） （控制和协调各部件工作）

取指令 → 分析指令 → 执行指令

- 取指令：从内存中提取指令
- 分析指令：分析指令的操作类型及操作数的来源
- 执行指令：发出控制信号序列

5.1 CPU的功能和组成

二、CPU 结构框图

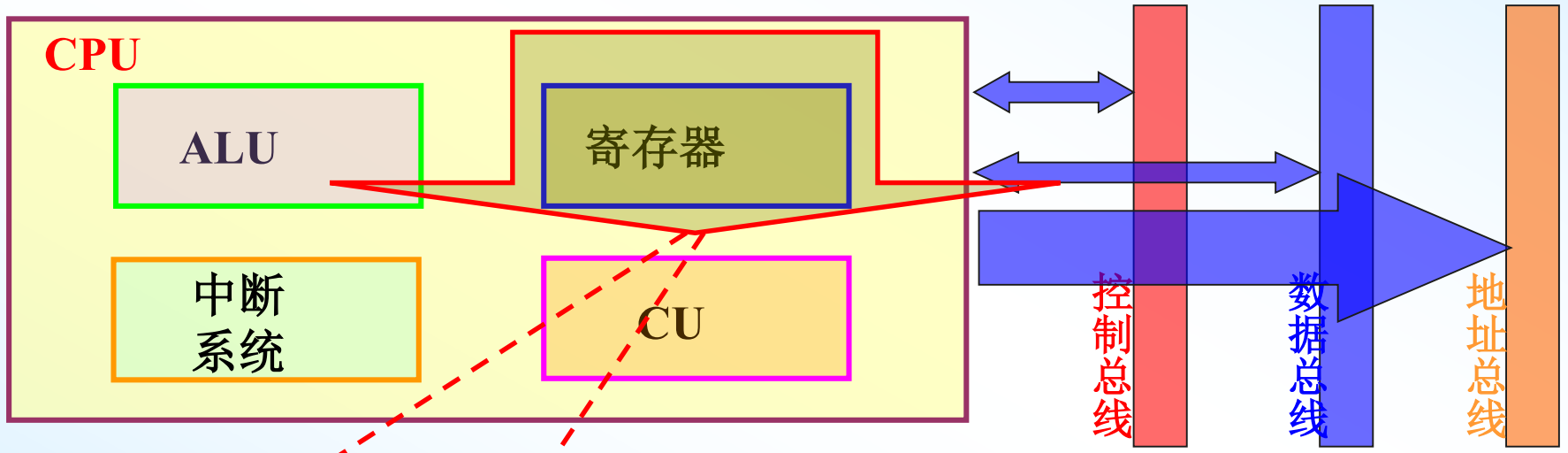


3. 中断系统（第八章介绍）

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。

5.1 CPU的功能和组成

二、CPU 结构框图



4.CPU 的寄存器

(1) 数据缓冲寄存器（DR）

(2) 指令寄存器（IR）

(3) 程序计数器（PC）

(4) 地址寄存器（AR）

(5) 通用寄存器

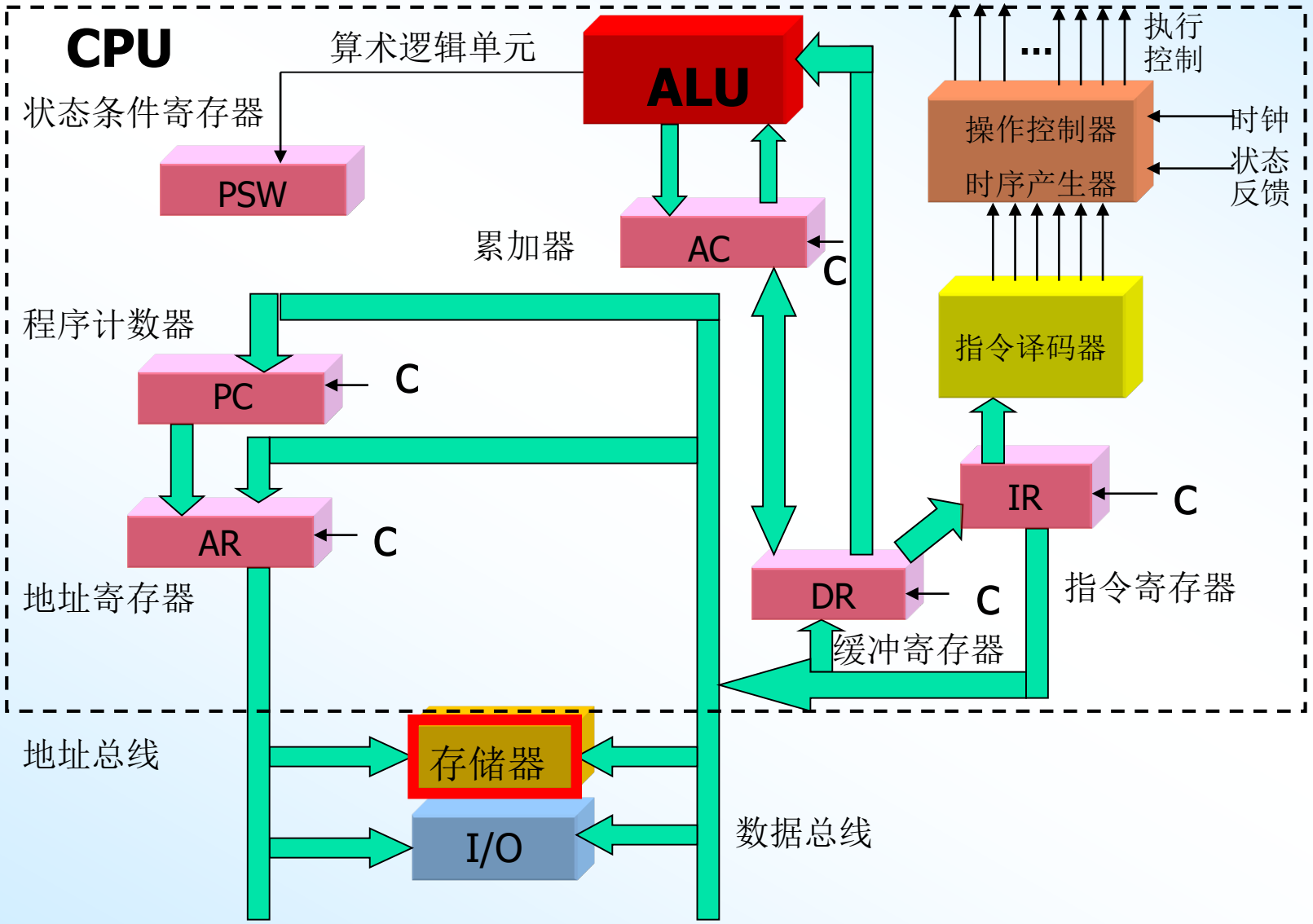
(6) 状态字寄存器（PSW）

PC　关系　DR

AR　怎样？

CPU模型

经典模型：


CPU
算术逻辑单元
ALU
状态条件寄存器
PSW
累加器
AC
C
程序计数器
PC
C
AR
C
地址寄存器
DR
C
缓冲寄存器
IR
C
指令寄存器
指令译码器
操作控制器
时序产生器
执行控制
时钟
状态反馈
地址总线
存储器
I/O
数据总线

哈佛模型：

- 指令cache
- 数据cache

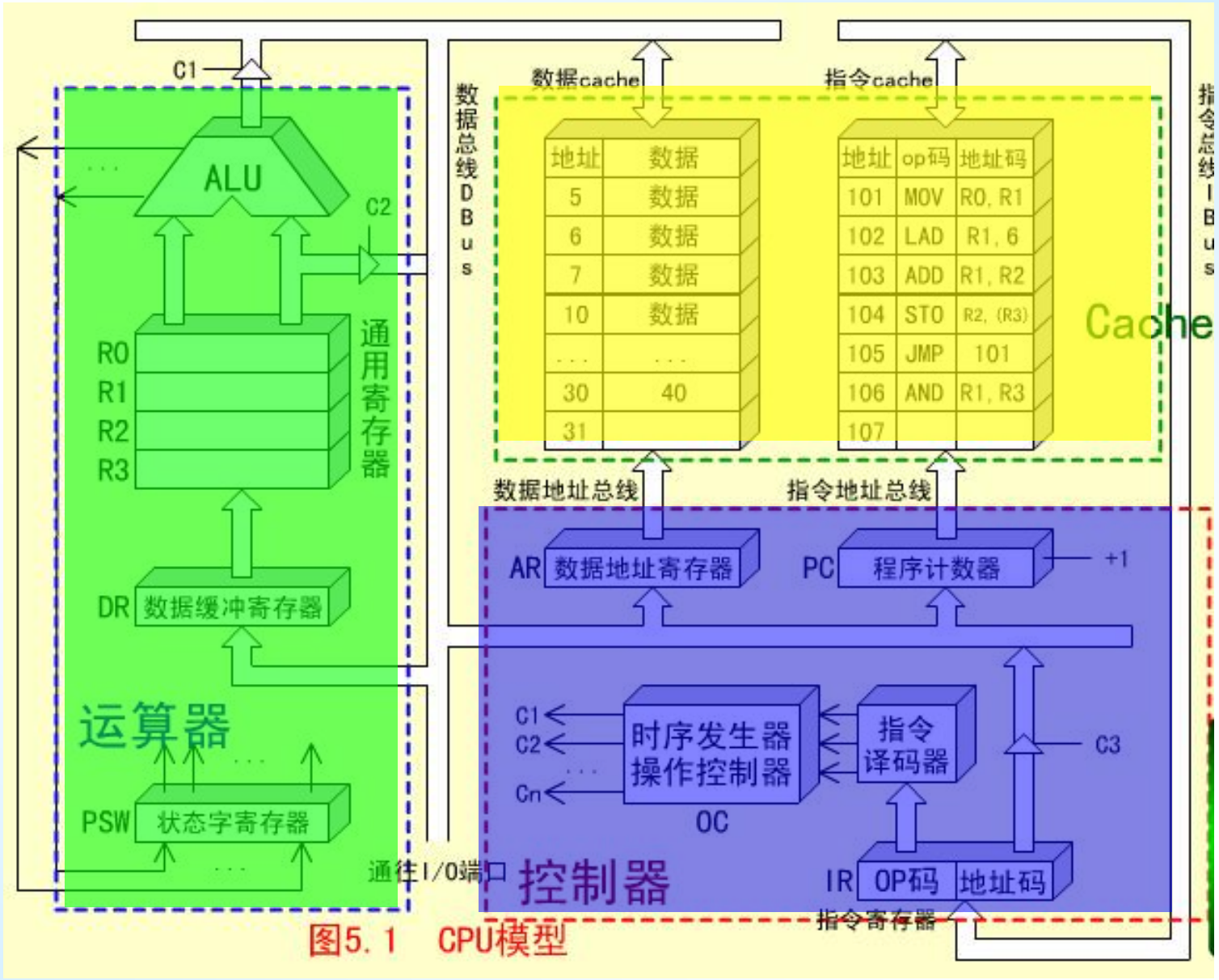


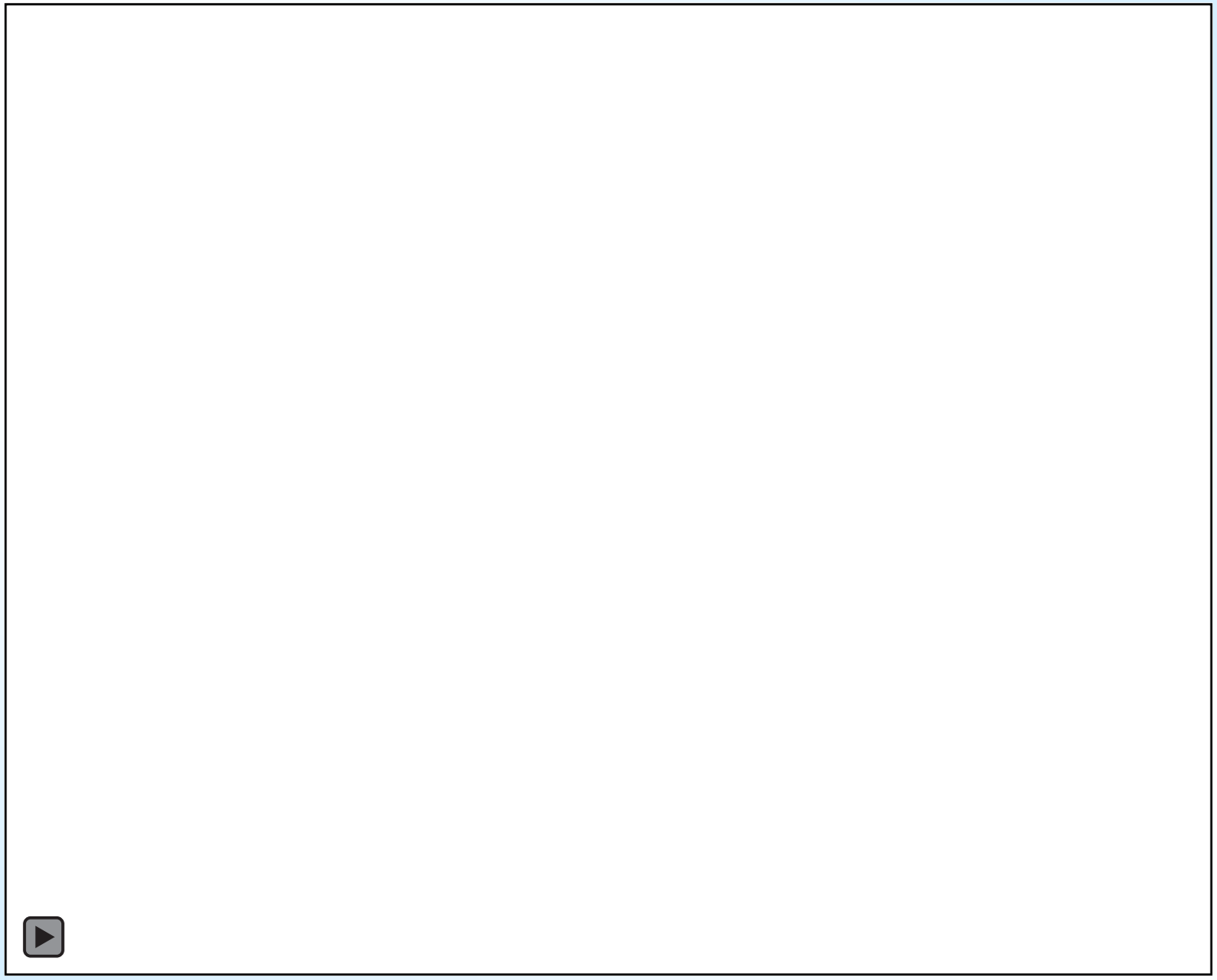
图5.1 CPU模型

PC什么时候自增？

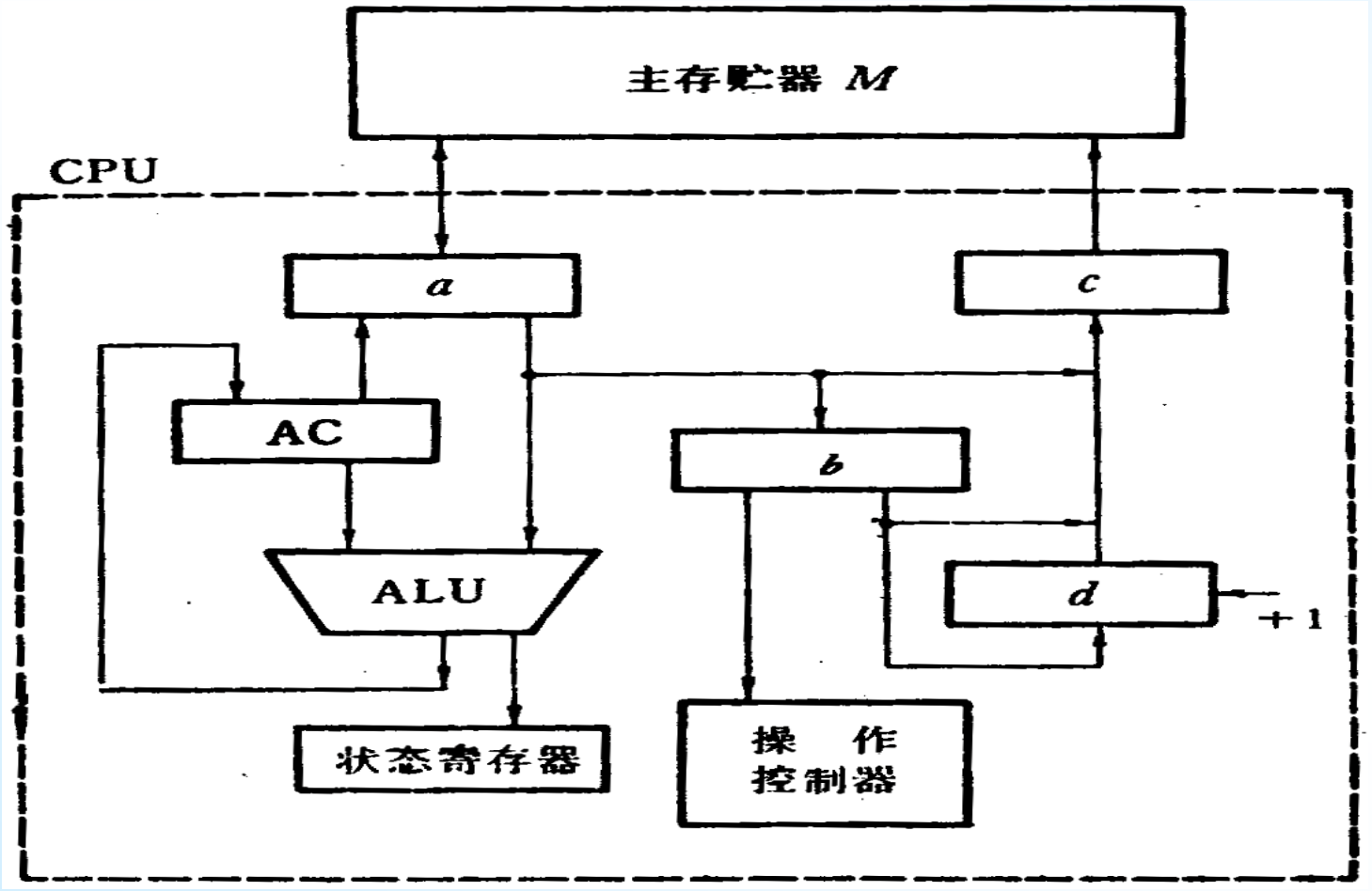
74161 型四位同步二进制计数器的功能表

清 0	预置	控制		时钟	预置数据输入				输出			
$\overline{R_D}$	$\overline{LD}$	EP	ET	CP	A_3	A_2	A_1	A_0	Q_3	Q_2	Q_1	Q_0
0	×	×	×	×	×	×	×	×	0	0	0	0
1	0	×	×	↑	d_3	d_2	d_1	d_0	d_3	d_2	d_1	d_0
1	1	0	×	×	×	×	×	×	保持			
1	1	×	0	×	×	×	×	×	保持			
1	1	1	1	↑	×	×	×	×	计数			

中央处理器

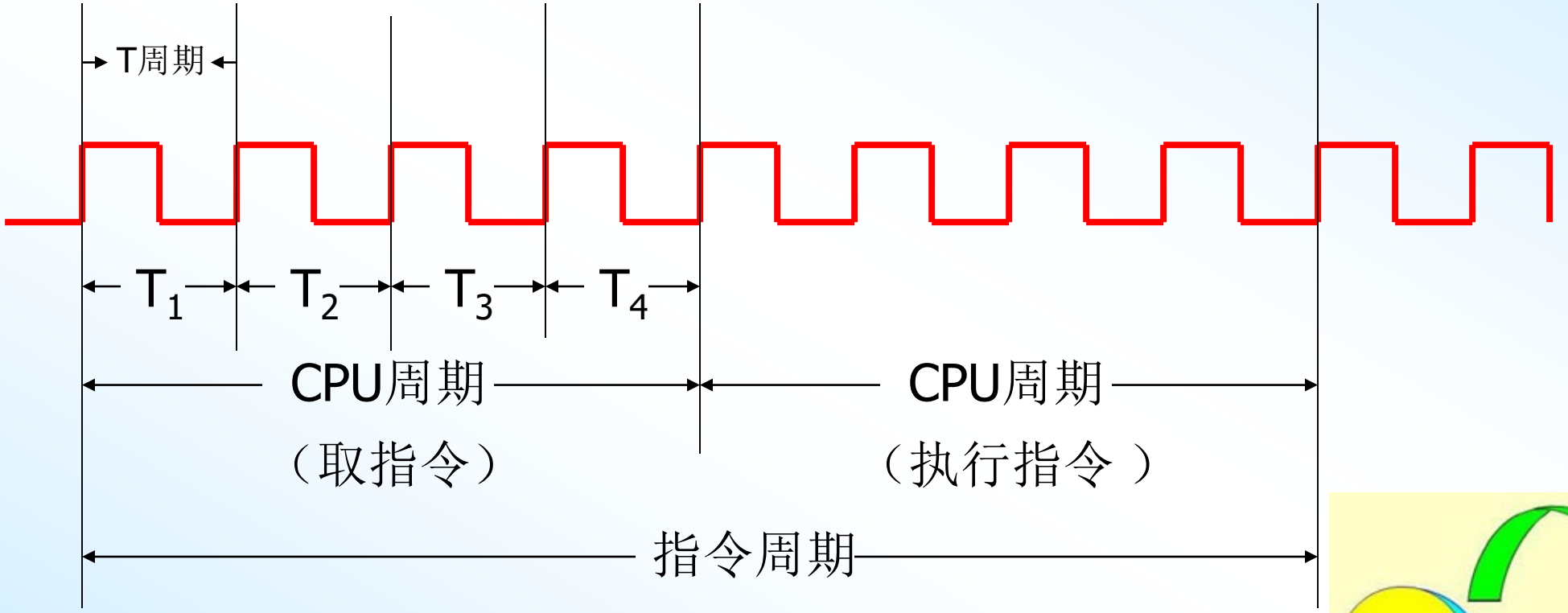


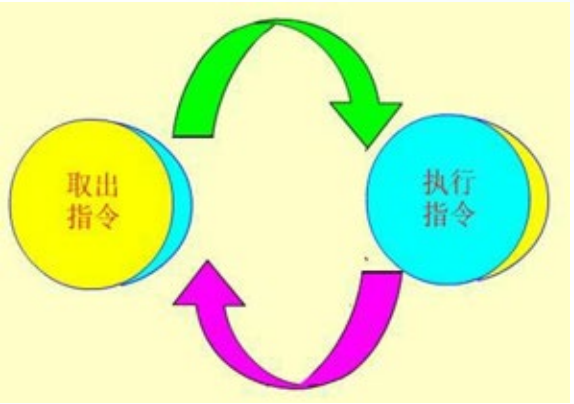
【测试题】CPU结构如下图所示，其中有一个累加寄存器AC，一个状态字寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向。要求：标明图中四个寄存器的名称。



a:DR
b:IR
c:AR
d:PC

5.2 指令周期





- 指令cache

序号	地址	指令助记符	所完成的操作
1	101	MOV R0，R1	（R1）→ R0
2	102	LAD R1，6	（6）→ R1
3	103	ADD R1，R2	（R1）+（R2）→ R2
4	104	STO R2，(R3)	（R2）→ （R3）为地址
5	105	JMP 101	101 → PC
6	106	AND R1，R3	（R1）·（R3）→ R3

- 数据cache

八进制地址	八进制数据
5	70
6	100
7	66
10	77
30	40

注：程序执行前（R0）= 00,（R1）= 10,（R2）= 20,（R3）= 30

中央处理器

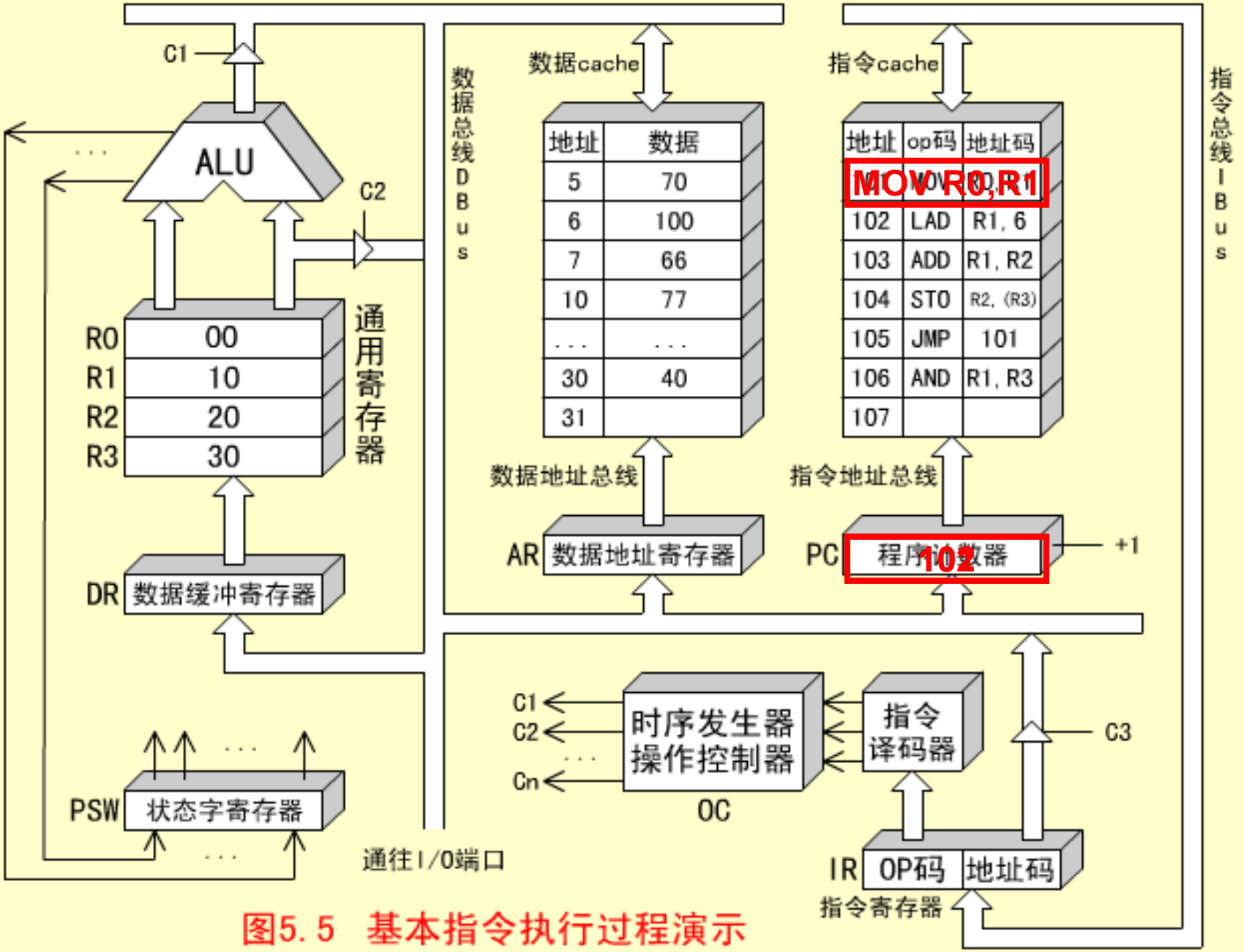


图5.5 基本指令执行过程演示

① 程序计数器PC中装入第一条指令地址101（八进制）；

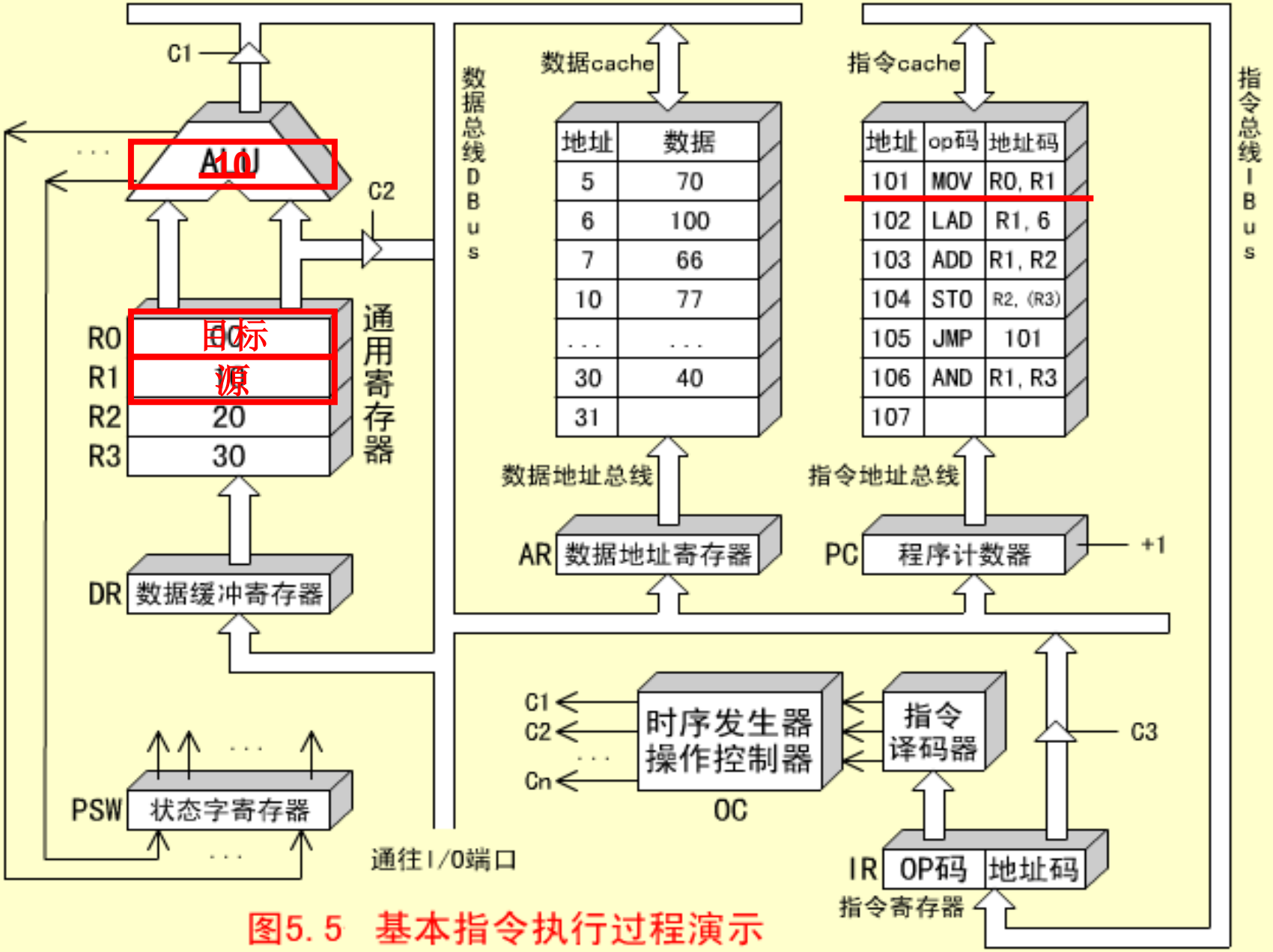
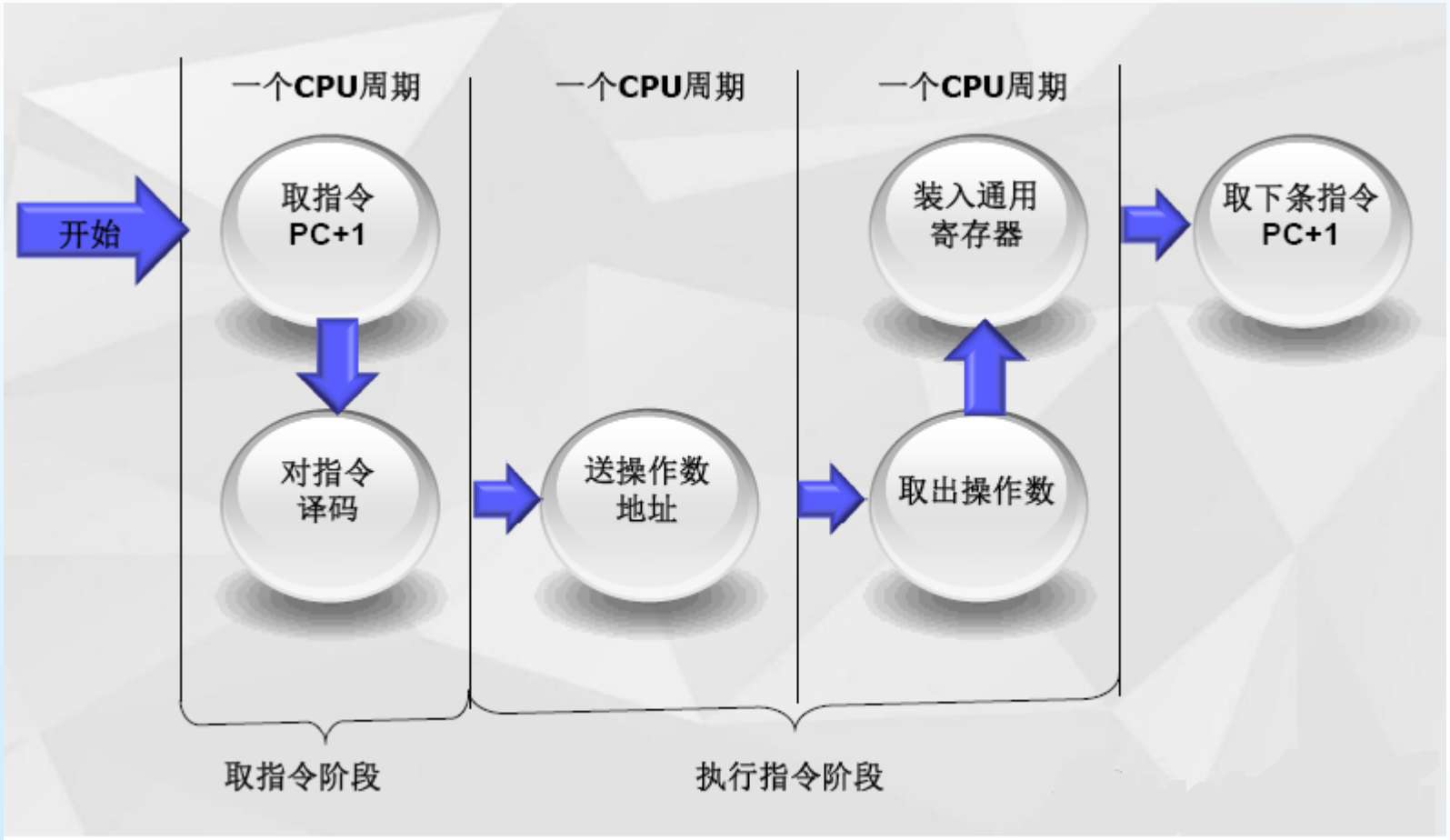


图5.5 基本指令执行过程演示

① 操作控制器（OC）送出控制信号到通用寄存器，选择R1（10）作源寄存器，选择R0作目标寄存器；

- 分析：为什么LAD指令周期包含三个CPU周期？



LAD R1，6

分析：为什么**LAD**指令周期包含三个**CPU**周期？

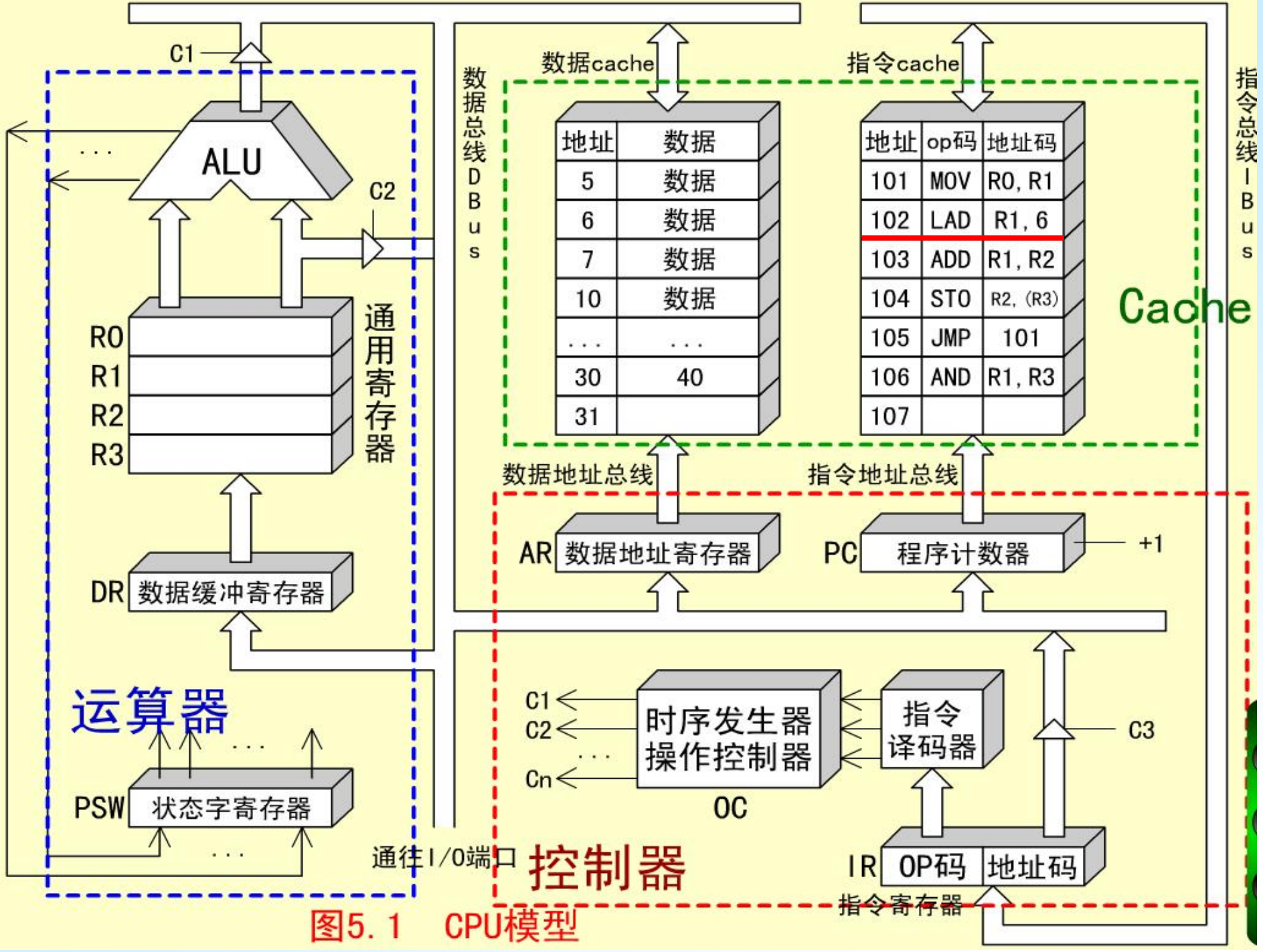
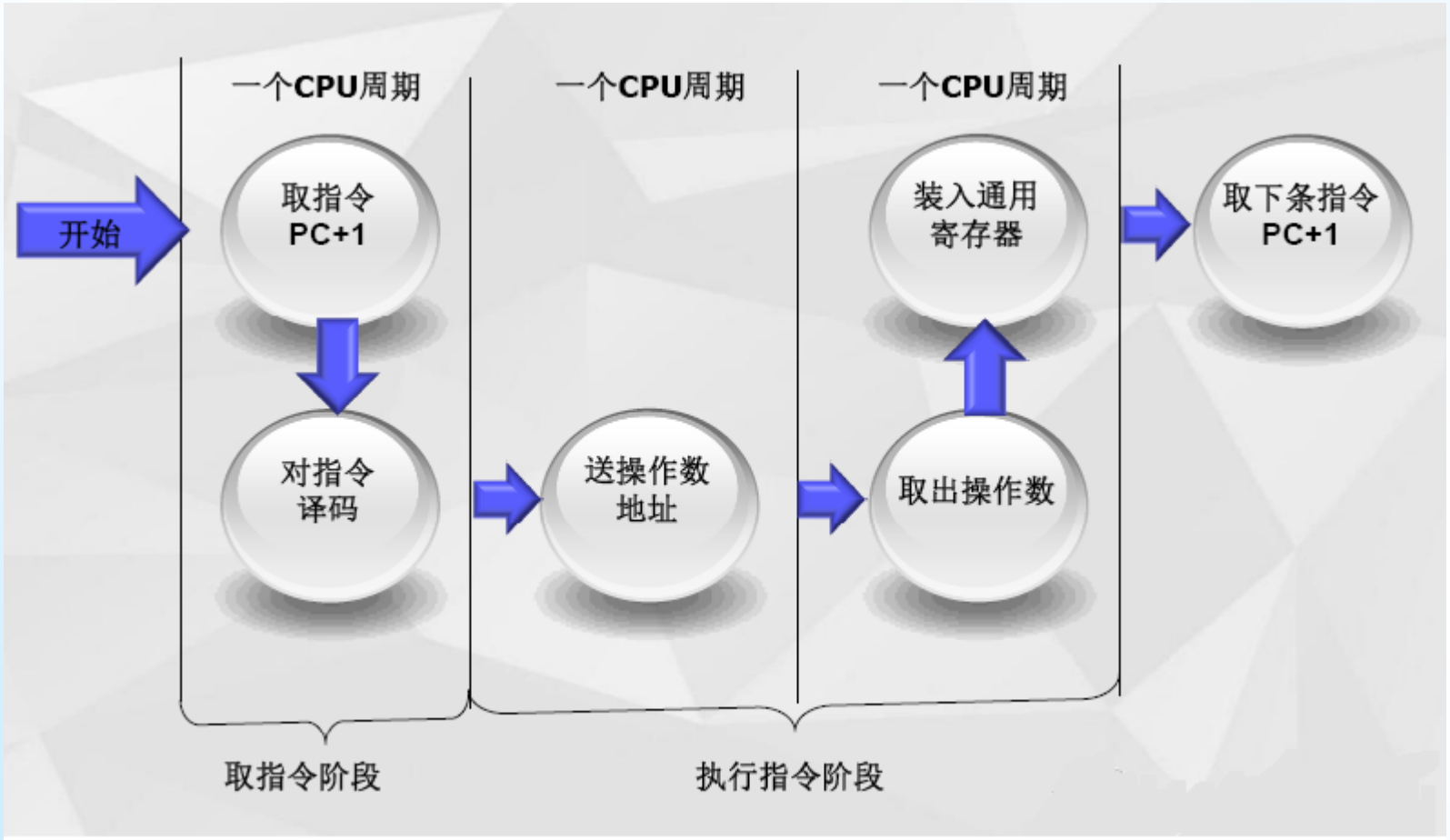


图5.1 CPU模型

- 分析：为什么LAD指令周期包含三个CPU周期？

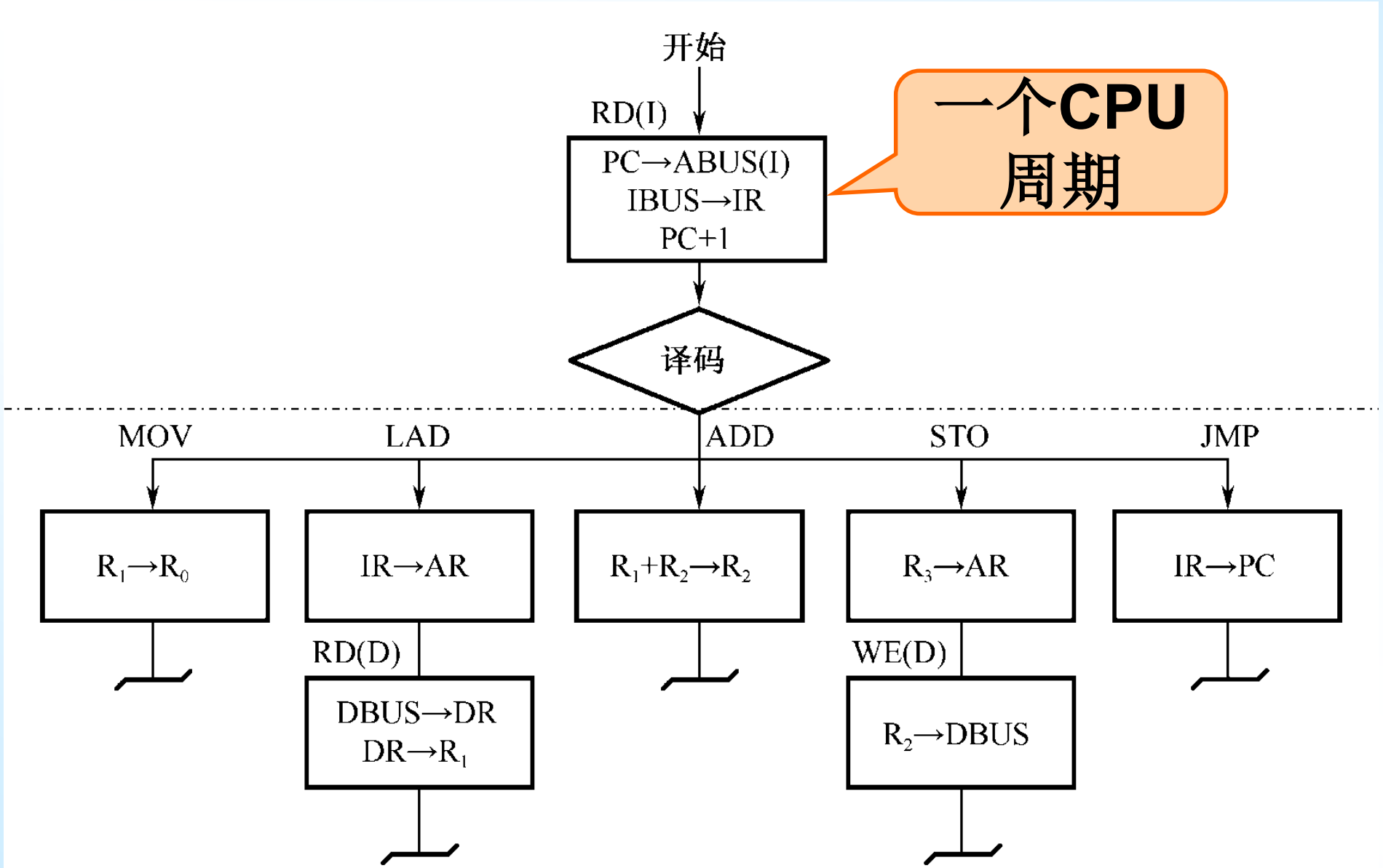


LAD R1，6

◊ 答：DBUS上分时进行地址传送和数据传送，一个数据通路占用一个CPU周期。

用方框图语言表示的指令周期

- 绘制指令周期流程图目的：控制器设计
- 具体实现方法：
  - 方框——CPU周期（机器周期）
  - 方框内容——数据通路操作或控制操作
  - 菱形符号——判别或测试（不独占CPU周期，依附于上个方框的CPU周期）
  - ~——公操作（处理外设，第8章学习）


开始
RD(I)
PC→ABUS(I)
IBUS→IR
PC+1
译码
一个CPU
周期
取指
执行
MOV
LAD
ADD
STO
JMP
R1→R0
IR→AR
R1+R2→R2
R3→AR
IR→PC
RD(D)
DBUS→DR
DR→R1
WE(D)
R2→DBUS

例题1：教材P162

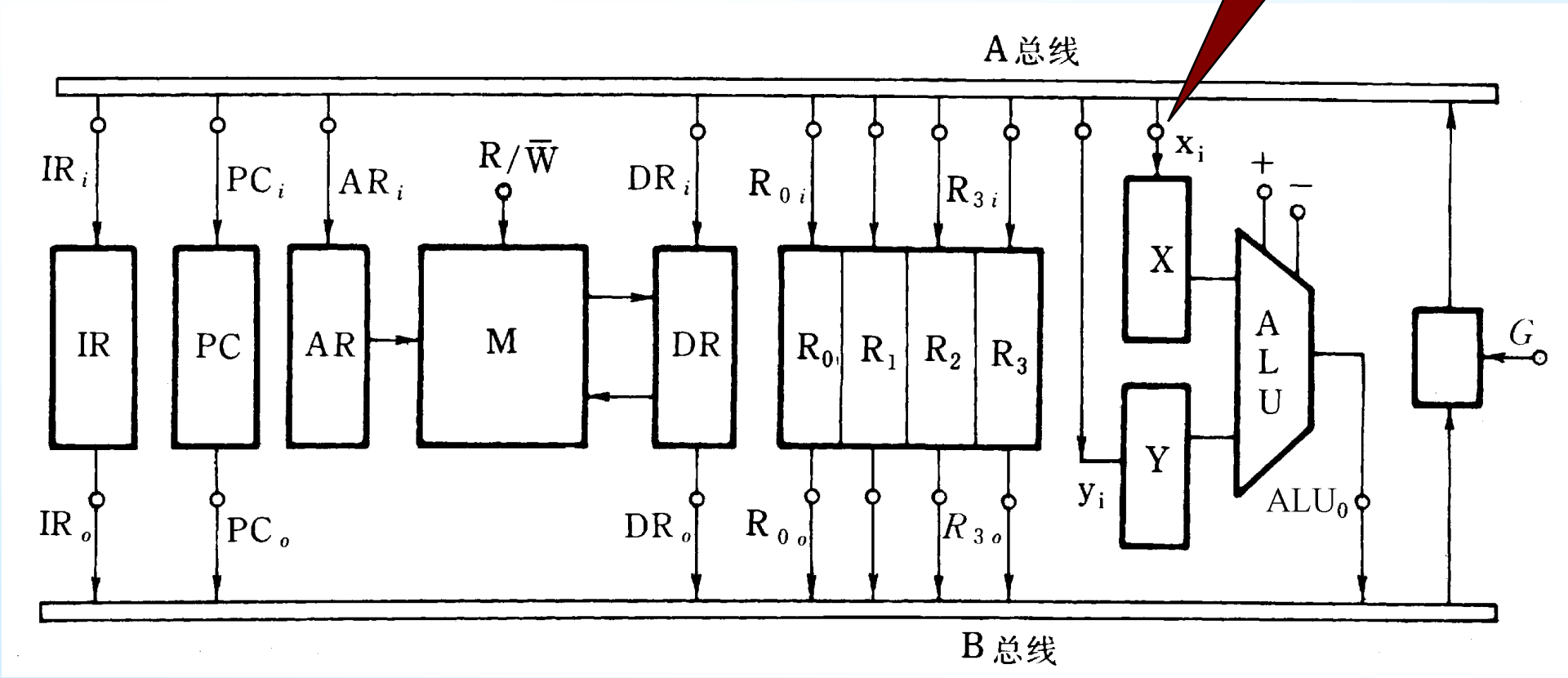


图5.16 双总线结构机器的数据通路

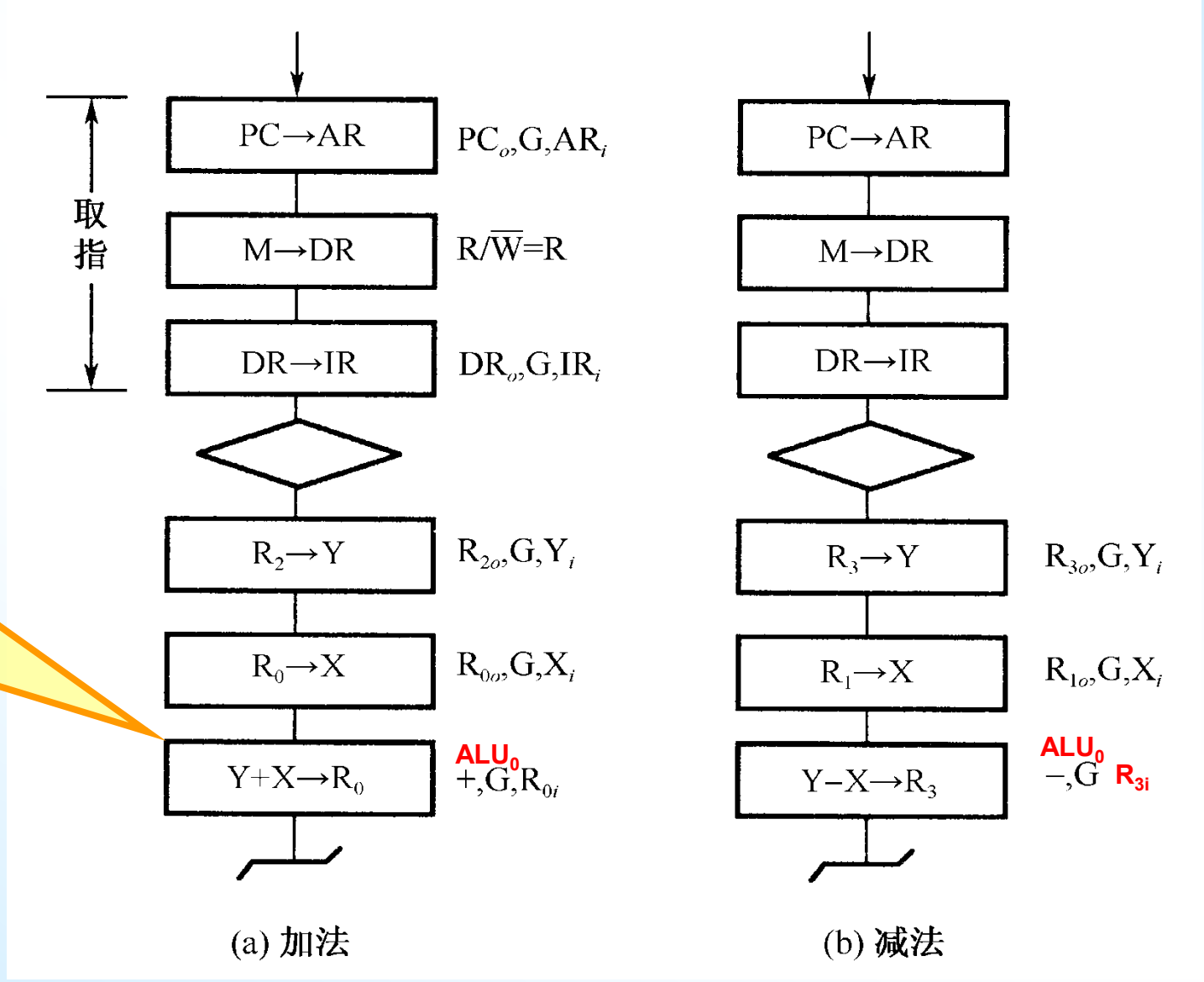
ADD R_2，R_0

$(R_0)+(R_2)\rightarrow R_0$

SUB R_1，R_3

$(R_3)-(R_1)\rightarrow R_3$

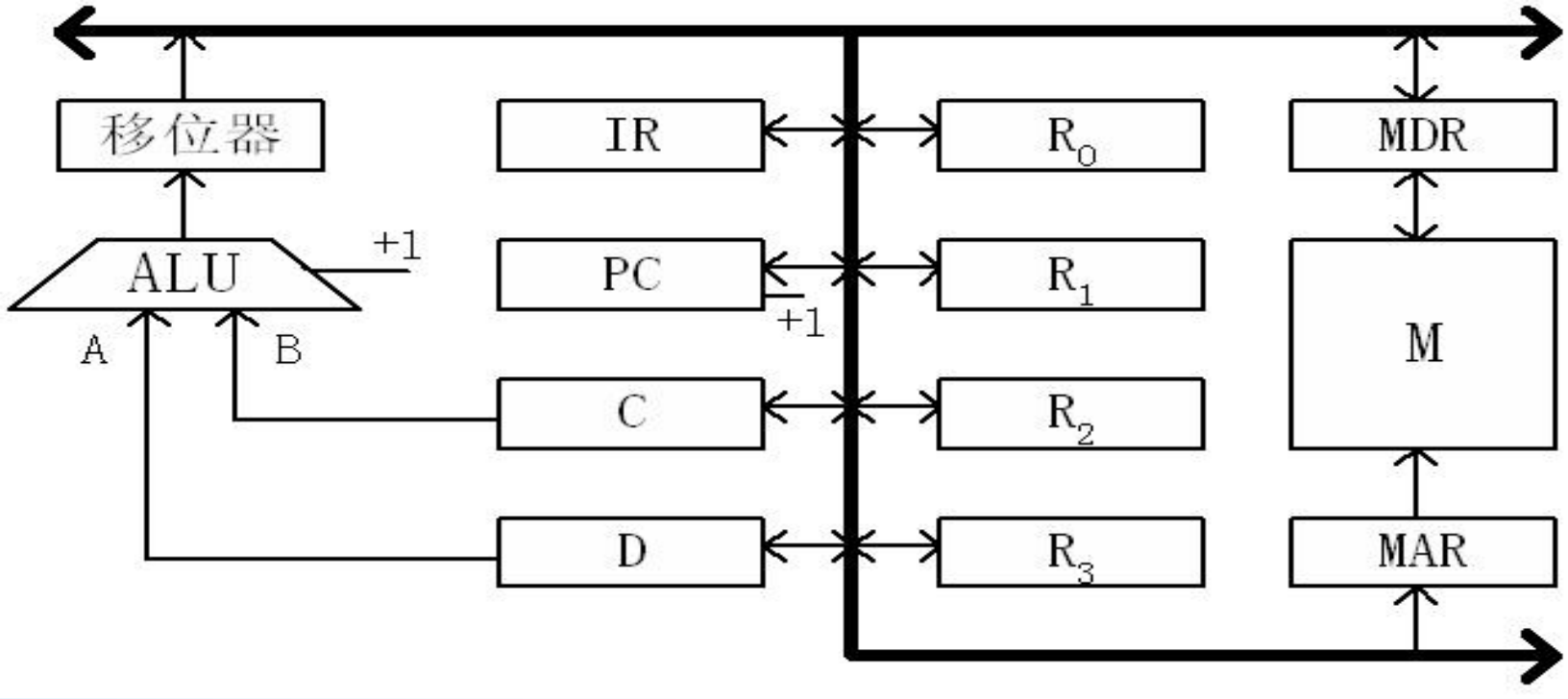
ADD R_2，R_0　$(R_0)+(R_2)\rightarrow R_0$　　SUB R_1，R_3　$(R_3)-(R_1)\rightarrow R_3$



注意
红色
勘误

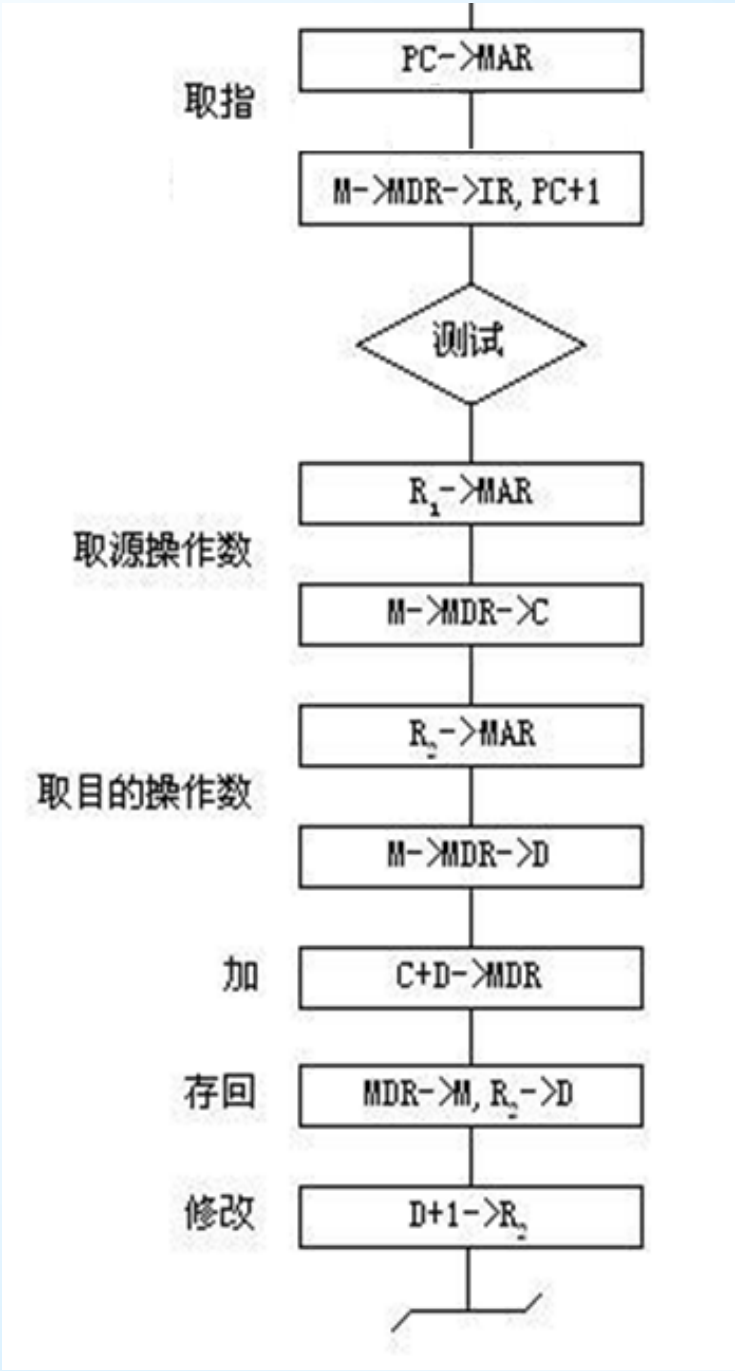
注意微操作控制信号，脚标 i 表示写入，o表示读出。桥耗费时间，使得寄存器传递数据占据了一个CPU周期

例题2 某假想机主要部件如图所示。



请按数据通路图，画出“ADD（R1），（R2）+”指令的指令周期流程图。该指令的含义是两个数进行求和操作。其中源操作地址在寄存器R1中，目的操作数寻址方式为自增型寄存器间接寻址（先取地址后加1）。

ADD（R1），（R2）+


取指
PC->MAR
M->MDR->IR, PC+1
测试
取源操作数
R1->MAR
M->MDR->C
取目的操作数
R2->MAR
M->MDR->D
加
C+D->MDR
存回
MDR->M, R2->D
修改
D+1->R2

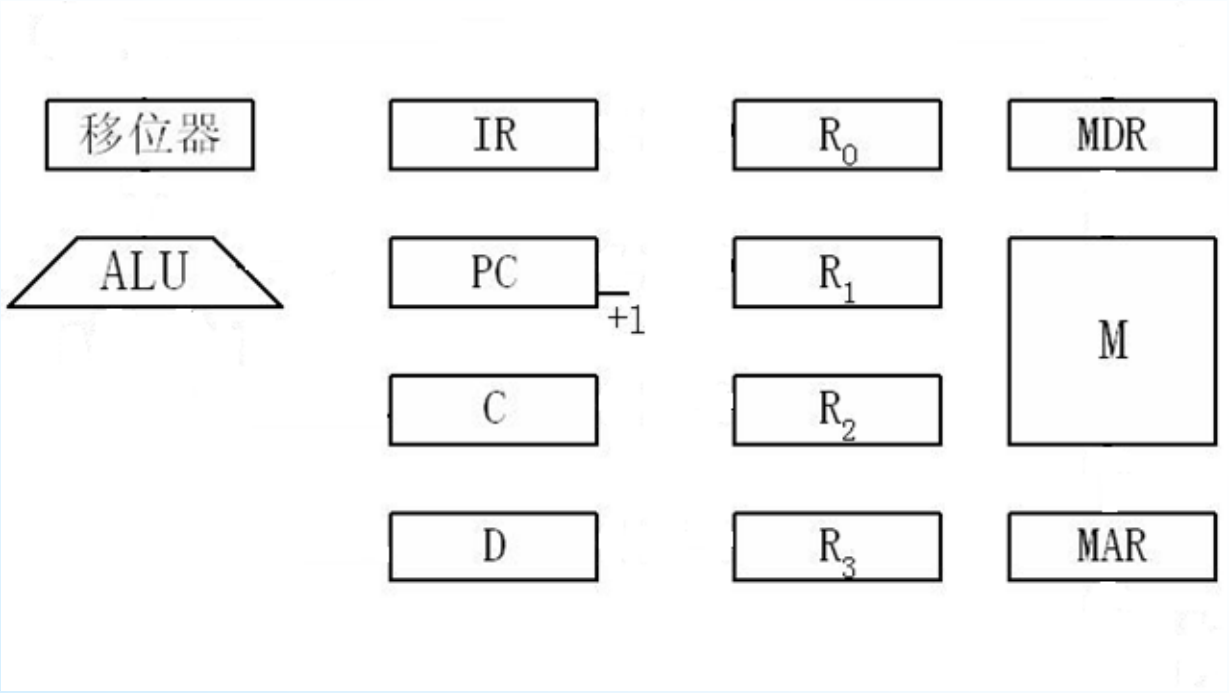
作业题：

P_{201} 2　　STO R_1，(R_2)

P_{201} 3　　LAD (R_3)，R_0

P_{202} 10　　ADD R_1，R_2

参见图P_{162} 5.16的数据通路



5.3 时序产生器和控制方式

- **5.3.1** 时序产生器作用和体制
- **5.3.2** 时序信号产生器
- **5.3.3** 控制方式

5.3.1 时序产生器作用和体制

1、日常生活中的时间控制

在日常生活中，我们学习、工作和休息都有一个严格的作息时间。

如在教学活动中，每个老师和学生都必须严格遵守作息时间和课表，在规定的时间里上课，在规定的时间里下课，否则就难以保证正常的教学秩序。

5.3.1 时序产生器作用和体制

2、计算机中的时间控制

CPU中也有一个类似“作息时间”的东西，它称为时序信号，使计算机可以准确、有条不紊地工作。

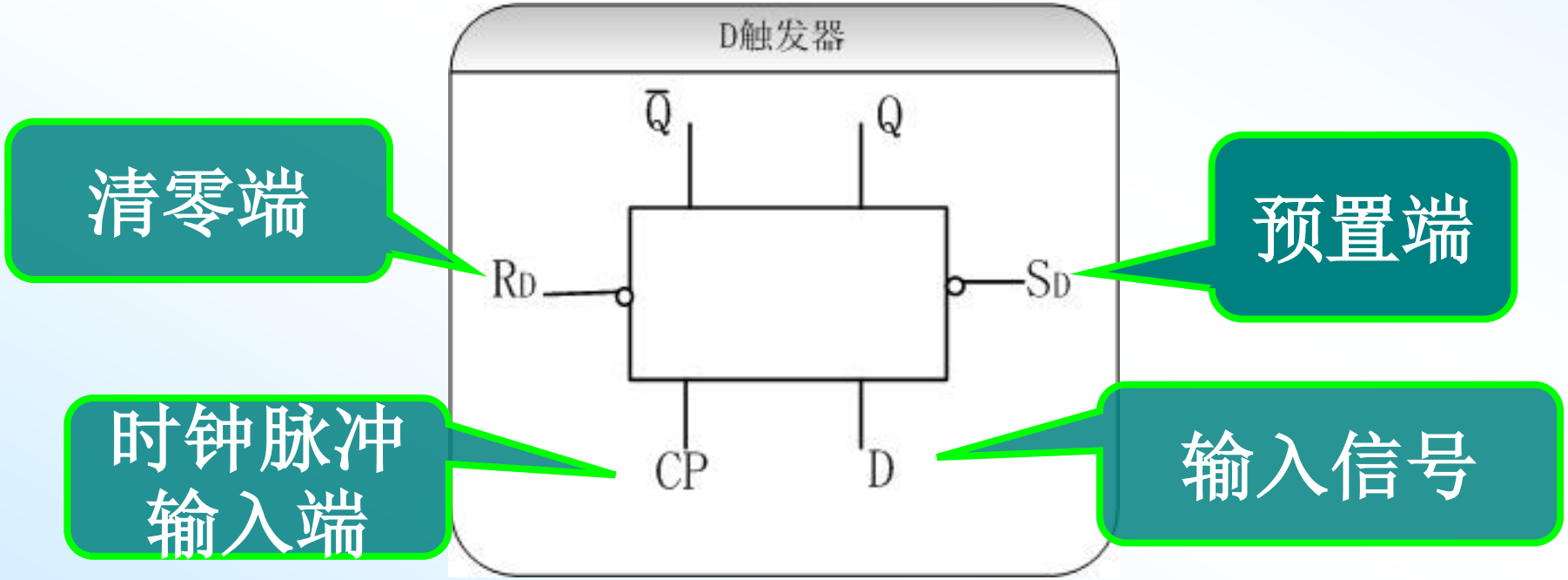
操作控制器发出不同时间间隔的各种定时信号（闹钟响），有条不紊地指挥各部件的动作，规定在这个信号到来时做什么，在那个信号到来时又做什么，给计算机各部分提供工作所需的各种时间标志（闹钟）。

5.3.1 时序产生器作用和体制

- 作用
  - 一个**CPU**周期中时钟脉冲对**CPU**的动作有严格的时序约束
  - 操作控制器发出的各种信号是时间（时序信号）和空间（部件操作信号）的函数

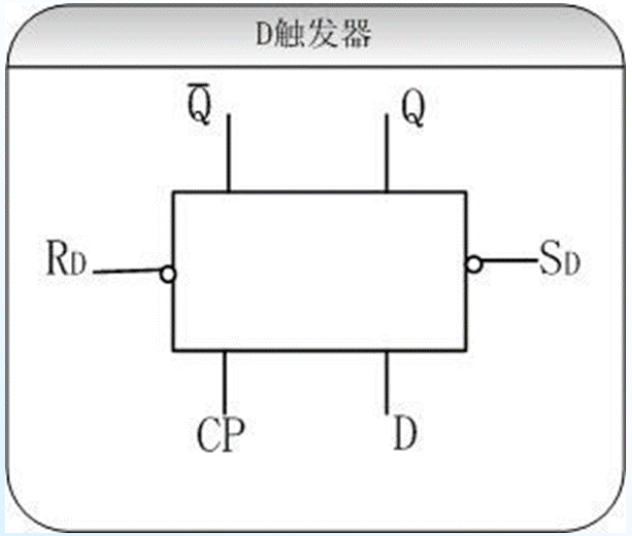
5.3.1 时序产生器作用和体制

- 体制
  - 组成计算机硬件的器件特性决定了时序信号的基本体制是电位—脉冲制（以触发器为例）



5.3.1 时序产生器作用和体制

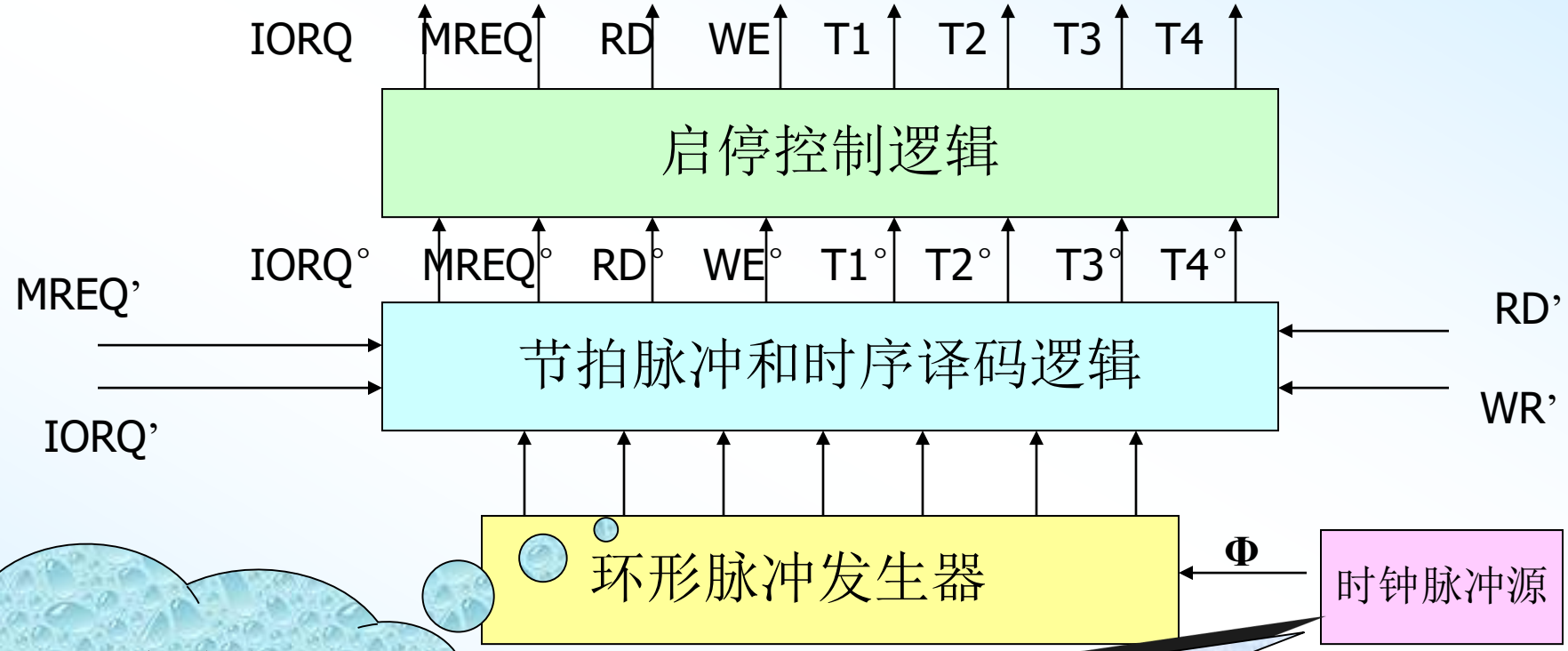
- 特性方程如下：
  - **D=0时，CP上升沿到来时，D触发器状态置0**
  - **D=1时，CP上升沿到来时，D触发器状态置1**



5.3.1 时序产生器作用和体制

- 硬布线控制器中
  - 时序信号采用**主状态周期——节拍电位——节拍脉冲** 三级体制
- 微程序控制中
  - 时序信号采用**节拍电位——节拍脉冲**二级体制

5.3.2 时序信号产生器



产生一组有序的间隔相等或不等的脉冲序列。通常采用循环移位寄存器。

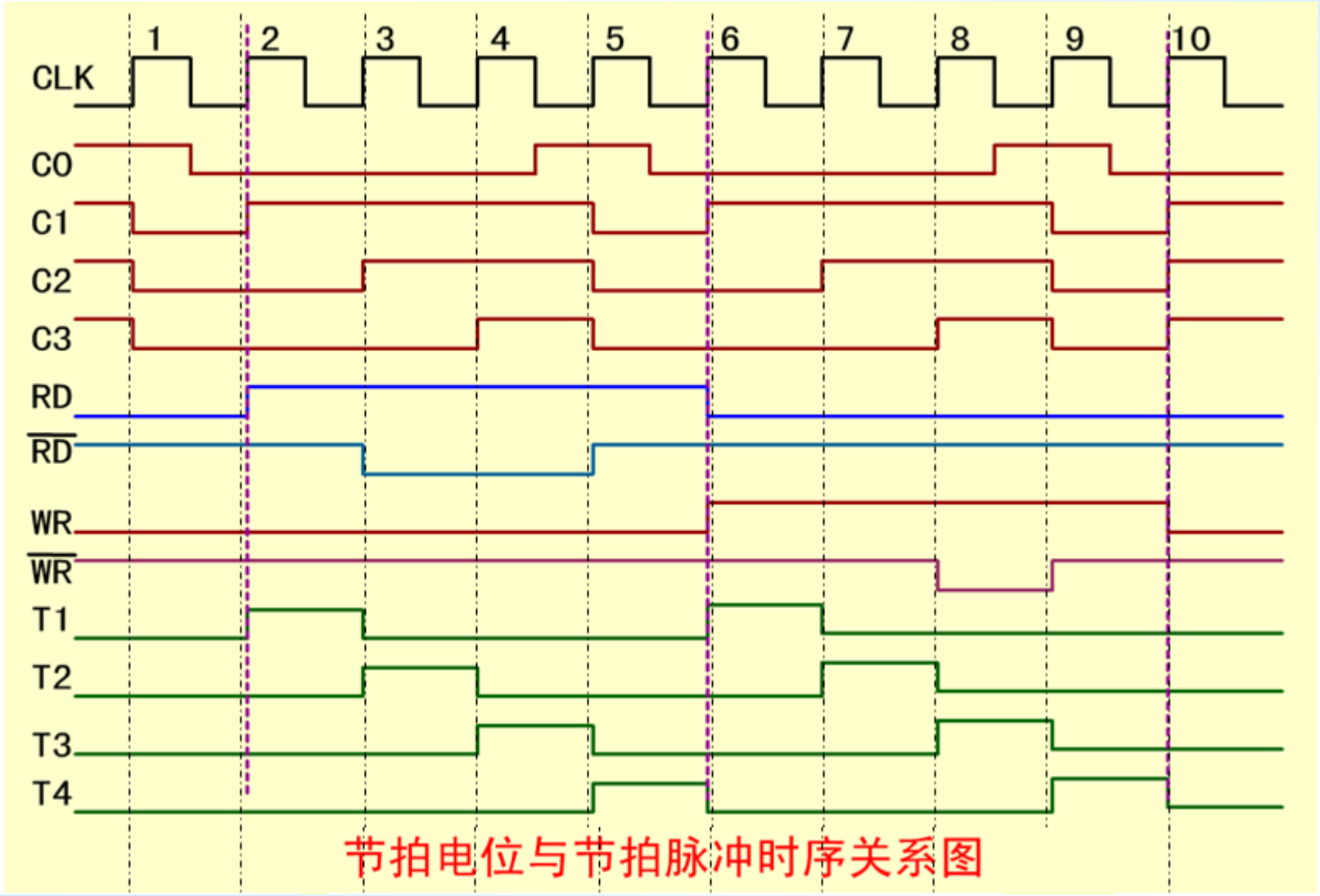
为环形脉冲发生器提供频率稳定且电平匹配的方波时钟脉冲信号，由石英晶体振荡器组成。

5.3.2 时序信号产生器

中央处理器

$$T_1^o = C_1 \cdot \overline{C_2}$$
$$T_2^o = C_2 \cdot \overline{C_3}$$
$$T_3^o = C_3$$
$$T_4^o = \overline{C_1}$$

C1：1 1 1 0
C2：0 1 1 0
C3：0 0 1 0



T1：1 0 0 0
T2：0 1 0 0
T3：0 0 1 0
T4：0 0 0 1

5.3.3 控制方式

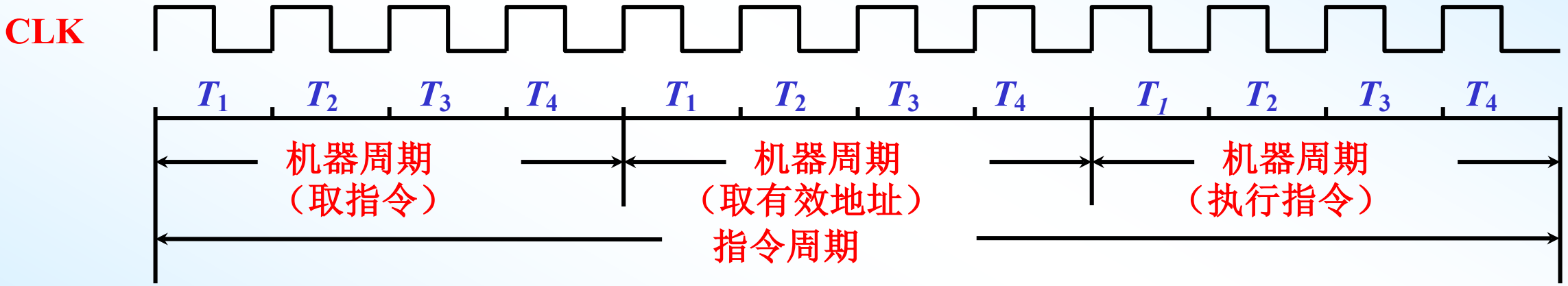
- 机器指令所包含的CPU周期数反映了指令的复杂程度，不同CPU周期的操作信号的数目和出现的先后次序也不相同。
- 控制方式：控制不同操作序列时序信号的方法。
- 分为以下几种：
  - 同步控制方式
  - 异步控制方式
  - 联合控制方式

5.3.3 控制方式

- 同步控制方式
  - 完全统一的机器周期执行各种不同的指令
  - 采用不定长机器周期
  - 中央控制与局部控制的结合
- 异步控制方式
  - 每条指令需要多长时间就占多长时间
- 联合控制方式
  - 大部分指令在固定的周期内完成，少数难以确定的操作采用异步方式
  - 机器周期的节拍脉冲固定，但是各指令的机器周期数不固定（微程序控制器采用）

1. 同步控制方式

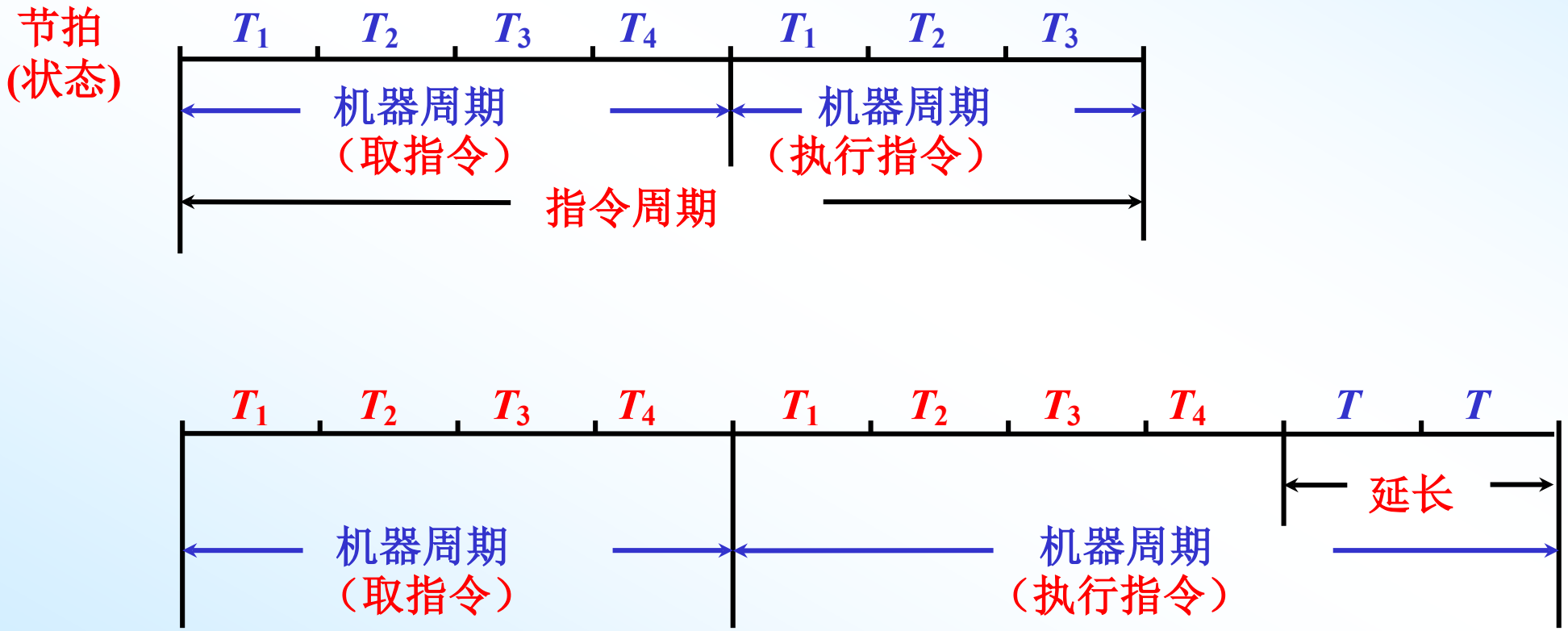
任一微操作均由 统一基准时标 的时序信号控制



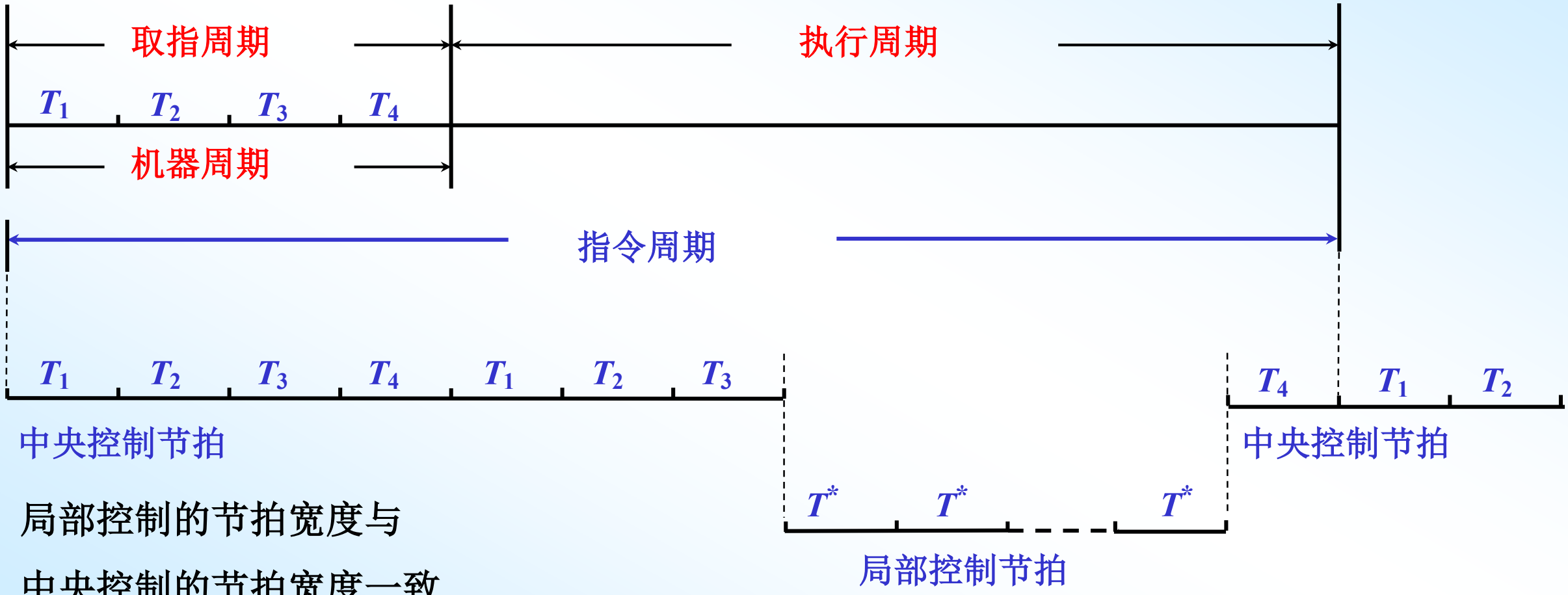
(1) 采用 完全统一 的机器周期和节拍

以 最长 的 微操作序列 和 最繁 的微操作作为 标准

(2) 采用不同节拍的机器周期



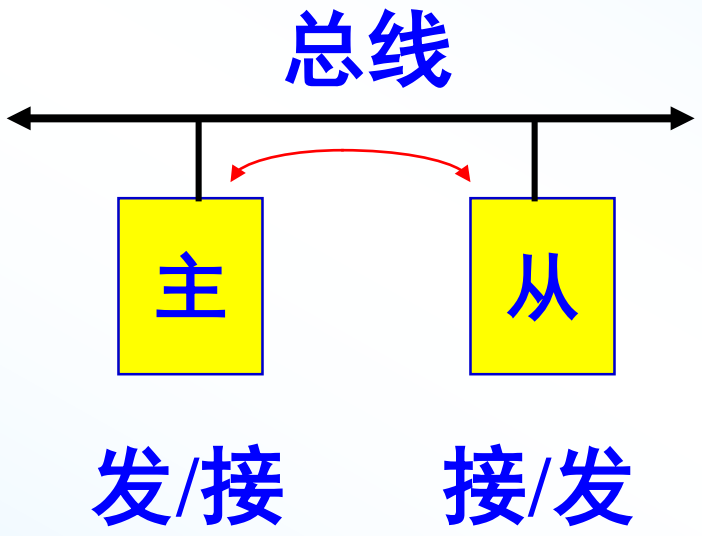
(3) 采用中央控制和局部控制相结合的方法



局部控制的节拍宽度与

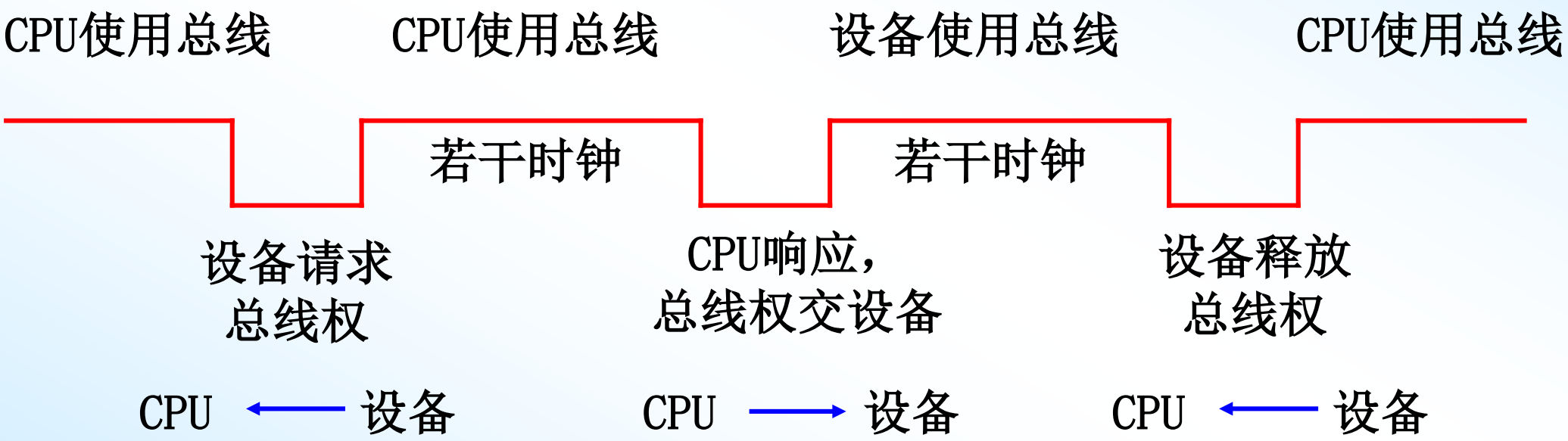
中央控制的节拍宽度一致

2. 异步控制方式



3. 联合控制方式

例：8088用一根总线请求/应答线实现总线权的转移。



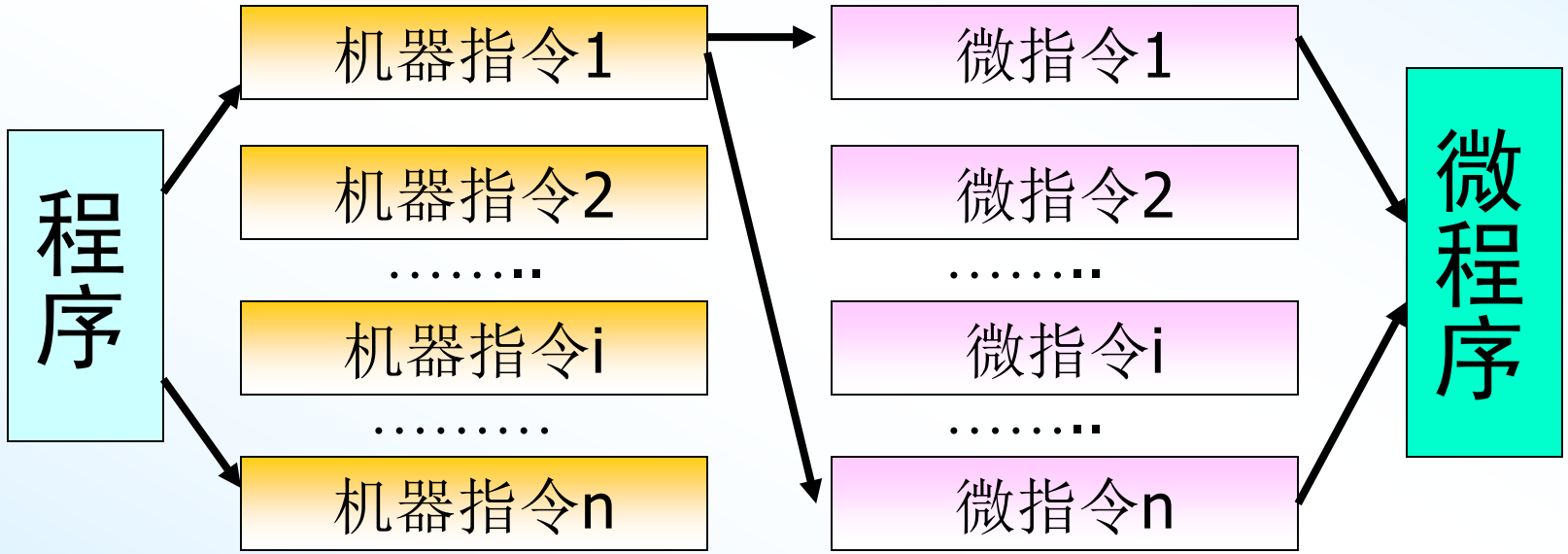
5.4 微程序控制器

5.4.1 微程序控制原理

5.4.2 微程序设计技术

5.4 微程序控制器——基本概念

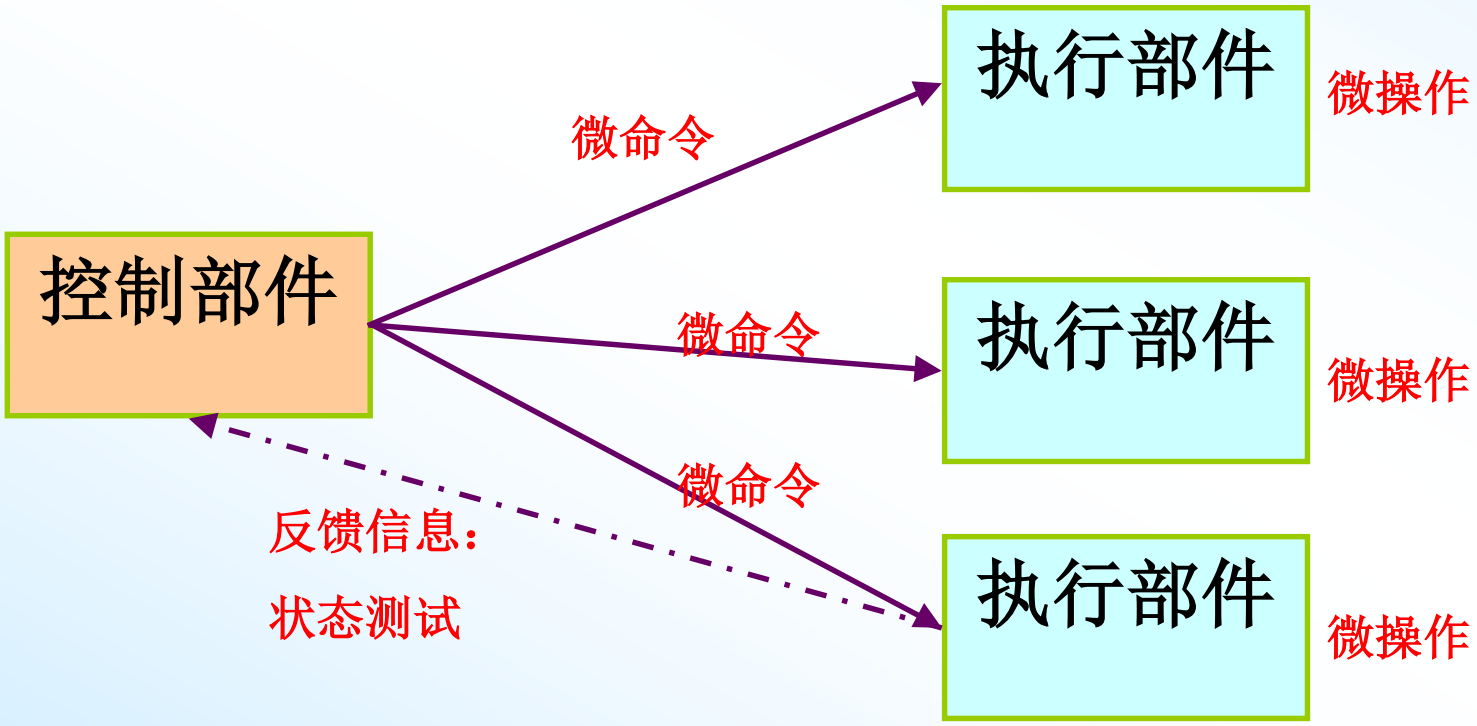
1951 英国剑桥大学教授　Wilkes



- 微程序控制器的基本思想
  - 仿照通常的解题程序的方法，把操作控制信号编成微指令，存放到一个只读存储器里。
  - 当机器运行时，一条又一条地读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执行所规定的操作。

5.4.1微程序控制原理

CPU中的功能部件可以划分为两大类：

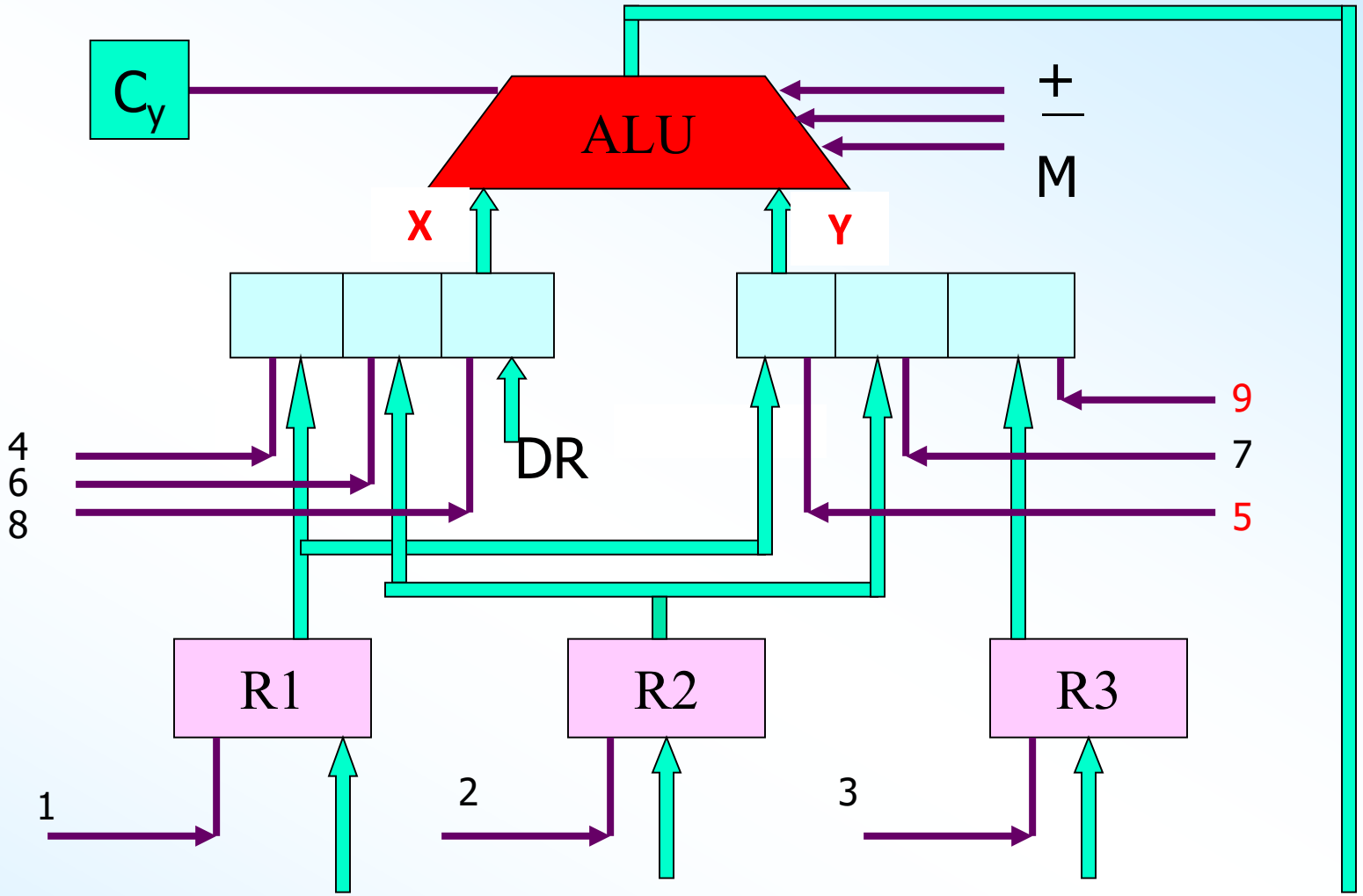


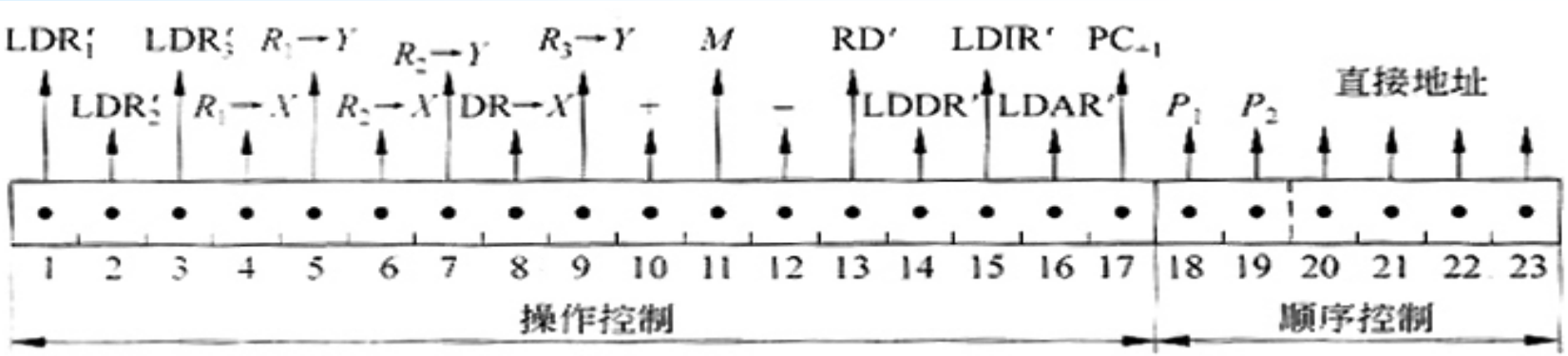
5.4.1微程序控制原理

- 1、微命令：控制部件向执行部件发出的各种控制命令叫作微命令，它是构成控制序列的最小单位。
  - 例如：打开或关闭某个控制门的电位信号、某个寄存器的打入脉冲等。
  - 微命令是控制计算机各部件完成某个基本微操作的命令。
  - 什么是微操作？

5.4.1微程序控制原理

- 2、微操作：是微命令的操作过程。
  - 微命令和微操作是一一对应的。
  - 微命令是微操作的控制信号，微操作是微命令的操作过程。
  - 微操作是执行部件中最基本的操作。
- 举例：
  - 控制门电位信号的变化、寄存器输入端的控制、ALU的基本执行过程...


Cy
ALU
±
M
X
Y
DR
4
6
8
9
7
5
R1
R2
R3
1
2
3


LDR1
LDR2
LDR3
R1→X
R2→X
DR→X
R1→Y
R2→Y
R3→Y
+
M
−
RD'
LDDR'
LDIR'
LDAR'
PC+1
P1
P2
直接地址
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
操作控制
顺序控制

5.4.1微程序控制原理

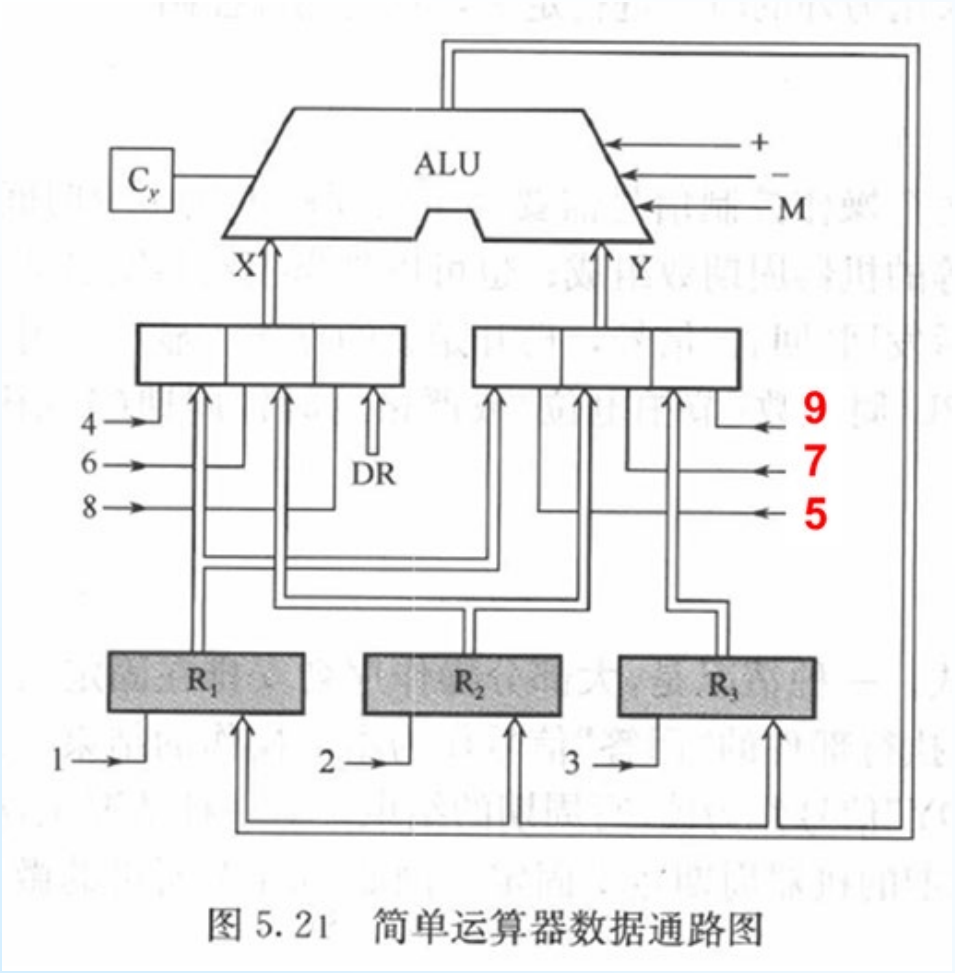


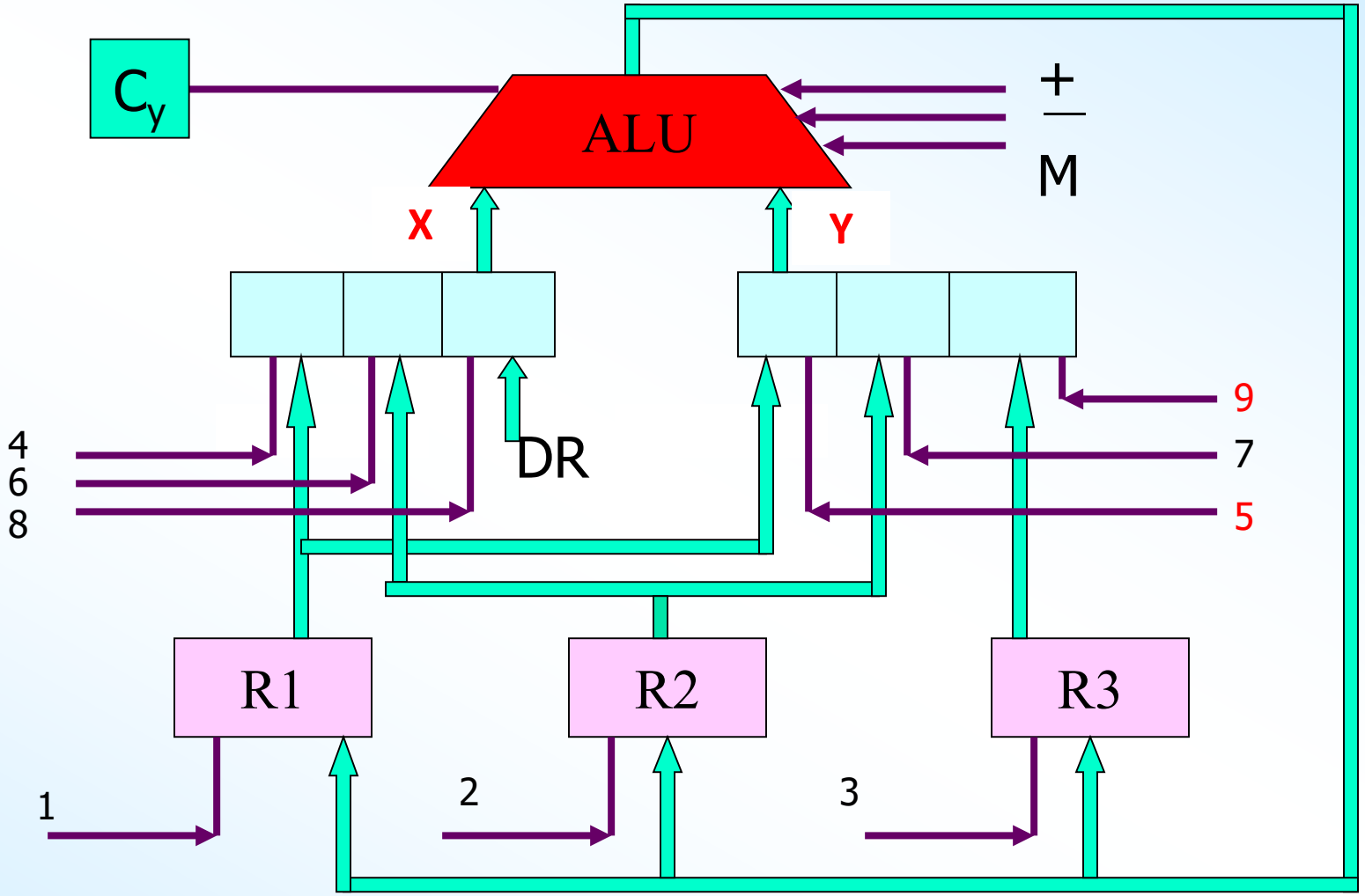
图 5.21　简单运算器数据通路图

微命令	微操作
1	ALU→ R1
2	ALU→ R2
3	ALU→ R3
4	RI →X
6	R2→X
8	DR →X
5	R1 →Y
7	R2 →Y
9	R3 →Y
+	ADD
-	SUB
M	MOV

5.4.1微程序控制原理

由于数据通路的结构关系，微操作可分为相容的和相斥的两种：

- 相斥的微操作，指不能同时或不能在同一个节拍内并行执行的微操作。
- 相容的微操作，指能够同时或在同一个节拍内并行执行的微操作，必须各占一位。



相斥：（4 6 8） （5 7 9） （+ - M） 相容：（1 2 3）

5.4.1微程序控制原理

- **3、微指令（Microinstruction）：在机器的一个CPU周期中，一组实现一定操作功能的微命令的组合，构成一条微指令。**

  - 微指令：指在同一CPU周期内并行或并发执行的微操作控制信息集合。
  - 它是微命令的组合，微指令存储在控制器中的控制存储器中。

5.4.1微程序控制原理

- 4、微程序
  - 一系列微指令的有序集合就是微程序
    - 一段微程序对应一条指令
    - 微地址：存放微指令的控制存储器的单元地址
  - 举例：以简单运算器通路图的微指令格式为例：

思考：

下图的微指令有没有冗余？如何消除？

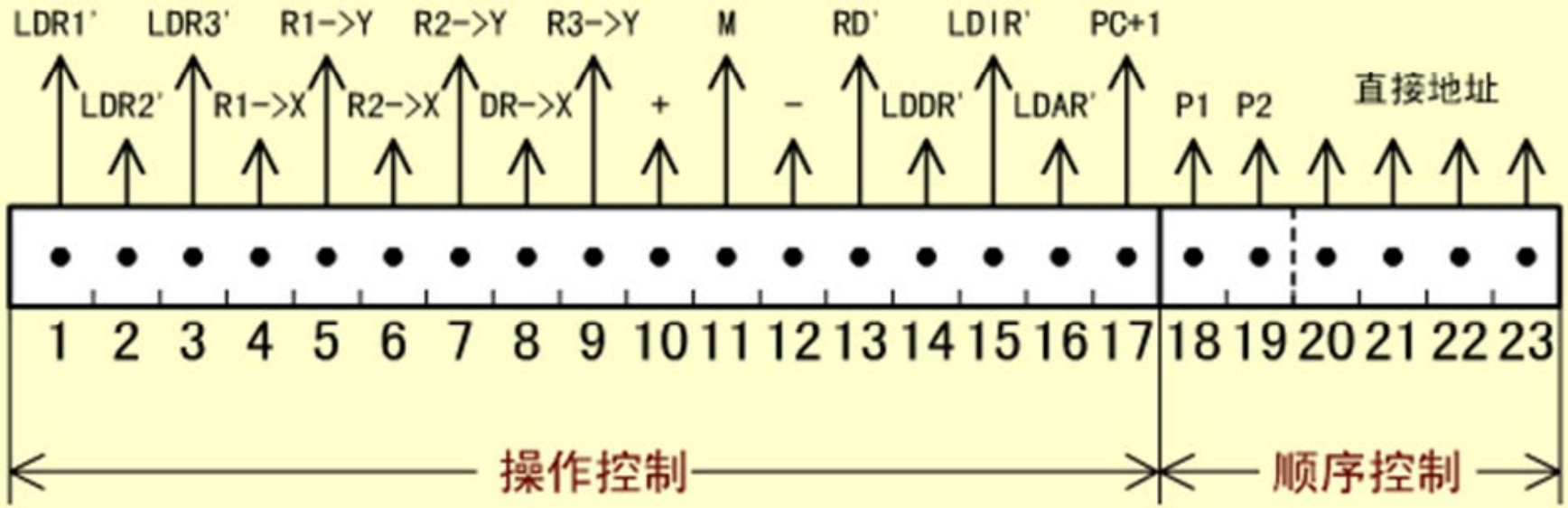
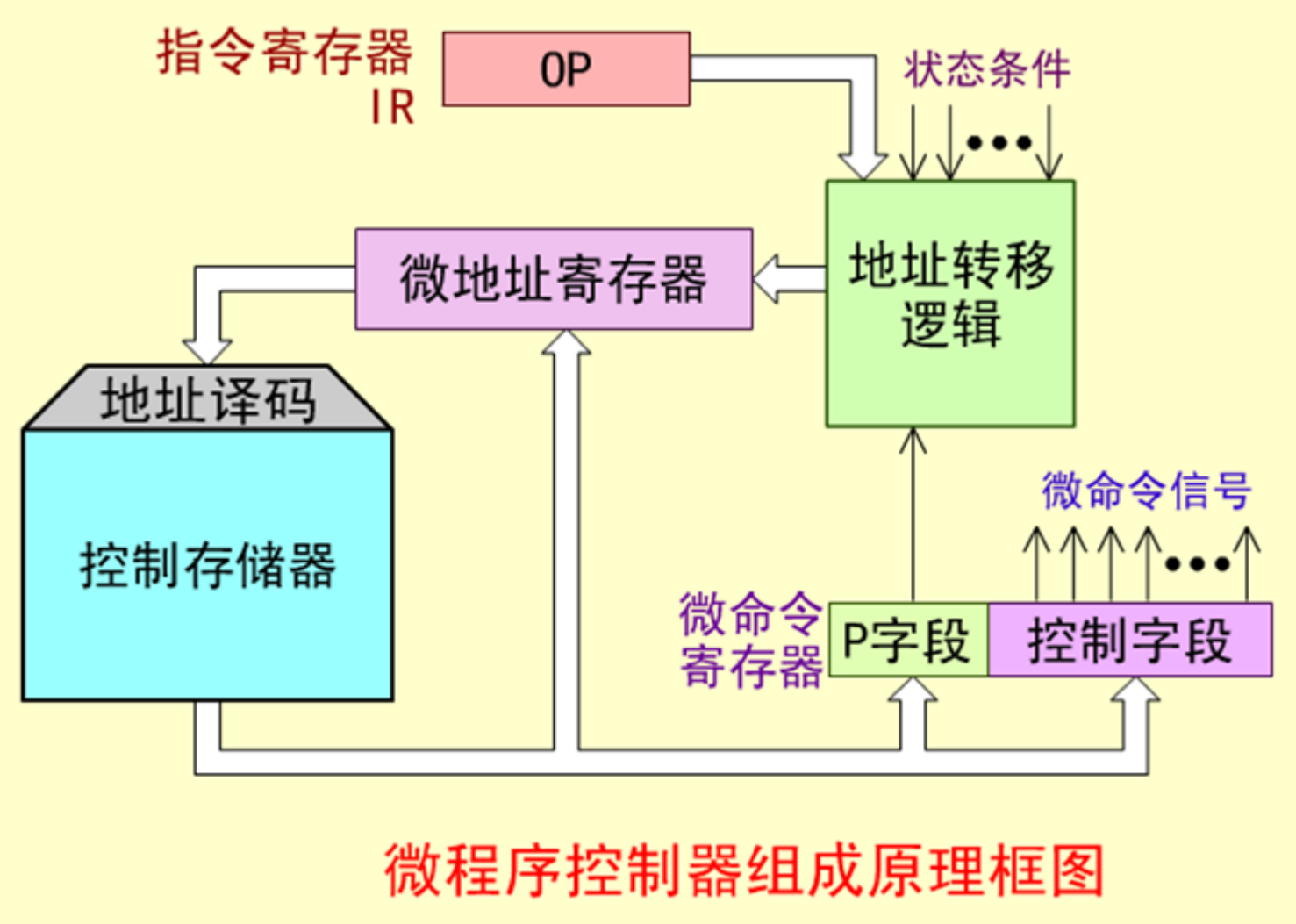


图 5.22 微指令基本格式

5.4.1微程序控制原理



微程序控制器组成原理框图

5.4.1微程序控制原理

- 设计阶段

画指令周期流程图

↓

编写微程序

↓

固化微程序

- 运行阶段

取指令、定位微程序段

↓

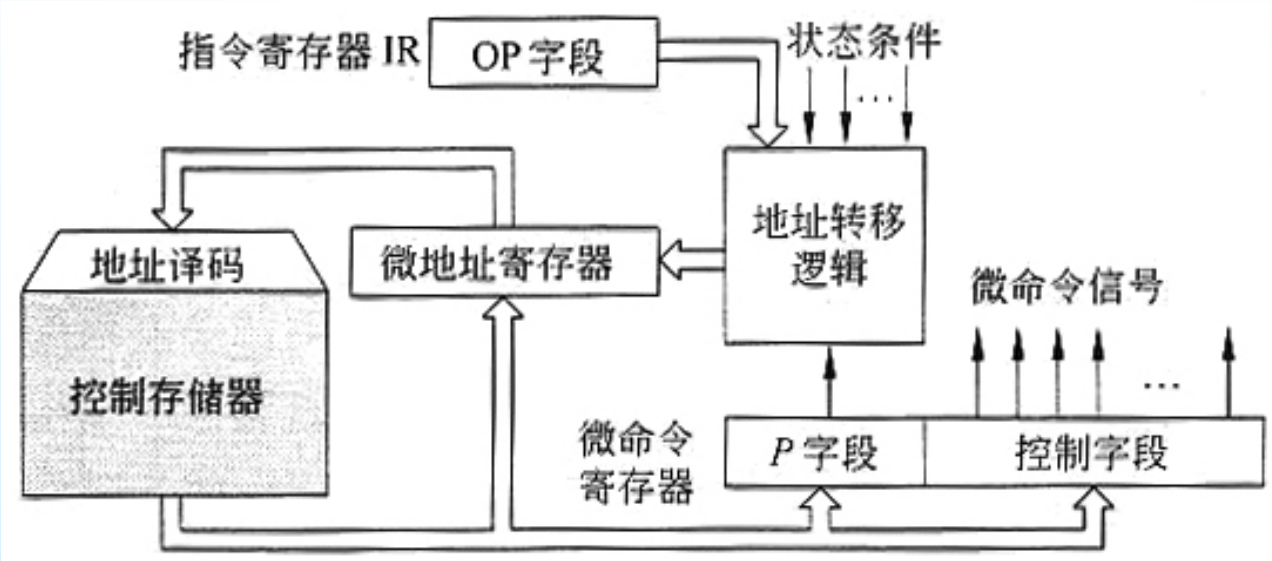
取微指令、译码产生微命令

↓

完成微操作

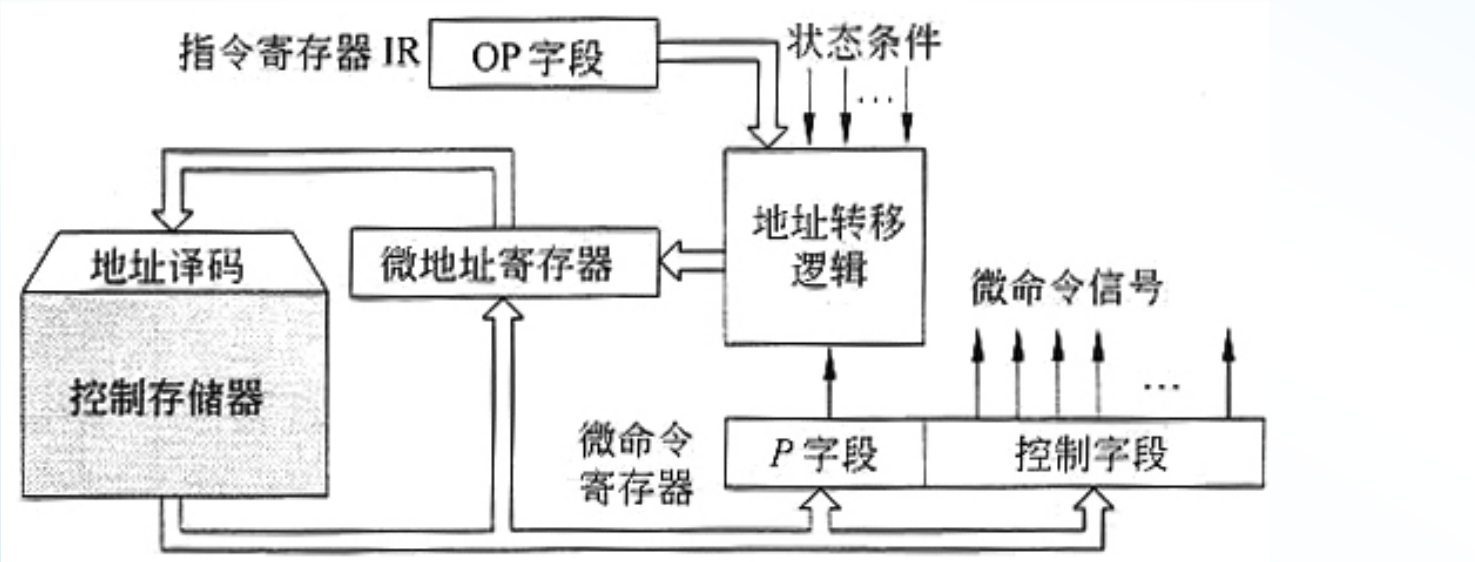
5.4.1微程序控制原理

设计阶段：首先，根据CPU的数据通路结构、指令操作定义等，画出每条指令的指令周期流程图（具体到每个时钟周期、微操作、微命令）。然后，根据微指令格式、指令周期流程图编写每条指令的微程序。最后，把整个指令系统的微程序（其中取指令的微程序段是公用的）固化到控制存储器中。



5.4.1微程序控制原理

运行阶段：首先，逐条执行取指令公用微程序段，控制取指令操作。然后，根据指令的操作码字段，经过变换，找到该指令所对应的特定微程序段，从控制存储器中逐条取出微指令，根据微操作控制字段，直接或经过译码产生微命令（控制信号），控制相关部件完成指定的微操作。一条微指令执行以后，根据微地址字段取下一条微指令。



5.4.1微程序控制原理

- 以一个典型例子说明微指令的工作过程
  - 首先做知识准备
  - 明确问题环境

5.4.1微程序控制原理

- **BCD码？**
- **用4位二进制数来表示1位十进制数中的0~9这10个数码**
- **Eg.（379）$_{10}$=（0011 0111 1001）$_{BCD}$**

5.4.1微程序控制原理

- 在我们的例子中：
  - **7+3=10**
- 用**BCD**码做加法期望得到：
  - **0111+0011= 0001 0000**
- 而实际
  - **0111+0011=1010**
- 用什么办法解决实际问题，满足应用需求？

5.4.1微程序控制原理

- 在图5.21运算器基础上的具体做法：

R1 + R2 + R3

a b 6

- 做加法：
  - **a+b+6有进位？**
  - 是：结果值正确
  - 否：结果值减6恢复a+b的真实结果

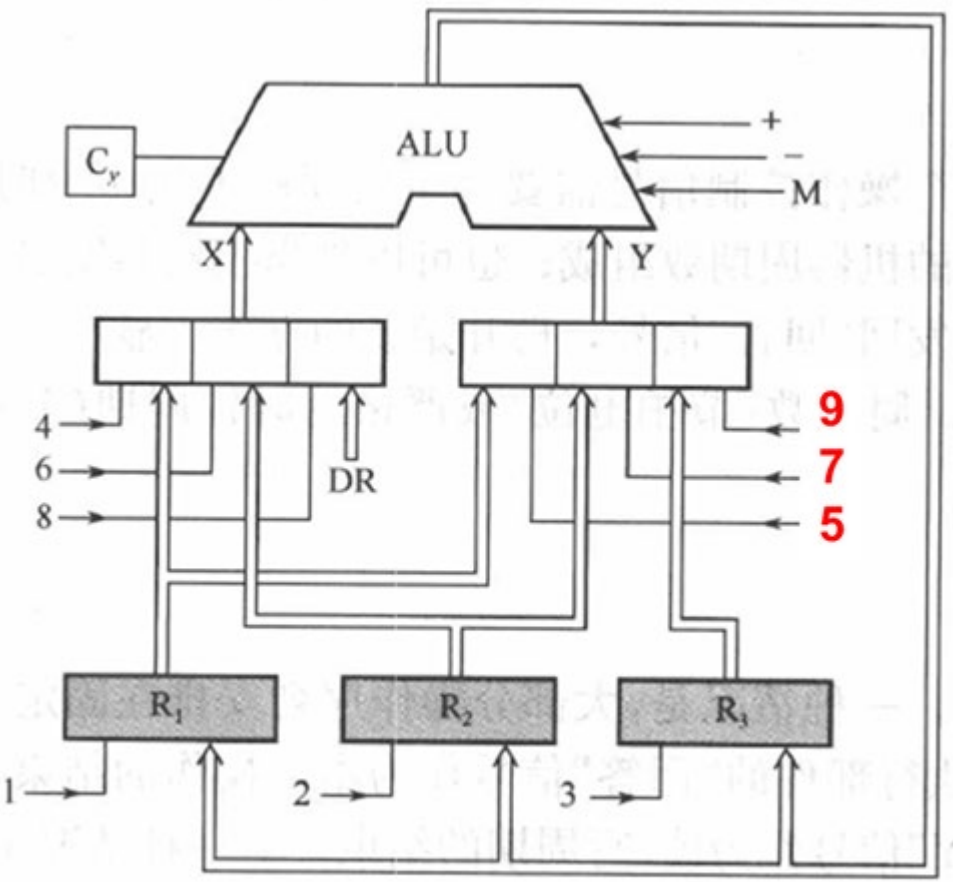


图 5.21　简单运算器数据通路图

5.4.1微程序控制原理

- 数据通路图
- 操作流程图

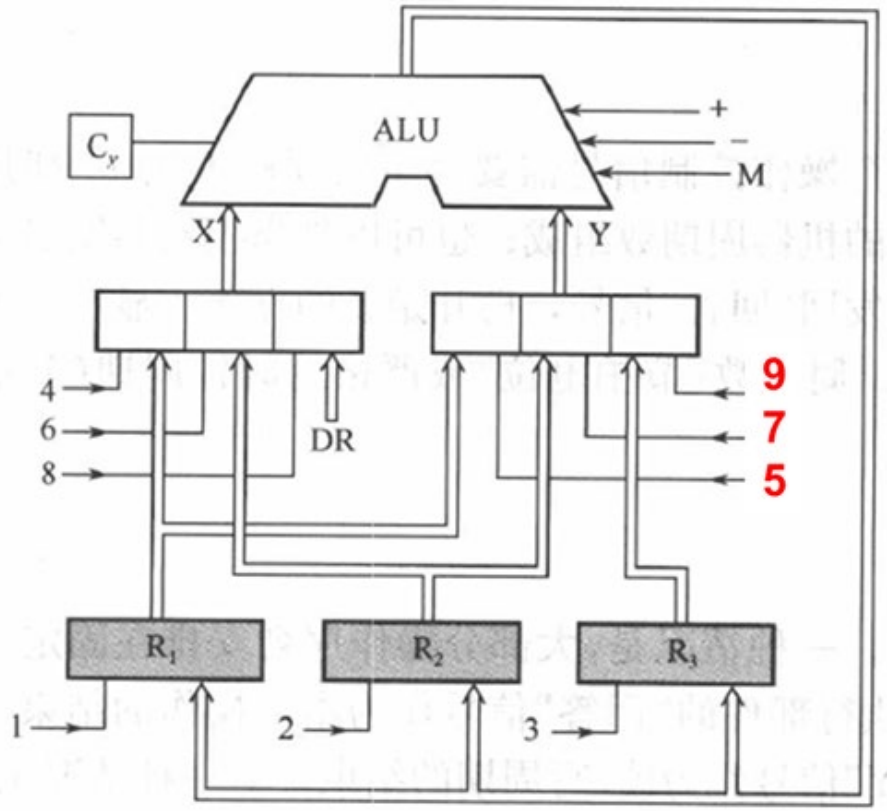


图 5.21　简单运算器数据通路图

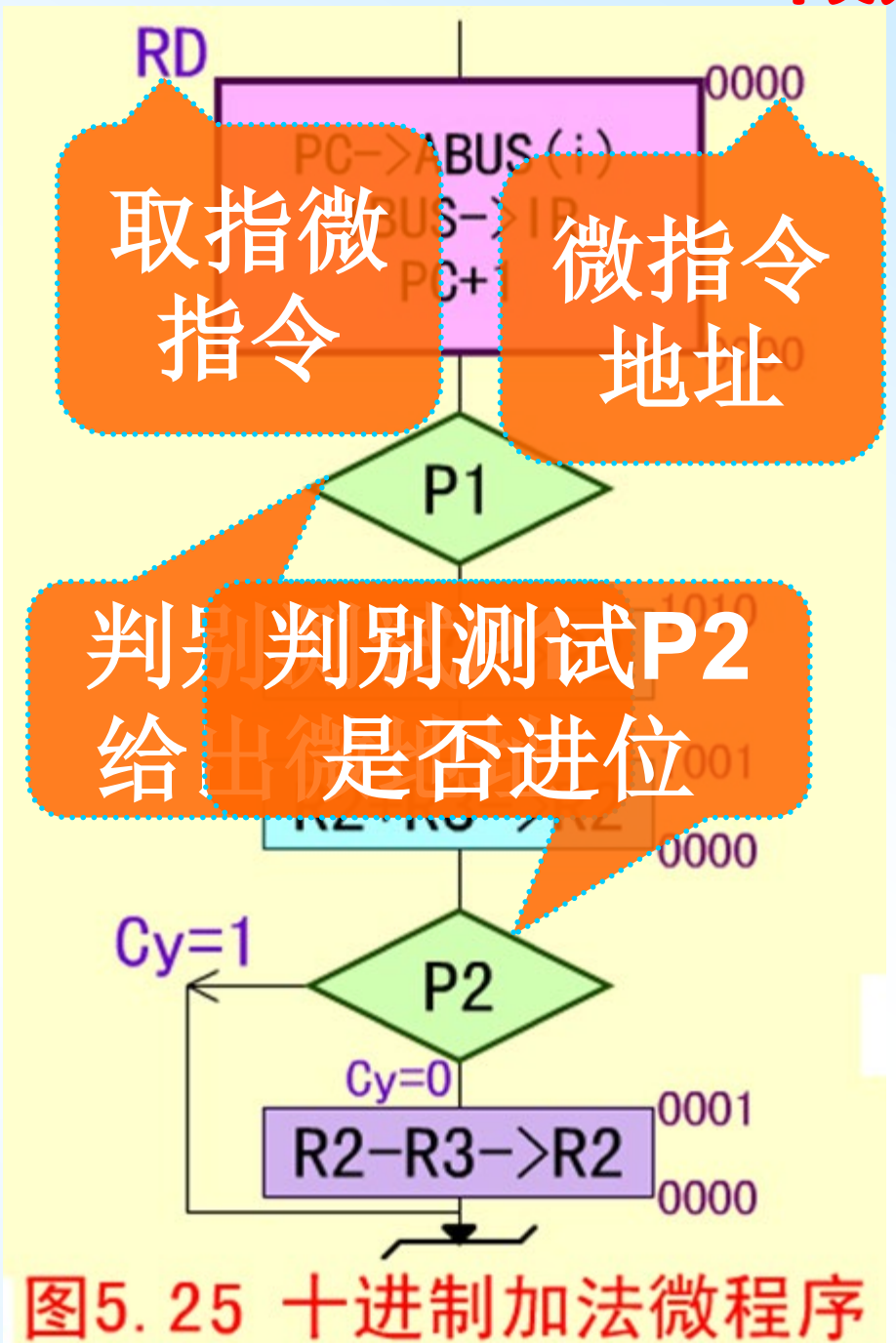
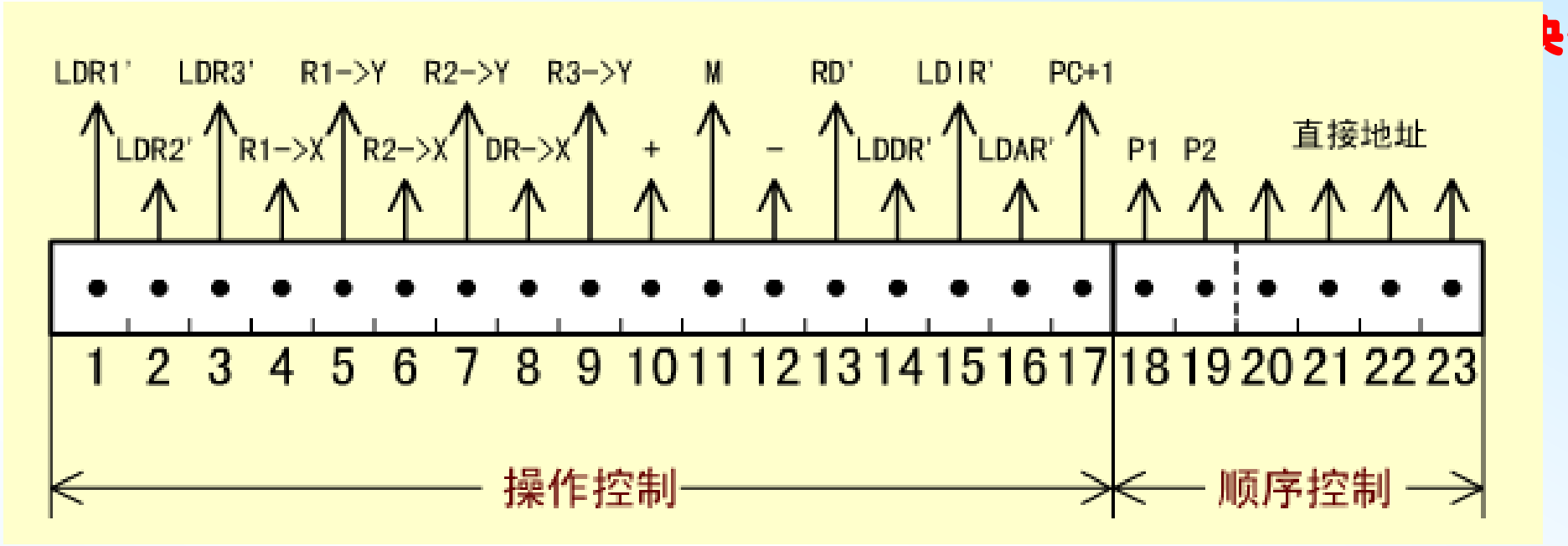


图5.25 十进制加法微程序



第1条微指令：取指令操作信号

0000

000	000	000	000	00101	10	0000

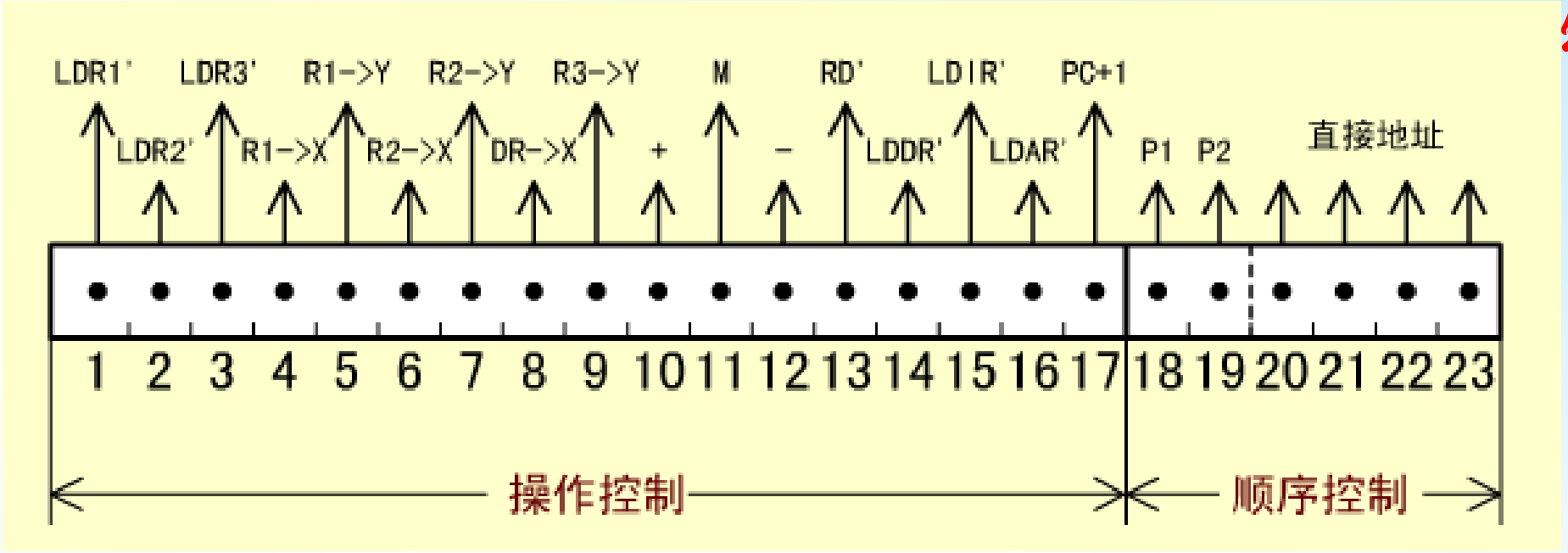
13.PC->ABUS（I）

15.LDIR’

17.PC+1

18.P1判别:操作码译码

“ADD”：1010



第2条微指令：R1+R2->R2

1010

010	100	100	100	00000	00	1001

2.存结果LDR2

4.R1->X

7.R2 -> Y

10.+

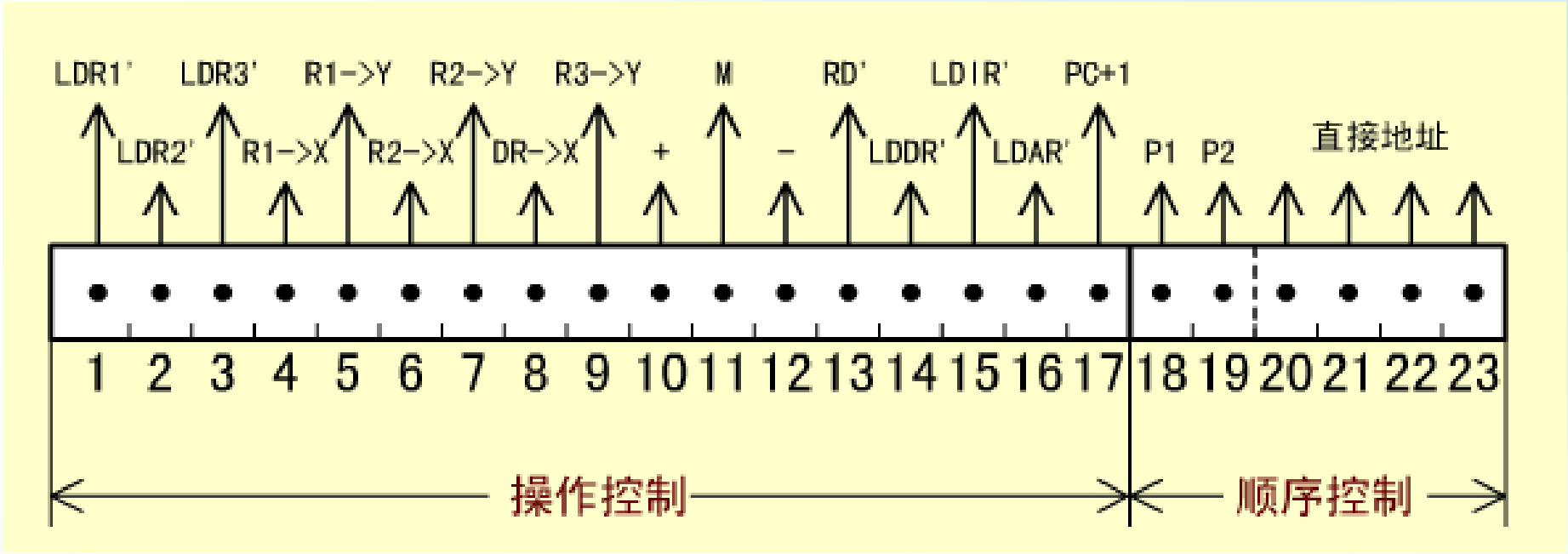
不判别测试，下一条微指令地址1001

说明：

a) 相容性微操作是指能够在同时或是同一个CPU周期内并行执行的微操作。

b) 相斥性微操作是指不能在同时或是同一个CPU周期内并行执行的微操作。

c) 为了保证寄存器能够接收到稳定有效的数据，需将微指令产生的节拍电位信号（如LDR1’,LDR2’,LDR3’等）与脉冲信号 相与 变为节拍脉冲信号。



第3条微指令：R2+R3->R2

1001

010	001	001	100	00000	01	0000

2.存结果LDR2

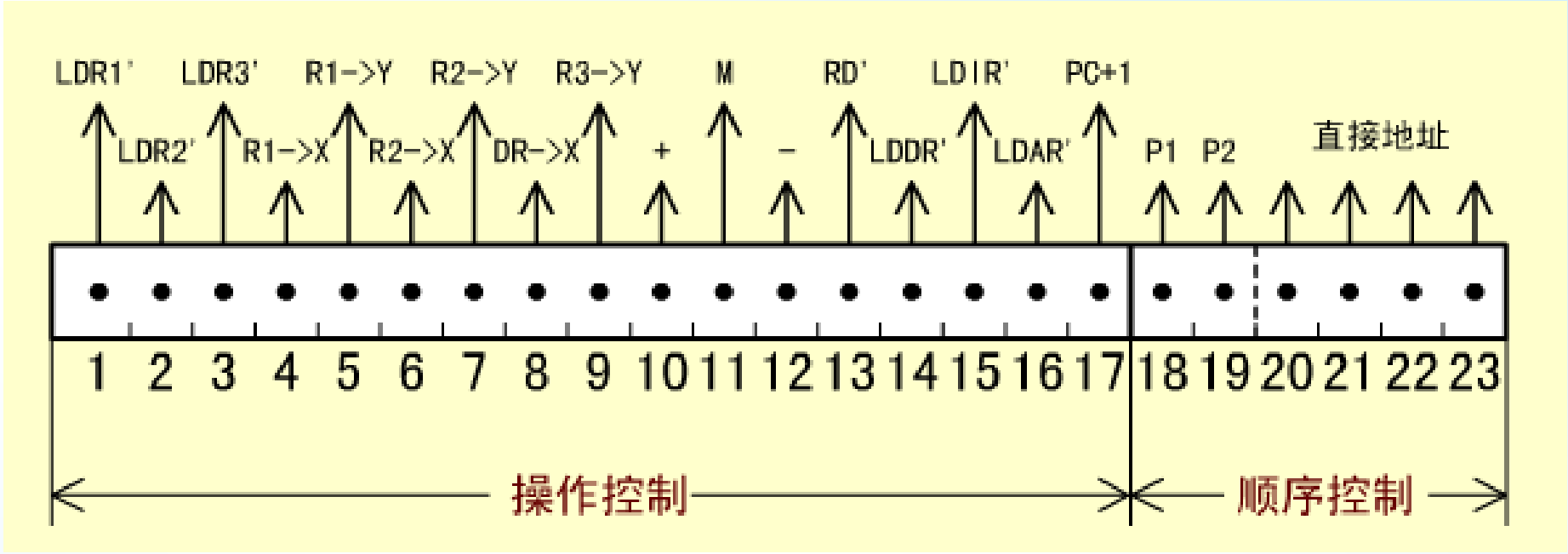
6.R2 -> X

9.R3 -> Y　10.+

P2判别：进位标志Cy，

Cy=0：　0001

Cy=1：　0000



第4条微指令：R2-R3->R2

0001

010	001	001	001	00000	00	0000

转地址0000，即取指令

2.存结果 LDR2

6.R2 -> X

9.R3 -> Y　12.-

24 下列关于机器指令与微指令关系的陈述中，正确的是（　　）。

A. 每条机器指令通过一条微指令解释执行

B. 每条机器指令由一段微程序解释执行

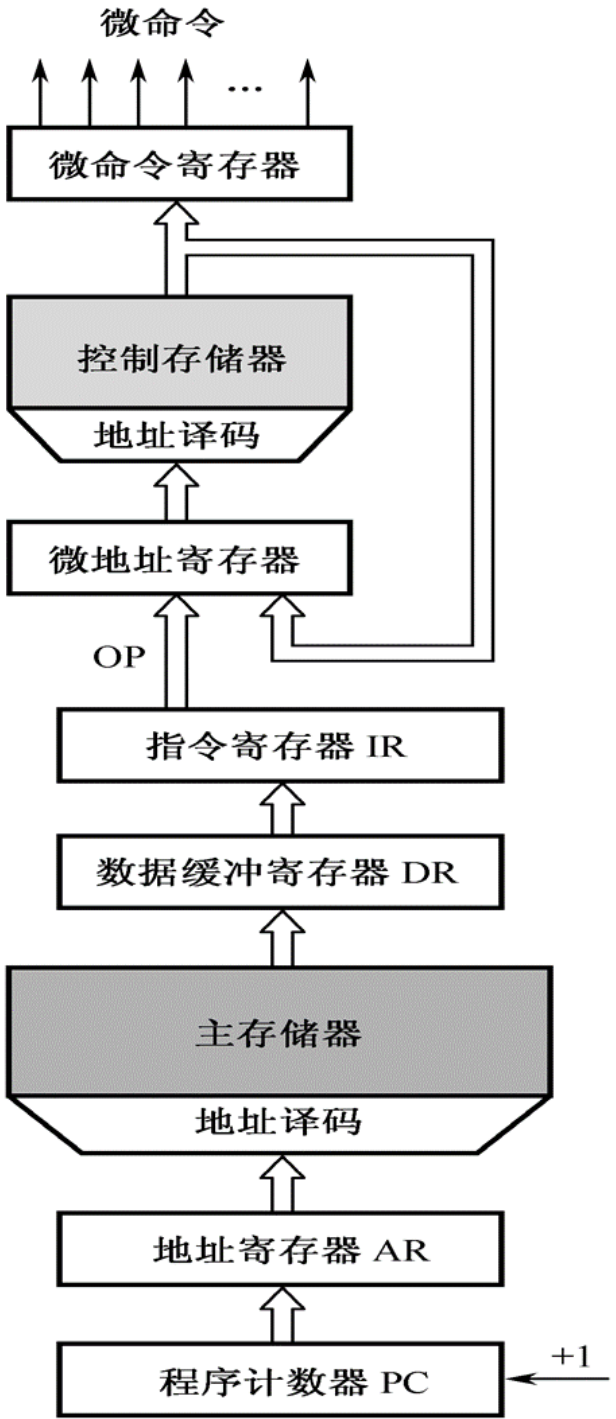
C. 每条微指令由若干条机器指令解释执行

D. 每条机器指令由若干条微程序解释执行

[答案]B

机器指令与微指令的关系

- **1.一条机器指令对应一个(段)微程序**
- **2.指令、程序、地址与内存有关，微指令、微程序、微地址与控制存储器有关**
- **3.每一个CPU周期对应一条微指令**



5.4.2 微程序控制器设计技术

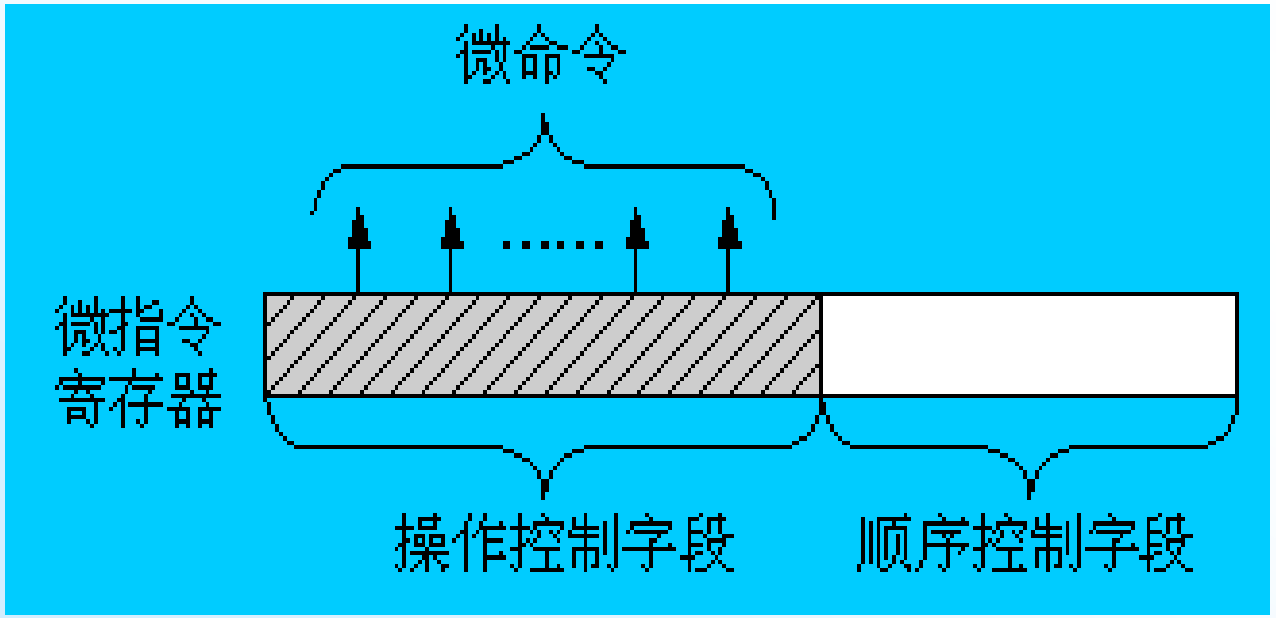
设计微指令结构应当追求的目标：

- 有利于缩短微指令字的长度
- 有利于减少控制存储器的容量
- 有利于提高微程序的执行速度
- 有利于对微指令的修改
- 有利于提高微程序设计的灵活性

5.4.2 微程序控制器设计技术

一、微命令的编码方法

- 编码有三种方法：直接表示法/编码表示法/混合表示法
- 1、直接表示法：操作控制字段中的各位分别可以直接控制计算机，不需要进行译码。

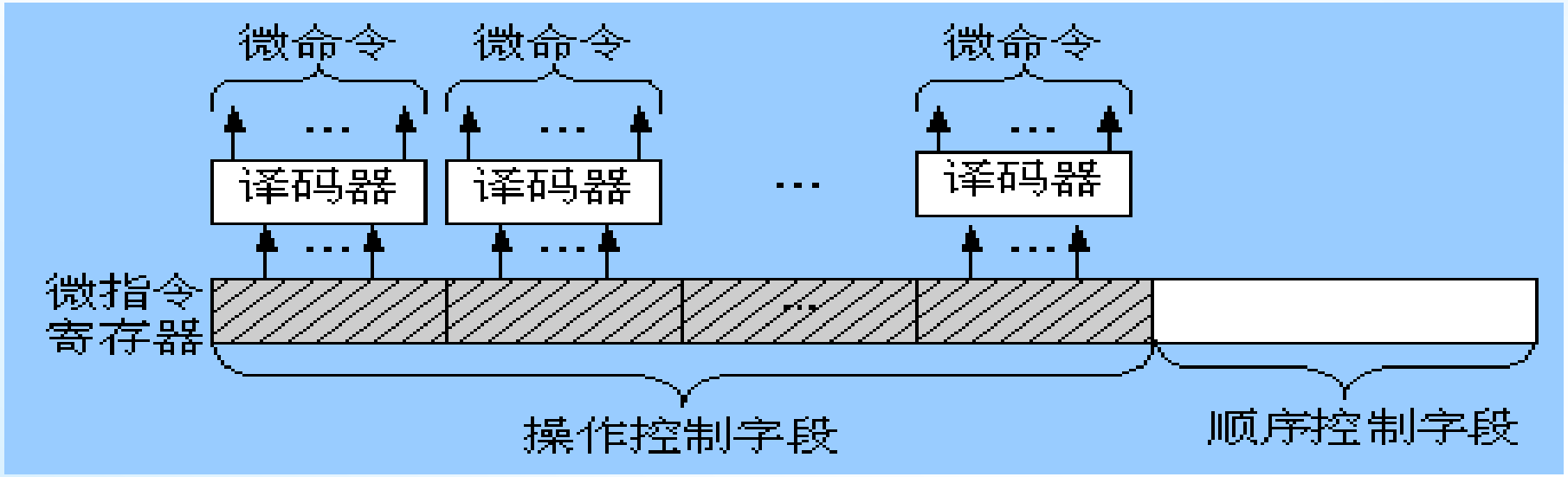


优点：速度最快

缺点：微指令字太长

5.4.2 微程序控制器设计技术

- **2、编码表示法：将操作控制字段分为若干个小组，每个组通过译码输出操作控制信号。**



每个字段中的命令是 互斥 的

缩短 了微指令 字长，增加 了译码 时间

5.4.2 微程序控制器设计技术

- 编码注意几点：

①把互斥性的微命令分在同一组内，兼容性的微命令分在不同组内。这样不仅有助于提高信息的利用率，缩短微指令字长，而且有助于充分利用硬件所具有的并行性，加快执行的速度。

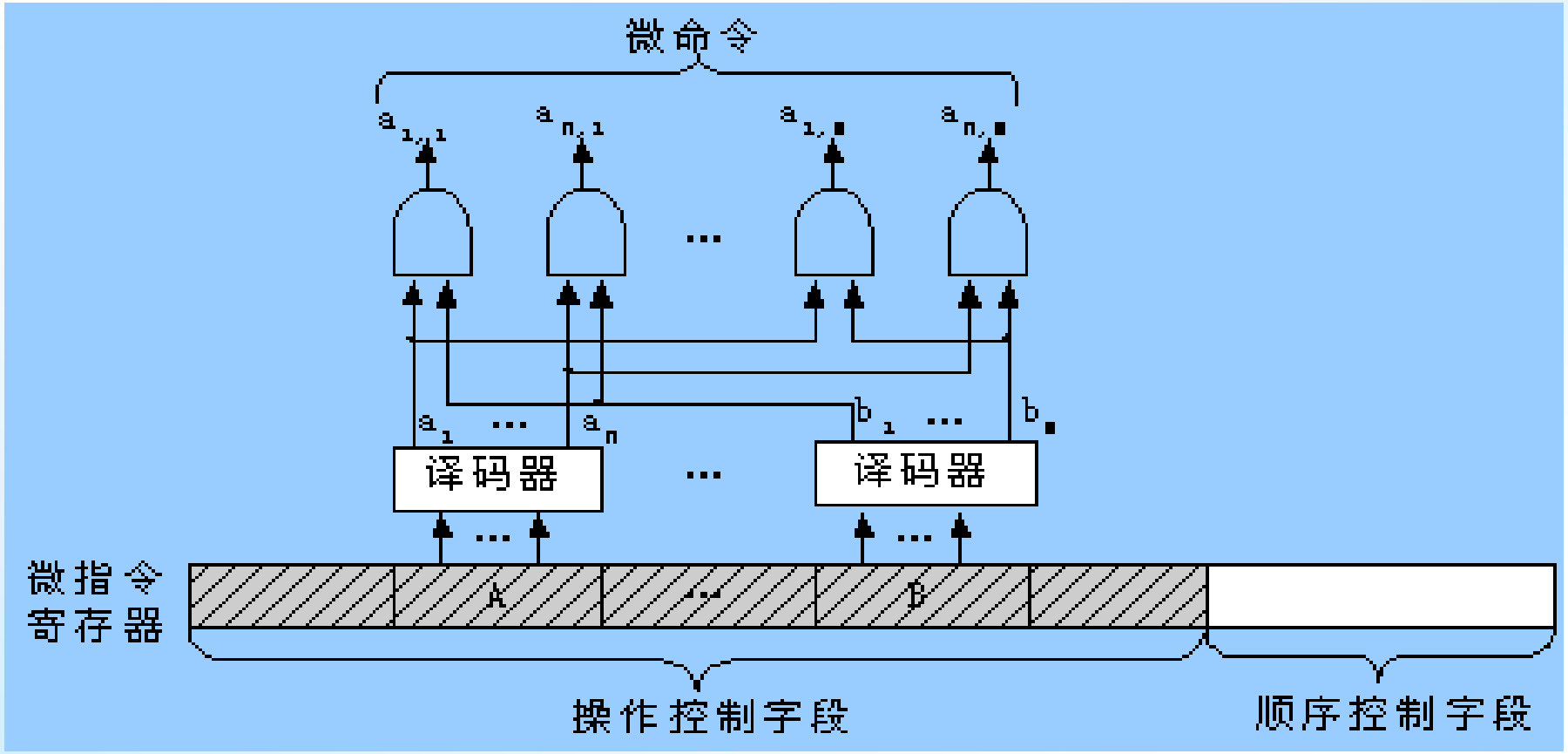
②应与数据通路结构相适应。

③每个小组中包含的信息位不能太多，否则将增加译码线路的复杂性和译码时间。

④一般每个小组还要留出一个状态，表示本小组不发出任何微命令。

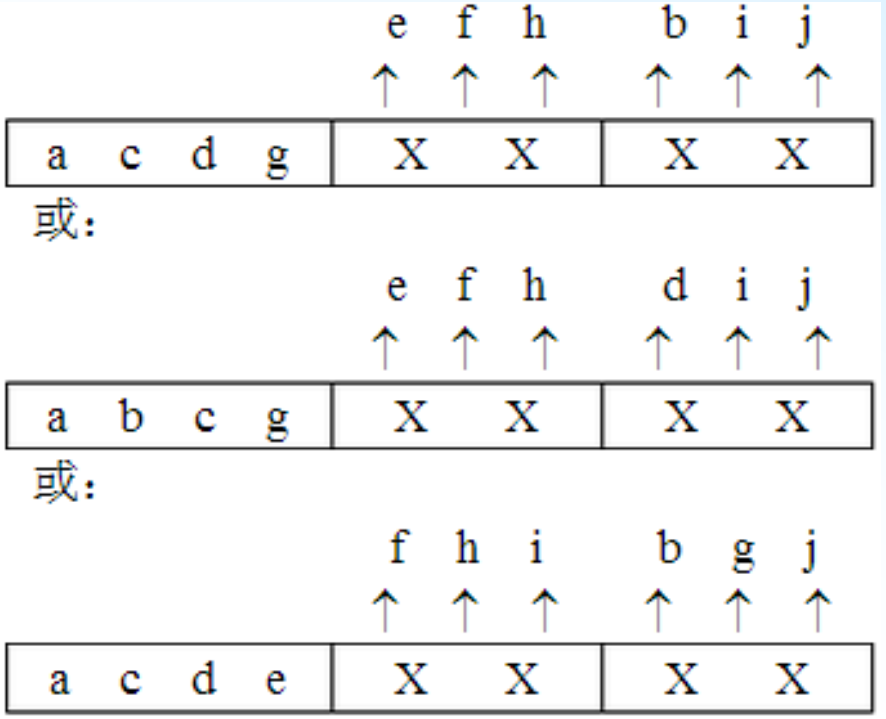
5.4.2 微程序控制器设计技术

- 3、混合编码法



【习题】某计算机有8条微指令I1—I8，每条微指令所包含的微命令控制信号见下表，a—j 分别对应10种不同性质的微命令信号。假设一条微指令的控制字段仅限8位，请安排微指令的控制字段格式。

微指令	a	b	c	d	e	f	g	h	i	j
I_1	√	√	√	√	√					
I_2	√			√		√	√			
I_3		√						√		
I_4			√							
I_5			√		√		√		√	
I_6	√							√		√
I_7			√	√				√		
I_8	√	√						√		



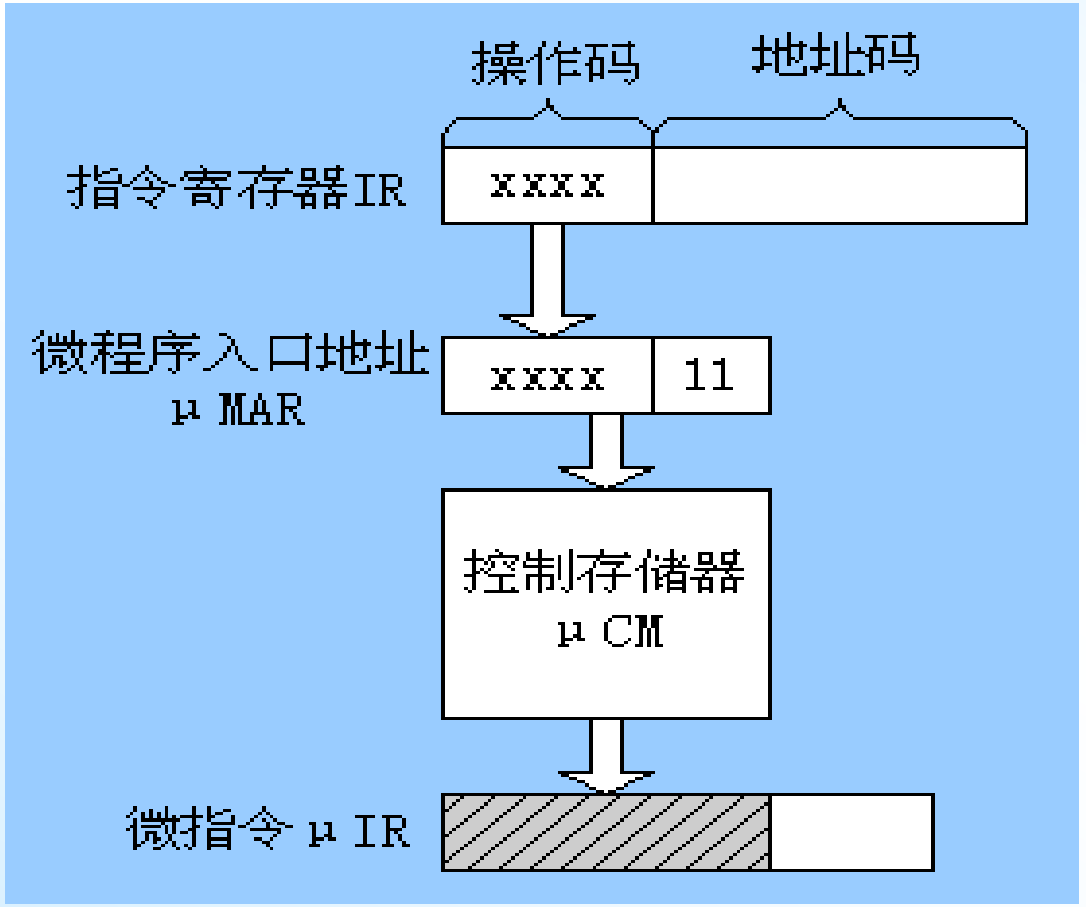
5.4.2 微程序控制器设计技术

二、微指令地址的形成

- 入口地址：每条机器指令对应一段微程序，当公用的取指微程序从主存中取出机器指令之后，由机器指令的操作码字段指出各段微程序的入口地址，这是一种多分支(或多路转移)的情况。
- 后继微地址：
  - 计数器的方式
  - 多路转移的方式

5.4.2 微程序控制器设计技术

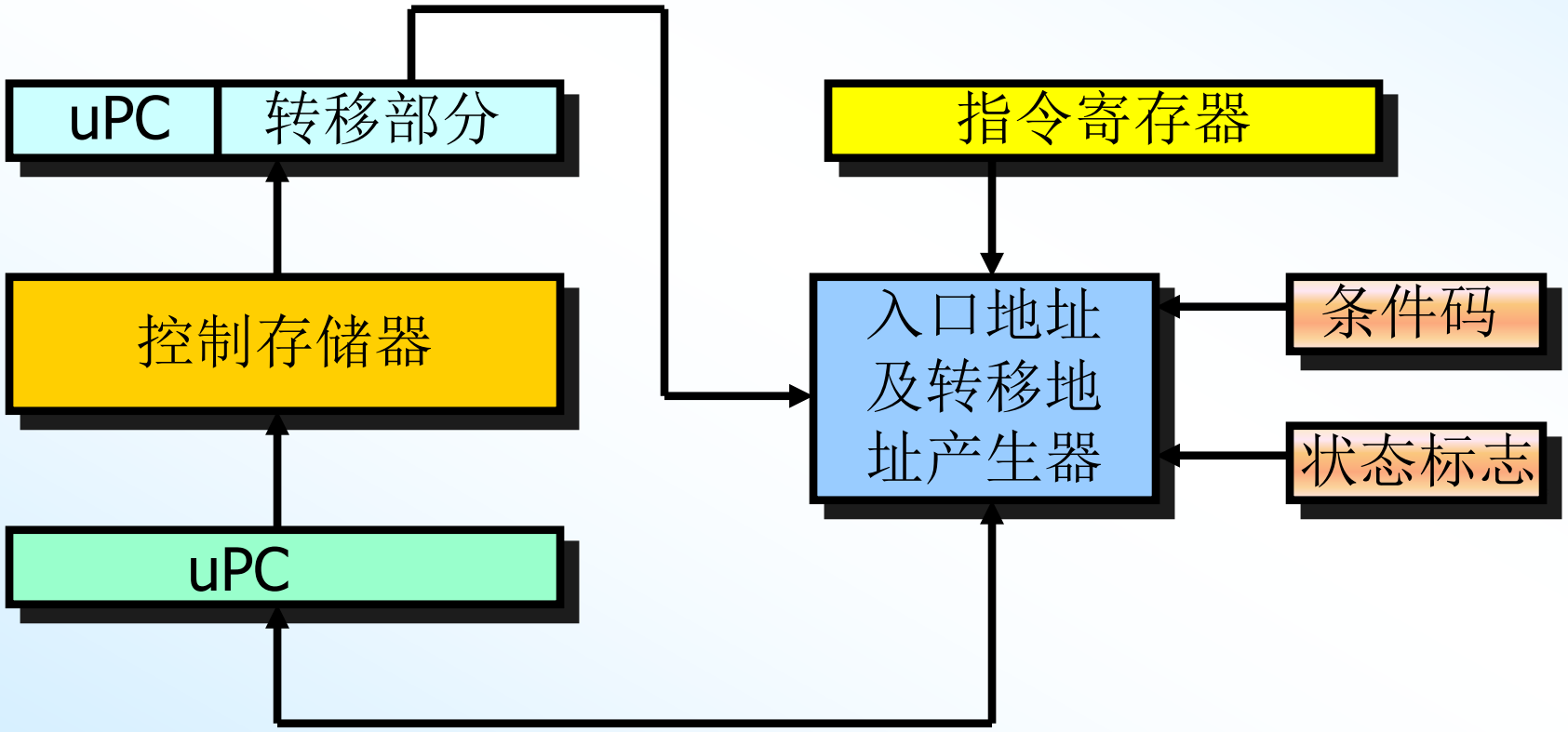
- 1、入口地址形成



5.4.2 微程序控制器设计技术

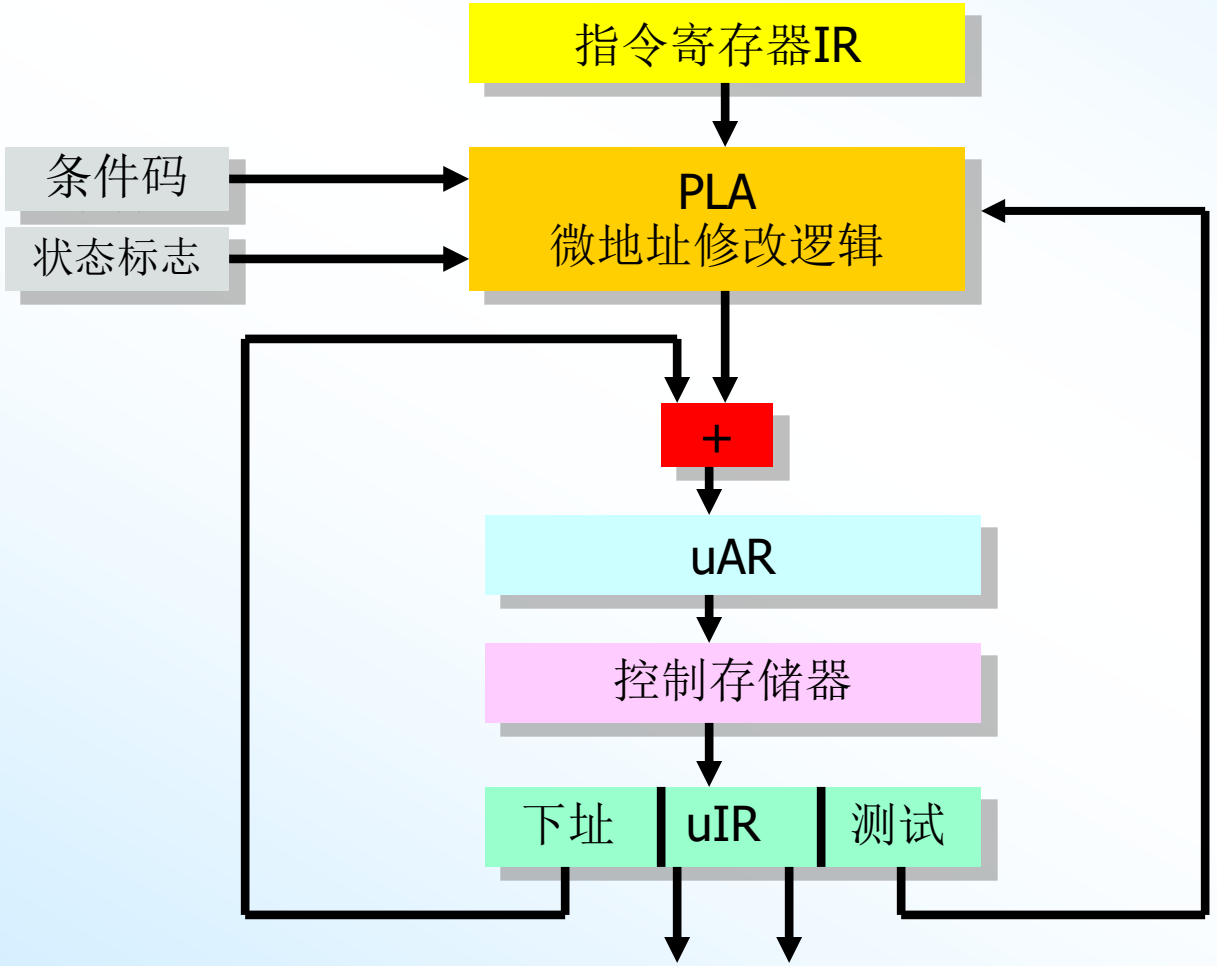
- **2、后继微地址形成方法**

（1）计数器方式



5.4.2 微程序控制器设计技术

（2）多路转移方式（判别测试、状态条件）



5.4.2 微程序控制器设计技术

- 2、后继微地址形成方法

（2）多路转移方式（判别测试、状态条件）

- 条件：状态条件/测试/微指令中微地址/操作码
- 例如：P字段的2位可以做4路转移
- 适用于：取指微指令执行之后，OP译码实现多路微程序入口微地址跳转

【教材P175例5.2】微地址寄存器有6位(μA_5-μA_0)，当需要修改其内容时，可通过某一位触发器的强置端S将其置“1”。现有三种情况：

(1)执行“取指”微指令后，微程序按IR的OP字段(IR_3-IR_0)进行16路分支；

(2)执行条件转移指令微程序时，按进位标志C的状态进行2路分支；

(3)执行控制台指令微程序时，按IR_4，IR_5的状态进行4路分支。

请按多路转移方法设计微地址转移逻辑。

- 问题分析：

指令格式：

进位标记C

微指令格式：

微命令	P3 P2 P1	uA5 uA4 uA3 uA2 uA1 uA0

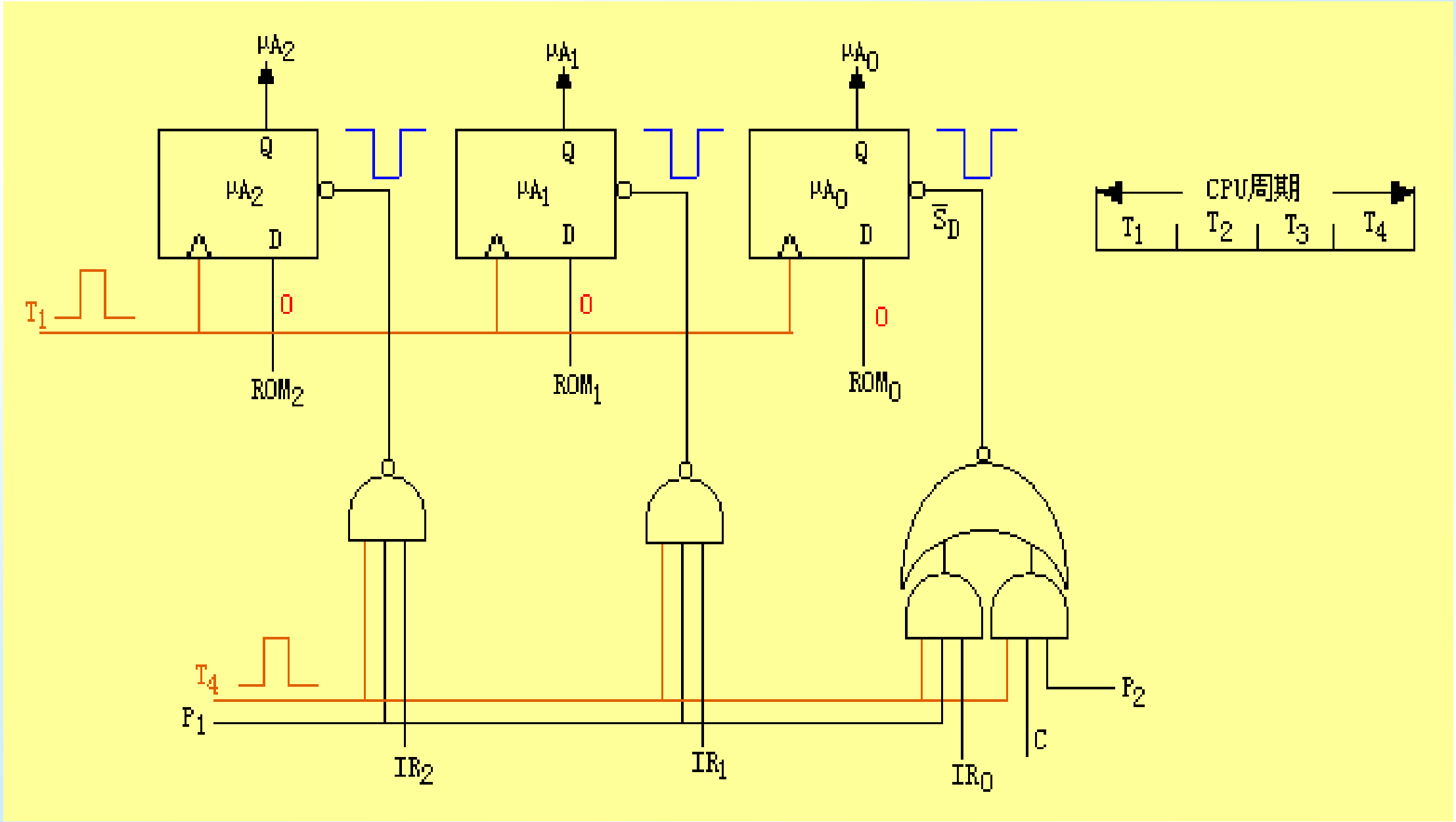
- 按所给设计条件，微程序有三种判别测试，分别为P_1，P_2，P_3。 由于修改μA_5-μA_0内容具有很大灵活性，现分配如下：
- (1)用P_1和IR_3-IR_0修改μA_3-μA_0；
- (2)用P_2和C修改μA_0；
- (3)用P_3和IR_5、IR_4修改μA_5、μA_4。
- 其中μA_0在P字段内容不同时，作用不同；
- 最后再加上节拍脉冲。

假设在CPU周期最后一个节拍脉冲获得微地址，转移逻辑表达式如下：

$$\mu A_5 = P_3 \cdot IR_5 \cdot T_4$$
$$\mu A_4 = P_3 \cdot IR_4 \cdot T_4$$
$$\mu A_3 = P_1 \cdot IR_3 \cdot T_4$$
$$\mu A_2 = P_1 \cdot IR_2 \cdot T_4$$
$$\mu A_1 = P_1 \cdot IR_1 \cdot T_4$$
$$\mu A_0 = P_1 \cdot IR_0 \cdot T_4 + P_2 \cdot C \cdot T_4$$

由于从触发器强置端修改，故前5个表达式可用“与非”门实现，最后一个用“与或非”门实现。

下图仅画出了μA_2、μA_1、μA_0触发器的微地址转移逻辑图。


μA2
μA1
μA0
Q
D
CPU周期
T1
T2
T3
T4
SD
0
ROM2
ROM1
ROM0
T4
P1
P2
C
IR2
IR1
IR0

【例题】某机采用微程序控制方式,微指令字长24位,采用水平型编码控制的微指令格式,断定方式。共有微命令30个，构成4个相斥类，各包含5个，8个，14个和3个微命令，外部条件3个。

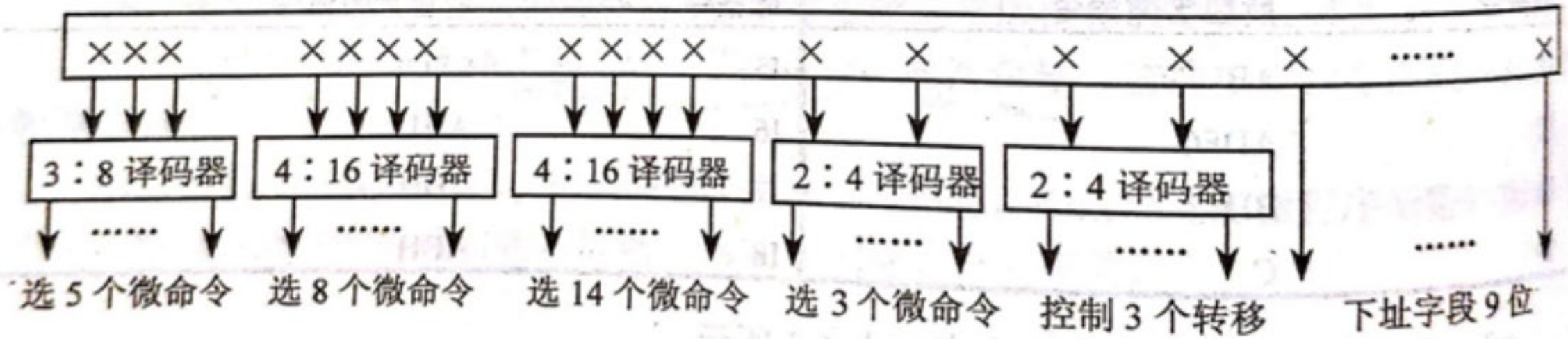
（1）控制存储器容量应为多少？

（2）设计出微指令的具体格式。

解：

（1）30 个微命令构成 4 个相斥类，其中 5 个相斥微命令需 3 位编码；8 个相斥微命令需 4 位编码，14 个相斥微命令需 4 位编码，3 个相斥微命令需 2 位编码；外部条件 3 个，采用断定方式需 2 位控制位。以上共需 15 位。微指令字长 24 位，采用水平型编码控制的微指令格式，所以还剩 9 位作为下址字段，这样控制存储器的容量应为 512×24 位。

（2）微指令的具体格式如图所示。



四、静态微程序设计和动态微程序设计

静态　微程序无需改变，采用 ROM

动态　通过 改变微指令 和 微程序 改变机器指令

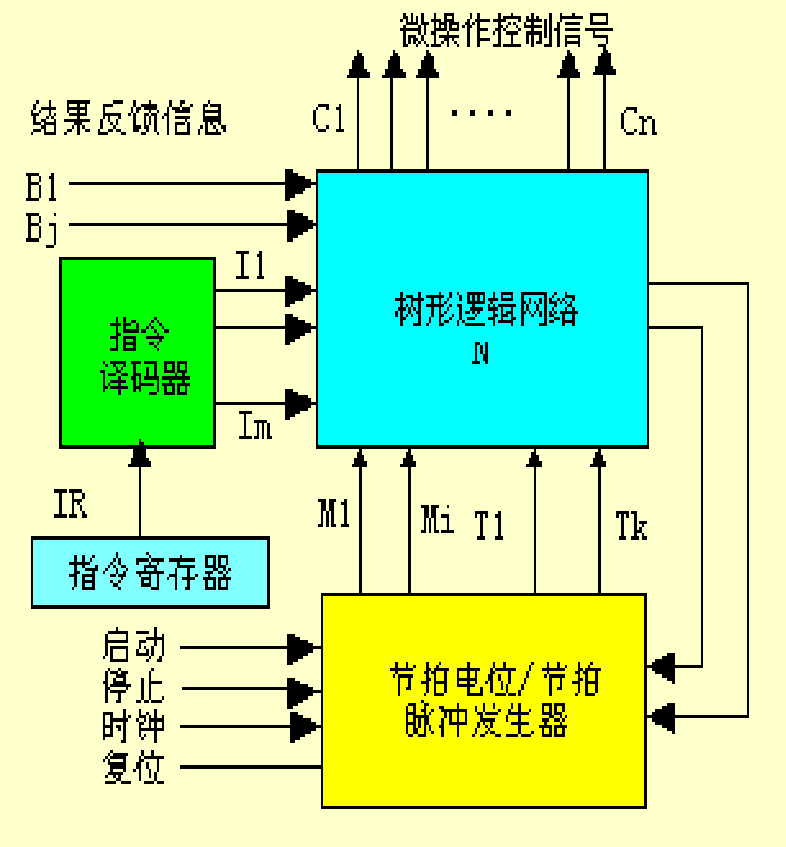
有利于仿真，采用 E^2PROM

5.5 硬布线控制器

- 实现方法
  - 通过逻辑电路直接连线而产生的，又称为组合逻辑控制方式，由门电路和触发器构成物理布线。
- 设计目标
  - 使用最少元件（复杂的树形网络）
  - 操作速度最高
  - 结构复杂不易调试，小的修改需要全盘重新设计布线
  - **VLSI**与速度需求
  - 硬布线控制器的没落和复兴...

5.5 硬布线控制器

1、基本思想



逻辑网络N的输入信号来源：

(1)指令译码器的输出I_m

(2)执行部件的反馈信息B_j（状态条件）

(3)来自时序产生器的时序信号M_i、T_k

逻辑网络N的输出C_n即微操作控制信号：

$$C_n = f(I_m, B_j, M_i, T_k) = \Sigma(M_i \cdot T_k \cdot B_j \cdot \Sigma I_m)$$

即：控制信号C_n在哪些CPU周期、T周期中，在什么样的状态条件下，对哪些指令应该有效。

5.5 硬布线控制器

- **2、硬连线控制 vs. 微程序控制**
  - **速度：**在微程序控制中，每条微指令都要从控存中读取。硬连线控制的速度更快。
  - **业界选择：**硬连线控制器一度被微程序控制器取代。随着VLSI技术的发展，硬连线控制器思想又逐步得到重视。某些超高速新型计算机结构又选用了硬连线控制器，或混合使用两种控制器。

5.5 硬布线控制器

- **3、硬连线控制器的设计方法**
  - 1) 确定指令系统、CPU内部部件构成、数据通路结构、时序信号和控制信号构成；
  - 2) 根据数据通路结构，画出指令系统中所有指令的执行流程，必须定位到每个CPU周期、每个*T*周期中要做的微操作和需要的相关控制信号；
  - 3) 根据指令执行流程，写出每条微命令的逻辑表达式；
  - 4) 根据选用的元器件，对逻辑表达式化简；
  - 5) 利用逻辑电路实现。

5.6 流水CPU

5.6.1 并行处理技术

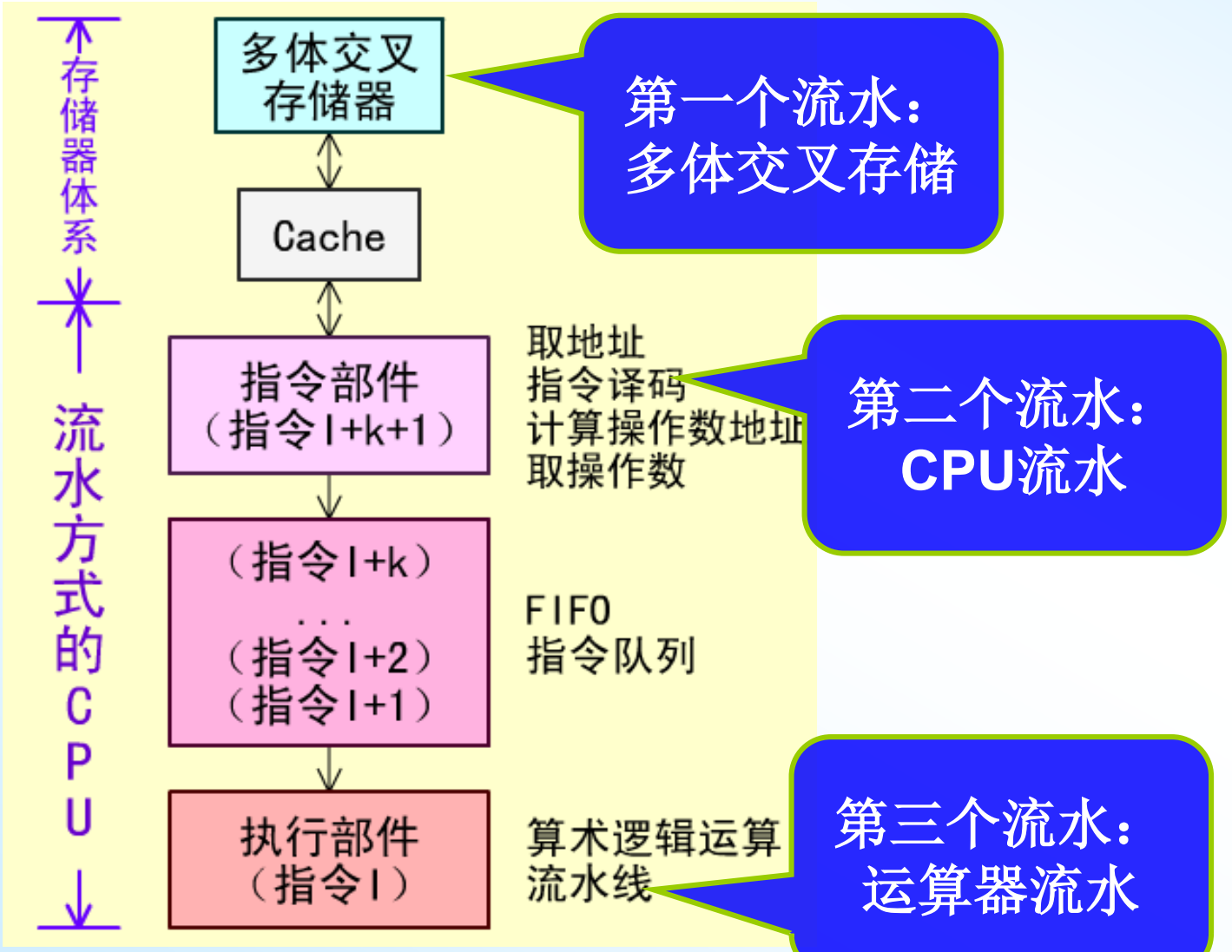
5.6.2 流水CPU的结构

5.6.3 流水线中的主要问题

5.6.1并行处理技术

- 并行的含义
  - 同时性：几个事件在同一时刻发生
  - 并发性：几个事件在同一时间间隔内发生
- 计算机并行处理的形式
  - 时间并行：轮流使用同一套硬件
  - 空间并行：设置多套硬件（CPU、I/O设备）
  - 时间并行+空间并行

5.6.2流水CPU的结构



5.6.2流水CPU的结构

- 具有什么特征的指令集有利于流水执行？
  - 长度尽量一致，有利于简化取指令和指令译码操作
  - 格式少，有利于在指令未知时就可取操作数
  - load/store指令才能访问存储器，有利于减少操作步骤，规整流水线
  - 内存中“对齐”存放，有利于减少访存次数和流水线的规整

5.6.2流水CPU的结构

- 流水线CPU时空图
  - IF（Instruction Fetch取指）
  - ID（Instruction Decode指令译码）
  - EX（Execution执行）
  - WB（Write Back写回）

5.6.2流水CPU的结构

- 非流水线时空图

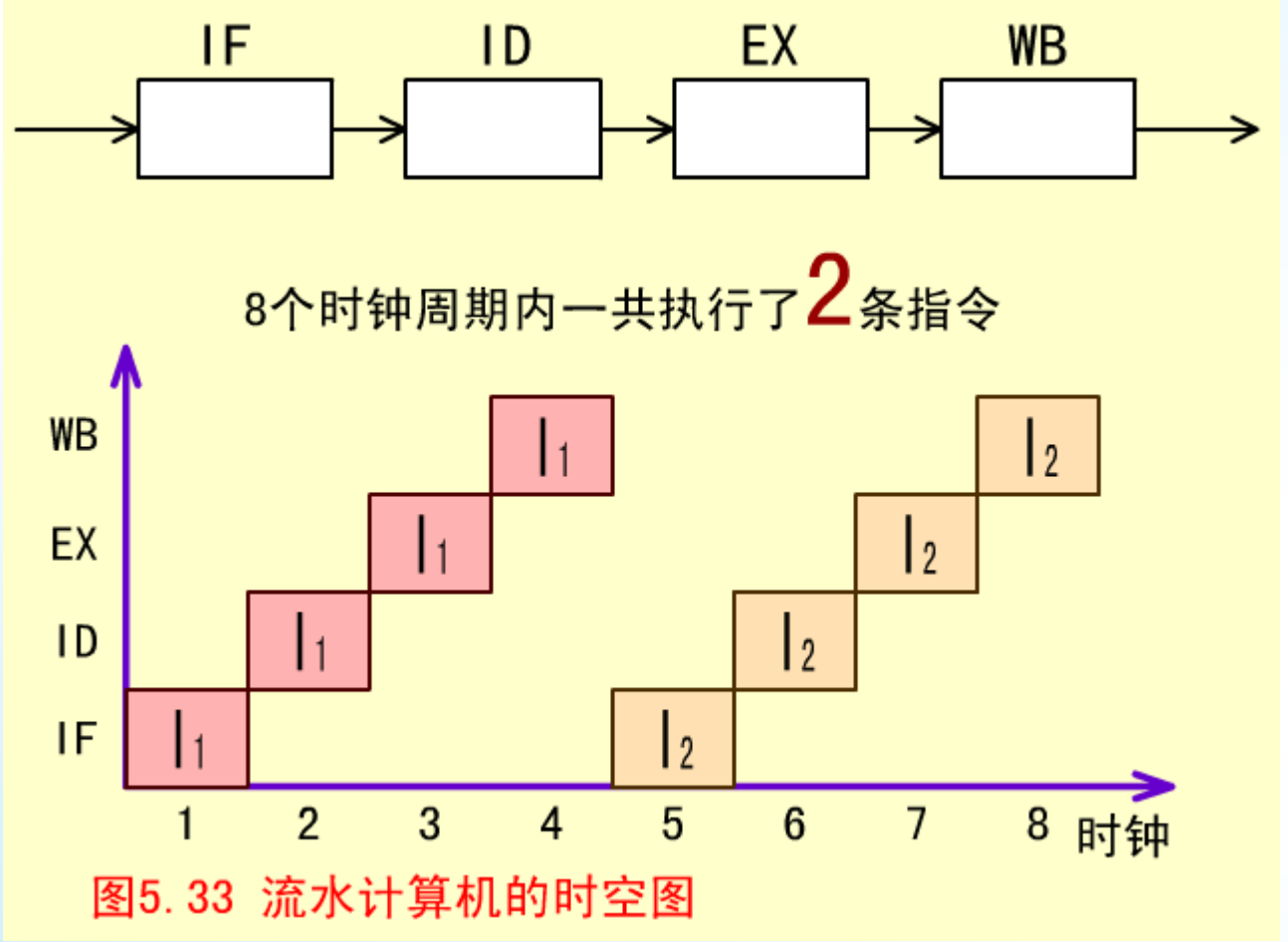


图5.33 流水计算机的时空图

5.6.2流水CPU的结构

- 标量流水线时空图

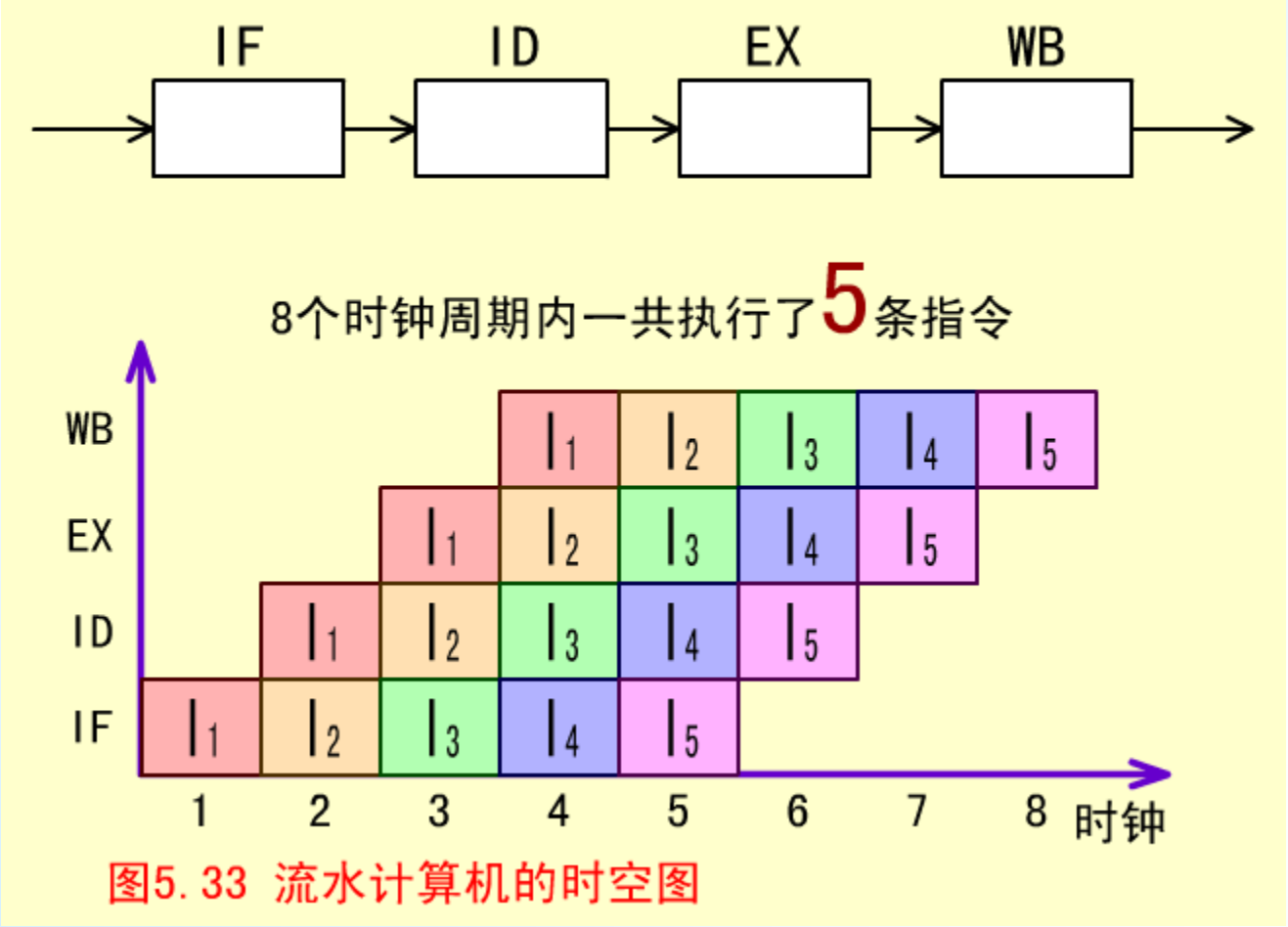


图5.33 流水计算机的时空图

5.6.2流水CPU的结构

- 超标量流水线时空图

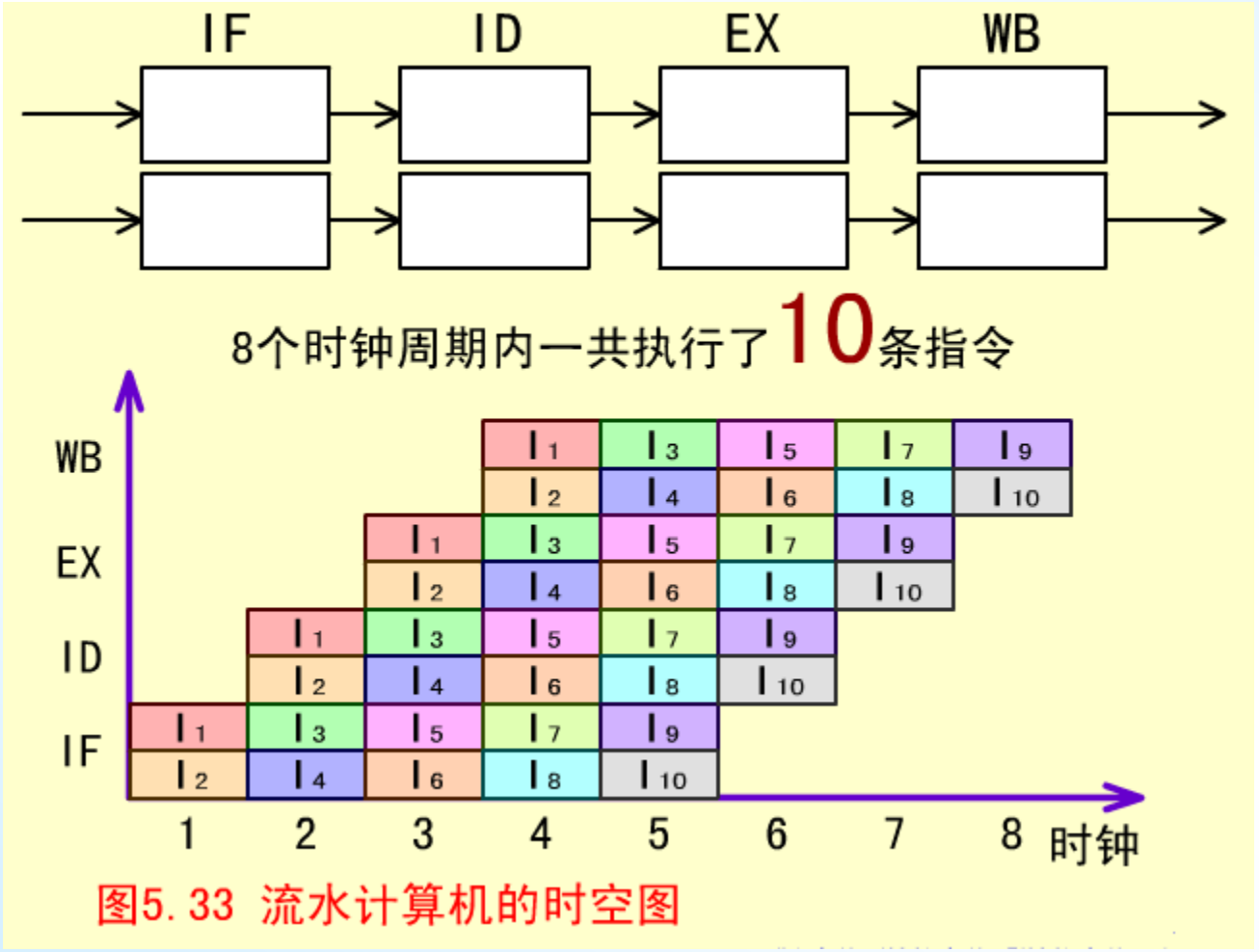


图5.33 流水计算机的时空图

79 某CPU主频为1.03GHz，采用4级指令流水线，每个流水段的执行需要1个时钟周期。假定CPU执行了100条指令，在其执行过程中没有发生任何流水线阻塞，此时流水线的吞吐率为（　　）。[2013年408统考]

A．0.25×10^9条指令/秒

B．0.97×10^9条指令/秒

C．1.0×10^9条指令/秒

D．1.03×10^9条指令/秒

[答案]C

5.6.3 流水线中的主要问题

- 要使流水线具有良好的性能，必须使流水线畅通流动，不发生断流。

- 但由于流水过程中会出现以下三种相关冲突，实现流水线的不断流是困难的。

1. 资源相关 (结构冒险)
2. 数据相关 (数据冒险)
3. 控制相关 (控制冒险)

5.6.3流水线中的主要问题

- **1.资源相关（结构冒险）：** 多条指令进入流水线后在同一时钟周期内争用同一功能部件。

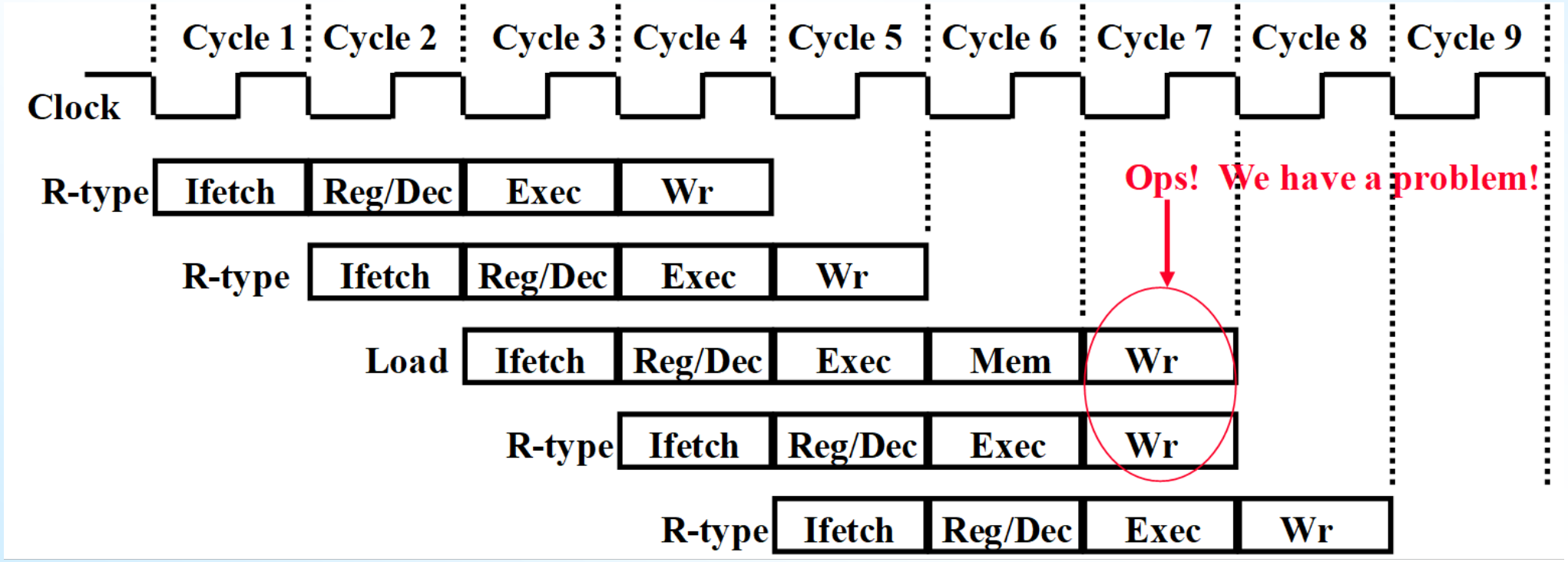
解决资源相关冲突的办法：

方法1：增设一个存储器，将指令和数据分别放在两个存储器中。

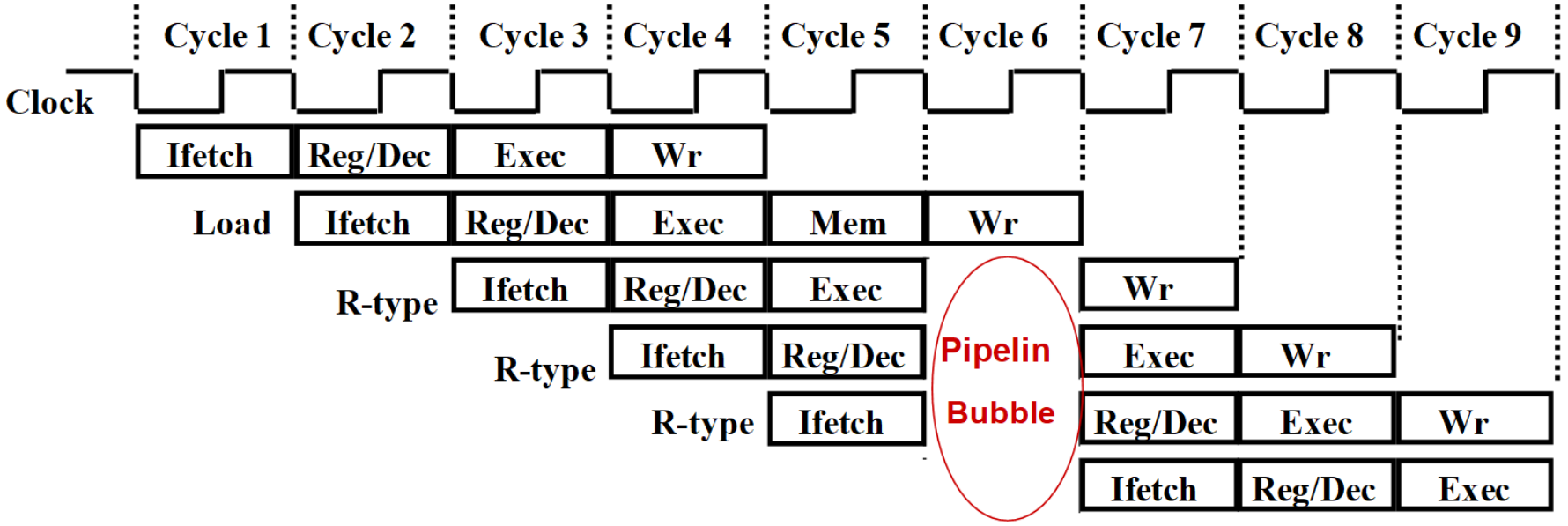
方法2：指令I_4停顿一个时钟周期后再启动（插入气泡）。

5.6.3流水线中的主要问题

- **1.资源相关（结构冒险）：** 多条指令进入流水线后在同一时钟周期内争用同一功能部件。

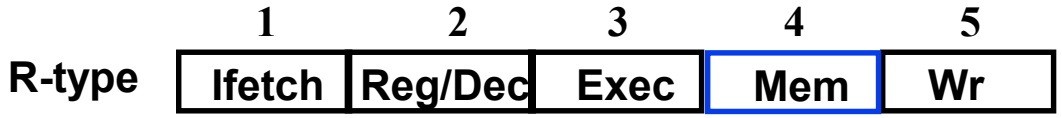


- 方法2:插入气泡

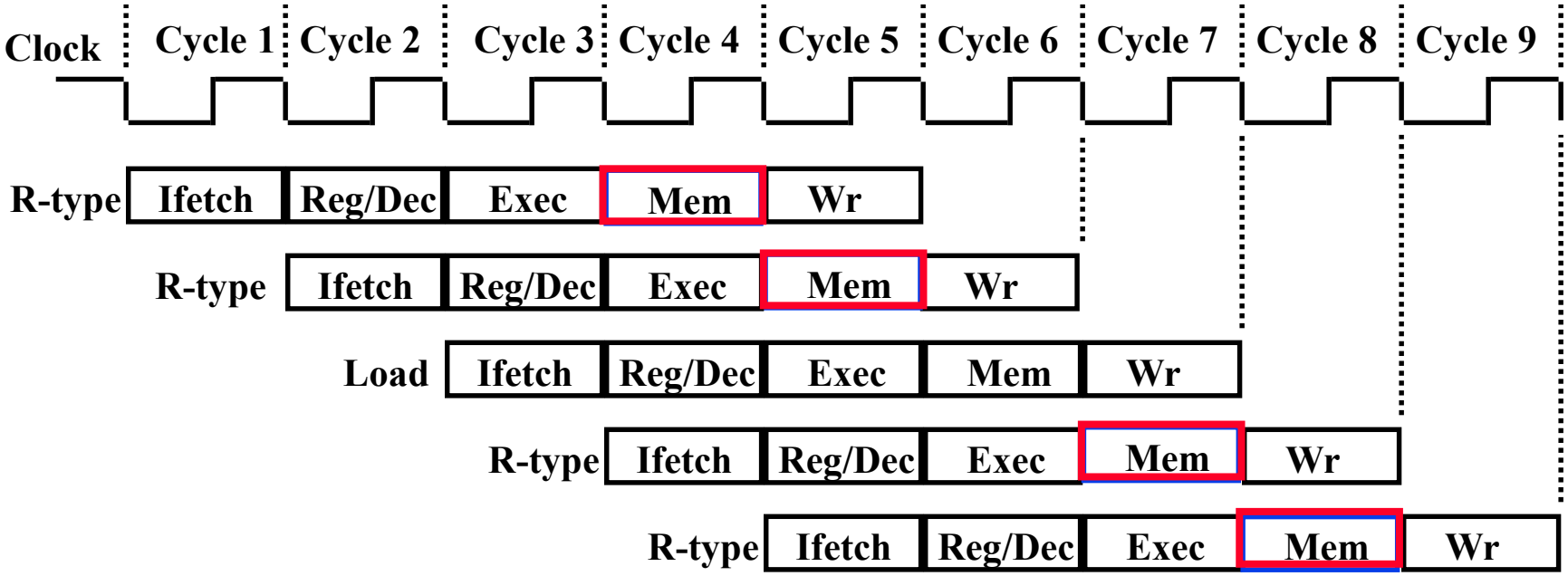


问题：1.控制逻辑复杂；2.第五周期没有指令被完成

方法3:对于长度不一致的指令，补一个NOP使流水线更规整



- 加一个NOP阶段以延迟“写”操作:
  - 把“写”操作安排在第5阶段，这样使R-Type的Mem阶段为空NOP



这样使流水线中的每条指令都有相同多个阶段!

5.6.3流水线中的主要问题

- **2.数据相关（数据冒险）：**
- 例：两条指令发生数据相关冲突

ADD R1, R2, R3 R2+R3-->R1

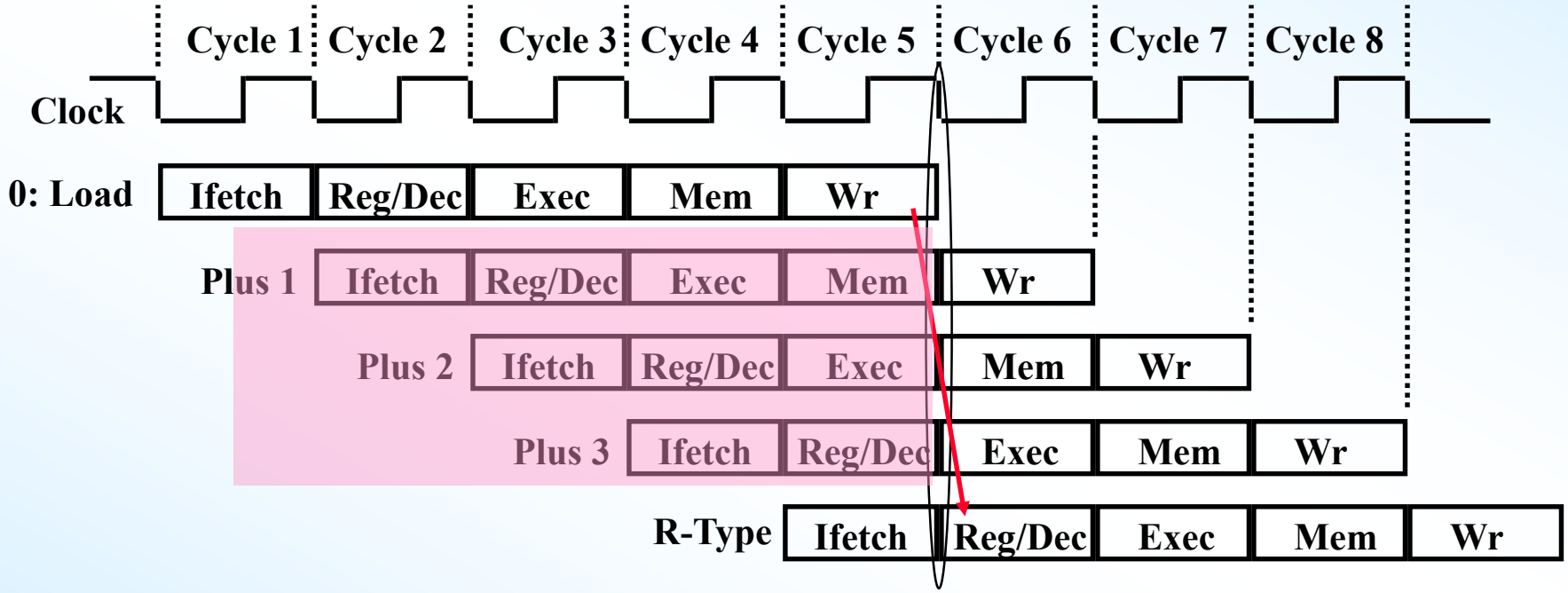
SUB R4, R1, R5 R1-R5-->R4

AND R6, R1, R7 R1^R7-->R6

指令＼时钟	1	2	3	4	5	6	7	8
ADD	IF	ID	EX	MEM	WB			
SUB		IF	ID	EX	MEM	WB		
AND			IF	ID	EX	MEM	WB	

5.6.3流水线中的主要问题

- **2.数据相关（数据冒险）：**



- 尽管Load指令在第一周期就被取出，但:
  - 数据在第五周期结束才被写入寄存器
  - 到第六周期写入的数据才能被用

  结果：如果随后指令要用到Load的数据的话，就需延迟三条指令!

5.6.3流水线中的主要问题

- 数据相关
  - **RAW(Read After Write)**写后读
    - 后面指令用到前面指令所写的数据
  - **WAR(Write After Read)**读后写
    - 后面指令覆盖前面指令所读的单元
  - **WAW(Write After Write)**写后写
    - 两条指令写同一个单元

【教材P179例5.4】流水线中有三类数据相关冲突：写后读（RAW）相关；读后写（WAR）相关；写后写（WAW）相关。判断以下三组指令各存在哪种类型的数据相关。

1）I1　ADD R1，R2，R3；（R2）+（R3）->R1
　　I2　SUB R4，R1，R5；（R1）-（R5）->R4

2）I3　STO M　（R3）->M(x)
　　M(x)是存储器单元
　　I4　ADD R3，R4，R5；（R4）+

3）I5　MUL R　（R1）×（R2）->R3
　　I6　ADD R3，R1，R5；（R4）+（R5）->R3

写后读

写后写

读后写

数据冒险的解决方法

- 方法1：硬件阻塞（stall）
- 方法2：软件插入“NOP”指令
- 方法3：合理实现寄存器堆的读/写操作（不能解决所有数据冒险）
  - 前半时钟周期写，后半时钟周期读，若同一个时钟内前面指令写入的数据正好是后面指令所读数据，则不会发生数据冒险
- 方法4：转发（Forwarding或Bypassing 旁路）技术
  - 若相关数据是ALU结果，则如何?

    可通过转发解决
  - 若相关数据是上条指令DM读出内容，则如何?

    不能通过转发解决，随后指令需被阻塞一个时钟 或 加NOP指令

    称为Load-use数据冒险!
- 方法5：编译优化：调整指令顺序（不能解决所有数据冒险）

实现“转发”和“阻塞” 要修改数据通路:

（1）检测何时需要“转发”，并控制实现“转发”

（2）检测何时需要“阻塞”，并控制实现“阻塞”

5.6.3流水线中的主要问题

- **3.控制相关（控制冒险）：**
  - 引起原因：转移指令
    - 当前指令有跳转，但流水已经开启后续指令处理过程。

控制冒险的解决方法

- 方法1：硬件阻塞（stall）
- 方法2：软件插入 “NOP”指令

  （以上两种方法的效率太低，需结合分支预测进行）
- 方法3：分支预测（Predict）
  - 简单（静态）预测：
    - 总是预测条件不满足(not taken)，即：继续执行分支指令的后续指令

      可加启发式规则：在特定情况下总是预测满足(taken)，其他情况总是预测不满足。如：循环顶部（底部）分支总是预测为不满足（满足）。能达65%-85%的预测准确率
  - 动态预测：
    - 根据程序执行的历史情况进行动态预测调整，能达90%的预测准确率

  注：流水线控制必须确保被错误预测指令的执行结果不能生效，而且要能从正确的分支地址处重新启动流水线工作
- 方法4：延迟分支（Delayed branch）（通过编译程序优化指令顺序！）
  - 把分支指令前面与分支指令无关的指令调到分支指令后执行，也称延迟转移

另一种控制冒险：异常或中断控制冒险的处理

总结：流水线的三种冲突/冒险（Hazard）情况

- **Hazards**：指流水线遇到无法正确执行后续指令或执行了不该执行的指令
  - **Structural hazards (hardware resource conflicts):**

    现象：同一个部件同时被不同指令所使用
    - 一个部件每条指令只能使用1次，且只能在特定周期使用
    - 设置多个部件，以避免冲突。如指令存储器IM 和数据存储器DM分开
  - **Data hazards (data dependencies):**

    现象：后面指令用到前面指令结果数据时，前面指令的结果还没产生
    - 采用转发(Forwarding/Bypassing)技术
    - Load-use冒险需要一次阻塞(stall)
    - 编译程序优化指令顺序
  - **Control (Branch) hazards (changes in program flow):**

    现象：转移或异常改变执行流程，后继指令在目标地址产生前已被取出
    - 采用静态或动态分支预测
    - 编译程序优化指令顺序(分支延迟)

5.6.4 奔腾 CPU

1．Pentium的技术性能

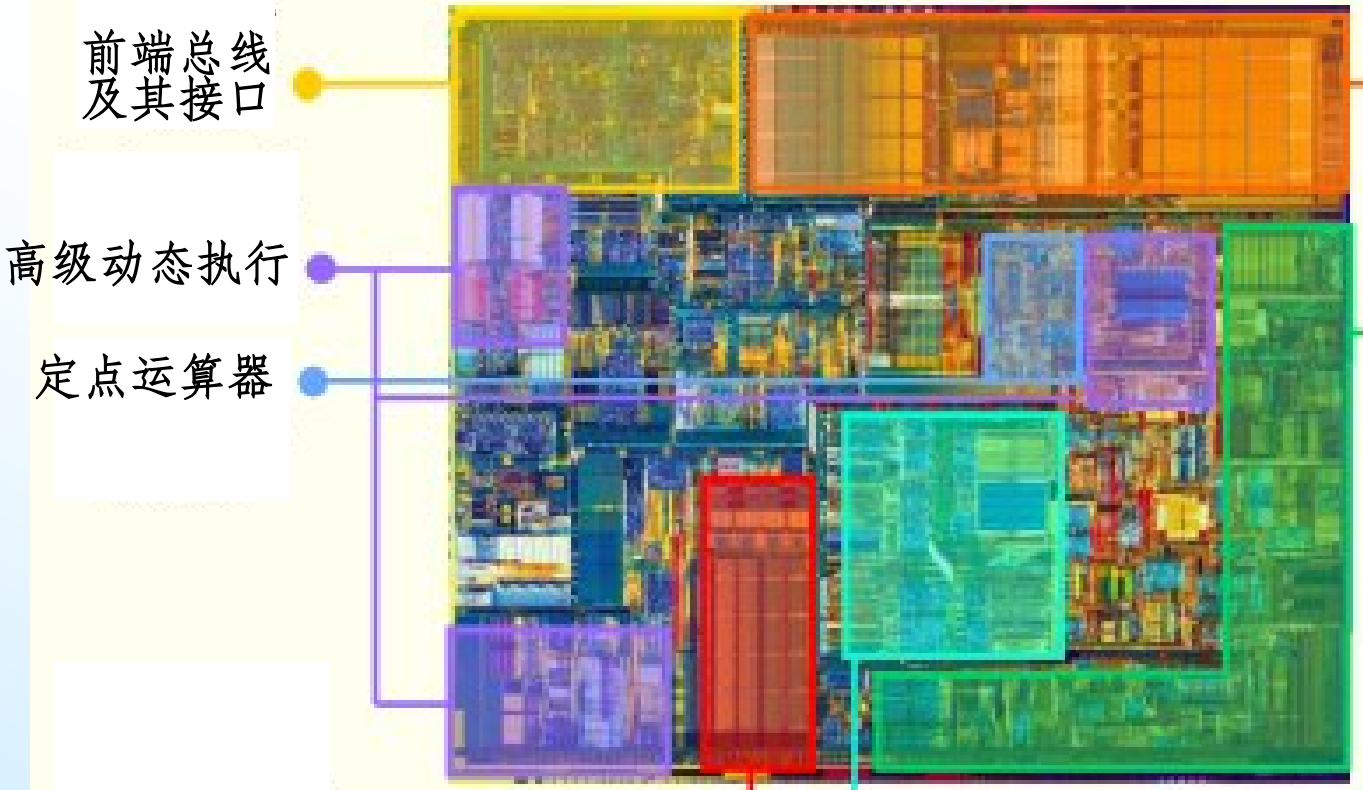
×2．奔腾CPU的结构框图

1．Pentium的技术性能

- Pentium是Intel公司生产的超标量流水处理器
- CPU内部的主要寄存器宽度为32位，故认为它是一个32位微处理器，但它通向存储器的外部数据总线宽度为64位
- CPU外部地址总线宽度是36位，但一般使用32位
- CPU内部分别设置指令Cache和数据Cache，外部还可接L2 Cache

- CPU采用U、V两条指令流水线，能在一个时钟周期内发射两条简单的整数指令，也可发射一条浮点指令
- 操作控制器采用硬布线控制和微程序控制相结合的方式。大多数简单指令用硬布线控制实现，在一个时钟周期内执行完毕。对微程序实现的指令，也在2-3个时钟周期内执行完毕
- Pentium具有非固定长度的指令格式，具有CISC和RISC两者的特性，被看成为一个CISC结构的处理器

Pentium 4 处理器的芯片布局



本章重点内容

1、基本概念

微命令、微操作、微指令、微程序

指令流水线、算术流水线、处理机流水线

2、CPU的功能

3、熟悉典型模型机结构、指令和数据的传输通路

4、指令周期流程图

5、指令周期、机器周期、时钟周期三级时序

6、微程序控制器原理及组成框图

7、流水线中的资源相关、数据相关、控制相关问题

作业：P201-202 6，8，11，12，13