北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

5.3 时序产生器和控制方式

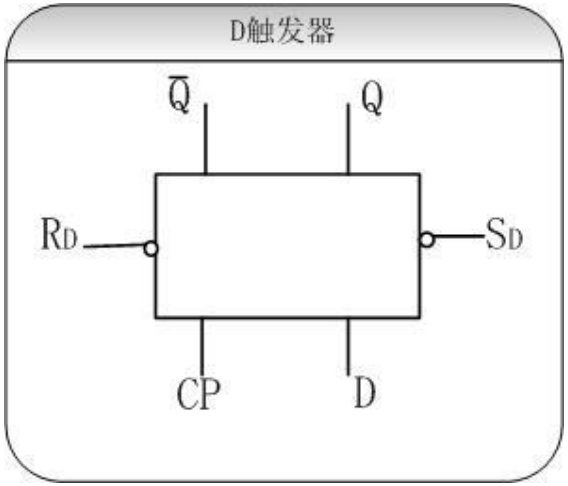
提纲

5.3.1 时序产生器作用和体制

- 计算机的协调动作需要时间标志，而时间标志则是用时序信号来体现
- 作用
  - CPU中的控制器用它指挥机器的工作
  - CPU可以用时序信号/周期信息来辨认从内存中取出的是指令（取指周期）还是数据（执行周期）
  - 一个CPU周期中时钟脉冲对CPU的动作有严格的约束
- 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

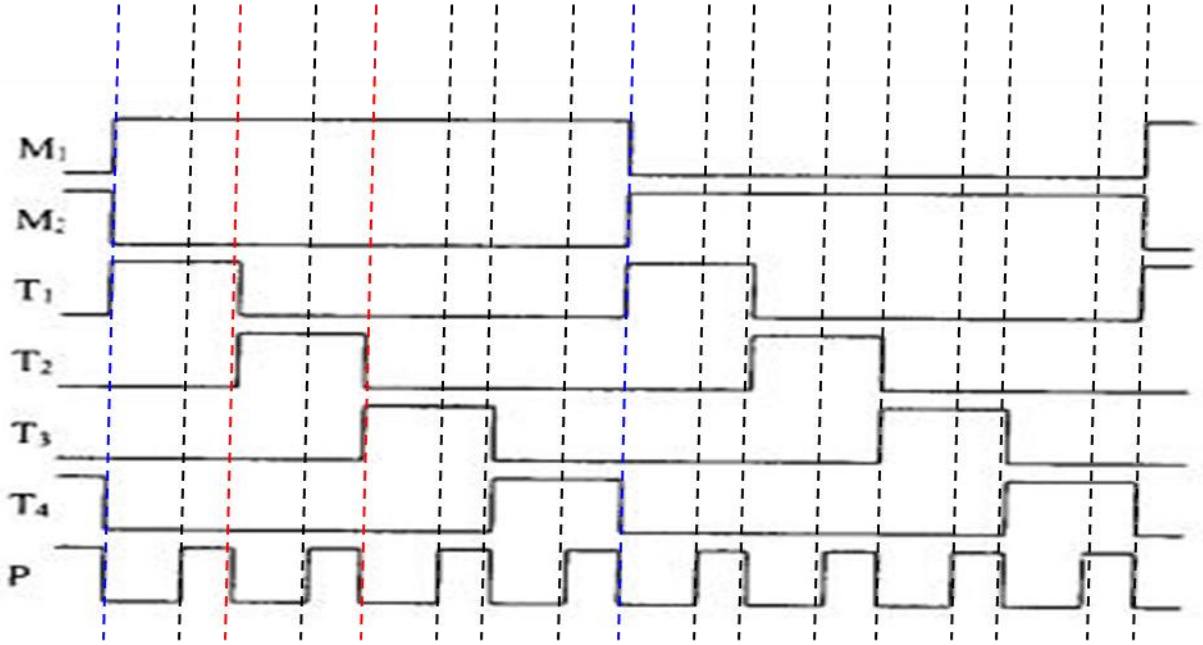
- 体制
  - 组成计算机硬件的器件特性决定了时序信号的基本体制是电位—脉冲制
  - 以触发器为例：D为电位输入端，CP（Clock Pulse）为脉冲输入端
  - R（复位），S（选择）为电位输入端



  - 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

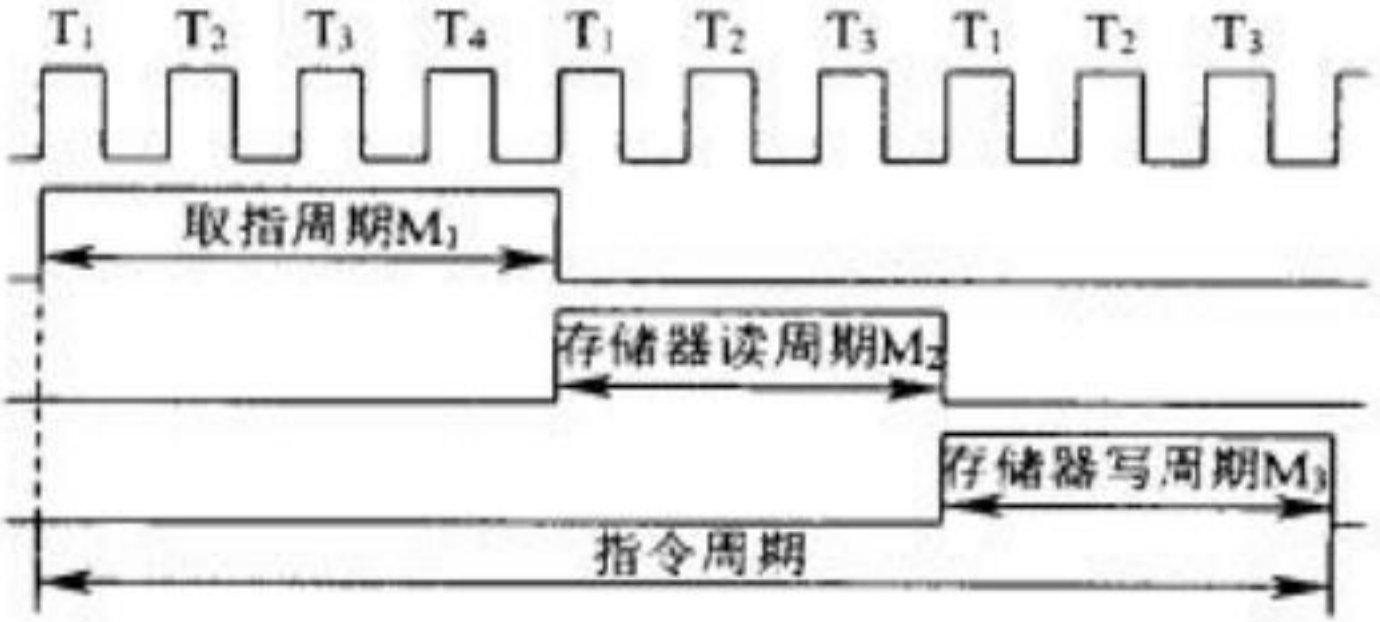
- 硬布线控制器中，时序信号往往采用主状态周期—节拍电位—节拍脉冲三级体制
  - 一个节拍电位表示一个CPU周期（较大的时间单位）
  - 一个节拍电位又包含若干个节拍脉冲（较小的时间单位）
  - 主状态周期可以包含若干个节拍电位（最大的时间单位）
  - 时序信号产生电路复杂



- M_1、M_2：主状态周期
- T_1-T_4：节拍电位
- P：节拍脉冲

5.3.1 时序产生器作用和体制

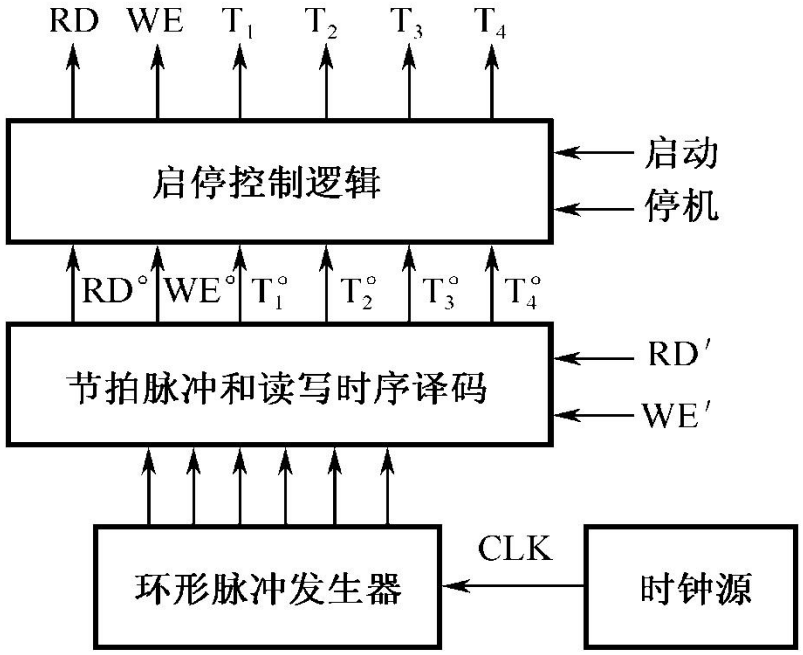
- 微程序控制器，节拍电位—节拍脉冲二级体制
  - 一个节拍电位（CPU周期）包含若干个节拍脉冲（T周期）
  - 节拍脉冲的时间间隔可以相等也可以不相等
  - 利用微程序顺序执行来实现微操作
  - 时序信号产生电路简单



5.3.2 时序信号产生器

- 功能：用逻辑电路产生时序信号
  - 各型计算机的时序信号产生电路不相同
  - 大、中型计算机的时序电路复杂，微型计算机的时序电路简单
  - 硬连线控制器的时序电路复杂
  - 微程序控制器的时序电路简单

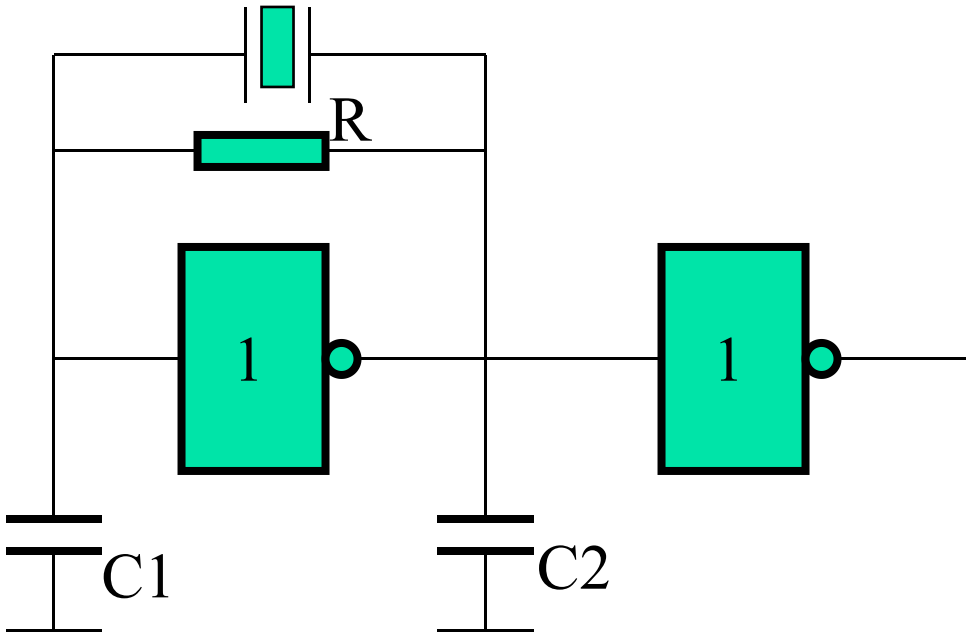
- 微程序控制器的时序信号产生器的构成：
  - 时钟源
  - 环形脉冲发生器
  - 节拍脉冲和读写时序译码逻辑
  - 启停控制逻辑



5.3.2 时序信号产生器

1、时钟脉冲源

- 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

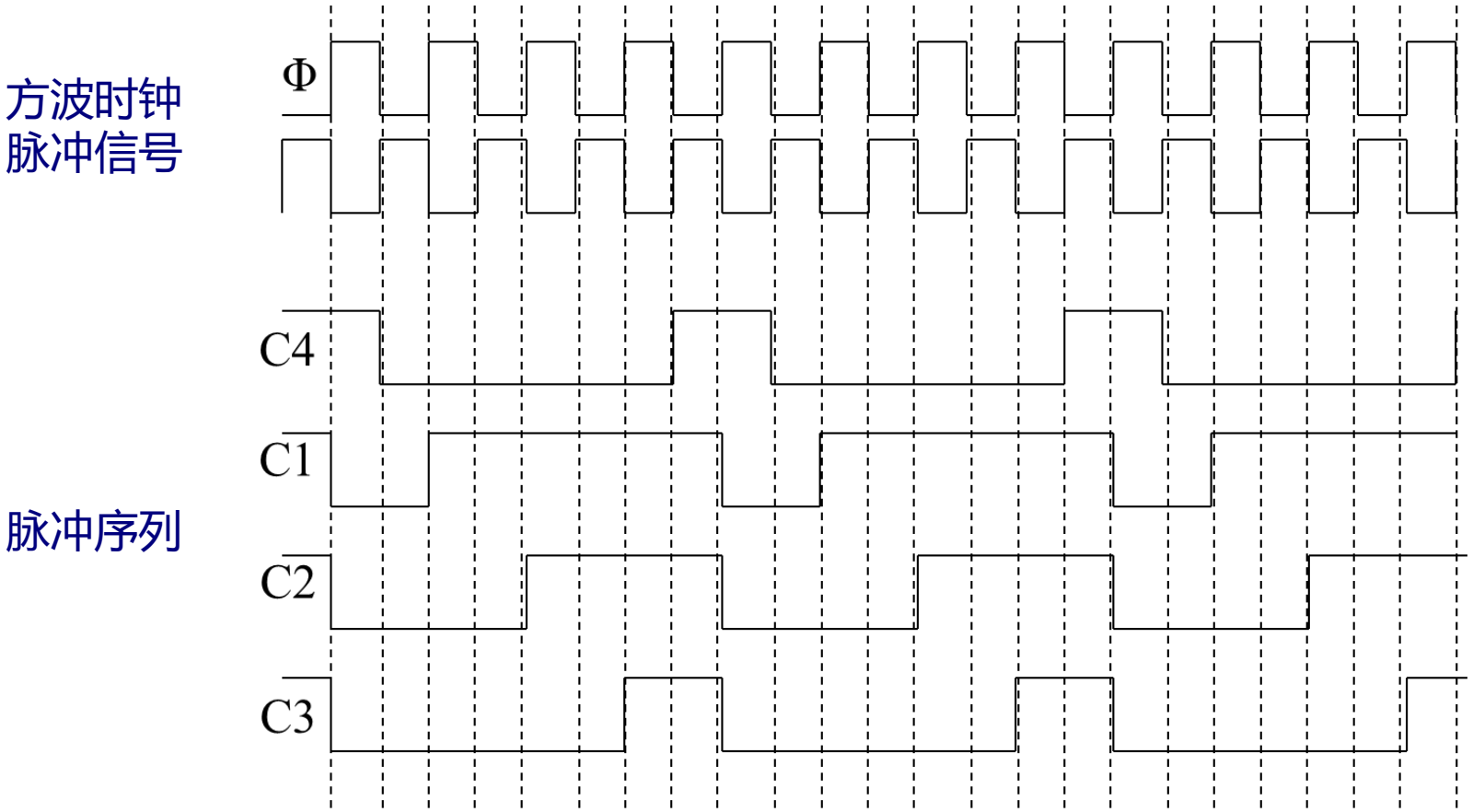


- 电路左边是振荡电路，右边是整形电路，左边的电路产生接近正弦波的波形，右边非门则将其整形为一个理想的方波

5.3.2 时序信号产生器

2、环形脉冲发生器

- 作用：产生一组有序的间隔相等或不等的脉冲序列，以便通过译码电路来产生最后所需的节拍脉冲



5.3.2 时序信号产生器

3、节拍脉冲和读/写时序的译码

- 节拍脉冲的译码逻辑
- 假设一个CPU周期包含4个等间隔的节拍脉冲

$$T_1^\circ = C_1 \cdot \overline{C_2}$$

$$T_2^\circ = C_2 \cdot \overline{C_3}$$

$$T_3^\circ = C_3$$

$$T_4^\circ = \overline{C_1}$$

5.3.2 时序信号产生器

3、节拍脉冲和读/写时序的译码

- 读写时序信号（用来进行存储器的读/写操作）的译码逻辑表达式

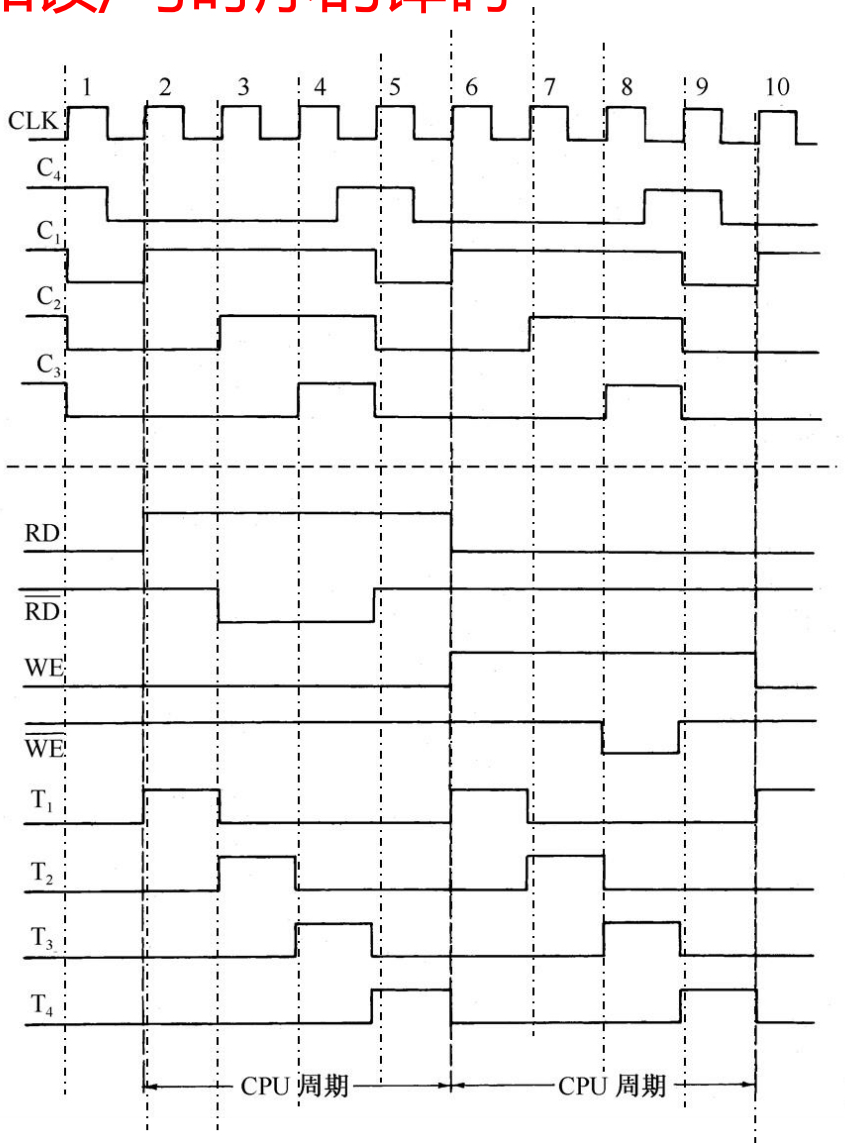
$$\overline{RD^{\circ}} = C_2 \cdot \mathrm{R}D'$$

$$\overline{\mathrm{WE}^{\circ}} = C_2 \cdot \mathrm{WE}'$$

- 以上带 ‘的表示信号来自微程序控制器，持续一个CPU周期
- 读写时序信号$\mathrm{R}D'$、 WE'是受到控制的信号
- 而节拍脉冲信号$\overline{RD^{\circ}}$、 $\overline{\mathrm{WE}^{\circ}}$是计算机加上电源后就产生

5.3.2 时序信号产生器

3、节拍脉冲和读/写时序的译码



5.3.2 时序信号产生器

4、启停控制逻辑

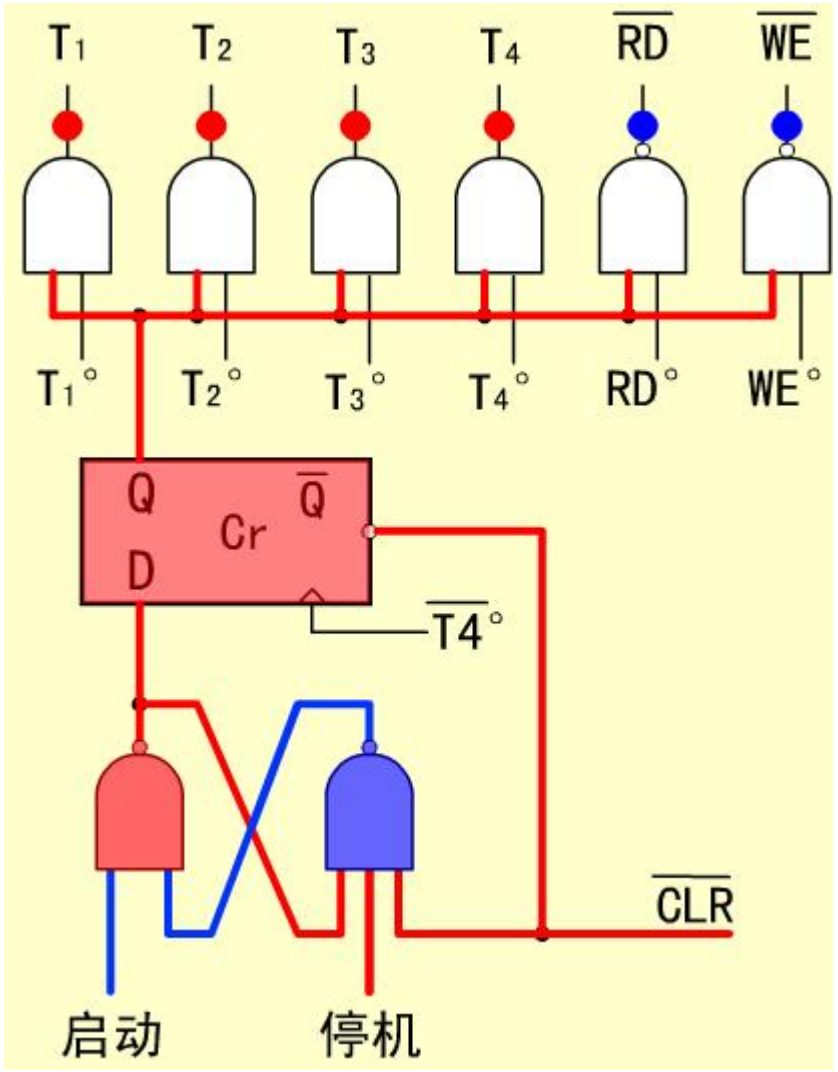
- 机器一旦接通电源，就会自动产生原始的节拍脉冲信号$T_1^\circ \sim T_4^\circ$
- 然而，只有在启动机器运行的情况下，才允许时序产生器发出CPU工作所需的节拍脉冲$T_1 \sim T_4$
- 为此需要启停控制逻辑来控制$T_1^\circ \sim T_4^\circ$的发生
- 同样，对读/写时序信号也要由启停逻辑加以控制

5.3.2 时序信号产生器

4、启停控制逻辑

- 核心是一个运行标志触发器C_r
- C_r的输出与原始节拍脉冲信号相与
- 当运行触发器为“1”时，打开时序电路
- 当运行触发器为“0”时，关闭时序产生器

- 由于启动和停机是随机的，一定要从第1个节拍脉冲前沿开始工作，停机时一定要在第4个节拍脉冲结束后关闭时序产生器，T_4°用于实现这个目的



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（指令的机器周期和时钟周期数不变），可选三种方案
  - （1）完全统一的机器周期执行各种不同的指令。所有指令周期具有相同的节拍电位数和相同的节拍脉冲数。缺点是对简单指令和简单操作来说，将造成时间浪费
  - （2）采用不定长机器周期。将大多数操作安排在一个较短的机器周期内完成，对某些时间紧张的操作，采取延长机器周期的办法来解决
  - （3）中央控制与局部控制的结合。将大部分指令安排在固定的机器周期完成，称为中央控制，对少数复杂指令(乘、除、浮点运算)采用另外的时序进行定时，称为局部控制。

5.3.3 控制方式

- 异步控制方式
  - 每条指令需要多长时间就占多长时间
  - 每条指令的指令周期可由多少不等的机器周期数组成，也可以是当控制器发出某一操作控制信号后，等待执行部件完成操作后发回“回答”信号，再开始新的操作
  - 没有固定的CPU周期数或严格的时钟周期

- 联合控制方式，两种方式
  - （1）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（以执行部件的“回答”信号作为本次操作的结束）
  - （2）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）

5.3.3 控制方式

1、同步控制方式

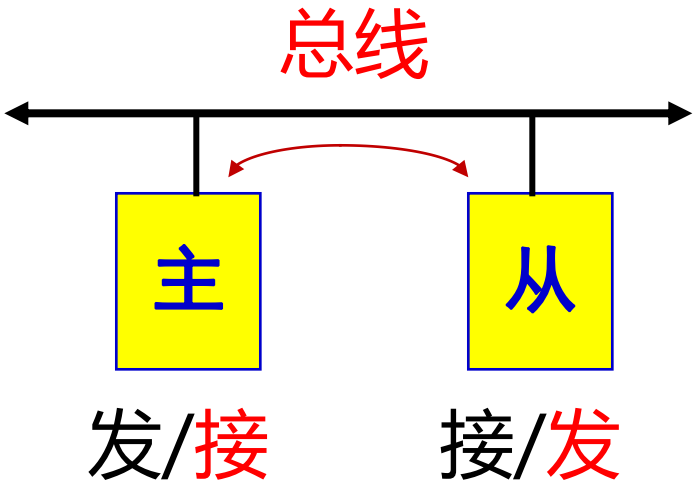
- 定义：各项操作受统一时序控制
- 特点：有明显时序时间划分，时钟周期时间固定，各步操作的衔接、各部件之间的数据传送受严格同步定时控制

- 优点：时序关系简单，时序划分规整，控制不复杂；控制逻辑易于集中，便于管理
- 缺点：时间安排不合理
- 应用场合：用于CPU内部、设备内部、系统总线操作（各挂接部件速度相近，传送时间确定，传送距离较近）

5.3.3 控制方式

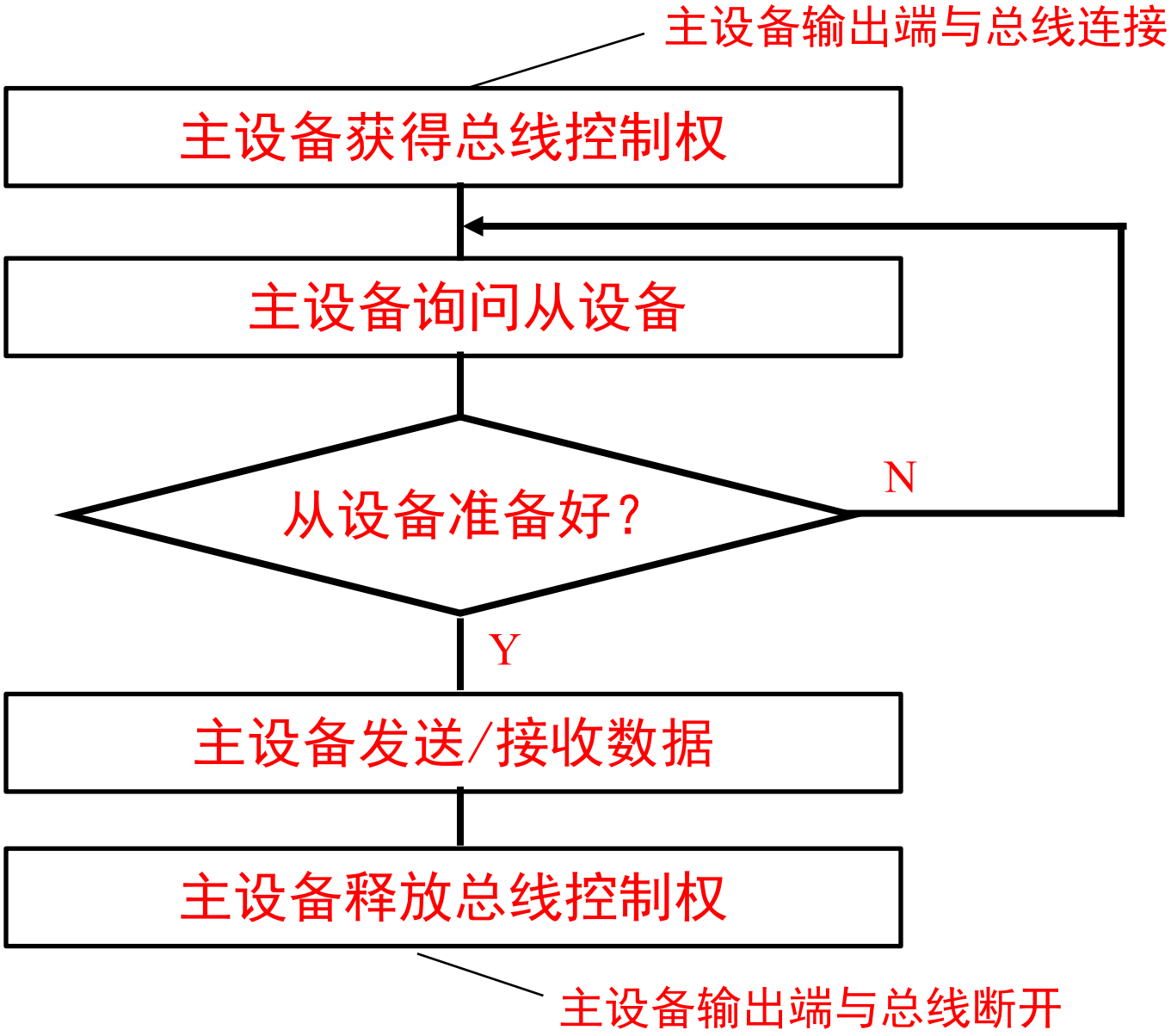
2、异步控制方式

- 定义：各项操作按不同需要安排时间，不受统一时序控制
- 特点：无统一时钟周期划分，各操作间的衔接和各部件之间的信息交换采用异步应答方式
- 例：异步传送操作
- 主设备：申请并掌握总线权的设备
- 从设备：响应主设备请求的设备



5.3.3 控制方式

- 操作流程



5.4 微程序控制器

控制方式

- 早期指令集比较简单，往往采用硬布线控制器
- 随着指令数量增加，控制变得复杂
- 微程序控制器逐渐取代了硬布线控制器
- 基本思想
  - 仿照解题的方法，把操作控制信号编制成微指令，存放到控制存储器里
  - 运行时，从控存中取出微指令，产生指令运行所需的操作控制信号，使相应部件执行所规定的操作
  - 从上述可以看出，微程序设计技术是用软件方法来设计硬件的技术
- 优点：规整性、灵活性、可维护性

控制方式

- 发展
  - 微程序的概念和原理是由英国剑桥大学的M·V·Wilkes教授于1951年在曼彻斯特大学计算机会议上首先提出来的，当时还没有合适的存放微程序的控制存储器的元件
  - 到1964年，IBM公司在IBM 360系列机上成功地采用了微程序设计技术
  - 20世纪70年代以来，由于VLSI技术的发展，推动了微程序设计技术的发展和应用
  - 目前，从大型机到小型机、微型机都普遍采用了微程序设计技术

提纲

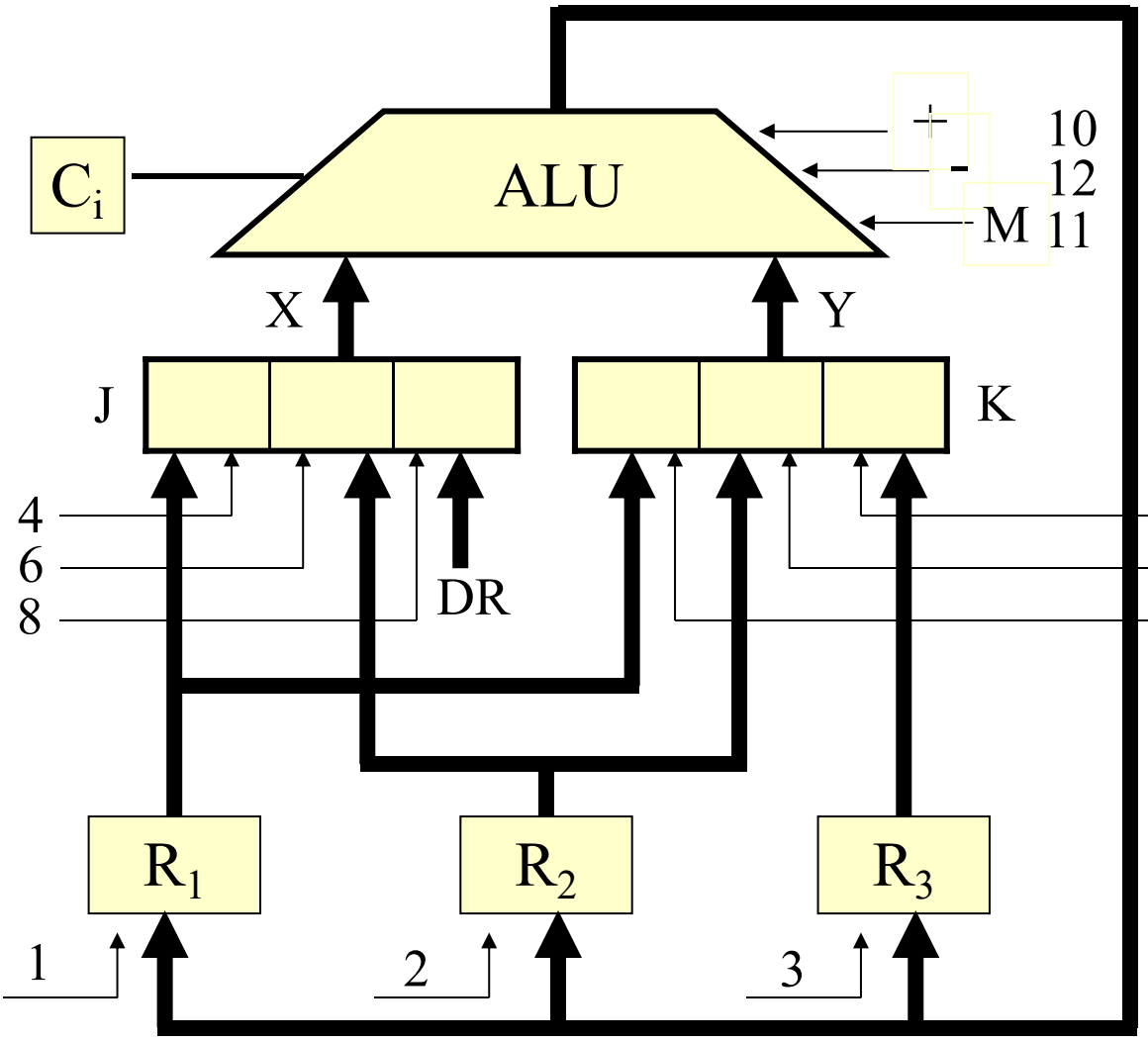
5.4.1 微程序控制原理

1、微命令和微操作

- 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
- 微操作：执行部件接受微命令后所进行的操作
  - 微操作是执行部件中最基本的操作

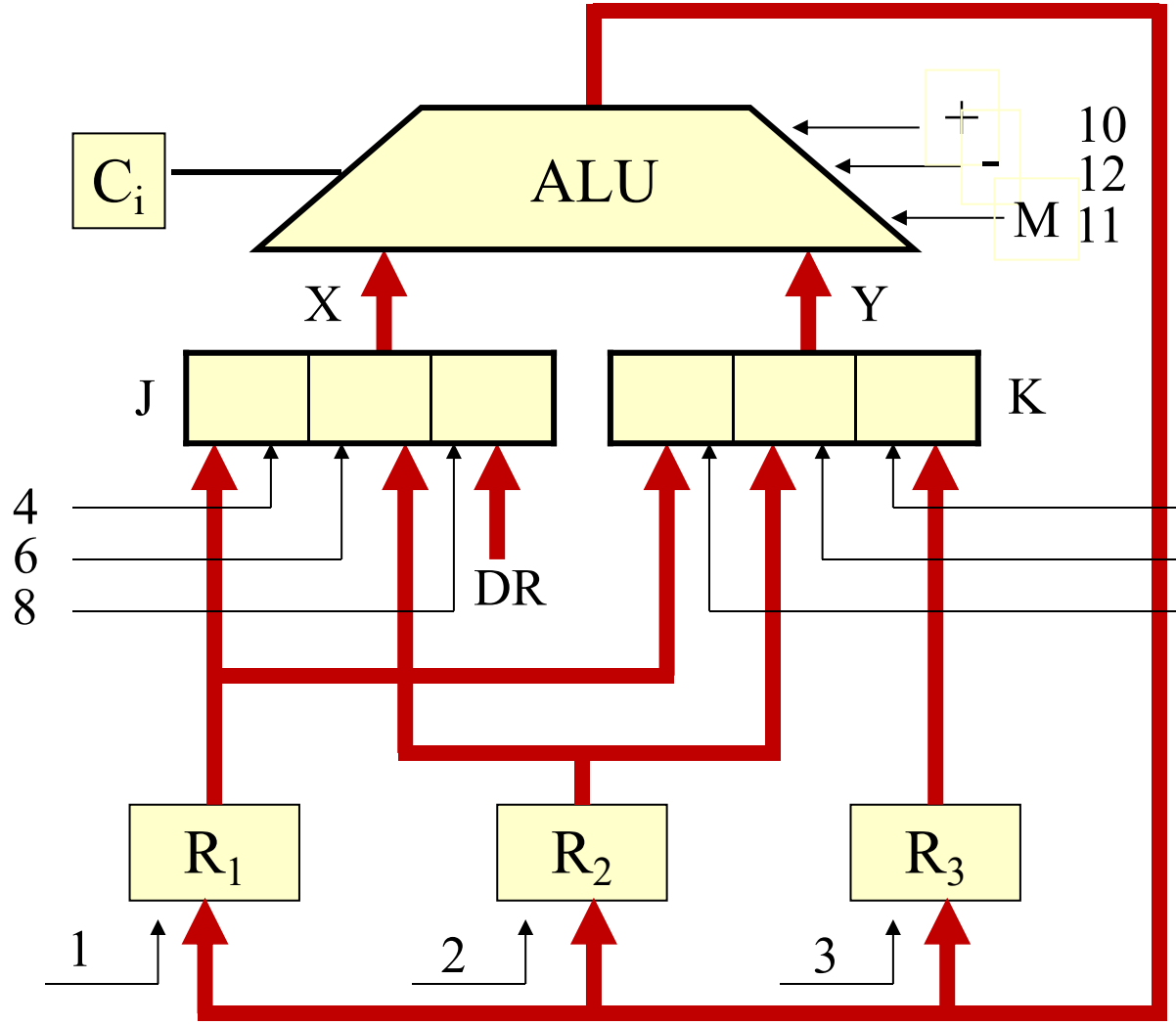
- 微命令是控制计算机各部件完成某个基本微操作的命令
- 微命令和微操作是一一对应的
- 微命令是微操作的控制信号，微操作是微命令的操作过程

5.4.1 微程序控制原理



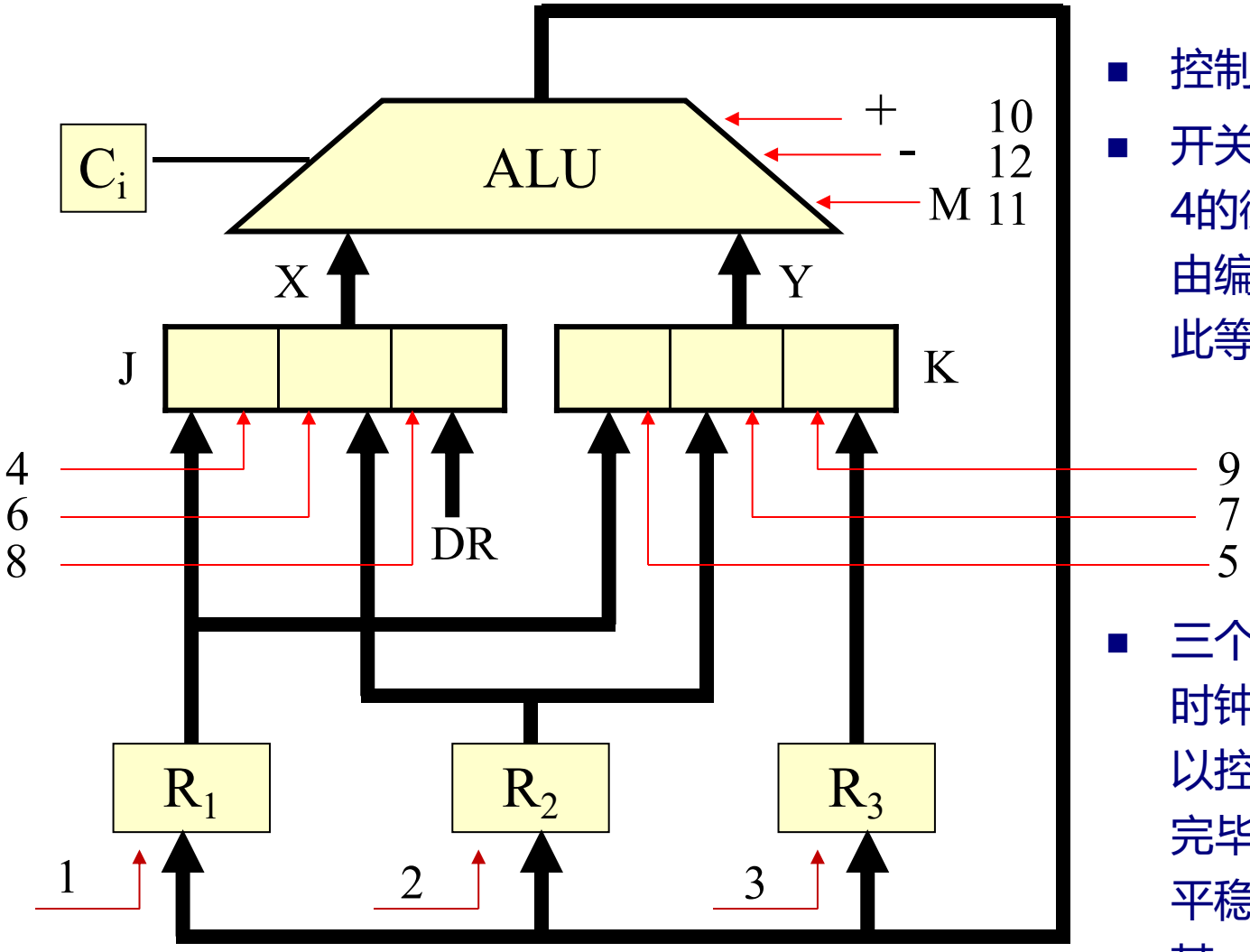
- 一个简单运算器数据通路模型
- ALU 为算术逻辑单元
- R_1、R_2、R_3为三个寄存器
- 其内容可以通过多路开关从ALU的X端或Y端送至ALU
- ALU的输出可以送往任何一个寄存器或同时送往R_1、R_2、R_3三个寄存器

5.4.1 微程序控制原理



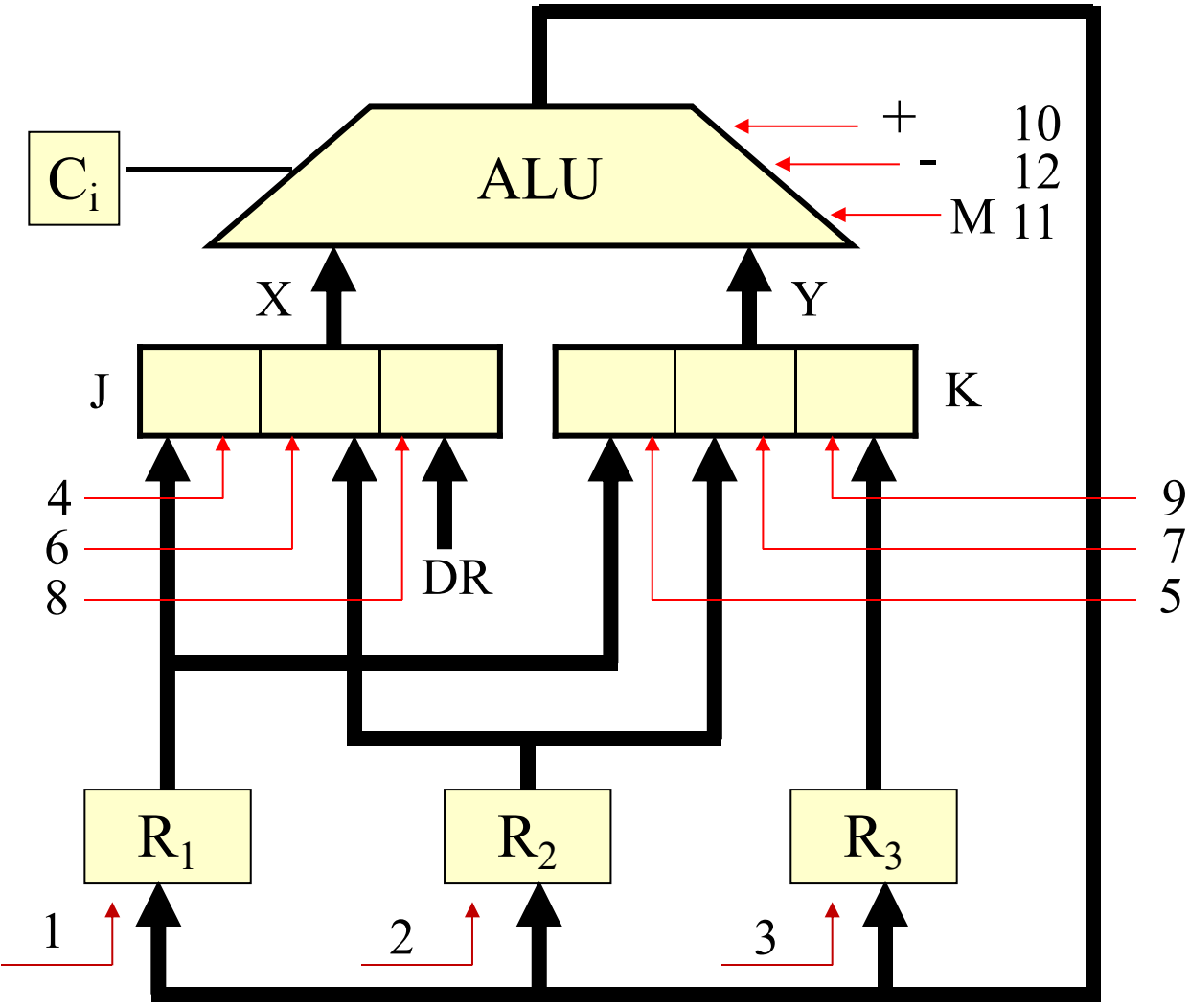
- 数据通路
- 多路开关的每个控制门是一个常闭的开关，它的一个输入端代表来自寄存器的信息，而另一个输入端则作为操作控制端
- 一旦两个输入端都有输入信号时，它才产生一个输出信号，从而在控制线能起作用的一个时间宽度中来控制信息在部件中流动
- 图中每个开关门由控制器中相应的微命令来控制

5.4.1 微程序控制原理



- 控制信号
- 开关门4由控制器中编号为4的微命令控制，开关门6由编号为6的微命令控，如此等等
- 三个寄存器R_1、R_2、R_3的时钟输入端1、2、3也要加以控制，以便在ALU运算完毕而输出公共总线上电平稳定时，将结果打入到某一寄存器

5.4.1 微程序控制原理



1	R_1输入
2	R_2输入
3	R_3输入

寄存器的输入脉冲

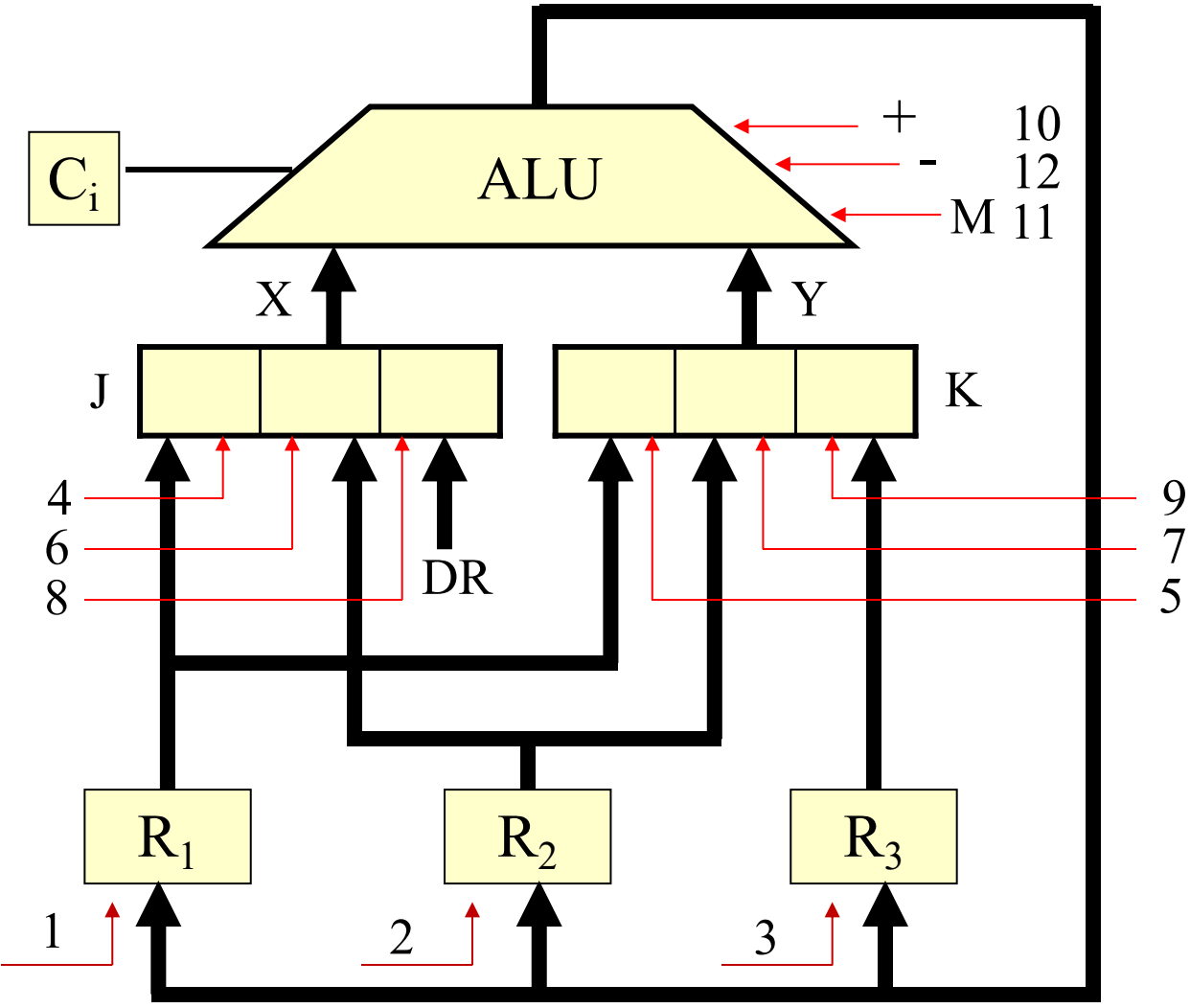
4	R_1送X
5	R_1送Y
6	R_2送X
7	R_2送Y
8	DR送X
9	R_3送Y

多路数据选择器的选择端

10	加
11	传送
12	减

运算控制端

5.4.1 微程序控制原理



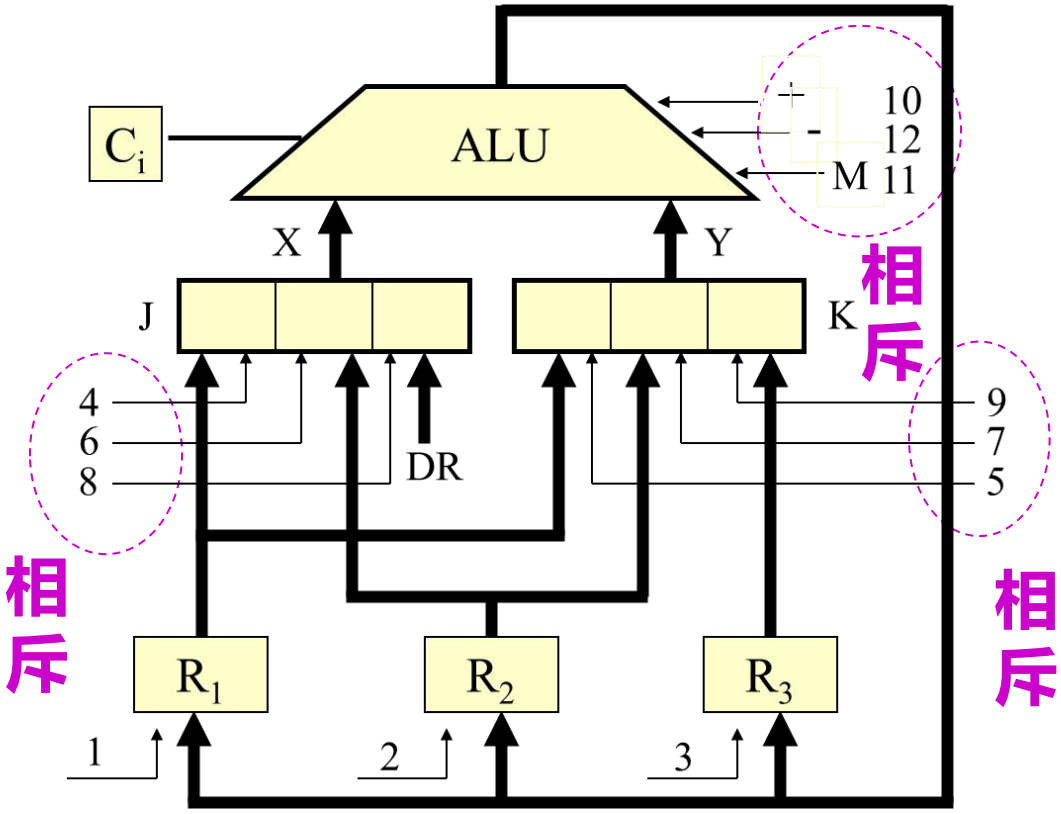
R1+R2→R3

4　R_1送X
7　R_2送Y
10　加
3　R_3输入

5　R_1送Y
6　R_2送X
10　加
3　R_3输入

5.4.1 微程序控制原理

- 相容性微命令：在同一个CPU周期中，可以同时执行的微操作命令
  - 1、2、3
  - 4、6、8与5、7、9任意两个微指令
- 相斥性微命令：在同一个CPU周期中，不能同时执行的微操作操作
  - ALU的+、-、M（传送）微命令
  - 4、6、8
  - 5、7、9



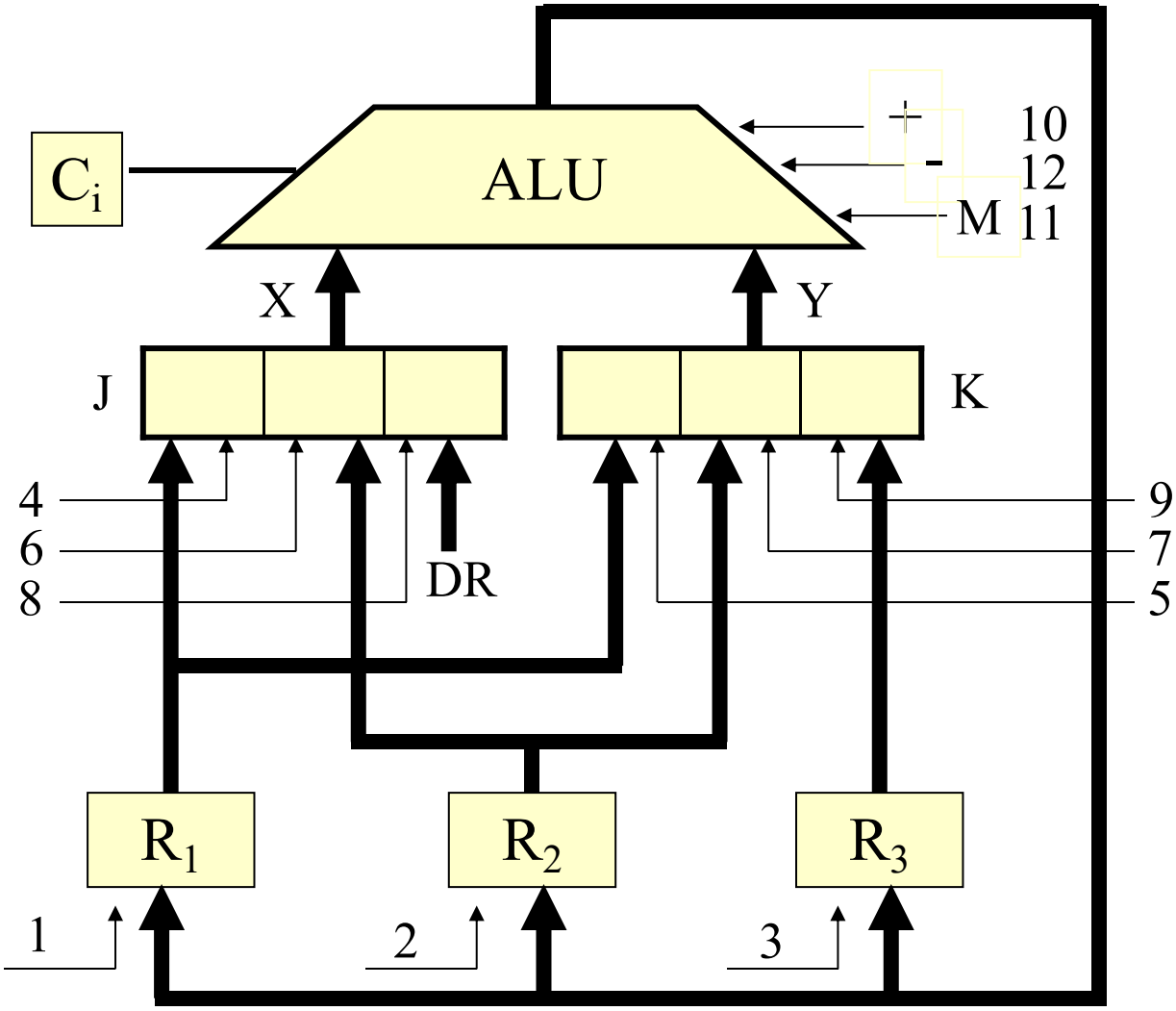
5.4.1 微程序控制原理

2、微指令和微程序

- 微指令：把在同一CPU周期内并行执行的微操作控制信息，存储在控制存储器里，称为一条微指令（Microinstruction）
  - 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
  - 微地址：存放微指令的控制存储器的单元地址

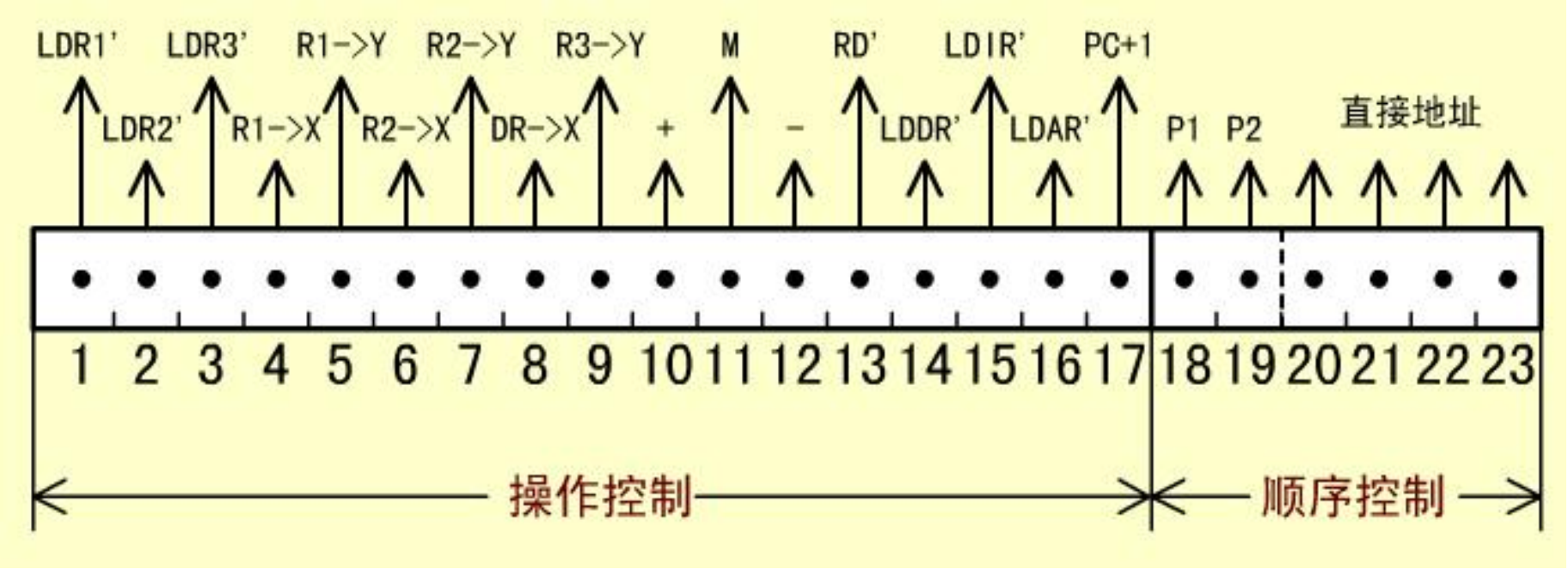
- 微程序：一系列微指令的有序集合就是微程序
  - 一段微程序对应一条机器指令

5.4.1 微程序控制原理



5.4.1 微程序控制原理

- 微指令基本格式



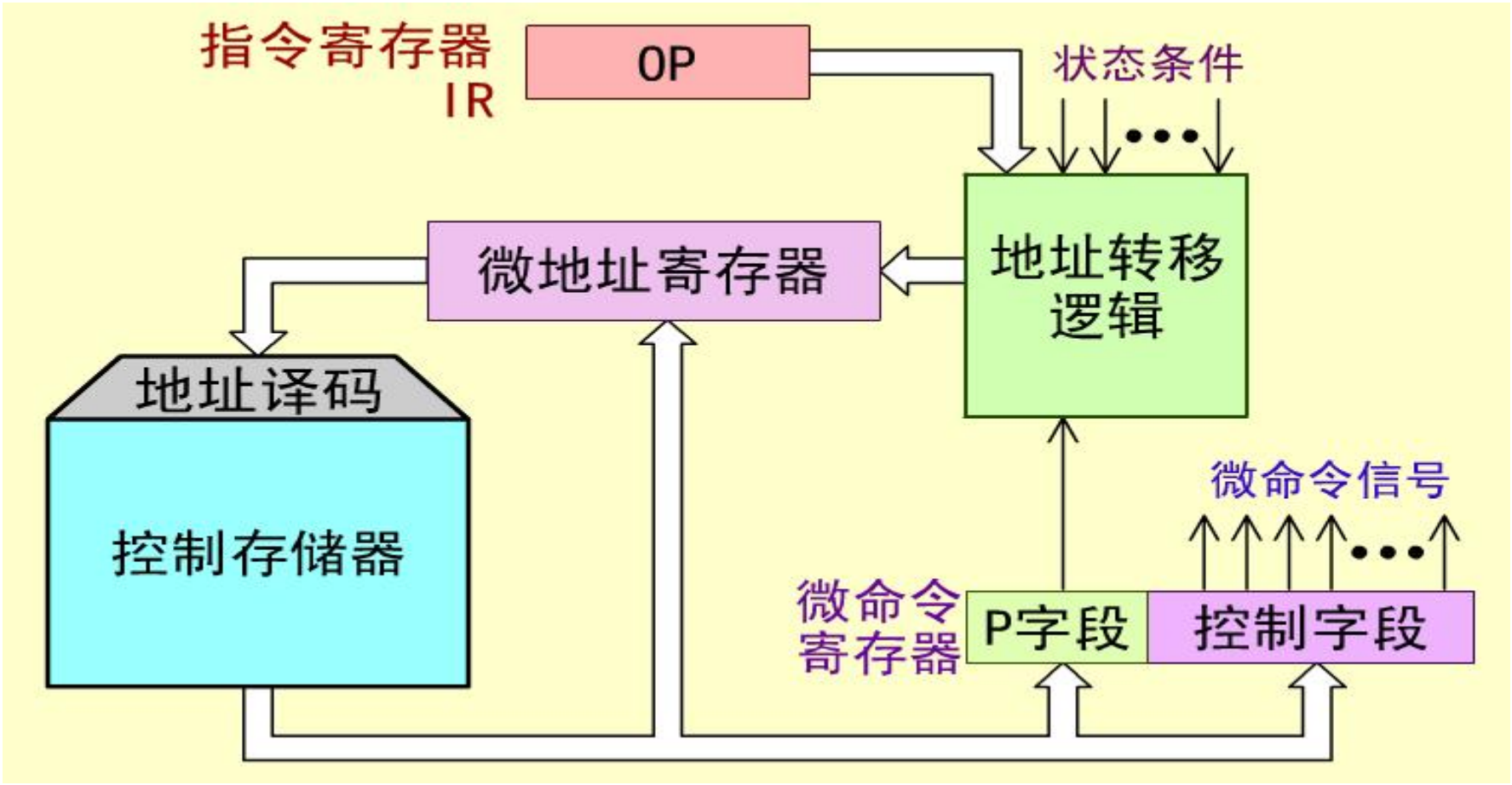
- 字长为23位
- 操作控制的每一位表示一个微命令

5.4.1 微程序控制原理

- 一条微指令通常至少包含两大部分信息：
  - **操作控制字段**，又称微操作码字段，用以产生某一步操作所需的各个微操作控制信号。
    - 某位为1，表明发微指令
    - 微指令发出的控制信号都是节拍电位信号，持续时间为一个CPU周期
    - 微命令信号还要引入时间控制。例如：同节拍脉冲T_4相与而得到LDR_1-LDR_3，从而保证运算器在前部分进行运算， T_4脉冲到来时将结果打入到相应寄存器
  - **顺序控制字段**，又称微地址码字段，用以控制产生下一条要执行的微指令地址
    - 4位（20~23）用来直接给出下一条微指令的地址
    - 18、19两位作为判断测试标志。当此两位为“0”时,表示不进行测试，直接按顺序控制字段第20~23位给出的地址取下一条微指令；当第18位或第19位为“1”时，表示要进行P1或P2的判断测试，根据测试结果，需要对第20~23位的某一位或几位进行修改，然后按修改后的地址取下一条微命令

5.4.1 微程序控制原理

3、微程序控制器原理框图



5.4.1 微程序控制原理

- 控制存储器(μCM)：这是微程序控制器的核心部件，用来存放实现全部指令系统的微程序。其性能（包括容量、速度、可靠性等）与计算机的性能密切相关
- 微指令寄存器(μIR)
  - 用来存放从μCM取出的正在执行的微指令，它的位数同微指令字长相等
- 地址转移逻辑
  - 用来产生初始微地址和后继微地址，以保证微指令的连续执行
- 微地址寄存器(μMAR)
  - 它接受微地址形成部件送来的微地址，为下一步从μCM中读取微指令作准备

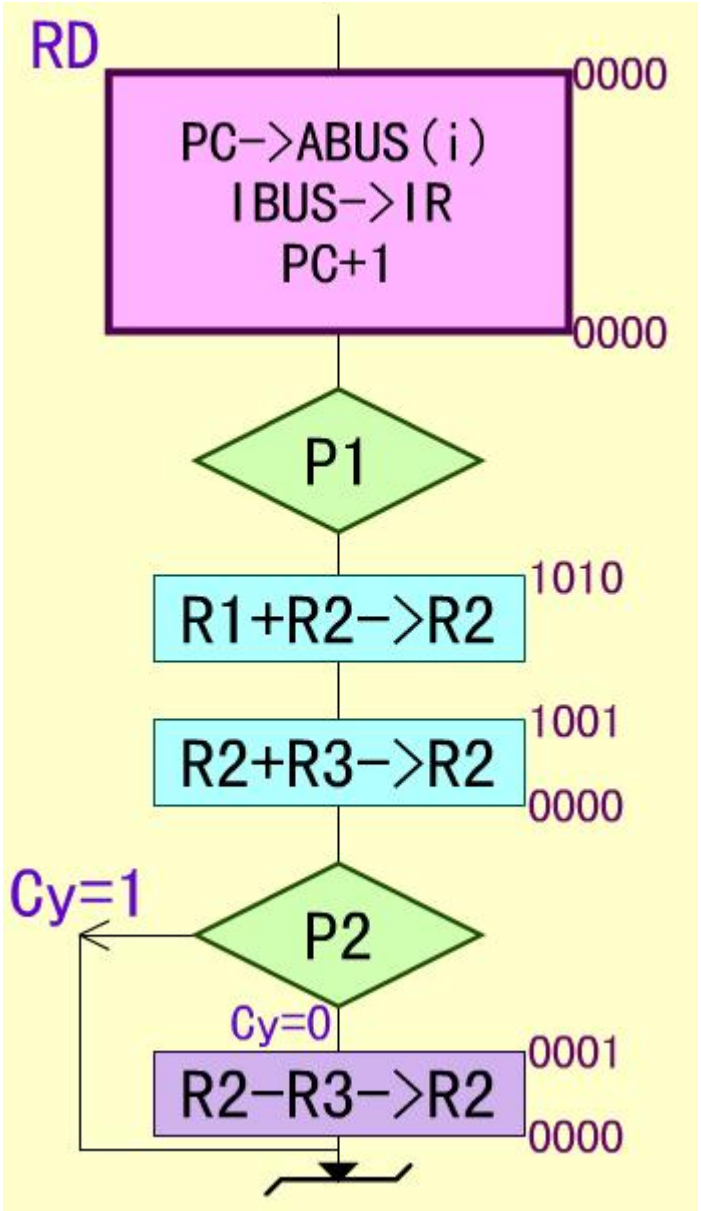
5.4.1 微程序控制原理

4、微程序举例

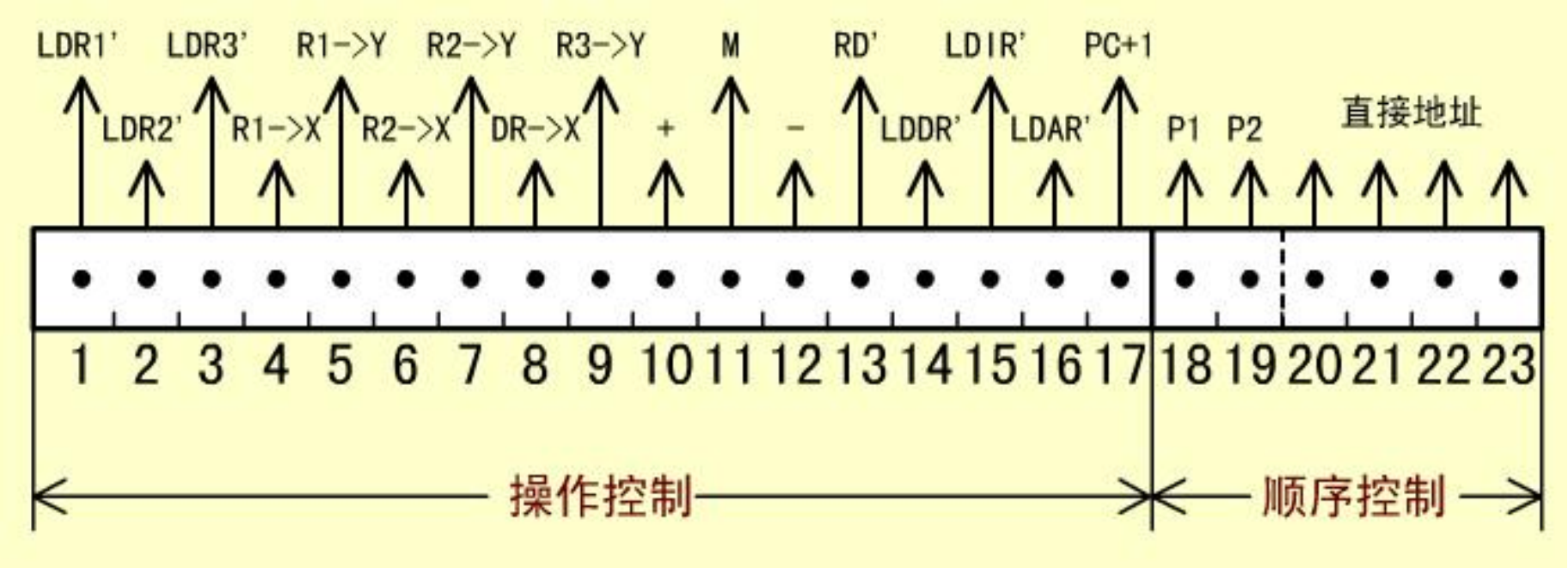
- 一条机器指令对应一个微程序
- 微程序的总和可实现整个指令系统
- 微程序执行过程举例（十进制加法）
  - 先进行a + b + 6的运算，然后判断结果有无进位
  - 当进位标志C_y=1，不减6；当C_y=0，减去6，从而获得正确结果
  - 设数a和b已存放在R1和R2两寄存器中，数6存放在R3寄存器中

5.4.1 微程序控制原理

- 十进制加法微程序流程图
  - 四条微指令，每一条微指令用一个长方框表示
  - 第一条为取机器指令的微指令
  - 每一条微指令的地址用数字示于长方框的右上角
  - 菱形符号代表判别测试
  - P1：机器指令的操作码测试
  - P2：进位标志测试



5.4.1 微程序控制原理


LDR1'
LDR3'
R1->Y
R2->Y
R3->Y
M
RD'
LDIR'
PC+1
LDR2'
R1->X
R2->X
DR->X
+
-
LDDR'
LDAR'
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

- 实现“十进制加法”的微程序的各条微指令编码如下：

第一条微指令的二进制编码是：

000 000 000 000 11111100000

第二条微指令的二进制编码是：

010 100 100 100 00000001001

第三条微指令的二进制编码是：

010 001 001 100 00000010000

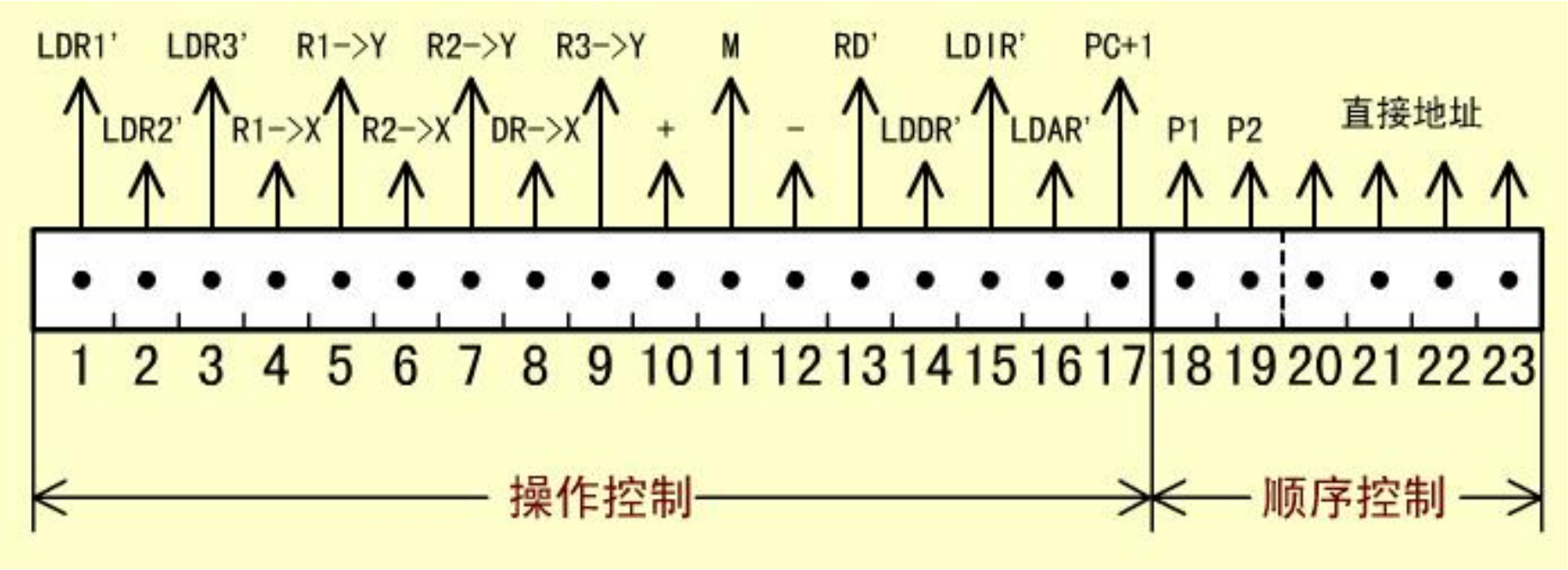
第四条微指令的二进制编码是：

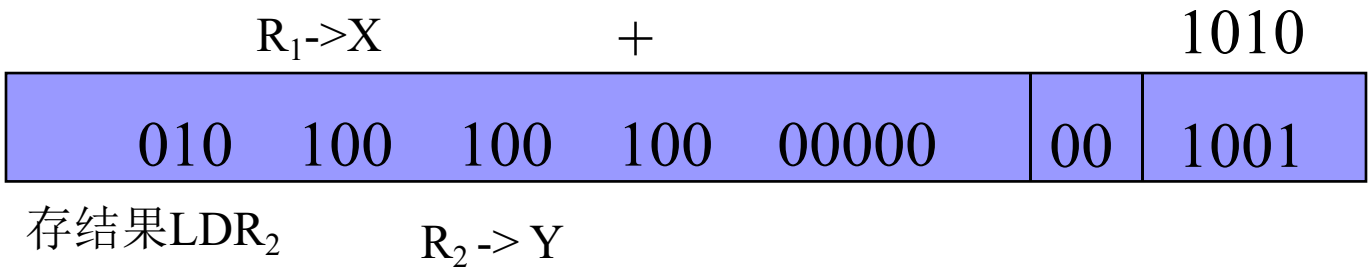
010 001 001 001 00000000000

组成一段微程序

5.4.1 微程序控制原理

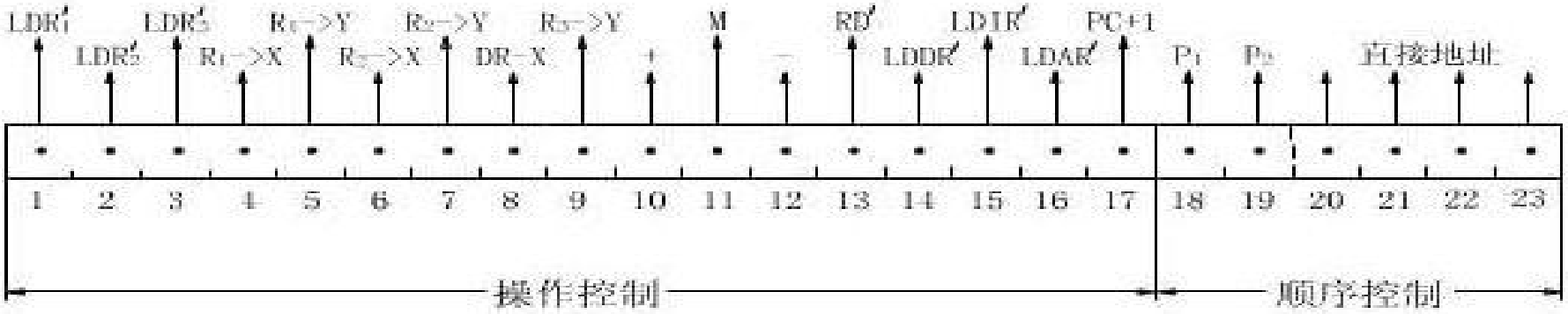
- 四条微指令如下





5.4.1 微程序控制原理

- 说明：关于P1、P2测试的约定
- P1测试的约定：若P1=1，则进行P1测试： 将机器指令的操作码OP作为下一条微指令的地址。
- P2测试的约定： 若P2=1，则进行P2测试：根据进位Cy的状态，决定下一条微指令的地址。
  - 若C_y=1，则当前微指令给出的后继地址0000就是下一条微指令的地址；
  - 若C_y=0，则下一条微指令的地址为0001（要执行 S-6 运算）



R_2 -> X　　+　　1001

010	001	001	100	00000	01	0000

存结果　　R_3 -> Y　　P2判别

LDR_2

P2条件为0，转到取指令的公操作

R_2 -> X　　-　　0001

010	001	001	001	00000	00	0000

存结果　　R_3 -> Y

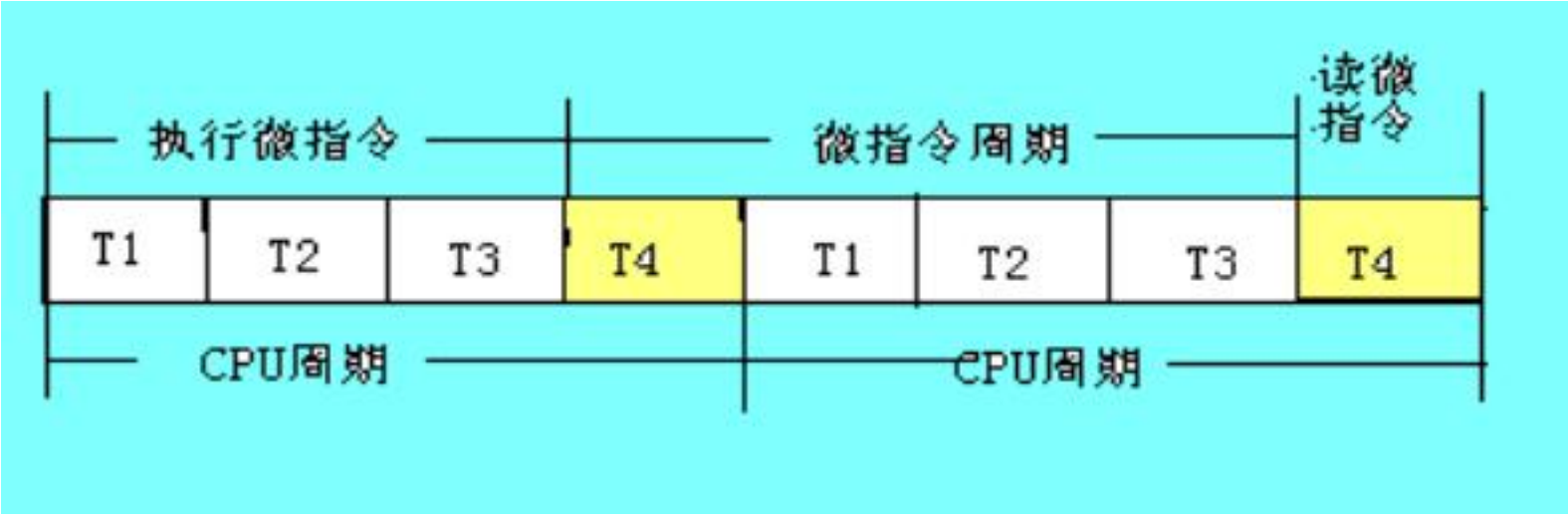
LDR_2

转到取指令的公操作

5.4.1 微程序控制原理

5、CPU周期和微指令周期的关系

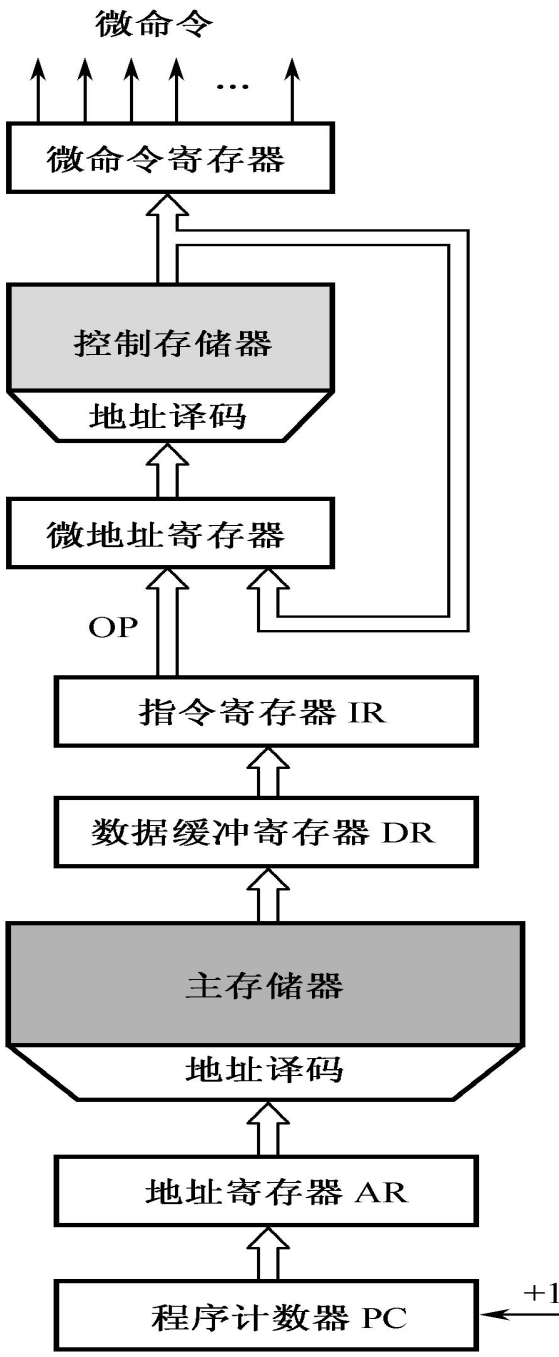
- 在串行方式的微程序控制器中，微指令周期等于读出微指令的时间加上执行该条微指令的时间
- 可以将一个微指令周期时间设计得恰好和CPU周期时间相等
- 例如：T_4读微指令的时间，$T_1+T_2+T_3$执行微指令的时间



5.4.1 微程序控制原理

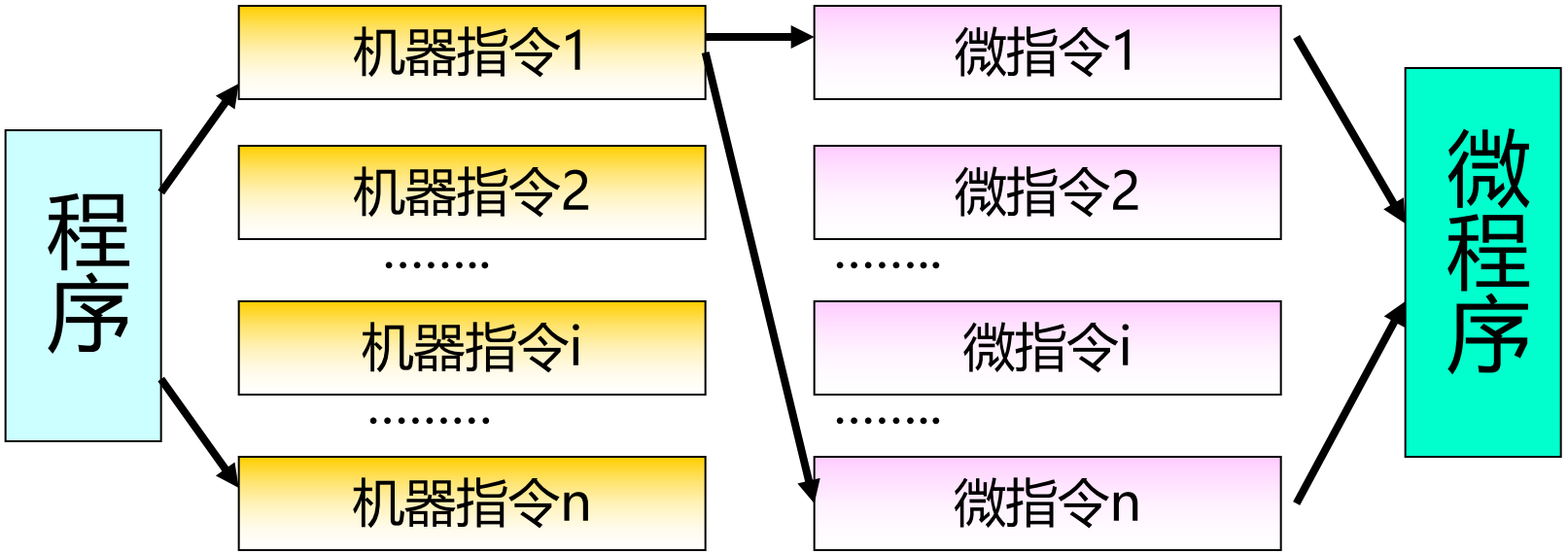
6、机器指令与微指令的关系

- 一条机器指令对应一段微程序，这段微程序是由若干条微指令序列组成的。因此，一条机器指令的功能是由若干条微指令组成的序列来实现的。简言之，一条机器指令所完成的操作划分成若干条微指令来完成，由微指令进行解释和执行
- 从指令与微指令，程序与微程序，地址与微地址的一一对应关系来看，前者与内存储器有关，后者与控制存储器有关。与此相关，也有相对应的硬件设备
- 每一个CPU周期对应一条微指令



5.4.1 微程序控制原理

5、机器指令与微指令的关系



5.4.2 微程序设计技术

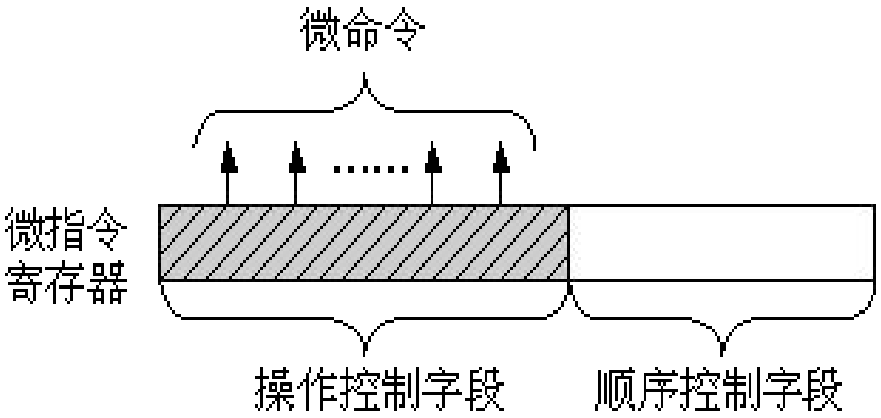
6、微程序设计技术

- 设计微指令应当追求的目标
  - 有利于缩短微指令的长度
  - 有利于减小控制存储器的容量
  - 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 微程序设计技术

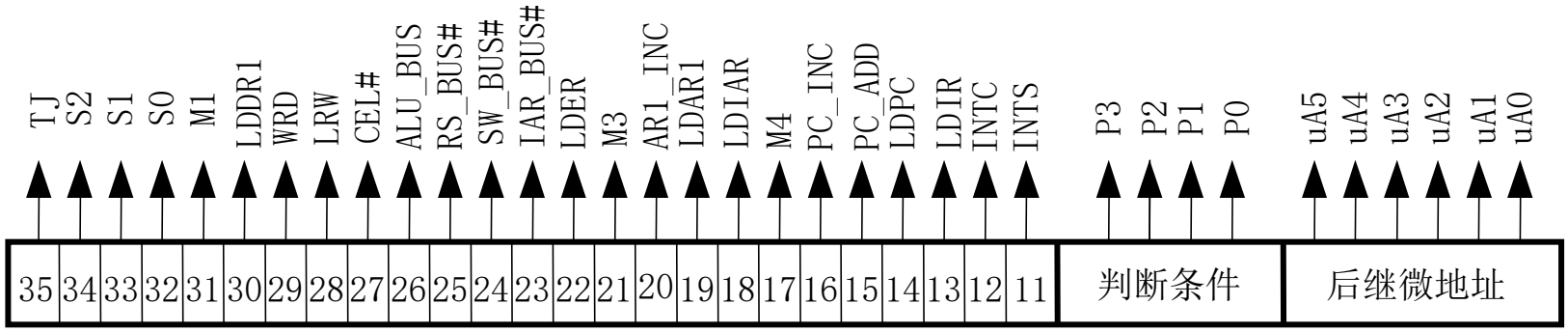
（1）微命令的编码方法

- 微命令编码：对微指令中的操作控制字段采用的表示方法
- 编码有三种方法：直接表示法/编码表示法/混合表示法
- 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（前面的例子就是直接表示法）
  - 每一位代表一个微命令
  - 优点：简单直观、输出直接用于控制
  - 缺点：微指令字较长，使控制存储器容量较大



5.4.2 微程序设计技术

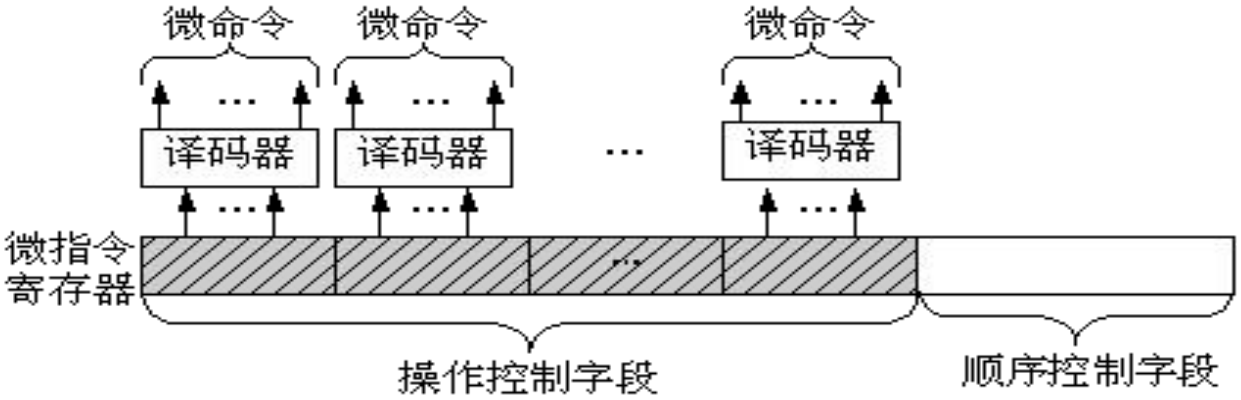
- 直接表示法举例，操作控制字段的每一个独立的二进制位代表一个微命令，该位为“1”表示这个微命令有效，为“0”表示这个微命令无效。



微指令格式举例（TEC_5实验平台格式）

5.4.2 微程序设计技术

- 编码表示法
  - 把一组相斥性的微命令信号组成一个小组（即一个字段），每段内采用最短编码法，段与段之间采用直接控制法



- 编码表示法特点：可以避免互斥，使指令字大大缩短，但增加了译码电路，使微程序的执行速度减慢
- 目前在微程序控制器设计中，该方法使用较普遍

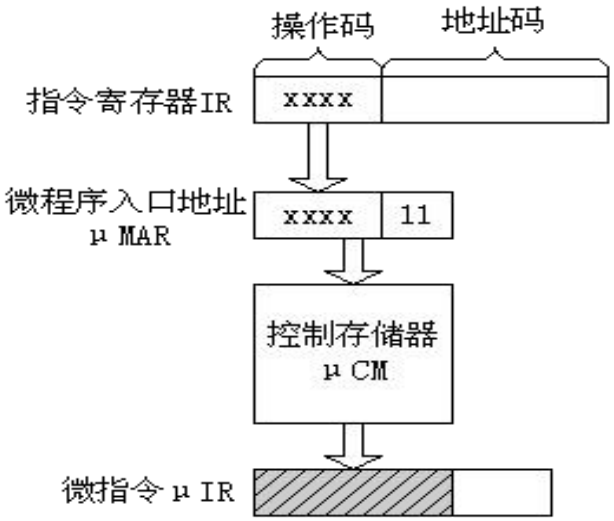
5.4.2 微程序设计技术

- 混合编码法
  - 将前两种结合在一起，兼顾两者特点
  - 一个字段的某些编码不能独立地定义某些微命令，而需要与其他字段的编码来联合定义

5.4.2 微程序设计技术

（2）微指令地址的形成方法

- 入口地址形成
  - 如果机器指令操作码字段的位数和位置固定，可以直接使操作码与微程序入口地址的部分位相对应
  - 如果操作码字段的位数和位置不固定，可以通过一定的映射关系得到入口地址

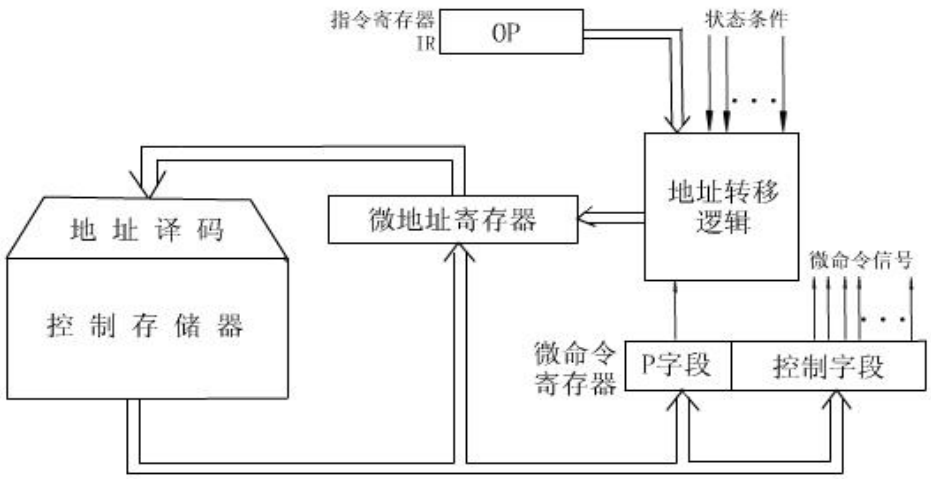


5.4.2 微程序设计技术

- 后继微地址形成方法
- ①计数器的方式
- 方法：
  - 微程序顺序执行时，其后继微地址就是现行微地址加上一个增量（通常为1）
  - 当微程序遇到转移或转子程序时，由微指令的转移地址段来形成转移微地址。
  - 在微程序控制器中也有一个微程序计数器μPC，一般情况下都是将微地址寄存器μMAR作为μPC
- 特点：
  - 优点是简单、易于掌握，编制微程序容易
  - 缺点是这种方式不能实现两路以上的并行微程序转移，因而不利于提高微程序的执行速度

5.4.2 微程序设计技术

- ②多路转移的方式
- 根据条件转移
  - 条件：状态条件/测试/微指令中微地址/操作码
  - 特点：能以较短的顺序控制字段配合，实现多路并行转移，灵活性好，速度较快，但转移地址逻辑需要用组合逻辑方法设计



5.4.2 微程序设计技术

- 【例2】微地址寄存器有6位(μA5-μA0)，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(IR3-IR0)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IR4，IR5的状态进行4路分支。

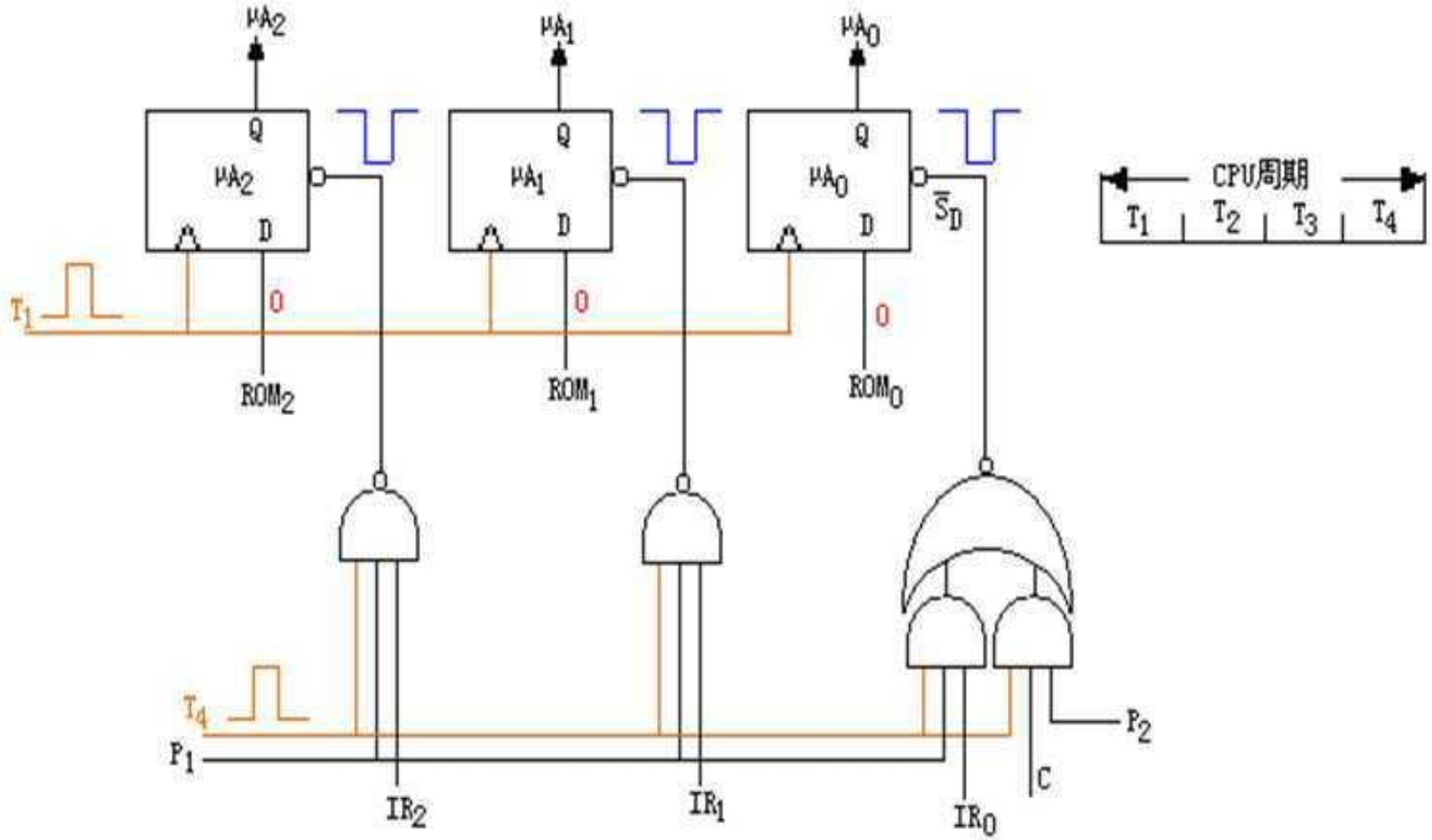
- 请按多路转移方法设计微地址转移逻辑。

5.4.2 微程序设计技术

- 按所给设计条件，微程序有三种判别测试，分别为P1，P2，P3。 由于修改μA5-μA0内容具有很大灵活性，现分配如下：
  - （1）用P1和IR3-IR0修改μA3-μA0；
  - （2）用P2和C修改μA0；
  - （3）用P3和IR5，IR4修改μA5，μA4。
- 另外还要考虑时间因素T4（假设CPU周期最后一个节拍脉冲），故转移逻辑表达式如下：
  - μA5=P3·IR5·T4
  - μA4=P3·IR4·T4
  - μA3=P1·IR3·T4
  - μA2=P1·IR2·T4
  - μA1=P1·IR1·T4
  - μA0=P1·IR0·T4+P2·C·T4
- 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

5.4.2 微程序设计技术

- 下图仅画出μA2、μA1、μA0触发器的微地址转移逻辑图



5.4.2 微程序设计技术

（3）微指令格式

- ①水平型微指令
- 水平型微指令是指一次能定义并能并行执行多个微命令的微指令（前面讲的就是水平型微指令）
- 格式如下

控制字段	判别测试字段	下地址字段

5.4.2 微程序设计技术

- 水平型微指令特点：
  - 优点：
    - 微指令字较长，速度较快
    - 微指令中的微操作有高度的并行性
    - 微指令译码简单
    - 控制存储器的纵向容量小，灵活性强
  - 缺点：
    - 微指令字比较长，明显地增加了控制存储器的横向容量
    - 水平微指令与机器指令差别很大，一般要熟悉机器结构、数据通路、时序系统以及指令执行过程的人才能进行微程序设计，这对用户来说是很困难的

5.4.2 微程序设计技术

②垂直型微指令：采用编码方式。

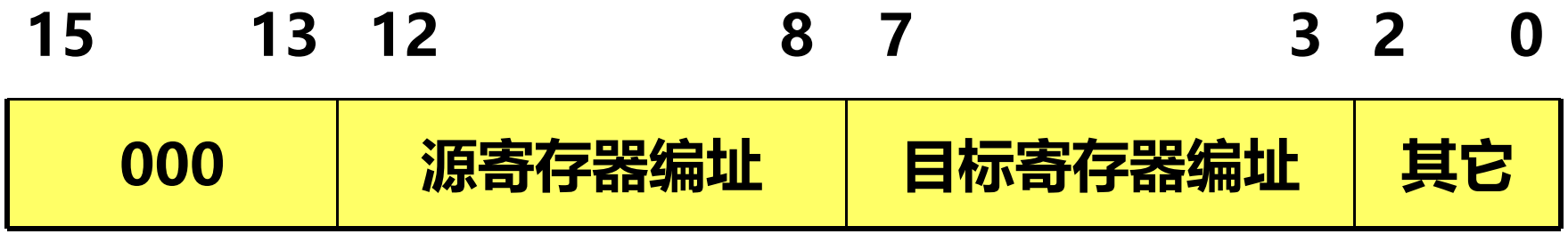
- 微指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能
- 结构类似于机器指令的结构
- 在一条微指令中只有1-2个微操作命令
- 每条微指令的功能简单

- 特点：实现一条机器指令的微程序要比水平型微指令编写的微程序长得多。它是采用较长的微程序结构去换取较短的微指令结构

5.4.2 微程序设计技术

- 设：微指令字长为16位，微操作码3位

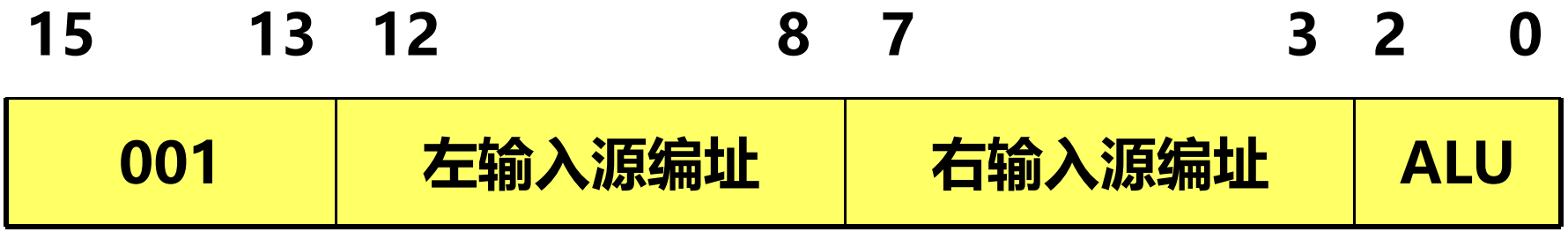
- 寄存器-寄存器传送型微指令



- 其功能是把源寄存器数据送目标寄存器
- 13—15位为微操作码（下同），源寄存器和目标寄存器编址各5位，可指定32个寄存器

5.4.2 微程序设计技术

- 运算控制型微指令



- 其功能是选择ALU的左、右两输入源信息，按ALU字段所指定的运算功能（8种操作）进行处理，并将结果送入暂存器中
- 左、右输入源编址可指定32种信息源之一

5.4.2 微程序设计技术

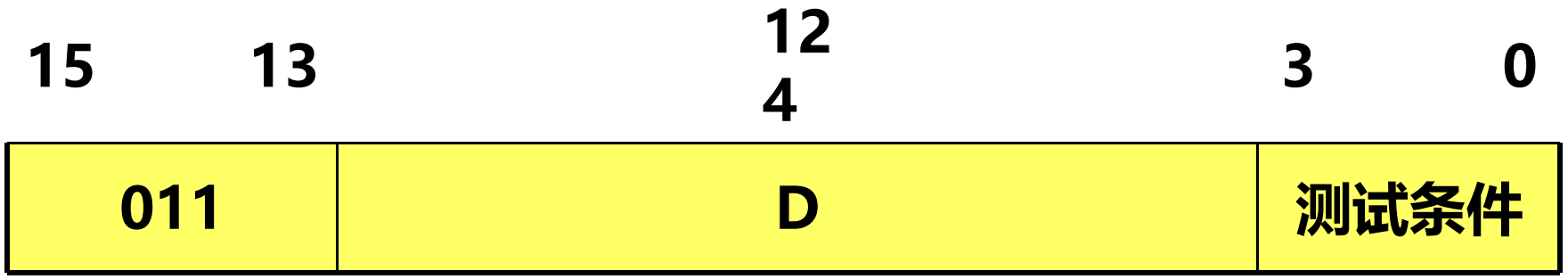
- 访问主存微指令

15 13	12 8	7 3	2 1	0
010	寄存器编址	存储器编址	读写	其他

- 其功能是：将主存中一个单元的信息送入寄存器或者将寄存器的数据送往主存。存储器编址是指按规定的寻址方式进行编址。第1，2位指定读操作或写操作（取其之一）

5.4.2 微程序设计技术

- 条件转移微指令



- 其功能是：根据测试对象的状态决定是转移到D所指定的微地址单元，还是顺序执行下一条微指令。9位D字段不足以表示一个完整的微地址，但可以用来替代现行μPC的低位地址
- 测试条件字段有4位，可规定16种测试条件

5.4.2 微程序设计技术

- 水平型微指令和垂直型微指令的比较
  - 水平型微指令并行操作能力强，效率高，灵活性强，垂直型微指令则较差
  - 水平型微指令执行一条指令的时间短，垂直型微指令执行时间长
  - 由水平型微指令解释指令的微程序，有微指令字较长而微程序短的特点。垂直型微指令则相反
  - 水平型微指令用户难以掌握，而垂直型微指令与指令比较相似，相对来说，比较容易掌握
  - 垂直型微指令的设计思想在Pentium 4、安腾系列机中得到了应用

5.4.2 微程序设计技术

- （4）动态微程序设计

- 对应于一台计算机的机器指令只有一组微程序，这一组微程序设计好之后，一般无须改变而且也不好改变，这种微程序设计技术称为静态微程序设计
- 采用EEPROM作为控制存储器，可以通过改变微指令和微程序来改变机器的指令系统，这种微程序设计技术称为动态微程序设计