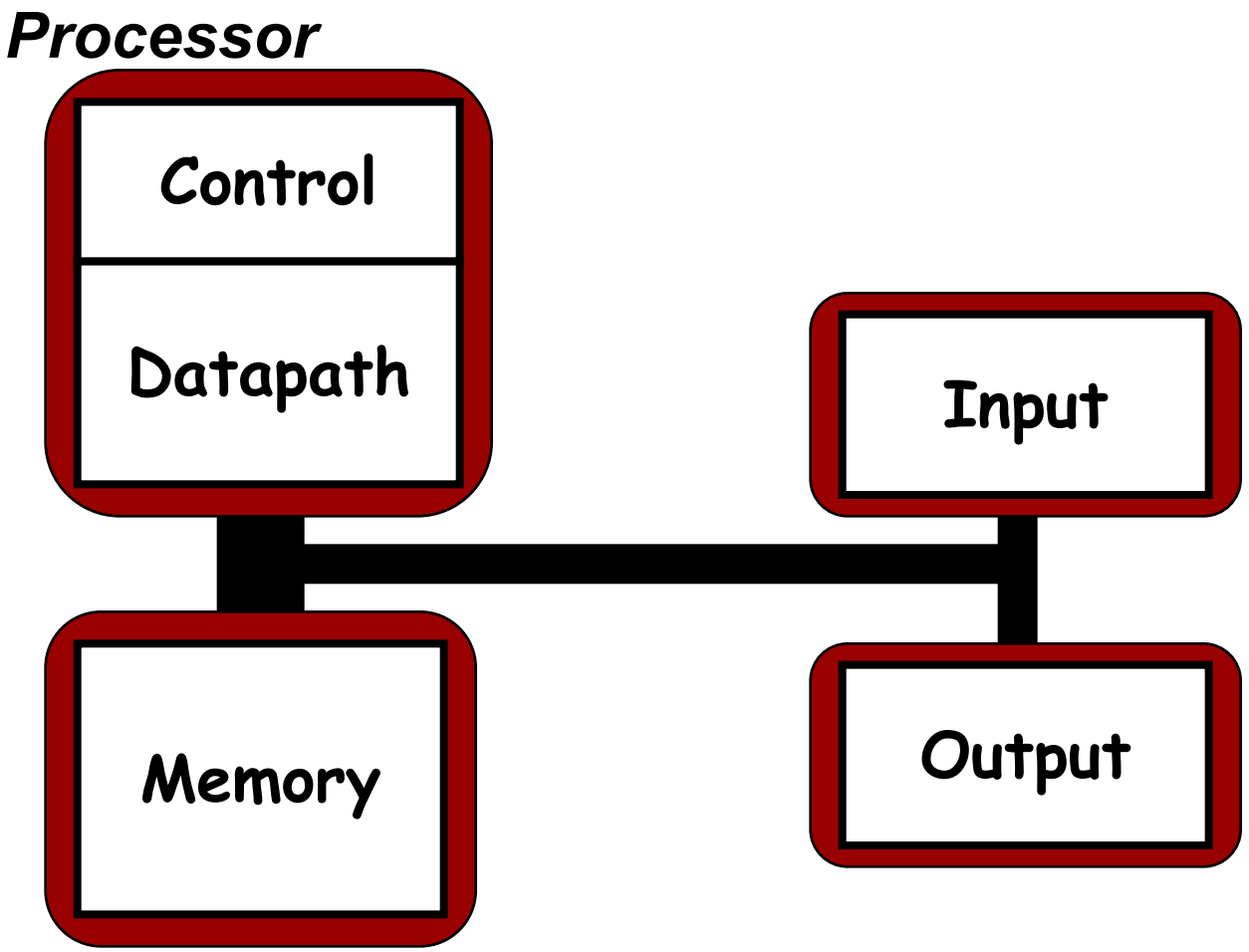

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS
北京邮电大学

第5章 中央处理器1

5.1 CPU的功能和组成

计算机的五个基本部件



处理器的功能

- 指令控制

  控制程序严格按规定的顺序执行

- 操作控制

  处理器根据从内存取出的每条指令产生相应的操作信号，送往相应的部件，控制这些部件按指令的要求进行动作

- 时间控制

  对各种操作实施时间上的控制

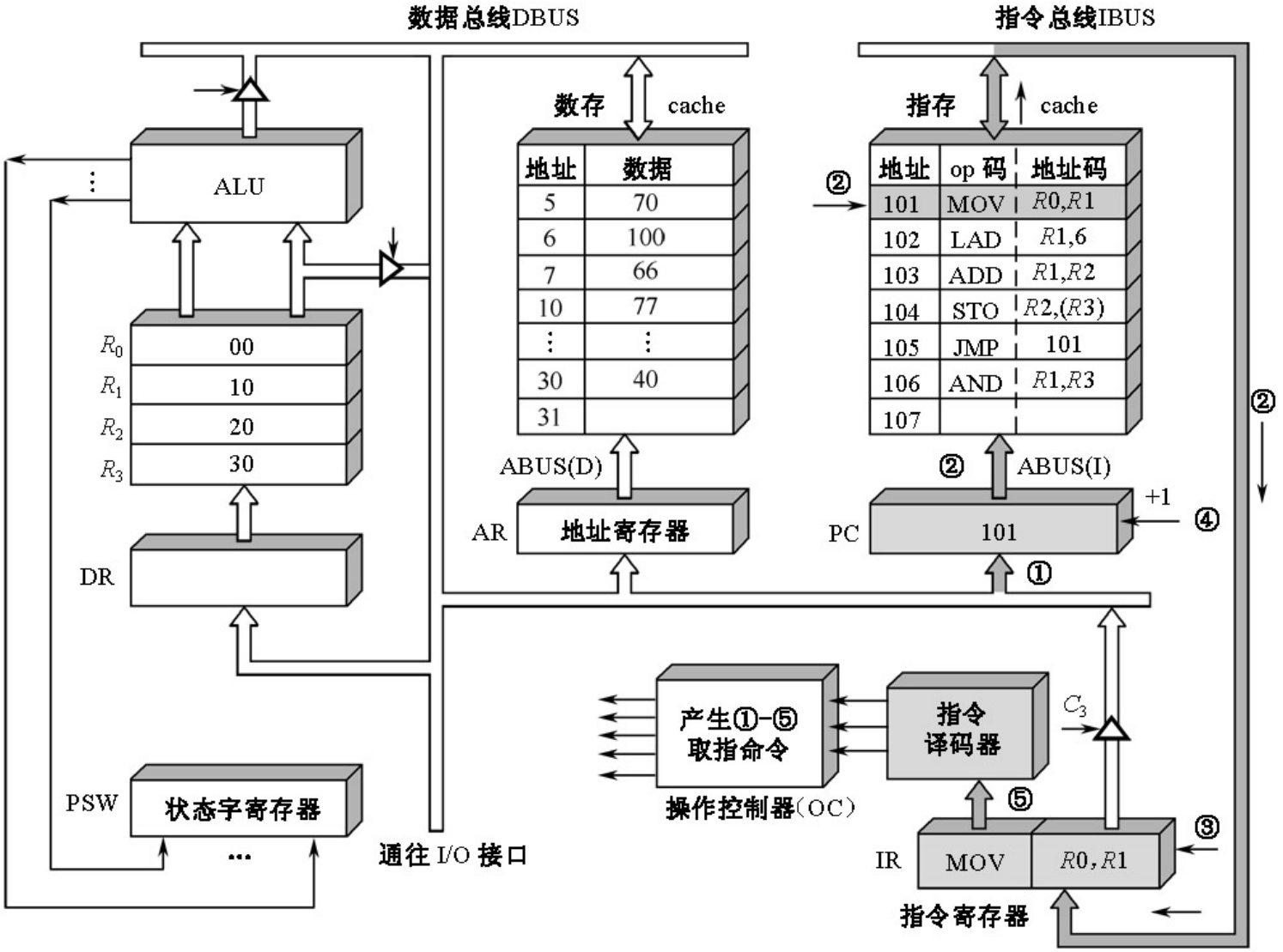
- 数据加工

  对数据进行算术运算和逻辑运算处理

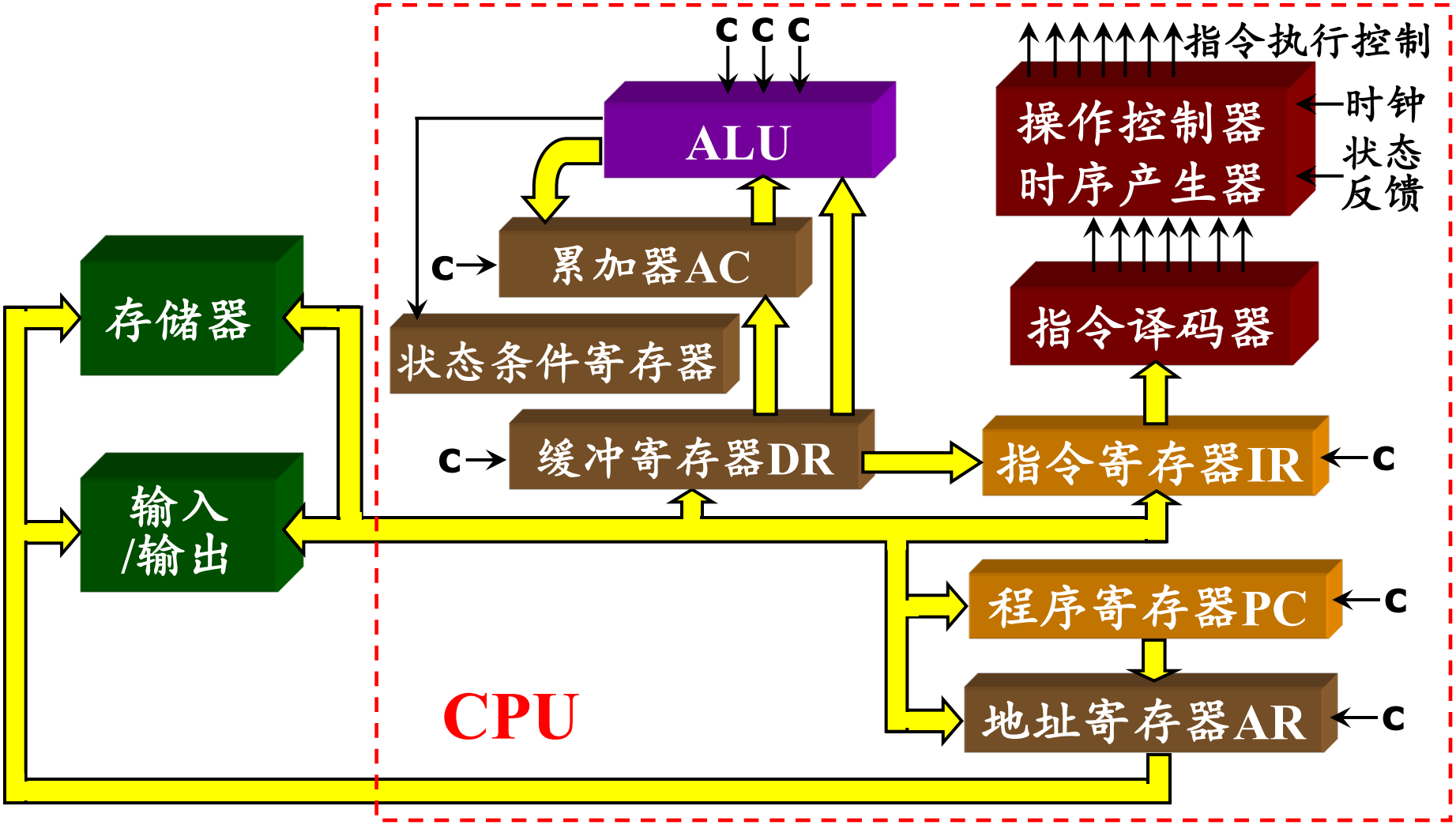
- 中断处理

  对计算机中出现的异常情况和特殊请求进行处理

CPU模型


数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址
数据
5
70
6
100
7
66
10
77
30
40
31
地址
op 码
地址码
101
MOV
R0,R1
102
LAD
R1,6
103
ADD
R1,R2
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
107
ALU
R0
00
R1
10
R2
20
R3
30
DR
ABUS(D)
AR
地址寄存器
②
ABUS(I)
PC
101
+1
④
①
②
产生①-⑤
取指命令
操作控制器(OC)
指令
译码器
C3
⑤
③
IR
MOV
R0,R1
指令寄存器
PSW
状态字寄存器
...
通往 I/O 接口

CPU模型（一般化）


C C C
指令执行控制
ALU
操作控制器
时序产生器
时钟
状态
反馈
C
累加器AC
存储器
状态条件寄存器
指令译码器
C
缓冲寄存器DR
指令寄存器IR
C
输入
/输出
程序寄存器PC
C
CPU
地址寄存器AR
C

CPU的基本组成（1）

- 控制器
  - 由程序计数器、指令寄存器、指令译码器、操作控制器和时序产生器组成
  - 完成指挥和协调整个计算机系统的操作
  - 主要功能
    - 从指令cache中取出一条指令，并指出下一条指令在指令cache中的位置
    - 对指令进行译码或测试，产生相应的操作控制信号，以便启动规定的动作
    - 指挥并控制CPU、内存和输入/输出设备之间数据流动的方向

CPU的基本组成（2）

- 运算器
  - 由算术逻辑单元(ALU)、累加寄存器、数据缓冲寄存器DR和状态条件寄存器PSW组成
  - 是数据加工处理部件。运算器接受控制器的命令而进行动作，它是执行部件
  - 主要功能
    - 执行所有的算术运算
    - 执行所有的逻辑运算，并进行逻辑测试，如零值测试或两个值的比较

CPU中的主要寄存器（1）

- 数据缓冲寄存器（DR）
- 指令寄存器（IR）
- 程序计数器（PC）
- 地址寄存器（AR）
- 通用寄存器（R0~R3）
- 状态条件寄存器（PSW）

CPU中的主要寄存器（2）

- 数据缓冲寄存器（DR）
  - 用来暂时存放从数据cache读出的一个数据字，或来自外部接口的一个数据字，或ALU的运算结果
  - 向数据cache写入一个数据时，也是用DR暂存的
  - 作用
    - 作为ALU运算结果和通用寄存器之间信息传送中时间上的缓冲
    - 补偿CPU和内存、外设之间在操作速度上的差别

CPU中的主要寄存器（3）

- 指令寄存器（IR）
  - 用来保存当前正在执行的一条指令
  - 指令寄存器中操作码字段的输出就是指令译码器的输入
  - 操作码经译码器译码后，即可向操作控制器发出特定的操作信号

CPU中的主要寄存器（4）

- 程序计数器（PC）
  - 也称为指令计数器，用来确定下一条指令的地址
  - 在程序开始执行前，将程序的第一条指令所在的内存单元地址送入PC
  - 当执行指令时，CPU将自动修改PC的内容。PC的值始终是将要执行的下一条指令地址

CPU中的主要寄存器（5）

- 地址寄存器（AR）
  - 用来保存当前处理器所访问的数据cache中单元的地址
  - 地址寄存器的结构与数据缓冲寄存器、指令寄存器一样，使用单纯的寄存器结构

CPU中的主要寄存器（6）

- 通用寄存器（R0~R3）
  - 通用寄存器有4个，R0、R1、R2和R3
  - 功能
    - 当运算器的算术逻辑单元(ALU)执行算术或逻辑运算时，为ALU提供一个暂存数据的工作区
    - 累加寄存器暂时存放ALU运算的结果信息
  - 处理器不同，通用寄存器数目也不相同。数目可从几十个到几百个。

CPU中的主要寄存器（7）

- 状态条件寄存器（PSW）
  - 保存根据算术指令、逻辑指令等运行的结果而设定的各种条件码内容
    - 如运算结果进位标志(C)，运算结果溢出标志(V)，运算结果为零标志(Z)，运算结果为负标志(N)等
    - 每个标志位通常由1位触发器保存
  - 保存中断和系统工作状态等信息
    - 处理器能及时了解机器运行状态和程序运行状态

操作控制器与时序产生器

- 数据通路
  - 各寄存器之间、寄存器与ALU之间传送数据的通路称为数据通路
  - 信息从什么地方开始，中间经过哪个寄存器或多路开关，最后传送到哪个寄存器，都要加以控制
  - 在各寄存器之间建立数据通路的任务，是由称为操作控制器的部件来完成的
- 操作控制器
  - 功能
    - 根据指令操作码和时序信号，产生各种操作控制信号，以便正确地建立数据通路，从而完成取指令和执行指令的控制
  - 根据设计方法不同，操作控制器可分为：
    - 时序逻辑型：采用时序逻辑技术来实现。称为硬布线控制器
    - 存储逻辑型：采用存储逻辑来实现。称为微程序控制器
- 时序产生器：对各种操作信号进行时间上先后顺序的控制

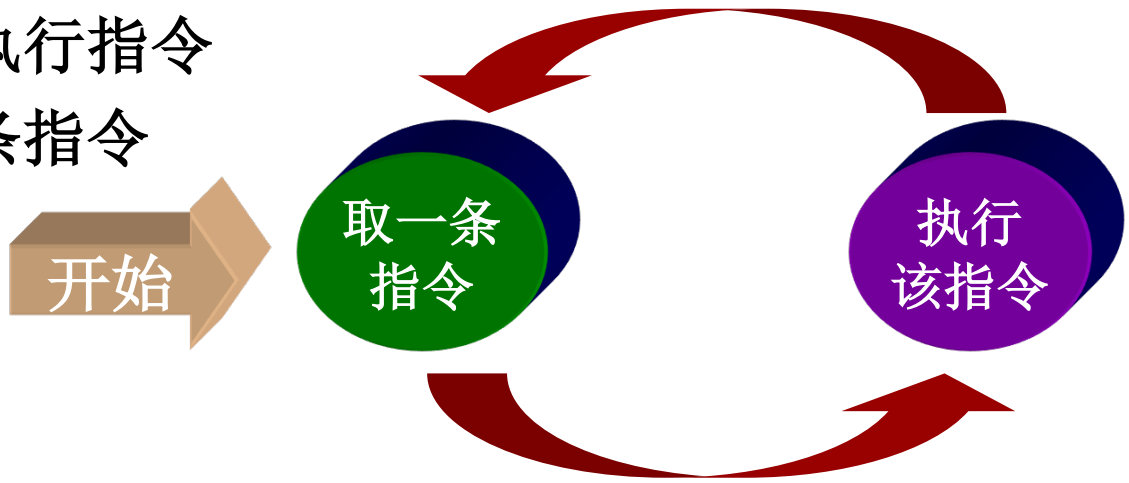
5.2 指令周期

基本概念（1）

- 冯. 诺依曼结构的计算机

  即存储程序式计算机，即在程序运行之前，将程序和数据存放到内存中。

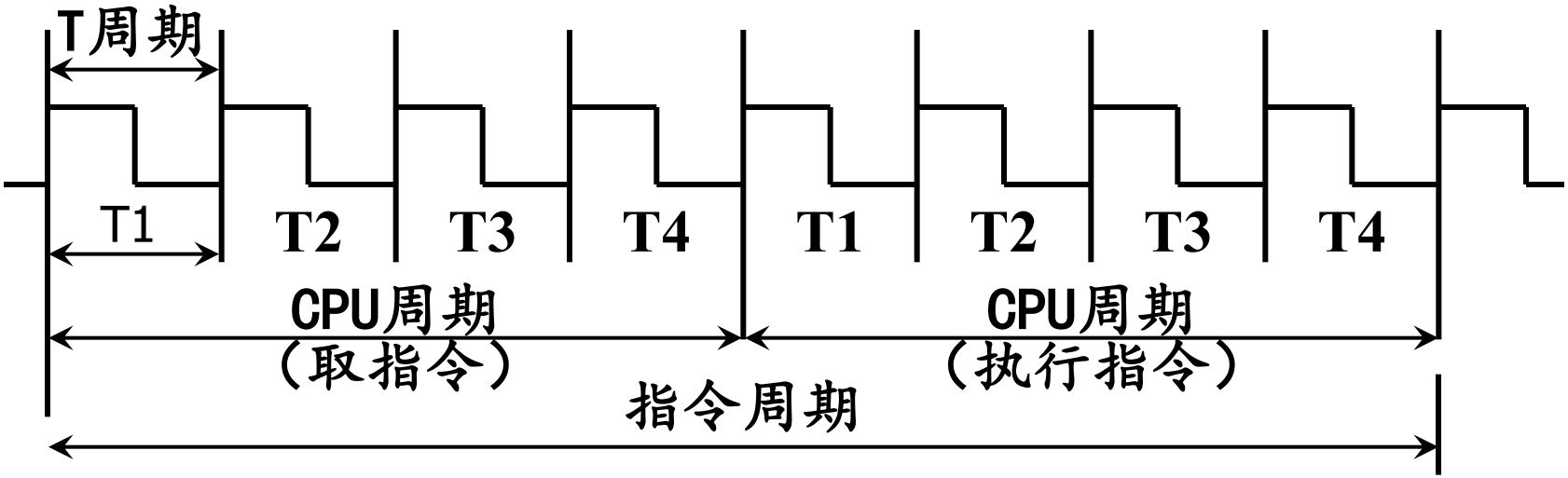
- 执行程序：
  - 将程序首地址送入到程序计数器PC中
  - 从内存（cache）中取出一条指令，正确执行指令
  - 形成下一条待执行指令的地址
  - 正确并自动地连续执行指令
  - 直到程序的最后一条指令



基本概念（2）

- 指令周期
  - 指令周期是取指令、分析指令并执行指令所需的总时间
  - 由于各种指令的操作功能不同，因此各种指令的指令周期长度是不相同的
  - 指令周期通常包含若干个CPU周期
- CPU周期（也称机器周期）
  - 是指内存中读取一个指令字的最短时间
  - CPU周期通常包含若干个时钟周期
- 时钟周期
  - 是处理操作的最基本单位，也称为节拍脉冲或T周期

时钟周期、CPU周期、指令周期



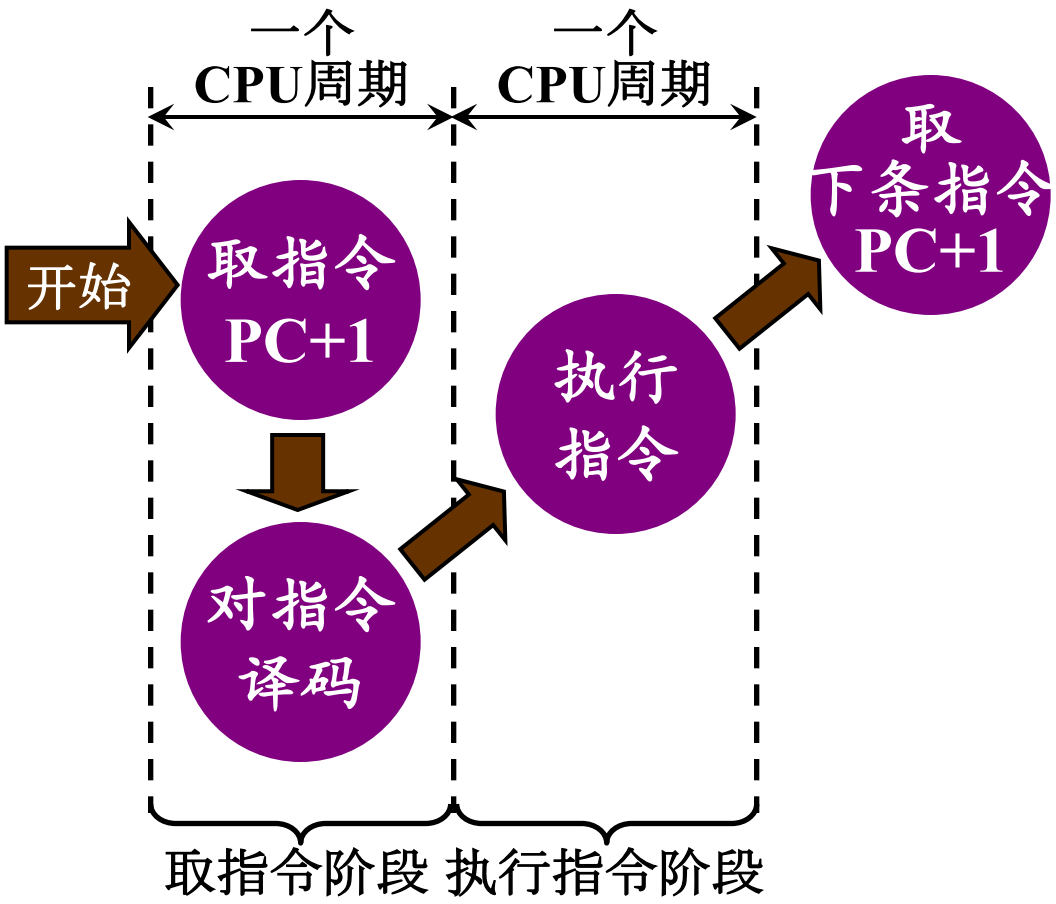
程序示例

- 六条典型指令组成的一个程序

执行前 (R0)=0，(R1)=10，(R2)=20，(R3)=30

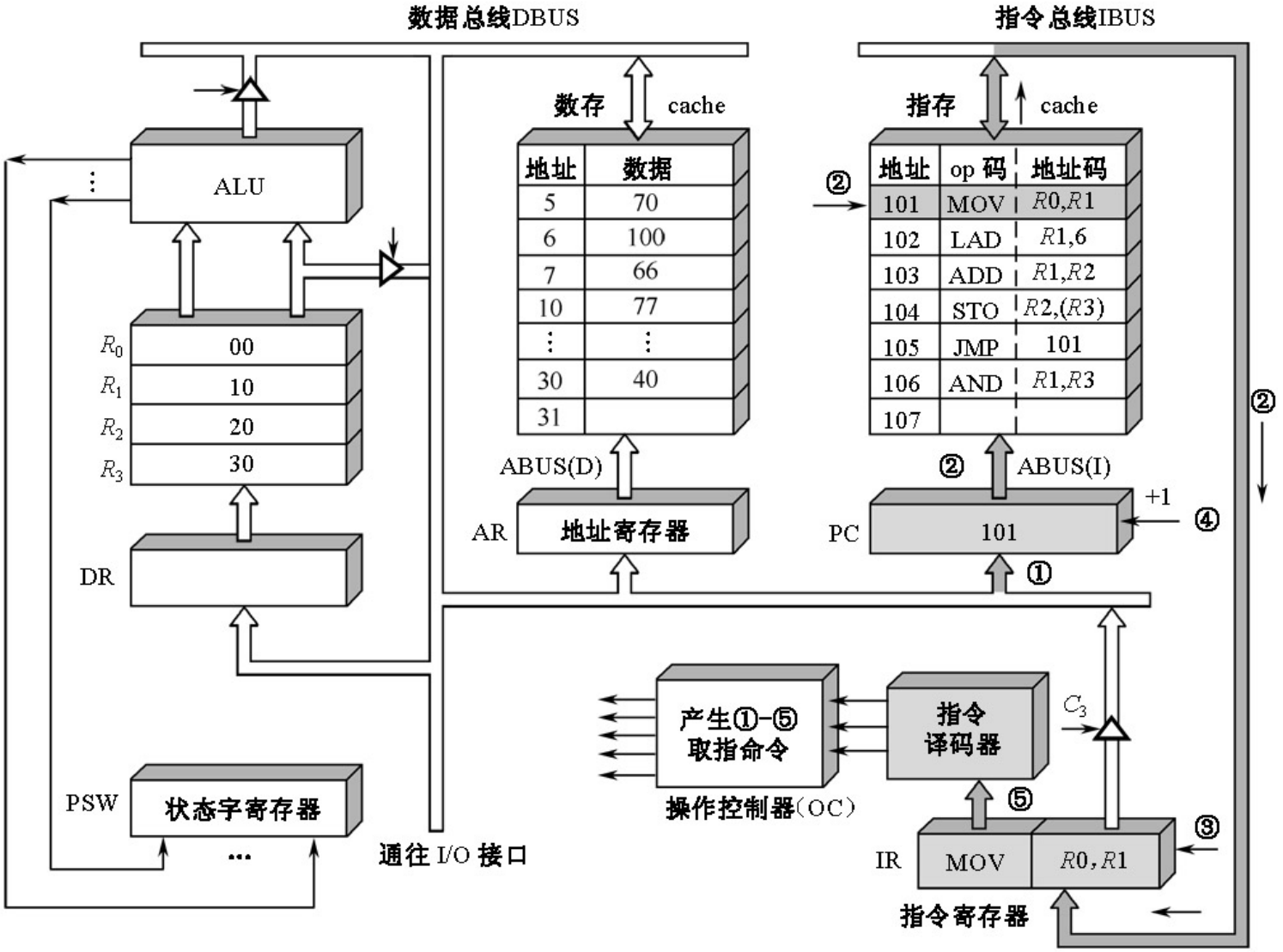
八进制地址	助　记　符	说明
101	MOV R1, R0	(R1)→R0
102	LAD 6, R1	（6）→R1
103	ADD R1, R2	(R1)+(R2) →R2
104	STO R2, (R3)	(R2) →(R3)
105	JMP 101	转跳至101
106	AND R1, R3	(R1)∧(R3) →R3
5	70	数据
6	100	
7	66	
10	77	
30	40（120）	

MOV指令的指令周期

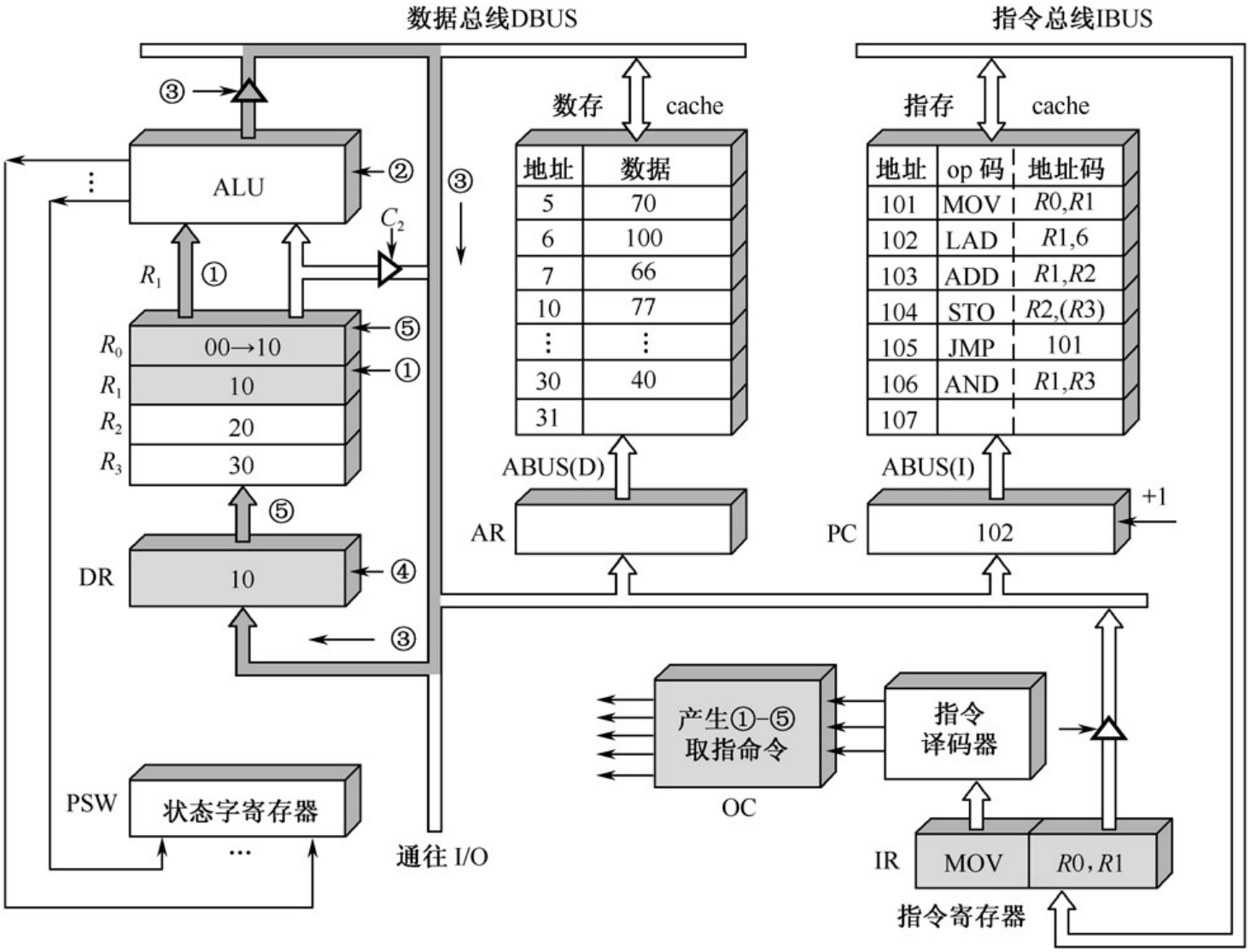


- MOV是一条RR型指令，需要两个CPU周期
- 取指阶段
  - 从内存取出指令
  - PC+1，为取下条指令做好准备
  - 指令译码或测试
- 执行指令阶段
  - 完成指令所要求的操作
  - 即完成通用寄存器R0、R1之间的数据传送操作

MOV指令取指周期

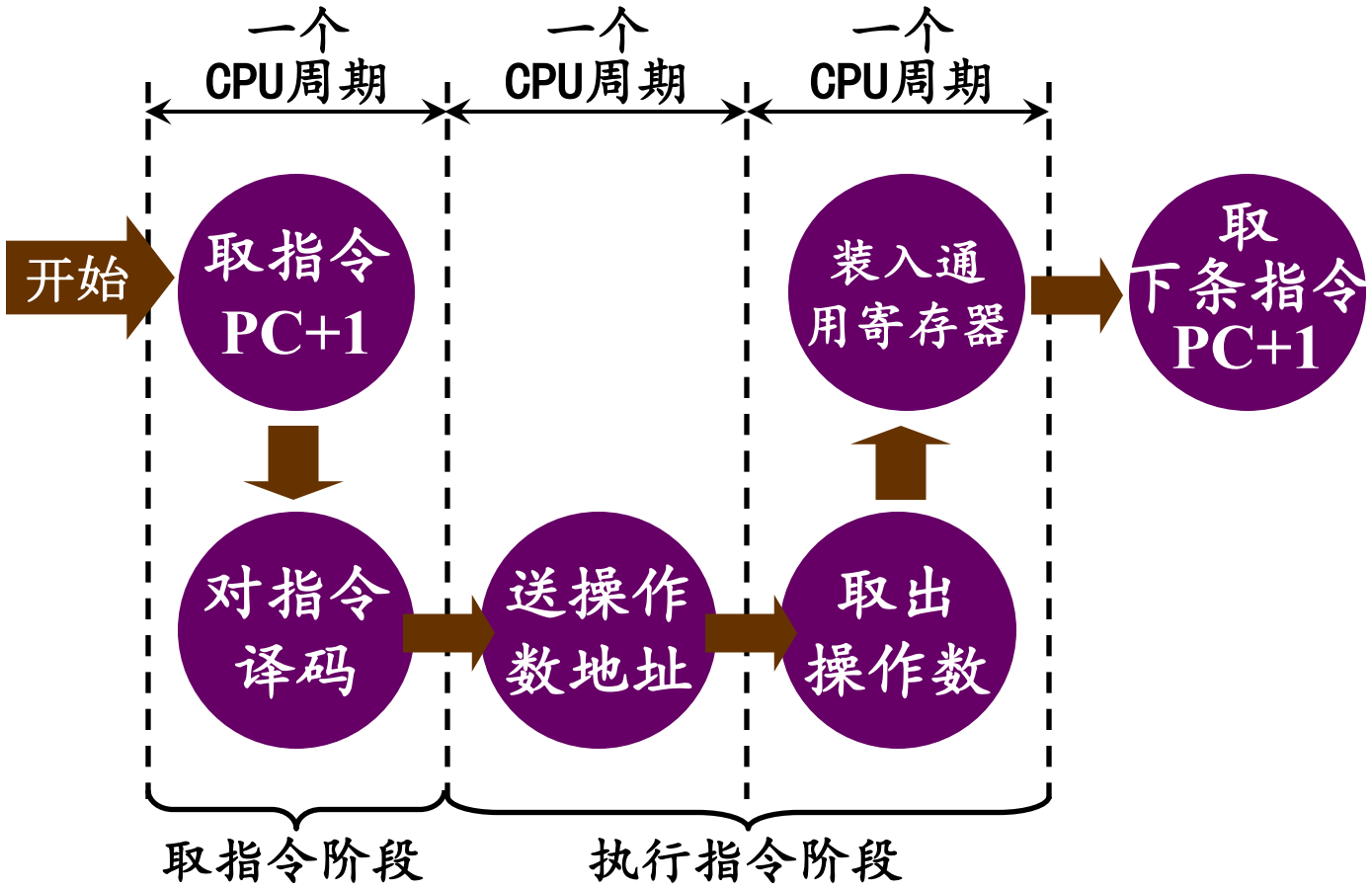

数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址 数据
5 70
6 100
7 66
10 77
30 40
31
地址 op 码 地址码
101 MOV R0,R1
102 LAD R1,6
103 ADD R1,R2
104 STO R2,(R3)
105 JMP 101
106 AND R1,R3
107
ALU
R0 00
R1 10
R2 20
R3 30
DR
ABUS(D)
AR 地址寄存器
②
ABUS(I)
PC 101
+1
④
①
②
产生①-⑤
取指命令
指令
译码器
C3
操作控制器(OC)
⑤
③
IR MOV R0,R1
指令寄存器
PSW 状态字寄存器
...
通往 I/O 接口

MOV指令执行周期


数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址
数据
地址
op 码
地址码
5
70
101
MOV
R0,R1
6
100
102
LAD
R1,6
7
66
103
ADD
R1,R2
10
77
104
STO
R2,(R3)
30
40
105
JMP
101
31
106
AND
R1,R3
107
ALU
C2
R1
R0
00→10
R1
10
R2
20
R3
30
DR
10
ABUS(D)
AR
ABUS(I)
PC
102
+1
PSW
状态字寄存器
通往 I/O
产生①-⑤
取指命令
OC
指令
译码器
IR
MOV
R0，R1
指令寄存器

LAD指令的指令周期（1）

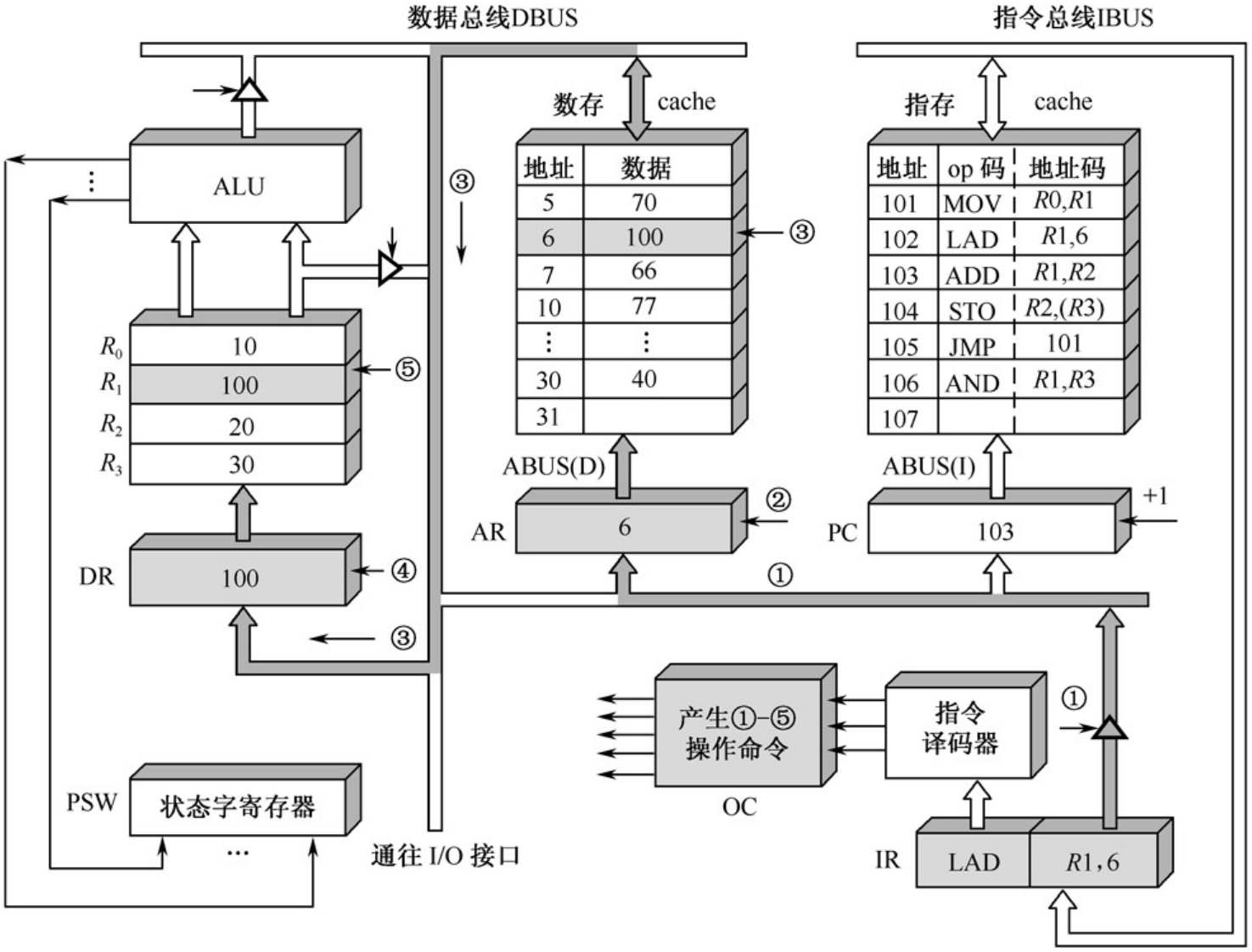
- LAD指令是一条RS型指令
- 由三个CPU周期组成



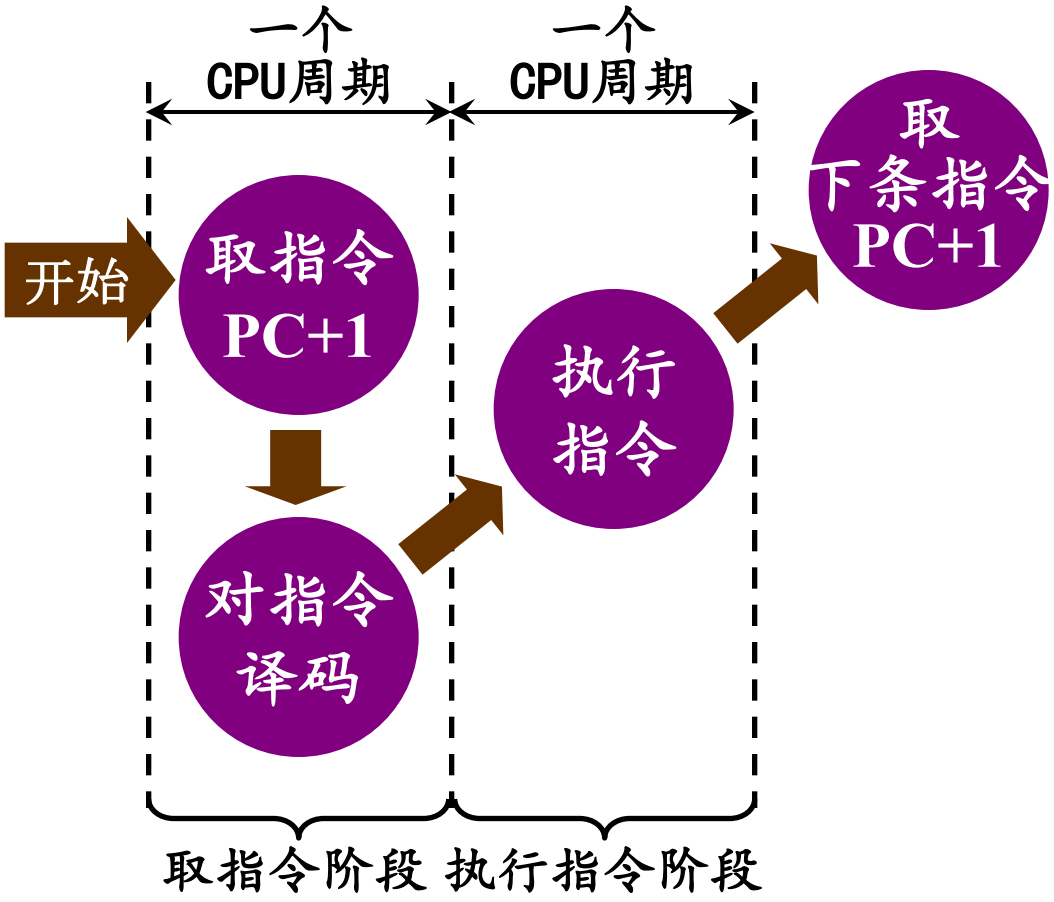
LAD指令的指令周期（2）

- 取指阶段
  - 从内存取出指令
  - PC+1，为取下条指令做好准备
  - 指令译码
- 执行指令阶段
  - 将指令中的直接地址6 装入地址寄存器AR中
  - 发出读命令，将数据存储器读出的数据装入DR中
  - 将DR中的数据装入通用寄存器R1中

LAD指令执行周期

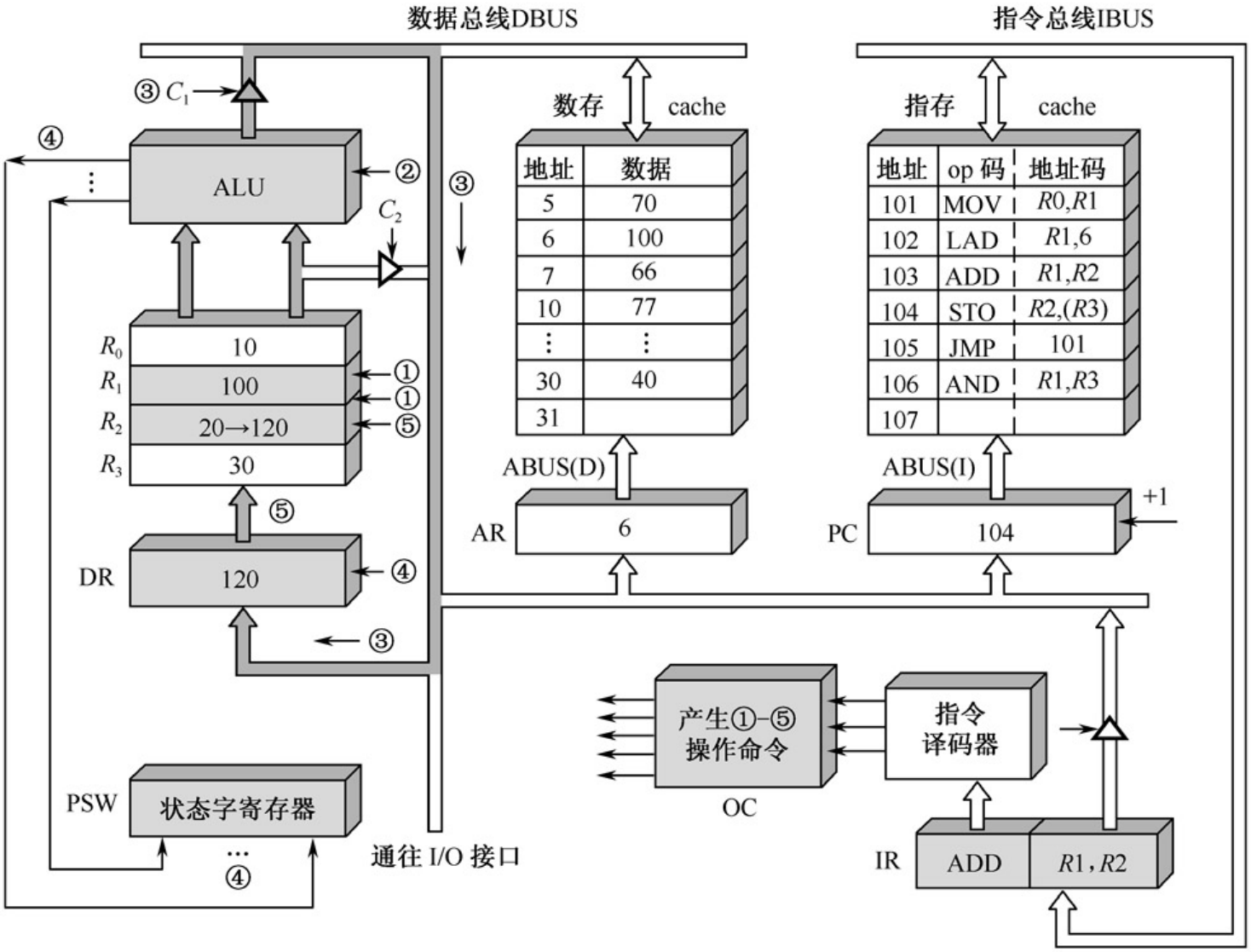

数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址
数据
地址
op 码
地址码
5
70
6
100
7
66
10
77
30
40
31
101
MOV
R0,R1
102
LAD
R1,6
103
ADD
R1,R2
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
107
ALU
R0
10
R1
100
R2
20
R3
30
DR
100
ABUS(D)
AR
6
ABUS(I)
PC
103
+1
①
②
③
④
⑤
产生①–⑤
操作命令
OC
指令
译码器
IR
LAD
R1,6
PSW
状态字寄存器
通往 I/O 接口

ADD指令的指令周期



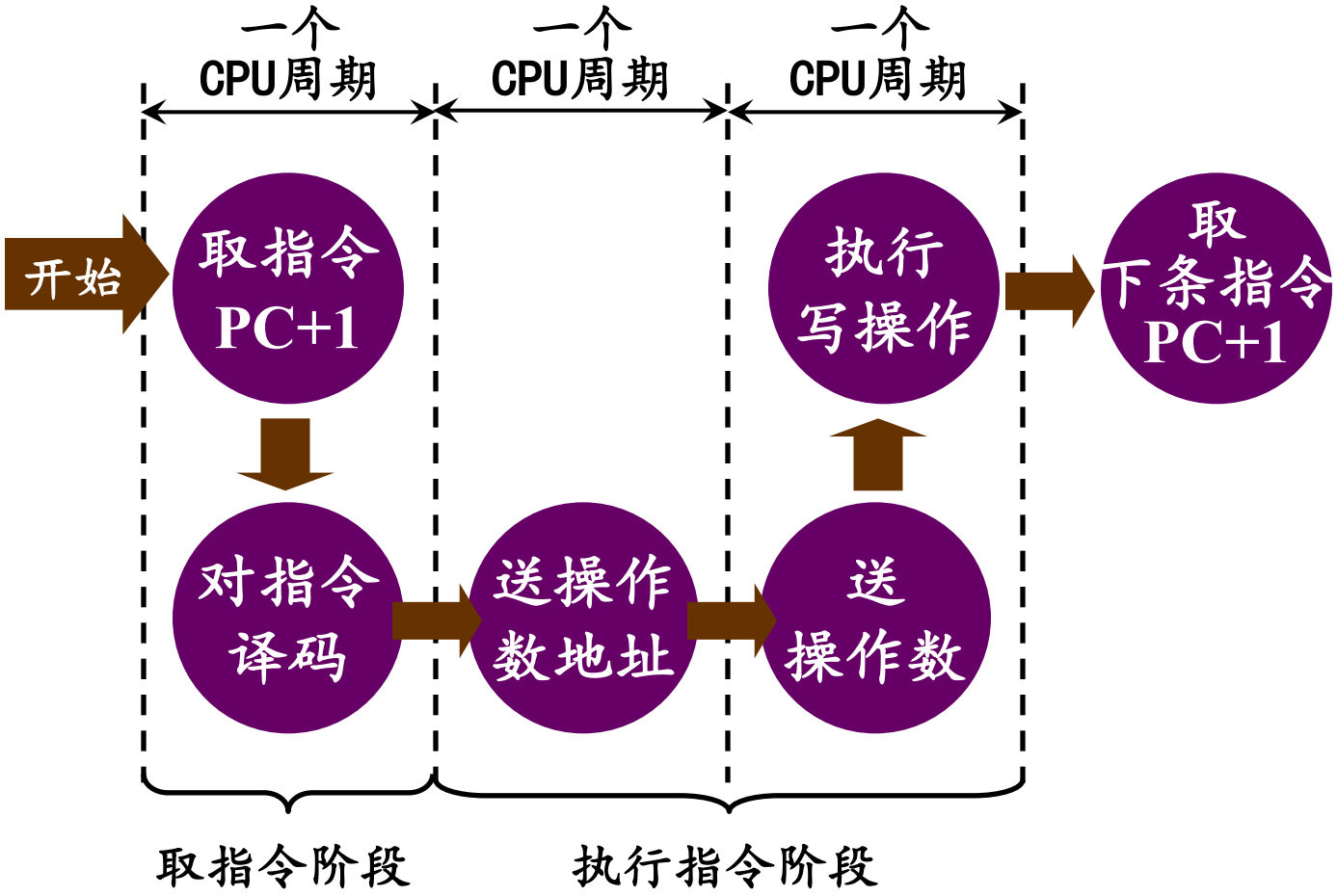
- ADD是一条RR型指令。需要两个CPU周期
- 取指阶段
  - 从内存取出指令
  - PC+1，为取下条指令做好准备
  - 指令译码
- 执行指令阶段
  - 完成指令所要求的操作
  - 即完成通用寄存器R1、R2内容相加，其结果送R2

ADD指令执行周期


数据总线DBUS
指令总线IBUS
③ C_1
④
ALU
②
C_2
③
数存
cache
地址
数据
5
70
6
100
7
66
10
77
30
40
31
指存
cache
地址
op 码
地址码
101
MOV
R0,R1
102
LAD
R1,6
103
ADD
R1,R2
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
107
R_0
10
R_1
100
R_2
20→120
R_3
30
①
①
⑤
⑤
DR
120
④
③
ABUS(D)
AR
6
ABUS(I)
PC
104
+1
产生①–⑤
操作命令
OC
指令
译码器
IR
ADD
R1，R2
PSW
状态字寄存器
…
④
通往 I/O 接口

STO指令的指令周期（1）

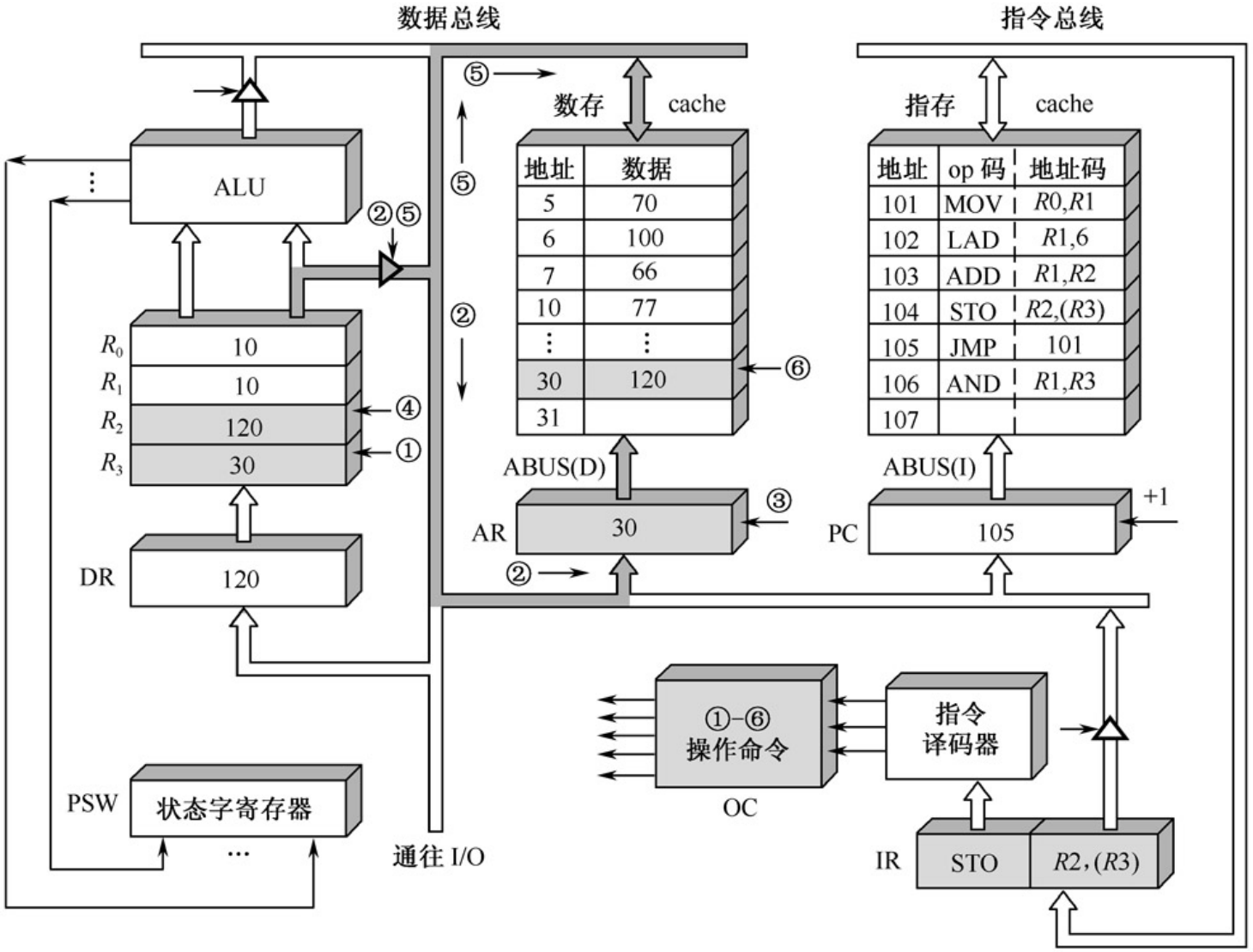
- STO指令是RS型指令，是访内存的存数指令,由三个CPU周期组成。



STO指令的指令周期（2）

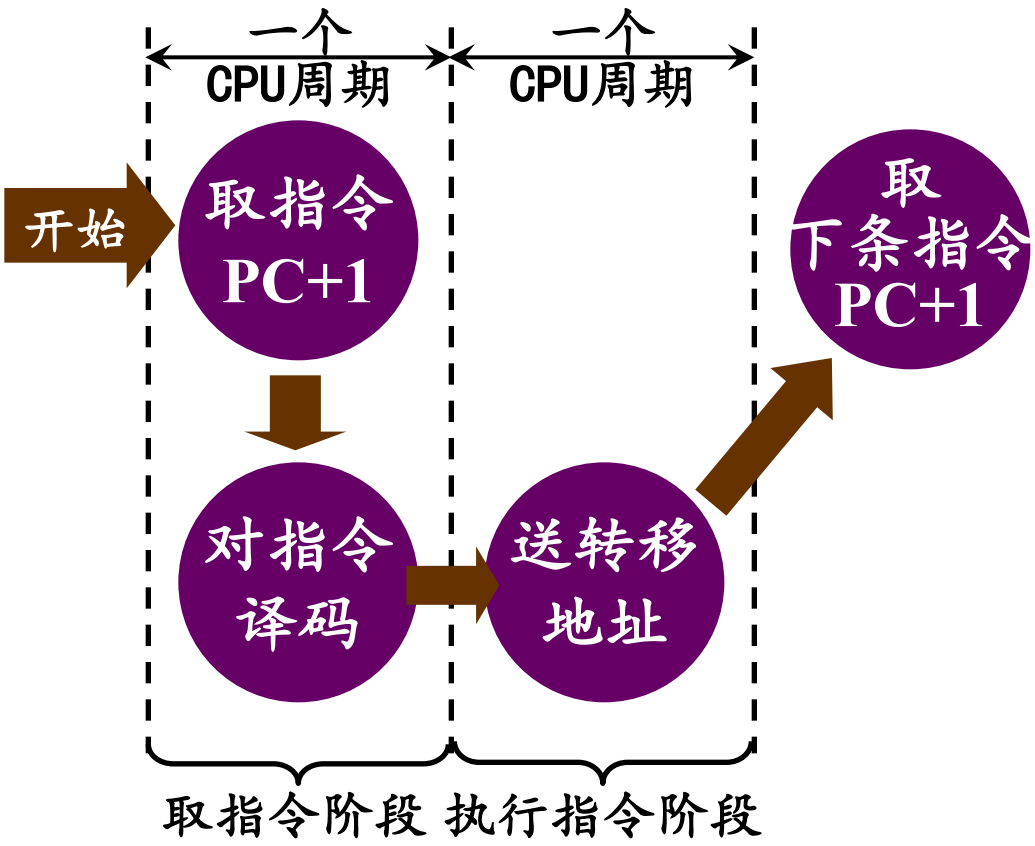
- 取指阶段
  - 从内存取出指令
  - PC+1，为取下条指令做好准备
  - 指令译码
- 执行指令阶段
  - 将通用寄存器R3的内容读出，装入地址寄存器AR中
  - 将通用寄存器R2的内容读出，将数据放到数据总线上
  - 发出写命令，将数据120写入到30号单元中

STO指令执行周期

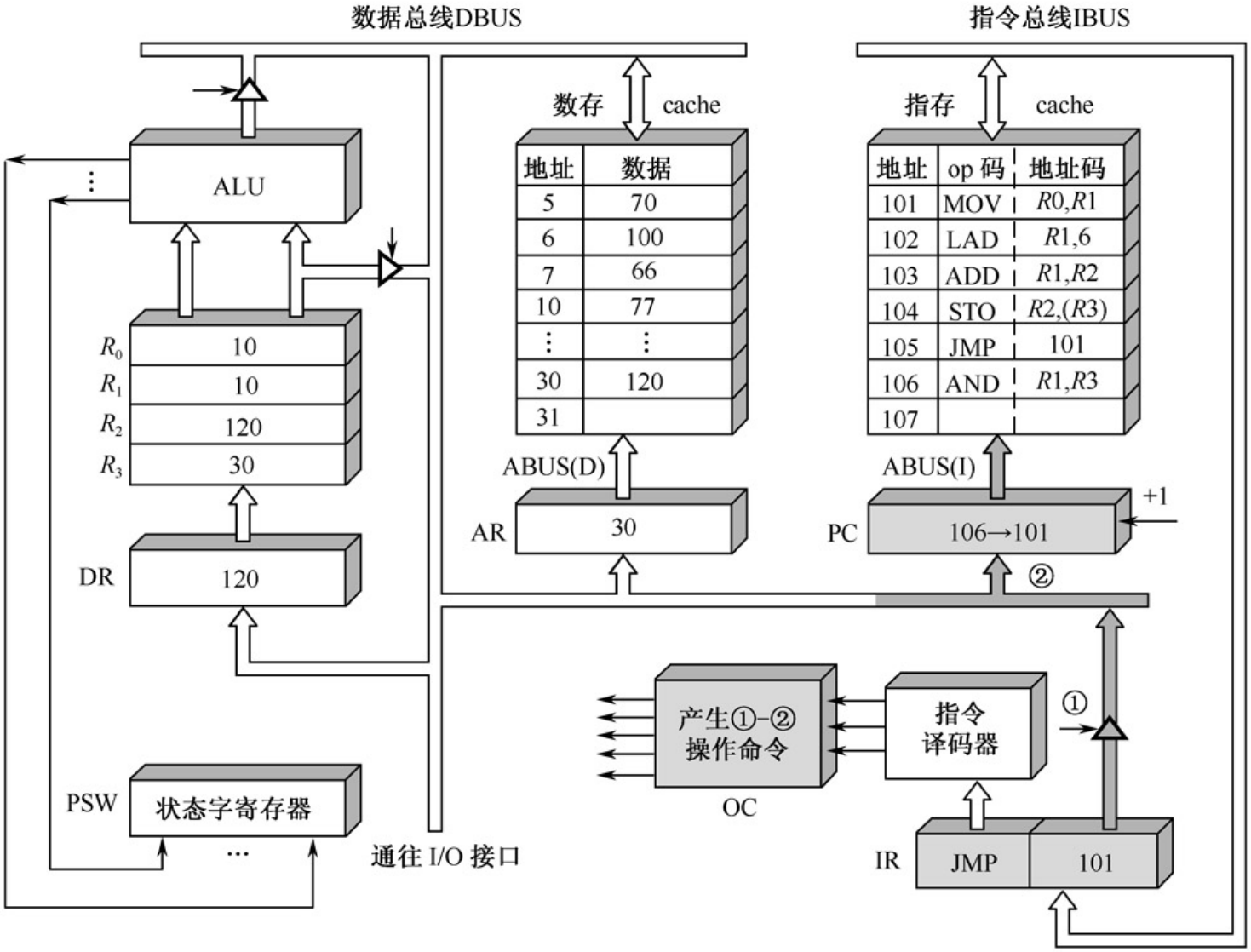

数据总线
指令总线
数存
cache
指存
cache
地址
数据
地址
op 码
地址码
5
70
101
MOV
R0,R1
6
100
102
LAD
R1,6
7
66
103
ADD
R1,R2
10
77
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
30
120
31
107
ALU
R_0
10
R_1
10
R_2
120
R_3
30
DR
120
ABUS(D)
AR
30
ABUS(I)
PC
105
+1
①-⑥
操作命令
OC
指令
译码器
IR
STO
R2,(R3)
PSW
状态字寄存器
通往 I/O

JMP指令的指令周期

- JMP指令是一条无条件转移指令
- 由两个CPU周期组成



JMP指令执行周期


数据总线DBUS
指令总线IBUS
数存
cache
指存
cache
地址
数据
5
70
6
100
7
66
10
77
30
120
31
地址
op 码
地址码
101
MOV
R0,R1
102
LAD
R1,6
103
ADD
R1,R2
104
STO
R2,(R3)
105
JMP
101
106
AND
R1,R3
107
ALU
R0
10
R1
10
R2
120
R3
30
DR
120
ABUS(D)
AR
30
ABUS(I)
PC
106→101
+1
②
①
产生①-②
操作命令
OC
指令
译码器
IR
JMP
101
PSW
状态字寄存器
通往 I/O 接口

总结：每条指令的执行步骤

1）读取指令

是一次读内存操作

指令地址送入主存地址寄存器

读主存，读出内容送入指令寄存器

} 公共操作

2）分析指令

3）按指令规定内容执行指令

R_R类型指令
读写内存类型指令
输入输出类型指令
其他类型指令

不同指令的操作步骤数和具体操作内容差异很大，是每一条指令的特定操作

可能执行一次或多次

4）检查有无中断请求

若有，则响应中断并转中断处理

若无，则转入下一条指令的执行过程

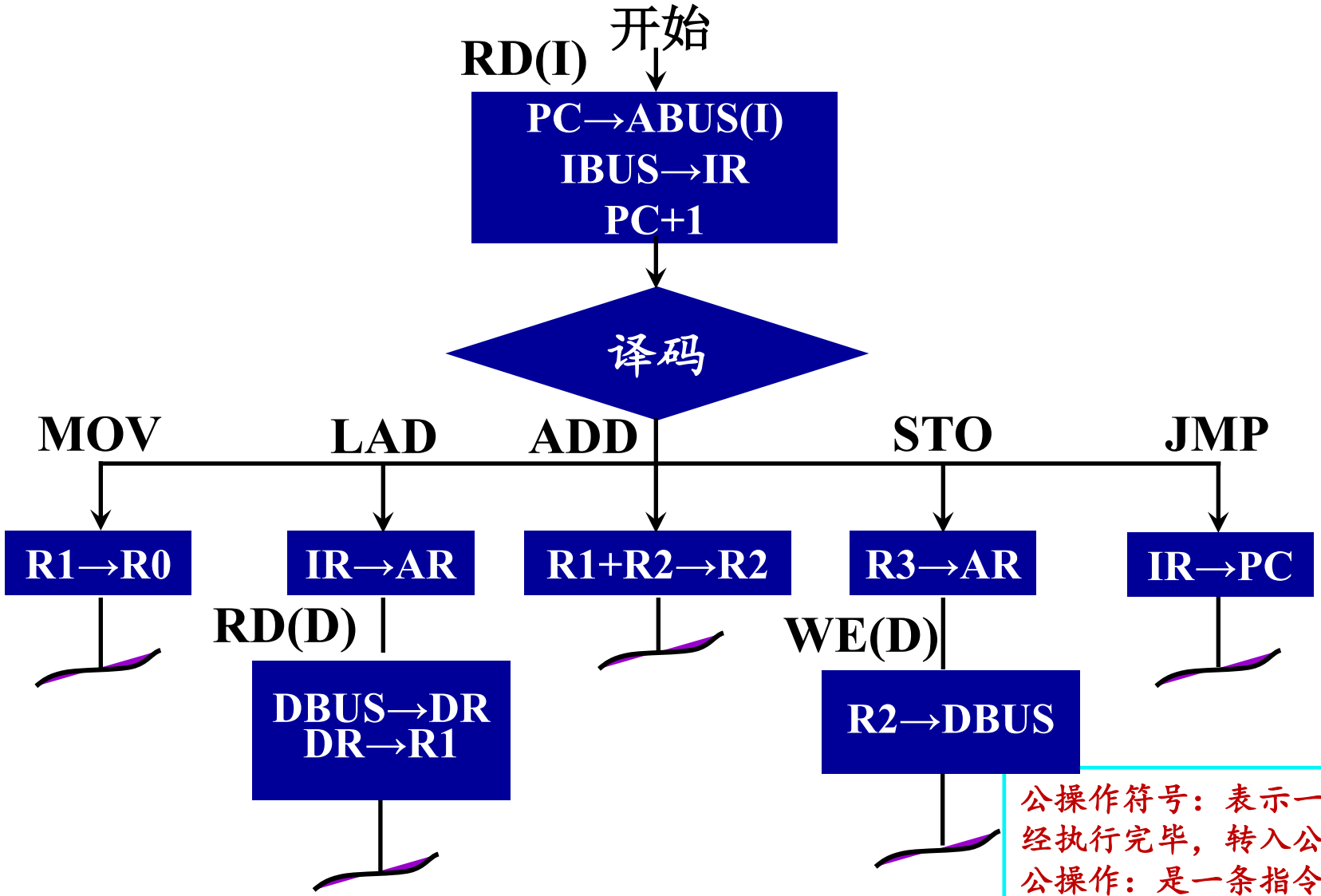
} 公共操作

形成下一条指令地址

方框图语言

- 在进行中央处理器设计时，可用方框图语言来描述一条指令的指令周期
  - 方框：代表一个CPU周期，方框中的内容表示数据通路的操作或某种控制操作
  - 菱形：通常用来表示某种判别或测试，在时间上依附于紧接它的前面一个方框的CPU周期，不单独占用一个CPU周期

五条典型指令的指令周期



公操作符号：表示一条指令已经执行完毕，转入公操作。
公操作：是一条指令执行完毕后，CPU所进行的公共操作，如中断处理等。

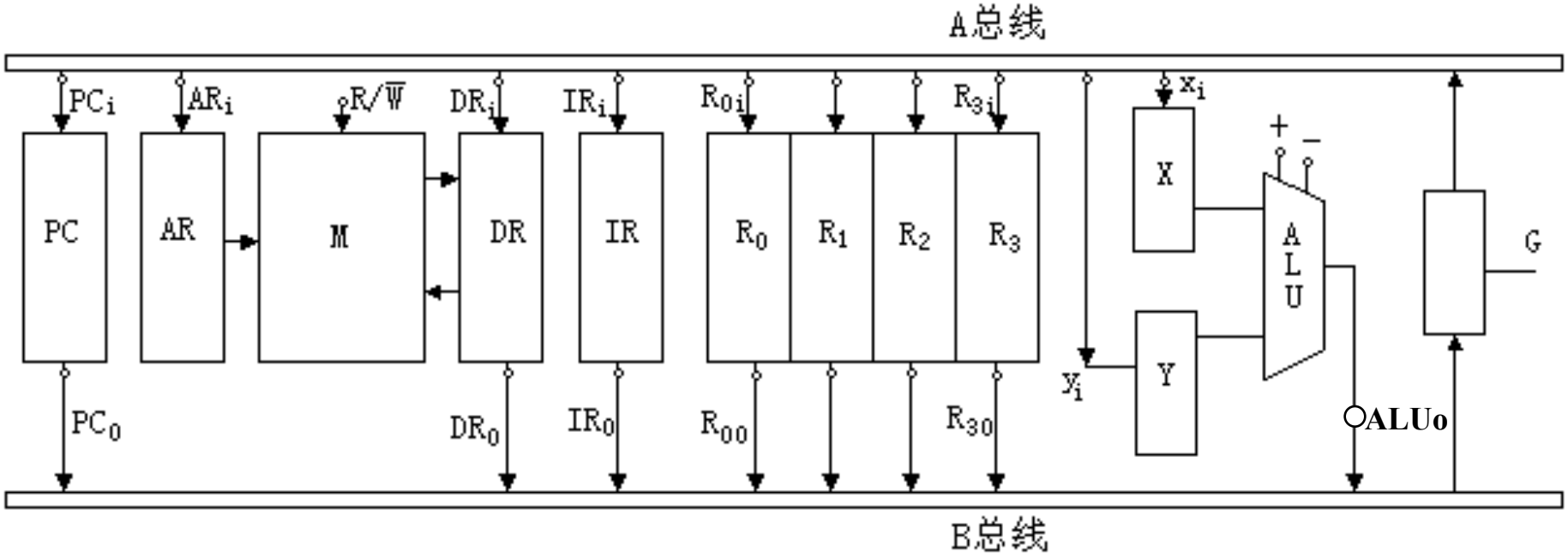
例1

双总线结构机器的数据通路如图所示，IR为指令寄存器，PC为程序计数器（具有自增功能），M为主存（受R/W#信号控制），AR为地址寄存器，DR为数据缓冲寄存器， ALU由加、减控制信号决定完成何种操作，控制信号G控制的是一个门电路。另外，线上标注的小圈表示有控制信号，yi表示y寄存器的输入控制信号，R1o为R1的输出控制信号，未标小圈的线为直通线，不受控制。

(1)“ADD R2，R0”指令完成(R0)+(R2)→R0的功能，画出其指令周期流程图，假设该指令的地址已放入PC中。列出相应的微操作控制信号序列。

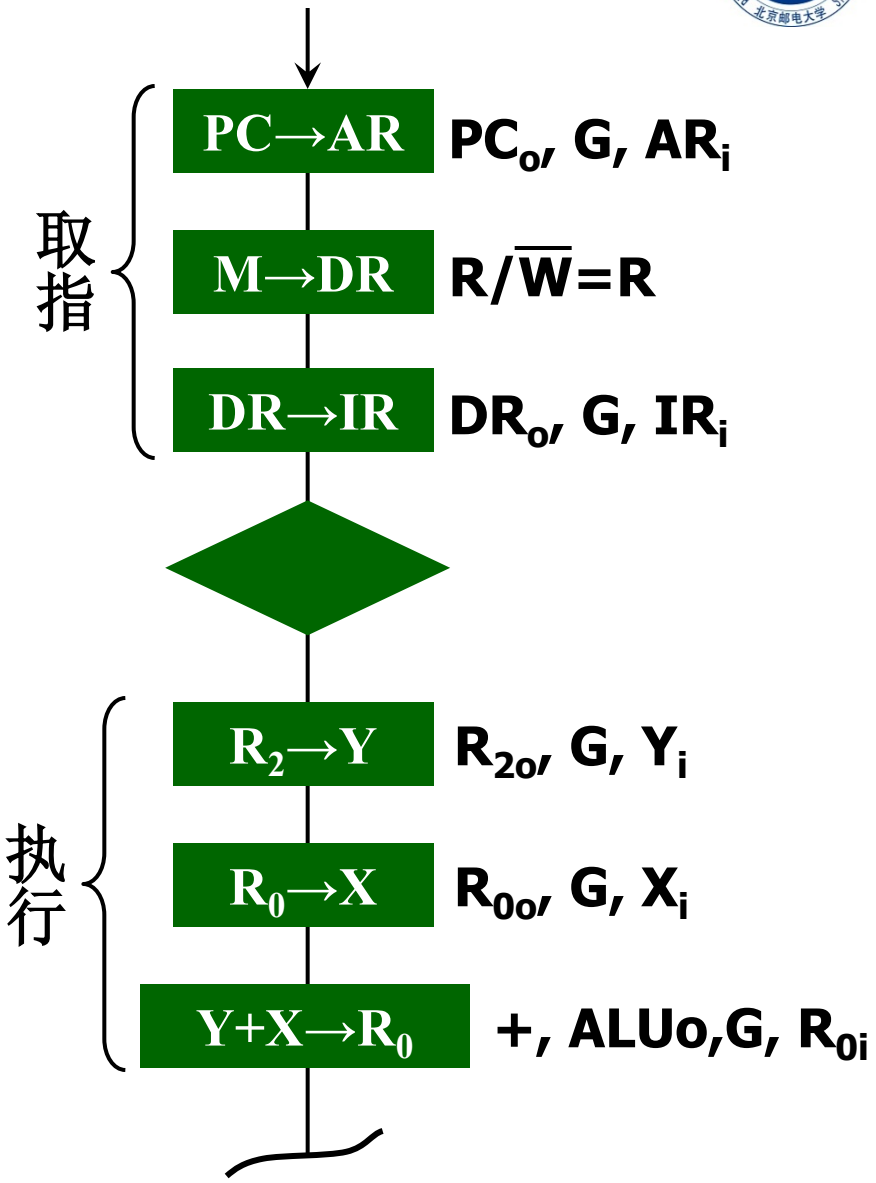
(2)“SUB R1，R3”指令完成(R3)-(R1)→R3的功能，画出其指令期流程图，列出相应的微操作控制信号序列。

例1图


A总线
PCi
ARi
R/W
DRi
IRi
R0i
R3i
xi
PC
AR
M
DR
IR
R0
R1
R2
R3
X
Y
ALU
G
yi
PC0
DR0
IR0
R00
R30
ALUo
B总线

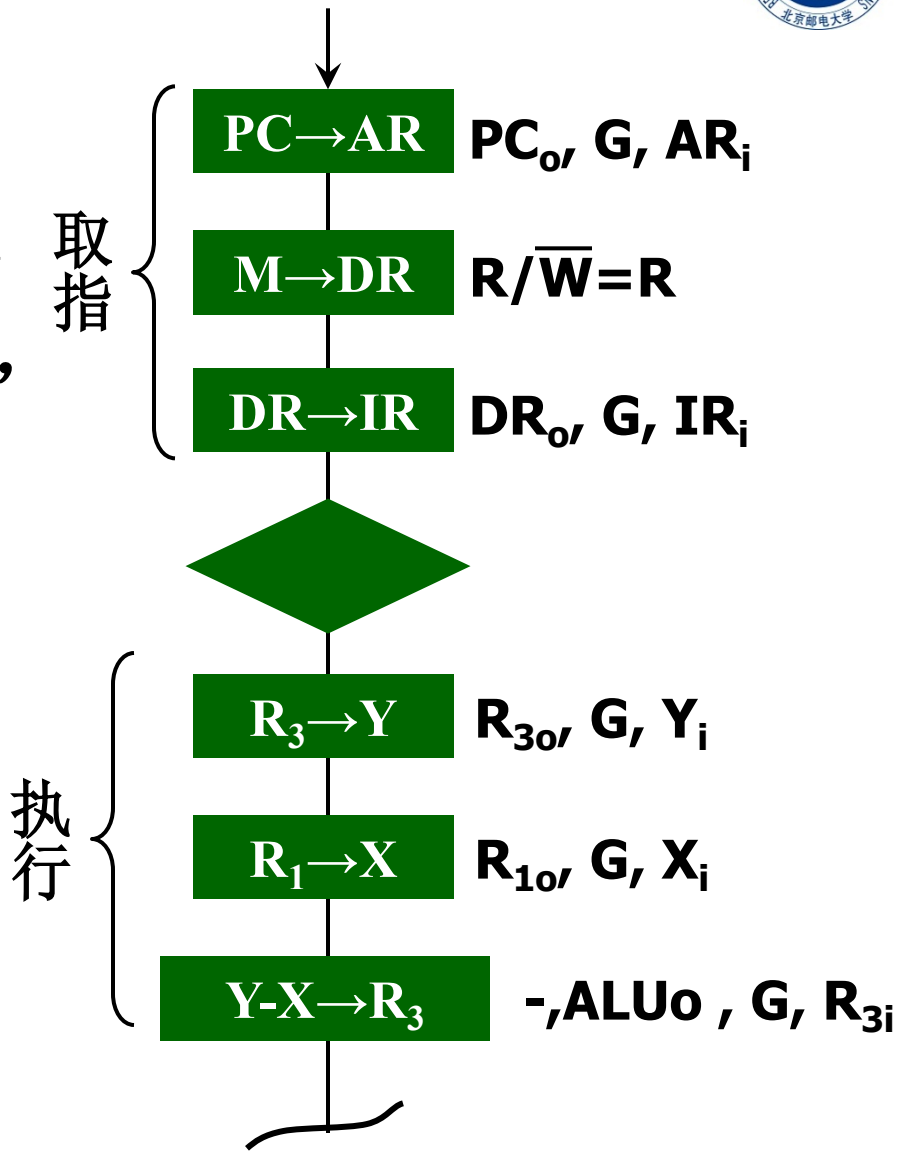
例1解（1）

(1)“ADD R2，R0”指令是一条加法指令，参与运算的两个数放在寄存器R2和R0中，指令周期流程图包括取指令阶段和执行指令阶段两部分。根据给定的数据通路图，“ADD R2，R0”指令的详细指令周期流程图如右图所示，图的右边部分标注了每一个机器周期中用到的微操作控制信号序列。



例1解（2）

(2)“SUB R1，R3”指令是一条减法指令，其指令周期流程图如右图所示。与ADD指令不同的是：在执行指令阶段，微操作控制信号序列有所不同。



5.3 时序产生器和控制方式

基本概念

- 时序（Timing）
- 在一个CPU周期中，时间可分为若干个小段，在每一个小段，CPU完成一个特定的操作
- 计算机以时序信号为基准完成协调动作

控制器分类：

- 硬布线控制器
  - 时序信号一般采用主状态周期-节拍电位-节拍脉冲三级体制。节拍电位的时间=CPU周期的时间
- 微程序控制器
  - 时序信号比较简单。一般采用节拍电位-节拍脉冲二级体制

问题？

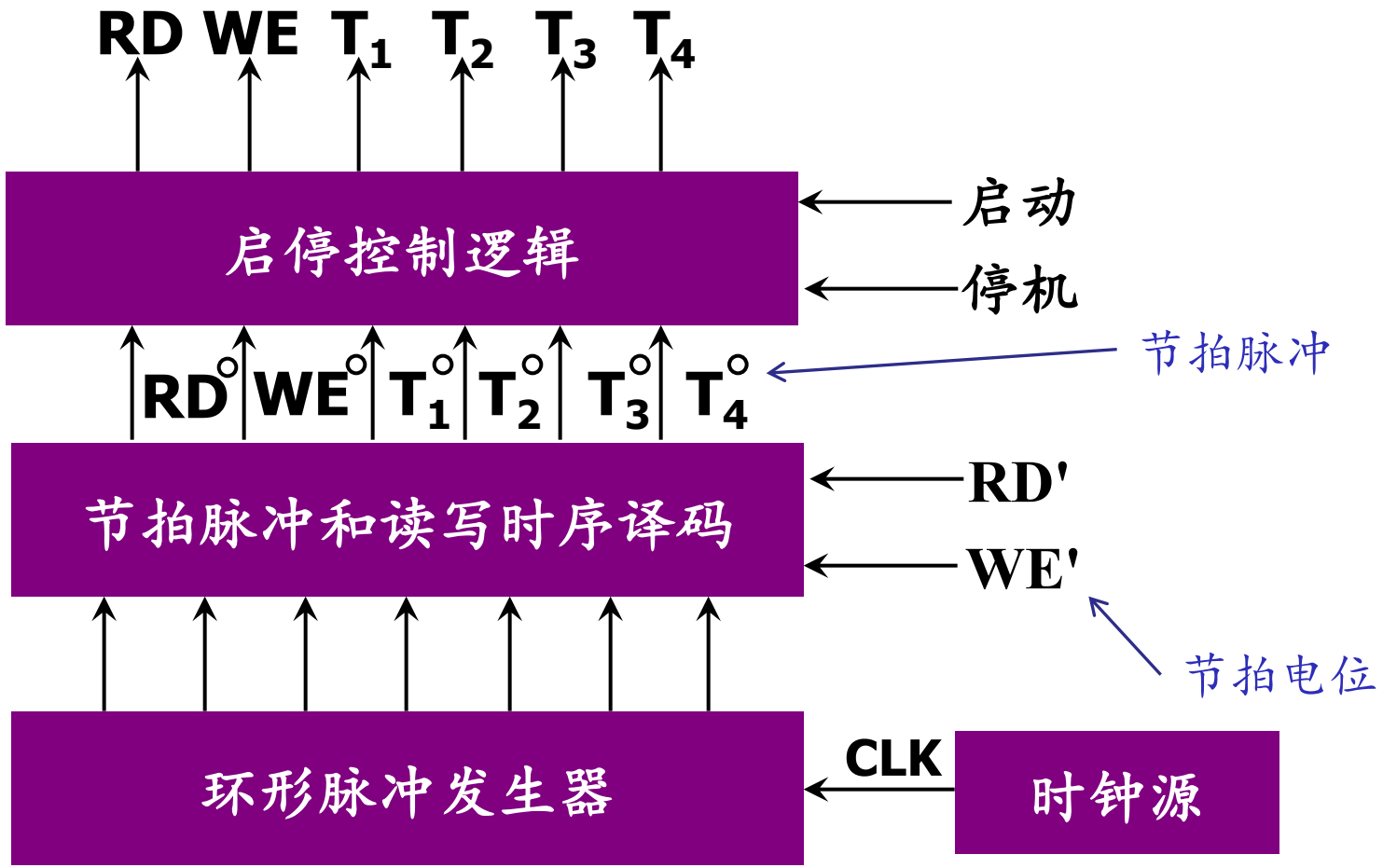
- 用二进制码表示的指令和数据都放在内存里，那么CPU是怎样识别出它们是数据还是指令呢?
  - 从时间上来说，取指令事件发生在指令周期的第一个CPU周期中，即发生在“取指令”阶段，而取数据事件发生在“执行指令”阶段
  - 从空间上来说，如果取出的代码是指令，那么一定送往指令寄存器，如果取出的代码是数据，那么一定送往运算器，由此可见，时间控制对计算机来说是太重要了

解释不太准确！

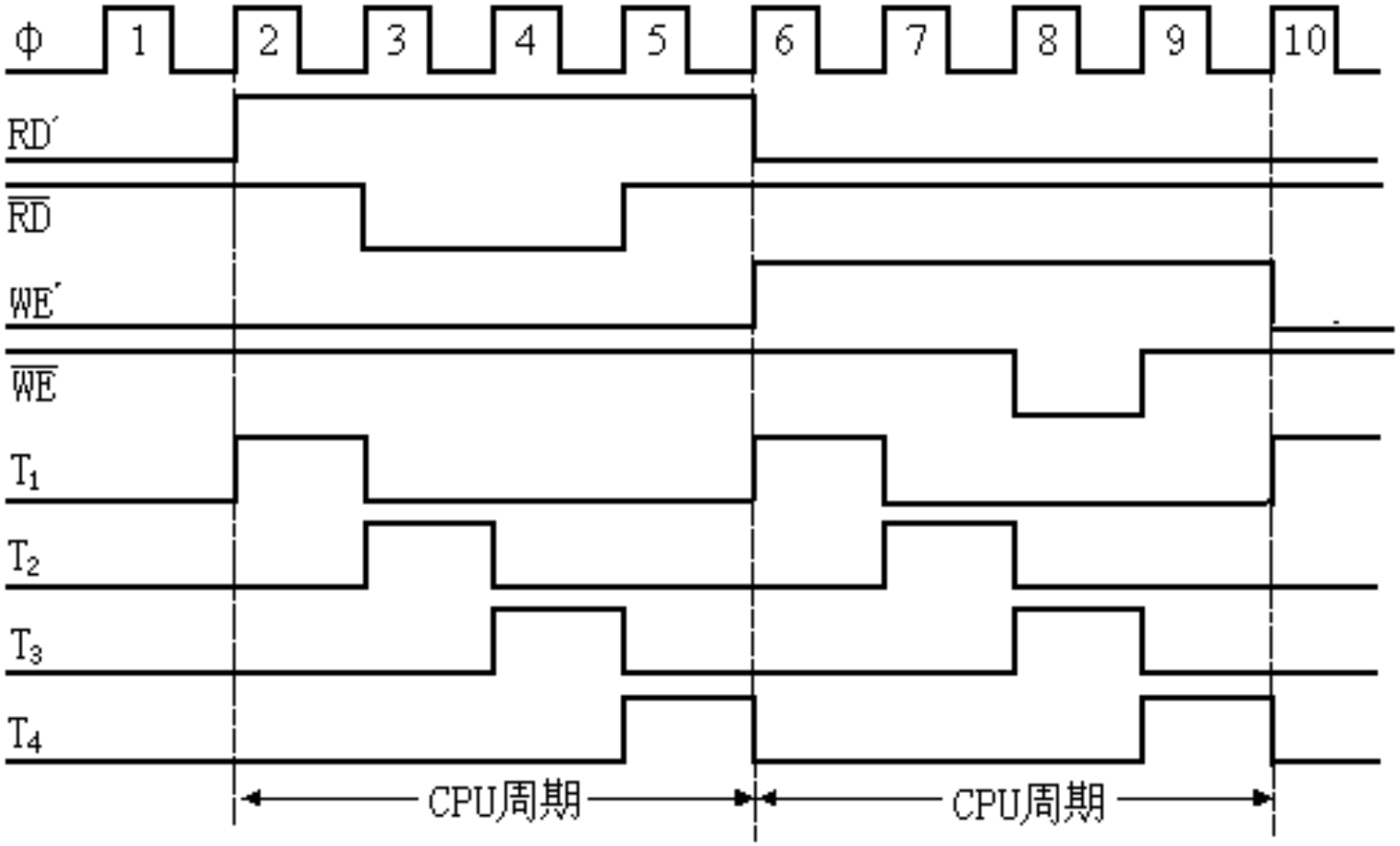
时序信号产生器

- 微程序控制器中时序信号产生器的组成
  - 时钟源
  - 环形脉冲发生器
  - 节拍脉冲和读/写时序的译码
  - 启停控制逻辑

时序信号产生器框图


RD WE T1 T2 T3 T4
启停控制逻辑
启动
停机
RD° WE° T1° T2° T3° T4°
节拍脉冲
节拍脉冲和读写时序译码
RD'
WE'
节拍电位
环形脉冲发生器
CLK
时钟源

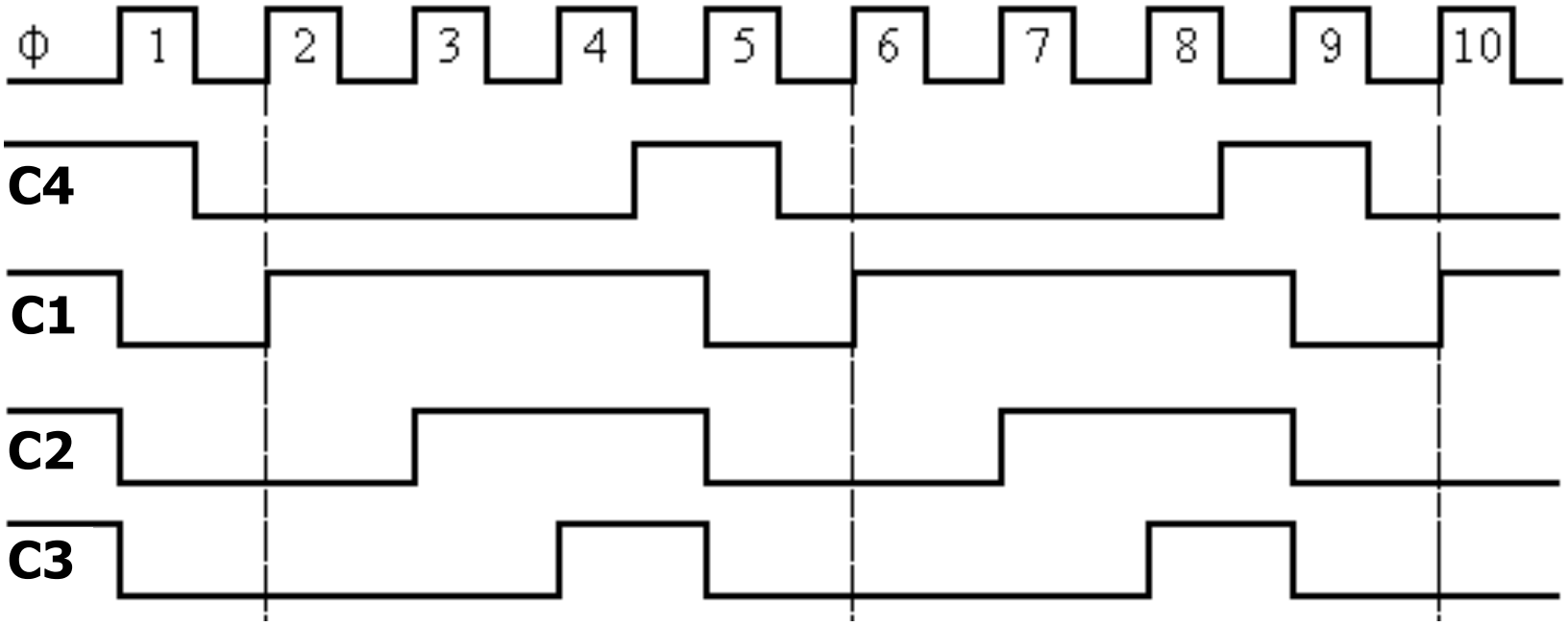
时序信号产生器的输出


Φ
1
2
3
4
5
6
7
8
9
10
RD′
$\overline{RD}$
WE′
$\overline{WE}$
T_1
T_2
T_3
T_4
CPU周期
CPU周期

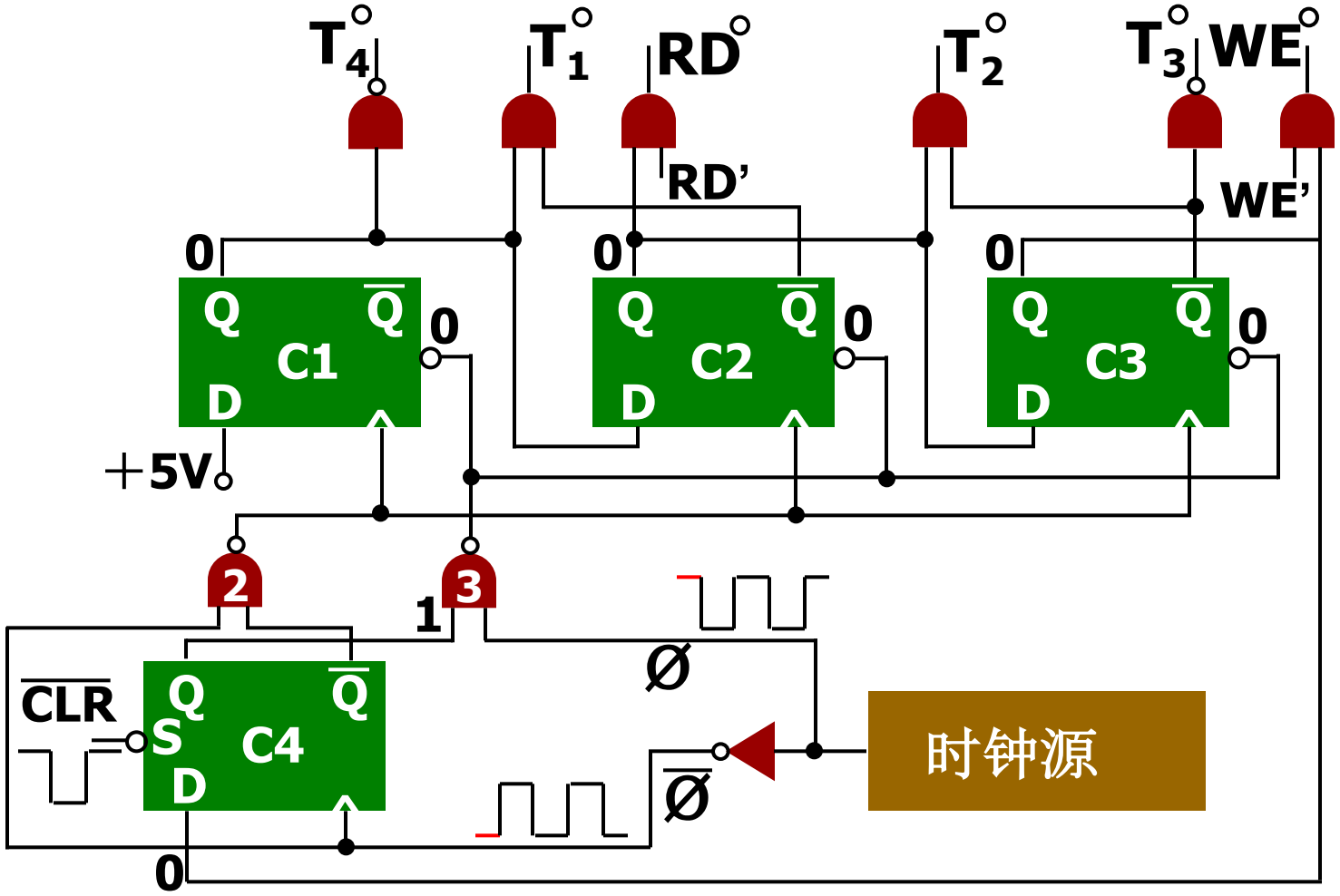
时序信号产生器（1）

- 环形脉冲发生器
  - 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
  - 为了在节拍脉冲上不带干扰毛刺，环形脉冲发生器通常采用循环移位寄存器形式

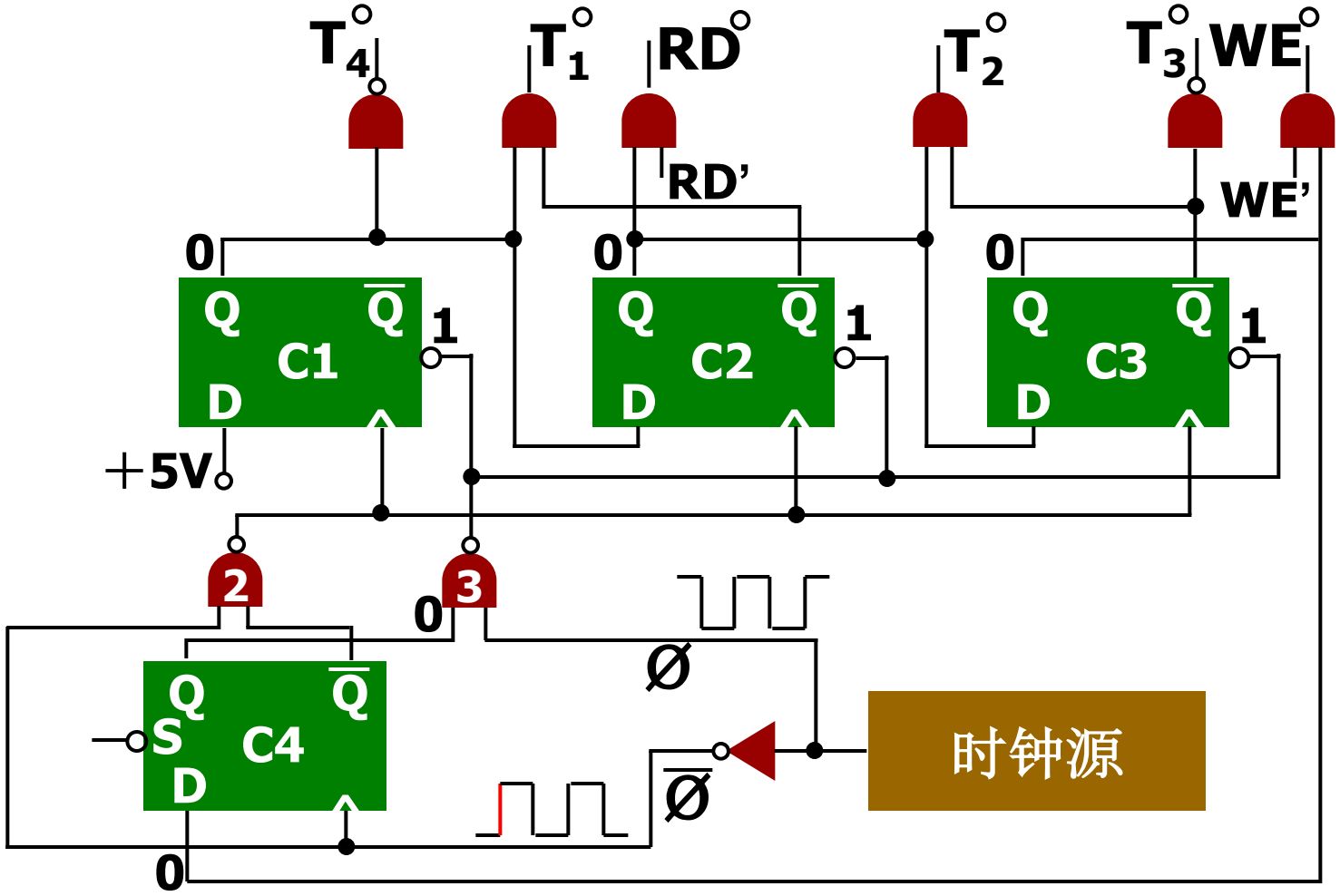
节拍电位与节拍脉冲时序关系图1



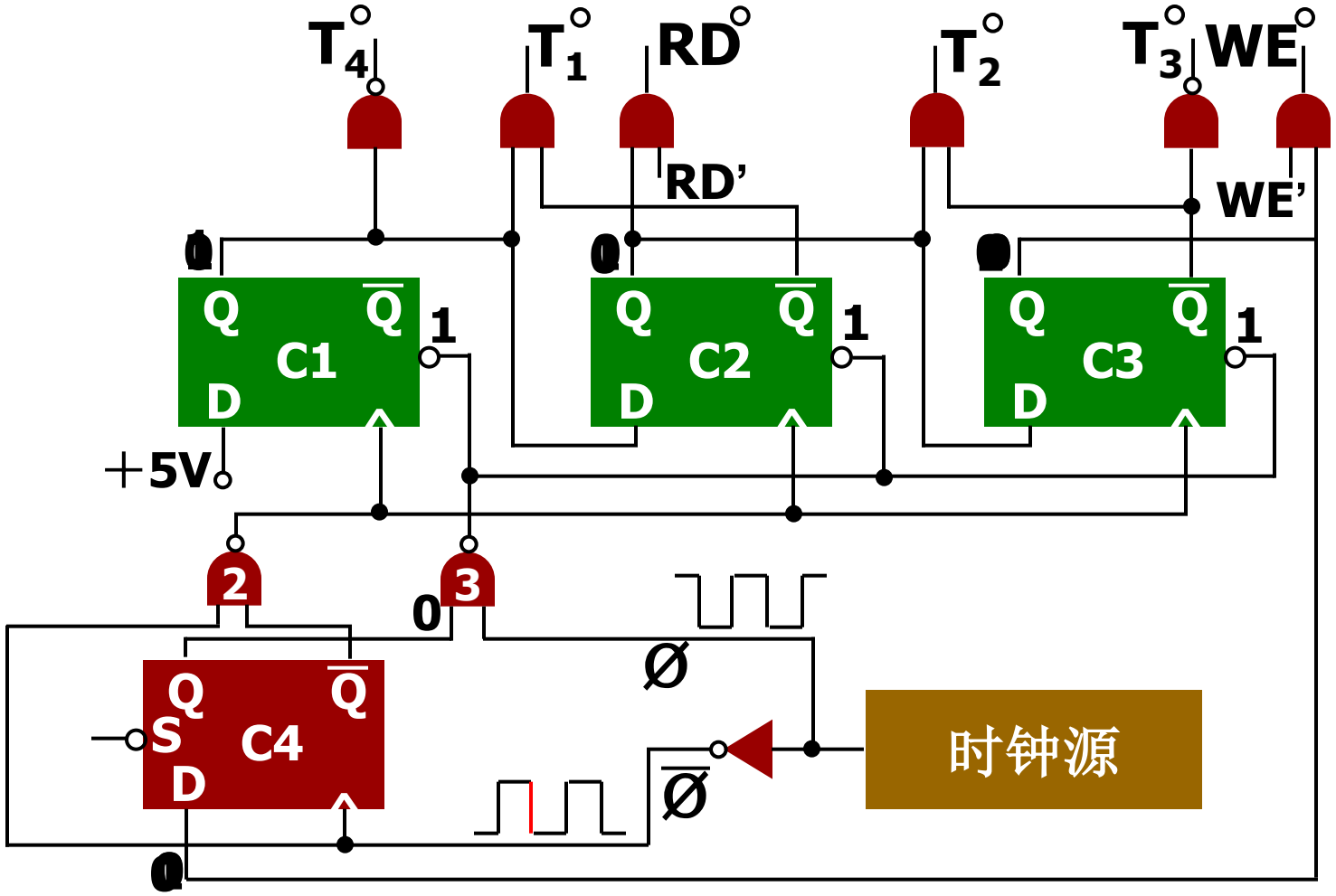
环形脉冲发生器与译码逻辑（1）



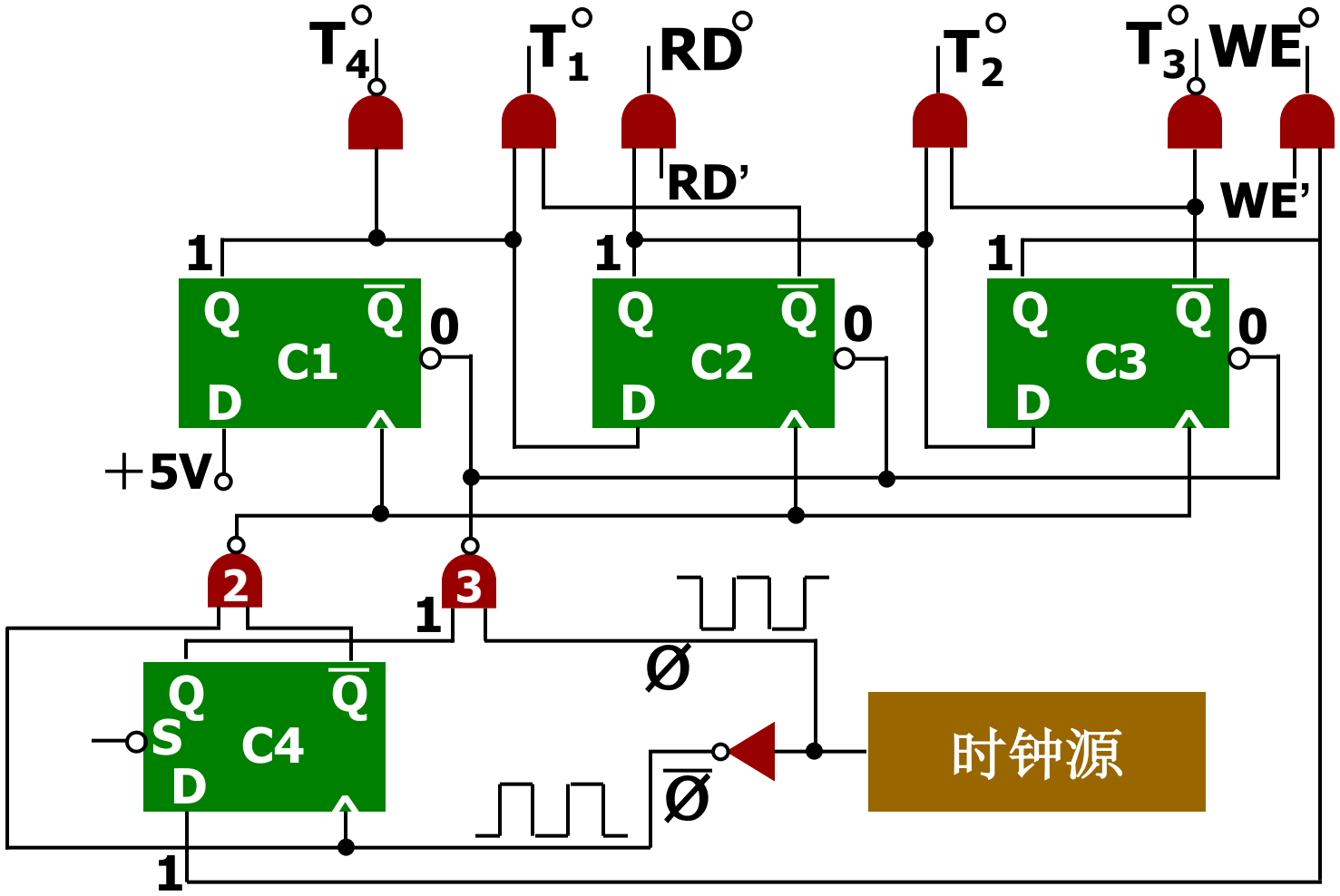
环形脉冲发生器与译码逻辑（2）



环形脉冲发生器与译码逻辑（3）

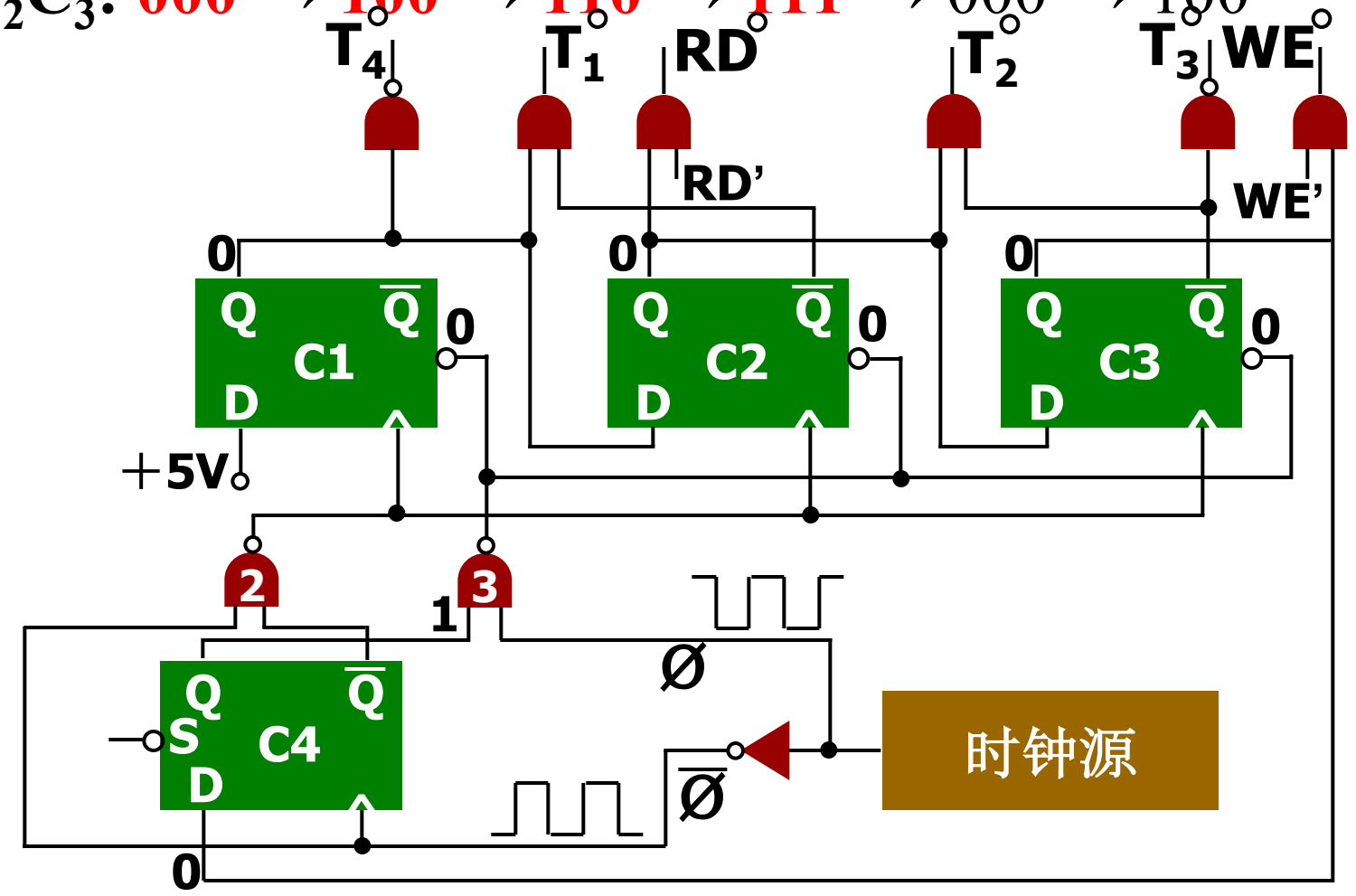


环形脉冲发生器与译码逻辑（4）



环形脉冲发生器与译码逻辑（5）

$C_1C_2C_3$: 000 → 100 → 110 → 111 → 000 → 100



时序信号产生器（2）

- 节拍脉冲和存储器读/写时序的译码
  - 若在一个CPU周期中产生四个等间隔的节拍脉冲，则其译码逻辑可表示为：

$$T_1^{\circ}=C_1\cdot\overline{C_2} \qquad T_2^{\circ}=C_2\cdot\overline{C_3}$$

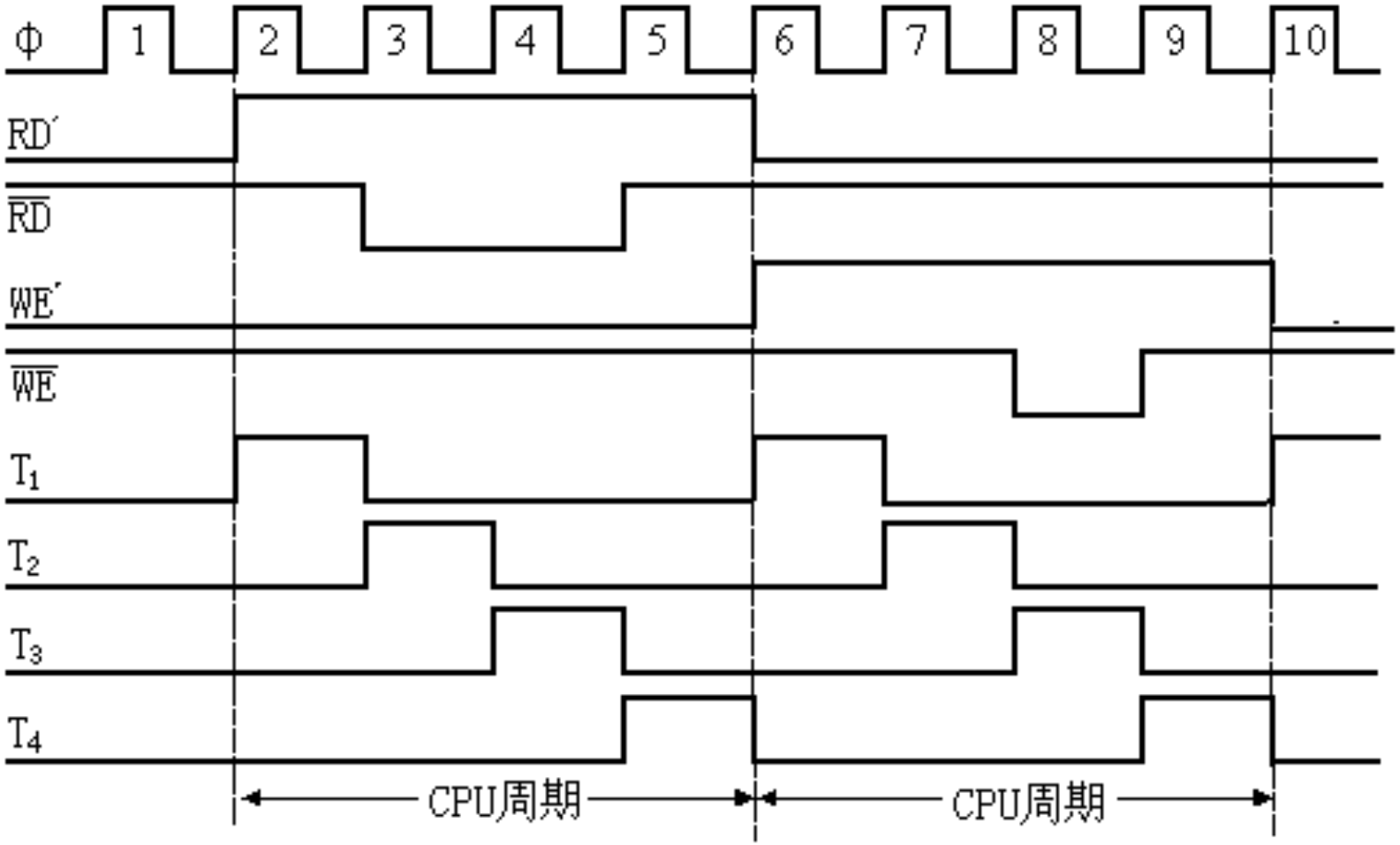
$$T_3^{\circ}=C_3 \qquad T_4^{\circ}=\overline{C_1}$$

  - 存储器读/写时序信号的译码逻辑表达式为

$$RD^{\circ}=C_2\cdot RD' \qquad WE^{\circ}=C_3\cdot WE'$$

RD′、WE′信号来自微程序控制器的控制信号，都是持续时间为一个CPU周期的节拍电位信号

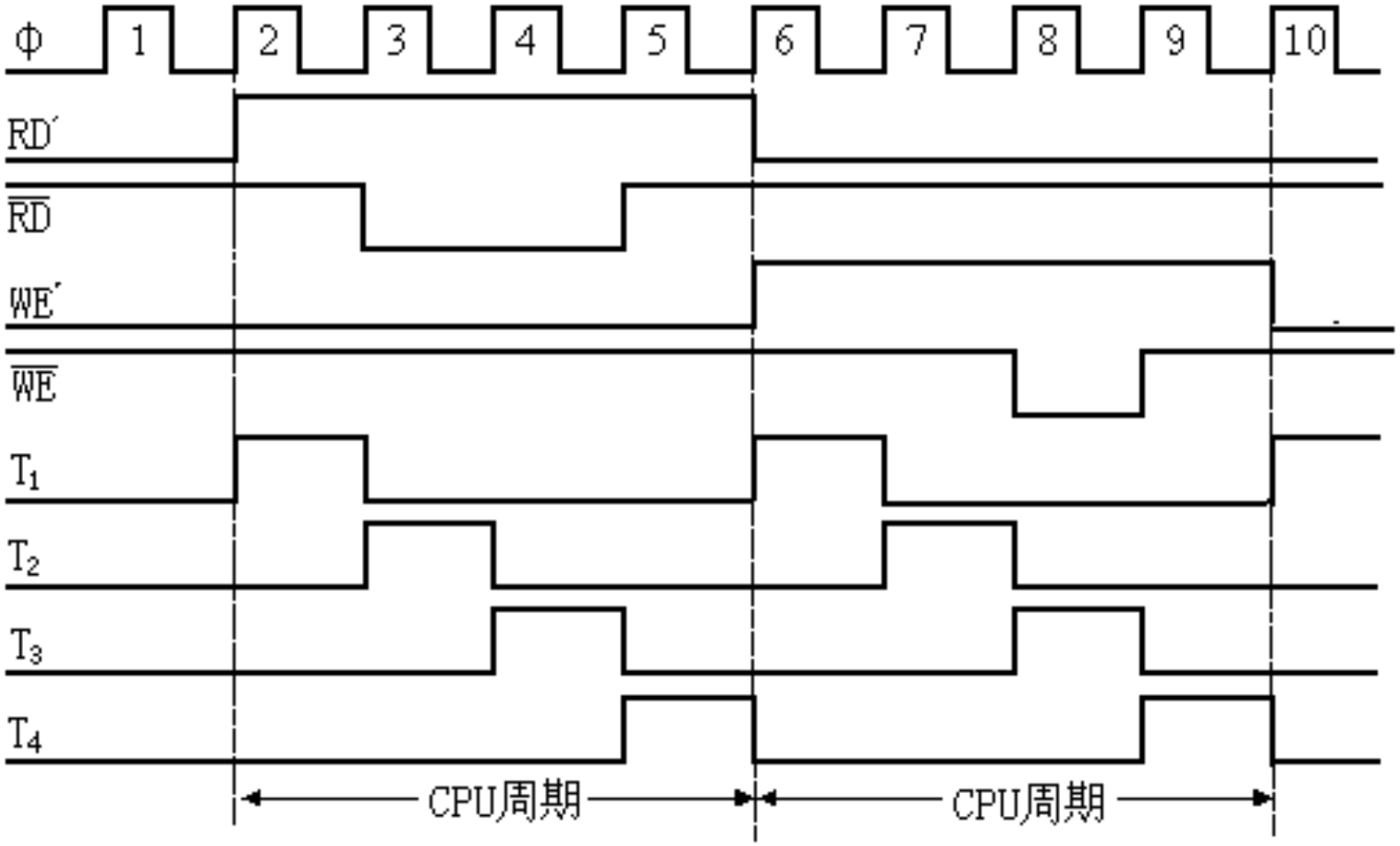
节拍电位与节拍脉冲时序关系图2



时序信号产生器（3）

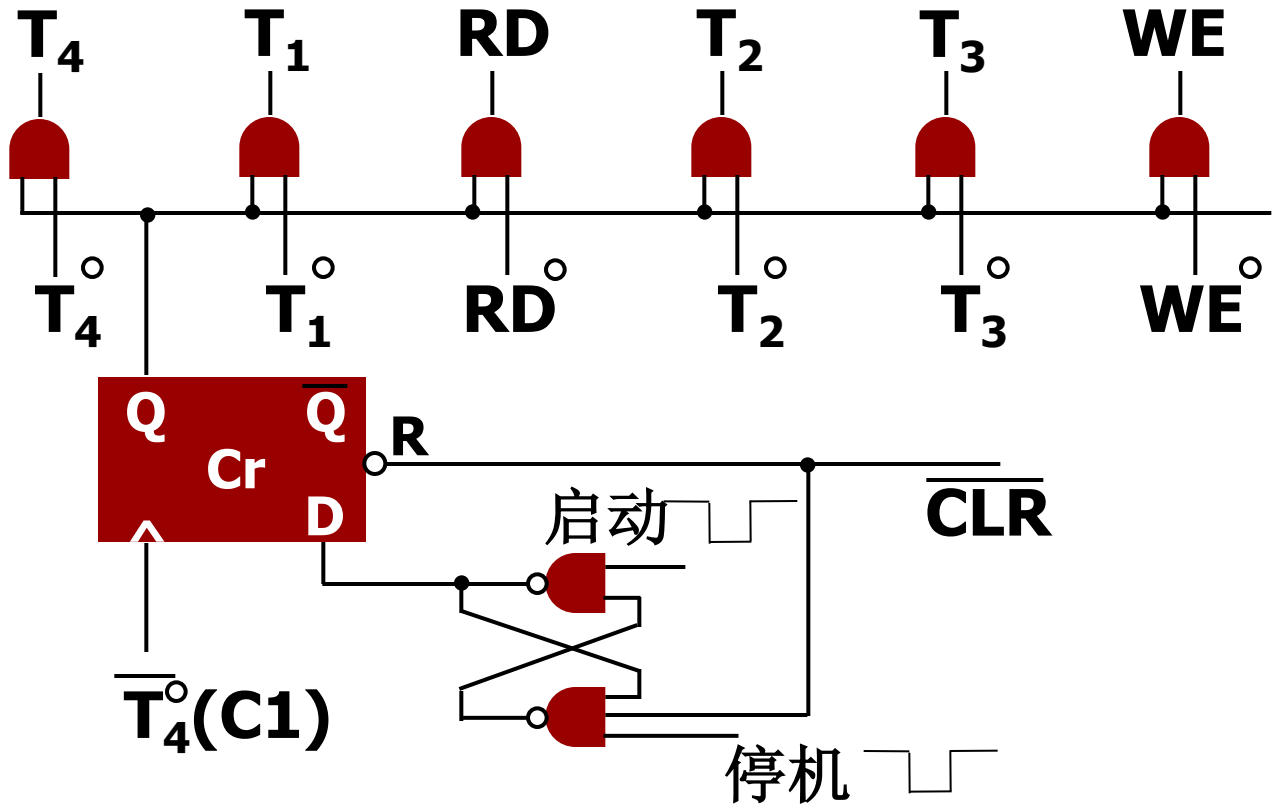
- 启停控制逻辑
  - 机器一旦接通电源，就会自动产生原始的节拍脉冲信号T_1° -T_4° 。但只有发出启动信号后，才允许时序产生器发出CPU工作所需的完整节拍脉冲T_1°-T_4°
  - 故需要由启停控制逻辑来控制T_1°-T_4°以及读/写信号的产生

启动信号保证产生完整的CPU周期信号



启停控制逻辑

- 当触发器Cr的Q端为“1”时，原始节拍脉冲和读/写信号通过门电路发送出去，变成CPU真正需要的节拍脉冲信号和读写时序，反之，就关闭时序产生器



控制方式

- 控制器的控制方式
  - 控制不同操作序列时序信号的方法
- 常用的控制方式
  - 同步控制
  - 异步控制
  - 联合控制

1、同步控制方式

- 在任何情况下，各指令在执行时所需的机器周期数和时钟周期数都固定不变
- 同步控制方式实现方案如下：
  1. 采用完全统一的机器周期执行各种不同的指令。简单指令会导致时间浪费
  2. 采用不定长机器周期。复杂指令采取延长机器周期的方法
  3. 中央控制与局部控制结合。上述方法的结合

2、异步控制方式

- 每条指令、每个操作控制信号按需占用时间
- 在异步控制方式中，各项操作不采用统一的时序信号控制，而是根据指令或部件的具体情况决定。这是一种“应答”方式，没有时间上的浪费，但控制比较复杂。
- 用这种方式产生的操作控制序列没有固定的机器周期数或严格的时钟周期与之同步

3、联合控制方式

同步控制和异步控制相结合的方式

1. 大部分操作序列安排在固定的机器周期中，对某些时间难以确定的操作则以执行部件的“应答”信号作为本次操作的结束
2. 机器周期的节拍脉冲数固定，但是各条指令周期的机器周期数不固定

5.4 微程序控制器

基本概念（1）

- 1951年，在曼彻斯特大学召开的计算机会议上，英国剑桥大学的Maurice Vincent Wilkes教授首先提出了微程序的概念和原理
- 微程序设计技术是利用软件方法来设计硬件的一门技术
  - 微程序控制器与硬布线控制器相比较，具有规整性、灵活性、可维护性等一系列优点

基本概念（2）

- 微程序控制的基本思想
  - 用多条微指令（即：一个微程序）解释每条指令的执行过程。全部的微程序有机地组合在一起，存储在控制存储器（只读存储器）中
  - 当机器运行时，一条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，使相应的部件执行特定的操作

微命令和微操作

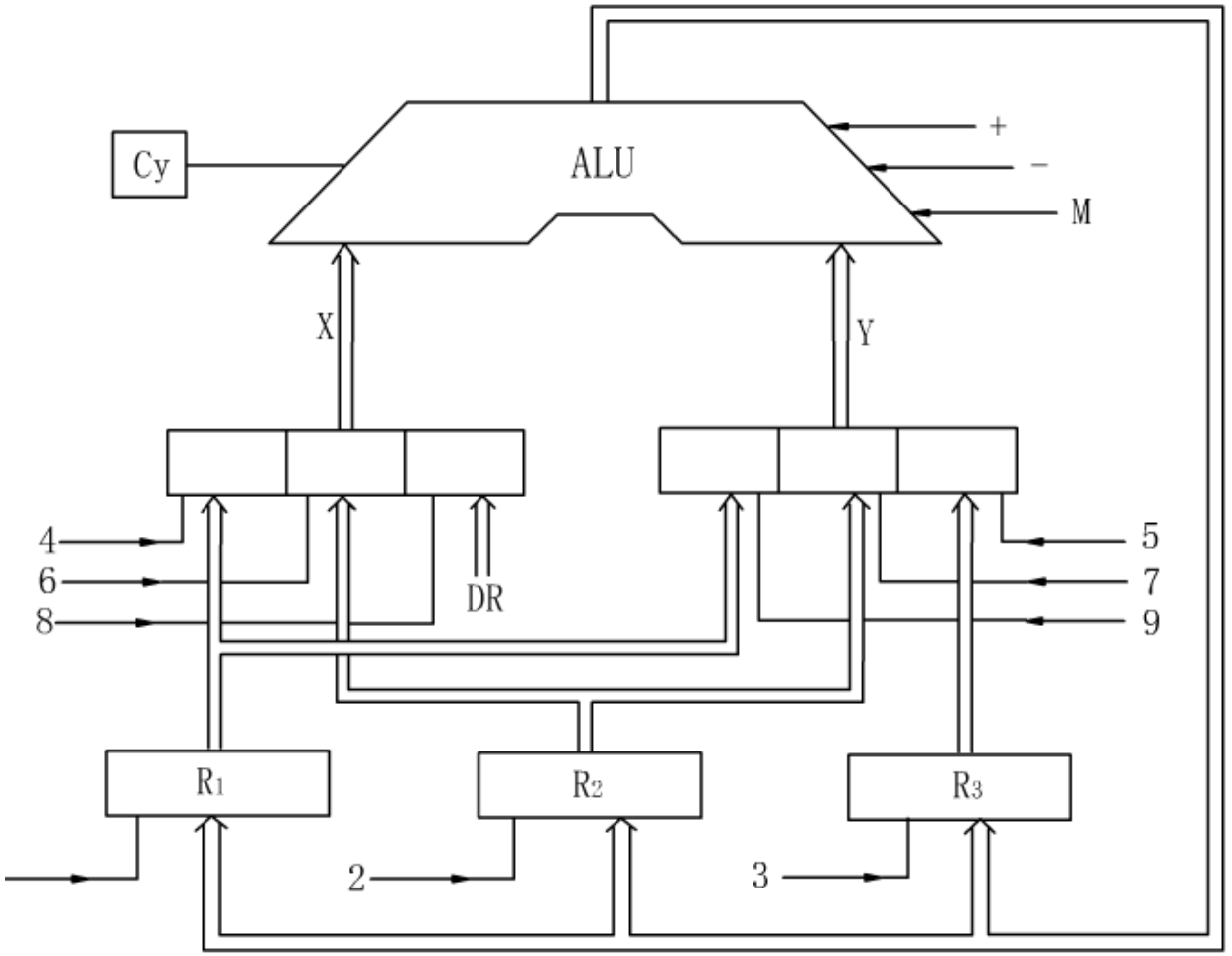
- 计算机的构成
  - 控制部件—控制器
  - 执行部件—运算器、存储器、外围设备
- 微命令—控制部件通过控制线向执行部件发出各种控制命令（即：控制信号）
- 微操作—执行部件接受微命令后所进行的特定操作
- 反馈信息
  - 通常执行部件使用反馈线向控制部件报告操作情况，控制部件则根据执行部件的“状态”发出新的微命令

微操作分类

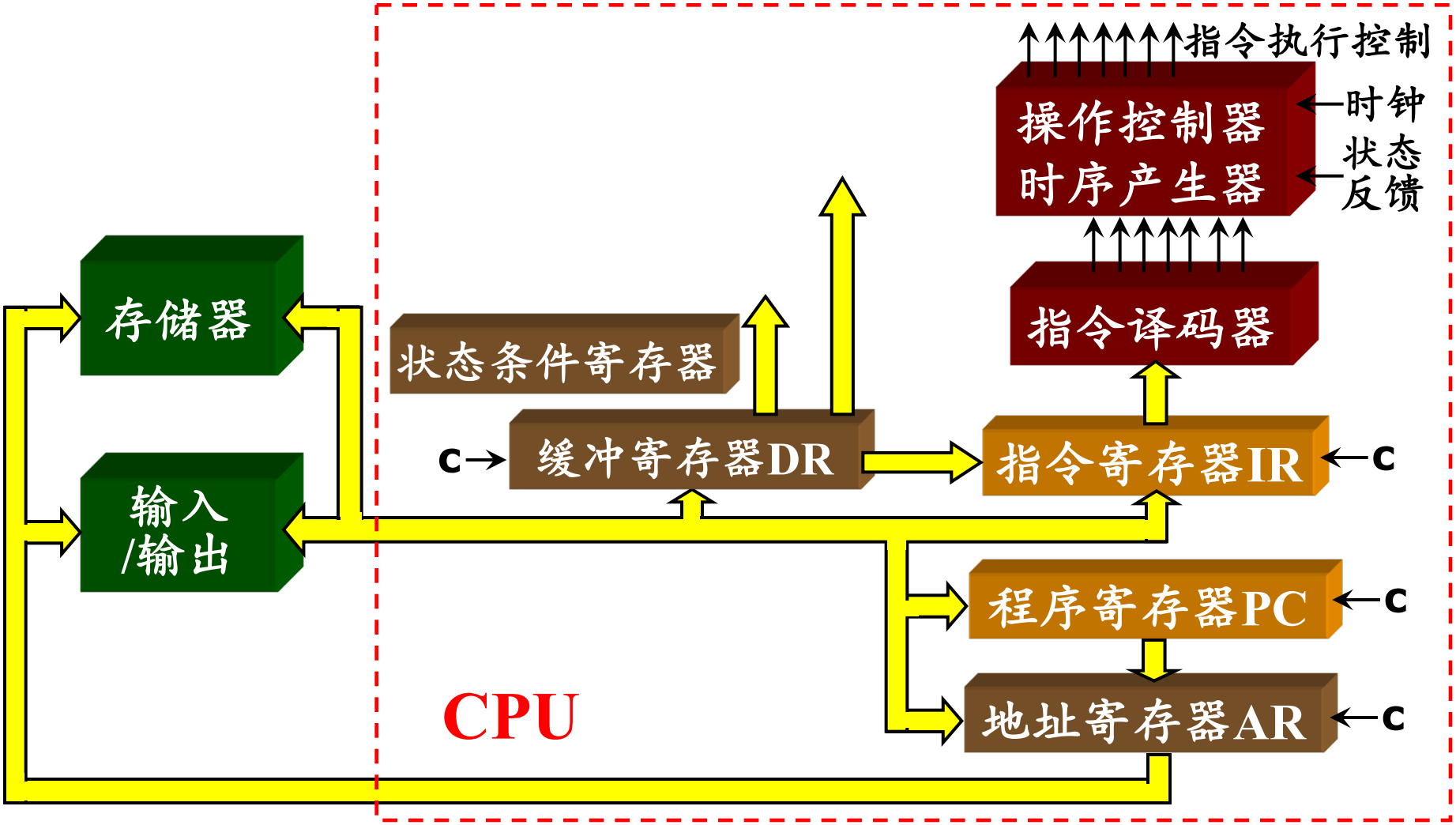
- 相容性操作：同时或在同一个CPU周期内可以并行执行的微操作
- 相斥性操作：不能同时或不能在同一个CPU周期内并行执行的微操作

简单运算器数据通路图

- “＋”、“－”、“M”三个微操作是相斥性微操作
- 4/6/8三个微操作是相斥的，5/7/9也是相斥性的
- 1/2/3可同时进行的，所以是相容性的微操作
- ALU的X输入微操作4/6/8和Y输入的5/7/9任意两个微操作，也都是相容性的



与之配套的控制器


指令执行控制
操作控制器
时序产生器
时钟
状态反馈
指令译码器
存储器
状态条件寄存器
C
缓冲寄存器DR
指令寄存器IR
C
输入/输出
程序寄存器PC
C
CPU
地址寄存器AR
C

微指令

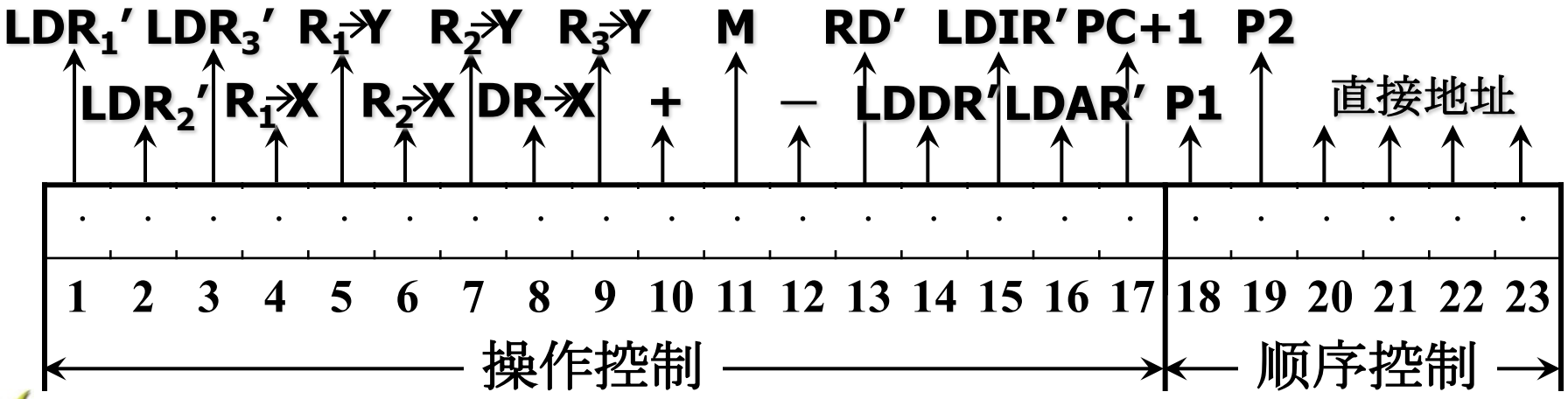
- 在机器的一个CPU周期中，一组实现一定操作功能的微命令的组合，构成一条微指令

或者说

- 用一条微指令对应一条机器指令的一个执行步骤
- 微指令需要具备的2个功能
  - 提供一条机器指令的一个执行步骤所需要的控制信号，以实现该执行步骤的操作功能
  - 提供读出下一条待用微指令的地址，以便自动有序地读出每一条微指令，解决机器指令执行步骤之间的正确的接续关系

微指令基本格式

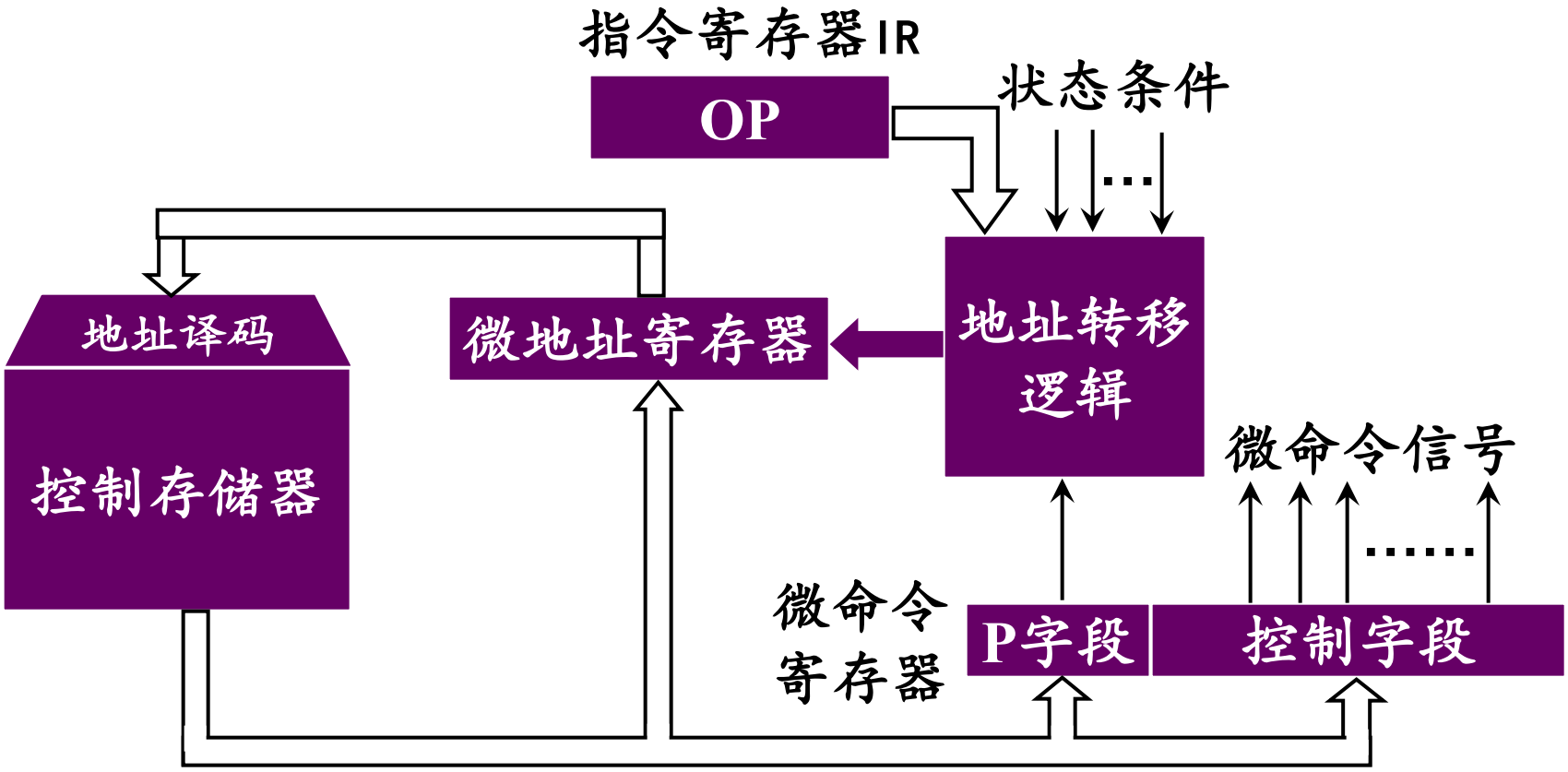
- 微指令由操作控制和顺序控制两大部分组成，图中微指令字长为23位，
- 操作控制部分用来发出控制信号，每一位表示一个微命令
  - 控制信号均为节拍电位信号，持续时间都是一个CPU周期，只有与节拍脉冲相结合才能得到实际控制信号
- 顺序控制部分用来指明下一条微指令的地址
  - 微指令中最后4位直接给出下一条微指令的地址，另外两位为测试方式位，根据测试结果来决定下一条微指令的地址



微程序控制器

- 微程序控制器
  - 控制存储器
  - 微指令寄存器
    - 微地址寄存器
    - 微命令寄存器
  - 地址转移逻辑

微程序控制器原理框图（1）



微指令寄存器＝微地址寄存器＋微命令寄存器

微程序控制器原理框图（2）

- 微指令寄存器
  - 用来存放由控制存储器读出的一条微指令
  - 其中微地址寄存器决定将要访问的下一条微指令的地址
- 地址转移逻辑
  - 通常由微指令的顺序控制部分直接给出下一条微指令的地址（微地址），存放在微地址寄存器中
  - 微程序出现分支时地址转移逻辑要完成修改微地址的任务
- 控制存储器
  - 用来存放实现全部指令系统的微程序的只读存储器
  - 读出一条微指令并执行的时间总和称为一个微指令周期
  - 在串行方式的微程序控制器中，微指令周期就是只读存储器的工作周期

控制存储器

000	微命令	P	001	取指周期微程序
001	微命令	p	002	
002	微命令	p	002	
	...			
xxx	微命令	P	xxx+1	ADD操作微程序
	微命令	p	xxx+2	
	微命令	p	000	
	...			
yyy	微命令	P	yyy+1	LDA操作微程序
	微命令	p	yyy+2	
	微命令	p	000	
	...			

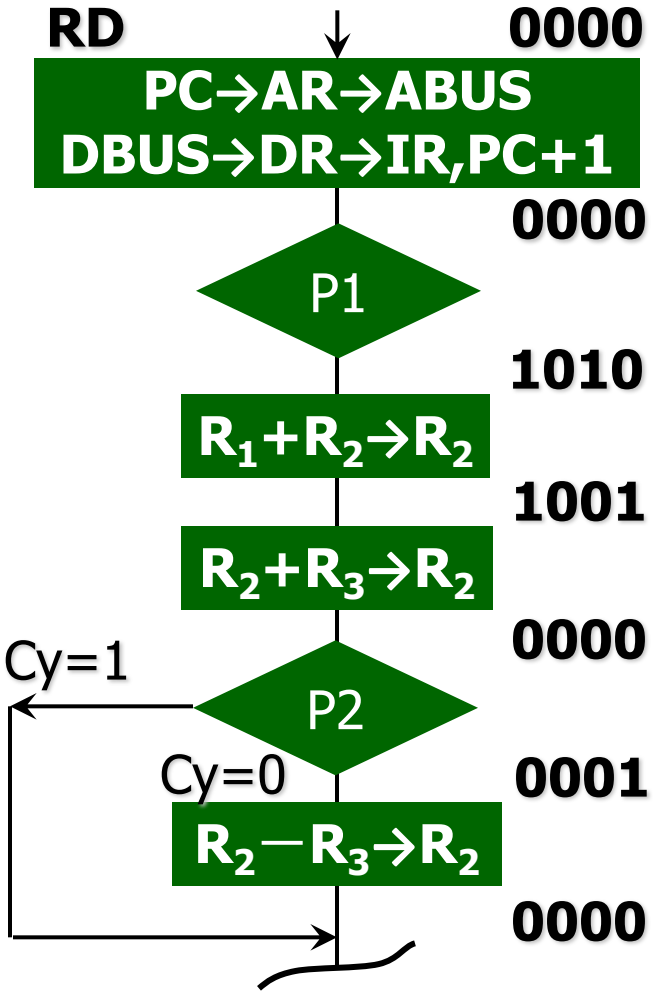
微程序特点

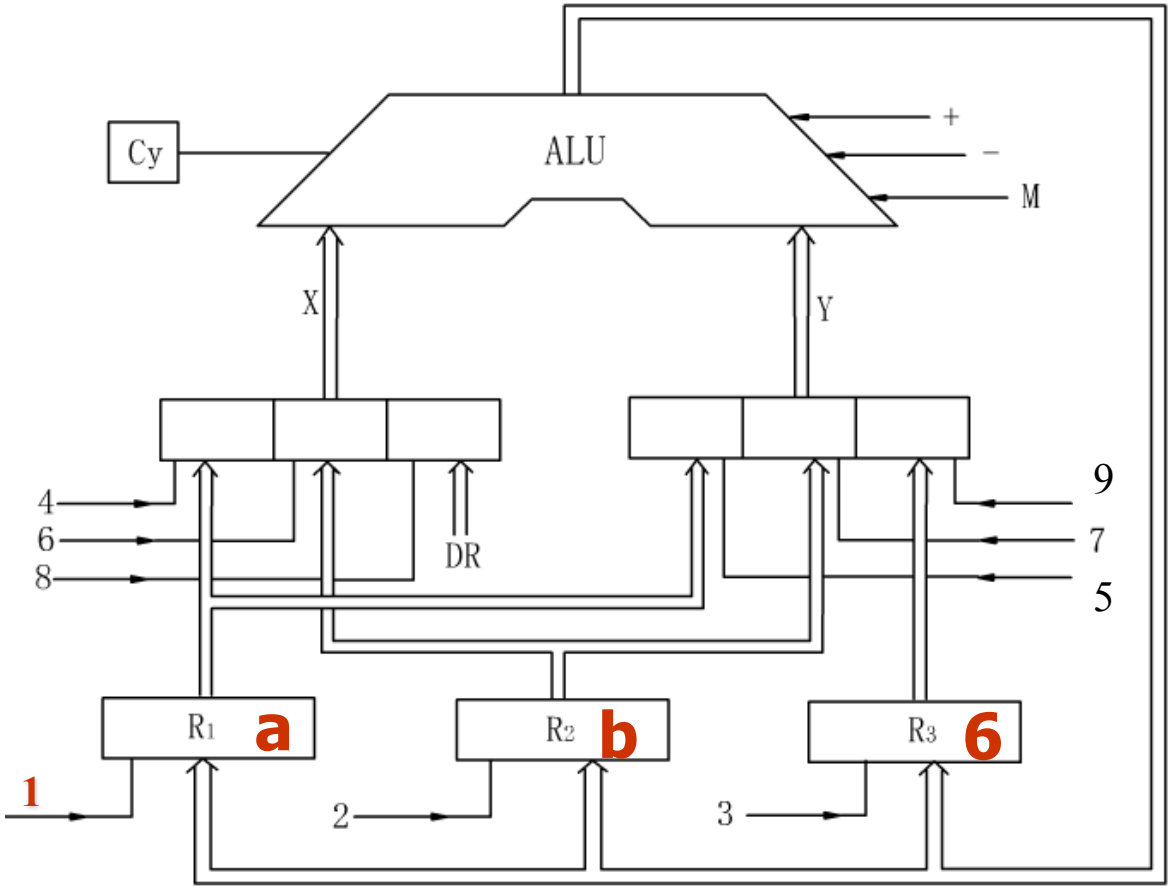
- 一条机器指令是若干条微指令组成的序列来实现的
- 一条机器指令对应着一段微程序，而微程序的总和便可实现整个的指令系统
- 微程序设计可以很容易地在不同的微体系结构上实现相同的指令系统

微程序举例（1）

- 十进制加法
  - 用BCD码来完成十进制数的加法运算
  - 当二数相加之和大于9时，必须对和数进行加6修正
- 假设数a和b已存放在R1和R2寄存器中，数6存放在寄存器R3中
- 算法要求先进行a＋b＋6运算，然后判断结果有无进位：当进位标志Cy＝1，不减6；当Cy＝0，减去6
- 画出十进制加法微程序流程图，由四条微指令组成
  - 第一条是“取指”微指令
  - 第二条微指令完成a＋b运算
  - 第三条微指令完成a+b+6运算，同时进行判别测试
  - 第四条微指令完成和数减6运算

微程序举例（2）


RD
0000
PC→AR→ABUS
DBUS→DR→IR,PC+1
0000
P1
1010
R1+R2→R2
1001
R2+R3→R2
0000
Cy=1
P2
Cy=0
0001
R2－R3→R2
0000


Cy
ALU
+
－
M
X
Y
4
6
8
DR
9
7
5
R1
a
R2
b
R3
6
1
2
3

微程序举例（3）

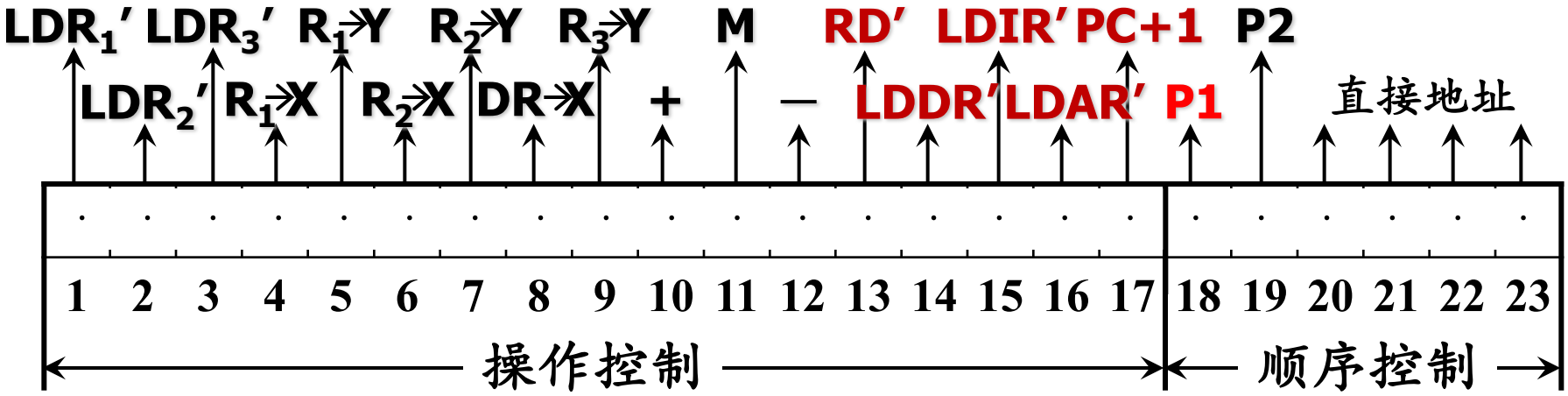
- 微程序的调用
  - 以十进制加法指令操作码为地址，去查微地址映射部件得到微程序在控制存储器中的地址，就可以调出所需要的微程序（如上图中微程序地址为1010）
  - 微地址映射部件是用ROM实现的，地址输入为指令寄存器IR的操作码，输出为该指令对应的微程序段的入口地址

微程序举例（4）

- 第一条微指令的二进制编码是

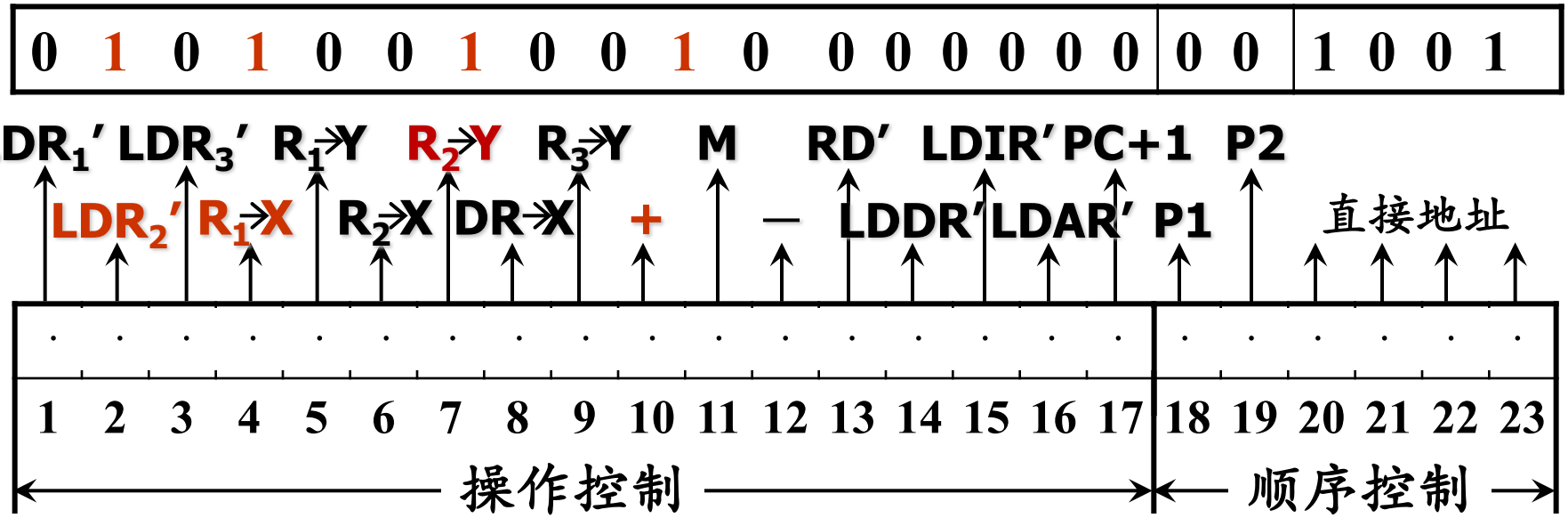
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0

  - 操作控制字段有五个微命令，完成“取指”操作，并对PC+1
  - 顺序控制字段指明下一条微指令的地址是0000，但由于第18位为1，是P1测试，即用OP字段（1010）作为形成下一条微指令的地址，故微地址寄存器的内容修改为1010



微程序举例（5）

- 在第二个CPU周期，按照1010这个微地址读出第二条微指令，其二进制编码是

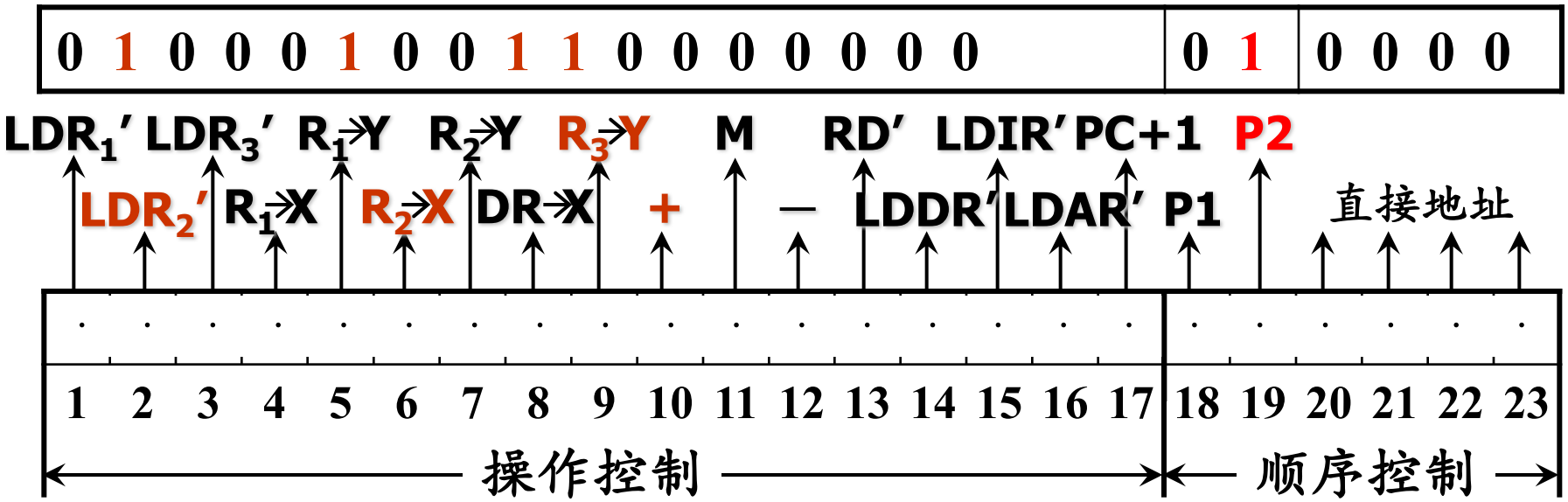


- 这条微指令中，操作控制部分发出四个微命令，完成R1+R2→R2运算

  顺序控制部分由于判别测试字段P1和P2均为0，表示不进行测试，于是直接给出下一条微指令的地址为1001

微程序举例（6）

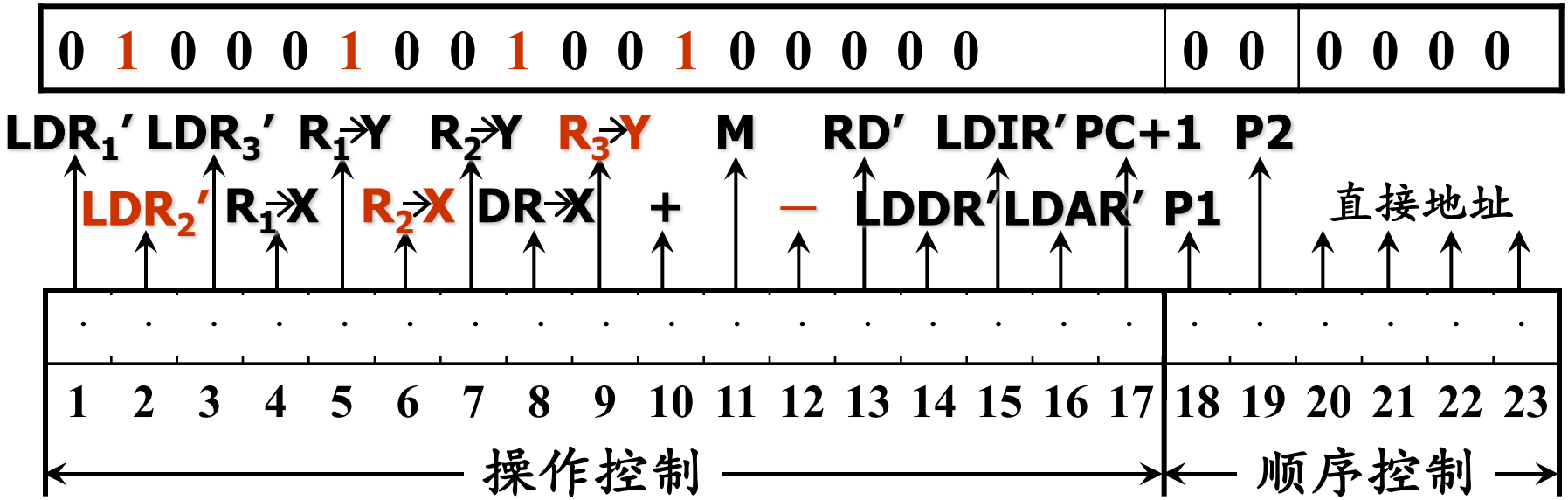
- 在第三个CPU周期，按照1001这个微地址读出第三条微指令，其二进制编码是



- 这条微指令中，操作控制部分发出四个微命令，完成R2+R3→R2运算
- 顺序控制部分的测试字段P2为1，表示需要进行P2测试，P2测试就是检测标志位Cy，当Cy＝0时，下一条微指令的地址为0001；而当Cy＝1时，下一条微指令的地址为0000

微程序举例（7）

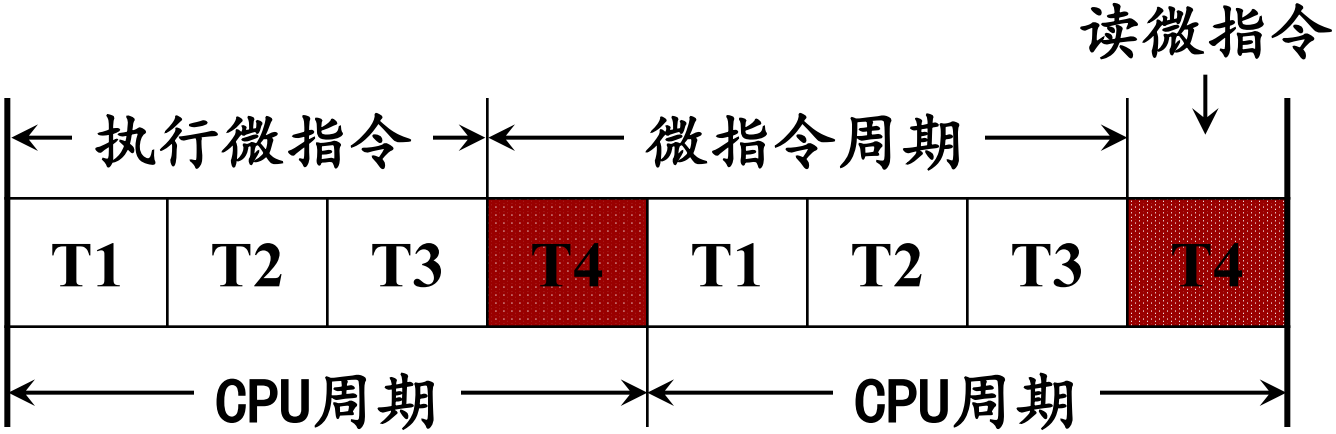
- 假设Cy＝0，第四个CPU周期开始时，按照0001这个微地址读出第四条微指令，其二进制编码是



- 这条微指令中，操作控制部分发出四个微命令，完成R2-R3→R2运算
- 顺序控制部分直接给出下一条微指令的地址为0000，该地址存放的微指令是“取指”微指令。即取下一条指令

微指令周期与CPU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
- 为了保证整个机器控制信号的同步，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



- 一个CPU周期为0.8μs，包含四个等间隔的节拍脉冲T1－T4，每个脉冲宽度为200ns
- 用T4作为读取微指令的时间，用T1＋T2＋T3时间作为执行微指令的时间

机器指令与微指令的关系（1）

- 一条机器指令对应一个微程序，这个微程序是由若干条微指令序列组成的
- 从指令与微指令，程序与微程序，地址与微地址的对应关系来看，前者与内存储器有关，后者与控制存储器有关

机器指令与微指令的关系（2）