计算机组成原理

第五章 中央处理器CPU

刘 超

中国地质大学（武汉）计算机学院

主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

CPU的组成和功能

- CPU的功能
- CPU的组成
- CPU中的主要寄存器
- 操作控制器
- 时序产生器

CPU的功能

- 取出指令并执行指令的部件------CPU
  - 指令控制：指令执行的顺序控制；
    - 程序是一个指令序列，这些指令的相互顺序不能任意颠倒，必须严格按程序规定的顺序进行。（首要任务）
  - 操作控制：产生各种操作信号；
    - 解释指令的操作码，通过若干操作信号组合控制来实现指令功能。
  - 时间控制：控制操作信号的发生时间；
    - 完成一条指令的若干操作信号定时，有序执行。
  - 数据加工：----ALU. 算术/逻辑运算；（根本任务）
  - 异常处理：接收、控制、管理信号资源及异常情况。

CPU的组成

- 运算器
  - 算术运算/逻辑运算
  - 累加器、状态条件寄存器、缓存寄存器、移码器、锁存器、求补器等。
- 控制器
  - 从内存取出一条指令,并指出下条指令的地址
  - 对指令进行译码,产生相应的控制信号
  - 指挥并控制CPU,内存和I/O设备之间的数据传送
  - 程序计数器、指令寄存器、指令译码器、时序产生器、操作控制器、地址寄存器等。

CPU的基本组成结构

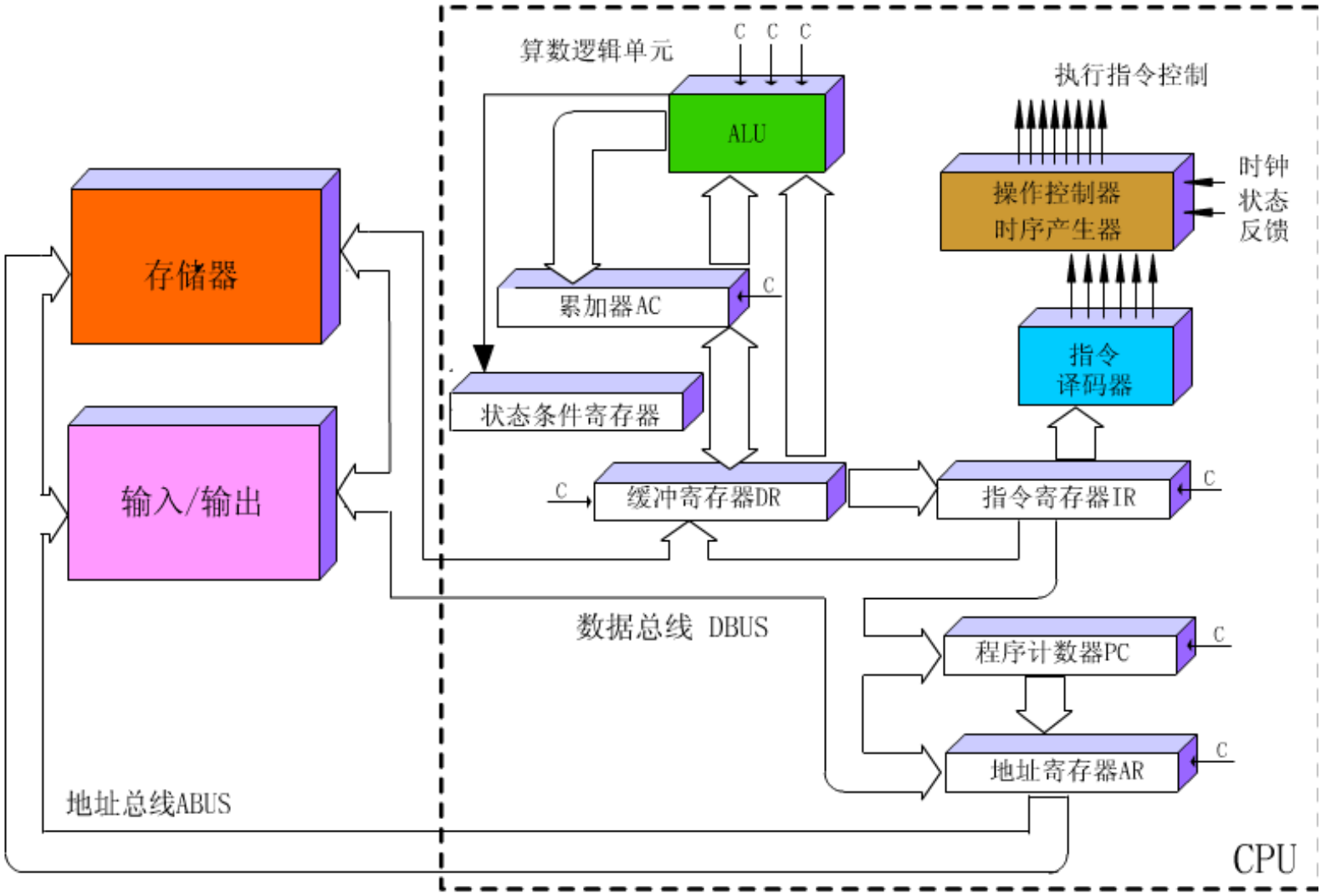


图5.1 C P U 的结构

CPU中的主要寄存器

- PC(Program Counter)----程序计数器
- AR(Address Register)---地址寄存器
- DR(Data Register)----数据缓冲寄存器
- IR(Instruction Register)-----指令寄存器
- AC(Accumulate Count)---累加寄存器
- PSW (Program Status Word)程序状态字

PC

- 为了保证程序能够连续地执行下去，CPU如何确定下一条指令的地址？
- 程序计数器，又称指令计数器。在程序开始执行前，首先将起始地址，即程序的第一条指令所在的内存单元地址送入PC，因此PC的内容即是从内存提取的第一条指令的地址。
- 当执行指令时，CPU将自动修改PC的内容，以便使其保持的总是将要执行的下一条指令的地址。由于多数指令都是按顺序来执行的，修改的过程通常只是简单的对PC加1。
- 当遇到转移指令如JMP指令时，那么后继指令的地址（即PC的内容)必须从指令的地址段取得。在这种情况下，下一条从内存取出的指令将由转移指令来规定，而不是像通常一样按顺序来取得。因此程序计数器的结构应当是具有寄存信息和计数两种功能的结构。

AR

- 地址寄存器用来保存当前CPU所访问的内存单元的地址。使用地址寄存器来保持地址信息，直到内存的读/写操作完成为止 。
- 当CPU和内存进行信息交换，即CPU向内存存/取数据时，或者CPU从内存中读出指令时，都要使用地址寄存器和数据缓冲寄存器。
- 地址寄存器的结构和数据缓冲寄存器、指令寄存器一样，通常使用单纯的寄存器结构。信息的存入一般采用电位-脉冲方式，即电位输入端对应数据信息位，脉冲输入端对应控制信号，在控制信号作用下，瞬时地将信息打入寄存器。

DR

- 数据缓冲寄存器用来暂时存放由内存储器读出的一条指令或一个数据字；反之，当向内存存入一条指令或一个数据字时，也暂时将它们存放在数据缓冲寄存器中。
- 缓冲寄存器的作用是：

(1)作为CPU和内存、外部设备之间信息传送的中转站；

(2)补偿CPU和内存、外围设备之间在操作速度上差别；

(3)在单累加器结构的运算器中，数据缓冲寄存器还可兼作为操作数寄存器。

IR

- 指令寄存器
- IR 用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存取到DR中，然后再传送至IR。指令划分为操作码和地址码字段，由二进制数字组成。为了执行任何给定的指令，必须对操作码进行测试，以便识别所要求的操作。 指令译码器就是做这项工作的。指令寄存器中操作码字段的输出就是指令译码器的输入。操作码一经译码后，即可向操作控制器发出具体操作的特定信号。

AC

- 累加寄存器AC通常简称为累加器，它是一个通用寄存器。其功能是：当运算器的算术逻辑单元ALU)执行算术或逻辑运算时，为ALU提供一个工作区。累加寄存器暂时存放ALU运算的结果信息。显然，运算器中至少要有一个累加寄存器。
- 目前CPU中的累加寄存器，多达16个，32个，甚至更多。当使用多个累加器时，就变成通用寄存器堆结构，其中任何一个可存放源操作数，也可存放结果操作数。在这种情况下，需要在指令格式中对寄存器号加以编址。

PSW

- 状态条件寄存器保存由算术指令和逻辑指令运行或测试的结果建立的各种条件码内容，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)等等。这些标志位通常分别由 1位触发器保存。
- 除此之外，状态条件寄存器还保存中断和系统工作状态等信息，以便使CPU和系统能及时了解机器运行状态和程序运行状态。因此，状态条件寄存器是一个由各种状态条件标志拼凑而成的寄存器。

控制器基本组成

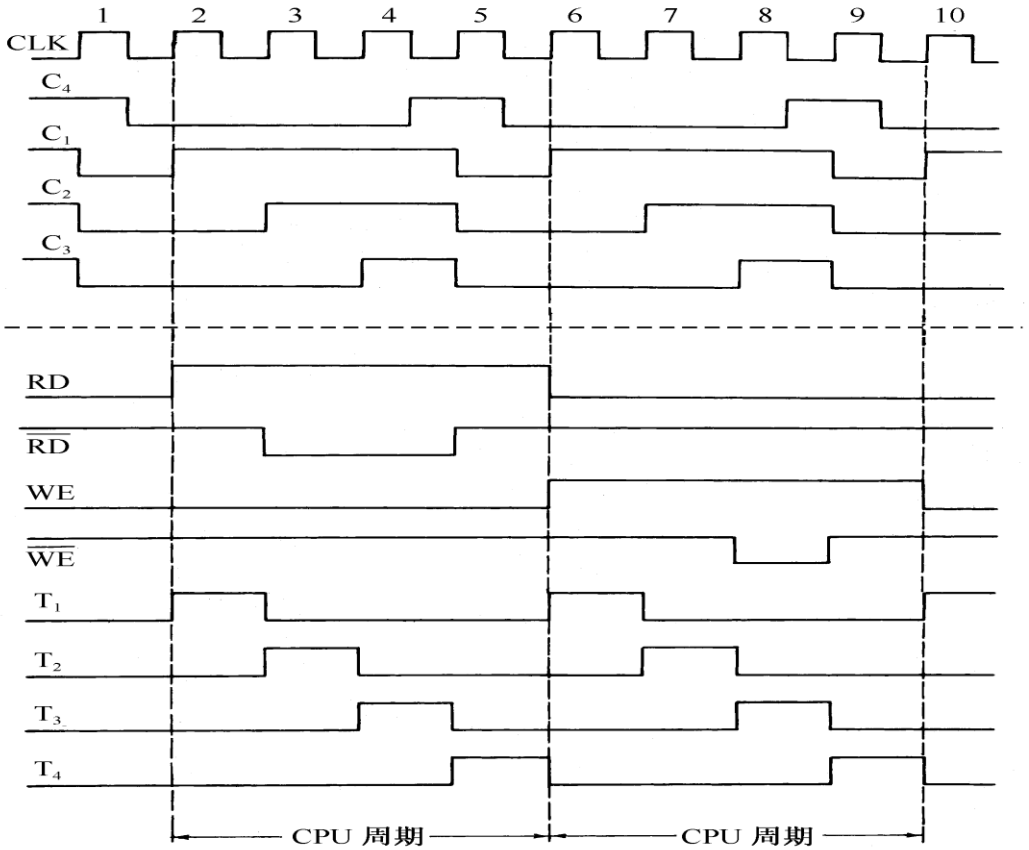
- PC (Program Counter)----程序计数器
- IR (Instruction Register)-----指令寄存器
- ID (Instruction Decoder)---指令译码器
- OC (Operate Controller)---操作控制器
- TG (Timer Generator) ---时序发生器

操作控制器

- **数据通路** 是许多寄存器之间传送信息的通路。
- **操作控制器的功能：** 根据指令操作码和地址码，产生各种控制信号序列，建立正确的数据通路，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、存储逻辑型、时序逻辑与存储逻辑结合型三种。
  - 硬布线控制器 (时序逻辑型) (硬件实现)
  - 微程序控制器 (存储程序型) (软件实现)

时序产生器

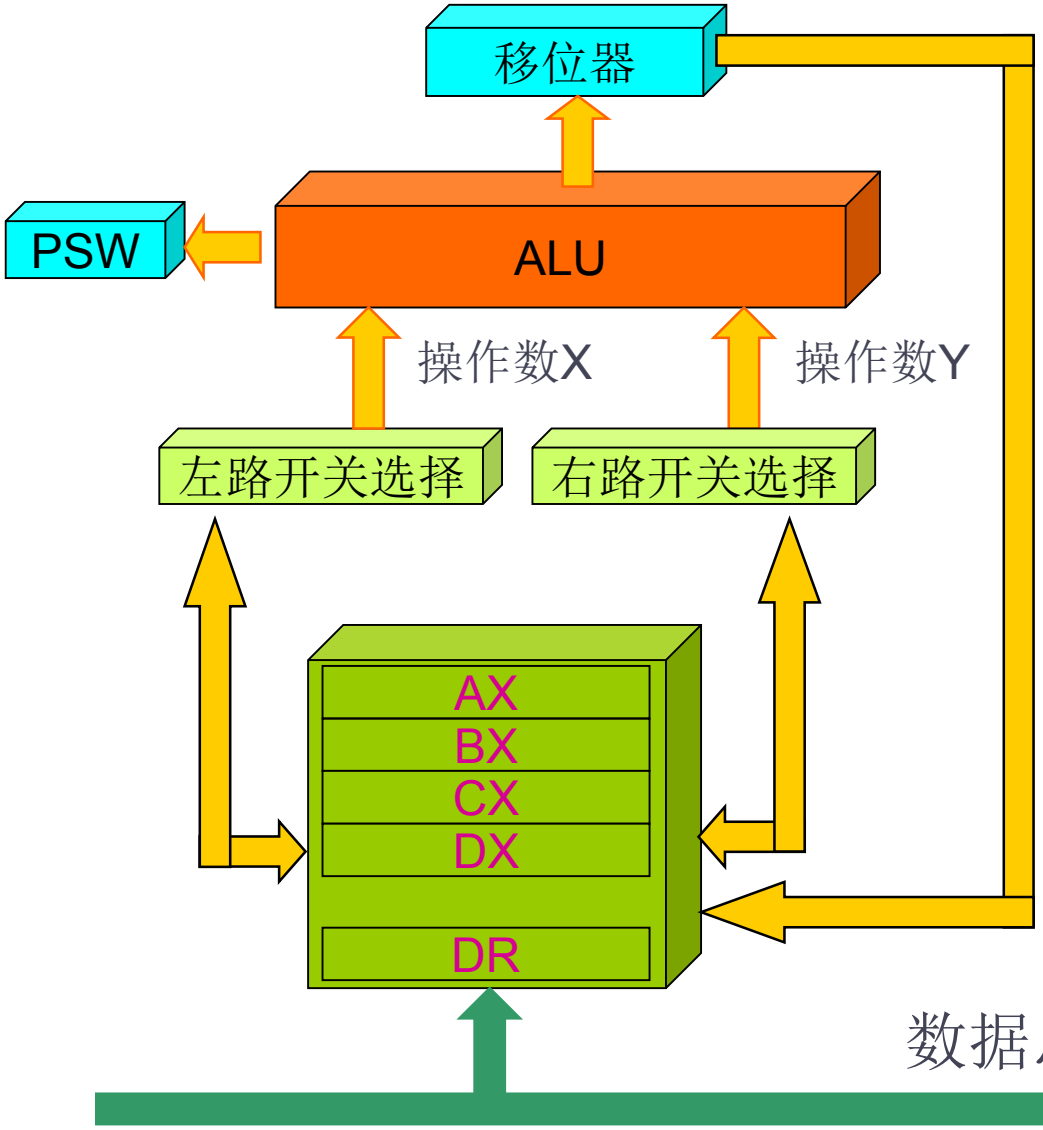
- 产生各种时序信号(电位,脉冲);
- 对各种操作实施时间上的控制。



CPU的主要参数

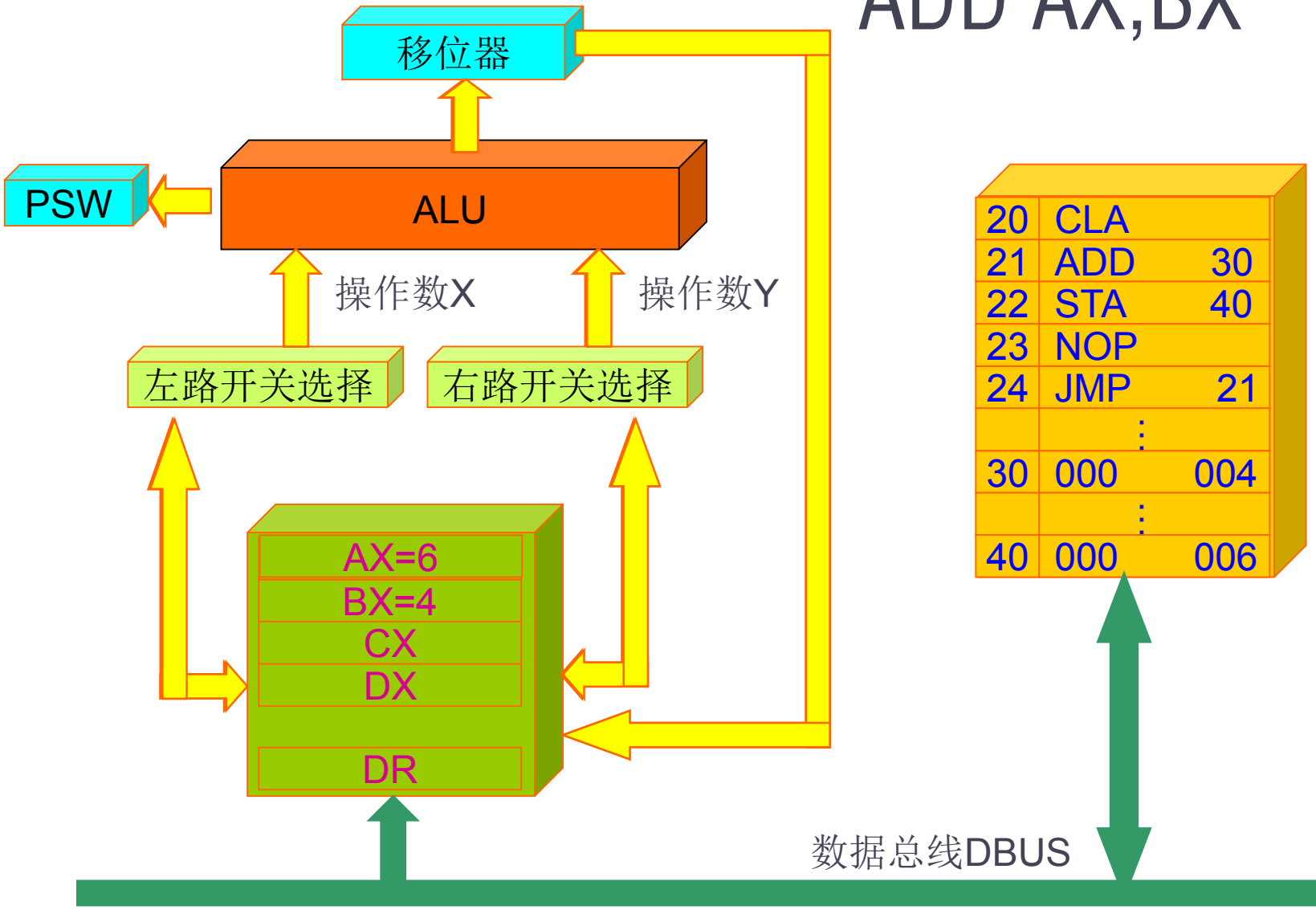
1、字长
2、时钟频率：主频和外频， 主频＝外频×倍频
3、片内Cache容量和速率
4、工作电压：早期CPU工作电压为5V，PIIICPU的电压为1.7V，P4的电压<1.5V(1.35,1.4)
5、地址总线和数据总线宽度
6、制造工艺：在0.25微米的生产工艺最高可以达到600MHz的频率。而0.18微米的生产工艺CPU可达到G赫兹的水平上。0.13微米生产工艺的Pentium4 CPU。例如，P4 2.4C：主频为2.4GHz，外频为200MHz，倍频为10；并集成512K二级缓存，支持800MHz前端总线；P4 2.26B：主频为2.26GHz，外频为133MHz，倍频为17；并集成512K二级缓存，支持533MHz前端总线。

运算器结构

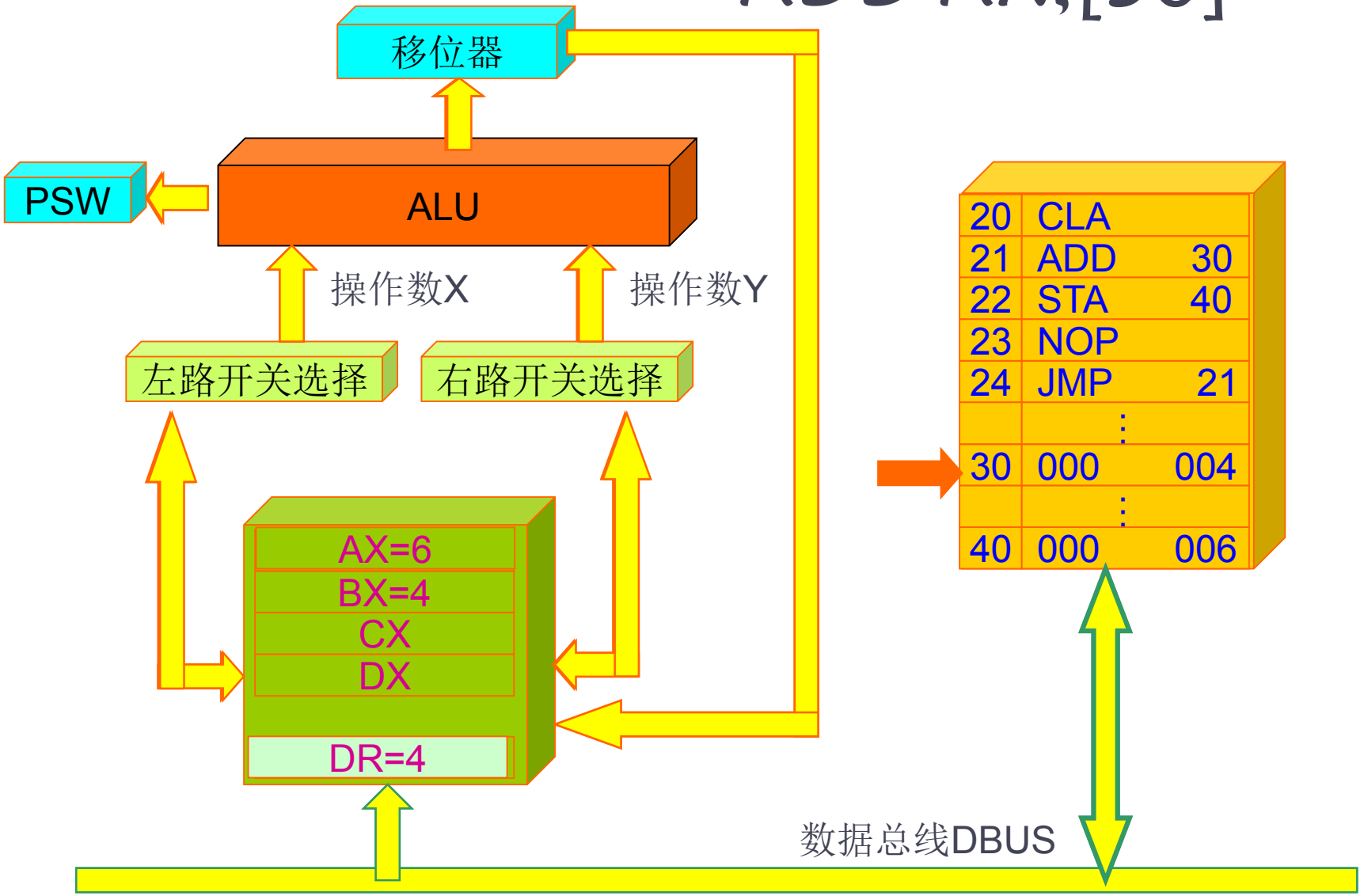


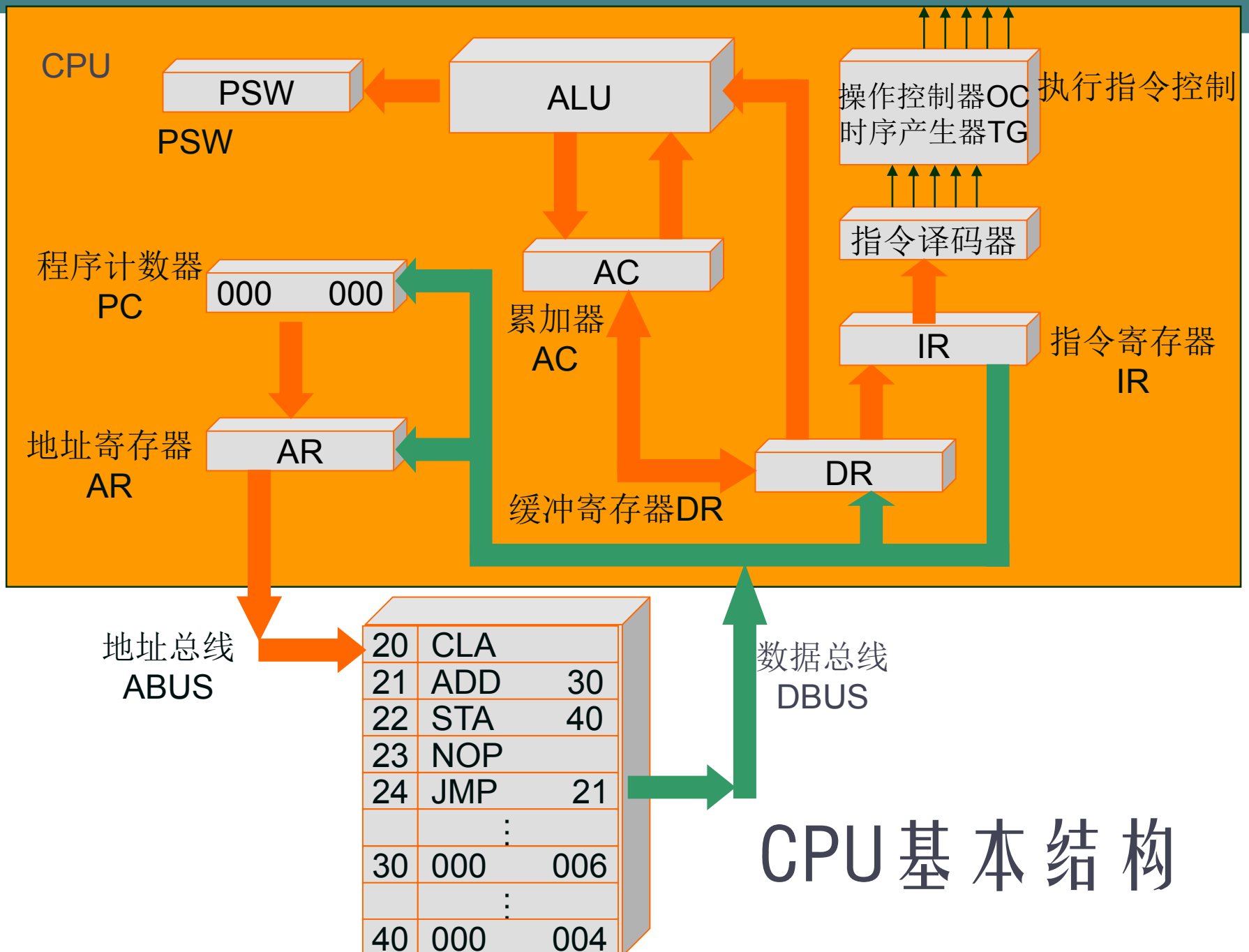
20	CLA	
21	ADD	30
22	STA	40
23	NOP	
24	JMP	21
	⋮	
30	000	004
	⋮	
40	000	006

ADD AX,BX


移位器
PSW
ALU
操作数X
操作数Y
左路开关选择
右路开关选择
AX=6
BX=4
CX
DX
DR
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 004
40 000 006
数据总线DBUS

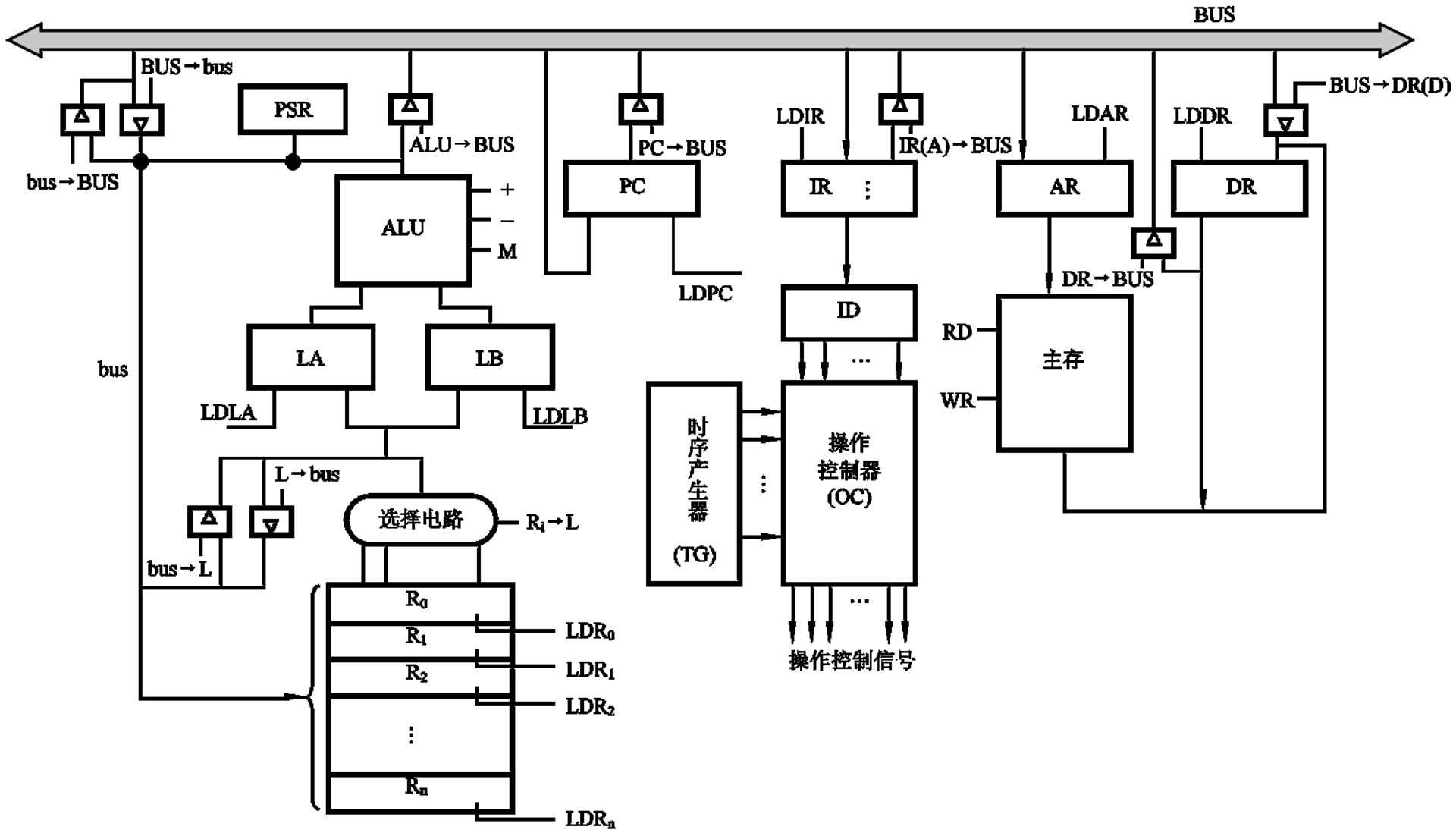
ADD AX,[30]


移位器
PSW
ALU
操作数X
操作数Y
左路开关选择
右路开关选择
AX=6
BX=4
CX
DX
DR=4
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 004
40 000 006
数据总线DBUS


CPU
PSW
PSW
ALU
操作控制器OC
时序产生器TG
执行指令控制
指令译码器
程序计数器
PC
000 000
AC
累加器
AC
IR
指令寄存器
IR
地址寄存器
AR
AR
DR
缓冲寄存器DR
地址总线
ABUS
数据总线
DBUS
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 006
40 000 004

CPU基本结构

主机基本组成


BUS
BUS→bus
PSR
ALU→BUS
PC→BUS
LDIR
IR(A)→BUS
LDAR
LDDR
BUS→DR(D)
bus→BUS
ALU
+
−
M
PC
IR
AR
DR
LDPC
ID
DR→BUS
RD
WR
主存
bus
LA
LB
LDLA
LDLB
L→bus
选择电路
Ri→L
bus→L
R0
R1
R2
Rn
LDR0
LDR1
LDR2
LDRn
时序产生器
(TG)
操作控制器
(OC)
操作控制信号

主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

指令周期

- 指令周期基本概念
- CLA指令周期
- ADD指令周期
- STA指令周期
- NOP指令周期
- JMP指令周期

指令周期

- 时钟周期：节拍脉冲， T周期。单位时间内脉冲发生器重复出现的脉冲次数称为频率，每个计算机系统都规定了时钟脉冲的最高频率，称为主频。频率的倒数即为时钟周期。它是计算机系统的时间基准，是计算机内部的最小时间度量单位。
- CPU 周期：机器周期，从内存读出一条指令的最短时间。机器周期内完成的操作为子操作，其中包含若干个时钟脉冲控制下的微操作。
- 指令周期：从内存取一条指令并执行该指令所用的时间。
  - 由若干个CPU周期组成。
  - CPU周期又包含若干时钟周期（节拍脉冲）

指令周期

- 指令周期由若干个（不同）机器周期组成（2~5个），机器周期由若干个时钟周期组成，每个时钟周期内在节拍信号的作用下完成一个微操作。
- 时钟周期是固定不变的，机器周期、指令周期可以是固定的、也可以是可变的。

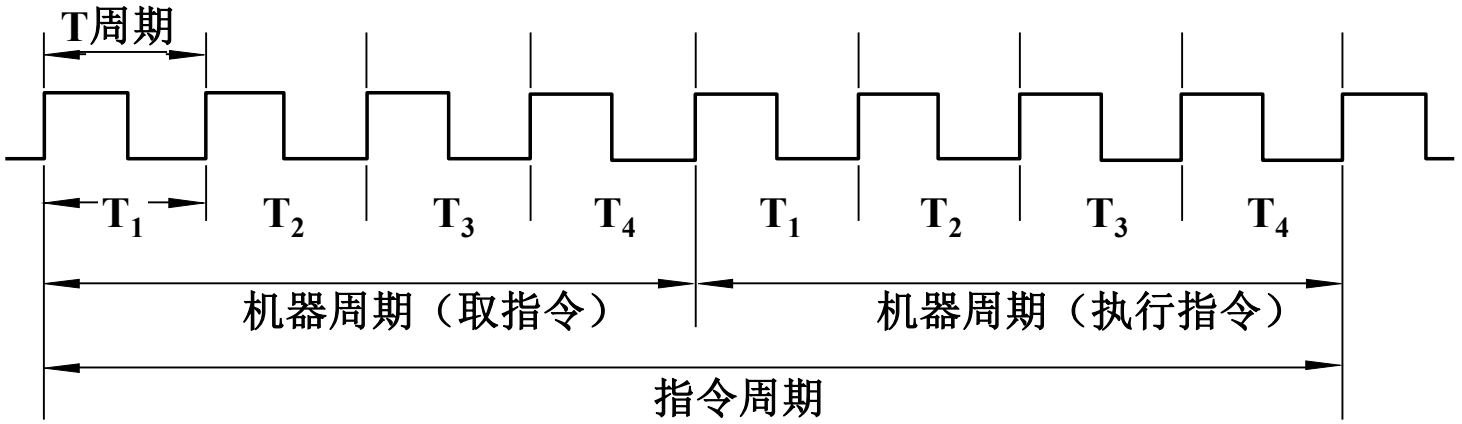
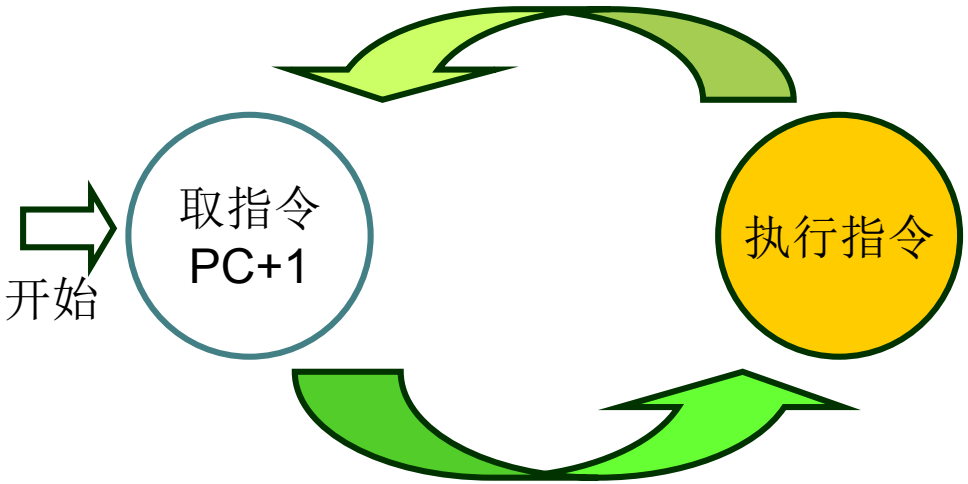


图5.3 指令周期

指令周期基本概念

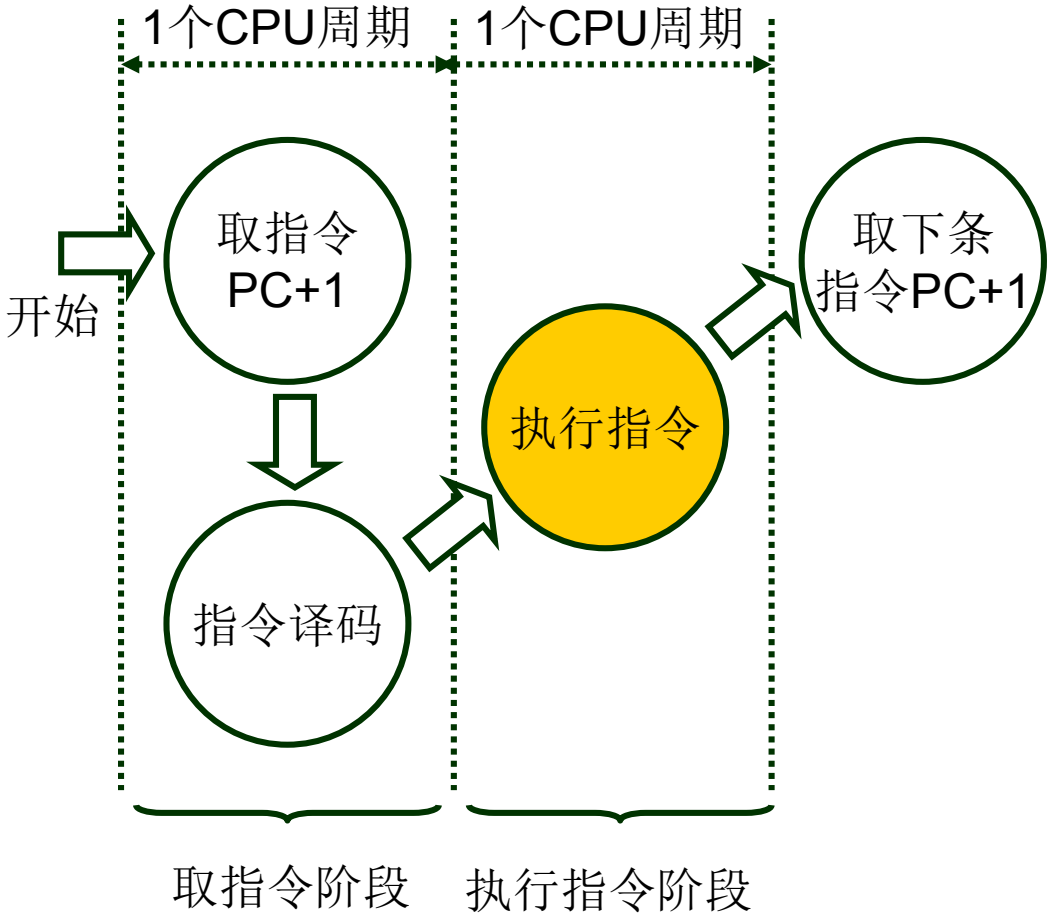
- 取指令周期
- 取操作数周期（可无）
- 执行周期

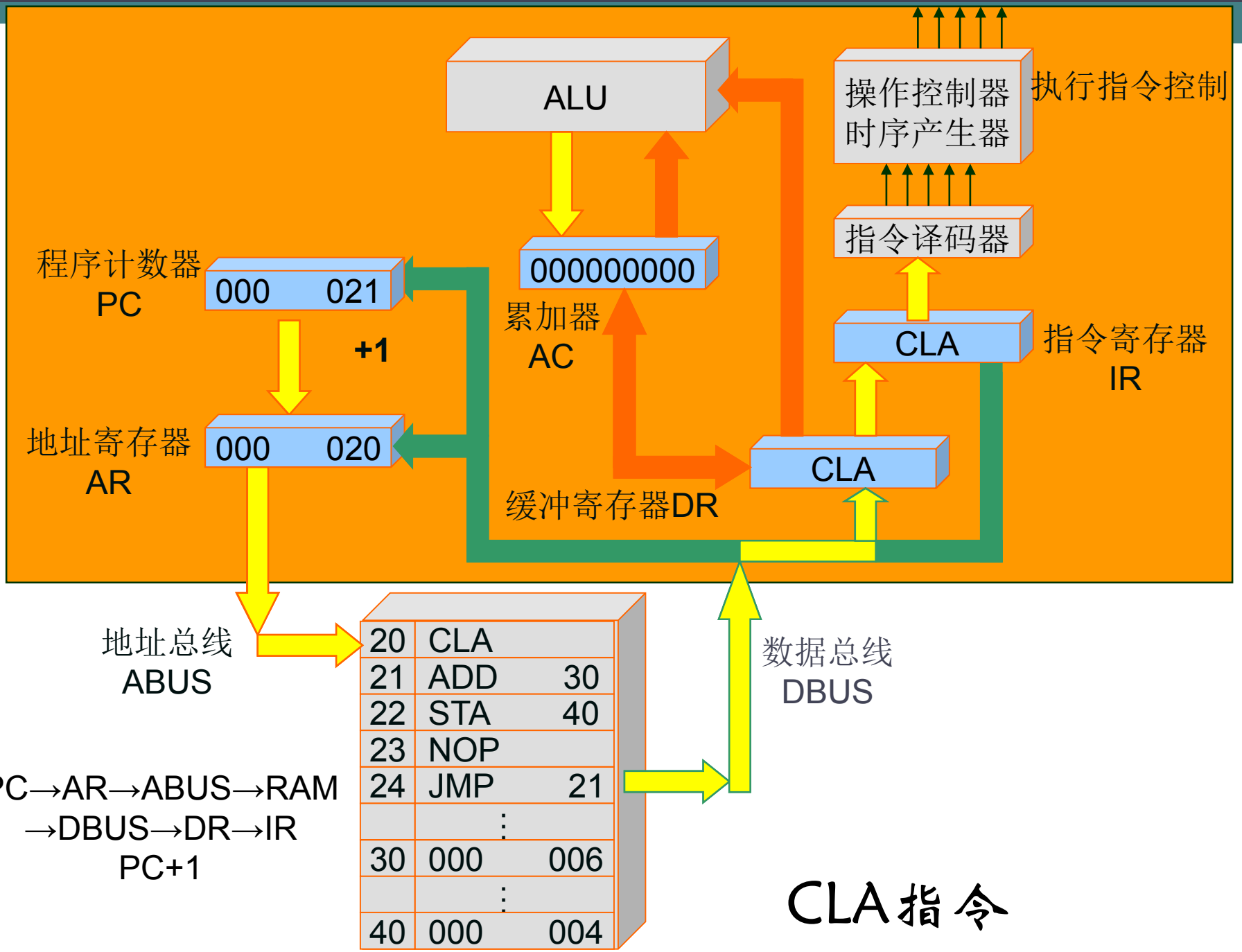


五条指令的执行指令周期及过程

八进制地址	八进制内容	助记符
020	250 000	CLA
021	030 030	ADD 30
022	021 031	STA 40
023	000 000	NOP
024	140 021	JMP 21
.	.	.
.	.	.
030	000 006	数据
031	000 040	数据
.	.	.
,	.	.
040	存和数单元	数据

CLA指令周期



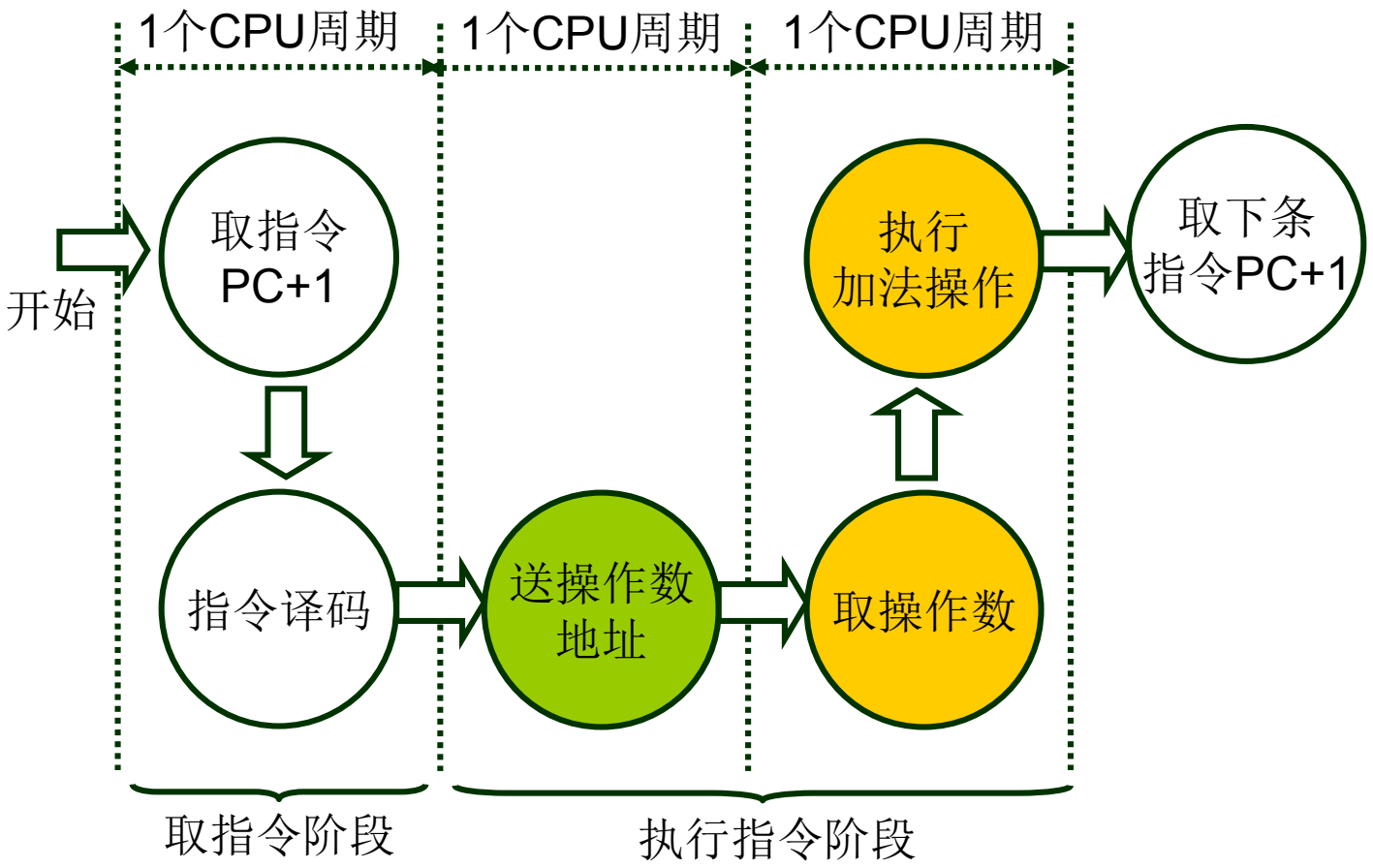

ALU
操作控制器
时序产生器
执行指令控制
指令译码器
000000000
累加器
AC
程序计数器
PC
000 021
+1
CLA
指令寄存器
IR
地址寄存器
AR
000 020
CLA
缓冲寄存器DR
地址总线
ABUS
数据总线
DBUS
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 006
40 000 004
PC→AR→ABUS→RAM
→DBUS→DR→IR
PC+1
CLA指令

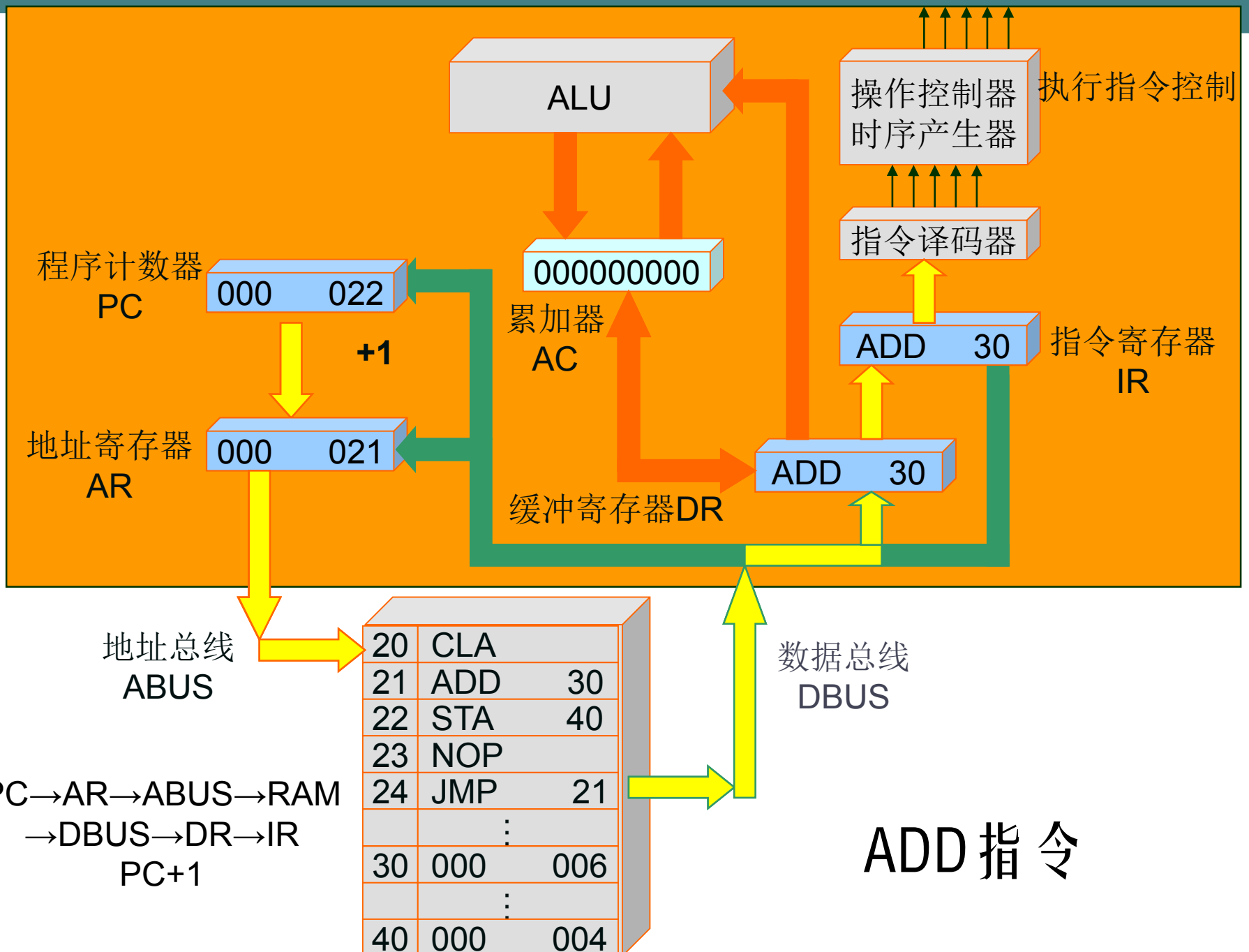
执行过程的操作

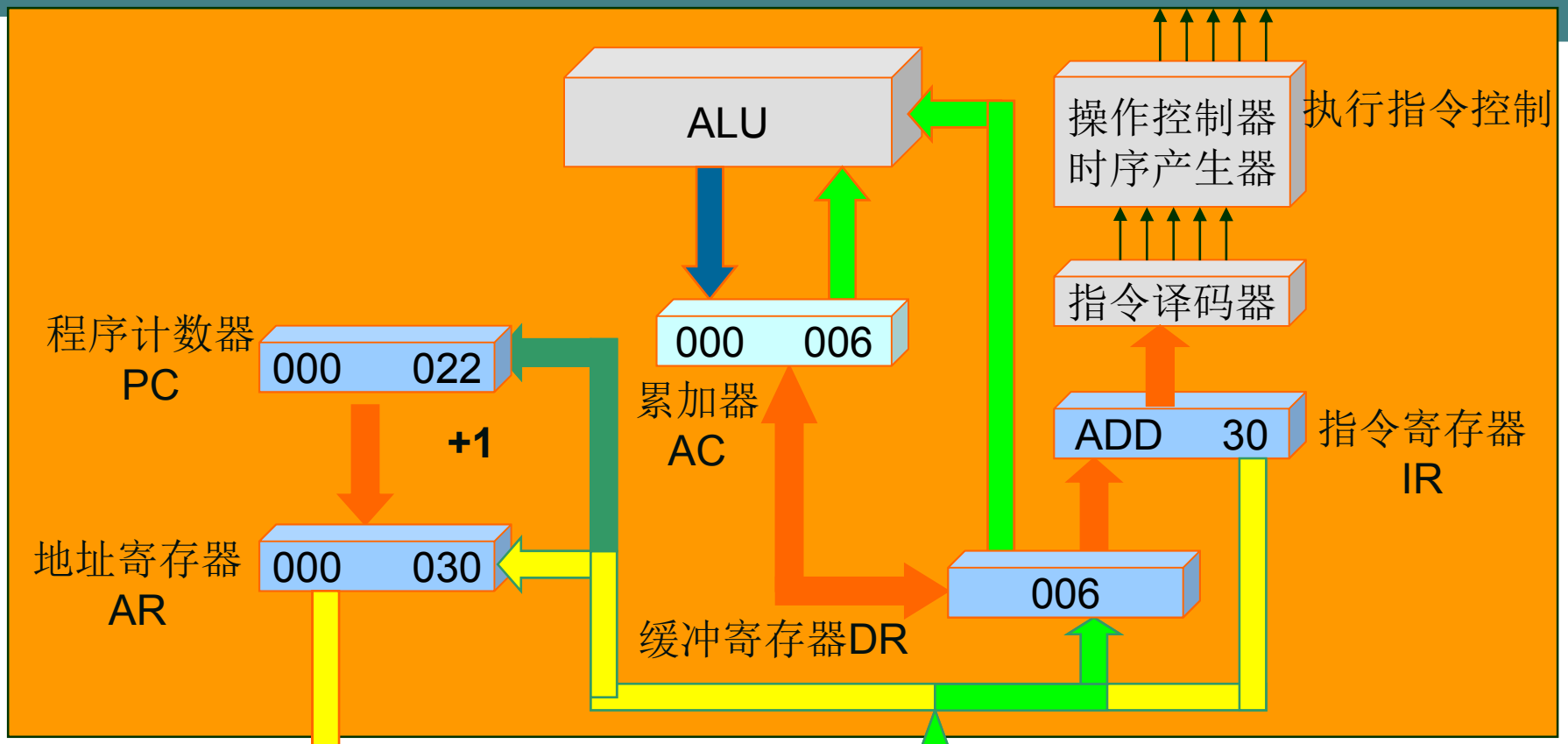
- PC→AR
- PC+1→PC
- AR →ABUS
- RAM→DBUS→DR
- DR→IR
- 操作码译码或测试（识别CLA指令，指令取值结束）

- 操作控制器送控制信号到ALU
- 0→AC（执行清零，指令执行结束）

ADD指令周期


1个CPU周期
1个CPU周期
1个CPU周期
开始
取指令
PC+1
指令译码
送操作数
地址
取操作数
执行
加法操作
取下条
指令PC+1
取指令阶段
执行指令阶段


ALU
操作控制器
时序产生器
执行指令控制
指令译码器
程序计数器
PC
000 022
+1
000000000
累加器
AC
ADD 30
指令寄存器
IR
地址寄存器
AR
000 021
ADD 30
缓冲寄存器DR
地址总线
ABUS
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 006
40 000 004
数据总线
DBUS
PC→AR→ABUS→RAM
→DBUS→DR→IR
PC+1
ADD指令



地址总线
ABUS

IR→AR（送OP地址）

→ABUS→RAM
→DBUS→DR
→ALU（执行相加）
ALU→AC

20	CLA	
21	ADD	30
22	STA	40
23	NOP	
24	JMP	21
	⋮	
30	000	006
	⋮	
40	000	004

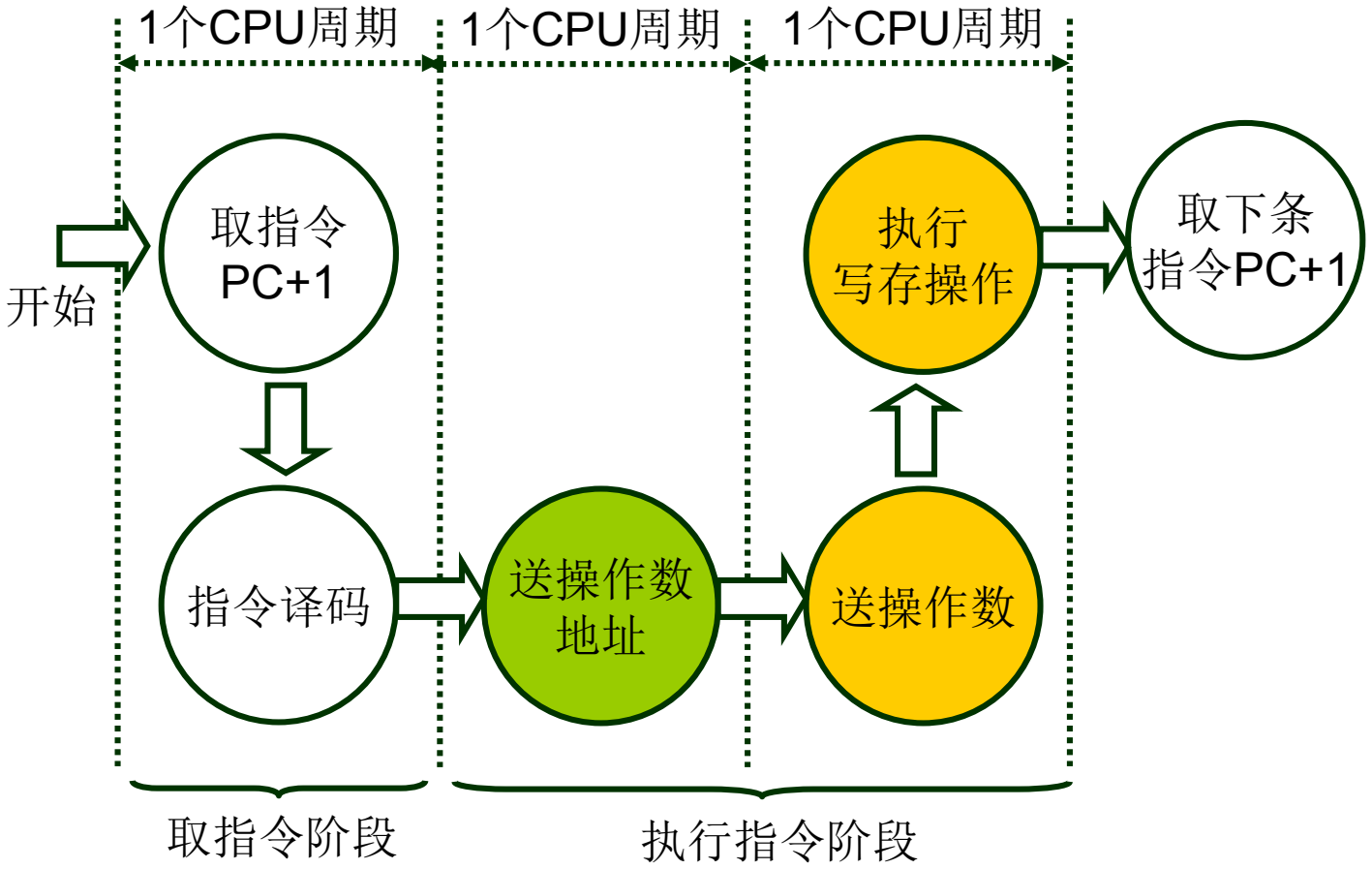
数据总线
DBUS

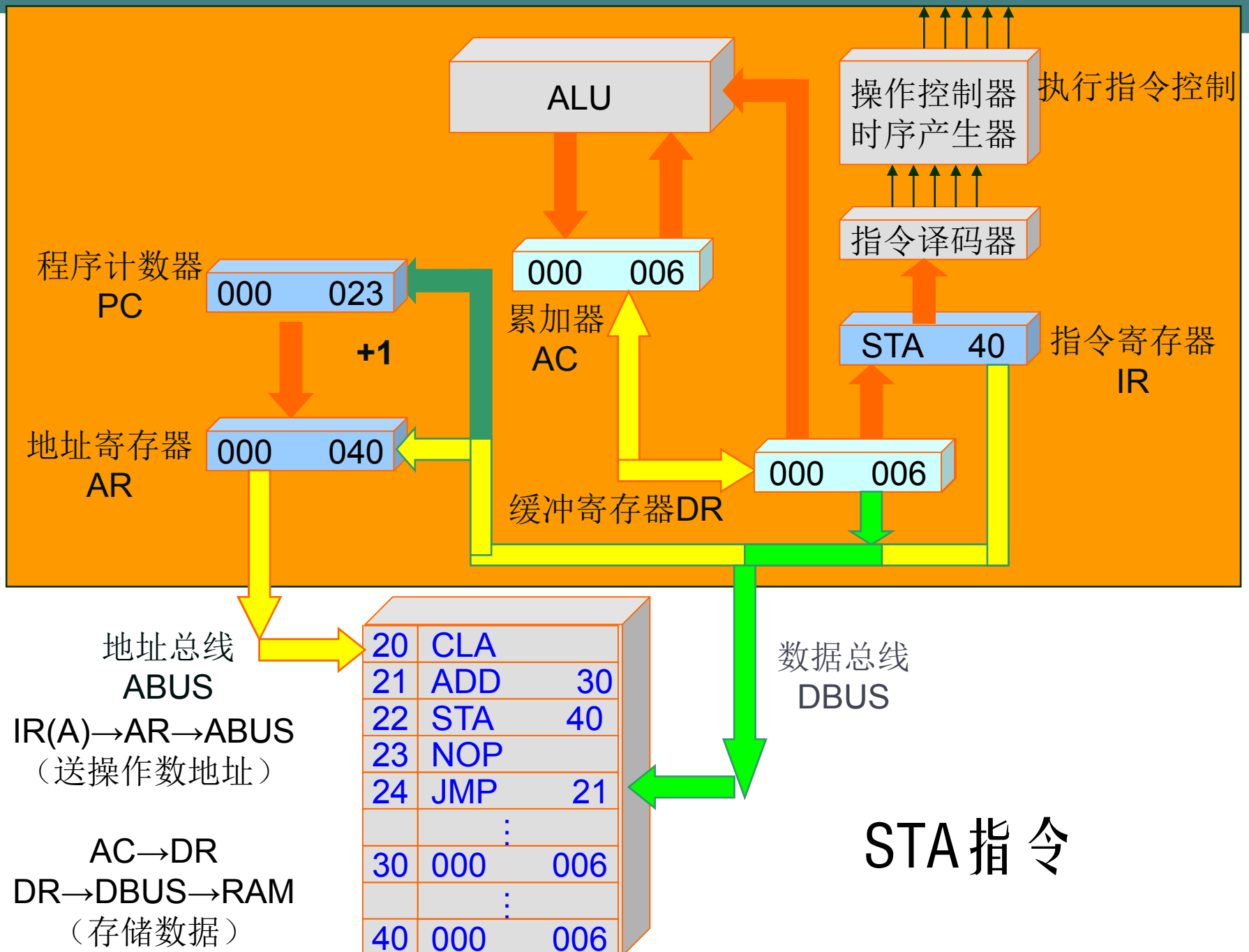
ADD指令

ADD执行过程的操作

- PC→AR
- PC+1→PC
- AR →ABUS→RAM→DBUS→DR
- DR→IR
- IR(A)→AR→ABUS→RAM
- →DBUS→DR→ALU
- ALU→AC

STA 40 指令周期

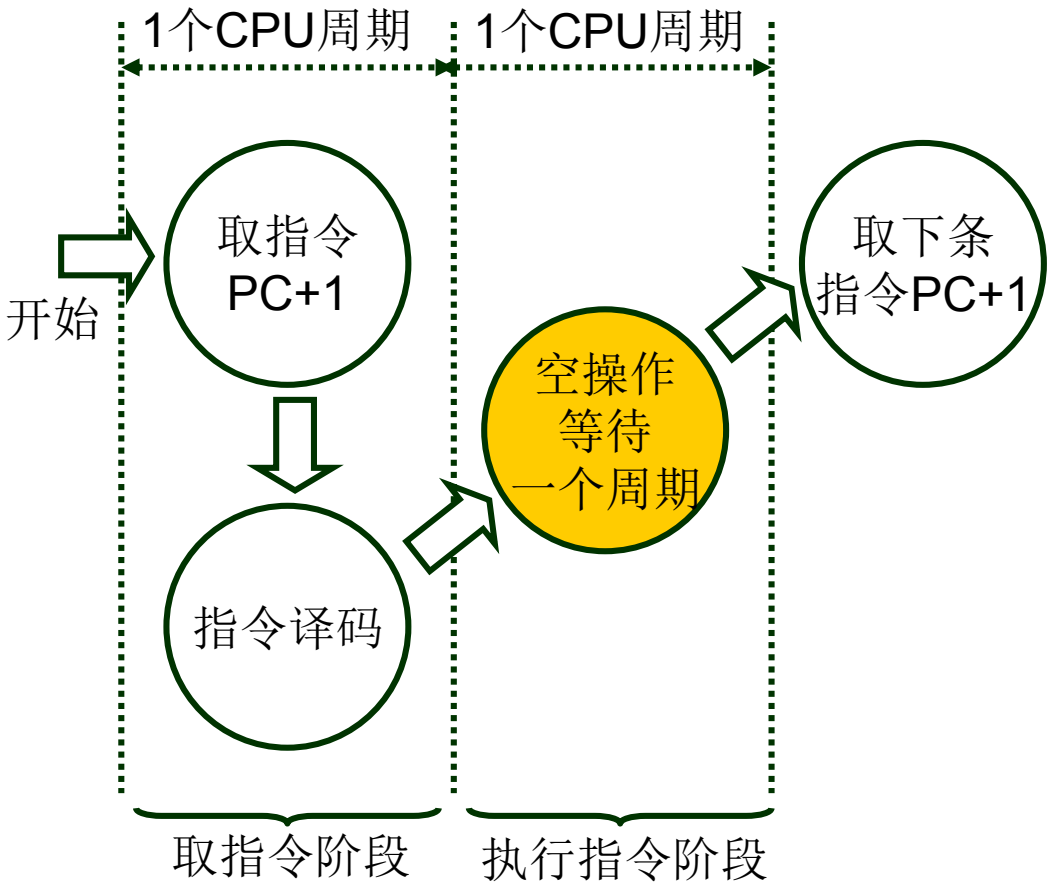

1个CPU周期
1个CPU周期
1个CPU周期
开始
取指令
PC+1
指令译码
送操作数
地址
执行
写存操作
送操作数
取下条
指令PC+1
取指令阶段
执行指令阶段


ALU
操作控制器
时序产生器
执行指令控制
指令译码器
程序计数器
PC
000 023
+1
000 006
累加器
AC
STA 40
指令寄存器
IR
地址寄存器
AR
000 040
000 006
缓冲寄存器DR
地址总线
ABUS
IR(A)→AR→ABUS
（送操作数地址）
AC→DR
DR→DBUS→RAM
（存储数据）
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 006
40 000 006
数据总线
DBUS
STA指令

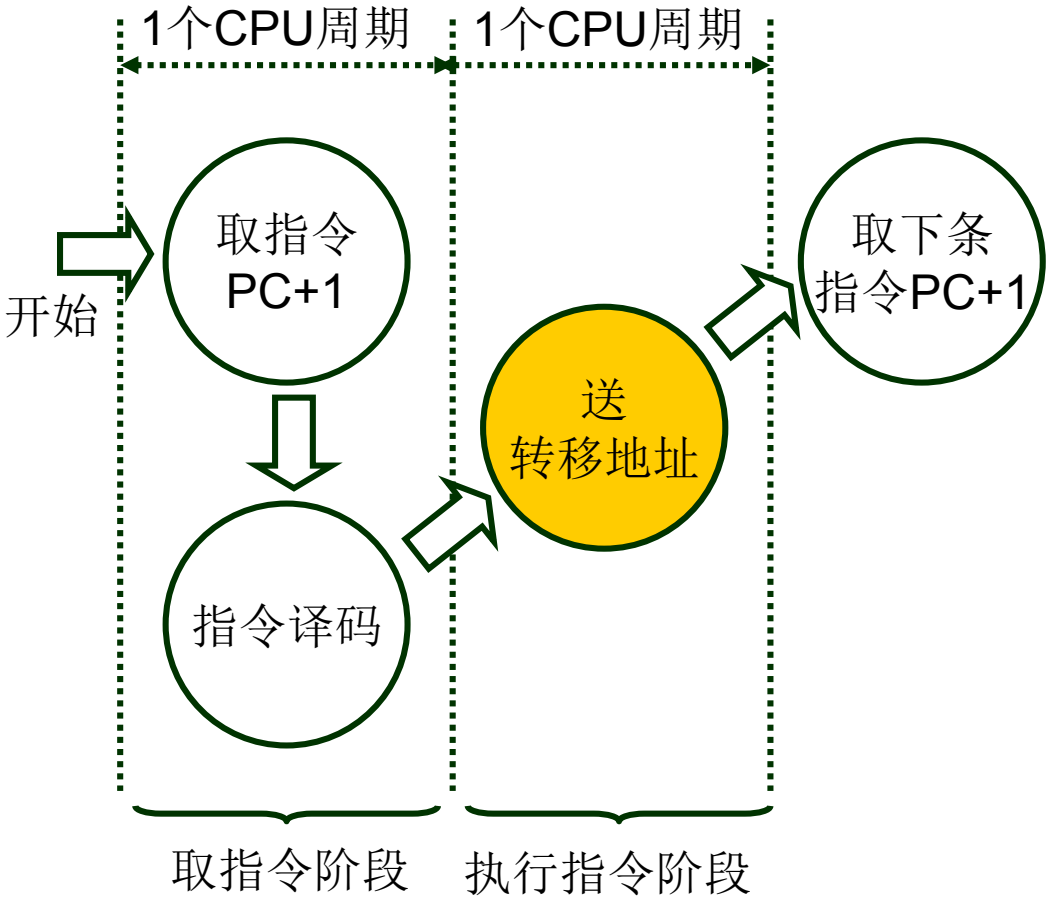
执行过程的操作

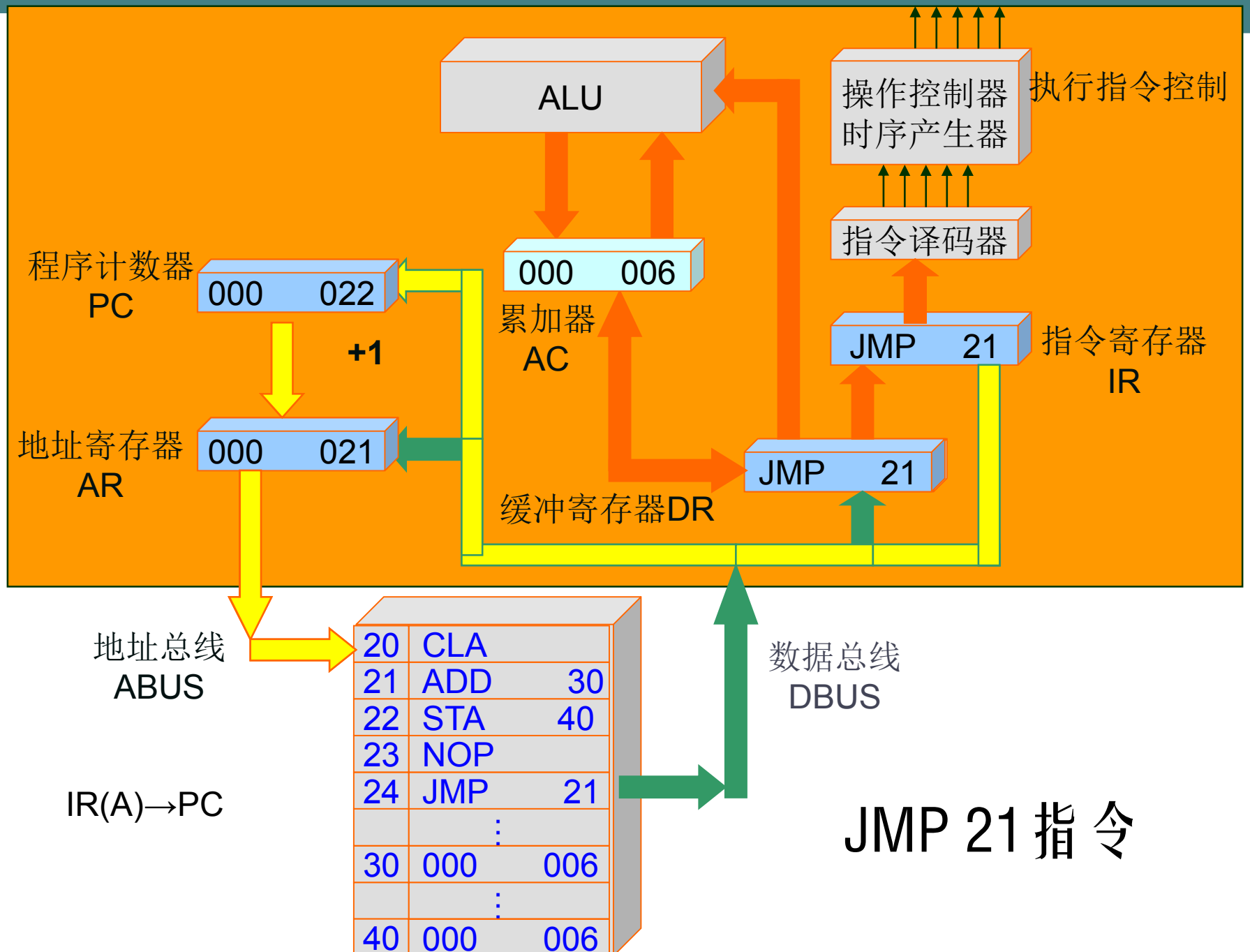
- PC→AR
- PC+1→PC
- AR →ABUS→RAM→DBUS→DR
- DR→IR
- IR(A)→AR→ABUS
- AC→DR
- DR→DBUS→RAM

NOP指令周期


1个CPU周期
1个CPU周期
开始
取指令
PC+1
指令译码
空操作
等待
一个周期
取下条
指令PC+1
取指令阶段
执行指令阶段

JMP 21指令周期

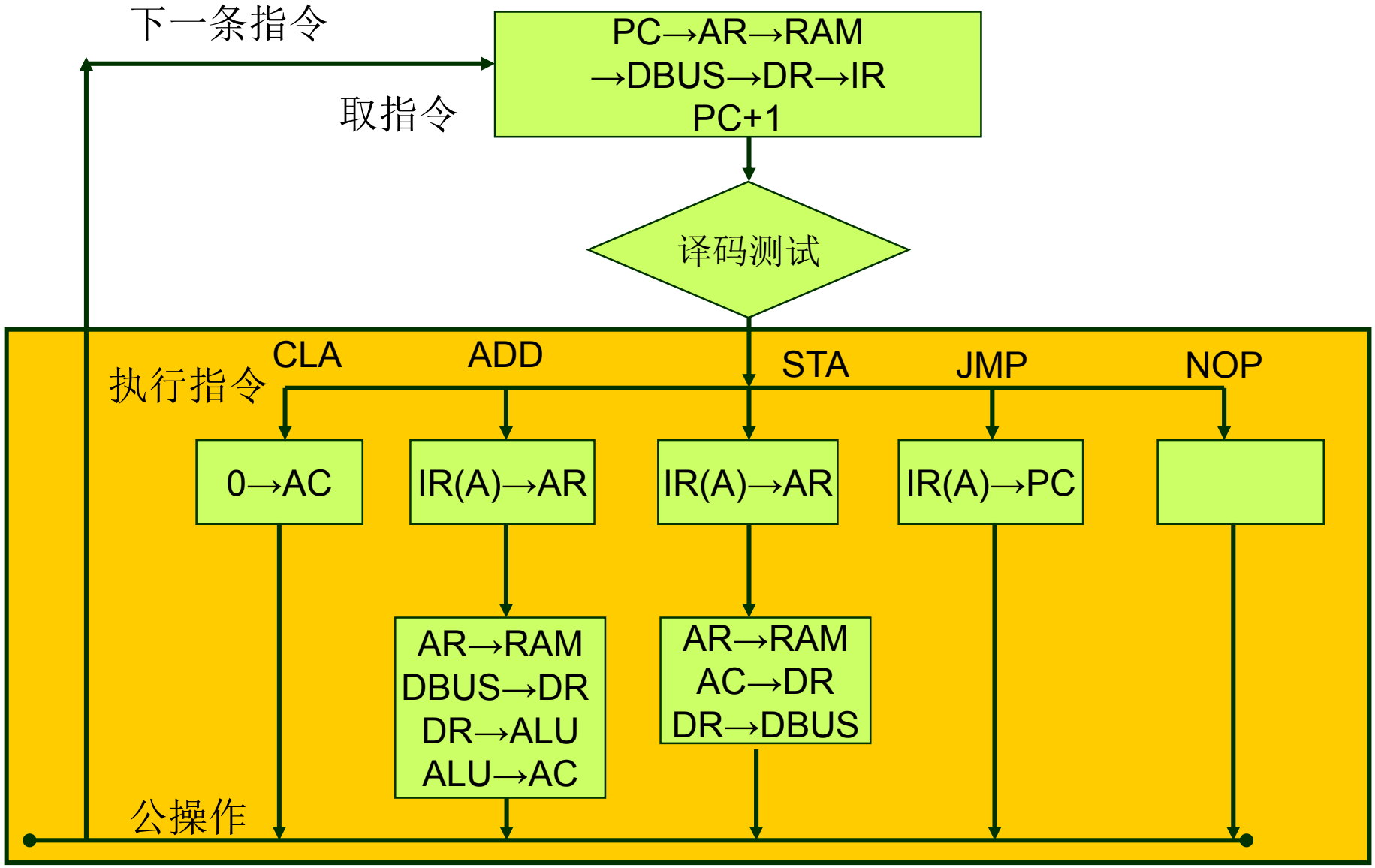

1个CPU周期
1个CPU周期
开始
取指令
PC+1
指令译码
送
转移地址
取下条
指令PC+1
取指令阶段
执行指令阶段


ALU
操作控制器
时序产生器
执行指令控制
指令译码器
程序计数器
PC
000 022
+1
累加器
AC
000 006
JMP 21
指令寄存器
IR
地址寄存器
AR
000 021
JMP 21
缓冲寄存器DR
地址总线
ABUS
数据总线
DBUS
20 CLA
21 ADD 30
22 STA 40
23 NOP
24 JMP 21
30 000 006
40 000 006
IR(A)→PC
JMP 21 指令

执行过程中的操作

- PC→AR
- PC+1→PC
- AR →ABUS→RAM→DBUS→DR
- DR→IR
- IR(A)→PC
- Next command

方框图表示


下一条指令
取指令
PC→AR→RAM
→DBUS→DR→IR
PC+1
译码测试
执行指令
CLA
ADD
STA
JMP
NOP
0→AC
IR(A)→AR
IR(A)→AR
IR(A)→PC
AR→RAM
DBUS→DR
DR→ALU
ALU→AC
AR→RAM
AC→DR
DR→DBUS
公操作

公操作

- 一条指令执行完后，CPU所进行的一些操作。
- 对外设请求的处理（中断，通道）
- 若无外设请求的处理，CPU则转而取下条指令。
- 由于取指令是每条指令都有的，所以，取指令也是公操作。

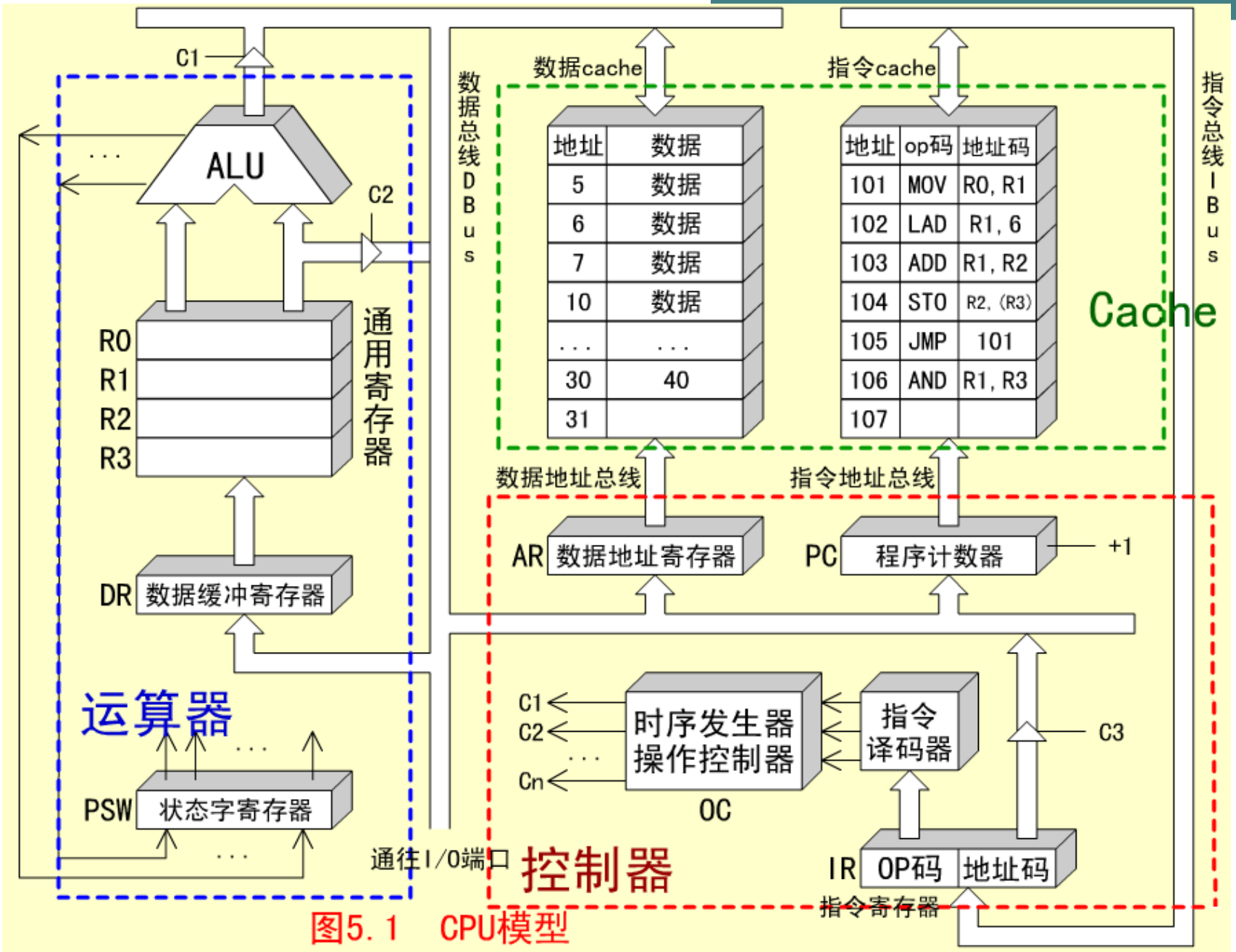
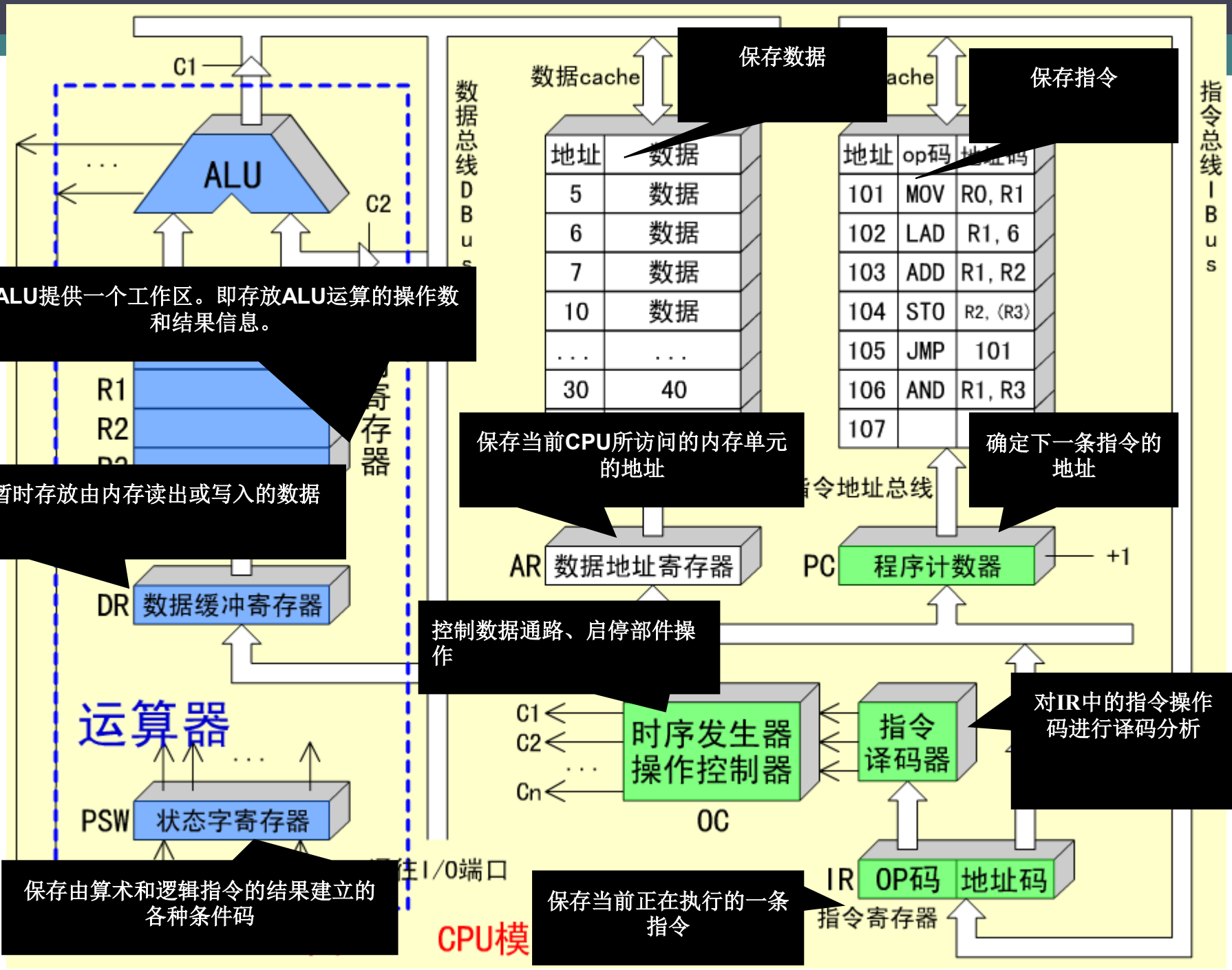

C1
ALU
C2
R0
R1
R2
R3
通用寄存器
DR
数据缓冲寄存器
运算器
PSW
状态字寄存器
通往I/O端口
数据总线DBus
数据cache
地址
数据
5
数据
6
数据
7
数据
10
数据
30
40
31
指令cache
地址
op码
地址码
101
MOV
R0, R1
102
LAD
R1, 6
103
ADD
R1, R2
104
STO
R2, (R3)
105
JMP
101
106
AND
R1, R3
107
Cache
指令总线IBus
数据地址总线
指令地址总线
AR
数据地址寄存器
PC
程序计数器
+1
C1
C2
Cn
时序发生器
操作控制器
OC
指令
译码器
C3
IR
OP码
地址码
指令寄存器
控制器

图5.1 CPU模型


C1
ALU
C2
为ALU提供一个工作区。即存放ALU运算的操作数和结果信息。
R1
R2
暂时存放由内存读出或写入的数据
DR 数据缓冲寄存器
运算器
PSW 状态字寄存器
保存由算术和逻辑指令的结果建立的各种条件码
数据总线DBus
数据cache
保存数据
地址 数据
5 数据
6 数据
7 数据
10 数据
30 40
保存当前CPU所访问的内存单元的地址
AR 数据地址寄存器
控制数据通路、启停部件操作
C1
C2
Cn
时序发生器
操作控制器
OC
cache
保存指令
地址 op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
指令地址总线
确定下一条指令的地址
PC 程序计数器
+1
指令译码器
对IR中的指令操作码进行译码分析
IR OP码 地址码
指令寄存器
保存当前正在执行的一条指令
指令总线IBus
往I/O端口
CPU模

5条典型指令构成的简单程序

指令地址　指令助记符

```
101    MOV  R0, R1     ;  （R1）→R0
102    LAD  R1, 6      ;  （6）→ R1
103    ADD  R1, R2     ;  （R1）+（R2）→R2
104    STO  R2, (R3)   ;  R2→( R3 )
105    JMP  101        ;  101 → PC
106    AND  R1, R3
       … …
  6    100
```

MOV指令的指令周期

MOV R0, R1是一条RR指令

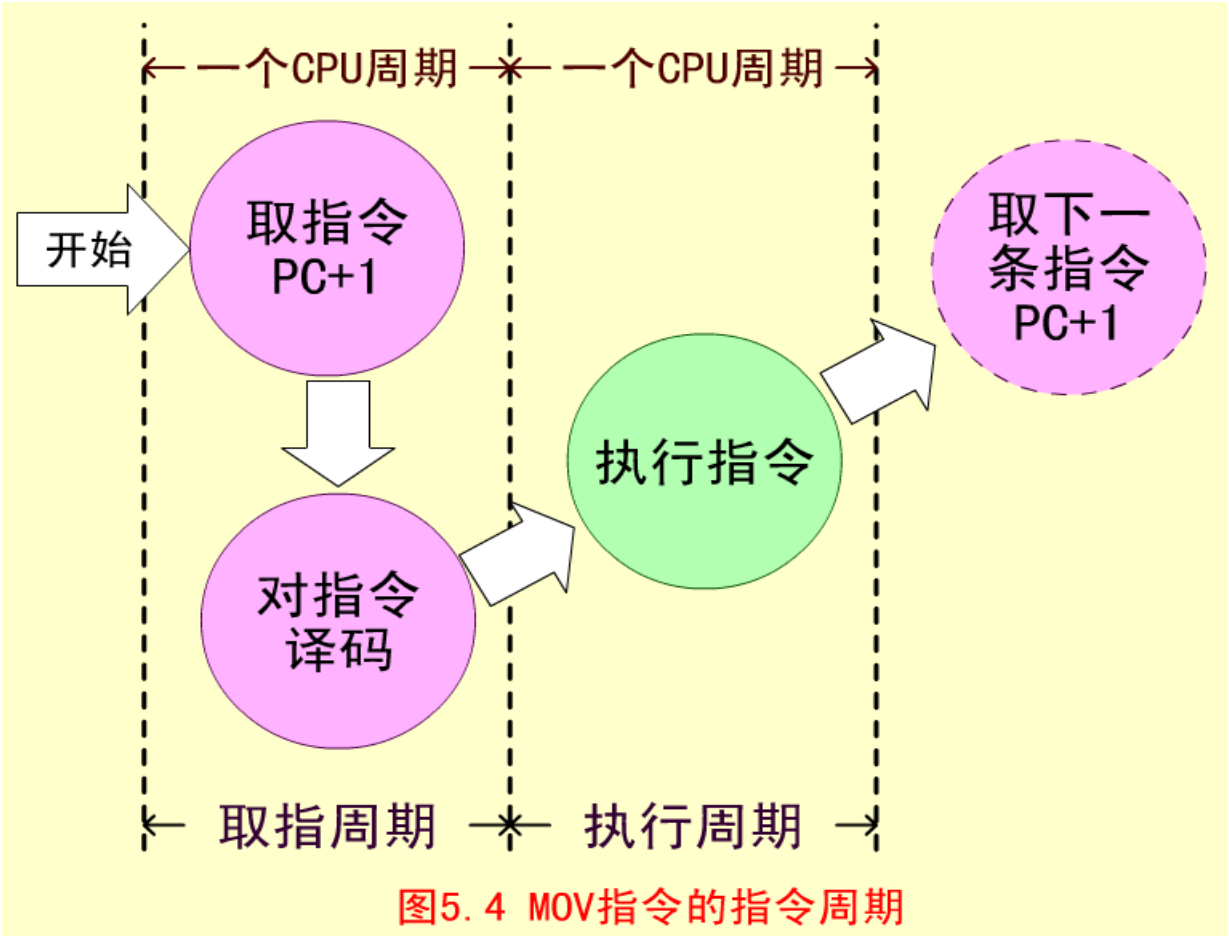
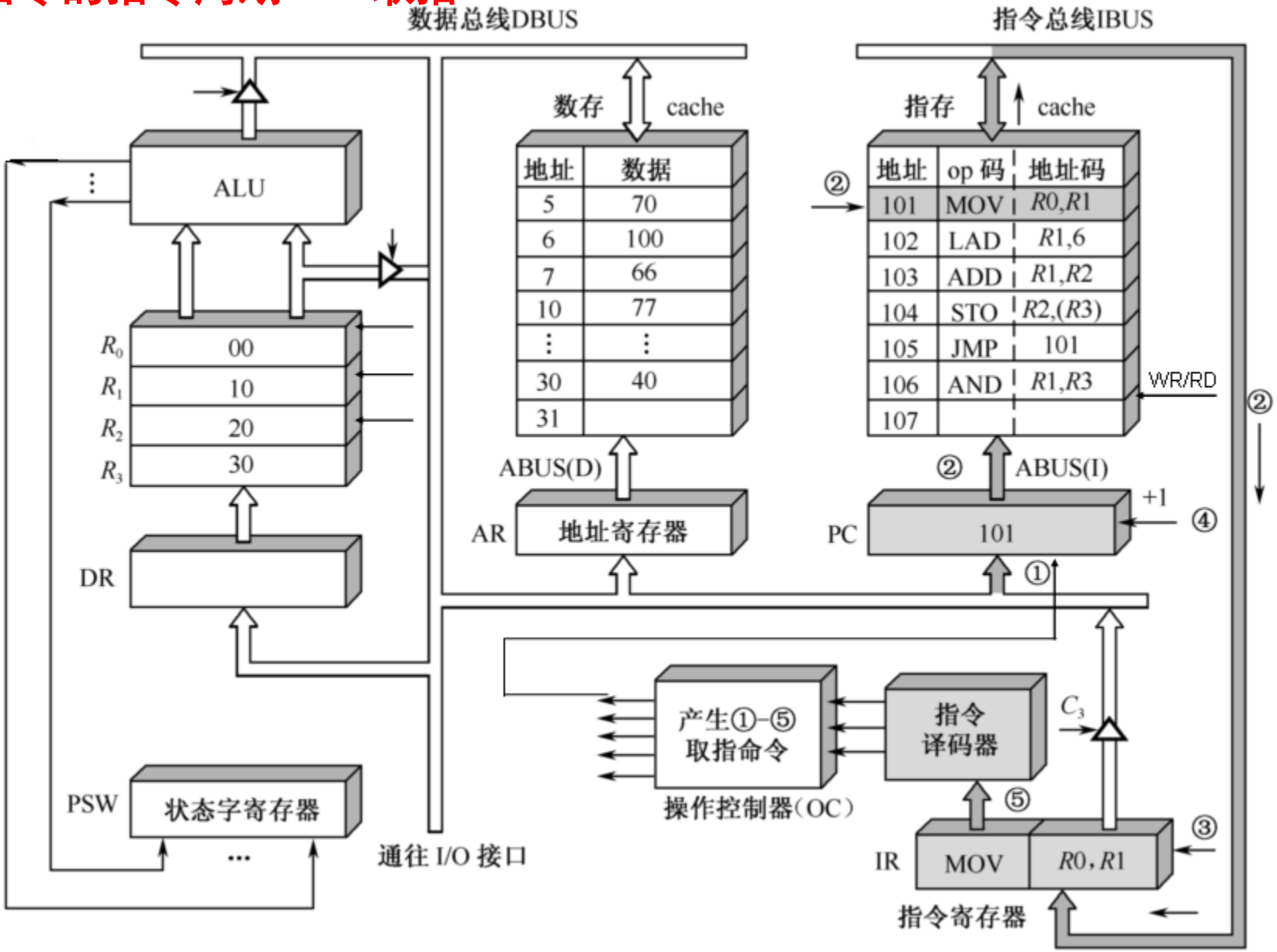
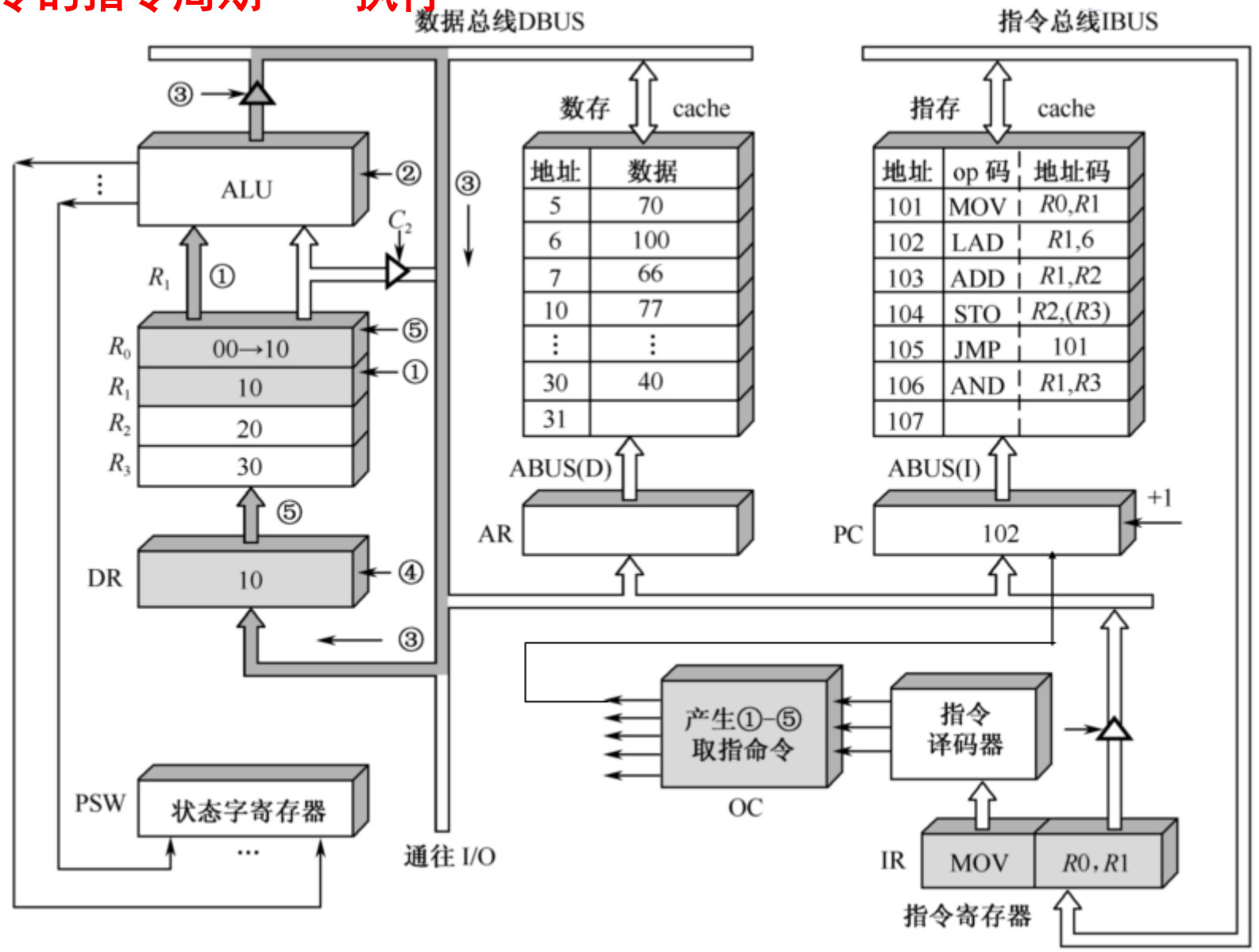


图5.4 MOV指令的指令周期

MOV指令的指令周期——取指

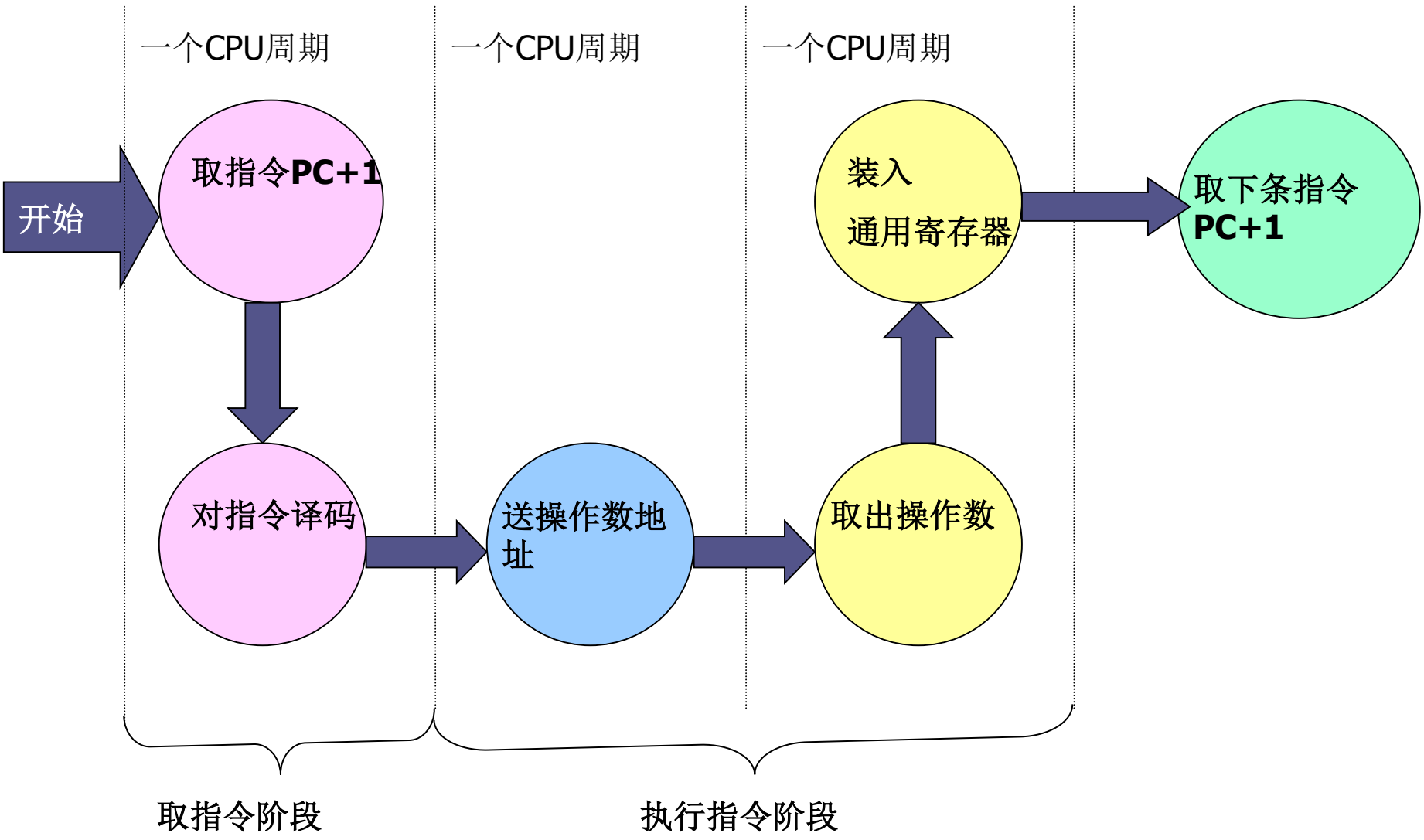


MOV指令的指令周期——执行

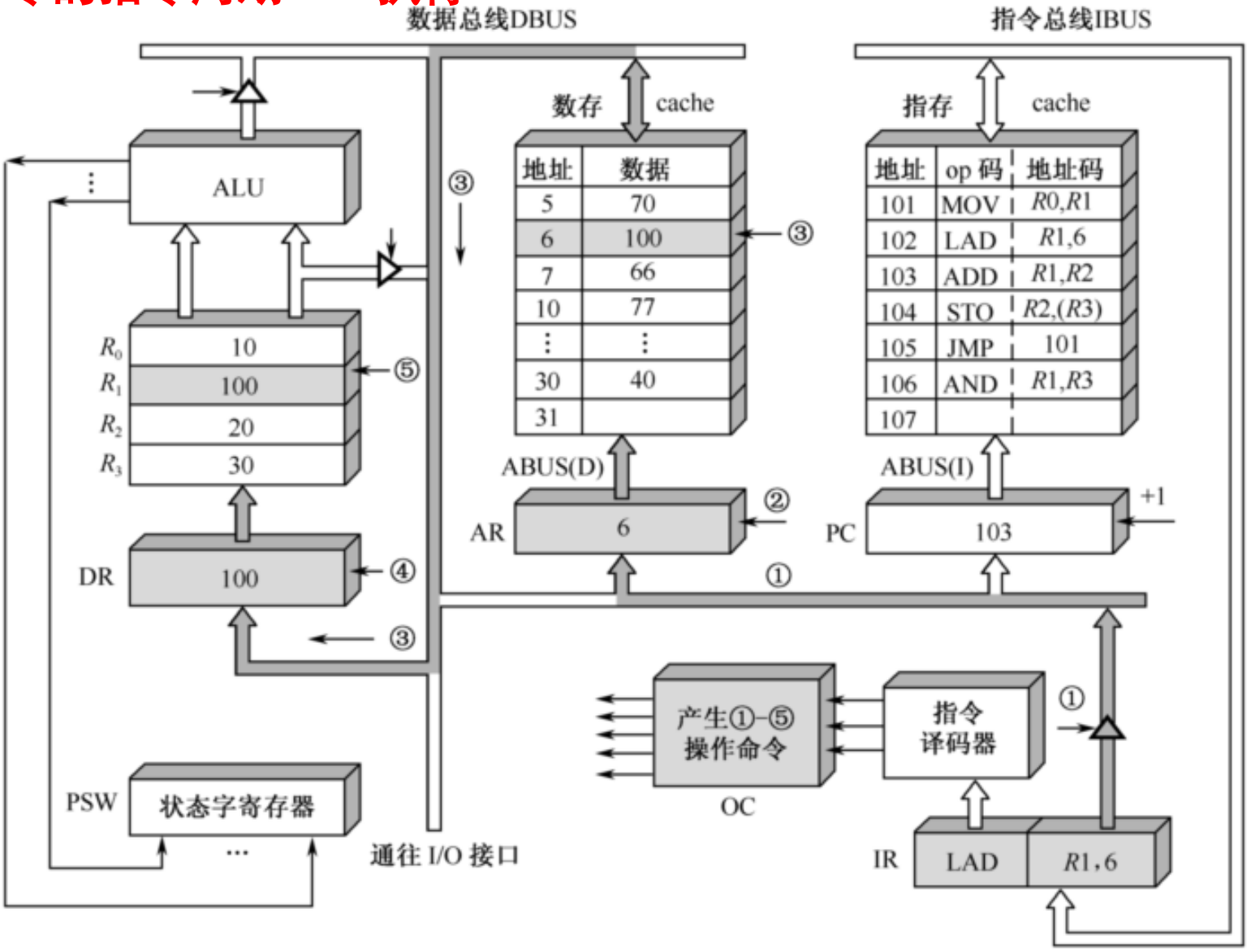


LAD指令的指令周期

LAD R1, 6是一条RS指令

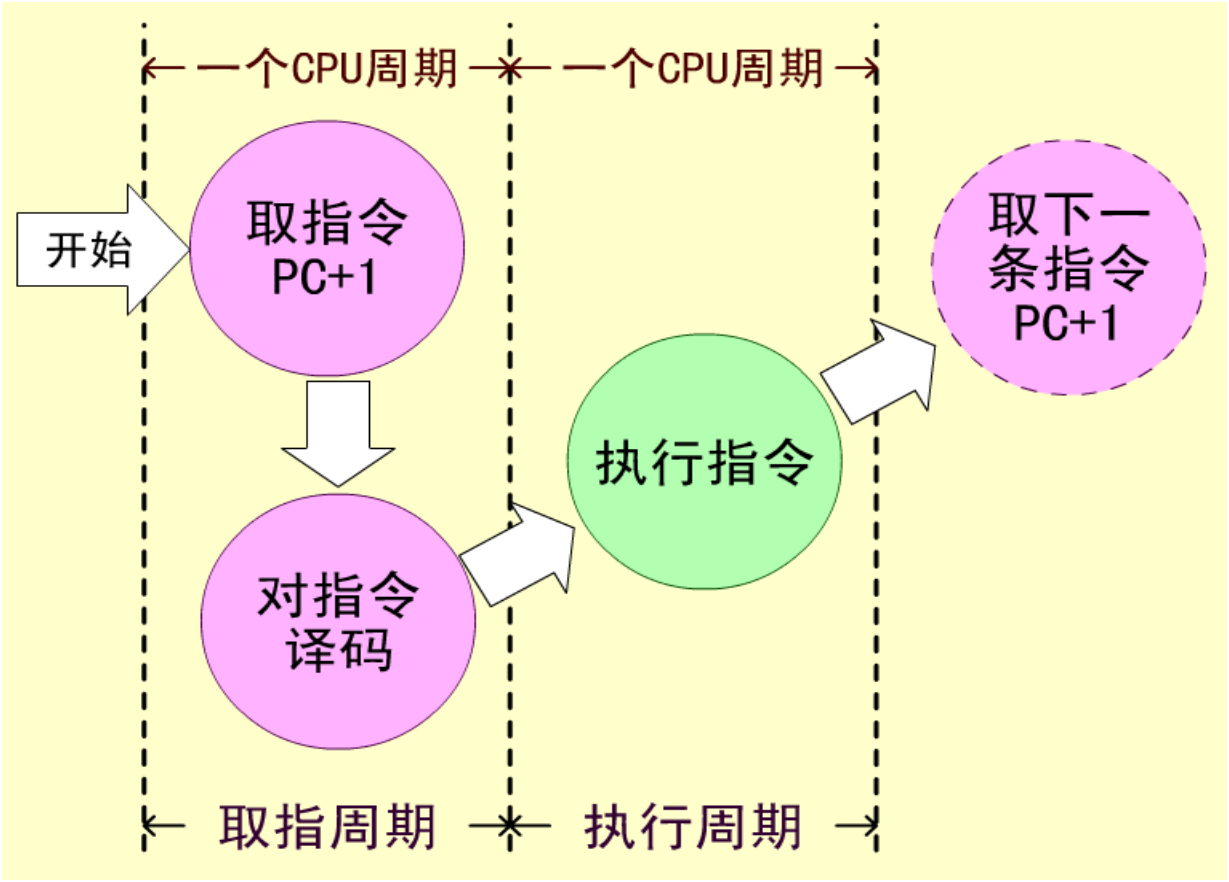


LAD指令的指令周期——执行

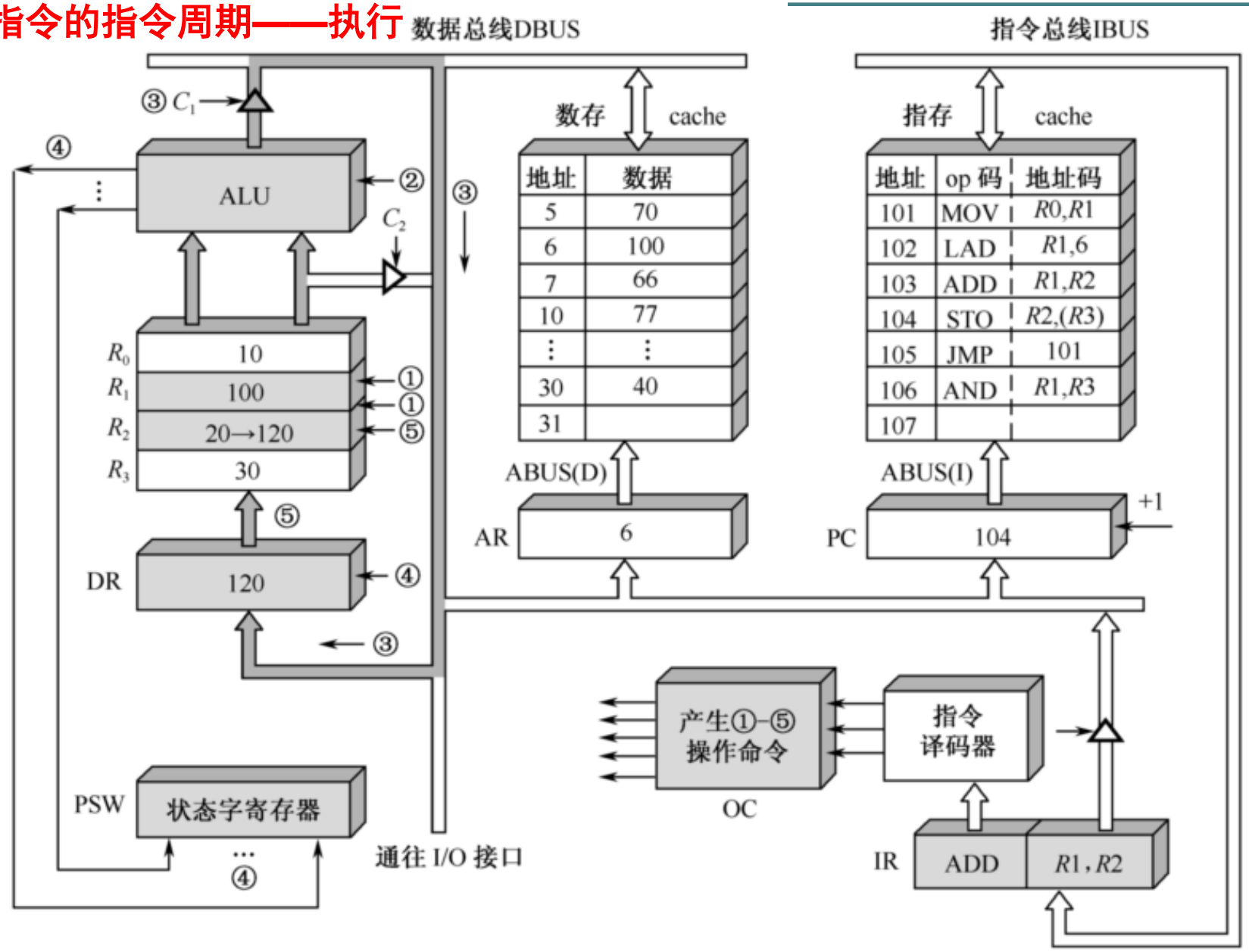


ADD指令的指令周期

ADD　R1, R2是一条RR指令

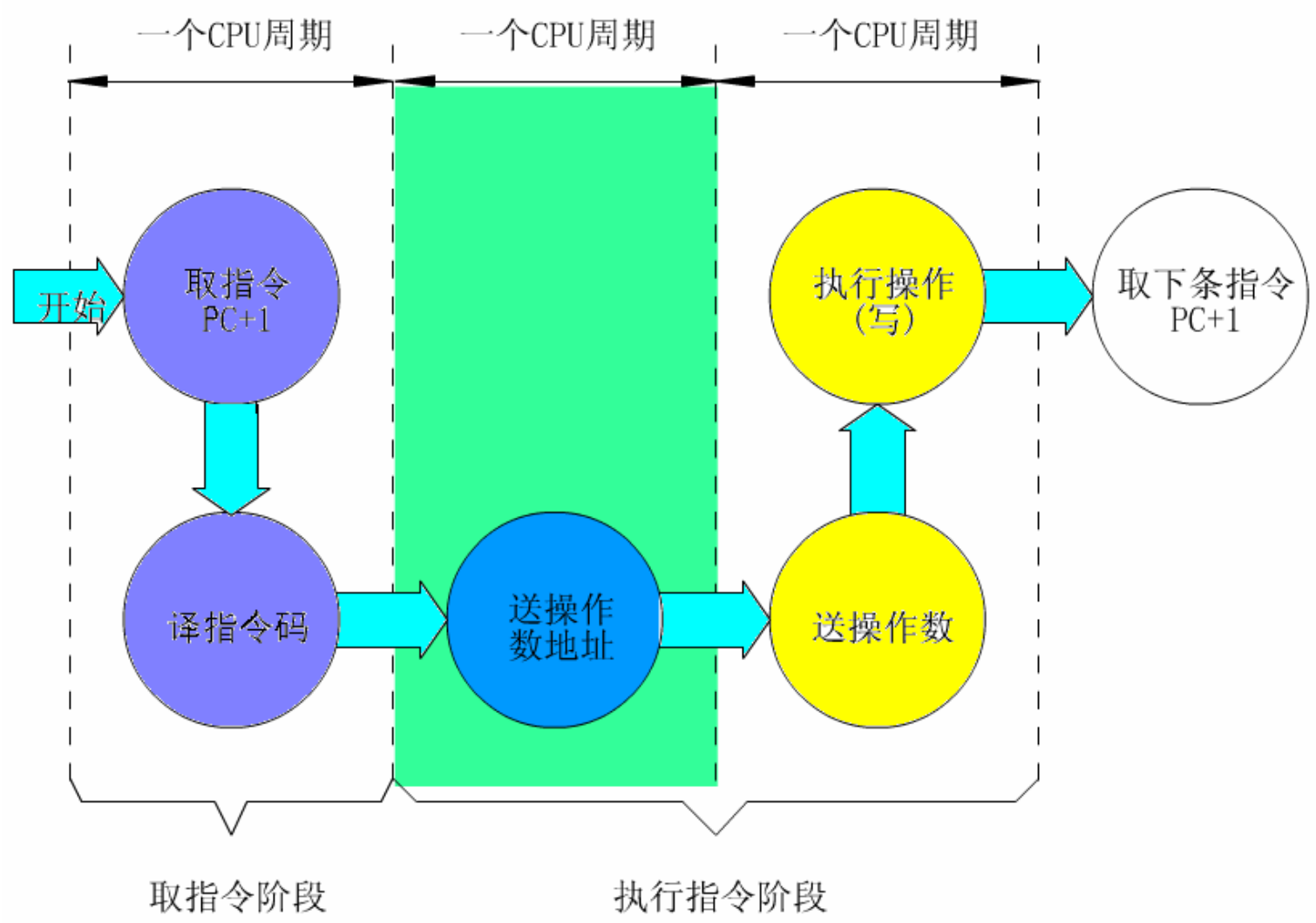


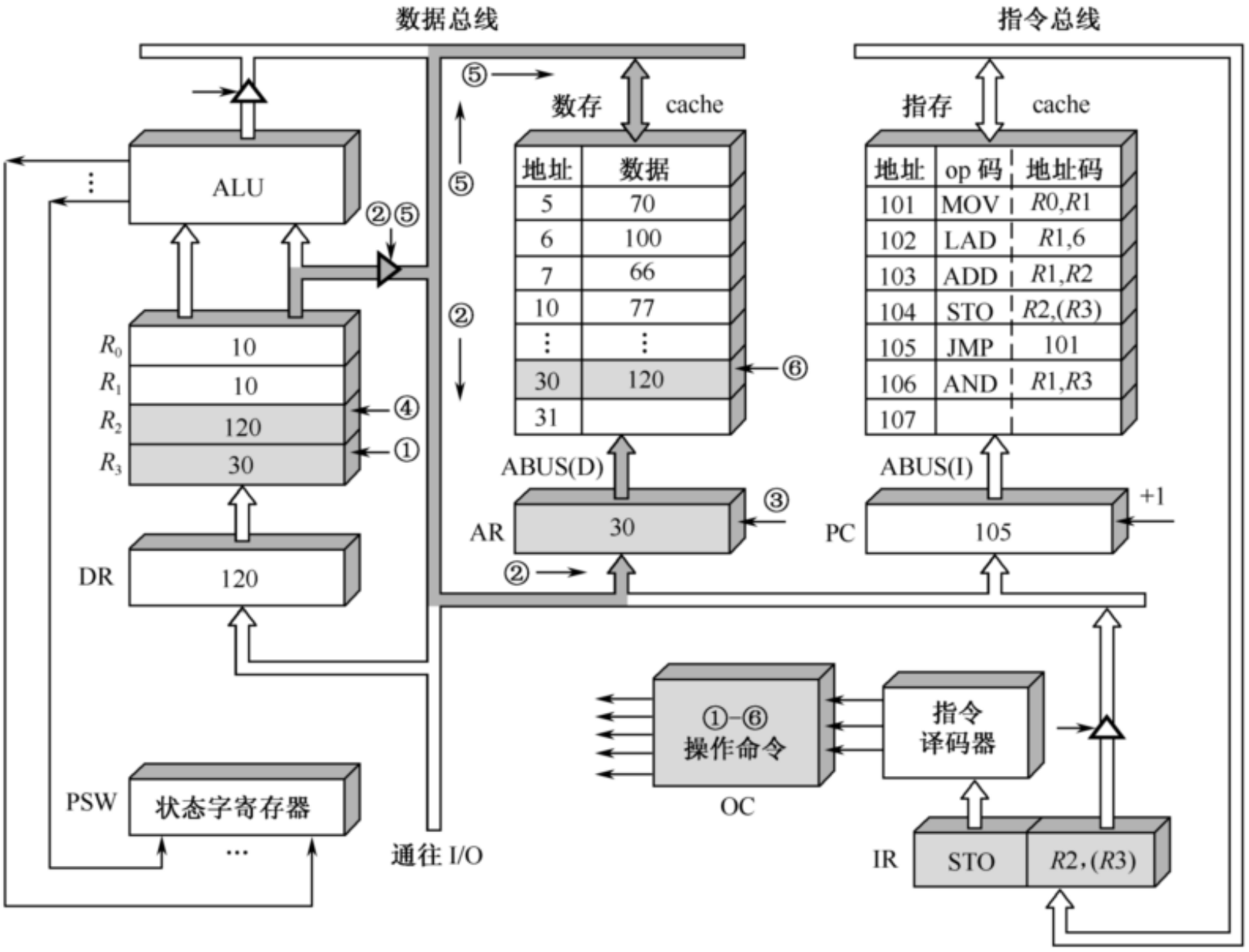
ADD指令的指令周期——执行



STO指令的指令周期

STO　R2, (R3)是一条RS指令

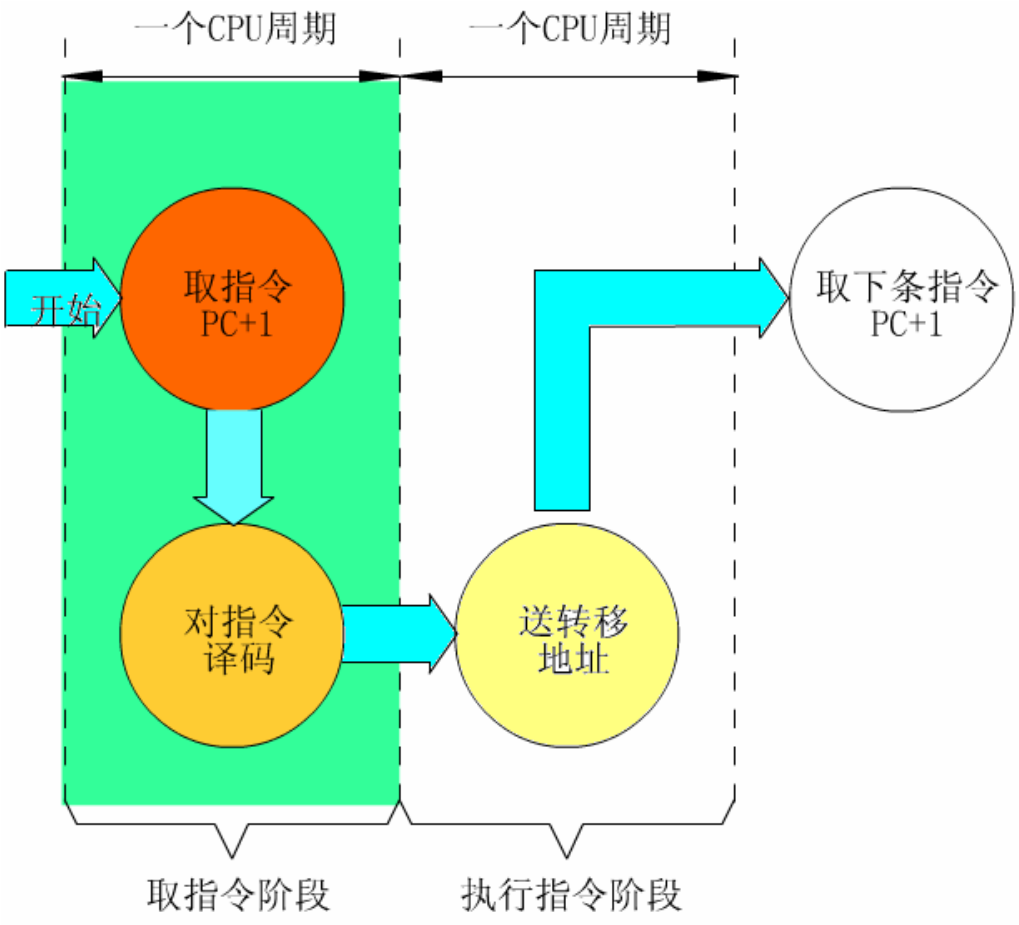


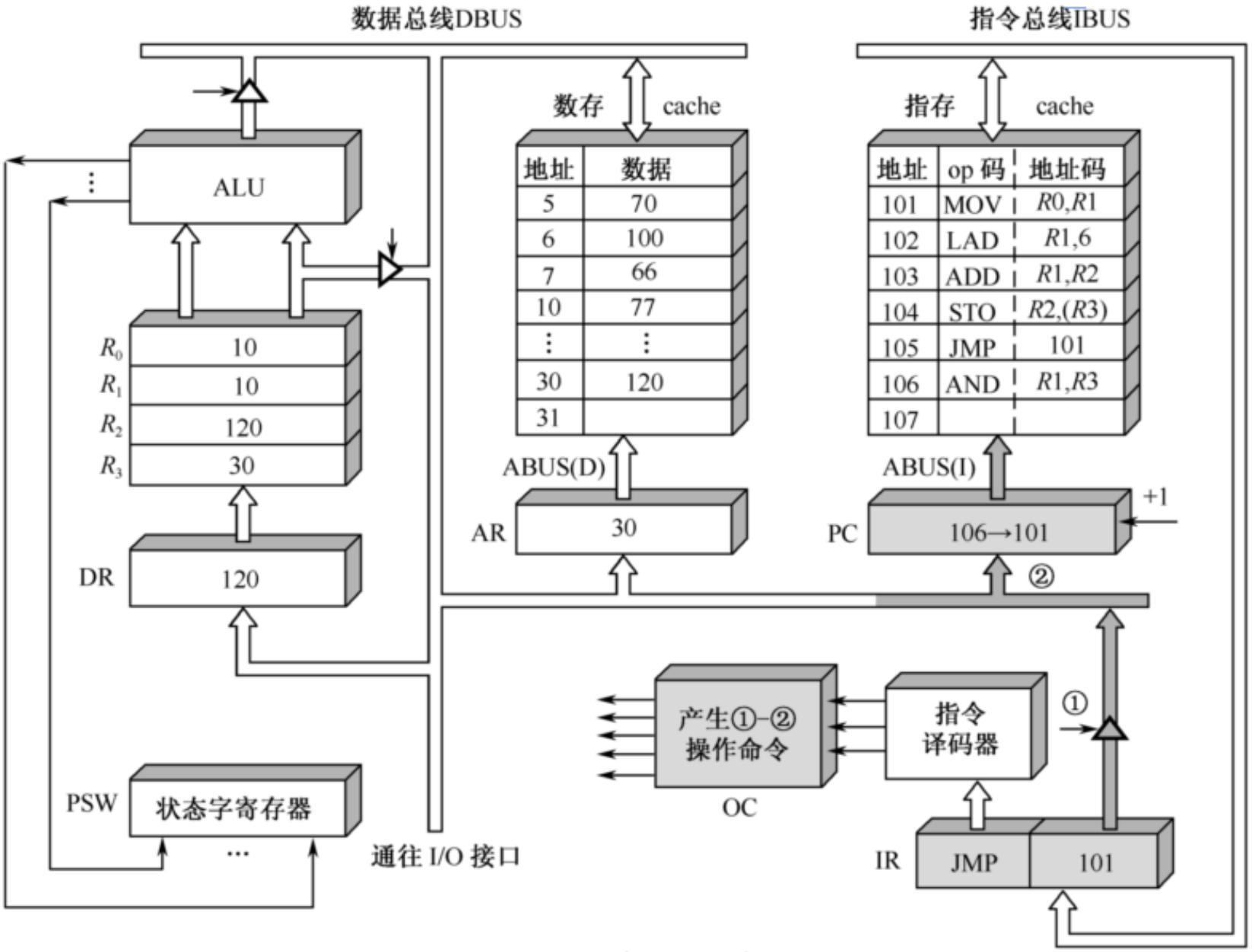


5.2.5 STO指令的指令周期

JMP指令的指令周期

JMP 101





5.2.6 JMP指令的指令周期

用方框图语言表示指令周期

- **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。时间上依附于紧接的前面一个CPU周期，而不单独占用一个CPU周期。

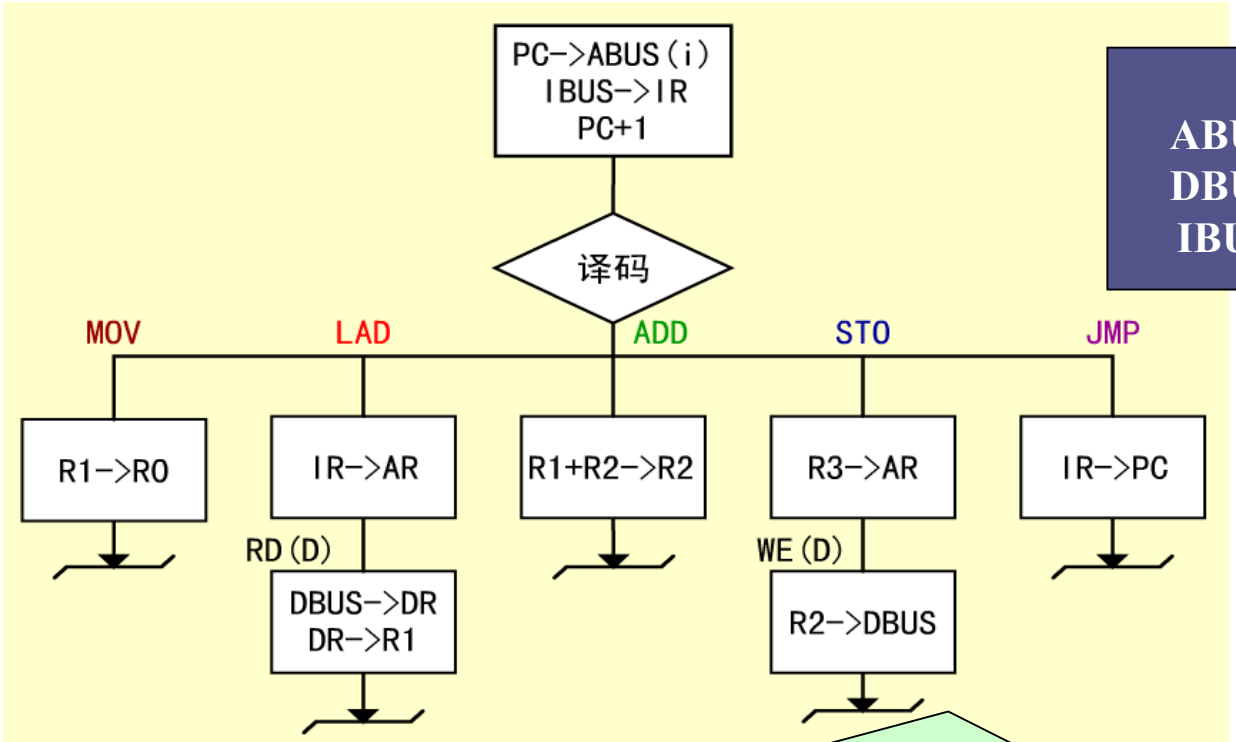
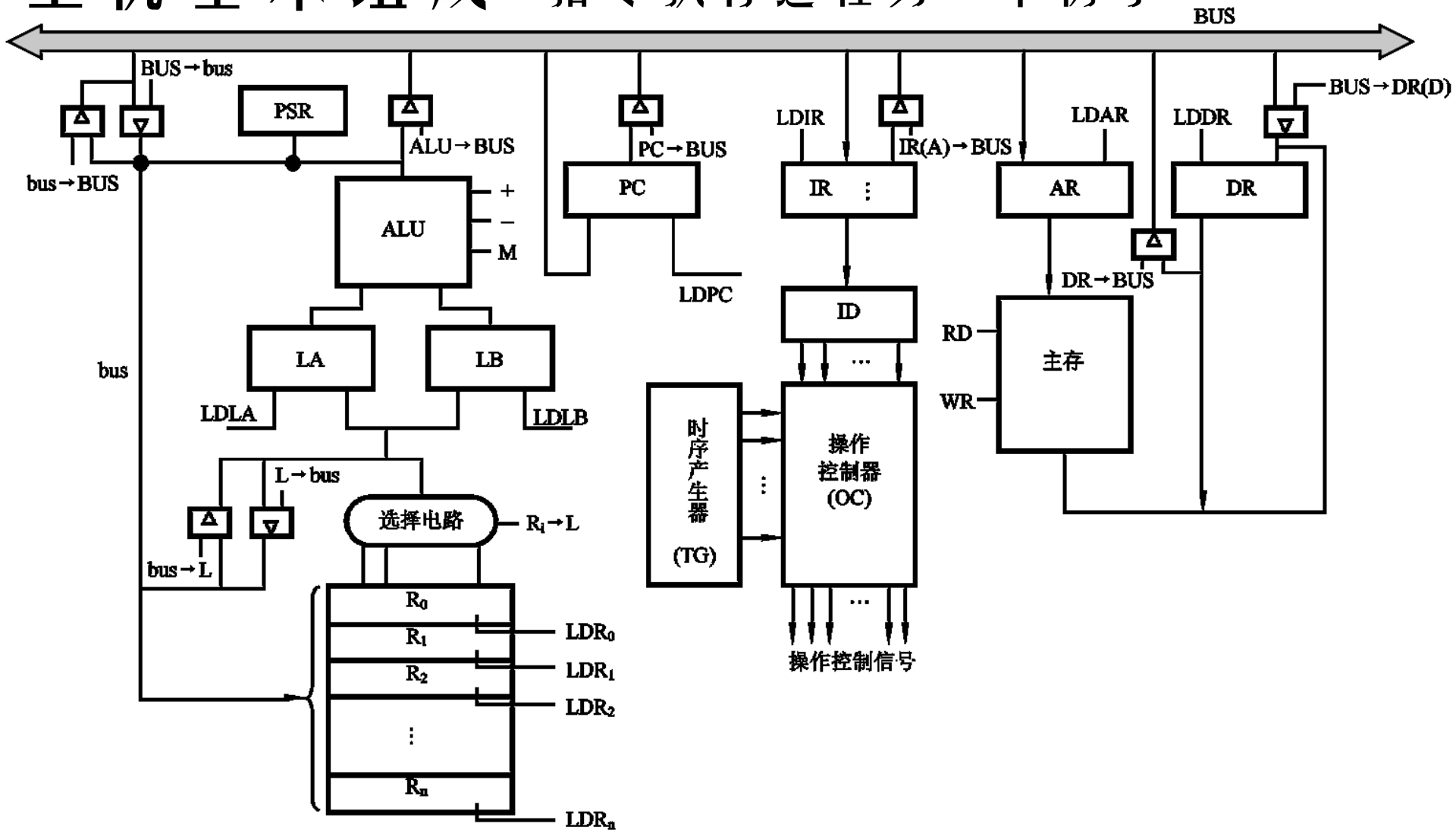


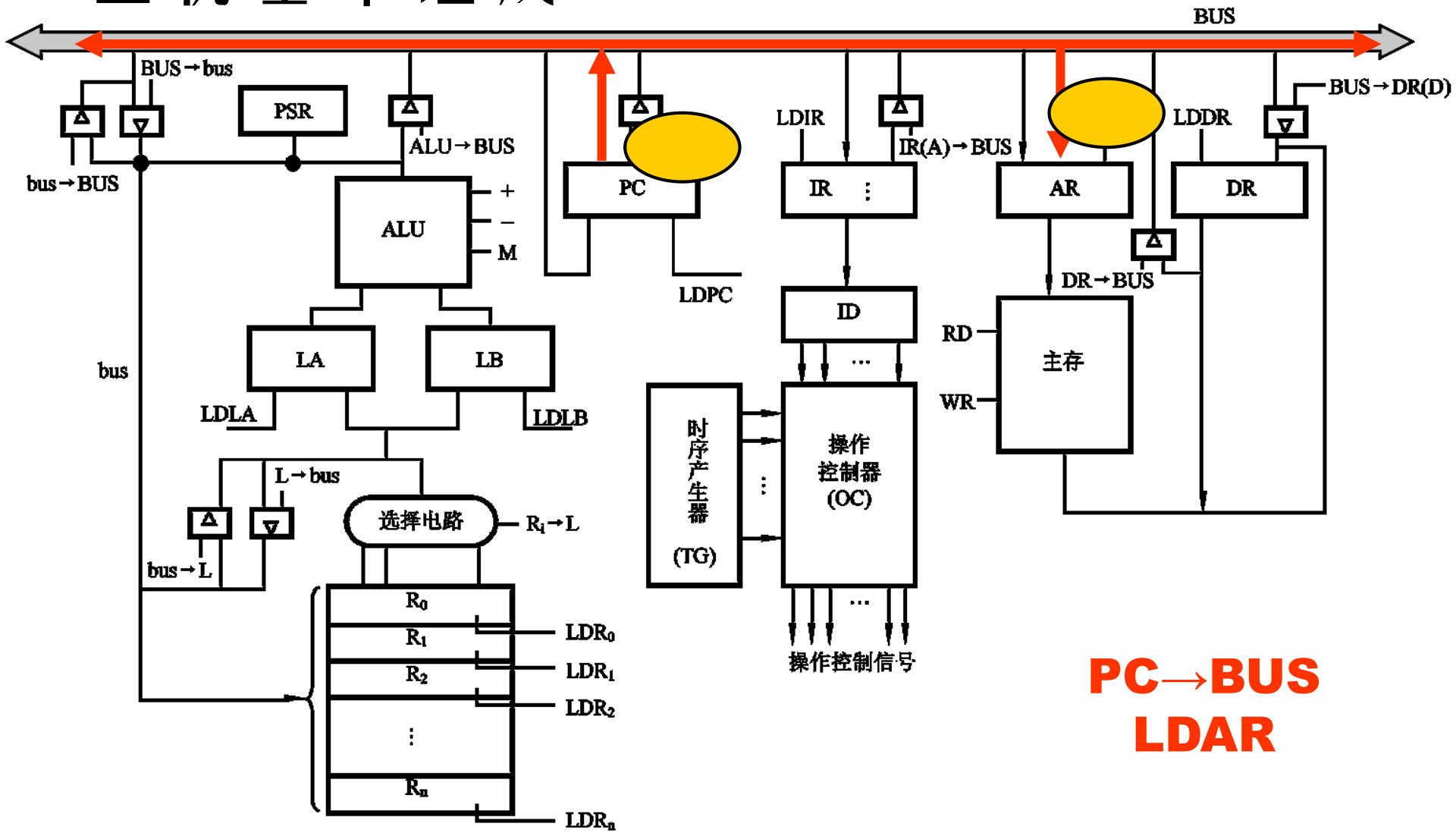
图5.14 用方框图语言

主机基本组成--指令执行过程另一个例子



主机基本组成

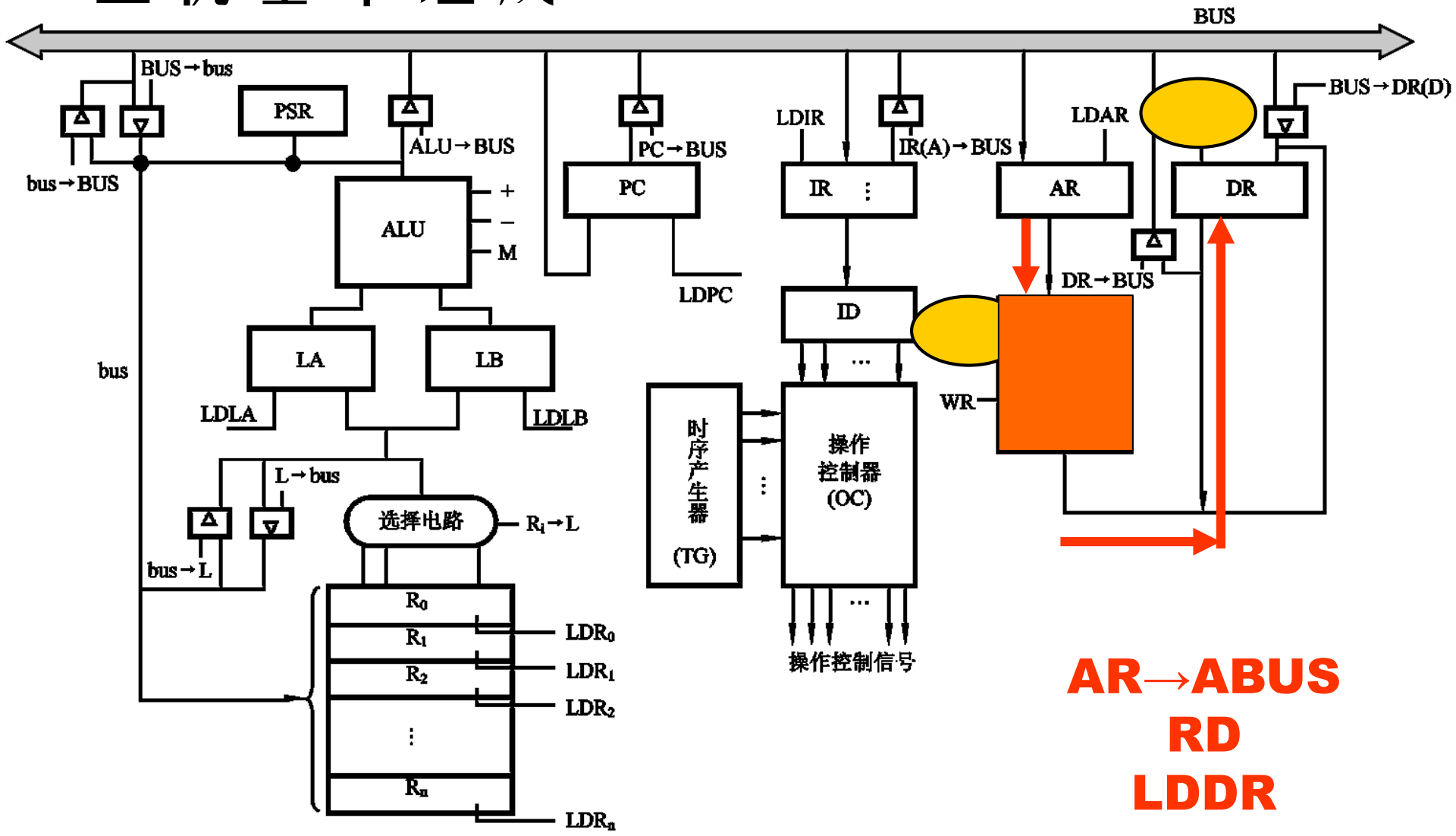
PC→AR



PC→BUS
LDAR

主机基本组成

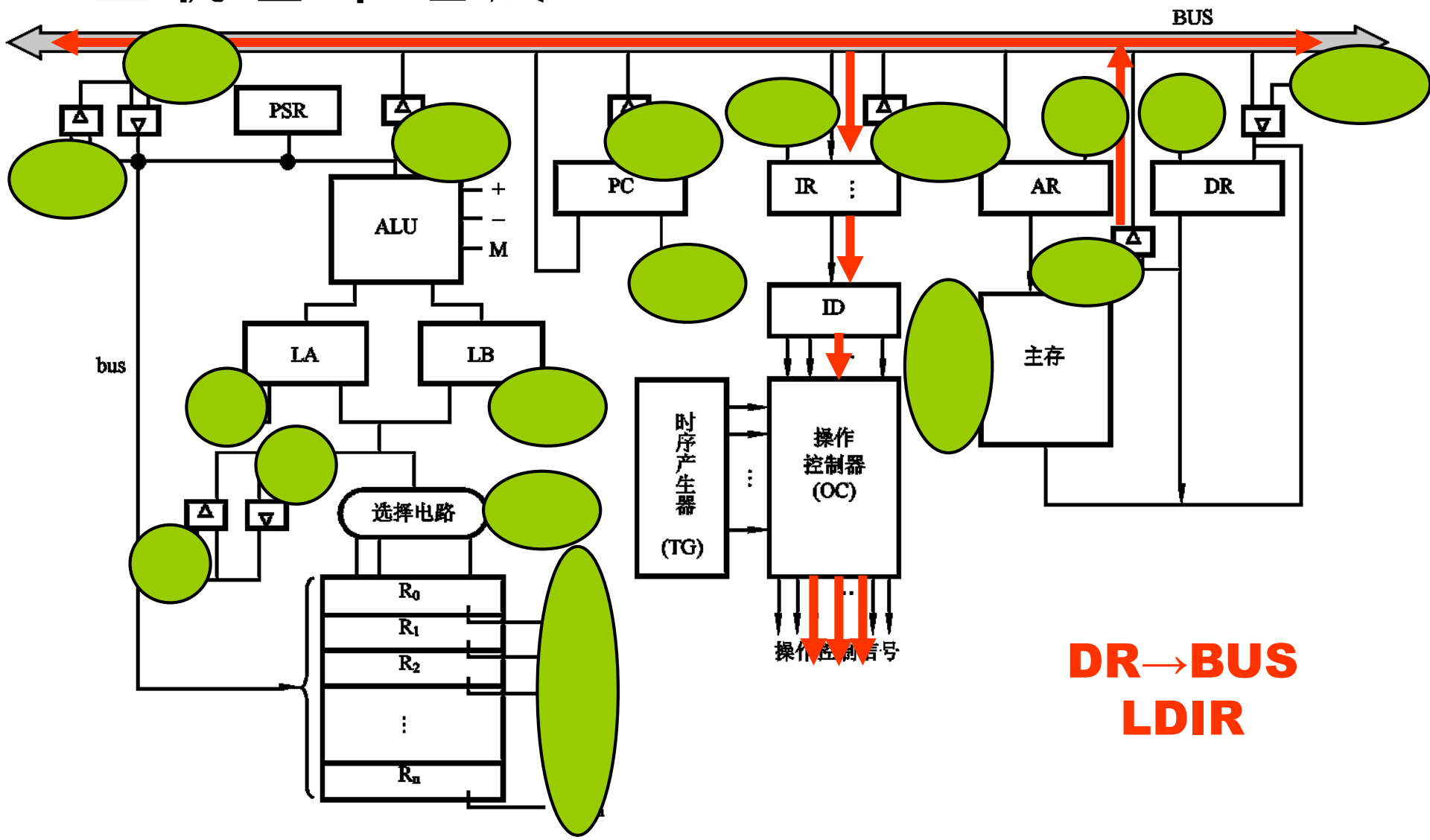
MEM →DR



AR→ABUS
RD
LDDR

主机基本组成

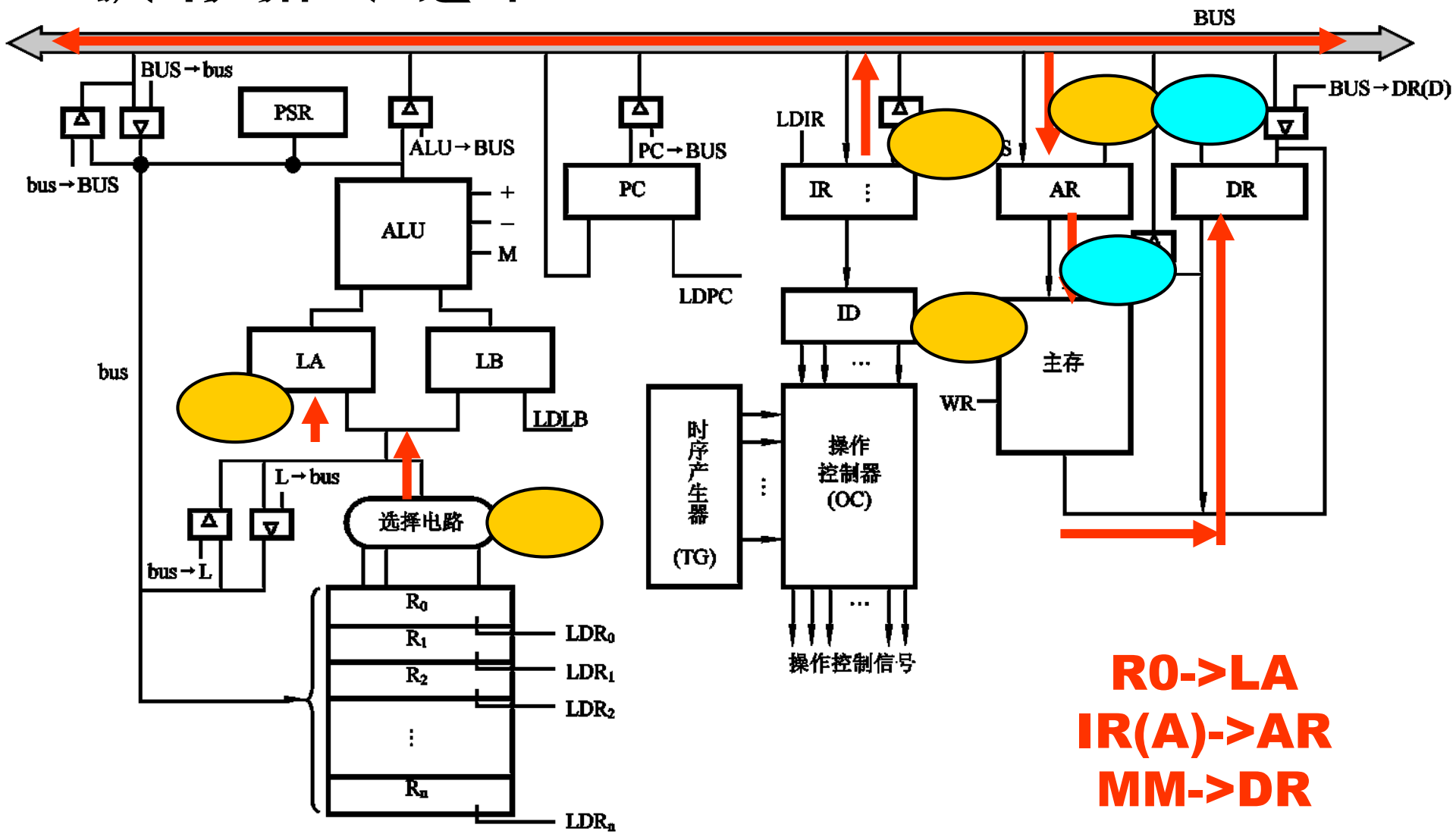
DR→IR



DR→BUS
LDIR

执行指令过程

ADD R0，（81）



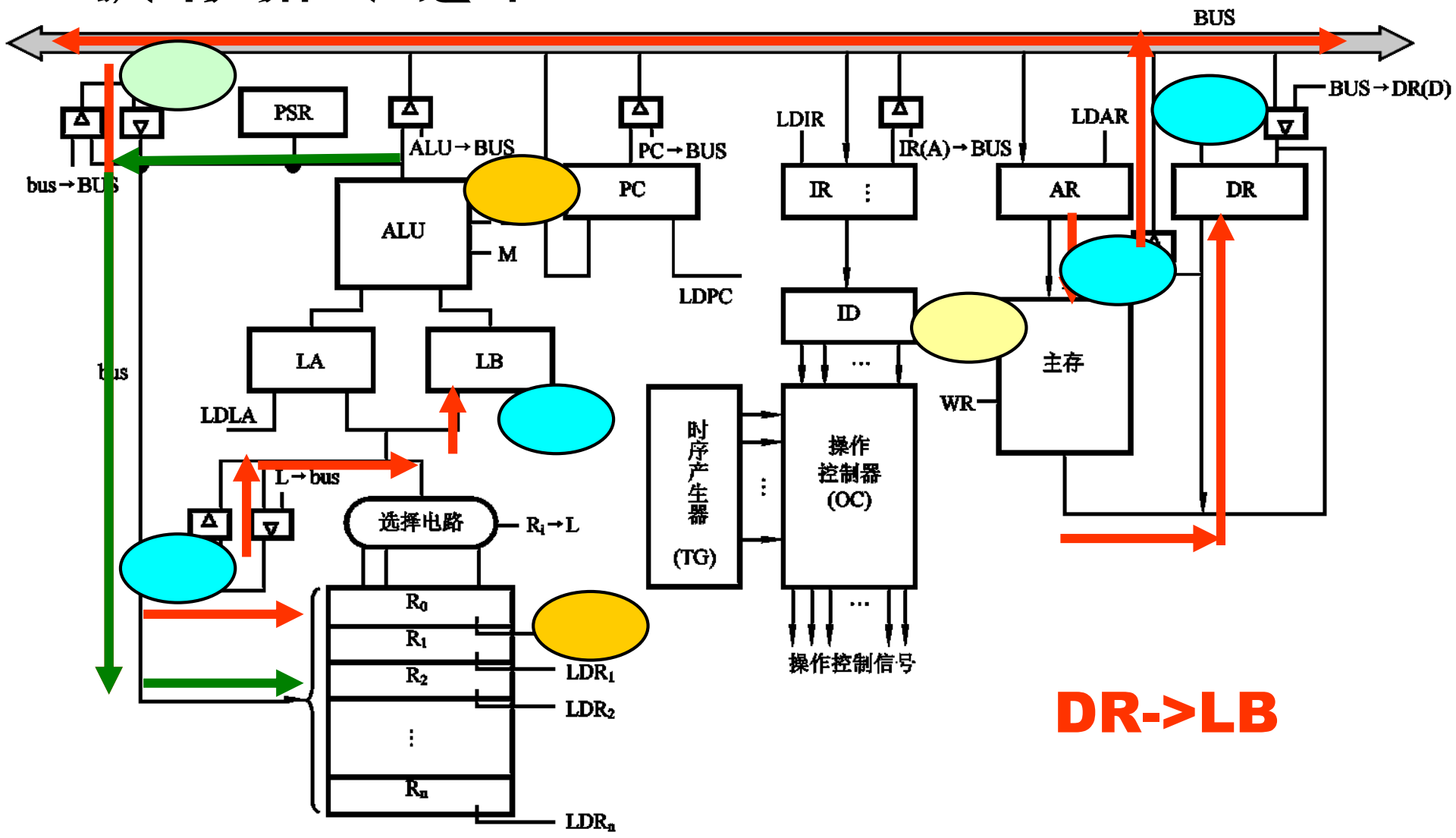
R0->LA

IR(A)->AR

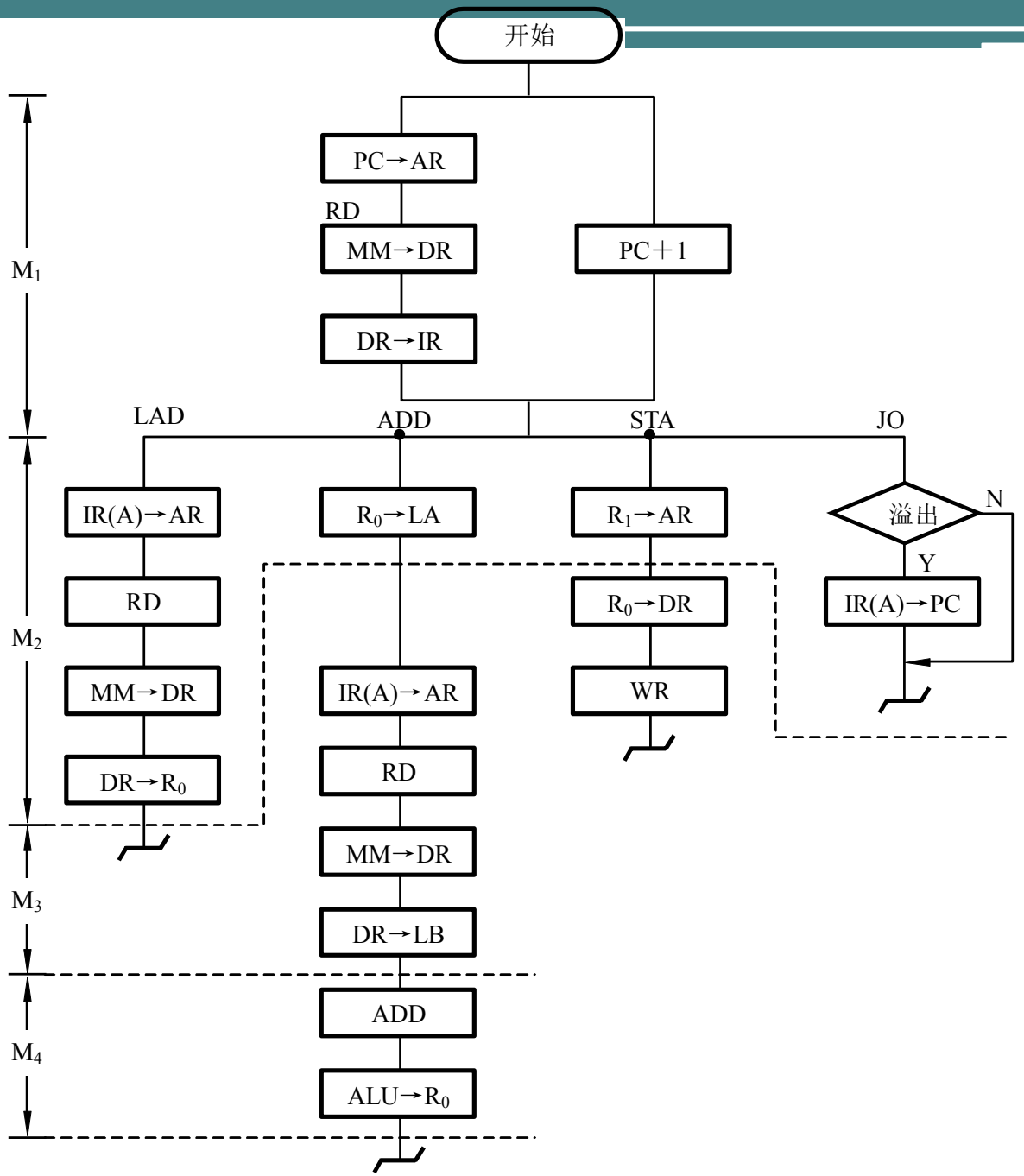
MM->DR

执行指令过程

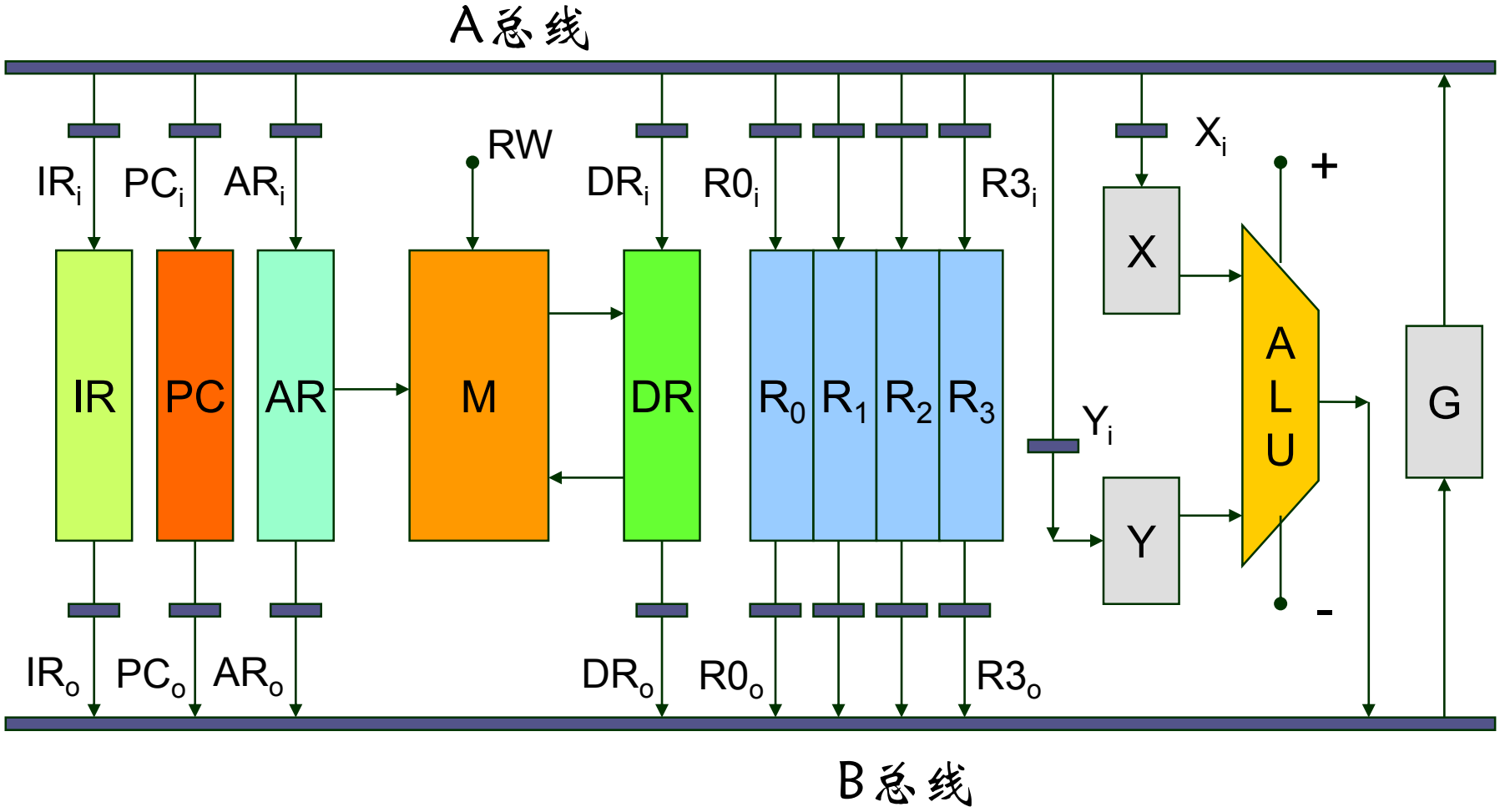
ADD R0，（81）



DR->LB

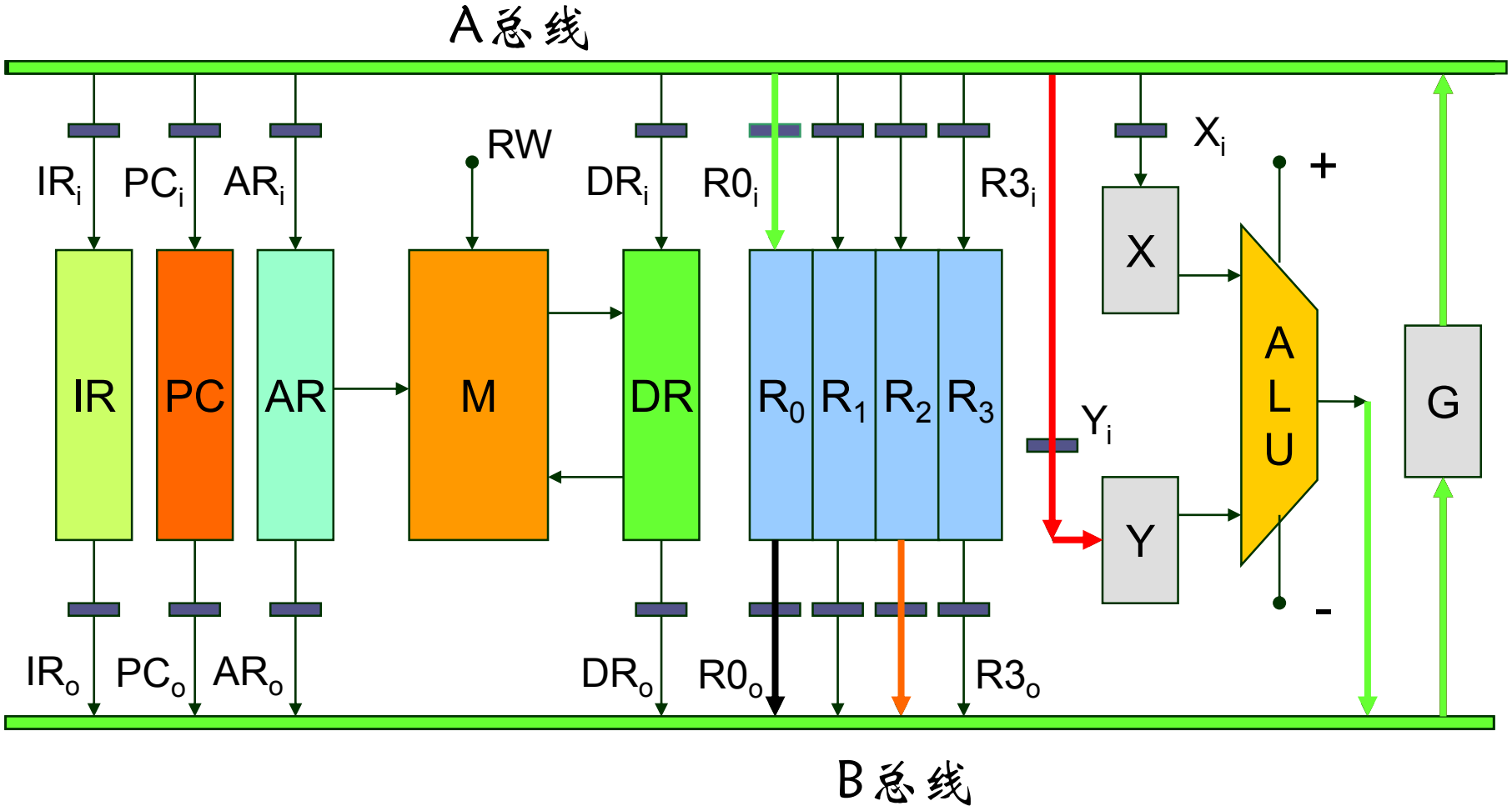

开始
PC→AR
RD
MM→DR
PC＋1
DR→IR
M1
LAD
ADD
STA
JO
IR(A)→AR
R0→LA
R1→AR
溢出
N
Y
RD
R0→DR
IR(A)→PC
M2
MM→DR
IR(A)→AR
WR
DR→R0
RD
MM→DR
M3
DR→LB
ADD
M4
ALU→R0

双总线结构机器的数据通路(例子)

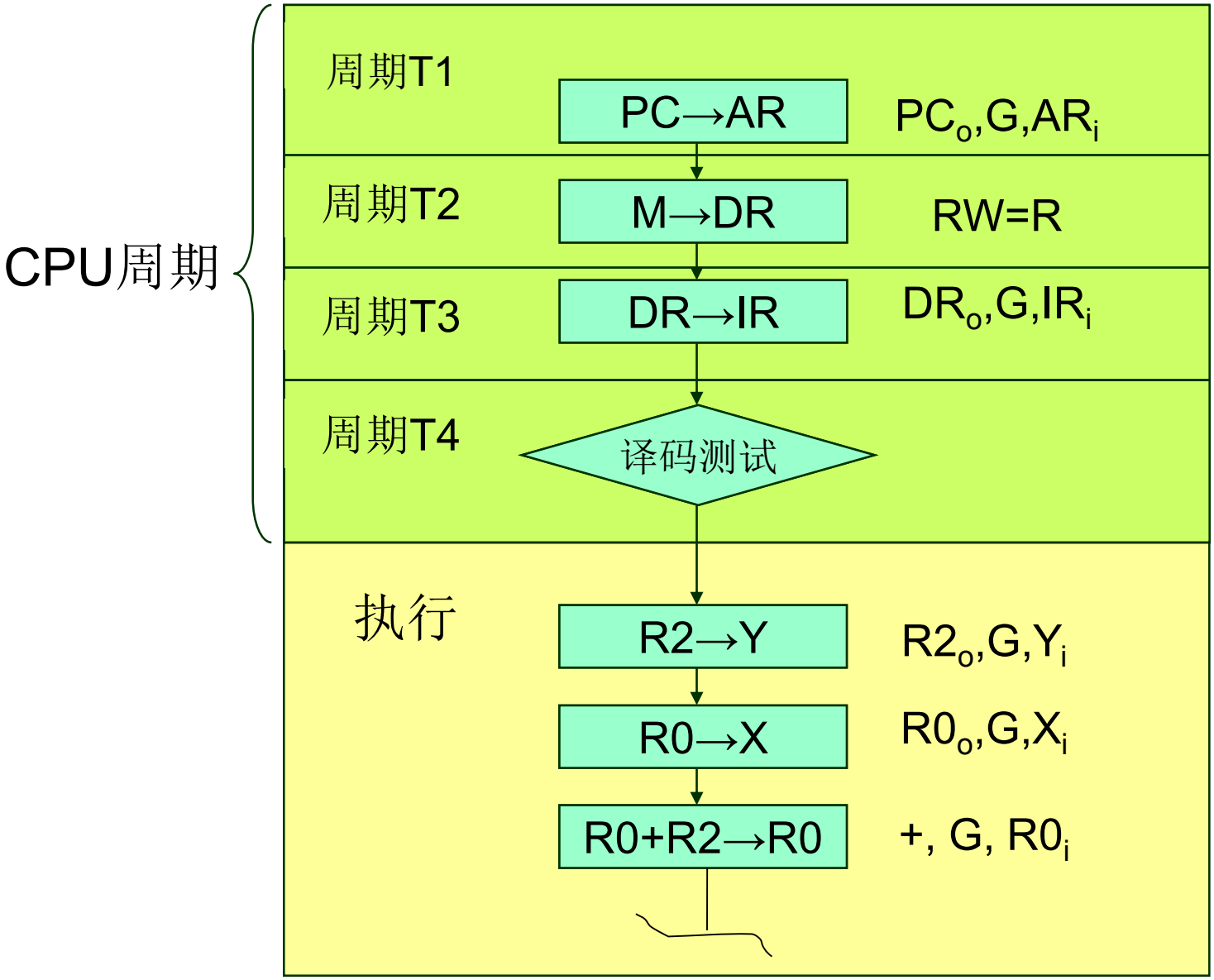


执行指令

ADD R0,R2



“ADD R2，R0”指令的指令周期框图



主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

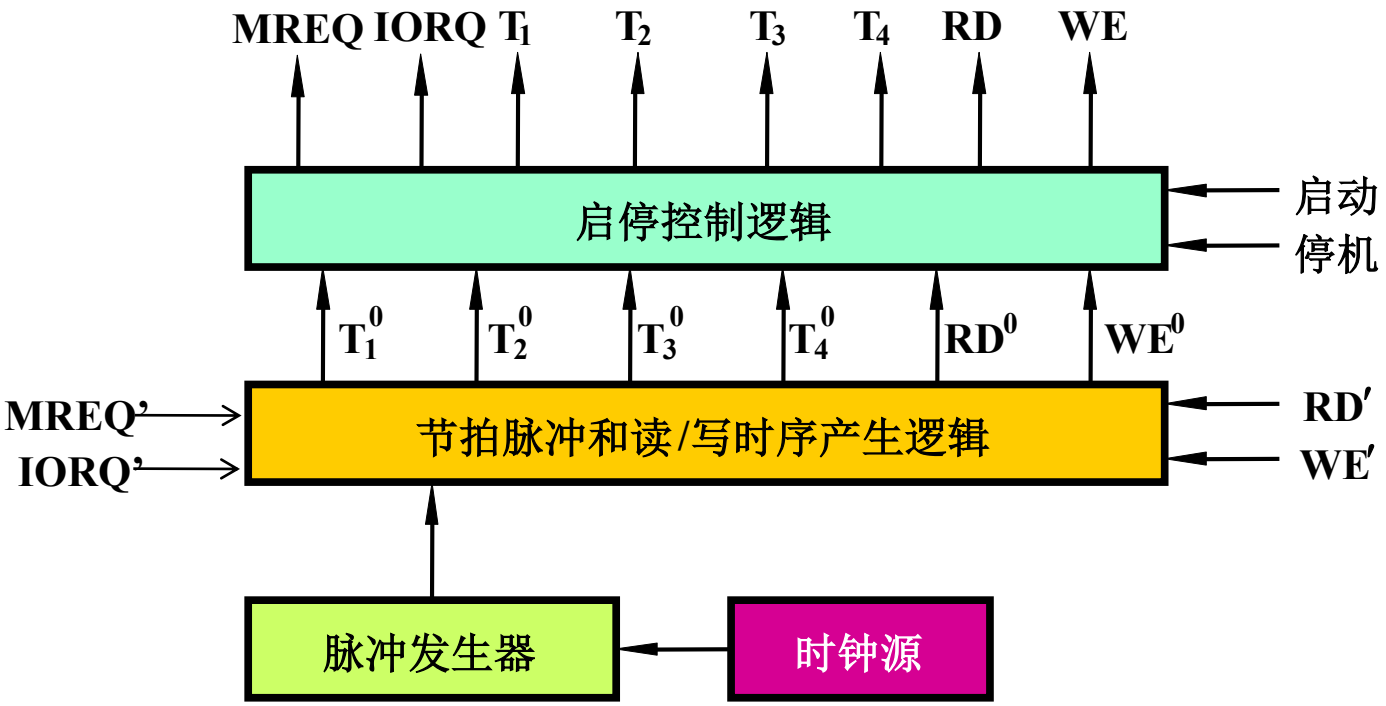
时序产生器和控制方式

- 时序信号来自CPU时序信号产生器。机器一旦被启动，即CPU开始取指令并执行指令时，操作控制器就利用定时脉冲的顺序和不同的脉冲间隔，有条理、有节奏地指挥机器的动作，规定在这个脉冲到来时做什么，在那个脉冲到来时又做什么。
- 问题：用二进制码表示的指令和数据都放在内存里，CPU是怎样识别出它们是数据还是指令呢?
- 从时间上来说，取指令是在指令周期的第一个CPU周期，即“取指令”阶段；而取数据是在指令周期的后面几个CPU周期中，即“执行指令”阶段。从空间上来说，如果取出的代码是指令，则送**指令寄存器**，如果取出的代码是数据，则送运算器。

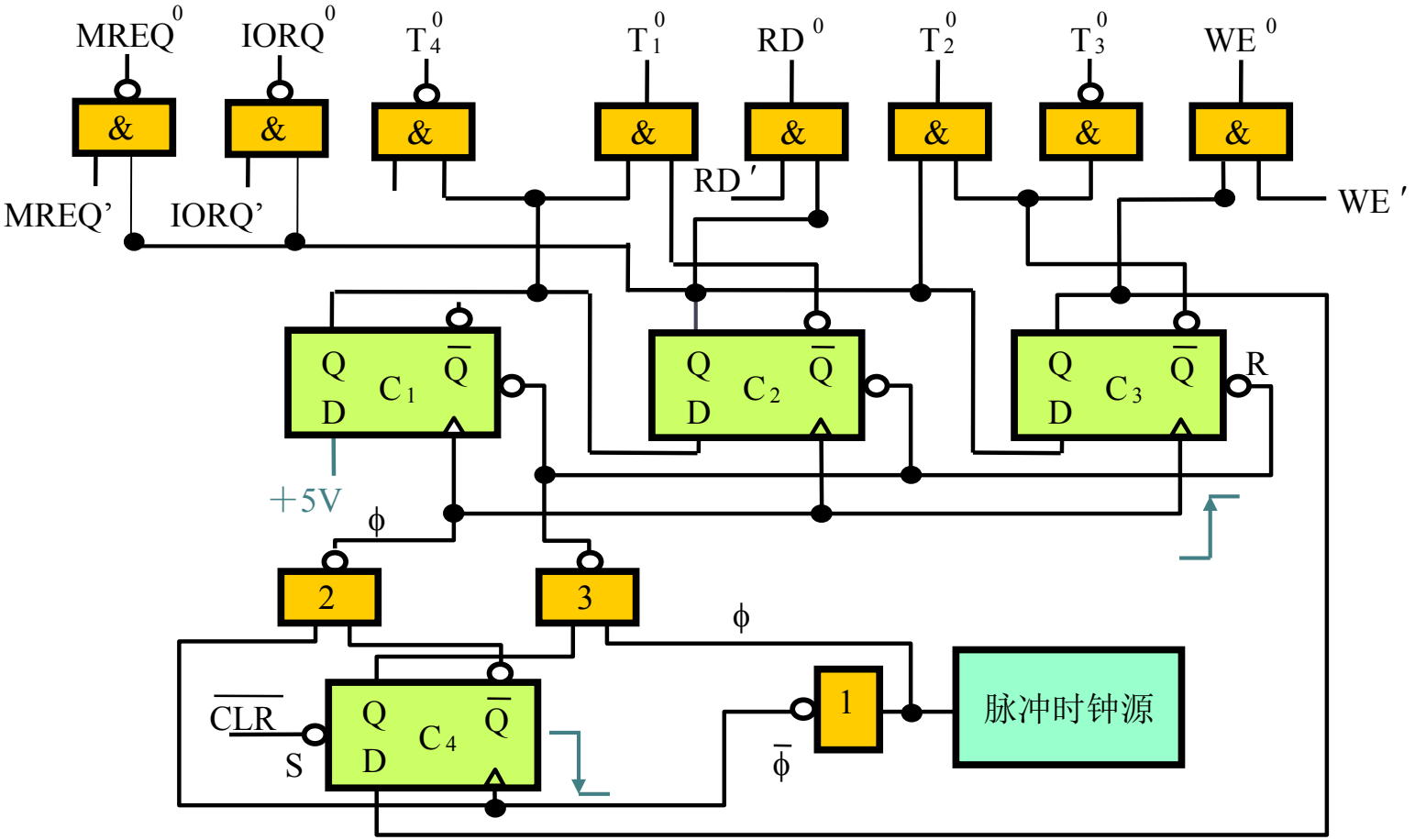
时序产生器和控制方式

- 计算机采用多级时序机制：
  - 硬布线控制器，时序信号往往采用主状态周期-节拍电位-节拍脉冲三级机制。
  - 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。节拍电位表示一个CPU周期的时间，而节拍脉冲把一个CPU周期划分成几个较小的时间间隔。

时序发生器（逻辑电路不作要求）

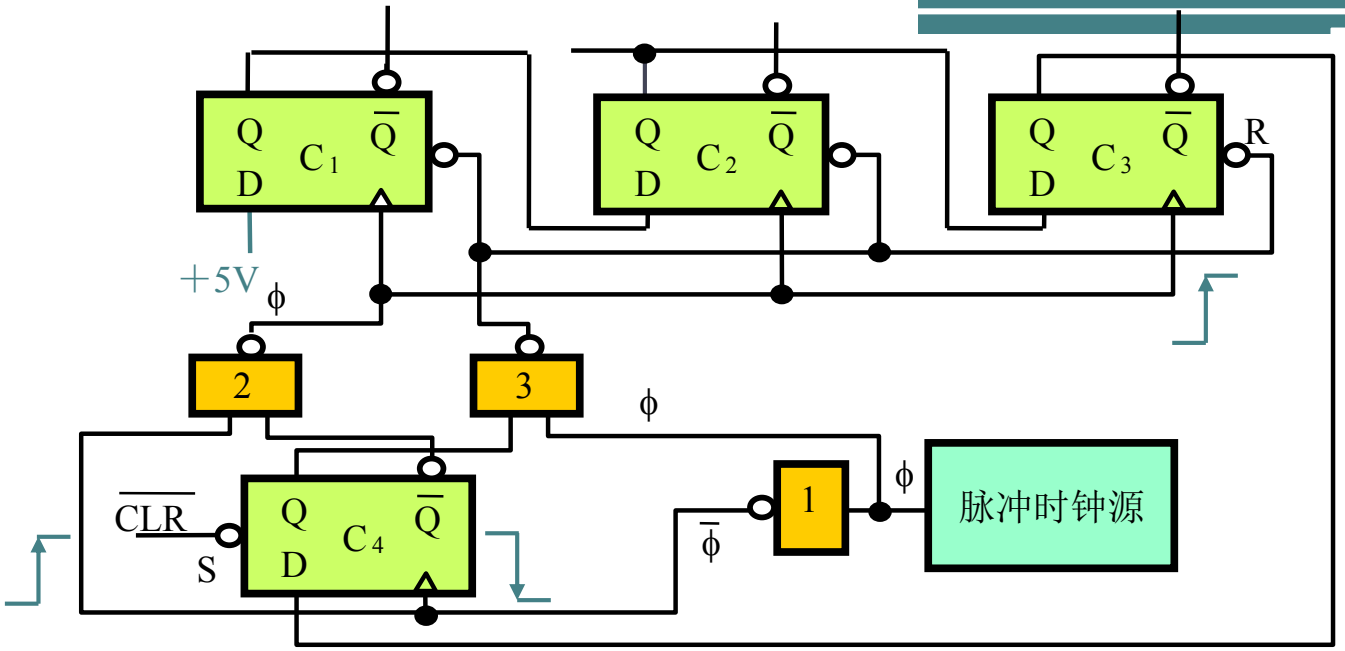


环形脉冲发生器与读写时序

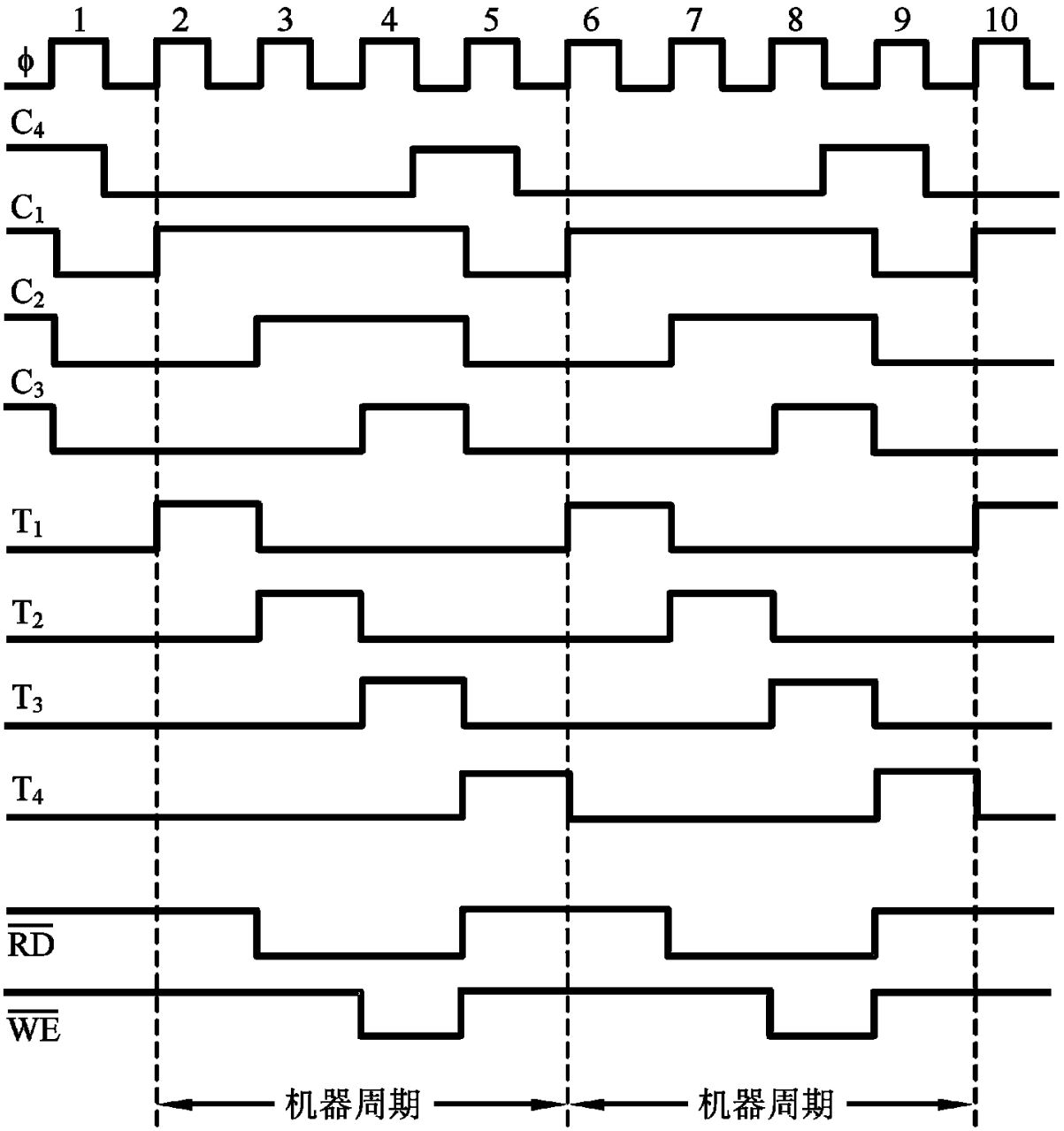


电路说明

- 4个触发器输入输出串联构成循环移位电路
- D触发器R/S端分别为Reset和Set
- C1 C2 C3 时钟信号为上跳沿
- C4 时钟信号为下跳沿



Φ	C4	C1	C2	C3
CLR 上跳沿	1	0	0	0
下跳沿	0	0	0	0
上跳沿	0	1	0	0
上跳沿	0	1	1	0
上跳沿	0	1	1	1
下跳沿	1	1	1	1
上跳沿	1	0	0	0



$$T_1^0 = C_1 \bullet \overline{C_2}$$

$$T_2^0 = C_2 \bullet \overline{C_3}$$

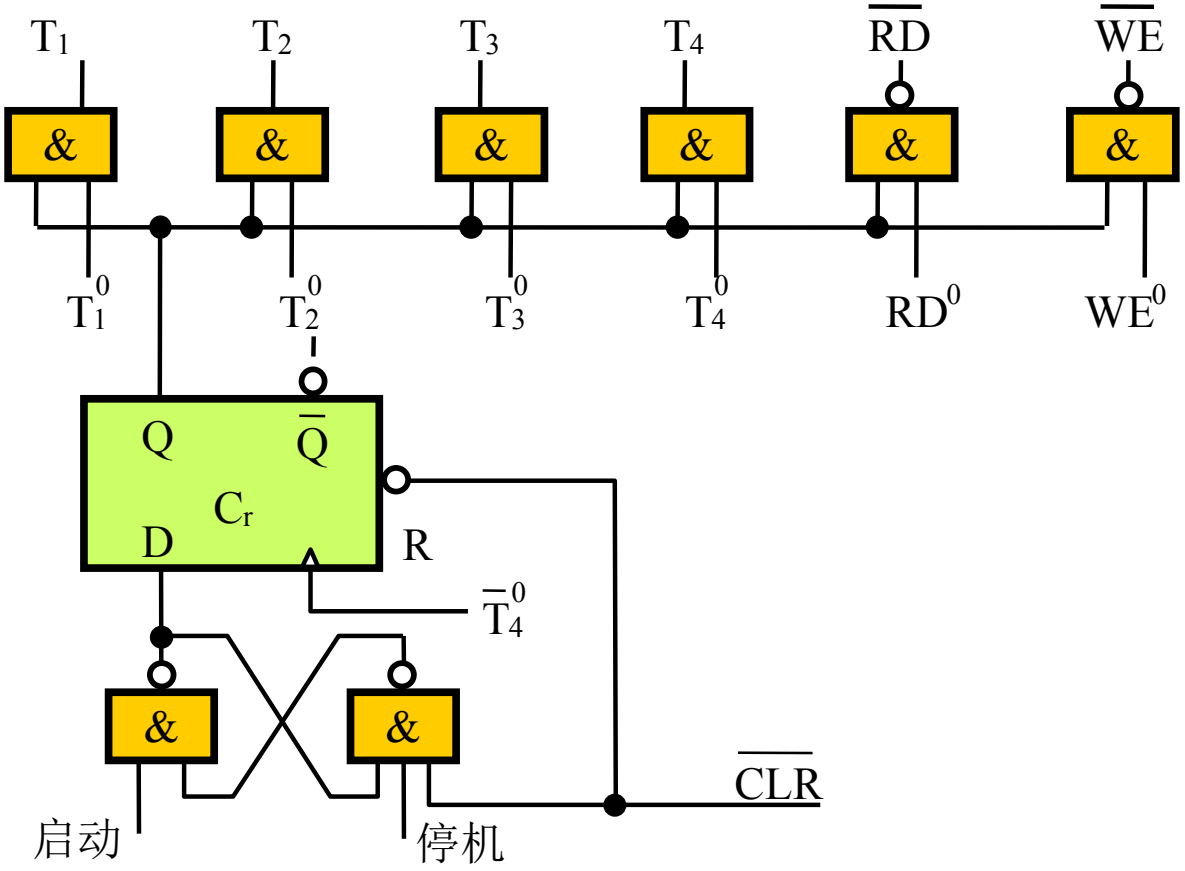
$$T_3^0 = C_3$$

$$T_4^0 = \overline{C_1}$$

$$RD^0 = C_2 \bullet RD'$$

$$WE^0 = C_3 WE'$$

启停控制逻辑



控制方式

- 每条指令和每个操作控制信号所需的时间各不相同，形成控制不同操作序列的时序信号的方法，称为控制器的控制方式。
- 常用的有同步控制、异步控制、联合控制三种方式。其实质则反映了时序信号的定时方式。
- 1.同步控制方式

  在任何情况下，已定的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。根据不同情况，同步控制方式可选取如下方案：

(1)采用完全统一的机器周期执行各种不同的指令。

(2)采用不定长机器周期。

(3)中央控制与局部控制结合。

控制方式

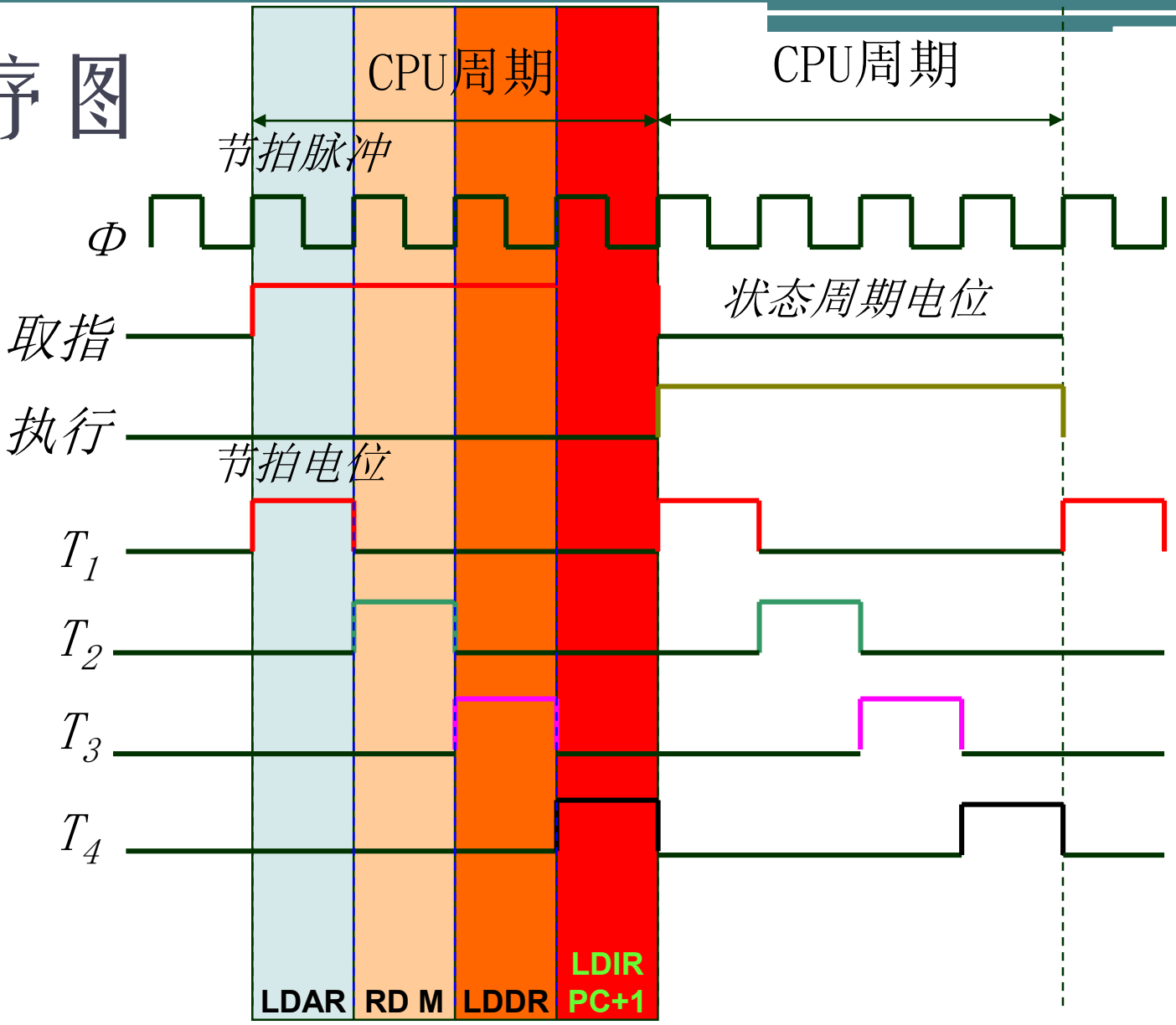
2.异步控制方式

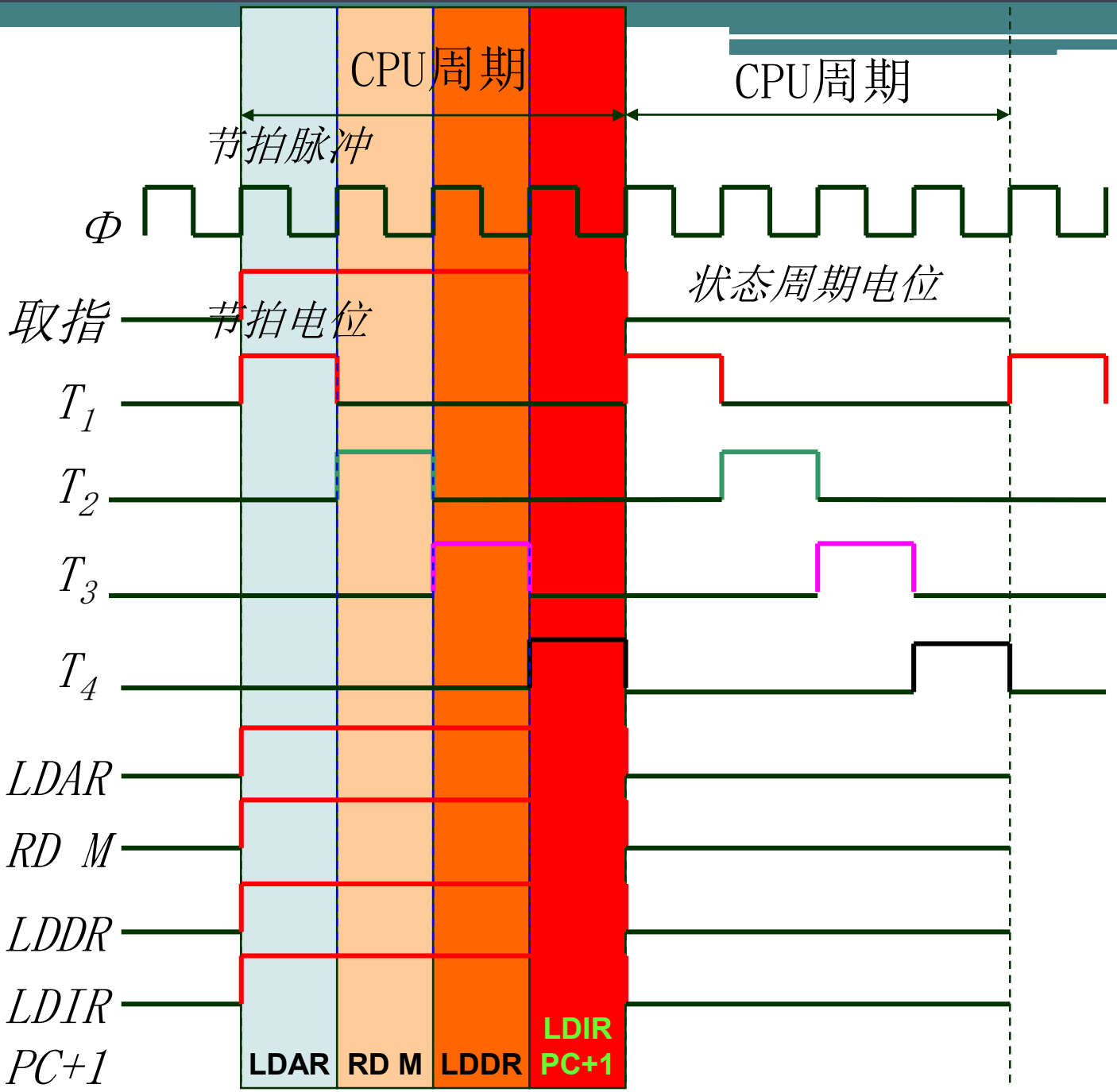
特点：每条指令、每个操作控制信号需要多少时间就占用多少时间。这意味着每条指令的指令周期可由多少不等的机器周期数组成；也可以是当控制器发出某一操作控制信号后，等待执行部件完成操作后发“回答”信号，再开始新的操作。显然，用这种方式形成的操作控制序列没有固定的CPU周期数(节拍电位)或严格的时钟周期(节拍脉冲)与之同步。

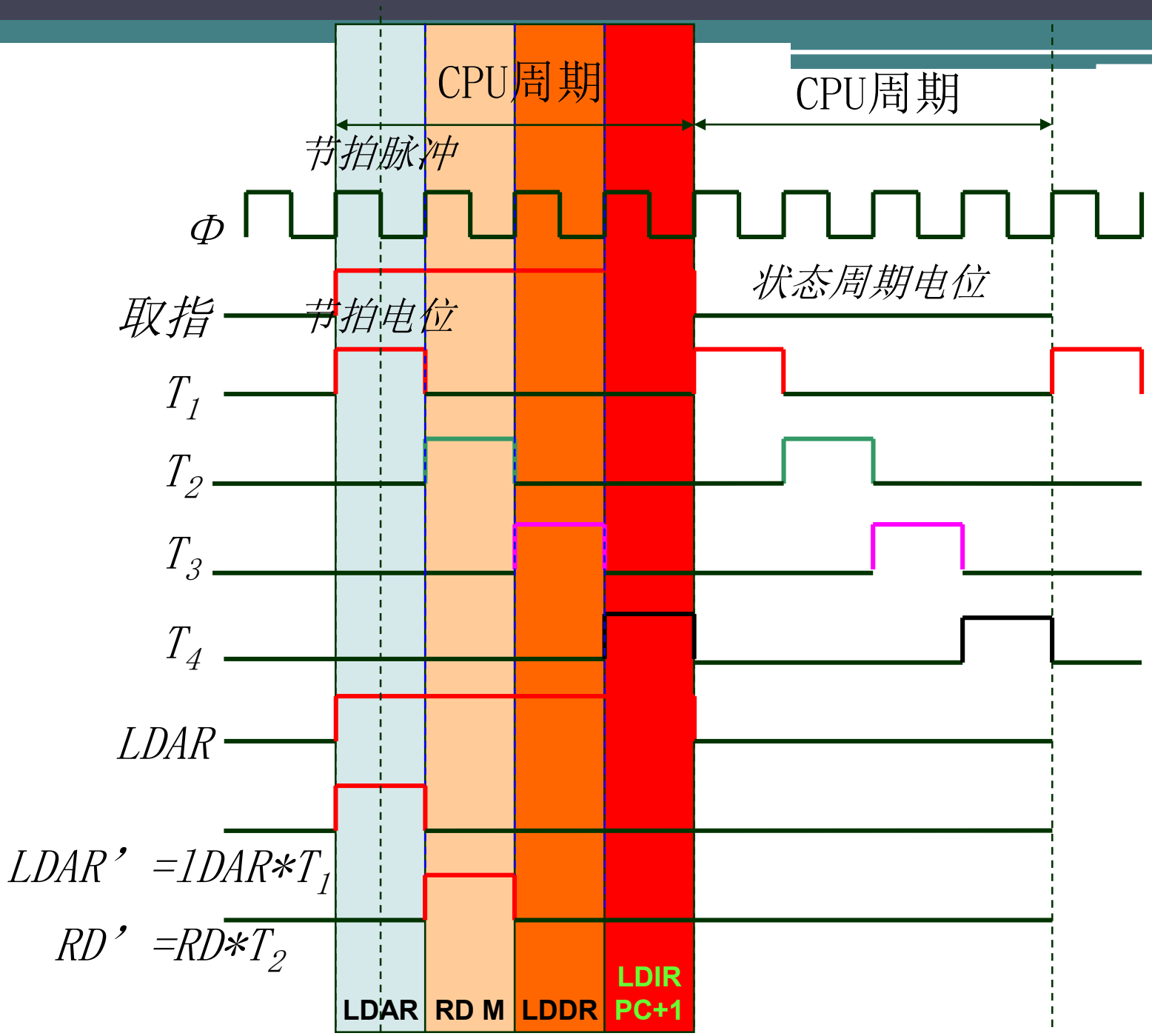
3.联合控制方式

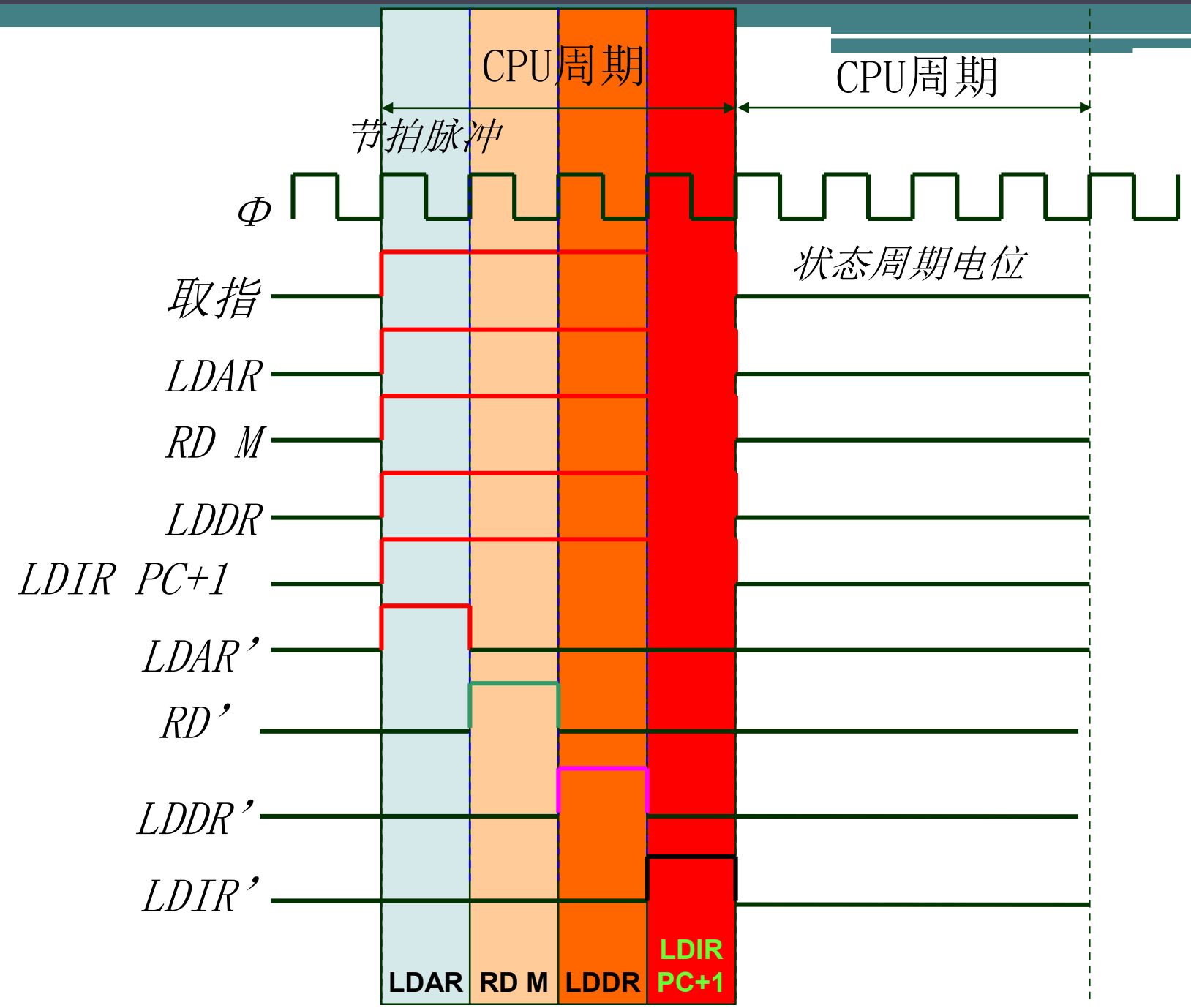
同步控制和异步控制相结合的方式。1）大部分操作序列安排在固定的机器周 期中，对某些时间难以确定的操作则以执行部件的“回答”信号作为本次操作的结束；2） 机器周期的节拍脉冲数固定，但是各条指令周期的机器周期数不固定。

时序图


CPU周期
CPU周期
节拍脉冲
Φ
状态周期电位
取指
执行
节拍电位
T1
T2
T3
T4
LDAR
RD M
LDDR
LDIR
PC+1


CPU周期
CPU周期
节拍脉冲
Φ
状态周期电位
取指
节拍电位
T1
T2
T3
T4
LDAR
RD M
LDDR
LDIR
PC+1
LDAR
RD M
LDDR
LDIR
PC+1


CPU周期
CPU周期
节拍脉冲
Φ
状态周期电位
取指
节拍电位
T1
T2
T3
T4
LDAR
LDAR’ =1DAR*T1
RD’ =RD*T2
LDAR
RD M
LDDR
LDIR
PC+1


CPU周期
CPU周期
节拍脉冲
Φ
状态周期电位
取指
LDAR
RD M
LDDR
LDIR PC+1
LDAR'
RD'
LDDR'
LDIR'
LDAR
RD M
LDDR
LDIR
PC+1

主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

微程序控制器

发展

- 微程序的概念和原理是由英国剑桥大学的·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。

微程序控制器

- 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。采用微程序控制方式的控制器称为微程序控制器。
- 微程序设计技术是用软件方法来设计硬件的技术。

微程序控制器 --微命令和微操作

- 控制部件与执行部件

  二者通过控制线，反馈线联系。

- 微命令

  控制部件通过控制线向执行部件发出的各种控制命令。微命令是控制计算机各部件完成某个基本微操作的命令。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进行的操作。微命令是微操作的控制信号，微操作是微命令的操作过程。

微程序控制器 --微命令和微操作

- 打开或者关闭控制门的控制信号为微命令
- 微命令是控制信号最小，最基本的单位
- 微命令带来的执行部件的动作称为微操作
- 由于数据通路的结构关系，微操作可分为相容的和互斥的两种：
  - 互斥性微命令
    - 互斥性的微操作，是指不能在同时或不能在同一个 CPU周期内并行执行的微操作。
  - 相容性微命令
    - 相容性的微操作，是指在同时或同一个CPU周期内可以并行执行的微操作。

微程序控制器 --微命令和微操作

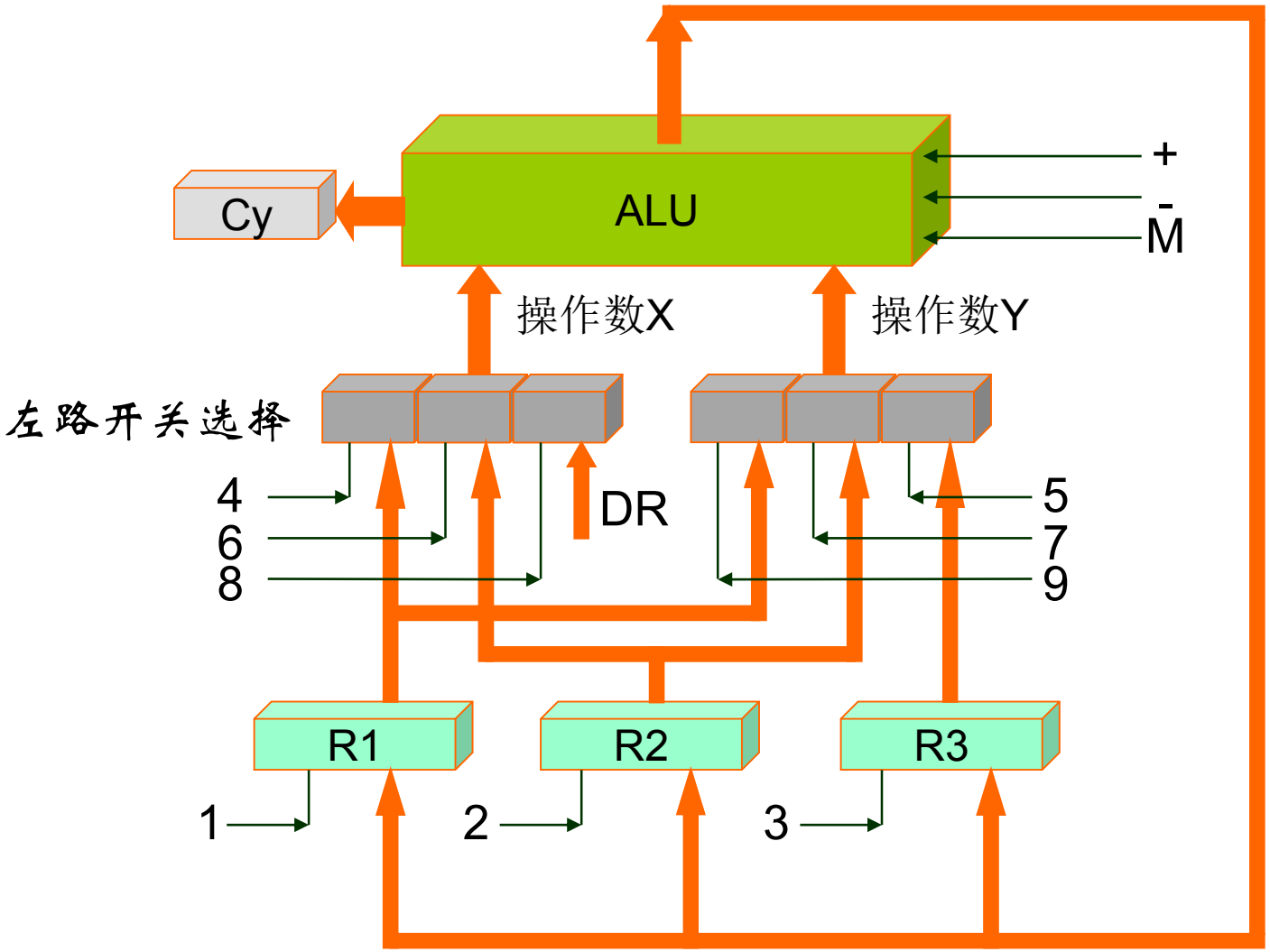


图5.21 简单运算器数据通路图

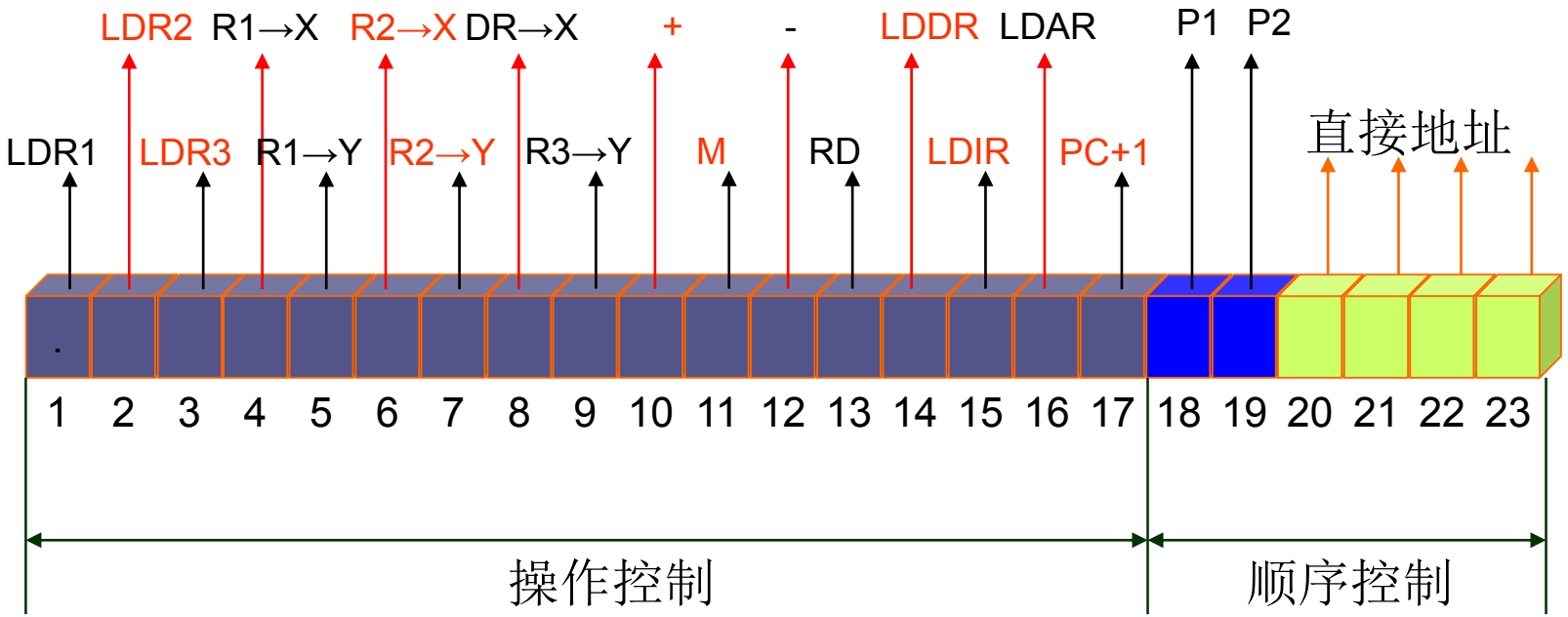
微程序控制器--微命令和微操作

- 一条指令的处理包含许多微操作序列
- 这些操作可以归结为信息传递、运算
- 将这些操作所需要的控制信号以多条微指令表示
- 执行一条微指令就给出一组微操作控制信号
- 执行一条指令也就是执行一段由多条微指令组成的微程序

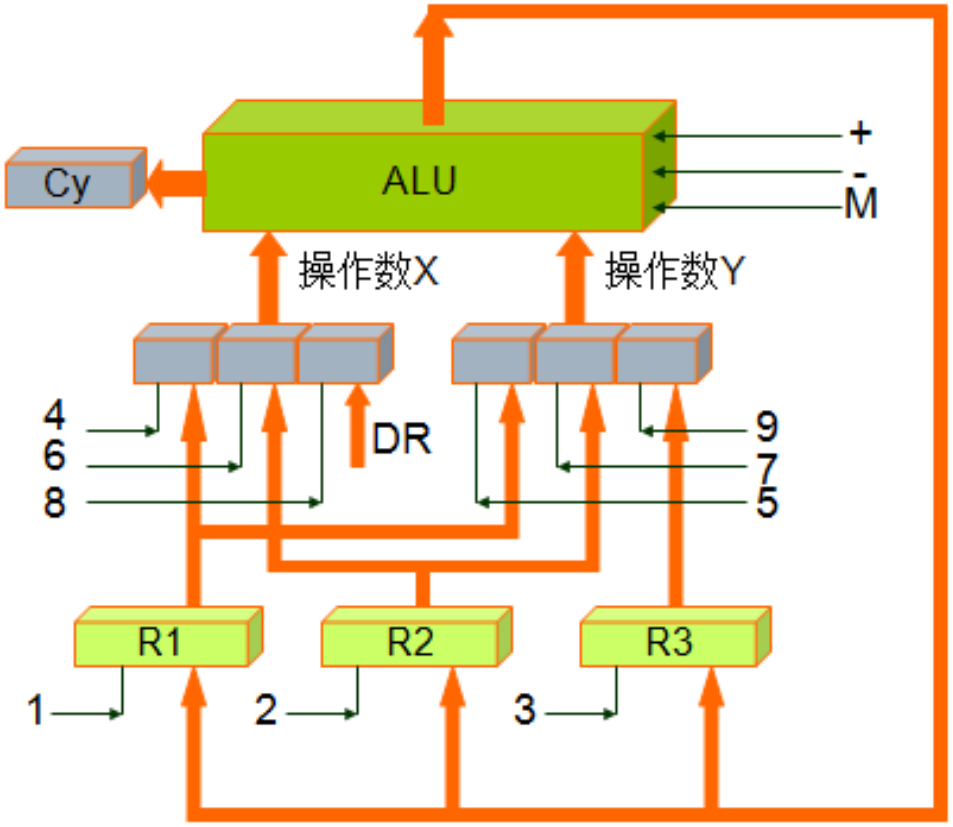
微指令和微程序

- 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**Microinstruction 。
- 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
  - 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。

- 图5.22表示一个具体的微指令结构，微指令字长为23位，它由操作控制和顺序控制两大部分组成 。



微命令



- 1: LDR1
- 2: LDR2
- 3: LDR3
- 4: R1→X
- 5: R1→Y
- 6: R2→X
- 7: R2→Y
- 8: DR→X
- 9: R3→Y
- 10: +
- 11: -
- 12: M
- 13: RD
- 14: LDDR
- 15: LDIR
- 16: LDAR
- 17: PC+1

微指令和微程序

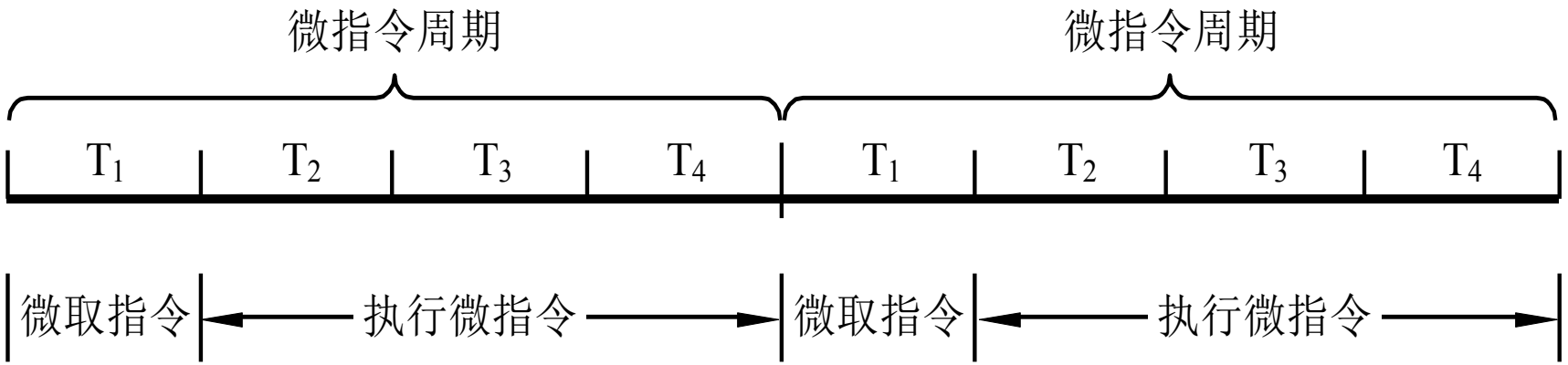
- 将指令系统功能实现所需的控制信号以微指令为单位存储。微指令中的每一位对应一根控制信号线
- 每条指令对应一段微程序
- 微程序由若干条微指令构成
- 机器执行指令时逐条取出微指令执行，使得相应部件执行规定的操作，执行完微程序，也就给出了该指令所需要的全部控制信号，从而完成一条指令的执行。

微指令格式

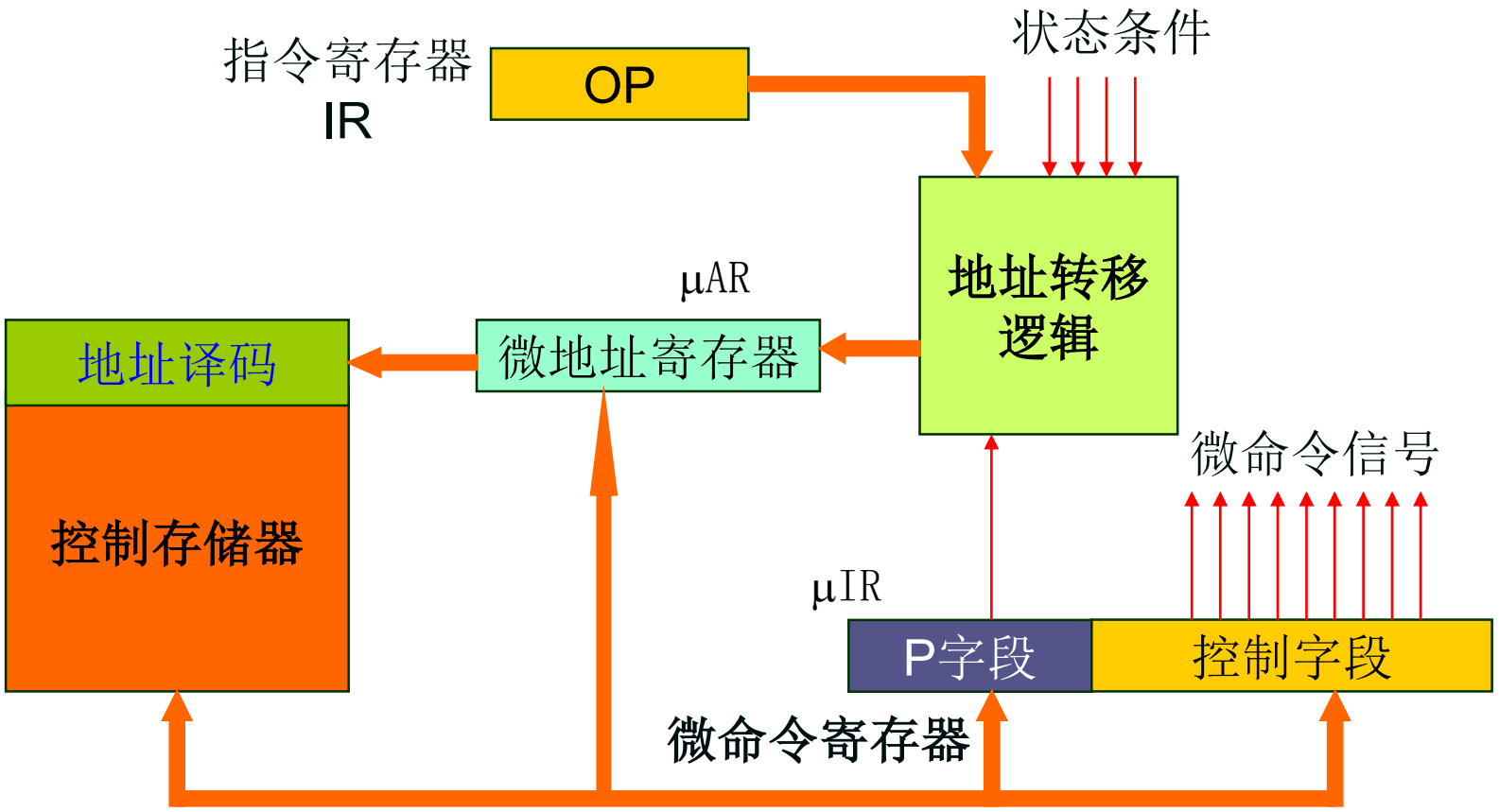
- 操作控制字段
  - 操作控制字段直接给出多种微操作的控制信号
- 顺序控制字段
  - 用于控制微程序的执行顺序
  - 包括判断逻辑字段和直接地址字段
  - 直接地址字段存放下一条微指令的地址
  - 判断逻辑非零，则按约定好的规则，根据状态修正直接地址字段，从而得到下一条微指令的地址

微指令周期


控制字段
测试字段
下址字段

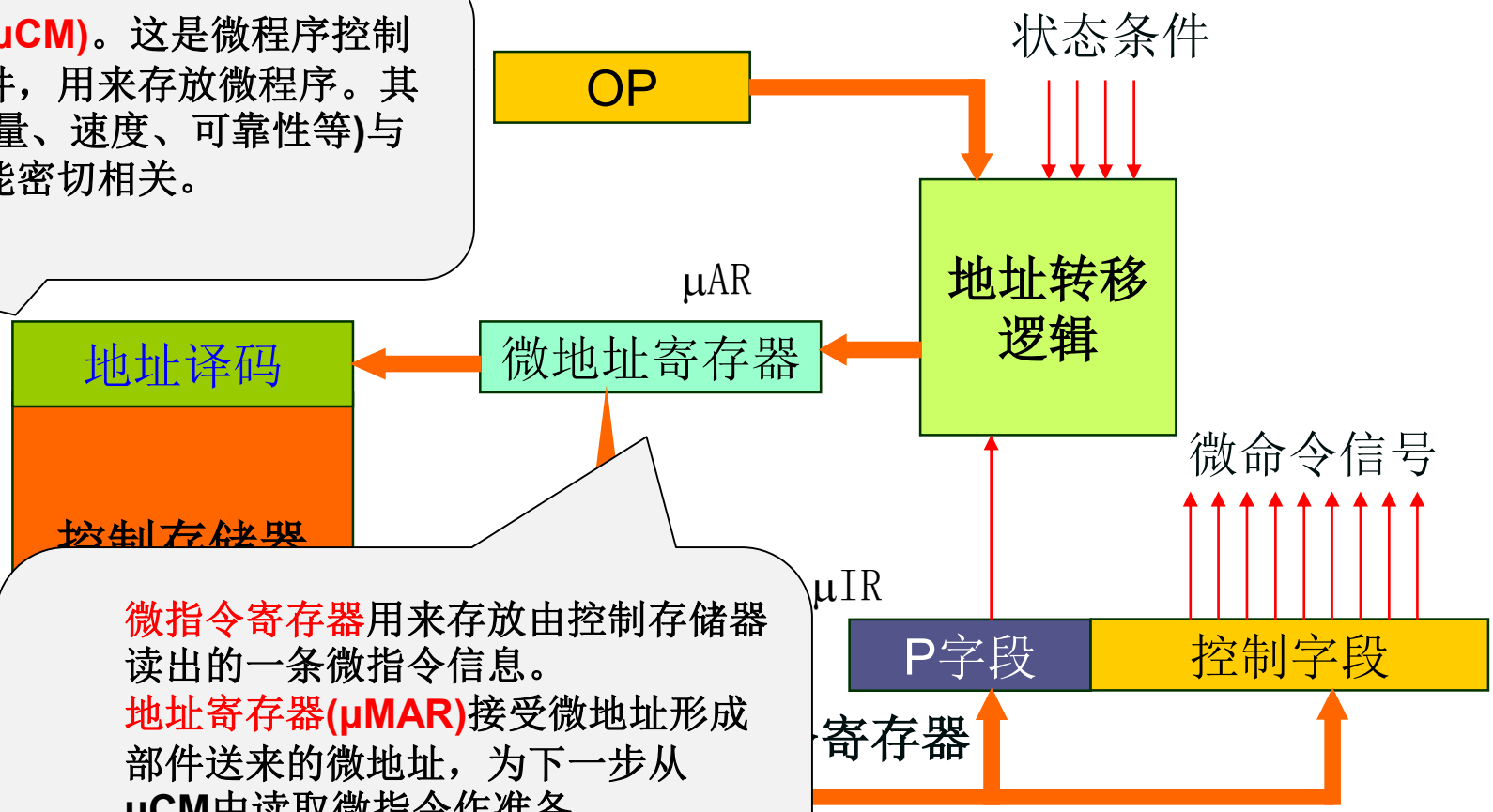

微指令周期
微指令周期
T1
T2
T3
T4
T1
T2
T3
T4
微取指令
执行微指令
微取指令
执行微指令

微程序控制器原理框图


指令寄存器
IR
OP
状态条件
地址转移
逻辑
μAR
地址译码
微地址寄存器
控制存储器
微命令信号
μIR
P字段
控制字段
微命令寄存器

微程序控制器原理框图

控制存储器(μCM)。这是微程序控制器的核心部件，用来存放微程序。其性能(包括容量、速度、可靠性等)与计算机的性能密切相关。



微指令寄存器用来存放由控制存储器读出的一条微指令信息。
地址寄存器(μMAR)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。
微命令寄存器则保存一条微指令的操作控制字段和判别测试字段的信息。

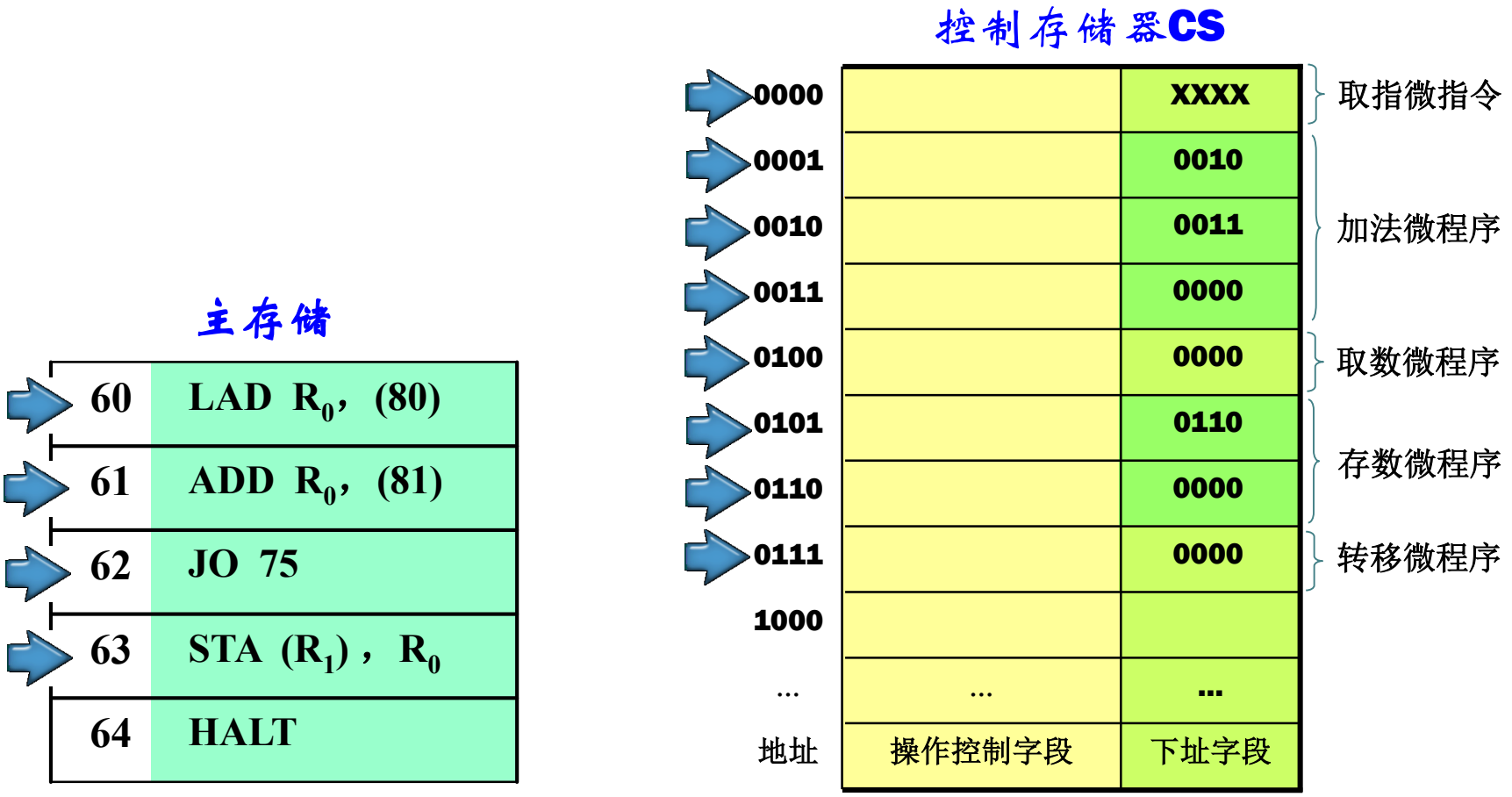
取指令微程序/执行指令微程序

- 取指令:
- 取指令的微指令(简称取指微指令)地址送μAR，并自动启动控制存储器进行读操作，将读出的微指令送μIR，执行微指令,读取指令到IR。
- 执行指令:
- 根据IR中指令的功能，产生该指令微程序入口地址，微程序入口地址送入μAR，读CS，读出的微指令送μIR、(下址字段送μAR)，
- 控制字段的微命令控制完成一组微操作。
- 同时由微地址产生逻辑或微指令下址字段形成下条微指令地址，按取微指令，执行微指令过程重复执行完微程序实现指令的功能。

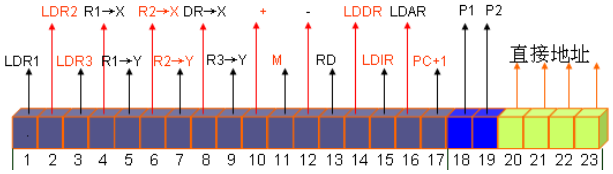
执行指令微程序...

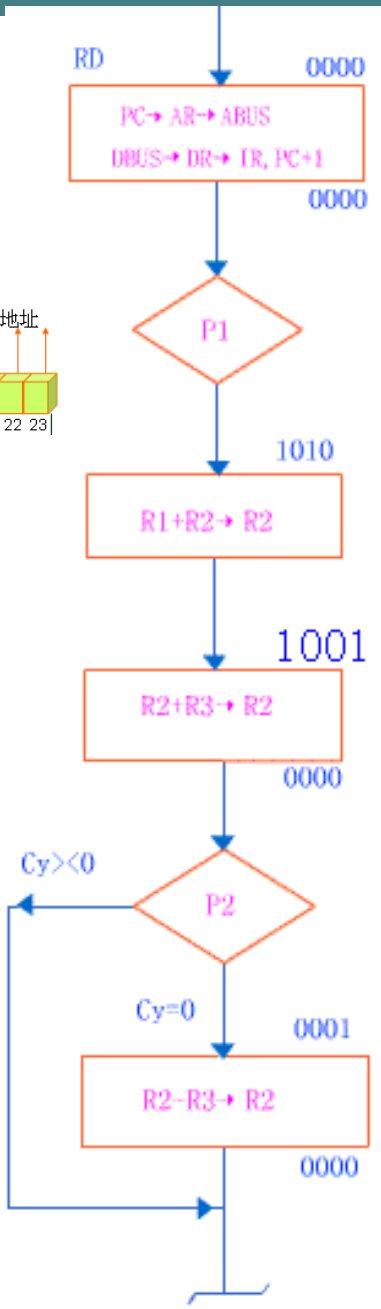
- 采用微程序控制的计算机的工作过程是执行微指令序列的过程。
- 微指令控制了取指令操作，
- 多条微指令实现了指令的功能。
- 而微指令中的微命令使执行部件完成微操作，计算机的工作过程是执行程序的过程，微观看，是执行指令的过程，再微观一点看，是执行部件进行微操作的过程 。

微程序存放示意图


主存储
60
LAD R0，(80)
61
ADD R0，(81)
62
JO 75
63
STA (R1)，R0
64
HALT
控制存储器CS
0000
XXXX
0001
0010
0010
0011
0011
0000
0100
0000
0101
0110
0110
0000
0111
0000
1000
…
…
…
地址
操作控制字段
下址字段
取指微指令
加法微程序
取数微程序
存数微程序
转移微程序

微程序举例





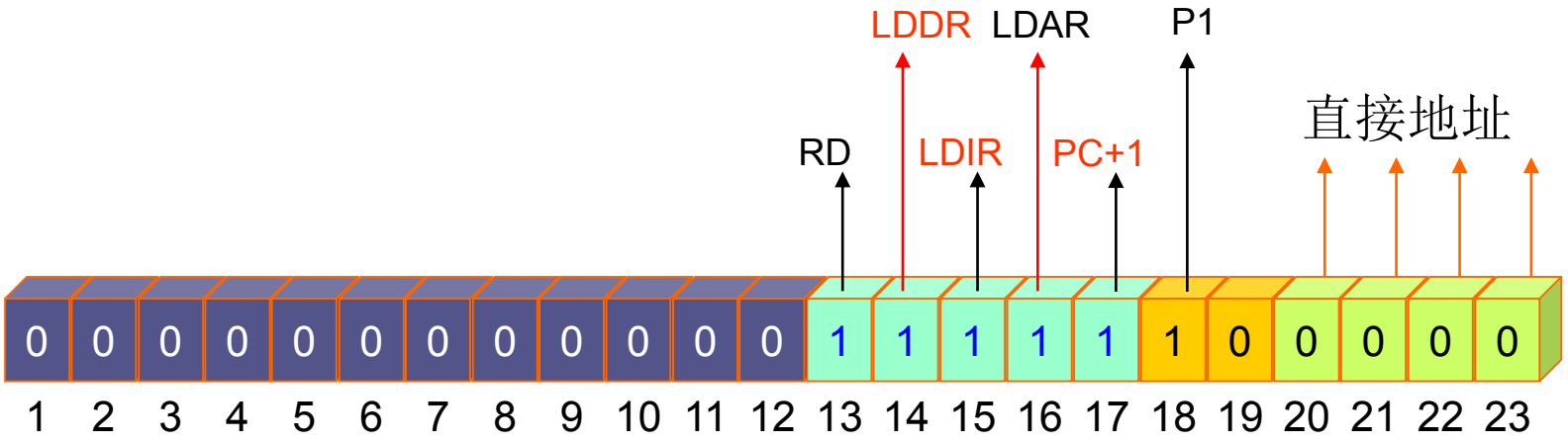
000 000 000 000 11111 10 0000

010 100 100 100 00000 00 1001

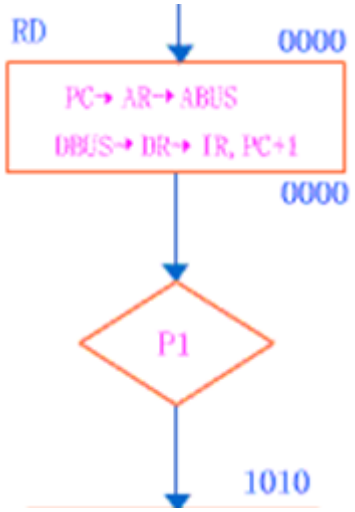
010 001 001 100 00000 01 0000

010 001 001 001 00000 00 0000

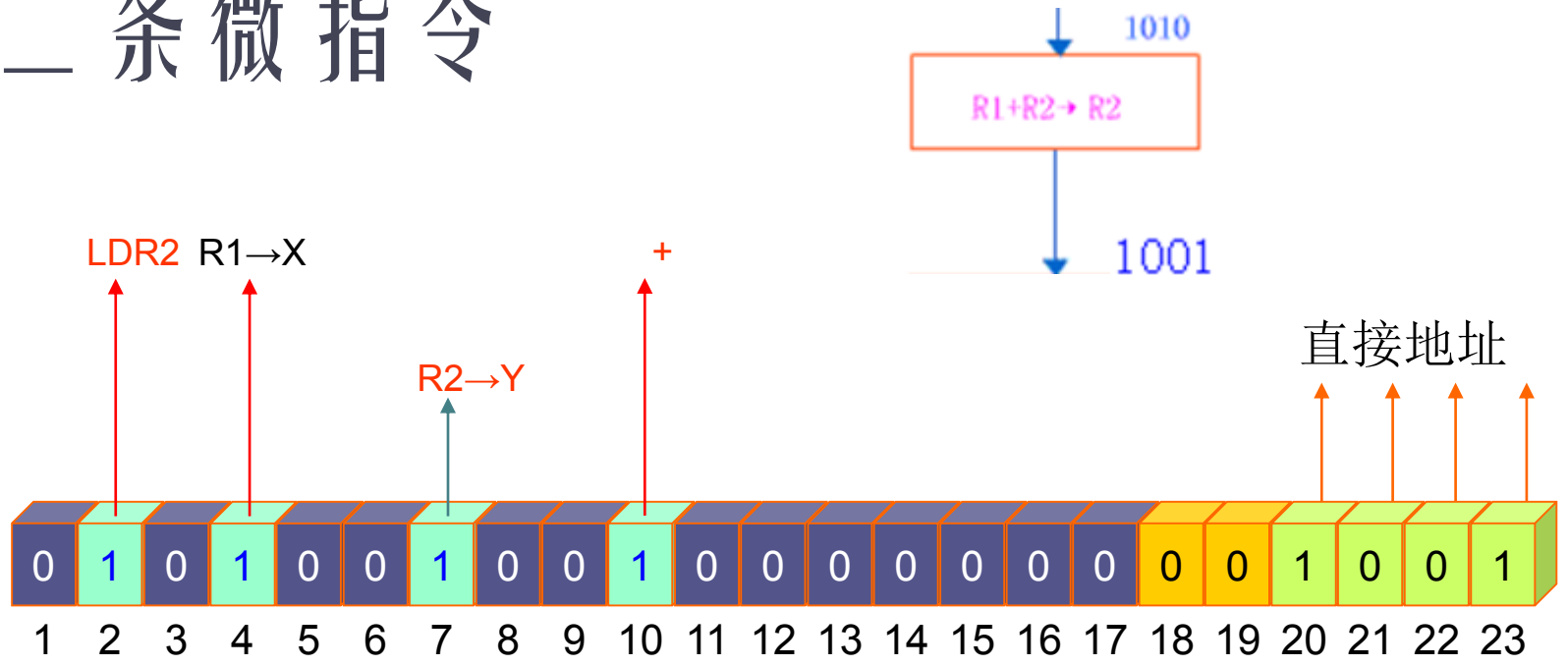
第一条微指令



- PC→AR→ABUS→DBUS→DR→IR
- PC+1
- LDAR RD LDDR LDIR PC+1

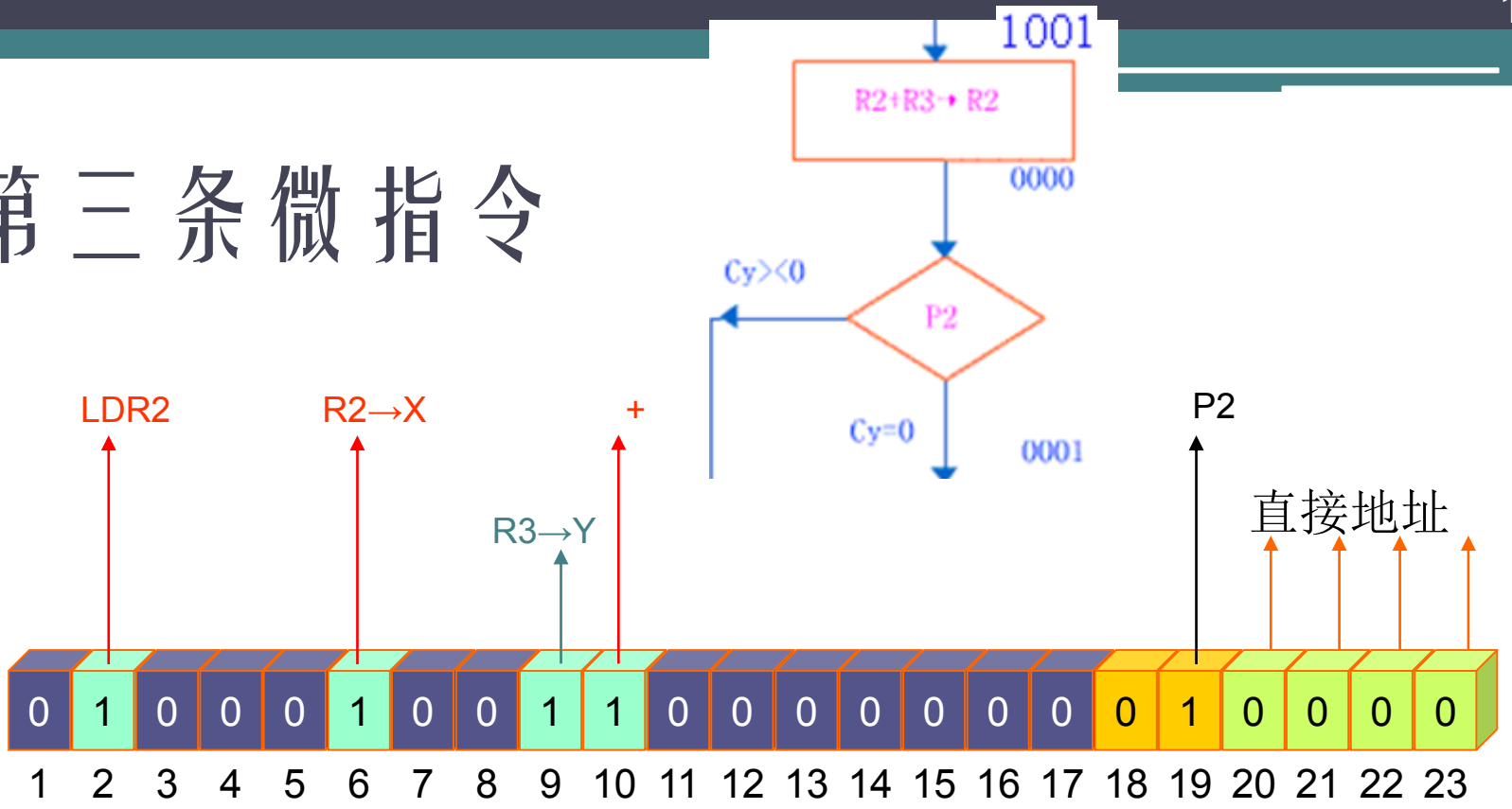


第二条微指令



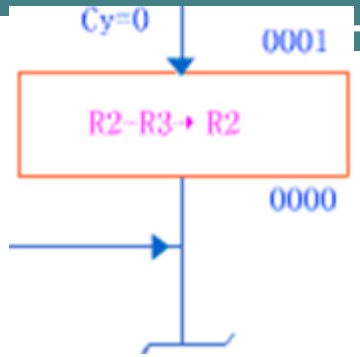
- R1→X　R2→Y　X+Y　X+Y→R2
- R1→X　R2→Y　+　LDR2

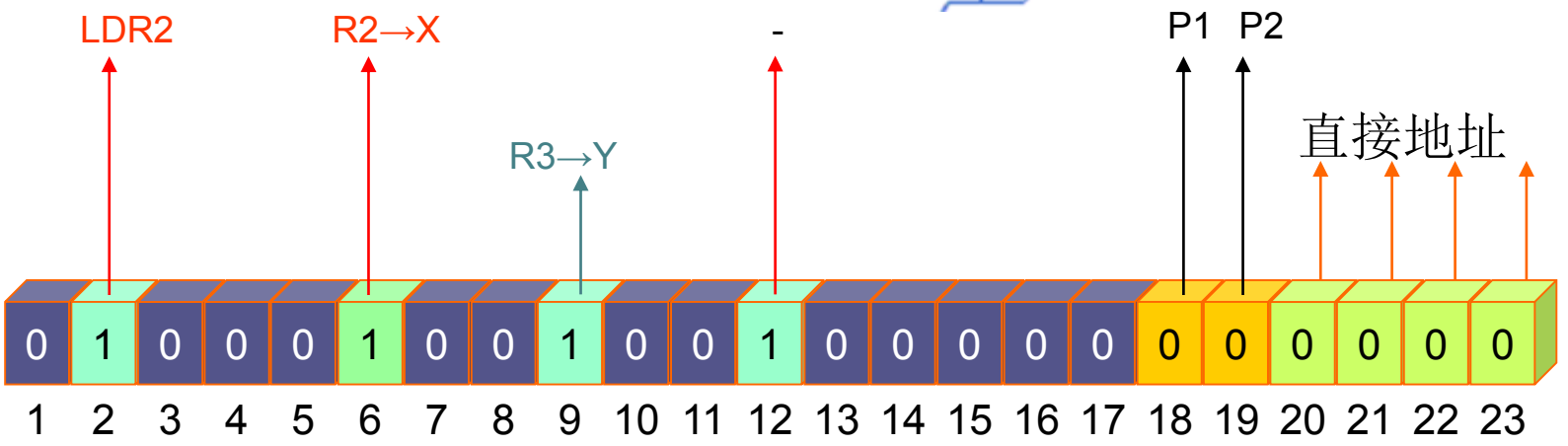
第三条微指令



- R2→X　R3→Y　X+Y　X+Y→R2
- R2→X　R3→Y　+　LDR2

第四条微指令





- R2→X　R3→Y　X-Y　X-Y→R2
- R2→X　R3→Y　-　LDR2

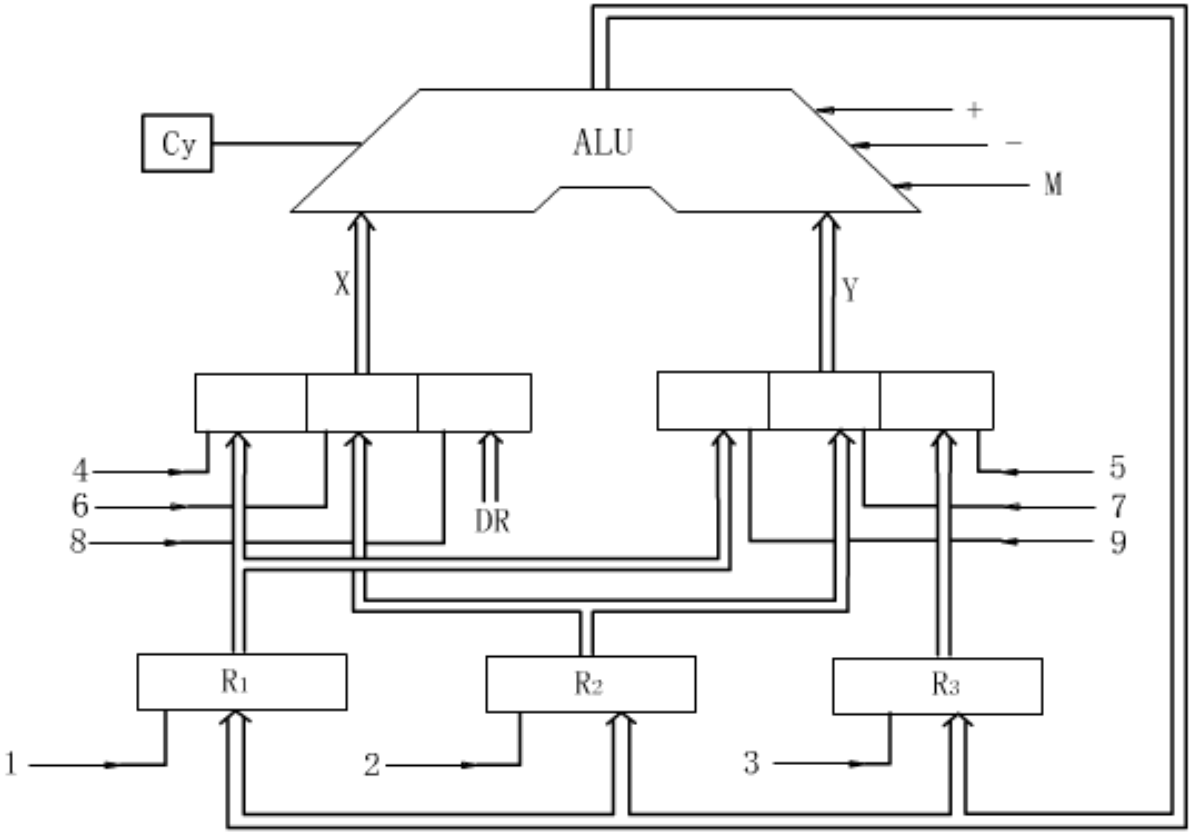


图5.21 简单运算器数据通路图

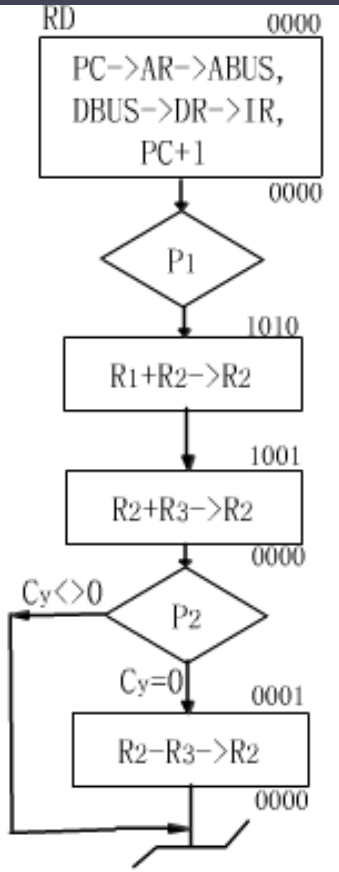


图5.25 十进制加法微程序

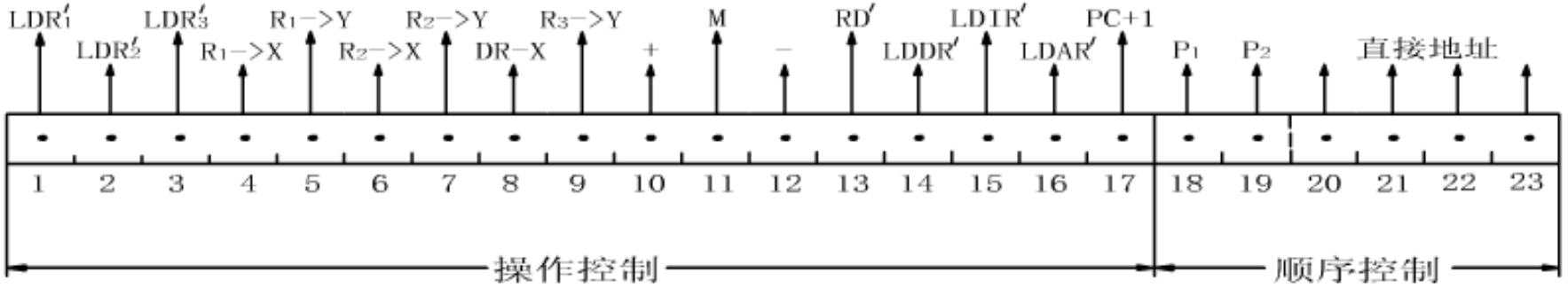


图 5.22 微指令基本格式

CPU周期与微指令周期的关系

- 在串行方式的微程序控制器中：
- **微指令周期 = 读出微指令的时间 + 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.下图示出了某小型机中CPU周期与微指令周期的时间关系：

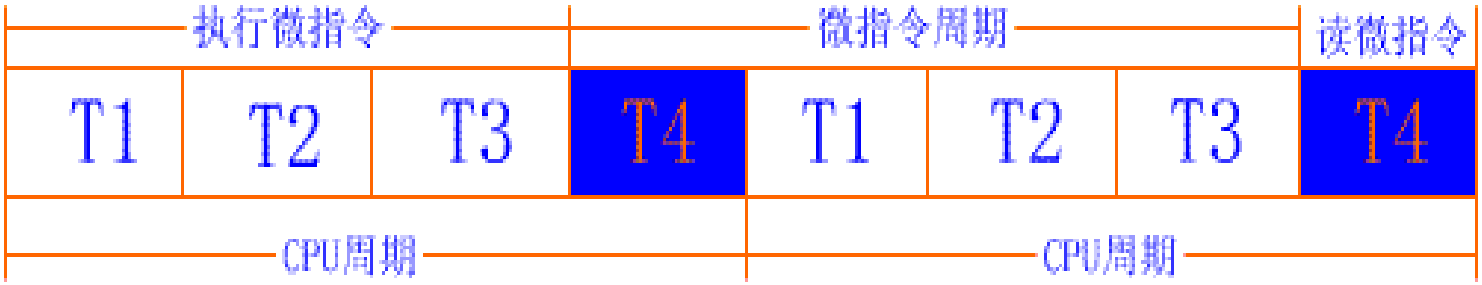


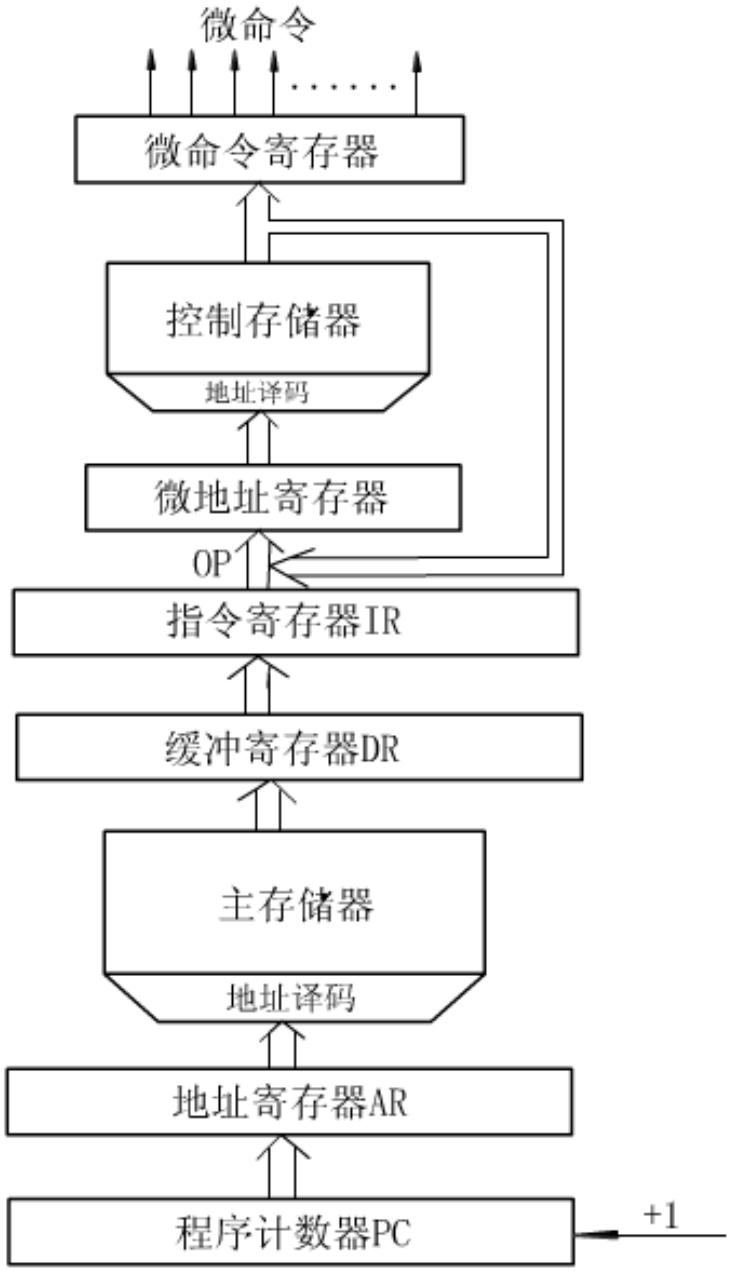
图5.27　CPU周期与微指令周期的关系

机器指令与微指令的关系

- 机器指令和取微指令之间到底是什么关系?

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。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.
2. 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。
3. 每一个CPU周期就对应一条微指令。

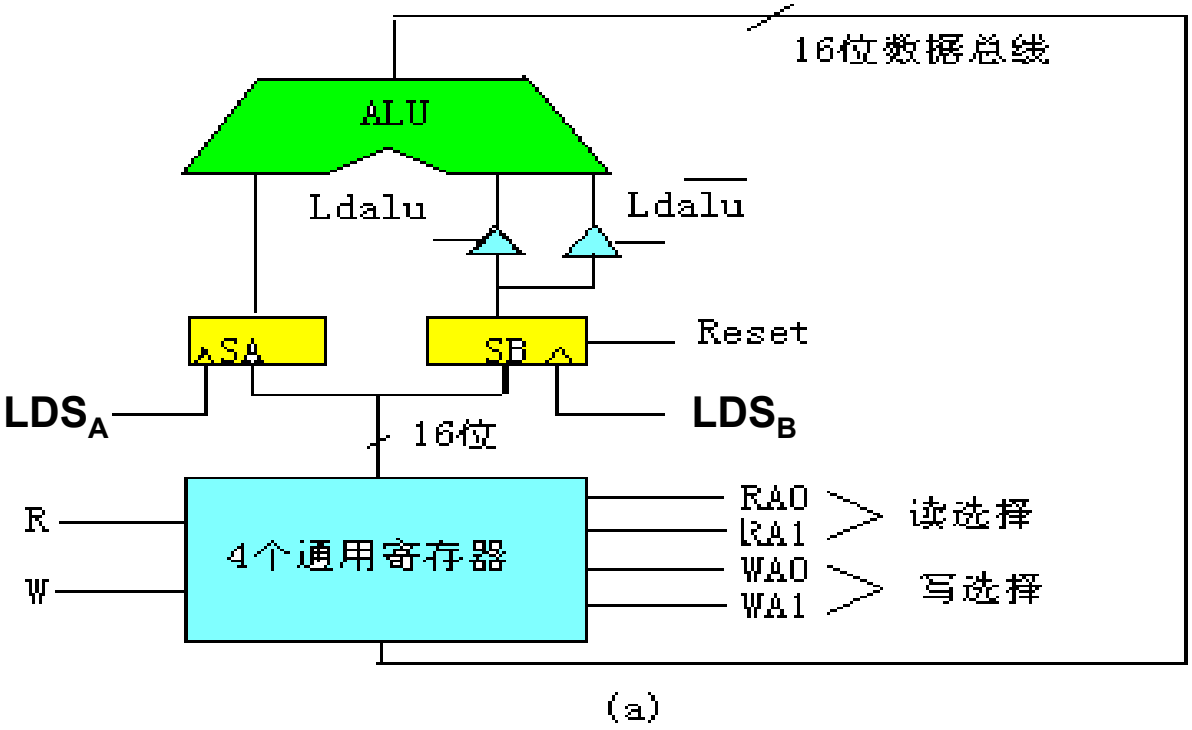
图5.27 机器指令与微指令的关系



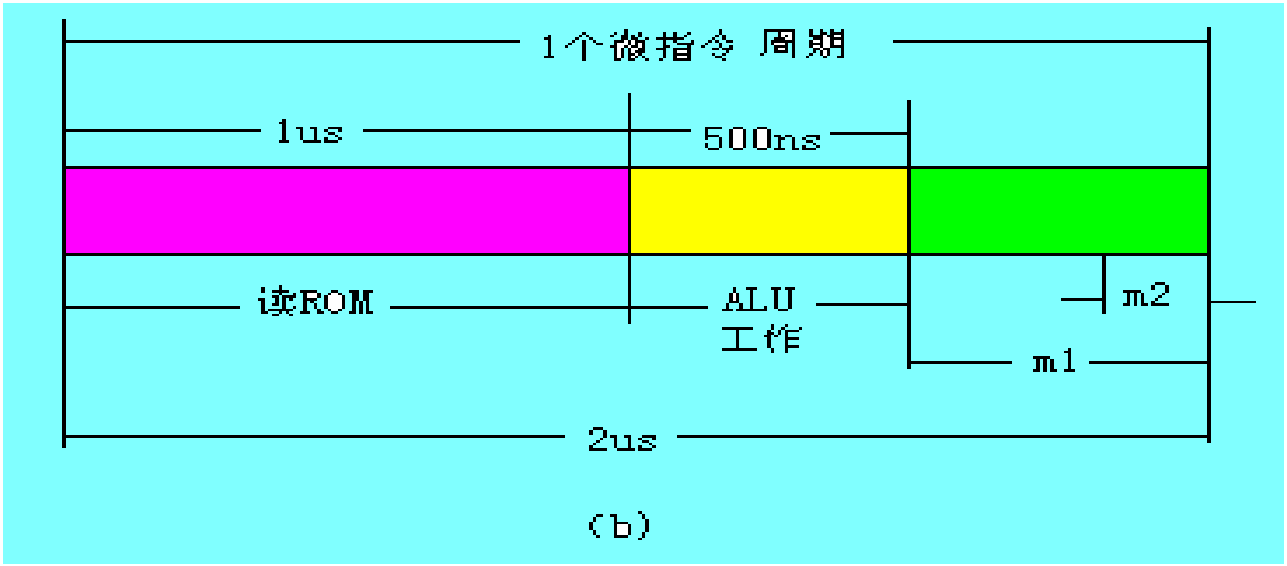
【例2】设某计算机运算器框图如图28(a)所示，其中ALU为16位的加法器(高电平工作)，SA,SB为16位暂存器。4个通用寄存器由D触发器组成，Q端输出，其读、写控制功能见下表。

读控制				写控制			
R	RA0	RA1	选择	W	WA0	WA1	选择
1	0	0	R0	1	0	0	R0
1	0	1	R1	1	0	1	R1
1	1	0	R2	1	1	0	R2
1	1	1	R3	1	1	1	R3
0	*	*	不读出	*	*	*	不写入

机器采用串行微程序控制方式，其微指令周期见图28(b)。其中读ROM是从控存中读出一条微指令时间，为1μs；ALU工作是加法器做加法运算，为500ns；m1是读寄存器时间，为500ns；m2是写寄存器的工作脉冲宽度，为100ns。


16位数据总线
ALU
Ldalu
Ldalu
SA
SB
Reset
LDSA
LDSB
16位
R
W
4个通用寄存器
RA0
RA1
读选择
WA0
WA1
写选择

(a)


1个微指令 周期
1us
500ns
读ROM
ALU
工作
m2
m1
2us

(b)

图5.28

微指令字长12位，微指令格式如下：

0	1	2	3	4	5	6	7	8	9	10	11
F1		F2		R	W	F3	F4	F5	F6	F7	F8

F1：读R0-R3的选择控制　　F2：写R0-R3的选择控制

F3：打入SA的控制信号　　F4：打入SB的控制信号

F5：打开非反相三态门的控制信号L D_{ALU}

F6：打开反相三态门的控制信号L D_{ALU}并使加法器低位加1

F7：清暂存器SB为零的Reset信号

F8：一段微程序结束，转入取机器指令的控制信号

R：寄存器读命令

W：寄存器写命令

要求：用二进制代码写出如下指令的微程序：

(1) “ADD R0，R1”指令，即(R0)+(R1)→R1

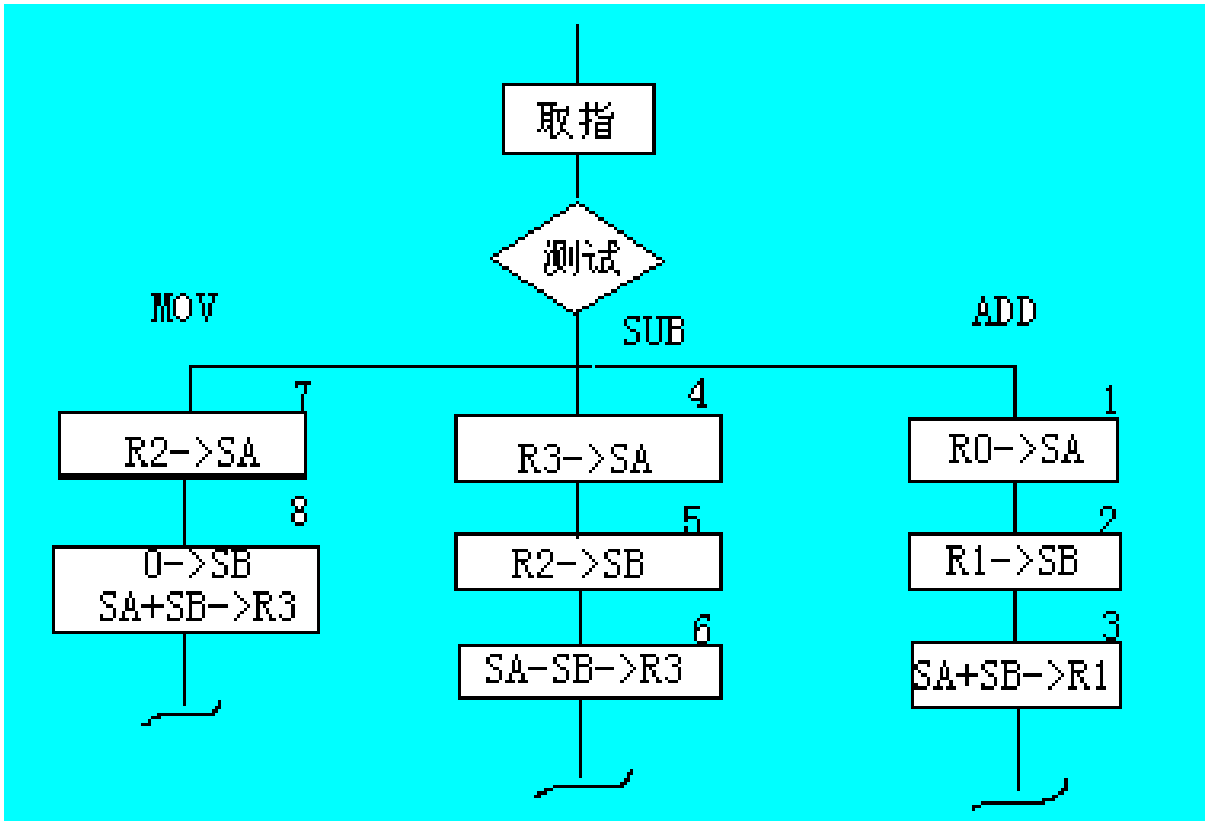
(2) “SUB R2，R3”指令，即(R3)-(R2)→R3

(3) “MOV R2，R3”指令，即(R2)→(R3)

【解】先画出三条指令的微指令的微程序流程图，如下图所示。

RA_0RA_1	WA_0WA_1	R	W	LDS_A	LDS_B	S_B-ALU	$\overline{S}_B$-ALU	Reset	~

其中未考虑“取指周期”和顺序控制问题，也即微程序仅考虑“执行周期”，微指令序列的顺序用数字标号标在每条微指令的右上角。每一框表示一条微指令。



ADD

00**10100000

01**10010000

**0101001001

SUB

11**10100000

10**10010000

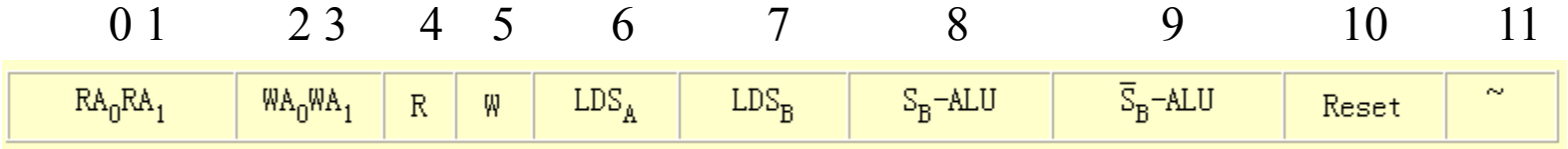
**1101000101

MOV

10**10100000

**1101001011

根据给定的微指令周期时间关系，完成ADD，SUB指令的执行动作需要3条微指令，MOV指令只需2条微指令。用二进制代码写出的三条指令的微程序列于下表中，其中*表示代码随意设置(0或1均可)。



指令	微程序代码
ADD	1. 00**10100000 2. 01**10010000 3. **0101001001
SUB	4. 11**10100000 5. 10**10010000 6. **1101000101
MOV	7. 10**10100000 8. **1101001011

主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

微程序设计技术

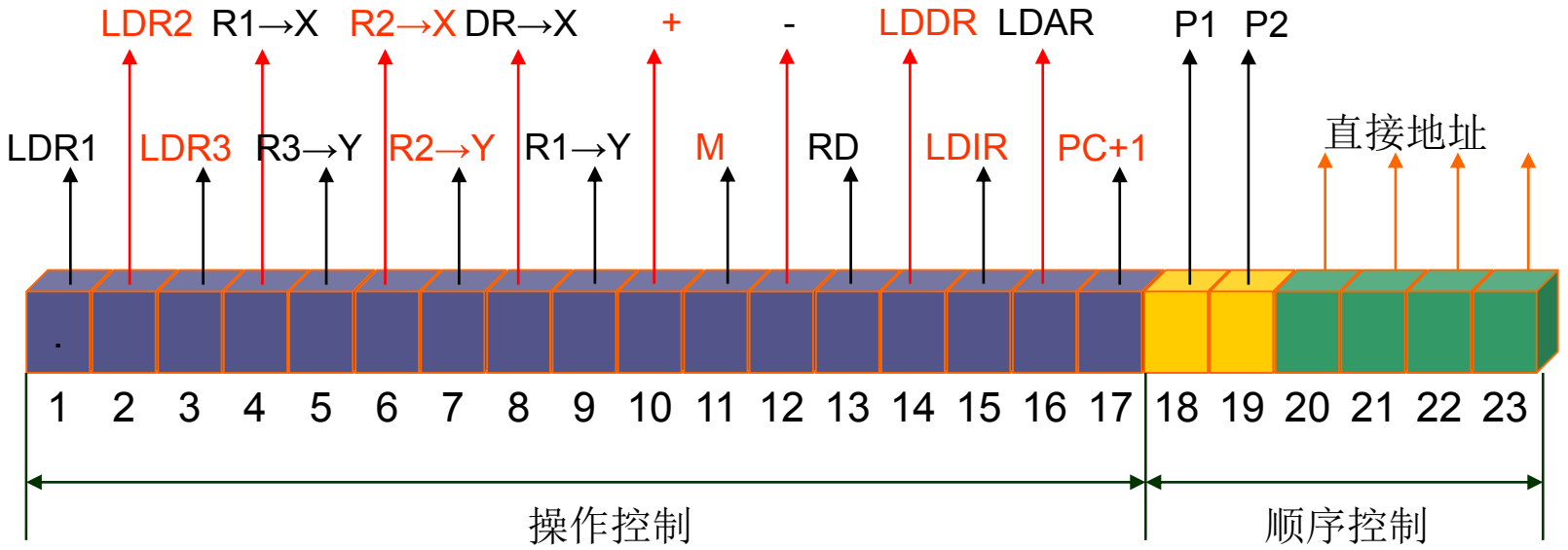
- 设计微指令结构应当追求的目标是：
  - 1）有利于缩短微指令字长度；
  - 2）有利于减小控制存储器的容量；
  - 3）有利于提高微程序的执行速度；
  - 4）有利于对微指令的修改；
  - 5）有利于微程序设计的灵活性。

微命令编码

- 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。
- 编码表示法
  - 将操作控制字段分为若干个小段，每段内采用最短编码法，段与段之间采用直接控制法。
- 混合表示法
  - 将前两种结合在一起，兼顾两者特点。

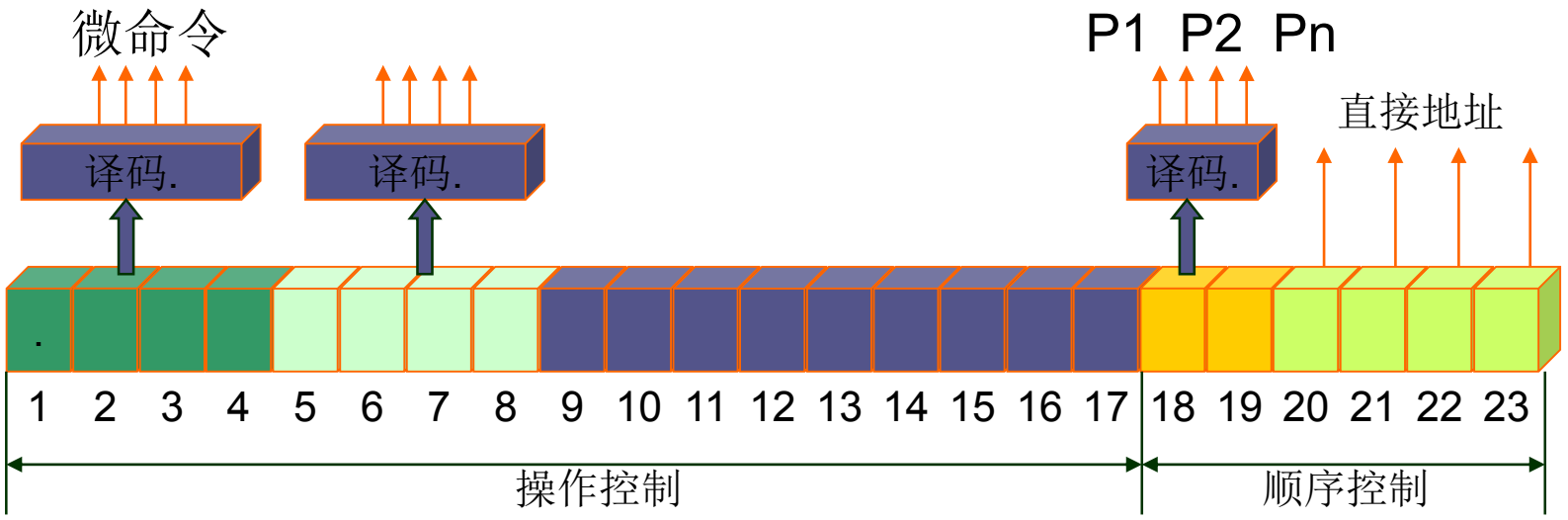
直接表示方法

- 特点：微指令中每一位代表一个微命令
- 优点：简单直观，便于输出控制，
- 缺点：字长太长，控制存储器容量大



编码表示方法

- 字段直接译码法
- 字长短，控制存储器容量小，增加了译码电路



混合表示法

- 混合表示法：把直接表示法与字段编码法相混合使用，以便能综合考虑微指令字长、灵活性和执行微程序速度等方面的要求。
- 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

微程序设计技术

- 编码注意几点：字段编码法中操作控制字段并非是任意的，必须要遵循如下的原则：

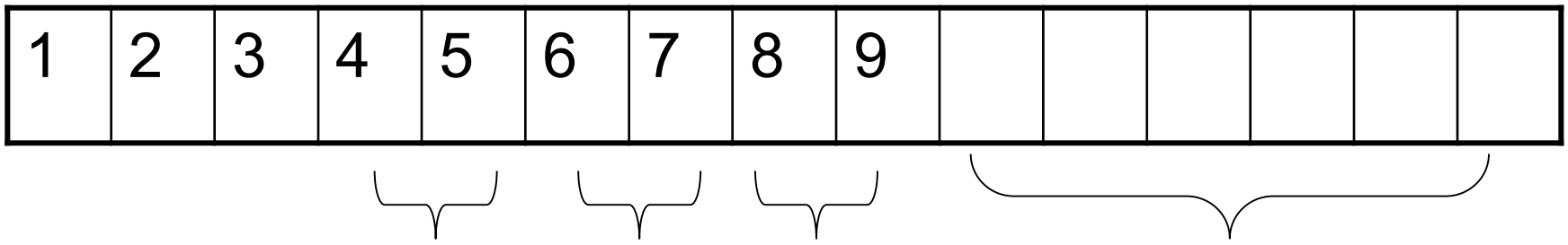
①把互斥性的微命令分在同一段内，兼容性的微命令分在不同段内。这样不仅有助于提高信息的利用率，缩短微指令字长，而且有助于充分利用硬件所具有的并行性，加快执行的速度。

②应与数据通路结构相适应。

③每个小段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。

④一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此当某字段的长度为三位时，最多只能表示七个互斥的微命令，通常用ooo表示不操作。

例 子



4、5：
00 无操作
01 R1—>X
10 R2 —> X
11 DR —>X

6、7：
00 无操作
01 R3—>Y
10 R2 —> Y
11 R1 —>Y

8、9：
00 无操作
01 ＋
10 —
11 M

混和表示法：
1、2、3位为直接表示法
4、5　6、7　8、9位为编码表示法

微地址形成方法（了解）

- 微指令执行的顺序控制问题，实际上是如何确定下一条微指令的地址问题．产生后继微 地址有两种方法：
- **1 计数器法 μPC**
  - 计数器方式：在顺序执行微指令时，后继地址由现行微地址加上一个增量来产生；在非顺序执行微指令时，必须通过转移方式，使现行微指令执行后，转去执行指定后继微地址的下一条微指令。在这种方法中，微地址寄存器通常改为计数器 。为此，顺序执行的微指令序列就必须安排在控制存储器的连续单元中。
  - 计数器方式的基本特点：微指令的顺序控制字段较短，微地址产生机构简单。但是多路并行转移功能较弱，速度较慢，灵活性较差。

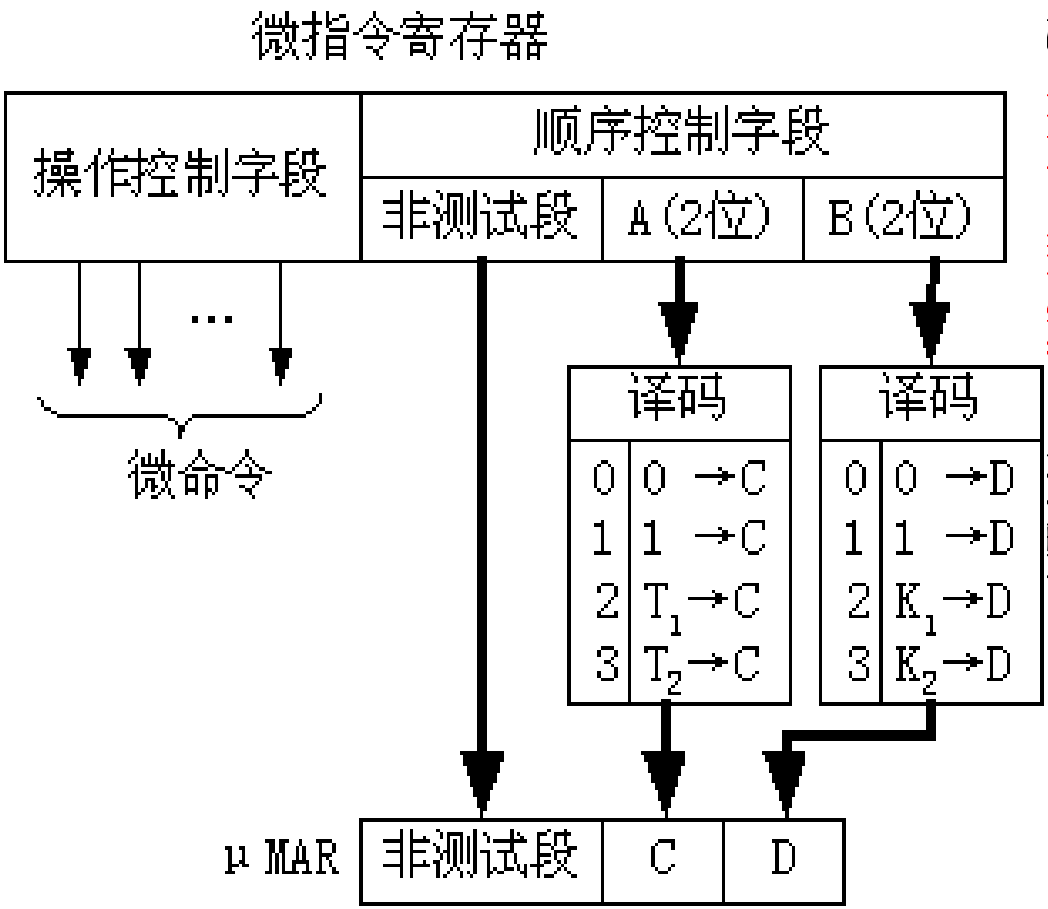
微地址形成方法

2 多路转移方式

- 一条微指令具有多个转移分支的能力称为多路转移。

例如，“取指”微指令根据操作码 OP产生多路微程序分支[illegible]方式中，当微程序[illegible]由微指令的顺序控制[illegible]寸，有若干“后选”[illegible]判字段的“判别测[illegible]选择其中一个微地址[illegible]

- 多路转移[illegible]控制字段配合，实现[illegible]度较快，但转移地址[illegible]



微指令格式

- **水平型微指令：**一次能定义并执行多个并行操作微命令的微指令

- 优点：
  - 微指令字较长，速度越快。
  - 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。

- 垂直型微指令：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。
  - 微指令字短，一般为10～20位左右。
  - 微指令的并行微操作能力有限，一条微指令包含一个微操作命令。
  - 微指令译码比较复杂。全部微命令用一个微操作控制字段进行编码，微指令执行时需行完全译码。
  - 设计用户只需注意微指令的功能，而对微命令及其选择、数据通路的结构则不用过多地考虑，因此，便于用户编制微程序。而且，编制的微程序规整、直观，便于实现设计的自动化。
  - 垂直微指令字较短，使控制存储器的横向容量少。
  - 用垂直微指令编制微程序要使用较多的微指令，微程序较长；要求控制存储器的纵向容量大。垂直微指令产生微命令要经过译码，微程序执行速度慢。
  - 不能充分利用数据通路具有多种并行操作能力

寄存器数据传送型	000	原寄存器	目的寄存器	其他
运算控制型	001	左输入源编址	右输入源编址	ALU
访问主存型	010	寄存器编址	存储器编址	读写/其他
条件转移型	011			测试条件

水平型与垂直型微指令比较

- 1）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。
- 2）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。
- 3）由水平型微指令指令的微程序，具有微指令字比较长，但微程序短的特点.垂直型微指令则相反，微指令字比较短而微程序长 。
- 4）水平型不便于用户掌握，垂直型与指令相似，易于掌握。
- 水平型微指令与机器指令差别很大，一般需要对机器的结构、数据通路、时序系统以及微命令很精通才能设计。

微程序控制器特点

- 设计规整，设计效率高
- 易于修改、扩展指令系统功能；
- 结构规整、简洁，可靠性高；
- 速度慢
  - 访存频繁
- 执行效率不高
- 用于速度要求不高、功能较复杂的机器中。
  - 特别适用于系列机

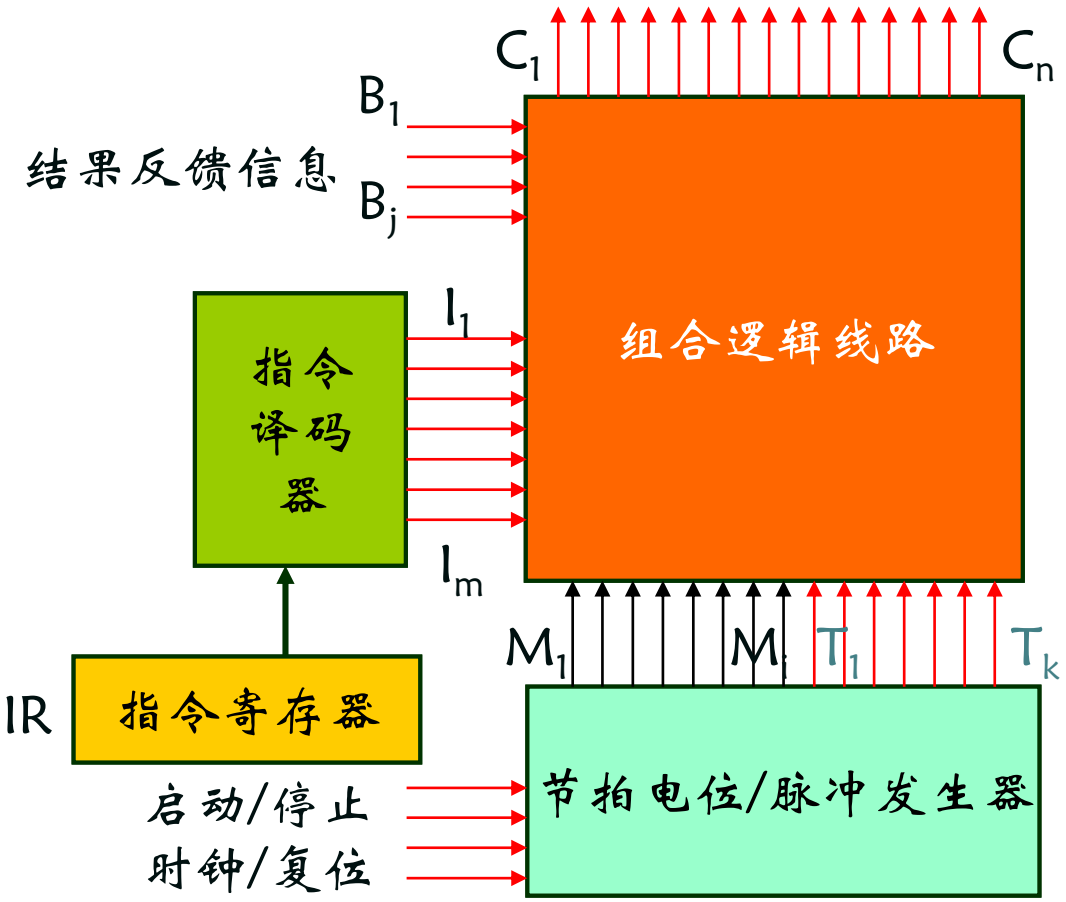
主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

硬布线控制器（了解）

- 将控制器看成产生固定时序控制信号的逻辑电路,这种逻辑电路是一种由门电路和触发器构成的复杂树形网络，称为**硬布线控制器**
- 输入信号：指令系统，时序信号，反馈信号
- 输出信号：计算机所需要的所有的控制信号
- 设计目标：用最少的元件，取得最高速度。
- 理论基础：布尔代数。
- 组成器件：门电路，触发器

硬布线控制器(组合逻辑控制器)



逻辑网络的输入信号来源有三个：

1）来自指令操作码译码器的输出 Im；

2）来自执行部件的反馈信息Bj；

3）来自时序产生器的时序信号，包括节拍电位信号 M和节拍脉冲信号T.

硬布线控制器基本原理

- 某一微**操作控制信号 C**是指令操作码译码器输出I_m、时序信号（节拍电位M_i，节拍脉冲T_k）和状态条件信号 B_j的函数，即，C=f（I_m，M_i，T_k，B_j）
- 这个控制信号是用门电路、触发器等许多器件采用组合逻辑设计方法来实现的。
- 微操作控制信号的函数表达式：

$$C_n = \sum_i (M_i \bullet T_k \bullet B_j \bullet \sum_m I_m)$$

设计过程

1. 列出所有机器指令的流程图；
2. 找出产生同一微操作控制信号的条件；
3. 写出各微操作控制信号的布尔表达式；
4. 化简各表达式；
5. 利用电路或门阵列实现。

表 3.2 组合逻辑译码表的一般格式

指令 IR	ADD	SUB	AND	……
LIR	W1	W1	W1	
M			W2	
S3	W2		W2	
S2		W2		
S1		W2	W2	
……				

相应的语言描述。以表 3.2 中的 ADD、SUB、AND 为例，可描述如下：

```
process (IR,W1,W2,W3)    --这里的 IR 实际上是指令操作码,即 IR4～IR7
    begin
        LIR <= '0';
        M <= '0';
         S3 <= '0';
        S2 <= '0';
        S1 <= '0';
        case IR is
            when "0001" =>
                LIR <= W1;
                S3 <= W2;
            when "0010" =>
                LIR <= W1;
                S2 <= W2;
                S1 <= W2;
            when "0011" =>
                LIR <= W1;
                M <= W2;
                S3 <= W2;
                S1 <= W2;
```

硬布线控制器特点

- 组成的网络复杂；
- 无规则；
- 设计和调试困难；
- 不可改变指令系统和指令功能
- 适用于VLSI
- 速度快（其原因是微程序控制中每条微指令都要从控存中读取一次，影响了速度，而组合逻辑控制主要取决于电路延迟。）

主要内容

- CPU的功能和组成
- 控制器控制原理
- 指令周期（★★★）
- 时序产生器和控制方式
- 微程序控制器（★★★）
- 微程序设计技术
- 硬布线控制器
- 流水线处理器

流水线原理

- 1.时间并行

  把任务分成若干子任务，使子任务在流水线的各阶段并发地执行。--------时间上并行性。

- 2.空间并行

  资源重复　多处理器系统和多计算机系统

- 3.时间并行+空间并行

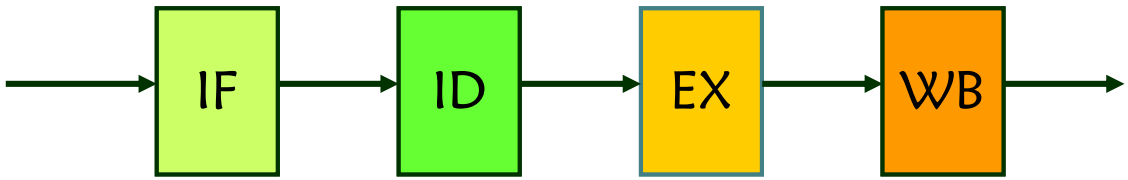
  时间重叠和资源重复的综合应用。

  奔腾CPU采用超标量流水技术，可在一个机器周期同时执行两条指令。

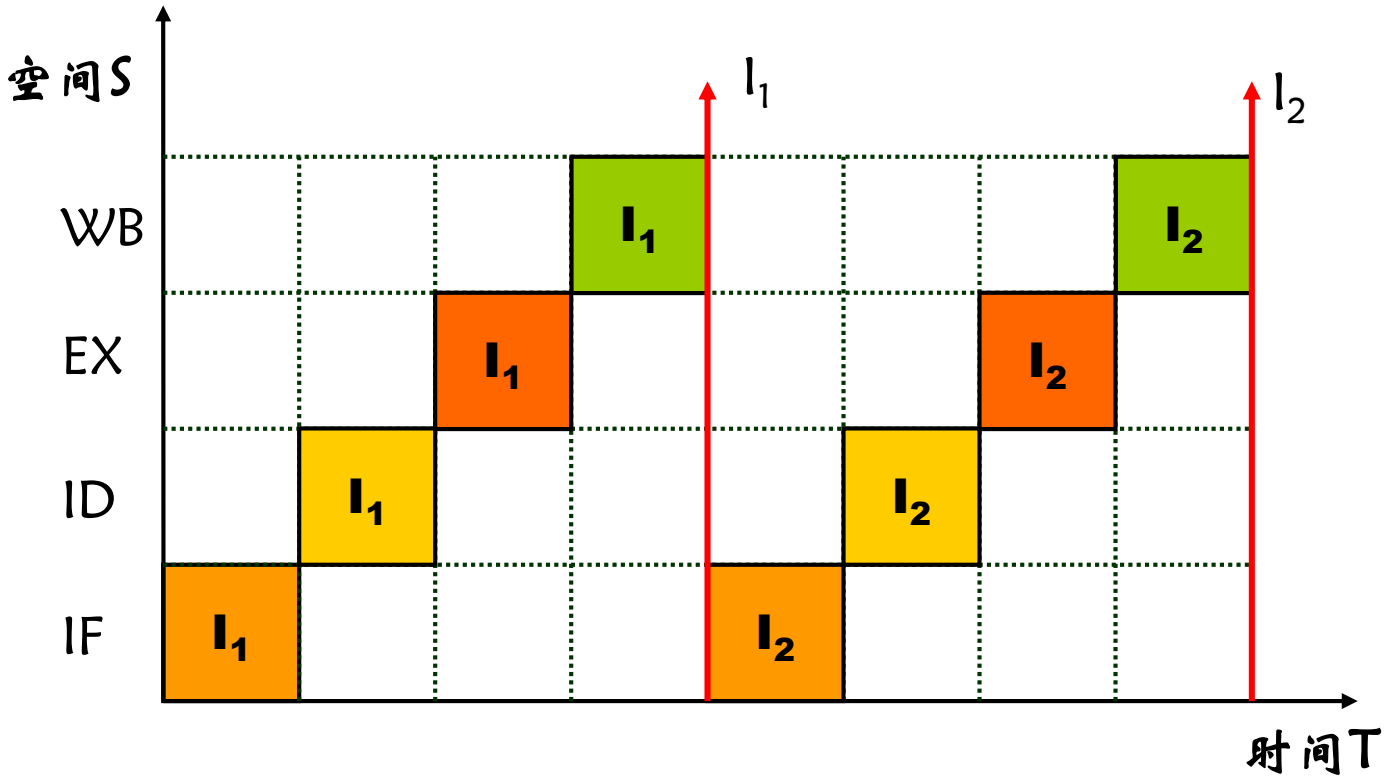
指令周期细分

- 1 取指令　　IF　(Instruction Fetch)
- 2 指令译码　ID　(Instruction Decode)
- 3 执行运算　EX　(Execution)
- 4 结果写回　WB　(Write Back)

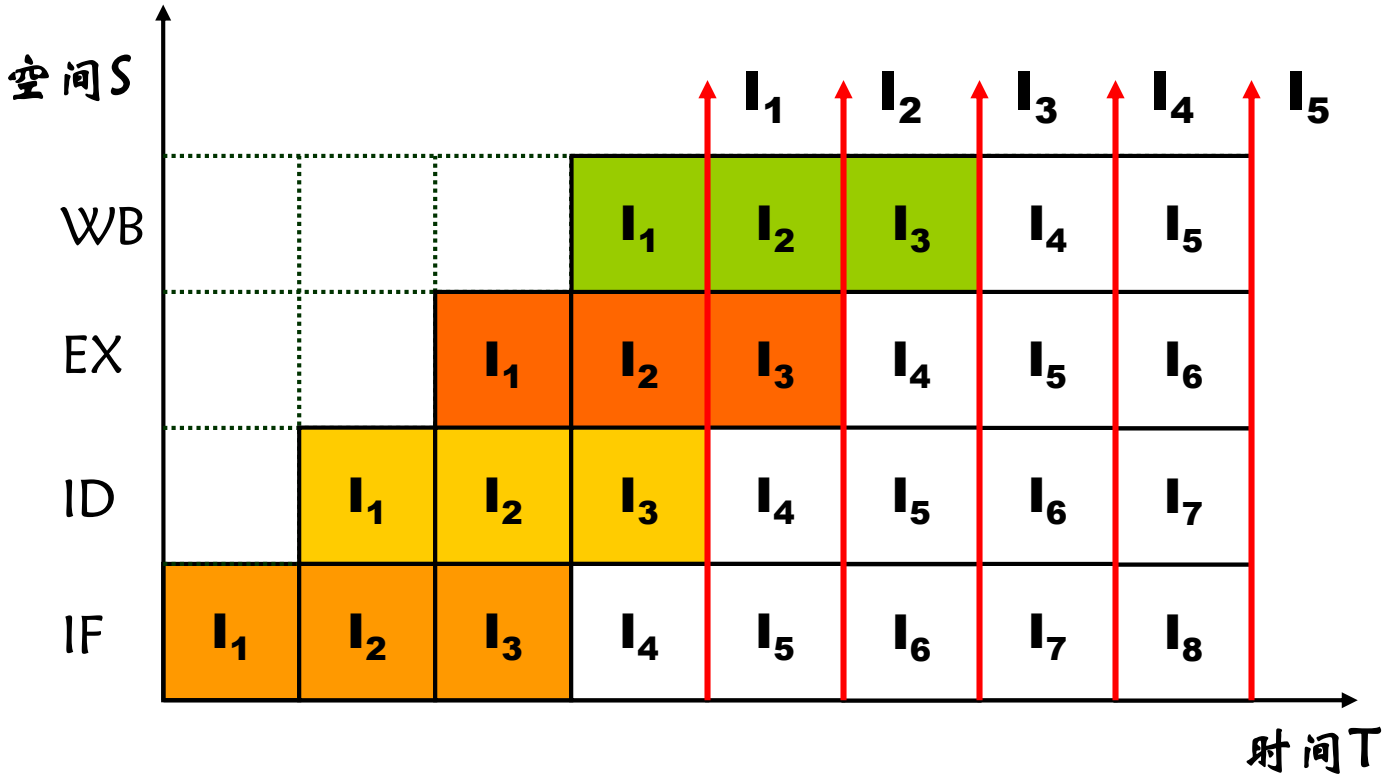
串行执行过程



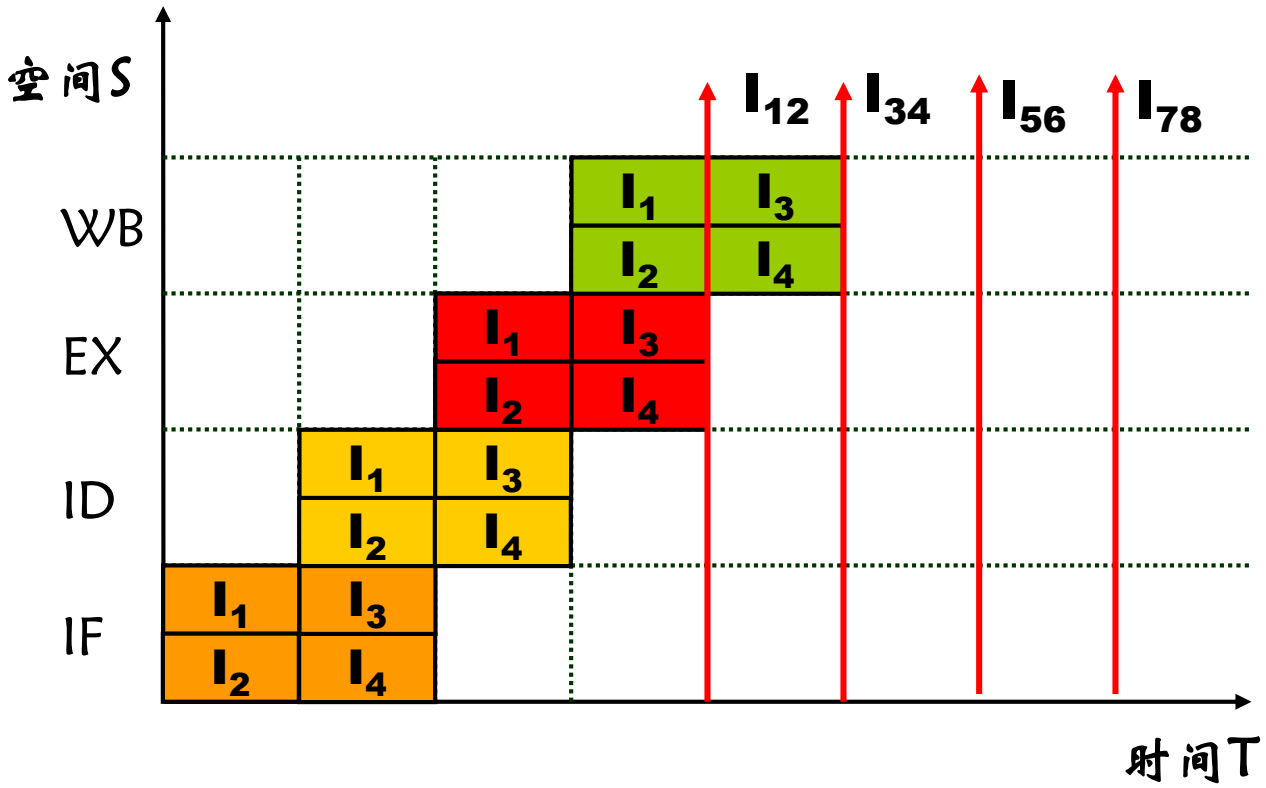
非流水线时空图



流水线时空图



超标量流水线时空图



具有两条以上的指令流水线。满载时，每一时钟周期可以执行2条指令

流水线分类

- 1.指令流水线

  取指---译码---取数---执行

- 2.算术流水线

  加法器，　　乘法器，　　快速傅里叶变换器

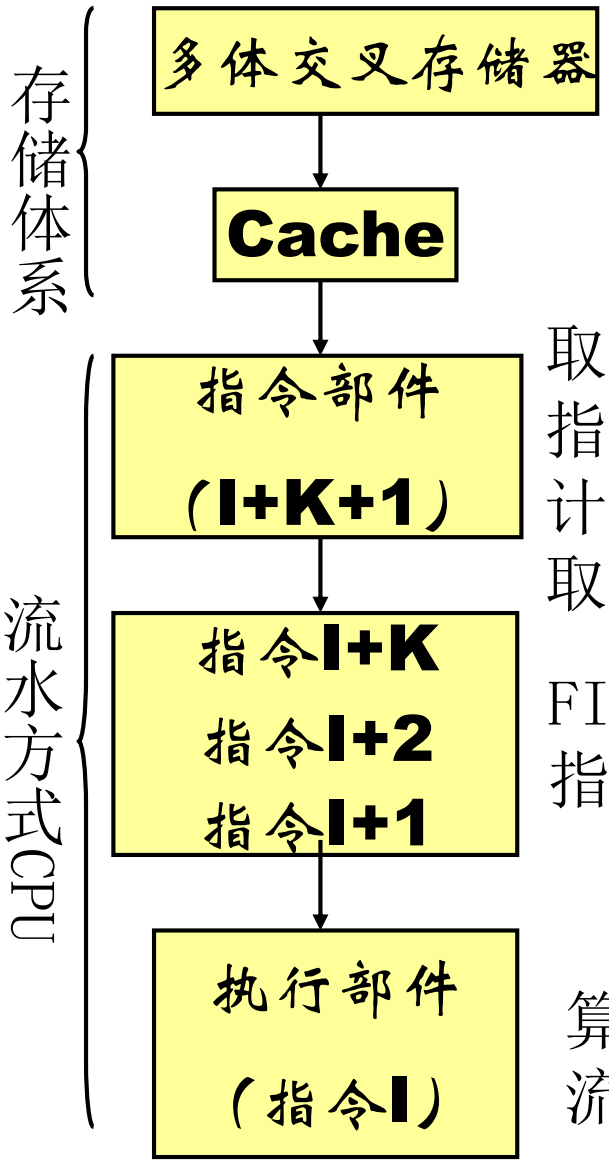
  STAR-100------4级

- 3.处理机流水线

  由一串级连的处理机组成.

  每台处理机负责某一特定任务.

流水计算机系统组成



- 指令部件
- 指令队列
- 执行部件

流水线的相关冲突

- 资源相关。例如：取操作数与取指令都需要访问主存
- 数据相关
  - 后一条指令的操作数依赖于前一条指令的执行结果。
  - 三类数据相关：先写后读相关RAW,先读后写相关WAR，写写相关WAW。
- 控制相关
  - 转移指令使得流水线发生中断。
  - 两种处理方法：1）延迟转移法，编译程序实现，发生转移时，不排空指令流水线，让已经进入流水线的指令继续完成，减少损失时间片。2）预测转移法，硬件实现，预测算法提前取指令。

习题

- 3、 8、10、 11