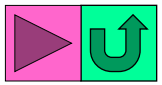
6.4 总线式数据通路与指令的执行

总线式数据通路: 数据通路中的主要部件，如通用寄存器、ALU和各种内部寄存器都连接在CPU内总线上。

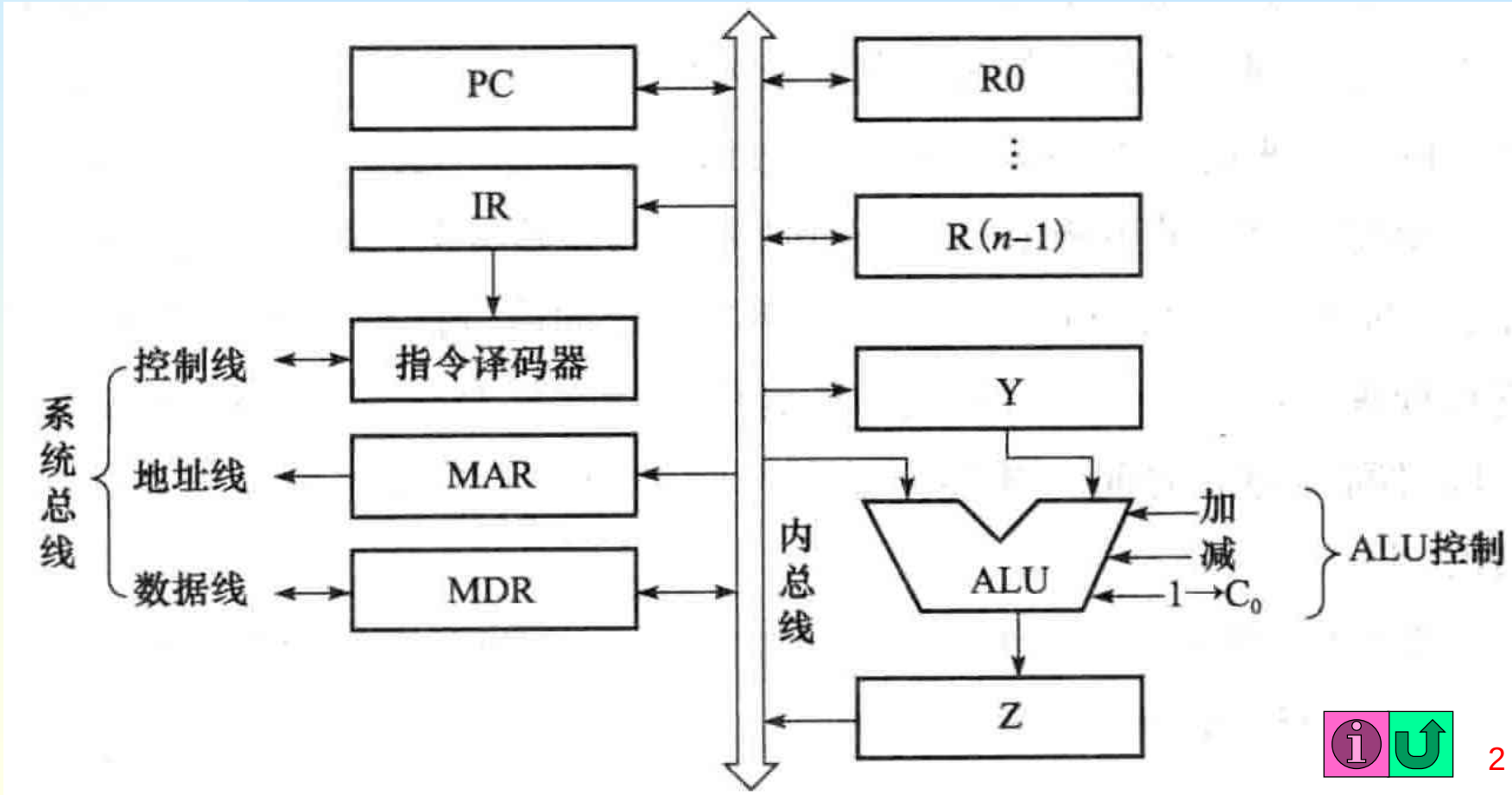
单总线数据通路

三总线数据通路



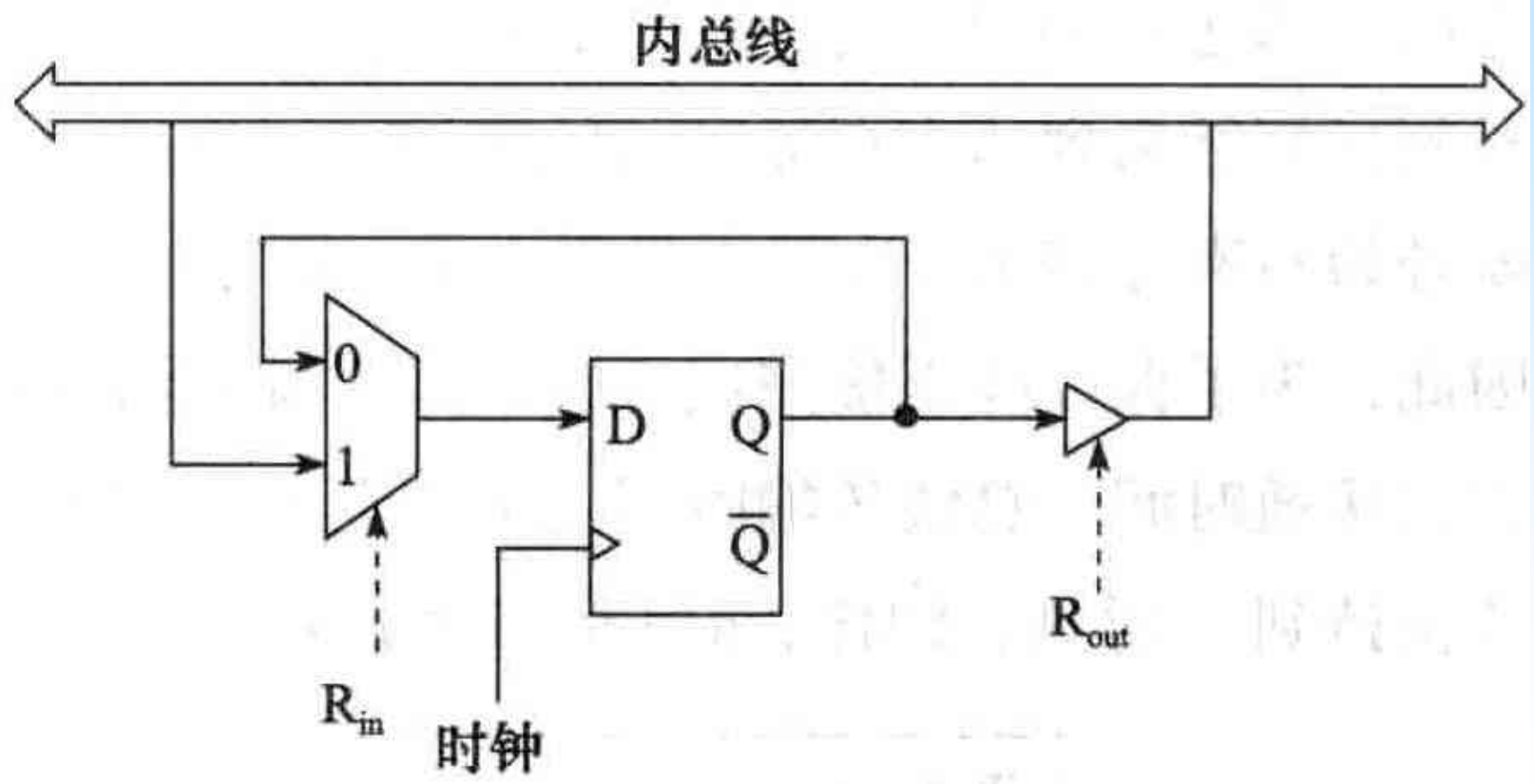
1、单总线数据通路

单总线数据通路：ALU的两个输入端和一个输出端都直接或间接连接到一个内总线上。



一位寄存器与内总线的连接和输入输出控制：

$$R_{in}=\begin{cases}1 & \text{总线}\rightarrow\text{寄存器}\\ 0 & \text{输出保持不变}\end{cases}$$

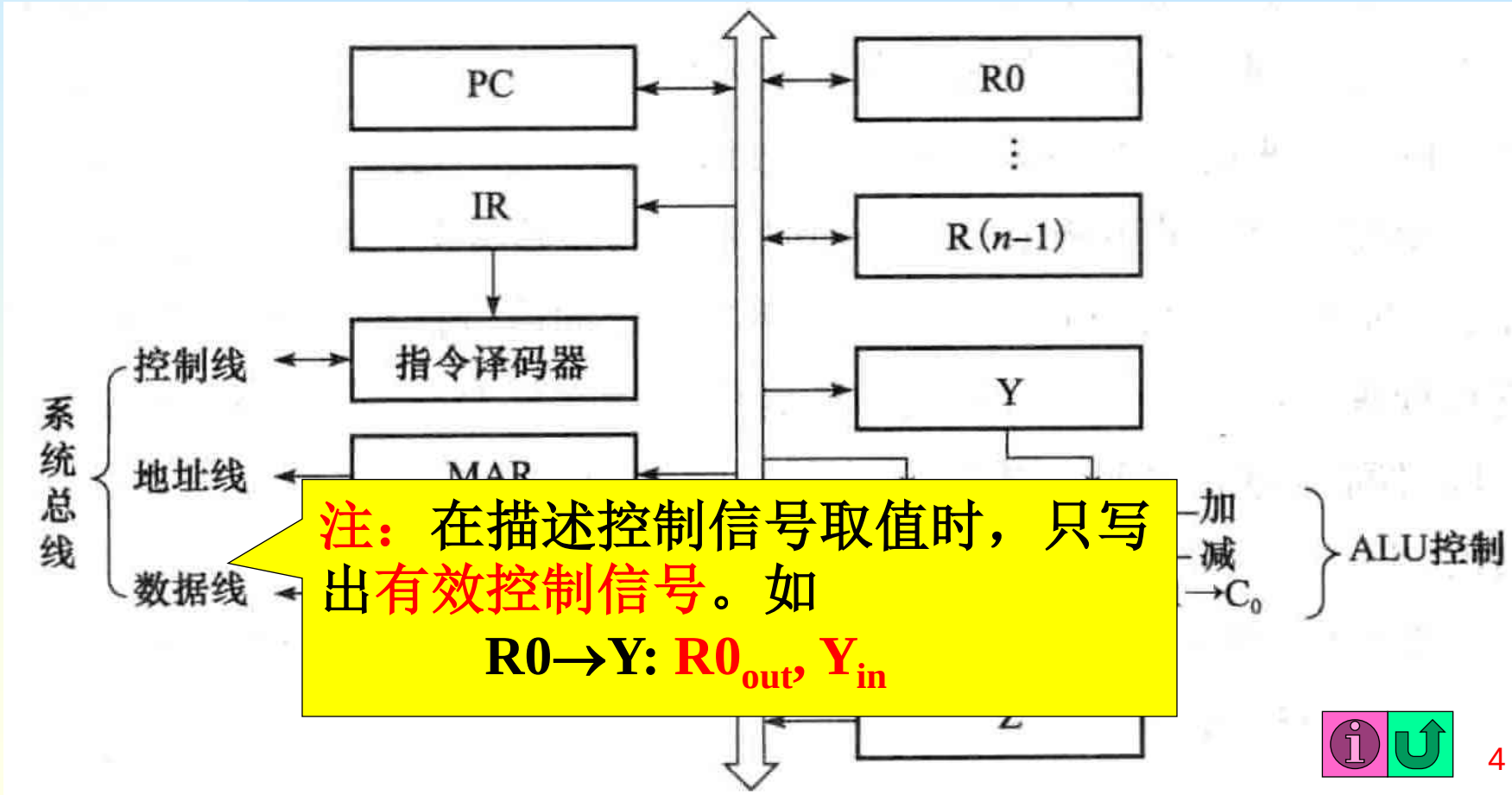


组成所有指令功能的4种基本操作过程：

（1）在通用寄存器间传送数据

如，R0→Y:　$R0_{out}$=1, Y_{in}=1, 有效控制信号。

其余寄存器的X_{out}, X_{in}信号都为0。



组成所有指令功能的4种基本操作过程：

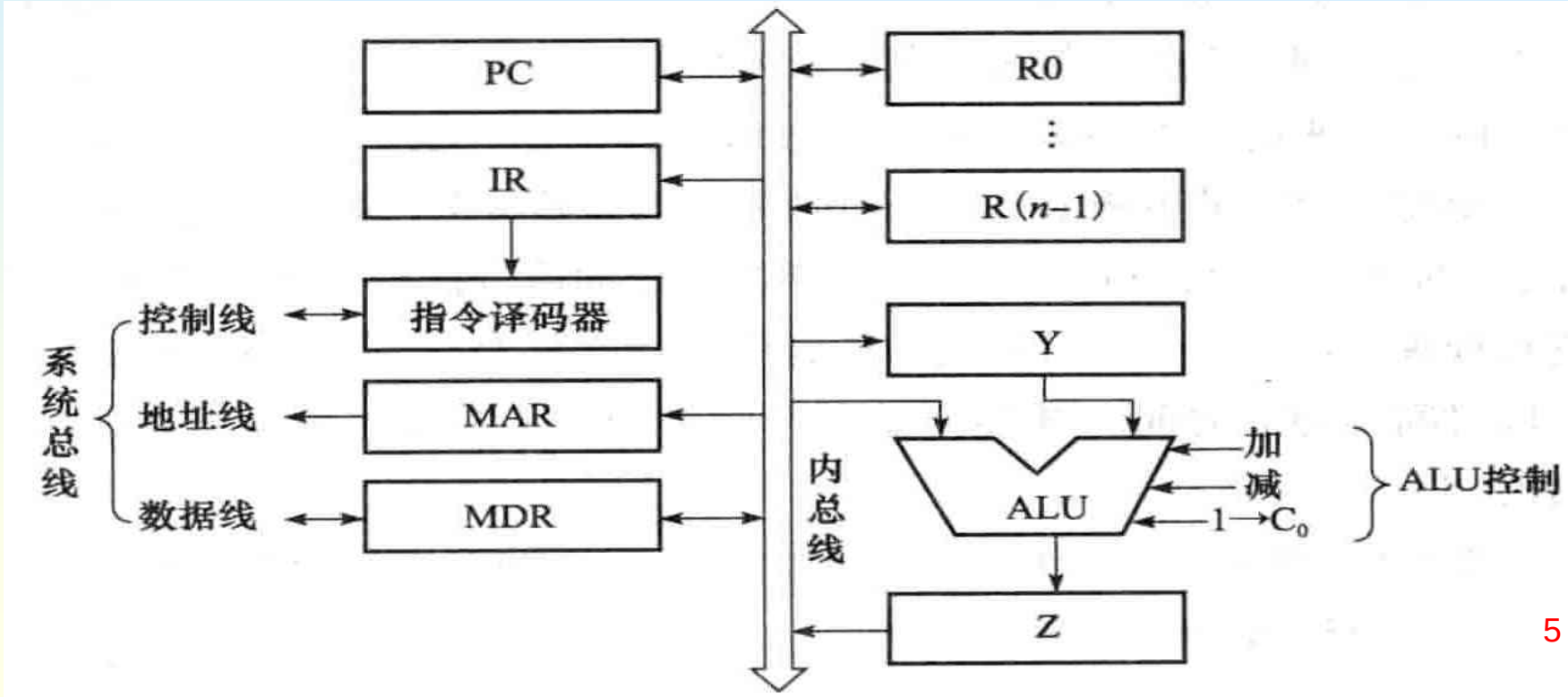
（2）完成算术或逻辑运算

如实现操作: **R[R3] ← R[R1] + R[R2]**，则有效控制信号序列:

① $R1_{out}$, Y_{in}

② $R2_{out}$, add

③ Z_{out}, $R3_{in}$



组成所有指令功能的4种基本操作过程：

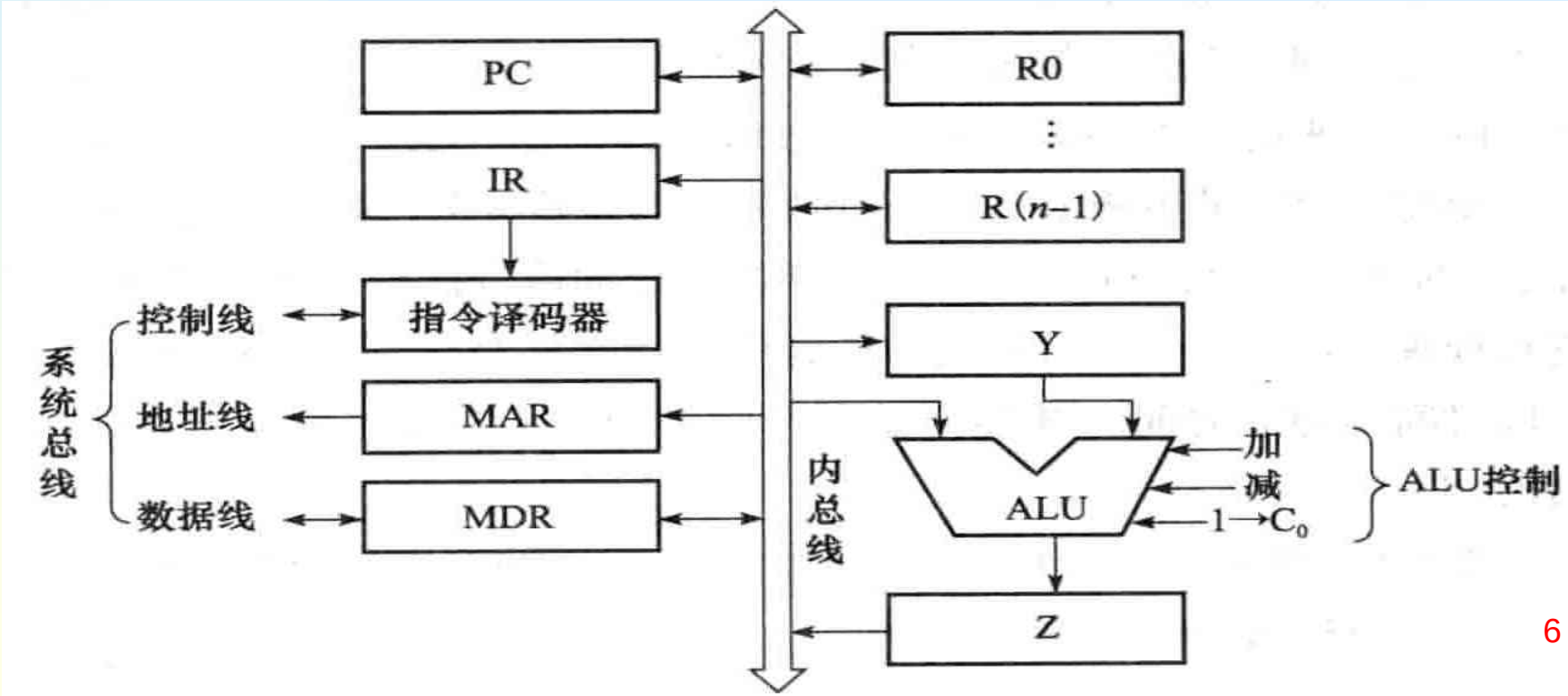
（3）从主存读取一个字（指令/数据/数据地址）

如实现操作: R[R2] ← M[R[R1]]，则有效控制信号序列:

① $R1_{out}$, MAR_{in}

② read (到MDR)

③ MDR_{out}, $R2_{in}$



组成所有指令功能的4种基本操作过程：

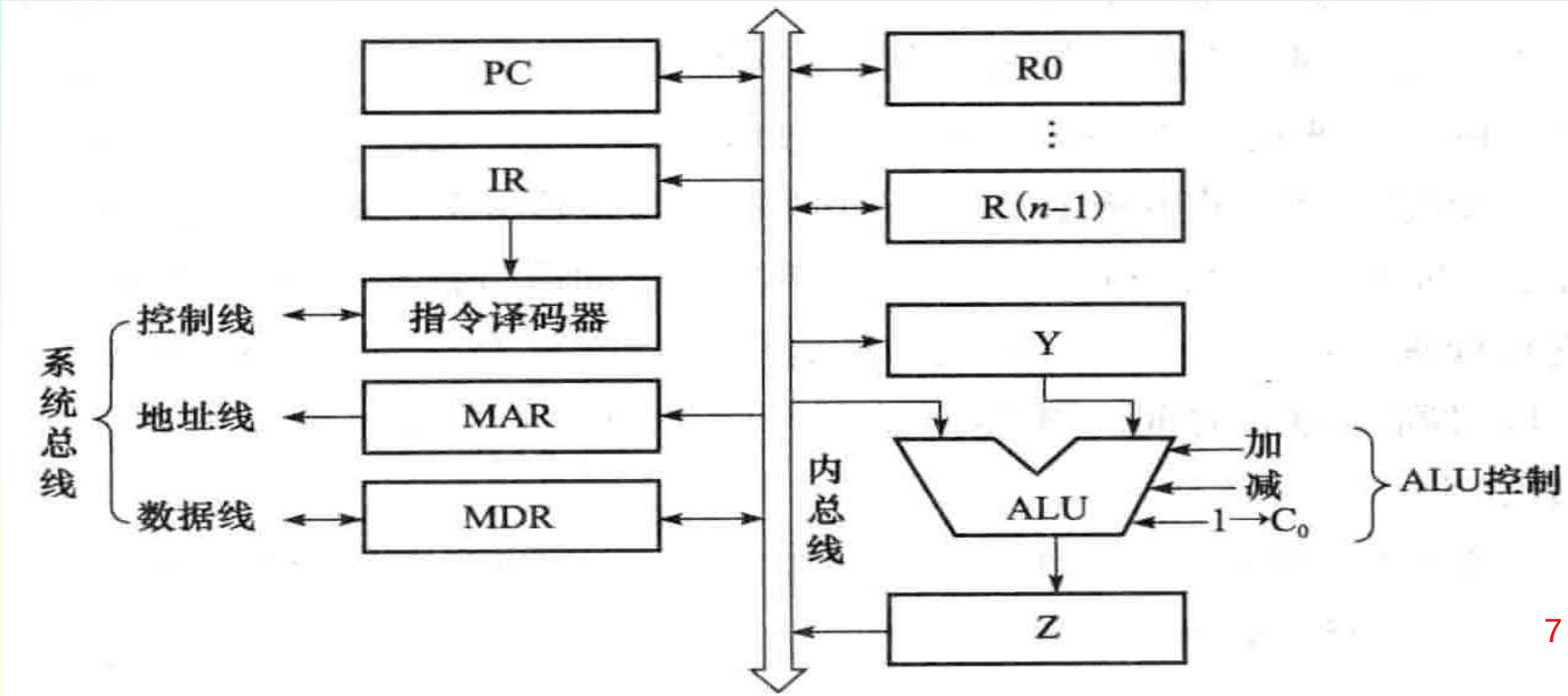
（4）把一个字（数据）写入主存

如实现操作: M[R[R2]] ← R[R1]，则有效控制信号序列:

① $R1_{out}$, MDR_{in}

② $R2_{out}$, MAR_{in}

③ write(到M[MAR])



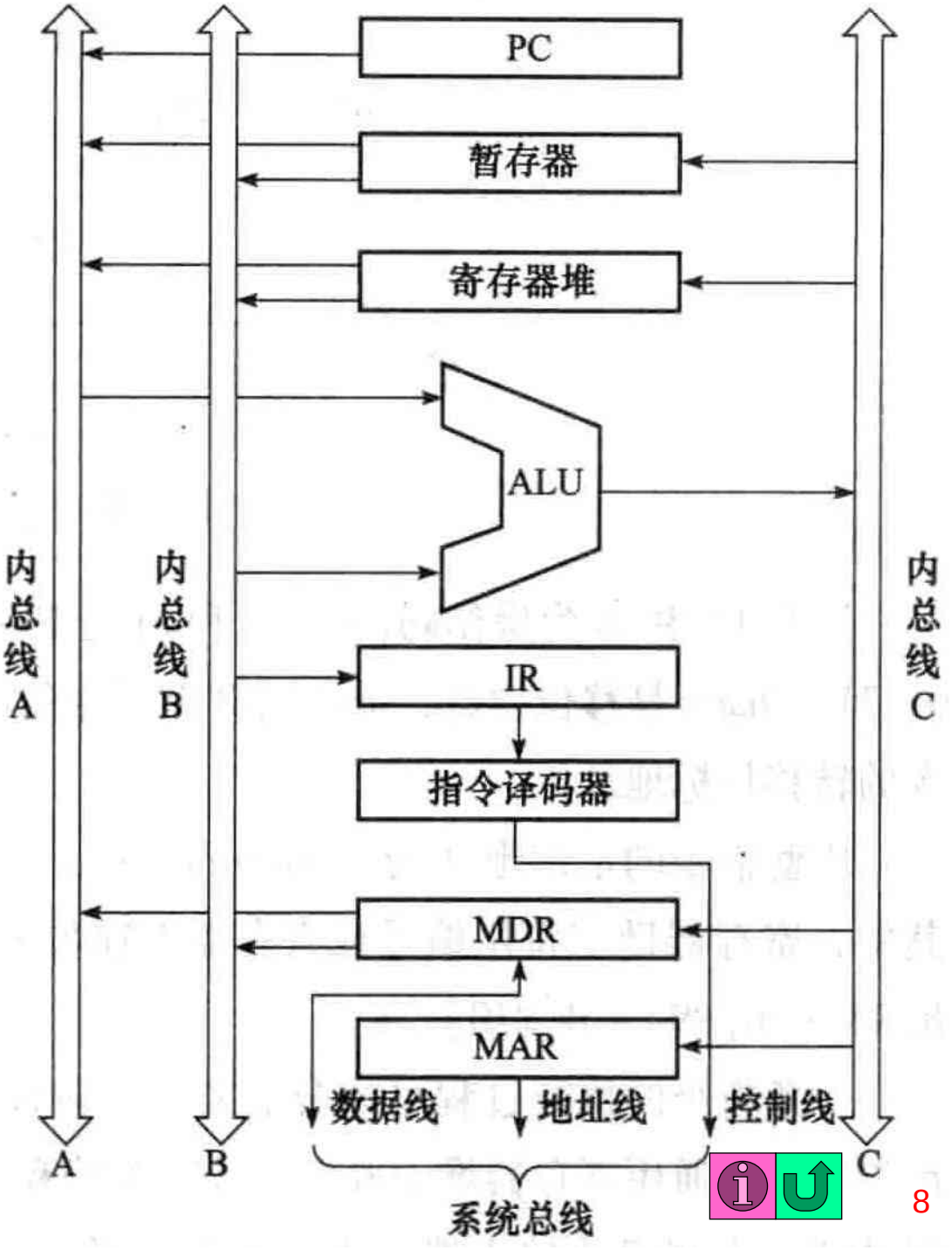
2、三总线数据通路

单总线数据通路，指令执行效率很低。

为提高计算机性能，尽量减少每条指令所用的周期数。

三总线数据通路: ALU的两个输入端和一个输出端分别连接到三个内总线上。

三总线结构：可同时传送三个数据，执行指令时需要的步骤大为减少。



（1）完成算术或逻辑运算

如: 三操作数指令,

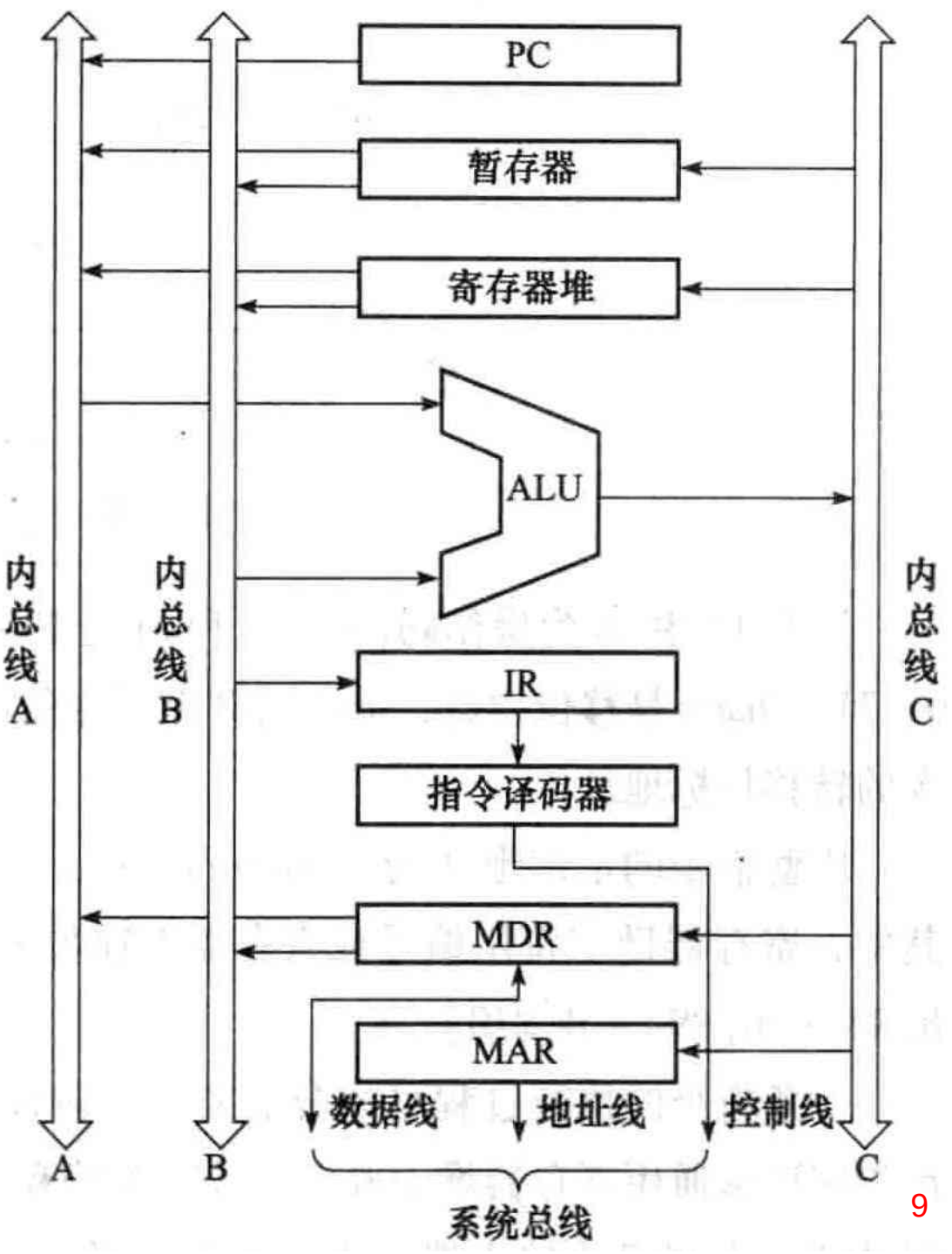
add R1, R2, R3

功能: **R[R3] ← R[R1] + R[R2]**

指令的执行可在一个时钟周期内完成。

所需的有效控制信号:

$R1_{outA}$, $R2_{outB}$, add, $R3_{inC}$



（2）在通用寄存器间传送数据

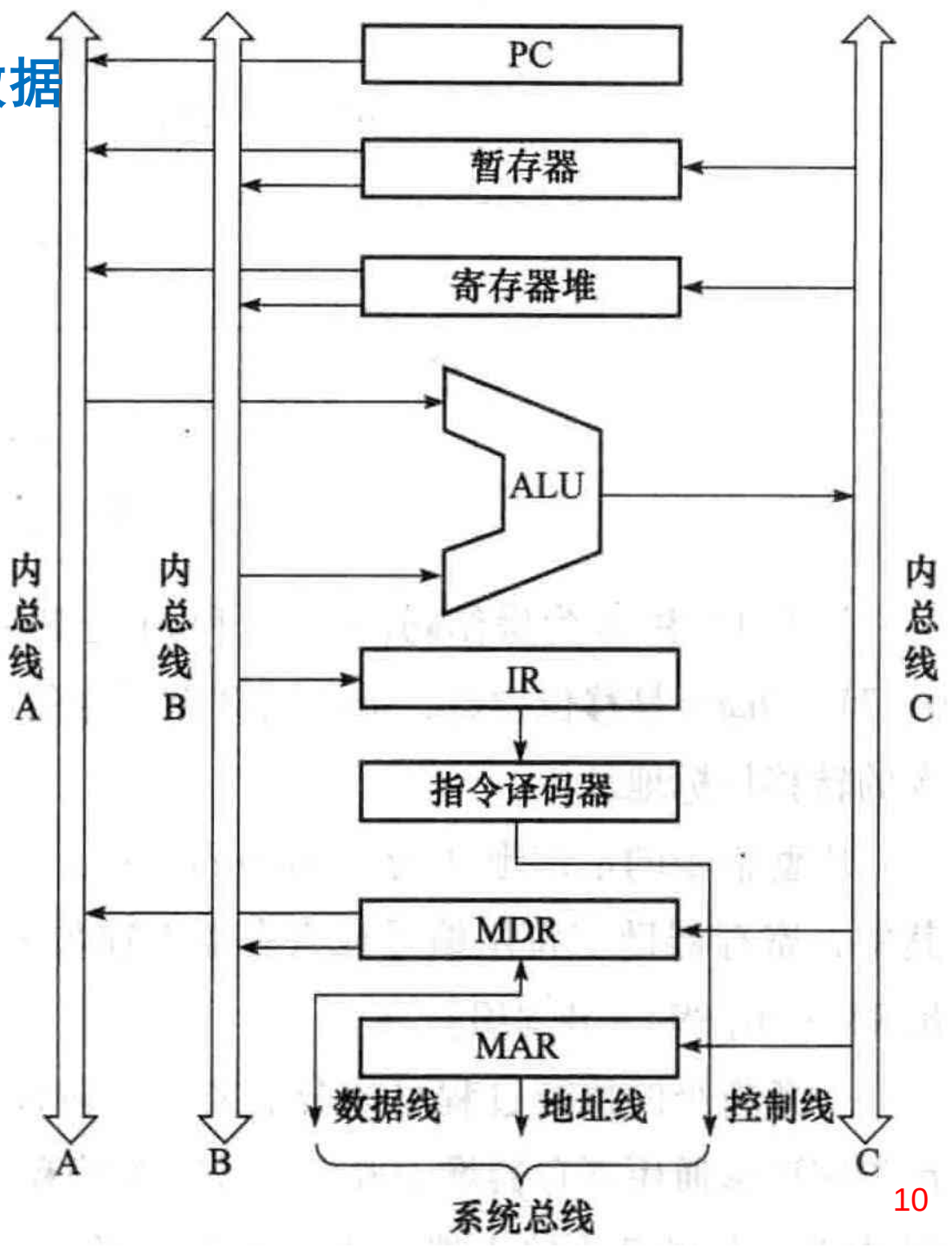
通过ALU完成。

如:实现操作R[R1] ← MDR

所需的有效控制信号:

MDR_{outA}, “直送A”, $R1_{inC}$

ALU的输入通路A、输入通路B和输出通路C，三者无冲突，无需Y、Z等锁存器。



总线式数据通路的缺点：

单总线或三总线方式连接的数据通路，很难实现指令的流水执行。

目前，几乎所有的CPU都采用流水线方式执行指令。

Intel的做法：“CISC壳RISC核”

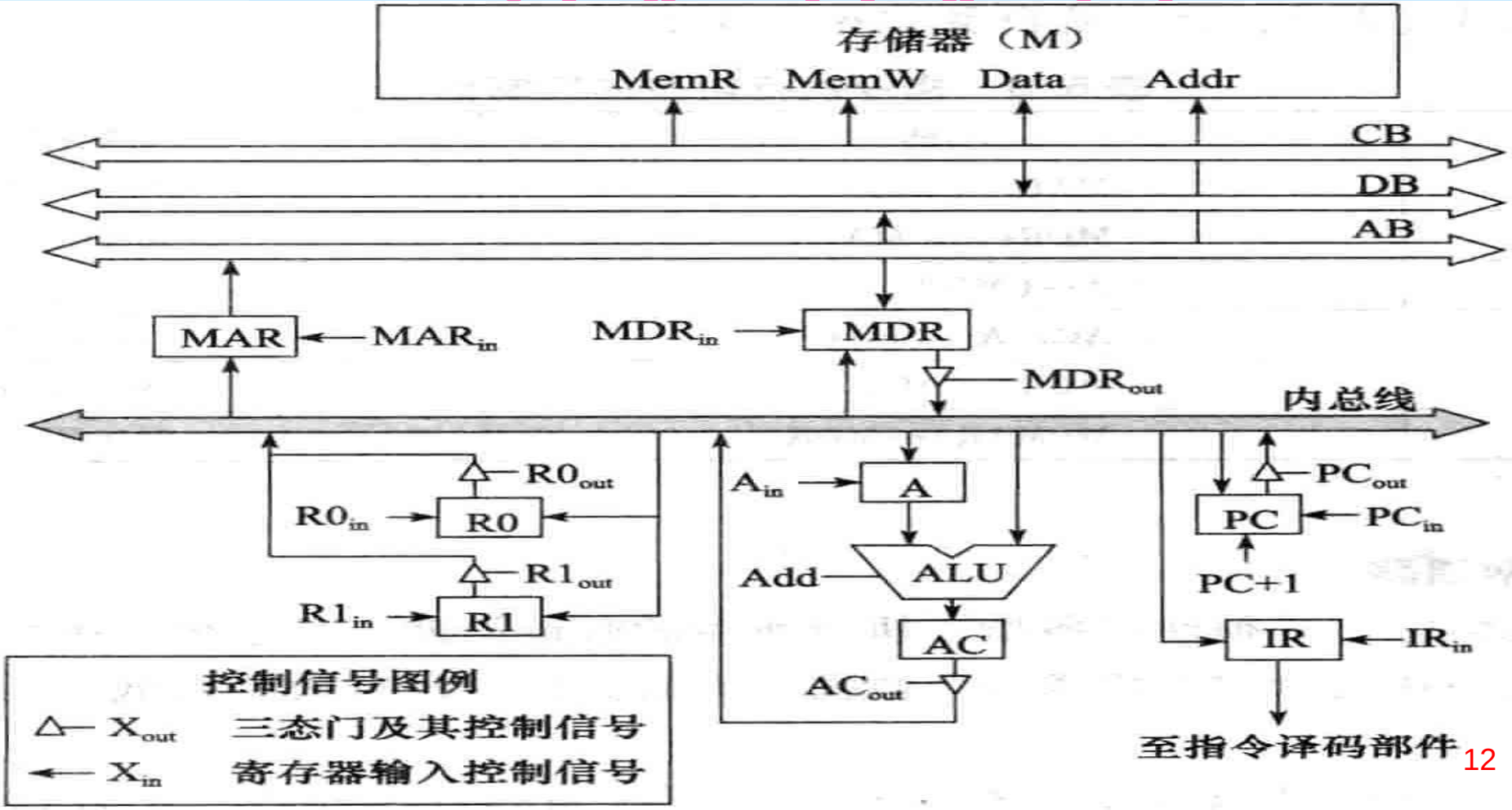
CPU对外接口： CISC 风格的指令系统。

执行指令的数据通路： RISC风格的流水线数据通路。

将一条复杂的CISC 风格的机器指令分解为多个RISC风格的微操作，微操作用流水线方式执行。

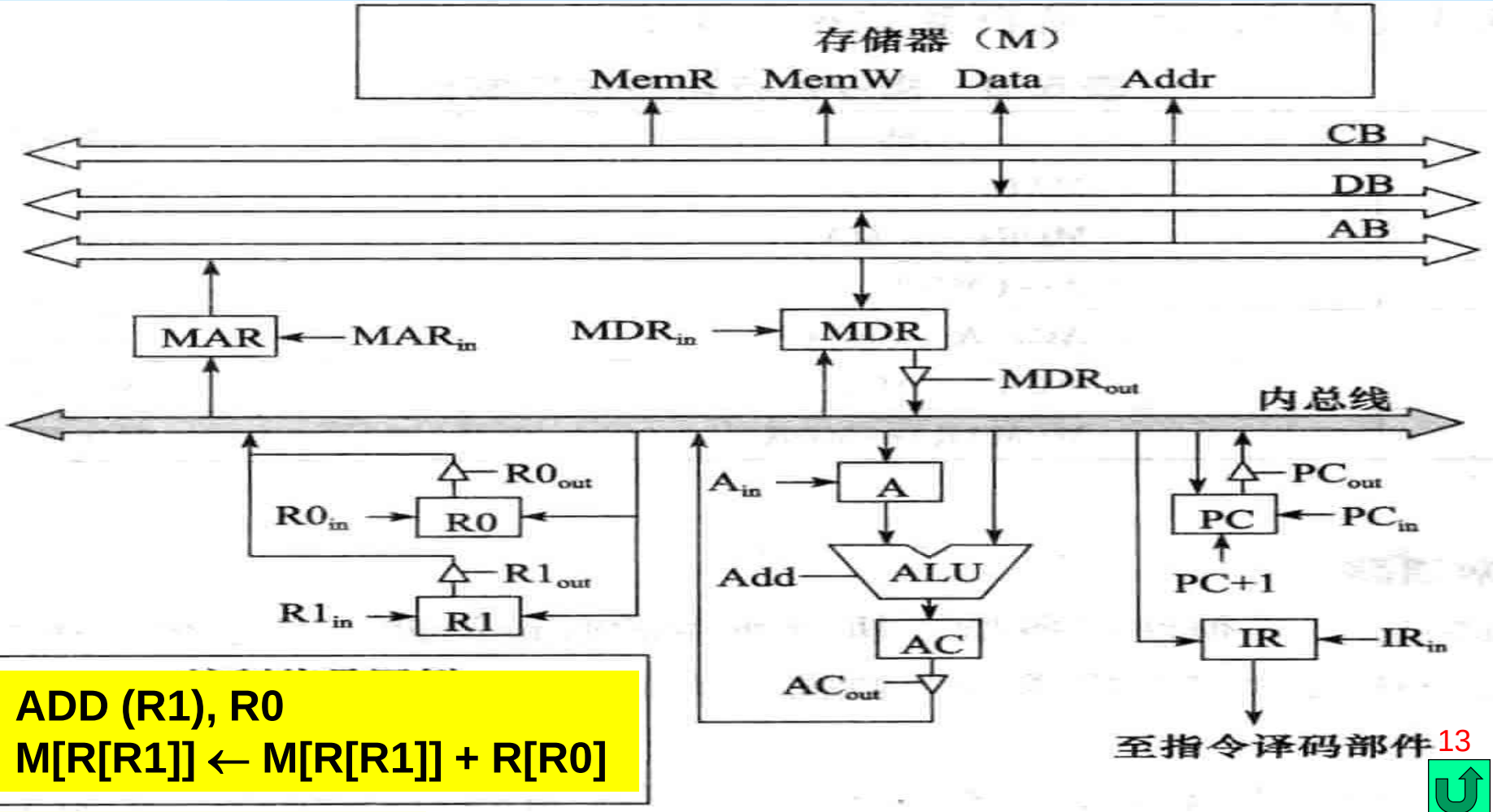
例，机器指令的读取与执行

某计算机字长16位，采用16位定长指令字结构，主存按字编址。部分数据通路结构如图所示。假设MAR的输出一直处于使能状态。给出指令ADD (R1), R0的读取与执行过程。该指令的功能为M[R[R1]] ← M[R[R1]] + R[R0]。



（1）取指令阶段的流程与控制信号：

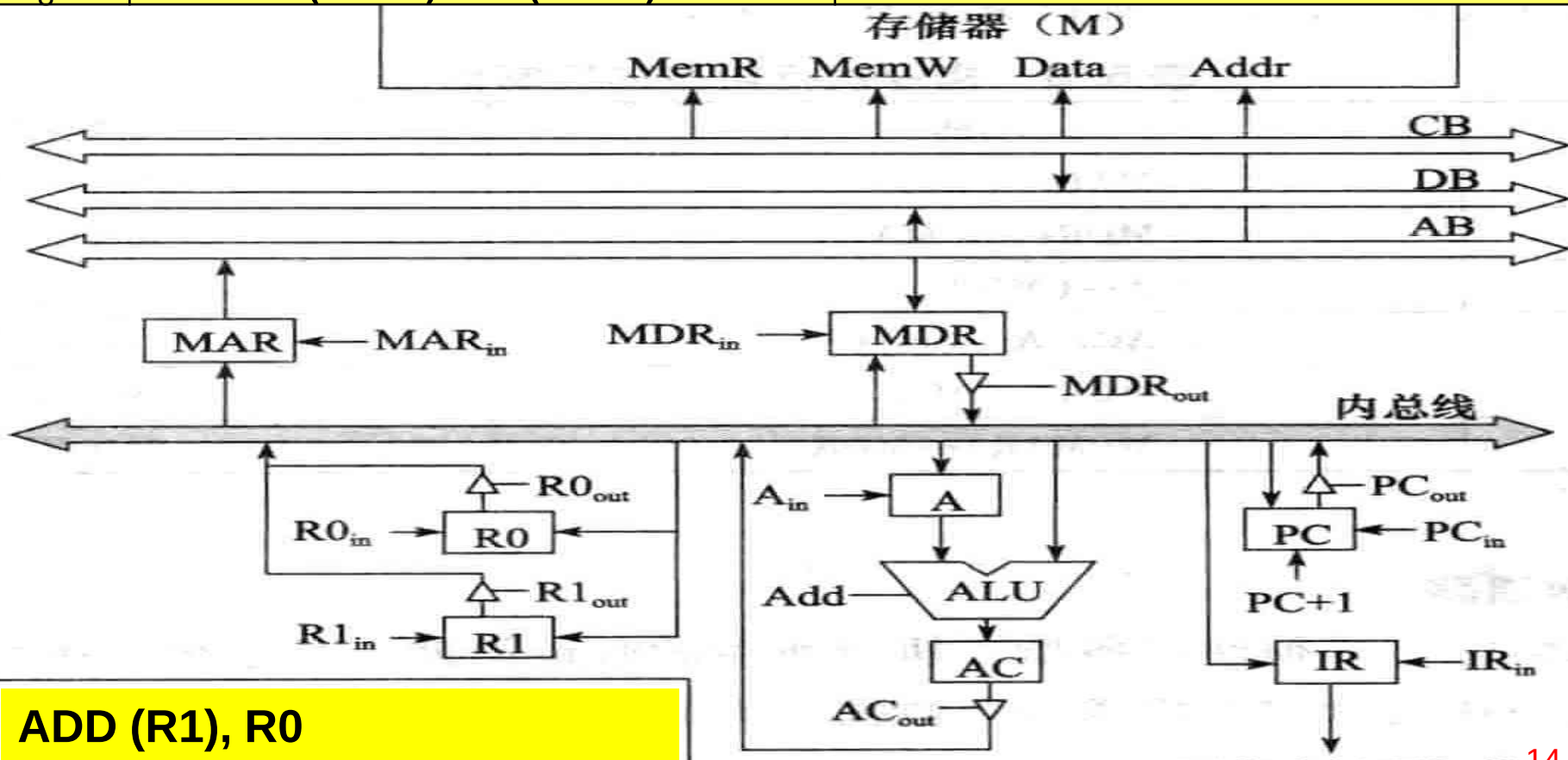
节拍	指令流程(功能)	有效控制信号
C_0	(PC) →MAR	PCout, MARin
C_1	M(MAR) →MDR, (PC)+1→PC	MemR, PC+1
C_2	MDR→IR	MDRout, IRin
C_3	指令译码	无



ADD (R1), R0
M[R[R1]] ← M[R[R1]] + R[R0]

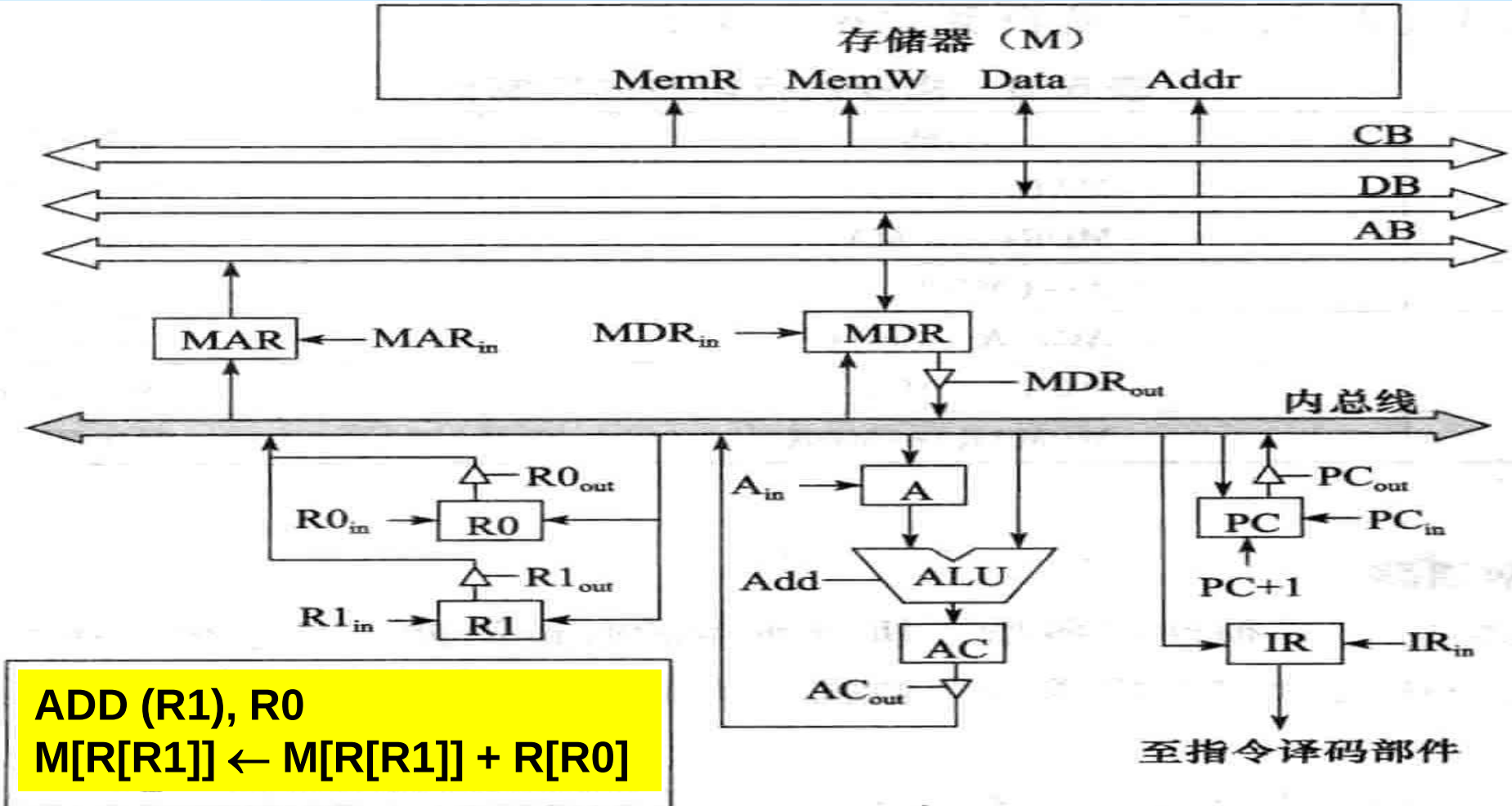
（2）指令执行阶段的流程与控制信号：

节拍	指令流程(功能)	有效控制信号
C_4	$(R_1) \rightarrow MAR$	R1out, MARin
C_5	$M(MAR) \rightarrow MDR$, $(R_0) \rightarrow A$	MemR, R0out, Ain
C_6	$(A) + (MDR) \rightarrow AC$	MDRout, Add
C_7	$(AC) \rightarrow MDR$	ACout, MDRin
C_8	$(MDR) \rightarrow M(MAR)$	MemW



ADD (R1), R0
M[R[R1]] ← M[R[R1]] + R[R0]

节拍	指令流程(功能)	有效控制信号
C_4	$(R_1) \rightarrow MAR$	R1out, MARin
C_5	$M(MAR) \rightarrow MDR$	MemR
C_6	$(MDR) \rightarrow A$	MDRout, Ain
C_7	$(A) + (R_0) \rightarrow AC$	$R0_{out}$, Add
C_8	$(AC) \rightarrow MDR$	ACout, MDRin
C_9	$(MDR) \rightarrow M(MAR)$	MemW



ADD (R1), R0
M[R[R1]] ← M[R[R1]] + R[R0]

PC系列CPU：

8086/8088：执行一条指令平均需要12个周期。

80286/80386：一条指令平均需要4.5个周期。

80486：一条指令平均需要2个周期。

Pentium：指令流水，每个周期可执行3条或更多指令。

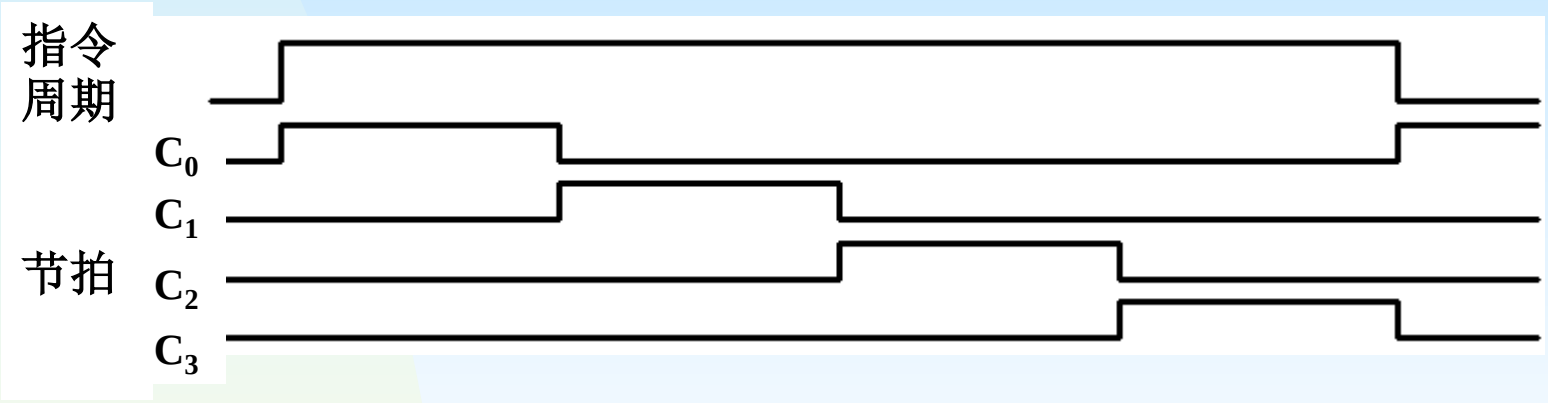
主频：CPU的工作频率。

倒数—时钟周期：　　CPU中最小的时间单位。

总线式数据通路必须有更复杂的时序控制

计算机的工作需分步执行：送地址、访存、运算等。

需要时间标志信号（节拍），为指令的执行提供定时信号。



节拍信号发生器

节拍信号发生器：产生一组在时间上有先后顺序的节拍信号。

用途：　机器执行指令时，是将一条指令分成一些有先后顺序的基本动作；

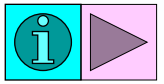
节拍信号控制这些基本动
指令的功能。

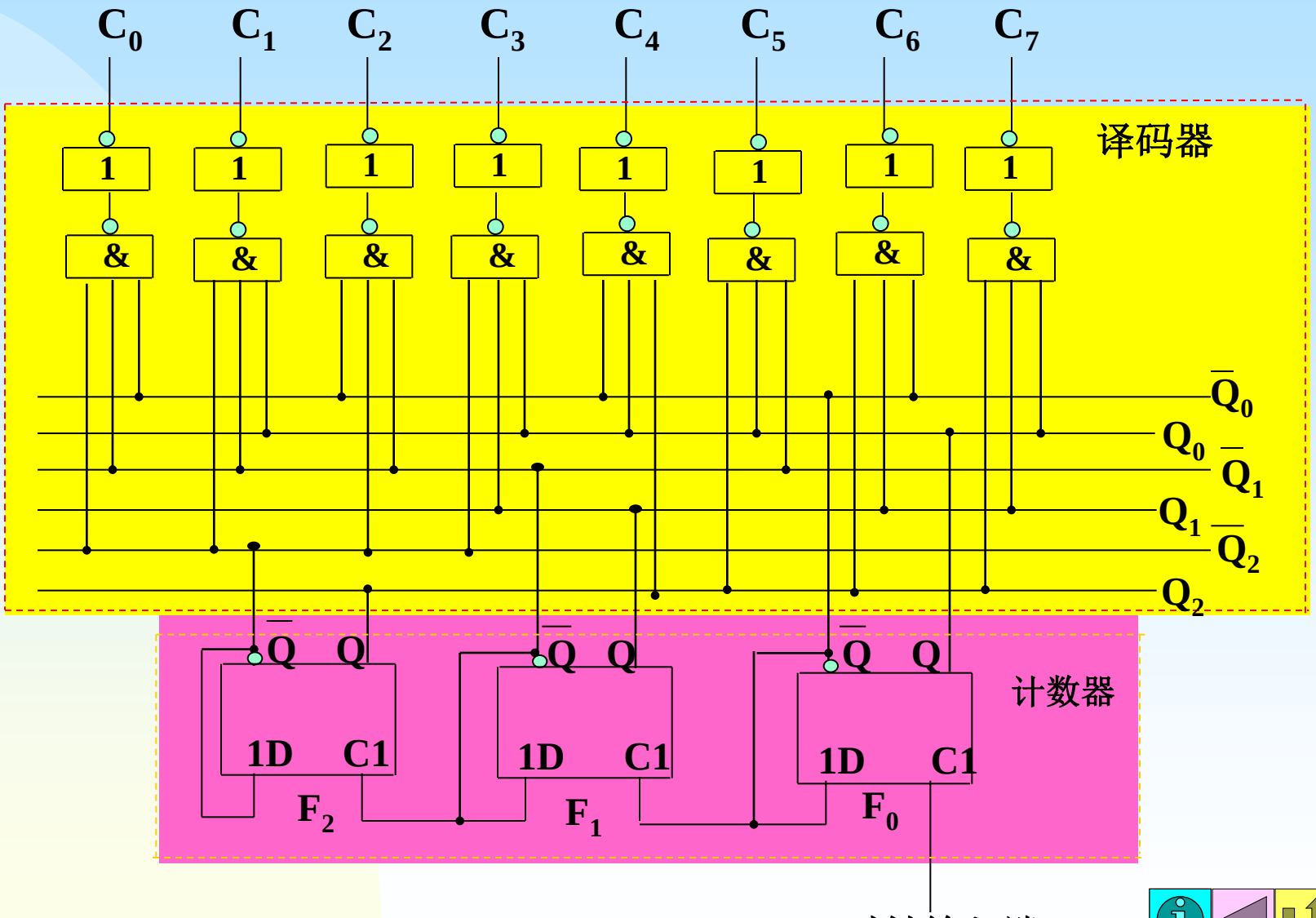
时钟周期/节拍/时钟脉冲：
Clock Period
Clock Cycle
Clock Pulse
在我们这门课里是一回事。

电路组成：

计数器：计脉冲CP的个数。

译码器：将计数器状态翻译成对应输出端的高低电平，顺序输出节拍信号。

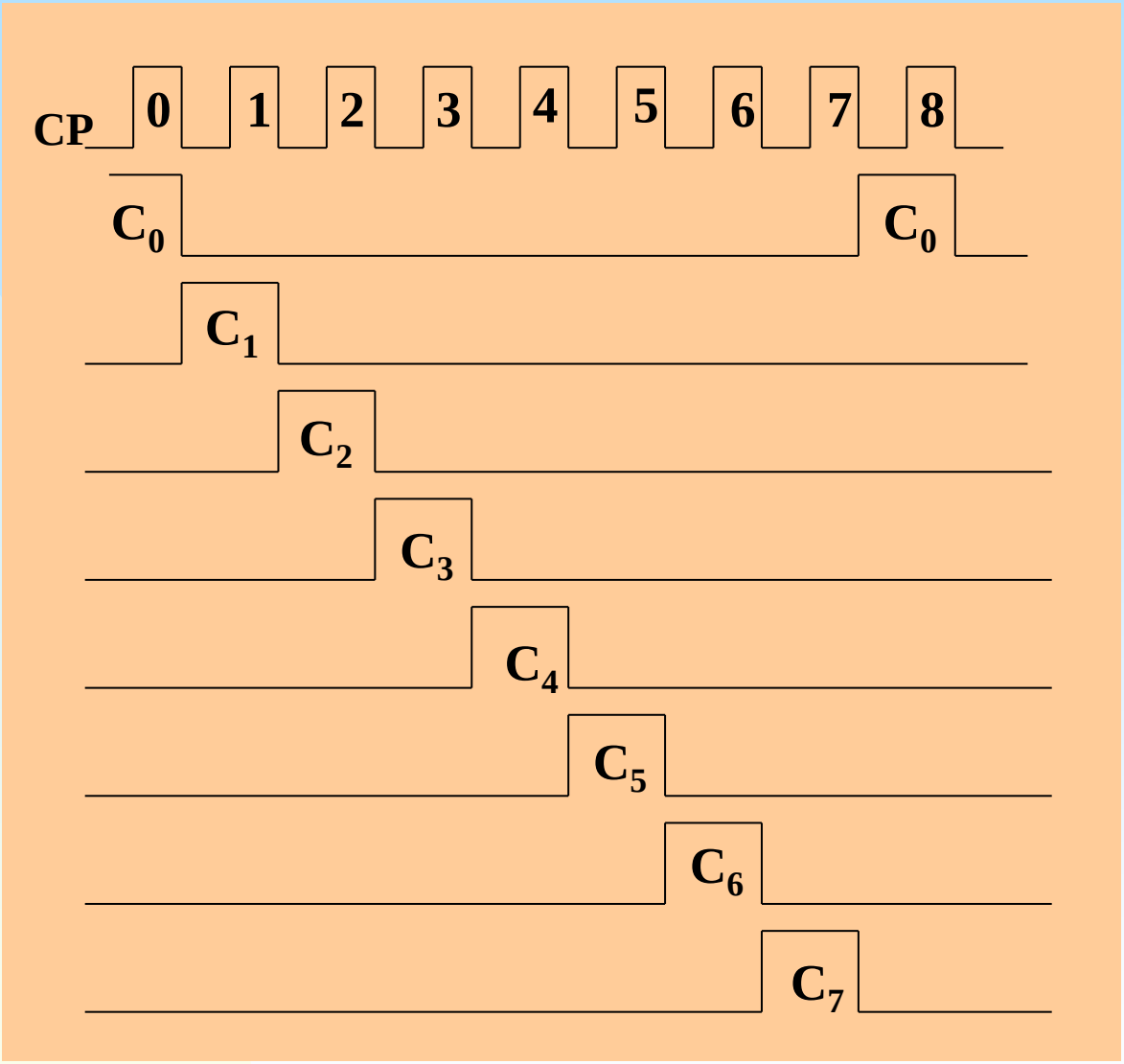



C0
C1
C2
C3
C4
C5
C6
C7
译码器
1
&
Q0
Q0
Q1
Q1
Q2
Q2
Q
Q
1D
C1
F2
F1
F0
计数器
时钟输入端

节拍信号发生器逻辑图

时钟信号
(主频)

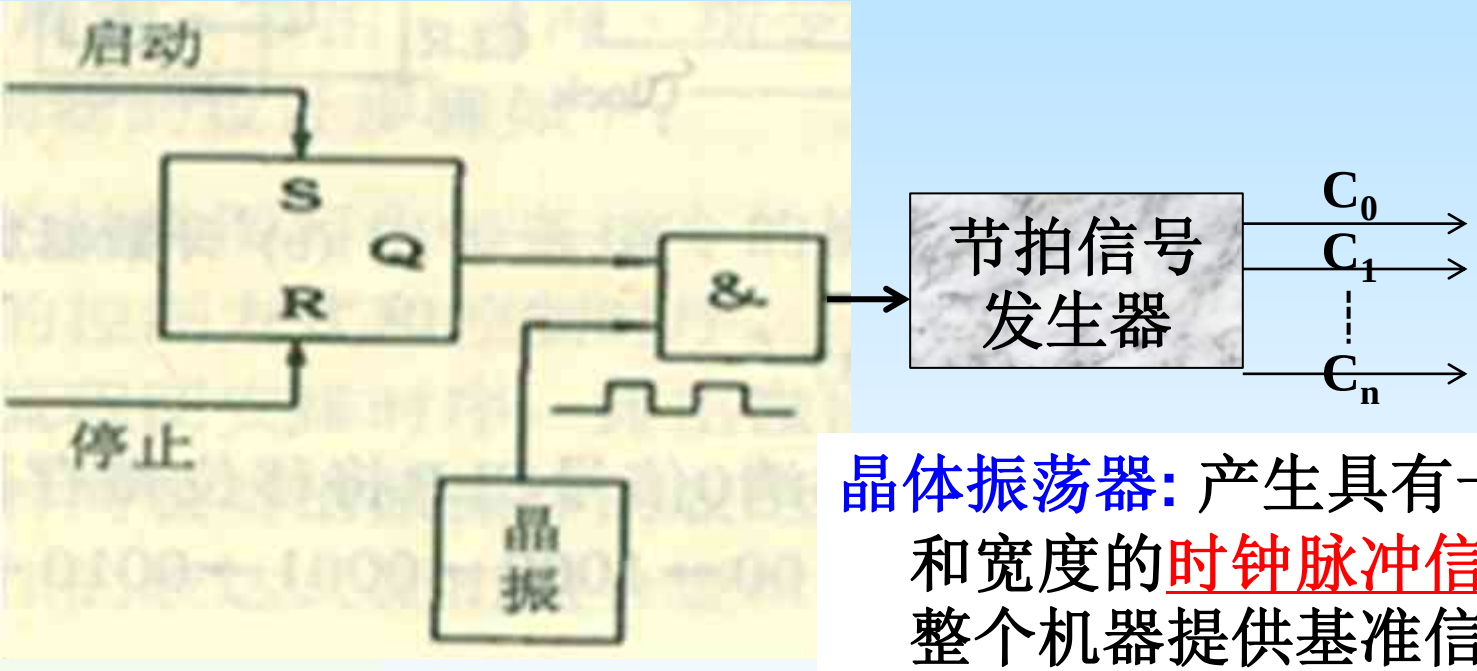
节拍信号



节拍信号发生器的波形图

时序部件

时序信号对控制信号定时：

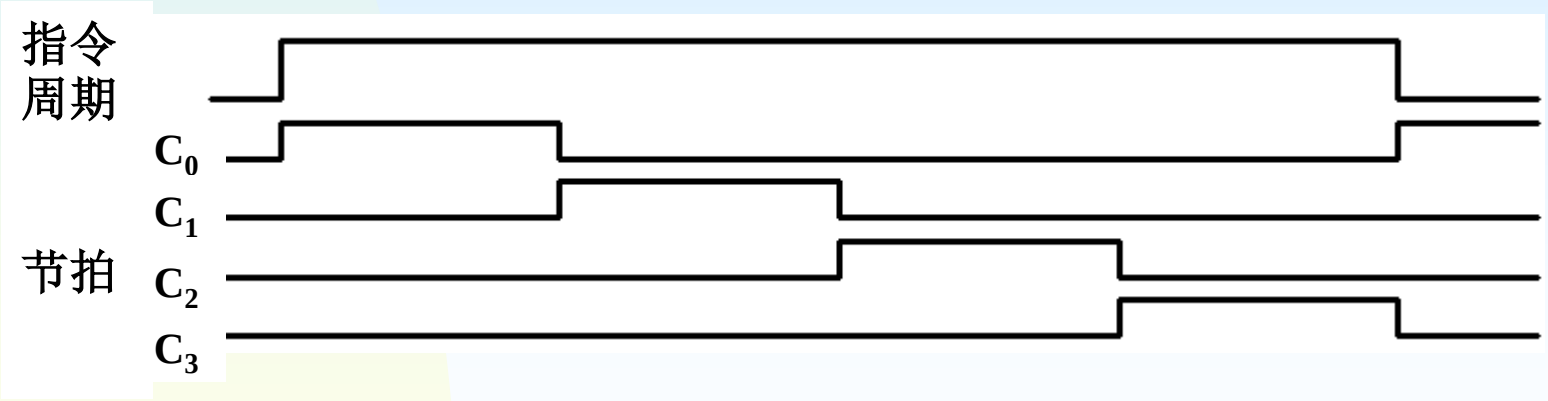


晶体振荡器：产生具有一定频率和宽度的时钟脉冲信号，为整个机器提供基准信号—主频。

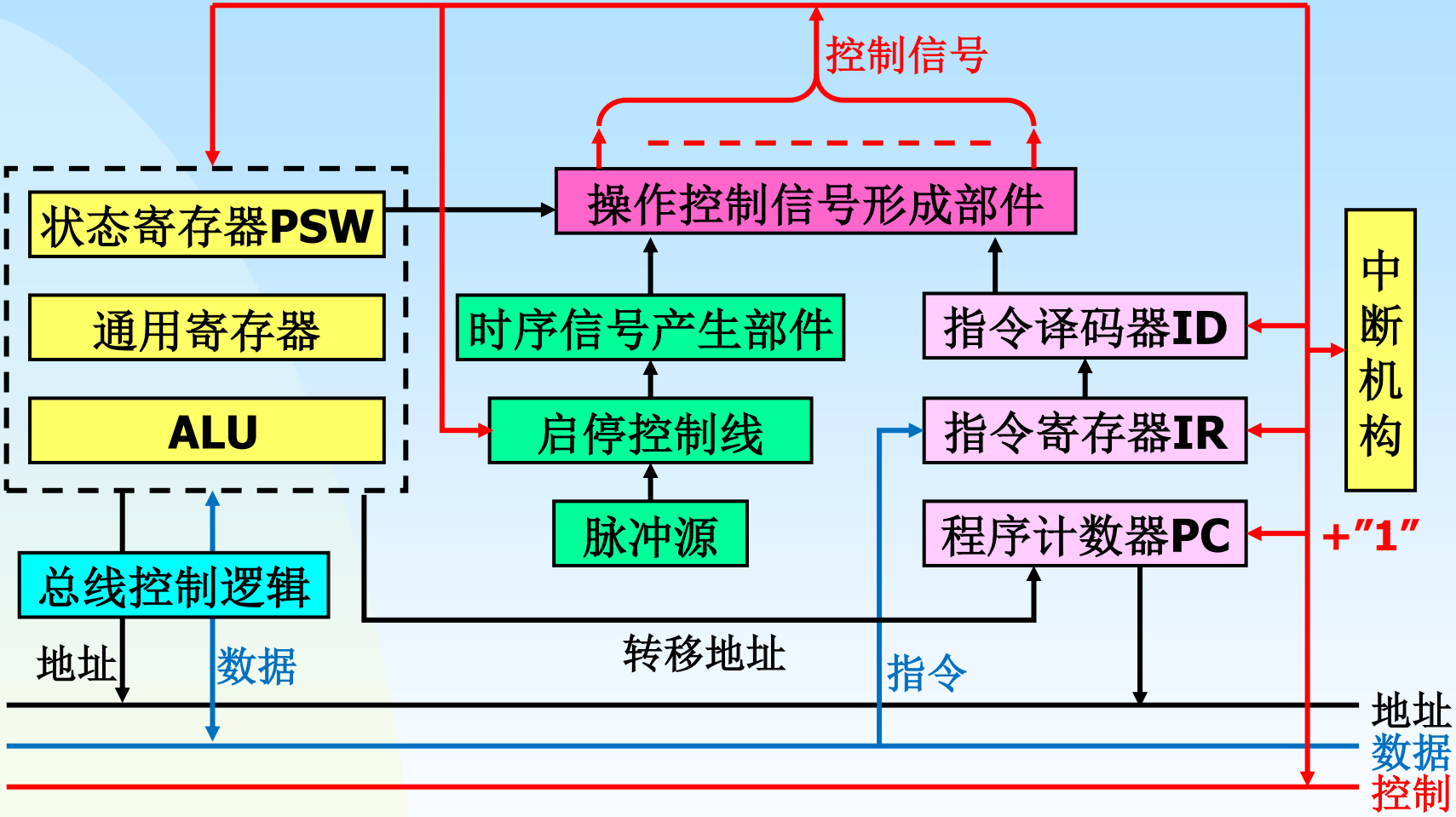
启停控制逻辑：开放或封锁脉冲 → 控制时序信号的发生或停止 →启动或停止整个机器。

有了时序信号，才能将计算机的工作安排在不同的时间段内有序完成。

每拍完成一步操作。



控制信号形成部件(控制单元CU)



工作原理：时间、指令代码、状态等信息作为逻辑变量，经微操作信号发生器产生微操作命令序列。

$$C = f(I, T, S)$$

控制信号形成部件(控制单元CU)

控制单元产生控制信号序列：如， $S_3 \sim S_0$，R1out, MARin， MemR, MemW等

（1）直接作用于逻辑电路；

（2）控制实现指令功能所要求的数据传送、运算处理等操作。

产生控制信号序列的方式不同。

控制单元（CU）的实现方法：

1. 硬连线逻辑方式——组合逻辑控制器
2. 存储逻辑方式——微程序控制器