

计算机组成原理 (第九讲)



厦门大学信息学院软件工程系 曾文华
2021年6月2日



第4篇 控制单元

第9章 控制单元的功能

第10章 控制单元的设计



第9章 控制单元的功能

9.1 微操作命令的分析

9.2 控制单元的功能



9.1 微操作命令的分析

- 一、取指周期
- 二、间指周期
- 三、执行周期
- 四、中断周期

8.2 指令周期

- 一、指令周期的基本概念
- 二、指令周期的数据流



完成一条指令分 4 个工作周期：

1、取指周期

2、间址周期

3、执行周期

4、中断周期

一、取指周期

取指周期的目的是：将指令的机器码从存储器中取到IR中

见P344（第8章）

① $PC \rightarrow MAR \rightarrow \text{地址线}$

② $1 \rightarrow R$

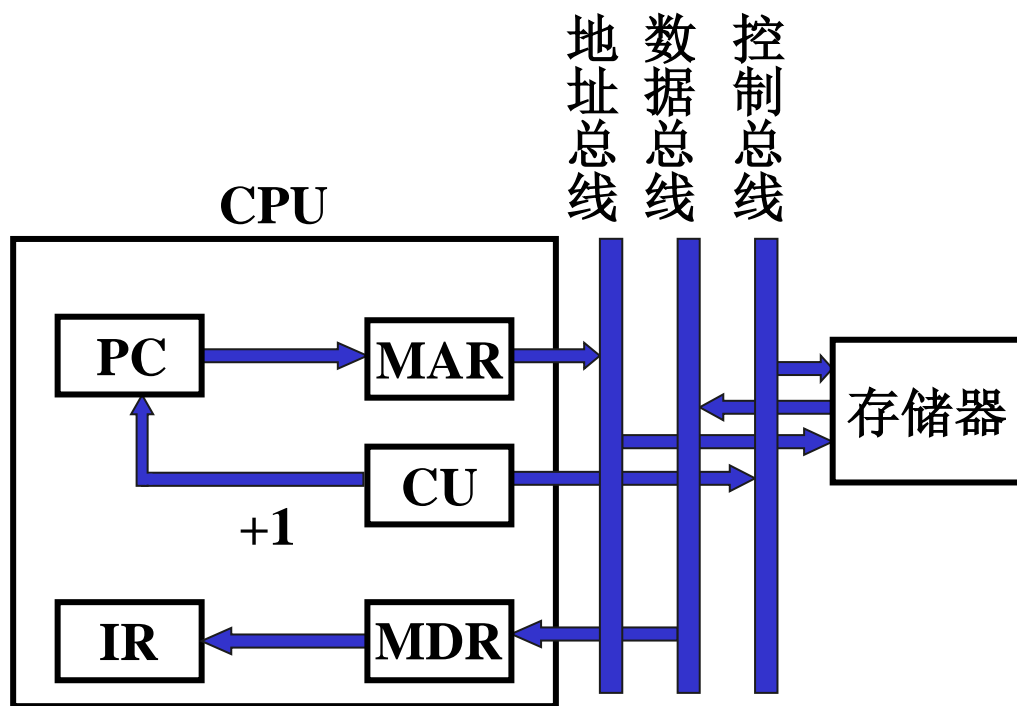
③ $M(MAR) \rightarrow MDR$

④ $MDR \rightarrow IR$

⑤ $OP(IR) \rightarrow CU$

⑥ $(PC) + 1 \rightarrow PC$

OP (IR)：指令的操作码部分



二、间址周期

间址周期的目的是：由形式地址得到有效地址

见P344（第8章）

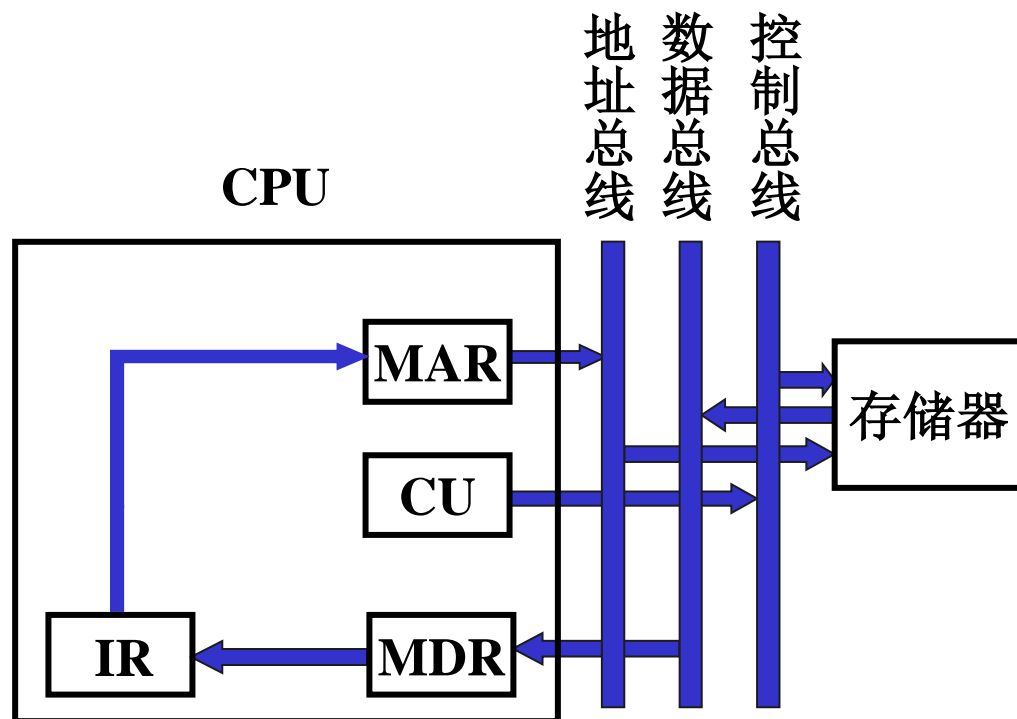
① 指令形式地址 \rightarrow MAR

即：Ad (IR) \rightarrow MAR

② 1 \rightarrow R

③ M (MAR) \rightarrow MDR

④ MDR \rightarrow Ad (IR)



Ad (IR) : 指令的地址码部分

三、执行周期

10条指令

1. 非访存指令

在执行周期不访问存储器

三、执行周期

10条指令

1. 非访存指令

在执行周期不访问存储器

(1) **CLA** 清A

$0 \rightarrow \text{ACC}$

三、执行周期

10条指令

1. 非访存指令

在执行周期不访问存储器

(1) **CLA** 清A

$0 \rightarrow \text{ACC}$

(2) **COM** 取反

$\overline{\text{ACC}} \rightarrow \text{ACC}$

三、执行周期

10条指令

1. 非访存指令

在执行周期不访问存储器

(1) **CLA** 清A

$0 \rightarrow \text{ACC}$

(2) **COM** 取反

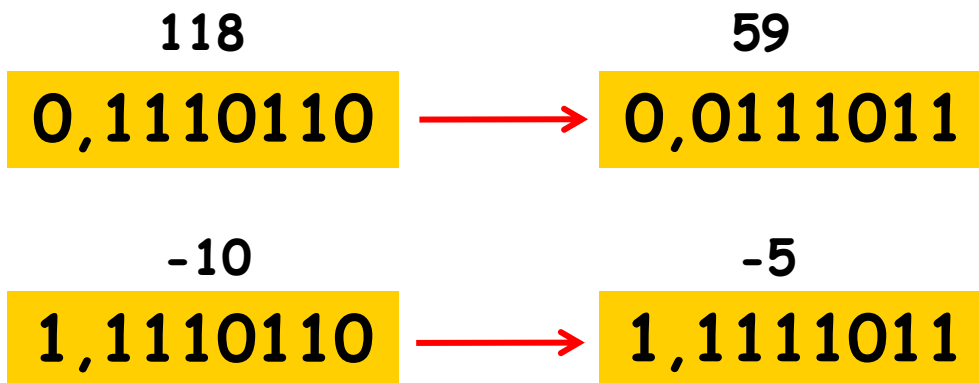
$\overline{\text{ACC}} \rightarrow \text{ACC}$

左边 \rightarrow 右边

(3) **SHR** 算术右移

$\text{L}(\text{ACC}) \rightarrow \text{R}(\text{ACC}), \text{ACC}_0 \rightarrow \text{ACC}_0$

符号位不变



三、执行周期

10条指令

1. 非访存指令 在执行周期不访问存储器

(1) **CLA** 清A

$0 \rightarrow \text{ACC}$

(2) **COM** 取反

$\overline{\text{ACC}} \rightarrow \text{ACC}$

左边 \rightarrow 右边

(3) **SHR** 算术右移

$\text{L}(\text{ACC}) \rightarrow \text{R}(\text{ACC}), \text{ACC}_0 \rightarrow \text{ACC}_0$

符号位不变

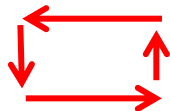
(4) **CSL** 循环左移

$\text{R}(\text{ACC}) \rightarrow \text{L}(\text{ACC}), \text{ACC}_0 \rightarrow \text{ACC}_n$

11110110



11101101



右边 \rightarrow 左边

三、执行周期

10条指令

1. 非访存指令 在执行周期不访问存储器

(1) **CLA** 清A

$0 \rightarrow \text{ACC}$

(2) **COM** 取反

$\overline{\text{ACC}} \rightarrow \text{ACC}$

左边 \rightarrow 右边

(3) **SHR** 算术右移

$\text{L}(\text{ACC}) \rightarrow \text{R}(\text{ACC}), \text{ACC}_0 \rightarrow \text{ACC}_0$

符号位不变

(4) **CSL** 循环左移

$\text{R}(\text{ACC}) \rightarrow \text{L}(\text{ACC}), \text{ACC}_0 \rightarrow \text{ACC}_n$

(5) **STP** 停机指令

$0 \rightarrow \text{G}$

右边 \rightarrow 左边

运行标志触发器

2. 访存指令 在执行周期要访问存储器

2. 访存指令 在执行周期要访问存储器

(1) 加法指令

ADD X

ADD A, (X)

$Ad(IR) \rightarrow MAR$

ADD AL, [2300H]

2300H

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$(ACC) + (MDR) \rightarrow ACC$

[2300H]

2. 访存指令 在执行周期要访问存储器

(1) 加法指令

ADD X

ADD A, (X)

$\text{Ad(IR)} \rightarrow \text{MAR}$

ADD AL, [2300H]

$1 \rightarrow R$

$\text{M(MAR)} \rightarrow \text{MDR}$

$(\text{ACC}) + (\text{MDR}) \rightarrow \text{ACC}$

(2) 存数指令

STA X

MOV (X), A

$\text{Ad(IR)} \rightarrow \text{MAR}$

MOV [2300H], AL

2300H

$1 \rightarrow W$

$\text{ACC} \rightarrow \text{MDR}$

[2300H]

$\text{MDR} \rightarrow \text{M(MAR)}$

(3) 取数指令 **LDA X** **MOV A, (X)**



2300H → **Ad (IR) → MAR**

MOV AL,[2300H]

1 → R

M (MAR) → MDR

[2300H] → **MDR → ACC**

(3) 取数指令

LDA X

MOV A, (X)

$Ad(IR) \rightarrow MAR$

MOV AL, [2300H]

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$MDR \rightarrow ACC$

3. 转移指令

(3) 取数指令 **LDA X** **MOV A, (X)**

$Ad(IR) \rightarrow MAR$

MOV AL, [2300H]

$1 \rightarrow R$

$M(MAR) \rightarrow MDR$

$MDR \rightarrow ACC$

3. 转移指令

(1) 无条件转 **JMP X**

$Ad(IR) \rightarrow PC$

X

(3) 取数指令

LDA X **MOV A, (X)**

$\text{Ad}(\text{IR}) \rightarrow \text{MAR}$

MOV AL, [2300H]

$1 \rightarrow \text{R}$

$\text{M}(\text{MAR}) \rightarrow \text{MDR}$

$\text{MDR} \rightarrow \text{ACC}$

3. 转移指令

(1) 无条件转

JMP X

$\text{Ad}(\text{IR}) \rightarrow \text{PC}$

条件成立: $A_0=1$

$A_0 \cdot \text{Ad}(\text{IR}) + \neg A_0(\text{PC}) = \text{Ad}(\text{IR})$

条件不成立: $A_0=0$

$A_0 \cdot \text{Ad}(\text{IR}) + \neg A_0(\text{PC}) = \text{PC}$

(2) 条件转移

BAN X (负则转, $A_0=1$)

$A_0 \cdot \text{Ad}(\text{IR}) + \bar{A}_0(\text{PC}) \rightarrow \text{PC}$

4. 三类指令的指令周期

非访存 指令周期

取指周期

执行周期

直接访存 指令周期

取指周期

执行周期

间接访存 指令周期

取指周期

间址周期

执行周期

转移 指令周期

取指周期

执行周期

间接转移 指令周期

取指周期

间址周期

执行周期

四、中断周期

中断周期，由中断隐指令自动完成：（1）保护程序断点；（2）寻找中断服务程序的入口地址；（3）硬件关中断

程序断点存入“0”地址

保护断点

- $0 \rightarrow \text{MAR}$
- $1 \rightarrow \text{W}$
- $\text{PC} \rightarrow \text{MDR}$
- $\text{MDR} \rightarrow \text{M}(\text{MAR})$

中断服务程序入口地址 $\text{M} \rightarrow \text{PC}$

$0 \rightarrow \text{EINT}$ (置“0”)

程序断点进栈

保护断点

- $(\text{SP}) - 1 \rightarrow \text{MAR}$
- $1 \rightarrow \text{W}$
- $\text{PC} \rightarrow \text{MDR}$
- $\text{MDR} \rightarrow \text{M}(\text{MAR})$

中断服务程序入口地址 $\text{M} \rightarrow \text{PC}$

$0 \rightarrow \text{EINT}$ (置“0”)

见P363-364 (第八章)

主程序执行过程中有一个中断

000H

101H

断点存于 特定地址（0 号地址） 内

程序断点

100H

MOV AX,BX

101H

INC AX

102H

DEC BX

主程序

中断服务程序入口地址

200H

PUSH AX

PUSH BX

POP BX

POP AX

EI

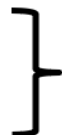
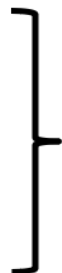
IRET

中断服务程序

断点进栈

101H

堆栈

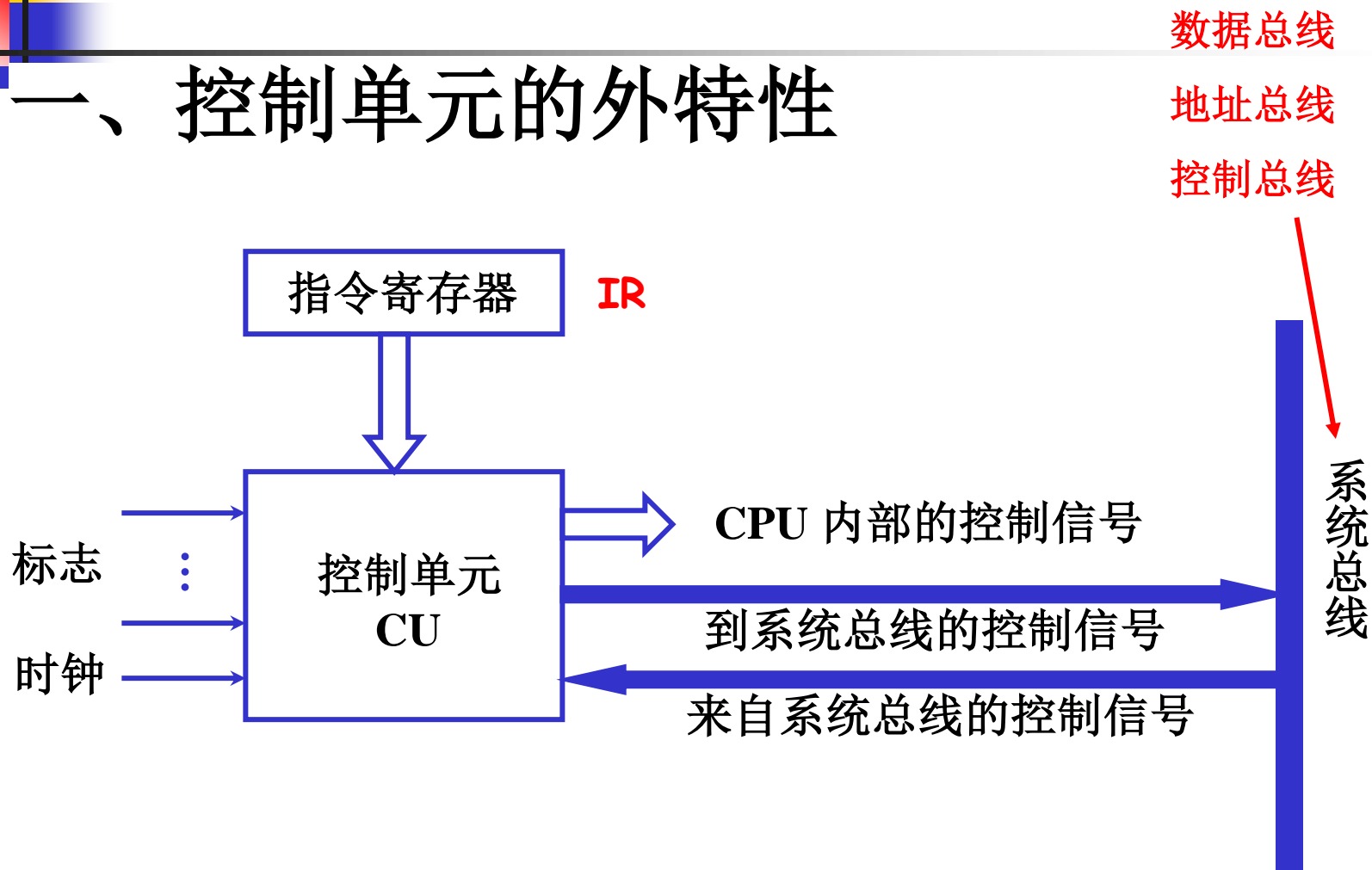




9.2 控制单元的功能

- 一、控制单元的外特性
- 二、控制信号举例
- 三、多级时序系统
- 四、控制方式
- 五、多级时序系统实例分析

一、控制单元的外特性



1. 控制单元的输入信号



(1) 时钟

CU 受时钟控制

一个时钟脉冲

发一个操作命令或一组需同时执行的操作命令

(2) 指令寄存器 $OP(IR) \rightarrow CU$

控制信号 与操作码有关

指令的操作码

(3) 标志

CU 受标志控制

状态寄存器

(4) 外来信号 来自系统总线的控制信号

FC、FZ

如 **INTR** 中断请求

HRQ 总线请求(DMA请求)

2. 控制单元的输出信号

(1) CPU 内的各种控制信号

$R_i \rightarrow R_j$

寄存器之
间的传递

$(PC) + 1 \rightarrow PC$

ALU +、-、与、或

(2) 送至控制总线的信号

CPU 外的控制信号

$\overline{\text{MREQ}}$

访存控制信号

$\overline{\text{IO/M}}$

访 IO/ 存储器的控制信号

$\overline{\text{RD}}$

读命令

$\overline{\text{WR}}$

写命令

INTA

中断响应信号

HLDA

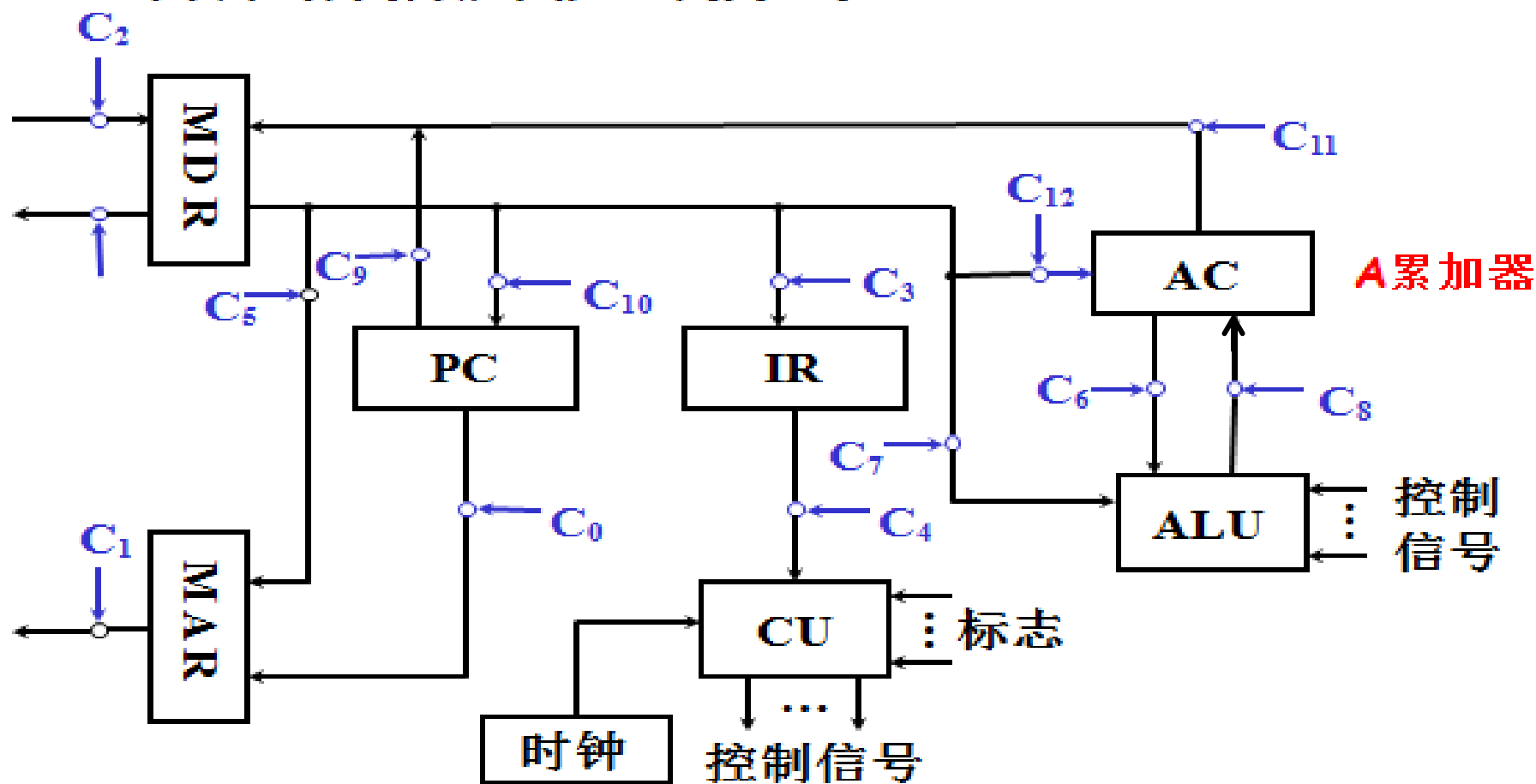
总线响应信号 (DMA响应信号)

二、控制信号举例

1. 不采用 CPU 内部总线的方式

图9.3

CPU 内部结构采用非总线方式



二、控制信号举例

1. 不采用 CPU 内部总线的方式

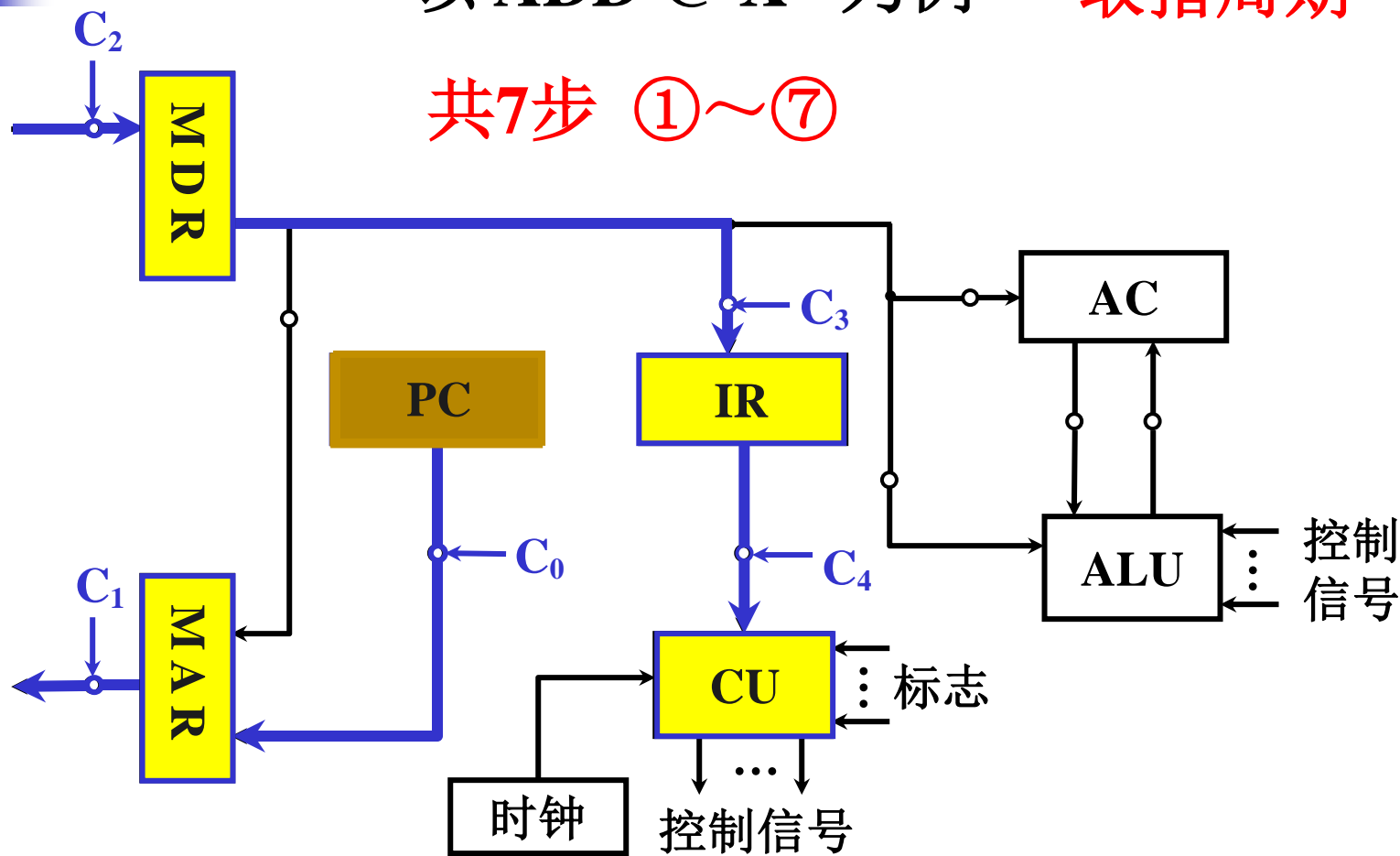
以 ADD @ X 为例

ADD A, ((X))

ADD AL, [[2300H]]

取指周期

共7步 ①~⑦



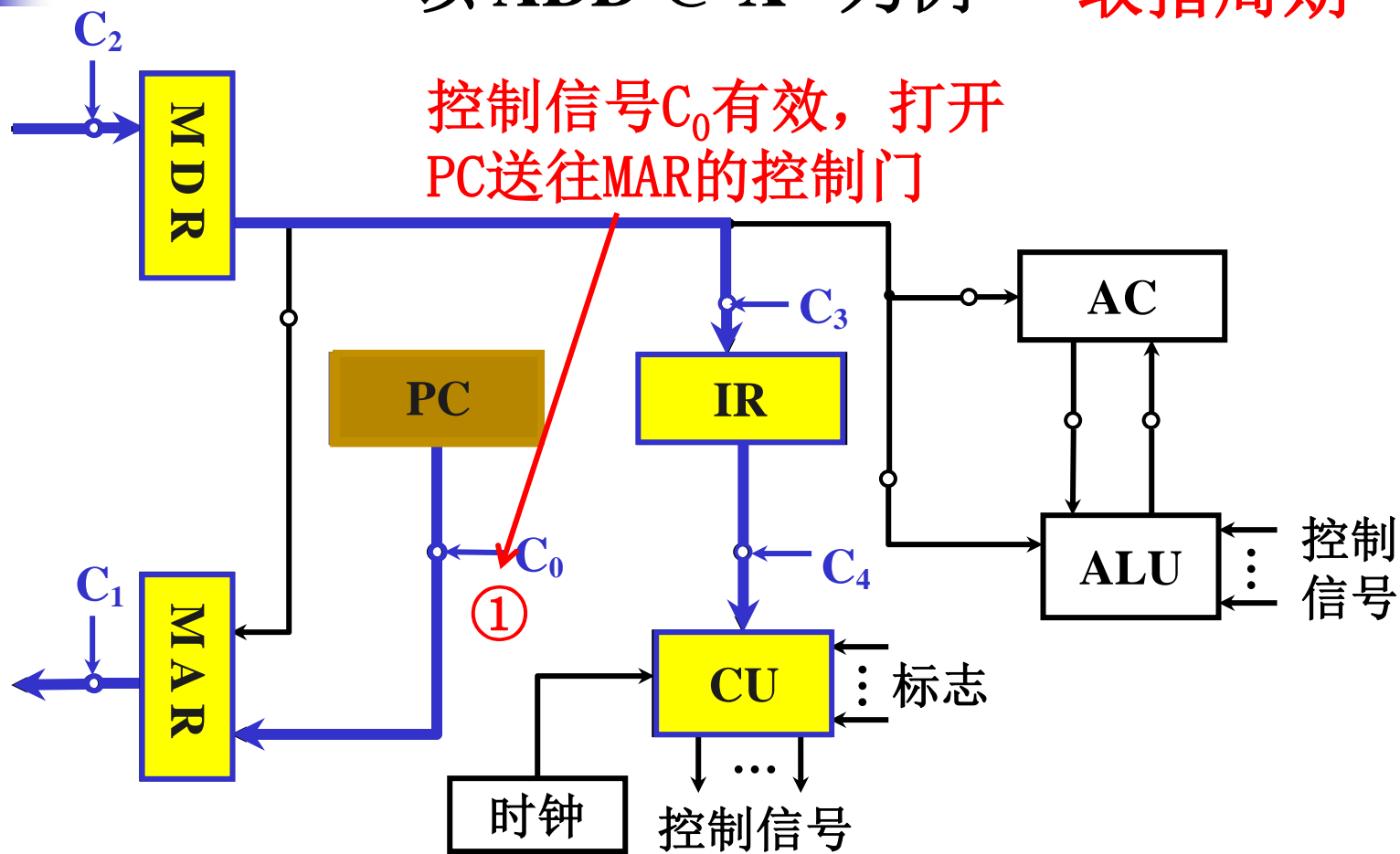
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))



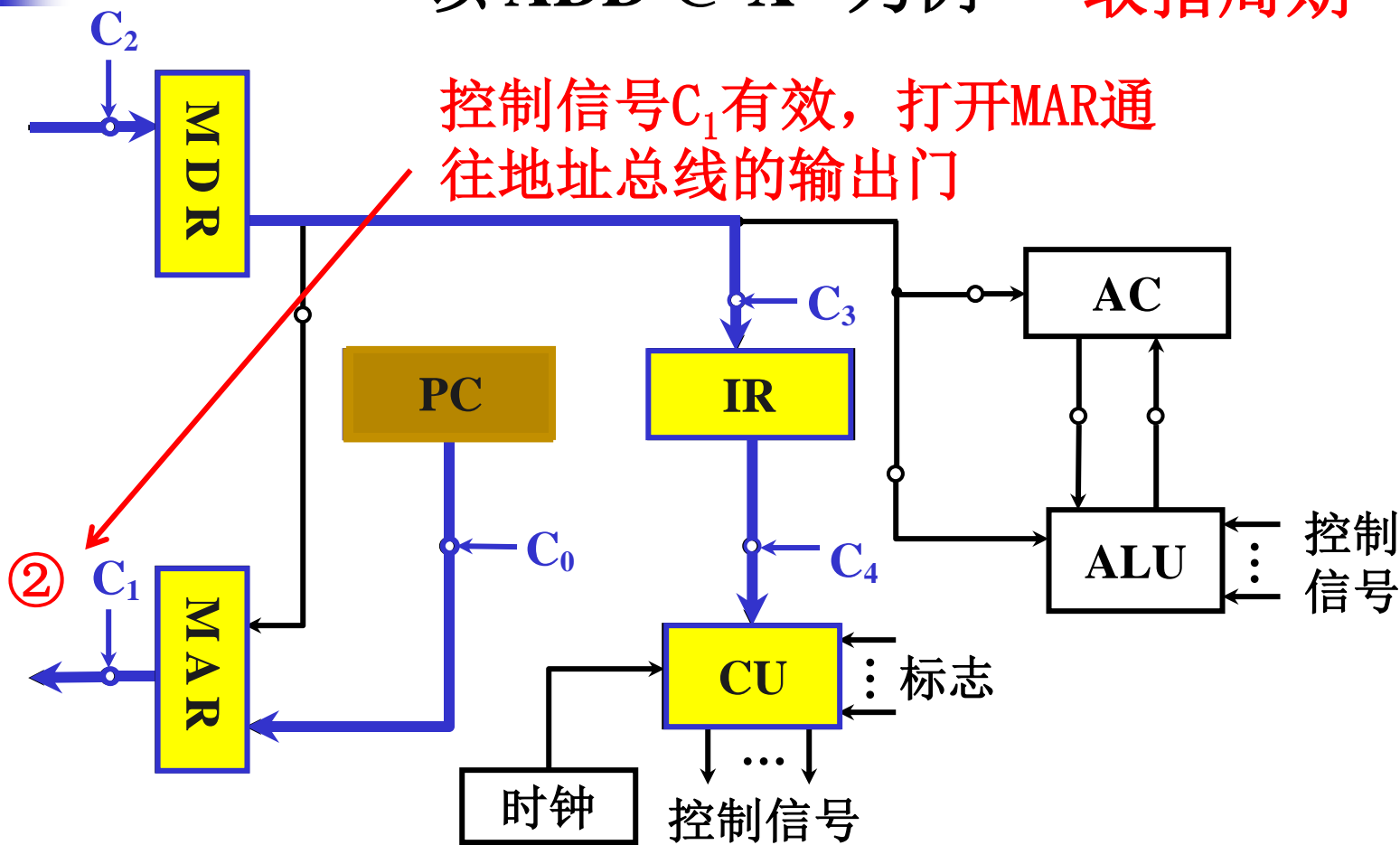
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))



二、控制信号举例

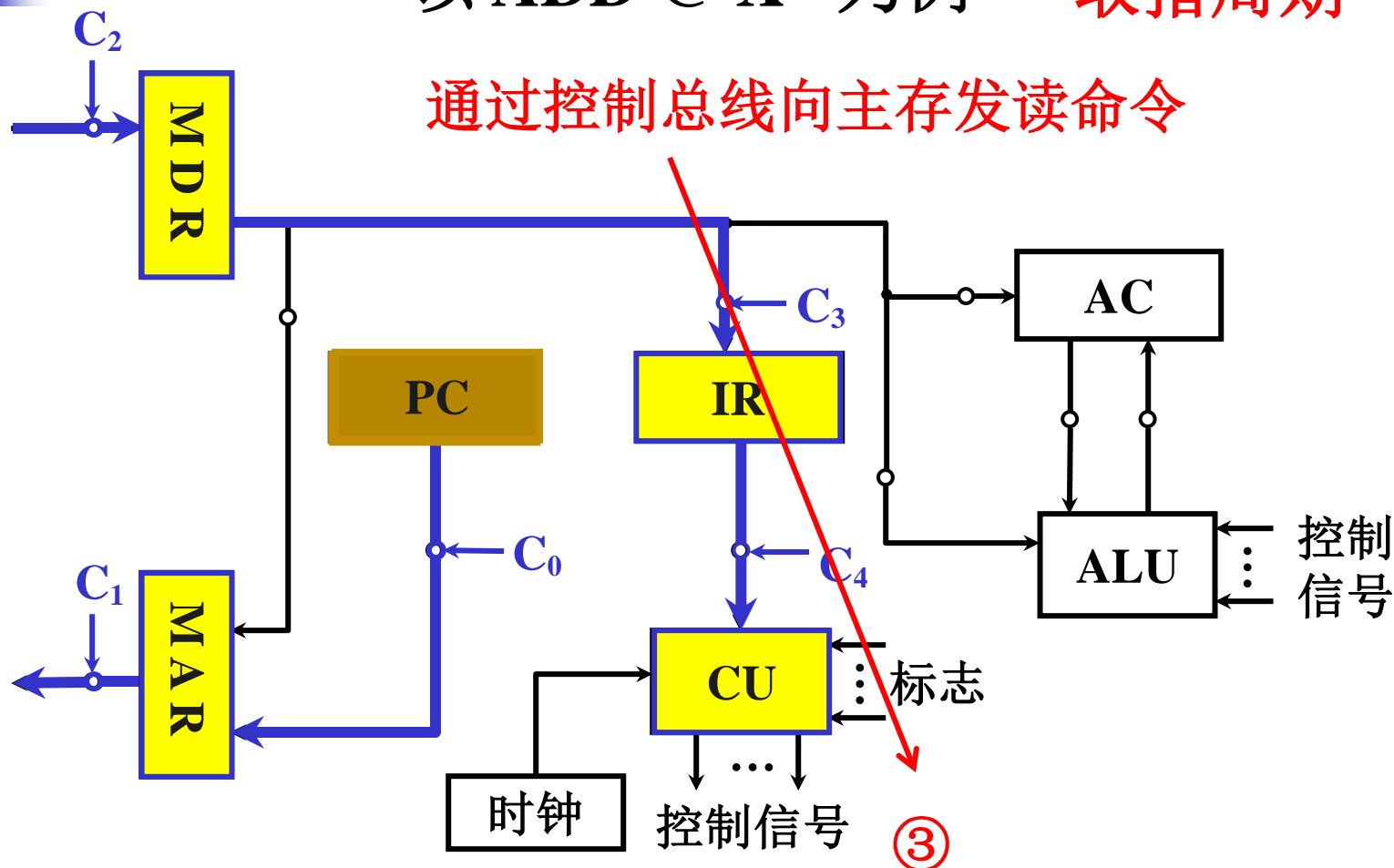
1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))

通过控制总线向主存发读命令



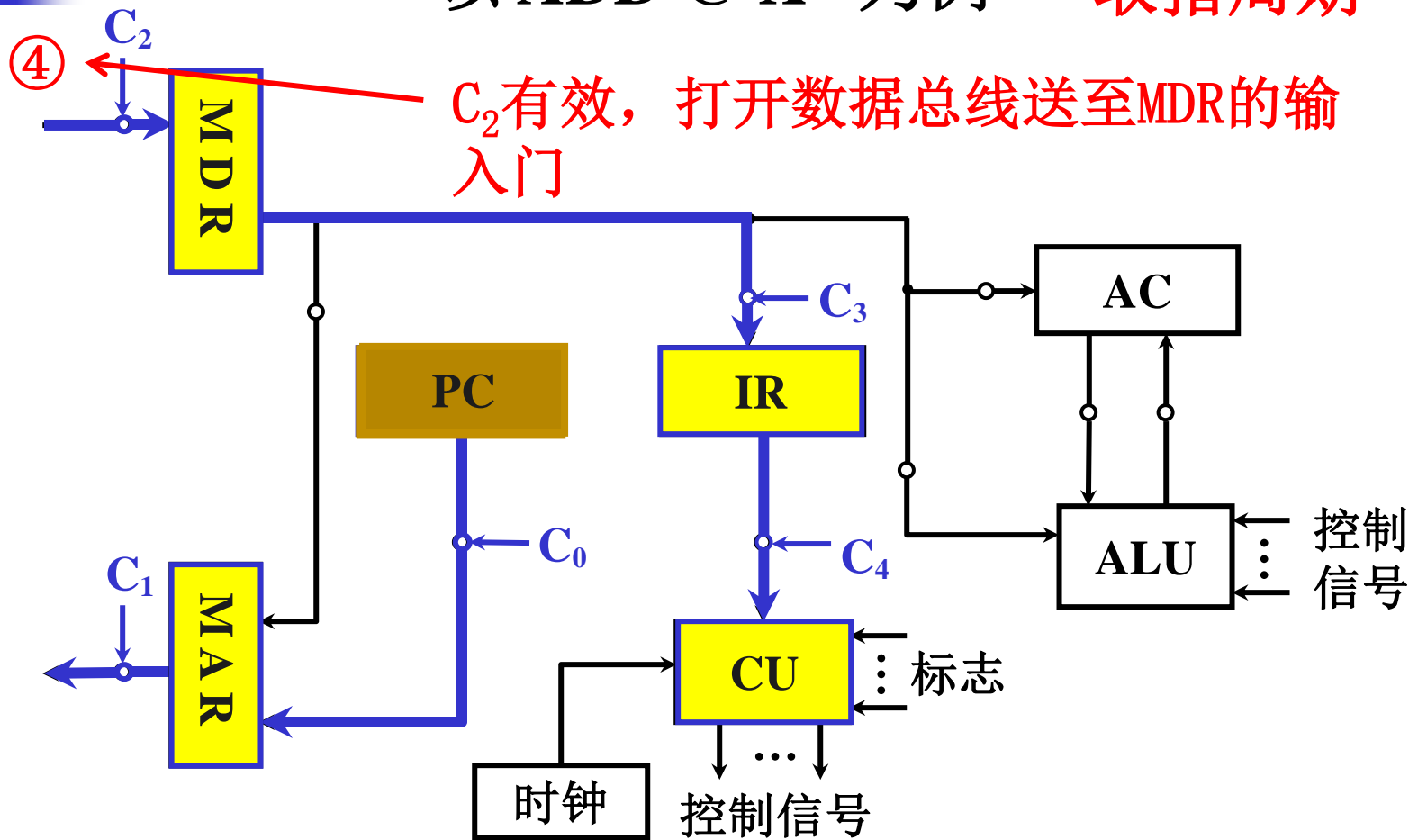
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))



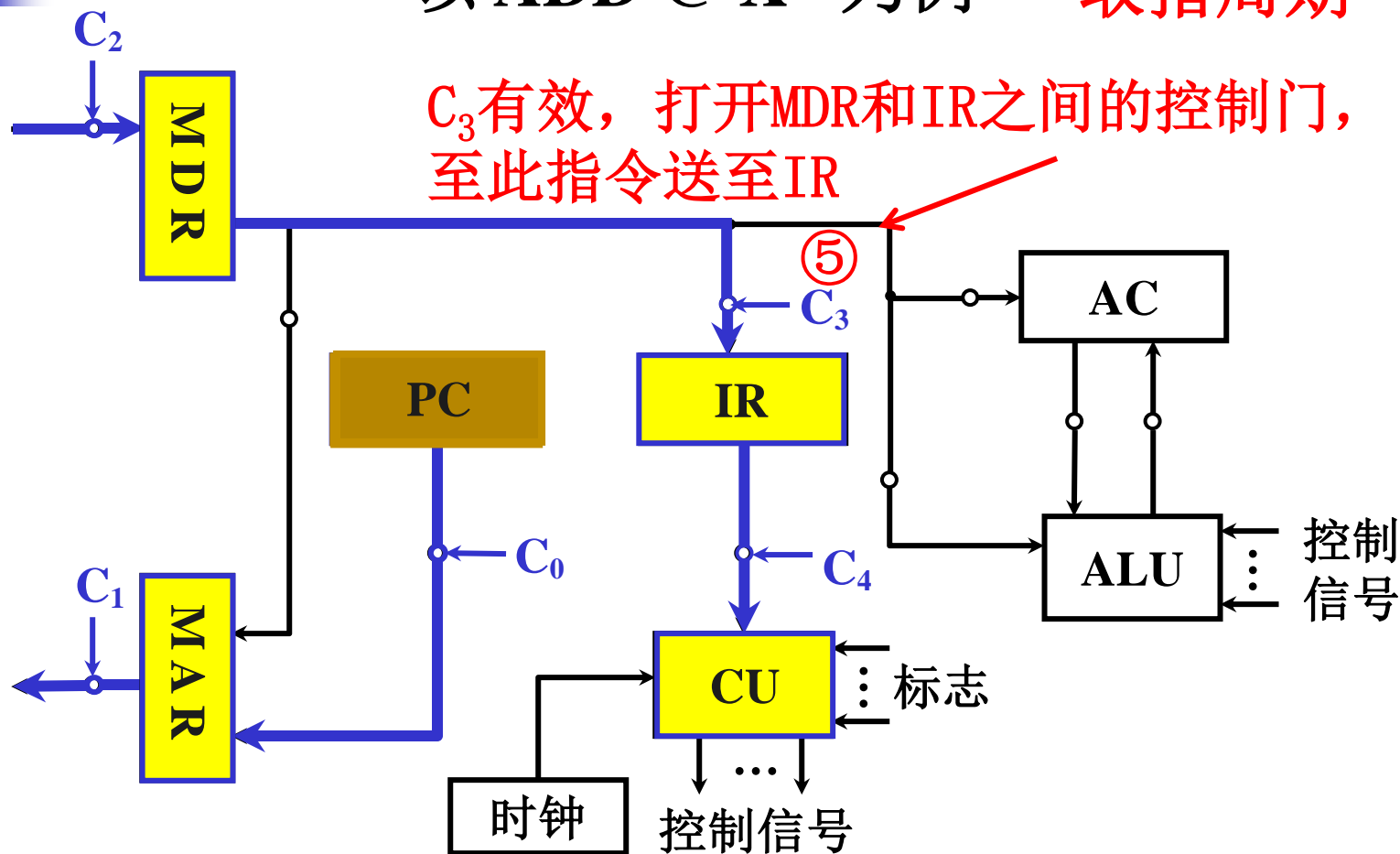
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))



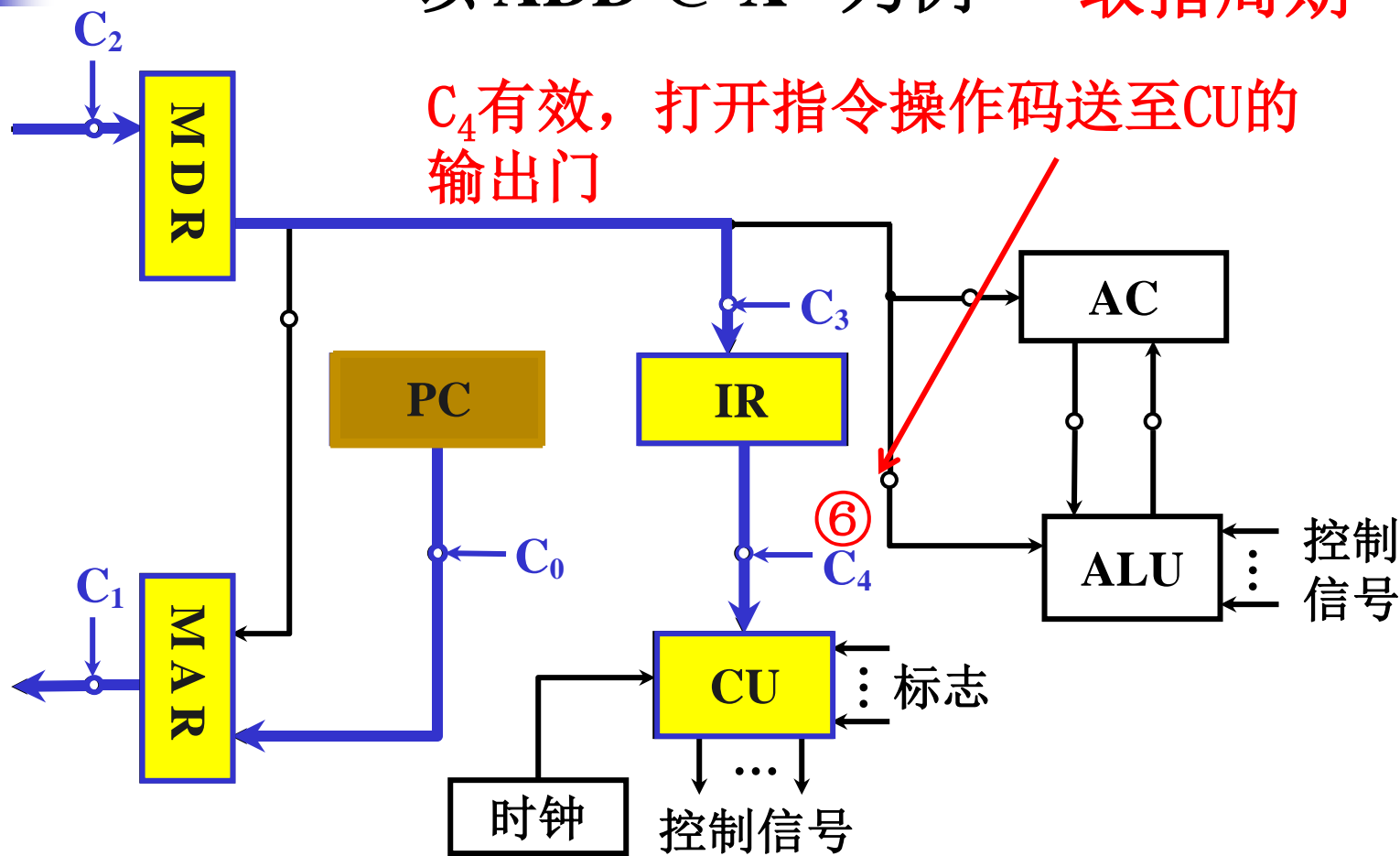
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

取指周期

ADD A, ((X))



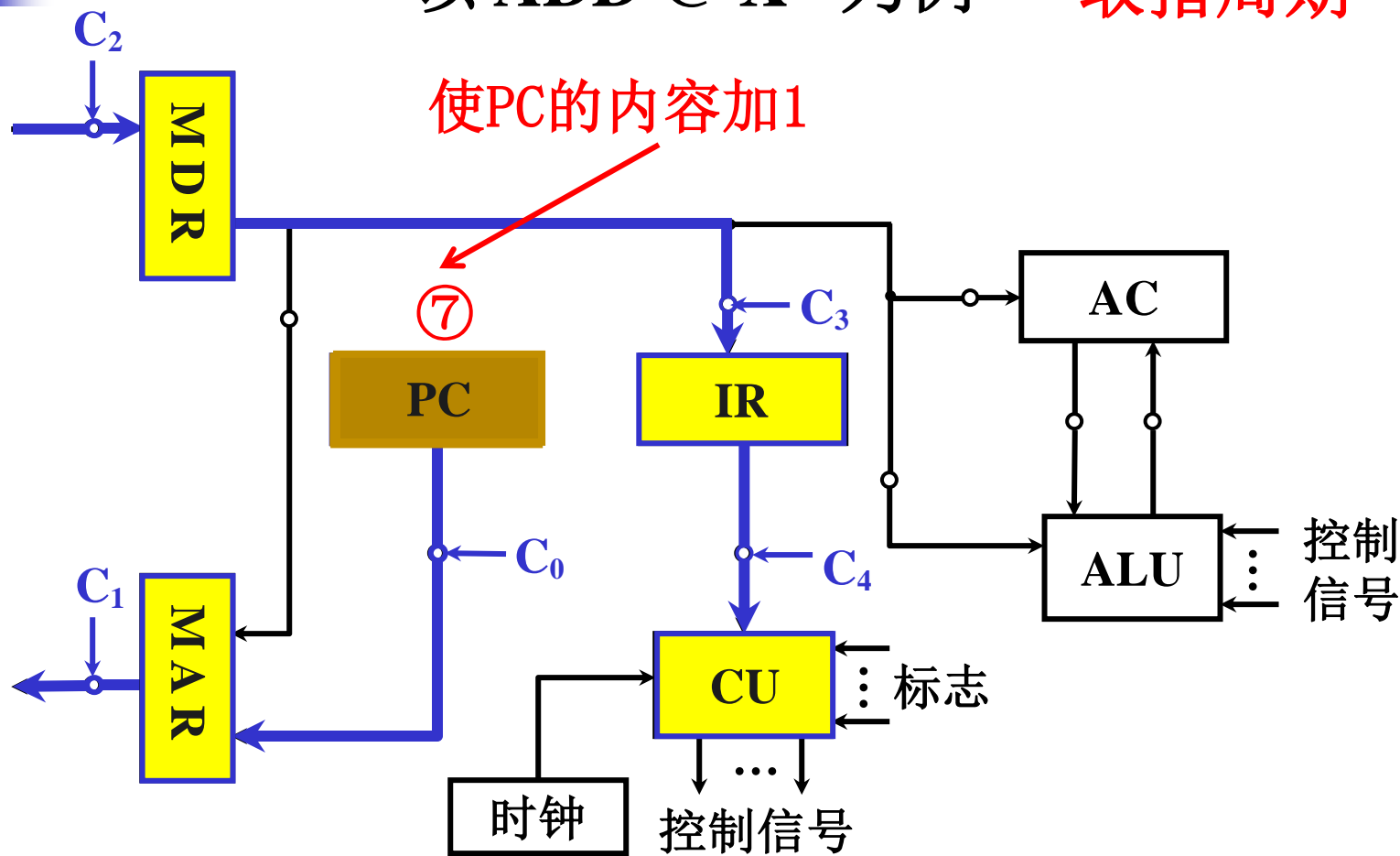
二、控制信号举例

1. 不采用 CPU 内部总线的方式

以 ADD @ X 为例

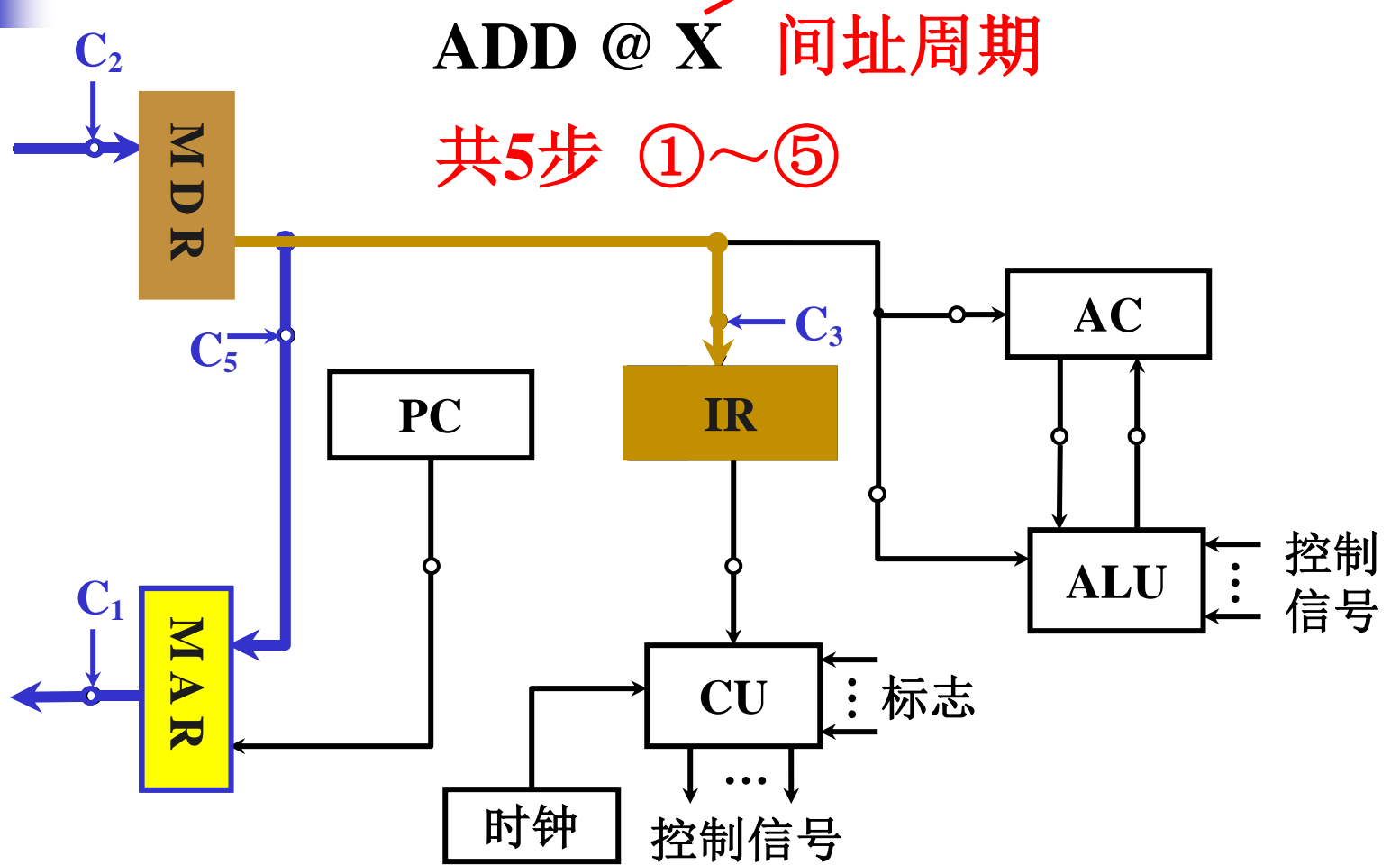
取指周期

ADD A, ((X))



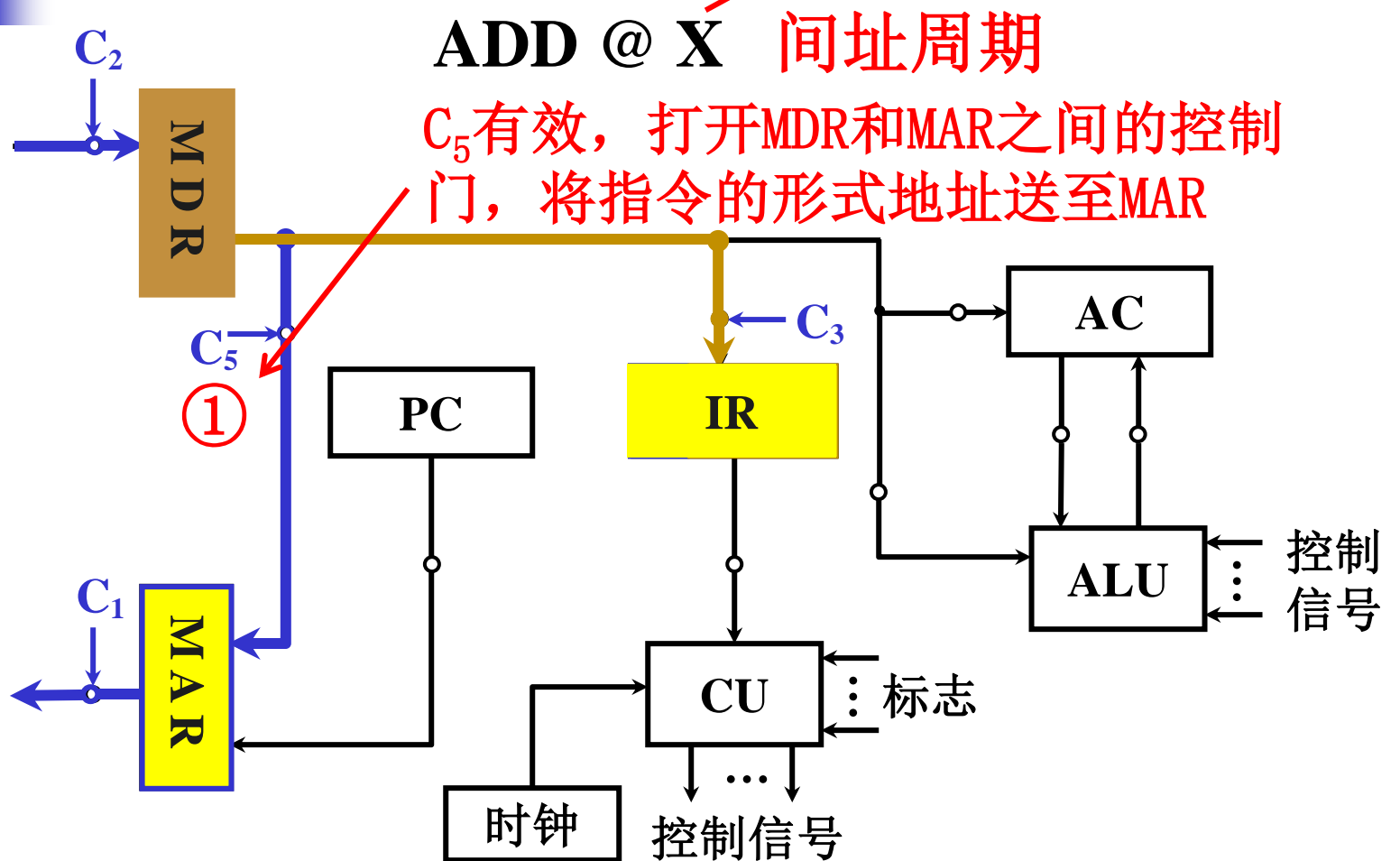
二、控制信号举例

1. 不采用 CPU 内部总线的方式



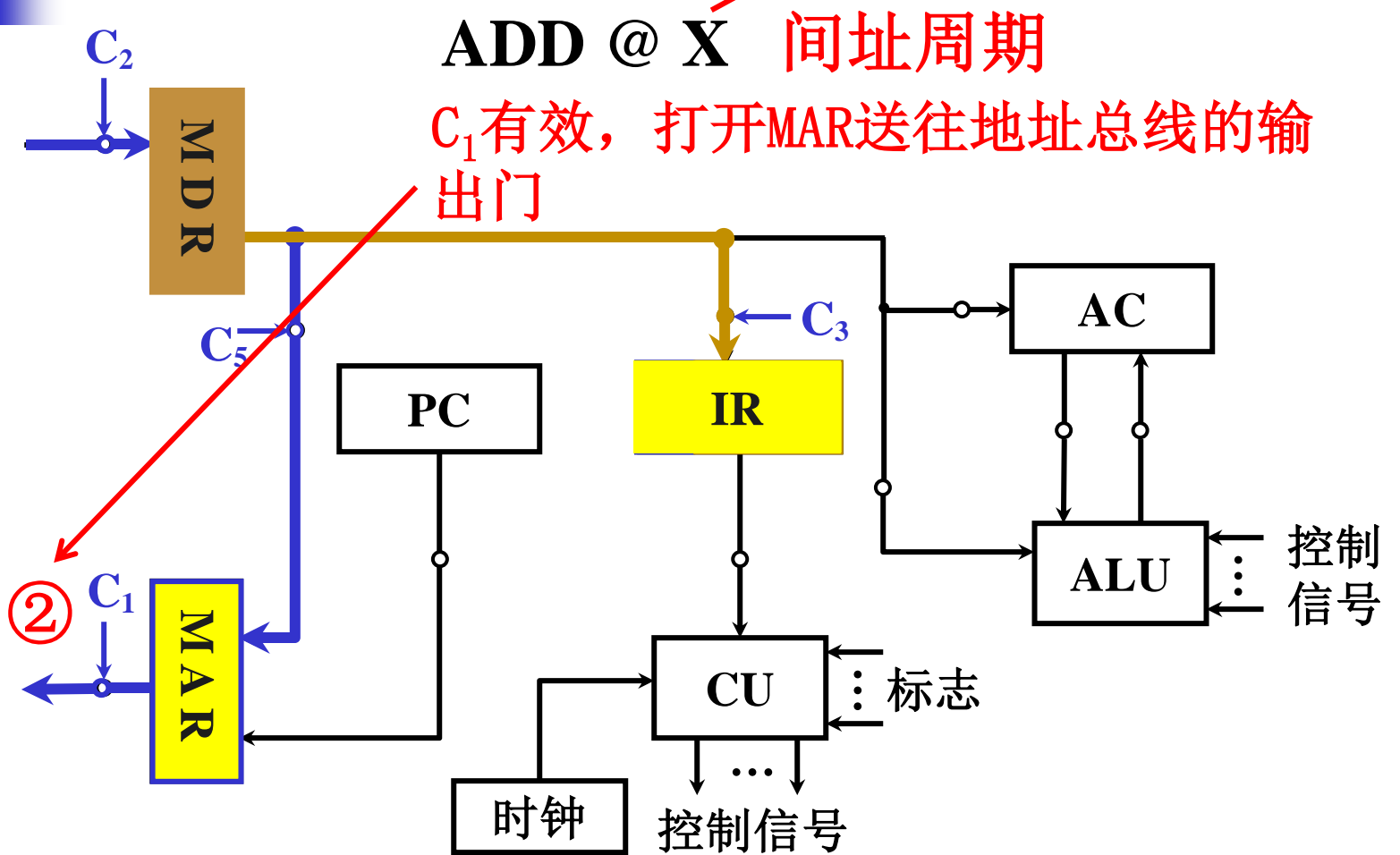
二、控制信号举例

1. 不采用 CPU 内部总线的方式



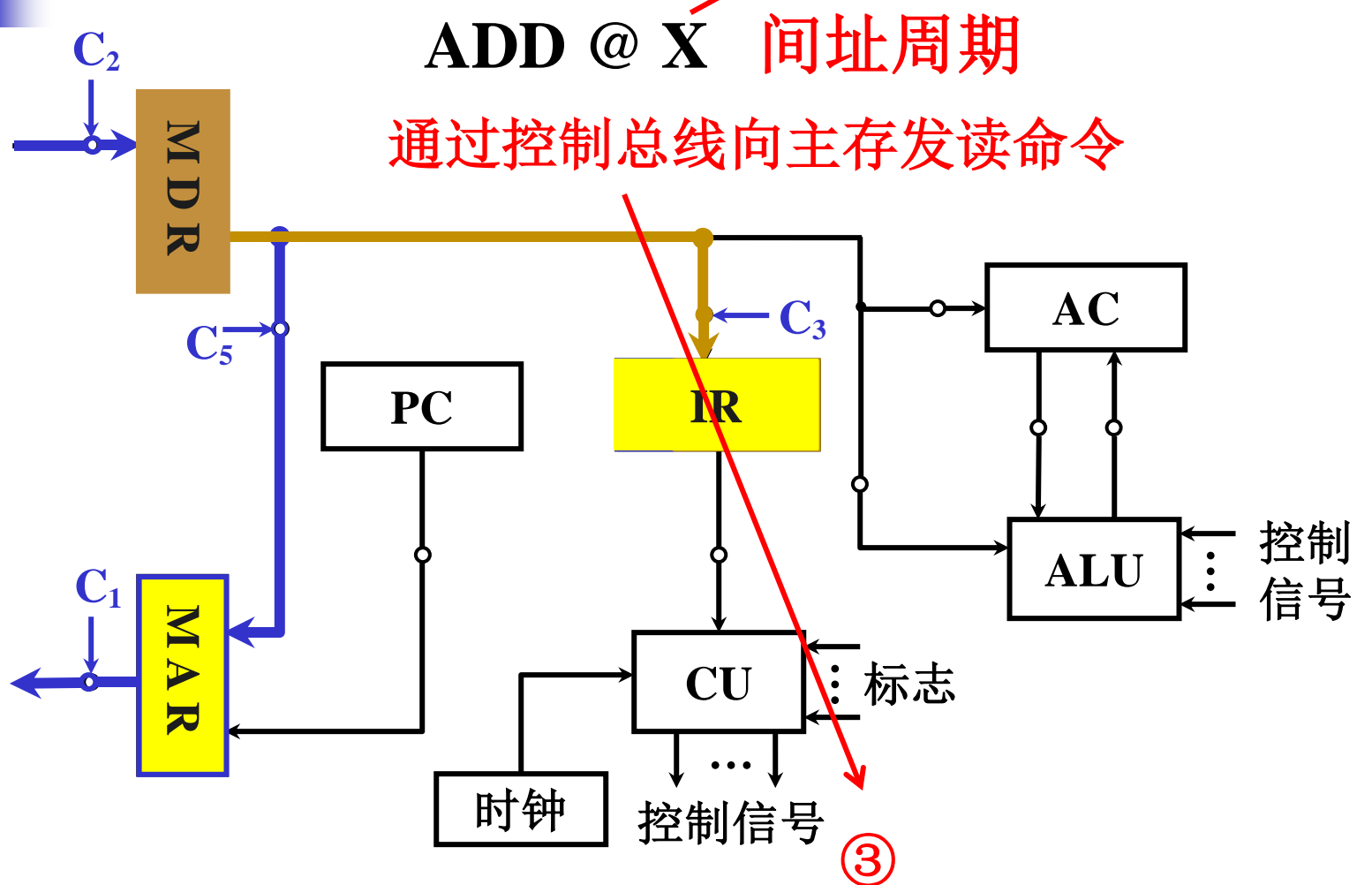
二、控制信号举例

1. 不采用 CPU 内部总线的方式



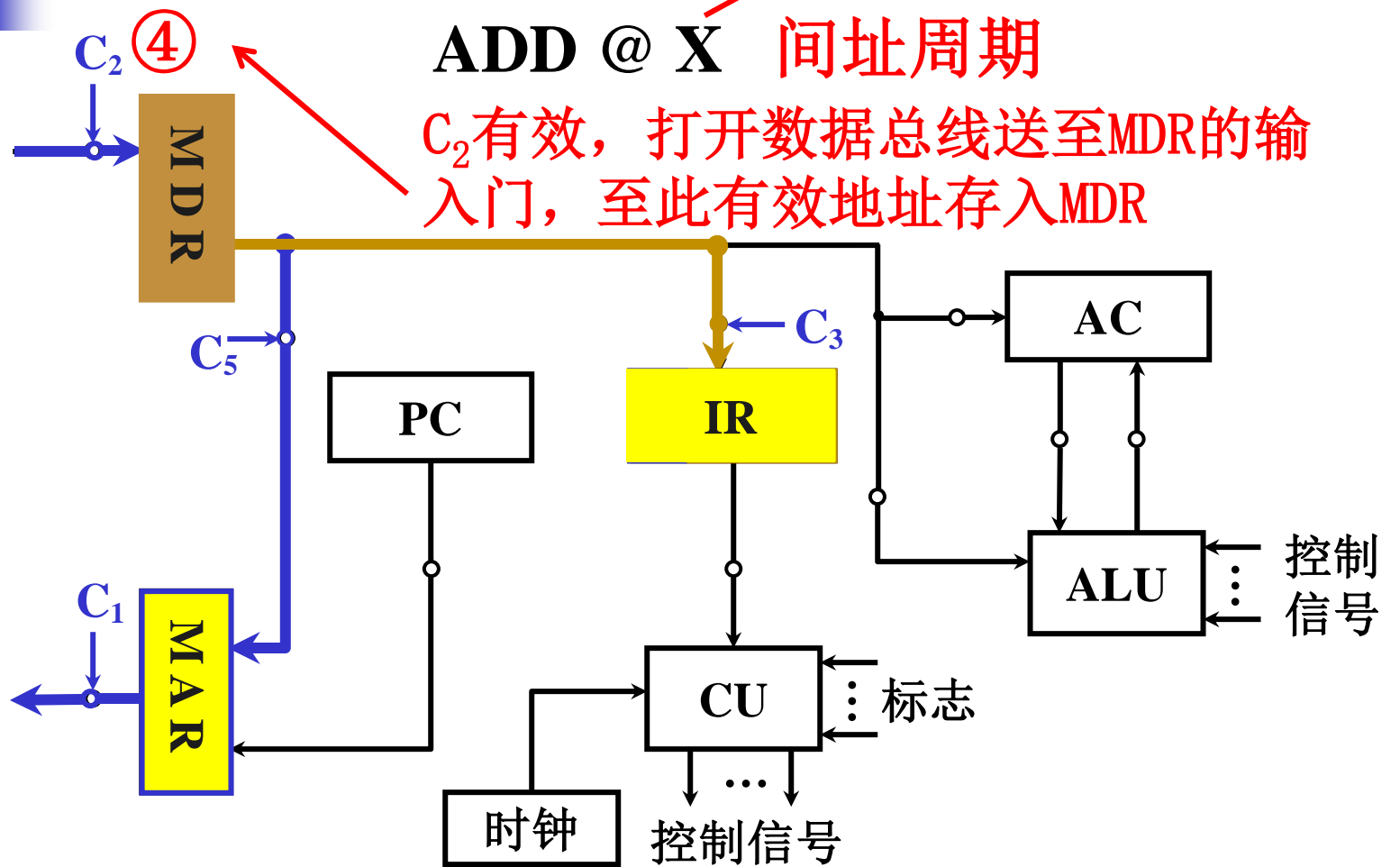
二、控制信号举例

1. 不采用 CPU 内部总线的方式



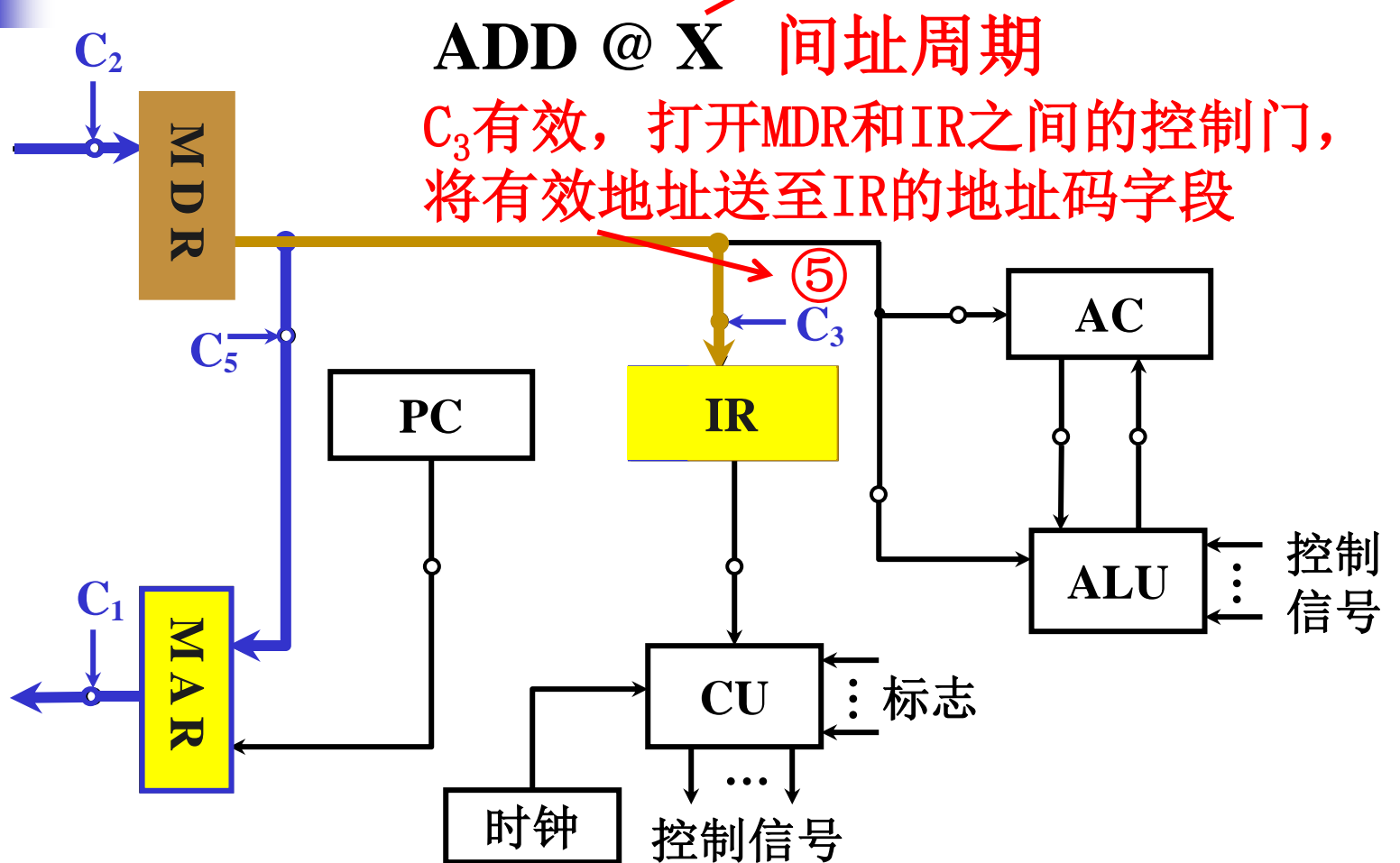
二、控制信号举例

1. 不采用 CPU 内部总线的方式



二、控制信号举例

1. 不采用 CPU 内部总线的方式



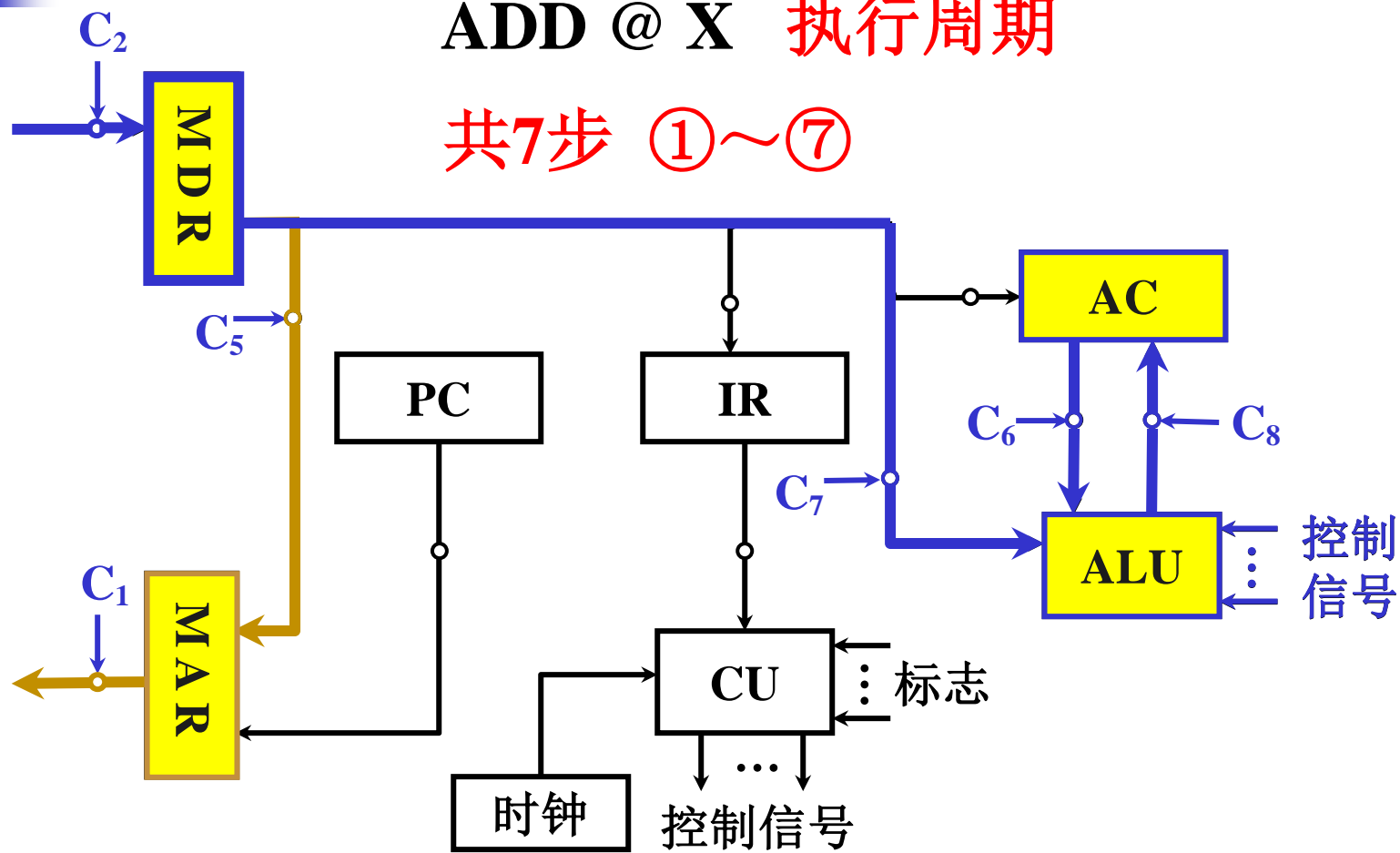
二、控制信号举例

1. 不采用 CPU 内部总线的方式

ADD A, ((X))

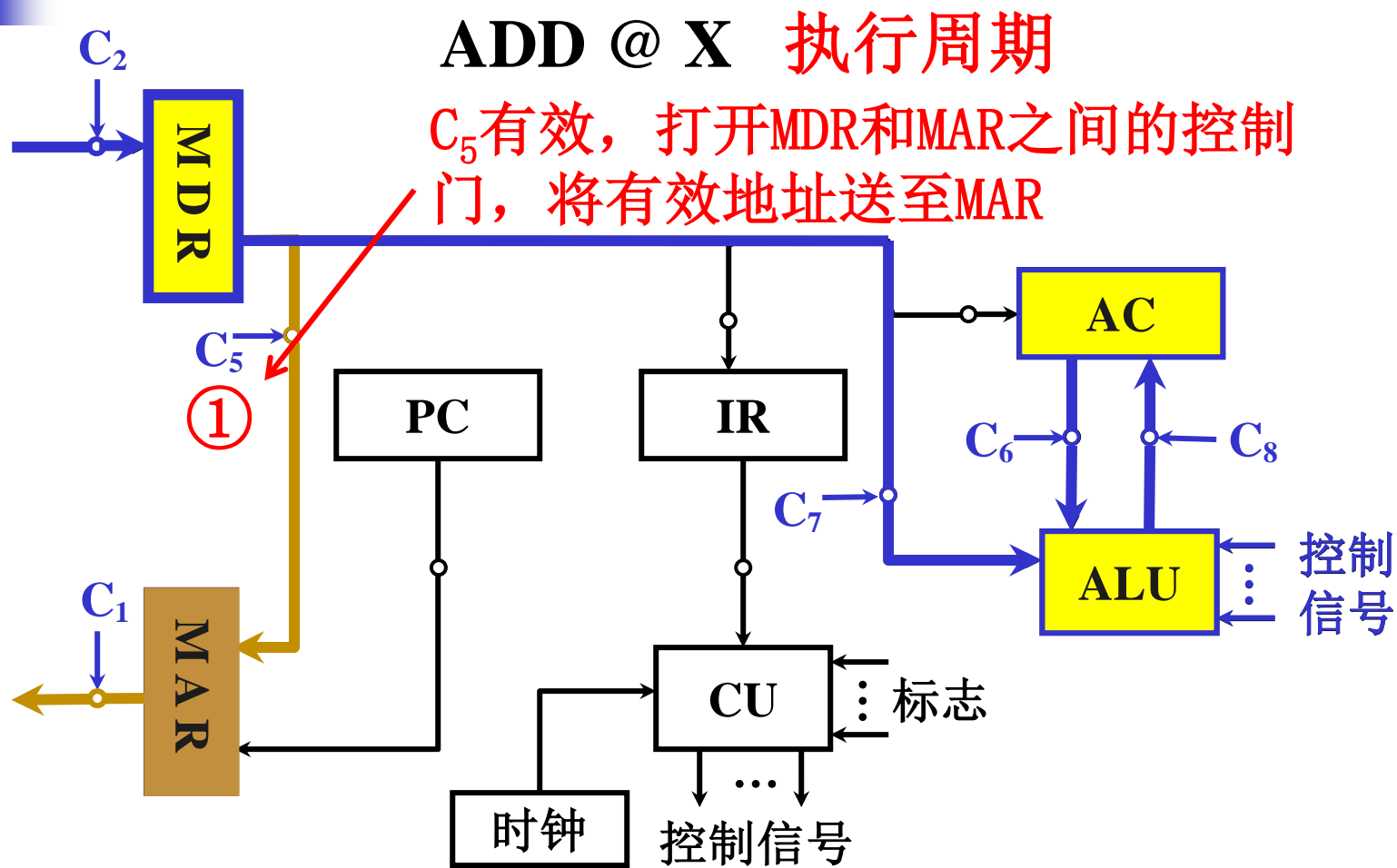
ADD @ X 执行周期

共7步 ①~⑦



二、控制信号举例

1. 不采用 CPU 内部总线的方式



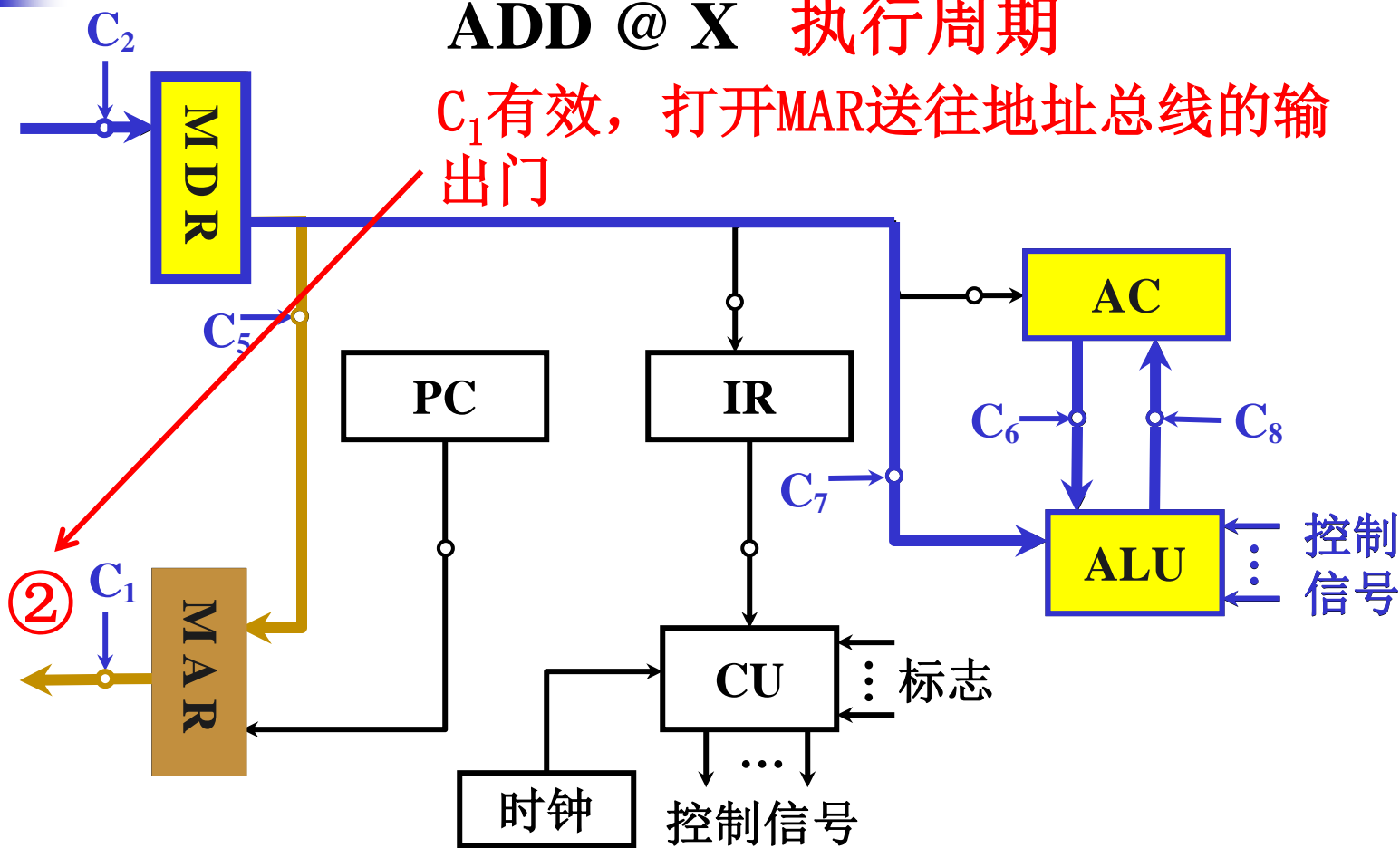
二、控制信号举例

1. 不采用 CPU 内部总线的方式

ADD A, ((X))

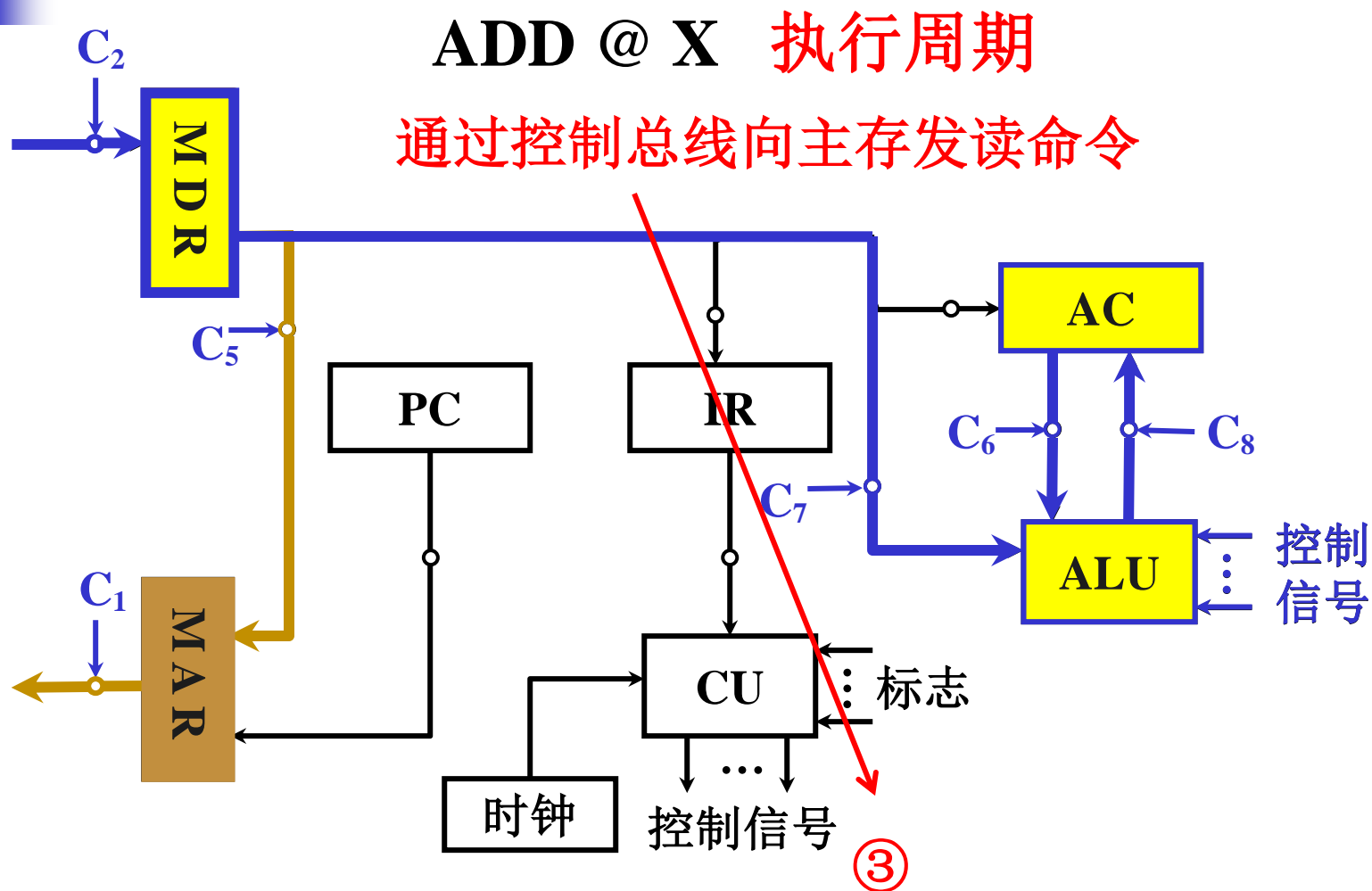
ADD @ X 执行周期

C₁有效，打开MAR送往地址总线的输出
门



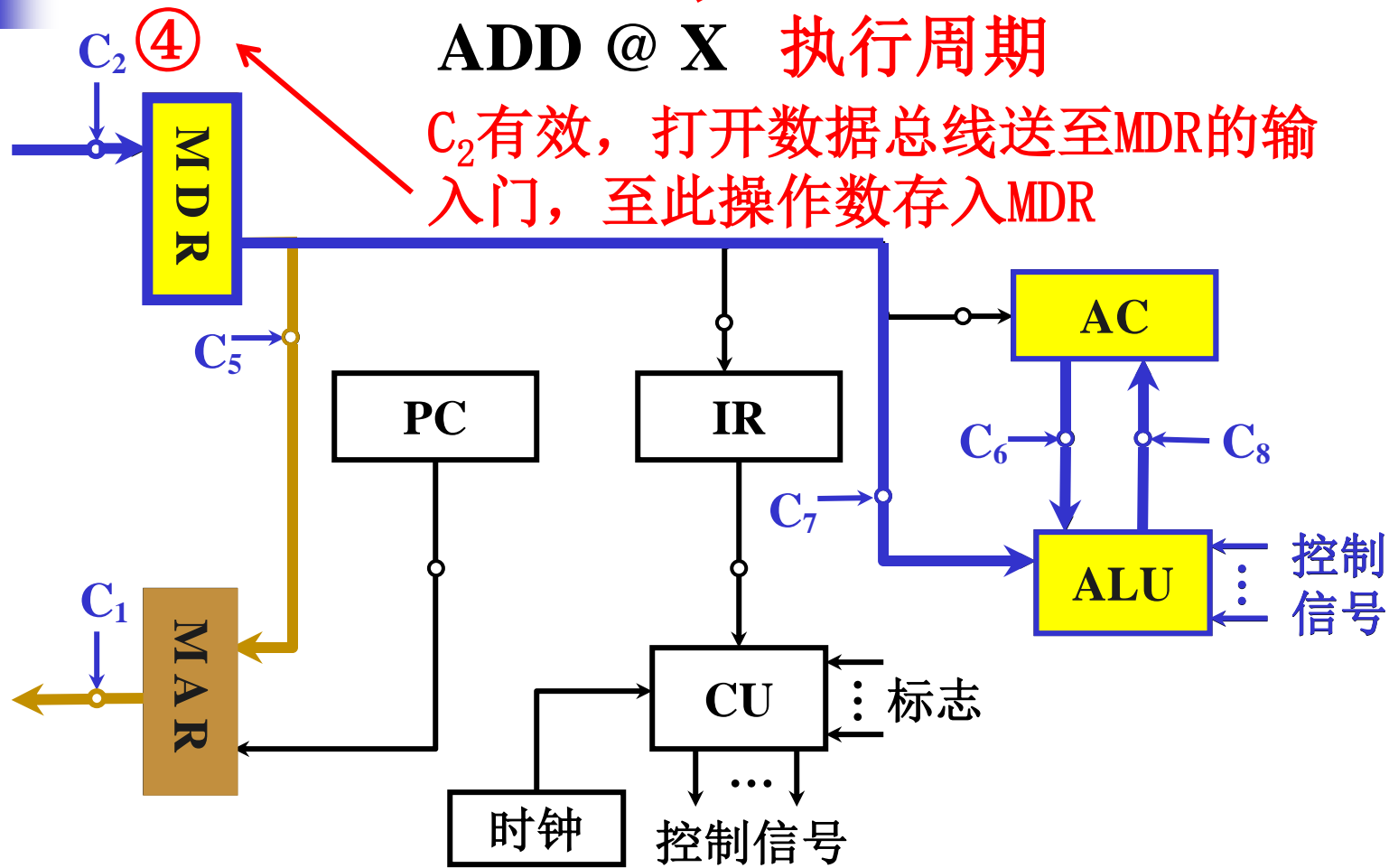
二、控制信号举例

1. 不采用 CPU 内部总线的方式



二、控制信号举例

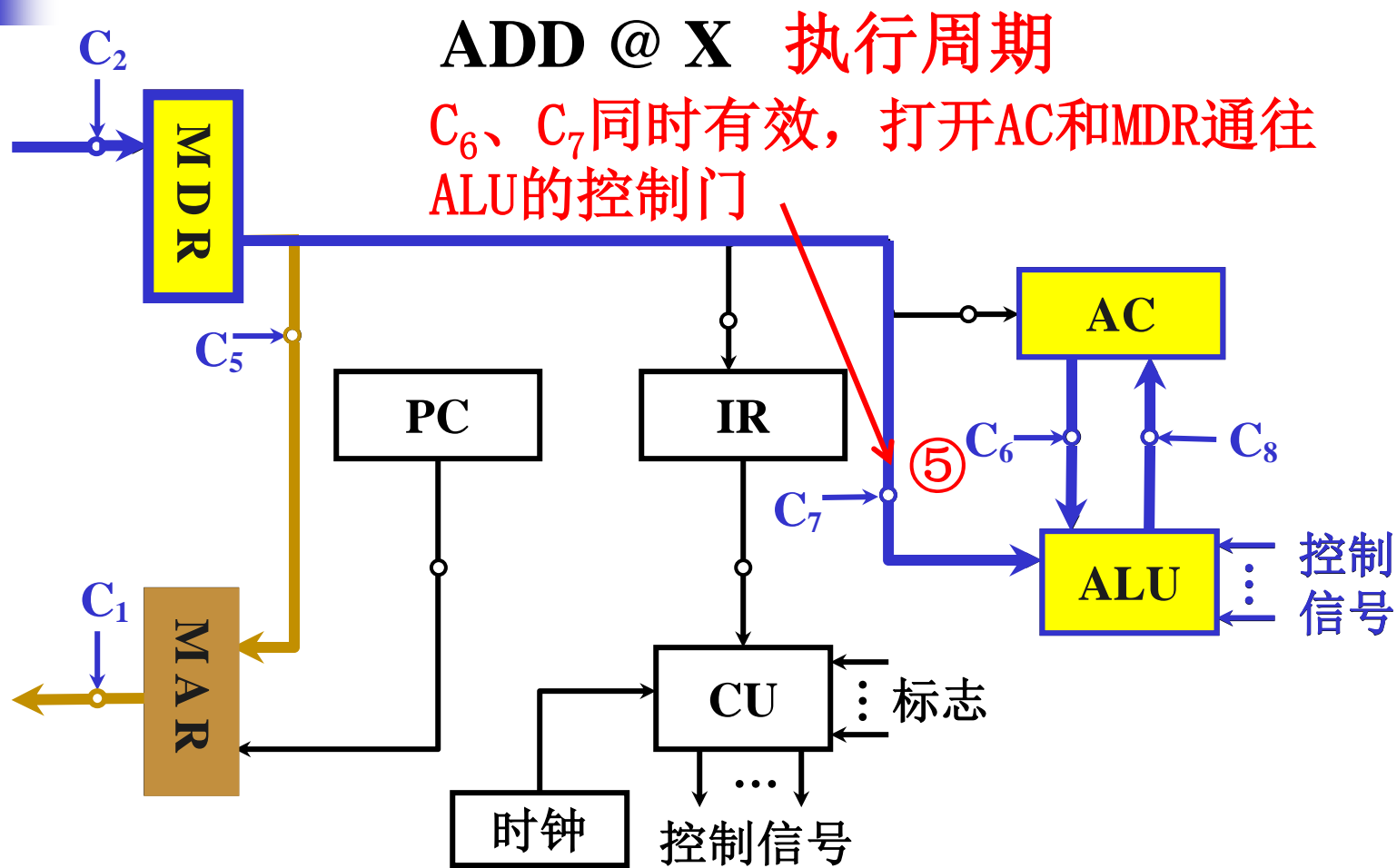
1. 不采用 CPU 内部总线的方式



二、控制信号举例

1. 不采用 CPU 内部总线的方式

ADD A, ((X))



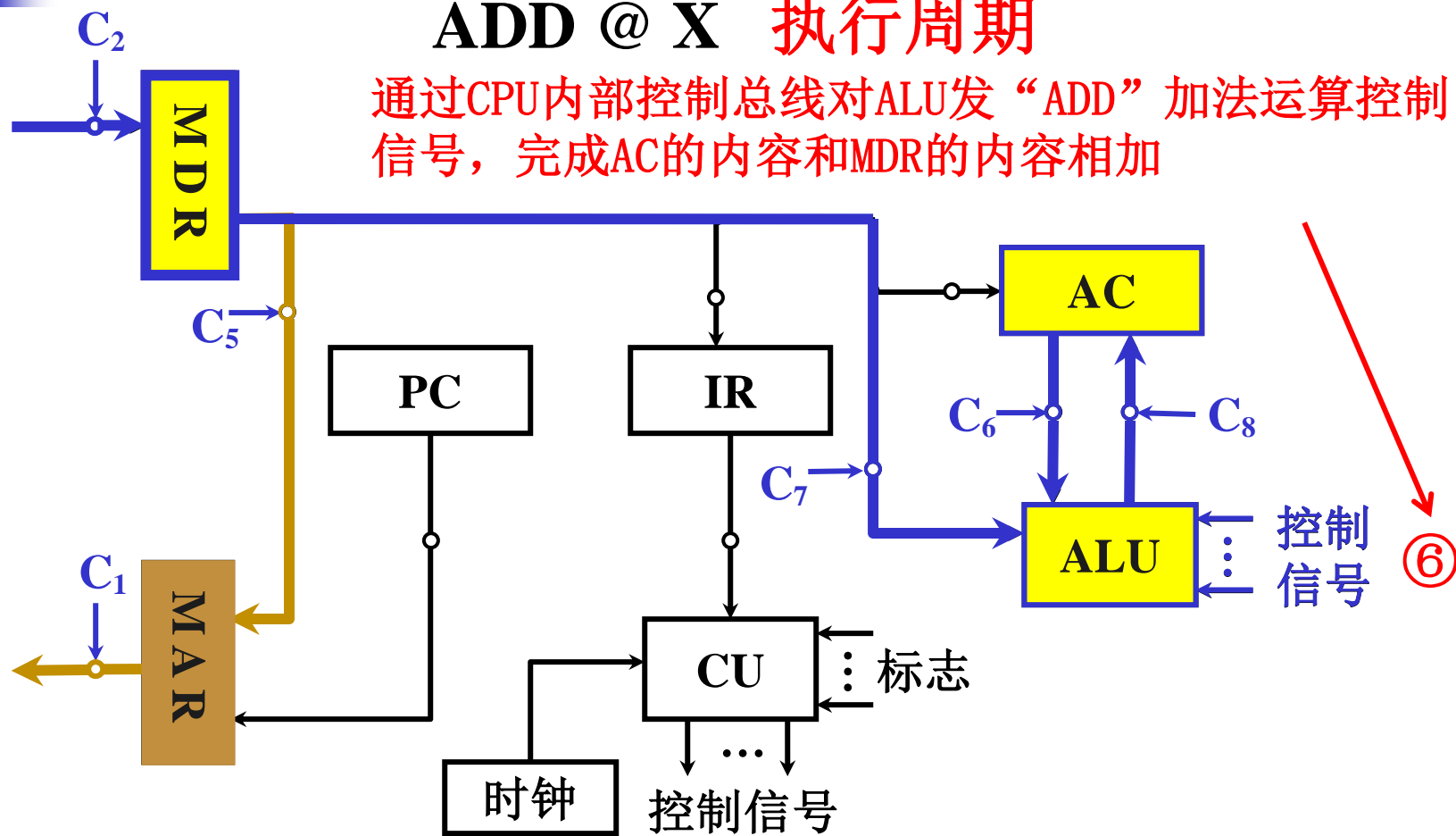
二、控制信号举例

1. 不采用 CPU 内部总线的方式

ADD A, ((X))

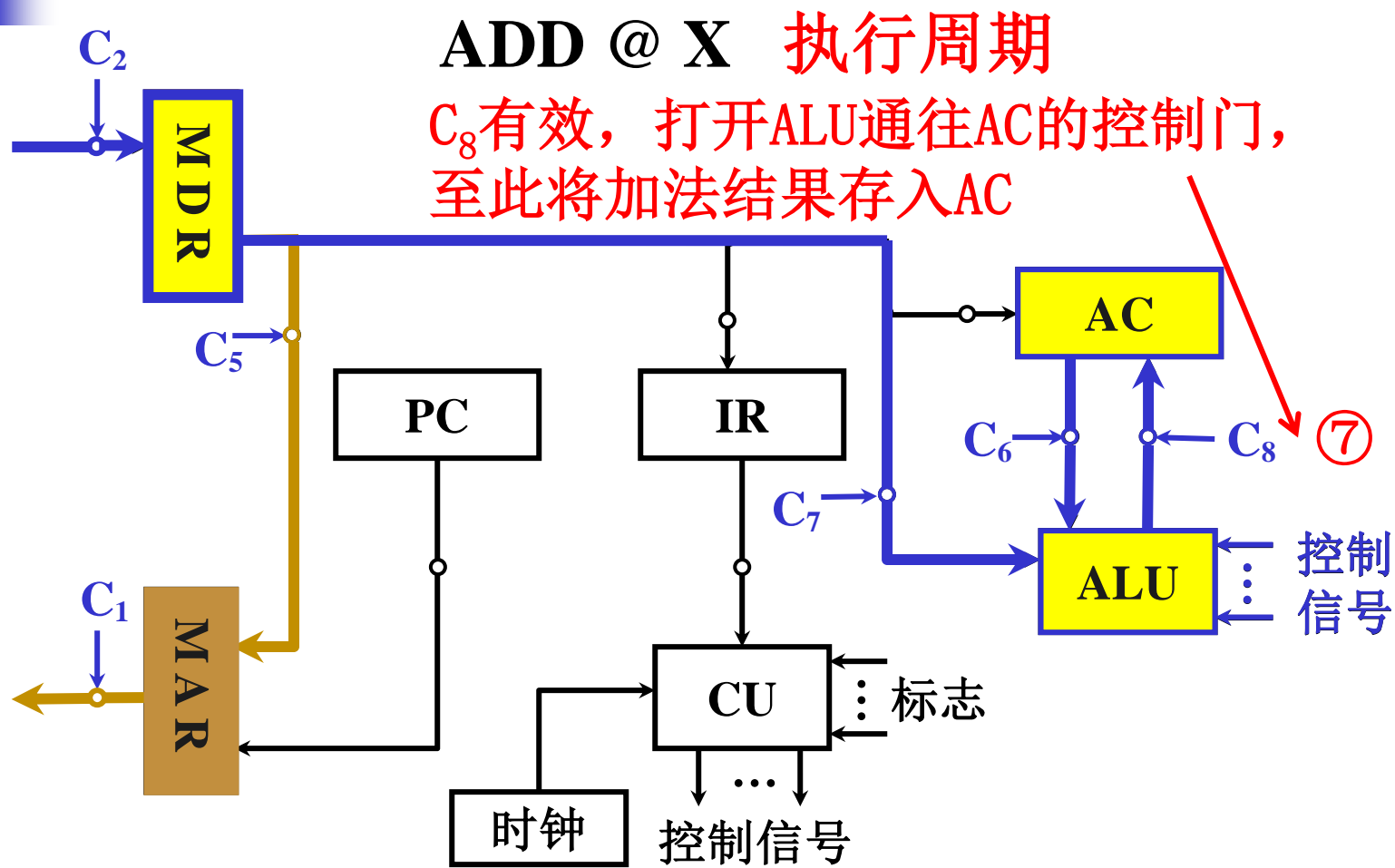
ADD @ X 执行周期

通过CPU内部控制总线对ALU发“ADD”加法运算控制信号，完成AC的内容和MDR的内容相加



二、控制信号举例

1. 不采用 CPU 内部总线的方式



2. 采用 CPU 内部总线方式

(1) ADD @ X 取指周期

① • PC \rightarrow MAR \rightarrow 地址线
PC₀ MAR_i

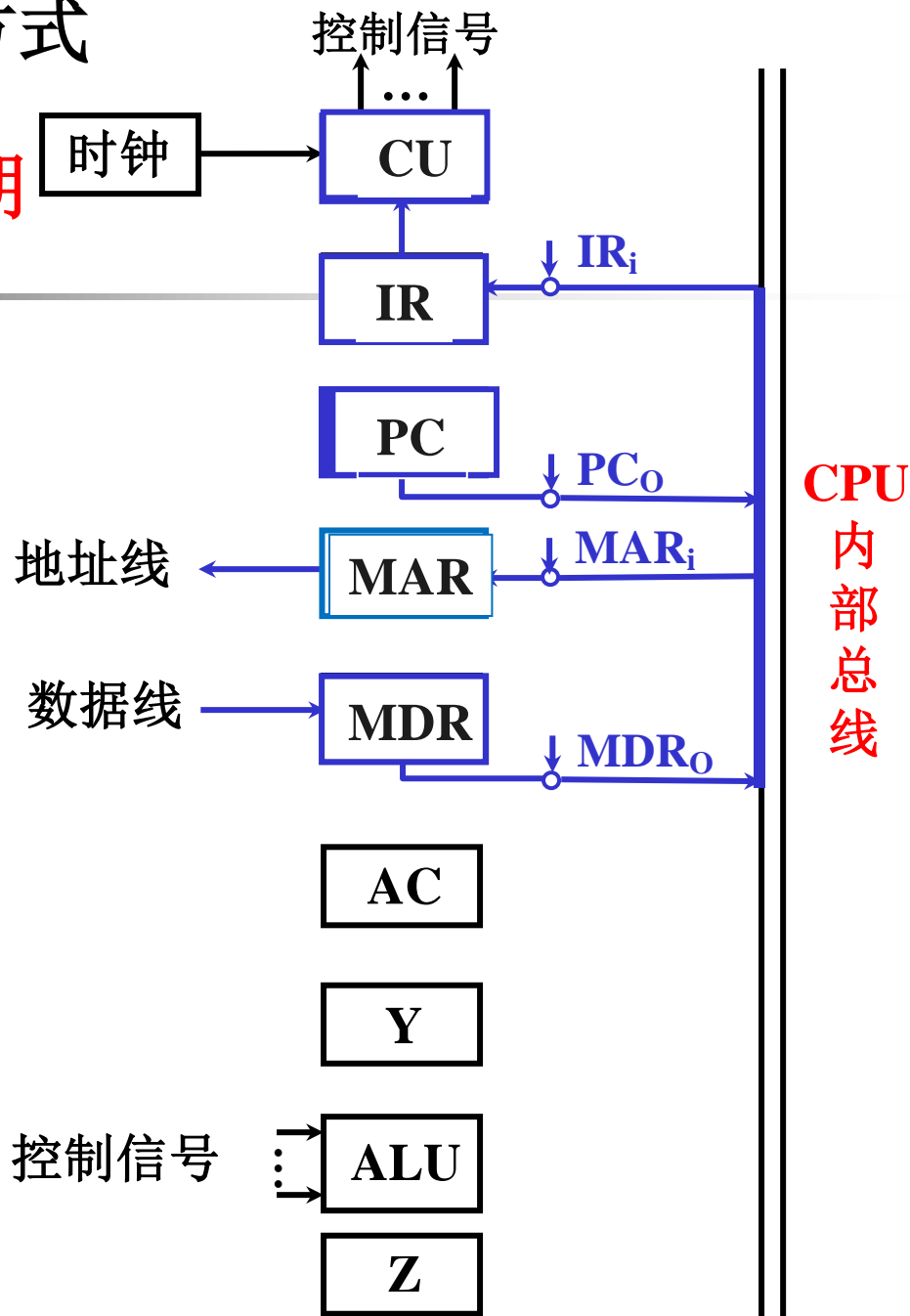
② • CU 发读命令 1 \rightarrow R

③ • 数据线 \rightarrow MDR

④ • MDR \rightarrow IR
MDR₀ IR_i

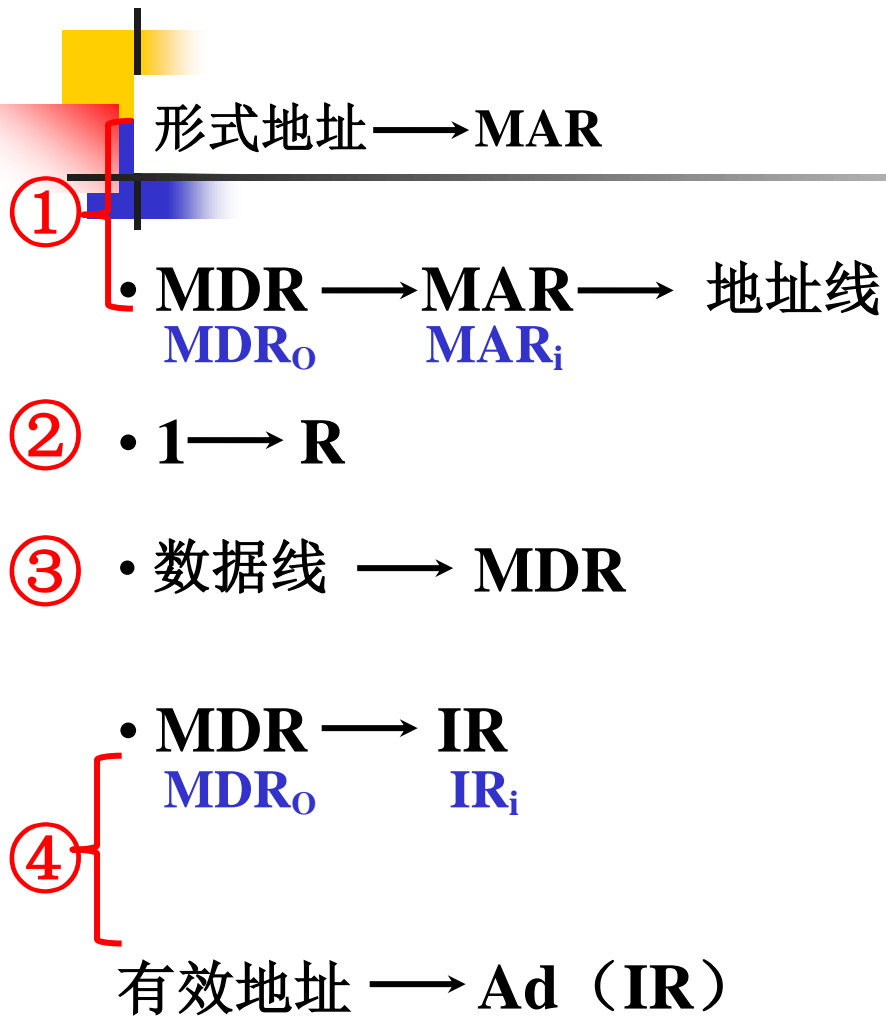
• OP (IR) \rightarrow CU

⑤ • (PC) + 1 \rightarrow PC

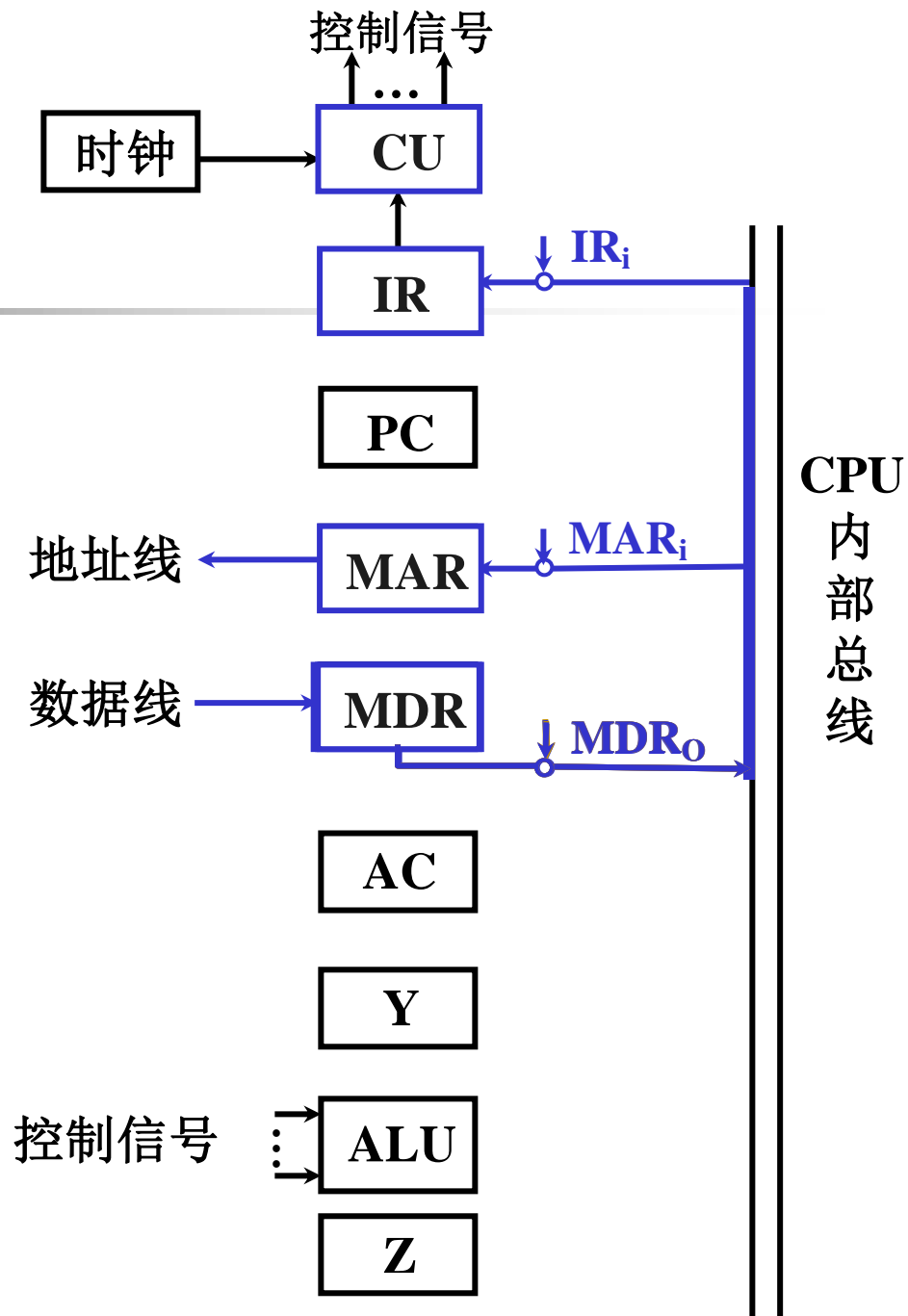


非总线方式7步

(2) ADD @ X 间址周期



非总线方式5步



(3) ADD @ X 执行周期

① • MDR \rightarrow MAR \rightarrow 地址线
MDR₀ MAR_i

② • 1 \rightarrow R

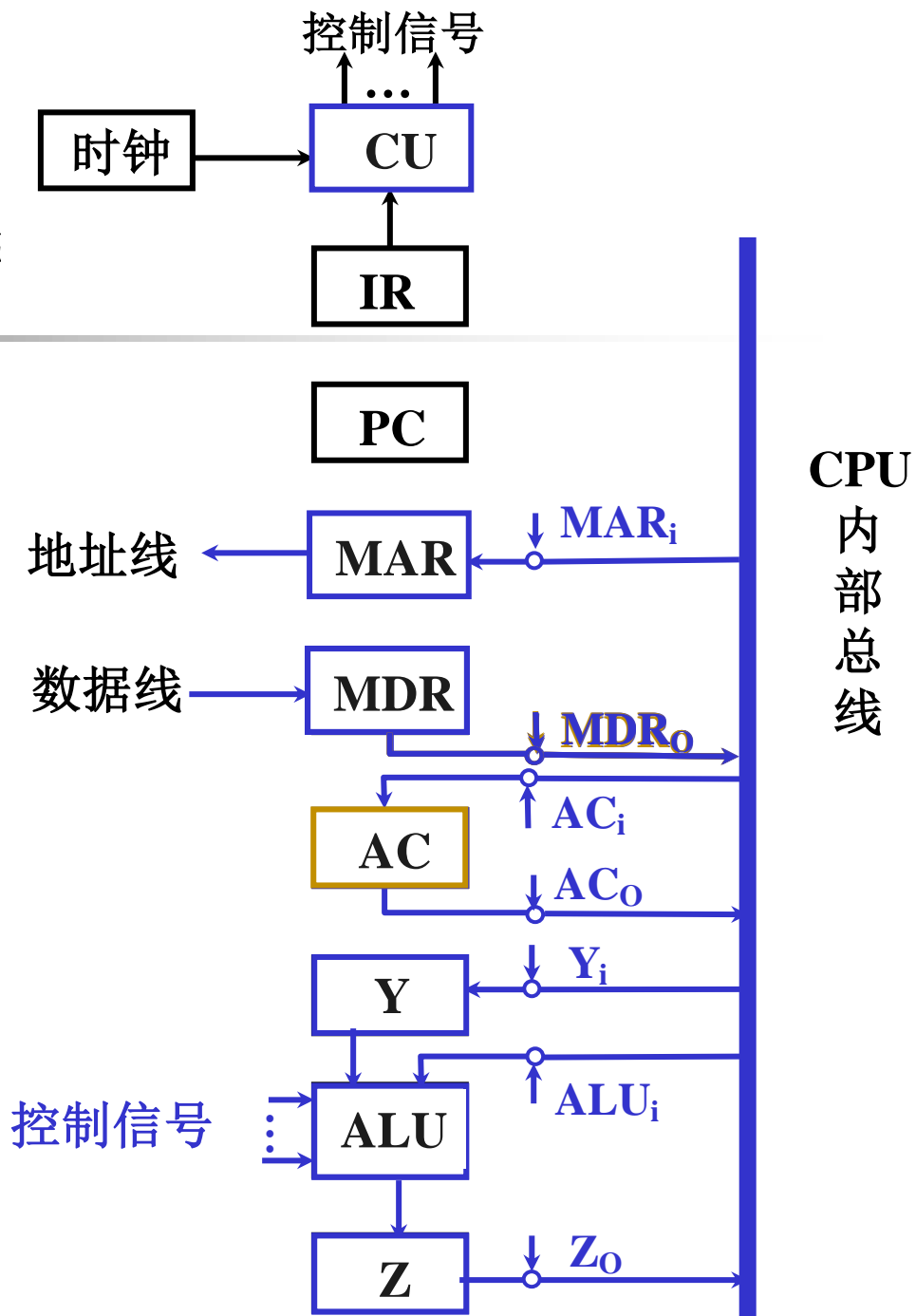
③ • 数据线 \rightarrow MDR

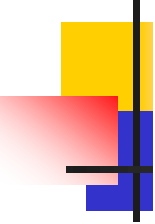
④ • MDR \rightarrow Y \rightarrow ALU
MDR₀ Y_i

⑤ { • AC \rightarrow ALU
AC₀ ALU_i
• (AC) + (Y) \rightarrow Z

⑥ • Z \rightarrow AC
Z₀ AC_i

非总线方式7步





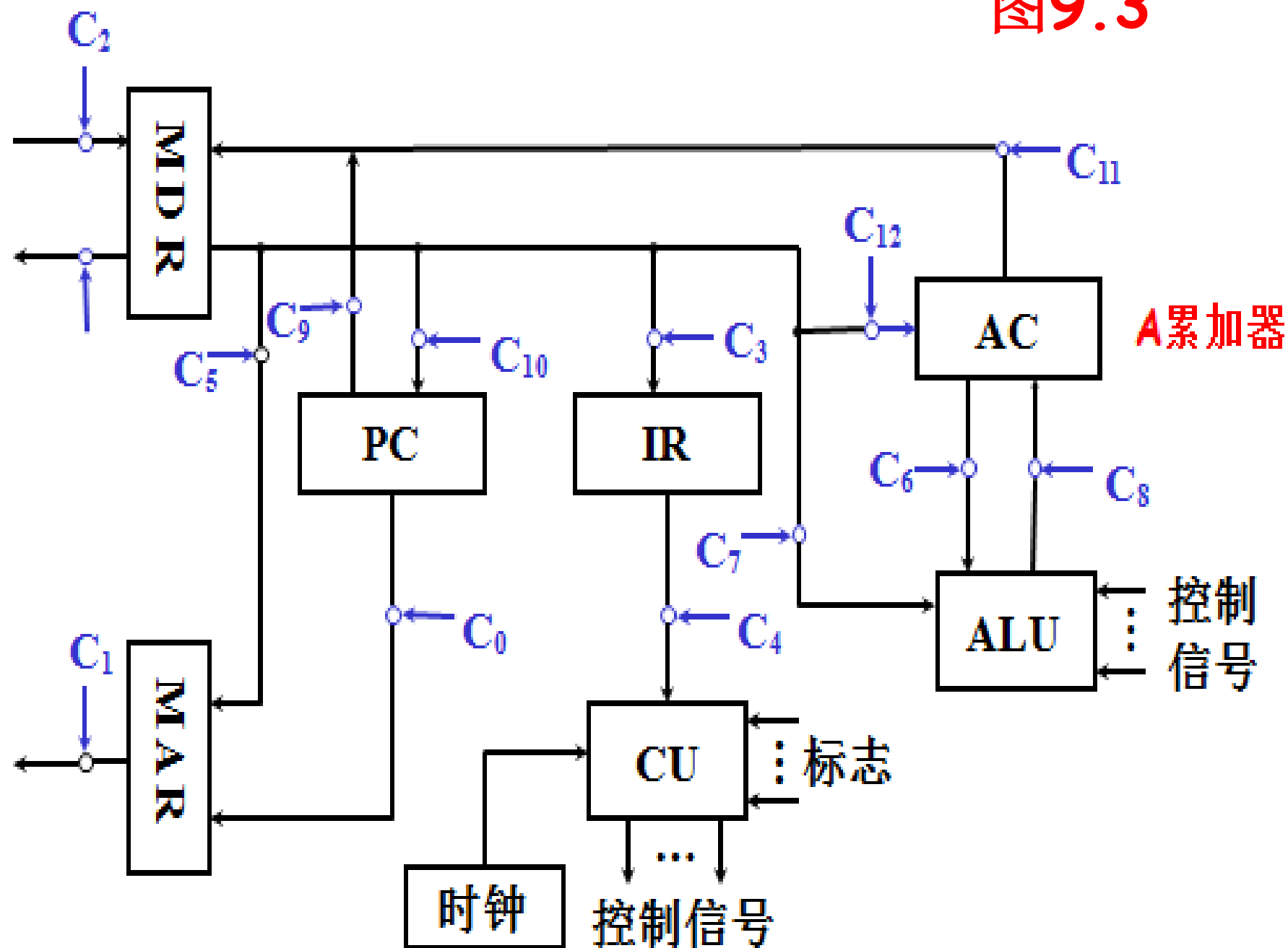
■ **例9.1：** CPU内部采用非总线结构，如图9.3所示：

- 写出取指周期的全部微操作；
- 写出取数指令“**LDA M**”、存数指令“**STA M**”、加法指令“**ADD M**”在执行阶段所需的全部微操作；
- 当上述指令均为间接寻址时，写出执行这些指令所需的全部微操作；
- 写出无条件转移指令“**JMP Y**”和结果为零则转指令“**BAZ Y**”在执行阶段所需的全部微操作。

■ **解：**

CPU 内部结构采用非总线方式

图9.3





■ (1)取指周期的全部微操作

- **PC->MAR** 现行指令地址->**MAR**
- **1->R** 命令存储器读
- **M(MAR)-MDR** 现行指令从存储器中读至**MDR**
- **MDR->IR** 现行指令->**IR**
- **OP(IR)->CU** 指令的操作码->**CU** 译码
- **(PC)+1->PC** 形成下一条指令的地址



■ (2) 执行周期

■ ①取数指令“**LDA M**”执行阶段所需的全部微操作

- **Ad(IR)->MAR** 指令的地址码字段->**MAR**
- **1->R** 命令存储器读
- **M(MAR)->MDR** 操作数从存储器中读至**MDR**
- **MDR->ACC** 操作数->**ACC**

■ ②存数指令“**STA M**”执行阶段所需的全部微操作

- **Ad(IR)->MAR** 指令的地址码字段->**MAR**
- **1->W** 命令存储器写
- **ACC->MDR** 欲写入的数据->**MDR**
- **MDR-> M(MAR)** 数据写至存储器中

■ ③加法指令“**ADD M**”执行阶段所需的全部微操作

- **Ad(IR)->MAR** 指令的地址码字段->**MAR**
- **1->R** 命令存储器读
- **M(MAR)->MDR** 操作数从存储器中读至**MDR**
- **(ACC)+(MDR)->ACC** 两数相加结果送**ACC**



■ (3)间址周期

- 当上述指令为间接寻址时，需增加**间址周期**的微操作，间址周期的微操作为：
 - **Ad(IR)->MAR** 指令的地址码字段（形式地址）->**MAR**
 - **1->R** 命令存储器读
 - **M(MAR)->MDR** 有效地址从存储器中读至**MDR**
- 进入执行周期，**3**条指令的第一个微操作均为**MDR->MAR**（原来是**Ad(IR)->MAR**），其余微操作不变



■ (4) 转移指令

- ①无条件转移指令“JMP Y” 执行阶段的微操作如下：

- $Ad(IR) \rightarrow PC$ 转移(目标)地址 $Y \rightarrow PC$

- ②结果为零则转指令“BAZ Y” 执行阶段的微操作如下：

- $Z \cdot Ad(IR) \rightarrow PC$ 当 $Z=1$ 时，转移(目标)地址 $Y \rightarrow PC$

当 $Z=0$ 时（结果不为0），什么也不做

- **例9.2:** 单总线计算机结构如图9.5所示, 其中**M**为主存储器, **XR**为变址寄存器, **EAR**为有效地址寄存器, **LATCH**为锁存器。假设指令地址已存于**PC**中, 请画出“**ADD X,D**”(X为变址寄存器**XR**, **D**为形式地址)和“**STA *D**”(*表示相对寻址, **D**为相对位移量)两条指令的指令周期信息流程图, 并列出的控制信号序列。

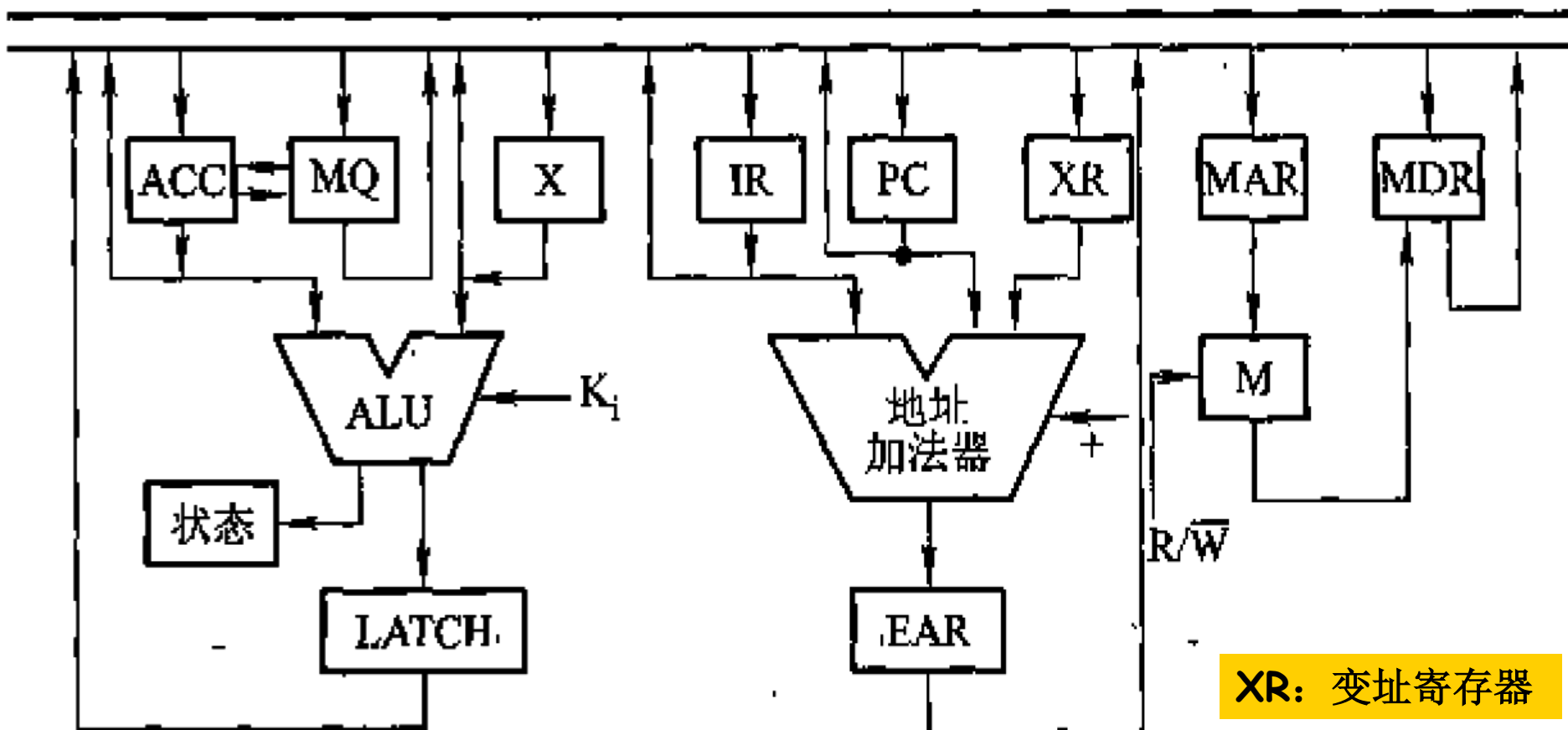
$ACC \rightarrow (PC+D)$

- **解:** $(XR+D) + ACC \rightarrow ACC$

MQ: 乘商寄存器

乘法时放乘积的低位部分，除法时放商

Multiplier & Quotient



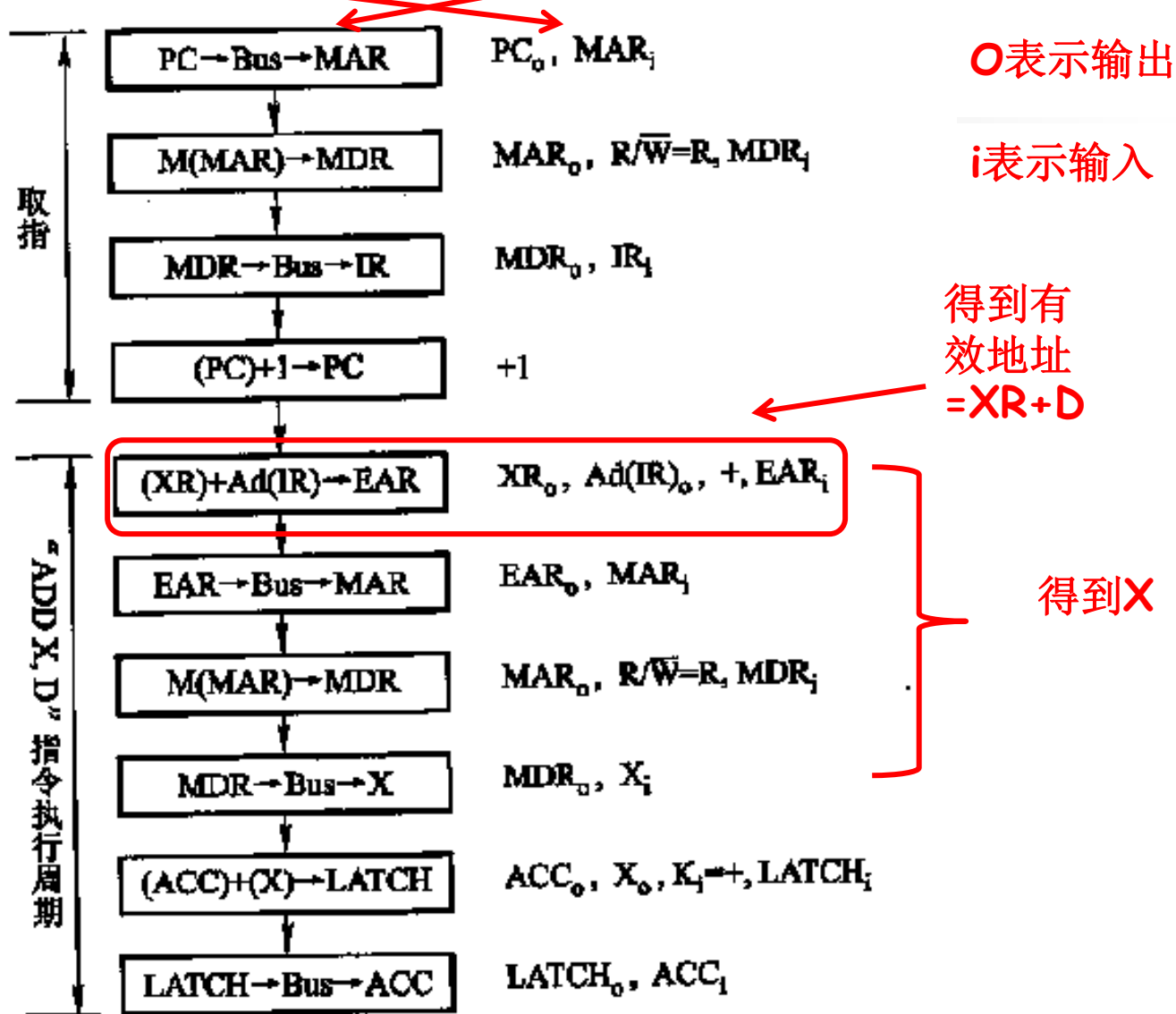
XR: 变址寄存器

EAR: 有效地址寄存器

LATCH: 锁存器

图 9.5 单总线计算机结构

- (1) “ADD X,D”指令取指周期和执行周期的信息流程及相应的控制信号如下，其中 $Ad(IR)$ 为形式地址：

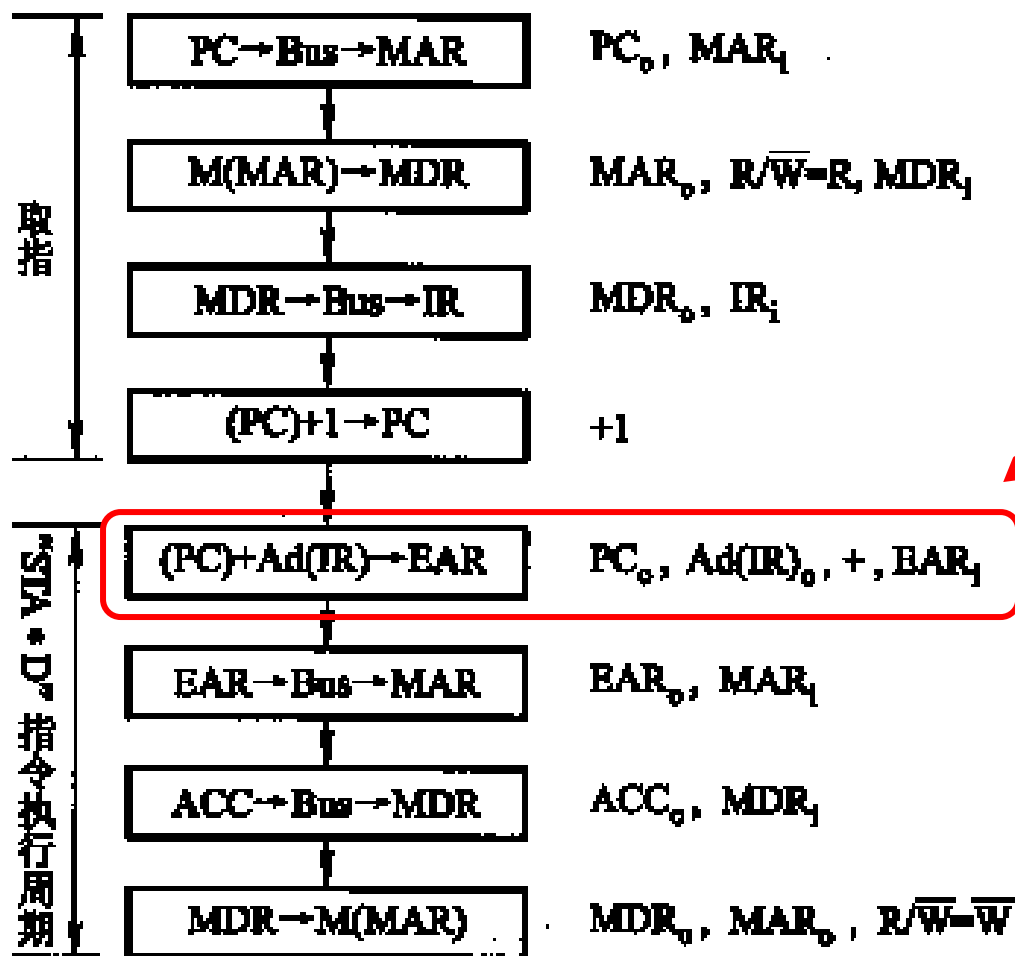


ADD AL, [XR+D]

变址寻址

- (2) “**STA *D**”指令取指周期和执行周期的信息流程及相应的控制信号如下，其中**Ad(IR)**为相对位移量：

取指周期
一样！



得到有效地址
=PC+D

MOV [PC+D], AL

相对寻址

三、多级时序系统

1. 机器周期

- “单机器指令”方式运行，每次执行1条指令
- “单周期”方式运行，每次执行1条微指令
- “单节拍”方式运行，每次执行1个时钟周期（T周期）

(1) 机器周期的概念

所有指令执行过程中的一个基准时间

(2) 确定机器周期需考虑的因素

每条指令的执行 步骤

每一步骤 所需的 时间

(3) 基准时间的确定

- 以完成 最复杂 指令功能的时间 为准
- 以 访问一次存储器 的时间 为基准

若指令字长 = 存储字长 取指周期 = 机器周期

2. 时钟周期（节拍、状态）

- “单机器指令”方式运行，每次执行**1**条指令
- “单周期”方式运行，每次执行**1**条微指令
- “单节拍”方式运行，每次执行**1**个时钟周期（**T**周期）

一个机器周期内可完成若干个微操作

每个微操作需一定的时间

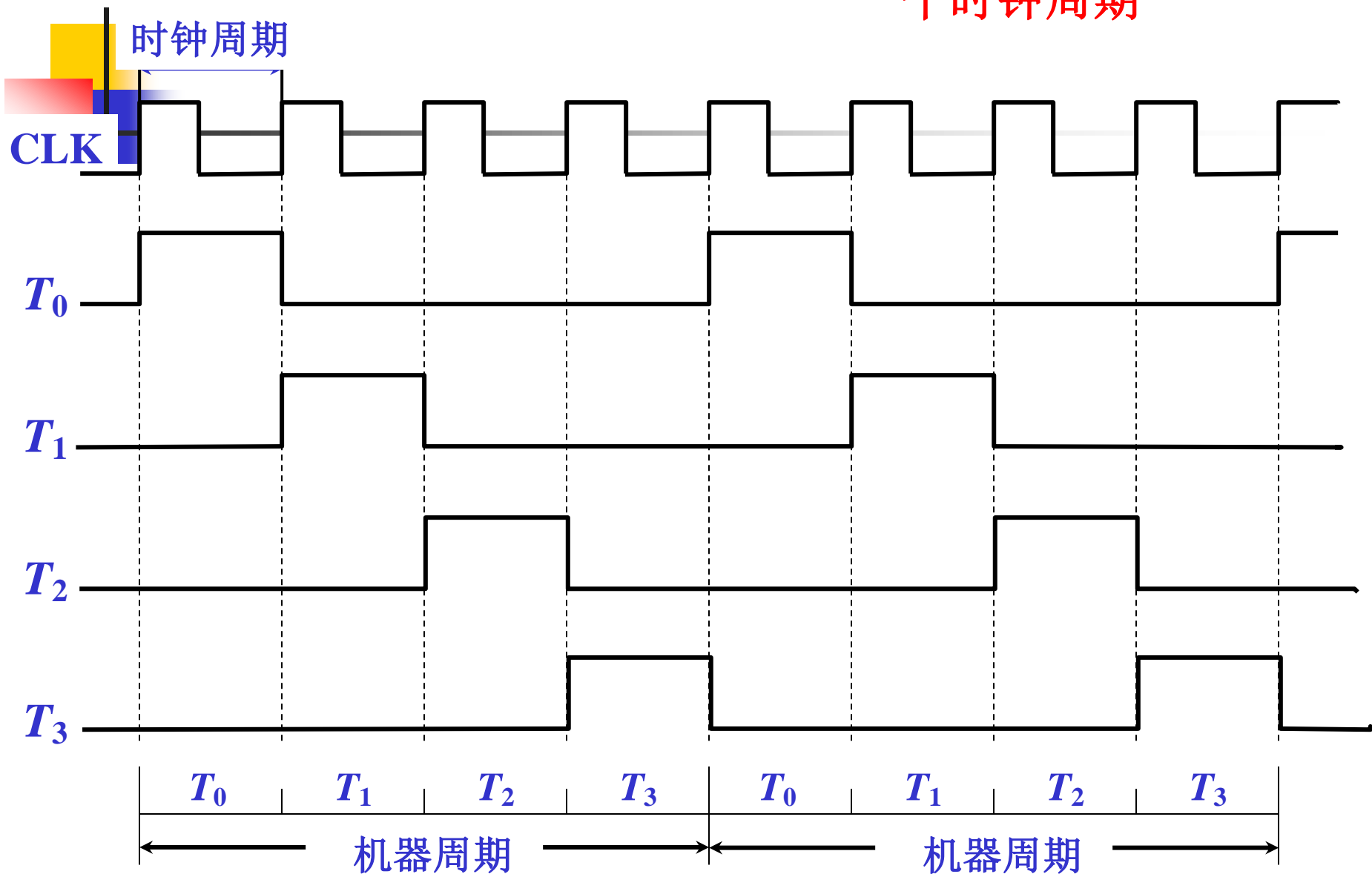
将一个机器周期分成若干个时间相等的时间段（节拍、状态、时钟周期）

时钟周期是控制计算机操作的最小单位时间

用时钟周期控制产生一个或几个微操作命令

2. 时钟周期（节拍、状态）

一个机器周期等于4个时钟周期



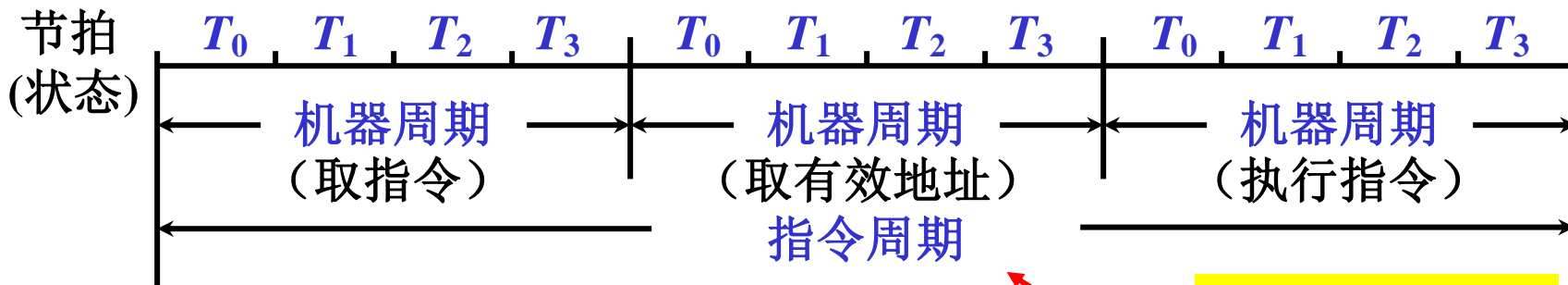
3. 多级时序系统

- “单机器指令”方式运行，每次执行1条指令
- “单周期”方式运行，每次执行1条微指令
- “单节拍”方式运行，每次执行1个时钟周期（ T 周期）

机器周期、节拍（状态）组成多级时序系统

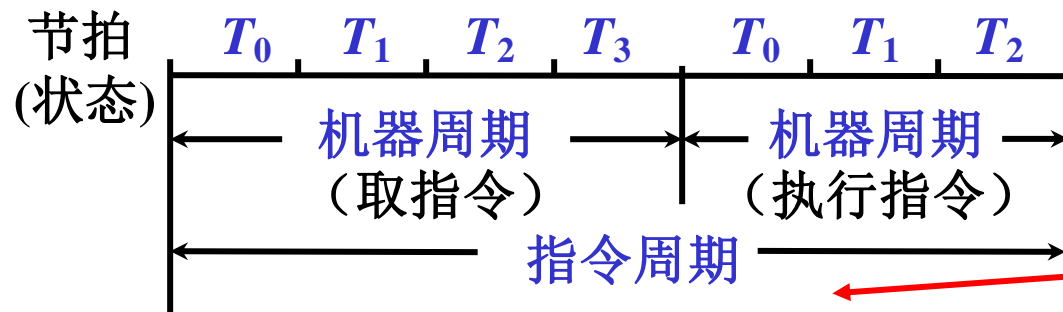
一个指令周期包含若干个机器周期（如2个或3个）

一个机器周期包含若干个时钟周期（如4个）



每个指令周期的长度一样

定长的机器周期



每个指令周期的长度不一样

不定长的机器周期

4. 机器速度与机器主频的关系

机器的 主频 f 越快 机器的 速度也越快

在机器周期所含时钟周期数 相同 的前提下，
两机 平均指令执行速度之比 等于 两机主频之比

$$\frac{\text{MIPS}_1}{\text{MIPS}_2} = \frac{f_1}{f_2}$$

机器速度 不仅与 主频有关，还与机器周期中所含
时钟周期（主频的倒数）数 以及指令周期中所含
的 机器周期数有关

- **例9.3:** CPU主频=8MHz, 每个机器周期平均含2个时钟周期, 每条指令的指令周期平均有2.5个机器周期, 试问该机器的平均指令执行速度为多少**MIPS**?
- 若主频不变, 每个机器周期平均含4个时钟周期, 每条指令的指令周期平均有5个机器周期, 试问该机器的平均指令执行速度为多少**MIPS**? 得出什么结论?

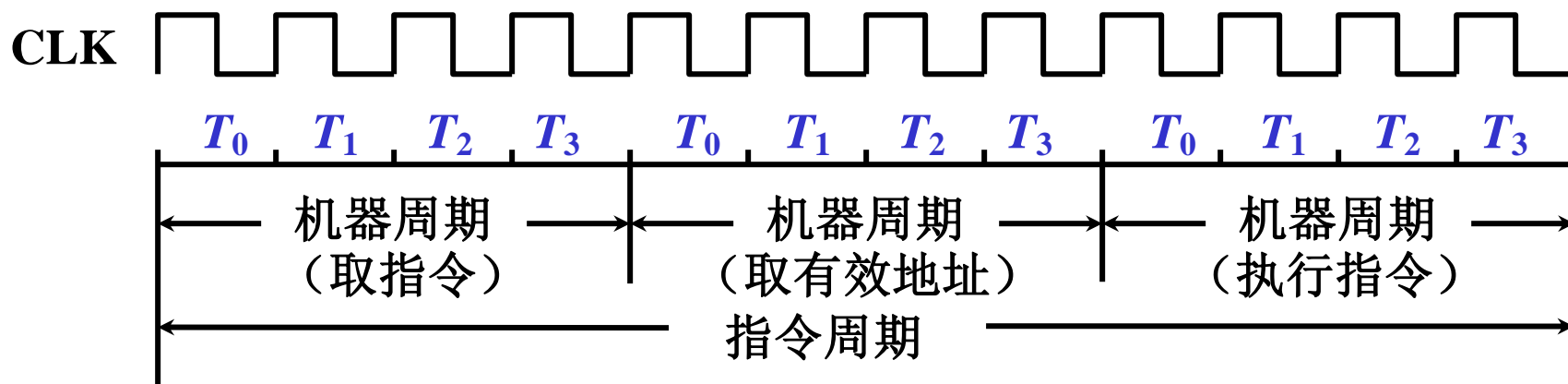
- **例9.3:** CPU主频=8MHz, 每个机器周期平均含2个时钟周期, 每条指令的指令周期平均有2.5个机器周期, 试问该机器的平均指令执行速度为多少**MIPS**?
- 若主频不变, 每个机器周期平均含4个时钟周期, 每条指令的指令周期平均有5个机器周期, 试问该机器的平均指令执行速度为多少**MIPS**? 得出什么结论?
- **解:**
- 时钟周期= $1/(8\text{MHz})=0.125\mu\text{s}$,
机器周期= $0.125\mu\text{s}\times 2=0.25\mu\text{s}$,
指令周期= $0.25\mu\text{s}\times 2.5=0.625\mu\text{s}$,
平均指令执行速度为 $1/0.625\mu\text{s}=1.6\text{MIPS}$
- 若CPU主频不变, 每个机器周期平均含4个时钟周期, 每条指令的指令周期平均有5个机器周期, 则指令周期= $0.125\mu\text{s}\times 4\times 5=2.5\mu\text{s}$, 故平均指令执行速度= $1/2.5\mu\text{s}=0.4\text{MIPS}$
- 结论: 机器的速度并不完全取决于主频

四、控制方式

产生不同微操作命令序列所用的时序控制方式

1. 同步控制方式

任一微操作均由 **统一基准时标** 的时序信号控制

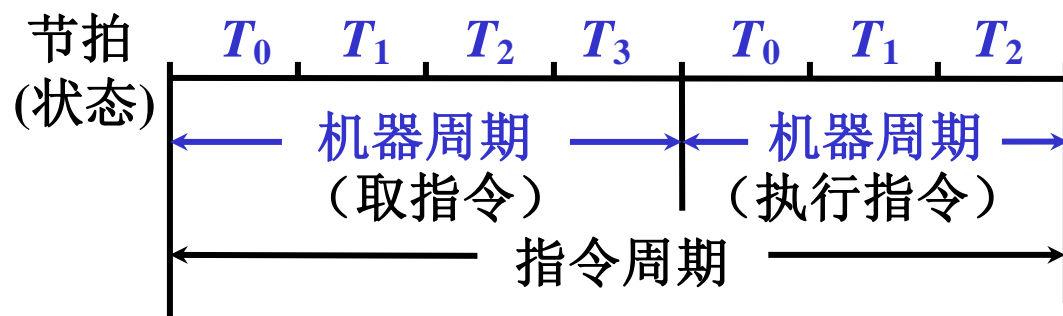


(1) 采用 定长 的机器周期

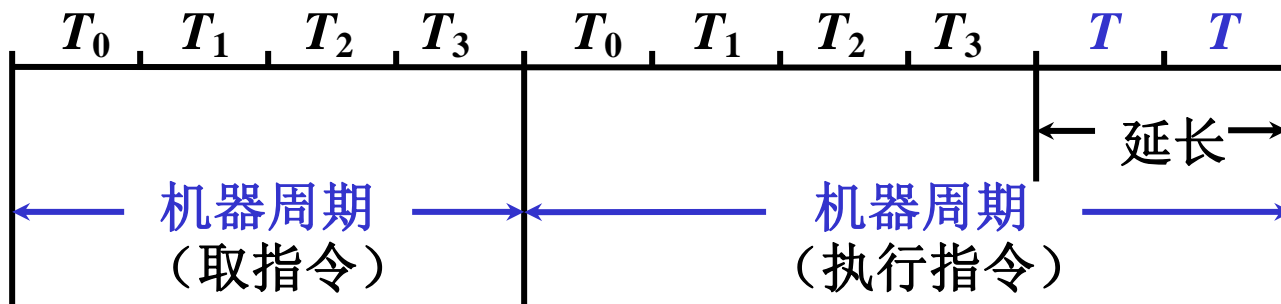
以 **最长** 的 **微操作序列** 和 **最繁** 的微操作作为 **标准**
机器周期内 **节拍数相同** (如都是**4个**)

(2) 采用不定长的机器周期

机器周期内 节拍数不等

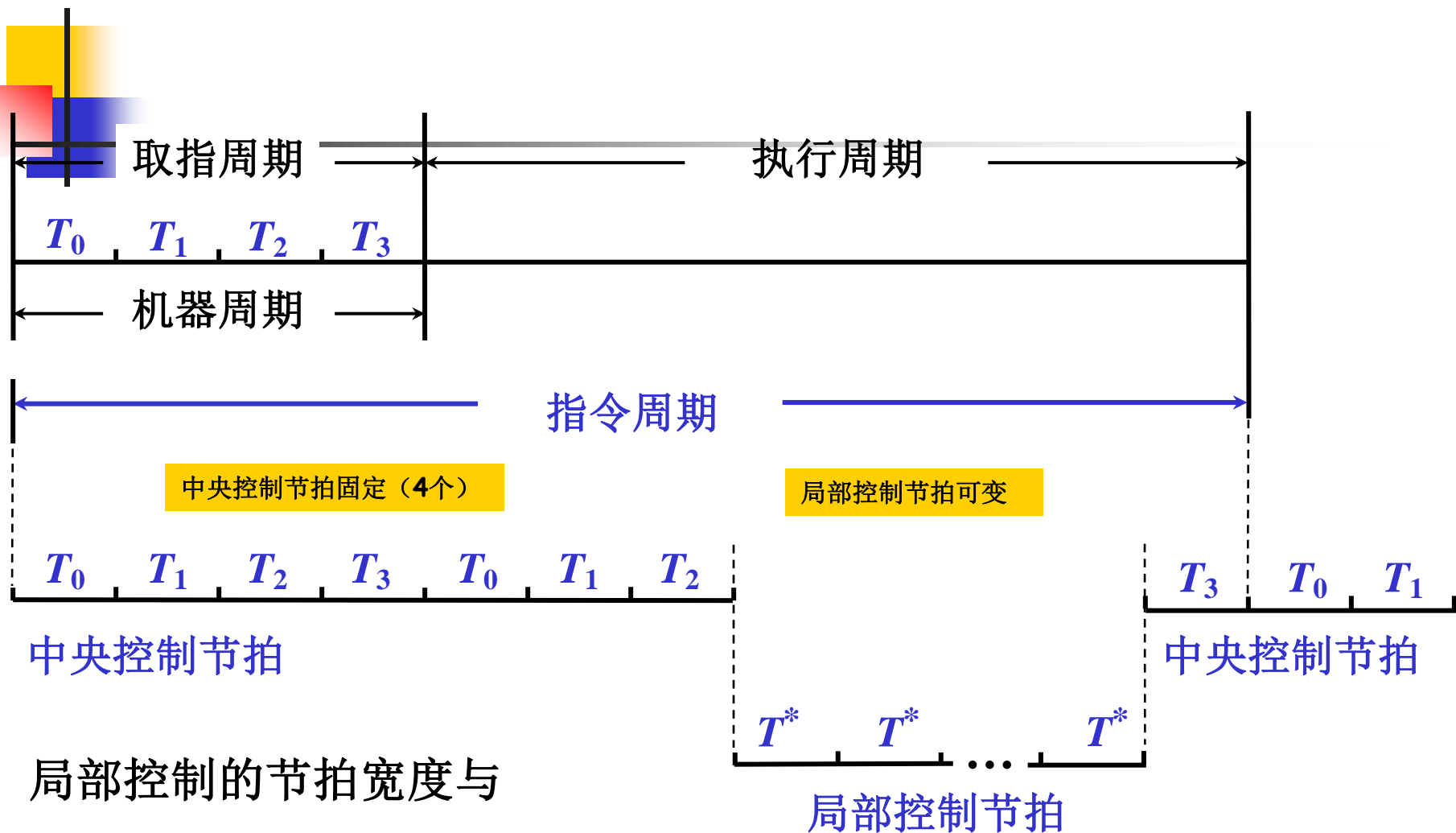


执行指令的机器周期=3个时钟周期



执行指令的机器周期=6个时钟周期

(3) 采用中央控制和局部控制相结合的方法



局部控制的节拍宽度与

中央控制的节拍宽度一致

中央控制节拍均为4个时钟周期

将少数操作复杂的指令中的某些操作（如乘除法和浮点运算等）采用局部控制方式来完成

2. 异步控制方式

无基准时钟信号

无固定的周期节拍和严格的时钟同步

采用 应答方式

3. 联合控制方式

同步与异步相结合

4. 人工控制方式

(1) **Reset**

(2) 连续 和 单条 指令执行转换开关

(3) 符合停机开关

连续执行、单步执行

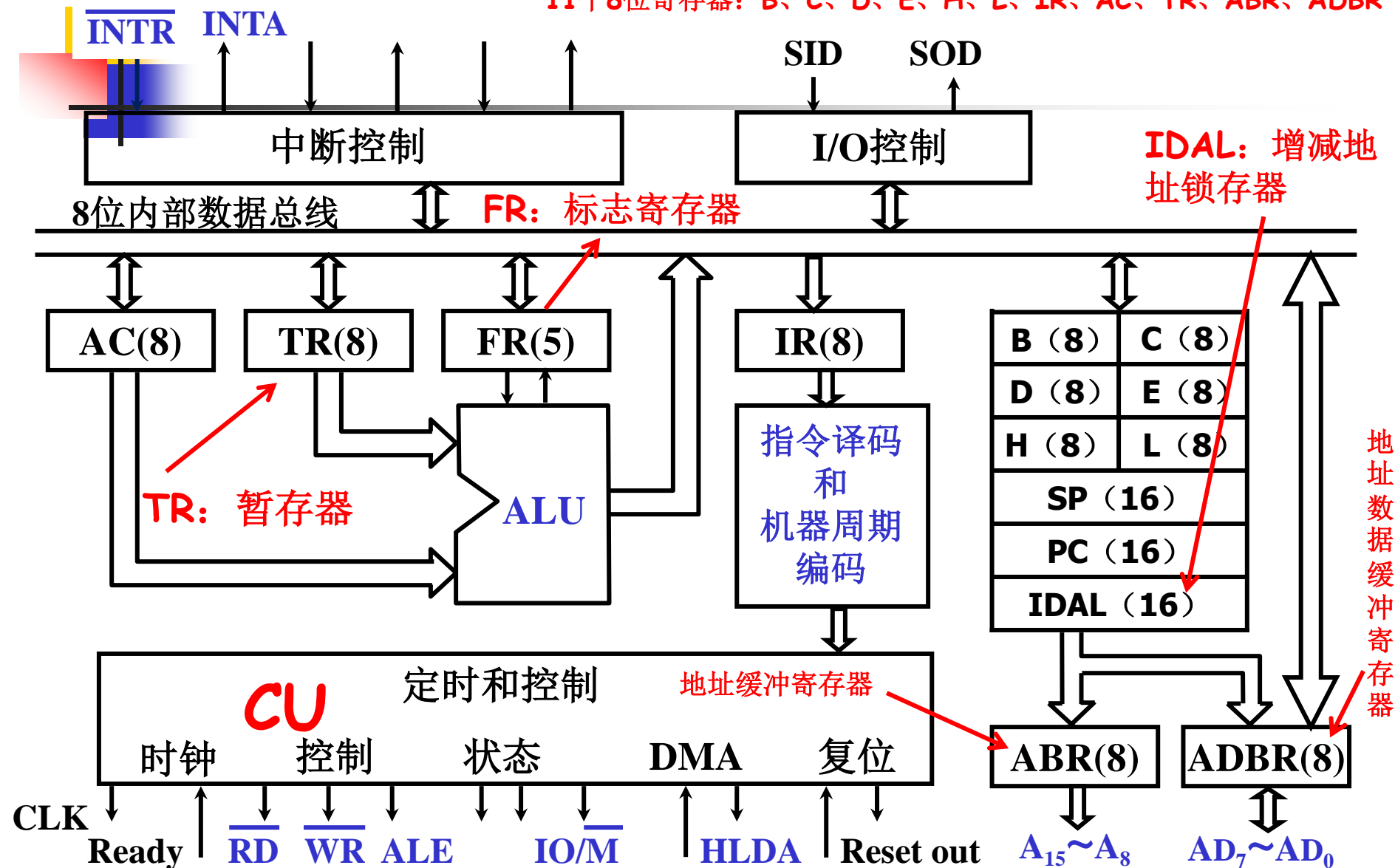
五、多级时序系统实例分析

1. 8085CPU 的组成

3个16位寄存器: SP、PC、IDAL

1个5位状态寄存器: FR

11个8位寄存器: B、C、D、E、H、L、IR、AC、TR、ABR、ADBR



2. 8085CPU 的外部引脚

(1) 地址和数据信号

$A_{15} \sim A_8$ $AD_7 \sim AD_0$

SID: 串行输入 SOD: 串行输出

X_1	1	40	V_{CC}
X_2	2	39	HOLD
Reset out	3	38	HLDA
SOD	4	37	CLK(out)
SID	5	36	Rstet in
Trap	6	35	Ready
RST7.5	7	34	$\overline{IO/M}$
RST6.5	8	33	S_1
RST5.5	9	32	\overline{RD}
\overline{INTR}	10	31	\overline{WR}
INTA	11	30	ALE
AD_0	12	29	S_0
AD_1	13	28	A_{15}
AD_2	14	27	A_{14}
AD_3	15	26	A_{13}
AD_4	16	25	A_{12}
AD_5	17	24	A_{11}
AD_6	18	23	A_{10}
AD_7	19	22	A_9
V_{SS}	20	21	A_8

(2) 定时和控制信号

输入 X_1 X_2 地址暂存使能信号

输出 CLK ALE S_0 S_1
 $\overline{IO/M}$ \overline{RD} \overline{WR}

用于标识读/写操作是否发生

(3) 存储器和 I/O 初始化

输入 HOLD Ready

输出 HLDA DMA总线请求信号
DMA总线响应信号

用于
CPU
与较
慢的
存储
器或
外设
同步

(4) 与中断有关的信号

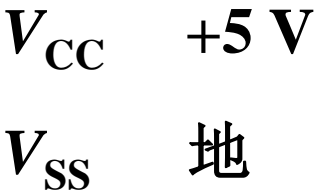


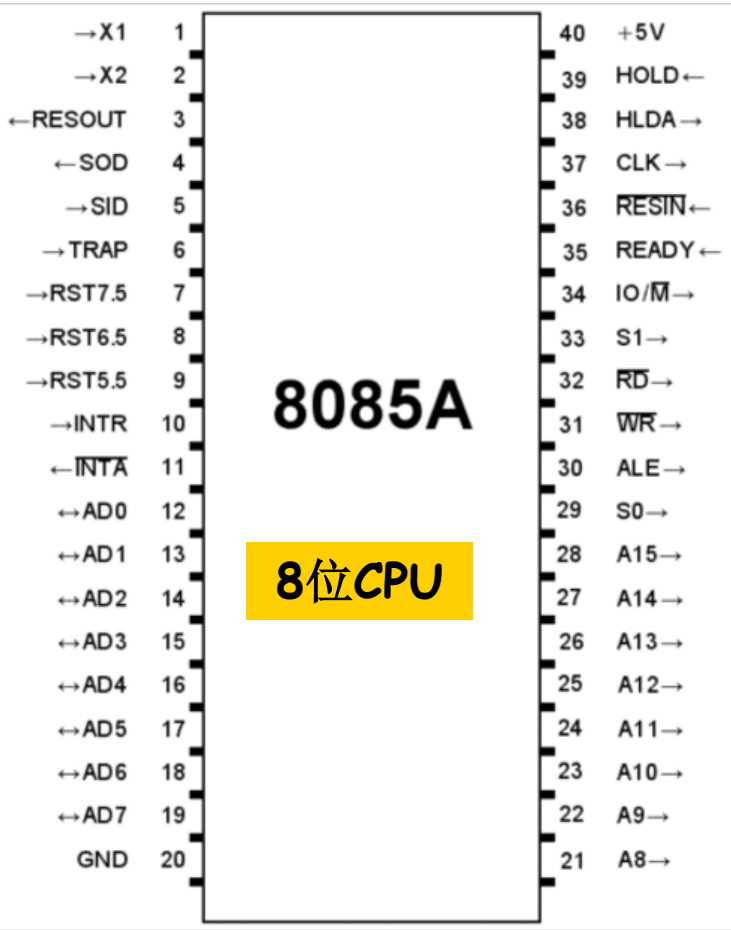
X_1	1	40	V_{CC}
X_2	2	39	HOLD
Reset out	3	38	HLDA
SOD	4	37	$\overline{\text{CLK}}(\text{out})$
SID	5	36	$\overline{\text{Rstet in}}$
Trap	6	35	Ready
RST7.5	7	34	$\overline{\text{IO/M}}$
RST6.5	8	33	$\overline{\text{S}}_1$
RST5.5	9	32	$\overline{\text{RD}}$
$\overline{\text{INTR}}$	10	31	$\overline{\text{WR}}$
INTA	11	30	ALE
AD_0	12	29	S_0
AD_1	13	28	A_{15}
AD_2	14	27	A_{14}
AD_3	15	26	A_{13}
AD_4	16	25	A_{12}
AD_5	17	24	A_{11}
AD_6	18	23	A_{10}
AD_7	19	22	A_9
V_{SS}	20	21	A_8

(5) CPU 初始化



(6) 电源和地





Intel 8085 registers

$1_5 \ 1_4 \ 1_3 \ 1_2 \ 1_1 \ 1_0 \ 0_9 \ 0_8 \ 0_7 \ 0_6 \ 0_5 \ 0_4 \ 0_3 \ 0_2 \ 0_1 \ 0_0$ (bit position)

Main registers

A	Flags	Program Status Word
B	C	B
D	E	D
H	L	H (indirect address)

Index registers

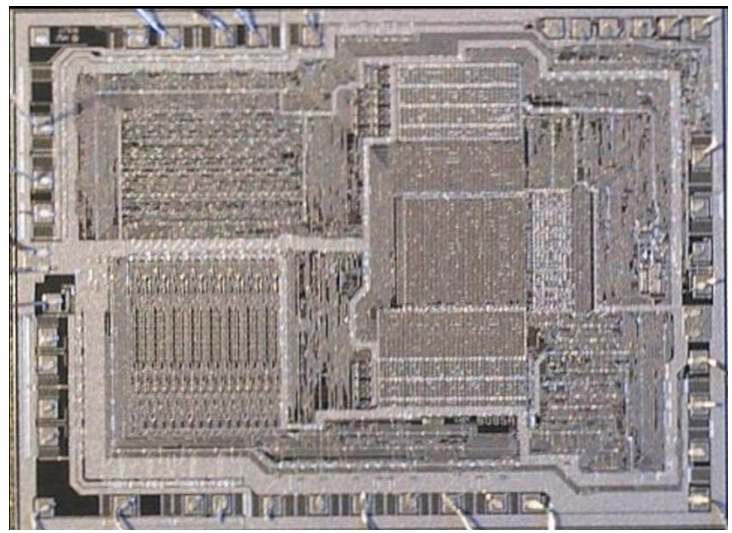
SP	Stack Pointer
----	---------------

Program counter

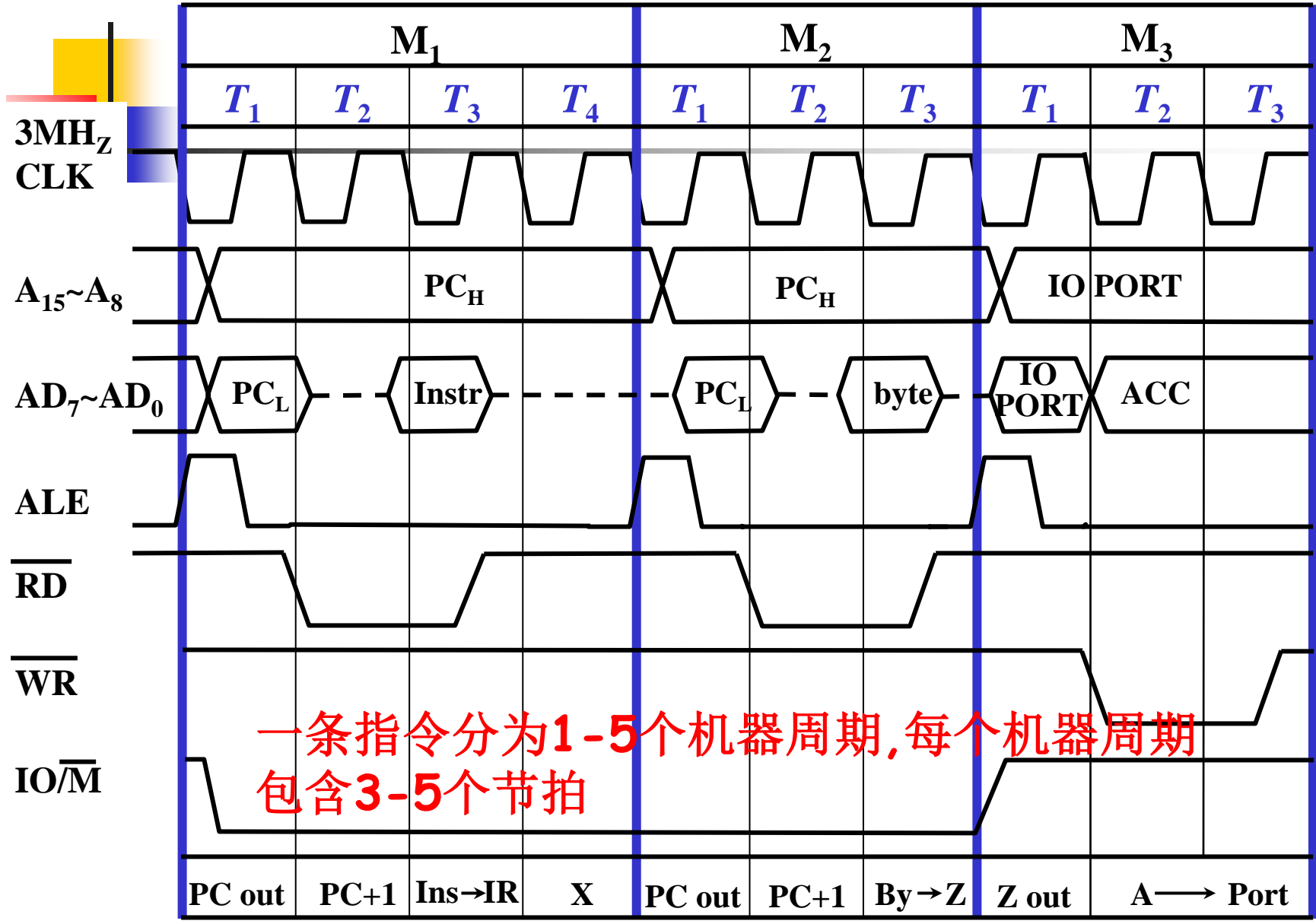
PC	Program Counter
----	-----------------

Status register

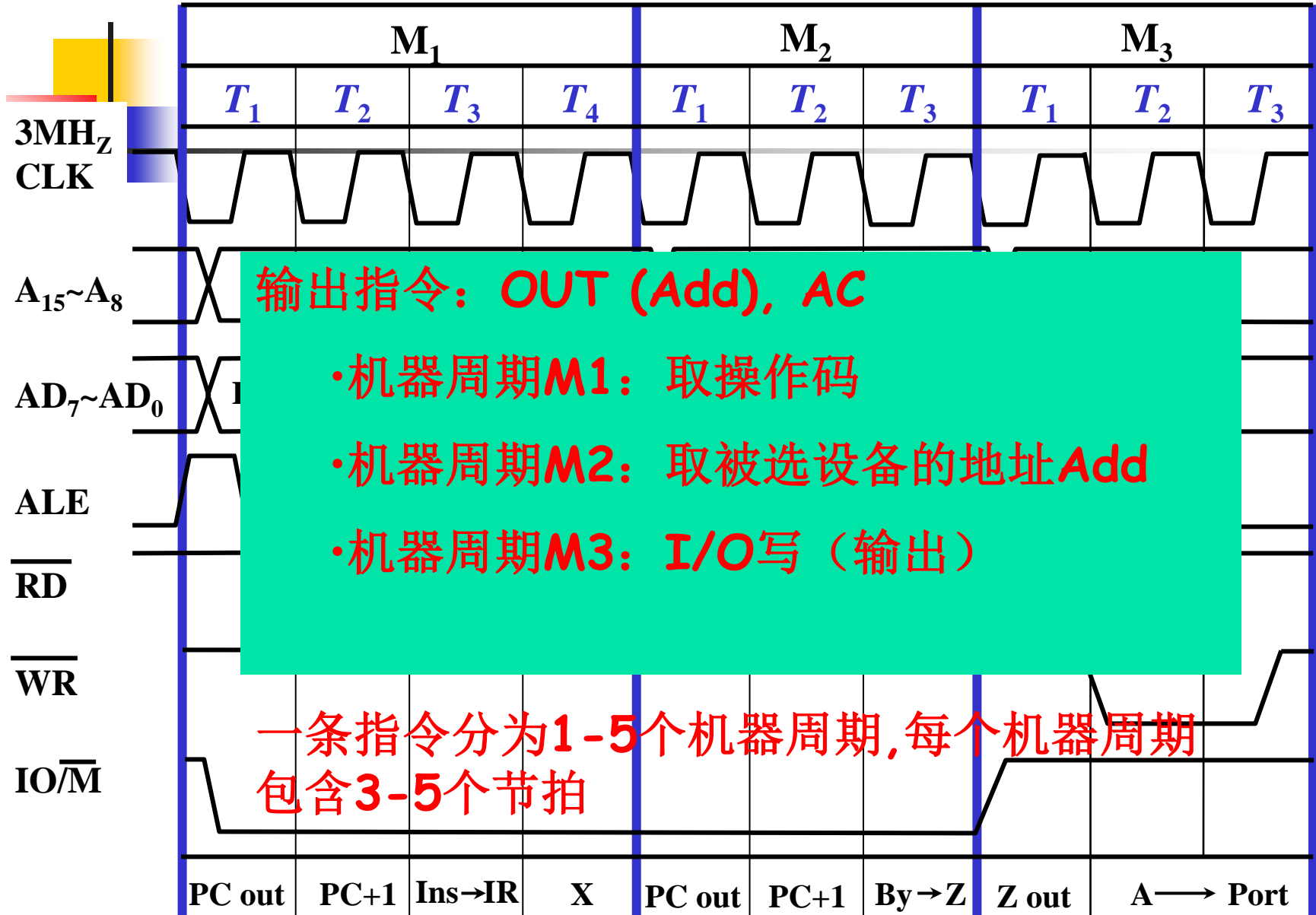
S	Z	-	AC	-	P	-	CY	Flags
---	---	---	----	---	---	---	----	-------



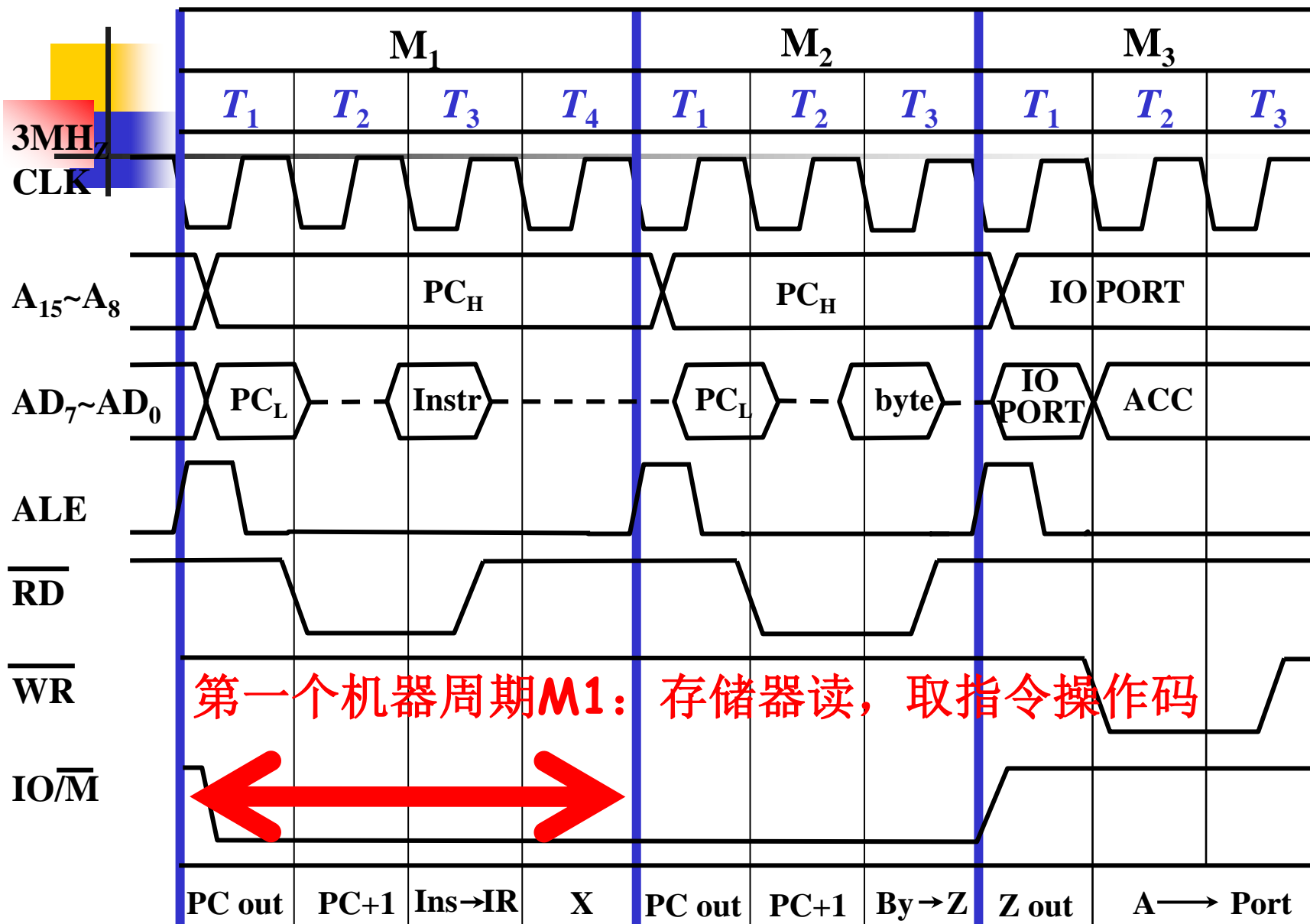
3. 机器周期和节拍（状态）与控制信号的关系



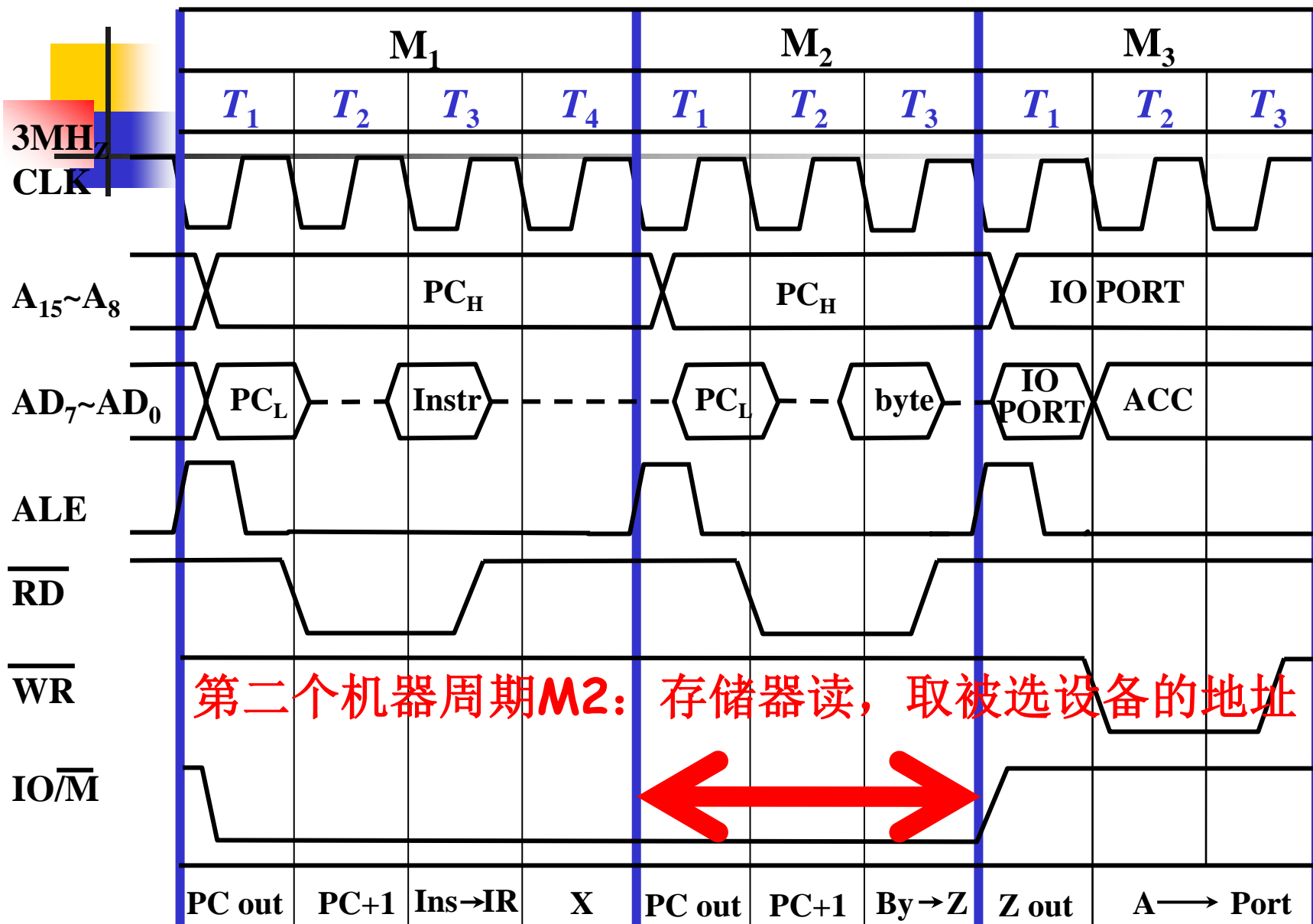
3. 机器周期和节拍（状态）与控制信号的关系



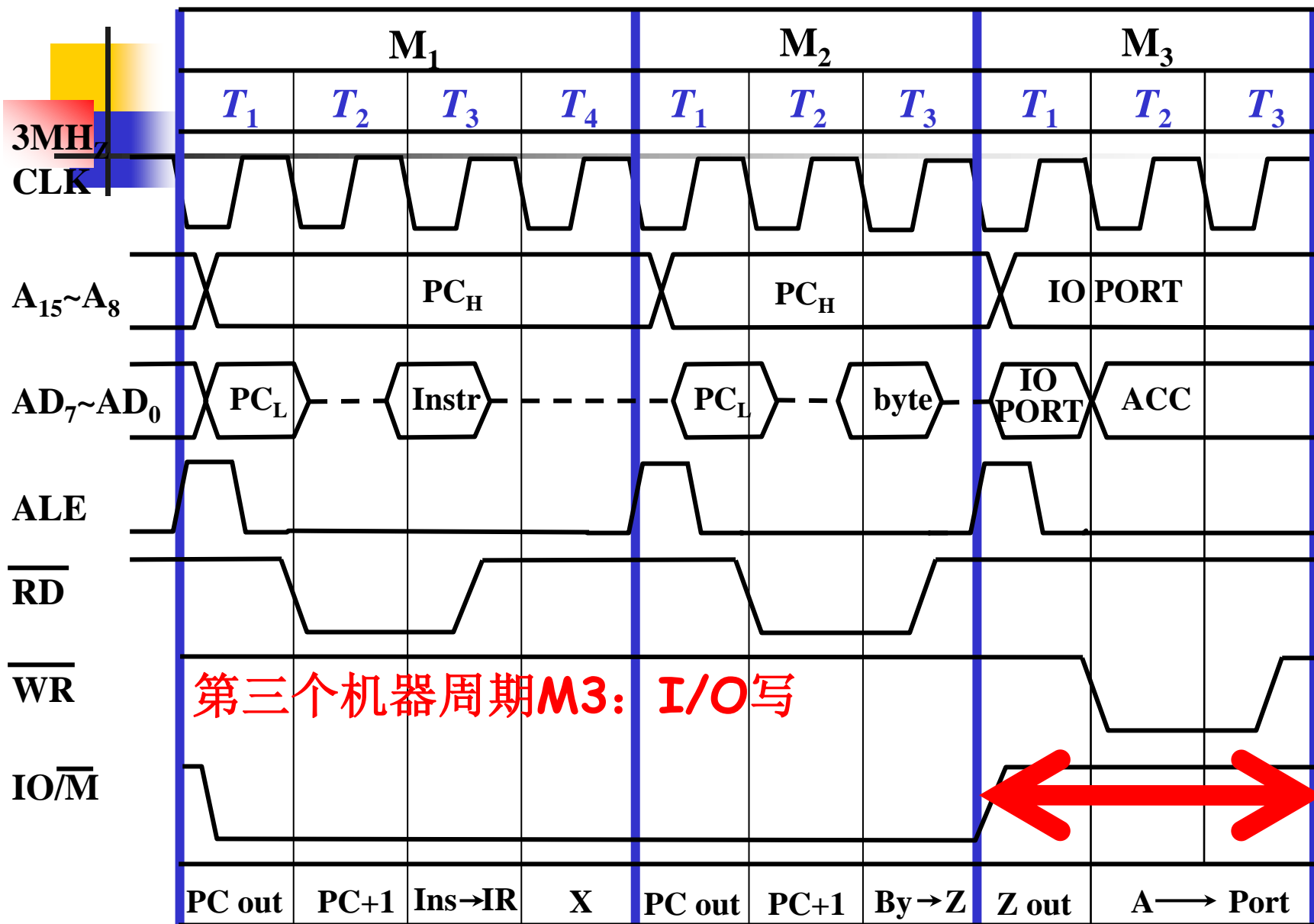
3. 机器周期和节拍（状态）与控制信号的关系



3. 机器周期和节拍（状态）与控制信号的关系



3. 机器周期和节拍（状态）与控制信号的关系



小结

以一条输出指令（I/O 写）为例

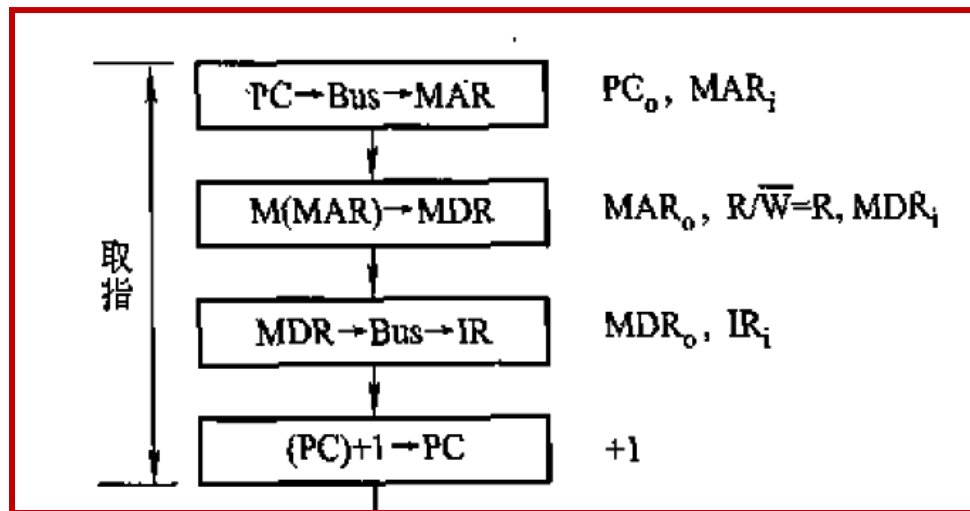
- 机器周期 M_1 取指令操作码
- 机器周期 M_2 取设备地址（I/O地址）
- 机器周期 M_3 执行 ACC 的内容写入设备

每个 控制 信号在 指定机器周期 的
指定节拍 T 时刻 发出

本章小结

- 取指周期——从存储器中取出指令的机器码（放到IR中）

- ① $PC \rightarrow MAR \rightarrow \text{地址线}$
- ② $1 \rightarrow R$
- ③ $M(MAR) \rightarrow MDR$
- ④ $MDR \rightarrow IR$
- ⑤ $OP(IR) \rightarrow CU$
- ⑥ $(PC) + 1 \rightarrow PC$



本章小结

- 间址周期——根据形式地址从存储器中得到有效地址

① 指令形式地址 \rightarrow MAR

即: $Ad(IR) \rightarrow MAR$

② $1 \rightarrow R$

③ $M(MAR) \rightarrow MDR$

④ $MDR \rightarrow Ad(IR)$

- **$Ad(IR) \rightarrow MAR$**

- **$1 \rightarrow R$**

- **$M(MAR) \rightarrow MDR$**

本章小结

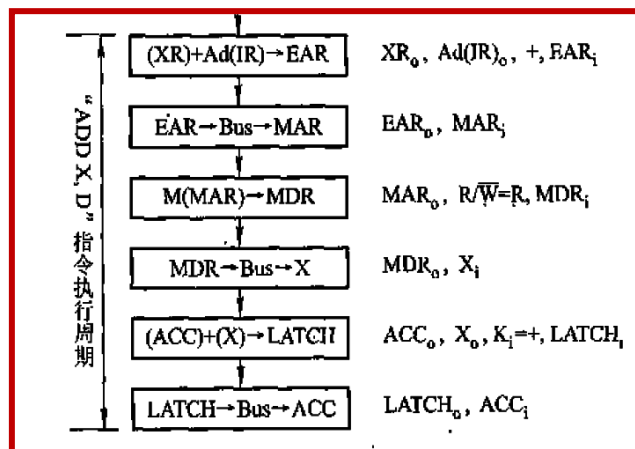
- 执行周期
 - 非访存指令
 - 访存指令
 - 转移指令

- ①取数指令“LDA M”执行阶段所需的全部微操作

Ad(IR)→MAR	指令的地址码字段→MAR
I→R	命令存储器读
M(MAR)→MDR	操作数从存储器中读至MDR
MDR→ACC	操作数→IR
- ②存数指令“STA M”执行阶段所需的全部微操作

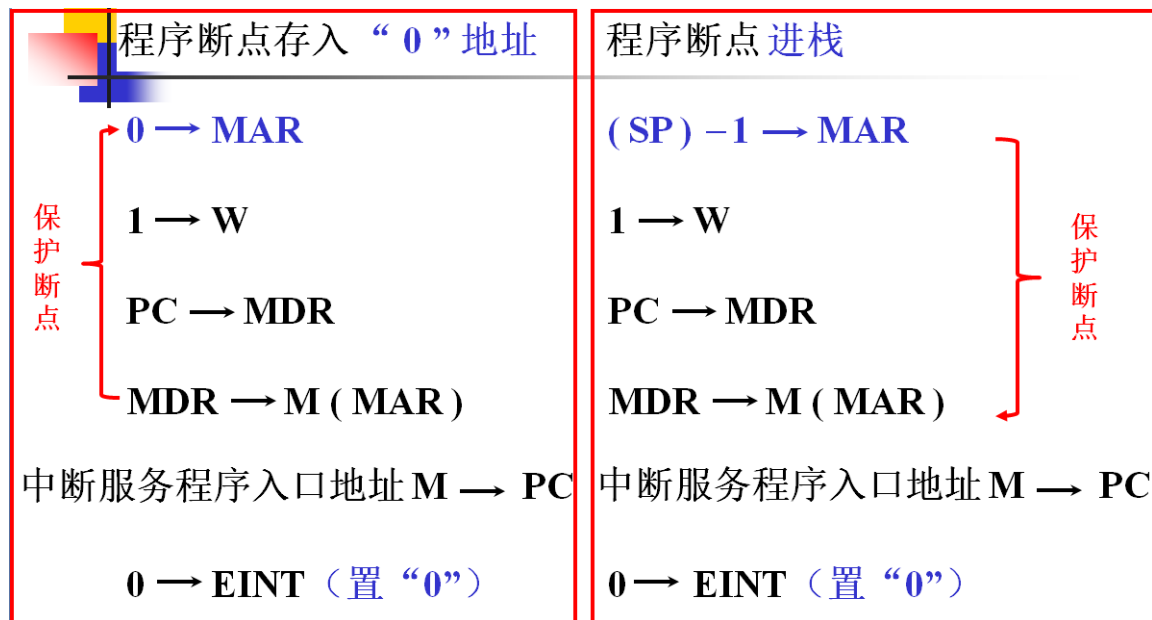
Ad(IR)→MAR	指令的地址码字段→MAR
I→W	命令存储器写
ACC→MDR	欲写入的数据→MDR
MDR→M(MAR)	数据写至存储器中
- ③加法指令“ADD M”执行阶段所需的全部微操作

Ad(IR)→MAR	指令的地址码字段→MAR
I→R	命令存储器读
M(MAR)→MDR	操作数从存储器中读至MDR
(ACC)+(MDR)→ACC	两数相加结果送ACC



本章小结

- **中断周期**——由中断隐指令自动完成：**(1)** 保护断点；**(2)** 寻找中断服务程序入口地址；**(3)** 硬件关中断



本章小结

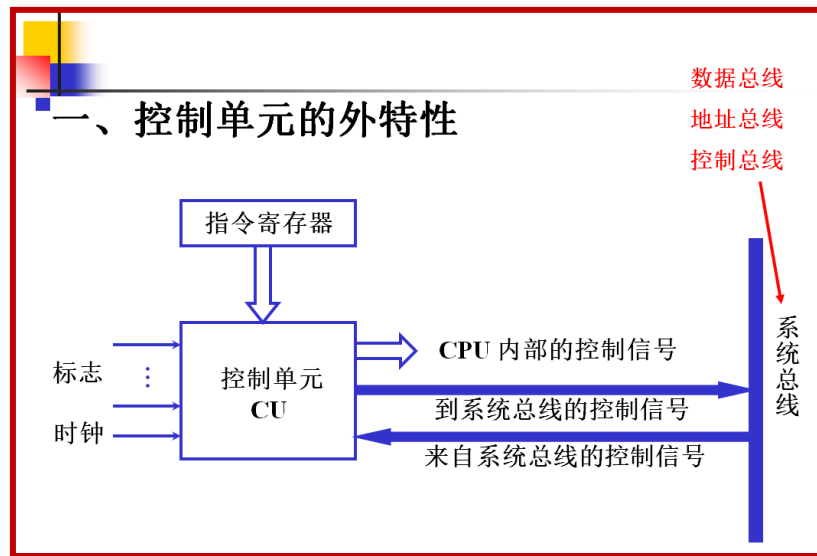
■ 控制单元的输入和输出

■ 输入：

- 指令的操作码
- 标志
- 来自系统总线的控制信号
- 时钟信号

■ 输出：

- CPU内容的控制信号
- 到系统总线的控制信号



本章小结

■ 指令周期

■ 机器周期

■ 时钟周期

→ “单机器指令”方式运行，每次执行**1**条指令

→ “单周期”方式运行，每次执行**1**条微指令

→ “单节拍”方式运行，每次执行**1**个时钟周期（**T**周期）

继续
停止



本章小结

■ 控制方式

- 同步控制方式
 - 采用定长的机器周期
 - 采用不定长的机器周期
 - 采用中央控制和局部控制相结合的方法
- 异步控制方式
- 联合控制方式：同步与异步的结合
- 人工控制方式：**RESET**、单步执行、连续执行



第15次作业——习题（P393-394）

- 9.1
- 9.2
- 9.3
- 9.4
- 9.6
- 9.8
- 9.14



关于作业的提交

- **1周内**必须提交（上传到学院的**FTP**服务器上），否则认为是迟交作业；如果期末仍然没有提交，则认为是未提交作业
 - 作业完成情况成绩=第**1**次作业提交情况*第**1**次作业评分+第**2**次作业提交情况*第**2**次作业评分+.....+第**N**次作业提交情况*第**N**次作业评分
 - 作业评分：**A**（好）、**B**（中）、**C**（差）三挡
 - 作业提交情况：按时提交（**1.0**）、迟交（**0.5**）、未提交（**0.0**）
- 请采用电子版的格式（**Word**文档）上传到**FTP**服务器上，文件名取“学号+姓名+第**X**次作业.doc”
 - 例如：**11920192203642+袁佳哲+第15次作业.doc**
- 第**15**次作业提交的截止日期为：**2021年6月9日晚上24点**



The End

Thanks