

计算机组成原理

中央处理器（2）

王浩宇,教授

haoyuwang@hust.edu.cn

<https://howiepku.github.io/>

本节目录

- 指令周期
- 硬布线控制器设计

指令周期

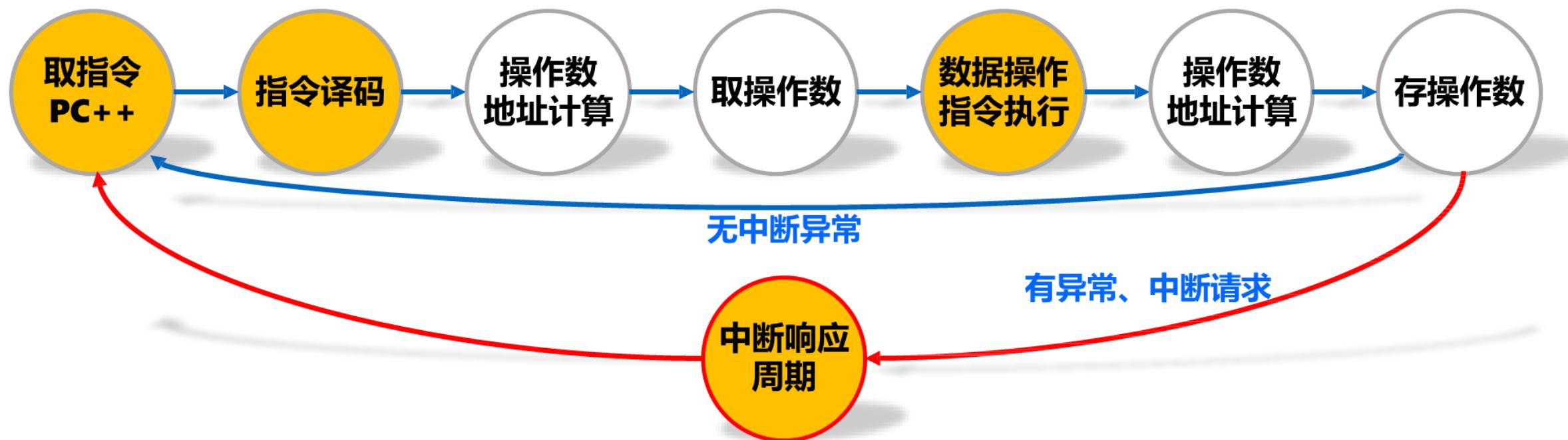
- 指令执行的一般流程
 - 不同指令功能不同，数据通路不同，执行时间不同，如何安排时序？
 - 访存指令，寄存器运算指令，加法指令与除法指令

指令周期

■ 指令执行的一般流程

■ 不同指令功能不同，数据通路不同，执行时间不同，**如何安排时序？**

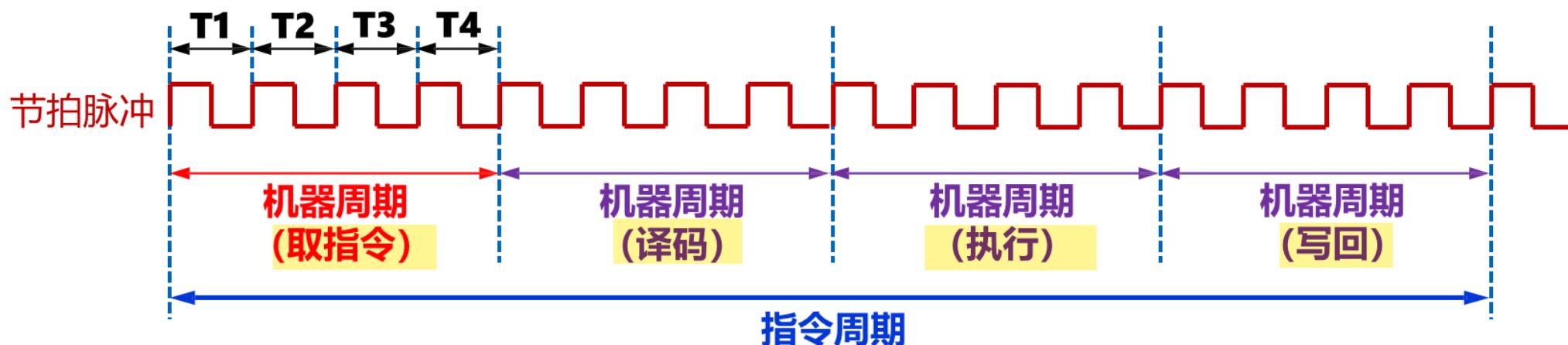
■ 访存指令，寄存器运算指令，加法指令与除法指令



指令周期

■ 指令周期的基本概念

- 时钟周期 = 节拍脉冲 = 震荡周期 能完成一次微操作
- 机器周期 = CPU周期 从主存读出一条指令的最短时间 可完成 复杂操作
- 指令周期：从主存取一条指令并执行指令的时间
◆ 由若干机器周期组成，机器周期包含若干时钟周期



指令周期

■ 指令控制同步

■ 不同指令功能不同，复杂度不同，如何进行时间控制？

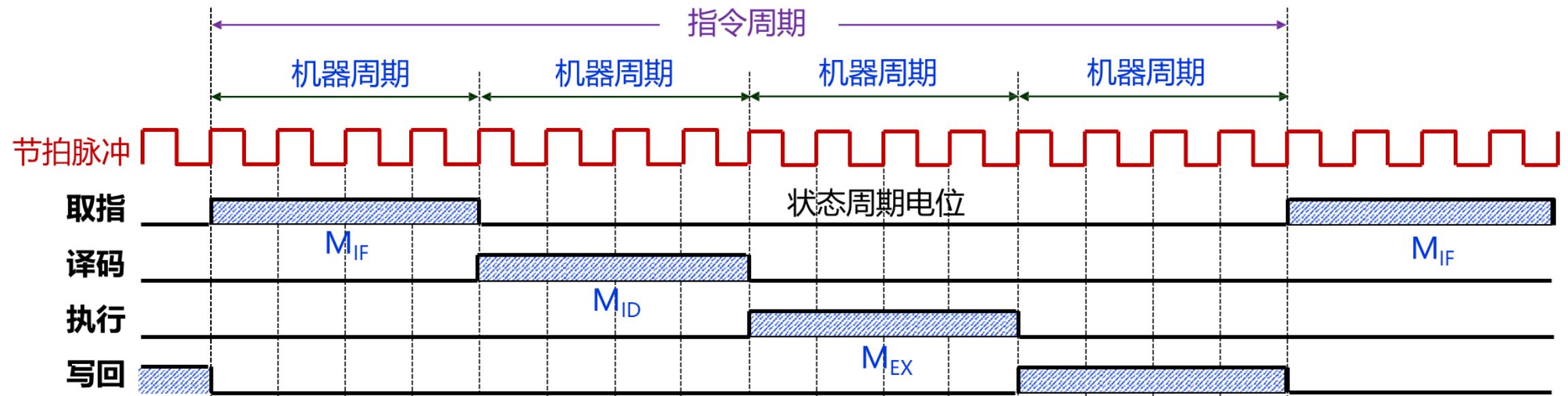
◆ 定长指令周期：早期三级时序系统

◆ 机器周期数固定，节拍数固定，按机器周期同步，mips单周期

◆ 变长指令周期：现代时序系统

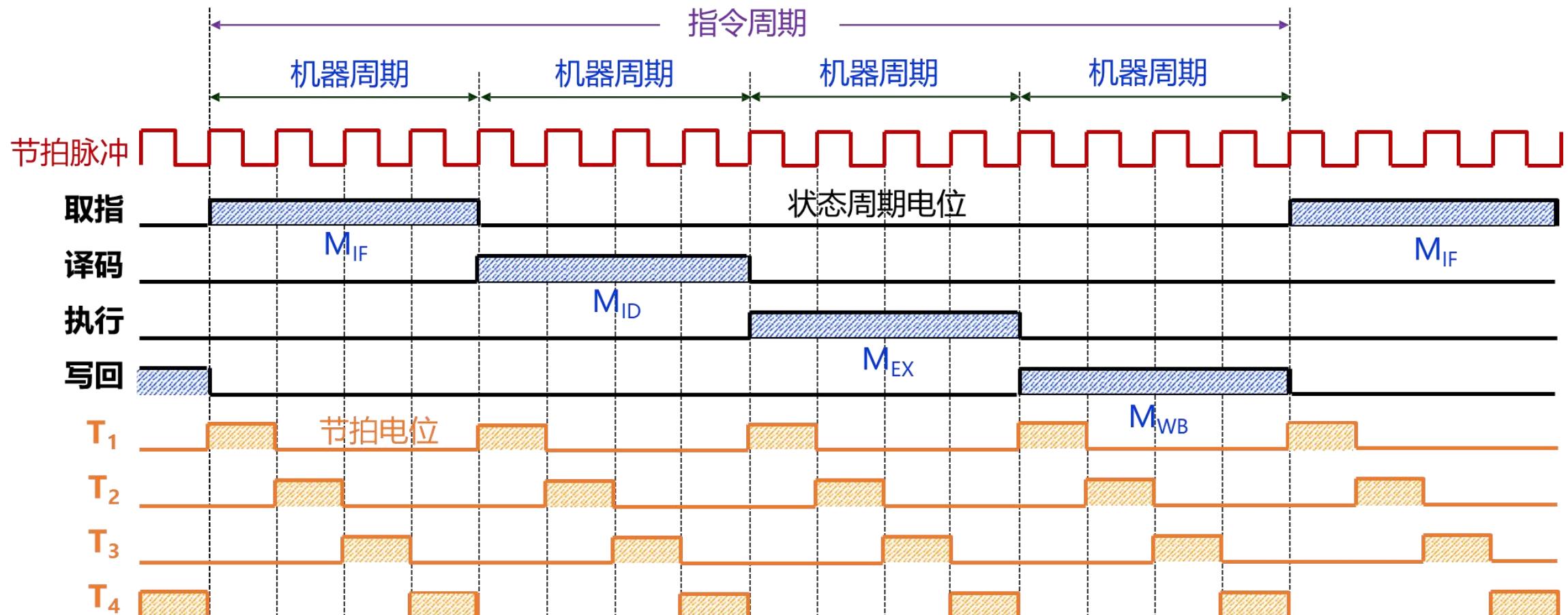
◆ 机器周期数可变，节拍数可变，按时钟周期同步，mips多周期

指令周期



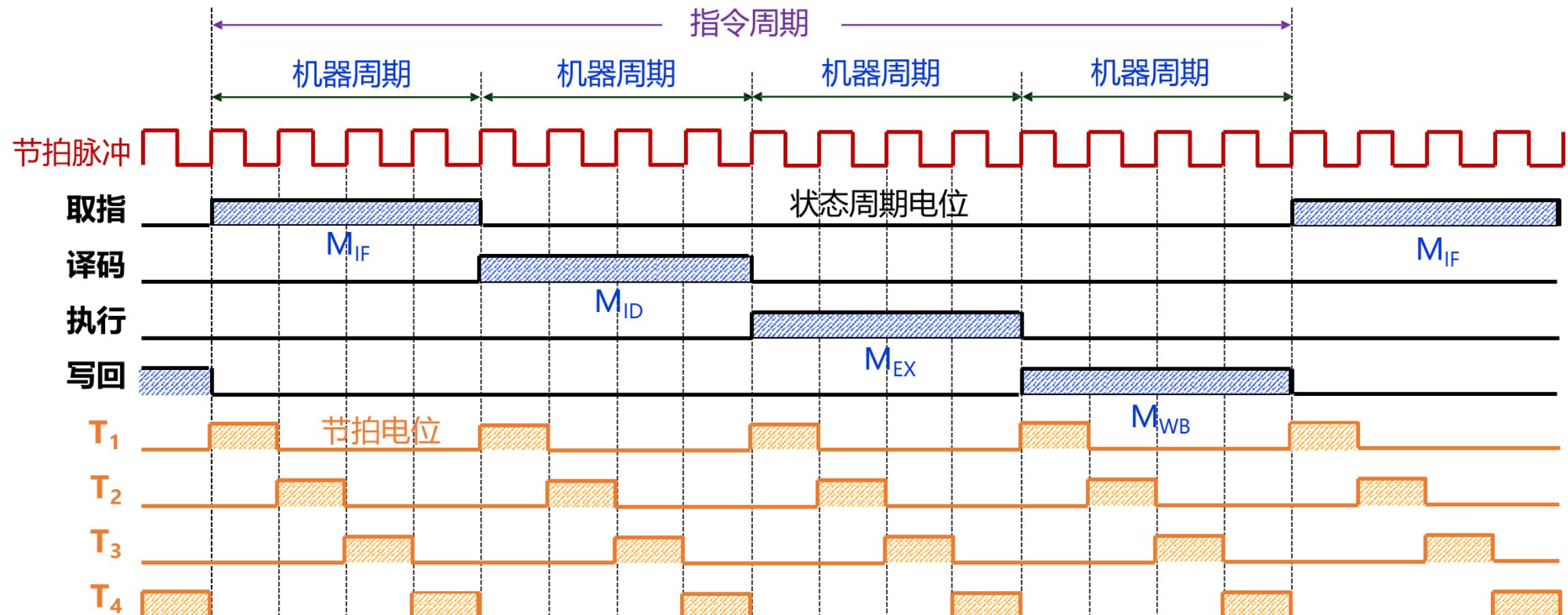
指令周期

早期三级时序系统



指令周期

早期三级时序系统

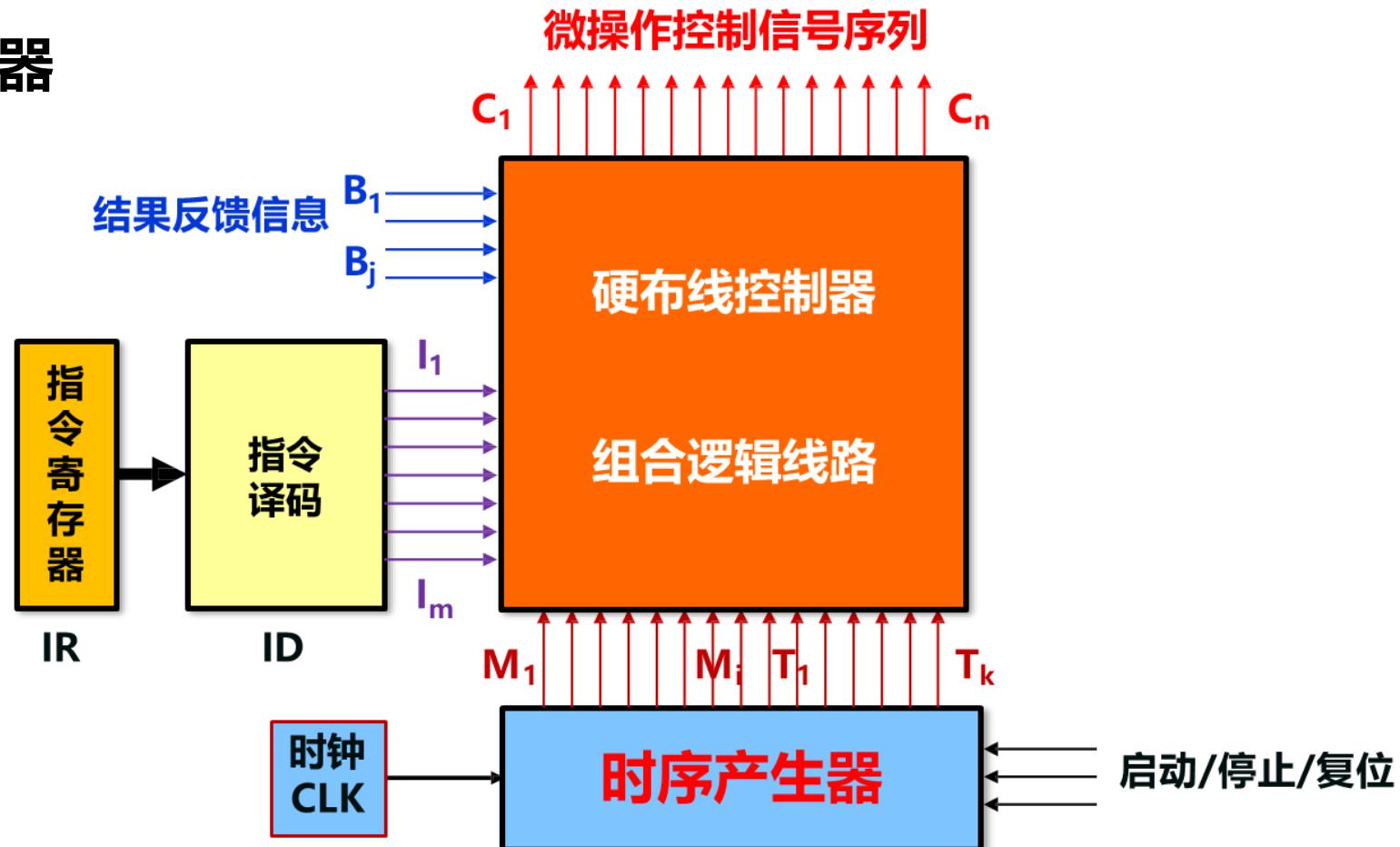


时钟脉冲→机器周期电位, 节拍电位信号, 硬布线控制器→组合逻辑

$$\text{MemRead} = M_{IF} \cdot (T_2 + T_3) + \text{Load} \cdot M_{EX} \cdot (T_2 + T_3)$$

指令周期

■ 时序产生器与控制器

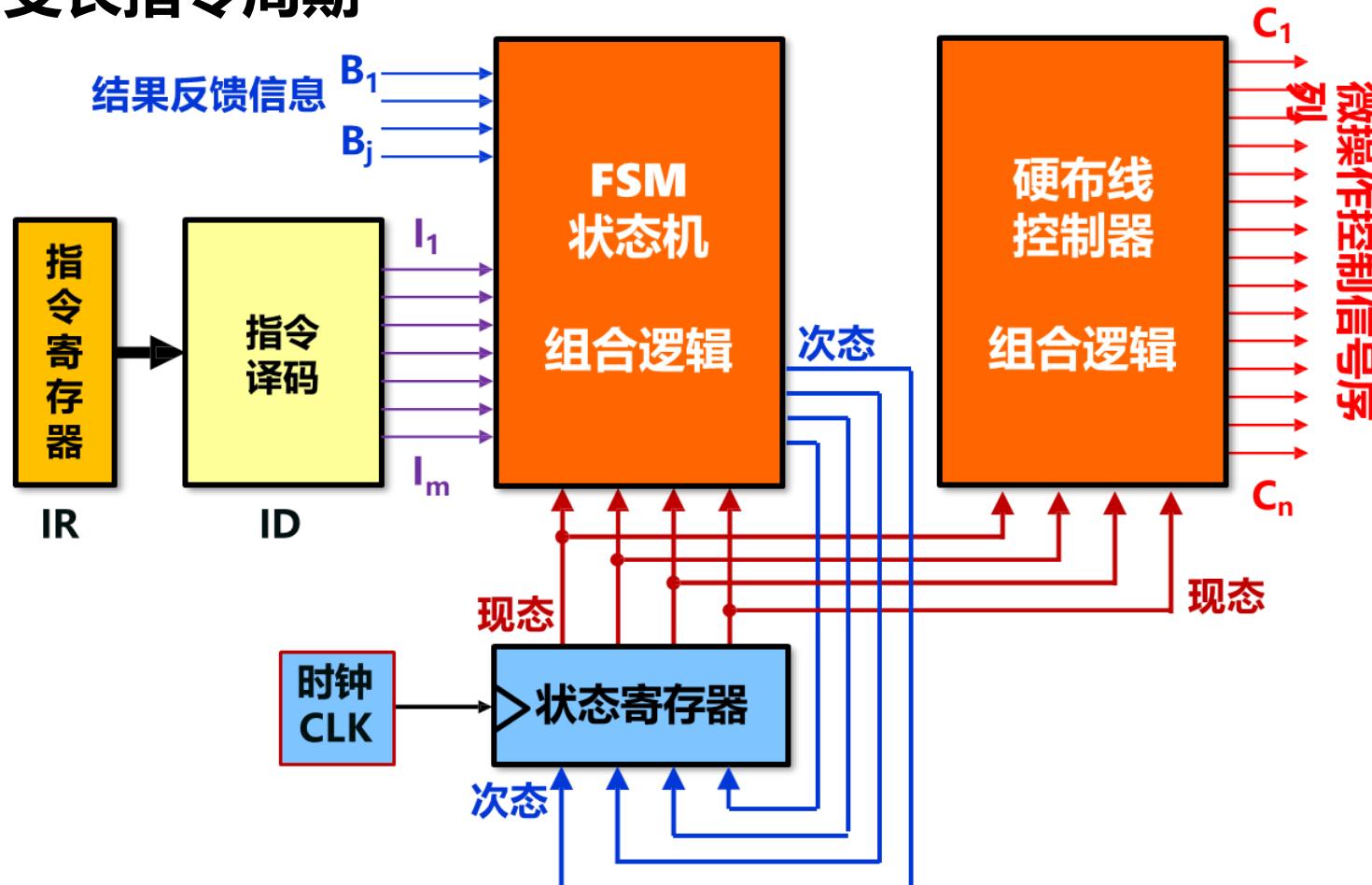


时序产生器循环产生周期电位、节拍电位，供控制器对信号进行时间调制

$$\text{MemRead} = M_{IF} \cdot (T_2 + T_3) + \text{Load} \cdot M_{EX} \cdot (T_2 + T_3)$$

指令周期

■ 现代时序系统：变长指令周期



操作控制信号仅仅与状态寄存器现态有关

寄存器传送语言

- 为了便于统一表示指令执行流程，采用寄存器传送语言（RTL）来表示指令执行过程中的操作
- 每条指令的执行过程都可以分解为一组操作序列，进而可分解为一组微操作序列
 - 操作是指功能部件级的动作，微操作是指令序列中最基本的、不可分割的动作
- $M[addr]$ ：主存addr单元或内容
- $R[i]$ ：通用寄存器组中第*i*号寄存器或其内容
- $B \leftarrow A$ 或者 $A \rightarrow B$ 表示数据传送
- $X_{y:z}$ 寄存器X的第y位到第z位的数据字段
- $\text{SignExt}(x)$ 表示将x符号扩展到32位
- $\{X, Y\}$ 将X、Y的比特位连接在一起

总线结构CPU指令周期

1

单总线结构CPU

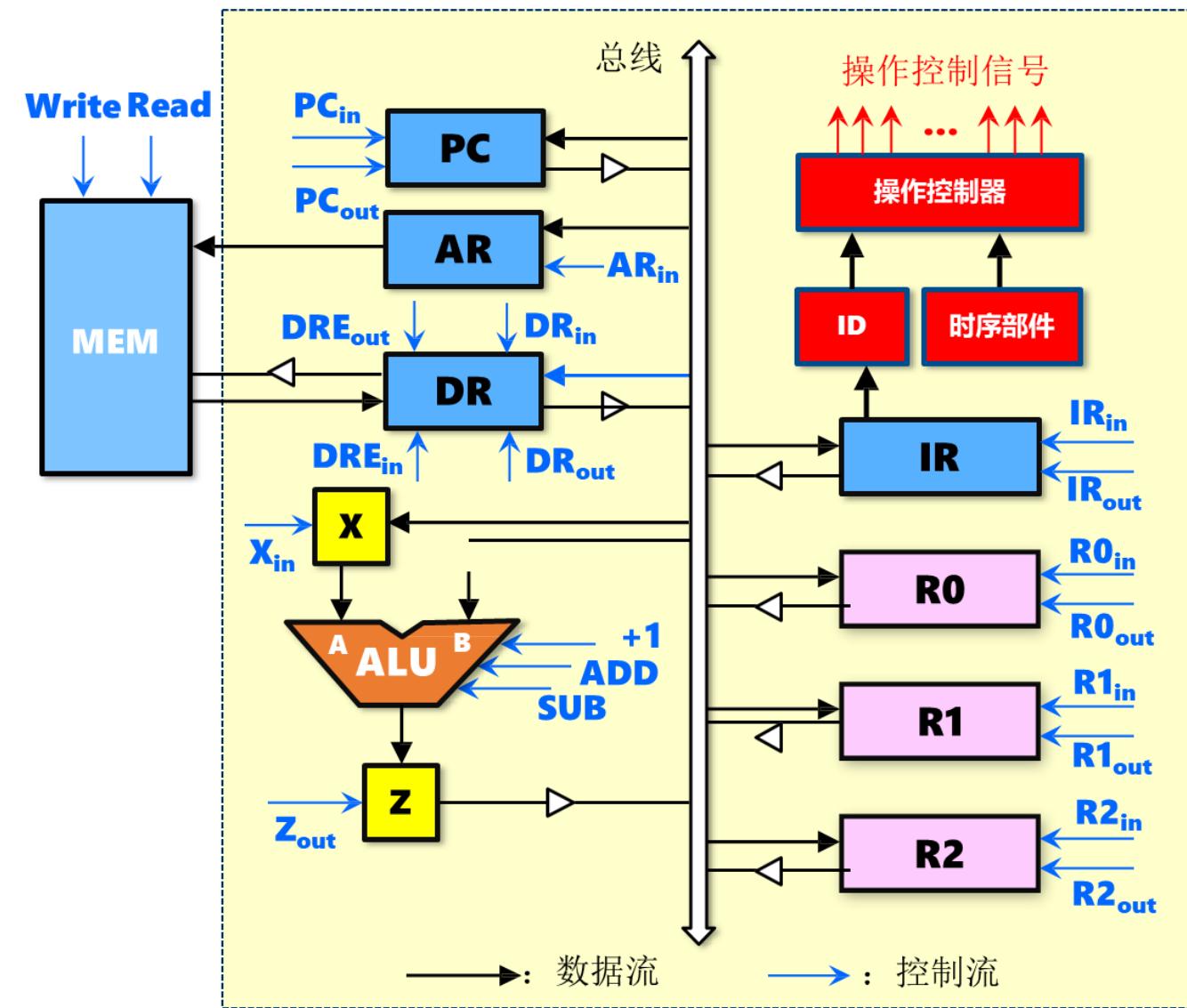
1. LOAD R0,6#

2. MOVE R1,10

3. ADD R0,R1

4. STORE R0,(R2)

5. JMP 1000



总线结构CPU指令周期

1

单总线结构CPU

#	指令	指令功能
1	LOAD R0,6#	Mem[6] → R0
2	MOVE R1,10	10 → R1
3	ADD R0,R1	(R0) + (R1) → R0
4	STORE R0,(R2)	(R0) → Mem[(R2)]
5	JMP 1000	1000 → PC

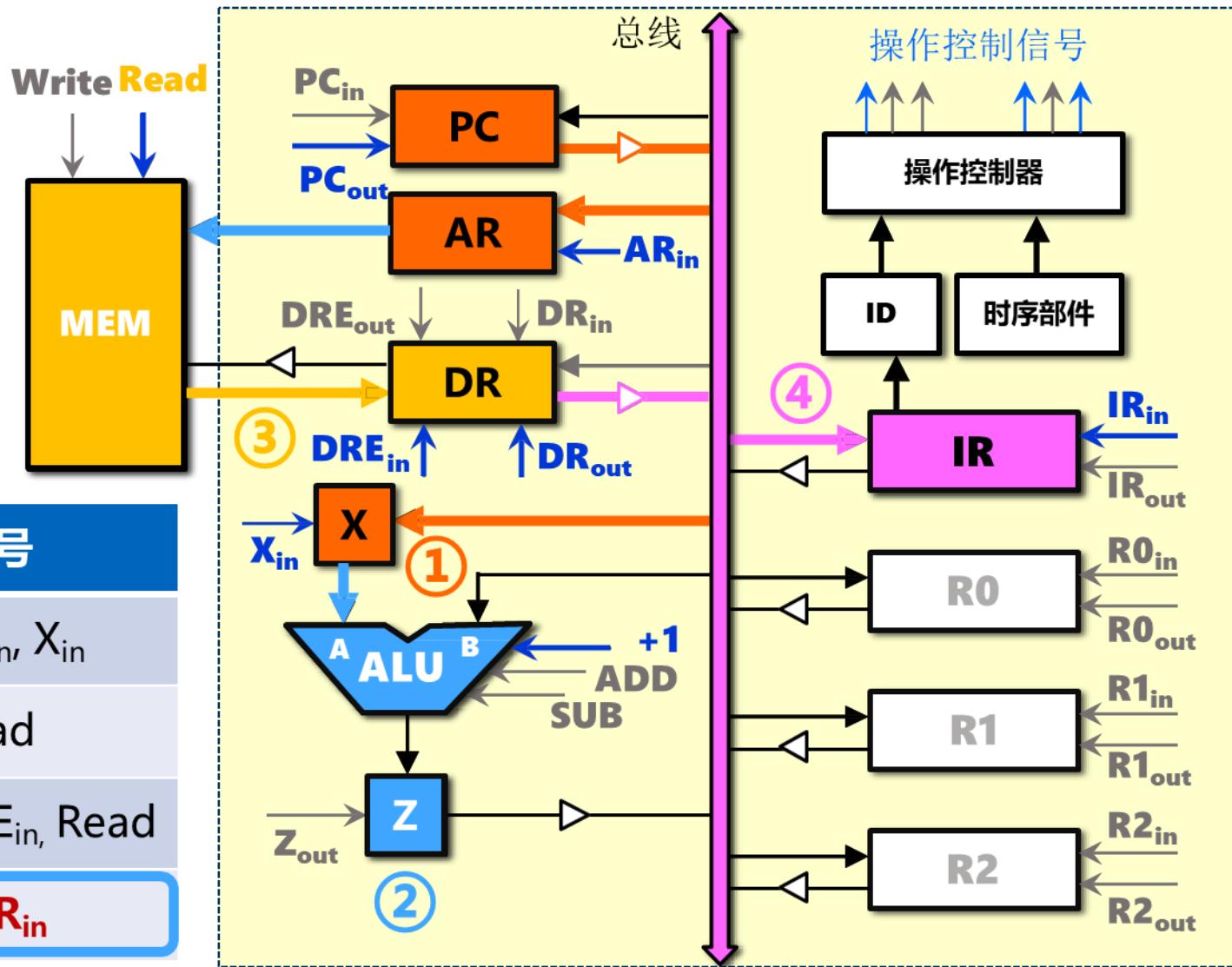
总线结构CPU指令周期

1

取指令数据通路

Mem[PC++] → IR

节拍	数据通路	控制信号
T1	(PC)→AR, (PC)→X	PC _{out} , AR _{in} , X _{in}
T2	(X)+1→Z	+1, Read
T3	(Z)→PC, Mem[AR]→DR	Z _{out} , PC _{in} , DRE _{in} , Read
T4	(DR)→IR	DR_{out}, IR_{in}



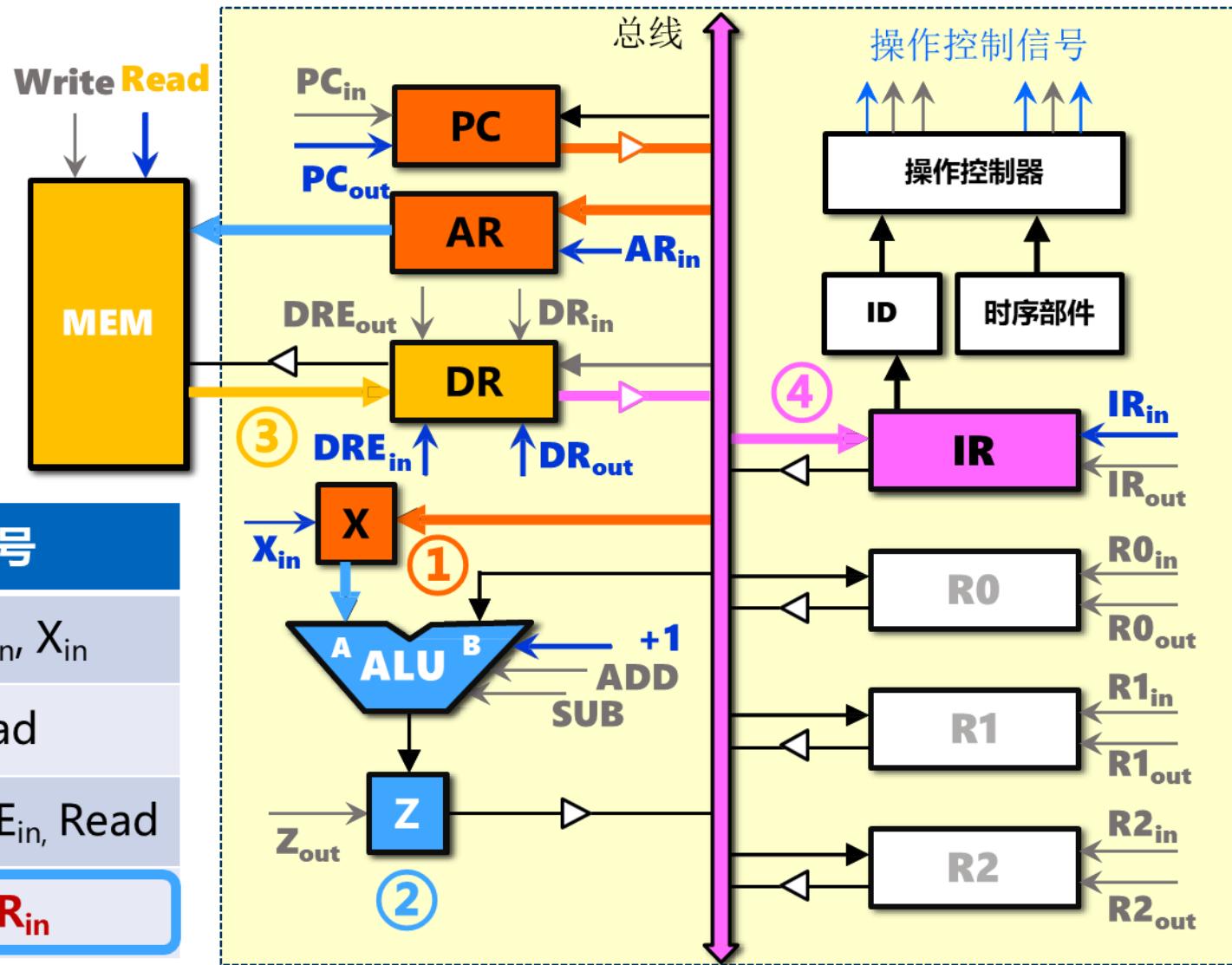
总线结构CPU指令周期

1

取指令数据通路

Mem[PC++] → IR

节拍	数据通路	控制信号
T1	(PC)→AR, (PC)→X	PC _{out} , AR _{in} , X _{in}
T2	(X)+1→Z	+1, Read
T3	(Z)→PC, Mem[AR]→DR	Z _{out} , PC _{in} , DRE _{in} , Read
T4	(DR)→IR	DR_{out}, IR_{in}



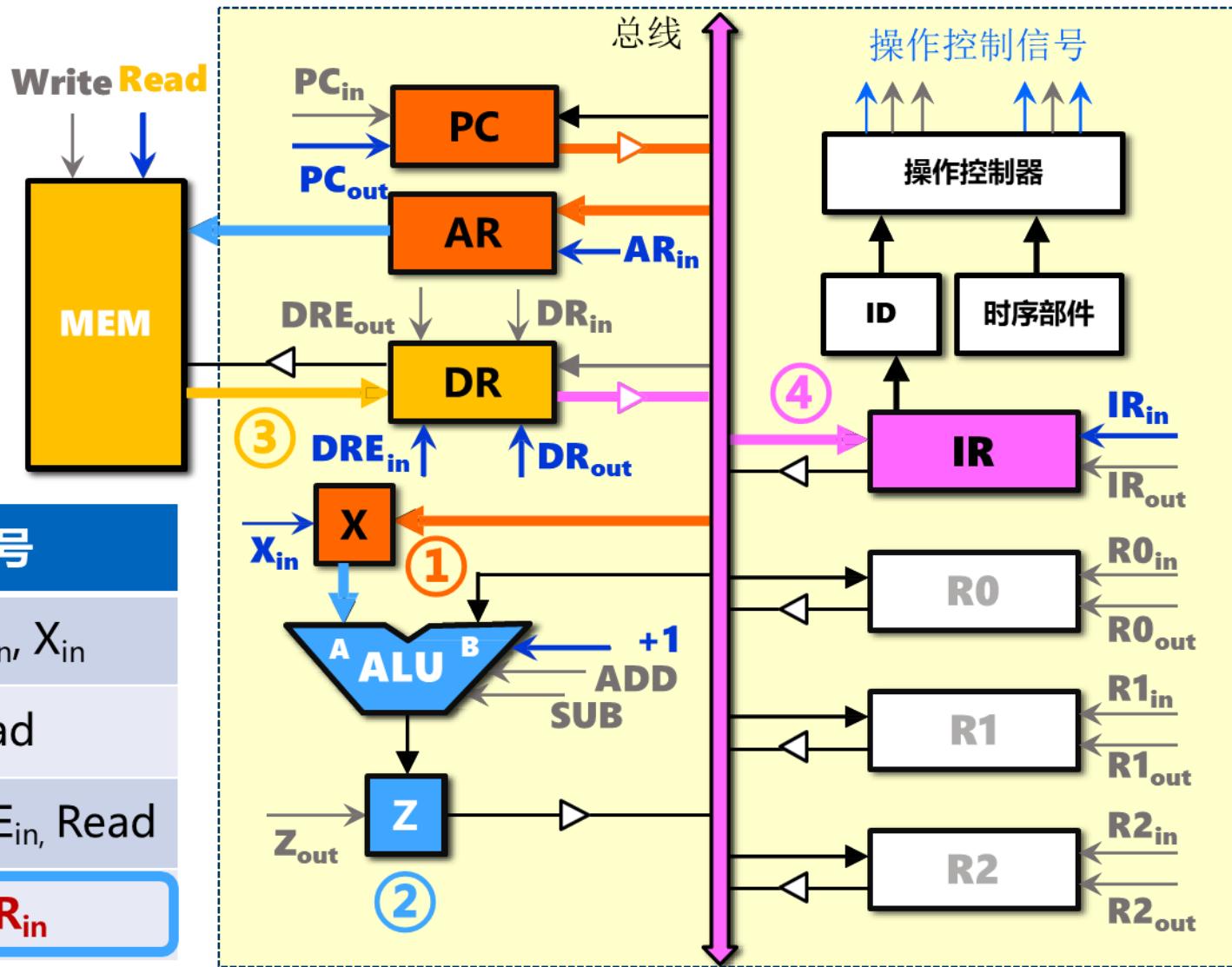
总线结构CPU指令周期

1

取指令数据通路

Mem[PC++] → IR

节拍	数据通路	控制信号
T1	(PC)→AR, (PC)→X	PC _{out} , AR _{in} , X _{in}
T2	(X)+1→Z	+1, Read
T3	(Z)→PC, Mem[AR]→DR	Z _{out} , PC _{in} , DRE _{in} , Read
T4	(DR)→IR	DR_{out}, IR_{in}



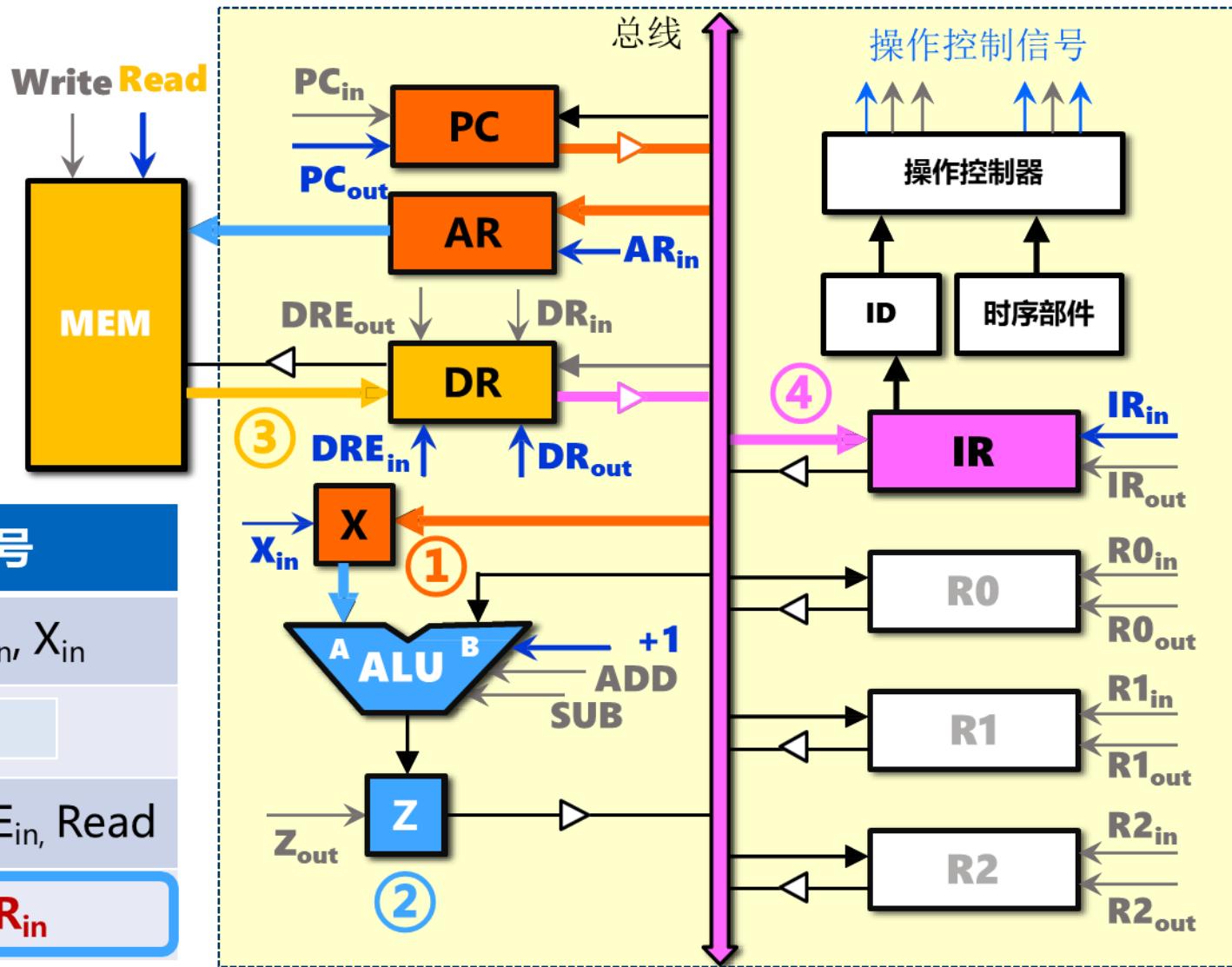
总线结构CPU指令周期

1

取指令数据通路

Mem[PC++] → IR

节拍	数据通路	控制信号
T1	(PC)→AR, (PC)→X	PC _{out} , AR _{in} , X _{in}
T2	(X)+1→Z	+1, []
T3	(Z)→PC, Mem[AR]→DR	Z _{out} , PC _{in} , DRE _{in} , Read
T4	(DR)→IR	DR_{out}, IR_{in}



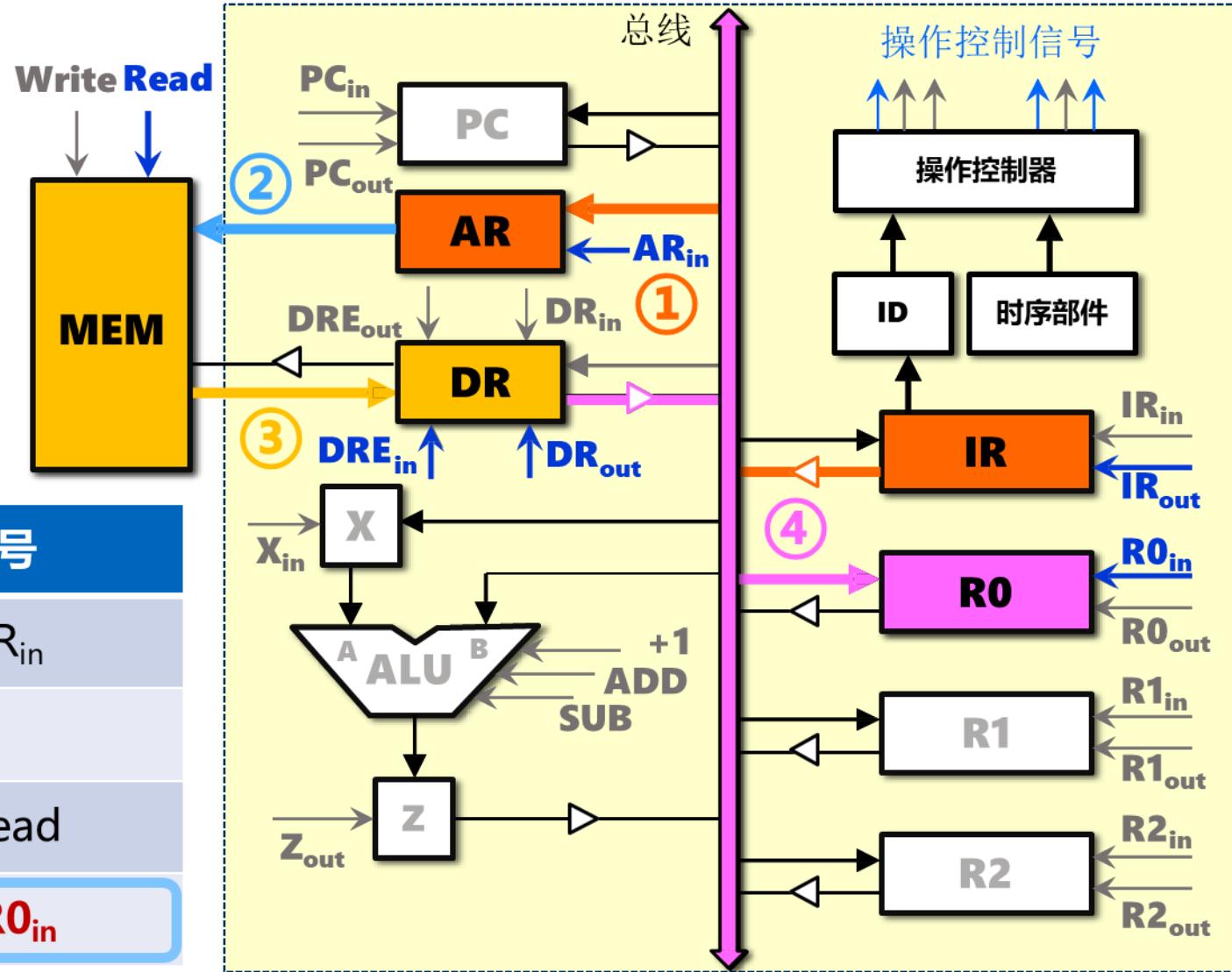
总线结构CPU指令周期

2 LOAD指令执行数据通路

LOAD R0,6#

Mem[IR_A] → Reg

节拍	数据通路	控制信号
T1	(IR _A)→AR	IR _{out} , AR _{in}
T2		Read
T3	Mem[AR]→DR	DRE _{in} , Read
T4	(DR)→R0	DR_{out}, R0_{in}

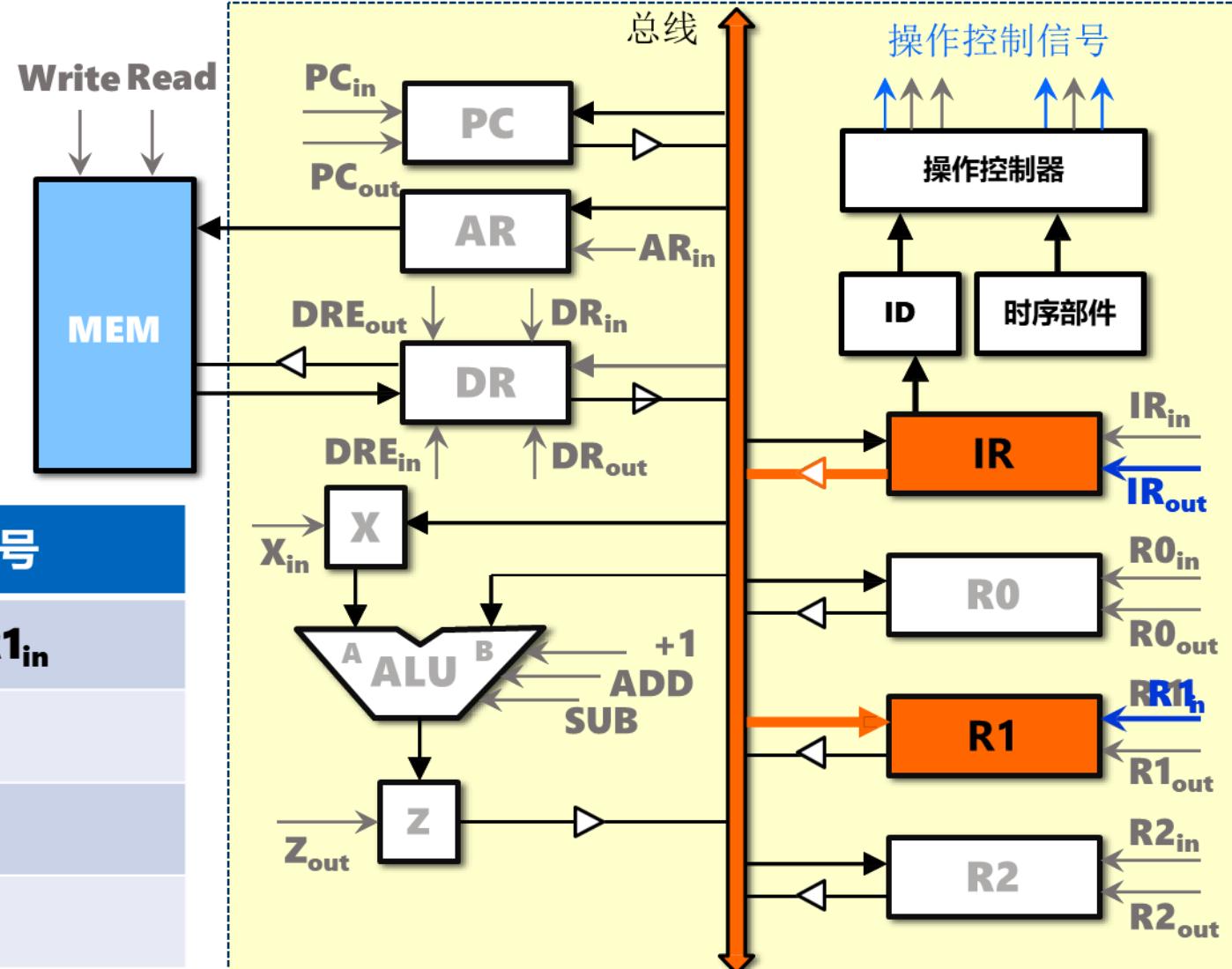


总线结构CPU指令周期

MOVE指令执行数据通路

MOVE R1,10
 $(IR_A) \rightarrow Reg$

节拍	数据通路	控制信号
T1	$(IR_A) \rightarrow R[0]$	$IR_{out}, R1_{in}$
T2		
T3		
T4		



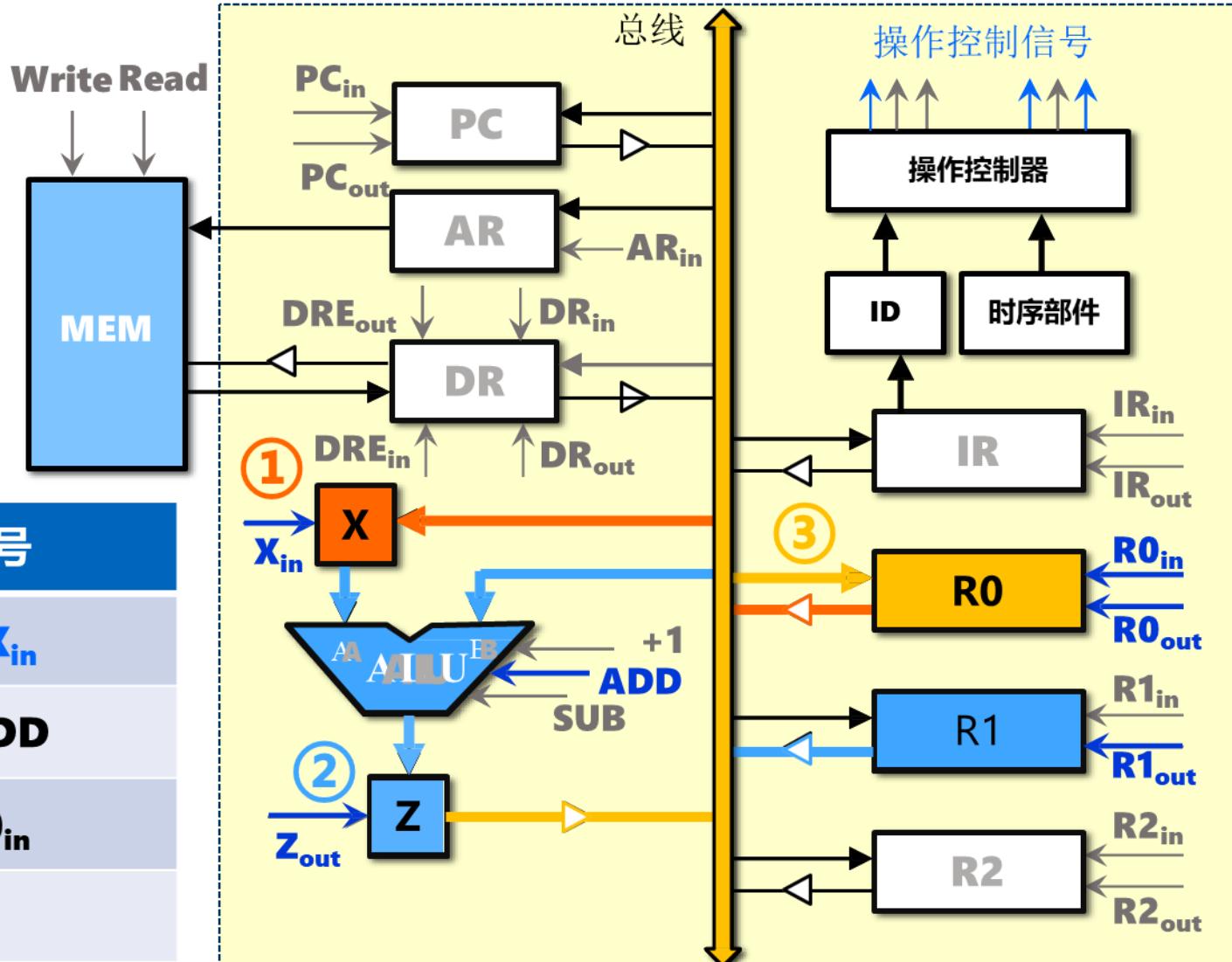
总线结构CPU指令周期

ADD指令执行数据通路

ADD R0,R1

$(R0) + (R1) \rightarrow R0$

节拍	数据通路	控制信号
T1	$(R0) \rightarrow X$	$R0_{out}, X_{in}$
T2	$(X) + (R1) \rightarrow Z$	$R1_{out}, ADD$
T3	$(Z) \rightarrow R0$	$Z_{out}, R0_{in}$
T4		

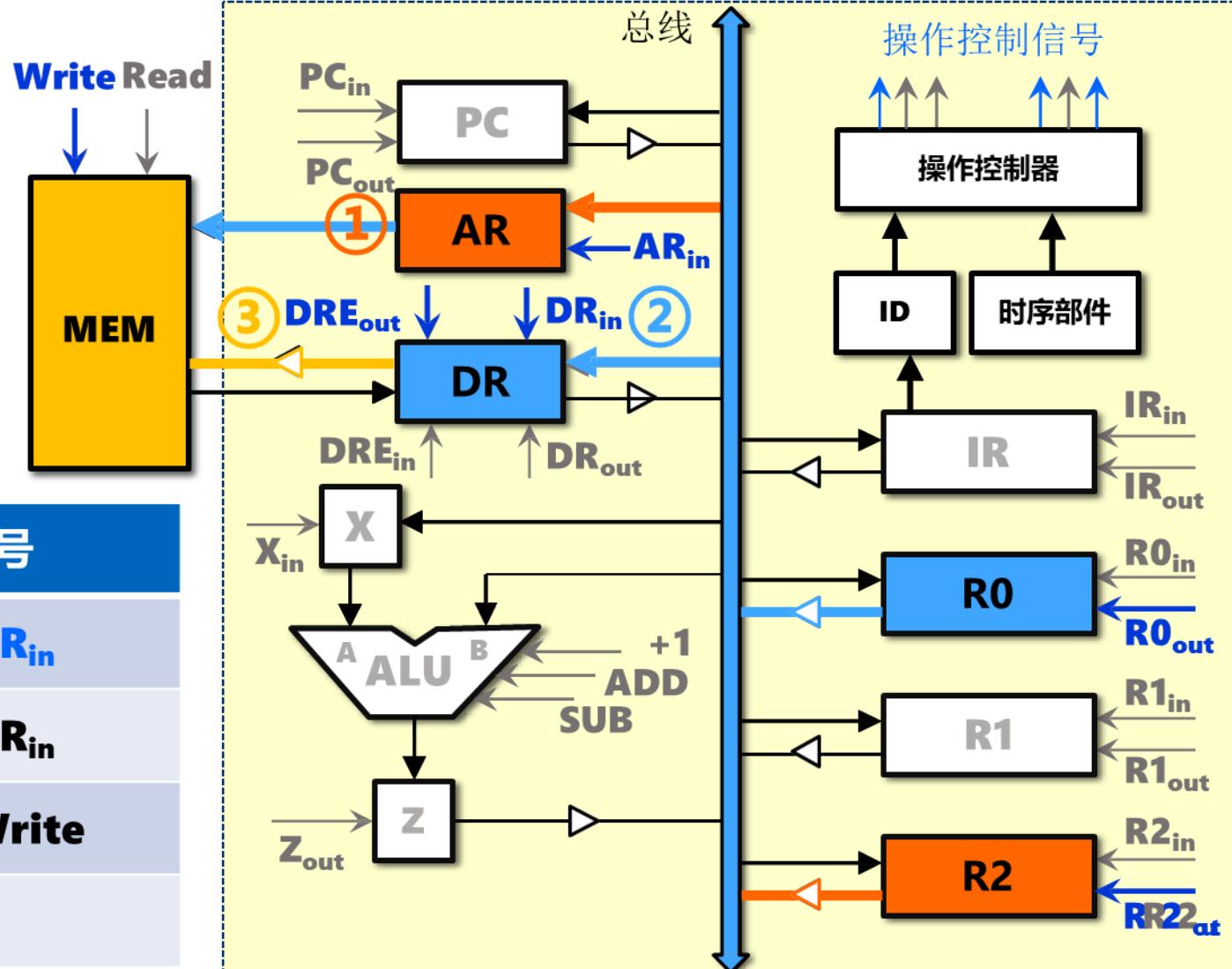


总线结构CPU指令周期

STORE指令 数据通路

STORE R0,(R2)
 $(R0) \rightarrow Mem[R2]$

节拍	数据通路	控制信号
T1	$(R2) \rightarrow AR$	$R2_{out}, AR_{in}$
T2	$(R0) \rightarrow DR$	$R0_{out}, DR_{in}$
T3	$(DR) \rightarrow Mem[AR]$	$DRE_{out}, Write$
T4		

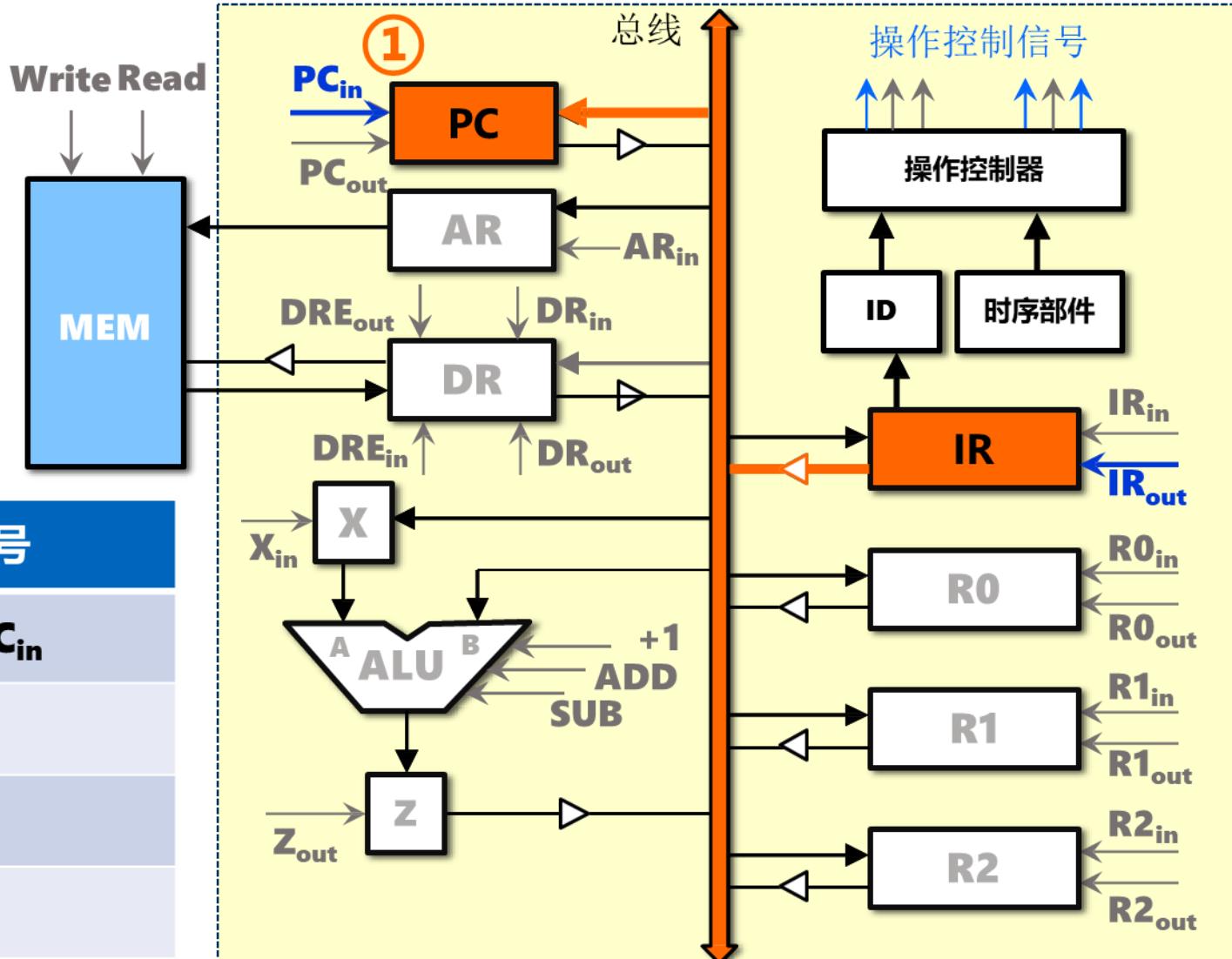


总线结构CPU指令周期

JMP指令数据通路

JMP 1000
 $(IR_A) \rightarrow PC$

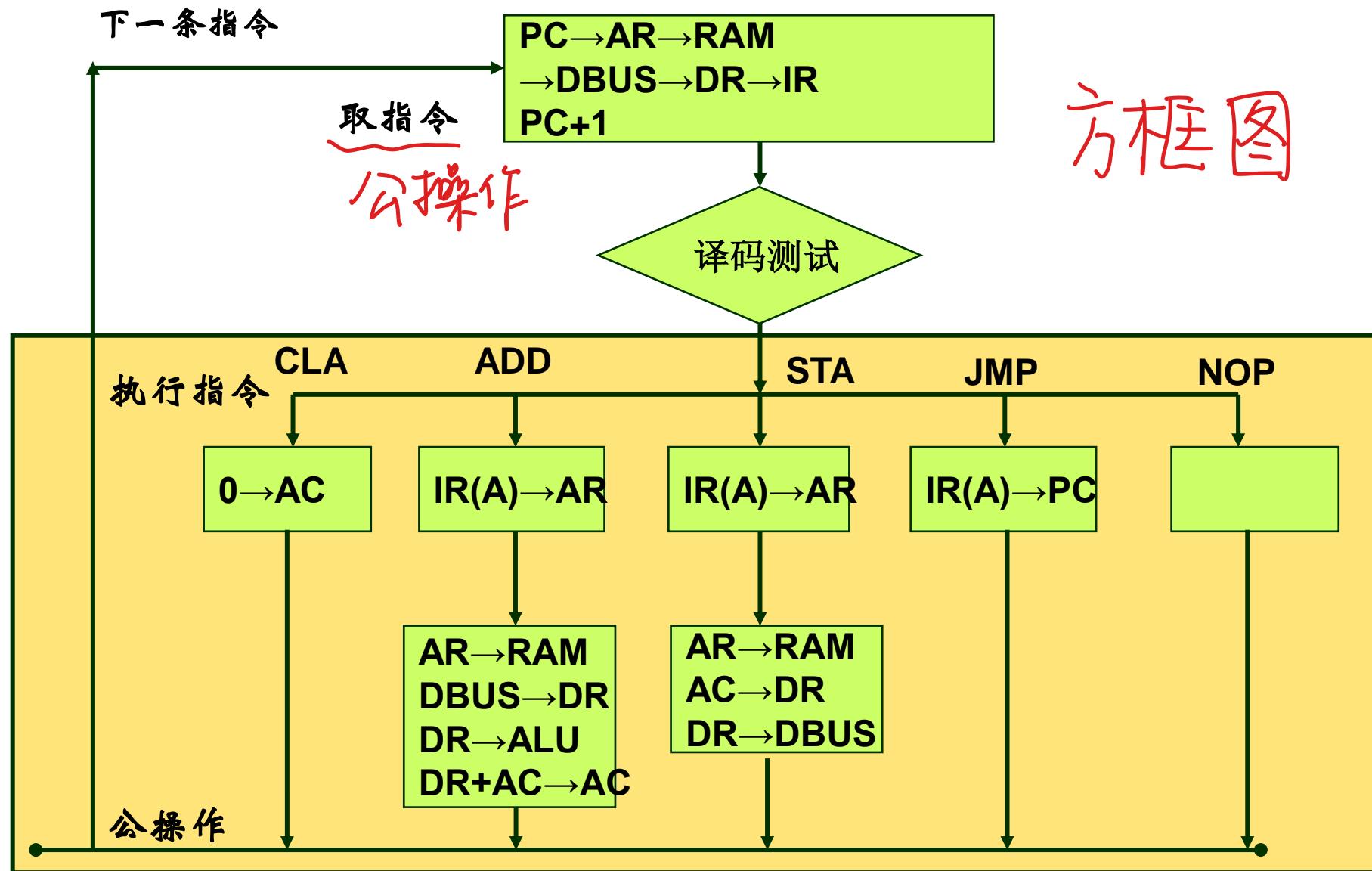
节拍	数据通路	控制信号
T1	$(IR_A) \rightarrow PC$	IR_{out}, PC_{in}
T2		
T3		
T4		



方框图语言

- 在进行计算机设计时，可以采用方框图语言来表示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，在时间上它依附于紧接它的前面一个方框的CPU周期，而不单独占用一个CPU周期。

方框图表示



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。
- 取指令是每条指令都有的，所以取指令也是公操作

本节目录

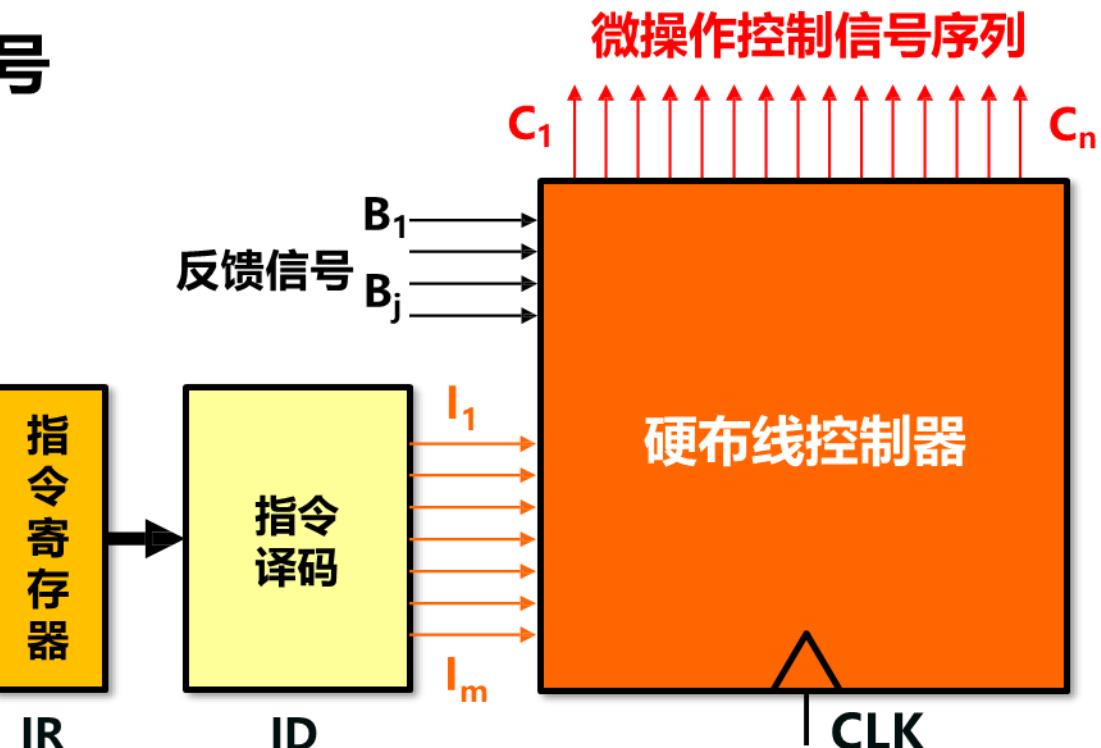
- 指令周期
- 硬布线控制器设计

硬布线控制器设计

基本原理

- 将控制器看成产生固定时序控制信号的逻辑电路
- 输入信号：指令译码，时钟信号，反馈信号
- 输出信号：功能部件控制信号序列
- 设计目标：最少元件，最快速度
- 理论基础：布尔代数
- 组成器件：门电路，触发器

控制器的核心功能是完成指令的自动执行，而指令的自动执行依赖于各功能部件之间的数据通路的建立，而数据通路的建立依赖于控制器生成控制信号的序列



机器指令字 → 控制器信号序列

硬布线控制器设计

单总线结构CPU

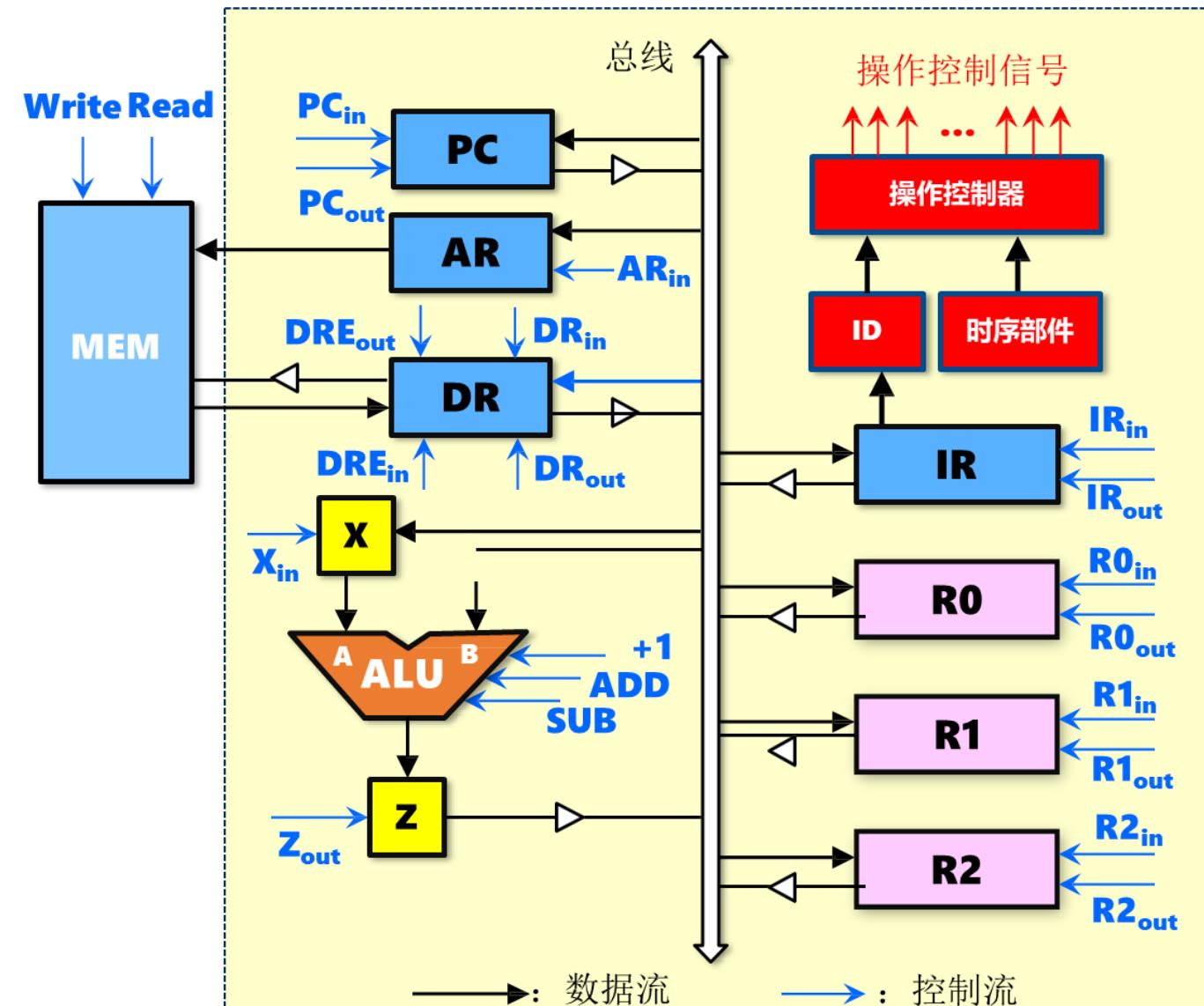
1. LOAD R0,6#

2. MOVE R1,10

3. ADD R0,R1

4. STORE R0,(R2)

5. JMP 1000



硬布线控制器设计

单总线结构CPU指令周期

节拍	控制信号(4 cycles)
T1	$PC_{out}, AR_{in}, X_{in}$
T2	+1, Read
T3	$Z_{out}, PC_{in}, DRE_{in}, Read$
T4	DR_{out}, IR_{in}

取指令周期

执行周期

■ 定长指令周期：传统三级时序

◆ 2个机器周期，8个时钟周期、慢、设计简单

■ 变长指令周期：现代时序

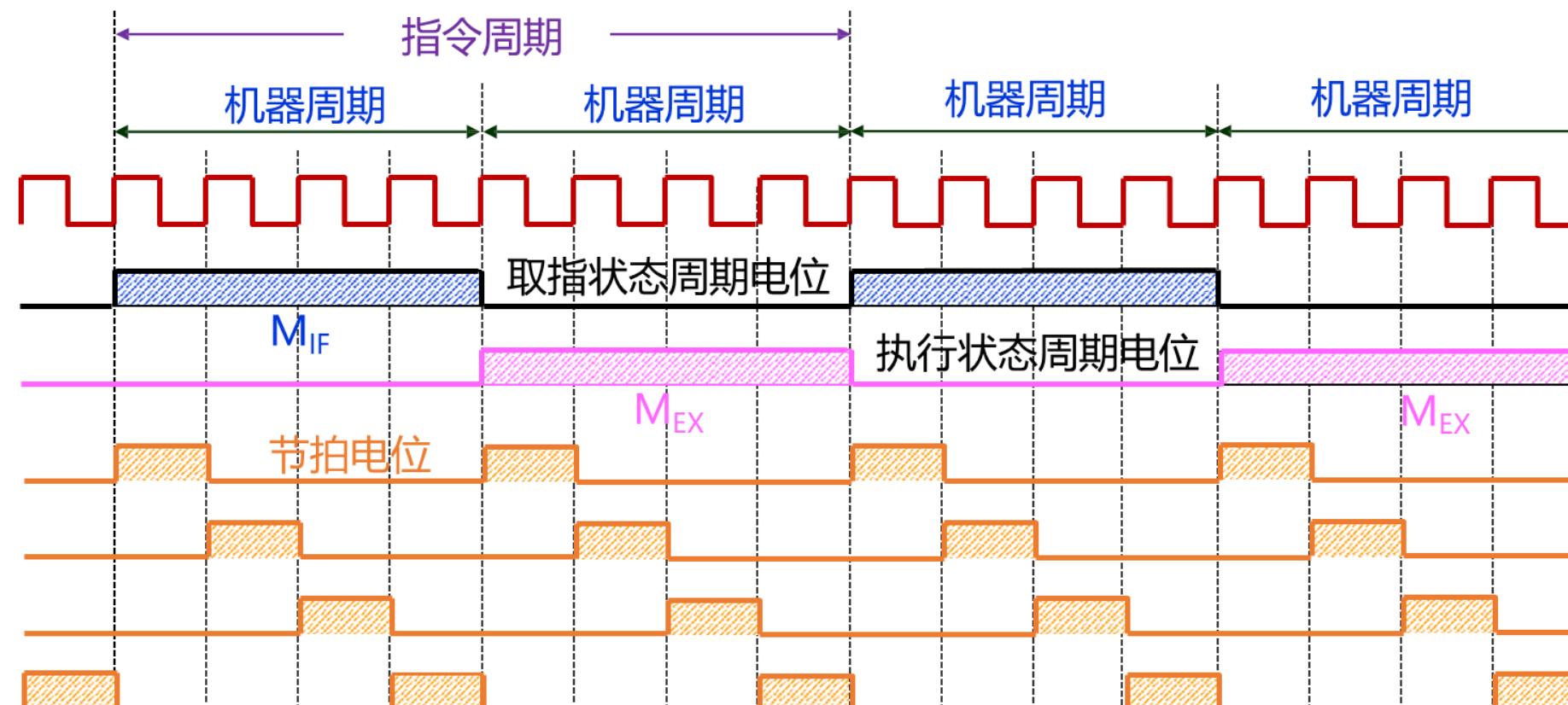
◆ 时钟周期数可变，快，设计复杂

节拍	LOAD (4 cycles)	MOVE (1 cycles)	ADD (3 cycles)	STORE (3 cycles)	JMP (1 cycles)
T5	IR_{out}, AR_{in}	$IR_{out}, R1_{in}$	$R0_{out}, X_{in}$	$R2_{out}, AR_{in}$	IR_{out}, PC_{in}
T6	Read		$R1_{out}, ADD$	$R0_{out}, DR_{in}$	
T7	$DRE_{in}, Read$		$Z_{out}, R0_{in}$	$DRE_{out}, Write$	
T8	$DR_{out}, R0_{in}$				

硬布线控制器设计

定长指令周期时序产生器 传统三级时序

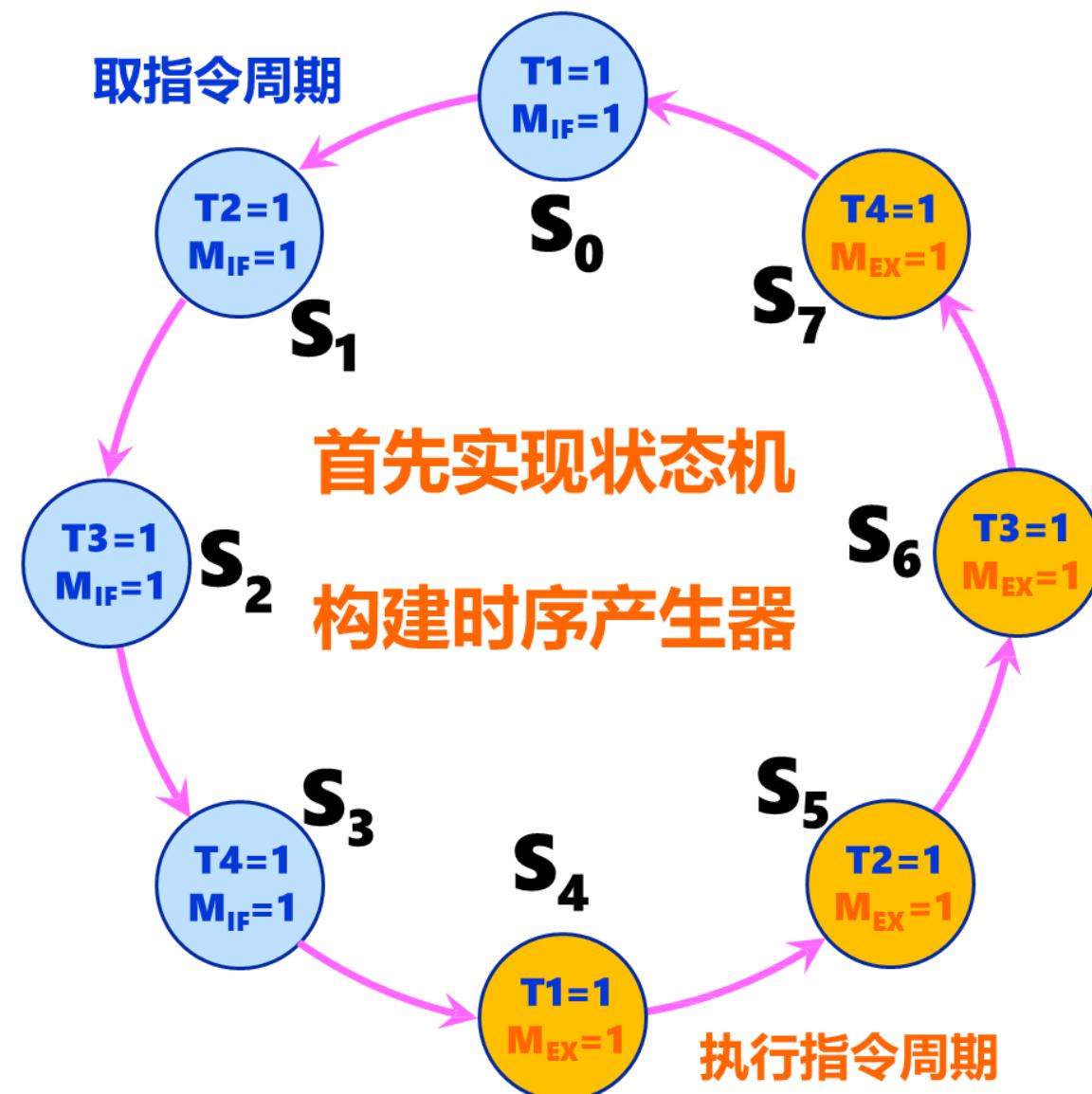
固定2个机器周期，8个时钟节拍



构建时序产生器

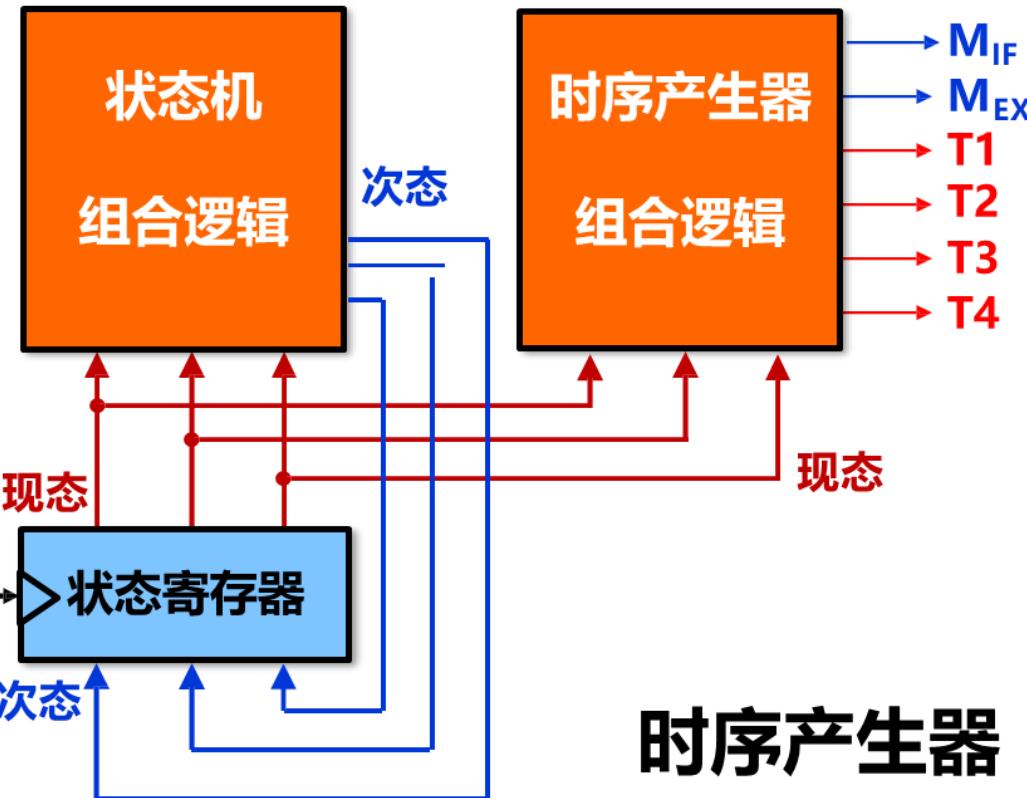
输出: M_{IF} , M_{EX} , T1, T2, T3, T4

硬布线控制器设计

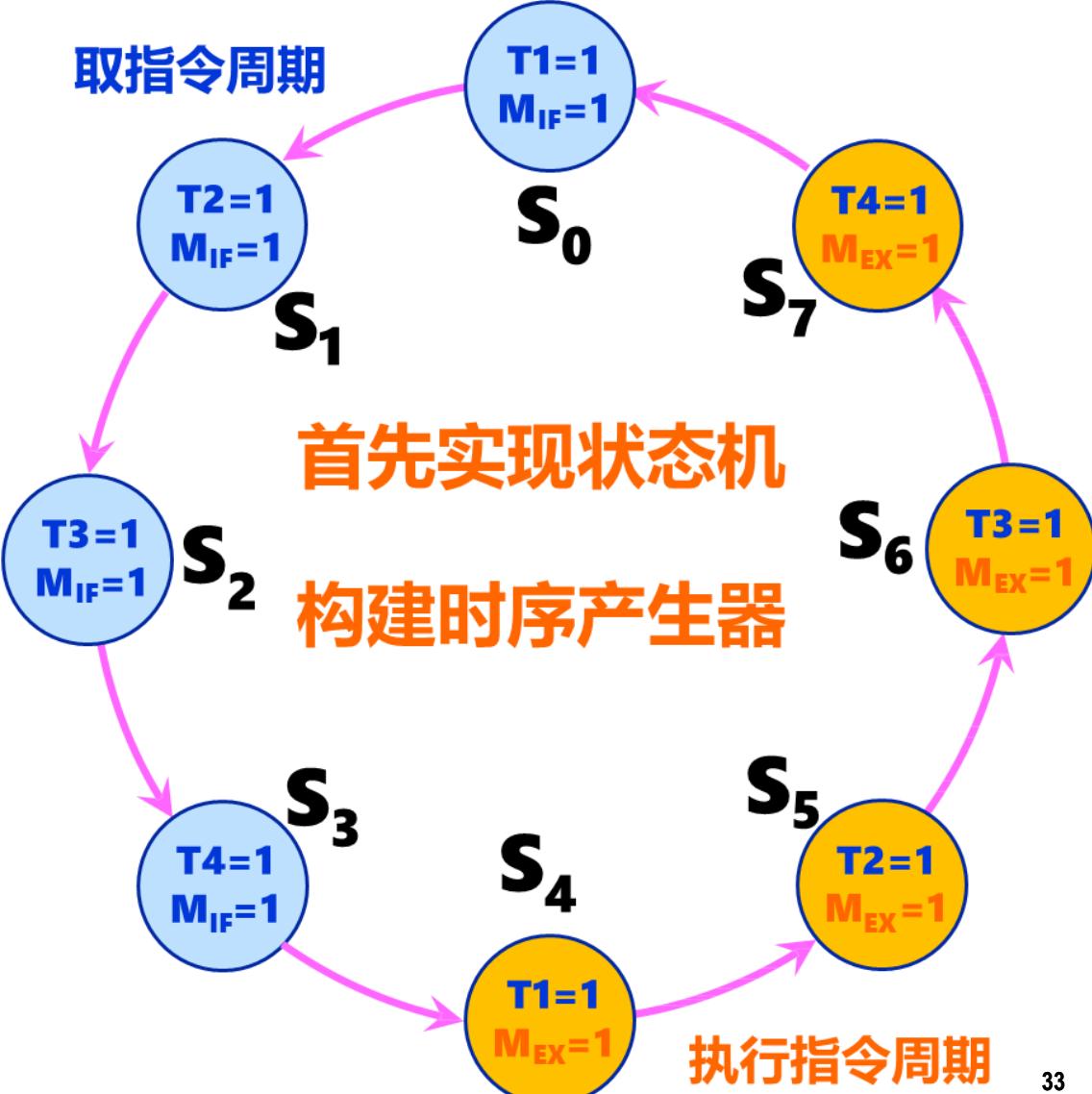


硬布线控制器设计

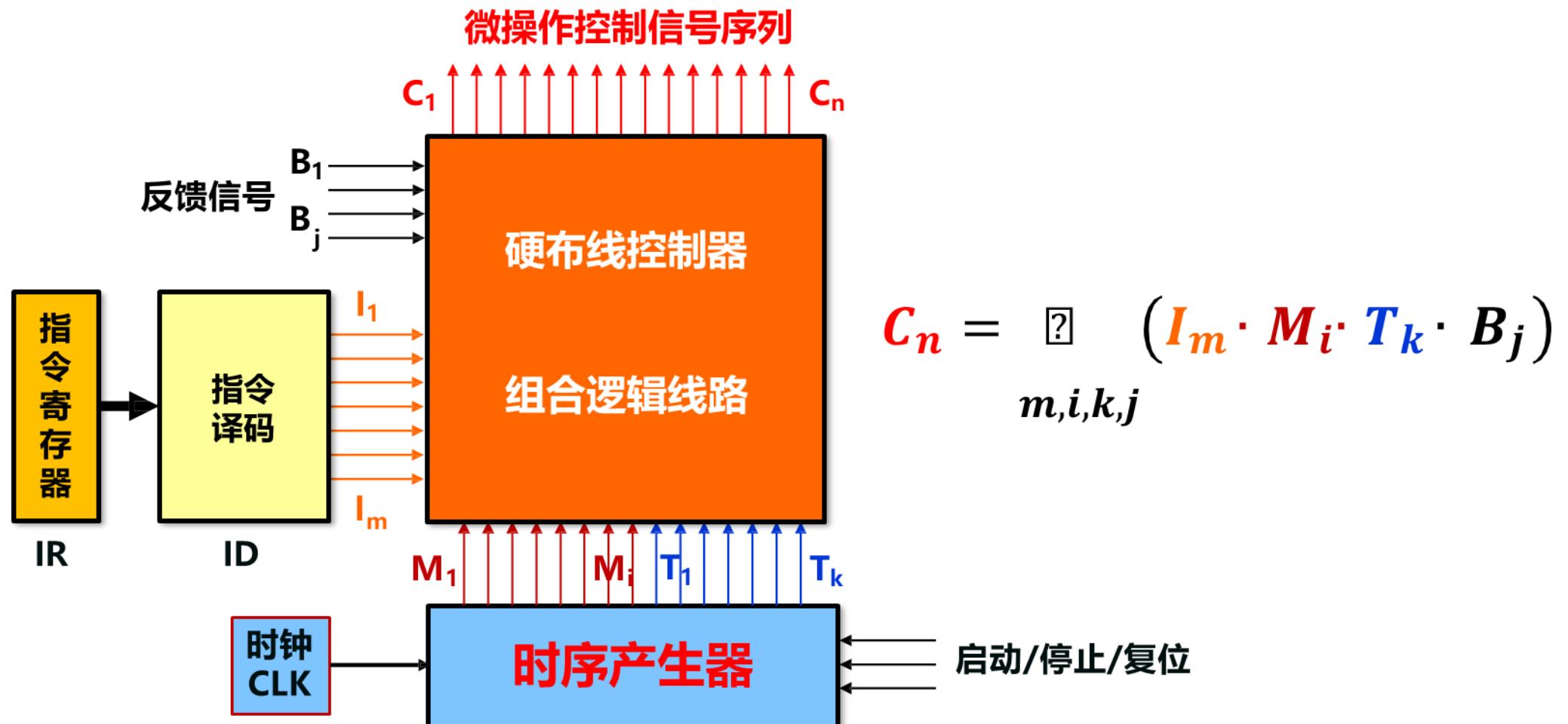
时序产生器状态机



时序产生器



硬布线控制器基本架构



时序产生器循环产生周期电位、节拍电位，供控制器对信号进行时间调制

硬布线控制器设计

单总线CPU控制信号生成

取指令周期
 M_{IF}

节拍	控制信号
T1	$PC_{out}, AR_{in}, X_{in}$
T2	+1, Read
T3	$Z_{out}, PC_{in}, DRE_{in}, Read$
T4	DR_{out}, IR_{in}

执行周期
 M_{EX}

节拍	LOAD	MOVE	ADD	STORE	JMP
T1	IR_{out}, AR_{in}	$IR_{out}, R1_{in}$	$R0_{out}, X_{in}$	$R2_{out}, AR_{in}$	IR_{out}, PC_{in}
T2	Read		$R1_{out}, ADD$	$R0_{out}, DR_{in}$	
T3	$DRE_{in}, Read$		$Z_{out}, R0_{in}$	$DRE_{out}, Write$	
T4	$DR_{out}, R0_{in}$				

$$C_n = \sum_{m,i,k,j} (I_m \cdot M_i \cdot T_k \cdot B_j)$$

- $Read = M_{IF} \cdot (T2+T3) + LOAD \cdot M_{EX} \cdot (T2+T3)$
- $AR_{in} = M_{IF} \cdot T1 + (LOAD+STORE) \cdot M_{EX} \cdot T1$

固定指令周期硬布线控制器设计过程

固定指令周期硬布线控制器设计过程

1. 设计三级时序产生器：所有指令固定机器周期数，节拍数，
2. 列出所有机器指令的指令周期流程图，明确每个节拍的控制信号；
3. 找出产生同一微操作控制信号的条件；
4. 写出各微操作控制信号的布尔表达式；
5. 化简各表达式；
6. 利用组合逻辑电路实现。

$$C_n = \bigvee_i \left(M_i \cdot T_k \cdot B_j \cdot \bigvee_m I_m \right)$$

变长指令周期硬布线控制器设计

单总线结构CPU控制信号表

节拍	控制信号(4 cycles)
T1	S₀ PC _{out} , AR _{in} , X _{in}
取指令周期	S₁ +1, Read
	S₂ Z _{out} , PC _{in} , DRE _{in} , Read
	S₃ DR _{out} , IR _{in}

■ 定长指令周期：传统三级时序

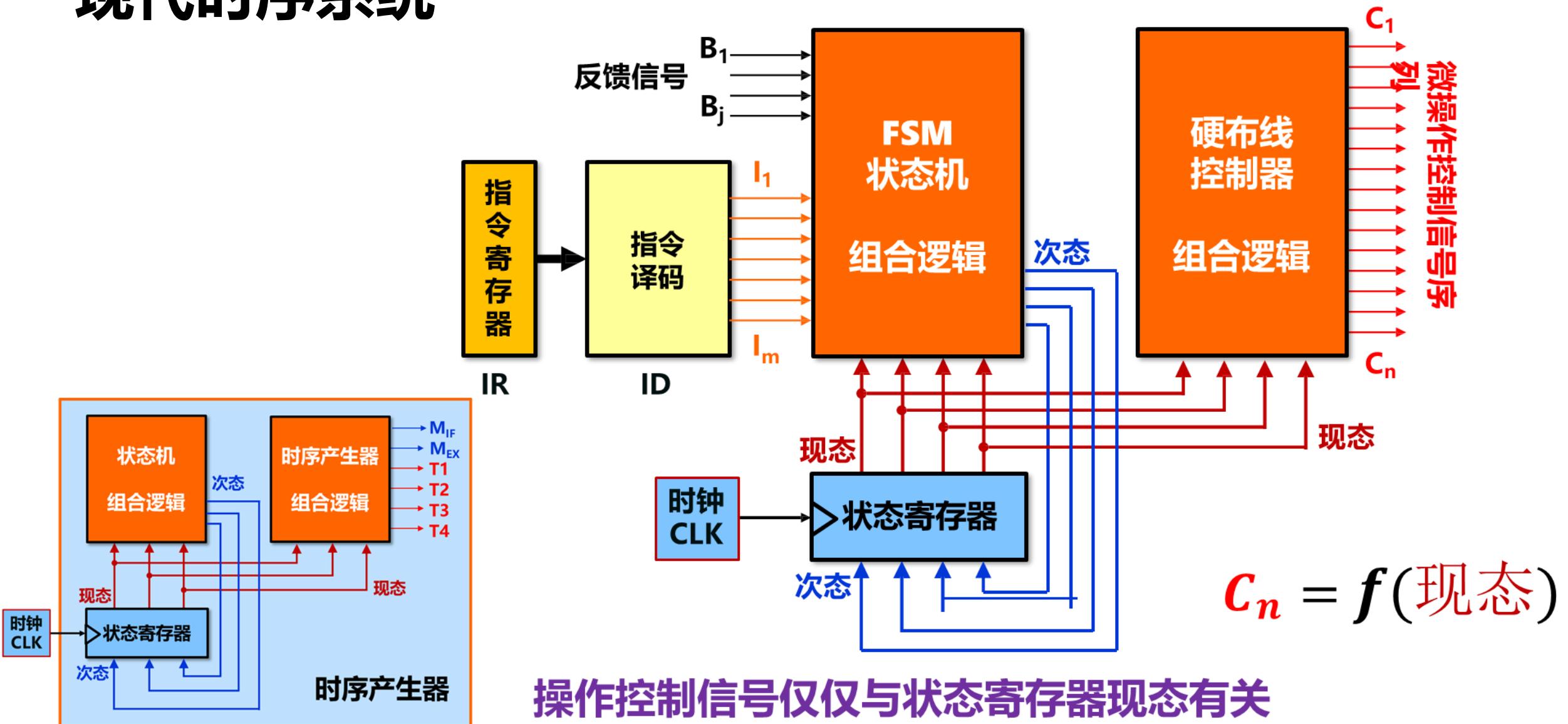
◆ 2个机器周期，8个时钟周期、慢、设计简单

■ 变长指令周期：现代时序

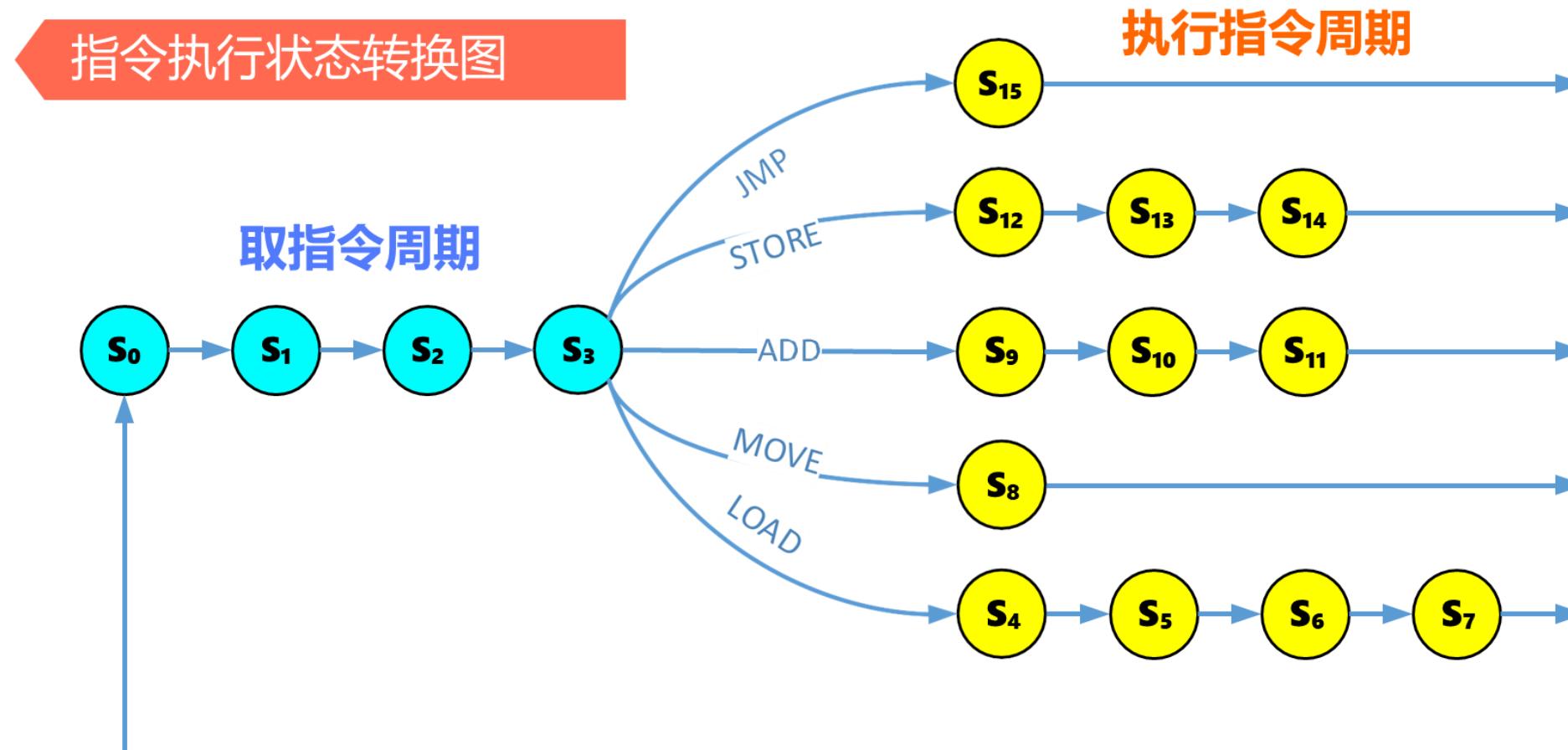
◆ 时钟周期数可变，快，设计复杂

节拍	LOAD (4 cycles)	MOVE (1 cycles)	ADD (3 cycles)	STORE (3 cycles)	JMP (1 cycles)
执行周期	S₄ IR _{out} , AR _{in}	S₈ IR _{out} , R1 _{in}	S₉ R0 _{out} , X _{in}	S₁₂ R2 _{out} , AR _{in}	S₁₅ IR _{out} , PC _{in}
	S₅ Read		S₁₀ R1 _{out} , ADD	S₁₃ R0 _{out} , DR _{in}	
	S₆ DRE _{in} , Read		S₁₁ Z _{out} , R0 _{in}	S₁₄ DRE _{out} , Write	
	S₇ DR _{out} , R0 _{in}				

现代时序系统



硬布线控制器设计



现态	LOAD	MOVE	ADD	STORE	JMP	次态
S_0	X	X	X	X	X	S_1
S_1	X	X	X	X	X	S_2

硬布线控制器设计

有限状态机真值表

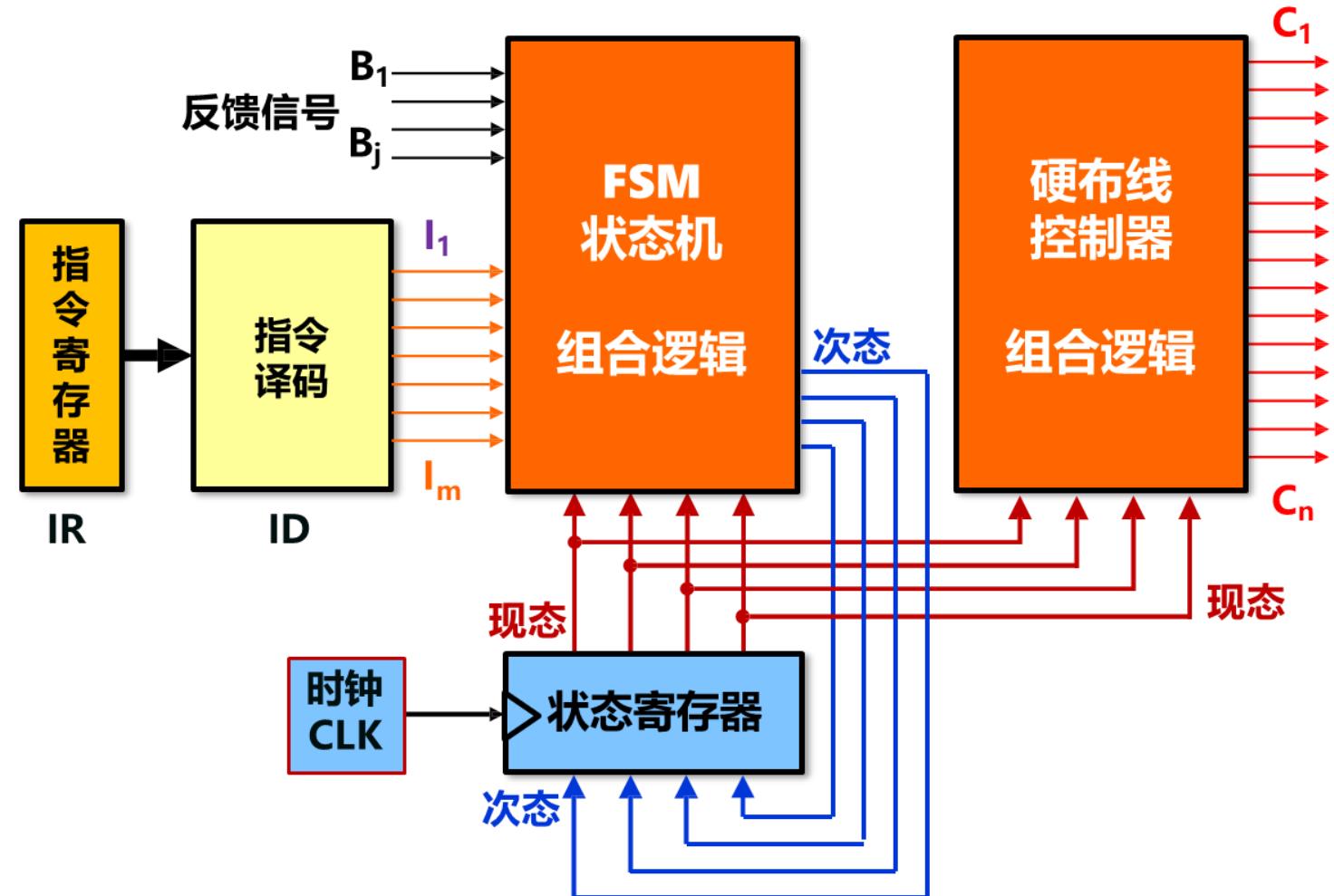
	现态	LOAD	MOVE	ADD	STORE	JMP	次态
	S0	X	X	X	X	X	S1
	S1	X	X	X	X	X	S2
	S2	X	X	X	X	X	S3
	S3	1					S4
	S3		1				S8
	S3			1			S9
	S3				1		S12
	S3					1	S15
LOAD	S4	X	X	X	X	X	S5
	S5	X	X	X	X	X	S6
	S6	X	X	X	X	X	S7
	S7	X	X	X	X	X	S0
MOVE	S8	X	X	X	X	X	S0

JMP	S15						S0

可变周期硬布线控制器设计

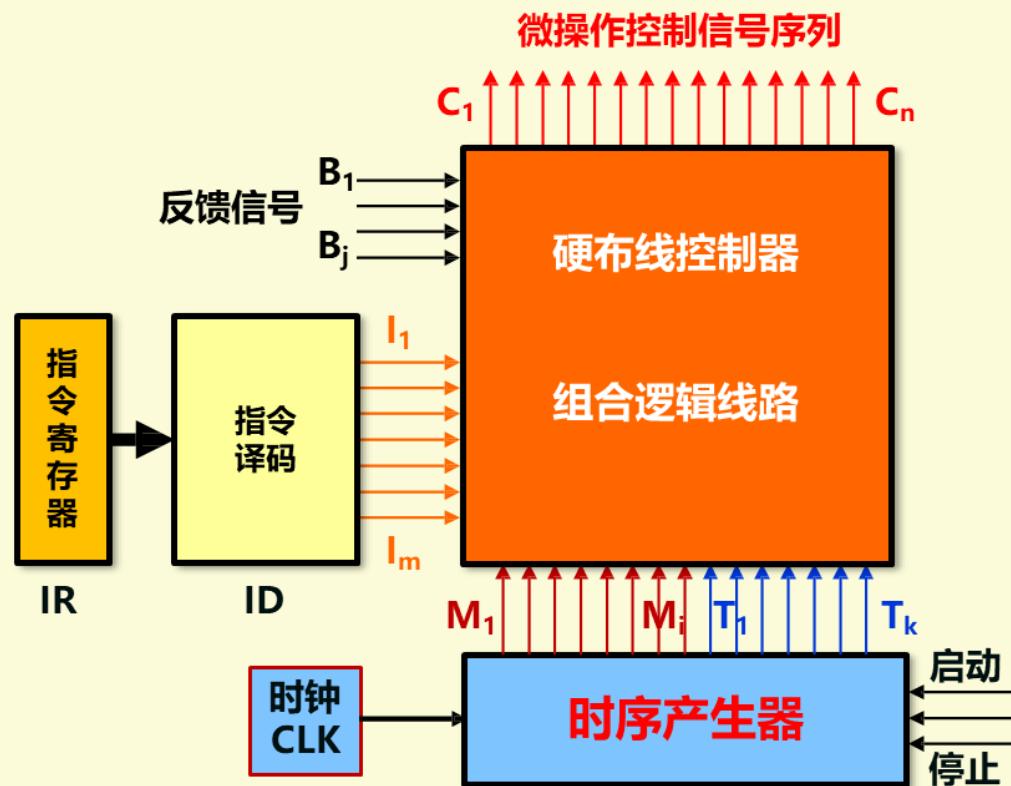
可变周期硬布线控制器设计步骤

1. 列出所有机器指令的指令周期流程图，明确每个节拍的控制信号；
2. 绘制指令执行状态转换图
3. 根据状态转换图构建状态机真值表，实现有限状态机组合逻辑
4. 实现控制器组合逻辑电路



硬布线控制器设计

传统时序与现代时序对比



传统三级时序：指令周期等长，慢，简单

现代时序：指令周期可变，快，复杂

