第10章　控制单元的设计

计算机组成原理

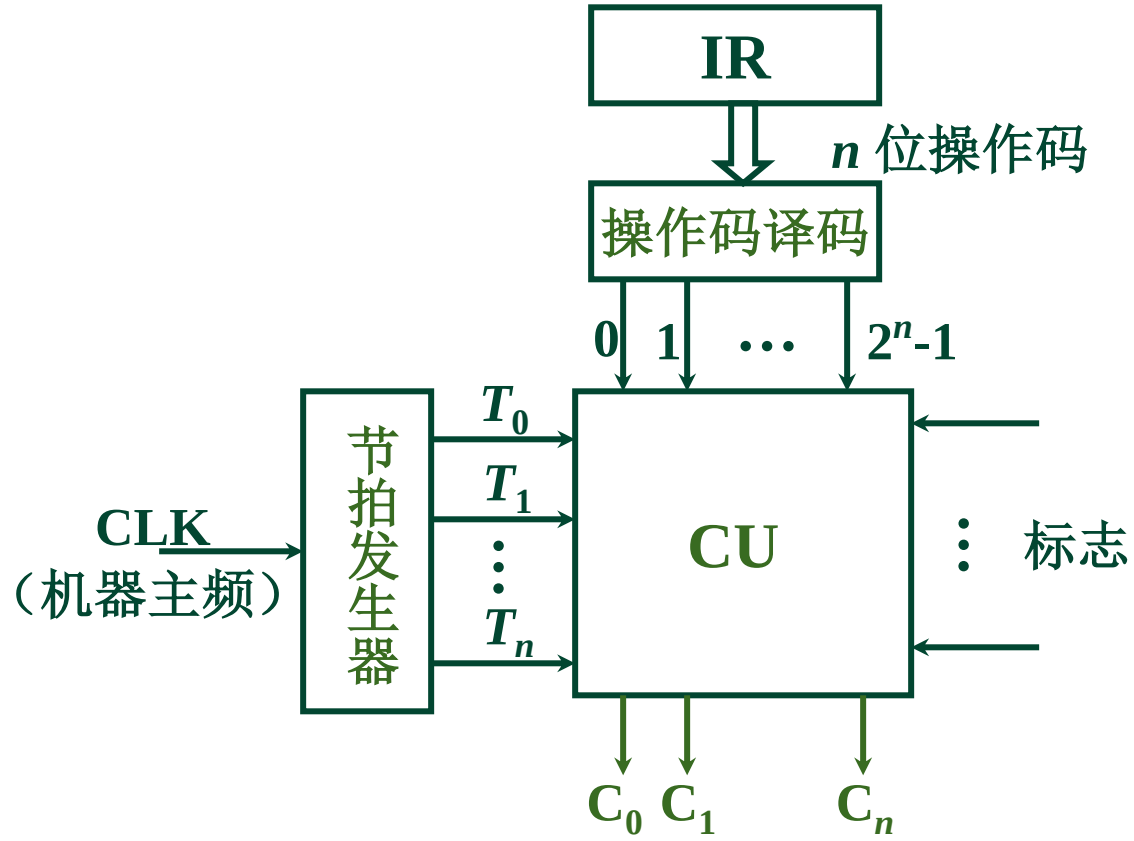
第10章　控制单元的设计

10.1　组合逻辑设计

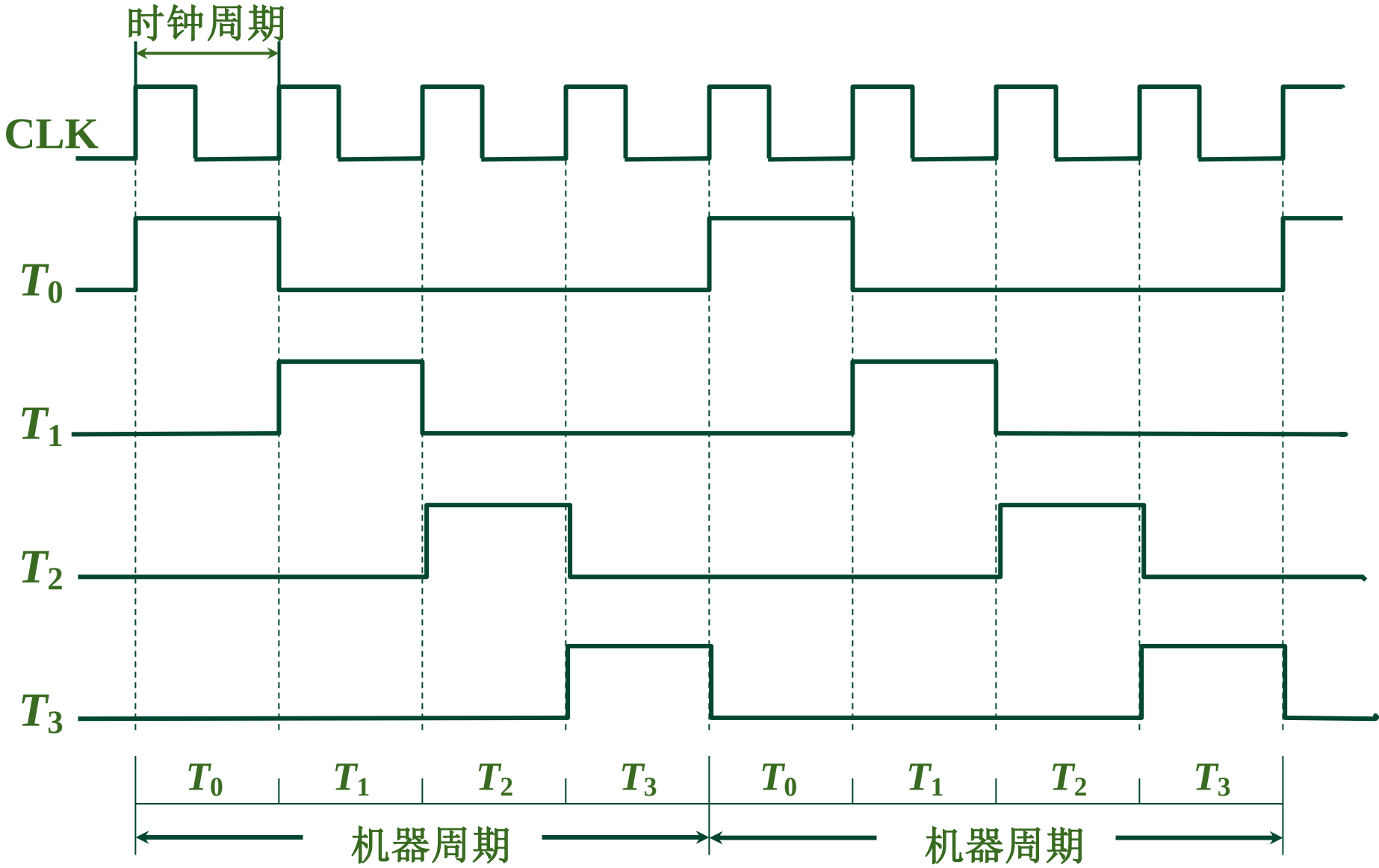
10.2　微程序设计

一、组合逻辑控制单元框图

1. CU 外特性



2. 节拍信号



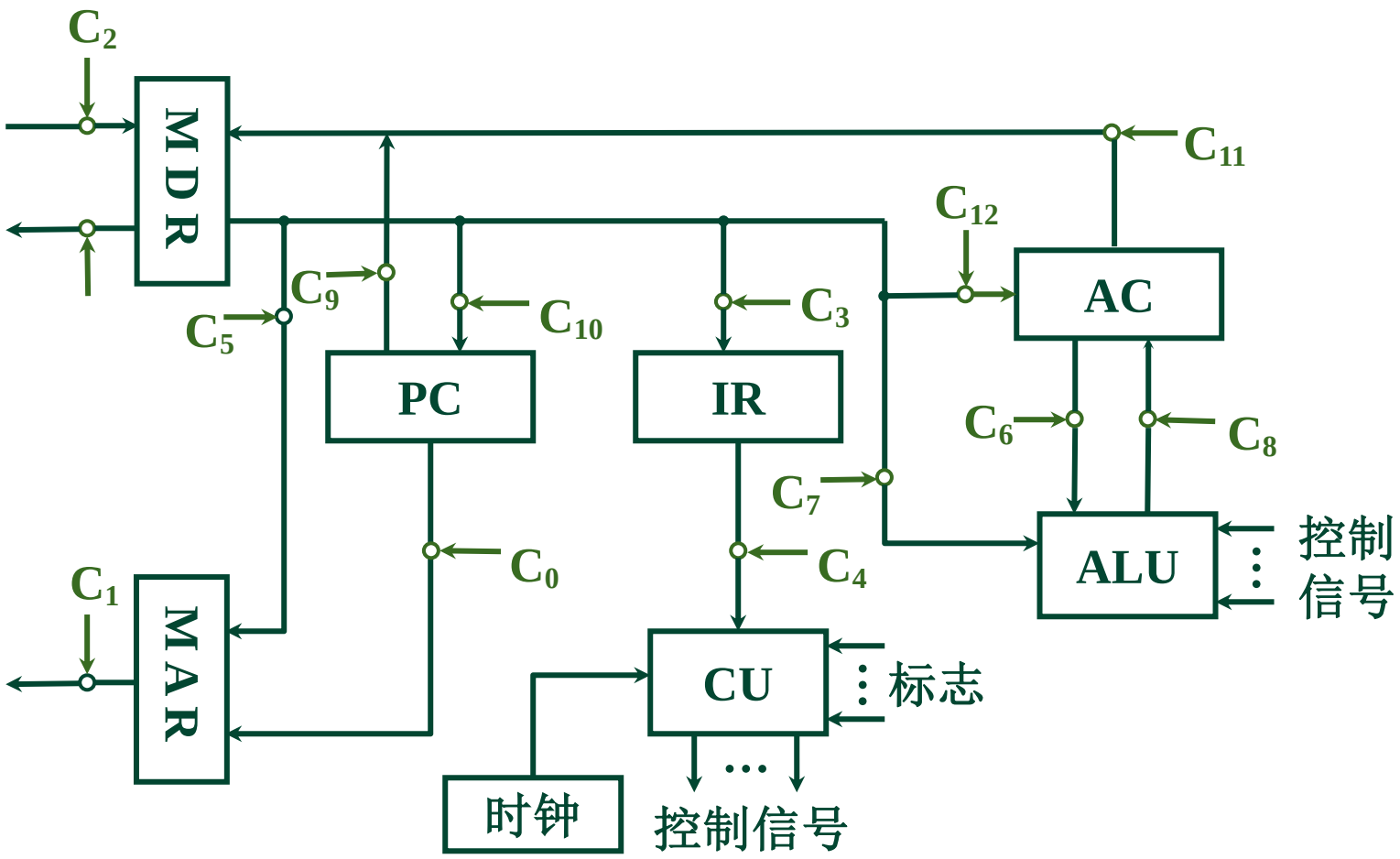
二、微操作的节拍安排

10.1

假设：采用 同步控制方式

一个 机器周期 内有 3个节拍（时钟周期）

CPU 内部结构采用非总线方式



1. 安排微操作时序的原则

原则一　微操作的 先后顺序 不得随意 更改

原则二　被控对象不同 的微操作

尽量安排在 一个节拍 内完成

原则三　占用 时间较短 的微操作

尽量 安排在 一个节拍 内完成

并允许有先后顺序

2. 取指周期 微操作的 节拍安排

T_0 PC ⟶ MAR

1 ⟶ R

T_1 M (MAR) ⟶ MDR

(PC) + 1 ⟶ PC

T_2 MDR ⟶ IR

OP (IR) ⟶ ID

原则二

原则二

原则三

3. 间址周期 微操作的 节拍安排

T_0 Ad (IR) ⟶ MAR

1 ⟶ R

T_1 M (MAR) ⟶ MDR

T_2 MDR ⟶ Ad (IR)

4. 执行周期 微操作的 节拍安排

① CLA T_0

T_1

T_2 $0 \longrightarrow AC$

② COM T_0

T_1

T_2 $\overline{AC} \longrightarrow AC$

③ SHR T_0

T_1

T_2 $L(AC) \longrightarrow R(AC)$

$AC_0 \longrightarrow AC_0$

④ CSL	T_0		
	T_1		
	T_2	R (AC) ⟶ L (AC)	$AC_0 \longrightarrow AC_n$
⑤ STP	T_0		
	T_1		
	T_2	0 ⟶ G	
⑥ ADD X	T_0	Ad (IR) ⟶ MAR	1 ⟶ R
	T_1	M (MAR) ⟶ MDR	
	T_2	(AC) + (MDR) ⟶ AC	
⑦ STA X	T_0	Ad (IR) ⟶ MAR	1 ⟶ W
	T_1	AC ⟶ MDR	
	T_2	MDR ⟶ M (MAR)	

⑧ LDA X

T_0 Ad (IR) ⟶ MAR　1 ⟶ R

T_1 M (MAR) ⟶ MDR

T_2 MDR ⟶ AC

⑨ JMP X

T_0

T_1

T_2 Ad (IR) ⟶ PC

⑩ BAN X

T_0

T_1

T_2 $A_0 \cdot Ad(IR) + \overline{A}_0 \cdot PC \longrightarrow PC$

5. 中断周期 微操作的 节拍安排

T_0　0 ⟶ MAR　　1 ⟶ W　　硬件关中断

T_1　PC ⟶ MDR

T_2　MDR ⟶ M (MAR)　　向量地址 ⟶ PC

中断隐指令完成

节拍安排例子

- 教材P398-400

例10.1~10.2

三、组合逻辑设计步骤

- 设计指令的操作码，确定指令长度是固定的还是变长的。
- 确定机器周期、节拍和时钟周期，确定机器周期是固定的还是变长的。
- 根据指令功能和CPU的结构图，绘制每条指令的微操作流程图并综合成一个总的流程图。
- 给微操作流程图安排时序，确定每条指令所需的机器周期及在各机器周期需完成的操作，排出微操作时间表。
- 根据操作时间表写出微操作的逻辑表达式，即

  微操作＝周期·节拍·时钟脉冲·指令码·其他条件
- 根据微操作的表达式，画出组合逻辑电路

三、组合逻辑设计步骤

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
FE 取指	T_0		PC⟶ MAR						
			1⟶ R						
	T_1		M(MAR) ⟶MDR						
			(PC) +1 ⟶PC						
	T_2		MDR⟶ IR						
			OP(IR) ⟶ID						
		I	1⟶ IND						
		$\bar{I}$	1⟶ EX						

间址特征

三、组合逻辑设计步骤

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
IND 间址	T_0		Ad (IR) ⟶ MAR						
			1⟶ R						
	T_1		M(MAR) ⟶MDR						
	T_2		MDR⟶ Ad (IR)						
		$\overline{\text{IND}}$	1⟶ EX						

间址周期标志

三、组合逻辑设计步骤

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
EX 执行	T_0		Ad (IR) ⟶ MAR						
			1 ⟶ R						
			1 ⟶ W						
	T_1		M(MAR) ⟶ MDR						
			AC ⟶ MDR						
	T_2		(AC)+(MDR) →AC						
			MDR ⟶ M(MAR)						
			MDR ⟶ AC						
			0 ⟶ AC						

三、组合逻辑设计步骤

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
FE 取指	T_0		PC⟶ MAR	1	1	1	1	1	1
			1⟶ R	1	1	1	1	1	1
	T_1		M(MAR) ⟶MDR	1	1	1	1	1	1
			(PC) +1⟶ PC	1	1	1	1	1	1
	T_2		MDR⟶ IR	1	1	1	1	1	1
			OP(IR)⟶ ID	1	1	1	1	1	1
		I	1⟶ IND			1	1	1	1
		$\bar{I}$	1⟶ EX	1	1	1	1	1	1

三、组合逻辑设计步骤

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
IND 间址	T_0		Ad (IR) ⟶ MAR			1	1	1	1
			1⟶ R			1	1	1	1
	T_1		M(MAR) ⟶ MDR			1	1	1	1
	T_2		MDR⟶ Ad (IR)			1	1	1	1
		$\overline{\mathbf{IND}}$	1⟶ EX			1	1	1	1

三、组合逻辑设计步骤

1. 列出操作时间表

教材P402

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
EX 执行	T_0		Ad (IR) ⟶ MAR			1	1	1	
			1⟶ R			1		1	
			1 ⟶ W				1		
	T_1		M(MAR) ⟶MDR			1		1	
			AC⟶ MDR				1		
	T_2		(AC)+(MDR)→AC			1			
			MDR⟶ M(MAR)				1		
			MDR ⟶AC					1	
			0⟶ AC	1					

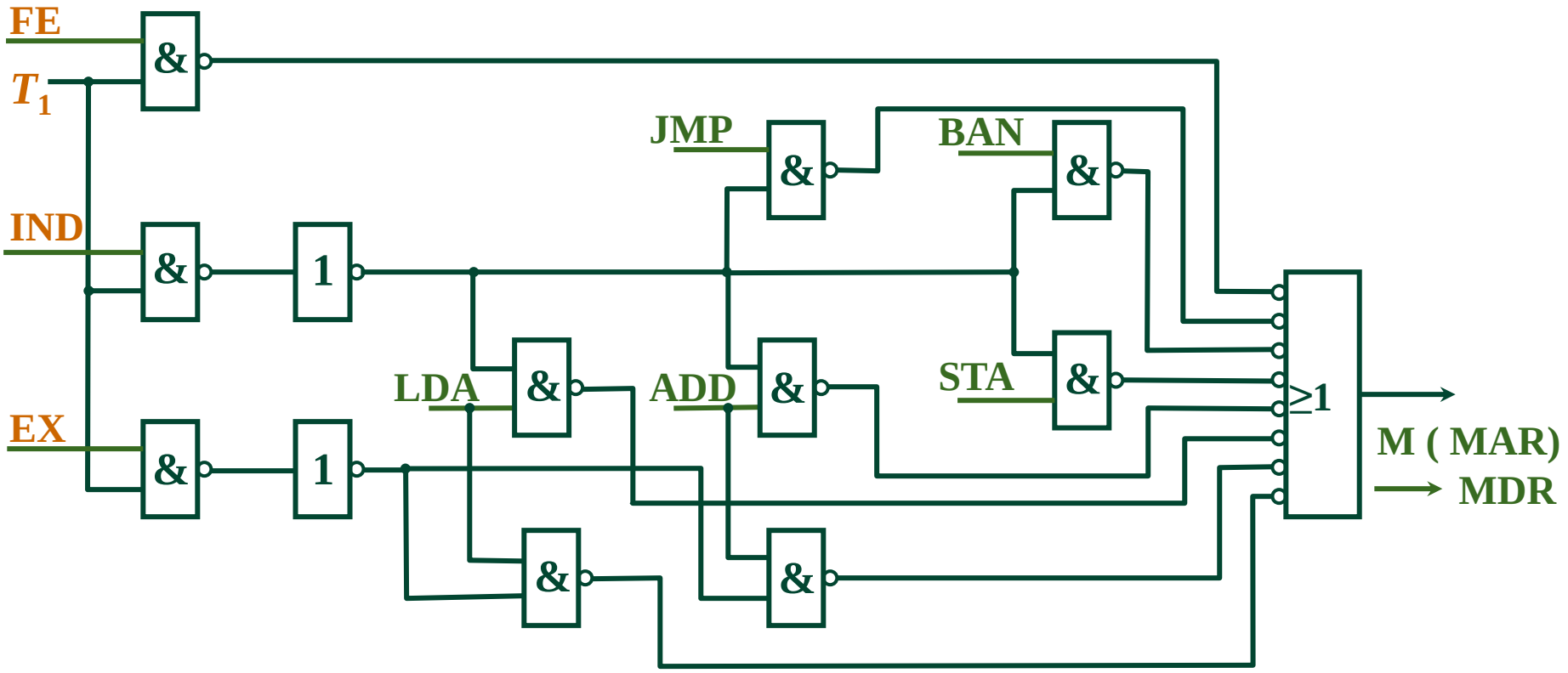
2. 写出微操作命令的最简表达式

$$M(MAR) \longrightarrow MDR$$

$$= FE \cdot T_1 + IND \cdot T_1 (ADD + STA + LDA + JMP + BAN) + EX \cdot T_1 (ADD + LDA)$$

$$= T_1\{FE + IND (ADD + STA + LDA + JMP + BAN) + EX (ADD + LDA)\}$$

3. 画出逻辑图



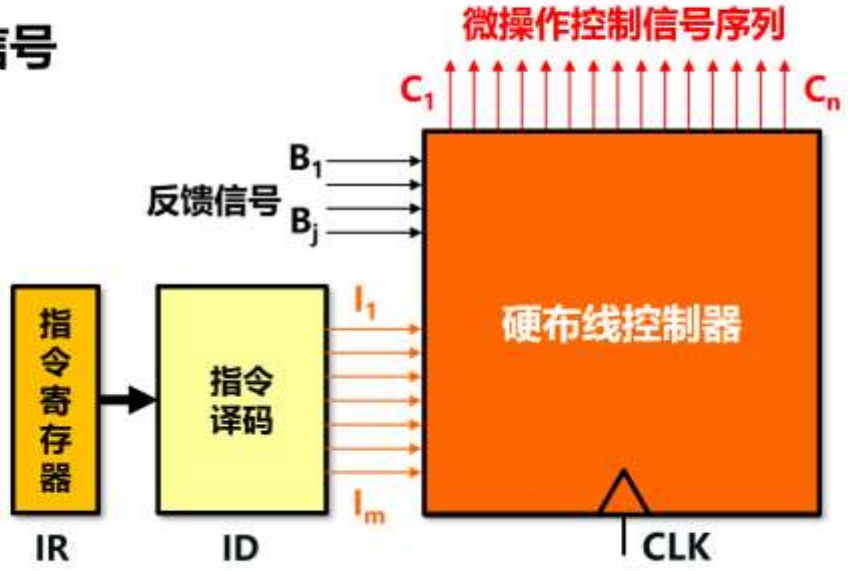
特点

- 思路清晰，简单明了
- 庞杂，调试困难，修改困难
- 速度快 （RISC）

实例：单总线CPU硬布线设计流程

1 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

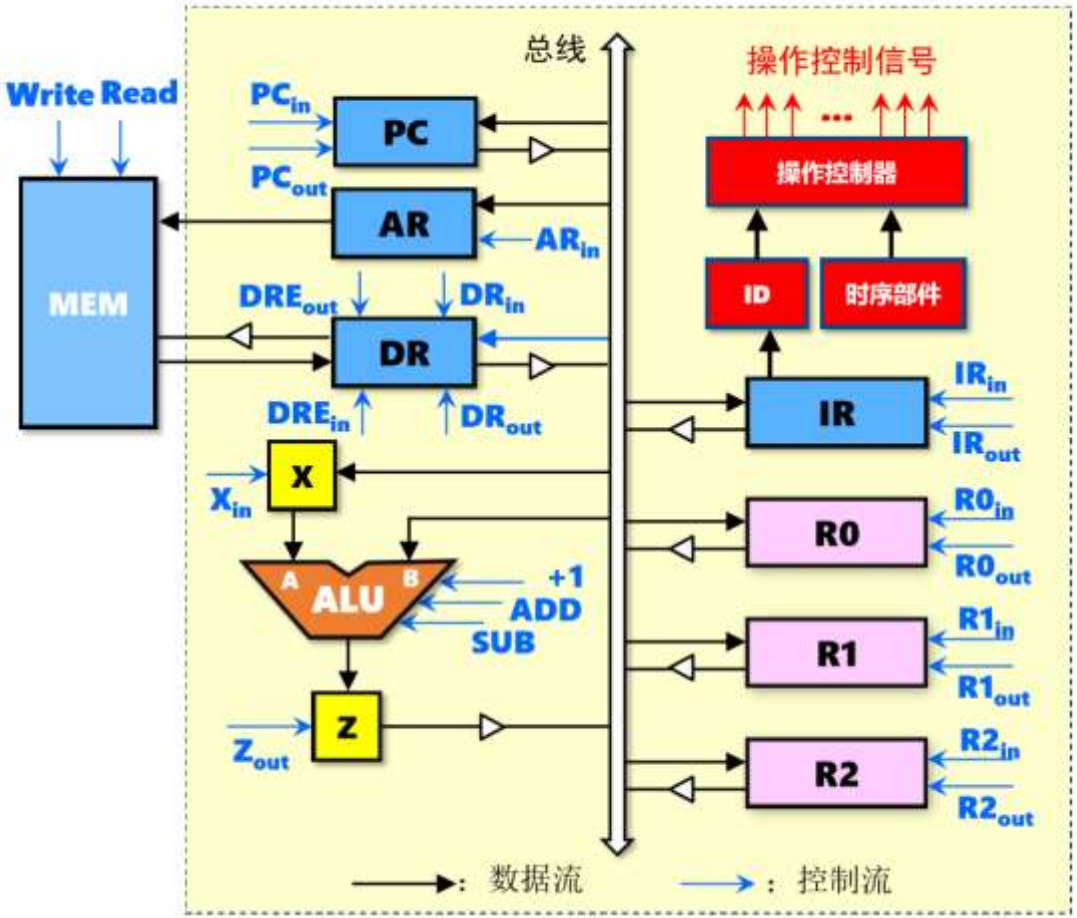


机器指令字 → 控制器信号序列

实例：单总线CPU硬布线设计流程

2 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



实例：单总线CPU硬布线设计流程

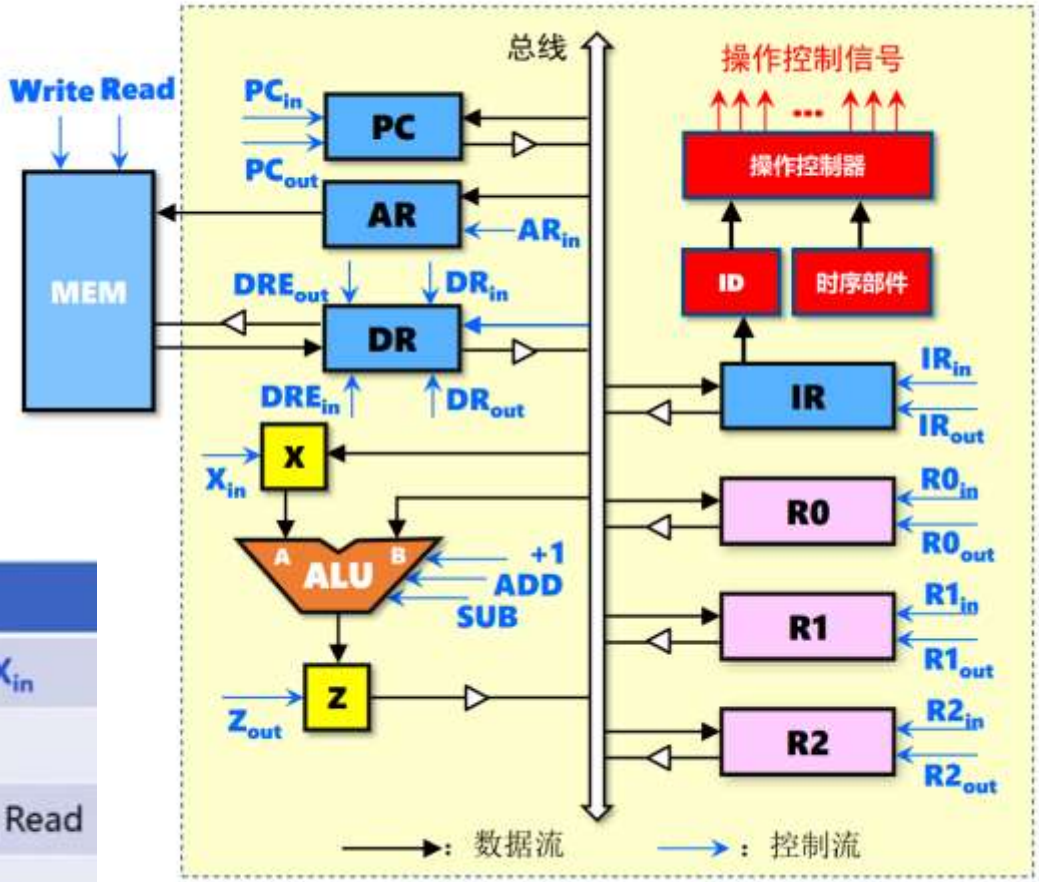
#	指令	指令功能
1	LOAD R0,6#	Mem[6] → R0
2	MOVE R1,10	10 → R1
3	ADD R0,R1	(R0) + (R1) → R0
4	STORE R0,(R2)	(R0) → Mem[(R2)]
5	JMP 1000	1000 → PC

实例：单总线CPU硬布线设计流程

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}



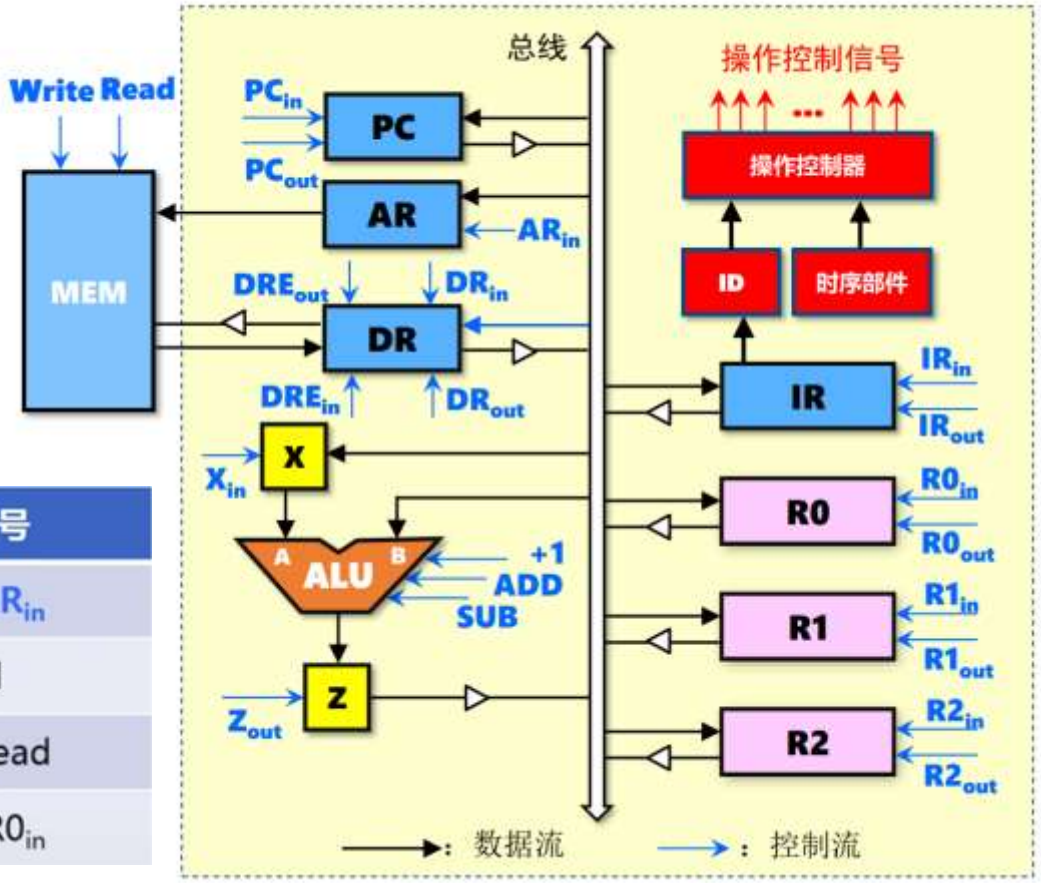
实例：单总线CPU硬布线设计流程

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}, $R0_{in}$



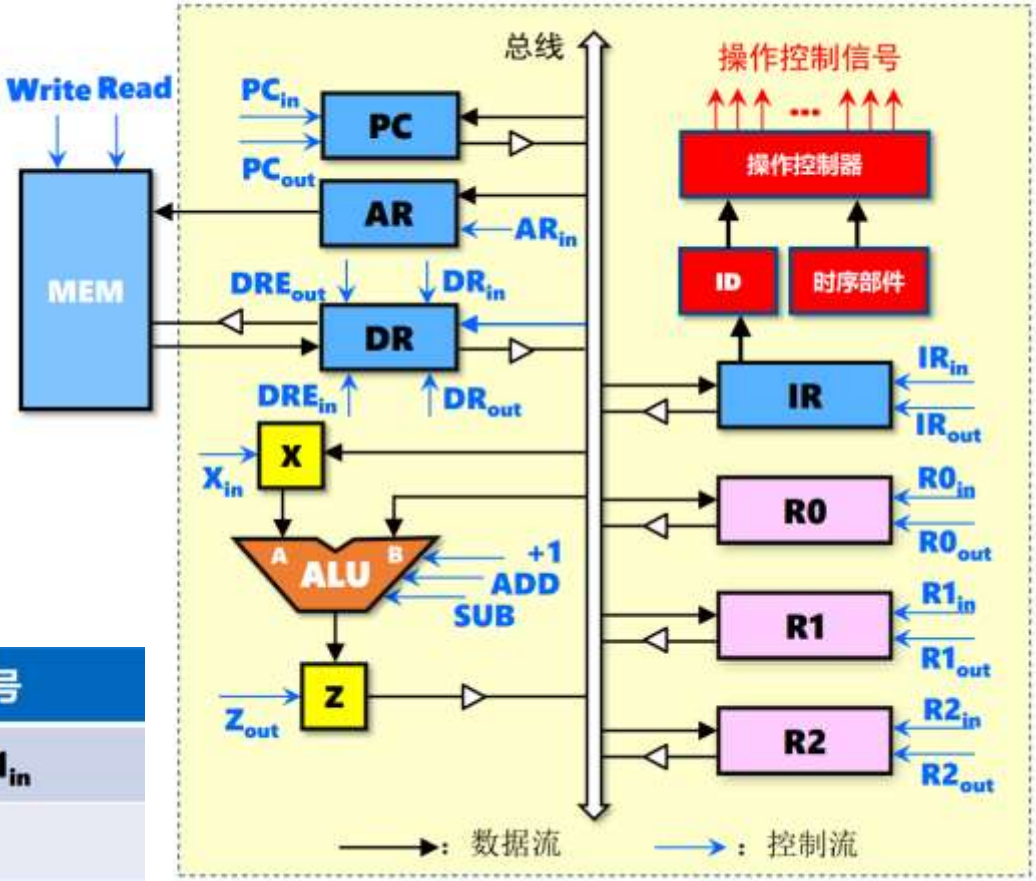
实例：单总线CPU硬布线设计流程

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	$(IR_A) \rightarrow R[1]$	IR_{out}, $R1_{in}$
T2		
T3		
T4		

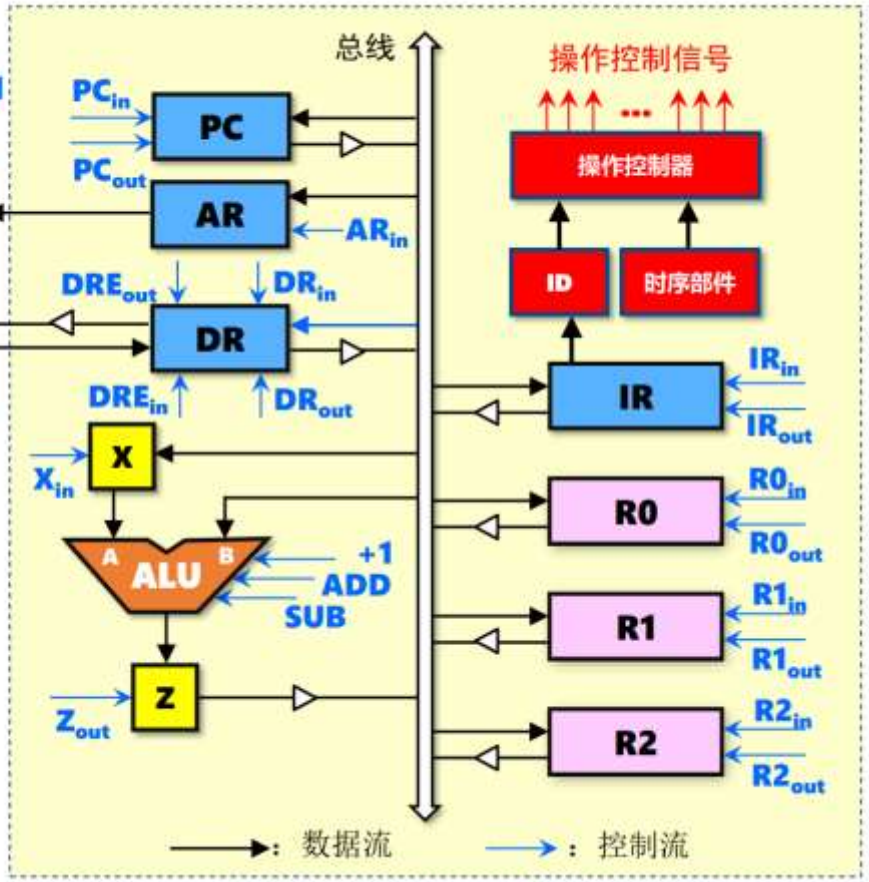


实例：单总线CPU硬布线设计流程

ADD指令执行数据通路

ADD R0,R1

(R0)+(R1) → R0



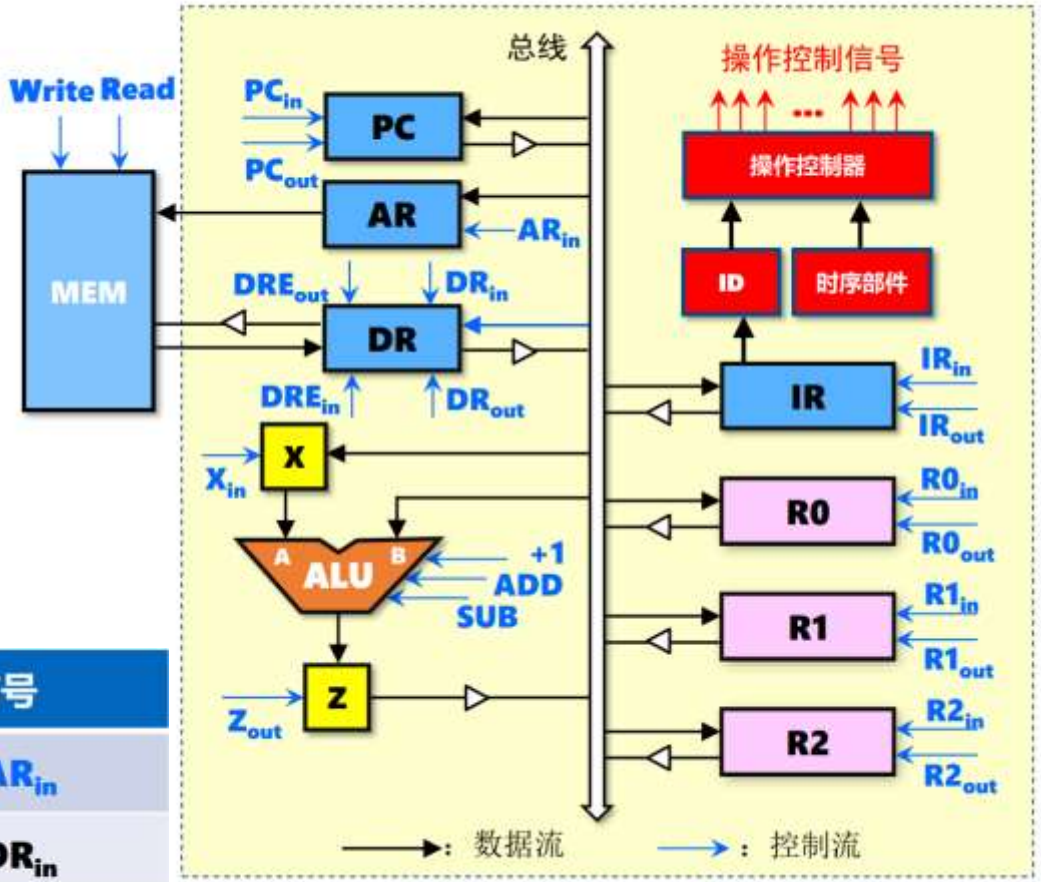
节拍	数据通路	控制信号
T1	(R0)→X	$R0_{out}$, X_{in}
T2	(X)+(R1)→Z	$R1_{out}$,ADD
T3	(Z)→R0	Z_{out},$R0_{in}$
T4		

实例：单总线CPU硬布线设计流程

STORE指令 数据通路

STORE R0,(R2)

(R0) → Mem[R2]



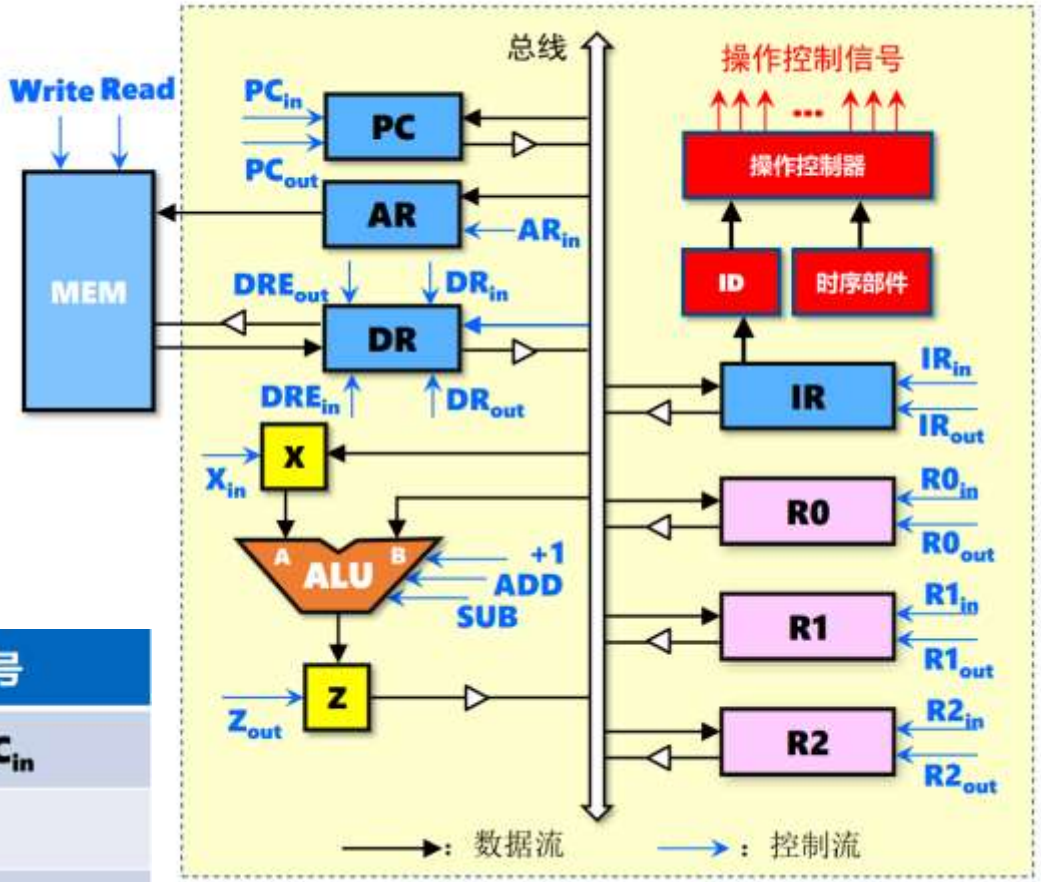
节拍	数据通路	控制信号
T1	(R2)→AR	$R2_{out}$, AR_{in}
T2	(R0)→DR	$R0_{out}$, DR_{in}
T3	(DR) →Mem[AR]	DRE_{out}, Write
T4		

实例：单总线CPU硬布线设计流程

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		

实例：单总线CPU硬布线设计流程

3 单总线结构CPU指令周期

取指令周期

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}

- 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

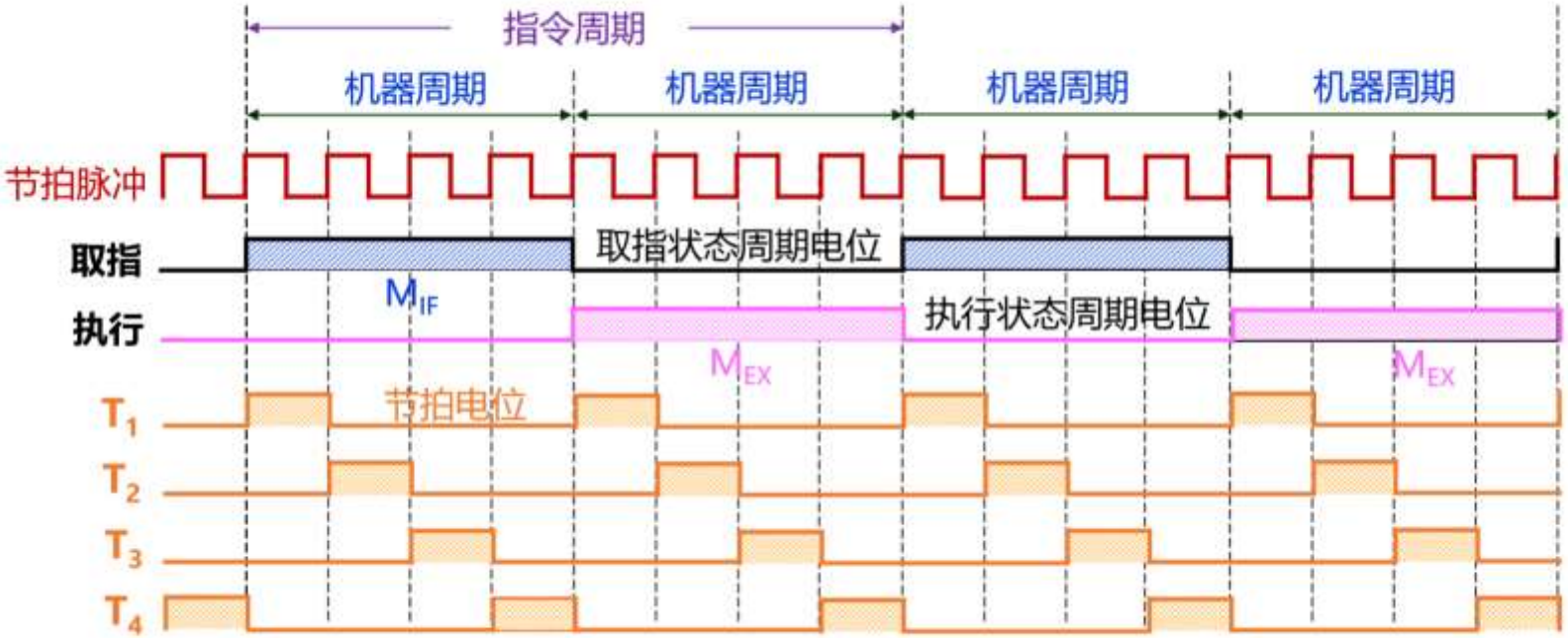
执行周期

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}, $R0_{in}$				

实例：单总线CPU硬布线设计流程

4 定长指令周期时序产生器 传统三级时序

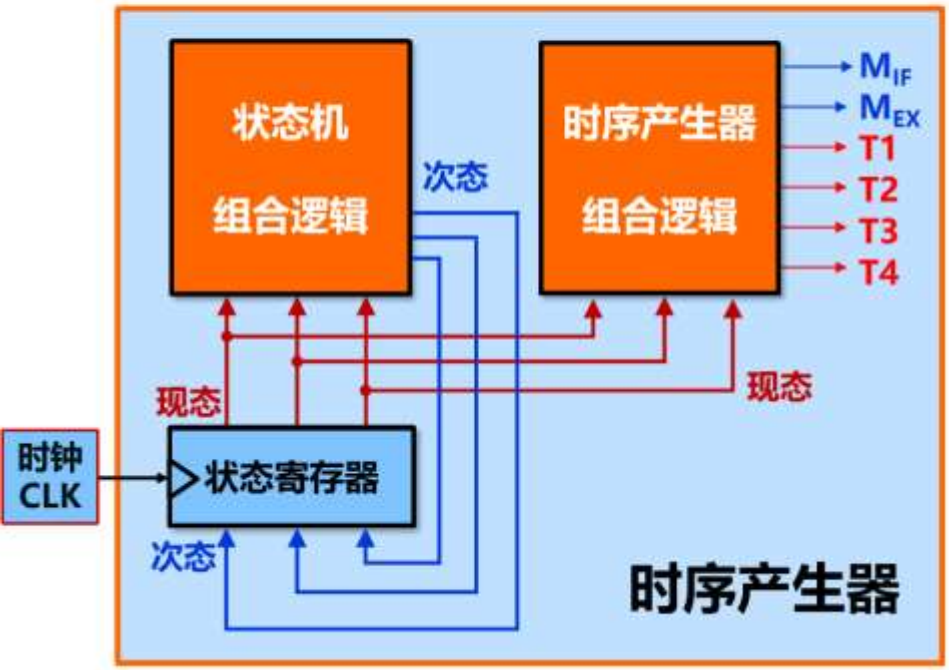
固定2个机器周期，8个时钟节拍

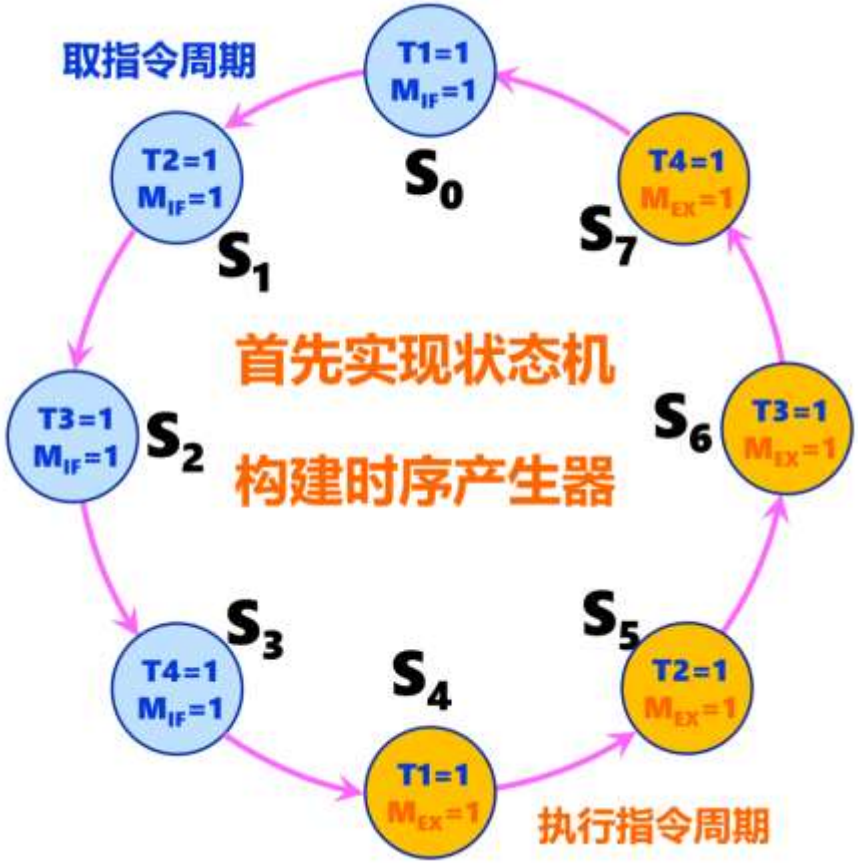


构建时序产生器 输出：M_{IF}，M_{EX}，T1，T2，T3，T4

实例：单总线CPU硬布线设计流程

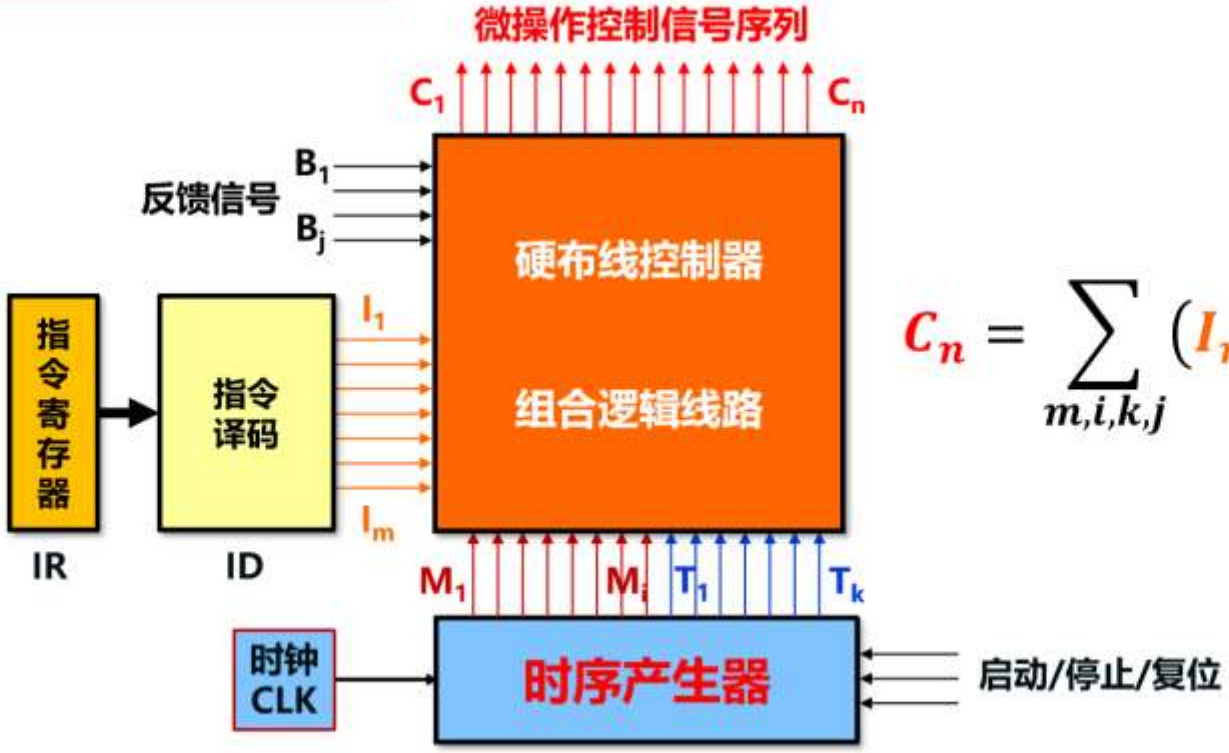
5 时序产生器状态机





实例：单总线CPU硬布线设计流程

6 硬布线控制器基本架构



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

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

实例：单总线CPU硬布线设计流程

7 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}

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

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

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}, $R0_{in}$				

实例：单总线CPU硬布线设计流程

8 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。

$$C_n = \sum_i \left(M_i \cdot T_k \cdot B_j \cdot \sum_m I_m \right)$$

单选题 1分

设置

【2016统考真题】单周期处理器中所有指令的指令周期为一个时钟周期。下列关于单周期处理器的叙述中，错误的是(　)。

A 可以采用单总线结构数据通路

B 处理器时钟频率较低

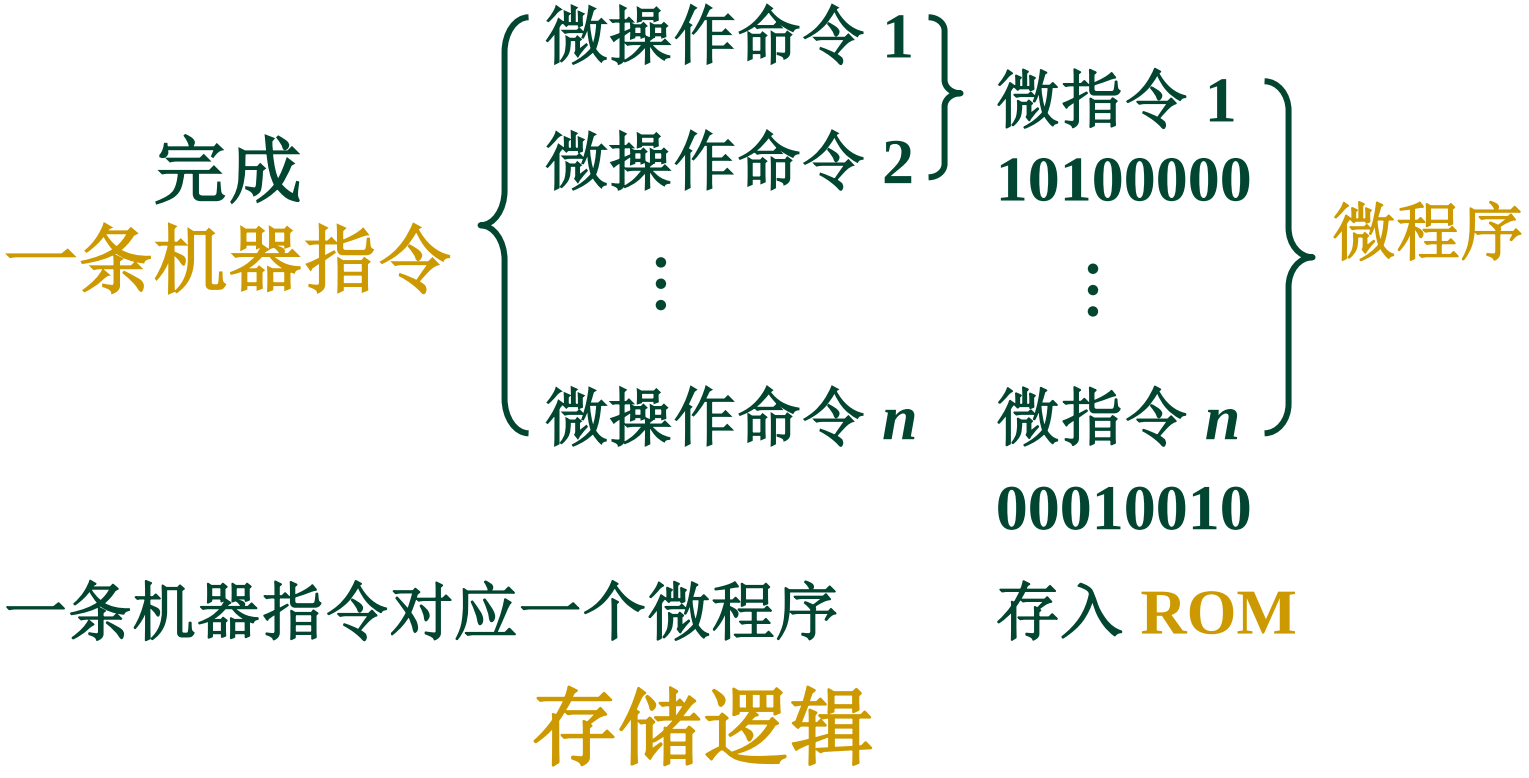
C 在指令执行过程中控制信号不变

D 每条指令的CPI为1

提交

一、微程序设计思想的产生

1951 英国剑桥大学教授 Wilkes



存储逻辑

微指令的格式

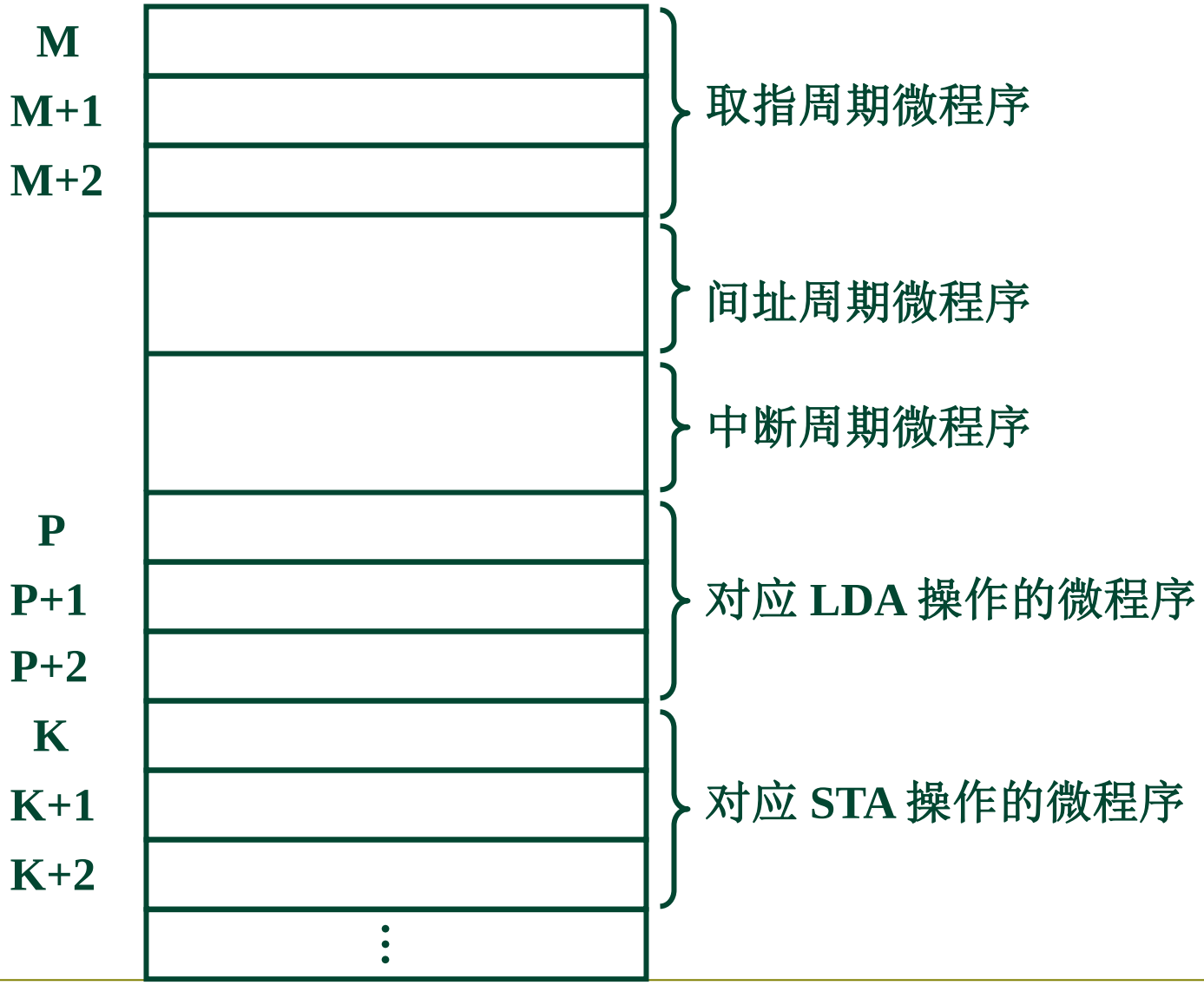
微指令：

控制字段	下址字段

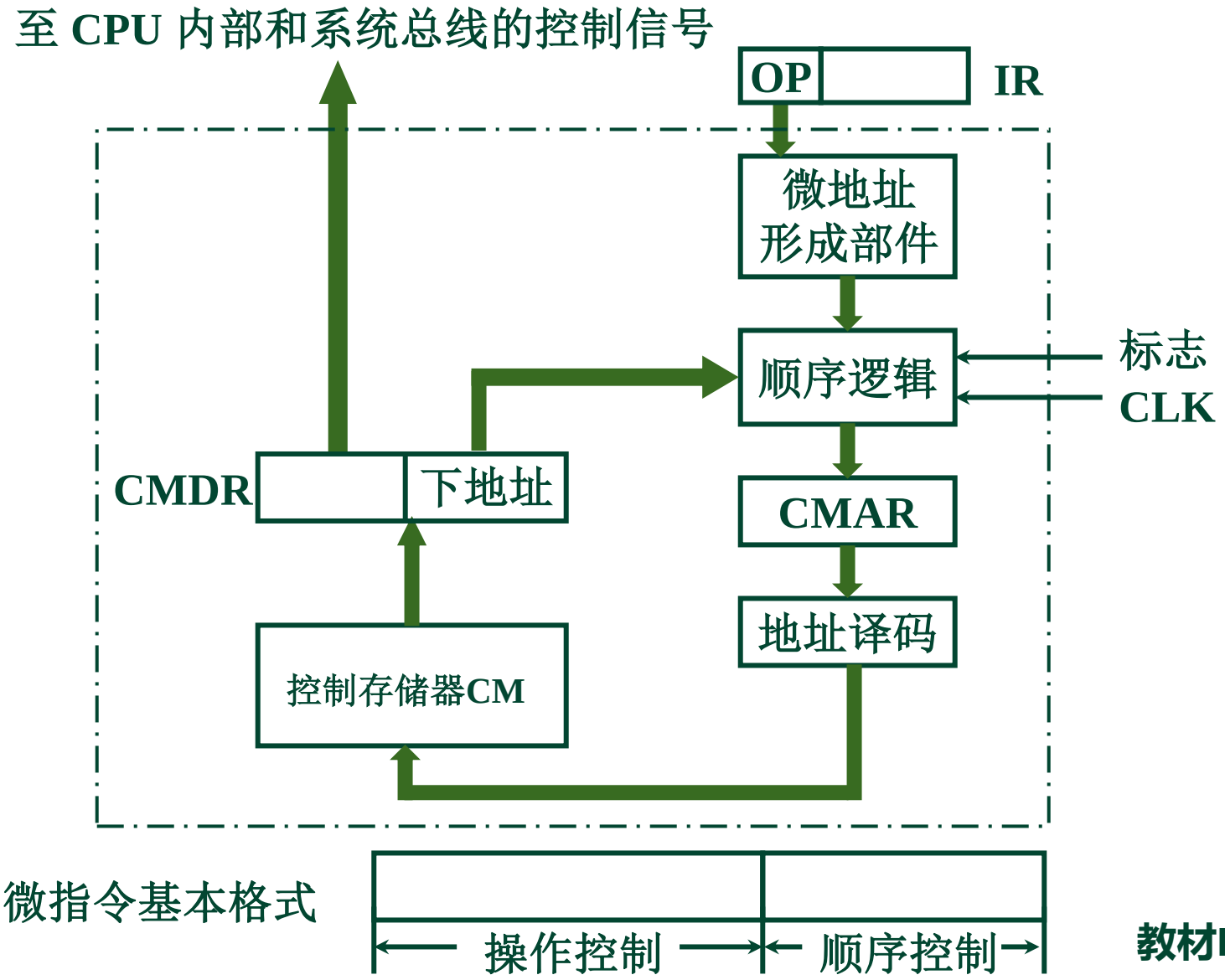
- 控制字段：操作控制，发出各种控制信号
- 下址字段：顺序控制，指出下条微指令地址，以控制微指令序列的执行顺序

二、微程序控制单元框图及工作原理

1. 机器指令对应的微程序

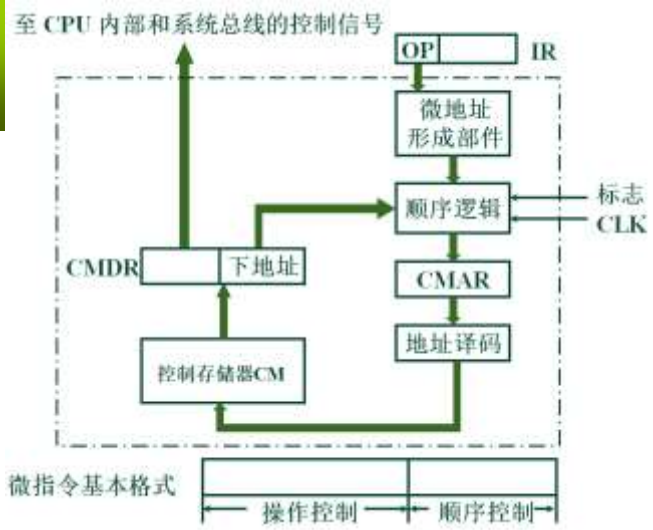


2. 微程序控制单元的基本框图



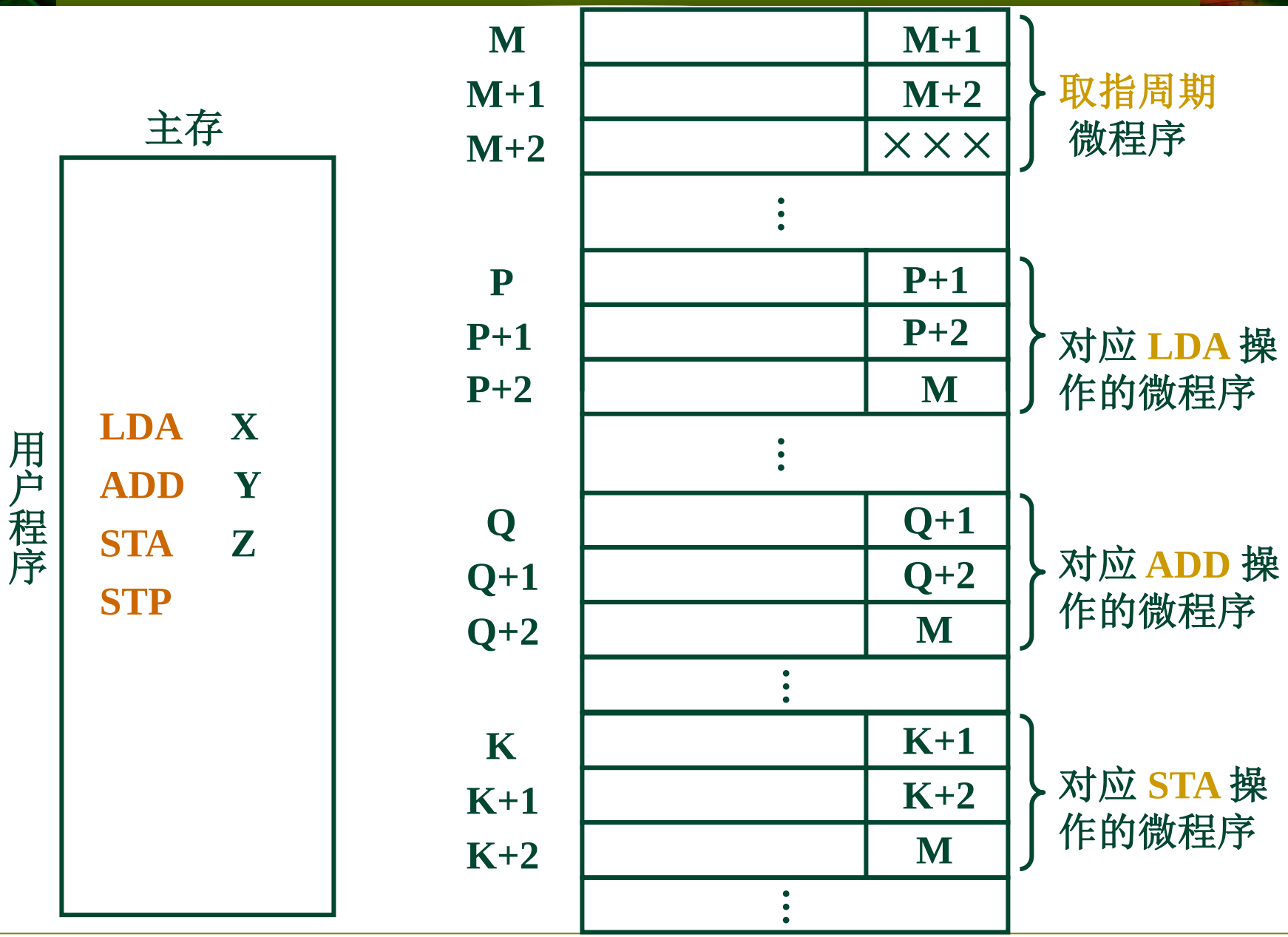
教材P405

二、微程序控制单元框图及工作原理

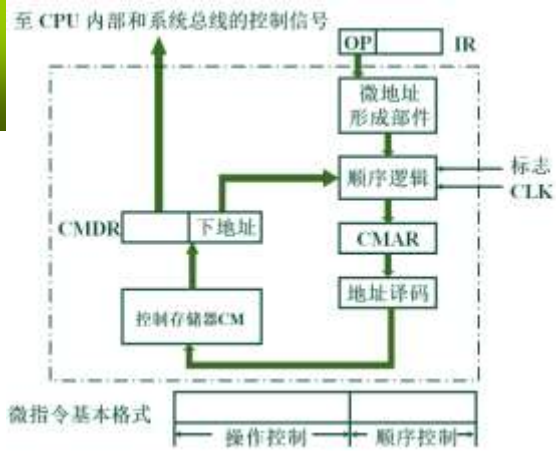


M	M+1	取指周期 微程序
M+1	M+2	
M+2	×××	
	转执行周期微程序	间址周期 微程序
	转取指周期微程序	中断周期 微程序
P	P+1	对应 LDA 操作的微程序
P+1	P+2	
P+2	M	
K	K+1	对应 STA 操作的微程序
K+1	K+2	
K+2	M	
	⋮	

3. 工作原理


控存
主存
用户程序
LDA X
ADD Y
STA Z
STP
M
M+1
M+2
M+1
M+2
×××
取指周期微程序
P
P+1
P+2
P+1
P+2
M
对应 LDA 操作的微程序
Q
Q+1
Q+2
Q+1
Q+2
M
对应 ADD 操作的微程序
K
K+1
K+2
K+1
K+2
M
对应 STA 操作的微程序

3. 工作原理



(1) 取指阶段　　执行 取指周期微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令（控制信号）

形成下条微指令地址　M + 1

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令（控制信号）

形成下条微指令地址　M + 2

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令（控制信号）

(2) 执行阶段　执行 LDA 微程序

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(P ⟶ CMAR)

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 P + 1

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 P + 2

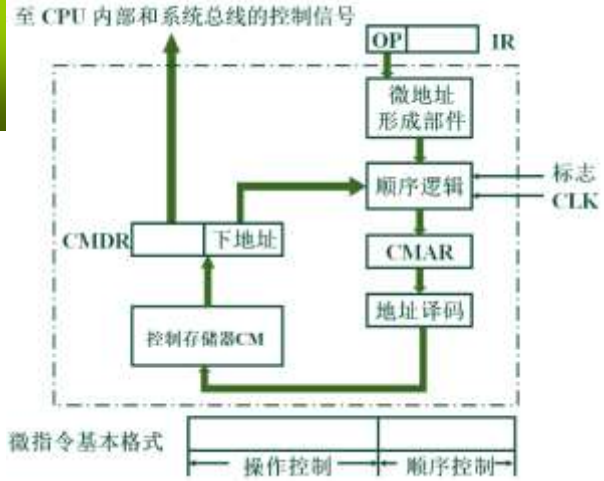
Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 M

Ad (CMDR) ⟶ CMAR



Ad (IR) ⟶ MAR　　1 ⟶ R

M (MAR) ⟶ MDR

MDR ⟶ AC

(M ⟶ CMAR)

(3) 取指阶段　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

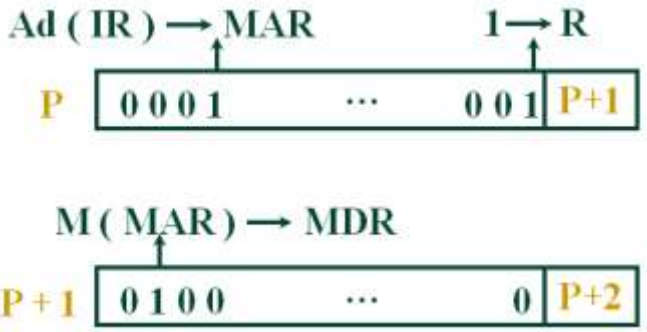
⋮

全部微指令存在 CM 中，程序执行过程中 只需读出

关键

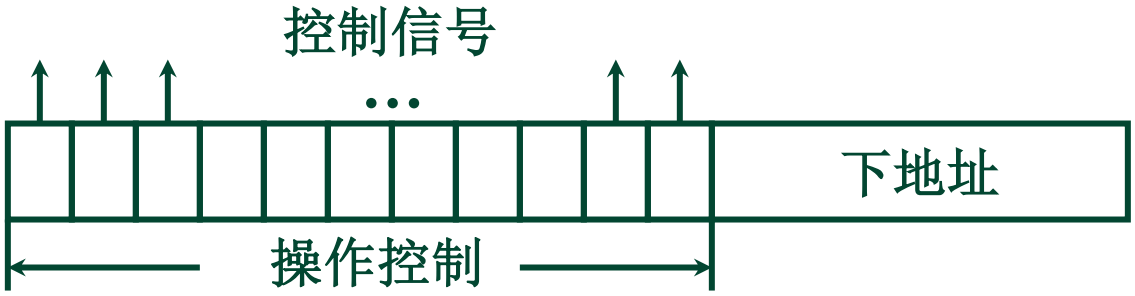
- 微指令的 操作控制字段如何形成微操作命令
- 微指令的 后续地址如何形成

1. 直接编码（直接控制）方式



在微指令的操作控制字段中，

每一位代表一个微操作命令



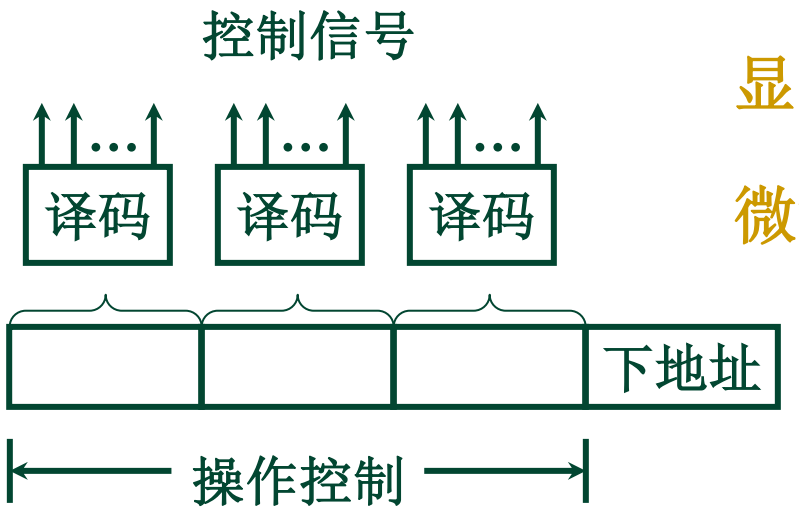
速度最快

某位为“1”表示该控制信号有效

2.字段直接编码方式

将微指令的控制字段分成若干 “段”，

每段经译码后发出控制信号



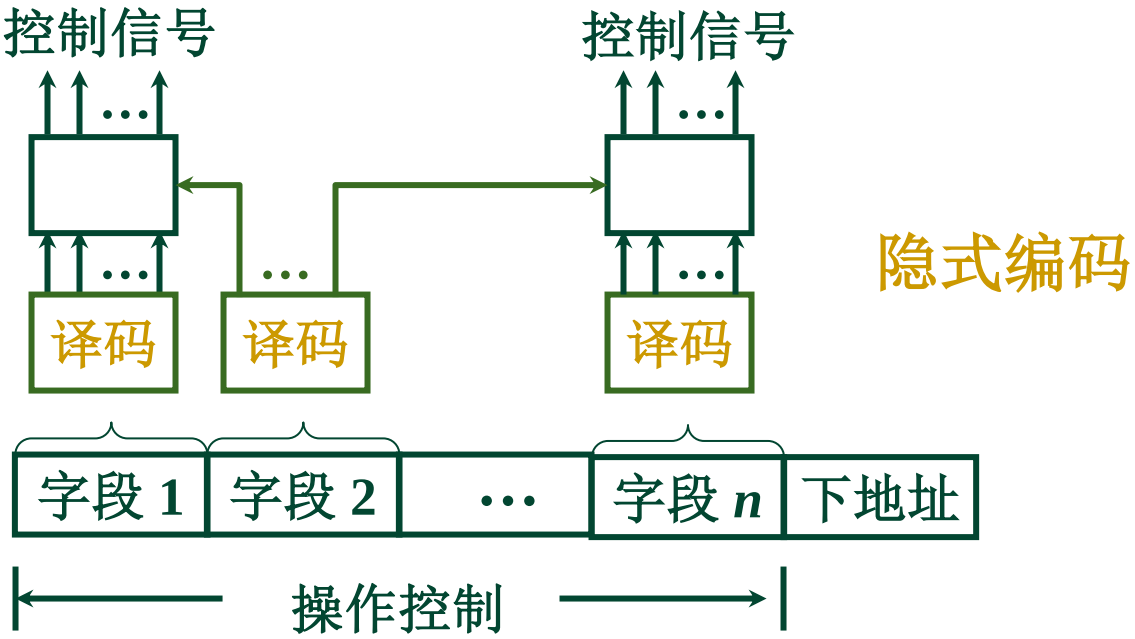
显式编码

微程序执行速度较慢

每个字段中的命令是 互斥 的

缩短 了微指令 字长，增加 了译码 时间

3. 字段间接编码方式



4. 混合编码

直接编码和字段编码（直接和间接）混合使用

5. 其他

四、微指令序列地址的形成

1. 微指令的 下地址字段 指出 （断定方式）

2. 根据机器指令的 操作码 形成：根据机器指令的操作码，由微地址形成部件形成对应该机器指令微程序的首地址。

3. 增量计数器（顺序地址）

$$(CMAR) + 1 \longrightarrow CMAR$$

4. 分支转移（转移指令）

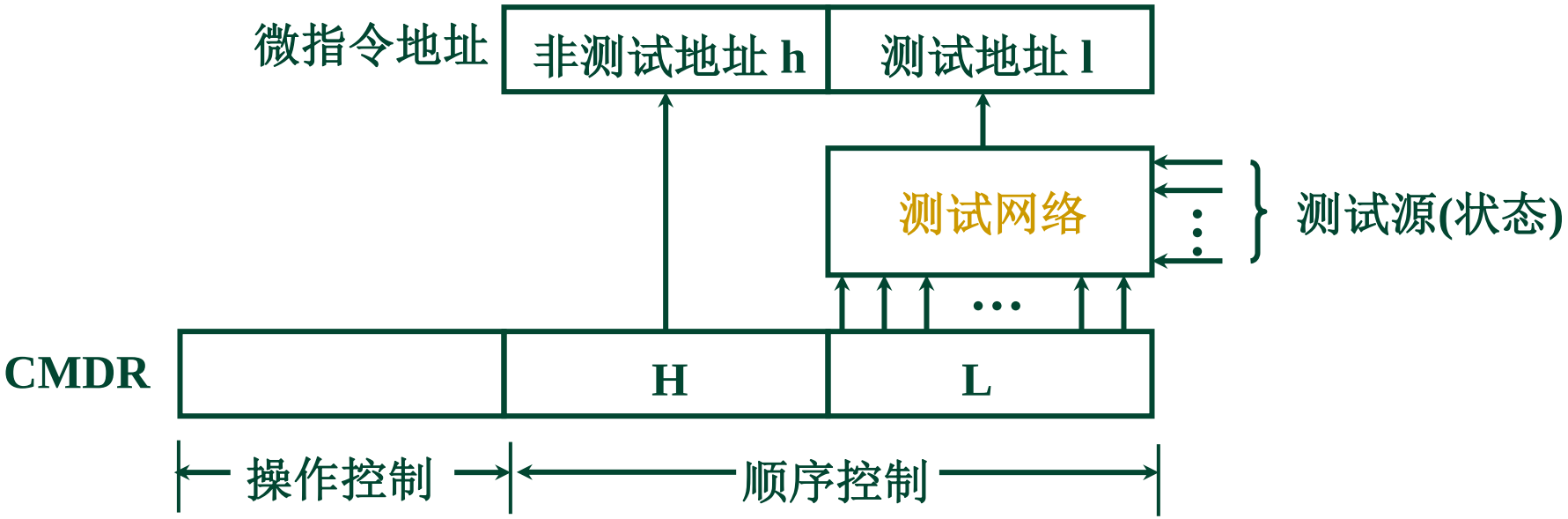
操作控制字段	转移方式	转移地址

转移方式　　指明判别条件

转移地址　　指明转移成功后的去向

四、微指令序列地址的形成

5. 通过测试网络

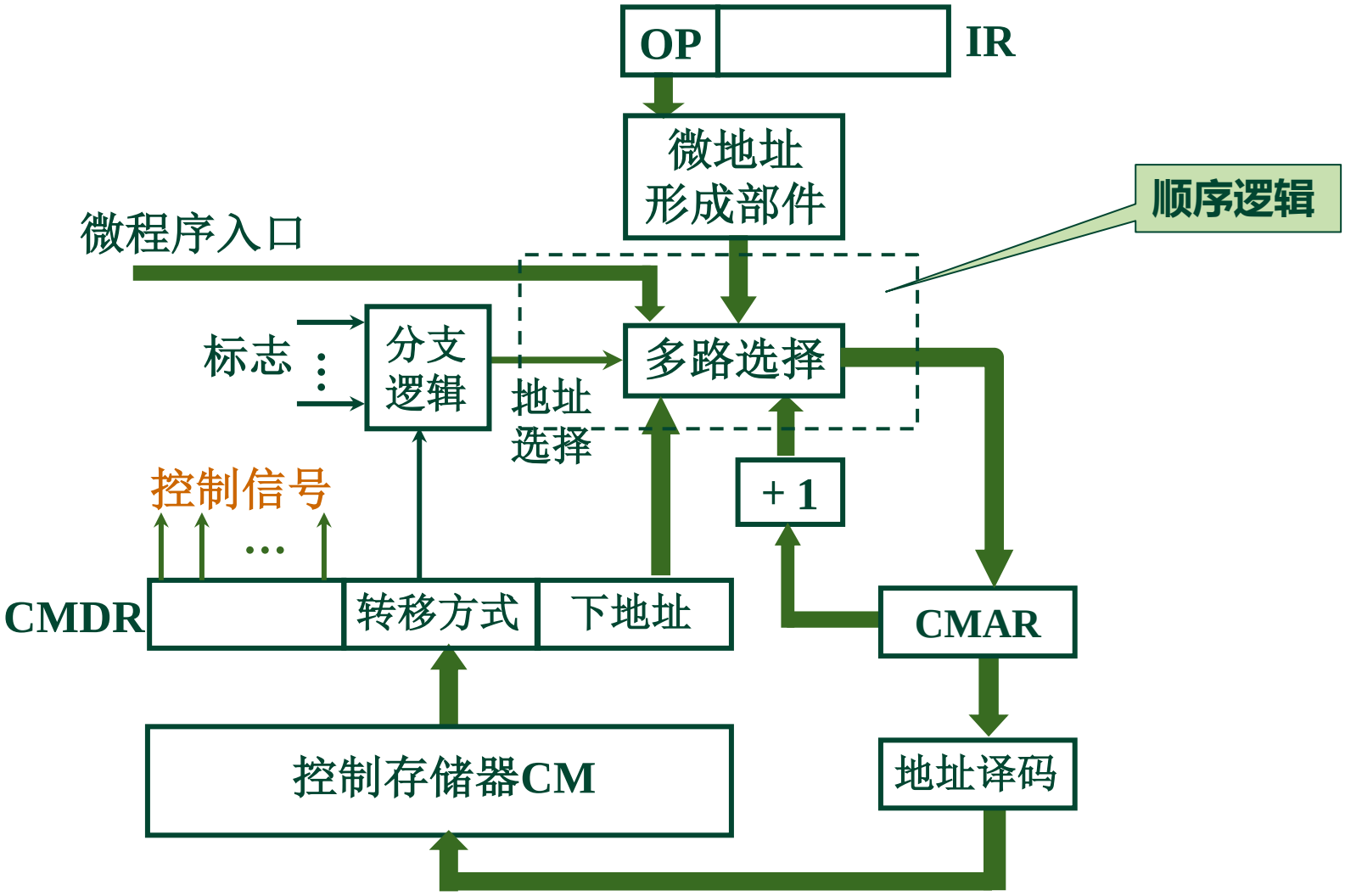


6. 由硬件产生微程序入口地址

加电后，第一条微指令地址　　由专门 硬件 产生

中断周期　　由 硬件 产生 中断周期微程序首地址

7. 后续微指令地址形成方式原理图



五、微指令格式

10.2

1. 水平型微指令　教材P411 P407图10.7

一次能定义并执行多个并行操作

如　直接编码、字段直接编码、字段间接编码、
直接和字段混合编码

2. 垂直型微指令　教材P411 表10.2

类似机器指令操作码 的方式

由微操作码字段规定微指令的功能

3. 两种微指令格式的比较

(1) 水平型微指令比垂直型微指令 并行操作能力强，灵活性强

(2) 水平型微指令执行一条机器指令所要的微指令 数目少，速度快

(3) 水平型微指令 用较短的微程序结构换取较长的微指令结构

(4) 水平型微指令与机器指令 差别大

单选题 1分

设置

【2012统考真题】某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用字段直接编码法，共有33个微命令，构成5个互斥类，分别包含7、3、12、5和6个微命令，则操作控制字段至少有(　)。

A 5位

B 6位

C 15位

D 33位

提交

六、静态微程序设计和动态微程序设计

静态　微程序无须改变，采用 **ROM**

动态　通过 改变微指令 和 微程序 改变机器指令，
有利于仿真，采用 **EPROM**

七、毫微程序设计

1. 毫微程序设计的基本概念

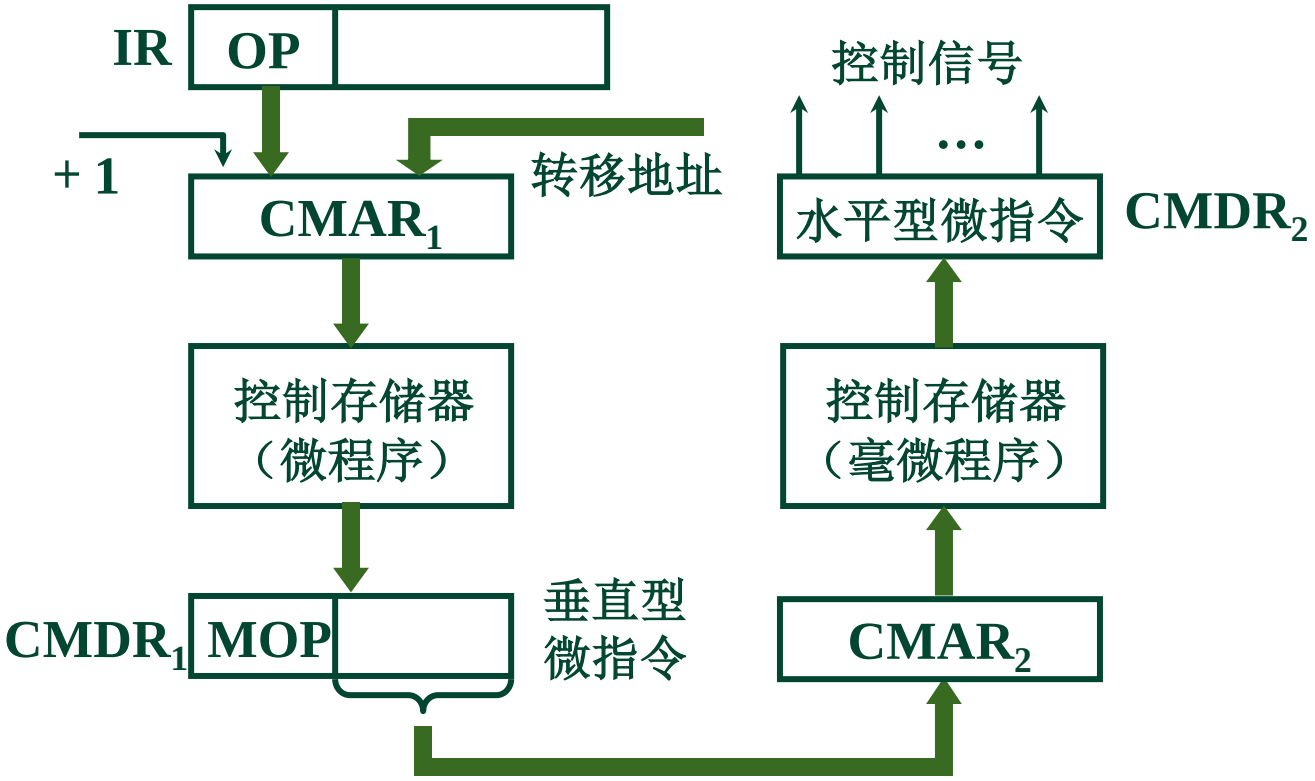
微程序设计　　用 微程序解释机器指令

毫微程序设计　用 毫微程序解释微程序

毫微指令与微指令 的关系好比 微指令与机器指令 的关系

2. 毫微程序控制存储器的基本组成

10.2



串行 微程序控制

取第 *i* 条微指令	执行第 *i* 条微指令	取第 *i*+1 条微指令	执行第 *i*+1 条微指令

并行 微程序控制

取第 *i* 条微指令	执行第 *i* 条微指令		
	取第 *i*+1 条微指令	执行第 *i*+1 条微指令	
		取第 *i*+2 条微指令	执行第 *i*+2 条微指令

九、微程序设计举例

10.2

1. 写出对应机器指令的微操作及节拍安排

假设 CPU 结构与组合逻辑相同（非总线方式）

(1) 取指阶段微操作分析　　3 条微指令

T_0　PC ⟶ MAR　　1 ⟶ R

T_1　M (MAR) ⟶ MDR　　(PC) + 1 ⟶ PC

T_2　MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

还需考虑 如何读出 这 3 条微指令？

Ad (CMDR) ⟶ CMAR

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

教材P415

(2) 取指阶段的微操作及节拍安排

考虑到需要 形成后续微指令的地址

T_0 PC ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR　(PC)+1 ⟶ PC

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

T_5 OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(3) 执行阶段的微操作及节拍安排

考虑到需形成后续微指令的地址

取指微程序的入口地址 M
由微指令下地址字段指出

- 非访存指令

① CLA 指令

T_0　0 ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

② COM 指令

T_0　$\overline{\text{AC}}$ ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

③ SHR 指令

T_0　　L (AC) ⟶ R (AC)　　AC_0 ⟶ AC_0

T_1　　Ad (CMDR) ⟶ CMAR

④ CSL 指令

T_0　　R (AC) ⟶ L (AC)　　AC_0 ⟶ AC_n

T_1　　Ad (CMDR) ⟶ CMAR

⑤ STP 指令

T_0　　0 ⟶ G

T_1　　Ad (CMDR) ⟶ CMAR

• 访存指令

⑥ ADD 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 (AC) + (MDR) ⟶ AC

T_5 Ad (CMDR) ⟶ CMAR

⑦ STA 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ W

T_1 Ad (CMDR) ⟶ CMAR

T_2 AC ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ M (MAR)

T_5 Ad (CMDR) ⟶ CMAR

⑧ LDA 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ AC

T_5 Ad (CMDR) ⟶ CMAR

• 转移类指令

⑨ JMP 指令

T_0　　Ad (IR) ⟶ PC

T_1　　Ad (CMDR) ⟶ CMAR

⑩ BAN 指令

T_0　　$A_0 \cdot Ad(IR) + \overline{A_0} \cdot (PC) \longrightarrow PC$

T_1　　Ad (CMDR) ⟶ CMAR

全部微操作　20个

微指令　　38条

2. 确定微指令格式

(1) 微指令的编码方式

采用直接控制

(2) 后续微指令的地址形成方式

由机器指令的操作码通过微地址形成部件形成

由微指令的下地址字段直接给出

(3) 微指令字长

由 20 个微操作

确定 操作控制字段　　最少 20 位

由 38 条微指令

确定微指令的 下地址字段 为 6 位

微指令字长 可取 20 ＋ 6 ＝ 26 位

(4) 微指令字长的确定

38 条微指令中有 19 条

是用于形成后续微指令地址 ⟶ CMAR

其中 { 1 条　OP (IR) ⟶ 微地址形成部件 ⟶ CMAR
　　 { 18 条　Ad (CMDR) ⟶ CMAR

若用 Ad (CMDR) 直接送控存地址线

则 省去了输至 CMAR 的时间，省去了 CMAR　教材P417

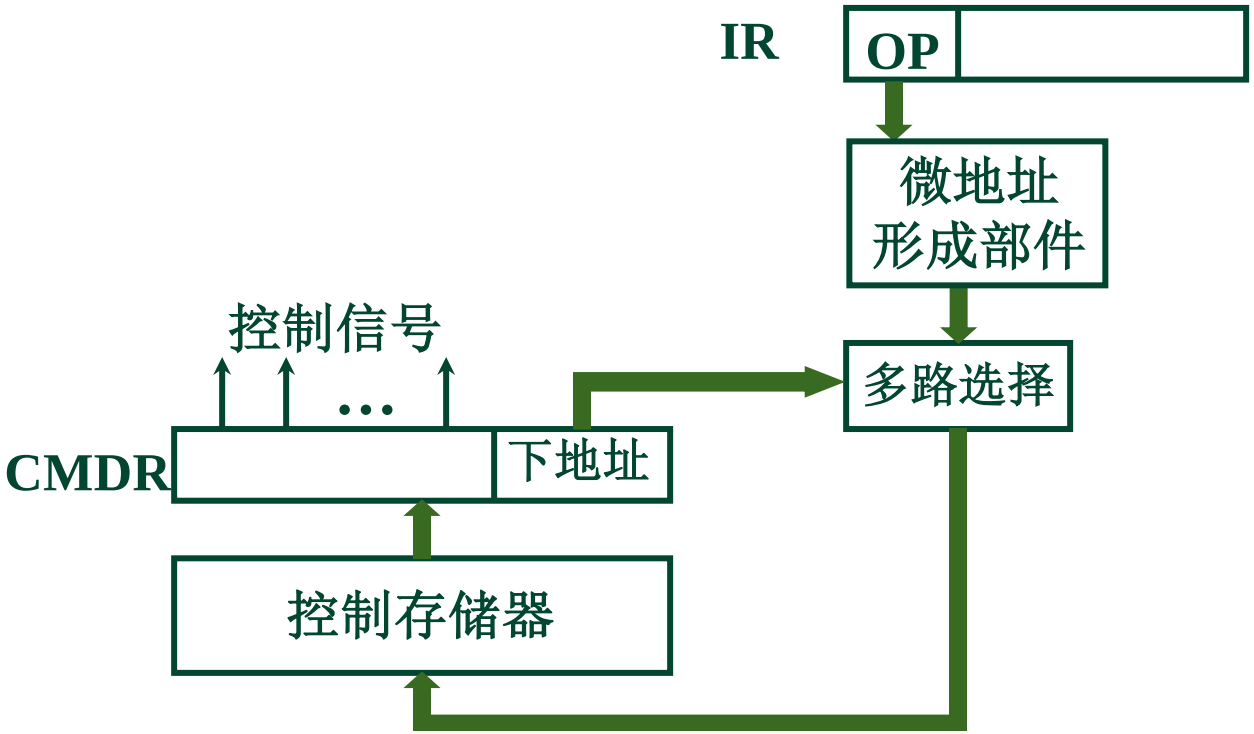
同理　OP (IR) ⟶ 微地址形成部件 ⟶ 控存地址线　教材P418 图10.16

可省去 19 条微指令，2 个微操作

38 － 19 ＝ 19　　20 － 2 ＝ 18

下地址字段最少取 5 位　　操作控制字段最少取 18 位

(5) 省去了 CMAR 的控制存储器



考虑留有一定的余量

取操作控制字段　18 位→24 位
下地址字段　5 位→ 6 位
} 共 30 位

(6) 定义微指令操作控制字段每一位的微操作

3. 编写微指令码点

微程序名称	微指令地址(八进制)	微指令（二进制代码）														
		操作控制字段									下地址字段					
		0	1	2	3	4	…	10	…	23	24	25	26	27	28	29
取指	00	1	1								0	0	0	0	0	1
	01			1	1						0	0	0		1	0
	02					1					×	×	×	×	×	×
CLA	03										0	0	0	0	0	0
COM	04										0	0	0	0	0	0
ADD	10		1					1			0	0	1	0	0	1
	11			1							0	0	1	0	1	0
	12										0	0	0	0	0	0
LDA	16		1					1			0	0	1	1	1	1
	17			1							0	1	0	0	0	0
	20										0	0	0	0	0	0

PC→MAR

M(MAR)→MDR

1→R

MDR→IR

PC+1→PC

Ad(IR)→MAR

1→R

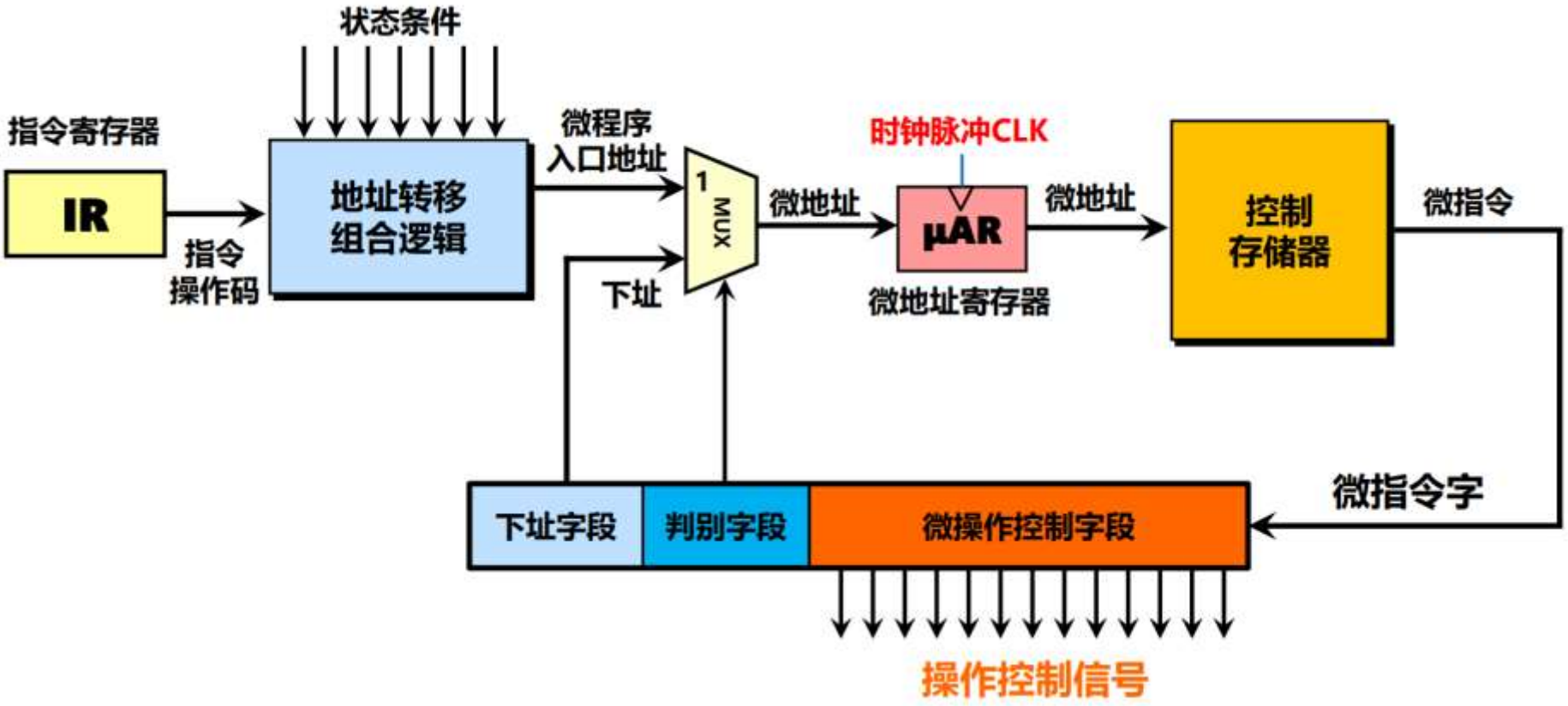
1→R

Ad(IR)→MAR

M(MAR)→MDR

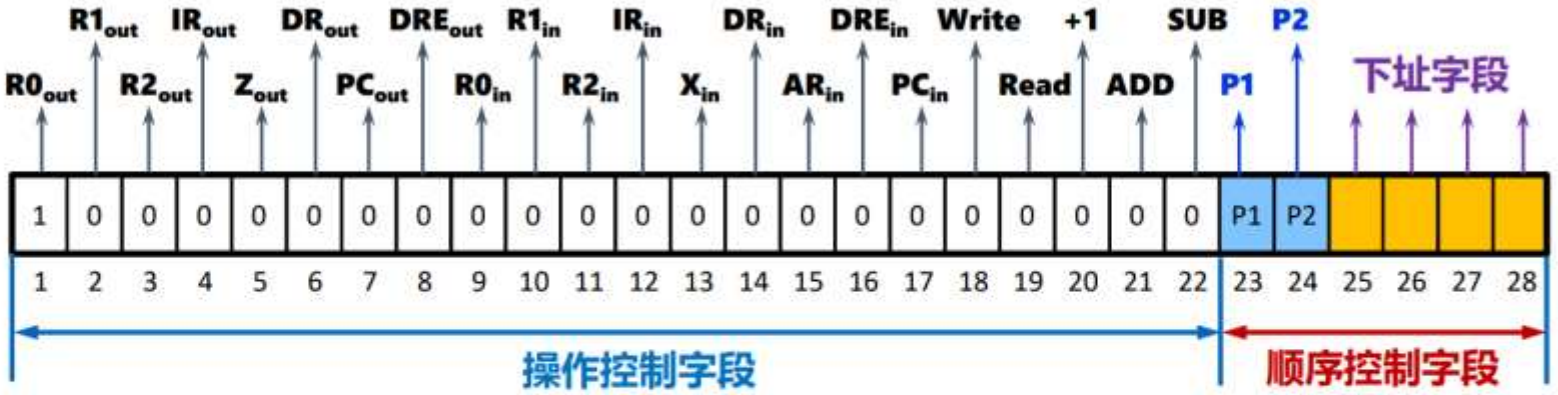
实例：微程序设计-单总线CPU

1 微程序控制器组成原理框图



实例：微程序设计-单总线CPU

2 微指令格式



- 一条微指令对应一个时钟周期
- 微指令操作控制字段的信号在该时钟周期内有效
- 指令需要多少时钟周期就包括多少微指令

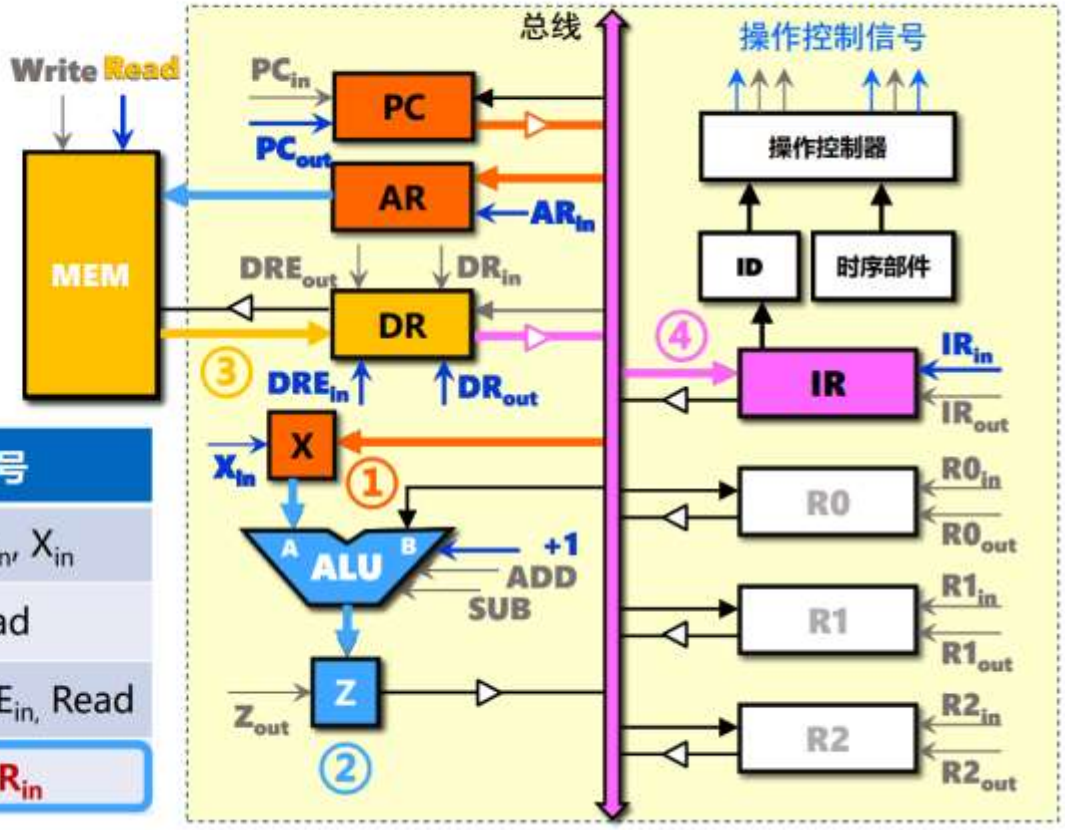
实例：微程序设计-单总线CPU

3 取指令数据通路

Mem[PC++] → IR

- **4个时钟周期**
- **四条微指令**

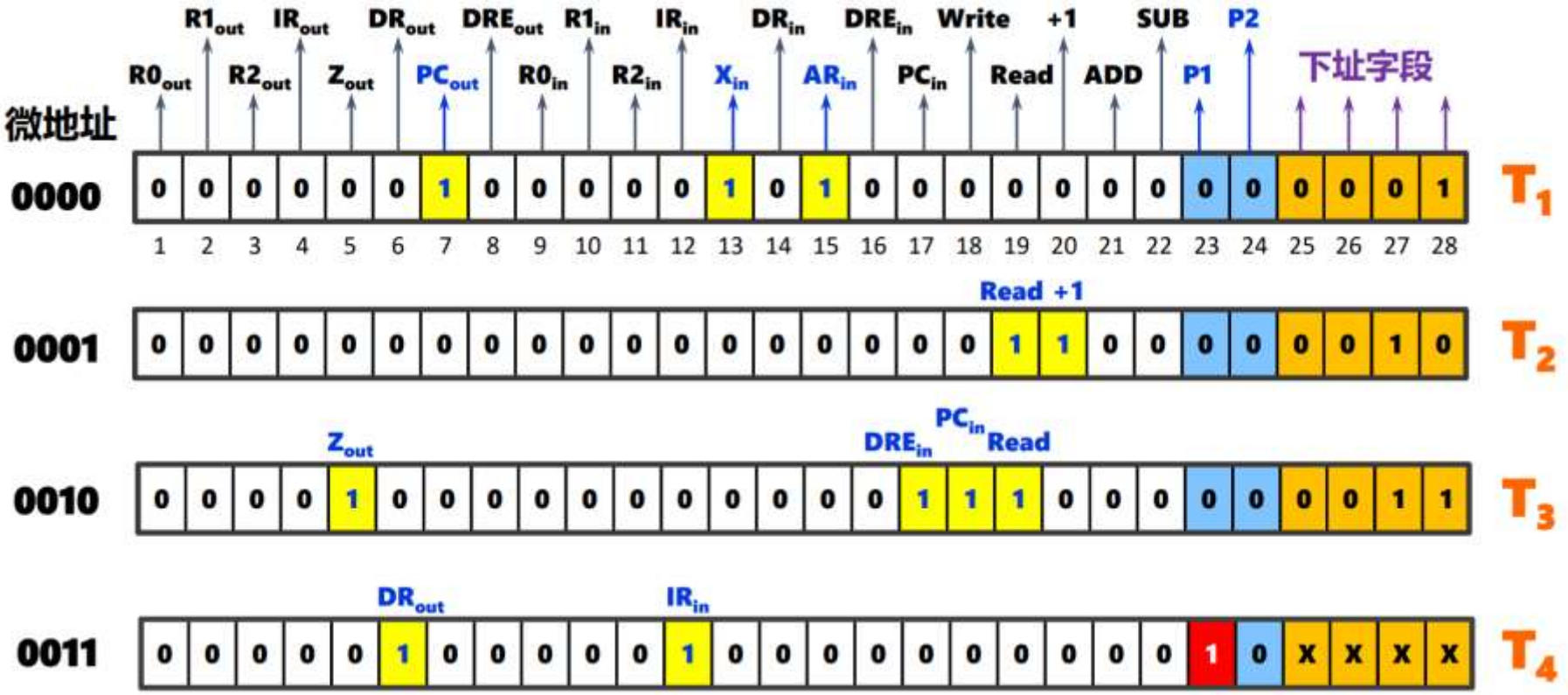
节拍	数据通路	控制信号
T1	(PC)→AR, (PC)→X	PC_{out}, AR_{in}, X_{in}
T2	(X)+1→Z	+1, Read
T3	(Z)→PC, Mem[AR]→DR	Z_{out}, PC_{in}, DRE_{in}, Read
T4	**(DR)→IR**	**DR_{out}, IR_{in}**



实例：微程序设计-单总线CPU

4 取指令微程序

节拍	取指令数据通路	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}



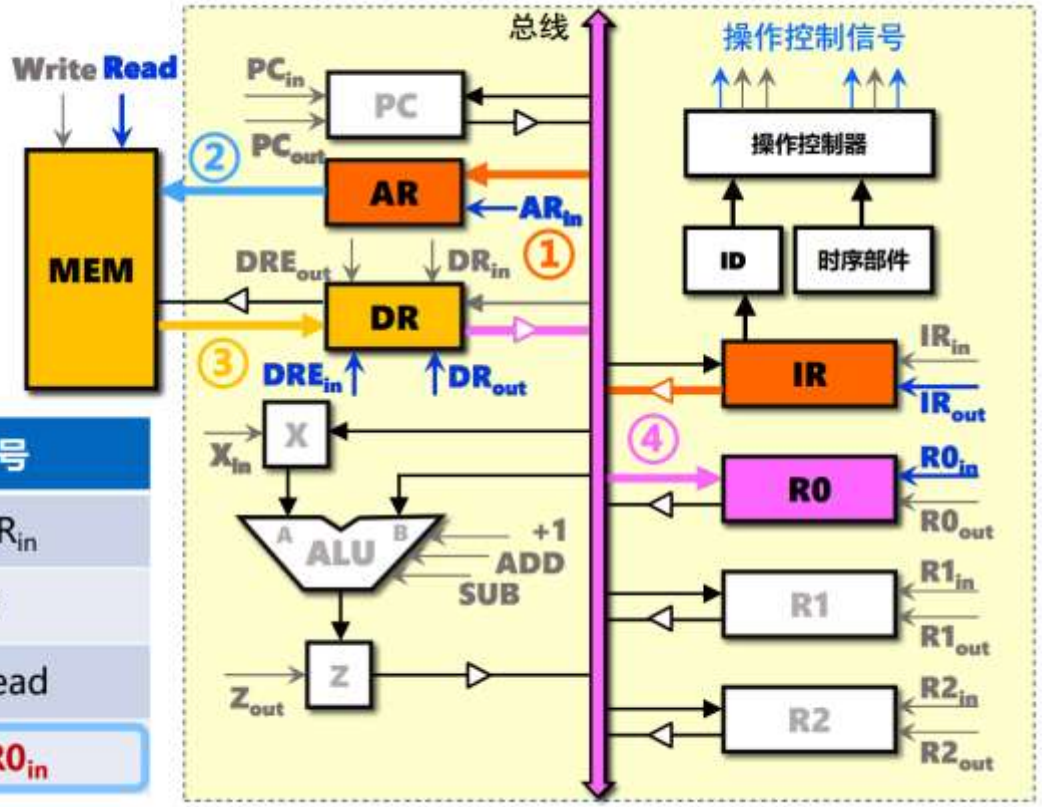
实例：微程序设计-单总线CPU

5 LOAD指令执行数据通路

LOAD R0,6#

Mem[IR_A] → Reg

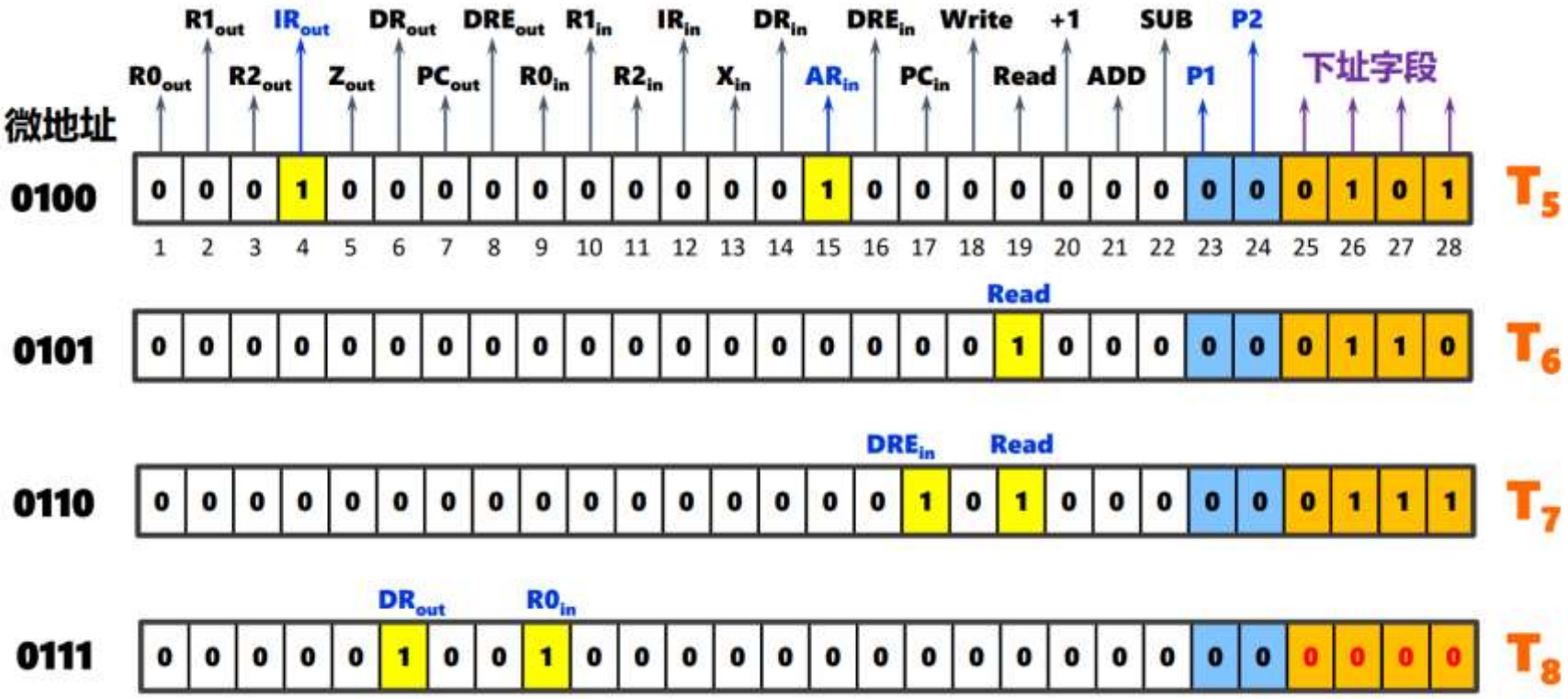
节拍	数据通路	控制信号
T1	(IR_A)→AR	IR_{out}, AR_{in}
T2		Read
T3	Mem[AR]→DR	DRE_{in},Read
T4	**(DR)→R0**	**DR_{out}, $R0_{in}$**



实例：微程序设计-单总线CPU

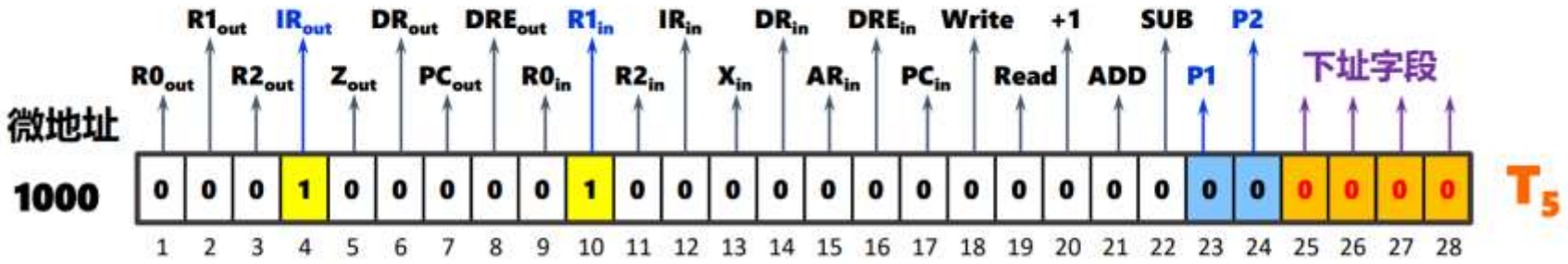
6 LOAD指令微程序

节拍	取指令数据通路	控制信号
T1	$(IR_A)\rightarrow AR$	IR_{out}, AR_{in}
T2		Read
T3	Mem[AR]→DR	DRE_{in}, Read
T4	(DR)→R0	DR_{out}, $R0_{in}$



实例：微程序设计-单总线CPU

7 MOVE指令微程序

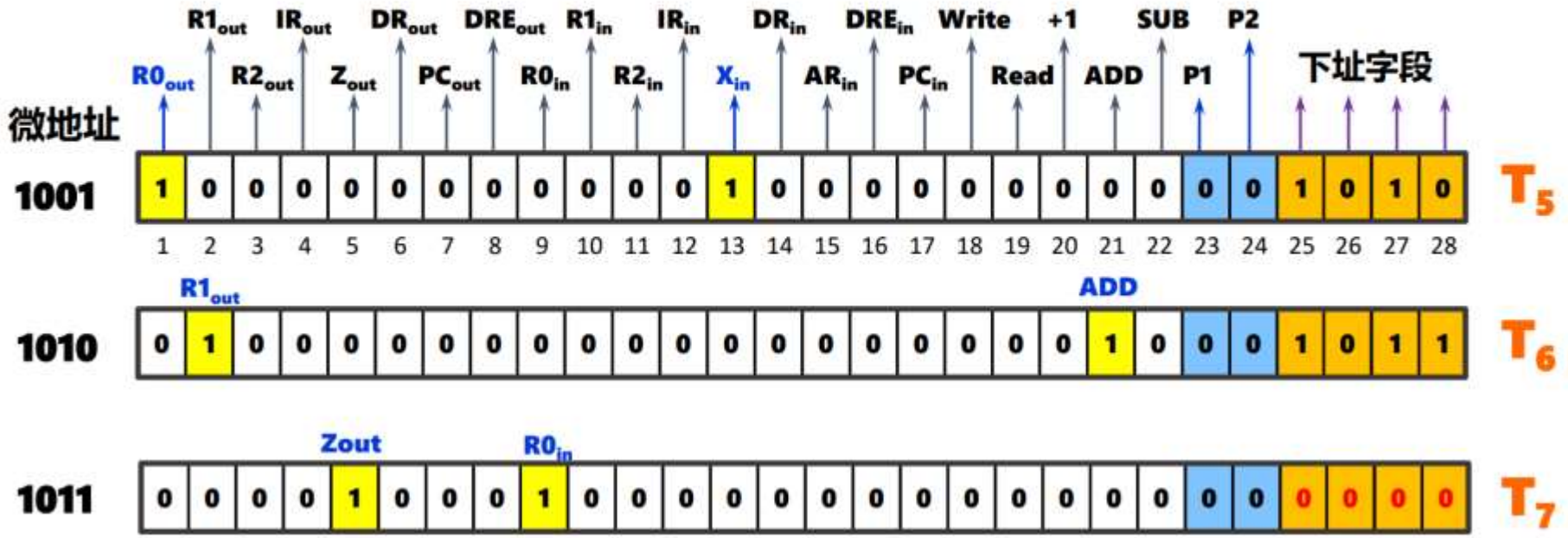


节拍	数据通路	控制信号
T1	$(IR_A) \rightarrow R[0]$	IR_{out}, $R1_{in}$
T2		
T3		
T4		

实例：微程序设计-单总线CPU

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

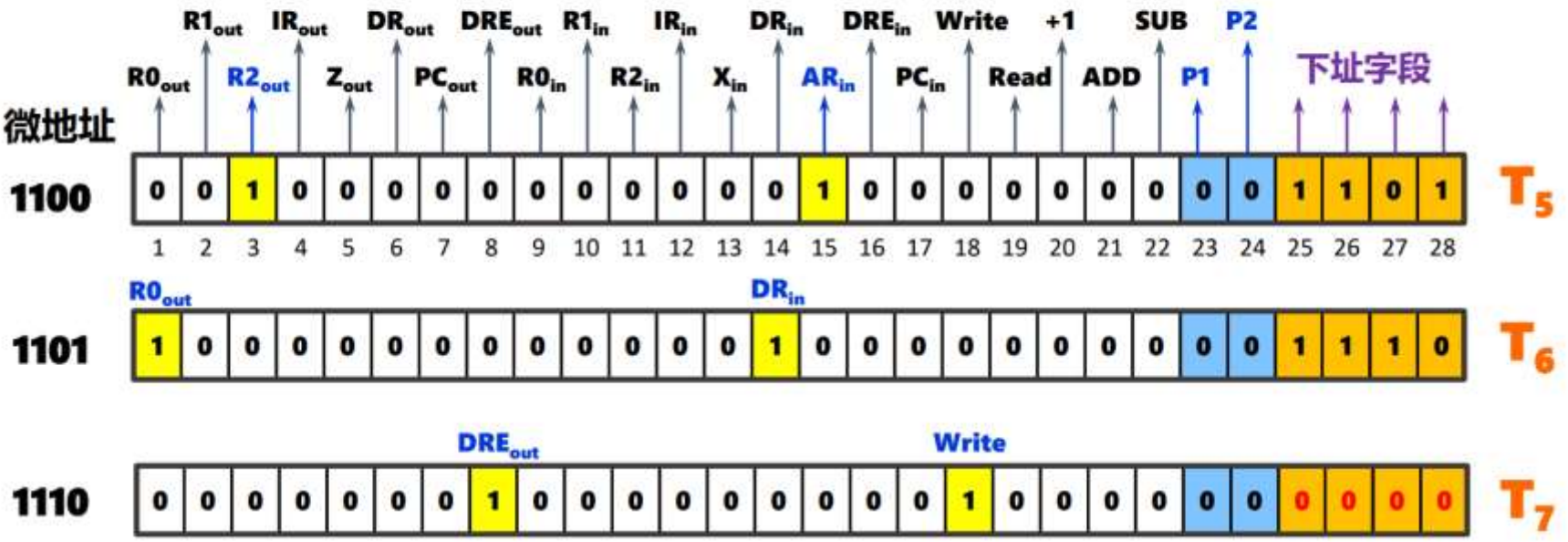
8 ADD指令微程序



实例：微程序设计-单总线CPU

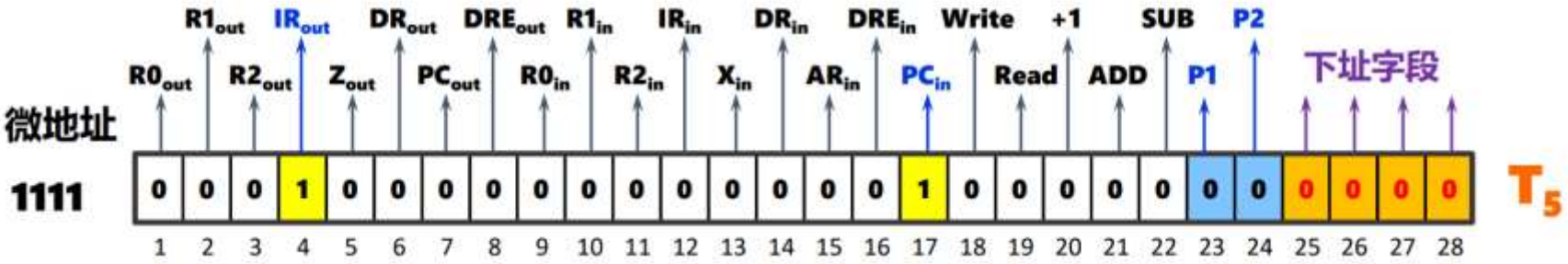
节拍	数据通路	控制信号
T1	(R2)→AR	$R2_{out}$, AR_{in}
T2	(R0)→DR	$R0_{out}$, DR_{in}
T3	(DR) →Mem[AR]	DRE_{out}, Write
T4		

9 STORE指令微程序



实例：微程序设计-单总线CPU

A JMP指令微程序



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

实例：微程序设计-单总线CPU

B 单总线CPU微程序

微地址	操作控制字段																					顺序控制字段						
0000	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0
0010	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1
0011	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	X	X	X	X
0100	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
0101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
0110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1
0111	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1001	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
1010	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1
1011	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1100	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1
1101	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1110	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1111	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

取指令微程序

LOAD微程序

MOVE微程序

ADD 微程序

STORE微程序

JMP 微程序

单选题 1分

设置

【2009统考真题】相对于微程序控制器，硬布线控制器的特点是(　)。

A 指令执行速度慢，指令功能的修改和扩展容易

B 指令执行速度慢，指令功能的修改和扩展难

C 指令执行速度快，指令功能的修改和扩展容易

D 指令执行速度快，指令功能的修改和扩展难

提交

Computer Organization

Thank You !

Computer Architecture Research Institute · Computer Organization