计算机组成原理

第 14、15讲

左德承

哈尔滨工业大学计算学部

容错与移动计算研究中心

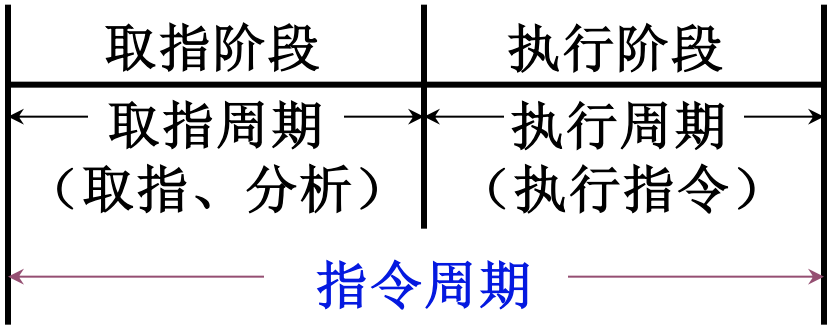
8.2 指 令 周 期

一、 指令周期的基本概念

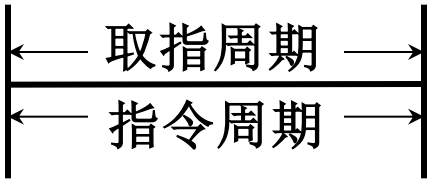
1．指令周期

取出并执行一条指令所需的全部时间

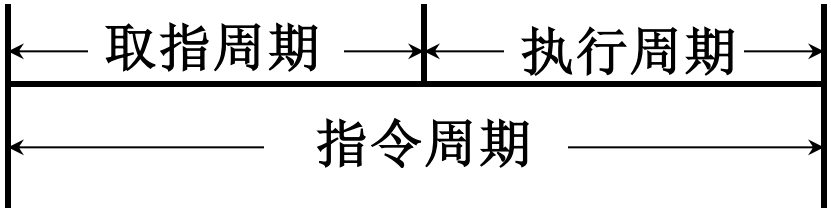
完成一条指令 { 取指、分析 —— 取指周期
执行 —— 执行周期



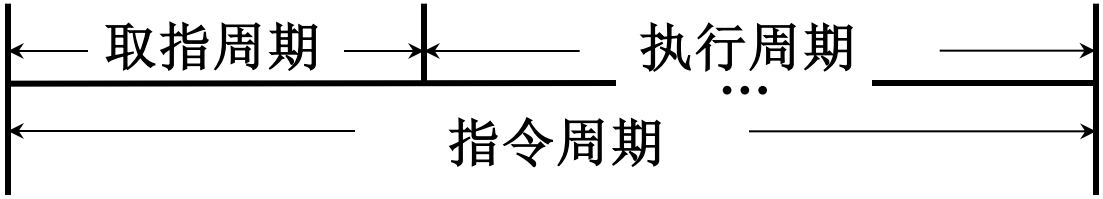
2. 每条指令的指令周期不同



NOP

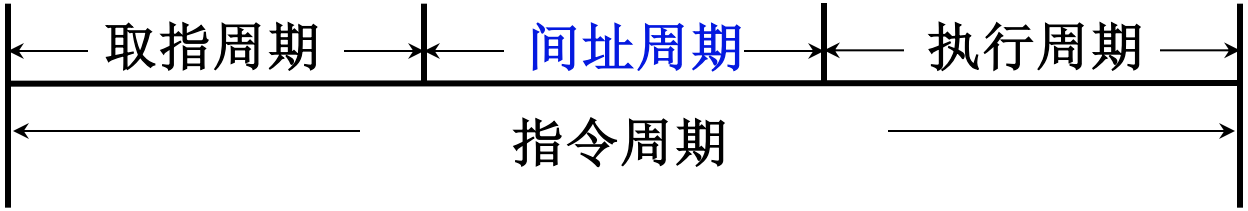


ADD mem

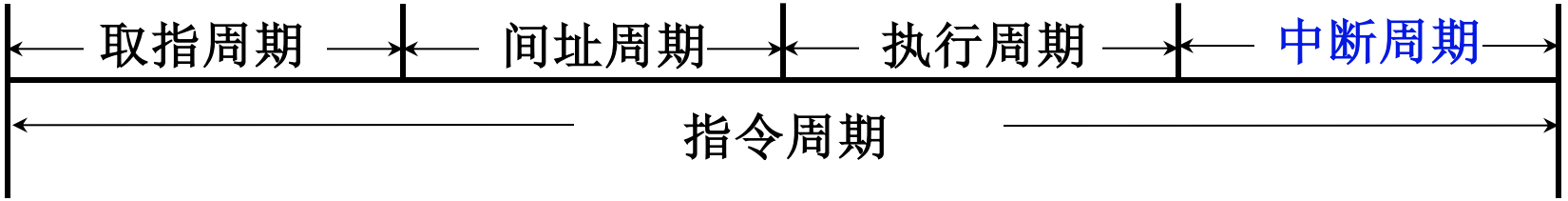


MUL mem

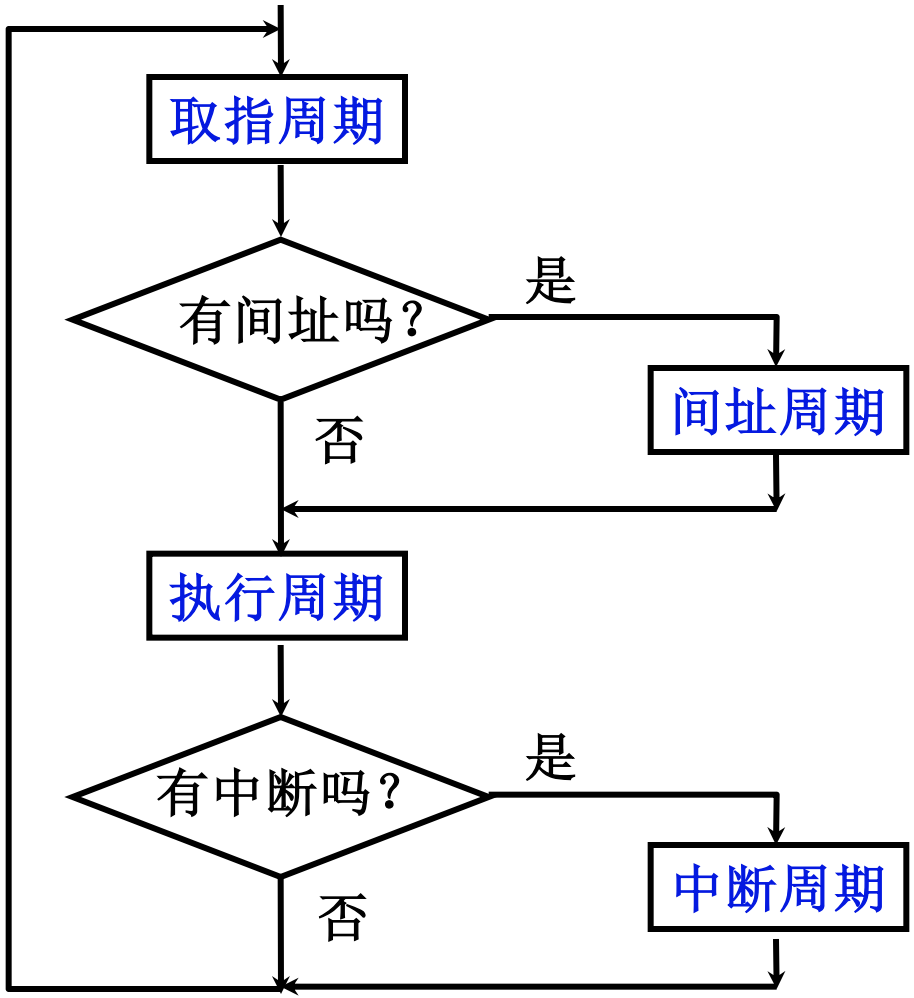
3. 具有间接寻址的指令周期



4. 带有中断周期的指令周期



5. 指令周期流程



6. CPU 工作周期的标志

CPU 访存有四种性质

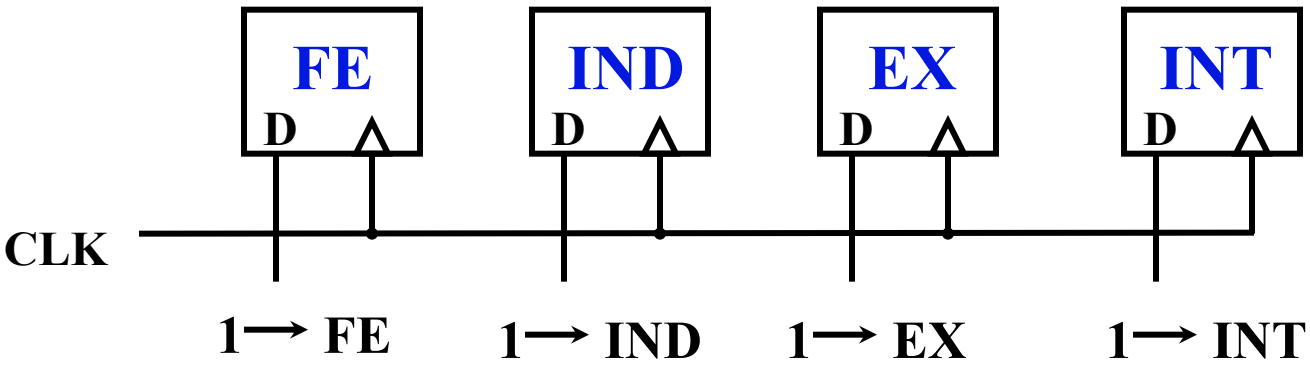
取 指令　　取指周期

取 地址　　间址周期

取 操作数　　执行周期

存 程序断点　　中断周期

CPU 的
4个工作周期



第 9 章　控制单元的功能

9.1　操作命令的分析

9.2　控制单元的功能

9.1 操作命令的分析

完成一条指令分 4 个工作周期

取指周期

间址周期

执行周期

中断周期

9.1 操作命令的分析

一、取指周期

PC ⟶ MAR ⟶ 地址线

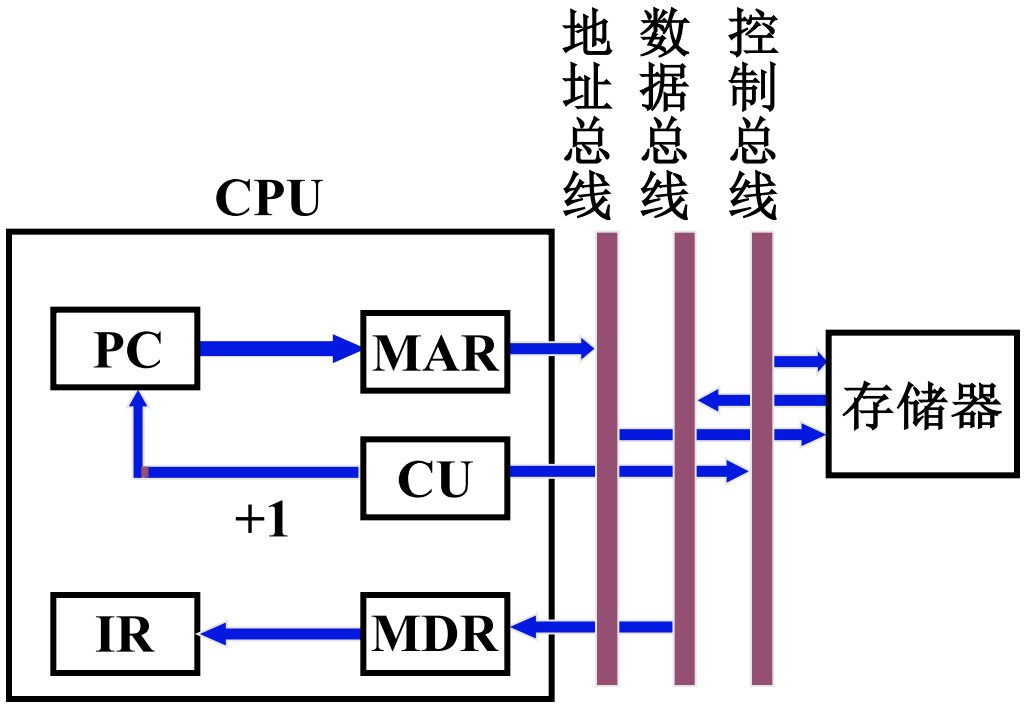
1 ⟶ R

M (MAR) ⟶ MDR

MDR ⟶ IR

OP（IR）⟶ CU

(PC) + 1 ⟶ PC



二、间址周期

9.1

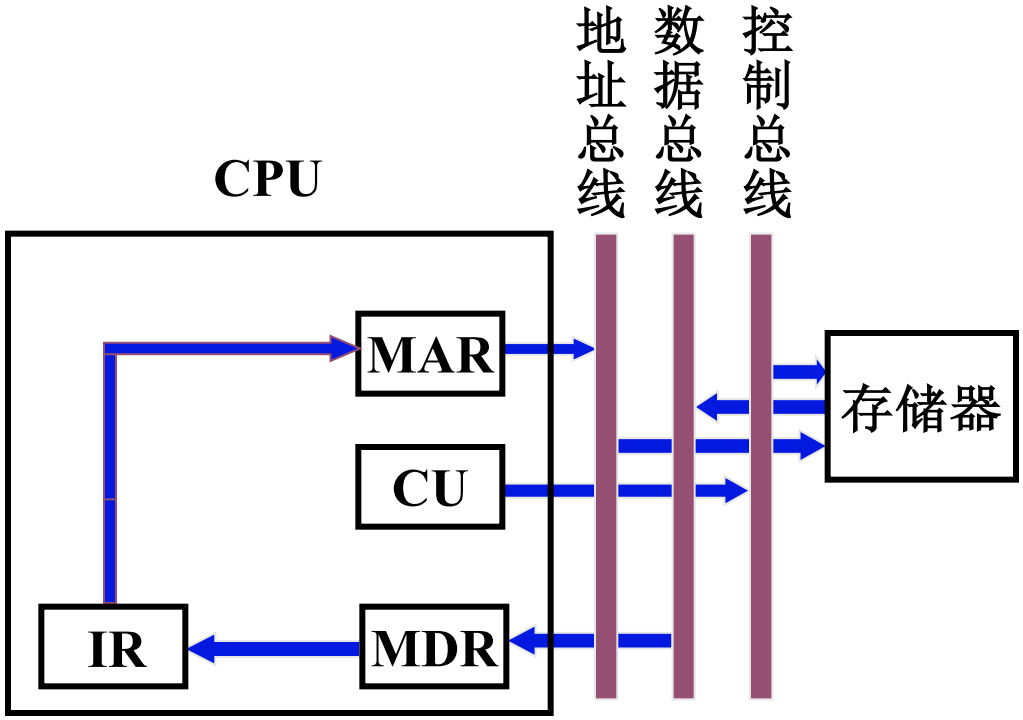
指令形式地址 ⟶ MAR

Ad (IR) ⟶ MAR

1⟶R

M (MAR) ⟶ MDR

MDR⟶ Ad (IR)



三、执行周期

9.1

1. 非访存指令

(1) **CLA** 清A　　$0 \rightarrow ACC$

(2) **COM** 取反　　$\overline{ACC} \rightarrow ACC$

(3) **SHR** 算术右移　　$L(ACC) \rightarrow R(ACC), ACC_0 \rightarrow ACC_0$

(4) **CSL** 循环左移　　$R(ACC) \rightarrow L(ACC), ACC_0 \rightarrow ACC_n$

(5) **STP** 停机指令　　$0 \rightarrow G$

2. 访存指令

(1) 加法指令　**ADD　X**

Ad(IR) ⟶ MAR

1 ⟶ R

M(MAR) ⟶ MDR

(ACC) + (MDR) ⟶ ACC

(2) 存数指令　**STA　X**

Ad(IR) ⟶ MAR

1 ⟶ W

ACC ⟶ MDR

MDR⟶ M(MAR)

(3) 取数指令　LDA　X

Ad (IR) ⟶ MAR

1 ⟶ R

M (MAR) ⟶ MDR

MDR ⟶ ACC

3. 转移指令

(1) 无条件转　JMP　X

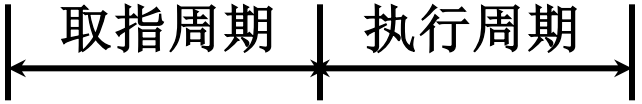
Ad (IR) ⟶ PC

(2) 条件转移　BAN　X　（负则转）

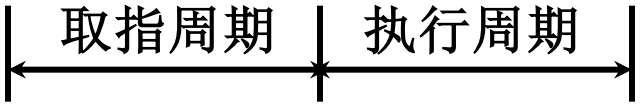
$A_0 \cdot Ad(IR) + \bar{A}_0 (PC) \longrightarrow PC$

4. 三类指令的指令周期

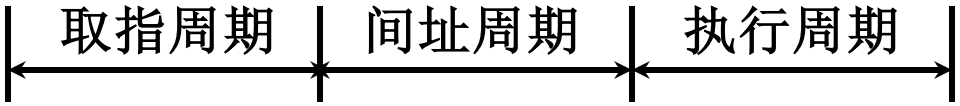
非访存 指令周期



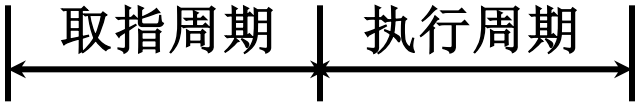
直接访存 指令周期



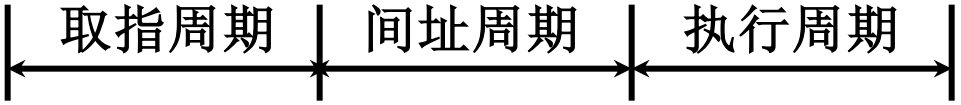
间接访存 指令周期



转移 指令周期



间接转移 指令周期



四、中断周期

9.1

程序断点存入“0”地址　　程序断点进栈

0 → MAR　　(SP) – 1 → MAR

1 → W　　1 → W

PC → MDR　　PC → MDR

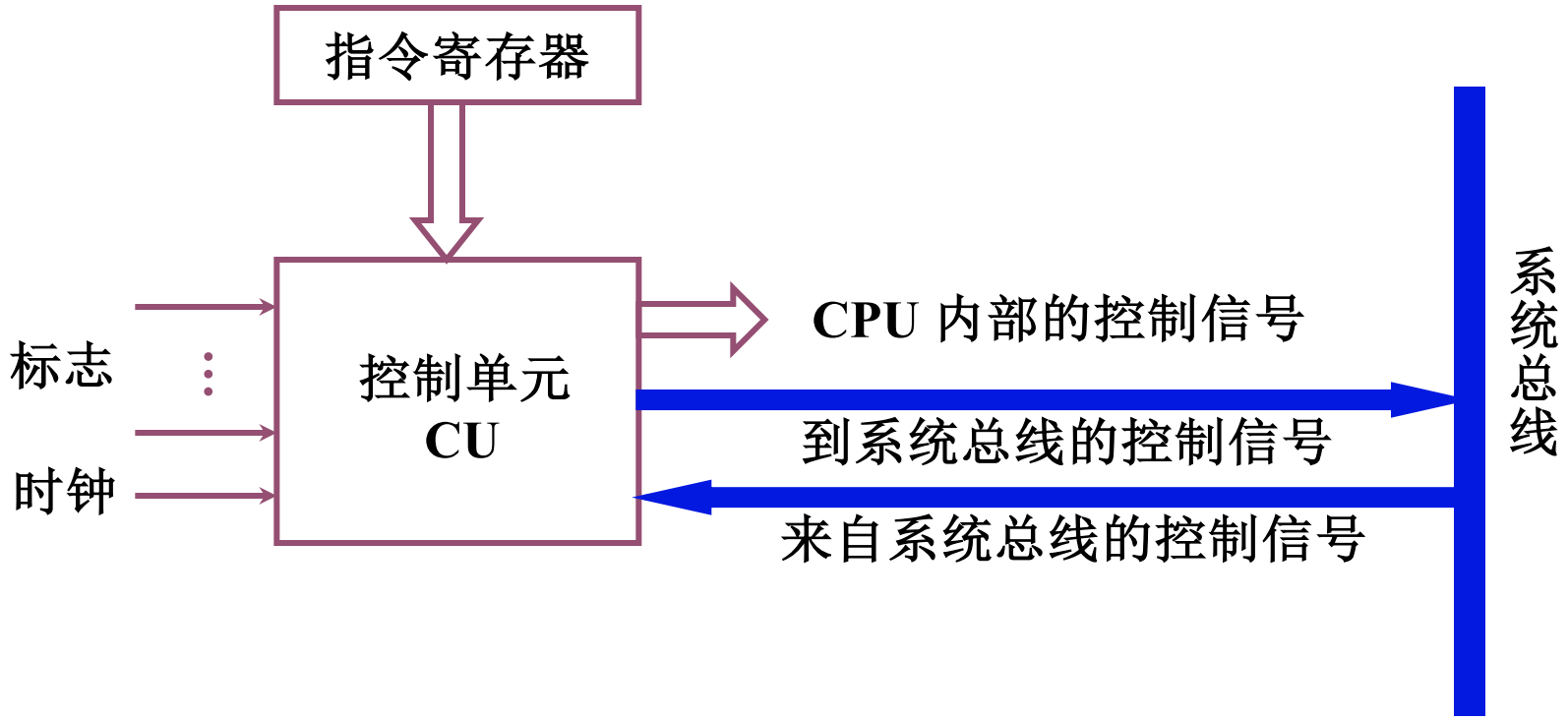
MDR → M(MAR)　　MDR → M(MAR)

中断识别程序入口地址 M → PC

0 → EINT（置“0”）　　0 → EINT（置“0”）

9.2 控制单元的功能

一、控制单元的外特性



1. 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) → CU

控制信号 与操作码有关

(3) 标志

CU 受标志控制

(4) 外来信号

如 INTR 中断请求

HRQ 总线请求

2. 输出信号

(1) CPU 内的各种控制信号

$R_i \longrightarrow R_j$

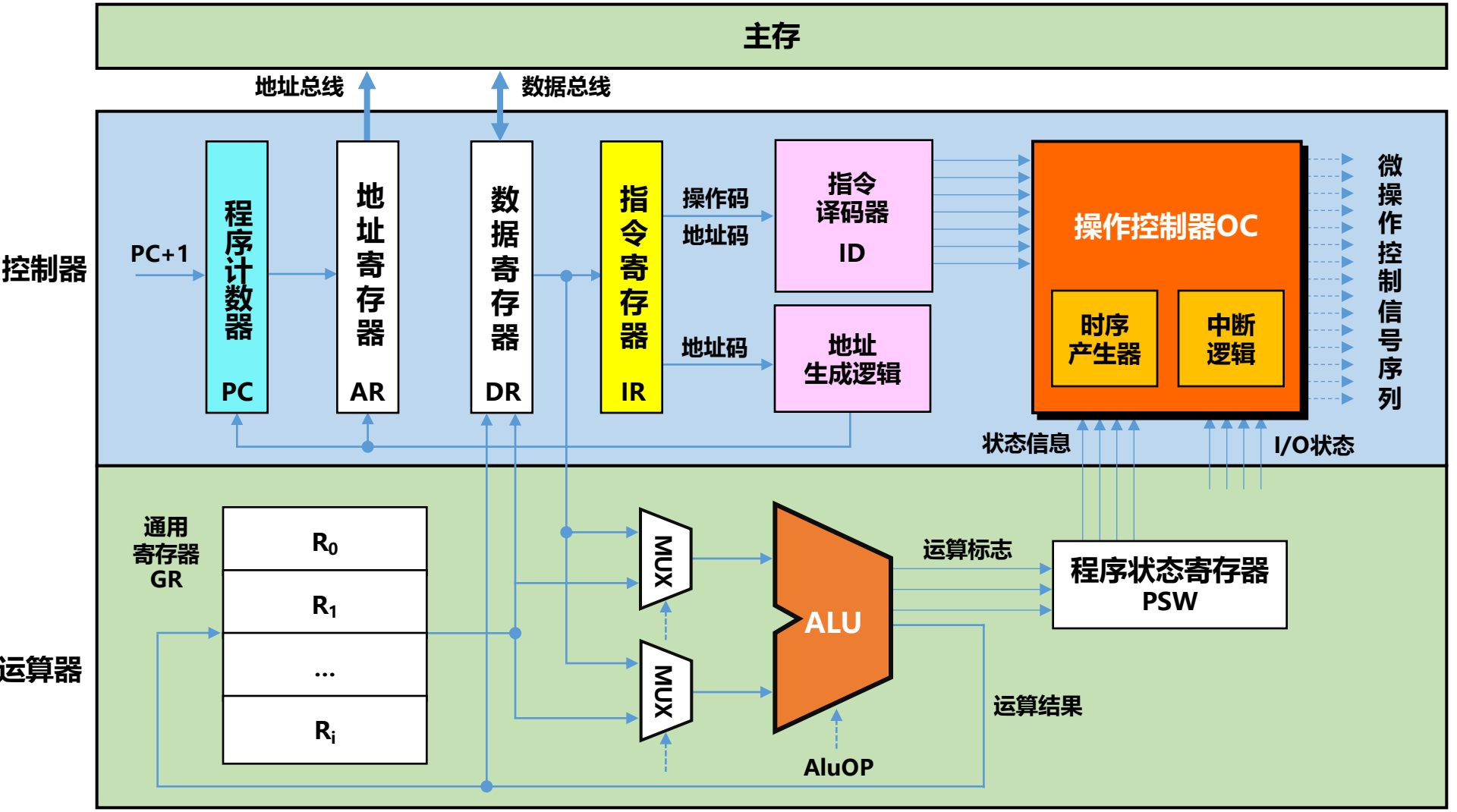
(PC) + 1 ⟶ PC

ALU　＋、－、与、或 ……

(2) 送至控制总线的信号

$\overline{\text{MREQ}}$	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

简单CPU模型

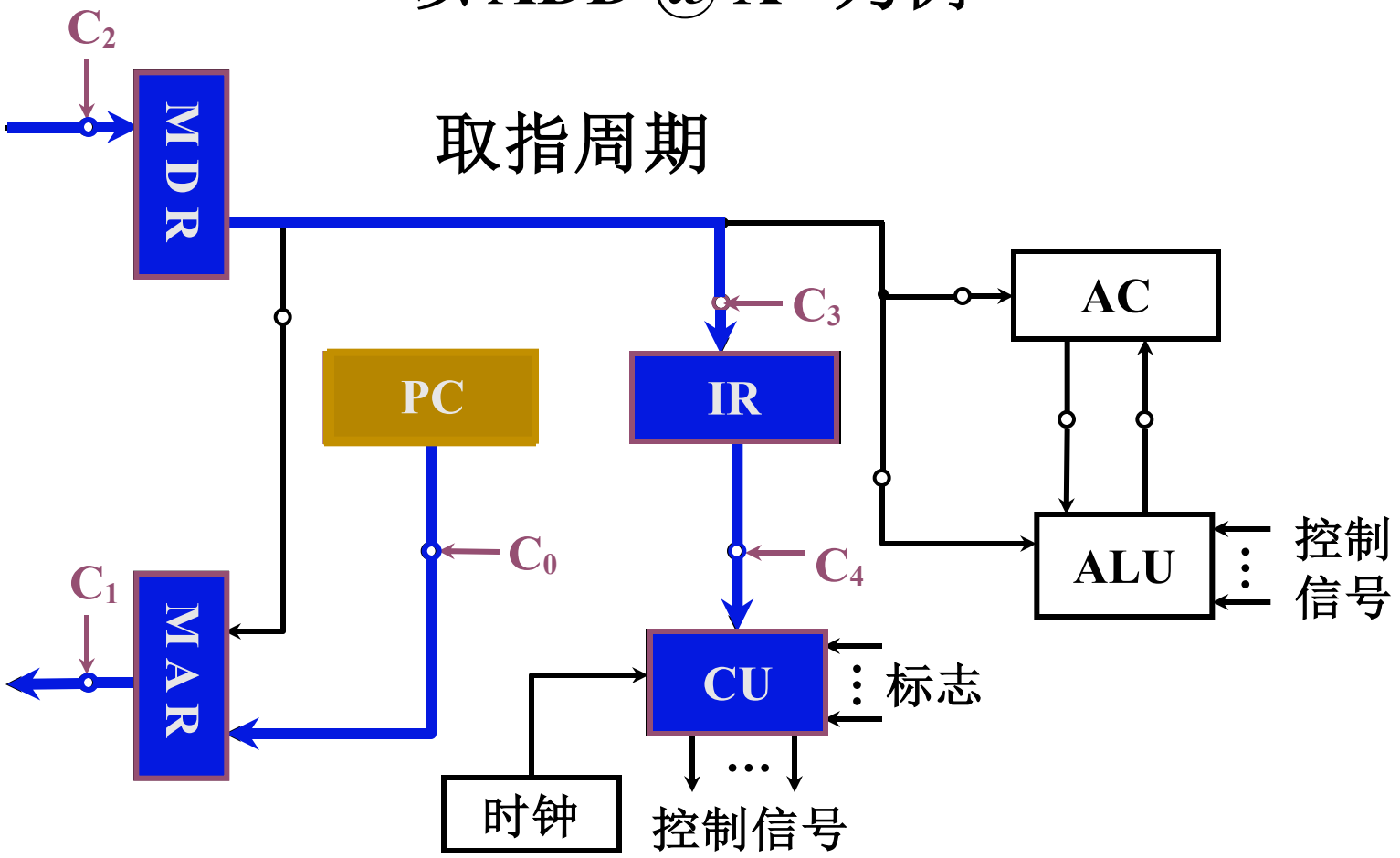

主存
地址总线
数据总线
控制器
PC+1
程序计数器
PC
地址寄存器
AR
数据寄存器
DR
指令寄存器
IR
操作码
地址码
指令译码器
ID
地址码
地址生成逻辑
操作控制器OC
时序产生器
中断逻辑
微操作控制信号序列
状态信息
I/O状态
运算器
通用寄存器
GR
R_0
R_1
...
R_i
MUX
MUX
ALU
运算标志
程序状态寄存器
PSW
运算结果
AluOP

二、控制信号举例

9.2

1. 不采用 CPU 内部总线的方式

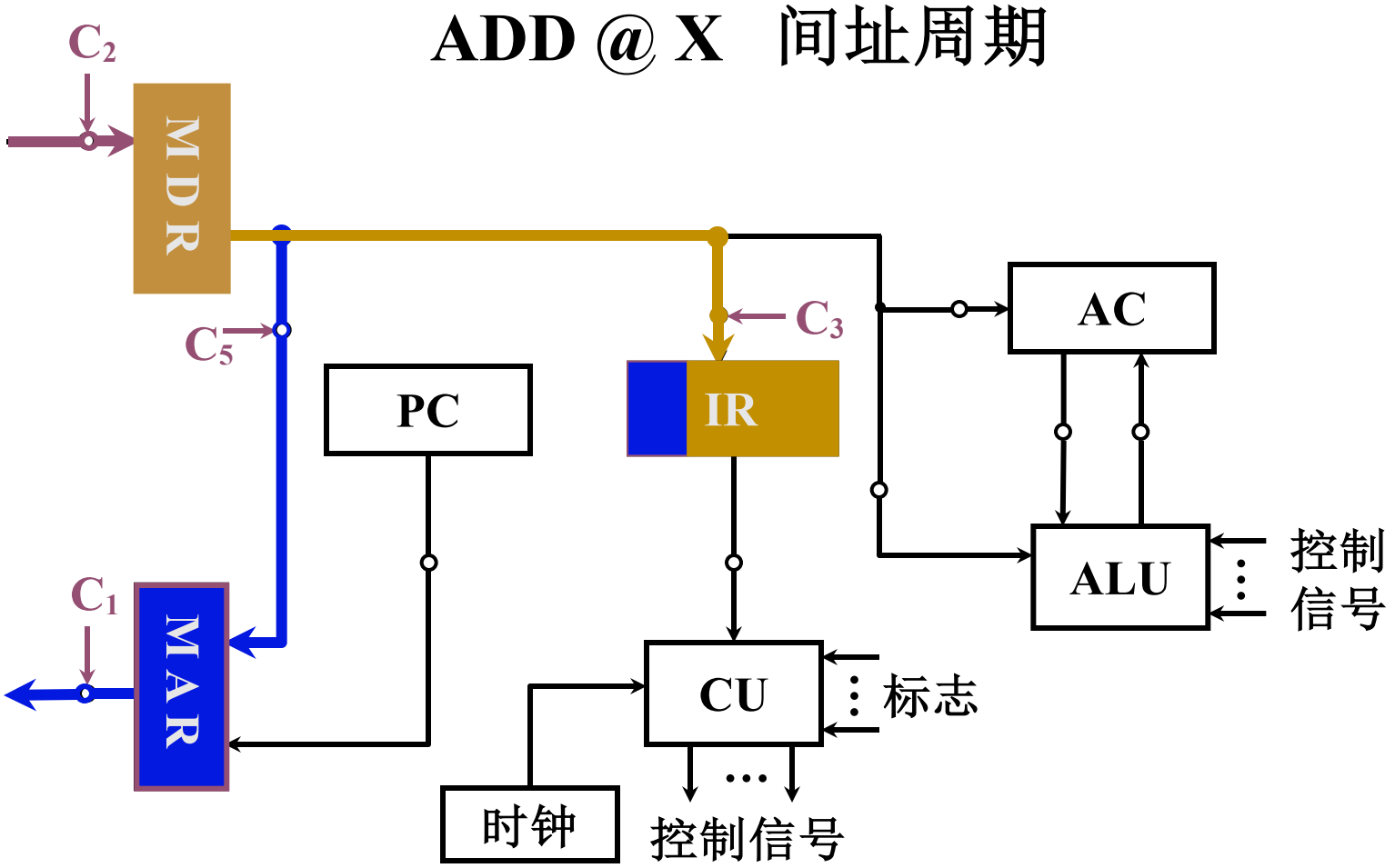
以 ADD @ X 为例



二、控制信号举例

9.2

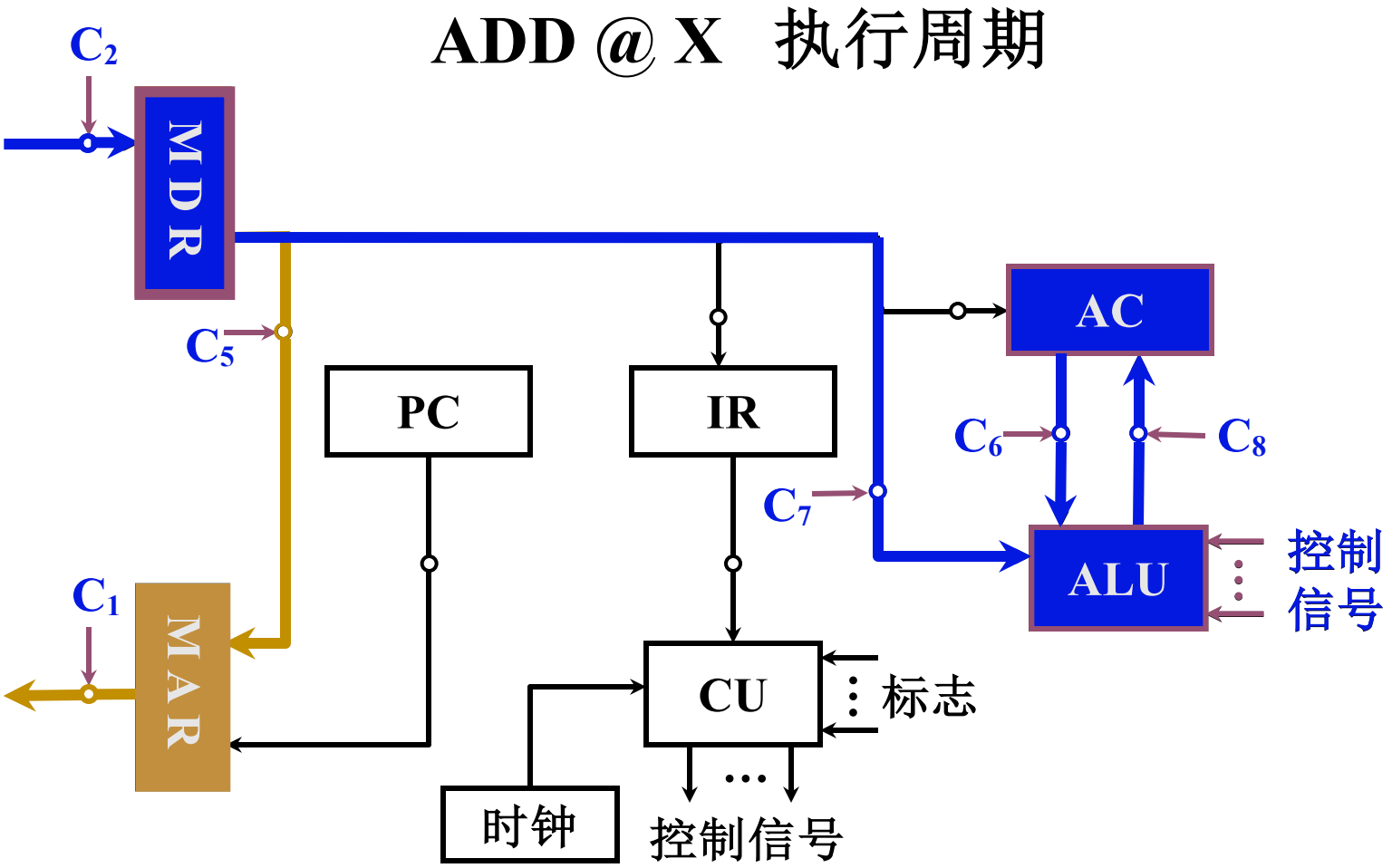
1. 不采用 CPU 内部总线的方式



二、控制信号举例

9.2

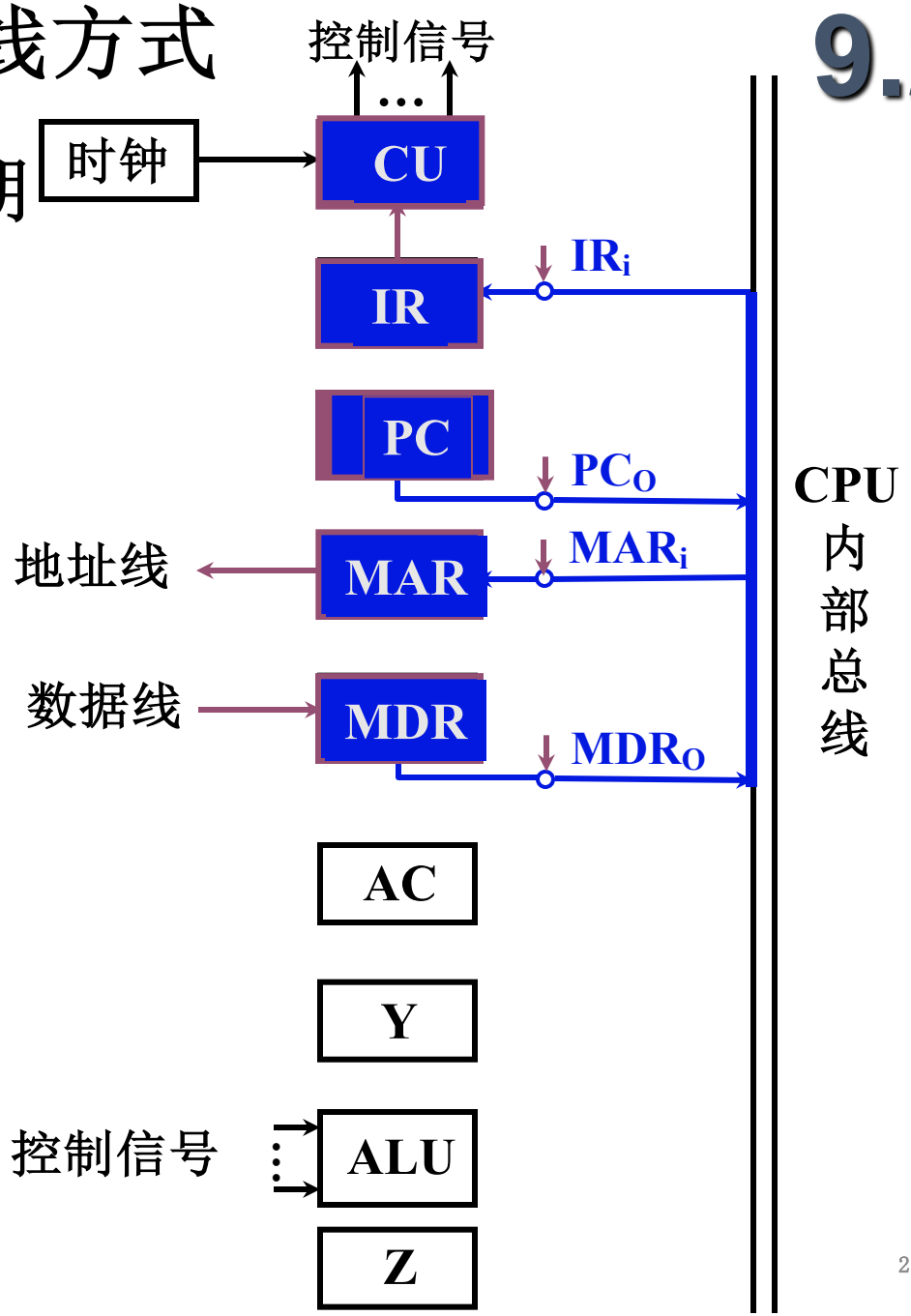
1. 不采用 CPU 内部总线的方式



2. 采用 CPU 内部总线方式

(1) ADD @ X 取指周期

- PC⟶ MAR⟶ 地址线
  PC_O MAR_i
- CU 发读命令 1⟶R
- 数据线 ⟶ MDR
- MDR ⟶ IR
  MDR_O IR_i
- OP（IR）⟶ CU
- （PC）+ 1⟶PC

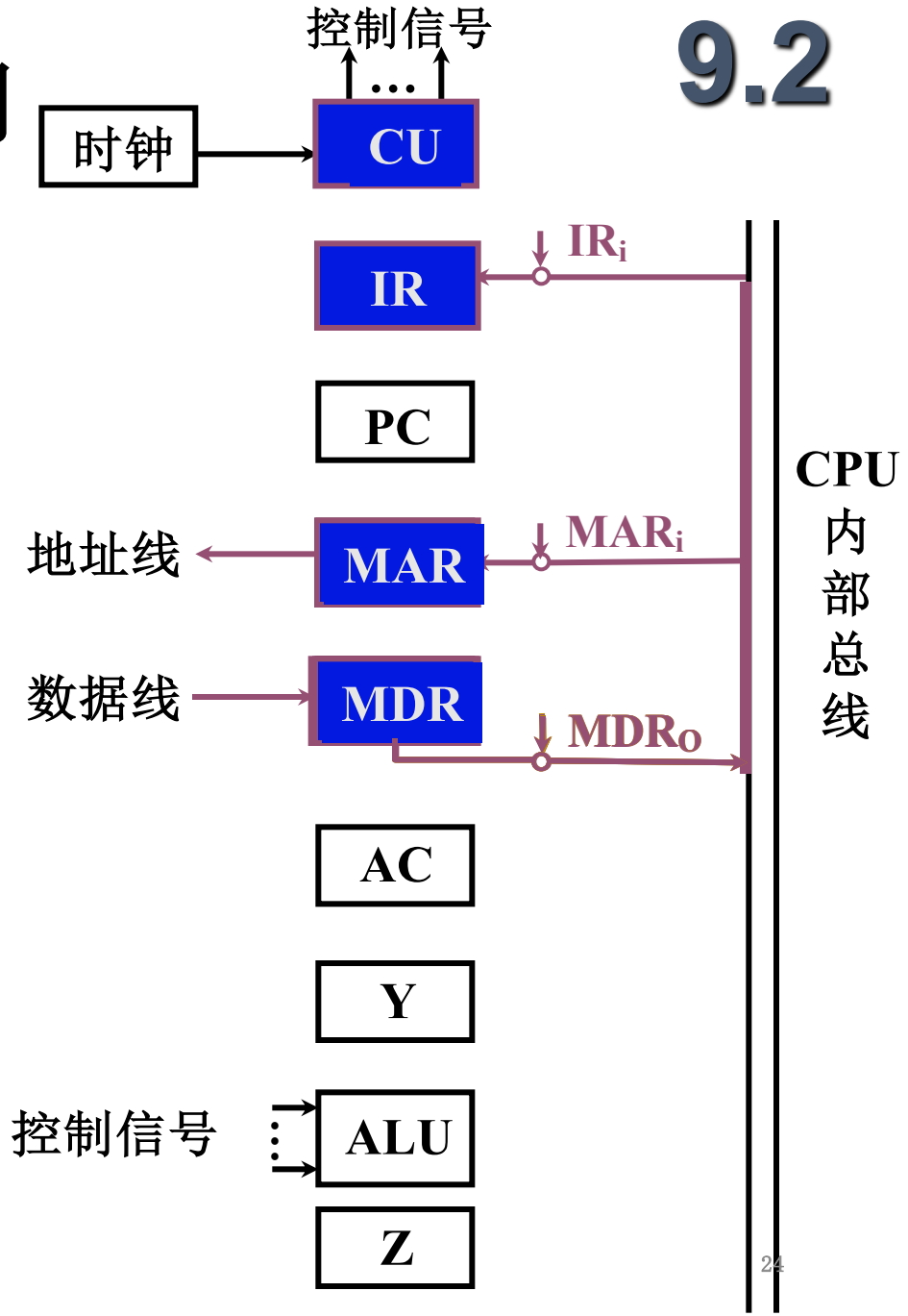


(2) ADD @ X 间址周期

形式地址 ⟶ MAR

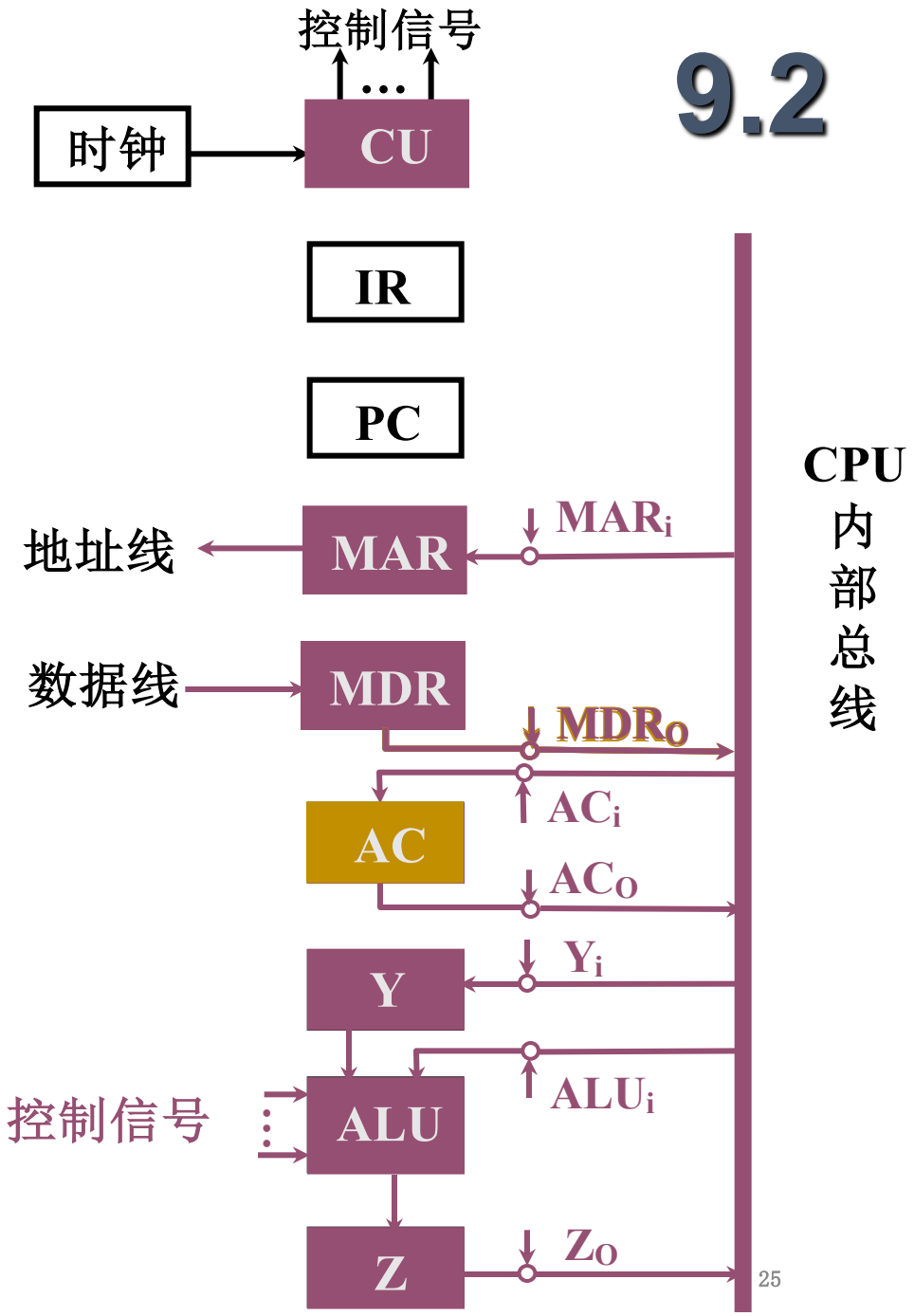
- MDR ⟶ MAR ⟶ 地址线
  MDR_O MAR_i
- 1 ⟶ R
- 数据线 ⟶ MDR
- MDR ⟶ IR
  MDR_O IR_i

有效地址 ⟶ Ad（IR）



(3) ADD @ X 执行周期

- MDR ⟶ MAR ⟶ 地址线
  MDR_O MAR_i
- 1 ⟶ R
- 数据线 ⟶ MDR
- MDR ⟶ Y ⟶ ALU
  MDR_O Y_i
- AC ⟶ ALU
  AC_O ALU_i
- （AC）+（Y）⟶ Z
- Z ⟶ AC
  Z_O AC_i



三、多级时序系统

9.2

1. 机器周期

(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）

一个机器周期内可完成若干个微操作

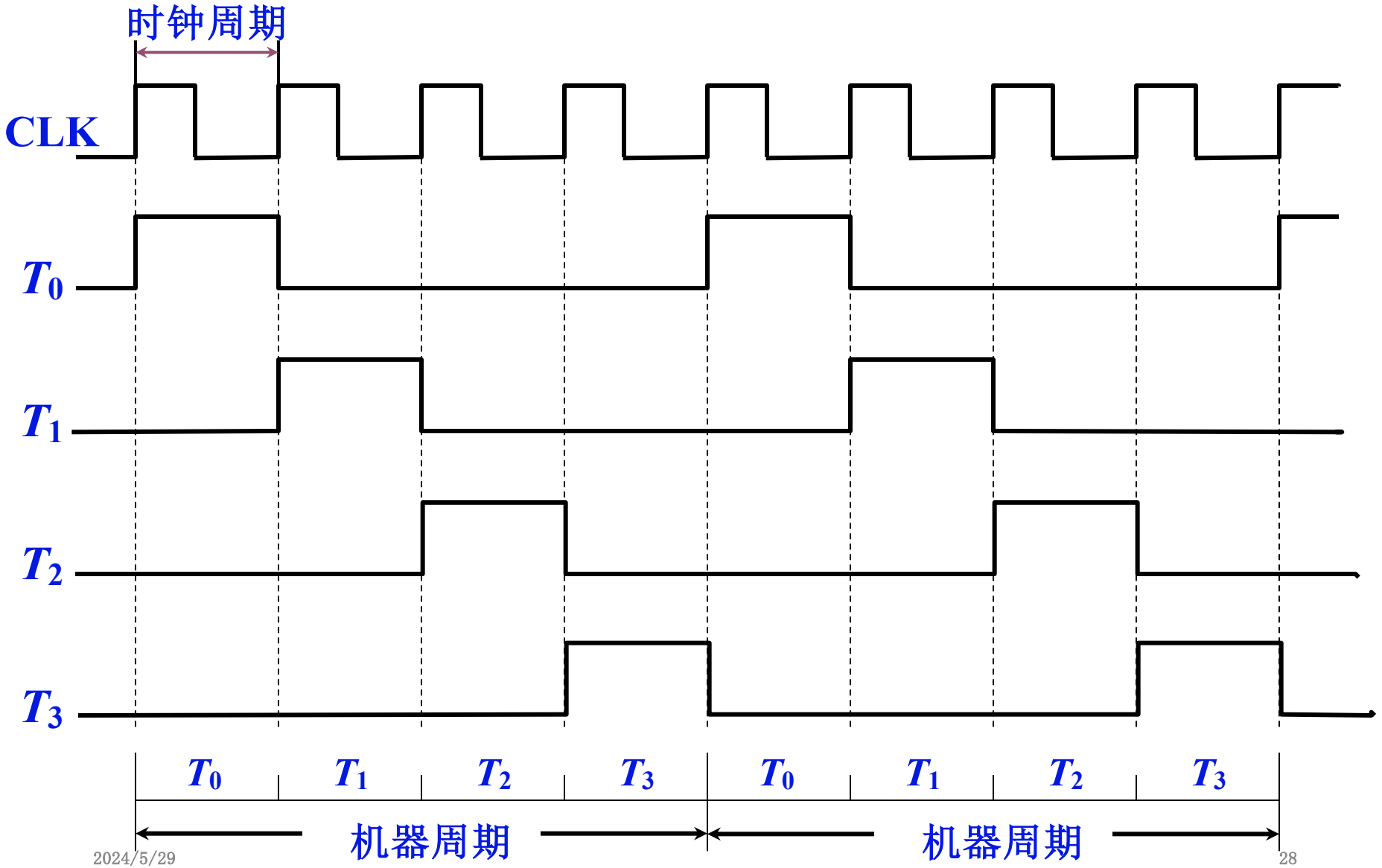
每个微操作需一定的时间

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）



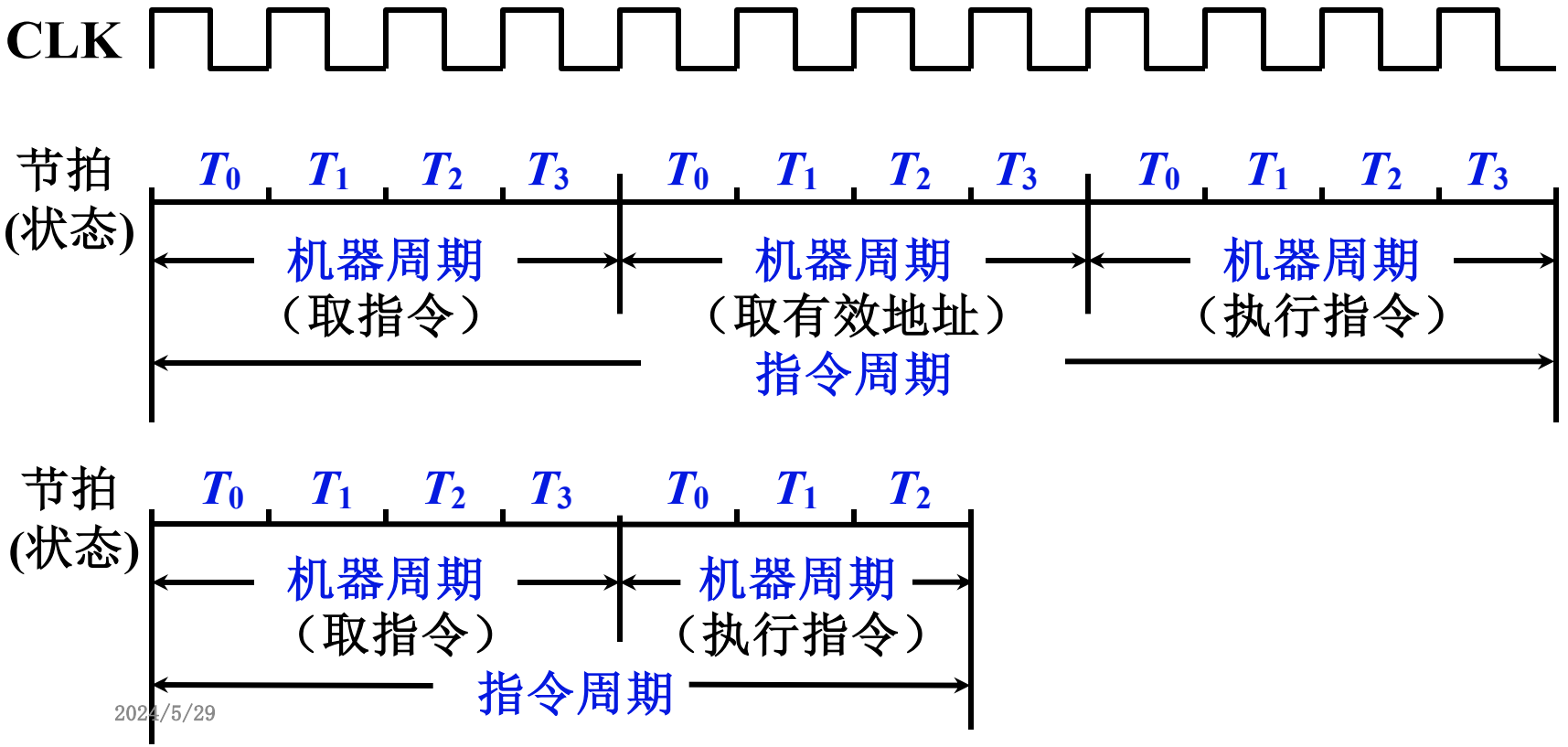
3. 多级时序系统

9.2

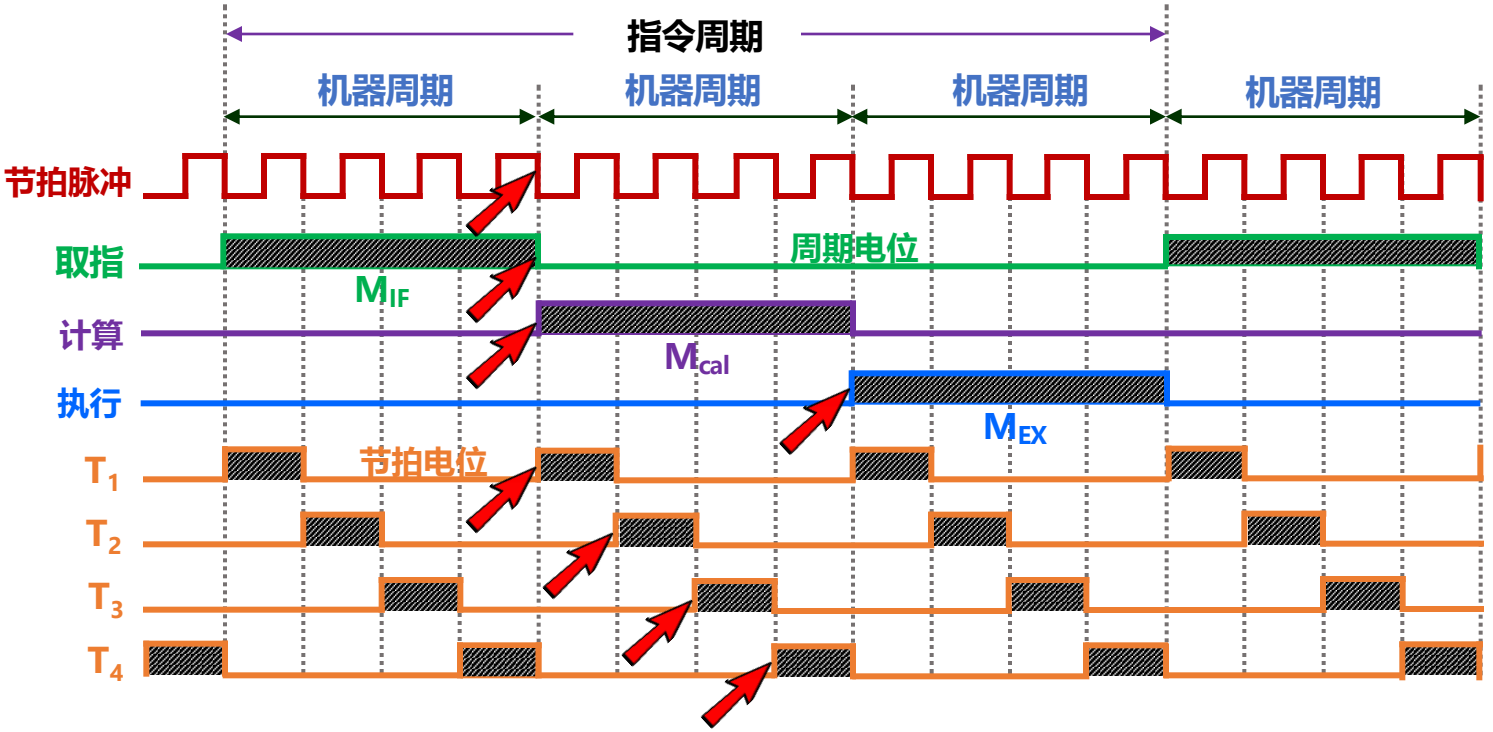
机器周期、节拍（状态）组成多级时序系统

一个指令周期包含若干个机器周期

一个机器周期包含若干个时钟周期



定长指令周期的三级时序发生器（举例）



构建时序发生器？ 输入：节拍脉冲 输出：M_{IF}，Mcal，M_{EX}，T1~T4

4. 机器速度与机器主频的关系 9.2

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

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

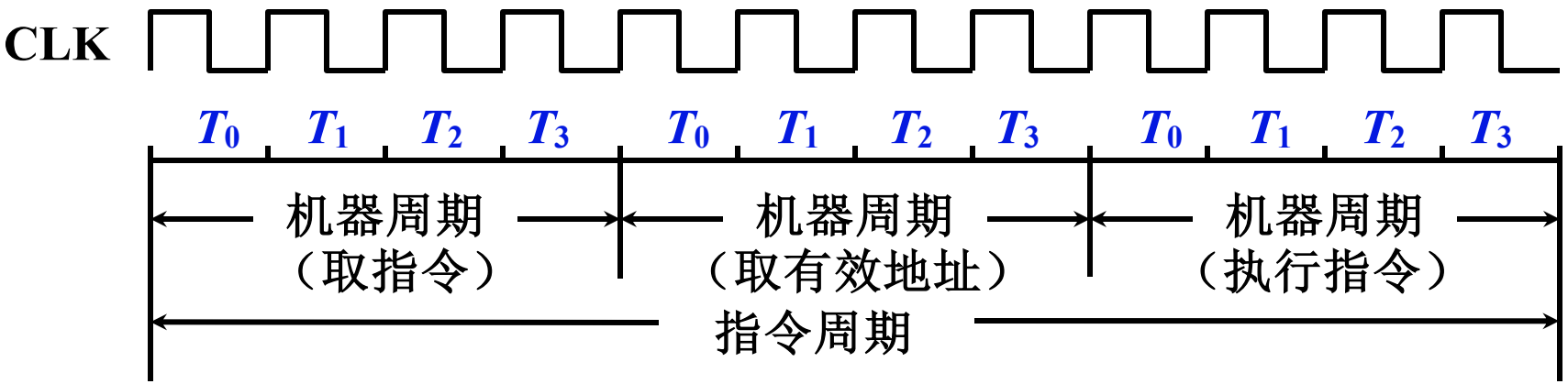
机器速度 不仅与 主频有关，还与机器周期中所含时钟周期（主频的倒数）数 以及指令周期中所含的 机器周期数有关

四、控制方式 9.2

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



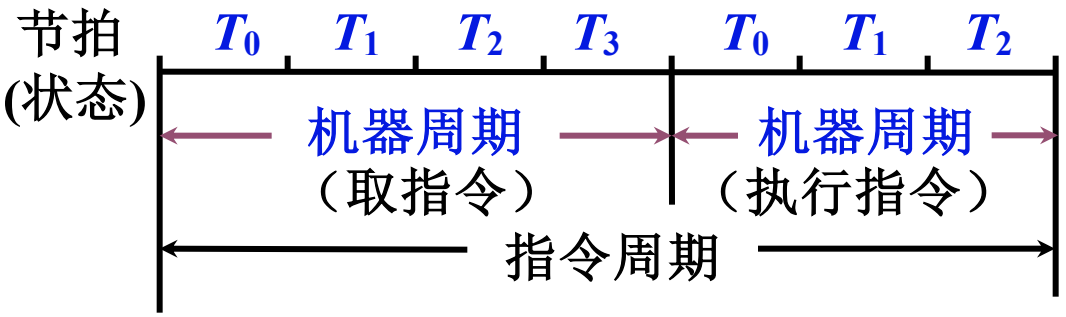
(1) 采用 定长 的机器周期

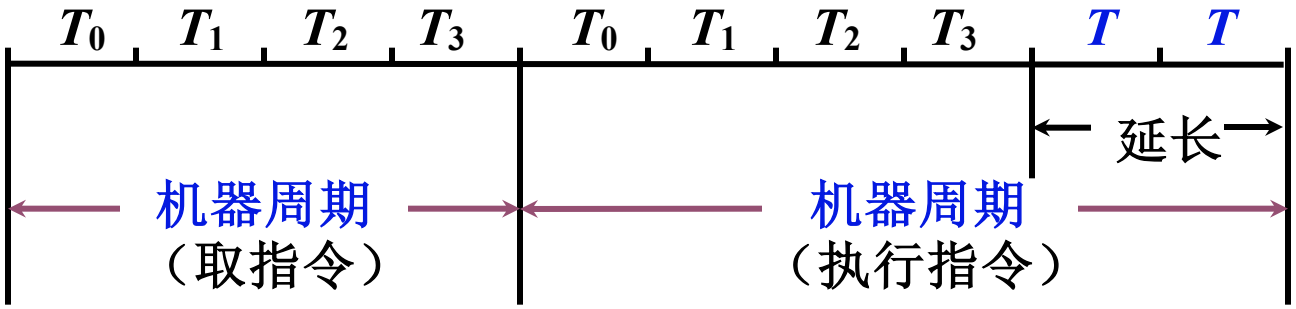
以 最长 的微操作序列和 最复杂 的微操作作为 标准

机器周期内 节拍数相同

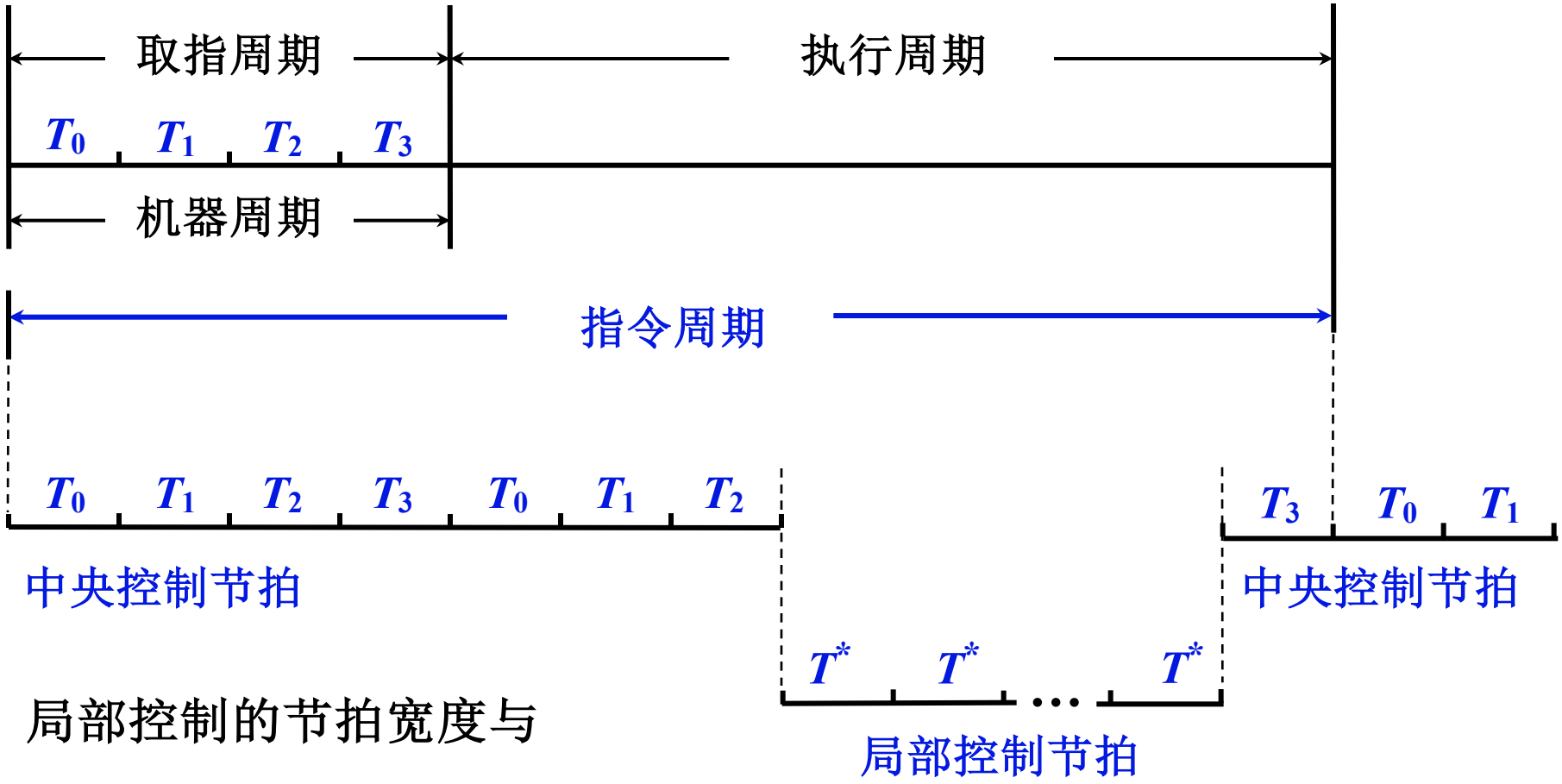
(2) 采用不定长的机器周期

机器周期内 节拍数不等





(3) 采用中央控制和局部控制相结合的方法 9.2



局部控制的节拍宽度与

中央控制的节拍宽度一致

2. 异步控制方式

无基准时标信号

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

第10章　控制单元的设计

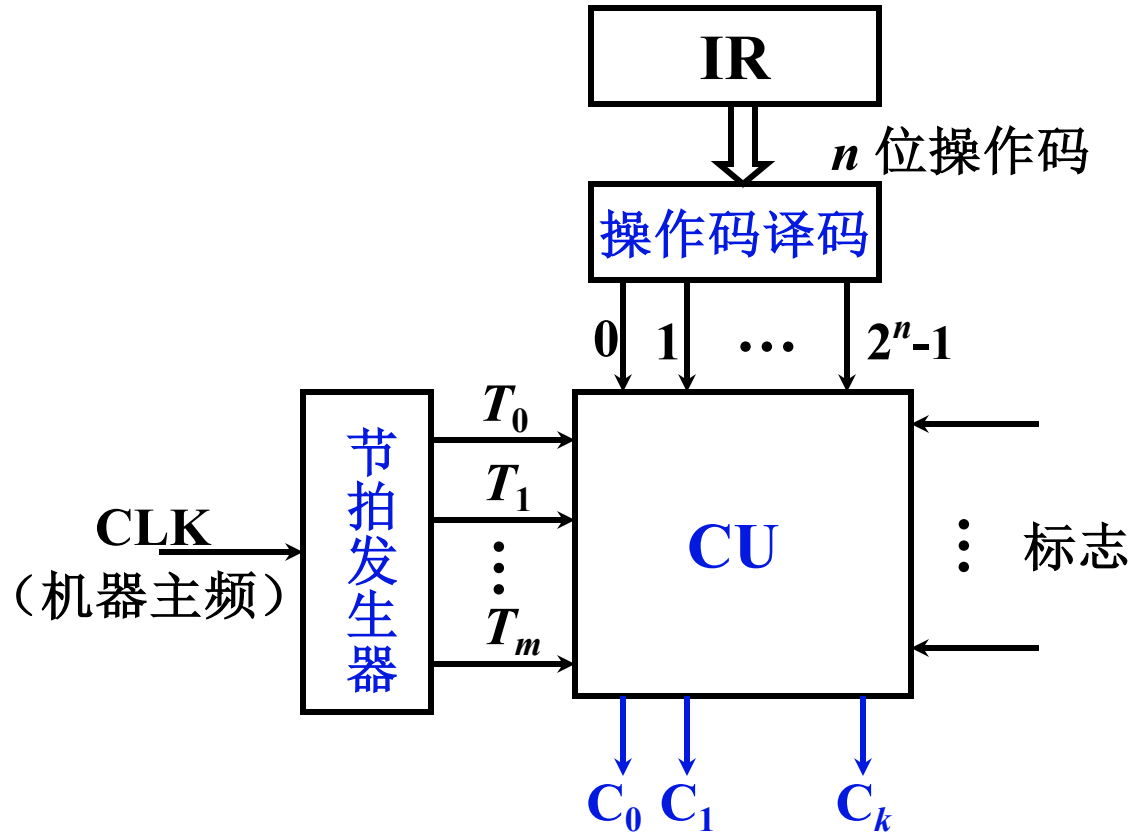
10.1　组合逻辑设计

10.2　微程序设计

10.1 组合逻辑设计

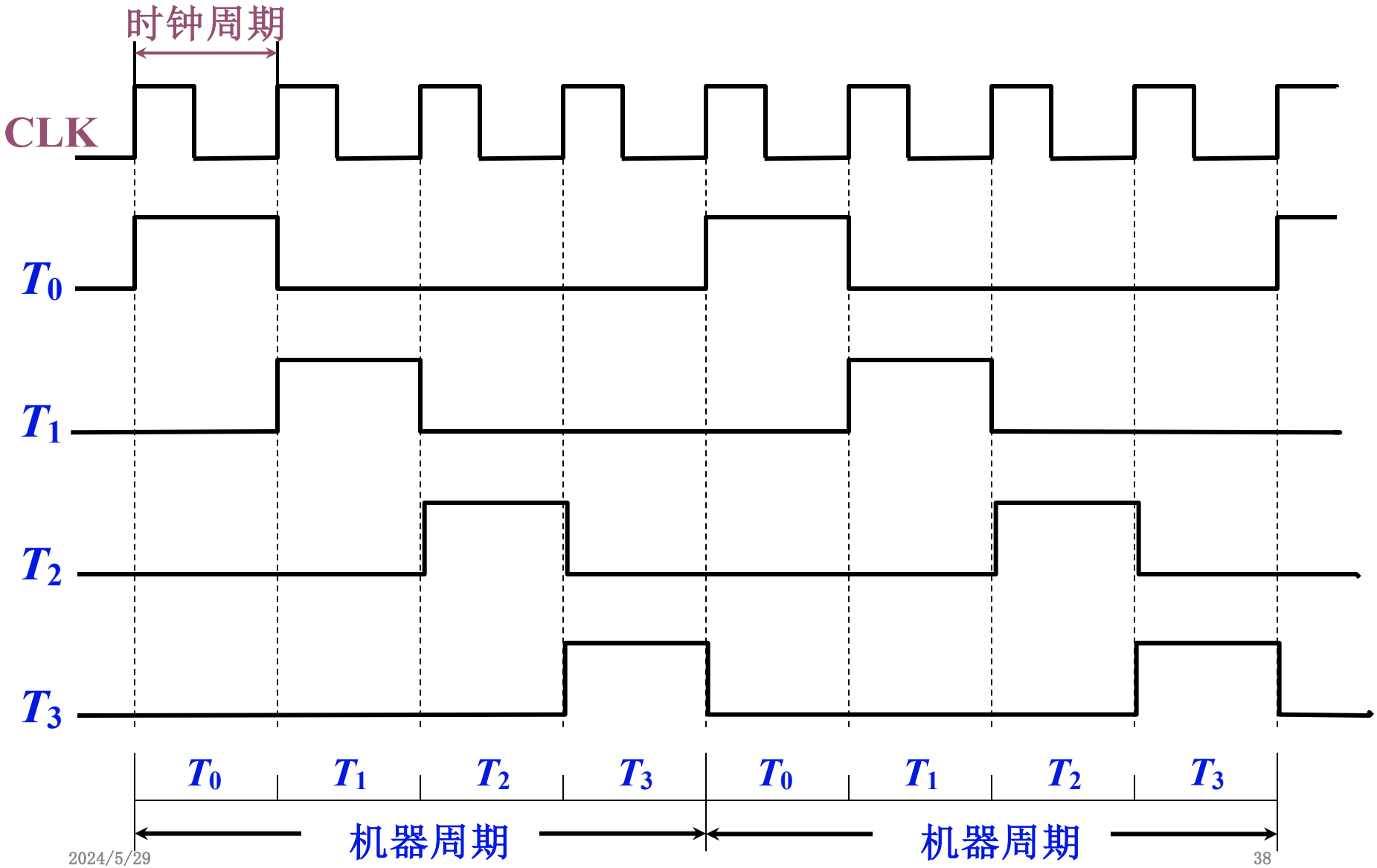
一、组合逻辑控制单元框图

1. CU 外特性



2. 节拍信号

10.1



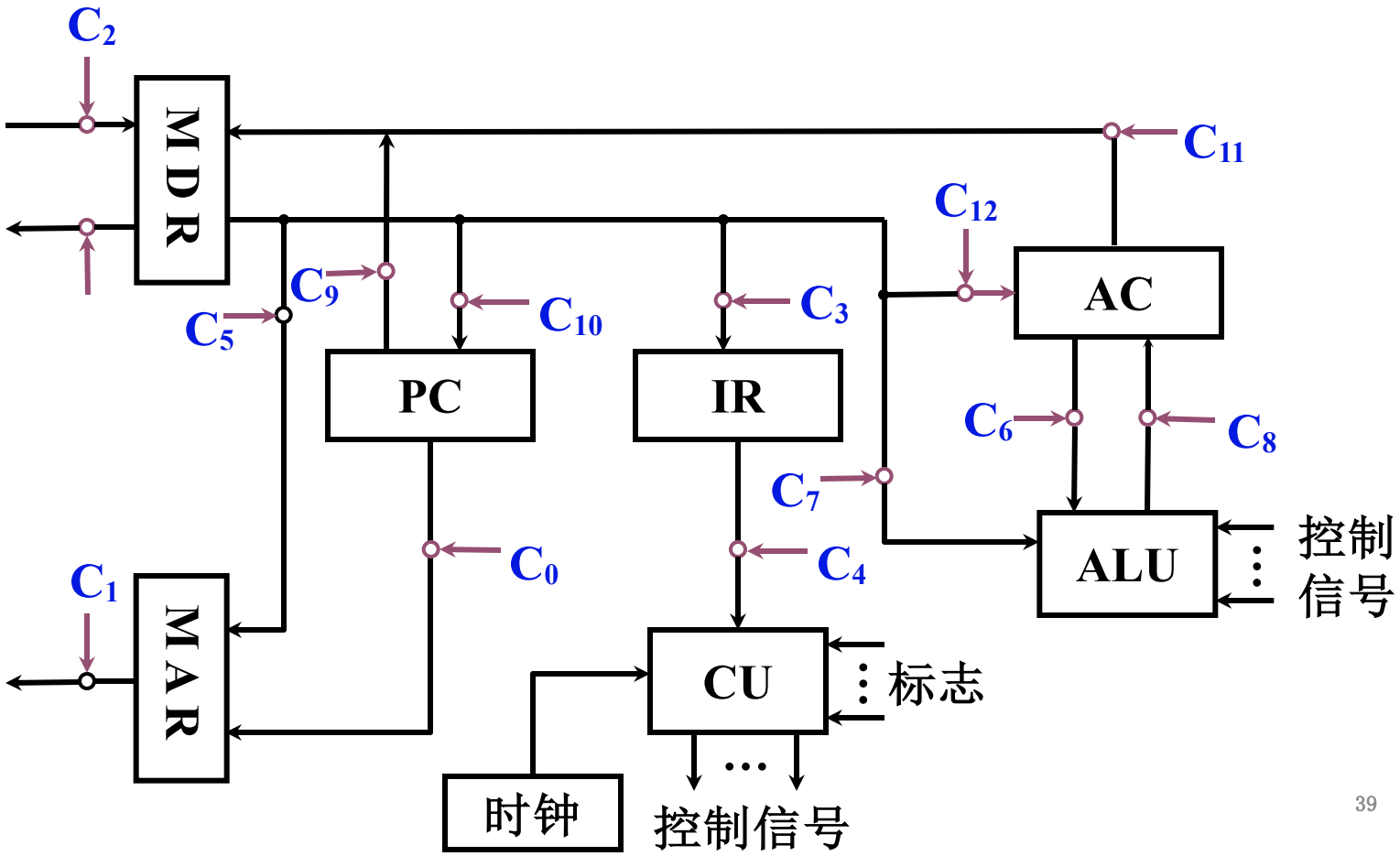
二、微操作的节拍安排

10.1

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)	

10.1

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

10.1

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

三、组合逻辑设计步骤

10.1

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

10.1

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

10.1

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

10.1

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

10.1

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
		$\overline{\text{IND}}$	1⟶ EX			1	1	1	1

三、组合逻辑设计步骤

10.1

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	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 10.1

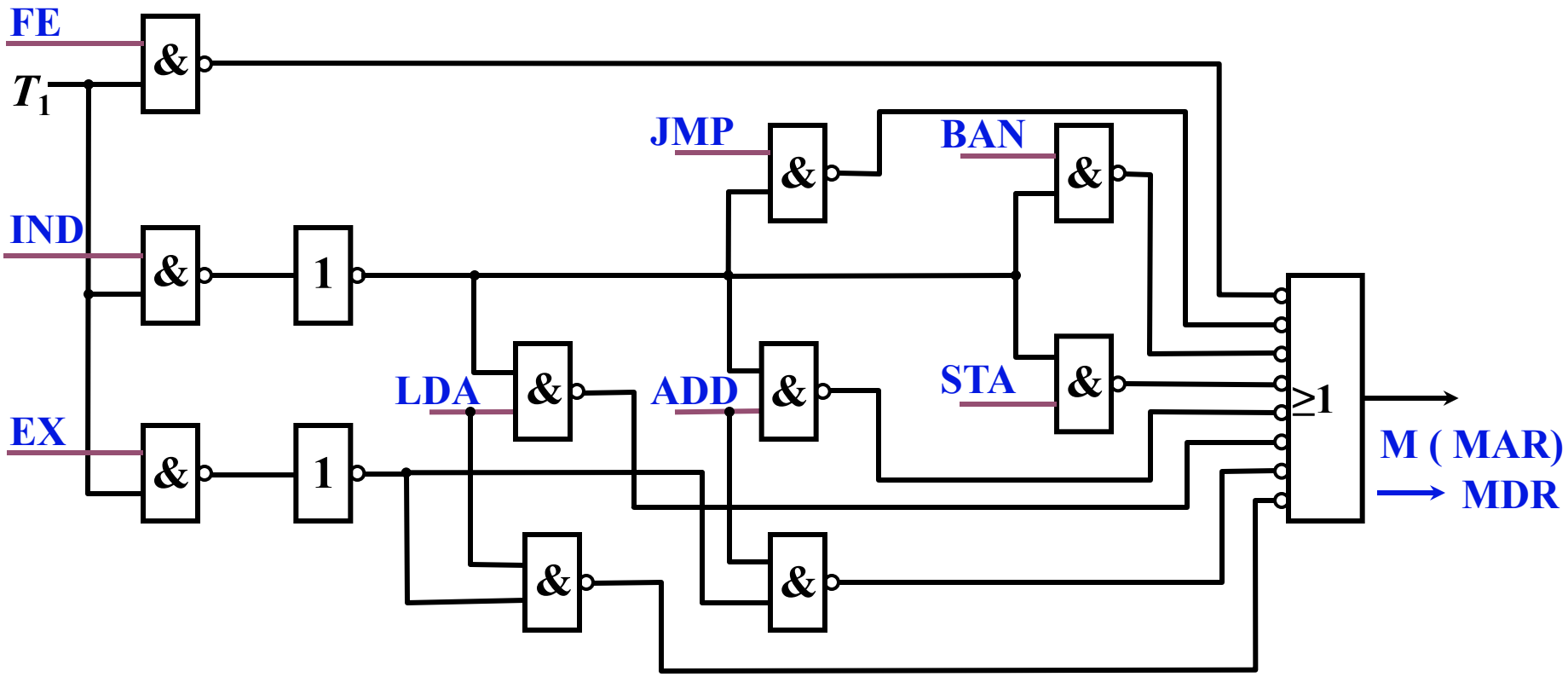
$$M(MAR) \longrightarrow MDR$$

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

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

3. 画出逻辑图

10.1



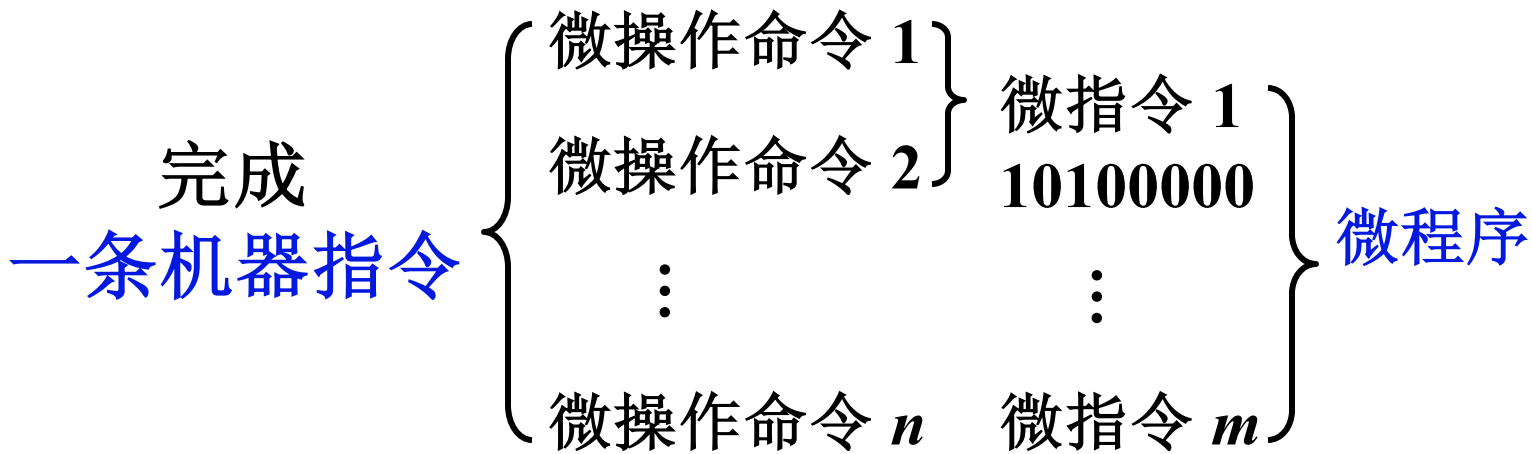
特点

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

10.2 微程序设计

一、微程序设计思想的产生

1951 英国剑桥大学教授 Wilkes

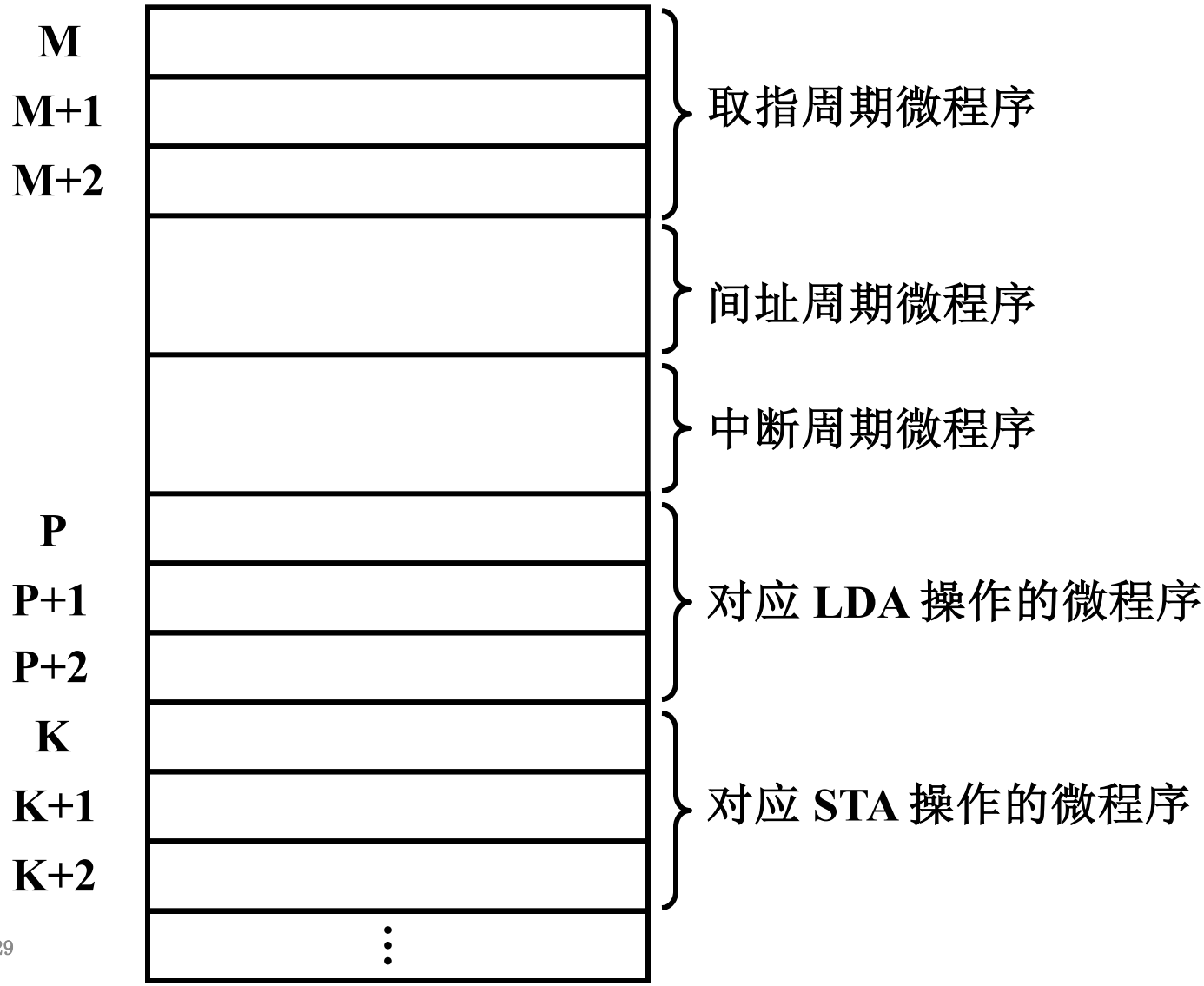


存储逻辑

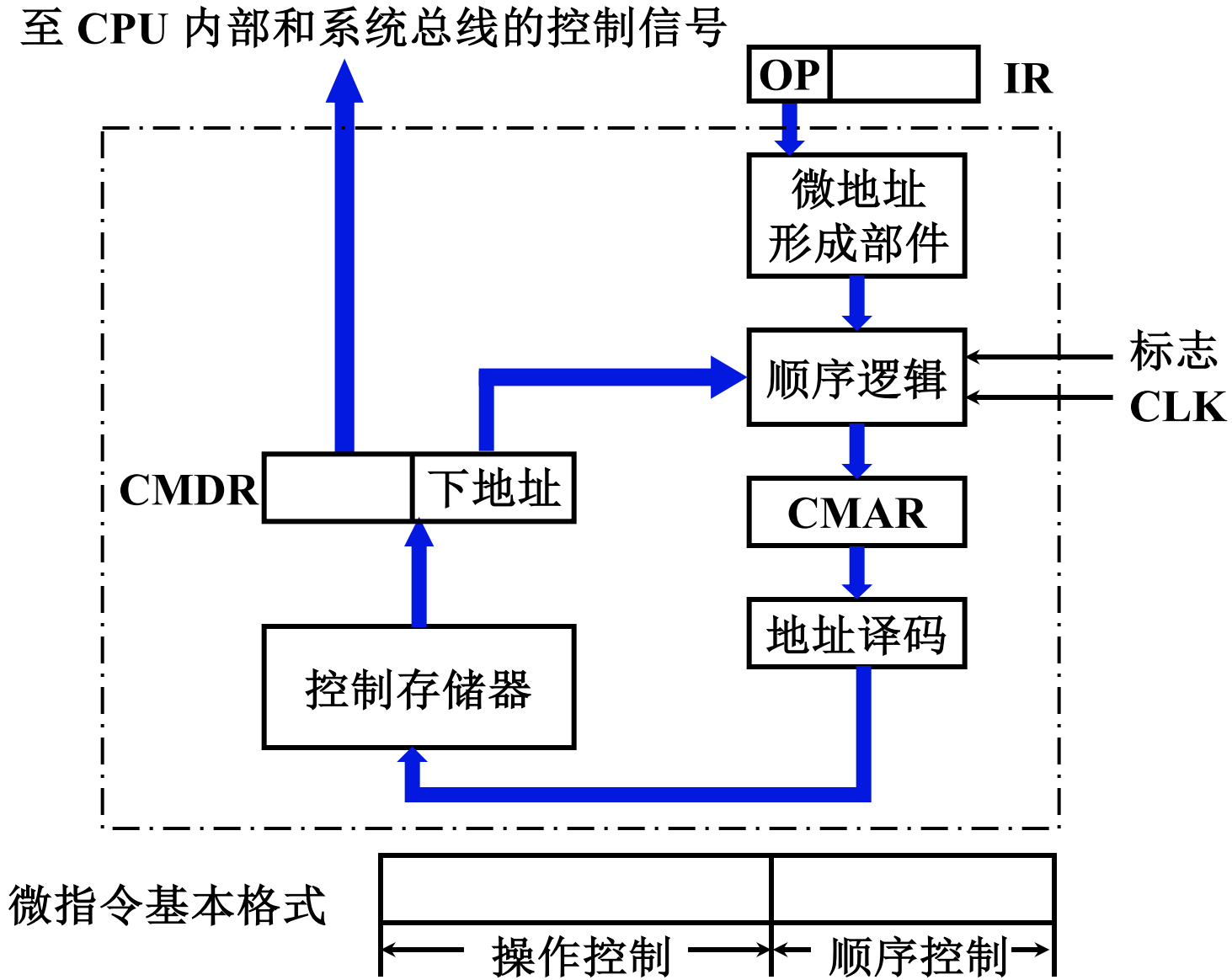
二、微程序控制单元框图及工作原理

10.2

1. 机器指令对应的微程序

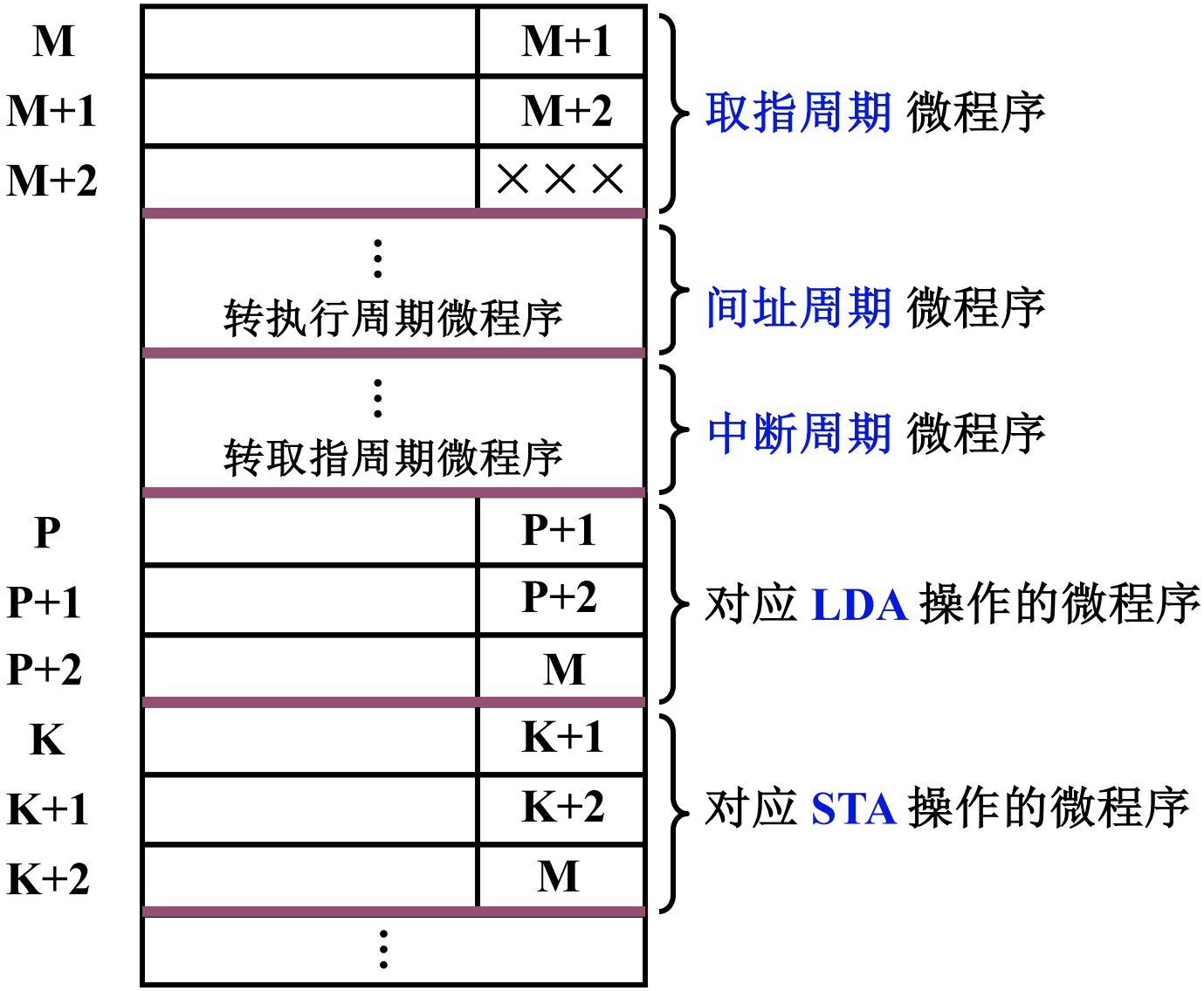


2. 微程序控制单元的基本框图

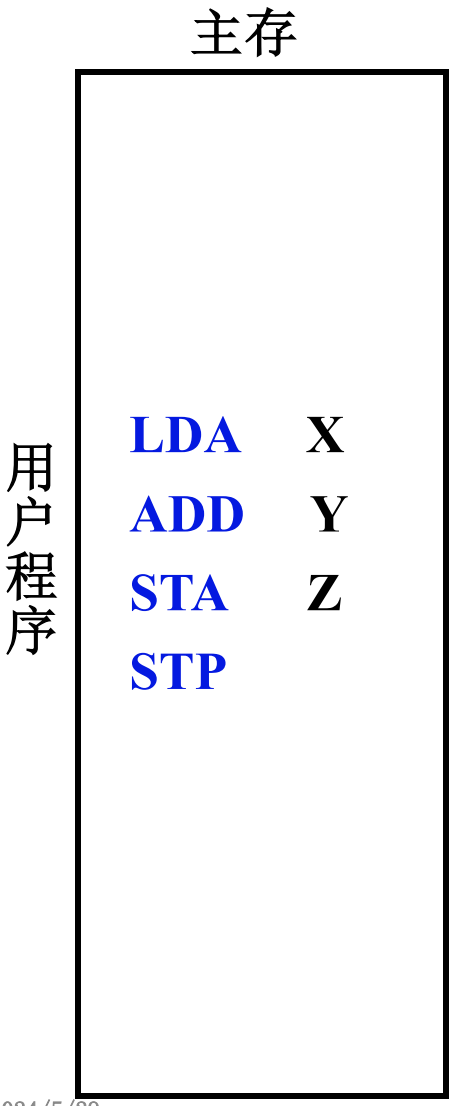


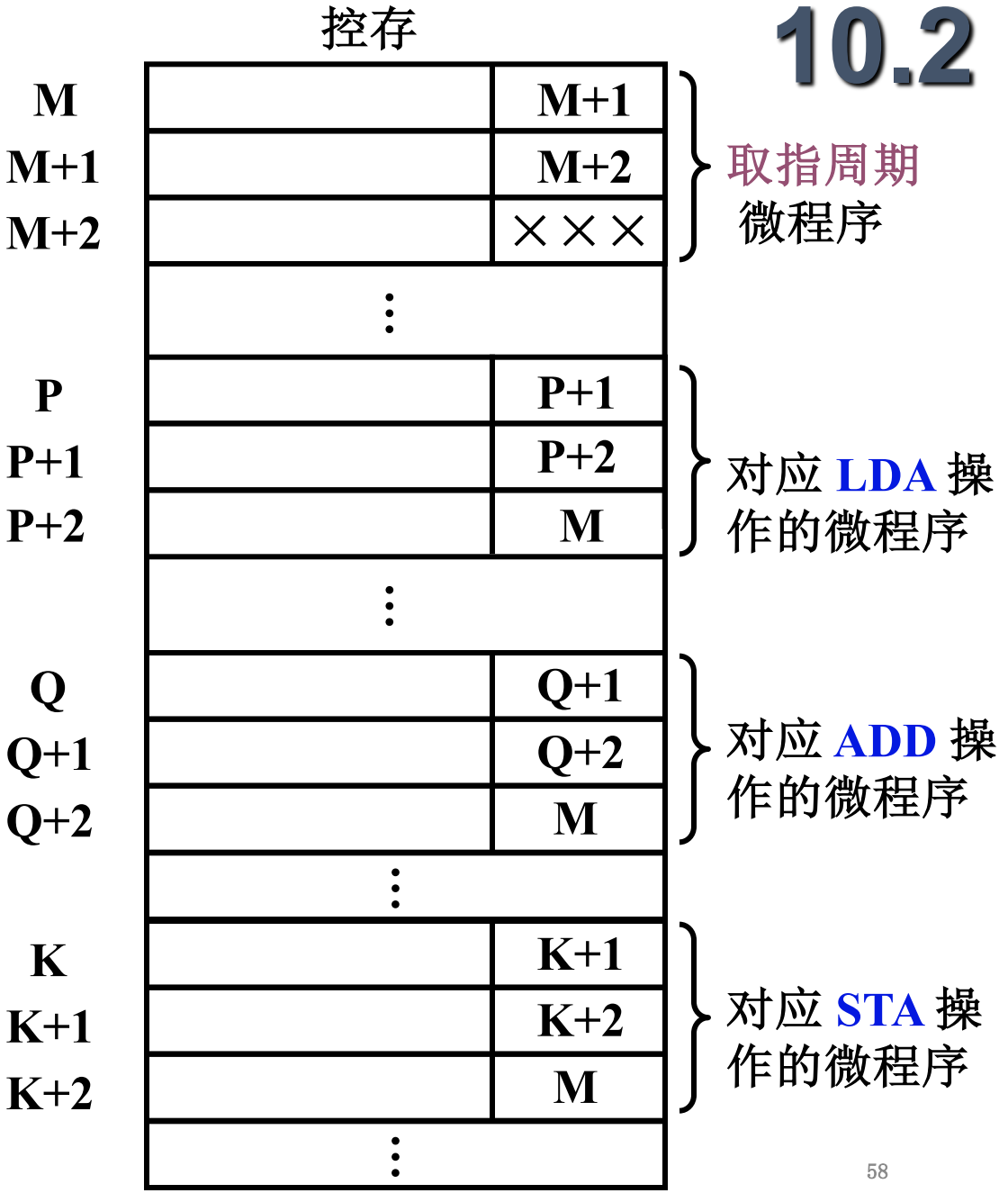
二、微程序控制单元框图及工作原理

10.2

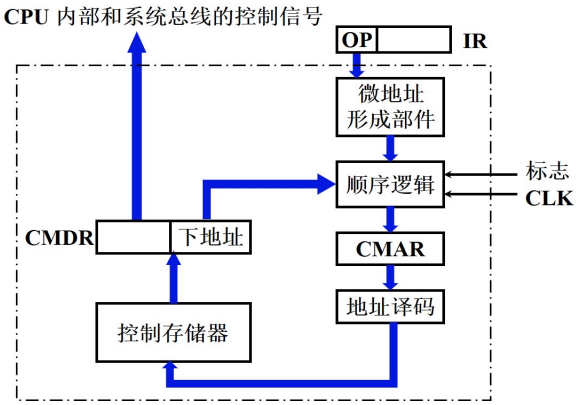


3. 工作原理


主存
用户程序
LDA X
ADD Y
STA Z
STP


控存
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



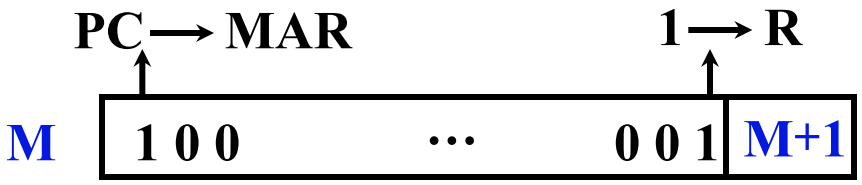
(1) 取指阶段　　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址　M + 1

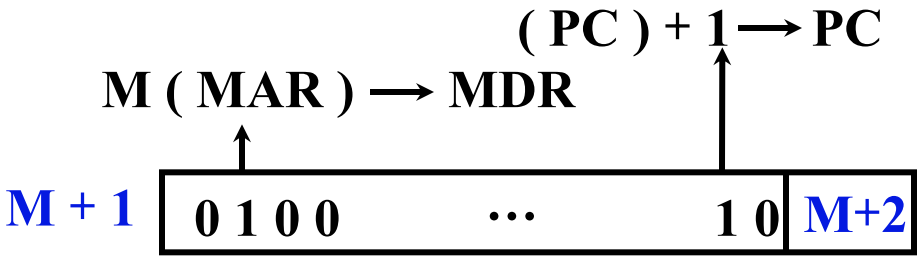


Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

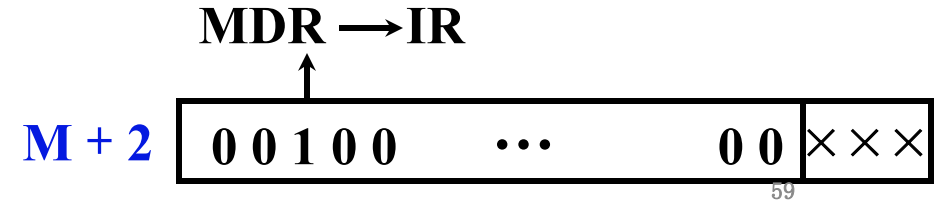
形成下条微指令地址　M + 2



Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令



(2) 执行阶段　执行 LDA 微程序

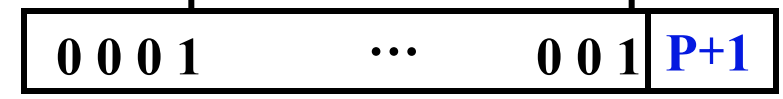
OP (IR) ⟶ 微地址形成部件 ⟶ CMAR　(P ⟶ CMAR)

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址

Ad (IR) ⟶ MAR　　1 ⟶ R



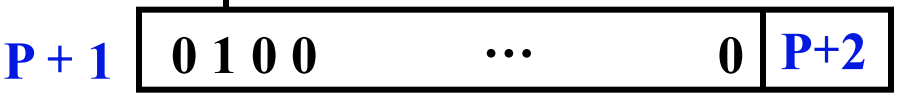
Ad (CMDR) ⟶ MAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址

M (MAR) ⟶ MDR



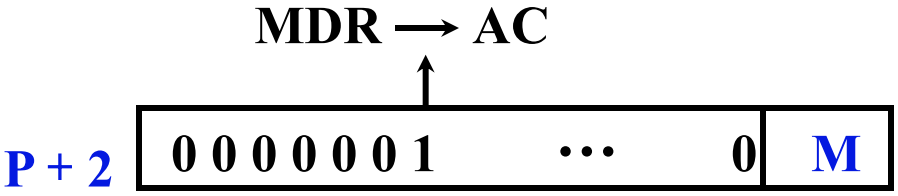
Ad (CMDR) ⟶ MAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址

MDR ⟶ AC



Ad (CMDR) ⟶ MAR　　(M ⟶ CMAR)

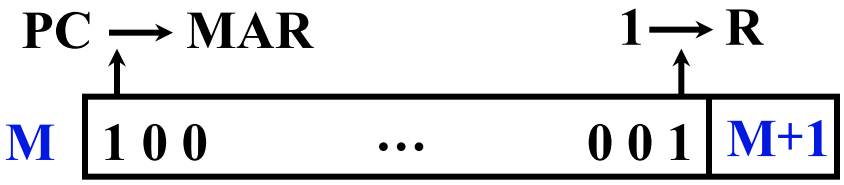
(3) 取指阶段　执行取指微程序

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

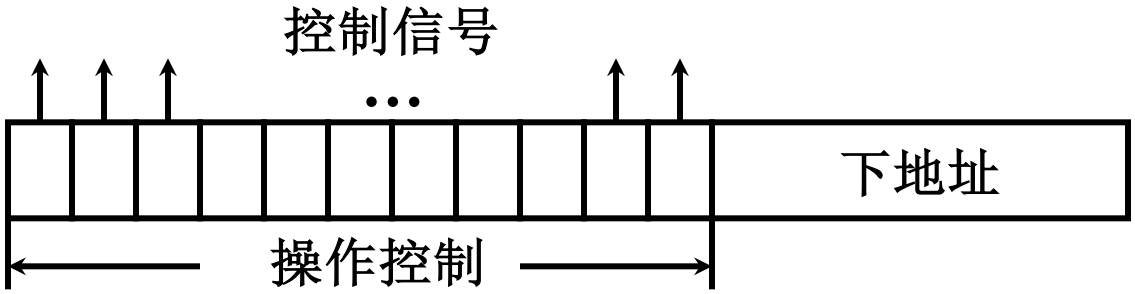
三、微指令的编码方式（控制方式）

10.2

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



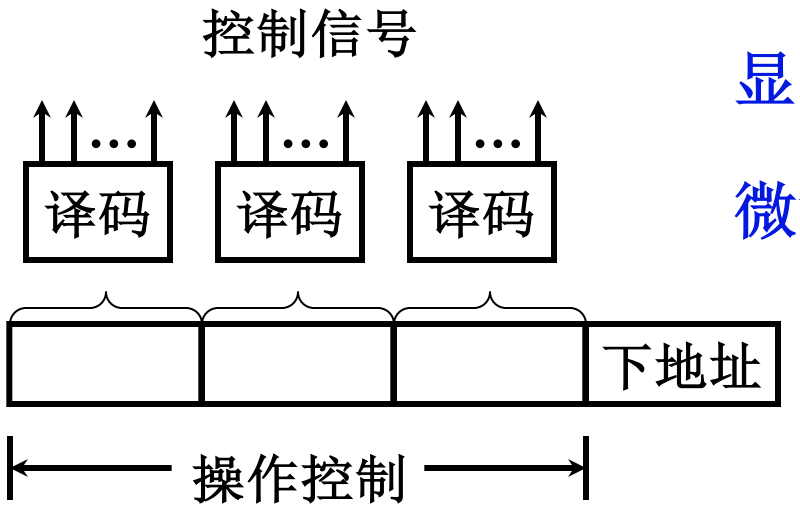
速度最快

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



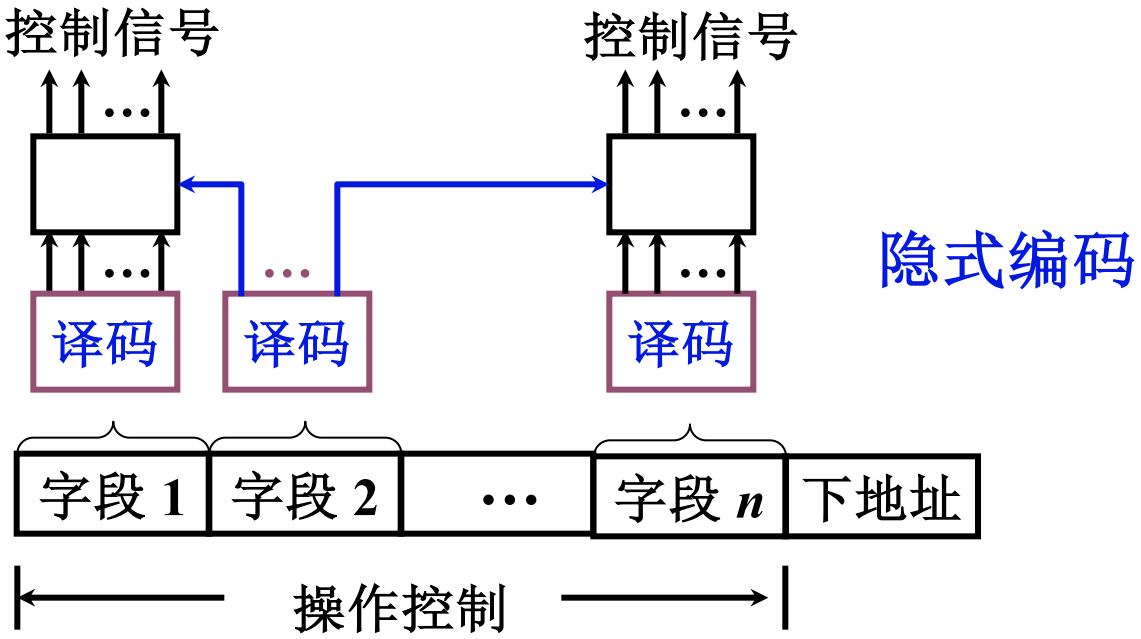
显式编码

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

10.2

1. 微指令的 下地址字段 指出

2. 根据机器指令的 操作码 形成

3. 增量计数器

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

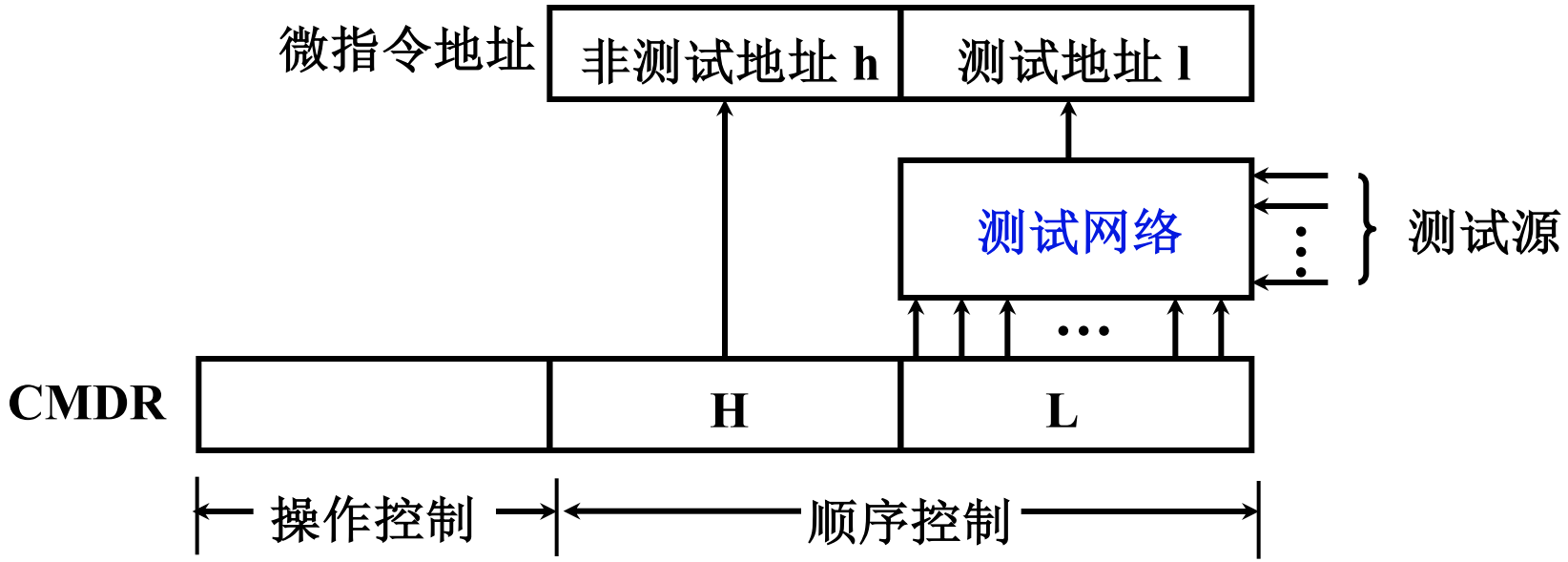
4. 分支转移

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

5. 通过测试网络



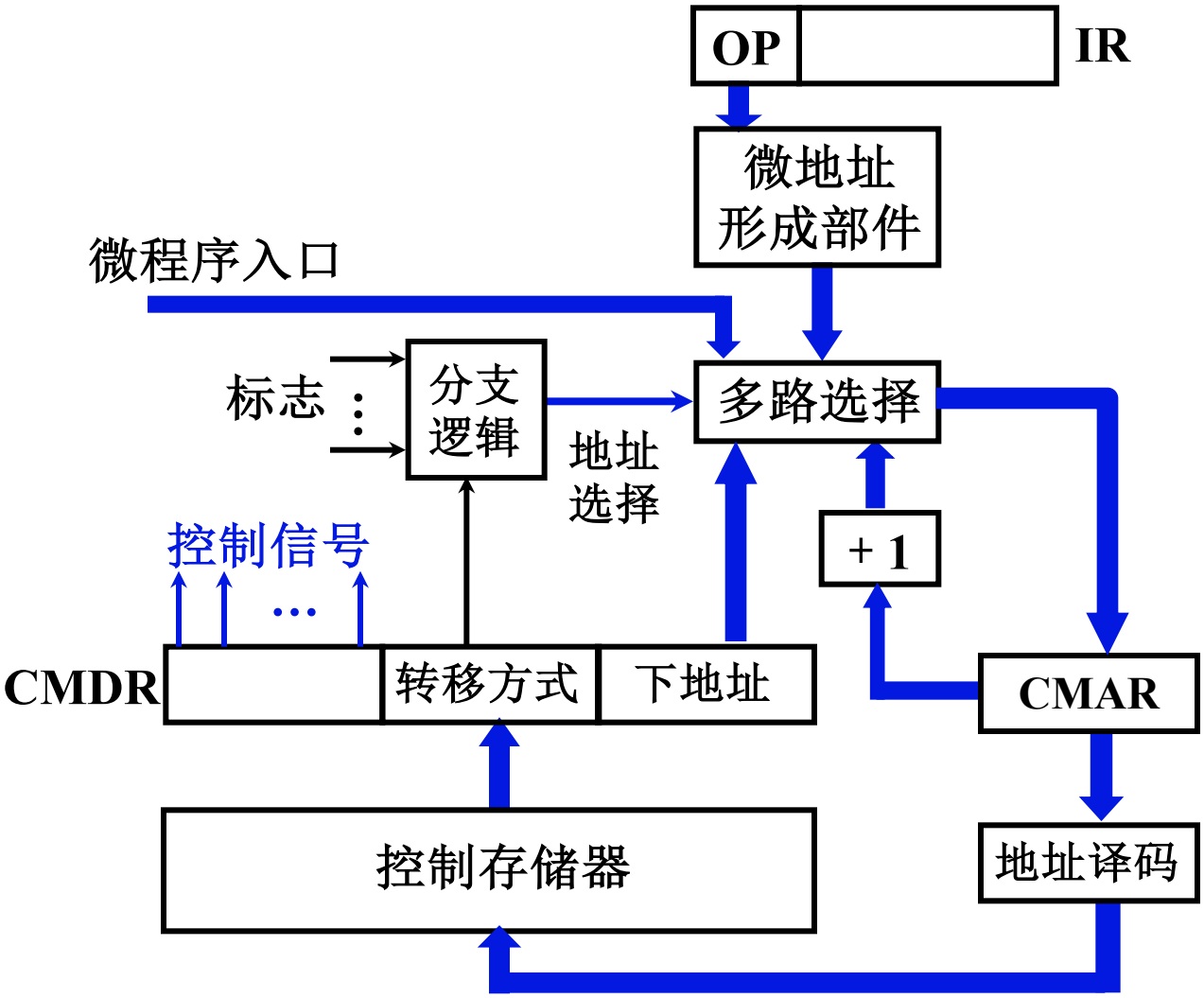
6. 由硬件产生微程序入口地址

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

10.2



五、微指令格式

10.2

1. 水平型微指令

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

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

六、静态微程序设计和动态微程序设计

10.2

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

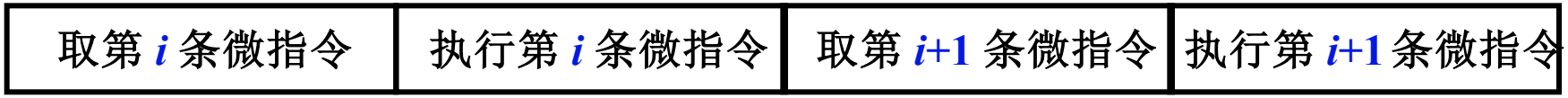
毫微程序设计　用 毫微程序解释微程序

毫微指令与微指令 的关系好比 微指令与机器指令 的关系

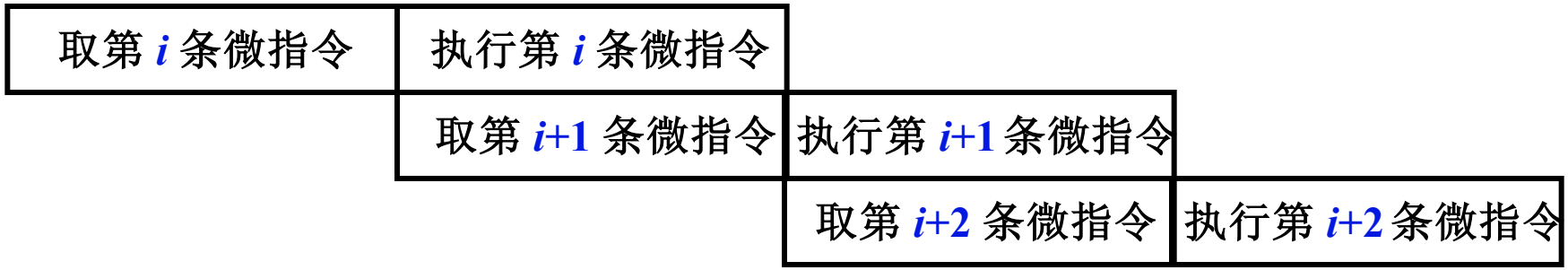
八、串行微程序控制和并行微程序控制

10.2

串行 微程序控制



并行 微程序控制



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

(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

若需考虑 T_1 如何安排这条微指令 ？

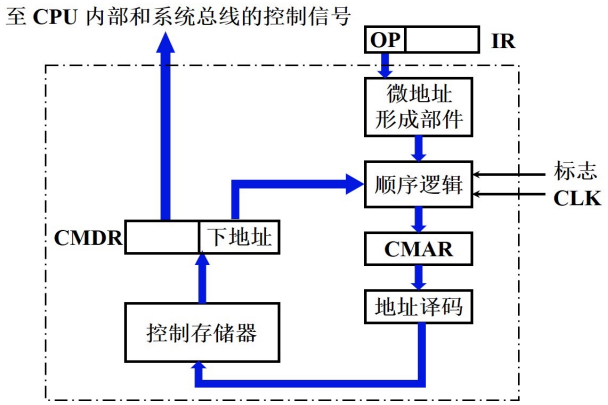
则取指操作需 3 条微指令

Ad (CMDR) ⟶ CMAR

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

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

是关于后续微指令地址 ⟶ 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

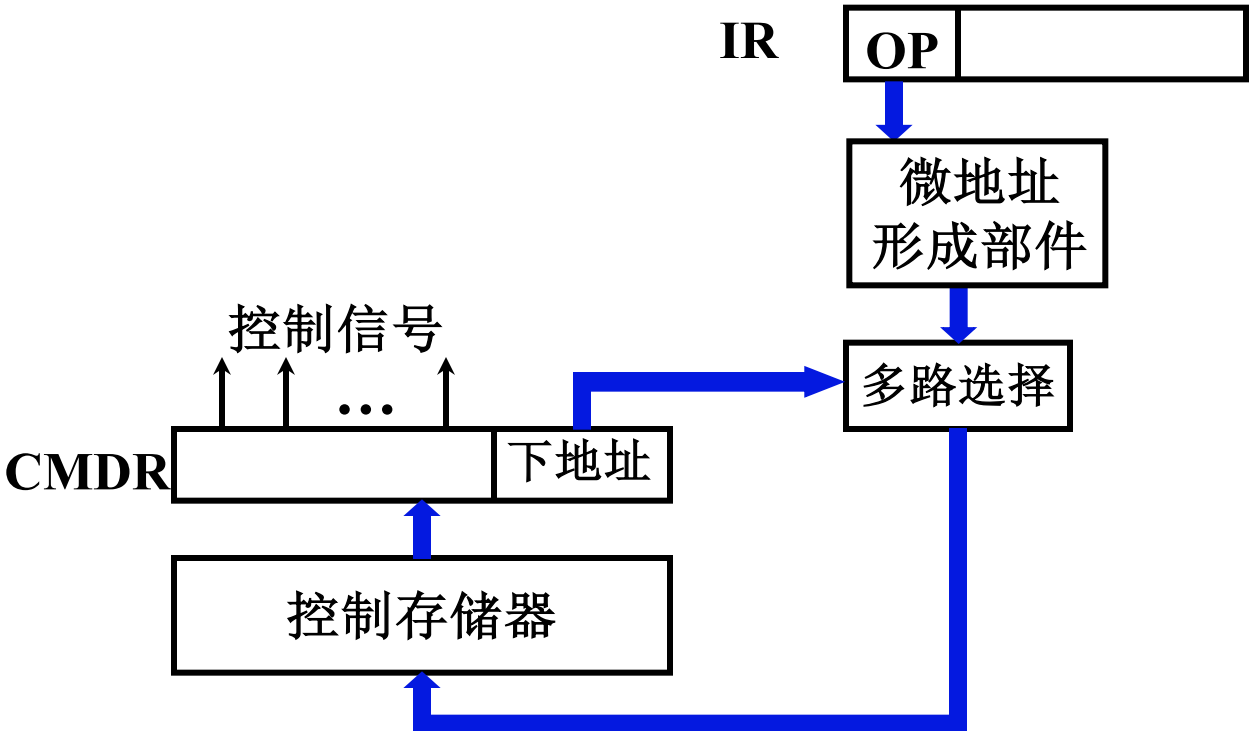
同理　OP (IR) ⟶ 微地址形成部件 ⟶ 控存地址线

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

10.2

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	0	1	0
	02					1					×	×	×	×	×	×
CLA	03								1		0	0	0	0	0	0
COM	04									1	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