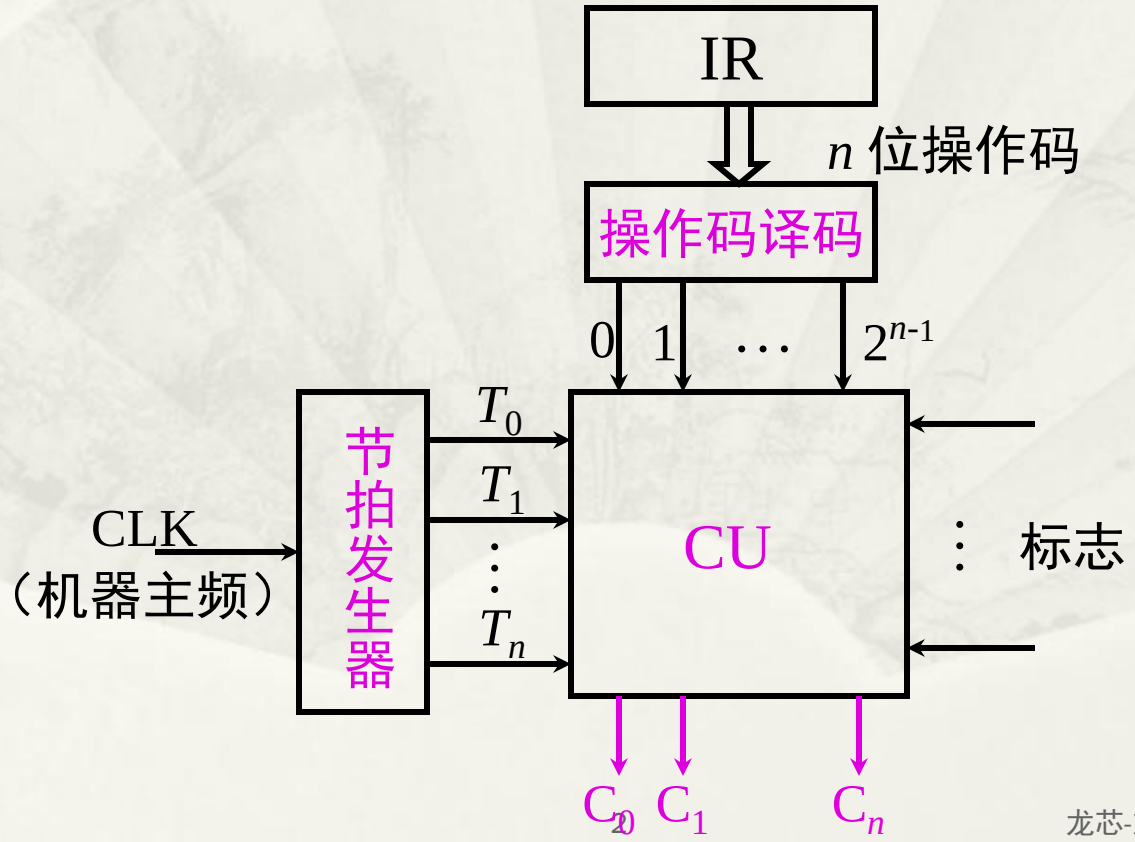
第10章　控制单元的设计

10.1 组合逻辑设计

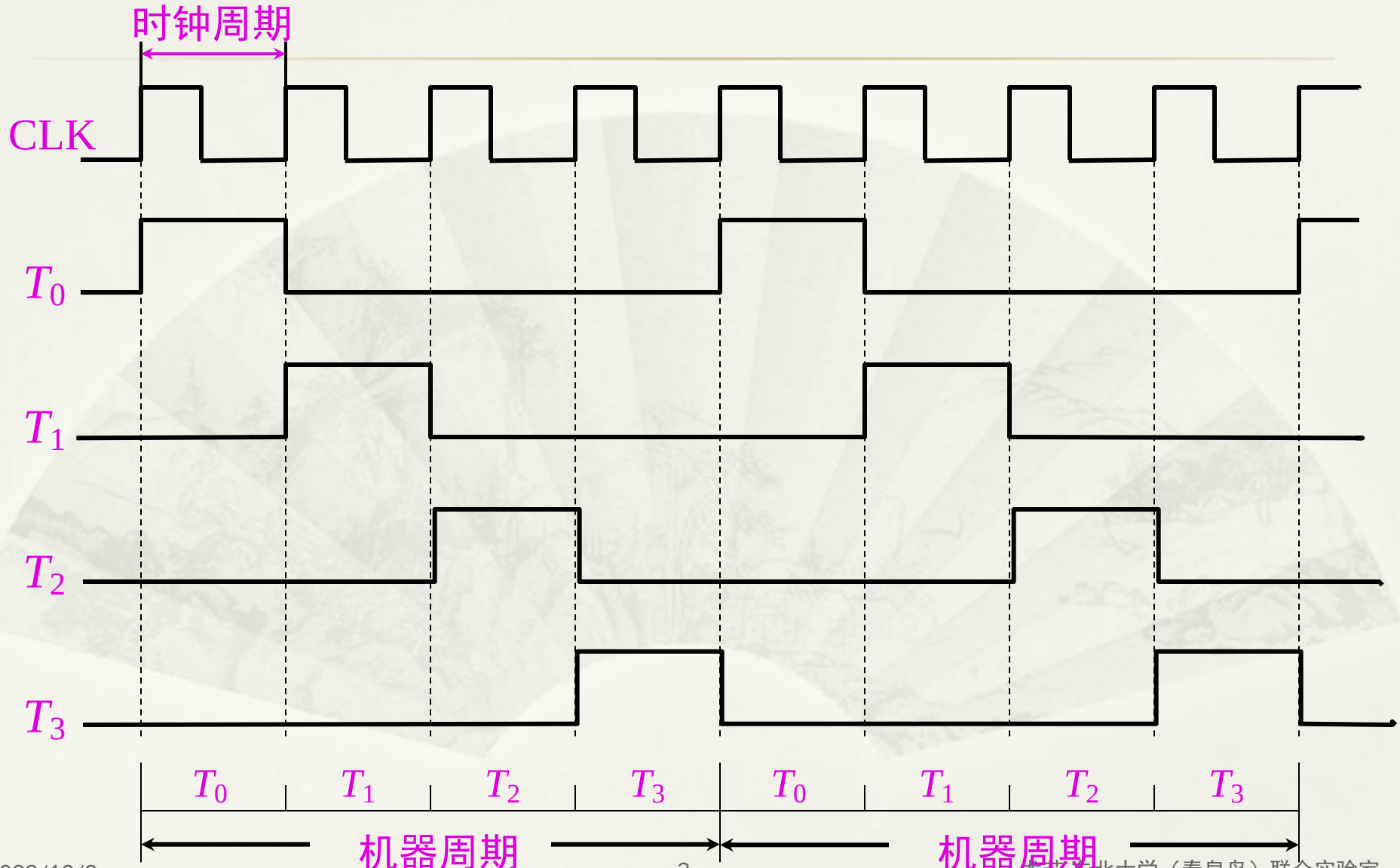
一、组合逻辑控制单元框图

1. CU 外特性



2.节拍信号

10.1



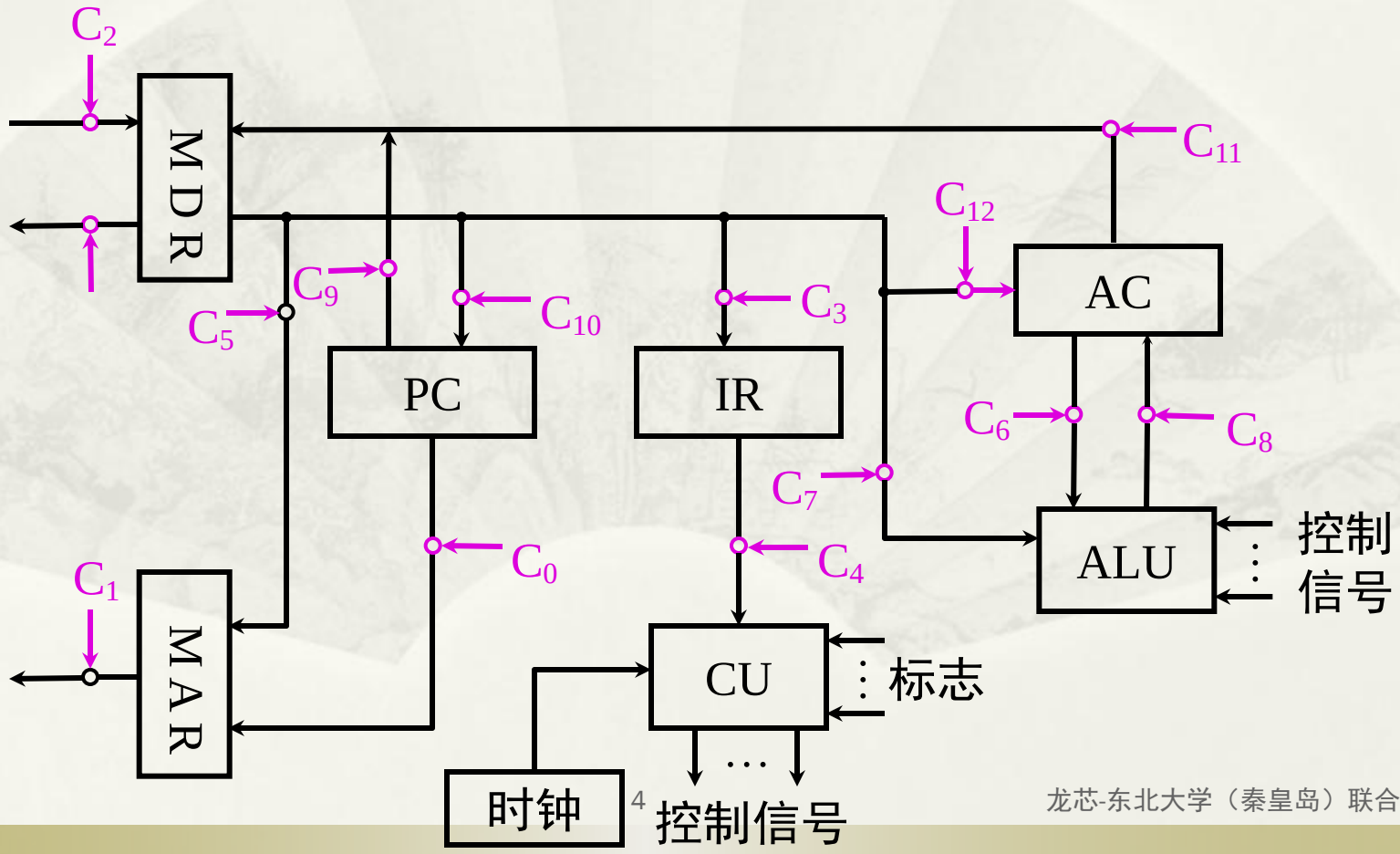
二、微操作的节拍安排

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

10.1

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

10.1

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

* 例10.1 设CPU中各部件及相互连接关系如图10.2所示。图中W是写控制标志，R是读控制标志，R1和R2是暂存器。
* （1）假设要求在取指周期由ALU完成（PC）+1->PC的操作（ALU可以对它的一个源操作数完成加1运算）。要求以最少的节拍写出取指周期全部微操作命令及节拍安排。
* （2）写出指令 $ADD \quad \#\alpha$ 在执行阶段微操作命令和节拍。

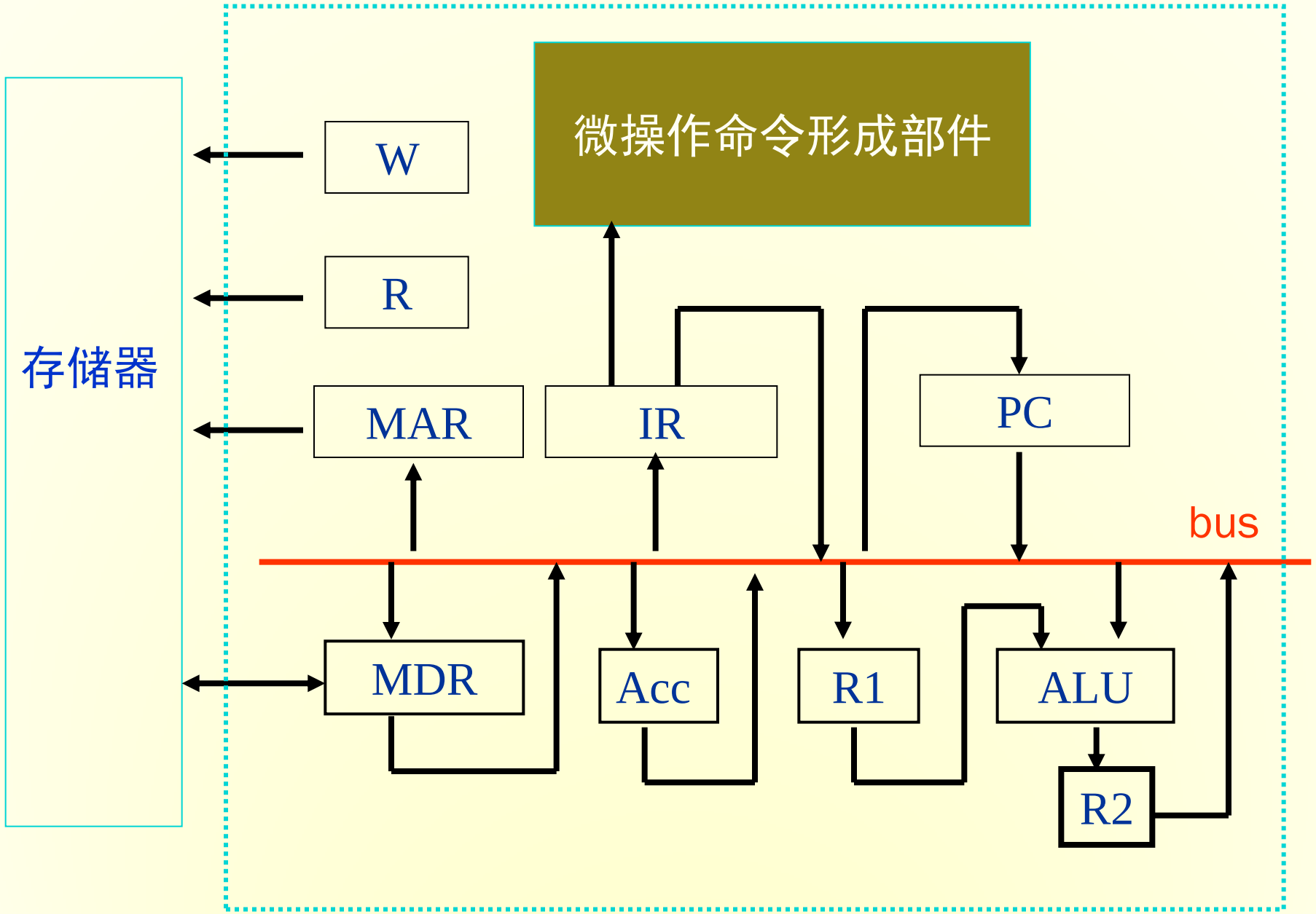


图10.2 CPU内部结构框图

* 例10.2 设CPU内部结构如图10.2所示，且PC有自加1功能。此外还有B、C、D、E、H、L6个寄存器。它们各自的输入端和输出端都和内部总线Bus相连，并分别受控制信号控制。要求写出完成下列指令组合逻辑控制单元所发出的微操作命令及节拍。
* （1）ADD B，C　　；（B）+（C）->B
* （2）SUB E，@H　；(E)-((H))->E
* （3）STA @mem　；ACC->((mem))

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

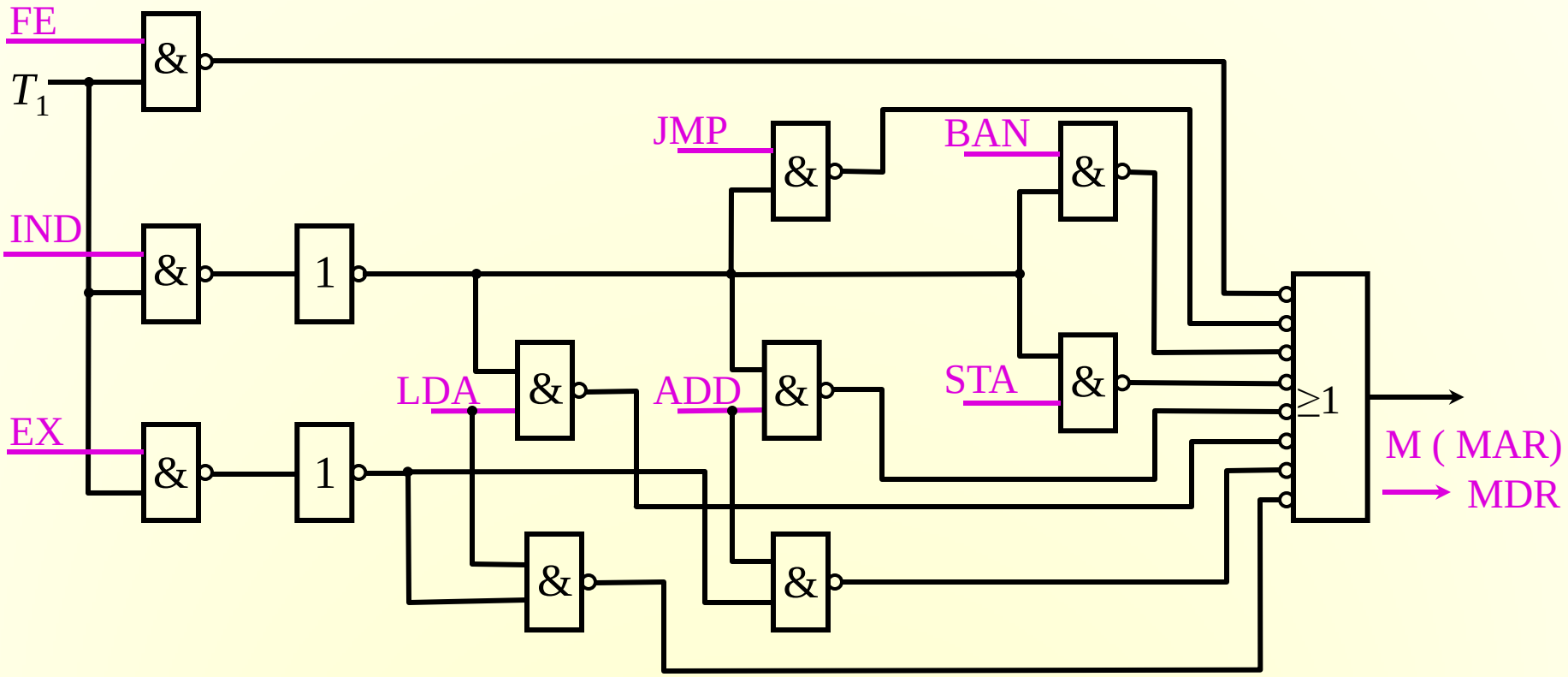
$$M(MAR) \longrightarrow MDR$$

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

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

3. 画出逻辑图

10.1



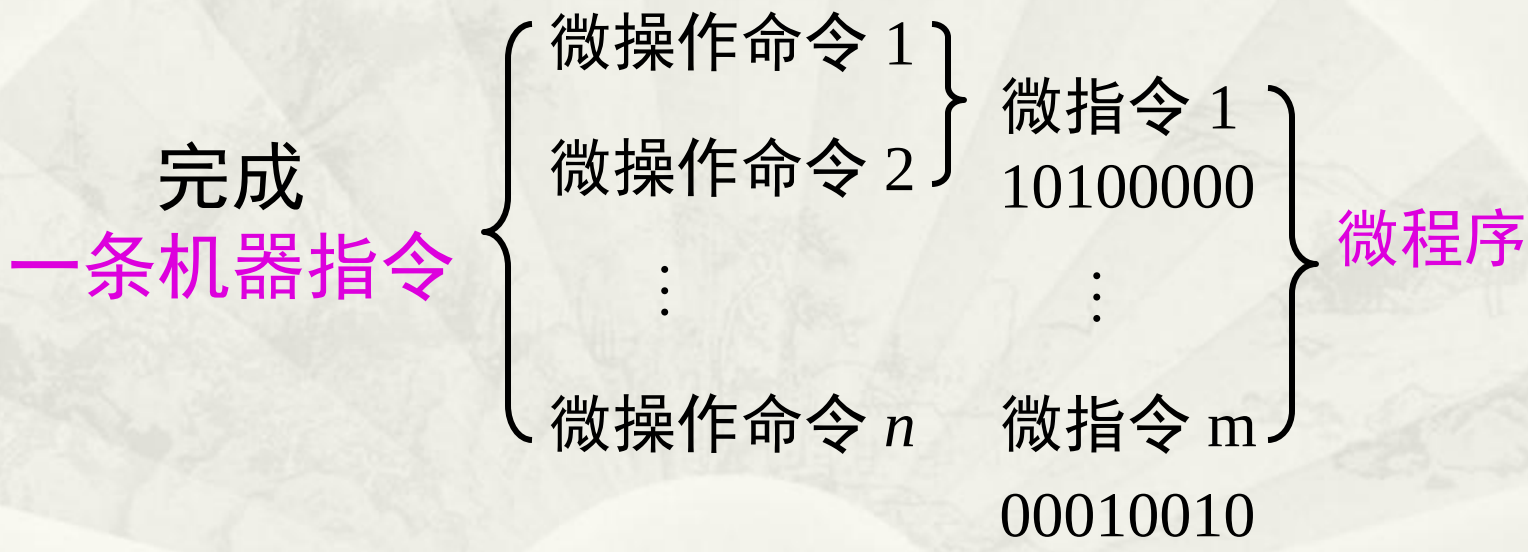
特点

- 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



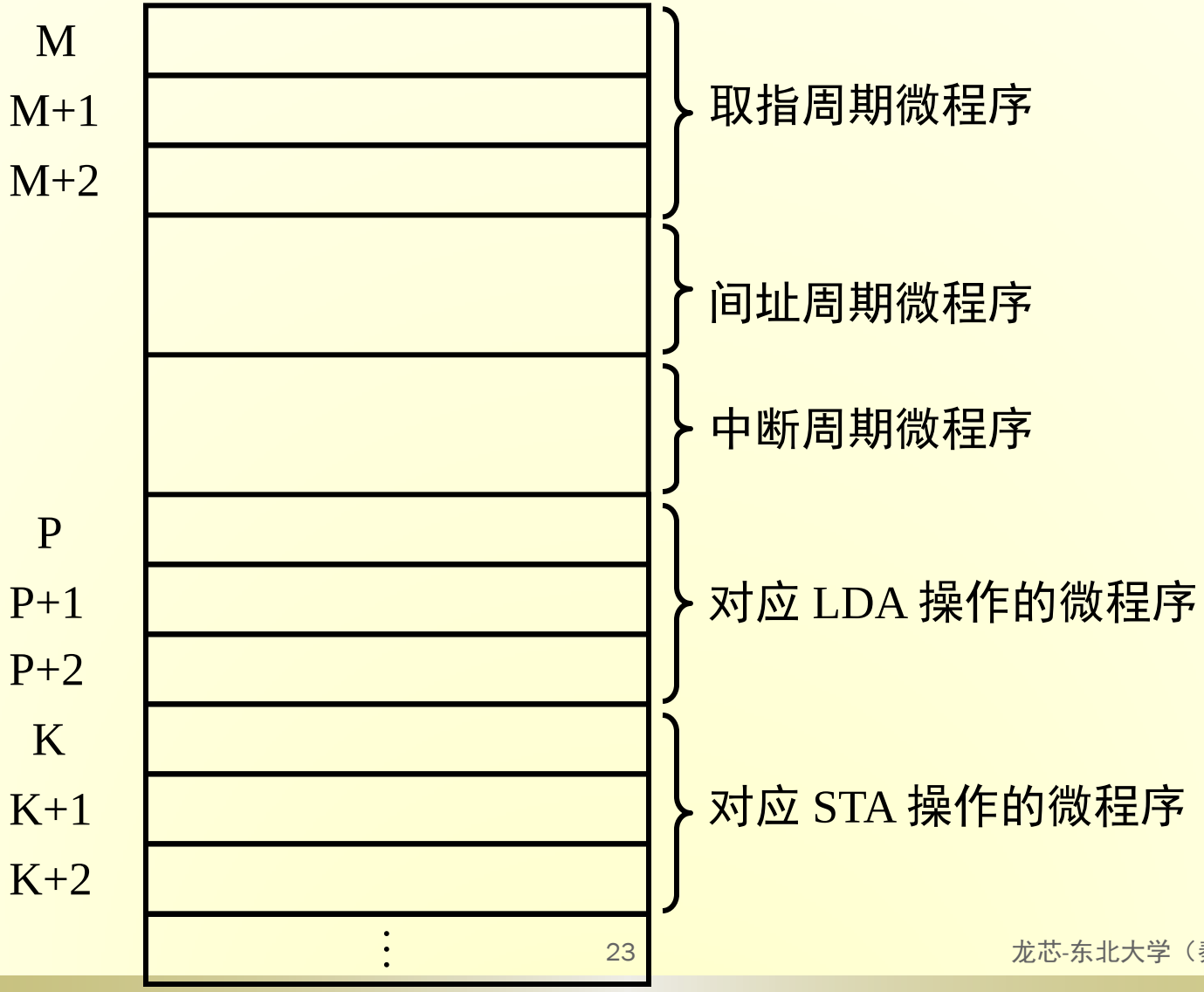
一条机器指令对应一个微程序　　存入 ROM

存储逻辑

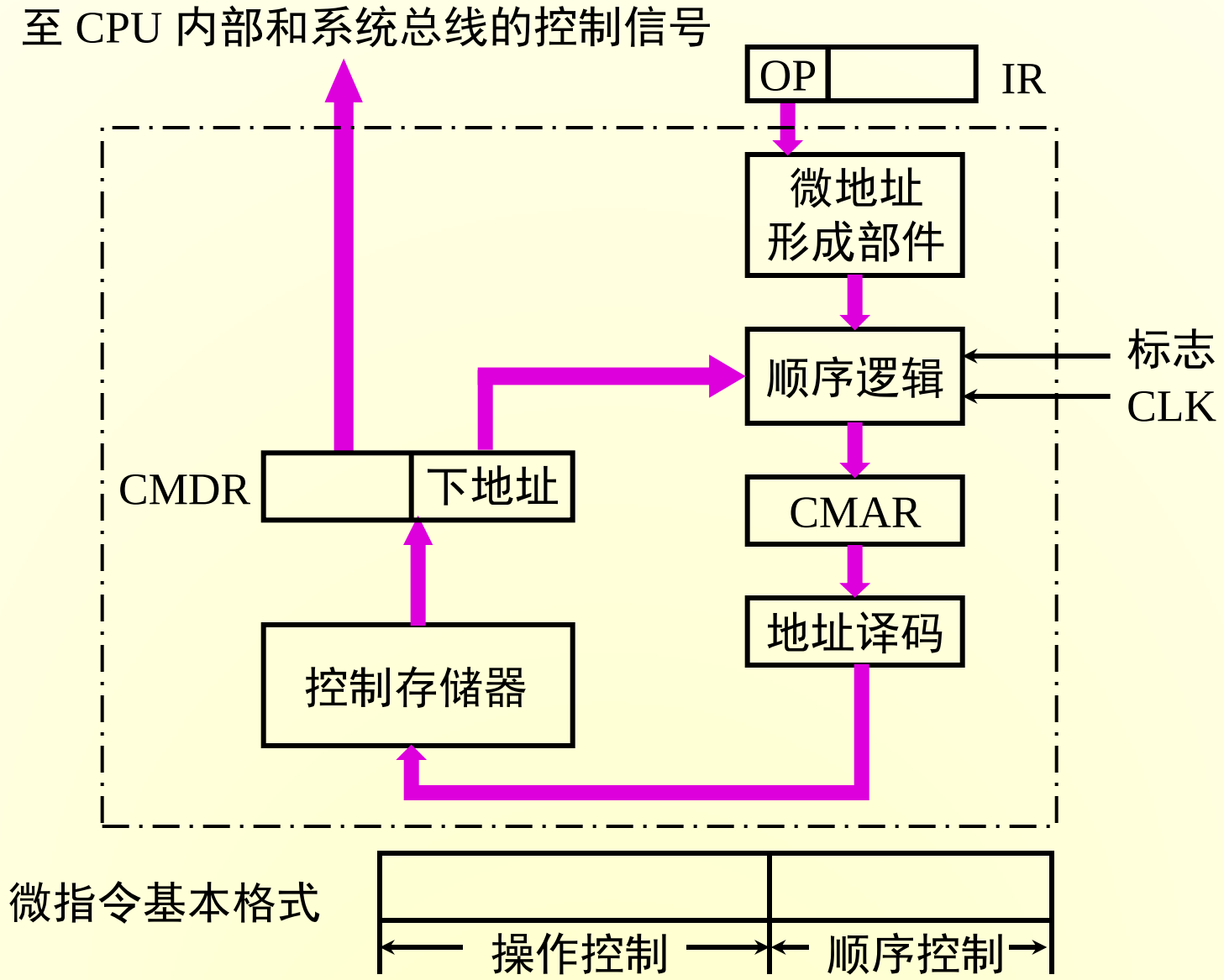
二、微程序控制单元框图及工作原理

10.2

1. 机器指令对应的微程序

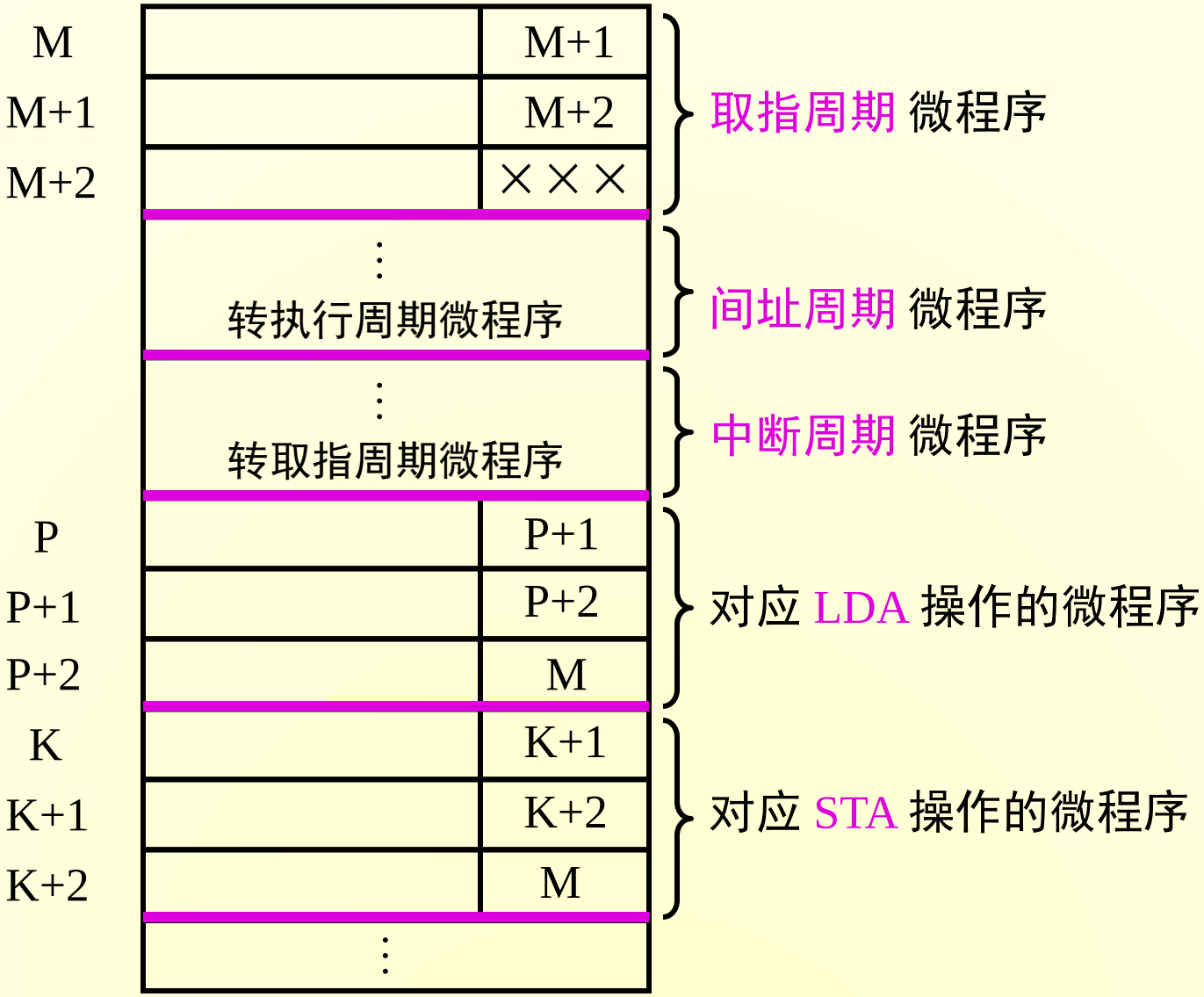


2. 微程序控制单元的基本框图

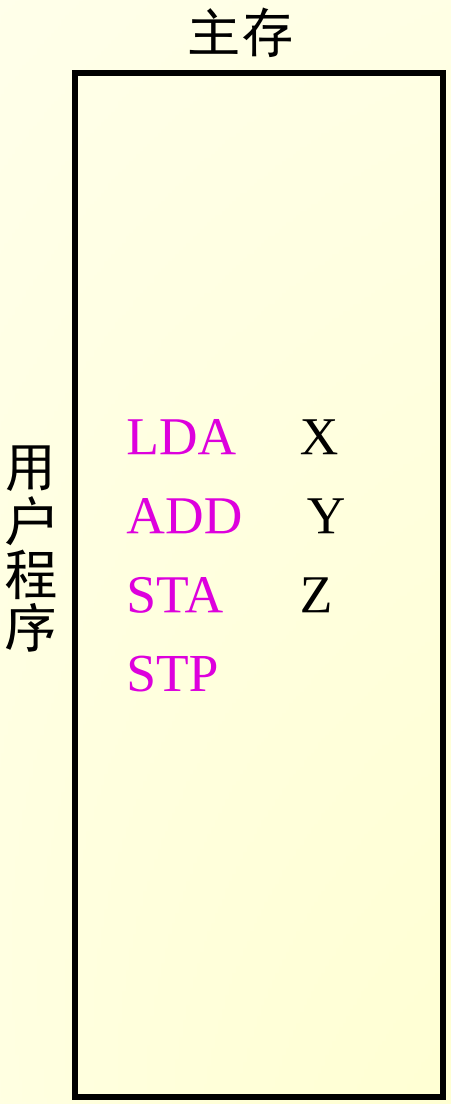


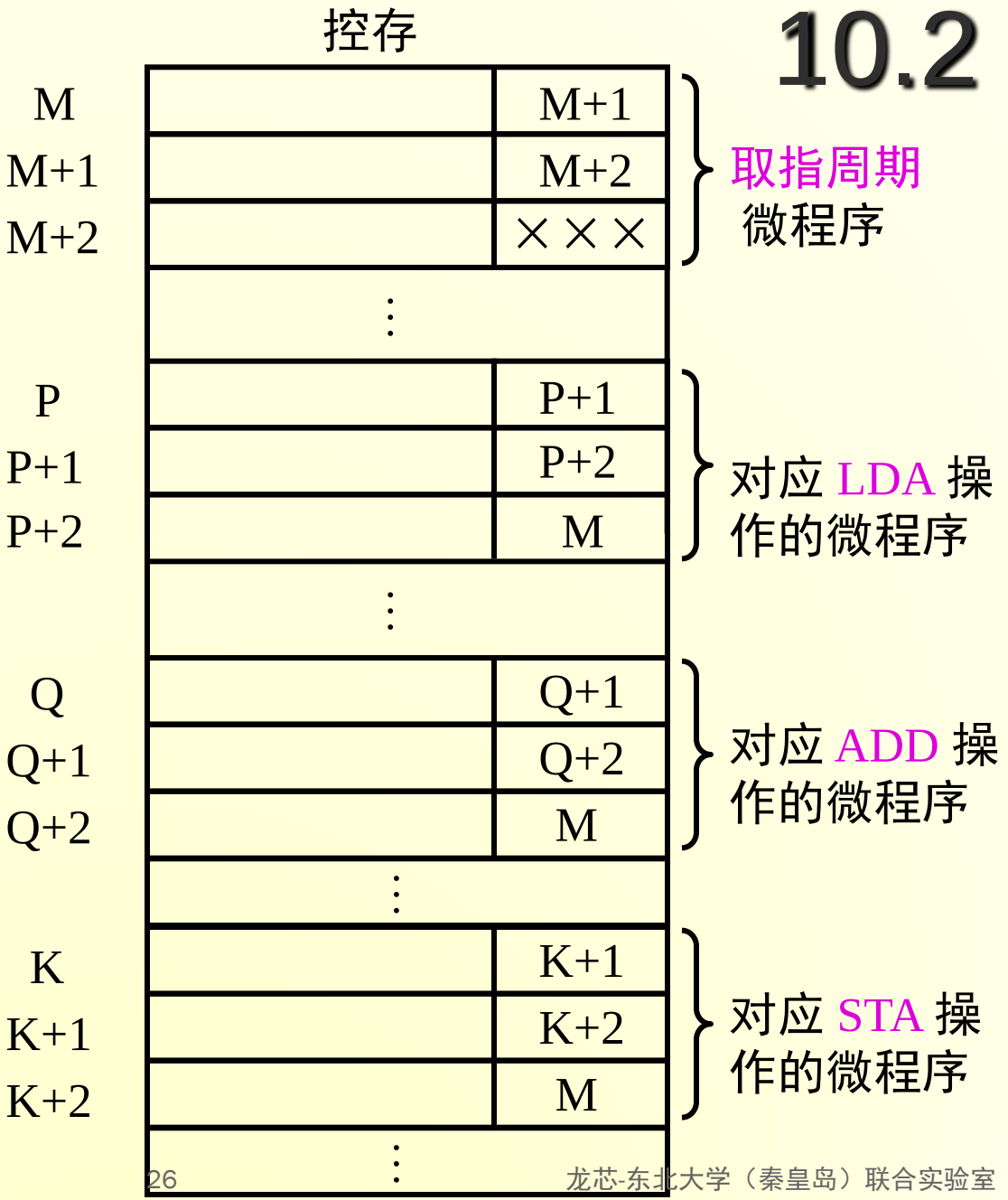
二、微程序控制单元框图及工作原理

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

10.2

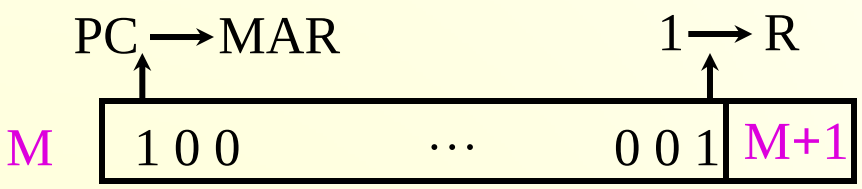
(1) 取指阶段　　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶CMDR

由 CMDR 发命令

形成下条微指令地址　M + 1

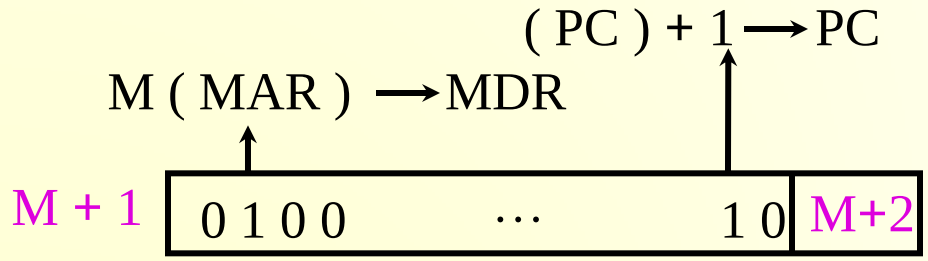


Ad (CMDR) ⟶CMAR

CM (CMAR) ⟶CMDR

由 CMDR 发命令

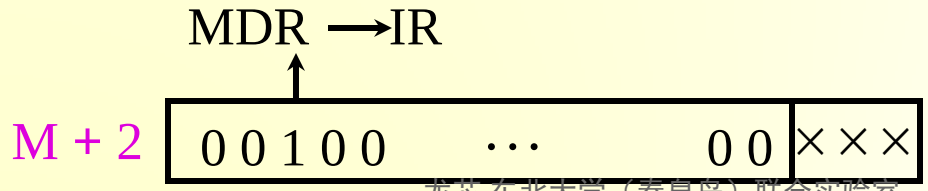
形成下条微指令地址　M + 2



Ad (CMDR) ⟶CMAR

CM (CMAR) ⟶CMDR

由 CMDR 发命令



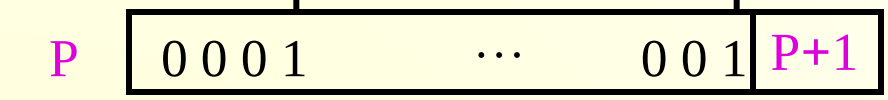
(2) 执行阶段　执行 LDA 微程序

10.2

OP (IR) ⟶微地址形成部件 ⟶CMAR　　(P ⟶ CMAR)

CM (CMAR) ⟶CMDR

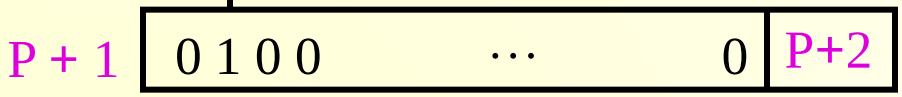
Ad (IR) ⟶MAR　　1⟶ R



由 CMDR 发命令
形成下条微指令地址 P + 1
Ad (CMDR) ⟶CMAR

CM (CMAR) ⟶CMDR

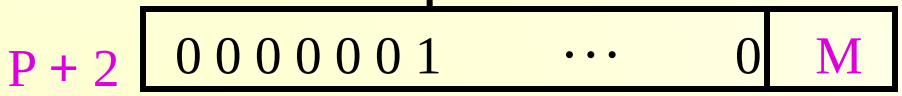
M (MAR) ⟶MDR



由 CMDR 发命令
形成下条微指令地址 P + 2
Ad (CMDR) ⟶CMAR

CM (CMAR) ⟶CMDR

MDR ⟶ AC



由 CMDR 发命令
形成下条微指令地址 M
Ad (CMDR) ⟶CMAR

(M ⟶CMAR)

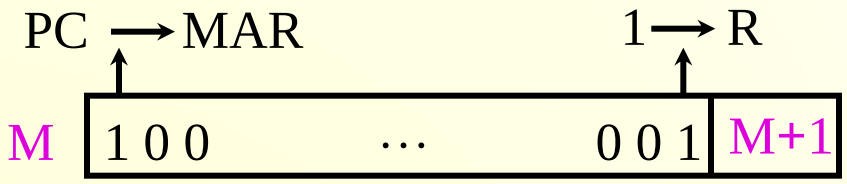
(3) 取指阶段　执行取指微程序

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

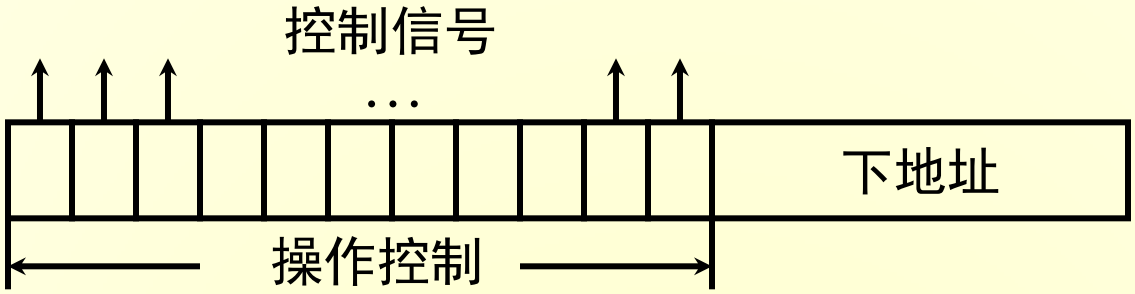
三、微指令的编码方式（控制方式）

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



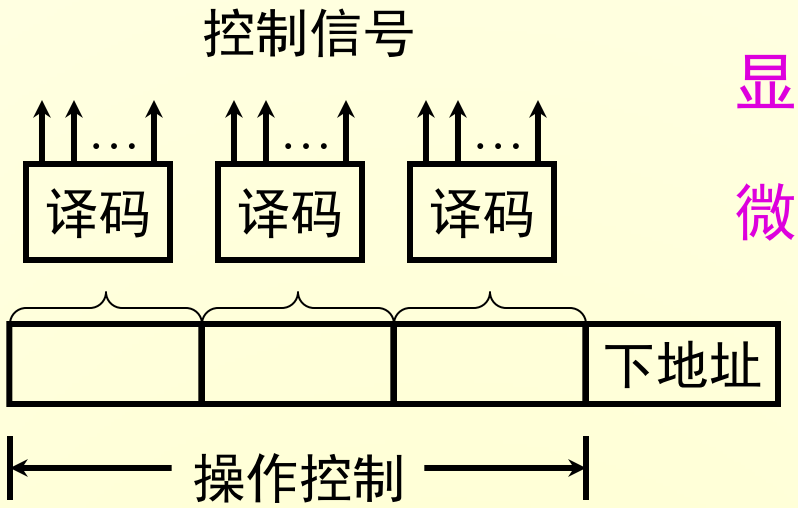
速度最快

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



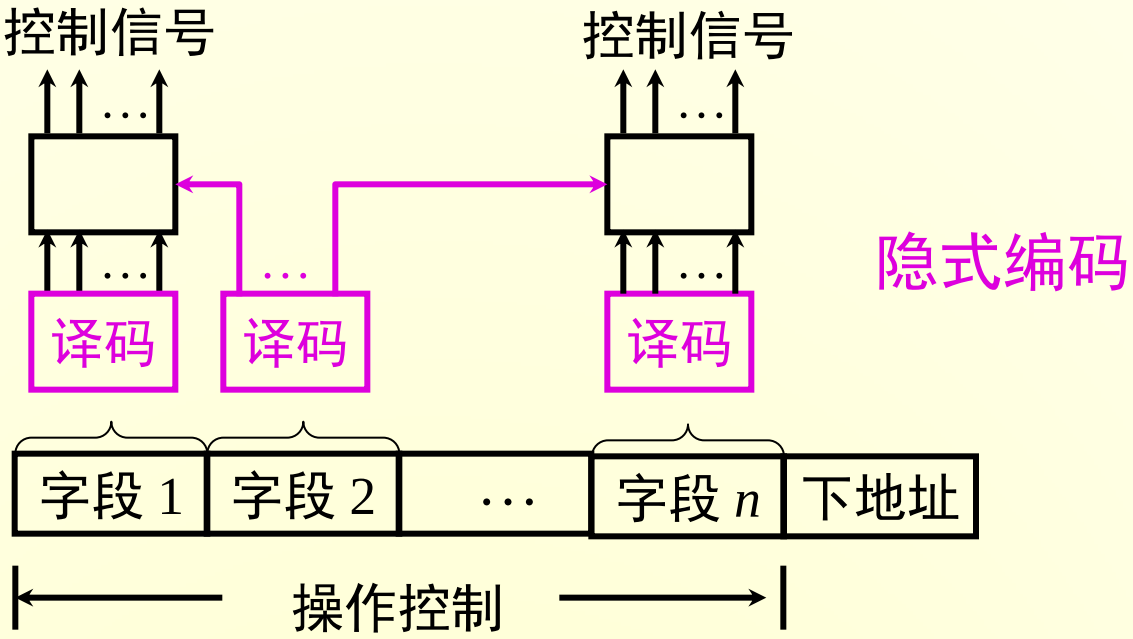
显式编码

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

例10.4 某机的微指令格式中，共有8个控制字段，每个字段可分别激活5，8，3，16，1，7，25，4种控制信号。分别采用直接编码和字段直接编码方式设计微指令的操作控制字段，并说明两种方式的操作控制字段各取几位。

- 解：（1）采用直接编码方式，微指令的操作控制字段的总位数等于控制信号数，即：5+8+3+16+1+7+25+4=69
- (2)采用字段直接编码方式，需要的控制位少，根据题目给出的10个控制字段及各段可激活的控制信号数，再加上每个控制字段至少要留一个码字表示不激活任何一条控制线，即 微指令的8个控制字段分别需要

四、微指令序列地址的形成

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

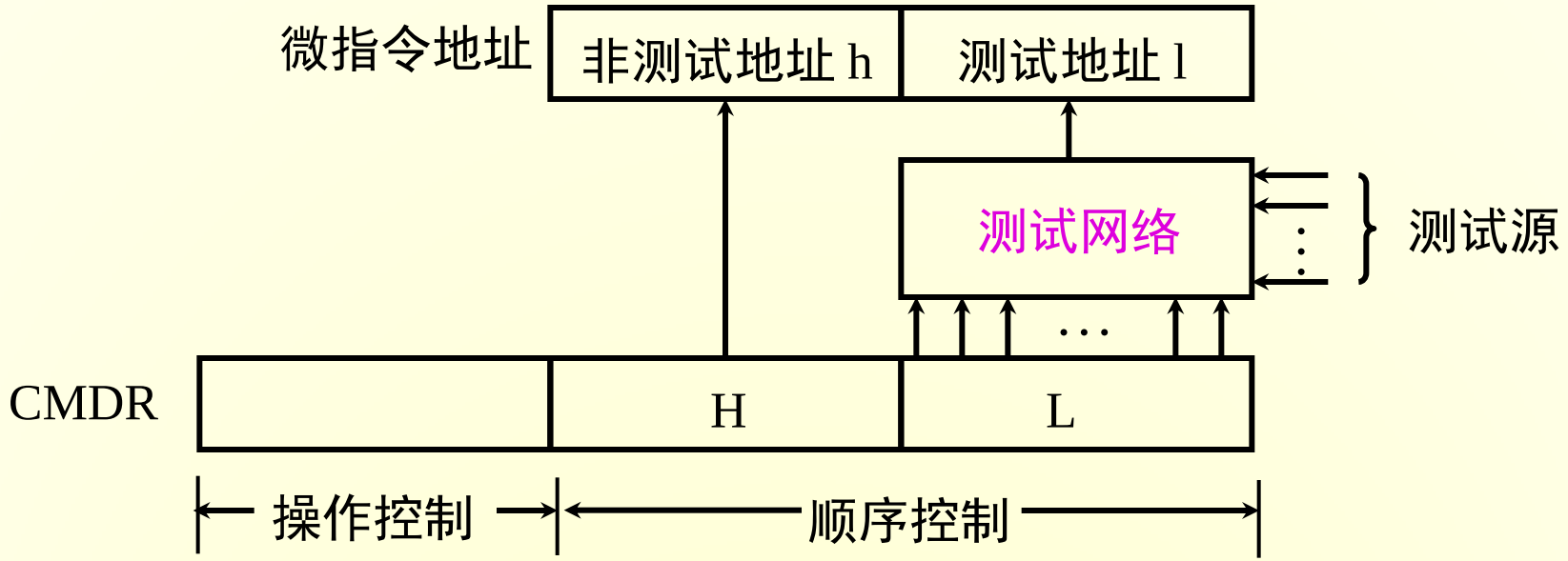
操作控制字段	转移方式	转移地址

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

10.2



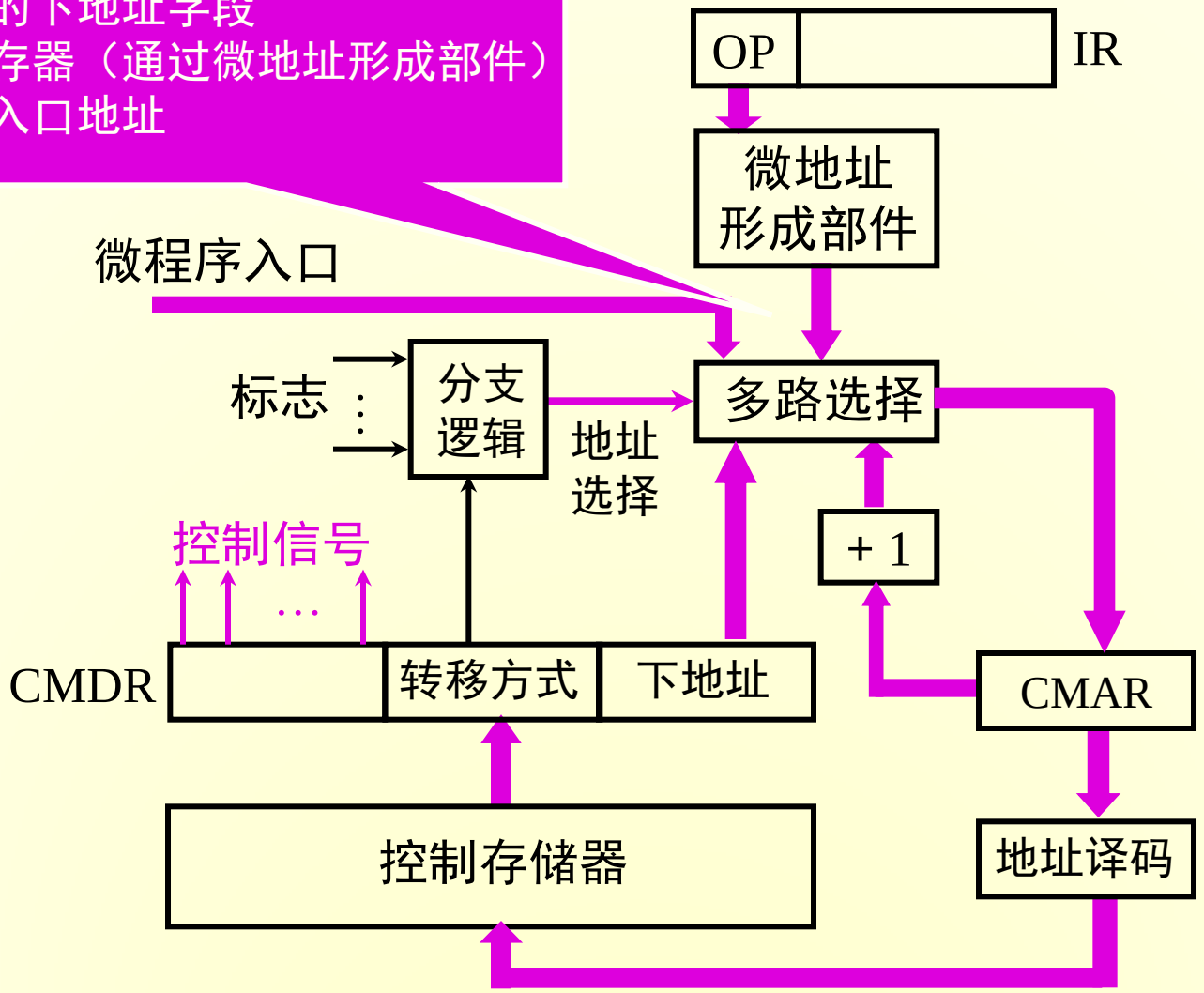
6. 由硬件产生微程序入口地址

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

- （CMAR+1）->CMAR
- 微指令的下地址字段
- 指令寄存器（通过微地址形成部件）
- 微程序入口地址



五、微指令格式

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

- 例10.5 某微程序控制器中，采用水平型直接控制（编码）方式的微指令格式，后续微指令地址由微指令的下地址字段给出。已知机器共有28个微命令，6个互斥的可判定的外部条件，控存的容量是512×40位。试设计微指令格式，并说明理由。

* 解：水平型微指令由操作控制字段、判别测试字段和下地址字段三部分组成。因为微指令采用直接控制方式，又由于后续微指令地址由下地址字段给出，姑其下地址字段的位数可根据控存的容量定为9位。当微程序出现分支时，后续微指令地址的形成取决于状态条件，6个互斥的可判定外部条件，编码3位。

操作控制	判断	下地址
28	3	9

* 例10.6 某机共有52个微操作控制信号，构成5个相斥类的微命令组，各组分别包含5，8，2，15，22个微命令。已知可判定的外部条件有两个，微指令字长28位。
* （1）按水平型微指令格式设计微指令，要求微指令的下地址字段直接给出后续微指令地址。
* （2）指出控存的容量。

* 解: (1) 根据5个相斥类的微命令组，各组分别包含5、8、2、15、22个微命令，考虑到每组必须增加一种不发命令的情况，条件测试字段应包含一种不转移情况，则5个控制字段分别需要6、9、3、16、23种状态，对应3、4、2、4、5位（共18位），条件测试字段取2位。下地址字段28-18-2=8

5个微命令	8个微命令	2个微命令	15个微命令	22个微命令	2个判定条件	
3位	4位	2位	4位	5位	条件测试（2位）	下地址（8位）

(2)根据下地址字段为8位，微指令字长28位，控存容量为256×28位。

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

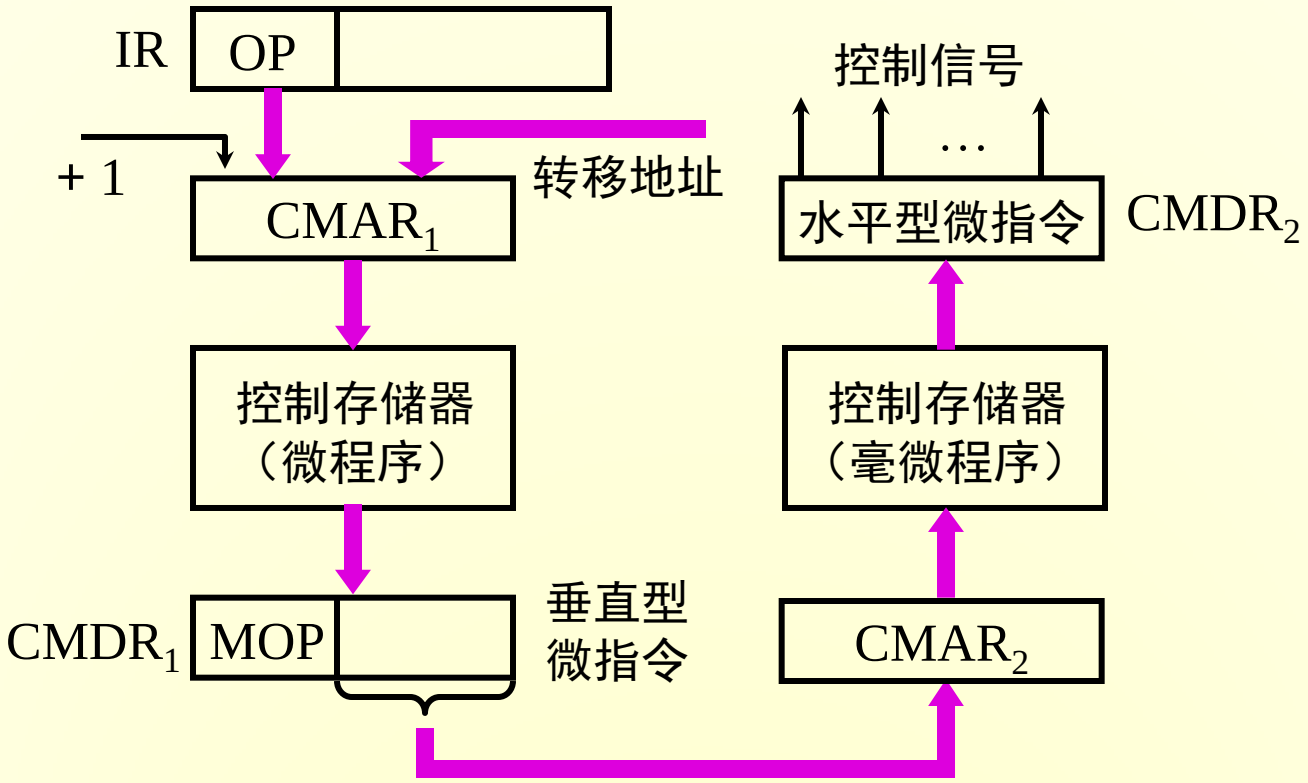
微程序设计　　用 微程序解释机器指令

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

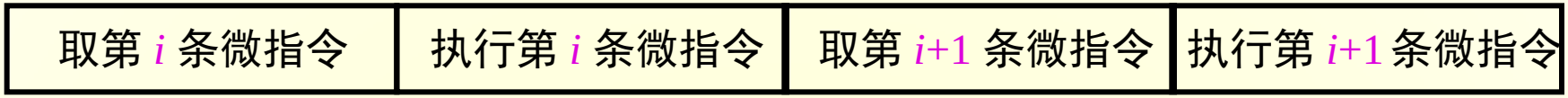
2. 毫微程序控制存储器的基本组成

10.2

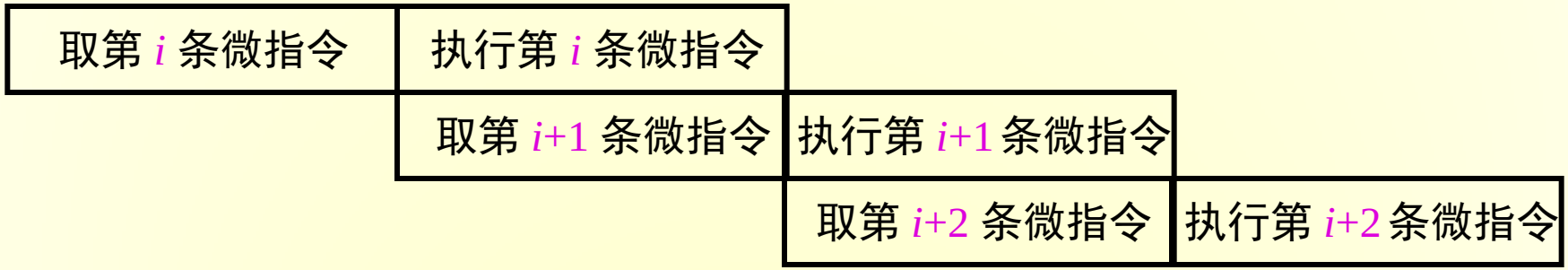


八、串行微程序控制和并行微程序控制

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_2 如何安排一条微指令？

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

10.2

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)　　$AC_0 \longrightarrow AC_0$

T_1　　Ad (CMDR) ⟶ CMAR

④ CSL 指令

T_0　　R (AC) ⟶ L (AC)　　$AC_0 \longrightarrow AC_n$

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

10.2

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

10.2

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

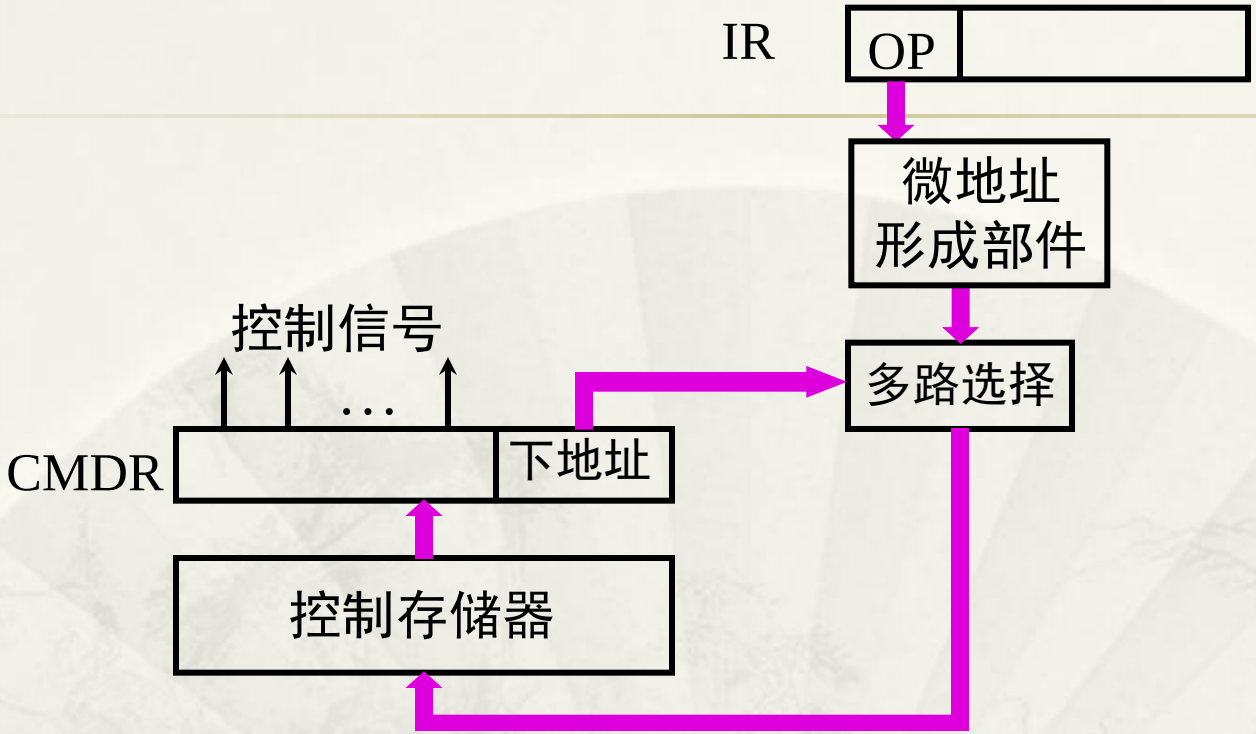
同理　OP (IR) ⟶ 微地址形成部件 ⟶ 控存地址线

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

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										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