第10章 控制单元的设计

> 课件来源：《计算机组成原理》课件第10章（Slide 498–545，含 10.1 组合逻辑设计、10.2 微程序设计），为本章唯一依据；个别衔接性常识已标注「（补充常识）」。本章是控制器设计的收官章，硬布线设计步骤、微指令编码/格式/地址形成、微程序控制器组成与字长计算是期末选择、填空、简答和综合设计题的高频出处。

0. 本章考点清单

考点	常见题型	重要度
微程序设计举例：微操作节拍安排、微指令字长计算与优化（20/38→26 位→30 位）	综合设计、计算	★★★
微指令编码方式：直接/字段直接/字段间接/混合	选择、计算	★★★
微指令格式：水平型 vs 垂直型及比较	选择、简答	★★★
微地址形成方式（下地址字段、操作码、计数器、分支、测试网络、硬件入口）	选择、填空	★★★
微程序控制器组成（CM、CMAR、CMDR、微地址形成部件）与工作原理	填空、简答	★★★
硬布线 vs 微程序控制器对比	简答	★★★
微操作节拍安排三原则；取指/间址/执行/中断周期微操作	填空、简答	★★
组合逻辑设计三步骤：操作时间表→最简表达式→逻辑图	计算、综合	★★
中断周期微操作与中断隐指令	填空、简答	★★
微程序基本概念：微操作/微指令/微程序、存储逻辑、Wilkes	选择	★★
微指令码点表识读（八进制地址、操作控制位、下地址字段）	综合	★★
CU 外特性与节拍信号（时钟周期/机器周期）	填空	★
静态/动态微程序设计、毫微程序设计	填空	★
串行/并行微程序控制	简答	★

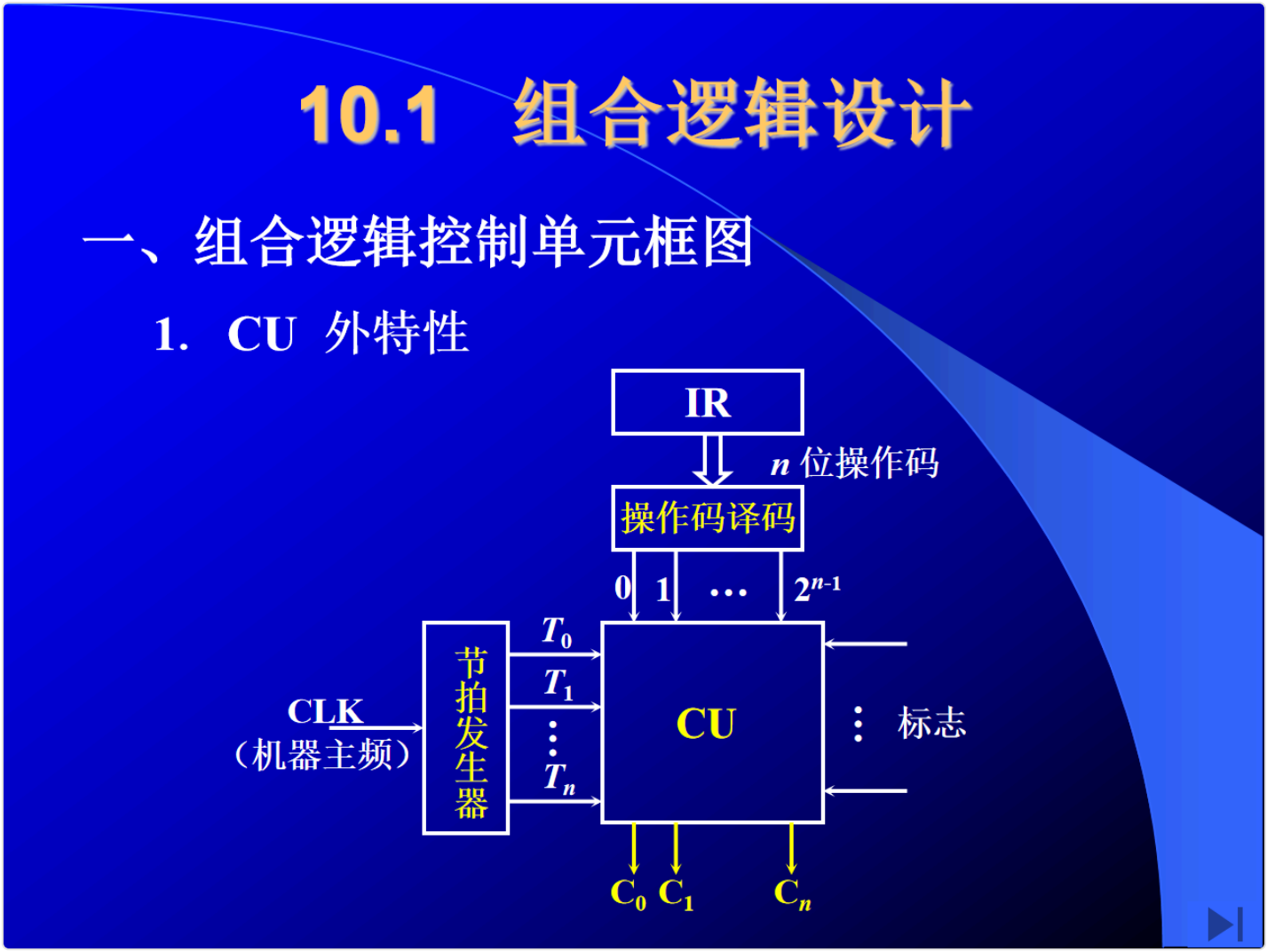
1. 组合逻辑（硬布线）控制单元框图与节拍信号

1.1 CU 外特性

组合逻辑控制单元（CU）是一个组合逻辑网络，其输入/输出关系即「外特性」（课件 p.499）：

- 输入①：IR 中 n 位操作码经**操作码译码**后产生的 2^n 条译码线（0、1、…、2^{n-1}），指明当前是哪条指令；
- 输入②：**节拍发生器**在机器主频 CLK 驱动下产生的节拍信号 T_0、T_1、…、T_n，指明当前处于哪个节拍；
- 输入③：来自执行部件的**标志**（状态条件）；
- 输出：微操作命令信号（控制信号）C_0、C_1、…、C_n，送往 CPU 内部和系统总线。

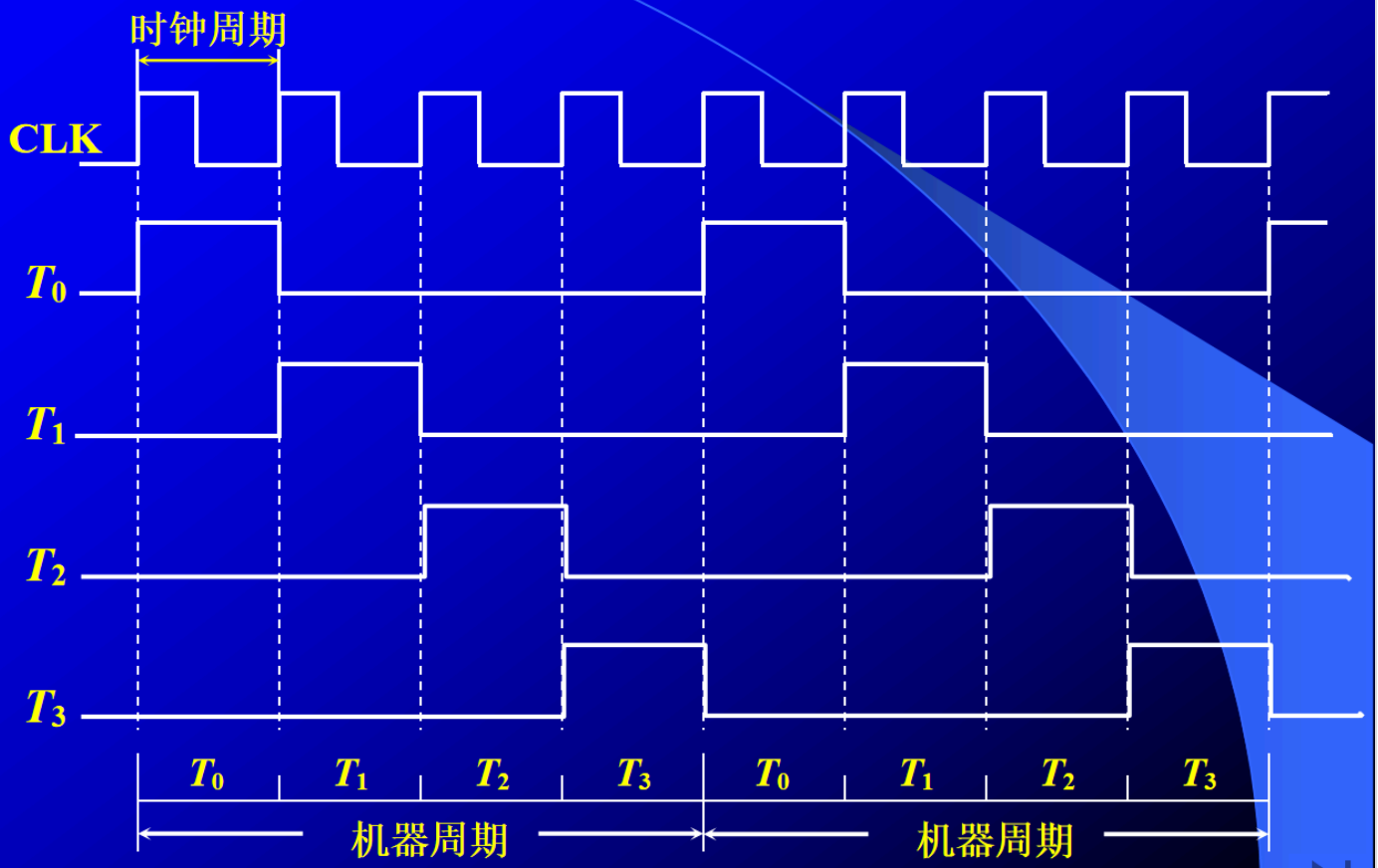
即：控制信号 = f(操作码译码, 节拍, 标志)。

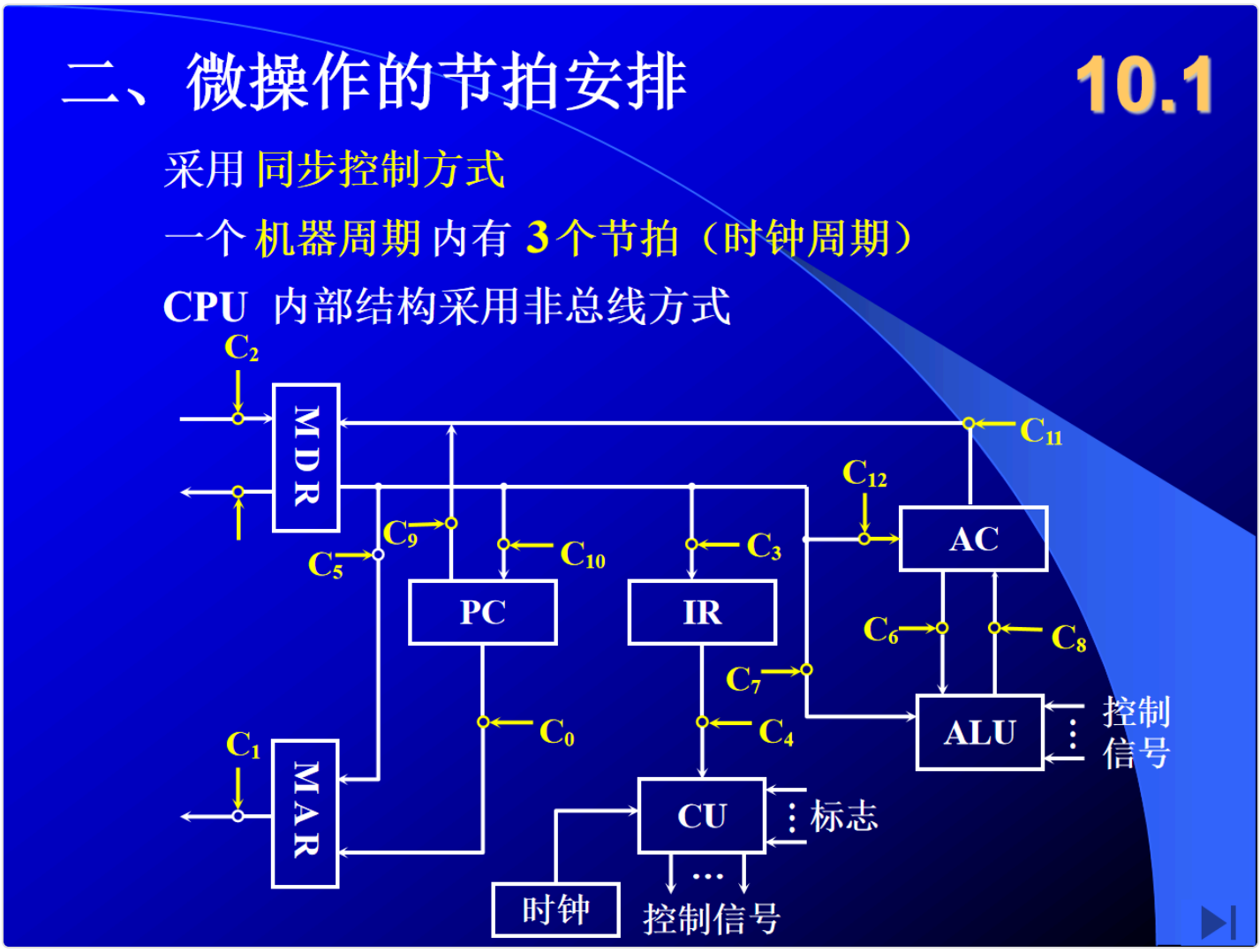


1.2 节拍信号：时钟周期与机器周期

- **节拍 = 时钟周期**：CLK 的一个周期对应一个节拍信号（T_0 或 T_1 …）的高电平宽度（课件 p.500）。
- **机器周期 = 若干节拍**：p.500 的示意图以 $T_0 \sim T_3$ 共 4 个节拍构成一个机器周期为例；而本章设计举例中约定**一个机器周期含 3 个节拍**（课件 p.501）。机器周期含几个节拍由设计约定决定，不要死记。
- 本章举例的三个前提（课件 p.501）：① 采用**同步控制方式**；② CPU 内部结构采用**非总线方式**（专用通路，故多个微操作可并行）；③ 一个机器周期内有 **3 个节拍**（时钟周期）。

2. 节拍信号





要点：非总线结构是「一个节拍内安排多个微操作」的硬件前提；若为单总线结构，同一节拍内通常只能完成一次总线传送（补充常识）。

易错警示：CU 的输入不包括「机器指令的地址码」，地址码 Ad(IR) 是作为数据通路操作数使用的，进入 CU 的只有操作码译码结果、节拍和标志（课件 p.499）。

2. 微操作的节拍安排

2.1 安排微操作时序的三原则（课件 p.502）

1. **原则一**：微操作的**先后顺序不得随意更改**（有依赖关系的必须保序）；
2. **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（可并行）；
3. **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**。

2.2 取指周期（课件 p.503）

节拍	微操作	依据
T_0	PC→MAR，1→R	原则二（被控对象 MAR 与读命令不同）
T_1	M(MAR)→MDR，(PC)+1→PC	原则二（MDR 与 PC 不同）
T_2	MDR→IR，OP(IR)→ID	原则三（时间短，允许先送 IR 再译码）

2.3 间址周期（课件 p.503）

节拍	微操作
T_0	Ad(IR)→MAR，1→R
T_1	M(MAR)→MDR
T_2	MDR→Ad(IR)（有效地址回送 IR 地址字段）

2.4 执行周期：10 条指令的微操作（课件 p.504–506）

指令	T_0	T_1	T_2
① CLA	—	—	0→AC
② COM	—	—	$\overline{AC}$→AC（AC 取反送 AC）
③ SHR	—	—	L(AC)→R(AC)，AC_0→AC_0（右移，符号位不变）
④ CSL	—	—	R(AC)→L(AC)，AC_0→AC_n（循环左移）
⑤ STP	—	—	0→G（停机，G 为运行标志）
⑥ ADD X	Ad(IR)→MAR，1→R	M(MAR)→MDR	(AC)+(MDR)→AC
⑦ STA X	Ad(IR)→MAR，1→W	AC→MDR	MDR→M(MAR)
⑧ LDA X	Ad(IR)→MAR，1→R	M(MAR)→MDR	MDR→AC
⑨ JMP X	—	—	Ad(IR)→PC
⑩ BAN X	—	—	A_0·Ad(IR) + $\bar{A}_0$·(PC)→PC

说明：A_0 为 AC 的符号位，BAN 为「AC 为负则转移」：A_0=1 时 Ad(IR)→PC，A_0=0 时 (PC)→PC（顺序执行）（课件 p.506）。注意在 10.1 组合逻辑举例中，单微操作指令的执行微操作统一安排在**执行周期 T_2**（课件 p.504–506、p.513）；而 10.2 微程序举例中同样的微操作安排在微程序的 T_0（课件 p.537），两者前提不同，答题按题设。

2.5 中断周期（课件 p.507）

节拍	微操作	备注
T_0	0→MAR，1→W	同时**硬件关中断**
T_1	PC→MDR	断点送 MDR
T_2	MDR→M(MAR)，向量地址→PC	断点存入 0 号单元，形成服务程序入口

上述操作由**中断隐指令**完成——它不是指令系统中的指令，而是硬件自动完成的操作序列（课件 p.507；「断点保护」的目的为补充常识）。

易错警示：中断周期的「硬件关中断」发生在 T_0，与写存储器命令 1→W 同节拍；不要把它漏掉或移到 T_2（课件 p.507）。

3. 组合逻辑设计三步骤

3.1 步骤一：列出操作时间表（课件 p.508–513）

操作时间表 = 以「工作周期标记（FE 取指 / IND 间址 / EX 执行）× 节拍 × 状态条件」为行、以各条指令为列的表格，凡某指令在某节拍需要某微操作，就在对应格填 1（课件 p.511–513 为填 1 后的表）。课件以 CLA、COM、ADD、STA、LDA、JMP 六列为例。

FE 取指周期（课件 p.508、511）：

节拍	状态条件	微操作命令	CLA	COM	ADD	STA	LDA	JMP
T_0		PC→MAR	1	1	1	1	1	1
T_0		1→R	1	1	1	1	1	1
T_1		M(MAR)→MDR	1	1	1	1	1	1
T_1		(PC)+1→PC	1	1	1	1	1	1
T_2		MDR→IR	1	1	1	1	1	1
T_2		OP(IR)→ID	1	1	1	1	1	1
T_2	I（间址特征）	1→IND	1	1	1	1	1	1
T_2	$\bar{I}$	1→EX	1	1	1	1	1	1

IND 间址周期（课件 p.509、512）：T_0 Ad(IR)→MAR、1→R；T_1 M(MAR)→MDR；T_2 MDR→Ad(IR)；以及 1→EX；仅 ADD、STA、LDA、JMP（及 BAN）等带地址且可间址的指令填 1。

EX 执行周期（课件 p.510、513）：

节拍	微操作命令	CLA	COM	ADD	STA	LDA	JMP
T_0	Ad(IR)→MAR			1	1	1	
T_0	1→R			1		1	
T_0	1→W				1		
T_1	M(MAR)→MDR			1		1	
T_1	AC→MDR				1		
T_2	(AC)+(MDR)→AC			1			
T_2	MDR→M(MAR)				1		
T_2	MDR→AC					1	
T_2	0→AC	1					
T_2	$\overline{AC}$→AC		1				
T_2	Ad(IR)→PC						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
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						
		$\overline{I}$	1 ⟶ EX						

间址特征

3.2 步骤二：写出微操作命令的最简表达式（课件 p.514）

把操作时间表中某微操作命令所有填 1 的格对应的「周期标记·节拍·指令」相或，再化简。以 M(MAR)→MDR 为例：

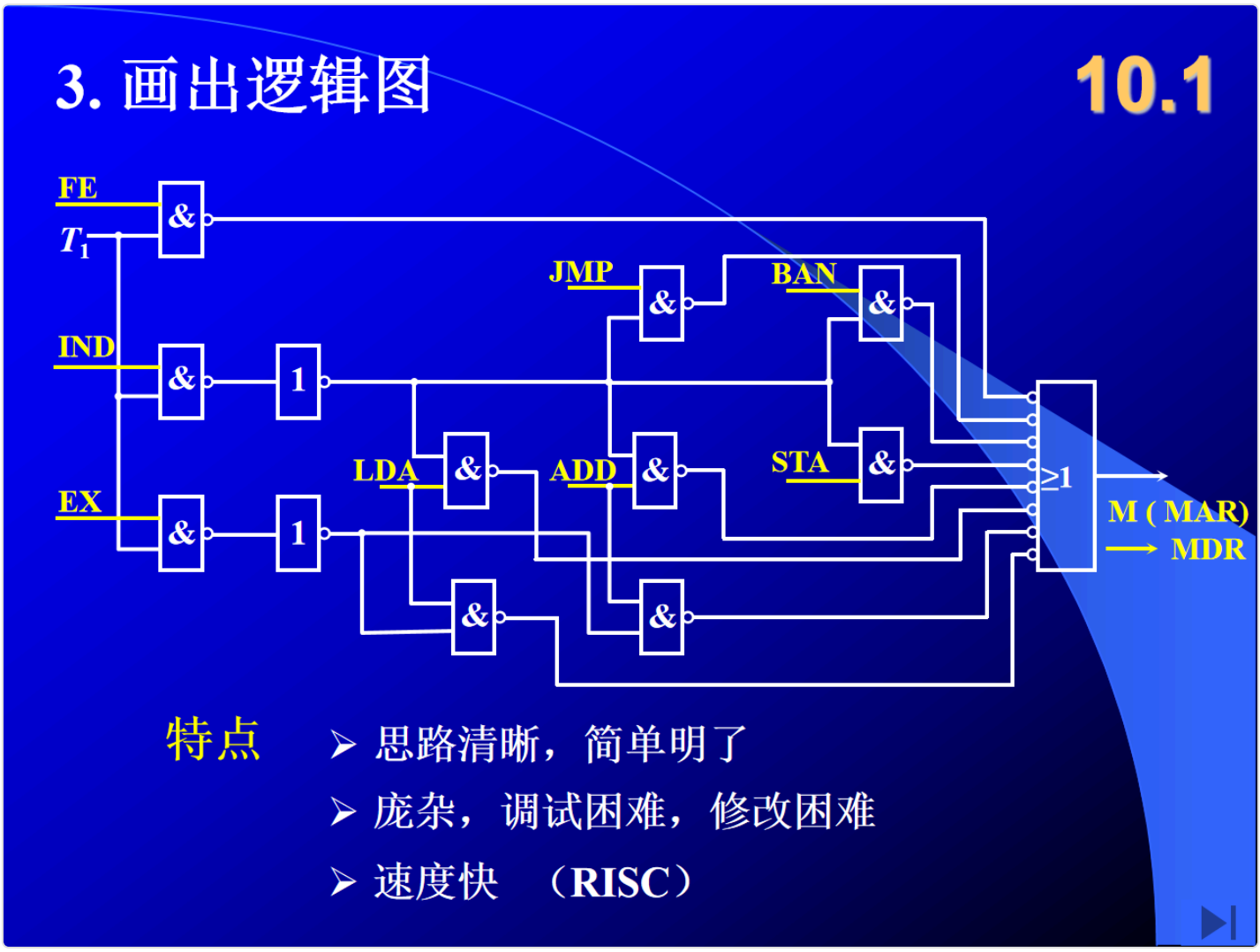
- 取指周期 T_1：所有指令都要 → $FE \cdot T_1$；
- 间址周期 T_1：ADD、STA、LDA、JMP、BAN → $IND \cdot T_1 \cdot (ADD+STA+LDA+JMP+BAN)$；
- 执行周期 T_1：ADD、LDA → $EX \cdot T_1 \cdot (ADD+LDA)$。

$$
\begin{aligned}
M(MAR)\rightarrow MDR &= FE \cdot T_1 + IND \cdot T_1 \cdot (ADD+STA+LDA+JMP+BAN) + EX \cdot T_1 \cdot (ADD+LDA) \\
&= T_1 \cdot \{\, FE + IND \cdot (ADD+STA+LDA+JMP+BAN) + EX \cdot (ADD+LDA) \,\}
\end{aligned}
$$

3.3 步骤三：画出逻辑图（课件 p.515）

用与门、或门、非门按最简表达式搭出组合逻辑网络（课件 p.515 即上式的与-或实现）。硬布线控制器特点（课件 p.515）：

- **思路清晰，简单明了**；
- **庞杂，调试困难，修改困难**（指令越多门电路越庞杂，改指令要改接线）；
- **速度快**（纯硬件产生控制信号，故用于 **RISC**）。



易错警示：写表达式时「周期标记」与「节拍」缺一不可：同一微操作在不同周期、不同节拍出现时要分项相或；漏掉 IND 项或 EX 项是最常见错误（课件 p.514）。

4. 微程序设计思想与基本概念

4.1 思想的产生（课件 p.516）

1951 年英国剑桥大学教授 **Wilkes** 提出：把完成一条机器指令所需的微操作命令编成**微指令**（二进制代码，如 10100000），若干微指令组成**微程序**，**一条机器指令对应一个微程序**，微程序**存入 ROM**。这种用存储逻辑实现控制的方式称为**存储逻辑**（与硬布线的「组合逻辑」相对）。

层次关系：微操作命令 →（组合成）微指令 →（序列成）微程序 →（解释）机器指令。

4.2 机器指令对应的微程序与控制存储器布局（课件 p.517、519、520）

控制存储器（CM）中存放：

- **取指周期微程序**（地址 M、M+1、M+2），为所有机器指令**公用**；
- 每条机器指令的**执行周期微程序**（如 LDA：P、P+1、P+2；STA：K、K+1、K+2；ADD：Q、Q+1、Q+2）；
- **间址周期微程序**、**中断周期微程序**，同样为各机器指令公用（课件 p.517）。

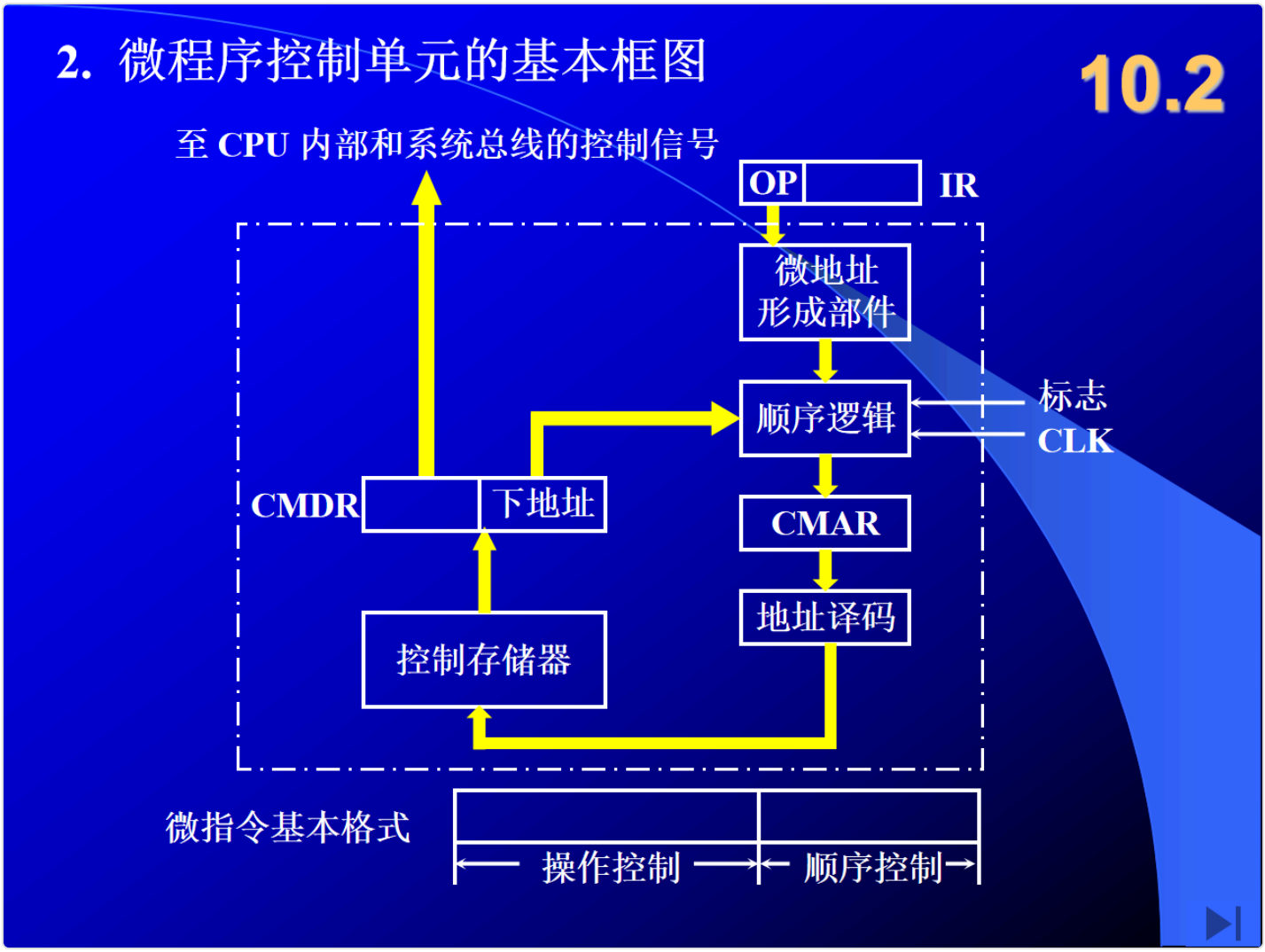
执行微程序的最后一条微指令把下地址指回取指微程序入口 M，实现「转取指周期微程序」；取指微程序末尾则按操作码「转执行周期微程序」（课件 p.519）。

要点：微程序个数 = 机器指令条数 + 公用的周期微程序数（取指、间址、中断），故**微程序数 ≠ 机器指令数**（由课件 p.517 布局可知）。

易错警示：微程序存在**控制存储器（ROM）**中，不是主存；用户程序才在主存（课件 p.516、520、523）。

5. 微程序控制单元组成与工作原理

5.1 基本框图（课件 p.518）



部件	功能
控制存储器 CM	存放全部微程序（ROM），按微地址读出微指令
CMAR（控存地址寄存器，即微地址寄存器）	存放欲读微指令的微地址（中文名为补充常识，课件用缩写 CMAR）
CMDR（控存数据寄存器，即微指令寄存器）	存放从 CM 读出的微指令（同上）
微地址形成部件	由机器指令操作码 OP(IR) 形成该指令执行微程序的入口微地址
顺序逻辑	在标志、CLK 及下地址/转移方式控制下产生后续微地址，送 CMAR
地址译码	对 CMAR 译码选中 CM 单元

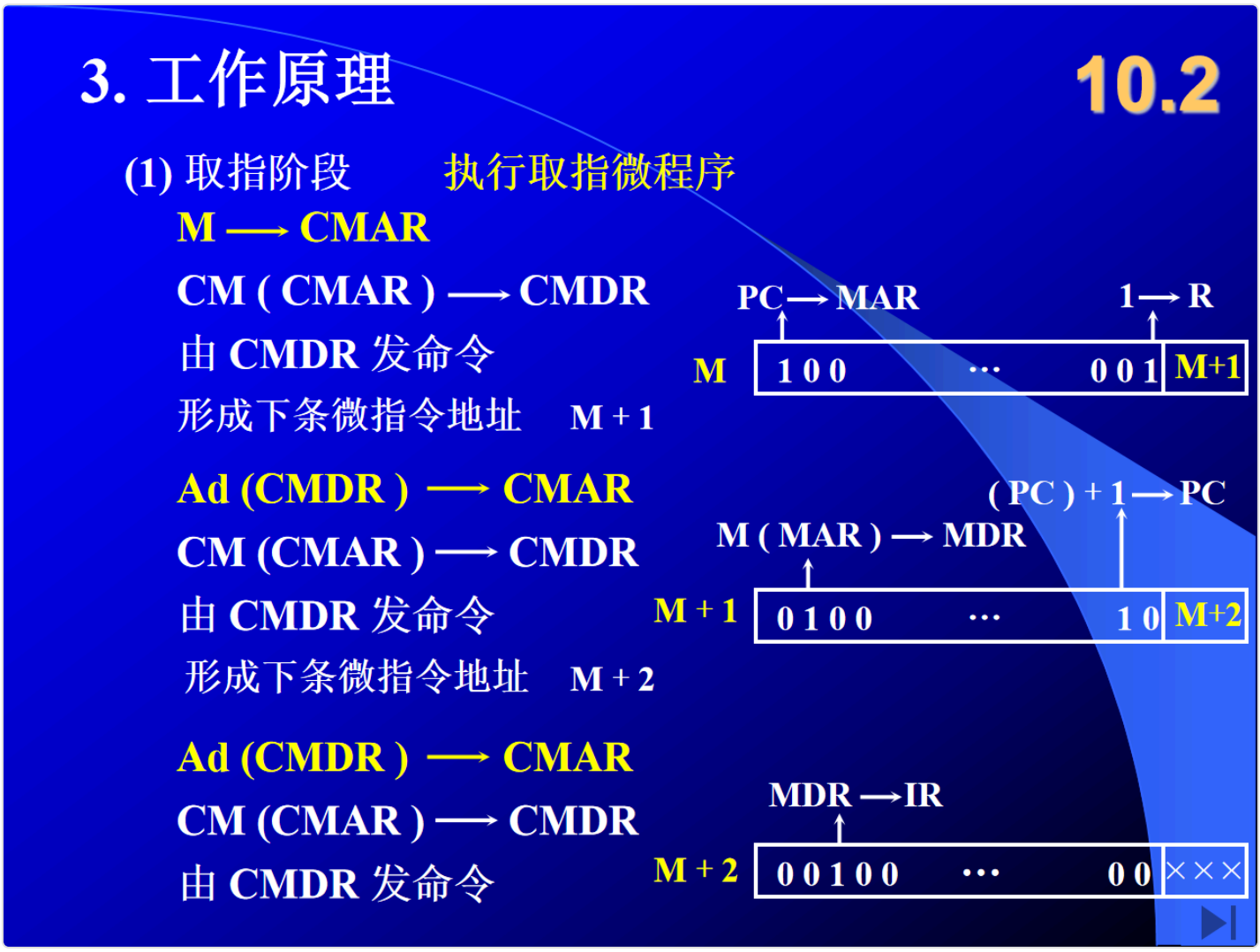
微指令基本格式 = 操作控制字段 + 顺序控制字段（顺序控制字段含下地址/转移方式）（课件 p.518）。操作控制字段发出「至 CPU 内部和系统总线的控制信号」。

5.2 工作原理（课件 p.520–523）

以用户程序 LDA X、ADD Y、STA Z、STP 为例（课件 p.520）：

(1) 取指阶段（执行取指微程序，课件 p.521）：

```
M→CMAR
CM(CMAR)→CMDR          ；读出微指令 M
由 CMDR 发命令          ；PC→MAR，1→R
形成下条微指令地址 M+1：Ad(CMDR)→CMAR
CM(CMAR)→CMDR          ；读出微指令 M+1
由 CMDR 发命令          ；M(MAR)→MDR，(PC)+1→PC
Ad(CMDR)→CMAR          ；下地址 M+2
CM(CMAR)→CMDR          ；读出微指令 M+2
由 CMDR 发命令          ；MDR→IR
```



(2) 执行阶段（课件 p.522）：OP(IR)→微地址形成部件→CMAR（即 P→CMAR），转入 LDA 执行微程序；依次执行 P、P+1、P+2 三条微指令（Ad(IR)→MAR、1→R；M(MAR)→MDR；MDR→AC），每条之后 Ad(CMDR)→CMAR 形成下地址；最后 M→CMAR 返回取指微程序。

(3) 再取指、再执行……循环直至 STP（课件 p.523）。

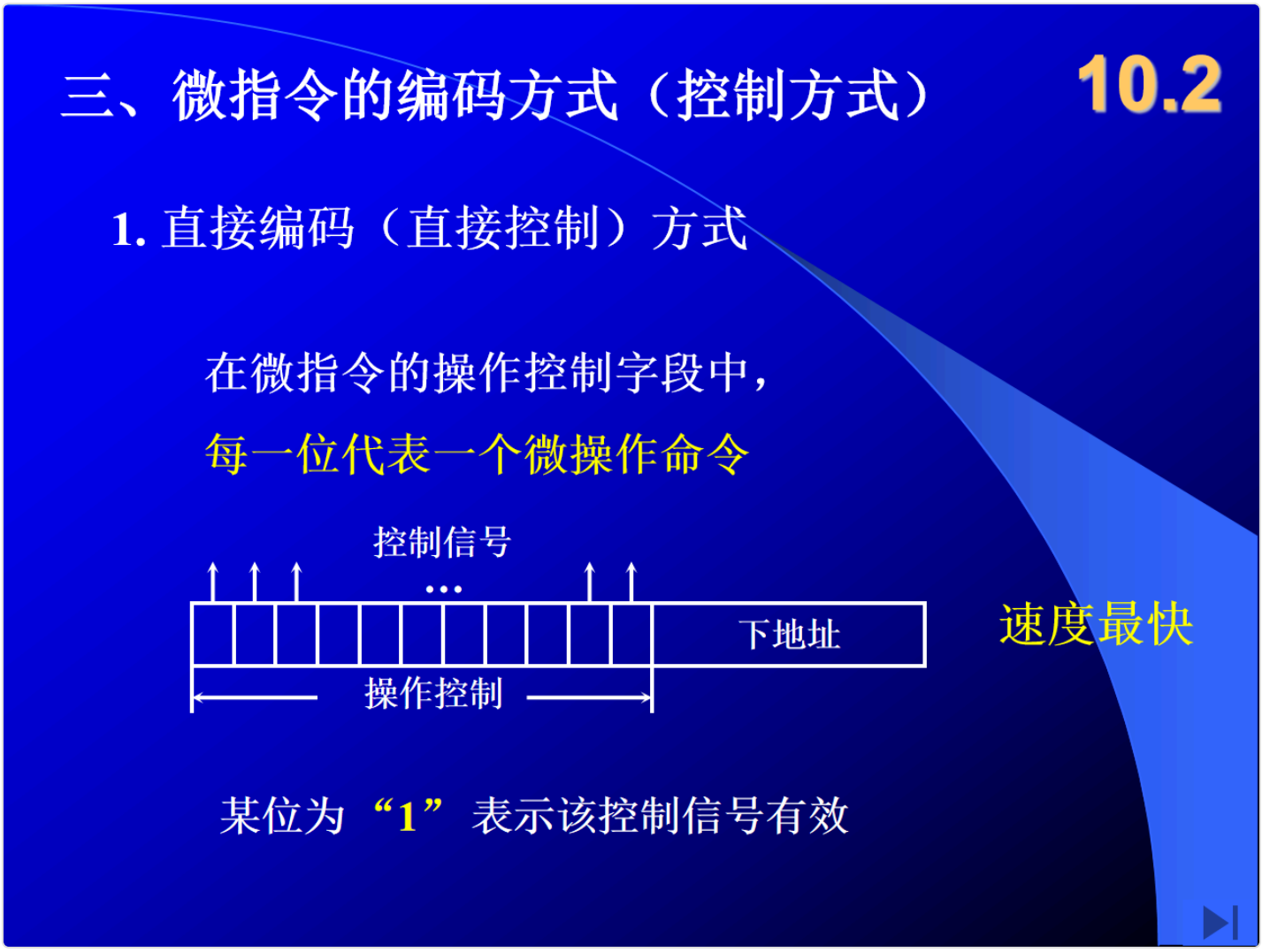
关键结论（课件 p.523）：全部微指令存在 CM 中，程序执行过程中**只需读出**（不改写）。由此引出微程序设计的两个关键问题：① 微指令的**操作控制字段如何形成微操作命令**（→编码方式）；② 微指令的**后续地址如何形成**（→地址形成方式）。

易错警示：CMAR/CMDR 是控存侧的寄存器，MAR/MDR 是主存侧的寄存器，两套不要混用；「CM(CMAR)→CMDR」读的是微指令，「M(MAR)→MDR」读的是机器指令/数据（课件 p.518、521）。

6. 微指令的编码方式（控制方式）

6.1 直接编码（直接控制）方式（课件 p.524）

操作控制字段中**每一位代表一个微操作命令**，某位为「1」表示该控制信号有效，无需译码。**速度最快**，但微指令字长最长。



6.2 字段直接编码方式（课件 p.525）

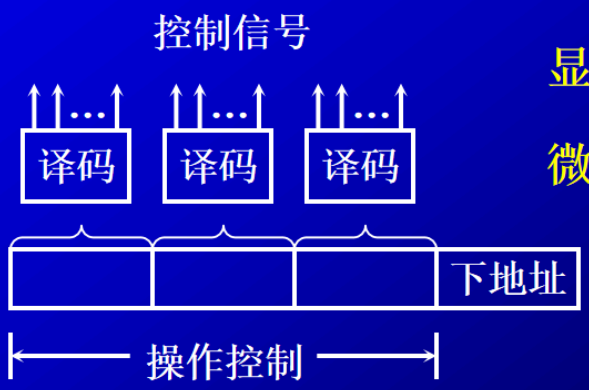
把控制字段分成若干「段」（字段），**每段经译码后发出控制信号**；**每个字段中的命令是互斥的**（不能同时有效的命令放同一段，可并行的命令放不同段）。又称**显式编码**。效果：**缩短了微指令字长，增加了译码时间**，微程序执行速度较慢。

2. 字段直接编码方式

10.2

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

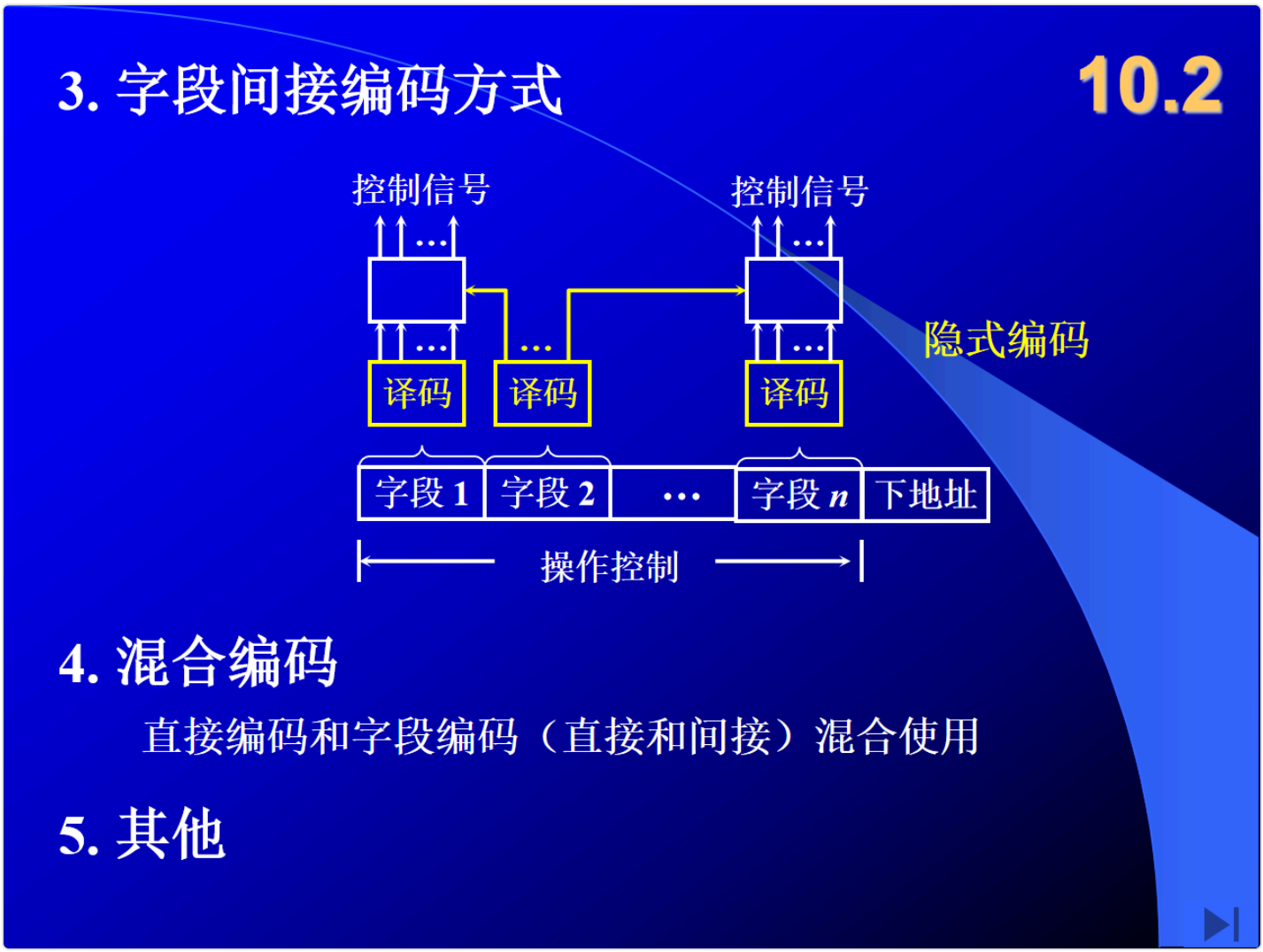
微程序执行速度较慢

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

6.3 字段间接编码方式（课件 p.526）

又称**隐式编码**：某一字段代码的含义还需由另一字段的代码来解释（课件 p.526 图中字段 2 的译码输出参与字段 1、字段 n 的译码），可进一步压缩字长（补充常识）。



6.4 混合编码与其他（课件 p.526）

混合编码 = 直接编码和字段编码（直接和间接）混合使用，兼顾字长与速度；课件另列「5. 其他」。

6.5 对比与计算口径

编码方式	是否译码	字长	速度	并行能力
直接编码	否	最长（=微操作数）	最快	强（各位独立）
字段直接	每段译码	较短	较慢	段间可并行
字段间接	多级译码	更短（补充常识）	更慢（补充常识）	受限
混合	部分译码	折中	折中	折中

字段直接编码位数计算口径：每段除 n_i 个互斥命令外通常还需留一个「不发命令」状态，故该段位数 = $\lceil \log_2(n_i+1) \rceil$（「留空状态」为补充常识，若题中未说明则按题设）。

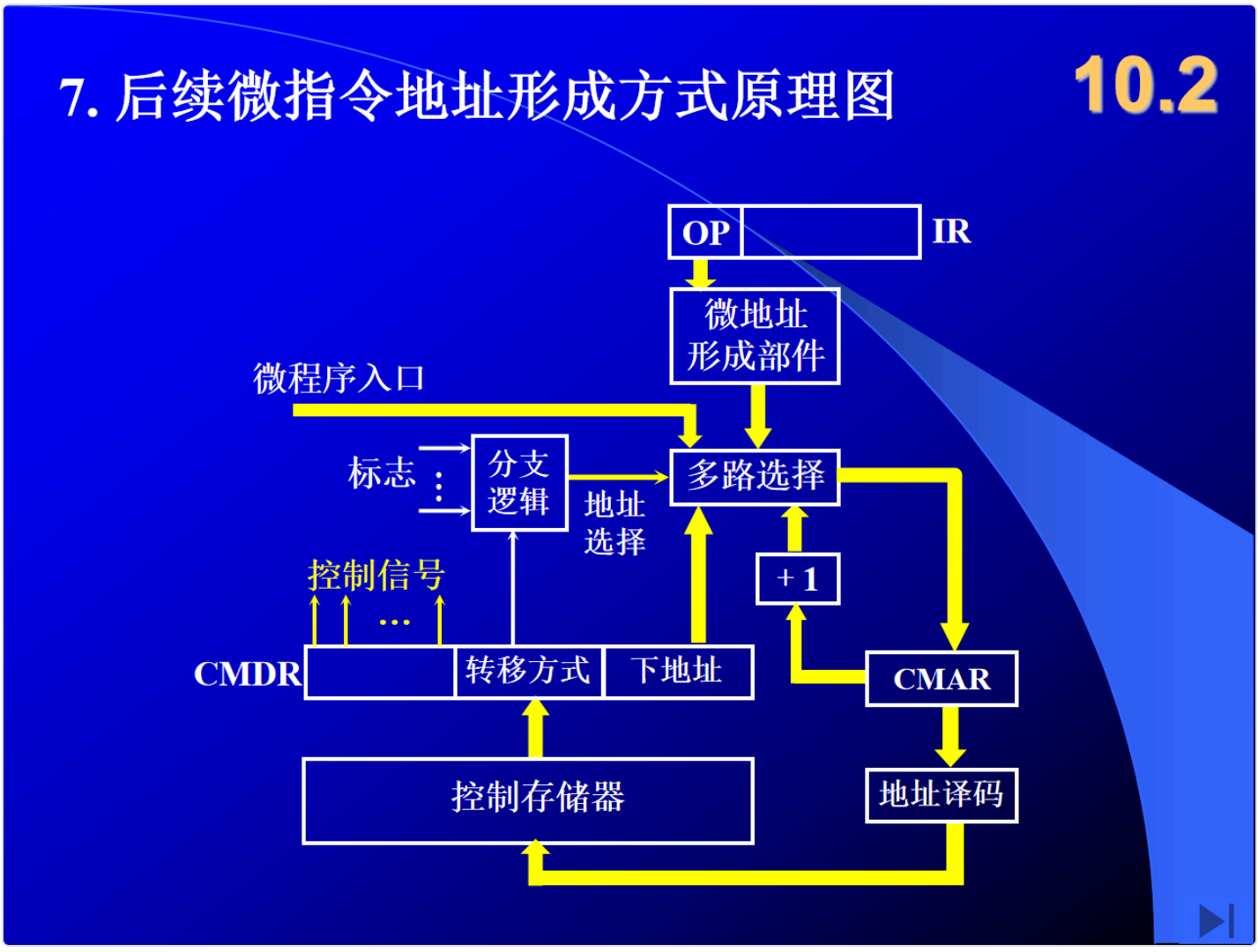
易错警示：「互斥」是字段**内**的要求，不是字段间；把可并行的微操作编进同一字段会丧失并行性（课件 p.525）。

7. 微指令序列地址的形成

课件 p.527–528 共给出 6 种方式：

1. **微指令的下地址字段指出**（断定方式）：顺序控制字段直接给出后继微地址；
2. **根据机器指令的操作码形成**：OP(IR)→微地址形成部件→微地址（用于取指结束后转入执行微程序）；
3. **增量计数器**：(CMAR)+1→CMAR（计数器方式）；
4. **分支转移**：微指令中设**转移方式**（指明判别条件）与**转移地址**（指明转移成功后的去向），由标志决定去向；
5. **通过测试网络**：微地址由非测试地址 h 与测试地址 l 经测试网络、结合测试源组合形成（课件 p.528）；
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；**中断周期微程序首地址**也由硬件产生（课件 p.528）。

（「断定方式」「计数器方式」这两个术语为补充常识，分别对应课件方式 1、3。）



原理图（课件 p.529）：**多路选择**器在「微程序入口（硬件）」「OP(IR) 经微地址形成部件」「分支逻辑输出（标志+转移方式）」「CMDR 下地址」「CMAR+1」之间选择，送 CMAR→地址译码→CM。

易错警示：下地址字段给出的是**微指令地址（控存地址）**，不是机器指令地址；取指微程序入口和中断微程序首地址**不能**靠下地址字段给出，必须硬件产生（课件 p.527–528）。

8. 微指令格式：水平型与垂直型

8.1 定义（课件 p.530）

- **水平型微指令**：一次能定义并执行**多个并行操作**的微指令，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的微指令。
- **垂直型微指令**：采用**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（一条微指令只完成一个/少量微操作，补充常识）。

8.2 两种格式的比较（课件 p.531）

1. 水平型比垂直型**并行操作能力强、灵活性强**；
2. 水平型执行一条机器指令所要的**微指令数目少，速度快**；
3. 水平型**用较短的微程序结构换取较长的微指令结构**（字长长、条数少）；
4. 水平型与机器指令**差别大**（垂直型与机器指令相似，补充常识）。

对比项	水平型	垂直型
并行能力	强	弱
微指令字长	长	短
微程序长度（条数）	短	长
执行速度	快	慢
与机器指令相似度	差别大	相似
编码	直接/字段/混合	微操作码

易错警示：「水平型微程序短」指**微指令条数少**，不是字长短；字长恰恰是水平型长（课件 p.531 第 3 条）。

9. 静态/动态微程序设计与毫微程序设计

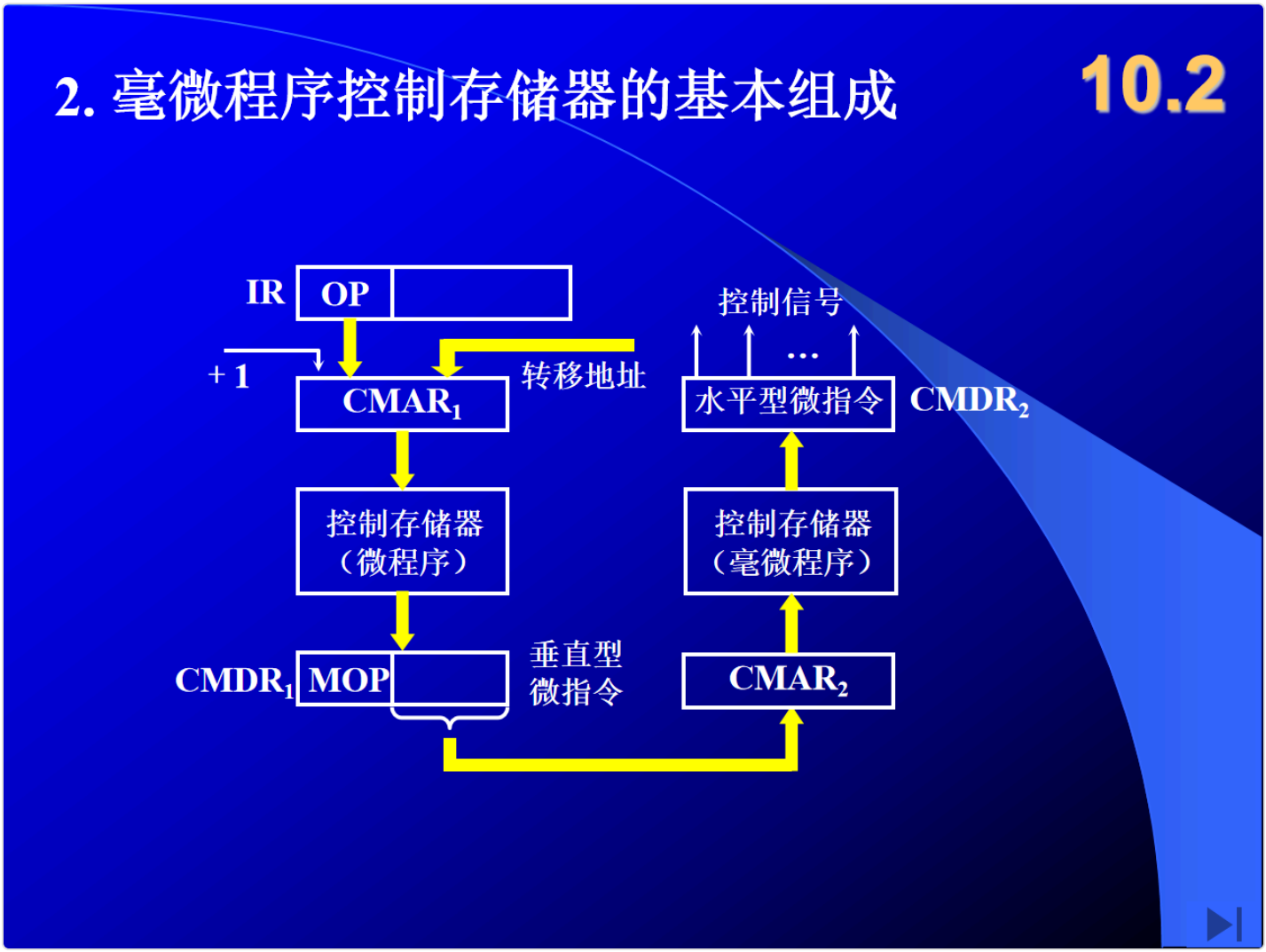
9.1 静态与动态（课件 p.532）

- **静态微程序设计**：微程序无须改变，采用 **ROM**；
- **动态微程序设计**：通过**改变微指令和微程序**来改变机器指令，有利于**仿真**，采用 **EPROM**。

9.2 毫微程序设计（课件 p.532–533）

- 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。

两级控制存储器组成（课件 p.533）：第一级 CM（微程序）由 $CMAR_1$ 寻址，读出**垂直型微指令**送 $CMDR_1$（含微操作码 MOP 字段）；MOP 作为第二级 CM（毫微程序）的微地址送 $CMAR_2$，读出**水平型微指令**送 $CMDR_2$，由其发出控制信号。即「垂直型微指令 → 水平型毫微指令」两级解释。

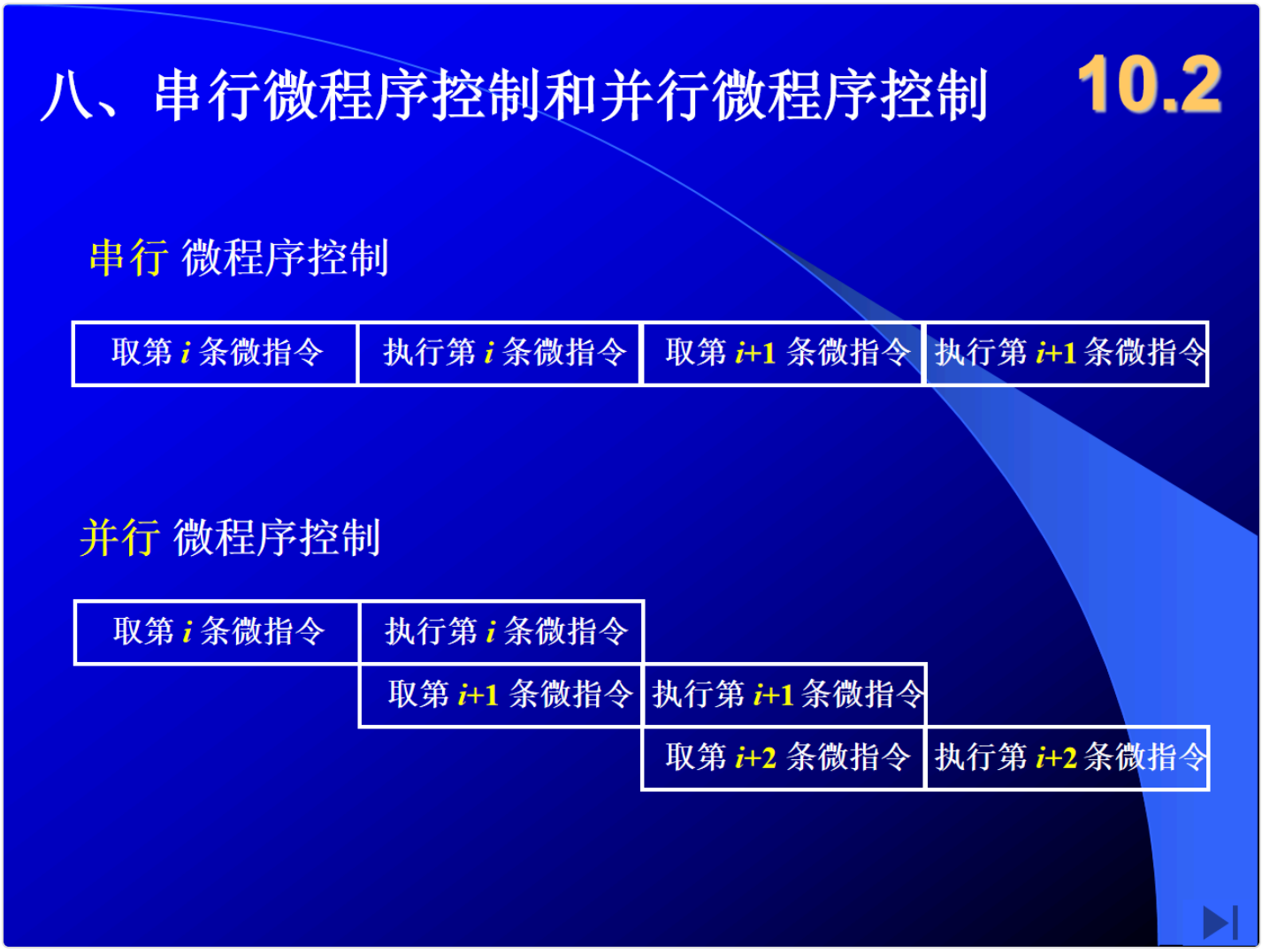


易错警示：动态微程序用 EPROM 而非 RAM（课件 p.532）；毫微程序解释的对象是**微程序**，不是机器指令（课件 p.532）。

10. 串行微程序控制与并行微程序控制

- **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 → 执行第 i+1 条……取与执行在时间上**串行**（课件 p.534）。

- **并行微程序控制**：**执行第 i 条微指令**与**取第 i+1 条微指令**在时间上**重叠**（课件 p.534 时序图），从而缩短微程序执行时间、提高速度。



要点：并行方式下「取微指令 i+1」与「执行微指令 i」同时进行，类似指令级流水的思想（补充常识）。

11. 微程序设计举例（课件 p.535–545）

前提：CPU 结构与组合逻辑举例相同（课件 p.535）。

11.1 写出微操作及节拍安排（考虑后续微地址形成）

取指阶段：若一个 T 内安排一条微指令，取指本需 3 条微指令；但还要安排形成后续微地址的微操作 Ad(CMDR)→CMAR 与 OP(IR)→微地址形成部件→CMAR，故展开为 6 个节拍（课件 p.535–536）：

节拍	微操作
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

执行阶段（课件 p.537–541）：取指微程序入口地址 M 由（执行微程序末条）微指令下地址字段指出。

- 非访存指令 CLA/COM/SHR/CSL/STP：T_0 执行微操作（0→AC / $\overline{AC}$→AC / 右移 / 循环左移 / 0→G），T_1 Ad(CMDR)→CMAR；
- 访存指令 ADD/STA/LDA：T_0 Ad(IR)→MAR、1→R（STA 为 1→W）；T_1 Ad(CMDR)→CMAR；T_2 M(MAR)→MDR（STA 为 AC→MDR）；T_3 Ad(CMDR)→CMAR；T_4 (AC)+(MDR)→AC（STA 为 MDR→M(MAR)，LDA 为 MDR→AC）；T_5 Ad(CMDR)→CMAR；
- 转移指令 JMP/BAN：T_0 Ad(IR)→PC / $A_0 \cdot Ad(IR) + \bar{A}_0 \cdot (PC) \rightarrow PC$；$T_1$ Ad(CMDR)→CMAR。

统计：全部微操作 20 个，微指令 38 条（课件 p.541）。38 条中含 19 条「纯送地址」微指令（18 条 Ad(CMDR)→CMAR + 1 条 OP(IR)→微地址形成部件→CMAR）。

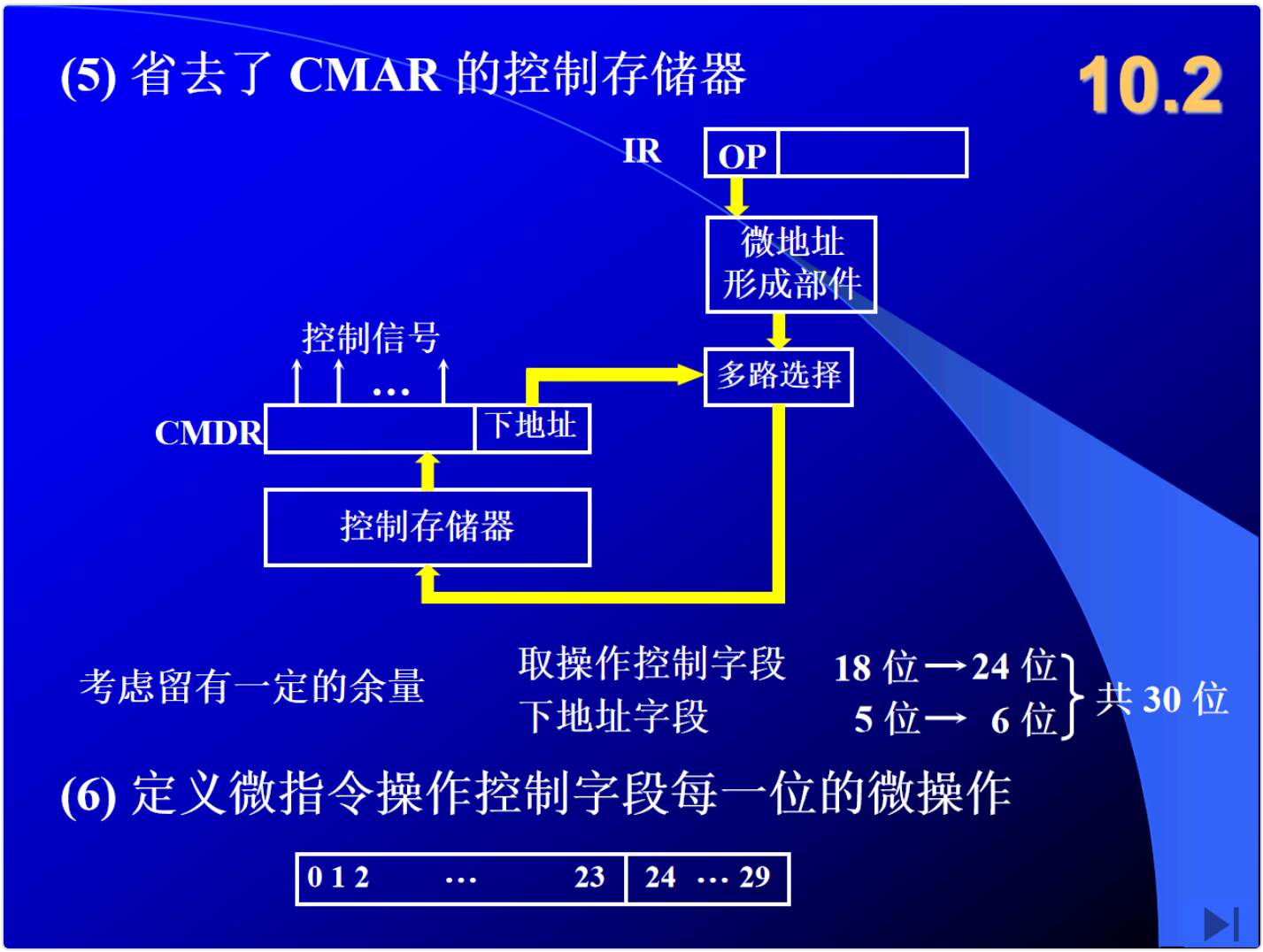
11.2 确定微指令格式与字长（课件 p.542）

- 编码方式：**直接控制**（直接编码）；
- 后续地址形成：① 机器指令操作码通过微地址形成部件形成；② 微指令下地址字段直接给出；
- 字长：20 个微操作 → 操作控制字段**最少 20 位**；38 条微指令 → 下地址字段 **6 位**（$2^5=32<38\le 2^6=64$）；微指令字长可取 **20+6=26 位**。

11.3 省去 CMAR 的优化（课件 p.543–544）

38 条微指令中有 **19 条**只完成「后续微地址→CMAR」（其中 18 条 Ad(CMDR)→CMAR、1 条 OP(IR)→微地址形成部件→CMAR）。若让 **Ad(CMDR) 直接送控存地址线**、OP(IR) 经微地址形成部件也直接送控存地址线（用多路选择器），则省去输至 CMAR 的时间、**省去 CMAR**，从而**省去 19 条微指令、2 个微操作**：

- 微指令数：38−19=**19** → 下地址字段最少 **5 位**（$2^4=16<19\le 2^5=32$）；
- 微操作数：20−2=**18** → 操作控制字段最少 **18 位**；
- 考虑留有一定余量：操作控制字段 18 位→**24 位**，下地址字段 5 位→**6 位**，**共 30 位**（课件 p.544）；位编号 0～23 为操作控制字段，24～29 为下地址字段。



11.4 编写微指令码点（课件 p.545）

微指令地址用**八进制**表示。课件码点表（节选可见列）：

微程序	微地址(八进制)	置 1 的操作控制位（可见列）	下地址字段(二进制)	后继微地址(八进制)
取指	00	0、1	000001	01
取指	01	2、3	000010	02
取指	02	4	××××××	由 OP(IR) 形成
CLA	03	（在省略列中）	000000	00（回取指）
COM	04	（在省略列中）	000000	00
ADD	10	1、10	001001	11
ADD	11	2	001010	12
ADD	12	—	000000	00
LDA	16	1、10	001111	17
LDA	17	2	010000	20
LDA	20	—	000000	00

由码点表可读出位定义：0=PC→MAR，1=1→R，2=M(MAR)→MDR，3=(PC)+1→PC，4=MDR→IR，10=Ad(IR)→MAR（课件 p.545；CLA/COM 等的操作位位于表中省略的列）。每个执行微程序末条微指令的下地址为 000000，即返回取指微程序入口 00；取指微程序末条（02）的下地址为×，因其后继地址由操作码经微地址形成部件形成。

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										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

易错警示：优化后「省去 CMAR」不等于「不要下地址字段」——下地址字段经多路选择器直接送控存地址线（课件 p.543–544）；字长计算必须用**优化后**的 19 条/18 个重新求位数，再留余量取 24+6=30（课件 p.544）。

12. 硬布线 vs 微程序控制器对比

对比项	硬布线（组合逻辑）	微程序
实现方式	组合逻辑网络（门电路）（p.515）	存储逻辑：CM+微程序（p.516）
控制信号产生	译码+节拍+标志直接组合产生（p.499）	读微指令、由操作控制字段发出（p.518、523）
速度	快（p.515）	较慢（需读控存/译码，补充常识）
规整性/可修改性	庞杂，调试、修改困难（p.515）	规整，改微程序即可改指令，动态微程序利于仿真（p.532）
典型应用	RISC（p.515）	CISC（补充常识）
核心部件	节拍发生器、操作码译码、逻辑网络（p.499）	CM、CMAR、CMDR、微地址形成部件（p.518）

要点：速度要求极高选硬布线；指令系统复杂、要求易修改易仿真选微程序（由 p.515、p.532 特点归纳）。

13. 易错点与高频陷阱汇总

1. 微指令≠机器指令：一条机器指令对应一个**微程序**（若干微指令），微指令存于 CM（ROM）而非主存（课件 p.516、523）。
2. 微程序数≠机器指令数：还有公用的取指、间址、中断周期微程序（课件 p.517）。
3. 字段直接编码要求**字段内互斥**；可并行的微操作必须分在不同字段（课件 p.525）。
4. 直接编码速度最快但字长最长；字段编码缩短字长却增加译码时间、速度变慢（课件 p.524–525）。
5. 水平型「微程序短」指条数少，其**字长反而长**；垂直型字长短但微程序长（课件 p.531）。
6. CMAR 存微地址、CMDR 存微指令；MAR/MDR 属主存侧，勿混（课件 p.518、521）。
7. 下地址字段给的是**微地址**；机器指令地址码 Ad(IR) 不是微地址（课件 p.527）。
8. 取指微程序首地址、中断周期微程序首地址由**硬件**产生，不能由下地址字段给出（课件 p.528）。
9. 中断周期由**中断隐指令**（硬件）完成，T_0 即硬件关中断；它不是指令系统中的指令（课件 p.507）。
10. 10.1 举例中非访存指令执行微操作在 T_2，10.2 微程序举例中在 T_0；按题设前提作答，勿混背（课件 p.504–506 vs p.537–538）。
11. 字长计算：下地址位数 k 满足 $2^k \geq$ 微指令条数；直接编码操作控制位数 ≥ 微操作个数；优化后要用 19/18 重算（课件 p.542–543）。
12. 机器周期含几个节拍是设计约定：p.500 图示 4 拍、本章举例 3 拍，勿死记「4 拍」（课件 p.500–501）。
13. BAN 的转移条件是符号位 A_0：A_0=1 转移、A_0=0 顺序，表达式 $A_0 \cdot Ad(IR) + \bar{A}_0 \cdot (PC) \rightarrow PC$（课件 p.506）。
14. 省去 CMAR 后下地址字段经**多路选择器直送控存地址线**，字段本身仍保留（课件 p.543–544）。
15. 动态微程序用 **EPROM**、静态用 **ROM**；毫微程序解释的是**微程序**（课件 p.532）。

14. 典型例题精讲

例1 微操作命令最简表达式（课件 p.514）

题：写出微操作 M(MAR)→MDR 的最简逻辑表达式。

解：① 查操作时间表找所有出现该微操作的格：$FE \cdot T_1$（全部指令）、$IND \cdot T_1$（ADD、STA、LDA、JMP、BAN）、$EX \cdot T_1$（ADD、LDA）。② 相或：$FE \cdot T_1 + IND \cdot T_1 \cdot (ADD+STA+LDA+JMP+BAN) + EX \cdot T_1 \cdot (ADD+LDA)$。③ 提公因子 T_1：$T_1 \cdot \{FE + IND \cdot (ADD+STA+LDA+JMP+BAN) + EX \cdot (ADD+LDA)\}$。④ 用与-或门实现即逻辑图（课件 p.515）。教学提示：IND 项的指令集合是「带地址且可间址」的指令，EX 项只有读内存的 ADD、LDA，STA 是写内存故不在其中。

例2 微指令字长计算与优化（课件 p.541–544）

题：举例机共 20 个微操作、38 条微指令，采用直接编码。求微指令字长；若省去 CMAR 优化后再求最小字长与留余量后的字长。

解：① 操作控制字段 ≥ 20 位；下地址字段 k：$2^5=32<38\le64=2^6$ → 6 位；字长 = 20+6 = **26 位**。② 优化：38 条中 19 条为纯送地址微指令（18 条 Ad(CMDR)→CMAR + 1 条 OP(IR)→微地址形成部件→CMAR），直送控存地址线后省去这 19 条及 2 个微操作 → 19 条微指令、18 个微操作。③ 下地址字段：$2^4=16<19\le32=2^5$ → 5 位；操作控制字段 ≥ 18 位；最小 18+5=23 位。④ 留余量取 24+6 = **30 位**（位 0–23 操作控制、24–29 下地址）。

例3 取指微程序的节拍与码点（课件 p.535–536、545）

题：说明取指微程序为何由 3 条微指令展开为 6 个节拍；写出微指令 00 的码点含义及 02 的下地址为何为×。

解：① 取指本身 3 拍（PC→MAR、1→R；M(MAR)→MDR、(PC)+1→PC；MDR→IR、OP(IR)→微地址形成部件），一拍一条微指令即 3 条；但每条微指令执行后还需形成下地址（Ad(CMDR)→CMAR 或 OP(IR)→微地址形成部件→CMAR），各占一拍，故 6 拍、优化前 6 条微指令。② 微指令 00：位 0=1（PC→MAR）、位 1=1（1→R），下地址 000001→微指令 01。③ 微指令 02 执行 MDR→IR 后，后继微地址应由 OP(IR) 经微地址形成部件产生（转入执行微程序），故下地址字段无意义，记×。

附：本章速查表

周期微操作速查

周期	T_0	T_1	T_2
取指	PC→MAR，1→R	M(MAR)→MDR，(PC)+1→PC	MDR→IR，OP(IR)→ID
间址	Ad(IR)→MAR，1→R	M(MAR)→MDR	MDR→Ad(IR)
中断	0→MAR，1→W，硬件关中断	PC→MDR	MDR→M(MAR)，向量地址→PC

微程序举例关键数据：微操作 20 个、微指令 38 条（p.541）；初取字长 20+6=26 位（p.542）；省 CMAR 后 19 条/18 个 → 最少 18+5 位（p.543）；留余量 24+6=30 位，位 0–23 操作控制、24–29 下地址（p.544）；码点八进制地址：取指 00–02、CLA 03、COM 04、ADD 10–12、LDA 16–20（p.545）。

字长公式：直接编码操作控制位数 ≥ 微操作个数；下地址位数 k 满足 $2^k \ge$ 微指令条数；字段直接编码操作控制位数 = $\Sigma\lceil\log_2(n_i+1)\rceil$（$n_i$ 为第 i 段互斥命令数，+1 为留空状态，补充常识）。

编码/格式/地址形成速记：直接编码最快字长最长；字段直接=显式、段内互斥；字段间接=隐式；混合=折中。水平型=并行强、字长长、条数少、速度快；垂直型=似机器指令、字长短、条数多。地址形成 6 法：下地址字段、操作码、(CMAR)+1、分支转移、测试网络、硬件入口（首条/中断首地址）。

器件与方式速记：静态微程序 ROM；动态微程序 EPROM、利于仿真；毫微程序解释微程序，两级控存=垂直型微指令→水平型毫微指令；串行微程序取执串行、并行微程序取 i+1 与执 i 重叠；硬布线快而难改（RISC），微程序慢而易改（CISC，补充常识）。