



《计组I》

第五章 处理器设计3

计算机科学与技术学院



温故 —关于指令系统和CPU(1/2)

- 组合逻辑控制器的实现原理?
- 枚举所有微操作所需的发生的条件，将其描述为与或门逻辑。
 - Case 1: 条件1*条件2*...*条件m
 - Case 2: 条件1*条件2*...*条件m
 - Case n: 条件1*条件2*...*条件m
- Case 1 + Case 2 + ...+ Case n

$$C_i = \sum (M_m \cdot T_n \cdot I_j \cdot F_k)$$

第m个CPU周期

第n个节拍

指令译码器的第j个输出

第k个CPU内部状态标志或CPU外部请求信号

在执行指令 I_j 时，若状态 F_k 满足要求，则在第m个机器周期 M_m 的第n个节拍 T_n ，控制单元发出 C_i 控制命令



温故 —关于指令系统和CPU(1/2)

- 组合逻辑控制器的优缺点？
 - 速度快
 - 修改不灵活、复杂的处理器的与或组合电路实现困难、扩展性差
- 简述微程序控制器的原理？
 - 将程序控制的思想引入控制信号的形成和控制
- 什么是微操作、微命令、微指令、微程序，以及它们之间的关系？
 - 微操作：CPU中的一个原子的、基本的最小操作
 - 微命令：触发/发起以上最小操作的命令/信号，由控制器发出（就是个电信号）
 - 微指令：微程序控制器内，对微命令的指令化对应；
 - 微程序：一组微指令，就是一微程序

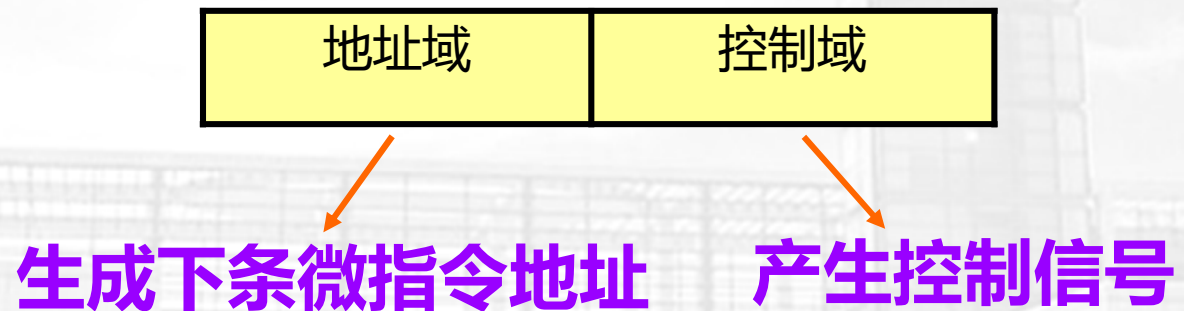


微程序控制器



(基于) 微程序 (的) 控制器

- 一条 (机器) 指令对应一个**微程序**, 该微程序包含从取指令到执行指令一个**完整微操作序列**对应的全部**微指令**, 它被存入一个称为**控制存储器** (control memory) 的ROM中。
- **CM中存放着指令系统中定义的所有指令的微程序。**
- **微指令周期**: 一条微指令执行的时间 (包括从控制存储器中**取得微指令**和**执行微指令**所用时间) 。
- **微指令的一般格式**:





(基于) 微程序 (的) 控制器

■ 微指令的一般格式:

- **地址域**: 决定如何取得微指令
- **控制域**: 微指令的执行

■ 设计微指令需要从两方面考虑:

- 微指令的**长度** → 减少控制器占CPU集成芯片的面积
- 微指令的**执行时间** → 提高CPU的工作速度



(基于) 微程序 (的) 控制器

- 核心结构:

- 控制存储器 (CM)

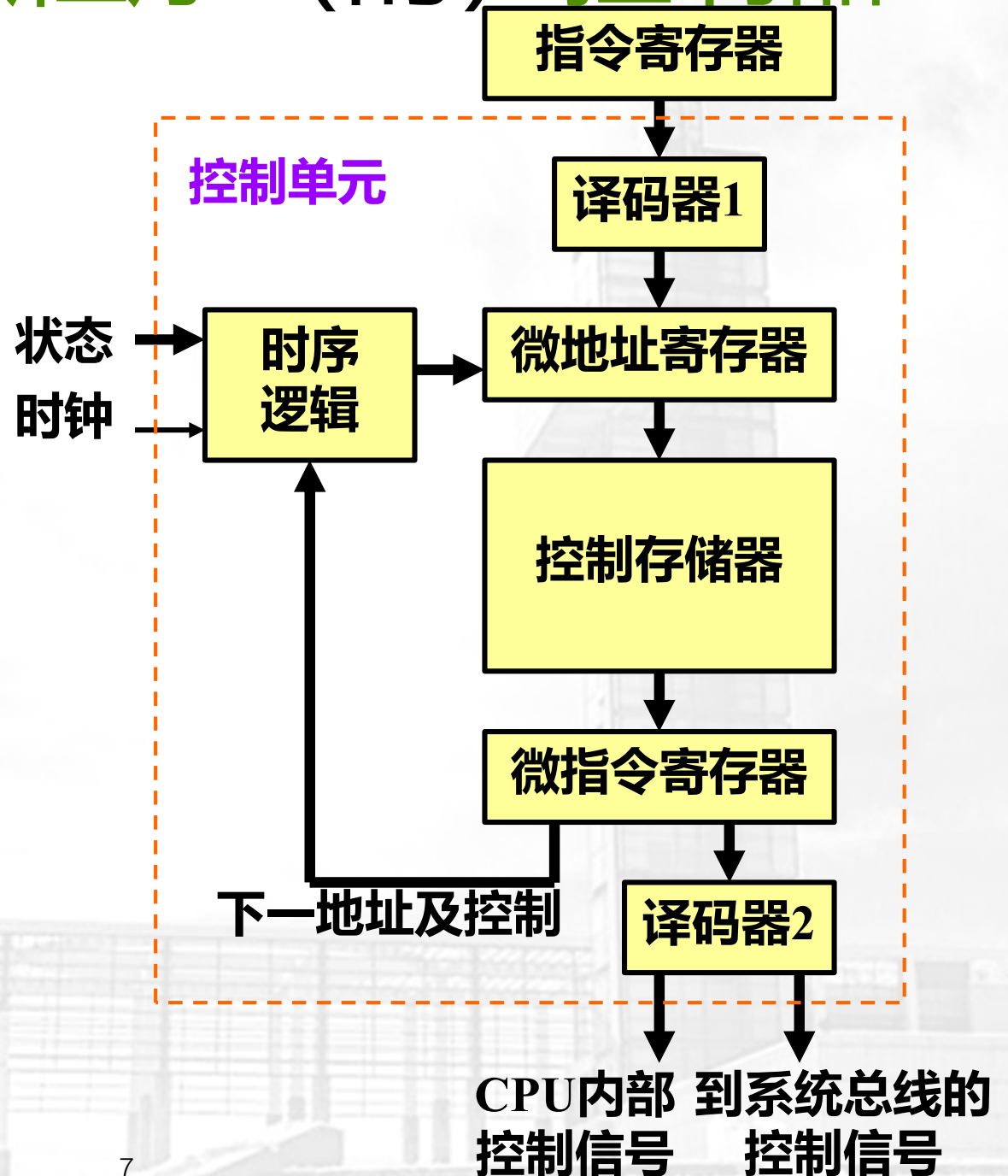
- 微指令长度
- 微程序占用的存储单元数

- 微指令寄存器 μIR 、微地址寄存器 μAR

- 微地址形成电路

- 时序逻辑

- 依据时钟按节拍为控制存储器提供读出控制信号。
- 在微程序运行时依据CPU内外状态 (ALU标志、中断请求、DMA请求等) 和当前微指令地址域的信息生成下一条微指令地址, 并将其装入到微地址寄存器中。

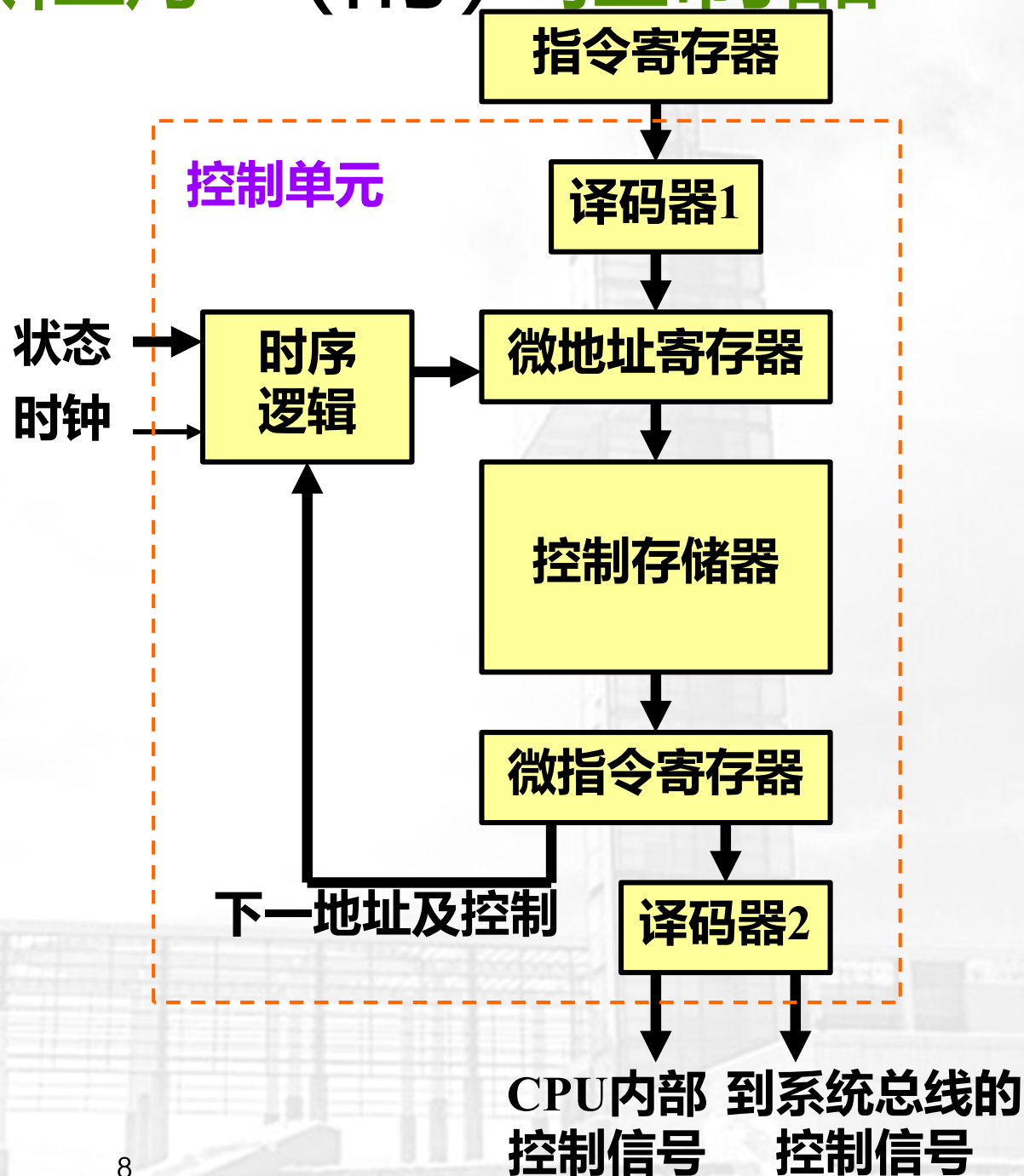




(基于) 微程序 (的) 控制器

微程序控制器在一个时钟周期内完成如下工作:

- ① 时序逻辑电路给控制存储器发出read命令;
- ② 从微地址寄存器 μAR 指定的控存单元读出微指令, 送入微指令寄存器 μIR ;
- ③ 根据微指令寄存器的内容, 产生控制信号, 给时序逻辑提供下条微地址信息;
- ④ 时序逻辑根据来自微指令寄存器的下条微地址信息和CPU内外状态, 给微地址寄存器加载一个新的微地址。





微程序控制器—地址形成



(基于) 微程序 (的) 控制器：地址形成

■ 下一条微指令的地址有三种可能：

① 由指令寄存器确定的微程序首地址：

每一个指令周期仅出现一次，且仅出现在刚刚获取一条指令之后。

② 下一条顺序地址

下一条微指令地址 = 当前微指令地址 + 1

③ 分支跳转地址

◆ 无条件 和 条件 跳转

◆ 两分支 和 多分支 跳转

● 两地址格式（断定方式）

● 单地址格式（计数方式，增量方式）

● 可变格式



The diagram illustrates the architecture of a microprogrammed control system. It features several key components and their interconnections:

- IR (Instruction Register):** The top component, which provides the initial instruction.
- K (Microprogram Counter):** Receives input from the IR and outputs the **微程序首地址** (Microprogram Start Address).
- 多路选择器 (Multiplexer):** A central component that selects between different addresses based on control signals. It receives inputs from the **分支逻辑** (Branch Logic) and the **微程序首地址**. It outputs the **顺序地址** (Sequence Address) to the **μAR**.
- 分支逻辑 (Branch Logic):** Receives **状态标志** (Status Flags) and provides **地址选择** (Address Selection) to the multiplexer.
- μAR (Microprogram Address Register):** Receives the **顺序地址** and outputs to the **CM**.
- CM (Control Memory):** The memory unit that stores microprograms. It outputs to the **μIR**.
- μIR (Microprogram Instruction Register):** Receives data from the CM and outputs to the **AC** and the **分支逻辑**.
- AC (Accumulator):** Receives data from the μIR and outputs to the **分支逻辑**.
- 控制域 (Control Field):** Part of the microprogram output, which can be used for **条件选择** (Conditional Selection) to influence the branch logic.

The flow of control is as follows: The IR provides the start address to K, which then feeds into the multiplexer. The multiplexer also receives branch logic input and outputs the sequence address to the μAR. The μAR feeds into the CM, which outputs to the μIR. The μIR feeds into the AC and the branch logic. The AC and status flags feed into the branch logic. The branch logic feeds back into the multiplexer. The control field of the microprogram output can be used for conditional selection to influence the branch logic.

两地址格式的分支控制逻辑

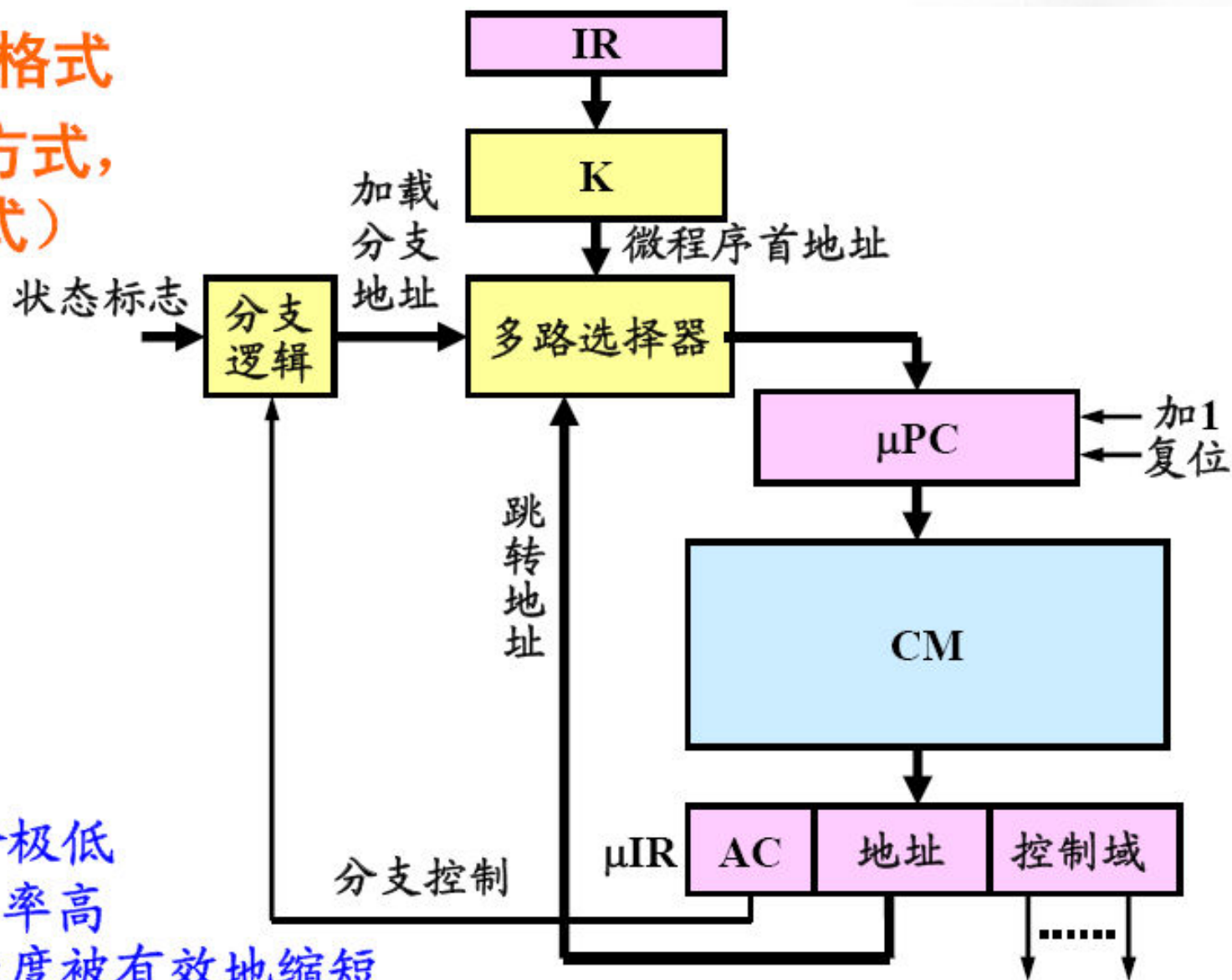


(基于) 微程序 (的) 控制器：地址形成

2. 单地址格式

(计数方式,
增量方式)

- 硬件代价极低
- μPC 利用率高
- 微指令长度被有效地缩短



单地址格式的分支控制逻辑



(基于) 微程序 (的) 控制器：地址形成

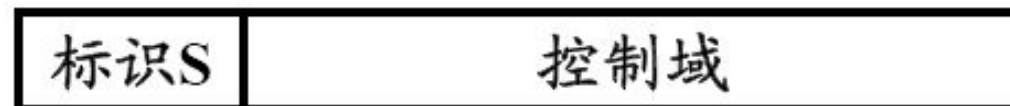
3. 可变格式

- 使任何微指令执行时不存在无用信息：让微指令在顺序执行时只提供控制信号的产生，需要分支时再提供跳转地址。→ 可变格式微指令

- 两种微指令格式

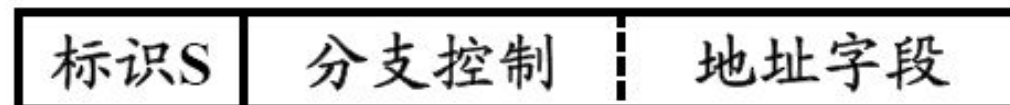
- 控制微指令

$S = 0$



- 转移微指令

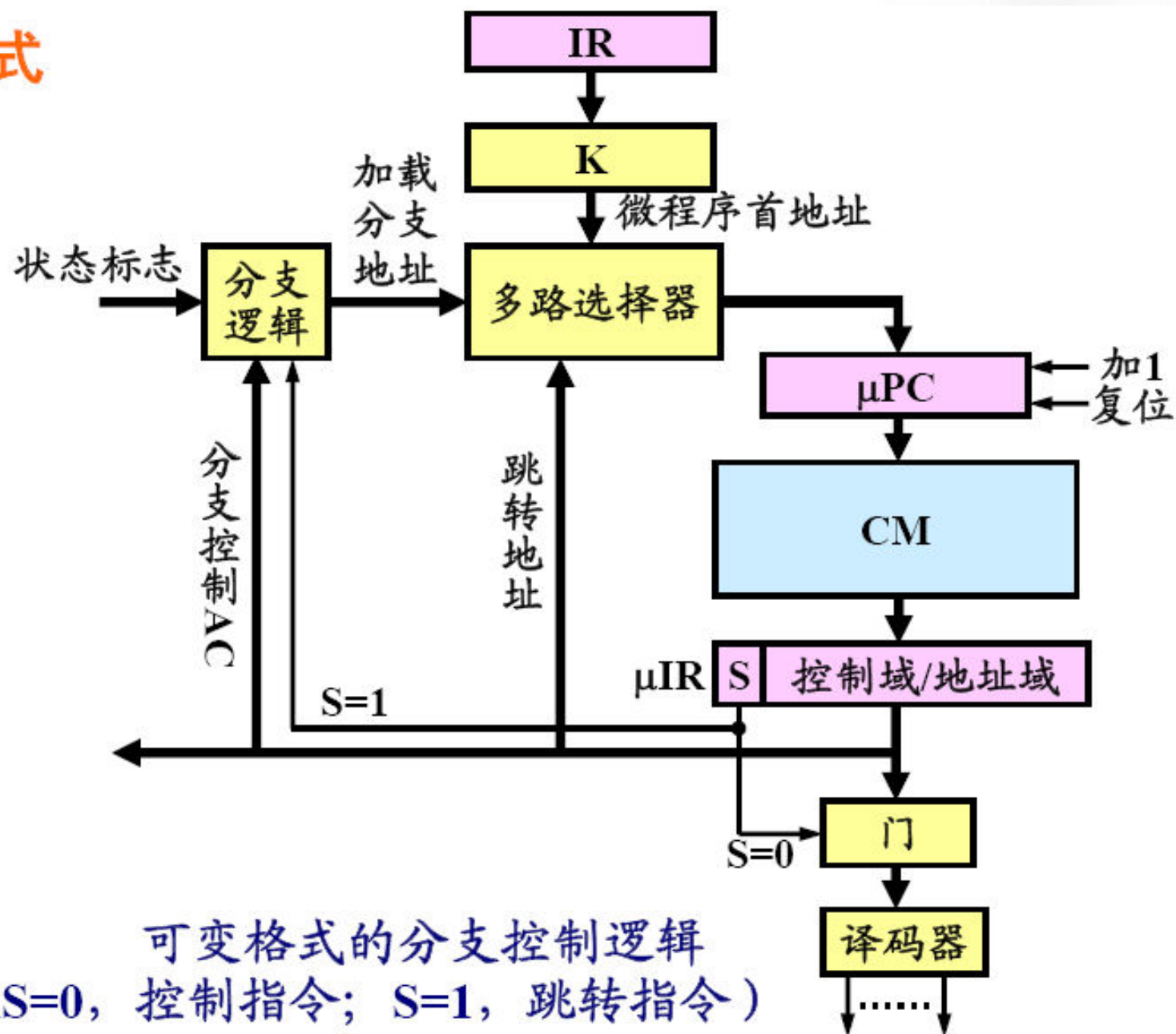
$S = 1$





(基于) 微程序 (的) 控制器：地址形成

3. 可变格式





微程序控制器——控制信号编码



(基于) 微程序 (的) 控制器：控制信号编码

■ 水平型微指令 (horizontal microinstruction)

多个控制信号同时有效 → 多个微操作同时发生。

■ 垂直型微指令 (vertical microinstruction)

类似于机器指令，利用微操作码的不同编码来表示不同的微操作功能。



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(1) 直接表示法 (水平编码)

- 可以在同一个时间有效的控制信号称为**相容信号**，具有**相容性**；
- 不能在同一个时间有效的控制信号称为**互斥信号**，具有**互斥性**。

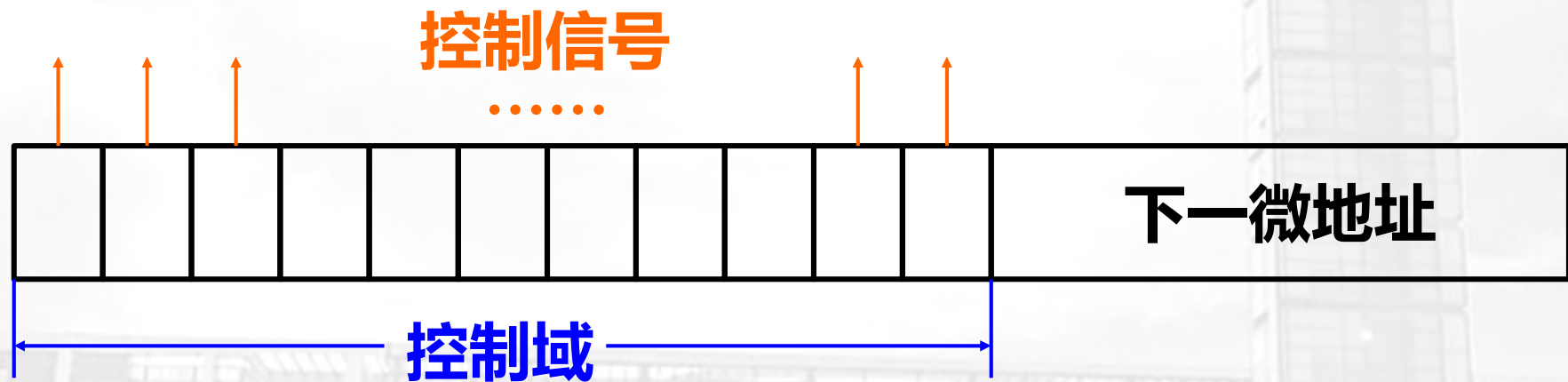


图6.15 直接表示法



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码法 (字段编码)

将控制域分为若干字段，**字段内垂直编码**，**字段间水平编码**。

👉 **互斥**的信号放在**同一字段**

👉 **相容**的信号放在**不同字段**

- 若各字段的编码相互独立，则通过各字段独立译码就可以获得计算机系统的全部控制信号，这被称作**直接译码**方式。
- 若某些字段的编码相互关联，则关联字段要通过两级译码才能获得相关的控制信号，这被称作**间接译码**方式。



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码 (字段编码)

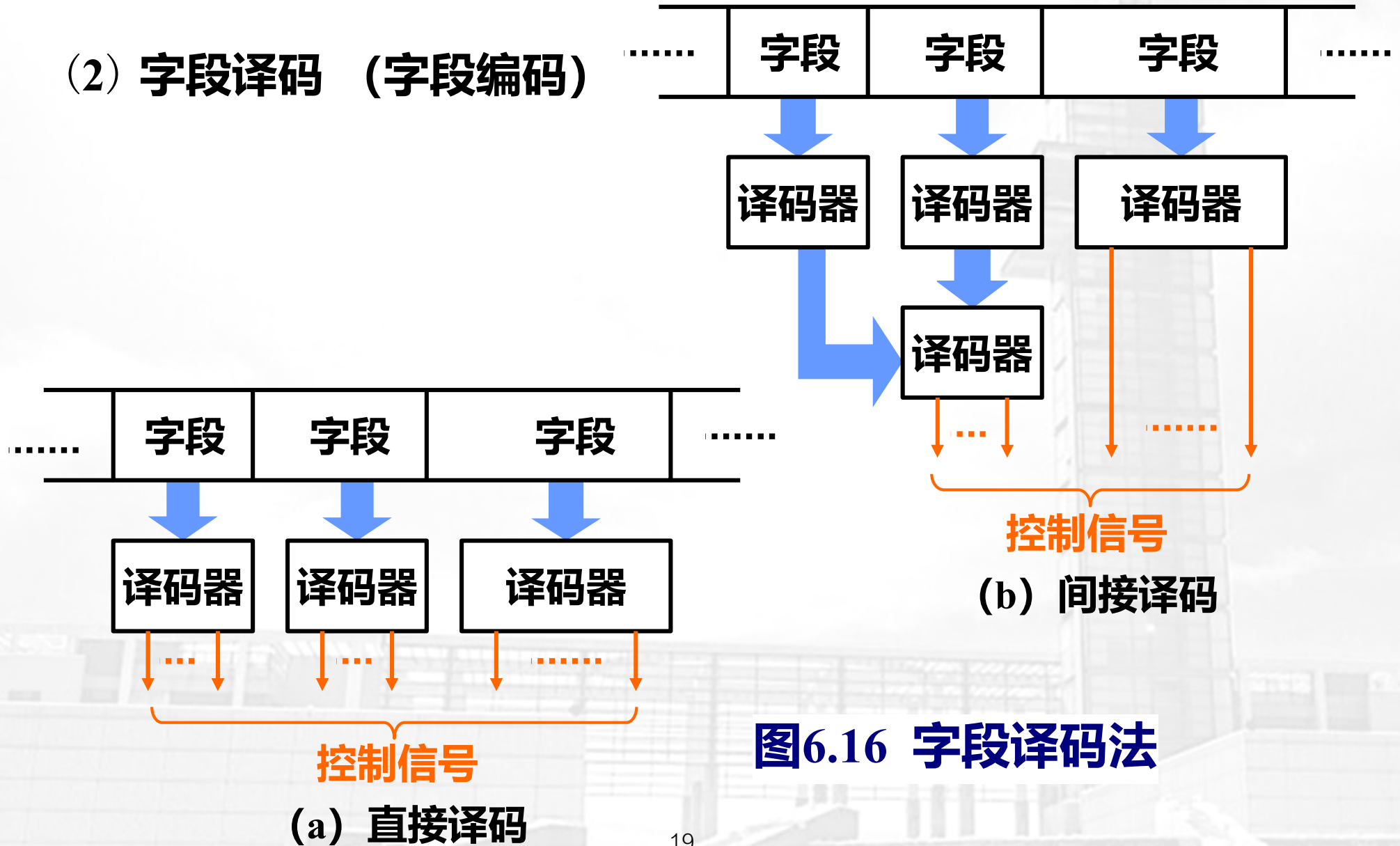


图6.16 字段译码法



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码法 (字段编码)

- 每个字段中要设计一个**无效**控制信号的编码
- 若控制域的某字段有 **m 位**，则可以提供 **2^m-1** 个控制信号的编码
- 字段组织的有效方法：
 - 按**功能**组织：把功能类同的各控制信号放在同一字段中。
 - 按**资源**组织：把加载到同一部件上的各控制信号放在同一字段中。



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码法 (字段编码)

按功能	按功能	按资源	按资源	按功能	按资源	按资源	
字段1 (4位)	字段2 (4位)	字段3 (2位)	字段4 (3位)	字段5 (4位)	字段6 (2位)	字段7 (2位)	字段8
NOP 0000	NOP 0000	NOP 00	NOP 000	NOP 0000	NOP 00	NOP 00	其他信号
R0 _{in} 0001	R0 _{out} 0001	PC _{in} 01	SP _{in} 001	ADD 0001	Mread 01	IOread 01	
R1 _{in} 0010	R1 _{out} 0010	PC _{out} 10	SP _{out} 010	SUB 0010	Mwrite 10	IOwrite 10	
...	PC+1 11	SP+1 011	AND 0011			
R7 _{in} 1000	R7 _{out} 1000		SP-1 100	OR 0100			
IR _{in} 1001	IR _{out} 1001			SHL 0101			
Y _{in} 1010	Z _{out} 1010			SHR 0110			
AR _{in} 1011	AR _{out} 1011			ROL 0111			
DRI _{in} 1100	DRI _{out} 1100			ROR 1000			
DRS _{in} 1101	DRS _{out} 1101						

*NOP为无效控制信号



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码法 (字段编码)

按功能	按功能	按资源	按资源	按功能	按资源	按资源	
字段1 (4位)	字段2 (4位)	字段3 (2位)	字段4 (3位)	字段5 (4位)	字段6 (2位)	字段7 (2位)	字段8
NOP 0000	NOP 0000	NOP 00	NOP 000	NOP 0000	NOP 00	NOP 00	其他信号
R0 _{in} 0001	R0 _{out} 0001	PC _{in} 01	SP _{in} 001	ADD 0001	Mread 01	IOread 01	
R1 _{in} 0010	R1 _{out} 0010	PC _{out} 10	SP _{out} 010	SUB 0010	Mwrite 10	IOwrite 10	
...	PC+1 11	SP+1 011	AND 0011			
R7 _{in} 1000	R7 _{out} 1000		SP-1 100	OR 0100			
IR _{in} 1001	IR _{out} 1001			SHL 0101			
Y _{in} 1010	Z _{out} 1010			SHR 0110			
AR _{in} 1011	AR _{out} 1011			ROL 0111			
DRI _{in} 1100	DRI _{out} 1100			ROR 1000			
DRS _{in} 1101	DRS _{out} 1101						

*NOP为无效控制信号



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

按功能		按功能		按功能/资源		按资源		
字段1(4位)		字段2(4位)		字段3(4位)		字段4(3位)		字段5
NOP	0000	NOP	0000	NOP	0000	NOP	000	其他信号
R0 _{in}	0001	R0 _{out}	0001	ADD	0001	Mread	001	
R1 _{in}	0010	R1 _{out}	0010	SUB	0010	Mwrite	010	
...	AND	0011	IOread	011	
R7 _{in}	1000	R7 _{out}	1000	OR	0100	IOwrite	100	
IR _{in}	1001	IR _{out}	1001	SHL	0101			
Y _{in}	1010	Z _{out}	1010	SHR	0110			
AR _{in}	1011	AR _{out}	1011	ROL	0111			
DRI _{in}	1100	DRI _{out}	1100	ROR	1000			
DRS _{in}	1101	DRS _{out}	1101	PC+1	1001			
PC _{in}	1110	PC _{out}	1110	SP+1	1010			
SP _{in}	1111	SP _{out}	1111	SP-1	1011			

优化后的字段组织和编码

*NOP为无效控制信号



(基于) 微程序 (的) 控制器：控制信号编码

1. 水平型微指令控制域的编码

(2) 字段译码法 (字段编码)

也可以对**字段**进行**关联设计**，使一个域用于解释另一个域。

表6.3 采用间接译码方式的字段编码

.....	字段i (2位)	字段i+1 (2位)
	NOP 00	ADD 00	
	算术 01	SUB 01	
	逻辑 10		
	移位 11		
		AND 00	
		OR 01	
		SHL 00	
		SHR 01	
		ROL 10	
		ROR 11	



(基于) 微程序 (的) 控制器：控制信号编码

2. 垂直型微指令控制域的编码

■ 采用与机器指令相似的格式

- 微操作码：指示作何种微操作
固定长度、可变长度
- 微操作对象：
为微操作提供所需的操作数（常量或地址），一个、多个

微操作码	微操作对象
------	-------

■ 特点：

- 控制域紧凑、短小
- 并行能力差，微程序长，执行速度减慢
- 可扩展性强



(基于) 微程序 (的) 控制器：控制信号编码

3. 水平型与垂直型微指令的比较

■ 水平型微指令特性：

- 需要较长~~的~~微指令控制域；
- 可以表示高度并行的控制信号；
- 对控制域提供的控制信息只需较少的译码电路，甚至不需要译码。

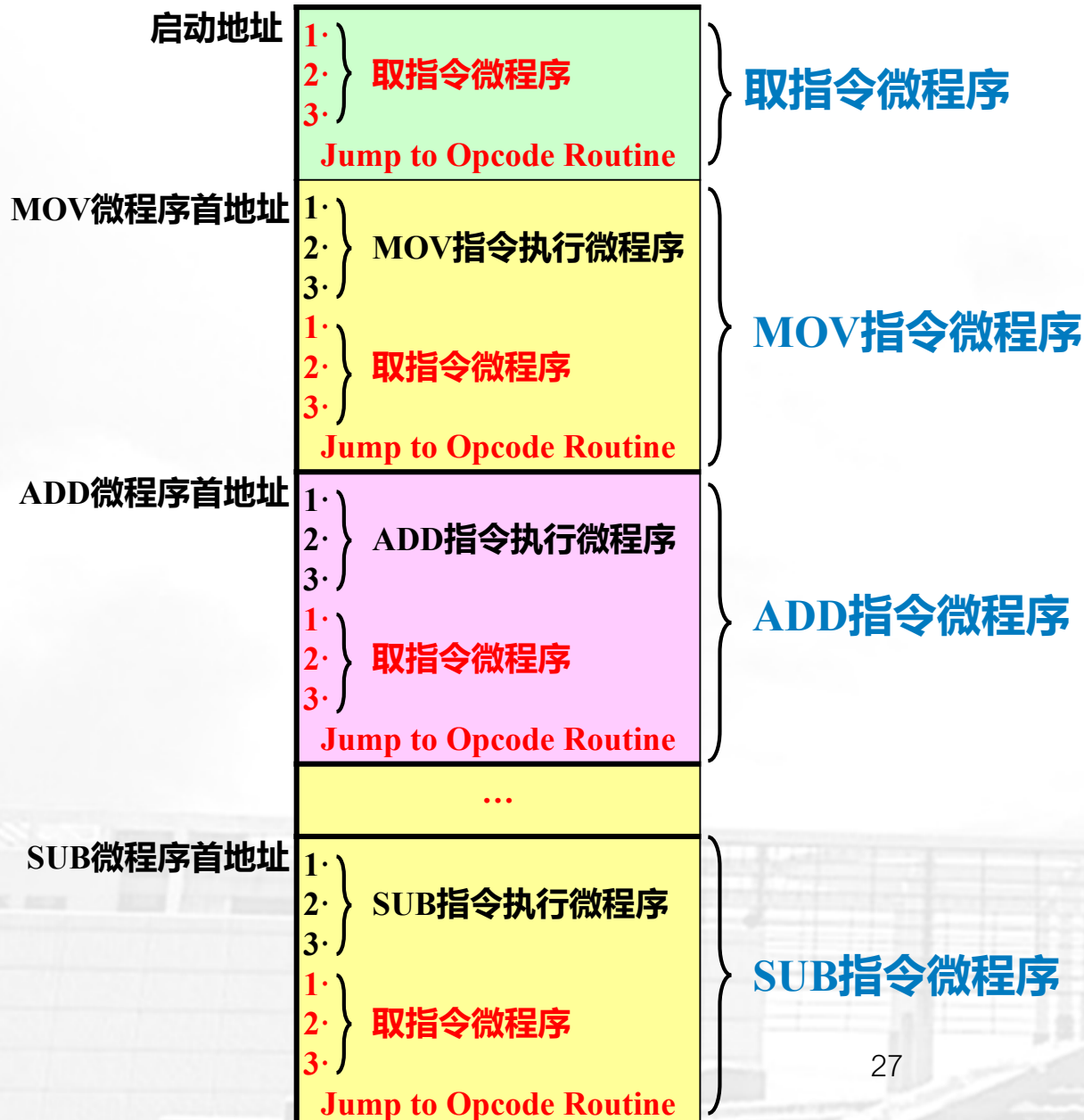
■ 垂直型微指令特性：

- 需要较短~~的~~微指令控制域；
- 并行微操作的表示能力有限；
- 对控制信息必须译码。



(基于) 微程序 (的) 控制器：控制信号编码

控制存储器组织结构 (存储微程序)



Jump to Opcode Routine:
根据**指令操作码**，跳转到
相应指令的**微程序首地址**



(基于) 微程序 (的) 控制器

6、微程序控制器与组合逻辑（硬布线）控制器的比较

■ 微程序控制器

- 比硬布线控制器速度慢
- 设计简单化、规范化
- 功能可修改、可扩充
- 实现成本低，出错概率小
- 常用于CISC处理器控制器的实现

■ 硬布线控制器

- 速度快
- 当计算机系统复杂时，设计困难
- 一旦实现，不可修改和扩充
- 常用于RISC处理器控制器的实现



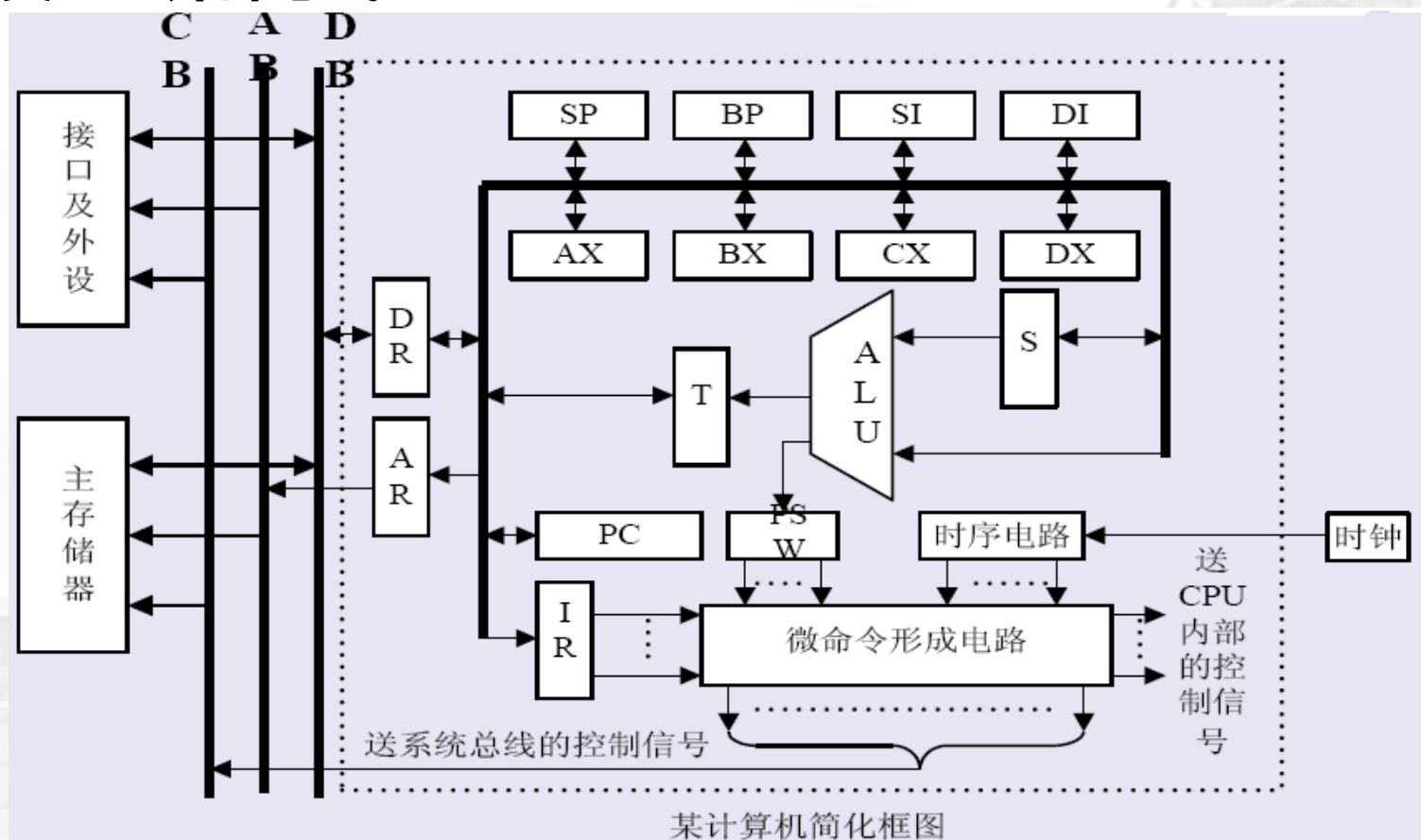
CPU设计

如何设计



CPU基本组成

1. 部件：寄存器、ALU、控制器、其它
2. 数据通路(DataPath)：总线(BUS)、专用通路
3. 外部交互：外部总线





指令系统

- 两大步骤：
 - 给指令编码 (设计编码规则)
 - 给指令解码 (让机器按规则去识别编码)



指令系统—指令编码

指令的设置

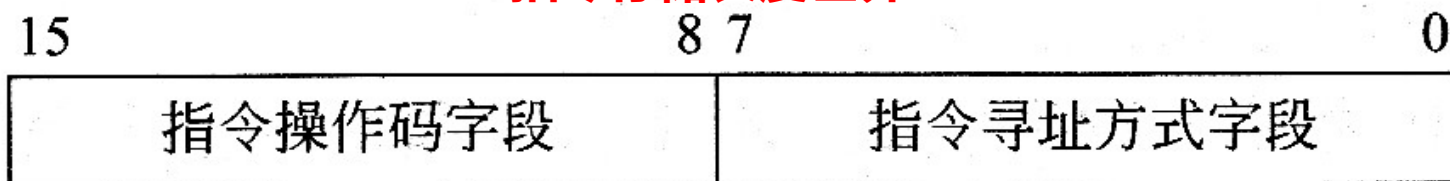
- 双操作数指令： 11条
MOV、ADD、SUB、ADC、SBC、CMP、MUL、DIV、AND、OR、XOR
- 单操作数指令： 13条
INC、DEC、PUSH、POP、NOT、SHL、SHR、SAR、ROL、ROR、RCL、RCR、CALL
- 转移指令： 9条
JMP、JZ、JNZ、JC、JNC、JG、JGE、JA、JAE
- 无操作数指令： 7条
RET、IRET、NOP、CLI、STI、SWI、DAA



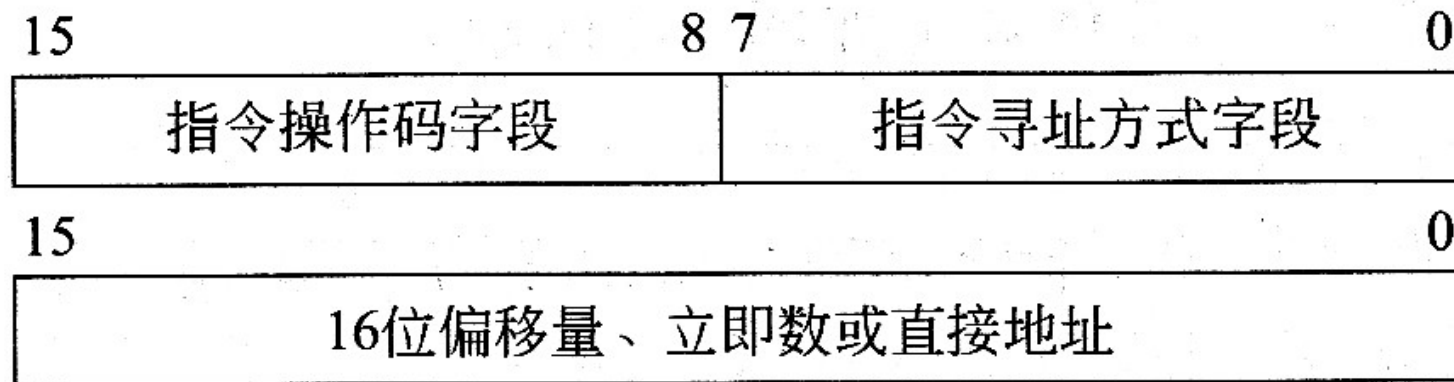
指令系统—指令编码

MOV AX,BX VS MOV AX,1000H

指令存储长度差异



(a) 单字指令



(b) 双字指令

图 5-3 指令码构成格式

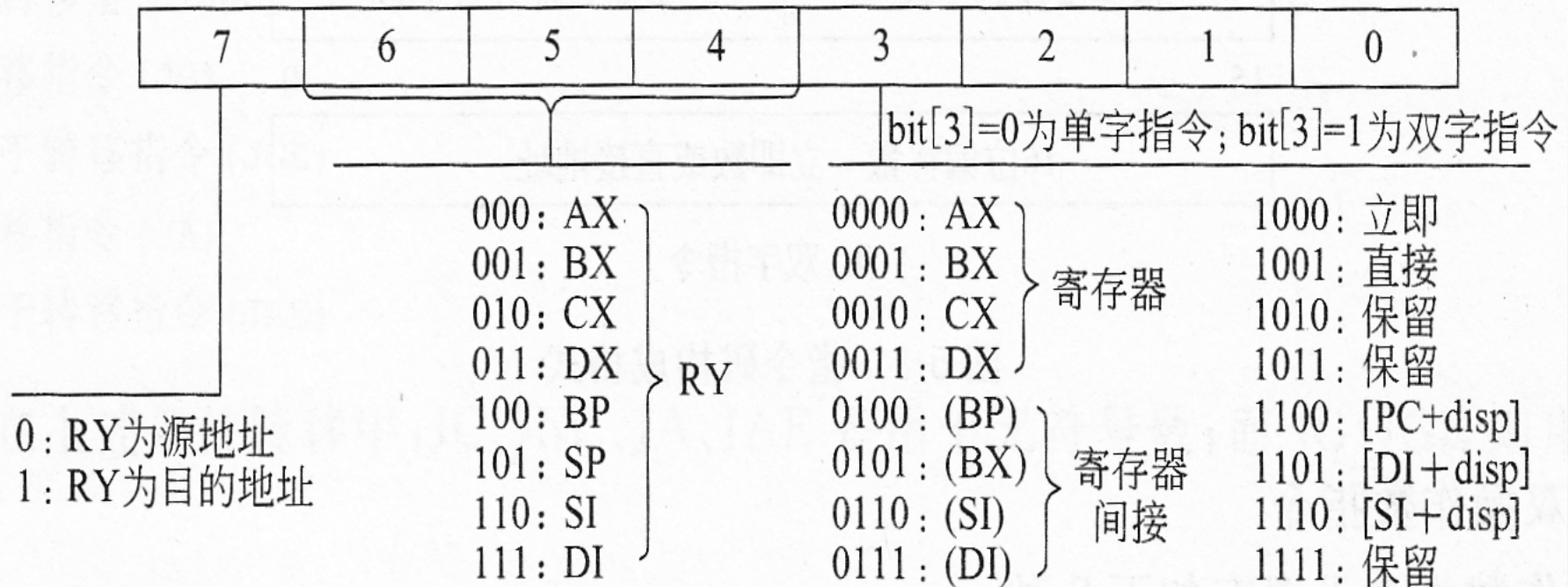


指令系统—指令编码

MOV AX,BX VS MOV AX,1000H

指令存储长度差异

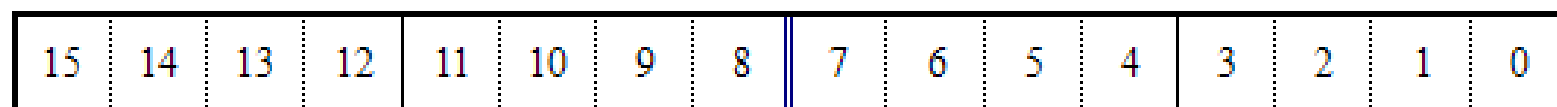
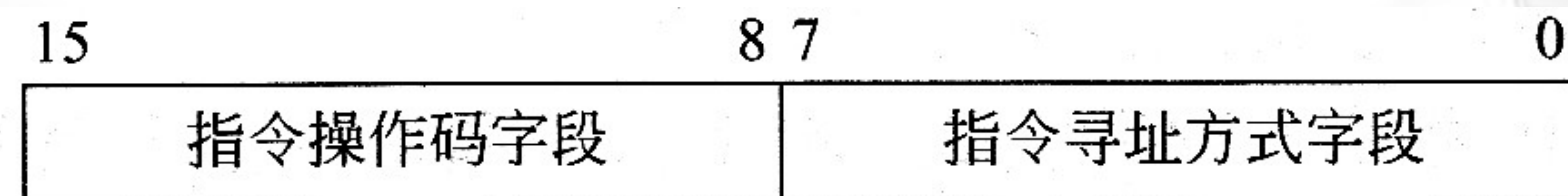
设计一个简单的寻址方式编码





指令系统—指令编码

操作码字段的编排



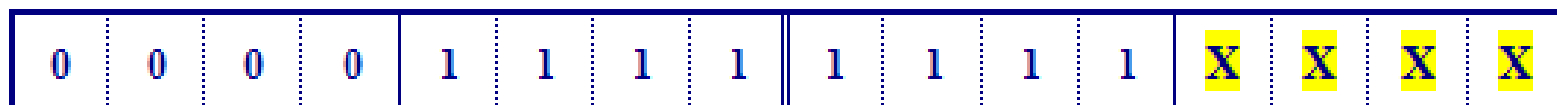
11条: 0000-1010



13条: 0000-1100



9条: 0000-1000



7条: 0000-0110



指令系统—指令编码

- 双操作数指令： 11条

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0	0	0	0	X	X	X	X	双操作数指令
0	0	0	0	0	0	0	0	MOV
				0	0	0	1	ADD
				0	0	1	0	SUB
				0	0	1	1	ADC
				0	1	0	0	SBC
				0	1	0	1	CMP
				0	1	1	0	MUL
				0	1	1	1	DIV
				1	0	0	0	AND
				1	0	0	1	OR
				1	0	1	0	XOR
				1	0	1	1	



指令系统—指令编码

- 单操作数指令： 13条

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0	0	0	0	X	X	X	X	双操作数指令							
---	---	---	---	---	---	---	---	--------	--	--	--	--	--	--	--

0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令			
---	---	---	---	---	---	---	---	---	---	---	---	--------	--	--	--

0	0	0	0	1	1	1	0	0	0	0	0	CALL
								0	0	0	1	INC
								0	0	1	0	DEC
								0	0	1	1	PUSH
								0	1	0	0	POP
								0	1	0	1	NOT
								0	1	1	0	SHL
								0	1	1	1	SHR
								1	0	0	0	SAR
								1	0	0	1	ROL
								1	0	1	0	ROR
								1	0	1	1	RCL
								1	1	0	0	RCR
								1	1	0	1	
								1	1	1	0	
								1	1	1	1	



指令系统—指令编码

- 转移指令： 9条

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0	0	0	0	X	X	X	X	双操作数指令							
---	---	---	---	---	---	---	---	--------	--	--	--	--	--	--	--

0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令			
---	---	---	---	---	---	---	---	---	---	---	---	--------	--	--	--

0	0	0	0	1	1	1	1	X	X	X	X	转移指令			
---	---	---	---	---	---	---	---	---	---	---	---	------	--	--	--

0	0	0	0	1	1	1	1	0	0	0	0	JMP
								0	0	0	1	JZ
								0	0	1	0	JNZ
								0	0	1	1	JC
								0	1	0	0	JNC
								0	1	0	1	JG
								0	1	1	0	JGE
								0	1	1	1	JA
								1	0	0	0	JAE
								1	0	0	1	



指令系统—指令编码

- 无操作数指令： 7条

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0	0	0	0	X	X	X	X	双操作数指令							
---	---	---	---	---	---	---	---	--------	--	--	--	--	--	--	--

0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令			
---	---	---	---	---	---	---	---	---	---	---	---	--------	--	--	--

0	0	0	0	1	1	1	1	X	X	X	X	转移指令			
---	---	---	---	---	---	---	---	---	---	---	---	------	--	--	--

0	0	0	0	1	1	1	1	1	1	1	1	X	X	X	X	无操作数指令	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--------	--

0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	RET
												0	0	0	1	IRET
												0	0	1	0	NOP
												0	0	1	1	CLI
												0	1	0	0	STI
												0	1	0	1	SWI
												0	1	1	0	DAA
												0	1	1	1	



指令系统—指令编码

- 根据以上指令规范，我们可以为下面的指令给出操作码编码：

- ADD AX [SI]
 - 01 86 (单字)
- Mov AX [2000H]
 - 00 89 20 00 (双字)
- INC [BX]
 - 0E 15 (单字)

0	0	0	0	X	X	X	X	双操作数指令
				0	0	0	0	MOV
				0	0	0	1	ADD
				0	0	1	0	SUB
				0	0	1	1	ADC
				0	1	0	0	SBC
0	0	0	0	0	1	0	1	CMP
				0	1	1	0	MUL
				0	1	1	1	DIV
				1	0	0	0	AND
				1	0	0	1	OR
				1	0	1	0	XOR
				1	0	1	1	

0	0	0	0	X	X	X	X	双操作数指令				
0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令
								0	0	0	0	CALL
								0	0	0	1	INC
								0	0	1	0	DEC
								0	0	1	1	PUSH
								0	1	0	0	POP
								0	1	0	1	NOT
								0	1	1	0	SHL
0	0	0	0	1	1	1	0	0	1	1	1	SHR
								1	0	0	0	SAR

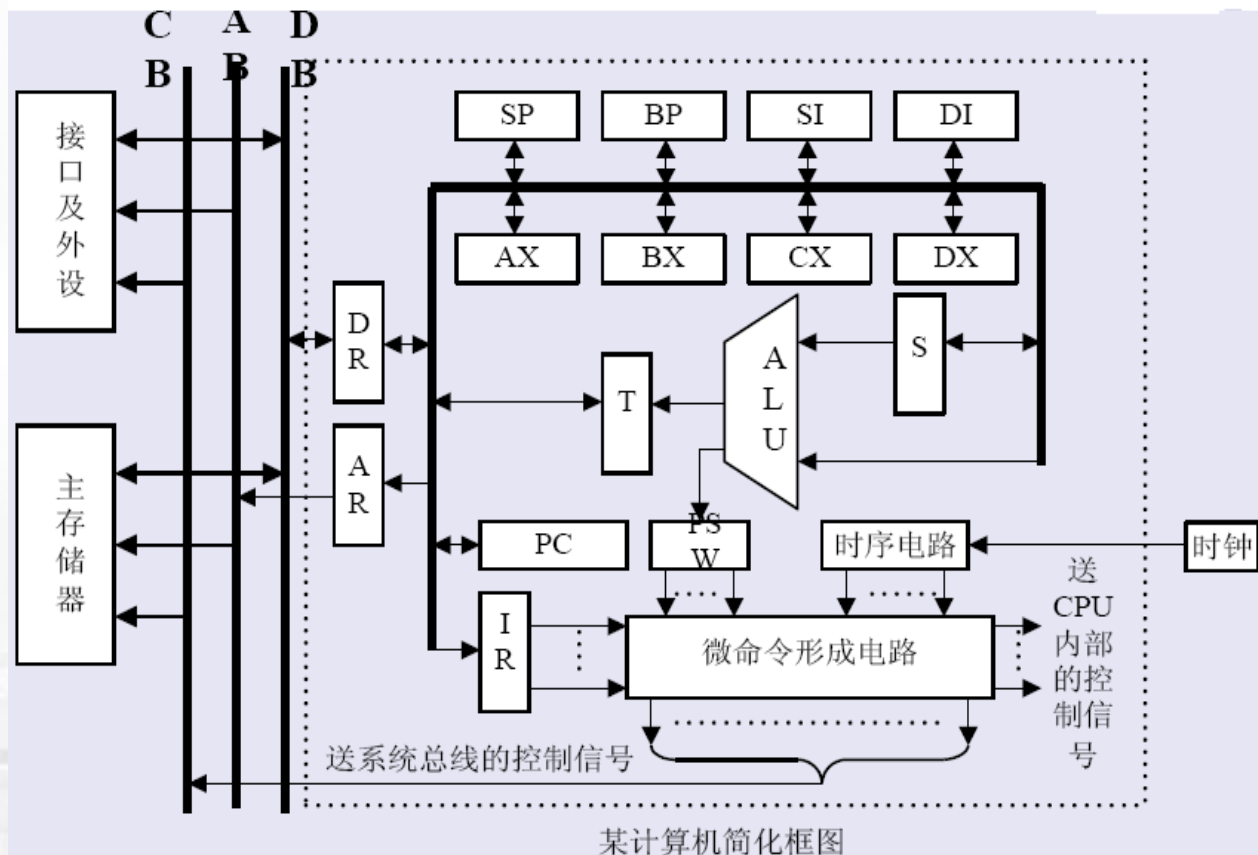
7	6	5	4	3	2	1	0	
				bit[3]=0为单字指令; bit[3]=1为双字指令				
				000: AX				0000: AX
				001: BX				0001: BX
				010: CX				0010: CX
				011: DX				0011: DX
				100: BP				0100: (BP)
				101: SP				0101: (BX)
				110: SI				0110: (SI)
				111: DI				0111: (DI)
								1000: 立即
								1001: 直接
								1010: 保留
								1011: 保留
								1100: [PC+disp]
								1101: [DI+disp]
								1110: [SI+disp]
								1111: 保留



指令系统—指令运行

a) 取指令过程：

● 计算机简单框图



某计算机简化框图

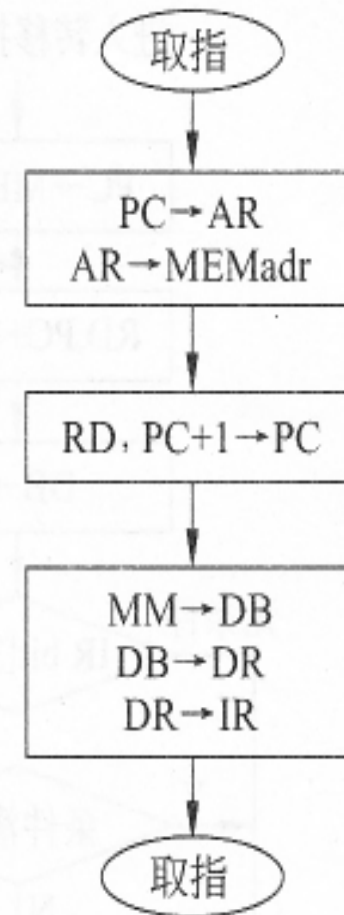


图 5-7 取指过程



指令系统—指令运行

b) 译码取操作码过程:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	X	X	X	X	双操作数指令								
0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令				
0	0	0	0	1	1	1	1	X	X	X	X	转移指令				
0	0	0	0	1	1	1	1	1	1	1	1	X	X	X	X	无操作数指令

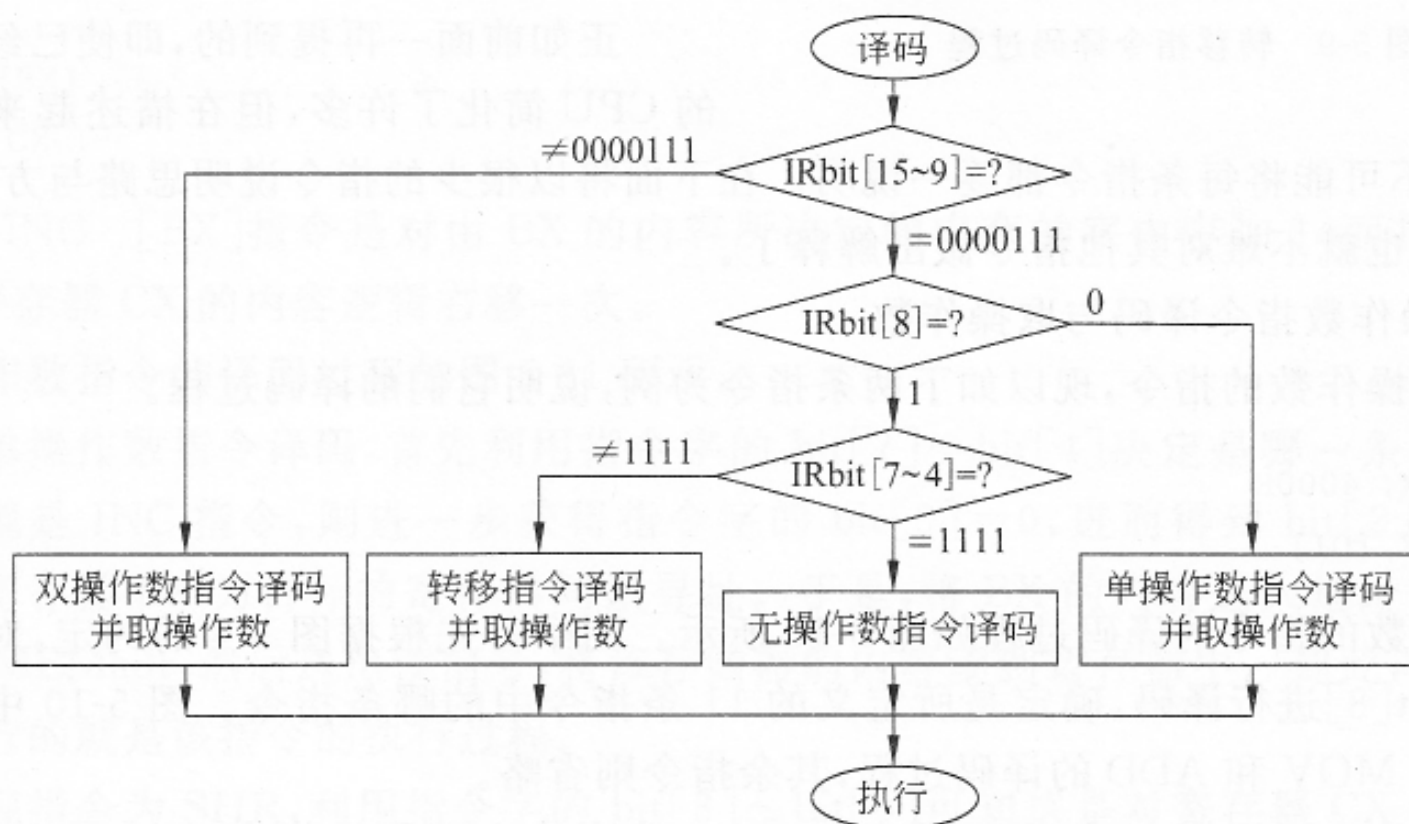
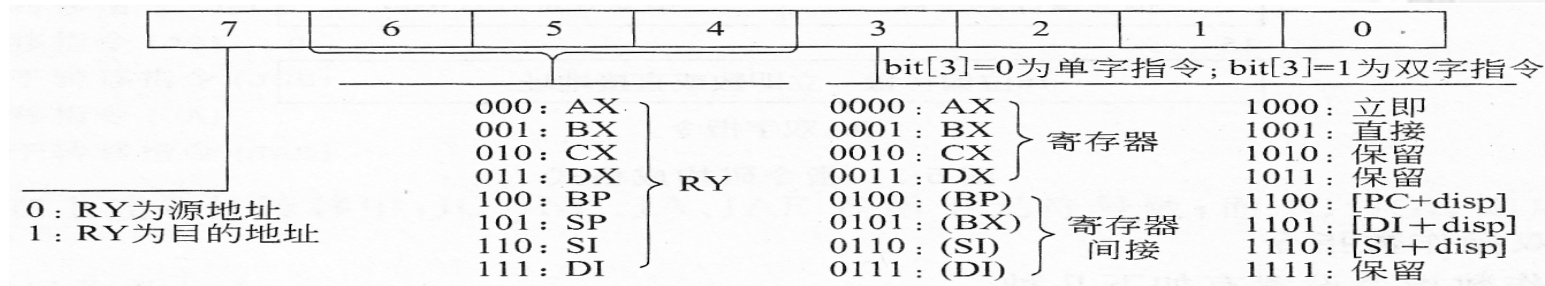


图 5-8 译码取操作数过程框图



指令系统—指令运行

- 双操作数指令译码取操作数过程（续）：



- ADD BX [DI]
- MOV AX 4000H

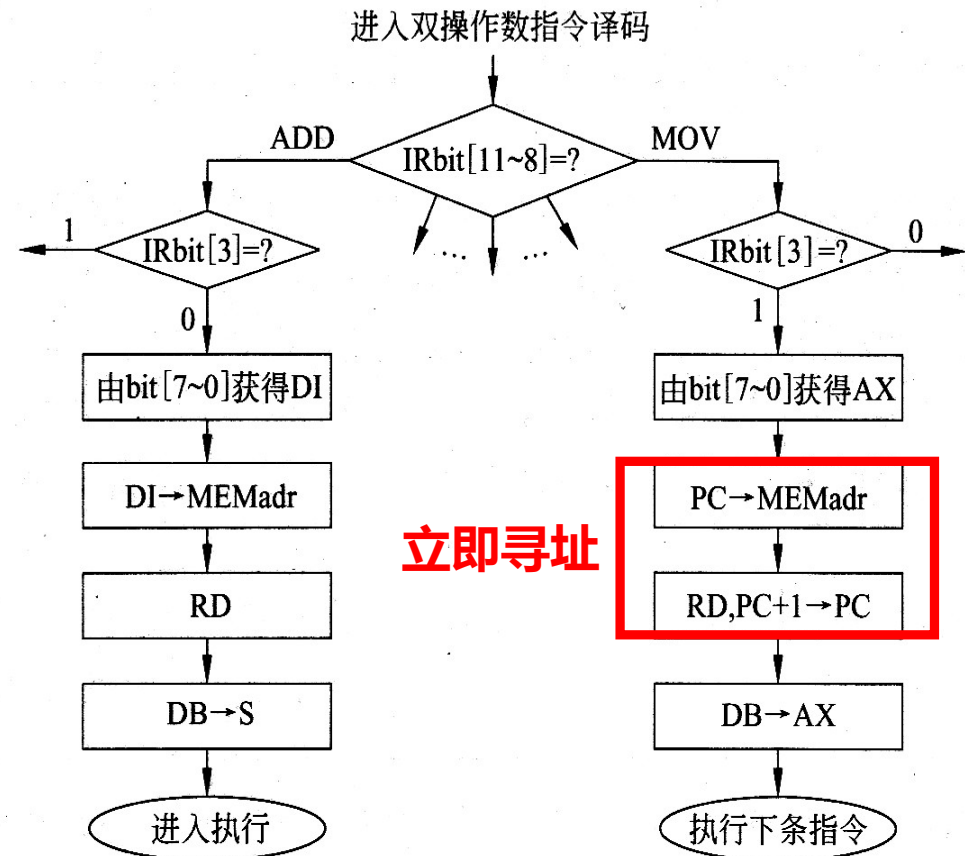


图 5-10 双操作数指令译码过程



指令系统—指令运行

• 单操作数指令译码取操作数过程：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	X	X	X	X	单操作数指令			
0	0	0	0	1	1	1	0	0	0	0	0	CALL			
								0	0	0	1	INC			
								0	0	1	0	DEC			
								0	0	1	1	PUSH			
								0	1	0	0	POP			
								0	1	0	1	NOT			
								0	1	1	0	SHL			
								0	1	1	1	SHR			
								1	0	0	0	SAR			
								1	0	0	1	ROL			
								1	0	1	0	ROR			
								1	0	1	1	RCL			
								1	1	0	0	RCR			
								1	1	0	1				
								1	1	1	0				
								1	1	1	1				

- SHR CX
- INC [BX]

3	2	1	0	
bit[3]=0为单字指令; bit[3]=1为双字指令				
0000	AX	寄存器	1000	立即
0001	BX		1001	直接
0010	CX		1010	保留
0011	DX		1011	保留
0100	(BP)	寄存器 间接	1100	[PC+disp]
0101	(BX)		1101	[DI+disp]
0110	(SI)		1110	[SI+disp]
0111	(DI)		1111	保留

进入单操作数指令译码

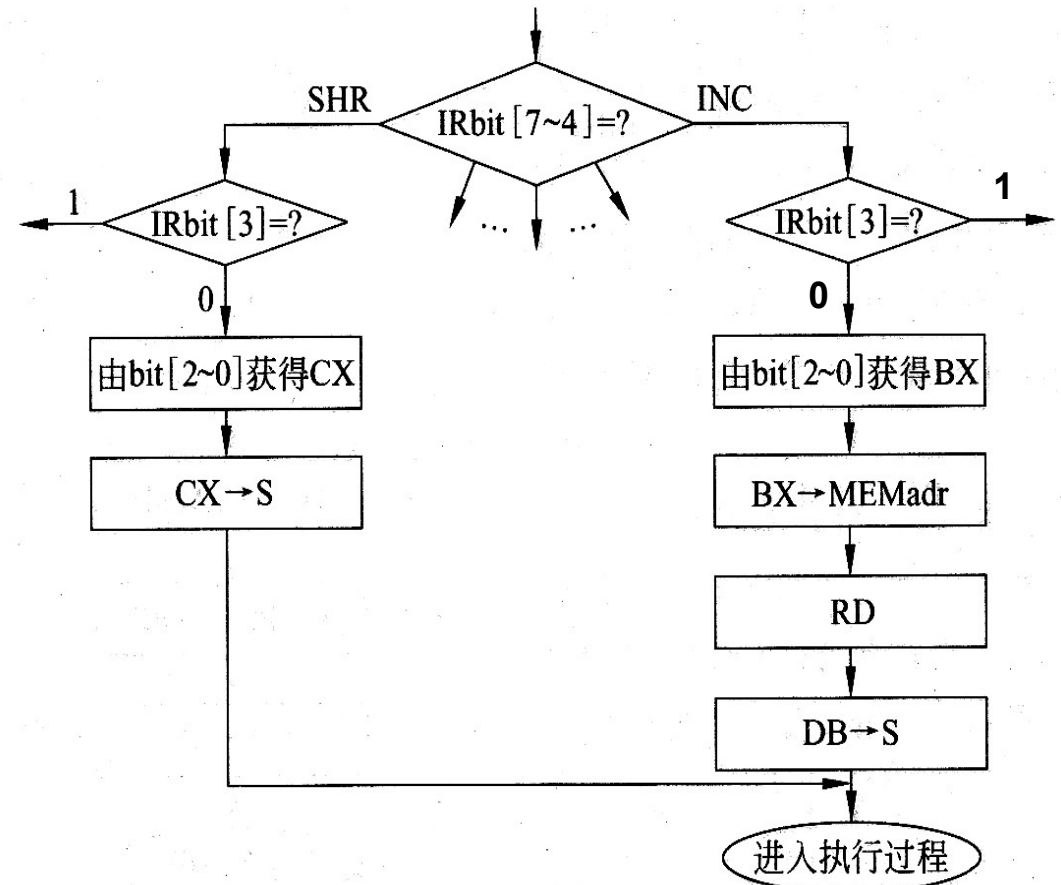


图 5-11 单操作数指令译码过程



指令系统—指令运行

- 无操作数指令译码取操作数过程：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	1	1	1	1	1	1	1	1	X	X	X	X	
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	RET
												0	0	0	1	IRET
												0	0	1	0	NOP
												0	0	1	1	CLI
												0	1	0	0	STI
												0	1	0	1	SWI
												0	1	1	0	DAA
												0	1	1	1	
												1	1	1	1	

无操作数指令

IRET
0F F1
NOP
0F F2

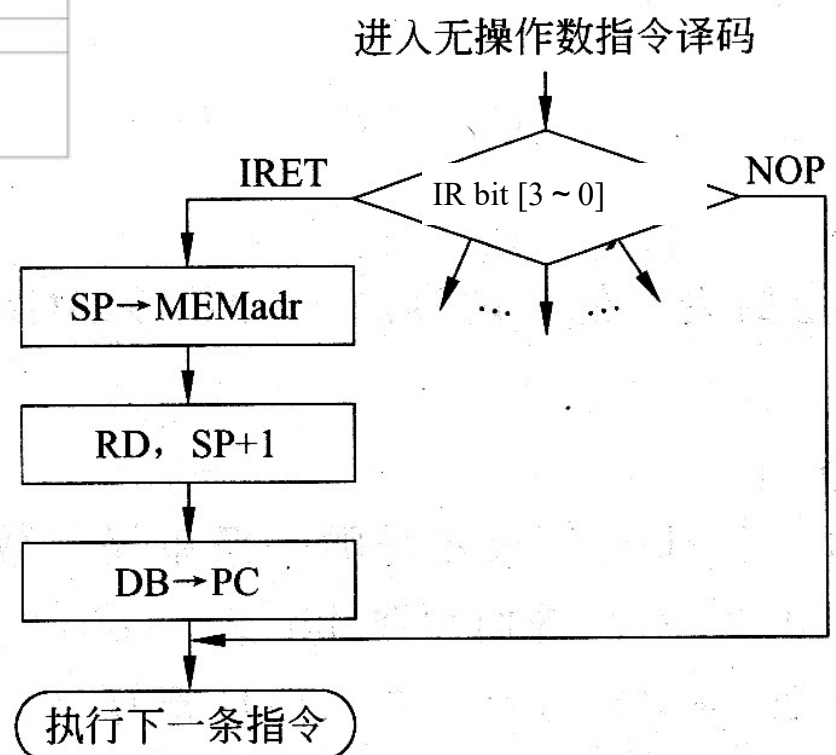


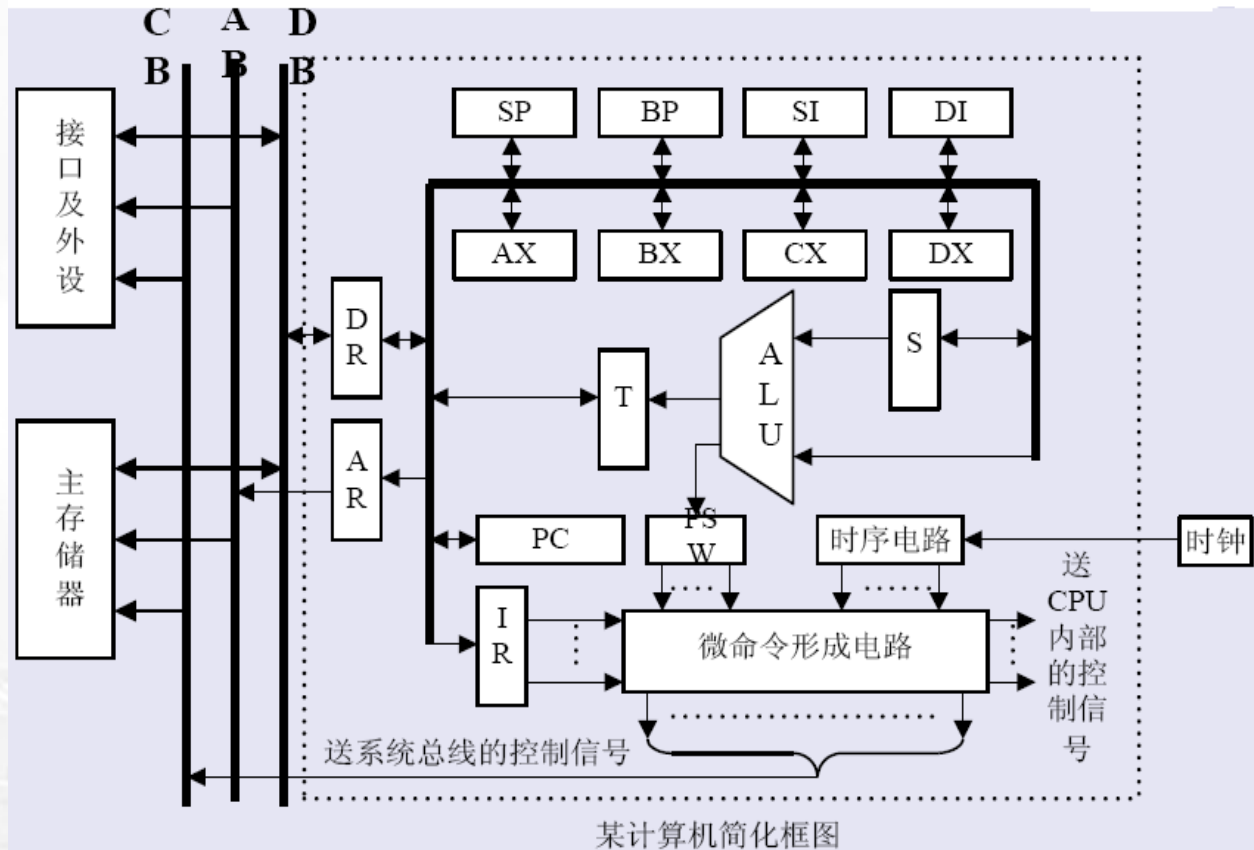
图 5-12 无操作数指令的译码过程



指令系统—指令运行

c) 指令执行过程:

● 计算机简单框图



某计算机简化框图

JMP disp

进入转移指令执行

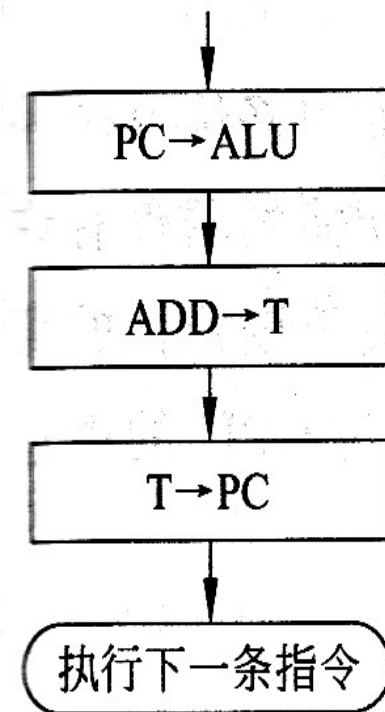


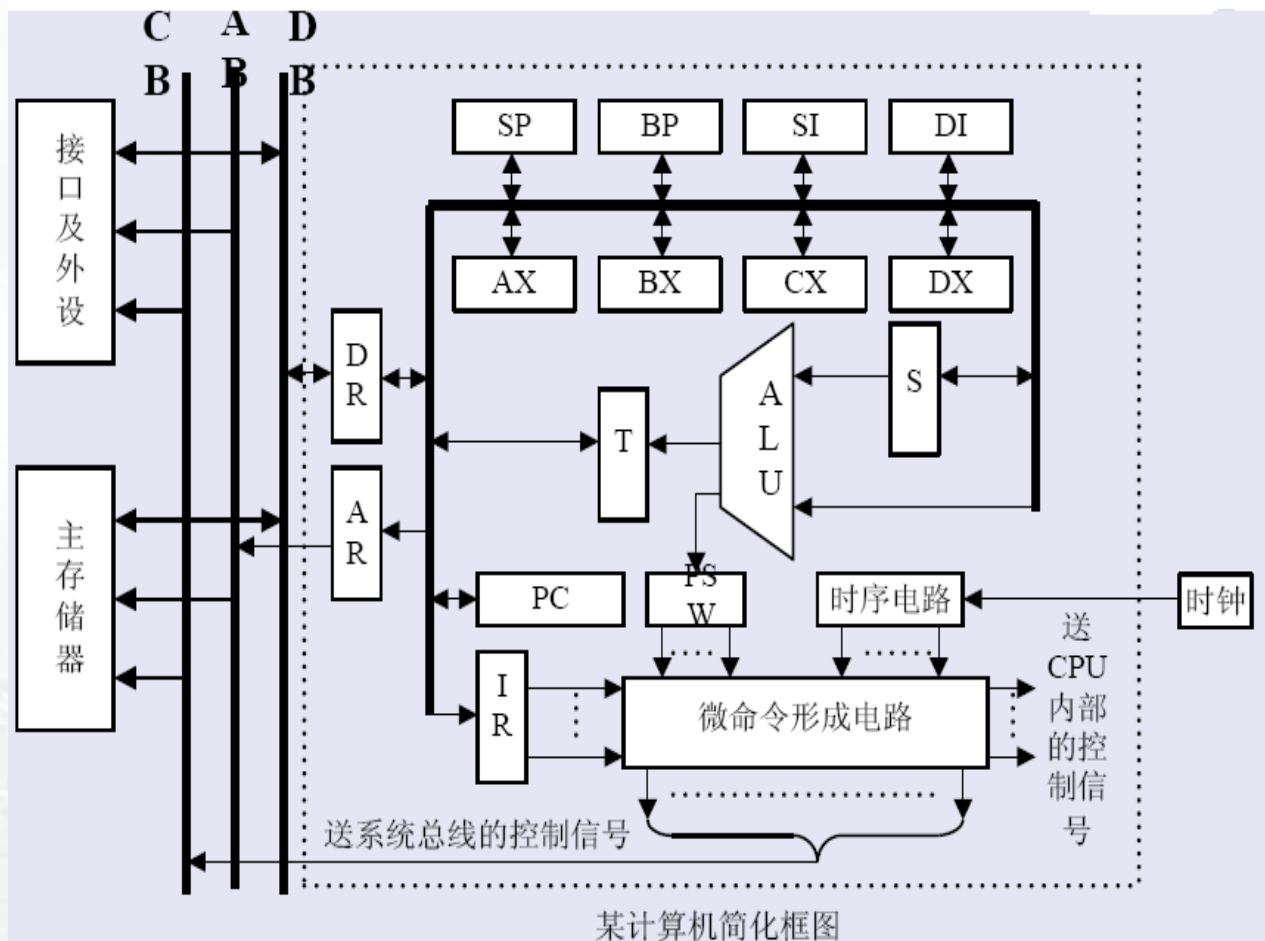
图 5-13 转移指令执行



指令系统—指令运行

c) 指令执行过程(续):

● 计算机简单框图



ADD BX, [DI]

进入ADD指令执行

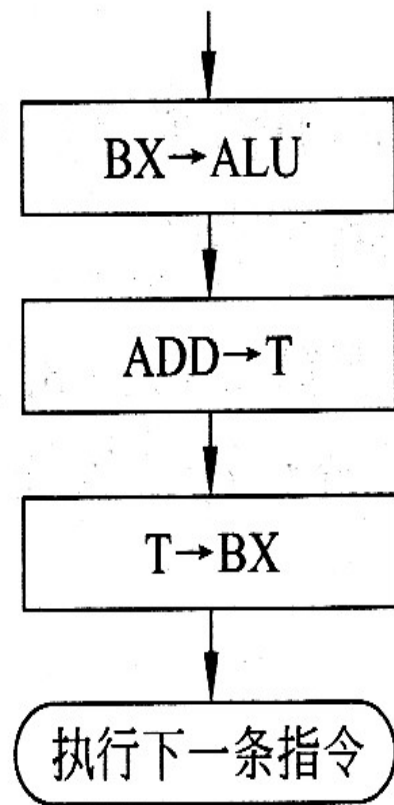


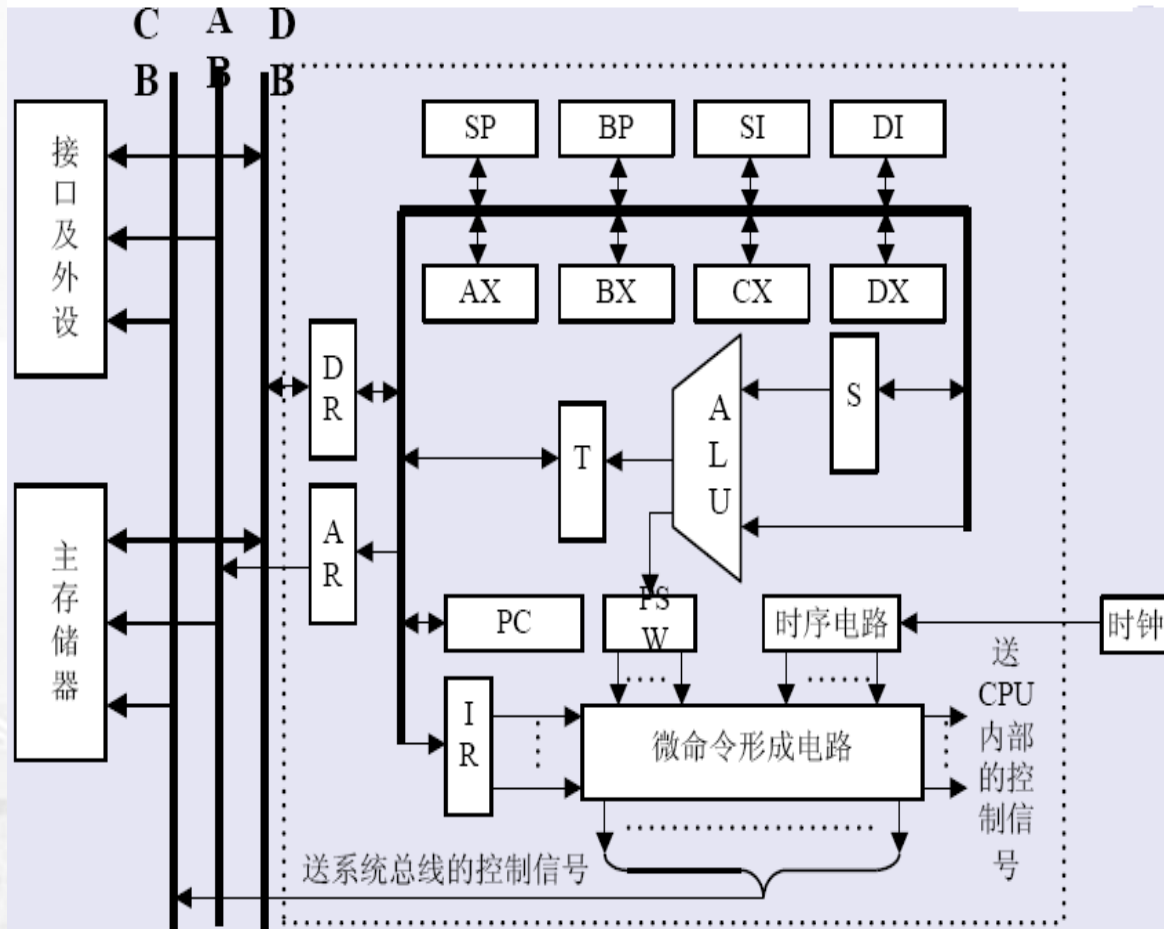
图 5-14 ADD 指令执行



指令系统—指令运行

c) 指令执行过程(续):

● 计算机简单框图



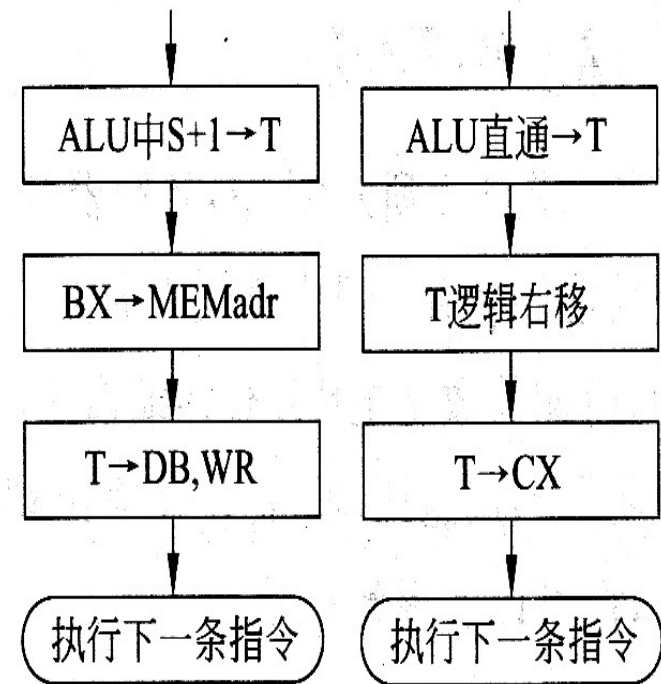
某计算机简化框图

INC [BX]

SHR CX

进入INC指令执行

进入SHR指令执行



(a)

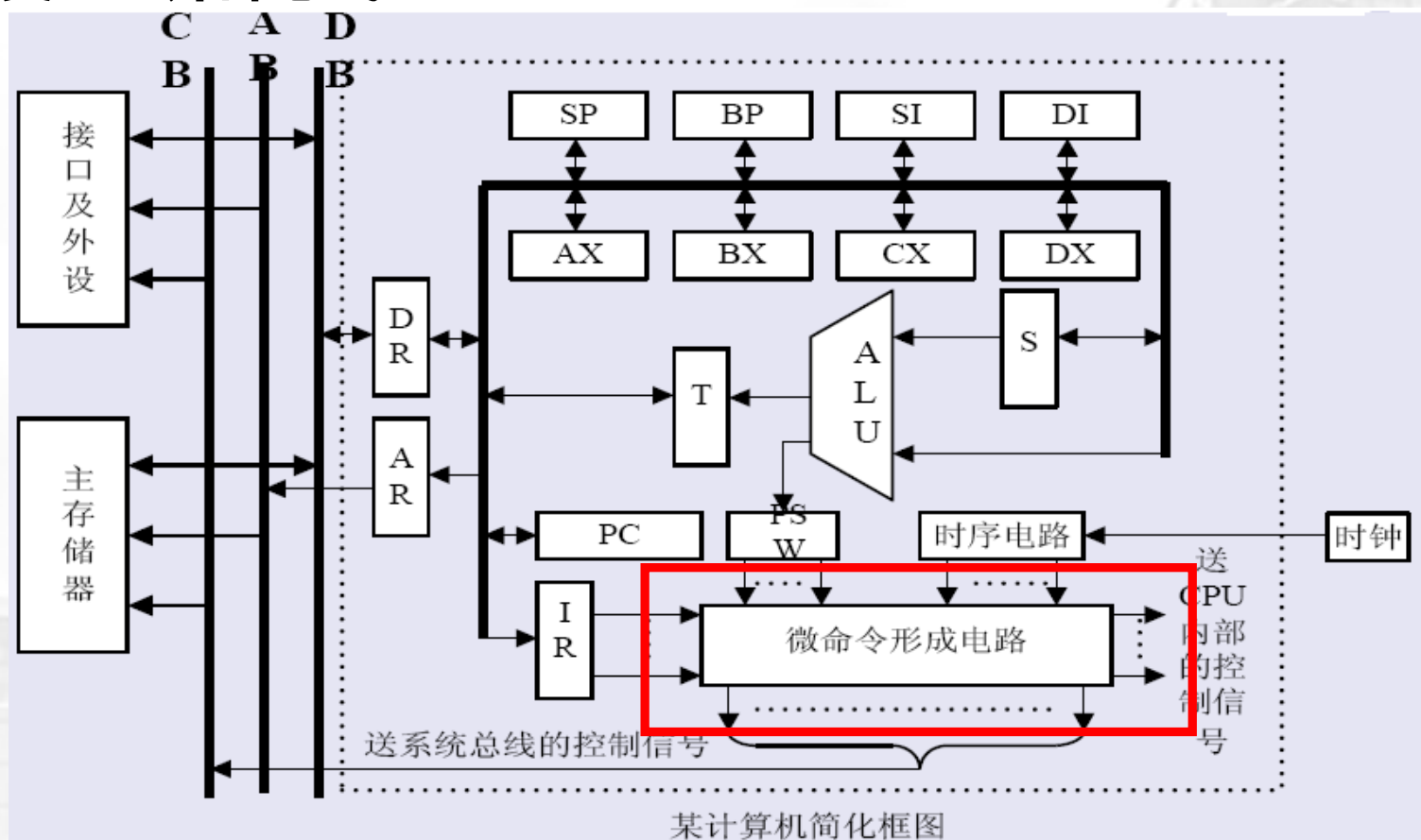
(b)

图 5-15 INC 指令及 SHR 指令执行过程



微程序编写

1. 部件：寄存器、ALU、**控制器**、其它
2. 数据通路(DataPath)：总线(BUS)、专用通路
3. 外部交互：外部总线





微程序编写

如何编写微程序？

1、按照设计好的微指令格式，将指令微操作(微命令)序列

按每节拍一条微指令写出每条微指令的具体编码；（包括地址域和控制域）

2、按照选定的微程序结构，将微指令组织成微程序或微子程序



(基于) 微程序 (的) 控制器: 微指令编码

1、按每节拍一条微指令写出每条微指令的具体编码;

控制域 (15位)

地址域 (1位)

微指令编码:

字段1(4位)		字段2(4位)		字段3(4位)		字段4(3位)		AC(1位)
NOP	0000	NOP	0000	NOP	0000	NOP	000	顺序执行 0
AX _{in}	0001	AX _{out}	0001	ADD	0001	Mread	001	根据IR中 指令操作码, 跳转到相 应指令的 微程序首地址
BX _{in}	0010	BX _{out}	0010	SUB	0010	Mwrite	010	
...	AND	0011	IOread	011	
DX _{in}	1000	DX _{out}	1000	OR	0100	IOwrite	100	
IR _{in}	1001	IR _{out}	1001	SHL	0101			
Y _{in}	1010	Z _{out}	1010	SHR	0110			
AR _{in}	1011	AR _{out}	1011	ROL	0111			
DRI _{in}	1100	D R I _{o u t}		ROR	1000			
DRS _{in}	1101	1 1 0 0		PC+1	1001			
PC _{in}	1110	D R S _{o u t}		SP+1	1010			
SP _{in}	1111	1 1 0 1		SP-1	1011			
		PC _{out}	1110					
		SP _{out}	1111					

*NOP为无效控制信号



(基于) 微程序 (的) 控制器：控制信号编码

编写微程序：

- 1、按照设计好的微指令格式，将指令微操作(微命令)序列按每节拍一条微指令写出每条微指令的具体编码；
- 2、按照选定的微程序结构，将微指令组织成微程序或微子程序

微程序名	地址	微指令					节拍	微操作	微命令
取指	000H	1011	1110	0000	000	0	T1	AR←PC	PC _{out} , AR _{in}
	001H	1101	1011	0000	001	0	T2	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	002H	1001	1100	1001	000	1	T3	PC=PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}
MOV AX,[X]	160H	1011	1110	0000	000	0	T1	AR←PC	PC _{out} , AR _{in}
	161H	1101	1011	0000	001	0	T2	DR←Memory[AR]	AR _{out} , Mread, DR _{in}
	162H	1011	1100	1001	000	0	T3	PC=PC+1, AR←DR	PC+1, DRI _{out} , AR _{in}
	163H	1101	1011	0000	001	0	T4	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	164H	0001	1100	0000	000	0	T5	AX←DR	DRI _{out} , AX _{in}
	165H	1011	1110	0000	000	0	T6	AR←PC	PC _{out} , AR _{in}
	166H	1101	1011	0000	001	0	T7	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	167H	1001	1100	1001	000	1	T8	PC←PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}
ADD AX,BX	210H	1010	0001	0000	000	0	T1	S←BX	BX _{out} , S _{in}
	211H	0000	0010	0001	000	0	T2	T←AX+S	AX _{out} , ADD
	212H	0010	1010	0000	000	0	T3	AX←T	T _{out} , AX _{in}
	213H	1011	1110	0000	000	0	T4	AR←PC	PC _{out} , AR _{in}
	214H	1101	1011	0000	001	0	T5	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	215H	1001	1100	1001	000	1	T6	PC←PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}



微程序名	地址	微指令					节拍	微操作	微命令
取指	000H	1011	1110	0000	000	0	T1	AR←PC	PC _{out} , AR _{in}
	001H	1101	1011	0000	001	0	T2	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	002H	1001	1100	1001	000	1	T3	PC=PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}
MOV AX,[X]	160H	1011	1110	0000	000	0	T1	AR←PC	PC _{out} , AR _{in}
	161H	1101	1011	0000	001	0	T2	DR←Memory[AR]	AR _{out} , Mread, DR _{in}
	162H	1011	1100	1001	000	0	T3	PC=PC+1, AR←DR	PC+1, DRI _{out} , AR _{in}
	163H	1101	1011	0000	001	0	T4	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	164H	0001	1100	0000	000	0	T5	AX←DR	DRI _{out} , AX _{in}
	165H	1011	1110	0000	000	0	T6	AR←PC	PC _{out} , AR _{in}
	166H	1101	1011	0000	001	0	T7	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	167H	1001	1100	1001	000	1	T8	PC←PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}
ADD AX,BX	210H	1010	0001	0000	000	0	T1	S←BX	BX _{out} , S _{in}
	211H	0000	0010	0001	000	0	T2	T←AX+S	AX _{out} , ADD
	212H	0010	1010	0000	000	0	T3	AX←T	T _{out} , AX _{in}
	213H	1011	1110	0000	000	0	T4	AR←PC	PC _{out} , AR _{in}
	214H	1101	1011	0000	001	0	T5	DR←Memory[AR]	AR _{out} , Mread, DRS _{in}
	215H	1001	1100	1001	000	1	T6	PC←PC+1, IR←DR	PC+1, DRI _{out} , IR _{in}

字段1(4位)		字段2(4位)		字段3(4位)		字段4(3位)		AC(1位)	
NOP	0000	NOP	0000	NOP	0000	NOP	000	顺序执行 0	
AX _{in}	0001	AX _{out}	0001	ADD	0001	Mread	001	根据IR中 1	
BX _{in}	0010	BX _{out}	0010	SUB	0010	Mwrite	010	指令操作码,	
...	AND	0011	IOread	011	跳转到相	
DX _{in}	1000	DX _{out}	1000	OR	0100	IOwrite	100	应指令的	
IR _{in}	1001	IR _{out}	1001	SHL	0101			微程序首地址	
Y _{in}	1010	Z _{out}	1010	SHR	0110				
AR _{in}	1011	AR _{out}	1011	ROL	0111				
DRI _{in}	1100	D R I _{o u t}		ROR	1000				
DRS _{in}	1101	1 1 0 0		PC+1	1001				
PC _{in}	1110	D R S _{o u t}		SP+1	1010				
SP _{in}	1111	1 1 0 1		SP-1	1011				
		PC _{out}	1110						
		SP _{out}	1111						

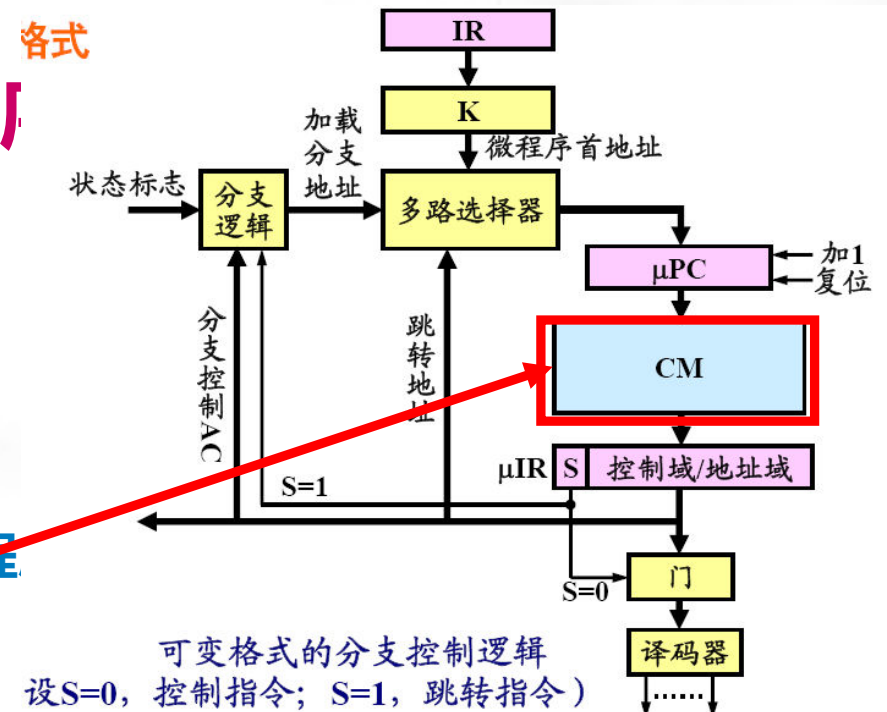
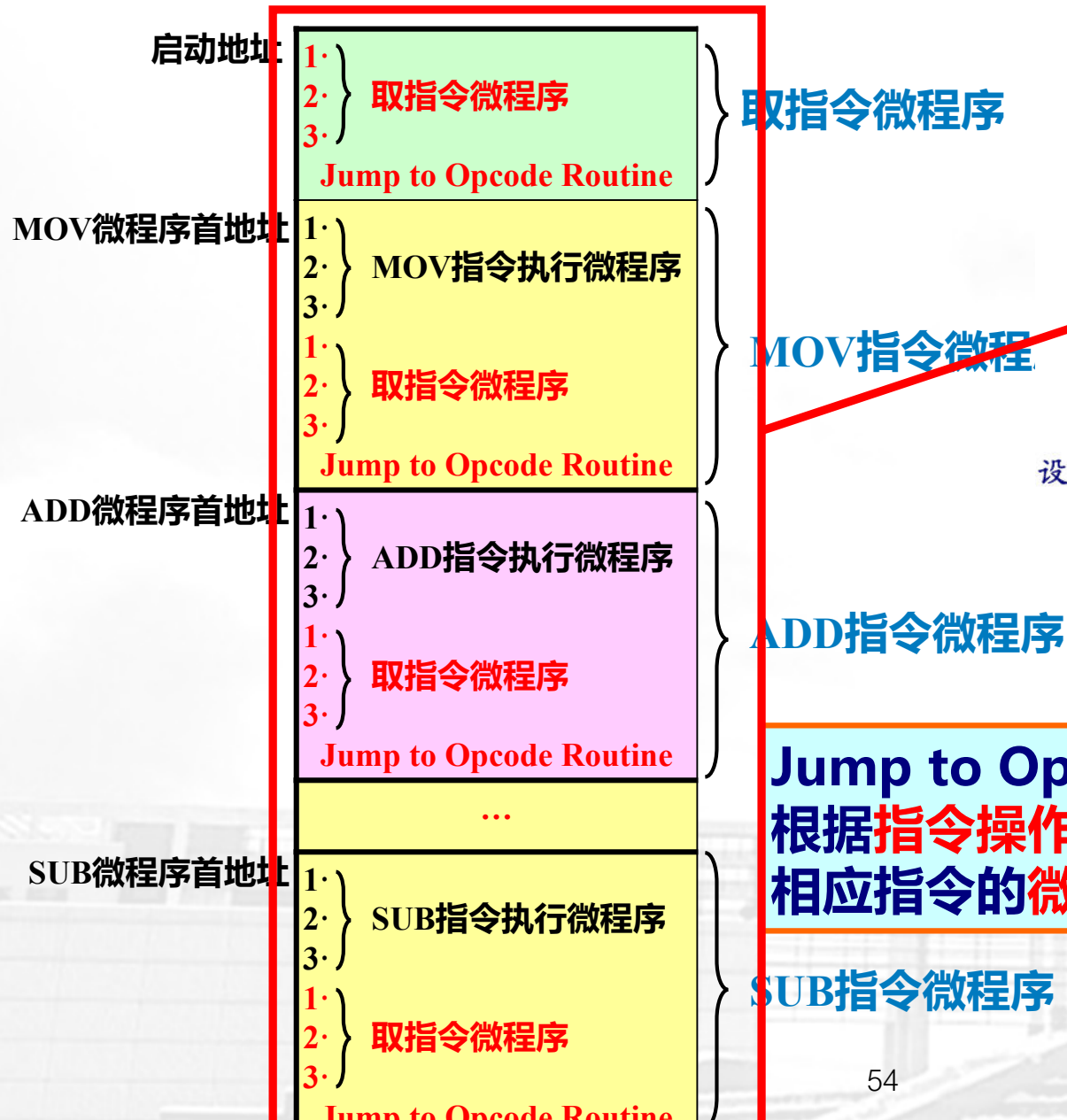
*NOP为无效控制信号



(基于) 微程序 (的) 控制器：控制信号编码

各式

控制存储器组织结构1 (存储微程)



Jump to Opcode Routine:
根据指令操作码, 跳转到
相应指令的微程序首地址



作业

• 8、13、14、17

