国防科技大学计算机学院

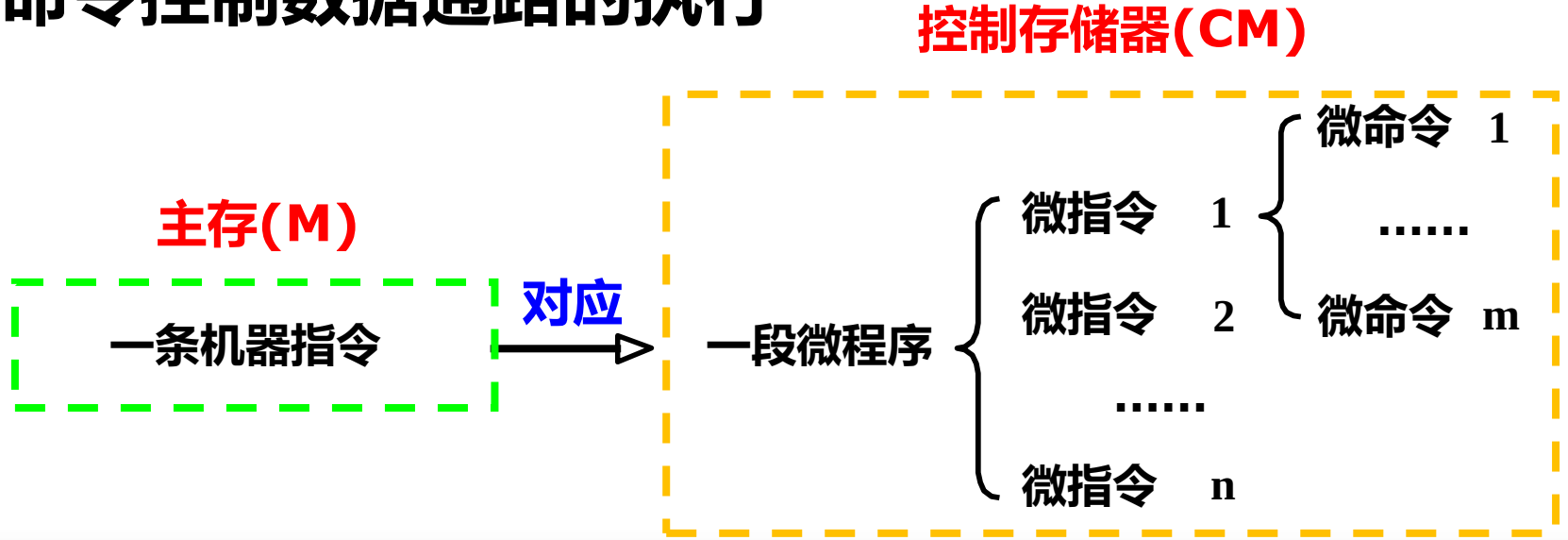
计算机原理

COMPUTER PRINCIPLE

第四章 第五节 (3) 微程序控制器设计的关键问题

1. 微程序\微指令\微命令\微操作的关系

- 将机器指令的执行转换为微程序的执行
- 微程序是一个微指令序列
- 每条微指令是一串0/1序列，包含若干个微命令(即：控制信号)
- 每个微命令控制数据通路的执行



1. 微程序\微指令\微命令\微操作的关系

- 将机器指令的执行转换为微程序的执行
- 微程序是一个微指令序列
- 每条微指令是一串0/1序列，包含若干个微命令(即：控制信号)
- 每个微命令控制数据通路的执行

控制程序执行要解决什么问题？

(1)指令的编码和译码

(2)下条指令到哪里去取

微程序执行也要解决两个问题：

(1)微指令中如何对微命令编码

(2)下条微指令在哪里

2. 微指令格式的设计

- 微指令中包含若干微命令、下条微指令地址(可选)、常数(可选)

微指令格式：

μOP：微操作码字段，产生微命令

μADD：微地址码字段，产生下条微指令地址

- 微指令格式设计风格取决于微操作码的编码方式

- 微操作码编码方式
  - 不译法（直接控制法）
  - 字段直接编码法
  - 字段间接编码法

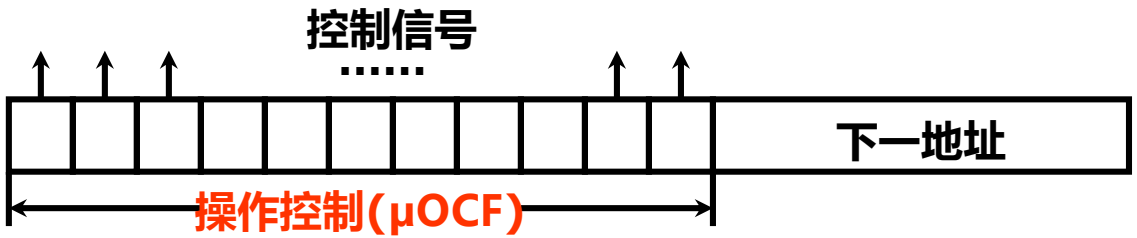
  } 水平型微指令风格

  - 最短（垂直）编码法 —— 垂直型微指令风格

2. 微指令格式的设计

❑ 不译法(直接控制编码)

- 微命令产生不必译码，从操作控制字段直接得到，即每一个微命令用一位信息表示

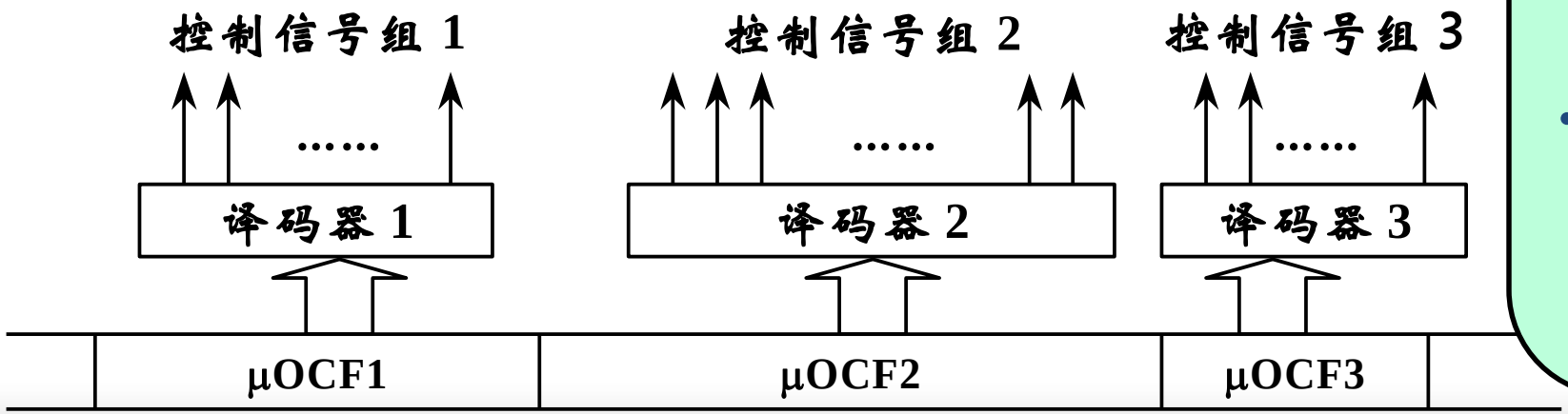


- 优点：并行控制能力强 ，不需译码，控制电路简单、速度快，编制的微程序短
- 缺点：微指令字长，编码空间利用率低，控制存储器的容量大
- 适于简单、高速控制部件，比如YH-1指令流水部件

2. 微指令格式的设计

❑ 字段直接编码

- 将微操作控制字段划分为若干小字段，每个小字段单独编码，每个字段表示一种微命令，**每段经译码后发出控制信号**
- **优点**：微指令分段数越多，并行控制能力越强；微指令字较短，能压缩到不译法的1/2到1/3，节省了控存容量
- **缺点**：增加译码电路，需开销一部分译码时间
- **已为大多数微程序控制的计算机所采用**

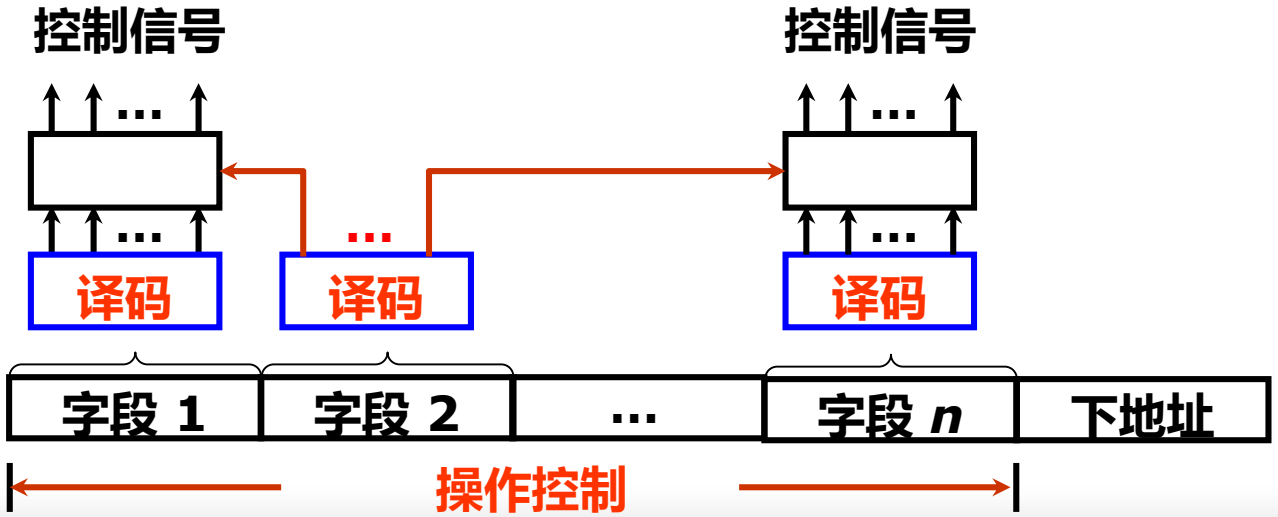


- **相容微操作**：能同时进行的微操作。对应的微命令称为相容微命令。相容的微命令分在不同字段。
- **互斥微操作**：不能同时进行的微操作。对应的微命令称为互斥微命令。互斥的微命令分在同一字段。

2. 微指令格式的设计

❑ 字段间接编码

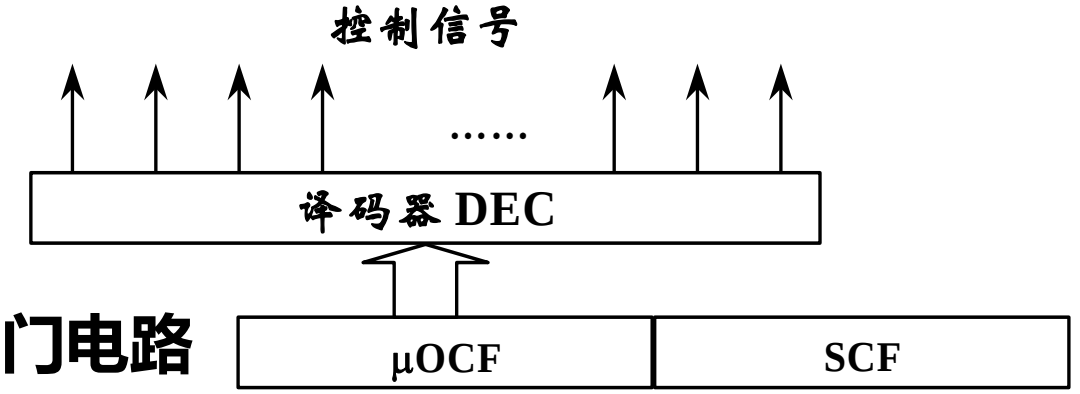
- 某些参与编码的微命令不能由一个控制字段直接定义，而需要两个或两个以上的控制字段来定义。即：一个微命令字段可以表示多个微命令组，到底代表哪一组微命令，则由另一个专门的字段来确定
- 优点：牺牲并行性、速度换取微指令字长的缩短，进一步节省控存容量。(意义不大！)
- 缺点：译码电路复杂，时间开销大
- 只限于局部场合使用



2. 微指令格式的设计

❑ 最短编码

- 将所有微命令统一二进制编码，每条微指令只包含一个微命令，每次只产生一个微操作，通过译码器产生微操作控制信号
- 优点：微程序规整、直观，易于编制，微指令字长短
- 缺点很严重（**不实用！**）：
  - 微程序长
  - 硬件设备复杂，需要大量的译码电路和门电路
  - 速度慢，每次只能产生一个微命令，难以提高微指令的执行速度



2. 微指令格式的设计

❑水平型微指令

- **内涵：**相容微命令尽量多地安排在一条微指令中
- **优点：**微指令译码简单，微指令能最大限度地表示微操作的并行性，编制的微程序短，微程序的执行速度较快，适合于较高速度的场合
- **缺点：**微指令字较长，编码空间利用率较低，且编制最佳水平微程序难度较大
- **水平型微指令是面向处理器内部控制逻辑的描述**

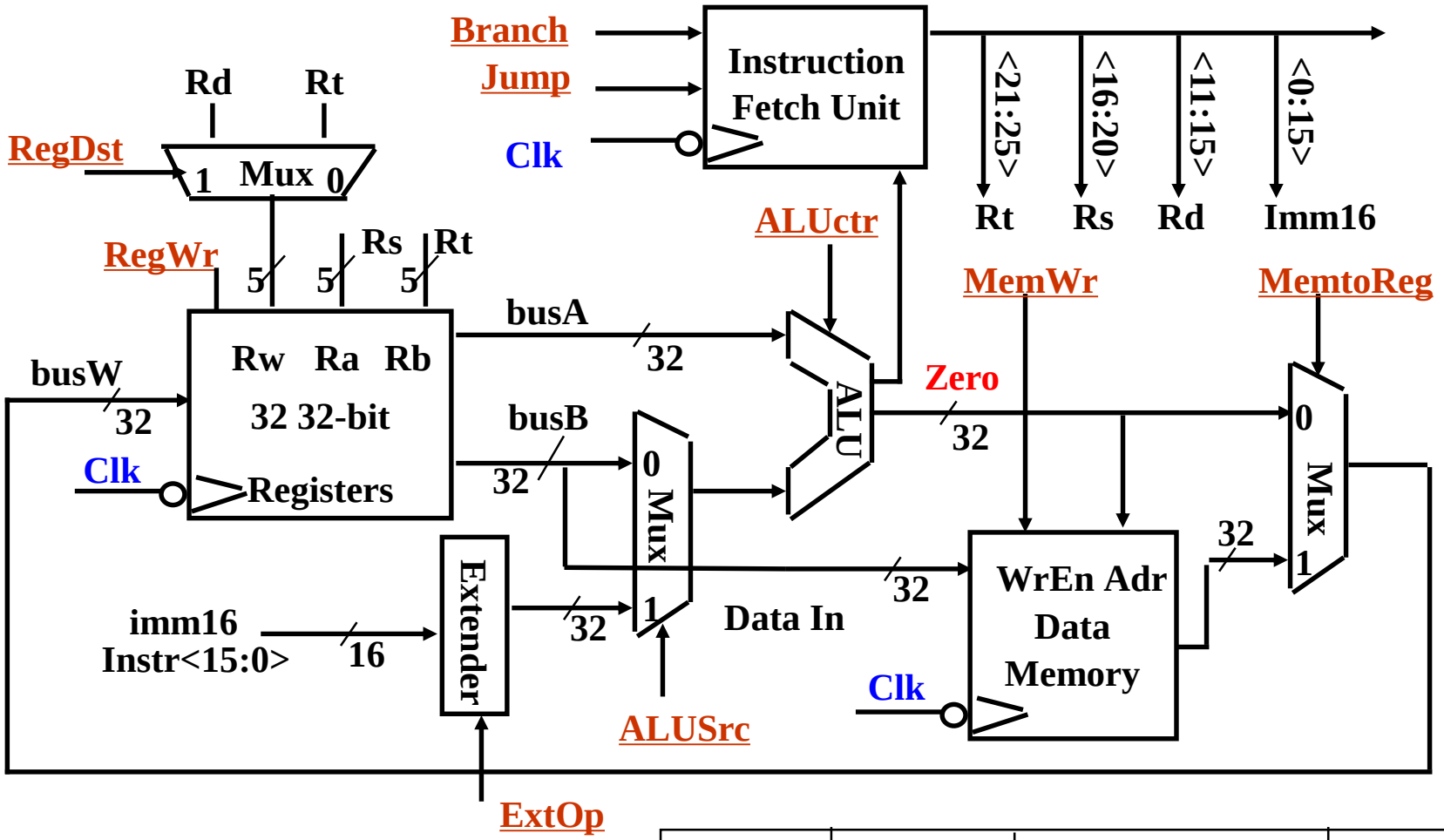
2. 微指令格式的设计

❑垂直型微指令

- **内涵：采用短格式，一条微指令只能控制一个微操作**
- **优点：微指令字长短，编码空间利用率高，格式与机器指令类似，故编写微程序容易**
- **缺点：微程序长，微指令的并行微操作能力有限，微指令译码比较复杂，速度慢**
- **垂直型微指令是面向算法的描述**

2. 微指令格式的设计

❑例：单周期数据通路对应的微操作码



控制字(即：微指令)的长度等于控制信号(微命令)的总位数。

采用不译法，则微操作码格式为：

RegWr	MemWr	……	ALUSrc	ALUctr	……

3. 下条微地址的确定方式

❑什么是微程序执行顺序的控制？

- ■ 在当前微指令执行完后，怎样控制产生下条微指令的地址

3. 下条微地址的确定方式

- 什么是微程序执行顺序的控制？
  - 在当前微指令执行完后，怎样控制产生下条微指令的地址
- 怎样控制微程序的执行顺序？
  - 微指令中明显或隐含地指定下条微指令在控存中的地址

3. 下条微地址的确定方式

- 什么是微程序执行顺序的控制？
  - 在当前微指令执行完后，怎样控制产生下条微指令的地址
- 怎样控制微程序的执行顺序？
  - 微指令中明显或隐含地指定下条微指令在控存中的地址
- 微指令地址的产生方法有两种
  - 顺序 - 转移(计数器)法：下条微指令地址隐含在微程序计数器μPC中
  - 断定(下址字段)法：当前微指令中显式指定下条微指令地址

3. 下条微地址的确定方式

❑当前微指令执行结束后，下条要执行的微指令有三种情况

① **取指微程序首址：**每条指令都要先执行“取指微程序”

② **某机器指令对应微程序的首条微指令：**当执行完取指微程序的最后一条微指令，需根据当前指令的译码结果确定执行哪条指令所对应的微程序，然后转移到对应微程序的首条微指令执行

③ 某个微程序执行过程中的一条微指令，又有三种情况：

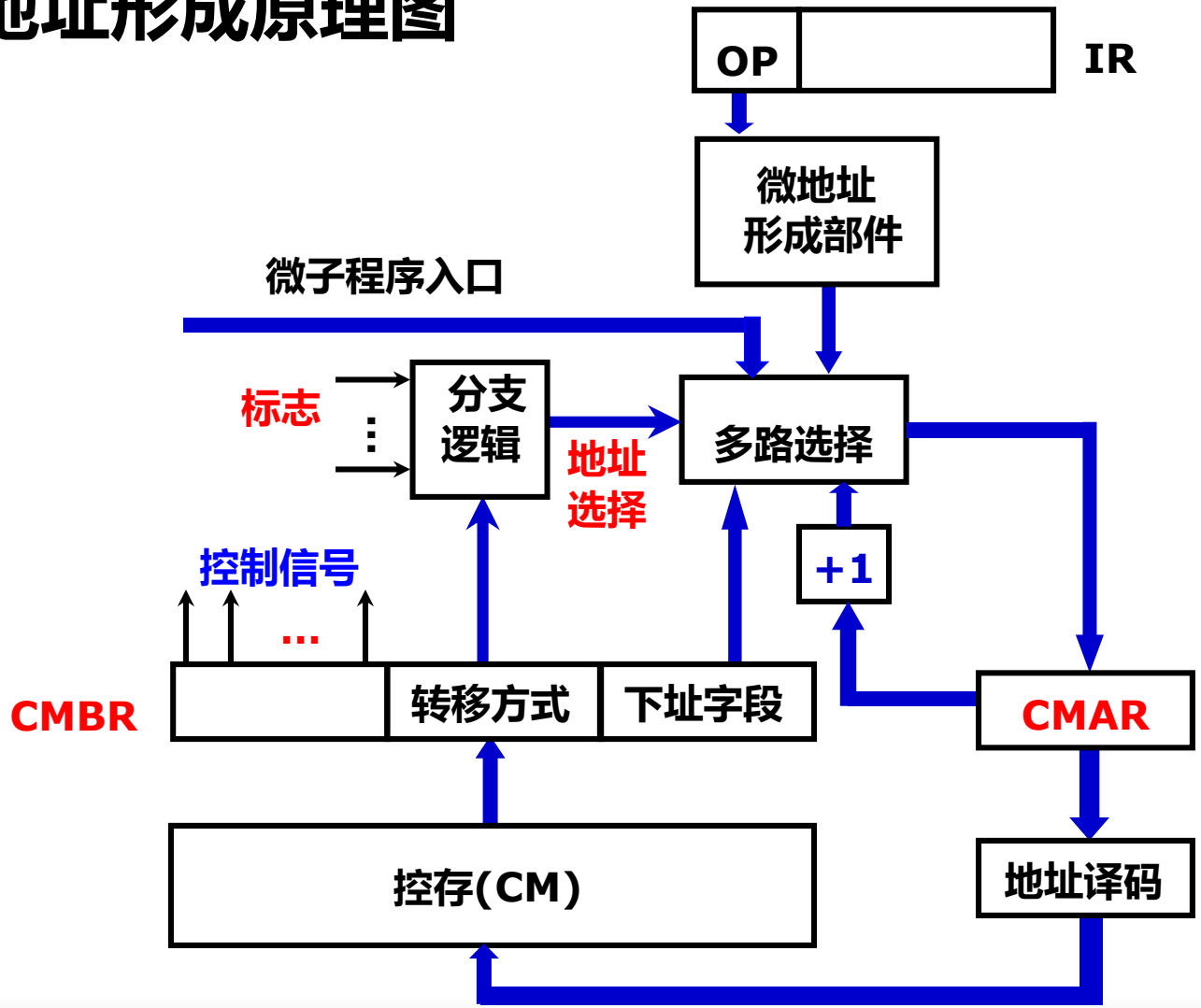
口顺序执行：按顺序取出下条微指令执行

口无条件执行：无条件转到另一处微指令执行

口分支执行：根据条件码或者指令操作码转移到不同微指令执行

3. 下条微地址的确定方式

❑下条微指令地址形成原理图



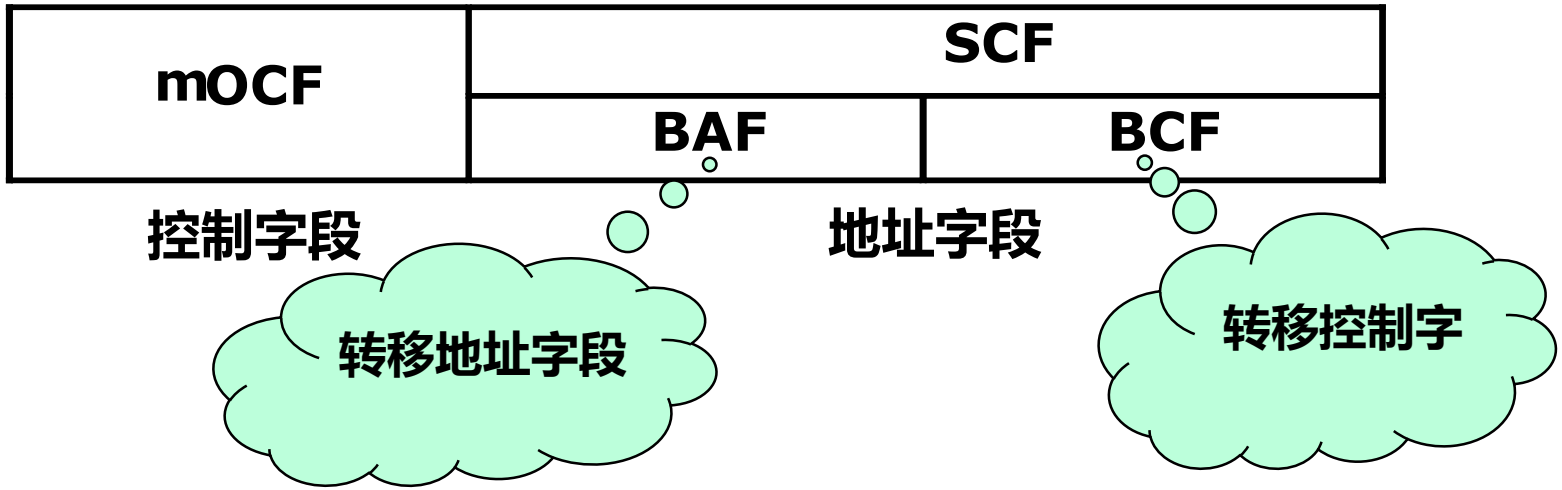
3. 下条微地址的确定方式

❑顺序-转移方式（基本思想）

- ■ 用类似程序计数器PC产生当前机器指令地址的方法
- ■ 设置一个微程序计数器μPC，指示当前微指令的地址
- ■ 顺序执行微指令时，下条微指令地址由μPC加上一个增量来产生
- ■ 遇到转移时，由微指令给出转移微地址

3. 下条微地址的确定方式

❑顺序-转移方式（微指令结构）



- BAF的位数决定了转移范围大小和灵活性
- BCF指出转移地址的来源
  - 由BAF确定的地址
  - 机器指令所对应的微程序的入口地址
  - 微子程序的返回地址

3. 下条微地址的确定方式

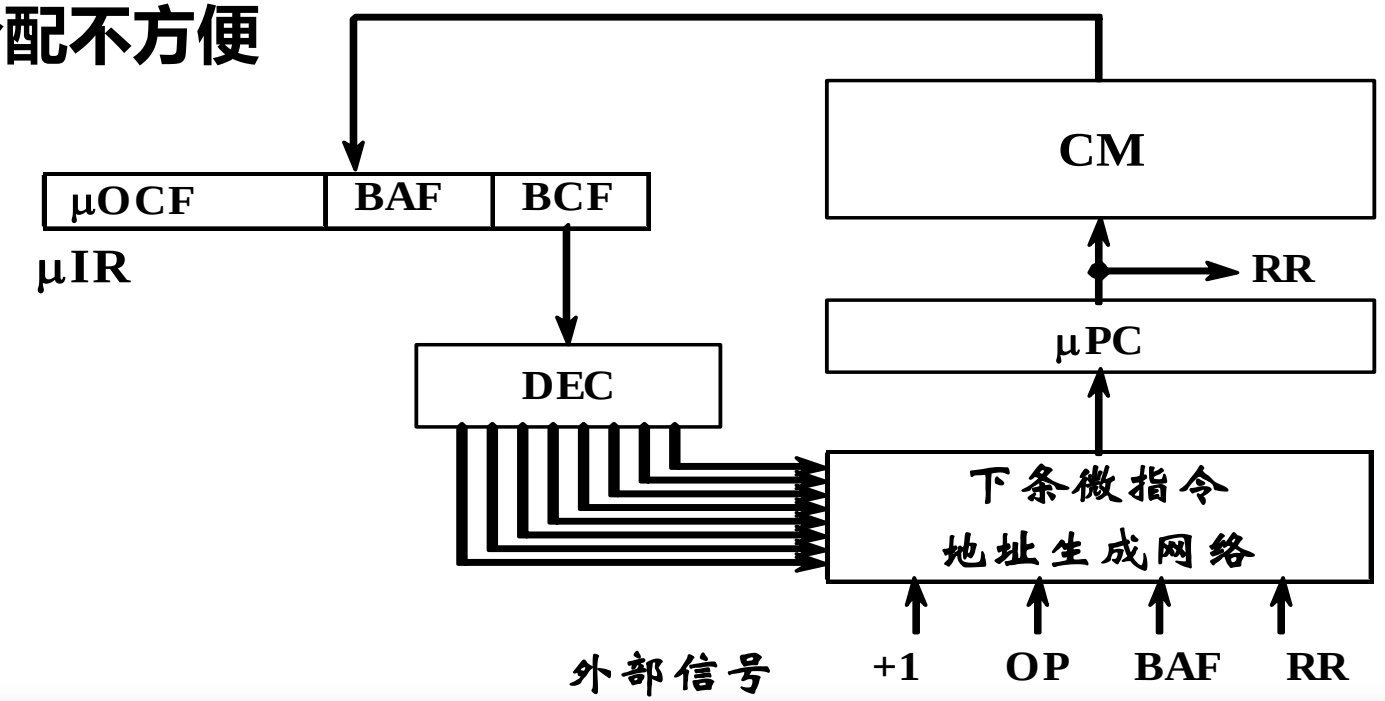
❑顺序-转移方式（实例分析）

■ 假设转移控制字段BCF为3位，用于实现顺序执行、初始转移、无条件转移、条件转移、测试循环、转微子程序、微子程序返回等7种情况，如下表所示：

BCF编码	顺序控制	测试条件	后继微地址
000	顺序执行	—	μPC + 1
001	初始转移	—	μPC ← OP
010	无条件转移	—	μPC和BAF的组合
011	条件转移	成立	μPC和BAF的组合
		不成立	μPC + 1
100	测试循环	成立	μPC + 1
		不成立	μPC和BAF的组合
101	转微子程序	—	μPC和BAF的组合
110	微子程序返回	—	μPC ← RR
111	（备用）		

3. 下条微地址的确定方式

- 顺序-转移方式（控制原理图）
  - 优点：微指令中SCF字段较短，下条微地址产生机构比较简单
- 缺点
  - 不利于解决两路以上的并行微程序转移，从而不利于提高微程序的执行速度
  - 微程序在CM中的物理分配不方便



3. 下条微地址的确定方式

- 断定方式（基本思想）
  - 下条微地址由微程序设计者直接指定，或者由微程序设计者指定的测试判别字段控制产生
  - 微地址码结构：非测试地址HF　　测试地址TF
- 微指令结构

μOCF	SCF	
	非测试字段 HF	测试控制字段 TCF

  - 转移的并行度
  - 测试地址的位数确定了转移的并行度：n位为2^n路转移

3. 下条微地址的确定方式

❑断定方式（基本思想）

- ■优点：能以较短的SCF配合实现多路并行转移，提高微程序执行效率和执行速度；微程序在CM中分配物理空间方便、灵活
- ■缺点：下条微地址码的生成机构比较复杂
- ■例：具有两个测试字段的微地址码产生过程