南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

第 4 章 指令系统与控制单元

主讲：张功萱

控制单元设计-微程序控制器

第4.7节

1. 什么是Wilkes模型?
2. 怎样理解微指令设计方法?
3. 微操作相容与互斥是什么?
4. CM怎样存储微程序?

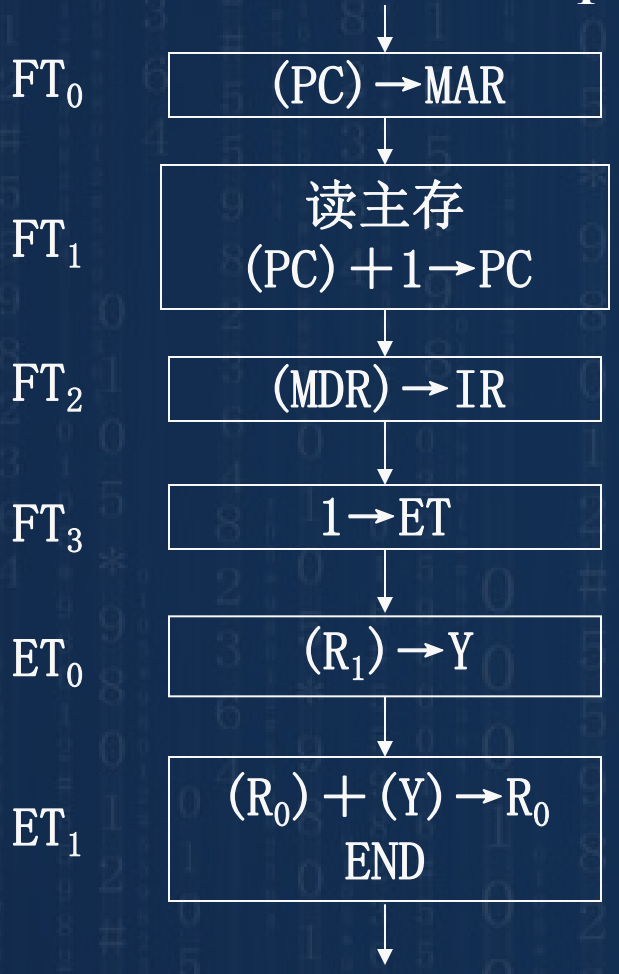
4.7.1 微程序控制器概述

- 硬联逻辑控制器的缺点
- ① 繁琐、杂乱，缺乏规律性，设计效率低，不利于检查调试。
- ② 不易修改和扩充，缺乏灵活性。
- 因为设计结果用印刷电路板(硬连逻辑)固定下来以后，就很难再修改与扩充。

微程序设计思想

- 微程序控制的基本概念
- 每一条指令都对应着自己的微操作序列，每一条指令的执行过程，都可以划分为若干基本的微操作。把各条指令的微操作序列，以二进制编码字（称为微指令）的形式编制成程序（称为微程序），并存放在一个存储器（称为控制存储器）中。执行指令时，通过读取并执行相应的微程序实现一条指令的功能，
- 微程序设计的实质
- 用程序设计的思想方法来组织操作控制逻辑，用规整的存储逻辑代替繁杂的组合逻辑。

- 例：模型机中指令 ADD R_1，R_0的指令执行过程及控制信号。



$PC \rightarrow BUS_1$，$S_3S_2S_1S_0M$，DM，CPMAR

MRQ，$R/\overline{W}=1$，$PC \rightarrow BUS_1$，$S_3S_2S_1S_0\overline{M}$，C_0，DM，CPPC

$MDR \rightarrow BUS_1$，$S_3S_2S_1S_0M$，DM，CPIR

1→ET, CPET

$R_1 \rightarrow BUS_1$，$S_3S_2S_1S_0M$，DM，CPY

$R_0 \rightarrow BUS_1$，$S_3\overline{S}_2\overline{S}_1S_0\overline{M}$，DM，$CPR_0, CPC_C, CPC_Z$

- 可以利用二进制编码描述操作系列。

3位	5位	1位	2位	2位	3位	1位	2位
BUSin	$S_3S_2S_1S_0M$	C_0	S	BUS_1out	BUS_2out	$R/\overline{W}$	$IO/\overline{M}$

BUSin	功能
000	无
001	$R_S \rightarrow BUS_1$
010	$R_D \rightarrow BUS_1$
011	$TEMP \rightarrow BUS_1$
100	$SP \rightarrow BUS_1$
101	$MDR \rightarrow BUS_1$
110	$IR(D) \rightarrow BUS_1$
111	$PC \rightarrow BUS_1$

$S_3S_2S_1S_0M$
按74181的规定

C_0	功能
0	$C_0=0$
1	$C_0=1$

S	功能
00	DM
01	SL
10	SR
11	EX

BUS_1out	功能
00	无
01	CPIR
10	CPMAR
11	CPC_Z、CPC_C

BUS_2out	功能
000	无
001	CPRS
010	CPRD
011	CPY
100	CPSP
101	CPMDR
110	CPTEMP
111	CPPC

$R/\overline{W}$	功能
0	写
1	读

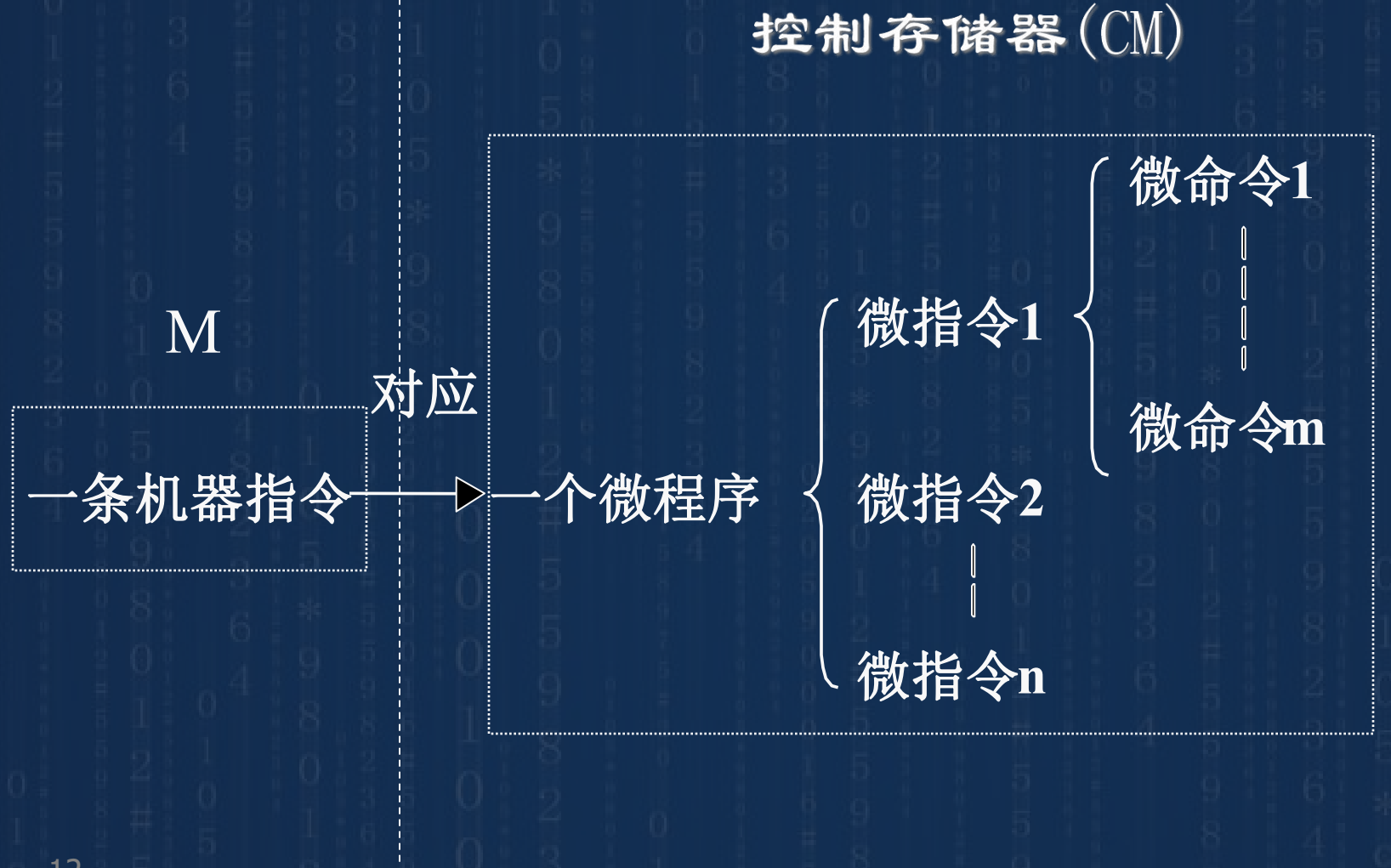
$IO/\overline{M}$	功能
00	无
01	MREQ＝1 访存
10	MREQ＝0 访I/O

- 微操作：PC→BUS_1，$S_3S_2S_1S_0M$，DM，CPMAR 对应的控制信号编码为：
- 111 11111 0 00 10 000 1 00
- 微操作：MRQ，R/W=1，PC→BUS_1，$S_3S_2S_1S_0\overline{M}$，C_0，DM，CPPC 对应的控制信号编码为：
- 111 11110 1 00 00 111 1 01
- 微操作：MDR→BUS_1，$S_3S_2S_1S_0M$，DM，CPIR 对应的控制信号编码为：
- 101 11111 0 00 01 000 × 00

- 将指令要执行的控制信号用二进制编码字编码后依次存入存储器。执行指令时，顺序读出编码字，产生控制命令，即可控制有关部件完成规定的操作。
- 这种控制思想称为微程序控制。
- 微程序设计将软件设计技术应用到硬件设计中，其特点是：应用灵活，控制规整，便于计算机设计自动化，并易于修改和扩充。
- CISC系列的小、微型机多采用微程序控制方法设计控制器。

- 微程序设计的实质
- 用程序设计的思想方法来组织操作控制逻辑，用规整的存储逻辑代替繁杂的组合逻辑。
- 因为微程序控制是将微操作控制信号以编码字(即微指令)的形式存放在控制存储器中。执行指令时，通过依次读取一条条微指令，产生一组组操作控制信号，控制有关功能部件完成一组组微操作。因此，又称为存储逻辑。

- 在微程序控制机器中，涉及到两个层次。
- 传统机器级 —— 机器指令
- 是使用机器语言的程序员所看到的。用机器指令编制工作程序，完成某一处理任务。CPU执行的程序存放在主存储器中。
- 微程序 —— 微指令
- 是硬件设计者所看到的。用微指令编制微程序，用以完成一条机器指令的功能。微程序存放在控制存储器中。微指令用于产生一组控制命令，称为微命令，控制完成一组微操作。


控制存储器(CM)
M
对应
一条机器指令
一个微程序
微指令1
微指令2
微指令n
微命令1
微命令m

1. 微程序控制器的基本概念

- 1）微命令；直接作用于部件或控制门电路的控制命令。是构成控制信号序列的最小单位。
- 例如模型机中的PC→BUS_1，R_0→BUS_1、CPIR、R/W等控制信号都称为微命令。
- 2）微操作；由微命令控制实现的最基本的操作。
- 微操作的定义可大可小
- 例：
- 微操作 (PC)→MAR：是在一组微命令 PC→BUS_1、$S_3S_2S_1S_0$M、DM、CPMAR的控制下实现的。
- 微操作打开PC与BUS_1之间的控制门：是在一个微命令PC→BUS_1的控制下实现的。

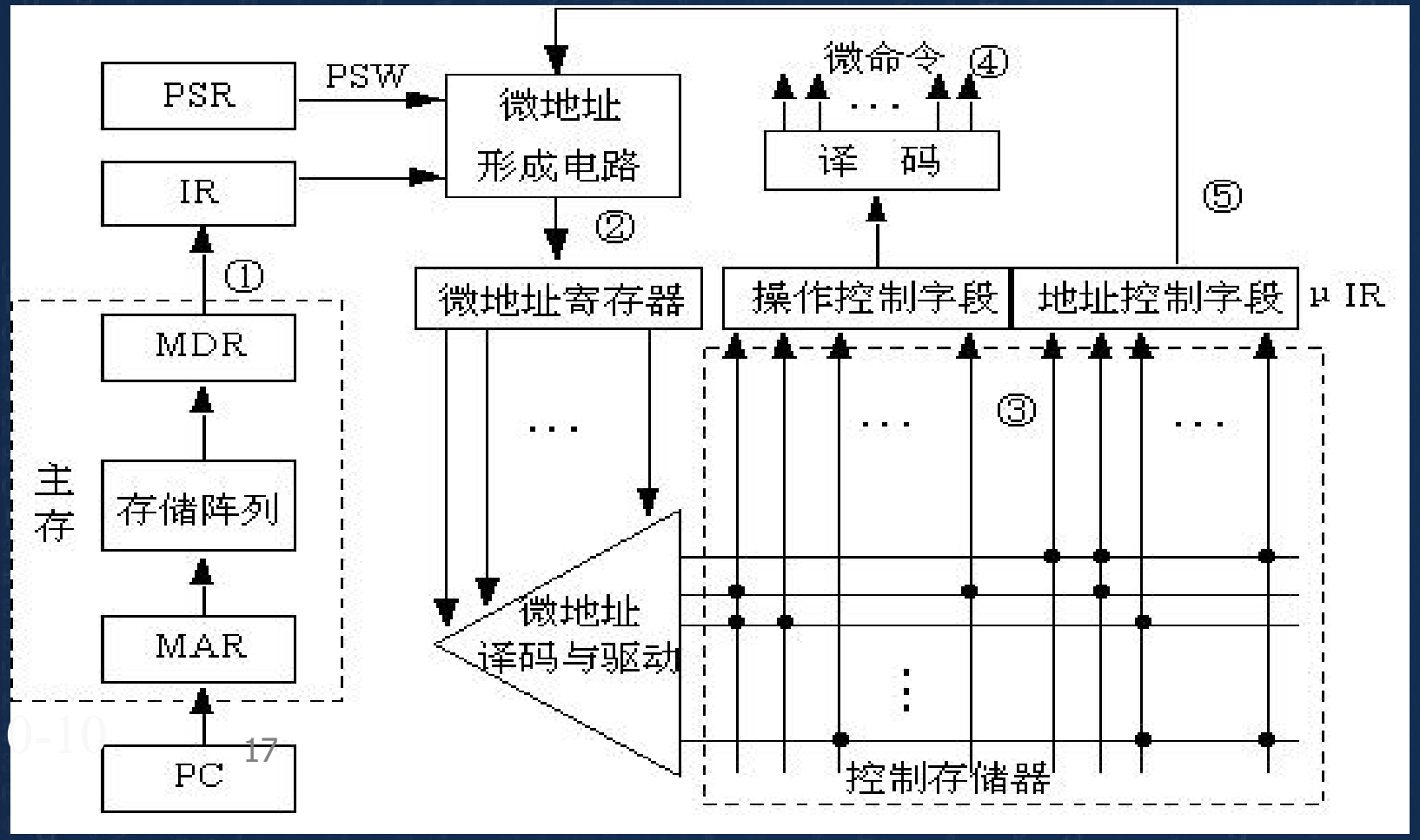
- 3）微指令：用以产生一组微命令，控制完成一组微操作的二进制编码字称为微指令。
- 微指令存放在控制存储器中。一条微指令通常控制实现数据通路中的一步操作过程。
- 4）微程序：一系列微指令的有序集合称为微程序。
- 在微程序控制的机器中，一条机器指令对应着一段由若干条有序的微指令构成的微程序；通过解释执行一段微程序，可以实现相应的一条机器指令的功能。

- 5）微周期：从控制存储器中读取一条微指令并执行相应的微操作所需的时间称为微周期。
- 在微程序控制的机器中，微周期是它的主要时序信号。通常一个时钟周期为一个微周期。
- 微指令的执行采用同步方式，每个微周期的时间相同。在模型机中，一个微周期相当于一个节拍。
- 6）控制存储器（CM）：存放微程序的存储器。简称控存，也称为微程序存储器。
- 一般计算讥指令系统是固定的，因而实现指令系统的微程序也是固定的，所以控制存储器通常用只读存储器实现。

- 相容
  - 指在同时或同一个CPU时钟单位（周期）内并行执行的微操作。
- 不相容（互斥）
  - 指不能同时或不能在同一个CPU时钟周期内并行执行的微操作。

2. 控制器组成与微程序执行

- 1951年，Wilkes提出了微程序控制的基本思想和特点，并于1965年，IBM 360系列机成功采用了微程序技术。



微程序控制器--Wilkes 模型

1）微程序控制器的组成结构

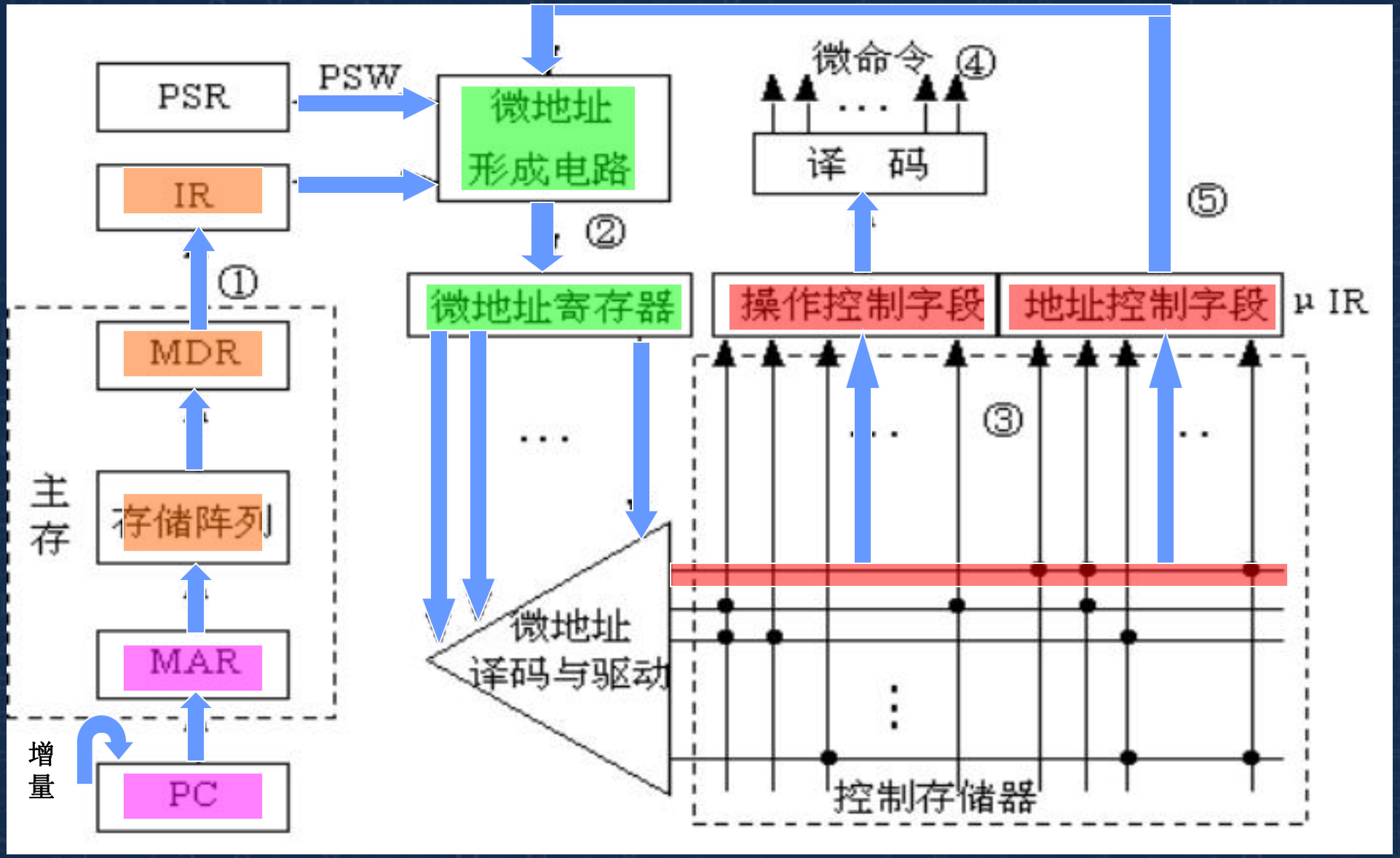
- （1）控制存储器CM：控存的每个单元存放一条微指令代码。
- 图中每条横线表示一个单元，其中每个交叉点表示微指令的一位，有“•”表示该位为1，无“•”表示该位为0。
- （2）微指令寄存器μIR：存放从控存中读取的微指令。
- 微指令通常分为两大字段：
- 操作控制字段：经译码或直接产生一组微命令，控制有关部件完成微指令所规定的微操作。
- 地址控制字段：指示下条微指令地址的形成方式或直接给出下条微指令地址。

- （3）微地址形成电路：用于产生起始微地址和后继微地址，保证微程序的连续执行。
- （4）微地址寄存器μMAR：接受微地址形成电路送来的地址，为读取微指令准备好控存的地址。
- （5）译码与驱动电路：对μMAR中的微地址进行译码，找到被访问的控存单元并驱动其进行读取操作，读取微指令并存放于微指令寄存器中。

- 在微程序控制器组成框图中，PC、IR、PSR等的功能与组合逻辑控制器中的功能一致。

- 微程序控制器与硬联逻辑控制器的主要区别：
- 微操作控制信号形成部件的不同。
- ① 组合逻辑控制器中的复杂硬联逻辑网络在微程序控制器被规整的存储逻辑所替代。
- ② 存储逻辑中包含了存放微程序的控制存储器。

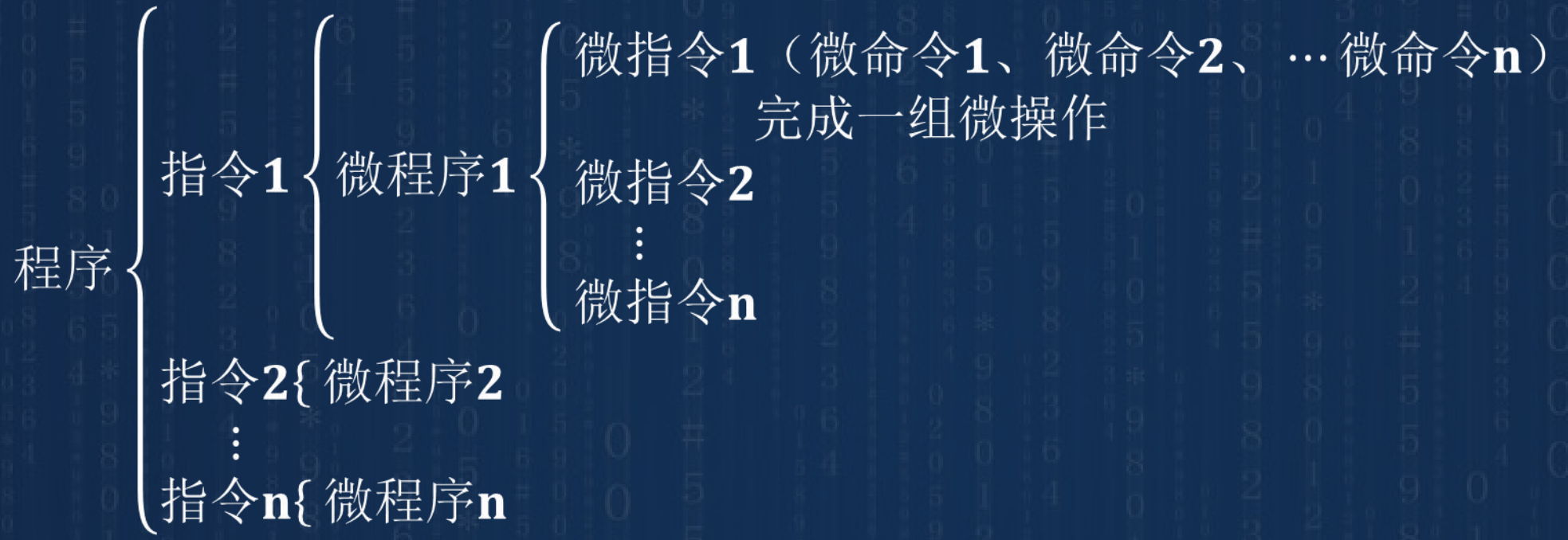
2）微程序的执行过程


PSR
PSW
IR
微地址
形成电路
微命令
④
译 码
⑤
①
②
微地址寄存器
操作控制字段
地址控制字段
μ IR
MDR
主
存
存储阵列
③
MAR
微地址
译码与驱动
增
量
PC
控制存储器

- ① 启动取指令微程序
- (PC)→MAR，读出机器指令→IR，PC增量，为下条指令准备地址。
- ② 根据IR中的指令码，通过微地址形成电路产生该指令的微程序的起始微地址，并送入μMAR中。
- ③ μMAR中的微地址经译码、驱动，从被选的控存单元中取出一条微指令并送入μIR。
- ④ μIR中的微指令的操作控制字段经译码或直接产生一组微命令并送往有关的功能部件，控制其完成所规定的微操作。

- ⑤ μIR中微指令的地址控制字段及有关状态条件送往微地址形成电路，产生下条微指令的地址，再去读取并执行下条微指令。如此循环，直到一条机器指令的微程序全部执行完毕。
- ⑥一条指令的微程序执行结束，再启动取指令微指令或微程序，读取下条机器指令。根据该指令码形成起始微地址，又转入执行它的一段微程序。

程序、指令、微程序、微指令、微命令、微操作的关系



- 微程序定义了计算机的指令系统，只要改变控制存储器的内容就能改变机器的指令系统，为计算机设计者及用户提供了极大的灵活性。
- 由于执行一条机器指令可能需要多次访问控制存储器，因此必须使用速度快的**ROM**芯片作为控存，以使计算机能够保持较高的性能。

微程序控制器设计的关键

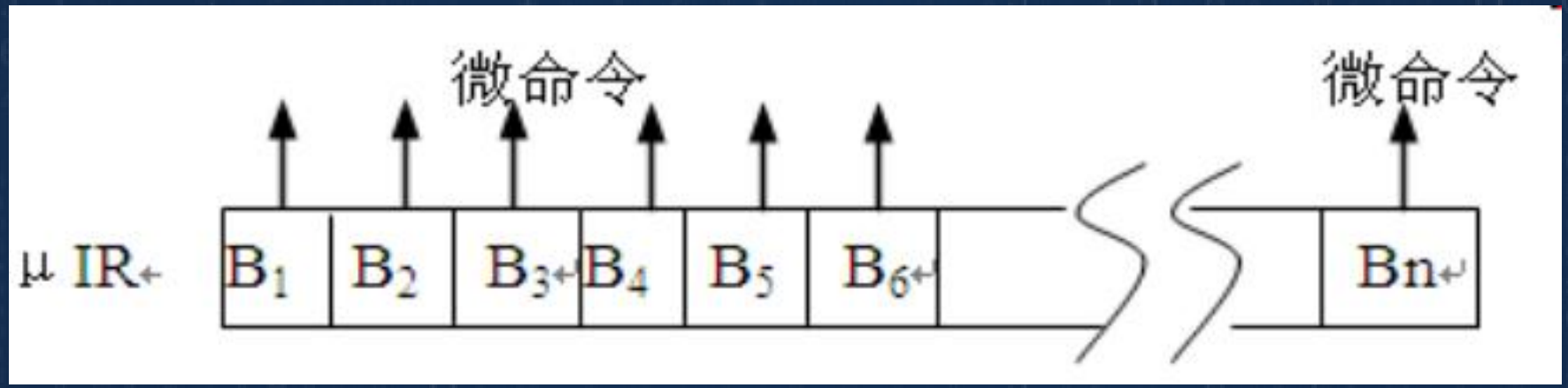
- ① 微指令的结构格式及编码方法
- 如何用一个二进制编码字表示各个微命令。
- ② 微程序顺序控制的方法
- 即如何解决微程序中微指令之间的衔接和不同机器指令所对应的微程序之间的逻辑衔接，保证微程序的连续执行。
- ③ 微指令执行方式
- 解决如何提高微指令的执行速度的问题。

4.7.2 微指令的编译方法

- 微指令的编译方法：指如何对微指令的操作控制字段进行编码来表示各个微命令，以及如何把编码译成相应的微命令。即微指令的格式应如何设计，应怎样进行微指令的译码。
- 设计微指令的结构格式时，主要考虑的问题：
- ① 如何有利于缩短微指令字长
- ② 如何有利于缩短微程序，减少所需的控存空间
- ③ 如何有利于提高微程序执行速度

常用的微指令编译方法

- 1．直接控制法（不译法）
- 微指令操作控制字段的每一位都直接表示一个微命令，该位为“1”，表示执行这个微命令，为“0”表示不执行该微命令。由于这种方法不需译码，所以也称不译法。



- 例：在模型机中，若只考虑输入信号，则有35个微命令，因此微指令的操作控制字段应长35位。
- 设μIR的第22位对应于C_0，则译码时有$C_0 = \mu IR_{22}$。
- C_0

- 直接控制法的优点：结构简单，并行性强，操作速度快。
- 直接控制法的缺点：微指令字太长，信息效率低。
- 因为在直接控制法中，有N个微命令，操作控制字段就需N位。在实际机器中，微命令数达几百个，使微指令字长达到难以接受的地步。同时在几百个微命令中有很多是互斥的，不允许同时出现的（如$R_0 \rightarrow BUS_1$，$R_1 \rightarrow BUS_1$），将它们安排在同一条微指令内，只会使信息效率降低。

- 在实际机器中，往往将直接控制法与其它方法混合使用，仅部分位采用直接控制法。

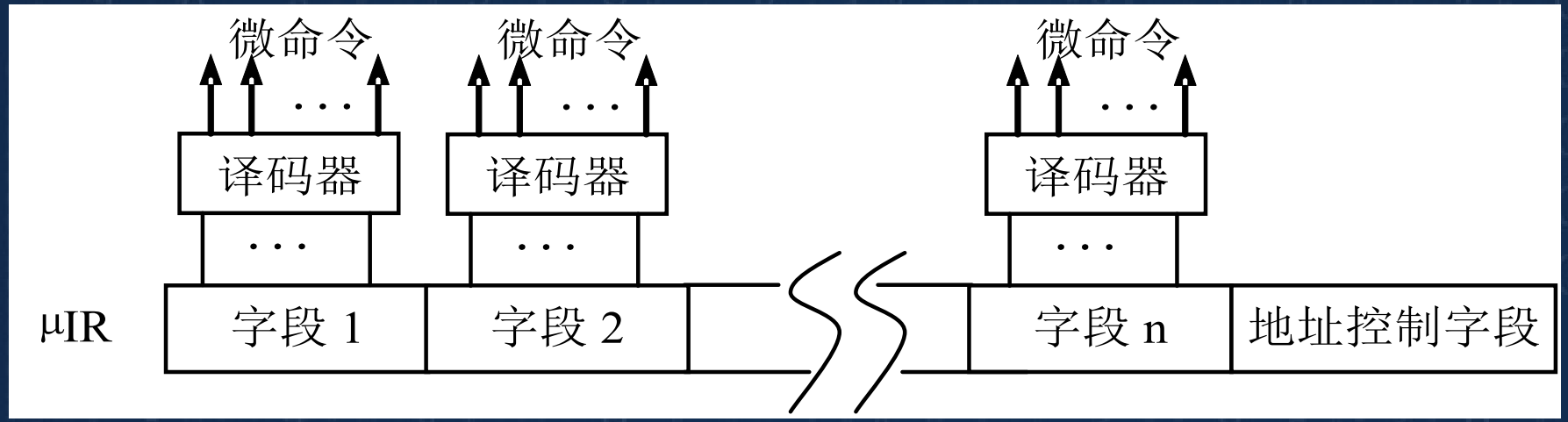
2. 最短编码法

- 将所有的微命令进行统一编码，每条微指令只定义一个微命令。若微命令总数为N，则最短编码法中操作控制字段的长度L，应满足下列关系：
- $L \geq \log_2 N$

- 最短编码法所得的微指令字长最短，但要通过微命令译码器译码才能得到所需的微命令。微命令越多，译码器就越复杂。同时这种方法在某一时间只能产生一个微命令，不能充分利用机器硬件所具有的并行性，使微程序很长，所以这种方法很少独立使用。

3. 字段直接编码法

- 将微指令操作控制字段划分为若干个子字段，每个子字段的所有微命令进行统一编码。因此在这种方法中，不同的子字段的不同编码，表示不同的微命令。

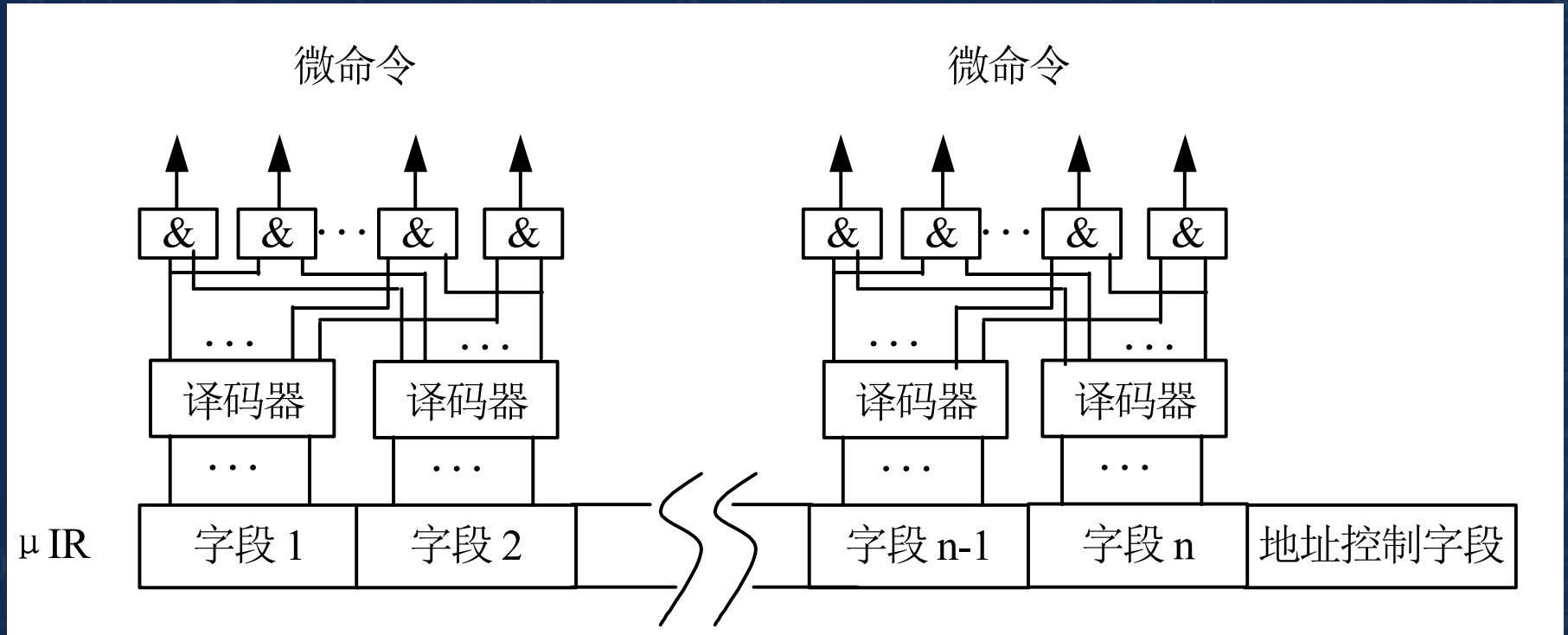


子字段的划分原则

- ① 把互斥的微命令（即不允许同时出现的微命令）划分在同一字段内，相容的微命令（即允许同时出现的微命令）划分在不同字段内。
- 例：各$R_i \rightarrow BUS_1$是互斥的，可以划分在同一字段；$R_S \rightarrow BUS_1$和CPR_S是相容的，应划分在不同字段。
- ② 字段的划分应与数据通路结构相适应。
- ③ 一般每个子字段应留出一个状态，表示本字段不发任何微命令。
- ④ 每个子字段所定义的微命令数不宜太多，否则将使微命令译码复杂。

4. 字段间接编码法

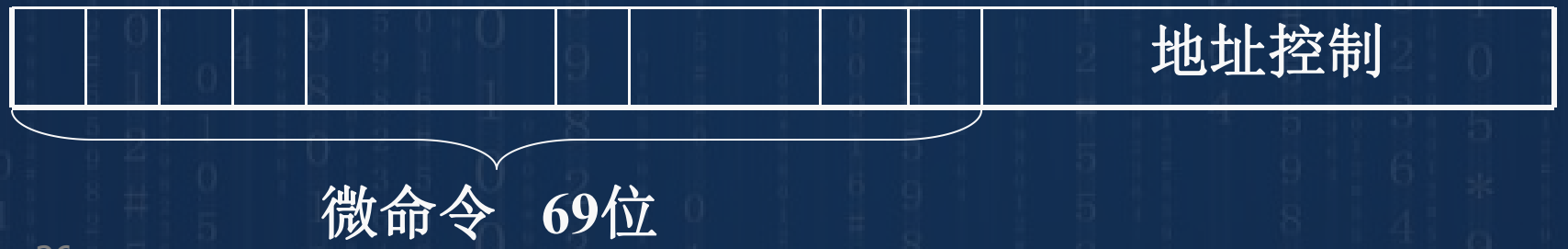
- 指一个字段的某一编码的意义由另一字段的编码来定义。也就是一个字段的编码不能直接独立地定义微命令，它必须与其它字段的编码联合定义。



5. 常数源字段的设置

- 在微指令字中，通常还设置一个常数源字段，如同指令字中的立即数一样，用来提供某些常数，如给计数器置初值，为某些数据提供修改量，配合形成微程序转移微地址等。

- 例：某机的微指令格式中共有8个控制字段，分别可以激活5、8、3、16、1、7、25、4种微命令。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，并说明两种方式的控制字段各需多少位。
  - 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微命令的个数，即：
  - 5＋8＋3＋16＋1＋7＋25＋4＝69



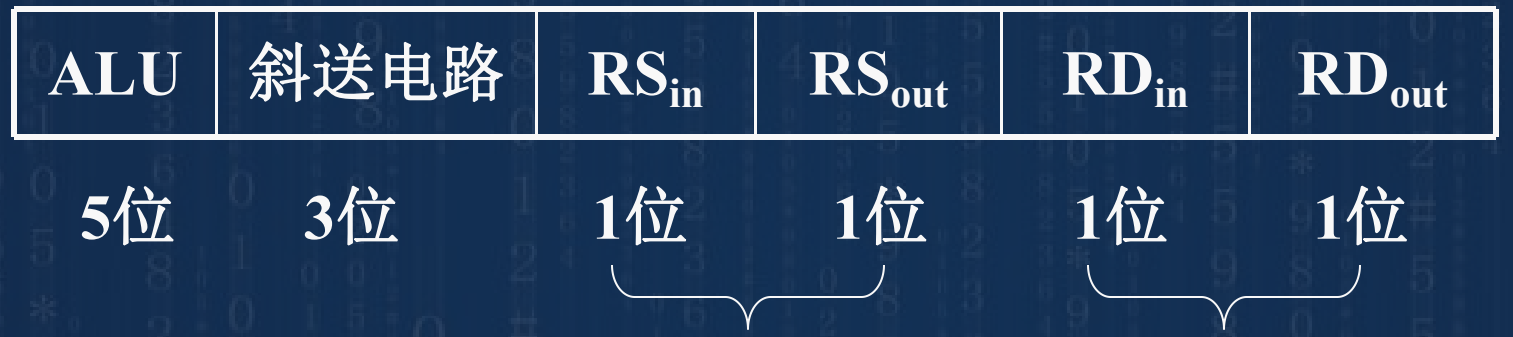
- （2）采用字段直接编码方式，共需8个控制字段，每个字段分别需激活5、8、3、16、1、7、25、4种微命令，共各需3、4、2、5、1、3、5、3位。微指令的操作控制字段的总位数为：
- 3＋4＋2＋5＋1＋3＋5＋3＝26

微命令　26位

- 例： 某计算机的CPU内部有一个支持16种算术运算和16种逻辑运算的ALU，一个具有8种操作的斜送电路，16个寄存器。若所有操作采用微程序设计，请采用字段直接编码方式给出该机的微指令格式（不考虑后继地址部分）。
- 控制ALU 32种功能需要 5位；
- 控制斜送电路8功能需要 3位，
- 16个寄存器，每个寄存器有接收、发送控制信号以及不操作信号，每个寄存器可以是源操作数，也可以是目的操作数，源和目的控制分开出现。寄存器编号由机器指令给定。

- 微指令可以分成下列字段：
- ALU＋斜送电路＋RS_{in}＋RS_{out}＋RD_{in}＋RD_{out}
- 微指令格式为：

寄存器编码由指令给出



4.7.3 微程序的控制设计

- 在微程序控制的计算机中，机器指令是通过微程序解释执行的。每一条指令都对应一段微程序，不同指令的微程序存放在控制存储器的不同存储区域内。
- 微程序顺序控制（或地址控制）需解决的问题是微程序应如何存放和执行，即需考虑初始微地址和后继微地址的各种形成方法。

- 微程序的初始微地址（微程序的入口地址）：指令所对应微程序的第一条微指令所在控制存储器单元的地址。
- 现行微指令：执行微程序过程中，当前正在执行的微指令。
- 现行微地址：现行微指令所在控制存储器单元的地址。
- 后继微指令：现行微指令执行完毕后，下一条要执行的微指令。
- 后继微地址：后继微指令所在控存单元的地址。

1. 初始微地址的形成

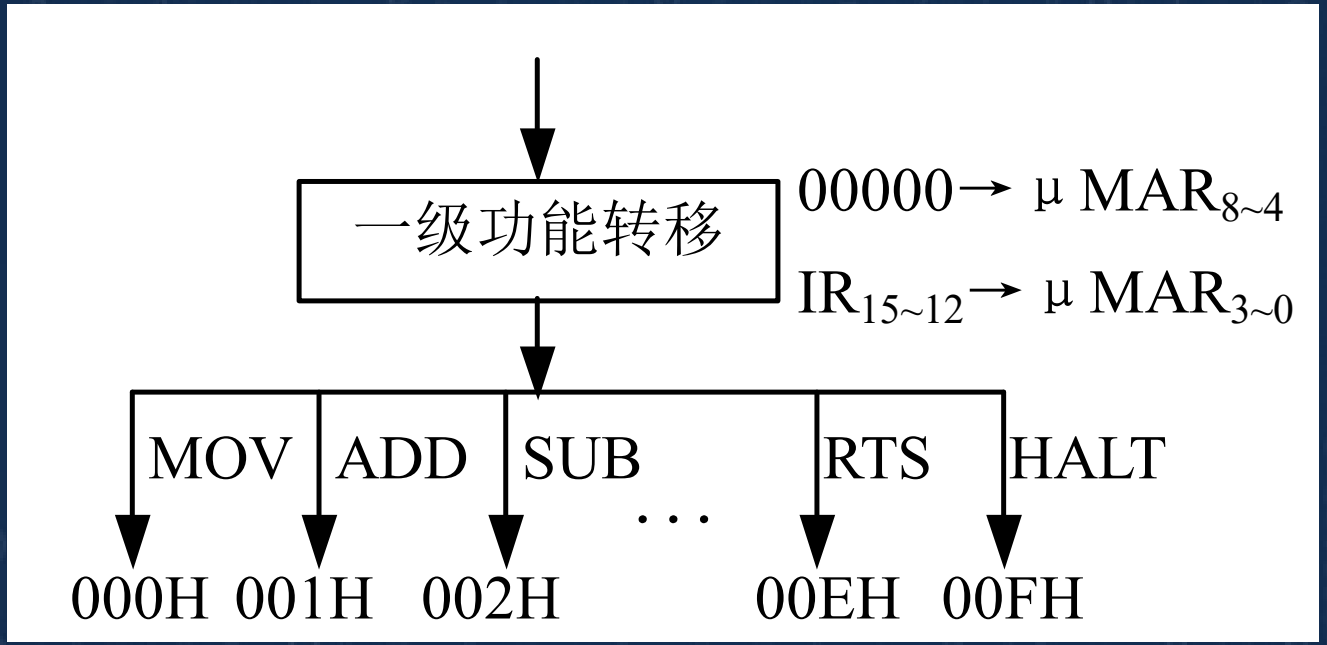
- 由于每一条机器指令的执行，都必须首先进行取指令操作，所以要有“取指令”微程序，控制从主存中取出一条机器指令。
- “取指令”微程序（通常由一条或几条微指令组成）是公用的，一般安排在从0号控存单元或其它特定的控存单元开始。
- 经过“取指令”后，机器指令从主存取到IR，然后根据机器指令操作码转换出该指令所对应的微程序入口地址，形成初始微地址。

初始微地址的几种形成方式

- （1）一级功能转移
- 根据指令操作码，直接转移到相应微程序的入口，即指令操作码直接参与形成微程序的入口地址。

- 当指令操作码的位置与位数均固定时，可直接用操作码作为微地址的低位。
- 一级功能转移适用于指令操作码的位置和长度比较规整的和固定的情况。

- 例：模型机有16条指令，操作码对应IR的15～12位，令微地址为00…0 OP，OP为指令操作码。当取出指令后，直接由$IR_{15\sim12}$。作为微地址的低4位。

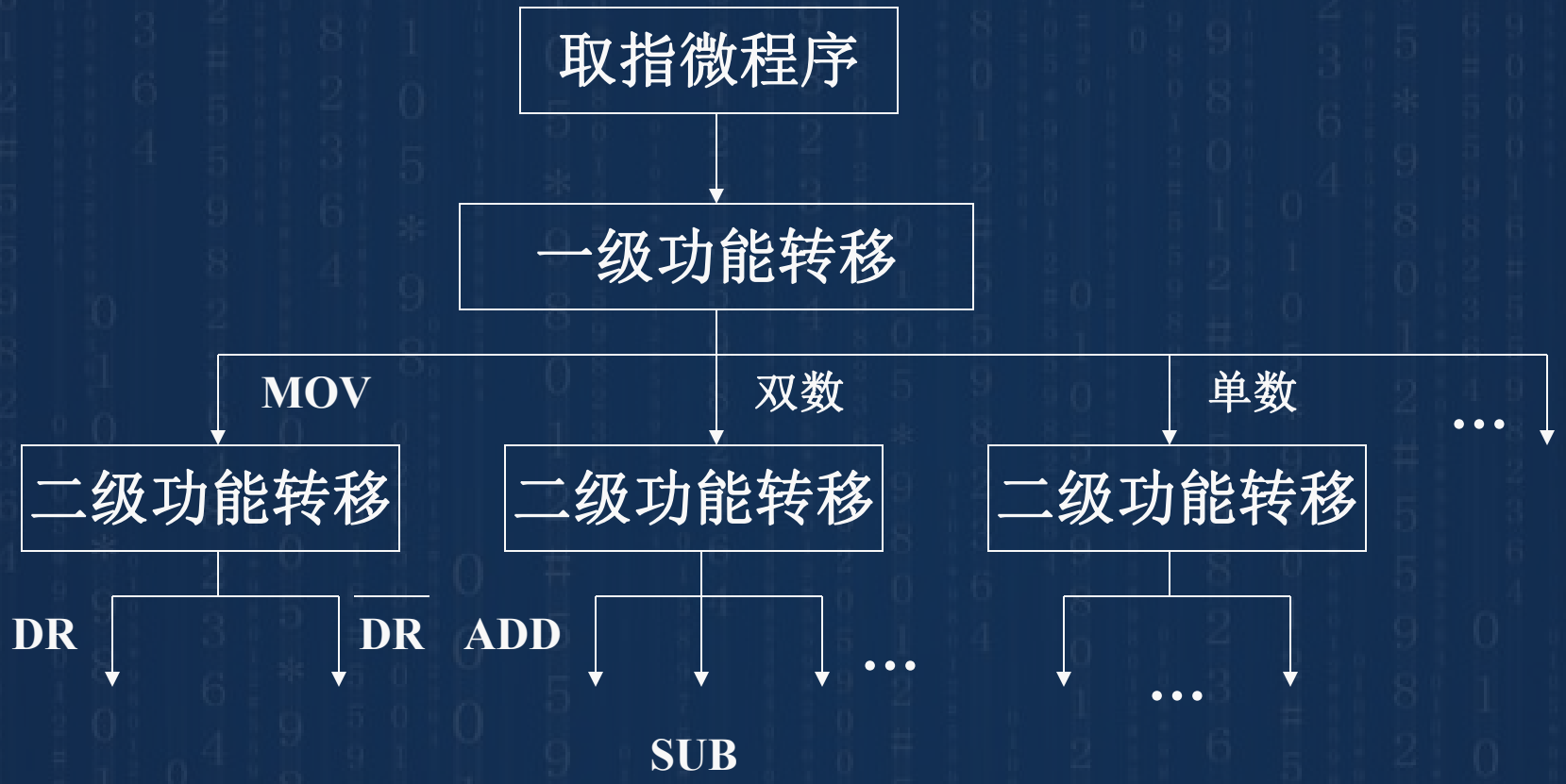


(2) 二级功能转移

- 先按指令类型标志转移，以区分出是哪一类指令。规定每类指令中位置和位数是固定的，在第二级按操作码区分出具体是哪条指令，转移到相应微程序入口。

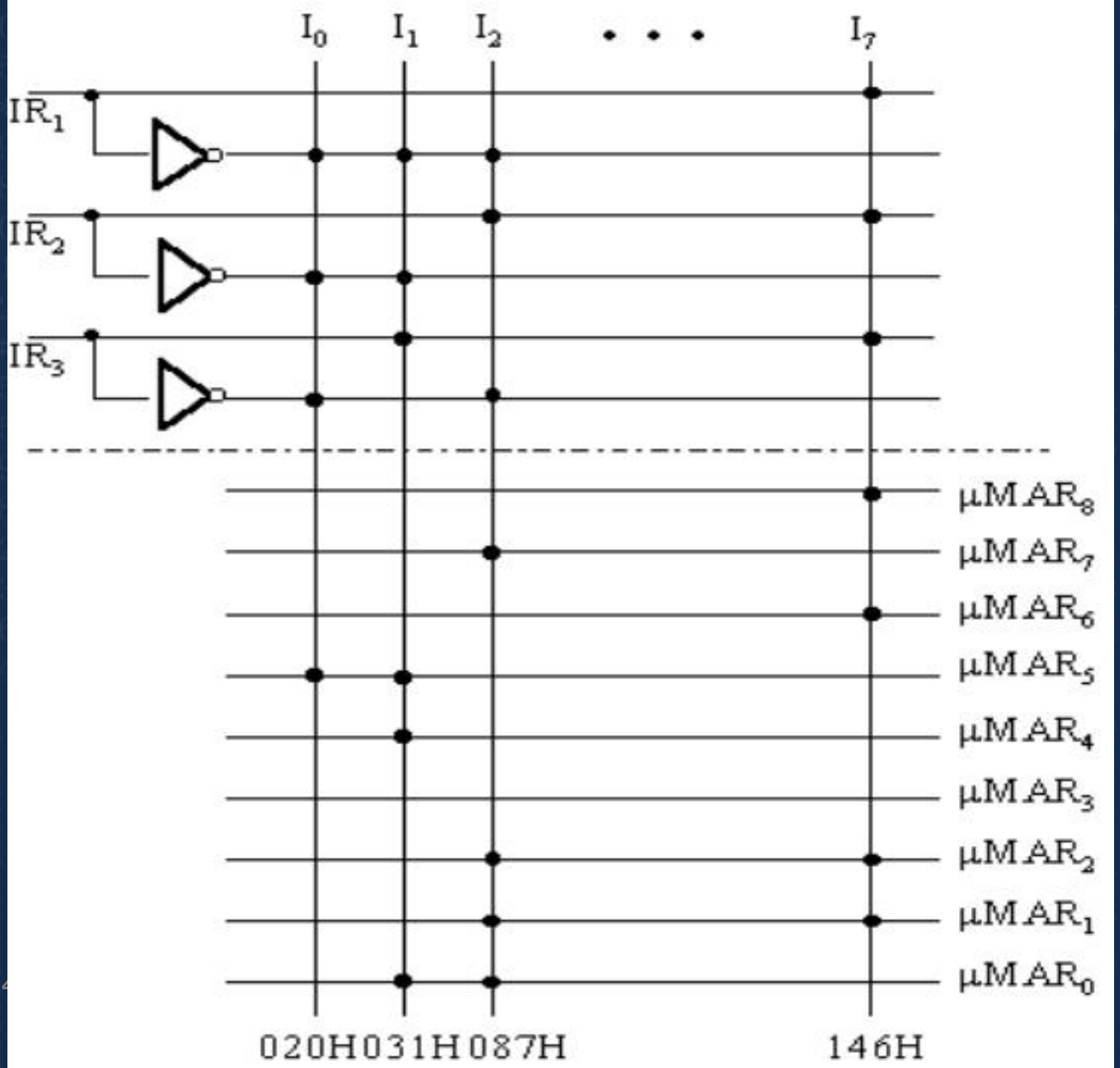
- 二级功能转移适合于机器指令的操作码的位数和位置不固定的情况。

二级功能转移



(3) 用PLA电路实现功能转移

- 利用PLA电路实现功能转移：将指令操作码作为可编程逻辑阵列PLA的输入，PLA的输出就是相应微程序入口地址。
- 这种方法对于变长度、变位置的操作码尤为有效，而且转移速度较快。
- 例：设有I_0、I_1、I_2、…、I_7共8条指令，其操作码分别为000、001、…、111，对应的微程序入口地址分别为020H、031H、…、146H。可用PLA实现微程序入口地址的寻址。


I0
I1
I2
I7
IR1
IR2
IR3
μMAR8
μMAR7
μMAR6
μMAR5
μMAR4
μMAR3
μMAR2
μMAR1
μMAR0
020H
031H
087H
146H

2. 后继微地址的形成

- 找到初始微地址后，开始执行相应的微程序。每条微指令执行完毕，都要根据要求形成后继微地址，以保证微程序的正常执行。
- 后继微地址的形成方法对微程序编制的灵活性影响很大。
- 后继微地址的形成方法主要有两种基本类型：增量方式和断定方式。

（1）增量方式

- 微地址的控制方式与程序地址控制方式相似，也有顺序执行、转移、转子之分。
- 增量方式：当微程序按地址递增顺序一条条地执行微指令时，后继微地址是现行微地址加上一个增量（通常为1）。
- (μPC)增量→μPC
- μPC是微程序计数器，微程序控制器中为节省设备，也可将μMAR做成具有计数功能的寄存器，与μPC合为一个寄存器，即：(μMAR)增量→μMAR

- 在增量方式中，当微程序转移或调用微子程序时，要解决转移微地址的形成问题。
- 转移微地址的形成方法
- ① 由微地址形成电路产生转移微地址
- 由微地址形成电路根据执行的微指令和当前状态形成转移微地址。
- 这种方法的电路比较复杂。

- ② 由微指令的下地址控制字段设置转移微地址。
- 通常把微指令的地址控制字段分为两个部分：
- 转移地址字段BAF：提供转移地址。
- 转移控制字段BCF：规定地址的形成方式。

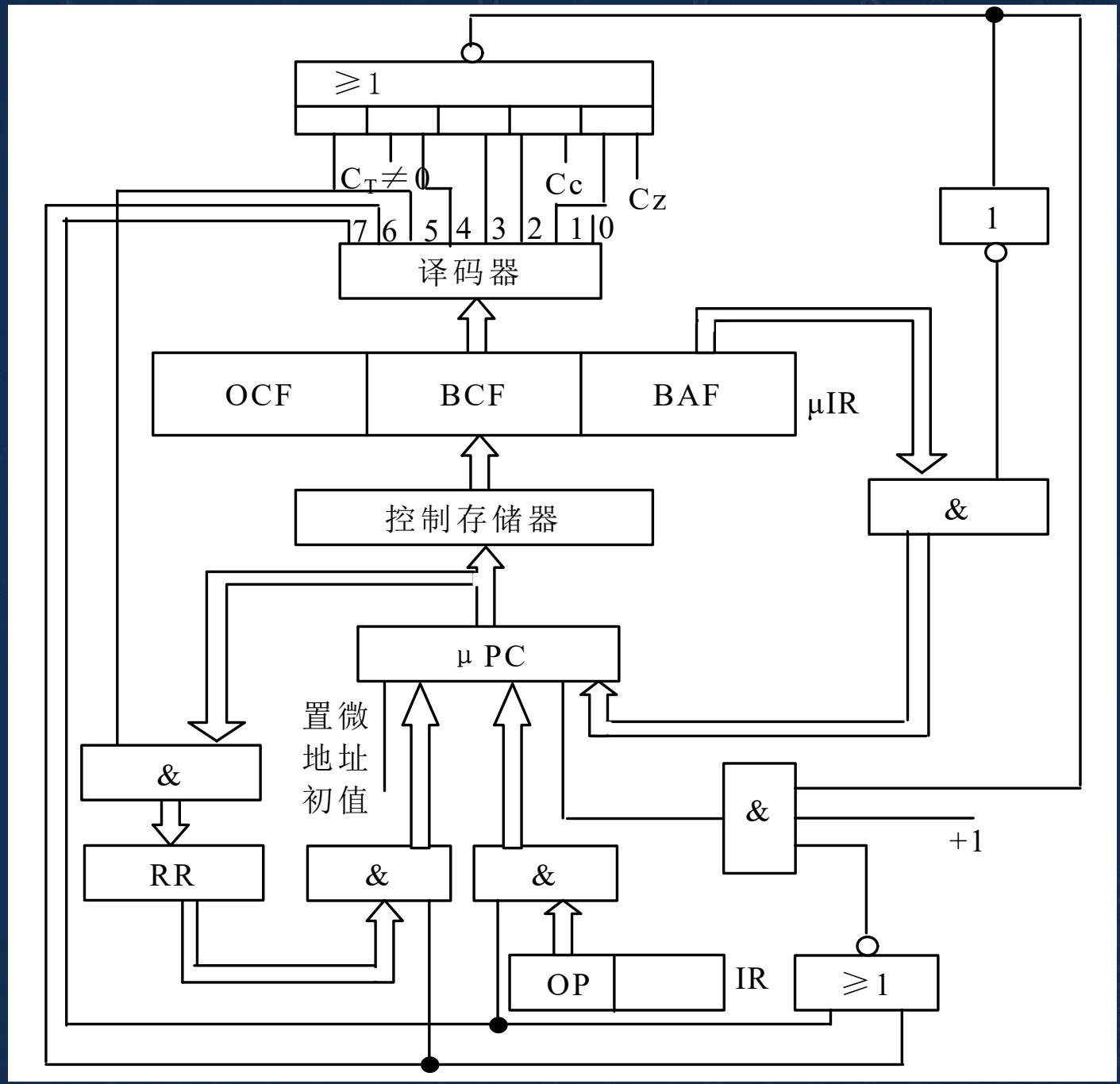
操作控制字段（OCF）	BCF	BAF

- 微指令执行完毕，根据BCF和BAF中的内容，控制是否转移和转移到何处。
- 若转移，BAF→μPC，否则，按μPC的现行内容顺序执行。

微地址形成方式举例

BCF		转移控制方式	硬件条件	后继微地址及有关操作
编码	二进制			
0	000	顺序执行		μPC +1→μPC
1	001	结果为 0 转移	Cz=0	μPC +1→μPC
			Cz=1	BAF→μPC
2	010	有进位转移	Cc=0	μPC +1→μPC
			Cc=1	BAF→μPC
3	011	无条件转移		BAF→μPC
4	100	循环测试	C_T=0	μPC +1→μPC
			$C_T \neq 0$	BAF→μPC
5	101	转微子程序		μPC +1→RR, BAF→μPC
6	110	返回		RR→μPC
7	111	操作码形成微地址		由操作码形成

微地址控制方式的原理框图



- C_Z：结果为“0”标志　C_C：进位标志
- C_T：循环计数器　　RR：返回地址寄存器
- 当执行转微子程序的转子微指令时，把现行微指令的下一微地址（μPC＋1）送入返回地址寄存器RR中。然后将转移地址字段送入μPC中。
- 当执行返回微指令时，将RR中的返回地址送入μPC，返回微主程序。

  - 微子程序一般不嵌套。（如果允许子程序嵌套，则需用微堆栈。）
  - 增量方式的特点：简单，编制微程序容易，但不能实现多路转移。
  - 当需多路转移时，通常采用断定方式。

(2) 断定方式

- 断定方式：后继微地址可由设计者指定或由设计者指定的测试判定字段控制产生。

- 采用断定方式的后继微地址一般由两部分组成：
- 非因变分量：由设计者直接指定的部分，一般是微地址的高位部分。
- 因变分量：根据判定条件产生的部分，一般对应微地址的低位部分。

- 例：某机的微指令格式为：

μIR	OCF	微地址高位	A	B

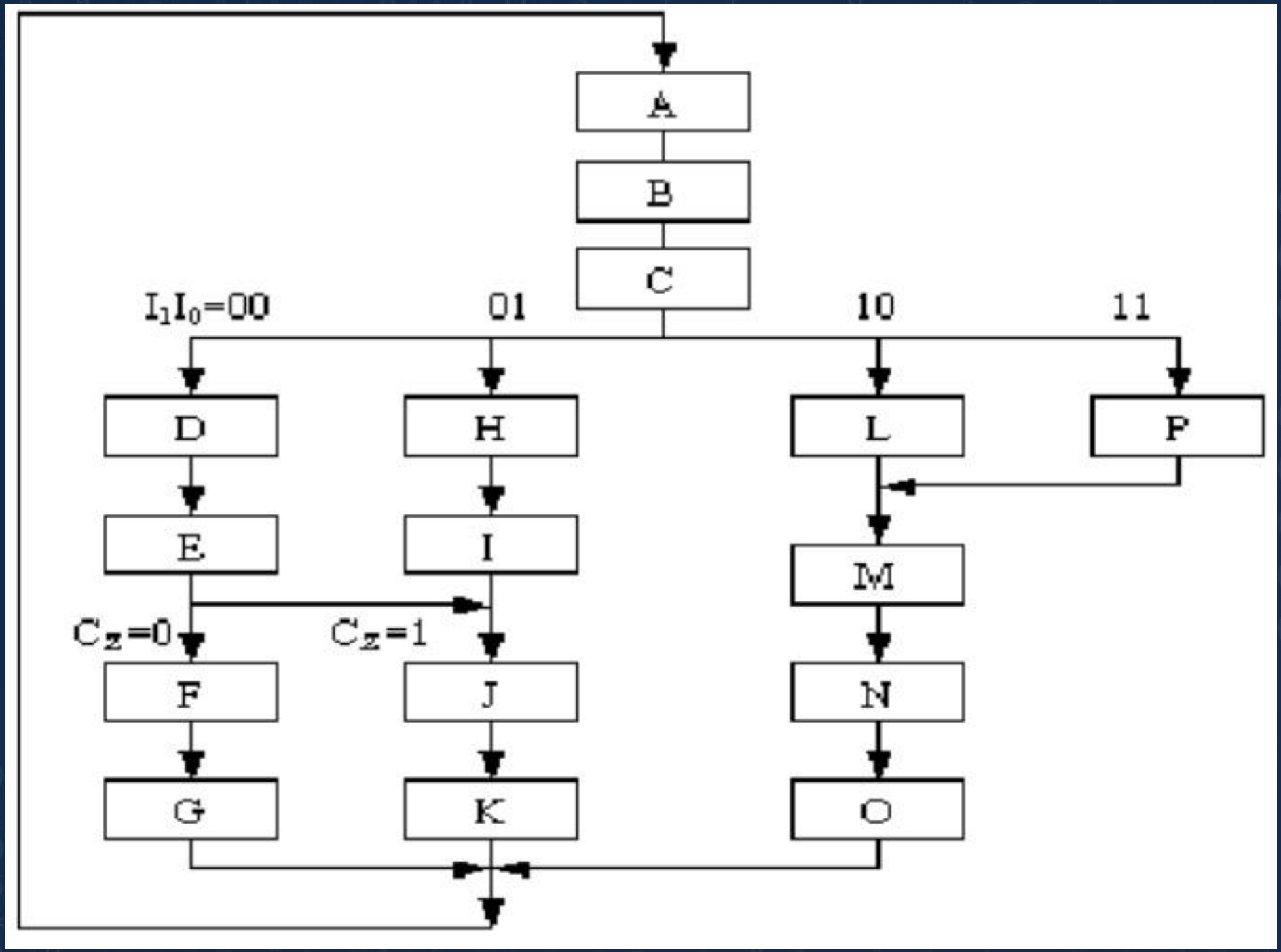
- A、B为两个判定条件

A	断定微地址低位	B	断定微地址低位
00	$0\rightarrow\mu MAR_1$	00	$0\rightarrow\mu MAR_0$
01	$1\rightarrow\mu MAR_1$	01	$1\rightarrow\mu MAR_0$
10	$C\rightarrow\mu MAR_1$	10	$V\rightarrow\mu MAR_0$
11	$Z\rightarrow\mu MAR_1$	11	$S\rightarrow\mu MAR_0$

- 设：微指令的高位地址为1011011
- A：01，B：11
- 则 后继微指令的微地址为： 1011011 1 S
- ∵是条件转移微指令，∴
- 若 S＝1，后继微地址为： 101101111＝16FH
- 若 S＝0，后继微地址为： 101101110＝16EH

- 断定方式的优点：
- 可以实现快速多路转移，适合于功能转移的需要。
- 断定方式的缺点：
- 编制微程序时，地址安排比较复杂，微程序执行顺序不直观。
- 在实际机器中，往往增量方式与断定方式混合使用。

- 例：图为一微程序流程，每一方框为一条微指令，用字母A～P分别表示微指令执行的微操作，该微程序流程的两个分支分别是：
- 指令的OP最低两位（I_1I_0）控制4路转移；
- 状态标志C_Z的值决定后继微地址的形成。
- 请设计该微程序的微指令的顺序控制字段，并为每条微指令分配一个微地址。


A
B
C
I_1I_0=00
01
10
11
D
H
L
P
E
I
M
C_Z=0
C_Z=1
F
J
N
G
K
O

- 解：由图可知，该微程序有两处存在分支：
- ① 指令操作码的I_1I_0（2位）：指出4条微指令的地址（控制转移），
- ② 根据运算结果标志C_Z的值决定2条微指令的执行次序。

- 该微程序中共有16条微指令，所以微指令中的下地址字段需要4位。

- 为了规定地址的形成方式，应在微指令中设置测试条件字段，用于描述后继地址的形成方式。
- 地址的形成方式有：
- ①直接根据下地址字段内容形成；
- ②根据指令操作码的I_1I_0形成；
- ③根据运算结果标志C_Z形成。
- 所以测试条件字段需要2位。

- 因此，本例的微指令格式由3部分内容组成，如下所示：

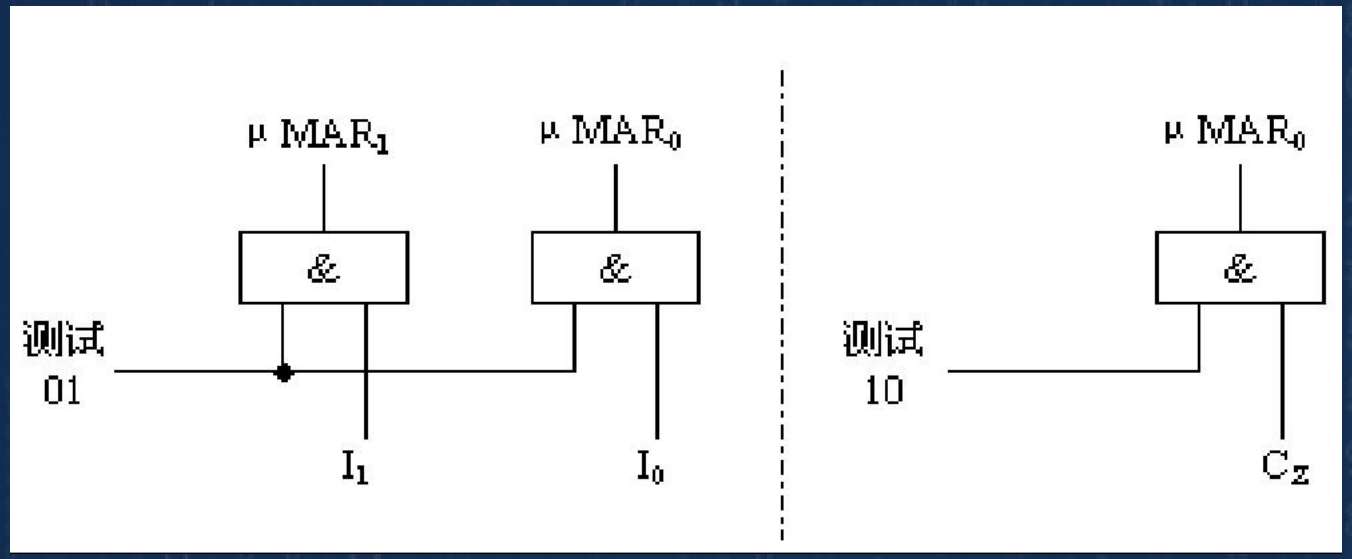
- 地址的分配关键在于分支微指令的安排，此时，对下地址字段的值具有一定的约束条件，一般取由测试条件控制的那几位为全0，目的在于简化地址修改逻辑。
- 在本题中，微指令C按指令OP（I_1I_0）实现4路分支，控制在末2位，这样，下地址的约束条件是末2位全为0，地址为0100，微指令C的后继4条微指令的地址分别为0100、0101、0110、0111，末2位实现了按I_1I_0转移；
- 同理，可以将按C_Z转移的地址定为10x0、10x1。

- 剩下的微指令的地址没有约束条件，可任意分配。一般可根据微程序流程从小地址到大地址（或从上到下、从左到右）顺序，将控制存储器中没有分配的微地址安排到不同的微指令中。

- 下表是本例微程序的微指令的地址分配结果，其中，绿色部分微指令下地址的形成受测试字段约束条件的控制。

微地址	微命令	测试条件	下地址	备　注
0000	A	00	0001	
0001	B	00	0010	
0010	C	01	$01I_1I_0$	按 OP 转移
0011	E	10	$101C_Z$	按 C_Z转移
0100	D	00	0011	由指令 OP 控制
0101	H	00	1000	
0110	L	00	1001	
0111	P	00	1001	
1000	I	00	1011	
1001	M	00	1100	
1010	F	00	1101	由 C_Z控制
1011	J	00	1110	
1100	N	00	1111	
1101	G	00	0000	
1110	K	00	0000	
1111	O	00	0000	

- 在微地址形成电路形成微地址时，可通过下图的修改电路生成受约束的微地址（高位地址指定）。



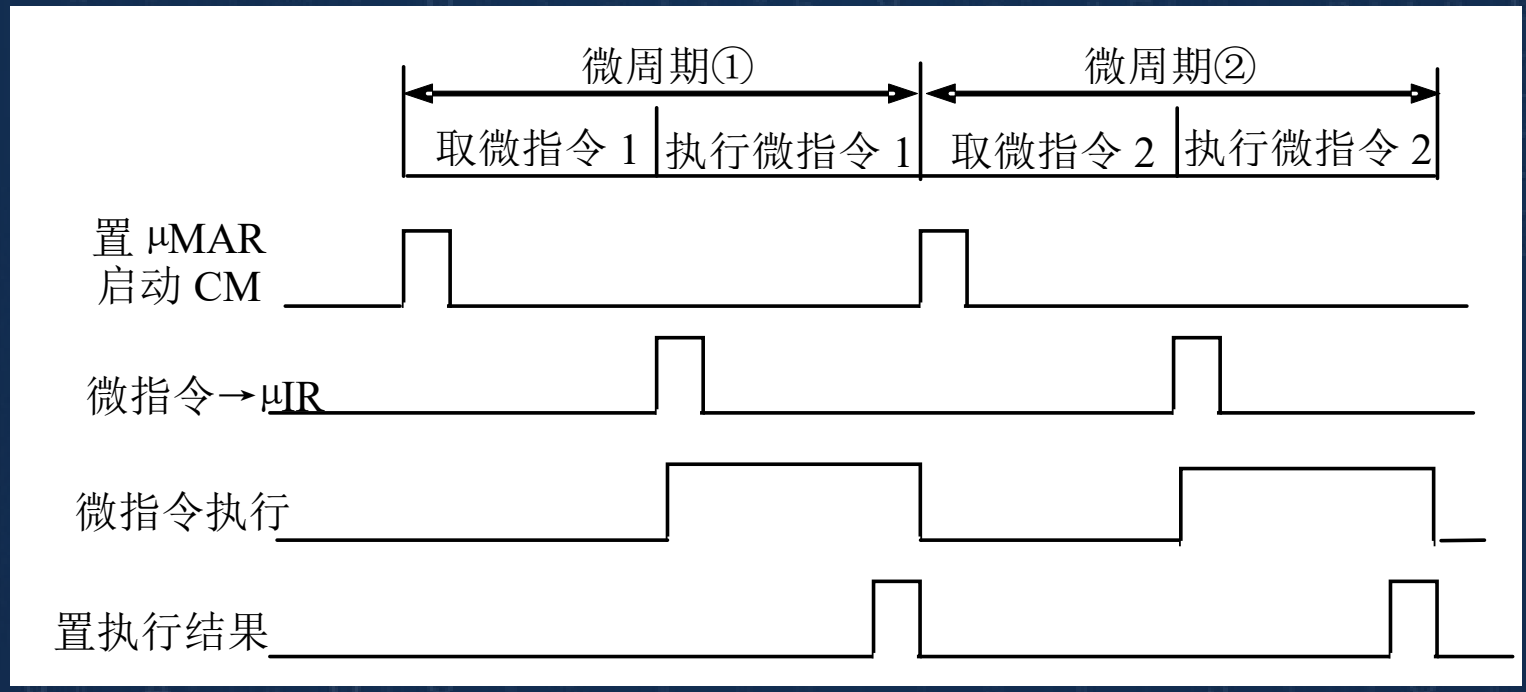
- *模型机的微程序设计（自学）

3. 微指令的执行方式

- 微程序控制器是通过一条一条地执行微指令来实现指令控制的。
- 执行一条微指令的过程：
- ① 取微指令：将微指令从控制存储器中取出。
- ② 执行微指令：执行微指令所规定的各个微操作。
- 根据取后继微指令和执行现行微指令之间的时间关系，微指令有两种执行方式：串行执行和并行执行。

1) 串行执行方式

- 取微指令和执行微指令是顺序、串行执行的，在一条微指令取出并执行完毕后，才能取下一条微指令。
- 串行执行时序图
- *

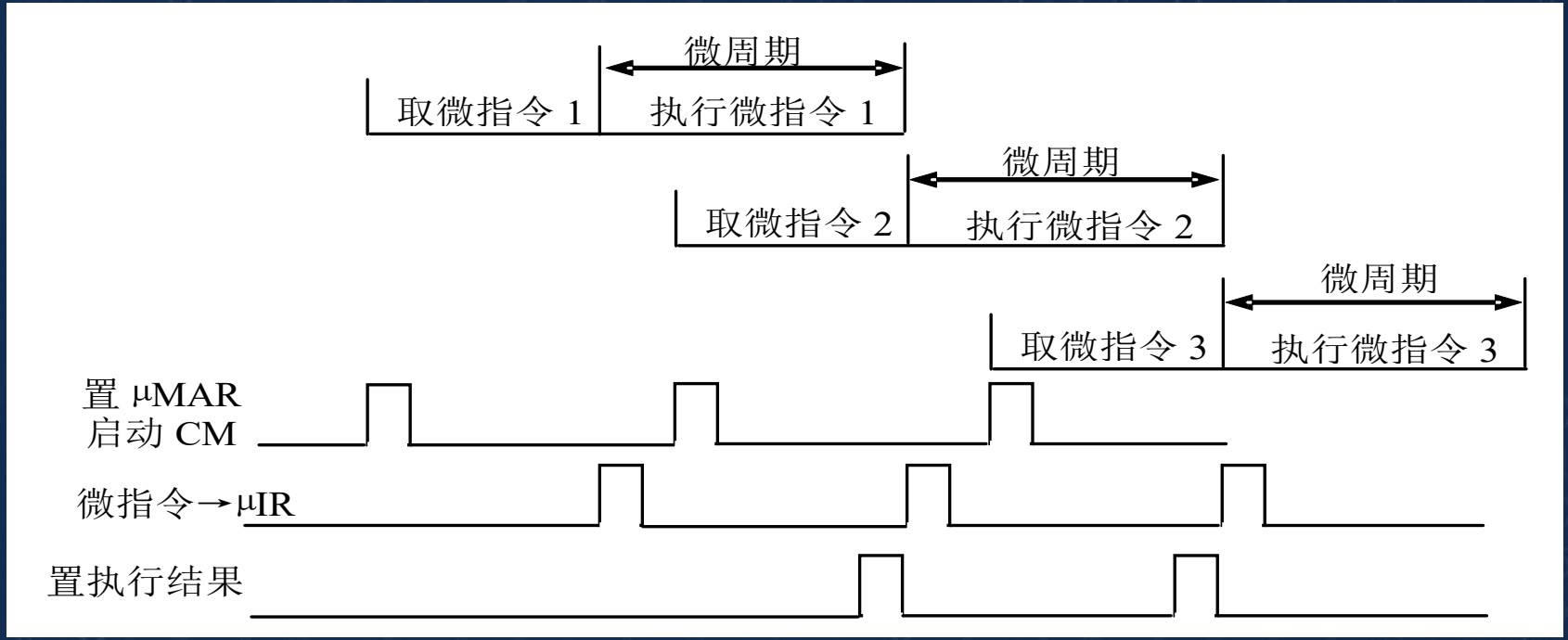


- 串行方式的优点：控制简单。
因为在每个微周期中，总要等到所有微操作结束，并建立了运算结果状态之后，才确定后继微指令地址。因此，无论后继微地址是按μPC增量方式，还是根据结果特征实现微程序转移，串行方式都可容易实现控制。

- 串行方式的缺点：设备效率低，执行速度慢。因为在一个微周期内的取微指令阶段，控制存储器工作，数据通路等待；而在执行微指令阶段，数据通路工作，控制存储器空闲，因而效率低。

2）并行执行方式

- 将取指令操作和执行微指令操作重叠起来。
- 由于取微指令与执行微指令分别在两个不同部件中执行，这种重叠是完全可行的。在执行本条微指令的同时，可预取下一条微指令。
- 并行执行时序图



- 假设取微指令所需时间比执行微指令短，所以可以将执行微指令时间作为微周期。
- 由于并行方式中取微指令与执行微指令在时间上有重叠，所以微程序执行速度比串行方式快，设备效率也高。

微指令的预取带来的控制问题

- 例如有时需根据运算结果特征实现微程序转移，而结果产生是在微周期的末尾，此时预取的微指令已经取出。若转移成功，预取的微指令无效。
- 如何处理并行方式中的微程序转移，是一个难度较大的问题。通常有延迟周期法、猜测法、预取多条转向微指令等方法。其中最简单方法就是延迟周期法。遇到按现行微指令结果特征转移时，延迟一个微周期再取微指令。

4. 微程序的设计方法

- 微指令格式的设计是微程序设计的主要部分，它直接影响微程序控制器的结构和微程序的编制，也影响着机器的处理速度和控制存储器的容量。
- 微指令格式的设计除了要实现计算机的整个指令系统之外，还要考虑具体的数据通路、控制存储器速度以及微程序的编制等因素。
- 在实际进行微程序设计时，应尽量缩短微指令字长，以减少微程序长度，提高微程序的执行速度。
- 不同机器有不同的微指令格式，但从其所具有的共性来看，通常可分为两大类。

1）水平型微指令与微程序设计

- 水平型微指令是一种广义的说法，并没有统一的确切定义。
- 水平型微指令：一次能定义并执行多个操作微命令的微指令。
- 水平型微指令通常采用直接控制和字段直接编码法对微指令进行编码，一条微指令中可包含多个微命令。

控制字段	判别测试字段	下地址字段

水平微指令的特点：

- ① 微指令字长，定义的微命令多。一般为几十位到上百位。
- 如VAX—11 / 780微指令字长为96位。巨型机ILLAIAC—IV微指令字长达280位。
- 微指令字较长，也意味着控制存储器的横向字长较长。
- ② 微指令中微操作并行能力强。在一个微周期中，一次能定义并执行多个并行操作微命令。
- ③ 微指令编码简单。一般采用直接控制方式和字段直接编码法，微命令与数据通路各控制点之间有比较直接的对应关系。

水平微指令的特点：

- ④ 编制的微程序短，效率高，微程序的执行速度快。
- ⑤ 所需控制存储器的纵向容量小。
- ⑥ 微指令字比较长，增加了控制存储器的横向容量。
- ⑦ 微指令长，定义的微命令多，使微程序编制困难、复杂，也不易实现设计自动化。

- 采用水平型微指令编制微程序称为水平微程序设计。

2）垂直型微指令与微程序设计

- 垂直型微指令在微指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能。
- 垂直型微指令一次只能控制信息从源部件到目的部件的一、两种信息传送过程。
- 例：某垂直型运算操作的微指令格式为：

μOP	源寄存器I	源寄存器II	目的寄存器	下地址
	R_1	R_2	R_S	

- μOP：微操作码
- 微指令功能：(R_1) μOP $(R_2) \rightarrow R_S$

- 例：某垂直型微程序转移微指令格式为：

微程序转移	条件测试	转移微地址

- 微指令功能：
- 根据条件测试字段所指定的条件对微指令的执行状态进行测试。若满足条件，微程序按指定的转移微地址转移。

垂直型微指令的特点：

- ① 微指令字短，一般为10～20位左右，使控制存储器的横向容量少。
- ② 微指令直观、规整、易于编制微程序，易于实现设计自动化。
- ③ 微指令字的微操作并行能力弱，一条微指令定义的微操作少，编制的微程序长，要求控存的纵向容量大。

垂直型微指令的特点：

- ④ 微指令编码复杂，微操作码字段需经过完全译码产生微命令，微指令的各个二进制位与数据通路的各个控制点之间不存在直接对应关系，执行速度较慢、效率较低。

- 采用垂直型微指令编制微程序称为垂直微程序设计。
- 垂直型微程序设计主要是面向算法的描述，故又称为软方法。

3） 毫微程序设计

- 毫微程序是一种解释微程序的微程序，而组成毫微程序的毫微指令是解释某一微指令的微指令。毫微程序设计就是用水平型的毫微指令来解释垂直型微指令的微程序设计。
- 毫微程序的设计思想：将水平和垂直两种微程序设计结合起来。

- 毫微程序设计采用两级微程序设计方法：

3） 毫微程序设计

- 第一级采用垂直型微程序设计，用于解释机器指令。这一级采用垂直型微指令编制垂直微程序，其并行功能不强，但有严格的顺序结构，可由它确定后继微指令的地址。

- 第二级采用水平型微指令编制水平微程序，用于解释垂直型微指令。这一级采用水平型微程序设计，具有很强的并行操作能力。

3） 毫微程序设计

- 毫微程序控制器中有两个控制存储器
- **微程序控制存储器(μCM)：**
  用于存放垂直微程序。
- **毫微程序控制存储器(nCM)：**
  用于存放毫微程序。

毫微程序的执行过程：

- 当执行一条指令时，首先进入第一级微程序，由于它是垂直型微指令，并行操作能力不强，当需要时，可由它调用第二级微程序(即毫微程序)中的毫微指令，用于解释该微指令的操作，毫微指令执行完毕，再返回第一级微程序。

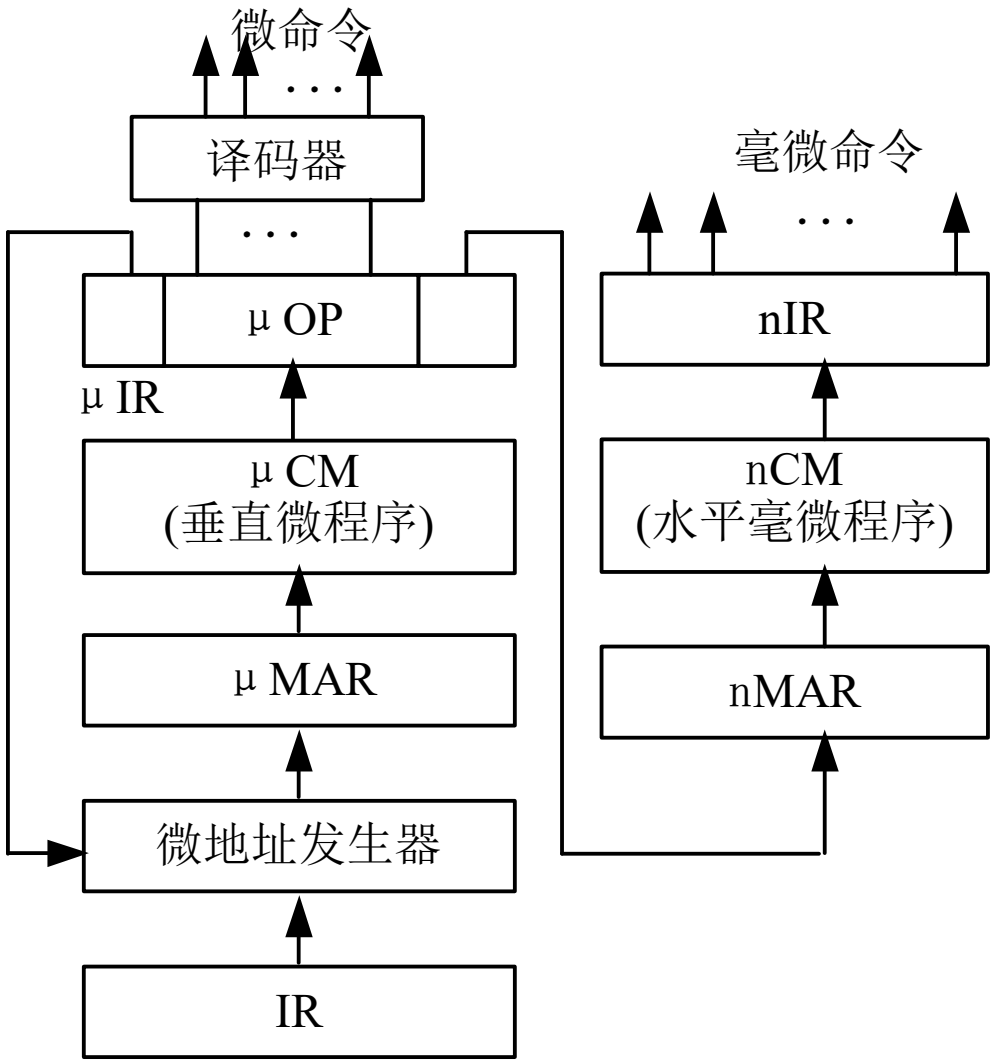


图4-55基于毫微程序设计的控制器结构

毫微程序的控制方式：

- 在毫微程序控制的计算机中，垂直型微程序是根据机器指令系统和其它处理过程的需要而编制的，它有严格的顺序结构。由于垂直型微指令很像机器指令，所以很易编制微程序。
- 水平型毫微程序是由垂直型微指令调用的，它具有较强的并行操作能力。若干条垂直微指令可以调用同一条毫微指令，所以在nCM中的每条毫微指令都是不相同的，相互之间也没有顺序关系。
- 从 μCM中读出一条微指令，除了可以完成自己的操作外，还可给出一个nCM地址，以便调用一条毫微指令，来解释该微指令的操作，实现数据通路和其它处理过程的控制。
- 毫微指令与微指令的关系类似于微指令与机器指令的关系。

毫微程序设计的缺点：

- ① 有时执行一条微指令需要两次访问控制存储器(一次访μCM，一次访nCM)，影响了速度。
- ② 增加了硬件成本。所以毫微程序设计多用于大、巨型机，而微小型机一般不用。

5. 微程序控制器的设计步骤

- 1） 确定微指令格式和执行方式
- 根据机器的微命令、微控制信号等具体情况决定是采用水平微指令格式还是垂直微指令格式，微指令是串行方式执行还是并行方式执行等。
- 2） 定义微命令集、确定微命令编码方式和微指令排序方式
- 根据机器指令的所有微控制信号拟定微命令集，确定微命令编码方式和字段的划分，选择微指令排序方法（增量式、断定式等）。

5. 微程序控制器的设计步骤

- 3） 编制微程序
- 列出机器指令的全部微命令节拍安排，按已定的微指令格式编制微程序，并对所有微程序进行优化和代码化。
- 4） 写入微程序
- 将二进制表示的全部微程序写入控制存储器。

- 例： 某计算机采用微程序控制器设计，已知每条机器指令的执行过程均可分解成8条微指令组成的微程序，该机指令系统采用定长格式，操作码字段为6位。
- 问：控制存储器至少应能容纳多少条微指令？如何确定机器指令操作码与该指令微程序的起始地址的对应关系？请给出具体方案。
- 解：⑴ 6位操作码可定义64种功能，所以共有64条机器指令。每条指令由8条微指令完成，所以控制存储器至少应能保存64×8＝512 条微指令。

- (2) 控存微地址宽度应为9位，以便寻址2^9个控存单元。
- 可以将各机器指令的6位操作码作为9位微地址的高6位，以形成该指令微程序的起始地址。
- μMAR地址格式(起始地址)：

- 模型机的微程序设计（自习，见附录B）