计算机组成原理与系统结构

第七章 控制器

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

杭州電子科技大學
HANGZHOU DIANZI UNIVERSITY

第七章　控制器

7.1 控制器的组成及指令的执行

7.2 硬布线控制器

7.3 微程序控制器

本章小结

BACK

7.3 微程序控制器

一 微程序控制的基本概念和工作原理

二 简单微程序控制器的设计

三 微程序设计技术

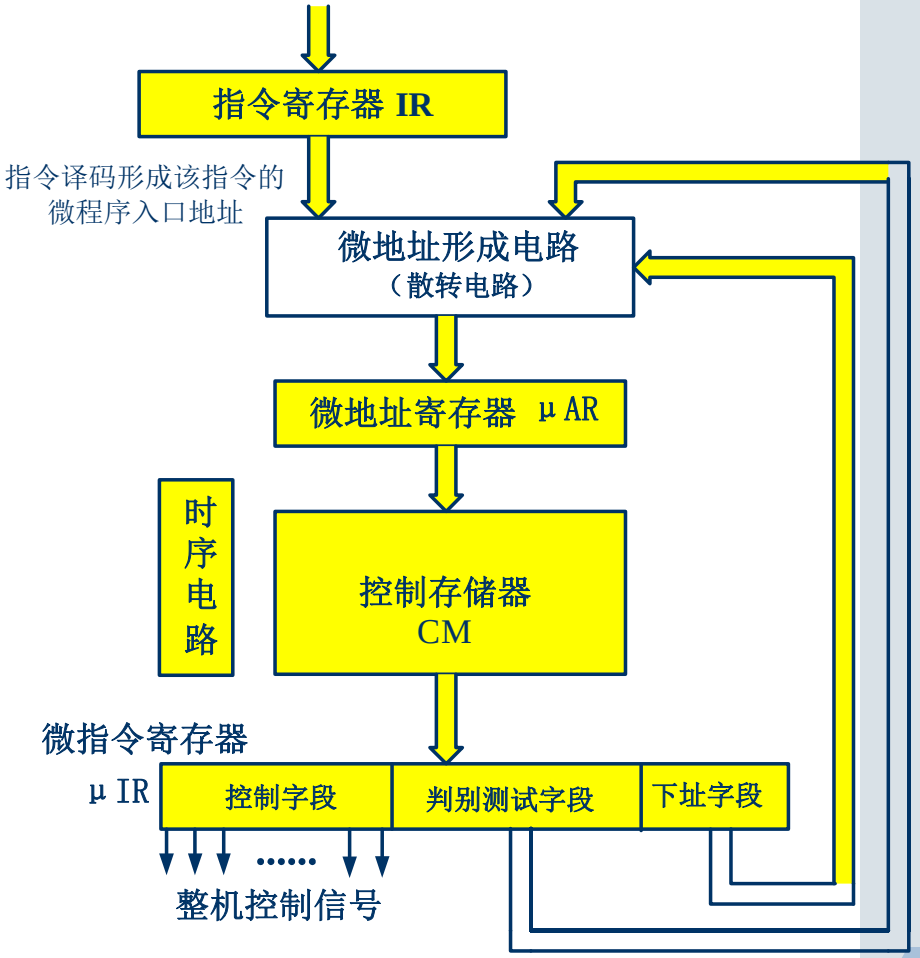
四 微程序控制方式下模型机的设计实例（自学）

五 模型机微程序设计（自学）

六 微程序控制器与硬布线控制器的比较

一、微程序控制的基本概念和工作原理

- 微地址寄存器μAR、控制存储器（简称控存）、微指令，微地址，微命令，微操作，微周期，微程序
- 微程序设计思想就是每条机器指令的功能都用一段相应的微程序来实现，在微程序设计中充分运用了软件的程序设计技术，使得微程序流程中也有微程序分支、微程序循环、微子程序等。
- 微指令由控制字段、判别测试字段和下址字段组成。



下址的生成方法

- 下址的生成方法有多种，常用的有以下四种：

① 第一种，下址就是控制存储器中的下一个地址，在这种情况下，控制器通常的实现方法是把当前地址加1来作为下址；

② 第二种，下址是由当前微指令提供的一个绝对控存地址，这个绝对地址可能是后继微地址的全部，也可能是后继微地址的一部分，在微程序发生转移的情况下，这种方法是很有效的；

下址的生成方法

① 第三种，根据机器指令操作码产生该指令对应的微程序入口地址（指令译码），通常这个工作由映象逻辑（mapping logic）来完成，指令操作码输入映象逻辑后，硬件将操作码映射成该指令对应执行指令周期中的第一条微指令的地址，即微程序入口地址，将微程序入口地址装入微地址寄存器就可以转入到正确的微程序。显然，对于整条指令来说，这种映象逻辑只用一次。

② 第四种，当调用微子程序时，其返回地址存储在微子程序寄存器或硬件堆栈里，微子程序寄存器或硬件堆栈都可以形成下址。

判别测试字段的作用就是指明下址的来源。

微程序控制器的基本工作原理

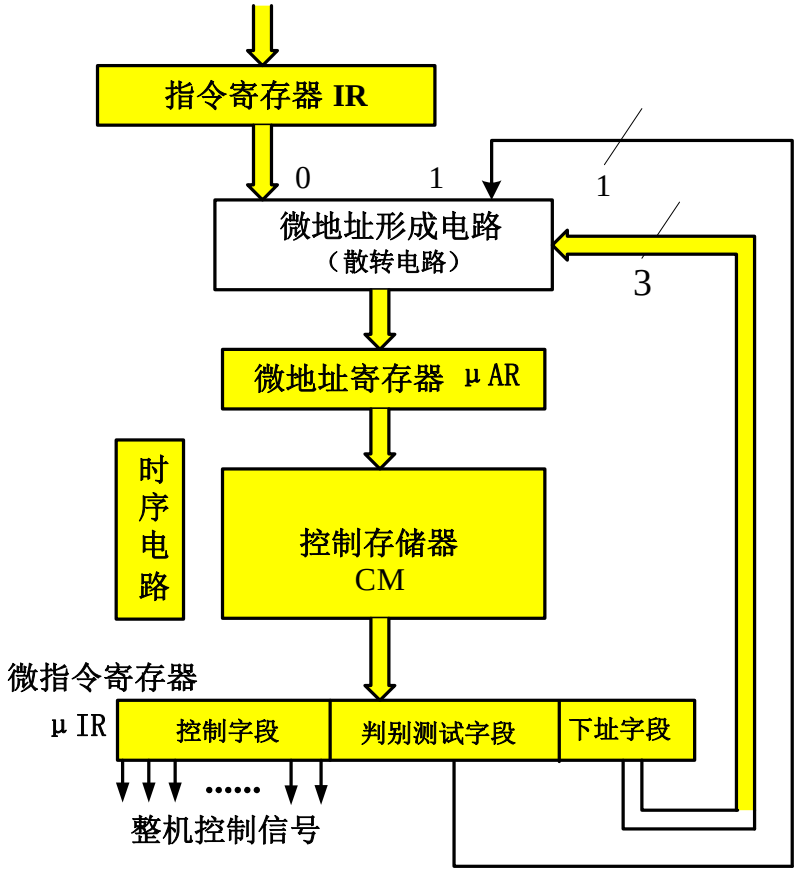
- 描述如下：
- 开机后首先使微地址寄存器置为取指令的第一条微指令地址，从控制存储器中取出第一条微指令，完成PC→AR、PC+1操作，然后根据微指令的下址字段分别取出第二条微指令，完成RAM→IR，即从内存中读出指令送指令寄存器IR，并且发译码信号使指令译码器工作，即形成该指令的执行指令阶段的微程序入口地址，从控存中取出该指令执行时的第一条微指令送到微指令寄存器，发出控制信号（微命令）实现微操作；然后该指令执行时的其余微指令地址是当前微地址加1或由当前微指令下址字段确定，依次从控存中取出其余微指令，实现该指令所需的所有微操作，即完成了该指令的执行。每一条指令的最后一条微指令执行完后均会回到取指令的第一条微指令执行，以取下一条指令，如此重复，直至用户要运行的程序指令执行完为止。

二、简单微程序控制器的设计

- 微程序控制器的设计主要完成两个任务：

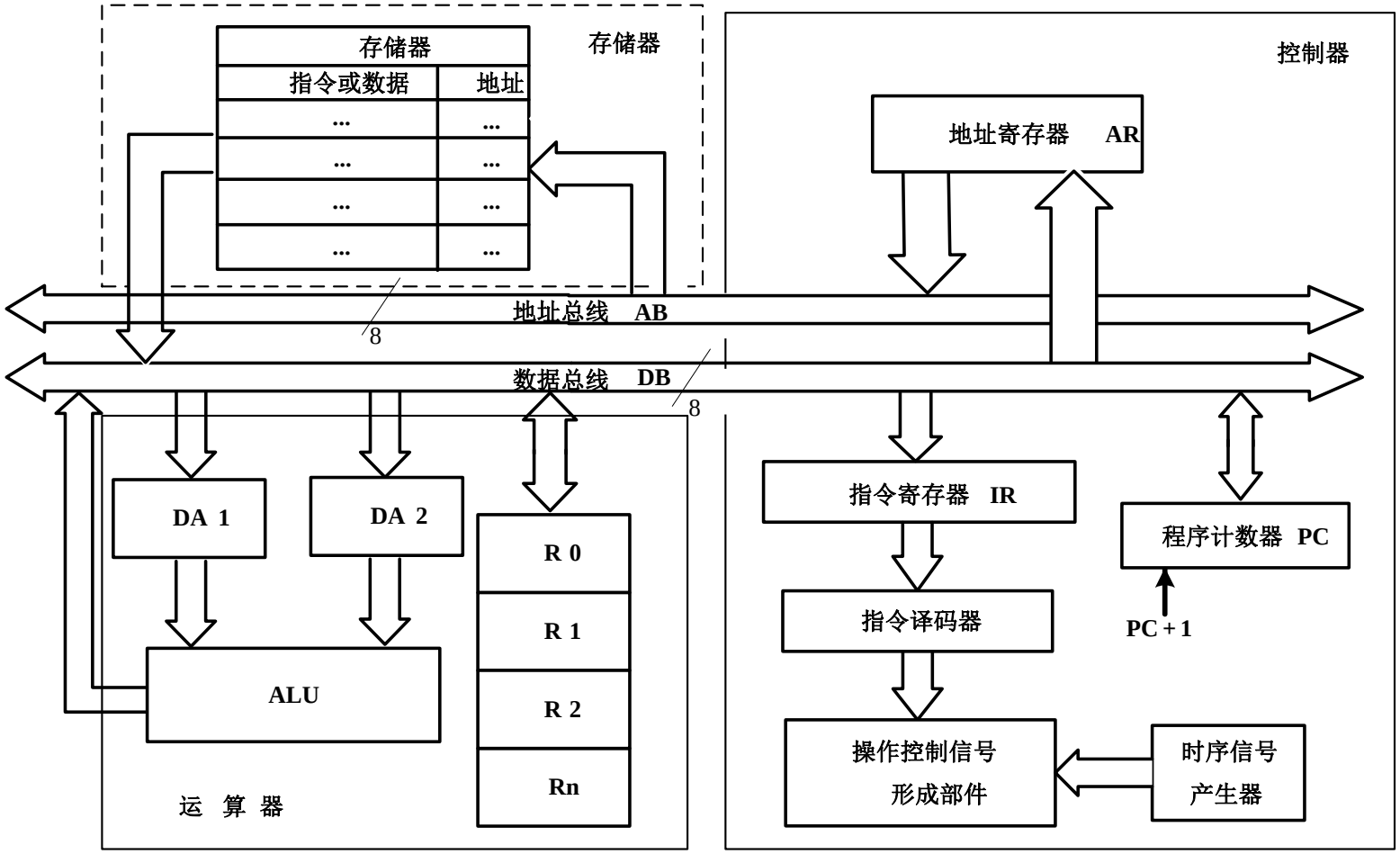
1. 产生正确的微命令；
2. 产生正确的微指令序列（即上述CPU状态转换序列）。

- 怎样采用微程序控制的方法来设计CPU呢？



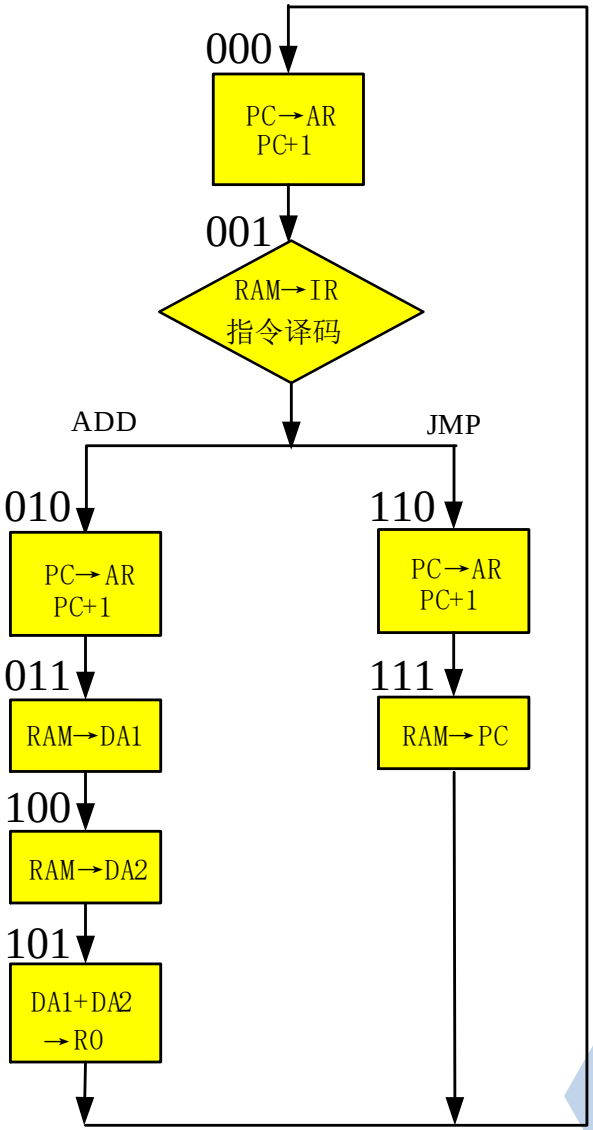
简单微程序控制器的组成框图

简单计算机系统的结构和数据通路



用微控方法来设计简单计算机系统的CPU

- 简单计算机系统的指令集：ADD和JMP
- CPU的有限状态机只有8个状态，可能产生8个下址。
- 在取指令周期的第二个状态（RAM→IR，指令译码）之后的微地址有两种可能性，微控器采用指令译码来产生下一微地址
- ADD和JMP指令的最后一个状态采用第二种方法即当前微指令提供下一微地址
- 其余状态下均把当前微地址加1来作为下一微地址。



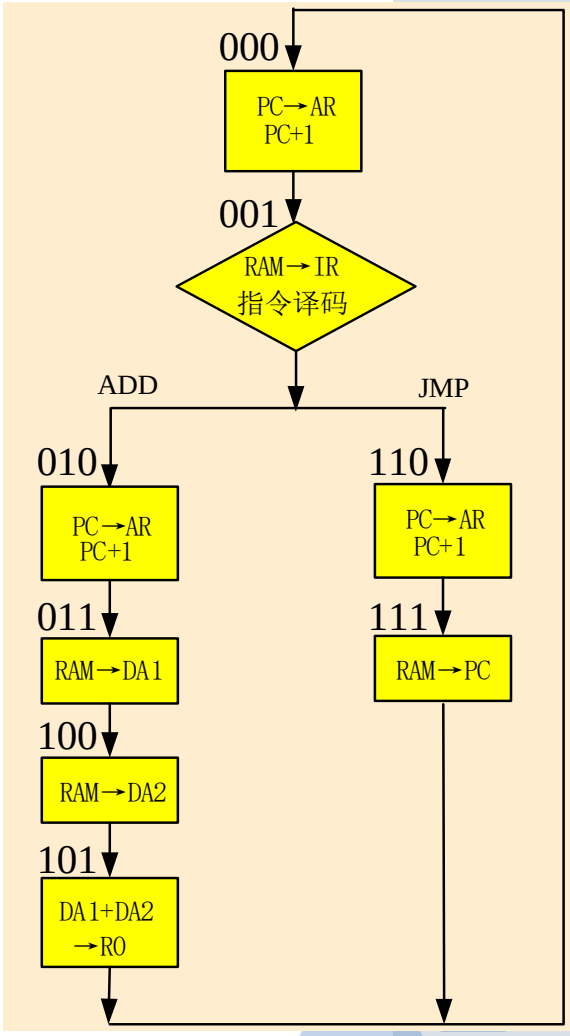
简单微控器的微程序流程图

简单微控器中微指令的下址设计表

- J1#=0表示下址来源是指令译码，J1#=1表示下址来源由微指令的下址字段提供

表7.5 简单微控器中微指令的下址设计表

微地址	状态	判别测试字段（J1#）	下址字段
000	取指 1：PC→AR,PC+1	1	001
001	取指 2：RAM→IR,译码	0	×××
010	ADD1：PC→AR,PC+1	1	011
011	ADD2：RAM→DA1	1	100
100	ADD3：RAM→DA2	1	101
101	ADD4：DA1+DA2→R0	1	000
110	JMP1：PC→AR,PC+1	1	111
111	JMP2：RAM→PC	1	000

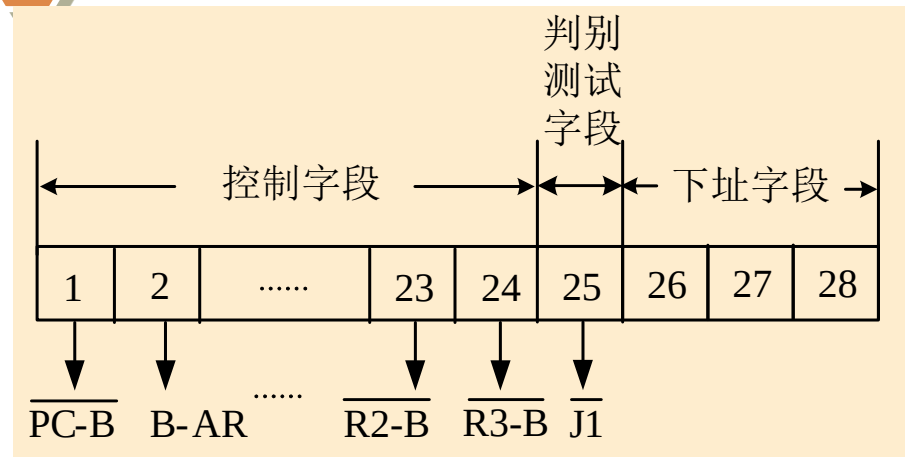


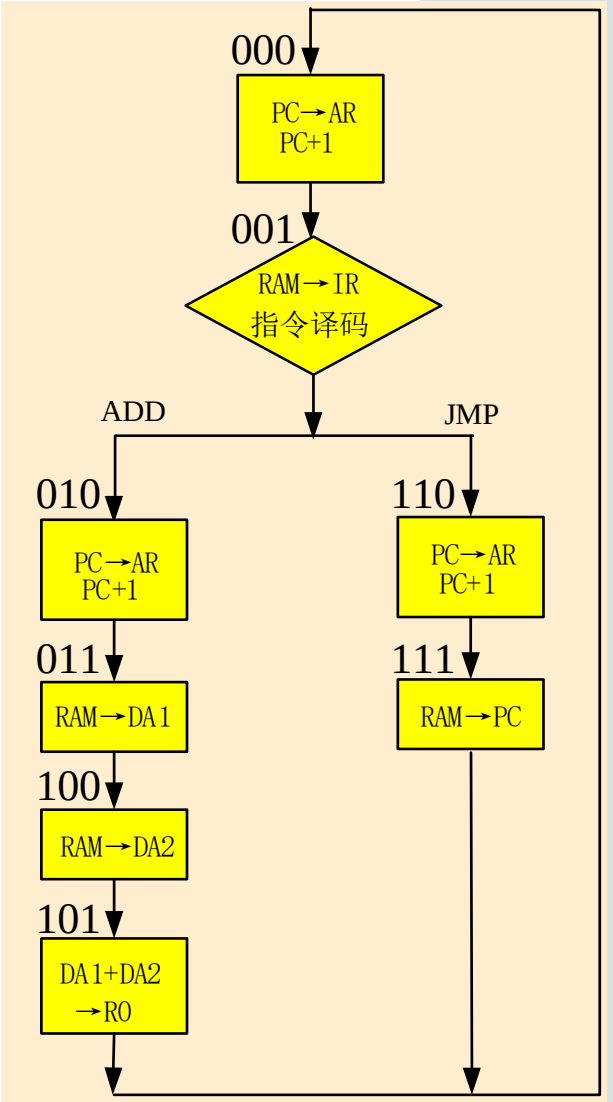
简单微控器的微程序流程图

简单计算机系统的控制信号一览表

序号	控制信号	功能	序号	控制信号	功能
1	PC-B#	指令地址（程序计数器）送总线	13	B-DA1	总线内容打入暂存器 DA1
2	B-AR	总线内容打入地址寄存器	14	B-DA2	总线内容打入暂存器 DA2
3	PC+1	程序计数器内容加一	15	ALU-B#	运算器 ALU 内容送总线
4	B-PC	总线内容打入程序计数器	16	Ci	ALU 进位输入
5	B-IR	总线内容打入指令寄存器	17	B-R0	总线内容打入 R0 寄存器
6	M-W#	存储器写	18	B-R1	总线内容打入 R1 寄存器
7	M-R#	存储器读	19	B-R2	总线内容打入 R2 寄存器
8	S_3	S_3- S_0选择 ALU16 种运算之一	20	B-R3	总线内容打入 R3 寄存器
9	S_2	同上	21	R0-B#	R0 寄存器内容送总线
10	S_1	同上	22	R1-B#	R1 寄存器内容送总线
11	S_0	同上	23	R2-B#	R2 寄存器内容送总线
12	M	M 为“1”选择 ALU 做逻辑运算，M 为“0”选择 ALU 做算数运算	24	R3-B#	R4 寄存器内容送总线

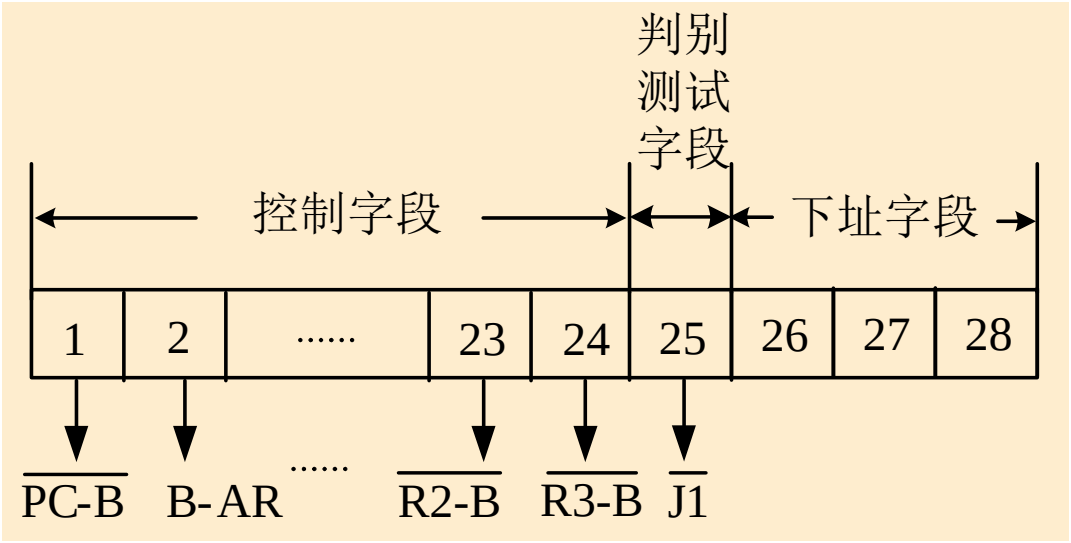
简单微控器中微指令的格式

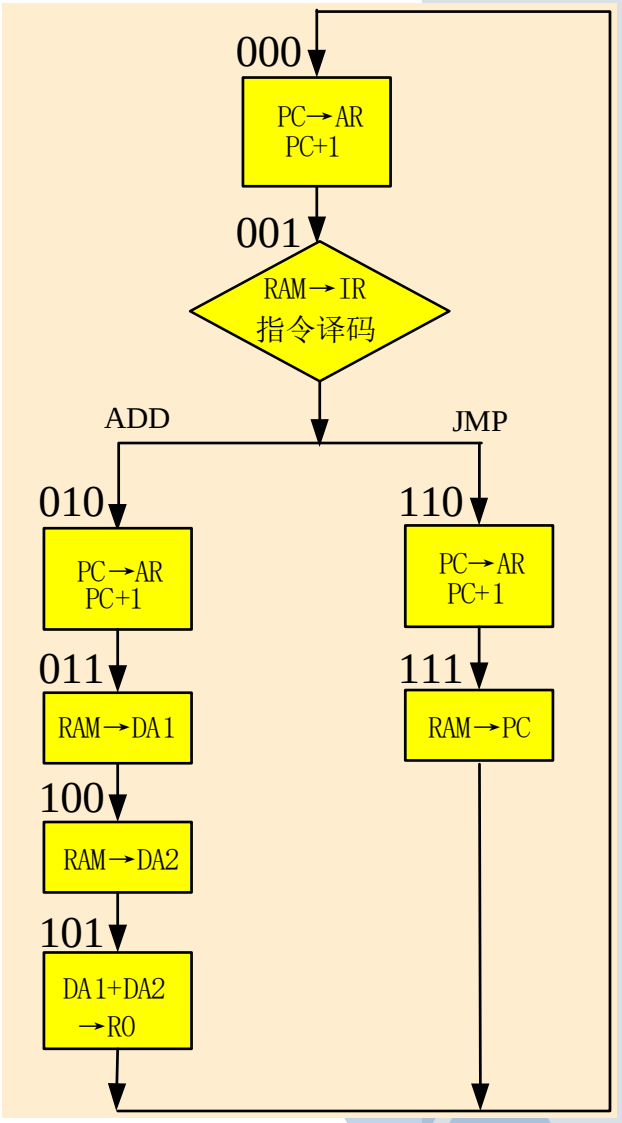




ADD指令的微指令

简单微控器中微指令的格式





JMP指令的微指令

三、微程序设计技术

- **进行微程序设计时，应考虑以下因素：**
  - ① 有利于缩短微指令字长；
  - ② 有利于减少控制存储器的容量；
  - ③ 有利于微程序的执行速度；
  - ④ 有利于对微指令的修改；
  - ⑤ 有利于微程序设计的灵活性。

三、微程序设计技术

1. 微指令的编译法
2. 微指令下址字段设计方法
3. 微指令格式的类型
4. 微程序控存和动态微程序设计
5. 毫微程序设计

1、微指令的编译法：指微指令中控制字段的设计方法

（1）直接控制法

（2）字段直接编译法

（3）字段间接编译法

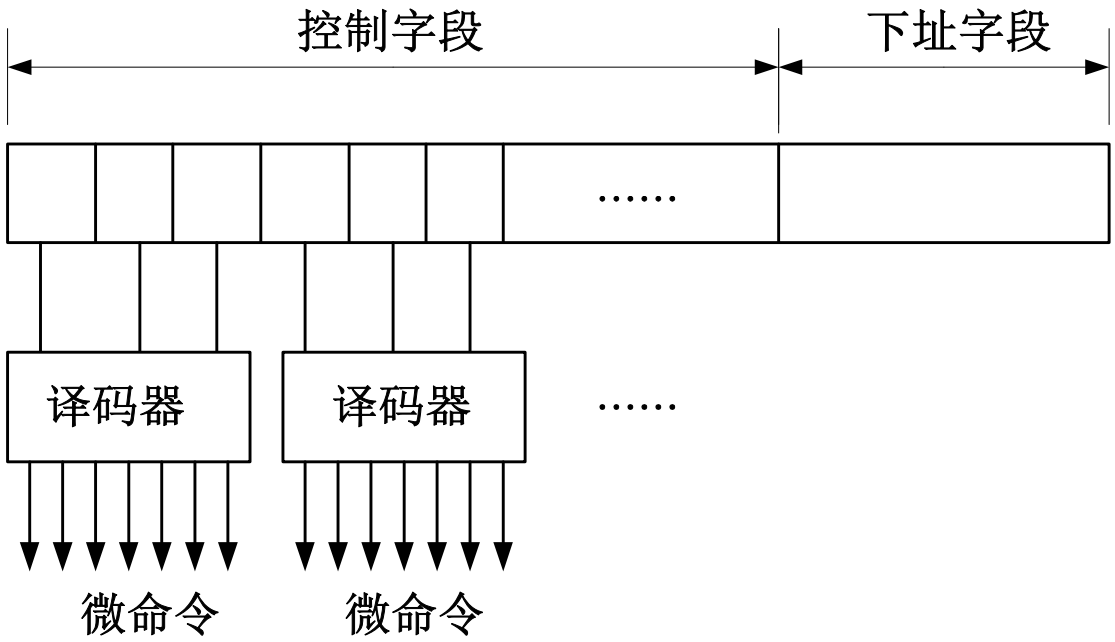
（1）直接控制法

- 在微指令的控制字段中，每一位代表一个微命令（控制信号），在设计微指令时，如果要发出某个微命令则将控制字段中对应位置有效，这样就可以打开或关闭某个控制门，这就是**直接控制法**。如果是编码控制则置相应编码值。
- 缺点：微指令的控制字段太长

（2）字段直接编译法

- 如果有一组微命令，在任一个微周期（一条微指令所需的执行时间）只有一个微命令起作用，那么这一组微命令是互斥的；通常将在同一个微周期中不能同时出现的微命令称为相斥性微命令，而将在同一个微周期中可以同时出现的微命令称为相容性微命令。
- 字段直接编译法的分段原则是：
  ① 相斥性微命令分在**同一字段**内，**相容性微命令**分在**不同字段**内。
  ② 一般将同类操作中互斥的微命令划分在一个字段内，这样使微指令结构清晰，易于编制微程序和易于扩充功能。
  ③ 每个小字段的信息位不能太长，一般**不超过6位**，否则将增加译码线路的复杂性和译码时间。

（2）字段直接编译法



用微指令的编译法设计简单微控器的微指令格式

- 重新设计的简单微控器的微指令格式如表7.7所示，其中BTO和OTB字段编码表如表7.8所示，由此就把微指令的长度从28位缩短到了17位。

表7.7 采用字段直接编译法设计的简单微控器的微指令格式

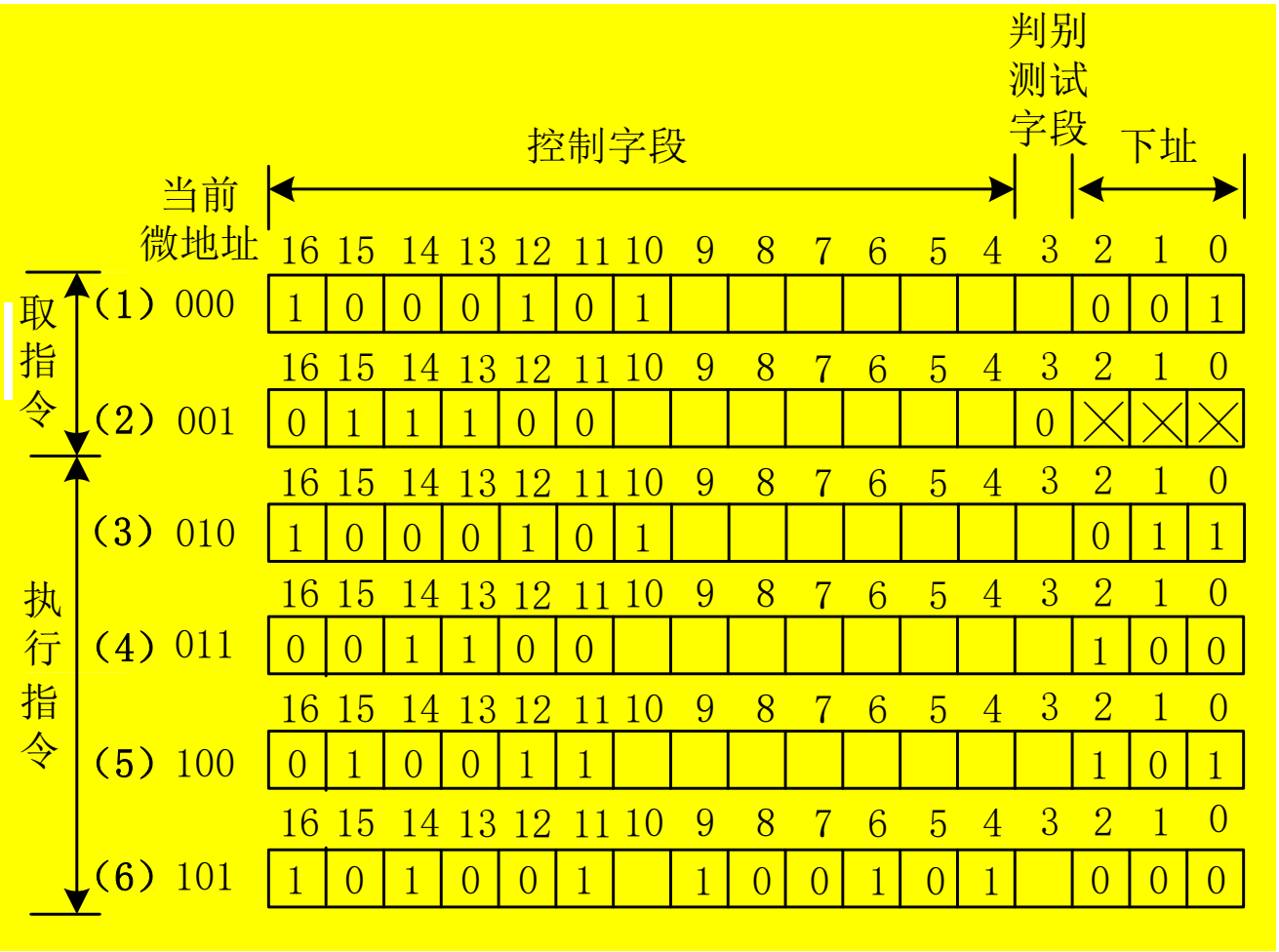
M16~M14	M13~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	PC+1	S3	S2	S1	S0	M	Ci	J1#	下址

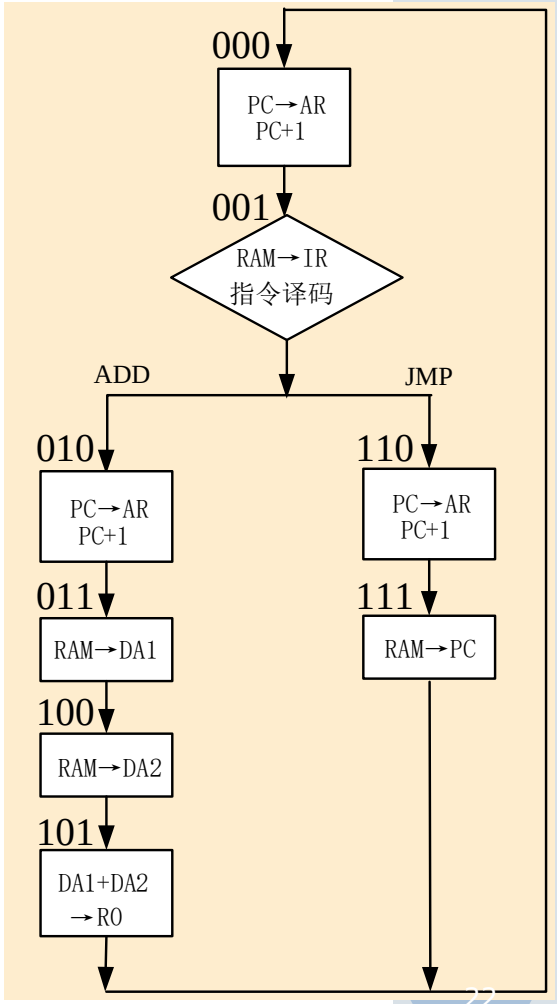
表7.8 微指令字段编码表

编码+译码	BTO	OTB
000		
001	B-DA1	ALU-B#
010	B-DA2	PC-B#
011	B-IR	R0-B#
100	B-AR	M-R#
101	B-R0	
110	M-W#	
111	B-PC#	

字段直接编译法设计的ADD指令的微指令

M16~M14	M13~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	PC+1	S3	S2	S1	S0	M	Ci	J1#	下址

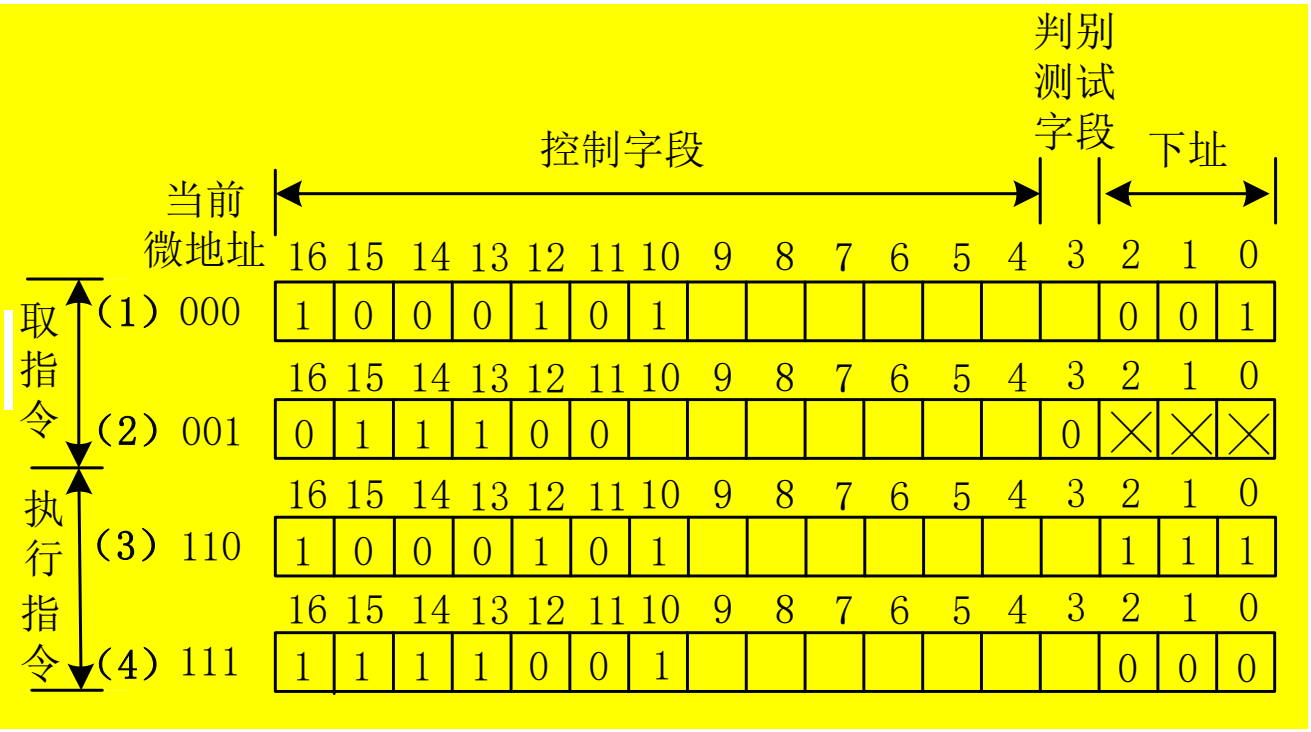


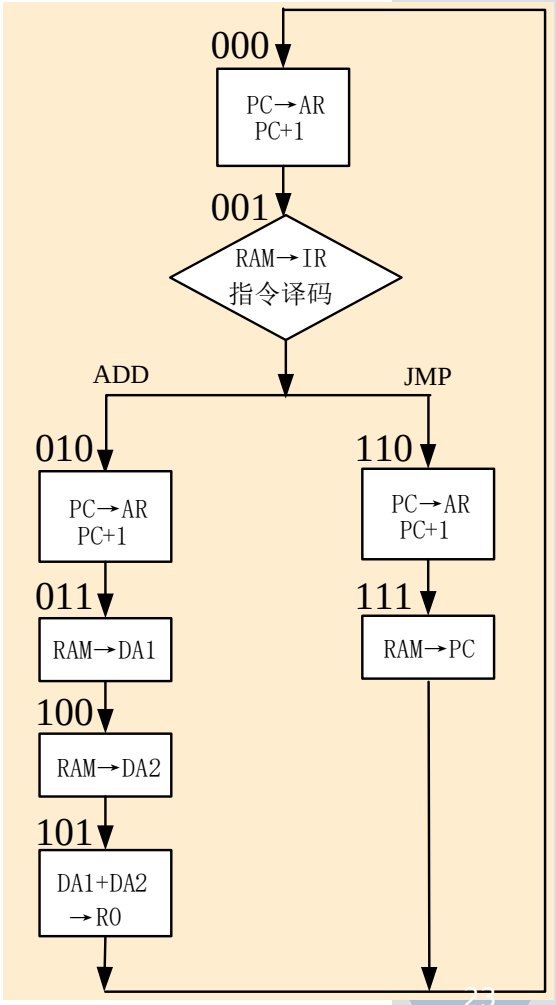


ADD指令的微指令

字段直接编译法设计的JMP指令的微指令

M16~M14	M13~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	PC+1	S3	S2	S1	S0	M	Ci	J1#	下址





JMP指令的微指令

（3）字段间接编译法

- 指某字段的编码含意，除了其本身的编码外，还需要由另一字段来加以解释的编码方法，
- 字段间接编译法是在字段直接编译法的基础上，进一步缩短微指令字长的一种编译法。
- 举例：
  - 用字段间接编译法来进一步设计简单微控器的微指令格式

表7.7 采用字段直接编译法设计的简单微控器的微指令格式

M16~M14	M13~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	PC+1	S3	S2	S1	S0	M	Ci	J1#	下址

表7.9 字段间接编译法设计的简单微控器的微指令格式

M15~M13	M12~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	FUNC	FS	S3	S2	S1	S0	M	Ci	下址

字段间接编译法举例

字段直接编译法的微指令各字段编码及微命令

编码+译码	BTO	OTB
000		
001	B-DA1	ALU-B#
010	B-DA2	PC-B#
011	B-IR	R0-B#
100	B-AR	M-R#
101	B-R0	
110	M-W#	
111	B-PC#	

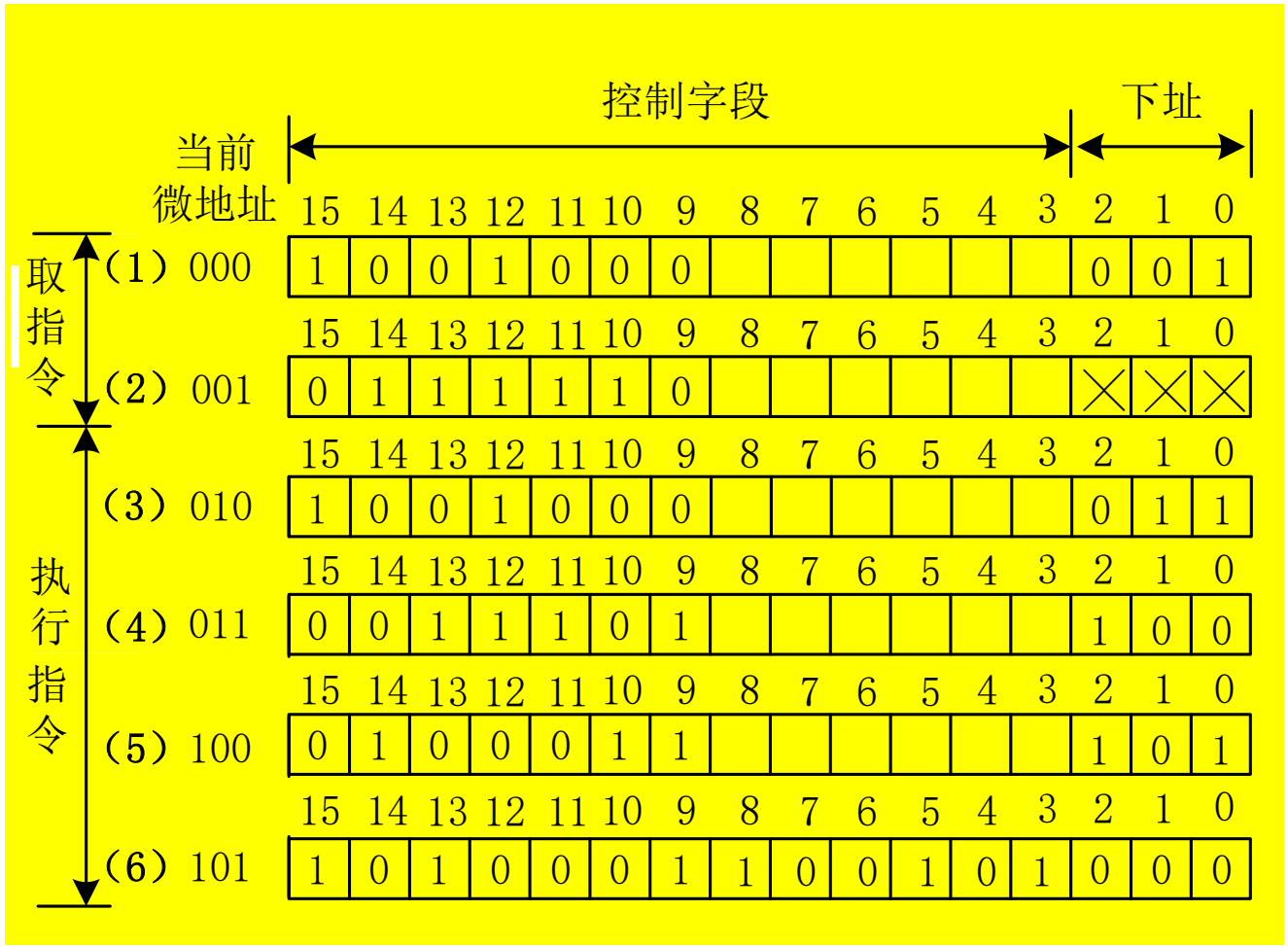
表7.10 字段间接编译法的微指令各字段编码及微命令

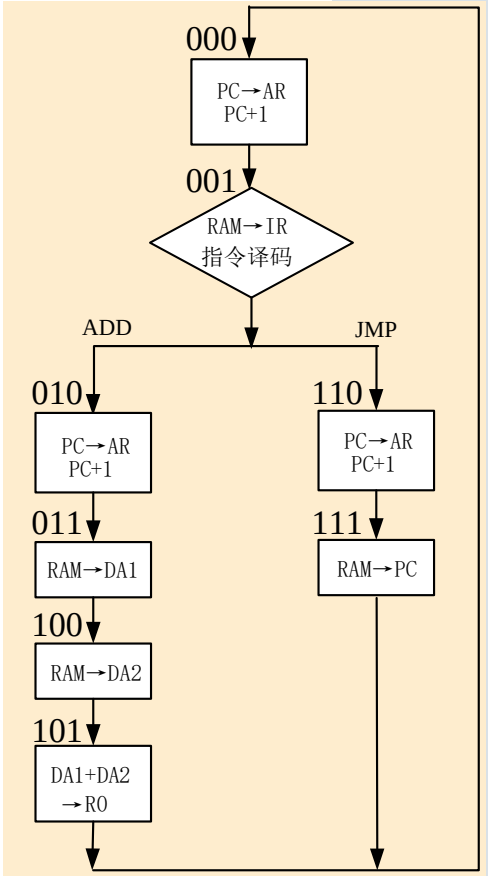
BTO 编码	BTO 信号	OTB 编码	OTB 信号	FUNC 编码	FS=0	FS=1
000		00		0	PC+1	
001	B-DA1	01	ALU-B#	1	J1#	R0-B#
010	B-DA2	10	PC-B#			
011	B-IR	11	M-R#			
100	B-AR					
101	B-R0					
110	M-W#					
111	B-PC					

用字段间接编译法设计的ADD指令的微指令

表7.9 字段间接编译法设计的简单微控器的微指令格式

M15~M13	M12~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	FUNC	FS	S3	S2	S1	S0	M	Ci	下址

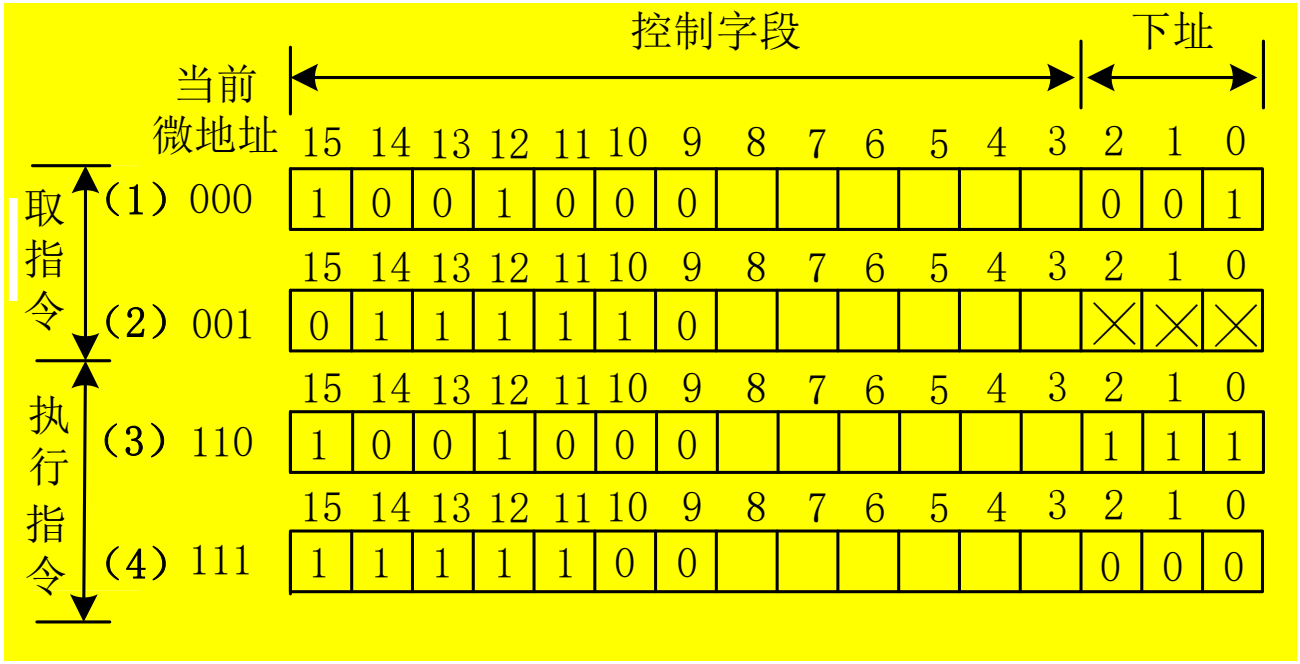


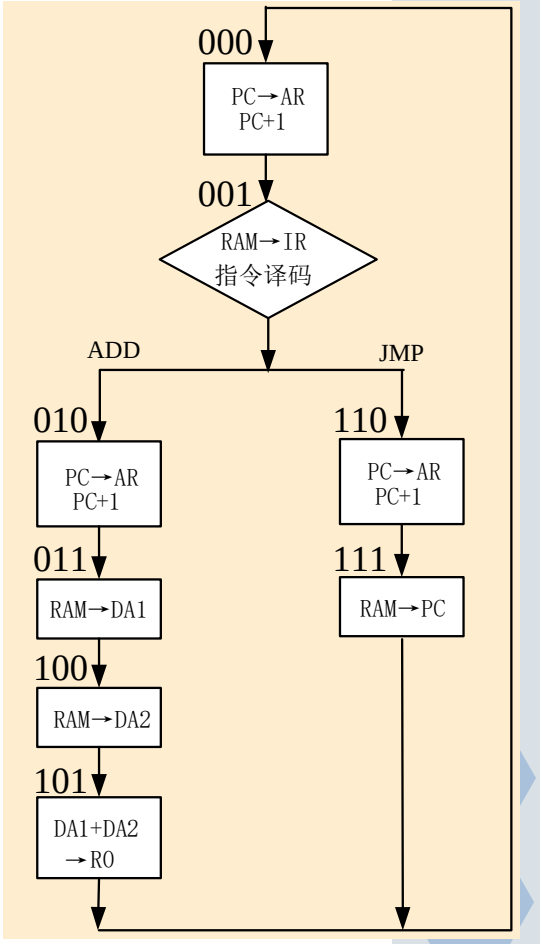


用字段间接编译法设计的JMP指令的微指令

表7.9 字段间接编译法设计的简单微控器的微指令格式

M15~M13	M12~M11	M10	M9	M8	M7	M6	M5	M4	M3	M2~M0
BTO	OTB	FUNC	FS	S3	S2	S1	S0	M	Ci	下址





2、微指令下址字段设计方法

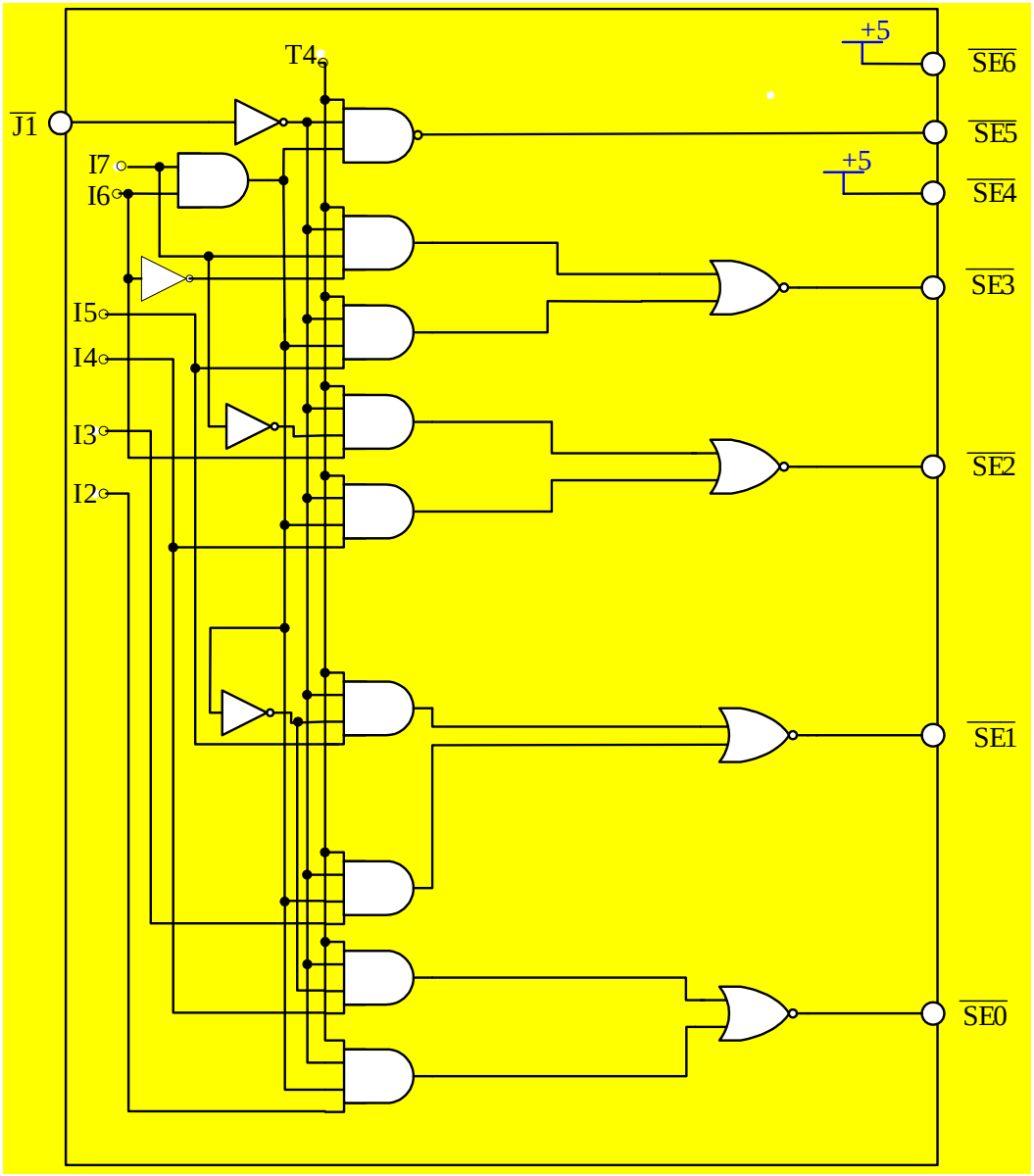
（1）
微程序入口地址的产生

（2）
后继微地址的产生

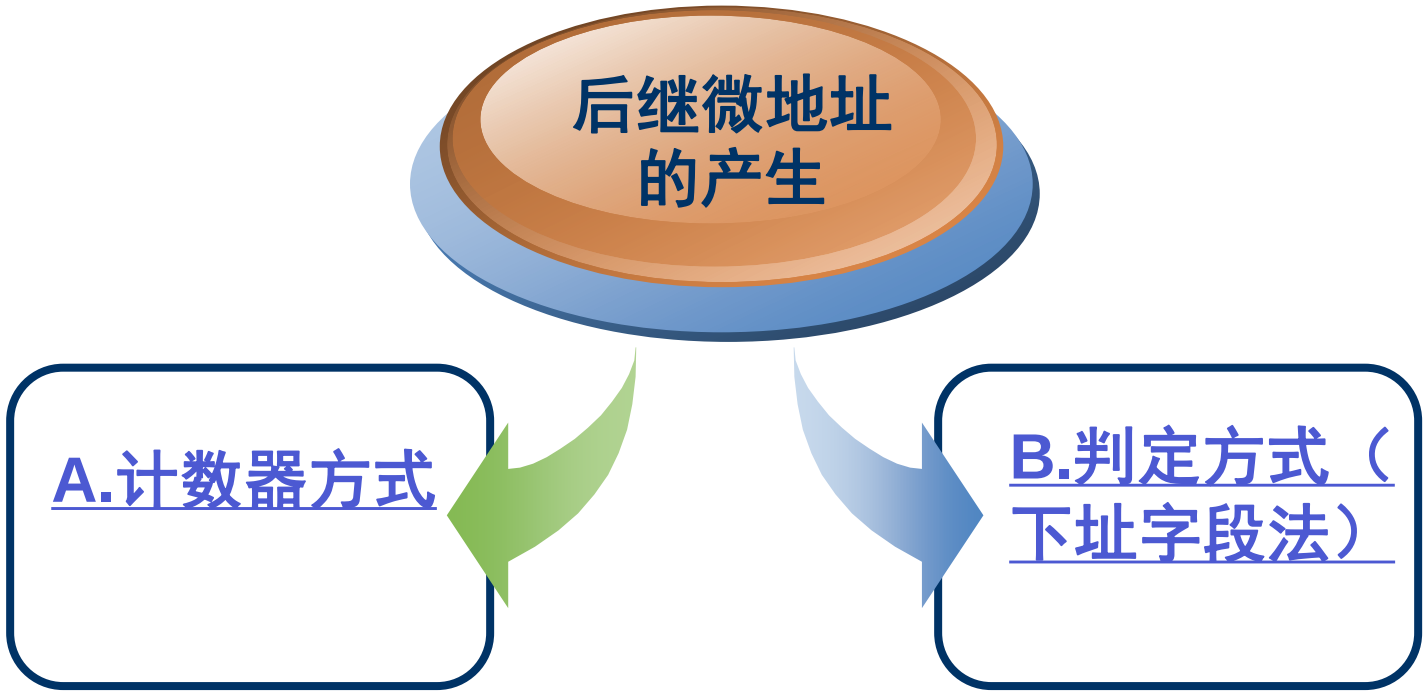
（1）微程序入口地址的产生

- 指令译码产生相应微程序入口地址如果采用PROM来实现，这种PROM称为映射存储器（MAPROM），它是将机器指令的操作码映射成该指令对应的微程序入口地址。
- 指令译码产生相应微程序入口地址也可以采用逻辑电路来实现。
- **举例：简单计算机需要执行多于两条指令**
  - 指令译码的规则是根据操作码I7-I2进行散转
    a）当I7 I6≠11时，微程序入口地址的逻辑表达式是MA6 MA5 MA4 I7 I6 I5 I4；
    b）当I7 I6=11时，微程序入口地址的逻辑表达式是MA6 MA5 MA4 I5 I4 I3 I2。
    c）MA6~ MA0是微指令的下址字段。
  - 如果采用逻辑电路产生微程序入口地址，且在T4时钟周期进行指令译码

指令译码产生微程序入口地址的逻辑电路图



（2）后继微地址的产生

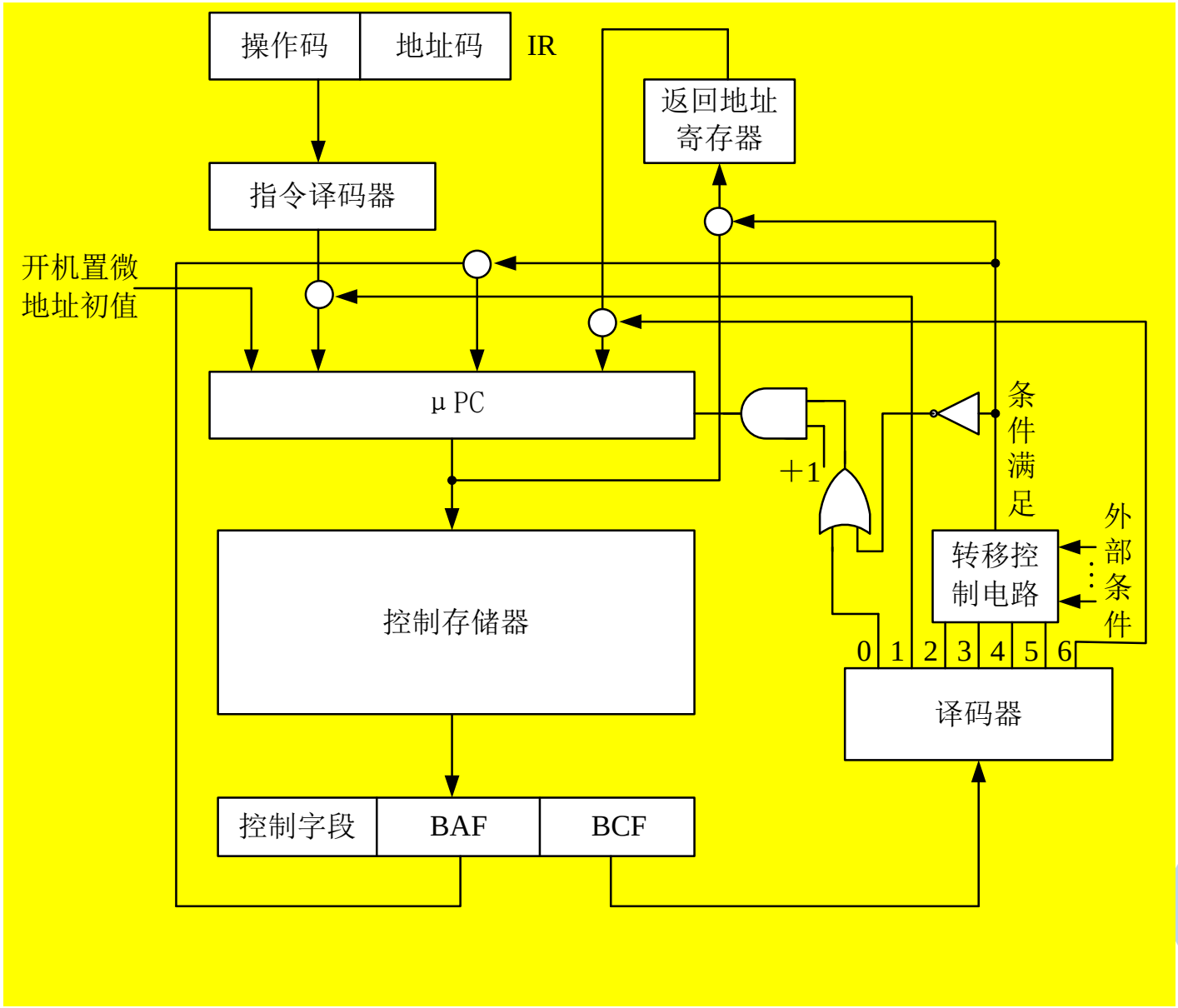


A. 计数器方式

- 在微程序控制器单元中设置一个微程序计数器μPC,
  - 在顺序执行微指令时，后继微指令地址由现行微地址加上一个增量（通常为1）来产生。
  - 遇到转移时，由微指令给出转移地址，使微程序按新的微地址顺序执行。
- 这种方式下微程序流控制由两部分信息确定：**转移控制字段BCF**和**转移地址字段BAF**，其微指令格式

  - 转移地址字段BAF用于给出微指令转移的部分微地址
  - 转移控制字段BCF用来确定后继微地址是顺序执行还是条件转移。
  - 当微程序转移条件成立时，将BAF送μPC，否则顺序执行下一条微指令（μPC＋1）

计数器方式产生后继微地址的原理图



计数器方式产生后继微地址的工作过程

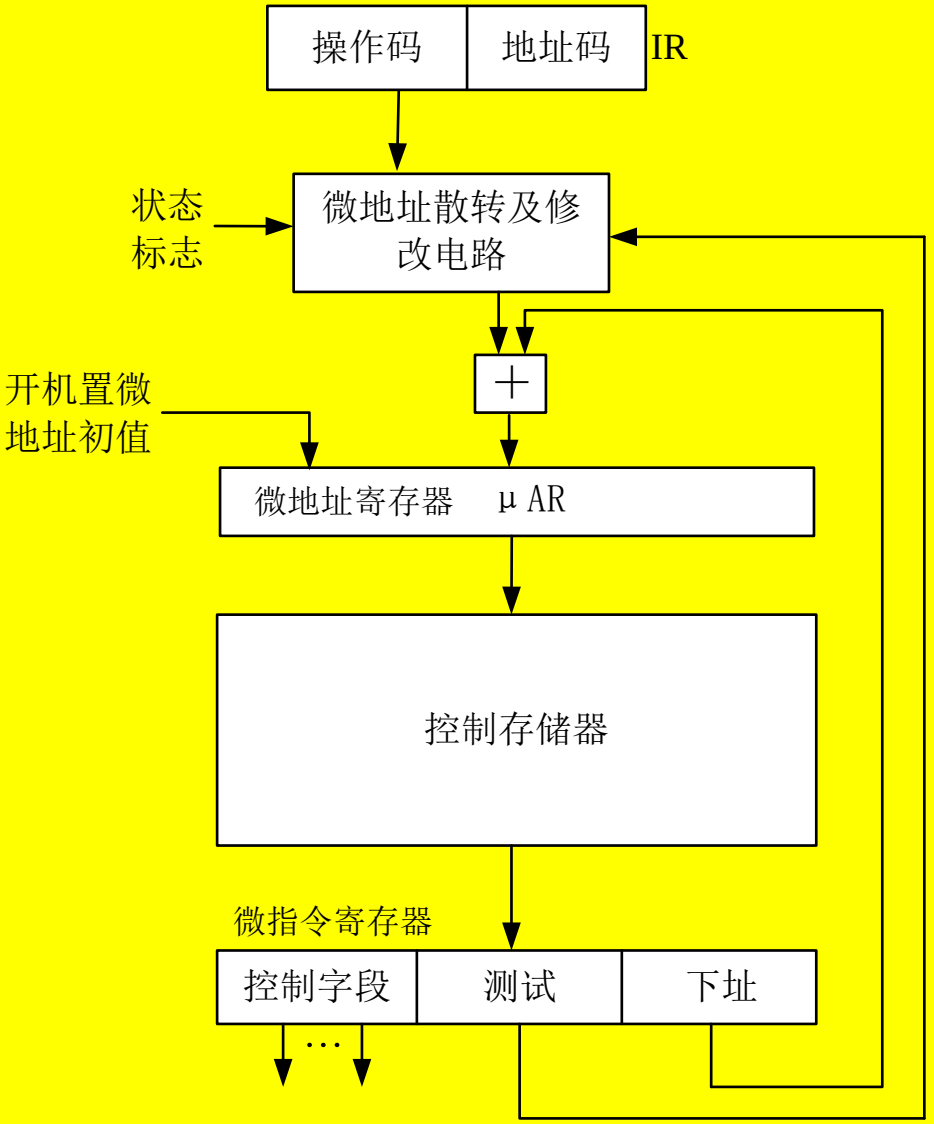
- 机器加电后执行的第一条微指令地址，也就是取指令的微程序起始地址由硬件电路置入μPC中，顺序执行取指令微程序，实现从主存中取出机器指令送指令寄存器IR；然后，由机器指令操作码译码后产生相应微程序入口地址，接下去若顺序执行微指令，则将μPC＋1作为后继微地址；若遇转移类微指令而且转移条件满足，则由BAF和μPC组合产生后继微地址，否则μPC＋1。这样，直至一段微程序执行完毕，完成一条机器指令为止；重复又转入取指令微程序……

B. 判定方式（下址字段法）

- 判定方式其后继微指令地址可由设计者指定或设计者指定的测试判别字段控制产生。
  - 当微程序不产生分支时，后继微指令地址直接由微指令的下址字段给出；
  - 当微程序出现分支时，按测试判别字段和状态条件通过逻辑电路来形成后继微地址。
- 这种方式要在微指令格式中设置一个字段用来指明下一条要执行的微指令地址，所以也称为下址字段法。
- 在这种方式中，因为每一条微指令至少都是一条无条件转移微指令，因此不必设置专门的转移微指令。

判定方式产生后继微地址的原理图

- 判定方式的优点：可以实现快速多路分支，以提高微程序的执行速度，微程序在控制存储器中的物理分配方便，微程序设计灵活；
- 缺点：微指令字加长，形成后继微地址的结构比较复杂。



3、微指令格式的类型

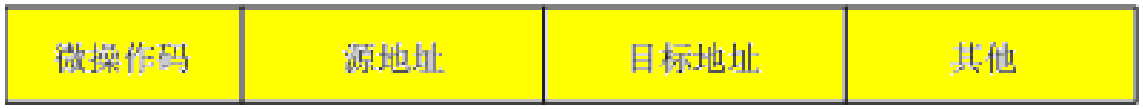
① 水平型微指令

- 基本特征是：一条微指令能控制数据通路中多个功能部件并行操作。
- 优点：一条微指令可同时发许多个微命令，且微指令控制字段直接控制，微指令执行效率高，速度快，灵活，各部件执行操作的并行能力强；编制的微程序比较短。
- 缺点：微指令字太长，明显地增加了控制存储器的横向容量。

3、微指令格式的类型

② 垂直型微指令

- 采用完全编码的方法，将一套微命令代码化构成微指令。因此，一条微指令只能控制1~2种微操作，类似机器指令，由微操作码、源地址和目标地址以及其他附带信息，垂直型微指令的格式

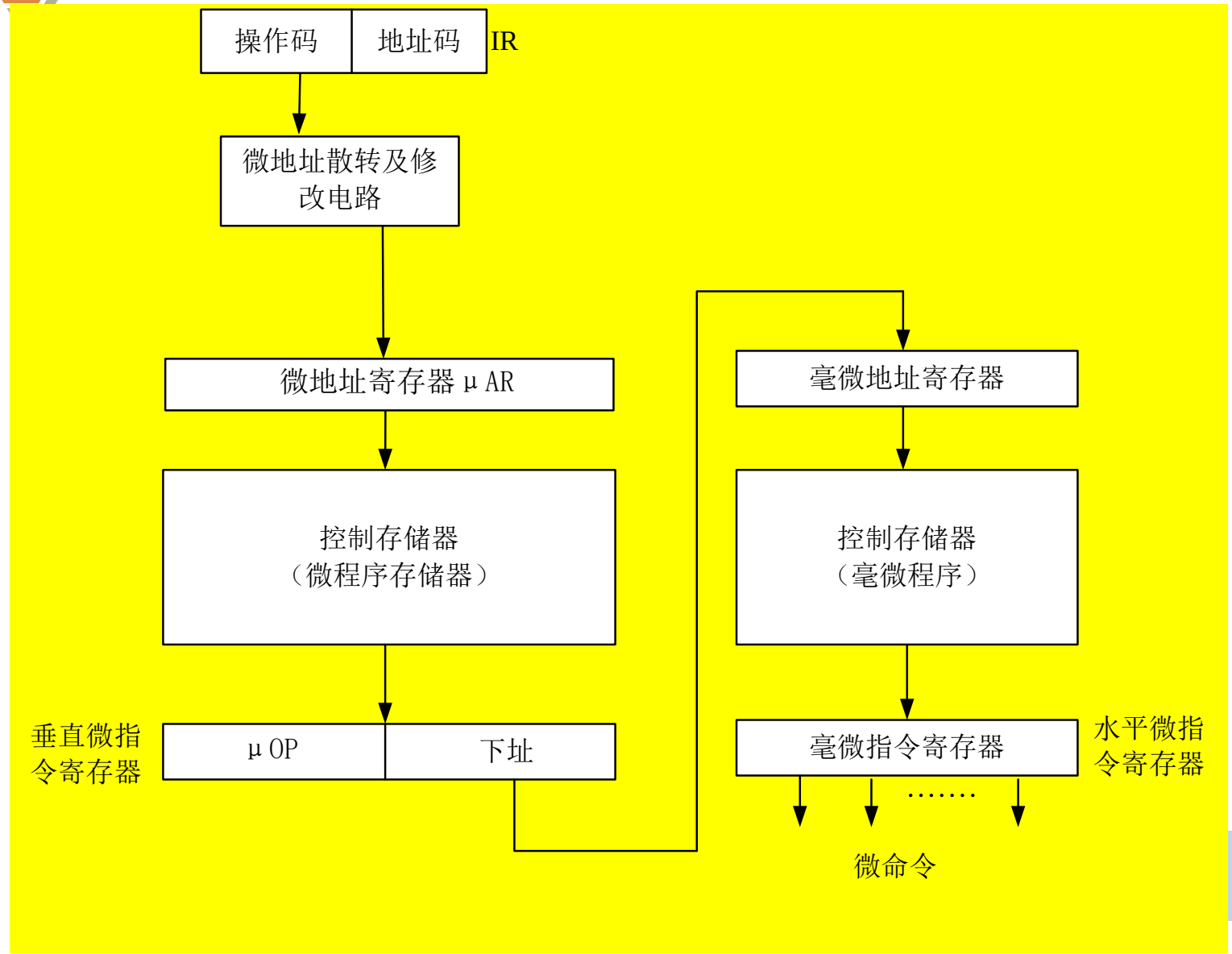


- 垂直型微指令比较直观，容易掌握和便于使用。微指令字短，减少了横向控制存储器的容量。
- 缺点是：微指令要经过译码才能发出微命令，微指令的执行效率低，并行操作性比较差，解释一条机器指令或完成某种功能所需要的微指令条数增多，增加了纵向微程序容量。

4、微程序控存和动态微程序设计

- 微程序控制存储器可由只读存储器(ROM)构成，也可以由可读/可写的随机存储器（RAM）构成。
- 采用RAM作为控制存储器的优点是可以修改微程序。在一台微程序控制的计算机中，假如能根据用户的要求改变微程序，那么这台机器就具有动态微程序设计功能。
- 动态微程序设计的目的是使计算机能更灵活、更有效地适应于各种不同的应用场合。
- 动态微程序设计要求用户对计算机硬件组成结构非常熟悉，因此真正由用户自行编写微程序是很困难的，所以尽管设想很好，实事上是难以推广，一般要由机器设计人员才能设计实现。

5、毫微程序设计


操作码
地址码
IR
微地址散转及修改电路
微地址寄存器 μ AR
控制存储器（微程序存储器）
垂直微指令寄存器
μ OP
下址
毫微地址寄存器
控制存储器（毫微程序）
毫微指令寄存器
水平微指令寄存器
.....
微命令

六、微程序控制器与硬布线控制器的比较

1. 它们的根本区别在于微操作控制信号的产生方法不同：
   - 微程序控制器事先将编写好的微代码放入控制存储器，执行过程中需要时再从控存中读取并送出的；
   - 硬布线控制器由组合逻辑电路产生微操作控制信号。
2. 从电路的规整性来说，微程序控制器电路相对规整，而硬布线控制器电路设计较为繁琐、不规整。
3. 从指令系统的易扩充性来说，微程序控制器易修改和扩充，而硬布线控制器不易修改和扩充。

六、微程序控制器与硬布线控制器的比较

4. 微程序控制器执行指令的速度相对硬布线控制器慢，因为前者需要读控存、微指令译码、发送微操作控制信号来完成一个CPU周期（机器周期），而后者经过一些门电路的延迟，即可产生微操作控制信号，所以更利于硬布线控制器的CPU提高主频。
5. 微程序控制器早先多应用于CISC系统，硬布线控制器多应用于RISC系统。在现代的RISC系统中，绝大多数指令为简单指令，均用硬布线方式实现；少数复杂指令则使用微程序实现。而在现代的CISC系统中，譬如Intel微处理器，也将一些原先采用微程序实现的指令，改用硬布线实现，以提高CPU速度和节省芯片面积。

本章小结

- 控制器是计算机硬件的核心部件，是根据机器指令来产生指令执行时全机所需要的操作控制信号，协调控制计算机各个部件有序工作。指令的执行阶段由其操作码决定，通过分析指令的各种微操作，从而形成控制器的设计思路。
- 控制器的设计方法有两种：
  - 硬布线控制器的设计方法，它是将指令执行时的各个机器周期的微操作信号用时序逻辑电路来实现，硬布线控制器速度快，但设计复杂繁琐，适合于RISC结构。
  - 微程序控制器的设计方法，基于程序设计的思想来设计控制器。其设计关键是指令译码形成微程序入口、微指令格式设计及确定微指令流（后继微指令地址）等技术。微程序控制器相对硬布线控制器速度慢，但设计比较规整，易于实现指令系统修改，适合于CISC结构。

本章小结

- 本章介绍了
  - 硬布线控制器的时序逻辑电路的设计方法，通过MIPS及其核心指令子集的硬布线控制器的设计实例，掌握硬布线控制器的时序逻辑电路的初步设计方法。
  - 模型机的微程序控制器设计方法，可以通过模型机的设计案例，掌握机器指令、微程序控制器及其微程序设计的方法。

The End!