第五章-微程序控制器

(一)微程序控制的基本原理

微程序控制的基本思想：用二进制编码字（称为微指令字）来代替硬连逻辑控制器中的微操作控制信号的产生。

把一条指令的执行过程中各节拍要进行的微操作集合用一个微指令字表示，然后把它们按节拍的先后顺序存放到一个特殊的存储器中（称为控制存储器 CM）。

执行该指令时，按顺序依次读出微指令字。

几个基本概念：

微命令：控制部件向执行部件发出的各种控制命令。构成控制信号序列的最小单位。例如打开或关闭某个控制门，多路器选择哪个输入等。

微操作：执行部件接受微命令后所进行的最基本的、不可再分割的操作。分为相容的微操作（可同时进行的微操作）和互斥的微操作（不可同时进行）。

微命令和微操作是一一对应的。微命令是微操作的控制信号，微操作是微命令的操作过程。

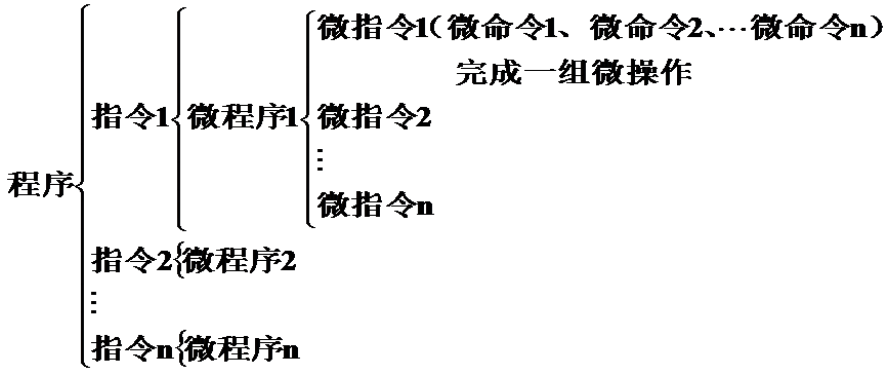
微指令：用来产生微控制信号的二进制编码字，用于控制完成一组微操作。

微程序：一序列微指令构成的有序集合。

每一条机器指令都对应于一段微程序，通过解释执行这段微程序，完成指令所规定的操作。

微指令周期：微程序控制器的工作周期。从控制存储器读取一条微指令到执行完相应的微操作所需时间的最大值。

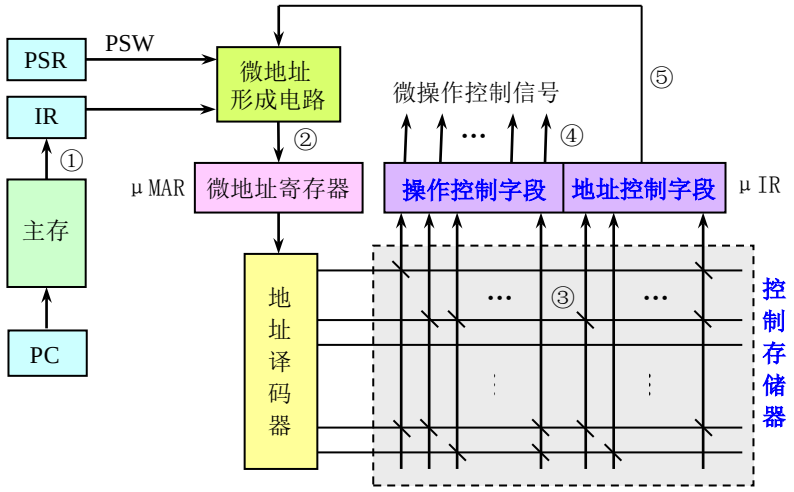
关系如下：



(二)微程序控制器的组成与工作过程

组成：

1. 控制存储器 CM
   简称控存，用于存放实现整个指令系统的所有微程序，其中每个单元存放一个微指令字。
2. 微指令寄存器 μIR
   用来存放从控存 CM 读出来的当前微指令。包含两个字段：操作控制字段、地址控制字段。
3. 微地址形成电路
   该电路根据控制地址字段中的信息产生后续微地址。
4. 微地址寄存器 μMAR
   接受微地址形成电路送来的地址，为读取下一条微指令做好准备。
5. 地址译码器
   将 μMAR 中的微地址进行译码，找到被访问的控存单元，将其中的微指令读出并存放于 μIR 中。



工作过程（假设其微程序已经在控存中准备好）：

1. 启动“取址”微指令，把要执行的机器指令（地址由 PC 给出）从主存取到指令寄存器 IR 中。然后 PC 进行增量操作。
2. 根据 IR 中指令的操作码，微地址形成电路产生指令的微程序入口地址，送入 μMAR。
3. μMAR 中的微地址经过译码，从控存中读取相应的微指令，送入 μIR。
4. μIR 中微指令的操作控制字段直接（或经过译码）产生一组微命令，送往相应的功能部件，控制他们完成所规定的微操作。
5. 微地址形成电路根据 μIR 中微指令的地址控制字段和机器的状态信息，产生下一条微指令地址并送往 μMAR。
6. 重复 3 到 5 步骤，直到机器指令的微程序全部执行完毕。

(三)微程序设计技术

1. 微指令的编码方法

一条微指令由两部分组成：微操作控制字段、地址控制字段。

编码方法有 4 种：

(1) 直接控制编码

微操作控制字段中每一位直接对应一个微操作。1 即为执行，0 为不执行。

优点是结构简单，并行性最好，操作速度快；缺点是微指令字太长。

(2) 最短字长编码

将微命令进行统一的二进制编码，每条指令只定义一个微操作。

微操作控制字段的长度 L 与微命令的总数 N 的关系是：$L \geq log_2 N$。

优点是微指令字长最短，缺点是要经过译码，速度和并行性差。

(3) 分段直接编码

把微操作控制字段进一步划分为若干个字段，每个字段单独编码，每个码点表示一个微命令。

上面两种方法的折中方案。

可以按功能和部件划分，对于机器中的每一种功能类型或每一个部件，分配一个字段；

把互斥的微操作分在同一字段，把相容的微操作分到不同的字段；

(4) 分段间接编码

字段的编码的含义（即表示什么微命令）要由另外一个字段的编码来解释确定。

一个解释字段要同时对多个字段进行控制（解释），才能有效地缩短字长。

解释字段应有某些分类的特征。

2. 微指令格式

(1) 水平型微指令

微指令字较长，一般几十位甚至上百位。描述并行微操作的能力强，译码简单。一般采用直接控制编码法或分段直接控制编码法。

(2) 垂直型微指令

微指令字短，一般一二十位，但并行微操作能力差，一条微指令只能控制数据通路的一两种信息传递。各个二进制位与数据通路的各个控制点之间完全不存在直接对应关系。

3. 微程序的顺序控制

初始微地址：机器指令所对应的微程序的入口地址。

形成下一条微指令的地址（称为后继微地址）。

(1) 入口地址的形成

“取指令”微程序一般存放在控制存储器中 0 号单元，是固定不变的。根据 IR 中的操作码，找到该指令所对用的微程序的入口地址。（实际上是一个从操作码到入口地址的映像问题）

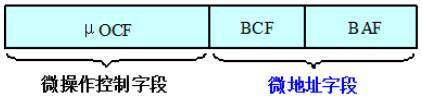
实现方法：**直接对应法**或**查表法**。

(2) 后继微地址的形成

两种方式：**增量方式**和**断定方式**。

增量方式就是设置一个微程序计数器 μPC。顺序执行的时候，给 μPC 增加一个增量（通常为 1），给出下一条微指令的地址。遇到转移（如 JMP 指令）时，由微指令给出转移目标的微地址。

采用这种方式的微指令格式：



微地址字段 SCF 分为 BCF（转移控制字段）和 BAF（转移地址字段）。

断定方式按以下方式确定后继微地址：由微程序设计者指定 或 由微程序设计者指定的测试判别逻辑字段控制产生。后继微地址由两部分组成：**非测试地址（设计时直接指定不变的）**和**测试地址（由条件状态决定）**。

- **微地址格式**

测试地址的位数决定了并行分支的路数，而且也决定了测试控制字段的个数。

例如：当测试地址位数为 m 时，分支的路数为 2m，而测试字段的个数为 m。至于测试字段的位数 n，则是取决于测试条件的个数 N，一般地，有 n= [log2N]+1。