第 10 章 控制单元的设计（简单学习）

10.1 组合逻辑设计

一、组合逻辑控制单元框图

1.外特性

如图 1.1 所示，CU 要发出各种控制命令，这些信号是由先后关系，每个控制信号发出都是有给定时间点，这种时间点是由节拍控制，所以要有**节拍发生器**。CU 要产生什么控制命令与现在执行命令有关系，当前命令保存在 **IR** 中，其 n 位操作码经过**操作码译码**后才能确定正在执行是哪条指令。译码结果针对每一条指令是一一对应的，输入到 CU 中。CU 发出什么信号还和标志有关系，所以输入还有一系列**标志**。CU 在以上作用下，有**输出 C0……Ck**。C0……Ck 可以有 1 个或多个高电平，多个高电平有两种含义：要么是它们并行，要么是他们在一个周期，但是它们每一个操作都不够一个周期。

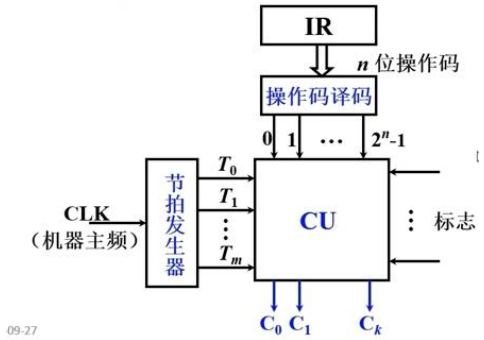

图 1.1 组合逻辑电路外特性

二、微操作的节拍安排

（1）采用**同步控制**方式

假设一个机器周期有 3 个节拍（时钟周期），CPU 采用非总线方式。

1.安排微操作时序的原则

原则一：微操作的先后顺序**不得**随意更改。

原则二：**被控对象不同**的微操作（可以并行执行的），尽量**安排在一个节拍**内完成。

原则三：占用**时间较短**的微操作，尽量安排在**一个节拍**内完成，并**允许**有先后顺序。

2.取指周期微操作的节拍安排

T0	PC->MAR, 1->R	原则二
T1	M(MAR)->MDR, (PC)+1->PC	原则二
T2	MDR->Ad(IR), OP(IR)->ID	原则三

3.间址周期微操作的节拍安排

T0	Ad(IR)->MAR, 1->R
T1	M(MAR)->MDR
T2	MDR->Ad(IR)

3.执行周期微操作的节拍安排

①非访存指令

1）清除累加器指令 CLA

该指令在执行周期只有一个微操作，按同步控制的原则，此操作可安排在 T0-T2 的任一节拍内，其余节拍空。如在 T2 周期内安排（T2: 0->AC）。

2）累加器取反指令 COM

同理，按同步控制的原则，此操作可安排在 T0-T2 的任一节拍。

3）算术右移一位指令 SHR

同理，按同步控制的原则，此操作可安排在 T0-T2 的任一节拍。如在 T2 周期内安排（L(AC)->R(AC), AC0-AC0）。

4）循环左移一位指令 CSL

同理，按同步控制的原则，此操作可安排在 T0-T2 的任一节拍。如在 T2 周期内安排（R(AC)->L(AC), AC0->ACn）。

5）停机指令 STP

T0	
T1	
T2	0->G

（2）访存指令

1）加法指令 ADD X

T0	Ad(IR)->MAR,1->R
T1	M(MAR)->MDR
T2	(AC)+(MDR)->AC

2）存数指令 STA X

T0	Ad(IR)->MAR,1->W
T1	AC->MDR
T2	MDR->M(MAR)

3）取数指令 LDA X

T0	Ad(IR)->MAR, 1->R
T1	AC->MDR
T2	MDR->M(MAR)

（3）转移类指令

1）无条件转移指令 JMP X

T0	
T1	
T2	Ad(IR)->PC

2）有条件转移（负则转）指令 BAN X

T0	
T1	
T2	$A_0 \cdot Ad(IR) + \bar{A}_0(PC) \rightarrow PC$

4.中断周期微操作的节拍安排

在**执行周期的最后时刻**，CPU 要向所有中断源发终端查询信号，若检测到某个中断源**有请求**，并且**未被屏蔽**又被**排队选中**，则在**允许中断**的条件下，CPU 进入中断周期，此时 CPU 由**中断隐指令**完成下列操作（假设程序断点存入主存 0 号地址单元内）：

T0	0->MAR, 1->W（关中断，可以放在这里）
T1	PC->MDR（PC 中保存着断点）
T2	MDR->M(MAR),向量地址->PC

CPU 进入中断周期，由硬件置“0”允许中断触发器 EINT，即关中断。

三、组合逻辑按设计步骤

1.列出操作时间表；

2.写出微操作命令的最简逻辑表达式；

3.画出微操作命令的逻辑图。

这种方法有以下特点：

（1）思路清晰，简单明了；

（2）庞杂，调试困难，修改困难；

（3）速度快。

用在精简中（RISC），通常情况下整型单元也是用这个。

10.2 微程序设计

一、微程序设计思想

一条机器指令对应一个微程序。完成一条机器指令有多个微操作命令，每几个微操作命令组成一个微指令，这些微指令称为微程序。

二、微程序控制单元框图及工作原理

1.机器指令对应的微程序

采用微程序设计方法设计控制单元的过程就是编写每一条机器指令的微程序，它是按照每条机器指令所需的微操作命令的先后顺序而编写的，因此，一条机器指令对应一个微程序，如图 2.1 所示。图中每一条机器指令都与一个以操作性质命名的微程序对应。

由于任何一条机器指令的取指令操作是相同的，因此将取指令操作的命令统一编成一个微程序，这个微程序只负责将指令从主存单元中取出送至指令寄存器中，如图 2.1 所示的取指周期微程序。此外，如果指令是间接寻址，其操作也是可以预测的，也可先编出对应间址周期的微程序。当出现中断时，中断隐指令所需完成的操作可由一个对应中断周期的微程序控制完成。这样，控制存储器中的微程序个数应为机器指令数再加上对应取指、间址和中断周期的 3 个微程序。

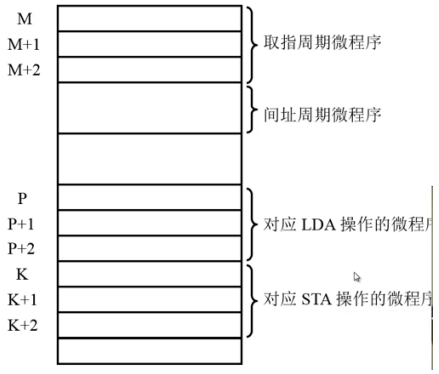

图 2.1 不同机器指令对应的微程序

2.微程序控制单元的基本框图

如图 2.2 所示，图中点画线框内为微程序控制单元，控制存储器（控存）是微程序控制单元的核心部件，用来存放全部微程序；CMAR 是控存地址寄存器，用来存放欲读出的微指令地址；CMDR 是控存数据寄存器，用来存放从控存读出的微指令；顺序逻辑是用来控制微

指令序列的，具体就是控制形成下一条微指令（即后续微指令）的地址，其输入与位地址形成部件（与指令寄存器相连）、微指令的下地址（简称“**下地址**”）字段以及外来的标志有关。

微指令基本格式如图 2.2 下所示，共分为两个字段，一个为操作控制字段，该字段发出各种控制信号；另一个为顺序控制字段，它可以指出下条微指令的地址，以控制微指令序列的执行顺序。

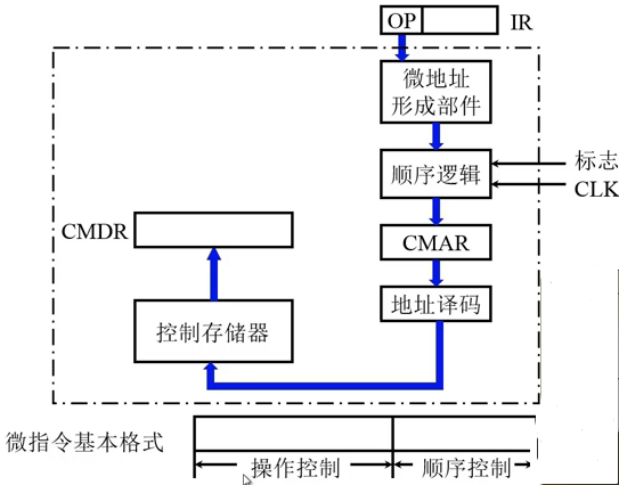

图 2.2 微程序控制单元的基本组成

3.工作原理

假设一个用户程序如下所示，它存于以 2000H 为首地址的主存空间内：

LDA X

ADD Y

STA Z

STP

首先将用户程序的首地址送至 PC，然后进入取指阶段。

（1）取指阶段

①将取指周期微程序首地址 M->CMAR；

②然后取微指令，将对应控存 M 地址单元中的第一条微指令读到控存数据寄存器中，记作 CM(CMAR)->CMDR。

③产生微操作命令，第一条微指令的操作控制字段中为“1”的各位发出控制信号，如 PC->MAR，1->R，命令主存接收程序首地址并进行读操作。

④形成下一条微指令的地址。下一条微指令地址为 M+1，将 M+1 送至 CMAR，即 Ad(CMDR)->CMAR。

⑤取下一条指令。将对应控存 M+1 地址单元中的第二条微指令读到 CMDR 中，即 CM(CMAR)->CMDR。

⑥产生微操作命令。

由第二条微指令的操作控制字段中对应“1”的各位发出控制信号，如 M(MAR)->MDR 使对应主存 2000H 地址单元中的地一条机器指令从主存中读出送至 MDR 中。

⑦形成下一条微指令地址。

（2）执行阶段（**执行 LDA 微程序**）

①取数指令微程序首地址的形成。

OP(IR)->微地址形成部件->CMAR。

②取微指令。CM(CMAR)->CMDR。

③产生微操作命令。Ad(IR)->MAR,1->R。

④形成下一条为指令地址。Ad(CMDR)->CMAR。

⑤取微指令，CM(CMAR)->CMDR。

⑥产生微操作命令。

……

☆关键点：

（1）微指令的操作控制字段如何形成微操作命令；

（2）微指令的后续地址如何形成。

三、微指令的编码方式（控制方式）

1.直接编码（直接控制）方式

在微指令的操作控制字段中，**每一位代表一个微操作命令**。如下所示，下地址左侧为控制信号，进行操作控制。**速度最快**。

2.字段直接编码方式

将微指令的操作控制字段分成若干**段**，将一组**互斥**的微操作命令放在一个字段内，通过对这个字段译码，便可以对应每一个微命令。这种方法靠字段**直接译码**发出微命令，又称**显式编码**。它**缩短**了微指令**字长**，**增加**了译码**时间**。

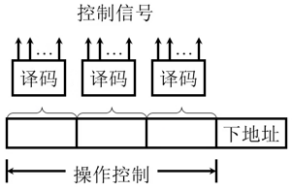

图 2.3 字段直接编码方式

3.字段间接编码方式

一个字段的某些微命令还需要由另外一个字段的某些微命令来解释，如图 2.4 所示，图中字段 1 译码的某些输出受字段 2 译码输出的控制，由于不少靠字段直接译码发出微命令，故称为字段间接编码，又称隐式编码。如图 2.4 所示。

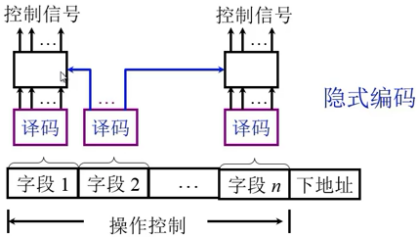

图 2.4 字段间接编码方式

4.混合编码

把直接编码和字段编码（直接或间接）混合使用。

5.其它

微指令中还可以设置常数字段，用来提供常数、计数器初值等。常数字段还可以和某些解释位配合。

此外微指令还可以用类似机器指令操作码的方式编码。

四、微指令序列地址的形成

1.直接由微指令的**下地址**字段指出（断定方式）；

2.根据机器指令的**操作码**形成；

3.增量计数器法；

4.分支转移；

5.通过测试网络形成；

6.由硬件产生微程序入口地址。

五、微指令格式

1.水平型微指令

一次能定义并执行多个并行操作。如直接编码、字段直接编码、字段间接编码、直接和字段混合编码等。

2.垂直型微指令

类似机器指令操作码的方式。在微指令字中，设置微操作码字段，由微操作码规定微指令的功能。不强调其并行控制能力。

3.两种微指令格式的比较

（1）水平型微指令比垂直型微指令**并行操作能力强，灵活性强**。

（2）水平微指令执行一条机器指令所需的微指令**数目少，速度快**。

（3）水平型微指令用**较短的微程序结构换取较长的微指令结构**，而垂直型微指令正好相反。

（4）水平型微指令与机器指令**差别较大**，而垂直型微指令与机器指令相似。

六、静态微程序设计和动态微程序设计

静态：微程序无需改变，采用 **ROM**。

动态：通过改变**微指令**和**微程序**改变机器指令，有利于仿真，采用 **EPROM**。

七、毫微程序设计

微程序设计用**微程序**解释**机器指令**；**毫微程序设计**用**毫微程序**解释**微指令**。

八、串行微程序控制和并行微程序控制

微程序指令也分为两个阶段：取微指令和执行微指令。串行微程序控制如下所示：

取第 i 条微指令	执行第 i 条微指令	取第 i+1 条微指令	执行第 i+1 条微指令

并行微程序控制增加了一个微指令寄存器来暂存下一条指令，以防止影响本条微指令的执行。如下所示：

取第 i 条微指令	执行第 i 条微指令		
	取第 i+1 条微指令	执行第 i+1 条微指令	
		取第 i+2 条微指令	执行第 i+2 条微指令

由于执行本条微指令与取下一条微指令是同时进行的，因此当遇到需要根据本条微指令的处理结果来决定下条微指令的地址时，就不能并行操作，此时可以延迟一个微指令周期再取微指令。