目　　录

第七章　微程序控制

第一节　概　　述

一、微程序控制的基本原理

一般计算机的控制方式是组合逻辑的，而微程序控制计算机是存储逻辑的。

一般计算机的设计过程，是在指令的功能确定之后，确定结构部件，门及通道的设置，排操作表，综合简化操作表，用布尔组合逻辑实现控制。

微程序计算机除了确定结构部件、门及通道之外，要根据机器指令系统中的每条指令功能设计微指令编码（只读存储器横向信息位的设计）及编制微程序（只读存储器纵向信息的设计）。控制的关键部件是以只读存储器为中心的微程序控制器。只读存储器的周期应与运算周期相匹配。由于全机的控制信息以二进制编码的形式存储于只读存储器中，所以称微程序计算机是存储逻辑的。

在微程序计算机内，我们将那些专管信息控制门的打开与关闭的逻辑命令称作微命令。微命令所实现的操作称作微操作。

在一个主脉冲周期中，按予定要求完成控制操作任务的互相配合的一组微命令便构成一条微指令。一条微指令被存放在只读存储器的一个单元里。实现机器指令（宏指令）操作功能的微指令序列称为该机器指令的微程序（如定点加法微程序）。

微程序设计，大体上可以说是把传统的程序设计方法引用到逻辑设计中来的一种逻辑设计方法。为了实现一条机器指令既定的操作功能，逻辑设计者可以选用适当的微命令以设计一组微指令，并顺序把它们组织起来。这样，执行了这一组微指令，就是实现了它对应的机器指令的操作功能了。把每条机器指令所对应的微指令序列及某些处理过程所对应的微指令序列集合起来，便是这台机器的全部微程序。

微程序计算机的工作方式可通过图7—1的简单模型予以描述。

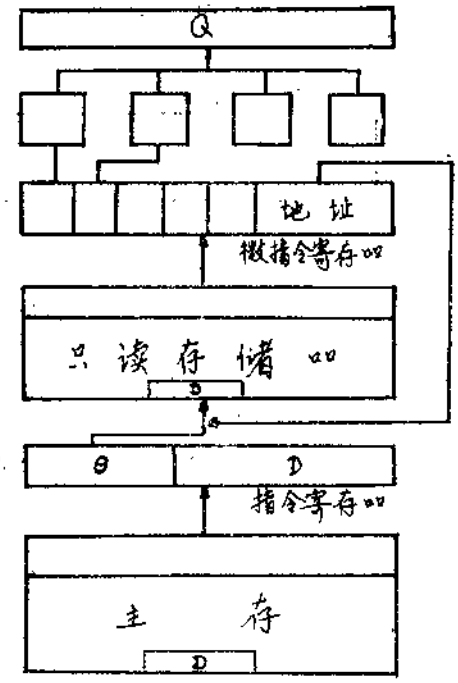


图 7—1　微程序计算机的简单模型

假定只读存储器周期等于运算周期。从主

存取指令到指令寄存器，指令寄存器的操作码内容打入到只读存储器的地址寄存器中去，经过一个只读存储周期取出微指令，送到微指令寄存器去，微指令的每一位（或几位编码）控制确定的一个（或一组）门或执行某一动作。同时在微指令中指出下一微指令地址，把此地址再打入到只读存储器的地址寄存器中去。这样顺序地控制，当一个微程序执行完毕后，再由指令寄存器新的内容去控制执行新的指令所对应的另一个微程序。这就是微程序计算机的工作过程。

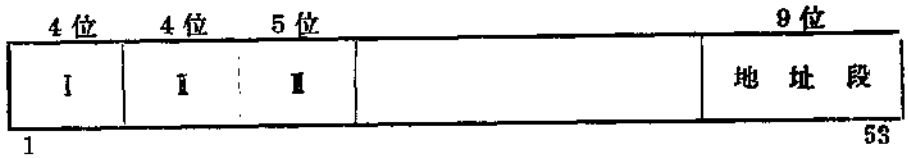
DJS－154 的微程序总共由 350 条微指令所构成。每条微指令是在总数为 168 左右的微命令系统中选择适当的微命令构成的。整个微程序存放在容量为 512 个单元、周期为 600nS 的只读存储器之中。只读存储器的每个单元存放一条微指令，每条微指令字长 53 位，可同时实现十种不同的微操作。

DJS－154 的微指令结构，微程序的执行方式，机器典型指令的微程序流程图及微程序，是我们下面所要学习的主要内容。

二、154 机的微指令结构

154 机微指令由 53 位二进制代码组成，采用分段编码结构。微指令字的长度 53 位被分或十段，其中九段为控制段，一段为地址。控制段中的不同码点形成打开不同信息通路的微命令。在操作上互相排斥的不同微命令被编制在同一控制段内；在时间上要求同时出现的互相配合的诸微命令被分配在同一微指令不同的控制段中。微指令的地址段则给出微程序条件转移的转移地址。

154 机微指令字各段的长度及主要功能如表 7－1 所示。



154 机微指令各段长度及功能　　表 7－1

段　号	长度（位）	微命令的主要功能
1	4	全加器输出
2	4	全加器左输入
8	5	全加器右输入及 CPU 和主存数据送 I/O 母线
4	5	寄存器接收脉冲及 I/O 控制
5	5	全加器输出送主存及 D 寄存器接收脉冲

续表 7－1

段　　号	长度（位）	微　命　令　的　主　要　功　能
6	5	置特征
7	5	微地址转移控制
8	4	微地址转移控制（浮点）
9	4	全加器右输入（浮点）
地　　址	9	微程序转移地址码

注：第17、34、53位为校验位。

三、主要微命令含义

+A
+C
+ZJ
+δ
⋮

将对应寄存器内容通过全加器Q输入门送往全加器。

mA
mC
mZJ
mδ
⋮

对应寄存器的接收脉冲。

$+Q_i$：全加器结果通过输出门直接输出。

$+2Q_i$：全加器结果通过输出门左斜一位输出。

$+Q/2_i$：全加器结果通过输出门右斜一位输出。

$Q\rightarrow ZC_i$：全加器内容送往主存储器。

$W\theta_{1i}$：判别指令特征的微命令（属于无条件转移性质），转移方向由机器指令特征给出。其功能表达式如下：

$W\theta_1 \cdot (C_{0\sim3} \neq 0) \cdot \lambda \cdot \delta' \longrightarrow$ 给出基本指令的寻址微程序入口地址。

$W\theta_1 \cdot (C_{0\sim3} = 0) \cdot C_4 \cdot \theta \longrightarrow$ 给出外部指令微程序入口地址。

$W\theta_1 \cdot (C_{0\sim4} = 0) \cdot C_5 \cdot \theta \longrightarrow$ 给出寄存器操作指令微程序入口地址。

$W\theta_1 \cdot (C_{0\sim5} = 0) \cdot C_6 \cdot \theta \longrightarrow$ 给出条件及特征指令微程序入口地址。

$W\theta_1 \cdot (C_{0\sim6} = 0) \cdot C_7 \longrightarrow$ 给出长指令公操作微程序入口地址700。

$W\theta_2$：识别基本指令的微命令（属于无条件转移性质），它和基本指令的操作码 θ 一起给出相应操作的微程序入口（即基本指令操作码给出转移方向）。其功能表达式为：

$W\theta_2 \cdot \theta \longrightarrow$ 给出相应操作的微程序入口地址。

$W\theta_3$：识别长指令微命令，它和长指令的操作码 θ 一起给出相应操作的微程序入口，其功能表达式为：

$W\theta_3 \cdot \theta \longrightarrow$ 给出相应操作的微程序入口地址。

$0 \rightarrow D$：无条件返回“取指令”微程序段的微命令。

J：禁止只读缓冲寄存器内的微指令执行的微命令。

第二节　154机微程序的执行方式

从硬件角度讲，154机微指令的执行方式也就是只读存贮器的工作方式。其工作方式可归纳如下：

一、微指令的重叠操作

微指令可采用逐条读取——执行——再读取——再执行的工作方式，这样主机的效率就得不到充分发挥，以至降低了机器的速度。154机采用了微指令重叠执行方式，即执行本条微指令的同时便读取下条微指令，这样主机一直在连续执行微指令，使机器效率可充分发挥出来。这两种工作方式可比较如下：

(a) | 读微指令1 | 执行微指令1 | 读微指令2 | 执行微指令2 |……

(b) | 读微指令1 | 执行微指令1 |
　　| 读微指令2 | 执行微指令2 |
　　| 读微指令3 | 执行微指令3 |

为此，机器中设置了一个长度为53位的由R—S触发器构成的微指令缓冲寄存器，微指令读出后首先放在该缓冲寄存器。待上一微指令执行周期末尾再送到微指令执行寄存器被执行。

二、微程序的顺序执行

一般情况下，微程序都是顺序执行的，并且执行每条微指令的时间是600ns。只要微地址不断加1就可获得新的微指令。

微地址寄存器具有计数功能，每个主脉冲使其计数加1。

综上所述，主机的每个主脉冲完成下列操作：

1. 执行一条微指令。
2. 将下条微指令由缓冲寄存器送到执行寄存器。
3. 只读存贮器的地址寄存器加“1”。
4. 启动只读存贮器。

例：

执行寄存器存放着101，

缓冲寄存器存放着102，

只读地址寄存器内容为102。

当主脉冲 m_{i+1} 到来时：

(1) 执行101，

(2) 缓冲寄存器内容送执行寄存器，即(102)⟶执行寄存器，

(3) 只读地址寄存器加“1”，即102+1=103，

(4) 启动只读存贮器。

上述过程可参见图7—2。

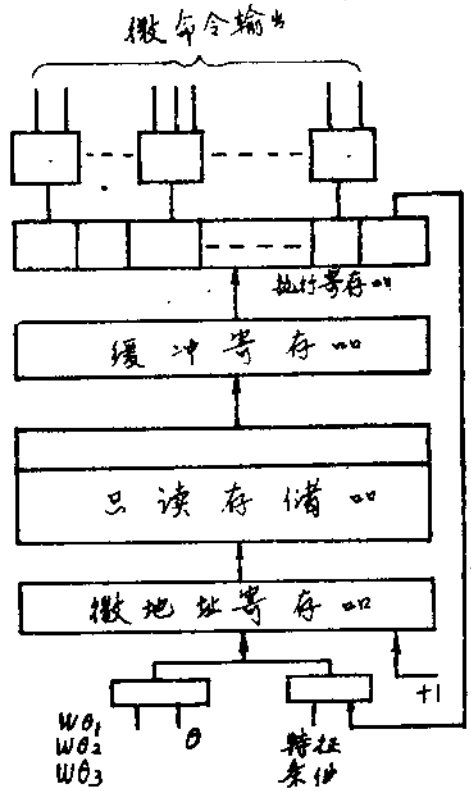


图 7—2　微程序的执行

三、微程序的转移

微程序和传统的程序设计一样有无条件转移和条件移转。

无条件转移：在154机微命令系统中设有无条件转移微命令，如 $W\theta_1$、$W\theta_2$、$W\theta_3$。这三个微命令与操作码组合，形成各指令的微程序入口地址码电位，从而实现不同指令的微指令序列分支，进而实现机器的各指令功能。

条件转移（特征转移）：在微程序执行过程中，往往需要根据某些特征决定微程序转移方向（如浮点运算中需要根据运算结果确定向左规格化还是向右规格化）。条件转移的地址码由该微指令的地址段给出，一旦条件满足时就实现转移，即将地址段内容打入只读地址寄

存器，取相应的微指令。否则，仍顺序执行下一条微指令。可以看出，每条微指令仅能实现一种特征转移。

另有一种情况，即当微指令需跳过一条执行时，在 154 微命令系统中设有相应的微命令 J。当发出微命令 J 时，则禁止下一条微命令由缓冲寄存器向执行寄存器传送，机器执行一次“空操作”。此时，微地址寄存器仍由计数输入控制。值得指出的是：由于重叠操作，伴随着转移性质微命令的发出，必须要发出 J 微命令。

四、微程序循环

在 154 机的定点、浮点的乘除法中，在浮点加减法的对阶和左规中，在移位操作中，都有重复执行某一条或几条微指令的情况，因此产生了微程序的循环问题。为了控制微程序的循环，机器中设置了定点踏步触发器 C_{Tb}，浮点踏步触发器 C_{ω}，微指令循环计数器 JS。在微程序循环前将循环次数的补码送到计数器，循环开始前一周期的操作脉冲将 C_{Tb} 置“1”，并将微地址加“1”（发一个 m_{zdd}），但由踏步触发器控制不再启动只读存贮器。这样微指令缓冲寄存器内容只能反复送到执行寄存器被执行，实现该条微指令的循环。每执行一次，计数器加“1”，在循环结束前一周期置“0” C_{Tb}（C_{Tb} 为 J—K 触发器），同时启动只读，然后再重复执行一次，计数器加“1”变全“0”（此时 $C_{Tb}=0$），循环结束。接着微程序仍顺序执行。

在浮点及双字长运算中，需要一条以上的微指令循环。此时不能停止启动只读存贮器，只需封锁微地址寄存器进位。图 7—3 表示了微指令循环波形图。

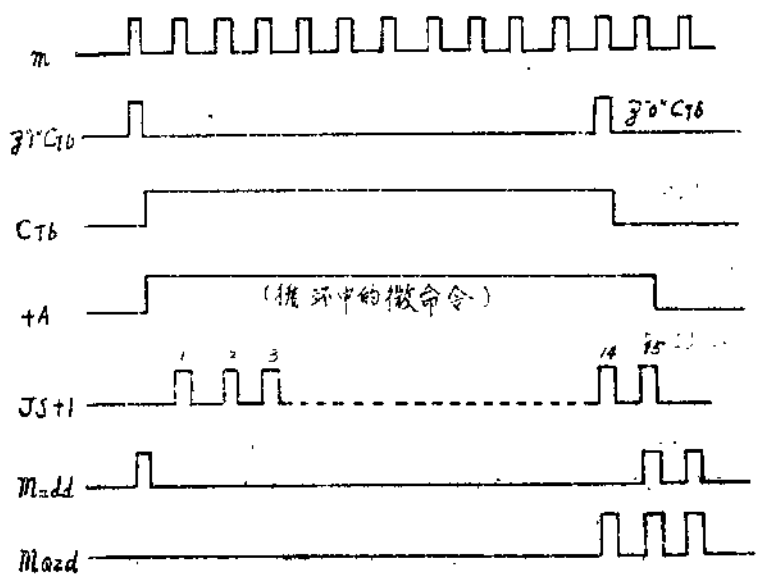


图 7—3 微指令循环波形

第三节　微程序转移的实现——微地址的置入

我们知道计算机的解题过程就是执行机器指令序列的过程。而每条机器指令的执行过程，在微程序控制的计算机内又是执行微指令序列（即微程序）的过程。为使结构简单和设计方便，通常情况下，微指令是顺序执行的。但是，每当执行完一条机器指令的微程序而进入另一条机器指令的微程序时，必然要产生微程序转移。为此，在机器的微命令系统内设有实现转移的微命令（如 $W\theta_1$，$W\theta_2$，$W\theta_3$，$0\rightarrow D$，……等），而且存储微程序的只读存储器的地址寄存器兼有计数和接收功能。

微程序转移的实现，就是在具有转移性质的微命令控制下，根据每条机器指令的操作码，将该指令所对应的微程序首地址置入到只读地址寄存器 J_{zdd} 再启动只读，这样就实现了微程序转移。

某些机器指令的微程序中存在着条件或特征转移。这种情况下，一旦条件成熟，就由形成的条件或特征控制，将正在执行的微指令地址段的内容置入 J_{zdd}，实现微程序转移（见图 7－2）。

下面根据 154 机的指令系统在只读存储器的具体分配情况，给出已介绍过的五类指令的微地址置入方法及框图。

一、基本指令微地址置入

1．寻址微地址置入

基本指令有两位变址特征码 λ，故有 λ＝00（直接地址型），λ＝01（直接数型），λ＝10（相对地址型），λ＝11（间接地址型）四种编址方式。根据 λ 的不同，由 $W\theta_1 \cdot C_{0\sim3}\neq 0\cdot\lambda\cdot\delta'$ 首先给出基本指令寻址微地址入口，置只读地址寄存器 J_{zdd} 为：

0	0	0	0	1	0	λ_1	λ_2	$\lambda_1\cdot\lambda_2\delta'$

2．执行相应操作的微地址置入

由识别基本指令的微命令 $W\theta_2$ 与操作码 θ 一起给出相应操作的微程序入口地址，即置 J_{zdd} 为：

0	1	1	×	×	×	×	0	0

其中 × × × × 为基本指令的操作码 θ。

但除法例外，即当操作码 θ＝0101（除法）时，J_{zdd} 寄存器前三位置为 001，即 J_{zdd} 置为：

0	0	1	×	×	×	×	0	0

二、外部指令微地址置入

由 $W\theta_1 \cdot C_{0\sim3}=0 \cdot C_4 \cdot \theta$ 给出外部指令微程序入口地址，置 J_{zdd} 寄存器为：

0	1	0	0	×	×	×	0	0

其中 × × × 为外部指令操作码 θ 。

三、寄存器操作指令的微程序地址置入

由 $W\theta_1 \cdot C_{0\sim4}=0 \cdot C_5 \cdot \theta$ 给出寄存器操作指令的微程序入口，置只读存贮器 J_{zdd} 为：

0	0	1	×	×	×	×	0	0

其中 × × × × 为寄存器操作指令操作码 θ 。

这类指令前八条为移位指令，后八条为其它指令。若操作码 $\theta_1=0$（即前八条指令），我们给它一个公共入口地址 100，在这个公共入口里，再根据移位性质、移位方向等区分八条指令。

四、条件及特征指令微地址置入

由 $W\theta_1 \cdot C_{0\sim5}=0 \cdot C_6 \cdot \theta$ 给出条件及特征指令的微程序入口，置 J_{zdd} 为：

0	0	0	1	×	×	×	0	0

其中 × × × 为条件及特征指令操作码 θ 的后三位。

五、长指令微地址置入

1. 由 $W\theta_1 \cdot C_{0\sim6}=0 \cdot C_7$ 首先给出长指令公共操作的微程序入口 700，置 J_{zdd} 为：

1	1	1	0	0	0	0	0	0

2. 由 $W\theta_8 \cdot \theta$ 给出相应操作的微程序入口地址，置 J_{zdd} 为：

1	0	0	×	×	×	×	0	0

其中××××为长指令操作码θ。

对长指令来说，第一次是取指令的前半段。因此须给出一个公共操作，在公共操作内完成以下工作：

1. 取指令后半段；
2. 区分是否广义指令；
3. 计算操作地址，取操作数。

其中操作数也是顺序分两次取出的。

六、指令类型判别及其微地址置入框图

综上所述，我们可给出 154 机指令类型判别及其微地址置入的框图如图 7－4。

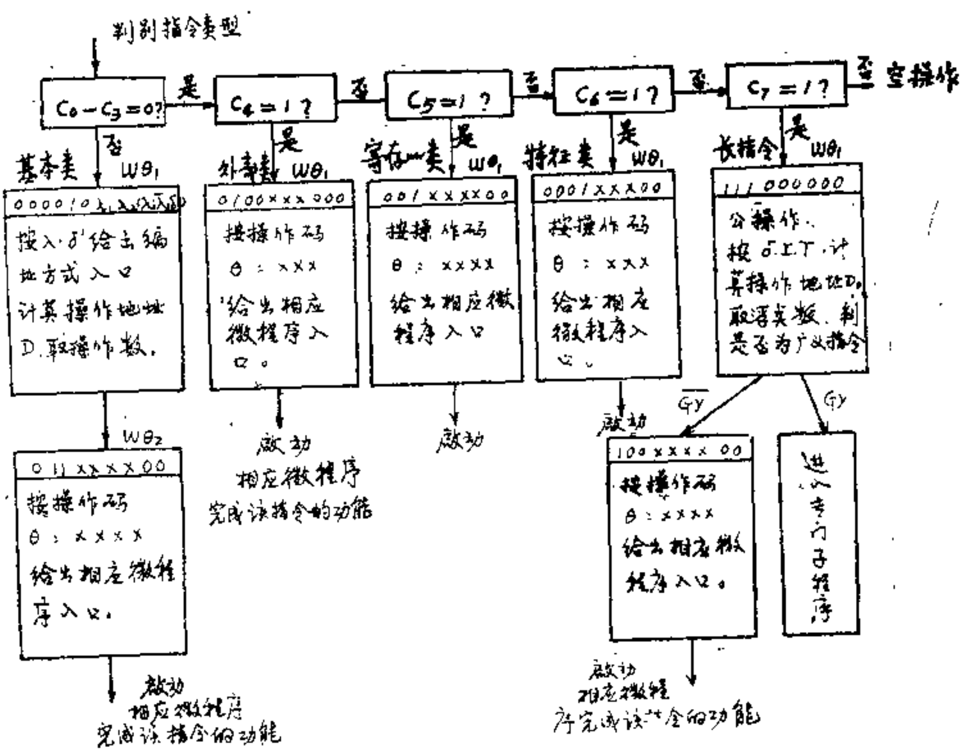


图 7—4　154机指令类型判别及其微地址置入框图

第四节　典型指令的微程序流程

一台计算机的微程序设计，要在确定了结构部件、数据通路的基础上，根据机器指令系统中每条指令的逻辑功能、运算方法，进行寄存器分配，画出微程序流程，并逐步地建立机器的微命令系统，设计微指令编码，进行只读地址分配……等一系列步骤才能完成。

我们已经介绍了 154 机的结构部件、数据通路及指令系统和运算方法。在这一节内我们给出 154 机典型指令的微程序流程，为进一步了解 154 机的微指令及微程序打下基础。

一、基本指令流程

1. 基本指令微程序总框图

154 机的 15 条基本指令（指令码从 0001 到 1111）的执行过程是相似的。首先通过 000～002 三条微指令取出现行指令（这一步是其它各类指令也都要经过的）；然后根据指令中的 λ 和 δ′ 进入不同的寻址微程序入口地址，计算出操作地址，据此取操作数；最后根据微命令 $W\theta_2$ 和操作码 θ 进入执行该基本指令功能的微程序入口地址，完成指定操作任务。基本指令微程序总框图见图 7－5。

2. 典型举例

下面我们以乘法、除法和带返转为例给出它们的微程序框图，说明基本指令的执行过程。框图从进入执行该基本指令功能的微程序入口地址开始，“取指令”和“寻址”部分不再列入。其它基本指令的微程序框图大家可自行绘制。

二、外部指令流程

执行外部指令时，首先根据 θ 进入不同的微地址入口；然后判别设备号，若为某一确定的设备号（非 77），则执行指定的输入、输出操作；若设备号为 77，则执行中央处理机内部的一组专用操作。例如，当 θ＝001 时，完成将指定设备的缓冲寄存器 I 的内容送 A 寄存器的操作，或者在设备号为 77 时，完成“读开关”的操作，即将面板上所设置的开关数经母线送内存数码寄存器，最后送入 A 寄存器。其微程序框图如 7－7 所示。

三、寄存器操作类指令流程

寄存器操作类指令的共同特点是不访问内存。其中前八条指令都是移位指令，有一个公共的微地址入口，不同的移位性质（A 单移或 A 与 B 联合移、算术移或逻辑移、左移或右移）由操作码对应位决定。其微程序框图见图 7－8。

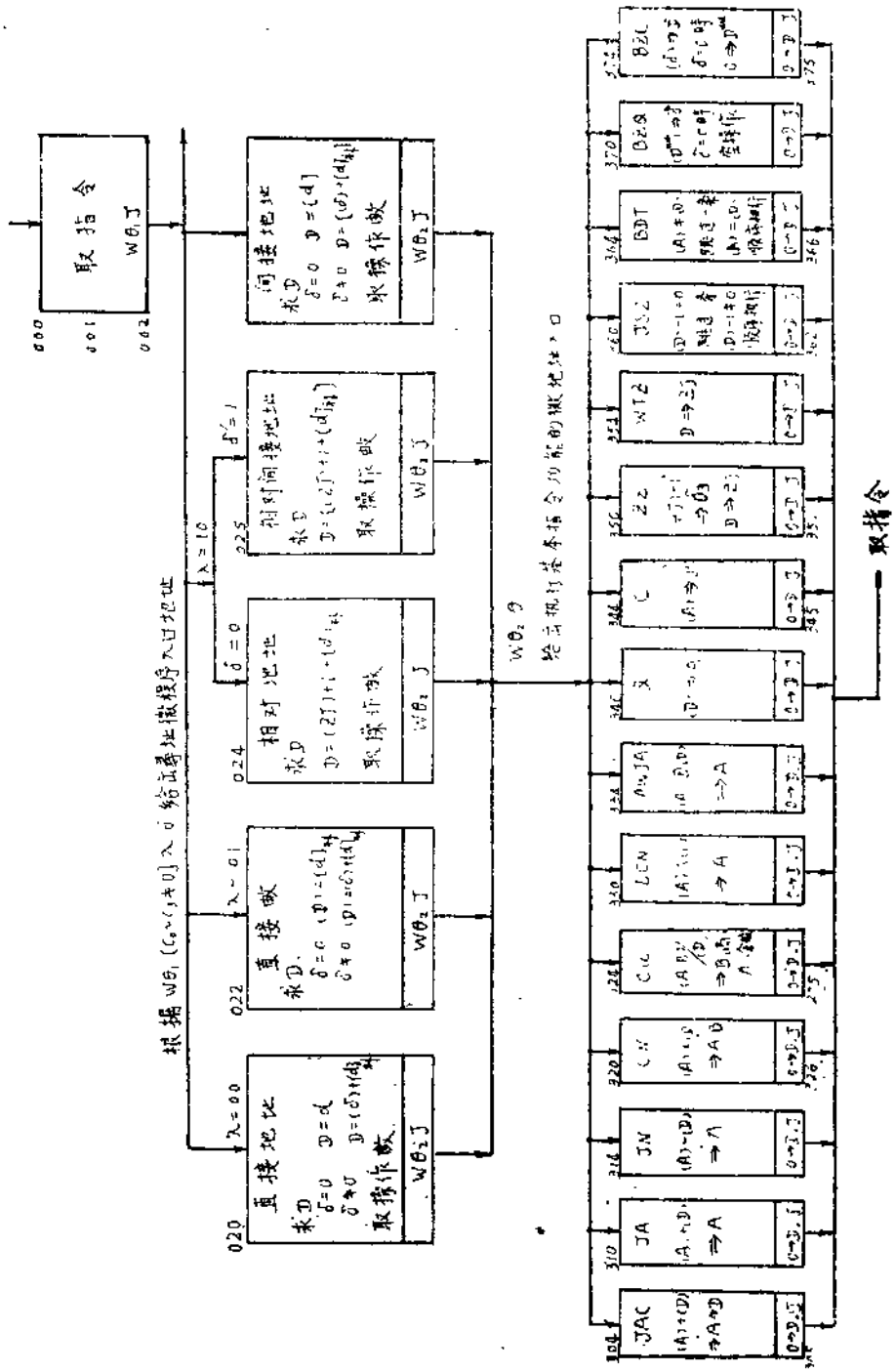
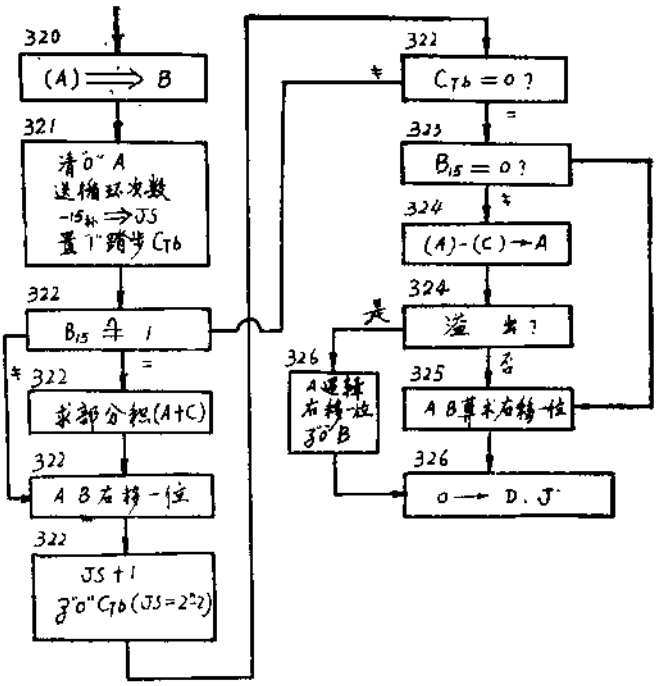
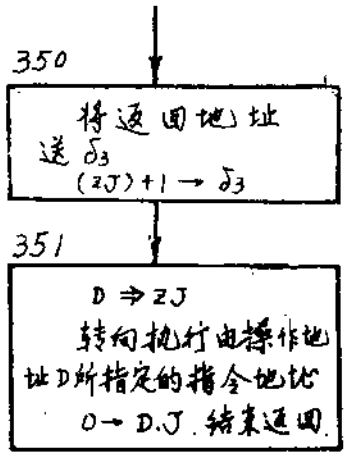


图 7—5　基本指令微程序总框图

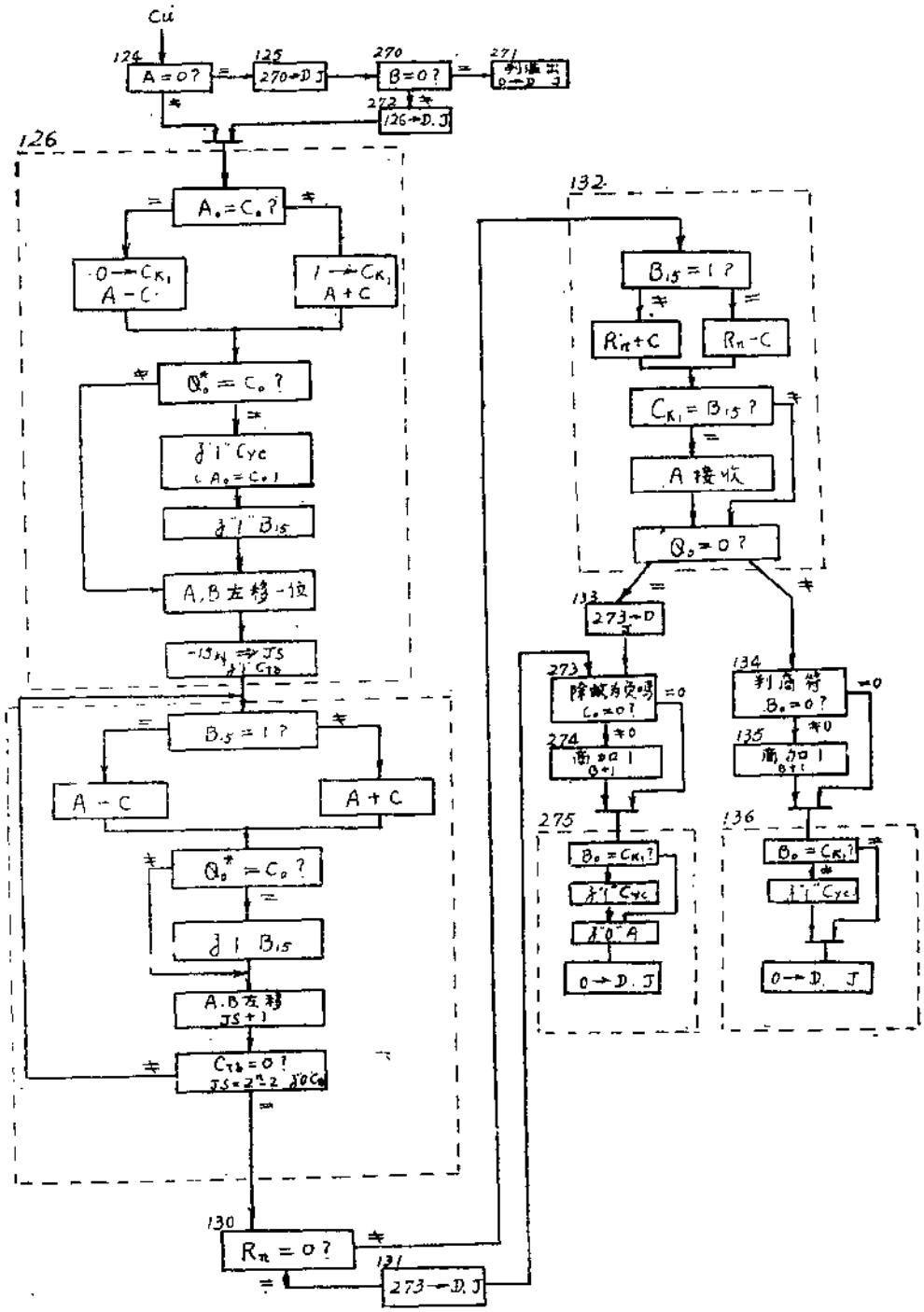


(a) 乘法



(b) 带返转

图 7—6 基本指令的微程序框图典型示例（1）



(c) 除法

图 7—6　基本指令的微程序框图典型示例(2)

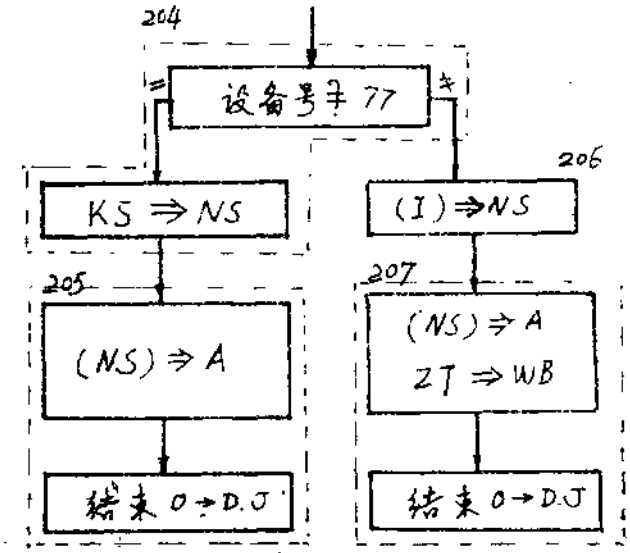


图 7—7 外部指令微程序框图（ $\theta=001$ ）

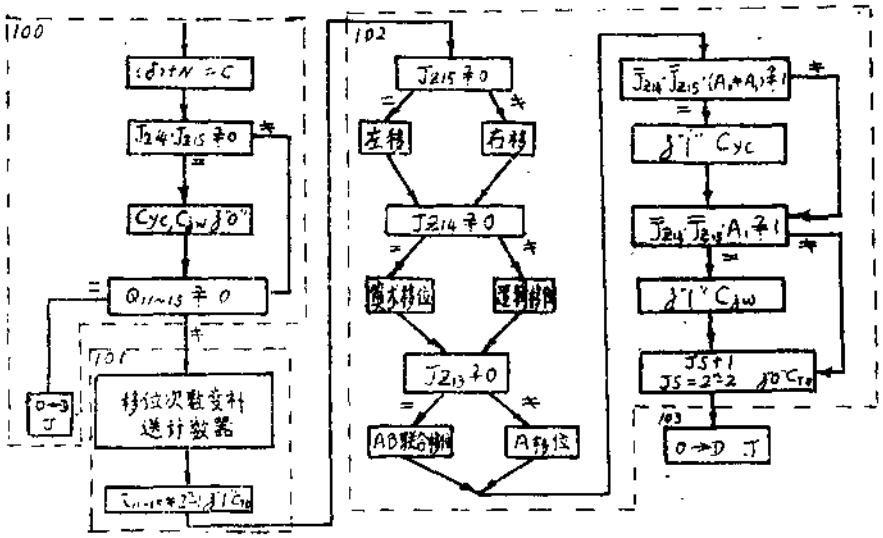


图 7—8 寄存器操作类指令微程序框图（前八条）

四、条件及特征类指令流程

这类指令，根据 $C_{0\sim3}=0\cdot C_8\cdot W\theta_1\cdot\theta$ 进入不同的微程序入口地址，完成指定的跳跃、加 1 或存取特征的操作。以真跳和假跳指令为例，其微程序框图如图 7－9。

五、长指令流程

长指令包括广义指令和非广义指令两大类。对于广义指令，它相当于一条间接地址型的转子指令，对应于某一广义指令，有一专门子程序完成该指令功能。对于非广义指令，其中有定点双字长运算指令，浮点运算指令和十进制运算指令。在长指令中，无论进入那种指令操作以前，都有继续取出长指令的形式地址，计算操作地址，取操作数，判别是否为广义指令等公操作。为此设计的长指令公操作框图如图 7－10 所示。在长指令公操作中，若判定为广义指令，则给出转向指定子程序首址和保存返回地址的微操序流程。

长指令的执行过程比较复杂，其微程序段比较长。为了弄清长指令的执行过程，下面我们给出浮点运算的三种指令（加减、乘、除），定点双字长运算的一种指令（双除）及十进制运算的两种指令（十进加和无符号十进加）的微程序框图，作为典型举例，见图 7－11～7－15。

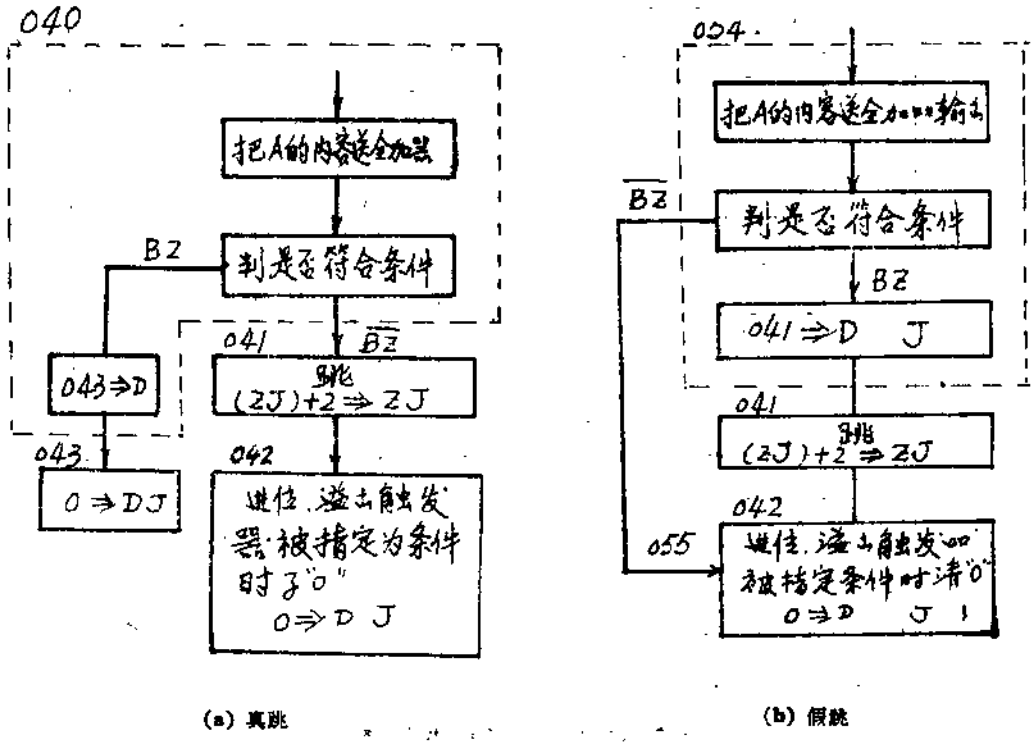


图 7—9　条件及特征类指令的微程序框图