第六章　控制器

§6·1· 概述

在工业控制计算机中，控制器担负着指挥各部件协调工作的控制中心，因此应保证：(1)自动地把计算程序和原始数据输入存储器。(2)自动与存储器交换讯息。(3)控制直接执行指令。(4)控制向存储器送运算结果。(5)控制过程输入输出讯息的交换等。

控制器分类：同步控制、异步控制、联合控制、微程序控制。

同步控制是机器执行各种指令操作均按同样节拍长短构成，这就是说，指令操作简单的和复杂的均在同样节拍下完成。

异步控制与同步控制相反它是根据每种指令操作自己完成节拍长短而构成，需要几拍就几拍，完成以后转下一条指令操作。因此可知异步控制速度快，但统一性较差。

联合控制是介于同步控制与异步控制之间的控制方式，它将大多数指令统一在相同数目的节拍内完成，而将个别需要节拍长的指令操作另行处理。通常分中央操作控制和局部操作控制，中央操作控制担负着大多数相同节拍的指令操作控制，局部操作控制完成特殊节拍长指令操作控制。

微程序控制器是由微指令组成。微指令一般细微到相当于过去由一个节拍控制完成的基本动作，象移一位、发送、相加和清除等等。它使控制器更加条理化、规律化，使控制器简化。

下面章节中考虑重点叙述联合控制器，因为用的较多。

无论那一种控制器一般都应以下列几种工作方式：

(1) 自动工作方式—— 机器正常工作时连续执行操作。

(2) 周期工作方式—— 机器执行一个指令操作周期后停下来，以备检查机器用。

(3) 单脉冲工作方式—— 也称单拍工作，更进一步检查机器执行情况。

因此，在设计时应注意单调，与联调互相配合方便，单调用于运控部件检查，因此也需具备有单脉冲工作，周期工作，连续工作及变

频率工作等方式。联调也应具有上述几种工作性能，才更进一步检查出运控与内存及输入输出之间联接时发生的问题。

控制屏也称调机屏，供单调时用，一般各大部件均设有上面设有指示灯及开关等，以完成单调与联调几种工作方式的完成。

控制台，则是控制全机的台，当各大部件调好后，整机总调时，应在控制台进行操作整机调试，当全机调好后，控制台成为人机联系的工具，它能方便的完成人对机器要求的各种操作，并且设有必要的指示灯和开关等，下面章节中准备结合具体机器的控制屏及控制台加以介紹，以更进一步了解其作用。

§6·2· 指令系统

在叙述控制器之前，应将指令系统介紹一下，因为控制器就是执行各种指令操作的部件。

指令就是指定的操作命令，规定好之后机器就按规定的操作步骤去执行。

指令系统就是若干条指令的集合称之为系统。指令系统包括下列几种类型：

1. 算术运算指令：也称四则运算指令

一般包括加、减、乘、除基本指令以及其它一些有关四则运算指令，这一些随着各种机器不同而不同。

2. 绝对值运算指令：|A|-|B| 指令及 |A-B| 指令

3. 邏輯运算指令：

常用的有邏輯乘、邏輯加、按位加。

4. 傳送指令：

常用的有取数、存数等指令，有磁鼓时有访鼓、写鼓等指令。

5. 控制指令：

常用的有无条件转移指令，条件转移指令，全等指令，停机指令，变址指令及頁面控制指令等。

6. 输入输出指令：

包括输入指令如：数据输入指令、光电机输入指令、时间输入指

令、控制台输入指令等等。

输出指令，如：制表指令、显示指令、输出通道指令、采样指令等。

7. 中断指令：

处理中断系统的指令。

8. 特殊指令：如送标志触发器状态指令等等。

9. 其它：有时设有左、右移指令等。

上述为常用的一些指令类型，由于各种机器有各种不同的指令，因此指令系统是一个变化的系统，可根据设计者方便情况而进行选择。一般设计指令系统时，有两种设计思想：其一是尽可能提高每条指令的功能；其二是每条指令只有一种功能，而以几条指令组合来构成多种有机的、综合能力强的高功能指令。如果用第一种方法，由于每条指令的功能提高，所以用较少的执行步数，就可组成高效率的程序。可是，计算机内部的操作就会复杂，其复杂程度与功能的高低成正比，对于指令的灵活性限制也较大，有时用较少的指令也可以实现操作，但编制程序过于费力，且不灵活。如果用第二种方法，编制程序灵活、通用性强。

因此，选择指令系统的原则是希望程序方便灵活，通用性强，且花费的硬设备少。

在国内外一些机器中，我们可以看出其指令系统有这样的情形。一种尽量少的指令以便减少硬设备，如arch－1000指令少10几条，一种是多的指令，如TQ-1机，还有介于二者之间的指令系统，其灵活性及硬件均处于合适的地位。下面准备以JDK—331机指令系统为例来说明各种指令的用途和操作过程。

§6·3· 字的形式

所谓字就是指令和数。

数的形式：

符　号	尾　　数

指令的形式根据各种机器不同而异，一般有这样几种形式：

1 °基本形：单地址

操作码	地　　址

操作码为实现指令操作的二进制代码，操作码位数愈多，可表示指令数目愈多。

后面部分为地址部分，表示对应该操作码的地址。

例如：操作码为加法操作，对应地址为128，就表示运算器中累加器中原来的数，与内存储器中第128单元地址中存的数相加。

2 °变址形：以JDK—331机为例

J_c	J_λ	J_b	J_d
操作码	变址特征	变址地址	形式地址

例如：设 J_c 5位，J_λ 1位，J_b 3位，J_d 9位

i) 当 $J_\lambda = 0$ 时

不变址，则真地址 $J_D = J_d$

ii) 当 $J_\lambda = 1$ 时

变址，则真地址 $J_D = (J_b) + J_d$

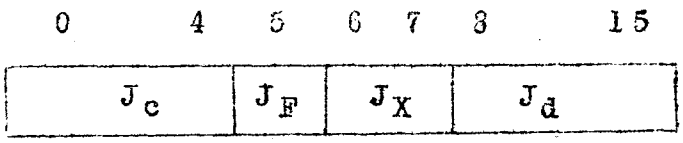
这就是说，当变址特征位 $J_\lambda = 0$ 时，表示这条指令是不变址的，即成为基本形，其真地址 $J_D = J_d$ 形式地址，而 J_b、J_λ 部分不起地址作用。当 $J_\lambda = 1$ 时，表示这条指令是变址指令，即成为变址形，这时真地址 $J_D = (J_b) + J_d$ 就是真正地址为变址地址 J_b 中的内容加上形式地址，若 J_b 有三位，则对应存储器中有8个变址单元，每个单元都有一个内容，其后面部分为地址部分，这部分与形式地址相加而得到真地址，从这也可以看到变址形实质上是变化地址的意思。

3 °1½地址形：以FACOM270—10为例：

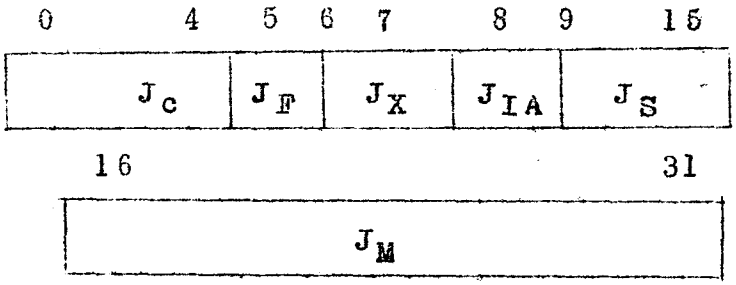
1½地址形就是即有1字形式的指令又有2字形式的指令。

1字形式为相对地址指令，2字形式为绝对地址指令及间接地址指令。

1字形式指令
（短字）



2字形式指令
（长字）



J_c 为操作码设为5位

J_F 字形式设置1位

J_F=0 表示1字形式指令

J_F=1 表示2字形式指令

J_X ：变址地址（同上J_b）设2位

J_d ：相对地址（形式地址）

J_s ：在2字形式应用，用于转移指令的条件指令，其它情形应该空着，即为“0”。

J_M ：用于存储器的地址（没有变址寄存器的设置时，为绝对地址）设置。只有2字形式时应用。

J_{IA}：（1位）用于间接地址设置

J_{IA}=1 时即为间接地址指令，只有2字时应用，内容加上J_d表示的数值。当J_X≠00时，则真地址J_D等于三个变址寄存器（J_{XR_1}—J_{XR_3}）中的内容加上J_d表示的数值。

2字形式指令的情形：当J_X=00 时，真地址J_D=J_M

当J_X≠00 时，真地址J_D等于三个变址寄存器（J_{XR_1}—J_{XR_3}）中的内容加上J_M所表示的数值。

当J_{IA}=1 时，即为间接地址指令，真地址J_D等于（J_M）中的内容所表示的地址，或者为{（J_{XR}）+J_M}地址中的内容所表示的地址。

4 °二地址形

J_C	J_{d_1}	J_{d_2}

J_C——操作码，J_{d_1}——第一地址
J_{d_2}——第二地址

操作方法是，第一地址 J_{d_1} 中的内容与第二地址 J_{d_2} 中的内容操作。例如：加法操作指令，表示第一地址 J_{d_1} 中的内容与第二地址 J_{d_2} 中的内容相加。

5 °三地址形

J_C	J_{d_1}	J_{d_2}	J_{d_3}

操作方法是第一地址 J_{d_1} 的内容与第二地址 J_{d_2} 中的内容依操作指令进行操作，并把操作结果送到第三地址 J_{d_3} 中。

在工业控制计算机中，大多数均采用单地址或 1 ½ 地址，很少采用二地址和三地址所以下面叙述也着重叙述单地址机器，同时再介紹一下 1 ½ 地址。而二地址及三地址用于大型通用数字计算机这里就不叙述了。

§6·4· 变址形指令系统

具有变址功能的指令系统，其各指令操作内容以 JDK—331 机为例，列于下表：

上述指令系统有几点这里要说明一下：

(1) 变址操作：

目前大多数机器都采用变址技术，这是因为采用变址技术对程序设计带来了方便、灵活，甚至于可以节省存储单元。变址操作一般完成下列动作：(1)地址修改；(2)变址运算；(3)“起——止”判断等。

地址修改就是将变址寄存器（在内存中，也可以另设专门变址寄存器）的变址值加到指令的地址部分（即形式地址），地址的和作为访问存储器的真地址。

变址运算就是变址寄存器中的内容（即变址值）的运算，往往是变址寄存器中事先存放有初始变址值供地址修改用，但是这个初始变址值也是需要变化的，通常采用一种简单的叠代操作方法，将一个增量与旧的变址值相加得到一个新的变址数。且这个增量一般为“1”，这就是说每进行一次变址运算，变址值＋1，一直进行下去，进行到何时为止呢？因此要进行终端判断，判断到终端时，需将变址寄存器的内容恢复到初始变址值，这个过程即为“起——止”判断过程。

终端判断每进行一次变址运算均需判断一次是否到达终端了，如果未到达则仍然转回再进行变址程序的执行，如果到达终端了，则转下条另外的程序去。

终端判断的方法有：比数、减数、计数等。

比数判断就是现行的变址值与一个给定的常数作比较。减数判断就是某一给定的变数逐次被增量值减去，并且测试是否为“0”，为“0”时则到达终端。

计数判断就是给定的变数每次减去1，并测试是否为“0”，为“0”时则到达终端。

由上述可知，变址操作一般均由几条指令组成，如TQ-1机变址组由四条指令构成：

变址Ⅰ（修改并计数）——地址修改及终端判断

变址Ⅱ（修改）——地址修改

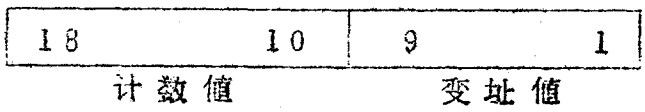
变址Ⅲ（修改，计数并转移）——地址修改、终端判断，在结束时转出。

变址傳送—— 恢复初始变址值。

JDK—331机则将上述变址动作用一条指令即变转指令完成。变转即完成地址修改，终端判断并转移动作。此机中未设变傳指令，也即恢复初始变址值，因为这个指令可用取数、存数指令代替，所以省略了。

(2) 控制字操作：

配合变址操作，产生了控制字操作问题，上述变址操作中谈到了变址寄存器问题，实际上变址寄存器中的内容即存放的控制字，这个控制字是将变址值和计数值分别放在一个字的两部分，其控制字的形式如下：



变址值部分存放初始变址值及变址运算后的变址值。

计数值部分用来终端判断，每变址运算一次将初始计数值－1，如果初始计数值为100，则变址运算100次后计数值部分为零，可判断出终端到达了。

JDK—331机，终端判断的方法稍稍有些不同，其不同是采用补码＋1方法，即是计数值9位，最大计数范围可以表示512，如果变址运算次数也为100次则计数值部分事先存放413即可。

控制字除了用在控制变址外，尚有下列用途：

数据傳送：

（a） 将两个存储区（或者頁面）中的数据互相傳送；

（b） 输入数据到存储器；

（c） 由存储器输出数据。

一次操作所傳送的数据称为一个“记录字段”。

数据排列：

数据排列的一般过程是将记录字段从这一存贮区域移到另一存贮区域。有了控制字以后，就可以将包括许多数据字的记录的傳送操作用一个控制字的傳送操作代替。这个控制字是规定该记录用的。这是

什么意思呢？这个意思就是将控制字中的内容互换一下，在执行指令时本来是执行这一区域内容，现在变为执行另一区域内容了。

假设有记录字段A……Z都存储在整个存储器中，相连的控制字A—a……Z—z按照字的排列次序排列。现在要从序列中抽出记录字段H，而将其存储区域空着。这个记录字段的控制字是C—c……K—k的一部分。将单元d与单元h中的控制字对掉后，H就不再是这个序列的一部分了。d和h再对掉后H又重新插入。这就说明了插入和抽出的互换性。如下列表所示：

	单元	控制字	单元	控制字	单元	控制字
	b	C—c	b	C—c		
	e	D—d	c	D—d		
*	d	H—h	d	E—e		
*	h	E—e			h	H—h
	e	F—f	e	F—f		
	f	G—g	f	G—g		
	g	I—i	⋮			
	i	J—j	⋮			
	j	K—k	⋮			
	抽出前		抽出后		被抽出	

由*所示，本来未抽出前执行d单元时，控制字指出下面执行h单元，抽出后，执行以后指出执行e单元，则将h空过。

Z程序控制：将上面叙述的记录字段作为Z程序，就可以控制了。

(3) 頁面控制：

在操作存储器较大时，有时分成頁面，所谓頁面，就是将一个存储器分成许多块（区域）例如：容量为8192，按每512为一个頁面，则可分为16个頁面。

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

分頁面的好处是：①在字长一定时，便指令数多，且能满足地址数要求。例如：18位字长时内存地址8192单元，不用頁面时，则一条指令地址部分占13位，只有5位是操作码部分。

18 14	13 1
操 作 码	地 址

而5頁面时，如512单元为一頁面，则地址部分仅9位即可，其余部分可作为操作码和变址用。

18 54	13	12 10	9 1
操作码	变址转征	变址	地址

而頁面更换，则由頁面控制指令（頁转指令）实现之。

② 采用頁面控制当工作在某一个頁面时，由于某种原因使该頁面程序冲乱，不会影响其它頁面的内容，因此从这种意义上来说，頁面控制有保护存储器内容之作用。

任何事情总是一分为二的，采用頁面控制也引起了一些缺点，主要是使程序编制不方便。

(4) 中断指令：

在控制计算机中由于和巡测及输出通道连接，因此设有中断系统（详細后述），配合这一操作，设有处理中断的指令。中断指令一般完成下列内容：将被中断的指令程序（指令计数器之内容）保存起来以便建立返回用，其次应保留被中断时的中间结果。在JDK-331机中，前一功能用一条中断指令实现，后一功能则用存数指令完成。

§ 6 · 5 · 操作表

操作表是具体的实现运算、控制和操作的表，因而若设计或者熟悉一台机器，应当首先把操作表设计出来。

为了清楚起见，这里举联合控制器的操作表为例，且叙述一下整机各部分的信息傳送关系，这样对加深理解操作表是有好处的。这里以JDK—331机的计算机部分为例，来说明各寄存器间信息的傳送，其框图如下：

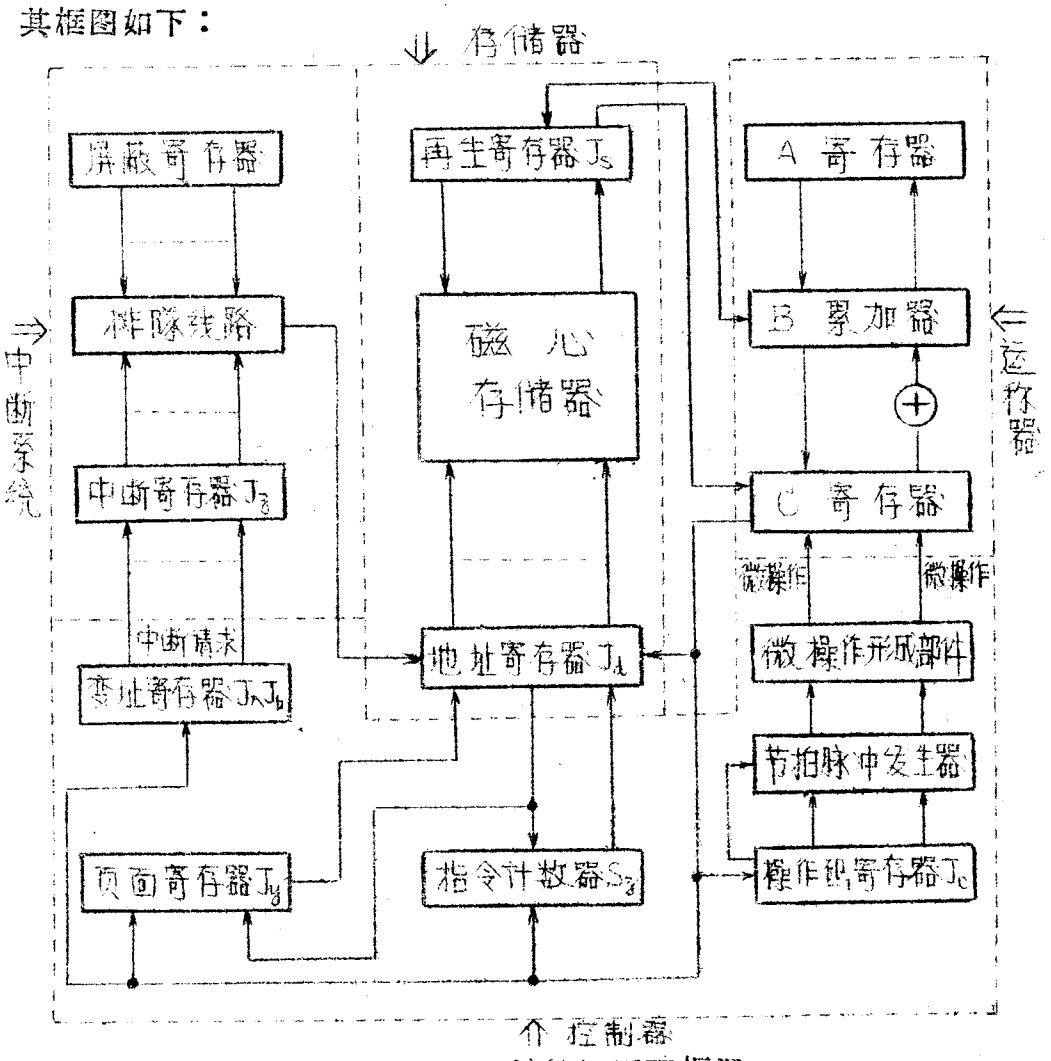


图 6 · 1　计算机原理框图

由上图知：计算机主要组成部分运算器、存储器、控制器及中断系统。运算器包括A、B、C三个寄存器，进行加、减、乘、除及其它逻辑判断；存储器包括磁心体，地址寄存器J_d，再生寄存器及内部的控制线路、驱动一译码线路及读出线路等。地址寄存器标志磁心体的对应单元。访问某一个单元，则对应的地址寄存器即可表示，再生寄存器J_S接受磁心体中读出的内容做寄存，以及写入磁心体内容时也通过再生寄存器，更进一步它还与输入输出诸部件交换信息。控制器包括：

① 节拍发生器，它完成操作节拍u_1……u_{24}；

② 操作码寄存器J_C，它将操作码译出相应的指令，送到微操作形式部件；

③ 微操作形成部件依相应的指令和对应的节拍u_1……u_{24}，发出各种不同的微操作以控制各部件工作。

④ 指令计数器S_Z，它决定要执行的指令序号。

⑤ 变址寄存器，它配合变址用和作为输入输出特征应用。

⑥ 页面寄存器，配合内存页面控制。

运算器与存储器交换信息是通过再生J_S，写入内容时将B累加器中的内容送入再生寄存器，读出时将再生寄存器J_S中的内容送到C寄存器。运算器加、减法以补码运算，乘除法以原码运算，内存以原码形式存储数据，因此运算结果在B中是原码形式。乘、除法指令转局部控制，其它指令均在24节拍内完成。一个基本指令的操作分下列三个步骤：

(1) 公操作节拍u_1——u_8完成取指令工作，就是将现时要执行的指令从存储器中取出来，并将操作码部分送到操作码寄存器，地址部分送到地址寄存器。

(2) 变址操作节拍u_9——u_{16}，当指令中变址特征位$J_\lambda=1$时，则进行变址运算，u_9——u_{16}节拍即是执行变址运算的动作。当不变址（$J_\lambda=0$）时则跳过此八拍。

(3) 执行操作节拍：u_{17}——u_{24}

完成指令操作，如加、减、乘、除等等。

下面根据前节中指令表和本节中所述的一些基本知识以及第四章中所述的基本运算方法，列出操作表：

上表中即为具体执行每条指令的操作过程，由表中知，在计算机中作某一条指令要经过许多小的操作称为微操作来完成，为了帮助理解起见这里举几条指令来说明是如何具体执行的，表中V号者表示要执行，空着不执行。表上面是节拍，接着是微操作。

例如取数指令的执行：

取数指令是将存储器中的数，按照相应的地址 J_D 以原码形式取到运算器B累加器中。

取数指令形式：

18 14	13	12 10	9 1
0 0 0 0 0	J_λ	J_b	J_d

当 $J_\lambda=0$ 时该指令不变址形式为：

18 14	13	12 10	9 1
0 0 0 0 0	0	J_b	J_d

在编制程序时操作码以8进制形式，则不变址取数指令操作码8进制形式为00。

当 $J_\lambda=1$ 时该指令变址，形式为：

	J_λ		
18 14		12 10	9 1
0 0 0 0 0	1	J_b	J_d

操作码8进制形式为01。

我們首先谈不变址取数执行过程：

u_1 节拍：向存储器发出读命令；同时清除再生寄存器 J_S ，目的是 J_S 准备接收存储器的数据；并且把C寄存器清除，目的是准备接收由再生寄存器 J_S 来的内容，在这一拍中读出的是指令。

即然向存储器发出读命令，一定要给出读出的地址，所以在 u_1 节拍中，在无中断情况下（有中断情况以后叙述）将指令计数器 S_Z

的內容，送到地址寄存器 J_{d1-9} 中。

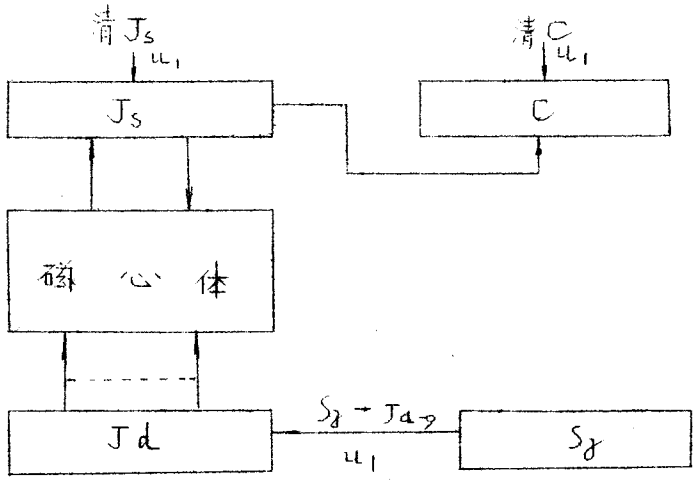


图 6 · 2　u_1 节拍动作图

u_2 节拍：空；u_3 节拍：空，空两拍表示没事情干，由于本机内存取数时间 4μS，这两拍是等待内存读出。

u_4 节拍：将内存读出到再生寄存器中的数送到 C 寄存器中，其微操作为 $J_S \rightarrow C$，同时进行奇偶校验（以后叙述）；另外 + C，目的是变原码时尾数 + 1 用，这 + C 动作是前一指令周期执行时，如果 $C_0 = 1$ 时则 + C 起作用了，否则这 + C 不起作用，因为 u_1 节拍 C 清除了，$C_0 = 1$ 时也正标志着前一指令作完后 B 内容变原码时尾数需加 1（∵补码运算，结果 B 中为补码，补码变原码是将补码变补码，所以需尾数加 1）。

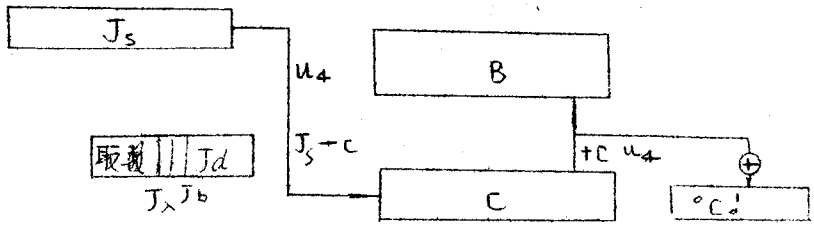


图 6 · 3　u_4 节拍动作图

也就是在 u_4 节拍将取数指令送到C寄存器中，这时C寄存器状态为：

0 0 0 0 0	0	J_b	J_d

u_5 节拍空，空目的等待内存将再生中的数重新写入本单元，一般叫再生回去。

u_6 节拍：将C寄存器中的操作码部分送到操作码寄存器 J_C 变址部分送变址寄存器 J_b ，变址特征送到变址特征寄存器 J_λ 。到此取数指令操作码已译出来。

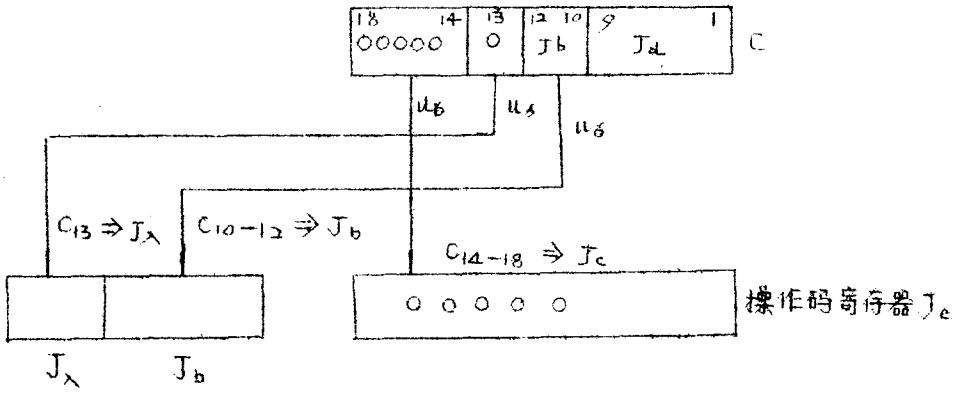


图 6 • 4　u_6 节拍动作图

u_7 节拍：在取数指令时空。

u_8 节拍：将地址寄存器 1 － 9 位清除，即清 J_{d_1-9} 因为原来地址已读出用过了，而新的地址在取数指令过程中已存放在 C_{1-9} 中，所以清除原来的，准备接受新的。

由于设 $J_\lambda=0$ ，所以跳过 u_9 —— u_{16} 节拍。在执行 u_{17} ——u_{24}。

u_{17} 节拍：读命令，同时将新的取数地址送到 J_{d_1-9} ，即 $C_{1-9}\rightarrow J_{d_1-9}$ ，也就是按照这个地址取数。这一拍当 C_{1-9} 送走后C寄存器清除，所以有清C微操作。

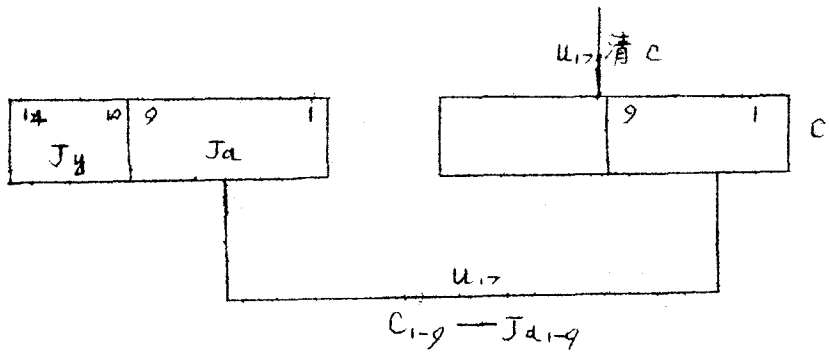


图 6 • 5　u_{17} 节拍动作

u_{18} 节拍：空•

u_{19} 节拍： 清 B，准备接受取出的数•这一节拍中有＋C，这个＋C在取数指令中是没意义的，只是为了配合邏輯设计时简单起见•

u_{20} 节拍：取出的数已读出在再生寄存器 J_S 中，这一 节拍 将 $J_S \rightarrow C$ 中同时进行奇偶校验•

在这一节拍中，将指令计数器 S_Z+1，以准备执行下一条指令。

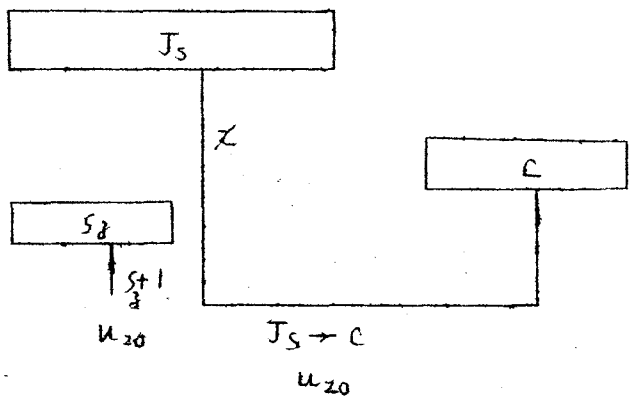


图 6 • 6　u_{20} 节拍动作

u_{21} 节拍：在作取数指令时空•

u_{22} 节拍：将送到C中的数X，送到B寄存器中，由于B中已经清除为“ 0 ”态，故将C的内容加到B中即可，所以这一节拍中有+C动作•这一拍执行完则完成了取数指令的操作，即将数取到B累加器中•

u_{23} 节拍，u_{24} 节拍对于取数指令空。

上面叙述了取数指令的动作，从中可以看到执行一条指令若有许多微操作才能实现。

其余指令均依上述方法一个节拍一个节拍的作，请读者自己化费一点功夫仔细搞清楚。

下面再重点叙述各指令几点不易懂的地方，以配合理解：

存数指令：公操作节拍同取数，在 u_{17} 发出写命令通知存储器将B的内容写入相应地址单元，同时 $B \rightarrow J_S$，就是B的内容送到再生寄存器中。由于内存有奇偶校验，所以在发生写命令时，也发出编码命令，这编码命令就是将写入的内容编成奇数或偶数。

加法指令：它完成将B累加器中的内容与新取出的数相加。

由上面运算方法所述加法是补码进行的。u_1 —— u_8 公操作节拍同上。执行节拍 u_{17} —— u_{24} 中，u_{17} 节拍中有 $B_{18}=1$ 时 B_{1-17} 变反，Co 置“1”微操作，其目的是当B累加器中的数是负数时，将内容变成反码，因为前一次运算结果是以原码方式留于B中，变补码的方法是 B_{1-17} 变反，尾数加1，所以Co 寄存器要置“1”，待 u_{19} 拍相加（+C）时，将这尾数“1”加上。u_{21} 节拍中操作表内写有 C_{18}，上面微操作有 C_{1-17} 变反，Co 置“1”，这意思是 $C_{18}=1$ 时，C_{1-17} 变反，Co 置“1”，目的是 $C_{18}=1$ 表示所取出的加数为负数，所以要变成补码形式，所以要 C_{1-17} 变反，然后在 u_{22} 拍+C时也将尾数“1”加到B中。

u_{23} 节拍中有 $B_{18}=1$，B_{1-17} 变反 Co 置“1”微操作，这目的是将B累加器运算结果若是负数的话变成原码，原码等于补码的补码。而这个Co 置“1”+C 动作是在下一公操作节拍 u_4 进行的。u_{23} 节拍中在表中写有“溢”字，微操作上有 J_Z 溢，这表示溢出微操作，因为定点数作加法时可能有溢出，所以在这一节拍中判断。

减法指令：取出减数与B累加器中的内容（被减数）进行相减，是以补码形式进行，结果以原码留于B中。

公操作节拍 u_1 —— u_8 同上。执行节拍 u_{17-24} 基本同加法指令。所不同的是 u_{19} 节拍中有 C_{18} 置“1”微操作，意思是将C的符