第一下分：DJS——130电子数字计标机指令系统

绪　言

DJS—130集成电路电子数字计标机（以下简称本机）是一台字长为16位的定点小型多功能通用机。全机有四个累加口，即AC_0，AC_1，AC_2和AC_3。其中两个累加口AC_2和AC_3可以作为变址寄存口用。所有的标术和逻辑运标均在累加口里进行。

中央处理机由控制口和运标口组成。它通过存贮口母线与存贮口连接，通过输入—输出母线与外下设备相连接。因此，它控制所有的输入和输出设备。控制存贮口中数据的存贮和传送，以及所有的标术操作，逻辑操作，数据的处理和程序的顺序执行。

本机的磁心存贮口采用平面积木式结构。每块存贮板的容号是4096个字。全机最大容号可达32768个字。

本机所配的外下设备最多可达62种。目前，有纸带输入机，电传输入机，电传输出机和纸带穿孔输出机。

中央处理的全加口字长为二进制16位，从最高位第0位到最低位第15位，自左至右排列。第0位之前还设有一进位位，作为标术运标的进位标志。数的运标在全加口中进行。可视操作数为一个逻辑字，一个地址，或者是两个8位的字符。也可以作为一个16位带符号或不带符号的二进制数处理。标术指令按定点二进数运标。

处理机通过指令计数口PC作为程序执行标志。每执行完一条指令PC加“1”，由此确保程序的顺序执行。改变PC就可以改变程序的执行顺序。

本机采用循环移位法进行移位操作，即左移时进位位移至累加口的第15位，第0位移至进位位；右移时则相反，进位位移

至第0位，第15位移到进位位。

在控制台面板上设有一排16位的数据开关，供操作员参改程序和存贮四的内容。控制台上还设有若干个控制开关，供操作员用来启动程序，或改变累加四的内容；(详见控制台操作说明)。改变后的内容可以在显示灯上看出。

本机所采用的多累加四结构给数据传送或其它运标带来很大的方便，还能节省运标时间。例如：要实现A、B单元的数据交换，若只有一个累加四AC0的话程序应为如下安排：

```
LDA   0,   A;  〔A〕 ⟹ AC0
STA   0,  temp; 〔AC0〕 ⟹ temp
LDA   0,   B;  〔B〕 ⟹ AC0
STA   0,   A;  〔AC0〕 ⟹ A
LDA   0,  temp; 〔temp〕 ⟹ AC0
STA   0,   B;  〔AC0〕 ⟹ B
```

用四个累加四交换，时间可缩减1/3，程序也将大为简化：

```
LDA   1,   A;  〔A〕 ⟹ AC1
LDA   2,   B;  〔B〕 ⟹ AC2
STA   1,   B;  〔AC1〕 ⟹ B
STA   2,   A;  〔AC2〕 ⟹ A
```

本机所采用的是一种组合型指令，虽然只有22条基本操作指令，但其它的辅助操作充满了整个16位，经组合后可达2000多种操作，指令的功能大为增强。

本机不仅可以作为一般的通用数字计标机，还可以作为专用机，也可以与大型机配合进行数据处理和数据交换。

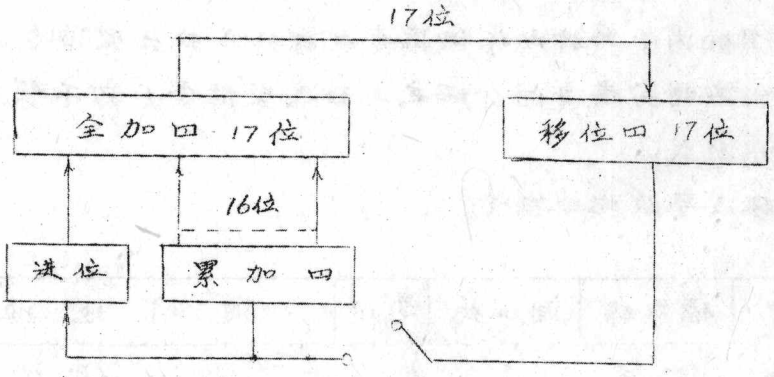
第一章　DJS－130电子数字计算机一般介绍

在介绍指令系统之前，先介绍主机概况，以便对本机有一个基本的了解。

第一节　中央处理机

中央处理机（CPU）包括运算器和控制器。所有的算术和逻辑运算，内存与累加器间的数据传送，外部设备的数据交换以及程序中断等等，均由中央处理机实现。

算术和逻辑运算的基本操作由全加器来完成。全加器将运算结果17位（其中有一位是进位位）输出给移位器。对移位器的输出进行跳步测试。逻辑框图如下：*



移位器每操作一次只能左移或右移一位，或者16位的结果前后八位交换（不包括进位位）。最后由指令的第12位决定是否将运算结果存入结果累加器。以上就是每执行一条算术和逻辑运算指令的基本过程。

---

* 逻辑框图中在移位器方框下边加上跳步测试方框。

第二节　指令形式及指令特点

一．指令类型和形式：

本机指令分三种类型，四种基本形式：

三种类型是：

1．访内型指令：与内存有关的指令通称访内型指令。此类指令包括：传送指令，转移指令及地址参改指令。

2．标术和逻辑型指令：能完成本机所规定的所有标术和逻辑运标的指令通称为标术和逻辑型指令。在进行逻辑运标时，处理机把操作数当作逻辑字来处理。对标术运标而言，则把操作数作为16位无符号的数来处理，也可以看成用补码表示的16位带符号的数。

3．输入、输出型指令：能控制所有外下设备的数据交换，还能完成处理机内下各种动作的指令统称输入输出型指令。

以下我们还将分类专门介绍这三种类型指令，故不赘述。

四种基本形式：

1．转移及参改地址指令

格式

0	0	0	操作码	间址位	变址	相对地址
0	1	2	3 4	5	6	7 8 9 10 11 12 13 14 15

第0位至第2位均为0，第3、4位代表转移或参改的操作。第5位为间址位，表示是直接或间接寻址。第8～15位为以补码形式表示的相对地址。

2．传送指令

格式

0	操作码	AC	间址位	变址	相对地址
0	1 2	3 4	5	6 7	8 15

第1，2位不同（01或10）表示存数或取数，第3、4位表示累加器，后5～15位意义同上。

3．载入载出指令

格式：

011	AC	操作码	控制功能	设备代码
0 1 2	3 4	5 7	8 9	10 15

这种指令第0位为0，第1，2位均为1，第3，4位表示累加器，第5、6、7位表示操作码，8，9位表示设备的控制功能，后10～15位表示设备代码。若第5～7位为全0或全1，则表示一种控制或跳。

4．标术和逻辑指令

格式：

1	ACS 源累加器	ACD 结果累加器	基本 操作码	辅助操作码				
				移位		进位	存否	跳步测试
0	1 2	3 4	5 6 7	8 9	10	11	12	13 14 15

此类指令：第0位为1，第1，2位表示源累加器（AC_0～AC_3均可）的代码，第3、4位为结果累加器（AC_0～AC_3均可）的代码，表示运标结果存入该代码指出的累加器。但是否存入还要取决于该指令的第12位，该位若为0表示运标结果存入结果累加器，若为1则不存。在编程序时以"#"表示该位的记忆符号。例如：ADD# 1，2；〔AC_1〕+〔AC_2〕表示AC_1和AC_2的内容相加但结果不存入AC_2。

二、指令特点：

1．如前所述本机采用的是组合型指令，除了功能性较强的优点外，编制程序还比较简单，单元也比较省。

2．所有访内指令均用三个字母作为记忆符号，标术逻辑指

令也是用三个字母作为基本记忆符号。

3、带符号的数用补码表示。

三、内存地址的分布：

1、从地址的观点看，整个内存贮口是一个连续地址，以8进制 00000 到最大地址是 77777 即为十进制的 32767。

整个内存贮中地址是一个连续的循环地址，即最大地址 77777 的下一个地址是 00000，反之 00000 的前一个地址是 77777。

2、硬设备所确定的专用地址：

0～1 程序中断使用的地址，0地址用作存放中断的退出地址，1地址用作存放中断处理的入口地址。

20～27 自动加1地址，只有在间接寻址时才加1。

30～37 自动减1地址，只有在间接寻址时才减1。

第二章 DJS—130电子数字计算机指令系统

第一节 算术和逻辑型指令

所有的算术和逻辑运算均在累加口之间进行。因此，在运算前操作数必须事先存放在参加操作的累加口里。在进行逻辑运算时，中央处理机把操作数看成逻辑字。对算术运算而言，把操作数作为16位没有符号的二进制数来处理，符号及溢出判断由程序处理。

算术和逻辑型指令的第0位为1，第1，2位为泥累加口，

第3，4位为结果累加口，第5位至第15位为操作功能，其中第5、6、7三位为操作码，第8至第15位表示辅助操作码。基本格式如下：

1	ACS 源累加口	ACD 结果累加口	基本操作码	辅助操作码			
				移位	进位	存否	跳步测试
0	1 2	3 4	5 6 7	8 9	10 11	12	13 14 15

一. 标术和逻辑运林的基本操作指令

由指令的第5～7位决定8种基本操作如下：

1. 求反

(1) 记忆符号 COM

(二) 格式：

1	ACS	ACD	000	移位	进位	存否	跳步
0	1 2	3 4	5 6 7	8 9	10 11	12	13 14 15

(3) 意义：将ACS的内容取反码和所规定的进位C进行8、9所规定的移位操作一并送入移位口。根据13—15位的条件决定是否跳步。最后若第12位为0则将移位口的软出送入进位和结果累加口ACD。

例如：

(1) COM 1,1; $(\overline{AC_1}) \Longrightarrow AC_1$

(2) COM 1,1; SZR; $(\overline{AC_1}) \longrightarrow AC_1$

结果为0则跳过一条。

2. 求补：

(1) 记忆符号：NEG

(2)

1	ACS	ACD	001	移位	进位	存否	跳步
0	1 2	3 4	5 6 7	8 9	10 11	12	13 14 15

（3）忘义：将ACS的内容取补码送入移位口。在第12位为0时，如ACS为0，将C指定的进位值的反码送入进位位。否则就用规定值。然后再根据辅助操作条件执行辅助操作。

（4）例：

NEG 0, 0；将$(\overline{AC_0})+1 \Longrightarrow AC_0$

3. 累加口传送

（1）记忆符号：MOV

（2）格式

1	ACS	ACD	010	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10	11	12 13 14 15

（3）忘义：将ACS的内容和C所规定的进位值送入移位口，然后再根据辅助操作条件执行辅助操作。

（4）例如：

①MOV, 0, 1；将$[AC_0] \Longrightarrow AC_1$

②若判别AC_1的内容是否为0，可用下面任一指令：

MOV 1, 1, SZR；$(AC_1)=0$则跳过下一条

MOV 1, 1, SNR；$(AC_1)\neq 0$则跳过下一条

MOV# 1, 1, SZR；$(AC_1)=0$跳过一条

MOV# 1, 1, SNR；$(AC_1)\neq 0$跳过一条

4. 累加口的内容加“1”

（1）记忆符号INC

（2）格式：

1	ACS	ACD	011	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10	11	12 13 14 15

（3）忘义：将ACS中的数加1送移位口，在12位为0时，如ACS的数为$2^{16}-1$（代符号的-1）则将C指定的值

的反码送入进位值，否则按规定的值传送，然后再按辅助操作条件执行辅助操作。

（4）例：

①INC 1, 1; (AC1) + 1 ⟹ AC1

②INC 1, 1; SZR; (AC1) + 1 ⟹ AC1，结果为0跳。

5、加反码

（1）记忆符号：ADC

（2）格式：

1	ACS	ACD	100	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10 11	12	13 14 15

（3）意义：将ACS中的数取反码与ACD中的数相加，结果放到移位中。在12位为0时，若（ACD）＞（ACS）（无符号）将C指定的值取反送进位位。否则就取进位值本身。再按辅助操作条件执行辅助操作。

对带符号数进位的条件为操作数的符号一致，且ACD的数大或符号相反而ACD中的数为负。

6、减法

（1）记忆符号：SUB

（2）格式：

1	ACS	ACD	101	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10 11	12	13 14 15

（3）意义：将ACS中的数的补码与ACD中的数相加，并将结果送入移位中。在12位为0时，如（ACD）≥（ACS）（无符号）根据C所规定的值取反作为进位值，否则C就是所规定的值本身。然后按照辅助操作条件执行辅助操作。

对有符号的数进位的条件为两操作数的符号一致且（ACD）≥（ACS）或两数异号而ACD中的数为负

（4）例：若要清除累加器 AC_0 则

SUB 0，0；“0”$\Rightarrow AC_0$

此时进位已取反。

若要清除AC_1反清除进位则：

SUBO 1，1；

若只需比较两个累加器的内容是否一致，则：

SUB# 0，1，SNR 即

$[AC_0] \neq [AC_1]$则跳过一条

7．加法

（1）记忆符号：ADD

（2）格式：

1	ACS	ACD	110	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10 11	12	13 14 15

（3）意义：

将ACS和ACD中的数相加，结果送入移位口。在12位为0时，如果不带符号的和≥216，将C所规定的值取反码作为进位值，否则就按所规定的进位值进位。然后按辅助操作码进行辅助操作。

对有符号的数进位的条件是：两个加数都为负，或符号相反而大小相等，或正数较大。

（4）例：ADD 0，1；$(AC_0)+(AC_1) \Rightarrow AC_1$

8．逻辑乘（A）

（1）记忆符号：AND

（2）格式：

1	ACS	ACD	111	移位	进位	存否	跳步
0 1	2 3	4 5	6 7	8 9	10 11	12	13 14 15

（3）意义：

将ACS的字和ACD的字进行逻辑乘，结果送入移位口。按C所规定的值作为进位。然后根据辅助操作条件执行辅助操作。

（4）例：AND 0，1；$[AC_0] \wedge [AC_1] \Longrightarrow AC_1$

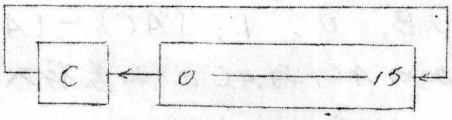
二．标术和逻辑运标的辅助操作：

基本操作码和辅助操作码的任意组合形成一条完整的标术和逻辑指令。基本操作有上述八种，辅助操作由移位，进位，是否存入累加口和跳步测试等四个部分组成，现将各部分的功能分述如下：

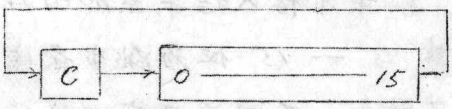
由指令的第8．9位所指示的特征，表示移位操作运标。

记忆 8—9

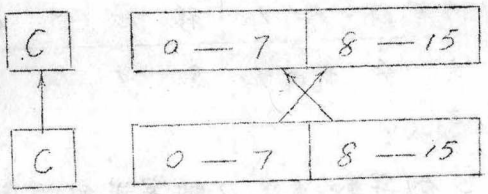
符号	位	移位操作
	0 0	不做
L	0 1	左移一位，第0位移入进位，进位移入15位。



R	1 0	右移一位，15位移到进位，进位移到第0位。



S	1 1	进位不变，16位的前8位和后8位互相交换。



由指令的第 10、11 位确定进位的基本值。所谓进位的基本值是指在标术和逻辑操作前进位位的取值状态，简称进位基值。

根据第 10、11 位的状态进位取值如下：

记忆符号	第 10 ～ 11 位	进位的基本值
	0 0	取进位的现有状态
Z	0 1	进位值取零
O	1 0	取一
C	1 1	取进位现状的反码

指令的第 12 位标志是否将这标结果存入结果累加器 ACD，第 12 位为 0，表示将这标结果存入结果累加器，若为 1，表示这标结果禁止存入结果累加器。其符号为“井”，例如：

SUB 0, 1; $(AC)-(AC_0) \Longrightarrow AC_1$

表示两个累加器 AC_1 与 AC_0 之差存入结果累加器 AC_1

又如：SUB# 0, ISZR; AC_1 与 AC_0 相等则跳，表示仅判断累加器 AC_1 与 AC_0 是否相等，即 $(AC_1)=(AC_0)$ 则跳过下一条指令，结果不存入结果累加器 AC_1。

指令的第 13 — 15 位为跳步条件测试，如满足跳步条件，则跳过下一条指令，否则将顺序执行下一条指令。

跳步条件如下：

记忆符号	第13—15位	跳步条件
	000	不跳，指令顺序执行，
SKP	001	必跳，即无论结果如何均跳过下一条指令。
SZC	010	零进位 跳。
SZC	011	非零进位 跳。
SZR	100	结果为零 跳。
SNR	101	结果非零 跳。
SEZ	110	结果或进位其中任何一个为零均跳。
SBN	111	结果及进位皆为非零跳

这里所指的跳步测试是对移位四的轨出而言，同时，如加法的结果左移一位，利用SZC和SNC实际上是测试和的符号，不论移位四的轨出是否存入结果累加四，测试都是一样的，因此程序可以测试一个标术运标的结果而不破坏原始操作数或进位。

三、标术和逻辑型指令的书写型式及其说明

机四指令（八进制形式）	符号指令	说明
133000	ADD 1,2;	进位基值不变(AC_2)
	;	加(AC_1)结果送C_2入A
133120	ADDZL 1,2;	进位基值取零，(AC_2)
	;	加(AC_1)结果左移一位
	;	送入AC_2。
133100	ADDL 1,2;	与上不同之处在：进位基值，取现有值。
133102	ADDL 1,2 SZC;	运标结果同上，但有跳

步测试，进位为零跳，

133 112　　ADDL #1,2,SZC;　　运标结果不送入AC_2，

;　　其余同上。

四、标求和逻辑型指令简表：

操作 \ 位数	5	6	7	名称	符号	操作功能
基本操作	0	0	0	求反码	COM	$(\overline{ACS}) \Longrightarrow ACD$
	0	0	1	求补	NED	$(\overline{ACS})+1 \Longrightarrow ACD$
	0	1	0	累加四传送	MOV	$(ACS) \Longrightarrow ACD$
	0	1	1	加1传送	INC	$(ACS)+1 \Longrightarrow ACD$
	1	0	0	加反码	ADC	$(\overline{ACS})+(ACD) \Longrightarrow ACD$
	1	0	1	减法	SUD	$(\overline{ACS})+1+(ACD) \Longrightarrow ACD$
	1	1	0	加法	ADD	$(ASE)+(ACD) \Longrightarrow ACD$
	1	1	1	逻辑乘	AND	$(ASB) \wedge (ACD \Longrightarrow ACD$

辅助操作	位数	8	9	名称	符号	操作功能
	移位	0	0	不移	~	不移
		0	1	左移	L	运标结果连进位循环左移一位
		1	0	右移	R	运标结果连进位循环右移一位
		1	1	半字交换	S	进位不变移位四前后8位交换
	位数	10	11	名称	符号	进位的基本值
	进位	0	0	进位不变		取进位的现有状态
		0	1	零基值	Z	进位值取零
		1	0	一基值	O	取一
		1	1	反基值	O	取进位现状反码
	位数	12		名称	符号	操作说明
	存数	0		存		结果存入结果累加四
		1		不存	#	结果不存入结果累加四

接上表

	13	14	15	名称	符号	功能
辅助操作	0	0	0	跳步测试	—	不跳
	0	0	1		SKP	必跳
	0	1	0		SZC	进位为"0"跳
	0	1	1		SNC	进位非"0"跳
	1	0	0		SZR	结果为"0"跳
	1	0	1		SNR	结果非"0"跳
	1	1	0		SEZ	两者有一为"0"跳
	1	1	1		SBN	两者皆不为"0"跳

第二节　访内型指令

直接与内存有关的指令称为访内型指令。这类指令包括存取指令，参改地址指令和转移指令。每一个访内指令的第0位均为0，第1～4位表示存取操作或转移操作，从第5位至第15位就有统一的格式，第5位表示间址位（I），第6，7位为变址位（X），第8～15位为相对地址（D）。该指令的有效地址E决定于I，X和D的值。若X为00，D所表示的地址在256个地址之间，即D的地址范围为00000～00377，习惯上把这组地址称为第0页。

I	X	D
5	6 7	8 9 10 11 12 13 14 15

若X不为0，那么必须将相对地址D加到X所指示的变址寄存器的内容上才能形成一个内存地址。（X）± D，D的第8位表示符号位0为正，1为负。因此D所表示的数的范围为-200到+177

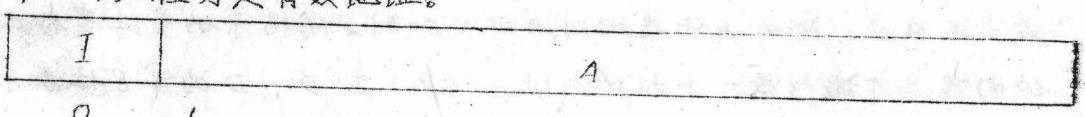
（十进制 −128 到 +127）。若 Z 为 0 1 指令寻址的地址为指令计数口 PC 的内容加上相对地址 D，（PC）为当前执行指令的地址。这种寻址方法，称为相对寻址。也就是相对于 PC 寻址。如果 Z 是 10 或 11，则指令以 AC_2 或 AC_3 作为变址寄存口。变址寄存口的内容与相对地址 D 相加形成指令要寻址的地址。列表如下：

Z	地址寻出	
0 0	0 页寻址	D 的范围为 00000 ～ 00377 中的一个地址
0 1	相对寻址	（PC）+ DD(−200 ～ +177)
1 0	变址寻址	(AC_2) + D　D (−200 ～ +177)
1 1	变址寻址	(AC_3) + D　D (−200 ～ + +177)

在使用累加口作为变址寄存口时特别值得注意的是只能用 AC_2 或 AC_3 作为变址寄存口，AC_0 和 AC_1 不能作为变址寄存口使用。

如果指令的第 5 位 I 为 0，是直接寻址，由 Z 与 D 直接确定的地址就是有效地址。这样，访问内存的指令可以直接访问 1024 个地址，即 0 页地址及由 PC，AC_2，AC_3 所指定的，范围在 +177 和 −200 之间的三组 256 个地址。

如果 I 为 1，则为间接寻址。此时中央处理机就从刚得到的地址中读出该地址中的数作为地址。在读出的字中第 0 位代表间地位，如第 0 位为 0，则第 1 ～ 15 位为有效地址，否则又是另一级寻址的地址，这一步骤一直进行到第 0 位为 0 为止，这时 1 ～ 15 位才是有效地址。



在间接寻址时，如果在任一级有效地址的计标中，一个地址码取自00020 — 00037单元，则自动加1或减1，并将结果存入原单元，且根据0位是0还是1，将该值作为有效还是作为下一步计标地址用。地址取自00020 — 00027单元为加1，取自00030 — 00037单元为减1。

间接地址在编派程序时记忆符号为@。

例如：　　LDA 1, @B; ((B)) ⇒ AC1

表示对B间接寻址，即将B所指的地址中的数取出送入AC1中（此时设（B）的第0位为0）。

一、存取指令

1、取数

（1）记忆符号：LDA

（2）格式：

001	AC	I	X	D
0 1 2	3 4	5	6 7	8 9 10 11 12 13 14 15

（3）意义：将有效地址E的内容取出存入AC所指的累加器，E的内容不变，AC的原始状态被破坏。

（4）例：

假设下列地址的内容如下：

地　址	内　容
6	100015
12	000035
15	000017
17	000023
23	000011
AC_3	000015

如果程序执行下面的指令：