TQ-15

数字电子计秤机

（上册）

TQ—15机简介

TQ—15是一台字长十六位的小型通用电子数字计算机，全机由中央处理部件、内存贮器，外部设备三大部分组成。它们之间的关系及系统框图见附图。

中央处理部件由运算器、控制器组成。

运控装置是全系统的控制单元，管理全部的输入—输出装置，形成全部的算术和逻辑运算，以及发出一系列控制脉冲序列，指挥整机协调地运行。运控部件通过输入、输出总线和外部设备相连。用内存总线和内存贮器相连。

运算器主要由十六位并行全加器Q和一位进位位C_3组成。全加器的进位链采用四位一小组，十六位共分四小组的组内、组间并行快速进位方式。进位位C_3，保存算术运算第0位产生的进位讯号和左右移位时第0位和第15位的讯号。

为了加快机器运算速度减少与内存存访次数，采用了多累加器结构。四个累加器L_0、L_1、L_2、L_3，每个累加器十六位，存放参加运算的数据和中间结果。在访内指令时，其中L_0和L_1还可兼作变址寄存器。在作输入输出指令时，也作为外部设备缓冲器同运控交换数据的对应寄存器。

由于采用了多累加器，为了选择参与运算的累加器内容，设置了累加器数据选择门M_{CS}操作数选择门D、M_{RC}结果选择门C各十六位。补码运算还需反相，也在此形成。

选择全加器的操作数，再由全加器输入门十六位控制，每位参与运算的操作数分别由一个（A及B）四与或非门输出。

全加器输出到移位器十六位，每位有一个四与或非门，可将全加器运算结果直接输出；左移一位；右移一位；前后八位交叉输出。需反相时，有十六位反相器，将移位器结果反相，同时在此又形成结果判零讯号。

控制器产生全机所有的控制讯号，它将内存读出的指令进行寄存、译码，以及为顺序作后继指令做好准备。它主要由指令寄存器（十六位），指令译码器，指令计数器（十五位）组成。此外为控制整机有条不紊的正常运行，还必须产生一系列时序脉冲，如节拍电位W_0、W_1、工作脉冲m^*、m_{RC}、m，以及读令DL、写令XL、选通XT等等讯号。本机器主时钟为5兆赫，执行一条算术逻辑指令时间1.6微秒，取送数时间3.2微秒。

内存贮器采用三度二线电流重合法，存取周期1.6微秒，基本容量4K，且以4K为模，最大容量可扩到32K。

为和磁芯体并行交换讯息，具有十七位重写寄存器 Jcx 和它的输入多路开关 CXD。为了选择内存单元还有十五位地址寄存器 Jd。重写寄存器不仅用来再生磁芯存贮器的内容，而且还用作主机和外部设备交换代码的接口寄存器。此外内存贮器还有一套读写驱动，读出放大、禁止电路以及读令、写令，选通时序脉冲的微调电路。

内存贮器不但能在程序控制下，与中央处理部件累加器交换数据，而且可通过数据通道形式，直接与高速外部设备交换数据。

外部设备采用标准接口，接口最多可带有三个缓冲寄存器，标准接口控制逻辑包括有：设备运行状态触发器，请求中断触发器，中断屏蔽触发器，设备码选择和中断优先排队编码线路。外部设备最多可接 62 种，它们分别用 16 级中断加以处理。也可用设置设备的屏蔽状态来任意调整中断级别高低，多重中断由程序处理。一般慢速设备由程序中断和主机累加器交换数据，高速设备和外存用数据通道形式直接和内存快速交换数据。输入输出最高交换频率为 666 千赫。

此外为适应科学计算，实时控制，多道分时运行。机器设有可扩部件：快速乘除部件，内存分配保护，实时时钟等，使它真正具有多种功能。

目　录

TQ—15机简介 …… 1
第一章　指令系统 …… 1
　第一节　算术逻辑指令 …… 3
　　一、算术逻辑指令的逻辑结构 …… 3
　　二、数的表示方法 …… 4
　　三、指令分析 …… 5
　　四、汇编语言举例 …… 8
　第二节　访问内存指令 …… 9
　　一、指令分析 …… 9
　　二、汇编语言举例 …… 12
　第三节　输入输出指令 …… 13
　　一、指令分析 …… 13
　　二、汇编语言举例 …… 17
　　三、"77"码指令 …… 17
第二章　运算器 …… 19
　第一节　加法器 …… 19
　　一、半加和 …… 20
　　二、全加和 …… 21
　　三、进位链 …… 21
　第二节　数据传送和移位、判零 …… 27
　　一、加法器输入数的接收 …… 27
　　二、加法器输出数的传送 …… 28
　　三、移位线路 …… 29
　　四、结果判零 …… 30
第三章　控制器 …… 31
　第一节　指令控制部件 …… 31
　　一、指令寄存器 …… 31

二、译码器……32
三、指令计数器……33
第二节 节拍电位、脉冲和周期……34
一、时序脉冲的产生……35
二、节拍电位的转换控制……36
三、内存时序控制……37
第三节 指令流程……38
一、取指状态流程……40
二、间址状态流程……43
三、执行状态流程……45
四、中断状态流程……50
五、数据通道状态……51
第四节 指令操作时间表……52
一、操作时间表编制说明……53
二、操作时间表……54
三、指令控制图……64
第五节 代码总线……70
第六节 启停控制……71
第七节 控制台……73
一、控制台指令……73
二、控制台面板指示灯……78
三、控制台面板开关……79
四、控制台线路……82
第四章 中断和数据通道……83
第一节 中断的基本结构……85
第二节 程序中断……86
一、中断请求……86
二、启动中断……87
三、中断处理……88
四、中断的优先权……89
五、中断的恢复……91
六、多级中断……92

七、中断管理指令……95
八、中断程序处理示例……97
九、中断有关问题的讨论……100
第三节 数据通道……101
一、数据通道存访请求……103
二、通道排队……104
三、数据通道存访启动……104
四、数据通道存访周期……106
五、数据通道结束中断……107
第五章 乘除部件……109
第一节 乘除法指令形式……109
一、乘一加……109
二、除法……109
三、取乘除结果并解除封中……110
第二节 乘除运算法……110
一、两位乘法……111
二、恢复余数除法……112
三、不恢复余数除法……112
第三节 乘除部件缓冲寄存器……113
一、A寄存器……113
二、B寄存器……114
三、C寄存器……114
四、A B C寄存器的打入讯号……114
第四节 乘除部件控制器……115
一、指令译码和乘除标志触发器……115
二、运行状态、封锁中断、溢出触发器……116
三、乘除部件局部计数器（次数计数器）……117
四、微程序控制……119
第五节 乘除部件运算器……123
第六章 实时时钟和自动引导程序……127
第一节 实时时钟……127
一、指令形式……127

二、工作过程 …… 128
三、标准频率的形成 …… 128
第二节 自动引导程序 …… 129
第七章 内存分配和保护 …… 135
第一节 概述 …… 135
一、问题的提出 …… 135
二、实现内存分配和保护的基本思想 …… 136
三、内存分配和保护的主要方法 …… 137
四、软硬件的保证 …… 140
第二节 内存分配和保护举例 …… 141
一、页面的分配和保护 …… 141
二、分时管理 …… 145
三、主机中断 …… 149
四、管态——目态的转换 …… 150
五、变址单元的浮动 …… 152

附录：缩语表 …… 153

第一章 指令系统

数字电子计算机在进行复杂的解题计算和实现某种自动控制时，它必须执行一系列的操作：如读、写、算术逻辑操作以及输入、输出等等。我们把给机器发布执行这些操作的命令称为计算机的指令，一台计算机执行多种操作，它必需具有一定数量的，不同类型的指令，这些指令构成了计算机的指令系统，它是衡量一台计算机功能的重要指标之一。

由于计算机面向的问题不同，因而各类的计算机它的指令系统并不一样，就大多数计算机来说，它都是能执行如下几类功能的。

1.把数据在存贮器和工作寄存器之间转送。

2.修改存贮器，进行程序转移判别。

3.转移——改变程序执行的顺序。

4.执行一个算术和逻辑操作。

5.检验一个字或特征标志的值，或把一个字和另一个字比较，以便决定是否要改变程序的顺序。

6 把数据对外围设备输入或输出。

为实现上述这些功能而进行指令设计时，在许多计算机中，第一组和第四组以及第三组和第五组是重叠的，就是说进行一个算术逻辑操作或比较计算与从内存取操作数两者是结合在一起的，这是一种自然的考虑方法。因为要执行操作那就应该有操作数，而操作数通常是来自内存的。这样就决定了这种指令格式的设置可以有如下形式：

它分为两部分即操作码部分和地址部分，其中特征位是用来修饰地址意义。

在这种形式的指令中，因地址部分个数的多少，它可分为单地址、二地址、三地址、四地址等类型：

单地址指令就是地址部分只有一个操作地址，以算术运算为例，这指令形式表示由存贮器地址码D中取出数码(D)和保留上次运算结果的累加器L的内容进行运算，把

本次运算的结果仍然送到累加器L中，表达式如下：

$$(L)\ \theta\ (D) \rightarrow L$$

二地址指令则表示地址部分有两个操作地址，它有如下的格式：

θ	D_1	D_2

这指令的功能是将存贮器单元D_1的内容和存贮器单元D_2的内容(D_2)进行运算，然后其结果送存贮器单元D_2或保留在累加器L中。

$$(D_1)\ \theta\ (D_2) \rightarrow D_2 \text{ 或 } L$$

至于三地址指令它主要是在二地址指令的基础上附加一个存放运算结果的存贮器单元地址，而四地址指令中的第四个地址却是用来指出下一条指令地址，这里我们就不多加叙述。

在上述几种指令中，以采用单地址指令的计算机结构简单、执行一条指令的时间短，是目前使用最多的一种，而多地址计算机编制程序方便，执行一条指令时所能完成的功能多，但是不管怎样，采用上述指令格式执行一条指令总是要执行二个内存周期以上，这在某种程度上对于计算机速度的提高是不利的因素。

根据这样的分析，TQ—15机在指令设计中采取了与前述稍有不同的考虑方法，它的前提是运控采用多累加器设计的思想，多累加器与只有一个累加器相比，由于它可以存放运算时的中间结果，这就显著地减少了与内存的数据交换，这样也就减少了指令的条数，因而提高了机器的效率。以交换内存A和B二个单元的内容为例，采用多累加器可比单累加器减少三分之一的时间：

一个累加器	二个累加器
A→L	A→L_1
L→中间寄存器	B→L_2
B→L	L_1→B
L→A	L_2→A
中间寄存器→L	
L→B	

正是基于多累加器的这个优点，TQ—15机在考虑开头提出的六种指令功能中，把第一组和第三组独立开来了，而让第四组和第五组重叠，就是说算术逻辑操作与检验特征和比较判别结合一起考虑，这时的操作数就取自累加器，运算器中间结果也存放在累加器中，因此算术逻辑操作就变成只是简单地在累加器中进行运算了，而与内存不发生关系，这是TQ—15机中的一种类型的指令，称之为算术逻辑型指令。由于这类指令不

包括内存地址，16位字的指令便空出许多位用作定义其他功能，例如除了对两个操作数完成其主要功能外，还可以对结果移位，判别跳步等等；还可以决定结果是否送回累加器。由此算术逻辑指令只需要一个取指周期，时间上是节省的。

① 算术逻辑指令的这种考虑方法可以看成是前面分析的二地址指令的特款，只是前者取自内存地址，后者是累加器地址罢了。

这样划分之后，第一、二、三组就都是同内存有关的了，他们统一为一类访问内存指令，采用单地址形式。并且由于字长的限制，通常指令中的变址和间址的使用也是势所必需的。

② 输入、输出单独作为一类指令，它也充分利用多累加的优点，而让外部设备与累加器进行信息交换，同时又可以对设备进行控制。

上面就是TQ—15计算机三种类型指令考虑的基础，它和传统的指令设计有所区别，由于三类指令是区别对待的，这种处理对于发挥指令中每一位的作用方面较为充分，而不发生浪费，同时指令形式又是复合的，具有多种操作，它向使用者提供了内存的微程序功能，使得程序设计较为方便灵活。

TQ—15计算机的指令统一字长16位，基本指令22条，下面我们将分别加以介绍，但是讨论中有关输入输出指令对外部设备所具有的特殊功能，如程序中断，实时时钟，乘除部件以及页面的分配和保护等将不包括在内，它们留在有关的章节中介绍。

为了书写、阅读的方便，在指令分析中将给出符号语言的应用，它克服了直接机器语言不直观，易写错的缺点，有利于减轻劳动强度，提高工作效率。

第一节　算术逻辑指令

一、算术逻辑指令的逻辑结构：

如前所述，算术逻辑指令是累加器与累加器运算，并且结果也送回累加器，逻辑结构如图1—1所示：

图中运算器可接受一个或二个累加器的数，每个参加操作的数为16位，运算器输出为17位，其中运算结果为16位，还有一位是进位位。

移位门的作用是将运算器输出的17位结果实行左移1位或右移1位或将16位结果前后8位交换（不包括进位值）经过移位操作的17位最后结果送入跳步测试。跳步是根据最后的16位结果和进位位是否为"0"而决定，再按照指令来定回收至累加器与否。

进位位的意义对不带符号的数来说是明显的，例如：在做加法或加1时，在数的范

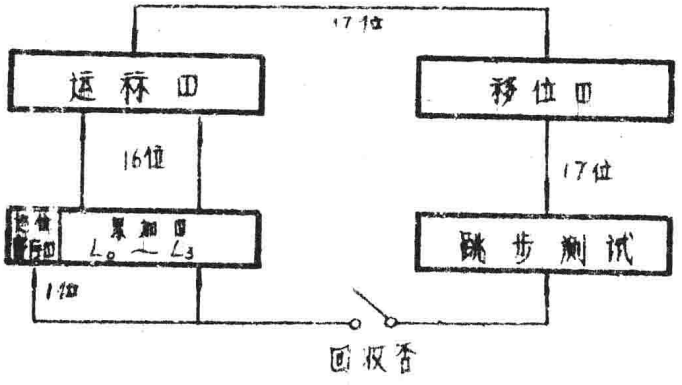


图 1-1　补逻指令的逻辑结构

围超过 $2^{16}-1$，就产生进位。在减法运算中，是通过加上减数的反码，再加 1 来实现的；在减法 A－B 运算时，若 A≥B 则产生进位。在求"0"的补码时也产生进位，因为求"0"的补码就是将 16 位为全"1"的数再加 1。对于有符号的数，如果也把它看成不带符号的数来处理，则和上述的情况是一样的。

二、数的表示方法：

逻辑运算时，机器把参加运算的数都看成逻辑字，因此不存在符号的问题，亦不会产生进位。

对算术运算来说，机器把参与运算的数当作十六位不带符号的数来处理，但是程序也可以作为用二进制补码表示的带符号的数。

对二进制补码来说，带符号的数和不带符号的数是一样处理的，在机器中将符号位作为最高位的数来处理。假设一个累加器的数如下：

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

这里的位号和字(指令)的排列一样，自左至右为 0～15 位。作为不带符号的数其值为：

$$100131_8 = 32857_{10}$$

作为以二进制表示的带符号的补码其值为：

$$-77647_8 = -32679_{10}$$

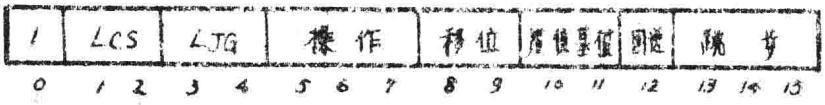
在机器中，不论我们事先对这些数的假定如何，其运算是完全一样的。但是，对于小数点在数码中的位置，计算机是无法加以识别的，因此，程序工作者必须设定一种小数点的表示方法，并将结果移位，以符合事先的设定。一般有二种规定：一种是把数看

成为整数，即小数点在最右边。另一种是把数看作为小于1的数，即小数点在第0位和第1位之间。在这二种情况下，带符号的数的范围分别为：

$$-2^{15}\sim 2^{15}-1 \quad 和 \quad -1\sim 1-2^{-15}$$

三、指令分析：

为了实现上述逻辑结构，指令字中每位作如下安排，第0位为1，则为算术逻辑指令，第5.6.7三位可以表示八种运算（算术运算五种，逻辑运算三种）。任何一种运算都使用由指令字的第1，2位所决定的累加器L_{CS}的内容，如果还需要第二个操作数，就采用由第3，4位所决定的累加器L_{JG}的内容。指令的格式如下：



指令字的第10，11二位指出取何种进位基值，其基值根据执行的运算及其结果选取，各种基值的定义及其符号如下：

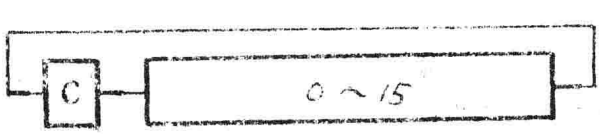
符号	10，11位	进位基值
	0 0	取进位寄存器的现有状态。
Z	0 1	取"0"
O	1 0	取"1"
C	1 1	取进位寄存器的现有状态的反码。

对三种逻辑运算（求反、传送、逻辑乘），只是按上述的基值作为进位，送到移位门；对五种算术运算来说，如果运算时有进位，则将上述的基值取反后送到移位门，反之就按上述的基值送移位门。根据进位寄存器的状态和结果的符号，这二者可以用来判别带符号的数在运算时是否溢出；但它的主要用途是记下不带符号的数在运算时最高位的进位。例如在双倍字长运算时，它的低位部分就是如此。基值的作用可归纳成一个简单的公式：

本次进位＝第0位进位⊕进位基值。

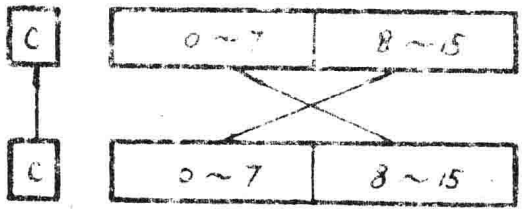
指令字的第8，9二位是指出对16位的结果和进位执行以下的操作：

符号	8，9位	操作
	0 0	无
L	0 1	左移一位。第0位移入进位位，进位位移入第15位。



R　　1 0　　右移一位。进位位移入第 0 位，第 15 位移入进位位。

S　　1 1　　第 0～7 位与 8～15 位的二半交换，进位位不变。



指令字的第 12 位用来决定是否回送。如果该位为"0"，则最后的 17 位结果通过移位门回送到进位寄存器和累加器，累加器由指令字的第 3、4 位决定；如果该位为"1"则不回送，也不改变原有的进位寄存器的状态。注意：要回送的内容是已经执行了指令字 8,9 位所指定的操作。

指令字的第 13, 14, 15 三位作为决定指令跳步条件，运算结果如果符合跳步条件则跳过下一条指令。每位的意义如下：

位　号	这位为 1 的作用
13	移位门输出的 16 位结果为"0"则跳。
14	移位门输出的进位值为"0"则跳。
15	取 13、14 位所规定的相反条件。

指令字的第 13, 14, 15 三位所指定的八种跳步条件：

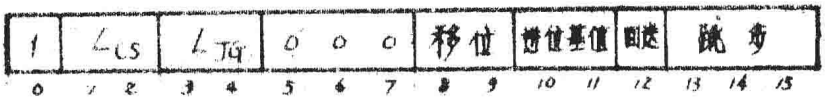
符　号	13, 14, 15 位	跳步条件
	0　0　0	不跳步
SKP	0　0　1	必　跳
SZC	0　1　0	进位为 0 则跳
SNC	0　1　1	进位非 0 则跳
SZR	1　0　0	结果为 0 则跳
SNR	1　0　1	结果非 0 则跳
SEZ	1　1　0	进位或结果为 0 则跳
SBN	1　1　1	进位及结果均为非 0 则跳。

这跳步测试是针对移位门的输出而言。因此，如果在加法时，把结果左移一位，则 SZC 或 SNC 实际上就是测试和的符号。应注意的是不论移位门的输出是否回送，跳步

测试还是正常地进行。因此，程序可以既测试一个算术逻辑运算的结果而不破坏原有的进位寄存器和累加器的内容。

指令字的第5～7三位操作码可对八种基本运算进行选择，现将每种运算叙述如下：（在指令字名称右边括号为汉语拼音缩符）

1. COM 取反　　(QFA)

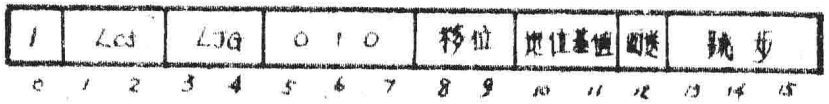


把 L_{CS} 的字取反，并和进位基值一起送入移位门，接着执行由第 8，9 二位所规定的移位操作。如果第 12 位不为 1，则将移位门的输出回送到进位寄存器和 L_{JG}。如移位门的输出符合跳步条件，则跳过下一条指令。

2. NEG 取补　　(QFU)

把 L_{CS} 的操作数取补并送入移位门，如果 L_{CS} 为 0，则把进位基值的反码送入移位门，否则就送指定的值。接着执行由第 8，9 位所规定的移位操作。如果第 12 位不为 1，则将移位门的输出回送到进位寄存器和 L_{JG}。如移位门的输出符合跳步条件，则跳过下一条指令。

3. MOV 传送　　(CHS)



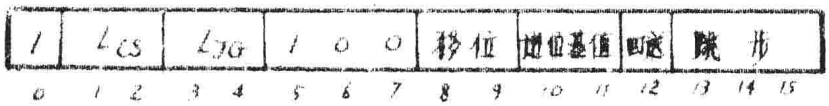
把 L_{CS} 的操作数和进位基值送入移位门，接着执行由第 8，9 二位所规定的移位操作。如果第 12 位不为 1，则将移位门的输出回送到进位寄存器和 L_{JG}。如移位门的输出符合跳步条件，则跳过下一条指令。

4. INC 加 1　　(JYI)

把 L_{CS} 的操作数加 1，并将结果送入移位门。如果 L_{CS} 为 $2^{16}-1$，则把进位基值的反码送入移位门，否则就送指定的值。接着执行由第 8，9 二位所规定的移位操作。如果第 12 位不为 1，则将移位门的输出回送到进位寄存器和 L_{JG}。如移位门的输出符合跳步条件，则跳过下一条指令。

5. ADC 加反　　(JFA)

把 L_{CS} 的数取反，加上 L_{JG} 的数，并将结果送入移位门。对不带符号的数来说，如

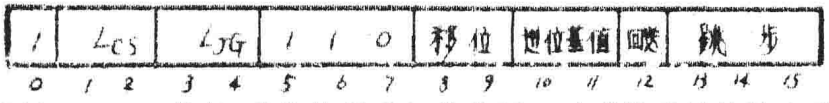


果($L_{JG} > L_{CS}$)，则把进位基值的反码送入移位门，否则就送指定的值。接着执行由第8,9二位所规定的移位操作。如果第12位不为1，则将移位门的输出回送到进位寄存器和L_{JG}。如移位门的输出符合跳步条件，则跳过下一条指令。

6. SUB 减法　　(JAN)

把L_{CS}的数取补，加上L_{JG}的数，并将结果送入移位门。对不带符号的数来说，如果($L_{JG} \geq L_{CS}$)，则把进位基值的反码送入移位门，否则就送指定的值，接着执行由第8,9二位所规定的移位操作。如果第12位不为1，则将移位门的输出回送到进位寄存器和L_{JG}。如果移位门的输出符合跳步条件，则跳过一条指令。

7. ADD 加法　　(JIA)



把L_{CS}的数加上L_{JG}的数，并将结果送入移位门。对不带符号的数来说，如果二数的和大于或等于2^{16}，则把进位基值的反码送入移位门，否则就送指定的值，接着执行由第8,9二位所规定的移位操作。如果第12位不为1，则将移位门的输出回送到进位寄存器和L_{JG}。如果移位门的输出符合跳步条件，则跳过下一条指令。

8. AND 逻辑乘　　(LYU)

把L_{CS}和L_{JG}的二个字进行逻辑乘，并将结果和进位基值送入移位门。接着执行由第8,9二位所规定的移位操作。如果第12位不为1，则将移位门的输出回送到进位寄存器和L_{JG}。如果移位门的输出符合跳步条件，则跳过下一条指令。

四、汇编语言举例：

用以上介绍指令功能时所用的符号，再加上一些规定的字符号来表示指令，并使它符合汇编程序的规定，这就是通常所说的汇编语言。例如：

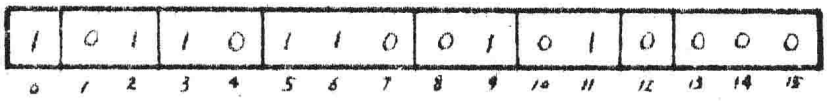
指令

汇编语言——ADD　1，2（机器语言——133000）这条指令的功能是将L_{CS1}与L_{JG2}的数相加，结果不执行移位操作就回送到L_{JG2}。如果最高位有进位，则将进位寄存器

的内容取反。

如果还需要规定进位基值和移位操作，只要在表示运算操作的符号右边用规定的符号表示，但是先写表示进位基值的符号(如有的话)。例如，在最高位有进位时，把进位值取1，并将17位的结果执行左移一位，最后结果送入进位寄存器和L_{JC_2}，写法如下：

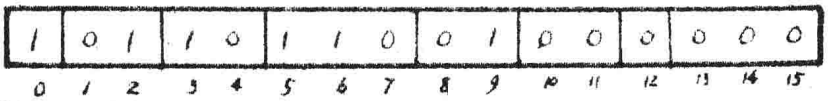
指令



汇编语言——ADDZL 1,2(机器语言——133120)该指令将和的符号送入进位寄存器，和的其余位送L_{JC_2}的0～14位，L_{JC_2}的15位为0或1，对应于最高位的进位。

如果进位基值取目前进位寄存器的状态，而不是取"0"，其他操作同上一条指令，写法如下：

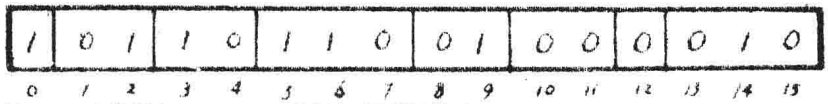
指令



汇编语言——ADDL 1,2(机器语言——133100)

如果操作同上一条指令，但和为正时跳过下一条指令写法如下：

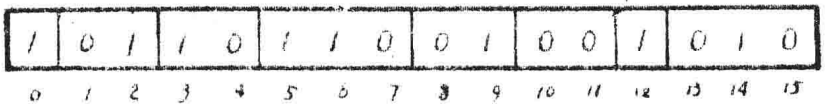
指令



汇编语言——ADDL 1,2 SZC(机器语言——133102)

如果操作及跳步条件同上一条指令，但指令字第12位为1，即不回送，使得进位寄存器和L_{JC_2}的原有值不受破坏。写法如下：

指令



汇编语言——ADDL* 1,2 SZC(机器语言——133112)

写在表示扩展操作的符号右边的"*"记号表示不做回送操作，即指令字的12位为1。

从上面分析可看出，汇编语言书写指令的顺序是：

操作码 进位基值 移位 是否回送 第一操作数寨加器 第二操作数寨加器 跳步测试

第二节 访问内存指令

一、指令分析：

访问内存型指令的第0位为"0"，第1、2位不同时为"1"，它的基本指令共有六条，

这六条指令字的第 5～15 位的定义是一样的。在叙述这些指令之前，先说明第 5～15 位的作用。

利用指令字的 5～15 位，可以求出最终所需的有效地址 E。其中 8～15 位是形式地址 D，6，7 二位表示变址方式。将形式地址 D，按照变址方式实行变址运算后，得到的就是变址地址 E*，然后再根据第 5 位可得到有效地址 E。

第 6,7 二位表示四种变址方式：

第 6 7 位	变 址 方 式
0 0	形式地址 D 当作变址地址 E*（D＝E*）其值的范围为：00000～00377
0 1	形式地址 D 为带符号的数，第 8 位为符号位，以补码表示，其值的范围为：－200～177。变址地址 E* 等于指令计数器 $J_{\swarrow}$ 的值加上形式地址 D 即 $E^* = (J_{\swarrow}) + D$
1 0	形式地址 D 为带符号的数，第 8 位为符号位，以补码表示，其值范围为－200～177。变址地址 E* 等于第 2 个累加器 L_2 的值加上形式地址 D，即 $E^* = (L_2) + D$
1 1	形式地址 D 为带符号的数，第 8 位为符号位，以补码表示，其值范围为－200～177，变址地址 E* 等于第 3 个累加器 L_3 的值加上形式地址 D，即 $E^* = (L_3) + D$

上述的形式地址 D 是由八位二进制数组成，第 8 位是符号位，为了适应机器的需要，在变址运算的过程中，要把它扩展成 16 位，其原理和二进制补码相同：如果 D 为正数时，第 8 位为 0，在第 9～15 位补 0，这是容易理解的。而当取 D 的负值时，应将 D 取反再加上 1，这时第 0～8 位都变成 1 了。由此可知，要扩展成 16 位的数，只要在第 0～7 位补充同符号位一样的数。