全国计算机软件专业技术资格(水平)考试辅导丛书

CASL
汇编语言及题解

(高级程序员级必考)

朱慧真　管有庆　唐肖光

中国软件行业协会考试指导中心

全国计算机软件人员水平考试辅导丛书

CASL汇编语言及题解

（高级程序员级必考、程序员级选考）

朱慧真　管有庆　唐肖光

中国软件行业协会考试指导中心

内　　简　介

本书为全国计算机软件人员水平考试的复习辅导材料，系由中国软件行业协会考试指导中心组织编写。

CASL汇编语言目前已列为我国“计算机软件人员水平考试大纲”中规定使用的汇编语言.它是高级程序员级的必考科目，也是程序员级的任选五种语言(FORTRAN、COBOL、PASCAL、C及CASL）之一。为此，我们特组织本书的编写、出版，内容包括：

CASL汇编语言基础（北京大学　朱慧真）

CASL汇编语言习题与解答（东南大学　管有庆　唐肖光）

IBM-PC CASL汇编语言系统（管有庆　唐肖光），配有教学实习软件。

CASL汇编语言文本（考试中心）

本书内容原刊于《软件产业》月刊，现经增订作为单行本出版。

顾问：杨天行　徐家福　杨芙清　王尔乾

全国计算机软件人员水平考试辅导丛书

CASL汇编语言及题解

朱慧真　管有庆　唐肖光　编著

责任编辑　徐一君、王照华

*

（内部发行）

中国软件行业协会考试指导中心

北京飞达印刷厂印刷

开本:787×1092 1／16　印张:10　字数:300千字

1991年4月修订第2版

前 言

计算机软件人员水平考试是造就宏大的计算机人才队伍的一项重要措施。几年来的实践证明，它对于激励在职计算机人员提高软件水平，配合国家人事部、国务院电子信息办、机械电子部对软件人员技术职称评定和职务聘任，以及加强对外软件人才交流等方面都起了积极的作用。

为了帮助各类软件人员准备应试，并供各地举办辅导班参考，我们邀请了北京大学、清华大学、中国科学院研究生院等许多高校的计算机专家、教授组成编委会，陆续编辑出版《全国计算机软件人员水平考试辅导丛书》。丛书包括各个级别的辅导教材与题解。

CASL汇编语言目前已被列为我国"计算机软件人员水平考试大纲"中规定使用的汇编语言。它是高级程序员级的必考科目，也是程序员级的任选五种语言(FORTRAN、PASCAL、C、COBOL及CASL) 之一。为此，我们特组织本书的编写、出版，内容包括：CASL汇编语言基础（北京大学朱慧真）、CASL汇编语言习题及解答、IBM-PC CASL汇编语言系统（东南大学管有庆、唐肖光），书末并附有考试中心公布的CASL汇编语言文本。

本书主要内容曾在1990年度《软件产业》月刊分期发表，IBM-PC CASL教学软件曾经国内许多单位使用，反映良好，对学员学习CASL汇编语言有帮助，并能验证汇编试题的正确性。现应读者需要，特将上述内容汇集成册，将由电子工业出版社作为单行本出版，相应内容也作了增补。相信能够满足考生们的学习需要。

本书曾经郑人杰、徐国平副教授审阅，并感谢《软件产业》编辑部及电子工业出版社第二编辑室所给予的支持和帮助。

中国软件行业协会考试指导中心

1991年4月

目 录

CASL汇编语言基础

前　言

汇编语言是一种初级计算机语言，实用的汇编语言都是面向机器的语言。然而CASL是一种抽象的汇编语言。它定义在抽象计算机COMET上。它的功能是各种计算机汇编语言最基本的共同部分。它具有语句种类少，每个语句功能单一，程序格式较规范等特点。因而它便于汇编试题的标准化，便于统一测验应试者汇编语言的编程水平。

CASL汇编语言原是1987年后日本计算机应用软件人员全国统考中使用的汇编语言文本。目前我国制订的“计算机软件人员水平考试大纲”中也规定使用CASL汇编语言。它是考高级程序员必考的科目，它又是考程序员时五种语言（FORTRAN、COBOL、PASCAL、C及CASL）任选一种之一。这里所讲的CASL文本与日本的CASL文本唯一的不同是所采用的字符编码不同，前者采用ASCII编码表（见附录1），后者使用的字编码表见《日本计算机应用软件人员全国统考试题及解答》一书的108页之表。

这里主要讲以下四部分：

1.1　CASL的硬件基础

这部分主要介绍抽象机COMET的定义。

1.2　CASL的语句功能

这部分主要讲四种伪指令语句，三种宏指令语句及二十三种指令语句。

1.3　CASL的编程基本技巧

这部分主要讲如何利用以上所讲语句，构造分支程序、循环程序及主程序和子程序。

1.4　CASL的题例分析

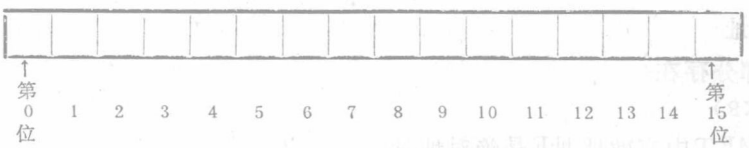
这部分对我国和日本历届软件水平考试的部分汇编试题进行了类型和解法分析。

1.1　CASL的硬件基础

CASL汇编语言依赖COMET机的下列特点：

一、内存储器

内存按字编址，字长为十六位（bit），内存容量为2^{16}，故内存地址编码也是十六位，存取地址为0--65535。程序中使用绝对地址，内存字中各位的编号如下所示：



二、程序中访问的寄存器

1．通用寄存器

COMET中有五个十六位通用寄存器，它们的名字分别为：GR0，GR1，GR2，GR3及

GR4，序号为0～4。它们用于算术、逻辑、比较及移位等语句中，其中1号～4号也可作为变址寄存器使用，4号寄存器还可作为堆栈指针使用，堆栈指针是存放堆栈最上面(stack top)地址用的寄存器。

2．指令寄存器

PC为十六位指令寄存器，它存放执行中指令的开始地址，当指令执行完毕，下一个要执行的指令的开始地址即被给定。一般地，指令执行完毕时，PC中地址加“2”；分支、调用及转移指令的场合，新的分支的地址被给定在PC中。地址运算按65536取模。

3．标志寄存器

FR为两位标志寄存器，它用于装入指令以及算术、逻辑及移位的运算指令执行后，记录其执行结果（在GRI里的）数据为正为负还是为零的信息，以及由比较运算指令的执行，而得到的两数间大小关系的信息。

一般来说FR的第一（左边）位表示运算结果的符号位（“0”为正、“1”为负）；第二位表示运算结果是否为零（“0”为非零，“1”为零）。

装入指令、运算指令及移位指令执行后FR的值见表1.1，比较指令执行后FR的值见表1.2。

表1.1

	GR里数据特征		
	正	负	零
PR的值	00	10	01

表1.2

	(GR)与(有效地址)比较		
	>	<	=
PR的值	00	10	01

三、指令形式及有效地址

指令具有双字长，即每条指令占两个字空间（32位），指令的具体形式无定义。

1．指令的符号书写格式

指令的符号书写格式如右所示：OP，　GR，　adr[,XR]

其中OP表示操作码，GR表示所用通用寄存器，adr表示直接地址数，XR表示所用变址寄存器。

2．指令的有效地址

有效地址E有两种形式：

(1) 直接地址

条件：XR部分省略

E=adr

(2) 变址地址

条件：XR部分存在

E=adr+(XR)

注意在COMET中有效地址E是绝对地址。

四、数据的机内表示

COMET是定点机，机内数据字长为十六位，各位编号与内存各位的编号相同。

数据有两种：

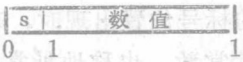
1．十六位无符号数

数值范围在0～65535。

2．十六位有符号

采用补码表示法

其中S为符号位，数值范围在-32768～32767．



五、字符编码

输入输出设备的字符编码统一由ASCII编码表所定义，详见附录1。

1.2 CASL的语句功能

CASL汇编语言中有两种语句：说明性语句与执行性语句。说明性语句又分为伪指令语句与宏指令语句两种。执行性语句只有一类指令语句。我们可把指令语句看成是每条语句对应一条目标指令，占用一个字长的内存空间。

CASL汇编语言中语句的一般形式为：〔标号〕　　操作符　　〔地址〕〔；注释〕

其中第一部分为标号，语句的这部分位置可称为标号位置，汇编语言中的标号和其它计算机语言的标号一样，是可省略的部分。语句中无必要，标号可以不写。语句中的第二部分为操作符，它是不可省略的。操作符通常也称操作码或指令码，它表示该语句执行什么操作。语句的第三部分是地址部分，地址有时也可不写，如“RET”语句。地址主要用于指示操作数。汇编语言中语句的地址形式是很重要的，以后将会详细讲解。语句的最后一部分是注释部分，用“；”号引入的注释是可选择的，也可以整行均为注释，注释中可写任何字符。

1.2.1 标号与常数

语句中的主要成分除了操作符之外就是标号与常数，下面分别讲解。

一、标号

标号的定义：标号是以大写字母打头，不超过六个字符的大写字母数字串。标号用作指令名，变量名。

例：LAB，LA1，P12，6B2

其中最后一个不是标号，其它均可作标号。

二、常数

CASL汇编语言中允许使用下列四种常数：

1．十进制整数

例：0，1，2，6，…255，256，-32768，65535等。

2．十六进制数

例：#000F，#0FAB，#FFFF，#1468等。

CASL汇编语言中使用的十六进制数必须写成四位，前零不能省略，并且用“#”符号打头。

3．字符数

字符数是用单引号括起来的字符，单引号中的字符，不能为空及逗号，长度也不限制。

字符数的数值是单引号中字符对应的ASCII编码。

例　'ABCD'，'PLEASE'，'CASL'等

′　　′------　　非法的

′A,B′　------　　非法的

字符数′A′与十六进制数#0041其值相等。

4．标号属性常数

标号在语句的标号位置出现时，表示该标号在此程序中有定义，标号在此位置的绝对地址是该标号的属性常数，也称地址常数。

标号在数据语句或指令语句的地址部分出现时，它代表该标号的地址常数。如果地址部分的标号在此程序中无定义，则作为其它程序的START语句的标号处理，其连接工作由操作系统实现。

为了描述方便，我们不妨用：

LAB=100表示标号LAB对应的地址为100。

以上四种常数允许哪些语句引用，各语句有明确规定，以后讲语句时再说明。

1.2.2 说明性语句

说明性语句的功能由汇编程序实现。它有两种：伪指令语句与宏指令语句。

一、伪指令语句

伪指令语句用于描述源程序的结构及数据的形式及结构。共有四种伪指令语句。

1．开始语句

语句形式：〔标号〕START〔执行起始地址〕

该语句表示源程序的开始，一个源程序的第一行必须是开始语句，即开始语句不可省略。标号可作为其它程序进入该程序的入口。“执行起始地址”是该程序中定义的标号。它表示该程序的启动地址，如果开始语句中省略了“执行起始地址”部分，则认为程序从开头执行。

2．结束语句

语句形式：END

该语句表示源程序结束，在源程序的末尾必须书写它。

3．常数语句

语句形式：〔标号〕DC　常数

其中：标号部分可以省略；“DC”为伪操作符，“常数”有四种，它可以写一个十进制常数，或写一个十六进制常数，或写一串字符数，或写一个标号属性常数。

该语句用于定义字常数及该常数在内存的位置。

该语句用于定义四种常数的形式及所占内存的结构见表1.3。

从表1.3定义可看出，除了定义字符数外，其它三种形式都是每个语句只定义一个字常数及该常数在内存的位置。对于定义字符数，该语句定义的内存空间由所定义的字符个数决定，每个字符对应一个字。

```
例：S1    DC    0                P2    DC    #ABCD
    S2    DC    5                P3    DC    #1234
    S3    DC    -1               CS1   DC    CHK
    S4    DC    32769            CS2   DC    'OK'
    P1    DC    #00FF            CS3   DC    'INPUT ERROR'
```

表1.3

常数的种类	写法		说明
	指令码	操作数	
10进常数	DC	n	用n指定的10进数，作为一个字的二进制数存储，如果n超出-32768～32767的范围，则将其低16位存储起来。
16进常数	DC	#h	h作为一个四位16进数(即0～9，A～F)，h指定的16进数作为1个字的二进制数存储(0000<h<FFFF)。
字符常数	DC	字符串	把字符串从左开始的每个字符的字符数据，按每个字符占一个字，依次存入连续的各字的低8位，即第一个字符存入第一个字的第8～15位，第二个字符存入第二个字的第8～15位，依次地，几个字符占几个字地存放，各字的第0～7位里放入0。在字符串中，可以写入空格和任意的图形、文字(详见附录1)，但不能写引号(')。字符串的长度不能是零(即空字符串)。
地址常数	DC	标号	和标号对应的地址作为一个字的二进制数存储，标号在该程序中没有定义时，汇编将保留地址的确定，并由操作系统处理。

4。变量语句

语句形式：〔标号〕DS　个数

该语句用于保留指定字数的内存空间。其中标号部分可有可无，如果有标号，它表示该内存空间的开始地址。此标号实质上是变量名。汇编语言中的变量与高级语言中的变量作用相同，只是前者不区分简单变量与成组变量，并且都是一维的变量。“个数”是指该内存空间所占的字数。如果“个数”为“1”，则该标号可看成简单变量，否则为成组变量。“个数”用大于等于零的十进制数表示。如果“个数”为零，表示该内存空间不存在，但如果此语句有标号，则该标号有效。

```
例：LAA     DS    1
SW          DS    100
VAR         DS    20
SS          DS    0
SY          DS    1
```

二、宏指令语句

CASL汇编语言中有三个宏指令语句，它们用于输入，输出及终止程序的运行。

1。输入语句

语句形式：〔标号〕IN　标号$_1$，标号$_2$

其中标号$_1$是输入缓冲区；标号$_2$是记录输入字符实际个数的变量。标号$_1$与标号$_2$通常由变量语句（DS）定义。

语句执行时，输入设备字符的ASCII编码从输入缓冲区的起始地址开始，一个字符对应一个字地顺序存入，各字的第0～7位放“0”，第8～15位放相应字符的编码（和常数语句之字符数的放法相同），同时把实际输入的字符个数，用二进制数据的形式存入标号$_2$的变量中。

输入字符的结束符（电传打字机的结束符是回车符号），不进入输入缓冲区，也不计个数于标号$_2$中。

下面两种情况标号$_2$中存放“0”或“-1”。

0：输入空记录，即只进入一个结束符。

-1：输入文件结束标志。

输入的字符填不满缓冲区的情况下，输入缓冲区剩下的部分保持输入前的内容，输入的字符超过其长度时，多余的字符被截掉。

例：

```
        ⋮
        .
        IN      BUF, LNG
        ⋮
        ⋮
BUF     DS      80
LNG     DS      1
        ⋮
```

2．输出语句

语句形式：〔标号〕OUT　标号$_1$，标号$_2$

其中标号$_1$是输出缓冲区，标号$_2$是存放输出字符的个数的变量，它以二进制形式存放。标号$_1$之输出缓冲区中的字符存放方式和输入缓冲区中字符存放方式相同，只是不要求字的第0～7位为“0”。

语句执行时，操作系统会自动删除每个字的第0～7位，并且，当需要间隔符（如电传打字机的回车换行符号）时，操作系统也会自动插入。

标号$_1$与标号$_2$可由常数语句定义，也可由变量语句定义。

例：

```
        ⋮
LOUT    OUT     LBUE, LNG
        ⋮
        ⋮
LBUE    DC      'INPUT PLEASE'
LNG     DC      12
        ⋮
LOT     OUT     LB, LN
        ⋮
LB      DS      12
LN      DS      1
```

3．终止执行语句

语句形式：〔标号〕EXIT

该语句的功能是终止程序的执行，控制返回操作　。

例：

```
        EXIT
LPI     EXIT
```

1.2.3 执行性语句

执行性语句在CASL中又称指令语句或简称指令。它与机器指令是一对一的，CASL 汇编语言中共有二十三种指令语句。每个指令语句的功能均由硬件完成，按其使用特点指令语

句可以分为四类。

一、执行性语句的一般形式

执行性语句的一般形式为：〔标号〕操作符〔GRI〔，adr〔，GRK)〕〕〔；注解〕

其中“〔GRI〔，adr〔，GRK)〕〕”为语句的地址部分，这部分也是可选择的。GRI表示通用寄存器（I=0，1，2，3，4），GRK表示变址寄存器(K=1,2，3，4)。adr表示标号或十进整数，32768～65535。

执行语句的地址形式可有如下五种选择：

1．无地址部分

这种语句除标号外，只有一个操作符，如“ret”语句。

2．寄存器地址形式：GRI

这种语句只有一个操作数，并在所示的通用寄存器中。

3．直接地址形式：

直接地址形式有两种

(1) 一地址直接地址形式：adr

这种地址的语句只有一个地址，由adr指示，有效地址E由adr直接得到：E=adr

(2) 两地址直接地址形式：GRI，adr

这种地址的语句有两个操作数，一个操作数在GRI中，另一个操作数在内存中，它的有效地址为：E=adr

4．变址地址形式

变址地址形式也有两种

(1) 一地址变址地址形式：adr，GRK

使用该地址的语句只有一个操作数在内存中，其有效地址为：E=adr+（GRK)

(2) 两地址变址地址形式：GRI，adr，GRK

使用该地址的语句有两个操作数，一个在GRI中，另一个操作数在内存中，其有效地址为：

E=adr+（GRK)

二、执行性语句的功能

这里从使用的特点出发分类讲解各语句的功能。

1．数据传送语句

数据传送语句一共有三种：

(1) LD--装入语句

形式：〔标号〕LD GRI，adr〔，GRK〕

功能：

将所指内存的内容装入到寄存器中

（adr+（GRK））→GRI

例：设(GRI)=100，标号VAR=50，(VAR)=6

内存(100)=65535，(150)=7

```
        LD      GR0, VAR, GR1;(GR0)=7
LA      LD      GR2, 50,  GR1;(GR2)=7
```

```
        LD      GR0, VAR,       ;(GR0)=6
        LD      GR0, 0     , GR1;(GR0)=65535
```

(2) ST--存储语句

形式：〔标号〕ST GRI, adr〔, GRK〕

功能：

将寄存器内容存放到内存中 (GRI) →adr+ (GRK)

```
  例：设(GR0)=32767, VAR=50, (GR1)=100
        ST      GR0, VAR, GR1;内存(150)=32767
LAB     ST      GR0, VAR        ;内存( 50)=32767
        ST      GR1, 50,    GR1;内存(150)=100
        ST      GR1, 0          ;内存(0)  =100
```

(3) LEA--装入地址语句

形式：〔标号〕LEA GRI, adr〔, GRK〕

功能：

将有效地址装入寄存器中并产生标志位 (FR的值)

adr+ (GRK) →GRI，FR的值因GRI的值而变化。

由于地址的可选择性，该语句有几种特殊情况，在编程中很有用处，举例如下。

例1 实现标号对应地址送寄存器。

```
        设LAB=100
        LEA     GR1, LAB        ;(GR1)=100, (FR)=00
LAA     LEA     GR2, LAB        ;(GR2)=100, (FR)=00
```

例2 实现对寄存器赋初值。

```
        LEA     GR0, 0          ;(GR0)=0,       (FR)=01
        LEA     GR1, -1         ;(GR0)=65535,   (FR)=10
        LEA     GR2, 1          ;(GR2)=1,       (FR)=10
```

例3 实现寄存器之间的传送。

```
LEA     GR0, 0, GR1;(GR1)→GR0, 产生FR
LEA     GR1, 0, GR2;(GR2)→GR1, 产生FR
LEA     GR3, 0, GR1;(GR1)→GR3, 产生FR
LEA     GR1, 0, GR3;(GR3)→GR1, 产生FR
```

例4 实现寄存器加或减一个数。

```
设LAB=100
LEA     GR3, LAB, GR3;(GR3)+100→GR3, 产生FR
LEA     GR1, 1    , GR1;(GR1)+1  →GR1, 产生FR
LEA     GR1, -1   , GR1;(GR1)-1  →GR1, 产生FR
LEA     GR2, -2   , GR2;(GR2)-2  →GR2, 产生FR
LEA     GR2, 65530, GR2;(GR2)-6  →GR2, 产生FR
```

例5 实现寄存器加或减一个数后送入另一个寄存器中。

```
设 LAB=50
LEA     GR3, LAB, GR1;(GR1)+50→GR3, 产生FR
LEA     GR1, 1    , GR2;(GR2)+1 →GR1, 产生FR
LEA     GR0, -1   , GR1;(GR1)-1 →GR0, 产生FR
```

2。运算语句

运算语句包括算术运算，逻辑运算及移位语句，共有九种。

(1) ADD--加法语句

形式：　　〔标号〕ADD　GRI，adr〔，GRK〕

功能：

寄存器与内存单元的内容相加送寄存器中，并产生标志位。

(GRI)+(adr(GRK)) →GRI

当(GRI)：
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：设 (GR_1) =65530，(GR_2) =100，(GR_3) =65530，LAB=6

内存 (6) =2，(106) =5

ADD　GR_1，LAB，GR_2；(GR_1)=65535，(FR) =10

ADD　GR_3，LAB，　；(GR_3)=65532，(FR) =10

(2) SUB—减法语句

形式：　〔标号〕SUB　GRI，adr〔，GRK〕

功能：

寄存器内容减去内存单元内容，结果送寄存器中，并产生标志位，不判溢出。

(GRI) - (adr+(GRK)) →GRI

当(GRI)：
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：设 (GR_1) =-6，(GR_2) =100，LAB=6，内存 (106) =5，内存 (6) =2

SUB　GR_1，LAB，GR_2；(GR_1) =-11，(FR) =10

SUB　GR_2，LAB，　；(GR_2) =98，(FR) =00

(3) AND—逻辑乘语句

形况：〔标号〕AND　GRI，adr〔，GRK〕

功能：

寄存器内容与内存单元内容进行逻辑乘运算，结果送到寄存器中，并产生标志位。

(GRI)　AND (adr+ (GRK)) →GRI

当(GRI)：
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：
```
AND  GR0, 1
AND  GR0, LAB, GR2
AND  GR3, 100 , GR1
```

(4) OR—逻辑加语句

形式：〔标号〕OR　GRI，adr〔，GRK〕

功能：

寄存器内容与内存单元内容进行逻辑加（或称逻辑或）运算，结果送到寄存器中，并产

生标志位。

(GRI)　OR (adr+ (GRK)) →GRI

当(GRI)
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：
```
OR    GR0, LAB, GR1
OR    GR2, LAB
OR    GR3, 100, GR2
```

(5) EOR--异或语句

形式：〔标号〕 EOR GRI, adr〔, GRK〕

功能：

寄存器内容与内存单元内容进行逻辑异或运算，结果送到寄存器中，并产生标志位。

(GRI)　EOR (adr+ (GRK)) →GRI

当(GRI)
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：
```
EOR    GR1, LAB, GR2
EOR    GR0, LAB
EOR    GR3, 100
EOR    GR0, 2,   GR2
```

(6) SLA--算术左移语句

形式：〔标号〕 SLA GRI, adr〔, GRK〕

功能：

先求出有效地址E=adr+ (GRK)

然后把GRI的内容，除符号位（第0位）外，左移E次，右边的空位补零。最后根据移位后GRI的内容产生标志位即：

当(GRI)
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

例：
```
SLA    GR0, 1; (GR1)符号位不动，其它左移1位；右边补一个"0"。
SLA    GR1, LAB
SLA    GR1, 2,  GR2
```

(7) SRA--算术右移语句

形式：〔标号〕 SRA GRI, adr〔, GRK〕

功能：

先求出有效地址 E=adr+ (GRK)

然后把GRI的内容连同符号位一起向右移E次，左边的空位按符号位的值填补。最后根据移位后GRI的内容产生标志位即：

当(GRI)
- =0　时 (FR) =01
- >0　时 (FR) =00
- <0　时 (FR) =10

```
例:     SRA     GR0, 2        ; (GR1)右移两位, 左边补两个符号位
        SRA     GR1, 1, GR2
        SRA     GR3, LD, GR2
```

(8) SLL--逻辑左移语句

形式: 〔标号〕 SLL GRI, adr〔, GRK〕

功能:

先求出有效地址E=adr+ (GRK)

然后把GRI的内容左移E位, 右边空的位补零。

最后根据移位后GRI的内容产生标志位即:

$$\text{当(GRI)}\begin{cases} =0 & \text{时 (FR) =01} \\ >0 & \text{时 (FR) =00} \\ <0 & \text{时 (FR) =10} \end{cases}$$

```
例:     SLL     GR0, 1,       ; (GR1)左移一位, 右边补一个 "0"
        SLL     GR3, 2, GR2
        SLL     GR1, LD
```

(9) SRL--逻辑右移语句

形式: 〔标号〕 SRL GRI, adr〔, GRK〕

功能:

先求出有效地址E=adr+ (GRK)

然后把GRI的内容右移E位, 左移空的位补零。

最后, 根据移位后GRI的内容产生标志位, 即:

$$\text{当(GRI)}\begin{cases} =0 & \text{时 (FR) =01} \\ >0 & \text{时 (FR) =00} \\ <0 & \text{时 (FR) =10} \end{cases}$$

```
例:     SRL     GR0, 5        , (GR0)右移五位, 左边补五个0
        SRL     GR1, 2   GR2
        SRL     GR3, LD GR1
```

3. 比较、转移语句

比较语句有两种, 它们专门用于产生标志位, 转移语句有五种, 它们用于识别标志位, 句本身对标志位无影响。

(1) 算术比较语句

形式: 〔标号〕 GPA GRI, adr〔, GRK〕

功能:

计算有效地址E=adr+ (GRK)

把GRI与内存E的内容看成有符号数进行比较, 比较的结果产生标志位, (GRI) 与 (E) 不变。

产生标志位的值如下所示

$$\text{当(GRI)}\begin{cases} = (E) & \text{时 (FR) =01} \\ > (E) & \text{时 (FR) =00} \\ < (E) & \text{时 (FR) =10} \end{cases}$$

```
例:     CPA     GR1, LAB, GR2; (GR1)与(LAB+(GR2))比较
        CPA     GR1, 100      ; (GR1)与(100)比较.
        CPA     GR0, 2,   GR3; (GR0)与((GR3+2)比较.
```

(2) CPL--逻辑比较语句

形式：〔标号〕 CPL GRI, adr〔, GRK〕

功能：计算有效地址：E=adr+(GRK)

把GRI与内存E的内容看成无符号数进行比较，比较的结果产生标志位，GRI与E的内容不变。产生标志位的值如下所示：

当(CRI) { =(E) 时 (FR) =01
        { >(E) 时 (FR) =00
        { <(E) 时 (FR) =10

例：

```
CPL  GR1, LAB, GR2; (GR1) 与 (LAB+ (GR2)) 比较.
CPL  GR2, 100,     ; (GR2) 与 (100) 比较.
CPL  GR0, 2,   GR3; (GR0) 与 ( (GR3) +2) 比较.
```

以上两个语句的区别在于把寄存器与内存单元内容看成是什么数进行比较，因而相同的两个数用不同的比较语句，产生的结果 (FR的值) 是不同的。

譬如 (GR1) =65535，内存 (100) =5。则有下列语句的不同结果：

```
CPL   GR1, 100; (FR) =00
CPA   GR1, 100; (FR) =10
```

(3) JPZ--非负转移语句

形式：〔标号〕 JPZ adr〔, GRK〕

功能：计算有效地址 E=adr+ (GRK)

判断FR的条件，如果 (ER) =00或者 (FR) =01则实现转移 (PC)=E

否则：(PC)=(PC)+2继续执行。

例：

```
JPZ  LAB
JPZ  LAB, GR1
JPZ  0,   GR2
JPZ  1,   GR3
JPZ  200,     ; 满足条件时, (PC) =200
```

(4) JMI--负转移语句

形式：〔标号〕 JMI adr〔, GRK〕

功能：计算有效地址 E=adr+(GRK)

判断FR的条件，(如果 (FR) =10则实现转移 (PC) =E

否则 (PC) = (PC) +2 继续执行。

例：

```
JMI  LAB
JMI  LAB, GR1
JMI  0  , GR2
JMI  -2, GR3
JMI  100      ; 满足条件时, (PC) =100
```