全国计算机软件专业
技术资格和水平考试复习辅导教材

CASL汇编语言程序设计及题解

朱慧真 编著

221

PEKING UNIVERSITY PRESS · 1979 ·
北京大学出版社

全国计算机软件专业
技术资格和水平考试复习辅导教材

CASL 汇编语言
程序设计及题解

朱慧真 编著

北京大学出版社
北京

内 容 简 介

本书是全国计算机软件专业技术资格和水平考试的复习辅导教材。

CASL 汇编语言已列入我国“计算机软件人员水平考试大纲”中规定使用的汇编语言，是高级程序员级考试的必考科目，也是程序员级考试任选的五种语言之一。

本书针对考试大纲的要求，着重讲述CASL 汇编语言基础、CASL 语句功能分析，还用大量的实例分析CASL 的常用算法和编程的基本技巧；并通过典型题例分析和 91—93 年度我国软件专业技术资格和水平考试题解，以及相当数量的日本全国统考题解，还有大量的练习题和答案，使考生能在复习 CASL 汇编语言的过程中，掌握基本知识，熟悉考题类型和模拟解法，提高解题能力，顺利通过考试。

本书是我国计算机软件专业人员参加高级程序员和程序员全国统考的一本很实用的复习辅导教材。

新登字(京)159 号

JS446/89.01

图书在版编目（CIP）数据

CASL 汇编语言程序设计及题解/朱慧真编. -北京：北京大学出版社，1994. 7

ISBN 7-301-02547-5

Ⅰ. C… Ⅱ. 朱… Ⅲ. 微型计算机，CASL-汇编语言-程序设计-问题解答 Ⅳ. TP312

书　　名：CASL 汇编语言程序设计及题解

著作责任者：朱慧真

责 任 编 辑：杨锡林

标 准 书 号：ISBN7-301-02547-5/TP・233

出 版 者：北京大学出版社

地　　址：北京市海淀区中关村北京大学校内　100871

电　　话：出版部 2502015　发行部 2559712　编辑部 2502032

排 版 者：蓝地公司激光照排

印 刷 者：北京飞达印刷厂

发 行 者：北京大学出版社

经 销 者：新华书店

版本记录：787×1092 毫米　16 开本　12.75 印张　315 千字

1994 年 7 月第一版　1994 年 7 月第一次印刷

定　　价：14.20 元

前　言

CASL 汇编语言是我国“计算机软件专业技术资格和水平考试”中所用的汇编语言。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。

该汇编语言的基础教材在培训班中已使用多年，在这些年的教学实践中，作者本人深感有必要再写一本较全面的，更深入浅出的 CASL 汇编语言的教学材料。本教材具有下列特点：

1．增加了对语句功能的分析及应用。

语句功能是 CASL 汇编语言程序设计的基础，试题中专考语句功能的填空约占一半以上，而多数学员中仅看语句功能的一般讲解，还不能掌握住这些语句的功能特点，故必须增加对语句功能的分析及应用的讲解。

2．讲解中介绍了许多典型的小程序段。

由于 CASL 汇编语言语句种类少，功能单一，故它的程序结构形式也较为固定，例如实现乘与除运算的程序结构形式就曾在试题中重复出现多次，熟练地掌握住这些常用的小程序段，将会提高解题的速度。

3．增加了许多练习题。

4．有大量的题例分析及近三年来我国软件专业技术与资格考试的题解，题例中有相当数量的日本全国统考试题。

这些对增加学员汇编常用算法及编程技巧的知识，提高他们的解题能力有很大帮助。

朱慧真

1994 年 3 月

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

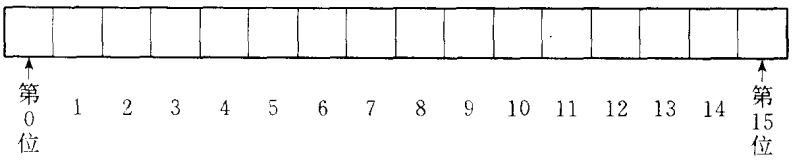
第一章　CASL 的硬件基础

CASL 汇编语言是一种抽象的汇编语言。它定义在抽象的计算机 COMET 上。它依赖于 COMET 机的特性仅有四个方面：存储器、寄存器、指令形式及数据表示。

1.1　主存储器及其分配方式

一、主存储器

主存储器又称内存。CASL 的内存按字编址，字长为十六位 (bit)，字中各位的编号如下所示：



内存容量为 2^{16}，即十六进制的“10000”。

内存地址为十六位，编号为 0—65535，或用十六进制表示为：0000—FFFF。

二、内存的分配方式

内存采用绝对地址的分配方式，程序中也使用绝对地址。程序与数据使用同一个存储空间，其分配的位置与源程序语句的顺序一致。

1.2　程序中访问的寄存器

一、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、比较及移位等语句存放数据，其中 1—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。

二、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。

三、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
FR的值	00	10	01

表 1.2

	(GR) 与 (有效地址) 比较		
	>	<	=
FR的值	00	10	01

1.3 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。

一、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。

二、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） 即直接地址数加上所示变址器的内容。

注意：(1) 在COMET中有效地址E是绝对地址。

(2) 以后凡是寄存器或内存地址外加圆括号时，表示其内容。例如GR1的内容为“5”时，可写成（GR1）=5。

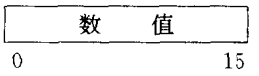
1.4 数据的机内表示

COMET是定点计算机。机内数据字长为十六位，各位的编号与内存字各位的编号相同。

数据有三种：

一、十六位无符号数

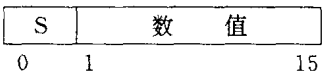
字中十六位全是数值，数据范围在0—65535。



二、十六位有符号数

采用补码表示法，其中S位（0位）为符号位，

$$\text{当 (S)} = \begin{cases} 0\text{时，} & \text{该数为正数} \\ 1\text{时，} & \text{该数为负数} \end{cases}$$



有关补码的性质及运算读者必须掌握，它属于计算机原理的数制及编码部分。这里仅写出其常用的运算。

(1) 正数的补码是其本身，设x为正数，则可记为$[x]_{补}=[x]_{原}$。

(2) 负数的补码是原码除符号位之外，每位求反后，再加“1”。

(3) 以上方法也适用于由补码求原码。

三、字符数

字符数是计算机输入输出设备使用的字符编码，字符编码由ASCII编码表所定义，详见附录1。

每个字放一个八位的字符编码，存放于低八位、高八位置零或无效。例如：字符“A”的编码为十六进制数“41”，设字符“A”进入内存100单元中，则（100）＝#0041。

练 习 一

1. CASL中能使用几个寄存器？它们的名字是什么？哪个寄存器不能作变址器。
2. 标志寄存器的名字是什么？有几位？各位的意义是什么？
3. CASL中每条指令语句占多少二进制位？指令中的有效地址是否是绝对地址？
4. 请写出下列各数的补码（用十六进制表示）。

(1) −1，(2) −2，(3) −128，(4) −32768，(5) −27

5. 对以下列出的十六进制数，按表中指出的不同理解填其值（用十进制数表示）。

十六进制数	无符号数	有符号数
FFFF		
FFFE		
8000		
0003		
7FFF		

第二章　CASL 汇编语言

这一章主要讲 CASL 的语法及语句功能，最后举例说明源程序对应的内存安排。

CASL 汇编语言中有两种类型语句：说明性语句及执行性语句。说明性语句又称伪指令语句；执行性语句又分为宏指令语句及指令语句两种。我们可把指令语句看成是每条语句对应一条目标指令，占用两个字长的内存空间。

CASL 汇编语言中语句一般形式为：

〔标号〕　　操作符　　〔地址〕　　〔；注释〕

其中第一部分是标号，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他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。地址主要是用于指示操作数。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“;”号引入的注释也是可选择的，也可以整行均为注释，注释中可写任何字符。

2.1　标号与常数

语句中的主要成分除了操作符与地址之外就是标号与常数，下面分别讲解。

一、标号

标号的定义：

标号是以大写字母开头，不超过六个字符的大写字母数字串，并且在标号位置出现且仅出现一次。标号用于作指令名、变量名、常数名及宏指令名。

例　LAB，LA1，P12，6B2

其中最后一个不是标号，其他全可作标号。

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串，单引号中的字符不能为空及引号，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'等为合法的。

' '——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，标号在此位置的地址（也就是绝对地址）是该标号属性常数，也称地址常数。从而对标号有下列结论：

(1) 指令语句中定义的标号，该标号的属性常数是该指令两个字的开始地址，更进一步可把标号看成相应指令语句始址的符号化表示，或称符号地址。

(2) 同理，变量语句中定义的标号是该变量始址的符号地址；常数语句中定义的标号是该常数始址的符号地址。

标号在语句的地址部分出现时，通常它应在该程序中有定义，并且它在地址中代表该标号的属性常数，即标号在地址中作常数（地址常数）使用。如果语句地址中的标号在该程序中无定义，在语法正确的情况下，该标号肯定是在其他程序的开始语句中有定义，其作用见下面开始语句的讲解。

综上所述，在汇编语言中可认为标号是个地址。为了描述方便，我们不妨用下列表示：LAB=100，表示标号 LAB 对应的地址（属性常数）是“100”。

以上四种常数允许哪些语句引用，各语句均有明确的规定，以后讲语句时将有说明。

2.2 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结构。它的功能由汇编程序实现。CASL 中共有四种伪指令语句。

一、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。其中“标号”可作为其他程序进入该程序的入口，其连接工作，由操作系统实现。“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，则认为该程序是从头开始执行。

二、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。

此为试读，需要完整PDF请访问：www.ertongbook.com

三、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，或写一个字符数，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，也可说是定义内存的内容。

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。

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 指定的十进制整数，作为一个字的二进制数存储，如果 n 超出 −32768—65535 的范围，则将其低十六位存储起来。
16 进常数	DC	#h	h 表示四位十六进制数（即 0—9，A—F），以二进制形式存储于一个字中（0000≤h≤FFFF）。
字 符 数	DC	'字符串'	把字符串从左开始的每个字符的字符数，按每个字符占一个字，依次存入连续的各字的低位，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，几个字符占几个字的存放。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。

从以上定义可看出，除了定义字符数外，其他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　设 S1＝100，下列常数语句的地址是确定的数：

```
S1    DC    0          ;    (100) = #0000
S2    DC    5          ;    (101) = #0005
S3    DC    −1         ;    (102) = #FFFF
S4    DC    32769      ;    (103) = #8001
P1    DC    #00FF      ;    (104) = #00FF
P2    DC    #ABCD      ;    (105) = #ABCD
P3    DC    #1234      ;    (106) = #1234
CS1   DC    'OK'       ;    (107) = #004F
                       ;    (108) = #004B
CS2   DC    'INPUT ERROR';    (109) = #0049
                       ;    (110) = #004E
```

```
                    ;     (111) = #0050
                          ⋮
                    ;     (119) = #0052
CS3   DC      S2    ;     (120) = #0065
```

四、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     ;   设 LAA=100
SW     DS    100   ;   则 SW=101
VAR    DS    20    ;     VAR=201
SS     DS    0     ;     SS=221
SY     DS    1     ;     SY=221
```

2.3　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。

一、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（由操作系统完成）。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：　只输入文件结束标志。

输入的字符长度小于缓冲区长度的情况下，输入缓冲区剩下的部分保持输入前的内容，输入的字符的个数超过缓冲区长度时，多余的字符被截掉。

例

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

二、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，操作系统也会自动的插入。

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

三、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，控制返回操作系统。

例

```
        EXIT
LFI     EXIT
```

练　习　二

1. 指出开始语句的三个作用。
2. 对一个源程序来说，开始语句是否可以省略？结束语句是否是必须的。

3. 指出语句“END”与语句“EXIT”有何不同。

4. 指出语句：“DC 100”与“DS 100”的区别。

5. 指出语句：“DC 'ABCDEF”占几个字？其内容是什么（用十六进制数写出）？

6. 简述输入输出语句的功能，并指出它们不相对应的功能。

7. 写出缓冲区始址为：“BUF”，输入 8 个字符的输入语句及相应的语句。

8. 写出缓冲区始址为：“BOUT”，输出信息为：“INPUT△PLEASE”的输出语句及相应的常数语句。（注：其中符号“△”，表示空格字符，以下也按此约定书写。）

2.4 指令语句的形式及种类

指令语句有时简称指令。它与机器指令是一对一的。CASL 汇编语言中共有 23 种指令语句。指令语句的功能由硬件完成。

一、指令语句的一般形式

指令语句的一般形式为：

〔标号〕　操作符　GRI，　adr〔，GRK〕　〔；注释〕

其中“GRI，adr〔，GRK〕”为语句的地址部分，该部分按语句的类型有不同的选择。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 表示标号或十进制整数，范围在－32768—65535。

指令语句的地址形式可有如下五种选择：

1. 无地址形式

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

这种地址形式有两个操作数，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，并在内存中，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、指令语句的种类

指令语句从使用角度可分为四类：数据传送语句、运算语句、比较与转移语句及堆栈操作语句，以下按类讲解其功能。

2.5 数据传送语句

数据传送语句一共有三种：

一、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（GR1）=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
```

二、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
```

三、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。

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

由于 CASL 汇编语言中的地址运算是十六位的运算，而它的数据运算也是十六位的，并且存放地址的变址器（GR1，GR2，GR3，GR4）也可作存放数据的寄存器使用，因此语句"LEA"不仅能装入地址，而且能装入数据；更进一步它不仅能进行地址运算，而且能进行数据运算。又由于该语句执行时能产生标志位，所以它还能起到比较作用。以下分别举例说明。

1. 实现标号对应地址送寄存器

语句形式：　LEA　GRI，标号

例　设标号 LAB＝100

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

2. 实现对寄存器赋初值

语句形式：　LEA　GRI，数

例

```
    LEA   GR0,   0    ;   (GR0) =0,      (FR) =01
    LEA   GR1,   -1   ;   (GR1) =65535,    (FR) =10
    LEA   GR2,   1    ;   (GR2) =1,      (FR) =00
```

3. 实现寄存器之间的传送

语句形式：　LEA　GRI，0，GRK

例

```
    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
```

4. 实现寄存器加或减一个数

语句形式：　LEA　GRK，数，GRK

例　设　LAB＝100

```
LEA   GR3,   LAB,   GR3   ;   (GR3) ＋100 ⟶GR3    产生 FR
LEA   GR1,   1,     GR1   ;   (GR1) ＋1 ⟶GR1      产生 FR
LEA   GR3,   -5,    GR3   ;   (GR3) －5 ⟶GR3      产生 FR
LEA   GR2,   65535, GR2   ;   (GR2) －1 ⟶GR2      产生 FR
```

5. 实现寄存器加或减一个数后送另一个寄存器中

语句形式：　LEA　GRI，数，GRK

例　设　LAB＝50

此为试读，需要完整PDF请访问：www.ertongbook.com