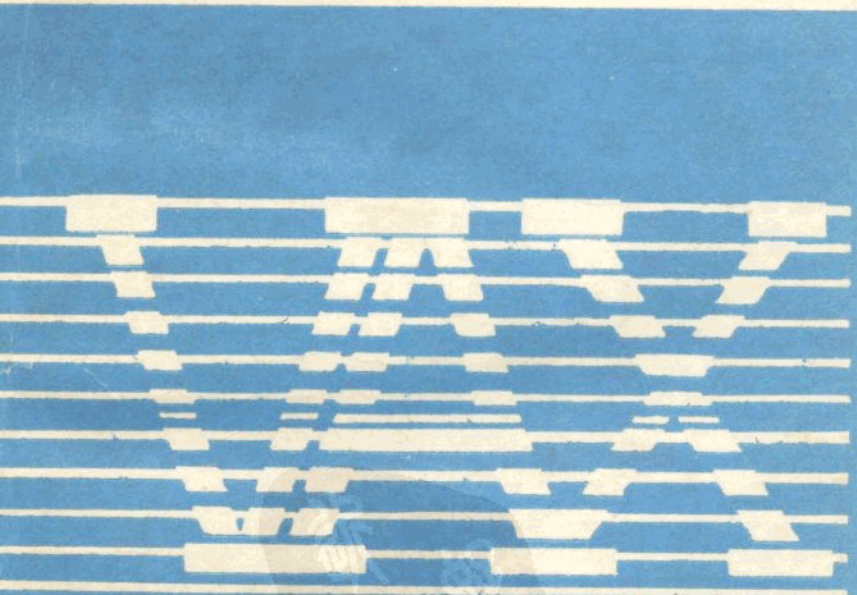
VAX汇编语言程序设计

黄水松　林子禹　肖邵武　编著



武汉大学出版社

前　言

随着计算机科学和技术的发展，尽管高级程序设计语言愈来愈多，越来越高级，但用汇编语言编写程序仍然是计算机工作者必不可少的基本功训练。首先，汇编语言反映了计算机硬件系统的所有基本功能和实现方法。例如，操作系统的核心程序和I/O设备驱动程序、一些应用系统中直接和硬设备有关的程序，一般都需要用汇编语言编写，汇编语言能处理那些高级语言无法处理的问题。其次，汇编语言程序的空间和执行效率均高于高级语言程序。第三，用汇编语言进行程序设计，是深入了解和熟悉计算机的有效途径。

本书有一定特点，它以具有丰富的指令系统和寻址方式的VAX MACRO汇编语言为对象，结合作者长期的教学和工作实践，详细讲述了计算机汇编语言程序设计的原理、方法和技巧。

全书共分十三章，其编排如下：

第一章讲述了VAX计算机的指令特点、格式和寻址方式。

第二章阐述了汇编语言的基础知识。

第三章至第十章讲述了汇编语言程序设计的基本方法和技巧。特别注重模块化和结构化程序设计方法。对于程序的三种基本控制结构作了详细叙述，并且列举了与这些结构对应的FORTRAN、BASIC、PASCAL语言程序，以及VAX的汇编语言程序。

第十一章是为调试汇编语言程序而编写的，本章的许多例子是调试程序时的经验总结。

第十二章介绍了一些特殊指令，这对系统分析和系统维护人员有较大的参考价值。

第十三章讲述了反汇编技术，这是作者多年来科研成果和工作经验的总结，为对此感兴趣的读者起到抛砖引玉的作用。

各章均附有一定数量的习题，供读者复习巩固所学知识之用。

书中列举出了大量的程序实例，它们都是独立的、完整的，并且经过了上机验证，完全可以运行。读者在上机运行这些实例时，有的可能看不到显示的结果，如有必要，请参阅第九章或第十一章给出的有关内容。

本书不仅可作为大专院校有关专业的教材，而且对专业技术人员来说也是一部良好的参考书。

由于作者水平有限，书中错误与不妥之处敬请读者批评指正，不胜感谢。

作　者

1990.11

目　录

第一章　指令格式与寻址方式

§1.1　概述

VAX计算机是美国数字设备公司（常简称为DEC或DIGITAL）生产的一种通用数字计算机，它是具有共同特性的各种型号的计算机的总称。VAX是Virtual Addresse Xtension 的缩写，其意为虚拟地址扩展。该机在设计时扩充了PDP—11计算机的指令系统和寻址能力，是一种崭新的体系结构。

VAX系统结构最为明显的表现形式是它的指令系统，三百多条指令使汇编级的程序员对计算机操作拥有广泛的控制权。该指令系统具有正交性（即独立性），即它所要完成的操作（如“加”）、所用的数据类型（如“长字”）及寻址方式（如“自增型”）均由汇编程序分别加以处理，这将有助于快速、高效和易于实现汇编程序。除此之外，每条指令能在从零至大到它能适应的自然数上进行运算，还有某些由高级语言产生的递归也由硬件来实现，以便用一条指令就能加以处理。

VAX的中央处理机中包含有16个32位的通用寄存器，前15个分别命名为R0～R14，R12、R13和R14，指定了专门的名字，分别为AP、FP和SP。第16个通用寄存器叫做PC，作为程序计数器使用。大部分指令的操作数要么是

存放在这些通用寄存器之中，要么是通过它们进行访问的。这就为程序员提供了一种高速简便处理数据的方法。VAX的指令可以按下列任何一种方式来使用通用寄存器。

1．作累加器。所要处理的数据存于寄存器内。

2．作指针。寄存器的内容是操作数地址，而不是操作数本身。因为寄存器内容经常是数据结构的基地址，所以这种形式常称为基址寄存器。

3．作自动步进存储单元的指针。例如，正向自动步进连续单元称为自增编址；反向自动步进连续单元称为自减编址。这些方式对于处理表格数据和堆栈是非常有用的。

4．作变址寄存器。用变址方式产生的偏移量与基操作数地址相加，得到变址单元。

VAX计算机的内存地址是长字，有四个字节长，所以地址空间信息字节是按32位地址定位的。内存被安排为一系列字节。每个字节都可寻址，它由一串8个二进制位组成。字节中的位按0～7从右至左编号。除此之外，所有的数据都由一到几个字节组成。它有5种不同的类型：字节、字（两个相邻的字节）、长字（4个相邻的字节）、四倍字（8个相邻的字节）和八倍字（16个相邻的字节）。所有类型的负数均以二进制补码形式存放。另外，在VAX中不存在按长字边界强制对齐的问题，即使大于一个字节的数据项，仍可按任意字节边界存放。

学习本章时，将逐渐熟悉典型的VAX指令的可变长度指令格式，这种指令格式包括一个字节的操作码，后跟0到6个操作数说明符。每个操作数说明符含有指令执行时定位操作数所需要的信息。在学习VAX所实现的各种不同的寻址方式时，特别要学习辨认和使用每种寻址方式所关联的汇

编写法。还将学习在特定的应用中怎样用这些寻址方式高效率地访问数据。所有这些，使得程序员能够选择在时间和空间上都是高效率的指令和寻址方式组合的各种技术。

只有牢固地掌握了数据类型、指令系统和寻址方式，才能有效地研究指令的编写，学习指令在存储器中的表示方法，进而编写复杂的汇编语言程序，理解计算机的工作原理。

§1.2 信息单位与数据类型

虽然计算机的主存结构允许每个存储周期存取固定的位数，但是表示数据的位数是随着数据类型的不同而变化的。通常在一个计算机系统中，定义若干基本的信息单位以适应该系统中使用的数据类型。有些计算机系统把主存的字长设计成等于保存一条指令或一项最常用的数据类型的信息单位，这种机器称为可按字寻址的，或面向字的机器。而另外一些计算机，存储器的存储单元按字节编址，并以字节为数据处理单位来进行传送和运算，这种机器称为可按字节寻址的，或面向字节的计算机。

1.2.1 基本信息单位

VAX计算机被设计成允许用户直接存取各种长度的信息单位，但是它的基本信息单位是8位的字节。VAX常被称为可按字节寻址或面向字节的计算机。它的指令系统除了对字节可进行操作外，还能对位组和字节所组成的其他信息单位进行操作。多字节单元有16位字（两字节）、32位长字（四字节）、64位偶长字（八字节）和128位的八倍字（十

六字节）几种，如图1.1所示。除了字节之外，其余的每一种

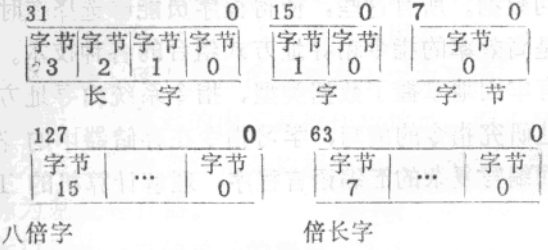


图1.1 VAX的基本信息单位

信息单位都是一系列在内存中相邻的字节。每个信息单位的地址都等于这一组内最低位字节的地址。要注意的是，任何一个信息单位内的各位的编号都是从最右边的一位算起，最右边的位为第0位，最左边的位为最高有效位。

尽管字节是最小的可寻址单位，但实际上常把4位组（字节的一半）称为半字节，它可存放一个十六进制数字。

在VAX中还有另外一种可变长度位字段信息单位。位字段的特点在于其基本可编址单位不是以字节计算长度，而是以位来计算长度，一个位字段可以是一组长度为0到32的相邻位，而且可以处于相对于一个字节的初始端的任何位置。位字段通常用于把多个信息域紧密地组装在一起。例如，每个字节可组装8个布尔值（有时称为“标志”）。在这个意义上，位字段指令使程序员能处理比字节还小的字段。

字节、字、长字、倍长字和八倍字都是VAX的基本信息单位。数据类型以这些基本信息单位为基础，并且对它们所包含的各个位的意义进行解释，所有信息单位都包含一串二进制数字，但是它可以表示整数、字符、实数等。VAX计算机具有处理多种数据类型的能力，这就使得程序员（或

编译程序）有可能应用最适合其需要的类型产生出非常简练程序。此外，VAX还有一整套指令可以很方便地用来把一种数据类型转换成另一种数据类型，从而减少了许多程序的复杂性。

1.2.2 数据类型

我们知道，信息单位是固定长度的位串，只要给这些单位赋予不同的解释，就可以定义出各种不同的数据类型，这点是很重要的。因为相同的信息单位可能解释成区别很大的数据类型；同样，不同的信息单位也可用来表示相同的数据类型。

VAX计算机的指令系统可使用如下几种主要的数据类型：整数、浮点数、字符、压缩十进制数、可变长度位字段、数串和队列。对于每种数据类型，操作选择信息将有关数据的长度及其解释立即通知处理机，以便处理机能接着把位字段作为用户定义的字段长度，并从已知字节地址的相对位置开始进行处理。下面我们将讨论整数、浮点数、字符串，而其他数据类型将在以后各章叙述。

图1.2列出了VAX各种数据类型及其进一步的划分。

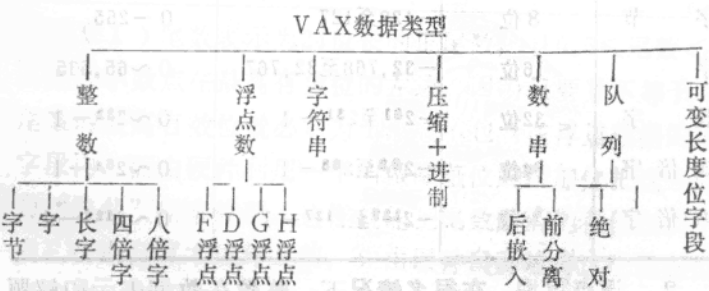


图1.2 VAX数据类型

1．整数　VAX中的整数数据是按二进制形式存储的它可以表示为无符号二进制数和带符号二进制补码数。VAX支持8位、16位、32位、64位和128位长度的整数数据类型，相应地存储在字节、字、长字、四倍字和八倍字内。整数存储表示法的选择决定了最大的数值以及表示法的效率，也就是有效使用的位数。表1.1列出了各种长度的整数所表示的数值范围。用户可根据其应用需要选择合适的长度，从而最有效地利用存储器。VAX有一套完整的指令可用于整数加、减、乘、除、求反以及移位，但是不完全支持四倍字和八倍字算术运算。另外，还有一组完整的条件转移指令，可使程序员根据前面的算术运算结果来改变控制流。由于二进制补码数的最高有效位总是用来表示符号，所以硬件可以很容易地测出正负，实现控制流的转移。

表1.1　整数表示方法

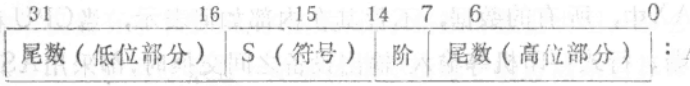
数据类型	长　度	范围（十进制）	
整　数		有符号	无符号
字　节	8位	−128至127	0 −255
字	16位	−32,768至32,767	0～65,535
长　字	32位	-2^{31}至$2^{31}-1$	0～$2^{32}-1$
四倍字	64位	-2^{63}至$2^{63}-1$	0～$2^{64}-1$
八倍字	128位	-2^{127}至$2^{127}-1$	0～$2^{128}-1$

2．浮点数据　在很多情况下，整数在数据表示和解题中十分有用，但在各种科学应用方面，它们常常缺少科学应

用的多样性所必需的动态范围，尽管整数能以足够的精度表示数据，但是数据的值或中间结果都可能超出其数据字长度所允许的范围。浮点数据则为一个巨大的实数集合提供了数据类型。

VAX有标准的F浮点（单精度）和D浮点（双精度）数据类型，扩展范围的G浮点和H浮点数据类型则作为选择项提供。每种浮点数据类型由一组连续的几个字节组成，可以开始于任意字节边界。这四种数据类型总是用它们的起始地址（A）作为参考，A是含有0位的那个字节的地址。每种浮点数据都由符号字段（S）、尾数字段和阶字段组成。

由于考虑兼容性的原因，VAX的浮点数约定是以PDP—11的格式为基础的。在VAX中，单精度浮点数存放在四个相邻的字节内，且用位0所在的字节来寻址。这个长字的第15位被指定为该数的符号，一个八位的阶码字段把它的符号与它的尾数高位字段（即高有效位部分）隔开，尾数的低位字段则放在其余的十六位中。VAX单精度浮点数按如下形式存放：



（1）尾数表示为24位长的正尾数，且0.5≤尾数＜1，二进制小数点在最高有效位的左边，因为只要数不等于0，尾数的最高有效位就必定为1，它不包含在浮点数据的尾数字段内，而由硬件利用一个叫做隐蔽位的附加位把这一位作为“隐蔽”位来处理。这样就能把尾数经济地存储在23个位内。在进行算术运算之前，先由硬件自动地将这个隐蔽位恢复起来。尾数的这种形式称为规格化的尾数。

（2）阶码占8位二进制，采用8位偏置正整数存放，

阶进行运算时，把它减去128所得结果表示2的幂数，将2的这一幂乘以尾数得到浮点数的真值。

（3）位15是符号位，当符号位为0时，数的符号为正，当S＝1时，数的符号为负，用32位字表示的浮点数如下：

$$X=(1-2\times S)\times 尾数\times 2^{(阶-128)}$$

其中

$$2^{-128}=2.939\times10^{-39}\leqslant|X|<2^{127}=1.701\times10^{38}$$

用于尾数的这些位保证了大约7位十进制数字的精度，在双精度浮点数中，另外一个32位的字附加在尾数部分，从而可获得16位十进制数字的精度，这样的双精度浮点数要求8个相邻的字节。注意，阶为零且符号位为零（S＝0），则不管尾数的值是多少，都认为数X为0.0。

3．字符串　计算机可用代码表示非数值信息。由于所有的信息单位存放的都是二进制数字，所以只要给每个字符指定一个数值代码，这就可以表示字符串。VAX采用美国信息交换标准码（ASCII），ASCII字符集包含有大小写英文字母，数字（0～9）、标点符号和专用控制字符。在VAX中，所有的数据，不管其在内部如何表示，当CPU和终端、行式打印机等输入/输出设备之间交换时，都采用ASCII码。

存储字符串时，每个字符占用一个字节，也就是使用了两个十六进制数字。字符串“VAX—11”存放在图1.3中。

1	1	—	X	A	V	
31	31	5F	58	41	56	：A

图　1.3

这个字符串的第一个字节存放字符V，它的符号地址为A，这个符号地址同时也就是整个字符串的引用地址。

对字符串不能执行算术或逻辑操作，VAX设计了可以用于处理相邻字符串的指令，这些指令可对字符串数据进行测试、比较、扫描、符合、排列、变换、拼写、插入及删除。字符串是存储器中相邻的字节序列，它具有两种属性，即地址（串的第一个字节的地址）和以字节为单位（也就是以字符为单位）的长度。在VAX中，字符串的长度可为0～65 535个字节，长度为0的串称为零串，它不包含字节，而且不用存储器存放，因而地址是无效的。

§1.3 指令系统

1.3.1 指令特点

我们知道，指令在计算机内部是用二进制编码表示的。指令执行时，只有当执行指令所必需的信息在指令中都已规定好之后，计算机才能正确地执行。这些规定包括：操作的类型、数据的格式、操作数的位置以及结果的存放位置。例如，如果机器有字节整数、字整数、长字整数等数据类型，那么，指令系统中就应该有一条加法指令按统一的方法来运算每种数据类型。如果长字可以存放整数或地址，那么这两种对象应当得到不同的处理。这些就是我们在用汇编语言编程序时必须掌握的。

VAX有一整套指令，它不仅能对原始数据类执行操作，而且还能把一种数据类型转换成另一种数据类型。除此以外，VAX为数值数据类型处理和地址处理提供了不同的指令。最后，指令执行结果的可测试性可用于判断指令执行的结果，通过这种方式，可以改变指令系列的执行顺序，针对不同数