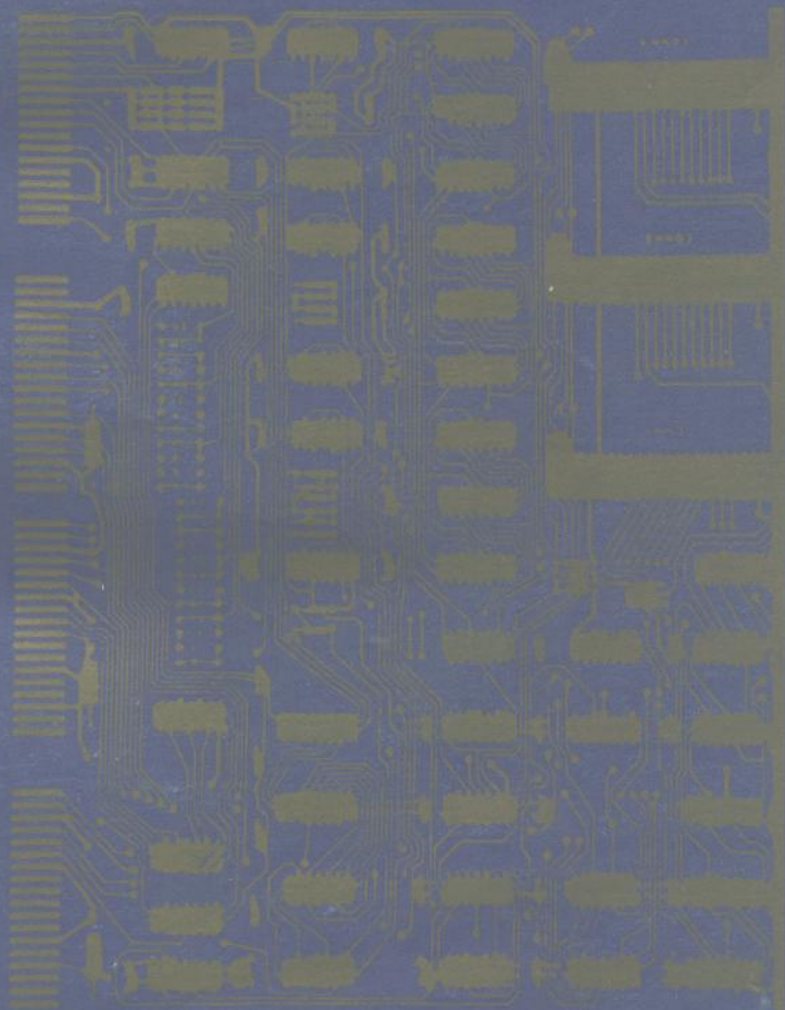


计算机组织与汇编语言 程序设计 (VAX-11)

〔美〕S·伊尔阿斯福里 O·约翰逊 W·K·金 著



出版社

计算机组织与汇编语言 程序设计 (VAX-11)

〔美〕 S. 伊尔阿斯福里 O. 约翰逊 W. K. 金 著

王祖永 译

科学出版社

1987

JSS51/17

内 容 简 介

本书以 VAX-11 系统为实例, 阐述计算机汇编语言程序设计的原理。在内容编排上由浅入深, 文例并重, 旨在进一步地引导读者, 使之学会编写 VAX-11 汇编语言程序并掌握汇编语言程序设计的一般原理。

全书共分十章和六个附录, 主要内容包括: 计算机硬件组织、汇编基础、VAX-11 MACRO 语言、寻址方式、子程序、汇编程序、宏和条件汇编、高等数据类型及输入/输出程序设计等。附录中给出读者在 VAX-11 计算机上实际操作所必须的参考资料。

本书可供从事计算机程序设计和应用的科技人员及高等院校有关专业的师生阅读参考。

Souhail El-Astouri, Olin Johnson, Willis K. King
COMPUTER ORGANIZATION AND
PROGRAMMING VAX-11
Addison-Wesley Publishing Company, 1984

计算机组织与汇编语言 程序设计(VAX-11)

[美] S. 伊尔阿斯福里 O. 约翰逊 W. K. 金 著
王祖永 译

责任编辑 孙月湘

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

1987 年 9 月第 一 版 开本: 787 × 1092 1/32
1987 年 9 月第一次印刷 印张: 12 3/8
印数: 0001—5,600 字数: 280,000

统一书号: 15034 · 852
本社书号: 4968 · 15—8

定价: 2.95 元

译 者 的 话

VAX-11 是美国数字设备公司 (简称 DEC) 继著名的 16 位 PDP-11 小型计算机系列之后推出的 32 位超级小型计算机系列。自 1978 年公布它的第一个型号 VAX-11/780 以来,已相继推出 750, 730, 782, 785, 725 等多种机型; VAX-11 已得到广泛应用。DEC 最近宣布的 8600 和 Micro VAX II 两个新的型号更为国内外用户所瞩目。8600 把 VAX-11 系列的单机性能提高了 5 倍, 而 Micro VAX II 把 780 的功能装进了微型计算机。这两项新的发展意味着 VAX-11 系列将扩展到更大的应用范围。随着四个现代化建设的进程, 我国应用的计算机越来越多, 其中 VAX-11 系列势必占有很大的比例。了解和掌握这个系列的计算机已成为大批计算机专业人员所面临的任务。译者相信, 本书的翻译出版将为他们完成这一任务助一臂之力。

本书是美国休斯敦大学为他们的学生编写的计算机基本组织和汇编语言程序设计课程的教材。作为一本教科书, 它是不能由数字设备公司出版的手册所代替的。手册通常适合在使用的现场供用户查阅, 向用户提供一系列功能描述和语法细节。但是这种手册通常不适用于初学者, 因为初学者往往希望首先掌握基本概念和方法。而本书的目的正是要让学生按认识的规律快速且系统地掌握汇编语言程序设计原理并能举一反三, 适应各种不同的机器。为了做到这一点, 原书作者选择了在美国大学中最为普遍的 VAX-11 系列作为实例。这种选择对于 VAX-11 目前的用户和潜在的用户无疑是十

分有利的。我们选择翻译这本书正是因为看中了这一点。

本书根据“Computer Organization and Programing VAX-11”一书直接译出,未作任何删节或增补。原文中有几处译者认为有错误的地方加了简短的“译注”。此外,读者将会发现,书中程序例子内的英文注释均未译为中文。这并非因译者“偷懒”或疏忽,而是有意的。首先,作为一个计算机程序员应有能力看懂程序中这类简单的英文注释,在自己编程序时也应养成写这种注释的习惯。原书的作者就十分重视程序注释的作用和正确的写法。所以,留下英文注释,作为范例供读者参考和练习阅读,无疑是有益的。另一方面,即使某些读者目前暂时不能理解这些英文注释,他们仍能通过正文中的解释弄懂程序例子,因而不会对本书的利用价值产生消极的影响。

由于工作任务的关系,译者比较关心与 VAX-11 有关的资料和书籍。比较起来,本书的编排和叙述都颇具特色。内容由浅入深,举例恰到好处。每章还附有习题,可供读者练习和测验自己的学习效果。因此它既可作为课堂讲授的教材,也可作为专业人员的自学材料。

本书的翻译出版曾得到北京信息工程学院杨天行校长的关心和支持。宋秀昆同志为本书的翻译工作提供了宝贵的意见和帮助。还有其他许多同志为此书的出版尽心尽力,付出了宝贵的劳动。在此谨向各位表示衷心感谢。

为译好本书,译者已尽了最大努力。但因水平所限,书中可能还有错误和不当之处,还望诸位读者指正。

译者

1985年8月

前 言

本书编写的目标是作为一个学期的计算机基本组织和汇编语言程序设计课程的教材。总的来说，它符合 ACM 教学大纲 78 所规定的 CS-3《计算机系统引论》课程的纲要。本书自始至终把数字设备公司 (DEC) 的 VAX-11 系统作为实例。在学习本书之前至少应学过 FORTRAN, Pascal 或 PL/1 等一种高级语言课程。

本书从让读者了解计算机所执行的基本动作开始，一步一步引导读者，直至他们能编写出水准较高的汇编语言程序。本书注重自上至下设计、模块化、结构化等良好的程序设计方法。一个读者如果没有机会在他所学的计算机上进行实际操作，他就不能很好地消化本书中的材料。为了使读者能尽早上机，在开头几章只介绍指令、寻址方式和数据类型的一个小的子集，比较复杂的寻址方式和数据类型推迟到后面几章讨论。其他的指令则在程序设计例子中用到时再进行解释。

之所以选用 VAX-11，不仅因为它的普遍性（尤其是在各大学的计算机科学系几乎都有这种计算机），而且也因为它有新型的体系结构、丰富的寻址方式以及设计得很好的指令系统。尽管如此，对于每个重要的概念，我们还总是首先从与机器无关的一般性予以介绍，然后再利用 VAX-11 作为具体实现的例子。凡在 VAX-11 实例中，本书均沿用数字设备公司的术语；当某个术语与工业上通用的术语有差别时，我们就指出来。我们相信，学过本书的读者能很快地在汇编语言一级学会使用其他计算机。

至于系统输入/输出 (I/O) 宏的用法在本书一开始是以“烹调书”的方式介绍的。我们鼓励学生利用这些宏进行输入/输出。这有助于他们摆脱对高级语言 I/O 操作的依赖。更进一步来说,这将使他们对这些操作所涉及的问题有一般的印象,为在最后一章学习 I/O 程序设计的细节作准备。在最后一章所讨论的 I/O 处理方法是使用高级语言和使用特权的 I/O 硬件指令之间的一种折衷。高级语言完全掩盖了计算机所执行的实际操作,而特权的 I/O 指令使学生能完全地控制机器。后者对于大多数计算机科学系而言是不切实际的。

书后的几个附录使本书自成一体。附录 A 透彻地讨论了数的表示方法,以及它们从一种数系至另一种数系的转换。它还包含了基本的二进制算术运算和逻辑操作。其余的附录教读者怎样使用终端、编辑程序和调整程序。它们将有助于读者尝试在终端上输入和调整其程序。当然,这些附录未能包括 VAX-11 汇编语言的每一细节。我们建议读者进一步参考数字设备公司的下列出版物:

1. VAX-11 体系结构手册;
2. VAX-11 MACRO 语言参考手册;
3. VAX-11 MACRO 用户指南。

本书的编排如下:

第一章给出汇编语言的概貌以及它在各级程序设计语言之间的位置。

第二章介绍计算机的基本组织和它的操作。这一章还讨论基本数据类型和机器指令格式。

第三章和第四章描述汇编语言的基础知识。第三章一般地介绍概念和术语;第四章描述 DEC VAX-11 MACRO 中实际采用的各种规定。

第五章是第四章的继续。这一章系统地讨论在 VAX-11

中使用的寻址方式。

第六章讨论子程序及其在 VAX-11 中的实现方法。

第七章展示二趟汇编程序的结构，并给出由 VAX-11 MACRO 汇编程序所产生的目的代码实例。这一章还讨论了链接程序和装入程序。

第八章介绍宏、条件汇编和重复块。

第九章扩展对数据类型的讨论。这一章描述了浮点数和十进制数的各种表示方法，以及由 VAX-11 系统所提供的“编辑”指令。

第十章介绍汇编语言级的输入/输出程序设计。

在美国休斯敦大学，本书作为二年级一学期十五周课程（四学分）的教材。它包括每周三小时的授课和三小时的实验。为完成这一课程，学生要编写五、六个程序。我们也认为，本书适于能接触到 VAX-11 系统的专业人员自学。

教师可根据本身的条件和教学方法重新安排本书材料的顺序。有的可在讲授了宏功能和高级数据类型（第八章和第九章）之后再讲第七章（汇编程序）；也可以把第七章提前到紧接第四章（关于 VAX-11 汇编语言）之后讲授。由于第七章是独立的，所以允许这种挪动。按我们自己的经验，有必要在学期开始时比较仔细地讲解附录 A 中的材料（数的表示和运算）。但是，有的愿意尽早进入 VAX-11 的汇编语言，甚至可跳过第二章。对于一年四学期的学校以及只关心 VAX-11 汇编语言的学校，教师可根据时间限制，第二章只选用 2.2, 2.3 和 2.5 节，第八、九章只选择其中一部分，第十章则可完全不讲授。

S. 伊尔阿斯福里 O. 约翰逊 W. K. 金

得克萨斯州休斯敦，1983 年 8 月

目 录

译者的话

前言

第一章 引论	1
1.1 机器语言和汇编语言	2
1.2 汇编语言程序	6
1.3 为什么要用汇编语言	13
1.4 操作系统	14
1.5 VAX-11 计算机系列	16
1.6 小结	20
习题	20
第二章 硬件组织	23
2.1 主存	24
2.2 信息单位	27
2.3 基本数据类型	30
2.4 中央处理机	33
2.5 指令的执行	39
2.6 指令格式	40
2.7 输入/输出系统	46
2.8 小结	47
习题	48
第三章 汇编语言基础	50
3.1 汇编语句格式	50
3.2 语句的类型	53
3.3 小结	62
习题	63

第四章	VAX-11 MACRO 语言引导	65
4.1	MACRO 语句的格式	65
4.2	简单的 MACRO 汇编命令	68
4.3	直接赋值语句	78
4.4	简单的 VAX-11 指令	80
4.5	简单的 VAX-11 MACRO 寻址	97
4.6	简单的输入/输出	102
4.7	程序例子	111
4.8	小结	115
	习题	115
第五章	寻址方式	120
5.1	基本寻址方式	120
5.2	其他寻址方式	128
5.3	小结	145
	习题	145
第六章	子程序	149
6.1	堆栈	151
6.2	子程序调用	153
6.3	过程调用	159
6.4	以地址调用和以值调用	165
6.5	递归过程	168
6.6	小结	171
	习题	171
第七章	汇编程序	174
7.1	汇编程序的功能	174
7.2	目的代码的生成	175
7.3	汇编程序的数据结构	179
7.4	VAX-11 MACRO 汇编程序	186
7.5	二趟汇编程序	196
7.6	链接和装入	201

7.7 小结	205
习题	205
第八章 宏和条件汇编	210
8.1 宏	210
8.2 VAX-11 MACRO 的宏	213
8.3 条件汇编	223
8.4 重复块	228
8.5 其他的 VAX-11 宏功能	230
8.6 小结	234
习题	236
第九章 高等数据类型	238
9.1 浮点表示和运算	239
9.2 十进制的表示和运算	250
9.3 位串和操作	258
9.4 列表和队列	261
9.5 小结	264
习题	265
第十章 输入/输出程序设计	267
10.1 输入/输出程序设计的等级	267
10.2 RMS 输入/输出宏概述	271
10.3 交互的终端输入/输出	277
10.4 行式打印机输出	282
10.5 磁盘上的顺序文件	285
10.6 相对文件和索引文件概述	289
10.7 小结	293
习题	294
附录 A 数的表示和运算	295
A.1 数	295
A.2 数的位置表示法	297
A.3 二进制和十六进制运算	306

A.4	负数的表示	308
A.5	补码运算	310
A.6	对二进制数的逻辑操作	314
习题	316
附录 B	MACRO 程序的输入、编辑和执行	320
B.1	注册 (Login)	320
B.2	程序的输入	321
B.3	现存文件的编辑	322
B.4	DCL 命令	325
B.5	程序的汇编、链接和运行	327
B.6	注销 (Logout)	329
附录 C	VAX-11 指令集	330
C.1	操作数说明符表示图例	330
C.2	条件码图例	331
附录 D	ASCII 字符集	342
附录 E	VAX-11 符号调整程序	344
附录 F	FORTRAN 和 Pascal 与 MACRO 的链接	351
F.1	FORTRAN 与 MACRO 的链接	351
F.2	Pascal 与 MACRO 的链接	354
术语汇编	358
参考文献	374
汉英对照索引	376

第一章 引 论

第一台电子计算机建造于本世纪四十年代中叶。从此之后，在不到四十年内我们已经亲眼看到了这种奇妙机器的史无前例的发展。今天，计算机在我们社会中已是无所不在。许多人可能在使用计算机而并未意识到这一点。当一个人看一下电子手表，用微波炉烹调菜肴或在银行的自动出纳机前使用银行卡片时；他（或她）正在使用计算机。另一些人认识到计算机可用于解决他们所特有的问题。他们学会了 BASIC, FORTRAN, COBOL 或 Pascal 等程序设计语言，使他们能与计算机通信。还有一些人则希望更深入地了解计算机，知道它如何工作。本书的对象正是这一批人。它将使读者初步了解计算机的内部工作。

为了深入了解计算机及其工作过程，最起码的要求是很好掌握机器语言。本章就是朝这一方向迈出的一步。1.1 节介绍机器语言，然后介绍与机器语言密切相关的汇编语言。我们将会看到，用符号的汇编语言远比用数码的机器语言容易使用。但是就研究计算机的内部工作而言，了解汇编语言和了解机器语言所起的作用相差不多。

由于各个计算机制造厂商采用不同的设计方案，也由于预期的应用不同，因此每种计算机有它自己的机器语言，相应地有它自己的汇编语言。换句话说，有多少种计算机就有多少种汇编语言。因而，汇编语言程序是针对特定的一种机器编写的（与之相反，高级语言程序通常不加修改或稍加修改就能在支持这种语言的任何一种机器上运行）。因此用汇编语

言编制程序必须熟悉为之编程的那种计算机。在本书中，我们以 VAX-11 计算机系统及其汇编语言作为例子进行讲解，同时尽量指出各种汇编语言之间的共同特点。在 1.2 节介绍了一种类似于 VAX 的汇编语言，用它表示一个汇编语言程序的例子。

汇编语言不仅仅是用于了解计算机，在 1.3 节详细说明了应学会汇编语言的理由。为了充分利用一个计算机系统，该计算机的硬件必须要有专门的软件加以补充。1.4 节简要地介绍这些软件。本章最后一节介绍 VAX-11 计算机系列的概貌。

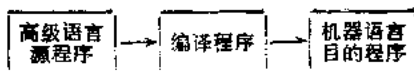
1.1 机器语言和汇编语言

这一节的主题是要回答这样一个问题：“什么是机器语言和汇编语言？”我们先来简要地弄清一个高级语言程序在其可由计算机硬件执行之前要经过哪些加工。本书假定读者不同程度地熟悉诸如 FORTRAN, Pascal 或 PL/1 等一种高级语言。

程序翻译

一般计算机不能直接执行类似于英语的高级语言语句。这类语言必须先翻译成计算机所理解的语言——**机器语言**。

通常把用一种高级语言编写的程序称做**源程序**；把被翻译成的机器语言程序称为**目的程序**。高级语言 (HLL) 程序的翻译由一个叫做**编译程序**的程序执行。编译程序读取高级语言的源程序生成机器语言的目的程序。因此，源程序是编译程序的输入，目的程序是编译程序的输出。这种关系用图描述如下：



注意,在翻译期间,计算机硬件执行的程序是编译程序而不是用户的源程序。只有在目的程序完全生成之后,用户的程序才处于可以执行的新形式。

机器语言

现在我们来看一看机器语言的性质,也就是研究一下目的程序。尽管如前所述,不同的计算机有不同的机器语言,但它们还是有某些共同的特点。下述特点是一般计算机都具有的。

计算机硬件只能执行基本指令,诸如加、减、比较、转移等等。因此,目的程序由基本指令组成,这些基本指令与源程序中的语句相对应。计算机利用一个称为**操作码**的代码识别每一条指令。这个代码常以数码的形式表示。例如,65在VAX中是表示**加**的操作码,81是表示**比较**的操作码。

某些指令需要**操作数(数据)**。这些操作数是指令操作的对象,它们必须与操作一起规定,以便指令执行。在计算机系统中,操作数存放在存贮单元内,每个单元利用一个称为**地址**的唯一编号识别。当一个需要操作数的操作提交硬件执行时,既要规定找到操作数的单元的地址,又要规定存放操作结果的单元的地址。

一条指令及其操作数说明符的格式可能是下面的样子:

65 20, 30, 40

它的意思是“加 20, 30, 40”。它命令计算机硬件把在存贮单元 20 和存贮单元 30 中的操作数相加(即把这两个单元的内容相加)并把结果存放在存贮单元 40 之中。图 1.1 表示出上面这条指令执行前和执行后有关存贮单元的情况。图中在存

单元 40 中的问号(?)表示这一单元原来的内容没有意义,因为它将被指令执行的结果冲掉。

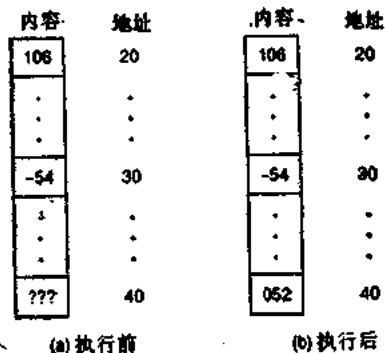


图 1.1 执行指令“加 20, 30, 40”前后存贮器的内容

一般计算机的另一个特点是它们由任何时刻只能处于两种状态之一的电子器件组成,这两种状态分别用 0 和 1 表示。因此,计算机硬件只能处理用二进制代码表示的信息,提交硬件处理的所有信息都由一串 0 和 1 组成。请注意,在图 1.1 中的数是用十进制表示的;但在计算机内部,内存单元的内容以及指令的操作码和指令中的地址都是用两个状态 0 和 1 表示的。例如,指令

65 20, 30, 40

在计算机内部可能表示为

1000001 010100 011110 101000

总而言之,目的(机器语言)程序是基本指令的序列,而指令和操作数是用二进制编码的。

汇编语言

在计算机的早期,程序员用机器语言编写程序,也就是他们用手工提供程序的二进制编码。不用说,用机器语言编制

程序非常麻烦而且容易出错。由于有这些困难，后来程序员用符号描述操作和规定操作数。例如他们用易于记忆的符号 A 和 S 代表加 (add) 和减 (subtract) 指令，以此代替这些指令的操作码。同样，他们利用符号名代替数值地址来引用存储器单元。例如，他们可能以

A X, Y, Z

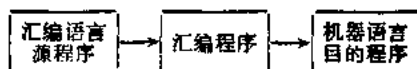
代替

1000001 010100 011110 101000

其中 X, Y, Z 分别表示存储器单元 010100, 011110, 101000。

当机器指令用符号名表示时，它们被称为**汇编指令**，**汇编语言**仅仅是符号形式的机器语言。

与高级语言程序一样，汇编语言程序必须翻译为机器语言后才能执行。一个称为**汇编程序**的程序执行这种翻译。与编译程序相似，汇编程序象读数据一样读取源程序，并产生与之等效的机器语言目的程序。这种关系用图描述如下：



一个汇编语言程序正被翻译为目的程序的时候被称为**汇编时**，实际翻译的过程被称为**汇编**。用户的程序(目的形式)被执行的时候称为**执行时**或**运行时**。在此期间，该程序(指令和数据)驻留在计算机的主存中。

与在高级语言中一样，一个汇编语言程序由一系列可执行语句及说明语句组成，一个可执行语句请求在执行时完成一个操作。所不同的是，在高级语言中的一个可执行语句可能要翻译为几条机器指令，而在汇编语言中的一个可执行语句仅被翻译为一条机器指令。

大多数汇编语言与某些高级语言(例如 Pascal)一样，需