● 高等学校试用教材

● 高等学校试用教材

● 西南交通大学 朱怀芳 [illegible] 合编

VAX—11计算机
汇编语言程序设计

高等学校试用教材

VAX-11 计算机

汇编语言程序设计

西南交通大学 朱怀芳 陈志 合编

北方交通大学 安维蓉 主审

中国铁道出版社

1989年·北京

内 容 简 介

本书介绍VAX-11计算机汇编语言程序设计的方法和技巧。全书共分十三章。在介绍VAX-11计算机体系结构、指令系统和寻址方式、信息表示的基础上，论述了分支、循环、子程序、输入／输出等程序设计方法，汇编语言程序设计技巧，汇编语言对数据结构的操作、高级汇编语言技术和高级程序设计技术。最后一章还介绍了汇编语言程序的排错和调试方法。

本书为有助于读者学习，书中程序范例丰富，每章后附有习题，书后附录给出较多有用的信息。这是一本适合于高等院校计算机科学与工程类专业的教材，也可供有关工程技术人员参考阅读。如果根据不同教学要求选取书中有关章节进行教学，本书也能适应不同层次教学要求。

高等学校试用教材

VAX 11计算机

汇编语言程序设计

西南交通大学　朱怀芳、陈　志　合编

中国铁道出版社出版、发行

责任编辑　倪嘉寨　封面设计　王镇平

中国铁道出版社印刷厂印

开本：787×1092毫米 1/16　印张：14.25 字数：351千

1989年11月　第1版　第1次印刷

印数：1—2,000册　定价：2.85 元

ISBN7-113-00563-2/TP57

前　言

程序设计已成为一门学科，它研究的课题是如何正确地、科学地、经济地为计算机准备一个有序的指令集合。随着程序设计学科的发展，编写程序所用的语言从简单到复杂，从低级到高级。所谓低级语言是指机器语言和汇编语言，它们是与具体所用的计算机指令系统密切相关的。所谓高级语言是指FORTRAN、BASIC、PASCAL、PL/1等程序语言，它们的用法与具体的计算机指令系统无关。

本书论述的问题是用汇编语言编制程序的概念、方法和技巧。鉴于汇编语言是与具体计算机型号有关的，因此，论述时不得不选定某个计算机为对象。本书选定美国数字设备公司（DEC）的VAX-11系列机作为数学机。该计算机系列功能很强，其指令系统具有典型性，我国引进较多，备受用户喜爱。选择这种计算机来研究汇编语言程序设计的方法和技巧，它也适于程序设计的共性，对其它计算机汇编语言程序设计有同样的指导意义。

汇编语言程序设计是计算机科学与工程类专业大学生必须掌握的知识和技能，是大学本科教育的主干课，是进一步学习编译原理、操作系统等专业课的基础。参照“全国计算机教育与培训学会”推荐的教学计划与教学大纲要求，汇编语言程序设计课程安排在第二学年进行，总学时为68小时。根据本书的内容和编者的教学经验，建议讲授48小时，编程练习和上机实习20小时。如果根据计算机专业的专科教学计划选取本书中的有关章节进行教学，本书也是一本能适应不同层次教学的教材。

全书共十三章，分为五个部分。

第一部分（第一章至第四章）是有关VAX-11计算机体系结构、信息表示、指令系统和寻址方式，以及用汇编语言进行程序设计的基本过程。这一部分是学习以后各章的基础。

第二部分（第五章至第八章和第十一章）讨论循环、子程序、对数据结构的操作、输入/输出等程序设计的方法和技巧。由于这部分内容是由简到繁、循序渐进写的，读者要逐章读下去才能深入理解。其中第十一章输入/输出程序设计是属于基本的程序设计范畴，但它要用到高级汇编语言技术，所以安排在靠后的章节。

第三部分（第九章和第十二章）是讨论高级汇编语言技术和高级程序设计技术。这部分内容是第二部分所讨论的方法和技巧的扩展，这些“高级”技术的采用，使得程序设计变得更方便、更灵活，并且能做到各种高级语言和汇编语言之间可以互相调用。

第四部分（第十章）是介绍汇编语言程序如何被翻译成机器目标码的过程，以及程序链接和装入的概念。这部分内容是相对独立的。

第五部分（第十三章）是实践性很强的内容，它将帮助读者在计算机上编辑、排错、调试他们的程序。这部分内容虽然安排在最后一章，但只要掌握了基本的汇编语言程序设计方法，并准备上机实习时，就应该学习这部分内容。

本书是编者多年从事该课程教学工作的总结，又在《PDP-11计算机汇编语言程序设计》一书（朱怀芳编，中国铁道出版社1985年出版）基础上大幅度增补、改写而成。第一、三、四、五、八、九、十章由朱怀芳编写，第二、六、七、十一、十二、十三章由陈志编

写。

在编写本书过程中，主审、北方交通大学安维蓉副教授对全书作了详尽的审阅，并提出许多宝贵的意见。铁路高等学校计算机，自动控制、工业电气自动化专业教学指导委员会主任、西南交通大学靳蕃教授，西南交通大学杜申华副教授和范平志老师仔细地审阅了本书并提出了很好的意见。对此，表示衷心感谢。由于编者水平所限，不妥之处，在所难免，请读者批评指正。

朱怀芳　　陈　志

一九八八年七月于四川成都

目　录

第一章　计算机体系结构综述

当今，计算机的应用已深入到社会的各个角落，许多人已在不知不觉中使用计算机，也许他对计算机的结构和工作原理了解很少，但使用自如，这并不奇怪，这正是现代计算机发展的方向：让计算机的应用更为方便和普及。

对于采用计算机高级语言（例如BASIC，FORTRAN，Pascal）编写程序来解决一些复杂问题的人们来说，计算机表现为一台执行高级语言语句及处理高级数据结构的机器，他们用不着了解其内部的硬件结构。对于用汇编语言来进行程序设计的人们来说，简略地了解计算机系统结构、程序执行过程是必要的。

本章介绍汇编语言程序设计者所需要的有关计算机体系结构的基本知识。由于确定以VAX-11计算机为教学机，故首先介绍VAX-11计算机的体系结构及其工作原理，然后介绍几个与编制汇编语言程序有密切关系的部件，如存贮器、通用寄存器和处理机状态长字等。

第一节　VAX-11的硬件组成及工作原理

VAX-11是美国数字设备公司（DEC）开发的一种通用数字计算机，该机器在设计时扩充了PDP-11的寻址能力。它的名字VAX是Virtual Address eXtension的缩写，意思为虚拟地址扩展。尽管VAX-11有较多的指令与PDP-11的指令相类似，但VAX-11是一种崭新的系统结构，其硬件和软件都是比较先进的。为了集中叙述程序设计的基本方法和技巧，在VAX-11系列机型中，本书将以VAX-11/750计算机为主要对象进行讨论。

一、硬件组成

大多数通用计算机都具有相同的基本结构，亦即由高速主存贮器、运控部件（即中央处理机或CPU）和外围输入/输出设备组成。通过一条或多条总线在这些部件之间传送数据，如图1—1所示

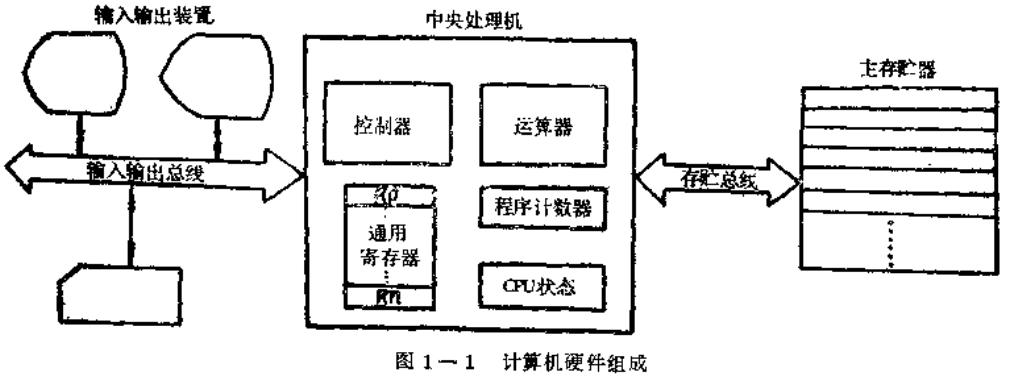

图1—1　计算机硬件组成

与多数通用计算机相似，VAX-11计算机最简单的硬件组成框图如图1—2。

图1—2中，CPU是VAX-11计算机系统的主控部件。处理机拥有十六个32位通用寄

存器，向程序提供高速局部存贮场所。这些通用寄存器的各种功用将在第四节中介绍。CPU的内部还有一个32位的处理机状态长字（PSL），它存放处理机信息和正在执行程序的当前状态。PSL的作用在第五节介绍。CPU中的算术逻辑运算部件，字长32位，是对信息进行加工的主要地方。

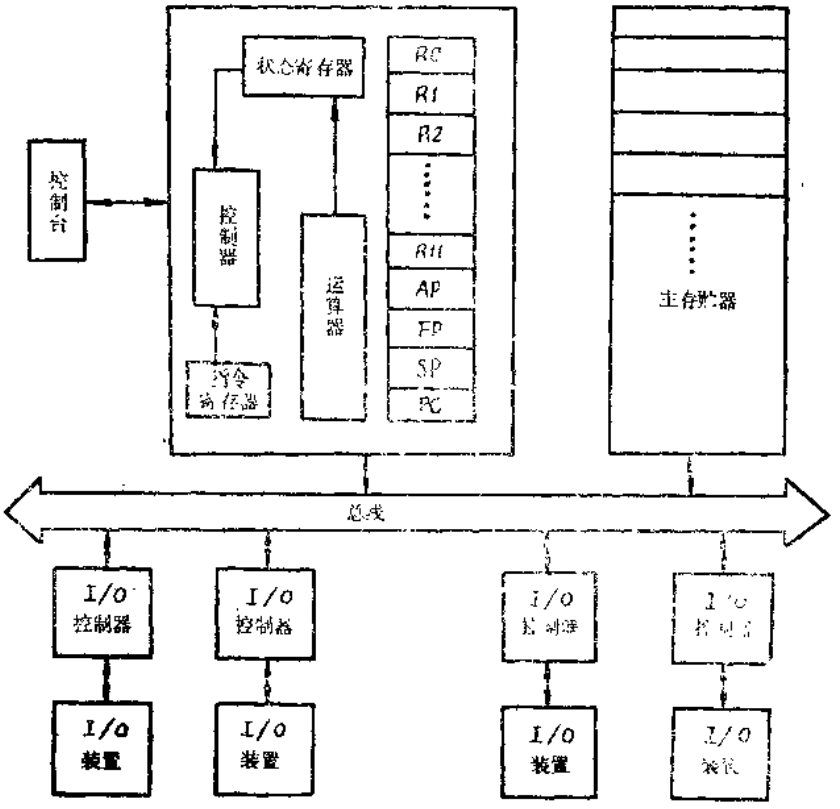

图 1—2　VAX 11硬件组成

主存贮器是VAX-11计算机用来存放信息（包括指令、数据和系统用的程序）的地方。大多数的信息是按 8 位字节为基本单位存放在存贮器中。但VAX计算机允许有32位虚拟地址空间，即可看作是一个下标从 0 到 $2^{32}-1$ 个单元的字节阵列。关于存贮器更进一步的介绍在第三节进行。

与处理机和存贮器相连的是若干台输入和输出设备（简称I/O设备）。最普通的设备是磁带、磁盘、显示终端和行式打印机。磁带和磁盘为程序和数据提供大量的联机或后备存贮，这些设备通常称为辅助存贮设备，它们能直接对主存贮器传输大量数据块、信息块（例如系统用的程序模块），而处理机对大多数终端设备每次只能送出或取回一个字符。I/O 设备使处理机能够与人、其它处理机以及辅助存贮设备进行通信。

连接CPU、存贮器和I/O设备的是总线。通过这条总线，所有I/O设备被连接起来，它们可以相互通信或者与CPU和存贮器通信。而且，I/O设备与存贮器之间交换数据，无需处理机干预。因为 VAX-11 计算机具有完整的单总线结构，所以对I/O设备的调用可以用对主存贮器的调用指令一样来操作。

应该指出，图 1—2 所示的VAX-11硬件组成是最基本的配置，一般用户使用的计算机可以比这种配置稍“丰富”些，附加的部件用于提高计算机的能力和速度，例如用来提高某

些指令速度的浮点加速器。附加的部件作为硬件实现的一部分，它们对在该计算机执行我们下面所举的程序例子来说都是“透明”的，不影响我们的程序设计，因此，在这里不再作进一步的介绍了。

二、工作原理

用很少的文字来说明计算机的工作原理是困难的，但从计算机操纵的基本概念（寄存器、指令系统，寻址方式等）和学习汇编语言程序设计技术角度来说明是可能的。存贮器是存贮指令和数据的地方；中央处理机是计算机的大脑，可以从存贮器中取出、存放指令和数据；计算机解决问题是通过程序来控制，程序是由指令序列组成；中央处理机通过解释指令来执行一个程序；与处理机和存贮器连接的是若干个输入和输出设备，它们使处理机能够与人、其它处理机以及辅助设备进行通信。

在使用计算机的最初年代里，计算机的语言就是机器的指令系统，我们称它为机器语言，程序是手工编制的。所谓手编程序就是用机器语言编写的能直接在机器上执行的程序。但是，用机器语言编写程序很繁琐、易出错，程序检查和调试都比较困难，这极大地影响了计算机的信息处理周期和使用效率。为此，人们改革机器语言，采用符号拼写的说明符来代替机器指令中的二进制数写的操作码、操作数地址码、指令地址码和变址量地址码等。这就是所谓汇编语言。

随着程序设计学科的发展，编写程序所用的语言从简单到复杂，从低级到高级。所谓低级语言是指机器语言和汇编语言，它们是与具体所用的计算机指令系统和体系结构密切相关的。所谓高级语言是指BASIC、FORTRAN、Pascal等程序语言，它们的用法与具体的计算机指令系统和体系结构无关。但是，无论是高级语言还是汇编语言，用它们编写的并放在计算机中准备执行的程序，都要经过一种翻译程序（系统软件之一）将程序翻译成该计算机能够“认识”的机器指令代码。对于高级语言的翻译程序通常称为编译程序，对于汇编语言的翻译程序通常称为汇编程序。

第二节　汇编语言程序执行过程

有了上节所给出的计算机硬件的知识和有关汇编语言、汇编程序的初步概念，本节进一步给出一个汇编语言编写的源程序的例子并说明其在计算机中的工作过程，以使我们对计算机如何工作有进一步的了解。

用汇编语言编写的源程序在计算机中工作的过程归纳为以下几步：

一、根据汇编语言编写程序时的硬性规定（如语句格式、语义、语法等）编写出源程序，通过输入装置送入主存贮器中。

表1—1给出一个完整的用汇编语言编写的程序例子，其中许多部分目前我们尚不明白，但它给出一个整体程序的概念，以利于我们今后的学习。表1—1的程序清单中，分号（；）后面是程序的注释部分，它不是程序中的指令，仅起到帮助了解程序功能的作用，是可有可无的，一般用英文写出，为清楚起见，这里用中文写出。

二、通过预先放在内存中的汇编程序将汇编语言源程序翻译为可执行的目标程序。如图1—3所示。

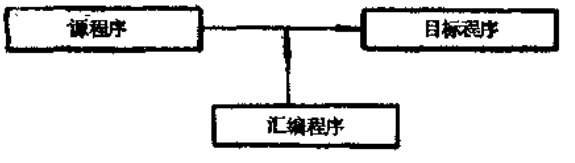

图1—3　源程序被汇编为目标程序

表 1 - 1

标　号	汇编语言	注　释
	CONSTANT = 4	; 相乘的常数 = 4
LIST:	.BYTE 1,2,3,4,5	; 将要被处理的 5 个元素
SIZE:	.BYTE SIZE - LIST	; 这个单元放的数是SIZE - LIST
	;	
	; 下面是可执行的编码	
	;	
	.ENTRY SCALE - ARRAY, 0	; 从这里开始执行
	CLRB R5	; R5是循环计数器，先清 0
	MOVAB LIST, R6	; 指向标号LIST下的第一元素
LOOP:	MULB #CONSTANT, (R6)	; 常数 4 乘第一个元素
	INCL R6	; 指向标号LIST下的第二个元素
	INCB R5	; 循环计数器加 1
	CMPB SIZE, R5	; 所有的元素处理完了吗?
	BGTR LOOP	; 未处理完，继续处理
	$EXIT_S	; 处理完了，程序终结
	.END SCALE_ARRAY	; 这里是程序的末尾

目标程序的每条指令是用二进制码表示，且与汇编语言源程序的每条指令一一对应。表 1 — 2 列出上例中每条指令对应十六进制的目标编码。

表 1 — 2

汇编语言源程序			机器语言目标程序		
标　号	操作码说明符	操作数说明符	位置计数器值	操作码	操作数
	CONSTANT = 4		0000	00000004	
LIST:	.BYTE	1,2,3,4,5	0000	01 02 03 04 05	
SIZE:	.BYTE	SIZE - LIST	0005	05	
	.ENTRY	SCALE - ARRAY, 0	0006	0000	
	CLRB	R5	0008	94	55
	MOVAB	LIST, R6	000A	9E	AF F3 56
	MULB	#CONSTANT, (R6)	000E	84	04 66
	INCL	R6	0011	D6	56
	INCB	R5	0013	96	55
	CMPB	SIZE, R5	0015	91	AF ED 55
	BGTR	LOOP	0019	14	F3
	$EXIT_S		001B		
	.END	SCALE_ARRAY	0024		

三、在控制器和系统软件的帮助下，分析目标程序的每条指令，根据指令中操作码的要求到存贮器存取有关数据或信息，由运算逻辑部件处理有关数据或信息，直至程序中最后一条指令被处理完为止。

现代计算机系统除了配置硬件系统外，还配置有系统软件，它的作用是帮助人们管理计算机的硬件，并使计算机硬件系统发挥出最大的效率。系统软件是由许多可多次使用的为用户所共享的程序组成，重要的有操作系统（其核心是监控程序）、控制程序、编辑程序（协助编辑和修改输入的源程序）、汇编程序、编译程序……等，在汇编语言程序输入，修改、汇编、执行等过程中要用到部分系统软件。硬件、系统软件和用户程序关系图如图 1 — 4。

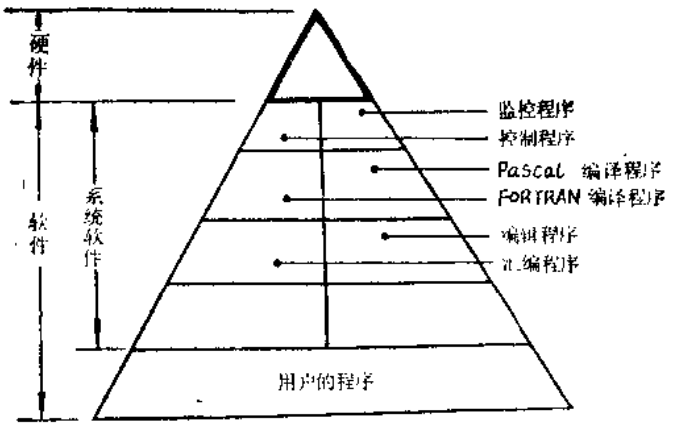

图 1－4　硬件和软件关系图

第三节　VAX-11计算机的虚拟存贮技术

VAX计算机的体系结构是用来支持多道程序工作的，即在单机系统中，同时可执行若干个进程。（一个进程是指一个程序连同它的数据在处理机上的动态执行过程。也可以说是一串按序执行的单机指令流）。VAX计算机中的进程存贮工作在2^{32}字节的存贮空间里，这意味着计算机能有效地识别四十亿个地址。

VAX计算机采用“虚拟存贮技术”，即通过计算机系统软件的存贮管理功能，允许程序寻址比实际可以利用的存贮器有更大的存贮空间。通过利用系统中的磁盘将当前没有使用的一些程序块存贮起来，提供“虚”的存贮器，当访问到虚拟存贮器中某一个程序块时，由操作系统再将该程序块装入主存，而将某些原驻留主存的程序块依次送回磁盘。主存贮器可以为多个进程共享，每个进程在其操作时好象已被装入到连续的实际存贮器中。这样就提高了处理机和存贮器的利用率。

由于实际存贮器的管理是通过操作系统实现的，其管理的方法对进程来说是“透明”的，若干进程可以同时占有主存，完全自由地使用进程空间地址，并独立地引用它自己的程序和数据。

虽然存贮器的地址对进程是“透明”的，程序员无需了解程序和数据占用存贮器的哪一部分，但对虚拟地址空间的分配有所了解还是有益的。进程场景中的地址空间分配如图 1－5。

把虚地址空间分为两半，一半称为系统空间，系统空间装操作系统等软件和具有系统级的数据。系统空间由所有的进程共享，它对系统中所有的进程都是相同的。另一半虚地址空间分别分配给每个进程，因而被称为进程空间。进程空间被进一步细分成P_0和P_1空间，在P_0空间中存放程序及其大部分数据；在P_1空间中是堆栈（一个特定的存贮区，它的一端是固定的，另一端是浮动的，对这个特定存贮区，所有的信息存入和取出都只能在浮动的一端进行，并且符合后进先出原则）和进程专用数据。因为P_1空间用于堆栈，故是存贮器中唯一从高地址向下分配的空间。每个进程都有它自己的P_0和P_1空间，图 1－5 说明了在多道程序系统中几个进程的地址空间，每个进程空间与其它的进程空间无关，而系统空间则由所有的进程共享。

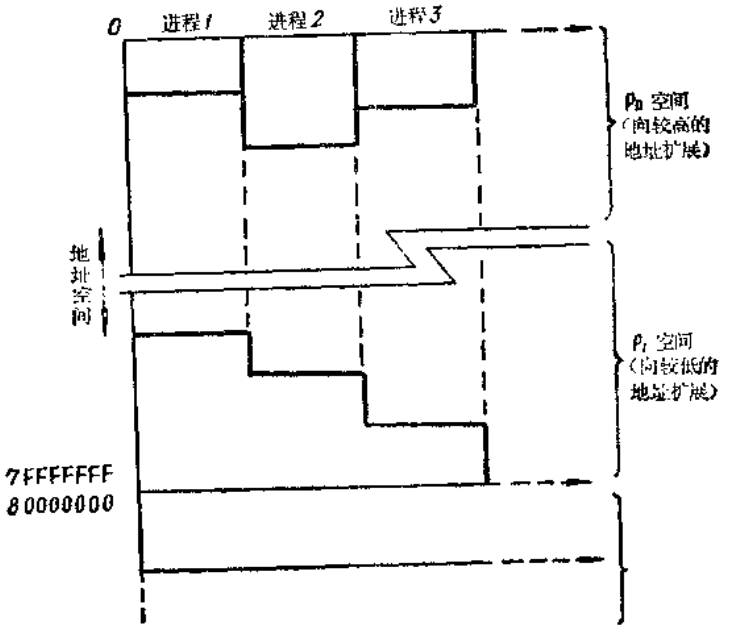

图 1—5　进程场景中的地址空间

第四节　VAX-11计算机的通用寄存器

VAX-11计算机的处理机内部有十六个32位的通用寄存器（见图1—2）。这些寄存器（通常表示为R0,R1,R2, …R10, …R15）可供本机指令集使用。它们之所以被称为“通用”，是因为每一个寄存器都可以用来作累加器（所要处理的数据存于寄存器内）、指示器（寄存器中的内容为操作数地址，而不是操作数本身）、自增寄存器（寄存器中的内容为操作数地址，其地址被使用后自动增加）、自减寄存器（寄存器中的内容为操作数的地址，首先被自动减少然后才被作为地址使用）、变址寄存器（寄存器中放有一个变址值，把它加到基本地址上，其结果作为操作数的地址）。

十六个通用寄存器中的R12、R13、R14、R15具有独特的功能，应该特别注意它们的用途。

R15，用作中央处理机的程序计数器（简称PC）。它存放着下一条将被执行指令的地址。每当从存贮器取一条指令后，PC自动地按指令字节数递增。为了保证程序的正常执行，它是一个仅用于寻址的通用寄存器，不能用作算术操作的累加器。

R14，用作硬件堆栈指示器（简称SP）。在VAX-11计算机中，任一寄存器均可在程序控制下作为SP。当数据或信息送入堆栈时，SP中内容递减，即地址递减，这可简便地递减地址及向堆栈中“压入”数据信息。另一方面，若从堆栈中读出数据或信息，SP中内容递增，即地址递增，这就可简便地递增地址并从堆栈中“弹出”数据或信息。因此，SP总是指向堆栈的最后入口。但是，应特别指出的是：一个程序执行过程中，与子程序联结、中断服务、陷阱（若计算机内部根据一条特殊的机器指令，模拟某一意外事件，而自动转移到一个特定的程序去处理它。这种转移叫做陷阱）、错误状态等能引起一个指令顺序改变为另一个指令顺序，而VAX-11计算机中“自动地”使用R14中保留一个返回到前一个指令序列的返回地

址，这就是为什么称R14为“硬件堆栈指示器”的缘由。

R13，用作帧指针（简称FP）。VAX-11过程调用的约定构成了在堆栈上建立一个称为堆栈帧的数据结构。CALL指令把堆栈帧的基地址装入FP，而RET（返回）指令按照装有堆栈帧地址的FP实现返回。而且，VAX-11的软件依靠FP的支持正确报告某些例外条件。

R12，用作变元指针（简称AP）。VAX-11过程调用约定使用一个称为变元表的数据结构，并用AP作为变元表的基地址。CALL指令依照约定装入AP，在用AP作其它用途时，不存在软件和硬件上的约束。

R11～R6，对硬件或操作系统均无特别含义。具体的程序可以对每个寄存器指定具体的用途。

R5～R0，通常可由程序作任何使用，但也可以由执行中必须被中断的指令装入特定值，象字符串和十进制运算、CRC和POLY指令。具体指令的说明将指出每个寄存器的使用和应向它们装入什么值。

由上可见，VAX-11计算机的通用寄存器的使用分配的基本点是：序号高的寄存器（如R15、R14、R13和R12等）具有更为全局性的意义，而序号低的寄存器则用来作为暂时的、局部的用途。虽然这条规则并没有技术上的根据，但这是硬件和软件所应遵循的约定问题。

第五节　VAX-11计算机的处理机状态长字

在VAX-11计算机中有一个非常重要的寄存器：处理机状态长字（简称PSL）。如图1—6。

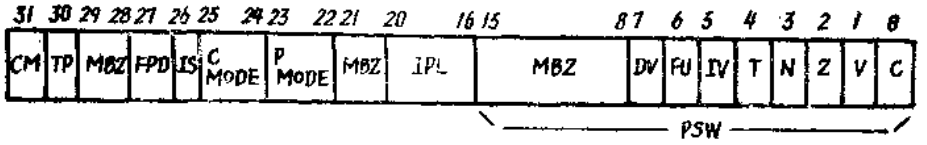

图 1 - 6　处理机状态长字

PSL是VAX计算机将若干个与每个进程相联系的处理机状态变量集中在一起组成32位的信息，这些信息由“1”或“0”来表示，代表被程序控制或者控制程序的不同含义。

一、PSW各位的说明

PSL的15～0位，单独被称为处理机状态字（简称PSW）。PSW包含非特权信息（特权是指具有某种特权才能使用的一组机器命令，使用此特权命令将产生一些特权信息，一般用户无权直接使用这些命令和信息），而且那些具有确定含义的PSW位可被任何程序自由地控制，它们在编写汇编语言程序时要遇到的。

PSW的3～0位称为条件码位；通常它们反映上一条指令操作结果的信息。条件码由条件转移指令进行测试。

3位——N，“负”的条件码位；通常它由上次指令操作结果为“负”时，置“1”，反之，置“0”。N位反映的是实际结果，即使结果的符号在代数上是不正确的也是如此，例如溢出时所造成的结果。

2位——Z，“零”的条件码位；通常由上次指令操作结果为“0”时，置“1”，反之，置“0”。同样，它反映的是实际结果，因而即使有溢出发生，也是如此。

1位——V，“溢出”的条件码位；如上次指令在执行过程出现算术溢出时，置“1”，反之，置“0”。对那些不可能产生溢出或者溢出没有意义的指令，对V置“0”与否均无影响。注意，如果相应的“自陷使能位”（即PSW的7～4位）陷”时，都要把处理机状态长字（PSL）和程序计数器（PC）的内容压入堆栈，并调用系统服务子程序进行处理。

7位——DV，是十进制上溢自陷使能位；当置“1”时，在执行任一指令之后，若所产生的十进制结果的绝对值太大，以致所提供的终点空间不能表示出来时，则导致十进制上溢自陷。当DV置“0”时，则不产生十进制上溢自陷。所存的结果，由低位数和代数上正确结果的符号组成。注意，对于其它自陷条件，如用零除和浮点上溢，没有使能标志。

6位——FU，是浮点下溢自陷使能位；当置“1”时，在执行某指令之后，其结果小于所能表达的数值时，则引起浮点下溢自陷。当FU置“0”时，则不产生浮点下溢自陷。当发生浮点被置“1”，对V位置“1”的溢出条件码，也可能导致自陷。

0位——C，“进位”的条件码位；通常，在算术操作之后，若最高有效位产生了进位或者借位于最高有效位时，则置“1”。在算术操作之后，没有进位或借位，则置“0”。其它指令执行后，C位不是被置“0”，就是没有影响。C位的独特之处是，它不仅确定条件转移的操作，而且还作为用来实现多精度算术运算的ADWC（加进位）和SBWC（减进位）指令的输入变量。

PSW的7～4位是自陷使能标志位，在一定条件下，可引起“自陷”。“自陷”是在指令结束时出现的例外状态，由本指令引起的例外事件。例外事件由操作系统来管理，当引起“自下溢”时，所得结果为0。

5位——IV，是整数溢出自陷使能位；当置“1”时，在执行一条指令之后，所产生的整数结果，不能在所提供的空间内正确地表达时，则引起整数溢出自陷。当置“0”时，则不产生整数溢出自陷，条件码V的置“1”与IV的状态无关。

4位——T，是跟踪位；当其置“1”时，在执行下条指令之后，导致跟踪自陷的产生。此条件可用于调试程序，每一次可按步通过程序的一条指令来完成软件的分析。

PSW的15～8位没有使用，留作备用。

二、PSL的31～16位的作用

20～16——IPL，表示处理机的“中断”优先级。中断请求可来自设备、控制器或处理机本身。为了让处理机确认一个中断，其优先级必须高于现行的IPL。实际上，软件均运行在IPL0级上，因而处理机对任何优先级的中断请求都可确认并为之服务。任何请求的中断服务程序也都运行在该请求的IPL级上，从而暂时把低于或等于的优先级中断请求屏蔽。由20～16位（共5位）能组成从16进制的01到1F共31个优先级，其中中断级$(01)_{16}$到$(0F)_{16}$完全由软件使用。虽然目前的VAX-11/750计算机系统仅支持$(14)_{16}$到$(17)_{16}$级，但$(10)_{16}$到$(17)_{16}$级由外围设备及其控制器使用。$(18)_{16}$到$(1F)_{16}$级用于紧急状态，其中包括间隔时钟、严重的错误和电源故障的中断请求。

21位必须是0。

23～22位——原先方式位，它包含来自最近一次特权较小例外中的现行方式字段的值。原先方式是有意义的，例如在PROBE指令（核实变元的可访问性的特权指令）中，它使特权程序能够决定在原先态的调用者是否有足够的特权来访问存贮器所给定的区域。

25～24位——现行方式位，它决定现在正执行程序的特权级。这个方式字段用两种方式批准其特权：一是某些指令（如停机、送入处理机寄存器、从处理机寄存器送出）是否可执行的，除非现行方式是核心态；二是存贮管理逻辑，根据程序的现行方式、访问类型（读或写）以及对地址空间的每个页面指定保护码，来控制对虚地址的访问。

26位——IS，中断堆栈标志，它表明处理机是用专门的“中断堆栈”而不是与现行方式有关的四个堆栈之一。当IS置“1”时，现行方式总是核心态；因此，在“中断堆栈”工作的软件拥有全部核心态的特权。

27位——FPD，是第二部分完成的标志，在某些指令中，处理机要使用这一标志，这些指令是在执行中间可以被中断或产生页故障的指令。如果FPD置“1”，当处理机从例外或中断返回时，将重新从断点开始，而不是重新启动该指令。

PSL的29～28位没有使用，留作备用。

30位——TP，是跟踪申请位。处理机用它来告诉：在跟踪位T（第4位）置“1”时，所执行的每一条指令产生一次而且仅仅产生一次跟踪自陷。

31位——CM，是兼容方式位。当CM置“1”时，处理机为PDP-11兼容方式并执行PDP-11的指令。当CM置“0”时，处理机为本机方式并执行VAX的指令。

三、PSL的使用

VAX-11计算机中PSL各位的具体含义已经说明，由此可以看到在使用时是很复杂的，有些作用一时还难以明了，这里我们仅仅给出一些概念，以利于对下面程序设计的有关问题的理解。应该指出：PSL的31～16位具有特权状态，往往由系统程序工程师来驾驶它们，虽然任何程序都可以执行一条REI指令（加载PSL），但REI将拒绝加载任何可能会增加进程特权或者在处理机中建立不确定状态的PSL。但是，PSL的15～0位包含非特权信息，它们可被任何程序自由地控制，对这些位的含义和作用我们应该逐步加深理解。

习　题

1.1　VAX三个字母代表什么？其涵义是什么？

1.2　指出计算机硬件的主要组成部分，简述每一部分的功能。

1.3　什么叫机器语言？什么叫汇编语言？它们之间有何关系？

1.4　什么叫地址空间？如果一台机器具有4位地址，那么，它有多少可寻址存贮单元？它们如何编号？

1.5　R15、R14又称为什么寄存器？其功能是什么？

1.6　简述汇编语言程序的执行过程，指令在存贮器中是否按顺序存放和执行？为什么？