VAX计算机系统的研究与应用

（上）

VAX-11计算机系统体系结构与硬件设计

（第一分册）

梁吟藻 编

序

此分册是："VAX-11计算机系统的研究与应用"一书的基础部分，编者想在这里通过介绍VAX的体系结构和硬件设计使读者对VAX计算机系统有一个基本的了解，并在此基础上能更好地阅读和掌握"VAX-11计算机系统的研究与应用"的其它部分，并能结合实际，推广应用。全文共分九章，并有四个附录。

第一、二章着重介绍VAX的主要性能和系统概貌。第三、四章详细描述VAX的数据类型和指令格式。第五至第七章重点讨论机器的内部结构、存储管理以及异常和中断。第八章讨论本机方式下的指令系统，并根据其功能和使用情况描述各类指令的功能。第九章介绍在VAX体系设计中如何使兼容方式下工作的VAX"看起来好像"是在PDP-11上运行"RSX-11M或IAS操作系统"似的。

本册系根据"VAX Architecture Handbook（81年版）"，"VAX Hardware Handbook（82-83年版），"VAX Technical Summary（80年版）"和"VAX-11/780 TB/Cache/SBI Control Technical description（79年版）"编译而成。编写过程中，尽可能将有关材料归纳合并，力图简明扼要。但由于水平有限，不恰当和错误之处，望读者指正。

内 容 介 绍

《VAX计算机系统的研究与应用》共有六个分册，150万字，分上、下两册出书。该书对VAX计算机系统的使用和维护人员以及有关科研人员和工程技术人员均有较大参考价值。各分册主要内容如下：

1．VAX计算机系统体系结构与硬件设计：主要介绍VAX系统结构、指令系统及为深入了解其它内容所需要的软硬件知识。

2．VAX/VMS的系统内部与数据结构：主要介绍VAX/VMS是如何执行任务的。为此叙述了VMS维护和管理的数据结构，讨论了在用户和VMS之间以及VMS本身的传递控制的结构。还叙述了VAX硬件在VMS使用时的特点。这些内容包含了执行的主要成分，如系统初始化和全部系统服务的操作。

3．VAX诊断程序设计指南：主要介绍VAX-11诊断系统，提供了VAX诊断原理和过程的概貌，说明如何书写VAX诊断程序。为维护工程师提供了VAX的硬件验证，排除故障和维护机器的工具。

4．DECnet-VAX的使用和管理：介绍了建立和维护DECnet-VAX局部和远程网络的知识。特别叙述了DECnet-VAX网络控制程序（NCP）的应用等。

5．BLISS语言：这是一种高级的系统实现语言，专门用于开发系统软件、数据库和文件系统、通信及应用软件。学习它，可以阅读、书写和修改用BLISS语言书写的源程序。

6．VAX/VMS设备驱动程序：介绍在VAX/VMS系统中如何书写或修改单总线或多总线的设备驱动程序。还介绍了学习VMS用到的I/O结构和几种驱动程序调试技术。

目 录（上 册）

VAX-11计算机系统体系结构与硬件设计（第一分册）

VAX/VMS的系统内部与数据结构（第二分册）

第一章 体系结构概述

1.1 引 言

VAX系列是一个32位、带有虚拟存储器的小型计算机系列。设计这个系列的目的是要达到：高速、易于使用和高度可靠。在今后十年或更多年内能满足用户提出的各种使用要求，并能获得广泛的应用(OEM、实验室实时处理、分布式数据处理、人机会话等)。

VAX计算机系列的核心是它的体系结构。体系结构是提炼与集中对系列中各机种都适合的公共属性，以保证软件在系列的所有机种上运行时不必加以修改，尤其是指令系统，存储管理的算法和有助于确定关联与进程设计中的某些方面，VAX都设法制定得比较确当。

在体系结构和体系结构的实现之间是有所不同的。例如，打字机的体系结构基本上是固定的：它是键盘的设计，键盘上的字母和标点符号是确定的，任何打字员都会使用它并处理其作业。但是，每个制造者可以以各自的方法实现其体系结构。某些制造者可能采用击打式打印键；某些可能采用球珠打印头；某些可能采用蓝色的键盘；某些可能采用黑色的。另外，制造者还可以根据性能对比作某些折衷。不过，不管其实现方法如何不同，所有机器仍具有相同的功能：打印。

计算机与它有类似之处。在系列中各台计算机的体系结构是一样的，但每台计算机在实现方法和性能折衷方面会略有不同，而它们又都能满足由计算机体系结构提出的全部要求，而且在系列中的所有机种都能为用户提供同样的服务。例如，一个掌握了VAX指令系统的程序员在VAX的任何一台处理机上操作时，都能得到同样的结果。

VAX体系结构中最重要的二个方面是VAX的能力和PDP-11之间的兼容关系。因为用户在PDP-11计算机和软件方面投入了大量资金。为了保护这种投资，需要简化程序在VAX和PDP-11之间的来回传送和转换过程。DIGITAL公司使用“兼容”这个词来指出VAX和PDP之间的衔接关系。DIGITAL公司做出决定，要求VAX在接收PDP-11的大多数程序时应作最小的转换，要为那些已在PDP-11计算机上运行的各种应用程序提供一个良好的开发环境。当然，这时也需要提出某些限制。不过这些限制在许多情况下只是对要求在VAX计算机上运行的某些PDP-11程序进行一些简单的重新编译工作。由于VAX提供了硬件级的兼容方式，实际上，许多PDP-11的程序能在VAX上不加修改地运行。并在VAX多处理机环境下，兼容方式与本机方式能同时工作。

VAX的能力可体现在以下几个方面：

·VAX的体系结构中最明显的标志是其指令系统。VAX通过采用独立的操作符、数据类型和寻址方式设计一个使之易于利用，并适合于高级语言使用的指令系统。VAX已有304条指令供汇编级程序员方便地控制计算机的操作。每条指令有一个记忆符，一个与其操作相应的短的名称。此外，为使各种应用和操作系统的程序码更高效，有支持队列，易于存取的可变长度位字段，和为保留与恢复关联的一些简单指令。例如找第一位指令（有多种应用）使操作系统能确定可执行的具有最高优先权的进程；二条

特殊的关联转换指令，使每次操作只要用一条指令就能将进程关联的各变元装入和保存起来。这些指令有助于实现快速有效的再调度。

在设计VAX的体系结构时，特别重视对于经常使用的高级语言的语句结构，尽可能地用硬件来实现，因而用一条VAX指令就可以处理它们，这对缩短程序长度和提高执行速度是有益的。例如FORTRAN的计算转向语句可以转换为VAX-11/780的选择转移（case）指令；循环结构语句可以转换为加比较并转移（ACB）指令；调用功能的扩充可按长字边界对齐堆栈，保存用户指定的各寄存器并在返回时清除堆栈等。

VAX-11/780指令和数据的长度都是可变的。它们在实际存储器中不必按长字边界对齐，而可从任一字节地址（奇地址或偶地址）开始。因此，不需要变元的指令只占用一个字节，而其它指令可以根据变元数和寻址方式而占用2个、3个甚至最多达54个字节。按字界对界的优点是指令流和数据结构可被存进小得多的实际存储器中。

VAX处理机所提供的寻址方式除了11种与PDP-11的寻址方式相同外，还提供另外两种寻址方式：变址方式和字面值方式。VAX通过对数据类型和寻址方式的广泛集中达到高的位利用率。

由于指令系统是如此灵活，所以执行大多数功能所需要的指令条数和存储容量较之非VAX处理机要少。其结果必然使程序更紧凑和更有效，程序执行和关联转换的速度更快，数学函数的运算更精确和更快速，并改进了编译程序所生成的程序码。

• VAX的另一个主要特点是采用虚拟地址空间。VAX这几个字就是暗示VAX计算机的主要特点——虚地址扩充（Virtual Address eXtension）。VAX用32位地址来确定信息的字节单元，所以可有效地识别40亿以上的字节地址，这对小型机和程序员来说，不能不认为是一个十分巨大的地址空间。关于这个巨大的地址空间引人注意的地方是“虚拟”这个概念：其实对于实际处理数据来说，计算机的主存没有必要处理大到接近40亿字节的容量，而且计算机内也没有这么大的存储器。这只是利用一个灵巧的“存储管理”的软件和自动进行地址转换的硬件来实现的，使程序员认为：他好象真正在一个大的虚地址空间上执行操作一样。存储管理还处理存放程序的全部细节，最后把程序送入主存，由程序员在主存处理它们。

从程序员的观点来看，虚地址空间中的下半部分20亿字节全可用于程序。因此程序员不必采用复杂的复盖或分段技术把一个较大的程序挤入一个较小的地址范围。并且，VAX计算机内部的逻辑结构又能迅速地自动把全部程序的虚地址转换为实地址，在适宜的地方（盘或主存）存放程序和数据，在需要时把程序和数据从外存送到内存。因此虚存大大提高了编制程序的效率。

• 快速关联转换是存储管理的另一方面。VAX是一个高性能的多处理机系统：许多程序和程序员可以同时使用它，而且每一个都认为它自己在控制着处理机。实际上，计算机在每一时刻只处理一个程序或程序中的一部分，只是许多程序的关联从主存转入或转出而已。一个转入关联使程序进入运行状态；一个转出关联使程序处于等待中央处理机的状态。结果，在任一时刻，VAX处理机似乎同时在进行许多不同的活动。

• 由于预见到将来会有越来越多的应用，VAX的体系结构应是可扩充的。新的数据类型和指令的操作符应该可以有效地、简便地加入到指令系统中去，并与以前的指令系统相一致。为此在VAX中采用了一个24K字节的用户可写控制存储器（WCS）。用户通过对WCS编程的修改来修改或扩充本机方式的指令系统。最后，使VAX所采用的一种体系结构既能包括和适应VAX系列中

的现有全部机种，又能有利于今后的发展。

为方便起见，我们可将VAX的体系结构分为两个主要方面：应用程序员(或用户)方面和系统程序员方面。

VAX系列中的下列属性是属于“用户体系结构”的：40亿字节虚地址空间；数据类型；指令格式；寻址方式；处理机状态字（处理机状态长字中的低端字）；在本机方式指令系统中的用户方式指令；兼容方式指令系统；用户可见的异常处理。

VAX系列中的下列属性是属于“系统程序员的体系结构”的：特权指令；处理机状态长字的高端字；进程结构；存储管理；中断结构和异常处理；数据共享和同步。

下面就这两方面作一概述。

1.2 用户程序设计的处理概念

程序是用户向操作系统提出请求，要求进行转换、链接、和执行的指令与数据流。一道可执行的程序称为程序映象。程序映象是通过把源语言模块转换为目的模块，并把目的模块链接在一起而产生的。通常，映象是存放在磁盘上的一个文件。当用户运行映象时，操作系统从映象文件的副本读入实际存储器，并执行之。

过程是为了要解一个问题而对要执行逻辑所作的说明；就是说，它是一个算法的静态定义。一个映象是由若干过程和由链接程序约束在一起的数据组成。链接所涉及的问题是解决模块之间的交叉链接以及虚地址空间的分配。

一个映象执行时所处的环境是它的关联。一个映象的全部关联不仅包含其在任一时刻的执行状态，而且还包含资源分配特权和限额的定义，诸如设备占有权、文件存取和最大的实存分配。某些软件信息，包括某些关键地址和某些由软件关联组成的软件数据结构。一个映象关联和在关联中执行的映象称为一个进程。进程是本计算机系统中的一个完整执行单位。在典型情况下，它可以包括若干个运行的映象，一个接着另一个地运行。进程关联中包括当前正在执行的映象的状态和允许映象执行的各种限制，这些限制主要与执行此映象的用户特权有关。

当进程执行时，在实存中只存放必要的进程页面的子集（进程的工作集），进程的其它页面存放在二级存储器中。在进程竞争取得中央处理机资源以前，其工作集必须常驻在主存中。在进程执行期间，其工作集可写到二级存储器。释放实存作其它用处，称之为交换。常驻在实存中的进程集合称之为平衡集。

进程的虚地址空间

在存储器中存放的大多数数据都利用一个8位的字节地址来定位。每个字节单元可由一个32位的虚拟地址加以识别，在操作系统控制下，经由处理机把它们转换后才能成为访问存储器的实地址。

虚拟地址不像实际存储地址那样是存储器单元中的唯一地址。例如，利用同一虚地址的两道程序可以访问二个不同的实际存储器单元；或者两道程序可以访问利用不同虚地址的同一个实际存储单元。

全部可能的32位虚拟地址的集合称为“虚拟地址空间”。它可被看作为由0至$2^{32}-1$个字节单元组成的数组。这个空间根据某些使用可划分为二个空间：被进程使用的那一半虚地址空间被指定为进程空间。余下的另一半地址由操作系统占有和保护，被指定为系统空间。

数据类型

指令操作数的数据类型说明存储器内的多少位应被看作为一个处理单位，以及对这个单位应作何解释。弄清操作数的数据类型是很重要的，因为在以后的章节中会见到，相同的位模式可以解释为非常不同的数据项；而且，不同的位模式又可以用来表示同

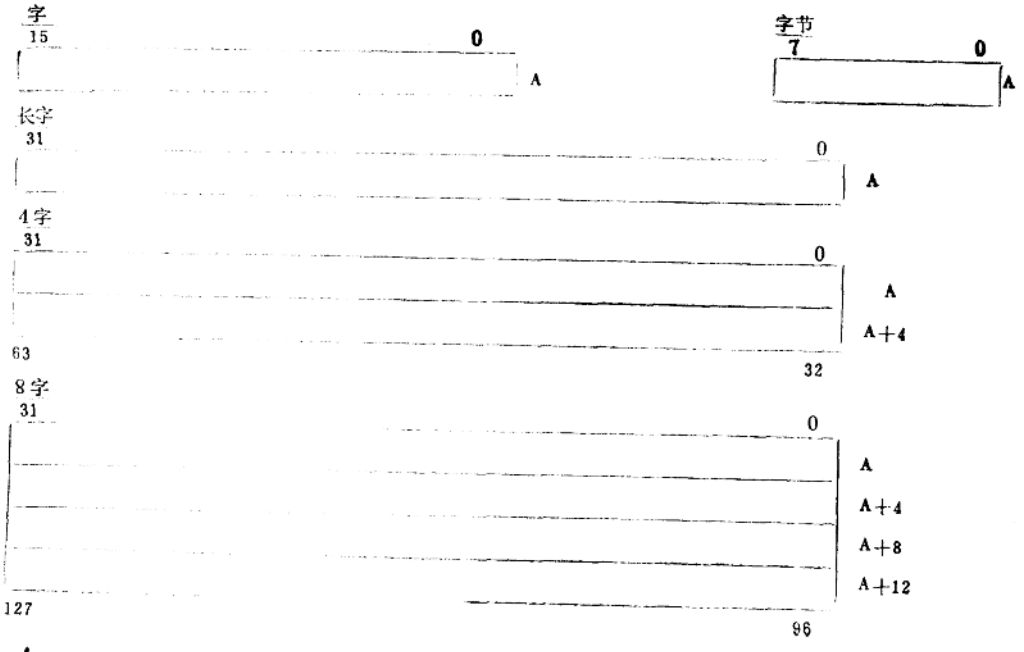
样的数据类型。

处理机的本机指令系统可以识别六种基本数据类型：整数、浮点、字符串、压缩十进制数串、数字串和可变长度位字段。

这六种基本数据类型都还有若干变化，如表1所示。图1中列举了各种数据类型的图解。

表1　数　据　类　型

数据类型	长　度	范　围（十进制）	
整　数		带符号	不带符号
字节	8位	−128到+127	0到255
字	16位	−32768到+32768	0到65535
长字	32位	-2^{31}到$+2^{31}-1$	0到$2^{32}-1$
4字	64位	-2^{63}到$+2^{63}-1$	0到$2^{64}-1$
8字	128位	-2^{127}到$+2^{127}$	0到$+2^{128}-1$
浮　点			
F浮点	32位	大约为七个十进制数字的精度	
D浮点	64位	大约为十六个十进制数字的精度	
G浮点	64位	大约为十五个十进制数字的精度	
H浮点	128位	大约为卅三个十进制数字的精度	
压缩十进制串	0到16个字节（31数字）	数，每个字节二个数字　符号位于最后一个字节的低半部分	
字符串	0到65535字节	每个字节一个字符	
可变长度位字段	0到32位	取决于解释	
数字串	0到31字节	$-10^{31}-1$到$+10^{31}-1$	
队列	≥2长字/队列条目	0到20亿条目	



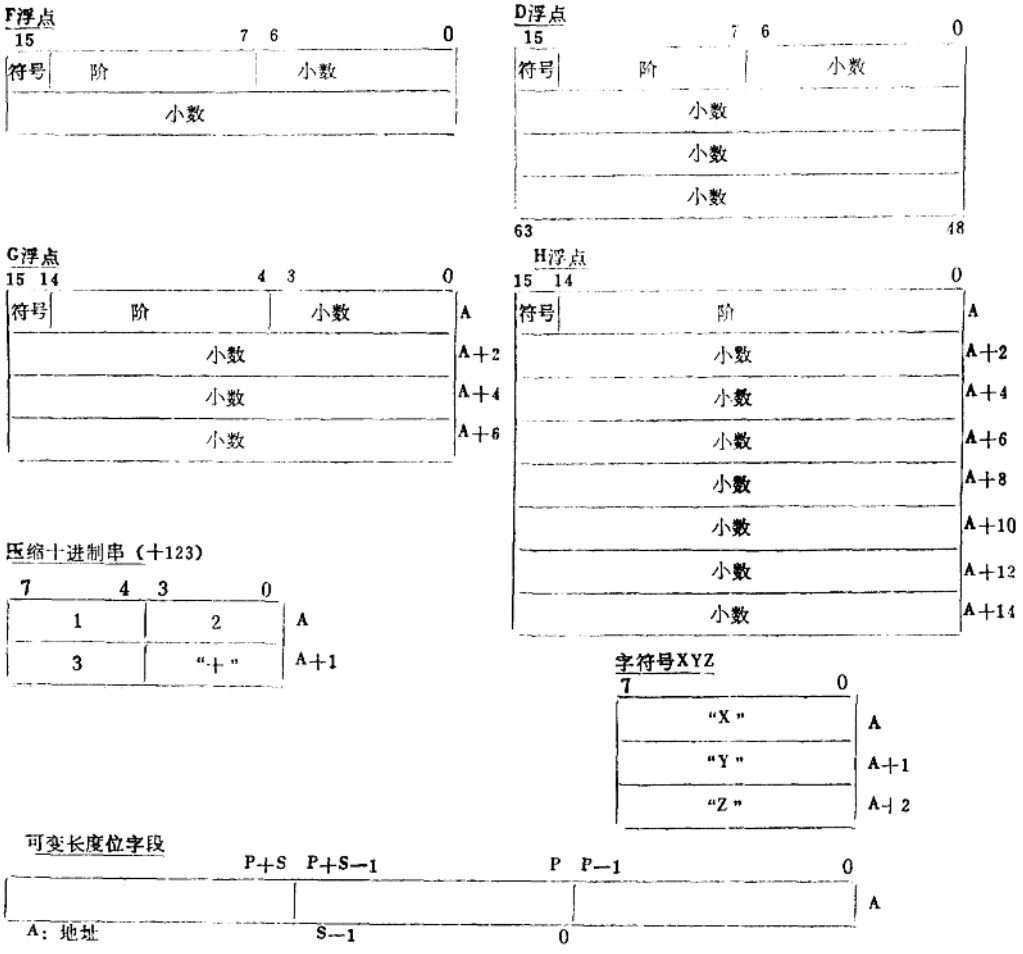


图 1　数据类型的表示法

任何数据项的地址都是该数据项所在的第一字节的地址。所有整数、浮点、压缩十进制数和字符数据都可以从任一字节边界开始存放。但是位字段不必在一个字节的边界上开始。位字段为一组连续的位（0-32），其开始位的位置相对于某一给定字节地址。本机指令系统可将位字段解释为一个带符号或不带符号的整数。

可变长度位字段数据可将小量整数装配在一起后放在一个较大的数据结构中，以此提高存储器的利用率。

队列数据为保持在一个循环的双链接表中（就是说，每个条目伴随二个长字，一个告诉其后面条目的单元，另一个指明其前面条目的单元。）的一种数据类型。这种数据类型又可分为绝对队列和相对队列两种类型。这在以后章节中将详细描述。

指令格式

处理机可以以两种方式中的任一种方式执行指令：本机方式和兼容方式。本机方式是机器执行指令的主要状态，兼容方式是机器执行指令的第二种状态。在兼容方式中，处理机执行一组 PDP-11指令，识别整数数据和使用8个16位通用寄存器。本机方式时，处理机执行大量可变长度指令（指令可在任意字节边界开始），识别各种数据类型和使

用16个32位通用寄存器。VAX的可变长度指令格式不仅使代码更为紧凑，而且还保证指令系统易于扩充。操作码有单字节和双字节两种。根据不同的指令，其后可以跟着0至6个操作数说明符。根据不同的寻址方式，操作数说明符可以是1至10个字节长。图2列举了自动递减方式的传送长字指令，开始的一串字节为操作码，其后有两个操作数说明符。在此例中，假设开始单元为00003000。当处理机完成一条指令的执行时，程序计数器内有下一条指令的第一个字节的地址。程序计数器的操作对程序员总是透明的。

程序计数器本身可被用来指明操作数的位置。汇编程序将许多操作数的访问类型转换为利用程序计数器的各种寻址方式。利用程序计数器的自动递增方式也叫立即方式，被用来直接指明常数而不是指明用字面值方式寻址的地址。利用程序计数器的自动递增延缓方式也叫绝对方式，被用来访问一个绝对地址。利用程序计数器的位移延缓方式来指明距当前位置有多少偏移值的一个操作数的位置。

利用程序计数器的寻址方式允许编写与位置无关的指令代码。位置无关代码经链接以后可在虚地址空间的任何地方执行，因为程序的链接信息可以以虚地址空间中的绝对位置来加以指明，而所有其它的地址可以用相对于当前指令的地址来指明其位置。

机器码：（假设开始单元00003000）

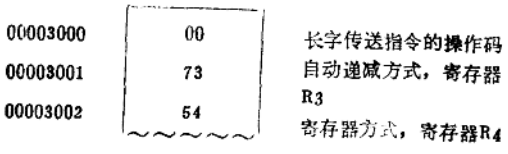


图2　自动递减长字传送指令

通用寄存器和寻址方式

在处理机内部有一些叫做通用寄存器的单元。这些单元可用作为暂时数据存储器和供寻址使用的单元。16个32位通用寄存器可供本机指令系统使用，但其中有些有特定的含义。有的寄存器被指定为程序计算器，有的寄存器被指定用于例行程序的链接：堆栈指示字、变元指示字和栈帧指示字。

指令的操作数可位于主存、通用寄存器或指令流本身之中。指明操作数位于什么地方的方法叫做操作数寻址方式。VAX处理机有各种寻址方式和寻址方式的优化方法：一种寻址方式是定位在寄存器中的操作数；另外几种寻址方式是定位在存储器中的操作数，并用寄存器来指明 1)操作数， 2)操作数的表， 3)操作数地址的表。

还有一些寻址方式是用变址修改的寻址方式，定位于存储器中的操作数。最后，还有两种寻址方式，指明在指令流中操作数的位置，一种是常数数据，另一种是转移指令的地址。VAX的寻址方式扼要地概括在表2中，并将在第四章给出详细的解释和示例。

表2　寻址方式

字面值方式 （立即方式）	{S↑ / I↑} ● 常数	
寄存器方式	Rn	变址〔Rx〕
寄存器递延方式 自减方式	(Rn) — (Rn)	
自增方式	(Rn) +	
自增递延方式 （绝对方式）	@ (Rn) + @● 地址	
位移方式	{B↑ / W↑ / L↑} 位移值 (Rn) 地　址	
位移递延方式	@ {B↑ / W↑ / L↑} 位移值 (Rn) 地　址	

1.3 系统程序设计的处理概念

VAX处理机为支持高性能多道程序设计的环境而进行了专门的设计。多道程序设计的主要优点是系统能使若干任务共享计算

机资源，同时执行多个应用。

支持多道程序设计的特点是：

- 快的关联转换
- 优先权指派
- 虚拟寻址和存储管理

作为一个多道程序系统，VAX不仅能使若干进程共享处理机，而且能在它们进行相互通信和共享代码与数据的同时，保护进程，以免一个进程被另一个进程破坏。

关联转换

在多道程序的环境中，若干独立的指令流可以准备在任一时刻执行，而不像在批处理中的指令流完全串行地执行。而且，操作系统能在指令流之间对它们进行干预和转换。转换是以平衡的方式利用计算机资源，提高其效率的一种方法，也是对要求优先处理的事件或进程进行干预的一种方法。

为了快速转换，在VAX中由硬件建立起快速转换的环境。对于每个选来执行的进程，操作系统产生一个称为软件进程控制块的数据结构。在软件进程控制块内有一个指向硬件进程控制块的数据结构指示字。硬件进程控制块内又包含有硬件进程关联。硬件进程关联可以在任一时刻确定正在执行的程序流。所以有关关联转换的信息必须装入进程控制块的可编程寄存器中。这些信息指明：

- 指令或数据流中的指令和数据存放的位置
- 下一时期将执行哪一条指令
- 在执行期间处理机的状态

进程可认为是由硬件关联定义的指令和数据流，所以操作系统在实现进程之间的转换时只要通过请求处理机保留一个进程的硬件关联和装入另一个进程的硬件关联就可实现之。为此，VAX提供了二条有关加速关联转换的指令。操作系统把硬件关联的实地址装入处理机的进程控制块的基地址寄存器，并发出装入进程关联的指令，则处理机用一次操作就可把进程关联装入相应的寄存器，然后准备在进程关联内执行之。所以保留进程关联和装入进程关联指令是使进程进行迅速转换的重要因素。

进程控制块内不仅含有可编程序寄存器的状态，而且还含有进程虚地址空间的定义信息，因而进程的地址转换是由进程关联自动地转换的。

此外，进程控制块还为用户进程提供了触发异步系统自陷（AST）的机构。异步系统自陷字段允许处理机调度一个软件中断来启动一个AST例行程序，并保证把它们递交给对进程合适的访问方式。

优先权指派

当在一个进程的关联中运行时，处理机执行指令和控制在外围设备和主存之间来往的数据流。为在许多进程之间共享处理机，存储器和外围设备资源，处理机有两种仲裁的机构来支持特高性能的多道程序：异常和中断。异常是相对于一个特定的指令流执行同步发生的事件（可预测的），而中断是异步发生的一些外部事件。

执行流可在任何时刻改变，而且可以由处理机来区别执行流的变化是在进程内部，还是在系统范围内。进程内部的变化是由于用户软件有错或者是在用户软件调用操作系统服务时发生的。进程内部的变化可以通过处理机的异常检测机构和操作系统的异常指派程序来处理它们。

系统范围执行流的变化通常由来自设备的中断或由操作系统软件产生的中断引起。这些中断由处理机的中断检测机构和操作系统的中断服务例行程序来处理。（系统范围的执行流的改变还可由严重的硬件错误产生，在此情况，它们可被处理为特定的异常或者高优先权的中断。）

系统范围执行流变化的优先级别高于进程范围执行流变化的优先级别。而且，处理机利用优先权系统为中断提供服务。对每一种中断赋予一个优先权，而处理机响应最高

优先权的待处理中断。例如，来自高速磁盘设备的中断优先于来自低速设备的中断。

处理机可以在二条指令之间进行中断服务，也可以在执行长的迭代指令的过程中规定的断点处进行中断服务。当处理机确认中断时，它迅速地转换到一个特定的系统范围关联，以使操作系统能为中断服务。处理系统范围内执行流的变化对各个进程来说，完全是透明的。

1.4 系统程序设计的环境

在任何一个进程关联内，用户级软件利用指令系统、通用寄存器和处理机状态字控制进程的执行。在多道程序设计环境内，操作系统利用专门的指令、处理机状态长字和内部处理机寄存器控制系统的执行。

处理机状态长字

处理机状态长字（PSL）存放在处理机的一个寄存器中，它决定处理机在任何时刻时的执行状态。处理机状态长字的低16位是供用户使用的处理机状态字，高16位是供系统特权控制的处理机状态字。

PSL字段可以按其功能分组控制：

- 控制当前指令的访问方式
- 控制处理机正在执行的指令系统
- 控制中断处理

处理机的访问方式

在一个高性能的多道程序设计系统中，处理机在竞争系统资源的所有进程之间必须提供共享和保护的基础。在本系统中，保护的基础是处理机的访问方式。访问方式负责确定两件事：

- 指令执行的特权，处理机将执行什么指令
- 存储访问的特权，当前指令可以访问存储器中的哪些单元

在任何时候，处理机不是在特定的进程中执行指令代码，就是在系统范围的中断服务关联中执行。在一个进程关联中，处理机可以识别4种访问方式：内核、执行、管理和用户。内核方式是具有最高特权的方式，用户方式是具有最低特权的方式。

处理机在大部分时间内是以用户方式在一个进程关联或另一个进程关联中执行操作。当用户软件需要操作系统提供更多的特权服务时，不论是要求获取资源，要求对I/O进行处理，还是要求提供信息，都需要调用这些服务。并在处理机执行它们时，处理机可以以进程的同一访问方式服务，也可以以更高特权的访问方式服务。

只有在内核方式时，处理机才能执行下述指令：打处理机、装入和保留进程关联或者访问控制存储管理的内部寄存器、中断处理和控制台的内部寄存器或者处理机时钟的内部寄存器。

以一种更高特权执行程序码的能力是由系统管理人员给予的，并由操作系统控制的。通常，以某种访问方式执行的程序码能够保护其自己及其数据结构的任何部分，它们不致于被其它以任何更低特权方式执行的程序码进行读或写的访问。

保护和特权指令

处理机提供三种类型的指令，使用户方式软件获取特权服务而又不危及操作系统的完整性，它们为：

- 改变方式指令
- 探测指令（PROBE）
- 从异常或中断返回的指令

用户方式软件通过用一条标准的调用（CALL）指令来调用操作系统服务过程而获取特权服务。操作系统的服务指派程序在实际进入该过程以前发出一条相应的改变方式指令。改变方式指令是一条特殊的自陷指令，也可认为是操作系统的一条服务调用指令。最终，由系统管理员授予特权编写各种程序码，将改变方式自陷处理为更高特权的

访问方式的程序码。

PROBE指令允许过程对照请求访问某一特定单元的调用程序的特权来检查在存储器中该页面的读（PROBER）和写（PROBEW）的访问保护。这使操作系统既能为较低特权的调用程序提供以特权方式执行的服务，又能防止调用程序访问保护的存储区。

操作系统的特权服务过程和中断以及异常服务例行程序利用从异常或中断返回的指令（REI）退出执行。REI是唯一降低处理机访问方式特权的方法。REI指令恢复程序计数器和处理机的状态以使进程回到被中断的点。

当操作系统调度一个关联转换操作时，关联转换过程使用保留关联进程（SVPCTX）和装入进程关联（LDPCTX）指令来保留当前进程和装入另一进程关联。操作系统的关联转换过程是通过更新一个内部处理机寄存器来指出硬件关联将被装入的位置。

存储管理

处理机除了在访问方式之间实现存储保护功能以外，存储管理硬件还能使操作系统提供一个绝对灵活的和有效的虚拟存储程序设计的环境。虚拟地址空间由最大可能的32位地址组成。它可以通过程序和处理机之间的转换来指明在实际存储器中的一个字节单元，把虚拟地址转换为实存中的地址。实际地址为处理机、存储器和外围接口之间进行交换的地址。（对于只涉及虚地址的程序员来说，实地址是透明的。）

虚地址空间被划分为若干页面，每一页面表示为512个连续编址的存储器字节。第一页从字节0开始，其后继续511个字节。第二页从512字节开始，其后继续至1023字节，以此类推。如果我们将前三页的虚地址空间，以十进制和十六进制记数法来表示它们的地址，则可表示如下：

页	地址$_{10}$	地址$_{16}$
0	0000—0511	0000—01FF
1	0512--1023	0200—03FF
2	1024—1535	0400—05FF

为了使存储器映像有效，处理机必须能迅速地进行虚地址与实地址之间的转换。进行快速地址转换的二个要点是处理机内部地址转换缓冲器和转换算法本身。

处理机有三对用于页面映象的寄存器，这三对寄存器与虚地址空间的三个主要区域相对应。操作系统存储管理软件把叫做页表的数据结构的长度和基地址装入各对寄存器，页表为系统中的每个虚页面提供映像的信息。因此三个区域的每一个都有一个页表。

页表是一个由页表条目组成的连续的数组，每一页表条目包含一个虚页的实际地址，因此，为了把虚地址转换为实地址，处理机只要把虚页号作为从给定页表的基地址开始算起的页表的索引，就可以从该索引指向的页表条目内找到实地址。

所有的进程页表都在虚地址空间的系统区内，其地址为虚地址，但是系统区的页表本身由它在实际存储器中的页表地址来定位。就是说，系统区的页表的基地址寄存器包含有页表基地址的实际地址，而进程页表基地址寄存器包含有进程的页表基地址的虚地址。

利用虚地址作为每个进程页表的基地址有两个优点。第一个优点是所有页表不必都存放在实际存储器中，只有系统区的页表才需要常驻在实际存储器中。进程页表可常驻在磁盘上，在必要时，才由它们自己进行调页或交换。

第二个优点是操作系统的存储管理软件可以动态地分配进程页表，因为它们不必映射为连续的实页面。虽然系统区的页表必须映射为连续的实页面，但这种要求并不限制实存储器的分配。在进程之间共享系统区却可以在关联转换时不再要求重新定义。