微处理器与微型计算机

第　三　册

〔美　国〕 BRANKO SOUČEK 著

南通电子仪器厂技术情报室　译

南京工学院计算机教研室　校

南通电子仪器厂技术情报室

目　　录

第 三 部 分

新型微处理器与专用微系统

第 十 二 章

PDP—11小型计算机与LSI—11微型计算机

本章介绍软件兼容的两个系统，PDP—11小型计算机与LSI—11微型计算机。它们都是16－位系统，具有高效能的指令系统，并有多种寻址方式。PDP—11是一种硬线机器，而LSI—11则用其存贮在控制只读存贮器里的微程序来实施PDP—11的指令系统。指令内容比任何微型计算机或小型计算机都更完整，由400多条指令构成。PDP—11与LSI—11的全部操作都是用同一组指令完成的。因为外围设备寄存器可由中央处理器按需要灵活处理，因此，用来处理存贮数据的指令也同样可以处理外围设备寄存器里的数据。本章还举例说明了系统结构，以及程序设计与接口技术。

12.1 PDP—11小型计算机

操作特点

所有PDP—11计算机都具有下列特点：

- 16－位字(两个8－位字节)
- 字或字节处理　无需进行旋转、交换或屏蔽之类的操作即可十分有效地处理8－位字符。
- 异步操作　系统部件可按其最高速度运行；换上速度更快的设备就可有更快的操作，而无需改变其硬件或软件。
- 组件式的器件　组成系统时十分简易、灵活。
- 堆栈处理　硬件的顺序存贮，使结构数据、子程序及中断的处理来得容易。

- 直接存贮器取存(DMA)　　多设备的直接存贮器存取是本结构固有的特点。
- 八个通用寄存器　　累加器或地址的产生采用快速集成电路。
- 自动优先中断　　4－线的多级系统可按响应要求将中断线分组。
- 矢量中断　　无需进行设备查询，中断响应快速。
- 单操作数及双操作数的指令　　程序设计指令系统效能高，使用方便。
- 电源故障与自动再起动　　对AC电源的起伏具有硬件检测与软件预防措施。

所有的计算机部件与外围设备之间的相互连接和信息交换都是通过一根称为单一总线的高速总线实现的——单一总线是PDP—11的许多独特功能的关键。地址、数据及控制信息都沿此单一总线的56根线传送。

单一总线上所有设备的信息交换形式都相同。处理器与存贮器交换信息时所用的信号和它与外围设备交换信息时所用的信号是同一组信号。外围设备与处理器、存贮器或其它外围设备进行信息交换时也使用这一组信号。每一个器件，包括存贮单元、处理器的寄存器以及外围设备的寄存器在内，在单一总线上都有一指定地址。因此，中央处理器处理外围设备寄存器时，可以象处理磁芯存贮器那样机动灵活。凡是可用于磁芯存贮器数据的指令都同样可以很好地用于外围设备寄存器。PDP—11指令的这种可把任一存贮单元当作一个累加器来处理的特殊本领，是其效能极高的突出所在。

由于可通过单一总线进行双向异步信息交换，各设备都可独立地进行数据的传送、接受及交换，而无需处理器的干预。例如，阴极射线管的显示可由磁盘文件自行刷新，而中央处理器则可从事其它任务。由于单一总线是异步操作的，所以它所能兼容的设备操作速度范围很广。

系统概述

整个计算机是围绕着一根单一总线布局的。处理器、存贮器以及所有的外围设备都共用这根高速总线。单一总线使处理器可以把外围设备都看作是实施功能的有效存贮单元。因此，外围设备也象存贮器那样可加以访问。换句话说，访问存贮器的指令也能直接用于外围设备中的控制寄存器、状态寄存器及数据寄存器。计算机字长16位。

本计算机有八个通用寄存器，R0—R7。全部寄存器都可按程序存取，可以用作累加器、存贮单元的指示器，或全字变址寄存器。寄存器R0—R5用于通用存取，而R6和R7两个寄存器则分别用作堆栈指示器和程序计数器：

R6≡SP　　（堆栈指示器）

R7≡PC　　（程序计数器）

图12.1(a)为处理器的简化方框图，分为五个主要的功能区域：单一总线、接口、数据通路、通用寄存器、及定时状态与控制逻辑。

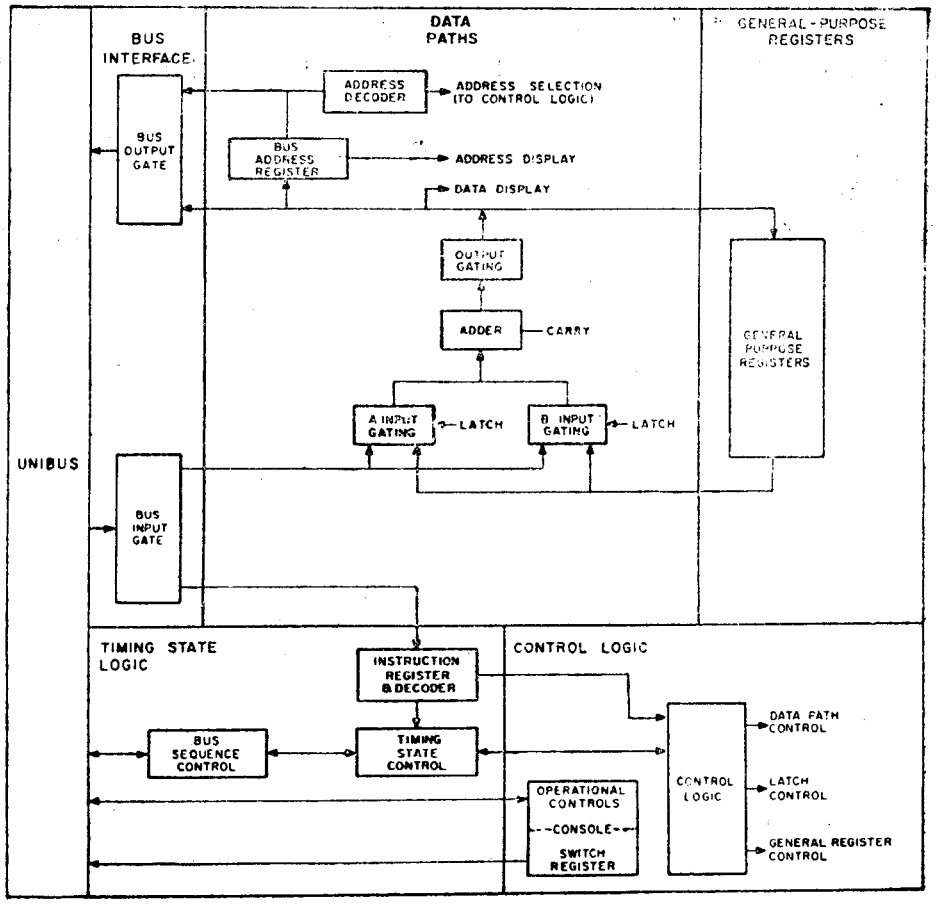

图12.1　PDP—11处理器方框图(a)简化的内部结构；(b)数据流

所有信息都通过数据通路传送。总线接口与通用寄存器之间的连接只经过数据通路，使频繁的数据传送形成一个“8”字形的信息流，见图12.1(b)。此信息流途径是从单一总线到数据通路的底部，从数据通路的顶部到通用寄存器，从通用寄存器到数据通路的底部，再从数据通路的顶部到单一总线。

寻址是通过通用寄存器进行的。通用寄存器可变替地用作变址寄存器或顺序表指示器，以便存取表格数据。地址运算可在通用寄存器中直接进行。

图12.2(a)给出了单操作指令的一般形式，它以三位规定寄存器，另外三位规定寻址方式。

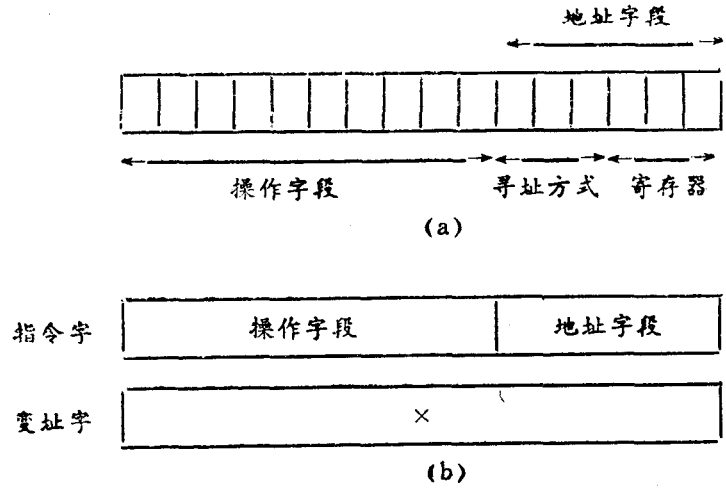

图12.2　指令格式

12.2　寻址方式

直接寄存器寻址

通用寄存器可用作简单的累加器，用来操作那些需要频繁地存取的变量。在直接存贮器寻址方式中，操作数直接保存在一个通用寄存器中。汇编程序把形如

OPR　R

的指令翻译成通用寄存器操作。R是寄存器的名称，OPR表示一般指令的记忆码。

例、求补操作COM R2。这条指令使R2的内容求补。这个例子可用图12.3加以说明，要求补的操作数是014。

间接或延期寻址

指定用作地址的(间接的或延期的)操作数由符号@向汇编程序说明。因此，形如：

OPR@R或OPR(R)

的指令表示延期寄存器寻址。寄存器内容即为该操作数地址。

例、指令COM@R2。图12.3中所要求补的操作数即为1234。

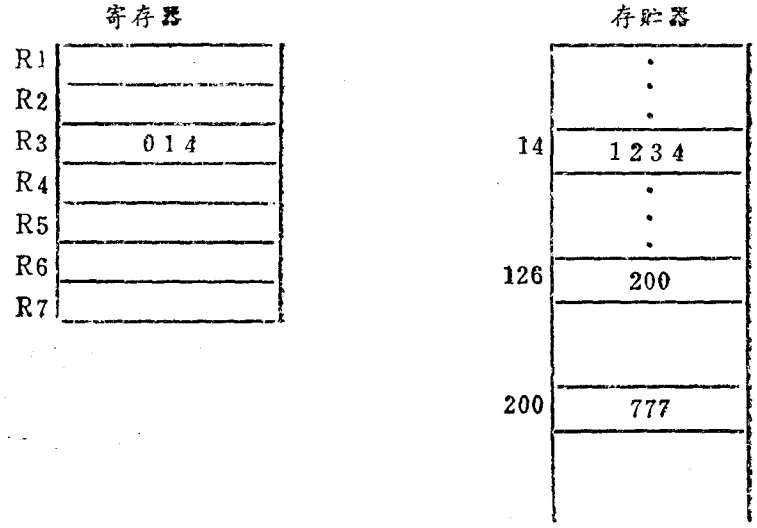

图12.3　寻址方式

变址寻址

通用寄存器可用作变址寄存器，实现对数据表或数据堆栈的随机存取。下列形式的指令

OPR X(R)

表示变址方式寻址。其有效地址为X(存贮单元地址)与指定通用寄存器R的内容之和。

在本计算机中，汇编程序将把变址指令译成两个二进制字，如图12.2(b)。

作为一个例子，设计算机需要执行求补指令COM112(R2)。假定变址寄存器R2内容为014。那么，操作数的有效地址即为112+014=126。一般情况下即为

M，有效地址=X+(R)

本例可由图12.3说明。有效地址为126，所要求补的操作数即为200。

变址方式寻址可以延期或间接实现。这可由下列形式的指令表示：

OPR@×(R)

变址地址的内容即为有效地址：

M，有效地址=(X+(R))

例、COM@112(R2)。R2的内容为014，见图12.3。变址地址即为112+014=126。因此，有效地址为200，所要求补的操作数为777。

在进行程序循环操作及数据表的处理中，变址方法是十分有用的。设在每次循环中都要用到同一个操作数，但它在数据表中都处于下一项。为了使操作得到修改，程序每通过一次循环就使变址寄存器加1即可。X表示数据表的起始地址(在上述例子中为112)。

变址寄存器的内容最初可以为0，经过第一次循环后将增量为1，经第二次后即为2，等等。因此，操作的修改过程为：

第一次循环　　COM112
第二次循环　　COM113
第三次循环　　COM114

变址寄存器的增量是根据指令要求自动进行的，因此，称为自动增量。

自动增量与自动减量寻址

自动增量寻址使指示器可按操作数据表的顺序自动步进。可用于变址操作或通用寄存器变址操作。

在这种操作方式中，从通用寄存器取得操作数地址，寄存器内容随即前进一步(增量)而指出下一个字的地址。下列形式的指令

OPR (R)+

表示自动增量寻址。

同样，指令

OPR -(R)

表示自动减量寻址。

这两种寻址方式也可以有间接或延迟操作，其指令形式为：

OPR@(R)+
OPR@-(R)

立即寻址

立即寻址通过把常数包括进指令的办法，使对恒定的操作数的存取在时间与空间方面得到改进。在本计算机中，汇编程序把写成#2形式的地址看作为立即操作数：

OPR #n

这种指令被汇编成两个计算机字，第二个字是操作数n。

绝对寻址、相对寻址及延迟—相对寻址。

绝对寻址、相对寻址及延迟—相对寻址都是标准的寻址方式，但这里仅定义一下它们的形式。

绝对寻址：

OPR@#A

相对寻址(相对于程序计数器)：

OPR@A

延迟－相对寻址：

OPR@A

这三种指令都被汇编成二个计算机字，第二个字是位移A。

12.3 指令系统与程序设计举例

指令系统

本计算机有一个处理器状态寄存器(PS)，见图12.4，其内容为处理器，现行优先权和先前各操作结果的信息，以及对程序排错期间所要捕捉的指令的实施情况进行检测的指示信息。PS的四个位被用来监控先前各指令的不同结果。这四个位是：

Z，结果为0时。
N，结果为负时。
C，操作结果引起最高有效位进位时。
V，操作结果导致溢出时。

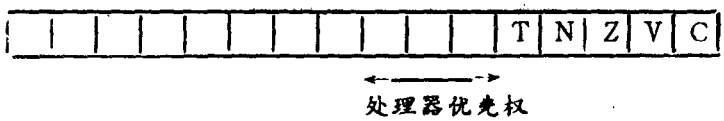

图12.4 处理器状态寄存器(PS)

指令系统见表12.1，说明如下，

1.本计算机有一组双操作指令。例如，指令MOVsrc,dst,它将从源端取得操作数传送到目的端。源端和目的端的寻址方式与单操作指令的寻址方式相同，

通用寄存器寻址：OPR RX, RY。
延迟寻址：OPR@RX, @RV或OPR(RX), (RY)。
变址寻址：OPR A(RX), B(RY)。
延迟变址寻址：OPR@A(RX), @B(RY)。
自动增量：OPR(RX)+, (RV)+。
延迟自动增量：OPR@(RX)+, @(RY)+。
自动减量：OPR－(RX), －(RY)。
延迟自动减量：OPR@－(RX), @－(RY)。
立即寻址：OPR＊C, DEST。
绝对寻址：OPR@＊A, DEST。

表12.1　PDP—11指令系统

记忆码	指　令　操　作	操作码	条件码 ZNCV	定时
双操作数指令：OPRsrc,dst				
MOV(B)	输送(字节) (src)→(dst)	.1SSDD	✓✓－0	2.3
CMP(B)	比较(字节) (src)→(dst)	.2SSDD	✓✓✓✓	2.3*
BIT(B)	位测试(字节) (src)∧(dst)	.3SSDD	✓✓－0	2.9*
BIC(B)	位清除(字节) ～(src)∧(dst)→(dst)	.4SSDD	✓✓－0	2.9
BIS(B)	位置位(字节) (src)∨(dst)→(dst)	.5SSDD	✓✓－0	2.3
ADD	加 (src)+(dst)→(dst)	06SSDD	✓✓✓✓	2.3
SUB	减 (dst)－(src)→(dst)	16SSDD	✓✓✓✓	2.3
条件转移指令：Bxx loc				
BR	转移(无条件) loc→(PC)	0004XX	——	2.6－
BNE	不相等则转移 loc→(PC)，若Z＝0	0010XX	——	2.6－
BEQ	相等则转移 loc→(PC)，若Z＝1	0014XX	——	2.6－
BGE	大于或相等则转移 loc→(PC)，若N∀V＝0	0020XX	——	2.6－
BLT	小于则转移 loc→(PC)，若N∀V＝1	0024XX	——	2.6－
BGT	大于则转移 loc→(PC)，若Z∨(N∀V＝0)	0030XX	——	2.6－
BLE	小于或等于则转移 loc→(PC)，若Z∨(N∀V)＝1	0034XX	——	2.6－
BPL	正则转移 loc→(PC)，若N＝0	1000XX	——	2.6－
BMI	负则转移 loc→(PC)，若N＝1	1004XX	——	2.6－
BHI	高于则转移 loc→(PC)，若C∨Z＝0	1010XX	——	2.6－
BLOS	低于或相同则转移 loc→(PC)，若C∨Z＝1	1014XX	——	2.6－
BVC	溢出位清除则转移 loc→(PC)，若V＝0	1020XX	——	2.6－
BVS	溢出位置位则转移	1024XX	——	2.6－

	loc→(PC)，若V=1			
BCC	进位位清除则转移	1030XX	——	2.6-
(或BHIS)	loc→(PC)，若C=0			
BCS	进位位置位则转移	1034XX	——	2.6-
(或BLO)	loc→(PC)，若C=1			
子程序调用指令：JSR reg,dst				
JSR	跳至子程序	004RDD	——	4.4
	(dst)→(tmp),(reg)↓			
	(PC)→(reg),(tmo)→(PC)			
子程序返回指令：RTS reg				
RTS	从子程序返回	00020R	——	3.5
	(reg)→(PC)，↑(reg)			
单操作指令：OPR dst				
CLR(B)	清除(字节)	.050DD	1000	2.3
	0→(dst)			
COM(B)	求补(字节)	.051DD	√√00	2.3
	～(dst)→(dst)			
INC(B)	增量字节	.052DD	√√-√	2.3
	(dst)+1→(dst)			
DEC(B)	减量(字节)	.053DD	√√-√	2.3
	(dst)-1→(dst)			
NEG(B)	求反(字节)	.054DD	√√√√	2.3
	～(dst)+1→(dst)			
ADC(B)	加进位(字节)	.055DD	√√√√	2.3
	(dst)+(C)→(dst)			
SBC(B)	减进位(字节)	.056DD	√√√√	2.3*
	(dst)-(C)→(dst)			
TST(B)	测试(字节)	.057DD	√√00	2.3*
	0—(dst)			
ROR(B)	右旋(字节)	.060DD	√√√√	2.3*
	用进位位右旋1位			
ROL(B)	左旋(字节)	.061DD	√√√√	2.3*
	用进位位左旋1位			
ASR(B)	算术右移(字节)	.062DD	√√√√	2.3*
	以符号扩展右移			
ASL(B)	算术左移(字节)	.063DD	√√√√	2.3
	以低位0左移			
JMP	跳变	0001DD	——	1.2
	(dst)→(PC)			
SWAB	交换字节	0003DD	√√00	2.3
	字的字节交换			
条件码操作码：OPR				1.5

表12.1　PDP—11指令系统（续）

记忆码	指令操作	操作码	条件码 ZNCV	定时

条件码操作码用来使备条件码位的组合置位或清除。若S＝1，则被选位置位、否则清除。被置位或清除的是与下列字中加有标记的位对应的条件码位。

条件码操作码：

0	0	0	2	4	S	N	Z	V	C
15				5	4	3	2	1	0

因此，SEC＝000261使C位置位，而对其它条件码位无影响(CLC＝000241使C位清除)。

操作指令：OPR

HALT	停机	000000	——	1.8
	处理器停；(RO)与停机地址由指示灯显示。			
WAIT	等待	000001	——	1.8
	处理器断开总线，等待中断。			
RTI	从中断返回	000002	√√√√	4.8
	↑(PC)，↑(PS)			
IOT	输入/输出捕捉	000004	√√√√	9.3
	(PS)↓，(PC)↓，(20)→(PC)，(22)→(PS)			
RESET	复位	000005	——	20ms
	CP发出一个INIT脉冲			
EMT	仿真程序捕捉	104000—104377	√√√√	9.3
	(PS)↓，(PC)↓，(30)→(PC)，(32)→(PS)			
TRAP	捕捉	104400—104777	√√√√	9.3
	(PS)↓，(PC)↓，(34)→(PC)，(36)→(PS)			

相对寻址：OPRA，DEST。

延迟相对寻址：OPR@A，DEST。

2.指令能访问全字或一个字节(B)，因此，全字的地址必为偶数地址。

3.子程序连接指令JSR和RTS，以及一些特殊指令将另作说明。

程序设计举例

本例说明数据表(频数图)的形成，该数据表表示另一数据表中1—100以内的每个数值出现的频率。1—100以外的数值忽略不计。其程序见表12.2，流程图见12.5。整个程序分为六个部分，即a～f。

(a)启动清除输出表的循环。指令MOVE #OTABLE，RO把地址OTABLE放入寄存器RO，指令MOVE#－100，R1把数－100放入寄存器R1。

表12.2

HIST:	MOV#OTABLE,R0	准备清除输出表
	MOV=#-100.,R1	-100放入输出表
CLOOP:	CLR(R0)+	清除下一个表目
	INC R1	检查是否做完
	BNE CLOOP	若未完，继续清除
	MOV#ITABLE,R0	建立输入指示器
	MOV#-1000.,R1	数据表长
	MOV#100.,R2	最大输入值
HLOOP:	MOV(R0)+,R4	取下一个输入值
	BLE NOCOUNT	若小于或等于0，则忽略
	CMP R4,R2	对照最大值检查
	BGT NOCOUNT	若大于则忽略
	ASL R4	每个表目2个字节
	INC OTABLE (R4)	增加适当的部分
NOCOUNT:	INC R1	输入完毕了没有?
	BNE HLOOP	若未完，继续扫描
	HALT	频数图完成

(b)指令CLR(R0)+，清除第一个存贮单元OTABLE，并使寄存器R0的内容增加为OTABLE+1(为下一个表目作准备)。

(c)指令INCR1，BNE CLOOP执行计数，直至R1的内容从初始值-100变到0。

(d)指令MOV#ITABLE，R0把地址ITABLE放入寄存器R0。再用两条指令把-1000推放入寄存器R1，100放入寄存器R2。这样就启动了分析操作。

(e)此后四条指令从ITABLE取出一项，并检查它是否在0—100的范围内。若在此范围内，则指令ASLR4用2(字地址必为偶数地址)去乘这一项的值，指令INC OTABLE(R4)使有效地址OTABLE+2X(ITABLE+i)加1，其中i表示所分析的是ITABLE的第i项。

(f)最后三条指令为循环计数，且在工作完成时用它们来中止程序。

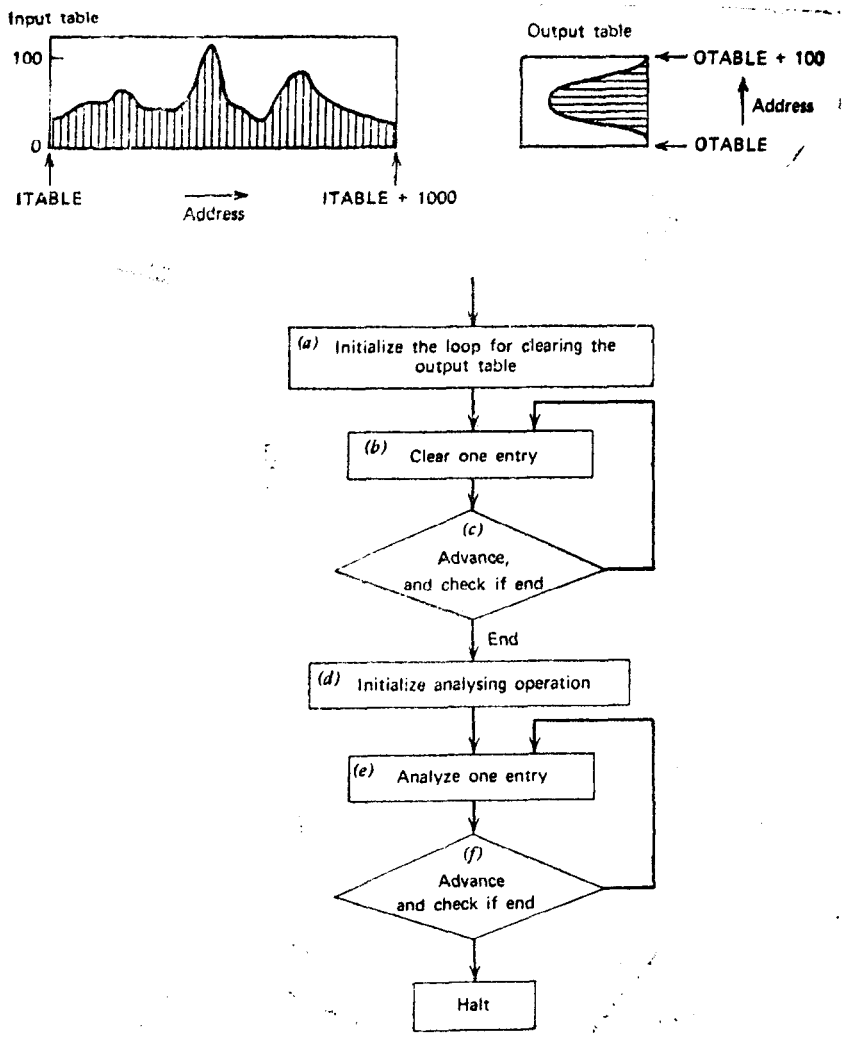

图12.5　形成频数图的流程图

12.4　堆栈、子程序及中断

堆栈

堆栈是一种单端存取的动态数据顺序表，也称下推表或后进先出表。对堆栈的存贮与检索分别称为下推(↓)(或打入)与退出(↑)。其操作过程见图12.6。

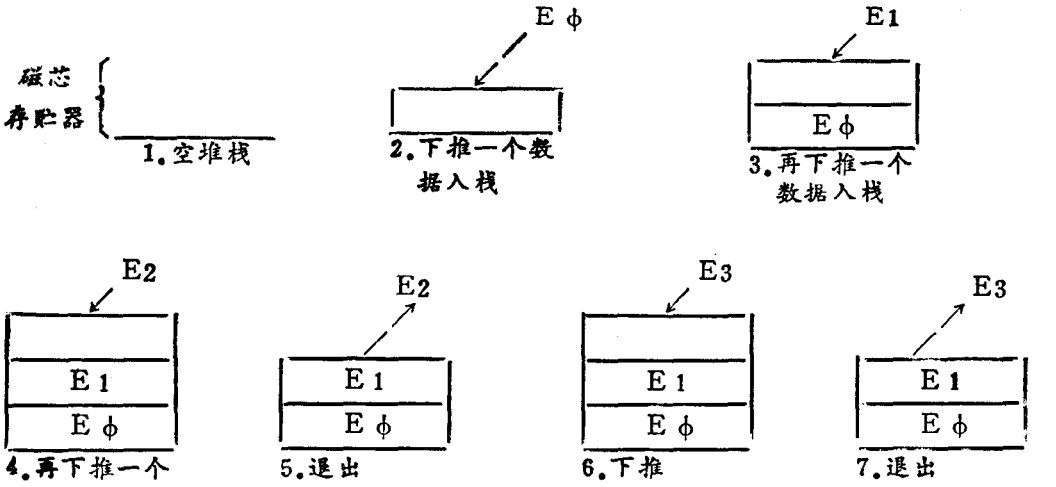

图12.6　堆栈的下推与退出图解

堆栈只在栈顶有一入口。入口地址存放在通用寄存器R6，R6用作堆栈指示器(SP)。利用SP可把信息存入堆栈或取出堆栈。

例、把寄存器R0—R4保存到堆栈。其程序段见表12.3，此程序将把寄存器R0—R4下推入栈，栈顶为R0。

表12.3

SAVE:	MOV R4,—(SP)	R5已由JSR推入堆栈。
	MOV R3,—(SP)	R5将处于栈底，
	MOV R2,—(SP)	R5上面依次为
	MOV R1,—(SP)	R4、R3、R2、R1、R0.
	MOV R0,—(SP)	R0在栈顶。

例、从堆栈恢复R0—R4。其程序段见图12.4。注意，第一例的SP每次都自动减1，从而指出堆栈中较高一级的地址。而在本例中，SP每次都自动加1，指出一个与栈底逐步接近的地址。

表12.4

	MOV (SP)+,R0	各寄存器按其被推
	MOV (SP)+,R1	入栈时相反的次序
	MOV (SP)+,R2	恢复。
	MOV (SP)+,R3	
	MOV (SP)+,R4	

使用堆栈的方法有三种：

- 通过程序设计(如例12.3与12.4)。
- 使用子程序指令JSR及RTS，即自动使用堆栈。
- 若发生中断，则自动使用堆栈。

子程序与堆栈

子程序调用指令为JSR reg,dst。在该指令执行过程中，计算机硬件将自动执行下列操作：

dst→(tmp)	其中tmp为内部处理器寄存器。
reg↓	寄存器内容推入处理器堆栈。
(PC)→(reg)	PC内容为紧接JSR之后的存贮地址；该地址现在被存入寄存器，用作程序返回地址。
(tmp)→(PC)	在目标地址处跳变到子程序。

注意，寄存器是用来存贮返回地址的。该寄存器先前的内容被自动地保存入堆栈。

由子程序指令RTS reg实现返回，操作如下：

(reg)→(PC)	使地址返回到程序计数器。
↑(reg)	寄存器原先的内容从堆栈退出，因而得以恢复。

表12.5是一个16—位无符号整数倍乘子程序的例子。第一条指令使程序跳到子程序。该指令之后紧接着三个存贮单元，分别存放被乘数、乘数及乘积的地址。寄存器R5在本例中用来保存返回地址。

表12.5

```
        JSR R5, MULT
        .WORD MCAND      被乘数地址
        .WORD MPLIER     乘数地址
        .WORD PROD       乘积地址
MOLT:   CLR R0
        MOV@(R5)+, R1    取乘数放入R1
        MOV@(R5)+, R2    取被乘数放入R2
        MOV#—16.,R3      计数器置位
MLOOP:  ASL R0           双倍精度移位
        ROL R1           移位并重复地加
        BCC NOADD        最高位控制加法操作
        ADD R2, R0       若置位，则加被乘数
        ADC R1           保存32－位乘积
NOADD:  INC R3           完成了没有?
        BNE MLOOP        若未完则继续
        MOV (R5)+, R2    取存放乘积的地址
        MOV R0(R2)+      丢开低位，移向高位
        MOV R1, (R2)     丢开高位
        RTS R5           返回调用程序
```

注意，子程序堆栈操作的下列重要特点：

子程序调用为实现子程序自动嵌套、重新进入及多重入口点创造了条件。子程序可以再调用自力的子程序(或者就是其本身)，嵌套级数不限，无需为各级子程序调用专门准备返回地址的存贮。子程序调用过程不会改变存贮器中的固定存贮单元，因此子程序调用也可用来实现重入，这就使一套子程序存贮可供多个中断处理共用。

中断与堆栈

本机器有四个中断优先级。优先级较高的请求可使优先级较低的中断服务过程中断下来，而在完成了对优先级较高的服务之后，再使控制自动返回优先级较低的中断服务程序。

每当发生中断时，处理器就自动地先把现行中央处理器状态(PS)、然后再把现行程序计数器(PC)推入堆栈。

每一中断程序都是以中断指令RT1的返回结束的 。指令RT1执行的操作如下：

SP↑(PC)	程序计数器由堆栈恢复
SP↑(PC)	处理器状态由堆栈恢复

图12.7是一个例子，说明堆栈在嵌套的中断服务中以及数据暂存中的应用。注意，SP所指示的总是栈顶的字。

12.5 单一总线接口技术

单一总线中的线

单一总线是PDP—11的唯一总线，它连接全部外围设备，存贮器，以及中央处理器。处理器与所有存贮器及设备进行信息交换时都使用同一组信号。处理器与外围设备进行信息交换的形式是相同的，这一点很重要。由于这一缘故，用来访问存贮器的同一组程序指令也被用来访问外围设备。设备状态寄存器、设备控制寄存器、以及设备数据寄存器都各有一个唯一的“存贮”地址。例如，指令MOVB RO, PUNCH可从RO中取出一个8—位字符放入穿孔缓冲寄存器。其它指令对穿孔状态进行监控，穿孔操作何时完结由程序决定。

单一总线由56根信号线组成。包括处理器在内的所有设备都並接在这56根线上，见图12.8。51根信号线的双向特性使信号可向两个方向中的任一方向传送，其余5根单向线用于优先总线控制。56种信号及其功能见表12.6。