**第十三章 自陷例程和中断**

1.**硬件寄存器有特权，不拥有适当特权级别的程序不能访问。**

2.输入输出时使用TRAP指令使操作系统完成，用户程序不必知道实现细节，其他用户程序被保护，避免用户程序员不恰当行为的后果。

3.**服务调用**或**系统调用**：用户程序请求操作系统完成任务。

4.DLX有256个服务例程，用TRAP机制调用。

|  |  |  |
| --- | --- | --- |
| x06 | GETC | 从键盘读取一个字符，将其ASCII码复制到R4[7:0] |
| x07 | OUT | 将R4[7:0]中的字符输出到显示器 |
| x08 | PUTS | 将R4所指的地址开头的一个字符串输出到显示器，每个字符占用一个存储单元，字符串以x00终止 |
| x09 | IN | 输出“Input a character>”到显示器，从键盘读取一个字符后回显到显示器上，并将其ASCII码复制到R4[7:0] |
| x0A | GETS | 两个参数R4（字符串起始地址）和R5（长度n）,从键盘读取n-1个字符，如果输入小于n-1,，则至回车结束，读入缓冲区，并在末尾加上x00 |
| X00 | HALT | 输出“Halting the machine.”，并停止程序 |

5.**TRAP向量表**或**系统控制块**

①存储在**x00000000到x000003FF中**。

②**包括256个服务例程的起始地址**，每个地址占用4个存储单元，每个服务例程占用210个存储单元（数据区占x100，服务例程占x300，共x400）。

③**每个服务例程的数据段起始地址均为其代码段起始地址前x100个单元的位置。**

④部分TRAP向量表

|  |  |  |  |
| --- | --- | --- | --- |
| TRAP向量 | 向量表地址 | 服务例程地址 | 数据区地址 |
| x06 GETC | x0000 0018 | x0000 2500 | x0000 2400 |
| x07 OUT | x0000 001C | x0000 2900 | x0000 2800 |
| x08 PUTS | x0000 0020 | x0000 2D00 | x0000 2C00 |
| x09 IN | x0000 0024 | x0000 3100 | x0000 3000 |
| x0A GETS | x0000 0028 | x0000 3500 | x0000 3400 |
| x00 HALT | x0000 0000 | xFFFE 0100 | xFFFE 0000 |

6.TRAP指令执行步骤

①26位TRAP向量符号扩展到32位，再左移2位（乘以4）形成地址，该地址加载到MAR中。

②加载到MAR中的地址中的记录被读取，并将其加载到MDR中。

③通用寄存器R31被加载为PC当前内容（已经被加4）。

④MDR内容被加载到PC中，并完成指令的执行。

⑤返回用户程序。

7.返回用户程序时使用指令**JR R31**，用助记符**RET**表示，**调用TRAP指令后R31中的数据被破坏**。

8.**MCR（机器控制寄存器）的[0]位存储运行锁。**

9.**HALT服务例程**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01 |  | .data | xFFFE0000 |  |
| 02 | SaveR1: | .space | 4 | ;保存寄存器的存储单元 |
| 03 | SaveR2: | .space | 4 |  |
| 04 | SaveR4: | .space | 4 |  |
| 05 | SaveR31: | .space | 4 |  |
| 06 | Newline: | .byte | x0A |  |
| 07 | Message: | .asciiz | “Halting the machine. ” | |
| 08 |  | .align | 2 |  |
| 09 | MCR: | .word | xFFFF00F8 |  |
| 0A | ; |  |  |  |
| 0B |  | .text | xFFFE0100 |  |
| 0C |  | sw | SaveR1(r0),r1 | ;保存例程所需寄存器 |
| 0D |  | sw | SaveR2(r0),r1 |  |
| 0E |  | sw | SaveR4(r0),r1 |  |
| 0F |  | sw | SaveR31(r0),r31 |  |
| 10 | ; |  |  |  |
| 11 | ;输出停机消息 | | | |
| 12 | ; |  |  |  |
| 13 |  | lb | r4,Newline(r0) |  |
| 14 |  | trap | x07 |  |
| 15 |  | addi | r4,r0,Message |  |
| 16 |  | trap | x08 |  |
| 17 |  | lb | r4,Newline(r0) |  |
| 18 |  | trap | x07 |  |
| 19 | ; |  |  |  |
| 1A | ;清空xFFFF 00F8的0位，停机 | | | |
| 1B | ; |  |  |  |
| 1C |  | lw | r1,MCR(r0) |  |
| 1D |  | lw | r2,0(r1) | ;加载MCR值到R2中 |
| 1E |  | andi | r2,r2,#-2 | ;清空MCR[0] |
| 1F |  | sw | 0(r1),r2 | ;将R2值存储到MCR中 |
| 20 | ;**至此已经停机，若未重置以下指令不会执行** | | | |
| 21 | ;从HALT例程返回 | | | |
| 22 | ; |  |  |  |
| 23 |  | lw | r1,SaveR1(r0) | ;将寄存器恢复为原先的值 |
| 24 |  | lw | r2,SaveR2(r0) |  |
| 25 |  | lw | r4,SaveR4(r0) |  |
| 26 |  | lw | r31,SaveR31(r0) |  |
| 27 |  | jr | r31 | ;从TRAP返回 |

10.**HALT指令使用了R1、R2、R4、R31且停机时数据被破坏。**

11.**PUTS服务例程**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01 |  | .data | x00002C00 |  |
| 02 | SaveR2: | .space | 4 | ;保存寄存器的存储单元 |
| 03 | SaveR3: | .space | 4 |  |
| 04 | SaveR4: | .space | 4 |  |
| 05 | SaveR5: | .space | 4 |  |
| 06 | DSR: | .word | xFFFF0008 |  |
| 07 | DDR: | .word | xFFFF000C |  |
| 08 | ; |  |  |  |
| 09 | ; |  |  |  |
| 0A |  | .text | x00002D00 |  |
| 0B |  | sw | SaveR2(r0),r2 | ;保存此例程所需寄存器 |
| 0C |  | sw | SaveR3(r0),r3 |  |
| 0D |  | sw | SaveR4(r0),r4 |  |
| 0E |  | sw | SaveR5(r0),r5 |  |
| 0F | ; |  |  |  |
| 10 | ;对字符串中的每一个字符进行循环 | | | |
| 11 | ; |  |  |  |
| 12 | LOOP: | lb | r2,0(r4) | ;取得字符 |
| 13 |  | beqz | r2,Return | ;如果是0，字符串结束 |
| 14 |  | lw | r5,DSR(r0) |  |
| 15 | L2: | lw | r3,0(r5) | ;测试输出寄存器是否就绪 |
| 16 |  | andi | r3,r3,#1 |  |
| 17 |  | beqz | r3,L2 | ;循环直到显示器就绪 |
| 18 |  | lw | r5,DDR(r0) | ; |
| 19 |  | sw | 0(r5),r2 | ;显示字符 |
| 1A |  | addi | r4,r4,#1 | ;指针加1 |
| 1B |  | j | LOOP | ;获取下一个字符 |
| 1C | ; |  |  |  |
| 1D | ;从服务调用请求返回 | | | |
| 1E | Return: | lw | r2,SaveR2(r0) | ;将寄存器恢复为原先的值 |
| 1F |  | lw | r3,SaveR3(r0) |  |
| 20 |  | lw | r4,SaveR4(r0) |  |
| 21 |  | lw | r5,SaveR5(r0) |  |
| 22 |  | jr | r31 | ;从TRAP返回 |

12.**caller-save**（**调用者保存**）：调用程序保存占用的寄存器。

13.**callee-save**（**被调用者保存**）：被调用程序保存占用的寄存器。

14.**中断驱动的I/O**的本质是I/O设备能够

①强制程序停止。

②让处理器执行I/O设备的请求。

③让停止的程序继续执行。

15.某个I/O设备能够生成中断请求，必须具备

①I/O设备需要服务。

②设备有权请求服务。

16.**IE（中断允许位）在设备状态寄存器（KBSR和DSR）的[1]位，1表示允许中断，0表示不允许中断。**

17.**中断请求信号**（**IRQ**）是IE位与就绪位逻辑与运算的结果。

18.**原因寄存器**（**CAUSE**）记录哪些设备发出中断信号，只有特权模式（操作系统）下才能访问。

[15:8]为中断未决位，[15:10]为硬件中断未决位，[9:8]为软件中断未决位。

**CAUSE[11]代表键盘中断未决位，CAUSE[10]代表显示器中断未决位。**

按照优先级高低从左至右排列。

未决中断等到相应SR中断屏蔽位为1时才能引起处理器处理。

19.**SR**（**状态寄存器**）

[0]位可以改写所有设备的IE位，只有特权模式（操作系统）下才能访问。

[1]位表示正在运行的程序处于特权（管理员或内核）模式时为0，处于用户模式时为1。

[2]位中断发生时保存SR[0]的值。

[3]位中断发生时保存SR[1]的值。

[15:8]位是**中断屏蔽位**（**中断掩码位**），给出**中断阻塞方案**，[15:10]为硬件中断屏蔽位，[9:8]为软件中断屏蔽位。优先级高低从左至右排列。1表示允许中断，0表示屏蔽中断。

20.**程序状态**：程序影响的所有资源所包含的内容的瞬态图。

21.**EPC**：保存中断发生时PC的值，只有特权模式（操作系统）下才能访问。

22.进入中断服务例程时，应屏蔽所有中断，原因：避免受到来自其他设备的中断信号干扰。

23.**优先级**：执行的紧急程度。**请求的优先级必须高于希望中断的程序**。

24.DLX硬件优先级从低到高分为PL0、PL1、PL2、PL3、PL4、PL5，速度越高的I/O设备优先级越高。

25.**键盘优先级为1，显示器优先级为0。**

26.**中断服务例程**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 01 |  | .data | x80000000 |  |
| 02 | SaveR1: | .space | 4 | ;保存寄存器的存储单元 |
| 03 | SaveR2: | .space | 4 |  |
| 04 | SaveR3: | .space | 4 |  |
| 05 | SaveR5: | .space | 4 |  |
| 06 | SaveR6: | .space | 4 |  |
| 07 | SaveR7: | .space | 4 |  |
| 08 | KBDR: | .word | xFFFF0004 |  |
| 09 | DDR: | .word | xFFFF000C |  |
| 0A | ; |  |  |  |
| 0B |  | .text | x80001000 |  |
| 0C | ;保存中断服务例程需要的寄存器 | | | |
| 0D |  | sw | SaveR1(r0),r1 |  |
| 0E |  | sw | SaveR2(r0),r2 |  |
| 0F |  | sw | SaveR3(r0),r3 |  |
| 10 |  | sw | SaveR5(r0),r5 |  |
| 11 |  | sw | SaveR6(r0),r6 |  |
| 12 |  | sw | SaveR7(r0),r7 |  |
| 13 | ;是否有允许的中断 | | | |
| 14 |  | movs2i | r1,x0D | ;CAUSE |
| 15 |  | movs2i | r2,x0C | ;SR |
| 16 |  | movs2i | r6,x0E | ;EPC |
| 17 |  | andi | r3,r1,xFF00 | ;CAUSE[15:8] |
| 18 |  | andi | r5,r2,xFF00 | ;SR[15:8] |
| 19 |  | and | r3,r3,r5 |  |
| 1A |  | beqz | r3,DONE | ;没有允许的中断，返回 |
| 1B | ;按照优先级顺序，依次处理 | | | |
| 1C | TEST5: | slli | r3,r3,#16 |  |
| 1D |  | slti | r5,r3,#0 |  |
| 1E |  | addi | r7,r0,TEST4 |  |
| 1F |  | bnez | r5,DEV5 | ;优先级为PL5的设备 |
| 20 | TEST4: | slli | r3,r3,#1 |  |
| 21 |  | slti | r5,r3,#0 |  |
| 22 |  | addi | r7,r0,TEST3 |  |
| 23 |  | bnez | r5,DEV4 | ;优先级为PL4的设备 |
| 24 | TEST3: | slli | r3,r3,#1 |  |
| 25 |  | slti | r5,r3,#0 |  |
| 26 |  | addi | r7,r0,TEST2 |  |
| 27 |  | bnez | r5,DEV3 | ;优先级为PL3的设备 |
| 28 | TEST2: | slli | r3,r3,#1 |  |
| 29 |  | slti | r5,r3,#0 |  |
| 2A |  | addi | r7,r0,TEST1 |  |
| 2B |  | bnez | r5,DEV2 | ;优先级为PL2的设备 |
| 2C | TEST1: | slli | r3,r3,#1 |  |
| 2D |  | slti | r5,r3,#0 |  |
| 2E |  | addi | r7,r0,TEST0 |  |
| 2F |  | bnez | r5,DEV1 | ;优先级为PL1的键盘 |
| 30 | TEST0: | slli | r3,r3,#1 |  |
| 31 |  | slti | r5,r3,#0 |  |
| 32 |  | bnez | r5,DEV0 | ;优先级为PL0的显示器 |
| 33 | ;…（软件中断测试） | | | |
| 34 |  | j | DONE |  |
| 35 | ;处理中断 | | | |
| 36 | ;…（其它设备） | | | |
| 37 | DEV1: | lw | r1,KBDR(r0) |  |
| 38 |  | lw | r4,0(r1) | ;输入字符加载到R4 |
| 39 |  | jr | r7 |  |
| 3A | DEV0: | lw | r1,DDR(r0) |  |
| 3B |  | sw | 0(r1),r4 | ;R4中的字符输出到显示器 |
| 3C | ;将寄存器恢复为原先的值 | | | |
| 3D | DONE: | lw | r1,SaveR1(r0) |  |
| 3E |  | lw | r2,SaveR2(r0) |  |
| 3F |  | lw | r3,SaveR3(r0) |  |
| 40 |  | lw | r5,SaveR5(r0) |  |
| 41 |  | lw | r6,SaveR6(r0) |  |
| 42 |  | lw | r7,SaveR7(r0) |  |
| 43 |  | movi2s | r0,x0D | ;清空CAUSE |
| 44 |  | rfe |  |  |

27.**中断嵌套**：中断服务例程执行时允许被比其优先级高的设备中断。

①保存SR和EPC。

②修改SR[15:8]，屏蔽比该设备优先级低（或相等）的其他设备的中断，允许优先级高的设备中断。

③将SR[0]设为1，允许中断。

④结束中断前设SR[0]为0（恢复SR和EPC时不允许被中断）

⑤恢复SR和EPC。

28.中断嵌套时保存程序状态使用栈。

29.C语言程序输入输出通过I/O库函数执行。

30.所有基于字符的输入和输出都是对**流**执行的。

31.当一个字符被键入，它被添加到输入流的结尾处。读取键盘的输入，总是从输入流开头处读取。

32.程序要打印的ASCII码字符序列被添加到输出流结尾处，输出时总是从输出流开头处输出，**每次流中的一个字符使用后被消耗，未被使用的字符将保留在流中**。

33.使用流使输入和输出以其各自速率操作而不用等待另一个就绪。

34.**stdin**（**标准输入流**）缺省映射到键盘，**stdout**（**标准输出流**）缺省映射到显示器。

35.**putchar()**输出一个字符，**getchar()**输入一个字符（**从输入流中读取一个字符，当输入流为空时等待**）。二者必须包含stdio.h头文件。

36.**I/O流缓冲**：每个键盘上的输入都被底层操作系统软件捕获，并被保存在一个小的数组缓冲区里，**直到用户按下回车键，缓冲区才被释放到输入流中，且回车键本身也作为换行字符加入输入流**。

优点：使用户能使用退格键删除以便编辑输入的内容，并按下回车键确认其输入。

37.**使用printf输出%使用序列”%%”。**

38.**空白字符**：空格、水平制表符、新行、回车、垂直制表符、换页。

39.**scanf输入时抛弃所有空白字符（但不会连接空白字符两边的内容），并依次匹配符合的格式，一旦遇到不符的格式即终止输入，以回车确认输入。此时未匹配的变量未被赋值而保持初值。**

40.scanf函数返回成功转换的参数个数，printf函数返回输出的字符串字符数（转义序列和格式说明计一个，不包括’\0’），putchar函数和getchar函数分别返回输出和输入的字符ACSCII码值。