

第 6 次作业

作业文件命名格式：专业班级_学号_姓名_第几次作业.pdf

例如：CS2201_U202212345_张三_6.pdf

提交到 qq 邮箱：2112745268@qq.com

提交截止时间：2024.04.26

1. 什么是中断矢量表？什么是中断描述符表？如何利用中断描述符表找到中断服务程序的地址？

答案：

存放每个中断源的中断处理程序的表称为中断向量表。在实模式下，中断矢量表中每个表项 4 个字节，分别存放中断处理程序的偏移地址和段地址。在保护模式下，中断矢量表被称为中断描述符表，每个表项（称作门描述符）8 个字节，存放中断处理程序的入口地址以及类别、权限等信息。根据门描述符中的段选择符找到中断处理程序所在段的段描述符，再根据段描述符找到中断处理程序所在的段，最后结合门描述符中的 32 位偏移地址，得到中断处理程序的入口地址。

2. 假定 PROG_N 为某一中断源的中断处理程序的入口地址（第一条指令的标号），该中断源的类型号为 n ，试编写实方式下的程序段，将该入口地址填入中断向量表中的相应位置。

答案：

```
MOV AX, 0
MOV DS, AX
MOV DS:[4*n+0], OFFSET PROGN
MOV DS:[4*n+2], SEG PROGN
```

3. 在实模式下，如何使用 CALL 指令去实现 int 21h 的功能。

答案：

```
MOV AX, 0
MOV DS, AX
PUSHF
CALL DWORD PTR DS:[21H*4]
```

4. Windows 的异常处理机制有哪些？各有什么特点？

答案：

Windows 的异常处理机制有 4 种：

(1) C++ 语言的关键字 try ... catch，只能捕捉/处理用语句 throw 显式抛出的异常（不能捕捉其他异常）。

(2) VC++自定义的语句 `_try ... __except` (不是 C++ 的关键字), 可以捕捉/处理 Windows 所有的异常。

(3) 使用 API 函数 `SetUnhandledExceptionFilter` 注册用于处理异常的筛选器回调函数。

(4) 在结构化异常处理 SEH 链表的表头插入一个新的结点 (新的异常处理函数)。

`try...catch` 和 `_try..._except` 的实现都是基于 Windows SEH 的: 按照 SEH 的规则, 把异常处理程序加入到当前线程的 SEH 异常处理链中。

5. 简述在 Windows SEH 链表的表头插入一个新的结点 (新的异常处理函数) 的实现步骤。

答案:

(1) 安装异常处理函数的签名编制新的异常处理函数 `SHE_handler()`;

(2) 在堆栈中创建一个 SEH 节点并插入到 SEH 链表的头部:

```
DWORD handler = (DWORD)SEH_handler;
_asm { push  handler
        push  FS:[0]          //获取前一结点的地址
        mov   FS:[0], ESP     //安装新的 SEH 链表
    }
```

6. 一个 Windows 异常可以被多个处理程序处理吗? 为什么?

答案:

一个 Windows 异常可以被多个处理程序处理。只要 Windows SEH 链表中结点中的处理函数返回值是 **ExceptionContinueSearch**, 则该异常会继续被 SEH 链表中后面的节点处理、以及被筛选器回调函数处理。

7. 程序填空

在下面的程序段, 测试字符串 STR 中是否存在 '+' 或者 '-' 字符。如存在, 则把 DL 的最高位为 1, 否则置为 0; DL 其他位内容保持不变。

.686P

.model flat, stdcall

ExitProcess proto stdcall :dword

includelib libcmnt.lib

includelib legacy_stdio_definitions.lib

.data

str1 db "ahhjhjsa + bbs -", 0

len equ \$ - str1

.stack 200

.code

main proc c

```

        mov     ecx, len - 1
        mov     esi, offset str1
L0:     mov     al, [esi]
        cmp     al, '+'
        jz      L1
        cmp     al, '-'
        jnz     L2
L1:     or      dl, 80H
        jmp     exit
L2:     inc     esi
        loop    L0
        and     dl, 7FH
exit:   invoke  ExitProcess, 0
main    endp
        end

```

8. 程序填空。变量 char 中定义了一个字符，将该字符的 ASCII 码转换成 16 进制字符串，然后调用 C 函数 printf 显示出来。

```

.686P
.model flat, stdcall
ExitProcess proto stdcall :dword
printf      proto C :vararg
includelib  libcmnt.lib
includelib  legacy_stdio_definitions.lib

.data
char        db      'A'
msg         db      0, 0, 0ah, 0dh, 0

.code
main        proc     c
        mov     al, char
        mov     ebx, 1
L1:         and     al, 0FH
        cmp     al, 10
        jb      L2
        sub     al, 10
        add     al, 'A'
        jmp     L3

```

```

L2:  add    al, '0'
L3:  mov     msg[ebx] , al
      cmp    ebx, 0
      jz     L4
      mov    al, char
      shr    al, 4
      dec    ebx
      jmp    L1
L4:  invoke  printf, offset msg
      invoke ExitProcess, 0
main  endp
      end

```

9. 阅读程序，回答问题。

.686P

```

.model      flat, stdcall
ExitProcess proto stdcall :dword
includelib  kernel32.lib          ;;ExitProcess()
includelib  libcmnt.lib           ;;_mainCRTStartup => main
.data
string      db      "12ABCKJHaaabjufdsalb47"
len         equ     $ - string    ;;$表示当前位置偏移地址
.stack      200
.code
main        proc     c
            lea      esi, string
            mov      ecx, len      ; ①
next:       mov      al, [esi]     ; ②
            cmp      al, 'A'
            jb       L1
            cmp      al, 'Z'
            ja       L1
            sub      al, 'A' - 'a'
            mov      [esi], al
L1:         inc      esi          ; ③
            loop     next
            invoke   ExitProcess, 0 ;返回 Windows 操作系统
main        endp
            end

```

(1) 上述程序的功能是什么？

答案：将 `string` 中的大写字母转为小写字符，其他字符不变。

(2) 若将语句②处的标号 `next` 上移一行，误写到语句①处，则程序执行结果会怎样？

答案：死循环。

(3) 若漏写了语句③，程序功能会发生什么变化？

答案：如果 `string` 中第一个字符为大写字符，则改为小写字符，其他字符不变。