6.2

PROGON:

START:

XOR AX,AX

MOV DS,AX

CLI;关闭中断

MOV AX,n

MOV WORD PTR DS:[AX\*4],OFFSET PROGON;保存中断处理函数的偏移量

MOV DS:[AX\*4]+2,CS;中断处理函数对应的段地址

STI;打开中断

MOV DX,OFFSET START +14

MOV CL,4

SHR DX,CL

ADD DX,10H

MOV AL,0

MOV AH,31H

INT 21H

**附加题：**

1. **下面的指令是否有错，有错则改正。**
2. **OUT DX, 60H**

**错误：OPS只能是寄存器**

1. **OUT 0, AX**

**正确**

1. **OUT 260, EAX**

**错误OUT指令中OPD为操作数时，端口地址小于等于255**

1. **OUT 60, EDX**

**正确**

1. **IN AL, 255**

**正确**

1. **IN BX, 60H**

**错误，寄存器只能使用AL//AX/EAX**

1. **IN AX, EDX**

**错误OPD如果要使用寄存器，只能使用DX**

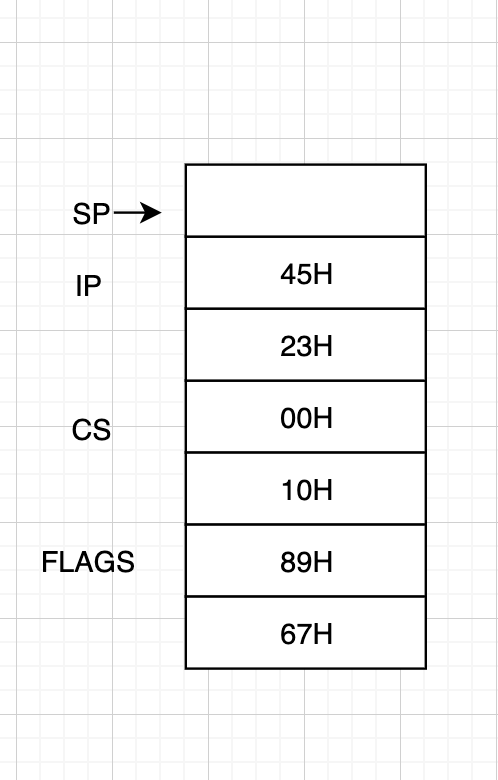
**2 CPU响应外部中断的过程**

1. **CPU在每条指令执行完后，采样中断信号NMI,INTR**
2. **若中断信号有效，**
3. **则关中断（IF=0)**
4. **选取优先级最高的中断源n，并将该中断源的中断信号设置为无效**
5. **执行指令INT n的操作**

**3.**

**进入中断程序后，CS的值为0**

**IP的值为 21H\*4 = 84H**



4.

**CALLINT:**

**XOR AX,AX**

**PUSH DS**

**MOV DS,AX**

**MOV AX,WORD PTR DS:[21H\*4];段偏移值**

**MOV BX,WORD PTR DS:[21H\*4+2] ;段值**

**POP DS**

**MOVZX EAX,AX;**

**MOVZX EBX,BX;求偏移地址**

**SHL EAX,4**

**ADD EAX,EBX**

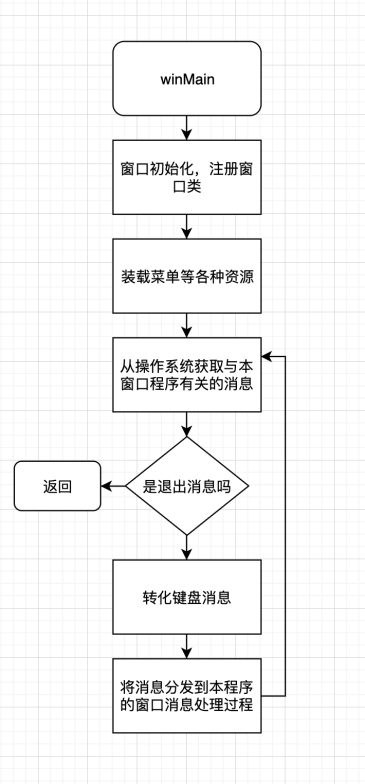
**CALL DWORD PTR EAX**

5.

主程序结构：

开始-> 得到本应用程序的句柄-> 当程序需要从命令行得到参数时，则需要获取命令行的参数地址->调用窗口主程序->结束

窗口主程序：



消息处理程序的作用：

对接收到的消息进行分类判断，以便于分类处理。

**6. 将下面标号 (1) ~ (9) 处的宏汇编指令翻译成汇编（机器）指令。**

STUDENT STRUCT

ID DB 10 DUP (0) ; ;学号

GRADE DB 2 ;年级

STUDENT ENDS

CODE SEGMENT USE16

ASSUME CS:CODE

S1 STUDENT < >

S2 STUDENT < >

SUM PROC A:WORD, B:WORD ;(1)

MOV AX, A ;(2)

ADD AX, B ;(3)

RET ;(4)

SUM ENDP ;(5)

BEGIN: MOV BX, OFFSET S1 ;(6)

MOV AL，S2.GRADE ;(7)

MOV AL, [BX].STUDENT.GRADE ;(8)

MOV AX, 1

INVOKE SUM AX, -2 ;(9)

MOV AX, 4C00H

INT 21H

CODE ENDS

END BEGIN

1. POP A;POP B
2. MOV AX,A
3. MOV AX,B
4. POP IP;
5. MOV BX,0
6. MOV AL,[CS]+22
7. MOV AL,[CS]+10
8. PUSH IP;

MOV DX,-2;

PUSH DX;

PUSH AX;

MOV IP,OFFSET SUM;

**7. 程序填空。下面程序的功能是：从键盘输入一个字符，将该字符的ASCII码按16进制显示出来。例如，若输入字符是‘C’，则显示：THE ASCII OF C IS 43H**

.386

STACK SEGMENT USE16 **\_ STACK \_**

DB 100 DUP(0)

STACK ENDS

DATA SEGMENT USE16

MSG DB ‘THE ASCII OF ?’

MSG\_1 DB ‘ IS ’

MSG\_2 DB ‘41H’, 0AH, 0DH, ‘$’

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:DATA, SS:STACK

BEGIN: MOV AX, DATA

**MOV DX,AX \_**

MOV AH, 1

INT 21H

MOV MSG\_1 - 1,  **AL \_**

MOV BX, 1

L1: AND AL, 0FH

CMP AL, **\_ 0AH \_\_**

JB L2

SUB AL, **\_ 0AH \_\_**

ADD AL, ‘A’

JMP L3

L2:  **ADD AL,’0’ \_\_\_\_**

L3: MOV MSG\_2 [BX], AL

**\_CMP BX,0 \_**

JZ L4

MOV AL, MSG\_1 - 1

SHR AL, 4

DEC BX

JMP L1

L4: MOV AH, 9

LEA DX, **\_ MSG\_2 \_**

**INT 21H \_\_**

MOV AX, 4C00H

INT 21H

JMP L1

CODE ENDS

END **\_ BEGIN \_**