汇编语言程序设计实验指导

【实验提要】 以下列举的10个实验，都是以Intel的8086及后续系列微处理器的指令系统为核心，采用宏汇编工具MASM6.X以及调试工具DEBUG或DEBUG32，针对本教材所述内容进行相关的上机实践。旨在帮助学生加深认识和理解理论教学知识，通过大量的上机实验熟悉8086 CPU的指令功能、用途和使用技巧，进而提高汇编语言程序设计的能力。 （带*号的为选作内容）

实验一 调试工具DEBUG的应用

实验目的

通过实验掌握下列知识:

1、8086指令: MOV,ADD,ADC,SUB,SBB,DAA,XCHG的功能;

2、DEBUG命令: A,D,E,F,H,R,T,U的使用;

3、BCD码、ASCII码及用十六进制数表示二进制码的方法;

4、寄存器: AX,BX,CX,DX,FLAGS,IP。

内容及步骤

注：本次实验可以参照教材上关于DEBUG的叙述内容进行。

一、DEBUG 命令使用

1、开机后,切换到命令提示符窗口下，出现提示符后键入命令DEBUG, 进入调试环境，显示提示符 '-'。

2、用命令 F 200 220 'AB' 将'AB'的两个ASCII码循环填入内存。
注：第一个参数200是当前段的起始偏移地址，第二个参数220是终了偏移地址，第三个参数‘AB’是被填入的数值，若不给出第二个参数则填入128（8行）个字节。

3、用命令 D200 观察内存中的十六进制码及屏幕右边的ASCII字符。

4、用命令 F230 23F 12 重复上二项实验,观察结果并比较。

5、用命令 E200 41 42 43 44 45将A-E的ASCII码写入地址为200开始的内存单元中,再用D命令观察结果,看键入的十六进制数和ASCII码的对应关系。

6、用H命令检查下列各组十六进制数的和与差（补码表示）：

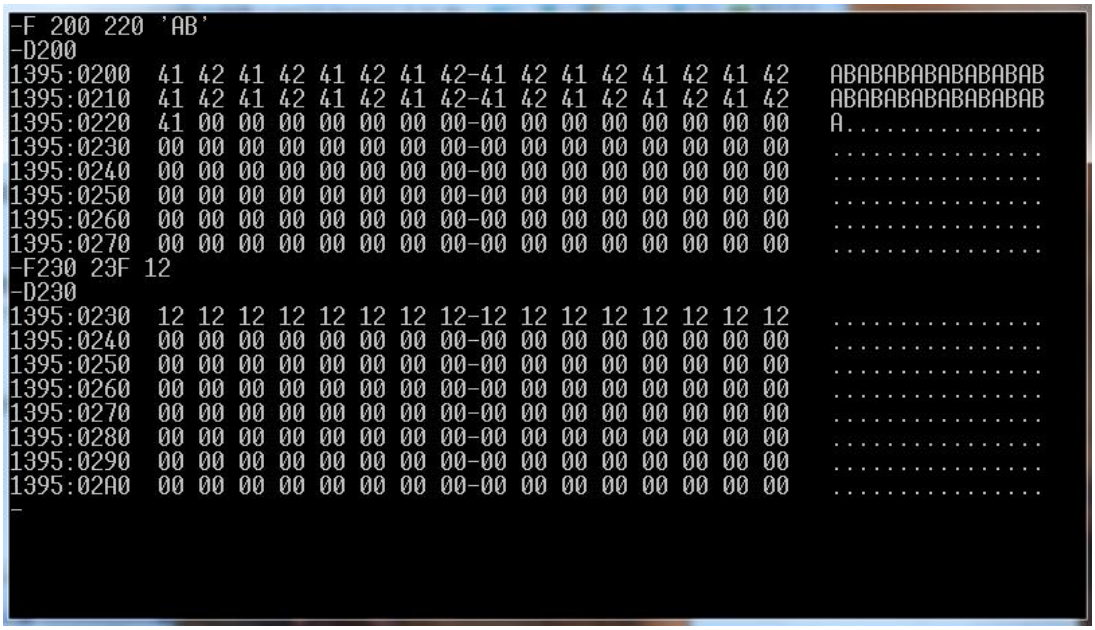
(1)56H,34H　(2)23H,45H　(3)AB,3045H

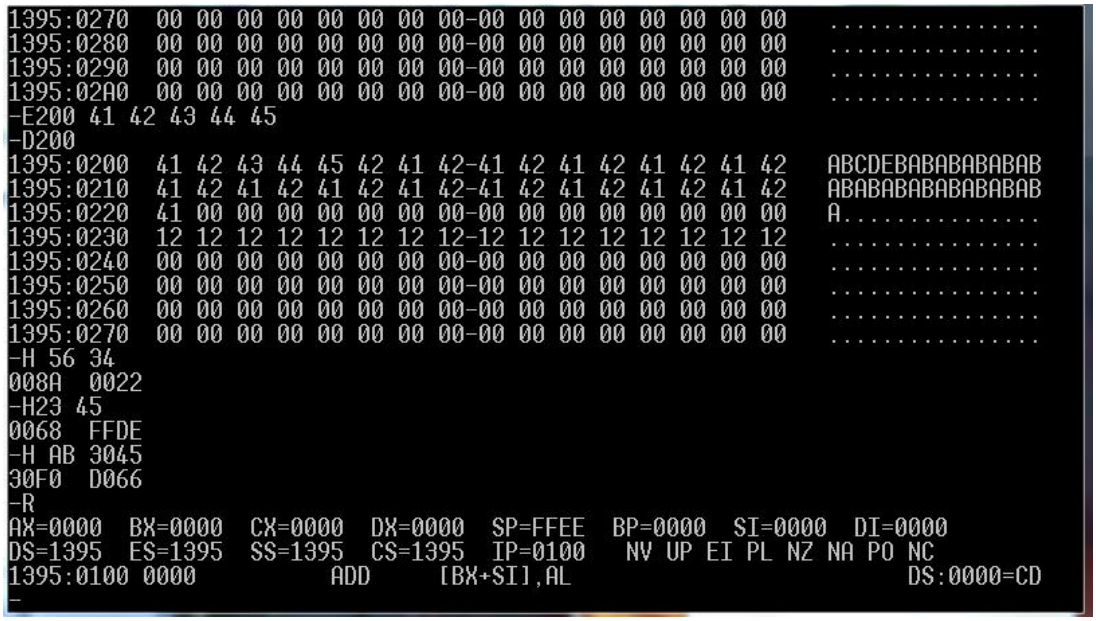
注：输入 H　12 34 则在下一行显示0046 FFDE，即二者的补码和与差。在DEBUG环境下所有数据和地址都是按16进制处理，所以不要加后面的H标志。

7、用R命令检查各寄存器内容,特别注意AX,BX,CX,DX,IP及标志位中ZF,CF和AF的内容。

注：若在DEBUG32环境下用R16和R32命令分别显示16位和32位寄存器内容。

8、用R命令将AX,BX内容改写为1050H及23A8H。





```
1395:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
1395:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
1395:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
1395:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-H 56 34
008A  0022
-H23 45
0068  FFDE
-H AB 3045
30F0  D066
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0100   NV UP EI PL NZ NA PO NC
1395:0100 0000          ADD     [BX+SI],AL                      DS:0000=CD
-R AX
AX 0000
:1050
-R BX
BX 0000
:23A8
-R
AX=1050  BX=23A8  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0100   NV UP EI PL NZ NA PO NC
1395:0100 0000          ADD     [BX+SI],AL                      DS:23A8=00
-
```

二、8086 常用指令练习

1、传送指令

1)用A命令在内存400H处键入下列内容:

```
-A 0400
****:0400    MOV    AX,1234
****:0403    MOV    BX,5678
****:0406    XCHG   AX,BX
****:0408    MOV    AH,10
****:040A    MOV    AL,20
****:040C    MOV    CX,89AB
****:040F    XCHG   AX,CX
****:0411
-
```

注：****为段寄存器CS的当前值，内容是不一定的，每行命令以回车键结束。

2)用U命令检查键入的程序并记录,特别注意左边的机器码与指令的对应关系。

```
-U 0400
```

3)用T命令逐条运行这些指令,每运行一行指令观察各寄存器及IP的变化情况，并注意标志位的变化情况。

```
-T =0400  （注：=400是表示从偏移地址400处开始单步执行）
-T        （不给出地址，则表示接续上一条指令执行）
-T
```

```
1395:0406 XCHG AX,BX
1395:0408 MOV AH,10
1395:040A MOV AL,20
1395:040C MOV CX,89AB
1395:040F XCHG AX,CX\
                ^ Error
1395:040F XCHG AX,CX
1395:0411
-U 0400
1395:0400 B83412        MOV     AX,1234
1395:0403 BB7856        MOV     BX,5678
1395:0406 87C3          XCHG    AX,BX
1395:0408 B410          MOV     AH,10
1395:040A B020          MOV     AL,20
1395:040C B9AB89        MOV     CX,89AB
1395:040F 87C1          XCHG    AX,CX
1395:0411 0000          ADD     [BX+SI],AL
1395:0413 0000          ADD     [BX+SI],AL
1395:0415 0000          ADD     [BX+SI],AL
1395:0417 0000          ADD     [BX+SI],AL
1395:0419 0000          ADD     [BX+SI],AL
1395:041B 0000          ADD     [BX+SI],AL
1395:041D 0000          ADD     [BX+SI],AL
1395:041F 0000          ADD     [BX+SI],AL
-
```

```
AX=5678  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0408   NV UP EI PL NZ NA PO NC
1395:0408 B410          MOV     AH,10
-T

AX=1078  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=040A   NV UP EI PL NZ NA PO NC
1395:040A B020          MOV     AL,20
-T

AX=1020  BX=1234  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=040C   NV UP EI PL NZ NA PO NC
1395:040C B9AB89        MOV     CX,89AB
-T

AX=1020  BX=1234  CX=89AB  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=040F   NV UP EI PL NZ NA PO NC
1395:040F 87C1          XCHG    AX,CX
-T

AX=89AB  BX=1234  CX=1020  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0411   NV UP EI PL NZ NA PO NC
1395:0411 0000          ADD     [BX+SI],AL                      DS:1234=00
-
```

2、加减法指令:

1)用A命令在内存100H处键入下列内容:

```
MOV     AH,11
MOV     AL,22
ADD     AL,AH
```

```
SUB     AL,33
MOV     CX,1234
MOV     DX,5678
ADD     CX,DX
SUB     CX,AX
SUB     CX,CX
```

2)用U命令检查键入的程序及对应的机器码。

3)用T命令逐条运行这些指令，检查并记录有关寄存器及ZF情况。

```
1395:0104 ADD AL,AH
1395:0106 SUB AL,33
1395:0108 MOV CX,1234
1395:010B MOV DX,5678
1395:010E ADD CX,DX
1395:0110 SUB CX,AX
1395:0112 SUB CX,CX
1395:0114
-U 0100
1395:0100 B411          MOV     AH,11
1395:0102 B022          MOV     AL,22
1395:0104 00E0          ADD     AL,AH
1395:0106 2C33          SUB     AL,33
1395:0108 B93412        MOV     CX,1234
1395:010B BA7856        MOV     DX,5678
1395:010E 01D1          ADD     CX,DX
1395:0110 29C1          SUB     CX,AX
1395:0112 29C9          SUB     CX,CX
1395:0114 0000          ADD     [BX+SI],AL
1395:0116 0000          ADD     [BX+SI],AL
1395:0118 0000          ADD     [BX+SI],AL
1395:011A 0000          ADD     [BX+SI],AL
1395:011C 3400          XOR     AL,00
1395:011E 8413          TEST    DL,[BP+DI]
-
```

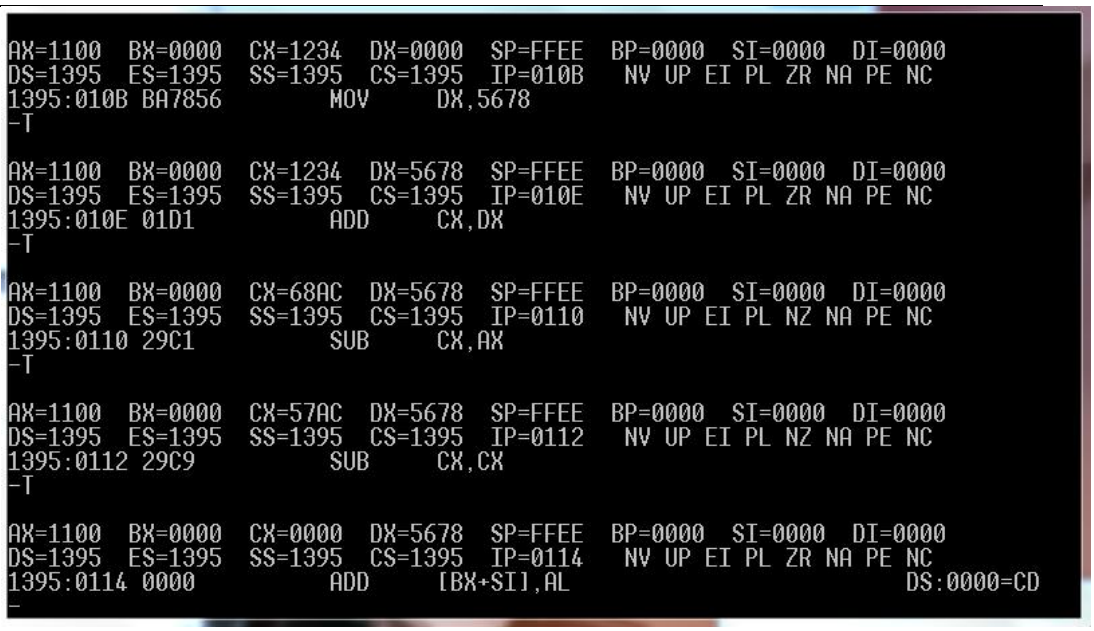
```
AX=1100  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0102   NV UP EI PL NZ NA PO NC
1395:0102 B022          MOV     AL,22
-T

AX=1122  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0104   NV UP EI PL NZ NA PO NC
1395:0104 00E0          ADD     AL,AH
-T

AX=1133  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0106   NV UP EI PL NZ NA PE NC
1395:0106 2C33          SUB     AL,33
-T

AX=1100  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0108   NV UP EI PL ZR NA PE NC
1395:0108 B93412        MOV     CX,1234
-T

AX=1100  BX=0000  CX=1234  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=010B   NV UP EI PL ZR NA PE NC
1395:010B BA7856        MOV     DX,5678
-
```



3、带进位加减法:

1)用A命令在内存200H处键入下列内容,并用U命令检查:

```
MOV     AH,12
MOV     AL,84
MOV     CH,56
MOV     CL,78
ADD     AL,CL
ADC     AH,CH
MOV     DH,A7
MOV     DL,58
SUB     DL,7F
SBB     DH,34
```

2)用T命令逐条运行这些指令，检查并记录有寄存器及CF内容。

```
AX=1200  BX=0000  CX=0000  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0202   NV UP EI PL ZR NA PE NC
1395:0202 B084          MOV     AL,84
-T

AX=1284  BX=0000  CX=0000  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0204   NV UP EI PL ZR NA PE NC
1395:0204 B556          MOV     CH,56
-T

AX=1284  BX=0000  CX=5600  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0206   NV UP EI PL ZR NA PE NC
1395:0206 B178          MOV     CL,78
-T

AX=1284  BX=0000  CX=5678  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0208   NV UP EI PL ZR NA PE NC
1395:0208 00C8          ADD     AL,CL
-T

AX=12FC  BX=0000  CX=5678  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=020A   NV UP EI NG NZ NA PE NC
1395:020A 10EC          ADC     AH,CH
-
```

```
AX=68FC  BX=0000  CX=5678  DX=5678  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=020C   NV UP EI PL NZ NA PO NC
1395:020C B6A7          MOV     DH,A7
-T

AX=68FC  BX=0000  CX=5678  DX=A778  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=020E   NV UP EI PL NZ NA PO NC
1395:020E B258          MOV     DL,58
-T

AX=68FC  BX=0000  CX=5678  DX=A758  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0210   NV UP EI PL NZ NA PO NC
1395:0210 80EA7F        SUB     DL,7F
-T

AX=68FC  BX=0000  CX=5678  DX=A7D9  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0213   NV UP EI NG NZ AC PO CY
1395:0213 80DE34        SBB     DH,34
-T

AX=68FC  BX=0000  CX=5678  DX=72D9  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1395  ES=1395  SS=1395  CS=1395  IP=0216   OV UP EI PL NZ NA PE NC
1395:0216 0000          ADD     [BX+SI],AL                  DS:0000=CD
-
```

3)上面这段程序若改用16位操作指令达到同样结果，怎么改？试修改并运行之。

* 4、BCD码加减法（选作）

1)内容:

```
MOV         AL,65
ADD         AL,16
DAA
```

2)要求:用A 命令键入，U命令检查，T命令逐条运行并记录有关寄存器及AF内容。

注：相加后AL值本来是7B，经DAA指令调整后变为81，即65和16两个BCD码的和。

实验二 内存操作数及寻址方法

实验目的

通过实验掌握下列知识:

1、DEBUG命令:A，T，D，F，G;

2、数据在内存中的存放方式和内存操作数的几种寻址方式;

3、汇编指令:INC，DEC，LOOP，INT 3的应用;

4、汇编语言伪指令:BYTE PTR和WORD PTR的应用。

实验内容和步骤

一、内存操作数及各种寻址方式使用

程序内容:

```
MOV     AX,2000
MOV     [200],AX
MOV     BX,210
MOV     BYTE PTR[BX],50
MOV     CL,40
INC     BX
MOV     [BX],CL
DEC     CL
MOV     SI,5
MOV     [BX+SI],CL
MOV     [BX+SI+1],CL
MOV     WORD PTR[BX+SI+2],1234
```

操作步骤

1)用A命令键入上述程序，并用T命令逐条运行。

2)每运行一条有关内存操作数的指令，要用D命令检查并记录有关内存单元的内容并注意是什么寻址方式。

注：D命令显示结果时，双字节数在内存的存放是高地址对应高数据位;

指令中出现的BYTE PTR及WORD PTR是因为操作数的宽度必须一致。

二、求累加和程序

程序内容:

```
MOV     BX，200
MOV     CX，9
XOR     AX，AX
ADD     AL，[BX]  ；按字节相加
ADC     AH，0     ；若有进位则到AH中
INC     BX
LOOP    108
INT3
```

操作步骤:

1）进入DEBUG环境。

2）用命令F 200 L10 40 在内存200H-20FH地址处填入一系列值40H。

3）用命令A 100 将上述程序键入到100H开始的内存中。

注：LOOP指令用到108H号地址，即为ADD指令的当前地址，构成一个循环。

4）用命令G =100 执行该程序段，程序运行后停在INT 3 指令上，此时观察AX寄存器的值为240H，即为9个40H的和。

注：INT 3指令是一条断点中断指令，程序遇到该指令则停止。

5）用T =100命令单步执行，观察IP、CX及AX寄存器的值，分析程序执行过程。

实验三 数据串传送和查表程序

实验目的

通过实验掌握下列知识:

1、利用简化段定义方法实现程序结构定义;

2、利用DOS的21H号中断调用完成输入输出;

3、查表法和查表指令XLAT;

4、数据串传送指令MOVS及重复前缀REP;

5、掌握EQU和DUP伪指令的用法。

实验内容及步骤

一、利用查表方法显示内存单元的内容

1、编辑下列程序:

```
.model small
.stack
.data
   str1 db 'ABCDEFGHIJ'  ; 待显示的内存区内容
   str2 db 'Please input the number you will display:',10,13,'$'
.code
.startup
  mov ah,9
  mov dx,offset str2
  int 21h            ; 显示STR2字符串的内容，即提示信息
  mov ah,1
  int 21h            ; 输入待显示的字符序号（0-9）
  mov bx,offset str1
  sub al,30h
  xlat               ; 查STR1表，对应序号的字符ASCII码进入AL
  mov dl,al
  mov ah,2
  int 21h            ; 显示对应字符
.exit 0
```

```
End
```

2、程序汇编通过后，在运行过程中输入0-9的任意数字，显示STR1字符串中对应位置的字符。

3、在DEBUG环境中，用P命令调试执行该程序，察看AL寄存器的变化情况及结果的输出，分析其执行过程。

二、数据串传送程序

1、编辑下列程序:

```
.model small
.stack
.data
   str1 db 'abcdefghijklmn'          ; 源串定义
   lengs equ $-str1
   str2 db lengs dup (?),'$'  ; 目标串
.code
.startup
   mov ax,ds
   mov es,ax              ; 使DS和ES为同一个段
   cld
   lea si,str1
   lea di,str2
   mov cx,lengs
   rep movsb              ; 串复制
   mov ah,9
   mov dx,offset str2
   int 21h                ; 显示目标串
.exit 0
End
```

2、程序汇编通过后，运行程序察看输出结果。

3、在DEBUG环境中，用P命令调试执行该程序，察看SI、DI寄存器及相应内存单元的变化情况，分析其执行过程。

4、如果把源串的小写字母复制到目标串的同时再修改为对应的大写字母，该怎么修改程序？

实验附录　ASCII 码表

编码	字符	编码	字符	编码	字符	编码	字符
00	NUL	20	SPACE	40	@	60	`
01	SOH	21	!	41	A	61	a
02	STX	22	“	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	‘	47	G	67	g
08	BS	28	(	48	H	68	h
09	HT	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DEL	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	DEL

实验四 完整段定义程序实现

实验目的

1、掌握完整段格式定义的程序设计方法；

2、掌握伪指令:SEGMENT、ENDS、ASSUME、END、OFFSET、DUP；

3、掌握汇编语言程序中指名程序入口的方法；

4、了解用INT 21H 的4C号功能调用返回系统的方法；

5、掌握用MASM、LINK工具进行汇编和链接的过程。

实验内容

要求程序中实现对键盘输入的字符作如下处理：

- 如果输入的是小写字母则转换成对应的大写字母并显示在屏幕上；
- 如果输入的是大写字母则转换成对应的小写字母并显示在屏幕上；
- 如果输入的是键盘上的Esc键则退出程序执行；
- 按其它任意可显示键，则不作处理，直接显示输出。

实验步骤

1、编辑下列程序:

```
DATA  SEGMENT                          ；数据段定义
          MESSAGE    DB  'Please input your key!',0DH,0AH,'$'
DATA  ENDS
STACK  SEGMENT   PARA STACK 'STACK'；堆栈段定义
        DB 50 DUP(?)
STACK  ENDS
CODE   SEGMENT                          ；代码段
        ASSUME   CS:CODE，DS:DATA，SS:STACK ；分配段寄存器
START:  MOV  AX，DATA
        MOV  DS，AX
        MOV  DX，OFFSET MESSAGE
        MOV  AH，9
        INT  21H      ；显示提示信息
AGAIN:  MOV AH，1
        INT  21H      ；读入一个键盘按键
        CMP  AL,1BH  ；按的是ESC键（ASCII码为1BH）则退出程序
        JE   EXIT
        CMP  AL，61H
        JB   NEXT
```

```
        CMP   AL，7AH
        JA    NEXT
        SUB   AL，20H  ；ASCII码在61H和7AH之间（小写字母）则转换为大写字母
        JMP   DISP
NEXT:   CMP   AL，41H
        JB    DISP
        CMP   AL，5AH
        JA    DISP
        ADD   AL，20H ；ASCII码在41H和5AH之间（大写字母）则转换为小写字母
DISP:   MOV   DL，AL
        MOV   AH，2
        INT   21H      ；显示当前字符
        JMP   AGAIN    ；循环
EXIT:   MOV   AH，4CH ；结束程序
        INT 21H
CODE  ENDS
      END   START      ；指定START标号为程序入口地址
```

2、把上述程序保存为ASM源文件，利用MASM根据对源文件进行汇编,产生.OBJ文件，若汇编时提示有错，编辑工具修改源程序后重新汇编，直至通过。

3、用LINK将.OBJ文件连接成可执行的.EXE文件。

4、在DOS状态下运行LINK产生的.EXE文件，按过键盘之后在屏幕上显示实验要求的字符，按ESC键可返回DOS。

注：汇编过程中若出现错误，可参阅本章最后的实验附录。

5.把上述程序改为简化段定义格式，再汇编，查看执行结果。

实验五 分支结构程序设计

实验目的

1、掌握利用无条件转移指令JMP和有条件转移指令实现多分支程序的设计方法。

2、掌握用DOS的中断调用，实现再程序执行过程中从键盘动态输入字符串的方法。

3、掌握程序中标号的定义和使用。

实验内容

实现从键盘输入一个月份数值（1-12），根据输入的月份数值显示相应的月份英文单词缩写，如输入5则显示May。

实验步骤

1、编辑下列程序:

```
.model small
.stack
.data
   mon db  'Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec'
   msg1 db 'Please input a month(1-12) :',13,10,'$'
   msg2 db 'Input error! Now try again...',13,10,'$'
   buffer label byte  ; 定义输入月份的输入缓冲区
   maxlen db 3        ; 最多 2 个数字，包括一个回车
   actlen db ?        ; 保存实际输入的字符个数
   string db 3 dup(?) ; 保存输入的月份数字内容（ASCII 码）
.code
.startup
    shuru:           ; 开始输入月份编号
   lea dx,msg1
   mov ah,09h
   int 21h           ; 显示提示信息
   lea dx,buffer
   mov ah,0ah
   int 21h           ; 输入月份数值
   cmp actlen,0      ; 若没有输入月份则转出错处理
   je shuruerr
                     ; 以下判断输入月份是否合法
   lea di,string
   cmp actlen,2
   je da10           ; 若输入的是 2 位数月份值则转到 da10 标号处执行
   mov al，string    ; 若只输入 1 位数字月份值，则读出该值
   and al，0fh       ; 把 ASCII 码转换为对应数字
```

```
    jmp jisuan
da10:
    mov al，string
    and al，0fh       ；把月份数值十位的 ASCII 码转换为对应数字(如 12 月的 1 字)
    mov bl,10
    mul bl
    and string[1]，0fh ；把月份数值个位的 ASCII 码转换为对应数字
    add al，string[1]  ；十位加上个位 (如 12 月)
jisuan:           ；以下计算偏移地址
    cmp al，1      ；比 1 小是非法月份
    jb shuruerr    ；若月份值小于 1 则转出错处理
    cmp al,12
    ja shuruerr    ；比 12 大也是非法月份
    sub al，1      ；月份值减 1
    shl al，1
    shl al，1      ；月份再乘 4 对应了 MON 字符串中从首地址开始的字符相对位置
    xor ah，ah     ；1 月份从 0 位置开始即 JAN,...5 月份从位置 16 开始即 MAY
    lea si，mon    ；找到被显示月份字符的位置
    add si，ax
    mov cx，3
output: mov dl，[si] ；输出对应月份英文缩写
    mov ah，2
    int 21h
    inc si
    loop output
  .exit 0
shuruerr:  lea dx,msg2  ；输入出错时提示出错，并转到程序起始处重新执行
    mov ah，09h
    int 21h
    jmp shuru
end
```

2、程序汇编通过后，运行程序察看输出结果。

3、程序运行中不输入月份值（直接回车），或输入非法月份值，观察程序执行的情况。

实验六 分支与循环混合结构程序设计

实验目的

1、掌握利用.IF语句实现分支程序的设计方法；

2、掌握利用.REPEAT语句实现循环程序的设计方法；

3、掌握DIV指令的使用特点；

4、掌握地址表的定义和使用方法。

实验内容

实现成绩单的分等级统计功能，在数据段中有成绩单的定义，并包含有若干个分数，程序根据成绩单数据的个数循环分别统计个分数段成绩的个数，最后输出个等级段的分数个数。要求：成绩单数据少于100字节，输出的每个档次统计值按两位数显示。

实验步骤

1、编辑下列程序:

```
.model small
.stack
.data
    HAN DB 42,62,68,70,79,82,85,90,95,99,66,88,78,90,98,200  ; 成绩单
    count equ $-han              ; 记录数据个数
    MEM DB 6 dup(0)              ; 分别用于存放各分数段成绩的个数
    scoree db 10,13,'Score "E" count is:$'  ; 定义各分数段的提示信息
    scored db 10,13,'Score "D" count is:$'
    scorec db 10,13,'Score "C" count is:$'
    scoreb db 10,13,'Score "B" count is:$'
    scorea db 10,13,'Score "A" count is:$'
    scoreo db 10,13,'Score "Other" count is:$'
    table dw scoreo,scoree,scored,scorec,scoreb,scorea ; 定义地址表
.code
.startup
       MOV CX,count
       mov si,0
    begin:                   ; 统计各个分数段的成绩个数
      .if han[si]<60
        add mem[1],1
```

```
    .elseif han[si]>=60 &&  han[si]<=69
      add mem[2],1
    .elseif han[si]>=70 &&  han[si]<=79
      add mem[3],1
    .elseif han[si]>=80 &&  han[si]<=89
      add mem[4],1
    .elseif han[si]>=90 &&  han[si]<=99
      add mem[5],1
    .else
      add mem[0],1
    .endif
    inc si
  loop begin

   mov cx,6
  .repeat                ; 开始循环
      mov si,cx
      dec si
      add si,si          ; 计算地址表中对应地址项与提示信息的对应关系
      mov dx,table[si]
      mov ah,9
      int 21h            ; 显示提示信息
      xor ah，ah
      mov si，cx
      MOV al，mem[si-1]  ; 读出统计好的一个数值（此处只能处理99以下的）
      mov bl,10
      div bl             ; ax/10
      mov mem[si-1],ah   ; AH为余数是个位，暂时存放回该内存单元
      mov dl,AL          ; al为商是十位
      add dl,30h         ; 转换为ASCII码
      MOV AH,2
      INT 21H            ; 以十进制形式显示十位
      MOV AH,2
      mov dl,mem[si-1]
      add dl,30h
      INT 21H            ; 再显示个位
  .untilcxz              ; CX自动减1，减到0结束循环
     .exit 0
  END
```

2、程序汇编通过后，运行程序察看输出结果。

3、修改成绩单的内容再重新汇编和运行程序，观察程序的结果变化情况。

4、利用DEBUG工具的U命令反汇编可执行程序，观察.IF语句及.REPEAT语句所对应的汇编指令情况。

5、利用MASM的命令行，汇编源文件并生成.LST列表文件，查看列表文件内容。

注：实际上高级程序结构控制语句在汇编时，都转换成相应的比较指令和条件转移指令了。高级程序结构控制语句使得程序设计、理解都变得非常方便。

实验七 多重循环及过程的应用

实验目的

1、掌握多重循环程序和排序程序设计方法；

2、掌握带符号数的比较及转移语句应用；

3、伪指令 EQU 及操作符 '$'的使用；

4、掌握过程的定义和调用方法；

5、掌握把内存数据（16位有符号2进制）按照10进制显示输出的方法。

实验内容

完成冒泡法排序程序，利用过程对排好序的数据按 10 进制显示输出

实验步骤

1、输入下列程序，汇编并连接通过。

```
.model small
.stack
.data
    array    dw  12,-66,108,9,5,-123,2000,-900；待排序数据
    count    equ $-array
    pushcount db 0          ；用于临时存放被显示数据的 10 进制总位数
.code
.startup
    mov cx,count
    shr cx,1
```

```
    dec cx
    mov bl,-1
again:                          ; 冒泡法排序外重循环
    mov dx,cx
    and bl,bl
    je  exit
    xor bl,bl
    xor si,si
    again1: mov ax,array[si] ; 内重循环
        .if sword ptr ax>sword ptr array[si+2]
            xchg array[si+2],ax
            mov array[si],ax
            mov bl,-1
        .endif
        inc si
        inc si
        dec dx
        jnz again1
    loop    again
    exit:

    mov cx,count
    shr cx,1
    mov si,0
   .repeat
      mov  ax,array[si]
      call bintodec          ; 显示排好序的数据
      add si,2
   .untilcxz
.exit 0
bintodec proc near            ; 过程定义
        push bx
        push dx
        .if  sword ptr ax<0 ; 对于负数则取其相反数统一按正数处理
             neg ax
             push ax
             mov dl,'-'      ; 显示负号
             mov ah,2
             int 21h
             pop ax
        .endif
         mov bx,10
loop1:   cwd
         div bx
```

```
        push dx             ；把每次除以 10 的余数压入堆栈，并记录数据位个数
        inc pushcount
        cmp ax,0
        jne loop1
print:  pop dx              ；从堆栈弹出每位数据并显示
        add dx,30h
        mov ah,2
        int 21h
        dec pushcount
        jnz print
        mov ah,2            ；显示完一个数据的各个数位后再显示空格进行分隔
        mov dl,' '
        int 21h
        pop dx
        pop bx
        ret
bintodec endp
    END
```

2、汇编并链接通过该程序，观察执行结果；

3、试着把被排序的数据类型改为字节数据，程序该怎么修改？

4、如果要把数据按16进制显示，该怎么修改程序？

实验八 结构的应用及宏程序设计

实验目的

1、掌握结构的定义与使用方法；

2、掌握宏定义及其调用和展开的具体过程；

3、掌握宏的参数和局部标号的使用方法；

4、掌握高级程序流程控制语句中判断是否相等的双等号应用；

实验内容

定义一个保存学生成绩单的结构，并且计算总分，最后按照总分的从高到低顺序显示成绩单，要求名次和成绩及总分都按照10进制显示输出。

实验步骤

1、输入下列程序，汇编并链接通过。

```
.model small
.stack
.data
   student struct                ；结构定义
          snumber dw 0           ；编号，代表名次
          sid db 15 dup('$')     ；学号
          sname db 20 dup('$')   ；最多 20 位，姓名
          score1  dw 0           ；成绩 1
          score2  dw 0           ；成绩 2
          score3  dw 0           ；成绩 3
          ttscore dw 0           ；总分
   student ends

   scoresheet student <1,'computer01$','zhangsan$',90,70,60,>
              student <2,'computer02$','lisi    $',100,0,100,>
              student <3,'computer03$','wanger  $',95,66,88,>
              student <4,'computer04$','zhuwu   $',120,90,99,>
              student <5,'computer05$','john    $',90,80,95,>
                       ；成绩单长度可以任意，这里只列举 5 个纪录
    sheetlength=($-scoresheet)/(type scoresheet) ；成绩单长度，即纪录条数
    pushcount db 0  ；临时存放被显示数据的位数
.code
```

```
    nextline macro  ; 显示换行的宏定义
        mov ah,2
        mov dl,10
        int 21h
        mov ah,2
        mov dl,13
        int 21h
     endm
     dispone macro one    ; 显示一个字符的有参数宏定义
         mov ah,2
         mov dl,one
         int 21h
     endm
 dispscore macro          ; 按照 10 进制显示内存单元数据的宏定义
       local loop1,print  ; 局部标号定义
         push bx
         push dx
         mov bl,10
    loop1: div bl
         mov dl,ah
         mov ah,0
         push dx          ; 把每次除以 10 的余数压入堆栈，并记录数据位个数
         inc pushcount
         cmp al,0
         jne loop1
   print: pop dx          ; 从堆栈弹出每位数据并显示
         add dl,30h
         dispone dl
         dec pushcount
         jnz print
         dispone 20h      ; 显示一个空格
         pop dx
         pop bx
 endm

.startup
  lea bx, scoresheet
  mov cx,sheetlength
  .repeat                        ; 统计每条记录的总分
    mov ax,[bx].student.score1
```

```
    add ax,[bx].student.score2
    add ax,[bx].student.score3
    mov [bx].student.ttscore,ax
    mov [bx].student.snumber, 0 ; 名次初始化为 0
    add bx, type scoresheet
  .untilcxz

   mov dx,0          ; DX 控制总循环次数，初始值为 0
   push dx
begin:
   lea bx, scoresheet
   mov cx, sheetlength
  .repeat
    .if [bx].student.snumber==0     ; 找到第一个未排名的纪录
      mov ax,[bx].student.ttscore   ; 总分记录在 AX 中
      .break
    .else
      add bx, type scoresheet
    .endif
  .untilcxz

    mov bx,offset scoresheet
    mov cx,sheetlength
   .repeat
    .if [bx].student.snumber==0
      .if [bx].student.ttscore>=ax ; 逐一比较找到一个未排序的且总分最高的纪录
         mov ax,[bx].student.ttscore; AX 保存最高的总分
         mov di,bx                  ; 有最高总分的记录地址保存在 DI 寄存器中
      .endif
    .endif
     add bx, type scoresheet
   .untilcxz

    nextline                        ; 在新的一行显示每条记录信息
    mov bx,di
    pop dx
    inc dx
    push dx
    mov [bx].student.snumber,dx     ; DX 加 1 为当前名次
    mov ax,dx                       ; 显示名次
```

```
    dispscore
    lea dx, [bx].student.sid      ; 显示学号
    mov ah,9
    int 21h
    dispone 20h
    lea dx, [bx].student.sname    ; 显示姓名
    mov ah,9
    int 21h
    dispone 20h
    mov ax, [bx].student.score1   ; 显示成绩 1
    dispscore
    mov ax, [bx].student.score2   ; 显示成绩 2
    dispscore
    mov ax, [bx].student.score3   ; 显示成绩 3
    dispscore
    mov ax, [di].student.ttscore  ; 显示总分
    dispscore
    pop dx
    .if dl==sheetlength
       .exit 0
    .endif
    push dx
    jmp begin
  end
```

2、运行程序，可以看到显示结果如下：

```
    1 computer04 zhuwu      120 90 99 309
    2 computer05 john       90 80 95 265
    3 computer03 wanger     95 66 88 249
    4 computer01 zhangsan   90 70 60 220
    5 computer02 lisi       100 0 100 200
```

各记录是按照总分的降序显示的，其中第一列为名次，最后一列为总分。

3、利用 MASM 工具汇编生成列表文件，并察看列表文件内容，注意其中的宏调用和展开过程，注意局部标号的重命名规律。

4、修改结构的定义，在其中再增加两门成绩，并适当修改程序，重新观察运行结果。

实验九 过程调用及模块化程序设计

实验目的

1、掌握过程定义、调用以及参数的传递方法。
2、掌握利用全局变量和堆栈传递参数的区别。
3、掌握过程定义伪指令：PROC、ENDP、NEAR和FAR。
4、掌握指令：CALL、RET、RET n。
5、掌握386指令集的定义方法。
6、掌握模块间的远调用和近调用的区别与实现。
7、掌握模块间变量访问的方法。

实验内容及步骤

一、利用堆栈传递参数的子程序调用

1、输入以下程序

```
.MODEL SMALL
.386                ；指定指令集，可以使用32位寄存器
.STACK
.DATA
    ARRAY1  DW   100,200,30,40,50,60,70,80,9,10
    COUNT1  EQU  ($-ARRAY1)/2
    SUM1    DD   ?
    ARRAY2  DW   10,110,120,103,14,150
    COUNT2  EQU  ($-ARRAY2)/2
    SUM2    DD   ?
.CODE
    .STARTUP
        MOV   AX，COUNT1
        PUSH  AX              ；第一个数组元素数入栈
        LEA   AX，ARRAY1
        PUSH  AX              ；第一个数组首地址入栈
        CALL  FAR PTR SUMS    ；调用求和过程
        MOV   AX，COUNT2
```

```
        PUSH   AX
        LEA    AX，ARRAY2
        PUSH   AX
        CALL   FAR PTR SUMS
    .EXIT 0

SUMS    PROC   FAR
        MOV    BP，SP
        MOV    CX，[BP+6]    ；从堆栈中获得数组元素个数
        MOV    BX，[BP+4]    ；从堆栈中获得数组首地址
        XOR    EAX，EAX
AGAIN:ADD      AX，[BX]      ；循环求数组元素的和
        ADC    EAX，0        ；处理进位
        INC    BX
        INC    BX
        LOOP   AGAIN
        MOV    [BX]，EAX     ；保存数组元素的和
        RET    4             ；返回主程序并堆栈平衡
SUMS    ENDP
        END
```

2、汇编和链接程序，生成可执行程序。

3、用DEBUG调试工具的T命令运行此程序，观察每次过程调用及进出栈指令前后的SP和堆栈内容的变化情况。

3、记录最后结果:SUM1,SUM2的段及偏移地址和它们的内容。

二、模块间的调用和转移

1、分别输入以下三个程序，保存文件名分别为P1.ASM、P2.ASM和P3.ASM。

P1.ASM为程序主模块，内容如下：

```
    EXTRN      SUB1:FAR    ；声明SUB1是外部模块定义的过程，调用方法
                           ；是一个远过程调用
CSEG SEGMENT  PARA  PUBLIC 'CODE'
      ASSUME CS: CSEG
   START: CALL  FAR PTR SUB1 ；主程序只调用了SUB1过程
          MOV   AH,4CH
          INT   21H
CSEG ENDS
```

```
    END     START
```

P2.ASM为子程序从模块，内容如下：

```
    PUBLIC     SUB1        ；声明SUB1是为外部程序调用的
    EXTRN      SUB2:NEAR ；声明SUB2是外部模块定义的过程，调用方法
                           ；是一个近距离过程调用
TEXT SEGMENT  PARA  PUBLIC 'CODE'
      ASSUME CS:TEXT
SUB1 PROC FAR
    MOV        DL,'A'     ；SUB1过程的功能是显示一个‘A’
    MOV        AH,2
    INT        21H
    CALL       SUB2       ；调用SUB2过程
    RET
SUB1      ENDP
TEXT      ENDS
    END
```

P3.ASM为另一个子程序从模块，内容如下：

```
    PUBLIC     SUB2        ；声明SUB2是为外部程序调用的
TEXT SEGMENT PARA  PUBLIC 'CODE'；此代码段与P2.ASM程序的代码段同名，
                               ；链接时可以与P2程序的代码段连成一个段
    ASSUME     CS:TEXT
SUB2 PROC NEAR
    MOV        DL,'B'          ；SUB1过程的功能是显示一个‘B’
    MOV        AH,2
    INT        21H
    RET
SUB2      ENDP
TEXT      ENDS
    END
```

2、在命令提示符窗口下分别执行以下命令，把三个源文件汇编成目标文件：

```
ml /c p1.asm
ml /c p2.asm
ml /c p3.asm
```

其中的 /c 选项是只汇编成OBJ文件，而不链接成EXE文件。

3、在命令提示符窗口下执行link p1.obj+p2.obj+p3.obj命令，把三个目标文件链接成可执行的EXE文件，即P1.EXE。

4、执行P1.EXE文件会显示字母A和B。

5、在DEBUG环境中，用U命令观看并记录远调用和近调用中的代码。

三、模块间的变量传递

1、分别输入以下两个程序，保存文件名分别为L1.ASM和L2.ASM。

L1.ASM 程序是主模块，内容如下：

```
STACK   SEGMENT STACK
        DW  10  DUP（？）
STACK   ENDS
PUBLIC  OPRN1，OPRN2，OPRN3 ；声明三个公共变量可以被外部模块使用
EXTRN   ADDSUB：FAR          ；声明外部模块定义的过程
DATA    SEGMENT
    OPRN1   DW  1234H
    OPRN2   DW  4321H
    OPRN3   DW  ?,?
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS：CODE，DS：DATA，SS：STACK
START：  MOV     AX，DATA
        MOV     DS，AX
        CALL    ADDSUB       ；调用外部过程
        MOV     AH，4CH
        INT     21H
CODE    ENDS
        END     START
```

L2.ASM 是子模块，内容如下：

```
EXTRN   OPRN1：WORD，OPRN2：WORD，OPRN3：WORD
                ；声明本模块中可以引用的公共变量及调用类型
PUBLIC  ADDSUB  ；声明可以被外部模块调用的过程
CODE    SEGMENT
        ASSUME  CS：CODE
ADDSUB  PROC    FAR
        PUSH    BX
        PUSH    AX
        MOV AX，OPRN1
        MOV BX，OPRN2
        ADD AX, BX
        MOV OPRN3，AX       ；存放两个参数的和
```

```
        MOV AX, OPRN1
        SUB AX,BX
        MOV OPRN3[2], AX    ; 存放两个参数的差
        POP     AX
        POP     BX
        RET
ADDSUB  ENDP
CODE    ENDS
        END
```

2、分别汇编以上两个源文件，生成对应的OBJ文件，再链接这两个目标文件得到EXE可执行程序。

3、用DEBUG工具调试此可执行程序，察看文件链接后代码段的内容构成，用T命令单步执行程序，并注意观察运算结果。

*实验十 汇编语言综合实验

实验目的

1、掌握C语言和汇编语言的混合编程方法；

2、了解了解磁盘文件的读写方法；

实验内容及步骤

一、汇编语言和 C 语言混合编程

1、输入以下C语言程序，其功能为演示在中模式下TC程序调用汇编语言子函数的过程，保存的文件名为MAIN.C

```
#include "stdio.h"
extern int subs(int,int); /* 声明subs为外部函数*/
main()
{ int x=0;int y=0;
  scanf("%d%d",&x,&y);
  printf("\n Call subs(%d,%d),the result is:%d ",x,y,subs(x,y));
}
```

2、输入以下汇编语言子程序，其功能为实现两个整数相减，保存程序文件名为SUB.ASM

```
.MODEL MEDIUM
.CODE
  PUBLIC _SUBS
  _SUBS PROC FAR
      PUSH  BP
      MOV BP,SP
      MOV AX,[BP+6]   ; AX是被减数
      SUB AX,[BP+8]   ; 相减
      POP BP          ; 恢复BP寄存器的内容
      RET
  _SUBS ENDP
END
```

3、在DOS提示符下输入如下两条命令：

```
ml  /c sub.asm
; 把SUB.ASM 汇编成SUB.OBJ文件，但不链接（注意大小写）。
tcc -mm -Iinclude -Llib main.c sub.obj
```

；该命令按照中型模式链接程序，直接把MAIN.C编译成目标文件，并和SUB.OBJ链接生成MAIN.EXE程序。

4、运行MAIN.C程序，输入两个整数，观察输出结果。

5、分析在SUB子程序中取得参数为什么用BP+6和BP+8地址获得？

二、磁盘文件读写实验

1、在 D:\建立一个文本文件 F1.TXT，并在文件中输入内容“南京航空航天大学信息科学与技术学院欢迎您!”，也可以输入其它信息。

2、输入以下汇编语言文件，并汇编、链接通过。

```
.model small
.stack
.DATA
   HANDLE    DW ?
   filename1  db 'd:\f1.txt',0
   filename2  db 'd:\f2.txt',0
   temp db 1000 dup (?)
.code
.startup
   lea dx,filename1
   mov al,0
   mov ah,3dh
   int 21h            ; 打开F1.TXT文件
   jc error
   mov handle,ax      ; 文件代号存入HANDLE
   mov ah,3fh
   mov bx,handle
   mov cx,34
   lea dx,temp        ; 读取打开的文件，读字节数在CX中
   int 21h            ; 读取的内容存放在TEMP中
   jc error
   mov bx,handle
   mov ah,3eh         ; 关闭该文件
   int 21h

   mov ah,3ch
   mov cx,0
   lea dx,filename2
   int 21h            ; 建立F2.TXT文件
   jc error
   mov handle,ax
```

```
        mov ah,40h
        mov bx,handle
        mov cx,34          ；把TEMP中的内容写入到当前文件中
        lea dx,temp
        int 21h
        mov bx,handle      ；关闭该文件
        mov ah,3eh
        int 21h
        error:
        .exit 0
    end
```

3、执行该程序，察看 d:\f2.txt 文件的内容

4、请修改程序，使得在运行时输入具体内容，并把输入的内容存入 F3.txt 文件中。

三、打印 ASCII 码表实验

1、输入以下程序，并汇编、链接通过。

```
.model small
.stack
.code
.startup
        mov bl,0h
        mov bh,0h
outer1: add bl,bh       ；打印特殊字符（控制字符）
        cmp bl,0ah
        je cout_break
        cmp bl,0dh
        je cout_break
        cmp bl,07h
        je cout_break
        cmp bl,08h
        je cout_break
        cmp bl,09h
        jne outer2
cout_break:mov dl,20h   ；显示空格
        mov ah,2
        int 21h
        jmp outer3
outer2: mov dl,bl       ；打印可见字符
        mov ah,2
        int 21h
```

```
outer3: mov dl,20h     ; 输出空格
        mov ah,2
        int 21h
        add bl,10h     ; 改变列循环变量
        sub bl,bh
        cmp bl,0f0h    ; 比较列循环变量与转移条件的值
        jne outer1
        mov bl,0f0h    ; 输出每一行的最后一个字符
        add bl,bh
        mov dl,bl
        mov ah,2
        int 21h
outer4: mov dl,0ah     ; 输出回车换行
        mov ah,2
        int 21h
        mov dl,0dh
        mov ah,2
        int 21h
        inc bh         ; 改变行循环变量
        mov bl,0h      ; 列循环变量置零
        cmp bh,0fh     ; 比较行循环变量与转移条件的值
        jle outer1
    .exit 0
      end
```

2、运行程序，观察显示结果，将看到 ASCII 码表，请与实验三的附录作比较，看有什么不同？

3、请仔细分析程序内容，读懂每行语句的功能

实验附录　汇编语言常用出错信息

汇编程序在对源程序的汇编过程中,若检查出某语句有语法错误,随时在屏幕上给出出错信息，以便操作人员即时查找错误，给予更正。MASM 出错信息格式如下：

错误编号	错误描述
0	Block nesting error 嵌套出错。嵌套的过程、段、结构、宏指令或重复块等非正常结束。例如在嵌套语句中有外层的结束语句，而无内层的结束语局
1	Extra characters on line 一语句行有多余字符，可能是语句中给出的参数太多
2	Internal error-Register already defined 这是一个内部错误。如出现该错误，请记下发生错误的条件，并使用 Product Assistance Request 表与 Microsoft 公司联系
3	Unkown type specifer 未知的类型说明符。例如类型字符拼错，把 BYTE 写成 BIT，NEAR 写成 NAER 等
4	Redefinition of symbol 符号重定义。同一标识符在两个位置上定义。在汇编第一遍扫描时，在这个标识符的第二个定义位置上给出这个错误
5	Symbol is multidefined 符号多重定义。同一标识符在两个位置上定义。在汇编第二遍扫描时，每当遇到这个标识符都给出这个错误
6	Phase error between passes 两次扫描间的遍错。一个标号在二次扫描时得到不同的地址值，就会给出这种错误。若在启动 MASM 时使用/D 任选项，产生第一遍扫描的列表文件，它可帮助你查找这种错误
7	Already had ELSE clause 已有 ELSE 语句。在一个条件块里使用多于一个的 ELSE 语句
8	Must be in conditional block 没有在条件块里。通常是有 ENDIF 或 ELSE 语句，而无 IF 语句
9	Symbol not defined 符号未定义，在程序中引用了未定义的标识符
10	Syntax error 语法错误。不是汇编程序所能识别的一个语句
11	Type illegal in context 指定非法类型。例如对一个过程指定 BYTE 类型，而不是 NEAR 或 FAR
12	Group name must be unique 组名应是唯一的。作为组名的符号作为其他符号使用
13	Must be declared during pass 1 必须在第一遍扫描期间定义。在第一遍扫描期间，如一个符号在未定义前就引用，就会出现这种错误。
14	Illegal public declaration 一个标识符被非法的指定为 PUBLIC 类型
15	Symbol already defferent kind 重新定义一个符号为不同种类符号。例如一个段名重新被当作变量名定义使用
16	Reserved word used as symbol

	把汇编语言规定的保留字作标识符使用
17	Forward reference illegal 非法的向前引用。在第一遍扫描期间，引用一个未定义符号。
18	Operand must be register 操作数位置上应是寄存器，但出现了标识符
19	Wrong type of register 使用寄存器出错
20	Operand must be segment or group 应该给出一个段名或组名。例如 ASSUME 语句中应为某段寄存器和指定一个段名或组名，而不应是别的标号或变量名等
21	Symbol has no segment 不知道标识符的段属性
22	Operand must be type specifier 操作数应给出类型说明，如 NEAR、FAR、BYTE 等
23	Symbol alread defined locally 以被指定为内部的标识符，企图在 EXTRN 语句中又定义外部标识
24	Segment paraneters are changed 段参数被改变。如同一标识符定义在不同段内
25	Improper align/combin type 段定义时的定位类型/组合类型使用出错
26	Reference to multidefined symbol 指令引用了多重定义的标识符
27	Operand expected 需要一个操作数，只有操作符
28	Operator expected 需要一个操作符，但只有操作数
29	Divdsion by 0 or overflow 除以 0 或溢出
30	Negative shift count\ 运算符 SHL 或 SHR 的移位表达式值为负数
31	Operand type must match 操作数类型不匹配。双操作数指令的两个操作数长度不一致，一个是字节，一个是字
32	Illegal use of external 外部符号使用出错
33	Must be record field name 应为记录字段名。在记录字段名位置上出现另外的符号
34	Must be record name or field name 应为记录名或记录字段名。在记录名或记录字段名位置上出现另外的符号
35	Operand must be size 应指明操作数的长度（如 BYTE、WORD 等）。通常使用 PTR 运算即可改正
36	Must be variable,label,or constant 应该是变量名、标号、或常数的位置上出现了其他信息
37	Must be stucture field name 应该为结构字段名。在结构字段名位置上出现了另外的符号
38	Lefe operand must segment 操作数的左边应该是段的信息。如设 DA1、DA2 均是变量名，下列语句就是错误的:“MOV AX，DA1：DA2”。DA1 位置上应使用某段寄存器名
39	One operand must constant 操作数必须是常数。

40	Operand must be in same segment or one constant “—”运算符用错。例如“MOV AL，—VAR”，其中 VAR 是变量名，应有一常数参加运算。又如两个不同段的变量名相减出错
41	Normal type operand expected 要求给出一个正常的操作数。
42	Constant expected 要求给出一个常数。
43	Operand must have segment 运算符 SEG 用错。
44	Must be associated with data 在必须与数据段有关的位置上出现了代码段有关的项
45	Must be associated with code 在必须与代码段有关的位置上出现了数据段有关的项
46	Multiple base registers 同时使用了多个基址寄存器。如“MOV AX ，[SI][BP]”
47	Multiple index registers 同时使用了多个变址寄存器。如“MOV AX ，[SI][DI]”
48	Must be index or base register 指令仅要求使用基址寄存器或变址寄存器，而不能使用其他寄存器。
49	Illegal use of register 非法使用寄存器出错
50	Value is out of range 数值太大，超过允许值。例如：“MOV AL ，100H”
51	Operand not in current CS ASSUME segment 操作数不在当前代码段内。通常指转移指令的目标地址不在当前 CS 段内
52	Improper operand type 操作数类型使用不当。例如：“MOV VAR1，VAR2”。两个操作数均为存储器操作数，不能汇编出目标代码
53	Jump out of range by %ld byte 条件转移指令跳转范围超过-128~+127 个字节。出错厂、信息同时给出超过的字节数
54	Index displacement must be constant 变址寻址的位移量必须是常数
55	Illegal register value 非法的寄存器值。目标代码中表达寄存器的值超过 7
56	Immediate mode illegal 不允许使用立即数寻址。例如“MOV DS，CODE”其中 CODE 是段名，不能把段名作为立即数传送给段寄存器 DS
57	Illegal size for operand 使用操作数大小（字节数）出错。例如：使用双字的存储器操作数
58	Byte register illegal 要求用字寄存器的指令使用了字节寄存器。如 PUSH，POP 指令的操作数寄存器必须是字寄存器
59	Illegal uer of CS register 指令中错误使用了段寄存器 CS。如：“MOV CS，AX”CS 不能做目的操作数
60	Must be accumulator register 要求用 AX 或 AL 的位置上使用可其他寄存器。如 IN，OUT 指令必须使用累加器 AX 或 AL
61	Improper uer of segment register 不允许使用段寄存器的位置上使用了段寄存器。如“SHL DS，1”
62	Missing or unreachable CS

	试图跳转去执行一个 CS 达不到的标号。通常是指缺少 ASSUME 语句中 CS 与代码段相关联
63	Operand combination illegal 双操作数指令中两个操作数组合出错
64	Near JMP/CALL to different CS 试图用 NEAR 属性的转移指令跳转到不在当前段的一个地址
65	Label cannot have segment override 段前缀使用出错
66	Must have instuction agter prefix 在重复前缀 REP，REPE，REPNE 后面必须有指令
67	Cannot override ES for destination 串操作指令中目的操作数不能用其他段寄存器替代 ES
68	Cannot address with srgment register 指令中寻找一个操作数，但 ASSUME 语句中未指明哪个段寄存器与该操作数所在段有关联
69	Must be in segment block 指令语句没有在段内
70	Cannot use EVEN or ALIGN with byte alignment 在段定义伪指令的定位类型中选用 BYTE，这时不能使用 EVEN 或 ALIGN 伪指令
71	Forward needs override or FAR 转移指令的目标没有在源程序中说明为 FAR 属性，可用 PTR 指定
72	Illegal value for DUP count 操作符 DUP 前的重复次数是非法的或未定义
73	Symbol id already external 在模块内试图定义的符号，它已在外部符号伪指令中说明
74	DUP nesting too deep 操作数 DUP 的嵌套太深
75	Illegak use of undefinde operand(?) 不定操作符“？”使用不当。例如“DB 1OH DUP（？+2)”
76	Too many valer for struc or record initialization 在定义结构变量或记录变量时，初始值太多
77	Angle brackets requored around initialized list 定义结构体变量时，初始值未用尖括号（<>）括起来
78	Directive illegal structure 在结构体定义中的伪指令使用不当。结构定义中的伪指令语句仅二种：分号（;）开始的注释语句和用 DB、DW 等数据定义伪指令语句
79	Override with DUP illegal 在结构变量初始值表中使用 DUP 操作符出错
80	Field cannot be overridden 在定义结构变量语句中试图对一个不允许修改的字段设置初值
81	Override id of wrong type 在定义结构变量语句中设置初值时类型出错
82	Circular chain of EQU aliases 用等值语句定义的符号名，最后又返回指向它自己。如： A EQU B B EQU A
83	Cannot emulate cooprocessor opcode 仿真器不能支持的 8087 协处理器操作码
84	End of file,not END directive 源程序文件无 END 结束命令
85	Data emitted with no segment 语句数据没有在段内