|  |  |
| --- | --- |
|  |  |

**合肥工业大学**

**实验报告**

**课 程： 汇编语言程序设计**

**专业班级： 计算机18-3班**

**学 号： 2018211991**

**姓 名： 余梓俊**

**实验一 Debug程序的使用**

1. **实验目的**

1、熟悉DEBUG程序中的命令，学会在DEBUG下调试运行汇编语言源程序。

2、掌握8086/8088的寻址方式及多字节数据的处理方法。

1. **实验内容**

1、利用DEBUG程序中的“E”命令，将两个多字节数“12345678H”和“FEDCBA98H”分别送入起始地址为DS:0200H和DS:0204H两个单元中。

2、分别用直接寻址方式和寄存器间接寻址方式编写程序段，实现将DS:0200H单元和DS:0204H单元中的数据相加，并将运算结果存放在DS:0208H单元中。

要求：

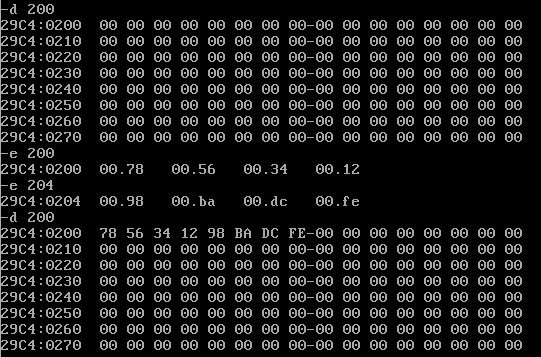
本次实验的内容均在DEBUG下完成，实现数据的装入、修改、显示；汇编语言程序段的编辑、汇编和反汇编；程序的运行和结果检查。

1. **实验过程和程序**

实验过程见截图

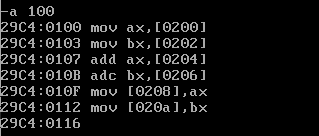
1. **实验结果（包括必要的截图）**

**1、利用DEBUG程序中的“E”命令，将两个多字节数“12345678H”和“FEDCBA98H”分别送入起始地址为DS:0200H和DS:0204H两个单元中。**

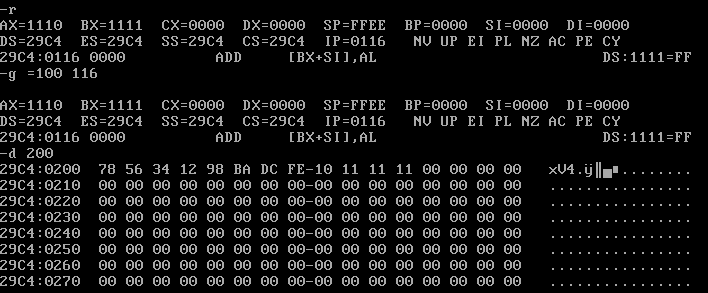


**2、分别用直接寻址方式和寄存器间接寻址方式编写程序段，实现将DS:0200H单元和DS:0204H单元中的数据相加，并将运算结果存放在DS:0208H单元中。**

**直接寻址方式**

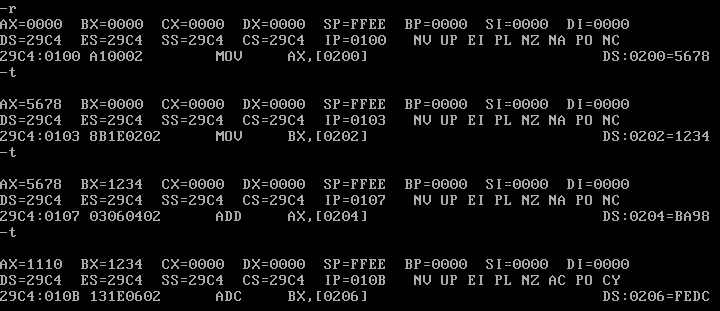


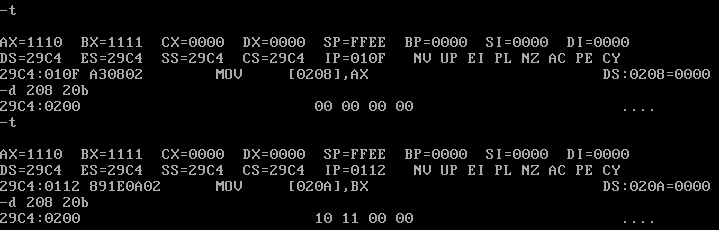
**直接执行结果：**

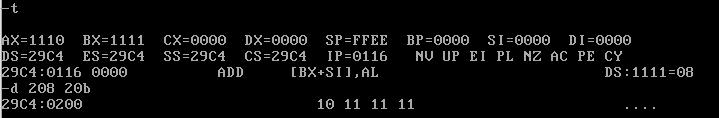


结果为11111110H外加CY即进位标志为1，结果正确

**单步执行过程：**

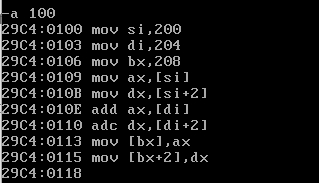




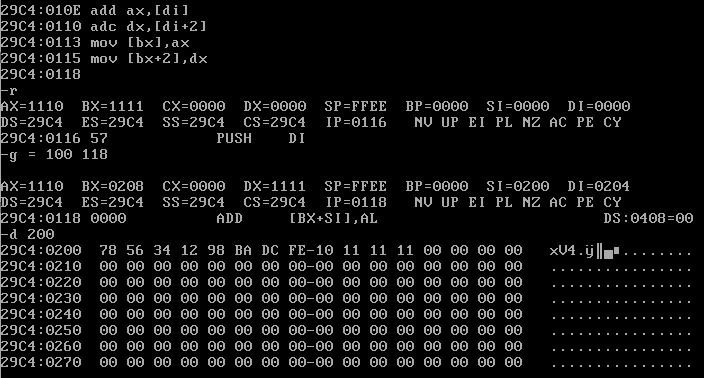


过程中的寄存器变化与内存变化与预期相同

**间接寻址方式**

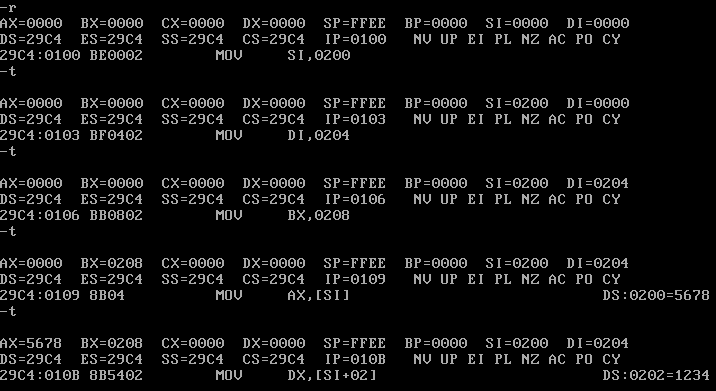


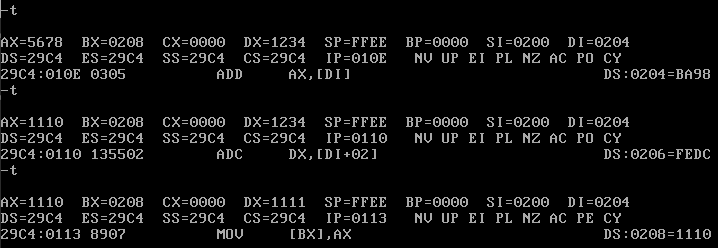
**直接执行结果**

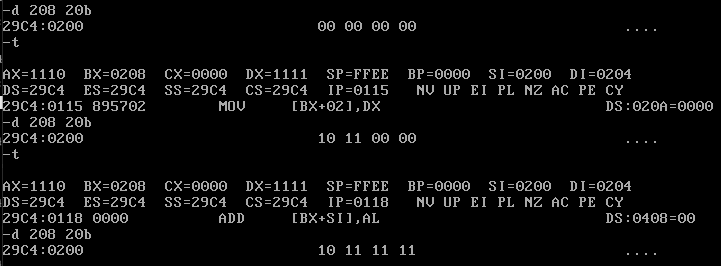


结果与直接寻址方式得到的结果相同

**单步执行过程：**







过程中的寄存器变化与内存变化与预期相同

1. **实验体会**

*使用e命令向内存中写入数据时应注意小端对齐。*

**实验二 汇编语言程序设计(顺序、多分支、循环)**

1. **实验目的**

1、掌握顺序和循环程序结构和设计方法；

2、熟悉在PC机上建立、汇编、连接、调试和运行8086/8088汇编语言程序的过程。

1. **实验内容**

1、X、Y、Z、V均为字变量，在X、Y、Z、V字单元中存放是16位带符号数。试编写汇编语言程序完成以下功能：

①计算表达式值（V–（X\*Y+Z-720））/X；

②将上述表达式运算结果整数放在SUM1单元，余数放在SUM2单元。

2、使用地址表实现如下功能：根据输入的数字1－7，分别显示相应的英文星期名，如果输入其他字符，则重新输入。

3、求一个班50名学生成绩的平均值、最大值和最小值,并将结果显示出来。

4、用串操作指令将DS：0000开始20个字节初始化为：0CCH。并将这20个字节的内容复制到其后的20个字节里。

1. **实验过程和程序**

**第一题**

; sum1 = ( V - ( X \* Y + Z - 720 ) ) / X

; sum2 = ( V - ( X \* Y + Z - 720 ) ) % X

.model small

.data

    X       dw -10

    Y       dw 1

    Z       dw 100

    V       dw 1000

    sum1    dw ?

    sum2    dw ?

.code

start:

    mov ax, @data

    mov ds, ax

    mov ax, X

    imul Y

    ; [dx, ax] = X \* Y

    add ax, Z

    adc dx, 0

    sub ax, 720

    sbb dx, 0

    ; [dx, ax] = X \* Y + Z - 720

    sub V, ax

    mov ax, 0

    sbb ax, dx

    mov dx, ax

    mov ax, V

    ; [dx, ax] =  V - ( X \* Y + Z - 720 )

    ; 1630 / -10

    idiv X

    ; ax =  ( V - ( X \* Y + Z - 720 ) ) / X

    ; dx =  ( V - ( X \* Y + Z - 720 ) ) % X

    mov sum1, ax

    mov sum2, dx

    mov ax, 4c00h

    int 21h

end start

**第二题**

; Display day of week based on the input number.

.model small

.stack 100h

.data

    table   dw day1, day2, day3, day4, day5, day6, day7

    crlf    db 0dh, 0ah, '$'

    s0      db 'please input number between 1 to 7$'

    s1      db 'Monday$'

    s2      db 'Tuesday$'

    s3      db 'Wednesday$'

    s4      db 'Thursday$'

    s5      db 'Friday$'

    s6      db 'Saturday$'

    s7      db 'Sunday$'

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

start:

    mov ax, @data

    mov ds, ax

input:

    lea dx, s0

    mov ah, 9

    int 21h

    printStr crlf

    mov ah, 1

    int 21h

    ; store the input char in bx

    mov bx, ax

    printStr crlf

    cmp bl, '1'

    jb input

    cmp bl, '7'

    ja input

    and bx, 000fh

    dec bx

    shl bx, 1

    jmp table[bx]

display:

    mov ah, 9

    int 21h

    printStr crlf

    mov ah, 4ch

    int 21h

day1:

    lea dx, s1

    jmp display

day2:

    lea dx, s2

    jmp display

day3:

    lea dx, s3

    jmp display

day4:

    lea dx, s4

    jmp display

day5:

    lea dx, s5

    jmp display

day6:

    lea dx, s6

    jmp display

day7:

    lea dx, s7

    jmp display

end start

**第三题**

; Find the average, maximum and minimum values

; of 50 students and display the results.

; Scores can be stored as unsigned number.

; 50 \* 100 = 5000 < ffffH (unsigned)

.model small

.stack 100h

.data

    crlf        db 0dh, 0ah, '$'

    maxStr      db 'The maximum score is $'

    minStr      db 'The minimum score is $'

    avgStr      db 'The average score is $'

    max         dw 0

    min         dw 100

    avg         dw 0

    count       equ 50

    scores      db 5 dup(100)

                db 5 dup(95)

                db 5 dup(90)

                db 5 dup(85)

                db 5 dup(80)

                db 5 dup(75)

                db 5 dup(70)

                db 5 dup(65)

                db 5 dup(55)

                db 5 dup(60)

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

printUNum proc

    ; print unsigned number

    ; param al: num

    push cx

    push dx

    mov cx, 0

    pushRemainders:

    and ah, 0

    mov dl, 10

    div dl

    mov dl, ah

    and dh, 0

    push dx

    inc cx

    cmp al, 0

    jne pushRemainders

    popRemainders:

    pop dx

    or dl, 30h

    mov ah, 2

    int 21h

    loop popRemainders

    pop dx

    pop cx

    ret

printUNum endp

start:

    mov ax, @data

    mov ds, ax

    mov cx, count

    lea bx, scores

    mov si, 0

loadScores:

    mov dl, [bx + si]

    and dh, 0

    cmp dx, max

    ; if dx <= max, don't update max

    jbe compareMin

    mov max, dx

    compareMin:

    ; if dx >= min, don't update min

    cmp dx, min

    jae addScore

    mov min, dx

    addScore:

    add avg, dx

    inc si

    loop loadScores

    mov ax, avg

    mov bl, count

    div bl

    and ah, 0

    mov avg, ax

    printStr maxStr

    mov ax, max

    call printUNum

    printStr crlf

    printStr minStr

    mov ax, min

    call printUNum

    printStr crlf

    printStr avgStr

    mov ax, avg

    call printUNum

    printStr crlf

    mov ah, 4ch

    int 21h

end start

**第四题**

; Initialize the 20 bytes starting at DS:0000

; to 0cch and copy to into the next 20 bytes.

assume cs:code, ds:data

data segment

    db 40 dup(10)

data ends

code segment

start:

    mov ax, data

    mov ds, ax

    mov es, ax

    cld

    mov cx, 20

    mov di, 0

    mov al, 0cch

    rep stosb

    mov cx, 20

    mov si, 0

    rep movsb

    mov ah, 4ch

    int 21h

code ends

end start

1. **实验结果（包括必要的截图）**

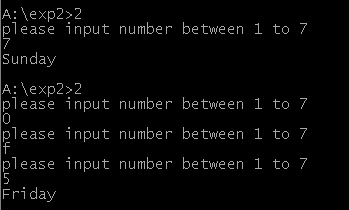
**第一题结果**

（V–（X\*Y+Z-720））/X；

当x=-10，y=1，z=100，v=1000，结果为商-163，ax应为ff5d，余0，dx应为0

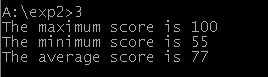


**第二题结果**



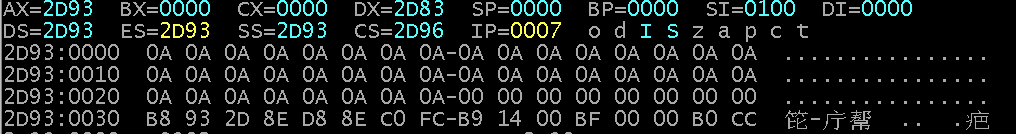
**第三题结果**

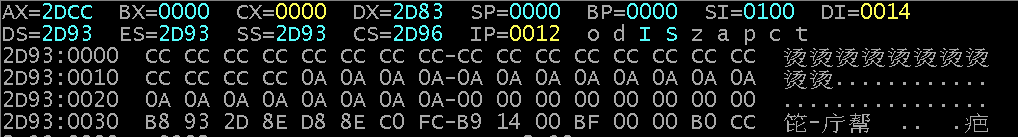
50个分数，分别是100，95，90 … ，55，每个分数重复5个，结果应为，最大值100，最小值55，平均值77



**第四题结果**

DS：0000初始值为40个10（0ah），第一次串操作使前20个变为cch，第二次串操作将这20个cch复制到后面的20个字节。







1. **实验体会**

*1.将输出字符串这样一个小功能编写为宏printStr，参数为字符串变量名，可以减少许多重复操作，例如在需要输出换行时，只需要写一句printStr crlf（在数据段定义crlf db 0dh, 0ah, ‘$’）。*

*2.在完成第四题的过程中发现，如果采用简化段的定义，data段地址会紧跟在code段中代码的最后一个字节后面，也就是说data段的首地址还存放着部分代码，这导致在初始化前20个字节为0cch时将后面部分代码全修改为0cch，导致实验失败，因此最终改用了完整段定义的形式。*

*3.在编写循环时一定要注意改变一些需要改变的值，例如通过bx+si来寻址时，要注意不要忘记在循环的最后写上inc si。*

**实验三 循环和子程序设计**

1. **实验目的**

1、掌握循环程序和子程序的设计方法；

2、熟悉在PC机上建立、汇编、连接、调试和运行8086/8088汇编语言程序的过程。

1. **实验内容**

1. 在内存Score缓冲区中存放有100个学生的成绩数据，为无符号字节数。设计程序完成如下功能：根据用户输入的一个2位十进制数，作为查找对象，在该数组中查找，若找到则显示“Y”，若没找到则显示“N” 。

2. BL中的只有一位为1。编写程序测试，并输出提示信息“The X Bit is 1”，要求：地址表。

3. 从键盘读入一个字符串，以Enter结束，字符串不超过50个字符，并打印该字符串(附加题：查找中间是否有‘asm’子串。如果有，输出‘Yes’；否则，输出‘No’)。

4. 编写一个子程序计算z=f(x,y)=x\*y+x-y（x,y,z有符号数内存数）。要求通过堆栈*(寄存器、内存)*传送所有参数。(要求输入输出实现，且有提示)

5. 实践C🡪Asm的转换过程。

1. **实验过程和程序**

**第一题**

.model small

.stack 100h

.data

    scores  db 10 dup(100)

            db 10 dup(95)

            db 10 dup(90)

            db 10 dup(85)

            db 10 dup(80)

            db 10 dup(75)

            db 10 dup(70)

            db 10 dup(65)

            db 10 dup(55)

            db 10 dup(60)

    y       db 'Y$'

    n       db 'N$'

    crlf    db 0dh, 0ah, '$'

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

start:

    mov ax, @data

    mov ds, ax

    mov es, ax

    mov ah, 1

    int 21h

    mov dh, al

    and dh, 0fh

    mov ah, 1

    int 21h

    mov dl, al

    and dl, 0fh

    mov al, dh

    mov dh, 10

    mul dh

    add al, dl

    printStr crlf

    mov cx, 100

    lea di, scores

    repnz scasb

    cmp cx, 0

    je printN

    printStr y

    jmp quit

    printN:

    printStr n

quit:

    mov ah, 4ch

    int 21h

end start

**第二题**

.model small

.stack 100h

.data

    table   dw b1, b2, b3, b4, b5, b6, b7, b8

    s1      db   'The first bit is 1', 0dh, 0ah, '$'

    s2      db  'The second bit is 1', 0dh, 0ah, '$'

    s3      db   'The third bit is 1', 0dh, 0ah, '$'

    s4      db  'The fourth bit is 1', 0dh, 0ah, '$'

    s5      db   'The fifth bit is 1', 0dh, 0ah, '$'

    s6      db   'The sixth bit is 1', 0dh, 0ah, '$'

    s7      db 'The seventh bit is 1', 0dh, 0ah, '$'

    s8      db  'The eighth bit is 1', 0dh, 0ah, '$'

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

start:

    mov ax, @data

    mov ds, ax

    mov bl, 00100000b

    mov cx, 8

    mov al, bl

    mov bx, 0

    l1:

    shl al, 1

    jc find

    inc bx

    inc bx

    loop l1

    jmp quit

    find:

    jmp table[bx]

    b1:

    printStr s1

    jmp quit

    b2:

    printStr s2

    jmp quit

    b3:

    printStr s3

    jmp quit

    b4:

    printStr s4

    jmp quit

    b5:

    printStr s5

    jmp quit

    b6:

    printStr s6

    jmp quit

    b7:

    printStr s7

    jmp quit

    b8:

    printStr s8

    jmp quit

quit:

    mov ah, 4ch

    int 21h

end start

**第三题**

.model small

.stack 100h

.data

    str db 50 dup(0)

    y   db 'Yes$'

    n   db 'No$'

    a   db 0

    s   db 0

    m   db 0

    crlf db 0dh, 0ah, '$'

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

findAsm proc

    .IF m == 1

        ret

    .ENDIF

    .IF a == 0

        .IF al == 'a'

            mov a, 1

        .ENDIF

        ret

    .ENDIF

    .IF s == 0

        .IF al == 's'

            mov s, 1

            ret

        .ENDIF

        .IF al == 'a'

            ret

        .ENDIF

        mov a, 0

        ret

    .ENDIF

    .IF al == 'a'

        mov s, 0

        ret

    .ENDIF

    .IF al == 'm'

        mov m, 1

        ret

    .ENDIF

    .IF al == 'a'

        mov s, 0

        ret

    .ENDIF

    mov a, 0

    mov s, 0

    ret

findAsm endp

start:

    mov ax, @data

    mov ds, ax

    mov di, 0

    lea bx, str

    input:

    mov ah, 1

    int 21h

    call findAsm

    mov [bx + di], al

    inc di

    cmp al, 0dh

    jne input

    mov [bx + di], 0ah

    inc di

    mov [bx + di], '$'

    printStr crlf

    printStr str

    .IF m == 1

        printStr y

    .ELSE

        printStr n

    .ENDIF

    mov ah, 4ch

    int 21h

end start

**第四题 – 采用寄存器传参**

; f = x \* y + x - y

; x, y: 1 byte signed

; f: 2 byte signed

.model small

.stack 100h

.data

    x dw ?

    y dw ?

    f dw ?

    hintX db 'Please input x between -128 to 127 (end with enter)', 0dh, 0ah, '$'

    hintY db 'Please input y between -128 to 127 (end with enter)', 0dh, 0ah, '$'

    hintError db 'Wrong number format, input again.', 0dh, 0ah, '$'

    hintRes db 'The result of f(x, y) is $'

.code

printStr MACRO strAddr

    push dx

    push ax

    lea dx, strAddr

    mov ah, 9

    int 21h

    pop ax

    pop dx

ENDM

inputNum proc

    ; param: 1. destination to store the input number

    push bp

    mov bp, sp

    push ax

    push bx

    push cx

    push dx

    mov dx, 0

    mov cx, 10

    mov ah, 1

    int 21h

    .IF al == '-'

        mov bx, 1

    .ELSE

        mov bx, 0

        and al, 0fh

        add dl, al

    .ENDIF

    l1:

    mov ah, 1

    int 21h

    cmp al, 0dh

    je endInput

    and al, 0fh

    push ax

    mov al, dl

    mul cl

    mov dl, al

    pop ax

    add dl, al

    jmp l1

    endInput:

    .IF bx == 1

        mov ax, 100h

        sub ax, dx

        cbw

        mov dx, ax

    .ENDIF

    mov bx, [bp + 4]

    mov [bx], dx

    mov ah, 2

    mov dl, 0dh

    int 21h

    mov dl ,0ah

    int 21h

    pop dx

    pop bx

    pop cx

    pop ax

    pop bp

    ret 2

inputNum endp

printNum proc

    ; param: 1. the number that is going to be displayed, length: 2byte

    push bp

    mov bp, sp

    push ax

    push bx

    push cx

    push dx

    mov ax, [bp + 4]

    cmp ax, 0

    jge continue

    push ax

    mov dl, 2dh  ; 2d: '-'

    mov ah, 2

    int 21h

    pop ax

    mov dx, 0

    sub dx, ax

    mov ax, dx

    continue:

    mov cx, 0

    mov bx, 10

    .WHILE ax > 0

        inc cx

        mov dx, 0

        div bx

        push dx

    .ENDW

    mov ah, 2

    cmp cx, 0

    jne l2

    mov dl, '0'

    int 21h

    jmp return

    l2:

    pop dx

    add dl, 30h

    int 21h

    loop l2

    return:

    pop dx

    pop cx

    pop bx

    pop ax

    pop bp

    ret 2

printNum endp

computeF proc

    ; use registry

    ; param: al: x, bl: y

    ; return: ax: f

    push dx

    mov dx, ax  ; store x in dl

    imul bl     ; ax = x \* y

    add ax, dx

    sub ax, bx

    pop dx

    ret

computeF endp

start:

    mov ax, @data

    mov ds, ax

    printStr hintX

    lea ax, x

    push ax

    call inputNum

    printStr hintY

    lea ax, y

    push ax

    call inputNum

    mov ax, x

    mov bx, y

    printStr hintRes

    call computeF

    push ax

    call printNum

    mov ah, 4ch

    int 21h

end start

**第四题 – 采用堆栈传参**

**（代码其余部分相同，仅展示computeF子过程和部分主程序）**

computeF proc

    ; use stack

    ; param: [bp + 4]: y, [bp + 6]: x

    ; return: [bp + 8]: f

    push bp

    mov bp, sp

    push ax

    push bx

    push dx

    mov bx, [bp + 4]

    mov ax, [bp + 6]

    mov dx, ax  ; store x in dl

    imul bl     ; ax = x \* y

    add ax, dx

    sub ax, bx

    mov [bp + 8], ax

    pop dx

    pop bx

    pop ax

    pop bp

    ret 4

computeF endp

    printStr hintRes

    sub sp, 2

    push x

    push y

    call computeF

    pop ax

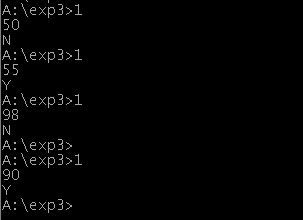
    call printNum

**第五题**

**代码见实验结果部分**

1. **实验结果（包括必要的截图）**

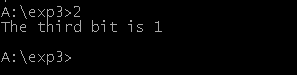
**第一题结果**



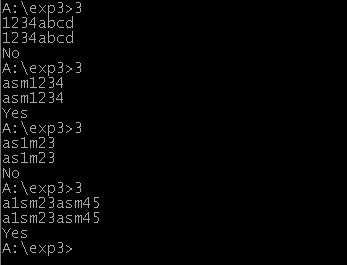
说明：从55到100，末位为5或0的成绩均存在，其余不存在

**第二题结果**

    mov bl, 00100000b

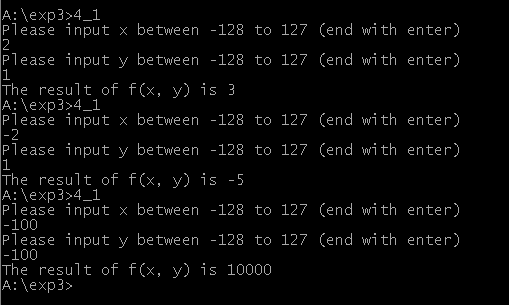


**第三题结果**



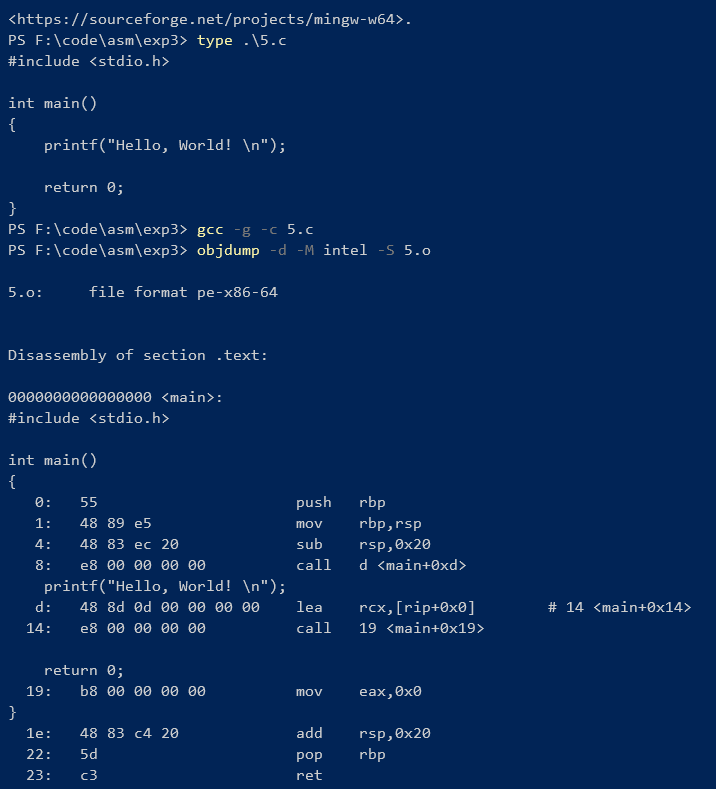
说明：实现了附加功能——判断输入串中是否含有“asm”字串

**第四题结果**



说明：堆栈传参方式的结果与上图相同，不再展示

**第五题结果**



1. **实验体会**

*1.体会到了高级语言与汇编语言的差距（差异），例如多层嵌套的条件、循环分支，函数调用的参数传递，运算过程对溢出的考虑等等，许多在高级语言编写过程中看似理所当然的事情，事实上高级语言的编译器/解释器帮我们做了大量的工作。*

*2.善用masm编译器的高级语法特性，可以大大减轻工作量，例如第三题判断是否包含特定字串的算法，用到了大量的分支语句，.IF语句帮了大忙。*

*3.堆栈传参初看别扭，但写过两三次之后，便感觉使用堆栈传参其实更加方便，不用考虑寄存器的语义（用哪些寄存器），也不用担心寄存器不够用。*

*4.注意条件跳转指令后只能接标号且必须是段内转移，因此第2题的地址表采用了先jc到标号再在标号处jmp table[bx]。*

*5.要注意编写过程中，变量或寄存器存放的是值还是址。*

**实验四 宏与中断程序设计**

1. **实验目的**

1、汇编语言宏程序设计；

2、掌握中断服务子程序的编写。

1. **实验内容**

1、编写一个宏，求三个数的最小数，原型为：MIN3 x,y,z,min

2、挂接1CH中断，正计时60秒后退出。要求屏幕显示0-59的秒数。

3、VC++中编写内联汇编，将字节数组的每个元素高低四位互换。

char dbyte[] = {0x34,0x45,0x56,0x67,0xaf};

1. **实验过程和程序**

第一题

min3 MACRO x, y, z, min

    push ax

    mov ax, x

    cmp ax, y

    jge j1

    j3:

    cmp ax, z

    jge j2

    jmp finally

    j2:

    mov ax, z

    jmp finally

    j1:

    mov ax, y

    jmp j3

    finally:

    mov min, ax

    pop ax

ENDM

**第二题**

.model small

.stack 100h

.data

    seconds db 0

    timer db 18

    oldIsr dw ?, ?

.code

printUNum proc

    ; print unsigned number

    ; param al: num

    push cx

    push dx

    mov cx, 0

    pushRemainders:

    and ah, 0

    mov dl, 10

    div dl

    mov dl, ah

    and dh, 0

    push dx

    inc cx

    cmp al, 0

    jne pushRemainders

    popRemainders:

    pop dx

    or dl, 30h

    mov ah, 2

    int 21h

    loop popRemainders

    mov dl, 20h

    mov ah, 2

    int 21h

    pop dx

    pop cx

    ret

printUNum endp

isr proc far

    push ax

    sti

    cmp timer, 0

    je print

    dec timer

    jmp endIsr

    print:

    mov al, seconds

    call printUNum

    inc seconds

    mov timer, 18

    endIsr:

    pushf

    call dword ptr oldIsr

    pop ax

    iret

isr endp

start:

    mov ax, @data

    mov ds, ax

    mov ax, 0

    mov es, ax

    mov ax, es:[1ch \* 4]

    mov oldIsr[0], ax

    mov ax, es:[1ch \* 4 + 2]

    mov oldIsr[2], ax

    mov word ptr es:[1ch \* 4], OFFSET isr

    mov word ptr es:[1ch \* 4 + 2], SEG isr

    again:

    cmp seconds, 59

    ja exit

    jmp again

    exit:

    mov ax, oldIsr[0]

    mov es:[1CH \* 4], ax

    mov ax, oldIsr[2]

    mov es:[1CH \* 4 + 2], ax

    mov ah, 4ch

    int 21h

end start

说明：printNum子过程代码同实验二第三题相同。

**第三题**

#include <iostream>

using namespace std;

char dbyte[] = { 0x34,0x45,0x56,0x67,0xaf };

int main() {

    for (int i = 0; i < 5; ++i) {

        cout << hex << (short)dbyte[i] << " ";

    }

    cout << endl;

    \_\_asm {

        mov ebx, 0

        mov ecx, 5

        l1:

        mov dl, dbyte[ebx]

        rol dl, 4

        mov dbyte[ebx], dl

        inc ebx

        loop l1

    }

    for (int i = 0; i < 5; ++i) {

        cout << hex << (short)dbyte[i] << " ";

    }

    cout << endl;

    return 0;

}

1. **实验结果（包括必要的截图）**

**第一题结果**

当x, y, z值为1, 2, 0时，min为0（mov ax, min后ax为0，即0000）

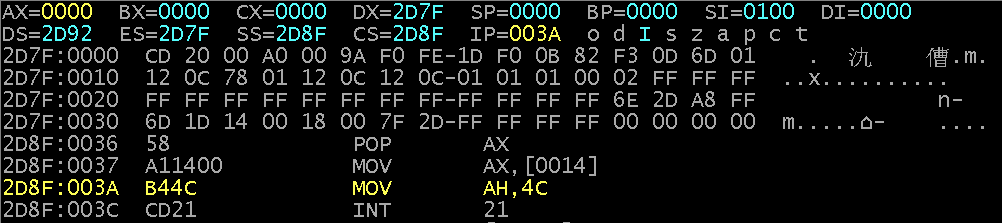
    mov x, 1

    mov y, 2

    mov z, 0

    min3 x, y, z, min

    mov ax, min



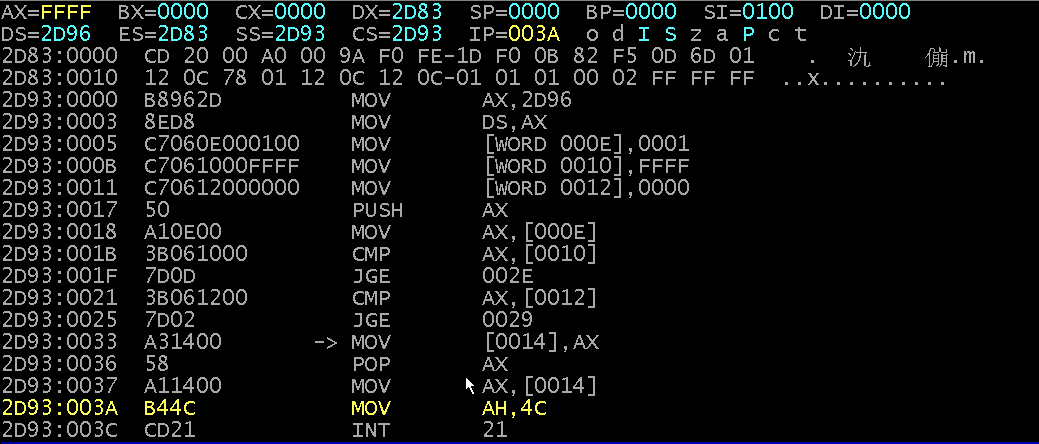
当x, y, z值为1, -1, 0时，min为-1（mov ax, min后ax为-1，即ffff）

    mov x, 1

    mov y, -1

    mov z, 0

    min3 x, y, z, min

    mov ax, min

**第二题结果**

**第三题结果**



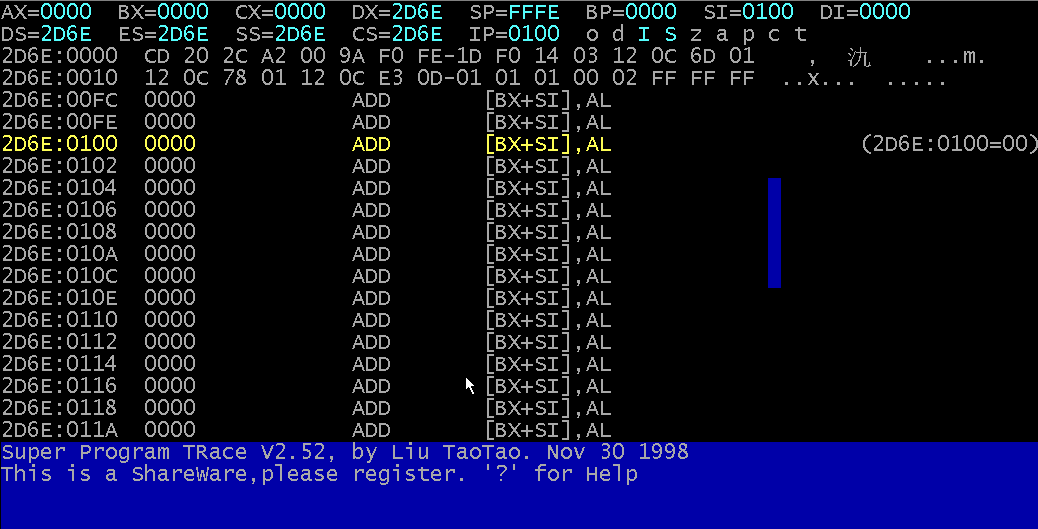
说明：使用了cout<<hex<<输出16进制，但因为ostream的hex只对整数有效，因此输出时把char强转成了short，导致后面有两个高位的ff。

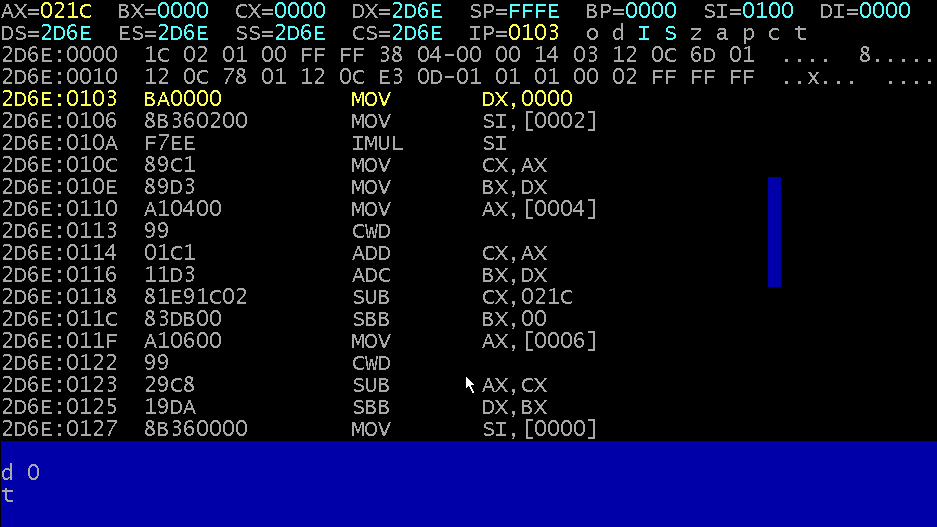
1. **实验体会**

*1.win98虚拟机里的dos方式在运行第二题的挂接中断的程序时总是崩溃，仍然不清楚原因，遂改用dosbox运行。*

**实验一附加题：**

1.从DS:0000H开始的5个16位带符号数分别记为x,y,z,v,w，用E命令初始化为：540,1,-1,1080,0, 试求w=(v-(x\*y+z-540))/x





540=2h, 1=1h, -1=ffffh, 1080=438h, 0=0h

w=(v-(x\*y+z-540))/x=(1080-(-1))/540=2。。。1

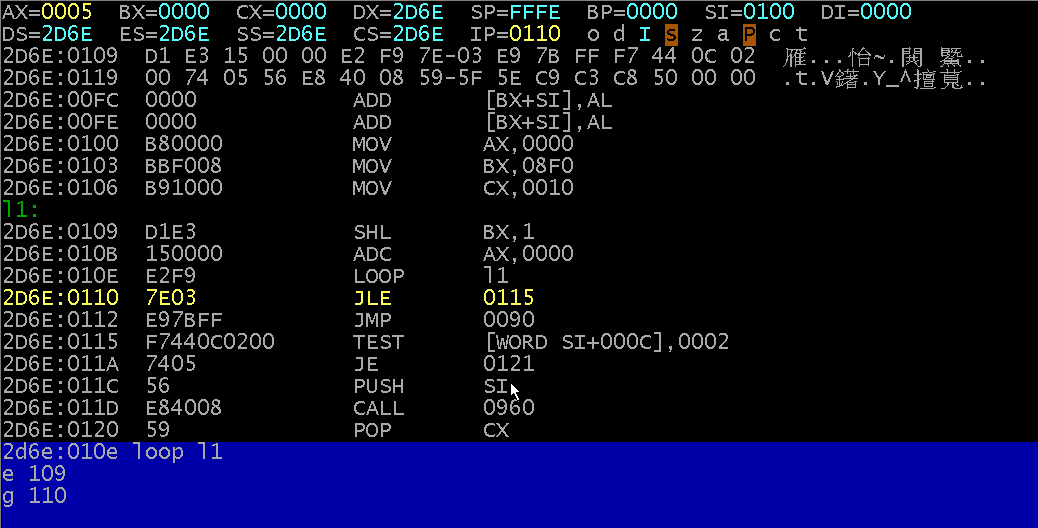
ax应为2，dx应为1，结果：



**2. 完成BX中1的个数的统计。**



bx为08f0，执行完后ax应为5



3.用串操作指令将DS：0000开始20个字节初始化为：0CCH。并将这20个字节的内容复制到其后的20个字节里。

同实验二第四题