微机原理第一次实验 实验报告

Cantjie

实验一：

实验任务：

1、 按要求对该程序进行修改，使其建立的数据为降序排列的十进制数。

实验思路较为简单，只需修改 AL 的初值以及循环中对 AL 的增序、降序即可。代码如下：

```
code segment
    assume cs:code
    ORG 2000H
start: mov DI,3500H
       mov CX,0010H
       mov AX,0015H
cnt:       mov [DI],AL
       inc DI
       sub al,01
       das
       loop cnt
       JMP $
code ends
    end start
```

2、 完成二进制双精度加法运算.计算 Z=X+Y。并将结果存入 3600H。测试标志寄存器各标志位的意义和指令执行对它的影响。

实验需要用到代码段，必须注意除了 assume ds:DATA 之外，必须在 start:之后手动将 ds 指向 DATA。

算法并不复杂，无需使用循环，因此只需将低 16 位相加之后，再使用带进位的加法指令将高 16 位相加即可。代码如下：

```
DATA segment
    x DD 12341234h
    y DD 11111111h
    z DD 0
DATA ends

code segment
    assume cs:code,ds:DATA
start:
    mov ax,DATA
    mov ds,ax
```

```
    lea bx,x
    mov ax,[bx+2]
    lea bx,y
    mov cx,[bx+2]

    add ax,cx

    lea bx,z
    mov [bx+2],ax

    lea bx,x
    mov ax,[bx+2]
    lea bx,y
    mov cx,[bx+2]
    adc ax,cx
    lea bx,z
    mov [bx],ax

    mov ah,4ch
    int 21h
code ends
    end start
```

实验一遇到的主要障碍及解决思路是：

①在对双字进行处理时，不知该如何仅取出低 16 位相加。之后发现利用：lea bx,x 和 mov ax,[bx+2]即可处理。

②在调试中发现即使 assume ds:data 后也写了将 ds 指向 data 的语句，执行时 ds 的值依然没有发生变化。原因是自己忘记 start 也是汇编关键字，在 start：之后的语句才会被执行，不能把 start:置于

```
mov ax,DATA
mov ds,ax
```

这两个语句之后。

实验二：

实验内容及思路：

有一个 10 字节的数组，其值分别是 06H，F2H，5AH，F4H，97H，64H，BBH，7FH，0FH，D8H。编程并显示结果：

（1） 如果数组是无符号数，按大小排序从内存单元(4000H)开始连续存入排序后数组，并求出最大值，并显示；

（2） 如果数组是有符号数，按大小排序从内存单元(4000H)开始连续

存入排序后数组，并求出最大值，并显示。

虽然是两个要求，但其实无符号数与有符号数的区别仅在于条件跳转时的一条语句，因此只需以无符号数为例写出程序即可。

实验中在冒泡排序和输出数组时需要用到双重循环，此时需要在内层循环前，用一个闲置的寄存器将 CX 保存起来，然后在内循环结束后，外循环结束前，再将外循环的 CX 恢复。这样就可以克服 loop 指令只检测 CX 是否为 0 的难点。

另一个难点在于将数值显示到屏幕，需要利用 DOS 中断调用 02H 来输出字符。同时需要将 0-9、A-F 转化到 ASCII 码才可输出，经过移位后与 OFH 相与后，判断是 0-9 还是 A-F，输出数字则将数字+30H，输出字母则将字母+37H。代码如下：

```
data segment
    ;ORG 4000H
    array db 06H,0F2H,5AH,0F4H,97H,64H,0BBH,7FH,0FH,0D8H,'$'
    len db 0AH
data ends

code segment
    assume cs:code,ds:data,es:data
start:
    mov ax,data
    mov ds,ax
    mov es,ax
    mov di,4000h ;init

    mov cx,0
    mov cl,len
    dec cx
outer_loop:
    mov si,cx ; put the outer iteration count into si
    mov bx,0
iner_loop:
    mov al,[bx+array]
    cmp al,[bx+array+1]
    JAE continue
    mov dl,[bx+array+1] ; exchange array[bx] and array[bx+1]
    mov [bx+array+1],al
    mov [bx+array],dl
    mov al,dl
continue:
    inc bx
    loop iner_loop
```

```
        ;mov [di],al
        ;inc di
        mov cx,si
        loop outer_loop


        mov bx,0 ;if data segment does not start from 4000H,then store the
array there
        mov cl,len
        mov di,4000h
    store:
        mov al,[array+bx]
        inc bx
        mov [di],al
        inc di
        loop store

        mov cl,len
        mov bx,0

    print_all:
        mov di,cx
        mov dl,[array+bx];transform to ascii to print
        mov cx,4
        shr dl,cl
        and dl,0Fh
        cmp dl,09h
        JBE digit_high
        add dl,37H
        JMP print_high
    digit_high:
        add dl,30H

    print_high:
        mov ah,02H
        int 21h

        mov dl,[array+bx]
        and dl,0Fh
        cmp dl,09H
        JBE digit_low
        add dl,37H
```

```
    JMP print_low
digit_low:
    add dl,30H

print_low:
    mov ah,02H
    int 21h

    inc bx
    mov ah,02h
    mov dl,20h
    int 21h

    mov cx,di
    loop print_all

    mov ah,4ch
    int 21h
code ends
    end start
```

运行后屏幕输出：
F4 F2 D8 BB 97 7F 64 5A 0F 06
同时内存 4000H 处也是对应结果。