

实验三 算术逻辑运算程序设计

一、 实验目的

1. 进一步掌握顺序程序设计方法。
2. 熟悉数据传送及算术和逻辑运算指令的用法。

二、 实验内容

1. 复习教材中顺序结构程序设计的相关内容。
2. 用 A 命令输入下面程序：

```
-A
xxxx:0100    MOV  AX,2
xxxx:0103    MOV  BX,AX
xxxx:0105    MOV  CL,2
xxxx:0107    SHL  AX,CL
xxxx:0109    ADD  AX,BX
xxxx:010B    SHL  AX,1
xxxx:010D    HLT
```

执行上述程序段，分析上述程序段的功能

3. 将 AX 寄存器中的 16 位数分成四组，每组四位，然后把这四组数分别放在 AL、BL、CL、DL 中。编写实现上述功能的程序 1。

4. 设 $X=8$ ，编写程序 2，求多项式 $Y = 2X^4 + 3X^3 + 5X^2 + 8X + 6$ 的值，并画出流程图。

三、 实验结果（截图）

1. 代码理解

该代码计算了 $(2^2AX + AX) \times 2 = 10AX$ 的结果，本代码完成后，AX 的值为 $2 \times 10 = 014H$ ，修改 0100 处的代码为 MOV AX,5 后，得到的结果为 028H，说明功能确实是计算 10AX

```
0740:0107 D3E0      SHL     AX,CL
-t
AX=0008 BX=0002 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=0109  NU UP EI PL NZ AC PO NC
0740:0109 01D8      ADD     AX,BX
-t
AX=000A BX=0002 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=010B  NU UP EI PL NZ NA PE NC
0740:010B D1E0      SHL     AX,1
-t
AX=0014 BX=0002 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=010D  NU UP EI PL NZ AC PE NC
0740:010D F4        HLT
-t
AX=0014 BX=0002 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=010E  NU UP EI PL NZ AC PE NC
0740:010E 0000      ADD     [BX+SI],AL      DS:0002=3F
-a 100
0740:0100 mov ax,5
0740:0103
```

```
AX=0005 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=0105  NU UP EI PL NZ NA PO NC
0740:0105 D3E0      SHL     AX,CL
-t
AX=0014 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=0107  NU UP EI PL NZ AC PE NC
0740:0107 01D8      ADD     AX,BX
-t
AX=0014 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=0109  NU UP EI PL NZ NA PE NC
0740:0109 D1E0      SHL     AX,1
-t
AX=0028 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=010B  NU UP EI PL NZ AC PE NC
0740:010B F4        HLT
-t
AX=0028 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=0740 ES=0740 SS=0740 CS=0740 IP=010C  NU UP EI PL NZ AC PE NC
0740:010C 0000      ADD     [BX+SI],AL      DS:0000=CD
```

2. 程序 1

按题目要求，将 AX 寄存器中的内容分四组存放在四个通用寄存器的低位中，整体思路就是每次取 AX 的低八位存入目标寄存器的低八位中，再使用与运算将低 5 位至低 8 位变为 0。此处 AX 一开始存入值 1234H，运行完结果如下，可以发现 AL、BL、CL、DL 中的值分别为 01H、02H、03H、04H。

```
077C:0013 B0E50F      AND     CH,0F
077C:0016 D3E8      SHR     AX,CL
077C:0018 8AD8      MOV     BL,AL
077C:001A B0E30F      AND     BL,0F
077C:001D D3E8      SHR     AX,CL
077C:001F 8ACD      MOV     CL,CH
-u
077C:0021 32ED      XOR     CH,CH
077C:0023 B8004C      MOV     AX,4C00
077C:0026 CD21      INT     21
077C:0028 218B163C     AND     [BP+DI+3C16],CX
077C:002C 218987BE     AND     [BX+DI+BE87],CX
077C:0030 228997C0     AND     CL,[BX+DI+C097]
077C:0034 225E8B      AND     BL,[BP-75]
077C:0037 E55D      IN      AX,5D
077C:0039 C3      RET
077C:003A 55      PUSH    BP
077C:003B 8BEC      MOV     BP,SP
077C:003D 81EC8600     SUB     SP,0086
-g 23

AX=0001 BX=0002 CX=0003 DX=0004 SP=0100 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076C CS=077C IP=0023  NU UP EI PL ZR NA PE NC
077C:0023 B8004C      MOV     AX,4C00
```

3. 程序 2

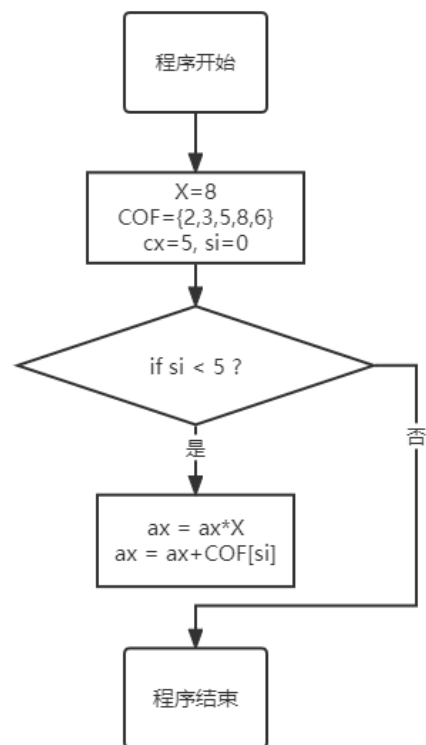
使用秦九韶算法将多项式进行变换，使得计算过程遵循相同的规律，以便使用循环进行计算。将 $Y = 2X^4 + 3X^3 + 5X^2 + 8X + 6$ 化为

$X\{X[X(X(2) + 3) + 5] + 8\} + 6$ 。原式中 Y 计算的结果应为 $(10118)_{10} = (2726)_{16}$ ，下图中 AX 确为 2786，结果正确。

```
077D:001D EBEF      JMP     000E
077D:001F B8004C      MOV     AX,4C00
-u
077D:0022 CD21      INT     21
077D:0024 225E8B      AND     BL,[BP-75]
077D:0027 E55D      IN      AX,5D
077D:0029 C3      RET
077D:002A 55      PUSH    BP
077D:002B 8BEC      MOV     BP,SP
077D:002D 81EC8600     SUB     SP,0086
077D:0031 57      PUSH    DI
077D:0032 56      PUSH    SI
077D:0033 B8BE05      MOV     AX,05BE
077D:0036 50      PUSH    AX
077D:0037 E8C371      CALL    71FD
077D:003A 83C402      ADD     SP,+02
077D:003D 8BF0      MOV     SI,AX
077D:003F 0BF6      OR      SI,SI
077D:0041 7461      JZ      00A4
-g 1f

AX=2786 BX=0000 CX=0005 DX=0000 SP=0100 BP=0000 SI=0005 DI=0000
DS=076C ES=075C SS=076D CS=077D IP=001F  NU UP EI PL ZR NA PE NC
077D:001F B8004C      MOV     AX,4C00
```

另附流程图如下



四、 实验报告要求（习题）

1. 程序 1

```
DATA SEGMENT
```

```
DATA ENDS
```

```
; 按惯例清出一片区域作为堆栈段
```

```
STACK SEGMENT STACK
```

```
    DW 128 DUP(0)
```

```
STACK ENDS
```

```
CODE SEGMENT
```

```
    ASSUME cs:CODE, DS:DATA, SS:STACK
```

```
START:
```

```
; 设置段寄存器
```

```
mov ax, DATA
```

```
mov ds, ax
```

```
; 以 1234H 为例，便于观察
```

```
; 计划将 01H、02H、03H、04H 最终分别放在 AL、BL、CL、DL 中
```

```
MOV AX, 1234H
```

```
    ; 每次操作完需要右移 4 位，预先存入 CL 中
    MOV CL, 4
    ; 将 AX 低八位放入 DX 低八位中
    MOV DL, AL
    ; 只取 DX 低四位 04H
    AND DL, 00001111b
    ; 逻辑右移 4 位（即 1 位十六进制）
    SHR AX, CL
    ; 由于进行大于 1 的移位只能使用 CL
    ; 原本应该给 CL 的结果先放在 CH 中
    MOV CH, AL
    ; 同样只取低四位得到 03H
    AND CH, 00001111b
    ; 逻辑右移 1 位十六进制
    SHR AX, CL
    ; 同样的操作
    MOV BL, AL
    ; 取得低四位 02H
    AND BL, 00001111b
    ; 这次移位完之后 AL 中剩下的就是 01H
    SHR AX, CL
    ; 赋值回来就可以
    MOV CL, CH
    ; 为了看起来更直观，将 CH 清零
    XOR CH, CH

    ; 带返回值退出的功能号
    MOV AX, 4C00H
    INT 21H
CODE ENDS

END START
```

2. 程序 2

```
DATA SEGMENT
    X DW 8
    ; 系数数组
    COF DW 02H,03H,05H,08H,06H
```

```
DATA ENDS

STACK SEGMENT STACK
    DW 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME cs:CODE, DS:DATA, SS:STACK
START:  ; 初始化代码段
        ; 设置段寄存器
        mov ax, DATA
        mov ds, ax
        ; ax 清零，后面始终保持内容在 ax
        mov ax, 0
        ; si 用作循环变量
        mov si, 0
CALCU:  ; 进行主体计算
        ; 循环体共执行 5 次
        cmp si, 0AH
        jz exit
        ; 在每次计算中都进行乘 X 再加上 COF[SI]的操作
        mov dx, X
        imul dx
        add ax, COF[si]
        ; si += 2
        inc si
        inc si
        ; 继续循环
        jmp CALCU
exit:   ; 退出段代码
        MOV AX, 4C00H
        INT 21H
CODE ENDS

END START
```

3. 如何用 debug 查看程序运行结果

下面叙述本次报告使用 debug 查看运行结果的过程

- ① 使用 debug [exe 程序名]，进入程序的调试
- ② 使用 g [断点]运行程序至断点处，断点可以使用相对地址指定，需要注意的是，指定的断点位置的语句并不会运行
- ③ 使用 r 命令观察寄存器，计算过程中把结果存在了 AX 中
- ④ 使用 d [内存地址]查看指定内存单元以后一段的内容，该命令中的地址也可以使用相对地址进行指定，默认查看的是 DS:指定偏移地址的位置

4. 遇到的问题及如何解决

在模拟器中似乎不支持将用伪指令定义的 Byte 类型直接给予寄存器，但 DOSBOX 上的汇编程序是支持的，在使用过程中总结了伪指令定义的变量名称用于寻址时的几个要点。

以下面的代码为例 `mov AL, COF[2]` 会被汇编程序翻译为直接寻址，即 `mov AL, [0004]`，而直接使用 `COF` 相当于 `COF[0]`，也是直接寻址，此处 `mov AH, COF` 被翻译为 `mov AH, [0002]`，而使用 `OFFSET` 伪指令才能得到伪指令定义变量在数据段中的偏移地址，例如 `mov BX, offset COF` 会被翻译为 `mov BX, 2`、`mov BX, offset COF[2]` 会被翻译为 `mov BX, 4`，可见是立即寻址。

```
DATA SEGMENT
    X DW 8
    COF DB 03H,02H,05H,08H,06H
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK
START:
    mov ax, DATA
    mov ds, ax
    mov al, COF[2]
    mov ah, COF
    mov bx, offset COF
    mov bx, offset COF[2]
CODE ENDS

END START
```

五、 个人体会与总结

此次实验尝试使用循环解决问题，一开始没有搞清楚不能用于寄存器变址寻址的寄存器，总是过不了汇编环节，加深了对各种寻址方式的印象。另外，对于

循环结构的程序，在调试时跳转指令中的伪指令均会被汇编成对应代码段的首地址。另外，仍需要注意使用 `mov` 指令时，数据长度的问题，两个操作数的长度一定要匹配对应。