微原第三次实验

# 实验报告

物理试验班 2160909016 张宇轩

## 顺序结构程序实验

实验代码：

```
DSEG SEGMENT
    U DB 09H
    V DB 16H
    W DB 02H
    X DB 03H
    Y DB 05H
    ; To be calculated:
    Z DW 00H
    ORG 100
    STR_1 DB "(U+V-W*X)/Y = $"
    ORG 200
    STR_WC DB "This program calculates (U+V-W*X)/Y",0ah,"$"
    STR_CS DB 0ah,"In Case of:",0ah,"$"
    ORG 300
    STR_S DB "  = "
    BUFFER DB ?
DSEG ENDS

SSEG SEGMENT
    DB 0
SSEG ENDS

CODE SEGMENT
    ASSUME cs:CODE, ds:DSEG, es:DSEG, ss:SSEG
START:
    mov  ax, DSEG
    mov  ds, ax
    mov  es, ax
    mov  ax, SSEG
    mov  ss, ax
    xor  ax, ax
    mov  sp, 0
    mov  bp, 0
    lea  dx, STR_WC
    mov  ah, 09H
    int  21H
    ; case 1
    call CALCULATE
    ; case 2
    mov  U, BYTE PTR 70
    mov  V, BYTE PTR 23
    mov  W, BYTE PTR 42
    mov  X, BYTE PTR 17
    mov  Y, BYTE PTR 41
    call CALCULATE
```

```
    mov  ah, 4CH
    mov  al, 00H
    int  21H
CALCULATE:
    call DISP_Val
    ;Step1 : W*X
    call CLR
    mov  al, W
    mul  X
    push ax
    ;Step2 : ax <- U+V-bx
    call CLR
    mov  al, U
    add  al, V
    adc  ah, 0
    pop  bx
    sub  ax, bx
    ;Step3 : ax <- (U+V-bx)/Y
    jns  CALC_NEXT
    mov  dx, 0FFFFH ;Make dx = FFFF (Negative Expansion)
CALC_NEXT:
    mov  cl, Y
    idiv cx
    mov  ds:Z, ax
    call  SHOW_RESULT
    ret
CLR:
    xor  ax, ax
    xor  bx, bx
    xor  cx, cx
    xor  dx, dx
    ret
FORMAT_INT:
    push ax
    push cx
    push dx
    push bx
    push bp
    test ax, 8000H
    js   FI_ZERO
    mov  dl, BYTE PTR '+'
    jmp  FI_NEXT
FI_ZERO:
    mov  cx, ax
    mov  ax, 0FFFFH
    sub  ax, cx
    inc  ax
    mov  dl, BYTE PTR '-'
FI_NEXT:
    push dx
    mov  bp, sp
    mov  dl, BYTE PTR '$'
    push dx
```

```
    mov  cx, 10
FI_LOOP1:
    xor  dx, dx
    div  cx
    add  dx, WORD PTR 30H
    push dx
    cmp  ax, 0
    jnz  FI_LOOP1
    mov  dx, ss:[bp]
    cmp  dl, BYTE PTR '-'
    jne  FI_LOOP2
    mov  ds:[bx], BYTE PTR '-'
    inc  bx
FI_LOOP2:
    pop  dx
    cmp  dl, '$'
    jz   FI_FINAL
    mov  ds:[bx], dl
    inc  bx
    jmp  FI_LOOP2
FI_FINAL:
    pop  bp ;sign BYTE, diserted
    pop  bp
    pop  dx
    sub  bx, dx
    pop  dx
    pop  cx
    pop  ax
    ret
FI_AFTER_N:
    mov  ds:[bx], BYTE PTR "-"
    inc  bx
    jmp  FI_FINAL
DISP_Val:
    push ax
    push bx
    push cx
    push dx

    lea  dx, STR_CS
    mov  ah, 09H
    int  21H

    xor  ax, ax
    mov  STR_S, BYTE PTR "U"
    mov  al, U
    call PRINT_Val

    xor  ax, ax
    mov  STR_S, BYTE PTR "V"
    mov  al, V
    call PRINT_Val
```

```
    xor  ax, ax
    mov  STR_S, BYTE PTR "W"
    mov  al, W
    call PRINT_Val

    xor  ax, ax
    mov  STR_S, BYTE PTR "X"
    mov  al, X
    call PRINT_Val

    xor  ax, ax
    mov  STR_S, BYTE PTR "Y"
    mov  al, Y
    call PRINT_Val

    pop  dx
    pop  cx
    pop  bx
    pop  ax
    ret

PRINT_Val:
    lea  bx, BUFFER
    call FORMAT_INT
    mov  ds:BUFFER[bx], BYTE PTR 0ah
    inc  bx
    mov  ds:BUFFER[bx], BYTE PTR "$"
    lea  dx, STR_S
    mov  ah, 09H
    int  21H
    ret

SHOW_RESULT:
    ;(U+V-W*X)/Y=Z
    lea  bx, BUFFER
    mov  ds:[bx], BYTE PTR '('
    inc  bx

    xor  ax, ax
    mov  al, U
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx
    mov  ds:[bx], BYTE PTR '+'
    inc  bx

    xor  ax, ax
    mov  al, V
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx
    mov  ds:[bx], BYTE PTR '-'
    inc  bx
```

```
    xor  ax, ax
    mov  al, W
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx
    mov  ds:[bx], BYTE PTR '*'
    inc  bx

    xor  ax, ax
    mov  al, X
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx
    mov  ds:[bx], BYTE PTR ')'
    inc  bx
    mov  ds:[bx], BYTE PTR '/'
    inc  bx

    xor  ax, ax
    mov  al, Y
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx
    mov  ds:[bx], BYTE PTR '='
    inc  bx

    xor  ax, ax
    mov  ax, Z
    mov  dx, bx
    call FORMAT_INT
    add  bx, dx

    mov  ds:[bx], BYTE PTR 0ah
    inc  bx
    mov  ds:[bx], BYTE PTR '$'
    lea  dx, BUFFER
    mov  ah, 09H
    int  21H
    ret
CODE ENDS
END START
```

实验结果：

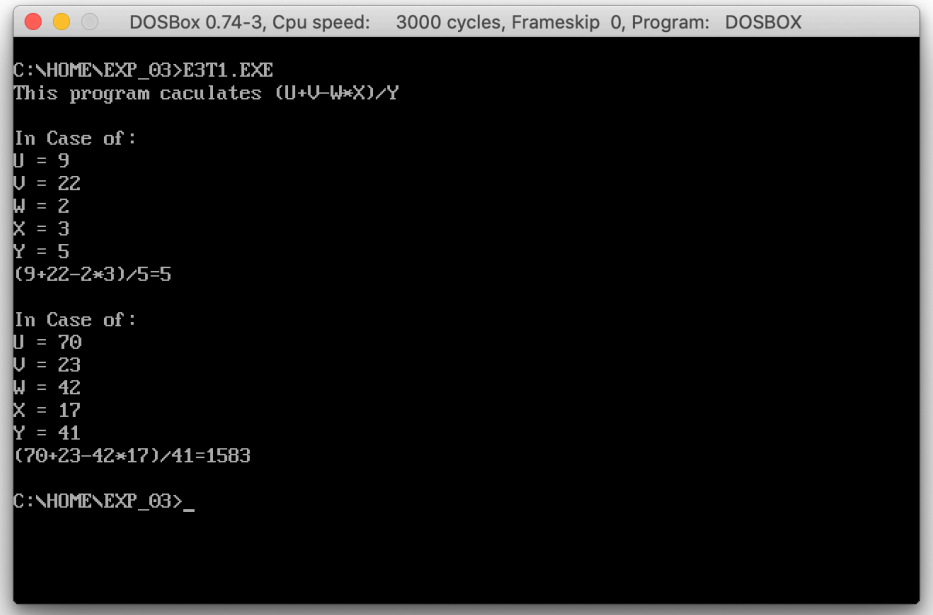


FIG 1.1

未优化的代码的运行结果

在计算第二组数值时由于减法产生负数，计算结果错误

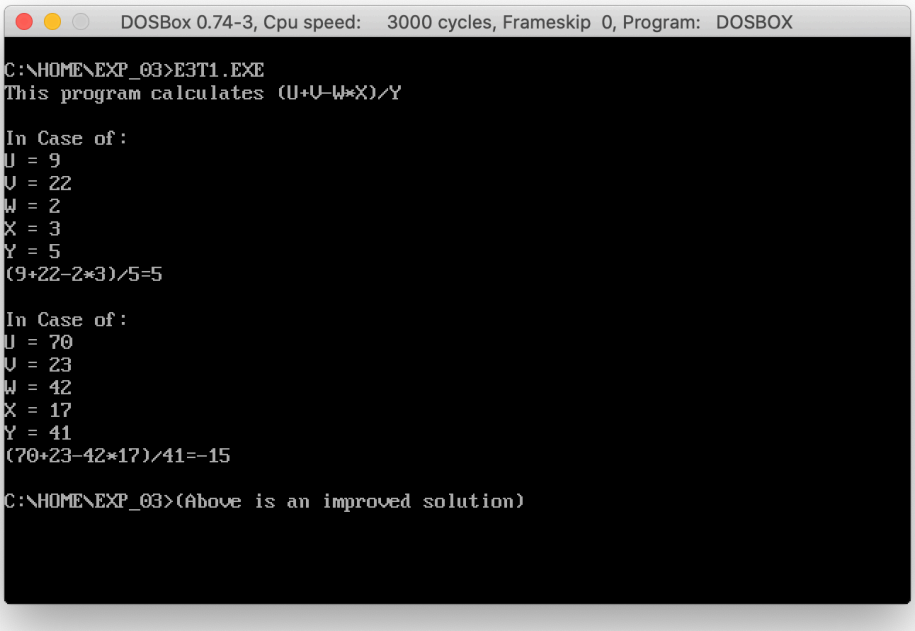


FIG 1.2

优化后的代码 (代码中蓝色部分为优化内容)，对于两组数值均可以正常计算