学院：**信息学院**　学期：**2022 年秋季学期**　课程名称：**汇编语言程序设计**　指导教师：**周丽华**

专业：**计算机科学与技术**　年级：**2020**　姓名：**胡诚皓**　学号：**20201060330**

实验九 子程序设计

一、实验目的

1. 学习子程序的定义和调用方法，加深对子程序的理解。
2. 掌握子程序的设计、编写及调试方法。

二、实验内容

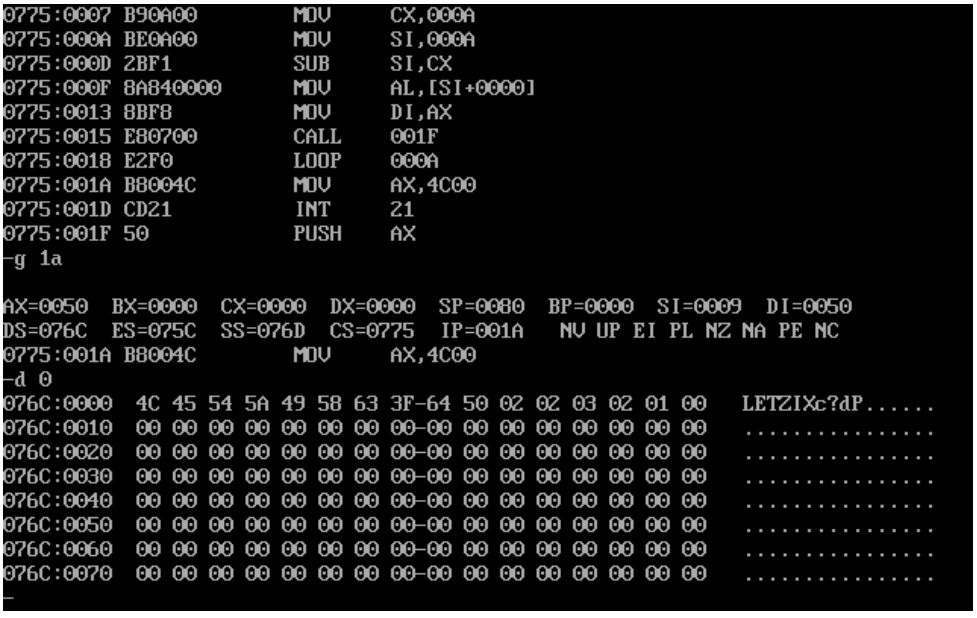
1. 复习教材中子程序设计的相关内容。

2. 编写程序完成下列功能：设有 10 个学生的成绩分别为 76、69、84、90、73、88、99、63、100 和 80 分。试编制一个子程序统计 60-69 分，70-79 分，80-89 分，90-99 分及 100 分的人数，并分别存放到 S6、S7、S8、S9 和 S10 单元中。

三、实验结果（截图）

1. 程序

S6~S10 分别存在 076C:000A 处，S6=2、S7=2、S8=3、S9=2、S10=1，与题目要求相符，正确。



四、实验报告要求（习题）

1. 程序 **1**——统计分数

```
DATA SEGMENT
    SCORE DB 76, 69, 84, 90, 73, 88, 99, 63, 100, 80
```

```
    S6 DB 0
    S7 DB 0
    S8 DB 0
    S9 DB 0
    S10 DB 0
DATA ENDS

STACK SEGMENT STACK
    DB 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME cs:CODE, ds:DATA, ss:STACK
START:
main PROC FAR
    ; 设置段寄存器
    mov ax, DATA
    mov ds, ax
    xor ax, ax
    ; 循环 10 次
    mov cx, 10
    ; 根据循环次数得到分数数组的偏移量 ax
AGAIN:
    mov si, 10
    sub si, cx
    ; 通过寄存器 DI 传参
    mov al, SCORE[si]
    mov di, ax
    call calcu
    loop AGAIN
    ; 退出代码
    mov ax, 4C00H
    int 21H
main ENDP

calcu PROC NEAR
    ; 保护寄存器
```

```
    push ax
    push bx
    push cx
    push dx
    ; 取得参数到 ax 中
    mov ax, di
    cmp ax, 100
    jne L1
    inc S10
    jmp DONE
L1: ; 90~99 分
    cmp ax, 90
    jb L2
    inc S9
    jmp DONE
L2: ; 80~89 分
    cmp ax, 80
    jb L3
    inc S8
    jmp DONE
L3: ; 70~79 分
    cmp ax, 70
    jb L4
    inc S7
    jmp DONE
L4: ; 60~69 分
    cmp ax, 60
    jb DONE
    inc S6
DONE:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
calcu ENDP
```

```
CODE ENDS
    END START
```

2. 寄存器的功能说明

在主过程 main 中，cx 共 loop 指令使用，以进行循环，si 用于进行基址变址寻址，di 用于与子过程 calcu 传递参数。在子过程 calcu 中，ax 用于存放欲判断的数。

3. 查看程序的运行结果

使用 debug 调试程序，用 g 运行后，使用 d 命令查看 DS:0 处内存单元的内容即可。

4. 主程序和子程序形式上的异同

主程序一般标识为 FAR，因为相对于 DOS 系统来说，该程序显然不处于同一个代码段中，DOS 系统 call 主程序就是以 FAR 的形式调用的。子程序可以与主程序写在同一个代码段中，使用 NEAR 进行标识。子程序往往在开头和结尾需要进行保护、恢复寄存器的状态，并且使用 ret 进行返回，以恢复 CS 与 IP 寄存器的内容。

5. 使用的参数传递方法

本实验中所写程序，使用的是寄存器传参的方法，使用 di 寄存器进行主程序和子程序之间的参数传递。入口参数是 di 寄存器，出口参数是 S6~S10 内存单元。

6. 使用不同的参数传递方法

将原来的代码改为通过变量，即某个内存单元进行传参，修改完成后的代码及运行结果如下，修改部分已用蓝色标出。

经过对 c 语言编译器 gcc 的编译结果的观察，发现在大部分情况下使用的都是寄存器传参，各个寄存器使用的优先级为 di、si、dx、cx、rxb。

```
DATA SEGMENT
    SCORE DB 76, 69, 84, 90, 73, 88, 99, 63, 100, 80
    S6 DB 0
    S7 DB 0
    S8 DB 0
    S9 DB 0
    S10 DB 0
    TMP DB 0
DATA ENDS
```

```
STACK SEGMENT STACK
    DB 128 DUP(0)
STACK ENDS

CODE SEGMENT
    ASSUME cs:CODE, ds:DATA, ss:STACK
START:
main PROC FAR
    ; 设置段寄存器
    mov ax, DATA
    mov ds, ax
    xor ax, ax
    ; 循环 10 次
    mov cx, 10
    ; 根据循环次数得到分数数组的偏移量 ax
AGAIN:
    mov si, 10
    sub si, cx
    ; 通过变量传参
    mov al, SCORE[si]
    mov TMP, al
    call calcu
    loop AGAIN
    ; 退出代码
    mov ax, 4C00H
    int 21H
main ENDP

calcu PROC NEAR
    ; 保护寄存器
    push ax
    push bx
    push cx
    push dx
    ; 取得参数到 ax 中
    mov al, TMP
```

```
    mov ah, 0
    cmp ax, 100
    jne L1
    inc S10
    jmp DONE
L1: ; 90~99 分
    cmp ax, 90
    jb L2
    inc S9
    jmp DONE
L2: ; 80~89 分
    cmp ax, 80
    jb L3
    inc S8
    jmp DONE
L3: ; 70~79 分
    cmp ax, 70
    jb L4
    inc S7
    jmp DONE
L4: ; 60~69 分
    cmp ax, 60
    jb DONE
    inc S6
DONE:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
calcu ENDP

CODE ENDS
    END START
```

运行结果如下

```
0775:0005 33C0          XOR     AX,AX
0775:0007 B90A00        MOV     CX,000A
0775:000A BE0A00        MOV     SI,000A
0775:000D 2BF1          SUB     SI,CX
0775:000F 8A840000      MOV     AL,[SI+0000]
0775:0013 A20F00        MOV     [000F],AL
0775:0016 E80700        CALL    0020
0775:0019 E2EF          LOOP    000A
0775:001B B8004C        MOV     AX,4C00
0775:001E CD21          INT     21
-g 1b

AX=0050  BX=0000  CX=0000  DX=0000  SP=0080  BP=0000  SI=0009  DI=0000
DS=076C  ES=075C  SS=076D  CS=0775  IP=001B   NV UP EI PL NZ NA PE NC
0775:001B B8004C        MOV     AX,4C00
-d 0
076C:0000  4C 45 54 5A 49 58 63 3F-64 50 02 02 03 02 01 50   LETZIXc?dP.....P
076C:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

7. 遇到的问题及解决

在尝试使用堆栈进行传参的过程中，发现使用堆栈传参的这种方式进行入口参数的操作时，由于函数调用 call 会将 IP 压入栈，若是 FAR 标识的子过程，会将 CS 也压入栈，因此需要在子过程中进行额外的处理以取得主过程中压入栈的参数。

五、个人体会与总结

本次实验的代码由于逻辑较为简单，直接使用汇编语言进行编写，一开始直接使用了寄存器进行传参，后面将代码改为使用变量(即某内存单元)进行传参。子过程对于代码的编写非常重要，可以将整个代码进行逻辑上的模块划分，便于调试和修改。