华 中 科 技 大 学

课 程 实 验 报 告

课程名称：汇编语言程序设计实验

实验名称：实验二分支程序、循环程序的设计

实验时间：**2019-9-30，19：00-22：20**

实验地点：南一楼

指导教师：鲁宏伟

专业班级：信息安全1802班

学 号：U201814864

姓 名：师天硕

报告日期：2019年9月30日

**成绩评定**

|  |  |  |
| --- | --- | --- |
| 实验完成质量（70分） | 报告撰写质量（30分） | 总成绩 |
| 实验步骤清晰、详细、深入，实验记录真实完整等 | 报告规范、完整、通顺、详实 |
|  |  |  |

**目 录**

[1实验目的 1](#_Toc20861972)

[2实验内容 1](#_Toc20861973)

[2.1 学生成绩查询程序 1](#_Toc20861974)

[2.1.1 实验说明 1](#_Toc20861975)

[2.1.2 功能一：提示并输入学生姓名 1](#_Toc20861976)

[2.1.3功能二：以学生姓名查询有无该学生 2](#_Toc20861977)

[2.1.4功能三：计算所有学生的平均成绩 2](#_Toc20861978)

[2.1.5功能四：将功能二查到的学生的平均成绩进行等级判断，并显示判断结果 2](#_Toc20861979)

[3实验过程 2](#_Toc20861980)

[3.1开发环境 2](#_Toc20861981)

[3.2设计思想 3](#_Toc20861982)

[3.2.1功能模块分析 3](#_Toc20861983)

[3.2.2 数据结构分析 3](#_Toc20861984)

[3.2.3算法设计 4](#_Toc20861985)

[3.3流程图 4](#_Toc20861986)

[3.4源程序 4](#_Toc20861987)

[3.5实验步骤 7](#_Toc20861988)

[3.6实验结果及分析 7](#_Toc20861989)

[4心得与体会 8](#_Toc20861990)

# 1实验目的

1. 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；
2. 加深对转移指令及一些常用的汇编指令的理解；
3. 加深对常用DOS功能调用指令的理解，进一步熟悉TD的使用。

# 2实验内容

## 2.1 学生成绩查询程序

设计实现一个学生成绩查询的程序

### 2.1.1 实验说明

在以BUF为首地址的字节数据存储区中，存放着N个学生的课程成绩表（百分制），每个学生的相关信息包括：10名（占10个字节），结束符为数值0；语文成绩（1个字节）；数学成绩（1个字节）；英语成绩（1个字节）；平均成绩（1个字节）。

例如：

N    equ 1000

BUF  db ‘ZhangSan’, 0, 0

;学生姓名，不足10个字节的部分用0填充

db 100, 85, 80, ？

;平均成绩还未计算

db ‘LiSi’, 6 dup(0)

db 80, 100, 70, ？

db N-3 dup(‘TempValue ’, 0, 80, 90, 95, ?)

; 除了已经定义了的学生信息及成绩表外，其他学生的暂时成绩假定是一样的

db ‘WangWu’, 4 dup(0)

;最后一个必须修改为自己名字的拼音

db 85, 85, 100, ？

### 2.1.2 功能一：提示并输入学生姓名

1. 使用9号DOS系统功能调用，提示用户输入学生姓名。
2. 使用10号DOS系统功能调用，输入学生姓名。输入的学生姓名字符串存放在以in\_name为首地址的存储区中。
3. 若只是输入了回车，则回到“（1）”处重新提示并输入；若仅仅输入字符q，则程序退出，否则，准备进入下一步处理。

### 2.1.3功能二：以学生姓名查询有无该学生

1. 使用循环程序结构，在成绩表中查找该学生。
2. 若未找到，提示用户该学生不存在，并回到“功能一（1）”的位置，提示并重新输入姓名。
3. 若找到，则将该学生课程成绩表的起始偏移地址保存到POIN字变量中。

提示：字符串比较时，当采用输入串的长度作为循环次数时，若因循环次数减为0而终止循环，则还要去判断成绩表中名字串的下一个字符是否是结束符0，若是，才能确定找到了。

### 2.1.4功能三：计算所有学生的平均成绩

1. 使用算术运算相关指令计算并保存每一个学生的平均成绩。
2. 平均成绩计算公式：（A\*2+B+C/2）/ 3.5，即将语文成绩A乘以权重2、英语成绩C除以权重2后，再与数学成绩B一起求和，再计算该生的平均成绩。要求避免溢出。

提示：使用循环程序结构，注意寻址方式的灵活运用。把小数3.5转换成分数后再运算以避免使用浮点指令。

### 2.1.5功能四：将功能二查到的学生的平均成绩进行等级判断，并显示判断结果

1. 平均成绩等级显示方式：若平均成绩大于等于90分，显示“A”；大于等于80分，显示“B”；大于等于70分，显示“C”；大于等于60分，显示“D”；小于60分，显示“F”。

提示：使用分支程序结构，采用2号DOS系统功能调用显示结果。

1. 使用转移指令回到“功能一（1）”处（提示并输入姓名）。

# 3实验过程

## 3.1开发环境

本次实验中使用的环境配置如下：

1. 虚拟机版本：DOSBox v0.74-3

打开后自动执行以下命令，挂载并进入目录，设置环境变量

**mount x C:\80x86-asm-learning**

**set PATH=%PATH%;x:\usr;x:\usr\masm615\BIN;**

**x:**

（2）编译器及其版本：masm6.11

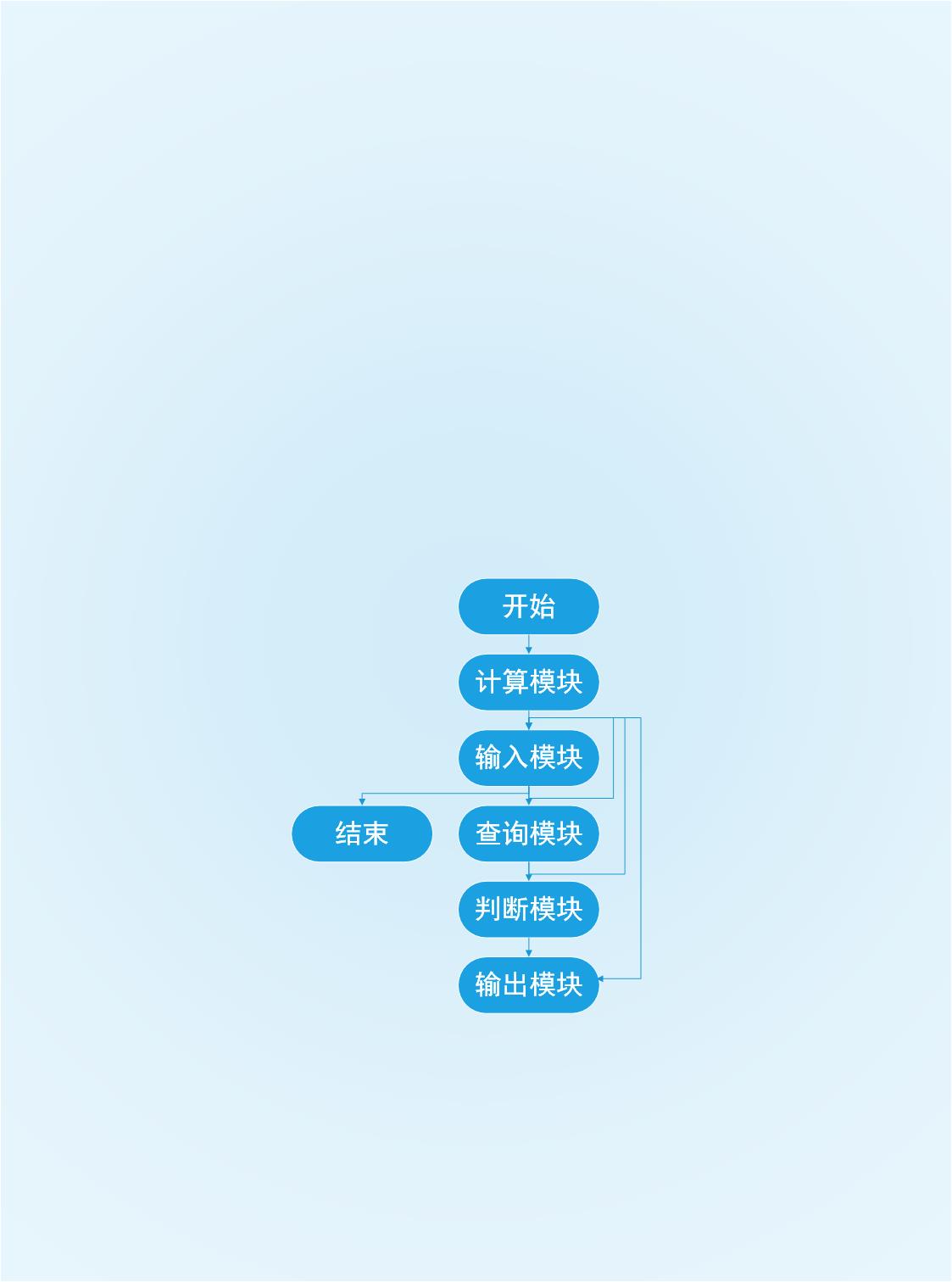
（3）调试工具：TD

（4）编程环境：Visual Studio Code

## 3.2设计思想

### 3.2.1功能模块分析

本程序要求支持4个功能，我们将其对应的大致分为5个部分：输入部分，查找部分，成绩计算部分，等级判断部分和输出部分。仔细分析相应功能后，发现成绩计算部分和其它部分耦合较少，在进入程序时统一计算即可。而其它四个部分是明显的顺序结构。细节上，在输入后需要由输入内容控制跳转，而在查找之前需要对输入的数据进行处理。综上，程序大致模块如图1所示。



**图1 程序模块图**

### 3.2.2 数据结构分析

程序所用程序记录在同一数据段中，主要分三段。

1. 程序运行中用于指示的字符串，用$号结尾的字节床表示。包括以下四条。

'Please input the name of the student: $'

'Student not found, please re-input ... ',0dh,0ah,'$'

'the level of the student are: $'

'exit!',0dh,0ah,'$'

1. 程序运行时的缓存，用以存储读入的查询学生姓名。由头上的meta信息和之后的数据你组成。一共12字节，第一个字节存缓冲区长度，第二个字节存缓冲区读入字节数，第3到12个字节存放读入的字符。
2. 学生的信息记录，每条记录有14字节，由十字节长度的姓名，和4字节的成绩组成，如图2所示。



**图2学生记录内存分布示意图**

### 3.2.3算法设计

总体上，所有的字符串输出使用 int 21H 的 09H 号中断，需要输出的内容全部保存在数 据段；所有需要输出的字符使用 int 21H 的 02H 号中断，需要输出的字符使用立即数表 示；需要输入数据则使用 int 21H 的 0AH 号中断，需要输入的数据存在数据区的数据缓 冲区中。

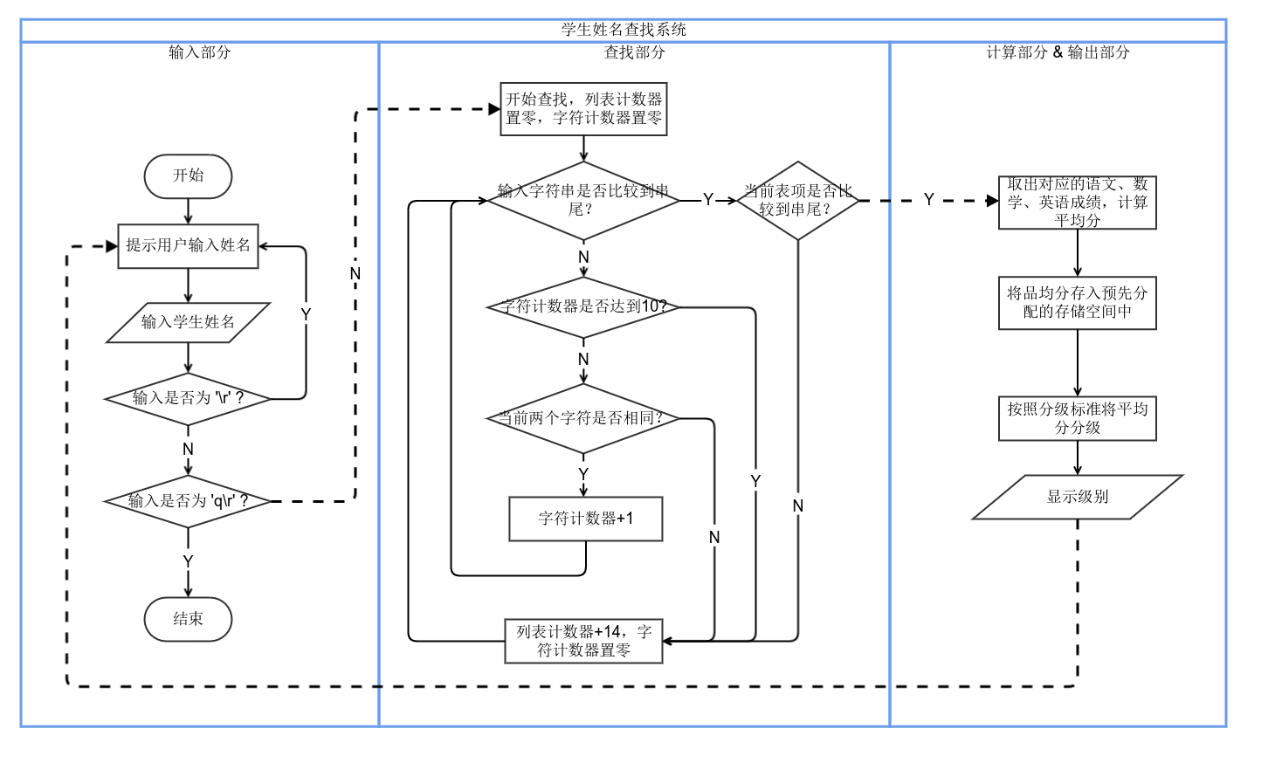
计算部分，发现计算平均成绩公式的特性，计算过程可以优化为一系列\*2（左移）和相加操作，最后利用div指令计算平均成绩，存入内存。

输入部分，对用户输入的数据进行判断，若为回车字符 (DH)，则判定为非法输入，提示 重新输入；若为’q\n’̊(71H, 0DH)，则判断为退出，直接终止程序；否则判断为姓名，进入查找部分继续执行。

查找部分，利用二重循环，遍历每条学生记录。在嵌套循环中：对姓名的每一个字进行逐字匹配，一旦发现失配则匹配不成功，进入下一个循环；若同时匹配到结尾，表明匹配成功，跳转进入判断部分；若所有的姓名均遍历完成仍没有匹配成功者，则表明姓名不存在，重新进入输入部分。

## 3.3流程图

如图3所示



**图3程序流程图**

## 3.4源程序

.386

BR macro

    mov ah, 02h     ; \r

    mov dl, 0dh

    int 21h

    mov ah, 02h     ; \n

    mov dl, 0ah

    int 21h

endm

stackseg segment use16 stack

    dw 10 dup(0)

stackseg ends

dataseg segment use16

    s1 db 'Please input the name of the student: $'

    s2 db 'Student not found, please re-input ... ',0dh,0ah,'$'

    s3 db 'the level of the student are: $'

    s4 db 'exit!',0dh,0ah,'$'

    in\_name\_head db 10, ?

    in\_name db 10 dup(0)

    N equ 10

    s\_name equ 0

    s\_mx\_name equ 9

    s\_a equ 10

    s\_b equ 11

    s\_c equ 12

    s\_avg equ 13

    s\_len equ 14

    BUF db 'ZhangSan', 2 dup(0) ;学生姓名，不足10个字节的部分用0填充

    db 100, 85, 80, ?   ;平均成绩还未计算

    db 'LiSi', 6 dup(0)

db 80, 100, 70, ?

db 'mxtest', 4 dup(0)

    db 100, 100, 100, ?

db 'mntest', 4 dup(0)

db 0, 0, 0, ?

    db N-5 dup('TempValue', 0, 80, 90, 95, ?); 除了已经定义了的学生信息及成绩表外，其他学生的暂时成绩假定是一样的

    db 'stonepage', 0    ;最后一个必须修改为自己名字的拼音

    db 85, 85, 100, ?

    poin dw 0

dataseg ends

codeseg segment use16

    assume ss:stackseg, ds:dataseg, cs:codeseg

start:

    mov ax, dataseg

    mov ds, ax

    mov ax, stackseg

    mov ss, ax

    mov sp, 20

avg\_all:

    mov bx, offset BUF

    mov cx, N

    avg\_iter:

        xor ah, ah

        mov al, s\_a[bx]

        shl ax, 1

        add al, s\_b[bx]

        adc ah, 0

        shl ax, 1

        add al, s\_c[bx]

        adc ah, 0

        mov dl, 7

        div dl

        mov s\_avg[bx], al

        add bx, s\_len

    loop avg\_iter

input:

    mov ah, 09h

    mov dx, offset s1

    int 21h

    mov ah, 0ah

    mov in\_name\_head, s\_mx\_name+1

    lea dx, in\_name\_head

    int 21h

    BR

    ;0dh=='\n' 71h=='q'

    cmp in\_name, 0dh

    jz  input

    cmp in\_name, 71h

    jnz filter

    cmp in\_name+01h, 0dh

    jnz filter

    jmp quit

filter:

    mov bl, in\_name\_head[1]

    xor bh, bh

    mov byte ptr in\_name[bX], 0

find:

    mov cx, n

    mov bx, offset BUF

    find\_iter:

        lea si, s\_name[bx]

        mov di, offset in\_name

        push bx

        push cx

        mov cl, in\_name\_head[1]

        xor ch, ch

        strcmp\_iter:

            mov dl, [si]

            cmp dl, [di]

            jnz next

            inc si

            inc di

        loop strcmp\_iter

        jmp found

    next:

        pop cx

        pop bx

        add bx, s\_len

    loop find\_iter

not\_found:

    mov ah, 09h

    mov dx, offset s2

    int 21h

    jmp input

found:

    mov poin, bx

    mov al, s\_avg[bx]

outs3:

    mov dx, offset s3

    mov ah, 09H

    int 21H

judge:

    mov dl, 'A'

    cmp al, 90

    jge outlevel

    inc dl

    cmp ax, 80

    jge outlevel

    inc dl

    cmp ax, 70

    jge outlevel

    inc dl

    cmp ax, 60

    jge outlevel

    add dl, 2

outlevel:

    mov ah, 02h

    int 21h

    BR

    jmp input

quit:

    mov ah, 09h

    mov dx, offset s4

    int 21h

    mov ax, 4c00h

    int 21h

codeseg ends

end start

## 3.5实验步骤

1.阅读代码，查看资料，确定程序功能

2.准备上机实验环境，编写代码。

3.经MASM汇编，LINK连接后并且确认源程序正确无误后，直接运行，观察程序对输入的处理和输出有无明显错误。

4.测试各类输入，覆盖所有情况，

5.修改数据段内数据，重新编译链接进行测试。

## 3.6实验结果及分析

1. 编译完成，无报错。直接执行二进制程序，出现提示字符串’Please input the name of the student:’

2. 输入一个列表中已经存在的姓名stonepage程序显示该学生平均分所对应的等级， 并继续提示用户输入学生姓名。

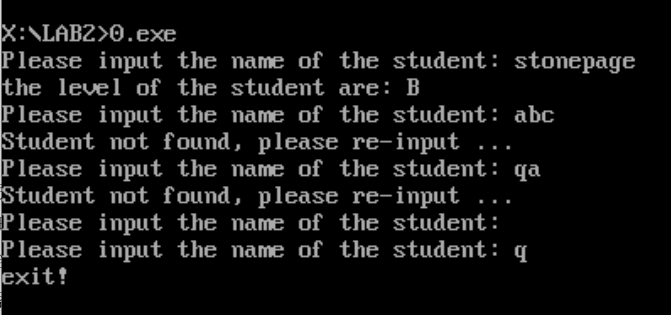
3. 输入一个列表中不存在的姓名abc，程序提示该学生不存在，并要求重新输入。

4. 输入一个列表中不存在，且以字母q为前缀的姓名qa，程序提示该学生不存在，并要求重新输入。

5. 直接输入回车，程序再次要求输入学生姓名。

6. 输入 q，程序直接退出。

1~6 步输出结果如图4所示，初步表明程序能够正常运行。



**图4初步测试记录**

7.测试计算部分：使用TD调试器，程序跳转到计算数据段中对应测试数据的位置。测试数据如下。

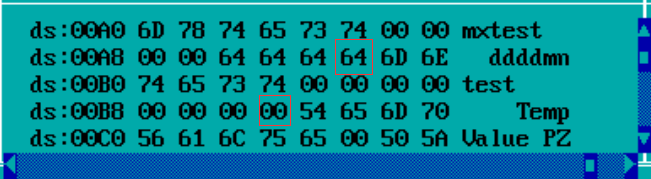
db 'mxtest', 4 dup(0)

    db 100, 100, 100, ?

db 'mntest', 4 dup(0)

db 0, 0, 0, ?

单步执行程序，观察平均分的计算以及存储是否正常运行，级别判断是否正确以及程序是否溢出。测试表明，即使总分数全部取最大，即 100,100,100，在执行加权以后总分为 350，没有超过一个16位寄存器的表示范围，而乘法的积与除法的被除数均为16位寄存器，因此不会溢出。测试后相关平均分计算的内存数据如图，答案正确



**图5极端数据测试后的内存**

# 4心得与体会

本次实验要完成一个完整的小程序。从分析问题，设计算法和数据结构，到编码和调试，经历了用汇编语言开发的完整流程。

一方面，这个程序让我几乎使用了迄今为止学到的所有汇编语法，这让我对汇编语言的各种指令、寻址方式等更加熟悉。在实际的编程中，出现的bug让我必须关注平日学习中有意无意忽略的细节，让我对指令和语法有更深的理解。与此同时我也更加习惯，在有限的寄存器下进行编程，只有较为简单的流程控制，等汇编语言这种极低抽象的语言特性。

另一方面，在不断的对错误的调试中，我对Turbo Debugger 的各种便于调试的功能和调试的方法（如打断点，监视，查找，跳 转，录制宏等）越发熟悉，举例来讲，此次调试中我就了解到了F7 Trace和F8 Step单步指令的区别。这些都大大提高了我的综合能力。