

目录

1	实验目的与要求.....	1
2	实验内容.....	1
3	实验过程.....	3
3.1	任务 1	3
3.1.1	实验步骤	3
3.1.2	实验记录与分析	3
3.1.3	总结分析	9
3.1.4	实验收获	10
3.2	任务 2	11
3.2.1	源程序录入	11
3.2.2	实验步骤	11
3.2.3	实验记录与分析	12
3.2.4	实验收获	15
3.3	任务 3	16
3.3.1	实验步骤	16
3.3.2	实验记录与分析	16
3.4	任务四	20
3.4.1	程序设计思想及存储单元分配	20
3.4.2	流程图	21
3.4.3	源程序	24
3.4.4	实验步骤描述	27
3.4.5	实验记录与分析	28
4	总结与体会	32
	参考文献	33

汇编语言程序设计实验报告

1 实验目的与要求

本次实验的主要目的与要求有下面 6 点，所有的任务都会围绕这 6 点进行，希望大家事后检查自己是否达到这些目的与要求。

- (1) 掌握汇编源程序编辑工具、汇编程序、连接程序、调试工具 TD 的使用；
- (2) 理解数、符号、寻址方式等在计算机内的表现形式；
- (3) 理解指令执行与标志位改变之间的关系；
- (4) 熟悉常用的 DOS 功能调用；
- (5) 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；
- (6) 加深对转移指令及一些常用的汇编指令的理解。

2 实验内容

任务 1: 《80X86 汇编语言程序设计》教材中 P31 的 1.14 题。

要求：

- (1) 直接在 TD 中输入指令，完成两个数的求和求差的功能。求和/差后的结果放在(AH)中。
- (2) 请事先指出执行指令后(AH)、标志位 SF、OF、CF、ZF 的内容。
- (3) 记录上机执行后的结果，与（2）中对应的内容比较。
- (4) 求差运算中，若将 A、B 视为有符号数，且 $A > B$ ，标志位有何特点？
若将 A、B 视为无符号数，且 $A > B$ ，标志位又有何特点？

任务 2: 《80X86 汇编语言程序设计》教材中 P45 的 2.3 题。

要求：

- (1) 分别记录执行 MOV CX, 10 和 INT 21H 之前的(BX), (BP), (SI), (DI) 各是多少。
- (2) 记录程序执行到退出之前数据段开始 40 个字节的内容，程序运行结果与设想的一致？
- (3) 在标号 LOPA 前加上一段程序，实现新的功能：显示提示信息“Press any key to begin!”，在按了一个键之后继续执行 LOPA 处的程序。

任务 3: 《80X86 汇编语言程序设计》教材中 P45 的 2.4 题的改写。

要求：

- (1) 对数据段中变量访问时所用到的寻址方式中的寄存器改成 32 位寄存器。
- (2) 内存单元中数据的访问采用变址寻址方式。
- (3) 记录程序执行到退出之前数据段开始 40 个字节的内容，程序运行结果是否与设想一致？
- (4) 在 TD 代码窗口中观察并记录机器指令代码在内存中的存放形式，并与 TD 中提供的反汇编语句及自己编写的源程序语句进行对照，也与任务 2 做对比。（相似语句记录一条即可，重点理解机器码与汇编语句的对应关系，尤其注意操作数寻址方式的形式）。

汇编语言程序设计实验报告

(5) 观察连续存放的二进制串在反汇编成汇编语言语句时,从不同字节位置开始反汇编,结果怎样?理解 IP/EIP 指明指令起始位置的重要性。

操作提示:要让 TD 从任意指定地址开始反汇编,需要使用 TD 在代码显示区的 Goto 功能,即鼠标选中代码显示区,点击右键将显示带有 Goto 的菜单,选中 Goto 菜单项,输入 CS:XXXX 即可(XXXX 是你希望录入的偏移地址)。

任务 4: 设计实现一个学生成绩查询的程序。

1、实验背景

在以 BUF 为首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:姓名(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)。

2、功能一:提示并输入待查询成绩的学生姓名

(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。输入的姓名字符串放在以 in_name 为首址的存储区中。

(3) 若只是输入了回车,则回到“(1)”处重新提示与输入;若仅仅输入字符 q,则程序退出,否则,准备进入下一步处理。

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,就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(这样做是为了避免输入的名字仅仅是数据段中所定义名字的子集的误判情况)。

4、功能三:计算所有学生的平均成绩

使用算数运算相关指令计算并保存每一个学生的平均成绩。

平均成绩计算公式: $(A*2+B+C)/3.5$,即将语文成绩 A 乘以权重 2、英语成绩 C 除以权重 2 后,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。

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。

(2) 使用转移指令回到“功能一(1)”处(提示并输入姓名)。

汇编语言程序设计实验报告

3 实验过程

3.1 任务 1

3.1.1 实验步骤

1. 准备上机实验环境。
2. 在 TD 窗口当前光标下输入各运算式对应的两个二进制数字加减法对应的汇编代码；
 - (1) 一小题加法：MOV AH, 0110011B MOV AL, 1011010B ADD AH, AL
 - (2) 一小题减法：MOV AH, 0110011B MOV AL, 1011010B ADD AH, AL
 - (3) 二小题加法：MOV AH, -0101001B MOV AL, -1011101B ADD AH, AL
 - (4) 二小题减法：MOV AH, -0101001B MOV AL, -1011101B SUB AH, AL
 - (5) 三小题加法：MOV AH, 1100101B MOV AL, -1011101B ADD AH, AL
 - (6) 三小题减法：MOV AH, 1100101B MOV AL, -1011101B SUB AH, AL
3. 确定当前 CS:IP 指向的是自己输入的第一条指令，单步执行三次，观察各寄存器的内容和标志寄存器的结果。
4. 重复这个过程直到 6 个表达式全部计算完毕，比较结果

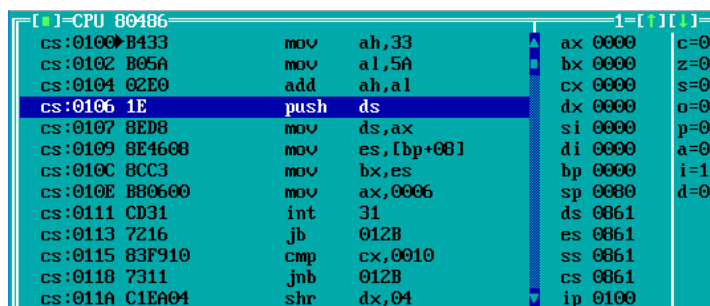
3.1.2 实验记录与分析

1. 实验环境条件：

操作系统：Windows 10 下的虚拟 DOSBox 0.74

TD: 5.0
2. 在 TD 的代码窗口中的当前光标下输入第一个运算式对应的两个 8 位数值对应的指令语句
 - (1) 一小题加法：MOV AH, 0110011B; MOV AL, 1011010B; ADD AH, AL;

<1>观察代码区显示的内容与自己输入字符之间的关系：



[] CPU 80486		1-[] []	
cs:0100 B433	mov ah,33	ax 0000	c=0
cs:0102 B05A	mov al,5A	bx 0000	z=0
cs:0104 02E0	add ah,al	cx 0000	s=0
cs:0106 1E	push ds	dx 0000	o=0
cs:0107 8ED8	mov ds,ax	si 0000	p=0
cs:0109 8E4608	mov es,[bp+08]	di 0000	a=0
cs:010C 8CC3	mov bx,es	bp 0000	i=1
cs:010E B80600	mov ax,0006	sp 0000	d=0
cs:0111 CD31	int 31	ds 0061	
cs:0113 7216	jb 012B	es 0061	
cs:0115 83F910	cmp cx,0010	ss 0061	
cs:0118 7311	jnb 012B	cs 0061	
cs:011A C1EA04	shr dx,04	ip 0100	

图 1-1 TD 代码区

观察结果：

- I. td 中显示的数字为自己输入数字的十六进制表示，输入的字母都使用了小写表示。
- II. 进一步了解了 TD 窗口的布局，主窗口显示的是 CS 段所指示的内容，可以直接进行修

汇编语言程序设计实验报告

改。左边是各寄存器的内容，最左边是各个标志寄存器的值。下方是内存中的值。
<2>确定 CS:IP 指向的是自己输入的第一条指令的位置，若没有指向自己的命令，则先改变 CS:IP 的值。单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

表 1-1 一小题加法标志寄存器结果预测

预测结果				
AH	SF	OF	CF	ZF
1000 1101B (8DH)	1	1	0	0

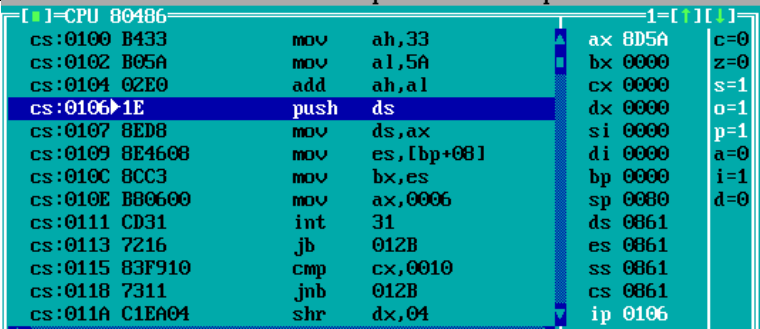


图 1-2 一小题加法最终结果

表 1-2 一小题加法标志寄存器实际结果

实际结果				
AH	SF	OF	CF	ZF
1000 1101B (8DH)	1	1	0	0

分析结果：

两个数字相加后其结果第八位为 1，若按照有符号数来看他小于 0，SF 也置 1。当作无符号数来看的时候，没有发生进位，所以 CF 置 0，结果不等于 0，所以 ZF 置 0。

在程序执行过程中我们可以观察到 IP 值的不断变化，CS:IP 指向了他接下来要执行的语句。TD 这个工具让人更加直观了解了这个机制。

(2)一小题减法：MOV AH,0110011B; MOV AL,1011010B; ADD AH,AL;

观察代码区显示的内容与自己输入字符之间的关系：

汇编语言程序设计实验报告

```

CPU 80486
cs:0106 B433      mov ah,33      ax 8D5A  c=0
cs:0108 B05A      mov al,5A      bx 0000  z=0
cs:010A 2AE0      sub ah,al      cx 0000  s=0
cs:010C 8CC3      mov bx,es      dx 0000  o=0
cs:010E B80600     mov ax,0006     si 0000  p=0
cs:0111 CD31      int 31         di 0000  a=0
cs:0113 7216      jb 012B        bp 0000  i=1
cs:0115 83F910     cmp cx,0010     sp 0000  d=0
cs:0118 7311      jnb 012B       ds 0061
cs:011A C1EA04     shr dx,04       es 0061
cs:011D C1E10C     shl cx,0C       ss 0061
cs:0120 0BD1      or dx,cx        cs 0061
cs:0122 8B4606     mov ax,[bp+06]  ip 0106

```

图 1-3 一小题减法代码

确定 CS:IP 指向的是自己输入的第一条指令的位置，若没有指向自己的命令，则先改变 CS:IP 的值。单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

表 1-3 一小题减法标志寄存器结果预测

预测结果				
AH	SF	OF	CF	ZF
1101 1001B (D9H)	1	0	1	0

```

CPU 80486
cs:0106 B433      mov ah,33      ax D95A  c=1
cs:0108 B05A      mov al,5A      bx 0000  z=0
cs:010A 2AE0      sub ah,al      cx 0000  s=1
cs:010C 8CC3      mov bx,es      dx 0000  o=0
cs:010E B80600     mov ax,0006     si 0000  p=0
cs:0111 CD31      int 31         di 0000  a=1
cs:0113 7216      jb 012B        bp 0000  i=1
cs:0115 83F910     cmp cx,0010     sp 0000  d=0
cs:0118 7311      jnb 012B       ds 0061
cs:011A C1EA04     shr dx,04       es 0061
cs:011D C1E10C     shl cx,0C       ss 0061
cs:0120 0BD1      or dx,cx        cs 0061
cs:0122 8B4606     mov ax,[bp+06]  ip 010C

```

图 1-4 一小题减法结果

表 1-4 一小题减法标志寄存器实际结果

实际结果				
AH	SF	OF	CF	ZF
1101 1001B (D9H)	1	0	1	0

分析结果：

两个数字相减后其结果第八位为 1，若按照有符号数来看，小于 0，所以 SF 置 1。当错无符号数来看的时候，计算过程发生进位，所以 CF 置 1，结果不等于 0，所以 ZF 置 0。而结果小于 -127，所以没有发生溢出，OF 置 0。

汇编语言程序设计实验报告

(3) 二小题加法: MOV AH, -0101001B MOV AL, -1011101B ADD AH, AL

```

[ ] CPU 80486
cs:010C B4D7 mov ah,D7
cs:010E B0A3 mov al,A3
cs:0110 02E0 add ah,al
cs:0112 317216 xor [bp+si+16],si
cs:0115 83F910 cmp cx,0010
cs:0118 7311 jnb 012B
cs:011A C1EA04 shr dx,04
cs:011D C1E10C shl cx,0C
cs:0120 0BD1 or dx,cx
cs:0122 8B4606 mov ax,[bp+06]
cs:0125 1F pop ds
cs:0126 5D pop bp
cs:0127 4D dec bp
ax D95A c=1
bx 0000 z=0
cx 0000 s=1
dx 0000 o=0
si 0000 p=1
di 0000 a=1
bp 0000 i=1
sp 0080 d=0
ds 0061
es 0061
ss 0061
cs 0061
ip 010C
    
```

图 1-5 二小题加法代码

确定 CS:IP 指向的是自己输入的第一条指令的位置, 若没有指向自己的命令, 则先改变 CS:IP 的值。单步执行三次, 观察寄存器内容的变化, 记录标志寄存器的结果。

表 1-5 二小题加法标志寄存器预测

预测结果				
AH	SF	OF	CF	ZF
0111 1010B (7AH)	0	1	1	0

```

[ ] CPU 80486
cs:010C B4D7 mov ah,D7
cs:010E B0A3 mov al,A3
cs:0110 02E0 add ah,al
cs:0112 317216 xor [bp+si+16],si
cs:0115 83F910 cmp cx,0010
cs:0118 7311 jnb 012B
cs:011A C1EA04 shr dx,04
cs:011D C1E10C shl cx,0C
cs:0120 0BD1 or dx,cx
cs:0122 8B4606 mov ax,[bp+06]
cs:0125 1F pop ds
cs:0126 5D pop bp
cs:0127 4D dec bp
ax 7AA3 c=1
bx 0000 z=0
cx 0000 s=0
dx 0000 o=1
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0080 d=0
ds 0061
es 0061
ss 0061
cs 0061
ip 0112
    
```

图 1-6 二小题加法结果

表 1-6 二小题标志寄存器结果

实际结果				
AH	SF	OF	CF	ZF
7AH	0	1	1	0

分析结果:

两个数字相加后其结果第八位为 0, 按照有符号数来看他大 0, SF 置 0. 无符号数来看的时候, 发生进位, 所以 CF 置 1, 果不等于 0, 所以 ZF 置 0。两负数相加结果为正数, 发生溢出, OF 为 1。

(4) 二小题减法 :MOV AH, -0101001B MOV AL, -1011101B SUB AH, AL

汇编语言程序设计实验报告

```

[ ]=CPU 80486
cs:0112 B4D7      mov     ah,D7      ax 7A93  c=1
cs:0114 B0A3      mov     al,A3      bx 0000  z=0
cs:0116 2AE0      sub     ah,al      cx 0000  s=0
cs:0118 7311      jnb     012B      dx 0000  o=1
cs:011A C1EA04     shr     dx,04      si 0000  p=0
cs:011D C1E10C     shl     cx,0C      di 0000  a=0
cs:0120 0BD1      or      dx,cx      bp 0000  i=1
cs:0122 8B4606     mov     ax,[bp+06]  sp 0080  d=0
cs:0125 1F        pop     ds      ds 0061
cs:0126 5D        pop     bp      es 0061
cs:0127 4D        dec     bp      ss 0061
cs:0128 CA0400     retf    0004     cs 0061
cs:012B 33D2      xor     dx,dx      ip 0112

```

图 1-7 二小题减法代码

确定 CS:IP 指向的是自己输入的第一条指令的位置，若没有指向自己的命令，则先改变 CS:IP 的值。单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

表 1-7 二小题减法标志寄存器结果预测

预测结果				
AH	SF	OF	CF	ZF
00110100B (34H)	0	0	0	0

```

[ ]=CPU 80486
cs:0112 B4D7      mov     ah,D7      ax 34A3  c=0
cs:0114 B0A3      mov     al,A3      bx 0000  z=0
cs:0116 2AE0      sub     ah,al      cx 0000  s=0
cs:0118 7311      jnb     012B ↓     dx 0000  o=0
cs:011A C1EA04     shr     dx,04      si 0000  p=0
cs:011D C1E10C     shl     cx,0C      di 0000  a=0
cs:0120 0BD1      or      dx,cx      bp 0000  i=1
cs:0122 8B4606     mov     ax,[bp+06]  sp 0080  d=0
cs:0125 1F        pop     ds      ds 0061
cs:0126 5D        pop     bp      es 0061
cs:0127 4D        dec     bp      ss 0061
cs:0128 CA0400     retf    0004     cs 0061
cs:012B 33D2      xor     dx,dx      ip 0118

```

图 1-8 二小题减法结果

表 1-8 二小题减法标志寄存器实际结果

实际结果				
AH	SF	OF	CF	ZF
00110100B (34H)	0	0	0	0

分析结果：

两个数字相加后其结果第八位为 0，按照有符号数来看他大 0，SF 置 0。无符号数来看的时候，没有进位，所以 CF 置 0，且不等于 0，所以 ZF 置 0。A>B 相减结果为正数，没有溢出，OF 为 0。

汇编语言程序设计实验报告

(5)三小题加法：MOV AH, 1100101B MOV AL, -1011101B ADD AH, AL

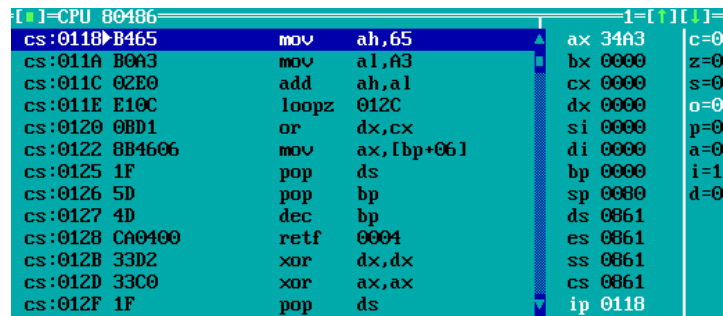


图 1-9 三小题加法代码

确定 CS:IP 指向的是自己输入的第一条指令的位置，若没有指向自己的命令，则先改变 CS:IP 的值。单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

表 1-9 三小题加法标志寄存器结果预测

预测结果				
AH	SF	OF	CF	ZF
00001000B (8H)	0	0	1	0

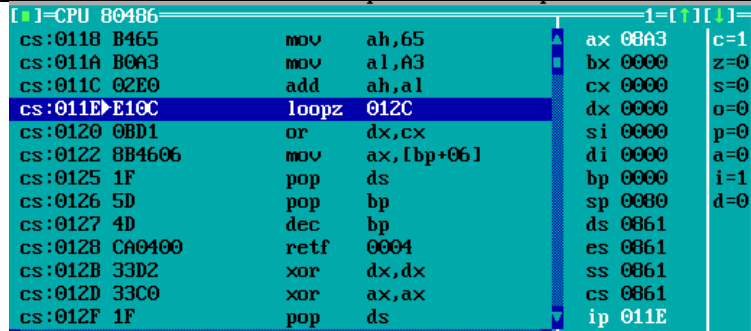


图 1-10 三小题加法结果

表 1-10 三小题加法标志寄存器结果预测

实际结果				
AH	SF	OF	CF	ZF
00001000B (8H)	0	0	1	0

分析结果：

两个数字相加后其结果第八位为 0，按照有符号数来看他大 0，SF 置 0。无符号数来看的时候，有进位，所以 CF 置 1，且不等于 0，所以 ZF 置 0。AB 相加结果为正数，没有溢出，OF 为 0。

(6)三小题减法：MOV AH, 1100101B MOV AL, -1011101B SUB AH, AL

汇编语言程序设计实验报告

```
[CPU 80486]
cs:011E B465    mov     ah,65
cs:0120 B0A3    mov     al,A3
cs:0122 2AE0    sub     ah,al
cs:0124 06      push    es
cs:0125 1F      pop     ds
cs:0126 5D      pop     bp
cs:0127 4D      dec     bp
cs:0128 CA0400  retf    0004
cs:012B 33D2    xor     dx,dx
cs:012D 33C0    xor     ax,ax
cs:012F 1F      pop     ds
cs:0130 5D      pop     bp
cs:0131 4D      dec     bp

ax 08A3  c=1
bx 0000  z=0
cx 0000  s=0
dx 0000  o=0
si 0000  p=0
di 0000  a=0
bp 0000  i=1
sp 0080  d=0
ds 0861
es 0861
ss 0861
cs 0861
ip 011E
```

图 1-11 三小题减法代码

表 1-11 三小题减法结果预测

预测结果				
AH	SF	OF	CF	ZF
11000010B (C2H)	1	1	1	0

```
[CPU 80486]
cs:011E B465    mov     ah,65
cs:0120 B0A3    mov     al,A3
cs:0122 2AE0    sub     ah,al
cs:0124 06      push    es
cs:0125 1F      pop     ds
cs:0126 5D      pop     bp
cs:0127 4D      dec     bp
cs:0128 CA0400  retf    0004
cs:012B 33D2    xor     dx,dx
cs:012D 33C0    xor     ax,ax
cs:012F 1F      pop     ds
cs:0130 5D      pop     bp
cs:0131 4D      dec     bp

ax C2A3  c=1
bx 0000  z=0
cx 0000  s=1
dx 0000  o=1
si 0000  p=0
di 0000  a=0
bp 0000  i=1
sp 0080  d=0
ds 0861
es 0861
ss 0861
cs 0861
ip 0124
```

图 1-12 三小题减法结果

表 1-12 三小题减法标志寄存器实际结果

实际结果				
AH	SF	OF	CF	ZF
11000010B (C2H)	1	1	1	0

分析结果：

两个数字相减后其结果第八位为 1，按照有符号数来看他小 0，SF 置 1。无符号数来看的时候，有借位，所以 CF 置 1，且不等于 0，所以 ZF 置 0。A>B 结果为负数，有溢出，OF 为 1。

3.1.3 总结分析

求差运算中，若将 A、B 视为有符号数，且 A>B，标志位有何特点？若将 A、B 视为无符号数，且 A>B，标志位有何特点？

答：若 A、B 为有符号数，且 A>B，分为下面几种情况：

汇编语言程序设计实验报告

1. $A > B > 0$ 此时, $A - B$ 的结果肯定 > 0 , 所以 $SF = 0$, 肯定不会溢出, 所以 $OF = 0$, CF 也等于 0, $ZF = 0$ (因为 AB 肯定不等)。

2. $A > 0 > B$ 此时, SF 、 ZF 的结果任然和上面相同, 但是结果可能溢出, 所以 OF 可能等于 1, CF 也可能等于 1. (例如: $111\ 1111B - (-111\ 1111B)$)。

3. $0 > A > B$ 此时, SF 肯定为 1, ZF 肯定为 0 (理由同上), OF 和 CF 也肯定为 0, 因为 $0 < A - B < A$ 肯定不会超过表示范围发生溢出, 而且 $A > B$ 也不会发生借位。

若 A 、 B 为无符号数, 这里我不太明白意思。计算机发生的永远是补码运算, 不知道怎么样就是当作无符号数?

3.1.4 实验收获

熟悉了标志位寄存器, 一开始我几个标志寄存器经常弄混, 现在基本没有这个现象了。也熟悉了 TD 的操作。

3.2 任务 2

3.2.1 源程序录入

```
.386
STACK SEGMENT USE16 STACK
    DB 200 DUP(0)
STACK ENDS

DATA SEGMENT USE16
    BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    BUF2 DB 10 DUP(0)
    BUF3 DB 10 DUP(0)
    BUF4 DB 10 DUP(0)
DATA ENDS

CODE SEGMENT USE16
    ASSUME CS, CODE, DS, DATA, SS, STACK
START:
    MOV AX, DATA
    MOV DS, AX
    MOV SI, OFFSET BUF1
    MOV DI, OFFSET BUF2
    MOV BX, OFFSET BUF3
    MOV BP, OFFSET BUF4
    MOV CX, 10
LOPA:
    MOV AL, [SI]
    MOV [DI], AL
    INC AL
    MOV [BX], AL
    ADD AL, 3
    MOV DS, [BP], AL
    INC SI
    INC DI
    INC BP
    INC BX
    DEC CX
    JNZ LOPA
    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

3.2.2 实验步骤

1. 检查 sublime 敲入计算机的程序是否抄错
2. 研究如何汇编链接一个源程序并且产生可执行文件
3. 编译链接源代码，调 BUG，直到程序跑通。
4. 研究如何设置断点，并且运行到断点处

汇编语言程序设计实验报告

5. 使用 TD 反汇编可执行文件, 在 MOV CX, 10 处打断点, 按下 F4 执行到该处, 观察 BX, BP, SI, DI 的值各是多少
6. 在 INT 21(程序结束之前)按下 F4, 程序执行到该处, 记录下 BX, BP, SI, DI 的值, 思考问什么会是这样的值
7. 记录程序执行到退出之前数据段开始 40 个字节的内容, 指出程序运行结果是否与设想的一致。
8. 熟悉 DOS 系统功能调用, 并检查自己三小题所写程序的正确性: 直接运行可执行文件, 观察输出是否与预想相同。

3.2.3 实验记录与分析

1. 检查自己录入程序是否与书中相同:
检查出三点错误, 自己在三个 offset 前忘加逗号
2. 查阅课本找到如何汇编链接一个源文件并生成可执行文件
查阅后直首先将文件保存在 DOS 的工作目录下, 然后使用 masm 编译, 使用 link 进行链接, 若程序没有 bug, 这样就生成了可执行文件。
3. 编译链接源文件, 直到没有 BUG

```
C:\>masm test
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [test.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
test.ASM(14): error A2105: Expected: colon
test.ASM(15): error A2062: Missing or unreachable CS
test.ASM(23): error A2062: Missing or unreachable CS
test.ASM(29): warning A4001: Extra characters on line

50034 + 463355 Bytes symbol space free

1 Warning Errors
3 Severe Errors
```

图 2-1 第一次编译程序

第一次编译时发现错误, 进过查阅资料, 研究代码, 将其修改直至无误。

```
C:\>masm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Source filename [.ASM]: work2
Object filename [work2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

49996 + 463393 Bytes symbol space free

0 Warning Errors
0 Severe Errors
```

图 2-2 编译程序直到无误

使用连接程序 LINK.EXE 将汇编生成的 FILENAME.OBJ 文件连接成执行文件。
即 LINK FILENAME;

汇编语言程序设计实验报告

```
C:\>link work2

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [WORK2.EXE]:
List File [NUL.MAP]:
Libraries [LIB]:

C:\>WORK2.EXE
```

图 2-3 链接程序

- 4. 研究打断点和执行到断点
查看课本附录部分有两种设置断点并且执行的方法：
1> 使用 F4 键。将鼠标移到该指令处按下 F4 即可执行到该行停止
2> 使用 F2 键。可以设置多个断点，设置断点的地方会被标红，按下 F9 键就会运行到第一个断点。
- 5. 记录执行到“MOV CX, 10”的(BX)，(BP)，(SI)，(DI)各是多少。

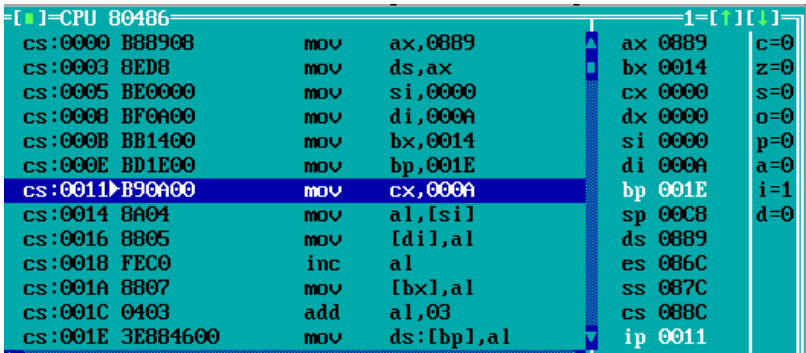


图 2-4 执行到 MOV CX,21 前时情况

表 2-1 执行到 MOV CX,21 前寄存器情况

结果记录			
BX	BP	SI	DI
0014H	001EH	0000H	000AH

分析：记录了偏移地址。

- 6. 继续执行到 INT 21H 之前：

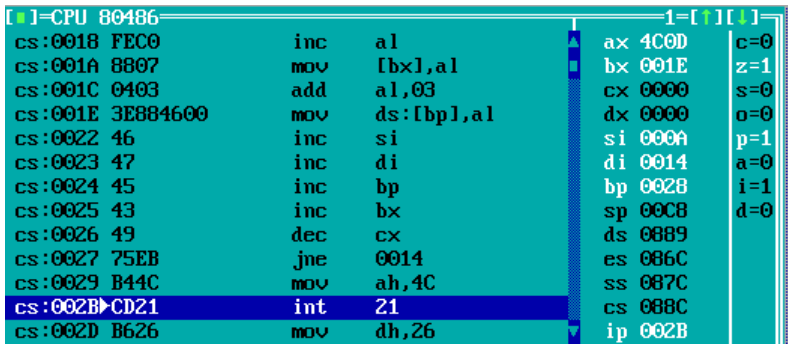


图 2-5 执行到程序末

表 2-2 程序末寄存器值

汇编语言程序设计实验报告

结果记录			
BX	BP	SI	DI
001EH	0028H	000AH	0014H

分析:

程序接着执行到一个循环体, 循环体结束条件是 $ZF = 1$, 具体到这个程序中只要满足 CX 自减到 0 就可以, 因为 CX 一开始等于 10, 所以循环执行 10 次。这四个寄存器都自加 10 就是结果。

- 记录程序执行到退出之前数据段开始 40 个字节的内容, 指出程序运行结果是否与设想的一致。

预测结果:

1 - 10	0	1	2	3	4	5	6	7	8	9
11 - 20	0	1	2	3	4	5	6	7	8	9
21 - 30	1	2	3	4	5	6	7	8	9	A
31 - 40	4	5	6	7	8	9	A	B	C	D

解释:

[SI]取前十个数据, 直接复制给 11-20 个数据, 然后加一后复制给 21-30 个数据, 再加 3 复制给 31-40 个数据。然后每个偏移地址都加 1, 循环十次, 正好得到了上述结果。

从 TD 中得到的结果: 与预测结果相同。

ds:0000	00	01	02	03	04	05	06	07
ds:0008	08	09	00	01	02	03	04	05
ds:0010	06	07	08	09	01	02	03	04
ds:0018	05	06	07	08	09	0A	04	05
ds:0020	06	07	08	09	0A	0B	0C	0D

图 2-6 数据区域实际结果

- 在标号 LOPA 前加上一段程序, 实现新的功能: 先显示提示信息 “Press any key to begin!”, 然后, 在按了一个键之后继续执行 LOPA 处的程序。

```
.386
STACK SEGMENT USE16 STACK
    DB 200 DUP(0)
STACK ENDS

DATA SEGMENT USE16
    BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    BUF2 DB 10 DUP(0)
    BUF3 DB 10 DUP(0)
    BUF4 DB 10 DUP(0)
    Msg db 'Press any key to begin!', 0dh, 0ah, '$'
DATA ENDS

CODE SEGMENT USE16
    ASSUME CS:CODE, DS:DATA, SS:STACK
START:
    MOV AX, DATA
    MOV DS, AX
    MOV SI, OFFSET BUF1
```

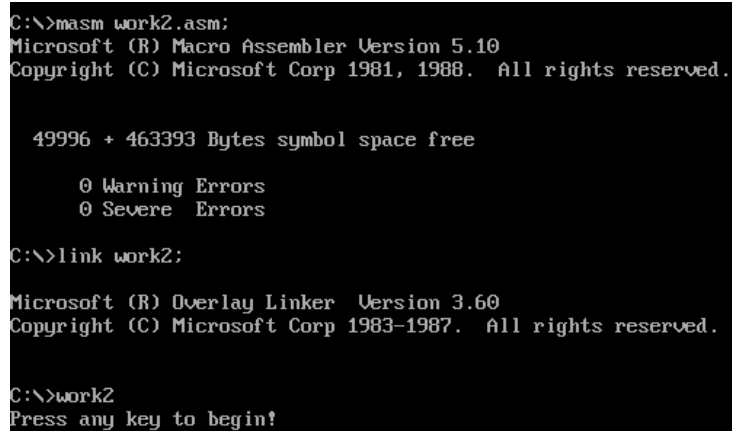
汇编语言程序设计实验报告

```
MOV DI, OFFSET BUF2
MOV BX, OFFSET BUF3
MOV BP, OFFSET BUF4
MOV CX, 10

MOV DX, OFFSET Msg
MOV AH, 9H
INT 21H
MOV AH, 1
INT 21H
LOPA:
MOV AL, [SI]
MOV [DI], AL
INC AL
MOV [BX], AL
ADD AL, 3
MOV DS:[BP], AL
INC SI
INC DI
INC BP
INC BX
DEC CX
JNZ LOPA
MOV AH, 4CH
INT 21H
CODE ENDS
END START
```

调用了 DOS 命令，实现了功能。

下面是编译，链接，执行过程：



```
C:\>masm work2.asm;
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

49996 + 463393 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link work2:

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

C:\>work2
Press any key to begin!
```

图 2-7 编译链接运行结果

3.2.4 实验收获

- (1) 熟悉了常用的 DOS 功能调用输出和输入。
- (2) 熟悉循环程序的结构及控制方法，对改变 CS:IP 值有了新的理解。
- (3) 写了较多汇编语句，对汇编语言也有了一些感觉。

汇编语言程序设计实验报告

3.3 任务 3

3.3.1 实验步骤

1. 对源代码进行修改使它满足前两个小题的要求，即第一个满足三十二位寻址，第二个满足变址寻址访问内存。调 BUG, 直到程序跑通。
2. 记录程序执行到退出之前数据段开始 30 个字符的内容，检查程序运行结果是否与设想一致
3. 在 TD 中观察指令代码在机器中的存放形式与源语句进行对比
4. 从不同位置开始反汇编，观察结果怎么样

3.3.2 实验记录与分析

任务一修改后源程序：

```
.386
STACK SEGMENT USE16 STACK
    DB 200 DUP(0)
STACK ENDS

DATA SEGMENT USE16
    BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    BUF2 DB 10 DUP(0)
    BUF3 DB 10 DUP(0)
    BUF4 DB 10 DUP(0)
DATA ENDS

CODE SEGMENT USE16
    ASSUME CS:CODE, DS:DATA, SS:STACK
START:
    MOV AX, DATA
    MOV DS, AX
    MOV ESI, 0
    MOV EDI, OFFSET BUF2
    MOV EBX, OFFSET BUF3
    MOV EBP, OFFSET BUF4
    MOV CX, 10
LOPA:
    MOV AL, [ESI]
    MOV [DI], AL
    INC AL
    MOV [EBX], AL
    ADD AL, 3
    MOV DS:[EBP], AL
```

汇编语言程序设计实验报告

```
INC ESI
INC EDI
INC EBP
INC EBX
DEC CX
JNZ LOPA
MOV AH, 4CH
INT 21H
CODE ENDS
END START
```

说明：将寻址方式改成了 32 位寄存器。

任务二修改后：（基址加变址寻址）

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:DATA, SS:STACK

START:

MOV AX, DATA

MOV DS, AX

MOV SI, 0

MOV DI, 10

MOV BX, 20

MOV BP, 30

MOV CX, 10

LOPA:

MOV AL, [SI]

MOV [DI], AL

INC AL

MOV [BX], AL

ADD AL, 3

MOV DS:[BP], AL

INC SI

INC DI

INC BP

INC BX

汇编语言程序设计实验报告

```
DEC CX
JNZ LOPA
MOV AH, 4CH
INT 21H
CODE ENDS
END START
```

2. 执行到退出之前数据段开始 40 个字节的内容，检查是否与预期一致。

预测结果：

1 - 10	0	1	2	3	4	5	6	7	8	9
11 -20	0	1	2	3	4	5	6	7	8	9
21 -30	1	2	3	4	5	6	7	8	9	A
31 -40	4	5	6	7	8	9	A	B	C	D

解释：
只改变寻址的寄存器位数大小不会改变结果。与前文相同。
从 TD 中得到的结果：

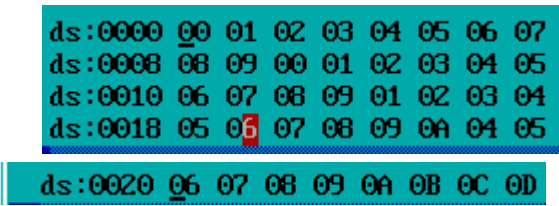


图 3-1 实际结果

与预测结果相同。

3. 在 TD 代码窗口观察到的反汇编语句：

```
mov ax,088F
mov ds,ax
call 00D5
mov dx,01B6
mov ah,09
int 21
lea dx,[01A6]
mov ah,0A
int 21
mov bl,[01A7]
mov bh,[01A8]
cmp bl,00
je 0008
```

图 3-2 反汇编语句

对比情况如表 3-1 所示。

表 3-1 反汇编语句与源程序语句对比

反汇编语句	源程序语句
MOV AX,088F	MOV AX,DATA

汇编语言程序设计实验报告

MOV DS,AX	MOV DS,AX
CALL 0005	CALL SET_AVERAGE_GRADE
MOV DX,01B6	MOV DX, OFFSET MSG1
MOV AH, 9H	MOV AH, 9

由此表格可以看出，反汇编语句将对应的源程序语句中的十进制数转换成了十六进制数，并将所有的变量名转换成了与其相对应的地址。

4. 从不同的地方反汇编：

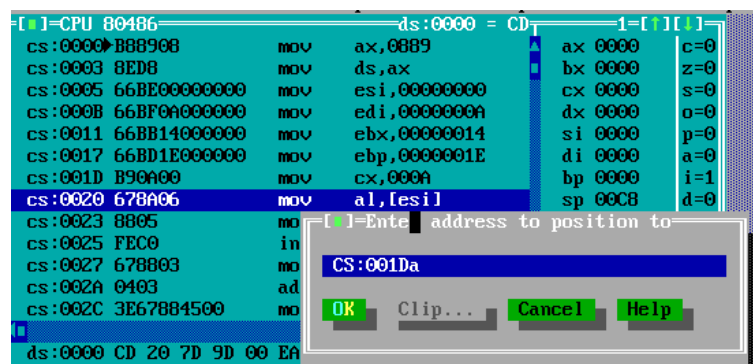


图 3-3 其他位置开始

结果：程序不工作。给定合适的起始地址十分重要

3.4 任务四

3.4.1 程序设计思想及存储单元分配

通过阅读任务要求，我发现这个实验每个功能各要求我们使用一项功能。功能一 主要要求我们熟悉 DOS 功能的调用，功能二 主要要求我们使用循环结构，还有各种寻址方式的使用，功能三 主要实现 DOS 下的数学运算，功能四 则是分支跳转结构。

由此再进一步确定题意可以基本确定我们的程序如何设计。依次完成每个功能，然后在每个功能之间相互跳转。

具体设计思路如下：

1. 按照流程图所示分配四个子程序，输入，查询，等级判断，计算平均值。

2. 输入输出子程序：每次查找显示结束之后回到这个子程序。

寄存器使用：DOS 调用所需：9 号：DX, AH, 10 号:AH。存储数据：IN_NAME 数据段。

3. 查询过程子程序：这个地方使用双重循环，第一层循环于 DATA 数据段，遍历全部的名字，第二层循环则对于每个遍历到的名字，对比它和输入的名字是否相同。具体方法是若每个字母都相同，并且在 DATA 中此人接下来的一个字符是 0，则查找成功，保存此时在 DATA 中的位置，返回；否则继续遍历 DATA 中的下一个人，直到 DATA 中全部的人得到遍历。

寄存器说明：CX 用来做循环变量，第一层循环中初始为人数，每次循环减一；第二次循环开始前先将此时的 CX 入栈，初始化为输入人的字符数，每对比一个字符减一，结束时再出栈。

SI 用来储存当前正在遍历的是第几个字符，在第二个循环中每循环一次加一，在第一个循环中初始化为 0，DI 用来存储第几个人（每 14 个单位是一个人），在第一个循环中每次循环增加 14。

4. 等级判断子程序：多个条件转移指令，大于 90 输出 A，否则大于 80 输出 B，否则大于 70 输出 C，否则输出 D。

寄存器使用：使用了 AX,DX,SI 三个寄存器，DX 使用用来输出换行，使格式更加好看，DL 用来二号调用输出 ABCD 四个等级，AX 用来存放这个人的成绩。SI 则是用来寻址的一个临时使用的寄存器。

5. 求平均值子程序：遍历所有的人，计算每个人的平均成绩，再写到指定位置。

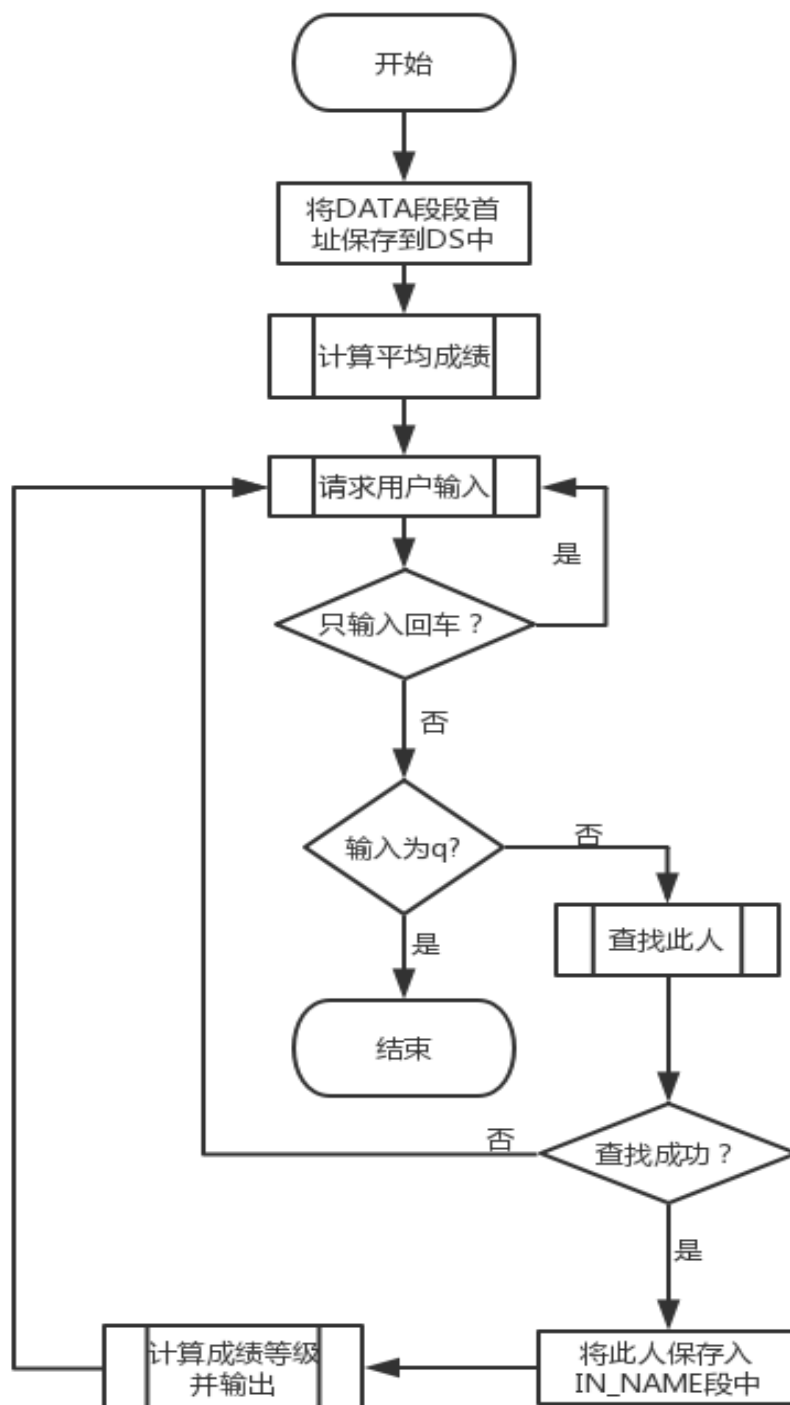
寄存器使用：AX, BX, CX, DX, SI

AX: 做乘法所需寄存器，BX: 临时存放当前结果；CX: 循环使用；DX: 临时存放结果；SI: 表示每个人的位置。

6. 主程序变量说明：N 存储有多少学生，CLRF 段存储一个换行，POIN 存储查找到人的成绩段首址。在以 BUF 为首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：姓名（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）。

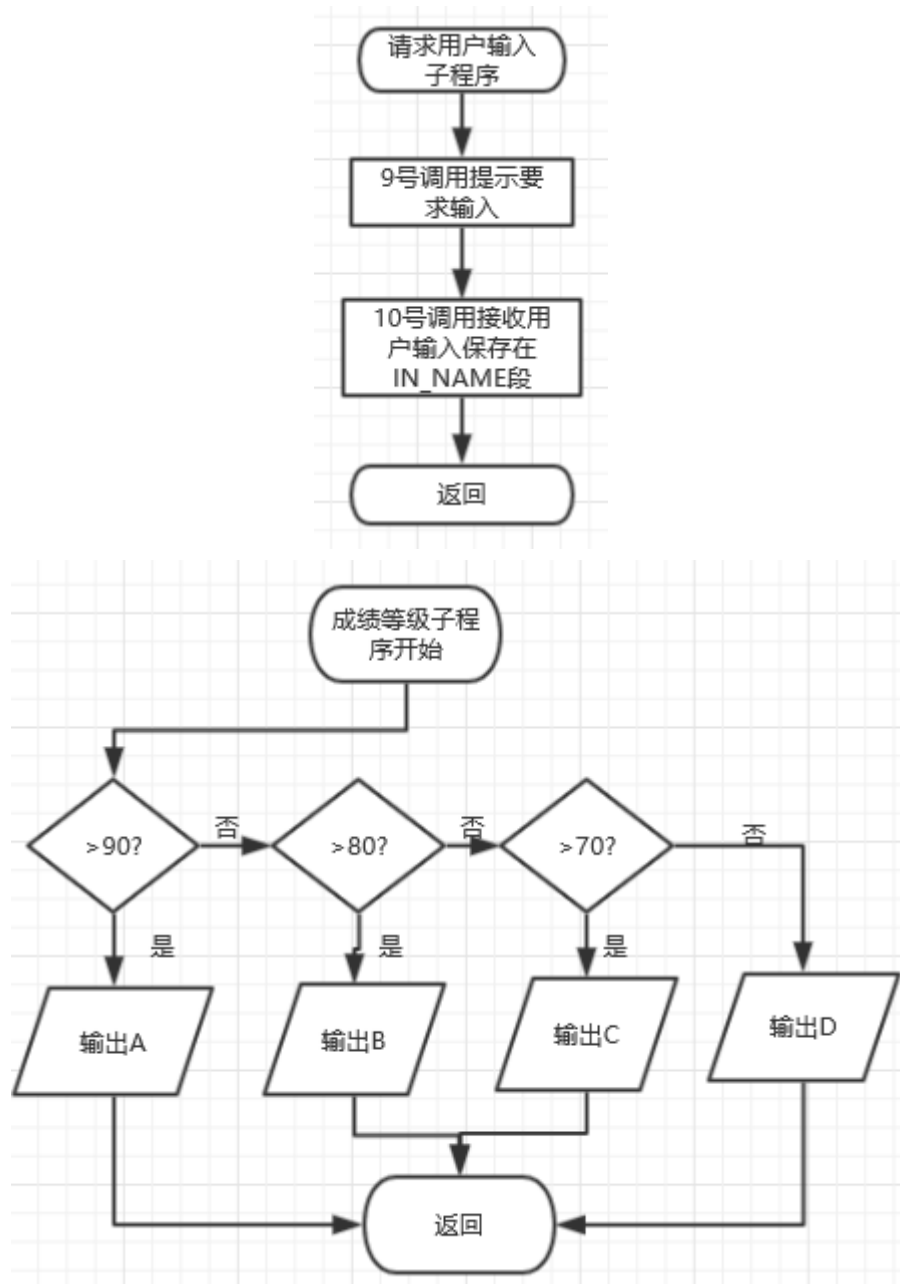
3.4.2 流程图

主程序流程图：

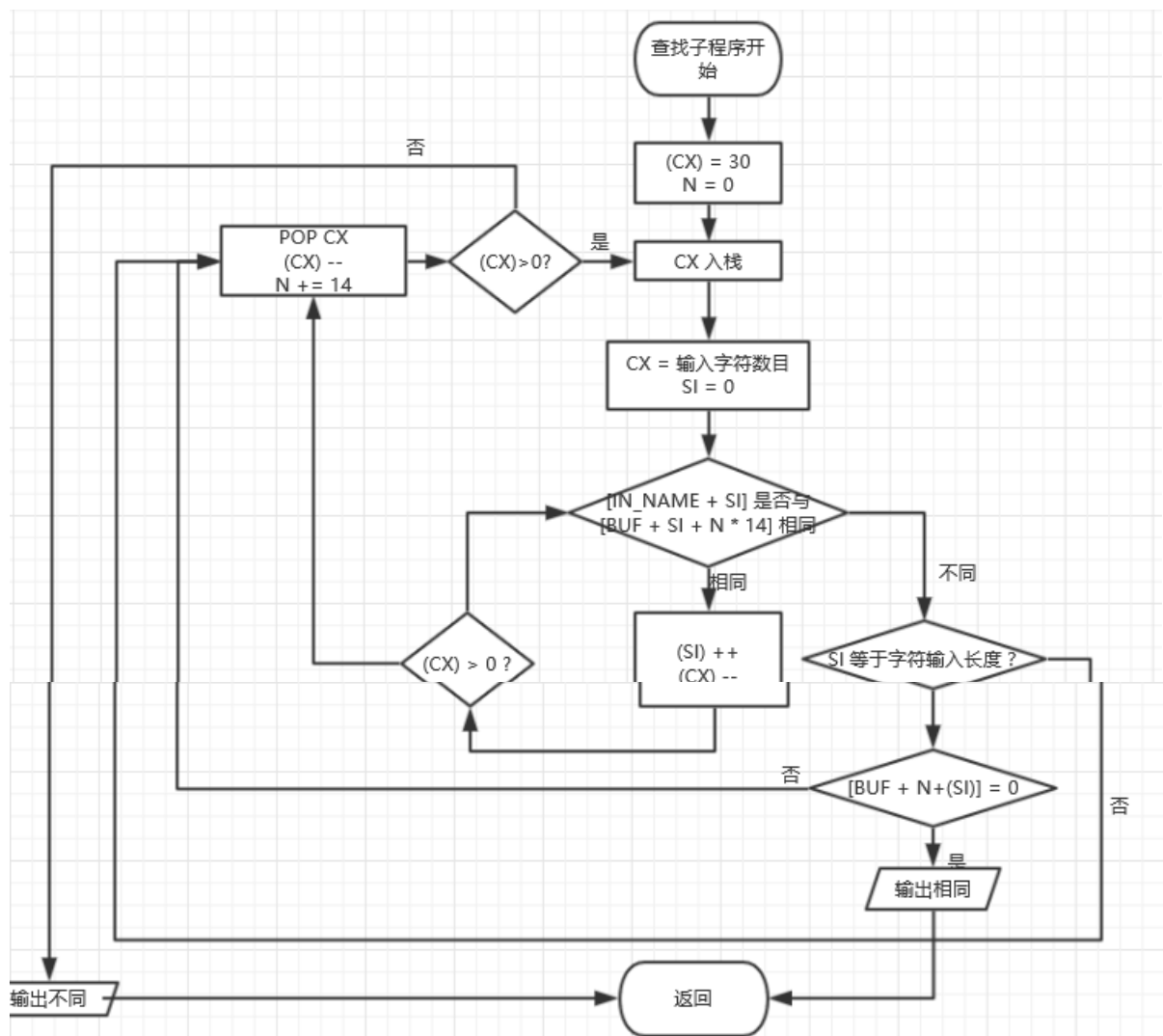


汇编语言程序设计实验报告

子程序流程图：



汇编语言程序设计实验报告



汇编语言程序设计实验报告

3.4.3 源程序

```
;. 386
STACK SEGMENT ;USE16 STACK
    DB 300 DUP(0)
STACK ENDS

DATA SEGMENT; USE16
N      EQU 30
POIN   DW 0

BUF DB 'zhangsan', 0, 0
    DB 100, 85, 80, ?
    DB 'lisi', 6 DUP(0)
    DB 80, 100, 70, ?
    DB 'B', 0, 0, 0, 0, 0, 0, 0, 0, 0
    DB 10, 20, 12, ?
    DB N-4 DUP('TempValue', 0, 80, 90, 95, ?)
    DB 'xinjie', 0, 0, 0, 0
    DB 100, 100, 100, ?

IN_NAME DB 11
        DB 0
        DB 11 DUP(0)

CRLF    DB 0DH, 0AH, '$'
MSG1    DB 0AH, 0DH, 'Please Input Your Name :$'
MSG2    DB 0AH, 0DH, 'Not Find This Student!:$'
MSGA    DB 0AH, 0DH, 'A!$'
MSGB    DB 0AH, 0DH, 'B!$'
MSGC    DB 0AH, 0DH, 'C!$'
MSGD    DB 0AH, 0DH, 'D!$'
DATA ENDS

CODE SEGMENT ;USE16
    ASSUME DS:DATA, CS:CODE, SS:STACK
START:
    MOV AX, DATA
    MOV DS, AX
    CALL SET_AVERAGE_GRADE

INPUT:
    MOV DX, OFFSET MSG1
    MOV AH, 9
    INT 21H ;功能一 小题

    LEA DX, IN_NAME
    MOV AH, 10
    INT 21H ;功能二 小题

    MOV BL, IN_NAME + 1
    MOV BH, IN_NAME + 2

    CMP BL, 0
    JE INPUT
    CMP BH, 'q'
```

汇编语言程序设计实验报告

JE DIE ;功能一 三小题
MOV BH, 0

FIND:

MOV CX, N
MOV DI, 0

FIND_S:

MOV SI, 0
PUSH CX
MOV CX, BX
CALL EQUAL
POP CX
CMP SI, BX
JE SUCCESS_FIND

CONTINUE_FIND:

CMP CX, 1
JE NOT_FIND
ADD DI, 14
LOOP FIND_S

DIE:

MOV AH, 4CH
INT 21H

NOT_FIND:

MOV DX, OFFSET MSG2
MOV AH, 9
INT 21H
JMP INPUT

SUCCESS_FIND:

ADD SI, DI
CMP [BUF + SI], 0
JNE CONTINUE_FIND
SUB SI, DI
MOV WORD PTR [POIN], OFFSET BUF + 10
ADD WORD PTR [POIN], DI
CALL G_ABCD
JMP INPUT

;使用寄存器 AX, SI

;需要传入参数 SI = 0

EQUAL:

MOV AL, [IN_NAME + SI + 2]
ADD SI, DI
CMP AL, [BUF + SI]
JNE NOT_EQUAL
SUB SI, DI
INC SI
LOOP EQUAL

NOT_EQUAL:

RET

G_ABCD:

PUSH AX
PUSH DX
PUSH SI

汇编语言程序设计实验报告

```
MOV DX, OFFSET CRLF
MOV AH, 9
INT 21H
MOV SI, [POIN]
ADD SI, 3
MOV AX, [SI]
MOV AH, 0
SUB AL, 90
JS G_BCD
MOV DL, 'A'
JMP SCREEN
G_BCD:
MOV AX, [SI]
MOV AH, 0
SUB AL, 80
JS G_CD
MOV DL, 'B'
JMP SCREEN
G_CD:
MOV AX, [SI]
MOV AH, 0
SUB AL, 70
JS G_D
MOV DL, 'C'
JMP SCREEN
G_D:
MOV DL, 'D'
JMP SCREEN
SCREEN:
MOV AH, 2
INT 21H
POP SI
POP DX
POP AX
RET

SET_AVERAGE_GRADE:
PUSH SI
PUSH AX
PUSH BX
PUSH CX
PUSH DX

MOV SI, 10
MOV CX, N
MATH:
MOV AX, 0
MOV BX, 0
MOV DX, 0
MOV AL, [BUF + SI]
MOV AH, 2
MUL AH ;AX IS CHINESE * 2 <= 200 16 IS ENOUGH
MOV BL, [BUF + SI + 1] ;MATH GRADE
MOV BH, 0
ADD BX, AX
MOV AL, [BUF + SI + 2] ;ENGLISH
MOV AH, 0
```

汇编语言程序设计实验报告

```
MOV DL, 2
DIV DL
ADD BX, AX
MOV AX, 2
MUL BX
MOV BL, 7
DIV BL
MOV [BUF + SI + 3], AL
ADD SI, 14
LOOP MATH
```

```
POP DX
POP CX
POP BX
POP AX
POP SI
RET
```

```
CODE ENDS
END START
```

3.4.4 实验步骤描述

1. 准备上机环境
2. 验证程序正确性
 - (1) 编译源程序，若有 bug，按照 bug 提示信息修改 bug，直到没有 bug，进行下一步。
 - (2) 进行样例输入测试，对处于数据段最前、最后、中间的数据和没有出现的数据都进行一次查询，先用计算机算出应该出现的值，再观察查询结果是否正确（ABCD 等级的显示）
 - (3) 使用 TD 验证平均值计算正确与否
 - <1> 调试执行到 SET_AVERAGE_GRADE 处，在 td 的内存查看中跳转到 DS:00H 处。
 - <2> 使用计算机计算每一个人的加权平均成绩
 - <3> 在 td 的内存查看中定位到平均值处，比对是否相同。
 - <4> 若不同，则自己的代码仍然存在逻辑错误，仔细审查，解决 bug。
3. 测试没有 '\$' 符号时，使用九号调用的结果
 - (1) 写测试 demo，定义数据，code 中写入标准的 9 号调用
LEA DX, MSG MOV AH, 9 INT 21H
 - (2) 在 MSG 数据段中定义字符串不要加入 '\$'
 - (3) 编译链接，运行观察结果，记录分析原因。
4. 测试未给予 DX/DS 正确的值，再使用九号调用结果如何？
 - (1) 写测试 demo，定义数据，code 中写入不给 DX 赋值的 9 号调用
MOV AH, 9 INT 21H
 - (2) 编译链接，运行观察结果，记录分析原因。
5. 再 td 中研究十号调用超出预设长度会发生什么现象？

汇编语言程序设计实验报告

- (1) 写测试 demo, 定义 BUF 区域长度为 5 (长度断容易测试), 使用标准 10 号调用。
LEADX, BUF MOV AH, 10 INT 21H
- (2) 编译链接, 首先直接运行, 看程序是否正确, 输入少于五个字符进行测试。
- (3) Td 反汇编该文件, 在 10 号调用处打断点, 执行到断点处, 输入超过五个字符, 观察现象。分析结果。
6. 观察指令机器码, 研究机器指令码与汇编代码之间的关系
 - (1) 研究的几条代码: mov 含常数的赋值指令, jmp, loop 循环指令
 - (2) 在 Demo 中写出使用这几条代码的指令 (源文件代码太多不宜观察, 可以先得到初步结论后取验证)
 - (3) 编译链接该文件, td 反汇编可执行文件, 观察结果, 得出初步结论。
 - (4) 反汇编原文件, 观察机器码, 与上文中得出的结论进行验证, 若发现结论不一致, 查找资料并且继续进行思考
7. 确定自己的循环是否会出现死循环
 - (4) 写测试数据, 看自己的代码思考死循环的可能。
 - (5) 对每个测试数据都进行测试, 看是否发生死循环
 - (6) 若发生死循环, 修改自己的代码直到没有死循环。
8. 选取特殊值验证其会不会发生溢出
 - a) 写测试数据, 例如 0, 100 等
 - b) 测试
 - c) Td 观察是否溢出

3.4.5 实验记录与分析

1. 上机环境确认:
操作系统: Windows 10 下 DOSBox 0.74
2. 验证程序正确性:
编译链接源程序 (由于我是一边写一边编译) 所以没有发现 BUG, 进行样例检查:
选择第一个人: zhangsan, 最后一个人 xinjie, 中间的人 B 都输出了结果

```
C:\>STUDENT.EXE
Please Input Your Name :xinjie
A
Please Input Your Name :zhangsan
A
Please Input Your Name :B
D
Please Input Your Name :
Please Input Your Name :q
C:\>
```

图 4-1 程序运行

使用 TD 验证平均值正确与否:

汇编语言程序设计实验报告

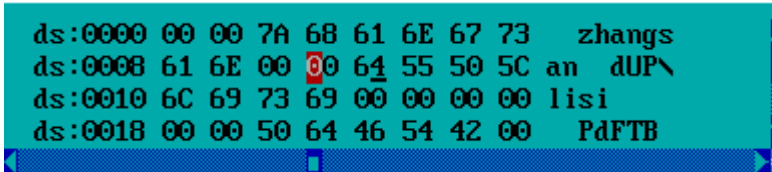


图 4-2 张三的成绩

使用计算机计算，得张三的成绩计算正确。依次查看其他人的成绩（人数太多只用张三代表这个过程）。

结果正确。

3. 测试没有'\$'符号时，使用九号调用的结果

测试代码	测试结果
<pre>.386 DATA SEGMENT USE16 MSG DB 0AH, 0DH, 'Please Input Your Name :' DATA ENDS CODE SEGMENT USE16 ASSUME DS:DATA, CS:CODE START: MOV AX, DATA MOV DS, AX LEA DX, MSG MOV AH, 9 INT 21 CODE ENDS END START</pre>	<div></div> <p>现象： 在所写字符串之后又输出了一串乱码。</p> <p>结果分析： 因为没有输入结束符，所以应该是输出了你要输出的字符后它还向后输出直到遇到了结束符。</p>

4. 测试未给予 DX/DS 正确的值，再使用九号调用结果如何？

测试代码	测试结果
<pre>.386 CODE SEGMENT USE16 ASSUME CS:CODE START: MOV AH, 9 INT 21H MOV AH, 4CH INT 21H CODE ENDS END START</pre>	<div></div> <p>结果： 显示了一对乱码</p> <p>分析： 没有给 DS/DX 赋予正确的值，DS:DX 寻址时便按照初始值寻址了，而这个初始值你是没有办</p>

汇编语言程序设计实验报告

法确定的，按照上文，输出乱码。

5. 再 td 中研究十号调用超出预设长度会发生什么现象？

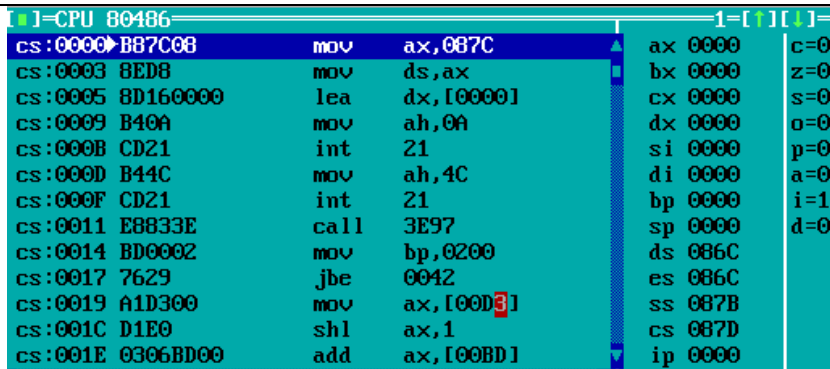
测试代码

```
.386
DATA SEGMENT USE16
    IN_NAME DB 5
             DB 0
             DB 5 DUP(0)
DATA ENDS
CODE SEGMENT USE16
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    LEA DX, IN_NAME
    MOV AH, 10
    INT 21H ;10 号调用

    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

TD 中研究结果



CS:0000	0000	mov	ax, 007C	ax	0000	c=0
CS:0003	8ED8	mov	ds, ax	bx	0000	z=0
CS:0005	8D160000	lea	dx, [0000]	cx	0000	s=0
CS:0009	B40A	mov	ah, 0A	dx	0000	o=0
CS:000B	CD21	int	21	si	0000	p=0
CS:000D	B44C	mov	ah, 4C	di	0000	a=0
CS:000F	CD21	int	21	bp	0000	i=1
CS:0011	E8833E	call	3E97	sp	0000	d=0
CS:0014	BD0002	mov	bp, 0200	ds	086C	
CS:0017	7629	jbe	0042	es	086C	
CS:0019	A1D300	mov	ax, [00D3]	ss	087B	
CS:001C	D1E0	shl	ax, 1	cs	087D	
CS:001E	0306BD00	add	ax, [00BD]	ip	0000	

```
C:\>TEST.EXE
xi
C:\>td TEST.EXE
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
xinji
```

结果分析：

第一幅图是进入 td 的初始状态，第二个是我在 td 中单步调试到 10 号调用的情况：我定义的段区是 5 个字符，我无法在这里输入超过五个的字符。也就是说最多显示五个字符（在输入五个字符后也无法按下回车，因为回车也算一个字符）

汇编语言程序设计实验报告

6. 观察指令机器码，研究对应偏移地址和该编码之间的关系

a) 研究的几条代码：mov 含常数的赋值指令，jmp, loop 循环指令

START: MOV AX, 0FFH

JMP START

b) 编译链接该文件，td 反汇编可执行文件，观察结果，得出初步结论。

```
cs:0000 B8FF00 mov ax,0FFH
cs:0003 EBFB jmp 0000
```

图 4-3

可以看出 mov ax, 0ffH 中的机器码出现了 FF 大胆假设常量一定会出现在机器码中，JMP 的机器码则没有观察到明显规律

7. 确定自己的循环是否会出现死循环

执行自己的程序，输入各种可能的情况，没有发生死循环：

```
Please Input Your Name :xinjie
A
Please Input Your Name :
Please Input Your Name :
Please Input Your Name :x
Not Find This Student!:
Please Input Your Name :xxa
Not Find This Student!:
Please Input Your Name :B
D
Please Input Your Name :lolo
Not Find This Student!:
Please Input Your Name :
Please Input Your Name :lili
Not Find This Student!:
Please Input Your Name :
Please Input Your Name :B
D
Please Input Your Name :zhangsan
A
Please Input Your Name :q
```

图 4-4 验证没有发生死循环

8. 选取特殊值验证其会不会发生溢出

选取测试数据 100, 100, 100

0, 0, 0

测试是否发生溢出：

```
ds:0000 00 00 7A 68 61 6E 67 73 zhangs
ds:0008 61 6E 00 00 00 00 00 an
```

图 4-5 张三的成绩都为 0 0 0 没有溢出

```
78 69 6E 6A 69 65 00 00 xinjie
00 00 64 64 64 64 0B 00 dddd
```

图 4-6 辛杰的成绩都为 100, 100, 100,没有发生溢出

汇编语言程序设计实验报告

4 总结与体会

通过任务 1 的实验，我明白了以下几点：首先是 TD 这个调试工具的使用，我学会了如何在 TD 中直接输入代码并单步执行，也了解了 TD 各个窗口内是什么功能。会判断当前寄存器的值和标志寄存器的值也可以观察内存中的值。其次，我更加深入得了解了标志寄存器的作用，了解了每个标志寄存器是用来干嘛的，进行 ADD, SUB 操作时标志寄存器变换的规律。

通过任务二的实验，我明白了以下几点：首先是对 TD 工具的进一步掌握，明白了如何设置断点，如何执行到断点，方便了以后程序的调试。其次学会了如何编译链接一个程序，如何反汇编一个程序，可以自己写文件然后运行这个汇编程序了。再者，学会了 DOS 系统功能调用，明白了系统功能调用的一般形式。而且在这个过程中也对汇编程序的寻址方式有了更加深入的了解。

通过任务三的实验，我明白了以下几点：首先是进一步熟悉了 TD 工具的调试操作，然后熟悉了寄存器直接寻址，基址加变址寻址这两种寻址方式，还进一步熟悉了反汇编语句与汇编语句之间的联系和区别以及 CS:IP 指向位置的重要性。

通过任务四的实验，我明白了以下几点：首先是如何使用汇编语言编写较复杂程序的一般思路：分析寄存器功能，画流程图。然后熟悉了数据段的访问，循环结构的设计，分支结构的设计，以及 DOS 功能调用的进一步熟悉。很有收获。最后还锻炼了优化程序的一般思路。

本次上机不仅提高了编程水平，熟悉了工具的使用，而且加深了对一些知识的理解。主要的经验教训如下：

首先，明白了实验前准备的重要性，实验前的准备可以让你在实验过程中更加具有目的性，有了对事情如何安排的看法。

其次，录入程序时要注意一些细节，比如中文分号、字母 O 等问题，虽然汇编程序指出其所在行有错，但很难发现具体是哪个符号错了，耽误了不少时间。TD 在程序细节的观察、动态修改方面有很大的作用，要主动用 TD 的调试功能帮助自己发现、理解与解决问题。

还有，在编写程序出现 BUG 时，我发现自己容易凭直觉去修改，而不是凭借知识取修改，这是个很严重的问题，需要克服。

汇 编 语 言 程 序 设 计 实 验 报 告

参考文献

- [1] 王爽. 汇编语言. 第三版. 清华大学出版社, 2003 01- 310
 - [2] 曹忠升 80X86 汇编语言程序设计 华中科技大学出版社
-