

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 程序执行时间与代码长度优化

实验时间： 2017-3-29, 14: 00-17: 30 实验地点： 南一楼 803 室 111 号实验台

指导教师： 张勇

专业班级： 计算机科学与技术 201307 班

学 号： U201314969 姓 名： 王镇宇

同组学生： 无 报告日期： 2017 年 3 月 30 日

原创性声明

本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名：

日期：

成绩评定

实验完成质量得分（70 分） （实验步骤清晰详细深入， 实验记录真实完整等）	报告撰写质量得分（30 分） （报告规范、完整、通顺、详实等）	总成绩（100 分）

指导教师签字：

日期：

汇编语言程序设计实验报告

目录

1	实验目的与要求.....	1
2	实验内容.....	1
3	实验过程.....	2
3.1	任务 1.....	2
3.1.1	设计思想及存储单元分配.....	2
3.1.2	流程图.....	3
3.1.3	源程序.....	3
3.1.4	实验步骤.....	7
3.1.5	实验记录.....	8
3.2	任务 2.....	8
3.2.1	设计思想及存储单元分配.....	8
3.2.2	流程图.....	8
3.2.3	源程序.....	8
3.2.4	实验步骤.....	12
3.2.5	实验记录.....	12
3.3	任务 3.....	13
3.3.1	设计思想及存储单元分配.....	13
3.3.2	流程图.....	13
3.3.3	源程序.....	13
3.3.4	实验步骤.....	13
4	体会.....	14
	参考文献.....	15

汇编语言程序设计实验报告

1 实验目的与要求

- (1) 熟悉汇编语言指令的特点，掌握代码优化的基本方法；
- (2) 理解高级语言程序与汇编语言程序之间的对应关系。

2 实验内容

任务 1. 观察多重循环对 CPU 计算能力消耗的影响

若有 m 个用户在同一台电脑上排队使用实验一任务四的程序，想要查询成绩列表中最后一个学生“wangwu”的平均成绩，那就相当于将实验一任务四的程序执行了 m 次。为了观察从第一个用户开始进入查询至第 m 个用户查到结果之间到底延迟了多少时间，我们让实验一任务四的功能二和功能三的代码重复执行 m 次，通过计算这 m 次循环执行前和执行后的时间差，来感受其影响。由于功能一和功能四需要输入、输出，速度本来就较慢，所以，没有纳入到这 m 次循环体内（但可以保留不变）。请按照上述设想修改实验一任务四的程序，并将 m 值尽量取大（建议 $m \geq 1000$ ，具体数值依据实验效果来改变，逐步增加到比较明显的程度，比如秒级延迟），以得到较明显的效果。

提示：（1）在进入功能二之前增加 m 次循环的初始化工作，在功能三结束之后增加 m 次循环的条件判断和转移语句。

（2）学校汇编教学网站的软件下载中提供了显示当前时间“秒和百分秒”的子程序。若在 m 次循环前调用一下该子程序， m 次循环执行完之后再调用一下该子程序，就能在屏幕上观察并感受到执行循环前后的时间差（时间差值需要自行手工计算，当然，你也可以选用网站上另一个计时程序，它是可以帮你计算好差值的）。注意，由于虚拟机环境下 CPU 会被分时调度，故该时间差值会因计算机运行环境与状态的不同而不同。

任务 2. 对任务 1 中的汇编源程序进行优化

优化工作包括代码长度的优化和执行效率的优化，本次优化的重点是执行效率的优化。请通过优化 m 次循环体内的程序，使程序的执行时间尽可能减少 10% 以上。减少的越多，评价越高！

优化方法提示：首先是通过选择执行速度较快的指令来提高性能，比如，把乘除指令转换成移位指令、加法指令等；其次，内循环体中每减少一条指令，就相当于减少了 $m \times n$ 条指令的执行时间，需要仔细斟酌；第三，尽量采用 32 位寄存器寻址，能有更多的机会提高指令执行效率。

汇编语言程序设计实验报告

任务 3. 观察用 C 语言实现的实验一任务四中功能一的程序与汇编语言实现的程序的差异

用汇编语言和 C 语言分别实现实验一任务四中功能一的功能（对汇编语言而言，就是把实验一中相关程序摘取出来成为独立的程序），对比两种语言实现的程序的代码情况，观察和总结 C 语言编写程序和自己的汇编语言程序的对应关系及差异，总结其中可以简化的地方。

提示：采用反汇编方法观察 C 语言编写生成的执行程序时，首先观察程序的整体结构特点（包括段的情况），然后重点分析、比较输出输入字符串对应的代码。

注意：C 语言程序的编译环境可以自选，但生成的程序要与汇编语言程序属于同一类型（比如，都是实模式下的 16 位段程序）。

3 实验过程

3.1 任务 1

3.1.1 设计思想及存储单元分配

查找和计算成绩的功能与第一次实验相同。循环是通过定义一个双字类型的 COUNT，每查找和计算一次后减一，COUNT 等于 0 时退出。

寄存器分配：

BP:用来存储输入字符串的地址

BX:用来存储 BUF 段的首地址

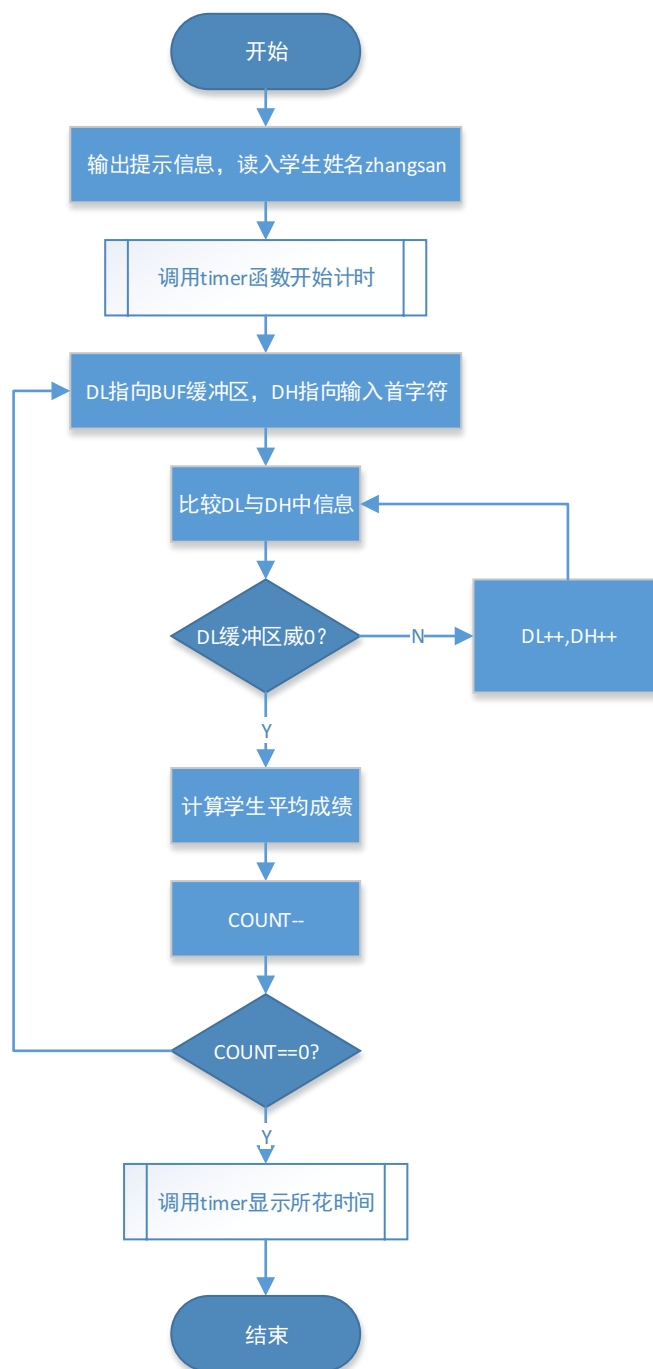
SI:查找字符时实现递增到下一字符

DX:字符串缓冲区地址，也用来计算成绩

AX:计算成绩是存储数据

汇编语言程序设计实验报告

3.1.2 流程图



3.1.3 源程序

优化前的代码（优化部分的代码见任务二实验记录）:

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

汇编语言程序设计实验报告

```
DATA    SEGMENT      USE16
        COUNT DD 100000000
        BUF   DB 'zhangsan', 0, 0
           DB 100, 85, 80, ?
        MSG   DB 'Input the name : $'
        CONTINU DB 'Enter q to exit:$'
        INPUT DB 10
           DB ?
           DB 10 DUP(0)
DATA    ENDS

CODE    SEGMENT      USE16
        ASSUME CS:CODE, DS:DATA, SS:STACK
START:  MOV     AX, DATA
        MOV     DS, AX
        JMP     BEGIN

BEGIN:  LEA     DX, OFFSET MSG      ;输出提示信息
        MOV     AH, 9H
        INT     21H
        LEA     DX, OFFSET INPUT   ; 读入学生姓名
        MOV     AH, 0AH
        INT     21H
        MOV     DL, 0AH            ; 换行符
        MOV     AH, 2H
        INT     21H

        MOV     AX, 0              ;表示开始计时
        CALL    TIMER

NEXT:   LEA     BP, OFFSET INPUT    ; 将 INPUT 基址存放至 BP
        ADD     BP, 2
        LEA     BX, OFFSET BUF     ; 将 BUF 基址存放至 BX
        MOV     SI, -1

COMP:   INC     SI
        MOV     DL, [BX + SI]
        MOV     DH, BYTE PTR DS:[BP + SI]
```

汇编语言程序设计实验报告

```
CMP     DL, 0
JZ      CAL
CMP     DL, DH
JZ      COMP

CAL:    ADD     BX, 10      ; 根据目标学生下标值, 找到分数缓冲区首地址 BX = 10
MOV     AX, 0
MOV     DX, 2
MOV     AL, [BX]          ; 计算平均成绩
IMUL    DX
MOV     DL, [BX + 1]
ADD     DX, AX             ; A*2+B
MOV     AL, [BX + 2]
MOV     CX, 2
IDIV    CL
ADD     AX, DX             ; AL=A*2+B+C/2
IMUL    CL
MOV     DL, 7
IDIV    DL                 ; AL = 2 * AL / 7
MOV     [BX + 3], AL; AVG = AL ( AL / 3.5)

DEC     COUNT
JNZ     NEXT

MOV     AX, 1
CALL    TIMER ;终止计时并显示计时结果(ms)

LOOPA:  MOV     DL, 0AH          ;换行符
MOV     AH, 2H
INT     21H
LEA     DX, OFFSET CONTINU      ;继续
MOV     AH, 9H
INT     21H
MOV     DL, 0AH          ;换行符
MOV     AH, 2H
INT     21H
MOV     DL, 0DH
MOV     AH, 2H
```

汇编语言程序设计实验报告

```
INT     21H
MOV     AH, 8H
INT     21H
CMP     AL, 71H
JE      OVER           ;输入 'q', 退出程序
```

```
TIMER  PROC
    PUSH  DX
    PUSH  CX
    PUSH  BX
    MOV   BX, AX
    MOV   AH, 2CH
    INT   21H           ;CH=hour(0-23), CL=minute(0-59), DH=second(0-
59), DL=centisecond(0-100)
    MOV   AL, DH
    MOV   AH, 0
    IMUL  AX, AX, 1000
    MOV   DH, 0
    IMUL  DX, DX, 10
    ADD   AX, DX
    CMP   BX, 0
    JNZ   _T1
    MOV   CS:_TS, AX
_T0:    POP   BX
    POP   CX
    POP   DX
    RET
_T1:    SUB   AX, CS:_TS
    JNC   _T2
    ADD   AX, 60000
_T2:    MOV   CX, 0
    MOV   BX, 10
_T3:    MOV   DX, 0
    DIV   BX
    PUSH  DX
    INC   CX
    CMP   AX, 0
    JNZ   _T3
    MOV   BX, 0
```

汇编语言程序设计实验报告

```
_T4:    POP    AX
        ADD    AL, '0'
        MOV    CS:_TMSG[BX], AL
        INC    BX
        LOOP   _T4
        PUSH   DS
        MOV    CS:_TMSG[BX+0], 0AH
        MOV    CS:_TMSG[BX+1], 0DH
        MOV    CS:_TMSG[BX+2], '$'
        LEA    DX, _TS+2
        PUSH   CS
        POP    DS
        MOV    AH, 9
        INT    21H
        POP    DS
        JMP    _T0
_TS DW    ?
        DB     'Time elapsed in ms is '
_TMSG DB     12 DUP(0)
TIMER  ENDP

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

3.1.4 实验步骤

1. 使用编辑程序 EDIT.EXE 录入源程序，存盘文件名为 CUBE.ASM。
2. 使用 MASM5.0 或 6.0 汇编源文件。即 MASM CUBE;
3. 观察提示信息，若出错，则用编辑程序修改错误，存盘后重新汇编，直至不再报错为止。
4. 使用连接程序 LINK.EXE 将汇编生成的 CUBE.OBJ 文件连接成执行文件，即 LINK CUBE;
5. 若连接时报错，则依照错误信息修改源程序。之后重新汇编和连接，直至不再报错并生成 CUBE.EXE 文件。
6. 执行该程序。即在命令行提示符后输入 CUBE 后回车，观察执行现象。
7. 调用 timer 函数直接显示我的 1 亿次循环花费多少时间，有两种方法可以获取程序执行时间，另一种是用 disptime 显示程序开始运行的时间和程序结束运行的时间，时间差即为程序执行时间。
8. 经测试，当循环次数为 1 亿次时，显示花费 2470ms，当循环次数为 1000 万次时，显示花费 270ms，可见循环次数对 CPU 资源消耗的影响还是很明显的。

汇编语言程序设计实验报告

3.1.5 实验记录

1、实验环境条件：MacBook Pro：2.9 Ghz Intel Core i5, 8G 内存；DosBox：EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。

2、刚开始直接在第一次实验基础上匹配字符串和查询部分直接添加一个大循环，但跑出来花费时间始终是 0，但在 td 里调试时又明显有时间，而且设置断点在 ds 里观察 COUNT 的确从 1 亿减到 0，之后把程序中多余的部分直接删除，又改变了程序部分代码的逻辑顺序，而且只查询 zhangsan 这一个同学，采取循环 1 亿次的方式来观察时间，终于得到了正确的结果。循环部分见如下：

```
COUNT DD 100000000
```

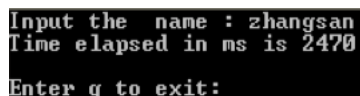
```
.....
```

```
DEC COUNT
```

```
JNZ NEXT
```

省略号部分是查询和计算平均成绩的代码

3、运行出来如下图



```
Input the name : zhangsan
Time elapsed in ms is 2470
Enter q to exit:
_
```

分析：循环 1 亿次进行查询和计算平均成绩花费时间 2470ms

3.2 任务 2

3.2.1 设计思想及存储单元分配

同任务一。

3.2.2 流程图

同任务一。

3.2.3 源程序

```
.386
```

```
STACK SEGMENT USE16 STACK
```

```
DB 200 DUP(0)
```

```
STACK ENDS
```

```
DATA SEGMENT USE16
```

```
COUNT DD 100000000
```

```
BUF DB 'zhangsan', 0, 0
```

```
DB 100, 85, 80, ?
```

```
MSG DB 'Input the name : $'
```

汇编语言程序设计实验报告

```
CONTINU DB 'Enter q to exit:$'
INPUT DB 10
        DB ?
        DB 10 DUP(0)
DATA    ENDS

CODE    SEGMENT USE16
        ASSUME CS:CODE, DS:DATA, SS:STACK
START:  MOV     AX, DATA
        MOV     DS, AX
        JMP     BEGIN

BEGIN:  LEA     DX, OFFSET MSG      ;输出提示信息
        MOV     AH, 9H
        INT     21H
        LEA     DX, OFFSET INPUT   ; 读入学生姓名
        MOV     AH, 0AH
        INT     21H
        MOV     DL, 0AH            ; 换行符
        MOV     AH, 2H
        INT     21H

        MOV     AX, 0              ;表示开始计时
        CALL    TIMER

NEXT:   LEA     BP, OFFSET INPUT    ; 将 INPUT 基址存放至 BP
        ADD     BP, 2
        LEA     BX, OFFSET BUF     ; 将 BUF 基址存放至 BX
        MOV     SI, -1

COMP:   INC     SI
        MOV     DL, [BX + SI]
        MOV     DH, BYTE PTR DS:[BP + SI]

        CMP     DL, DH
        JZ      COMP

CAL:    ADD     BX, 10              ; 根据目标学生下标值, 找到分数缓冲区首地址 BX = 10
        MOV     AX, 0
```

汇编语言程序设计实验报告

```
MOV     DX, 0
MOV     AL, [BX]      ; 计算平均成绩
ADD     AX, AX        ; A*2
MOV     DL, [BX +1]
ADD     AX, DX        ; A*2+B
MOV     DL, [BX +2]
SAR     DL, 1         ; C/2
ADD     AX, DX        ; AL=A*2+B+C/2
SAL     AX, 1         ; AX*2
MOV     DL, 7
IDIV    DL            ; AL = 2 * AL / 7
MOV     [BX + 3], AL ; AVG = AL ( AL / 3.5)

DEC     COUNT
JNZ     NEXT

MOV     AX, 1
CALL    TIMER        ; 终止计时并显示计时结果(ms)

LOOPA:  MOV     DL, 0AH          ; 换行符
        MOV     AH, 2H
        INT     21H
        LEA     DX, OFFSET CONTINU ; 继续
        MOV     AH, 9H
        INT     21H
        MOV     DL, 0AH          ; 换行符
        MOV     AH, 2H
        INT     21H
        MOV     DL, 0DH
        MOV     AH, 2H
        INT     21H
        MOV     AH, 8H
        INT     21H
        CMP     AL, 71H
        JE      OVER            ; 输入 'q', 退出程序

TIMER   PROC
        PUSH    DX
        PUSH    CX
```

汇编语言程序设计实验报告

```
PUSH  BX
MOV   BX, AX
MOV   AH, 2CH
INT   21H           ;CH=hour(0-23), CL=minute(0-59), DH=second(0-
59), DL=centisecond(0-100)
MOV   AL, DH
MOV   AH, 0
IMUL  AX, AX, 1000
MOV   DH, 0
IMUL  DX, DX, 10
ADD   AX, DX
CMP   BX, 0
JNZ   _T1
MOV   CS:_TS, AX
_T0:   POP   BX
      POP   CX
      POP   DX
      RET
_T1:   SUB   AX, CS:_TS
      JNC   _T2
      ADD   AX, 60000
_T2:   MOV   CX, 0
      MOV   BX, 10
_T3:   MOV   DX, 0
      DIV   BX
      PUSH  DX
      INC   CX
      CMP   AX, 0
      JNZ   _T3
      MOV   BX, 0
_T4:   POP   AX
      ADD   AL, '0'
      MOV   CS:_TMSG[BX], AL
      INC   BX
      LOOP  _T4
      PUSH  DS
      MOV   CS:_TMSG[BX+0], 0AH
      MOV   CS:_TMSG[BX+1], 0DH
      MOV   CS:_TMSG[BX+2], '$'
```

汇编语言程序设计实验报告

```
LEA    DX, _TS+2
PUSH   CS
POP     DS
MOV     AH, 9
INT     21H
POP     DS
JMP     _T0
_TS DW    ?
DB      'Time elapsed in ms is '
_TMSG DB    12 DUP(0)
TIMER  ENDP

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

3.2.4 实验步骤

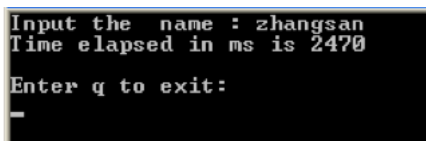
- 1.可以尽量减少循环里面语句的条数，还可以将乘除运算改为移位运算和加法指令。
- 2.优化后程序执行的时间应该会有明显减少

3.2.5 实验记录

- 1、具体优化的代码如下：

```
CMP     DL, 0
JZ       CAL
```

经分析，这两句代码删除后不会影响程序的功能（输入姓名正确的前提下），故删除之，删除后程序运行如下：



可见时间并没有减少。分析应该是循环次数太少不足以体现差距

把	MOV	AL, [BX]	; 计算平均成绩
	IMUL	DX	
改为	MOV	AL, [BX]	
	ADD	AX, AX	;A*2
把	MOV	AL, [BX +2]	
	MOV	CX, 2	

汇编语言程序设计实验报告

改为 IDIV CL
 MOV AL, [BX +2]
 SAR DL, 1 ;C/2

把 MOV CX, 2
 IMUL CL
改为 SAL AX, 1

（上述改动仅为运算改动，部分寄存器改动没有列出，详细见源代码）
运行如下图：

```
Input the name : zhangsan
Time elapsed in ms is 1260
Enter q to exit:
```

分析：优化后时间减少了 49%，可见将乘除运算改为加运算和移位运算能够极大提高程序运行效率，极大减少程序运行时间。说明此次优化是成功的。

3.3 任务 3

3.3.1 设计思想及存储单元分配

无

3.3.2 流程图

无

3.3.3 源程序

略

3.3.4 实验步骤

采用反汇编方法观察 C 语言编写生成的执行程序时，首先观察程序的整体结构特点（包括段的情况），然后重点分析、比较输出输入字符串对应的代码。

优化前与优化后运行的时间对比：

```
C:\Masm60>ex3D.exe
Average() address = 393
The time was: 55ms

C:\Masm60>ex3F.exe
Average() address = 392
The time was: 0ms
```

汇编语言程序设计实验报告

4 体会

通过这次试验，我进一步加深了对 TD 调试器的使用理解，更好的使用调试器来检测并定位程序段错误的位置，提高了效率。刚开始的时候，程序每次都在执行到 Input the name 之后就卡住，后来仔细优化了代码结构之后用 TD 调试发现了问题所在，是由于编写代码的时候一段逻辑错误。

这次试验的代码工作量相比上次实验多了很多，任务艰巨了不少，但是通过对汇编代码的编写和优化，我进一步加深了对课堂上所学知识的理解。再加上任务三汇编与 C 语言的联系，让我能够进一步了解到反汇编的知识。这次试验收获颇丰，对今后的实验具有很好的指导意义。

汇 编 语 言 程 序 设 计 实 验 报 告

参考文献

- [1] 王元珍等. 80x86 汇编语言程序设计. 版本(第 1 版)
- [2] 王晓虹等. 汇编语言程序设计教程. 版本(第 1 版)