

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： 2018-4-9, 14:00-17:30 实验地点： 南一楼 804 室

指导教师： 朱虹

专业班级 计算机 201601 班

学 号： U201614532 姓 名： 吕鹏泽

同组学生： 无 报告日期： 2018 年 4 月 10 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：

目录

| | | |
|-------|------------------|----|
| 1. | 实验目的与要求 | 2 |
| 2. | 实验内容..... | 3 |
| 3. | 实验过程..... | 4 |
| 3.1 | 任务 1..... | 4 |
| 3.1.1 | 设计思想及存储单元分配..... | 4 |
| 3.1.2 | 流程图 | 5 |
| 3.1.3 | 源程序 | 6 |
| 3.1.4 | 实验步骤 | 6 |
| 3.1.5 | 实验记录与分析..... | 6 |
| 3.2 | 任务 2..... | 7 |
| 3.2.1 | 设计思想及存储单元分配..... | 7 |
| 3.2.2 | 源程序 | 8 |
| 3.2.3 | 实验步骤 | 14 |
| 3.2.4 | 实验记录与分析..... | 15 |
| 4. | 总结与体会 | 17 |
| 5. | 参考文献..... | 18 |

1. 实验目的与要求

- (1) 了解程序计时的方法以及运行环境对程序执行情况的影响。
- (2) 熟悉汇编语言指令的特点，掌握代码优化的基本方法。

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

应用场景介绍：以实验一任务四的背景为基础，只要顾客买走了网店中的一件商品，老板就需要重新获得全部商品的平均利润率。现假设在双十一零点时，SHOP1 网店中的“Bag”商品共有 m 件，有 m 个顾客几乎同时下单购买了该商品。请模拟后台处理上述信息的过程并观察执行的时间。

上述场景的后台处理过程，可以理解为在同一台电脑上有 m 个请求一起排队使用实验一任务四的程序。为了观察从第 1 个顾客开始进入购买至第 m 个顾客购买完毕之间到底花费了多少时间，我们让实验一任务四的功能三调整后的代码重复执行 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。功能三之外的其他功能不纳入到这 m 次循环体内（但可以保留不变）。

调整后的功能三的描述：

（1）在 SHOP1 中找到“Bag”商品，判断已售数量是否大于等于进货总数，若是，则回到功能一（1），否则将已售数量加 1。

（2）刷新全部商品的平均利润率。首先计算 SHOP1 中第一个商品的利润率 $PR1$ ，然后在 SHOP2 网店中寻找该商品，也计算其利润率 $PR2$ 。最后求出该商品的平均利润率 $APR = (PR1 + PR2) / 2$ ，并保存到 SHOP1 的利润率字段中。重复上述步骤，依次将每个商品的平均利润率计算出来。

请按照上述设想修改实验一任务四的程序，并将 m 和 n 值尽量取大（比如大于 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，比如秒级的时间间隔），以得到较明显的效果。

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！

3. 实验过程

3.1 任务 1

3.1.1 设计思想及存储单元分配

设计思想:

功能 1-4 的设计思想与实验 1 的相同, 只是增加了循环调用功能 3 操作和模拟商品销售的操作, 令循环次数足够大, 观察程序运行时间

寄存器分配:

CX: 控制循环;

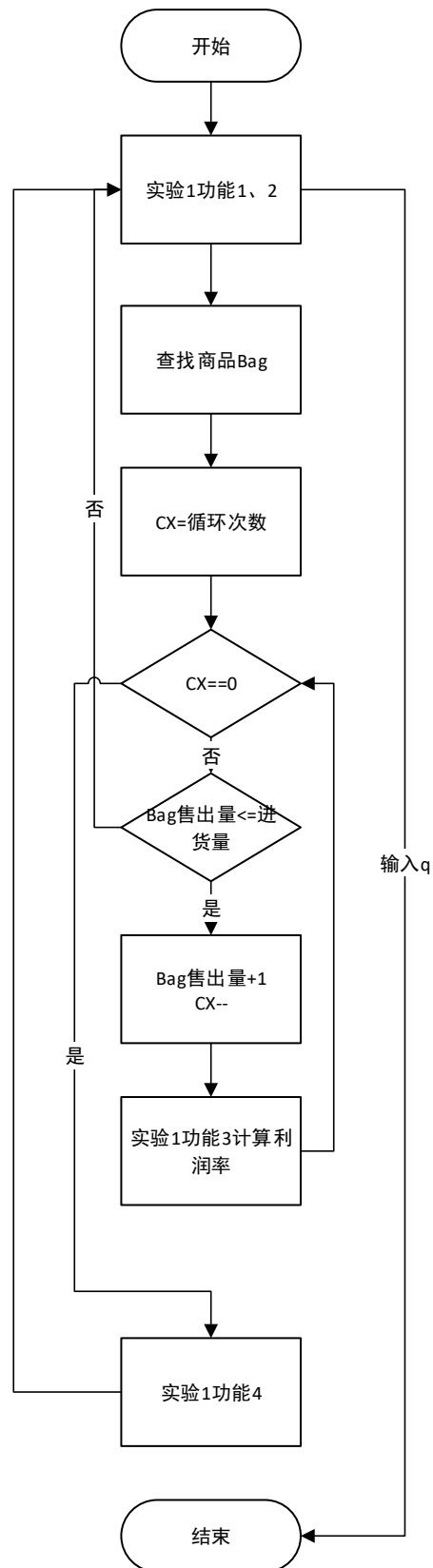
BX: 基址寄存器;

BP: 存放缓冲区基地址;

AL: 临时读取数据域的值;

AX, DX, SI: 临时寄存器;

3.1.2 流程图



3.1.3 源程序

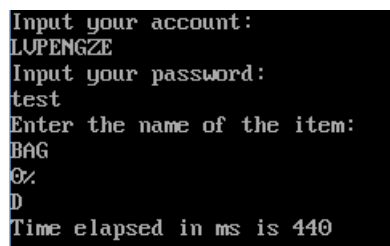
```
.....此之前为任务 1 功能 1、2 相关代码
MOV AX,0
CALL TIMER                      ;开始计时
MOV CX,M
R:
PUSH CX
.....任务 3 功能 3 查询商品相关代码
MOV CX,N
MOV BP,0
.....任务 3 功能 3 计算利润率相关代码
POP CX
DEC CX
JNZ R                          ;循环计算 M 次
.....任务 3 功能 4 相关代码
```

3.1.4 实验步骤

1. 准备上机环境，编辑、汇编、连接文件 T
2. 运行程序 T.EXE，观察
3. 输入姓名、密码、商品名后回车
4. 观察程序运行到第 m 次循环的最后一件商品利润率计算完后系统的运行时间
5. 修改运行次数，观察运行时间的变化

3.1.5 实验记录与分析

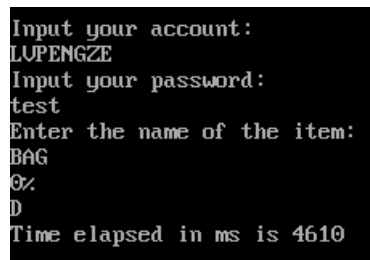
1. 当 m=1000，n=30 时程序的运行时间为 440ms，运行结果如图 3.1.1 所示



```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 440
```

3.1.1 运行结果

2. 当 m=10000，n=30 时程序的运行时间为 4610ms，运行结果如图 3.1.2 所示



```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 4610
```

3.1.2 运行结果

3.当 $m=20000$, $n=30$ 时程序的运行时间为 9110ms, 运行结果如图 3.1.3 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 9110
```

3.1.3 运行结果

4.当 $m=1000$, $n=60$ 时程序的运行时间为 880ms, 运行结果如图 3.1.4 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 880
```

3.1.4 运行结果

5.当 $m=1000$, $n=300$ 时程序的运行时间为 4340ms, 运行结果如图 3.1.5 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 4340
```

3.1.5 运行结果

3.2 任务 2

3.2.1 设计思想及存储单元分配

主要从减少对外部变量的访问, 在不溢出的前提下使用位数低的寄存器进行加减乘除运算, 运用 `movcx` 指令将字数据读取到双字寄存器中, 使用 32 位寄存器寻址, 减少冗余的指令, 使用宏汇编代替重复语句, 合理分配寄存器, 将乘除指令转换为移位指令等方便进行优化。

寄存器的分配与任务 1 相同

3.2.2 源程序

3.2.2.1 ASM 文件

```
.386
INCLUDE MACRO.LIB
STACK SEGMENT USE16 STACK
DB 200 DUP(0)
STACK ENDS
DATA SEGMENT USE16
BNAME DB 'LVPENGZE',0,0           ;管理员姓名
BPASS DB 'test',0,0               ;密码
AUTH DB ?                         ;标记登陆状态
N EQU 30
M EQU 20000
S1 DB 'SHOP1',0                   ;网店 1 名称，用 0 结束
GA1 DB 'PEN',7 DUP(0)             ;商品 1
    DW 35,56,70,50,?              ;进货价、销售价、进货总数、已售数量、利润率
GA2 DB 'BOOK',6 DUP(0)            ;商品 2
    DW 12,30,25,25,?
GA3 DB 'BAG',7 DUP(0)             ;商品 3
    DW 15,30,M,0,?
GAN DB N-3 DUP('Temp-Value',15,0,20,0,30,0,30,0,?,?);其他商品

S2 DB 'SHOP2',0                   ;网店 2 名称，用 0 结束,商品类型同网店 1
GB1 DB 'PEN',7 DUP(0)             ;商品 1
    DW 35,56,70,50,?              ;进货价、销售价、进货总数、已售数量、利润率
GB2 DB 'BOOK',6 DUP(0)            ;商品 2
    DW 12,30,25,25,?
GB3 DB 'BAG',7 DUP(0)             ;商品 3
    DW 15,30,M,0,?
GBN DB N-3 DUP('Temp-Value',15,0,20,0,30,0,30,0,?,?);其他商品

PR1 DW 0                          ;商店 1 利润率
PR2 DW 0                          ;商店 2 利润率
APR DW 0                          ;平均利润率
MSG1 DB 0AH,0DH,'Input your account:',0DH,0AH,'$'
MSG2 DB 'Input your password:',0DH,0AH,'$'
MSG3 DB 'WRONG ACCOUNT',0DH,0AH,'$'
MSG4 DB 'Enter the name of the item:',0DH,0AH,'$'

CRLF1 DB 0DH,0AH,'$'              ;回车换行
in_name DB 11                     ;姓名缓冲区
    DB ?
    DB 11 DUP(0)
in_pwd DB 7                       ;密码缓冲区
    DB ?
    DB 7 DUP(0)
in_goods DB 11                    ;商品名缓冲区
    DB ?
    DB 11 DUP(0)

DATA ENDS
CODE SEGMENT USE16
ASSUME CS:CODE,DS:DATA,SS:STACK
START:
MOV AX,DATA
```

```

MOV DS,AX
NEXT:                                     ;功能 1，提示登陆
WRITE MSG1
READ in_name
CRLF

WRITE MSG2
READ in_pwd
CRLF

MOV BL,in_name[2]
CMP BL,'q'
JE EXIT                                  ;输入'q'，退出程序

MOV BL,in_name[1]
CMP BL,0
JE UNLOGIN                              ;输入回车,转未登录
JMP LOGIN                               ;否则转登录

UNLOGIN:
MOV AH,0
MOV AUTH,AH                             ;标记登陆状态
JMP PROFIT                             ;转利润率计算

LOGIN:                                   ;功能 2，判断登陆
XOR ECX,ECX
MOV CL,in_name[1]
MOV ESI,0                               ;循环条件初始化
LOOP1:                                  ;循环比较姓名
MOV BL,in_name[ESI+2]
CMP BL,BNAME[ESI]
JNE ACCOUNTWRONG                       ;姓名不匹配转 ACCOUNTWRONG
INC ESI
DEC ECX
JNZ LOOP1

MOV BL,0
CMP BL,BNAME[ESI]
JNZ ACCOUNTWRONG                       ;输入的字符串是定义字符串的子集

XOR ECX,ECX
MOV CL,in_pwd[1]
MOV ESI,0                               ;循环条件初始化
LOOP2:
MOV BL,in_pwd[ESI+2]
CMP BL,BPASS[ESI]
JNE ACCOUNTWRONG                       ;密码不匹配转 ACCOUNTWRONG
INC ESI
DEC ECX
JNZ LOOP2

MOV BL,0
CMP BL,BPASS[ESI]
JNZ ACCOUNTWRONG                       ;输入为子集，转 ACCOUNTWRONG
JMP SUCCESS                             ;账号密码都相同转登陆成功

ACCOUNTWRONG:                           ;登陆失败
WRITE MSG3
JMP NEXT                                ;回到初始状态

SUCCESS:                                ;登陆成功

```

```

MOV AH,1
MOV AUTH,AH
JMP PROFIT

PROFIT:                                     ;功能 3， 计算利润率
WRITE MSG4
READ in_goods
CRLF

MOV BL,in_goods[1]
CMP BL,0
JE NEXT                                   ;输入回车,回到初始状态
MOV AX,0
CALL TIMER                               ;开始计时

MOV EDI,M
R:
MOV ECX,N
MOV EBX,0                               ;BX 基址寄存器
LOOP3:                                   ;第一层循环， 顺序查询 N 件商品
MOV ESI,0
MOV AL,in_goods[1]
CBW
MOV DX,AX                               ;DX 控制第二层循环
LOOP4:                                   ;第二层循环， 比较商品名称
    MOV AH,GA1[EBX][ESI]
    CMP AH,in_goods[ESI+2]
    JNE F1                               ;名称不同
    INC ESI                             ;变址++
    DEC DX
    JZ JUDGE                             ;找到商品,转 JUDGE
    JMP LOOP4

F1:
ADD EBX,20
DEC ECX
JNZ LOOP3
JMP PROFIT                               ;未找到商品， 重新输入商品名称
JUDGE:
CMP ESI,10                              ;商品名称恰好 10 个字符
JE F2

MOV AH,GA1[EBX][ESI]
CMP AH,0
JNE F1                                   ;商品名称为子集,属于未找到

F2:
MOV AH,AUTH
CMP AH,1
JE L1                                   ;登陆状态转 L1
JMP L2                                   ;未登录状态转 L2
L1:                                     ;计算商品利润率
MOV EAX,0
    LEA EBP,GA1[EBX][10]               ;S1 中商品信息基址

MOV AX,DS:[EBP+4]                       ;进货总数
CMP AX,DS:[EBP+6]
JLE NEXT                               ;进货数<售出数， 转 NEXT
MOV AX,1
ADD DS:[EBP+6],AX                       ;售出数+1

```

```

MOV SI,N
MOV EBP,0                ;0172H
R_PROFIT:
MOV SX EAX,WORD PTR GA1[EBP][10]
MOV SX EDX,WORD PTR GA1[EBP][14]
IMUL EAX,EDX
MOV EBX,EAX              ;ebx=成本

MOV SX EAX,WORD PTR GA1[EBP][12]
MOV SX EDX,WORD PTR GA1[EBP][16]
IMUL EAX,EDX             ;eax=收入

SUB EAX,EBX              ;EAX=利润
IMUL EAX,100
CDQ
IDIV EBX                 ;EAX=利润率
MOV PR1,AX

MOV SX EAX,WORD PTR GB1[EBP][10]
MOV SX EDX,WORD PTR GB1[EBP][14]
IMUL EAX,EDX
MOV EBX,EAX              ;ebx=成本

MOV SX EAX,WORD PTR GB1[EBP][12]
MOV SX EDX,WORD PTR GB1[EBP][16]
IMUL EAX,EDX             ;eax=收入

SUB EAX,EBX              ;EAX=利润
IMUL EAX,100
CDQ
IDIV EBX                 ;EAX=利润率
MOV PR2,AX
MOV WORD PTR GB1[EBP][18],AX;存储商店 2 商品利润率到商品利润字段中

ADD AX,PR1
SAR AX,1
MOV APR,AX
MOV WORD PTR GA1[EBP][18],AX ;总利润率
;CALL PRINTAX
;CRLF
;MOV AH,01H
;INT 21H                 ;调试，观察利润率是否计算正确
ADD EBP,20
DEC SI
JNZ R_PROFIT

DEC EDI
JNZ R                    ;循环计算 M 次

MOV AX,WORD PTR GA1[40][18]
JMP LEVEL

L2:
MOV AL,in_goods[1]
CBW
MOV SI,AX
MOV AH,'$'
MOV in_goods[SI+2],AH
WRITE in_goods[2]
CRLF                    ;输出商品名称
JMP NEXT                ;回到初始状态
LEVEL:                  ;功能 4，商品等级判断

```

```

MOV SX EAX,WORD PTR GA3[18]      ;输出 BAG 的利润率 025DH
MOV EBX,10
CALL PRINTAX
OUT1 '%'
CRLF
MOV AX,WORD PTR GA3[18]
CMP AX,90
JGE _A      ;利润率大于 90%
CMP AX,50
JGE _B      ;利润率大于 50%
CMP AX,20
JGE _C      ;利润率大于 20%
CMP AX,0
JGE _D      ;利润率大于 0%
JMP _F

_A:
OUT1 'A'
CRLF
JMP T
_B:
OUT1 'B'
CRLF
JMP T
_C:
OUT1 'C'
CRLF
JMP T
_D:
OUT1 'D'
CRLF
JMP T
_F:
OUT1 'F'
CRLF
JMP T
T:
MOV AX,1
CALL TIMER      ;结束计时
JMP NEXT
EXIT:
MOV AH,4CH
INT 21H      ;退出程序

;-----以 10 进制输出 AX 中的无符号整数
PRINTAX PROC
MOV EBX,10
PUSH CX
PUSH EDX      ;保护现场
XOR CX,CX      ;计数器清 0
CMP EAX,0
JNL _LOP1
NOT EAX
ADD EAX,1
PUSH EAX
MOV AH,2
MOV DL','
INT 21H      ;输出负号
POPEAX
_LOP1:      ;(EAX)除以 P，所得商->EAX，余数入栈，CX++，记录余
数个数
XOR EDX,EDX

```

```

    DIV EBX
    PUSH DX
    INC CX
    OR EAX,EAX
    JNZ _LOP1
    _LOP2:                                     ;从栈中弹出一位 P 进制数，并将该数转换成 ASCII 码后输出
    POP AX
    CMP AL,10
    JB _L1
    ADD AL,7
    _L1:                                       ;输出 P 进制数
    ADD AL,30H
    MOV DL,AL
    MOV AH,2
    INT 21H
    LOOP _LOP2
    POP EDX
    POP CX                                   ;恢复现场
    RET
PRINTAX ENDP
;-----
;时间计数器(ms),在屏幕上显示程序的执行时间(ms)
;使用方法:
;   MOV  AX, 0      ;表示开始计时
;   CALL TIMER
;   ... ..   ;需要计时的程序
;   MOV  AX, 1
;   CALL TIMER      ;终止计时并显示计时结果(ms)
;输出: 改变了 AX 和状态寄存器
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

```

```

_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

CODE ENDS
END START

```

3.2.2.2 LIB 文件

```

READ    MACRO A
      LEA DX,A
      MOV AH,10
      INT 21H
      ENDM
WRITE    MACRO A
      LEA DX,A
      MOV AH,9
      INT 21H
      ENDM
CRLF MACRO
      MOV AH,2
      MOV DL,0AH
      INT 21H
      MOV DL,0DH
      INT 21H
      ENDM
OUT1 MACRO A
      MOV DL,A
      MOV AH,2
      INT 21H
      ENDM
STACK0  MACRO A
STACK    SEGMENT USE16 PARA STACK 'STACK'
      DB A
STACK    ENDS
      ENDM

```

3.2.3 实验步骤

1. 准备上机环境，编辑、汇编、连接文件 T1
2. 运行程序 T1.EXE，观察

3. 输入姓名、密码、商品名后回车
4. 观察程序运行到第 m 次循环的最后一件商品利润率计算完后系统的运行时间
5. 修改运行次数，与任务 1 的运行时间比较

3.2.4 实验记录与分析

1. 当 $m=1000$, $n=30$ 时程序的运行时间为 330ms, 运行结果如图 3.2.1 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
bag
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 330
```

3.2.1 运行结果

2. 当 $m=10000$, $n=30$ 时程序的运行时间为 3460ms, 运行结果如图 3.2.2 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 3460
```

3.2.2 运行结果

3. 当 $m=20000$, $n=30$ 时程序的运行时间为 6870ms, 运行结果如图 3.2.3 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
0%
D
Time elapsed in ms is 6870
```

3.2.3 运行结果

4. 当 $m=1000$, $n=60$ 时程序的运行时间为 660ms, 运行结果如图 3.2.4 所示


```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
Q%
D
Time elapsed in ms is 660
```

3.2.4 运行结果

5.当 m=1000，n=300 时程序的运行时间为 3240ms，运行结果如图 3.2.5 所示

```
Input your account:
LUPENGZE
Input your password:
test
Enter the name of the item:
BAG
Q%
D
Time elapsed in ms is 3240
```

3.2.5 运行结果

6.经过优化后各个数据规模下提升的速度如表 3.2.1 所示

表 3.2.1 优化率

| | 数据规模 | | 原耗时/ms | 优化后耗时/ms | 减少时间/ms | 优化率 |
|---|-------|-----|--------|----------|---------|-------|
| | m | n | | | | |
| 1 | 1000 | 30 | 440 | 330 | 110 | 25% |
| 2 | 10000 | 30 | 4610 | 3460 | 1150 | 24.9% |
| 3 | 20000 | 30 | 9110 | 6870 | 2240 | 24.6% |
| 4 | 1000 | 60 | 880 | 660 | 220 | 25% |
| 5 | 1000 | 300 | 4340 | 3240 | 1100 | 25.3% |

经过多组测试可知，优化后程序运算速度提升了大概 25%左右，提升效率还是非常大的。

4. 总结与体会

这次实验相对于第一次实验来说较为简单，主要的框架在实验 1 中已经实现，工作量会少一些。而且，通过这次实验，我学到了许多新的知识。

任务 1 让我明白了随着数据规模的增大，程序的运行速度会变慢许多，对于这个问题来说运行时间是与 $m*n$ 成正比的，当规模达数十万级别时，程序的运行速度会增加到秒级，可以想象，对于数据规模更大的应用，比如淘宝，12306 等，将面临这更大挑战，如何在短时间内给出运算结果是十分重要的。

这个问题在任务 2 中给出了一种解决方案——代码优化，通过减少冗余语句、使用 32 位寄存器寻址、换用更高效的指令等方式我成功地使程序地运行时间减少地 25%左右，有了明显的进步。此外，还有别的方法可以使程序更高效，比如运用更高效的算法、使用性能更优的计算机、采用更合适的数据结构等方法将大大减少程序运行的时间，这在实际应用中有着重要的指导作用。

5. 参考文献

[1] 王元珍,曹忠升,韩宗芬.80X6 汇编语言程序设计.版本(第 1 版).武汉.华中科技大学出版社,2005 年 4 月.