

華中科技大学

课程实验报告

课程名称：汇编语言程序设计实验

实验名称：实验三 模块化程序设计

实验时间：2018-4-16, 14: 00-17: 30 实验地点：南一楼

指导教师：朱虹

专业班级：计算机科学与技术 1601 班

学 号：U201614531 姓 名：刘本嵩

同组学生：罗皓云 报告日期：2018 年 4 月 25 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.....	1
2 实验内容.....	2
2.1 任务 1：宏与子程序设计.....	2
2.2 任务 2：在 C 语言程序中调用汇编语言实现的函数.....	3
3 实验过程.....	4
3.1 任务 1.....	4
3.1.1 实验步骤.....	4
3.1.2 模块源代码.....	4
3.1.3 实验记录与分析.....	20
3.2 任务 2.....	22
3.2.1 实验步骤.....	22
3.2.2 源代码.....	23
3.2.3 实验记录与分析.....	32
4 总结与体会.....	37
5 参考文献.....	38

1 实验目的与要求

1. 掌握子程序设计的方法与技巧，熟悉子程序的参数传递方法和调用原理；
2. 掌握宏指令、模块化程序的设计方法；
3. 掌握较大规模程序的合作开发与调试方法；
4. 掌握汇编语言程序与 C 语言程序混合编程的方法；
5. 熟悉 C 编译器的基本优化方法；
6. 了解 C 语言编译器的命名方法，主、子程序之间参数传递的机制。

2 实验内容

2.1 任务 1：宏与子程序设计

进一步修改与增强实验一任务四的网店商品信息管理程序的功能，主要调整功能三。

1. 调整后的功能三的描述

(1) 首先显示一个功能菜单（格式自行定义。若是未登录状态，只显示菜单“1”和“6”）：

1=查询商品信息，2=修改商品信息，3=计算平均利润率，

4=计算利润率排名，5=输出全部商品信息，6=程序退出。

输入 1-6 的数字进入对应的功能。

(2) 查询商品信息

提示用户输入要查询的商品名称。若未能在第一个网店中找到该商品，重新提示输入商品名称。若只输入回车，则回到功能三(1)。

找到该商品之后，按照：“SHOP1，商品名称，销售价，进货总数，已售数量”顺序显示该商品的信息，同时还要将“SHOP2”中该商品的信息也显示出来。显示之后回到功能三(1)。

(3) 修改商品信息

提示用户输入要修改信息的商品名称（先指定网店名称）。[若把接下来的处理步骤写成子程序，则网店名称和商品名称（或其偏移地址）就是子程序的入口参数，是否找到、是否是回车或者修改成功的信息是出口参数]。若未能在指定网店中找到该商品，重新提示输入网店名称和商品名称。若只输入回车，则回到功能三(1)。

找到该商品之后，按照：进货价，销售价，进货总数的次序，逐一先显示原来的数值，然后输入新的数值（若输入有错，则重新对该项信息进行显示与修改。若直接回车，则不修改该项信息）。

如：进货价：25» 24 //符号“»”仅作为分隔符，也可以选择其他分隔符号

销售价: 46» 5A6 //输入了非法数值, 下一行重新显示和输入

销售价: 46» 56

进货总数: 30» //直接回车时, 对这项信息不做修改

当对三项信息都处理完毕后, 回到功能三 (1)。

(4) 计算平均利润率

首先计算 SHOP1 中第一个商品的利润率 PR1, 然后在 SHOP2 网店中寻找到该商品, 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。回到功能三 (1)。

(5) 计算利润率排名

对 SHOP2 中的每个商品按照平均利润率的大小排名, 排名信息存放到 SHOP2 中商品的利润率字段中。回到功能三 (1)。

(6) 输出全部商品信息

将 SHOP1 和 SHOP2 中的所有商品信息显示到屏幕上, 包括平均利润率和排名 (替代了商品原有的利润率字段)。具体的显示格式自行定义 (可以分网店显示, 也可以按照商品排名显示, 等等, 显示方式可以作为子程序的入口参数)。回到功能三 (1)。

2.2 任务 2：在 C 语言程序中调用汇编语言实现的函数

对于任务 1 的程序进行改造, 主控程序、以及输入输出较多的某一个功能 (如功能 (1)、(2)、(5) 中的某一个) 用 C 语言实现, 其他功能用独立的汇编语言子程序的方式实现; 在 C 语言程序中调用汇编语言子程序。

3 实验过程

3.1 任务 1

3.1.1 实验步骤

1. 根据任务的要求完成相应程序模块的编写，并声明公共符号以供另一个模块使用。
2. 测试自己编写的模块的正确性，经过编译，并保证没有错误。
3. 将自己编写的模块同队友编写的模块同进行编译、连接，并保证可以正确运行。

3.1.2 模块源代码

```
name module2
public calc_profit_rate, calc_rank, dump_303
.model small
.stack 256
.data
user_name db 'bensongliu',0
user_pswd db 'test',0,0
n equ 30
s1 db 'shop1',0
ga1 db  'pen$', 6 dup(0)
        dw  35,56,70,25,?
ga2 db  'book$', 5 dup(0)
        dw  12,30,25,5,?
gan db n-2 dup( 'tempvalue$',15,0,20,0,30,0,2,0,?,?) ;网店名称,用0结束
s2 db 'shop2',0      ;网店名称,用0结束
gb1 db  'book$', 5 dup(0);商品名称
        dw  12,28,20,15,?
gb2 db  'pen$', 6 dup(0) ;商品名称
        dw  35,50,30,24,?
gbn db n-2 dup( 'tempvalue$',15,0,20,0,30,0,2,0,?,?) ;商品名称
name_buffer db 12, ?, 12 dup(0)
pswd_buffer db 8, ?, 8 dup(0)
product_buffer db 12, ?, 12 dup('$')
shop_buffer    db 0
buffer1 dw 8,?,8 dup(0)
in_cho db 2, ?, 0, 0
pr1 dw 0,0
pr2 dw 0,0
```

```
pr_sum dw 0,0
logged db 0
login_ok db ?
mark db ?
must_return db 0
buffer3 db 0ah,0dh,'$'
buffer4 db 0h

menu_prompt db 'choice ~',10
db '1, query product info',10
db '2, edit product',10
db '3, calc profit',10
db '4, calc rank',10
db '5, print calc-ed info',10
db '6, exit',10
db '?','$'

str1 db 12 dup(?)
str2 db 'User: $'
str3 db 'Password: $'
str4 db 'Exception occurred.$'
str5 db 'Permission denied.$'
str6 db 'Product ?$'
str7 db 'Shop (1/2) ?$'
str8 db 'Buy price $'
str9 db 'Sell price $'
str10 db 'On stock $'
str11 db '>> $'

str12 db 'commodity information:', '$'
str13 db 'shop1', '$'
str14 db 'shop2', '$'
str15 db 'Buy price ', '$'
str16 db 'Sell price ', '$'
str17 db 'On stock ', '$'
str18 db 'Sold ', '$'
str19 db 'Profit ', '$'
str20 db 'Rank ', '$'

.code
rlib_macro_print macro buffer
    lea dx,buffer
    mov ah,9
    int 21h
```

```
endm
rlib_macro_readstr macro buffer
    lea dx,buffer
    mov ah,10
    int 21h
endm

; Written by Bensong Liu <root@recolic.net>
; Partner := Haoyun Luo
calc_profit_rate:
    call calc_profit_rate_impl
        lea dx,str17
    rlib_macro_print buffer3
    jmp tmp112

; Written by Bensong Liu <root@recolic.net>
; Partner := Haoyun Luo
calc_rank:
    call calc_rank_impl
        lea dx,str18
    rlib_macro_print buffer3
    jmp tmp112
tmp111:
    call tmp167
        push cx
    mov cx,[di+12]
    rlib_macro_print buffer3
    rlib_macro_print str16
        mov cx,ax
    mov dx,16
        pop cx
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print str17
        mov ax,di
    mov dx,16
    mov ax,[ax+14]
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print str18
        xor dx,dx
        or dx,di
    mov ax,[dx+16]
    mov dx,16
```

```
call tmp167
call locate_product
    jmp tmp129
```

; Written by Bensong Liu <root@recolic.net>

; Partner := Haoyun Luo

calc_profit_rate_Impl proc

push ax

push bx

push cx

push dx

push di

push si

lea di, str20

tmp136:

mov ax,[ga1+12]

mov bx,[ga1+16]

imul bx

xor cx, cx

mov cx,ax

inc ax

mov ax,[ga1+10]

mov bx,[ga1+14]

imul bx

xor bx, bx

mov bx,ax

mov ax,cx

mov cx,bx

sub ax,bx

cwde

mov ebx,100

imul eax,ebx

mov bx,cx

idiv bx

mov word ptr[pr1],ax

call locate_product

jmp tmp137

tmp137:

mov ax,[si+12]

mov bx,[si+16]

```
imul bx
    xor cx, cx
    mov cx,ax
        mov ax,di
    mov ax,[si+10]
    mov bx,[si+14]
imul bx
    xor bx, bx
    mov bx,ax
    mov ax,cx
    mov cx,bx
    sub ax,bx
    cwde
    mov ebx,100
    imul eax,ebx
    mov bx,cx
    idiv bx
    mov word ptr[pr2],ax

    mov bx,offset pr1
    mov cx,word ptr[bx]
    add ax,cx
    sar ax,1
    mov word ptr[di+18],ax
    add di,20
    mov bx,offset s2
    cmp di,bx
    jge term_307
    jmp tmp136
term_307:
    pop ax
    pop bx
    pop dx
    pop cx
    pop bx
    pop ax
    ret
calc_profit_rate_impl endp
; Written by Bensong Liu <root@recolic.net>
; Partner := Haoyun Luo
calc_rank_impl proc
    push ax
    push bx
    push cx
```

```
push dx
push di
push si
    lea bx, s2
    lea di, ga1
tmp140:
    mov dx,1
    mov cx,[di+18]
    mov si,offset ga1

tmp141:
    add si,20
    mov ax,[si+18]
    cmp si,bx
    jl tmp125
    jmp tmp142
tmp125:
    cmp cx,ax
    jge tmp141
    inc dx
    jmp tmp141

tmp142:
    mov [si+18],dx
    call locate_product
    add di,20
    cmp di,bx
    jl tmp140
```

```
term_308:
    pop si
    pop di
    pop dx
    pop cx
    pop bx
    pop ax
    ret
calc_rank_Impl endp
```

; Written by Bensong Liu <root@recolic.net>

; Partner := Haoyun Luo

dump_303 proc

```
rlib_macro_print str12
rlib_macro_print buffer3
rlib_macro_print str13
rlib_macro_print buffer3
rlib_macro_print product_buffer+2
rlib_macro_print buffer3
rlib_macro_print str15
mov dx,16
mov ax,[di+10]
jmp tmp111
rlib_macro_print product_buffer+2
rlib_macro_print buffer3
    cmp eax,7ffff
    jc tmp176
    dec cx
    jne tmp174
    jnz tmp172
    cmp dx,16
    ja tmp176
```

```
tmp129:
rlib_macro_print buffer3
rlib_macro_print buffer3
rlib_macro_print str14
rlib_macro_print buffer3
rlib_macro_print product_buffer+2
rlib_macro_print buffer3
rlib_macro_print str15
mov ax,[si+10]
mov dx,16
call tmp167
rlib_macro_print buffer3
rlib_macro_print str16
mov ax,[si+12]
mov dx,16
call tmp167
rlib_macro_print buffer3
rlib_macro_print str17
mov ax,[si+14]
mov dx,16
call tmp167
rlib_macro_print buffer3
rlib_macro_print str18
mov ax,[si+16]
```

```
    mov dx,16
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print buffer3
    ret
dump_303 endp
```

; Written by Bensong Liu <root@recolic.net>

; Partner := Haoyun Luo

find_304 proc

```
    push ax
    push bx
    push cx
    push dx
    push si
```

tmp10:

```
    rlib_macro_print buffer3
    rlib_macro_print str6
    rlib_macro_readstr product_buffer
    rlib_macro_print buffer3
    cmp product_buffer[1],0
    je tmp147
    jmp tmp9
```

tmp147:

```
    mov [mark],-1
    jmp term_306
; search the product
tmp9: mov di,offset s1
    add di,6
    mov bl,[product_buffer+1]
    mov bh,0
    mov si,offset product_buffer
    cmp bx,10
    je tmp149
    mov dl,[di+bx]
    cmp dl,0
    je tmp7
```

tmp149:

```
    mov ch,[si+bx+1]
    mov cl,[di+bx-1]
```

```
        cmp cl,0
        je tmp7
        cmp ch,cl
        jne tmp7
        jmp tmp8

tmp8:
        cmp bx,1
        je tmp6
        dec bx
        jmp tmp149

tmp7:
        and ch, ch
        xor bx, bx
        mov bl,[product_buffer+1]
        mov cl,[logged]
        add di,20
        dec cx
        lea cx, s2
        cmp cx,di
        je tmp10
tmp154:
        jmp tmp149

tmp6:
        mov [mark],1
        jmp term_306
term_306:
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        ret
find_304 endp

; Written by Bensong Liu <root@recolic.net>
; Partner := Haoyun Luo
locate_product proc
        push ax
        push bx
        push cx
        push dx
        push di
```

```
push si
mov bx,10

    mov cx,'q'
    lea si,gb1
    inc bx
loop302:
    mov cl,[di+bx-1]
    mov ch,[si+bx-1]
    dec bx
    sub ch,cl
    jnz tmp161
tmp160:
    cmp bx,1
    je tmp162
    jmp loop302
tmp161:
    add si,20
    mov bx,11
    jmp loop302
tmp162:
    pop ax
    pop di
    pop dx
    pop cx
    pop bx
    pop ax
    ret
locate_product endp
```

start:

```
    mov ax,data
    mov ds,ax
    lea dx,str2
    mov ah,9
        int 21h
    lea dx,offset name_buffer
    mov ah,10
        int 21h
    lea dx,buffer3
    mov ah,9
    int 21h
```

```
        cmp name_buffer[1],0
        je tmp14
        mov bh,name_buffer+2
        cmp bh,'q'
        je tmp17
        lea dx,str3
        mov ah,9
        int 21h
        lea dx,offset pswd_buffer
        mov ah,10
        int 21h
        jmp tmp13
        mov ax, 4c00h
        int 21h
```

```
tmp18:mov ax, 4c00h
int 21h
```

```
tmp17: mov bh,name_buffer+1
        cmp bh,1
        je tmp18
```

```
tmp16:    mov ax, 4c00h
        int 21h
```

```
tmp14:
        mov buffer4,0
        jmp tmp112
```

```
tmp13:
        mov di,offset name_buffer
        mov bx,di
        inc bx
        mov si,10
        mov bx,[bx]
        mov bh,0
        cmp si,bx
        jne tmp11
```

```
tmp24:
```

```
    mov ch,1[di][bx]
    mov cl,[user_name+bx-1]
    cmp ch,cl
    jne tmp11
    dec si
    jne tmp24
    jmp tmp12
```

```
tmp12:    mov di,offset pswd_buffer
```

```
    mov bx,di
    inc bx
    mov si,4
    mov bx,[bx]
    mov bh,0
    cmp si,bx
    jne tmp11
```

```
tmp23:
```

```
    mov ch,1[di][bx]
    mov cl,[user_pswd+bx-1]
    cmp ch,cl
    jne tmp11
    dec si
    jne tmp23
    mov buffer4,1h
    jmp tmp112
```

```
tmp11: lea dx,buffer3
```

```
    mov ah,9
    int 21h
    lea dx,str5
    mov ah,9
    int 21h
    lea dx,buffer3
    mov ah,9
    int 21h
    jmp start
```

```
tmp112:
```

```
    cmp buffer4,0
    jz tmp113
    jmp tmp114
```

```
tmp113:
```

```
rlib_macro_print buffer3
rlib_macro_print menu_prompt
rlib_macro_print buffer3
    rlib_macro_print menu1
rlib_macro_print buffer3
    rlib_macro_print menu6
rlib_macro_print buffer3
    rlib_macro_print menu7
rlib_macro_readstr in_cho
rlib_macro_print buffer3
lea di,in_cho+2
mov al,[di]
cmp al,'1'
jz query_products
cmp al,'6'
jz tmp129
jmp start

tmp114:rlib_macro_print buffer3
    rlib_macro_print menu_prompt
rlib_macro_print buffer3
    rlib_macro_readstr in_cho
lea di,in_cho+2
mov al,[di]
cmp al,'1'
jz query_products
cmp al,'2'
jz edit_input
cmp al,'3'
jz calc_profit_rate
cmp al,'4'
jz calc_rank
cmp al,'5'
jz tmp122
cmp al,'6'
jz tmp129
rlib_macro_print str4
rlib_macro_print buffer3
jmp tmp114
```

```
tmp122:rlib_macro_print str12
rlib_macro_print buffer3
rlib_macro_print str13
rlib_macro_print buffer3
mov cx,3
```

```
lea di,s1+6
jmp tmp123

tmp123:rlib_macro_print buffer3
    rlib_macro_print [di]
    rlib_macro_print buffer3
    rlib_macro_print str15
    mov ax,[di+10]
    mov dx,16
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print str16
    mov ax,[di+12]
    mov dx,16
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print str17
    mov ax,[di+14]
    mov dx,16
    call tmp167
    rlib_macro_print buffer3
    rlib_macro_print str18
    mov ax,[di+16]
    mov dx,16
    call tmp167
    rlib_macro_print buffer3
    cmp logged,0
    jz tmp124
    jmp tmp125

tmp124:rlib_macro_print str19
    jmp tmp126

tmp125:rlib_macro_print str20
    jmp tmp126

tmp126:mov ax,[di+18]
    mov dx,16
    call tmp167
    rlib_macro_print buffer3

tmp127:add di,20
    dec cx
    jnz tmp123
    cmp logged,1
    jz tmp112

tmp128:mov logged,1
```

```
rllib_macro_print str14
rllib_macro_print buffer3
mov cx,3
lea di,gb1
jmp tmp123
```

```
tmp129: mov ah,4ch
        int 21h
```

```
tmp163 proc
    push cx
    push edx
    xor cx,cx
tmp164: xor edx,edx
    div ebx
    push dx
    inc cx
    or eax,eax
    jnz tmp164
tmp165: pop ax
    cmp al,10
    jb tmp166
    add al,7
tmp166:     add al,30h
    mov [si],al
    inc si
    loop tmp165
    pop edx
    pop cx
    ret
tmp163 endp
```

```
tmp167 proc far
    push ebx
    push si
    lea si,str1
    cmp dx,32
    je tmp168
    movsx eax,ax
```

```
tmp168:      or eax, eax
            jns tmp169
            neg eax
            mov byte ptr[si], '-'
            inc si
tmp169:      mov ebx, 10
            call tmp163
            mov byte ptr[si], '$'
            lea dx, str1
            mov ah, 9
            int 21h
            pop si
            pop ebx
            ret
tmp167 endp
```

```
tmp170 proc far
    push ebx
    mov eax, 0
    mov login_ok, 0
    mov bl, [si]
    cmp bl, '+'
    je tmp171
    cmp bl, '-'
    jne tmp173
    mov login_ok, 1
tmp171: dec cx
    jz tmp176
tmp172: inc si
    mov bl, [si]
tmp173: cmp bl, '0'
    jb tmp176
    cmp bl, '9'
    ja tmp176
    sub bl, 30h
    movzx ebx, bl
    imul eax, 10
    jo tmp176
    add eax, ebx
    jo tmp176
    js tmp176
    jc tmp176
    dec cx
    jnz tmp172
```

```
    cmp dx,16
    jne tmp174
    cmp eax,7ffffh
    ja tmp176
tmp174:cmp login_ok,1
    jne tmp175
        neg eax
tmp175:pop ebx
    ret
tmp176:mov si,-1
    jmp tmp175
tmp170 endp
```

```
end start
end
```

3.1.3 实验记录与分析

1. 首先完成模块二的 8086asm 的编写。
2. 如下图所示，汇编链接运行。

The screenshot shows a terminal window with the following text:

```
C:\>make m2
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1991. All rights reserved.

Invoking: ML.EXE /I. /Zm /c /Ta m2.asm

Microsoft (R) Macro Assembler Version 6.00
Copyright (C) Microsoft Corp 1981-1991. All rights reserved.

Assembling: m2.asm

Microsoft (R) Segmented Executable Linker Version 5.20.034 May 24 1991
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.

C:\>
```

3. 分别测试 3 个功能。按照要求，3/4 功能没有输出，功能 5 的输出取决于预设的数值，其他功能由模块 1 完成，不进行测试。

```
OS 1, query product info
--2, edit product
IX 3, calc profit
LS 4, calc rank
EC 5, print calc-ed info
EC
OS 6, exit
OP ? 3
-- ON
IX Choice ~
LS
ID 1, query product info
EC
SA 2, edit product
EC
EC 3, calc profit
EC
OS 4, calc rank
OP
-- 5, print calc-ed info
ON 6, exit
TX
LS ? 4
ID
EC
EC Choice ~
EC
OS 1, query product info
OP 2, edit product
-- 3, calc profit
ON 4, calc rank
IX 5, print calc-ed info
EC
EC 6, exit
V: 6, exit
EC
EC
```

```
Sold 4
Rank 1

PEN
Buy price 31
Sell price 32
On stock 33
Sold 44
Rank 1

TempValue
Buy price 15
Sell price 20
On stock 30
Sold 2
Rank 2

Choice ~
1, query product info
2, edit product
3, calc profit
4, calc rank
5, print calc-ed info
6, exit
? -
```

4.模块 1 的测试如下，使用模块 1 提供的主控程序。

```
C:\>MY9
please input your name:LUO
please input password:12300
LAND OK!
1
1. inquire commodity information
2. modify commodity information
3. average profit rate
4. rank profit rate
5. All commodity information
6. exit the system
```

***PLEASE INPUT THE NUMBER:

```
***PLEASE INPUT THE NUMBER:
1
PLEASE INPUT THE NAME OF THE GOODS:PEN
SHOP1
PEN 35 56 1000 0
SHOP2
PEN 35 50 6 5
```

***PLEASE INPUT THE NUMBER:

```
2
PLEASE INPUT 1(SHOP1) OR 2(SHOP2):1
PLEASE INPUT THE NAME OF THE GOODS:PEN
SHOP1
PEN 35 56 1000 0 CHANGE:
35>>15
56>>16
1000>>17
0>>18
```

```
***PLEASE INPUT THE NUMBER:
1
PLEASE INPUT THE NAME OF THE GOODS:PEN
SHOP1
PEN 15 16 17 18
SHOP2
PEN 35 50 6 5
```

3.2 任务 2

3.2.1 实验步骤

- 根据任务 2 的要求编写 C 语言程序.

2. 修改模块化的汇编语言程序的符号，使其适合与 C 编译器生成的汇编代码链接。修改汇编程序中的调用，学习 watcomC 的标准库接口写法，使其能够调用 libc。为了符合 wlink 的接口规定以便调用其提供的 glibc，这里按照由当年的 watcom 官方文档提供的指导进行符号定义和汇编重写。例如，使用 80387 提供的浮点数运算指令重写 profit 计算部分，使用 wcc 指定的传参方式和返回方式，使用 C 语言内存布局。其中，由汇编实现不包含 IO 的利润率计算、排名计算等 CPU 密集型功能，并与 C 语言实现的 IO 程序进行链接。

3. make

3.2.2 源代码

```
~/t/d/exp32 >>> make report
echo ';;;;;;main.c' && cat main.c && echo ';;;;;;afx.h' && cat afx.h && echo ';;;;;;impl.asm' && cat impl.asm && echo
';;;;;;makefile' && cat makefile
;;;;;;main.c
/**
 * This source is extremely unsafe. You must not use it under any consequence
 *   except hust homework.
 */

#include "afx.h"

char user[32] = "bensong liu";
char pswd[32] = "test";
char shopA[16] = "SHOP1";
char shopB[16] = "SHOP2";

char products_name[PRODUCT_NUM][32] = {"pen", "book", "scala"};
int input_price[PRODUCT_NUM] = {35, 22, 61};
int output_price[PRODUCT_NUM] = {11, 28, 1000};
float profit_table[PRODUCT_NUM] = {0,0,0};
int rank_table[PRODUCT_NUM] = {-1,-1,-1};

bool logged_in = false;

void echo(const char *txt) {
    printf("%s\r\n", txt);
}
```

```
void readline_unsafe(char *out) {
    char c;
    int pos = 0;
    while(true) {
        c = getchar();
        if(c == '\r' || c == '\n')
            break;
        out[pos] = c;
        ++pos;
    }
    out[pos] = '\0';
}

bool login() {
    char buffer[32] = "";
    int cter;

again:
    printf("Your name please: ");
    readline_unsafe(buffer);

    if(buffer[0] == '\0')
        goto again;
    if(buffer[0] == 'q')
        exit(0);
    if(!strcmp(buffer, user)) {
        echo("Permission denied.");
        goto again;
    }

    printf("Your password please: ");
    readline_unsafe(buffer);
    if(!strcmp(buffer, pswd)) {
        echo("Permission denied.");
        goto again;
    }

    echo("Passed.");
    return true;
}

void func1_ls() {
    int cter;
    for(cter = 0; cter < PRODUCT_NUM; ++ctер) {
        printf("%s > InputPrice %d OutputPrice %d.\r\n", products_name[ctер], input_price[ctер], output_price[ctер]);
    }
}
```

```
}

void func2_edit() {
    char name[32];
    int input, output;
    int pos;
    echo("Give '$name $inputPrice $outputPrice' please.");
    scanf("%s %d %d", name, &input, &output);

    pos = search_product(name);
    if(pos == -1) {
        echo("Not found error.");
        return;
    }

    input_price[pos] = input;
    output_price[pos] = output;
    echo("Done.");
}

void func5_print_profit_and_rank() {
    int cter;
    for(cter = 0; cter < PRODUCT_NUM; ++cter) {
        printf("%s: profit mark %c rank %d\r\n", products_name[cter], profit_mark_of(cter), rank_table[cter]);
    }
}

void func6_exit() {
    exit(0);
}

int main() {
    char option[16];
    login(); /*Always true*/
    while(true) {
        echo("");
        echo("func1_ls");
        echo("func2_edit");
        echo("func3_calc_profit");
        echo("func4_calc_rank (depend: func3)");
        echo("func5_print_profit_and_rank (depend: func4)");
        echo("func6_exit");
        printf(" 1-6 ? ");
    }
}
```

```
readline_unsafe(option);
switch(option[0]) {
    case '1':
        func1_ls();
        break;
    case '2':
        func2_edit();
        break;
    case '3':
        func3_fill_profit_table();
        break;
    case '4':
        func4_fill_rank_table();
        break;
    case '5':
        func5_print_profit_and_rank();
        break;
    case '6':
        func6_exit();
        break;
}
}

};;;;;;;;;;;;;;;;;;afx.h
#ifndef AFX_H
#define AFX_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/* C99 only */
typedef int bool;
#define true 1
#define false 0

#define streq !strcmp

#define PRODUCT_NUM 3

void func1_ls();
void func2_edit();
void func3_fill_profit_table();
void func4_fill_rank_table();
void func5_print_profit_and_rank();
```

```
void func6_exit();
char profit_mark_of(int index);
int search_product(const char *name);

#endif
;;;;;;impl.asm
.387

PUBLIC  func3_fill_profit_table_
PUBLIC  func4_fill_rank_table_
PUBLIC  profit_mark_of_
PUBLIC  search_product_
EXTRN   __STK:BYTE
EXTRN   _output_price:BYTE
EXTRN   _input_price:BYTE
EXTRN   FIDRQQ:BYTE
EXTRN   _profit_table:BYTE
EXTRN   FIWRQQ:BYTE
EXTRN   _rank_table:BYTE
EXTRN   _products_name:BYTE
EXTRN   strcmp_:BYTE
EXTRN   __8087:BYTE
EXTRN   __init_87_emulator:BYTE
EXTRN   _small_code_:BYTE

DGROUP    GROUP CONST,CONST2,_DATA
_TEXT     SEGMENT BYTE PUBLIC USE16 'CODE'
ASSUME CS:_TEXT, DS:DGROUP, SS:DGROUP

func3_fill_profit_table_:
    mov     ax,0cH
    call    near ptr __STK
    push   bx
    push   dx
    push   si
    push   bp
    mov    bp,sp
    sub    sp,2
    xor    bx,bx

tmp1:
    mov    si,bx
    shl    si,1
    mov    ax,word ptr _output_price[si]
    cwd
    idiv   word ptr _input_price[si]
    dec    ax
    mov    word ptr -2[bp],ax
```

```
fld    word ptr -2[bp]
mov    si,bx
shl    si,1
shl    si,1
fstp   dword ptr _profit_table[si]
inc    bx
nop
fwait
cmp    bx,3
jl    tmp1
mov    sp,bp
pop    bp
pop    si
pop    dx
pop    bx
ret

func4_fill_rank_table_:
mov    ax,0eH
call   near ptr __STK
push   bx
push   cx
push   dx
push   si
push   bp
mov    bp,sp
sub    sp,2
xor    cx,cx
jmp    tmp3

tmp2:
inc    cx
cmp    cx,3
jge    tmp6

tmp3:
mov    bx,cx
shl    bx,1
mov    word ptr _rank_table[bx],1
xor    dx,dx
jmp    tmp5

tmp4:
inc    dx
cmp    dx,3
jge    tmp2

tmp5:
mov    bx,cx
```

```
shl    bx,1
shl    bx,1
mov    si,dx
shl    si,1
shl    si,1
fld    dword ptr _profit_table[bx]
fcomp   dword ptr _profit_table[si]
fstsw   word ptr -2[bp]
nop
fwait
mov    ax,word ptr -2[bp]
sahf
jae    tmp4
mov    bx,cx
shl    bx,1
inc    word ptr _rank_table[bx]
jmp    tmp4
tmp6:
    mov    sp,bp
    pop    bp
tmp7:
    pop    si
    pop    dx
    pop    cx
    pop    bx
    ret
profit_mark_of_:
    push   ax
    mov    ax,18H
    call   near ptr __STK
    pop    ax
    push   bx
    push   dx
    push   bp
    mov    bp,sp
    sub    sp,10H
    mov    bx,ax
    shl    bx,1
    mov    ax,word ptr _output_price[bx]
    cwd
    idiv   word ptr _input_price[bx]
    dec    ax
    mov    word ptr -4[bp],ax
    fild   word ptr -4[bp]
```

```
fstp    dword ptr -8[bp]
fldz
fcomp   dword ptr -8[bp]
fstsw   word ptr -2[bp]
nop
fwait
mov     ax,word ptr -2[bp]
sahf
jbe    tmp8
mov     al,46H
jmp    tmp12
tmp8:
fld    dword ptr -8[bp]
fst    qword ptr -10H[bp]
fcomp   qword ptr DGROUP:tmp16
fstsw   word ptr -2[bp]
nop
fwait
mov     ax,word ptr -2[bp]
sahf
jbe    tmp9
mov     al,41H
jmp    tmp12
tmp9:
fld    qword ptr -10H[bp]
fcomp   qword ptr DGROUP:tmp17
fstsw   word ptr -2[bp]
nop
fwait
mov     ax,word ptr -2[bp]
sahf
jbe    tmp10
mov     al,42H
jmp    tmp12
tmp10:
fld    qword ptr -10H[bp]
fcomp   qword ptr DGROUP:tmp18
fstsw   word ptr -2[bp]
nop
fwait
mov     ax,word ptr -2[bp]
sahf
jbe    tmp11
mov     al,43H
```

```
jmp    tmp12
tmp11:
    mov    al,44H
tmp12:
    mov    sp,bp
    pop    bp
    pop    dx
    pop    bx
    ret
search_product_:
    push   ax
    mov    ax,0aH
    call   near ptr __STK
    pop    ax
    push   bx
    push   cx
    push   dx
    push   si
    mov    si,ax
    xor    bx,bx
    mov    cl,5
    jmp    tmp14
tmp13:
    inc    bx
    cmp    bx,3
    jge    tmp15
tmp14:
    mov    dx,bx
    shl    dx,cl
    add    dx,offset _products_name
    mov    ax,si
    call   near ptr strcmp_
    test   ax,ax
    jne    tmp13
    mov    ax,bx
    jmp    near ptr tmp7
tmp15:
    mov    ax,0ffffH
    jmp    near ptr tmp7
_TEXT      ENDS
CONST      SEGMENT WORD PUBLIC USE16 'DATA'
tmp16:
    DB 0cdH, 0ccH, 0ccH, 0ccH, 0ccH, 0ccH, 0ecH, 3fH
tmp17:
```

```
DB 0, 0, 0, 0, 0, 0e0H, 3fH
tmp18:
DB 9aH, 99H, 99H, 99H, 99H, 99H, 0c9H, 3fH

CONST      ENDS
CONST2     SEGMENT WORD PUBLIC USE16 'DATA'
CONST2     ENDS
_DATA      SEGMENT WORD PUBLIC USE16 'DATA'
_DATA      ENDS
END
;;;;;;makefile
fake: main impl-asm
    wlink name main.exe file main.obj,impl.obj

main:
wcc main.c -i=..\watcom\h

impl-asm:
wasm impl.asm

clean:
del main.obj main.exe impl.obj main.err

report:
echo ';;;;;;main.c' && cat main.c && echo ';;;;;;afx.h' && cat afx.h && echo ';;;;;;impl.asm' && cat impl.asm
&& echo ';;;;;;makefile' && cat makefile
```

3.2.3 实验记录与分析

如图，在配置好 OpenWatcom 工具链之后，直接运行 make 或 nmake 进行编译链接。

```
DOSBox 0.74, Cpu speed: 10000 cycles, Frameskip 0, Program: WCC  
Copyright (c) Rational Systems, Inc. 1990-1994  
Open Watcom Linker Version 1.9  
Portions Copyright (c) 1985-2002 Sybase, Inc. All Rights Reserved.  
Source code is available under the Sybase Open Watcom Public License.  
See http://www.openwatcom.org/ for details.  
loading object files  
searching libraries  
creating a DOS executable  
  
C:\EXP32>nmake  
Loading NMAKE  
  
Microsoft (R) Program Maintenance Utility Version 1.20  
Copyright (c) Microsoft Corp 1988-92. All rights reserved.  
  
wcc main.c -i=..\\watcom\\h  
DOS/4GW Protected Mode Run-time Version 1.97  
Copyright (c) Rational Systems, Inc. 1990-1994  
Open Watcom C16 Optimizing Compiler Version 1.9  
Portions Copyright (c) 1984-2002 Sybase, Inc. All Rights Reserved.  
Source code is available under the Sybase Open Watcom Public License.  
See http://www.openwatcom.org/ for details.  
afx.h(17): Warning! W138: No newline at end of file  
main.c: 90 lines, included 1746, 1 warnings, 0 errors
```

```
DOSBox 0.74, Cpu speed: 10000 cycles, Frameskip 0, Program: DOSBOX  
Source code is available under the Sybase Open Watcom Public License.  
See http://www.openwatcom.org/ for details.  
main.c(135): Warning! W138: No newline at end of file  
main.c: 135 lines, included 1755, 1 warnings, 0 errors  
Code size: 586  
    wasm impl.asm  
DOS/4GW Protected Mode Run-time Version 1.97  
Copyright (c) Rational Systems, Inc. 1990-1994  
Open Watcom Assembler Version 1.9  
Portions Copyright (c) 1992-2002 Sybase, Inc. All Rights Reserved.  
Source code is available under the Sybase Open Watcom Public License.  
See http://www.openwatcom.org/ for details.  
impl.asm: 224 lines, 0 warnings, 0 errors  
    wlink name main.exe file main.obj,impl.obj  
DOS/4GW Protected Mode Run-time Version 1.97  
Copyright (c) Rational Systems, Inc. 1990-1994  
Open Watcom Linker Version 1.9  
Portions Copyright (c) 1985-2002 Sybase, Inc. All Rights Reserved.  
Source code is available under the Sybase Open Watcom Public License.  
See http://www.openwatcom.org/ for details.  
loading object files  
searching libraries  
creating a DOS executable  
  
C:\EXP32>
```

然后，运行编译好的可执行文件，经过简单测试，功能正常。

```
DOSBox 0.74, Cpu speed: 10000 cycles, Frameskip 0, Program: MAIN
creating a DOS executable

C:\EXP32>main
Your name please: bensong liu
Your password please: test
Passed.

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 1
pen > InputPrice 35 OutputPrice 11.
book > InputPrice 22 OutputPrice 28.
scala > InputPrice 61 OutputPrice 1000.

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? -
dosbox root>>> dosbox
```

```
DOSBox 0.74, Cpu speed: 10000 cycles, Frameskip 0, Program: MAIN
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 4

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 5
pen: profit mark F rank 3
book: profit mark D rank 2
scala: profit mark A rank 1
r1

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ?
dosbox root>>> dosbox
```

```
dosbox, dos
DOSBox 0.74, Cpu speed: 10000 cycles, Frames

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 2
Give '$name $inputPrice $outputPrice' please.
book 111 1
Done.

func1_ls
```

```
DOSBox 0.74, Cpu speed: 10000 cycles, Frame

func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 4

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? 5
pen: profit mark F rank 2
book: profit mark F rank 2
scala: profit mark A rank 1

func1_ls
func2_edit
func3_calc_profit
func4_calc_rank (depend: func3)
func5_print_profit_and_rank (depend: func4)
func6_exit
1-6 ? -
```

```
func1_ls  
func2_edit  
func3_calc_profit  
func4_calc_rank (depend: func3)  
func5_print_profit_and_rank (depend: func4)  
func6_exit  
1-6 ? 6  
C:\EXP32>  
dosbox root >> dosbox
```

4 总结与体会

在本次实验中，我主要掌握了模块化汇编的方法，了解到了在汇编语言中如何协同工作。任务一主要需要两人一组编写两个汇编模块并让两个模块协同工作。在这个实验中，我使用了宏来简化汇编程序，较少常用系统调用对程序员时间的开销。我在合作中实现 345 功能，是 lib 的提供者，但也在静态链接的过程中了解了 public, extrn 等符号的用法，也明白了如何修改段说明使得程序再连接时能够按照所预想的方式进行连接，以实现模块间互相调用，协同工作的结果。这也充分说明了 20 世纪 90 年代 C 语言编译器处理链接的过程中的代码翻译的方法，有利于加深对基础高级编程语言的了解。

在任务 2 中，OpenWatcom 工具链相比 Turbo 系列显然具有优势。例如，wlink 会自动将需要静态链接的 libc 到目标程序，wasm 可以完全兼容 masm 的语法，但只有使用 wasm 生成的 obj 才能被 wlink 进行链接。需要特别注意的是，C 语言是大小写敏感的语言，因此 dos 和 C 的习惯风格会发生冲突从而无法链接，必须注意符号对应的问题，我在此令 asm 服从 C。同时，C 语言倾向于使用 struct 表示 product，但这会引入内存对齐的问题，并给汇编代码的移植带来问题。我在此使用尽最大可能减少汇编程序对全局变量的访问的问题来减少移植过程中的出错概率，并重新实现了所要求的 asm 调用接口。

5 参考文献

- [1] Intel(R) 64 and IA-32 Architectures Software Developer's Manual(<https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf>)
- [2] (out-of-date) INTEL 80386 PROGRAMMER'S REFERENCE MANUAL
1986(<https://css.csail.mit.edu/6.858/2015/readings/i386.pdf>)
- [3] (out-of-date) Open Watcom Toolset (<http://www.openwatcom.org/>)
- [4] (out-of-date) Microsoft ASM Language for MS-DOS
(https://en.wikipedia.org/wiki/Microsoft_Macro_Assembler)
- [5] 80386 instruction set indexed by MIT.edu (<https://pdos.csail.mit.edu/6.828/2017/readings/i386/c17.htm>)
- [6] (out-of-date) MASM directives (<http://stanislavs.org/helppc/directives.html>)
- [7] (out-of-date) DOS Interrupts Reference by SCU.edu.au (<http://spike.scu.edu.au/~barry/interrupts.html>)
- [8] DOSBox: DOS Simulator for modern computer(not a VM) (<https://www.dosbox.com/wiki/>)
- [9] x86 arch introduction by wikibooks.org (https://en.wikibooks.org/wiki/X86_Assembly/X86_Architecture)