

汇编语言程序设计实验报告

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 程序优化

实验时间：2019-4-3, 14:00-17:30 实验地点：南一楼 804 室 30 号实验台

指导教师：曹忠升

专业班级：计算机科学与技术 ACM1701 班

学号：U201714780

姓名：刘晨彦

同组学生：无

报告日期：2019 年 4 月 3 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：2019.04.03

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

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	5
3.1.4	实验步骤	8
3.1.5	实验记录与分析	8
3.2	任务 2	13
3.2.1	设计思想	13
3.2.2	源程序	14
3.2.3	实验步骤	17
3.2.4	实验记录与分析	17
4	总结与体会	19
	参考文献	20

汇编语言程序设计实验报告

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 4 的背景为基础，只要有一个顾客访问网店中的商品，系统就需要计算一遍所有商品的推荐度，然后再处理顾客实际购买的商品的信息。现假设在双十一零点时，SHOP 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 4 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 4 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。

调整后的功能三的描述：

- (1) 提示用户输入要购买的商品名称。【此后可插入计时、循环】
- (2) 计算 SHOP 中所有商品的推荐度。
- (3) 在 SHOP 中找到顾客购买的商品（比如“Bag”，若未能找到该商品，回到（1）重新输入。若只输入回车，则回到功能一（1））。
- (4) 判断该商品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。【循环控制，计时结束】
- (5) 回到功能三（1）。

请按照上述设想修改实验一任务 4 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。另外，也可以把定义“Bag”的位置放在所有商品的最后，使得搜索它的时间变长），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。

3 实验过程

3.1 任务 1

3.1.1 设计思想及存储单元分配

a) 程序设计思想:

在实验一现有代码的基础上进行调整,使代码功能满足本次实验要求。

故功能一、功能二的设计思路与实验一相同。

功能三中首先增加了一个循环以遍历计算所有商品的推荐度。为了方便计算推荐度,将原有的推荐度计算代码封装成子程序以供主程序调用。完成推荐度计算后仍使用原有的双层循环结构比较输入的商品是否存在于商店中。若不存在则回到功能三重新进入,否则判断是否该商品已经售完,若未售完则将已售数量加一,若已售完则回到功能三开始处重新进入。功能三同时还包括了一个 m 次的循环,从用户输入商品名开始,此时将 0 送入 AX 以启动时钟程序,至用户输入的商品卖完(m 次循环结束)为止,并将 1 送入 AX 以结束计时程序。功能三全部结束后回到功能三最开始。计时程序计时单位为毫秒。

功能四由于在本次实验的功能三中未被调用,故删去。

b) 程序的主要算法:

源程序封装:鉴于推荐度计算部分的代码会被反复使用,故为了增强代码可读性,将计算推荐度部分的代码封装为子程序以供调用。

鲁棒性设计:程序在搜索存货量低于 m 件的商品时存在跳至功能三开始处的情况,此时时钟程序未被停止,下次进入再次调用时钟程序可能存在计时不准的情况,故在程序跳转至功能三开始处前,先行停止时钟程序,防止计时结果出错。

c) 存储单元分配:

BNAME: 用户名的首地址。

BPASS: 密码的首地址。

IN_NAME: 存放输入用户名的首地址。

IN_PWD: 存放输入密码的首地址。

IN_ITEM: 存放输入商品名称的首地址。

AUTH: 字节变量,存放登录状态。

SNAME: 字节变量,存放网点名称。

GA1、GA2、GAN: 字节变量,存放商品信息。

BUF1、BUF2、BUF3、BUF4、BUF5、BUF6: 字节变量,存放输出提示语句。

汇编语言程序设计实验报告

CRLF: 字节变量, 存放回车换行。

d) 寄存器分配:

AX: 主要用于算术运算。

CX, BX: 主要用于计数器。

SI: 临时寄存器, 存储计算结果。

DI: m 次循环计数器; 临时寄存器, 存储计算结果。

3.1.2 流程图

源代码总体流程图如图 2.1 所示:

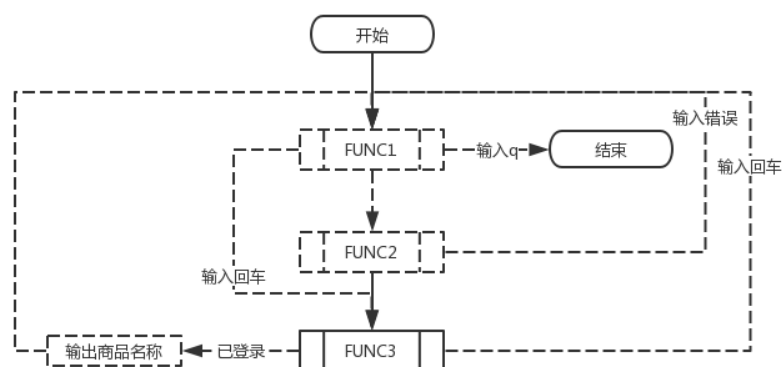


图 2.1 程序总体流程图

程序各功能流程图如图 1.13, 1.14 和 2.2(a), (b)所示, 其中图 1.13 和 1.14 为实验一中的功能一二的流程图, 本次实验保留了实验一中的功能一和功能二。

汇编语言程序设计实验报告

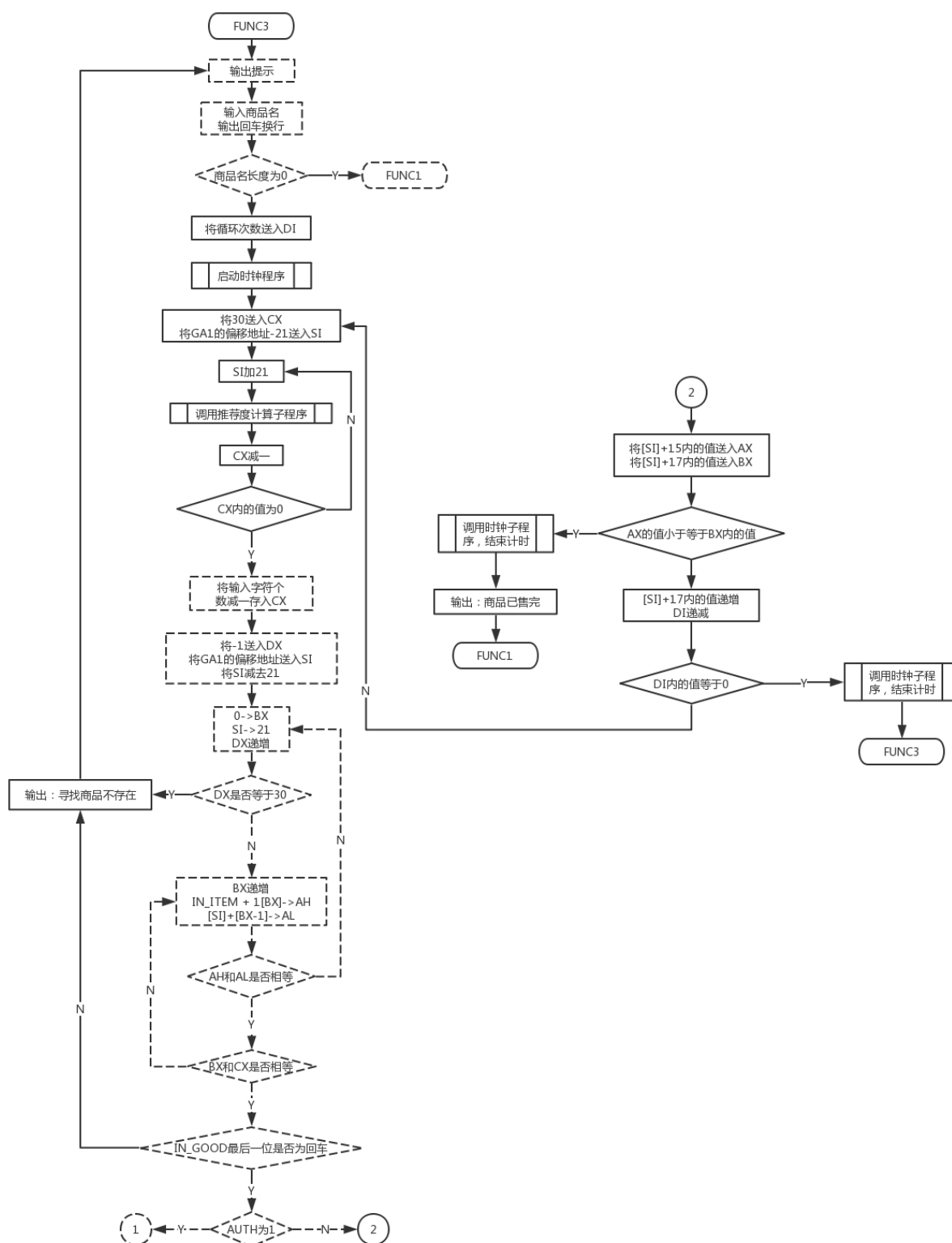


图 2.2(a) 功能三程序实现流程图 1

汇编语言程序设计实验报告

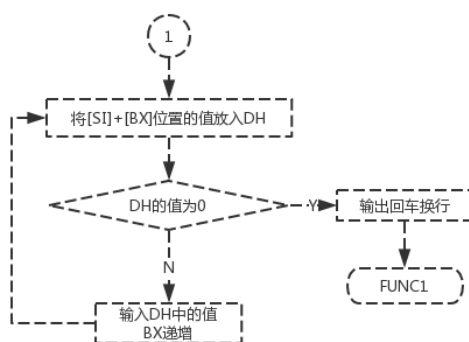


图 2.2(b) 功能三程序实现流程图 2

3.1.3 源程序

```
. 386
STACK SEGMENT USE16 STACK
    DB 200 DUP(0)
STACK ENDS
:
BUF4 DB 'PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:', OAH, ODH, '$'
BUF5 DB 'SORRY THE ITEM YOU WANT ISNT AVAILABLE', OAH, ODH, '$'
BUF6 DB 'THE ITEM YOU WOULD LIKE TO PURCHASE HAVE BEEN SOLD OUT!', OAH, ODH, '$'
CRLF DB ODH, OAH, '$'
:
FUNC3:
:
    JE FUNC1
    MOV DI, 5000          ;SET THE NUMBER OF M
    MOV AX, 0
    CALL TIMER
HERE1:  MOV CX, WORD PTR 30      ;SET HOW MANY TIMES IT LOOPS TO COUNT THE
RECOMMONDATION POINT
    MOV SI, OFFSET GA1      ;此八行为新增代码
    SUB SI, 21
COUNT_ALL_RECOM:
    ADD SI, 21
    CALL RECOM
    DEC CX
    CMP CX, 0
    JNE COUNT_ALL_RECOM
    MOV CL, IN_ITEM + 1     ;START TO COMPARE AND FIND THE ITEM
    :
    CMP AUTH, 1             ; IF THE OWNER LOGGED IN, PRINT THE NAME OF THE ITEM
    JE OPITEM               ;此一行为新增代码
```


汇编语言程序设计实验报告

```
    CMP AUTH, 0                ;IF IN THE CUSTOMER MODE,
CANT_FIND:                    ; 此五行为新增代码
    LEA DX, BUF5              ; print: CANT FIND THE ITEM
    MOV AH, 9
    INT 21H
    JMP FUNC3
:
    JMP FUNC1
SUB_ITEM:                    ;此二十二行为新增代码
    MOV AX, [SI] + 15
    MOV BX, [SI] + 17
    CMP AX, BX
    JNA BFUNC1                ;IF AX <= BX, FUNCTION GOES BACK TO FUNCTION1
    INC WORD PTR [SI] + 17    ;
    DEC DI
    CMP DI, 0
    JNE HERE1
    JMP BFUNC3
BFUNC3: MOV AX, 1
    CALL TIMER
    JMP FUNC3
BFUNC1: MOV AX, 1
    CALL TIMER
    JMP FUNC1
RECOM PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI
    MOV AL, [SI] + 10         ;DISCOUNT IN AX
:
    MOV [SI] + 19, BX
    POP DI                   ;此七行为新增代码
    POP DX
    POP CX
    POP BX
    POP AX
    RET
RECOM ENDP

TIMERPROC                    ;此子程序为新增代码
    PUSH DX
```

汇编语言程序设计实验报告

```
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
```

汇编语言程序设计实验报告

```
INT    21H
POP    DS
JMP    _T0
_TS    DW    ?
        DB    'Time elapsed in ms is '
_TMSGDB    12 DUP(0)
TIMER    ENDP

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

3.1.4 实验步骤

1. 准备上机实验环境。
2. 使用 VISUAL STUDIO 修改实验一中的程序，要求满足本次实验要求，保存至 SHOP.ASM。使用 MASM6.0 汇编源文件，观察提示信息，若出错则返回重新编辑 SHOP.ASM，保存后重新汇编，直至不再报错为止。
3. 使用连接程序 LINK.EXE 将生成的 SHOP.OBJ 文件连接成执行文件。
4. 执行程序。按照程序设计要求进行交互，检查是否达到程序设计要求。
5. 使用 TD.EXE 观察 SHOP.EXE 的执行情况。即 TD SHOP.EXE 回车：
 - (1) 观察调用推荐度计算子程序时，堆栈内的数值改变情况、SP 的变化情况和挂起的寄存器内的数值变化。
 - (2) 观察推荐度计算完成后存入对应内存位置的操作
 - (3) 改变搜索的商品在内存中存储的先后顺序，观察对程序运行时间的影响
 - (4) 改变商品的名称长度，观察对程序运行时间的影响。
 - (5) 若程序循环体中有信息显示的代码，观察对程序执行的影响。

3.1.5 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G 内存；WINDOWS 10 下 DOSBox0.72；TD.EXE 5.0。
2. 汇编源程序时未发生异常
3. 连接过程中未发生异常
4. 程序功能测试：
 - (1) 功能一、二测试：
程序功能一、二源代码与实验一完全相同，故测试见报告一。
 - (2) 功能三测试：

汇编语言程序设计实验报告

输出提示用户输入测试，测试结果如图 2.3 所示，测试结果显示功能正常。

```
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

图 2.3 输出提示用户输入测试截图

计算 SHOP 中所有商品推荐度测试，测试结果如图 2.4(a), (b), (c) 所示，测试结果显示功能正常。

```
ds:0050 50 42 4F 4F 4B 00 00 00 PBOOK
ds:0058 00 00 00 09 0C 00 1E 00 00 00 00 00 00 00 00
ds:0060 19 00 05 00 AA 00 54 65 00 00 00 00 00 00 00
ds:0068 6D 70 2D 56 61 6C 75 65 mp-Value
ds:0070 08 0F 00 14 00 1E 00 02 00 00 00 00 00 00 00
```

图 2.4(a) SHOP 中商品推荐度计算存储测试截图一

```
ds:0078 00 6A 00 54 65 6D 70 2D j Temp-
ds:0080 56 61 6C 75 65 08 0F 00 00 00 00 00 00 00 00
ds:0088 14 00 1E 00 02 00 6A 00 00 00 00 00 00 00 00
ds:0090 54 65 6D 70 2D 56 61 6C Temp-Val
ds:0098 75 65 08 0F 00 14 00 1E ue 00 00 00 00 00 00
```

图 2.4(b) SHOP 中商品推荐度计算存储测试截图二

```
ds:00A0 00 02 00 6A 00 54 65 6D 00 00 00 00 00 00 00
ds:00A8 70 2D 56 61 6C 75 65 08 p-Value
ds:00B0 0F 00 14 00 1E 00 02 00 00 00 00 00 00 00 00
ds:00B8 6A 00 54 65 6D 70 2D 56 j Temp-U
ds:00C0 61 6C 75 65 08 0F 00 14 alue 00 00 00 00 00 00
```

图 2.4(c) SHOP 中商品推荐度计算存储测试截图三

未找到商品测试，测试结果如图 2.5 所示，测试结果显示功能正常。

```
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
PEN
SORRY THE ITEM YOU WANT ISNT AVAILABLE
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

图 2.5 未能找到商品测试截图

只输入回车测试，测试结果如图 2.6 所示，测试结果显示功能正常。

```
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:

WELCOME! YOU ARE VISITING ONLINE SHOP
PLEASE ENTER YOUR NAME AND PASSWORD:
```

图 2.6 只输入回车测试截图

未售商品数大于等于循环数量测试，测试结果如图 2.7 所示，测试结果显示功能正常。

```
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
BAG
Time elapsed in ms is 2750
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

图 2.7 未售商品数大于等于循环数量测试截图

汇编语言程序设计实验报告

未售商品数小于循环数量测试，测试结果如图 2.8 所示，测试结果显示功能正常。

```
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
BOOK
Time elapsed in ms is 0
WELCOME! YOU ARE VISITING ONLINE SHOP
PLEASE ENTER YOUR NAME AND PASSWORD:
```

图 2.8 未售商品数小于循环数量测试截图

功能三结束回到功能三开始测试，测试截图如图 2.7 所示，测试结果显示功能正常。

5. 使用 TD.EXE 观察 SHOP.EXE 的执行情况：

(1) 在调用推荐度计算子程序后，将数据区 GOTO 至堆栈的存储段，观察数据被压入栈的过程，如图 2.9、图 2.10 所示：

[.] = CPU 80486			1
cs:00D3 83C615	add	si,0015	ax 1162
cs:00D6 E89900	call	0172	bx 0000
cs:00D9 49	dec	cx	cx 001A
cs:00DA 83F900	cmp	cx,0000	dx 03C0
cs:00DD 75F4	jne	00D3	si 00A5
cs:00DF 8A0E3000	mov	cl,[00301	di 1388
cs:00E3 80E901	sub	cl,01	bp 0000
cs:00E6 B500	mov	ch,00	sp 00C8
cs:00E8 BAF0FF	mov	dx,FFFF	ds 0AD7
cs:00EB BE5100	mov	si,0051	es 0ABA
cs:00EE 83EE15	sub	si,0015	ss 0ACA
cs:00F1 BBFFFF	mov	bx,FFFF	cs 0B14
cs:00F4 83C615	add	si,0015	ip 00D6

图 2.9 进入子程序前测试截图

[.] = CPU 80486			1	[.]
cs:0172 50	push	ax	ax 1162	c=0
cs:0173 53	push	bx	bx 0000	z=0
cs:0174 51	push	cx	cx 001A	s=0
cs:0175 52	push	dx	dx 03C0	o=0
cs:0176 57	push	di	si 00A5	p=1
cs:0177 8A440A	mov	al,[si+0A]	di 1388	a=0
cs:017A B400	mov	ah,00	bp 0000	i=1
cs:017C 66B4C0D	mov	ecx,[si+0D]	sp 00C0	d=0
cs:0180 66F7E1	mul	ecx	ds 0AD7	
cs:0183 66B90A000000	mov	ecx,0000000A	es 0ABA	
cs:0189 66BA00000000	mov	edx,00000000	ss 0ACA	
cs:018F 66F7F1	div	ecx	cs 0B14	
cs:0192 66BBD8	mov	ebx,eax	ip 0175	
cs:0195 66B440B	mov	eax,[si+0B]		
cs:0199 66B980000000	mov	ecx,00000080		
ss:0018 00 00 00 00 00 00 00 00			ss:00C8 0000	
ss:0020 00 00 00 00 00 00 00 00			ss:00C6 00D9	
ss:0028 00 00 00 00 00 00 00 00			ss:00C4 1162	
ss:0030 00 00 00 00 00 00 00 00			ss:00C2 0000	
ss:0038 00 00 00 00 00 00 00 00			ss:00C0 001A	

图 2.10 进入子程序后测试截图

根据测试可知，子程序结束后应该返回的地址在进入子程序后首先被压入栈中，然后根据 POP 指令，依次将不同寄存器内的值压入栈中。同时可观察到，栈底地址最大，堆栈向偏移地址小的方向堆放数据。

SP 的变化情况如图 2.11、图 2.12 所示，测试显示栈顶指针 SP 每次压栈完成递减 2 的

汇编语言程序设计实验报告

操作。

```
[ ]-CPU 80486-
cs:0172 50      push  ax      ax 1162
cs:0173 53      push  bx      bx 0000
cs:0174 51      push  cx      cx 0017
cs:0175 52      push  dx      dx 03C0
cs:0176 57      push  di      si 00E4
cs:0177 8A440A   mov    al,[si+0A]  di 1388
cs:017A B400     mov    ah,00      bp 0000
cs:017C 668B4C0D mov    ecx,[si+0D]  sp 00C6
```

图 2.11 SP 变化截图一

```
[ ]-CPU 80486-
cs:0172 50      push  ax      ax 1162
cs:0173 53      push  bx      bx 0000
cs:0174 51      push  cx      cx 0017
cs:0175 52      push  dx      dx 03C0
cs:0176 57      push  di      si 00E4
cs:0177 8A440A   mov    al,[si+0A]  di 1388
cs:017A B400     mov    ah,00      bp 0000
cs:017C 668B4C0D mov    ecx,[si+0D]  sp 00C4
```

图 2.12 SP 变化截图二

挂起的寄存器在子程序中的使用情况如图 2.13、图 2.14、图 2.15 所示，测试显示寄存器挂起后子程序中的数据被送入 AX 中，AX 正常使用，子程序结束前保存在堆栈中的数据(1162)_H 被送回 AX 中，回到主程序继续运行。

```
[ ]-
ax 1162  c=0
bx 0000  z=0
cx 0016  s=0
dx 03C0  o=0
si 00F9  p=1
di 1388  a=0
bp 0000  i=1
sp 00C4  d=0
ds 0AD7
es 0ABA
ss 0ACA
cs 0B14
ip 0173

ss:00CC 0000
ss:00CA 0000
ss:00C8 0000
ss:00C6 00D9
ss:00C4 1162
```

图 2.13 AX 被压入栈中的测试截图

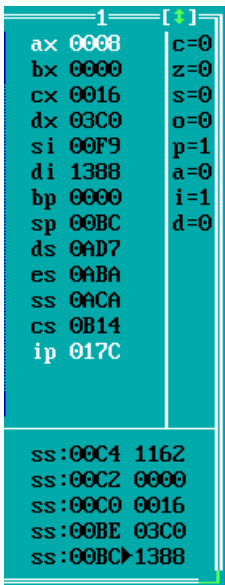


图 2.14 AX 被子程序使用的测试截图

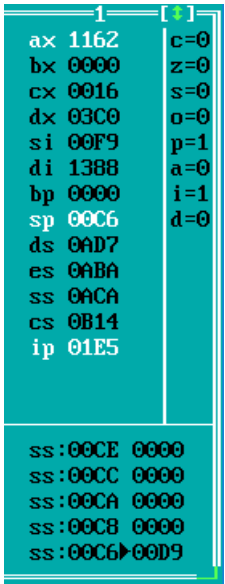


图 2.15 AX 在主程序中的数据被还原的测试截图

- (2) 观察推荐度计算完成后存入对应内存位置的操作：如图 2.4(a), (b), (c)所示，推荐度正确存入了对应的内存位置。
- (3) 将商品 BAG 的存储位置改为最靠前的位置（源程序 BAG 位于商品最后），测试结果如图 2.16 所示：

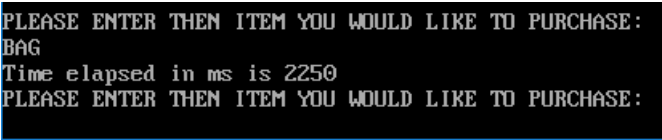


图 2.16 BAG 位置靠前时执行时间测试

与图 2.7 相比，程序运行时间由明显缩短，表明商品名称的比较次数和程序运行时间存在正相关。

与图 2.7 的测试结果相比, 程序运行时间有延长, 表明商品名称长度与程序运行时长存在负相关关系。

```

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
Time elapsed in ms is 4610
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

与图 2.7 的测试结果相比，程序运行时间有明显增加，表明显示程序对运行时间的影响明显。

3.2.1 设计思想

具体包括：

-
- 13

汇编语言程序设计实验报告

- 3.将数据移入低位，高位补零的操作用 MOVZX 代替；
- 4.将存储推荐度的内存位置加入推荐度的运算，减少运算指令。

本次实验中 $m = 5000$, $n = 30$ 。

3.2.2 源程序

```

    :
FUNC3:  LEA DX, BUF4
    :
HERE1:  MOV CX, WORD PTR 30          ;SET HOW MANY TIMES IT LOOPS TO COUNT THE
RECOMMENDATION POINT

    MOV SI, OFFSET GA1 - 21
;将“GA1 偏移地址送入 SI, SI 减 21”简化为一行语句
COUNT_ALL_RECOM:
    ADD SI, 21
    CALL RECOM
    DEC CX
    JNZ COUNT_ALL_RECOM
;将“DX 递减，比较 DX 的值与 0，若不等回到 COUNT_ALL_RECOM”改为两句
RECOMMENDATION
    MOVZX CX, IN_ITEM + 1
;将“把 IN_ITEM + 1 送入 CL，把 0 送入 CH”改为一句
    SUB CX, 1
    MOV DX, -1                      ;DX counts the term of item
    MOV SI, OFFSET GA1 - 21
;将“GA1 偏移地址送入 SI, SI 减 21”简化为一行语句
LOP3: MOV BX, -1                    ;BX counts the term of literal
    ADD SI, 21
    INC DX
    CMP DX, 30
    JE  CANT_FIND
LOP4: INC BX
    MOV AH, IN_ITEM + 2[BX]
    MOV AL, [SI]+[BX]
    CMP AH, AL
    JNE LOP3
    CMP BX, CX
    JNE LOP4
    CMP IN_ITEM + 2[BX] + 1, 0DH
;将“BX 递减，将 0DH 与 IN_ITEM + 2[BX]内所存内容比较”改为一行语句
    JNE CANT_FIND                  ;WHEN REACHES THIS LINE OF CODE, IT FIND THE ITEM

    CMP AUTH, 0                    ;IF IN THE CUSTOMER MODE
```

汇编语言程序设计实验报告

```
JE SUB_ITEM ;NEWLY ADDED
```

```
MOV BX, 0
```

```
CMP AUTH, 1
```

```
JE OPITEM
```

;调换了以上五行的顺序，将不经常跳转的“CMP AUTH, 1”等语句移到经常跳转的语句之后

```
CANT_FIND:
```

```
LEA DX, BUF5 ;print: CANT FIND THE ITEM
```

```
MOV AH, 9
```

```
INT 21H
```

```
JMP FUNC3
```

```
OPITEM: MOV DH, [SI]+[BX] ;PRINT ITEM'S NAME IN A LOOP
```

```
CMP DH, 0
```

```
JE OPITEM1
```

```
MOV DL, DH
```

```
MOV AH, 2
```

```
INT 21H
```

```
INC BX
```

```
JMP OPITEM
```

```
OPITEM1:LEA DX, CRLF ;回车换行
```

```
MOV AH, 9 ;回车换行
```

```
INT 21H
```

```
JMP FUNC1
```

```
SUB_ITEM: ;NEWLY ADDED
```

```
MOV AX, [SI] + 15 ;NEWLY ADDED
```

```
MOV BX, [SI] + 17 ;NEWLY ADDED
```

```
CMP AX, BX ;NEWLY ADDED
```

```
JNA BFUNC1 ;IF AX <= BX, FUNCTION GOES BACK TO FUNCTION1
```

```
INC WORD PTR [SI] + 17 ;NEWLY ADDED
```

```
DEC DI ;NEWLY ADDED
```

```
JNZ HERE1
```

; 将“DI 递减，比较 DI 的值与 0，若不等回到 HERE1”改为以上两句

```
JMP BFUNC3 ;NEWLY ADDED
```

```
BFUNC3: MOV AX, 1 ;NEWLY ADDED, IMTER ENDS
```

```
CALL TIMER ;NEWLY ADDED
```

```
JMP FUNC3 ;NEWLY ADDED
```

```
BFUNC1: MOV AX, 1 ;NEWLY ADDED, TIMER ENDS
```

```
CALL TIMER ;NEWLY ADDED
```

```
LEA DX, BUF6 ;print: ITEM HAS BEEN SOLD
```

```
MOV AH, 9
```

```
INT 21H
```

汇编语言程序设计实验报告

```
JMP FUNC1                ;NEWLY ADDED
RECOM PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI
    MOVZX AX, BYTE PTR [SI] + 10        ;DISCOUNT IN AX
;把 “[SI] + 10 的值送入 AL, 0 送入 AH” 简化为一条语句
    MOV CX, [SI] + 13                ;SALE PRICE IN CX
    MUL CX                          ;SALE * DISCOUNT IN AX
    MOV CX, 10
    XOR DX, DX
    DIV CX                          ;ACTUAL SALE PRICE IN AX
    MOV BX, AX                      ;ACTUAL SALE PRICE IN BX
    MOV AX, [SI] + 11                ;PURCHASE PRICE
    SHL AX, 8                        ;PURCHASE PRICE * 128 IN AX
;以上将乘法的两句改为一句位移
    MOV CX, BX                      ;ACTUAL SALE PRICE IN CX
    XOR DX, DX
; 以上将 “送 0 入 DX” 改为效率更高的异或
    DIV CX                          ;PURCHASE PRICE * 128 / ACTUAL SALE PRICE IN AX
    MOV [SI] + 19, AX                ;PURCHASE PRICE * 128 / ACTUAL SALE PRICE IN BX
    MOV AX, [SI] + 15                ;NUM OF PURCHASE IN AX
    MOV CX, 2                        ;2 IN CX
    MUL CX                          ;2 * NUM OF PURCHASE IN AX
    MOV DI, AX                      ;2 * NUM OF PURCHASE IN DI
    MOV AX, [SI] + 17                ;NUM OF SALE
    SHL AX, 8                        ;NUM OF SALE * 128 IN AX
;以上将乘法的两句改为一句位移
    MOV CX, DI                      ;2 * NUM OF PURCHASE IN CX
    XOR DX, DX
; 以上将 “送 0 入 DX” 改为效率更高的异或
    DIV CX                          ;NUM OF SALE * 128 / 2 * NUM OF PURCHASE IN AX
    ADD [SI] + 19, AX
;将推荐度直接送入内存，删去了加到 BX 在由 BX 送进内存的操作
    POP DI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
RECOM ENDP
```

汇编语言程序设计实验报告

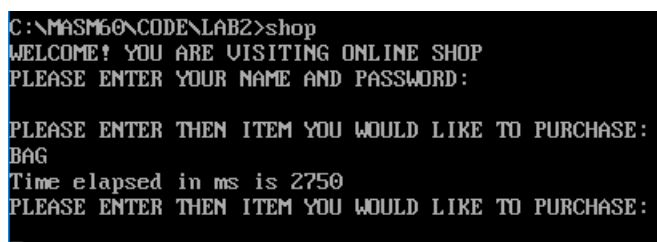
```
    :  
CODE    ENDS  
END START
```

3.2.3 实验步骤

1. 准备上机实验环境。
2. 使用 VISUAL STUDIO 编辑程序，实现要求功能，保存至 SHOPM.ASM。使用 MASM6.0 汇编源文件，观察提示信息，若出错则返回重新编辑 SHOPM.ASM，保存后重新汇编，直至不再报错为止。
3. 使用连接程序 LINK.EXE 将生成的 SHOPM.OBJ 文件连接成执行文件。
4. 根据优化思想对源代码进行优化。
5. 计算优化比。

3.2.4 实验记录与分析

1. 实验环境条件:i7-7700HQ 2.80GHz, 8G 内存; WINDOWS 10 下 DOSBox0.72; TD.EXE 5.0。
2. 汇编源程序时未发生异常
3. 连接过程中未发生异常
4. 优化记录:
 - (1) 本次测试选取 m 为 5000 次循环时为基础，并且进行优化。未优化前的代码在执行 M 值为 5000 时所耗费时间为 2750ms，如图 2.19 所示：



```
C:\MASM60\CODE\LAB2>shop  
WELCOME! YOU ARE VISITING ONLINE SHOP  
PLEASE ENTER YOUR NAME AND PASSWORD:  
  
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:  
BAG  
Time elapsed in ms is 2750  
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

图 2.19 优化前程序运行时间

- (2) 使用执行效率更高但功能相同的语句替换，如图 2.20(a)、(b)所示：

```
MOV EAX, [SI] + 17      ;NUM OF SALE  
MOV ECX, 128  
MUL ECX                ;NUM OF SALE * 128 IN AX
```

图 2.20(a) 优化前

```
MOVZX EAX, WORD PTR [SI] + 17      ;NUM OF SALE  
SHL EAX, 8                        ;NUM OF SALE * 128 IN AX
```

图 2.20(b) 优化后

- (3) 将两行语句替换为一行功能相同的语句，如图 2.21(a)、(b)所示：

汇编语言程序设计实验报告

```
MOV AL, [SI] + 10      ;DISCOUNT IN AX
MOV AH, 0
```

2.21(a) 优化前

```
MOVZX EAX, BYTE PTR [SI] + 10      ;DISCOUNT IN AX
```

2.21(b) 优化后

(4) 将循环中经常使用的判断和跳转调整至不经常使用的跳转与判断之前，同时不变程序功能，如图 2.22(a)、(b)

```
JNE CANT_FIND
MOV BX, 0      ;WHEN REACHES THIS LINE OF CODE, IT FIND THE ITEM

CMP AUTH, 1      ;IF THE OWNER LOGGED IN, PRINT THE NAME OF THE ITEM
JE OPITEM

CMP AUTH, 0      ;IF IN THE CUSTOMER MODE,
JE SUB_ITEM      ;NEWLY ADDED
```

图 2.22(a) 优化前

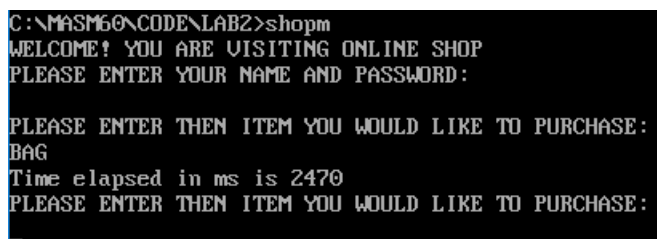
```
JNE CANT_FIND      ;WHEN REACHES THIS LINE OF CODE, IT FIND THE ITEM

CMP AUTH, 0      ;IF IN THE CUSTOMER MODE
JE SUB_ITEM      ;NEWLY ADDED

MOV BX, 0
CMP AUTH, 1      ;IF THE OWNER LOGGED IN, PRINT THE NAME OF THE ITEM
JE OPITEM
```

图 2.22(b) 优化后

5. 优化前后运行时间如图 2.23 所示：



```
C:\MASM60\CODE\LAB2>shopm
WELCOME! YOU ARE VISITING ONLINE SHOP
PLEASE ENTER YOUR NAME AND PASSWORD:

PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
BAG
Time elapsed in ms is 2470
PLEASE ENTER THEN ITEM YOU WOULD LIKE TO PURCHASE:
```

图 2.23 优化后

最终优化比 = $\frac{2750-2470}{2750} = 10.18\%$ ，达成优化要求。

4 总结与体会

在本次实验中，第一次尝试将代码封装成子程序，第一次调用子程序，第一次使用外部代码完成设计。在使用子程序时，意识到子程序拥有：可读性强，便于管理修改等功能。通过调用外部的时钟代码，提高了阅读并使用他人代码能力。

在任务一测试中，通过改变商品的存储先后顺序，商品名称长度和添加输出代码，了解到程序执行时间和运行代码行数的正相关关系，其中输出内容对运行时间的影响最大。

通过对堆栈的观察，我也加深了对堆栈使用的理解，其中包括对子程序进入时保护现场和子程序结束时的恢复现场，栈顶指针 SP 如何移动。对堆栈的理解能够帮助我更好的组织代码。

在任务二的实验中，为了达到优化的要求，我了解了许多指令，如 MOVZX、SHL 等指令，了解到许多功能相同但是执行效率更高的指令。

通过本次实验，加深了对汇编语言的了解与应用，感觉到自己在不断的进步。

汇 编 语 言 程 序 设 计 实 验 报 告

参考文献

- [1] 许向阳. 80X86 汇编语言程序设计上机指南. 武汉: 华中科技大学出版社, 2007: 1-61
- [2] 王元珍, 曹忠升, 韩宗芬. 80X86 汇编语言程序. 武汉: 华中科技大学出版社, 2007