**汇编语言程序设计**

作业三

2005010130 侯杰

第六章习题：

**6.4** 分析下面程序的功能，并写出堆栈最满时各单元的地址及内容（程序略）

答：此程序的功能是以十六进制格式输出AX的内容。输出结果为”4321”。

此题采用递归方式实现，当栈内容最满时堆栈内容如下：

|  |
| --- |
|  |
| 050eeh  050f0h  050f2h  050f6h  050fah  050fch |
| (IP) |
| 0003h |
| (IP) |
| 0002h  050f4h |
| (IP) |
| 0001h  050f8h |
| (IP) |
| 0  050feh |
| DS  05100h |
|  |

**6.7** 设有10个学生的成绩分别是76，69，84，90，73，88，99，63，100和80分。试编写一个子程序统计60~69分, 70~79分, 80~89分, 90~99分和100分的人数并分别保存到S6,S7,S8,S9和S10单元中。

答:程序如下（另附程序源文件6\_7.asm）

data segment

score db 76,69,84,90,73,88,99,63,100,80

count db 10

S6 db 0

S7 db 0

S8 db 0

S9 db 0

S10 db 0

data ends

code segment

main proc far

assume cs:code,ds:data

start:

;initialize

mov ax,data

mov ds,ax

;call subroutine

call calc

;exit

mov ax,4c00h

int 21h

main endp

calc proc near

mov cx,10 ;count of the number of persons

mov si,0

startCalc:

mov ax,0

mov al,score[si]

mov dl,10

div dl

cmp al,10

je AddToS10 ;100

cmp al,9

je AddToS9 ;90-99

cmp al,8

je AddToS8 ;80-89

cmp al,7

je AddToS7 ;70-79

cmp al,6

je AddToS6 ;60-69

jmp next ;else do nothing

AddToS10:

add S10,1

jmp next

AddToS9:

add S9,1

jmp next

AddToS8:

add S8,1

jmp next

AddToS7:

add S7,1

jmp next

AddToS6:

add S6,1

jmp next

next:

inc si

dec cx

jz exit

jmp startCalc

exit:

ret

calc endp

code ends

end start

**6.8** 编写一个有主程序和子程序结构的程序模块。子程序的参数是一个N字节数组的首地址TABLE，数N及字符CHAR。要求在N字节数组中查找字符CHAR，并记录该字符的出现次数。主程序则要求从键盘接收一串字符以建立字节数组TABLE，并逐个显示从键盘输入的每个字符CHAR以及它在TABLE数组中出现的次数。

答:程序如下（另附程序源文件6\_8.asm）

data segment

TABLE db 100 dup (?) ;maximal characters that can be received:100

N dw ? ;length of string

Info1 db 'Please input a string:','$'

Info2 db 'Please input the char you want to search for:','$'

Info3 db ' times','$'

data ends

code segment

main proc far

assume cs:code,ds:data,es:data

start:

;initialize

mov ax,data

mov ds,ax

mov es,ax

;main procedure

;input a string

mov si,0

lea dx,Info1

mov ah,9

int 21h

input:

mov ah,1

int 21h

cmp al,0dh ;compare to return

je startSearch ;stop input if receiving a return

mov TABLE[si],al

inc si

mov N,si ;update character count

jmp input

startSearch:

mov si,0

next:

call RETURN

lea dx,Info2

mov ah,9

int 21h

mov bx,0

mov ah,1 ;get the char to be searched

int 21h

and ax,00ffh

push ax ;pass the parameter by stack

push N

mov ax,offset TABLE

push ax

call search

call RETURN

pop ax

mov dl,al

mov ah,2

int 21h

mov dl,3ah

mov ah,2

int 21h

mov dl,bl

add dl,30h

mov ah,2 ;output the times of appearence

int 21h

lea dx,Info3

mov ah,9

int 21h

call RETURN

inc si

jmp next

;exit

mov ax,4c00h

int 21h

main endp

search proc near

mov bp,sp

add bp,2

mov di,[bp]

push cx

push ax

push dx

mov dx,0

mov cx,[bp+2]

mov al,[bp+4]

again: ;search

cmp al,[di]

jne continue

inc dx

continue:

inc di

dec cx

jz exit

jmp again

exit:

mov bx,dx ;restore the stack

pop dx

pop ax

pop cx

ret 4

search endp

RETURN proc near

mov dl,0ah

mov ah,2

int 21h

mov dl,0dh

mov ah,2

int 21h

ret

RETURN endp

code ends

end start

**6.10** 编写子程序嵌套结构的程序，把整数分别用二进制和八进制形式显示出来。

主程序BANDO:把整数字变量VAL1存入堆栈，并调用子程序PAIRS

子程序PAIRS:从堆栈中取出VAL1，调用二进制显示程序OUTBIN显示出与其等效的二进制数，输出8个空格。调用八进制显示程序OUTOCT显示出与其等效的八进制数，调用输出回车及换行符的子程序。

答:程序如下（另附程序源文件6\_10.asm）

data segment

VAL1 dw -9527

spaces db ' ','$'

data ends

code segment

BANDO proc far

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

push VAL1 ;push the operating number to stack

call PAIRS

;exit

mov ax,4c00h

int 21h

BANDO endp

PAIRS proc near

mov bp,sp

mov bx,[bp+2] ;load the operating number to bx

call OUTBIN

lea dx,spaces ;output 8 spaces

mov ah,9

int 21h

mov bx,[bp+2] ;load the operating number to bx

call OUTOCT

call RETURN

ret

PAIRS endp

OUTBIN proc near

mov cx,16 ;count

next:

rol bx,1

mov ax,bx

and ax,0001 ;mask

mov dl,al

add dl,30h

mov ah,2

int 21h

dec cx

jnz next

ret

OUTBIN endp

OUTOCT proc near

rol bx,1

mov ax,bx

and ax,0001 ;mask

mov dl,al

add dl,30h

mov ah,2

int 21h ;finish outputing the first number

mov ch,5 ;count

mov cl,3

next2:

rol bx,cl

mov ax,bx

and ax,0007h ;mask

mov dl,al

add dl,30h

mov ah,2

int 21h

dec ch

jnz next2

ret

OUTOCT endp

RETURN proc near

mov dl,0ah

mov ah,2

int 21h

mov dl,0dh

mov ah,2

int 21h

ret

RETURN endp

code ends

end start

第七章习题：

**7.1** 编写一条宏指令CLRB，完成用空格符将一字符区中的字符取代的工作。字符区首地址及其长度为变元。

答：

CLRB macro FIRST,LENGTH

local next

mov cx,LENGTH

mov si,0

next:

mov FIRST[si],20h

inc si

loop next

endm

**7.2** 给定宏定义如下

DIF MACRO X,Y

MOV AX,X

SUB AX,Y

ENDM

ABSDIF MACRO V1,V2,V3

LOCAL CONT

PUSH AX

DIF V1,V2

CMP AX,0

JGE CONT

NEG AX

CONT: MOV V3,AX

POP AX

ENDM

试展开以下调用，并判定调用是否有效

(1)ABSDIF P1,P2,DISTANCE

展开结果：

PUSH AX

MOV AX,P1

SUB AX,P2

CMP AX,0

JGE CONT

NEG AX

CONT: MOV DISTANCE,AX

POP AX

调用有效

(2)ABSDIF [BX],[SI],X[DI],CX

展开结果：

PUSH AX

MOV AX,[BX]

SUB AX,[SI]

CMP AX,0

JGE CONT

NEG AX

CONT: MOV X[DI],AX

POP AX

调用有效

(3)ABSDIF [BX][SI],X[BX][SI],240H

调用无效

(4)ABSDIF AX,AX,AX

展开结果：

PUSH AX

MOV AX,AX

SUB AX,AX

CMP AX,0

JGE CONT

NEG AX

CONT: MOV AX,AX

POP AX

调用有效

**7.5** 宏指令BIN\_SUB完成多个字节数据连减的功能

RESULT<-(A-B-C-D-…)

要相减的字节数据顺序存放在首地址为OPERAND的数据区中，减数的个数存放在COUNT单元中，最后结果存入RESULT单元。请编写此宏指令

答：

BIN\_SUB macro first,n,rslt

local next

push ax

push si

push cx

mov si,2

mov cx,n

dec cx

mov ax,first

next:

sub ax,first[si]

add si,2

loop next

mov rslt,ax

pop cx

pop si

pop ax

endm

**7.7** 下面宏指令CNT和INC1完成相继字存储：

CNT MACOR A,B

A&B DW ?

ENDM

INC1 MACRO A,B

CNT A,% B

B=B+1

ENDM

请展开下列宏调用：

C=0

INC1 DATA,C

INC1 DATA,C

答：

DATA0 DW ?

DATA1 DW ?

**7.9** 宏指令STORE定义如下

STORE MACRO X,N

MOV X+I,I

I=I+1

IF I-N

STORE X,N

ENDIF

ENDM

试展开下列调用：

I=0 STORE TAB,7

答：

MOV TAB+0,0

MOV TAB+1,1

MOV TAB+2,2

MOV TAB+3,3

MOV TAB+4,4

MOV TAB+5,5

MOV TAB+6,6

建立一个宏（数学）库，扩展已有的指令系统。

要求：

宏库包含**n的开方**、**n的平方**、**n的绝对值**、**以2为底n的对数**的宏定义，运算结果仅取整数部分，不考虑溢出（字长16bit），但要考虑n的正负；

编写完整程序，代码段中要有相应的宏调用，以检验宏定义的正确性。

答： 宏库内容如下（另附宏库文件MathLib.mac以及测试文件LibTest.asm）

;Square

square macro src,dst

push ax

mov ax,src

imul ax

mov dst,ax

pop ax

endm

;Absolute

absolute macro src,dst

local cont

push ax

mov ax,src

add ax,0;get SF

jns cont

neg ax

cont:

mov dst,ax

pop ax

endm

;Square root

squareroot macro src,dst

local exit,wrong,cont,found

push ax

push bx

push cx

push dx

mov bx,src

add bx,0;get SF

js wrong

mov cx,1;start search from 1

cont:

mov ax,cx

mul ax

cmp ax,bx

ja found

inc cx;not found, continue experiment

jmp cont

found:

dec cx

mov dst,cx

jmp exit

wrong:

mov dst,-1;For negative src, return -1

jmp exit

exit:

pop dx

pop cx

pop bx

pop ax

endm

;Logarithm with base 2

logarithm macro src,dst

local exit,wrong,cont,found

push ax

push bx

push cx

push dx

mov ax,src

add ax,0;get SF

js wrong

mov cx,1

mov bx,2

cont:

div bx

cmp ax,1

je found

inc cx;//not found, continue experiment

mov dx,0

jmp cont

found:

mov dst,cx

jmp exit

wrong:

mov dst,-1

jmp exit

exit:

pop dx

pop cx

pop bx

pop ax

endm

第八章习题：

**8.1** 写出分配给下列中断类型号在中断向量表中的物理地址

(1)INT 12H

物理地址：0000:0048H~0000:004BH

(2)INT 8

物理地址：0000:0020H~0000:0023H

**8.2** 用CALL指令来模拟实现INT21H显示字符T的功能

答：代码如下（另附8\_2.asm）

code segment

main proc far

assume cs:code

start:

mov ah,2

mov dl,54h

mov bx,0

mov ds,bx

mov bx,84h

call dword ptr [bx]

;exit

mov ax,4c00h

int 21h

main endp

code ends

end start

**8.6** 试编写程序，它轮流测试两个设备的状态寄存器，只要一个状态寄存器的第0位为1，则与其相应的设备就输入一个字符。如果其中任一状态寄存器的第3位为1，则整个输入过程结束。两个状态寄存器的端口地址分别是0024和0036，与其相应的数据输入寄存其的端口则为0026和0038,输入字符分别存入首地址为BUFF1和BUFF2的存储区中。

答：代码如下（另附8\_6.asm）

data segment

BUFF1 db 100 dup (?)

BUFF2 db 100 dup (?)

data ends

code segment

main proc far

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

mov es,ax

mov si,0

mov di,0

next1:

;测试设备 1

in al,0024h

test al,08h

jne exit;第三位为1，整个过程结束

test al,01h

je next2;第0位为0，不输入，试下一个设备

in al,0026h;否则输入

mov BUFF1[si],al

inc si

next2:

;测试设备 2

in al,0036h

test al,08h

jne exit;第三位为1，整个过程结束

test al,01h

je next1;第0位为0，不输入，试下一个设备

in al,0038h;否则输入

mov BUFF2[di],al

inc di

jmp next1

exit:

mov ax,4c00h

int 21h

main endp

code ends

end start

**8.14** 试编制以程序，要求测出任意程序的运行时间，并把结果打印出来

答：代码如下(另附8\_14.asm)

;---------------------------------------------------------

;---------------------------------------------------------

data segment

time dw 0

Dec\_0\_Outputable db 0 ;(used when Dec output) 0:'0' unoutputable, 1:'0' outputable

const1 dw 1000

const2 dw 18

data ends

;---------------------------------------------------------

;---------------------------------------------------------

code segment

;---------------------------------------------------------

main proc far

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

;main program

;backup old interrupt program

mov al,1ch

mov ah,35h

int 21h

push es

push bx

push ds

;write new interrupt program

mov dx,offset timing

mov ax,seg timing

mov ds,ax

mov al,1ch

mov ah,25h

int 21h

pop ds

;set interrupt mask bits

in al,21h

and al,11111110b

out 21h,al

mov bx,0;bx is used to count interruption times

;FROM HERE YOU CAN PLACE THE PROGRAM THAT YOU WANT TO

;COUNT THE RUNNING TIME

mov di,50000

delay1:

mov si,55000

delay2:

dec si

jnz delay2

dec di

jnz delay1

;PROGRAM ENDS HERE

;restore interrupt program

pop dx

pop ds

mov al,1ch

mov ah,25h

int 21h

;restore ds

mov ax,data

mov ds,ax

;convert to millisecond

mov ax,bx

mov cx,const1

mul cx

mov dx,0

mov cx,const2

div cx

mov bx,ax

;display running time

call BIN\_DEC

;output "ms"

mov dl,6dh

mov ah,2

int 21h

mov dl,73h

mov ah,2

int 21h

;exit

mov ax,4c00h

int 21h

main endp

;---------------------------------------------------------

timing proc near

inc bx

iret

timing endp

;---------------------------------------------------------

BIN\_DEC proc near

;the number to be changed must be saved in bx

add bx,0 ;in order to get sign flag

jns next

outputSign:

mov dl,2dh ;if sf=1, then first output a '-'

mov ah,2

int 21h

neg bx

next:

mov cx,10000d

call DEC\_DIV

mov cx,1000d

call DEC\_DIV

mov cx,100d

call DEC\_DIV

mov cx,10d

call DEC\_DIV

mov dl,bl

add dl,30h

mov ah,2

int 21h

ret

BIN\_DEC endp

;---------------------------------------------------------

DEC\_DIV proc near

;Screen output, and the number to be divided must be saved in bx

mov ax,bx

mov dx,0

div cx ;quotient in ax, residue in dx

mov bx,dx ;residue is saved in bx, quontient is saved in ax(in fact, al)

cmp al,0

jnz output ;for non-zero number, output it directly

cmp Dec\_0\_Outputable,0

je exit2

output:

mov dl,al

add dl,30h

mov ah,2

int 21h

mov Dec\_0\_Outputable,1

exit2:

ret

DEC\_DIV endp

;---------------------------------------------------------

code ends

;---------------------------------------------------------

;---------------------------------------------------------

end start