title addTwoNumber

include irvine32.inc ; Assembly lib

.data ; Data Segment

t0 db 13h ; 13h -> t0, db means 8b

t1 db 56h ; 56h -> t1

sum dw 0 ; 0 -> sum, dw means 16b

.code ; Code Segment

main proc ; Main Procedure

xor ax,ax ; Make AX(16b Operator) and CF(Carry Flag) equal to 0

mov al,t0 ; Make al = t0, al(ah) is the lower(higher) 8b of AX

add sum,ax ; Add the value of ax(t0) to sum

xor ax,ax ; Make AX(16b Operator) and CF(Carry Flag) equal to 0

mov al,t1 ; Make al = t1

add sum,ax ; Add the value of ax(t1) to sum

exit ; Exit the operations

main endp ;

end main ; End of Main Procedure

title countNumber

include irvine32.inc

.data

t0 db 12,13,10,-5,-85,37,92,-25,94,10,-36

count EQU ($-t0)

p\_number db 0 ; p\_number stores the total number of positive number

n\_number db 0 ; n\_number stores the total number of negative number

.code

main proc

mov cx,count

mov si,0

again: cmp t0[si],0

jg p\_number\_add ; t0[si] > 0

jl n\_number\_add ; t0[si] < 0

jz next ; t0[si] = 0

loop again

p\_number\_add: inc p\_number

jmp next

n\_number\_add: inc n\_number

jmp next

next: inc si

loop again

exit

main endp

end main

title maxMinNumber

include irvine32.inc ; Assembly lib

.data ; Data Segment

t0 db 12,13,10,37,92,94,10 ; 12 -> t0, 13 -> t0+1, 10 -> t0+2 ...

count EQU ($-t0) ; $ means Program Counter, its original value is 0, current $=t0+3

; EQU means "equal to", make count equal to value ($-t0)=3

max db -128 ; -128 -> max, db means 8b

min db 127 ; 127 -> min, db means 8b

.code ; Code Segment

main proc ; Main Procedure

mov cx,count ; Make CX(16b Operator) equal to the value of count

; SI stores the index of data array

mov si,0 ; Make SI(16b Operator) equal to 0

; AX stores the MAX number, and BX stores the MIN number

xor ax,ax ; Make AX(16b Operator) and CF(Carry Flag) equal to 0

xor bx,bx ; Make BX(16b Operator) and CF(Carry Flag) equal to 0

mov al,max ; Make al = max, al(ah) is the lower(higher) 8b of AX

mov bl,min ; Make bl = min, bl(bh) is the lower(higher) 8b of BX

; Main loop of this procedure

; At the beginning, the procedure carries out cx = cx-1,

; then it judges if the cx > 0, if not ends up this loop

again: cmp t0[si],al ; Loop again point, Compare the value stored in (t0+si) and al

jg max\_number ; if t0[si] > al (jump great transition): Move to max\_number

jl next ; if t0[si] < al (jump less transition): Move to next

; if t0[si] > al

max\_number: mov al,t0[si] ; max\_number: 1.Make al = t0[si];

mov max,al ; 2.Make max = al;

jmp next ; 3.Move to next

next: cmp t0[si],bl ; next: 1.Compare t0[si] and bl

jl min\_number ; 2.if t0[si] < bl, move to min\_number

jg final ; 3.if t0[si] > bl, move to final

; if t0[si] < bl

min\_number: mov bl,t0[si] ; min\_number: 1.Make bl = t0[si];

mov min,bl ; 2.Make min = bl;

jmp final ; 3.Move to final

final: inc si ; final: 1.add 1 to si

loop again ; 2.loop again, move to loop again point

exit ; exit the loop

main endp ; End of Procedure

end main

title bubbleSortingASC

include irvine32.inc

.data

tab db 12,9,4,5,7,3,1,8 ; $ = tab+8

count EQU ($-tab) ; count = 8

dab db count dup(?) ; "db count dup(?)" decalres 8 uninitialized bytes in memory

.code

main proc

mov cx,count ; cx = count = 8

mov si,0 ; si = 0

; print the original array

put1:

movsx eax,tab[si] ; eax = tab[si](a signed number, so leverage movsx)

call writeint ; write eax to the console

mov al,32 ; TBD: why make al(lower 8b of eax) equal to 32?

call writechar ; write al to the console

inc si ; si += 1

loop put1 ; loop back, cx -= 1

mov al,10

call writechar

; give values to dab array

mov cx,count

mov si,0

s:

mov bl,tab[si]

mov dab[si],bl

inc si

loop s

; bubble sorting, ascending

mov cx,count ; cx = count

dec cx ; cx -= 1

lop1:

mov di,cx ; di = cx

mov si,0 ; si = 0

lop2:

mov al,dab[si] ; al = array[i]

cmp al,dab[si+1] ; compare array[i] and array[i+1]

jb go\_on ; if array[i] < array[i+1]: move to go\_on

; else if array[i] > array[i+1]:

xchg al,dab[si+1] ; exchange al(val=old\_array[i]) and array[i+1]

mov dab[si],al ; array[i] = al(val=old\_array[i+1])

go\_on:

add si,1 ; si += 1

loop lop2 ; move to lop2

mov cx,di ; cx = di

loop lop1 ; move to lop1

; print to CRT

mov cx,count

mov si,0

put2:

movsx eax,dab[si]

call writeint

mov al,32

call writechar

inc si

loop put2

exit

main endp

end main