; 实验二 分支,循环程序设计

;一.实验目的:

; 1.开始独立进行汇编语言程序设计;

; 2.掌握基本分支,循环程序设计;

; 3.掌握最简单的 DOS 功能调用.

;

;二.实验内容:

; 1.安排一个数据区（数据段）,内存有若干个正数,负数和零.每类数的个数都不超过 9.

; 2.编写一个程序统计数据区中正数,负数和零的个数.

; 3.将统计结果在屏幕上显示.

;

;三.预习题:

; 1.十进制数 0 -- 9 所对应的 ASCII 码是什么? 如何将十进制数 0 -- 9 在

; 屏幕上显示出来?

; 2.如何检验一个数为正,为负或为零? 你能举出多少种不同的方法?

;

;四.选作题:

; 统计出正奇数,正偶数,负奇数,负偶数以及零的个数.

.model small ;define memory model

.stack 100h ;define stack segment

.data ;define data segment

doscall equ 21h ;DOS interrupt number

display equ 2h

;numbers in [-128,127]

array db -128,127,-4,12,34,0,5,1,4,6,9,-8,-7,0,0,3 ; define an array to store the numbers we want to identify.

n\_num equ $-array ; get the array length

positive\_num db 00h ;number of positive numbers.

negative\_num db 00h ;number of negative numbers.

positive\_odd\_num db 00h ;number of positive odd numbers.

positive\_even\_num db 00h ;number of positive even numbers.

zero\_num db 00h ;number of zeros.

negative\_odd\_num db 00h ;number of negative odd numbers.

negative\_even\_num db 00h ;number of negative even numbers.

;string table

str0 db 'The array is: $'

positive\_string db 'num of positive: $' ; a strings to print

negative\_string db 'num of negative: $'

positive\_odd\_string db 'num of positive odd: $'

positive\_even\_string db 'num of positve even: $'

zero\_string db 'num of zeros: $'

negative\_odd\_string db 'num of negative odd: $'

negative\_even\_string db 'num of negative even: $'

.code ;define code segment

main proc far

push ds ;save old data segment

sub ax,ax ;put zero in AX

push ax ;zero on stack

mov ax,@data ;data segment address

mov ds,ax ;into DS register

;main part of program goes here

push cx

mov cx,n\_num

mov si,0h

again:

mov al,array[si]

cmp al,0h

jz zero

cmp al,0h

js negative ;jump to negative.

inc positive\_num

shr al,1

jnc positve\_even ;jump to positive\_even.

inc positive\_odd\_num

jmp continue\_loop

positve\_even:

inc positive\_even\_num

jmp continue\_loop

negative:

inc negative\_num

shr al,1

jnc negative\_even;

inc negative\_odd\_num

jmp continue\_loop

negative\_even:

inc negative\_even\_num

jmp continue\_loop

zero:

inc zero\_num;

continue\_loop:

inc si

loop again

call print\_array

call print\_results ; print results to screen.

exit:

mov ax,4c00h ;go back to DOS

int doscall

main endp ;end of main program

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

;input: DL,the ascii of the char.

;output: a char on screen

print\_char proc near

mov ah,02h ;display function

int doscall ;call DOS

ret ;return from print\_char

print\_char endp

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

;打印换行

endline proc near

mov dl,0dh ;carriage return

mov ah,02h ;display function

int doscall ;call DOS

mov dl,0ah ;linefeed

mov ah,02h ;display function

int doscall ;call DOS

ret ;return from endline

endline endp

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

;DX=string address,the string ends in $

;output: screen.

print\_string proc near

mov ah,09h

int doscall

ret

print\_string endp

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

;input: al = number

;output: a number char on screen.

print\_num proc near

add al,30h ;the ASCII code for the character '0' is 30h

mov dl,al ;DL=the output character

mov ah,02h

int doscall

ret

print\_num endp

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

;input: AL=the data

;output:print a byte in decimal on screen

printdb proc near

;Subroutine to convert hex to binary

cmp al,0H; al<0(positive?)

jge pos

push ax

mov dl,'-'-0h

call print\_char

pop ax

xor ax,0ffh

inc ax

pos:

push bx

push cx

mov bx,0

mov ah,0

add bx,ax

cmp bx,10d

jl b1;bx<10

cmp bx,100d

jl b10;bx<10

mov cx,100d

call dec\_div

b10:

mov cx,10d

call dec\_div

b1:

mov cx,1d

call dec\_div

pop cx

pop bx

ret

;Sub routine to divide number in BX by number in CX

;pri nt quotient on screen

;(nu merator in DX+AX,denominator in CX)

dec\_div proc near

mov ax,bx

mov dx,0

div cx

mov bx,dx

;Sub routine to print the number

mov dl,al

add dl,30h

mov ah,display

int doscall

ret

dec\_div endp

printdb endp

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

; print the array to screen.

print\_array proc near

lea dx,str0

call print\_string

push cx

push si

mov cx,n\_num

mov si,0h

print\_next:

mov al,array[si]

call printdb

mov dl,','-0h

call print\_char

inc si

loop print\_next

pop si

pop cx

call endline

ret

print\_array endp

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

; print results to screen.

print\_results proc near

lea dx,positive\_string ;print positive odd number

call print\_string

mov al,positive\_num

call print\_num

call endline

lea dx,negative\_string ;print positive odd number

call print\_string

mov al,negative\_num

call print\_num

call endline

lea dx,positive\_odd\_string ;print positive odd number

call print\_string

mov al,positive\_odd\_num

call print\_num

call endline

lea dx,positive\_even\_string ;print positive even number

call print\_string

mov al,positive\_even\_num

call print\_num

call endline

lea dx,zero\_string ;print zeros

call print\_string

mov al,zero\_num

call print\_num

call endline

lea dx,negative\_odd\_string ;print negative odd number

call print\_string

mov al,negative\_odd\_num

call print\_num

call endline

lea dx,negative\_even\_string ;print negative even number

call print\_string

mov al,negative\_even\_num

call print\_num

ret

print\_results endp

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

end main ;end of assembly