; Define macro with two parameters

; Implements the write system call

%macro write\_string 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

section .data ; Data section

userMsg db 'Please enter a number: ', 0xa ; Ask the user to enter a number

lenUserMsg equ $-userMsg ; The length of the message

dispMsg db 'You have entered: ', 0xa

lenDispMsg equ $-dispMsg

evenMsg db 'You have entered an even number ', 0xa

lenevenMsg equ $-evenMsg

oddMsg db 'You have entered an odd number ', 0xa

lenoddMsg equ $-oddMsg

ncMsg db ' ', 0xa

lenncMsg equ $-ncMsg

sumMsg db 'The sum is', 0xa

lenSumMsg equ $-sumMsg

section .bss ;Uninitialized data

num resb 1

num2 resb 1

result resb 1

section .text ;Code Segment

global \_start

\_start: ;User prompt

;mov eax, 4

;mov ebx, 1

;mov ecx, userMsg

;mov edx, lenUserMsg

;int 80h

; call macro instead

write\_string userMsg, lenUserMsg

; write newline using macro

write\_string ncMsg, lenncMsg

; Read and store the user input

mov eax, 3

mov ebx, 0

mov ecx, num

mov edx, 1

int 80h

; Output the message 'The entered number is: '

;mov eax, 4

;mov ebx, 1

;mov ecx, dispMsg

;mov edx, lenDispMsg

;int 80h

; call macro instead

write\_string dispMsg, lenDispMsg

; Output the number entered

;mov eax, 4

;mov ebx, 1

;mov ecx, num

;mov edx, 1

;int 80h

; call macro instead

write\_string num, 1

; newline carriage return

; mov eax, 4

; mov ebx, 1

; mov ecx, ncMsg

; mov edx, lenncMsg

; int 80h

; call macro instead

write\_string ncMsg, lenncMsg

; Determine if even or odd

mov al, [num]

sub al, '0'

and al, 1

jz EVEN\_NUMBER

; Have odd number so print that it is odd

; mov eax, 4

; mov ebx, 1

; mov ecx, oddMsg

; mov edx, lenoddMsg

; int 0x80

; call macro instead

write\_string oddMsg, lenoddMsg

jmp Continue

EVEN\_NUMBER:

; mov eax, 4

; mov ebx, 1

; mov ecx, evenMsg

; mov edx, lenevenMsg

; int 0x80

; call macro instead

write\_string evenMsg, lenevenMsg

Continue:

; newline carriage return

;mov eax, 4

; mov ebx, 1

; mov ecx, ncMsg

; mov edx, lenncMsg

; int 80h

; call macro instead

write\_string ncMsg, lenncMsg

; Read and store the another user input

mov eax, 3

mov ebx, 0

mov ecx, num2

mov edx, 1

int 80h

; Output the message 'The entered number is: '

; mov eax, 4

; mov ebx, 1

; mov ecx, dispMsg

; mov edx, lenDispMsg

; int 80h

; call macro instead

write\_string dispMsg, lenDispMsg

; Output the number entered

; mov eax, 4

; mov ebx, 1

; mov ecx, num2

; mov edx, 1

; int 80h

; call macro instead

write\_string num2, 1

; newline carriage return

; mov eax, 4

; mov ebx, 1

; mov ecx, ncMsg

; mov edx, lenncMsg

; int 80h

; call macro instead

write\_string ncMsg, lenncMsg

; Determine if even or odd

mov al, [num2]

sub al, '0'

and al, 1

jz EVEN\_NUMBER2

; Have odd number so print that it is odd

; mov eax, 4

; mov ebx, 1

; mov ecx, oddMsg

; mov edx, lenoddMsg

; int 0x80

; call macro instead

write\_string oddMsg, lenoddMsg

jmp exit

EVEN\_NUMBER2:

; mov eax, 4

; mov ebx, 1

; mov ecx, evenMsg

; mov edx, lenevenMsg

; int 0x80

; call macro instead

write\_string evenMsg, lenevenMsg

; Exit code

exit:

; newline carriage return

; mov eax, 4

; mov ebx, 1

; mov ecx, ncMsg

; mov edx, lenncMsg

; int 80h

; call macro instead

write\_string ncMsg, lenncMsg

; sum the numbers

; first get binary numbers and then call sum

mov ecx, [num]

sub ecx, '0'

mov edx, [num2]

sub edx, '0'

call sum

; print sum

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mov eax, 1

int 80h

; declare sum procedure

; assuming ecx and edx have the numbers to sum

sum:

mov eax, ecx

add eax, edx

add eax, '0'

ret