

Practica 2

Instituto Politecnico Nacional Escuela Superior de Computo

Sistemas operativos

Interfaz de llamadas al sistema

Ethan Jezreel Lopez Torres
Gonzaga Martínez José Alberto
Sebastian Absalon Cortes

Marco Teorico

Programa 1

```
segment .data ;Segmento de datos
cadena db 'Programando en ensamblador para Linux',0xA ;Cadena a imprimir
segment .text ;Segmento de código
global _start;Punto de entrada al programa (usado en el enlazador ld)
_start: ;Inicio del programa
    mov     edx,38d ;Longitud de cadena
    mov     ecx,cadena ;Cadena a escribir
    mov     ebx,1 ;Salida estandar
```

```

mov    eax,4 ;Numero de llamada al sistema "sys_write"
int    0x80 ;Interrupción de llamadas al sistema del kernel de Linux
mov    eax,1 ;Número de llamada al sistema "sys_exit"
int    0x80 ;Interrupción de llamadas al sistema del kernel de Linux

```

Este programa nos permite ver de forma detallada el proceso que sigue una computadora para realizar todas las acciones que le permiten imprimir una línea en la consola.

- Primeramente vemos como declaramos un segmento inicial.
- A continuación vemos como se declara una variable de tipo cadena
- A continuación vemos como se inicia otro segmento de código, en este caso para el texto
- A continuación se inicia el bloque de código
- A continuación se realiza con la acción mov se especifica la longitud de la cadena, posteriormente se establece la cadena que se imprimirá y finalmente se establece la salida estándar
- Podemos ver a continuación la llamada al sistema para escribir en consola
- Finalmente se hace una interrupción al sistema

Es importante señalar que el código está segmentado en bloques de datos y de código, esto se hace debido a que el ensamblador debe realizar procesos separados para declarar los datos que usará, como el código que manipulará estos datos.

Programa 2

```

segment .bss ;Segmento de datos
cadena resb 50 ;Espacio en memoria para la cadena a almacenar
segment .text ;Segmento de código
global _start ;punto de entrada al programa (usado con el enlazador ld)
_start: ;Inicio del programa
mov     edx,50d ;Longitud del bufer
mov     ecx,cadena ;Cadena a leer
mov     ebx,0 ;Entrada estándar
mov     eax,3 ;Numero de llamada al sistema "sys_read"
int     0x80 ;Interrupción de llamadas al sistema del kernel de Linux
mov     edx,50d ;Longitud de cadena
mov     ecx,cadena ;Cadena a escribir
mov     ebx,1 ;Salida estándar
mov     eax,4 ;Numero de llamada al sistema "sys_write"
int     0x80 ;Interrupción de llamadas al sistema del kernel de Linux
mov     eax,1 ;Número de llamada al sistema "sys_exit"
int     0x80 ;Interrupción de llamadas al sistema del kernel de Linux

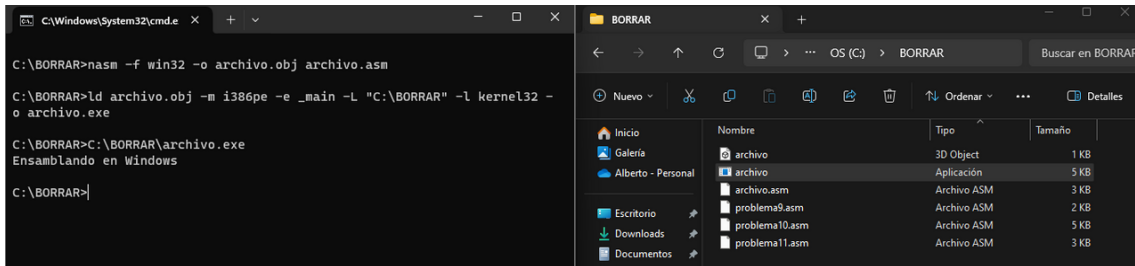
```

Este programa nos muestra paso a paso el procedimiento para poder leer una cadena de texto ingresada por el usuario y luego imprimir la misma cadena en la terminal. Por parte del programa anterior sabemos como es que se imprime una línea de código con el ensamblador, sin embargo en este programa podemos ver a detalle el proceso para poder obtener una cadena de caracteres por parte del usuario.

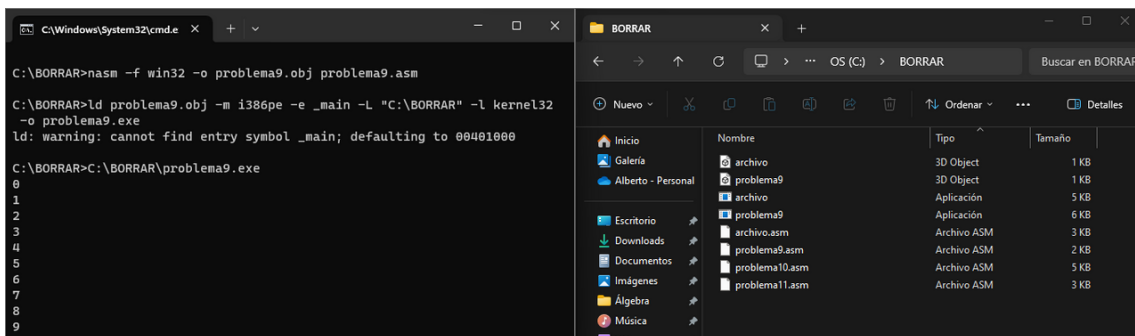
- El programa sigue la misma secuencia de ejecución que el programa anterior, hasta el momento en el que establece una longitud de buffer de 50 y entonces establece el punto de lectura de la entrada estándar
- A continuación este

Sección Windows

Punto número 3 de la práctica. Se crea el archivo obj Se crea el ejecutable



Problema 9



```
segment .data
    cont db '0'; se reserva un byte de enters
    enters db 0xA
segment .bss
    handleConsola resd 1
    longitudCadena resd 1
    caractEscritos resd 1
    ultimoArgumento resd 1
segment .text
global _start
extern _GetStdHandle@4;
extern _WriteConsoleA@20;
extern _ExitProcess@4;
_start:

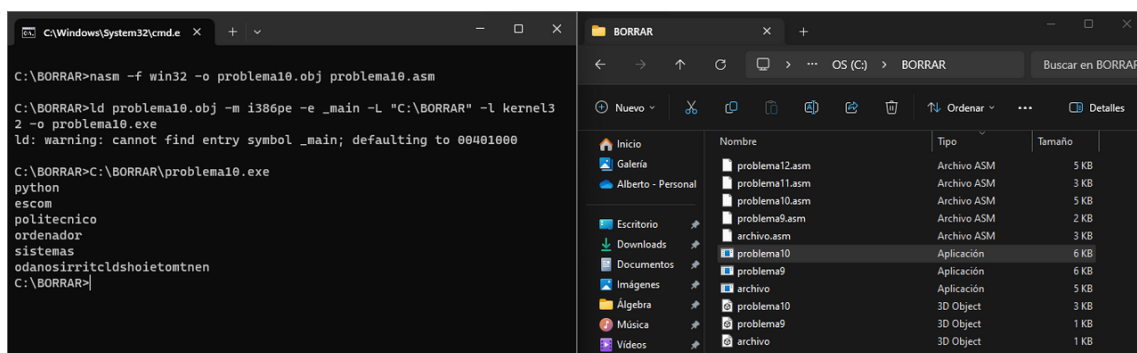
    jmp loop1
```

```

loopi:
    cmp byte [cont], ':'
    jne incr
    je fin
fin:
    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;
incr:
    ;imprime incremento
    push dword -11
    call _GetStdHandle@4;
    mov [handleConsola],eax;
    xor eax,eax;
    mov eax,2d;
    mov [longitudCadena],eax;
    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    push dword caractEscritos;
    push dword [longitudCadena];
    push dword cont;
    push dword [handleConsola];
    call _WriteConsoleA@20;
    ;imprime enter
    inc byte [cont]
    jmp _start

```

Problema 10



```

segment .data
    l1 dd 1
    l2 dd 1
    l3 dd 1

```

```

l4 dd 1
l5 dd 1
l6 dd 1
salto db 0xa
section .bss
handleConsola resd 1
caractEscritos resd 1
longitudCadena resd 1
ultimoArgumento resd 1
s1 resb 50
s2 resb 50
s3 resb 50
s4 resb 50
s5 resb 50
d resb 150

segment .text
global _start
extern _GetStdHandle@4
extern _ReadConsoleA@20
extern _WriteConsoleA@20
extern _ExitProcess@4

_start: push dword -10
        call _GetStdHandle@4
        mov [handleConsola],eax
        xor eax,eax
        mov eax,50d
        mov[longitudCadena],eax
        xor eax,eax
        mov eax,0d
        mov [ultimoArgumento],eax
        push dword [ultimoArgumento]
        push dword caractEscritos
        push dword [longitudCadena]
        push dword s1
        push dword [handleConsola]
        call _ReadConsoleA@20

        push dword -10
        call _GetStdHandle@4
        mov [handleConsola],eax
        xor eax,eax
        mov eax,50d
        mov[longitudCadena],eax
        xor eax,eax
        mov eax,0d
        mov [ultimoArgumento],eax
        push dword [ultimoArgumento]
        push dword caractEscritos
        push dword [longitudCadena]

```

```
push dword s2
push dword [handleConsola]
call _ReadConsoleA@20
```

```
push dword -10
call _GetStdHandle@4
mov [handleConsola],eax
xor eax,eax
mov eax,50d
mov[longitudCadena],eax
xor eax,eax
mov eax,0d
mov [ultimoArgumento],eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword [longitudCadena]
push dword s3
push dword [handleConsola]
call _ReadConsoleA@20
```

```
push dword -10
call _GetStdHandle@4
mov [handleConsola],eax
xor eax,eax
mov eax,50d
mov[longitudCadena],eax
xor eax,eax
mov eax,0d
mov [ultimoArgumento],eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword [longitudCadena]
push dword s4
push dword [handleConsola]
call _ReadConsoleA@20
```

```
push dword -10
call _GetStdHandle@4
mov [handleConsola],eax
xor eax,eax
mov eax,50d
mov[longitudCadena],eax
xor eax,eax
mov eax,0d
mov [ultimoArgumento],eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword [longitudCadena]
push dword s5
push dword [handleConsola]
call _ReadConsoleA@20
```

```
mov edi,s1
call _strlen
mov [l1],eax
mov [l6],eax
mov edi,s2
call _strlen
mov [l2],eax
add [l6],eax
mov edi,s3
call _strlen
mov [l3],eax
add [l6],eax
    mov edi,s4
call _strlen
mov [l4],eax
add [l6],eax
    mov edi,s5
call _strlen
mov [l5],eax
add [l6],eax
mov eax,0
xor edi,edi
mov edi,d
jmp cop1
```

modu:

```
add eax,1
cmp eax,[l6]
jle cop1
jmp final
```

cop1:

```
cmp eax,[l1]
jge cop2
```

```
mov esi,s1
add esi,eax
mov ecx, 1
rep movsb
```

cop2:

```
cmp eax,[l2]
jge cop3
```

```
mov esi,s2
add esi,eax
mov ecx, 1
rep movsb
```

cop3:

```
cmp eax,[l3]
jge modu
```

```
mov esi,s3
add esi,eax
mov ecx, 1
rep movsb
```

```
cop4:
  cmp eax,[l4]
  jge modu
```

```
mov esi,s4
add esi,eax
mov ecx, 1
rep movsb
```

```
cop5:
  cmp eax,[l5]
  jge modu
```

```
mov esi,s5
add esi,eax
mov ecx, 1
rep movsb
```

```
jmp modu
```

```
final:
push dword-11
  call _GetStdHandle@4;
  mov [handleConsola],eax;
  xor eax,eax;
  mov eax,0d;
  mov [ultimoArgumento],eax;
  push dword [ultimoArgumento];
  push dword caractEscritos;
  push dword [l6];
  push dword d;
  push dword [handleConsola];
  call _WriteConsoleA@20;
```

```
push dword-11
  call _GetStdHandle@4;
  mov [handleConsola],eax;
  xor eax,eax;
  mov eax,0d;
  mov [ultimoArgumento],eax;
  push dword [ultimoArgumento];
  push dword caractEscritos;
  push dword [l6];
  push dword d;
```



```

        push dword [handleConsola];
        call _WriteConsoleA@20;

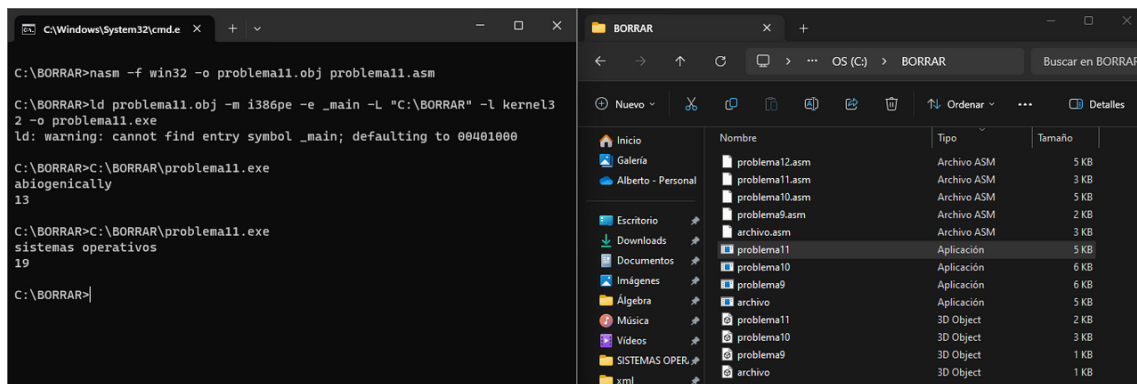
xor     eax,eax;
mov     eax,0d;
mov     [ultimoArgumento],eax;
push    dword [ultimoArgumento];
call    _ExitProcess@4;

;-----

_strlen:
    mov     ebx, edi        ; rbx = rdi
    xor     al, al          ; limpiar al
    mov     ecx, 0xffffffff ; maximo numero de bytes
    repne scasd             ; while [rdi] != al, sigue escaneando
    sub     edi, ebx        ; longitud = dist2 - dist1
    sub     edi, 2
    mov     eax, edi        ; eax guarda la longitud
    ret                   ; volver

```

Problema 11



```
segment .data
```

```
section .bss
```

```

    caract resd 1
    handleConsola resd 1
    caractEscritos resd 1
    longitudCadena resd 1
    ultimoArgumento resd 1
    s1 resd 100
    espDigito resd 100
    posDigito resd 8

```

```
segment .text
```

```

global _start
extern _GetStdHandle@4
extern _ReadConsoleA@20
extern _WriteConsoleA@20
extern _ExitProcess@4

_start:
push dword -10
    call _GetStdHandle@4
    mov [handleConsola],eax
    xor eax,eax
    mov eax,100d
    mov[longitudCadena],eax
    xor eax,eax
    mov eax,0d
    mov [ultimoArgumento],eax
    push dword [ultimoArgumento]
    push dword caractEscritos
    push dword [longitudCadena]
    push dword s1
    push dword [handleConsola]
    call _ReadConsoleA@20

    mov edi,s1

    call len
    ;mov eax,126
    sub eax,2

    call imprimir
;_ExitProcess
    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;

imprimir:
    mov ecx, espDigito
    mov ebx, 10
    mov [ecx], ebx
    inc ecx
    mov [posDigito], ecx

bucle:
    mov edx, 0
    mov ebx, 10
    div ebx

```

```

;push eax
add edx, 48

mov ecx, [posDigito]
mov [ecx], dl
inc ecx
mov [posDigito], ecx

;pop eax
cmp eax, 0
jne bucle

bucle2:
    mov ecx, [posDigito]
;_WriteConsoleA
;    mov eax, 1
;    mov edi, 1
;    mov rsi, ecx
;    mov edx, 1

    ;mov dword [caract],
    push dword-11
    xor eax,eax;
    call _GetStdHandle@4;
    mov [handleConsola],eax;
    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    push dword caractEscritos;
    push dword 1;
    push dword ecx
    push dword [handleConsola];
    call _WriteConsoleA@20;

    mov ecx, [posDigito]
    dec ecx
    mov [posDigito], ecx

    cmp ecx, espDigito
    jge bucle2
    ret

len:
    xor ecx, ecx ;limpiar
    ;mov eax, '*'
siguiente:
    cmp [edi],byte 0 ;fin de cadena
    je salir
    inc ecx
    inc edi

```

```

    jmp    siguiente
salir:

    mov    eax, ecx        ; guardar en eax
    ret                    ; salir

```

Problema 12

```

segment .data
    salto db 0xa

    ,                                d                                db                                '

section .bss
    caract resd 1
    handleConsola resd 1
    caractEscritos resd 1
    longitudCadena resd 1
    ultimoArgumento resd 1
    laux resb 1
    l4 resb 1
    s1 resb 50
    dr resb 500
    espDigito resb 100
    posDigito resb 8
    aux resb 500

segment .text
global _start
extern _GetStdHandle@4
extern _ReadConsoleA@20
extern _WriteConsoleA@20
extern _ExitProcess@4
_start:
    mov edi,d

    push dword -10
        call _GetStdHandle@4
        mov [handleConsola],eax
        xor eax,eax
        mov eax,50
        mov[longitudCadena],eax
        xor eax,eax
        mov eax,0d
        mov [ultimoArgumento],eax
        push dword [ultimoArgumento]
        push dword caractEscritos
        push dword [longitudCadena]
        push dword s1
        push dword [handleConsola]

```

```

        call _ReadConsoleA@20

mov esi,s1
call buscarfin

times 2 call guardar

mov edi,d

call len
;add eax,1
;add eax,2
mov [l4],eax
mov [laux],eax
push laux

call imprimir

push dword-11
xor eax,eax;
call _GetStdHandle@4;
mov [handleConsola],eax;
xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword 30;
push dword d
push dword [handleConsola];
call _WriteConsoleA@20;

push dword-11
xor eax,eax;
call _GetStdHandle@4;
mov [handleConsola],eax;
xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword 1;
push dword salto
push dword [handleConsola];
call _WriteConsoleA@20;

;mov eax,l4
call inverso

push dword-11
xor eax,eax;

```

```

call _GetStdHandle@4;
mov [handleConsola],eax;
xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword 30;
push dword dr
push dword [handleConsola];
call _WriteConsoleA@20;

```

```

push dword -11
xor eax,eax;
call _GetStdHandle@4;
mov [handleConsola],eax;
xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword 1;
push dword salto
push dword [handleConsola];
call _WriteConsoleA@20;

```

```

xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
call _ExitProcess@4;

```

guardar:

```

push dword -10
    call _GetStdHandle@4
    mov [handleConsola],eax
    xor eax,eax
    mov eax,50d
    mov[longitudCadena],eax
    xor eax,eax
    mov eax,0d
    mov [ultimoArgumento],eax
    push dword [ultimoArgumento]
    push dword caractEscritos
    push dword 50
    push dword s1
    push dword [handleConsola]
    call _ReadConsoleA@20
mov esi,s1
inc edi
;dec edi

```

```

    ;dec edi
buscarfin:

    cmp    [esi], byte '*' ;fin de cadena
    jz     fin
    ;inc esi
    mov ecx, 1
    rep movsb
    ;inc edi
    jmp    buscarfin
len:
    xor    ecx, ecx ;limpiar
siguiente:
    cmp    [edi], byte ' ' ;fin de cadena
    jz     salir
    inc    ecx
    inc    edi
    jmp    siguiente
salir:
    mov    eax, ecx      ; guardar en eax
    ret                    ; salir
imprimir:
    mov ecx, espDigito
    mov ebx, 10
    mov [ecx], ebx
    inc    ecx
    mov [posDigito], ecx

bucle:
    mov edx, 0
    mov ebx, 10
    div ebx
    ;push eax
    add edx, 48

    mov ecx, [posDigito]
    mov [ecx], dl
    inc    ecx
    mov [posDigito], ecx

    ;pop eax
    cmp    eax, 0
    jne    bucle

bucle2:
    mov ecx, [posDigito]
;_WriteConsoleA
;    mov eax, 1
;    mov edi, 1
;    mov rsi, ecx
;    mov edx, 1

```

```

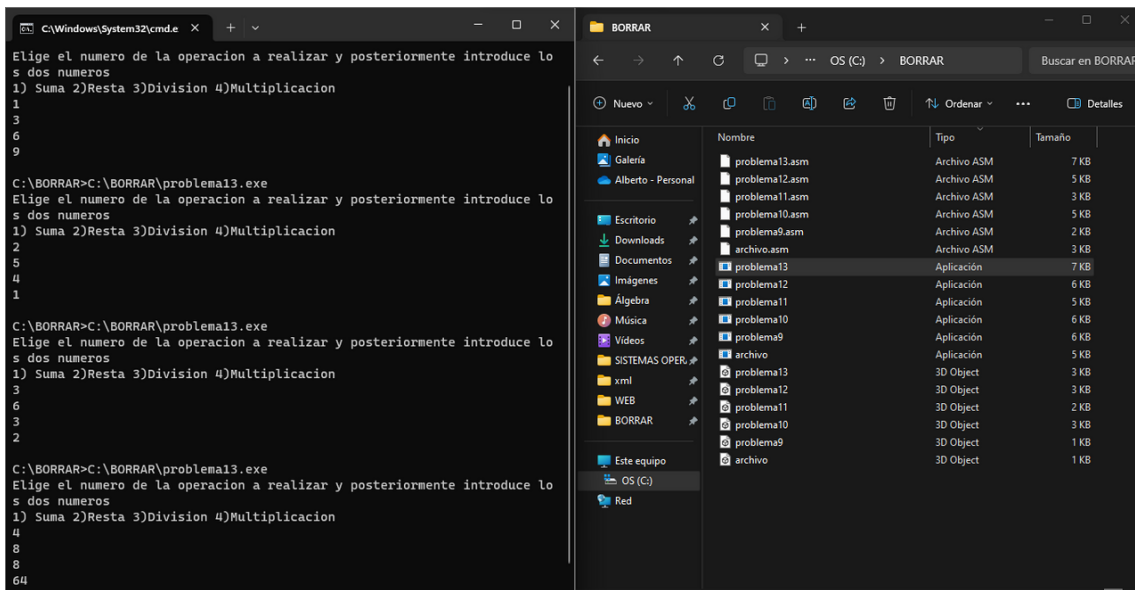
        ;mov dword [caract],
        push dword -11
        xor eax, eax;
        call _GetStdHandle@4;
        mov [handleConsola], eax;
        xor eax, eax;
        mov eax, 0d;
        mov [ultimoArgumento], eax;
        push dword [ultimoArgumento];
        push dword caractEscritos;
        push dword 1;
        push dword ecx
        push dword [handleConsola];
        call _WriteConsoleA@20;

        mov ecx, [posDigito]
        dec ecx
        mov [posDigito], ecx

        cmp ecx, espDigito
        jge bucle2
        ret
inverso:
        mov esi, d
        sub esi, 1
invimprimir:
        cmp [esi], byte 0
        jz invfin
        mov edi, dr
        add edi, [laux]
        mov ecx, 1
        rep movsb
        sub byte [laux], 1
        jmp invimprimir
invfin:
        ret
fin:
        sub edi, 1
        ret

```

Problema 13



```
segment .data
```

```
    intro db 'Elige el numero de la operacion a realizar y posteriormente introduce los dos numeros',10
```

```
    introl equ ($-intro)
```

```
    opc db '1) Suma 2)Resta 3)Division 4)Multiplicacion',10
```

```
    opcl equ ($-opc)
```

```
    nega db '-'
```

```
    negal equ ($-nega)
```

```
    suma db 'Suma',10
```

```
    sumal equ ($-suma)
```

```
    salto db 0xa
```

```
section .bss
```

```
    caract resd 1
```

```
    handleConsola resd 1
```

```
    caractEscritos resd 1
```

```
    longitudCadena resd 1
```

```
    ultimoArgumento resd 1
```

```
    opcion resb 3
```

```
    num1 resb 50
```

```
    num2 resb 50
```

```
    espDigito resb 100
```

```
    posDigito resb 8
```

```
segment .text
```

```
global _start
```

```
extern _GetStdHandle@4
```

```
extern _ReadConsoleA@20
```

```
extern _WriteConsoleA@20
```

```

extern _ExitProcess@4
_start:
push dword -11
xor eax, eax;
call _GetStdHandle@4;
mov [handleConsola], eax;
xor eax, eax;
mov eax, 0d;
mov [ultimoArgumento], eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword intro1;
push dword intro
push dword [handleConsola];
call _WriteConsoleA@20;

push dword -11
xor eax, eax;
call _GetStdHandle@4;
mov [handleConsola], eax;
xor eax, eax;
mov eax, 0d;
mov [ultimoArgumento], eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword opcl;
push dword opc
push dword [handleConsola];
call _WriteConsoleA@20;

push dword -10
call _GetStdHandle@4
mov [handleConsola], eax
xor eax, eax
mov eax, 2
mov [longitudCadena], eax
xor eax, eax
mov eax, 0d
mov [ultimoArgumento], eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword 3
push dword opcion
push dword [handleConsola]
call _ReadConsoleA@20

push dword -10
    call _GetStdHandle@4
    mov [handleConsola], eax
    xor eax, eax
    mov eax, 50

```

```

mov[longitudCadena],eax
xor eax,eax
mov eax,0d
mov [ultimoArgumento],eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword 50
push dword num1
push dword [handleConsola]
call _ReadConsoleA@20

```

```

push dword -10
call _GetStdHandle@4
mov [handleConsola],eax
xor eax,eax
mov eax,50
mov[longitudCadena],eax
xor eax,eax
mov eax,0d
mov [ultimoArgumento],eax
push dword [ultimoArgumento]
push dword caractEscritos
push dword 50
push dword num2
push dword [handleConsola]
call _ReadConsoleA@20

```

```

sub byte [opcion],'0'
cmp byte [opcion],byte 1
je sum
cmp byte [opcion],byte 2
je res
cmp byte [opcion],byte 3
je divi
cmp byte [opcion],byte 4
je multi

```

sum:

```

mov eax, num1
call atoi
push eax

mov eax, num2
call atoi
pop edx
add eax,edx

call imprimir

xor eax,eax;

```

```

    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;
res:
    mov eax, num1
    call atoi
    push eax

    mov eax, num2
    call atoi
    pop edx
    cmp eax,edx
    jg cambio

    sub edx,eax

    mov eax, edx
    call imprimir

    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;
cambio:
    push eax
    push edx

    push dword -11
        call _GetStdHandle@4;
        mov [handleConsola],eax;
        xor eax,eax;
        mov eax,0d;
        mov [ultimoArgumento],eax;
        push dword [ultimoArgumento];
        push dword caractEscritos;
        push dword nega1;
        push dword nega;
        push dword [handleConsola];
        call _WriteConsoleA@20;

    pop edx
    pop eax

    sub eax,edx
    call imprimir

    xor eax,eax;
    mov eax,0d;
    mov [ultimoArgumento],eax;
    push dword [ultimoArgumento];

```

```

        call _ExitProcess@4;
divi:
    mov eax, num1
    call atoi
    push eax

    mov eax, num2
    call atoi
    mov ebx, eax
    pop eax

    div ebx

    ;mov eax, edx
    call imprimir

    xor eax, eax;
    mov eax, 0d;
    mov [ultimoArgumento], eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;

```

```

multi:
    mov eax, num1
    call atoi
    push eax

    mov eax, num2
    call atoi
    mov ebx, eax
    pop eax

    mul ebx

    ;mov eax, edx
    call imprimir

    xor eax, eax;
    mov eax, 0d;
    mov [ultimoArgumento], eax;
    push dword [ultimoArgumento];
    call _ExitProcess@4;

```

```

atoi:
    mov     esi, eax        ; mueve puntero en eax a esi
    mov     eax, 0          ; inicializa con 0
    mov     ecx, 0

```

```

.potencia10:
    xor     ebx, ebx        ; limpia ebx
    mov     bl, [esi+ecx]   ; mueve el numero de bytes ecx
    cmp     bl, 48          ; compara con 48 (0 en ascii)
    jl      .finatoi     ; salta al final si es menor

```

```

cmp     bl, 57          ;compara con 57 (9 en ascii)
jg      .finatoi     ; salta al final si es mayor
cmp     bl, 10          ; compara con enter
je      .finatoi     ; va al final si es igual
cmp     bl, 0           ;compara con final de cadena
jz      .finatoi     ; va al final si es 0

sub     bl, 48          ; convierte el valor en entero
add     eax, ebx        ; agrega el valor al numero
mov     ebx, 10         ; mueve 10 a ebx
mul     ebx             ; multiplica eax por ebx
inc     ecx             ; incrementa ecx, numero de bytes
jmp     .potencia10

```

.finatoi:

```

mov     ebx, 10 ;divide entre 10 pues sobro una potencia de 10
div     ebx
ret

```

imprimir:

```

mov ecx, espDigito
mov ebx, 10
mov [ecx], ebx
inc ecx
mov [posDigito], ecx

```

bucle:

```

mov edx, 0
mov ebx, 10
div ebx
;push eax
add edx, 48

```

```

mov ecx, [posDigito]
mov [ecx], dl
inc ecx
mov [posDigito], ecx

```

```

;pop eax
cmp eax, 0
jne bucle

```

bucle2:

```

mov ecx, [posDigito]
;_WriteConsoleA
; mov eax, 1
; mov edi, 1
; mov rsi, ecx
; mov edx, 1

```

```

;mov dword [caract],

```

```
push dword-11
xor eax,eax;
call _GetStdHandle@4;
mov [handleConsola],eax;
xor eax,eax;
mov eax,0d;
mov [ultimoArgumento],eax;
push dword [ultimoArgumento];
push dword caractEscritos;
push dword 1;
push dword ecx
push dword [handleConsola];
call _WriteConsoleA@20;

mov ecx, [posDigito]
dec ecx
mov [posDigito], ecx

cmp ecx, espDigito
jge bucle2
ret
```