

Practica 3 (ensamblador)



**Saldaña Aguilar Gabriela**

**Materia** SISTEMAS OPERATIVOS

**Profesor** CORTES GALICIA JORGE

**Grupo** 2CM8

**Fecha de entrega** 23/05/15

**Tr**

DESARROLLO

\*Programa que imprime una cadena insertada previamente:

|  |
| --- |
| **CODIGO**  segment .data  cadena db 'Programado en ensamblador para LInux',0XA  segment .text  global \_start  \_start:  mov edx,38d  mov ecx,cadena  mov ebx,1  mov eax,4  int 0X80  mov eax,1  int 0x80 |
|  |

\*Programa que imprime una cadena dada por el ususario

|  |
| --- |
| **CODIGO**  segment .bss  cadena db 50 ;cadena es un flujo de bytes  segment .text  global \_start  \_start:  mov edx,50d ;guardando el tamañode la cadena (50)  mov ecx,cadena ;guardando cadena  mov ebx,0 ;file descriptor, a la entrada(teclado)  mov eax,3 ;read  int 0X80 ;kernel    ;HASTA AQUI YA LEIMOSLACADENA  mov edx,50d ;tamaño  mov ecx,cadena ;lo que va a escribir  mov ebx,1 ;file decriptor, a la salida(terminal)  mov eax,4 ;write  int 0X80 ;kernel  mov eax,1 ;exit  int 0X80 ;kernel |
|  |

Programa que muestra en pantalla la longitud de una cadena que haya sido ingresada por el usuario.

|  |
| --- |
| **CODIGO**  .MODEL **small**  .stack 100h  segment .data  LF equ 10  OD equ 13  mens db "Introduzca cadena "**,**LF**,**OD**,**"$"  salto db LF**,**OD**,**"$"  cadena label **byte**  chare db 30  max db 30  campo db **?**  num DW 0  start**:**  **mov** **ax,**@data  **mov** **ds,ax**  **mov** **ah,**09  **lea** **dx,**mens  **int** 21h  **mov** **ah,**0ah  **lea** **dx,** cadena  **int** 21h  **mov** **ah,**09  **lea** **dx,**salto  **int** 21h  **lea** **si,**campo  **mov** **cx,**00  loop0**:**  **mov** **al,[si]**  **cmp** **al,**0dh ;si detecta un enter  **je** salir ;deja de contar los caracteres  **inc** **cx**  **inc** **si**  **add** num**,**1  **jmp** loop0  salir**:**  **mov** **bl,**10  **mov** **ax,cx**  **div** **bl**  **mov** **dx,ax**  **or** **dx,**3030h  **mov** **ah,**02h  **int** 21h  **xchg** **dl,dh**  **mov** **ah,**02h  **int** 21h  **mov** **ax,** 4C00h  **int** 21h  **mov** **ecx,**num  **mov** **ebx,**1 ;salida estandar  **mov** **eax,**4 ;numero de llamada al sistema "sys\_write"  **int** 0x80 ;interrupcion de llamadas al sistema del kernel de linux  **mov** **eax,**1 ;numero de llamada al sistema "sys\_exit"  **int** 0x80 ;interrupcion de llamadas al sistema del kernel de linux  end start |
|  |

Programa que solicite al usuario que introduzca 5 cadenas de caracteres. Posteriormente el programa deberá mostrarle al usuario dos cadenas: una cadena que es el resultado de concatenar las 5 introducidas y la otra que es el resultado de concatenar las 5 cadenas introducidas pero en orden inverso.

|  |
| --- |
| **CODIGO**  .section .rodata  .LC0:  .string "Introduzca cadena 1"  .LC1:  .string "Introduzca cadena 2"  .LC2:  .string "Introduzca cadena 3"  .LC3:  .string "Introduzca cadena 4"  .LC4:  .string "Introduzca cadena 5"  .align 4  .LC5:  .string "La concatenacion de las 5 cadenas es: %s\n"  .align 4  .LC6:  .string "La concatenacion en sentido inverso es: %s\n"  .align 4  .LC7:  .string "La longitud de la cadena concatenada es: %d\n"  .text  .globl main  .type main, @function  main:  .LFB0:  .cfi\_startproc  pushl %ebp  .cfi\_def\_cfa\_offset 8  .cfi\_offset 5, -8  movl %esp, %ebp  .cfi\_def\_cfa\_register 5  pushl %edi  pushl %ebx  andl $-16, %esp  subl $512, %esp  movl %gs:20, %eax  movl %eax, 508(%esp)  xorl %eax, %eax  movl $0, 44(%esp)  movl $0, 48(%esp)  movl $0, 52(%esp)  movl $.LC0, (%esp)  .cfi\_offset 3, -16  .cfi\_offset 7, -12  call puts  movl stdin, %eax  movl %eax, (%esp)  call fflush  leal 358(%esp), %eax  movl %eax, (%esp)  call gets  movl $.LC1, (%esp)  call puts  movl stdin, %eax  movl %eax, (%esp)  call fflush  leal 388(%esp), %eax  movl %eax, (%esp)  call gets  movl $.LC2, (%esp)  call puts  movl stdin, %eax  movl %eax, (%esp)  call fflush  leal 418(%esp), %eax  movl %eax, (%esp)  call gets  movl $.LC3, (%esp)  call puts  movl stdin, %eax  movl %eax, (%esp)  call fflush  leal 448(%esp), %eax  movl %eax, (%esp)  call gets  movl $.LC4, (%esp)  call puts  movl stdin, %eax  movl %eax, (%esp)  call fflush  leal 478(%esp), %eax  movl %eax, (%esp)  call gets  movl $0, 44(%esp)  jmp .L2  .L3:  leal 358(%esp), %eax  addl 44(%esp), %eax  movzbl (%eax), %edx  leal 58(%esp), %eax  addl 48(%esp), %eax  movb %dl, (%eax)  addl $1, 48(%esp)  addl $1, 44(%esp)  .L2:  movl 44(%esp), %ebx  leal 358(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L3  movl $0, 44(%esp)  jmp .L4  .L5:  leal 388(%esp), %eax  addl 44(%esp), %eax  movzbl (%eax), %edx  leal 58(%esp), %eax  addl 48(%esp), %eax  movb %dl, (%eax)  addl $1, 48(%esp)  addl $1, 44(%esp)  .L4:  movl 44(%esp), %ebx  leal 388(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L5  movl $0, 44(%esp)  jmp .L6  .L7:  leal 418(%esp), %eax  addl 44(%esp), %eax  movzbl (%eax), %edx  leal 58(%esp), %eax  addl 48(%esp), %eax  movb %dl, (%eax)  addl $1, 48(%esp)  addl $1, 44(%esp)  .L6:  movl 44(%esp), %ebx  leal 418(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L7  movl $0, 44(%esp)  jmp .L8  .L9:  leal 448(%esp), %eax  addl 44(%esp), %eax  movzbl (%eax), %edx  leal 58(%esp), %eax  addl 48(%esp), %eax  movb %dl, (%eax)  addl $1, 48(%esp)  addl $1, 44(%esp)  .L8:  movl 44(%esp), %ebx  leal 448(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L9  movl $0, 44(%esp)  jmp .L10  .L11:  leal 478(%esp), %eax  addl 44(%esp), %eax  movzbl (%eax), %edx  leal 58(%esp), %eax  addl 48(%esp), %eax  movb %dl, (%eax)  addl $1, 48(%esp)  addl $1, 44(%esp)  .L10:  movl 44(%esp), %ebx  leal 478(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L11  leal 58(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  subl $1, %eax  movl %eax, 52(%esp)  movl $0, 44(%esp)  jmp .L12  .L13:  leal 58(%esp), %eax  addl 52(%esp), %eax  movzbl (%eax), %edx  leal 208(%esp), %eax  addl 44(%esp), %eax  movb %dl, (%eax)  subl $1, 52(%esp)  addl $1, 44(%esp)  .L12:  movl 44(%esp), %ebx  leal 58(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  subl $1, %eax  cmpl %eax, %ebx  jb .L13  movl $.LC5, %eax  leal 58(%esp), %edx  movl %edx, 4(%esp)  movl %eax, (%esp)  call printf  movl $.LC6, %eax  leal 208(%esp), %edx  movl %edx, 4(%esp)  movl %eax, (%esp)  call printf  leal 58(%esp), %eax  movl $-1, 28(%esp)  movl %eax, %edx  movl $0, %eax  movl 28(%esp), %ecx  movl %edx, %edi  repnz scasb  movl %ecx, %eax  notl %eax  leal -1(%eax), %edx  movl $.LC7, %eax  movl %edx, 4(%esp)  movl %eax, (%esp)  call printf  movl 508(%esp), %edx  xorl %gs:20, %edx  je .L14  call \_\_stack\_chk\_fail  .L14:  leal -8(%ebp), %esp  popl %ebx  .cfi\_restore 3  popl %edi  .cfi\_restore 7  popl %ebp  .cfi\_def\_cfa 4, 4  .cfi\_restore 5  ret  .cfi\_endproc |
|  |

Programa que hace la funcion de una calculadora usando valores insertados previamente

|  |
| --- |
| **CODIGO**  jmp 220  e 01 03 "1) Suma" 0a 0d "$"  e 01 20 "2) Resta" 0a 0d "$"  e 01 40 "3) Multiplicacion" 0a 0d "$"  e 01 60 "4) Division" 0a 0d "$"  e 01 80 "Selecciona una opcion" 0a 0d "$"  e 02 00 "Regresar menu s/n" 0a 0d "$"  a 22 0  mov ah,09 ;Comienza por mostrar el menú  mov dx,0103  int 21  mov ah,09  mov dx,0120  int 21  mov ah,09  mov dx,0140  int 21  mov ah,09  mov dx,0160  int 21  mov ah,09  mov dx,0180  int 21  mov ah,01  int 21  cmp al,31  je 2 5d  cmp al,32  je 2 93  cmp al,33  je 2 cA  cmp al,34  jmp 302  cmp al,35  jmp 33F  mov ah,01 ;Si la opción elegida es 1, comienza la suma  int 21  sub al,30  mov bl,al  mov ah,01  int 21  sub al,30  add al,bl  or ax,3030  mov bx,ax  sub bh,01  mov ah,02  mov dl,bh  int 21  mov ah,02  mov dl,bl  int 21  mov ah,09  mov dx,0200  int 21  mov ah,01 ;Si la opción elegida es 2, comienza la resta  int 21  cmp al,73  je 2 20  int 20  mov ah,01  int 21  sub al,30  mov bl,al  mov ah,01  int 21  sub al,30  sub al,bl  aas or ax,3030  mov bx,ax  sub bh,01  mov ah,02  mov dl,bh  int 21  mov ah,02  mov dl,bl  int 21  mov ah,09  mov dx,0200  int 21  m ov ah,01 ;Si la opción elegida es 3, comienza la multiplicación  int 21  cmp al,73  jmp 220  int 20  mov ah,01  int 21  sub al,30  mov bl,al  and bl,0f  mov ah,01  int 21  sub al,30  mul al,bl  aam or ax,3030  mov bx,ax  mov ah,02  mov dl,bh  int 21  mov ah,02  mov dl,bl  int 21  mov ah,09  mov dx,0200  int 21  mov ah,01 ;Si la opción elegida es 4, comienza la división  int 21  cmp al,73  jmp 220  int 20  mov ah,01  int 21  sub ah,01  sub al,30  mov cx,ax  mov ah,01  int 21  sub ah,01  sub al,30  mov bx,ax  aad mov ax,cx  mov cx,0000  mov dx,0000  div bx  mov cx,dx  mov ah,02  mov dl,al  add dl,30  int 21  mov ah,09  mov dx,0200  int 21  mov ah,01 ;al terminar de hacer la operación, si la letra digitada es s re torna desde el comienzo.  int 21  clc cmp al,73  je 2 20  int 20 |
|  |

|  |
| --- |
|  |

**CONCLUSIONES**