

Universidad Autónoma de Baja California

Ingeniería en Computación



Facultad de Ciencias Químicas e Ingeniería

Organización y Arquitectura de Computadoras

Práctica 9

Zavala Román Irvin Eduardo

Grupo: 551

20/10/2021

Periodo 2021-2

Objetivo

Seleccionar las instrucciones de control de flujo del programa adecuadas, para desarrollar aplicaciones de sistemas basados en microprocesador, mediante el análisis de su funcionalidad, de forma responsable y eficiente.

Desarrollo

1. Cree un programa llamado P9.asm que contenga las siguientes rutinas:

a) printNumBase: imprime el número en EAX en el formato según la base dada en el registro BL. La representación del número en caracteres ASCII además de desplegarse en pantalla, también se almacena en una cadena en memoria apuntada por ESI.

En el código anterior se ejemplifica la impresión del registro EAX en decimal y hexadecimal, sin embargo, el procedimiento debe ser funcional para cualquier base solicitada que sea imprimible de acuerdo al límite de caracteres en la tabla ASCII. El procedimiento debe ser genérico, no realice invocaciones a printBin o printHex, haga la conversión por medio de divisiones.

b) SetBit: activa un bit del registro EAX. El número de bit a activar está dado por CL.

c) ClearBit: desactiva un bit del registro EAX. El número de bit a desactivar está dado por CL.

d) NotBit: invierte un bit del registro EAX. El número de bit a invertir está dado por CL.

e) TestBit: copia un bit del registro EAX a la bandera de acarreo. El bit a copiar está dado por CL.

Resultado

El código es NASM queda de la siguiente manera:

```
%include "pc_io.inc"
```

```
section .data
```

```
section .bss
```

```
    cad    resb 32
```

```
    cadena    resb 32
```

```
    aux    resb 32
```

```
section .text
```

```
    global _start:
```

```
_start:
```

```
    call clrscr
```

```
    mov eax, 0F4567h
```

```
    mov ebx, 10
```

```
    call printNumBase ;Si convierto F4567 a decimal la salida debe ser 1000807
```

```
    call salto
```

```
    mov edx,cadena ;Para ver si se guardo el resultado de printNumBase en
```

```
    cadena
```

```
    call puts
```

```
    call salto
```

```
    mov ecx,3 ;Si activo el 3er bit de eax, el 7 se convierte en F
```

```
    call setBit
```

```
    call salto
```

```
    mov ecx,3 ;Se deberia desacer el cambio del setBit
```

```
    call clearBit
```

```
    call salto
```

```
    mov ecx, 4 ;Si niego el 4to bit el F4567 deberia pasar a F4577
```

```
    call notBit
```

```
    call salto
```

```
    mov ecx, 3 ;Si recorro con carry el 3er bit deberia quedar en cf
```

```
    call testBit
```

```
    mov eax, 1 ;FIN
```

```
    mov ebx,0
```

```
    int 80h
```

```
printNumBase:
```

```
    push eax ;Salvamos los registros
```

```
    push ecx
```

```

    push edx
    mov edx,0    ;Reseteo edx y ecx para las divisiones
    mov ecx,0
.division:
    div ebx      ;eax=eax/ebx, edx = eax%ebx
    add edx,'0'  ;Convierto edx a caracter
    cmp edx,'9'
    jbe .push_edx
.push_edx:
    push edx     ;Guardo edx en la pila
    inc ecx
    mov edx, 0   ;Reseteo edx
    cmp eax,0    ;Si eax es diferente de 0 es porque no se acabo la division
    jne .division
    mov ebx,0
.print:
    pop edx
    mov eax, edx
    mov esi,cadena
    mov byte[esi+ebx],al
    inc ebx
    loop .print
    pop edx      ;Recuperar registros de la pila
    pop ecx
    pop eax
    ret

```

```

setBit:
    mov esi,cad ;Pa ver que hay en eax
    call printHex
    call salto
    mov ebx,1
    shl ebx,cl
    or eax,ebx
    mov esi,cad
    call printHex

    ret

```

```

clearBit:
    mov esi,cad ;Pa ver que hay en eax
    call printHex
    call salto
    mov ebx,1
    shl ebx,cl
    not ebx
    and eax,ebx
    mov esi,cad
    call printHex

```

ret

notBit:

```
mov esi,cad ;Pa ver que hay en eax  
call printHex  
call salto  
mov ebx,1  
shl ebx,cl  
xor eax, ebx  
mov esi,cad  
call printHex  
ret
```

testBit:

```
push eax  
mov esi,cad ;Pa ver que hay en eax  
call printHex  
call salto  
mov ebx,1  
shl ebx,cl  
mov ecx,ebx  
rcr eax,cl  
pop eax  
ret
```

printHex:

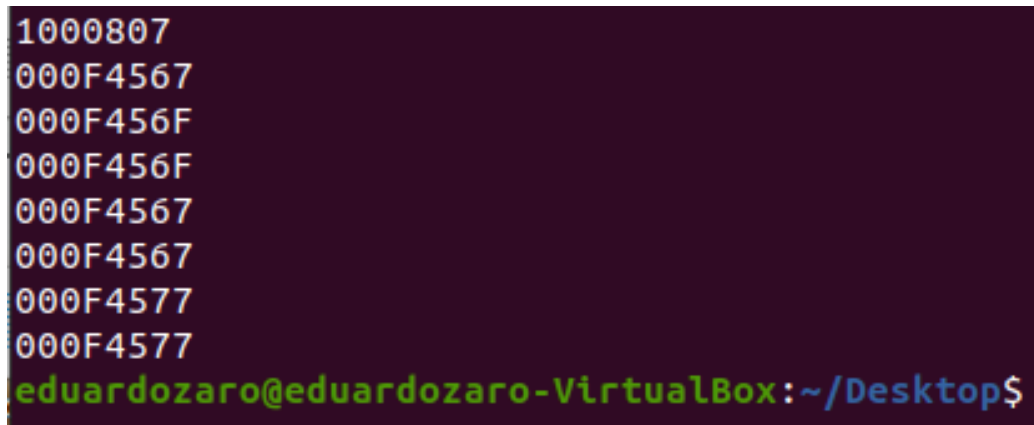
```
pushad  
mov edx, eax  
mov ebx, 0fh  
mov cl, 28  
.nxt:shr eax,cl  
.msk:and eax,ebx  
    cmp al, 9  
    jbe .menor  
    add al,7  
.menor:add al,'0'  
    mov byte [esi],al  
    inc esi  
    mov eax, edx  
    cmp cl, 0  
    je .print  
    sub cl, 4  
    cmp cl, 0  
    ja .nxt  
    je .msk  
.print:mov eax, 4  
    mov ebx, 1  
    sub esi, 8  
    mov ecx, esi  
    mov edx, 8
```

```

    int 80h
    popad
    ret
salto:
    push eax
    mov al,10    ;SALTO
    call putchar
    int 80h
    pop eax
    ret

```

Dando el siguiente resultado:



```

1000807
000F4567
000F456F
000F456F
000F4567
000F4567
000F4577
000F4577
eduardozaro@eduardozaro-VirtualBox:~/Desktop$

```

Figura 1: Resultado del P9.asm

Conclusión

El enmascaramiento es algo que ya habíamos es sencillo de aplicar en NASM si sabemos como funcionan de antemano, aparte esta practica me forzó a usar saltos condicionados para poder bifurcar el código y obtener el resultado deseado, aunque la aplicación de los ciclos pudo ser mejor, cumple con lo especificado aplicando lo visto en clase.

Dificultades en el desarrollo

El código de conversión me costó bastante en todos los aspectos, desde saber de donde empezar hasta la propia codificación, pero aun así se logró hacer la función.