Universidad Autónoma de Baja California
Ingeniero en computación

Organización y Arquitectura De Computadoras

Practica 7

Nombre Del Alumno: López Mercado Brayan

Matrícula: 1280838

Grupo: 551

Docente: José Isabel García Rocha

Fecha de entrega: 19 de octubre del 2023

Objetivo

Seleccionar las instrucciones de transferencia de datos correctas en aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y responsable.

Desarrollo

1- ***Copie el código del listado 1 en un archivo llamado Ej_Lib.asm. Descargue de Moodle la biblioteca de funciones libpc_io (archivos llamados libpc_io.a y pc_io.inc). Abra una terminal en Linux y ensamble el código con NASM por medio del comando:***

```
nasm -f elf Ej_Lib.asm .
```

```
:~/OacAgain/p7$ nasm -f elf Ej_Lib.asm
```

2- ***Encadene el archivo por medio de uno del siguiente comando:***

```
ld -m elf_i386 -s -o Ej_Lib Ej_Lib.o libpc_io.a
```

```
:~/OacAgain/p7$ ld -m elf_i386 -s -o p7 Ej_Lib.o libpc_io.a
```

3- ***Ejecute el archivo por medio del comando:***

```
./Ej_Lib
```

```
:~/OacAgain/p7$ ./p7
```

El programa solicita al usuario el ingreso de su nombre y despliega un mensaje de saludo. Posteriormente, solicita un carácter y lo despliega en pantalla.

1- **Cree un programa llamado p7.asm que contenga la subrutina printHex de la practica 5, la cual recibe en EAX un valor que se quiere imprimir en hexadecimal. Agregue a p7.asm las instrucciones necesarias para hacer lo que se indica a continuación:**

   a) ***Reservar dos espacios en memoria no inicializados, uno de 64 bytes etiquetado como A y otro de 1 byte etiquetado como C.***

```
;Inciso A
A resb 64
C resb 1
```

Figura 1: Reservación de espacios en memoria.

b) Solicitar una cadena que se almacene en A e imprimirla usando la biblioteca.

```
;Inciso B
mov edx,IncisoB
call puts
mov eax,3
mov ebx,0
mov ecx,A
mov edx,64
int 0x80
mov edx,A
call puts
```

Figura 2: Instrucciones necesarias para capturar una cadena desde terminal.

Descripción: la manera más primitiva de capturar una cadena en lenguaje ensamblador es por medio de las llamadas al Kernel (en este caso para la lectura de terminal se usa la 3), ECX contiene la dirección donde se guardará la cadena (A en este caso), en este caso, la cadena tiene una longitud de 64 bytes, y se realiza una interrupción para tomar el dato de la terminal.

c) Solicitar un carácter y almacenarlo en C sin utilizar y utilizando la biblioteca, e imprimirlo utilizando la biblioteca.

```
;Inciso C
mov edx,IncisoC ;Sin Biblioteca
call puts
mov eax,3
mov ebx,0
mov ecx,C
mov edx,1
int 0x80
mov al,[C]
call putchar
call new_line
call getch ;Con Biblioteca
call putchar
```

Figura 3: Instrucciones necesarias para capturar un carácter desde terminal.

Descripción: en este inciso se realiza lo mismo que el inciso B, con la diferencia de en este caso, el dato se guardara en C y la longitud es de un byte. Por algún extraño motivo, la subrutina getch y getche de la biblioteca no funcionan correctamente, en pocas palabras, no capturan datos; aunque para evitar problemas se escribió el código como si funcionaran. Para mostrar el carácter capturado se copió el contenido de C a AL para imprimirlo utilizando la subrutina putchar.

d) Multiplicación con signo de X por Y y guardarlo en tmp con modo de direccionamiento de memoria directa.

```
;Inciso D
mov edx,IncisoD
call puts
call clearReg
mov eax,[X]
mov ebx,[Y]
imul ebx
mov dword[0x0804A058],eax
mov eax,[tmp]
call printHex
call new_line
mov [W],eax
mov eax,[W]
call printHex
call new_line
```

Figura 4: Instrucciones necesarias para multiplicar X e Y.

Descripción: primero se realiza una limpieza de registros para evitar que estos afecten en el resultado de las operaciones, luego se cargan los contenidos de las variables X e Y a EAX y EBX respectivamente, para posteriormente realizar la multiplicación con signo donde EAX contiene el resultado, luego se realiza una copia del contenido de EAX a la dirección de tmp, para la comprobación se copia el contenido de tmp a EAX para mostrarlo en pantalla por medio de printHex; se guarda el contendio de EAX (el resultado de la multiplicación) en W, para la comprobación se copia el contenido de W a EAX para mostrarlo en terminal por medio de printHex.

e) División de W entre X y guardarlo en Z con modo de direccionamiento de memoria indirecta.

```
;Inciso E
mov edx,IncisoE
call puts
cdq
mov eax,[W]
idiv dword[X]
mov [W],eax
mov eax,[W]
call printHex
call new_line
lea ecx,Z
mov [ecx],eax
mov eax,[Z]
call printHex
call new_line
```

Figura 5: Instrucciones necesarias para dividir W entre X.

Descripción: en este caso debido a que realmente no se ocupan más de 32 bits en W se puede realizar una copia directa del contenido a EAX y luego usar IDIV dword[X] para realizar la división y guardar el resultado en EAX; luego se

pasa el contenido a W para guardarlo, para la comprobación se realiza una copia del contenido de W a EAX para mostrarlos en terminal por medio de printHex. Para guardar el resultado en Z se carga a ECX la dirección efectiva de Z por medio de la instrucción LEA, luego se copia el contenido a la dirección contenida por ECX; para la verificación se copia el contenido de Z a EAX para mostrarlo en terminal por medio de printHex.

f) Negar del octavo byte al onceavo byte de tmp con la instrucción NEG, guardarlo en tmp con modo de direccionamiento base más desplazamiento.

```
;Inciso F
mov edx,IncisoF
call puts
lea ebx,tmp
neg byte [ebx + 8]
neg byte [ebx + 9]
neg byte [ebx + 10]
neg byte [ebx + 11]
neg byte [ebx + 12]
mov eax,[tmp+8]
call printHex
mov eax,[tmp+4]
call printHex
mov eax,[tmp]
call printHex
call new_line
```

Figura 6: Instrucciones necesarias para negar los bytes 8 -> 11 de tmp.

Descripción: primero se carga en EBX la dirección efectiva de tmp por medio de la instrucción LEA, luego se procede a obtener el complemento A2 de los bytes 8 a 11 de tmp, para mostrar los cambios a tmp, se pasa a EAX las partes donde encontrarían los bytes negados hasta los bytes menos significativos y se muestran en terminal por medio de printHex.

Por cada inciso, despliegue en terminal el nuevo valor de variable o registro modificado. Haga uso de la rutina printHex para desplegar valores numéricos, puts para desplegar cadenas y putchar para caracteres.
Agregar antes de cada impresión una cadena que indique que inciso es el que se está mostrando en terminal.

Subrutinas Auxiliares:

```
clearReg:
    xor eax,eax ; Limpieza De Registros
    xor ebx,ebx
    xor ecx,ecx
    xor edx,edx
    ret
```

Figura 7: Instrucciones para limpiezas de registros.

Resultados

Sección 1:

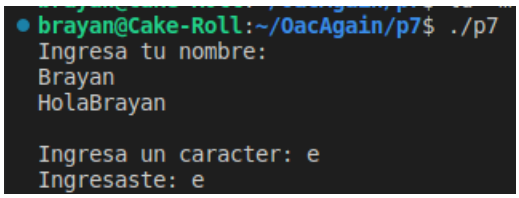

Figura 8: Salida en terminal para la sección 1.

Sección 2:

Inciso A:

No hay salida en terminal

Inciso B:

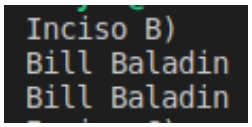

Figura 9: Salida en terminal para Inciso B.

Inciso C:

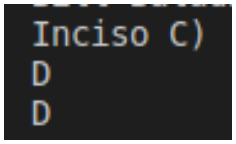

Figura 10: Salida en terminal para Inciso C.

Inciso D:

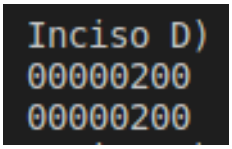

Figura 11: Salida en terminal para Inciso D.

Inciso E:

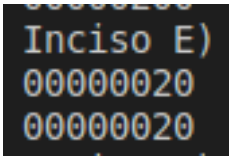

Figura 12: Salida en terminal para Inciso E.

Inciso F:

Figura 13: Salida en terminal para el Inciso F.

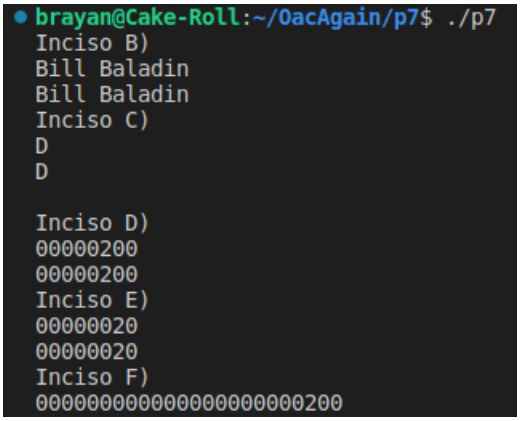

Figura 14: Salida completa en terminal de la sección 2

Link a GitHub Con Código Completo

https://github.com/BrayanLMercado/OAC_Practica7_v2.git

Conclusiones y comentarios

- Se repasó el uso de las instrucciones aritméticas con signo, específicamente la multiplicación y la división, aunque al final ninguno de los números contaba con el signo en su cifra más significativa.
- La forma que se planteó la práctica fue un tanto abstracta en algunas secciones, específicamente en la segunda, que necesidad existía de guardar los resultados en dos variables diferentes cuando algunas de ellas no se volvían a utilizar.
- En caso de que no se hubieran tomado incisos de la práctica anterior, la practica sería mucho más simple.

Dificultades en el desarrollo

- Las subrutinas getch y getche, no funcionaban como debían para evitar ese problema se piensa utilizar la forma larga para capturar los caracteres y puts para mostrarlos en la terminal.
- Para guardar los datos en tmp se tuvo que modificar la dirección debido a que esta cambiaba con la declaración de variables que faltaban para que las demás subrutinas funcionaran bien.

Referencias