Universidad Autónoma de Baja California
Ingeniero en computación

Organización y Arquitectura De Computadoras

Practica 6

Nombre Del Alumno: López Mercado Brayan

Matrícula: 1280838

Grupo: 551

Docente: José Isabel García Rocha

Fecha de entrega: 12 de octubre del 2023

Objetivo

Identificar los errores en un código de ensamblador, así mismo aplicar correctamente las instrucciones aritméticas en aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y responsable.

Desarrollo

Cargar el código anexado a la practica en Moodle y subirlo a google Colab para trabajar con él en ensamblador.

1- **Corregir para que cumpla lo que se desea realizar correctamente.**

```
section .data
    ;1) corregir para que
    NL: db  13,10
    NL_L:    equ $-NL
```

Figura 1: Sección .data corregida.

Descripción: Se le agrego el número que representa el carácter de final de cadena (10) a NL, y los ":" que le hacían falta para funcionar.

2- **Corregir para que cumpla lo que se desea realizar correctamente.**

```
section .bss
    ;2) corregir para que cumpla
    X       resb 4
    Y       resb 4
    Z       resb 4
    W       resb 8
    tmp     resb 32
    cad     resb 16
```

Figura 2: Sección .bss corregida.

Descripción: La variable W se le asignaron 8 bytes de memoria (64 bits) ya que esta nunca los supera.

3- **Inicialización de variables X=0x10, Y=0x20, Z=0x30 y W=4000000000000000h.**

```
mov dword [X], 0x10
mov dword [Y], 0x20
mov dword [Z], 0x30
mov dword [W], 0x40000000
mov dword [W+4],0x00000000
```

Figura 3: Inicialización de variables.

Nota: En este caso se guardó la parte más significativa de W a partir de la dirección de inicial de W debido a que resulta más entendible, aunque en realidad debería de estar al revés por la forma en la que maneja la memoria.

4- Imprimir el paso anterior y comprobar que se cumpla lo que se les indico.

```
;4 imprimir y comprobar
mov eax,[X]
call printHex
call salto_linea
mov eax,[Y]
call printHex
call salto_linea
mov eax,[Z]
call printHex
call salto_linea
mov edi,0x0
mov eax,[W+edi*4]
call printHex
mov edi,0x1
mov eax,[W+edi*4]
call printHex
call salto_linea
```

Figura 4: Instrucciones necesarias para mostrar los resultados en la terminal.

Descripción: En estas instrucciones se realiza una copia de los contenidos de las variables al registro EAX para luego ser mostrados en la terminal por medio de printHex y un salto de línea para separar a cada una de ellas, en el caso de W, primero se pasa la parte más significativa sin hacer el salto de línea después de mostrarlo en terminal con la parte menos significativa de manera que parezca una sola cadena.

5- Suma de X e Y con modo de direccionamiento de registro y guardarlo en Z.

```
mov eax,[X]
mov ebx,[Y]
add eax,ebx
mov [Z],eax
```

Figura 5: Instrucciones necesarias para sumar X e Y con modo de registro.

Descripción: primero se copian los contenidos de X e Y a EAX y EBX respectivamente para luego ser sumar los contenidos de dichos registros cuyo resultado se guarda en EAX y ese resultado se guarda en Z.

6- Imprimir el paso anterior y comprobar que se cumpla lo que se les indico.

```
;6 imprimir y comprobar
call printHex
call salto_linea
```

Figura 6: Instrucciones necesarias para imprimir el resultado de Z en terminal.

Nota: En este caso no es necesario pasar el contenido de Z a EAX debido a que este registro ya contenía el resultado de la suma, que es el mismo que contiene la variable Z.

7- Resta de Y a W y guardarlo en tmp con modo de direccionamiento indirecto.

```
mov eax,[Y]
mov ebx,[W] ;Parte Alta
mov ecx,[W+4]; Parte Baja
sub [W+4],eax
mov eax,[W+4] ; Resultado Parte Baja
sub dword[W],0x1   ; Resultado Parte Alta
mov edx,tmp
mov ebx,[W] ;Parte Baja
mov [edx],eax
mov [edx+4],ebx
```

Figura 7: Instrucciones necesarias para restarle Y a W y guardarlo en tmp.

Descripción: Primero se utiliza el registro EAX para guardar el valor de Y, EBX contiene la parte más significativa de W [32 bits], mientras ECX contiene la parte menos significativa [32 bits], luego a [W+4] se le resta el valor de EAX, lo que será la parte menos significativa de W, para la parte más significativa simplemente se le resta 0x1, para guardar el resultado en tmp de manera indirecta se utiliza EDX para guardar la dirección de tmp, se realiza una copia del contenido de W al registro EBX, para finalizar, se pasa el contenido de EAX a EDX lo que será la parte menos significativa de tmp, el siguiente paso funciona de manera similar, solo que este contendrás la parte más significativa.

8- Imprimir el paso anterior y comprobar que se cumpla lo que se les indico.

```
;8 imprimir y comprobar
mov eax,[tmp+4]
call printHex
mov eax,[tmp]
call printHex
call salto_linea
```

Figura 8: Instrucciones necesarias para imprimir el contenido de tmp en terminal.

Descripción: se pasa la parte más significativa de tmp a EAX se muestra en terminal por medio de printHex, luego se pasa la parte menos significativa a EAX para mostrarla en terminal.

9- Incremento en 16777216 decimal a W con modo de direccionamiento base más índice escalado trabajando con movimientos de 8 bits.

```
mov edi,0x7
inc byte[W+edi*1] ;Parte Baja
inc dword[W] ;Parte Alta
```

Figura 9: Instrucciones necesarias para imprimir el contenido de tmp en terminal.

Descripción: se le pasa a EDI el offset donde se entra el byte a incrementar (recordando que el número está almacenado al revés), luego se realiza el incremento para el byte por medio de direccionamiento de índice escalado (aunque solo al final el 1 se podría quitar), el último incremento se realiza debido a que el bit incrementado producía un acarreo, para compensarlo simplemente se incrementa la parte más significativa.

10-Imprimir el paso anterior y comprobar que se cumpla lo que se les indico.

```
mov eax,[W] ;Parte Alta
call printHex
mov eax,[W+4] ;Parte Baja
call printHex
call salto_linea
```

Figura 10: Instrucciones necesarias para imprimir el contenido de W en terminal.

11-Sumar W con W sin utilizar memoria en la suma y guardarlo en W.

```
mov ebx,[W] ; Parte Alta
mov ecx,[W+4]; Parte Baja
add ecx,ecx
mov [W+4],ecx
adc ebx,ebx
mov [W],ebx
```

Figura 11: Instrucciones necesarias para sumar W con W.

Descripción: Se pasa a EBX la parte más significativa y a ECX la menos significativa, se suma la parte menos significativa con ella misma cuyo se resultado se encuentra en ECX para luego guardarse en la parte menos significativa de W, para la parte más significativa se utiliza la suma con acarreo para compensar el acarreo causado en la parte menos significativa y ese resultado se encuentra en EBX, luego se pasa a la parte más significativa.

12-Imprimir el paso anterior y comprobar que se cumpla lo que se les indico.

```
mov eax,[W] ;Parte Alta
call printHex
mov eax,[W+4] ;Parte Baja
call printHex
call salto_linea
```

Figura 12: Instrucciones necesarias para imprimir el contenido de W en terminal.

Resultados En Terminal

Inciso 4:

```
brayan@Cake-Roll:~/OacAgain/p6$ ./P6
00000010
00000020
00000030
4000000000000000
```

Inciso 6:

```
00000030
```

Inciso 8:

```
3FFFFFFFFFFFFFE0
```

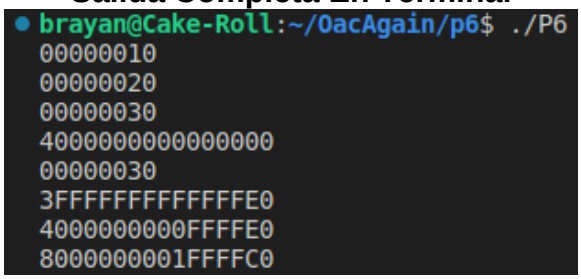
Inciso 10:

```
4000000000FFFFE0
```

Inciso 12:

```
8000000001FFFFC0
```

Salida Completa En Terminal



Repositorio Con El Código Completo

https://github.com/BrayanLMercado/OAC_Practica6_v2.git

Conclusiones y Comentarios

- La práctica ayudo a reforzar el cómo realizar operaciones aritméticas de números que exceden los 32 bits y el invertir partes específicas de un número y el detectar errores de un código de ensamblador hecho por otra persona.

- Aunque los números se encontraban al revés, se logró obtener los resultados esperados, los cuales fueron comprobados a través de una calculadora.

Dificultades en el desarrollo

- Debido a que los números fueron almacenados al revés, la lógica original del código se tuvo que cambiar, los valores numéricos de los offsets fueron obtenidos por prueba y error.
- En la parte de resta se tuvo que dejar el código con la lógica inversa debido a que cuando se intentó cambiar, se obtenía un resultado erróneo.

Referencias