



**Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño**



Ingeniería en Software y Tecnologías Emergentes

Materia: Organización de Computadoras

Alumno: Jesus Eduardo Rodríguez Ramírez

Profesor: Jonatan Crespo Ragland

Grupo 932

Trabajo: Taller 5

Ensenada, B.C; a 27 de septiembre del 2024

1. De acuerdo con el siguiente programa en ensamblador, identifica, desarrolla y describe su funcionamiento. (Qué secciones conforman al programa, qué tipo de registros se utilizan y por qué.

```
section .data msg db 'Imprimir input del teclado: ',0 ;  
Mensaje que se mostrará antes de la entrada, //se  
cambió por el 'input: '
```

```
section .bss input resb 1 ; Espacio para almacenar el  
carácter ingresado  
sum resb 1 ; Espacio para almacenar la suma
```

```
section .text  
global _start
```

```
_start:  
; Mostrar mensaje en consola  
mov eax, 4  
mov ebx, 1  
mov ecx, msg ; dirección del mensaje  
mov edx, 30 ; longitud del mensaje //se extendió la  
longitud del mensaje para que mostrar todo el mensaje  
completo que se pide  
int 0x80
```

```
; Leer un carácter desde el teclado
```

```
mov eax, 3  
mov ebx, 0  
mov ecx, input ; dirección para almacenar la entrada  
mov edx, 80 ; leer 1 byte (1 carácter) //se extiende la  
longitud de bytes que lee para el mensaje que se pide  
en el taller  
int 0x80
```

```
; Mostrar el carácter ingresado  
mov eax, 4 ; syscall número 4 es write (sys_write)  
mov ebx, 1 ; descriptor de archivo 1 es stdout
```

```
mov ecx, input ; dirección del carácter
mov edx, 20 ; longitud del carácter //igual se extendió la
longitud del carácter para el mensaje que se pide, pero
este es específicamente para el carácter que se ingresa
int 0x80 ; llamada al sistema
```

```
; Calcular la suma de los caracteres
mov al, [input]
add al, [input]
mov [sum], al ; almacenar la suma en la variable sum
```

```
; Mostrar la suma
mov eax, 4
mov ebx, 1
mov ecx, sum ; dirección de la suma
mov edx, 1 ; longitud de la suma
int 0x80
```

```
; Terminar el programa mov eax, 1 xor ebx, ebx ; código
de salida 0 int 0x80
```

2. **Utiliza el compilador de ensamblador en línea para realizar el desarrollo del taller:**
<https://onecompiler.com/assembly>
3. **Investiga y desarrolla el funcionamiento de las siguientes instrucciones en ensamblador x86.**

1. mov eax, 4

Esta instrucción carga el valor 4 en el registro EAX. En Linux, el valor 4 en EAX indica que se está preparando una llamada al sistema `sys_write`. Esta llamada al sistema se utiliza para escribir datos a un archivo o al terminal (stdout).

- EAX se utiliza para almacenar el número de la llamada al sistema en Linux.
- En este caso, `sys_write` tiene el número de sistema 4.

2. mov ebx, 1

Esta instrucción carga el valor 1 en el registro EBX. En el contexto de la llamada a `sys_write`, EBX contiene el descriptor de archivo al cual se va a escribir. Un descriptor de archivo es un identificador que el sistema operativo asigna a archivos y dispositivos.

- El valor 1 en EBX representa stdout (la salida estándar, generalmente la terminal).

3. int 0x80

Esta instrucción invoca la interrupción 0x80, que le indica al kernel de Linux que ejecute la llamada al sistema que se ha preparado. En este caso, ejecuta `sys_write` para escribir datos a stdout.

- En resumen, con las instrucciones anteriores, esta interrupción escribiría algo en la salida estándar (como un mensaje o un carácter).

4. mov eax, 3

Aquí, EAX se carga con el valor 3, lo que en Linux corresponde a la llamada al sistema `sys_read`. Esta llamada al sistema se utiliza para leer datos de un archivo o de la entrada estándar (stdin).

- EAX = 3 prepara la llamada al sistema `sys_read`.

5. mov ebx, 0

Se carga el valor 0 en el registro EBX. En el contexto de `sys_read`, EBX contiene el descriptor de archivo desde donde se leerán los datos.

- El valor 0 en EBX representa stdin (la entrada estándar, generalmente el teclado).

6. mov al

Esta instrucción está incompleta, ya que falta el operando con el cual se va a mover un valor hacia el registro AL (la parte baja de EAX). Sin embargo, si tuviera un operando, esta instrucción movería el valor de dicho operando al registro AL (8 bits).

7. add al

Esta instrucción también está incompleta, ya que le falta un operando. En su forma completa, la instrucción suma el valor del operando al contenido del registro AL.

8. mov eax, 1

Se carga el valor 1 en el registro EAX, lo que prepara la llamada al sistema `sys_exit` en Linux. Esta llamada al sistema se utiliza para terminar un proceso.

- EAX = 1 es la llamada al sistema `sys_exit`.

4. Cambia los valores que se están asignando a los registros `eax`, `ebx`, `int 0x80`. Desarrolla qué pasa cuando cambias estos valores y por qué.

```
12 ; Mostrar mensaje en consola
13 mov eax, 4
14 mov ebx, 1
15 mov ecx, msg ; dirección del mensaje
16 mov edx, 30 ; longitud del mensaje
17 int 0x80
18
19 ; Leer un carácter desde el teclado
20
21 mov eax, 3
22 mov ebx, 0
23 mov ecx, input ; dirección para almacenar la entrada
24 mov edx, 80 ; leer 1 byte (1 carácter)
25 int 0x80
26
27 ; Mostrar el carácter ingresado
28 mov eax, 4 ; syscall número 4 es write (sys_write)
29 mov ebx, 1 ; descriptor de archivo 1 es stdout
30 mov ecx, input ; dirección del carácter
31 mov edx, 20 ; longitud del carácter
32 int 0x80 ; llamada al sistema
```

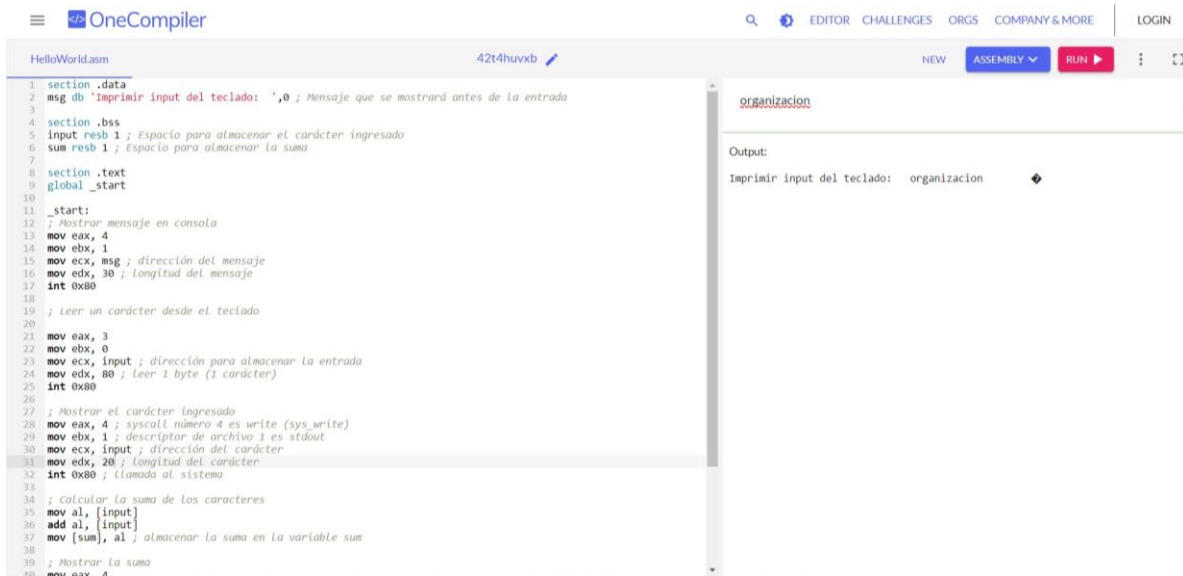
- * si no se cambia la longitud del mensaje no se mostrará completo y saldrá cortado
- * si no se cambia la capacidad de bytes que puede leer será incompleto si es que excede la magnitud de bytes
- * si no se cambia la longitud del carácter, aquí afecta el mensaje que se pone por el usuario y si no le ponemos la capacidad suficiente saldrá cortado

5. Cambia el mensaje de salida 'Input: ' 'Imprimir input del teclado: '. Documenta que pasa al imprimir el resultado. Modifica el programa para que imprima la nueva cadena en su totalidad y explica tus cambios.

```
1 section .data
2 msg db 'Imprimir input del teclado: ',0 ; Mensaje que se mostrará antes de la entrada
3
```

al cambiar el 'input: ' por 'Imprimir input del teclado: ', se muestra tal cual el mensaje en la consola seguido de la cadena de caracter dada por el usuario

6. Modifica el programa para que imprima la siguiente cadena de texto y documenta en tu práctica de taller: Imprimir input del teclado: organización. Para esto debes ingresar un valor de input en el STDIN



The screenshot shows the OneCompiler IDE interface. The left pane displays the assembly code for 'HelloWorld.asm'. The code includes sections for data, bss, and text, with various instructions for printing a message, reading input, and calculating a sum. The right pane shows the output of the program, which is 'Imprimir input del teclado: organización'.

```
1 section .data
2 msg db 'Imprimir input del teclado: ',0 ; Mensaje que se mostrará antes de la entrada
3
4 section .bss
5 input resb 1 ; Espacio para almacenar el carácter ingresado
6 sum resb 1 ; Espacio para almacenar la suma
7
8 section .text
9 global _start
10
11 _start:
12 ; Mostrar mensaje en consola
13 mov eax, 4
14 mov ebx, 1
15 mov ecx, msg ; dirección del mensaje
16 mov edx, 30 ; longitud del mensaje
17 int 0x80
18
19 ; Leer un carácter desde el teclado
20
21 mov eax, 3
22 mov ebx, 0
23 mov ecx, input ; dirección para almacenar la entrada
24 mov edx, 80 ; Leer 1 byte (1 carácter)
25 int 0x80
26
27 ; Mostrar el carácter ingresado
28 mov eax, 4 ; syscall número 4 es write (sys_write)
29 mov ebx, 1 ; descriptor de archivo 1 es stdout
30 mov ecx, input ; dirección del carácter
31 mov edx, 20 ; longitud del carácter
32 int 0x80 ; llamada al sistema
33
34 ; calcular la suma de los caracteres
35 mov al, [input]
36 add al, [input]
37 mov [sum], al ; almacenar la suma en la variable sum
38
39 ; Mostrar la suma
40 mov eax, 4
```

Output:

Imprimir input del teclado: organización