UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de Ingeniería, Arquitectura y Diseño

Ingeniero en Software y Tecnologías Emergentes

Manual de Talleres de la materia Organización de Computadoras

AUTOR(ES)
Jonatan Crespo Ragland

Taller No. 6 Generación de un programa

ejecutable

Objetivo: Desarrollar los códigos de operación y proceso de ensamblaje

Fundamentos teóricos del taller

- Lenguaje Ensamblador x86

Instrucciones para el desarrollo del taller

1. Desarrolla los siguientes puntos

Recursos

1. Apuntes de clase.
2. Fuentes bibliográficas.
3. Lápiz.
4. Equipo de cómputo.

Tiempo e instrucciones de entrega

Duración: 2 horas.

Desarrollar lo siguiente en su cuaderno o computadora: 1. Define las características principales de una máquina multinivel.

- Está compuesta por capas jerárquicas (hardware, microarquitectura, sistema operativo, lenguaje de alto nivel, aplicación).

- Cada nivel oculta la complejidad del inferior y ofrece abstracciones más fáciles al programador.
- Permite independencia tecnológica, ya que los programas de alto nivel

no dependen directamente del hardware.

● Mejora la portabilidad y escalabilidad de los programas. ● Usa

interfaces y traductores (compiladores, intérpretes, drivers).

2. **Describe a detalle como es la comunicación entre cada nivel de una máquina multinivel (hardware-microarquitectura-sistema operativo-lenguaje de alto nivel-aplicación). Puedes incluir interfaces, traductores, protocolos, capas, etc.** ● Hardware-Microarquitectura: El hardware ejecuta instrucciones básicas (códigos binarios). La microarquitectura traduce esas señales en operaciones reales con registros, memoria y buses.

   ● Microarquitectura-Sistema Operativo: El sistema operativo usa controladores y llamadas al sistema (syscalls) para comunicarse con el hardware.

   ● Sistema Operativo-Lenguaje de alto nivel: Los compiladores e intérpretes traducen instrucciones de C, Java, Python, etc., a instrucciones de máquina comprensibles por el hardware.

   ● Lenguaje de alto nivel-Aplicación: El desarrollador escribe la lógica del programa en un lenguaje entendible; el compilador y SO gestionan la ejecución, entrada/salida y recursos.

   ● Interfaces y protocolos: Incluyen compiladores, ensambladores, librerías, drivers y APIs.

3. **Características del lenguaje de bajo nivel.**
   ● Cercano al hardware.

● Usa instrucciones específicas de la CPU (ej. MOV, ADD, JMP).

● Alta velocidad de ejecución, pero difícil de programar y mantener.

● Requiere conocimiento de arquitectura (registros, memoria).

4. Características del lenguaje de alto nivel. ● Fácil de leer y escribir (sintaxis similar al lenguaje humano).

● Independiente del hardware.

● Portabilidad entre distintos sistemas.

● Uso de estructuras avanzadas (funciones, objetos, librerías).

● Menos eficiente en velocidad que el ensamblador, pero más productivo.

5. Investiga el funcionamiento de las siguientes salidas del sistema (Syscalls) en arquitectura x86: a. Sys_exit(1)

sys_exit (1): Termina el programa y regresa un valor de salida al sistema operativo.

● Registro eax = 1 → código de la syscall.

● Registro ebx = valor → código de salida.

sys_write (4): Escribe datos en un archivo o en pantalla.

● eax = 4, ebx = file descriptor (1=stdout), ecx = dirección del buffer, edx = tamaño.

sys_read (3): Lee datos desde teclado o archivo.

eax = 3, ebx = file descriptor (0=stdin), ecx = dirección del buffer,
edx = tamaño máximo.

b. Sys_write(4)

sys_write (4): Escribe datos en un archivo o en pantalla.

- eax = 4, ebx = file descriptor (1=stdout), ecx = dirección del buffer,
  edx = tamaño.

c. Sys_read(3)

sys_read (3): Lee datos desde teclado o archivo.

eax = 3, ebx = file descriptor (0=stdin), ecx = dirección del buffer,
edx = tamaño máximo.

6. De acuerdo al código ensamblador anexo (también lo puedes encontrar aqui: https://onecompiler.com/assembly/43xsbvujq , realiza lo siguiente:

a. Modifica el código para que imprima los siguientes caracteres utilizando solo sumas: i. A

Holamundo.asm 43xt4agp3

```
sección .datos
    Núm1 DB 16 ; Primera variable (entre 1 y 3)
    Núm2 db 1 ; Segunda variable (entre 1 y 3)
    resultado db 0 ; Espacio para almacenar el resultado convertido a ASCII

sección .Mensaje de texto
    global _empezar

_empezar:
    mov al, [num1] ; Cargar num1 en AL
    añadir al, [num2] ; Sumar num2 a AL
    agregue al, '0' ; Convertir el resultado a ASCII

    mov [resultado], al ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4 ; syscall: sys_write
    mov ebx, 1 ; descriptor de archivo: stdout
    mov ecx, resultado; Dirección del resultado
    mov edx, 1 ; Longitud del resultado
    int 0x80 ; Llamada al sistema

    ; Salir del programa
    mov eax, 1 ; syscall: sys_exit
    xor ebx, ebx ; Código de salida 0
    int 0x80 ; Llamada al sistema
```

STDIN

Entrada

Salida:

A

ii. :

iii. =

iv. ? v. _

Holamundo.asm 43xt4agp3 IA NUEVO ENSAMBLAJE

```
sección .datos
    Núm. 1 db 60 ; Primera variable (entre 1 y 3,
    Núm2 db 1 ; Segunda variable (entre 1 y 3)
    resultado db 0 ; Espacio para almacenar el re

sección .Mensaje de texto
    global _empezar

_empezar:
    mov al, [num1] ; Cargar num1 en AL
    añadir al, [num2] ; Sumar num2 a AL

    mov [resultado], al ; Guardar el carácter AS

    ; Imprimir el número (un solo dígito)
    mov eax, 4 ; syscall: sys_write
    mov ebx, 1 ; descriptor de archivo: stdout
    mov ecx, resultado; Dirección del resultado
    mov edx, 1 ; Longitud del resultado
    int 0x80 ; Llamada al sistema

    ; Salir del programa
    mov eax, 1 ; syscall: sys_exit
    xor ebx, ebx ; Código de salida 0
    int 0x80 ; Llamada al sistema
```

STDIN

Entrada p

Salida:

=

Holamundo.asm 43xt4agp3 IA NUEVO ENSAMBLAJE

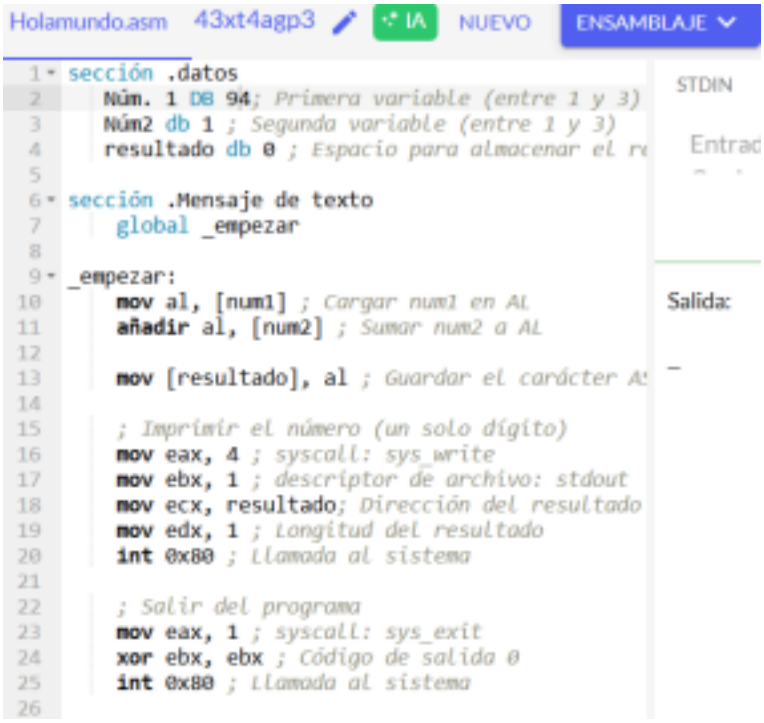
```
sección .datos
    Núm. 1 DB 62; Primera variable (entre 1 y 3)
    Núm2 db 1 ; Segunda variable (entre 1 y 3)
    resultado db 0 ; Espacio para almacenar el re

sección .Mensaje de texto
    global _empezar

_empezar:
    mov al, [num1] ; Cargar num1 en AL
    añadir al, [num2] ; Sumar num2 a AL

    mov [resultado], al ; Guardar el carácter AS

    ; Imprimir el número (un solo dígito)
    mov eax, 4 ; syscall: sys_write
    mov ebx, 1 ; descriptor de archivo: stdout
    mov ecx, resultado; Dirección del resultado
    mov edx, 1 ; Longitud del resultado
    int 0x80 ; Llamada al sistema

    ; Salir del programa
    mov eax, 1 ; syscall: sys_exit
    xor ebx, ebx ; Código de salida 0
    int 0x80 ; Llamada al sistema
```

STDIN

Entrad

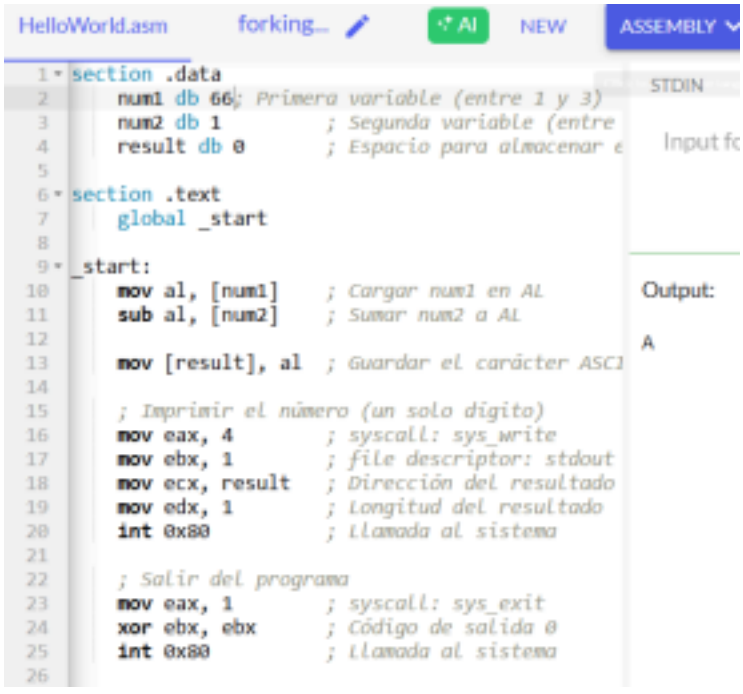
Salida:

?



b.Ahora modificarlo para imprima los siguientes caracteres utilizando al menos una resta dentro del código:

i. B



ii. x

HelloWorld.asm forking... AI NEW ASSEMBLY

```
section .data
    num1 db 121; Primera variable (entre 1 y 3)
    num2 db 1          ; Segunda variable (entre
    result db 0        ; Espacio para almacenar e

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    sub al, [num2]     ; Sumar num2 a AL

    mov [result], al   ; Guardar el carácter ASCI

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

STDIN

Input for

Output:

x

iii. + iv. ‘

HelloWorld.asm forking... AI NEW ASSEMBLY

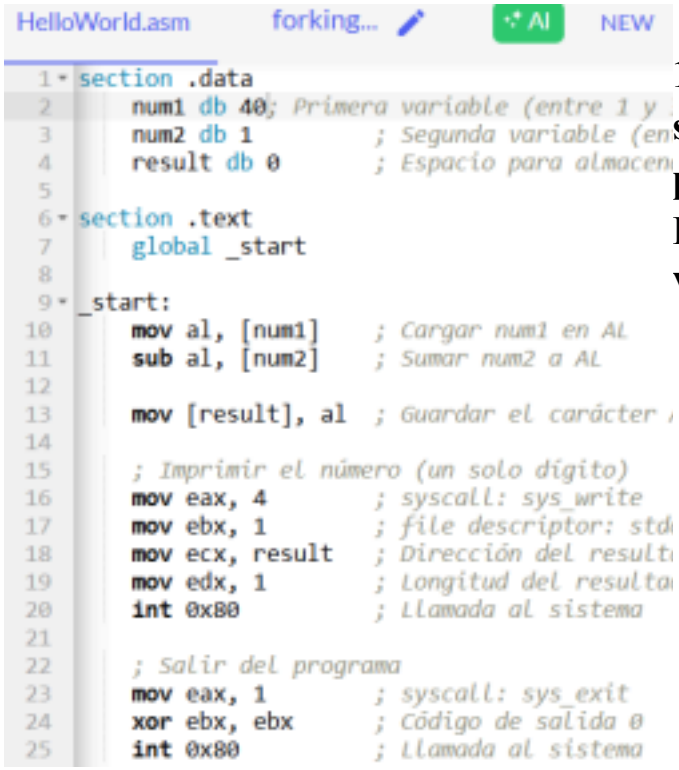
```
section .data
    num1 db 44; Primera variable (entre 1 y 3)
    num2 db 1          ; Segunda variable (entre
    result db 0        ; Espacio para almacenar e

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    sub al, [num2]     ; Sumar num2 a AL

    mov [result], al   ; Guardar el carácter ASCI

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

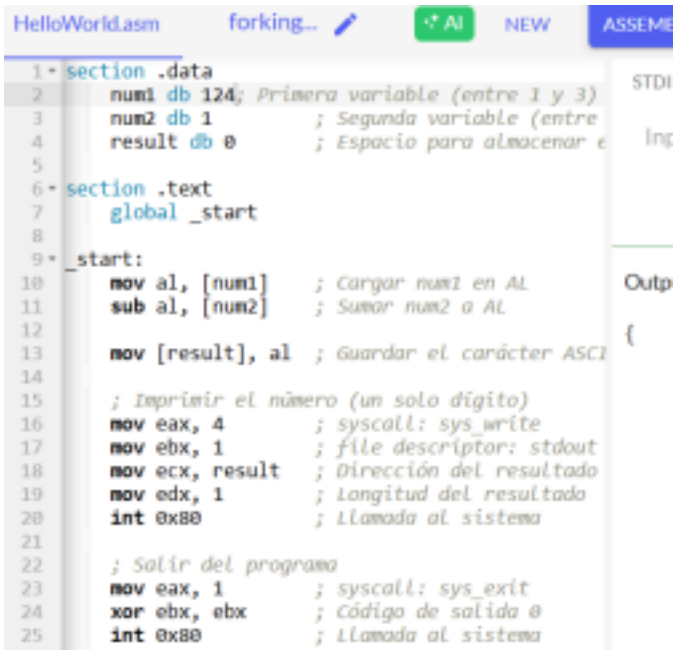
STDIN

Input f

Output:

+





1. Para el repositorio de código es suficiente documentar solo un carácter para cada inciso.

Referencias

v. {

1. Ornare quam viverra orci sagittis eu volutpat. Aenean et tortor at risus. Feugiat in ante metus dictum at tempor commodo.

2. Senectus et netus et malesuada fames ac. Dictum sit amet justo donec enim diam vulputate ut pharetra. Tristique senectus et netus et malesuada fames

Anexos