Práctica 5

# Objetivo

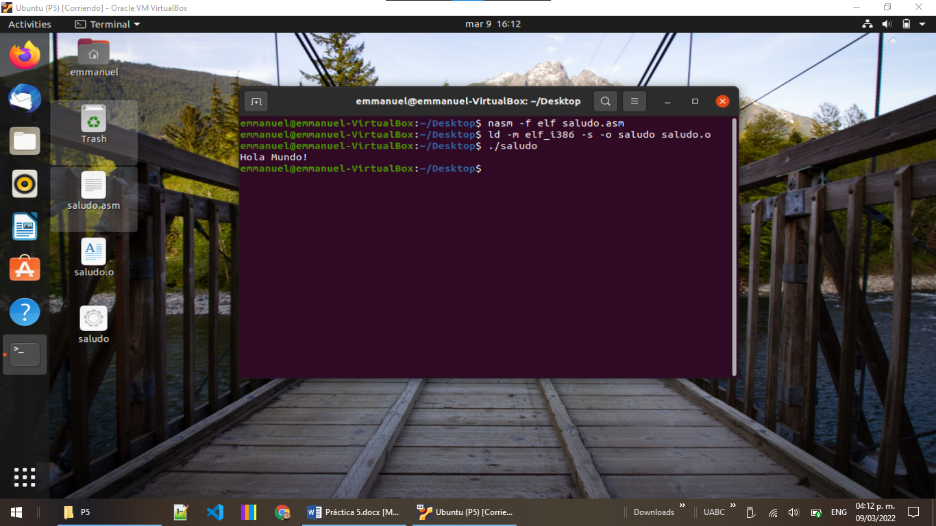
Identificar los modos de direccionamiento adecuados para manejo de memoria en aplicaciones de sistemas basados en microprocesador mediante la distinción de su funcionamiento, de forma lógica y responsable.

# Desarrollo

1. Responda los siguientes cuestionamientos sobre los programas en lenguaje ensamblador del 80386 usando NASM.
   1. ¿Qué es la sección? data? **Es la sección utilizada para inicializar datos o constantes que no cambian en el tiempo de ejecución del programa.**
   2. ¿Qué es la sección .bss? **Es la sección utilizada para declarar variables.**
   3. ¿Qué es la sección .text? **En esta sección se encuentra almacenado el código, la declaración debe ser inicializada con “global\_start”, lo cual le indica al kernel donde comienza el programa.**
   4. ¿Qué es la directiva global? **Es la que le indica que una etiqueta puede ser compartida con otros archivos además de aquel en el que se encuentra definida.**
2. Copie el código del Listado 1 en un archivo llamado saludo.asm. Abra una terminal en Linux y ensamble el código con NASM por medio del comando:

nasm -f elf saludo.asm

El cual generará el archivo objeto saludo.o.



Encadene el archivo por medio de uno de los siguientes comandos:

* 1. En un sistema operativo de 32 bits:

ld -s -o saludo saludo.o

* 1. En un sistema operativo de 64 bits:

ld -m elf\_i386 -s -o saludo saludo.o

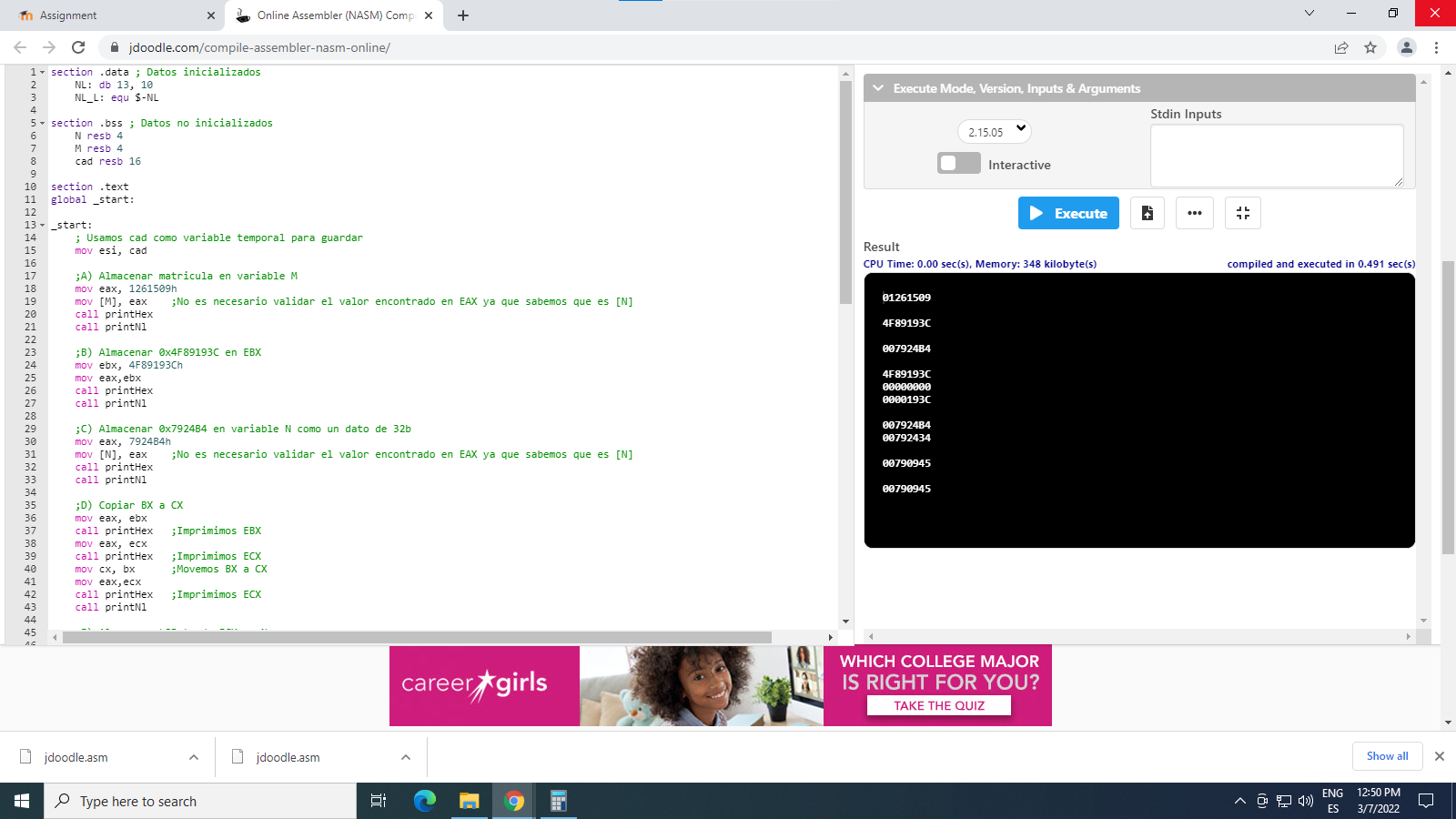
El cual generará el archivo ejecutable saludo. Ejecute el archivo por medio del comando:

./saludo

El programa desplegará en pantalla el mensaje “Hola mundo!”.

1. Copie el código del Listado 2 en un archivo llamado P5.asm. Complete el código agregando las instrucciones necesarias para hacer los siguientes movimientos de datos:
   1. Almacenar en la variable M su matrícula (use el valor como si fuera hexadecimal).
   2. Almacenar 0x4F89193C en EBX.
   3. Almacenar 0x7924B4 en la variable N como un dato de 32 bits.
   4. Copiar BX a CX.
   5. Almacenar el byte menos significativo de ECX en N.
   6. Almacenar 0x945 en N como un dato de 16 bits.
   7. Copiar a ESI la variable N.

Por cada inciso, despliegue en pantalla el nuevo valor del registro o variable modificada. Haga uso de la rutina printHex, la cual recibe en EAX el valor que se quiere imprimir.



CODIGO

section .data ; Datos inicializados

    NL: db 13, 10

    NL\_L: equ $-NL

section .bss ; Datos no inicializados

    N resb 4

    M resb 4

    cad resb 16

section .text

global \_start:

\_start:

    ; Usamos cad como variable temporal para guardar

    mov esi, cad

    ;A) Almacenar matricula en variable M

    mov eax, 1261509h   ;Movemos la matricula al registro EAX y del registro a la variable M

    mov [M], eax    ;No es necesario validar el valor encontrado en EAX ya que sabemos que es [N]

    call printHex

    call printNl

    ;B) Almacenar 0x4F89193C en EBX

    mov ebx, 4F89193Ch

    mov eax,ebx

    call printHex

    call printNl

    ;C) Almacenar 0x7924B4 en variable N como un dato de 32b

    mov eax, 7924B4h

    mov [N], eax    ;No es necesario validar el valor encontrado en EAX ya que sabemos que es [N]

    call printHex

    call printNl

    ;D) Copiar BX a CX

    mov eax, ebx

    call printHex   ;Imprimimos EBX

    mov eax, ecx

    call printHex   ;Imprimimos ECX

    mov cx, bx      ;Movemos BX a CX

    mov eax,ecx

    call printHex   ;Imprimimos ECX

    call printNl

    ;E) Almacenar LSByte de ECX en N

    mov eax, [N]

    call printHex   ;Imprimimos N antes de modificar

    and al,cl       ;Aplicamos la mascara para modificar el byte menos significativo

    mov [N], eax    ;Guardamos el valor modificado en N

    call printHex   ;Imprimimos N despues de modificar

    call printNl

    ;F) Almacenar 0x945 en N como un dato de 16b

    mov eax, 945h   ;Guardamos el valor utilizando eax para que sea tratado como dato de 32 bits, limpiando la basura de eax

    mov [N], ax     ;Guardamos el valor utilizando  ax para que sea tratado como dato de 16 bits

    mov eax, [N]    ;Obtenemos el valor completo de N (32 bits)

    call printHex

    call printNl

    ;Copiar a ESI la variable N

    mov esi, N

    call printHex

    call printNl

    mov eax, 1  ;Acaba el programa

    mov ebx,0

    int 80h

printHex:           ;Imprime un valor hexadecimal

    pushad          ;Guardamos todos los registros en la pila

    mov edx, eax    ;Copiamos el valor a imprimir en el registro edx

    mov ebx, 0fh

    mov cl, 28

.nxt: shr eax,cl    ;Empezamos de izq a derecha

.msk: and eax,ebx   ;Aplicamos la mascara para obtener el byte mas significativo de la palabra menos significativa  (0000 0000 0000 <XX00> )

    cmp al, 9       ;Si es menor que 9

    jbe .menor      ;salta menor

    add al,7        ;Si es mayor, se convierte en su valor hex (A-F)

.menor:add al,'0'   ;   Se convierte en su valor ascii

    mov byte [esi],al;Guardamos el byte en la cadena temporal

    inc esi         ;Apuntamos en la sig localidad de memoria

    mov eax, edx    ;Obtenemos el valor EAX original

    cmp cl, 0       ;Cuando lleguemos al final

    je .print       ;   Imprimimos la cadena temporal

    sub cl, 4       ;Si no, buscamos el sig byte

    cmp cl, 0       ;Si aun no llegamos al byte final

    ja .nxt         ;   Buscamos el siguiente byte

    je .msk         ;Hacemos la mascara directamente

.print: mov eax, 4  ;Seleccionamos servicio

    mov ebx, 1      ;Seleccionamos unidad de salida

    sub esi, 8      ;Seleccionamos el primer caracter

    mov ecx, esi

    mov edx, 8      ;Caracteres a imprimir

    int 80h         ;Interrupcion para imprimir

    call printNl    ;Imprimir el salto de linea

    popad

    ret

printNl:            ;Imprime un salto de linea

    pushad

    mov eax, 4

    mov ebx, 1

    mov ecx, NL

    mov edx, NL\_L

    int 80h

    popad

    ret

# Conclusiones y comentarios

Es interesante ver como es necesario especificar al ensamblador en que parte comienza determinada sección e indicar de que sección se está especificando. Además del uso de las directivas que ayudan a hacer el trabajo más sencillo para el ensamblador.

# Dificultades en el desarrollo

Mis complicaciones se dieron al entender las interrupciones y servicios para poder imprimir caracteres (o cadenas) ya que este lenguaje ensamblador cuenta con pequeñas diferencias al ensamblador utilizado anteriormente en otros cursos.

## **References**

2.3: Assembler Directives. (2022). Retrieved 14 March 2022, from https://eng.libretexts.org/Bookshelves/Electrical\_Engineering/Electronics/Implementing\_a\_One\_Address\_CPU\_in\_Logisim\_(Kann)/02%3A\_Assembly\_Language/2.03%3A\_Assembler\_Directives