Tema: Introducción al lenguaje de ensamble y al uso de herramientas de desensamble
Habilidad Personal: Habilidad para implementar *software*.

Program Outcomes:
CDIO:
4.5. IMPLEMENTING
4.5.3. Software Implementing Process
ABET:
(2.) Develop an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.

INSTRUCCIÓN Nº 1:

Por favor genere el código objeto de la siguiente función minimalista en Lenguaje C. Ud. debe editarla y salvarla en un archivo de texto plano con el nombre "f.c". Si no recuerda o incluso si aún no sabe cómo generar el código objeto, acuda al **TIP Nº 1**, al final de este documento. Si tiene inquietudes, puede consultar a sus monitores.

```
_(){}
```

INSTRUCCIÓN Nº 2:

Usando el programa "brw.exe", cuyo código fuente fue compartido previamente, por favor identifique la primera y última instrucción en código máquina y determine el número total de bytes en bajo nivel generado para un procesador INTEL x86. Si no recuerda o incluso si aún no sabe usar el programa "brw.exe", acuda al **TIP Nº 2**, al final de este documento. Si tiene inquietudes, puede consultar a sus monitores.

INSTRUCCIÓN Nº 3:

Por favor traduzca el código fuente "f.c" a lenguaje de ensamble e indague las instrucciones que se ejecutan. Investigue la arquitectura de un procesador INTEL x86 e indague la extensión y uso de los

registros de las instrucciones y explique qué hace el programa. Si no recuerda o incluso si aún no sabe cómo compilar a lenguaje de ensamble, acuda al **TIP Nº 3**, al final de este documento.

INSTRUCCIÓN Nº 4:

Ejecute el siguiente comando y explique qué hace:

```
objdump -d f.o
```

Cambie el parámetro "-D" por "-d", "-S", "-s". Explore otras opciones, viendo la ayuda cuando invoca al comando sin parámetro alguno.

INSTRUCCIÓN Nº 5:

Modifique el programa original, re-escribiéndolo de la siguiente manera. Genere el código objeto y luego ejecute el comando "objdump" de la INSTRUCCIÓN Nº 4. Explique lo que acaba de hacer.

```
_(){
asm( "NOP" );
asm( "NOP" );
}
```

INSTRUCCIÓN Nº 6:

Para el siguiente programa, genere el código objeto y luego ejecute el comando "objdump" de la INSTRUCCIÓN Nº 4. Observe y explique el resultado.

```
_(){
asm( "NOP" );

asm( "MOV %ESP,%EAX" );
asm( "MOV %ESP,%ECX" );
```

```
asm( "MOV %ESP,%EDX" );
asm( "MOV %ESP,%EBX" );

asm( "MOV %ESP,%ESP" );
asm( "MOV %ESP,%EBP" );
asm( "MOV %ESP,%ESI" );
asm( "MOV %ESP,%EDI" );

asm( "MOV %EBP,%EAX" );
asm( "MOV %EBP,%ECX" );
asm( "MOV %EBP,%EDX" );
asm( "MOV %EBP,%EBX" );

asm( "MOV %EBP,%ESP" );
asm( "MOV %EBP,%EBP" );
asm( "MOV %EBP,%ESI" );
asm( "MOV %EBP,%EDI" );

asm( "MOV %ESI,%EAX" );
asm( "MOV %ESI,%ECX" );
asm( "MOV %ESI,%EDX" );
asm( "MOV %ESI,%EBX" );

asm( "MOV %ESI,%ESP" );
asm( "MOV %ESI,%EBP" );
asm( "MOV %ESI,%ESI" );
asm( "MOV %ESI,%EDI" );

asm( "MOV %EDI,%EAX" );
asm( "MOV %EDI,%ECX" );
asm( "MOV %EDI,%EDX" );
asm( "MOV %EDI,%EBX" );

asm( "MOV %EDI,%ESP" );
asm( "MOV %EDI,%EBP" );
asm( "MOV %EDI,%ESI" );
asm( "MOV %EDI,%EDI" );

asm( "PUSH %AX" );
asm( "PUSH %CX" );
asm( "PUSH %DX" );
asm( "PUSH %BX" );
```

```
asm( "PUSH %SP" );
asm( "PUSH %BP" );
asm( "PUSH %SI" );
asm( "PUSH %DI" );

asm( "POP %AX" );
asm( "POP %CX" );
asm( "POP %DX" );
asm( "POP %BX" );

asm( "POP %SP" );
asm( "POP %BP" );
asm( "POP %SI" );
asm( "POP %DI" );

asm( "PUSH %EAX" );
asm( "PUSH %ECX" );
asm( "PUSH %EDX" );
asm( "PUSH %EBX" );

asm( "PUSH %ESP" );
asm( "PUSH %EBP" );
asm( "PUSH %ESI" );
asm( "PUSH %EDI" );

asm( "POP %EAX" );
asm( "POP %ECX" );
asm( "POP %EDX" );
asm( "POP %EBX" );

asm( "POP %ESP" );
asm( "POP %EBP" );
asm( "POP %ESI" );
asm( "POP %EDI" );

asm( "NOP" );

}
```

INSTRUCCIÓN Nº 7:

Use en el campo apropiado el número 55, especificando cuando sea requerido la arquitectura x86 en cada uno de los siguientes enlaces WEB:

https://defuse.ca/online-x86-assembler.htm#disassembly2

https://onlinedisassembler.com/odaweb/

http://shell-storm.org/online/Online-Assembler-and-Disassembler/

https://disasm.pro

INSTRUCCIÓN Nº 8:

Investigue otras formas alternativas para desensamblar un código objeto para la arquitectura INTEL x86.

INSTRUCCIÓN Nº 9:

¿Cómo abordaría Ud. el desarrollo de un desensamblador minimalista para un procesador muy básico que maneje tan solo unas 10 instrucciones máquina?

TIP Nº 1:

El código objeto se genera con el parámetro "-c" especificando como entrada el archivo fuente "f.c" y el archivo de salida con el parámetro "-o" y el nombre de archivo objeto "f.o"

En consola de Windows	En consola de Linux
`gcc -c f.c -o f.o`	`gcc -c f.c -o f.o`

TIP Nº 2:

El programa "brw" se ejecuta especificando el archivo binario de entrada "f.o", y la salida por defecto será stdout que corresponde a la pantalla:

En consola de Windows	En consola de Linux
`brw f.o`	`./brw f.o`

Otra forma alternativa para lograr lo mismo, es ejecutando lo siguiente:

En consola de Windows	En consola de Linux
`brw < f.o`	`./brw < f.o`

La primera instrucción del código máquina a la función "_()" empieza por 55 y termina por C3. Puede identificar la ocurrencia de dichos códigos, redireccionando la salida estándar al comando "find" en Windows o "grep" en Linux así:

En consola de Windows	En consola de Linux
`brw f.o | find "55"` `brw f.o | find "C3"`	`./brw f.o | grep "55"` `./brw f.o | grep "C3"`

TIP Nº 3:

La generación del código en lenguaje de ensamble se hace con el parámetro "-S" y el código generado se puede visualizar con el comando "type" en Windows o "cat" en Linux:

En consola de Windows	En consola de Linux
`gcc -S f.c -o f.s` `type f.s`	`gcc -S f.c -o f.s` `cat f.s`