

Instituto Tecnológico de Costa Rica

Arquitectura de computadoras

Tarea de Decoder

Profesor: Daniel Kohkemper



Estudiante: Luis Alfaro Alfaro
2019057649

Periodo: IS-2020

1. Realice un código en c que sume dos enteros:

```
int main(void) {  
    int x=1;  
    int y=2;  
    int R=x+y;  
    return 0;  
}
```

2. Compile el código fuente:

	main	12/3/2020 14:33	Archivo	8 KB
	main	12/3/2020 14:33	Archivo C	1 KB

3. Utilice IDA PRO para abrir el binario generado:

```
Load file C:\Users\Luis Alfaro\Desktop\Tarea Decoder\main as  
ELF64 for x86-64 (Executable) [elf64.dll]  
Binary file  
  
seg000:0000000000000000 ;  
seg000:0000000000000000 ; +-----+  
seg000:0000000000000000 ; | This file has been generated by The Interactive Disassembler (IDA) |  
seg000:0000000000000000 ; | Copyright (c) 2018 Hex-Rays, <support@hex-rays.com> |  
seg000:0000000000000000 ; | Freeware version |  
seg000:0000000000000000 ; +-----+  
seg000:0000000000000000 ;  
seg000:0000000000000000 ; Input SHA256 : C1627F9802BF1F2E46883D558948B71FD5B8DC8FEA3013CE0A90F941E1DD78BB  
seg000:0000000000000000 ; Input MD5 : 31DD1AFD07DEB97F83074934713CD7C7  
seg000:0000000000000000 ; Input CRC32 : 479998D5  
seg000:0000000000000000 ;  
seg000:0000000000000000 ; File Name : C:\Users\Luis Alfaro\Desktop\Tarea Decoder\main  
seg000:0000000000000000 ; Format : Binary file  
seg000:0000000000000000 ; Base Address: 0000h Range: 0000h - 1F80h Loaded length: 1F80h  
seg000:0000000000000000 ;  
seg000:0000000000000000 .686p  
seg000:0000000000000000 .mmx  
seg000:0000000000000000 .model flat  
seg000:0000000000000000 ;  
seg000:0000000000000000 ; =====  
seg000:0000000000000000 ; Segment type: Regular  
seg000:0000000000000000 seg000 segment byte public '' use64  
seg000:0000000000000000 assume cs:seg000  
seg000:0000000000000000 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing  
seg000:0000000000000000 db 7Fh ;  
seg000:0000000000000001 db 45h ; E  
seg000:0000000000000002 db 4Ch ; L  
seg000:0000000000000003 db 46h ; F  
seg000:0000000000000004 db 2  
seg000:0000000000000005 db 1  
seg000:0000000000000006 db 1  
seg000:0000000000000007 db 0  
seg000:0000000000000008 db 0  
seg000:0000000000000009 db 0
```

Aquí no entendí muy bien lo que estaba haciendo el decoder así que lo abrí con el modo desensamblador

4. Revise si puede entender lo que hace su código observando las instrucciones en ensamblador:

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

push    rbp
mov     rbp, rsp
xor     eax, eax
mov     [rbp+var_4], 0
mov     [rbp+var_8], 1
mov     [rbp+var_C], 2
mov     ecx, [rbp+var_8]
add     ecx, 2
mov     [rbp+var_10], ecx
pop     rbp
retn
main endp
```














Aquí el código se ve más legible y en este caso el programa usa la pila para almacenar las variables locales. Las inicializaciones de las variables no son los enteros a sumar sino son un puntero la posición que van a tener en la pila

El push a rbp guarda la posición actual del puntero a la base y al mover el rsp a rbp se crea una nueva base en la cima de la pila, esto para empezar a guardar las variables.

A ecx se le pasa el 1 y a ese uno se le hace un add 2 de direccionamiento inmediato y ecx mueve a la variable que contiene el resultado y el pop lo que hace es resetear la pila.

5. Revise que opciones le da IDA Pro para entender mejor el código:

En este caso IDA nos permite ver función y biblioteca de c que se usó en el ensamblado.

	<code>__do_global_dtors_aux</code>	<code>.text</code>
	<code>__gmon_start__</code>	<code>extern</code>
	<code>__libc_csu_fini</code>	<code>.text</code>
	<code>__libc_csu_init</code>	<code>.text</code>
	<code>__libc_start_main</code>	<code>extern</code>
	<code>_dl_relocate_static_pie</code>	<code>.text</code>
	<code>_init_proc</code>	<code>.init</code>
	<code>_start</code>	<code>.text</code>
	<code>_term_proc</code>	<code>.fini</code>
	<code>deregister_tm_clones</code>	<code>.text</code>
	<code>frame_dummy</code>	<code>.text</code>
	<code>main</code>	<code>.text</code>
	<code>register_tm_clones</code>	<code>.text</code>