Lab 1

CSC472-01

Michael Burns

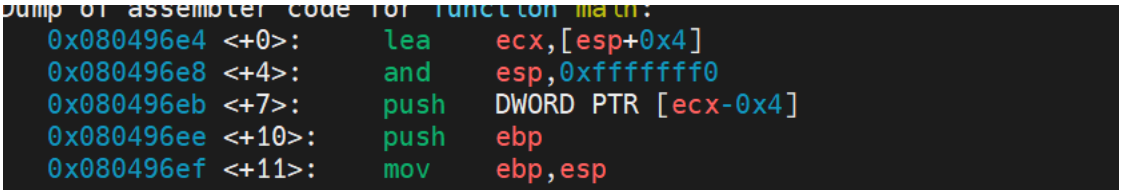
September 12th, 2023

Introduction

In this lab, I am using the GDB debugger to examine the x86 ASM register instructions for executing basic programs in C. Understanding how different types and categories of registers, and how data is manipulated on a stack, is essential. This will allow a cyber security analyst to see how a program encodes code and data, and finding data.

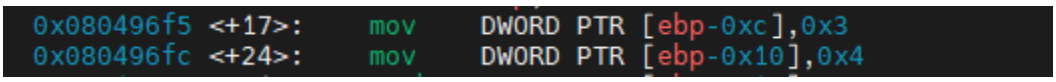
Analysis and Results

Q1: Identify the assembly instructions for creating the stack frame of the main() function.



The assembly instructions for creating the stack are push ebp (saving the base pointer to the stack) and mov ebp, esp (move the pointer from base pointer to stack pointer.

Q2. Identify and explain the purpose of the two lines related to setting variables p and q.



The first line, mov DWORD PTR [ebp-0xc], 0x3 is assigning the value 3 to the memory location pointed to by [ebp-0xc]. The second line, mov DWORD PTR [ebp-0x10], 0x4 is assigning the value 4 to the memory location pointed to by [ebp-0x10]. DWORD PTR indicates that a Double Word pointer (a 32-bit value) is

being used to hold the value 0x3 and 0x4 (which are variables p and q). (GalaxyMentor Tutor was able to assist with this problem).

Q3: Before calling multiply_by_two(), why does the stack contain two sets of "3,4" instead of just one set?

I believe the stack contains two sets of "3,4" instead of just one set because of little and big endian byte ordering.

Q4: Explain the meaning of add eax,edx and add eax,eax. Why not using mul (Multiply) instruction instead?

The meaning of add eax,edx means "add the value stored in edx and add and store the value in the eax register". The meaning of add eax,eax means "add the value stored in eax to the value stored in eax, and stored the value in the eax". I believe that the reason the ADD instruction is used instead of MUL is that the return statement of the program is to add the values together.

Q5: Which register is used to store the final multiplication result?

```
Dump of assembler code for function multiply_by_two:
   0x080496d5 <+0>:     push   ebp
   0x080496d6 <+1>:     mov    ebp,esp
   0x080496d8 <+3>:     mov    edx,DWORD PTR [ebp+0x8]
   0x080496db <+6>:     mov    eax,DWORD PTR [ebp+0xc]
   0x080496de <+9>:     add    eax,edx
   0x080496e0 <+11>:    add    eax,eax
   0x080496e2 <+13>:    pop    ebp
   0x080496e3 <+14>:    ret
```

Eax is used to store the final result.

Discussion and Conclusion

The lab satisfied the stated purpose. I observed the x86 instructions of a simple program in C with two functions. I used the GDB debugger to obtain the assembly dump, and analyze the instructions each function uses. I learned how a stack frame is created and popped off when the return instruction is executed. I learn how variables are assigned to register and manipulated throughout a program. The learning achievements of this lab are as I expected them to be, and has furthered my knowledge of ASM.