Lab 10 Solutions

Lab 10 - Disassembly Challenge

The following is the disassembled output of a program; Can you translate the following code to its high-level equivalent?. Use the techniques and the concepts that you learned previously to solve this challenge

```
mov     dword ptr [ebp-4], 1
cmp     dword ptr [ebp-4], 0
jnz     loc_40101C
mov     eax, [ebp-4]
xor     eax, 2
mov     [ebp-4], eax
jmp     loc_401025

loc_40101C:
  mov     ecx, [ebp-4]
  xor     ecx, 3
  mov     [ebp-4], ecx

loc_401025:
```

Solution

Let's start by assigning the symbolic names to the address ***(ebp-4)***

```
mov     dword ptr [ebp-4], 1
cmp     dword ptr [ebp-4], 0
jnz     loc_40101C
mov     eax, [ebp-4]
xor     eax, 2
mov     [ebp-4], eax
jmp     loc_401025

loc_40101C:
  mov     ecx, [ebp-4]
  xor     ecx, 3
  mov     [ebp-4], ecx

loc_401025:
```

```
mov dword ptr [x], 1
cmp dword ptr [x], 0
jnz loc_40101C
mov eax, [x]
xor eax, 2
mov [x], eax
jmp loc_401025

loc_40101C:
  mov ecx, [x]
  xor ecx, 3
  mov [x], ecx

loc_401025:
```

- Notice the ***cmp*** and ***jnz*** instructions at ❶ and ❷ (this is a conditional statement) and note that ***jnz*** is the same as ***jne*** (jump if not equal to).

- Now, let's try to determine what type of conditional statement this is (***if***, or ***if/else***, or ***if/else if/else***, and so on); to do that, focus on the jumps. The conditional jump at ❷ is taken to ***loc_401010C***, and before the ***loc_40101C***, there is an unconditional jump at ❸ to ***loc_401025***.

- From what we learned previously, this has the characteristics of an ***if-else*** statement. To be precise, the code from ❹ to ❸ is part of the ***if*** block and the code from ❺ to ❻ is part of the ***else*** block.
Let's rename ***loc_40101C*** to ***else*** and ***loc_401025*** to ***end***:

```
mov dword ptr [x], 1
cmp dword ptr [x], 0 ❶
jnz loc_40101C ❷
mov eax, [x] ❹
xor eax, 2
mov [x], eax
jmp loc_401025 ❸

loc_40101C:
  mov ecx, [x] ❺
  xor ecx, 3
  mov [x], ecx ❻

loc_401025:
```

- In the assembly code, ***x*** is assigned a value of ***1*** at ❼; the value of ***x*** is compared with ***0***, and if it is equal to ***0*** (❶ and ❷), the value of ***x*** is ***xor***ed with ***2***, and the result is stored in ***x*** (❹ to ❽).

- If ***x*** is not equal to ***0***, then the value of ***x*** is ***xor***ed with ***3*** (❺ to ❻). Reading the assembly code is slightly tricky, so let's write the preceding code in a high-level language equivalent. We know that ❶ and ❷ is an ***if*** statement, and you can read it as the ***jump is taken to else if x is not equal to 0*** (***JNZ*** is an alias for ***JNE***).

- To write these statements back to a high-level language, you need to reverse the condition and ask the question when will the jump not be taken at ❷.

- The jump will not be taken when x is equal to 0, so you can write the preceding code to a pseudocode, shown as follows. Note that in the following code, the cmp and jnz instruction is translated to an if statement; also, note how the condition is reversed:

```
mov dword ptr [x], 1 ❼
cmp dword ptr [x], 0 ❶
jnz else ❷
mov eax, [x] ❹
xor eax, 2
mov [x], eax ❽
jmp end ❸

else:
  mov ecx, [x] ❺
  xor ecx, 3
  mov [x], ecx ❻

end:
```

```
x = 1
if(x == 0)
{
    eax = x
    eax = eax ^ 2
    x = eax
}
else {
    ecx = x
    ecx = ecx ^ 3
    x = ecx
}
```

- Now that we have identified the conditional statements when we replace all of the registers on the right side of the = operator (at ❾) with their corresponding values, we get the shown code:

```
x = 1
if(x == 0)
{
    eax = x
    eax = eax ^ 2 ❾
    x = eax ❾
}
else {
    ecx = x
    ecx = ecx ^ 3 ❾
    x = ecx ❾
}
```

```
x = 1
if(x == 0)
{
    eax = x
    eax = x ^ 2
    x = x ^ 2
}
else {
    ecx = x
    ecx = x ^ 3
    x = x ^ 3
}
```

- Removing all of the entries containing the registers on the left-hand side of the = operator (at ⑩), we get the shown code:

```
x = 1
if(x == 0)
{
   eax = x ⑩
   eax = x ^ 2 ⑩
   x = x ^ 2
}
else {
   ecx = x ⑩
   ecx = x ^ 3 ⑩
   x = x ^ 3
}
```

```
x = 1;
if(x == 0)
{
   x = x ^ 2;
}
else {
   x = x ^ 3;
}
```