Lab 11 Solutions

Lab 11 - 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-8], 1
 mov dword ptr [ebp-4], 0
loc_401014:
 cmp dword ptr [ebp-4], 4
 jge short loc_40102E
 mov eax, [ebp-8]
 add eax, [ebp-4]
 mov [ebp-8], eax
 mov ecx, [ebp-4]
 add ecx, 1
 mov [ebp-4], ecx
 jmp short loc_401014
loc_40102E:
```

Solution

The code consists of two memory addresses (***ebp-4*** and ***ebp-8***); let's rename ***ebp-4*** to ***x*** and ***ebp-8*** to ***y***

```
mov dword ptr [ebp-8], 1
 mov dword ptr [ebp-4], 0
loc_401014:
 cmp dword ptr [ebp-4], 4
 jge short loc_40102E
 mov eax, [ebp-8]
 add eax, [ebp-4]
 mov [ebp-8], eax
 mov ecx, [ebp-4]
 add ecx, 1
 mov [ebp-4], ecx
 jmp short loc_401014
loc_40102E:
```

```
mov dword ptr [y], 1
mov dword ptr [x], 0

loc_401014:
  cmp dword ptr [x], 4
  jge loc_40102E
  mov eax, [y]
  add eax, [x]
  mov [y], eax
  mov ecx, [x]
  add ecx, 1
  mov [x], ecx
  jmp loc_401014
loc_40102E:
```

- At ❶, there is a backward jump to ***loc_401014***, indicating a ***loop***

- At ❷ and ❸, there is a condition check for the variable ***x*** (using ***cmp*** and ***jge***); the code is checking whether ***x*** is greater than or equal to ***4***. If the condition is met, it will jump outside of the loop to ***loc_40102E*** (at ❹).

- The value of ***x*** is incremented to 1 (from ❺ to ❻), which is the update statement. Based on all of this information, it can be deduced that ***x*** is the loop variable that controls the loop.

- We can write the preceding code to a high-level language equivalent; but to do that, remember that we need to reverse the condition from ***jge*** (***jump if greater than or equal to***) to ***jump if less than***.

```
mov dword ptr [y], 1
mov dword ptr [x], 0

loc_401014:
  cmp dword ptr [x], 4 ❷
  jge loc_40102E ❸
  mov eax, [y]
  add eax, [x]
  mov [y], eax
  mov ecx, [x] ❺
  add ecx, 1
  mov [x], ecx ❻
  jmp loc_401014 ❶
loc_40102E: ❹
```

After Translation

```
mov dword ptr [y], 1
mov dword ptr [x], 0

loc_401014:
  cmp dword ptr [x], 4
  jge loc_40102E
  mov eax, [y]
  add eax, [x]
  mov [y], eax
  mov ecx, [x]
  add ecx, 1
  mov [x], ecx
  jmp loc_401014
loc_40102E:
```

```
y = 1
x = 0
while (x<4) {
    eax = y
    eax = eax + x
    y = eax
    ecx = x
    ecx = ecx + 1
    x = ecx
}
```

Replacing all of the registers on the right-hand side of the = operator (at ❼) with their previous values, we get the translated code:

```
y = 1
x = 0
while (x<4) {
eax = y
eax = eax + x ❼
y = eax ❼
ecx = x
ecx = ecx + 1 ❼
x = ecx ❼
}
```

```
y = 1
x = 0
while (x<4) {
   eax = y
   eax = y + x
   y = y + x
   ecx = x
   ecx = x + 1
   x = x + 1
}
```

Now, removing all of the entries containing registers on the left side of the = sign (at ❽), we get the following code:

```
y = 1
x = 0
while (x<4) {
   eax = y ❽
   eax = y + x ❽
   y = y + x
   ecx = x ❽
   ecx = x + 1 ❽
   x = x + 1
}
```

```
y = 1;
x = 0;
while (x<4) {
   y = y + x;
   x = x + 1;
}
```