Lab 2 08227 Advanced Programming

This tutorial introduces the reader to assembly language.

1.0 Setup

Download the Lab2.zip file and extract the files to G:/08227/Lab2/.

Create a new empty C++ project as described in the previous laboratory.

Add the **source.cpp** to the new project.

The **source.cpp** file contains the following code:

```
#include <iostream>
using namespace std;

void main(int, char**) {
     const int start = 3;
     const int end = 10;
     int total = 0;

     int count = start;
     while (count < end) {
        total += count;
        count++;
     }

     cout << "Total= " << total << endl;

     system("PAUSE");
}
```

The program executes a single loop, adding the value of the loop counter to a running total.

Compile and run the program.

2.0 Debugging

View **source.cpp** within Visual Studio.

Place a breakpoint on line:

```
const int start = 3;
```

The easiest way to place the breakpoint is to click in the margin of the main.cpp file. The break point appears as a red circle.

Make sure that Debug mode is selected on the menu bar.

Run the program.

Execution should halt at your breakpoint.

Single set through your programme by pressing F10. This executes one line of C++ code with each press.

Now open the Local variable windows by selecting Debug->Windows->Local. This shows the values of all the C++ variables currently in scope.

Hit F5 to execute the program to completion.

3.0 Getting lost

Restart your programme and execute to the same break point as before.

Last time we used F10 to execute a line of code. If the code is a function call e.g. the streaming operator, then F10 will execute the function call in a single press.

F11 is similar to F10, but it will attempt to single step through a function.

Press F11 until you enter the streaming operator.

Now you're debugging the C++ streaming library! Provided the source code is available, it is possible to debug any library in C++.

To avoid getting too lost, you can press Shift-F11 to jump back out of the most recent function.

Press shift-F11 now.

Hit F5 to execute the program to completion.

4.0 Disassemble

Restart your programme and execute to the same break point as again.

Now disassemble your code, by selecting Debug->Windows->Disassembly.

You should get:

```
        const int start = 3;
⇨ 001D5EA8  mov         dword ptr [start],3
        const int end = 10;
  001D5EAF  mov         dword ptr [end],0Ah
        int total = 0;
  001D5EB6  mov         dword ptr [total],0

        int count = start;
  001D5EBD  mov         dword ptr [count],3
        while (count < end) {
  001D5EC4  cmp         dword ptr [count],0Ah
  001D5EC8  jge         main+5Eh (01D5EDEh)
          total += count;
  001D5ECA  mov         eax,dword ptr [total]
  001D5ECD  add         eax,dword ptr [count]
  001D5ED0  mov         dword ptr [total],eax
          count++;
  001D5ED3  mov         eax,dword ptr [count]
  001D5ED6  add         eax,1
  001D5ED9  mov         dword ptr [count],eax
        }
  001D5EDC  jmp         main+44h (01D5EC4h)
```

Familiarise yourself with the display. You should see your C++ instruction above a series of assembly language instructions. Reading the assembly language from left to right you should see the memory location, operation and finally the operands. Note the memory locations will vary between executions.

Notice also that the execution point (yellow arrow) has moved from the C++ instruction to the assembly instruction

Hit F5 to execute the program to completion.

5.0 Registers

Execute your programme to the same break point as before, and open the Disassembly window.

Now open the Register window, by selecting Debug->Windows->Registers

Single step through the code by pressing F10

The execution point moves to the next assembly instruction and any registers that change are highlighted within the register window, in red. In this case the EIP (instruction pointer) and EFL (flags) have changed.

Continue to execute your code one line at a time, taking note of the change in registers.

6.0 Release mode

Now select Release mode on the menu bar

Recompile the programme.

Execute your programme to the same break point as before, and open the Disassembly window.

Notice that the breakpoint has been moved, automatically. The following line no longer has any associated assembly instructions

```
const int start = 3;
```

The optimiser has been at work!!!

Step through your code. If it still exists it has been greatly optimised. Much of the function overhead has been removed.

In release mode it is virtually impossible to associate a piece of assembly with its original C++.

Enjoy assembly language!!