University of Liverpool
Department of Computer Science

COMP124 – Computer Systems

---

## Lab Sheet 1

These tasks are not assessed, so you don't have to submit anything. However, the final exam might have some questions based on things mentioned in the lab sheets. Use the drop-in sessions to ask for help.

---

### Task 1 – Create a Simple Program

Create a new Visual Studio project from your template. If you haven't already created a template, refer back to the instructions on Canvas.

Modify the code to add a variable declaration and some simple assembly instructions. The whole program in the text editor should look like this. The assembly language block is embedded within a C program. Any variables that you need are declared outside that block, using C syntax (which looks almost identical to Java, so you should be familiar with it).

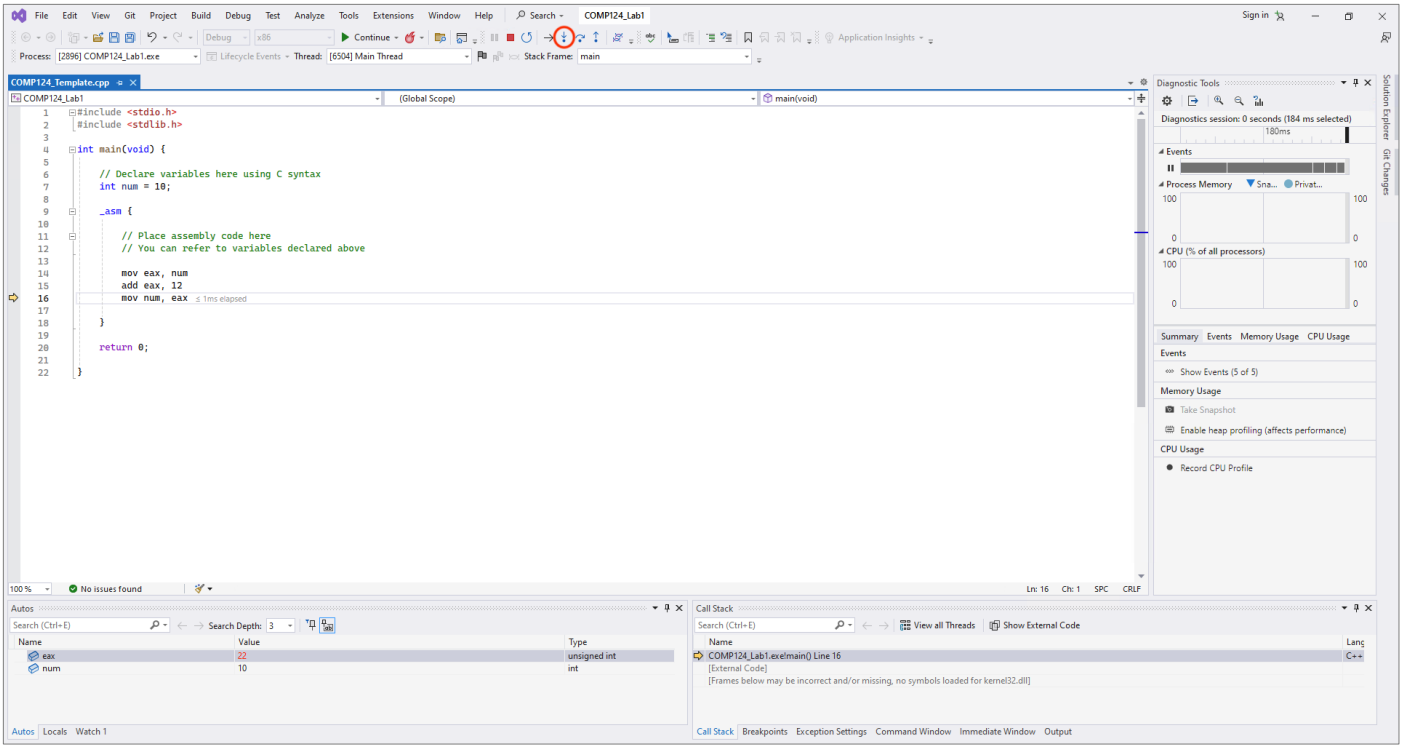
```
#include <stdio.h>
#include <stdlib.h>

int main(void) {

    int num = 10;

    _asm {

        mov eax, num
        add eax, 12
        mov num, eax

    }

    return 0;

}
```

This code should be easy to understand. We declare an integer variable called **num** and store **10** into it, using the high-level C language. Within the assembly code itself, we load the value of **num** into the accumulator register (**eax**), add **12** to it, then store the accumulator value back into the **num** variable.

### Task 2 – Compile and Debug the Program

Note that there is security monitoring software on the lab PCs that can throw an exception when you first try to debug a project. If this happens, follow the steps at the end of this sheet.

You can compile the program with the 'Build Solution' option in the 'Build' menu. To fully see what is happening within the variables and registers, select 'Step Into' from the 'Debug' menu. You can also press **F11** to start. Once you are in debug mode, use the 'Step Into' button on the toolbar (see the red circle in the screenshot below) or press **F11** to step line by line through the code.




In the bottom left of the Visual Studio layout you can see the debug area. The 'Autos' tab is populated with variables and registers as they are used. The 'Locals' tab shows the current values of any local variables, even if they are not being used.

Step through the code, line by line, and observe what is happening in the debug area. You can also experiment with using the other debug buttons, such as 'Step Over' (**F10**) and 'Step Out' (**Shift+F11**). Press the red stop button or press **Shift+F5** to stop the program (if it gets stuck in a loop or doesn't terminate normally).

If you are working with nested functions (methods) in the high-level language, the various 'Step…' buttons do slightly different things. Note that this only works with high-level functions (in C, C++, etc.) and not with any subroutines or procedures in the assembly code block.

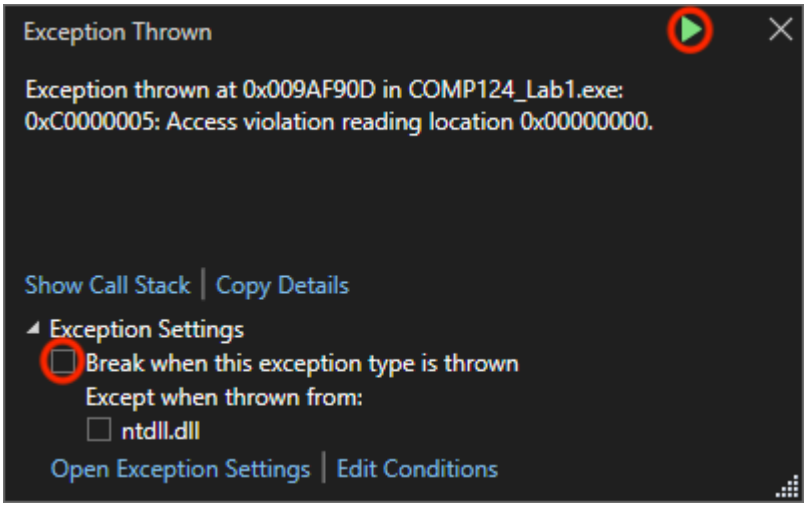
- Step Into – If the line of code is a function (method) call, this will follow the call and allow you to step through each line within the function.
- Step Over – If the line of code is a function (method) call, the function will be called and executed, but the debugger will skip over it and place the next step point after the function returns.
- Step Out – If the debugger is currently part way through a function, it will run the remaining lines of code without stepping through them, and place the next step point after the function returns.

Sometimes when you use 'Step Into' at the end of your code, it will jump into library code called *exe_common.inl*. If this happens, just press the red stop button and then close the library code with the small cross in the top right of the editor (or press **Ctrl+F4**).

**Dealing with Security Exceptions**

If you encounter an exception when you first try to debug your code, simply untick the box next to 'Break when this exception type is thrown', then click the green arrow, as shown below. You might have to do this several times for different exceptions. It's an unfortunate side effect of making our lab PCs more secure.

After doing this, you could export this project as another template and use that instead, so it remembers to ignore these exceptions in future labs. Or just dismiss these exceptions as they occur.


Exception Thrown
Exception thrown at 0x009AF90D in COMP124_Lab1.exe:
0xC0000005: Access violation reading location 0x00000000.
Show Call Stack | Copy Details
Exception Settings
Break when this exception type is thrown
Except when thrown from:
ntdll.dll
Open Exception Settings | Edit Conditions