

## Coursework – Assembly Language Programming

**Deadline: Thursday 31<sup>st</sup> March at 17:00**

**Weighting: 20%**

Follow the instructions to correctly submit your work. Penalties for late work will be applied in accordance with the Code of Practice on Assessment. Marking criteria are shown at the end of this document.

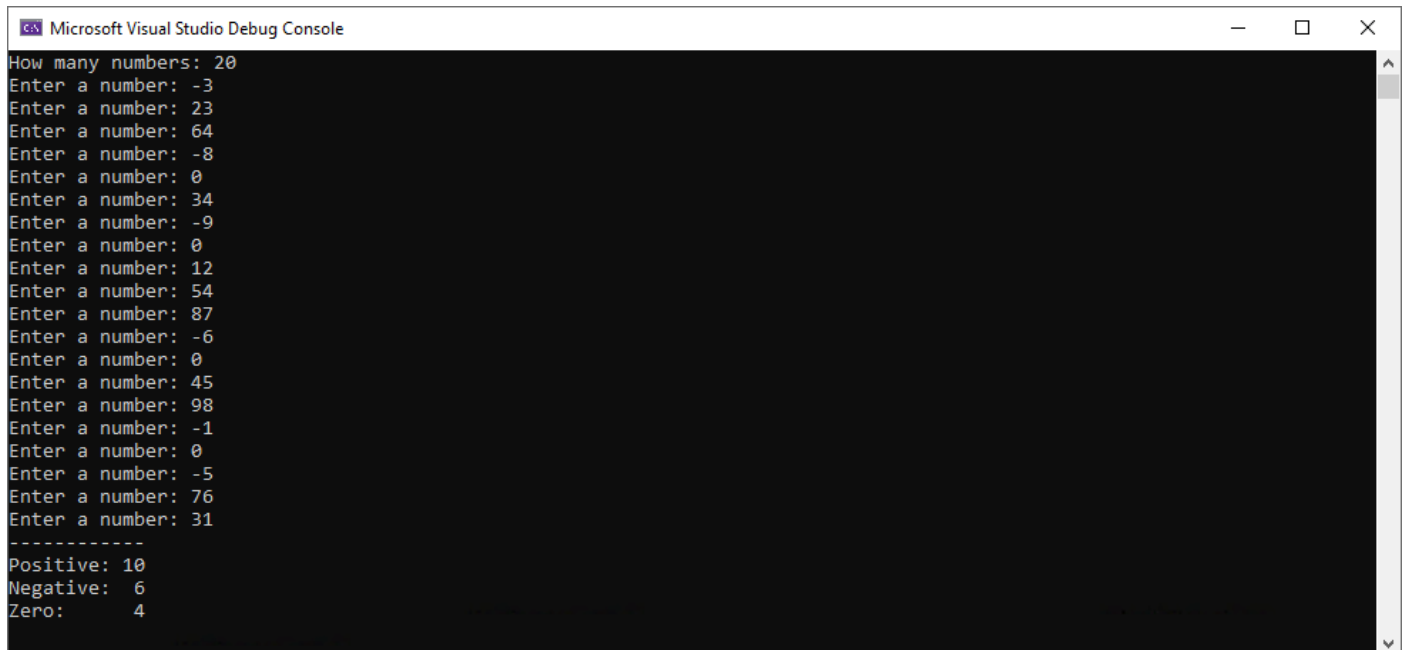
---

### Overview

The purpose of this assessment is to test your ability to write efficient assembly language code and describe succinctly how it works. Write a small assembly language program that...

- Asks the user how many numbers they would like to enter
- Loops for the given number of times
- Prompts the user to input a number each time round the loop
- Displays a summary of how many positive, negative and zero numbers were entered

Your assembly code will be inside an `_asm` block within a C program, with variables declared using C syntax, as shown during the lab tasks. Create a new Visual Studio project from your existing template. The screenshot below shows the expected output of the program.

A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio icon and the text "Microsoft Visual Studio Debug Console". The console output is as follows:

```
How many numbers: 20
Enter a number: -3
Enter a number: 23
Enter a number: 64
Enter a number: -8
Enter a number: 0
Enter a number: 34
Enter a number: -9
Enter a number: 0
Enter a number: 12
Enter a number: 54
Enter a number: 87
Enter a number: -6
Enter a number: 0
Enter a number: 45
Enter a number: 98
Enter a number: -1
Enter a number: 0
Enter a number: -5
Enter a number: 76
Enter a number: 31
-----
Positive: 10
Negative: 6
Zero: 4
```

All the aspects of the program have been covered in the lab tasks, so your program will be based on code that you've already written. In particular, you will use code to output strings, input integers, test or compare register values, jump according to status flags, and loop according to a counter register. You should try to make your code as optimal and efficient as possible.

## Useful Hint

This program will require a slightly more complex set of tests and jumps than the lab examples, because you are testing for three possible values (positive, negative, and zero). Write an initial version that detects only positive and negative numbers, because this will be easier. Submit this version if you can't work out how to detect zero values. You will still be able to obtain a passing mark for this assessment. A higher mark will be awarded if your code can detect all three input possibilities.

## Code Comments & Structure

Use the correct comment notation (C or assembly) to place useful explanations within the code. However, do not write lengthy comments that get in the way of readability. **Include your student ID as a comment at the top of your code.**

Assembly language doesn't make much use of indentation, but you should still make sure your code is easy to understand, including correct placement of any code labels. The C code containing your `_asm` block should be properly indented.

## How to Submit

Locate the Visual Studio project folder in your file system and navigate to the source file. It will have a **.cpp** extension. Refer to the very first lab sheet for guidance. Copy this to another location so you don't corrupt your project, then change its name so it uses the format below...

`username_studentID.cpp`

For example, if your username is **sgnab23** and your student ID is **201355426**, your file would be named...

`sgnab23_201355426.cpp`

Submit **only** this single file via the assessment page on Canvas. Please don't submit any other documents. We can only mark your work if we can easily locate and open the file. Please **do not** submit the entire Visual Studio project. Submissions will only be accepted via Canvas.

If you submit multiple attempts, we'll mark whichever was submitted most recently (up to the deadline). Canvas automatically adds a version number to the end of each submission. To be very clear, marks will **not** be deducted if Canvas renames your file.

## Marking Penalties

We will deduct **5 marks** if you do not follow the submission and naming instructions. In other words, if we need to extract your code from a project and/or rename the source file before we can mark it. Standard late penalties will be applied if you do not submit on time. Do not wait until the final moment to submit your work, as the deadline is strict.

## Non-Working Code

If you can't complete the solution, or if it doesn't work, submit something anyway. We can still give some marks for code that doesn't work, provided there is something for us to see.

Make use of code comments to tell us what isn't working, why you think it isn't working, and how you've tried to solve it. We can take all this into account when we look at the submission.

## Marking Criteria

There are 10 marking criteria, and each is weighted equally. When the marks are released, you will see your total mark (out of 100) and some brief comments about the criteria that impacted the mark the most. Feedback will be released in line with department policy, typically within three weeks of the deadline.

### Structure

- How you structure the code, including indentation and readability

### Comments

- How you succinctly and clearly explain what your code does

### Correctness

- Whether your code compiles and runs successfully with test input

### Output

- Your use of correct code to output text and numbers via system calls

### Input

- Your use of correct code to input numbers via system calls

### Conditions

- Your use of code jumps to execute code conditionally

### Loops

- Your use of code jumps to execute code repeatedly

### Efficiency

- Whether your code is efficient, in terms of register, instruction, and memory usage

### Stack

- Your use of the stack to pass parameters

### Display

- How closely your program output matches the example shown above

In all cases, be aware that we are looking for code that reflects the style and usage demonstrated during the lectures and practiced in the labs. You will lose marks if your code is written in a different format. Do not implement your solution in a high-level language. You are expected to demonstrate an understanding of the material presented in bundles 1, 2 and 3 of the module material.

## Academic Misconduct

Your submission **must be all your own work**. Do not share solutions with other students. Do not ask the teaching assistants for help with this assessment. The penalties for misconduct can be severe.