CMPS 290 - Lab #1
Getting Started

For this lab we are going to get started working in MARS and will just be performing a simple task. The main goal of the assignment will be to gain an understanding of how to perform basic I/O functions and how to end a program properly in addition to general usage. We must make use of calls to the system software to do any input or output operations and to end our program correctly. In order to do this, we will be making use of the **syscall** instruction, however, we must also set up specific registers properly in order to make it function how we would like it. See the accompanying resources on Canvas for more information on the syscall command and its use. Try to comment your code as much as possible (remember that # denotes a comment).

Part 1:

- Open MARS, create a new file, and call it "Lab1LastNameFirstName.asm".
- Create the MIPS version of the function: **f = g + (h - i) + j**
- Use the following values for the variables:
  - g = 20, h = 14, i = 6, and j = 22
- In order to place an immediate value into a register we have two options:
  - The first is how the language actually handles it which is to use an addition instruction with a 0 at one of the operands: addi $s0, $zero, 10#$s0 = 10 + 0 = 10
  - The second is a shorthand instruction load immediate: li $s0, 10 # $s0 = 10
- After getting the function itself working, let's print the result of the function.
- Since we are working with, and will want to print, an integer value then we will need to place a value of 1 into the $v0 register as found on the provided chart.
- We will then place the integer we wish to print into the $a0 register (f in this case).
  - Again using the add instruction and the $zero register will let us move values between registers. Ex. destination = original + $zero
  - We also have the option of a pseudo-instruction to perform the same operation in the move instruction: move $s0, $s1 # $s0 = $s1
- Finally, we are ready to use the syscall command to have our value printed to the screen.

Part 2:

- **For this part we will repeat what we have above and then modify it.**
  - **Feel free to copy and paste what you need to save yourself some time in this section. Read through the section to see what is repeatable.**
- Next we will take in two values from the user for use in the formula from part 1. We will replace h with 10 and j with 1 to keep things consistent.
- To take in an integer, we can see on the syscall chart that we must place a 5 in the $v0 register and then perform our syscall. We can then enter a value, with that value being stored in $v0 after the syscall.
- Take in both values and then using the same method from part 1, print the result.

Part 3:

- **Part 3 will technically occur between parts 1 and 2 in your program layout.**
- Once we have the result of our two operations printed, you'll notice that there is no spacing between the first output and the first input, so let's fix that.
- Before taking in the first number for part 2, insert a new line character using a similar method as printing an integer, but in this case using the code for printing a character as found on the chart of syscall codes.
- In order to specify which character to print, can provide the decimal number of that character as found on the ASCII table. We can also use the hexadecimal number of that character or, thanks to the assembler, the escaped character notation if applicable.
- A version of the ASCII table has been provided.
- This will allow separation between your two values.

Part 4:

- The final part of the assignment will be to close the program correctly, meaning it won't have the "dropped off bottom" portion mentioned in your program's output.
- In order to do this, wherever we want the program to terminate, we use the exit syscall as seen on the chart (use the first one).

Submission:

- Once you have completed the assignment, submit your assembly file to Canvas. Make sure your file assembles and runs correctly before uploading it. Double check that you have uploaded the correct file before submitting your assignment.

Sample Outputs:

After Part 1:

```
50
-- program is finished running (dropped off bottom) --
```

After Part 2:

```
5010
1
25
-- program is finished running (dropped off bottom) --
```

After Part 3:

```
50
10
1
25
-- program is finished running (dropped off bottom) --
```

Final Output:

```
50
10
1
25
-- program is finished running --
```