Observations with Installing ARM Assembly Programming on Raspberry Pi

To access the program we created we need to first type “nano” to be able to Read/Write in the file.
To assemble the code in

line 1 we use the instruction code, mov, to move the immediate, 5, to the general purpose register, r1.

In line 2, the instruction code, sub, is used to subtract the immediate, 1, from general purpose register, r1, and the result is stored in the general purpose register, r1.

Line 3, the instruction code, add, is used to add the immediate, 4, to the general purpose register, r1, and the result is stored in the general purpose register, r1.

Line 4 we use the instruction code, mov, to move the immediate, 1, to the general purpose register, r7.

To make comments we use the “@” sign in order for other team members to know what each line of code does.
To save the file we need to use “Ctrl W” and “Ctrl X” to exit the editor.
At first when we tried to run the code we did not see an output because in our program we did not have a line of code to print the contents of the general purpose register to the monitor. There is no output because we did not use the GDB debugger to step through the program and examine the contents of the registers and memory.
When debugging the code we used the GDB debugger to set a breakpoint at the end of the code and we looked at the contents of the registers. We saw that the general purpose register, r1, held the value 8 and the general purpose register, r7, held the value 1. The first row held the information about what register we were looking at and the second row held the values of the hexadecimal number while the third row held the values of the decimal number in each register.

Part 2:
In our code we used the instruction code, mov, to move the immediates, 10; 11; 7; 2, into the general purpose registers, r1; r2; r3; r4, respectively. We manipulated the values in the registers by adding the contents of general purpose register, r2, to general purpose register, r1 using the instruction code, add; multiplying the contents of general purpose register, r4, to general purpose register, r3 using the instruction code, mul; and then subtracted general purpose register, r3, from general purpose register, r1 using the instruction code, sub. Then we moved the immediate, 1, to the general purpose register, r7, using the instruction code, mov.

We debugged the program and saw that general purpose register, r1, held the value 7; general purpose register, r2, held the value 11; general purpose register, r3, held the value 14; general purpose register, r4 held the value 2; and general purpose register, r7, held the value 1.