P10

Due No Due Date **Points** 100 **Submitting** an external tool **Available** after Nov 7 at 12am

P10

2 of 2

Item 1

Item 2

Programming Practice: More Data Movement and Control Flow

An expected instruction count is given for each to allow you to sanity-check your solution while creating it (if you have five times the required number of instructions, you're probably going about it wrong), and let you know whether or not you can improve the efficiency of code after you have verified it works correctly.

A common mistake is to focus too much on the instruction count at the start, and to think that writing the correct number of instructions is more important than writing ones that work. **That is not the case, so please do not fall into that trap.** Having the "correct" instruction count is **completely useless** if the code does not function correctly.

Note that the given instruction count does not include the BR START instruction given in the code provided below. In some cases the count is approximate because there are multiple reasonable ways to structure the code.

Always start your code from a template file containing the .ORIG directive, header comments, etc. You can paste the following into a blank .asm file as a starting point:

```
; Filename:      <name of file>
; Author:         <your name>
; Description: <explanation of what the program does>

               .ORIG x0200
START

; add code here

DONE         BR START    ; repeat forever

; any program data you need to create goes below here
```

◂ Finish ▸

```
                    .END
```

Basic Skills

- Create a variable in memory called **MY_VAL** that has an initial value of **x2048**. Then use your program to increment (add 1 to) that value. At the end, the value in memory should be updated.
  Requires: 3 instructions and 1 .FILL directive
- Allocate an array called **NUMBERS** with 3 locations that are not initialized. Then (without using a loop) use your program to write the value **-1** to each of those locations.
  Requires: 5-6 instructions, 0-1 .FILL directives (depending on the implementation) and 1 .BLKW directive for the **NUMBERS** array.
- Modify the above program to use a loop to write the values to those three memory locations.
  Requires: 7-9 instructions, 0-2 .FILL directives (depending on the implementation), and 1 .BLKW directive for the **NUMBERS** array.
- Modify the loop version to make the uninitialized **NUMBERS** array have 32 locations, which are all modified by the loop to become **-1**.
  Think about how many instructions would be required for this version if you did not use a loop!
  Requires: 7-9 instructions, 1-2 .FILL directives (depending on the implementation), and 1 .BLKW directive for the **NUMBERS** array.

More Looping Behaviors with Memory

- Set **R0** to be the maximum value (assume signed values) of the 20 words starting at label **MyData**. The code should assume the data is in the range ±16383 to avoid overflow problems when comparing values (and your data should meet this requirement).
  Difficulty: 3.5
  Requires: 12-15 instructions and 1 .FILL directive (plus 20 .FILL directives with the array data, where the first one is labeled **MyData** and the remaining ones do not have labels).