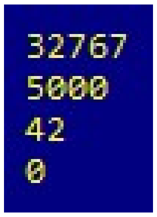
ECE220: Computer Systems & Programming **Fall 2021 ZJUI**
Machine Problem 1 **due: Saturday 25 September, 11:59:59 p.m.**

Measuring and Printing Subroutines



```
32767
5000
42
0
```

Your task this week is to write two subroutines: one to find the length of a string, and a second to print a non-negative integer as decimal to the display, as shown to the right. Subsequent MPs will make use of these two subroutines.

The objective for this week is to give you some experience with decomposing tasks into algorithms, with writing LC-3 assembly code, and with formatting output.

The Task

The first subroutine is STRLEN. The address of a string (the address of the first character) is passed to your subroutine in R0. Your subroutine must count the number of ASCII characters in the string, not including the terminal NUL, and return that value in R1.

The second subroutine is PRINT_DECIMAL. A non-negative number is passed into your subroutine in R1 (as 2s complement). The subroutine must convert the number into a sequence of ASCII digits (recall that '0' is x30, '1' is x31, and so forth) and print them to the display. The number should appear as a human would write it typically, without leading zeroes—see the examples above. The subroutine must NOT print any spaces, line feeds, or other characters to the display.

To make your life easier in the later MPs, except for the R1 value returned from STRLEN, neither of these subroutines may change any register values. In other words, except for R7, which is modified by JSR, both subroutines must preserve all bits in all other registers (again, STRLEN must modify R1 as well). Note that you are probably going to want to use TRAP instructions in PRINT_DECIMAL, so you should also preserve the return address (in R7) and restore it before executing the RET (JMP R7) instruction.

Rather than writing separate code for each decimal digit in PRINT_DECIMAL, you must use a look-up table. A look-up table specifies a function from a small, contiguous set of integers to an arbitrary result type. For example, a look-up table can translate a small integer into a decimal place value (10000, 1000, 100, and so forth). If each element in a look-up table requires a single memory address, one can add the index to the base address for the table to find the address at which the desired element is stored.

As you develop your subroutines, you may want to include code to test them at the start of the assembly file. Since the LC-3 will begin execution at the start of the file by default, you can then easily test your subroutines with the LC-3 tools.

Specifics

- Your code must be written in LC-3 assembly language and must be contained in a single file called **`mp1.asm`** in the **`mp/mp1`** subdirectory of your repository. We **will not grade** any other files.
- Your subroutine for computing the length of a string must be called STRLEN.
    - You may assume that the string starting at the address in R0 is valid and is terminated by an ASCII NUL character (x00, extended to x0000 in LC-3 memory).
    - You may NOT make any assumptions about string length. In particular, string length can be any non-negative integer, including 0.
    - Your subroutine must return the length of the given string in R1.
    - Your subroutine must preserve all register values other than R1 and R7.
- Your subroutine for printing a decimal number must be called PRINT_DECIMAL.
    - You may assume that the number given in R1 is in the range [0,32767] (non-negative when interpreted as 2's complement).
    - Your subroutine must print the number provided in R1 as a decimal value to the display.
    - Your subroutine must preserve all register values other than R7.
- Your code must not execute data—any bits generated using .FILL, .STRINGZ, .BLKW, or other data-generating pseudo-ops or directives.
- Except for loads (NOT stores) from the string passed to STRLEN, neither subroutine may access LC-3 memory outside of that allocated by your **`mp1.asm`** program. If you need space, use .FILL or .BLKW to create it, and use labels to address it.
- Your code must not produce any output to the display other than that specified, and your code must not read anything from the keyboard.
- You may not assume anything about how many times either subroutine is called, nor about the sequence of calls to your subroutines, nor about the input values other than as specified here.
- Your code must be well-commented and must include a table describing how registers are used within the PRINT_DECIMAL subroutine. Follow the style of examples provided to you in class and in the textbook.
- You may leave any code that you have used for testing at the start of your file, provided that it does not in any way interfere with your subroutines' functionality.

Testing

Testing is your responsibility. We have provided a file called **`test1.asm`** that performs a basic test on your code. Instructions on how to use it are included at the start of the file, and you may want to generalize the approach to suit your own purposes in testing your code.

We suggest that you test several different string lengths for STRLEN, including an empty string (just a terminal NUL). For PRINT_DECIMAL, be sure to test at least the values shown in the image at the start of this document.

Tools

ZJUI alumni developed a set of tools to help you to improve your coding style and programming ability. Please be sure to try out the VSCode extension for LC-3 when you are developing your code as well as the Webtool interface when debugging it. When you submit a copy of your code to Github, we will automatically test it against our solution and give you feedback about errors in your code as well as differences between your code and our solution. Please note that while the tools are fairly thorough, they do not guarantee that your code is free of bugs, nor that you will earn full points.

Grading Rubric

Functionality (65%)

- STRLEN
  - 10% - produces correct output in R1 for any string, including an empty string
  - 5% - does not modify any register values other than R1 and R7 when called, only reads the string provided, and otherwise does not access memory outside of **`mp1.asm`**
- PRINT_DECIMAL
  - 30% - produces correct output digits
  - 10% - does not print leading zeroes
  - 10% - does not modify any register value other than R7 when called, and does not access memory outside of **`mp1.asm`**

Style (15%)

- 15% - PRINT_DECIMAL organized using nested loops (rather than as a separate loop per digit to be printed); handling the last digit (the 1s digit) separately is acceptable
- **-10% PENALTY** - VSCode extension reports any errors or warnings; note that **even a single warning** incurs the full 10% penalty

Comments, Clarity, and Write-up (20%)

- 5% - each subroutine has a paragraph explaining what it does and how it must be called (these are given to you; you just need to document your work)
- 5% - the PRINT_DECIMAL subroutine has a register table (comments) explaining how registers are used in that part of the code
- 10% - code is clear and well-commented

Note that some categories in the rubric may depend on other categories and/or criteria. For example, if your code does not assemble, you will receive no functionality points. Note also that the remaining LC-3 MPs (two of them) will build on these subroutines, so you may have difficulty testing those MPs if your code does not work properly for this MP. You will also be penalized heavily if your code executes data, for example. Finally, your subroutines must be written so that they can be called any number of times in any order with any legal input values, and we will deduct points if such is not the case.