CS 61 - Programming Assignment 04

Objective

The purpose of this assignment is to illustrate how the .FILL pseudo-op performs the task of translating textual numbers (such as the string "#5392") into actual numbers (i.e. five thousand three hundred and ninety two, represented of course as a 16-bit two's complement binary value).

High Level Description

Prompt the user to enter in a signed multi-digit number (max 5 digits) from the keyboard. Convert the string of characters entered (as separate ascii codes for decimal numeric digits) into the 16-bit number they represent, and store the result in **R5**. The range of acceptable values is [-32768, +32767]; the absence of a sign means the number is positive.

Your Tasks

Your program can be broken down into the following tasks:

Read in the '+' or '-'. If the character is a '-', remember to make the final result negative *(i.e. take the 2's complement of **R5** at the end).* If the result is positive then ... don't.

Convert the string of characters input by the user into the binary number they represent (see examples). To do this, you can follow this algorithm:

- Initialize **R5** to 0 (**DO NOT** *do this by LD'ing a 0 from memory! There is a much simpler & faster way!)*
- Convert each digit to binary as it is typed in, and add it to **R5**; if another digit is entered, multiply **R5** by 10, and repeat. Stop when you detect the ENTER (x0A):
    - For example, if the user types '2', then **R5** will contain
      #2 == b0000 0000 0000 0010
    - If the user then types a '3', making the string now read "23", then **R5** will contain 2 x 10 + 3 == #23 == b0000 0000 0001 0111
    - If the user then types '4', making the string read "234", then **R5** will contain
      23 x 10 + 4 == #234 == b0000 0000 1110 1010

You must also perform ***input character validation*** with this assignment – i.e. reject any <u>non-numeric</u> input character.
That is, if the user enters "+23g", your program should "choke" on the 'g', print an error message (see sample output), and start over at the beginning with the initial prompt.
However, you do not have to detect overflow in this assignment – we will only test your code with inputs in the range [-32768, +32767].

Expected/ Sample output
Output

- Prompt
  - Input a positive or negative decimal number (max 5 digits), followed by ENTER
    - Newline terminated
- Error Message
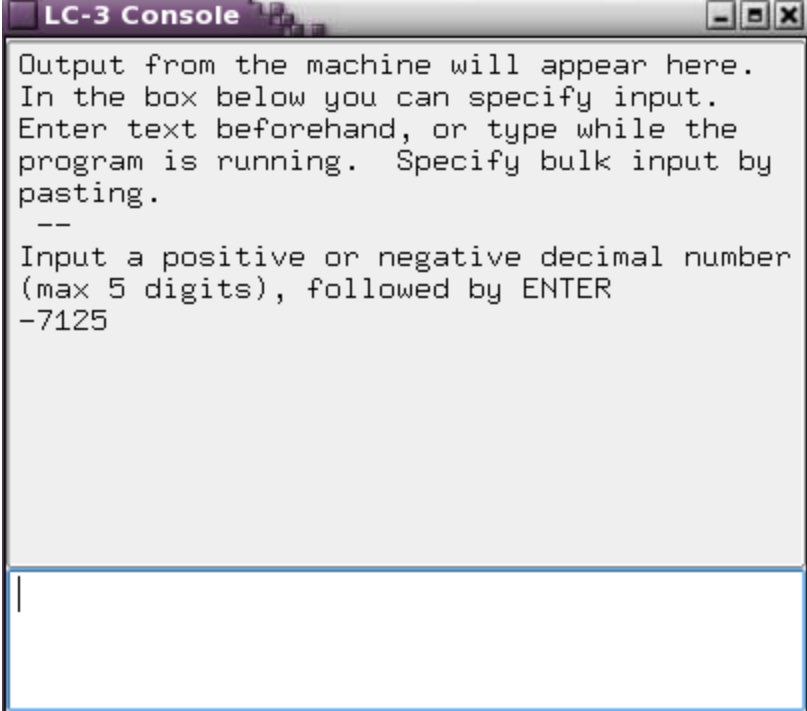 - ERROR INVALID INPUT
    - Newline terminated

Example

If the user enters "+7246", your program should read the '+', '7', '2', '4', '6' and end up with the value b0001 1100 0100 1110 in **R5** *(which corresponds to the number #7246, or x1C4E).*

If the users enters "-14237", your program should read the '-', '1', '4', '2', '3', '7' and end up with the value #-14237 == xC863 == b11001000 01100011 in **R5**.

<u>NOTE</u>: In the following examples, the final result is shown in R2.
This is NOT the register you will be using in your code - use the register specified above!!

```
LC-3 Console                    _ □ ×
Output from the machine will appear here.
In the box below you can specify input.
Enter text beforehand, or type while the
program is running.  Specify bulk input by
pasting.
 --
Input a positive or negative decimal number
(max 5 digits), followed by ENTER
-7125

|
```

| R0 | x000A | 10 | R1 | x0000 | 0 | R2 | xE42B | -7125 | R3 | x0004 | 4 |
| R4 | xFFF6 | -10 | R5 | x0000 | 0 | R6 | xFFFF | -1 | R7 | x3049 | MAX_INPUT |

(Valid input with a negative sign)

```
LC-3 Console                              [-][□][X]
Output from the machine will appear here.
In the box below you can specify input.
Enter text beforehand, or type while the
program is running.  Specify bulk input by
pasting.
 --
Input a positive or negative decimal number
(max 5 digits), followed by ENTER
+12345




```

| RO | x000A | 10 | R1 | x0000 | 0 | R2 | x3039 | 12345 | R3 | x0005 | 5 |
| R4 | xFFF6 | -10 | R5 | x0000 | 0 | R6 | x0001 | 1 | RZ | x3049 | MAX_INPUT |

(Valid input with a positive sign)

```
LC-3 Console                              [-][□][X]
Output from the machine will appear here.
In the box below you can specify input.
Enter text beforehand, or type while the
program is running.  Specify bulk input by
pasting.
 --
Input a positive or negative decimal number
(max 5 digits), followed by ENTER
123




```

| RO | x000A | 10 | R1 | x0000 | 0 | R2 | x007B | 123 | R3 | x0003 | 3 |
| R4 | xFFF6 | -10 | R5 | x0000 | 0 | R6 | x0000 | 0 | RZ | x3049 | MAX_INPUT |

(Valid input with No sign)

```
LC-3 Console                         _ □ ✕
 Output from the machine will appear here.
 In the box below you can specify input.
 Enter text beforehand, or type while the
 program is running.  Specify bulk input by
 pasting.
 --
 Input a positive or negative decimal number
 (max 5 digits), followed by ENTER
 +2g
 ERROR INVALID INPUT
 Input a positive or negative decimal number
 (max 5 digits), followed by ENTER
 126
```
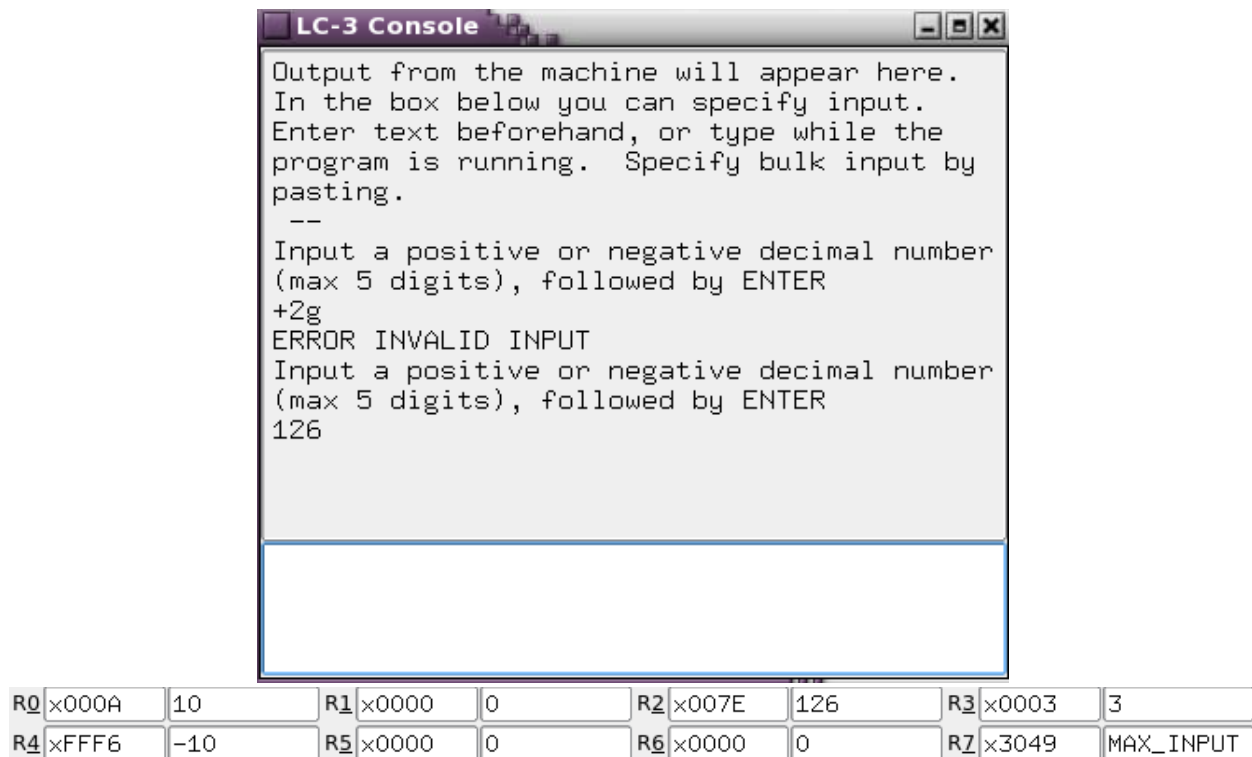
| R0 | x000A | 10 | R1 | x0000 | 0 | R2 | x007E | 126 | R3 | x0003 | 3 |
| R4 | xFFF6 | -10 | R5 | x0000 | 0 | R6 | x0000 | 0 | R7 | x3049 | MAX_INPUT |

(Invalid input)

Note:

- You must echo the digits as they are input (no "ghost writing").
- You do **not** have to output the converted binary number. It should "simply" be sitting happily in **R5**, where you can check it in simpl.
- What should happen when an error occurs?
  - Output the error message and reprompt the user for input
- Other Errors:
  - Nothing entered before ENTER
  - only sign is entered before ENTER
  - first character entered is neither a sign nor a digit
- *REMEMBER: all outputs must be newline terminated*

Your code will obviously be tested with a range of different values: Make sure you test your code likewise!

Uh...help?

Try to write this program out in C++/pseudocode before directly tackling it in LC3. Doing so often helps to simplify the process and usually only takes a few minutes to do if you think it through carefully.

To mark the distinction between a positive number and a negative one, set a "flag" (say... R5). If the first character is a '-', then put a negative number (like #-1) into R5. Otherwise, set R5 to #0 (i.e. non-negative). That way, after you translate the rest of the input characters into the number they represent, you can use a quick IF-statement (like BRn MAKE_NEGATIVE) to toss in the two lines of code it takes to take the 2's complement of the result.

Submission Instructions

Submit to iLearn for testing, feedback and grading. Some feedback will be sent via E-mail.

Comments/Feedback

In the diff files sent to you in the feedback email, there will be two columns: the left column is the output from your program, and the right is the expected output. Any line that has a | on it means your output did not match the solution. The only line numbers of interest are 13 - 17. You can ignore the rest, even if they have a '|'.

Comment	Meaning
Invalid prompt [1]	Means that your first prompt is wrong. Likely do to not using the provided template. Tsk Tsk
Did not echo input/incorrect echoing of input [1]	Means that you did not echo the first number entered correctly. Most common cause is not ending the output with a newline
Invalid Error Message	Means that your error message is wrong. Likely do to not using the provided template. Tsk Tsk
Invalid prompt [2]	Means that your second prompt is wrong. Likely do to not using the provided template. Tsk Tsk
Did not echo input/incorrect echoing of input [2]	Means that you did not echo the second number entered correctly. Most common cause is not ending the output with a newline
Wrong Register Value	The value in Register **R5** is not correct. Something is wrong with your algorithm
Did not have ending NEWLINE	Program did not end with a NEWLINE

Rubric

- Grades fall into the following four ranges:

  (i) 8 - 10: pass (perhaps with minor errors - missing spaces, mis-spellings, etc.)

  (ii) 5 - 7: failed, but easily fixable (e.g. missing newlines)

  (iii) 1 - 4: failed - wrong output, you should probably throw it away and start over!

  (iv) 0: abject fail - no submission, or did not assemble *(same as fail, but even more embarrassing!)*

- Using the provided template is **required**.
- Not putting the constructed value into **R5** will result in a failing score.

Comics?!Sweet!!

Source: http://xkcd.com/237/