CS 2110 Timed Lab 4
Subroutines in LC-3 Assembly

Austin Adams, Cem Gokmen, Preston Olds, Cliff Panos, Michael Xu

Fall 2018

Contents

1 Before You Begin **2**

2 Timed Lab Rules - Please Read **2**

2.1 General Rules . . . 2

2.2 Submission Rules . . . 2

2.3 Is collaboration allowed? . . . 3

3 Overview **4**

3.1 The `divide` Subroutine . . . 4

3.2 Reminder: `JSR` vs. `JSRR` . . . 4

3.3 Reference Sheets . . . 4

3.3.1 The LC-3 Calling Convention . . . 4

3.3.2 The LC-3 Boilerplate . . . 4

4 Instructions **5**

4.1 Pseudocode . . . 5

4.2 Restrictions . . . 5

5 Testing Your Work **6**

6 Common Errors **6**

7 Rubric **6**

8 Deliverables **6**

9 LC-3 Assembly Programming Requirements **7**

9.1 Overview . . . 7

1 Before You Begin

Please take the time to read the entire document before starting the assignment. We have made some important updates, and it is your responsibility to follow the instructions and rules.

2 Timed Lab Rules - Please Read

2.1 General Rules

1. You are allowed to submit this timed lab starting at the moment the assignment is released, until you are checked off by your TA as you leave the recitation classroom. Gradescope submissions will remain open until 7:15 pm - but you are not allowed to submit after you leave the recitation classroom under any circumstances. **Submitting or resubmitting the assignment after you leave the classroom is a violation of the honor code - doing so will automatically incur a zero on the assignment and might result in you being referred to the Office of Student Integrity.**
2. Make sure to give your TA your Buzzcard before beginning the Timed Lab, and to pick it up and get checked off before you leave. **Students who leave the recitation classroom without getting checked off or submit after getting checked off will receive a zero.**
3. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. **The information provided in this Timed Lab document takes precedence.** If in doubt, please make sure to indicate any conflicting information to your TAs.
4. Resources you are allowed to use during the timed lab:
    - Assignment files
    - Previous homework and lab submissions
    - Your mind
    - Blank paper for scratch work (please ask for permission from your TAs if you want to take paper from your bag during the Timed Lab)
5. Resources you are **NOT** allowed to use:
    - The Internet (except for submissions)
    - Any resources that are not given in the assignment
    - Textbook or notes on paper or saved on your computer
    - Email/messaging
    - Contact in any form with any other person besides TAs
6. **Before you start, make sure to close every application on your computer.** Banned resources, if found to be open during the Timed Lab period, will be considered a violation of the Timed Lab rules.
7. We reserve the right to monitor the classroom during the Timed Lab period using cameras, packet capture software, and other means.

2.2 Submission Rules

1. Follow the guidelines under the Deliverables section.

2. You are also responsible for ensuring that what you turn in is what you meant to turn in. After submitting, you should be sure to download your submission into a brand new folder and test if it works. There are no excuses if you submit the wrong files; what you turn in is what we grade. In addition, your assignment must be turned in via Gradescope. Under no circumstances whatsoever will we accept any email submission of an assignment. Note: if you were granted an extension, you will still turn in the assignment over Gradescope.

3. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

2.3 Is collaboration allowed?

Absolutely NOT. No collaboration is allowed for timed labs.

3 Overview

For this assignment, you will implement the subroutine called `converge` in LC-3 assembly according to the calling convention you have learned in lecture and practiced in Homework 07.

This subroutine recursively calls itself (as well as calls `divide`) until the parameter `n` converges to one.

3.1 The `divide` Subroutine

The `divide` subroutine takes two arguments:

`n` : the numerator
`d` : the denominator

The subroutine returns `n / d`.

Consider the following example outputs:

```
divide(15, 2)  == 7
divide(100, 5) == 20
```

```
divide(n, d):

  var q = 0
  var r = n

  while (r >= d):
    q = q + 1
    r = r - d

  return q          ;; q == n / d
```

Note: The `divide` subroutine has already been written and assembled for your use in `timedlab4.asm`!

3.2 Reminder: `JSR` vs. `JSRR`

Subroutines are sometimes farther away in memory than `JSR`'s `PCoffset11` allows. Recall that `JSRR BaseR` jumps to the address specified in `BaseR`, as opposed to an offset (i.e. `JSR LABEL`)!

3.3 Reference Sheets

You've been provided with two extra PDFs under `reference/`:

`lc3-boilerplate.pdf`

`lc3-convention.pdf`

3.3.1 The LC-3 Calling Convention

Please reference `lc3-convention.pdf` for some guidance on the LC-3 calling convention. Also, consider the following visual representation of a stack frame:

Last Local
...
First Local
Old Frame Pointer
Return Address
Return Value
First Argument
...
Last Argument

← `R6`: Stack pointer

← `R5`: Frame pointer

3.3.2 The LC-3 Boilerplate

Please reference `lc3-boilerplate.pdf` for templates and guidance translating pseudocode to LC-3 assembly.

4 Instructions

The following files have been provided for you:

```
timedlab4.asm
reference/
    lc3-boilerplate.pdf
    lc3-convention.pdf
```

You will be editing `timedlab4.asm`. We have defined the assembly language label `STACK`. You should not access or alter the value at this label, as it is intended for use by the tester.

The subroutine `converge` takes the following arguments:

`n`: an integer

Given any value of `n`, this subroutine will recursively call itself until `n` *converges* to one. The `return` value is the number of recursive calls it took to converge.

For example:

`converge(8)`: `converge(4)` --> `converge(2)` --> `converge(1)` which returns `3`.

`converge(3)`: `converge(10)` --> `converge(5)` --> `converge(16)` -->
`converge(8)` --> `converge(4)` --> `converge(2)` -->
`converge(1)` which returns `7`.

Note: You need not understand why this works to complete the assignment!

Note: In the above examples, there would also be calls to `divide`.

4.1 Pseudocode

Implement this pseudocode following the label `converge` in `timedlab4.asm`:

```
converge(n):

  if (n == 1):
    return 0

  var div = divide(n, 2)
  var mod = n % 2                   ;; Hint: a % b == a & (b - 1) in this case
                                    ;;       since b is a power of 2!
  if (mod == 0):
    return converge(div) + 1
  else:
    return converge((3 * n) + 1) + 1
```

Notice you'll need to call the `divide` subroutine. The address of this subroutine is provided at the label `DIV_ADDR`. Refer to the prior section for information on this subroutine's arguments and return value.

4.2 Restrictions

You are not allowed to use Appendix A or the textbook during this assignment.

5 Testing Your Work

To test your program, upload `timedlab4.asm` to the Timed Lab 4 assignment on Gradescope. You may resubmit your work as many times as needed, until you sign out and leave the classroom.

6 Common Errors

To trace problems with your code, load the file into `complx`, the LC-3 simulator. To use complx:

1. In the Terminal, type `complx`
2. In the `File` menu, click `Reload`, and open your assembly file (`timedlab4.asm`)
3. Use the `Step` button to run each instruction one step at a time, or use the `Run` button to execute all of the instructions until `HALT`

7 Rubric

The output of the Gradescope autograder is an approximation of your score on this assignment. The tool is provided so you can evaluate whether your submission fulfills the assignment expectations.

However, we reserve the right to run additional tests, fewer tests, different tests, or potentially change individual tests – your final score will be determined by your instructors, and there is no guarantee your score will correlate with the tester output.

8 Deliverables

Please upload the following file to the assignment on Gradescope:

1. `timedlab4.asm`

Do NOT upload an archive; upload the file individually.

Be sure to check your Gradescope test score before you leave the room.

9 LC-3 Assembly Programming Requirements

9.1 Overview

1. Your code must assemble with **NO WARNINGS OR ERRORS**. To assemble your program, open the file with Complx. It will complain if there are any issues. **If the code in this file does not assemble, you WILL get a zero for that file.**

2. **Comment your code!** This is especially important in assembly, because it's much harder to interpret what is happening later, and you'll be glad you left yourself notes on what certain instructions are contributing to the code. Comment things like what registers are being used for and what less intuitive lines of code are actually doing. To comment code in LC-3 assembly just type a semicolon (;), and the rest of that line will be a comment.

3. Avoid stating the obvious in your comments; it doesn't help in understanding what the code is doing. Try to write high-level pseudo-code instead!

   Good Comment

   ```
   ADD R3, R3, -1          ; counter--
   BRp LOOP                ; if counter == 0 don't loop again
   ```

   Bad Comment

   ```
   ADD R3, R3, -1          ; Decrement R3
   BRp LOOP                ; Branch to LOOP if positive
   ```

4. **DO NOT assume that ANYTHING in the LC-3 is already zero.** Treat the machine as if your program was loaded into a machine with random values stored in the memory and register file.

5. Following from 3. You can randomize the memory and load your program by doing File → Randomize and Load.

6. Do not add any comments beginning with @plugin or change any comments of this kind.

7. **Test your assembly.** Don't just assume it works and turn it in.