CS 2110 Timed Lab 2
Subroutines in LC-3 Assembly

Preston Olds

Summer 2018

Contents

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 6 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 be 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 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 turned 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. 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 Canvas/Gradescope. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Canvas/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

3.1 Assignment

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

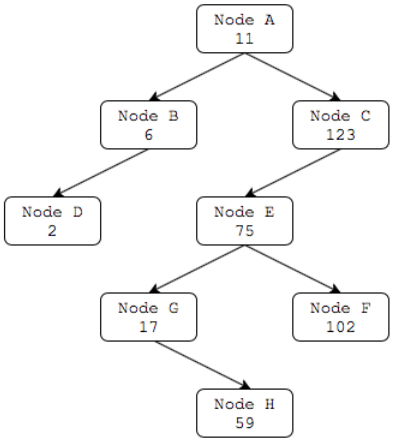
This function recursively traverses a binary tree, checking the `data` held by each of the nodes against a `query`. If the matching `data` is found, the program prints the node's `name` and returns the node's address. If the matching integer is **not** found, the program prints `"Did not find the data"` and returns `NULL` (i.e. `0x0000`).

Each node of this binary tree consists of four words in memory and has the following structure:

```
Word 1: Address of the name string
Word 2: Address of the left node
Word 3: Address of the right node
Word 4: Data
```

Remember: Each of the four components is a full word (i.e. 16 bits).

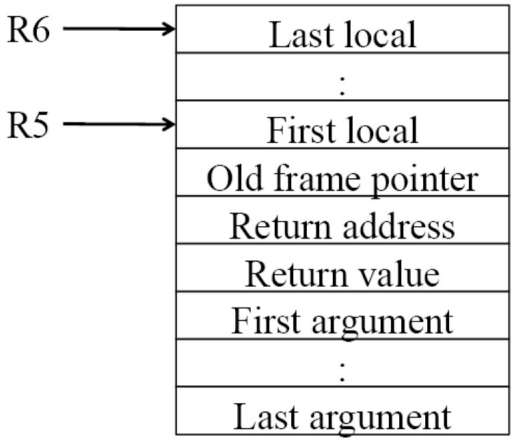
Consider a binary tree with eight nodes, depicted in the figure below. The right side of the diagram portrays how these nodes could be stored in memory. Given the example tree, Node A's `left` would point to address `x6004`, the address of Node B, and `right` would point to address `x6008`, the address of Node C. The `name` component is an address pointing to the beginning of a string elsewhere in memory. For example, in the file you've been provided, Node A's `name` is stored at address `x7000` (i.e. `x7000` is stored at address `x6000`).



ADDRESS	CONTENTS		ADDRESS	CONTENTS
x6000	name	Node A	x7000	'N'
x6001	left		x7001	'o'
x6002	right		x7002	'd'
x6003	data		x7003	'e'
x6004	name	Node B	x7004	' '
x6005	left		x7005	'A'
x6006	right		x7006	'\0'
x6007	data		x6007	'N'
x6008	name	Node C	x6008	'o'
x6009	left		x6009	'd'
x600A	right		x600A	'e'
x600B	data		x600B	' '
x600C	name	Node D	x600C	'B'
x600D	left		x600D	'\0'
x600E	right			
x600F	data			
x6010	name	Node E		
x6011	left			
x6012	right			
x6013	data			
x6014	name	Node F		
x6015	left			
x6016	right			
x6017	data			
x6018	name	Node G		
x6019	left			
x601A	right			
x601B	data			
x601C	name	Node H		
x601D	left			
x601E	right			
x601F	data			

3.2 Calling Convention

A visual representation of the stack according to the LC-3 calling convention is posted here for your reference:



4 Instructions

The following files have been provided for you.

1. `timedlab2.asm`
2. `timedlab2_test.xml`

You will be editing `timedlab2.asm`. We have defined the assembly language labels `STACK`, `ROOT`, and `QUERY`. You **should not** access or alter the values at these labels, as they are intended for use by the tester.

`btree_search` takes the following arguments:

`root`	The address of the root node
`query`	The data we are searching for

Implement this pseudocode following the label `btree_search` in the provided `.asm` file:

```
btree_search(root, query)
{
    if (root == 0) {
        puts("Did not find the data")
        return root;
    } else if (query == (root -> data)) {
        puts("Found the data at ")
        puts(root -> name)
        return root;
    } else if (query < (root -> data)) {
        return btree_search(root -> left, query);
    } else {
        return btree_search(root -> right, query);
    }
}
```

Notice the `->` notation: You should interpret `root -> data` as accessing `data` stored by the `root` node.

4.1 Restrictions

You are **not allowed** to use Appendix A or the textbook during this timed lab.

5 Testing Your Work

We have provided you with test cases in `timedlab2_test.xml`. To run the test,

1. In the terminal, navigate to the folder where you've saved `timedlab2.asm` and `timedlab2_test.xml`
2. Run the command `lc3test timedlab2_test.xml timedlab2.asm`

6 Common Errors

To debug, we recommend using complx, an LC-3 simulator. To use complx:

1. In the terminal, type `complx`
2. In the 'File' menu, click 'Reload', and open your assembly file (`timedlab2.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.

7 Rubric

The output of the autograder is an approximation of your score on this timed lab. It is a tool provided to students so that you can evaluate how much of the assignment expectations your submission fulfills. However, **we reserve the right to run additional tests, fewer tests, different tests, or change individual tests** - your final score will be determined by your instructors and no guarantee of tester output correlation is given.

8 Deliverables

Please upload the following files to Canvas:

1. `timedlab2.asm`

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 your code 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.

   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. Use the LC-3 calling convention. This means that all local variables, frame pointer, etc... must be pushed onto the stack. Our autograder will be checking for correct stack setup.
7. Start the stack at xF000. **The stack pointer always points to the last used stack location.** This means you will allocate space **first**, then store onto the stack pointer.
8. Do NOT execute any data as if it were an instruction (meaning you should put .fills after **HALT** or RET).
9. Do not add any comments beginning with @plugin or change any comments of this kind.
10. **Test your assembly.** Don't just assume it works and turn it in.