CS 2110 Timed Lab 3: Assembly with Subroutines

Your TAs

Spring 2020

Contents

Please take the time to read the entire document before starting the assignment. It is your responsibility to follow the instructions and rules.

1 Timed Lab Rules - Please Read

1.1 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 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.

1.2 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 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 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.

1.3 Is collaboration allowed?

Absolutely NOT. No collaboration is allowed for timed labs.

2 Overview

2.1 Purpose

The purpose of this timed lab is to test your understanding of the LC-3 calling convention by testing your ability to call subroutines by using the stack.

2.2 Task

You will implement one subroutine listed below in LC-3 assembly language. Please see the detailed instructions for the subroutine on the following pages. We have provided pseudocode for the subroutine, and suggest that you follow the algorithm when writing your assembly code. Your subroutine must adhere to the LC-3 calling convention.

1. `tl3.asm`

2.3 Criteria

Your assignment will be graded based on your ability to correctly translate the given pseudocode for a subroutine (function) into LC-3 assembly code, following the LC-3 calling convention. Please use the LC-3 instruction set when writing these programs. Check the deliverables section for what you must submit to gradescope and the deadlines.

You must obtain the correct values for each function. In addition, registers R0-R5 and R7 must be restored from the perspective of the caller, so they contain the same values after the caller's JSR subroutine call. Your subroutine must return to the correct point in the caller's code, and the caller must find the return value on the stack where it is expected to be. If you follow the LC-3 calling convention correctly, each of these things will happen automatically.

While we will give partial credit where we can, your code must assemble with no warnings or errors. (Complx will tell you if there are any.) If your code does not assemble, we will not be able to grade that file and you will not receive any points.

3 Detailed Instructions

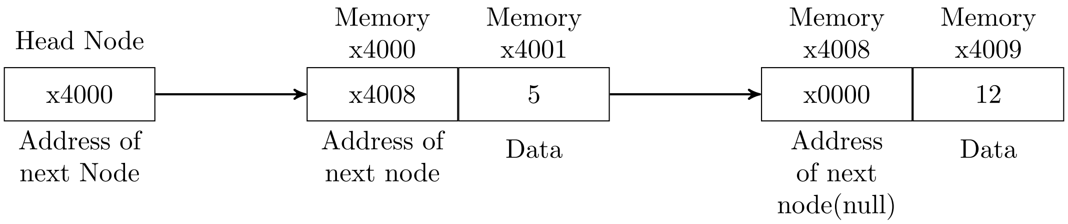
For this Timed Lab, you will be implementing a function named `sumPowers`, which takes in three arguments:

- `head`: The head of a linked list
- `exp`: A natural number
- `func`: The address of a function named func

To complete the timed lab, `sumPowers` must do the following: For each node in the linked list starting at `head`, you will calculate (node.data)^{exp}, and add this value to a running sum named `sum`. Once this has been done for each node in the linked list, you will return the value `func(sum)`. Pseudocode for what is expected can be found below. However, first a quick refresher on what a linked list looks like in LC-3 Assembly.

3.1 Linked List Data Structure

The below figure depicts how each node in our linked list is laid out. Each node will have two attributes: the next node, and a value for that node.



4 Pseudocode and Checkpoints

4.1 Pseudocode

Note: the function pow(int val, int exp) is provided for you at label POW in tl3.asm.

```
int sumPowers(Node head, int exp, function func) {
    int sum = 0;
    while(head != Null) {
        sum += pow(head.data, exp);
        head = head.next;
    }
    return func(sum);
}
```

4.2 Checkpoints (70 points)

In order to get all of the points for this timed lab, your code must meet these checkpoints:

- Checkpoint 1 (10 points): Store the address of the first node in the linked List to the label `FIRSTNODE`. This would be the address x4000 in the linked list example found above the pseudocode.
- Checkpoint 2 (25 points): Iterate through the linked list and for each node, calculate the value `pow(node.data, exp)` and add it to the sum.
- Checkpoint 3 (25 points): Once `head == Null`, store the value of sum into the label `FINISHEDSUM`.
- Checkpoint 4 (10 points): Match the pseudocode by returning the value `func(sum)` at the end of the subroutine. **Hint: if you can't figure out how to do this step, just return sum instead so that the auto-grader doesn't complain and you will receive 60 points for the checkpoints.**

All of the labels (`FIRSTNODE`, `FINISHEDSUM`, `POW`) mentioned above are provided in `tl3.asm`.

4.3 Other Requirements (30 points)

Your subroutine must follow the LC-3 calling convention. Specifically, it must fulfill the following conditions:

- When your subroutine returns, every register must have its original value preserved (except R6).
- When your subroutine returns, the stack pointer (R6) must be decreased by 1 from its original value so that it now points to the return value.
- During the execution of your subroutine, you must make n calls to `pow`, where n = length of the linked list, and one call to `func` (as described in the pseudocode).
  - If the autograder claims that you are making an unknown subroutine call to some label in your code, it may be that your code has two labels without an instruction between them. Removing one of the labels should appease the autograder.

5 Deliverables

Turn in the files

1. `tl3.asm`

on Gradescope during your assigned Lab section on Monday, March 9th.

6 LC-3 Assembly Programming Requirements

6.1 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. You should not use a compiler that outputs LC3 to do this assignment.

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

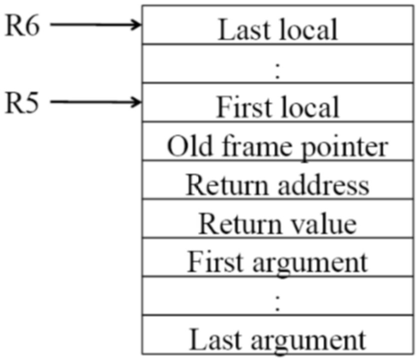
7 Appendix

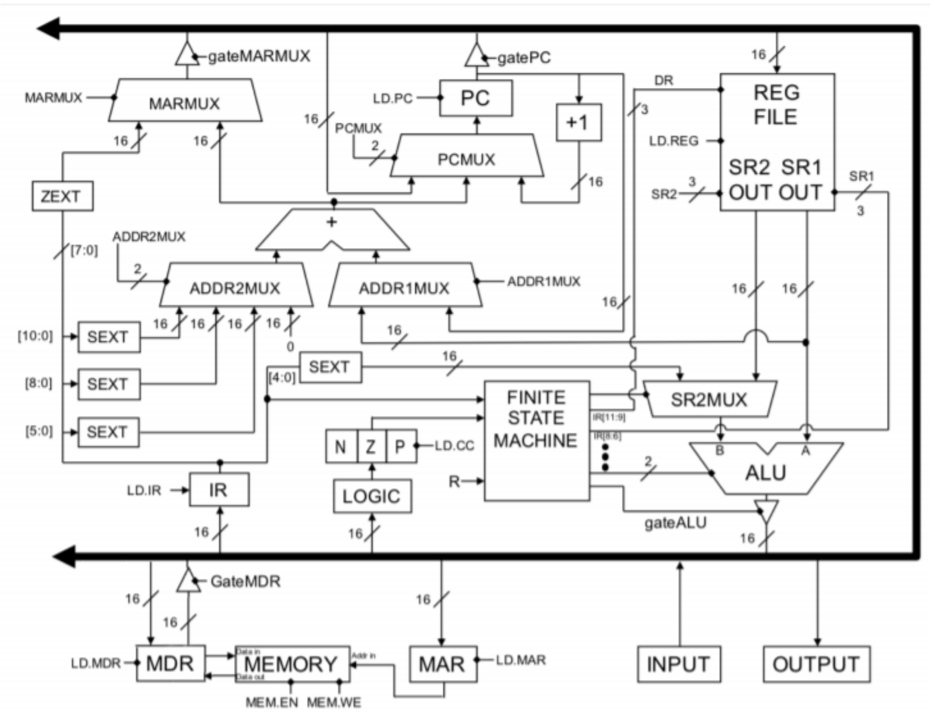
7.1 Appendix A: LC-3 Instruction Set Architecture

ADD	0001	DR	SR1	0	00	SR2
ADD	0001	DR	SR1	1	imm5	
AND	0101	DR	SR1	0	00	SR2
AND	0101	DR	SR1	1	imm5	
BR	0000	n	z	p	PCoffset9	
JMP	1100	000	BaseR	000000		
JSR	0100	1	PCoffset11			
JSRR	0100	0	00	BaseR	000000	
LD	0010	DR	PCoffset9			
LDI	1010	DR	PCoffset9			
LDR	0110	DR	BaseR	offset6		
LEA	1110	DR	PCoffset9			
NOT	1001	DR	SR	111111		
ST	0011	SR	PCoffset9			
STI	1011	SR	PCoffset9			
STR	0111	SR	BaseR	offset6		
TRAP	1111	0000	trapvect8			

Trap Vector	***Assembler Name***
x20	GETC
x21	OUT
x22	PUTS
x23	IN
x25	HALT

Device Register	***Address***
Keybd Status Reg	xFE00
Keybd Data Reg	xFE02
Display Status Reg	xFE04
Display Data Reg	xFE06





Boolean Signals	
LD.MAR	GateMARMUX
LD.MDR	GateMDR
LD.REG	GatePC
LD.CC	GateALU
LD.PC	LD.IR
MEM.EN	MEM.WE

MUX Name	Possible Values
ALUK	ADD, AND, NOT, PASSA
ADDR1MUX	PC, BaseR
ADDR2MUX	ZERO, offset6, PCoffset9, PCoffset11
PCMUX	PC+1, BUS, ADDER
MARMUX	ZEXT, ADDER
SR2MUX	SR2, SEXT