CS 2110 Timed Lab 3: Assembly

Your TAs

Fall 2019

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.
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

In this timed lab, you will writing code to search an string for a given character, and change the case of that character each time you encounter it in the string. You will be graded on 3 "checkpoints." For each checkpoint you achieve you will earn a set number of points. The string will be stored starting at the label "STRING" and the character will be stored at the label "CHAR". The pseudocode for this algorithm is provided and includes the checkpoints.

You are guaranteed the following:

- The string stored at label "STRING" will either be ALL uppercase OR ALL lowercase.
- The character to search for stored at label "CHAR" will always be lowercase.
- All characters you are checking will be part of the alphabet (no punctuation and such).
- You will be provided values at lables "TOUPPER", "TOLOWER", and "LOWER_A" to help you both check and change the case of a given character.
- You are provided a "NULLTERM" label holding a string null terminator.
- You are provided an ASCII table later in this document that you can consult to check the ASCII value of a character.

3 Instructions

3.1 Checkpoint 1: Word Case (20 points)

Check the case of the string given. Once again, you are guaranteed that the string will be all uppercase or all lowercase. Once you've identified this, store a 1 at the label "CHKCASE" if the word is uppercase, and a 0 at the label if it is all lowercase.

3.2 Checkpoint 2: First instance (20 points)

This checkpoint is satisfied if case of the first instance of the character in the string is flipped (if it started as uppercase, it should be changed to lower, and vice-versa). If there are no instances of the character, the string remains unchanged.

3.3 Checkpoint 3: All instances (60 points)

This checkpoint is satisfied if the case of all instances of the character in the string is flipped (if it started as uppercase, it should be changed to lower, and vice-versa). If there are no instances of the character, the string remains unchanged.

3.4 Pseudocode

You will only be writing the body of this function, it is not recursive, you do not need to use the stack.

/* NOTE: Each of the variables below can be accessed through labels in tl3.asm */

```
STRING[] = "MISSISSIPPI";    // String is either ALL UPPERCASE or all lowercase
CHAR = ‘s’;                  // Character is always lowercase
CHKCASE = 0;                 // 0 for lowercase / 1 for UPPERCASE

if(STRING[0] < 97){
       CHAR -= 32;
       CHKCASE = 1;          // CHKCASE = UPPERCASE
}

for(i = 0; STRING[i] != ‘\0’; i++){
     if(STRING[i] == CHAR){
         if (CHKCASE == 1){
               STRING[i] += 32;
         } else{
               STRING[i] -= 32;
         }
     }
}
```

3.5 Example

Provided this input:

```
    STRING = "MISSISSIPPI"
    CHAR = ’s’
```

Program Output:

```
    STRING = "MIssIssIPPI"
```

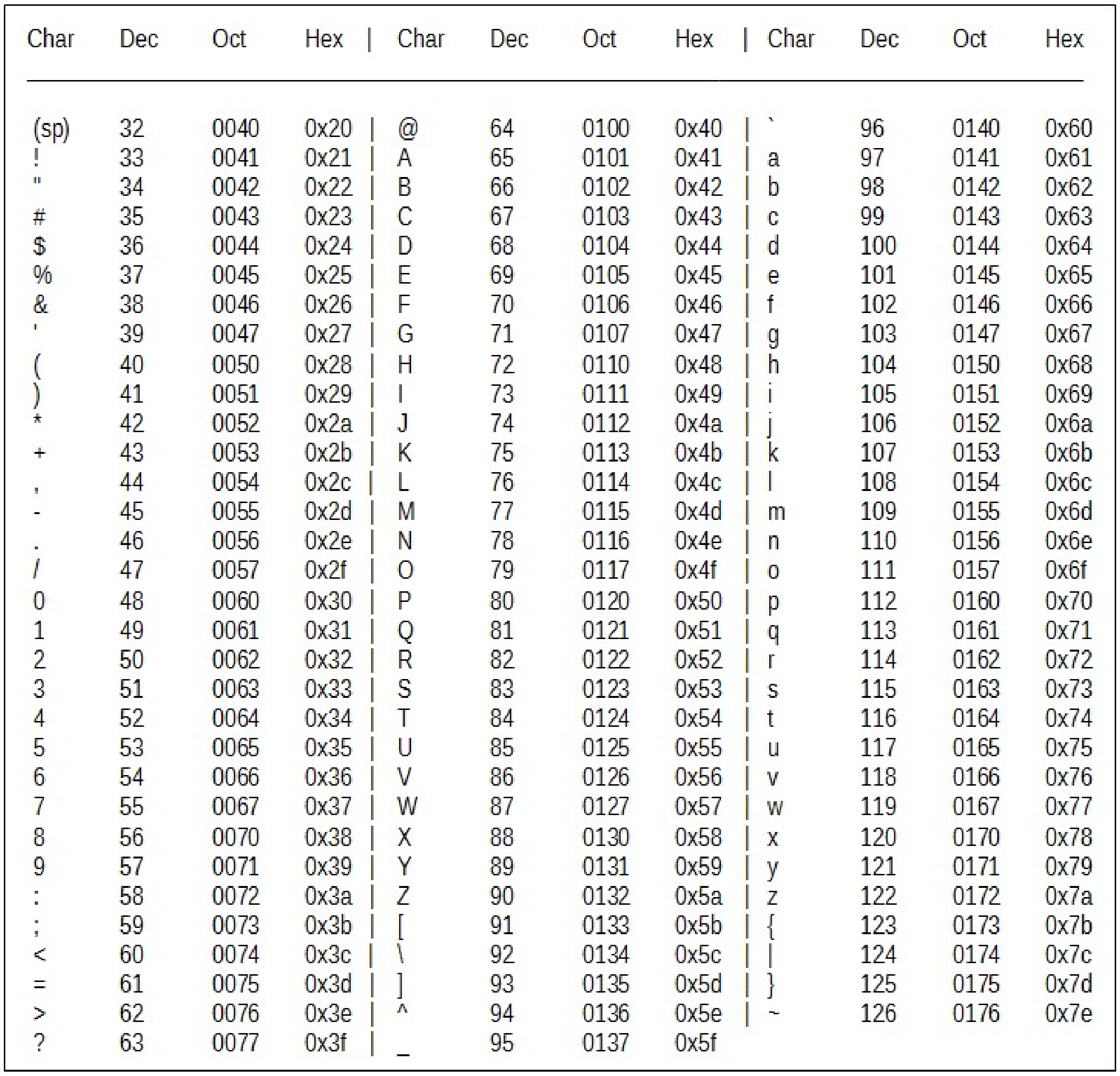
Provided this input:

```
    STRING = "apple"
    CHAR = ’p’
```

Program Output:

```
    STRING = "aPPle"
```

3.6 ASCII Table



Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Figure 1: ASCII Table — Very Cool and Useful!

4 Local Grader

To run the autograder locally, follow the steps below depending upon your operating system:

- Mac/Linux Users:
  1. Navigate to the directory your timed lab is in. **In your terminal, not in your browser**
  2. Run the command `sudo chmod +x grade.sh`
  3. Now run `./grade.sh`
- Windows Users:
  1. On **docker quickstart**, navigate to the directory your homework is in

2. Run `./grade.sh`

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.

5 Deliverables

Please upload the following files to Gradescope:

1. tl3.asm

Download and test your submission to make sure you submitted the right files

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.