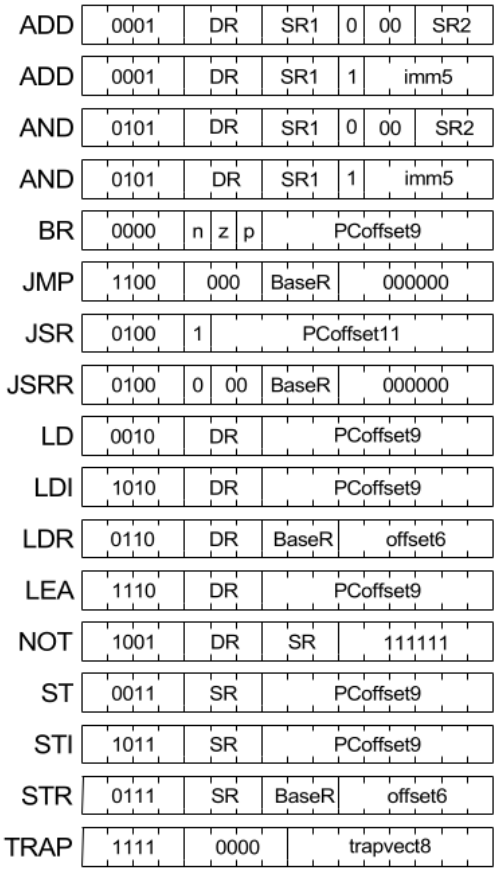
## Q1 LC-3 Reference Sheet & Assumptions

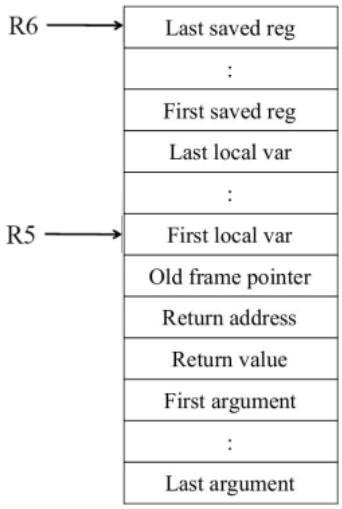
### 0 Points

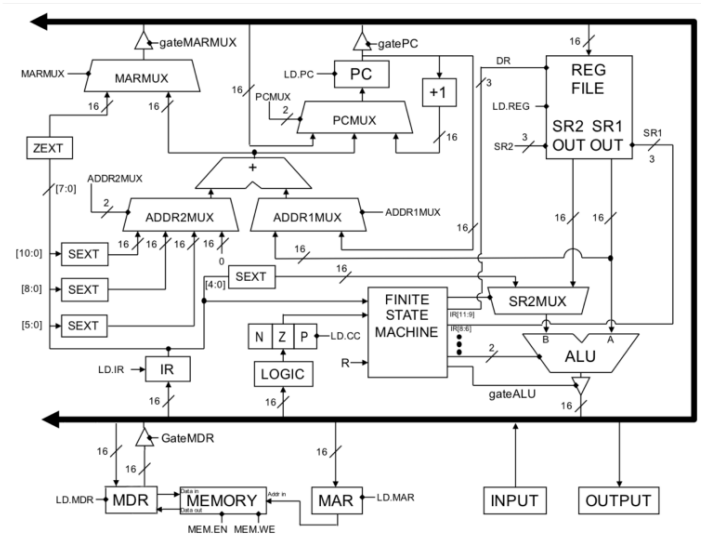
**CS2110 REFERENCE SHEET**




| *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 |

If you have to make any unstated assumptions while answering any of the questions on the quiz, let us know the question numbers and assumptions you made here. You are not required to answer this question.

Q11.1 -> There was no label, so I put the number 3 instead

## Q2
**4 Points**

If the MAR in the LC-3 was changed to hold only 11 bits, and the MDR was changed to hold only 6 bits, what would be the LC-3's new total memory, in bits? You may leave your answer un-simplified, but do not write the units.

6 * (2^11)

## Q3
**7.5 Points**

### Q3.1

**2 Points**

Select all that apply for caller responsibilities

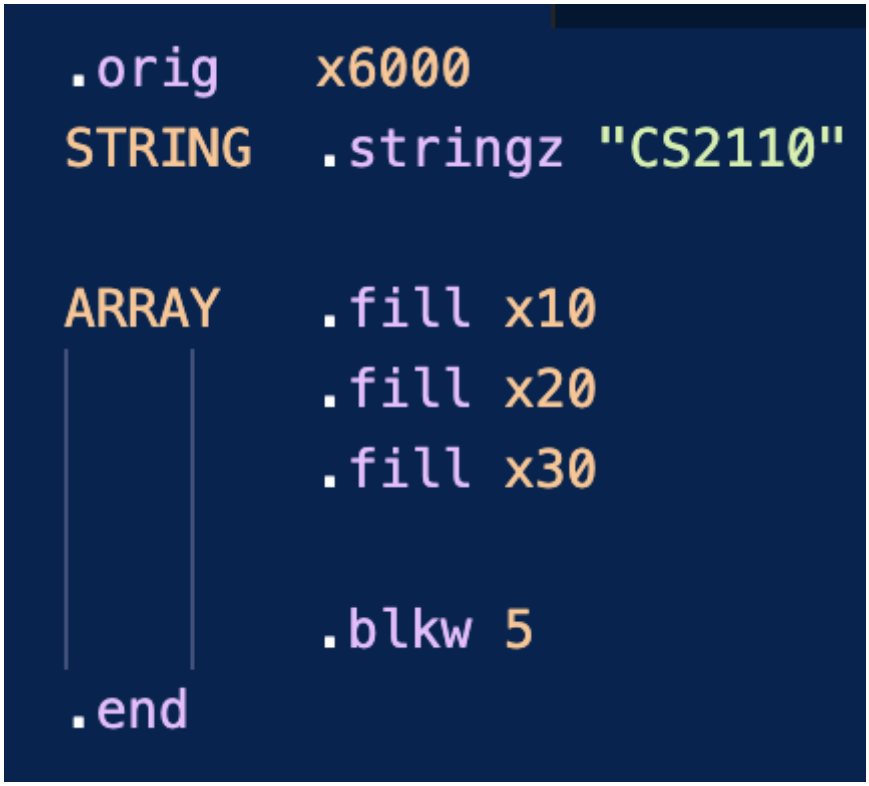
- [ ] Allocate space for Return Value
- [ ] Save copy of R7, and copy of R5
- [ ] Set R5 to be R6
- [ ] Allocate space for local variables & saved registers
- [ ] Save registers R0-R4 used by the function
- [ ] Execute the code in the function
- [x] Push arguments onto stack
- [x] Jump to subroutine
- [ ] Save the Ret Val (at R5 + 3)
- [ ] Restore saved registers (R0-R4)
- [ ] Set SP to FP, to pop off local vars and saved registers
- [x] Grab the return value
- [x] Deallocate space for return value and arguments
- [ ] Restore the Ret Addr (to R7), and old Frame Pointer (to R5)
- [ ] Pop off 3 words (ret addr, old FP, first local var)

**Q3.2**
**5.5 Points**

Select all that apply for callee responsibilities

- [x] Allocate space for Return Value
- [x] Save copy of R7, and copy of R5
- [x] Set R5 to be R6
- [x] Allocate space for local variables & saved registers
- [x] Save registers R0-R4 used by the function
- [x] Execute the code in the function
- [ ] Push arguments onto stack
- [ ] Jump to subroutine
- [x] Save the Ret Val (at R5 + 3)
- [x] Restore saved registers (R0-R4)
- [x] Set SP to FP, to pop off local vars and saved registers
- [ ] Grab the return value
- [ ] Deallocate space for return value and arguments
- [x] Restore the Ret Addr (to R7), and old Frame Pointer (to R5)
- [x] Pop off 3 words (ret addr, old FP, first local var)

**Q4**
**8 Points**




*Fill in the blanks below:*
In the code above, the pseudo-ops allocate a total of how many memory blocks?

15

The value of ARRAY[1] would be x20 and this value would be located at memory address **blank**. Express your answer in hexadecimal and do not include the '0x' prefix. (Ex: 789A)

6008

**Q5**
**4 Points**

Choose the best answer regarding the purpose of Pseudo-Ops:

The compiler treats them as special alternatives to the LD and ST instructions

They provide directions to the assembler on how to clean up the memory *after* the execution of the code

They provide early access to registers before the rest of the code gets executes

They provide directions to the assembler on how to set-up the memory *before* the execution of the code

**Q6**
**9 Points**

Read the following Assembly Code. Determine what values are stored within R1, R3, and R7 after the entire code block executes.

Assume each register holds garbage data values before execution begins.

Answer Format: All of your answers must be in 16-bit Hexadecimal. Suppose R0's value is 1, then type "0001" into the blank (without the quotations). If R0 holds garbage data, then type "garbage" (without the quotations).

Express your answer in hexadecimal and do not include the '0x' prefix. (Ex. 789A)

```
 .orig x3000
    AND R3, R3, #0
    ADD R3, R3, #1
    ADD R0, R3, R7
    LD R1, Z
    JSR SR
    ADD R7, R7, #2
    ADD R3, R3, R3
    HALT
SR   ADD R7, R7, #1
    ADD R3, R3, R3
```

```
    RET
X   .fill x302F
Z   .fill x301A
  .end
```

**Q6.1**
**3 Points**

R1's value

301A

**Q6.2**
**3 Points**

R3's value

0002

**Q6.3**
**3 Points**

R7's value

301C

## Q7
**7 Points**

In this question you have to write a program with 5 instructions maximum. You cannot use pseudo-ops or TRAPs. The template code is shown in the image below. You can only use R4-R7 as temporary registers. Do NOT touch R0-R3. If you go over 5 instruction, use pseudo-ops, TRAPs, or modify R0-R3, you will be given 0 points for this problem.

Copy the value at STRING[5] to STRING[3]

```
.orig   x3000

    ;; Your code goes HERE

.end

.orig   x3020
STRING  .stringz "NEVERGONNA"
```

```
LD R0, STRING
LDR R1, R0, 5
STR R1, R0, 3
```

**Q8**
**4 Points**

Read the following Assembly Code. Determine what value the offset **END** will take when the instruction `BRn END` is compiled to machine code.

Express your answer in hexadecimal and **do not** include the '0x' prefix. (Ex: 789A)

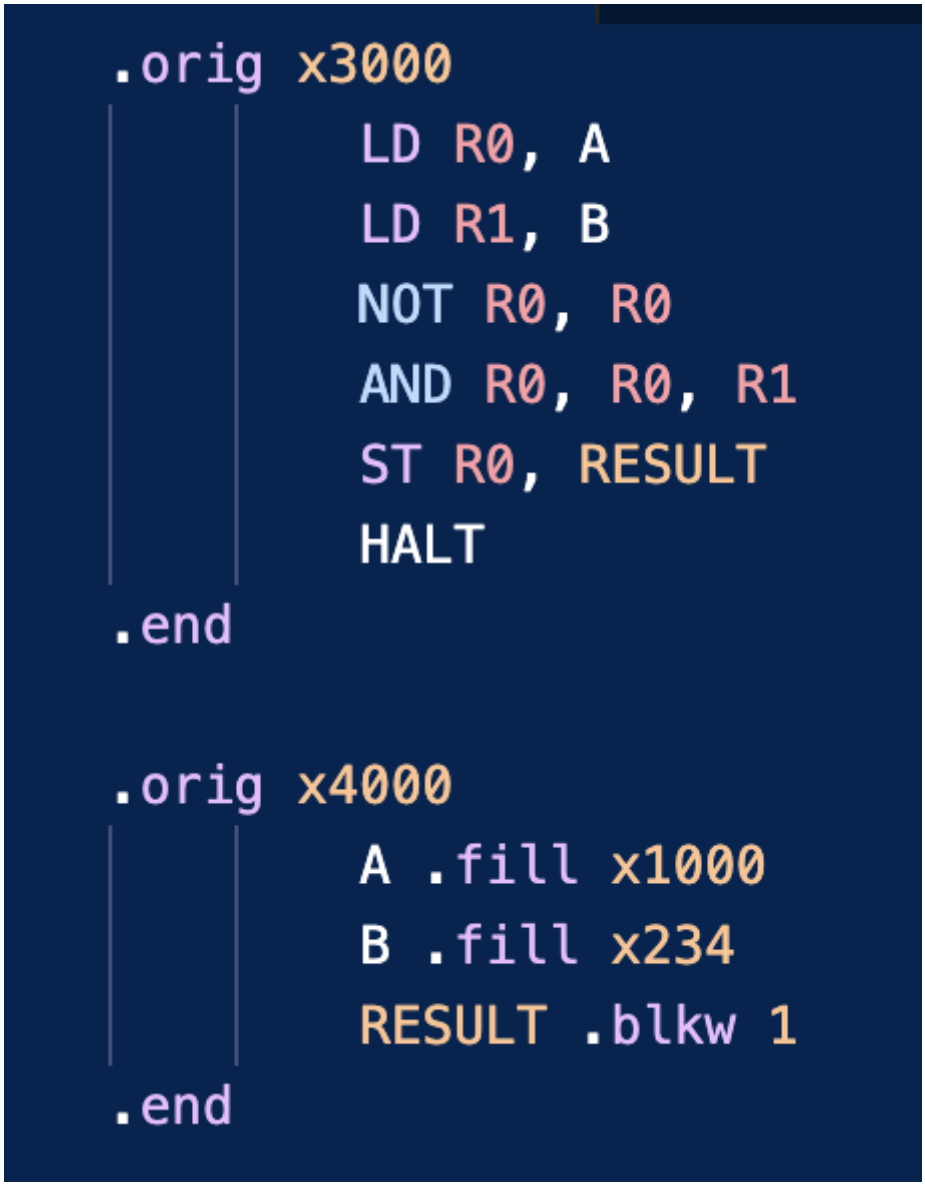
```
.orig x3000
        LDI R0, CHAR
        LD R1, OFFS
        AND R2, R2, 0
        ADD R0, R0, R1
        BRn END
        ADD R2, R0, 0
END     RET
```

```
OFFS    .fill -48
CHAR    .fill x4000
.end
```

0002

**Q9**
**5 Points**

The assembly program below will not compile. What cause this?



```
.orig x3000
        LD R0, A
        LD R1, B
        NOT R0, R0
        AND R0, R0, R1
        ST R0, RESULT
        HALT
.end

.orig x4000
        A .fill x1000
        B .fill x234
        RESULT .blkw 1
.end
```

**Q9.1**
**1 Point**

The LD/ST instructions are incorrect and do not assemble

The AND instruction is incorrect and does not assemble

The calculated result address, x1234, is not a valid address and causes a compilation error

We have two .orig pseudo-ops which causes the assembler to throw an error

**Q9.2**
**4 Points**

Please explain why the compilation error you found occurs in 1-3 sentences. Ensure you explain why the instruction, address, or pseudo-op is incorrect/invalid and do not simply restate the answer choice you selected.

The reason I chose this is that the LD and ST instructions must use an offset of 9, however, they are using values stored at x4000, and x1000 (difference between addresses x3000 and x4000) cannot be represented with 9 bits of memory. Therefore, the instructions are not configured correctly and this will not assemble.

**Q10**
**4 Points**

Jessica thinks there's a problem with popping the arguments and return value in her code. Which **single** line is the error on and what should the code be replaced with? Assume the code pushes correctly.

Answer in the format `LN. INSTRUCTION` ex. `23. AND R1, R1, R3`.

```
27   ; Push continues here ;
28   STR R1, R6, #0  ;; Pushing First Variable
29   STR R0, R6, #1  ;; Pushing Second Variable
30   STR R2, R6, #2  ;; Pushing Third Variable
31   JSR FOO
32   LDR R3, R6, #0
33   ADD R7, R7, #2
34   ADD R1, R3, #3
35   ADD R0, R1, R3
```

33. ADD R6, R6, 4

**Q11**
**8 Points**

**Q11.1**
**4 Points**

Please convert the following hexadecimal value to its corresponding assembly instruction.
**0xE603**

LEA R3, 3

**Q11.2**
**4 Points**

Convert the instruction marked by "????" on **line 7** to its corresponding hexadecimal value. Express your answer in hexadecimal and do not include the '0x' prefix. (Ex: 789A)

```
6       NOT     R1,R1
7       ADD     R1,R1,#1 ;????;
8       ADD     R0,R0,R1
9       BRzp    ENDW1
```

1261

**Q12**
**3 Points**

If you wanted to modify the **SR2** in an instruction what bit range would you modify?

Please answer in the format: [Upper:Lower], with both sides being inclusive, including the square brackets. ex. **[21:16]**

[2:0]

**Q13**
**4 Points**

Given the following memory locations and the data stored there along with the contents of provided registers, determine the state of the LC-3 after the execution of some instruction. Express your answer in hexadecimal and do not include the '0x' prefix. (Ex: 789A)

## Memory

| LABEL | ADDRESS | VALUE |
|---|---|---|
| ARRAY | x3001 | x4000 |
| VAL | x3002 | 4 |
| | ... | |
| | x4000 | 3 |
| | x4001 | -6 |
| | x4002 | 17 |
| | x4003 | x3001 |
| | x4004 | x3002 |

## Registers

| REGISTER | VALUE |
|---|---|
| R1 | x3001 |
| R2 | 6 |
| R3 | x5432 |

After executing the instruction **ST R3, VAL** what is the value at x3002?

4

**Q14**
**9 Points**

The following program should iterate over an array, take a sum of all the even values, and store the sum at label RES. Fill in the 3 missing instructions denoted by **'Blank X'** to make this code functional.

```
1   .orig x3000
2
3   LD R0, SIZE
4   LD R1, ARRAY
5   ; Blank 1 ;
6
7   BLOCK
8       ADD R0, R0, #-1
9       ; Blank 2 ;
10      LDR R2, R1, #0
11
12      AND R4, R2, #1
13      BRp KEEP
14      ADD R3, R3, R2
15
16      KEEP
17          ; Blank 3 ;
18      BRnzp BLOCK
19  SKIP
20      ST R3, RES
21      HALT
22
23  ARRAY .fill x4000
```

```
24    SIZE .fill 7
25    RES .blkw 1
26    .end
27
28    .orig x4000
29        .fill -1
30        .fill 10
31        .fill 6
32        .fill -2
33        .fill 6
34        .fill 8
35        .fill 4
36    .end
```

**Q14.1**
**3 Points**

Blank 1

AND R3, R3, 0

**Q14.2**
**3 Points**

Blank 2

BRn SKIP

**Q14.3**
**3 Points**

Blank 3

ADD R3, R3, 0

### Q15 Instruction Clock Cycles
**6 Points**

For each clock cycle, select the proper signals that will need to be used in order to execute the **LD instruction**. Each clock cycle should contain all of the relevant signals, and not contain any unnecessary signals. *If the MUX signal needed for the cycle is not listed, you can assume it is the default value of zero and do not need to select it.*

#### Q15.1 1st Clock Cycle
**2 Points**

- [ ] GateALU
- [ ] GateMDR
- [x] GateMARMUX
- [ ] ADD1MUX = SR1
- [ ] ADD2MUX = SEXT (5:0)
- [x] ADD2MUX = SEXT (8:0)
- [ ] PCMUX = ADDER
- [ ] PCMUX = BUS
- [x] MARMUX = ADDER
- [ ] ALUK = ADD
- [ ] ALUK = PASSA
- [ ] MEM.WE
- [ ] MEM.EN
- [x] LD.MAR
- [ ] LD.CC
- [ ] LD.PC
- [ ] LD.MDR
- [ ] LD.REG

**Q15.2 2nd Clock Cycle**
**2 Points**

- [ ] GateALU
- [ ] GateMDR
- [ ] GateMARMUX
- [ ] ADD1MUX = SR1
- [ ] ADD2MUX = SEXT (5:0)
- [ ] ADD2MUX = SEXT (8:0)
- [ ] PCMUX = ADDER
- [ ] PCMUX = BUS
- [ ] ALUK = ADD
- [ ] ALUK = PASSA
- [ ] MEM.WE
- [x] MEM.EN
- [ ] LD.MAR
- [ ] LD.CC
- [ ] LD.PC
- [x] LD.MDR
- [ ] LD.REG

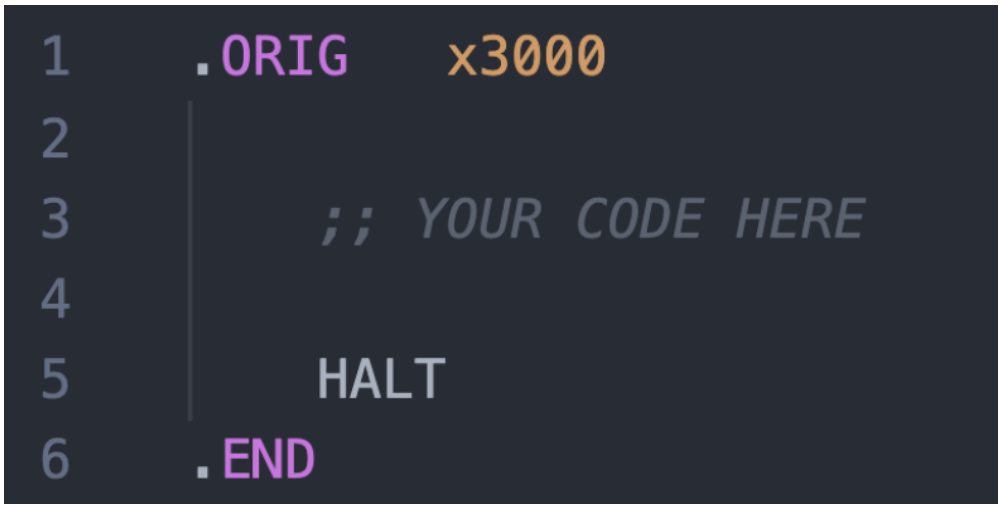
**Q15.3 3rd Clock Cycle**
**2 Points**

- [ ] GateALU
- [x] GateMDR
- [ ] GateMARMUX
- [ ] ADD1MUX = SR1
- [ ] ADD2MUX = SEXT (5:0)
- [ ] ADD2MUX = SEXT (8:0)
- [ ] PCMUX = ADDER
- [ ] PCMUX = BUS
- [ ] ALUK = ADD
- [ ] ALUK = PASSA
- [ ] MEM.WE
- [ ] MEM.EN
- [ ] LD.MAR
- [x] LD.CC
- [ ] LD.PC
- [ ] LD.MDR
- [x] LD.REG

## Q16
**7 Points**

In this question, you have to write a program with 5 instructions maximum. The template code is shown in the image below. You cannot use any extra pseudo-ops, TRAPS. Also, you may only use register R3, but not any others.

Multiply the value in R3 by 16 and make sure the final answer is in R3.



```
1    .ORIG   x3000
2
3        ;; YOUR CODE HERE
4
5        HALT
6    .END
```

```
ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
```

## Q17
**10.5 Points**

Write a program which will iterate over a null-terminated string, replacing every alternating character in the string with a space. You may assume that the string is non-empty.

For example the string "GoodLuck" would become " o d u k".

Rules:

- You cannot use more than 20 instructions

- You can use all 8 registers from R0 to R7
- This can be done in ~15 lines
- No additional pseudo-ops or TRAPs are allowed
- Your code must work generally, and not be specific to the provided test case

Pseudocode:

```
i = 0;
while (i < length) {
   str[i] = ' ';
   i = i + 2
}
```

A skeleton program is provided below:

```
.orig x3000


;;YOUR CODE HERE

HALT

SPACE .fill 32 ;; The ASCII value of space
STRING .stringz "HelloWorld" ;; The string provided
LENGTH .fill 10 ;; the length of the string

.end
```

```
AND R0, R0, 0
LD R1, LENGTH
LD R5, SPACE
NOT R2, R1
ADD R2, R2, 1
WHILE
LD R4, STRING
ADD R3, R0, R2
BRzp ENDWHILE
ADD R4, R4, R0
```

```
STR R5, R4, 0
ADD R0, R0, 2
BR WHILE
ENDWHILE
```

## Quiz 3C

● Graded

**Student**
Vidit Dharmendra Pokharna

**Total Points**
**75 / 100 pts**

**Question 1**
LC-3 Reference Sheet & Assumptions — **0** / 0 pts

**Question 2**
(no title) — **4** / 4 pts

**Question 3**
(no title) — **7.5** / 7.5 pts

3.1 (no title) — **2** / 2 pts

3.2 (no title) — **5.5** / 5.5 pts

**Question 4**
(no title) — **8** / 8 pts

**Question 5**
(no title) — **4** / 4 pts

**Question 6**
(no title) — **3** / 9 pts

6.1 (no title) — **3** / 3 pts

6.2 (no title) — **0** / 3 pts

6.3 (no title) **0** / 3 pts

**Question 7**

(no title) **0** / 7 pts

**Question 8**

(no title) **0** / 4 pts

**Question 9**

(no title) **5** / 5 pts

9.1 (no title) **1** / 1 pt

9.2 (no title) **4** / 4 pts

**Question 10**

(no title) **4** / 4 pts

**Question 11**

(no title) **8** / 8 pts

11.1 (no title) **4** / 4 pts

11.2 (no title) **4** / 4 pts

**Question 12**

(no title) **3** / 3 pts

**Question 13**

(no title) R **0** / 4 pts

**Question 14**

(no title) **6** / 9 pts

14.1 (no title) **3** / 3 pts

14.2 (no title) **3** / 3 pts

14.3 (no title) **0** / 3 pts

**Question 15**

Instruction Clock Cycles **6** / 6 pts

15.1 1st Clock Cycle **2** / 2 pts

15.2 2nd Clock Cycle **2** / 2 pts

15.3 3rd Clock Cycle **2** / 2 pts

**Question 16**

(no title) **7** / 7 pts

**Question 17**

(no title)

**9.5** / 10.5 pts