

This information might be useful. Unless specified otherwise, assume that instructions in memory start at 0x3000. Data you might reference starts at 0x4000, and the stack if you need it is at 0x5000. Each question is worth 5 points.

|      | 15 | 14 | 13 | 12 | 11             | 10         | 9 | 8         | 7         | 6 | 5       | 4    | 3 | 2   | 1 | 0 |
|------|----|----|----|----|----------------|------------|---|-----------|-----------|---|---------|------|---|-----|---|---|
| ADD  | 0  | 0  | 0  | 1  | DR             |            |   | SR1       |           |   | 0       | 0    | 0 | SR2 |   |   |
| ADD  | 0  | 0  | 0  | 1  | DR             |            |   | SR1       |           |   | 1       | imm5 |   |     |   |   |
| AND  | 0  | 1  | 0  | 1  | DR             |            |   | SR1       |           |   | 0       | 0    | 0 | SR2 |   |   |
| AND  | 0  | 1  | 0  | 1  | DR             |            |   | SR1       |           |   | 1       | imm5 |   |     |   |   |
| BR   | 0  | 0  | 0  | 0  | n              | z          | p | PCoffset9 |           |   |         |      |   |     |   |   |
| LD   | 0  | 0  | 1  | 0  | DR             |            |   | PCoffset9 |           |   |         |      |   |     |   |   |
| LDI  | 1  | 0  | 1  | 0  | DR             |            |   | PCoffset9 |           |   |         |      |   |     |   |   |
| LDR  | 0  | 1  | 1  | 0  | DR             |            |   | BaseR     |           |   | offset6 |      |   |     |   |   |
| ST   | 0  | 0  | 1  | 1  | DR             |            |   | PCoffset9 |           |   |         |      |   |     |   |   |
| STI  | 1  | 0  | 1  | 1  | SR             |            |   | PCoffset9 |           |   |         |      |   |     |   |   |
| STR  | 0  | 1  | 1  | 1  | SR             |            |   | BaseR     |           |   | offset6 |      |   |     |   |   |
| LEA  | 1  | 1  | 1  | 0  | SR             |            |   | PCoffset9 |           |   |         |      |   |     |   |   |
| JSR  | 0  | 1  | 0  | 0  | 1              | PCoffset11 |   |           |           |   |         |      |   |     |   |   |
| RET  | 1  | 1  | 0  | 0  | 000 111 000000 |            |   |           |           |   |         |      |   |     |   |   |
| TRAP | 1  | 1  | 1  | 1  | 0              | 0          | 0 | 0         | trapvect8 |   |         |      |   |     |   |   |

1. What does the LC3 instruction 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 0 do?

2. Write the bit pattern for the LC3 instruction to add R5 to R3, storing the result in R4.

Name:

CS210 Exam 3, April 2008

3. Convert to assembly, and briefly explain the instruction  
0010010110101111
4. Convert to assembly, and briefly explain the instruction  
0110010110101111
5. Convert to assembly, and briefly explain the instruction  
1010010110101111
6. Write the bit pattern for an LC3 instruction which causes the program counter to jump forward 7 locations if the “negative” status register is set. The instruction is located at 0x3000, and we want to start executing at 0x3007.
7. Convert the following code to binary machine code. The code starts at memory location 0x3000.  
LEA R4, DATA  
ADD R3, R2, R1  
ADD R3, #6  
DATA .FILL 25
8. Describe what happens during a BR instruction.
9. Using a conditional branch instruction, how far away from the current program counter can you jump?

10. Write a few lines of LC3 assembly language that results in  $R0 = R1 - R2$ .

11. For the LC3, what causes the Z,P, and N status registers to change? When are they altered?

12. In C (and most programming languages), you can do something like this:

```
if (a > 4)
```

```
    a = a + 1;
```

Assume the value of a is in register 0 of the LC3. Translate the code above into LC3 assembly code (your code will need two adds, a branch, and some labels).

13. What numbers are in registers R1, R2, and R3 when the following code finishes?

```
        LD  R1, A
        LD  R2, B
        AND R3, R3, #0
        ADD R3, R3, #5
X       ADD R3, R3, R2
        ADD R1, R1, #-1
        BRp X
        HALT
A       .FILL 5
B       .FILL 7
```

14. Each of the instructions causes values in registers and memory to be moved around (the finite state machine is directing all these things). What instruction would cause the value in **R7** to be moved into **PC**?

15. Write LC3 assembly code for both PUSH and POP, using register R0.
16. When you enter a subroutine, you might want to push a different register on the stack. Which one would you probably push?
17. Assume you have a subroutine called MULT that multiplies R1 and R2, leaving the result in R3, and that you're using R6 as your stack pointer. Write an LC3 assembly routine for SQUARE, which copies R1 into R2, calls MULT (using a JSR), and then returns. This question is really checking to make sure you know what to do with the stack. Write LC3 assembly code, not pseudocode.
18. There's an LC3 instruction that does  $\text{mem}[\text{BaseR} + \text{SEXT}(\text{offset6})] = \text{SR}$ . Which one is it?
19. From the LC3 diagram (1 point each), what are the following abbreviations
- PC
  - ALU
  - FSM
  - MDR
  - MAR
20. True or false (1 pt each)
- The lab kit uses an Atmel microprocessor
  - LEDs can be inserted in only one direction
  - There are 28 pins on the microprocessor
  - There were two different blinking patterns on the pre-programmed chip
  - Jason is looking scruffy, and needs a haircut