**CSCI240 – Computer Organization and Assembly Language Programming**

**Name:**  Jonathan Limpus

**Student ID:** 1429394

**Assignment:** Homework: The LC-3

6. a. AND R3, R2, #4 -> 0101 011 010 1 00100

b. AND R3, R2, #12 -> 0101 011 010 1 01100

c. AND R3, R2, #31 -> 0101 011 010 1 11111

d. There is a problem here because there is only 5 bits in the AND instruction

7. 11111, which is decimal 15.

8.It would require 4 bits to represent 32 registers in the LC-3 instructions, which would cause a problem due to how many bits are required to represent the entire instruction.

9.0001 001 001 1 00000 (Add register 1 by zero and store result in register 1) and 0000 000 00000000 (Break with no n, z, or p flag set) both do nothing without effecting the program. Unlike the others, the ADD instruction actually performs a calculation on operands.

10.

A. 0000111101010101 (BR\_nzp offset -171)

B. 0100111101010101 (JSR offset -171)

Both of these set the PC to -171. The difference between to two is that JSR saves information in R7.

11. No. There is no instruction to subtract in LC-3. This operation would require multiple steps to take the two’s complement version of the number and add the numbers together to perform subtraction.

13. a. Use the add instruction to add the value in R2 with 0 and store the result in R3

ADD R3 R2 #0 -> 0001011010100000

b. Not R3, Add 1 to R3, Add R2 and R3 and store result in R1

NOT R3, R3 -> 1001011011111111

ADD R3, R3, 1 -> 0001011011100001

ADD R1, R2, R3 -> 0010 001 010 0 00011

c. Add the value of a register and 0 and store the result back into a register

d. There is no sequence that could cause that condition, because the contents of a register can’t be both zero and negative.

e. AND R2, R2, 0 -> 0101010010100000

16. a. It would be best to use the LD instruction, as it allows you to offset to a location within 9 bits (29 – 1)

b. You would want to add a specific address to a Register, and use LDR to load from that register

c. You would want to use the LD instruction, and add 1 to the instruction code for every address

34. The elements that implement NOT are: IR, Reg File, NZP, and ALU

35. The elements that implement ADD are: IR, Reg File, SEXT unit, SR2MUX, NZP, and ALU

36. The elements that implement LD are: IR, Memory, MDR, MAR, IR, PC, Reg File, SEXT, ADDR2MUX, ADDR1MUX, ADDER, MAXMUX, GateMARMUX, NZP

37. The elements that implement LDI are: Memory, MDR, MAR, IR, PC, Reg File, SEXT, ADDR1MUX, ADDR2MUX, ADDER, MARMUX, GateMARMUX, NZP

38. The elements that implement LDR are: Memory, MDR, MAR, IR, Reg File, SEXT, ADDR1MUX, ADDR2MUX, ADDER, MAXMUX, GateMARMUX, NZP

39. The elements that implement LEA are: Memory, IR, PC, Reg File, SEXT, ADDR1MUX, ADDR2MUX, ADDER, MARMUX, GateMARMUX, NZP