NCSU ECE 109 Sections 602 and 603 and UNCA CSCI 255.001
Exam 3 Spring 2009 Solution
21 April, 2009

Problem 2 (6 points) Loopy code
Examine the following LC-3 code. Briefly explain what it does when it executes. If it gets into an "infinite" loop (that is, it never reaches the HALT), explain why this happens. If it reaches the HALT, state the final values of the registers when the program halts.

```
         AND   R4,R4,#0
LOOP     AND   R4,R4,#5
         AND   R5,R4,#2
         BRz   LOOP
         HALT
```

The first instruction (AND R4,R4,#0) sets R4 to 0. The second instruction (AND R4,R4,#5) sets R4 to 5. The third (AND R5,R4,#2) AND's R4 and the literal 2 and stores the result, 0, into R5. Because the result is 0, the fourth instruction will branch back to the second instruction.

Now let's look at the next time through the loop. On the second instruction, 5 is added to R4, giving it the value 10. The third instruction AND's R5 and R4. This time the result is 2, which is stored in R5. Since the result isn't zero, the branch of the fourth instruction isn't taken, and the program proceeds to the HALT with R4 equal to 10 and R5 equal to 2.

Problem 3 (4 points) Memory-mapped I/O
Write a *short* section of LC-3 code that uses the LC-3 device registers to write the value stored in register R2 to the display. No calls to TRAP routines are allowed on this question.

```
POOLLP   LDI   R0,DSR
         BRzp  POOLLP
         STI   R2,DDR
..................
DSR      .FILL xFE04
DDR      .FILL xFD06
```

Problem 4 (4 points) TRAP routines
Write a *very short* section of LC-3 code that uses TRAP routines to write the character stored in register R2 to the display. No reading or writing of device registers is allowed in answers to this question.

```
         ADD  R0,R2,#0
         OUT
```

Problem 5 (8 points) A few calculations
Write short pieces of LC-3 code to solve the following problems. For full credit, do not modify any LC-3 registers other than R2. Neither of these should require more than three lines of code.

Set R2 to 8*R5 **ADD R2,R5,R5** **ADD R2,R2,R2** **ADD R2,R2,R2**
Subtract R0 from R1 and store the result in R2 **NOT R2,R0** **ADD R2,R2,#1** **ADD R2,R2,R1**

Problem 6 (6 points) Memory
A computer memory has 24-bit words stored in 64 k locations. What is the size of this memory in bits?

24 × 64 k or 1536 k

How many address bits are needed to address the 64 k words of this memory?

Since 64 k is $2^6 \times 2^{10}$ or 2^{16}, 16 bits are required

Problem 7 (24 points) Hand assembled
Use the symbol table shown below in this question.

BUNCOMBE	x3307
HAYWOOD	x3367
MACON	x33C7
MADISON	x3447

Write the appropriate 16-bit LC-3 machine language word, in binary or hex, for each assembly language statement shown in the left column of the table below. Assume that the instruction is located at address x3302 in all cases. If the assembly language statement is illegal, state the reason why.

ADD R0,R2,#-16	**0001 000 010 1 10000**
ADD R7,R7,MACON	**Illegal, the third operand for ADD must be either a register or immediate value.**
AND R2,R3,#16	**Illegal, 16 is outside the range of allowed values (5 bit two's complement)**
BR MADISON	**Illegal, MADISON is too far away to be a 9-bit offset**
BRz BUNCOMBE	**000 010 000000100**

`JSR MADISON`	**`0100 1 00101000100`**
`LD R0,R5`	**Illegal, second operation to `LD` must be an address**
`LDR R2,R3,#20`	**`0110 010 011 010100`**
`LEA R2,R3,#20`	**Illegal, `LEA` only takes two arguments**
`SUB R3,R4,R5`	**Illegal, no `SUB` instruction**
`STI R3,HAYWOOD`	**`1011 011 001100100`**
`STR R3,R4,x20`	**Illegal, `x20` is outside the range of allowed values (6 bit two's complement)**

Problem 8 (16 points)

Assume that the eight LC/3 registers have the values shown on the left below and that the eight words of memory starting at memory location x3220 have the values shown on the right.

Register	***Value***
R0	x0000
R1	x0000
R2	x2222
R3	x3333
R4	x0000
R5	x0000
R6	x6666
R7	x0000

Address	***Value***
x3220	x0000
x3221	x0000
x3222	x2121
x3223	x3131
x3224	x0000
x3225	x5151
x3226	x0000
x3227	x0000

For the eight addresses shown below, write a single LC/3 instruction to load the value **stored in** the specified memory location into register `R1`. (For example, when x3222 is specified, x2121 should be stored in `R1`.) Assume that each instruction is located at memory address x3200.

If this location cannot be loaded in one instruction, state why this is not possible.

x0011	**`LDR R1,R0,x11`**
x2121	**`LDI R1,x21`**
x2222	**`LDR R1,R2,#0`**
x3200	**`LD R1,#-1`**
x3284	**`LD R1,x83`**

x3384	**Can't be done. Nearest register is x51 away. PC is x183 away.**
X6656	**LDR R1,R6,#-16**
x6676	**LDR R1,R6,#16**

Problem 9 (20 points) Assignment 4 Revisited

Write an LC-3 *subroutine* that performs the following task:

```
If R0 is an ASCII digit ('0' to '9' – ASCII x30 to x39)
        Set R1 to 1 and return
If R0 contains a dollar sign (ASCII x24),
        Set R1 to 2 and return
Otherwise
        Set R1 to 3 and return
```

```
Prob9     LD      R1,Masc0
          ADD     R1,R0,R1
          BRn     LT0
;; arrive here only if R0 >= '0'
          LD      R1,Masc9
          ADD     R1,R0,R1
          BRp     Ret3
;; arrive here only if R0 is a digit
Ret1      AND     R1,R1,#0
          ADD     R1,R1,#1
          RET
;; arrive here only if R0 < '0'
LT0       LD      R1,Mdolsgn
          BRnp    Ret3
;; arrive here only if R0 = '$'
Ret2      AND     R1,R1,#0
          ADD     R1,R1,#2
          RET
Ret3      AND     R1,R1,#0
          ADD     R1,R1,#3
          RET

Masc0     .FILL   #-48    ; -'0'
Masc9     .FILL   #-57    ; -'9'
Mdolsgn   .FILL   #-36    ; -'$'
```