NCSU ECE 109 Sections 602 and 603 and UNCA CSCI 255.001
Exam 4 Spring 2009 Solution
5 May, 2009

Problem 1 (14 points) Hand assembled
Use the symbol table shown below in this question.

DURHAM	x3401
ORANGE	x3481
WAKE	x3501

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 x3400 in all cases. If the assembly language statement is illegal, state the reason why.

ADD R1,R2,x10	**Illegal, x10 (16) is too big to be a five bit two's complement immediate operation.**
ADD R3,R4,WAKE	**Illegal, ADD cannot be given a memory location as an operand field.**
BR DURHAM	**0000 111 000000000**
BRz ORANGE	**0000 010 010000000**
LD R5,WAKE	**Illegal, PC offset of WAKE (x100) is too big for the nine-bit two's complement field.**
JSRR DURHAM	**Illegal, JSRR can only take a register as argument.**
STI R3,ORANGE	**1011 011 010000000**

Problem 2 (10 points) Addressing modes
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 x3040 have the values shown on the right.

Register	***Value***
R0	x0000
R1	x0000
R2	x2222
R3	x3333
R4	x0000
R5	x5555
R6	x0000
R7	x0000

Address	***Value***
x3040	x0000
x3041	x0000
x3042	x4444
x3043	x6666
x3044	x0000
x3045	xAAAA
x3046	x0000
x3047	x0000

For the five addresses shown below, write a single LC/3 instruction to load the value **stored in** the specified memory location into register 4. Assume that each instruction is located at memory address `x3001`.

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

x3042	**LD R4,x41**
x3101	**LD R4,xFF**
x4455	**Cannot be loaded.**
x5535	**LDR R4,R5,#-32**
xAAAA	**LDI R4,x43**

Problem 3 (6 points) Memory

A computer memory has 16-bit words stored in 4 M locations. What is the size of this memory in bits?

16 × 4 M or 64 M

How many address bits are needed to address the 4 M words of this memory?

22, since 2^{22} is $2^2 \times 2^{20}$ or 4 M

Problem 4 (5 points)
Write a piece of LC-3 assembly code that copies the value stored in `R4` into `R5`.

`R5 ← R4`

```
ADD   R5,R4,#0
```

Note: `STR R5,R4,#0` is very, very wrong.

Problem 5 (5 points)
Write a piece of LC-3 assembly code that subtracts 32 from `R5`.

`R5 ← R5-32`

```
ADD   R5,R5,#-16
ADD   R5,R5,#-16
```

Note: `ADD R5,R5,#-32` is not a legal LC-3 instruction.

Problem 6 (5 points)
Write a piece of LC-3 assembly code that sets `R5` to `10-R3`.

```
R5 ← 10-R3
```

```
        NOT    R5,R3
        ADD    R5,R5,#11
```

Note: `ADD R5,R5,#11` in place of `ADD R5,R5,#1` followed by `ADD R5,R5,#10`

Problem 7 (5 points)
Write a piece of LC-3 assembly code that triples the value stored in `R4`.

```
R4 ← 3*R4
```

```
        ADD    R5,R4,R4
        ADD    R4,R5,R4
```

Note: This one really does need an extra register (or memory location).

Problem 8 (5 points)
Write a piece of LC-3 assembly code that prints the letter '`T`' (ASCII `x4A`) on the display.

```
        LD     R0,AscT
        OUT
        ..........
AscT    .FILL x4A
```

Problem 9 (5 points)
Write a piece of LC-3 assembly code that adds the contents of memory locations `x4444` and `x5555` and stores the result in memory location `x9999`.

```
        LDI    R4,M4444   ;; R4 <- M[x4444]
        LDI    R5,M5555   ;; R5 <- M[x5555]
        ADD    R5,R5,R4   ;; R5 <- M[x4444] + M[x5555]
        STI    R5,M9999
        ..........
M4444   .FILL x4444
M5555   .FILL x5555
M9999   .FILL x9999
```

Note: Yes, it is that boring

Problem 10 (10 points)

Write a piece of LC-3 assembly code that sets `R4` to be equal to bits 0 to 4 of `R3` sign-extended to a full 16-bit value. For example, if `R3` is `x000F`, `R4` should be set to `x000F`; and, if `R3` is `x0010`, `R4` should be set to `xFFF0`. (This is the same extension that occurs with the five-bit immediate operand fields of the `ADD` and `AND` instructions.)

```
        AND   R4,R3,xF      ;; copy in bits 0 to 3
        LD    R5,Bit4       ;; test bit 4
        AND   R5,R5,R3
        BRz   DONE          ;; if bit 4 set
        LD    R5,SgnExt
        ADD   R4,R4,R5      ;;   then extend with 1's
DONE    ......................
        ......................
Bit4    .FILL x0010
SgnExt  .FILL xFFF0
```

Problem 11 (15 points)

Write a piece of LC-3 assembly code that sets `R4` depending on the value stored in `R3`.

1) If `R3` is positive or zero, set `R4` to 0.
2) If `R3` is `xBEEF`, set `R4` to 1.
3) If `R3` is not `xBEEF` and is neither positive nor zer0, set `R4` to 2.

```
        AND   R4,R4,#0      ;; R4 <- 0
        ADD   R3,R3,#0      ;; if R3 is zero of positive
        BRzp  DONE          ;;   then done
        ADD   R4,R4,#1      ;; R4 <- 1
        LD    R5,NegBF      ;; if R3 is xBEEF
        ADD   R5,R5,R3
        BRz   DONE          ;;   then done
        ADD   R4,R4,#1      ;; R4 <- 2
DONE    ......................
        ......................
NegBf   .FILL x4111         ;; ~xBEFF+1 = x4110+1
```

Problem 12 (15 points)
Write an LC-3 subroutine that receives in register `R0` the starting address of an eight-word "packet" of values. The LC-3 subroutine should add those eight words and return the result of that addition in register `R1`.

```
        AND   R1,R1,#0      ;; R1 <- 0
        ADD   R4,R1,#8      ;; R4 <- 8
LOOP    LDR   R3,R0,#0      ;; R3 <- Mem[R0]
        ADD   R1,R1,R3      ;; add next word to R1
        ADD   R0,R0,#1      ;; move R0 to following word
        ADD   R4,R4,#-1
        BRp   LOOP          ;; do loop eight times
        RET
```

Note: This routines modifies `R3` and `R4`