ECE 109 Sections 602 to 605
Exam 2 Fall 2007 Solution
6 November, 2007

Problem 1 (15 points) Data Path
In the table below, the columns correspond to two LC/3 instructions and the rows to the six phases of the LC/3 instruction cycle as described in the textbook. Within this table describe how the **PC, MAR, MDR, and register files** of the LC/3 datapath are used or modified in each instruction cycle phase for the two instructions. [The FETCH and DECODE rows only have one cell, since these two phases act similarly for all instructions.]

Rubric and common problems:
It's most important to know how this internal registered are used and in what order that are accessed and modified rather than matching actions with specific phases. The detailed answer below comes from a careful reading of section 4.3.2 and particularly Example 4.4.
The grading was complex. There are 10 "actions" performed in the instruction and there are 3 no-action cells. 1.5 points was given for identifying each of these. Incorrect statement were marked, but generally ignored in grading though points were sometimes deducted for serious errors. Of course, no answer received more than 15 points and no answer less than 0 points.

	`ADD R3, R4, #5`	`LDR R3, R4, #5`
FETCH	**The PC is moved into the MAR (memory address register) while the PC is "simultaneously" incremented. The memory then reads a value (the next instruction) into the MDR (memory data register).**	
DECODE	**No action on targeted registers on this phase.**	
EVALUATE ADDRESS	**Skipped on ADD.**	**R4 is added 5.**
FETCH OPERANDS	**R4 is obtained from the register file.**	**The sum of R4 and 5 is sent to the MAR. The memory reads a data value into MDR.**
EXECUTE	**The ALU adds R4 and 5.**	**Skipped on LDR.**
STORE RESULT	**Result is stored in R3.**	**Result is stored in R3.**

Problem 2 (15 points) Memories

Using the symbol table shown below

BASIE	x3442
CARMICHAEL	x3462
DORSEY	x3482
ELLINGTON	x34A2

write the appropriate 16-bit LC-3 machine language word, in binary or hex, for each assembly language statement shown in the left column. Assume that the instruction is located at address x3400 in all cases. If the assembly language statement is illegal, state the reason why this is so.

Rubric and common problems:

Generally 0.5 points were taken off for each mistake.

Many people did unnecessary translations from hexadecimal to decimal and back. For example, if the target address is x3462 and the PC is x3401, the offset is x61. Subtractions like that can be made directly in hexadecimal. The hexadecimal can then be translated into binary as 0 0110 0001 without conversion to decimal. One common, and time consuming, problem was inappropriately treating 61 as a decimal number and translating it into the binary number 111101.

On the last problem, some people pointed out that the standard LC/3 does not support a trap with number x55.

ADD R0,R2,#12	**0001 000 010 1 01100**
AND R7,R7,x12	**18 (x12) too big for signed 5 bits**
AND R7,R7,R7	**0101 111 111 000 111**
BRnp DORSEY	**0000 101 010000001**
BRpz BASIE	**BRpz is not a valid opcode**
LD R3,CARMICHAEL	**0010 011 001100001**
LEA R2,ELLINGTON	**1110 010 010100001**
NOT R5,R6	**1001 101 110 111111**
STI R5,BASIE	**1011 101 001000001**
STR R3,R4,x14	**0111 011 100 010100**
TRAP x55	**1111 0000 0101 0101**

Problem 3 (15 points)
The binary program shown in the left column below is loaded into memory at location x3000. In the right column, write the LC/3 assembly instructions or appropriate psuedo-ops corresponding to this program. Be sure to include appropriate labels and `.ORIG` and `.END` statements.

Rubric and common problems:

Many people did not use labels and had answers like:

```
      LDI    R4,x3007
```

for the second instruction. This is not a legal LC/3 instruction. Both of the following are correct LC/3 assembly language instructions:

```
      LDI    R4,#5
      LDI    R4,x5
```

even though they are difficult to understand. By the way

```
      LDI    R4,x3006
```

is even a bit more incorrect, as it doesn't take into account that the offset is added to the PC, which is one more than the address of the current instruction.

Generally 0.5 points were deducted for each mistake.

Binary	Assembly
	.ORIG x3000
0101000000100000	AND R0,R0,#0
1010100000000101	LDI R4,Lable07
0000011000000001	Lable02 BRzp Lable04
0001000000100001	ADD R0,R0,#1
0001100100000100	Lable04 ADD R4,R4,R4
0000101111111100	BRnp Lable02
1111000000100101	HALT
0100000000000000	Lable07 .FILL x4000
	.END

Problem 4 (15 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 x3020 have the values shown on the right.

Register	*Value*
R0	x0000
R1	x0001
R2	x0002
R3	x0003
R4	x0004
R5	x0005
R6	x4444
R7	x5555

Address	*Value*
x3020	x0000
x3021	x0001
x3022	x0002
x3023	x0003
x3024	x0004
x3025	x0005
x3026	x6666
x3027	x7777

For ***six** of the following **seven unanswered** cases* shown below, write either a single LC/3 instruction or a series of two LC/3 instructions to load the value stored in the specified memory location into register 5. Assume that each instruction is located at memory address x3010.

Only three of the seven require the use of two instructions. Because I'm only grading six of the seven, you can miss one without penalty. In the difficult cases, you'll do well to give an explanation of your strategy.

Rubric and common problems:
The point of this question was to test facility with LD, LDI, and LDR and knowledge of the restrictions imposed by the size of instruction bit fields. The question really should have prohibited .FILL's, since they make it a bit too easy. (Also, that x3111 was supposed to be x3101.)
2.5 points, with liberal partial credit, were given to each correct answer for the best six of seven answers.

x3021	LD R5,x10
x3111	**LEA R5,xF0** **LDR R5,R5,x10**
x4424	**LDR R5,R6,#-32**
x4444	**LDR R5,R6,#0**
x4464	**ADD R5,R6,x10** **LDR R5,R5,x10**
x6666	**LDI R5,x15**
x6667	**LD R5,x15** **LDR R5,R5,#1**
x8888	**ADD R5,R4,R4** **LDR R5,R5,#0**

Problem 5 (40 points)
In this long question of many parts, write little (many only two or three instructions long) LC/3 programs to solve the following small problems. Answers that are unnecessary long or complicated will not receive full credit.

3 points
Some people forgot set R3 to 0 before adding in five

Write LC/3 code to set R3 to 5.

```
        AND     R3,R3,#0
    ADD   R3,R3,#5
```

3 points
0.5 point deducted for unnecessary load from memory (.FILL)

Write LC/3 code to turn "off" bits 3 to 0 of register R2. For example, if R2 contains x8ADE, it should be set to x8AD0. In other words, "and" R2 with xFFF0.

```
        AND     R2,R2,xFFF0
```

5 points
Many people needlessly changed R3

Write LC/3 code to set R5 from R3, according to the following formula:

```
    R5 = 3*R3 + 1
        ADD     R5,R3,R3
        ADD     R5,R5,R3
    ADD   R5,R5,#1
```

5 points

Write LC/3 code to subtract R3 from R4. The result should be stored in R5. This is like computing the following equation:

```
    R5 = R4-R3
        NOT     R5,R3
        ADD     R5,R5,#1
        ADD     R5,R5,R4
```

8 points
The solution below is certainly not the obvious one. The typical solution was

```
        ADD     R4,R4,#0
        BRn     VNEG
        ADD     R5,R4,#0
        BR      DONE
VNEG    AND     R5,R5,#0
DONE
```

gradeing - 2 points for each BR, 1.0 points for test of R4, and 1.5 points for each update of R5

Write LC/3 code to compare R4 to zero and to (1) set R5 to R4, if R4 is positive, or (2) set R5 to 0, if R4 is negative. (This is similar to the IRS directive: "Enter line 4 in line 5, if line 4 is a positive number. Otherwise, enter 0".)

```
        ADD     R5,R4,#0
        BRzp    R5pos           ; skip next if R4<0
        AND     R5,R5,#0
R5pos ....
```

8 points

Write LC/3 code to test if R5 contains the ASCII character for 'n' or for 'c'. If so, set R3 to contain the value 1. Otherwise, set R3 to contain 0.

```
        LD      R3,NEGn
        BRz     FndChr
        LD      R3,NEGc
        BRz     FndChr
;; No match - Set R3 to 0
        AND     R3,R3,#0
        BR      Done
;; Found a match - Set R3 to 1 (it's already 0)
FndChr  ADD     R3,R3,#1
Done    ......

NEGn    .FILL   #-110      ; ASCII for 'n' is 110
NEGc    .FILL   #-99       ; ASCII for 'c' is  99
```

8 points

(A) If R4 is greater than 0, keep doubling R4 until it is bigger than 100. If R4 is not greater than 0, don't change it.

or

(B) Add up the numbers stored in the 256 memory locations from x4000 to x40FF and store them in R2.

Answer either (A) or (B).

```
;;  (A)
        ADD    R4,R4,#0
        BRnp   Done          ; Don't change is R4<=0
LOOP    LD     R5,NEG100
        ADD    R4,R4,R4      ; Double R4
        ADD    R5,R5,R4      ; Test if R4>100
        BRnp   LOOP
Done    ....
        ....
NEG100  .FILL #-100

;;  (B)
        LD     R5,K256        ;; R5 is the countdown
        LD     R4,Kx4000      ;; R4 is the pointer
        AND    R2,R2,#0       ;; R2 is the sum
LOOP    LDR    R3,R2,#0       ;; R3 gets next number
        ADD    R2,R2,R3       ;; Add R3 into the sum
        ADD    R4,R4,#1       ;; increment pointer
        ADD    R5,R5,#-1      ;; decrement counter
```

```
        BRp   LOOP          ;; do it 255 times
        .....
K255    .FILL #256
Kx4000  .FILL x4000
```