Quiz 2 **Solution** CSCI 255 Spring 2001

9 April, 2001

Name: ______________________

This is a closed book exam. Use of calculators is also not allowed. Be sure to show your work in order to get full credit for the problem. When possible place your answers in the provided boxes. There are 9 questions for a total of 150 points on this quiz.

Problem 1 (64 points): 3:15-3:47

In this problem you are asked to write **eight** *independent* sections of LC-2 assembly code to set registers R0 or R1 or LC-2 memory locations based on constants, the present values of R3 and R4, or LC-2 memory locations. You may use registers R6 or R7 as "scratch" registers but should not modify any other registers. You must assume that your code will be located somewhere between memory locations x3000 and x30FF. You may use .fill's when needed to initial memory locations. You should assume that these .fill's would also be stored in memory locations x3000 to x30FF.

In these subproblems, the code to implement is given in the psuedo-C notation used in class lectures. R*n* will be used as a reference to LC-2 register *n*. M[*n*] will be used as a reference to LC-2 memory location *n*.

There are many possible right answers. These are probably the shortest.

R0 ← 5 * R3 ;	ADD R0,R3,R3 ADD R0,R0,R0 ADD R0,R0,R3
R0 ← R3 – R4 ;	NOT R0,R4 ADD R0,R0,#1 ADD R0,R0,R3
R0 ← R3 & R4 ;	AND R0,R3,R4

if (R3 == 15)
    R0 ← R4 ;
else
    R0 ← R4 + 1 ;
	ADD R0,R4,#0
	ADD R6,R3,#-15
	BRz DONE
	ADD R0,R0,#1
	DONE …

R0 ← R4 ;
while (R0 < 107)
    R0 ← R0 + R0 ;
	LD 6,M107
	ADD R0,R4,#0
	BR MDLOOP
	BGLOOP ADD R0,R0,R0
	MDLOOP ADD R7,R6,R0
	BRn BGLOOP
	…
	M107 .FILL #-107

M[x3100] ← M[x3100] + 5 ;		**LD**	**R6,x3010**
		ADD	**R6,R6,#5**
		ST	**R6,x3010**
M[x4100] ← M[x4100] + 5 ;		**LDI**	**R6,PTR**
		ADD	**R6,R6,#5**
		STI	**R6,PTR**
		…	
	PTR	**.FILL**	**x3010**
R0 ← R3 + 1 ;		**ADD**	**R0,R3,#1**
R1 ← R4 + ’1’ ;		**LD**	**R6,ASC1**
		ADD	**R1,R4,R6**
		…	
	ASC1	**.FILL**	**X31**

Problem 2 (16 points): 3:47-3:55

Translate into LC-2 machine language (binary) program the LC-2 assembly language program shown below:

A fairly common problem was starting the first instruction (LD) at x3001 rather than x3000.

	.ORIG	x3000	
	LD	R1,MX	**0010001000000101**
	LDI	R2,MX	**1010010000000101**
	LEA	R3,MX	**1110011000000101**
	LDR	R4,R1,#1	**0110100001000001**
	HALT		**1111000000100101**
MX	.FILL	0x3006	**0011000000000110**
MY	.FILL	0x3007	**0011000000000111**
MZ	.FILL	0x3008	**0011000000001000**
	.END		

Problem 3 (12 points): 3:55-4:01

What are the values of registers R1 to R4 after the LC-2 assembly language program in Problem 2 is executed?

R1 = **x3006**
R2 = **x3007**
R3 = **x3005**
R4 = **x3008**

If you (incorrectly) assumed that MX was located at x3006, then R1, R2, and R3 would be set to x3006 and R4 would be set to x3007.

Problem 4 (8 points): 4:01-4:05

What Linux command would you use to assemble the LC-2 assembly program `lab9.asm`? Give not only the name of the command, but the arguments you use with it.

```
lc2asm lab9.asm lab9
```

Problem 5 (10 points): 4:05-4:10

Write some LC-2 assembly code to write the contents of register 5 to the CRT using the CRT data and status registers?

```
OUTLOOP    LDI       R6, ACRTSR
           BRzp      OUTLOOP
           STI       R5, ACRTDR
           ……
ACRTSR     .FILL     xF3FC
ACRTDR     .FILL     xF3FF
```

Problem 6 (8 points): 4:10-4:14

Write some LC-2 assembly code to write the contents of register 5 to the CRT using a LC-2 trap routine?

```
           ADD       R0, R5, #0
           OUT
```

Problem 7 (12 points): 4:14-4:20

The VAX computer has an instruction called BIC (Bit Clear) that performs the logical operation $\alpha\,\beta'$. Write a LC-2 subroutine called BIC in assembly language that performs this operation on registers R0 and R1, that is, the subroutine performs the operation:

```
R0 ← R0 & ~R1 ;
```

```
BIC        NOT       R1, R1
           AND       R0, R0, R1
           NOT       R1, R1          ; restore R1
           RET
```

Problem 8 (8 points): 4:20-4:24

Show the complete LC-2 instruction needed to call the BIC subroutine of Problem 7. You may assume that both the calling and called subroutine are on the same page.

```
           JSR       BIC
```

Problem 9 (10 points): 4:24-4:30

Translate the following two LC-2 binary instructions into LC-2 assembly code.

0001011011111110	ADD R3, R3, #-2
0101011011000111	AND R3, R3, R7