UNIVERSITY OF CALIFORNIA, RIVERSIDE

Department of Computer Science and Engineering

CS61 – Machine Organization and Assembly Language

Final

August 31, 2001

20

Name: Solution Key **Student ID#:** _____

Please print legibly

(Numbers in parenthesis denote total possible points for question.)

1. Trace the following LC-2 program and determine the contents of all the general registers when the CPU reaches the halt statement. (4)

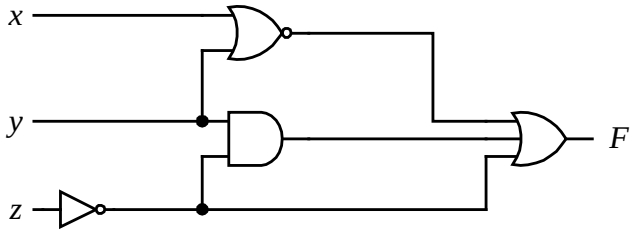
```
          .orig   x3000
          ld      r1, data1
          not     r2, r1
          ld      r3, data2
          not     r4, r3
          brp     there
          and     r5, r2, r4
          not     r6, r5
          jsr     here
there     halt
here      ld      r7, data3
          not     r3, r3
          ret

data1     .fill   xcdef
data2     .fill   x789a
data3     .fill   x3003
          .end
```

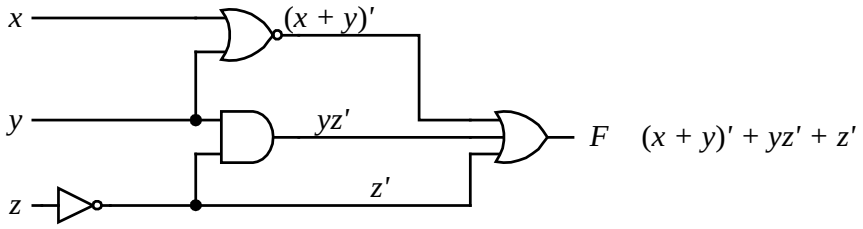
Answer

```
   R0  x0000  0          R4  x789A  30874      PC  x3008  12296
   R1  xCDEF  -12817     R5  x0200  512        IR  x0208  520
   R2  x3210  12816      R6  xFDFF  -513       CC  P
   R3  x8765  -30875     R7  x3003  12291
 x3000  0010001000001100  x220C            LD     R1, datal
 x3001  1001010001111111  x947F            NOT    R2, R1
 x3002  0010011000001101  x260D            LD     R3, data2
 x3003  1001100011111111  x98FF            NOT    R4, R3
 x3004  0000001000001000  x0208            BRP    there
 x3005  0101101010000100  x5A84            AND    R5, R2, R4
 x3006  1001110101111111  x9D7F            NOT    R6, R5
 x3007  0100100000001001  x4809            JSR    here
 x3008  1111000000100101  xF025  there     TRAP   HALT
 x3009  0010111000001110  x2E0E  here      LD     R7, data3
 x300A  1001011011111111  x96FF            NOT    R3, R3
 x300B  1101000000000000  xD000            RET
 x300C  1100110111101111  xCDEF  datal     JSRR   R7, x002F
 x300D  0111100010011010  x789A  data2     STR    R4, R2, x001A
 x300E  0011000000000011  x3003  data3     ST     R0, x3003
```

2. Derive the truth table for the following logic circuit: (4)



Answer



x	*y*	*z*	*F*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

3. Based on the circuit diagram from question 2 above, write a LC-2 assembly program that will input three 16-bit values for *x*, *y*, and *z*, and output the resulting 16-bit value for *F* according to the circuit diagram. When doing I/O's just take the values as is, i.e. you do not need to convert the values from/to ASCII. Hint: DeMorgan's Theorem gives the equality

$$a+b+c = (a'b'c')' \tag{4}$$

Answer

```
        .orig    x3000
        trap     x20                    ;x
        add      r1,r0,#0
        trap     x20                    ;y
        add      r2,r0,#0
        trap     x20                    ;z
        add      r3,r0,#0

        ;x or y
        not      r1,r1                  ;x'
        not      r4,r2                  ;y'
        and      r4,r1,r4               ;x'y'
        not      r1,r4                  ;x+y

        ;z'
        not      r3,r3

        ;y and z'
        and      r2,r2,r3

        ;F
        not      r1,r1
        not      r2,r2
        not      r3,r3
        and      r4,r1,r2
        and      r4,r3,r4
        not      r4,r4

        trap     x21
        halt
        .end
```

4. Write a LC-2 assembly program that will do the following. Assume that register R1 contains the numerical value of a positive number. Divide this number in R1 by 4. The remainder from the division is going to be a number between 0 and 3 inclusive. Depending on this remainder, print out the following appropriate string:

   "The remainder is 0"
   "The remainder is 1"
   "The remainder is 2"
   or
   "The remainder is 3" (4)

Answer

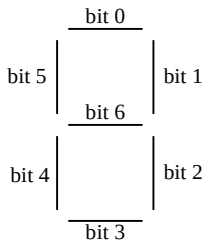
```
        .orig    x3000
        ld       r0,neg4
again   add      r2,r1,r0           ;r1 - 4
        brn      done
        add      r1,r1,r0
        br       again

done    ;remainder is in r1
        ;print first part of message
        lea      r0,msg
        trap     x22

        ;convert remainder to ascii
        ld       r3,asc30
        add      r0,r1,r3
        trap     x21
        halt

neg4    .fill    #-4
asc30   .fill    x30
msg     .stringz "The remainder is "
        .end
```

5. Assume that one seven-segment light (LED) similar to those used in a digital clock is connected to the LC-2 using the memory mapped technique at address xE000. The seven lights are turned on or off using the first seven bits of xE000 as shown below



To turn on a particular light, that corresponding bit is set to a 1. A 0 bit will turn that corresponding light off. For example, to display the number 2, bits 0, 1, 3, 4 and 6 are set to1 while the remaining bits are set to 0. Thus, you would store the value 0000 0000 0101 1011 into location xE000.

Write a LC-2 assembly program to read a one-digit number and display that number on the seven-segment LED. (4)

Answer

```
        .orig    x3000
        ld       r1, neg30
        trap     x20
        add      r1, r0, r1
        lea      r2, Dig0
        add      r2, r2, r1
        ldr      r3, r2, #0
        ld       r4, LED
        str      r3, r4, #0
        halt

neg30   .fill    xFFD0
LED     .fill    xE000              ;where the 7-segment is connected
Dig0    .fill    x003F              ;0,1,2,3,4,5
Dig1    .fill    x0006              ;1,2
Dig2    .fill    x005B              ;0,1,3,4,6
Dig3    .fill    x004F              ;0,1,2,3,6
Dig4    .fill    b01100110          ;1,2,5,6       can use binary or hex
Dig5    .fill    b01101101          ;0,2,3,5,6
Dig6    .fill    b01111100          ;2,3,4,5,6
Dig7    .fill    b00000111          ;0,1,2
Dig8    .fill    b01111111          ;0,1,2,3,4,5,6
Dig9    .fill    b01100111          ;0,1,2,5,6
        .end
```

LC-2 Instruction Summary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0001				DR			SR1			0	0	0	SR2		
ADD	0001				DR			SR1			1	imm5				
AND	0101				DR			SR1			0	0	0	SR2		
AND	0101				DR			SR1			1	imm5				
BR	0000				n	z	p	pageoffset9								
JSR	0100				L	0	0	pageoffset9								
JSRR	1100				L	0	0	BaseR			index6					
LD	0010				DR			pageoffset9								
LDI	1010				DR			pageoffset9								
LDR	0110				DR			BaseR			index6					
LEA	1110				DR			pageoffset9								
NOT	1001				DR			SR			1	1	1	1	1	1
RET	1101				0	0	0	0	0	0	0	0	0	0	0	0
RTI	1000				0	0	0	0	0	0	0	0	0	0	0	0
ST	0011				SR			pageoffset9								
STI	1011				SR			pageoffset9								
STR	0111				SR			BaseR			index6					
TRAP	1111				0	0	0	0	trapvect8 x20 = GetC x21 = Out x22 = PutS x23 = In x25 = Halt							