UNIVERSITY OF CALIFORNIA, RIVERSIDE

Department of Computer Science and Engineering

CS61 – Machine Organization and Assembly Language

Midterm 2

August 21, 2001

16

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

Please print legibly

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

1. Assemble the following LC-2 assembly program, i.e., convert the assembly language to machine language. Refer to the table on the last page for opcodes. (4)

```
        .orig    x3000
start   ld       r1, number
        lea      r2, start
        and      r3, r1, #5
        str      r1, r2, #4
        add      r4, r3, #5
        halt

number  .fill    x2806
        .end
```

Answer

x3000	0010001000000110	x2206	start	LD	R1, number
x3001	1110010000000000	xE400		LEA	R2, start
x3002	0101011001100101	x5665		AND	R3, R1, x0005
x3003	0111001010000100	x7284		STR	R1, R2, x0004
x3004	0001100011100101	x18E5		ADD	R4, R3, x0005
x3005	1111000000100101	xF025		TRAP	HALT
x3006	0010100000000110	x2806	number	LD	R4, number

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

Answer

R1 = x2806
R2 = x3000
R3 = x0004
R4 = x2806

```
  R0  x0000  0          R4  x2806  10246      PC  x3005  12293
  R1  x2806  10246      R5  x0000  0          IR  x2806  10246
  R2  x3000  12288      R6  x0000  0          CC  P
  R3  x0004  4          R7  x0000  0

x3000  0010001000000110  x2206  start     LD     R1, number
x3001  1110010000000000  xE400             LEA    R2, start
x3002  0101011001100101  x5665             AND    R3, R1, x0005
x3003  0111001010000100  x7284             STR    R1, R2, x0004
x3004  0010100000000110  x2806             LD     R4, number
x3005  1111000000100101  xF025             TRAP   HALT
x3006  0010100000000110  x2806  number     LD     R4, number
```

3. Write a LC-2 assembly language program to solve the following problem:

   Given three 16-bit words (A, B, and C) as defined by the following three .FILL statements

```
A       .FILL    x00B2
B       .FILL    x004A
C       .FILL    x0000
```

   Write a program that will concatenate the first byte of A (i.e. B2) with the first byte of B (i.e. 4A) to form xB24A and write it into C. In other words, at the termination of the program, the memory location at C should contain xB24A. (4)

Answer

```
        .orig    x3000
        ld       r1, A              ; get value in A
        ; shift A left 8 bits by adding to itself 8 times
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left
        add      r1, r1, r1         ; shift A 1 bit to the left

        ; A or B = (A' and B')'
        not      r1, r1             ; A'

        ld       r2, B              ; get value in B
        not      r2, r2             ; B'
        and      r3, r1, r2         ; A' and B'
        not      r3, r3             ; (A' and B')'
        st       r3, C
        halt

A       .fill    x00B2
B       .fill    x004A
C       .fill    x0000
        .end
```

4. Very often, we perform a table lookup to convert a numerical value to a different character. For example, if I want to do the following conversions:

   $1 \rightarrow A$
   $2 \rightarrow B$
   $3 \rightarrow C$
   $4 \rightarrow D$

   I will use the following table

Index	Content of table
1	A
2	B
3	C
4	D

   so that, for instant, given the number 3, I do a table lookup and retrieve the character C.

   Assume that the above table is defined using the following statements:

```
Table   .FILL    x0041              ; ASCII for the character A
        .FILL    x0042              ; ASCII for the character B
        .FILL    x0043              ; ASCII for the character C
        .FILL    x0044              ; ASCII for the character D
```

   Translate the following high-level pseudo-code to LC-2 assembly code:

```
loop:   input number
        if 1 ≤ number ≤ 4 then {
             do table lookup using the number as the index
             output the corresponding character from that index location
             goto loop
             }
        halt
```

(4)

Answer

```
        .orig    x3000
loop    trap     x20                    ; input character
        ld       r1, NegASCIIOffset  ; -30 hex
        add      r1, r0, r1          ; subtract x30 to convert from hex to
                                     dec

        ; check for correct range
        add      r0, r1, #-1         ; >= 1
        brn      end
        add      r0, r1, #-4         ; <= 4
        brp      end

        ; do table lookup
        lea      r2, Table
        add      r2, r2, r1
        add      r2, r2, #-1         ; r2 contains correct address
        ldr      r0, r2, #0          ; load corresponding character from
                                     table
        trap     x21                 ; output character
        br       loop

end     halt

NegASCIIOffset   .fill              xFFD0    ; negative 30 in hex
Table   .fill    x0041              ; ASCII for the character A
        .fill    x0042              ; ASCII for the character B
        .fill    x0043              ; ASCII for the character C
        .fill    x0044              ; ASCII for the character D
        .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							