

UNCA CSCI 255.001 and NCSU ECE 109 Sections 002, 602, 604, and 605
Final Exam Spring 2008 Solution
 1 May, 2008

Problem 1 (6 points) Data Path

Once the following LC-3 instruction has been loaded into the IR and decoded

STR R4, R5, #6

the actions listed below must be performed. Describe how each of these actions is accomplished within the LC/3 data path by making specific reference to its components:

- a) R5 and the immediate value #6 are added to generate a memory address
R5 is transmitted through the ADDR1MUX while 6 is transmitted through the ADD2MUX. They are added and sent through the MARMUX.
- b) The generated memory address is sent to the memory module
The output of the MARMUX is sent through the bus and loaded into the MAR.
- c) The value in R4 is sent to the memory module and stored in memory.
The value in R4 is sent through the bus and loaded into the MDR. A write memory operation is then enabled.

Problem 2 (9 points) Hand assembled

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

Binary	Assembly
	.ORIG x3000
0010010000000100	LD R2, LABV
0001000000000001	LP ADD R0, R0, R1
0001010010111110	ADD R2, R2, #-2
0000001111111101	BRp LP
1100000111000000	RET
1111111111111011	LABV .FILL #-5
	.END

Problem 3 (7 points) Floating point

Express the following three numbers in IEEE floating point notation. I've left some spaces between the major bit fields in the number. None of these answers should involve long calculations.

12.5 0 1000010 100100000000000000000000
0.125 0 01111100 000000000000000000000000
-12 0 1000010 100000000000000000000000

Problem 4 (4 points) Ranges

What is the largest number that be represented in 8-bit twos-complement notation?

127 (2^7-1)

What is the smallest number that can be represented in 8-bit twos-complement notation?

-128 (-2^7)

Problem 5 (4 points) Bitwise operations

Perform the following two bit-wise logical operations on 8-bit numbers expressed as two hexadecimal digits. Your answer should also be expressed in hexadecimal.

NOT(25)	-->	DA
AND(25 , F0)	-->	20

Problem 6 (6 points) Memories

a) How many **bits** are in a memory with 2K words and a word size of 64?

128 K ($2K * 64$)

b) How many **bits** are required to address a memory with 16K words?

14 (because 16 K is $2^4 * 2^{10}$ or 2^{14})

c) How many 8-bit words can be stored in a 1MB (1M byte) memory?

1M (because there are 8 bits in a byte)

Problem 7 (6 points) Vocabulary

Compare and contrast three of the following four pairs of related terms. Cross out the one you do not want graded. Otherwise, I'll grade all four with equal weight.

I think you can look these up in the textbook

Device data register vs Device status register
Combinational logic vs. Sequential logic
Assembly language vs. C
Programmed I/O vs Memory-mapped I/O

Problem 8 (3 points) C

What does the following program do when 255 and 109 are entered as input.

```
#include <stdio.h>
main() {
    int i, j ;
    scanf("%d", &i) ;
    scanf("%d", &j) ;
    i = i*2 ;
    j = j/2 ;
    printf("Your results are %d and %d", i, j) ;
}
```

(You most definitely do **not** need a calculator to solve this problem.)

It prints the line

Your results are 510 and 54

Problem 9 (11 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 x1040 have the values shown on the right.

Register	Value	Address	Value
R0	x0000	x1040	x4101
R1	x0000	x1041	x5111
R2	x0000	x1042	x6121
R3	x0000	x1043	x7131
R4	x4111	x1044	x0000
R5	x5111	x1045	x0000
R6	x6111	x1046	x0000
R7	x7111	x1047	x0000

For the six addresses shown below, write a single LC/3 instruction to load the value **stored in** the specified memory location into register 4. (For example, when x1041 is specified, x5111 should be stored in R4.) Assume that each instruction is located at memory address x1020.

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

x1052	LD R4, x31
x1152	Can't be done in one
x4101	LDI R4, x1F or LDR R4, R4, #-15
x4102	LDR R4, R4, #-15
x7131	LDI R4, x22
x7132	Can't be done in one

Problem 10 (4 points)

Assuming the following symbol table with only one location

EGGHEAD	x3015
---------	-------

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 x3011 in both cases. If the assembly language statement is illegal, state the reason why.

LD R0, EGGHEAD	0010 000 00000011
ADD R2, R4, EGGHEAD	Can't use address as third operand

Problem 11 (4 points) Adding

Add the following pairs of seven-bit two's complement numbers and indicate which additions result in an overflow.

$\begin{array}{r} 1111111 \\ + 1110100 \\ \hline 1110111 \end{array}$	$\begin{array}{r} 1011111 \\ + 1011111 \\ \hline 0111110 \\ \text{overflow} \end{array}$
$\begin{array}{r} 0011000 \\ + 0111111 \\ \hline 1010111 \\ \text{Overflow} \end{array}$	$\begin{array}{r} 0000111 \\ + 1111001 \\ \hline 0000000 \end{array}$

Problem 12 (5 points) Transistors

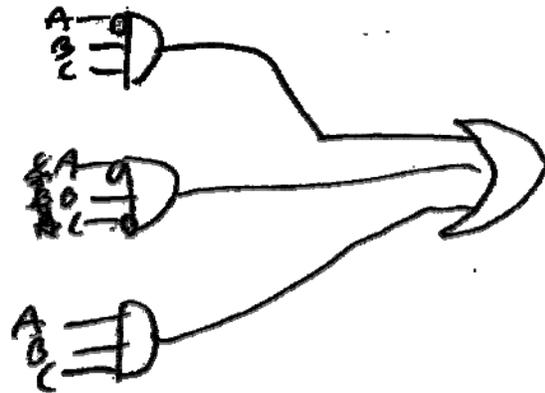
In the space below, draw a CMOS implementation of a 2-input AND gate.

Look in the textbook

Problem 13 (6 points) Truth to Gates

Draw a circuit, at the gate level, that will implement the following truth table, where A, B, and C are inputs and where Z is the single output

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Problem 14 (5 points) LC/3 programming

In this problem, write a section of LC/3 code that

- 1) tests the character in R0
- 2) sets R1 depending on the values of the character in R0
 - a) R1 is set to 1, if the character in R0 is a space (ASCII x20)
 - b) R1 is set to 0, if the character in R0 is not a space

```
        AND  R1,R1,#0
        LD   R2,negSP
        ADD  R2,R2,R0
        BRnp skipADD
        ADD  R1,R1,#0
skipADD .....
```

```
negSP   .FILL #-32
```

Problem 15 (2 points) Simple L/3 subroutines

If Problem 14 were to be implemented as a subroutine that receives its input in R0 and returns its output in R1, what LC/3 instruction(s) would need to be added to your code?

Add a RET instruction at the end and also code to save and restore registers other than R0 and R1.

Problem 16 (3 points) Not-so-simple LC/3 subroutines

Suppose Problem 14 were to be implemented as a subroutine that follows the LC/3 ABI presented in class (and pictured on the reference sheet) and the arguments were passed and returned on the stack.

Write a single LC/3 instruction that would load the input argument into R0.

```
LDR  R0,R5,#4
```

Write a single LC/3 instruction that would store R1 into the return value slot for the subroutine.

```
LDR  R1,R5,#3
```

(Big hint: Both of these instructions involve the use of register R5.)

Problem 17 (5 points)

Suppose R6 has the initial value of x5555. What are the values of register R1, R2, R3, R4, and R6 after the following sequence of PUSH and POP operations have been performed. Drawing the stack might increase your change of receiving partial credit. (Remember that in the PUSH operation R6 is decremented before a value is stored on the stack.)

```
PUSH x2
PUSH x3
POP  R1          R1 will be x3
PUSH x5
PUSH x7
POP  R2          R2 will be x7
PUSH xB
POP  R3          R3 will be xB
POP  R4          R4 will be x5
PUSH x11        R6 will be x5553
```

Problem 18 (10 points) LC/3 I/O

In this program you are going to write a little piece of LC/3 **twice** that

- 1) reads a character from the keyboard
- 2) outputs a single character to the display
 - a) the output character is 'Y', if the input character is a space
 - b) the output character is 'N', if the input character is not a space

(By the way, the ASCII value for space is x20, the ASCII value for 'Y' is x59, and the ASCII value for 'N' is x4E.)

First, write the code using the LC/3 TRAP routines. The names of the TRAP routines are on the reference sheet. This part should have a shorter answer than the next part.

```
GETC          negSP      .FILL  #-32
LD           R1,negSP    ascY      .FILL  #89
ADD          R1,R0,R1    ascN      .FILL  #78
BRnp        notSP
LD           R0,ascY
BR          prtCh
notSP        LD           R0,ascN
ptrCh        OUT
```

Second, write the code using the LC/3 device data and status registers. This is part where you'll have to use polling. The addresses of the device registers are also on the reference sheet. You may use any .FILL's you defined above in your answer here.

```
InPoll       LDI          R1,KBSR      negSP      .FILL  #-32
              BRzp        InPoll      ascY      .FILL  #89
              LDI          R0,KBDR     ascN      .FILL  #78
              LD           R1,negSP    KBSR      .FILL  xFE00
              ADD          R1,R0,R1    KBDR      .FILL  xFE02
              BRnp        notSP        DSR       .FILL  xFE04
              LD           R0,ascY     DDR       .FILL  xFE06
              BR          OutPoll
notSP        LD           R0,ascN
OutPoll      LDI          R1,DSR
              BRzp        OutPoll
              STI          R0,DDR
```