

120 minutes (maximum), Closed Book, No Calculators

- You may use one side of one sheet (8.5x11) of paper with any notes you like. The notes must have your name and ID, and must be turned in with the exam.
- Place the notes and a writing utensil on your desk and put all other material under your desk.
- This exam has 14 pages, including the LC3 instruction formats and a blank sheet for extra work at the end of the exam. Do all your work on these exam sheets.
- Be specific and clear in your answers. If there is any question about what is being asked, then indicate the assumptions you need to make to answer the question.
- Show all your work if you wish to be considered for partial credit.

**Name:** \_\_\_\_\_

**Email:** \_\_\_\_\_

**EID:** \_\_\_\_\_

**DO NOT TURN TO NEXT PAGE TILL YOU GET PERMISSION**

Question	Points	Score
1	10	
2	15	
3	15	
4	20	
5	15	
6	25	
total	100	

1. [10 points] **Number Representation**

**Note: Hexadecimal is base<sub>16</sub>, decimal is base<sub>10</sub>, and binary is base<sub>2</sub>.**

Convert hexadecimal to binary and vice versa:

a)  $1A2B3C4D_{16} = \underline{\hspace{15em}}_2$

b)  $1111101101110010_2 = \underline{\hspace{15em}}_{16}$

Convert hexadecimal (2's complement) to a **signed** decimal number:

c)  $FFFC_{16} = \underline{\hspace{15em}}_2 = \underline{\hspace{15em}}_{10}$

Convert floating point to binary to hexadecimal:

d) 2.75f

Binary  $\underline{\hspace{15em}}_2$

Hexadecimal  $\underline{\hspace{15em}}_{16}$

Convert hexadecimal to binary to floating point:

e)  $40880000_{16}$

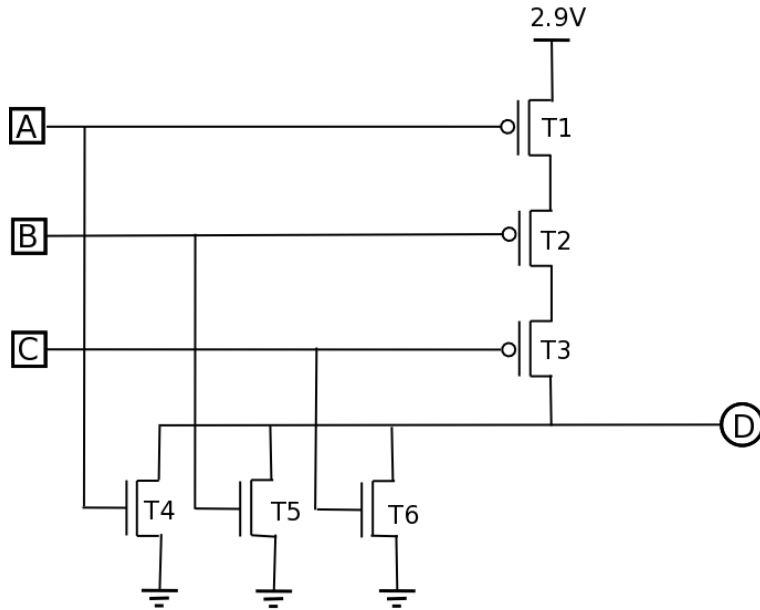
Binary  $\underline{\hspace{15em}}_2$

Float  $\underline{\hspace{15em}}_f$

Hint: IEEE has 1 sign bit, 8 exponent bits, 23 mantissa/fraction bits.

2. [15 points] **Transistors and Gates**

a) Fill in the table for the circuit shown below:



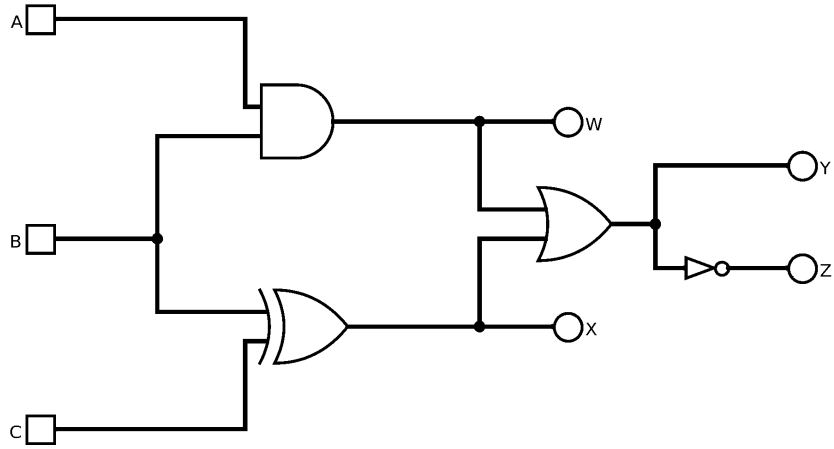
Open or closed for all transistors, and 0 or 1 for all outputs.

A	B	C	T1	T2	T3	T4	T5	T6	O
0	0	0	closed	closed	closed	open	open	open	
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

Hint: You can tell how the transistors work from the first line of the truth table.

b) What is the logical function of the circuit? \_\_\_\_\_

c) Fill in the truth table for the combinational circuit shown below:



A	B	C	W	X	Y	Z
0	0	0	0	0	0	1
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

d) How many test cases are needed to **exhaustively** test the circuit? \_\_\_\_\_

3. [15 points] **LC-3 Interpretation**

Answer all the following questions using the LC-3 instruction set, which is shown on the last page of this exam.

a) Translate the binary LC-3 machine instruction 0111001010110111 to assembly code:

b) Translate the LC-3 assembly statement NOT R4,R6 to a binary instruction:

c) If binary LC-3 instruction 0010110111111101 or LD R6,DATA is executed at address 0x3050, then what is the address of DATA?

d) Will the instruction AND R3,R2,#0 followed by BRnp LABEL take the conditional branch? Explain why or why not.

e) What LC-3 instruction must be used to store data when the destination is more than a 256 address offset from the instruction?

f) What is the range of immediate values for the ADD instruction?

g) How many instructions could LC-3 support if the opcode were extended to 5 bits?

4. [20 points] **LC-3 Assembly Coding and Memory Model**

Consider the following mystery function in LC-3 assembly code:

```
;; Mystery function
FUNCTION    ;; stack entry
            ADD    R6,R6,#-1
            ADD    R6,R6,#-1
            STR    R7,R6,#0
            ADD    R6,R6,#-1
            STR    R5,R6,#0
            ADD    R5,R6,#0

            ;; function body
            LDR    R0,R5,#3
            LDR    R1,R5,#4
            LDR    R2,R5,#5
            ADD    R3,R2,R1
            ADD    R3,R3,R0
            STR    R3,R5,#2

            ;; stack exit
EXIT        LDR    R5,R6,#0
            ADD    R6,R6,#1
            LDR    R7,R6,#0
            ADD    R6,R6,#1
            RET
```

Hint: Draw a box around PUSH and POP pairs of instructions.

a) How many parameters does the function require? \_\_\_\_\_

b) Does the function have a return value (Yes/No)? \_\_\_\_\_

c) What calculation is performed by the function? \_\_\_\_\_

d) Write the main entry point code to call the function, assuming the parameters are in R0, R1, R2, and you want the result in R3. Each line should contain exactly one instruction. The LC-3 instruction set is on the final page of this exam.

```

.ORIG x3000
LD R6, STACK
_____ ;; Push parameters
_____
_____
_____
_____
JSR FUNCTION
_____ ;; Retrieve result
_____ ;; Clean up stack
STACK .FILL x4000
.END

```

e) Draw the a diagram of the stack just before the instruction at the EXIT label is executed. Fill in the stack contents and label the location of the Stack Pointer and Frame Pointer for the function. Note that the table may show more addresses than are needed.

Stack Address	Stack Contents
x3FF8	
x3FF9	
x3FFA	
x3FFB	
x3FFC	
x3FFD	
x3FFE	
x3FFF	



5. [15 points] **C Interpretation**

For the C code snippets shown below, write what will be printed next to each printf statement:

a) C bitwise operators

```
int i0 = 0x12345678;
int i1 = 0x0FF00FF0;
printf("0x%x\n", i0 & i1);
printf("0x%x\n", i0 | i1);
```

b) C pointer usage

```
int i;
i = 0x12345678;
int *p = &i;
*p = 0x11223344;
printf("0x%x 0x%x\n", i, *p);
```

c) C loop control

```
for (int i = 0; i < 4; ++i) {
    int j = 2;
    while (j--) {
        printf("%d\n", i + j + 1);
    }
}
```

d) Finish the code require to delete a node in the following linked list example:

```
typedef struct _Node {
    int id;
    _Node *next;
} Node;
```

```
// Delete the node which follows the specified previous node in the linked list:
```

```
void deleteNode(Node *previous) {
```

```
    // Get current node
    Node *current =
```

```
    // Unlink current node
    previous->next =
```

```
    // Free current node
    free (          );
```

```
}
```

Hint: Checking any of the pointers for NULL is not required.
--

## 6. [25 points] C Programming

Write a C program that allows the user to enter student names, ID numbers, and GPAs. The program should store the data in an array of structures, and write the data to an output file which is the only command-line argument. Fill in the comment block with the description, author, and date, all other comments are provided. Step by step instructions are given below.

Grading will be based on correct syntax and semantics of the program, i.e. would the code compile and work. Do your best, and try to work around things you don't know. Hint: The length of my program for this specification is 7 lines for the structure definition, 2 lines for prototypes, 17 lines in main, 8 lines in *inputStudent*, and 1 line in *outputStudent*.

```
// Name: final.c
// Description:
// Author:
// Date:

// Include the header files for the printf/scanf and malloc/free/exit functions.
#include <                >
#include <                >

// Declare and typedef a Student structure with first and last name, id, and gpa.
// Names are strings (32 characters), id is integer, and gpa is floating-point.

        struct
{

}

// Define a function prototype called inputStudent that takes a structure pointer.
// Define a function prototype called outputStudent that takes a FILE * and structure.
// The return type on both functions is void.
```

```

// Write the main entry point with the usual arguments.
int main(
)
{
    // Dynamically allocate a local array of at least 10 Student structures.

    // Declare an integer variable to hold the count.

    // Prompt the user to input the number of students.

    // Read the number of students from standard input to the count.

    // Declare a file pointer

    // Open the file specified in the first argument for writing.

    // Check file pointer and print out an error with the filename if necessary.

    // Otherwise create a loop that iterates over the structure array.
    for (
)
    {
        // Call inputStudent with a pointer to the current structure.

        inputStudent(
);

        // Call outputStudent with the file pointer and current structure.

        outputStudent(
);
    }

    // Free the array of Student structures.

    // Close the file and return success.

}

```

```
// Implement inputStudent to read data from the user into the structure.
// Prompt the user for each field.
void inputStudent(                               )
{

}

// Implement outputStudent to write from the structure to the file pointer.
// Output should be one structure per line, fields separated by a space.
void outputStudent(                             )
{

}

}
```

Extra Work Area:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD <sup>+</sup>	0001			DR			SR1			0	00		SR2			
ADD <sup>+</sup>	0001			DR			SR1			1	imm5					
AND <sup>+</sup>	0101			DR			SR1			0	00		SR2			
AND <sup>+</sup>	0101			DR			SR1			1	imm5					
BR	0000			n	z	p	PCoffset9									
JMP	1100			000			BaseR			000000						
JSR	0100			1	PCoffset11											
JSRR	0100			0	00		BaseR			000000						
LD <sup>+</sup>	0010			DR			PCoffset9									
LDI <sup>+</sup>	1010			DR			PCoffset9									
LDR <sup>+</sup>	0110			DR			BaseR			offset6						
LEA <sup>+</sup>	1110			DR			PCoffset9									
NOT <sup>+</sup>	1001			DR			SR			111111						
RET	1100			000			111			000000						
RTI	1000			000000000000												
ST	0011			SR			PCoffset9									
STI	1011			SR			PCoffset9									
STR	0111			SR			BaseR			offset6						
TRAP	1111			0000			trapvect8									
reserved	1101															

Figure A.2 Format of the entire LC-3 instruction set. Note: + indicates instructions that modify condition codes