CS 270 Computer Organization	Final
Fall 2010	December 14, 2010

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: _____

CONFIDENTIAL: ANSWER SHEET AND GRADING CRITERIA

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$_{16}$, decimal is base$_{10}$, and binary is base$_2$.

   Convert hexadecimal to binary and vice versa:

   a) 1A2B3C4D$_{16}$ = 0001 1010 0010 1011 0011 1100 0100 1101$_2$ [1 point]

   b) 1111101101110010$_2$ = FB72$_{16}$ [1 point]

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

   c) FFFC$_{16}$ = 1111 1111 1111 1100$_2$ = -0000 0000 0000 0100$_2$ = -4$_{10}$ [2 points]

   Convert floating point to binary to hexadecimal:

   d) 2.75f [3 points]

   Binary 0 10000000 01100000000000000000000$_2$

   Hexadecimal 40300000$_{16}$

   Convert hexadecimal to binary to floating point:

   e) 40880000$_{16}$ [3 points]

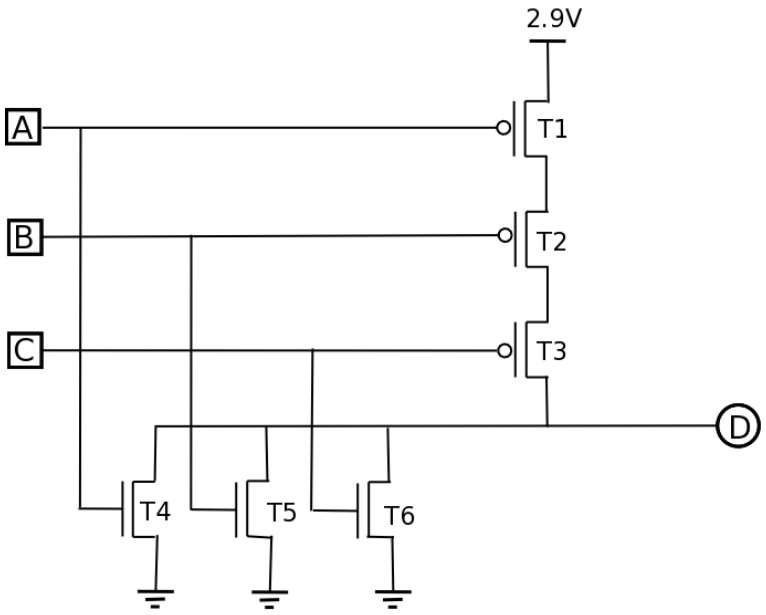
   Binary 0 10000001 00010000000000000000000$_2$

   Float 4.25f

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: [6 points]



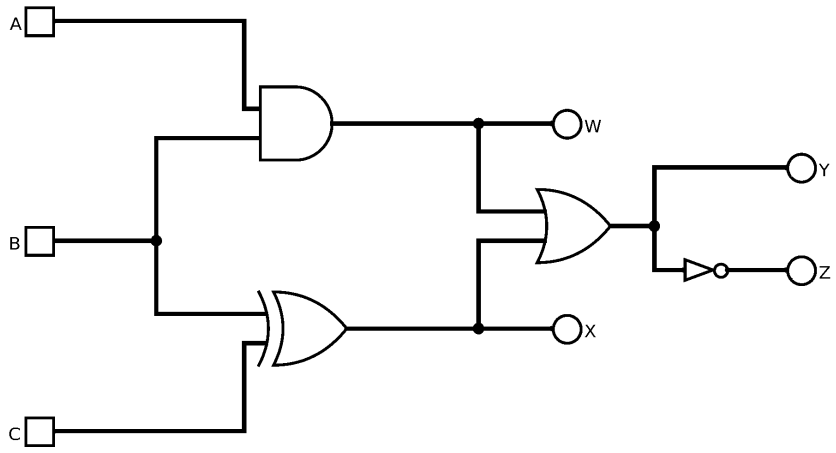
Open or closed for all transistors, and 0 or 1 for all outputs.

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

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? 3-input NOR gate [2 points]

c) Fill in the truth table for the combinational circuit shown below:[6 points]



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

d) How many test cases are needed to **exhaustively** test the circuit? $2^3 = 8$ [1 points]

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:

STR R1,R2, # -9 [2 points]

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

0x99BF [2 points]

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

111111101 = -000000011 = -3, so 0x3050 + 1 - 3 = 0x304E [3 points]

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

Will not branch since only the ZERO condition bit is set. [2 points]

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

Both STI and STR can access full address space, ST only allows -256..+255 from instruction. [2 points]

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

-16 to +15 [2 points]

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

32 instructions [2 points]

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? 3 parameters [2 points]

b) Does the function have a return value (Yes/No)? Yes [2 points]

c) What calculation is performed by the function? Add 3 integers and return result [2 points]

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. [6 points]

```
        .ORIG x3000
        LD R6, STACK
        ADD R6,R6,#-1
        STR R2,R6,#0
        ADD R6,R6,#-1
        STR R1,R6,#0
        ADD R6,R6,#-1
        STR R0,R6,#0
        JSR FUNCTION
        LDR R3,R6,#0
        ADD R6,R6,#4
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. [8 points]

Stack Address	Stack Contents
x3FF8	
x3FF9	
x3FFA	Frame Pointer
x3FFB	Return address
x3FFC	Return value
x3FFD	Parameter 0
x3FFE	Parameter 1
x3FFF	Parameter 2

Frame Pointer (R5) is 0x3FFA and Stack Pointer (R6) is 0x3ffA

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 [3 points]

```
int i0 = 0x12345678;
int i1 = 0x0FF00FF0;
printf("0x%x\n", i0 & i1); // 0x02300670
printf("0x%x\n", i0 | i1); // 0x1FF45FF8
```

b) C pointer usage [3 points]

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

c) C loop control [3 points]

```
for (int i = 0; i < 4; ++i) {
    int j = 2;
    while (j--) {
        printf("%d\n", i + j + 1); // 2 1 3 2 4 3 5 4, one per line of output since '\n'
    }
}
```

d) Finish the code require to delete a node in the following linked list example: [6 points]

```
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 = previous->next;

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

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

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*.

Grading Criteria: header files: 2 points, structure definition: 3 points, function prototypes: 2 points, main arguments: 1 point, dynamic allocation: 2 points, input count: 1 point, file pointer and fopen: 2 points, file error check: 1 point, loop counter: 1 point, inputStudent call: 1 point, outputStudent call: 1 point, fclose call: 1 point, free call: 1 point, exit call: 1 point, inputStudent code: 3 points, outputStudent code: 2 points

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

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

// 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.
typedef struct
{
    char first[32];
    char last[32];
    int id;
    float gpa;
} Student;

// 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.
void inputStudent(Student *pStudent);
void outputStudent(FILE *fp, Student student);
```

```
// Write the main entry point with the usual arguments.
int main(                                          )
{
    // Dynamically allocate a local array of at least 10 Student structures.
    Student *students = (Student *) malloc(sizeof(Student) * 10);

    // Declare an integer variable to hold the count.
    int count;

    // Prompt the user to input the number of students.
    printf("Enter number of students");

    // Read the number of students from standard input to the count.
    scanf("%d", &count);

    // Declare a file pointer
    FILE *fp;

    // Open the file specified in the first argument for writing.
    fp = fopen(argv[1], "e");

    // Check file pointer and print out an error with the filename if necessary.
    if (fp == NULL)
    {
        printf("Cannot write %s\n", argv[1]);
        exit(-1);
    }

    // Otherwise create a loop that iterates over the structure array.
    for (int s = 0; s < count; ++s)
    {
        // Call inputStudent with a pointer to the current structure.
        inputStudent(&students[s]);

        // Call outputStudent with the file pointer and current structure.
        outputStudent(fp, students[s]);
    }

    // Free the array of Student structures.
    free(students);

    // Close the file and return success.
    fclose(fp);
    exit(0);
}
```

```
// Implement inputStudent to read data from the user into the structure.
// Prompt the user for each field.
void inputStudent(Student *pStudent)
{
    printf("First name: ");
    scanf("%s", pstudent->first);
    printf("Last name: ");
    scanf("%s", pstudent->last);
    printf("Identification: ");
    scanf("%d", &(pstudent->id));
    printf("GPA: ");
    scanf("%f", &(pstudent->gpa));
}

// 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(                                                  )
{
    fprintf(fp, "%s %s %d %f\n", student.first,
                                 student.last,
                                 student.id,
                                 student.gpa);
}
```

Extra Work Area:

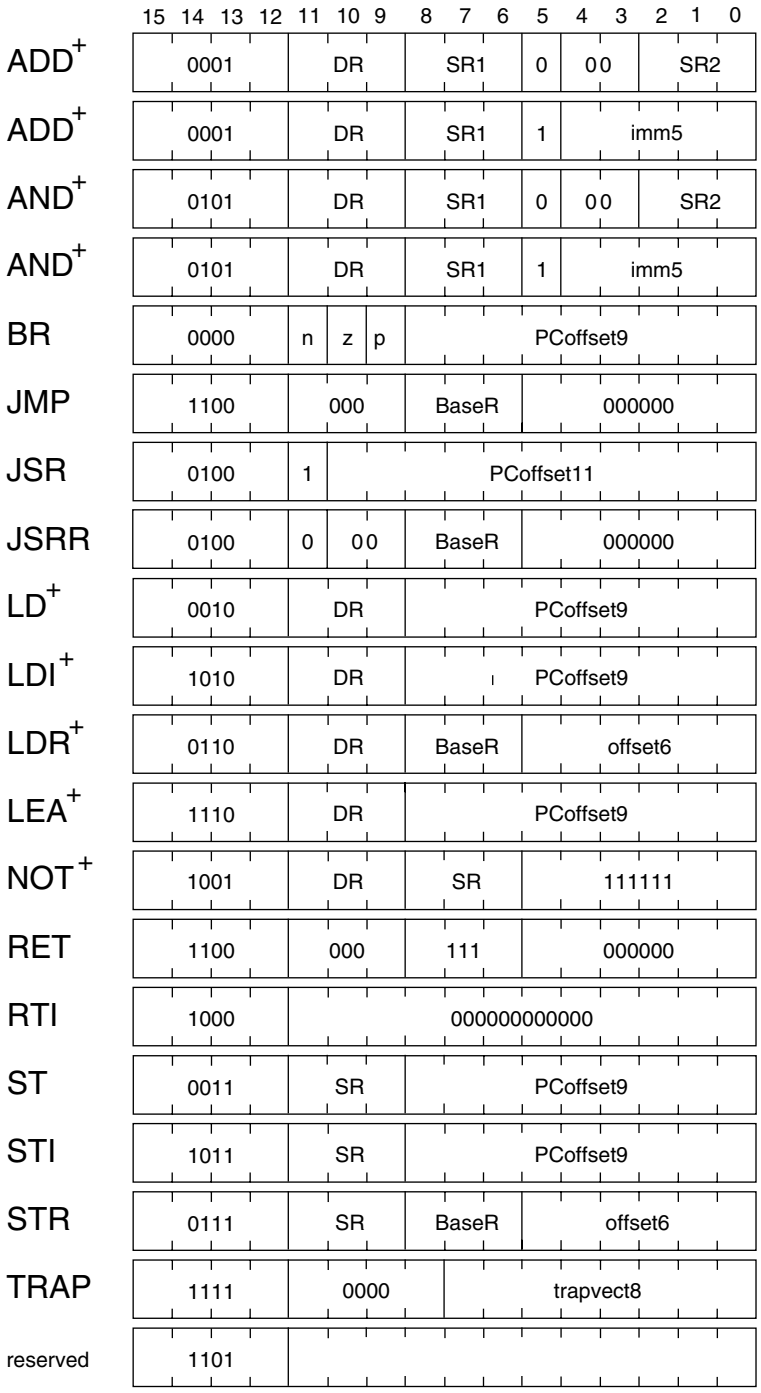


Figure A.2 Format of the entire LC-3 instruction set. **Note:** + indicates instructions that modify condition codes