CS 2110 Spring 2019 Practice Final Examination

GTID#:_____

Name:_____ GT Account: _____

Problem	Points	Lost	Gained	Category	TA
1	100				
2	100				
3	100				
4	100				
5	100				
6	100				
7	100				
8	100				
9	100				
10	100				
11	100				
12	100				
13	100				
Total	1300				

*We will **not** be publishing the answers for this test. It is of much greater value if you determine the answers for yourself. However, feel free to create Piazza topics to check and discuss your answers.*

Note: By writing your name at the top of this test you are certifying that this test is entirely your own work.

You may ask proctors for clarification but you are ultimately responsible for the answer you write on the paper.

Illegible answers are, regrettably, wrong answers.

Fill out the top of this page. Putting at least your GT Account on every page helps us in case a page gets detached.

CS 2110 Spring 2019 Practice Final Examination
GTID#:_____

C – Analyze and Explain Data Type Declarations and Access Paths

1. What are the resulting types of the following expressions? Write them in English using among other things, the phrases "pointer to", "array of", and "function returning". Note, these are **expressions**, not **type declarations**. The first two are done for you.

```
char *a = "Last";
struct c {
        int c1;
        float c2;
        char *c3;
        } c;
struct c *d;
int **f(int a, int b);
float *(*g)[];
int h[10][5][3];
```

Expression	**Resulting Type**
a	Pointer to char
*a	char
a[0]	
c.c3	
*f(1, 2)	
f(3, 4)	
(*d).c3[1]	
d->c2	
g	
*g	
h[3]	
h[3][4][2]	

C – Analyze Data Types and Explain Memory Layouts

2. Presume the following variables are all outside a function definition and are assigned to the memory addresses listed.

Declaration	**Memory address assigned to the first byte**	**Total memory allocated**
int a[] = { 1, 1, 2, 3, 5, 8 };	0x20000	0x18
char *b = “string”	0x20018	0x4
struct c {		*(0x10)*
double c1;		*(0x8)*
int c2;		*(0x4)*
int c3;		*(0x4)*
};		
struct c cx[4];	0x2001c	0x40
struct c *cp = cx;	0x2005c	0x4
struct c cs;	0x20060	0x10

What will be the resulting value of the following expressions? You may not always be able to express the value as a number or character. The first is done for you.

Expression	**Value**
a+1	0x20004
&a[0]	
a[0]	
b	
b[2]	
cx	
&cx[2]	
&cx[3].c2	
cx+2	
cp + 1	
&cp[1]	

C – Trace Code Execution

3. Trace the values of the variables in the following code by writing in the values that change when each line of code is executed. The first line of the table shows the value of each variable when the declarations are finished.

```
int a = 12, b = 9;
double c = 3.5, d = -2.7;
char e[] = "badly";
char f = 'w';
int *pa = &a, *pb = &b;
double *pc = &c, *pd = &d, **ppc = &pc, **ppd = &pd;
char *pf = &f, **ppf =&pf;
```

	a	**b**	**c**	**d**	**e**	**f**	**pa**	**pb**	**pc**	**pd**	**ppc**	**ppd**	**pf**	**ppf**
	12	9	3.5	-2.7	badly	w	&a	&b	&c	&d	&pc	&pd	&f	&pf
*pa = 5														
*pb = *pa +2														
pa = pb														
*pa = 1														
pd = pc														
**ppd = 4.0														
**ppc = 1.0														
**ppf = 'X'														
pf = e														
(*ppf)[3] = 0														

C - Generate Source Code

4. You are to write a function mactonum() that takes a character string as its argument and a pointer to a struct containing 3 unsigned shorts. The character string will be in the form of an Ethernet MAC address, which consists of six pairs of hexadecimal digits separated by colons. For example an input string of "08:00:20:f5:cc:a1" should return three unsigned shorts with the values 0x8000, 0x20f5, and 0xcca1. The function should return 0 if the MAC address is badly formed and 1 otherwise.

You have the function **ctoh()** which takes a pointer to a pointer to char and returns the integer value of the hexadecimal number starting at the first character in the string. **ctoh()** increments the pointer to char to the first non-hexadecimal character it encounters.

```
struct mac {
     unsigned short macaddr[3];
};

int ctoh(char **buf) {
     int n = 0, hdigit;
     char *p;
     while (1) {
          if (**buf >= '0' && **buf <= '9')
               hdigit = **buf - '0';
          else if (**buf >= 'a' && **buf <= 'f')
               hdigit = **buf - 'a' + 10;
          else if (**buf >= 'A' && **buf <= 'F')
               hdigit = **buf - 'A' + 10;
          else
               break;
          n = n * 16 + hdigit;
          (*buf)++;
     }
     return n;
}
int mactonum(char *in, struct mac *out) {
```

```
}
```

C – Generate Source Code with Dynamic Memory Allocation

5. Finish this function called getInts. The basic idea is to have a dynamically allocated buffer of ints that grows as the number of inputs increases. Use **realloc**() to increase the buffer size. Hint: You may want to have one variable to tell you how big the buffer is and one variable to tell you how many ints are currently in the buffer. Don't forget integers are sizeof(int) bytes long. Return the number of ints in the buffer in the parameter **n**; return the dynamically allocated buffer itself as the value of the function.

Note: The parameter **max** is used to indicate the maximum number of ints that can be read in. If this number is exceeded or memory allocation fails, free up all dynamically allocated memory, set the parameter **n** to **max+1**, and return NULL.

Bonus points for not using **malloc** (i.e. only **realloc**). Bonus points for not calling realloc each time a number is read in. i.e. increase the buffer size by more than one int at a time.

```
void *realloc(void *mem, size_t newsize);
int *getInts(size_t *n, size_t max)
{
        // Declare any necessary variables here.
        int temp;
        int *retval = NULL;
        *n = 0;









        printf("Int> ");
        while(scanf("%d", &temp) != EOF)
        {
                // Your code here
```

```
            printf("Int> ");
        }
        // Don't forget n!!!


        return retval;
    }
```

C – Troubleshoot and Identify Errors in Source Code

6. What is wrong in each of the following code segments? Explain and then fix it. (Hint: Look for storage and type problems, not trivial syntax issues that a compiler would catch.)

```
// Convert a string to upper case and return a copy without changing the original
char *upcase(char *str) {
        char newstr[1024];
        for (int i = 0; str[i] != '\0' && i < sizeof(newstr)/sizeof(newstr[0]); i++)
                newstr[i] = toupper(str[i]);
        return newstr;
}
```

```
int a = 2, b = 4;

swap(&a, &b);
        …
// Swap the value of two integers
void swap(int x, int y) {
        int t = y;
        y = x;
        x = t;
}
```

```
char *s, buf[1024], c;
int i = 0;

while ((c = getchar()) != EOF && c != '\n') {
        buf[i++] = c;
}
buf[i] = '\0';
// Make a new line with the first letter capitalized
s[0] = (buf[0] >= 'a' && buf[0] <= 'z') ? buf[0] – 'a' + 'A' : buf[0];
for (i = 1; buf[i] != '\0'; i++)
        s[i] = buf[i];
s[i] = '\0';
```

Computer Architecture – Trace Assembly-Level Code

7. Show assembly instruction and the changes to register/memory values for the sequence of instructions listed below. ***(This is a good candidate for separate questions.)***

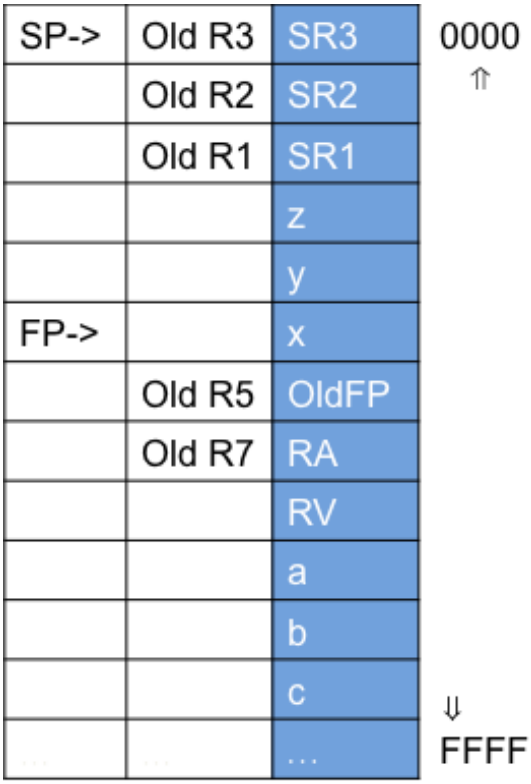
Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
x30F6	1	1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	`R1<- PC-3`
x30F7	0	0	0	1	0	1	0	0	0	1	1	0	1	1	1	0	`R2<- R1+14`
x30F8	0	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	`M[x30F4]<- R2`
x30F9	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	`R2<- 0`
x30FA	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	`R2<- R2+5`
x30FB	0	1	1	1	0	1	0	0	0	1	0	0	1	1	1	0	`M[R1+14]<- R2`
x30FC	1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	`R3<- M[M[x30F4]]`

Assembly Instruction	R1	R2	R3	0x30f4	0x3102

Opcodes
ADD = 0x1, LEA = 0xE, ST = 0x3, AND = 0x5, STR = 0x7, LDI = 0xA

Computer Architecture – Stack Frames

8. Presume you are coding the body of a function named MATH in assembly language. It takes 3 arguments named a, b, and c and it uses 3 local variables named x, y, and z. Each argument and variable take up a 16 bit word. You are given the following diagram of the LC-3 stack frame when MATH executes.



Answer the following questions in reference to this active instance of a stack frame

a. Write the assembly code to implement
z = a + b

b. Write the assembly code to implement
*y = c * 4 + z*

c. Write the assembly code to implement
x = DIV(a, y)
(DIV is a function defined elsewhere)

d. What is the purpose of the memory location labeled RV?

e. Which registers' values must not be changed while the body of MATH executes?

f. When MATH executes its final RET instruction, which words will be left on the stack?

Computer Architecture – Generate Assembly-Level Code

9. Write the assembly code instructions to perform the following actions**. You may use R4 as a temporary register**.

a. Convert the number in R3 to its twos-complement.

b. Compute the inclusive-OR of R1 and R2 with result in R0. Remember that a|b is the same as ~((~a)&(~b)).

c. Place the value 1 in R0 if the number in R1 is even; 0 otherwise

d. Multiply the number in R3 by 4.

e. Compute the NAND of R1 and R2 with result in R0

f. Compute the absolute value of R1 in R0

g. Subtract R1 from R0.

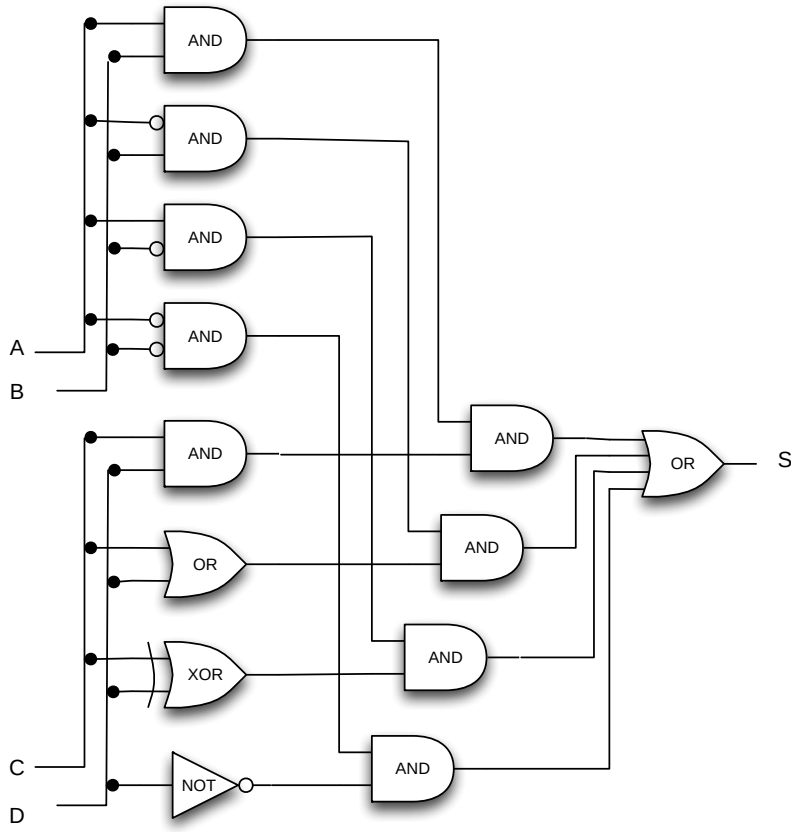
Computer Architecture – Explain the Functions/Significance of LC-3 Components

10. (a) For what purpose are multiple sign-extenders (SEXT) connected between IR and ADDR2MUX in the datapath of the LC-3. (See the reference document for the datapath diagram and instruction set table.)

(b) Why is there a need for three of them to be connected to ADDR2MUX?

Digital Logic – Analyze and Explain Logic Circuits

11. Given the following circuit, compute the truth table below. (Hint: think about what kind of circuit A and B are connected to.)



A	B	C	D	S
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

C – Identify storage location and name scope of variables

12. Given the following C code, for each variable, mark (1) the location of the allocated storage and (2) the scope of the variable name.

```
char Title[200];
extern int errorstatus;
static double stats[100][100];

double calculate(int a, int b) {
     double result;
     static int count = 0;
     result = (double)(a + count) / b;
     if (result > 0)
          count++;
     return result;
}
```

		Statically addressed	Stack	Heap		Visible to all	Visible within C file	Visible within function
Title								
errorstatus								
stats								
a								
b								
result								
count								

13. True or False

_____ (a) The BR and JMP LC-3 instructions use the same addressing mode

_____ (b) The instruction TRAP x27 loads hexadecimal 27 into the PC

_____ (c) Representing the number -81 in twos-complement requires at least 8 bits

_____ (d) An 8-bit twos-complement integer can represent numbers in the range -127 to +127

_____ (e) A twos-complement integer can be multiplied by two by shifting it one bit to the left, filling the low-order bit with zero

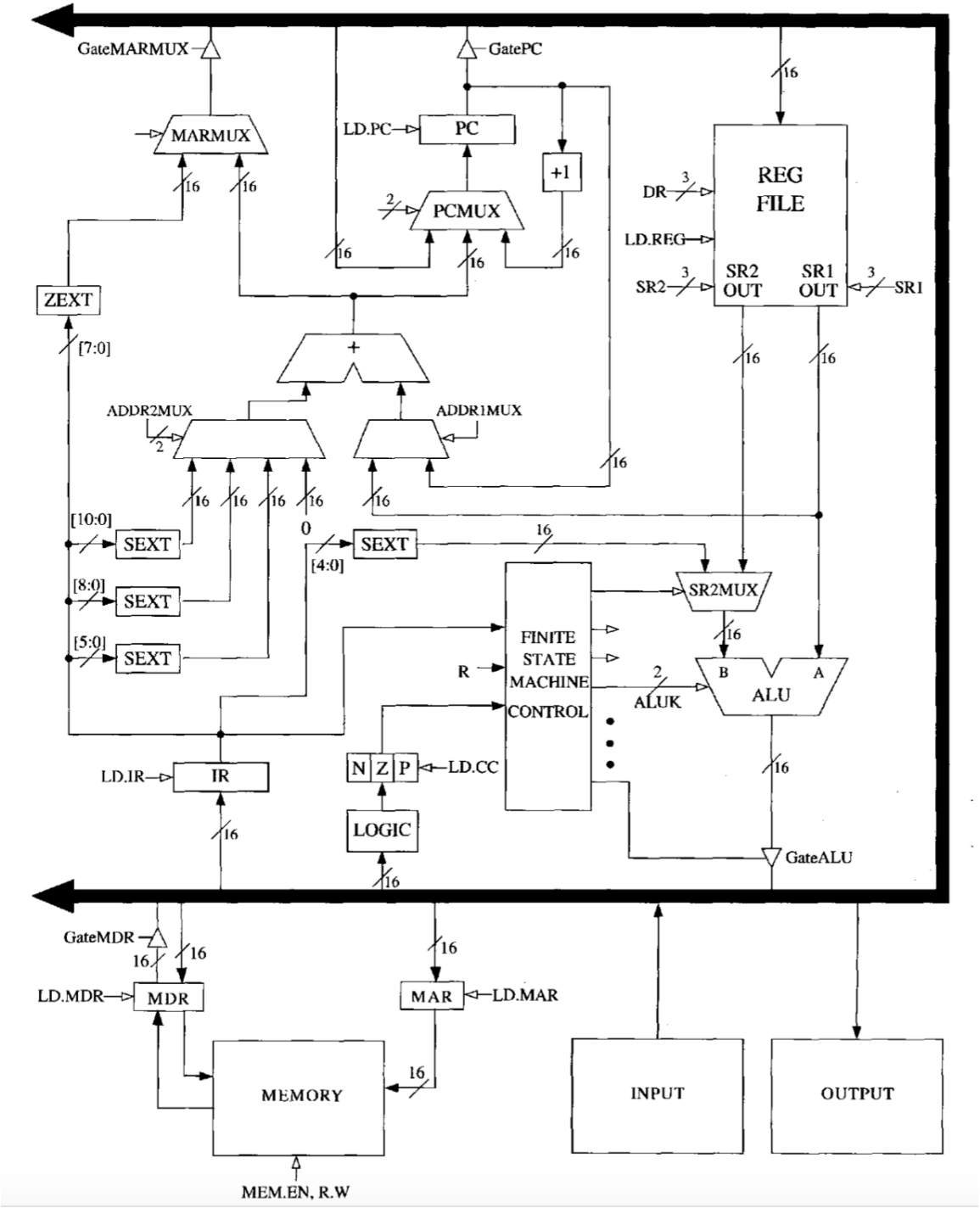
_____ (f) Any Boolean function can be calculated by a proper combination of NOR gates

_____ (g) In C, computing (14 & 19) gives the result of 2

_____ (h) The hexadecimal number 3F07 can be represented in octal as 37407

_____ (i) Sign-extending the 8-bit twos-complement integer 0xA7 to 32 bits yields 0xFFFFFFA6

_____ (j) The twos-complement of 0x0000 is 0xFFFF


GateMARMUX
GatePC
MARMUX
LD.PC
PC
+1
PCMUX
REG FILE
DR
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
ZEXT
[7:0]
+
ADDR2MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
SEXT
[4:0]
SR2MUX
FINITE STATE MACHINE CONTROL
R
ALUK
B
A
ALU
LD.IR
IR
N
Z
P
LD.CC
LOGIC
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MEMORY
INPUT
OUTPUT
MEM.EN, R.W

CS 2110 Spring 2019 Practice Final Examination
GTID#:_____