

# ECE 199 Final Exam Fall 2003

Tuesday, December 16<sup>th</sup>, 2003

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

- **Be sure your exam booklet has 11 pages.**
- **Write your name at the top of each page.**
- **This is a closed book exam.**
- **You are allowed three 8.5 x 11 sheets of notes.**
- **Absolutely no interaction between students is allowed.**
- **Show all of your work.**
- **Challenge questions are marked with \*\*\***
- **Don't panic, and good luck!**

Problem 1	20 points	_____
Problem 2	20 points	_____
Problem 3	20 points	_____
Problem 4	20 points	_____
Problem 5	20 points	_____
Total	100 points	_____

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

**Problem 1** (20 points): Short Answer

**Part A** (8 points): Briefly describe the error that this program makes. The comment reflects what the program is supposed to do. Note that there are no syntax errors.

```
/* Copy array a to array b. Each has 25 elements. */  
i = 0;  
while( i++ < 25 )  
    b[ i ] = a[ i ];
```

**Part B** (12 points): Using the basic logic structures discussed in lecture, design a circuit to convert an 8-bit 2's complement number to its absolute value (also an 8-bit 2's complement number).

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

**Problem 2 (20 Points): LC-3**

**Part A (10 points):** The two snippets of code below do the same thing. Fill in each of the three blank lines with a single instruction or statement.

int x;	LD R0, X
int *y;	AND R0, R0, #0
int z;	ST R0, X
x = 0;	.
.	.
.	.
.	LD R0, X
x++;	ADD R0, R0, #1

\_\_\_\_\_  
z = (\*y) + 1;

\_\_\_\_\_  
LEA R0, X  
ST R0, Y

\_\_\_\_\_  
ADD R0, R0, #1  
ST R0, Z

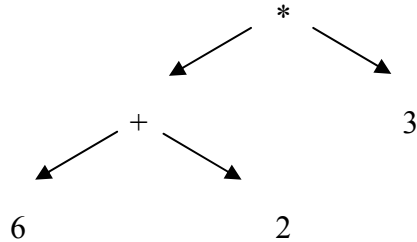
**Part B (10 points):** Write assembly code to sum up all the 16-bit integers pointed to by an array of integer pointers. Assume that the array begins at x4002, and that its size is located at x4001. After summing the values, store your result to x4000.

.ORIG x3000

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

**Problem 3 (20 Points): Recursion and I/O**

**Part A (12 points):** Write a function to evaluate a parse tree that contains only integer operands and the multiplication and addition operators. Your function must return the result of the expression. For example, the following parse tree should produce the result:  $(6 + 2) * 3 = \mathbf{24}$



```
typedef struct node Node;
struct node {
    Node *left, *right;
    int type; /* 0 = Operator, 1 = Operand */
    int value; /* If type is 0, then 0 = Addition, 1 = Multiplication
                 If type is 1, then value is just the operand's value */
};
```

```
int Evaluate( Node* root )
{
```

```
}
```

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

**Problem 3, continued:**

**Part B** (8 points): The function below reads an expression from a file and builds a tree structure for the expression. The function takes a file and a pointer to a place in which the root node of the resulting expression can be stored. The function returns 0 on success and -1 on failure. For simplicity, the function does not properly free dynamically allocated memory; ignore this problem. Read the function, then write the contents of a file representing the expression:  $(2 * (4 * 3)) + 18$ .

```
typedef struct node Node;
struct node {
    Node *left, *right;
    int type; /* 0 = Operator, 1 = Operand */
    int value; /* If type is 0, then 0 = Addition, 1 = Multiplication
                If type is 1, then value is just the operand's value */
};

int parse_expression_file( FILE *file_in, Node **n )
{
    char buf[200];
    int num;

    (*n) = (Node *) calloc( 1, sizeof( Node ) );
    if( *n == NULL || fgets( buf, 200, file_in ) == NULL ) {
        return -1;
    }
    if( sscanf( buf, "%d", &num ) == 1 ) {
        (*n)->type = 1;
        (*n)->value = num;
        return 0;
    }
    (*n)->type = 0;
    if( buf[0] == '+' ) {
        (*n)->value = 0;
    } else if( buf[0] == '*' ) {
        (*n)->value = 1;
    } else {
        return -1;
    }
    if( parse_expression_file( file_in, &(*n)->left ) == -1 ||
        parse_expression_file( file_in, &(*n)->right ) == -1 ) {
        return -1;
    }
    return 0;
}
```

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

**Problem 4 (20 Points): Reading C**

In this problem, you must describe the operation of two functions provided to you (the two are unrelated). Include in your description what the parameters and return values are, if applicable. As an example, if a function to sum all the nodes in a linked list were given, a good response would be, "This function takes a linked list and returns the sum of all of its nodes."

**Part A (10 points):**

```
int FunctionA( const char *p1, const char *p2, int num )
{
    char c1 = 0;
    char c2 = 0;

    while ( num > 0 ) {
        c1 = *p1++;
        c2 = *p2++;
        if ( c1 != c2 ) {
            break;
        }
        if ( ( c1 == '\0' ) || ( c2 == '\0' ) ) {
            break;
        }
        num--;
    }

    return ( c1 != c2 );
}
```

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

**Problem 4, continued:**

**\*\*\*Part B (10 points):**

```
typedef struct node Node;
struct node {
    int value;
    Node *next;
};

void FunctionB( Node *head, Node **headA )
{
    Node *temp = head;
    Node **tempA = headA;
    int num;

    *tempA = NULL;
    if( head != NULL ) {
        num = temp->value;
        *tempA = (Node *) malloc( sizeof( Node ) );
        (*tempA)->value = num;
        (*tempA)->next = NULL;
        tempA = &((*tempA)->next);

        while ( temp != NULL ) {
            if ( temp->value > num ) {
                num = temp->value;
                *tempA = (Node *) malloc( sizeof( Node ) );
                (*tempA)->value = num;
                (*tempA)->next = NULL;
                tempA = &((*tempA)->next);
            }
            temp = temp->next;
        }
    }
}
```

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

**Problem 5 (20 Points): Writing C**

**Part A (12 points):** Write a function that takes two lists of numbers, each of which is sorted in increasing order, and determines if the first list is a subset of the second list, returning 1 if this relationship holds, and 0 otherwise. (Recall that a list is a subset of a second list if the second list contains all elements in the first list.) You may use either a recursive or iterative style, but your function must take advantage of the fact that the lists are sorted for full credit.

```
typedef struct node Node;
struct node {
    int value;
    Node *next;
};

int sorted_list_is_subset( Node *sub, Node *super )
{
```

```
}
```

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

**Problem 5, continued:**

**Part B** (8 points): Given the structure definition and global array declaration shown below, write a function to print the names of all students who are younger than 21 and are male (sex is 'm'), assuming that all entries in the array of students contain valid data.

```
struct person {
    char name[50];
    int age;
    char sex;
};
struct person student[60];
```

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

**Scratch Paper for Calculations**

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

	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															