Final Exam CSCI 255 Spring 2001

7 May, 2001

Name: _____

This is a closed book exam. Use of calculators is also not allowed. Be sure to show your work in order to get full credit for the problem. When possible place your answers in the provided boxes. There are 11 questions for a total of 200 points on this quiz.

This exam is to be turned in by 5:45 pm.

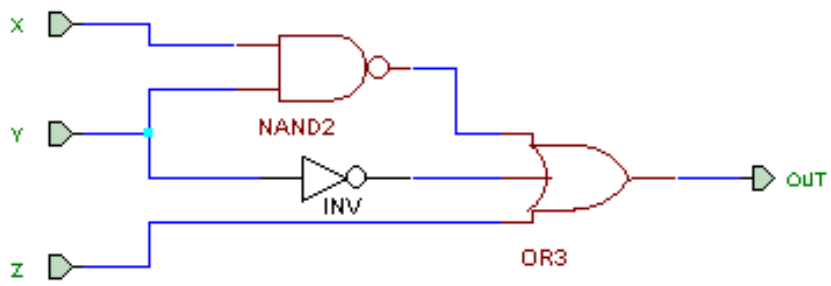
Problem 1 (10 points):

Convert the following numbers from eight-bit twos-complement notation into decimal notation.

00001111	11110000

Problem 2 (10 points):

Fill in the truth table on the right to reflect the output of the circuit on the left.



x	*y*	*z*	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Problem 3 (10 points):

Complete the following truth table for the given Boolean equation:

x	*y*	$(x + y')'+y$
0	0	
0	1	
1	0	
1	1	

Problem 4 (15 points):
Translate into LC-2 machine language (binary) the LC-2 assembly language program shown below:

```
      .ORIG     x3000

      LEA       R0,X

      AND       R1,R0,#15

      LDR       R2,R0,#2

      ADD       R3,R0,R2

      LDR       R4,R3,#1

      LDI       R5,Y

      ADD       R6,R0,#-4

      LDR       R7,R6,#5

      HALT

X     .FILL     x1

      .FILL     x2

      .FILL     x3

      .FILL     x4

      .FILL     x5

Y     .FILL     x3004

      .END
```

Problem 5 (15 points):
What are the values of registers R0 to R7 when the LC-2 assembly language program in Problem 4 is executed and reaches the `HALT` trap?

R0 =	R4 =
R1 =	R5 =
R2 =	R6 =
R3 =	R7 =

Problem 6 (40 points):
In this problem you are asked to write **five** *independent* sections of LC-2 assembly code to set registers `R0` or `R1` or LC-2 memory locations based on constants, the present values of `R3` and `R4`, or LC-2 memory locations. You may use registers `R6` or `R7` as "scratch" registers but should not modify any other registers. You must assume that your code will be located somewhere between memory locations `x3000` and `x30FF`. You may use `.fill`'s when needed to initialize memory locations. You should assume that these `.fill`'s would also be stored in memory locations `x3000` to `x30FF`.

In these subproblems, the code to implement is given in the psuedo-C notation used in class lectures. `R`*n* will be used as a reference to LC-2 register *n*. `M[`*n*`]` will be used as a reference to LC-2 memory location *n*.

```
R0 ← R4 | R5 ;
```

```
R0 ← 3*R0 + 1 ;
```

```
if (R0 > 30)
     R0 ← R0 - 30 ;
```

```
while (R0 >= 0)
     R0 ← R0 + R0 ;
```

```
M[x4100] ← M[x4100] + 5 ;
```

Problem 7 (20 points):
Suppose A is "declared" as an array of 100 uninitialized LC-2 integers with:

```
A    .blkw    100
```

Write LC-2 code to set elements A[2], A[98], and A[R5], where R5 refers to the contents of register R5, to the value 3. You may assume that the array A resides in the same page as your code, and you may use registers R0, R1, and R2 as "scratch" registers.

```
A[2] = 3 ;

A[98] = 3 ;

A[R5] = 3 ;
```

Write your answers to Problems 8 and 9 on the back of page 5.

Problem 8 (20 points):
The VAX computer has an instruction called BIC (Bit Clear) that performs the logical operation $\alpha\ \beta'$. Write an LC-2 subroutine called BIC in assembly language that performs this operation on two arguments X and Y. You should assume that BIC receives and returns its arguments on a standard LC-2 stack frame. In other words, implement the C function shown below in LC/2 assembler using the stack frame format of chapter 14.

```
int BIC(int x, int y) {
    return x & ~y;
}
```

Problem 9 (20 points):
Show how to call the LC-2 BIC subroutine of Problem 8. The two arguments passed to BIC are stored in registers R4 and R5 and the result should be stored in R3. Assume the size of the activation record of the calling routine is six words. That is, do:

```
R3 = BIC(R4, R5) ;
```

Write your answers to Problem 10 on the back of page 6.

Problem 10 (20 points):
Translate the following worthless function from C to LC-2 assembler. Use Chapter 14 style activation records to transmit parameters.

```
int dmbprg(int X, int *C, int *A) {
  int T ;
  if (X < 0)
    T = *C + 1
  else
    T = A[X] + 1 ;
  return T ;
}
```

Problem 11 (20 points):
How have the following four concepts, programs, or standards been used in CSCI 255:

combinational circuit
IA-32
make
memory-mapped I/O

Final exam reference and tally page

Number	Name
$20	GETC
$21	OUT
$22	PUTS
$23	IN
$24	PUTS
$25	HALT

Location	Name
$F3FC	CRT Status Register
$F3FF	CRT Data Register
$F400	Keyboard Status Register
$F401	Keyboard Data Register

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0001				DR			SR1			0	00		SR2		
ADD	0001				DR			SR1			1	imm5				
AND	0101				DR			SR1			0	00		SR2		
AND	0101				DR			SR1			1	imm5				
BR	0000				N	Z	P	pageoffset9								
JSR	0100				L	00		pageoffset9								
JSRR	1100				L	00		BaseR			index6					
LD	0010				DR			pageoffset9								
LDI	1010				DR			pageoffset9								
LDR	0110				DR			BaseR			index6					
LEA	1110				DR			pageoffset9								
NOT	1001				DR			SR			111111					
RET	1101				000000000000											
RTI	1000				000000000000											
ST	0011				SR			pageoffset9								
STI	1011				SR			pageoffset9								
STR	0111				SR			BaseR			index6					
TRAP	1111				0000			trapvect8								

Return value	Stored by callee
Return address	Stored by callee
Dynamic link	Stored by caller
Passed arguments	Stored by caller
Local variables, saved registers, etc.	

p. 1	/30
p. 2	/30
p. 3	/40
p. 4	/60
p. 5	/40
TOT	/200