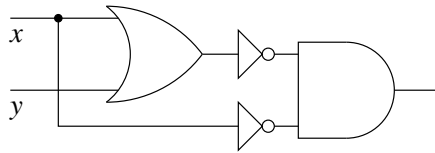


Exam 0, CSCI 150, Fall 2003

Name: _____

1. [5 pts] Write the Boolean expression most closely corresponding to the following circuit.



2. [10 pts]

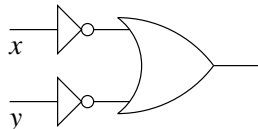
a. Give a sum-of-products Boolean expression corresponding to the truth table at right.

x	y	z	out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

b. Simplify your expression.

c. At far right, draw a circuit corresponding to your expression.

3. [5 pts] At right, draw a smaller circuit (i.e., fewer gates) accomplishing the same task as the circuit at left (which represents $\bar{x} + \bar{y}$).



4. [10 pts] Perform each of the following conversions.

- a. $35_{(10)}$ to binary
- b. $1101100_{(2)}$ to decimal
- c. $1101100_{(2)}$ to octal
- d. $1101100_{(2)}$ to hexadecimal
- e. $2D_{(16)}$ to binary

5. [5 pts] How many bits are in a kilobyte of memory?

6. [5 pts] Represent each of the following integers as specified.

- a. -1 in a 7-bit two's-complement format
- b. -36 in a 7-bit two's-complement format
- c. 36 in a 7-bit two's-complement format

7. [10 pts] Suppose we define a seven-bit floating point system with one sign bit, three exponent bits (using excess 3), and three mantissa bits.

a. Represent each of the following decimal numbers in this seven-bit system.

$$10_{(10)}$$

$$-0.25_{(10)}$$

b. For each of the following bit patterns in this seven-bit floating-point system, express its numerical equivalent as a base-10 decimal number or as a base-10 fraction.

1 110 110

0010 110

8. [10 pts] Draw a truth table diagramming a full adder's c_{out} output.

9. [10 pts] At right, draw a circuit with a single input T and a single output, where the output toggles each time T changes from 0 to 1. (Your circuit may incorporate D flip-flops.)

T	Q	explanation
0	0	
1	1	T changes to 1, so Q changes
0	1	T changes to 0; Q remains at 1
1	0	T changes to 1, so Q changes
0	0	T changes to 0; Q remains at 0
1	1	T changes to 1, so Q changes
0	1	T changes to 0; Q remains at 1
1	0	T changes to 1, so Q changes

10. [5 pts] Explain in detail what HYMN does during the fetch phase of the fetch-execute cycle. (Your explanation should describe how the computer accesses values in registers and memory.)

11. [15 pts] Suppose that HYMN begins with the following in memory.

addr	data	translation
00000	100 01001	LOAD 01001
00001	010 01000	JZER 01000
00010	110 01010	ADD 01010
00011	101 01010	STORE 01010
00100	100 01001	LOAD 01001
00101	110 00000	ADD 00000
00110	101 00000	STORE 00000
00111	001 00000	JUMP 00000
01000	000 00000	HALT
01001	000 00001	1
01010	000 00010	2
01011	000 00100	4
01100	000 00000	0

- a. In the above table, list all new values stored in memory as the program executes. Express your answers in binary or hexadecimal.
- b. What values does the AC hold in the course of executing this program? Express your answers in binary or hexadecimal.

12. [10 pts] Translate the following assembly language program into machine language.

```

up:   READ
      JZER done
      STORE n
      JUMP up
done: LOAD n
      WRITE
      HALT
n:    0

```

addr	data	addr	data
00000		00101	
00001		00110	
00010		00111	
00011		01000	
00100		01001	

code	op	behavior
000	HALT	nothing further happens (computer halts)
001	JUMP	$PC \leftarrow data$
010	JZER	if AC = 0 then $PC \leftarrow data$ else $PC \leftarrow PC + 1$
011	JPOS	if AC > 0 then $PC \leftarrow data$ else $PC \leftarrow PC + 1$
100	LOAD	$AC \leftarrow M[data]$; $PC \leftarrow PC + 1$
101	STORE	$M[data] \leftarrow AC$; $PC \leftarrow PC + 1$
110	ADD	$AC \leftarrow AC + M[data]$; $PC \leftarrow PC + 1$
111	SUB	$AC \leftarrow AC - M[data]$; $PC \leftarrow PC + 1$

13. [10 pts] Write an assembly language program that reads a number n from the user and then displays n 's absolute value. (The **absolute value** of a number is that number with any negative sign removed. The absolute value of -5 is 5, while the absolute value of 3 is 3 itself.)

14. [10 pts] Translate the following pseudocode into an assembly language program.

```
Initialize sum to 0.  
Read n.  
while  $n \neq 0$ , do:  
    Increase sum by n.  
    Read n.  
end while  
Write sum.  
Stop.
```

Solutions, Exam 0, CSCI 150, Fall 2003

Statistics

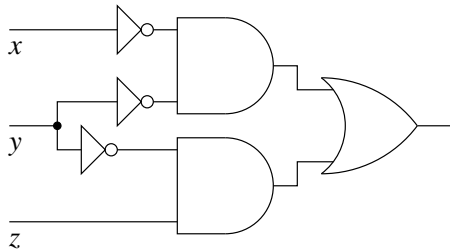
mean	86.346 (2245.000/26)
stddev	23.329
median	90.500
midrange	68.000-107.000
#1.	3.81 / 5
#2.	9.27 / 10
#3.	3.12 / 5
#4.	8.88 / 10
#5.	2.50 / 5
#6.	3.88 / 5
#7a.	3.96 / 5
#7b.	3.35 / 5
#8.	4.27 / 10
#9.	5.88 / 10
#10.	2.27 / 5
#11.	5.23 / 15
#12.	8.00 / 10
#13.	5.77 / 10
#14.	6.54 / 10
+ 10-point bonus	

1. $(\overline{x + y}) \bar{x}$

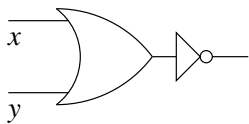
2. a. $\bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + x\bar{y}z$

b. $\bar{x}\bar{y} + \bar{y}z$

c.



3. This involves an application of DeMorgan's law $\overline{xy} = \bar{x} + \bar{y}$.



4. a. $35_{(10)} = 100011_{(2)}$

b. $1101100_{(2)} = 108_{(10)}$

c. $1101100_{(2)} = 154_{(8)}$

d. $1101100_{(2)} = 6C_{(16)}$

e. $2D_{(16)} = 101101_{(2)}$

5.

$$\frac{8 \text{ bits}}{\text{byte}} \times \frac{1,024 \text{ bytes}}{\text{KB}} \times = \frac{8,096 \text{ bits}}{\text{KB}}$$

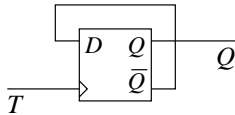
6. a. 1111111

- b. 1011100
- c. 0100100

7. a. $5_{(10)}$ 0 110 010
 $-2_{(10)}$ 1 001 000
- b. 1 110 110 $-14_{(10)}$
 0 010 110 $0.875_{(10)}$ or $\frac{7}{8}$

c_{in}	a	b	c_{out}
0	0	0	0
0	0	1	0
0	1	0	0
8. 0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

9.



10. The CPU looks in the PC for a memory address, loads from memory at that address, and stores the data found there into the IR.

11. a.

addr	data
00000	8A 8B 8C
01010	03 06 0A

b. 01 03 89 8A 03 06 8A 8B 06 0A 8B 8C 0A

12.

addr	data	translation
00000	100 01010	LOAD 11110
00001	010 01000	JZER 00100
00010	101 01011	STORE 00111
00011	001 01011	JUMP 00000
00100	100 00000	LOAD 00111
00101	101 01010	STORE 11111
00110	000 00000	HALT
00111	000 00000	0

13.

- READ
- JPOS ok
- STORE n
- SUB n
- SUB n
- ok: WRITE
- HALT
- n: 0

14.

```
LOAD zero      # Initialize sum to 0
STORE sum
READ           # Read n
STORE n
up:  LOAD n     # while n /= 0, do:
     JZER done
     LOAD sum   #      Increase sum by n.
     ADD n
     STORE sum
     READ      #      Read n
     STORE n
     JUMP up   # end while
done: LOAD sum  # Write sum.
     WRITE
     HALT     # Stop.
zero: 0
sum:  0
n:    0
```