Points missed: _____ Student's Name: ___________________________________

Total score: _____/100 points

East Tennessee State University – Department of Computer and Information Sciences
CSCI 2150 (Tarnoff) – Computer Organization – ***Section 001***
TEST 1 for Fall Semester, 2007

Read this before starting!

- The total possible score for this test is 100 points.
- This test is ***closed book and closed notes***
- ***Please turn off all cell phones & pagers during the test.***
- You may ***NOT*** use a calculator. Leave all numeric answers in the form of a formula.
- You may use one sheet of scrap paper that you must turn in with your test.
- All answers must have a box drawn around them. This is to aid the grader (who might not be me!) Failure to do so might result in no credit for answer. Example:

$$\boxed{32F1_{16}}$$

- **1 point will be deducted** per answer for missing or incorrect units when required. **No** assumptions will be made for hexadecimal versus decimal, so you should always include the base in your answer.
- If you perform work on the back of a page in this test, indicate that you have done so in case the need arises for partial credit to be determined.
- Statement regarding academic misconduct from Section 5.7 of the East Tennessee State University Faculty Handbook, June 1, 2001:

  "Academic misconduct will be subject to disciplinary action. Any act of dishonesty in academic work constitutes academic misconduct. This includes plagiarism, the changing of falsifying of any academic documents or materials, cheating, and the giving or receiving of unauthorized aid in tests, examinations, or other assigned school work. Penalties for academic misconduct will vary with the seriousness of the offense and may include, but are not limited to: a grade of 'F' on the work in question, a grade of 'F' of the course, reprimand, probation, suspension, and expulsion. For a second academic offense the penalty is permanent expulsion."

Basic Rules of Boolean Algebra

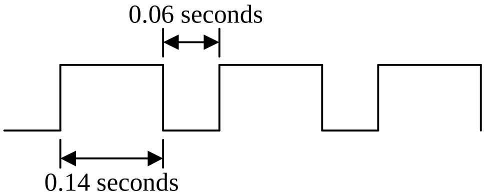
	OR	**AND**	**XOR**
Combined w/0	$A+0=A$	$A\cdot 0=0$	$A\oplus 0=A$
Combined w/1	$A+1=1$	$A\cdot 1=A$	$A\oplus 1=\overline{A}$
Combined w/self	$A+A=A$	$A\cdot A=A$	$A\oplus A=0$
Combined w/inverse	$A+\overline{A}=1$	$A\cdot\overline{A}=0$	$A\oplus\overline{A}=1$
Other rules	$A+A\cdot B=A$	$A+\overline{A}\cdot B=A+B$	$(A+B)\cdot(A+C)=A+B\cdot C$
DeMorgan's Th.	$\overline{(A\cdot B)}=\overline{A}+\overline{B}$		$\overline{(A+B)}=\overline{A}\cdot\overline{B}$

Short-ish Answer (2 points each unless otherwise noted)

1. Which unit of measurement is equivalent to (the same as) Hertz?

a.) Cycles per second b.) Percent c.) Seconds d.) Seconds per cycle e.) Cycles

2. What is the frequency of the signal show to the right?



Frequency is the inverse of the period, so the first thing we need to do is determine the period. The period is equal to the time of a full cycle, which in the figure to the right is 0.14 seconds plus 0.06 seconds. This gives us a measurement for the period of 0.2 seconds. Therefore, the answer is:

$$\text{frequency} = \frac{1}{\text{period}} = \frac{1}{0.2 \text{ seconds}} = 5 \text{ cycles/seconds} = 5 \text{ Hz}$$

You could have left your answer as 1/(0.2) Hz if you wanted to, but I wanted to see the units of Hz to be sure that you knew the units for frequency.

3. The duty cycle for the previous problem is:

a.) greater than 50% b.) equal to 50% c.) less than 50%

Remember that the duty cycle represents the percentage of time that a periodic signal is a logic 1. Officially, the expression used to determine the duty cycle is:

$$[\text{ time high } (t_h) / \text{period } (T)] \times 100\%$$

Since during a single period, the signal from problem 2 is high 0.14 seconds and low 0.06 seconds, then the signal is high more than it is low, and therefore the duty cycle is ***greater than 50%***. Officially, the duty cycle is (0.14 seconds ÷ 0.2 seconds) × 100% = 70%.

4. How many patterns of ones and zeros can be made using 9 bits?

The number of patterns of ones and zeros that can be represented with n bits is 2^n. For 9 bits, substitute 9 for n and get 2^9 which equals 512. Many people confuse the total number of combinations of ones and zeros with the maximum value that can be represented with unsigned binary notation. For that, the equation is $2^n - 1$. The error in that reasoning is that you miss the first pattern, $0_{10} = 000000000_2$.

$$\text{Ans.} = 2^9 = 512$$

5. What is the most negative value that can be stored using a 9-bit 2's complement representation?

a.) $-(2^7 - 1)$ b.) $-(2^8 - 1)$ c.) $-(2^9 - 1)$ d.) -2^7 e.) -2^8 f.) -2^9

You could have memorized the fact that the most negative number of an n-bit 2’s complement value is -2^{n-1}, which in the case of this problem is -2^8. Otherwise, you could have remembered that the most negative 2’s complement value is a 1 in the MSB followed by all zeros. For 9-bits this is 100000000. Converting this to the positive equivalent gives us, well, 100000000. This is the unsigned binary representation which equals 2^8. Making it a negative number makes it -2^8.

6. Gray code is:

   a.) a numbering system designed to best represent the color levels of a gray scale image
   b.) a representation of binary that allows for quick conversion to and from decimal
   c.) a secret language spoken only by people from Gray, TN
   d.) a binary representation meant to improve the speed with which data is stored to memory
   e.) a sequence of numbers where only a single bit changes when incrementing or decrementing through the sequence

   It seems that a few people forgot that Gray codes were going to be on the test. Sorry. Check out section 2.9 of the textbook for a refresher.

7. For each of the following applications, what would be the optimum (best) binary representation, unsigned binary (UB), 2's complement (TC), IEEE 754 Floating Point (FP), or binary coded decimal (BCD)? Identify your answer in the blank to the left of each application. (2 points each)

   ___TC___ the distance above (positive) or below (negative) sea level in feet to the nearest integer

   This needs to be a signed integer. The only signed integer provided above in the selections is 2's complement.

   ___FP___ the number of atoms in a grain of salt (a really huge number)

   This number needs to be represented in scientific notation, and therefore any of the integer representations would require far too many bits. We needed to go to floating point.

   ___BCD__ the value in dollars and cents of a financial portfolio

   Remember our discussion in class of the importance of keeping things in decimal for financial institutions? That means BCD.

8. Write the complete truth table for a 2-input NOR gate.
   Remember that the output of a NOR gate is just the inverted output of an OR gate which means that it's a 0 when any input is a 1. Some people confused this with the XOR gate. Not the same.

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

9. In the boolean expression below, circle the operation that would be performed first.

$$A + B \cdot \overline{C \cdot D}$$

10. Multiply the 16-bit value 0000110111000000_2 by 8. ***Leave your answer in 16-bit binary.*** (Hint: Remember the shortcut!) (3 points)

    Since 8 is a power of two, i.e., $8 = 2^3$, then the multiplication can be performed by simply shifting the binary number ***left*** 3 positions. This is the same as adding three zeros to the least significant side (right side) of the number. Typically, with a limited register size, left shifts result in bits being shifted OUT from the top. (The zeros in the figure below are the added digits.)

$$0110111000000\mathbf{000}_2$$

    Some people confused this with division which is what was used in the Spring 2007 test.

11. Convert 101001001001010101101_2 to hexadecimal. (3 points)

First, let's create the conversion table between binary and hex. That table is shown to the right.

Binary				Hexadecimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Once the table has been created, divide the number to be converted into nibbles. You must do this starting from the ***right side*** with the least significant bits. Starting from the left might leave you with a partial nibble on the right side. The result is shown below:

$0001\ 0100\ 1001\ 0010\ 1010\ 1101_2$

Notice that three leading zeros needed to be added. Each of these nibbles corresponds to a pattern from the table to the right. Now it just becomes a straight conversion.

$1492AD_{16}$

And yes, I did that on purpose. ☺

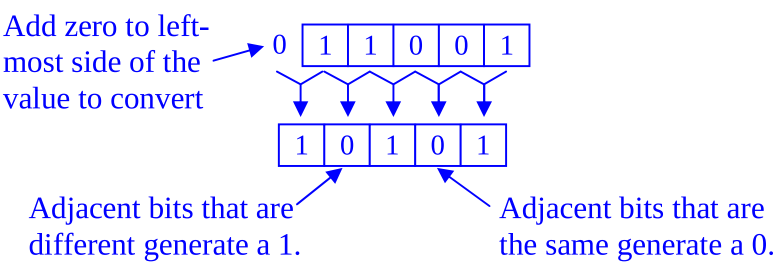
12. Convert the decimal value 86_{10} to 8-bit BCD.

This uses the identical process as the hexadecimal to binary conversion except that the conversion is from decimal to binary using the exact same table. Okay, so there are no letters in decimal, but that only means that the nibble patterns 1010, 1011, 1100, 1101, 1110, and 1111 never appear in a BCD number. 86_{10} converts to the following in BCD.

1000 0110

13. Convert the unsigned binary value 11001_2 to its corresponding 5-bit binary Gray code. (3 points)

Once again, Tarnoff thwarts us with the dreaded Gray code. Well, from page 39 of the textbook, we have the conversion sequence which says to begin by adding a leading zero to the number to be converted. For each boundary between bits, place a 1 if the bits on either side of the boundary are different and place a 0 if the bits on either side of the boundary are the same.



Therefore, the answer is 10101.

Medium-ish Answer (4 points each unless otherwise noted)

14. Convert the 32-bit IEEE 754 floating-point number 11000000110011100000000000000000 to its binary exponential format, e.g., 1.1010110 x 2^{-12}, (which, by the way, is not even close to correct).

Once again, begin by dividing up the floating-point number into its components.

S	E	F
1	10000001 = 128 + 1 = 129	10011100000000000000000

Substituting into the expression $\pm 1.F \times 2^{(E-127)}$ gives us our answer.

$$\pm 1.F \times 2^{(E-127)} = -1.100111 \times 2^{(129-127)} = -1.100111 \times 2^{2} = -110.0111$$

15. Convert 11001.101_2 to decimal. (You may leave your answer in expanded form if you wish.)

Remember that binary digits to the right of the point continue in descending integer powers relative to the 2^0 position. Therefore, the powers of two are in order to the right of the point $2^{-1} = 0.5$, $2^{-2} = 0.25$, $2^{-3} = 0.125$, and $2^{-4} = 0.0625$. (Note that 2^{-4} is not needed for this problem.)

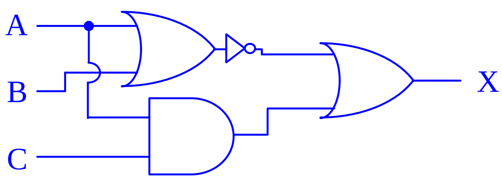
2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	1	0	0	1	1	0	1	0

Therefore, the answer is:

$$2^4 + 2^3 + 2^0 + 2^{-1} + 2^{-3} = 16 + 8 + 1 + 1/2 + 1/8 = 25.625$$

You could have left your answer in any of these three forms in order to receive full credit.

16. Draw the circuit ***exactly*** as it is represented by the Boolean expression $\overline{A+B} + A \cdot C$.



17. Prove that $A \oplus A = 0$. (Remember that $\oplus$ is the XOR or exclusive-OR)

The table below is all that is needed to prove the theorem. The important part is that both the columns for A⊕A ***and*** A are both shown so that the relationship is obvious. Basically, anything exclusive-OR'ed with itself results in an even number of ones and hence, an output of zero.

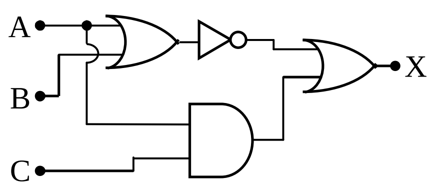
A	A⊕A
0	$0 \oplus 0 = 0$
1	$1 \oplus 1 = 0$

18. Use any method you wish to prove the rule $A + \overline{A} \cdot B = A + B$. Show all steps.

As discussed in class, the only easy way to prove this theorem is to use the truth table. The truth table needs to develop columns for both sides of the expression. Note that due to MS Word issues, the tilde ~ is used to represent the not function, i.e., NOT A = ~A.

A	B	~A	~A·B	A + ~A·B	A+B
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

19. In the space to the right, create the truth table for the circuit shown below.



A	B	C	$A+B$	$\overline{A+B}$	$A \cdot C$	$\overline{(A+B)} + A \cdot C$
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	1	0	0	0
1	0	1	1	0	1	1
1	1	0	1	0	0	0
1	1	1	1	0	1	1

20. Write the Boolean expression for the circuit shown in the previous problem. ***Do not simplify!***

The answer below is simply copied from the truth table above where the Boolean expression was derived.

$$\boxed{\overline{(A+B)} + A \cdot C}$$

Longer Answers (Points vary per problem)

21. Assume that an 8-bit binary number is used to represent an analog value in the range from 0 to 30. Convert all four of the following binary values to their analog equivalent, i.e., what analog value does each of these binary values represent? (You may leave your answer in the form of a fraction in some cases if you wish.) (5 points)

Remember that the range of bit patterns for an 8-bit binary value is distributed evenly across the analog range where all zeros represents the low end of the range and all ones represents the high end. That should immediately answer parts a and d, i.e., 00000000_2 = the lowest value, i.e., 0, and 11111111_2 = highest value, i.e., 30.

For b and c, you need to first calculate how much a single increment changes the analog value. For 8 bits, there are $2^8 - 1 = 255$ increments over a range of 0 to 30. That means that a single increment represents a difference in the analog value of (30 – 0)/255. This immediately answers part b because 00000001_2 is exactly one increment above 0 and hence represents 30/255.

Part d is the hard one. It represents $00000101_2 = 8 + 2 = 10_{10}$ increments above 0. Therefore, the analog value it corresponds to is $10 \times 30/255 = 300/255$.

a.) 00000000_2 = 0

b.) 00000001_2 = 30/255

c.) 00001010_2 = $10 \times 30/255$

d.) 11111111_2 = 30

22. Use DeMorgan's Theorem to distribute the inverse of the expression $\overline{A + B + \overline{C + D}}$ all of the way to the individual input terms. ***Do not simplify!***

$\overline{A + B + \overline{C + D}}$	There are actually a couple of ways to do this, but the easiest is to first assume the inverted OR of C and D are a single element.
$\overline{A}\cdot\overline{B}\cdot(\overline{\overline{C + D}})$	This means that the final NOT is to be distributed across A, B, and ~(C+D). Remember to distribute the inverse across the OR and change all of the OR's to AND's.
$\overline{A}\cdot\overline{B}\cdot(C + D)$	The double inverses over (C+D) will cancel.

23. Mark each Boolean expression as ***true*** or ***false*** depending on whether the right and left sides of the equal sign are equivalent. Show all of your work to receive partial credit for incorrect answers. (3 points each)

a.) $(A+B)\cdot(\overline{A}+\overline{B}) = A\cdot\overline{B} + \overline{A}\cdot B$ Answer: **True**

$A\cdot\overline{A} + A\cdot\overline{B} + B\cdot\overline{A} + B\cdot\overline{B}$	Apply F-O-I-L
$0 + A\cdot\overline{B} + B\cdot\overline{A} + 0$	Anything AND'ed w/inverse = 0
$A\cdot\overline{B} + B\cdot\overline{A}$	Anything OR'ed w/0 = itself

b.) $(A\cdot B + C)\cdot(A\cdot B + D) = A\cdot B + C\cdot D$ Answer: **True**

There are 2 ways to do this. First, you could apply the rule (A + B)·(A + C) = A + B·C substituting A·B for A, C for B and D for C. That would give you (A·B + C)·(A·B + C) = A·B + B·C. You could also start off with F-O-I-L and simplify from there.

A·B·A·B + A·B·D + A·B·C + C·D	Apply F-O-I-L
A·B + A·B·D + A·B·C + C·D	Anything AND'ed w/self = self
A·B(1 + D + C) + C·D	Pull out A·B
A·B·1 + C·D	Anything OR'ed w/1 = 1
A·B + C·D	Anything AND'ed w/1 = self

c.) $A + A \cdot B + \overline{A} \cdot C + \overline{C + A} = A$ Answer: **False**

$A + A \cdot B + \overline{A} \cdot C + \overline{C} \cdot \overline{A}$	Apply DeMorgan's Theorem
$A + \overline{A} \cdot C + \overline{C} \cdot \overline{A}$	A + A·B = A
$A + \overline{A} \cdot (C + \overline{C})$	Pull out $\overline{A}$
$A + \overline{A} \cdot 1$	Anything OR'ed w/inverse = 1
$A + \overline{A}$	Anything AND'ed w/1 = self
1	Anything OR'ed w/inverse = 1

24. Fill in the blank cells of the table below with the correct numeric format. ***For cells representing binary values, only 8-bit values are allowed!*** If a value for a cell is invalid or cannot be represented in that format, write "X". (7 points per row)

Decimal	2's complement binary	Signed magnitude binary	Unsigned binary	Unsigned BCD
130	**X**	**X**	**10000010**	**X**
68	**01000100**	**01000100**	**01000100**	**01101000**
–67	**10111101**	**11000011**	**X**	**X**

First row:

- Begin by converting the positive number to unsigned binary. We know we can do this because 8-bit unsigned binary goes up to 255. We see that 130 is made up of the powers of two $2^7 = 128$ and $2^1 = 2$.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	0	0	0	1	0

- Now, note that the unsigned (positive) representation uses the 8^{th} bit for magnitude, i.e., 128. Since the MSB is used for magnitude, it cannot be used for a sign bit. Therefore, this value cannot be represented with either 2's complement or signed magnitude representation. Put X's in those columns.
- Lastly, 8-bit BCD only goes up to 99_{10} = 10011001. 130_{10} would require 12 bits or 3 nibbles. Therefore, put an X in that column. If we had had 12 bits to represent this number, we could have represented 130 in BCD with 0001 0011 0000

Second row:

- The number represented in the 2's complement column of the second row is a positive number. We know this because the MSB is set to 0. Therefore, the binary value is the same for signed magnitude and for unsigned binary. Just copy 01000100 to the other two columns.
- Decimal: To perform the conversion process, simply convert the value to a decimal number by adding the powers of two represented by the ones in the binary value. This gives us $2^6 + 2^2 = 64 + 4 = 68_{10}$.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	1	0	0

Second row continued:

- BCD: BCD must be computed from the decimal values. It is similar to converting from hexadecimal to binary except that there are no letters A through F. Using the hexadecimal to binary conversion table shown earlier in this document, we see that 6_{10} = 0110 and 8_{10} = 1000. Therefore, the BCD column is set to 01101000.

Third row:

- First, the value is negative because in the signed magnitude representation the MSB is set to 1. Therefore, there is no unsigned magnitude or unsigned BCD representation. Put X's in those columns.
- Decimal: To begin the conversion process to decimal, we must first convert the value to a positive number by clearing the sign bit to 0. This gives us the unsigned value 01000011_2. To calculate the decimal value, add the powers of two represented by the ones in the unsigned value. This gives us $2^6 + 2^1 + 2^0 = 64 + 2 + 1 = 67_{10}$. But remember that the number is negative, so add a negative sign to get the decimal value -67_{10}.
- To calculate the 2's complement representation, take the unsigned value 01000011_2 from the previous operation, and compute the 2's complement using the shortcut we presented in class (red indicates inverted bits):

67:	0	1	0	0	0	0	1	1
–67:	1	0	1	1	1	1	0	1

This gives us 10111101.