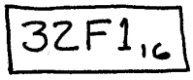
Points missed: _____ Student's Name: _____

Total score: _____ /100 points

East Tennessee State University – Department of Computer and Information Sciences
CSCI 2150 (Tarnoff) – Computer Organization
TEST 1 for Spring Semester, 2008

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. Failure to do so might result in no credit for answer. Example:



- **1 point will be deducted** per answer for missing or incorrect units when required. **No** assumptions will be made for hexadecimal versus decimal versus binary, 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.	$\overline{A\cdot B}=\overline{A}+\overline{B}$		$\overline{A+B}=\overline{A}\cdot\overline{B}$

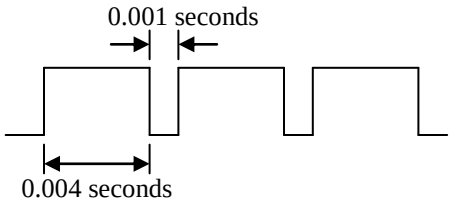
Short-ish Answer (2 points each unless otherwise noted)

1. Which unit of measurement is used to represent the period of a periodic waveform?

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

   The period of a periodic signal is a duration of time, i.e., the time it takes for a single cycle to pass. Therefore, since the only measurement offered that is time based is seconds per cycle, there is your answer.

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 0.004 seconds plus 0.001 seconds. This gives us a measurement for the period of 0.005 seconds. Therefore, the answer is:

$$\text{frequency} = \frac{1}{\text{period}} = \frac{1}{0.005 \text{ seconds}} = 200 \text{ cycles/seconds} = 200 \text{ Hz}$$

   You could have left your answer as 1/(0.005) 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.) 0% b.) 20% c.) 40% c.) 50% d.) 60% e.) 80% f.) 100%

   Before looking at this through the application of a formula, let's look at this in kind of a process of elimination way to show that this problem has more than one way of being solved. Duty cycle is the percentage of time that a signal is high, i.e., a logic 1. Looking at the diagram for problem 2, we see that the signal is high more than half the time, but not all of the time. This eliminates 0% (never high) and 100% (always high). It also eliminates 20%, 40%, and 50% because these all represent times that are less than or equal to the signal being high half of the time. This leaves only 60% and 80%. Okay, picking between these two may not be as simple. 80%, however, means that for every 10 seconds of time, the signal is high 8 seconds. In other words, for every 5 seconds, the signal is high 4 seconds. This ratio looks a lot like that shown in the figure, and therefore, E should be the answer.

   We can also look at it using the equation that represents duty cycle. 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 0.004 seconds and low 0.001 seconds, then the duty cycle is (0.004 seconds ÷ 0.005 seconds) × 100% = 80%.

4. How many patterns of ones and zeros can be made using 6 bits?

   The number of patterns of ones and zeros that can be represented with n bits is 2^n. For 6 bits, substitute 6 for n and get 2^6 which equals 64. 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, $0_{10} = 000000_2$.

   Ans. = $2^6 = 64$

5. What is the most positive value that can be stored using a 8-bit signed magnitude representation?

   a.) $2^6 - 1$ b.) $2^7 - 1$ c.) $2^8 - 1$ d.) 2^6 e.) 2^7 f.) 2^8

   You could have memorized the fact that the most positive value of an n-bit signed magnitude value is $2^{n-1} - 1$, which in the case of this problem is $2^7 - 1$. You could also, however, have figured it out by making the most positive value you could and figuring it out from that. Remember that signed magnitude using the MSB for the sign bit and the remaining n – 1 bits are used for magnitude. Therefore, for 8-bits, the largest number is 01111111_2. Converting this to decimal gives us $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$ which is one less than 2^7.

6. Which signed binary representation works for addition of both positive and negative values, signed magnitude or 2's complement?

7. True or false: A 4-bit ***Gray code sequence*** consists of 16 different patterns of ones and zeros.

   The Gray code doesn't remove any of the possible patterns of ones and zeros, it simply rearranges their order so that only one bit changes from value to value in the sequence. Therefore, 4 bits will give you $2^4 = 16$ possible patterns.

8. True or False: The number 01100101011110010111 is a valid BCD number.

   Remember that BCD is just hex without the A, B, C, D, E, or F. So begin by breaking the BCD value into its nibbles starting from the right side.

   0110 0101 0111 1001 0111

   Examining these values to see if any of them are outside the range from 0000 to 1001. None of them are, so the BCD number is valid and is equal to:

0110	0101	0111	1001	0111
6	5	7	9	7

   Note that doing the conversion was not required for this problem.

9. Is the IEEE-754 32-bit floating-point value 11111111000000001010101000000000 positive or negative?

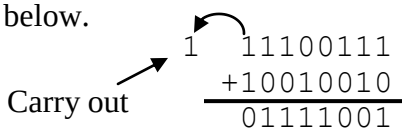
   The MSB is set to 1, and therefore, the value is negative.

10. Write the complete truth table for a 2-input NAND gate. (3 points)

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

Remember that the output of a NAND gate is just the inverted output of an AND gate which means that it's a 1 when any input is a 0, and a 0 only if all inputs are 1.

The following two questions are based on the 8-bit binary addition shown below.



11. True or False: If the addition above is considered 8-bit 2's complement, an overflow has occurred.

True. If two negative numbers are added together to result in a positive number, an overflow has occurred. Remember that the only way a 2's complement can have an overflow is if the sign bits of the numbers that are being added are the same as each other, but different from the sign bit of the result.

12. True or False: If the addition above is considered 8-bit unsigned, an overflow has occurred.

True. If a non-zero carry occurs in unsigned addition, an overflow has occurred.

13. In the boolean expression below, circle the ***single*** operation that would be performed first.

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

14. Divide the 16-bit value 0000110111000000_2 by 16. ***Leave your answer in 16-bit binary.*** (Hint: Remember the shortcut!)

Since 16 is a power of two, i.e., $16 = 2^4$, then the division can be performed by simply shifting the binary number right 4 positions. This is the same as removing the four least significant digits from the right side of the number. In general, the digits removed on the right from a right shift are replaced by duplicating the sign bit on the left, e.g., if the first bit of the number is a 0, fill in with leading zeros on the left.

$$0000110111000000_2 \rightarrow 0000000011011100_2$$

If you want to do things the hard way, you could have converted the numbers. 0000110111000000_2 equals $2^{11} + 2^{10} + 2^8 + 2^7 + 2^6 = 2048 + 1024 + 256 + 128 + 64 = 3520_{10}$. $3520/16 = 220_{10}$. 220_{10} equals $2^7 + 2^6 + 2^4 + 2^3 + 2^2 = 128 + 64 + 16 + 8 + 4$ which means in binary it equals 11011100_2. But who in their right mind would want to do such a thing? ☺

15. Convert 1100100001101111011111_2 to hexadecimal.

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:

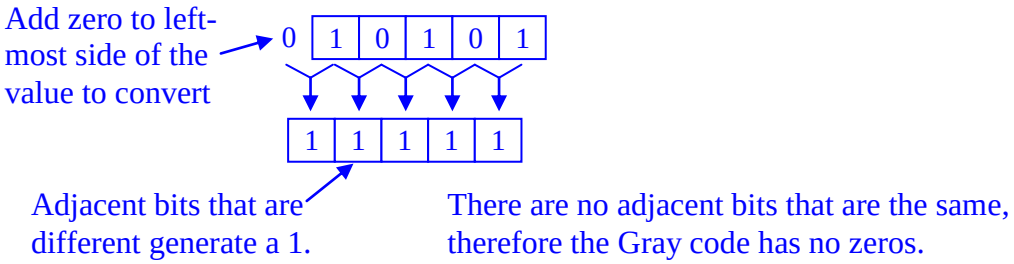
$0001\ 1001\ 0000\ 1101\ 1110\ 1111_2$

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.

$190DEF_{16}$

16. Convert the unsigned binary value 10101_2 to its corresponding 5-bit binary Gray code. (3 points)

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 11111.

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

17. Convert the 32-bit IEEE 754 floating-point number 01000001100011100000000000000000 to its binary exponential format, e.g., 1.1010110×2^{-12}, (which, by the way, is not even close to correct).

Begin by dividing up the floating-point number into its components.

S	E	F
0	10000011 = 128 + 2+ 1 = 131	00011100000000000000000

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

$$\pm 1.F \times 2^{(E-127)} = +1.000111 \times 2^{(131-127)} = 1.000111 \times 2^4 = 10001.11 = 17.75$$

The last step, converting to 10001.11 = 17.75 was not necessary.

18. Convert 1100.011_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.)

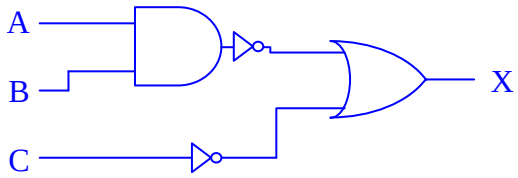
2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
0	1	1	0	0	0	1	1	0

Therefore, the answer is:

$$2^3 + 2^2 + 2^{-2} + 2^{-3} = 8 + 4 + 1/4 + 1/8 = 12.375$$

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

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



20. Prove that $A \oplus 1 = \overline{A}$. (Remember that $\oplus$ is the XOR or exclusive-OR) For full credit, please show all steps in detail.

The table below is all that is needed to prove the expression is true. The important part is that both the columns for A⊕1 ***and*** not-A (shown as ~A in the table) are both shown so that the relationship is obvious. Basically, anything exclusive-OR'ed with 1 results in an inverse.

A	~A	A⊕1
0	1	0 ⊕ 1 = 1
1	0	1 ⊕ 1 = 0

21. Use any method you wish to prove the rule $A \cdot B + \overline{A} \cdot B = B$. For full credit, please show all steps.

First, let's prove this using boolean algebra.

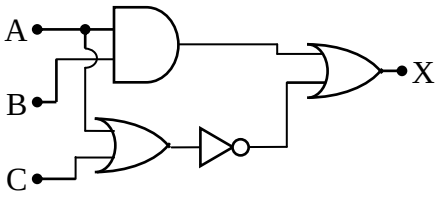
```
A·B + Ā·B

B·(A + Ā)                    Distributive law

B·1                          Anything OR'ed w/inverse is 1

B                            Anything AND'ed w/1 is self
```

This can also be proved using the truth table. The truth table needs to develop columns for both sides of the expression. Note once again 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·B	A·B + ~A·B	B
0	0	1	0	0	0	0
0	1	1	0	1	1	1
1	0	0	0	0	0	0
1	1	0	1	0	1	1

Therefore, the two sides of the expression are equal.

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



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

23. 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.

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

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

$\overline{\overline{A \cdot B} \cdot C}$	Start by distributing the inverse across the ^(AB) term and the C term.
$\overline{\overline{A \cdot B}} + \overline{C}$	Now the double inverses over (AB) will cancel.
$A \cdot B + \overline{C}$	And we get the final answer.

25. Assume that an 8-bit binary number is used to represent an analog value in the range from 20 to 95. 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.)

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., 20, and 11111111_2 = highest value, i.e., 95.

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 20 to 95 = 75. That means that a single increment represents a difference in the analog value of (95 – 20)/255. This immediately answers part b because 00000001_2 is exactly one increment above 20 and hence represents 20 + 75/255. Part c is the hard one. It represents $00000110_2 = 4 + 2 = 6_{10}$ increments above 20. Therefore, the analog value it corresponds to is 20 + (6 × 75/255).

a.) $00000000_2 = 20$

b.) $00000001_2 = 20 + (75/255)$

c.) $00000110_2 = 20 + (6 \times 75/255)$

d.) $11111111_2 = 95$

Longer Answers (Points vary per problem)

26. 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.) $\overline{A} \cdot (\overline{A} + \overline{B}) + A \cdot (A \cdot B) = 1$ Answer: **False**

$\overline{A} \cdot \overline{A} + \overline{A} \cdot \overline{B} + A \cdot A \cdot B$	Distribute ^A
$\overline{A} + \overline{A} \cdot \overline{B} + A \cdot B$	$\overline{A} \cdot \overline{A} = \overline{A}$ and $A \cdot A \cdot B = A \cdot B$
$\overline{A} + A \cdot B$	$A + A \cdot B = A$ where $\overline{A} \rightarrow A$ and $\overline{B} \rightarrow B$
$\overline{A} + B$	$A + \overline{A} \cdot B = A + B$ where $A \rightarrow B$

b.) $\overline{A}B\overline{C}D + A\overline{C}DE + \overline{C}D\overline{E} + \overline{C}DE = \overline{C}DE$ Answer: **False**

$\overline{C}\cdot D\cdot(\overline{A}\cdot B + A\cdot E + \overline{E} + E)$	Pull out a ^C·D
$\overline{C}\cdot D\cdot(\overline{A}\cdot B + A\cdot E + 1)$	Anything OR'ed w/inverse equals 1
$\overline{C}\cdot D\cdot(1)$	Anything OR'ed w/1 equals 1
$\overline{C}\cdot D$	Anything AND'ed w/1 equals self

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

$B\cdot(\overline{A} + \overline{B} + \overline{C})$	Apply DeMorgan's Theorem
$B\cdot\overline{A} + B\cdot\overline{B} + B\cdot\overline{C}$	Multiply B through
$B\cdot\overline{A} + 0 + B\cdot\overline{C}$	Anything AND'ed w/inverse is 0
$B\cdot\overline{A} + B\cdot\overline{C}$	Anything OR'ed w/0 equals self
$B\cdot(\overline{A} + \overline{C})$	Pull out B

27. 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
24	**00011000_2**	**00011000_2**	**00011000_2**	**00100100**
-70	**10111010_2**	**11000110**	**X**	**X**
38	**00100110**	**00100110_2**	**00100110_2**	**00111000**

First row:

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

- Begin by converting the positive number 24 to unsigned binary. We know we can do this because 8-bit unsigned binary goes up to 255. We see that 24 is made up of the powers of two $2^4 = 16$ and $2^3 = 8$.
- Since 24 is positive, it has the same representation in all three binary formats. Also, note that the unsigned (positive) representation has a zero in the MSB which means it isn't too large to be represented with either of the signed representations. Simply copy 00011000_2 to 2's complement, signed magnitude, and unsigned binary.
- Eight-bit BCD uses a nibble to represent 2 and a nibble to represent 4. Get these nibbles from the hex to binary table from problem 15 to get 00100100.

Second row:

- The number represented in the signed magnitude binary column of the second row is a negative number. We know this because the MSB is set to 1. Therefore, we need to find the magnitude by changing the MSB to 0, then putting a negative sign in front of the result to see what it represents.

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

  Therefore, the magnitude is $2^6 + 2^2 + 2^1 = 64 + 4 + 2 = 70$. Therefore, the decimal value is –70.
- For the 2's complement representation, we take the positive magnitude, and do the bit-flipping thing.

70:	0	1	0	0	0	1	1	0
–70:	1	0	1	1	1	0	1	0

  So the final 2's complement value is 10111010_2.
- There are no negative values in unsigned binary or unsigned BCD. Just put X's in those columns

Third row:

- 00100110_2 in 2's complement is a positive number since the MSB is 0. Therefore, the same bit pattern is used for signed magnitude and unsigned binary. Just copy 00100110_2 to those two columns.
- For the decimal, add the powers of two represented by the positions of the ones 00100110_2. This gives us $2^5 + 2^2 + 2^1 = 32 + 4 + 2 = 38_{10}$.
- For BCD, we must use the decimal value to start with. Convert the 3 to the nibble 0011 and the 8 to the nibble 1000. Remember that these come from the hex to binary table from problem 15. The final answer is 00111000.