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, 2009

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.
- Please draw a box around your answers. This is to aid the grader. 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 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.	$\overline{A\cdot B}=\overline{A}+\overline{B}$		$\overline{A+B}=\overline{A}\cdot\overline{B}$

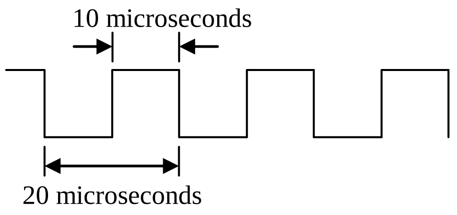
Short-ish Answer (2 points each unless otherwise noted)

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

   a.) Seconds b.) Percent [✓ c.) Hertz] d.) Seconds per cycle e.) Cycles

   Frequency is a measure of how fast the cycles of a signal are coming at you, i.e., a measure of the number of cycles over a period of time. Typically, it's a measure of the number of cycles that occur in a second, i.e., cycles per second. Another name for this is Hertz.

2. What is the frequency of the signal show to the right? (Note: 1 microsecond = 1×10^{-6} seconds)



   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 20 microseconds. (Some people added the 10 microseconds to the 20 microseconds, but note that the 20 microseconds is the period and the 10 microseconds is the pulse width which is included in the period.) Therefore, the answer is:

$$\text{frequency} = \frac{1}{\text{period}} = \frac{1}{20 \times 10^{-6}\text{ sec}} = 50{,}000\text{ cycles/seconds} = 50\text{ kHz}$$

   You could have left your answer as $1/(20 \times 10^{-6})$ 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.) 25% b.) 33% c.) 40% [✓ c.) 50%] d.) 66% e.) 75% f.) 100%

   Before doing this mathematically, let's look at this by beginning with the definition of duty cycle. ***The duty cycle is the percentage of time that a signal is high, i.e., a logic 1.*** Looking at the diagram for this problem we see that the signal is high exactly half of the time or 50%. Therefore, C 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 the problem is high 10 microseconds during a period of 20 microseconds, then the duty cycle is (10 μseconds ÷ 20 μseconds) × 100% = 50%.

4. How many patterns of 1's and 0's can be made using 7 bits?

   If there are 2 ways that each of the 7 bits can be set, then there are $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^7$ = 128 different ways 1's and 0's can be set with 7 bits. If you answered $2^7 - 1 = 127$, you were giving me the largest value that can be represented with 7 bits in unsigned binary notation. The reason this isn't right is that one of the patterns is $0000000_2 = 0_{10}$.

5. You can quickly determine whether a number is positive or negative in the signed representations 2's complement, signed magnitude, and IEEE-754 floating-point by simply looking at the _____.

[✓ a.) most significant bit] b.) least significant bit c.) number of bits

6. What is the most positive value that can be stored using 10-bit unsigned binary representation?

a.) 2^{11} b.) $2^{11} - 1$ c.) 2^{10} d.) $2^{10} - 1$ e.) 2^9 f.) $2^9 - 1$

There are a number of ways to calculate this. First, we know that for ***unsigned binary representation***, the largest binary value is all 1's. For ten bits this is 1111111111_2. Converting this to decimal equals $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 + 2^9 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512 = 1023 = 2^{10} - 1$. Another way to look at this is to see that the next highest binary value is a 1 with 10 zeros after it, i.e., 10000000000_2 which equals 2^{10}. One less than that, therefore, is $2^{10} - 1$.

7. What is the most negative value that can be stored using 9-bit signed magnitude representation?

a.) -2^{10} b.) $-(2^{10} - 1)$ c.) -2^9 d.) $-(2^9 - 1)$ e.) -2^8 [✓ f.) $-(2^8 - 1)$]

Signed magnitude representation uses the most significant bit as a sign flag and the rest as magnitude. Therefore, the most negative value that can be stored in 9-bit signed magnitude representation is a 1 for the negative sign and eight more 1's for the magnitude. Using the same methodology we used in problem 6 to figure out the magnitude of eight 1's we get that the magnitude is $2^8 - 1$. Therefore, the most negative value is $-(2^8 - 1)$.

8. What is the minimum number of bits needed to represent 1250_{10} in unsigned binary representation?

a.) 8 b.) 9 c.) 10 [✓ d.) 11] e.)12 f.) 13

This is yet another problem that could have been done a couple of different ways. First, you could have memorized the formula for the largest possible unsigned binary value is $2^n - 1$ where n is the number of bits. Trying a couple of values for n might have resulted in the table to the right. The table shows us that 11 bits is the first level that has a high enough maximum value.

n	$2^n - 1$
8	255
9	511
10	1023
11	2047

Another way to do this is to realize that the largest unsigned binary value is all ones. We can create a table similar to the previous one for every unsigned binary value with all ones.

Binary Value	Decimal Value
11111111_2	255
111111111_2	511
1111111111_2	1023
11111111111_2	2047

Once again, we see that 11 bits is the threshold where the representation of 1250_{10} is possible.

9. The IEEE-754 32-bit floating-point value 11000000100101000000000000000000 is ________.

a.) positive [✓ b.) negative]

The MSB is the sign bit, and in this case it is a 1 which means it is a negative value.

10. Write the complete truth table for a 2-input XOR gate. (3 points) $\longrightarrow$

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

Remember that the output of a XOR gate is one if the number of ones at the inputs is an odd number. For two inputs, the only times this happens is for A=0, B=1 and A=1, B=0.

11. Perform the binary addition shown below.

```
1  1 1 1 1
     1 0 1 1 0 0 1 0
   + 0 1 0 1 0 0 0 1
   -----------------
 1   0 0 0 0 0 0 1 1
```

12. [✓ True] or false: The above addition works the same regardless of whether the numbers are in 2's complement notation or unsigned binary notation. (1 point)

True: Regardless of whether the binary values represent unsigned binary or 2's complement, the mechanics of addition are exactly the same. Only the interpretation of the results is different, and in that case only when the MSB is set to one indicating a negative result.

13. True or [✓ false] If the above addition is in 8-bit unsigned binary notation, the result can be contained in 8 bits. (1 point)

As is indicated by the carry out of the 8th bit position into the 9th, treating this addition as an unsigned binary addition would mean the result would require 9 bits.

14. [✓ True] or false: If the above addition is in 8-bit 2's complement notation, the result can be contained in 8 bits. (1 point)

In 2's complement notation, the carry out of the 8th bit represents an adjustment for the sign, not a magnitude bit. Overflows in 2's complement are detected by checking to see if two positive numbers have been added together to result in a negative number or if two negative numbers have been added together to result in a positive number. In this case, there isn't a problem.

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

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

Remember that when spanning more than one input, inverses act like parenthesis. This means that the AND of C and D underneath the inverse must be performed first.

16. Divide the 16-bit value 0000010110100000_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 thing as adding deleting the 4 least significant zeros on the right hand side of the number and adding 4 zeros to the most significant side of the number. The additional bits are shown below in red. The zeros are added on the left side because the processor has a fixed number of bits used to store the value.

0000000001011010_2

17. Convert 1100000110100010101_2 to hexadecimal.

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

Create the conversion table between binary and hex. (That table is shown to the right.) Next, partition 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 correct result is shown below:

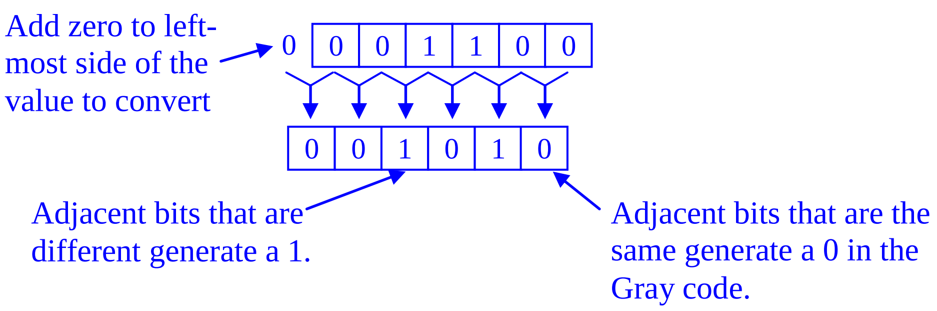
$0110\ 0000\ 1101\ 0001\ 0101_2$

Notice that one leading zero (in red) is needed on the left side. Each of these nibbles corresponds to a pattern from the table to the right. Now it just becomes a straight conversion.

$\mathbf{60D15_{16}}$

18. Convert the unsigned binary value 001100_2 to its corresponding 6-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.



Add zero to left-most side of the value to convert
0
0 0 1 1 0 0
0 0 1 0 1 0
Adjacent bits that are different generate a 1.
Adjacent bits that are the same generate a 0 in the Gray code.

Therefore, the answer is 001010.

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

19. Convert the 32-bit IEEE 754 floating-point number 11000000000110100000000000000000 to its binary exponential format, e.g., -1.001010×2^{-12}, (which, by the way, is not the answer).

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

S	E	F
1	10000000 = 128	00110100000000000000000

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

$$\pm 1.F \times 2^{(E-127)} = -1.001101 \times 2^{(128-127)} = -1.001101 \times 2^{1} = -10.01101 = -2.40625$$

The last two steps, converting to –10.01101 and then to –2.40625 were not necessary.

20. Convert 0110.011_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	0	1	1	0	0	1	1	0

Therefore, the answer is:

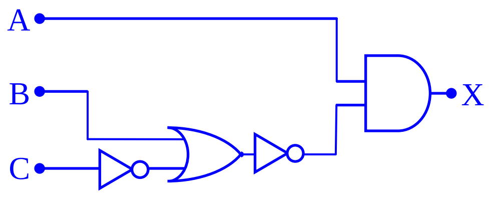
$$2^2 + 2^1 + 2^{-2} + 2^{-3} = 4 + 2 + 1/4 + 1/8 = 6\ 3/8 = 6.375$$

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

By the way, there is another way to do this problem. Remember that multiplying by powers of two shift a binary value left. If we multiply 0110.011 by $2^3 = 8$, we get 0110011_2 which is simply an unsigned binary integer. Converting 0110011_2 to decimal gives us $2^5 + 2^4 + 2^1 + 2^0 =$ $32 + 16 + 2 + 1 = 51$. Now let's put it back to the decimal value by dividing by 8. $51 \div 8 = 6.375$.

21. In the space to the right, draw the circuit ***exactly*** as it is represented by the following Boolean expression:

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

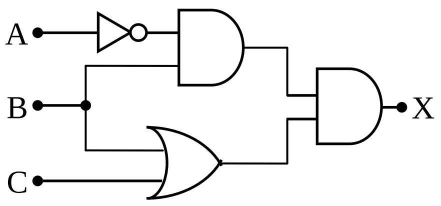


22. Prove the rule $A \oplus 1 = \overline{A}$. (Remember that $\oplus$ is the XOR or exclusive-OR) ***For full credit, please show <u>all</u> steps in detail.***

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*** ~A (~A represents "not A") are both shown so that the data sources are obvious. Basically, anything exclusive-OR'ed with 1 equals its inverse.

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

23. In the space to the right, create the truth table for the circuit shown below. (6 points)



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

24. 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.

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

25. Use DeMorgan's Theorem to distribute the inverse of the expression $\overline{A+(B\cdot C)+D}$ all of the way to the individual input terms. Be careful of the AND and OR precedence. It might help to see the expression as a logic circuit. ***Do not simplify!***

$\overline{A+(B\cdot C)+D}$	Start by distributing the inverse across the three OR'ed terms A, (B·C), and D. This will flip the + (OR) functions to · (AND's).
$\overline{A}\cdot(\overline{B\cdot C})\cdot\overline{D}$	As far as A and D are concerned, the inverses have been distributed all the way to the inputs. The B·C term, however, still needs to have the inverses distributed to the inputs. Note that it is vitally important to maintain the parenthesis!
$\overline{A}\cdot(\overline{B}+\overline{C})\cdot\overline{D}$	This gives us our final answer. If you were not able to keep B and C together with the parenthesis, I took off points. If the parenthesis disappeared, order of precedence would have A and B AND'ed and C and D AND'ed before performing the OR.

26. Assume that an 8-bit binary number is used to represent an analog value in the range from 10 to 20. 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., 10, and 11111111_2 = highest value, i.e., 20.

For b and c, you need to calculate how much a single increment changes the analog value. For 8 bits, there are $2^8 - 1 = 255$ increments over a range from 10 to 20 = 10. That means that a single increment represents a difference in the analog value of 10/255. This immediately answers part b because 00000001_2 is exactly one increment above 10 and hence represents 10 + 10/255. Part c is the hard one. In unsigned notation, c represents $00000110_2 = 4 + 2 = 6_{10}$ increments above 10. Therefore, the analog value it corresponds to is $10 + (6 \times 10/255)$.

a.) $00000000_2 = 10$

b.) $00000001_2 = (1\times10)/255 + 10$

c.) $00000110_2 = (6\times10)/255 + 10$

d.) $11111111_2 = 20$

Longer Answers (Points vary per problem)

27. 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.) $\overline{A \cdot B} + A = 1$ Answer: **True**

$\overline{A} + \overline{B} + A$	Apply DeMorgan to A·B
$\overline{A} + A + \overline{B}$	Rearrange using Associative Law
$1 + \overline{B}$	Anything OR'ed w/inverse equals 1
1	Anything OR'ed w/1 equals 1

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

```
(A̅ + B̅)·(A + B)              Apply DeMorgan to A·B
A̅·A + A̅·B + B̅·A + B̅·B        Apply F.O.I.L.
0 + A̅·B + B̅·A + 0            Anything AND'ed w/inverse equals 0
A̅·B + B̅·A                    Anything OR'ed w/0 equals itself
```

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

$\overline{(A \cdot B \cdot C)} \cdot \overline{\overline{(A \cdot B \cdot C)}}$ Apply DeMorgan across OR

$\overline{(A \cdot B \cdot C)} \cdot (A \cdot B \cdot C)$ Double inverses cancel

0 Anything AND'ed w/inverse equals 0

28. 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
83	**01010011**	**01010011**	**01010011**	**10000011**
137	**X**	**X**	**10001001**	**X or 000100110111**
–23	**11101001**	**10010111**	**X**	**X**

Negative values: Let's take care of the negative values first. Note that the last row is a negative value, i.e., –23. Therefore, there should be X's in the unsigned binary and unsigned BDC columns for this row. An easy 3 points.

Out of range values: The second row has an unsigned binary value of 10001001. This value uses the MSB for magnitude and therefore cannot be used as a 0 (positive) sign bit for either of the signed representations, i.e., twos complement and signed magnitude. Therefore, X's must go in these two columns for this row. Another easy 3 points.

In-range positive values: The first row has a signed magnitude value where the first bit equals 0. Therefore, it is a positive value and can simply be copied to both of the other binary representations, 2's complement binary and unsigned binary. Four more points.

Converting from negative decimal: Now we need to start doing some binary to decimal conversions. Let's start with the -23. Conversion for any negative value must begin first with the positive representation. Therefore, begin converting –23 to binary by starting with 23. Breaking 23 into its powers of 2 components gives us 23 = 16 + 4 + 2 + 1 = $2^4 + 2^2 + 2^1 + 2^0$. Therefore, the binary value we get is the one shown in the table below.

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

Since the MSB is 0, the magnitude of 23 is small enough to be converted to 8-bit twos complement and signed magnitude.

For the 2's complement representation, we take the positive magnitude, and do the bit-flippy thing. This is shown in the table below where the blue numbers are just copied down and the red numbers are the inverted bits.

112:	0	0	0	1	0	1	1	1
–112:	1	1	1	0	1	0	0	1

So the final 2's complement value is 11101001_2.

To convert 00010111_2 to signed magnitude, simply flip the MSB (sign bit) to get 10010111_2.

Converting from unsigned binary to decimal: The top two rows are both in unsigned binary representation. To convert, we only need to add all the powers of two that correspond to each of the bit positions that contain ones. Let's start with the first row which contains a binary 01010011_2.

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

Therefore, 01010011_2 equals $2^6 + 2^4 + 2^1 + 2^0$ = 64 + 16 + 2 + 1 = 83. Converting 83 to BCD is done by using the first ten rows of the hex-to-binary conversion table from problem 17. This means that 8_{10} is equal to 1000 in BCD and 3_{10} is equal to 0011 in BCD. This gives us the 8-bit BCD value 10000011.

The second row in unsigned binary is 10001001_2.

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

Therefore, 10001001_2 equals $2^7 + 2^3 + 2^0$ = 128 + 8 + 1 = 137. 137_{10} cannot be representedin in 8-bit BCD, but just in case you converted it to 12-bit BCD, here is the answer. Converting 137 to BCD is done by using the first ten rows of the hex-to-binary conversion table from problem 17. This means that 1_{10} is equal to 0001 in BCD, 3_{10} is equal to 0011 in BCD, and 7_{10} is equal to 0111 in BCD. This gives us the 12-bit BCD value 000100110111.