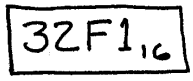
Points missed: _____ Student's Name: _____

Total score: _____ /100 points

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




- **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. What is the period of a signal with a frequency of 125 MHz? ***Be sure to include the appropriate units for period!*** (Note: MHz means Megahertz = $10^6$ Hertz)

   Frequency is the inverse of the period. Frequency is measured in Hertz or cycles/second whereas period is measured in seconds/cycle or just seconds. (Notice the comment about being sure to include appropriate units for the period in bold.) Therefore, the answer is:

   $$\text{period} = \frac{1}{\text{frequency}} = \frac{1}{125 \times 10^6 \text{ Hz}} = \frac{1}{125 \times 10^6} \text{ seconds}$$

   You could have left your answer as $1/(125\times10^6)$ seconds if you wanted to, but I needed to see the units of seconds to be sure that you knew the units for the period.

2. If the period of a pulse train is 100 mS and its duty cycle is 30%, what is the signal's pulse width?

   Duty cycle is the percentage of time that a signal is high, and with respect to the period, it is the percentage of the pulse width ($t_w$) over the period. Therefore, for a pulse train with a duty cycle of 30%, the pulse width is 30% of the period.

   $$t_w = 30\% \times 100\text{mS} = 0.3 \times 100\text{mS} = 30 \text{ mS}$$

3. How many patterns of ones and zeros can be made using 12 bits?

   The number of patterns of ones and zeros that can be represented with n bits is $2^n$. For 12 bits, substitute 12 for n and get $2^{12}$. 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, $0_{10} = 000000000_2$.

   $$\text{Ans.} = 2^{12}$$

4. What is the most positive value that can be stored using a 12-bit 2's complement representation?

   a.) $2^{10} - 1$ b.) $2^{11} - 1$ c.) $2^{12} - 1$ d.) $2^{10}$ e.) $2^{11}$ f.) $2^{12}$

   The most positive value in 2's complement is the same as the maximum value for unsigned binary notation except that you remove the sign bit. In other words, positive values in a 12-bit 2's complement value are the same as those for 11-bit unsigned binary notation. Since the maximum unsigned binary notation is $2^n - 1$ for n bits, then for 11 bits it is:

   $$2^{11} - 1$$

5. What is the ***minimum*** number of bits needed to represent $512_{10}$ using unsigned binary representation?

a.) 6 b.) 7 c.) 8 d.) 9 e.) 10 f.) 11

The easiest way to do this is just figure out what the maximum values are for each number of bits:

8 bits → $2^8 - 1 = 255$
9 bits → $2^9 - 1 = 511$
10 bits → $2^{10} - 1 = 1023$
11 bits → $2^{11} - 1 = 2047$
12 bits → $2^{12} - 1 = 4095$
13 bits → $2^{13} - 1 = 8191$

It looks like 9 is not enough, but 10 is.

6. 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)

**FP** Avogadro's number ($6.02214199 \times 10^{23}$)

**TC** Temperature in degrees (rounded to nearest integer)

**UB** A person's weight (rounded to the nearest integer)

Note that negative numbers are needed in this case.

7. True or False: The ***magnitude*** of the 32-bit IEEE 754 floating-point number 11000110100100111011011010110010 is greater than 1, i.e., the exponent is non-negative.

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

| S | E | F |
|---|---|---|
| 1 | 10001101 = 128 + 8 + 4 + 1 = 141 | 00100111011011010011001 |

Since E is greater than 127, the exponent in the expression shown below will be greater than 0.

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

8. Write the complete truth table for a 2-input XOR gate.

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

9. Divide the 16-bit value $0000011010100000_2$ by 8. ***Leave your answer in binary.*** (Hint: Remember the shortcut!) (3 points)

Since 8 is a power of two, i.e., $8 = 2^3$, then the division can be performed by simply shifting the binary number right 3 positions. This is the same as removing the three least significantdigits from the right side of the number. In general, the digits removed on the right from a right shift are replaced with leading zeros.

$$0000011010100\text{~~000~~}_2 \rightarrow 0000000011010100_2$$

If you want to do things the hard way, you could have converted the numbers. $0000011010100000_2$ equals 1024 + 512 + 128 + 32 = 1696. $0000000011010100_2$ equals 128 + 64 + 16 + 4 = 212 which is equal to 1696 divided by 8.

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

$$A + B \odot C$$

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

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

12. Convert $0001010110100001 1_2$ 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:

$0000\ 0010\ 1011\ 0100\ 0011_2$

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 1 to 1 conversion.

**$02B43_{16}$**

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

13. Convert the 32-bit IEEE 754 floating-point number 10111110010011001011000000000000 to its binary exponential format, e.g., $1.1010110 \times 2^{-12}$, (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 | 01111100 = 64 + 32 + 16 + 8 + 4 = 124 | 10011001011000000000000 |

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

$$\pm 1.F \times 2^{(E-127)} = -1.10011001011 \times 2^{(124-127)} = -1.10011001011 \times 2^{-3} = -0.00110011001011$$

14. Convert $1011.011_2$ 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 $2^{-4} = 0.0625$.

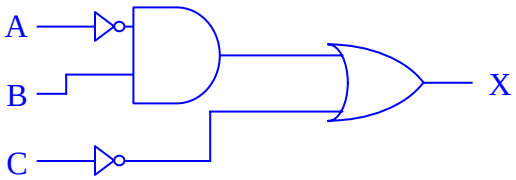
| $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

Therefore, the answer is:

$$2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} = 8 + 2 + 1 + 1/4 + 1/8 = 11.375$$

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

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




16. Prove that $A \oplus 1 = \overline{A}$ . (Remember that $\oplus$ is the XOR or exclusive-OR)

The table below proves the above theorem. The important part is that both the columns for A⊕1 ***and*** A inverse are both shown. This is necessary to show equality.

| A | A⊕1 | $\overline{A}$ |
|---|---|---|
| 0 | 0 ⊕ 1 = 1 | 1 |
| 1 | 1 ⊕ 1 = 0 | 0 |

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

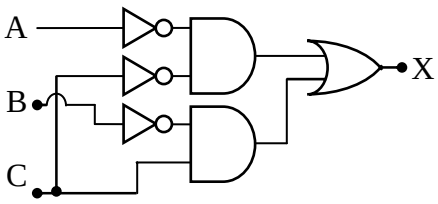
First, the rule can be proven using Boolean algebra.

| | | |
|---|---|---|
| A + A·B | = A(1 + B) | Distributive Law |
| | = A·1 | Anything OR'ed w/1 is 1: A + 1 = 1 |
| | = A | Anything AND'ed w/1 is itself: A·1 = A |

Or, you can use a truth table.

| A | B | A·B | A + A·B |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

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




| A | B | C | $\overline{A}$ | $\overline{B}$ | $\overline{C}$ | $\overline{A} \cdot \overline{C}$ | $\overline{B} \cdot C$ | $\overline{A} \cdot \overline{C} + \overline{B} \cdot C$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

19. 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.

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

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

```
_________
(A + B)·C       First, distribute the inverse across the AND
                outside the parenthesis.  This places an
                inverse over (A + ^B) and an inverse over the
                ^C while changing the AND to an OR.  The
                double inverses over C will cancel.
_______         Distributing the inverse across the (A+^B)
(A + B) + C     term places an inverse over A and an inverse
                over ^B and changes the OR to an AND.  The
                double inverses over B cancel.
                This gives us the final answer.  Note the
(A·B) + C       importance of the parenthesis.
```

***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 25. 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)

This wasn't meant to be an evil question; it was actually meant to give you a couple of easy 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., $00000000_2 = 0$ and $11111111_2 = 25$.

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 25. That means that a single increment represents a different in the analog value of (25 – 0)/255. This immediately answers part b because $00000001_2$ is exactly one increment above 0 and hence represents 25/255.

d is the hard one. It represents $01010000_2 = 64 + 16 = 80_{10}$ increments above 0. Therefore, the analog value it corresponds to is $80 \times 25/255 = 2000/255$.

a.) $00000000_2 = 0$

b.) $00000001_2 = 25/255$

c.) $01010000_2 = 80 \times 25/255$

d.) $11111111_2 = 25$

22. 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)

There are a few ways to do each of these. I simply present one for each of them here.

a.) $A \cdot \overline{B} \cdot \overline{C} + \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot E + \overline{(B + C)} = A \cdot \overline{B} \cdot \overline{C}$ Answer: **False**

| | |
|---|---|
| $A \cdot \overline{B} \cdot \overline{C} + \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot E + \overline{B} \cdot \overline{C}$ | DeMorgan's Theorem |
| $\overline{B} \cdot \overline{C} \cdot (A + D + A \cdot E + 1)$ | Pull out ^B·^C |
| $\overline{B} \cdot \overline{C} \cdot (1)$ | Anything OR'ed w/1 equals 1 |
| $\overline{B} \cdot \overline{C}$ | Anything AND'ed w/1 equals itself |

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

| | |
|---|---|
| $A + B \cdot C \cdot \overline{C} \cdot D$ | (A + B)(A + C) = A + BC |
| $A + B \cdot 0 \cdot D$ | Anything AND'ed w/inverse equals 0 |
| $A + 0$ | Anything AND'ed w/0 equals 0 |
| $A$ | Anything OR'ed w/0 equals itself |

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

| | |
|---|---|
| $B \cdot C + \overline{(B \cdot C)}$ | Apply DeMorgan's Th. "backwards" |
| 1 | Anything OR'ed w/inverse equals 1 |

23. 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** | **10101101** | **11010011** | **X** | **X** |
| **84** | **01010100** | **01010100** | **01010100** | **10000100** |
| **–96** | **10100000** | **11100000** | **X** | **X** |

First row:

- First, note that signed magnitude binary (a representation which allows for negative numbers) uses the first bit as a sign bit. Since it is a 1 in this case, the value represented must be negative. Therefore, there will be no equivalent representation in unsigned binary or in unsigned BCD. Put X's in those columns.
- Decimal: First of all, the value is negative. To begin the conversion process, we must first convert the value to a positive number by clearing the sign bit to 0. This gives us the unsigned value $01010011_2$. To calculate the decimal value, add the powers of two represented by the ones in the unsigned value. This gives us $2^6 + 2^4 + 2^1 + 2^0 = 64 + 16 + 2 + 1 = 83_{10}$. Therefore, the decimal value is $-83_{10}$.

- To calculate the 2's complement representation, take the unsigned value 01010011 from the previous step, and compute the 2's complement using the shortcut we presented in class (red indicates inverted bits):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 83: | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| –83: | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

  This gives us 10101101.

Second row:

- Unsigned binary can only represent positive values, but since those values could go outside the range of the signed values, we need to check to make sure it isn't too large. This is done by examining the first bit. Since it is a zero, it can be represented in both unsigned binary and 2's complement. If the MSB was a 1, its magnitude as a positive value would be too great for either of the signed representations, 2's complement and signed magnitude.
  Since it is positive and not outside the range for either of the signed representations, the binary representation is the same across all three: 2's complement, signed magnitude, and unsigned binary.
- Decimal: We simply need to figure out which powers of two are represented by the unsigned value $01010100_2$ in order to calculate the decimal value. This gives us $2^6 + 2^4 + 2^2 = 64 + 16 + 4 = 84_{10}$.
- 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 $8_{10} = 1000$ and $4_{10} = 0100$. Therefore, the BCD column is set to 10000100.

Third row:

- First of all, the decimal value is negative, so neither of the "positives only" representations can represent this value. Put X's in both the unsigned binary and unsigned BCD columns.
- 2's Complement: To convert –96 to 2's complement, we begin by converting the positive value 96 to unsigned binary. Do this by breaking it into its powers of two components: 64 and 32.

$96 = 64 + 32 = 2^6 + 2^5$ →

| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

  Using the shortcut to convert 01100000 to 2's complement, we get (red indicates inverted bits):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 96: | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| –96: | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

- Signed Magnitude: Simply flip the MSB of the unsigned binary value 01100000 to get 11100000.