Name Solution

Digital Logic I

EE 2720-2

Midterm Examination 1

3 October 2011, 14:40–15:30 CDT

Exam Rules

Use only a pencil or pen. No calculators of any kind are allowed. Texting is out of the question.

Problem 1		(15 pts)
Problem 2		(9 pts)
Problem 3		(9 pts)
Problem 4		(24 pts)
Problem 5		(9 pts)
Problem 6		(9 pts)
Problem 7		(16 pts)
Problem 8		(9 pts)
Exam Total		(100 pts)

Alias Logic Won

Good Luck!

Problem 1: (15 pts) Perform the conversions indicated below. For conversions to decimal all you need to show is the arithmetic that needs to be done to compute the value, there's no need to perform the computation. For example, a decimal value can be written as $\boxed{12 + 3 \times \frac{4}{5+6} + 7 \times \pi^{-9}}$. This only applies to answers in decimal.

☑ Convert 257_{10} to hexadecimal.

Hint: $16^2 = 256$.

Hmmm ... the value of a 4-digit hexadecimal number consisting of digits $d_2d_1d_0$ is $d_2 \times 16^2 + d_1 \times 16 + d_0$. Now, we need to find digits such that $d_2 \times 256 + d_1 \times 16 + d_0 = 257_{10}$. That must mean $d_2 = 1$ and $d_1 = 0$ and $d_0 = 1$. So the number in hex is $\boxed{101_{16}}$.

☑ Convert $2f7_{16}$ to decimal.

The $\boxed{\text{value is } 2 \times 16^2 + 15 \times 16 + 7}$ which is sufficient to get full credit. But for completeness, that's 759_{10} in decimal.

☑ Convert $2.f7_{16}$ to decimal. (Don't overlook the radix point.)

The only difference here is the exponents on the 16's. The $\boxed{\text{value is } 2 \times 16^0 + 15 \times 16^{-1} + 7 \times 16^{-2}}$. For completeness, that's $\frac{759_{10}}{256_{10}} = 2.96484375$ in decimal.

☑ Convert 257_9 to decimal.

Here we use powers of 9. The $\boxed{\text{value is } 2 \times 9^2 + 5 \times 9 + 7}$. For completeness, that's 214_{10} in decimal.

☑ Convert $1011\,0100_2$ to decimal.

Here we use powers of 2. The $\boxed{\text{value is } 2^7 + 2^5 + 2^4 + 2^2}$, which is 180_{10}. Note that terms for the digits of value zero were omitted, and the $1\times$ was omitted from terms for the digits of value 1. Those with more experience might by eye convert it to $b4_{16}$ then compute $11 \times 16 + 4 = 180$.

Problem 2: (9 pts) Perform the conversions below, taking advantage of the fact that all radices are powers of 2.

☑ Convert abc_{16} to binary.

Because the radix is a power of 2, $16 = 2^4$, we know that each digit corresponds to, in this case, 4 bits. If the starting radix were 2^r each digit would correspond to r bits. So we convert each hex digit into 4 bits.

Maybe we remember that $a_{16} = 1010_2$, if not its easy enough to derive. Given that we can convert the other two hex digits: $b_{16} = 1011_2$ and $c_{16} = 1100_2$. Combining them gives the $\boxed{\text{binary number } 1010\,1011\,1100_2}$.

☑ Convert abc_{16} to octal.

We can convert from binary to a radix that is a power of 2, say 2^s, by arranging the digits into groups of s and then converting each group to a digit. For octal (radix 8) we have $s = 3$, so we need to group digits by 3's:

$$1010\,1011\,1100_2 = 101\,010\,111\,100_2 = 5274_8$$

So the answer is $\boxed{5274_8}$.

☑ Convert abc_{16} to radix 4.

This is solved the same way as the previous question, except that $s = 2$:

$$1010\,1011\,1100_2 = 10\,10\,10\,11\,11\,00_2 = 222330_4$$

So the answer is $\boxed{222330_4}$.

Problem 3: (9 pts) For each pair below briefly indicate an advantage (in some situations) of the first coding over the second coding.

☑ Advantage of Gray code over binary code.

In Gray code only a single bit changes between consecutive numbers. In contrast, with binary encoding multiple bits can change. For example, consider a four-bit quantity changing from a numeric value of seven to eight. For binary that would be `0111` $\rightarrow$ `1000`. For Gray code the corresponding change would be `0100` $\rightarrow$ `1100`. Consider again the binary case, `0111` $\rightarrow$ `1000`. Note that every digit position changes. No real device can change all four bits instantaneously, meaning the bits will change one at a time over some very brief interval. Suppose the order in which the digits changed was random. Then between the time the value is `0111` and the time the value is at its correct value of `1000` any other number could appear, albeit briefly. In contrast with Gray code only one bit changes and so no matter when the bits are examined either the initial value, `0100` (seven), or the final value, `1100` (eight), can appear.

☑ Advantage of BCD over binary.

See Brown & Vranesic Section 5.7.3. Conversion to decimal is easy.

☑ Advantage of binary over BCD.

Larger dynamic range for a given number of bits.

Problem 4: (24 pts) Perform the conversions indicated below. The number representations in this problem have specific sizes (*e.g.*, 8 bits).

☑ Convert the following 8-bit binary unsigned to decimal: `1100 0101`

Full-credit answer $\boxed{2^7 + 2^6 + 2^2 + 2^0}$. Note that 7, 6, 2, and 0 are the positions of the 1's in `1100 0101`. One of the problems below requires the decimal value of this number, the calculation is $2^7+2^6+2^2+2^0 = 128+64+4+1 = 197$. *Tip: Knowing the powers of 2 up to 2^{16}, and approximate values for 2^{20} and 2^{30} will make a good impression on job interviews.*

☑ Convert the following 8-bit binary signed magnitude to decimal: `1100 0101`

In signed magnitude the leftmost digit is a sign (1 means minus, 0 means plus). The remaining digits are the magnitude represented as an unsigned number. Here the leftmost bit is 1, meaning the result is negative. The remaining bits are `100 0101` and their value is $2^6+2^2+2^0$. So the value of the whole thing is $\boxed{-\left(2^6 + 2^2 + 2^0\right)}$. For the record, that's -69. *Grading Note: On many exams the parentheses were omitted in this and subsequent problems.*

☑ Convert the following 8-bit binary 2's complement to decimal: `1100 0101`

There are several methods to solve this. In all methods the first step is checking whether the number is positive, and if so the number is converted to decimal in the usual way. A number is positive if the leftmost bit is 0, but in this case the leftmost bit is 1 so its negative.

One method to convert a negative 2's complement number to decimal is to first invert each bit, then add 1, then convert it as an unsigned number. Using this procedure: $\texttt{1100 0101} \xrightarrow{\text{Invert}} \texttt{0011 1010} \xrightarrow{\text{Add 1}} \texttt{0011 1011}$. Next convert $11\,1011_2$ to decimal and negate the result. That yields $\boxed{-\left(2^5 + 2^4 + 2^3 + 2^1 + 2^0\right)}$. For the record, that's -59.

☑ Convert the following 8-bit binary 1's complement to decimal: `1100 0101`

A quick way to solve this is to remember that the 1's complement is just one smaller in magnitude than 2's complement (because a 1 isn't added). So take the answer from the last part and add 1: $-\left(2^5 + 2^4 + 2^3 + 2^1 + 2^0\right) + 1$ which is $\boxed{-\left(2^5 + 2^4 + 2^3 + 2^1\right)}$.

Or one can start from the beginning: Follow the same procedure as for 2's complement, except don't add one: $\texttt{1100 0101} \xrightarrow{\text{Invert}} \texttt{0011 1010}$. Next convert $11\,1010_2$ to decimal and negate the result. That yields $\boxed{-\left(2^5 + 2^4 + 2^3 + 2^1\right)}$ or -58.

☑ Convert the following 8-bit binary 2's complement to 12-bit binary 2's complement: `1100 0101`

Notice that the answer is in binary too. To convert to 12-bit binary 2's complement all we need to do is prepend four more bits (put four bits on the left-hand side). If the original number is positive those four bits should be 0's, otherwise if the number is negative (which it is in this case) they should be 1's. So the answer is $\boxed{\texttt{1111 1100 0101}}$.

☑ Convert the following 12-bit (yes 12 bit even though 8 bits are shown) binary 2's complement to decimal: 1100 0101

If this were an 8-bit 2's complement number it would be negative and the value would be -59 (see one of the earlier parts). But it's 12 bits and so the leftmost bit is 0. Notice that leftmost bit refers to the full 12-bit representation, 0000 1100 0101. Since the number is positive convert it as unsigned. The answer will be the same as the first part, $\boxed{2^7 + 2^6 + 2^2 + 2^0}$ or 197.

☑ Convert the following 8-bit binary unsigned to 12-bit BCD: 1100 0101

The 8-bit number is the same as the first part, its value is 197_{10}. The 12-bit BCD representation consists of 3 digits, 1, 9, and 7: they are $\boxed{\texttt{0001 1001 0111}}$.

☑ Convert the following 8-bit excess-3 code to decimal: 1100 0101

Excess 3 is similar to BCD except that the four bits representing a digit are set to the digit value plus 3. The most significant digit is 1100. The value would be that minus 3, or $12 - 3 = 9$. Similarly the least significant digit is 0101 so the digit value would be $5 - 3 = 2$. The decimal value is then $\boxed{92_{10}}$.

☑ Double-check that minus signs were included where necessary in the answers above.

Problem 5: (9 pts) Find the minimum number of bits to encode one thousand in each of the representations below. For example, the number of bits needed for 10^3 in unsigned binary would be 10 because $1000_{10} = 3e8_{16}$ and each hex digit spans four bits. *Hint: The solution does not require a tedious conversion.*

☑ Number of bits needed to encode one thousand in BCD:

BCD encodes decimal numbers, one thousand needs four digits to represent in decimal. Each BCD digit is 4 bits so the total number of bits is $\boxed{4 \times 4 = 16}$.

☑ Number of bits needed to encode one thousand in decimal using ASCII:

The encoding is supposed to be decimal using ASCII characters. In decimal, 1000 requires 4 digits. In ASCII each character is 8 bits (including the parity bit). So the total number of bits is $\boxed{4 \times 8 = 32}$.

☑ Number of bits needed to encode one thousand in English using ASCII:

The encoding is supposed to be English using ASCII characters. In English "One Thousand" takes 12 characters, each needing 8 bits. So the total number of bits is $\boxed{12 \times 8 = 96}$.

Grading Note: For the first two parts, BCD, and decimal/ASCII, the number of bits specified can represent not just 1000 but any number between 0 and 1000. That's not the case here because twelve characters, for example, cannot represent "nine hundred ninety nine." This did not seem to bother anyone.

Problem 6: (9 pts) Solve the following arithmetic problems. Show the result (sum) in the same representation as the operands. To avoid confusion, box your answer and show your work.

☑ Add the following 8-bit unsigned numbers.

In decimal the sum is $197+83 = 280$, or in binary the arithmetic sum is 1 0001 1000. (The problem should be solved by doing the addition in binary.) The sum is supposed to be put in an 8-bit unsigned representation. So the solution is $\boxed{\texttt{0001 1000}}$. Note that this is not the correct arithmetic sum due to overflow.

☑ Indicate whether there is overflow:

Yes. Any carry out in unsigned addition indicates overflow (when all operands are the same size).

$$\begin{array}{r} 1100\,0101 \\ +\ \underline{0101\,0011} \end{array}$$

☑ Add the following 8-bit 2's complement numbers.

Follow exactly the same procedure as used for unsigned numbers. The result is $\boxed{\texttt{0001 1000}}$, For the record, the arithmetic is $-59 + 83 = 24$, and $24_{10} = 1\,1000_2$.

☑ Indicate whether there is overflow:

No, because there can never be overflow when one operand is positive and the other is negative.

$$\begin{array}{r} 1100\,0101 \\ +\ \underline{0101\,0011} \end{array}$$

☑ Add the following 8-bit 1's complement numbers.

Follow the same procedure as for unsigned numbers except if there is carry out add it to the initial sum to get the final sum. In this case there is a carry out so the sum will be $\boxed{\texttt{0001 1001}}$. For the record, the arithmetic is $-58+83 = 25$, and $25_{10} = 1\,1001_2$.

☑ Indicate whether there is overflow:

No, because there can never be overflow when one operand is positive and the other is negative.

$$\begin{array}{r} 1100\,0101 \\ +\ \underline{0101\,0011} \end{array}$$

Problem 7: (16 pts) Transform the following Boolean expressions as indicated.

☑ Simplify.

$$x \cdot y + x \cdot y \cdot z' + y \cdot w \cdot x$$

Apply the cover theorem: $a + a \cdot b = a$, with $a \rightarrow xy$. This eliminates the last two terms leaving just $\boxed{x \cdot y}$.

☑ Simplify by eliminating one term.

$$x \cdot (a + b) + (a + b)' \cdot (c + e) + (c + e) \cdot x$$

Apply the consensus theorem: $\alpha \cdot w + \alpha' \cdot y + w \cdot y = \alpha \cdot w + \alpha' \cdot y$. In this case $\alpha \rightarrow (a + b)$, and $w \rightarrow x$, and $y \rightarrow (c + e)$. The result is $\boxed{x \cdot (a + b) + (a + b)' \cdot (c + e)}$.

☑ Simplify.

Those who recognize something from Homework 2 can solve this quickly.

$$(a + b + c) \cdot (a + b + e) \cdot (a + b + f)$$

In Homework 2 we proved that $x + y_1 \cdot y_2 \cdot \cdots \cdot y_n = (x + y_1) \cdot (x + y_2) \cdot \cdots \cdot (x + y_n)$. Here we set $x \rightarrow a + b$, $y_1 \rightarrow c$, $y_2 \rightarrow e$, and $y_3 \rightarrow f$. That yields $\boxed{a + b + c \cdot d \cdot e}$.

☑ Write the following as a sum of products.

$$[a + b + (c + e)']' + f$$

First Demorganize

$$[a' \cdot b' \cdot (c + e)] + f$$

remove the now superfluous brackets

$$a' \cdot b' \cdot (c + e) + f$$

finally multiply out to get the answer:

$$a' \cdot b' \cdot c + a' \cdot b' \cdot e + f$$

Grading note: many people made errors applying Demorgan's theorem. Be careful.

☑ Write the following as a product of sums.

$$a + b \cdot (c + e)$$

Apply the theorem $x + y \cdot z = (x + y) \cdot (x + z)$, here set $x \rightarrow a$, $y \rightarrow b$, and $z \rightarrow (c + e)$. That yields $\boxed{(a + b) \cdot (a + c + e)}$.

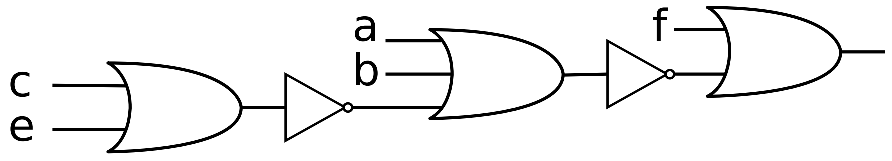
Problem 8: (9 pts) Convert the Boolean expressions as indicated.

(*a*) Show logic gates corresponding to the expression below as written. (That is, don't simplify or otherwise transform it.)

$$[a + b + (c + e)']' + f$$

☑ Show logic gates.

Diagram appears below. *Grading Note: On many exams the expression was first simplified. The warning about not simplifying did not appear on the original exam.*



(*b*) Show the dual of the Boolean equation below:

$$x + x' \cdot y = x + y$$

☑ Dual of equation above.

An expression is transformed into its dual by changing AND to OR, OR to AND, and the constants 1 to 0 and 0 to 1 (the expression above does not have constants). One should take care that the order of operations **is not changed**. A safe way to do that is to insert parentheses around every operator/operand set. The parenthesized version of the expression above is

$$(x + (x' \cdot y)) = (x + y)$$

Now swap the operators

$$(x \cdot (x' + y)) = (x \cdot y)$$

and then remove unnecessary parenthesis to yield the cleaned-up solution

$$x \cdot (x' + y) = x \cdot y$$

The following is the **wrong** answer because the order of operations has changed: $x \cdot x' + y = x \cdot y$. Examining this for a moment one can see that it can't be right because since $x \cdot x' = 0$ the equality would reduce to $y = x \cdot y$. *Grading Note: This was a common mistake.*

(c) Prove that the following equality is valid by constructing a truth table.

$$x + x' \cdot y = x + y$$

[✓] Truth table to prove equality.

`Solution:`

Inputs		Left Hand Side			Right Hand Side
x	y	x	x'y	x+x'y	x+y
0	0	0	0	0	0
0	1	0	1	1	1
1	0	1	0	1	1
1	1	1	0	1	1