ECE 120: Introduction to Computing Spring 2019 ZJUI

due at start of section on Friday 8 March 2019

Homework 2: Operations on Bits

1. **Boolean Functions**

   Write a truth table for the following Boolean functions (all in one table)

   a. f1 = A AND [B OR (NOT C)] AND (NOT D)
   b. f2 = (NOT A) OR (NOT B) OR [(NOT C) AND D]
   c. f3 = NOT {(NOT A) OR [(NOT B) AND C] OR D}
   d. f4 = A AND B AND [C OR (NOT D)]
   e. Now compare the functions. From the truth table, what can you say about the four functions?

A	B	C	D	f1	f2	f3	f4
0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	1	1	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	1	0	1	1
1	1	0	1	0	1	0	0
1	1	1	0	1	0	1	1
1	1	1	1	0	0	0	1

From the truth table, we see that f1 = f3, while f2 = (NOT f4)

2. **Bitwise Logic**

   Calculate the resulting bit patterns for the following operations. The operand bit patterns are given in hexadecimal notation (preceded by an "x," following Patt & Patel's notation). Write your answers using binary notation, NOT HEXADECIMAL.

   a. xECEB AND xCAFE

   xECEB = 1110 1100 1110 1011

   xCAFE = 1100 1010 1111 1110

   xECEB AND xCAFE = 1100 1000 1110 1010 = xC8EA

b. NOT (x1234 XOR xFEDC)

x1234 = 0001 0010 0011 0100

xFEDC = 1111 1110 1101 1100

(x1234 XOR xFEDC) = 1110 1100 1110 1000

NOT (x1234 XOR xFEDC) = 0001 0011 0001 0111 = x1317

c. x9347 OR xCB03

x9347 = 1001 0011 0100 0111

xCB03 = 1100 1011 0000 0011

x9347 OR xCB03 = 1101 1011 0100 0111 = xDB43

3. Logical Completeness

a. Prove that the 2-input NOR function (OR followed by NOT) is logically complete.

From class, we know that the set {2-input AND, 2-input OR, NOT} is logically complete, so we need merely show how to construct these functions using 2-input NOR.

Given an input A, NOT A = (A NOR A)

Given inputs A and B, and NOT as shown above,

A OR B = NOT (A NOR B)

And then DeMorgan's law gives us AND:

A AND B = NOT ((NOT A) OR (NOT B))

b. Write the function F = (A XOR B) AND (C XOR D) using only AND, OR, and NOT.

Write a truth table for the function first.

A	B	C	D	A XOR B	C XOR D	F
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	0	0	0

Then pick all the cases where F = 1: A'BC'D, A'BCD', AB'C'D, and AB'CD'

and OR them together: F = A'BC'D + A'BCD' + AB'C'D + AB'CD'.

4. **Precision of Floating-Point**

Consider the IEEE 754 single-precision floating-point representation discussed in the book and in lecture.

a. Can every fraction F ($0 \le F \le 1$) representable using only two digits after the decimal point (such as 0.42) be represented exactly in floating-point? Explain your answer.

No. Take 0.80 as an example (almost any number will do) and call the bits F_i.

$0.8 \times 2 = 1.6$, so $F_{-1} = 1$. Subtracting 1, we have 0.6.
$0.6 \times 2 = 1.2$, so $F_{-2} = 1$. Subtracting 1, we have 0.2.
$0.2 \times 2 = 0.4$, so $F_{-3} = 0$.
$0.4 \times 2 = 0.8$, so $F_{-4} = 0$.

But now we have the original 0.8 back, so the process will go on forever. An infinite number of bits cannot fit into a 23-bit mantissa, so we must approximate.

Why? In decimal, a finite number of digits can represent only those rational numbers that can be expressed with denominators that are powers of 10 (any product of powers of 2 and 5 can be multiplied appropriately to form a power of 10). In binary, the denominator must be expressible as a power of 2. Most of the numbers in question do not satisfy this requirement, and are thus infinite in binary, and not exactly representable in floating-point.

b. Can the number 10^8 be represented exactly in floating-point? Explain your answer.

Yes. Here 10^8 can be factored into $2^8 5^8$. The 2^8 becomes part of the exponent, and the 5^8 fits into the 24 bits available with the implicit leading 1 and the 23-bit mantissa. In particular, 10^8 can be represented as 0 10011001 01111101011110000100000.

c. Can the number 10^{11} be represented exactly in floating-point? Explain your answer.

No. Here again we can factor into $2^{11}5^{11}$, but $5^{11} = 48828125 = x2E90EDD$ requires more than 24 bits, so it cannot fit exactly into the mantissa—the low bits must be dropped, and the number represented is not exactly 10^{11}.

5. Conversion to Floating-Point

For each of the following decimal numbers, convert the number to base 2 notation, using only as many bits as are necessary to precisely specify the number (using no leading nor trailing zeroes), then convert your answer into the IEEE 754 single-precision floating-point representation. Show your work.

a. -101.3839111328125

First convert to base 2 to obtain
-1100101.0110001001001
Next, rewrite as binary scientific notation:
$-1.1001010110001001001 \times 2^{6}$
Then convert to IEEE 754 single-precision floating-point representation
1 10000101 10010101100010010010000

b. 23.8046875

First convert to base 2 to obtain
10111.1100111
Next, rewrite as binary scientific notation:
$1.01111100111 \times 2^{4}$
Then convert to IEEE 754 single-precision floating-point representation
0 10000011 01111100111000000000000

6. Conversion from Floating-Point

For each of the following bit patterns, calculate the number represented by the bits when interpreted using the IEEE 754 single-precision floating-point representation. Write your answer in decimal (scientific notation is fine). Show your work.

a. 1 01011000 10111010000000000000000

$1.1011101 = 221/128 = 1.7265625$, and the exponent gives 2^{-39}.
So we have $-1.7265625/ 2^{39} = 3.14059889 \times 10^{-12}$.

b. 0 10010011 10010001000000000000000

$1.10010001 \times 2^{20} = 1\ 1001\ 0001\ 0000\ 0000\ 0000 = 1642496$.

7. Pizza Time!

Almost every college student in the United States loves pizza, but not everyone can agree on what toppings to include. Help four of your friends to determine what they should order to avoid starvation while completing their ECE120 homework…

Available toppings include:
B – bell pepper (a vegetable)
H – ham (a meat)
M – mushroom (a vegetable)
P – pepperoni (a meat)
S – sausage (a meat)

Start by translating each person's needs into a Boolean logic expression based on the variables B, H, M, P, and S. Cheese is considered a vegetarian food for the purpose of this problem (so a pizza with no toppings is vegetarian, for example).

a. Jan hates sausage, but won't eat vegetarian pizza.
   S' (H + P)
b. Alice wants at least one vegetable and at least one meat.
   (B + M) (H + P + S)
c. Bob hates ham.
   H'
d. Xin doesn't want more than one topping with "pepper" in its name.
   B' + P'
e. Finally, determine what toppings to include on the pizza!
   The idea is to find the combination of (B, H, M, P, S) such that all four expressions above are true. In other words, AND them together:
   S' (H + P) (B + M) (H + P + S) H' (B' + P')

   Since S' and H' must be true, S and H must both be false, so we can simplify both (H + P) and (H + P + S) to P, then remove one copy of P. Now we have…
   S' P (B + M) H' (B' + P')

   Since P must be true, P' must be false, so we can simplify (B' + P') to B':
   S' P (B + M) H' B'

   And, finally, since B' must be true, B must be false, so we can simplify (B + M) to M to find the single solution:
   S' P M H' B'

   The toppings: pepperoni and mushroom.

   You can also use a truth table as we did in class for my movie club, of course.