CS/COE 0447 Fall 2009
Homework 3
Solution

1. Convert the following decimal numbers into 8-bit binary numbers in biased notation with a bias of 127.

121 is represented by the bit pattern of 121 + 127 = 248
248 / 2 = 124 + **0** / 2
124 / 2 = 62 + **0** / 2
62 / 2 = 31 + **0** / 2
31 / 2 = 15 + **1** / 2
15 / 2 = 7 + **1** / 2
7 / 2 = 3 + **1** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
248 = 11111000b

-25 is represented by the bit pattern of -25 + 127 = 102
102 / 2 = 51 + **0** / 2
51 / 2 = 25 + **1** / 2
25 / 2 = 12 + **1** / 2
12 / 2 = 6 + **0** / 2
6 / 2 = 3 + **0** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
102 = 01100110b

-71 is represented by the bit pattern of -71 + 127 = 56
56 / 2 = 28 + **0** / 2
28 / 2 = 14 + **0** / 2
14 / 2 = 7 + **0** / 2
7 / 2 = 3 + **1** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
56 = 00111000b

2. Convert the following decimal numbers to binary numbers.

Whole part of 5.75 is 5:
5 / 2 = 2 + **1** / 2
2 / 2 = 1 + **0** / 2
1 / 2 = 0 + **1** / 2

5 = 101b
Decimal fraction of 5.75 is 0.75:
0.75 x 2 = **1**.5
0.5 x 2 = **1**.0 (done because decimal fraction is 0)
Binary fraction is 0.11b
5.75 = 101.11b

Whole part of 45.375 is 45:
45 / 2 = 22 + **1** / 2
22 / 2 = 11 + **0** / 2
11 / 2 = 5 + **1** / 2
5 / 2 = 2 + **1** / 2
2 / 2 = 1 + **0** / 2
1 / 2 = 0 + **1** / 2
45 = 101101b
Decimal fraction of 45.375 is 0.375:
0.375 x 2 = **0**.75
0.75 x 2 = **1**.5
0.5 x 2 = **1**.0 (done because decimal fraction is 0)
Binary fraction is 0.011b
45.375 = 101101.011b

Whole part of 13.40625 is 13:
13 / 2 = 6 + **1** / 2
6 / 2 = 3 + **0** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
13 = 1101b
Decimal fraction of 13.40625 is 0.40625:
0.40625 x 2 = **0**.8125
0.8125 x 2 = **1**.625
0.625 x 2 = **1**.25
0.25 x 2 = **0**.5
0.5 x 2 = **1**.0 (done because decimal fraction is 0)
Binary fraction is 0.01101b
13.40625 = 1101.01101b

3. Convert the following binary numbers to decimal numbers:

110010110.11001b:
2^-5 + 2^-2 + 2^-1 + 2^1 + 2^2 + 2^4 + 2^7 + 2^8 = 0.03125 + 0.25 + 0.5 + 2 + 4 + 16 + 128 + 256 = 406.78125

1001110.0000101b:
2^-7 + 2^-5 + 2^1 + 2^2 + 2^3 + 2^6 = 0.0078125 + 0.03125 + 2 + 4 + 8 + 64 = 78.0390625

100111011.11001101b:
2^-8 + 2^-6 + 2^-5 + 2^-2 + 2^-1 + 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^8 = 0.00390625 + 0.015625 + 0.03125 + 0.25 + 0.5 + 1 + 2 + 8 + 16 + 32 + 256 = 315.80078125

4. Write down the binary representation of the following decimal numbers, assuming the IEEE 754 single precision format:

-1609.5:
Whole part of 1609.5 is 1609:
1609/ 2 = 804 + **1** / 2
804 / 2 = 402 + **0** / 2
402 / 2 = 201 + **0** / 2
201 / 2 = 100 + **1** / 2
100 / 2 = 50 + **0** / 2
50 / 2 = 25 + **0** / 2
25 / 2 = 12 + **1** / 2
12 / 2 = 6 + **0** / 2
6 / 2 = 3 + **0** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
1609 = 11001001001b
Decimal fraction of 1609.5 is 0.5:
0.5 x 2 = **1**.0 (done because decimal fraction is 0)
Binary fraction is 0.1b
-1609.5 = -11001001001.1b
In scientific notation: -11001001001.1 x 2^0
Normalizing: -1.10010010011 x 2^10
10 is bias notation is 10 + 127 = 137 = 10001001b
-1609.5 in IEEE 754 single precision format is: 1100 0100 1100 1001 0011 0000 0000 0000b

-938.8125
Whole part of 938.8125 is 938:
938/ 2 = 469 + **0** / 2
469 / 2 = 234 + **1** / 2
234 / 2 = 117 + **0** / 2
117 / 2 = 58 + **1** / 2
58 / 2 = 29 + **0** / 2
29 / 2 = 14 + **1** / 2
14 / 2 = 6 + **0** / 2
7 / 2 = 3 + **1** / 2
3 / 2 = 1 + **1** / 2
1 / 2 = 0 + **1** / 2
938 = 1110101010b
Decimal fraction of 938.8125 is 0.8125:
0.8125 x 2 = **1**.625
0.625 x 2 = **1**.25
0.25 x 2 = **0**.5
0.5 x 2 = **1**.0 (done because decimal fraction is 0)

Binary fraction is 0.1101b
- 938.8125 = -1110101010.1101b
In scientific notation: -1110101010.1101 x 2^0
Normalizing: - 1.1101010101101 x 2^9
9 is bias notation is 9 + 127 = 136 = 10001000b
-938.8125 in IEEE 754 single precision format is: 1100 0100 0110 1010 1011 0100 0000 0000b

130.59375
Whole part of 130.59375 is 130:
130/ 2 = 65 + **0** / 2
65 / 2 = 32 + **1** / 2
32 / 2 = 16 + **0** / 2
16 / 2 = 8 + **0** / 2
8 / 2 = 4 + **0** / 2
4 / 2 = 2 + **0** / 2
2 / 2 = 1 + **0** / 2
1 / 2 = 0 + **1** / 2
130 = 10000010b
Decimal fraction of 130.59375 is 0.59375 :
0.59375 x 2 = **1**.1875
0.1875 x 2 = **0**.375
0.375 x 2 = **0**.75
0.75 x 2 = **1**.5
0.5 x 2 = **1**.0 (done because decimal fraction is 0)
Binary fraction is 0.10011b
130.59375 = 10000010.10011b
In scientific notation: 10000010.10011 x 2^0
Normalizing: 1.000001010011 x 2^7
7 is bias notation is 7 + 127 = 134 = 10000110b
130.59375 in IEEE 754 single precision format is: 0100 0011 0000 0010 1001 1000 0000 0000b

5. Show the steps for the division of 1101b and 0011b (unsigned) using Hardware Design 3 (available here: http://www.cs.pitt.edu/~childers/CS0447/lectures/division-floats.pdf). Draw a table similar to the following one and fill up the columns:

Iteration	Divisor	Step	Remainder (8 bits)
0	0011	Initial Values	0000 1101
		Shift remainder left by 1	0001 1010
1	0011	remainder = remainder - divisor	1110 1010
		(remainder < 0) => remainder +=divisor; shift left; r0 = 0	0011 0100
2	0011	remainder = remainder - divisor	0000 0100
		(remainder >= 0) => shift left; r0 = 1	0000 1001

3	0011	remainder = remainder - divisor	1101 1001
		(remainder < 0) => remainder +=divisor; shift left; r0 = 0	0001 0010
4	0011	remainder = remainder - divisor	1110 0010
		(remainder < 0) => remainder +=divisor; shift left; r0 = 0	0010 0100
done	0011	shift left half of remainder right by 1	0001 0100

6. Show the steps for the division of 1101b and 0011b (unsigned) using Hardware Design 3 and non-restoring division (available here: http://www.cs.pitt.edu/~childers/CS0447/lectures/division-floats.pdf). Draw a table similar to the following one and fill up the columns:

Iteration	Divisor	Step	Remainder (8 bits)
0	0011	Initial Values	0000 1101
		Shift remainder left by 1	0001 1010
1	0011	remainder = remainder - divisor	1110 1010
		(remainder < 0) => shift left; r0 = 0	1101 0100
2	0011	remainder = remainder + divisor	0000 0100
		(remainder >= 0) => shift left; r0 = 1	0000 1001
3	0011	remainder = remainder - divisor	1101 1001
		(remainder < 0) => shift left; r0 = 0	1011 0010
4	0011	remainder = remainder + divisor	1110 0010
		(remainder < 0) => shift left; r0 = 0	1100 0100
done	0011	shift left half of remainder right by 1	1110 0100
		(remainder < 0) => remainder = remainder + divisor	0001 0100

7. Write down the function represented by the following Karnaugh map as a sum of products. Make sure you minimize the number of products.

	AB = 00	**AB = 01**	**AB = 11**	**AB = 10**
CD = 00	1	0	0	1
CD = 01	0	0	1	1
CD = 11	1	1	1	1
CD = 10	1	0	0	1

F = CD + B'D' + AD

8. Write down the function represented by the following Karnaugh map as a sum of products. Make sure

you minimize the number of products.

	AB = 00	**AB = 01**	**AB = 11**	**AB = 10**
CD = 00	0	1	1	0
CD = 01	1	1	0	1
CD = 11	0	0	0	0
CD = 10	0	1	1	0

F = BD' + A'C'D + B'C'D
or
F = BD' + A'BC' + B'C'D

9. Consider two 2-bit inputs X and Y, each consisting of individual bits x1 and x0, and y1 and y0, respectively. Write down the truth table for each of the following relations:

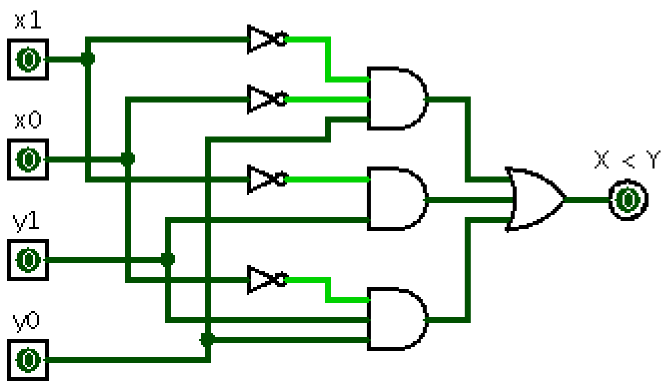
X < Y, where X and Y are unsigned binary numbers
X = Y

Write down each function as a sum of products and draw the circuit using AND/OR/NOT gates. Use Karnaugh maps to minimize the number of products.

x1	**x0**	**y1**	**y0**	**X**	**Y**	**X < Y**	**X = Y**
0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	0
0	0	1	0	0	2	1	0
0	0	1	1	0	3	1	0
0	1	0	0	1	0	0	0
0	1	0	1	1	1	0	1
0	1	1	0	1	2	1	0
0	1	1	1	1	3	1	0
1	0	0	0	2	0	0	0
1	0	0	1	2	1	0	0
1	0	1	0	2	2	0	1
1	0	1	1	2	3	1	0
1	1	0	0	3	0	0	0
1	1	0	1	3	1	0	0
1	1	1	0	3	2	0	0
1	1	1	1	3	3	0	1

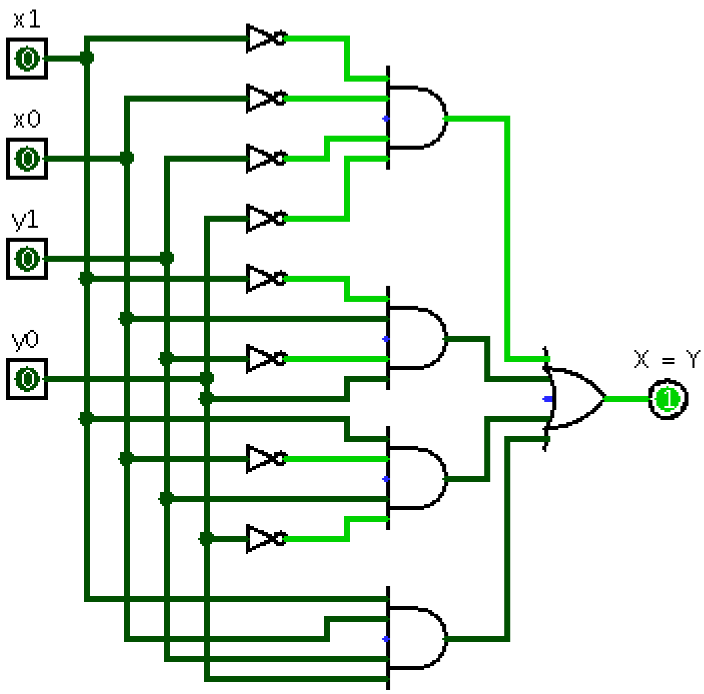
X < Y	**x1x0 = 00**	**x1x0 = 01**	**x1x0 = 11**	**x1x0 = 10**
y1y0 = 00	0	0	0	0
y1y0 = 01	1	0	0	0
y1y0 = 11	1	1	0	1
y1y0 = 10	1	1	0	0

F = x1'y1 + x1'x0'y0 + x0'y1y0

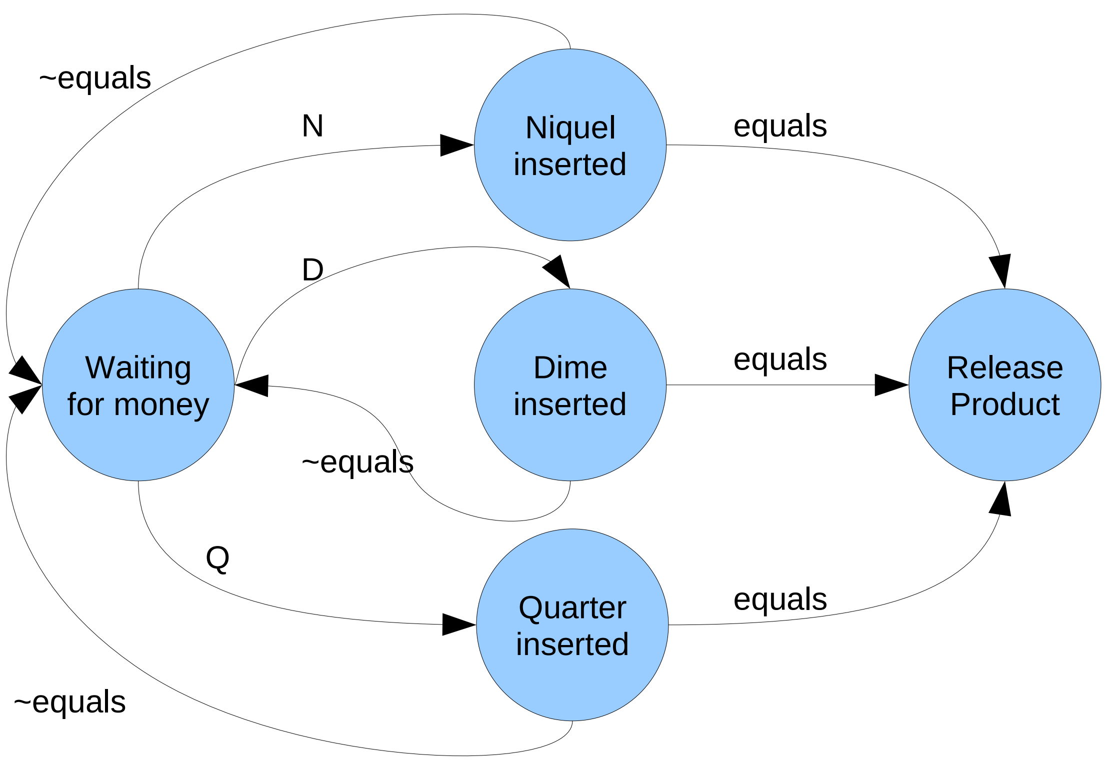


X = Y	**x1x0 = 00**	**x1x0 = 01**	**x1x0 = 11**	**x1x0 = 10**
y1y0 = 00	1	0	0	0
y1y0 = 01	0	1	0	0
y1y0 = 11	0	0	1	0
y1y0 = 10	0	0	0	1

F = x1'x0'y1'y0' + x1'x0y1'y0 + x1x0y1y0 + x1x0'y1y0'



10. There are several solution to this problem. Some combine several operations into just one state, while others split those operation into several states. A possible solution is the following:



The outputs are the following:

	mux	write	clear	done
Waiting for money	0	0	0	0
Niquel inserted	0	1	0	0
Dime inserted	1	1	0	0
Quarter inserted	2	1	0	0
Release product	0	0	1	1