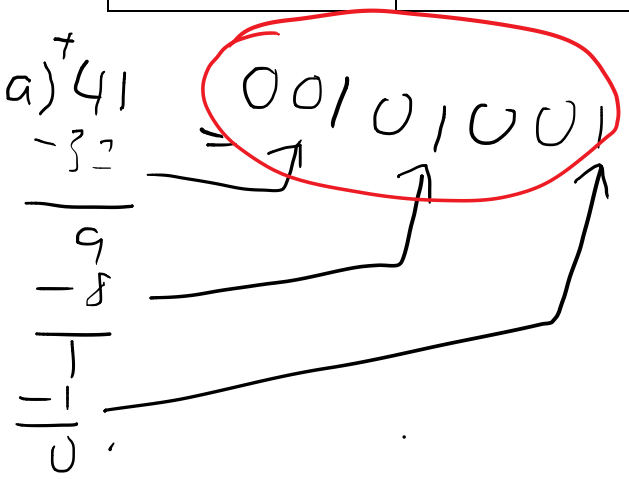
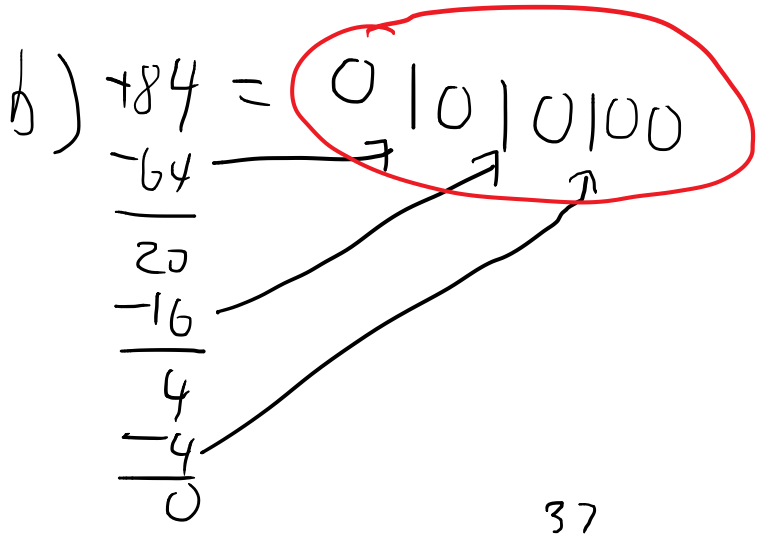
ECE 2504: Introduction to Computer Engineering
Homework Assignment 5 (60 points)

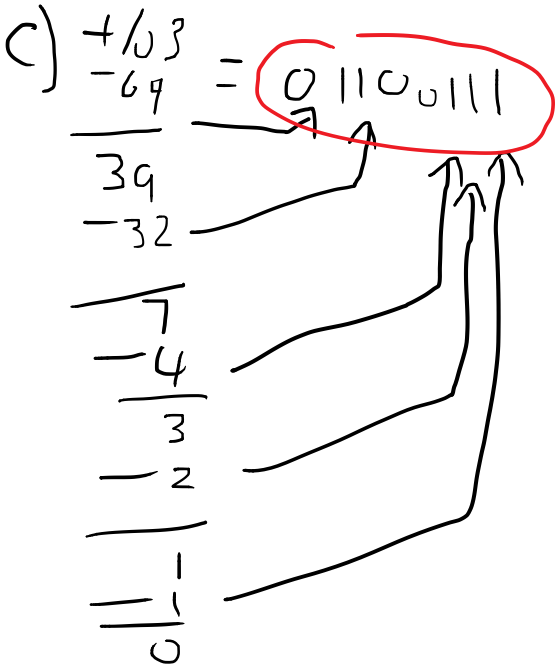
Make a reasonable effort to show your work. Clearly indicate your answers.

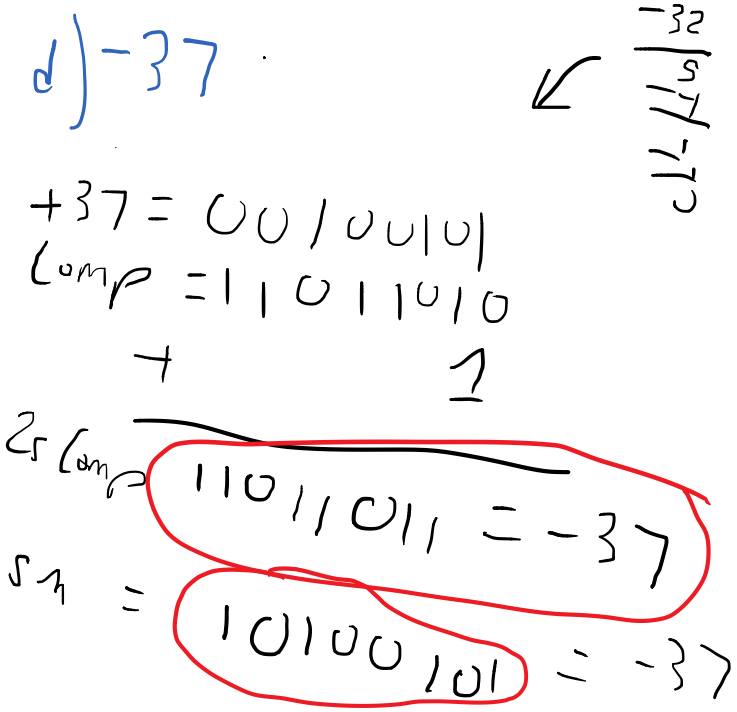
Problem 1 (12 points)
Express each of the following signed decimal numbers using **8-bit signed magnitude representation** and **8-bit 2's complement representation**.

	Signed-magnitude	2's complement
a. +41	00101001	00101001
b. +84	01010100	01010100
c. +103	01100111	01100111
d. -37	10100101	11011011
e. -88	11011000	10101000
f. -106	11101010	10010110









e) -88

+88 = 01011000

-64
24
-16
8
-8
0

So SM = 11011000

2s comp = 01011000
10100111
+ 1
10101000

f) -106

+106 = 01101010

-64
42
-32
10
-8
2
-2
0

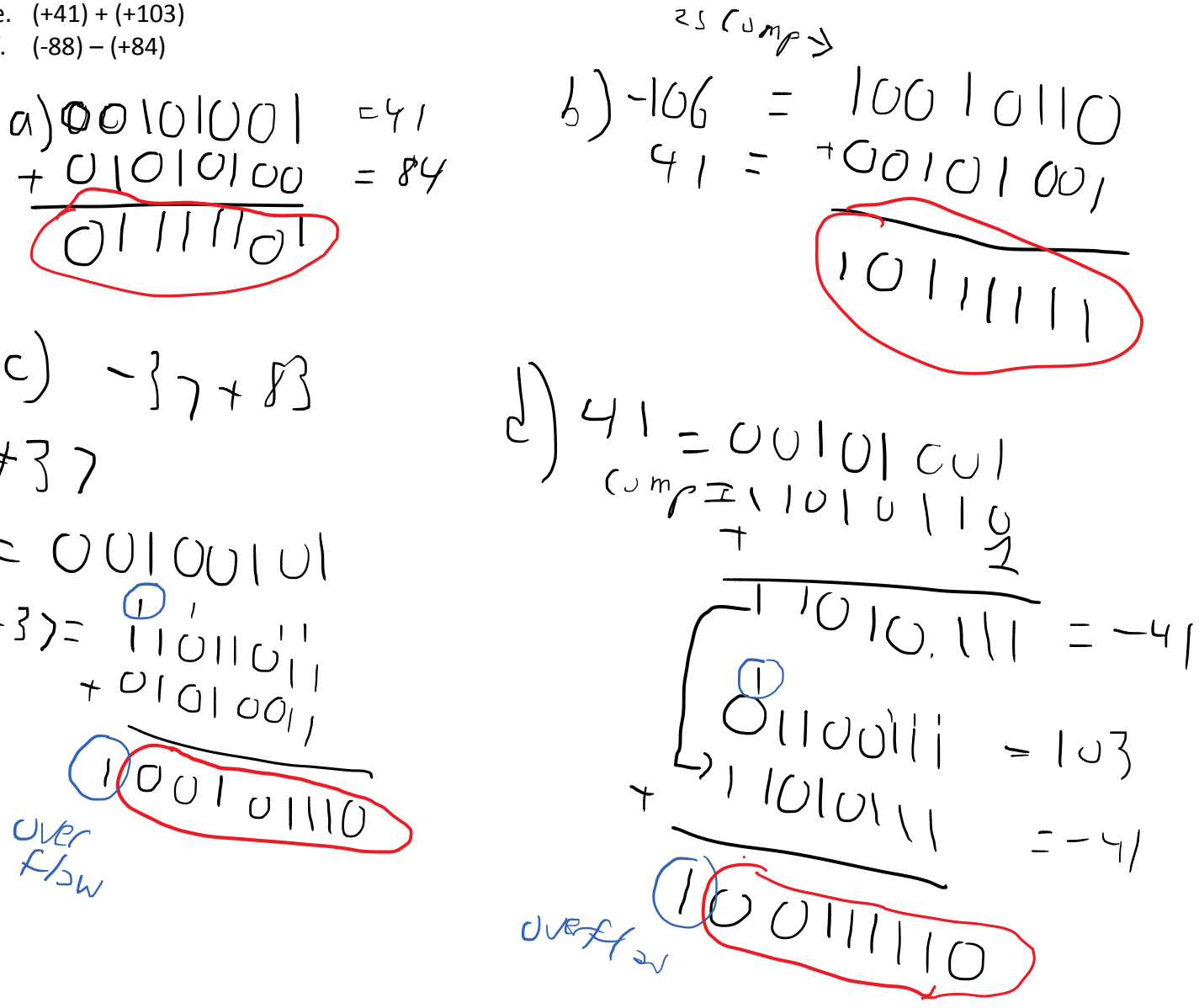
So SM = 11101010

2s comp = 01101010
10010101
+ 1
10010110

Problem 2 (18 points)
Perform the following arithmetic operations in **binary** using **8-bit 2's complement representation**.

- Perform subtraction by finding the 2's complement of the subtrahend and then performing addition.
- Give an appropriate indicator of overflow, if it occurs. *Make sure that you identify the specific sign in your arithmetic that indicates overflow.*
- I did not choose the numbers in this problem by accident; where appropriate, you need not repeat your work to represent the values you use in solving this problem.

a. (+41) + (+84)
b. (-106) + (+41)
c. (-37) + (+83)
d. (+103) – (+41)
e. (+41) + (+103)
f. (-88) – (+84)



e) 41 + 13

01010001 = 41
+ 00001101 = 13

01011110

F) +88

01011000
comp= 10100111
+ 1
10101000

+84

01010100
comp= 10101011
+ 1
10101100

1 10101000
+ 10101100
1 01010100

overflow

Problem 3 (8 points)

Represent the number $(+26.6875)_{10}$ as a 16-bit floating point number. This particular floating point format, uses signed-magnitude to store numbers in the mantissa (a normalized fraction) and the exponent (an integer). The mantissa has 10 bits (including the mantissa's sign bit) and the exponent has 6 bits (including the exponent's sign bit).

$26 - 16 = 10$, $10 - 8 = 2$, $2 - 2 = 0$

$= 11010.1011 = 26.6875$

$.6875 \times 2 = 1$

$.375 \times 2 = 0$

$.75 \times 2 = 1$

$.5 \times 2 = 1$

11010.1011

$= .110101011 \times 2^5$

S	mantissa	S	exponent
0	110101011	0	00101

10 bits | 6 bits

$= 0110101011000101$ for FP

Problem 4 (2 points)
One day, I overheard Michael Mistake-Maker expressing wondrous amazement about binary floating point numbers:

"You really can get something for nothing! For a particular number of bits, you can define a floating-point representation that gives a greater range and a greater precision than the same number of bits give in a fixed-point representation. Why would you ever use a fixed-point representation?"

Based on your own knowledge or upon research, gently deflate Michael's enthusiasm. I am not looking for essay-length answers; a few sentences will suffice. Answer the question in your own words, and briefly cite any sources that you use.

Because as floating point utilizes a set number of allocated bits for the mantissa and then exponent. What you get actually relates to rounding (Advantages). For numbers like say the 26.6875, it is fine, but when you have very large numbers and are utilizing floating point. You are pretty much using a form of rounding where you are cutting off some of the bits in the mantissa – or if it is big enough – the exponent. This is because you are using only a set number of bits for both mantissa and exponent (Advantages).

A fixed point representation, although inefficient for some smaller cases. Will provide exact certain accuracy for representing extremely large numbers exactly as they are without leaving it to rounding error (Benefits)

In the days of the past, when space exploration defined a nation, extremely large and accurate numbers were required. Fixed point is also easier to compute (Benefits). And as such in the days of space exploration, may be preferred in some cases to provide an 'absolute' value that was also faster to calculate at the cost of say memory – which is often cheaper to have more of back then - as opposed to a floating point one (Benefits).

Works Cited

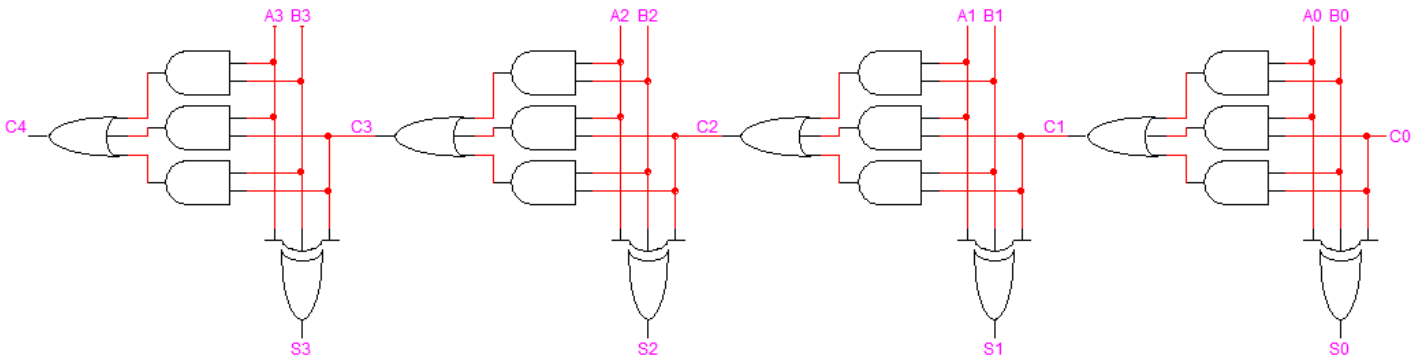
"Advantages and Disadvantages of Floating Point and Fixed Point Representations." *Stack Overflow*. Stack Exchange Inc., 10 Mar. 2012. Web. 22 Mar. 2015.

"Benefits of Using Fixed-Point Hardware." *MathWorks*. The MathWorks Inc. Web. 22 Mar. 2015.

Problem 5 (20 points)

Use the “design by contraction” strategy to derive the equations that implement a 4-bit increment-by-3 circuit from the equations that implement a standard adder.

If it helps to visualize the circuit, imagine that you are starting with a 4-bit adder:



Your strategy should consist of choosing specific values for B (B3 B2 B1 B0) and C0 that will cause the output S (S3 S2 S1 S0) to always equal A (A3 A2 A1 A0) plus 3, and then using Boolean algebra to reduce the equations that contain constants.

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{(i+1)} = A_i B_i + A_i C_i + B_i C_i$$

$S = A + B + C_0$

where $B = 3$

$B = 0011$

where $C = 0$

---

where $B(1) = 1$

$S_1 = A_1 \oplus B_1 \oplus C_1$

$= A_1 \oplus 1 \oplus A_0$

$S_1 = \overline{A_1} \oplus A_0$

$C_2 = A_1 B_1 + A_1 C_1 + B_1 C_1$

$= A_1(1) + A_1 A_0 + (1) A_0$

$= A_1 + A_1 A_0 + A_0$

$= A_1 (1 + A_0) + A_0$

$C_2 = A_1 + A_0$

$S_0 = A_0 \oplus B_0 \oplus C_0 = A_0 \oplus 1 \oplus 0$ where $B(0) = 1$, $C = 0$

$= A_0 \oplus 1 = \overline{A}$ so $S_0 = \overline{A}_0$

$C_1 = A_0 B_0 + A_0 C_0 + B_0 C_0$

$= A_0(1) + A_0(0) + (1)(0)$ so $C_1 = A_0$

$= A_0 + 0 + 0 = A_0$

$S_2 = A_2 \oplus B_2 \oplus C_2$ where $B(2) = 0$

$= A_2 \oplus 0 \oplus (A_1 + A_0) = A_2 \oplus (A_1 + A_0) = S_2$

$C_3 = A_2 B_2 + A_2 C_2 + B_2 C_2$

$= A_2(0) + A_2(A_1 + A_0) + (0) C_2$

$= A_2 (A_1 + A_0) = C_3$

$S_3 = A_3 \oplus B_3 \oplus C_3$ where $B(3) = 0$

$= A_3 \oplus 0 \oplus A_2 (A_1 + A_0)$

$S_3 = A_3 \oplus A_2 (A_1 + A_0)$

$C_4 =$ don't need