

4CS 4341: Digital Logic and Computer Architecture

Homework 5 Guide for Section 002

Question 1

Represent **(a)** A= -83, **(b)** B= 15 **(c)** C= 41 and **(d)** D= -37 each in 2's complement notation using 8 bits (including the sign bit, of course).

- For each number, first convert from decimal to binary ignoring sign.
 - This can be accomplished by successive subtraction of powers of 2.
- If the sign of the number is negative, flip all the bits of the number, then add 1.
 - Alternatively, for speed of manual conversion, beginning with the least significant bit and proceeding toward the most significant bit, copy each 0 bit and the first 1 bit, then flip all the remaining bits.

Part A

83 can be converted to binary by successive subtractions of powers of 2. Start with the number represented by the most significant (highest) bit and proceed toward the least significant bit. If the bit's value is greater than the number being converted, then that bit is zero. If not, the bit is 1. Subtract the amount it represents from the number being converted and continue the process using the remainder.

Bit Position	7	6	5	4	3	2	1	0
Number Being Converted	83	83	19	19	3	3	3	1
Value of Bit Position	128	64	32	16	8	4	2	1
Remainder	<0	19	<0	3	<0	<0	1	0
Result	0	1	0	1	0	0	1	1

To make the result negative in 2's complement notation, flip all the bits, then add 1

Positive Number	0	1	0	1	0	0	1	1
1's complement	1	0	1	0	1	1	0	0
2's complement	1	0	1	0	1	1	0	1

-83 = 0b10101101

Part B

15 = 0b0001111

Part C

41 = 0b00101001

Part D

$$-37 = 11011011$$

Question 2

Perform **(a)** $A + C$, **(b)** $A - C$, **(c)** $C - A$, **(d)** $D - B$, and **(e)** $A + D$

Part A

$$-83 + 41 = -42$$

$$\begin{array}{r} \\ \\ + \\ \hline = \end{array}$$

Part B

$$-83 - 41 = -83 + -41 = -124$$

$$\begin{array}{r} \\ \\ + \\ \hline = \end{array}$$

Part C

$$41 - -83 = 41 + 83 = 124$$

$$\begin{array}{r} \\ \\ + \\ \hline = \end{array}$$

Part D

$$-37 - 15 = -37 + -15 = -52$$

$$\begin{array}{r} \\ \\ + \\ \hline = \end{array}$$

Part E

$$-83 + -37 = -120$$

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \\ +\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Question 3

Prove that the procedure for signed decrement also works for unsigned wrap-around count-down, if overflow is ignored.

Consider breaking this into equivalence classes. Three which come easily to mind are the highest value, lowest value, and all other values. If it works for examples of all these classes then it should work in all cases.

Edge Case: Lowest Value

$$\begin{array}{r} 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ +\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \end{array}$$

So, the counter successfully wraps from the lowest value to the highest value.

Normal Case:

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ +\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ \hline 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{array}$$

Edge Case: Highest Value

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ +\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \end{array}$$

What is going on here? Why does this work? Because of 0, the greatest unsigned number that can be expressed in any number of digits is always exactly 1 less than the number of values that can be expressed by those digits. Ignoring overflow turns those values into a loop. Advancing 1 step fewer than the number of steps in a loop brings you to the same place you would reach by backing up 1 step.

Question 4

Develop combinational functions for C_n , C_{n+1} , and OVF from a_n , b_n , and s_n .

The scenario for this question is that you've been given an adder component which does not provide a carry out or overflow output. You know only the value of the inputs that were added, and the sum. How do you calculate the overflow?

Overflow can be determined by examining the carry in to the most significant bit, and the carry out from that bit. There are 8 combinations for the possible values of a, b, and s any column of addition. From these, can the value of C_n and C_{n+1} be inferred? See the table below.

Inputs			Outputs		
a_n	b_n	s_n	C_n	C_{n+1}	OVF
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	1	1	0

C_n

		b_n, s_n			
		00	01	11	10
a_n	0	0	1	0	1
	1	1	0	1	0

$$!a_n!b_n s_n + !a_n b_n !s_n + a_n !b_n !s_n + a_n b_n s_n$$

C_{n+1}

		b_n, s_n			
		00	01	11	10
a_n	0	0	0	0	1
	1	1	0	1	1

$$b_n !s_n + a_n !s_n + a_n b_n$$

OVF

		b_n, s_n			
		00	01	11	10
a_n	0	0	1	0	0
	1	0	0	0	1

$$!a_n !b_n s_n + a_n b_n !s_n$$