Comp. Org. (CS 1410) Lecture 3 Name:

unsigned overflow

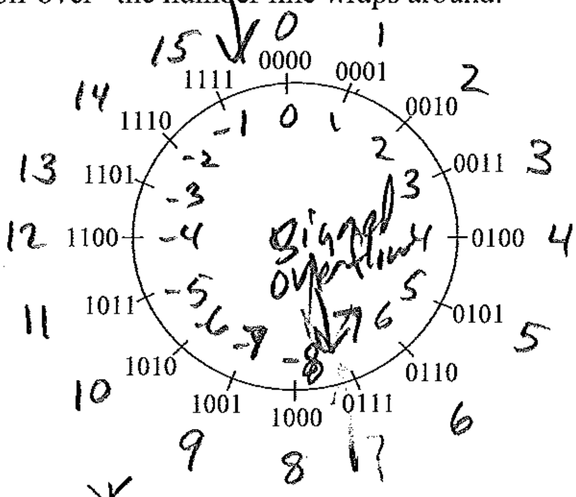
1. Consider 4-bit BINARY numbers, because of "roll-over" the number line wraps around.

a) List unsigned decimal values on the outside

b) List signed (two's complement) decimal values on the inside

c) Mark the point of unsigned overflow

d) Mark the point of signed overflow



Perform the following additions:

e) for unsigned numbers:

$$\begin{array}{rl} & 0100_2\ (4_{10}) \\ + & \underline{0110_2}\ (6_{10}) \\ & 1010\ (10_{10}) \end{array} \qquad \begin{array}{rl} & 1001_2\ (9_{10}) \\ + & \underline{1010_2}\ (10_{10}) \\ & 0011\ (3) \end{array}$$

f) for signed numbers: (two's compliment)

$$\begin{array}{rl} & 0100_2\ (4_{10}) \\ + & \underline{0110_2}\ (6_{10}) \\ & 1010 \end{array} \qquad \begin{array}{rl} & 0100_2\ (4_{10}) \\ + & \underline{1010_2}\ (-6_{10}) \\ & 1110\ \checkmark\ -2 \end{array} \qquad \begin{array}{rl} & 1100_2\ (-4_{10}) \\ + & \underline{1010_2}\ (-6_{10}) \\ & 0110 \end{array}$$

wrong neg. wrong pos.

2. For **4-bit unsigned numbers**, when do we have overflow and get the wrong result during addition? (Hint: think about the carry bits into and/or out of the most-significant bit)

carry out of MSb (most-sign. bit) is a 1

3. a) For **4-bit signed numbers,** complete the following table about signed overflow:

Sign of Operands for addition		**Expected Sign of Result**	**Wrong Sign of Result (indicates overflow)**
Operand 1	**Operand 2**		
+	+	+	−
+	−	These two rows cannot cause signed overflow in addition	
−	+		
−	−	−	+

b) For **4-bit signed numbers**, when do we have overflow and get the wrong result during addition? (Hint: think about the carry bits into and/or out of the most-significant bit)

$$\begin{array}{rl} & 0 \\ + & \underline{0} \\ & 1 \end{array} \qquad \begin{array}{rl} & 1 \\ + & \underline{1} \\ & 0 \end{array}$$

If carry into and out of MSb differ, then signed overflow.

4. How would you subtract two signed, 2's-complement numbers? Try the following:

$$\begin{array}{rl} & 0110_2\ (6_{10}) \\ - & \underline{0111_2}\ (7_{10}) \end{array} \qquad \begin{array}{rl} & 0011_2\ (+3_{10}) \\ - & \underline{1111_2}\ (-1_{10}) \end{array} \qquad \begin{array}{rl} & 1111_2\ (-1_{10}) \\ - & \underline{0011_2}\ (+3_{10}) \end{array}$$

$A - B = A + (-B)$

5. Use Booth's algorithm to calculate the 8-bit product of 0110_2 x 1101_2.

Multiplicand , M

$\boxed{0110}$

– Multiplicand , -M

$\boxed{1010}$

"Initial Product"	"Multiplier"	"Previous bit"	
$\boxed{0000}$	$\boxed{1101}$	$\boxed{0}$	start of run add – M
+ 1010			
1010	1101	0	
1101	0110	1	end of run add M
+ 0110			
0011	0110	1	
0001	1011	0	start of run add – M
+ 1010			
1011	1011	0	
1101	1101	1	middle of run
1110	1110	1	

Booth table

current bit of multiplier	previous bit	Action
0	0	don't add
0	1	end run of 1's add multiplicand
1	0	start of run add neg. multiplicand
1	1	middle of run don't add.