Arithmetic Operations on Signed and Unsigned Numbers and Overflow Detection

Unsigned Number:

- Unsigned numbers are positive numbers.
- Unsigned number addition is just like regular binary addition.
- We start the addition by adding corresponding digits from the right hand side and gradually move leftwards.

$$\begin{array}{r} (11011)_2 \\ +\ (10011)_2 \\ \hline (101110)_2 \\ \hline \end{array}$$

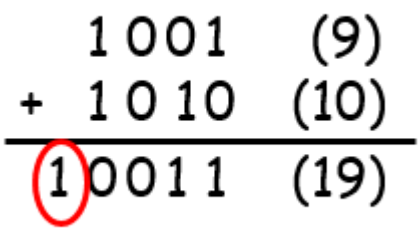
- To check if our answers match we can convert the binary numbers to decimal and then cross check.
- For subtraction, each pair of digits are subtracted and “borrowing” is done whenever required.

$$\begin{array}{r} (11001)_2 \\ -\ (10011)_2 \\ \hline (00110)_2 \\ \hline \end{array}$$

Overflows:

- Carry-out can be used to detect overflow

- The largest number that we can represent with 4-bits using unsigned numbers is 15
- Suppose that we are adding 4-bit numbers: 9 (1001) and 10 (1010).



•The value 19 cannot be represented with 4-bits

•When operating with unsigned numbers, **a carry-out** of 1 **can be used to indicate overflow.**

Signed Number:

- Overflow:
  - With two's complement, the largest representable decimal number is +7, and the smallest is -8.

- What if you try to compute 4 + 5, or (-4) + (-5)?

```
  0100   (+4)        1100   (-4)
+ 0101   (+5)      + 1011   (-5)
-------            -------
 01001   (-7)       10111   (+7)
```

- We cannot just include the carry out to produce a five-digit result, as for unsigned addition. If we did, (-4) + (-5) would result in +7!
- Also, unlike the case with unsigned numbers, the carry out cannot be used to detect overflow.
  - In the example above, the carry out is 0 but there is overflow.
  - Conversely, there are situations where the carry out is 1 but there is no overflow.
- The easiest way to detect signed overflow is to look at all the sign bits.

```
  0100   (+4)        1100   (-4)
+ 0101   (+5)      + 1011   (-5)
-------            -------
 01001   (-7)       10111   (+7)
```

- Overflow occurs only in the two situations above:
  - If you add two positive numbers and get a negative result.
  - If you add two negative numbers and get a positive result.

- Signed binary numbers are of a fixed range.
- If the result of addition/subtraction goes beyond this range, overflow occurs.
- In case of unsigned no, if there is a carry out in MSB, then overflow has occurred.
- In signed number, two conditions under which overflow can occur are:
  (i) positive add positive gives negative
  (ii) negative add negative gives positive
-

2s Complement Addition/Subtraction

- Algorithm for addition, A + B:

1. Perform binary addition on the two numbers.
2. Ignore the carry out of the MSB (most significant bit).

3. Check for overflow: Overflow occurs if the ‘carry in’ and ‘carry out’ of the MSB are different, or if the result is the opposite sign of A and B.

- Algorithm for subtraction, A – B:

  A – B = A + (–B)

1. Take 2s complement of B by inverting all the bits and adding 1.
2. Add the 2s complement of B to A.

Example 1

Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X – Y and (b) Y - X by using 2's complement.

(a)		
	X =	1010100
	2's complement of Y =	+0111101
	Sum =	10010001
	Discard end carry =	0010001
	Answer. X – Y =	0010001

(b)		
	Y =	1000011
	2's complement of X =	+ 0101100
	Sum =	1101111

There is no end carry. Therefore, the answer is $Y - X$ = – (2's complement of 1101111) = – 0010001.

Example 2: 4 bit binary system

```
  +3        0011
+ +4      + 0100
----      -------
  +7        0111
----      -------
```

```
  -2        1110
+ -6      + 1010
----      -------
  -8       11000
----      -------
```

```
  +6        0110
+ -3      + 1101
----      -------
  +3       10011
----      -------
```

```
  +4        0100
+ -7      + 1001
----      -------
  -3        1101
----      -------
```

```
  -3        1101
+ -6      + 1010
----      -------
  -9       10111
----      -------
```

```
  +5        0101
+ +6      + 0110
----      -------
 +11        1011
----      -------
```

Which of the above is/are overflow(s)?

1s Complement Addition/Subtraction

- Algorithm for addition, A + B:
  1. Perform binary addition on the two numbers.
  2. If there is a carry out of the MSB, add 1 to the result.
  3. Check for overflow: Overflow occurs if result is opposite sign of A and B.
- Algorithm for subtraction, A – B:

  A – B = A + (–B)
  1. Take 1s complement of B by inverting all the bits.
  2. Add the 1s complement of B to A.

Example 3

Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction (a) X – Y and (b) Y - X by using 1's complement.

(a) $X - Y = 1010100 - 1000011$

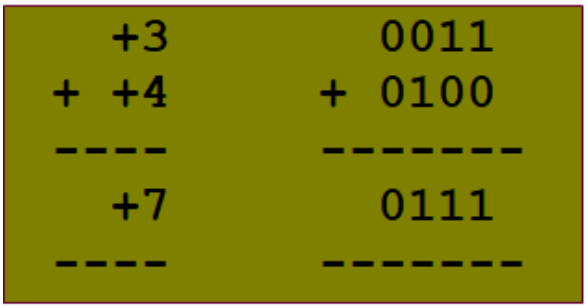
$$
\begin{aligned}
X &= \quad 1010100 \\
\text{1's complement of } Y &= + \; 0111100 \\
\text{Sum} &= \quad 10010000 \\
\text{End-around carry} &= \quad + \qquad 1 \\
\text{Answer. } X - Y &= \quad 0010001
\end{aligned}
$$

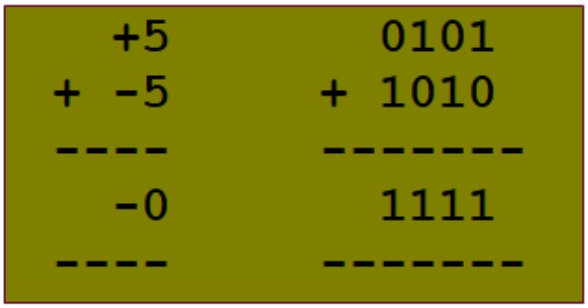
(b) $Y - X = 1000011 - 1010100$

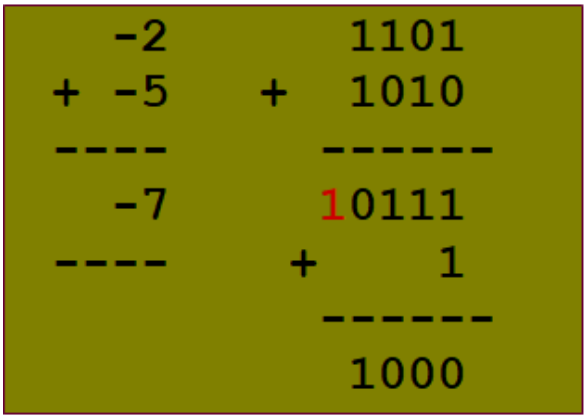
$$
\begin{aligned}
Y &= \quad 1000011 \\
\text{1's complement of } X &= \; +0101011 \\
\text{Sum} &= \quad 1101110
\end{aligned}
$$

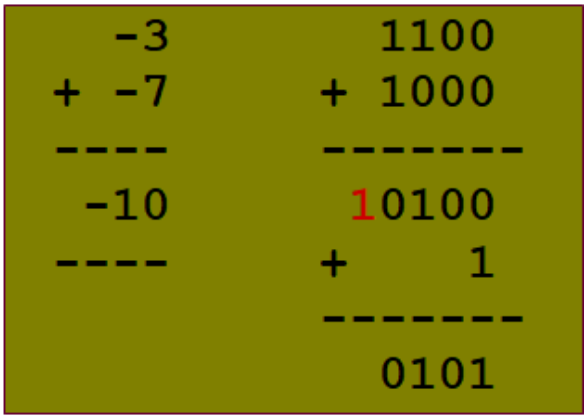
There is no end carry, Therefore, the answer is $Y - X = -$ (1's complement of 1101110) = – 0010001.

Example 2: 4 bit binary system









Which of the above is/are overflow(s)?