Key to Tutorial 2
Encoded Integers

Exercise 1

Work out the decimal range of signed and unsigned binary numbers for each of the following word lengths: 4, 8, 16, 32 and *n* bits (for 32 and *n* bits, you can use a power of two notation).

Bits	Unsigned		Signed	
	Min	**Max**	**Min**	**Max**
4	0	15	-8	7
8	0	255	-128	127
16	0	65535	-32768	32767
32	0	$2^{32} - 1$	-2^{31}	$2^{31} - 1$
n	0	$2^{n} - 1$	-2^{n-1}	$2^{n-1} - 1$

Exercise 2

Encode the following negative integers into 8-bit signed words: -1, -35, -60, -127, -128.

Here is the method of encoding a negative integer into an *n*-bit signed word:

- Convert its absolute value into an *n*-bit binary word.
- Work out the *n*-bit two's complement of that word.

$1_{10} = 0000\ 0001_2 \rightarrow -1_{10} = 1111\ 1111_2$

$35_{10} = 0010\ 0011_2 \rightarrow -35_{10} = 1101\ 1101_2$

$60_{10} = 0011\ 1100_2 \rightarrow -60_{10} = 1100\ 0100_2$

$127_{10} = 0111\ 1111_2 \rightarrow -127_{10} = 1000\ 0001_2$

$128_{10} = 1000\ 0000_2 \rightarrow -128_{10} = 1000\ 0000_2$

Two's complement

Exercise 3

For the whole exercise, we use the two's complement encoding (used to encode signed integers).

1. Let us consider the following 8-bit words (in their hexadecimal form): FF, 27, 7E, 9A, B7, 6D, 81. Do they represent positive or negative integers?

Word	Sign	Binary form	MSB
FF	Negative	**1**111 1111	1
27	Positive	**0**010 0111	0
7E	Positive	**0**111 1110	0
9A	Negative	**1**001 1010	1
B7	Negative	**1**011 0111	1
6D	Positive	**0**110 1101	0
81	Negative	**1**000 0001	1

2. Write down the range of 8-bit words used to encode positive integers (use the hexadecimal form).

   $\mathbf{00_{16}} \leq$ positive integers $\leq \mathbf{7F_{16}}$ (MSB = 0)

3. Write down the range of 8-bit words used to encode negative integers (use the hexadecimal form).

   $\mathbf{80_{16}} \leq$ negative integers $\leq \mathbf{FF_{16}}$ (MSB = 1)

4. Complete the following table for the addition (write down *Yes* or *No* for the signed overflow).

1st operand	+	2nd operand	=	Result	Signed Overflow
Positive	+	Positive	=	Positive	No
Positive	**+**	**Positive**	**=**	**Negative**	**Yes**
Positive	+	Negative	=	Positive	No
Positive	+	Negative	=	Negative	No
Negative	**+**	**Negative**	**=**	**Positive**	**Yes**
Negative	+	Negative	=	Negative	No

   According to the above table, a signed overflow occurs when the signs of the two operands are the same and the sign of the result is different:
   - Both of the operands are positive (MSB = 0) and the result is negative (MSB = 1).
   - Both of the operands are negative (MSB = 1) and the result is positive (MSB = 0).

5. Complete the following table for the subtraction (write down *Yes* or *No* for the signed overflow).

1st operand	-	2nd operand	=	Result	Signed Overflow
Positive	-	Positive	=	Positive	No
Positive	-	Positive	=	Negative	No
Positive	-	Negative	=	Positive	No
Positive	**-**	**Negative**	**=**	**Negative**	**Yes**
Negative	**-**	**Positive**	**=**	**Positive**	**Yes**
Negative	-	Positive	=	Negative	No
Negative	-	Negative	=	Positive	No
Negative	-	Negative	=	Negative	No

According to the above table, a signed overflow occurs when the sign of the two operands are different and the sign of the result is the sign of the 2nd operand.

Exercise 4

Perform the following 8-bit binary operations (the two operands and the result are 8 bits wide). Then, convert the result into unsigned and signed decimal values. If an overflow occurs, write down 'ERROR' instead of the decimal value.

Operation	Binary Result	Decimal Value	
		Unsigned	Signed
`11101101 + 11111010`	`(1) 1110 0111`	**ERROR**	**-25**
`11101000 - 11001010`	`0001 1110`	**30**	**30**
`01011100 - 10011010`	`1100 0010`	**ERROR**	**ERROR**
`01111110 + 00001001`	`1000 0111`	**135**	**ERROR**
`11011011 - 00011010`	`1100 0001`	**193**	**-63**
`10000000 + 11110110`	`(1) 0111 0110`	**ERROR**	**ERROR**
`10010011 - 00001010`	`1000 1001`	**137**	**-119**

Method for unsigned integers:

- Addition: if the expected result is 9 bits wide → overflow.
- Subtraction: if the left operand is smaller than the right one → overflow.

Method for signed integers (see exercise 3):

If the sign of the 8-bit result is wrong → overflow (just compare the signs).

For instance:

- Positive + Positive = Positive only → If the result is negative → overflow
- Positive – Positive = Positive or Negative → Whatever the result → no overflow
- Etc.