Key to Tutorial 4
Encoded Integers

Exercise 1: Signed Numbers

1. Encode the following negative integers into 8-bit signed words:

   Here is the method of encoding a negative integer into *n*-bit signed words:
   - Convert its absolute value into its *n*-bit binary form.
   - Work out the *n*-bit two's complement of the absolute value.

1_{10}	=	00000001_2	→	-1_{10}	=	11111111_2
29_{10}	=	00011101_2	→	-29_{10}	=	11100011_2
40_{10}	=	00101000_2	→	-40_{10}	=	11011000_2
127_{10}	=	01111111_2	→	-127_{10}	=	10000001_2
128_{10}	=	10000000_2	→	-128_{10}	=	10000000_2

Two's complement

2. Write down the hexadecimal form of the 8-bit signed binary words used to encode the smallest and largest values of positive and negative integers.

   - $00_{16} \leq$ positive integers $\leq 7F_{16}$ (MSB = 0)
   - $80_{16} \leq$ negative integers $\leq FF_{16}$ (MSB = 1)

Exercise 2: 8-bit Signed Operations

1. 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**
11110101 + 11111010	1110 1111	**ERROR**	**–17**
11101000 – 11000110	0010 0010	**34**	**34**
01011110 – 10011110	1100 0000	**ERROR**	**ERROR**
01111110 + 00000101	1000 0011	**131**	**ERROR**
11001011 – 00011010	1011 0001	**177**	**–79**
10000000 + 11111010	0111 1010	**ERROR**	**ERROR**
10000011 – 00001010	0111 1001	**121**	**ERROR**

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:
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.

2. When it comes to adding 8-bit signed binary words, find a way to detect any overflow by comparing the sign bit of the result and those of both operands. Be careful not to confuse this overflow with the carry, which is an unsigned overflow (the 9th bit of the result).

Let us consider all the possibilities:

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:

- Both of the operands are positive (sign bits = 0) and the result is negative (sign bit = 1).
- Both of the operands are negative (sign bits = 1) and the result is negative (sign bit = 0).

Exercise 3: Gray Code

1. Write down the Gray code numbers from 0 to 15 and deduce the Gray code of 17, 24 and 31.

Decimal	Natural Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Axis of symmetry for the LSB

Axis of symmetry for the 2 LSBs

Axis of symmetry for the 3 LSBs

To encode numbers from 0 to 31, five bits are required. From the table above, we can conclude that there is an axis of symmetry for the four LSBs. This axis is located between the values 15 and 16:

- The four LSBs of 15 are equal to the four LSBs of 16 (and the MSB of 16 is one).
- The four LSBs of 14 are equal to the four LSBs of 17 (and the MSB of 17 is one).
- …
- The four LSBs of 7 are equal to the four LSBs of 24 (and the MSB of 24 is one).
- …
- The four LSBs of 0 are equal to the four LSBs of 31 (and the MSB of 31 is one).

Therefore:

- 14_{10} = 0**1001**$_{Gray}$ → 17_{10} = 1**1001**$_{Gray}$
- 7_{10} = 0**0100**$_{Gray}$ → 24_{10} = 1**0100**$_{Gray}$
- 0_{10} = 0**0000**$_{Gray}$ → 31_{10} = 1**0000**$_{Gray}$

2. To encode a binary number (*Xb*) into a Gray code number (*Xg*), we can apply the following method:

   Assuming that:
   - $Xb = b_n\ b_{n-1}\ b_{n-2} \ldots b_1\ b_0$
   - $Xg = g_n\ g_{n-1}\ g_{n-2} \ldots g_1\ g_0$

   We have:
   - $g_n = b_n$
   - $g_{n-1} = 0$ if and only if $b_n = b_{n-1}$
   - $g_{n-2} = 0$ if and only if $b_{n-1} = b_{n-2}$
   - $g_0 = 0$ if and only if $b_0 = b_1$

   In other words, two consecutive bits are compared: if they are equal, we write down '0'; otherwise, we write down '1'.

   Example: $11001011_2 \rightarrow 10101110_{Gray}$

   Convert the following decimal numbers into Gray code numbers: 42, 109, 128.

Decimal	Natural Binary	Gray Code
42	0010 1010	0011 1111
109	0110 1101	0101 1011
128	1000 0000	1100 0000

3. To encode a Gray code number into a binary number, you can apply the following method:

   Count the number of 1s from the most significant bit up to the same position as the bit we are looking for: if the number of 1s is even, the value of the bit is 0; otherwise, it is 1.

   Example: $1011_{Gray} \rightarrow 1101_2$

   Convert the following Gray code numbers into decimal numbers:
   - 11001000
   - 11101011
   - 10000001

Gray Code	Natural Binary	Decimal
1100 1000	1000 1111	143
1110 1011	1011 0010	178
1000 0001	1111 1110	254

Exercise 4: Binary Coded Decimal (BCD)

Convert the following decimal numbers into BCD numbers: 421; 404; 3,009; 7,408.

Decimal	BCD
421	0100 0010 0001
404	0100 0000 0100
3,009	0011 0000 0000 1001
7,408	0111 0100 0000 1000