

INTRODUCTION TO COMPUTER SCIENCE

The John Paul II Catholic University of Lublin - mgr Sara Jurczyk – 2022/2023

numeral systems

Exercise 1. Do the conversions below:

$$52_{(10)} = \dots\dots\dots(2)$$

$$13_{(10)} = \dots\dots\dots(2)$$

$$64_{(10)} = \dots\dots\dots(2)$$

$$27_{(10)} = \dots\dots\dots(2)$$

$$107_{(10)} = \dots\dots\dots(2)$$

$$17_{(10)} = \dots\dots\dots(2)$$

$$25_{(10)} = \dots\dots\dots(2)$$

$$67_{(10)} = \dots\dots\dots(2)$$

$$67_{(10)} = \dots\dots\dots(16)$$

Check your results by converting them back to decimal system.

Horner's algorithm (also Horner Scheme, Horner's rule and Horner's method):

Horner's rule is an efficient algorithm for converting a number written in any base into its decimal notation.

To get the decimal value of a number in base b , create a table of two rows, in which in the first row you write the digit pattern to be converted. The first bit on the left is rewritten to the cell below. Then, you multiply every number you get in the second row by base b and add the next bit on the right (value in the next column). You repeat the process until you get the decimal value in the last cell.

Example:

$$x = (11001111)_2$$

c_k	1	1	0	0	1	1	1	1
b_k	1	3	6	12	25	51	103	207

Thus:

$$(11001111)_2 = (207)_{10}$$

Exercise 2. Use Horner's algorithm to get the decimal values of the numbers below:

a) $101001_{(2)} = \dots\dots\dots_{(10)}$

b) $110111_{(2)} = \dots\dots\dots_{(10)}$

c) $1000000_{(2)} = \dots\dots\dots_{(10)}$

d) $1001111_{(2)} = \dots\dots\dots_{(10)}$

e) $1234_{(16)} = \dots\dots\dots_{(10)}$

Binary Addition:

Binary addition follows the same rules as addition in the decimal system except that rather than carrying a 1 over when the values added equal 10, carry over occurs when the result of addition equals 2. So, in the binary system:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0, \text{ carry over the } 1$$

Example:

$$\begin{array}{r} 1101 \\ + 101 \\ \hline 10010 \end{array}$$

Exercise 3. Perform the binary addition:

a) $\begin{array}{r} 1111 \\ + 101 \\ \hline \end{array}$

b) $\begin{array}{r} 10001 \\ + 1111 \\ \hline \end{array}$

c) $\begin{array}{r} 1111 \\ + 111 \\ \hline \end{array}$

d) $\begin{array}{r} 1101 \\ 101 \\ + 10010 \\ \hline \end{array}$

Binary Subtraction * - optional:

Borrowing occurs in any instance where the number that is subtracted is larger than the number it is being subtracted from. In binary subtraction, the only case where borrowing is necessary is when 1 is subtracted from 0. When this occurs, you have to borrow 1 from the column on the left. If the following column is also 0, borrowing will have to occur from each subsequent column until a column with a value of 1 can be reduced to 0. The rules are:

$$0 - 0 = 0$$

$$0 - 1 = 1, \text{ borrow } 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Exercise 4. Perform the binary subtraction:

a)
$$\begin{array}{r} 1111 \\ - 101 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 10001 \\ - 1111 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 1111 \\ - 111 \\ \hline \end{array}$$

Two's complement notation:

Two's complement notation is used to write both positive and negative numbers.

To get the two's complement negative notation of an integer, you write out the number in binary. If it is needed, write zeros on the left to write the number using expected number of bits. You then invert the digits so that every 0 becomes 1 and every 1 becomes 0. The last step is to add one to the result.

Example:

Suppose we are working with 6 bit quantities and suppose we want to find how -21 would be expressed in two's complement notation. First we write out 21 in binary form using 6 bits.

21		1
10		0
5		1
2		0
1		1
0		

We get 010101.

The next step is to invert the digits (0 becomes 1, 1 becomes 0): 101010.

The last step is to add 1 to the result above. Finally, we get 101011.

Now let's perform the reverse conversion and get the decimal value of 101011. While getting the decimal value of a number in two's complement notation one have to remeber that the first bit on the left is a sign bit and 1 is treated as -1.

$$\underline{1}01011 = \underline{-1} * 2^{6-1} + 1 + 1*2 + 0*4 + 1*8 + 0*16 = -32 + 11 = -21$$

Note, that to get the two's complement positive notation of an integer, you just write out the number in binary.

Exercise 5. Represent numbers below in 2's complement notation using 8 bits:

- a) -35
- b) -17
- c) -30
- d) -55
- e) -3
- f) 7
- g) -33

And then check your results by converting them back to decimal system.

Horner's algorithm for two's complement notation:

We also can use Horner's algorithm to get the decimal value of a number in two's complement notation.

Example:

Let's get the decimal value of a number 10111110.

i	7	6	5	4	3	2	1	0
c_i	1	0	1	1	1	1	1	0
$p = 2$	-1	-2	-3	-5	-9	-17	-33	-66

The first bit on the left is a sign bit so 1 is treated as -1.

Thus:

$$(10111110)_{u2} = -66$$

Exercise 6. Get the decimal value of numbers from exercise 5 using Horner's algorithm.