Department of Mathematics
Dr. Ulf Lorenz
Christian Brandenburg

TECHNISCHE
UNIVERSITÄT
DARMSTADT

winter term 2008/2009
10/11/2008

Introduction to Mathematical Software

2nd Exercise Sheet

Exercise 1 (Representation of integers)

Preliminary Remark: In this exercise, you are supposed to develop an understanding of how integers are represented in computers and how computations with them are performed. Most of this exercise is a tutorial, but there are also some small exercises within each section for you to check if you are familiar with the material.
You should not spend much more than half of the lab session on this exercise. If you don't get through with this exercise and still have questions, come and see us in our office hours.

Binary Representation

For computers the least logical entity is a *bit*, which can have two states – typically written as 0 and 1. Thus, in most programming languages the natural numbers are represented binary, i.e. they are written in base 2.
For example, the pattern $1101_{(2)}$ means

$$\begin{aligned} 1101_{(2)} &= 1 \cdot 2^3_{(10)} + 1 \cdot 2^2_{(10)} + 0 \cdot 2^1_{(10)} + 1 \cdot 2^0_{(10)} \\ &= 8_{(10)} + 4_{(10)} + 1_{(10)} \\ &= 13_{(10)} \end{aligned}$$

In fact, this representation works in exactly the same way as we are used to write numbers in the decimal system, where we have base 10:

$$\begin{aligned} 423_{(10)} &= 4 \cdot 10^2_{(10)} + 2 \cdot 10^1_{(10)} + 3 \cdot 10^0_{(10)} \\ &= 400_{(10)} + 20_{(10)} + 3_{(10)} \end{aligned}$$

Here, we have the letters $0, 1, 2, \ldots, 9 = 10 - 1$ and each position has 10 times the value of the position one further to the right.

In the same manner, in base 2 we have the letters 0 and 1 and each position is 2 times the value of the position one further to the right.

Conversion

As you have seen, conversion from base 2 to base 10 is straightforward; we just add the values of the positions with non-zero digits. Unfortunately, the other way round is more involved.

For converting from base 10 to base 2 there are two approaches, the naïve and the systematic one. In the naïve approach we iteratively find the largest power of 2 which is less than or equal to the number we want to convert, place a 1 at the corresponding position of the binary representation and substract this power of two from our decimal number.
For example, if we want to convert $817_{(10)}$ to the binary representation, we find that $2^9 = 512$ is the largest power of 2 less than or equal to $817_{(10)}$. Thus, we place a 1 at the 9th position from the right (note that we start counting positions with 0!) After substracting, we have $817 - 512 = 305$, we find that $2^8 = 256$, place a 1 at the 8th position and substract $305 - 256 = 49$. Continuing, we place ones at positions 5,4 and 0. Thus, the binary representation of $817_{(10)}$ is given by $1100110001_{(2)}$.

In the systematic approach, we make use of the following properties:

- if we add a zero at the end of a binary number, we multiply it by 2
- if we erase the rightmost digit, we divide by two, neglecting the remainder
- every number can be written in the form $2n$ or $2n + 1$
- we can read of the last digit of a binary number by checking if it is even or odd

Combining these properties, we arrive at an algorithm for converting numbers from base 10 (and in fact from any base) to base 2:

- decide whether the number is even or odd; if it is even, we place a 0, otherwise we append a 1 to the left of the binary representation (in the first iteration, we just place the digit).
- divide by 2 without remainder and iterate until you arrive at 0.

With our example, this algorithm goes as follows:

decimal	binary
817	
408	1
204	01
102	001
51	0001
25	10001

decimal	binary
12	110001
6	0110001
3	00110001
1	100110001
0	1100110001

Exercises:

(a) Convert the following decimals to binaries: $4_{(10)}$, $5_{(10)}$, $78_{(10)}$, $127_{(10)}$

(b) Convert the following binaries to decimals: $111_{(2)}$, $10101_{(2)}$, $10111_{(2)}$, $1110111_{(2)}$

Negative Numbers

To also represent negative integers, somehow the "$-$"-sign has to be represented. In the "sign and magnitude"-approach, this is achieved by first fixing the number of bits the representation of an integer may use (e.g. 8 bits) and then using the leftmost bit for the sign. However, this approach has several disadvantages: Firstly, there exist two represenations for "zero", i.e. $+0$ and -0, and secondly, for all arithmatic operations one needs special cases to check for signs.

Two's Complement

Today, negative integers are typically represented by their "Two's Complement". The Two's Complement of a negative integer n is built by inverting all bits of the representation of $|n|$ and then adding 1 to it. To represent e.g. $-13_{(10)}$ in an 8 bits wide representation, you first represent $13_{(10)} = 00001101_{(2)}$, then invert all its bits to $11110010_{(2)}$ and then add 1 to it, thus $-13_{(10)} = 11110010_{(2)} + 1_{(2)} = 11110011_{(2)}$. Whether a signed binary number is positive or negative can be determined by looking at the highest bit, which is 0 for nonnegative and 1 for negative numbers.

Exercises:

(a) Find the 8 bits wide representations for $-27_{(10)}$ and $-78_{(10)}$.

(b) Which integers may be represented at all with a

  i. 8 bits wide representation?

  ii. 32 bits wide representation?

(c) How could you build the Two's Complement using decimal calculations?

Arithmetic

Arithmatic for the binary system works exactly as for the decimal system, which means that you can reuse all the algorithms you learned in school. In fact, the correctness of these algorithms is independent from the base.

Addition

Computing $15_{(10)} + 5_{(10)} = 20_{(10)}$ in a binary 8-bit representation:

$$\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0_1 & 0_1 & 1_1 & 0_1 & 1 \\
\hline
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}$$

Note that, in this example, we produce a carry in each bit.

Substraction

For substracting two binary numbers, we can again use the algorithm taught in school, or we can reduce the problem to addition by negating the second argument (using the two's complement).

For computing $5_{(10)} - 15_{(10)} = -10_{(10)}$ in a binary 8-bit representation, we first negate the second argument using two's complement: $15_{(10)}$ is in binary representation given by $00001111_{(2)}$, so its two's complement is $11110000_{(2)} + 1_{(2)} = 11110001_{(2)}$. Thus, we can perform an addition:

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0_1 & 1 \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Multiplication and Division

They also work as taught in school.

Exercises:

Do the following calculations like a computer using 8 bits:

(a) $13 + 8$

(b) $27 - 78$

(c) $-27 - 78$

(d) $-78 - 78$

(e) $7 \cdot 6$

What happens in (d)?

Exercise 2 (Lists and sets in Maple)

(a) Explain the difference of lists and sets in Maple.

(b) Use Maple to find the common divisors of 23545800, 25491186 and 229420674.

(c) Let Maple evaluate the function sin for all the solutions of the equation

$$x^4 - 4x^3\pi + \frac{26}{9}x^2\pi^2 + \frac{4}{9}x\pi^3 - \frac{1}{3}\pi^4 = 0$$

by using the `map` command.

(d) Program a Maple function using lists that sums up all factorials that are less than a given natural number.

(e) Program a Maple function that returns for a given set M and a given number k all the subsets of M with exactly k elements.

Exercise 3 (Advanced Maple*)

(a) Find out which of the following numbers are factorials:
720, 4320, 39916800, 25852016738884976640000, 26976017466662584320000

(b) Find out by searching the internet what “perfect number” means. Then decide which of the following numbers are perfect:
28, 120, 496, 8192, 13164036458569648337239753460458722910223472318386943117783728128