HPPS aflevering 1

August Pallesen, Sky Wong og Alexander Husted

November 2023

Indhold

1 Introduction

In general the quality of our functions are good when used probably. For numbers within their limitations i.e. numbers that are positive and less than or equal to 255. We haven't used any assertion, which means that instead of returning error they just crash.

Our test can be run with our makefile using the following command:

```
make test_numbers.c && ./test_numbers
```

2 Tasks

2.1 Converting to and from C integers

Functionally correct
both bits_from_int and bits_to_int works for the positive numbers.

Improvement
It wouldn't need improvement if it's name was unsigned_bits8_from_int. But it would be better if it was also able to convert negative integers. Same argument follows for bits_to_int.

2.2 Implementing addition

Functionally correct
We know the algorithm works, because we followed the formulas from the assignment and tested it thoroughly.

Improvement
There is room to optimize the structure of our code, to make a easier to read.
The output cannot be greater than 255, since we only have 256 combination to work with.

2.3 Implementing negation

Functionally correct
We have a problem here. We cannot test the functionality, since we can't convert negative numbers into an integer and print it. Instead we have printed the bit representation and checked it by hand.

Improvement
We don't believe this can be improved.

2.4 Implementing multiplication

Functionally correct
We followed the formula from the assignment and tested our results, everything seems to function correctly.
The output cannot be greater than 255, since we only have 256 combination to work with.

Improvement
As with a lot of the other function it doesn't work with negative numbers.

3 Questions

1
The function works as intended. The applied algorithm works for negative numbers as well. Addition is the same for binary numbers whether they are negative or not, it's a question about interpretation. That is why we haven't used the bits8_to_int in the test, since it assumes positive numbers.

2
The functions provides the correct results, when being passed negative numbers in two's compliment representation. The argument here is the same as before; multiplication is the same for binary numbers whether they are negative or not, it's a question about interpretation. Again this is the reason we haven't used the bits8_to_int in the test, since it assumes positive numbers.

3
In order to implement a bits8_sub() function for subtracting 8-bit numbers, the function needs to take two bits8 structures as input. The function the does subtraction by subtracting the second input from the first one. The function should handle the chance of underflow, by checking if the first input is smaller than the second. If it is, tweak the result as needed, maybe set it to zero or apply some strategy for underflow. Finally, return the result as a new bits8 structure.