### Representation of Negative Binary Numbers in Two’s Complement

In the following we will briefly look at the question of how negative binary numbers can be represented in a computer system. For the sake of simplicity, we will restrict ourselves to numbers without decimal places. There are several possibilities for this. The easiest to

tional symbols such as + and −. This has the advantage that no additional control logic is understand and most common is the two’s complement. This procedure offers the possibility to represent negative numbers in the binary system without having to resort to addirequired in digital circuits. The basic idea is as follows:

First you define a constant number of digits, which all binary numbers must have. In the 8-

used. If necessary, shorter numbers are then filled with leading zeros, i.e., bit two’s complement, for example, as the name suggests, numbers with eight digits aredisplayed as 0001 01102, for example. left, indicates whether the number is101102 is then

The most significant bit, i.e., the bit furthest to the negative or not. If this bit is 0, a non-negative number is present. With negative numbers this bit has the value 1.

In the 8-bit two’s complement, all numbers between are therefore non-negative numbers. All numbers between are negative.

The determination of the (decimal) value of numbers in two’s complement is slightly diftakes into account that the most significant bit represents the sign by giving the highestcomplement is calculated as follows:power a negative sign. The value b of an (n + 1) digit binary number bn . b0 in two’s ferent from that described above for general binary numbers. The calculation implicitly b = bn · −2n + ni = 0∑−1bi ·2i

The following table shows the value range of eight-digit binary numbers, once as an unsigned decimal number and once as an interpretation in the 8-bit two’s complement. As you can see, the largest positive number that can be displayed in the two’s complement is no longer 255, but only 127, although the negative numbers up to and including −128 can now also be displayed. The value range has thus shifted from 0 ... 255 to −128 ... 127 due to the use of the two’s complement. In general, it is therefore important to know how the value of a binary number should be interpreted.

It is easy to think about the fact that this method can be easily transferred to longer binary numbers, e.g., 32 bit or 64 bit.

Table 3: Value range of binary numbers in 8-bit two's complement

Source: Brückmann, 2013.

#### Example: Representation of negative binary numbers in two’s complement

You already know how to convert binary numbers to the decimal system to determine their value as an unsigned decimal number. Below are some examples of how to interpret the values of negative binary numbers in 8-bit two’s complement:1010 01012==1·=−−128+32+4+191−−102277 +1·2+1·225+1·2+1·212+1·2+1·200

100001112=1·==−−128+4+2+1121210===1·−−128+6464−1027 +1·26

1100 0000

Binary Arithmetic 1100 11002=1·==−−128+64+8+452−1027 +1·26 +1·23 +1·22

In this section we will briefly describe how to calculate in the binary system. We will deal with the basic arithmetic operations of addition, subtraction, and multiplication. You will see that the execution of these arithmetical operations in the binary system is very simple and can therefore be realized efficiently with logic circuits, which is an enormous gain for computer science. In order not to go beyond the scope of this course, we will not consider the somewhat complicated division of binary numbers.

Addition works just like the written addition you learned in elementary school. The numbers are written on top of each other and then added up in places from right to left. The following rules apply to the addition of the individual digits:The fourth formula means that we have a carry of 00112222 +0+1+0+12222=0=1=1=1022221 at the corresponding position, which

is added to the next position.

Let us make some examples of addition:1. Let b1 := 10112 and b2 := 00112, we add the numbers from right to left.

bit 0 of the result is 0 and we note a carry of 1 for bit 1.Bit 0 is 1 for both numbers, so according to the above rule the result is 12 +12 =102, so

Bit 1 is again 1 for both numbers, the sum of which is again we also have a carry for bit 2.bit 0. The result is therefore 12 +12 +12 =102 +12 =112, so bit 1 of the result is 1 and1 +1 = 10 . In addition, however, we must also take into account the carry that resulted from the addition at2 2 2

So bit 2 of the result is 1. There is no carry to the next bit this time.Bit 2 is 0 for both numbers, but we have to add the carry bit. So it is 02 + 02 + 12 = 12.

result remains for bit 3.Bit 3 of b1 is 1, bit 3 of b2 is 0. The sum of 12 +02 = 1. Since no carry is added, this

2. So it’s Let b1 := 110110112 + 00112 and b2 = 11102 := 10012. 2 and we add the numbers again from right to left.

have a carry of 1.The addition of the bits in place 0 gives 12 +12 =102, so bit 0 of the result is 0 and we

For bit 1 the result is accordingly 02 +02 +12 = 12. There is no carry.

1Bit 2. 2 of b1 is 1, bit 2 of b2 is 0. There is no carry. Thus bit 2 of the result is 12 +02 +02 =

result is therefore 0 and another digit is added to the result which is set to 1.Bit 3 is 1 for both numbers. There is no carry. This results in 12 +12 = 102 bit 3 of the

well have more digits than the two summands.Thus, it is 11012 + 10012 = 101102, so as you can see, the result of the addition may

Let’s look at the subtraction. This works similar to addition. Here too, the numbers are subtracted from right to left and here too there are four rules:

00

value 1 to the next digit, as in the case of addition, we subtract the value 1 from the next0digit in the case of a negative carry. At the same time a 1 is written to the current position2 −12 = −12 means that we can have a negative carry this time. So instead of adding the112222 −−−−01012222=0=1=0=−22212

as the result. The procedure is similar to the written subtraction of decimal numbers.

Let’s take a few examples here as well:1. Let b1 := 10112 and b2 := 01112 and subtract the numbers in places from right to left.

For bit 0 the result is 12 −12 =02. We do not have a carry.

For bit 1 the result is 12 −12 =02, again there is no carry.

For bit 2 the result is digit and note 1 as the result for the current digit.02 −12 = −12, so this time we have a carry from −1 to the next b1:

Finally, for bit 3 the result is12 −02 −12 = 02, taking the carry into account.

2. Let So the result is b1 := 010021011 and 2b − 01112 := 00112 = 01002 and subtract the numbers again from right to 2. left.

For bit 0,02 −12 = −12 results in a 1 at this position and a carry to the next position.

We have to be careful with the next bit! Bit 1 of next digit as in the previous case, but we have to note a 0 instead of a 1 as the resultin 02 −12 = −12. However, we must also consider the carry. So we have a carry to theb1 is 0, bit 1 of b2 is 1. This would result

for this bit.

−For bit 02 −12 we must again consider the carry from the previous position. The result is 2 = 02 and this time we have no carry over to the next digit. 12

Bit 3 is 0 for both numbers, so since we have no carry, the result for bit 3 is02 −02 =02.

When subtracting, please note that the above procedure only works if the first numberSo the result is 01002 − 00112 = 00012.

(the minuend) is greater than the second number (the subtrahend). If this is not the case, the subtraction must be made by adding the two’s complement of these numbers.

Next we will look at multiplication. Again, there are four calculation rules for this:The written multiplication for binary numbers works exactly as you know it from decimal00112222 ·0·1·0·12222=0=0=0=12222

numbers: You write both numbers with a ℤ linked next to each other. Then start with the last digit of the second factor and multiply it by the first factor. The result is written rightjustified under the second factor. Now multiply the penultimate digit of the second factor by the first factor and write the result under the first result, but shifted one bit to the left. The last digit is filled with a zero. This procedure is continued iteratively until all digits of the second factor have been multiplied once by the first factor. Each intermediate result is one place further to the left than the result above it. Finally, the individual results, which now stand under each other, are summed up. This gives the overall result of the multiplication.

Let us illustrate this procedure with some examples:

1. leave some space underneath for the result.Let b1 := 10102 and b2 := 10112. We write these linked by a ⋅ next to each other and 10102 ⋅ 10112

first number in sequence. The following applies: Now we first multiply the last digit of the second number (the 1) by all digits of theand . The first intermediate result is therefore12 ⋅ 02 =02,12 ⋅ 12 =12,12 ⋅ 02 =02

Now we repeat this procedure with the penultimate digit of the second number, which is also a 1. We now write the result one place shifted to the left under the first intermediate result, whereby we replace the last place with a 0 fill.

10102 ⋅ 10112

10102

101002

The next digit of the second number is 0. Accordingly, the entire interim result is 0, which we write — shifted one more digit to the left — below the previous result. We fill the last two digits with 0.

0000002

The last digit of the second number is a 1, so 10102 is the intermediate result. Again, we write this under the previous result, moving one digit further to the left and filling the last three digits with 0.

0000002

It results in:

Again, we add the individual results and get by to add a 0 as the last digit to the positive binary number.102 =210. In such a case, the calculation is extremely simple because you only have1101 ⋅ 0010 = 11010 . This example illustrates a nice special case, namely the multiplication of a positive binary number2 2 2

#### Binary-coded decimal

You already know how to convert a binary number to the decimal system to represent its value as a decimal number. Let us now examine the opposite direction and consider how we can convert decimal numbers into the binary system. Let us first look at the simpler case of non-negative numbers. The procedure works iteratively and can be described as Let d ∈ ℕ be the decimal number to be converted and let b be the binary number to be follows: Step 1: Set Step 2: Determine the power of two Step 3: Set the bit with index b := 0. i b to 1 (note that indexing starts at 0, i.e., the index of the2iwith i ∈ℕ0 so that 2i ≤ is d and 2i+1 > d. determined.

in

lowest bit is 0 and not 1).Step 4: Set d := d − 2i.

Step 5: If d > 0, repeat the procedure from step 2 with changed d. If d = 0, b is the binary number searched for and the procedure is terminated.

Let us illustrate this algorithm with some examples:

Let d := 14b.

Step 1: We set

1110Step 5: It‘s Step 2: It’s Step 5: It’s Step 3: We set the bit with index 2 to 1, so Step 2: It’s Step 2: It’s Step 3: We set the bit with index 3 to 1, so Step 4: We set Step 5: It’s 2d = 6 > 0d = 0 = 2 ≤ 2 < 4 = 2 = 4 ≤ 6 < 8 = 2d := 2 −2d := 6 −2d := 14 −2,, so we repeat the procedure from step 2. so we repeat the procedure from step 2. = 2 − 2 = 0 = 6 − 4 = 23 = 14 − 8 = 6. So . So i = 1i = 2. ..b := 1110b := 1100b := 1000. 2.. 14 Step 4: We set 22 3 2.

d = 2 > 0 2

Let us briefy verify that we have calculated correctly! We see thatStep 4: We set Step 3: We set the bit with index 1 to 1, so 2. 1 , so the procedure ends and the binary representation of 1 2 . 2 10 is

11102=1·2=8+4+2=14 3 +1·22 +1·21 +0·20

14d := 1234 = 1110 10d b .

so 10 2.

Step 2: It’s Step 1: We set Let Step 2: It’s Step 3: We set the bit with index 10 to 1, so 22 10b := 0d := 1234 −2. We convert . = 1234 − 1024 = 210, so b := 0100 0000 0000i = 7. .. . Step 4: We set 102. d = 210 > 0 = 128 ≤ 210 < 256 = 210

Step 3: We set the bit with index 7 to 1, so Step 5: It’s 7 so we repeat the procedure from step 2.8

Step 4: We set

Step 2: It’s Step 4: We set Step 3: We set the bit with index 6 to 1, so Step 2: It’s Step 5: It’s Step 5: It’s 26 d := 82 −2, so we repeat the procedure from step 2. = 82 − 64 = 18, so , so i = 4i = 6b := 0100 1101 0000b := 0100 1100 0000. .. . 2. d = 18 > 0 = 16 ≤ 18 < 32 = 2d := 18 −26 = 18 − 16 = 24 to 1, so

Step 4: We set Step 3: We set the bit with index 24 4 5 2.

Step 5: It’s ,

Step 2: It’s 1 = 0. . 2. so 1234 = 0100 1101 0010=1024+128+64+16+2=1234+0·210 +1·2. +0·2

2 1 0

Finally, let us look at how we can convert decimal numbers to the two’s complement. It is10 2 important to note that if the numbers are too large, an overflow can occur. This is a wellknown problem in computer science, which occurs whenever you want to store a number complement can only assume values between −128 . 127. So if we now try to store a in a data container (e.g., a variable), but the value of this number is outside the limits of

value > 127 or < −128 in such an 8-bit number, this will obviously lead to problems, the data container’s value range. We have seen above that a binary number in 8-bit two’s

because the uppermost bits of such a number can simply not be represented by the availrange. If you want to convert a number −32768 . 32767 x ∈ ℤ with −128x ≤ x ≤ < −128127 into the two’s com-x > 127 able number of digits. This falsifies the result. Therefore, you should first take a brief look at the number to be converted and make sure that it actually fits into the available value

plement, you can choose the 8-bit two’s complement. If is or , the 16-bit two’s complement would be used instead. This allows numbers in the value range

to be represented. For even larger or smaller numbers, you can choose

corresponding two’s complements with even more digits.

Let d ∈ ℤ be the decimal number to be converted and let d b be the binary number to be The algorithm for converting decimal numbers to the two’s complement is as follows:

determined in two’s complement. The choice of the two’s complement (8-bit, 16-bit, etc.) depends on the size of . described above. Save the result in b. d ≥ 0 b Step 1: Ignore the sign and transfer the number into the binary system using the algorithm

Step 2: If d is < 0, continue with step 3. If , already contains the binary number in

Step 3: Invert all bits of b by replacing all zeros with ones and vice versa. two’s complement representation and the algorithm is terminated.

Step 4: Add the value 110, that is 0000 ... 00012, to b. The result is the searched binary number in two’s complement representation.

The algorithm therefore makes a case distinction: Positive numbers can be converted as described above and otherwise do not need to be changed further. If it is a negative number, we must also negate the binary digits and add the value 110.

Let us illustrate this algorithm with some examples:

a)Let ment.d := 14Step 1: We transfer the number with the above algorithm into the binary system.10. Because −128 ≤ d ≤ is 127, we can choose the 8-bit two’s comple-

ment.represent 1110We have already seen in the example above that 2, please note the leading zeros! We have added these because we want tobd ≥ 0 as a number in 8-bit two’s complement, i.e., as a binary number with. Because , so we are done and into the binary system without taking the sign into account−128 ≤ d ≤ 127b = 0000 11101410 = 11102. So b := 0000 b) Step 2: It is eight digits.ing for in 8-bit two’s complement representation. 2 is the number we are look-

b)d)Let d := −27Step 4: We add the value 0000 0001Step 1: We transfer and obtain 10b := 0001 1011d = 27 and get b := 1110 0100, we can choose the 8-bit two’s comple-b := 1110 0101.

Step 2: It is d < 0, so we continue with step 3.b2 102. and get

c) Step 3: We invert all bits in Let’s make sure that the result is correct: It’s 2 2. 1110 01012=1·==−−128+64+32+4+127−27 +1·26 +1·25 +0·24 +0·23 +1·22 +0·21 +1·20

Let . Because 10 is , the 8-bit two’s complement is no longer b)c)d) 1001Step 2: It is Step 3: We invert all bits in Step 4: We add the value We want to make sure again that the result is correct:2. d < 0b := 0000 0001 0111 01110000 0000 0000 0001b and get b := 1111 1110 1000 10002 and get b := 1111 1110 10002. sufficient. So we have to choose the 16-bit two’s complement for representation.

into the binary system without taking the sign into accountand obtain .

1111 1110 1000 10012=1·=+0·2−−+8+137532768+16384+8192+4096+2048+1024+512+128−2815+1·2+1·27 +0·214 +1·26 +0·213 +1·25 +0·212 +1·24 +1·2113+1·2+0·2102+1·2+0·291 +1·20

=