representing negative binary numbers in a computer system

Here I will explain two schemas used to represent negative binary numbers in a computer system.

1. sign magnitude

   In sign magnitude, a number's positive or negative value is represented by the left most bit. 0 represents the positive value of the number, and 1 represents the negative value of the number. The rest of the digits represent the number itself, this is called the magnitude. There are problems with sign magnitude however. There are two representations of zero, 10000000 and 00000000. Also, it can be awkward while performing addition, it isn't as simple and easy as adding and carrying values.

2. Complementary representations

   complementary representations are used to get the negative notion of an integer in binary. To get two's complement of an integer, you first do 1's complement, which is to write out the number in binary and invert the digits, and then add 1 to the result.

what is meant by 9's complement and 10's complement using two 5-digit worked examples

9's complement is a complementary representation used to find the subtraction of the decimal numbers. It is obtained by counting the digits of the decimal number, and then subtracting each digit of the number by nine, or in other words, the largest number within the same number of digits as the decimal number, by the absolute value of the decimal number. The representation is equal to the original number from 0 up to 49, but any representation from 50 to 99 is equal to -49 up to -0, with 50=-49 and 99=-0.
For example, 5 digit 9's complement of -16638
99999-16638 = 83361
10's complement is calculated by subtracting what would be the largest number within the amount of digits of the decimal number, + 1. It is the same as 9's complement plus 1. Anything represented by 50 to 99 is equal to the original number of -50 to -1, with 50=-50 and 99=-1, and everything represented from 0 to 49 is the same in its original number form. There is only 1 representation of 0, unlike 9's complement.

For example, 4 digit 10’s complement of -28590
(99999 + 1) – 28590 = 71410

what is meant by 1’s complement and 2’s complement using two 5-digit worked examples

1’s complement of a binary number is used to represent a negative binary number by inverting all the bits of the original binary number, i.e, changing every 1 into 0 and every 0 into 1. In the 1’s complement format, positive numbers remain unchanged.
For example, + 7 will be represented as =
00000111
and -7 will be represented as
11111000,
Which is the 1’s complement of 00000111 and was obtained by inverting each digit.
In 2’s complement of a binary number, the most significant bit is used as a sign, 0 representing a positive sign and 1 representing a negative sign. For 2’s complement, firstly we figure out 1’s complement. Then we look starting from the least significant bit for a 0. Then all 1’s are changed into 0 until a 0 is found. And finally, the found 0 is changed into a 1. If all digits are 1’s from the 1’s complement of the number , for example, 1’s complement of 000 is 111, then an extra 1 is added onto the end of the number after all 1’s are changed into 0’s, in this case 2’s complement of 000 is 1000.

the differences between overflow and carry with examples of each using 6-bit numbers

The difference between overflow and carry is;
Overflow occurs when the result of a calculation doesn’t fit into the amount of bits that are available to represent the signed number and compromise the sign bit, this can give the wrong value of a number in arithmetic.
For example,
011111 is equal to 31 in decimal
010011 is equal to 19 in decimal

011111 + 010011 = 110010
31 + 19 is not equal to -18
Carry occurs when adding 2 binary numbers together and the amount we can possibly represent with 1 bit is exceeded, so the excess value is carried into the next column. A carry occurs when 1 + 1 bits are added, the result is 0 and 1 is carried over. If it occurs that 1 + 1 with an additional 1 carried over, the answer is 1 with 1 carried over.
For example,
011010
+ 100111
1111 - carry's
110001

what is meant by exponential notation

exponential notation is a method of simplifying the writing of very large or very small numbers. It simplifys the number by removing excess zeros by shortening the number by moving the decimal point, and displaying the shortened non-zero numbers (known as the mantissa) multiplied by a base , usually 10, to an exponent. Its crucial to know the base of the exponent, and the exact location of the decimal point (or binary point if base 2 is involved) to complete the representation. We must also acknowledge the sign of the number and the exponent, and the magnitude of the exponent.
For example; 0.000002567
One possible representation of this number is:

0.2567×10^{5}.

how Excess N Notation works

excess N notation is used to represent both negative and positive numbers so that the order of the bit pattern stays the same. The steps of how this method works are as follows; firstly, a number, which we'll call n, is picked somewhere near the middle of all the values of the exponent. N is then given a value of zero, an everything that comes before n is negative and everything over n is positive.

the steps taken to represent a decimal fraction as a 32-bit floating point number

to explain how to convert a decimal fraction I will use 6.18 as an example.
Firstly, convert the numbers before and after the decimal point to binary separately.
6 = 110
0.18=
To convert a decimal fraction to binary, it is done by simply multiplying fraction by 2 repeatedly and recording the number before the decimal point each time, and every time the output is 1, 1 is removed and reverted to zero, and this multiplication of 2 continues until the fraction is even at 0 or until further multiplication is not necessary. In this example, the mantissa of this 32-bit floating point number is 10 bits long. The first digit received is entered in as the msb and the last digit is the lsb.
0.18 x 2= 0.36
0.36 x 2 =0.72
0.72 x 2= 1.44
0.44 x 2=0.88
0.88 x 2=1.76
0.76 x 2= 1.52
0.52 x 2=1.04
0.04 x 2=0.08
0.08 x 2=0.16
0.16 x 2=0.32….

0.18 = 0.0010111000
Add the two numbers together,
110.0010111000
Move the decimal point so as that there is only 1 digit to the left of it,
1.100010111000
The exponent of this number is 11111101 as,
6/2 =3 with 0 remainder
3/2 =1 with 1 remainder
½=0 with 0 remainder
010= 2, 2-127 = -125 = 11111101
6.18 is positive, therefore, the sign is 0.
The full number is:
0 11111101 10000101110000000000000

References:

https://youtu.be/RuKkePyo9zk

https://allthedifferences.com/carry-flag-vs-overflow-flag/

http://www.yorku.ca/pkashiya/binary/signed-numbers/excess-notation.html

https://www.sciencedirect.com/topics/engineering/floating-point-number#:~:text=The%20exponent%20needs%20to%20represent,127%20%3D%20134%20%3D%20100001102.

https://www.javatpoint.com/9s-and-10s-complement-in-digital-electronics