Data Types

|  |  |
| --- | --- |
| **Sign and Magnitude**  Example  00000011 = 3  10000011 = -3 | One way to represent negative numbers is to make the leftmost bit, called the **most significant bit,** a sign bit.  **•** Ifthe most significant bit is zero, the number is positive  **•** if the most significant bit is one, the number is negative. |
| **Two’s Complement**  28 expression in Two’s Complement  First write out 28 in binary  00011100  Then we invert in the digits  11100011  Then add 1  **11100100 = -28 in binary two’s complement** | A much better way of representing numbers in binary is called **two's complement.**  To get the two's complement negative notation of an integer, you write out the number in binary. You then invert the digits, and add one to the result. |
| **Binary Addition**  00010110 - 22  00101110 - 46  01000100 - 68 | 1. 0+0 =0  2. 0 + 1 = 1  3. 1 + 0 = 1  4. 1 + 1 = 0 Carry 1 (This is 2 in denary or 10 in binary.)  5. 1 + 1 + 1 = 1 Carry 1 (This is 3 in denary or 11 in binary.) |
| **Binary Subtraction**  Example denary 17-14 would be:   |  |  |  | | --- | --- | --- | | 14 | = | 00001110 | | -14 | = | 11110010 | | 17 | = | 00010001 | | 17 + (-14) | = | (1) 00000011 |   The carry on the addition is ignored, and the correct answer is given. | Binary subtraction is best done by using the negative two's complement number and then adding the second number. |
| **Hexadecimal**   |  |  |  |  | | --- | --- | --- | --- | | **4096** | **256** | **16** | **1** | |  |  | **A** | **C** |   **A - (16 \* 10) = 160**  **C – (1\*12) = 12**  **160 + 12 = 172** | Base 16 in Maths, or **hexadecimal** in Computing. We abbreviate this to **hex.**  Decimal Hex  1 1  2 2  3 3  4 4  5 5  6 6  7 7  8 8  9 9  10 A  11 B  12 C  13 D  14 E  15 F |
| **Fixed Point Binary**    In the binary example above, the left hand section before the point is equal to 5 (4+ 1) and the right hand section is equal to 1/2 + 1/4 3/4), or 0.5 + 0.25 = 0.75. So, using four bits after the point, 0101 1100 is 5.75 in denary. | Fixed point binary numbers can be a useful way to represent fractions in binary. A binary point is used to separate the whole place values from the fractional part on the number line:  The **range** of a fixed point binary number is also limited by the fractional part. For example, if you have only 8 bits to store a number to 2 binary places, you would need 2 digits after the point, leaving only 6 bits before it. |
| **Floating point binary numbers – Positive Exponents**  In all the examples below, eight bits are used for the mantissa and four bits for the exponent. The implied binary point is to the right of the sign bit.    To convert the floating point binary number above to denary:  • Write down the mantissa, 0.1011010  • Translate the exponent from binary to denary 0011 = 3, This means that you have to move the point 3 places to the right, as the mantissa has to be multiplied by 2³  • The binary number is therefore 101.1010 | The leftmost bit of both the mantissa and the exponent is a sign bit, with 0 indicating a positive number, and 1 a negative number  To convert the floating point binary number above to denary:  • Write down the mantissa  • Translate the exponent from binary to denary.   * Move the point to where the exponent suggests   • Translate this to binary using fixed point scale |
| **Floating point binary numbers – Negative Exponents**    • Find the two's complement of the exponent. (Remember that to convert a positive to negative binary number using two's complement you must flip the bits and add 1.) Exponent = -2  • Move the binary point of the mantissa two places to the left, to make it smaller, The mantissa is therefore 0.001 (You can ignore the trailing zeros)  • Translate this to denary. The answer is 0.125 | If the exponent is negative, the decimal point must be moved left instead of right. |
| **Floating point binary numbers – Negative Mantissa**    • Find the twos complement of the mantissa. It is 0.1010011, so the bits represent -0.1010011  • Translate the exponent to denary, 01 01 = 5  • Move the binary point 5 places to the right to make it larger. The mantissa is -10100.11  • Translate this to binary with the help of the below. The answer is -20.75. | A negative floating point number will have a 1 as the sign bit or MSB (Most Significant Bit) of the mantissa indicating a negative place value. |
| **Normalisation floating point numbers**  Normalise the binary number 0.0001011 0101, held in an 8-bit mantissa and a 4-bit exponent.  • The binary point needs to move 3 places to the right so that there is a 1 following the binary point.  • Making the mantissa larger means we must compensate by making the exponent smaller, so subtract 3 from the exponent, resulting in an exponent of 0010.  • The normalised number is 0.1 011000 0010 | Normalisation is the process of moving the binary point of a floating point number to provide the maximum level of precision for a given number of bits. This is achieved by ensuring that the first digit after the binary point is a significant digit.  **A positive number has a sign bit of 0 and the next digit is always 1.**  **A negative number has a sign bit of 1 and the next digit is always 0.** |
| **Examples of normalisation**  Normalise the following number, using an 8-bit mantissa and a 4-bit exponent:  0.0000110 0001  0.0000110 0001 = 0.0000110 0001 x 21  = 0.1100000 x 21-4  Exponent = -3.  3 = 0011. Take 2s complement, or work it out as -8 for the sign bit, -3 = 1101  Normalised binary number is 0.1100000 1101 | **Examples of normalisation**  Normalise the following number, using an 8-bit mantissa and a 4-bit exponent:  1.11100110011  1.1110011 0011 = 1.1110011 x 23  = 1.00110000 x 23-3  Exponent = 0  Normalised binary number is 1.0011000 0000 |
| **Converting from denary to normalised binary floating point**  Convert the number 14.25 to normalised floating point binary, using an 8-bit mantissa and a 4-bit exponent.  • In fixed point binary, 14.25 = 01110.010  • Remember that the first digit after the sign bit must be 1 in normalised form, so move the binary point 4 places left and increase the exponent from 0 to 4. The number is equivalent to 0.111 0010 x 2⁴  • Using a 4-bit exponent, 14.25 = 0 11100100100 | To convert a denary number to normalised binary floating point, first convert the number to fixed point binary.  Remember that the first digit after the sign bit must be significant to be in normalised form, |
| **Converting from denary to normalised binary floating point**  If the denary number is negative, calculate the two's complement of the fixed point binary:  e.g. Calculate the binary equivalent of -14,25  14.25 = 01110.010  -14,25 = 10001.110 (two's complement)  In normalised form, the first digit after the point must be 0, so the point needs to be moved four places left,  10001.11 0 = 1.0001110 x 2⁴ = 10001110 0100 | If the denary number is negative, calculate the two's complement of the fixed point binary: |
| **Floating Point Arithmetic – Addition**  Convert the denary numbers 0.25 and 10.5 to normalised floating point binary form using an 8-bit mantissa and a 4-bit exponent. Add together the two normalised binary numbers, giving the result in normalised floating point binary form.  **Step 1**  The numbers in normalised form are  0.25  0.1000000 1111  10.5  0.1010100 0100  **Step 2**  Write the mantissas with a binary point , and convert the exponents to denary, giving  0.1000000 exponent -1 and  0.1010100 exponent 4  **Step 3**: Make both exponents 4 and shift the binary points accordingly  0.0000010  0.1010100  **Step 4**: Add the numbers, giving 0.1010110 exponent 4 0100 (In this case it's already normalised) | The rules for addition and subtraction can be stated as:   * line up the points by making the exponents equal * add or subtract the mantissas * normalise the result |
| **Floating Point Arithmetic – Subtraction**  **0.1000100 0110 minus 0.1000010 0101**  **Step 1:** Convert the exponents to denary. giving  0.1000100 exponent 6 and  0.1000010 exponent 5  **Step 2**: Make both exponents 6 and shift the binary point of the second number accordingly  0.1000100 exp 6  0.0100001 exp 6 (make the number smaller as you increase the exponent)  **Step 3**: Find the twos complement of the second number  1.1011110 +1 = 1.1011111  **Step 4:** Add the numbers  0.1000100  1.101111 1  (1)0.0100011 exp 6 (ignore the carry)  Now normalise the number by moving the binary point right 1 place, which increases the number, and decrease the exponent by 1  Result is: 0.1000110 0101 | The rules for addition and subtraction can be stated as:   * line up the points by making the exponents equal * Find the twos complement of the second number * add the numbers * normalise the result |