Numbers:

This worksheet will cover all of the number types you need to know for 107, there are answers at the end of the worksheet.

**Signed magnitude:**

* Left most bit is a signed bit where 0 = + and 1 = -
* 1000 and 0000 both = 0, however 1000 = -0
* Largest number is 011…1
* Smallest number is 111…1

In signed magnitude the leftmost bit becomes a signed bit where it doesn’t hold any value except for the sign of the number where 0 = + and 1 = -. This means that the leftmost bit has no actual value.

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Binary | Visualisation | signed magnitude | Unsigned |
| 0000 | + 000 | 0 | 0 |
| 1001 | - 001 | -1 | 9 |
| 0111 | + 111 | 7 | 7 |
| 1111 | - 111 | -7 | 15 |

**Ones complement:**

* Rather than a signed bit the most significant bit I more like a NOT gate (!number)
* 1111 and 0000 = 0 (-/+ respectfully)
* 011…1 is the biggest number
* 100…0 is the smallest number
* The left most bit = - (value of all other bits). In an 8 bit string this means that:

1000 0000 = -127 as 0111 1111 = 127

This means that 1111 in ones complement = 0

|  |  |  |  |
| --- | --- | --- | --- |
| Binary | Visualisation | Ones complement | Unsigned |
| 0000 | 0 + 000 (0) | 0 | 0 |
| 1000 | -7 + 000 (0) | -7 | 8 |
| 1111 | -7 + 111 (4 + 2 + 1) | -0 | 15 |
| 0101 | 0 + 101 (4 + 1) | 5 | 5 |

**Twos complement:**

* Similar to ones complement but has a wider range of numbers
* Smallest number is 100…0
* Largest number is 011…1
* 0 = 000…0
* The leftmost bit is the negative of that bits value, so in an 8 bit string the leftmost bit is = -128

|  |  |  |  |
| --- | --- | --- | --- |
| Binary | Visualisation | Twos complement | Unsigned |
| 0000 | 0 | 0 | 0 |
| 1000 | -8 | -8 | 8 |
| 1011 | -8 + 11 (2+1) | -5 | 11 |
| 0111 | -0 + 111(4+2+1) | 7 | 7 |

**Excess representation**:

* Shown as Excess-K where K is a real number (int)
* 0000 = -K, so if Excess-16 then 0000 = -16
* 0 in excess is when the binary string = K
* You can easily work it out by finding the unsigned value of the string then do value - K

|  |  |  |  |
| --- | --- | --- | --- |
| Binary | Visualisation | Excess-8 | Unsigned |
| 0000 | -8 + 0 | -8 | 0 |
| 1000 | -8 + 8 | 0 | 8 |
| 1011 | -8 + 11 | 3 | 11 |
| 0111 | -8 + 7 | -1 | 7 |

**Floating point numbers**

* Similar to significant figures in science (1.2\*2^12)
* Contains a mantissa (1.2 in the above example) that is 23 bits long
* Has an exponent that Is 8 bits long
* A sign bit to show +/- where 0 = + and 1 = -

How it is structured:

|  |  |  |
| --- | --- | --- |
| Sign bit | Exponent | Mantissia |
| 0 10000110 11110011100100101011011 | | |

How to convert to IEEE 754:

Start with a number such as: 147.625

In binary this number is = 10010011.101

Normalise the number. This means that you move the decimal place until only one 1 is left of the point.

Now our number is 1.0010011101

We moived the decimal place 7 places to the left to get here, so now we need an exponent of 7 to allow us to move the point back to where it should be.

7 in binary is 00000111, however the exponent is stored using excess-127. This means that we need to store 7 + 127, which is 134.

In binary this is 10000110.

147.625 is positive so the sign is 0.

Now we have the sign bit (0), the exponent (1000110) and the mantissia (1.0010011101)

Making our final answer: 0 1000110 10010011101

|  |  |
| --- | --- |
| Floating point number | Decimal |
| 01000011101000111110000000000000 |  |
| 01000010110001000100000000000000 |  |
| 11000011111001110001000000000000 |  |

|  |  |
| --- | --- |
| Decimal | Floating point |
| -74.5 |  |
| 235.75 |  |
| -658.0625 |  |

Convert into decimal as if it was stored in the different number systems:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Binary | Unsigned | Signed | Ones | Twos | Excess-128 |
| 01110101 |  |  |  |  |  |
| 10100100 |  |  |  |  |  |
| 10101000 |  |  |  |  |  |
| 11101010 |  |  |  |  |  |
| 01010010 |  |  |  |  |  |
| 11101111 |  |  |  |  |  |
| 10100101 |  |  |  |  |  |
| 10010010 |  |  |  |  |  |

Convert into the different types of numbers (8 bits, excess-128):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decimal | Signed | ones | twos | Excess-128 |
| 24 |  |  |  |  |
| -89 |  |  |  |  |
| -54 |  |  |  |  |
| 86 |  |  |  |  |
| -21 |  |  |  |  |
| 78 |  |  |  |  |
| 57 |  |  |  |  |
| -29 |  |  |  |  |

Answers:

|  |  |
| --- | --- |
| Floating point number | Decimal |
| 01000011101000111110000000000000 | 327.75 |
| 01000010110001000100000000000000 | 98.125 |
| 11000011111001110001000000000000 | -462.125 |

|  |  |
| --- | --- |
| Decimal | Floating point |
| -74.5 | 11000010100101010000000000000000 |
| 235.75 | 01000011011010111100000000000000 |
| -658.0625 | 11000100001001001000010000000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Binary | Unsigned | Signed | Ones | Twos | Excess-128 |
| 01110101 | 117 | 117 | 117 | 117 | -10 |
| 10100100 | 164 | -92 | -91 | -92 | 36 |
| 10101000 | 168 | -88 | -87 | -88 | -88 |
| 11101010 | 234 | -22 | -21 | -22 | -22 |
| 01010010 | 82 | 82 | 82 | 82 | -46 |
| 11101111 | 239 | -17 | -16 | -17 | -17 |
| 10100101 | 165 | -91 | -90 | -91 | -91 |
| 10010010 | 146 | -110 | -109 | -110 | -110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decimal | Signed | ones | twos | Excess-128 |
| 24 | 00011000 | 00011000 | 00011000 | 10011000 |
| -89 | 11011001 | 10100110 | 10100111 | 00011111 |
| -54 | 10110110 | 11001001 | 11001000 | 01001010 |
| 86 | 01010110 | 01010110 | 01010110 | 11010110 |
| -21 | 10010101 | 11101010 | 11101011 | 01101011 |
| 78 | 01001110 | 01001110 | 01001110 | 11001110 |
| 57 | 00111001 | 00111001 | 00111001 | 10111001 |
| -29 | 10011101 | 11100010 | 11100011 | 01101011 |