Floating Point Binary Exercise 2

**Example 1** - How would 0.0001001 be represented in binary floating point representation using 16 bits for the mantissa (including the sign bit) and 8 bits for the exponent?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fixed Point** | **Floating Point** | **Sign (1 bit)** | **Mantissa (15 bit)** | **Exponent (8 bit)** |
| 0.0001001 | 0.1001 x 2-11 | 0 | 100100000000000 | 11111101 |

For the floating point representation, the decimal point has moved 3 places to the **right**. However, as this needs to be represented in binary, you need to convert 3 into:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

(we can ignore the six 0s at the start of this number and just use 11)

To indicate that the decimal point moves to the right, you need to use -11.

The sign bit indicates whether the number is positive or negative. If the number is positive, then the sign bit must be 0. If the number is negative, then the sign bit must be 1. In this case the sign bit is 0 because the number is positive (0.0001001).

The full number after the decimal point is used as the mantissa. This is 1001, which uses 4 bits in total. However, as we are required to use 15 bits for the mantissa, you need to add **eleven** 0s at the **end** of the number. This is 100100000000000.

Similarly with the exponent, we already know that we are moving 3 places. However, as we are moving to the right and not to the left, this has to be represented as -3, which means we need to use two’s complement to represent this. This gives us:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **-128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

**Example 2** – How would 0.0000110111 be represented in binary floating-point representation using 16 bits for the mantissa (including the sign bit) and 8 bits for the exponent?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fixed Point** | **Floating Point** | **Sign (1 bit)** | **Mantissa (15 bit)** | **Exponent (8 bit)** |
| 0.0000110111 | 0.110111 x 2-100 | 0 | 110111000000000 | 11111100 |

For the floating point representation, the decimal point has moved 4 places to the right. However, as this needs to be represented in binary, you need to convert 4 into:

|  |  |  |
| --- | --- | --- |
| **4** | **2** | **1** |
| 1 | 0 | 0 |

(this becomes -100 as we are moving to the **right**)

Again, the sign bit is 0 because the number is positive.

The full number after the decimal point is used as the mantissa. This is 110111, which uses 6 bits in total. However, as we are required to use 15 bits for the mantissa, you need to add **nine** 0s at the end of the number. This is 110111000000000.

Similarly with the exponent, we already know that we are moving 4 places. However, as we are moving to the right and not to the left, this has to be represented as -4, which means we need to use two’s complement to represent this. This gives us:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **-128** | **64** | **32** | **16** | **8** | **4** | **2** | **1** |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

**EXERCISE**

Complete the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fixed Point** | **Floating Point** | **Sign (1 bit)** | **Mantissa (15 bit)** | **Exponent (8 bit)** |
| 0.000010001 |  |  |  |  |
| 0.000001111 |  |  |  |  |
| 0.000000101 |  |  |  |  |
| 0.000000011 |  |  |  |  |
| 0.0000000011 |  |  |  |  |