Nom: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nom de Plume: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Homework 2 Part 3: Two’s Complement Worksheet**

We’ve been only using positive numbers up until now… because negative numbers are a little weird in binary. Rather than using a bit to hold the sign (“1’s Complement”), which would result in a plus and minus zero, a waste of one number we could otherwise use, we use a system called “Two’s Complement” instead. It works like this: the Most Significant Bit (the MSB, i.e., the leftmost bit, i.e. the bit that represents the biggest value) represents the negative of that value instead. And that’s it: the math just works when you add positive and negative numbers together, as long as you don’t overflow.

So In 8-bit two’s complement numbers, the MSB will mean -128 instead of 128. Fill in the blanks:

|  |  |  |
| --- | --- | --- |
| **Question #** | **Binary** | **Value** |
| 00000001 | 00000000 |  |
| 00000010 | 00000001 |  |
| 00000011 | 10000000 |  |
| 00000100 | 10000001 |  |
| 00000101 | 10000010 |  |
| 00000110 | 10000011 |  |
| 00000111 | 10000100 |  |
| **Question #** | **Binary** | **Value** |
| 00001000 | 11111111 |  |
| 00001001 | 11111110 |  |
| 00001010 | 11111101 |  |
| 00001011 | 11111100 |  |
| 00001100 |  | -127 |
| 00001101 |  | -1 |
| 00001110 |  | -5 |

There’s a trick for multiplying by -1 in 2’s Complement: write down the original number in binary, flip all the bits, add one and discard any carry. So to figure out what -3 should look like in 2’s Complement binary, we write down the binary for positive 3 (00000011), flip all the bits (11111100), and then add one (11111101). This also works for negative to positive. Check this result against the table above.

|  |  |  |
| --- | --- | --- |
| **Question #** | **Positive #** | **\* -1 in Binary** |
| 00000001 | 0 |  |
| 00000010 | 1 |  |
| 00000011 | 2 |  |
| 00000100 | 3 |  |
| 00000101 | 127 |  |
| 00000110 | 128 |  |
| 00000111 | 5 |  |
| **Question #** | **Negative #** | **\* -1 in Binary** |
| 00001000 | 11111111 |  |
| 00001001 | 11111110 |  |
| 00001010 | 11111101 |  |
| 00001011 | 11111100 |  |
| 00001100 | -127 |  |
| 00001101 | -1 |  |
| 00001110 | -5 |  |

Now you are ready for the final reveal: when adding 2’s compliment numbers together, all the math just works out beautifully, without any special rules. Adding a positive and a negative works exactly as when doing unsigned binary math. Add the bits, do a carry, and that’s it. The only danger is the danger we always have, which is overflowing or underflowing a binary number. 127+1 = -128, and -128 – 1 = 127.

|  |  |  |
| --- | --- | --- |
| **Question #** | **Math** | **Result in Binary** |
| 00000001 | 0 + 1 |  |
| 00000010 | 1 – 2 |  |
| 00000011 | 2 + 3 |  |
| 00000100 | 2 – 3 |  |
| 00000101 | 127 - 1 |  |
| 00000110 | 127 + 1 |  |
| 00000111 | 5 – 16 |  |