

Examples Assignment #3

Sign-representation negative values in Two's complement:

In your input file, you must place the number in SIGN-MAGNITUDE.

Example (using 8 bits):

If you want to add the value 1 to the value -2 in sign-magnitude:

Your input file must have the values:

1 → (expected binary in sign-magnitude) 00000001

-2 → (expected binary in sign-magnitude) 10000010

Since your compiler uses two's complement, the values will be represented as:

1 → (actual binary in two's complement) 00000001

-2 → (actual binary in two's complement) 11111110

Now you will transform the binary representation to sign-magnitude:

00000001 → 00000001 (this will yield a two's complement value of: 1)

11111110 → 10000010 (this will yield a two's complement value of: -126)

Print the values (to show that the transformation has been successful)

The values printed would be 1 and -126

After printing, you must convert back the values to two's complement

00000001 → 00000001 (this will yield a two's complement value of: 1)

10000010 → 11111110 (this will yield a two's complement value of: -2)

Now you will perform the addition:

Result: $1 + -2 = -1$

Print the final result

NOTE: This example uses 8bit numbers; you're to use 32bit numbers.

Struct to Array casting:

To know what the values in the array were going to be, we have to calculate the binary representation of each value that is going into the struct. Once we have all the binary representations, we pack them in 16 bits (2 byte) segments to find the value of each element of the array.

Input (255, 32.455, 0, 1.1, 1, 12)

255 (8bits - char) → 11111111

32.455 (64bits - double) → 0100000001000000001110100011110101110000101000111101011100001010

0 (24bits - int) → 00000000000000000000000000000000

1.1 (64bits - double) → 0011111111110001100110011001100110011001100110011001100110011010

1 (16bits - short) → 0000000000000001

12 (8bits - char) → 00001100

Now that we have all the data in binary representation, we get the bits, 16 by 16 finding the value of each of the shorts in the array (12 shorts)

```
1111111101000000 → -192
010000000111010  →  16442
0011110101110000 →  15728
1010001111010111 → -23593
0000101000000000 →  2560
0000000000000000 →  0
000000000111111  →  63
1111000110011001 → -3687
1001100110011001 → -26215
1001100110011001 → -26215
1001101000000000 → -26112
0000000100001100 →  268
```