Two's Complement Representation

Two's Complement

Two's Complement is very similar to One's Complement, **except**:

1. Eliminates two zeros
2. No multiple additions to get the correct result

Note: To eliminate the two zeros the range of negative value ends up being shifted up by one

Two's Complement

Assume a 4-bit fixed length

Decimal Number	Two's Complement Representation	Hex
7	0111	7
6	0110	6
5	0101	5
4	0100	4
3	0011	3
2	0010	2
1	0001	1
0	0000	0
-1	1111	F
-2	1110	E
-3	1101	D
-4	1100	C
-5	1011	B
-6	1010	A
-7	1001	9
-8	1000	8

Two's Complement

Positive traits from One's Complement are kept:

- Positives all start with zero, negatives all start with one
- Positive numbers are unchanged

Traits that have changed:

- Range for Two's Complement
  - $-(2^{(N-1)})$ to $2^{(N-1)} - 1$

Two's Complement - Addition

Add any positive value and its negative

Example: 5 and -5

```
1 111
  0101
+ 1011
  ----
  0000
```

Adding two value together gets zero, not negative zero

Two's Complement

- Adding a 2's complement value and its negative will give the max value + 1
  - E.g. 5 (0101) + (-5) (1011) = 10000
  - $10000 = 2^4 = 16$
  - max 4-bit value = F_{16} or 15_{10}
- $NEG_{2's\ Comp}(N)$ = (max value + 1) - N **or**
- $NEG_{2's\ Comp}(N)$ = flip each bit **then** add 1
  - **Order of the two operations is IMPORTANT!**
  - **Order does not change regardless of direction of negation**

Two's Complement - Conversion

Convert decimal to n-bit two's complement binary

1. result = abs(num) converted to binary
2. **IF** (num != -(2^n) **AND** MSBit(result) != 0) **OR** length(result) > n **THEN**
   "num cannot be represented in n bits"
3. **ELSE**
   IF num < 0 **THEN**
   result = $NEG_{2's\ Comp}$(result)

Two's Complement

Convert 115_{10} to 8-bit two's complement binary, expressed in hex

1. abs(115) = 115
2. 115_{10} converted to 8-bit binary is 01110011_2
3. Since the bit 7 is 0 (and no bit 8 or more was required) 115_{10} is valid
4. Since the number is positive nothing further is done
5. In hex => 73_{16}

Two's Complement

Convert -67_{10} to 8-bit two's complement binary, expressed in hex

1. abs(-67) = 67
2. 67_{10} converted to 8-bit binary is 01000011_2
3. Since bit 7 is 0 (and no bit 8 or more was required) -67_{10} is valid
4. Since the number is negative apply $NEG_{2\text{'s comp}}(01000011_2) = 10111101_2$
5. In hex => BD_{16}

Two's Complement - Conversion

Convert two's complement to decimal

1. **IF** MSbit(binary) == 1 **THEN**
   binary = $NEG_{2's\ Comp}$(binary)
2. result = binary converted to decimal
3. **ELSE**
   IF original binary was negative **THEN**
   result = -1 * result

Two's Complement

Convert 00011011_2 2's comp to base 10

1. The MSBit is 0 so this is a positive number

   No changes made to the number

2. 00011011_2 converted to base-10:

   `result = 16 + 8 + 2 + 1 = 27`$_{10}$

3. Since the MSBit = 0 nothing further needs to be done

   `result = 27`$_{10}$

Two's Complement

Convert 10100110_2 2's comp to base 10

1. The MSBit is 1 so this is a negative number

   $\mathtt{NEG}_{\mathtt{2's\ comp}}(\mathtt{10100110}_2) = \mathtt{01011010}_2$

2. 01011010_2 converted to base-10

   $\mathtt{result} = 64 + 16 + 8 + 2 = 90_{10}$

3. The MSBit is 1

   $\mathtt{result} = -1 * \mathtt{result} = -90_{10}$

Two's Complement

Convert the following 12 bit fixed length binary numbers that are represented in Two's Complement to base-10.

1. 000000101101_2
2. 111011100100_2

Two's Complement

Now we do not have the issue of two representations of zero. Also for a signed fixed-length number of N bits we can represent the numbers

$$-(2^{N-1}) \text{ to } 2^{N-1}-1$$

Because there is no **-0** there is an additional negative number

It could have been made positive, but then not all positive numbers would start with 1

Two's Complement

Addition:

```
                 1          1 11
3 + 2 =       0011   3 + -2 =  0011
            + 0010           + 1110
              ----             ----
              0101             0001

                               YAY?
```

Two's Complement

Pros:

- ✓ addition is identical to unsigned (no special cases)
- ✓ one zero (zero-test circuit will be simple)
- ✓ the MSB indicates the sign (negative-test circuit will be simple)
- ✓ subtraction is just addition of negation (no new circuit required)

Two's Complement

Cons:

- × negative numbers not directly identifiable
- × two numbers have no inverse, but OH WELL!
  - zero and maximum negative number
  - their inverse is themselves
    - $NEG_{2's\ Comp}(0) = 0$
    - $NEG_{2's\ Comp}(-(2^n)) = -(2^n)$

Binary Number Ranges Summary

For a fixed length of N bits we get the following number ranges based on number representation

Representation	Min Number	Max Number
Unsigned	0	2^N-1
Signed Magnitude	$-(2^{(N-1)}-1)$	$2^{(N-1)}-1$
One’s Complement	$-(2^{N-1}-1)$	$2^{N-1}-1$
Two’s Complement	$-(2^{N-1})$	$2^{N-1}-1$