CS2100

Jia Cheng

January 2021

1 Definitions

- MSB: Most significant bit
- LSB: Least significant bit

2 Conversion Table

Dec	Hex	Bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	a	1010
11	b	1011
12	c	1100
13	d	1101
14	e	1110
15	f	1111

3 Number representations

Note: The following terminology may be slightly confusing. $2s$ complement representation refers to the number representation format. $2s$ complement of a number N refers to how $-N$ is represented under the $2s$ complement representation.

Ditto for $1s$ complement representation and $1s$ complement of a number N.

3.1 Bs complement

Given a base B, positive integer n. The n-digit B's complement representation operates modulo B^n.

We now denote Bs as n-digit B's complement representation.

An interesting observation is that we can map the Bs values, i.e. the **ordered** set $\{(b_{n-1} \dots b_0) : b_i \in \{0, 1, \dots, B-1\}\}$ to **any** translation + cyclic permutation of $\mathbb{Z}/B^n\mathbb{Z}$.

We now arbitrarily fix the integer 10, so as to demonstrate the meaning of translation and cyclic permutation. An example of translation of $\mathbb{Z}/10\mathbb{Z}$ is $\{2, 3, 4, \dots, 11\}$. In this case, this is a translation of $+2$. An example of cyclic permutation of $\mathbb{Z}/10\mathbb{Z}$ is $\{2, 3, 4, \dots, 9, 0, 1\}$.

In the case of the regular 8-bit $2s$ complement representation, we have a combination of translation and cyclic permutation of $\mathbb{Z}/256\mathbb{Z}$.

Example To show the completely arbitrary nature of such a map from $\{(b_{n-1} \dots b_0) : b_i \in \{0, 1, \dots, B-1\}\}$ to any translation + cyclic permutation of $\mathbb{Z}/B^n\mathbb{Z}$, consider the map of 5-bits to $\mathbb{Z}/32\mathbb{Z}$. Call this representation R. Note that under $2s$ complement, the 5-bits would have been mapped to the ordered set $\{0, 1, \dots, 15, -16, -15, \dots, -1\}$

Then, $31 = (11111)_R$, $-31 \equiv 32 - 31(\mod 32) = 1 = (00001)_R$.
Hence, $31 - 31 = 32 + (-31) = (11111)_R + (00001)_R = (00000)_R = 0$, which is indeed the expected result.

3.2 Fractional complements

3.2.1 (B-1)'s complement

Suppose a fractional number N has n integer digits and m fractional digits, then $(B-1)$'s complement of N is given by $B^n - B^{-m} - N$.

To see why, simply remove the decimal point by multiplying N by B^m (and then dividing B^m). Hence,

$$B^{-m}(B^{n+m} - 1 - B^m N) = B^n - B^{-m} - N$$

3.3 Sign extension to the left

3.3.1 Binary case

For positive numbers, i.e. numbers with the MSB 0, in both $2s$ and $1s$ complement, it suffices to pad the "left" of the number with 0's.

It is particularly easy to mentally prove the sign extension rules for $1s$, since for a negative number x, we simply invert it to get positive $-x$, apply the positive sign extension rules (pad with 0's), then invert it back.

Regardless, it is easy to prove the validity of the sign extension by doing summations.

For **both** 1s and 2s complement representations

- Sign extending number with 0 as MSB: Pad with 0's.
- Sign extending number with 1 as MSB: Pad with 1's.
- Note that the reason why I didn't say positive/negative here is because in $1s$ complement, there is a positive and a negative 0 value. Referring to the MSB would be the most general.

3.3.2 General case

3.4 Sign extension to the right

i.e. after the decimal point

Technically this is not called sign extension. But I don't know what is the technical term for this, so I'll just call this sign extension to the right for convenience.

1s complement (fractional)

- Sign extending number with 0 as MSB: Pad with 0's.
- Sign extending number with 1 as MSB: Pad with 1's.

2s complement (fractional)

- Pad with 0's regardless of MSB

3.5 Excess representation

4 High level code to MIPS Assembly

4.1 if-elseif-else

4.2 switch-case

4.3 while

4.4 do-while

4.5 for