1.2.3 Two's Complement

- for integers

$$-X = \begin{cases} 0x_{n-2}x_{n-3}\ldots x_1x_0, & \text{if } X \geq 0 \\ (1\overline{x_{n-2}}\,\overline{x_{n-3}}\ldots\overline{x_1}\,\overline{x_0} + 1)\bmod 2^n, & \text{if } X < 0 \end{cases}$$

- for fractionals

$$-X = \begin{cases} 0x_{n-2}x_{n-3}\ldots x_1x_0, & \text{if } X \geq 0 \\ (1\overline{x_{n-2}}\,\overline{x_{n-3}}\ldots\overline{x_1}\,\overline{x_0} + 0.00\ldots01)\bmod 2, & \text{if } X < 0 \end{cases}$$

- $\underline{\bmod 2^n}$ (for ints) & $\underline{\bmod 2}$: ignore the carry out from the msb

Practical rule for conversion between $SM \leftrightarrow C_2$:
- Keep the sign bit unchanged
- starting from left towards the right, complement all bits except the rightmost bit of 1 and all the 0s that might follow it

Ex:

$$-103 = 1110\ 0111_{SM}$$

$$\begin{array}{r} 1001\ 1000_{C1} \\ +\quad 1 \\ \hline -103 = 1001\ 1001_{C2} \end{array}$$

Practical rule:

$$-103 = 1110\ 0111_{SM}$$

$$-103 = 1001\ 1001_{C2}$$

$$-68 = 1100\ 0100_{SM}$$

$$-68 = 1011\ 1100_{C2}$$

a) Range of values:
- for integers: $[-2^{n-1} : 2^{n-1}-1]$
- for fractionals: $[-1 : 1-2^{-n+1}]$

b) Precission: similar to SM's $p \cong \lceil (n-1)\log_{10} 2 \rceil$

c) HW complexity:
- addition & subtraction: simpler than SM/C1
- multiplication: somewhat more complex than SM's

d) Disadvantages of C1

(A) value of 0

$$-0 = 1\,000\ldots000_{SM} =$$
$$1\,1\,1\,1\ldots1\,1\,1_{C1}$$
$$+\quad 1$$
$$\cancel{1}\,0000\quad000_{C2}.$$

ignore it because of $\underline{\text{mod } 2^n}$

$$-0 \equiv +0$$

(B) Addition:

$X = +5$ $\quad 0101_{C2}$
$Y = +2$ $\quad 0010_{C2}$
$\quad 0111_{C2} = +7$

$X = +5$ $\quad 0101_{C2}$
$Y = -2$ $\quad 1110_{C2}$
$\cancel{1}\ 0011_{C2} = +3$

$X = -5$ $\quad 1011_{C2}$
$Y = +2$ $\quad 0010_{C2}$
$\quad 1101_{C2} = -3$
$\quad 1011_{SM} = -3$

$X = -5$ $\quad 1011_{C2}$
$Y = -2$ $\quad 1110_{C2}$
$\cancel{1}\ 1001_{C2} = -7$
$\quad 1111_{SM} = -7$

C2's arithmetic:

- correct operation regardless of operands signs
  $\Rightarrow$ implement subtraction $X - Y = X + (-Y)$
- carry out from msb is ignored.
- sign bit is treated just like any magnitude bit

Comparative code representation for integers on 5 bits.

Decimal number	Fixed-point binary codes		
	SM	C1	C2
+15	01111	01111	01111
+14	01110	01110	01110
⋮	⋮	⋮	⋮
+2	00010	00010	00010
+1	00001	00001	00001
+0	00000	00000	00000
−0	10000	11111	00000
−1	10001	11110	11111
−2	10010	11101	11110
⋮	⋮	⋮	⋮
−14	11110	10001	10010
−15	11111	10000	10001
−16			10000

C2's anomaly

- by convention, $\underbrace{100\ldots00}_{\text{<n bits>}}{}_{C2}$ $\begin{cases} -2^{n-1} & \text{: for ints.} \\ -1 & \text{: for fractionals} \end{cases}$

! for unsigned no. $100\ldots00 = +2^{n-1}$

Arithmetic overflow:

result of an arithmetic operation exceeds storing capacity

a) for unsigned numbers

Ex: X, Y – 6-bit, unsigned $X = 35$,
$Y = 33$

$$\begin{array}{rl} X = & 1\,0\,0\,0\,1\,1 \\ Y = & 1\,0\,0\,0\,0\,1 \\ \hline \cancel{1} & \underbrace{0\,0\,0\,1\,0\,0}_{= \cancel{4}\,!!!} \end{array} \; +$$

storing capacity

overflow

Had the X, Y were on 7 bits.

storing capacity

$$\begin{array}{rl} X = 35 = & 0\,1\,0\,0\,0\,1\,1 \\ Y = 33 = & 0\,1\,0\,0\,0\,0\,1 \\ \hline & \underbrace{1\,0\,0\,0\,1\,0\,0}_{= 68} \end{array} \; +$$

overflow for unsigned operands $\equiv$ carry out from msb

b) for signed numbers

X, Y on 6 bits, C2

$X = +19$

$Y = +14$

storing capacity

$$\begin{array}{rl} X = +19 = & 0\,1\,0\,0\,1\,1_{C2} \\ Y = +14 = & 0\,0\,1\,1\,1\,0_{C2} \\ \hline & 1\,0\,0\,0\,0\,1_{C2} = \cancel{-31}\,!!! \end{array} \; +$$

overflow.

Had X, Y were on 7 bits.

$$\begin{array}{rl} X = +19 = & 0\,0\,1\,0\,0\,1\,1_{C2} \\ Y = +14 = & 0\,0\,0\,1\,1\,1\,0_{C2} \\ \hline & 0\,1\,0\,0\,0\,0\,1_{C2} = +33 \end{array} \; +$$

17 − 1

16

overflow for signed operands $\equiv$
adding same sign operands produces the opposite sign result.

Q: $Z = X + Y$ $\quad X \geq 0$, $Y < 0$ NO overflow. $|Z| \leq \max\{|X|, |Y|\}$

for subtraction

1.2.4. Alternative representations of C2

– Robertson's interpretation. ⟹ multiply

Let X, negative in C2, on n bits, integer

$$X = 1\, x_{n-2}^{*}\, x_{n-3}^{*} \ldots x_1^{*}\, x_0^{*}$$

$$= (1\, x_{n-2}^{*}\, x_{n-3}^{*} \ldots x_1^{*}\, x_0^{*}) \bmod 2^n$$

$$= \left(\begin{array}{l} 1\,0\,0 \ldots 0\,0\ + \\ 0\, x_{n-2}^{*}\, x_{n-3}^{*} \ldots x_1^{*}\, x_0^{*} \end{array} \right) \bmod 2^n$$

$$= \Big(-2^{n-1} + \underbrace{0\, x_{n-2}^{*}\, x_{n-3}^{*} \ldots x_1^{*}\, x_0^{*}}_{\text{C2 positive value}}\Big) \bmod 2^n$$

$$= -2^{n-1} + 0\, x_{n-2}^{*}\, x_{n-3}^{*} \ldots x_1^{*}\, x_0^{*}$$

The value of a negative in C2 is obtained by subtracting the weight associated with the sign bit from the positive number obtained by clearing the sign bit.

Ex. $-103 = 10011001_{C2}$

$$-2^7 + 00011001_{C2} =$$

$$-128 + 25 = -103\ !$$

$$-68 = 10111100_{C2}$$

$$-2^7 + 0\,0\,\underbrace{1\,1\,1\,1}_{15}\,\underbrace{0\,0}_{*\,2^2}{}_{C2} = -128 + 60 = -68$$

Robertson's interpretation also stands for positives

$$+103 = 0\ 1100111_{C2}$$

$$-\cancel{0\cdot 2^7} + 0\ 1100111_{C2} = +103$$

In general:

for X in C2, integer $X = x_{n-1}x_{n-2}x_{n-3}\dots x_1 x_0$

$$\boxed{X = -x_{n-1}\cdot 2^{n-1} + \sum_{i=0}^{n-2} x_i\cdot 2^i}$$

The same considerations apply to fractional nos.

1.3. Representation of fixed-point decimal numbers.

$0.2_{10} \rightarrow$

Comparative decimal representation codes

Decimal digit	Fixed-point decimal codes		
	BCD 8421	Excess of 3	Two-out-of-five
0	0000	0011	11000
1	0001	0100	00011
2	0010	0101	00101
3	0011	0110	00110
4	0100	0111	01001
5	0101	1000	01010
6	0110	1001	01100
7	0111	1010	10001
8	1000	1011	10010
9	1001	1100	10100