ITI1100

Addition of Chapter 1 Notes: Complements

The notes are for ITI1100B&C Winter 2017 internal use only.

Further distribution is strictly prohibited.

One’s Complement Representation

- The one’s complement of a binary number involves **inverting all bits.**
  - 1’s comp of 00110010 is **11001101**
  - 1’s comp of 10101011 is **01010100**
- For an n bit number N the 1’s complement is $(2^n-1) - N$.
- Called diminished radix complement by Mano since 1’s complement for base (radix 2).
- To find negative of 1’s complement number take the 1’s complement.

$\underline{0}\,\underline{0001100}_2 = 12_{10}$

Sign bit Magnitude

$\underline{1}\,\underline{1110011}_2 = -12_{10}$

Sign bit Magnitude

Two's Complement Representation

- The two's complement of a binary number involves **inverting all bits and adding 1**.
  - 2's comp of 00110011 is **11001101**
  - 2's comp of 10101010 is **01010110**
- For an n bit number N the 2's complement is $(2^n-1) - N + 1$.
- Called radix complement by Mano since 2's complement for base (radix 2).
- To find negative of 2's complement number take the 2's complement.

$0\underline{0001100}_2 = 12_{10}$

Sign bit Magnitude

$1\underline{1110100}_2 = -12_{10}$

Sign bit Magnitude

Two’s Complement Shortcuts

- Algorithm 1 **– Simply complement each bit and then add 1 to the result.**
  - **Finding the 2’s complement of $(01100101)_2$ and of its 2’s complement…**

N	**= 01100101**	**[N] =**	**10011011**
	10011010		**01100100**
	+ 1		**+ 1**
	---------------		**---------------**
	10011011		**01100101**

- Algorithm 2 **– Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits.**
  - **N = 0 1 1 0 0 1 0 1**

    [N] = 1 0 0 1 1 0 1 1

Complements of Numbers

- The complement of the complement = the original number.

$$[N] = r^n - N$$

$$[r^n - N] = r^n - (r^n - N) = N$$

- For a number N with a radix point, remove the point temporarily to form the r's or (r-1)'s complement. The radix point is restored to the complemented number in the same position.

Subtraction of Unsigned Numbers with r's Complements

The subtraction of two **n-digit unsigned numbers** M-N in base r can be done as follows:

$$\boldsymbol{M + r's\ complement\ of}\ \mathbf{N}$$
$$= \mathbf{M} + (\boldsymbol{r^n} - \boldsymbol{N})$$
$$= \mathbf{M} - \mathbf{N} + \boldsymbol{r^n}$$

1. When M>=N, sum produces an end carry "1" . Discard the end carry. The left is a positive number, M-N.
2. When M<N, the end carry is "0" (or no end carry). The result is a negative number. Take the r's complement of the sum and place a negative sign in front.

Subtraction of Unsigned Numbers with (r-1)'s Complements

The subtraction of two n-digit unsigned number M-N in base r can be done by means of the (r-1)'s complement:

$$\boldsymbol{M} + (\boldsymbol{r}-\mathbf{1})'\boldsymbol{s\ complement\ of}\ \mathbf{N}$$
$$= \mathbf{M} + (\boldsymbol{r}^{\boldsymbol{n}} - \mathbf{1}) - \boldsymbol{N}$$
$$= (\mathbf{M}-\mathbf{N}-\mathbf{1}) + \boldsymbol{r}^{\boldsymbol{n}}$$

1. When M>=N, add the end carry "1" to low order bit. The result is a positive number.
2. When M<N, the end carry is "0" (or no end carry). The result is a negative number. Take the (r-1)'s complement of the sum and place a negative sign in front.

How To Represent Signed Numbers

- **Plus and minus sign used for decimal numbers: 25 (or +25), -16, etc.**
- **For computers, desirable to represent everything as *bits.***
- **Three types of signed binary number representations: signed magnitude, signed 1's complement, signed 2's complement.**
- **In each case: left-most bit indicates sign: positive (0) or negative (1).**

Consider ***signed magnitude***:

$0\underline{0000011}_2 = 3_{10}$

Sign bit Magnitude

$1\underline{0000011}_2 = -3_{10}$

Sign bit Magnitude

Signed-Complements

- Computer arithmetic uses the signed-complements to representation the binary number.
  - ➢ Signed-complement representation of +ve = sign bit "0"+ the original binary number ve.
  - ➢ Signed-complement representation of -ve :
    - = the (1's or 2's) complement of +ve (including the sign bit);
    - Or, = sign bit "1" + the (1's or 2's) complement of ve (excluding sign bit);

```
  0 0 1 1 (ve)
0 0 0 1 1 (+ve)

1 1 1 0 1 (-ve)

1 1 1 0 1 (-ve)
```

- Signed-2's complement representation is commonly used in computer arithmetic (only 1 representation for 0).
- When X is in signed complement form:
  - ➢ 1's or 2's complement of X (including the sign bit) = -X

$00000011_2 = 3_{10}$

Sign bit

$11111101_2 = -3_{10}$

Sign bit

Signed-2's Complement Addition and Subtraction

- Addition of two signed binary numbers represented in signed-2's complement form:
  - Is an addition of the two binary numbers (including sign bits). A carry out of the sign-bit is discarded.
  - The sum is in signed- 2's complement form. The sign bit decides if the result is positive or negative.

- Subtraction of two signed numbers in signed-2's complement form:

$$A - B = A + (-B)$$

$$= A + \text{2's complement of } B \text{(including the sign bit)}$$

$$= A + [B]_2$$

Signed-2's Complement Addition

- Using 2's complement numbers, adding numbers is easy.
- For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.
- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 2's comp.
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 2's comp.

Step 1: Add binary numbers
(*including the sign bit*)
Step 2: Ignore carry bit

Add

$$
\begin{array}{rc}
 & 0\ 1\ 1\ 0\ 0 \\
+ & 0\ 0\ 0\ 0\ 1 \\
\hline
0 & \boxed{0\ 1\ 1\ 0\ 1}
\end{array}
$$

Final Result

Ignore

Signed-2's Complement Subtraction

- Using 2's complement numbers, follow steps for subtraction
- For example, suppose we wish to **subtract** $+(0001)_2$ from $+(1100)_2$.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 2's comp.
  - $(-1)_{10} = -(0001)_2 = 11111_2$ in 2's comp.

Step 1: Take 2's complement of 2^{nd} operand
Step 2: Add binary numbers
(*including the sign bit*)
Step 3: Ignore carry bit

```
      0 1 1 0 0
−    [0 0 0 0 1]
---------------
2's comp

      0 1 1 0 0
Add + 1 1 1 1 1
---------------
Final Result  1 [0 1 0 1 1]

Ignore Carry
```

Signed-2's Complement Subtraction: Example #2

- Let's compute $(13)_{10} – (5)_{10}$.
  - $(13)_{10} = +(1101)_2 = (01101)_2$
  - $(-5)_{10} = -(0101)_2 = (11011)_2$
- Adding these two 5-bit codes…

```
              0 1 1 0 1
carry   +     1 1 0 1 1
        -------------
          1   0 1 0 0 0
```

- Discarding the carry bit, the sign bit is seen to be zero, indicating a correct result. Indeed,

$$(01000)_2 = +(1000)_2 = +(8)_{10}.$$

Signed-2's Complement Subtraction: Example #3

- Let's compute $(5)_{10} - (12)_{10}$.
  - $(-12)_{10} = -(1100)_2 = (10100)_2$
  - $(5)_{10} = +(0101)_2 = (00101)_2$
- Adding these two 5-bit codes…

```
        0  0  1  0  1
+       1  0  1  0  0
---------------------
        1  1  0  0  1
```

- Here, there is no carry bit and the sign bit is 1. This indicates a negative result, which is what we expect. $(11001)_2 = -(7)_{10}$.

Signed-1's Complement Addition and Subtraction

- Addition of two signed binary numbers represented in signed-1's complement form:
  - Is an addition of the two binary numbers (including sign bits). A carry out of the sign-bit is added to the low order bit.
  - The sum is in signed- 2's complement form. The sign bit decides if the result is positive or negative.

- Subtraction of two signed numbers in signed-1's complement form:

  A - B = A + (-B)

  = A + 1's complement of B(including the sign bit)

  = A+ $[B]_1$

Signed-1's Complement Addition

- Using 1's complement numbers, adding numbers is easy.
- For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.
- Let's compute $(12)_{10} + (1)_{10}$.
  - $(12)_{10} = +(1100)_2 = 01100_2$ in 1's comp.
  - $(1)_{10} = +(0001)_2 = 00001_2$ in 1's comp.

Step 1: Add binary numbers (*including the sign bit*)
Step 2: Add carry to low-order bit

```
                  0 1 1 0 0
Add        +      0 0 0 0 1
           ----------------
              0   0 1 1 0 1
Add carry     └──────────→ 0
           ----------------
Final
Result            0 1 1 0 1
```

Signed-1's Complement Subtraction

- Using 1's complement numbers, subtracting numbers is also easy.
- For example, suppose we wish to **subtract** $+(0001)_2$ from $+(1100)_2$.
- Let's compute $(12)_{10}$ - $(1)_{10}$.
  - $(12)_{10}$ = $+(1100)_2$ = 01100_2 in 1's comp.
  - $(-1)_{10}$ = $-(0001)_2$ = 11110_2 in 1's comp.

Step 1: Take 1's complement of 2^{nd} operand
Step 2: Add binary numbers
(including the sign bit)
Step 3: Add carry to low order bit

```
      0 1 1 0 0
-     0 0 0 0 1
---------------
1's comp

      0 1 1 0 0
Add + 1 1 1 1 0
---------------
    1 0 1 0 1 0
Add carry     1
---------------
Final
Result
      0 1 0 1 1
```