

Binary Numbers Again

Recall that N binary digits (N bits) can represent unsigned integers from 0 to 2^N-1 . 2^N total numbers for N bits (remember 0).

4 bits = 0 to 15

8 bits = 0 to 255

16 bits = 0 to 65535

Besides simply representation, we would like to also do arithmetic operations on numbers in binary form. Principle operations are addition and subtraction.

Binary Addition and Subtraction

The rules for binary addition are:

$$0 + 0 = 0, \text{ carry} = 0$$

$$1 + 0 = 1, \text{ carry} = 0$$

$$0 + 1 = 1, \text{ carry} = 0$$

$$1 + 1 = 0, \text{ carry} = 1$$

The rules for binary subtraction are:

$$0 - 0 = 0, \text{ borrow} = 0$$

$$1 - 0 = 1, \text{ borrow} = 0$$

$$0 - 1 = 1, \text{ borrow} = 1$$

$$1 - 1 = 0, \text{ borrow} = 0$$

Borrows, Carries from digits to *left* of current of digit.

Binary subtraction, addition works just the same as decimal addition, subtraction.

Fixed Precision

With paper and pencil, I can write a number with as many digits as I want:

1,027,830,032,034,532,002,391,030,300,209,399,302,992,092,920

- A digital system usually uses **FIXED PRECISION** for integers
- Limit the numbers to a fixed number of bits:

$(AF4500239DEFA231)_{16}$	64 bit number, 16 hex digits
$(9DEFA231)_{16}$	32 bit number, 8 hex digits
$(A231)_{16}$	16 bit number, 4 hex digits
$(31)_{16}$	8 bit number, 2 hex digits

- Special circuits called **REGISTERS** hold numbers
- **REGISTERS** cause fixed-length bit strings to be used

Fixed Precision Overflow

- Overflow occurs when
 - adding or subtracting two numbers, **AND**
 - correct result is a number that is
 - outside range of numbers for that fixed precision
- 8 bits - unsigned integers 0 to $2^8 - 1$ or 0 to 255
- 16 bits - unsigned integers 0 to $2^{16} - 1$ or 0 to 65535

EXAMPLE: 8-bit Addition

$$\begin{array}{r} 110101.01 \\ + 011110.11 \\ \hline 1\ 010100.00 \end{array}$$

Unit in Last Place
ulp = $(1/2^2)_{10}$

Signed Integer Representation

- Have ignored large sets of numbers so far
 - Sets of signed integers
 - Signed fractional numbers
 - Floating point numbers
- We will not talk about floating point representation (i.e. 9.23×10^{13}).
- We WILL talk about signed integer/fractional representation.
- The **PROBLEM** with signed integers (-45, +27, -99) is the SIGN! How do we encode the sign?
- The sign is an extra piece of information that has to be encoded in addition to the magnitude

Complement Representations

- Negative Value, Y , Represented as $R - |Y|$ Where R is a Constant
- Positive Values Represented Same Way as Unsigned Value
- Two Types of Complement Representations Most Common
 1. **radix complement** (binary – 2's complement)
 2. **diminished radix complement** (binary – 1's comp.)
- Obeys the Identity (negating a negative value):

$$\begin{aligned} R - Y &\Rightarrow R - (R - |Y|) \\ &= R - R + |Y| \\ &= |Y| \end{aligned}$$

Advantage is No Decisions Needed Based on Operand Sign Before Operations are Applied

Complement Representation Example

- X is Positive, Y is Negative, Compute $X+Y$ Using Complement Representation

$$\begin{aligned}X + (R - |Y|) &= -[-X - (R - |Y|)] \\ &= -[-R + |Y| - X] \\ &= R - (|Y| - X)\end{aligned}$$

- If $|Y| > X$, Then the Answer is $R - (|Y| - X)$
- If $X > |Y|$, Then the Answer Should be $X - |Y|$
 - But $R - (|Y| - X) = X + (R - |Y|) = R + (X - |Y|)$,
Thus R *Must be Discarded!*
- Solution is to Choose the Value of R Carefully
- *Intentional Overflow* to Discard R

Complementation of a Digit

- Select R to Simplify (or Eliminate) Correction for the $X > |Y|$ Case
- Calculation of Complement of Y , $(R - Y)$ Should be Simple and Fast
- Definition of Complement for Single Digit, x_i

$$\bar{x}_i = (\beta - 1) - x_i$$

- Definition of Complement for a Word, X

$$\bar{X} = (\bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_{-m})_\beta$$

What is Complementation Value, R

- Add Word and Complement Together:

$$\begin{array}{r} \bar{X} = (\bar{x}_{k-1} \quad \bar{x}_{k-2} \quad \dots \quad \bar{x}_{-m})_{\beta} \\ + X = (x_{k-1} \quad x_{k-2} \quad \dots \quad x_{-m})_{\beta} \\ \hline \end{array}$$

Answer to
Addition
goes here

What is Complementation Value, R

- Add Word and Complement Together:

$$\begin{array}{r} \bar{X} = (\bar{x}_{k-1} \quad \bar{x}_{k-2} \quad \dots \quad \bar{x}_{-m})_{\beta} \\ + X = (x_{k-1} \quad x_{k-2} \quad \dots \quad x_{-m})_{\beta} \\ \hline (\beta - 1)(\beta - 1) \dots (\beta - 1) \end{array}$$

Answer to
Addition

Must Add a Correction Factor

What is Complementation Value, R

- Add Word and Complement Together:

$$\begin{array}{r} \bar{X} = (\bar{x}_{k-1} \quad \bar{x}_{k-2} \quad \dots \quad \bar{x}_{-m})_{\beta} \\ + X = (x_{k-1} \quad x_{k-2} \quad \dots \quad x_{-m})_{\beta} \\ \hline (\beta-1)(\beta-1) \dots (\beta-1) \end{array}$$

$$+ulp = \underline{0 \quad 0 \quad \dots \quad 1}$$

Now Add
1 *ulp* as
Correction
Factor

Answer to
Addition

What is Complementation Value, R

- Add Word and Complement Together:

$$\begin{array}{r} \bar{X} = (\bar{x}_{k-1} \quad \bar{x}_{k-2} \quad \dots \quad \bar{x}_{-m})_{\beta} \\ + X = (x_{k-1} \quad x_{k-2} \quad \dots \quad x_{-m})_{\beta} \\ \hline (\beta-1)(\beta-1) \dots (\beta-1) \end{array}$$

$$\begin{array}{r} +ulp = \underline{0 \quad 0 \quad \dots \quad 1} \\ 1 \quad 0 \quad 0 \quad \dots \quad 0 \end{array}$$

Now Add
1 *ulp*

Answer to
Addition

- Therefore, we see that:

$$X + \bar{X} + ulp = \beta^k$$

$$\beta^k - X = \bar{X} + ulp$$

Complementation Value, R

- Add Word and Complement Together:

$$\begin{array}{r}
 \bar{X} = (\bar{x}_{k-1} \quad \bar{x}_{k-2} \quad \dots \quad \bar{x}_{-m})_{\beta} \\
 + X = (x_{k-1} \quad x_{k-2} \quad \dots \quad x_{-m})_{\beta} \\
 \hline
 (\beta-1)(\beta-1) \dots (\beta-1) \\
 \hline
 +ulp = \frac{0 \quad 0 \quad \dots \quad 1}{1 \quad 0 \quad 0 \quad \dots \quad 0}
 \end{array}$$

Now Add
1 ulp

Maybe this is
a good choice
for R

- Therefore, we see that:

$$X + \bar{X} + ulp = \beta^k$$

$$\beta^k - X = \bar{X} + ulp$$

Let this be $-X$

Answer to
Addition

Radix Complement Form

- The Radix Complement Form is Defined When:

$$R = \beta^k$$

$$R - X = \beta^k - X = \bar{X} + ulp$$

- Using β^k is Convenient Since Storing Result in Fixed Precision n Causes MSD of 1 to be Discarded
- Easy to Compute the Complement of X by:
 1. Take the Complement of X
 2. Add 1 ulp to Complement

$$X + \bar{X} + ulp = \beta^k$$

$$\beta^k - X = \bar{X} + ulp$$

Radix Complement Form (cont)

- No Correction is Needed When We have Positive X and Negative Y Such That:

$$X + (R - |Y|) > 0$$

- Since $R = \beta^k$

$$\begin{aligned} X + (R - |Y|) &= R + (X - |Y|) \\ &= \beta^k + X - |Y| \end{aligned}$$

- And β^k is discarded Automatically Due to Finite Precision!!!!

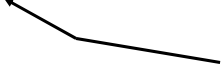
Radix Complement Arithmetic

$$\begin{array}{ccc} n = 5 & & \beta = 2 \\ X = 13_{10} & Y = -8_{10} & X + Y = ? \end{array}$$

Radix Complement; In this case 2's Complement

$$\begin{array}{r} 01101 \quad (+13_{10}) \\ 11000 \quad (-8_{10}) \\ \hline 100101 \quad (+5_{10}) \end{array}$$

Carry-out
Does NOT Mean
Overflow



Twos Complement Representation

- Twos complement – Radix Complement way to represent signed integers
- To encode a negative number:
 - get the binary representation of its magnitude
 - COMPLEMENT each bit
 - then ADD 1 (1 ulp)

Two's-Complement Representation

EXAMPLE

- What is $(-5)_{10}$ in Twos Complement, 8-bit Finite Precision?
- The magnitude 5 in 8-bits is $(00000101)_2 = (05)_{16}$
- Now complement each bit: $(11111010)_2 = (FA)_{16}$
- Now add one (1 ulp): $(FA)_{16} + 1 = (FB)_{16}$
- $(FB)_{16}$ is the 8-bit, two's-complement representation of $(-5)_{10}$.

NOTE: positive numbers in 2's-complement are simply their binary representation.

Two's-Complement Examples

8-bit Wordsize

$$\begin{aligned}(-5)_{10} &= (?)_2 = (?)_{16} \\ (+5)_{10} &= (?)_2 = (?)_{16} \\ (+127)_{10} &= (?)_2 = (?)_{16} \\ (-127)_{10} &= (?)_2 = (?)_{16} \\ (-128)_{10} &= (?)_2 = (?)_{16} \\ (+0)_{10} &= (?)_2 = (?)_{16} \\ (-0)_{10} &= (?)_2 = (?)_{16}\end{aligned}$$

- For 8 bits, can represent the signed integers -128 to +127.
- For N bits, can represent the signed integers
 $-2^{(N-1)}$ to $+2^{(N-1)} - 1$
- Note that negative range extends one more than positive range.
- MSB is the Sign Bit – Always Indicates Sign of Magnitude

Two's-Complement Examples

$$\begin{aligned}(-5)_{10} &= (11111011)_2 = (\text{FB})_{16} \\ (+5)_{10} &= (00000101)_2 = (\text{05})_{16} \\ (+127)_{10} &= (01111111)_2 = (\text{7F})_{16} \\ (-127)_{10} &= (10000001)_2 = (\text{81})_{16} \\ (-128)_{10} &= (10000000)_2 = (\text{80})_{16}\end{aligned}$$

(note the extended range!)

$$\begin{aligned}(+0)_{10} &= (00000000)_2 = (\text{00})_{16} \\ (-0)_{10} &= (00000000)_2 = (\text{00})_{16} \quad (\text{only 1 zero!!!})\end{aligned}$$

- For 8 bits, can represent the signed integers -128 to +127.
- For N bits, can represent the signed integers
 $-2^{(N-1)}$ to $+2^{(N-1)} - 1$
- Note that negative range extends one more than positive range.
- MSB is the Sign Bit – Always Indicates Sign of Magnitude

2's Complement Overflow

- If X, Y have **opposite signs** overflow never occurs whether carry-out exists or not

$\begin{array}{r} 00101 \quad (+ 5_{10}) \\ 10110 \quad (- 10_{10}) \\ \hline 11011 \quad (- 5_{10}) \end{array}$	$\begin{array}{r} 01010 \quad (+ 10_{10}) \\ 11011 \quad (- 5_{10}) \\ \hline 1\ 00101 \quad (+ 5_{10}) \end{array}$
<p>No Carry-out</p>	<p>Carry-out</p>

- If X, Y have **same sign** and result sign differs, overflow occurs

$\begin{array}{r} 11001 \quad (- 7_{10}) \\ 10110 \quad (- 10_{10}) \\ \hline 1\ 01111 \quad (+ 15_{10}) \end{array}$	$\begin{array}{r} 00111 \quad (+ 7_{10}) \\ 01010 \quad (+ 10_{10}) \\ \hline 10001 \quad (- 15_{10}) \end{array}$
<p>Carry-out, Overflow</p>	<p>No Carry-out, Overflow</p>

Radix Complement Example

$$\beta = 5 \quad n = 4$$

- Consider:

$$X = (3422)_5$$

$$Y = (4432)_5$$

- $X+Y$
- $Y-X$ (normal subtraction)
- $Y-X$ (use Radix-complement: $-X \rightarrow R=X+1 \text{ ulp}$)
- $Y-X$ (use Radix-complement definition: $R=\beta^k-|X|$)

Diminished Radix Complement

$$\beta = 2 \quad n = 4$$

- In **Diminished Radix Complement**, the Complementation Process is Easier Since the Addition of 1 *ulp* is Avoided

$$R = \beta^k - ulp$$

$$R - X = \beta^k - ulp - |X| = \bar{X}$$

- Range of Positive Numbers is: $[(0000)_2, (0111)_2] = [(0)_{10}, (7)_{10}]$
- 1's Complement of Largest is $(1000)_2 = (-7)_{10}$
- 1's Complement of Zero is $(1111)_2$
- Two Representations of Zero!

$$-7 \leq X \leq +7$$

- In All Cases MSB is Sign Bit

One's-Complement Representation

- One's-complement – Diminished Radix way to represent signed integers.
- To encode a negative number,
 - get the binary representation of it's magnitude,
 - COMPLEMENT each bit.
 - Complementing each bit mean that 1s are replaced with 0s, 0s are replaced with 1s.

One's-Complement Representation

EXAMPLE

- What is $(-5)_{10}$ in Ones Complement, 8 bits?
- The magnitude $(5)_{10}$ in 8-bits is $(00000101)_2 = (05)_{16}$
- Now complement each bit: $(11111010)_2 = (\text{FA})_{16}$
- $(\text{FA})_{16}$ is the 8-bit, ones complement number of -5

NOTE: positive numbers in 1's complement are simply their binary representation.

1's Complement Error

- One's Complement Can Be Better Since Easier to Form Complement
- Carry-out in One's Complement Indicates a Correction is Needed

$$X > 0; \quad Y < 0$$

$$-|Y| \Rightarrow R - |Y| \quad \text{Definition}$$

$$R - |Y| = \beta^k - ulp - |Y| = \bar{Y}$$

$$X + \bar{Y} = X + 2^n - ulp - |Y|$$

$$= (X - |Y|) + (2^n - ulp)$$

- if $X > |Y|$, then answer should be $X - |Y|$ however register contains $X - |Y| - ulp$ since 2^n is carry-out bit, therefore must correct by adding 1 *ulp* to answer

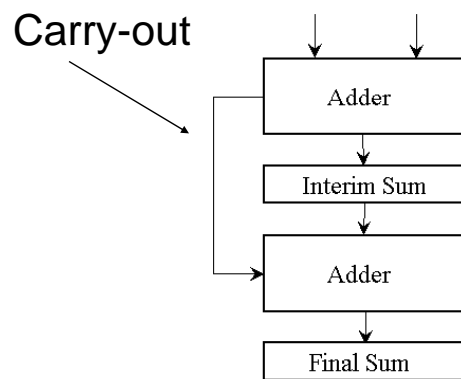
Example of 1's Complement Overflow

Need
Correction
Since
Overflow

$$\begin{array}{r} 01010 \quad (+ 10_{10}) \\ 11010 \quad (- 5_{10}) \\ \hline 100100 \quad (+ 5_{10}) \\ + \quad \quad \quad 1 \quad (+ \text{ulp}) \\ \hline 00101 \quad (+ 5_{10}) \end{array}$$

So-called
"end-around"
carry

"End-around" Carry Design



- This is "end-around" carry
– always add carry-out to LSD

Ones Complement Examples

$$\begin{aligned}(-5)_{10} &= (?)_2 = (?)_{16} \\ (+5)_{10} &= (?)_2 = (?)_{16} \\ (+127)_{10} &= (?)_2 = (?)_{16} \\ (-127)_{10} &= (?)_2 = (?)_{16} \\ (+0)_{10} &= (?)_2 = (?)_{16} \\ (-0)_{10} &= (?)_2 = (?)_{16}\end{aligned}$$

- For 8 bits, can represent the signed integers
-127 to +127.
- For N bits, can represent the signed integers
 $-2^{(N-1)} - 1$ to $+2^{(N-1)} - 1$

Ones Complement Examples

$$\begin{aligned}(-5)_{10} &= (11111010)_2 = (\text{FA})_{16} \\ (+5)_{10} &= (00000101)_2 = (\text{05})_{16} \\ (+127)_{10} &= (01111111)_2 = (\text{7F})_{16} \\ (-127)_{10} &= (10000000)_2 = (\text{80})_{16} \\ (+0)_{10} &= (00000000)_2 = (\text{00})_{16} \\ (-0)_{10} &= (11111111)_2 = (\text{FF})_{16}\end{aligned}$$

- For 8 bits, can represent the signed integers
-127 to +127.
- For N bits, can represent the signed integers
 $-2^{(N-1)} - 1$ to $+2^{(N-1)} - 1$

One's-Complement Comments

- Have the problem that there are two ways of representing 0 (-0, and +0)
- Mathematically - no such thing as two representations for zeros
- Addition of $K + (-K)$ does yield (one of the) Zeros

$$(-5)_{10} + (5)_{10} = (\text{FA})_{16} + (05)_{16} = (\text{FF})_{16} = (-0)_{16}$$

- $K + 0 = K$ only works +0 used
- Does not work if use -0.

$$5 + (+0) = (05)_{16} + (00)_{16} = (05)_{16} = 5 \text{ (ok)}$$

$$5 + (-0) = (05)_{16} + (\text{FF})_{16} = (04)_{16} = 4 \text{ !!! (wrong)}$$

Signed Magnitude Representation

- Signed Magnitude (SM) is a method for encoding signed integers.
- Most Significant Bit is used to represent the sign.
- '1' is used for a '-' (negative sign), a '0' for a '+' (positive sign).

- Format of a SM number in 8 bits is:

$$(smmmmmmm)_2$$

- where 's' is the sign bit and the other 7 bits represent the magnitude.

NOTE: for positive numbers, the result is the same as the unsigned binary representation

Signed Number Arithmetic

Signed Magnitude – Only Use Magnitude Digits

$$\begin{array}{r} 0 \ 1011 \ (+11_{10}) \\ 0 \ 0110 \ (+6_{10}) \\ \hline 0 \ 1 \ 0001 \ (+1_{10}) \end{array}$$

Carry-out
→ Overflow

Signed Magnitude Comments

- Two Representations for zero, +0 and -0
- Addition of +K and -K is not zero

EXAMPLE

$$\begin{array}{r} 10001010.00_2 \\ +00001010.00_2 \\ \hline 10010100.00_2 \end{array}$$

$-10_{10} + 10_{10}$ Yields a Sum of -20_{10} !!!!

Signed Magnitude Examples (8 bits)

$$\begin{aligned}(-5)_{10} &= (?)_2 = (?)_{16} \\ (+5)_{10} &= (?)_2 = (?)_{16} \\ (+127)_{10} &= (?)_2 = (?)_{16} \\ (-127)_{10} &= (?)_2 = (?)_{16} \\ (+0)_{10} &= (?)_2 = (?)_{16} \\ (-0)_{10} &= (?)_2 = (?)_{16}\end{aligned}$$

For 8 bits, can represent the signed integers -127 to +127

For N bits, can represent the signed integers

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

Signed Magnitude Examples (8 bits)

$$\begin{aligned}(-5)_{10} &= (1\ 0000101)_2 = (85)_{16} \\ (+5)_{10} &= (0\ 0000101)_2 = (05)_{16} \\ (+127)_{10} &= (0\ 1111111)_2 = (7F)_{16} \\ (-127)_{10} &= (1\ 1111111)_2 = (FF)_{16} \\ (+0)_{10} &= (0\ 0000000)_2 = (00)_{16} \\ (-0)_{10} &= (1\ 0000000)_2 = (80)_{16}\end{aligned}$$

For 8 bits, can represent the signed integers -127 to +127

For N bits, can represent the signed integers

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

Signed Magnitude Comments

- Signed magnitude easy to understand and encode.
It is used today in some applications.
- One problem is that it has two ways of representing 0 (-0, and +0) . Mathematically speaking, no such thing as two representations for zeros.
- Another problem is that addition of $K + (-K)$ does not give Zero!

$$(-5)_{10} + (5)_{10} = (85)_{16} + (05)_{16} = (8A)_{16} = (-10)_{10}$$

- Have to consider the sign when doing arithmetic for signed magnitude representation.

Two's-Complement Comments

- Twos complement is the method of choice for representing signed integers when arithmetic will be performed.
- It has none of the drawbacks of Signed Magnitude or Ones Complement.
- There is only one zero, and $K + (-K) = 0$

$$(-5)_{10} + (5)_{10} = (FB)_{16} + (05)_{16} = (00)_{16} = (0)_{10}$$

- Normal binary addition is used for adding numbers that represent twos complement integers.

Common Student Comments

- Given a hex number, how do I know if it is in 2's complement or 1's complement? Is it already in 2's complement or do I have put it in 2's complement, etc. ?
- Is this a 4 in Hexadecimal ?
- A Hex or binary number BY ITSELF can represent ANYTHING (unsigned number, signed number, character code, colored beads, etc). You MUST HAVE additional information that tells you what the encoding of the bits mean.
- You MUST KNOW the word size for Signed Numbers