Number Systems and Conversions

Number Systems and Conversion

We are used to working with decimal numbers (base 10). But we must be able to convert between decimal numbers and binary numbers (base 2). We must also be accustomed to converting from other bases (base 16, base 8, etc.).

Thus, this will be our starting point and will continue with binary arithmetic and negative number representation.

Number Systems and Conversion

Positional notation is used when we use decimal (base 10) numbers. Each digit in the number is multiplied by a power of 10 (depending on its position).

$$953.78_{10} = 9 \times 10^{2} + 5 \times 10^{1} + 3 \times 10^{0} + 7 \times 10^{-1} + 8 \times 10^{-2}$$

Conversely, positional notation is used again when we use binary (base 2) numbers. Only now, each digit in the number is multiplied by a power of 2 (depending on its position).

$$1011.11_{2} = 1 \times 2^{3} + 0 \times 2^{2} + 1 \times 2^{1} + 1 \times 2^{0} + 1 \times 2^{-1} + 1 \times 2^{-2}$$

Number Systems and Conversion

* Binary: Base 2
* Decimal: Base 10
* Hexadecimal: Base 16

* It takes 4 binary digits (bits) to represent the numbers 0-15
* Each group of 4 binary digits corresponds to exactly one hex digit

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Decimal to Binary Conversion

$2 \,|\, 53$

$2 \,|\, 26$ rem. = 1 = a_0

$2 \,|\, 13$ rem. = 0 = a_1

$2 \,|\, 6$ rem. = 1 = a_2

$2 \,|\, 3$ rem. = 0 = a_3

$2 \,|\, 1$ rem. = 1 = a_4

0 rem. = 1 = a_5

Least significant digit

$53_{10} = 110101_2$

Most significant digit

Decimal to Binary Conversion

What we just did was convert an *integer* from decimal (base 10) to binary (base 2). What about *fraction* conversion?

$$\begin{array}{rl} F = & .625 \\ \times & 2 \\ \hline & 1.250 \end{array} \qquad \begin{array}{rl} F_1 = & .250 \\ \times & 2 \\ \hline & 0.500 \end{array} \qquad \begin{array}{rl} F_2 = & .500 \\ \times & 2 \\ \hline & 1\boxed{.000} \end{array}$$

$(a_{-1} = 1)$ $(a_{-2} = 0)$ $(a_{-3} = 1)$

terminate

$.625_{10} = .101_2$

Most significant digit

Least significant digit

Decimal to Binary Conversion

$$
\begin{array}{rr}
 & .7 \\
 & \underline{\quad 2} \\
(1) & .4 \\
 & \underline{\quad 2} \\
(0) & .8 \\
 & \underline{\quad 2} \\
(1) & .6 \\
 & \underline{\quad 2} \\
(1) & .2 \\
 & \underline{\quad 2} \\
(0) & .4 \\
 & \underline{\quad 2} \\
(0) & .8
\end{array}
$$

What if there is no termination in the calculation?

$0.7_{10} = 0.1\ \underline{0110}\ \underline{0110}\ \underline{0110}\ \ldots_2$

A repeating fraction has occurred.

process starts repeating here since .4 was previously obtained above

Base 4 to Base 7 Conversion

1. You should convert from base 4 to base 10
2. Then convert from base 10 to base 7

$231.3_4 = 2 \times 16 + 3 \times 4 + 1 + 3/4 = 45.75_{10}$

$$
\begin{array}{r|l}
7 & \underline{45} \\
7 & \underline{\ 6} \quad \text{rem. } 3 \\
 & \ 0 \quad \text{rem. } 6
\end{array}
$$

$$
\begin{array}{rr}
 & .75 \\
 & \underline{\quad 7} \\
(5) & .25 \\
 & \underline{\quad 7} \\
(1) & .75 \\
 & \underline{\quad 7} \\
(5) & .25 \\
 & \underline{\quad 7} \\
(1) & .75
\end{array}
$$

$45.75_{10} = 63.5151\ldots_7$

Binary to Hexadecimal Conversion

Recall the format of the hexadecimal number system (slide 4)

padded zeros

start from here

$$1001101.010111_2 = \underbrace{\boxed{0}100}_{4}\ \underbrace{1101}_{D}.\underbrace{0101}_{5}\ \underbrace{11\boxed{00}}_{C}{}_2 = 4D.5C_{16}$$

1. The bits are divided into groups of four
2. Then their numerical value is replaced by a hexadecimal digit

Addition Table for Binary Numbers

The addition table for binary numbers is:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ and carry the 1 to the next column

Carrying 1 to a column is equivalent to *adding* 1 to that column.

Binary Addition

$$\begin{array}{rl} 1111 & \leftarrow \text{carries} \\ 1101 & (13_{10}) \\ +\ \underline{1011} & \underline{(11_{10})} \\ 11000 & (24_{10}) \end{array}$$

A quick check.

$$\begin{array}{r|l} 2 & 24 \\ 2 & 12 \quad \text{rem.} = 0 = a_0 \\ 2 & 6 \quad \text{rem.} = 0 = a_1 \\ 2 & 3 \quad \text{rem.} = 0 = a_2 \\ 2 & 1 \quad \text{rem.} = 1 = a_3 \\ & 0 \quad \text{rem.} = 1 = a_4 \end{array}$$

Subtraction Table for Binary Numbers

The subtraction table for binary numbers is:

$0 - 0 = 0$
$0 - 1 = 1$ and borrow 1 from the next column
$1 - 0 = 1$
$1 - 1 = 0$

Borrowing 1 from a column is equivalent to *subtracting* 1 from that column.

Binary Subtraction

(a) 1 ← borrow from 3rd column

$$\begin{array}{r} 11101 \\ -\ \underline{10011} \\ 1010 \end{array}$$

(b) 1111 ← cascade borrowing

$$\begin{array}{r} 10000 \\ -\ \underline{11} \\ 1101 \end{array}$$

(c) 111 ← cascade borrowing

$$\begin{array}{r} 111001 \\ -\ \underline{1011} \\ 101110 \end{array}$$

Multiplication Table for Binary Numbers

The multiplication table for binary numbers is:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Multiplication in Binary

$$
\begin{array}{r}
1101\ (13_{10}) \\
\times\ \underline{1011\ (11_{10})} \\
1101\phantom{\ (13_{10})} \\
1101\phantom{0\ (13_{10})} \\
0000\phantom{00\ (13_{10})} \\
\underline{1101}\phantom{\ (13_{10})} \\
10001111\phantom{\ (13_{10})}
\end{array}
$$

Negative Numbers

* What about negative numbers?
  - We need to represent numbers less than zero as well as zero or higher
  - In **n** bits, we get 2^n combinations (half are positive and half are negative)

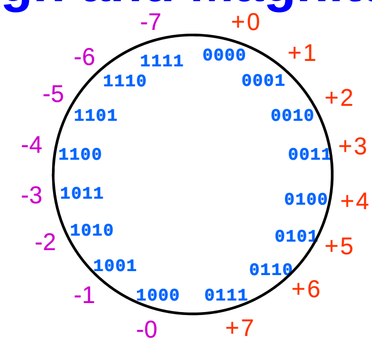
First method: use an extra bit for the sign

$$
\begin{array}{l}
\boxed{\begin{array}{c}0\\1\end{array}}\ \begin{array}{l}101 \longrightarrow +5\\ \underbrace{101}\longrightarrow -5\end{array}
\end{array}
$$

Used for sign bit.
0 is positive 1 is negative

3 remaining bits used for magnitude

Sign and Magnitude

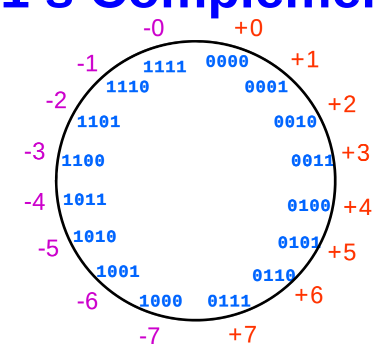


We have 7 negative numbers and a *negative* 0

We have 7 positive numbers and a *positive* 0

* High order bit is sign: 0 = positive (or zero), 1 = negative
* Three low order bits represent the magnitude: 0 (000) through 7 (111)
* Number range for **n** bits = +/- 2^{n-1} –1
* But we have two different representations for 0!

1's Complement

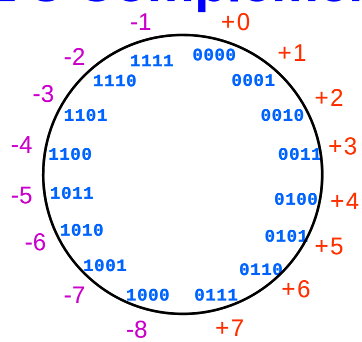


We have 7 negative numbers and a *negative* 0

We have 7 positive numbers and a *positive* 0

* High order bit is sign: 0 = positive (or zero), 1 = negative
* We complement all the bits of a positive number to arrive at a negative number
* Number range for **n** bits = +/- 2^{n-1} –1
* We *still* have two different representations for 0!

2's Complement



We have 8 negative numbers

We have 7 positive numbers and a *positive* 0

* High order bit is sign: 0 = positive (or zero), 1 = negative
* Number range for **n** bits = $+(2^{n-1}-1)$ to $-(2^{n-1})$
* Now we have only <u>one</u> representation for 0

How was this derived?

2's Complement

1. Complement the entire number
2. Then add one (1)

Example 1: 0110 = 6 $\xrightarrow{\text{complement}}$ 1001

$$
\begin{array}{r}
1001 \\
+\quad\ 1 \\
\hline
1010
\end{array}
$$

add one

1010 = –6

Example 2: 01000100 = 68 $\xrightarrow{\text{complement}}$ 10111011

$$
\begin{array}{r}
10111011 \\
+\qquad\quad 1 \\
\hline
10111100
\end{array}
$$

add one

10111100 = –68

2's Complement

A quicker way

1. Begin at the least significant bit and look to the left for the first occurrence of a one (1)
2. Copy the first one and everything to its right as is
3. Complement the remaining bits (to the left)

Example 1: 0110 = 6 $\xrightarrow{\text{copy}}$ 10 $\xrightarrow{\text{Complement remaining bits}}$ 1010 = –6

First one

Example 2: 01000100 = 68 $\xrightarrow{\text{copy}}$ 100 $\xrightarrow{\text{Complement remaining bits}}$ 10111100 = –68

First one

(Same answers as in previous slide)

Signed Binary Integers

+N	Positive Integers (all systems)	-N	Sign and Magnitude	2's Complement N*	1's Complement $\overline{N}$
+0	0000	-0	1000	——	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	——	1000	——

Binary Division

Quotient

1101

1011 | 10010001

- 1011

1110

- 1011

1101

- 1011

10

Remainder

A quick check

1101

x 1011

1101

1101

0000

1101

10001111

+ 10

10010001

2’s Complement Addition

Example 1: Addition of 2 positive numbers, sum $< 2^{n-1}$

+3 0011

+4 0100

+7 0111 (correct answer)

Example 2: Addition of 2 positive numbers, sum $\geq 2^{n-1}$

+5 0101

+6 0110

+11 1011 (wrong answer! The answer +11 produces an *overflow*. We need 5 bits total to include the sign bit)

2's Complement Addition

Example 3: Addition of positive and negative numbers (negative number has greater magnitude)

```
 +5   0101
 -6   1010
 -1   1111  (correct answer)
```

Example 4: Same as above but positive number has greater magnitude.

```
 -5      1011
 +6      0110
 +1   (1)0001  (correct answer when carry from sign bit is
               ignored—not to be confused with an
               overflow!)
```


2's Complement Addition

Example 5: Addition of two negative numbers, $|sum| \leq 2^{n-1}$)

```
 -3      1101
 -4      1100
 -7   (1)1001  (correct answer when last carry is ignored.
               This also is not an overflow.)
```

Example 6: Addition of two negative numbers, $|sum| > 2^{n-1}$)

```
 -5      1011
 -6      1010
 -11  (1)0101  (wrong answer! The answer -11 produces
               an overflow.  We need 5 bits total to
               include the sign bit)
```

1's Complement Addition

Example 3: Addition of positive and negative numbers (negative number has greater magnitude)

+5 0101
-6 1001
-1 1110 (correct answer)

Example 4: Same as above but positive number has greater magnitude.

-5 1010
+6 0110
+1 (1)0000
1 (end-around carry)
0001 (correct answer, *no overflow*)

1's Complement Addition

Example 5: Addition of two negative numbers, $|sum| \leq 2^{n-1}$)

-3 1100
-4 1011
-7 (1)0111
1 (end-around carry)
1000 (correct answer, *no overflow*)

Example 6: Addition of two negative numbers, $|sum| > 2^{n-1}$)

-5 1010
-6 1001
-11 (1)0011
1 (end-around carry)
0100 (wrong answer because of overflow)

1's Complement Addition

Example 1: Add -11 and -20 in 1's complement.

+11 = 00001011 +20 = 00010100

Taking the bit-by-bit complement,

-11 is represented by 11110100 and -20 by 11101011

11110100	(-11)	
+ 11101011	+ (-20)	
(1) 11011111	-31	
+ 1		(End-around carry)
11100000	= -31	

2's Complement Addition

Example 2: Add -8 and +19 in 2's complement.

+8 = 00001000 complementing all bits to the left of the first 1, -8 = 11111000

11111000	(-8)
+ 00010011	+ (+19)
~~(1)~~ 00001011	+11

(discard last carry)