Data representation and number systems (A)

Decimal - used in everyday's life, base 10

Binary - used by digital systems, base 2

Octal - " , base 8

Hexadecimal - " , base 16

Binary-Coded Decimal (BCD) - used to represent decimal digits by binary codes

Positional number system

base (or radix) r and r digits $0, \cdots, r-1$

Decimal: $r = 10$; $0, 1, \cdots, 8, 9$

Binary: $r = 2$; $0, 1$

Octal: $r = 8$; $0, 1, \cdots, 6, 7$

Hexadecimal: $r = 16$; $0, 1, \cdots, 9, A, B, C, D, E$

$(N)_r = b_{n-1} b_{n-2} \cdots b_1 b_0$, $\quad 0 \leq b_i \leq r-1$

b_i are the digits of N

n is the length of N representation in base r

b_{n-1} is called most significant digit

b_0 is called least significant digit

In binary representation a single digit is called bit, 4-bit string is called nibble, 8-bit string is called byte, 16-bit is a word, 32-bit double word, etc.

Numerical Value

Obtained by weighted sum from the digit string

$$\text{Numerical Value of } N = \sum_{i=0}^{n-1} b_i r^i =$$

$$b_{n-1} r^{n-1} + b_{n-2} r^{n-2} + \cdots + b_1 r^1 + b_0 r^0$$

Fractions

We use radix point convention

$(N)_r \doteq (b_{n-1} b_{n-2} \cdots b_1 b_0 \bullet b_{-1} b_{-2} \cdots b_{-m})_r$

$$(N)_r = b_{n-1} r^{n-1} + \cdots + b_1 r + b_0 + b_{-1} r^{-1} + \cdots + b_{-m} r^{-m} =$$

$$= \sum_{i=0}^{n-1} b_i r^i + \sum_{i=1}^{m} b_{-i} r^{-i} = \sum_{i=-m}^{n-1} b_i r^i$$

b_n – most significant digit

b_{-m} – least significant digit

Base r to base 10 conversion – Numerical Value

$$(101.1101)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 +$$

$$1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$

$$4 + 1 + 0{,}5 + 0.25 + 0.0625 = (5.8125)_{10}$$

$$(32AF.C4)_{16} = 3 \cdot 16^3 + 2 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 +$$

$$12 \cdot 16^{-1} + 4 \cdot 16^{-2} =$$

$$12288 + 512 + 160 + 15 + 0.75 + 0.015625 =$$

$$(12975.765625)_{10}$$

Base r to base t conversion

Repeated division of the integral part

Repeated multiplication of the fractional part

$(15247)_{10} = (?)_{16}$

15247/16 = 957 + reminder 15 = $(F)_{16}$

957/16 = 59 + reminder 8 = $(8)_{16}$

59/16 = 3 + reminder 11 = $(B)_{16}$

3/16 = 0 + reminder 3 = $(3)_{16}$

$(15247)_{10} = (3B8F)_{16}$

$(12A7C)_{16} = (?)_8$

We first convert to decimal (numerical value)

$(12A7C)_{16} = (76412)_{10}$

76412/8 = 9551 + reminder 4 = $(4)_8$

9551/8 = 1193 + reminder 7 = $(7)_8$

1193/8 = 149 + reminder 1 = $(1)_8$

149/8 = 18 + reminder 5 = (5)8

18/8 = 2 + reminder 2 = (2)8

2/8 = 0 + reminder 2 = (2)8

$(12A7C)_{16} = (76412)_{10} = (225174)_8$

$(0.761)_{10} = (?)_2$

0.761 × 2 = 1.522 integral part = 1

(1.522 − 1) × 2 = 1.044 integral part = 1

(1.044 − 1) × 2 = 0.088 integral part = 0

0.088 × 2 = 0.176 ←

0.176 × 2 = 0.352 ←

0.352 × 2 = 0.704 ←

0.704 × 2 = 1.408 integral part = 1

(1.408 − 1) × 2 = 0.816 integral part = 0

⋮

$$(0.761)_{10} = (0.11000010\ldots)_2$$

Representation can be infinite. Finite representation results an error or finite precision

$(0.761)_{10} = (0.11)_2$ with an error ~ 1.4%

$(0.761)_{10} = (0.1100001011)_2$ with an error ~ 0.03%

Binary, Octal and Hexadecimal conversions

It is very convenient to use a binary intermediate representation

$8 = 2^3$, hence all digits $0, \ldots, 7$ can be represented as $b_2 b_1 b_0$, $b_i \in \{0,1\}$. Similarly, $16 = 2^4$ hence all digits $0, \ldots, F$ can be represented as $b_3 b_2 b_1 b_0$, $b_i \in \{0,1\}$.

$(1101000101.01011)_2 = (?)_8$

$\underbrace{1}_{1}\underbrace{101}_{5}\underbrace{000}_{0}\underbrace{101}_{5}.\underbrace{010}_{2}\underbrace{11}_{6} = (1505.26)_8$

$(4736.25)_8 = (?)_2$

4	7	3	6	.	2	5
100	111	011	110	.	010	101

$(4736.25)_8 = (100111011110.010101)_2$

$(12AF.CB)_{16} = (?)_2$

1	2	A	F	.	C	B
0001	0010	1010	1111	.	1100	1011

$(12AF.CB)_{16} = (100101010111.11001011)_2$

$(1110111000.1011001)_2 = (?)_{16}$

11	1011	1000	.	1011	001
3	C	8	.	C	2

Binary-coded decimal representation (BCD)

A mix of decimal semantics with binary representation. Very useful for I/O and display

Decimal	Binary	BCD
0	00000	0000
1	00001	0001
⋮	⋮	
8	01000	1000
9	01001	1001
10	01010	0001 0000
11	01011	0001 0001
⋮		
19	10011	0001 1001
20	10100	0010 0000

Decimal - BCD conversions

$(3729)_{10} = (?)_{BCD}$

3	7	2	9
0011	0111	0010	1001

$(3729)_{10} = (0011011100101001)_{BCD}$

$(110010011.01101)_{BCD} = (?)_{10}$

1	1001	0011	.	0110	1
1	9	3		6	8

BCD - Binary conversions

Use intermediate decimal conversion

$(101110101101)_2 = 2^{11}+2^9+2^8+2^7+2^5+2^3+2^2+2^0 =$

$= (2989)_{10} = (0010\ 1001\ 1000\ 1001)_{BCD}$

$(100101000001)_{BCD} = (941)_{10}$ → repeated divisions

→ $= (1110101101)_2$

BCD - Hexadecimal conversions

$(3F6)_{16} = 3 \cdot 16^2 + 15 \cdot 16 + 6 = (1014)_{10} =$

$= (1\ 0000\ 0001\ 0100)_{BCD}$

$(1110010111\ 0101)_{BCD} = (3975)_{10}$ →

repeated divisions → $= (F87)_{16}$

Number Representation

Numbers are designated by sign, by magnitude and by position of radix point.

There are two types of radix point: fixed position or floating point, the difference is in the range of numbers that can be represented

The most commonly used positions of radix point are extreme left, making the number a fraction, and extreme right, making the number integer.

One can be obtained from the other by shifting.

Shifting radix point k positions leftward divides by r^k and k positions rightward multiplies by r^k.

Example

$10110.101 = 10.110101 \times 2^3$

$1011\overset{\circ}{0}101 = 1011010.1 \times 2^{-2}$

Sign Notation

Dogital systems cannot recognize symbols as they are, so + and − must be coded somehow.

We'll reserve the leftmost digit of number's representation for sign.

$$(N)_r = (b_{n-1} b_{n-2} \cdots b_1 b_0 . b_{-1} b_{-2} \ldots b_{-m})_r$$

b_{n-1} is the sign, $\sum_{i=-m}^{n-2} b_i r^i$ is the magnitude

For $N \geq 0$ $b_{n-1} = 0$. Negative fixed point numbers can be represented in 3 methods, in each the sign is using the digit $r-1$: Sign-magnitude, $(r-1)$'s complement and r's complement.

Sign-magnitud

$$(\overline{N})_r = ((r-1)\, b_{n-2} b_{n-3} \cdots b_1 b_0 . b_{-1} \cdots b_{-m})_r$$

$(+687)_{10} \rightarrow (0687)_{10}$

$(-687)_{10} \rightarrow ((10-1)687)_{10} = (9687)_{10}$

$(+1101.011)_2 \Rightarrow (01101.011)_2$

$(-1101.011)_2 \rightarrow (11101.011)_2$

(r-1)'s complement

$$(\bar{N})_r = (r^n - r^{-m}) - (N)_r$$

$$(\bar{N})_r = ((r-1)\bar{b}_{n-2} \cdots \bar{b}_1 \bar{b}_0 . \bar{b}_{-1} \bar{b}_{-2} \cdots \bar{b}_{-m})_r$$

$$\bar{b}_i = (r-1) - b_i \quad i = n-2, \ldots, 1, 0, -1, \ldots, -m$$

Examples:

① $\overline{(04857.43)_{10}} =$ $(n-1=4)$

$((10^5 - 10^{-2}) - 04857.43)_{10} = \Rightarrow (\bar{0}\bar{4}\bar{8}\bar{5}\bar{7}.\bar{4}\bar{3})_{10}$

$(99999.99 - 04857.43)_{10} = (95142.56)_{10}$

② $\overline{(0.2731)_{10}} = ((10^1 - 10^{-4}) - 0.2731)_{10} =$ $(n-1=$

$(9.9999 - 0.2731)_{10} = (9.7268)_{10} = (\bar{0}.\bar{2}\bar{7}\bar{3}\bar{1})_{10}$

③ $(\overline{04B7})_{16} = ((16^4 - 16^0) - 04B7)_{16} =$ (n-1=3)

$(FFFF - 04B7)_{16} = (FB48)_{16} =$

$(\overline{0}\,\overline{4}\,\overline{B}\,\overline{7})_{16}$ $= (2^5 - 2^{-3}) - (N)_2 = 11111.111$

④ $(\overline{01101.101})_2 = (10010.010)_2$ (n-1=4)

r's complement

(*) $(\overline{N})_r = r^n - (N)_r$

It follows from (*) that we leave all least significant 0's unchanged, subtract first non-zero from r and then subtract all the rest from r-1.

For 2's complement this means:

- leave all least significant 0's unchanged
- leave first 1 unchanged
- complement all the rest

Examples:

① $\overline{(04857.43)_{10}} = (10^5 - 04857.43)_{10} =$ (n-1=4)

$(100{,}000 - 04857.43)_{10} = (95142.57)_{10}$

Notice that this could be obtained from the 9's complement by adding 10^{-2}.

② $\overline{(0.2731)_{10}} = (10^1 - 0.2731)_{10} = (9.7269)_{10}$ (n-1=0)

③ $\overline{(04B7)_{16}} = (16^4 - 04B7)_{16} = (10000 - 04B7)_{16} =$

$(FB49)_{16} = (FB48)_{16} + 10^0$ (n-1=3)

↑ 15's complement ↖ r^{-m}

④ $\overline{(01101.101)_2} = (\bar{0}\,\bar{1}\,\bar{1}\,\bar{0}\,\bar{1}.\bar{1}\,\bar{0}\,\bar{1})_2 =$ ← Leave as is

$(10010.011)_2$

Example: Represent $-(9)_{10} = -(1001)_2$ by 8 bits (byte) in all 3 notations

Sign-magnitude

$\overline{(00001001)_2} = (10010012)_2$

1's complement

$\overline{(00001001)_2} = (1110110)_2$

2's complement

$\overline{(00001001)_2} = (1110111)_2$

Taking the negative (complement) of the negative (complement) restores the original value

$$\left[\overline{(\bar{N})_r}\right]_r = (N)_r$$

Representation of zero (4 bits)

Signed-magnitude : $\overline{(0000)_2} = (1000)_2$ bad!

1's complement : $\overline{(0000)_2} = (1111)_2$ bad!

2's complement : $\overline{(0000)_2} = \overline{(0000)_2}$ good!

4-bit signed Binary number representation

Decimal	1's complement	2's complement	Sign-magnitude
+7	0111	0111	0111
+6	0110	0110	0110
⋮	⋮	⋮	⋮
+1	0001	0001	0001
+0	0000	0000	0000
-0	1111	0000	1000
-1	1110	1111	1001
⋮			
-6	1001	1010	1110
-7	1000	1001	1111
-8	×	1000	×
	15 values	16 values	15 values

Range of signed Binary numbers.

Given n-bit number, largest positive value, called upper bound, is $(2^{n-1}-1)_{10}$

Smallest negative value, called lower bound is

$-(2^{n-1})_{10}$ for 2's complement and $-(2^{n-1}-1)_{10}$ for 1's complement.

Example: What is the unsigned value, sign-magnitude, 1's and 2's complement values of $(EC)_{16}$?

$(EC)_{16} = (1110\ 1100)_2 = (236)_{10}$ unsigned

$= -(108)_2$ sign-magnitude

In order to find the magnitude in 1's complement we'll look at the negative, since the original number is negative, we'll obtain the positive

$\overline{(1110\ 1100)}_2 = (0001\ 0011)_2 = (19)_{10}$

hence $(EC)_{16} = -(19)_{10}$ in 1's complement interpretation.

$\overline{(1110\ 1100)}_2 = (0001\ 0100)_2 = (20)_{10}$ in 2's complement, hence $(EC)_{16} = -(20)_{10}$

Fixed Point Arithmetic Operations

Numbers are represented by finite length hence results of arithmetic operation have finite precision. This is in contrast to real arithmetic where infinite precision is possible.

Addition of unsigned numbers

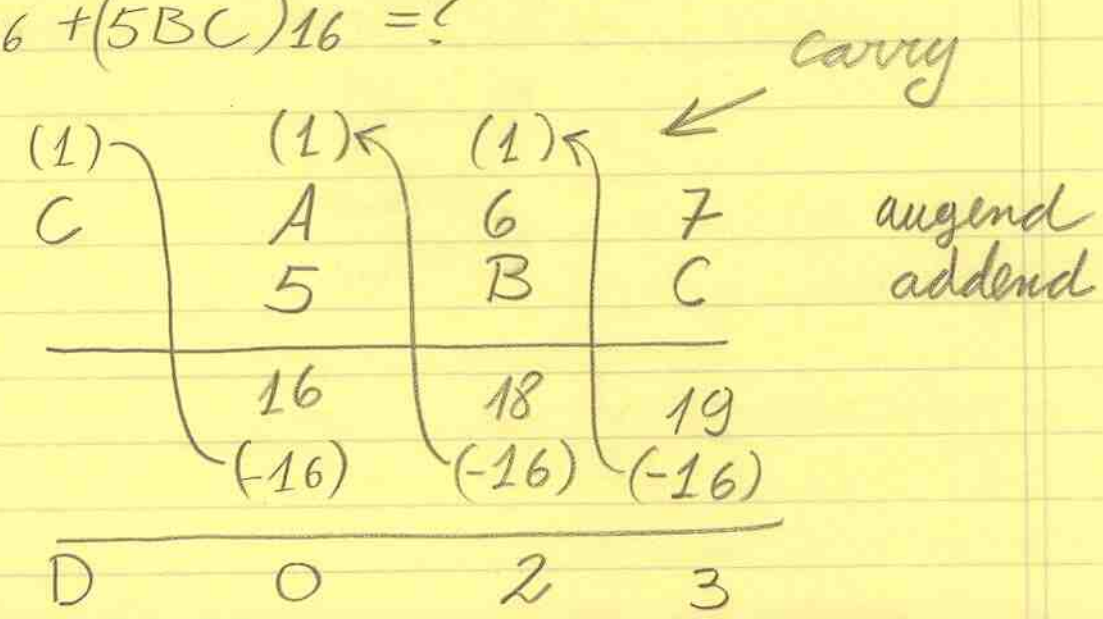
Binary operation involving augend (מחובר) and addend (מחבר)

Consider unsigned numbers

$(1011)_2 + (1101)_2 = ?$

(1)	(1)	(1)	(1)	← carry	
	1	0	1	1	augend
	1	1	0	1	addend
	3	2	2	2	
	(-2)	(-2)	(-2)	(-2)	
1	1	0	0	0	

$(CA67)_{16} + (5BC)_{16} = ?$



Subtraction (addition of signed numbers)

Subtraction works similar in all number systems. It uses **borrow** rather than **carry**. It is however more complicated and requires several steps: In signed numbers we **first** check the **sign** of the two numbers. If **it** is the same we **add** the number and **assign** the appropriate sign. If the signs are **opposite** we compare the **magnitude** and then **subtract** the smaller from

larger and assign the sign of larger to the result.

$$+21 + (+45) = +(21+45) = +66$$

same sign

$$-21 + (-45) = -(21+45) = -66$$

same sign

$$+21 + (-45) = -(45-21) = -24$$

opposite sign

$$-21 + (+45) = +(45-21) = +24$$

opposite sign

Long sequence of decisions: compare, add, subtract, sign determination. Using r's or (r-1)'s makes all these into single addition, regardless of whether it is addition or subtration

addition and subtraction using complements

r's complement addition

6-bit representation

$$\begin{array}{rr} +14 & +\ 0\,0\,1\,1\,1\,0 \\ +12 & \quad 0\,0\,1\,1\,0\,0 \\ \hline +26 & \quad 0\,1\,1\,0\,1\,0 \end{array}$$

$$\begin{array}{rr} +14 & +\ 0\,0\,1\,1\,1\,0 \\ -12 & \quad 1\,1\,0\,1\,0\,0 \\ \hline +2 & \quad 1\,0\,0\,0\,1\,0 \end{array}$$

drop end carry

$$\begin{array}{rr} -12 & +\ 1\,1\,0\,1\,0\,0 \\ +14 & \quad 0\,0\,1\,1\,1\,0 \\ \hline +2 & \quad 1\,0\,0\,0\,1\,0 \end{array}$$

drop end carry

$$\begin{array}{rr} -14 & +\ 1\,1\,0\,0\,1\,0 \\ -12 & \quad 1\,1\,0\,1\,0\,0 \\ \hline -26 & \quad 1\,1\,0\,0\,1\,1\,0 \end{array}$$

drop end carry

↖ negative

Notice that sign bit is negative. Hence, in order to get the magnitude, we perform 2's complement $(\overline{100110})_2 = (011010)_2$

$$\begin{array}{rr} -14 & +\ 1\,1\,0\,0\,1\,0 \\ +12 & \quad 0\,0\,1\,1\,0\,0 \\ \hline -2 & \quad 1\,1\,1\,1\,1\,0 \end{array}$$

↖ negative

overflow problems - be careful!

8-bit signed binary numbers range from -(128)10 to +(127)10

+ -99	+ 1 0 0 1 1 1 0 1	negative
+ -92	1 0 1 0 0 1 0 0	negative
-191	1 0 1 0 0 0 0 0 1	positive! +65

↑ out of range!

+ +57	+ 0 0 1 1 1 0 0 1	positive
+ +75	0 1 0 0 1 0 1 1	positive
+132	1 0 0 0 0 1 0 0	negative!

↑ out of range!

Here is the rule: If the sign of the arguments is similar and the sign of the result is opposite, the result is wrong!

1's complement subtraction

The rule is to complement the subtrahend and perform addition.

$$
\begin{array}{rl}
- & +14 \\
 & +12 \\
\hline
 & +2
\end{array}
\qquad
\begin{array}{l}
\overline{(0\,0\,1\,1\,0\,0)} \\
\\
\\
\end{array}
\qquad
\begin{array}{rl}
+ & 0\,0\,1\,1\,1\,0 \quad \text{minuend} \\
 & 1\,1\,0\,1\,0\,0 \quad \text{subtrahend} \\
\hline
 & 1\,0\,0\,0\,0\,1\,0
\end{array}
$$

$$
\begin{array}{rl}
- & +12 \\
 & +14 \\
\hline
 & -2
\end{array}
\qquad
\begin{array}{l}
\overline{(0\,0\,1\,1\,1\,0)} \\
\\
\\
\end{array}
\qquad
\begin{array}{rl}
+ & 0\,0\,1\,1\,0\,0 \\
 & 1\,1\,0\,0\,1\,0 \\
\hline
 & 1\,1\,1\,1\,1\,0
\end{array}
$$

$$
\begin{array}{rl}
- & +14 \\
 & -12 \\
\hline
 & +26
\end{array}
\qquad
\begin{array}{l}
\overline{(1\,1\,0\,1\,0\,0)} \\
\\
\\
\end{array}
\qquad
\begin{array}{rl}
+ & 0\,0\,1\,1\,1\,0 \\
 & 0\,0\,1\,1\,0\,0 \\
\hline
 & 0\,1\,1\,0\,1\,0
\end{array}
$$

$$
\begin{array}{rl}
- & -14 \\
 & +12 \\
\hline
 & -26
\end{array}
\qquad
\begin{array}{l}
\overline{(0\,0\,1\,1\,0\,0)} \\
\\
\\
\end{array}
\qquad
\begin{array}{rl}
+ & 1\,1\,0\,0\,1\,0 \\
 & 1\,1\,0\,1\,0\,0 \\
\hline
 & 1\,1\,0\,0\,1\,1\,0
\end{array}
$$

$$
\begin{array}{rl}
- & -12 \\
 & -14 \\
\hline
 & +2
\end{array}
\qquad
\begin{array}{l}
\overline{(1\,1\,0\,0\,1\,0)} \\
\\
\\
\end{array}
\qquad
\begin{array}{rl}
+ & 1\,1\,0\,1\,0\,0 \\
 & 0\,0\,1\,1\,1\,0 \\
\hline
 & 1\,0\,0\,0\,0\,1\,0
\end{array}
$$

Subtraction of oppositely signed numbers may produce overflow. Detection by sign bit.