CPNM Lecture 8 - Number Systems and Number Representations

Mridul Sankar Barik

Jadavpur University

2022

Number System

- A number system is merely a convention for representing quantities
- A base is the number used as the reference for constructing the system and also defines number of distinct symbols (digits)
- Different quantities are represented as combinations of the basic digits, with the position or place value specifying the magnitude
- Positional notation: $d_n d_{n-1} \ldots d_0 . d_{-1} d_{-2} \ldots d_{-m}$ is equal to $d_n \times b^n + d_{n-1} \times b^{n-1} + \ldots + d_0 \times b^0 + d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + \ldots + d_{-m} \times b^{-m}$
- A radix point is the symbol used in numerical representations to separate the integer part of a number from its fractional part

Binary I

- Base: 2
- Distinct Symbols: 0, 1
- Any binary number $a_n a_{n-1} \ldots a_0 . a_{-1} a_{-2} \ldots a_{-m}$ is equal to $a_n \times 2^n + a_{n-1} \times 2^{n-1} + \ldots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + a_{-2} \times 2^{-2} + \ldots + a_{-m} \times 2^{-m}$ in decimal
- Example: $(1011)_2 = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_2 = (11)_{10}$
- Example: $(101011)_2 = (1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_2 = (43)_{10}$
- Example: $101.011 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-2} = 4 + 0 + 1 + 0 + \frac{1}{4} + \frac{1}{8} = 5 + 0.25 + 0.125 = 5.375$

Binary II

- Decimal to Binary
  - For the integral part successively divide by 2 until the quotient is 0; take the remainders in reverse order
  - For the fraction part successively multiply by 2 until the fraction part becomes 0; take the integral part of the products in order
- Example: Convert 10.625 to binary

	Quotient	Remainder
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1

Table 1: Integral Part

- $(10)_{10} = (1010)_2$

Binary III

		Integral Part
0.625 * 2	1.250	1
0.250 * 2	0.500	0
0.500 * 2	1.000	1

Table 2: Fraction Part

- $(0.625)_{10}$=$(0.101)_2$
- $(10.625)_{10}$=$(1010.101)_2$
- Binary Addition:
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 0$ and carry 1
- Example

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

Binary IV

- Example

	1	1	1	0	1	1
+	1	0	1	0	0	1
1	1	0	0	1	0	0

- Binary Subtraction:
  - $0 - 0 = 0$
  - $0 - 1 = 1$ and borrow 1
  - $1 - 0 = 1$
  - $1 - 1 = 0$
- Example

	1	0	1	0
-	0	1	0	1
	0	1	0	1

Octal I

- Base: 8
- Distinct Symbols: 0, 1, 2, 3, 4, 5, 6, 7
- Example: $(26)_8 = (2 \times 8^1 + 6 \times 8^0)_{10} = (22)_{10}$
- Example: $(723)_8 = (7 \times 8^2 + 2 \times 8^1 + 3 \times 8^0)_{10} = (448 + 16 + 3)_{10} = (467)_{10}$
- Binary to Octal: group bits from least significant position in group size of three
    - Example: $(100101110101)_2 = (100\ 101\ 110\ 101)_2 = (4565)_8$
    - Add extra zeros in most significant position if number of bits is not a multiple of three
    - Example: $(1010110010)_2 = (1\ 010\ 110\ 010)_2 = (001\ 010\ 110\ 010)_2 = (1262)_8$
    - Example: $(1001111010.1010010)_2 = (001\ 001\ 111\ 010.101\ 001\ 000)_2 = (1172.510)_8$

Hexadecimal

- Base: 16
- Distinct Symbols: 0, 1, 2, 3, 4, 5 ,6, 7, 8, 9, a, b, c, d, e, f
- Example: $(2a)_{16} = (2 \times 16^1 + 10 \times 16^0)_{10} = (42)_{10}$
- Example: $(3e8)_{16} = (3 \times 16^2 + 14 \times 16^1 + 8 \times 16^0)_{10} = (768 + 224 + 8)_{10} = (1000)_{10}$
- Binary to Hexadecimal: group bits from least significant position in group size of four
  - Example: $(1100101011001011)_2 = (1100\ 1010\ 1100\ 1011)_2 = (cacb)_{16}$
  - Add extra zeros in most significant position if number of bits is not a multiple of four
  - Example: $(1001010110010)_2 = (1\ 0010\ 1011\ 0010)_2 = (0001\ 0010\ 1011\ 0010)_2 = (12b2)_{16}$

Numbers with Different Bases

Decimal	Binary	Octal	Hexadecimal
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	a
11	1011	13	b
12	1100	14	c
13	1101	15	d
14	1110	16	e
15	1111	17	f

(r-1)'s Complement I

- To find (r-1)'s complement, subtract each digit of a given number from the largest digit in that number system
- Example: Find 9's complement of $(7548)_{10}$

9	9	9	9
-7	-5	-4	-8
2	4	5	1

- Example: Find 7's complement of $(7543)_8$

7	7	7	7
-7	-5	-4	-3
0	2	3	4

- Example: Find 15's complement of 15a7

f	f	f	f
-1	-5	-a	-7
e	a	5	8

(r-1)'s Complement II

- Example: Find 1's complement of 1101

1	1	1	1
-1	-1	-0	-1
0	0	1	0

r's Complement

- To find r's complement of a given number add 1 to (r-1)'s complement of that number
- Example: Find 10's complement of 2394

9	9	9	9
-2	-3	-9	-4
7	6	0	5
		+	1
7	6	0	6

- Example: Find 2's complement of 1101

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

Subtraction using r's Complement

- Subtraction of two positive numbers (M-N) both of base r may be done as follows
  - Add the minuend to the r's complement of subtrahend N
  - Inspect the result obtained in step1 for an end carry
    - If an end carry occurs discard it
    - If an end carry does not occur, take the r's complement of the number obtained in step1 and place a negative sign in front

Subtraction using r's Complement - Example

- Subtract 2957 - 1396
- Find 10's complement of 1396

9	9	9	9
-1	-3	-9	-6
8	6	0	3
		+	1
8	6	0	4

- Add 8604 to 2956

	2	9	5	7
	8	6	0	4
1	1	5	6	1

- Discard end carry and the result is 1561

Subtraction using r's Complement - Example I

- Subtract 1396 - 2957
- Find 10's complement of 2957

9	9	9	9
-2	-9	-5	-7
7	0	4	2
		+	1
7	0	4	3

- Add 8604 to 2956

1	3	9	6
7	0	4	3
8	4	3	9

- No end carry, so result is negative and is in 10's complement form
- Find 10's complement of 8439

Subtraction using r's Complement - Example II

	9	9	9	9
	-8	-4	-3	-9
	1	5	6	0
			+	1
	1	5	6	1

- So, the result is -1561

Integer Representation I

- Assume that, 3 bits are used for storing integers
- If the integers are unsigned, then there are 8 distinct integer values that can be stored. Range $= 0$ to $2^n - 1$

Integer Value	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- If the integers are signed, then most significant bit is used as the sign bit
- Using **Sign Magnitude** method for representing signed numbers. Range $= +(2^{n-1} - 1)$ to $-(2^{n-1} - 1)$; *Drawback* - Two representations of zero

Integer Value (Signed Magnitude)	Binary
+0	000
+1	001
+2	010
+3	011
-0	100
-1	101
-2	110
-3	111

Integer Representation II

- Using **1's Complement Method** for representing signed numbers. Range $= +(2^{n-1} - 1)$ to $-(2^{n-1} - 1)$; *Drawback* - Two representations of zero

Integer Value (1's Complement)	Binary
+0	000
+1	001
+2	010
+3	011
-3	100
-2	101
-1	110
-0	111

- Using **2's Complement Method** for representing signed numbers. Range $= +(2^{n-1} - 1)$ to $-(2^{n-1})$; Single representation of zero

Integer Value (2's Complement)	Binary
0	000
+1	001
+2	010
+3	011
-4	100
-3	101
-2	110
-1	111

Floating Point Representation

- A fixed point representation uses fixed number of digits after the radix point whereas a floating point representation allows varying number of digits
- Floating point numbers are represented in the form $m \times b^e$, where m is a fractional part and e an integer part
- m is called a *mantissa* or *significand* and e an *exponent* or *characteristic*
- Normalization: requires that $\frac{1}{b} \leq m < 1$
- Example: $\frac{1}{34} = 0.029411765...$ can be stored as 0.0294×10^0 in a system that allows four decimal places in mantissa. However, normalized representation 0.2941×10^{-1} allows us to retain one additional significant digit
- Source of Error - mantissa holds only a finite number of significant digits

Floating Point Numbers - Example of a Hypothetical Machine I

[Taken from Rajaraman]

- Each location can store 6 digits and has provision to store one or more signs
- Fixed point representation: | 2456 | 24 |
  - Max. no. = 9999.99
  - Min. no = 0000.01
- Normalized floating point representation:
  - Mantissa: less than 1 or greater than equal to 0.1
  - Exponent: power of 10 which multiplies mantissa
  - Example: $44.85 \times 10^6 = 0.4485 \times 10^8$
    $(+)$ | 4485 | $(+)08$ |
  - Example: $.004854 = 0.4854 \times 10^{-2}$
    $(+)$ | 4854 | $(-)02$ |
  - Range: 0.9999×10^{99} to 0.1000×10^{-99}

Floating Point Numbers - Example of a Hypothetical Machine II

[Taken from Rajaraman]

- Arithmetic operations with normalized floating point numbers
  - Addition:
    - .4546E5 + .5433E5
      Exponents are equal add mantissas, sum = .9979E5
    - .4546E5 + .5433E7
      Operand with larger exponent is kept as it is
      Operand with smaller exponent is shifted right by number of places equal to the difference in two exponents
      .4546E5 + .5433E7 = .0045E7 + .5433E7 = .5478E7
    - If the two numbers are $m_1 \times b^{e_1}$ and $m_2 \times b^{e_2}$ and $e_1 < e_2$ then the first number is transformed as $m_1 \times b^{-(e_2 - e_1)}) \times b^{e_2}$ before addition
    - .4546E3 + .5433E7 = .0000E7 + .5433E7 = .5433E7
    - .6434E3 + .4845E3 = 1.1279E3 = .1127E4
    - .6434E99 + .4845E99 = 1.1279E99 = .1127E100
      Exponent part cannot store more than two digits ⇒ **OVERFLOW**

Floating Point Numbers - Example of a Hypothetical Machine III

[Taken from Rajaraman]

- Subtraction
  - .5452E-3 - .9432E-4 = .5452E-3 - .0943E-3 = .4509E-3
  - .5452E3 - .5424E3 = .0028E3 = .2800E1
  - .5452E-99 - .5424E-99 = .0028E-99 = .2800E-101
    Cannot store a number smaller than the smallest number $\Rightarrow$ **UNDERFLOW**
- Multiplication:
  - (i) multiply the mantissa; (ii) add the exponent; (iii) result mantissa is normalized and the exponent adjusted appropriately
  - .5543E12 $\times$ .4111E-15 = .22787273E-3 = .2278E-3
  - .1111E10 $\times$ .1234E15 = .01370974E25 = .1370E24
  - .1111E51 $\times$ .4444E50 = .04937284E101 = .4937E100 $\Rightarrow$ **OVERFLOW**
  - .1234E-49 $\times$ .1111E-54 = .01370974E-103 = .1370E-104 $\Rightarrow$ **UNDERFLOW**

Floating Point Numbers - Example of a Hypothetical Machine IV

[Taken from Rajaraman]

- Division:
  - (i) mantissa of numerator is divided by that of the denominator; (ii) denominator exponent is subtracted from the numerator exponent; (iii) quotient mantissa is normalized and exponent adjusted appropriately
  - .9998E1 $\div$ .1000E-99 $=$ 9.9980E100 $=$ .9998E101 $\Rightarrow$ **OVERFLOW**
  - .9998E-5 $\div$ .1000E98 $=$ 9.9980E-103 $=$ .9998E-102 $\Rightarrow$ **UNDERFLOW**
  - .1000E5 $\div$ .9999E3 $=$ .100010001E2 $=$ .1000E2

Floating Point Numbers - Example I

- Assume 8 bits are used for representation. 1 Sign bit, 3 bits for exponent and 4 bits for mantissa $\rightarrow b \mid bbb \mid bbbb$
- $110.11 \xrightarrow{\text{normalized}} 0.11011 \times 2^3 \rightarrow 0 \mid 011 \mid 1101$
- $0.01011 \xrightarrow{\text{normalized}} 0.1011 \times 2^{-1} \rightarrow 0 \mid 111 \mid 1011$
- $0.001011 \xrightarrow{\text{normalized}} 0.1011 \times 2^{-2} \rightarrow 0 \mid 110 \mid 1011$
- When we normalize a decimal fraction, the digit after decimal point can be any one of 1, 2, ..., 9
- When we normalize binary fraction, the bit after the binary point is always 1 $\rightarrow$ so, we can store an extra bit by assuming that the first bit in the mantissa is always 1 and this bit is not actually stored

  $0.01011 \xrightarrow{\text{normalized}} 1.0110 \times 2^{-2} \rightarrow 0 \mid 110 \mid 0110$

  $0.001011 \xrightarrow{\text{normalized}} 1.0110 \times 2^{-3} \rightarrow 0 \mid 101 \mid 0110$