02 Information theory

02.02 Binary arithmetic

- Positional notation
- Unsigned integers
- Unsigned fixed-point
- Signed numbers
- Floating point numbers
- Base conversions
- Binary arithmetic

Positional notation of unsigned integers

- The *base-b positional representation* of an integer number of *n* digits has the form

$$c_{n-1}c_{n-2}...c_1c_0 \quad (1)$$

- The value of the number is

$$c_{n-1}b^{n-1} + c_{n-2}b^{n-2} + ... + c_1b^1 + c_0b^0 \quad (2)$$

- *n* digits encode all integer numbers from 0 to b^n-1

  Example:

  b=2, *n*=5 10011=1*16+1*2+1*1=19

  11111=31=2^5-1

Base conversion

- From base *b* to decimal:

$$c_{n-1}b^{n-1} + c_{n-2}b^{n-2} + \ldots + c_1 b^1 + c_0 b^0 \qquad (2)$$

- From decimal to base *b*:

  digits from c_0 to c_{n-1} are obtained as remainders of subsequent divisions by *b* of the digital number

$$c_{n-1}b^{n-1} + c_{n-2}b^{n-2} + \ldots + c_1 b^1 + c_0 b^0 = \qquad (3)$$
$$= (c_{n-1}b^{n-2} + c_{n-2}b^{n-3} + \ldots + c_1 b^0)b + c_0$$

Base conversion
example

$(29)_{(10)}=(11101)_{(2)}$

	29		
29	14	**1**	29=14*2+**1**
14	7	**0**	14=7*2+**0**
7	3	**1**	7=3*2+**1**
3	1	**1**	3=1*2+**1**
1	0	**1**	1=0*2+**1**

$(11001)_{(2)}=(25)_{(10)}$

position	digit	weight		
0	**1**	1	1	+
1	**0**	2	0	+
2	**0**	4	0	+
3	**1**	8	8	+
4	**1**	16	16	
			25	

Fixed-point notation of unsigned rational numbers

- The *base-b positional representation* of a rational number of *n+m* digits has the form

$$c_{n-1}c_{n-2}\ldots c_1c_0.c_{-1}\ldots c_{-m} \quad (4)$$

- The value of the number is

$$c_{n-1}b^{n-1}+c_{n-2}b^{n-2}+\ldots+c_1b^1+c_0b^0+c_{-1}b^{-1}+\ldots+c_{-m}b^{-m} \quad (5)$$

- *n+m* digits encode all rational numbers of the form

$$\frac{Num}{2^m} \quad (6)$$

with $0 \le Num \le 2^{n+m}-1$

Base conversion

- From base *b* to decimal:

$$c_{n-1}b^{n-1}+c_{n-2}b^{n-2}+\ldots+c_1b^1+c_0b^0+c_{-1}b^{-1}+\ldots+c_{-m}b^{-m} \quad (5)$$

- From decimal to base *b*:
  - Solution 1: Given a digital number $X=Num/2^m$, convert *Num* and shift the decimal point *m* positions left
  - Solution 2: Given a digital number $X=X_{int}.X_{frac}$, convert the integer part X_{int} as outlined before, then convert the fractional part X_{frac} obtaining each digit as the integer part of the result of subsequent multiplications by *b*:

$$(c_{-1}b^{-1}+c_{-2}b^{-2}\ldots+c_{-m}b^{-m})b = c_{-1}+c_{-2}b^{-1}\ldots+c_{-m}b^{-m-1} \quad (7)$$

Base conversion
example

$(29.375)_{(10)}=(11101.011)_{(2)}$

	29		
29	14	1	29=14*2+**1**
14	7	0	14=7*2+**0**
7	3	1	7=3*2+**1**
3	1	1	3=1*2+**1**
1	0	1	1=0*2+**1**

0.375	*2 =	0.75	=	0	+	0.75
0.75	*2 =	1.5	=	1	+	0.5
0.5	*2 =	1	=	1	+	0

Binary arithmetics

- Binary addition:

	0	1
0	0	1
1	1	10

1			1	1	1		
1	**5**		**0**	**1**	**1**	**1**	**1**
	6		**0**	**0**	**1**	**1**	**0**
2	**1**		**1**	**0**	**1**	**0**	**1**

- Binary multiplication:

	0	1
0	0	0
1	0	1

	1	**2**
	1	**5**
	6	0
1	2	-
1	**8**	**0**

	1	1	1				
				1	**1**	**0**	**0**
				1	**1**	**1**	**1**
				1	1	0	0
			1	1	0	0	-
		1	1	0	0	-	-
	1	1	0	0	-	-	-
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**

Signed numbers

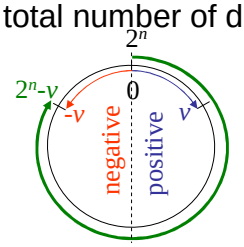
- Sign bit: 0=positive, 1=negative
  $(3)_{(10)}$ = 0 0011; $(-3)_{(10)}$ = 1 0011
- 1's complement
  $(3)_{(10)}$ = 0 0011; $(-3)_{(10)}$ = 1 1100 = (0 0011)'
- 2's complement
  $(3)_{(10)}$ = 0 0011; $(-3)_{(10)}$ = 1 1101 = (0 0011)'+1

2's complement numbers

- The 2's complement representation of a negative number (say, -*v*) is obtained from the representation of *v,* by complementing each bit (including the sign bit) and adding 1

  $-v \rightarrow v'+1$

- The resulting binary configuration corresponds to the unsigned number 2^n-*v* (*i.e.*, the complement of *v* to 2^n), where *n* is the total number of digits, including the sign bit.



2's complement arithmetic

- 2's complement numbers can be added by using the conventional rules of binary addition

  $a+(-b) \rightarrow a+(2^n-b)=2^n-(b-a) \rightarrow -(b-a)=a-b$

- Example:

  $(3)_{(10)}-(5)_{(10)}$
  $(00011)_{(2)}-(00101)_{(2)} = (00011)_{(2)}+(-00101)_{(2)}$
  $(00011)_{(2c)}+(11011)_{(2c)} = (11110)_{(2c)}$
  $(-00010)_{(2)}$
  $(-2)_{(10)}$

Floating point numbers

$$s \bullet 0.M \bullet b^{se \bullet E}$$

- s = sign (1 bit)
- M = Mantissa (m bits)
- b = base (0 bits)
- se = exponent sign (1 bit)
- E = exponent (e bits)

Encoding floating point numbers requires words of n digits, with $n=1+m+1+e$

Floating point encoding

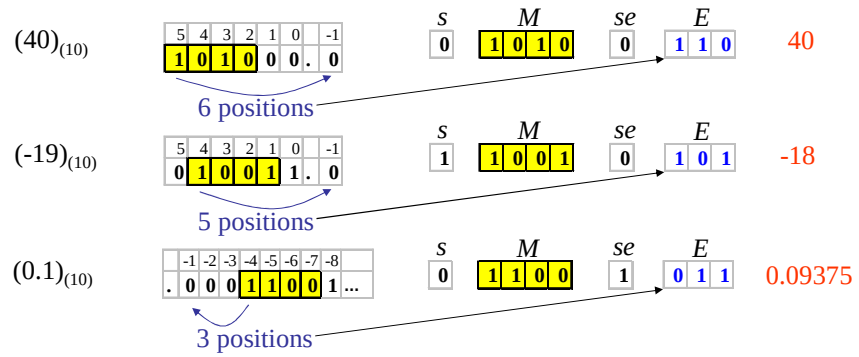
$s \bullet 0.M \bullet b^{se \bullet E}$

1. Start from a fixed-point binary encoding using a sufficient number of bits
2. Evaluate the number of shifts required to bring the most significant 1 in position –1
3. Take as mantissa (*M*) the most significant *m* bits (starting from the first 1) of the fixed-point encoding
4. Set *se*=1 if the number is lower than 0.5, *se*=0 otherwise
5. Assign the exponent *E* with the binary representation of the number of shifts computed at step 2
6. Set the sign bit *s* as usual

Floating point encoding

$s \bullet 0.M \bullet b^{se \bullet E}$

Example: *m*=4, *e*=3



Floating point arithmetic

$$A = s_A 0.M_A 2^{se_A E_A} \qquad B = s_B 0.M_B 2^{se_B E_B}$$

$$C = A + B = s_A 0.M_A 2^{se_A E_A} + s_B 0.M_B 2^{se_B E_B}$$
$$= \left(s_A 0.M_A + s_B 0.M_B \frac{2^{se_B E_B}}{2^{se_A E_A}} \right) 2^{se_A E_A}$$
$$= \left(s_A 0.M_A + s_B 0.M_B 2^{-(se_A E_A - se_B E_B)} \right) 2^{se_A E_A}$$

$$C = A * B = s_A 0.M_A 2^{se_A E_A} * s_B 0.M_B 2^{se_B E_B}$$
$$= (s_A 0.M_A * s_B 0.M_B) 2^{se_A E_A} * 2^{se_B E_B}$$
$$= (s_A 0.M_A * s_B 0.M_B) 2^{(se_A E_A + se_B E_B)}$$

Floating point arithmetic

$s \bullet 0.M \bullet b^{se \bullet E}$

A = 3.5 = $+0.111*2^{+2}$
B = 1.5 = $+0.110*2^{+1}$
C = A+B = 5 = $+0.101*2^{+3}$

		M	E
A	.	1 1 1	2
B	.	1 1 0	1
A	.	1 1 1	2
B	.	0 1 1	2
C	1 .	0 1 0	2
C	.	1 0 1	3

A = 3.5 = $+0.111*2^{+2}$
B = 1.5 = $+0.110*2^{+1}$
C = A*B = 5.25 = $+0.10101*2^{+3}$

		M	E
A	.	1 1 1	2
B	.	1 1 0	1
C	.	1 0 1 0 1	3
C	.	1 0 1	3

5

IEEE 754

$$(-1)^s \times 1.M \times 2^E$$

$$E = Exp - bias$$

Name	Common name	Bits	Bits s	Bits M	Bits Exp	Emin	Emax	Bias
binary16	Half precision	16	1	10	5	-14	15	15
binary32	**Single precision**	**32**	**1**	**23**	**8**	**-126**	**127**	**127**
binary64	Double precision	64	1	52	11	-1022	1023	1023
binary128	Quadruple precision	128	1	112	15	-16382	16383	16383

Conventional assumptions binary32

Exp	M	Meaning
0	0	0
255	0	Infinity
255	non zero	NaN