CM1604 Computer Systems Fundamentals

Signed Integers, Bicimal & IEEE754

Week No :02 | Rathesan

ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

CONTENT

- Two’s complement (representing negative numbers)
- Floating Point Numbers
  - Bicimal
  - IEEE 754

Signed Integers

- Sometimes sign as well as value is required, for example

- Two’s complement

Two’s Complement

- To store positive number
  - Just store binary equivalent
  - So 37 in 8-bits would be stored as 00100101
- To store negative number
  - Get positive binary number
  - Flip or complement each bit (i.e. change 1 to 0 and 0 to 1 from the previous step)
  - Add 1 to the resultant binary

Two’s Complement -83

- Decimal Number   -83
- **Step 01: Write original number as positive number**

  01010011
- **Step 02: Flip the Bits**

  10101100
- **Step 03**: **Add 1**

  Binary representation of –83 is 10101101

Two's Complement -113

- Decimal Number -113
- **Step 01: Write original number as positive number**

  01110001
- **Step 02: Flip the Bits**

  10001110
- **Step 03: Add 1**

  Binary representation of –113 is 10001111

Range of Values Two's Complement

- 8 bit -128 to +127
- 16 bit -32,768 to +32,767
- 32 bit -2,147,483,648 to +2,147,483,647

- n Bits
- 2^n values
- Values: $-(2^{n-1}) \rightarrow 0 \rightarrow ((2^{n-1}) - 1)$

Floating Point Numbers & IEEE 754

- Real numbers
- Includes fractions and integers

  eg: 3.14159, 0.1235, -128.3
- Decimal fractions
- Value in a column = 10 times value in column to its right

Floating Point Numbers

	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}
Positional Value	1000	100	10	10		0.1	0.01
Multiplier	4	2	7	8	.	6	5
Value	(4x1000)+(2x100)+(7x10)+(8x1)+(6x0.1)+(5x0.01) =4278.65						

Bicimal

- Binary format for representing fractional values
- Fixed Point

	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
	0.5	0.25	0.125	0.0625
.				

Decimal to Bicimal 0.75

		2^{-1}	2^{-2}	2^{-3}	2^{-4}
		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
		0.5	0.25	0.125	0.0625
Decimal 0.75		0.75= 0.5+0.25			
	.	1	1	0	0

Decimal to Bicimal 0.625

		2^{-1}	2^{-2}	2^{-3}	2^{-4}
		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
		0.5	0.25	0.125	0.0625
Decimal 0.625		0.625=0.5+0.125			
	.	1	0	1	0

Fixed Point Format

- Not suitable for storing very small or very large real numbers
- Avogadro's number 6.0221367 x 10^{+23} would require about 80 bits for the integer part
- Mass of a hydrogen atom 1.6733 x 10^{-24} would have required well over 80 bits for the fractional part
- Hence fixed point format is not suitable for use in computer representation of very small or very large numbers

Floating Point Format

- Can be used to represent very small or very large numbers fairly easily
- Before IEEE 754 standard, different manufacturers use different methods
- IEEE 754 standardised the method and this is now adopted by all computer manufacturers
- IEEE 754 is a simple and efficient way to represent floating point format

Normalised Format

- 3 parts to a normalised representation
  - The integer part (single digit)
  - The part after the decimal point
  - The power part

- Examples
- 15 in normalized form $1.5x10^{+1}$
- 415 in normalized form $4.15x10^{+2}$
- 0.0017 in normalized form $1.7x10^{-3}$
- -645 in normalized form $-6.45x10^{2}$

Mantissa and Exponent Examples

Mantissa and Exponent Examples

- 6244
- -735
- -0.0045
- -2
- -0.0025

Floating Point in Binary

- Format is similar to floating point in decimal numbers

  Examples

  $$11011101 = 1.1011101 \times 2^{+7}$$

  Mantisa = 1.1011101

  Exponent = +7

  Decimal point is referred in this case as the bicimal point

Floating Point in Binary

- 0.00101
- $=1.01 \times 2^{-3}$

- Mantissa = 1.01   Exponent = -3

- -101101.01
- $= -1.0110101 \times 2^{+5}$

- Mantissa = -1.0110101 Exponent = +5

Decimal to Normalised Bicimal Form

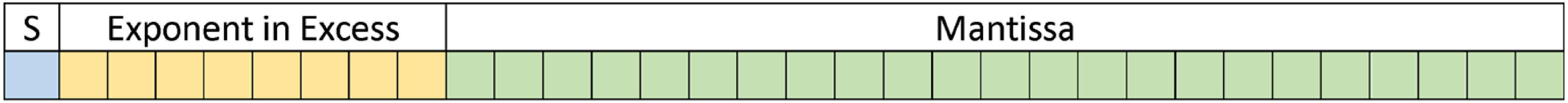
- 43.625
- **Step 1** Integer part to binary format = 101011
- **Step 2** Fractional Part to Bicimal = .101
- **Step 3** Combine Step 1 and Step 2 = 101011.101
- **Step 4** Normalise = 1.01011101 x 2^{+5}

- Mantissa = 1.01011101
- Exponent = +5

IEEE 754

- There are 32 bits in Standard (single precesion) IEEE754 representation of floating point numbers in binary and is divided into three parts namely:

- Sign bit – First Bit
- Exponent in excess form – Next 8bits
- Mantissa – Last 23 bits



IEEE 754 Format Example

Example: 12.5

- **Step 1** Integer part to binary format 1100
- **Step 2** Fractional Part to Bicimal = .1
- **Step 3** Combine Step 1&2 = 1100.1
- **Step 4** Normalise the result = 1.1001 x2 $^{+3}$
- **Step 5** Mantissa = 1.1001
- **Step 6** Exponent in excess form (8bits) = 127+3=130 = 10000010

Sign Bit = 0 Exponent Bits = 10000010 Mantissa Bits = 10010000000000000000000

12.5 in IEEE 754 Format = 0 10000010 $10010000000000000000000_2$

12.5 in IEEE 754 Hex Format = 0 100|0001|0 100|1000|0000|0000|0000|0000_2

4 1 4 8 0 0 0 0

41480000

IEEE 754 Format Example

Example -112.625

- **Step 1** Integer part to binary format 1110000
- **Step 2** Fractional Part to Bicimal = .101
- **Step 3** Combine Step 1&2 = 1110000.101
- **Step 4** Normalise the result =1.110000101 x 2^{+6}
- **Step 5** Mantissa = 1.110000101
- **Step 6** Exponent in excess form (8bits) = 127+6=133 = 10000101

Sign Bit = 1 Exponent Bits = 10000101 Mantissa Bits = 11000010100000000000000

-112.5 in IEEE 754 Format = 1 10000101 11000010100000000000000 $_2$

-112.5 in IEEE 754 Hex Format = 1 100|0010|1 110|0001|0100|0000|0000|0000

C 2 E 1 4 0 0 0

C2E14000

IEEE754 Format to Decimal

Example: C2E14000

- **Step 1:** Write it in binary 1 10000101 11000010100000000000000
- **Step 2:** Sign Bit= 1 = Negative Number
- **Step 3:** Exponent in Excess form= 10000101_2 = 133
- **Step 4:** Exponent = 133-127 = 6
- **Step 5:** Mantissa(adding the leading 1. and removing the trailing zeros) = 1.110000101
- **Step 6:** Number in Binary Form = $-1.110000101 \times 2^{+6}$ = -1110000.101
- **Step 7:** Decimal Number = **-112.625**

READING

Chapter # 3

- Computer science illuminated. Jones & Bartlett Learning.