07 – NON-INTEGER DATA

COMP256 – COMPUTING ABSTRACTIONS

DICKINSON COLLEGE

REPRESENTING NON-INTEGER DATA

- Decimal Numbers
  - Rational Numbers
  - Fixed Point
  - Floating Point
- Character Data
  - EBCDIC
  - ASCII
  - UNICODE
- Colors
  - RGB

RATIONAL REPRESENTATION

- *Rational representations* use the integer representations (e.g. Two's Complement) for the numerator and denominator of a fraction.
  - Example: `4.1`
    - Decimal: `41/10`
    - Binary: `0010 1001  0000 1010`
`(41)  (10)`
  - Example: `-12.25`
    - Decimal: `-49/4`
    - Binary: `1100 1111 0000 0100`
`(-49)  (4)`

FIXED POINT REPRESENTATION

- *Fixed point representation* uses a fixed number of bits to the right of the binary point to represent decimal values.
  - Example: 8 bit fixed point with 4 bits to the right of the binary point.
    - `1011 1101` → `1011.1101`

FIXED POINT REPRESENTATION

- Converting from base 10 to fixed point
  - Example: Convert to 8-bit fixed point with 4 bits to right of decimal.
    - 4.3125

FIXED POINT REPRESENTATION

- Example:
  - Convert 2.3 to 8-bit fixed point representation with 4 bits to the right of the binary point.

ROUNDING MODES

- When fixed (or floating point) values cannot be represented exactly they must be rounded. There are alternative ways of rounding:
  - Round to nearest - ties to even
  - Round up (toward +∞)
  - Round down (toward -∞)
  - Truncation (Round toward 0)

PRECISION ERRORS

- A precision error occurs when there are insufficient bits to the right of the binary point to represent a value exactly.
  - Example: 0.1

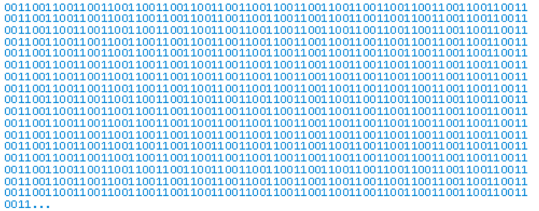


Image From: https://www.exploringbinary.com/why-0-point-1-does-not-exist-in-floating-point/

```
System.out.println(101.0 * 0.1);  →      10.100000000000001
```

SCIENTIFIC NOTATION

- The familiar scientific notation forms the basis for the floating point representation.
  - Examples:
    - Base 10:
      - 115.875
      - 115875 x 10^{-3}
      - 1.15875 x 10^{2}
    - Base 2:
      - 0111 0011.101
      - 0111 0011 101 x 2^{-3}
      - 1.11 0011 101 x 2^{6}

NORMALIZED FORM

- The *normalized form* of a number in scientific notation has exactly one significant digit to the left of the *radix point*.
  - Examples:
    - 23793.13 = **2**.379313 x 10^{4}
    - 0.003298 = **3**.298 x 10^{-3}
    - 1001000.0110 = **1**.0010000110 x 2^{6}
    - 0.0101101 = **1**.01101 x 2^{-2}
    - 0.00010111 =
    - 0010001.01101 =

FLOATING POINT REPRESENTATION

- A 14-bit FP Model

$$n_{10} = (-1)^{Sign} \cdot 2^{(Exponent - excess)} \cdot Significand$$

- **Exponent:** Unsigned representation
- **excess:** Constant value subtracted from the Exponent.
  - excess = half of the number of Exponent values minus 1.
    - e.g. excess = $2^5/2 - 1 = 32/2 - 1 = 15$ *for this 14 bit model*.
- **Significand:** The bits to the right of the binary point.

FLOATING POINT EXAMPLES

- What are the following values in our 14-bit FP model:
  - 1 10101 1001 1010
  - 22.75

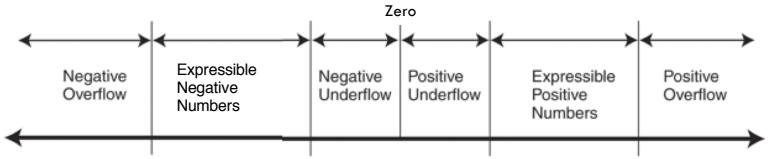
WORKED EXAMPLE

- 1 10101 1001 1010

• $n_{10} = (-1)^{Sign} \cdot 2^{(Exponent\ -excess)} \cdot Significand$	1. Start with formula
• $n_{10} = (-1)^{1} \cdot 2^{(10101_2 - 15_{10})} \cdot 1.1001\ 1010_2$	2. Add sign, exponent, excess significand
• $n_{10} = -1 \cdot 2^{(21_{10} - 15_{10})} \cdot 1.1001\ 1010_2$	3. Evaluate sign and convert exponent
• $n_{10} = -1 \cdot 2^{(6_{10})} \cdot 1.1001\ 1010_2$	4. Evaluate exponent
• $n_{10} = -1 \cdot 11001\ 10.10_2$	5. Convert significand to fixed point
• $n_{10} = -1 \cdot 102.5_{10}$	6. Convert significant to base 10
• $n_{10} = -102.5_{10}$	7. Obtain final answer

FLOATING POINT ERRORS

- A *precision error* is when a value in the expressible range cannot be represented exactly and must be rounded.
- An *overflow error* is when a value is out of range. It is too positive or too negative to be represented in the number of bits being used.
- An *underflow error* is when a result is too close to zero to be represented in the number of bits being used.



EBCDIC

- Enhanced Binary Coded Decimal Interchange Code
- A
  - 1100 0001
  - 0xC1
  - 193_{10}

EBCDIC CODE

Least Significant Nibble		Most Significant Nibble															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	DS		SP	&	-						{	}	\	0
0001	1	SOH	DC1	SOS						a	j	~		A	J		1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	DC3							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	6	LC	BS	EOB / ETB	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	PRE / ESC	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9	RLF	EM						\	i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!	¦	:								
1011	B	VT				.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@								
1101	D	CR	IGS	ENQ	NAK	(	)	___	'								
1110	E	SO	IRS	ACK		+	;	>	=								
1111	F	SI	IUS	BEL	SUB	!	¬	?	"								

Image from: http://www.rtty.com/CODECARD/codecrd1.htm

STF 078

ASCII CODE

- American Standard Code for Information Interchange
- A
  - 0100 0001
  - 0x41
  - 65_{10}

MSB / LSB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000 0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001 1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010 2	SP	!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
0011 3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100 4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101 5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
0110 6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111 7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

BINARY — HEX — ASCII

UNICODE

- Universal Character Encoding
  - https://home.unicode.org/
    - Currently 137,994 *code points*
    - 1,111,998 possible code points
  - U+ prefix
    - But Hex Values

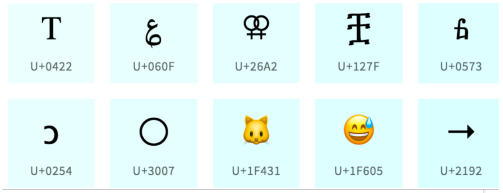


Image from: https://home.unicode.org/

UTF-8 ENCODING

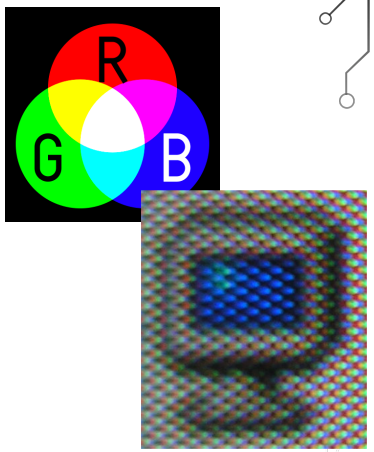
- 8-bit Unicode Transformation Format
  - Encoding of Unicode character code points into variable length representations.

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Image from: https://en.wikipedia.org/wiki/UTF-8

RGB COLOR MODEL

- The RGB color model specifies the intensity of red, green and blue light to mix to create a color.
  - 24 bit color (3 bytes)
    - `0xFF0000` = bright red
    - `0x00FF00` = bright green
    - `0x0000FF` = bright blue
    - `0x000000` = black
    - `0xFFFFFF` = white



Images from: https://en.wikipedia.org/wiki/RGB_color_model