Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 11: Floating Point Arithmetic

Yale Patt

The University of Texas at Austin

Austin, Texas

Spring, 2021

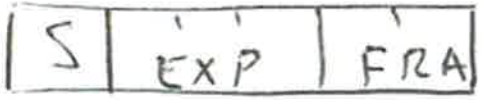
Scientific Notation and its representation in binary

- Avogadro's Number: 6.022 x 10^23
- Where is the Decimal Point
  - What determines where the decimal point is?
  - Ergo, "floating point"
- Bits for range vs bits for precision
  - Von Neumann's Quip
- Binary Representations
  - 32 bits: 1 bit sign, 8 bits exponent, 23 bits of fraction
  - 64 bits (the default): 1 bit sign, 11 bits exponent, 52 bits fraction
  - 16 bits (recently for graphics): 1 bit sign, 5 bits exp, 10 bits fra

Outline

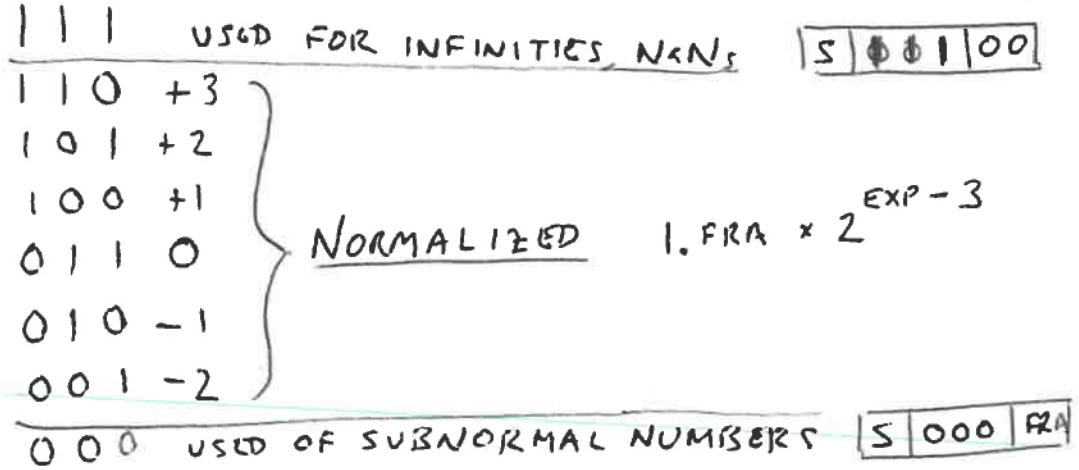
- Scientific Notation and its representations in binary
  - Avogadro's Number
  - Format (Precision vs Range)
  - Common floating point data types
  - Why it is called floating point
- My 6-bit floating point data type
  - Redundant most significant bit
  - Excess code for exponents (infinity, subnormals)
- Rounding
  - Four rounding modes
  - Guard, Round, Sticky bits
  - Wobble
- Infinities, NaNs
- Subnormal numbers (gradual underflow)
- Exceptions: Invalid, DivBy0, Underflow, Overflow, Inexact
  - Quiet vs Signaling

My 6-bit floating point data type



$1.FRA \times 2^{EXP - EXCESS}$

- **Excess code for range (excess is 011, i.e. 3)**



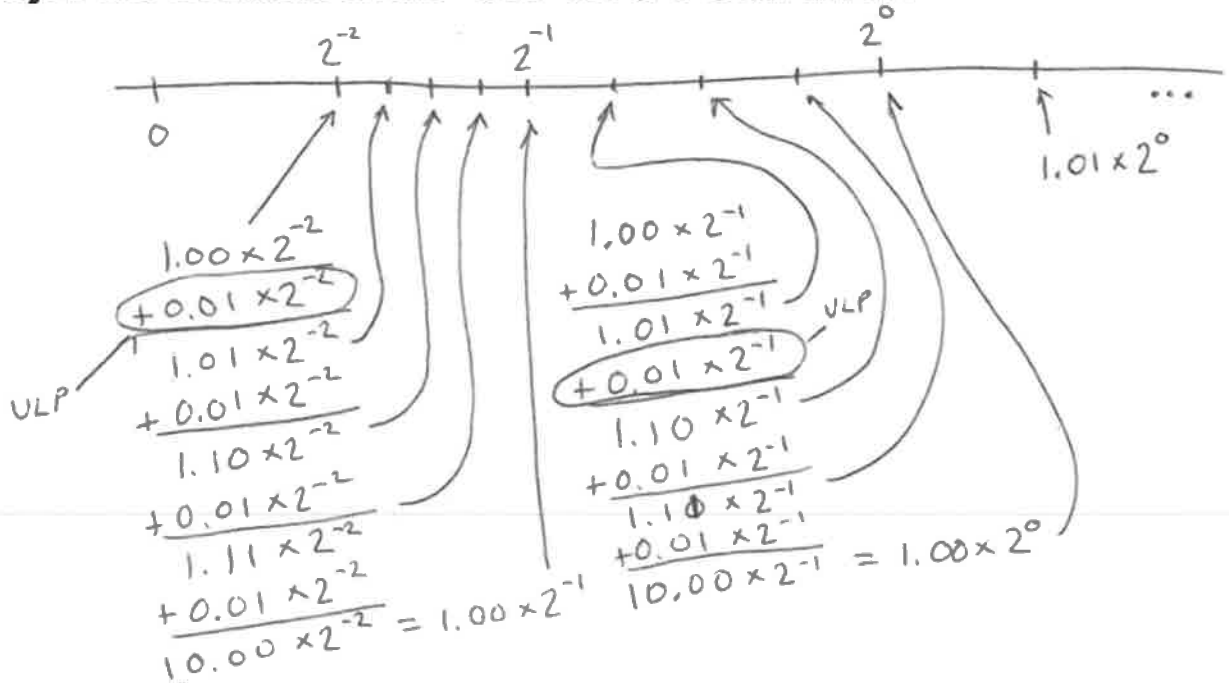
- **Signed-magnitude for precision**
  - **Redundant most significant bit (because radix = 2)**

- **An example: 6 5/8. 110.101 (Unnormalized)**
  - **Normalize it: 1.10101 x 2^2**

  - **Store in memory: Subsequent read from memory: 1.10 x 2^2 = 6**
    - **We lost 5/8 because we had 2 fraction bits, but we needed 5 fraction bits**

My 6-bit floating point data type

- **Exact representations on the real line**



- **Maximum value:**

| 0 | 110 | 11 | $= 1.11 \times 2^{3} = 14$

- **Minimum normalized value:**

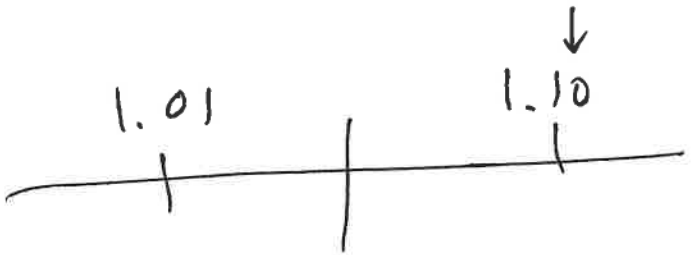
| 0 | 001 | 00 | $= 1.00 \times 2^{-2} = \frac{1}{4}$

0.01×2^{-2}

Rounding

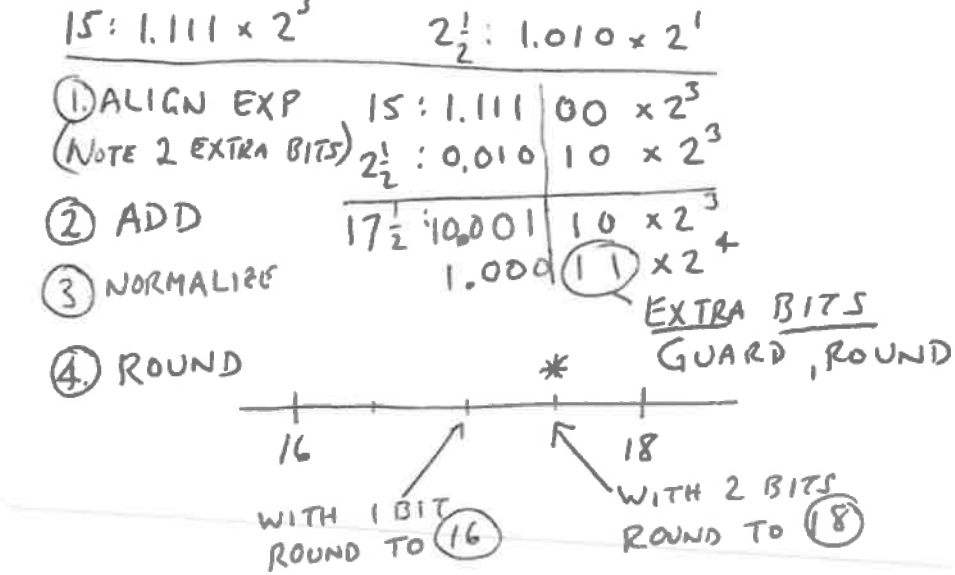
- **Why, When**
  - **Why: when a value can not be represented exactly**
  - **When: Often, most values can not be represented exactly**
  - **Example (with our 6 bit floating point): 1.011 (1 3/8)**

- **Rounding Modes**
  - **Round up: 1.10 (1 ½)**
  - **Round down: 1.01 (1 ¼)**
  - **Round to zero: 1.01 (1 ¼)**
  - **Unbiased Nearest: 1.10 (1 ½)**
    - **Why not 1.01 (1 ¼)?**
    - **1 ¼ is just as "near" to 1 3/8 as 1 ½ is**
    - **Unbiased → when equal, round to the value with 0 in the ULP**

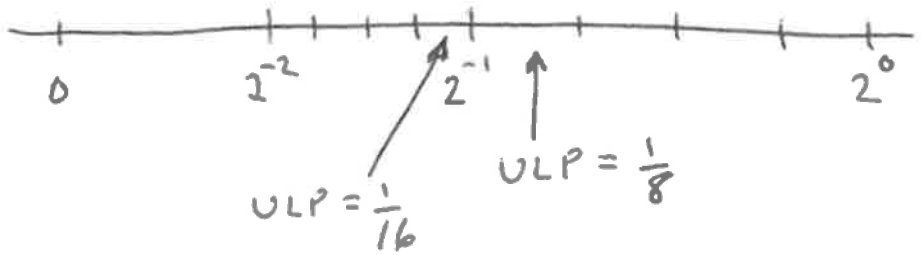


Rounding

- **Guard, Round, Sticky bits**
  - **Extra bits carried along to help in rounding (1 bit or 2 bits?)**
  - **Example: Add 15 + 2 ½ (This example uses 3 bits of fraction!)**



- **Wobble (since 1 ULP < 2^k is half the size of 1 ULP > 2^k)**
  - **Best case, because radix = 2**



Infinity

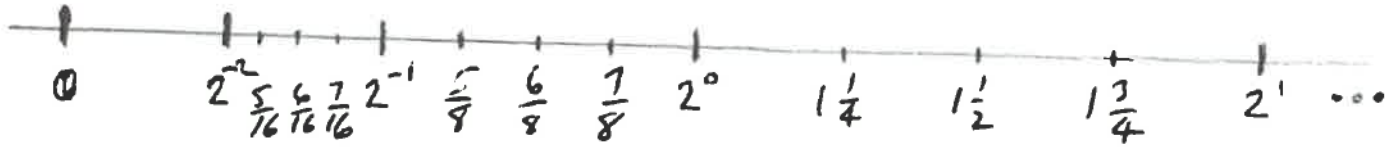
5 11100 ±∞

- **Exact vs Overflow**
  - (finite operands) → infinite results
  - Examples: tan(90 degrees), 5 divided by 0
- **Infinity is NOT the same as undefined**
  - Example: continued fraction expansion
  - Simple examples:
    - infinity + 7 = infinity
    - infinity + infinity = infinity
    - 5 divided by infinity = 0
  - Code example
    - X=5
    - Y=0
    - Z=X/Y
    - W= arctan(Z)

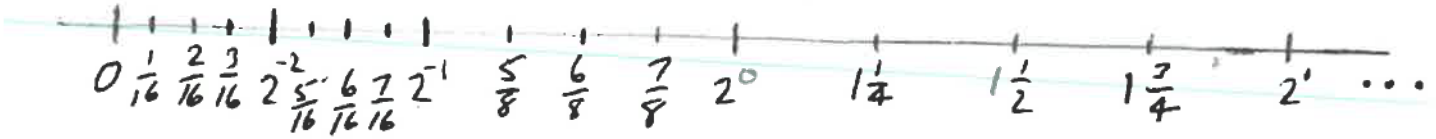
Subnormals

- **Why?**
  - Underflow vs inexact discrepancy was unacceptable
    - 1 divided by underflow produces infinity
- **Tradeoff**
  - Subnormals provide gradual underflow
    - Underflow is no worse than inexact
  - Cost is loss of precision

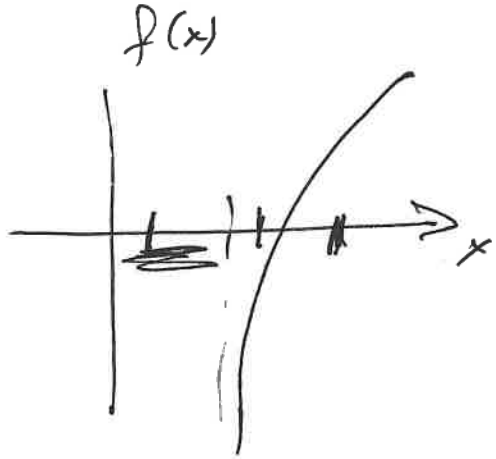
- **Without subnormals: Store zero**



- **With subnormals: Gradual underflow**



Not a Number (NaN)



- *Examples*
  - *Infinity minus infinity, infinity divided by infinity, 0 divided by 0*
  - *arcsin(2), sqroot (negative number)*
  - *A function that asymptotes*
- *It was here before IEEE Floating Point*
  - *Supercomputers had them, for example*
- *The difference:*
  - *IEEE Floating Pt allows exception handlers to be involved*
  - *Allows correction of the problem and continue processing*

Five Floating Point Exceptions

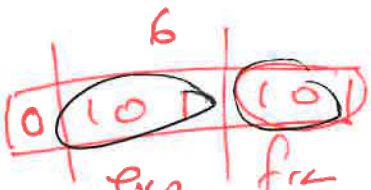
- *What are they?*
  - *Overflow: too large to represent in normalized form)*
  - *Underflow: too small to represent as a subnormal number*
  - *Inexact: not a value that can be represented exactly)*
  - *Divide by zero: function (finite arguments) → infinity*
  - *Invalid: creation of a NaN*

- *Quiet vs Signaling*
  - *Quiet: sets a sticky bit, handled under program control*
    - *For example, usual way to deal with "inexact"*
  - *Signaling: takes an exception to deal with the problem*
    - *For example, usual way to deal with "NaNs"*

Arigato!

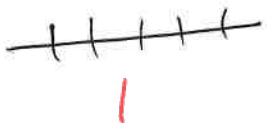
ROUNDING

$1.\text{frac} \times 2^{(\text{exp})-\text{BIAS}}$ 2^{52}



6 5/8

1^5



2^{-2}

1.101

1.1001×2^2

WOBBLE

→ 110
→ 001

1.01×2^0
$+\ 0.01 \times 2^0$
1.10×2^0
0.01×2^0
1.11×2^0
0.01×2^0
10.00×2^0

1.00×2^{-2}
0.00×2^{-2}

1.00×2^0
0.01×2^0
1.01×2^0

BIAS

EXCESS CODE 3

BINAID

1.00×2^1 1.10×2^2

4 ROUNDING

111?
110 +3
101 +2
100 +1
011 ← 0
010 −1
001 −2
000

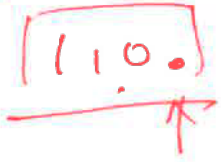
1.00×2^1

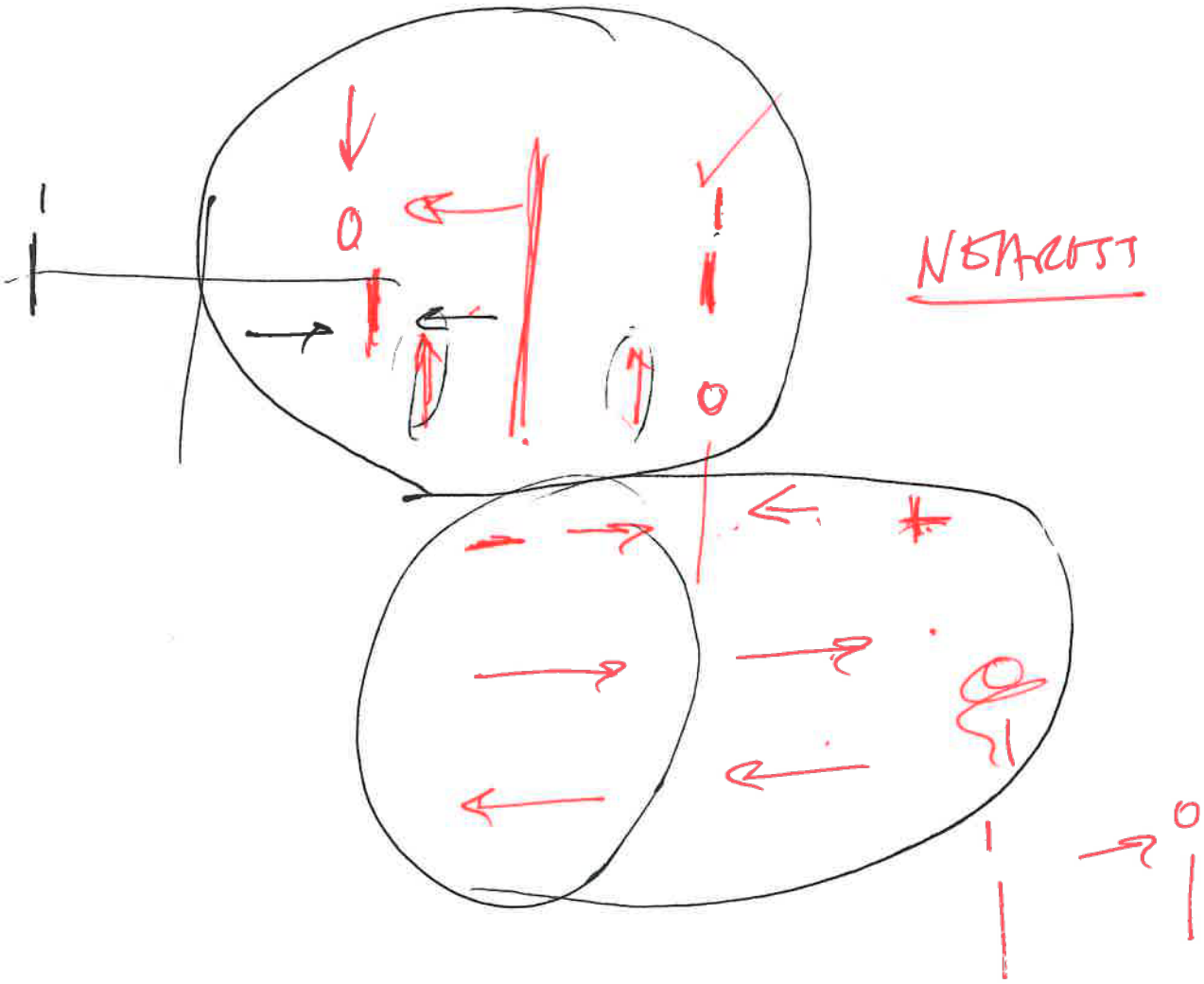
ULP

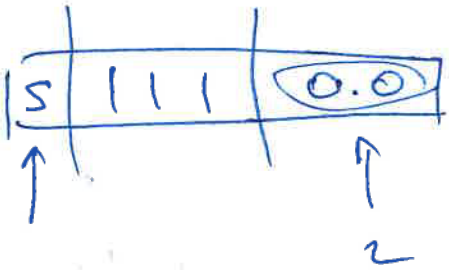
1.10×2^1
11.0

1.10×2^2

6




NOR gate



INFINITY

NaN

$\tan\left(\frac{\pi}{2}\right) = \infty$

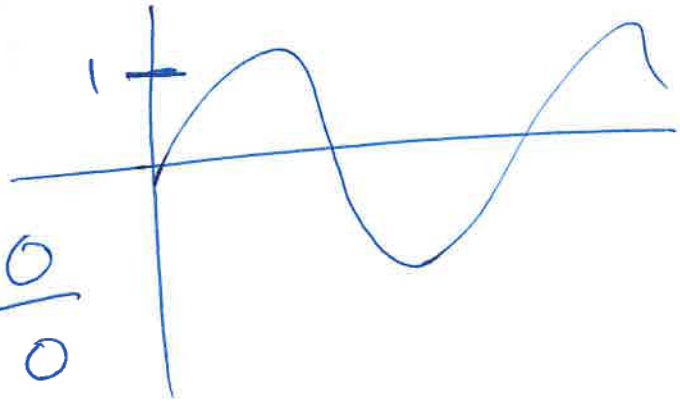
$x = 2$

$y = \arcsin(x)$

$\frac{3}{4}$

$\frac{5}{0}$

$$x = \frac{3}{4 + \frac{2}{7 + \frac{3}{0}}}$$



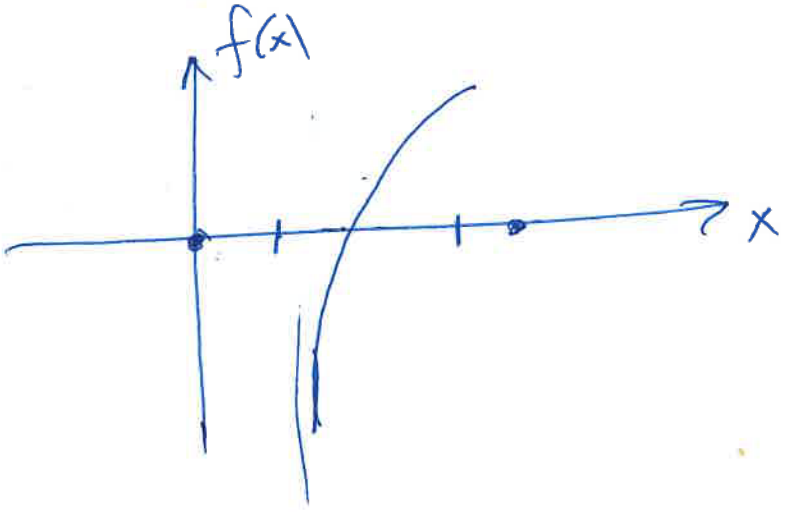
$\frac{0}{0}$

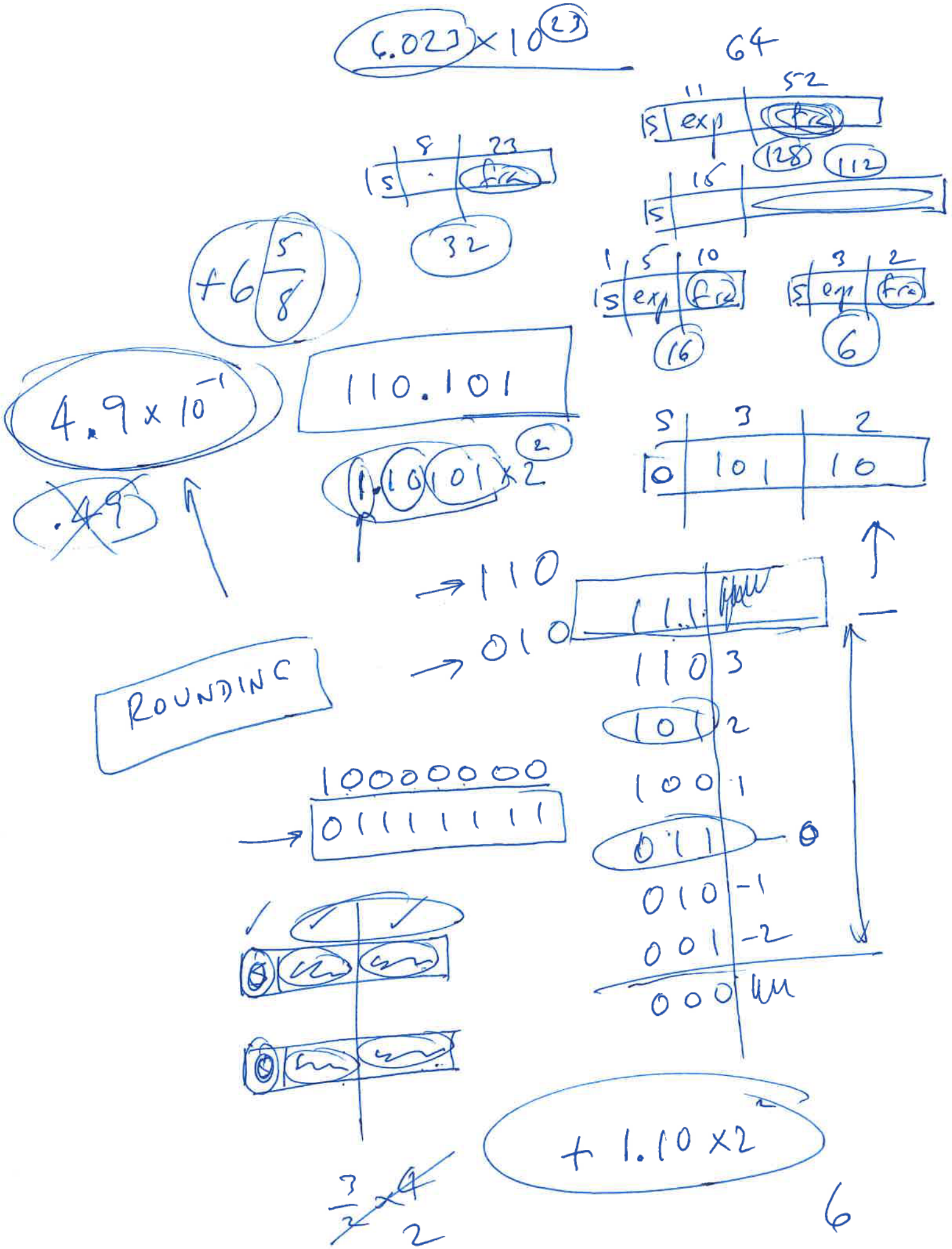
$\frac{\infty}{\infty}$

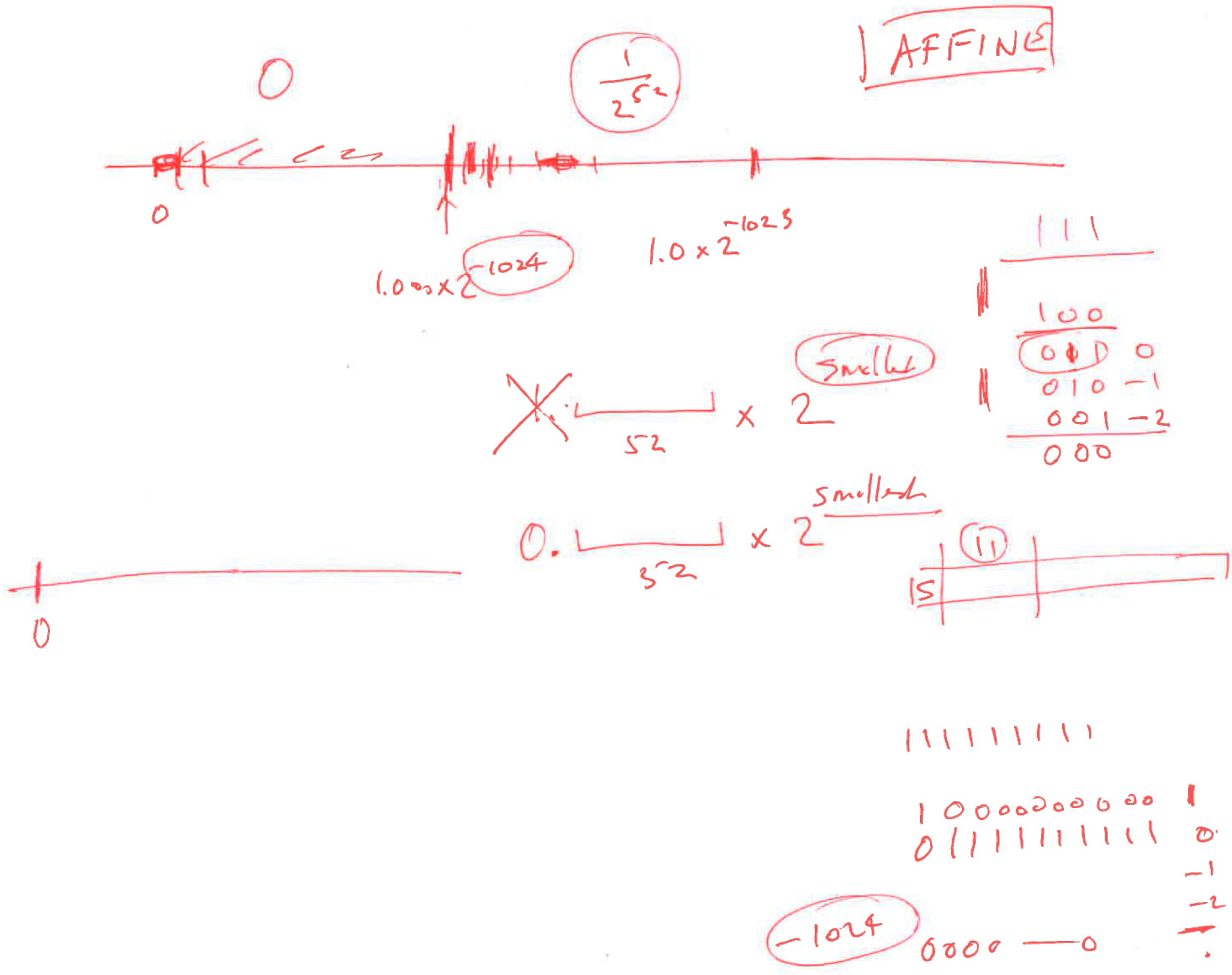
CONTINUED FRACTION EXPANSION

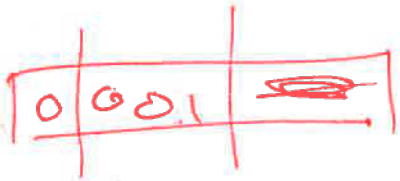
$\infty + \infty = \infty$

~~$\infty - \infty =$~~



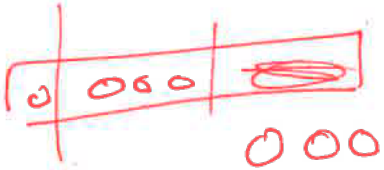

6.023 × 10^23
64
11
52
S
exp
frac
128
1023
16
S
1
5
10
S
exp
frac
16
3
2
S
exp
frac
6
8
5
32
4.9 × 10^-1
.49
110.101
1.10101 × 2
2
S
3
2
0
101
10
110
010
ROUNDING
1110
3
101
2
1001
0111
0
010
-1
001
-2
000
10000000
01111111
+ 1.10 × 2
2
3/2 × 4
2
6


AFFINE
0
$\frac{1}{2^{52}}$
0
1.0×2^{-1024}
1.0×2^{-1025}
smallest
$\times 2$
52
111
100
001 0
010 −1
001 −2
000
smallest
$0. \times 2$
52
0
11
15
11111111
1000000000 1
0111111111 0
−1
−2
−1024
0000 —— 0



$1.\bigcirc \times 2^{-2}$

001



000

$0.\bigcirc \times 2^{-2}$

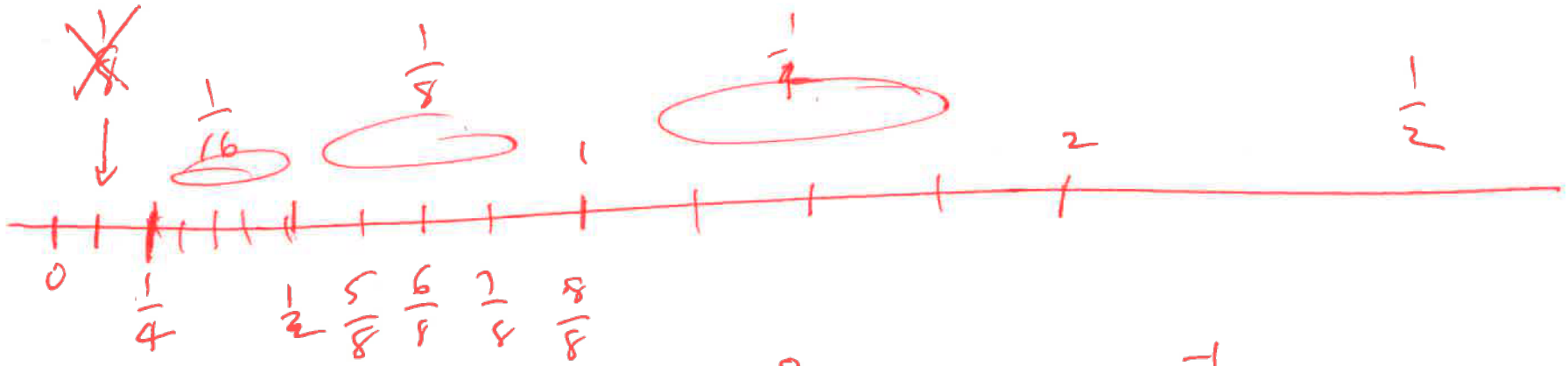
0.00×2

0.01

0.10

0.11

1.00×2



1.00×2^{0} → 1.00×2^{-1}

0.01×2^{-1}

1.01×2^{-1}

01

1.10×2^{-1}

$\frac{5}{4} \times \frac{1}{2} = \boxed{\frac{5}{8}}$

1.00×2^{-2}

1.00×2^{-3}

0.01×2^{-2}

$\frac{1}{4} \cdot \frac{1}{4}$

1.11×2^{-1}

0.01×2^{-1}

10.00×2^{-1}

$\frac{3}{2} \times \frac{1}{2} = \frac{6}{8}$

$\frac{7}{8}$

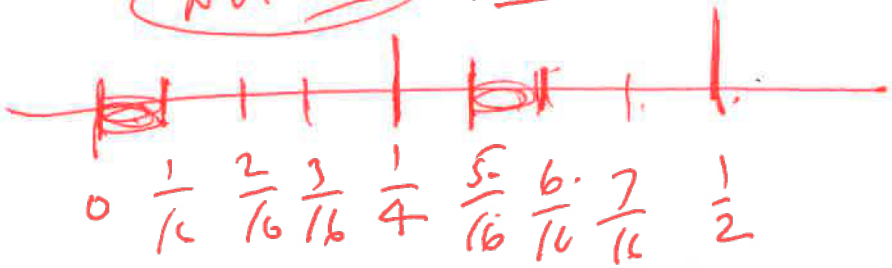
0	000	0.0

NOT NORM

NORM

1.01×2^{-2}

$0.0\,\times 2^{-2}$



1.00×2^{-2}

0.00×2^{-2}