Chapter 4
Number Representations
SKEE2263 Digital Systems

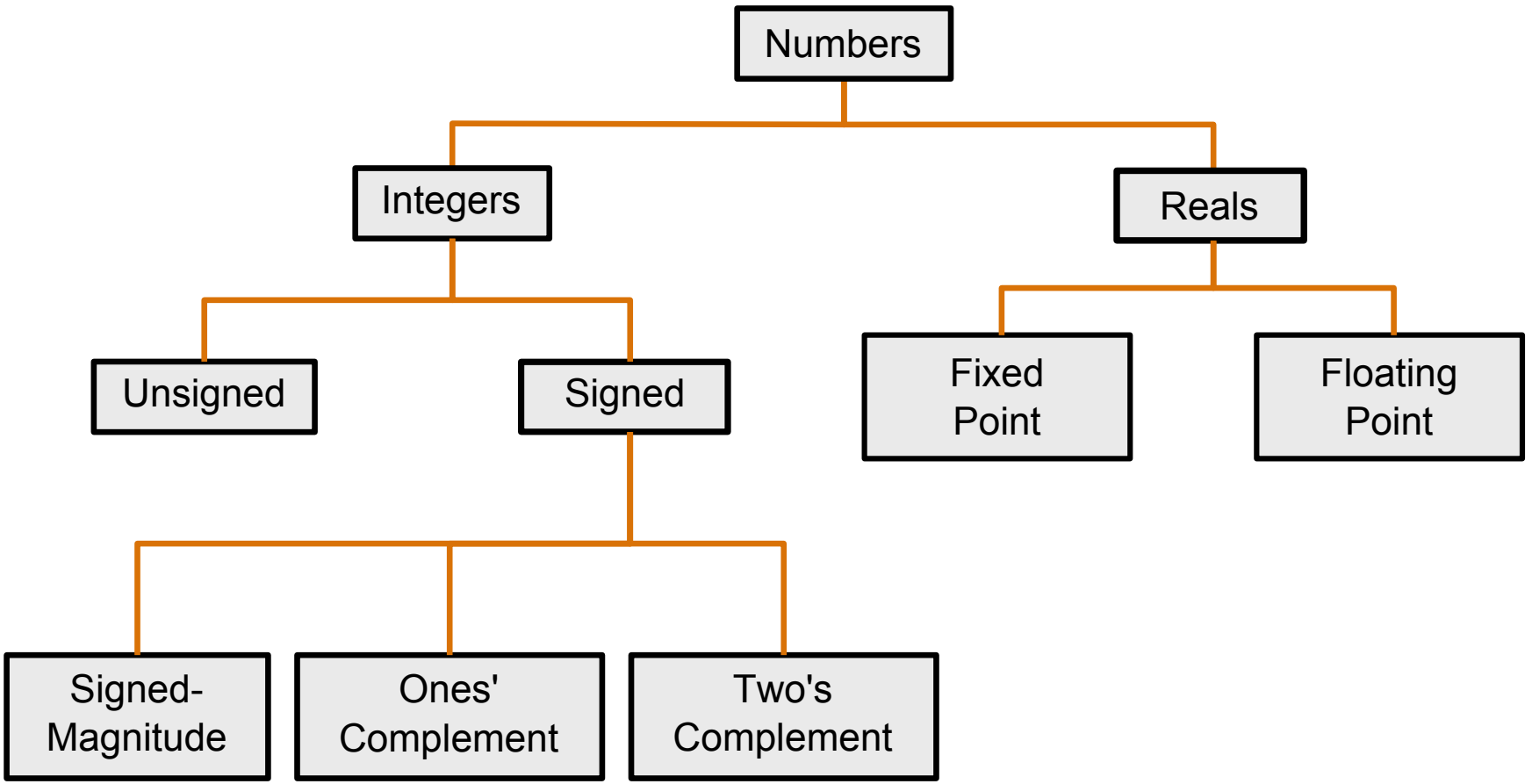
Mun'im Zabidi {munim@utm.my}
Ismahani Ismail {ismahani@fke.utm.my}
Izam Kamisian {e-izam@utm.my}

Faculty of Electrical Engineering, Universiti Teknologi Malaysia

February 15, 2018

Table of Contents

Taxonomy of Number Systems



Integers

Number of bits	Number of values	Machine
4	16	Intel 4004
8	256	8080, 6800
16	65536	PDP11, 8086, 68000
32	4×10^9	68020, VAX11, IEEE single
48	1×10^{14}	Unisys
64	1.8×10^{19}	Cray, IEEE double

Integers

Value for integer bit pattern:

$$V_{\text{unsigned}} = \sum_{i=0}^{N-1} b_i \times 2^i$$

Example 10110_2:

$$\begin{aligned} 10110_2 &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 22_{10} \end{aligned}$$

Signed-Magnitude

Value for N-bit signed-magnitude pattern is:

$$V_{\text{SM}} = (-1)^{b_{N-1}} \times \sum_{i=0}^{N-2} b_i \times 2^i$$

Example 1010_{SM}:

$$\begin{aligned} V_{\text{SM}} &= (-1)^{b_3} \times [b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0] \\ &= (-1)^1 \times [0(4) + 1(2) + 0(1)] \\ &= -1 \times 2 \\ &= -2 \end{aligned}$$

Signed-Magnitude

Signed Integer	Signed-Magnitude
+127	0111 1111
+126	0111 1110
...	...
+2	0000 0010
+1	0000 0001
+0	0000 0000
-0	1000 0000
-1	1000 0001
-2	1000 0010
...	...
-126	1111 1110
-127	1111 1111

Ones' Complement

Value for N-bit ones' complement pattern is:

$$V_{1C} = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i + b_{N-1}$$

Example 1010_{1C}:

$$\begin{aligned} V_{1C} &= -b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0 + b_3 \\ &= -1(8) + 0(4) + 1(2) + 0(1) + 1 \\ &= -8 + 2 + 1 \\ &= -5 \end{aligned}$$

Ones' Complement

Signed Integer	Ones' Complement
+127	0111 1111
+126	0111 1110
...	...
+2	0000 0010
+1	0000 0001
0	0000 0000
	1111 1111
-1	1111 1110
-2	1111 1101
-3	1111 1100
...	...
-126	1000 0001
-127	1000 0000

Two's Complement

Value for N-bit Two's complement pattern is:

$$V_{2C} = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i$$

Example 1010_{2C}:

$$\begin{aligned} V_{2C} &= -b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0 \\ &= -1(8) + 0(4) + 1(2) + 0(1) \\ &= -8 + 2 \\ &= -6 \end{aligned}$$

Twos' Complement

Signed Integer	Twos' Complement
+127	0111 1111
+126	0111 1110
...	...
+2	0000 0010
+1	0000 0001
0	0000 0000
-1	1111 1111
-2	1111 1110
...	...
-126	1000 0010
-127	1000 0001
-128	1000 0000

Sign Extension

- convert a number to a larger format.
- just copy the sign bit to fill the new “high order” bits

+ 100 in 8-bit two’s-complement binary	**0**110 0100
+ 100 in 16-bit two’s-complement binary	**0000 0000 0**110 0100
- 100 in 8-bit two’s-complement binary	**1**001 1100
- 100 in 16-bit two’s-complement binary	**1111 1111 1**001 1100

Offset binary

- a.k.a. biased-K representation
- Variation of two's complement
- Uses a value K as biasing value

Applications:

- Exponent of floating-point number (biased-127 or biased-1023)
- Analog interfacing
- Excess-3 code (actual value = binary - 3)

Comparing Number Systems

Decimal	Signed-Magnitude	One's Complement	Two's Complement	Offset Binary
7	0111	0111	0111	1111
6	0110	0110	0110	1110
5	0101	0101	0101	1101
4	0100	0100	0100	1100
3	0011	0011	0011	1011
2	0010	0010	0010	1010
1	0001	0001	0001	1001
0	0000	0000	0000	1000
-0	1000	1111	–	–
-1	1001	1110	1111	0111
-2	1010	1101	1110	0110
-3	1011	1100	1101	0101
-4	1100	1011	1100	0100
-5	1101	1010	1011	0011
-6	1110	1001	1010	0010
-7	1111	1000	1001	0001
-8	–	–	1000	0000

Signed Systems Compared

	Unsigned	Signed-Magnitude	Ones' Complement	Two's Complement
Smallest	0	$-(2^{n-1}-1)$	$-(2^{n-1}-1)$	-2^{n-1}
Largest	2^n-1	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$

Real Numbers

Number System	Format	Characteristics
Fixed-point	$\pm i.f$	Low-precision
Rational	$\pm p/q$	Difficult to work with
Floating-point	$\pm m \cdot b^e$	Most common way to handle reals

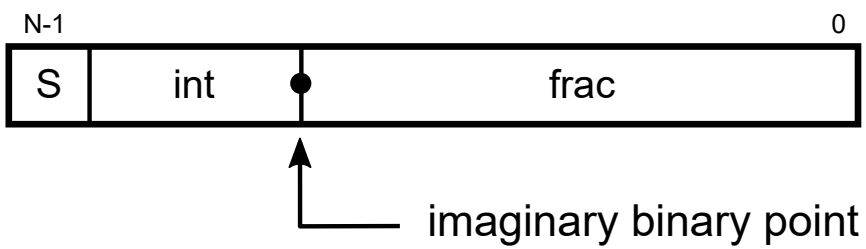
Fixed-Point Numbers: General

The general expression for an N-bit fixed point 2's complement

$$x = \frac{-b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i}{2^f}$$

where:

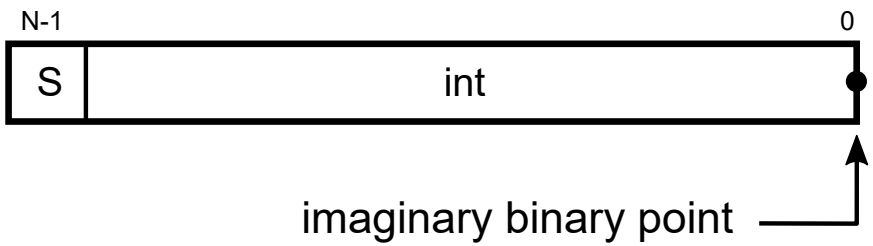
- N = total #bits
- f = #bits in fraction ($0 \leq f \leq N-1$)



Fixed Point-Numbers: Two's Complement

Same expression as before but $f = 0$

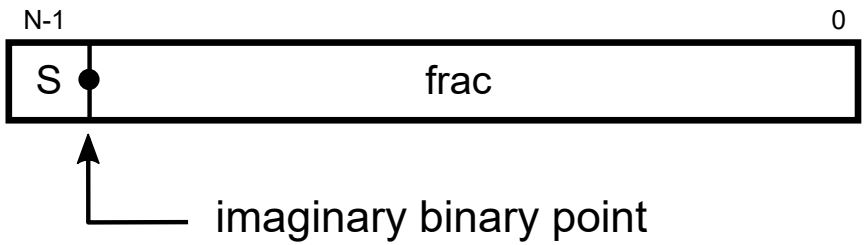
$$x = \frac{-b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i}{2^0}$$
$$= -b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \cdots + b_1 2^1 + b_0 2^0$$



Fixed-Point Numbers: Q-Format

$f = N - 1$, No integer part

$$x = \frac{-b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i}{2^{N-1}}$$
$$= -b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-(N-1)}2^{-(N-1)}$$



$N = 8, f = 4$

Weights	-2^3	2^2	2^1	2^0	$\cdot$	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Bit value	0	1	0	1	$\cdot$	1	1	0	0

$$\begin{aligned} 0101.1100_2 &= 2^2 + 2^0 + 2^{-1} + 2^{-2} \\ &= 4 + 1 + 0.5 + 0.25 \\ &= 5.75_{10} \end{aligned}$$

OR

Weights	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Bit value	0	1	0	1	1	1	0	0

$\div 2^4$

$$\begin{aligned} x &= (2^6 + 2^4 + 2^3 + 2^2) \div 2^4 \\ &= (64 + 16 + 8 + 4) \div 16 \\ &= 5.75_{10} \end{aligned}$$

$N = 8, f = 7 \rightarrow Q7 format$

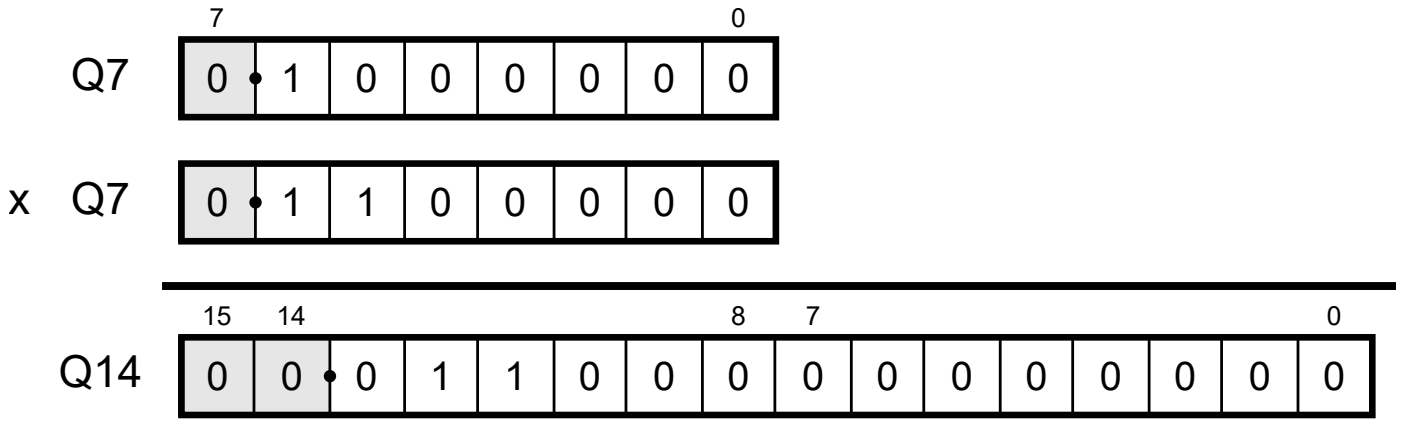
Weights	-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Bit value	0	1	1	1	1	1	1	1

$$\begin{aligned} max &= 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} \\ &= (2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0) \div 2^7 \\ &= 127/128 \\ &= 0.9921875 \end{aligned}$$

Weights	-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Bit value	1	0	0	0	0	0	0	0

$$\begin{aligned} min &= -2^0 \\ &= -1 \end{aligned}$$

Q7 multiplication



In Q7:

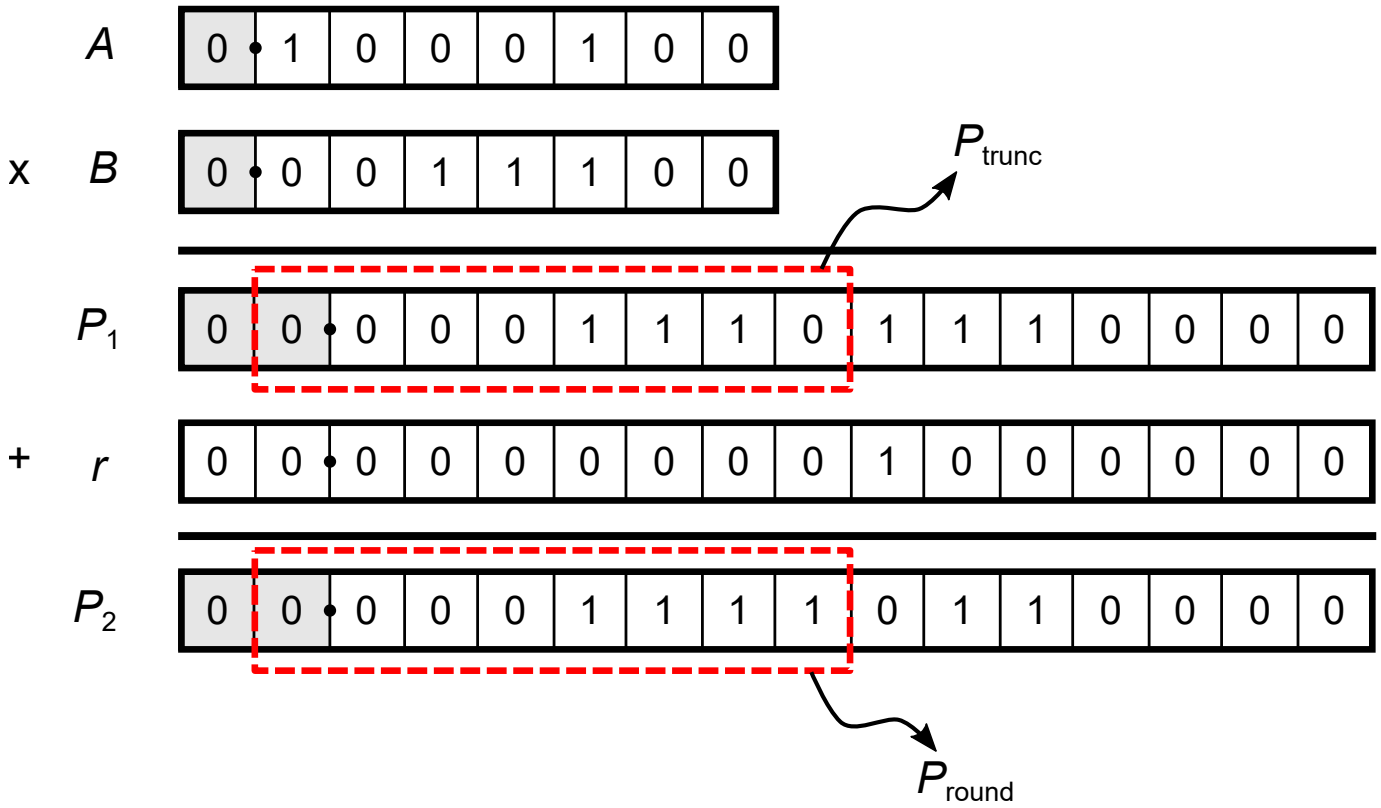
- Multiplicand: $0.5_{10} = 0.1_2$
- Multiplier: $0.75_{10} = 0.11_2$
- Result: $0.375_{10} = 0.011_2$

In hardware:

- Multiplicand: 0x40 = 64_{10}
- Multiplier: 0x60 = 96_{10}
- Result = 0x1800 = 6144_{10}

$\therefore$ to stay in Q7 system (maintain 8 most useful bits), keep bits 14:7 only

Inexact Q7 Result Bits



Truncating:

- Result = 0.0001110

Rounding:

- Result = 0.0001111

Truncate or Round?

$$\%\text{error} = \frac{|\ \text{Exact value} - \text{Approximate value}\ |}{\text{Exact value}} \times 100$$

- Truncating takes 8 top bits as is

$$\begin{aligned}\%\text{error} &= \frac{|\ 0.1162109375 - 0.109375\ |}{0.1162109375} \times 100 \\ &= 5.9\%\end{aligned}$$

- Rounding add rounding factor r before tuncating

$$\begin{aligned}\%\text{error} &= \frac{|\ 0.1162109375 - 0.1171875\ |}{0.1162109375} \times 100 \\ &= 0.8\%\end{aligned}$$

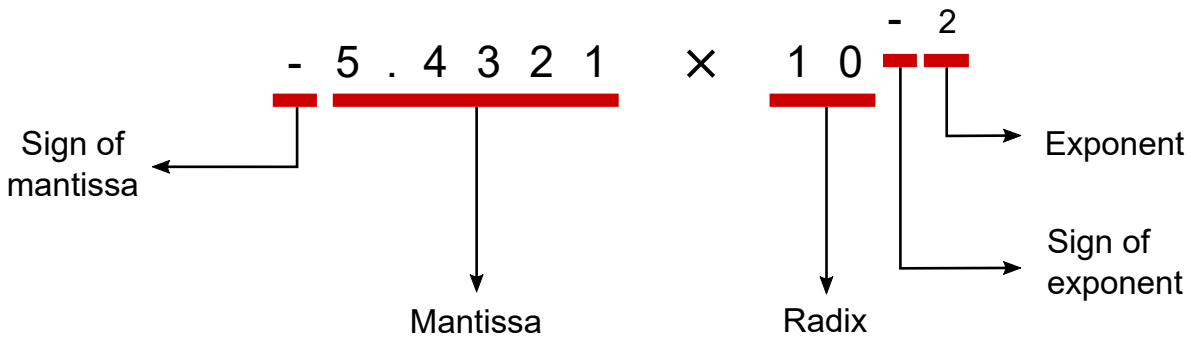
Rounding reduces error but needs extra step.

What is Floating Point?

$$
\begin{aligned}
5.25_{10} &= 101.01 \times 2^0 \\
&= 10.101 \times 2^1 \\
&= 1.0101 \times 2^2 \leftarrow \\
&= 0.10101 \times 2^3
\end{aligned}
$$

- Binary point “floats” to a pre-defined position
- Process is called **normalization**

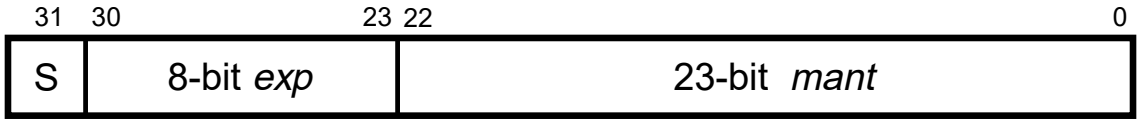
Floating Point Parts



$$\pm X = m \times b^e$$

where m = mantissa, b = number base and e = exponent.

IEEE Single-Precision Format



$$\pm X = (-1)^s \times 1.m \times 2^{e-127}$$

Sign field: 0 for positive numbers ($-1^0 = +1$)
1 for negative numbers ($-1^1 = -1$).

Exponent field: Unsigned 8 bit, biased-127.

Mantissa field: Bits to the right of normalized binary number.

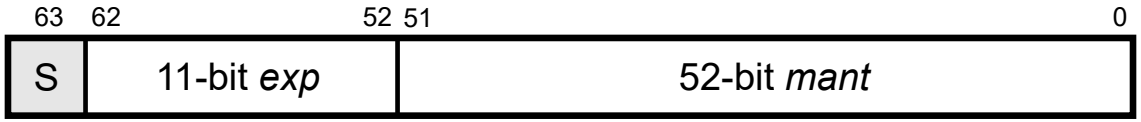
IEEE Single-Precision Format

1	1000 0001	110 0000 0000 0000 0000 0000

$$
\begin{aligned}
X &= (-1)^1 \times 1.11_2 \times 2^{129-127} \\
&= -1 \times 1.75_{10} \times 2^2 \\
&= -7
\end{aligned}
$$

IEEE Double-Precision Format

Double precision is more common.



$$\pm X = (-1)^s \times 1.m \times 2^{e-1023}$$

```
public class TryFP {
   public static void main(String[ ] args) {
     double d = 1/3.; // Java likes double-prec more
     float f = 1f/3f; // Must force use of single-prec
     System.out.println("Value of d="+d);
     System.out.println("Value of f="+f);
   }
}
Value of d=0.3333333333333333
Value of f=0.33333334
```

FX vs FP

Fixed Point Arithmetic	**Floating-Point Arithmetic**
Simple circuit	Complex circuit (due to rounding and normalization)
Small area and faster	Large area and slower
Less accurate (the result is truncated if it exceeds the size)	More accurate (high precision)
Smaller range of values can be handled	Wider range of values can be handled