CSC 252: Computer Organization
Spring 2023: Lecture 4

Instructor: Yuhao Zhu

Department of Computer Science
University of Rochester

Announcement

- Programming Assignment 1 is out
  - Details: https://www.cs.rochester.edu/courses/252/spring2023/labs/assignment1.html
  - Due on Jan. 27, 11:59 PM
  - You have 3 slip days

15	16	17	18	19	20 **Today**	21
22	23	24	25	26	27 **Due**	28

Announcement

- Programming assignment 1 is in C language. Seek help from TAs.
- TAs are best positioned to answer your questions about programming assignments!!!
- Programming assignments do NOT repeat the lecture materials. They ask you to synthesize what you have learned from the lectures and work out something new.

Multiplication

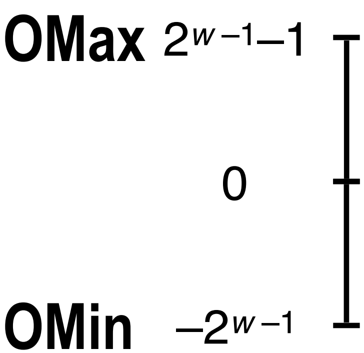
Multiplication

- Goal: Computing Product of w-bit numbers x, y

Multiplication

- Goal: Computing Product of *w*-bit numbers *x*, *y*

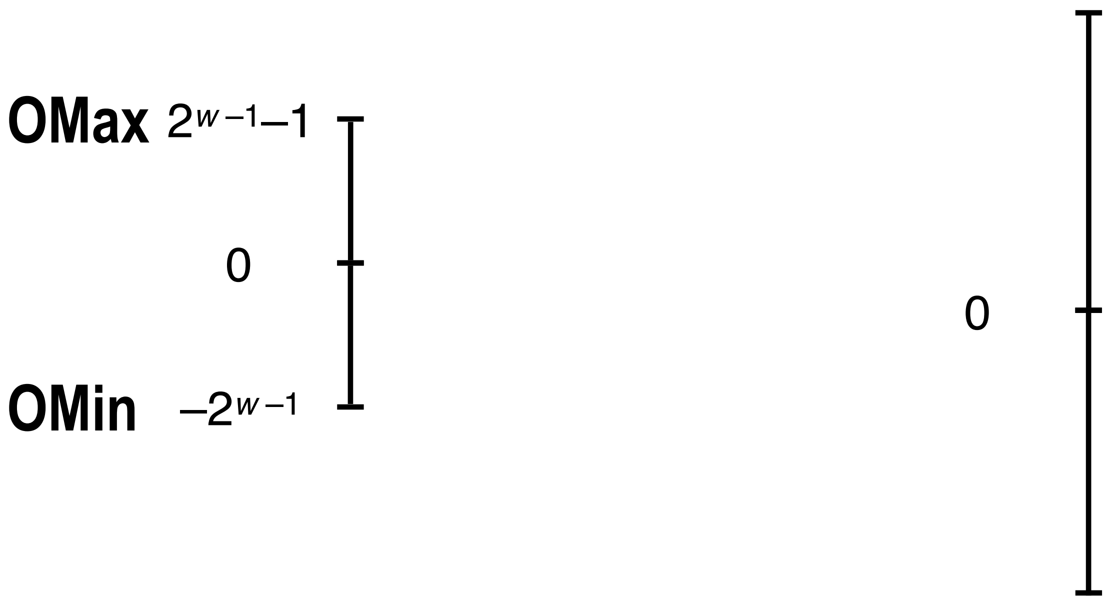
Original Number (w bits)



Multiplication

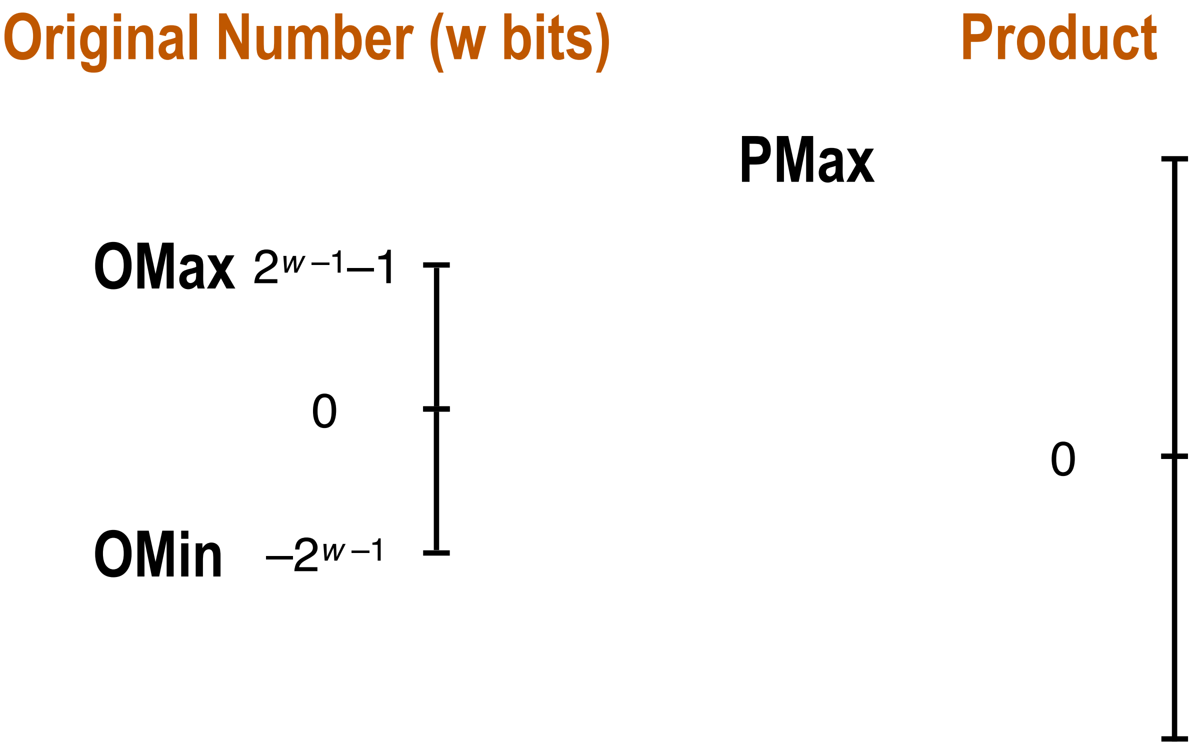
- Goal: Computing Product of *w*-bit numbers *x*, *y*

Original Number (w bits) **Product**



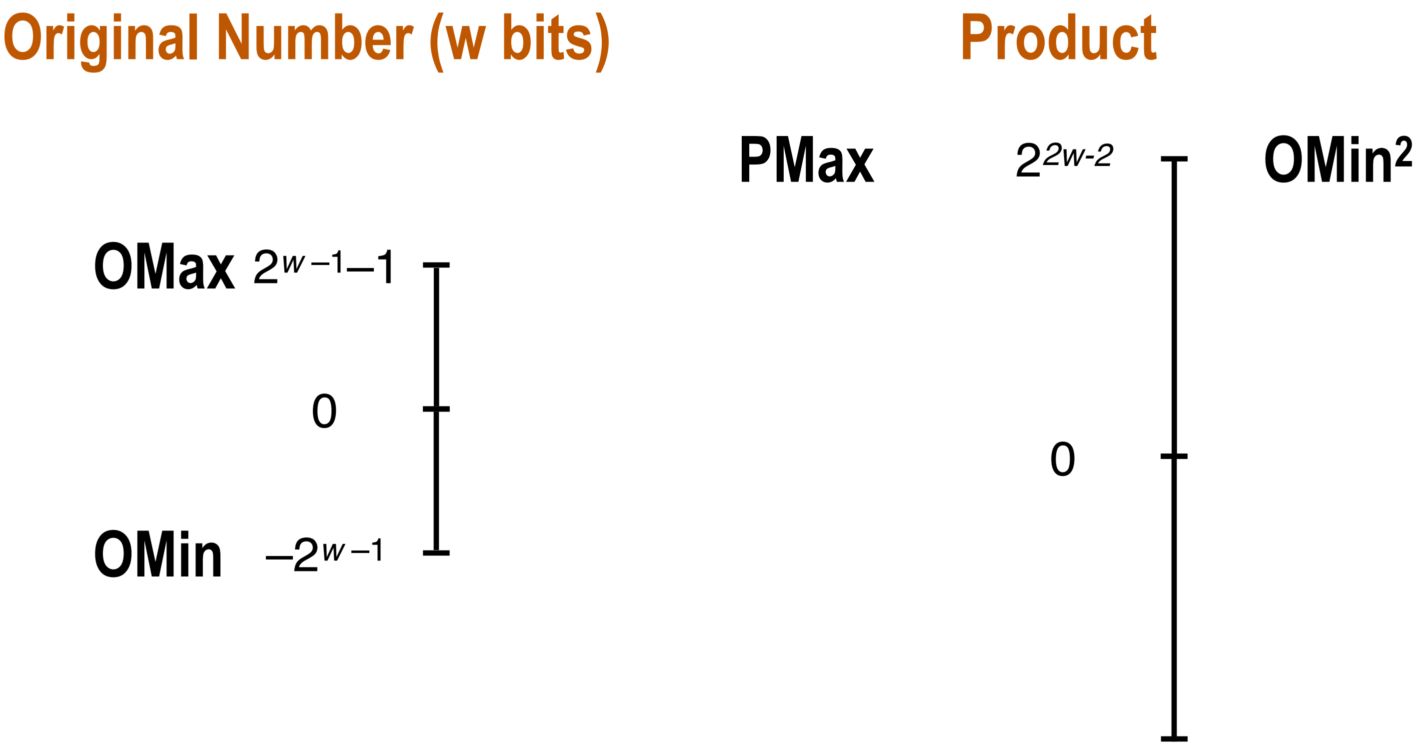
Multiplication

- Goal: Computing Product of *w*-bit numbers *x*, *y*



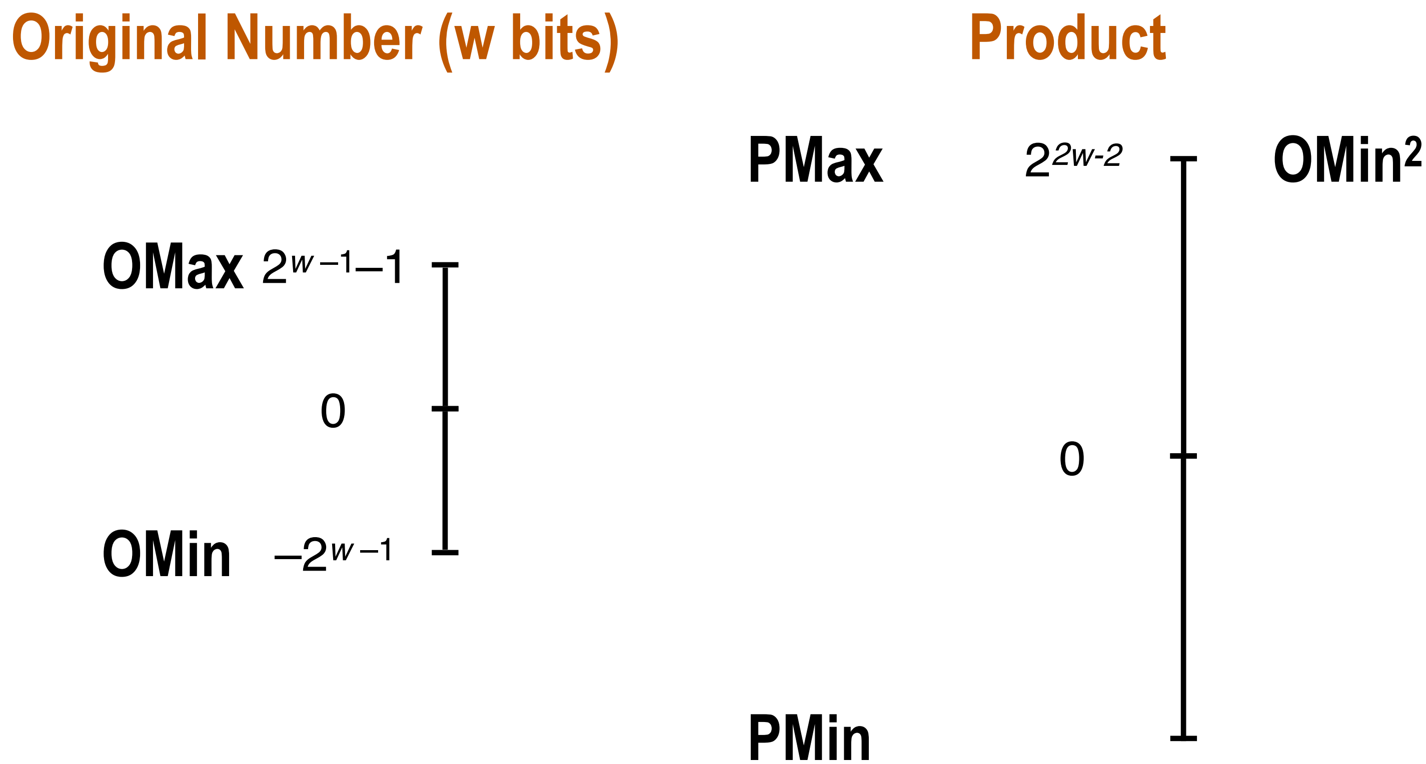
Multiplication

- Goal: Computing Product of w-bit numbers x, y



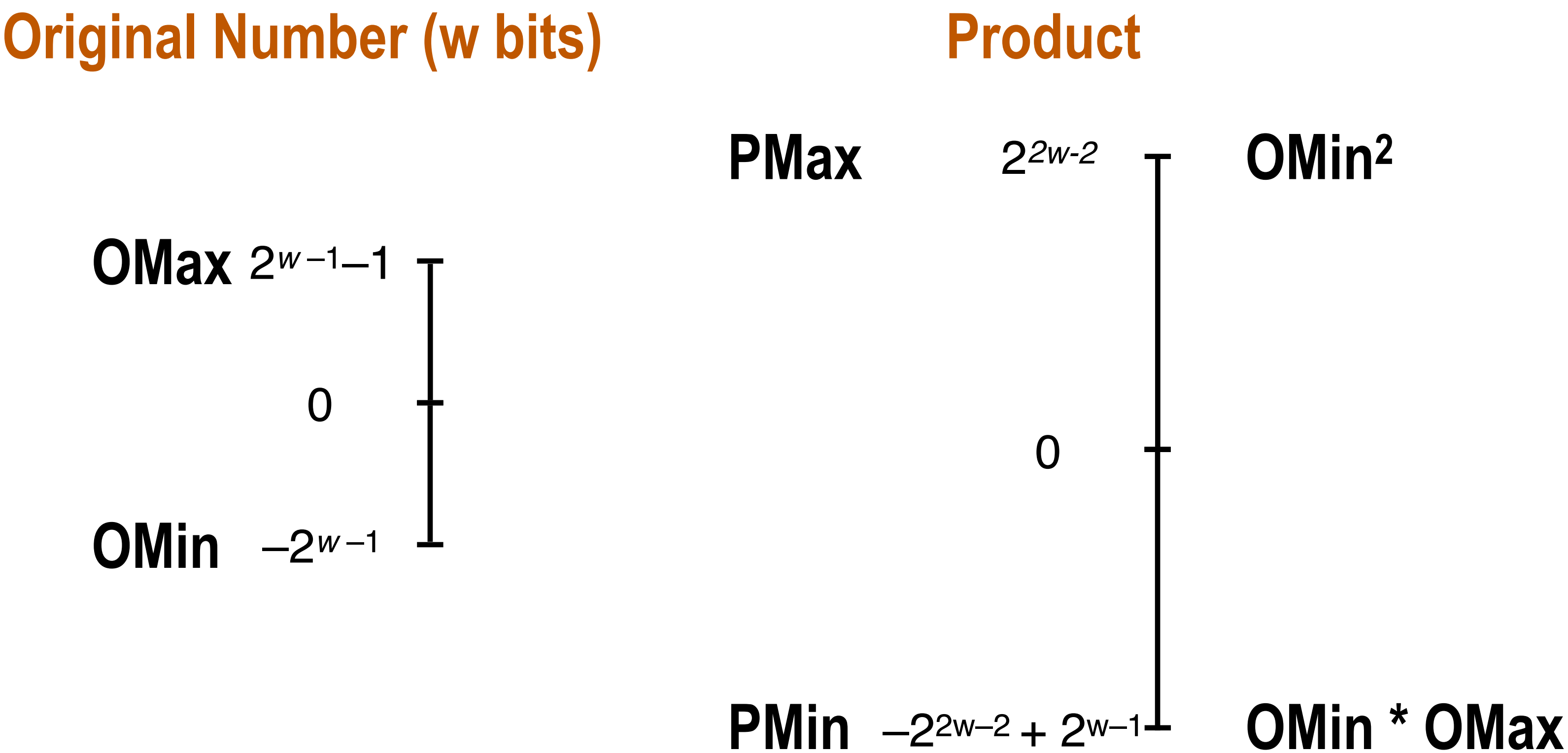
Multiplication

- Goal: Computing Product of *w*-bit numbers *x*, *y*



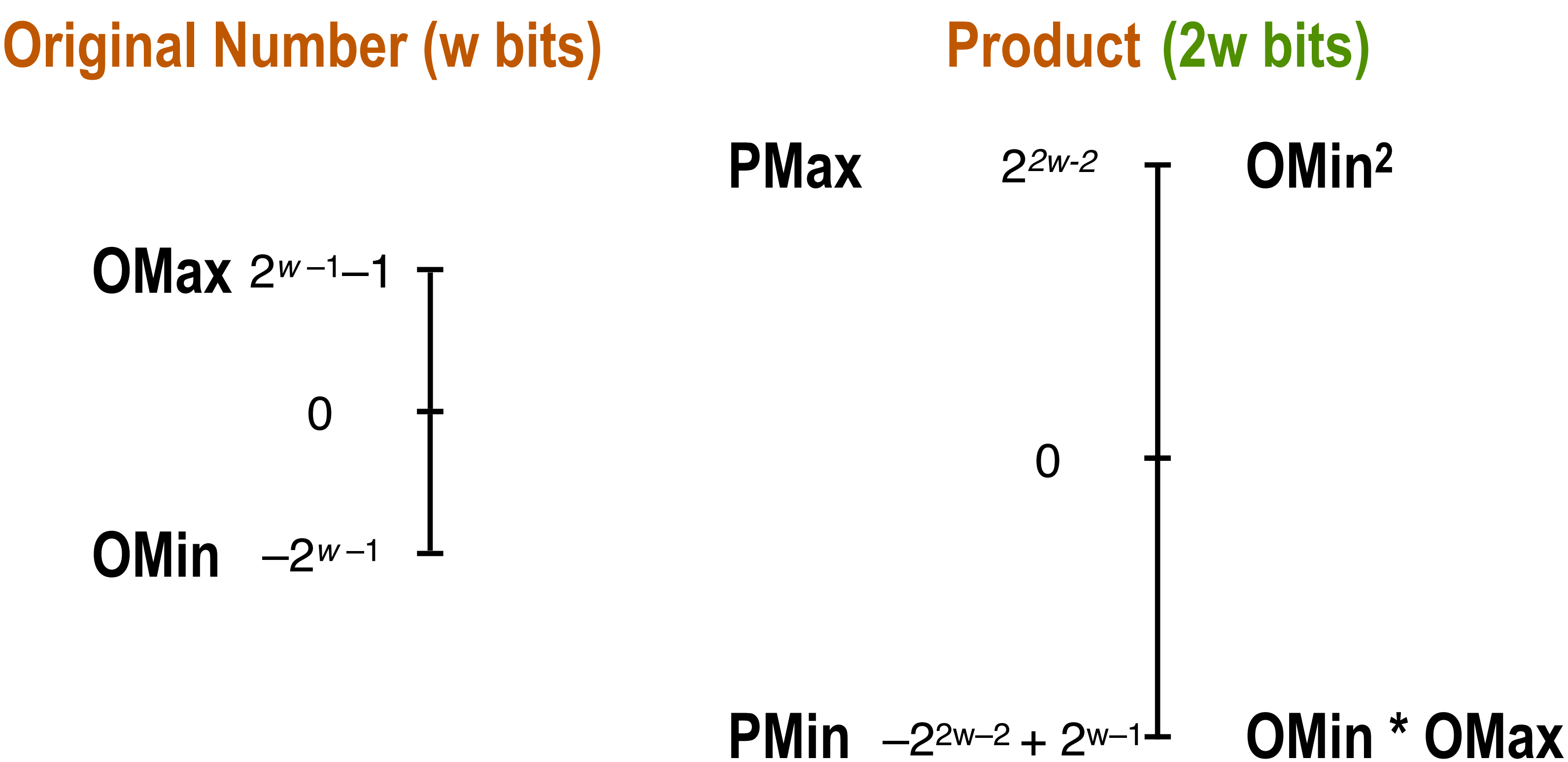
Multiplication

- Goal: Computing Product of *w*-bit numbers *x*, *y*



Multiplication

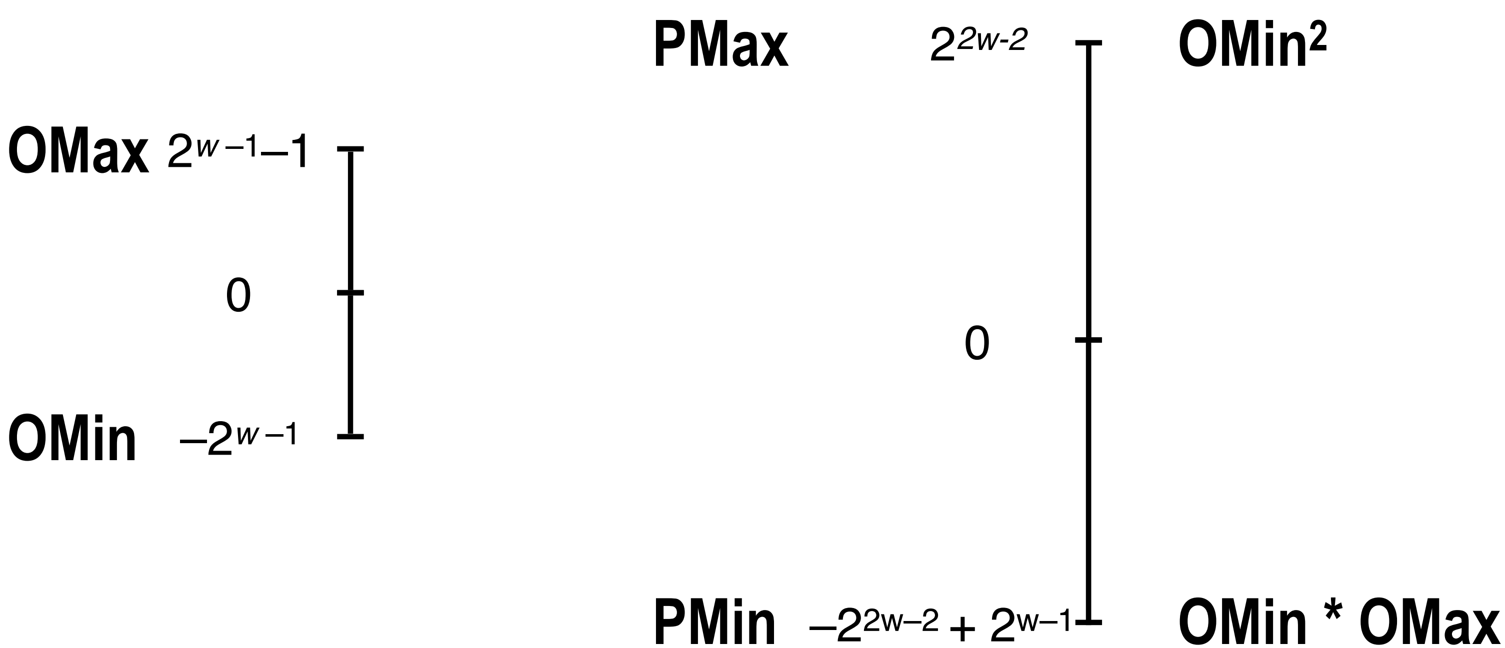
- Goal: Computing Product of *w*-bit numbers *x*, *y*



Multiplication

- Goal: Computing Product of *w*-bit numbers *x*, *y*
- Exact results can be bigger than *w* bits
  - Up to *2w* bits (both signed and unsigned)

Original Number (w bits) **Product (2w bits)**

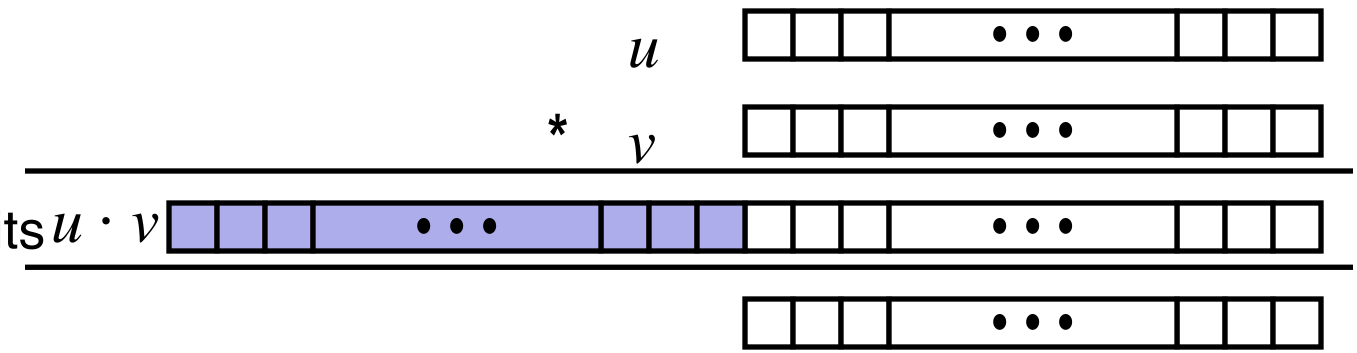


Unsigned Multiplication in C

Operands: *w* bits

True Product: 2*w bits

Discard *w* bits: *w* bits



- Standard Multiplication Function
  - Ignores high order *w* bits
- Effectively Implements the following:

  $\text{UMult}_w(u, v) = u \cdot v \mod 2^w$

Today: Floating Point

- Background: Fractional binary numbers and fixed-point
- Floating point representation
- IEEE 754 standard
- Rounding, addition, multiplication
- Floating point in C
- Summary

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
    - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = \boxed{1*10^1} + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + \boxed{2*10^0} + 4*10^{-1} + 5*10^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^{1} + 2*10^{0} + \boxed{4*10^{-1}} + 5*10^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + \boxed{5*10^{-2}}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = 1*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = \boxed{1*2^1} + 0*2^0 + 0*2^{-1} + 1*2^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^{1} + 2*10^{0} + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = 1*2^{1} + \boxed{0*2^{0}} + 0*2^{-1} + 1*2^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = 1*2^1 + 0*2^0 + \boxed{0*2^{-1}} + 1*2^{-2}$$

Recall: Represent Fractions in Binary

- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = 1*2^1 + 0*2^0 + 0*2^{-1} + \boxed{1*2^{-2}}$$

Recall: Represent Fractions in Binary

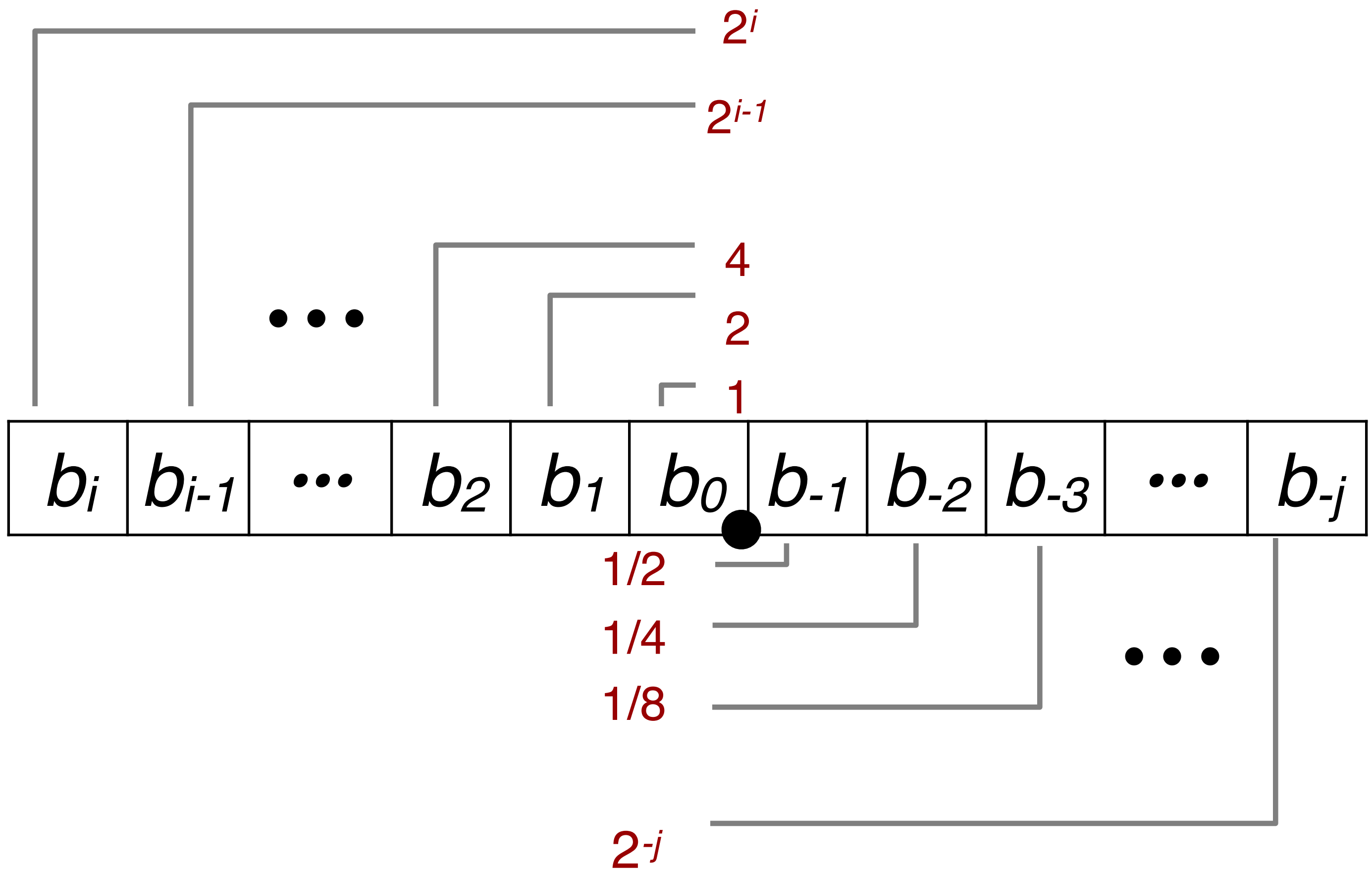
- What does 10.01_2 mean?
  - C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$

$$10.01_2 = 1*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2}$$

$$= 2.25_{10}$$

Fractional Binary Numbers



Fixed-Point Representation

Fixed-Point Representation

- Binary point stays fixed

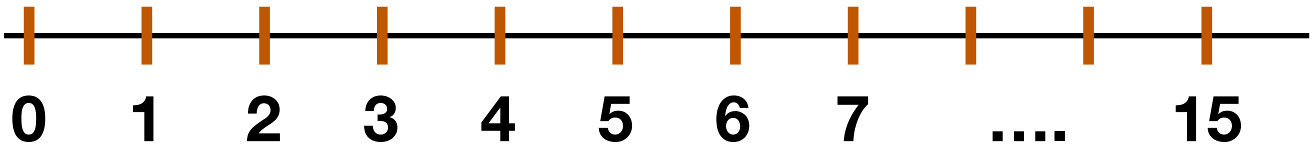
Fixed-Point Representation

- Binary point stays fixed

Decimal	Binary
0	0000.
1	0001.
2	0010.
3	0011.
4	0100.
5	0101.
6	0110.
7	0111.
8	1000.
9	1001.
10	1010.
11	1011.
12	1100.
13	1101.
14	1110.
15	1111.

Fixed-Point Representation

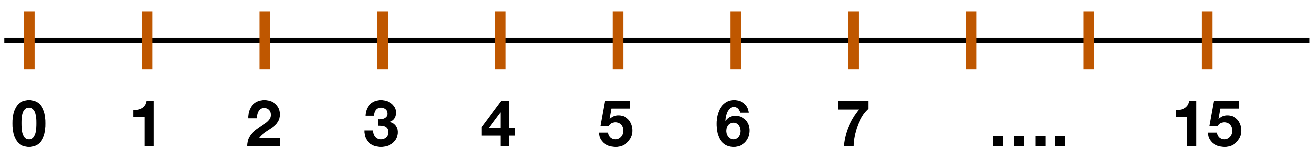
- Binary point stays fixed



Decimal	Binary
0	0000.
1	0001.
2	0010.
3	0011.
4	0100.
5	0101.
6	0110.
7	0111.
8	1000.
9	1001.
10	1010.
11	1011.
12	1100.
13	1101.
14	1110.
15	1111.

Fixed-Point Representation

- Binary point stays fixed



Decimal	Binary
0	00.00
0.25	00.01
0.5	00.10
0.75	00.11
1	01.00
1.25	01.01
1.5	01.10
1.75	01.11
2	10.00
2.25	10.01
2.5	10.10
2.75	10.11
3	11.00
3.25	11.01
3.5	11.10
3.75	11.11

Fixed-Point Representation

- Binary point stays fixed

Decimal	Binary
0	00.00
0.25	00.01
0.5	00.10
0.75	00.11
1	01.00
1.25	01.01
1.5	01.10
1.75	01.11
2	10.00
2.25	10.01
2.5	10.10
2.75	10.11
3	11.00
3.25	11.01
3.5	11.10
3.75	11.11

Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - The interval in this example is 0.25_{10}

Decimal	Binary
0	00.00
0.25	00.01
0.5	00.10
0.75	00.11
1	01.00
1.25	01.01
1.5	01.10
1.75	01.11
2	10.00
2.25	10.01
2.5	10.10
2.75	10.11
3	11.00
3.25	11.01
3.5	11.10
3.75	11.11

Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - The interval in this example is 0.25_{10}

- Still need to remember the binary point, but just once for all numbers, which is implicit given the data type

Decimal	Binary
0	00.00
0.25	00.01
0.5	00.10
0.75	00.11
1	01.00
1.25	01.01
1.5	01.10
1.75	01.11
2	10.00
2.25	10.01
2.5	10.10
2.75	10.11
3	11.00
3.25	11.01
3.5	11.10
3.75	11.11

Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - The interval in this example is 0.25_{10}

- Still need to remember the binary point, but just once for all numbers, which is implicit given the data type
- Usual arithmetics still work
  - No need to align (already aligned)

Decimal	Binary
0	00.00
0.25	00.01
0.5	00.10
0.75	00.11
1	01.00
1.25	01.01
1.5	01.10
1.75	01.11
2	10.00
2.25	10.01
2.5	10.10
2.75	10.11
3	11.00
3.25	11.01
3.5	11.10
3.75	11.11

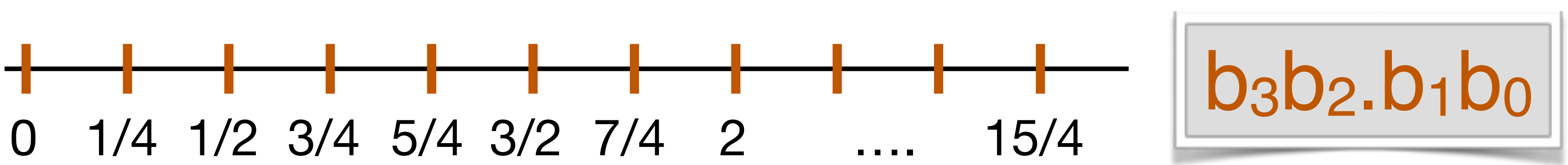
Limitations of Fixed-Point (#*1*)

Limitations of Fixed-Point (#*1*)

- Can exactly represent numbers only of the form $x/2^k$

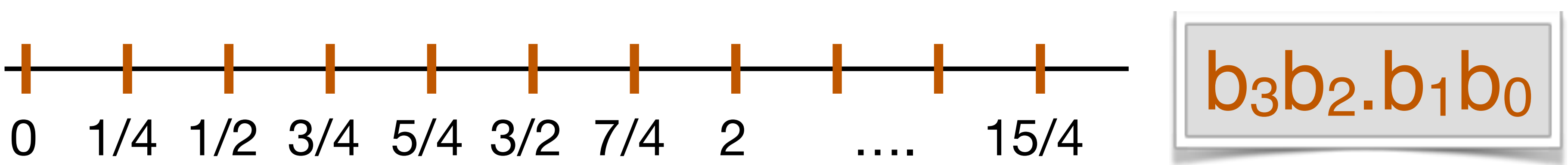
Limitations of Fixed-Point (#*1*)

- Can exactly represent numbers only of the form $x/2^k$



Limitations of Fixed-Point (#*1*)

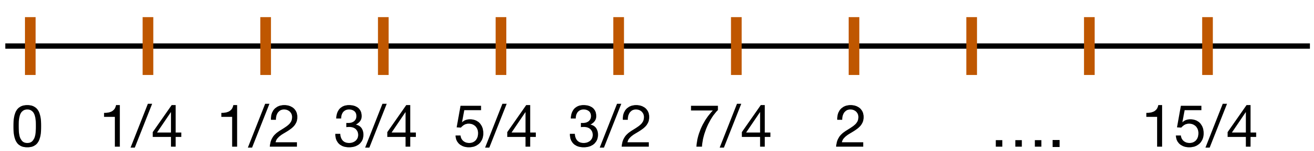
- Can exactly represent numbers only of the form $x/2^k$
  - Other rational numbers have repeating bit representations



Limitations of Fixed-Point (#*1*)

- Can exactly represent numbers only of the form $x/2^k$
    - Other rational numbers have repeating bit representations

Decimal Value	**Binary Representation**
1/3	0.0101010101[01]…
1/5	0.001100110011[0011]…
1/10	0.0001100110011[0011]…



$b_3b_2.b_1b_0$

Limitations of Fixed-Point (#2)

Limitations of Fixed-Point (#2)

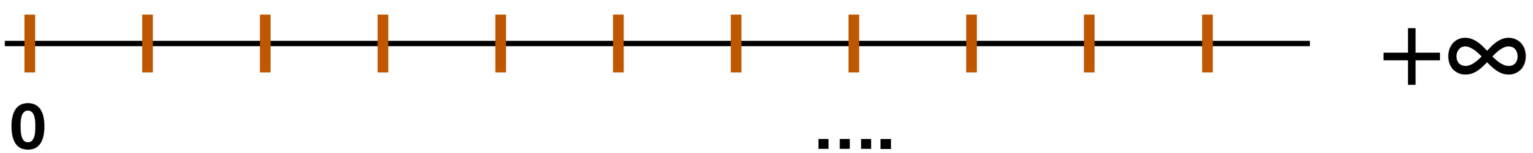
- Can’t represent very small and very large numbers at the same time

Limitations of Fixed-Point (#2)

- Can’t represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers

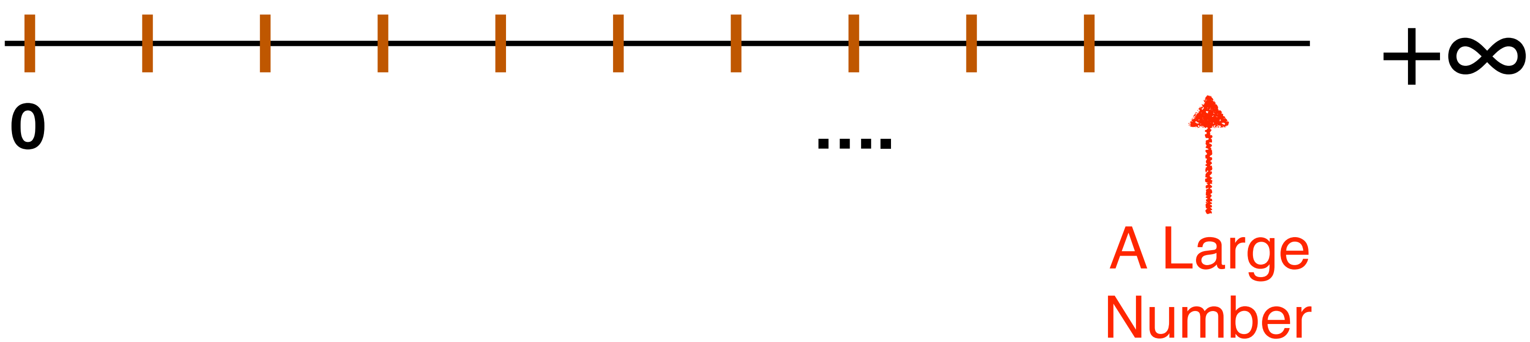
Limitations of Fixed-Point (#2)

- Can't represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers



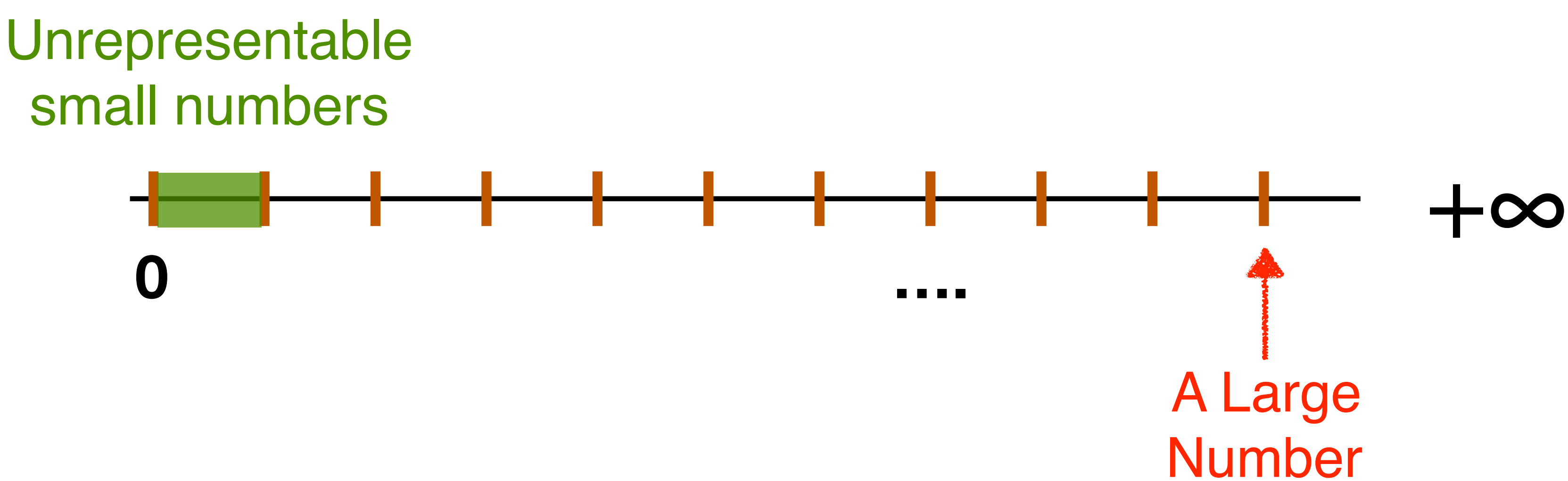
Limitations of Fixed-Point (#2)

- Can't represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers



Limitations of Fixed-Point (#2)

- Can't represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers



Limitations of Fixed-Point (#2)

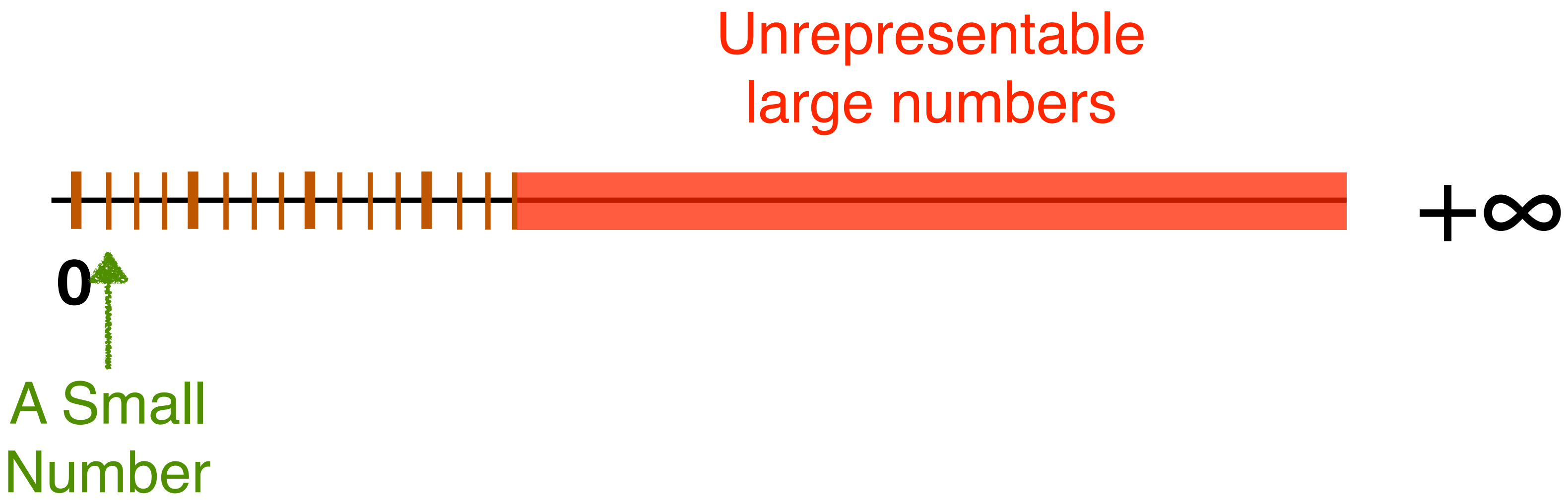
- Can’t represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  - To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers

Limitations of Fixed-Point (#2)

- Can’t represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  - To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers

Limitations of Fixed-Point (#2)

- Can't represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  - To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers



Today: Floating Point

- Background: Fractional binary numbers and fixed-point
- Floating point representation
- IEEE 754 standard
- Rounding, addition, multiplication
- Floating point in C
- Summary

Primer: (Normalized) Scientific Notation

Primer: (Normalized) Scientific Notation

Decimal Value	Scientific Notation
2	2×10^{0}
-4,321.768	-4.321768×10^{3}
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - E is an integer
  - Normalized form: $1 <= |M| < 10$

Decimal Value	Scientific Notation
2	2×10^0
-4,321.768	-4.321768×10^3
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - *E* is an integer
  - Normalized form: $1 <= |M| < 10$

$$M \times 10^E$$

Decimal Value	Scientific Notation
2	2×10^0
-4,321.768	-4.321768×10^3
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - E is an integer
  - Normalized form: $1 <= |M| < 10$

$$M \times 10^E$$

Significand

Decimal Value	Scientific Notation
2	2×10^0
-4,321.768	-4.321768×10^3
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - E is an integer
  - Normalized form: $1 <= |M| < 10$

$$M \times 10^{E}$$

Significand Base

Decimal Value	Scientific Notation
2	2×10^0
-4,321.768	-4.321768×10^3
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^{E}$
  - E is an integer
  - Normalized form: $1 <= |M| < 10$

$$M \times 10^{E}$$

Exponent

Significand Base

Decimal Value	Scientific Notation
2	2×10^{0}
-4,321.768	-4.321768×10^{3}
0.000 000 007 51	7.51×10^{-9}

Primer: (Normalized) Scientific Notation

Binary Value	Scientific Notation
1110110110110	$(-1)^0$ 1.110110110110 x 2^{12}
-101.11	$(-1)^1$ 1.0111 x 2^2
0.00101	$(-1)^0$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

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

Binary Value	**Scientific Notation**
1110110110110	$(-1)^{0}$ 1.110110110110 x 2^{12}
-101.11	$(-1)^{1}$ 1.0111 x 2^{2}
0.00101	$(-1)^{0}$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

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

Base

Binary Value	Scientific Notation
1110110110110	$(-1)^0$ 1.110110110110 x 2^{12}
-101.11	$(-1)^1$ 1.0111 x 2^2
0.00101	$(-1)^0$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

Exponent

$$(-1)^{s}\, M \times 2^{E}$$

Base

Binary Value	Scientific Notation
1110110110110	$(-1)^{0}$ 1.110110110110 x 2^{12}
-101.11	$(-1)^{1}$ 1.0111 x 2^{2}
0.00101	$(-1)^{0}$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

Exponent

$$(-1)^{s}\, M \times 2^{E}$$

Significand   Base

Binary Value	Scientific Notation
1110110110110	$(-1)^0\ 1.110110110110 \times 2^{12}$
-101.11	$(-1)^1\ 1.0111 \times 2^2$
0.00101	$(-1)^0\ 1.01 \times 2^{-3}$

Primer: (Normalized) Scientific Notation

$$(-1)^{\overset{\text{Sign}}{\downarrow}\, s}\ \underset{\overset{\uparrow}{\text{Significand}}}{M} \times \underset{\overset{\uparrow}{\text{Base}}}{2}^{\overset{\text{Exponent}}{\downarrow}\, E}$$

Binary Value	**Scientific Notation**
1110110110110	$(-1)^0$ 1.110110110110 x 2^{12}
-101.11	$(-1)^1$ 1.0111 x 2^2
0.00101	$(-1)^0$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

- In binary: $(-1)^s\ M\ 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$

$$(-1)^{\overset{\text{Sign}}{s}}\ \underset{\text{Significand}}{M} \times \underset{\text{Base}}{2}^{\overset{\text{Exponent}}{E}}$$

Binary Value	**Scientific Notation**
1110110110110	$(-1)^0\ 1.110110110110 \times 2^{12}$
-101.11	$(-1)^1\ 1.0111 \times 2^{2}$
0.00101	$(-1)^0\ 1.01 \times 2^{-3}$

Primer: (Normalized) Scientific Notation

- In binary: $(-1)^s\ M\ 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$

    Fraction

$$(-1)^{\overset{\text{Sign}}{s}}\ \underset{\text{Significand}}{M} \times \underset{\text{Base}}{2}^{\overset{\text{Exponent}}{E}}$$

Binary Value	Scientific Notation
1110110110110	$(-1)^0$ 1.110110110110 x 2^{12}
-101.11	$(-1)^1$ 1.0111 x 2^2
0.00101	$(-1)^0$ 1.01 x 2^{-3}

Primer: (Normalized) Scientific Notation

- In binary: $(-1)^{s}\, M\, 2^{E}$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0 b_1 b_2 b_3 \ldots$

    Fraction

$$(-1)^{\overset{\text{Sign}}{s}}\ \underset{\text{Significand}}{M} \times \underset{\text{Base}}{2}^{\overset{\text{Exponent}}{E}}$$

- If I tell you that there is a number where:
  - Fraction = 0101
  - s = 1
  - E = 10
  - You could reconstruct the number as $(-1)^{1} 1.0101 \times 2^{10}$

Primer: Floating Point Representation

- In binary: $(-1)^s\, M\, 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0 b_1 b_2 b_3 \ldots$

    Fraction

Sign Exponent

$$(-1)^{s}\, M \times 2^{E}$$

Significand Base

Primer: Floating Point Representation

- In binary: $(-1)^{s}\, M\, 2^{E}$
- Normalized form:
    - $1 <= M < 2$
    - $M = 1.\underbrace{b_0 b_1 b_2 b_3 \ldots}_{\text{Fraction}}$
- Encoding

$$(-1)^{\overset{\text{Sign}}{s}}\ \underset{\text{Significand}}{M} \times \underset{\text{Base}}{2}^{\overset{\text{Exponent}}{E}}$$

Primer: Floating Point Representation

- In binary: $(-1)^s \, M \, 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$

    Fraction
- Encoding

Sign

Exponent

$$\underbrace{(-1)^{\overset{\text{Sign}}{s}}}\ \underbrace{M}_{\text{Significand}} \times \underbrace{2}_{\text{Base}}{}^{\overset{\text{Exponent}}{E}}$$

Significand

Base

Primer: Floating Point Representation

- In binary: $(-1)^s\ M\ 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$

    Fraction
- Encoding
  - MSB s is sign bit s

Sign Exponent

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

Significand Base

Primer: Floating Point Representation

- In binary: $(-1)^s\, M\, 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$
    Fraction
- Encoding
  - MSB s is sign bit s
  - *exp* field encodes Exponent (but not exactly the same, more later)

Sign Exponent

$$(-1)^{s}\, M \times 2^{E}$$

Significand Base

Primer: Floating Point Representation

- In binary: $(-1)^s\ M\ 2^E$
- Normalized form:
  - $1 <= M < 2$
  - $M = 1.b_0b_1b_2b_3\ldots$

    Fraction

Sign Exponent

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

Significand Base

- Encoding
  - MSB *s* is sign bit s
  - *exp* field encodes Exponent (but not exactly the same, more later)
  - *frac* field encodes Fraction (but not exactly the same, more later)

s	exp	frac

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

6-bit Floating Point Example

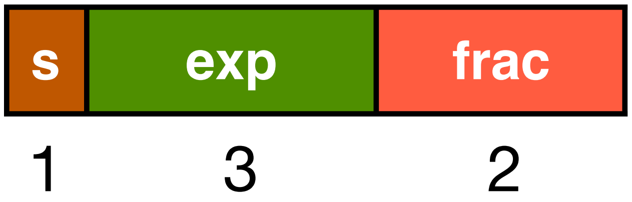
$$v = (-1)^s\, M\, 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value

6-bit Floating Point Example

$$v = (-1)^s M 2^E$$



- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?

6-bit Floating Point Example

$$v = (-1)^s M\, 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)

6-bit Floating Point Example

$v = (-1)^s \, M \, 2^E$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
    - If *exp* were *E*, we could represent exponents from **0 to 7**
    - How about negative exponent?
    - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
    - bias is always $2^{k-1} - 1$, where k is number of exponent bits

6-bit Floating Point Example

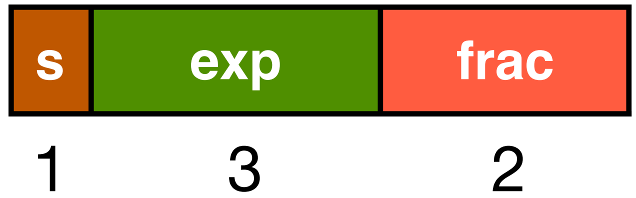
$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always 2^{k-1} - 1, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., *k* = 3):

6-bit Floating Point Example

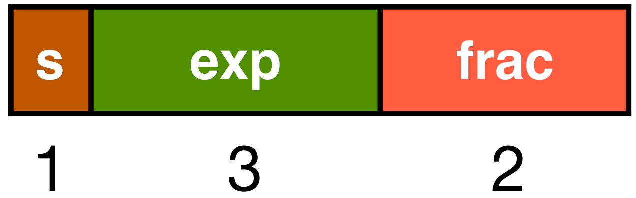
$v = (-1)^s M\, 2^E$



- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., $k = 3$):
  - bias = 3

6-bit Floating Point Example

$v = (-1)^s \, M \, 2^E$



- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., *k* = 3):
  - bias = 3
  - If *E* = -2, *exp* is 1 (001_2)

6-bit Floating Point Example

$$v = (-1)^s M 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., *k* = 3):
  - bias = 3
  - If *E* = -2, *exp* is 1 (001_2)

E	**exp**
-3	000
-2	001
-1	010
0	011
1	100
2	101
3	110
4	111

6-bit Floating Point Example

$$v = (-1)^s M \, 2^E$$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., *k* = 3):
  - bias = 3
  - If *E* = -2, *exp* is 1 (001_2)
  - Reserve 000 and 111 for other purposes (more on this later)

E	exp
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

6-bit Floating Point Example

$v = (-1)^s \, M \, 2^E$

s	exp	frac
1	3	2

- *exp* has 3 bits, interpreted as an unsigned value
  - If *exp* were *E*, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Subtract a bias term: *E = exp - bias* (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits
- Example when we use 3 bits for exp (i.e., *k* = 3):
  - bias = 3
  - If *E* = -2, *exp* is 1 (001_2)
  - Reserve 000 and 111 for other purposes (more on this later)
  - We can now represent exponents from **-2 (exp 001) to 3 (exp 110)**

E	**exp**
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

- *frac* has 2 bits, append them after “1.” to form M
    - *frac* = 10 implies M = 1.10

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

- *frac* has 2 bits, append them after "1." to form M
    - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \, 1.01 \times 2^1$$

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac
1	3	2

- *frac* has 2 bits, append them after "1." to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \; 1.01 \times 2^1$$

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

- *frac* has 2 bits, append them after "1." to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \, 1.01 \times 2^1$$

6-bit Floating Point Example

$v = (-1)^s \, M \, 2^E$

1	exp	frac
1	3	2

- *frac* has 2 bits, append them after “1.” to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \, 1.01 \times 2^1$$

6-bit Floating Point Example

$$v = (-1)^s \, M \, 2^E$$

1	exp	frac
1	3	2

- *frac* has 2 bits, append them after "1." to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \; 1.01 \times 2^1$$

E	exp
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

6-bit Floating Point Example

$v = (-1)^s\, M\, 2^E$

1	100	frac
1	3	2

- *frac* has 2 bits, append them after "1." to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1\ 1.01 \times 2^1$$

E	exp
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

6-bit Floating Point Example

$$v = (-1)^s\, M\, 2^E$$

1	100	frac
1	3	2

- *frac* has 2 bits, append them after “1.” to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1\ 1.01 \times 2^1$$

E	exp
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

6-bit Floating Point Example

$v = (-1)^s \, M \, 2^E$

s	exp	frac
1	100	01
1	3	2

- *frac* has 2 bits, append them after "1." to form M
  - *frac* = 10 implies M = 1.10
- Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \; 1.01 \times 2^1$$

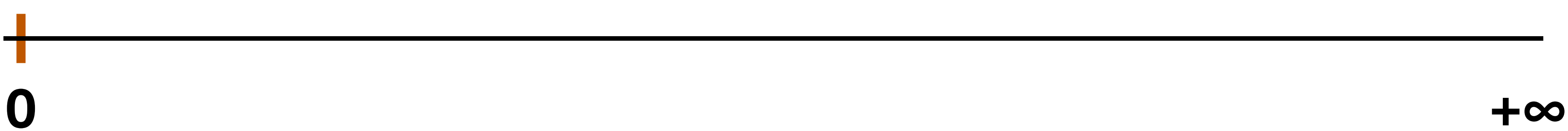
E	exp
~~-3~~	~~000~~
-2	001
-1	010
0	011
1	100
2	101
3	110
~~4~~	~~111~~

Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

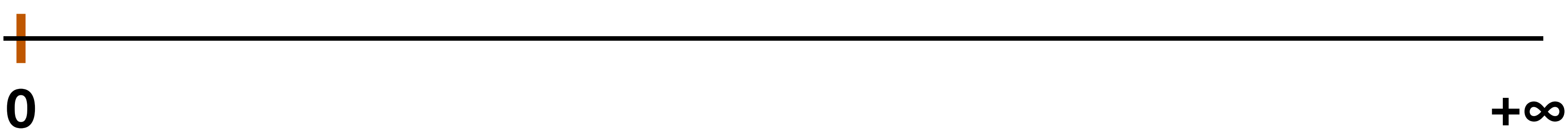


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	exp	frac

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

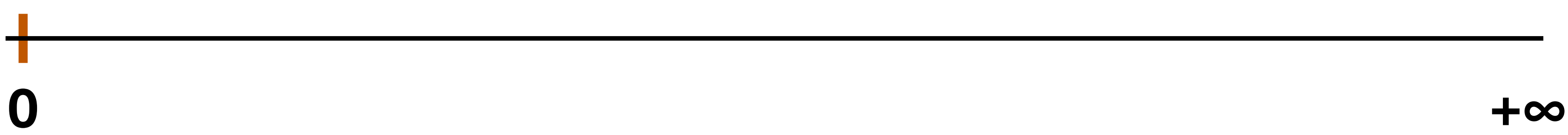


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	exp	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

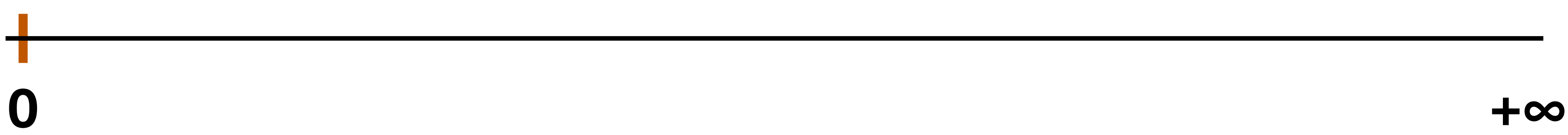


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	001	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~



Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	001	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	010	00

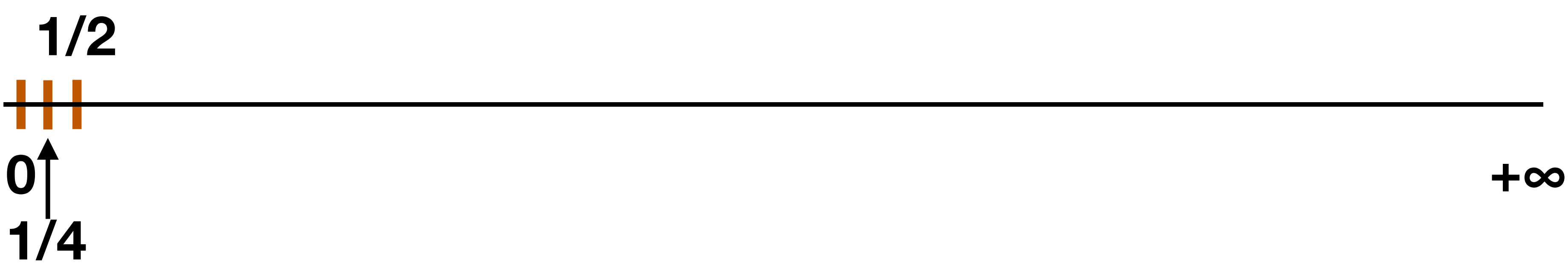
E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	010	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

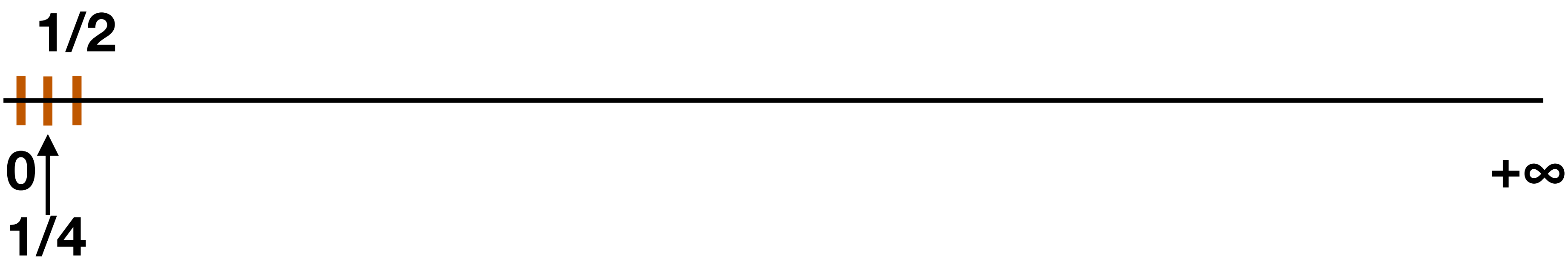


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	011	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

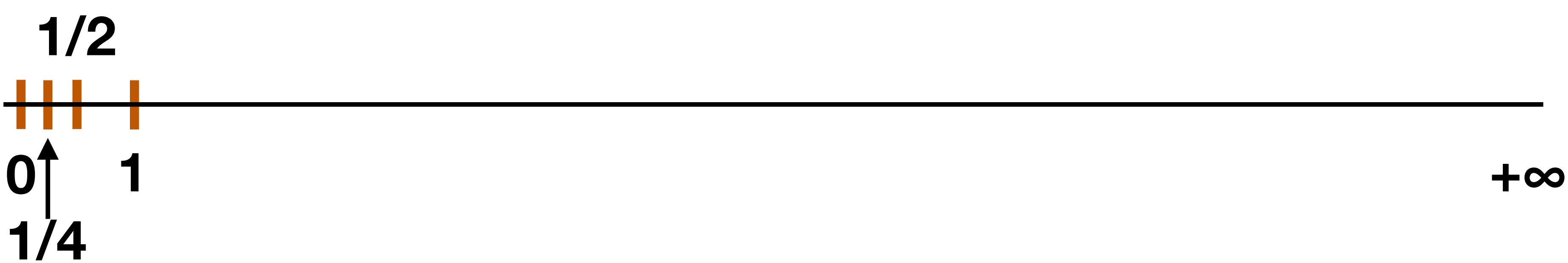


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

0	011	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

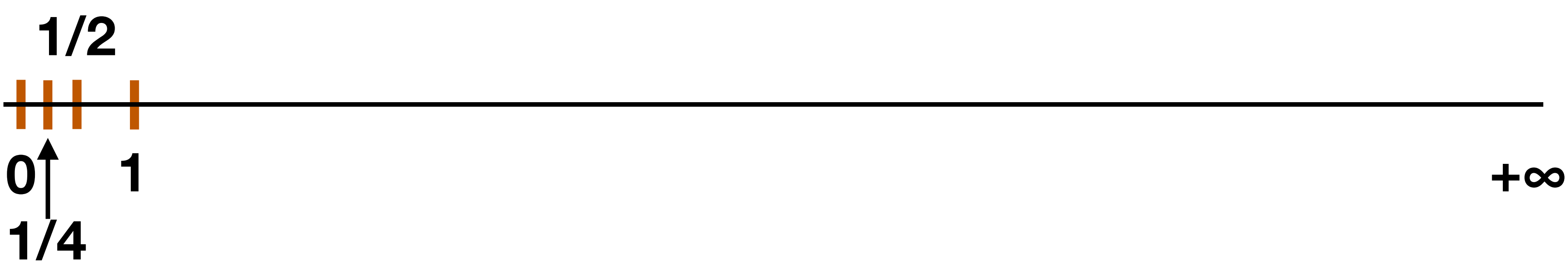


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	100	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

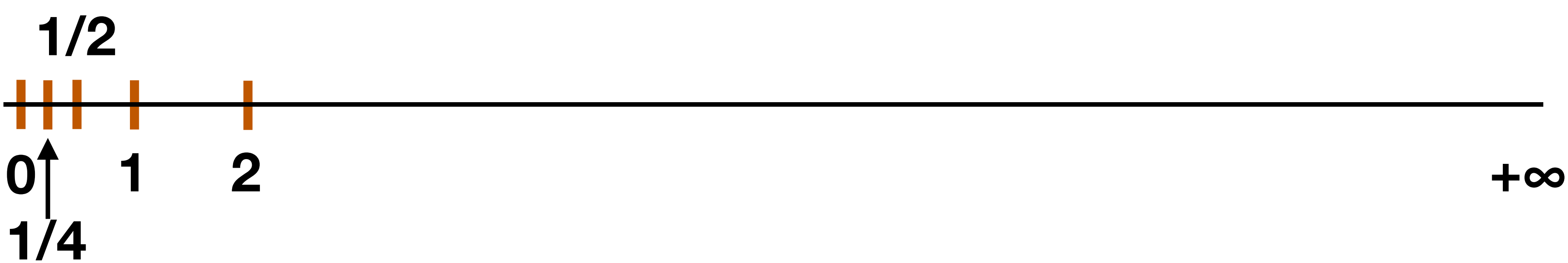


Representable Numbers (Positive Only)

$v = (-1)^s M\, 2^E$

0	100	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

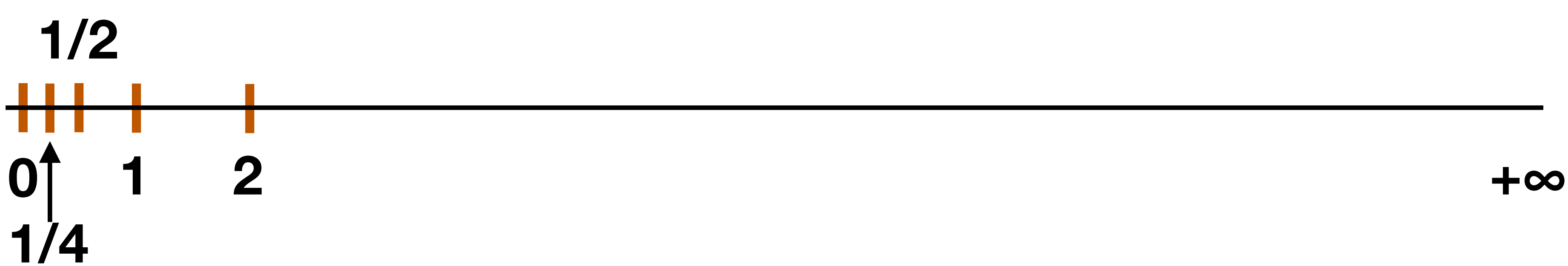


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

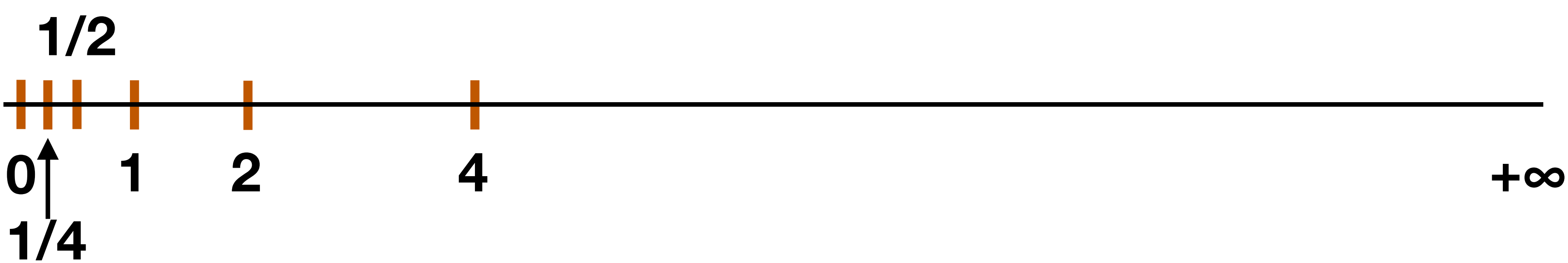


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

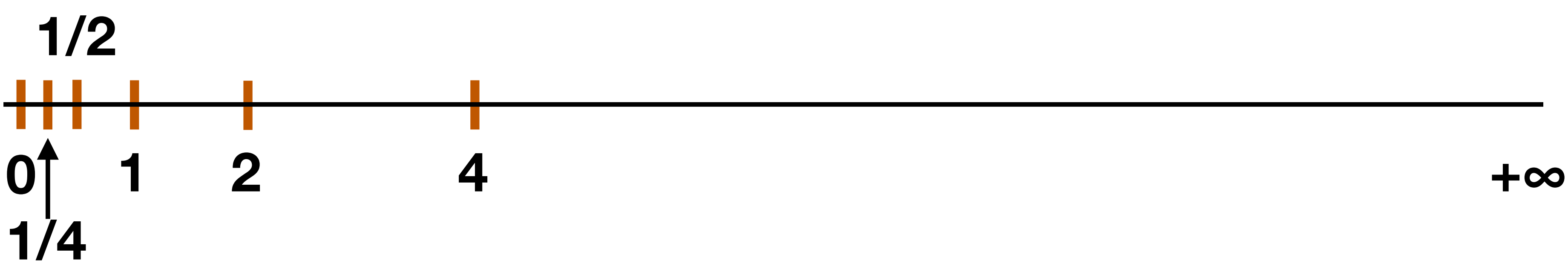


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

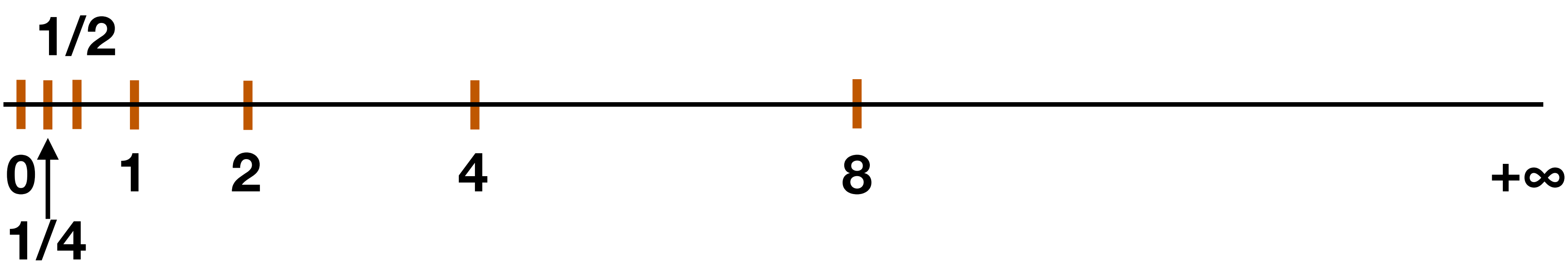


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	110	00

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

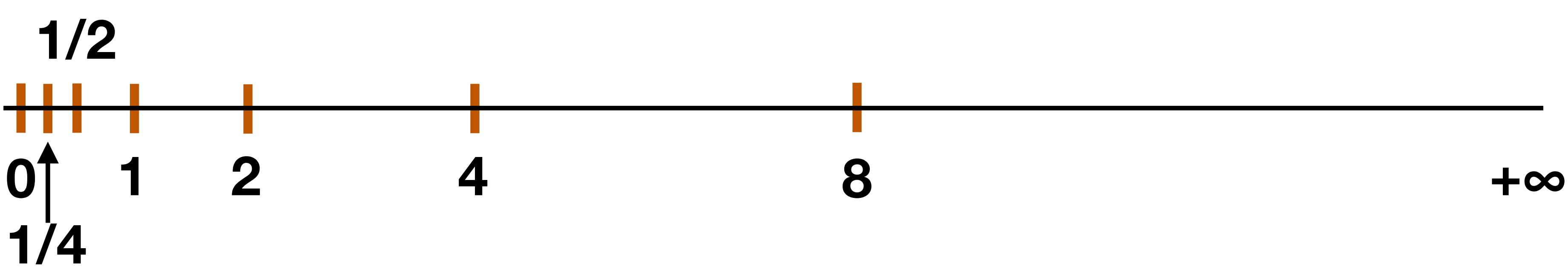


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	110	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

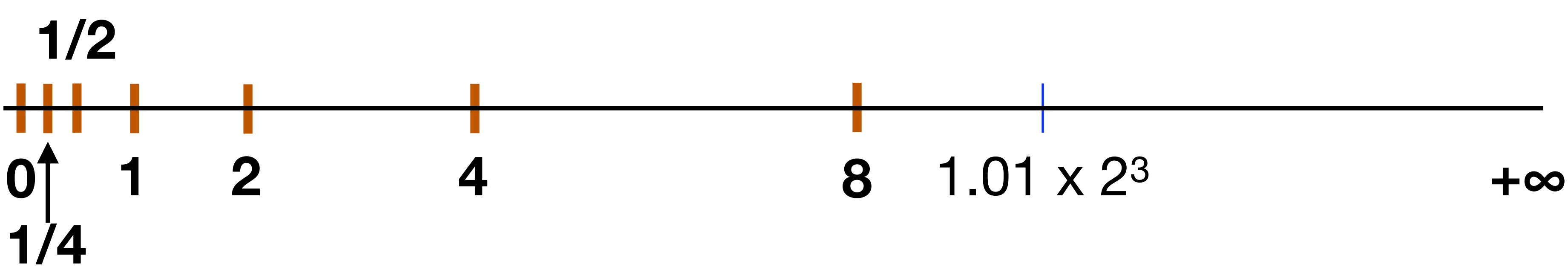


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

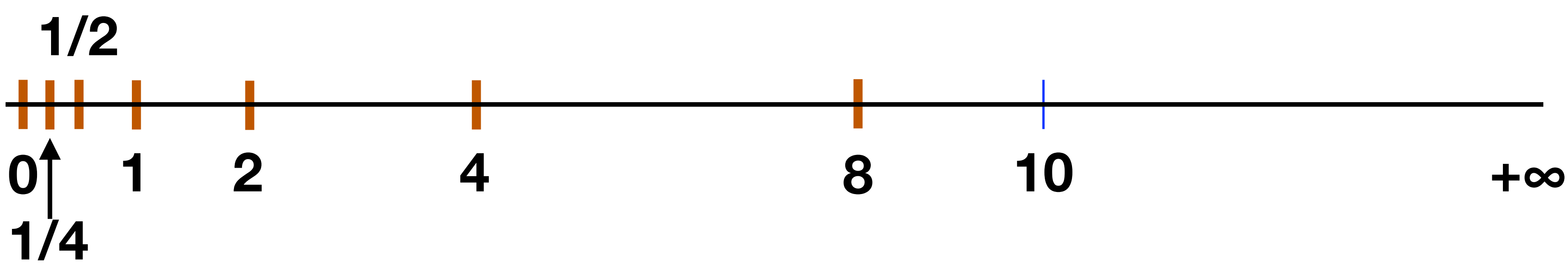


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

0	110	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

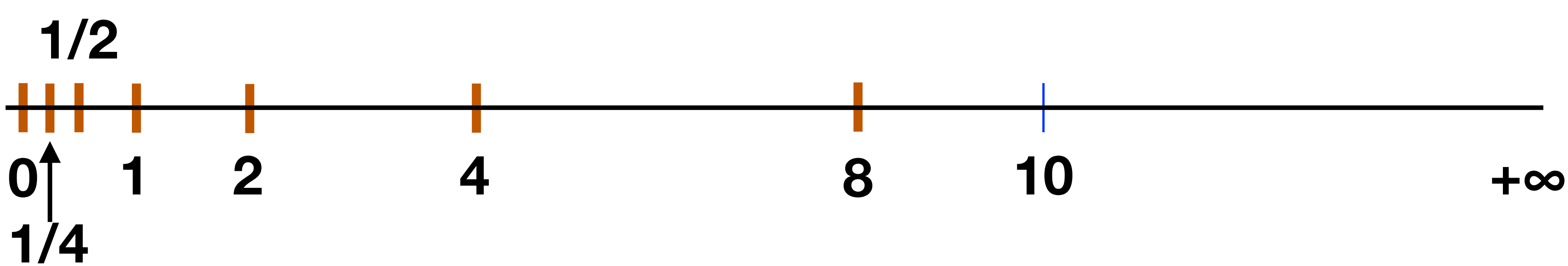


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

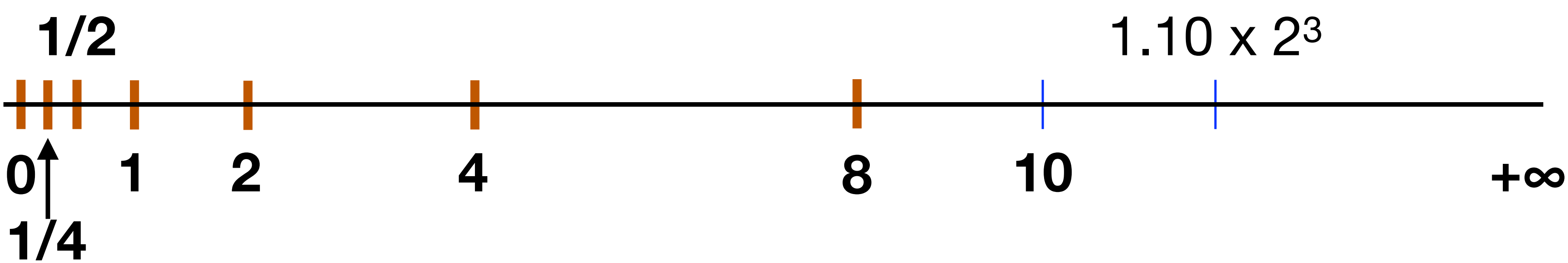


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

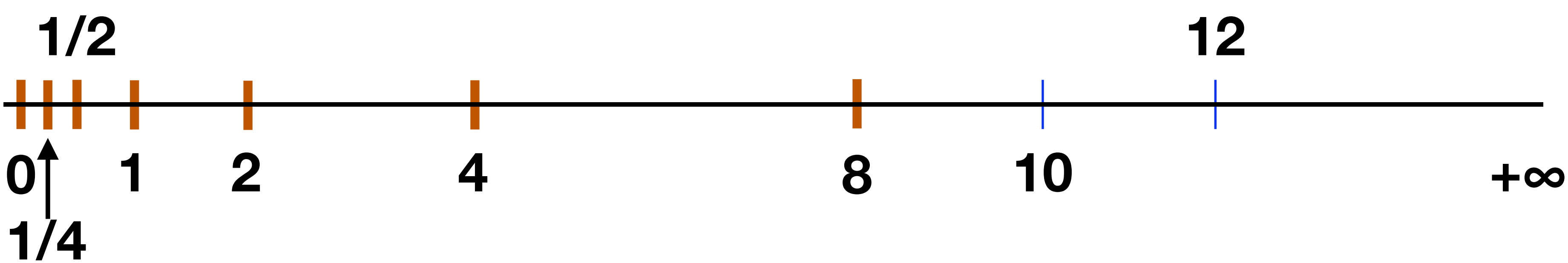


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

0	110	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

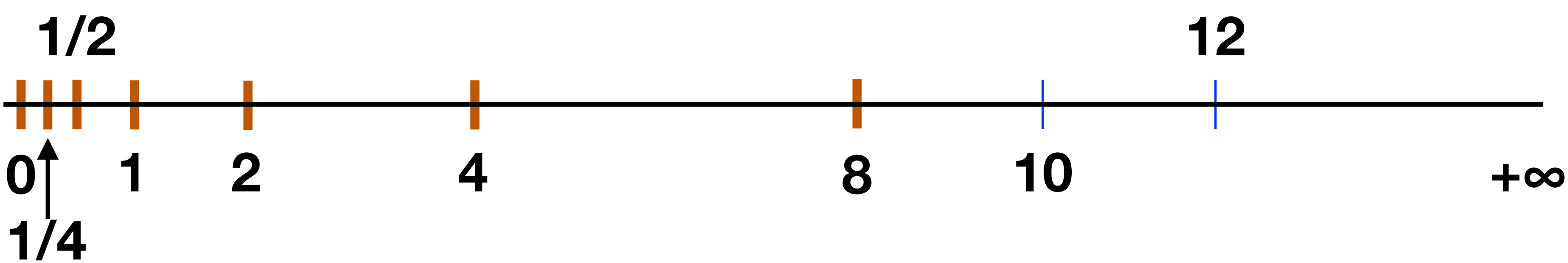


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

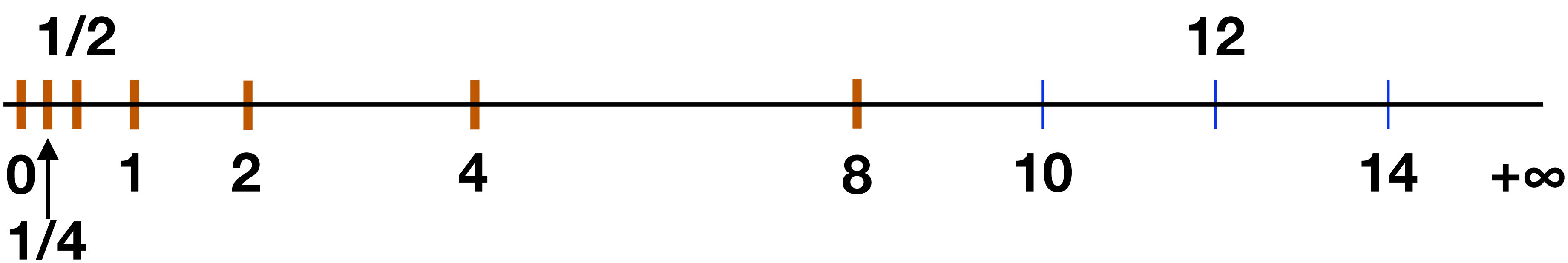


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	110	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

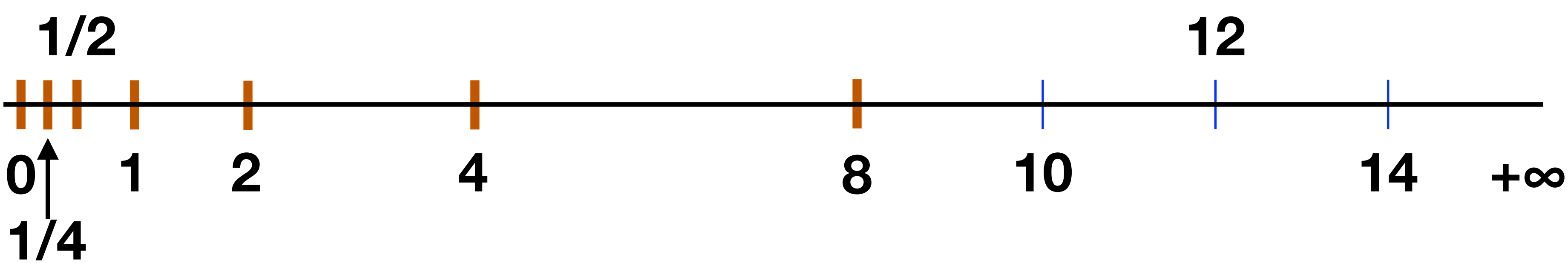


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	101	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

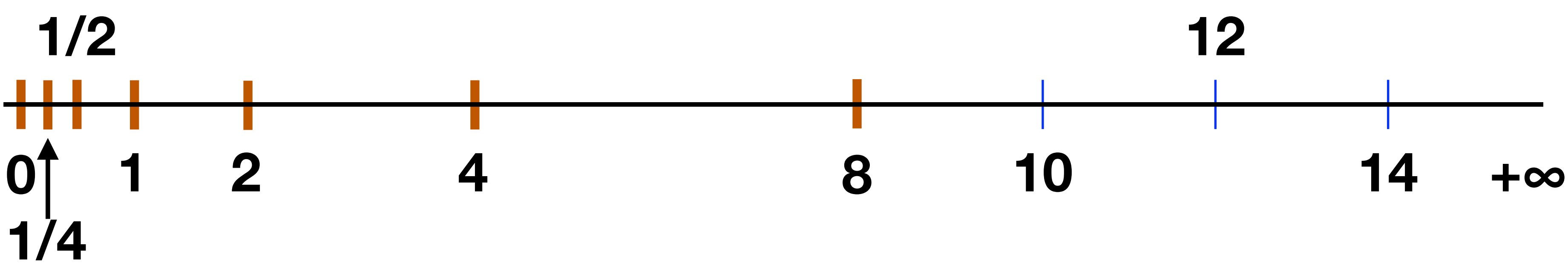


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

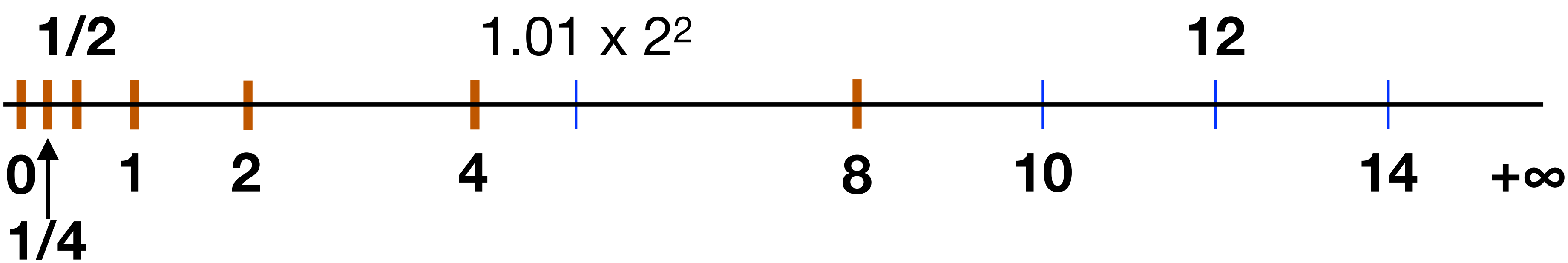


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

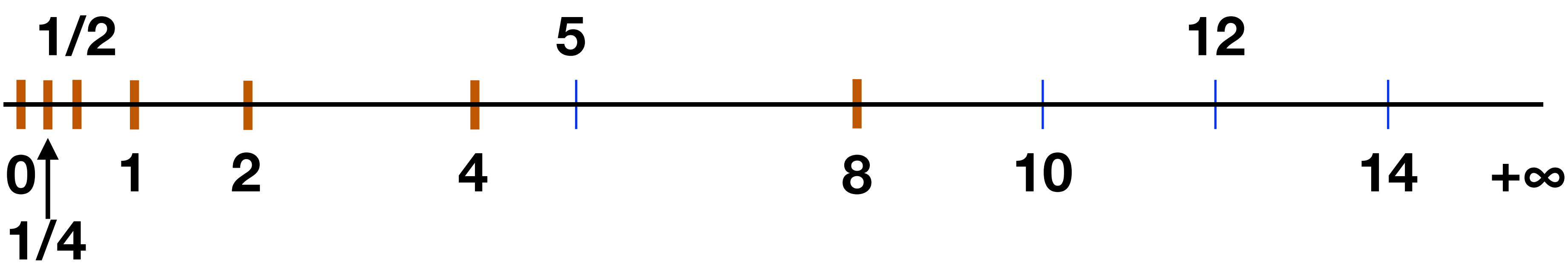


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	101	01

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

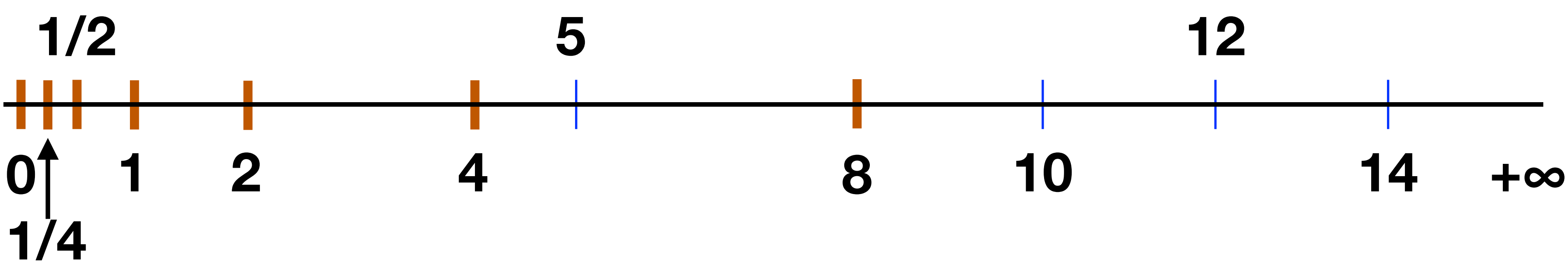


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	101	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

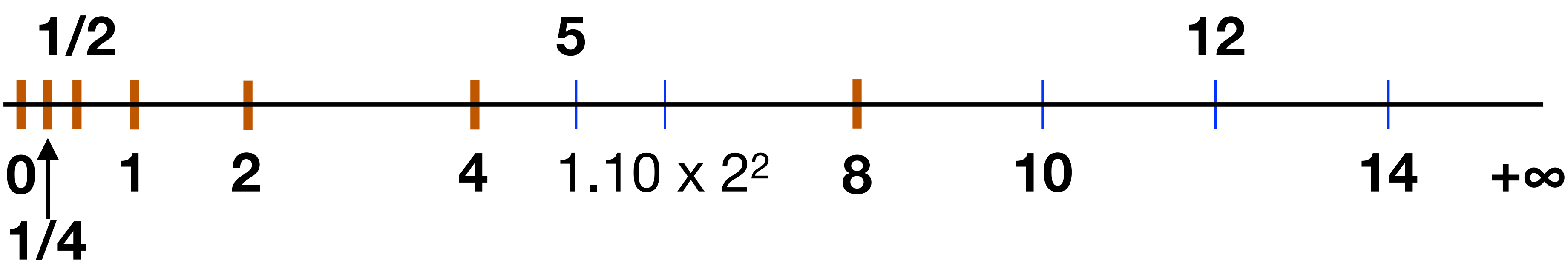


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

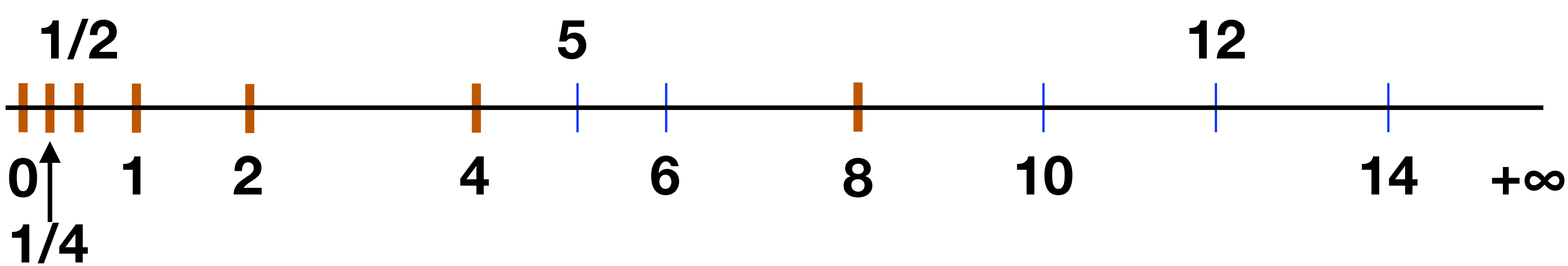


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	10

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

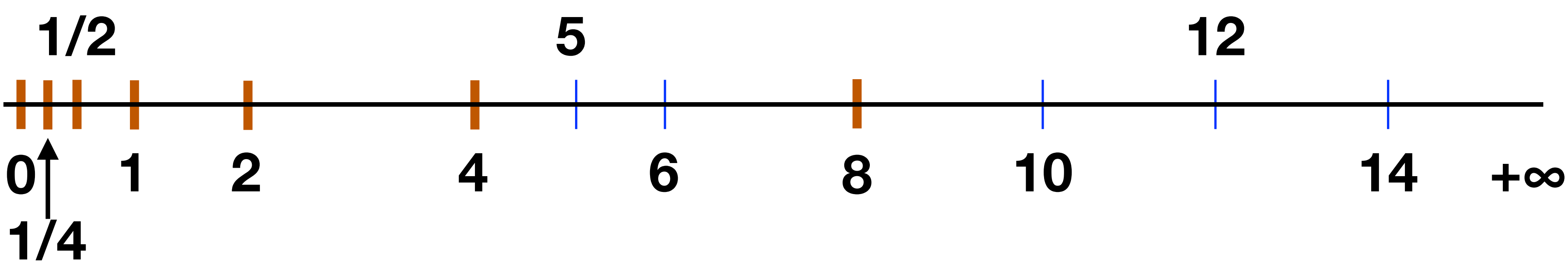


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

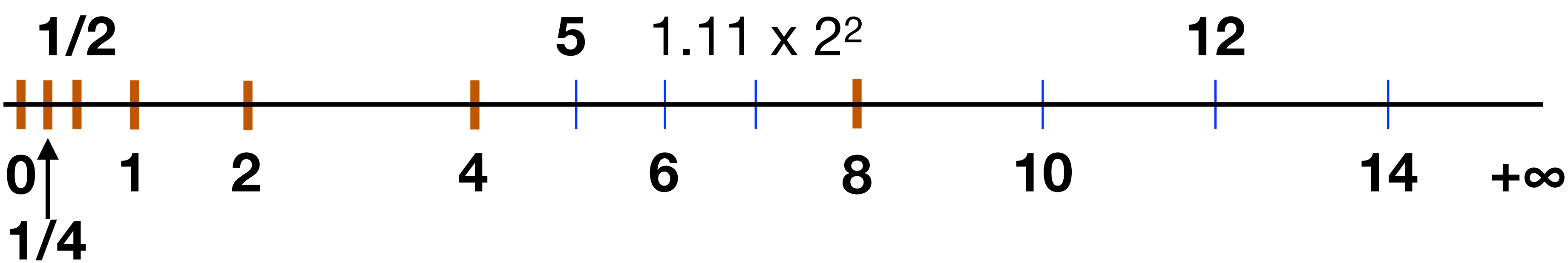


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	101	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

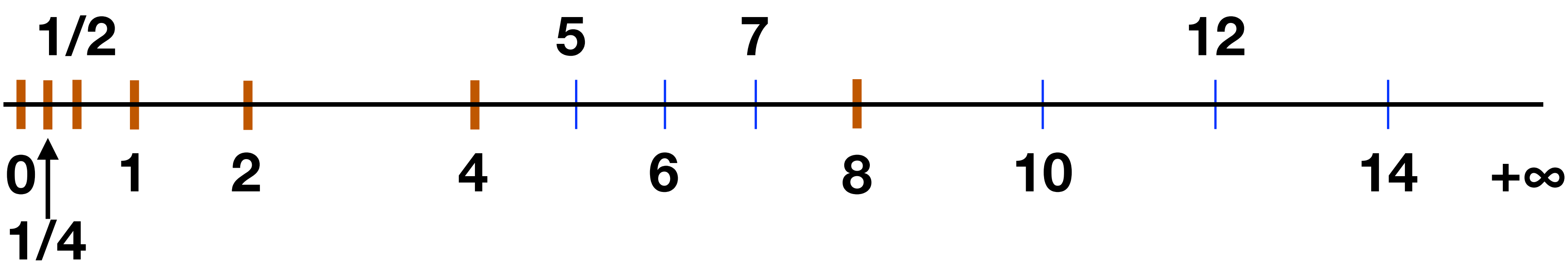


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	101	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

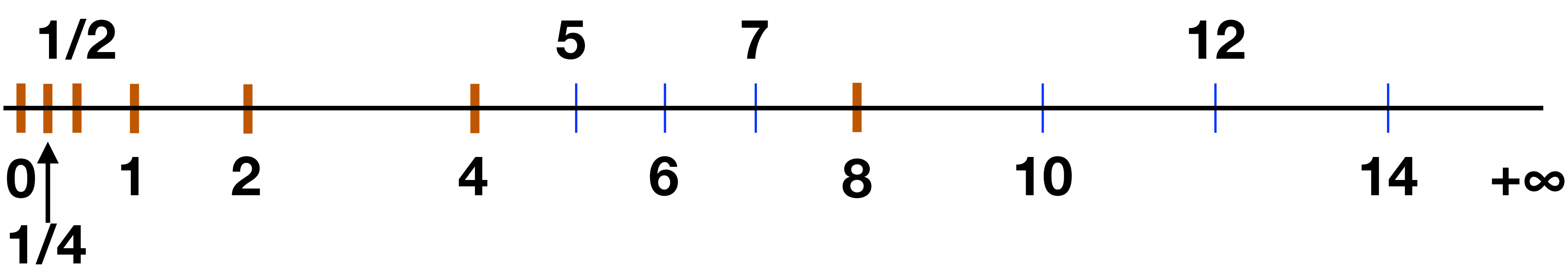


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	100	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

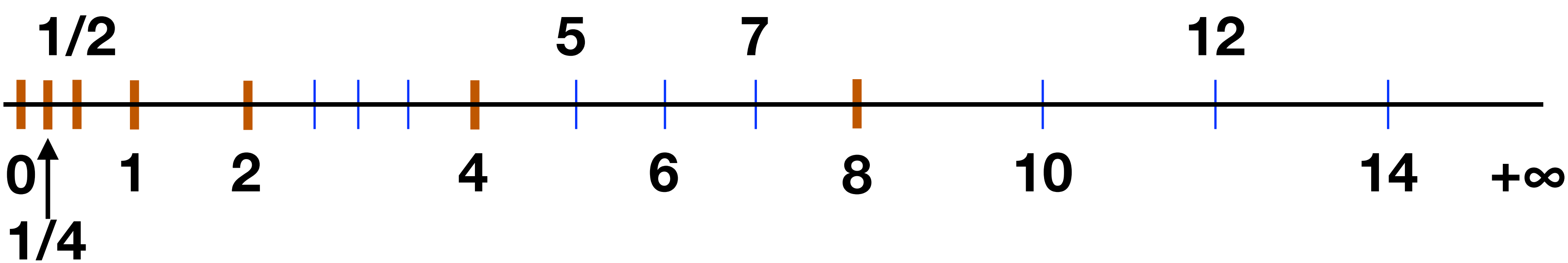


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	100	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

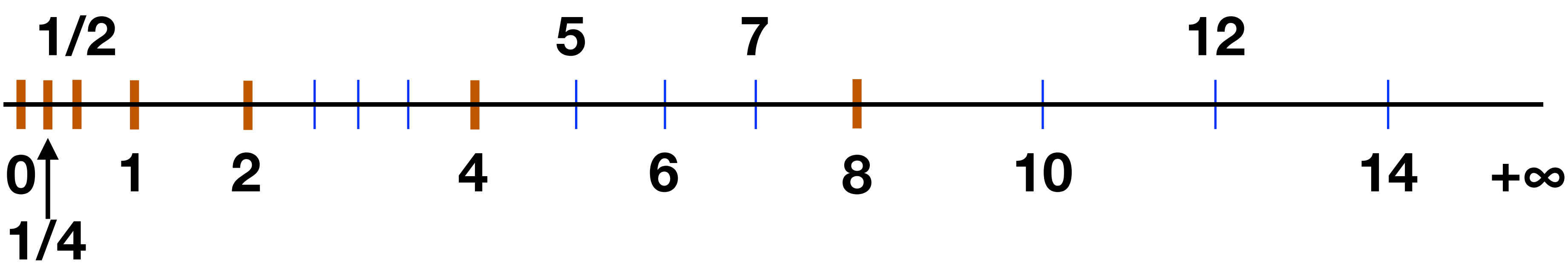


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	011	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

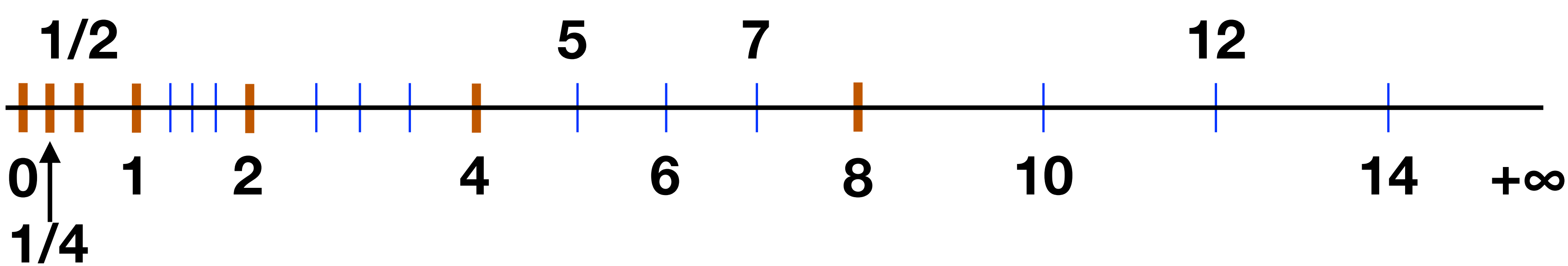


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

0	011	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

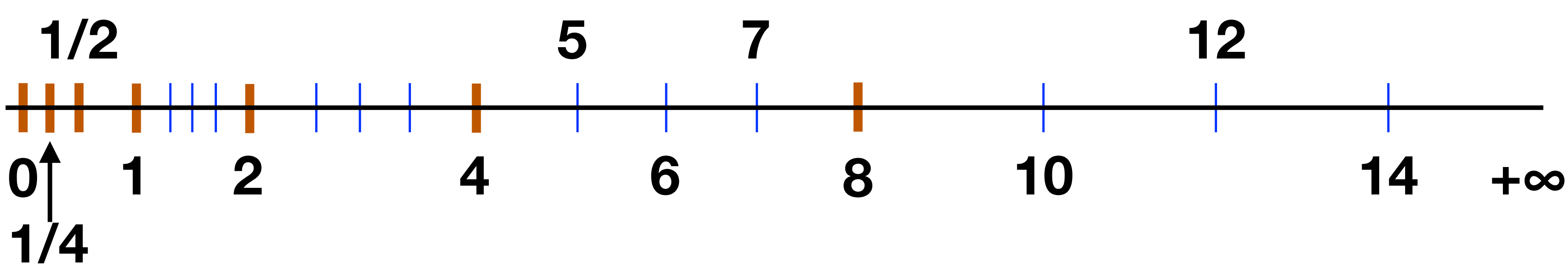


Representable Numbers (Positive Only)

$v = (-1)^s\, M\, 2^E$

0	010	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

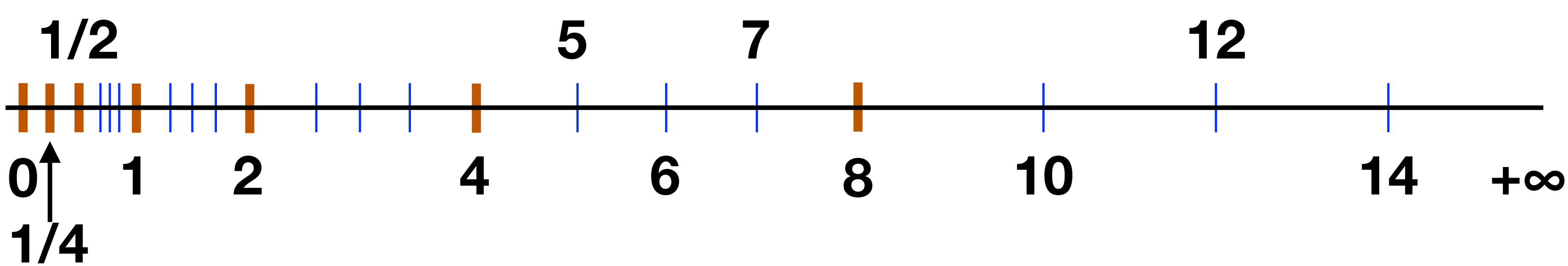


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

0	010	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

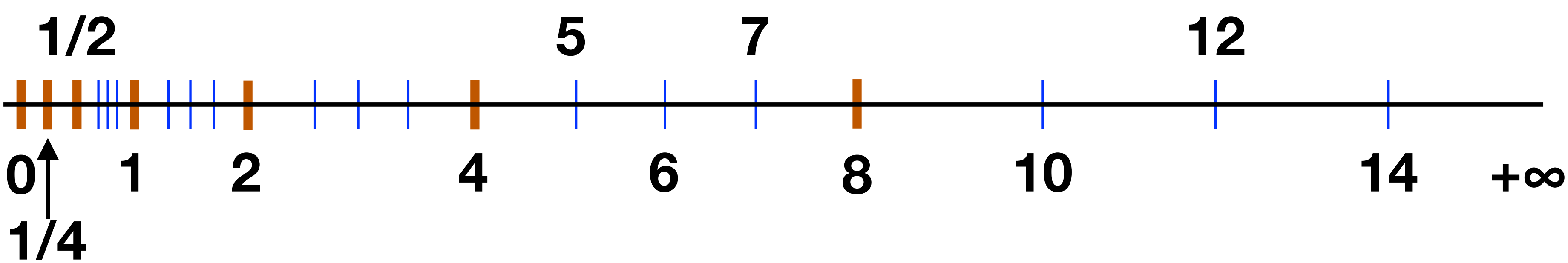


Representable Numbers (Positive Only)

$v = (-1)^s\, M\, 2^E$

0	001	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

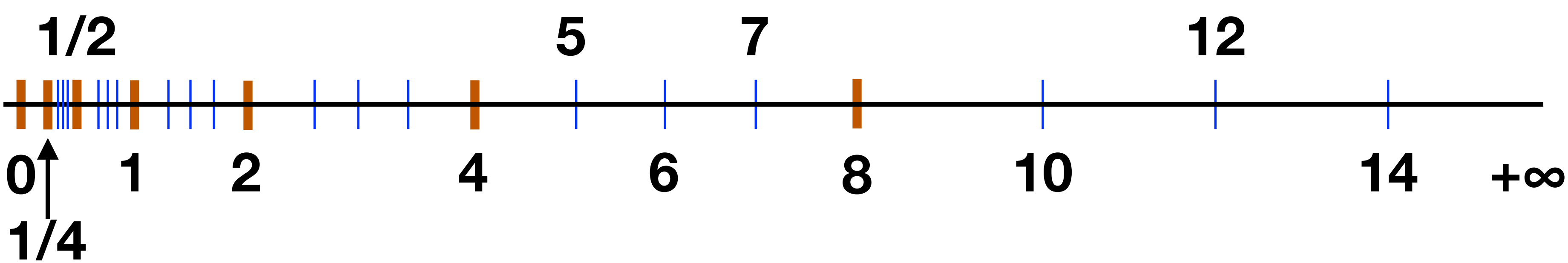


Representable Numbers (Positive Only)

$v = (-1)^s\ M\ 2^E$

0	001	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~



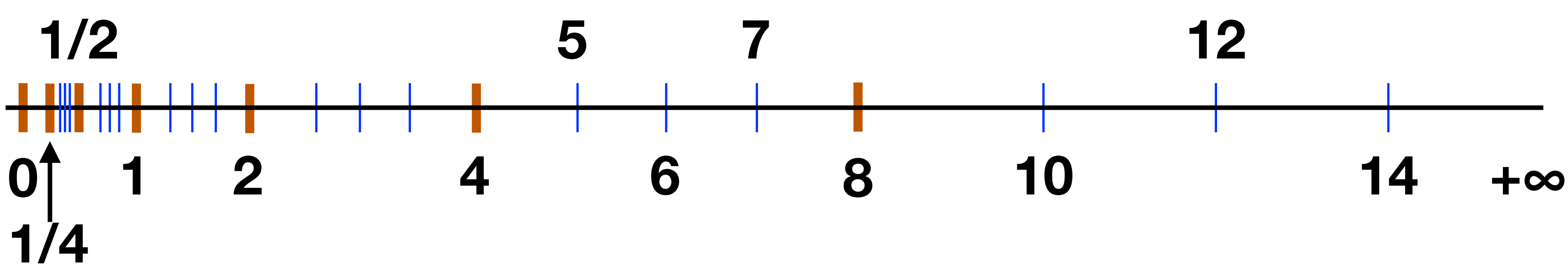
Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

0	001	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Uneven interval (c.f., fixed interval in fixed-point)
  - More dense toward 0, sparser toward infinite
  - Allow encoding small and large numbers at the same time



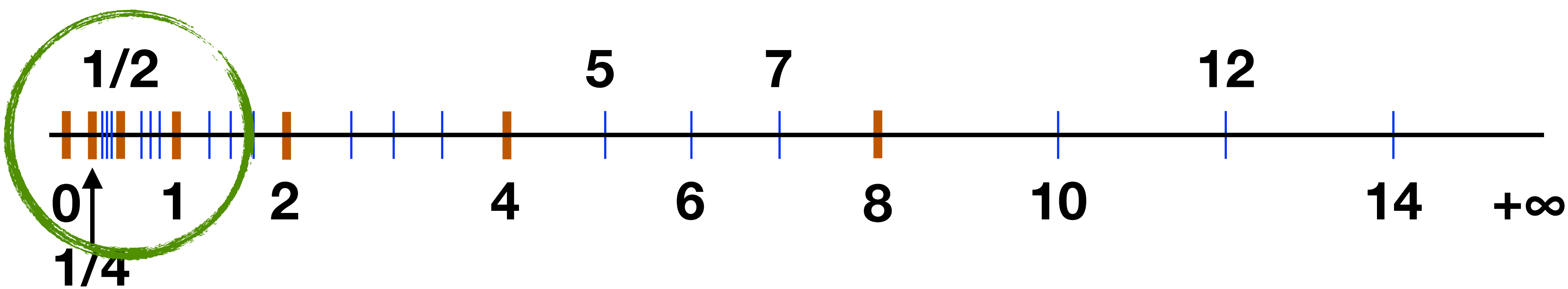
Representable Numbers (Positive Only)

$$v = (-1)^s M 2^E$$

0	001	11

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Uneven interval (c.f., fixed interval in fixed-point)
    - More dense toward 0, sparser toward infinite
    - Allow encoding small and large numbers at the same time

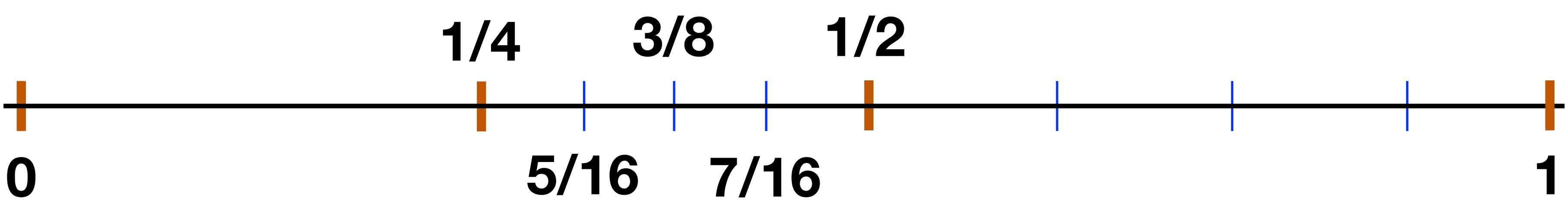


Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

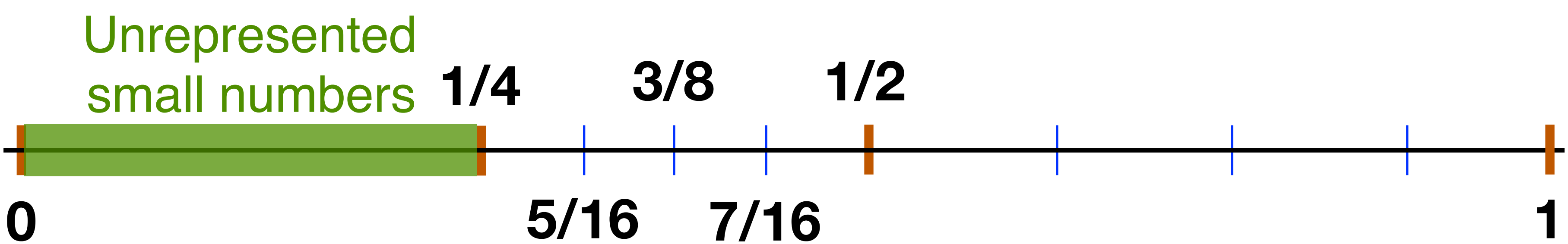


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac

E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

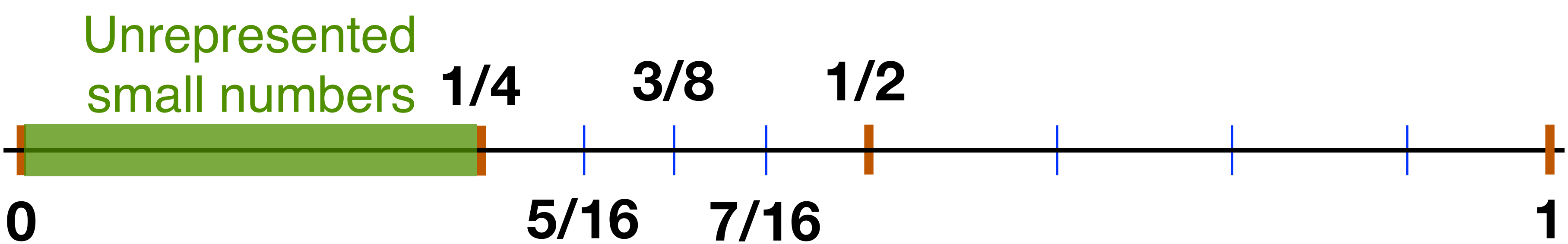


Representable Numbers (Positive Only)

$v = (-1)^s M 2^E$

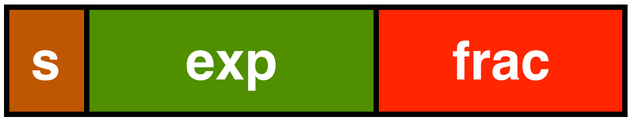
E	exp	E	exp
~~-3~~	~~000~~	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Always round to 0 is inelegant



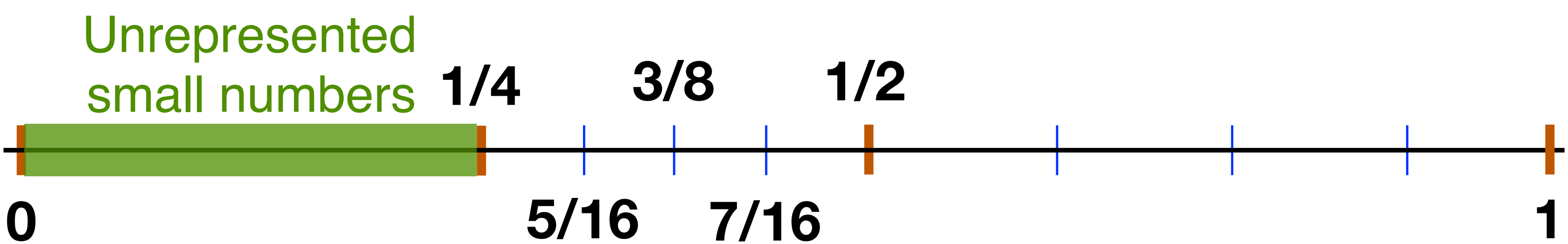
Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$



E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Always round to 0 is inelegant



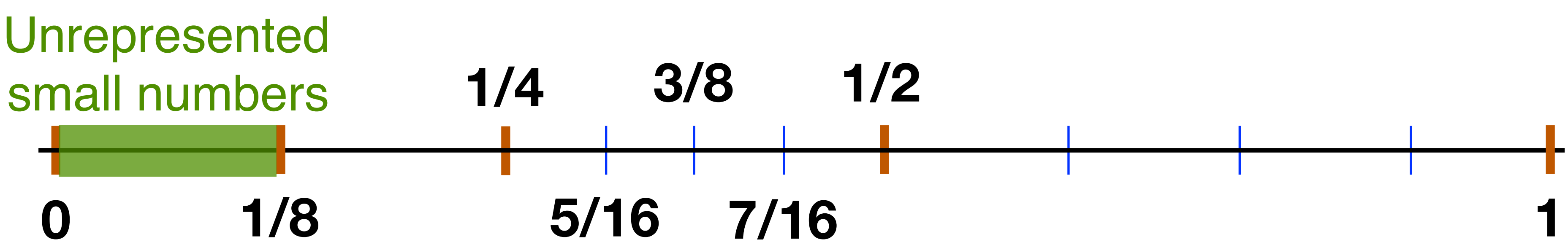
Representable Numbers (Positive Only)

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
$\boxed{-3}$	$\boxed{000}$	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Always round to 0 is inelegant

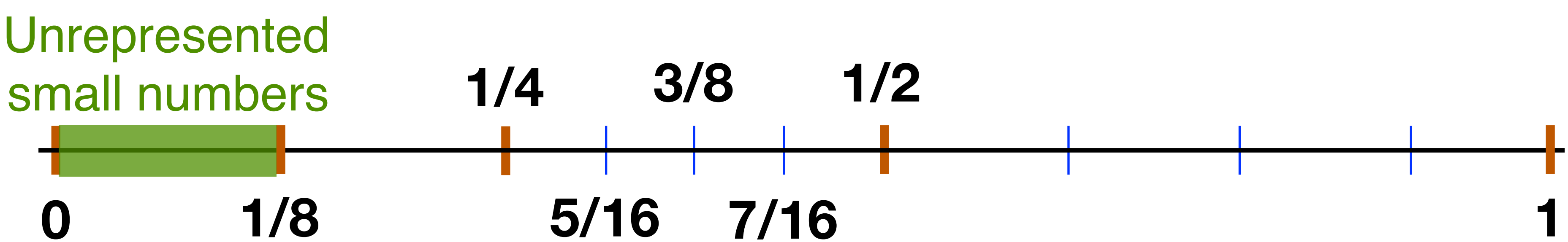


Representable Numbers (Positive Only)

$$v = (-1)^s \, M \, 2^E$$

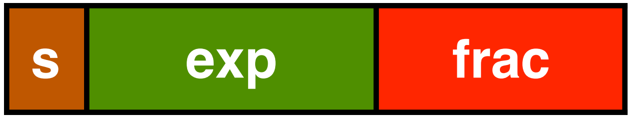
E	exp	E	exp
$\boxed{-3}$	$\boxed{000}$	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Always round to 0 is inelegant
- Using 000 for *exp* doesn't solve it either



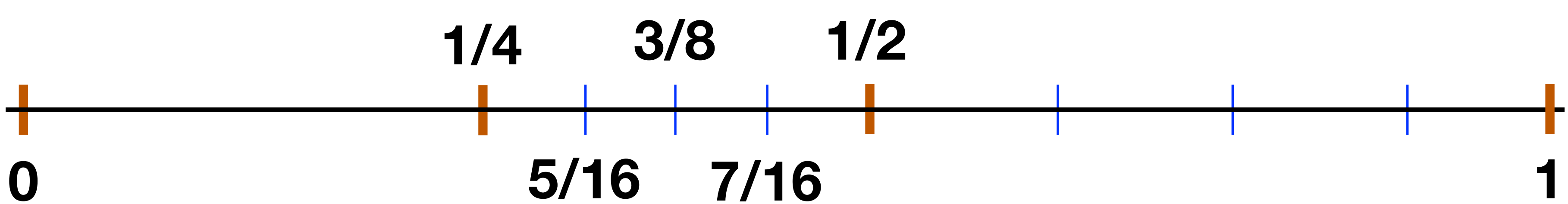
Subnormal (De-normalized) Numbers

$$v = (-1)^s \, M \, 2^E$$



E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing **when *exp* = 0** (subnormal/denormalized numbers)



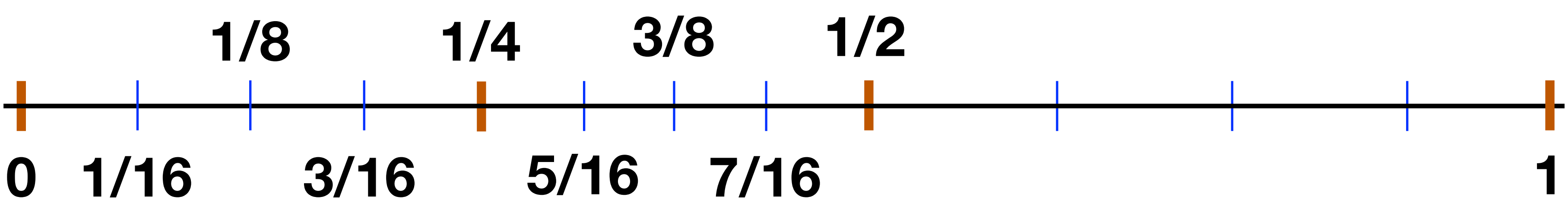
Subnormal (De-normalized) Numbers

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac

E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing **when *exp* = 0** (subnormal/denormalized numbers)



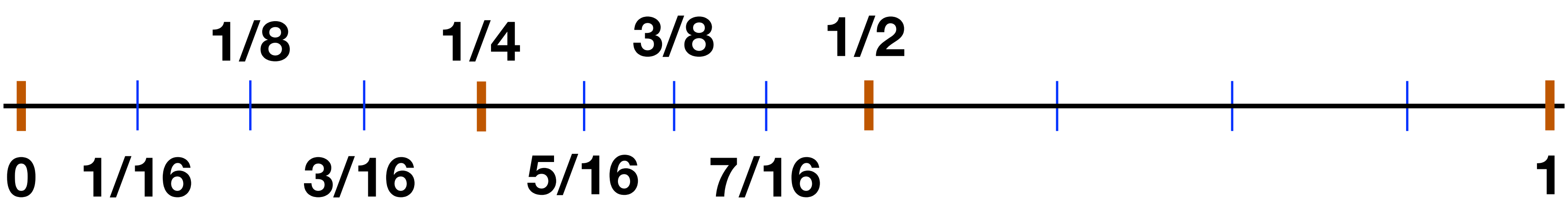
Subnormal (De-normalized) Numbers

$$v = (-1)^s \, M \, 2^E$$

s	exp	frac

E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing **when *exp* = 0** (subnormal/denormalized numbers)
- $E = (exp + 1) - \text{bias}$ (instead of *exp* - *bias*)
- $M = 0.frac$ (instead of 1.frac)



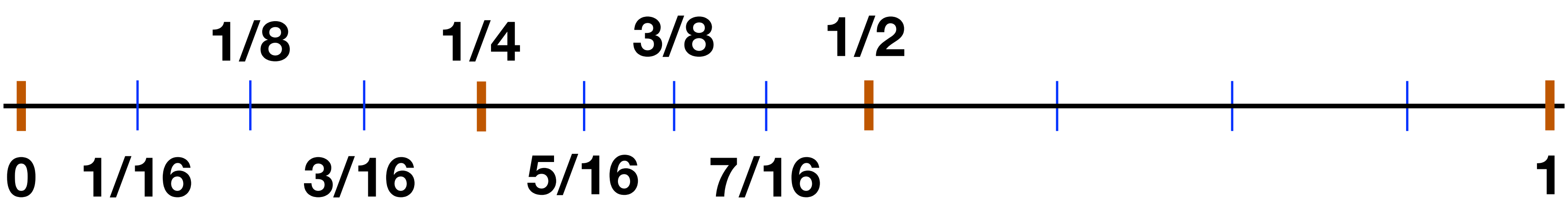
Subnormal (De-normalized) Numbers

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing **when *exp* = 0** (subnormal/denormalized numbers)
- $E = (exp + 1) - bias$ (instead of *exp* - *bias*)
- $M = 0.frac$ (instead of 1.frac)



0	000	01

$= (-1)^0 \, 0.01 \times 2^{(0+1-3)} = 1/16$

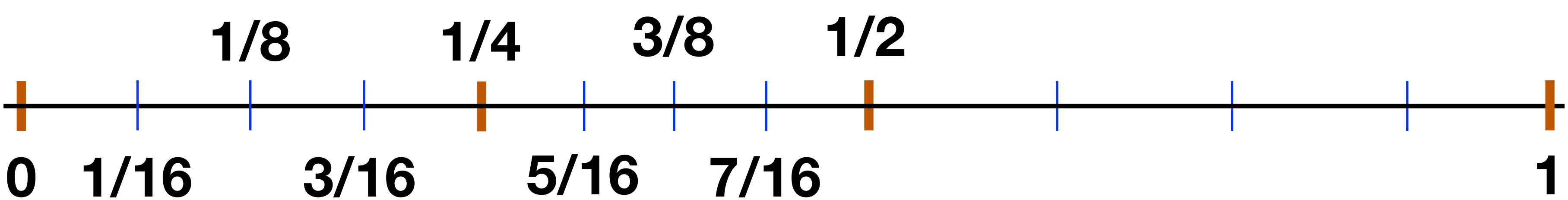
Subnormal (De-normalized) Numbers

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
-3	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing **when *exp* = 0** (subnormal/denormalized numbers)
- $E = (exp + 1) - bias$ (instead of *exp* - *bias*)
- $M = 0.frac$ (instead of 1.frac)
- Subnormal numbers allow graceful underflow



0	000	01

$= (-1)^0 \, 0.01 \times 2^{(0+1-3)} = 1/16$

Special Values

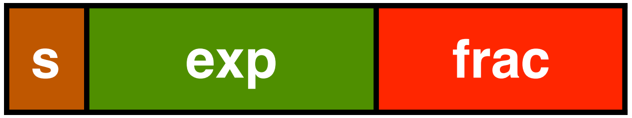
$$v = (-1)^s M\, 2^E$$

s	exp	frac

E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

Special Values

$$v = (-1)^s M 2^E$$



E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- There are many special values in scientific computing
  - +/- ∞, Not-a-Numbers (NaNs) (e.g., 0 / 0, 0 / ∞, ∞ / ∞, sqrt(–1), ∞ - ∞, ∞ x 0, etc.)

Special Values

$v = (-1)^s \, M \, 2^E$

s	exp	frac

E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011	~~4~~	~~111~~

- There are many special values in scientific computing
  - +/- ∞, Not-a-Numbers (NaNs) (e.g., 0 / 0, 0 / ∞, ∞ / ∞, sqrt(–1), ∞ - ∞, ∞ x 0, etc.)
- exp = 111 is reserved to represent these numbers

Special Values

$v = (-1)^s\, M\, 2^E$

s	exp	frac

E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011		111

- There are many special values in scientific computing
  - +/- ∞, Not-a-Numbers (NaNs) (e.g., 0 / 0, 0 / ∞, ∞ / ∞, sqrt(–1), ∞ - ∞, ∞ x 0, etc.)
- exp = 111 is reserved to represent these numbers

Special Values

$v = (-1)^s \, M \, 2^E$

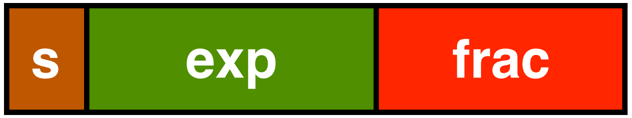
s	exp	frac

E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011		111

- There are many special values in scientific computing
  - +/- ∞, Not-a-Numbers (NaNs) (e.g., 0 / 0, 0 / ∞, ∞ / ∞, sqrt(–1), ∞ - ∞, ∞ x 0, etc.)
- exp = 111 is reserved to represent these numbers
- exp = 111, frac = 00
  - +/- ∞ (depending on the *s* bit). Overflow results.
  - Arithmetic on ∞ is exact: 1.0/0.0 = –1.0/–0.0 = + ∞, 1.0/–0.0 = -∞

Special Values

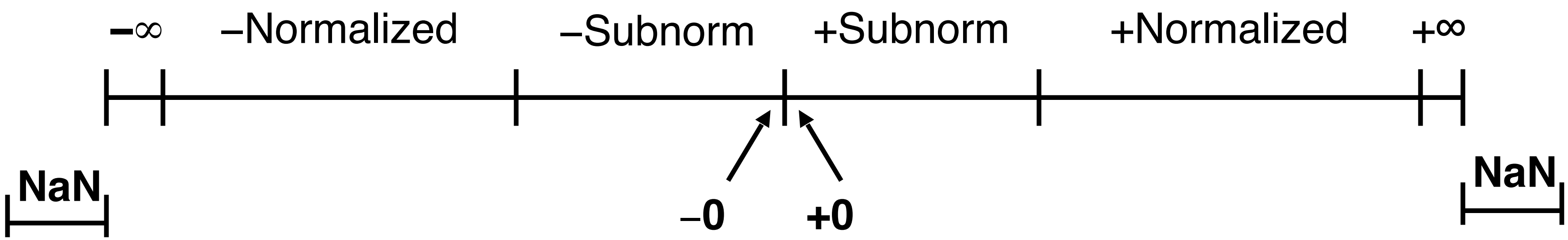
$$v = (-1)^s\, M\, 2^E$$



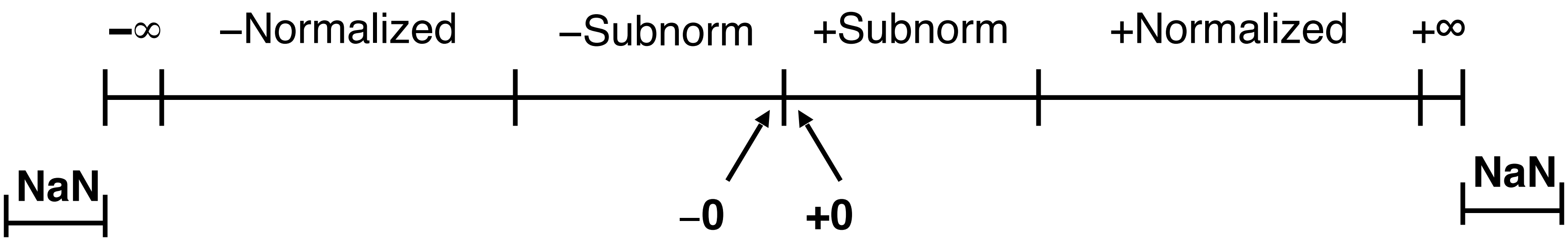
E	exp	E	exp
-2	000	1	100
-2	001	2	101
-1	010	3	110
0	011		111

- There are many special values in scientific computing
  - +/- ∞, Not-a-Numbers (NaNs) (e.g., 0 / 0, 0 / ∞, ∞ / ∞, sqrt(–1), ∞ - ∞, ∞ x 0, etc.)
- exp = 111 is reserved to represent these numbers
- exp = 111, frac = 00
  - +/- ∞ (depending on the *s* bit). Overflow results.
  - Arithmetic on ∞ is exact: 1.0/0.0 = –1.0/–0.0 = + ∞, 1.0/–0.0 = -∞
- exp = 111, frac != 00
  - Represent NaNs

Visualization: Floating Point Encodings

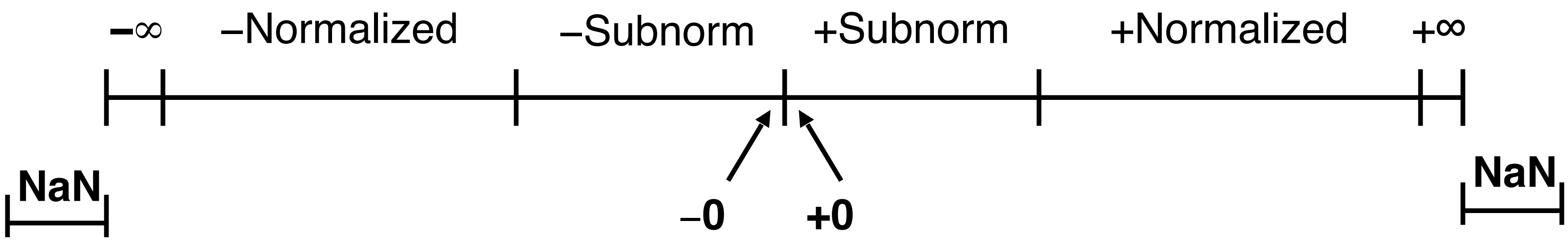


Visualization: Floating Point Encodings

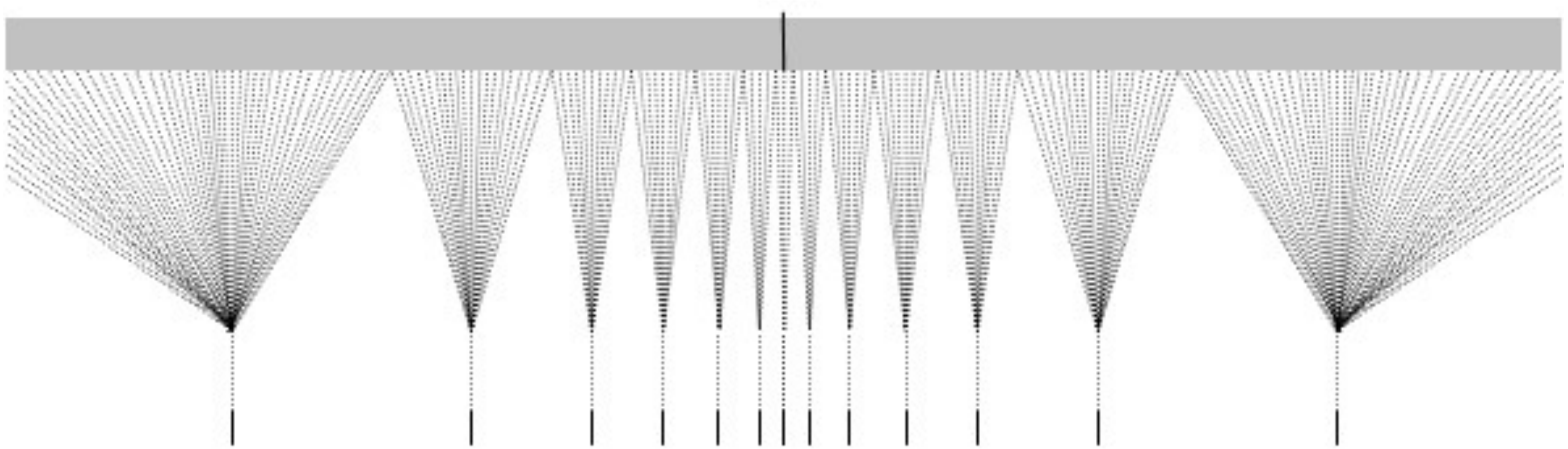


Infinite Amount of Real Numbers

Visualization: Floating Point Encodings

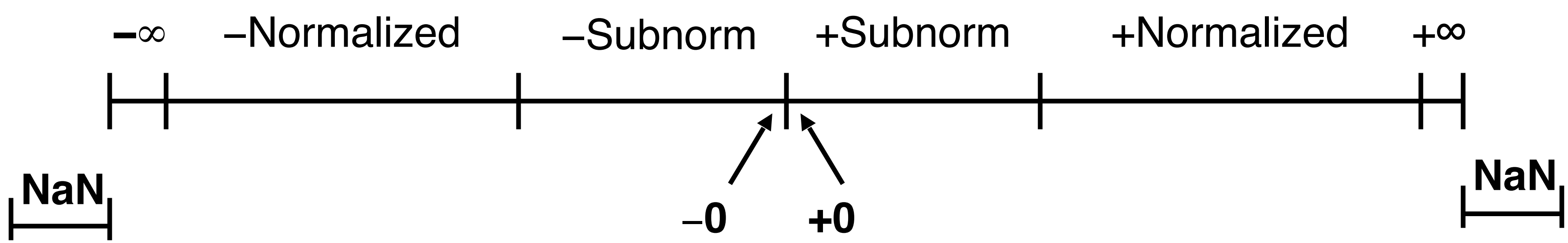


Infinite Amount of Real Numbers

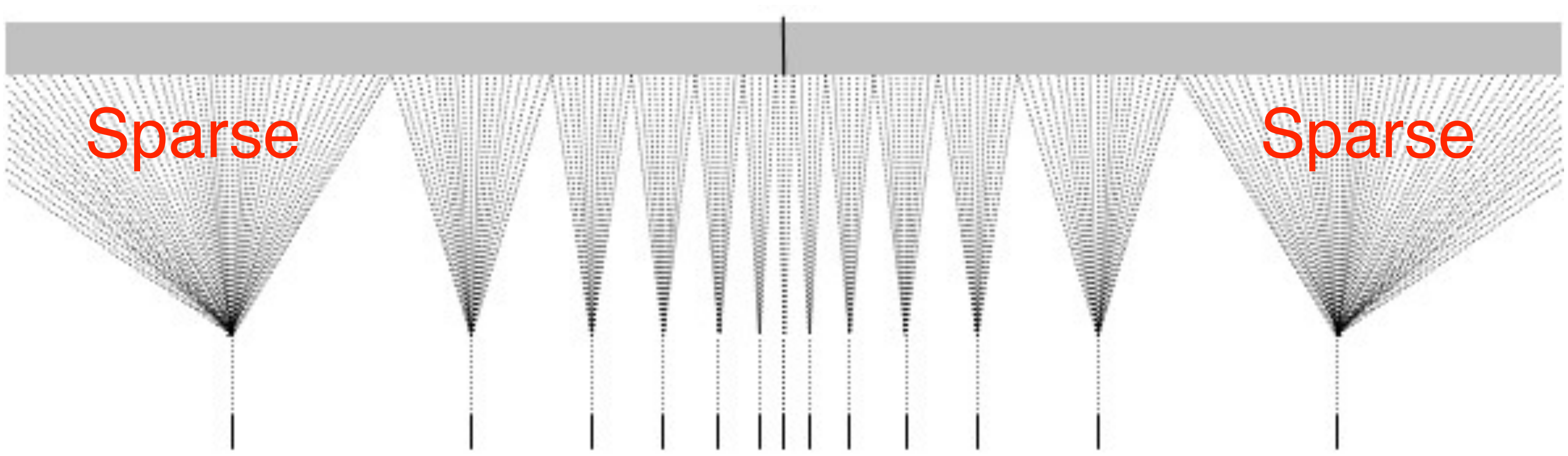


Finite Amount of Floating Point Numbers

Visualization: Floating Point Encodings



Infinite Amount of Real Numbers



Finite Amount of Floating Point Numbers

Visualization: Floating Point Encodings