Number Systems

- Decimal numbers

1's column
10's column
100's column
1000's column

5374_{10} =

- Binary numbers

1's column
2's column
4's column
8's column

1101_2 =

Number Systems

- Decimal numbers

1's column
10's column
100's column
1000's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands | three hundreds | seven tens | four ones

- Binary numbers

1's column
2's column
4's column
8's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight | one four | no two | one one

FROM ZERO TO ONE
ELSEVIER

Powers of Two

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$

FROM ZERO TO ONE

ELSEVIER

Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- Handy to memorize up to 2^9

FROM ZERO TO ONE

ELSEVIER

Number Conversion

- Decimal to binary conversion:
  - Convert 10011_2 to decimal

- Decimal to binary conversion:
  - Convert 47_{10} to binary

Number Conversion

- Decimal to binary conversion:
  - Convert 10011_2 to decimal
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$

- Decimal to binary conversion:
  - Convert 47_{10} to binary
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

FROM ZERO TO ONE

ELSEVIER

Binary Values and Range

- *N*-digit decimal number
  - How many values?
  - Range?
  - Example: 3-digit decimal number:

- *N*-bit binary number
  - How many values?
  - Range:
  - Example: 3-digit binary number:

FROM ZERO TO ONE

ELSEVIER

Binary Values and Range

- *N*-digit decimal number
  - How many values? **10^N**
  - Range? **$[0, 10^N - 1]$**
  - Example: 3-digit decimal number:
    - **10^3 = 1000 possible values**
    - **Range: [0, 999]**
- *N*-bit binary number
  - How many values? **2^N**
  - Range: **$[0, 2^N - 1]$**
  - Example: 3-digit binary number:
    - **2^3 = 8 possible values**
    - **Range: [0, 7] = [000_2 to 111_2]**

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

FROM ZERO TO ONE

ELSEVIER

Hexadecimal Numbers

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

FROM ZERO TO ONE

ELSEVIER

Hexadecimal Numbers

- Base 16
- Shorthand for binary

FROM ZERO TO ONE

Hexadecimal to Binary

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary

- Hexadecimal to decimal conversion:
  - Convert 0x4AF to decimal

FROM ZERO TO ONE

Hexadecimal to Binary

- Hexadecimal to binary conversion:
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  - $0100\ 1010\ 1111_2$

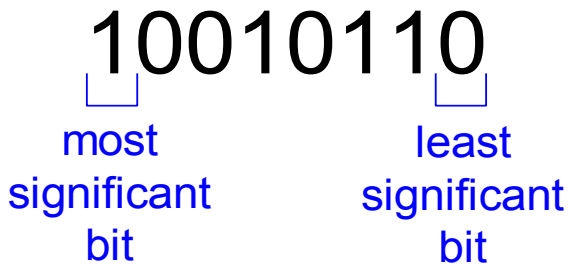
- Hexadecimal to decimal conversion:
  - Convert $4AF_{16}$ to decimal
  - $16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}$

FROM ZERO TO ONE

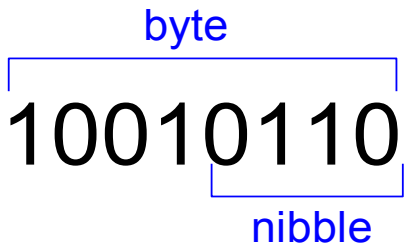
ELSEVIER

Bits, Bytes, Nibbles…

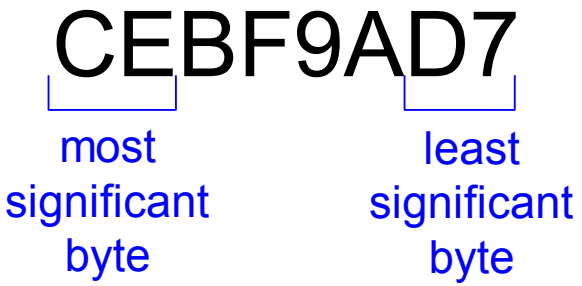
- Bits



- Bytes & Nibbles



- Bytes



Large Powers of Two

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 billion (1,073,741,824)

FROM ZERO TO ONE

Estimating Powers of Two

- What is the value of 2^{24}?
- How many values can a 32-bit variable represent?

FROM ZERO TO ONE

ELSEVIER

Estimating Powers of Two

- What is the value of 2^{24}?
  - **$2^4 \times 2^{20} \approx$ 16 million**
- How many values can a 32-bit variable represent?
  - **$2^2 \times 2^{30} \approx$ 4 billion**

FROM ZERO TO ONE

ELSEVIER

Addition

- Decimal

```
   11   ← carries
    3734
+   5168
-------
    8902
```

- Binary

```
   11   ← carries
    1011
+   0011
-------
    1110
```

FROM ZERO TO ONE

ELSEVIER

Binary Addition Examples

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline \end{array}$$

FROM ZERO TO ONE

ELSEVIER

Binary Addition Examples

- Add the following 4-bit binary numbers

```
   1
 1001
+0101
-----
 1110
```

- Add the following 4-bit binary numbers

```
  111
 1011
+0110
-----
10001
```

Overflow!

FROM ZERO TO ONE

ELSEVIER

Overflow

- Digital systems operate on a **fixed number of bits**
- Overflow: when result is too big to fit in the available number of bits
- See previous example of 11 + 6

FROM ZERO TO ONE

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two's Complement Numbers

FROM ZERO TO ONE

Sign/Magnitude Numbers

- 1 sign bit, N-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \text{L}\ \ a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 =

  - 6 =

- Range of an N-bit sign/magnitude number:

Sign/Magnitude Numbers

- 1 sign bit, N-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1

$$A : \{a_{N-1}, a_{N-2}, \mathrm{L}\ \ a_2, a_1, a_0\}$$

$$A = (-1)^{a_{n-1}} \sum_{i=0}^{n-2} a_i 2^i$$

- Example, 4-bit sign/mag representations of ± 6:

  +6 = **0110**

  - 6 = **1110**

- Range of an N-bit sign/magnitude number:

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Sign/Magnitude Numbers

- Problems:
  - Addition doesn't work, for example -6 + 6:

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline 10100 \end{array}$$ (wrong!)

  - Two representations of 0 (± 0):

1000

0000

Two's Complement Numbers

- Don't have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

FROM ZERO TO ONE

ELSEVIER

Two's Complement Numbers

- Msb has value of -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number:
- Most negative 4-bit number:
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's comp number:

FROM ZERO TO ONE

Two's Complement Numbers

- Msb has value of -2^{N-1}

$$A = a_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number: **0111**
- Most negative 4-bit number: **1000**
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's comp number:

$[-(2^{N-1}), 2^{N-1}-1]$

FROM ZERO TO ONE
ELSEVIER

"Taking the Two's Complement"

- Flip the sign of a two's complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$

FROM ZERO TO ONE

ELSEVIER

"Taking the Two's Complement"

- Flip the sign of a two's complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$
  1. **1100**
  2. **+ 1**

  1101 = -3_{10}

FROM ZERO TO ONE

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$

- What is the decimal value of 1001_2?

FROM ZERO TO ONE

Two's Complement Examples

- Take the two's complement of $6_{10} = 0110_2$
  1. 1001
  2. \+ 1

  $1010_2 = -6_{10}$

- What is the decimal value of the two's complement number 1001_2?
  1. 0110
  2. \+ 1

  $0111_2 = 7_{10}$, so $1001_2 = -7_{10}$

FROM ZERO TO ONE

ELSEVIER

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

$$\begin{array}{r} 0110 \\ +\ 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two's complement numbers

$$\begin{array}{r} 1110 \\ +\ 0011 \\ \hline \end{array}$$

FROM ZERO TO ONE

ELSEVIER

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

```
  111
   0110
+  1010
------
  10000
```

- Add -2 + 3 using two's complement numbers

```
  111
   1110
+  0011
------
  10001
```

FROM ZERO TO ONE

ELSEVIER

Increasing Bit Width

- **Extend number from *N* to *M* bits (*M* > *N*) :**
  - Sign-extension
  - Zero-extension

FROM ZERO TO ONE

Copyright © 2012 Elsevier

ELSEVIER

Sign-Extension

- Sign bit copied to msb's
- Number value is same

- **Example 1:**
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- **Example 2:**
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011

FROM ZERO TO ONE

Zero-Extension

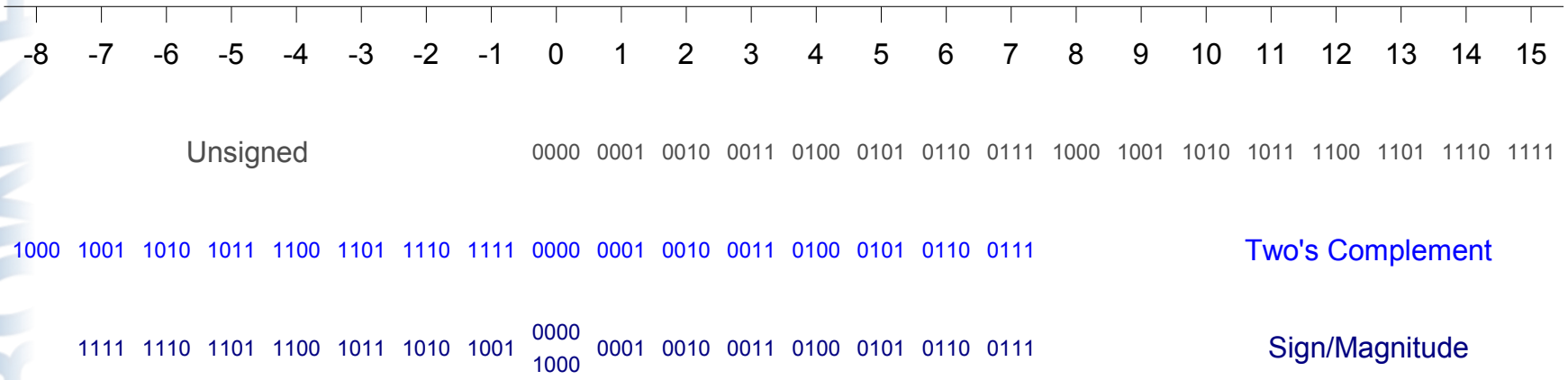
- Zeros copied to msb's
- Value changes for negative numbers

- **Example 1:**
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: 00000011 = 3_{10}
- **Example 2:**
  - 4-bit value = $1011 = -5_{10}$
  - 8-bit zero-extended value: 00001011 = 11_{10}

Number System Comparison

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:



FROM ZERO TO ONE

ELSEVIER

Number Systems

- Numbers we can represent using binary representations
  - **Positive numbers**
    - Unsigned binary
  - **Negative numbers**
    - Two's complement
    - Sign/magnitude numbers
- What about **fractions**?

FROM ZERO TO ONE

ELSEVIER

Numbers with Fractions

- Two common notations:
  - **Fixed-point:** binary point fixed
  - **Floating-point:** binary point floats to the right of the most significant 1

FROM ZERO TO ONE

Fixed-Point Numbers

- 6.75 using 4 integer bits and 4 fraction bits:

$$01101100$$

$$0110.1100$$

$$2^2 + 2^1 + 2^{-1} + 2^{-2} = 6.75$$

  - Binary point is implied
  - The number of integer and fraction bits must be agreed upon beforehand

FROM ZERO TO ONE

Fixed-Point Number Example

- Represent 7.5_{10} using 4 integer bits and 4 fraction bits.

FROM ZERO TO ONE

Fixed-Point Number Example

- Represent 7.5_{10} using 4 integer bits and 4 fraction bits.

01111000

FROM ZERO TO ONE

Signed Fixed-Point Numbers

- **Representations:**
  - Sign/magnitude
  - Two's complement
- **Example:** Represent -7.5_{10} using 4 integer and 4 fraction bits
  - **Sign/magnitude:**
  - **Two's complement:**

FROM ZERO TO ONE

Signed Fixed-Point Numbers

- **Representations:**
  - Sign/magnitude
  - Two's complement
- **Example:** Represent -7.5_{10} using 4 integer and 4 fraction bits
  - **Sign/magnitude:**

    11111000
  - **Two's complement:**

1. +7.5:	01111000
2. Invert bits:	10000111
3. Add 1 to lsb:	+ 1
	10001000

FROM ZERO TO ONE

ELSEVIER

Floating-Point Numbers

- Binary point floats to the right of the most significant 1
- Similar to decimal scientific notation

- For example, write 273_{10} in scientific notation:

$$273 = 2.73 \times 10^2$$

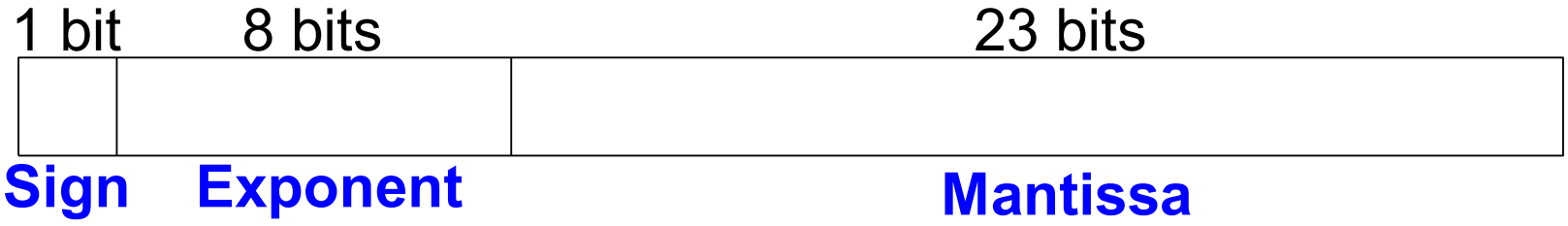
- In general, a number is written in scientific notation as:

$$\pm M \times B^E$$

  - **M** = mantissa
  - **B** = base
  - **E** = exponent
  - In the example, M = 2.73, B = 10, and E = 2

FROM ZERO TO ONE

Floating-Point Numbers



- **Example:** represent the value 228_{10} using a 32-bit floating point representation

We show three versions –final version is called the **IEEE 754 floating-point standard**

FROM ZERO TO ONE

ELSEVIER

Floating-Point Representation 1

1. Convert decimal to binary (**don't reverse steps 1 & 2!**):

   $228_{10} = 11100100_2$

2. Write the number in "binary scientific notation":

   $11100100_2 = 1.11001_2 \times 2^7$

3. Fill in each field of the 32-bit floating point number:
   - The sign bit is positive (0)
   - The 8 exponent bits represent the value 7
   - The remaining 23 bits are the mantissa

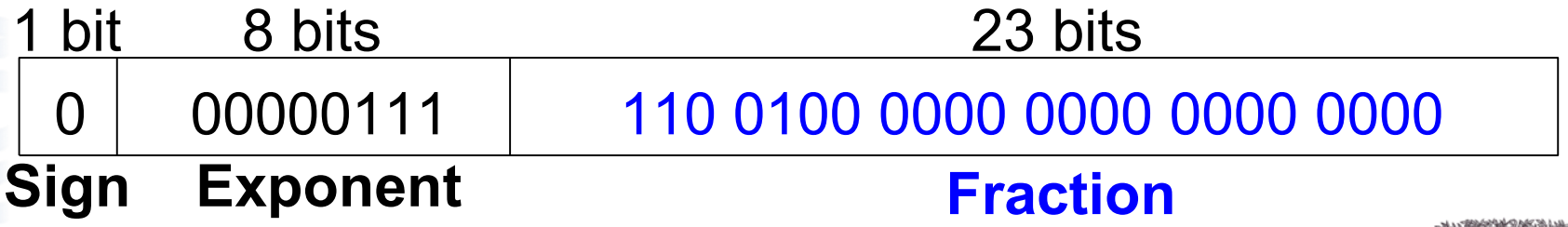
1 bit	8 bits	23 bits
0	00000111	11 1001 0000 0000 0000 0000
Sign	**Exponent**	**Mantissa**

Note: There should be an extra "0" bit on the right of the Mantissa.

FROM ZERO TO ONE
ELSEVIER

Floating-Point Representation 2

- First bit of the mantissa is always 1:
  - $228_{10} = 11100100_2 = \mathbf{1}.11001 \times 2^7$
- So, no need to store it: *implicit leading 1*
- Store just fraction bits in 23-bit field



FROM ZERO TO ONE

ELSEVIER

Floating-Point Representation 3

- *Biased exponent*: bias = 127 (01111111_2)
  - Biased exponent = bias + exponent
  - Exponent of 7 is stored as:

$$127 + 7 = 134 = 0x10000110_2$$

- The **IEEE 754 32-bit floating-point representation** of 228_{10}

1 bit	8 bits	23 bits
0	10000110	110 0100 0000 0000 0000 0000
Sign	**Biased Exponent**	**Fraction**

in hexadecimal: **0x43640000**

FROM ZERO TO ONE

ELSEVIER

Floating-Point Example

Write -58.25_{10} in floating point (IEEE 754)

FROM ZERO TO ONE

Floating-Point Example

Write -58.25_{10} in floating point (IEEE 754)

1. Convert decimal to binary:

   $58.25_{10} = \mathbf{111010.01_2}$

2. Write in binary scientific notation:

   $\mathbf{1.1101001 \times 2^5}$

3. Fill in fields:

   Sign bit: 1 (negative)
   8 exponent bits: (127 + 5) = 132 = $\mathbf{10000100_2}$
   23 fraction bits: **110 1001 0000 0000 0000 0000**

1 bit	8 bits	23 bits
1	100 0010 0	110 1001 0000 0000 0000 0000
Sign	**Exponent**	**Fraction**

in hexadecimal: **0xC2690000**

FROM ZERO TO ONE
ELSEVIER

Floating-Point: Special Cases

Number	Sign	Exponent	Fraction
0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000
- ∞	1	11111111	00000000000000000000000
NaN	X	11111111	non-zero

FROM ZERO TO ONE

ELSEVIER

Floating-Point Precision

- **Single-Precision:**
  - 32-bit
  - 1 sign bit, 8 exponent bits, 23 fraction bits
  - bias = 127
- **Double-Precision:**
  - 64-bit
  - 1 sign bit, 11 exponent bits, 52 fraction bits
  - bias = 1023

FROM ZERO TO ONE

Floating-Point: Rounding

- **Overflow:** number too large to be represented
- **Underflow:** number too small to be represented
- **Rounding modes:**
  - Down
  - Up
  - Toward zero
  - To nearest
- **Example:** round 1.100101 (1.578125) to only 3 fraction bits
  - Down: 1.100
  - Up: 1.101
  - Toward zero: 1.100
  - To nearest: 1.101 (1.625 is closer to 1.578125 than 1.5 is)

FROM ZERO TO ONE

Floating-Point Addition

1. Extract exponent and fraction bits
2. Prepend leading 1 to fraction to form mantissa
3. Compare exponents
4. Shift smaller mantissa if necessary
5. Add mantissas
6. Normalize mantissa and adjust exponent if necessary
7. Round result
8. Assemble exponent and fraction back into floating-point format

FROM ZERO TO ONE

Floating-Point Addition Example

Add the following floating-point numbers:

0x3FC00000

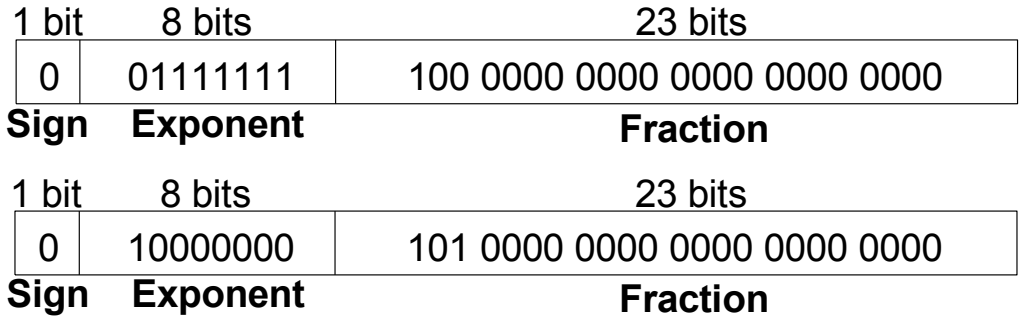
0x40500000

FROM ZERO TO ONE

ELSEVIER

Floating-Point Addition Example

1. Extract exponent and fraction bits



For first number (N1): S = 0, E = 127, F = .1

For second number (N2): S = 0, E = 128, F = .101

2. Prepend leading 1 to form mantissa

N1: 1.1

N2: 1.101

FROM ZERO TO ONE
ELSEVIER

Floating-Point Addition Example

3. Compare exponents

127 – 128 = -1, so shift N1 right by 1 bit

4. Shift smaller mantissa if necessary

shift N1's mantissa: $1.1 >> 1 = 0.11$ $(\times 2^1)$

5. Add mantissas

$$\begin{array}{r} 0.11 \times 2^1 \\ +\ 1.101 \times 2^1 \\ \hline 10.011 \times 2^1 \end{array}$$

FROM ZERO TO ONE

ELSEVIER

Floating Point Addition Example

6. **Normalize mantissa and adjust exponent if necessary**

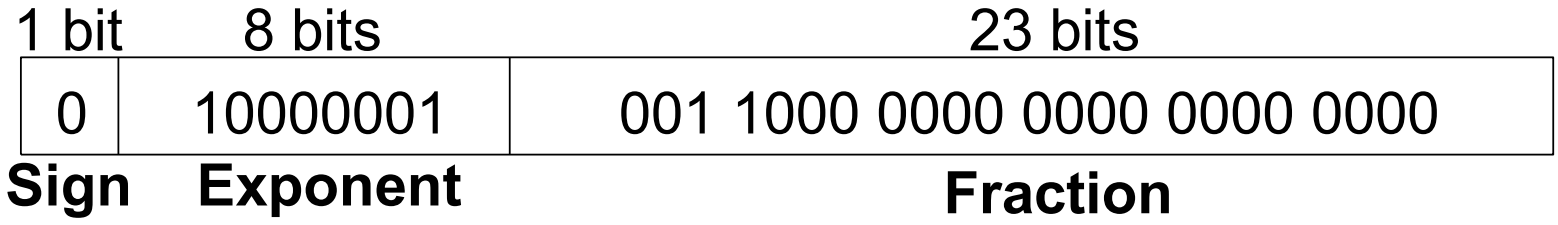
   $10.011 \times 2^1 = 1.0011 \times 2^2$

7. **Round result**

   No need (fits in 23 bits)

8. **Assemble exponent and fraction back into floating-point format**

   $S = 0, E = 2 + 127 = 129 = 10000001_2, F = 001100..$



in hexadecimal: **0x40980000**

FROM ZERO TO ONE
ELSEVIER