DIGITAL DESIGN & COMPUTER ORGANISATION

Floating Point

Sudarshan T S B., Ph.D.
Department of Computer Science
& Engineering

DIGITAL DESIGN & COMPUTER ORGANISATION

Floating Point

Sudarshan T S B., Ph.D.
Department of Computer Science & Engineering

FLOATING POINT

Course Outline

- Digital Design
  - Combinational logic design
  - Sequential logic design
    - Floating Point

- Computer Organisation
  - Architecture (microprocessor instruction set)
  - Microarchitecture (microprocessor operation)

Concepts covered

- Floating Point Representation

Floating Point

Not Just Integers

Floating Point

Not Just Integers

- Real numbers can be represented using:
  - Fixed point
  - Floating point

Floating Point

Not Just Integers

- Real numbers can be represented using:
  - Fixed point
  - Floating point

- Fixed point notation is where the decimal point is fixed and numbers to the right of decimal point are the fraction portion and to the left is the integer portion.
  - Limited by the digits used
  - Not suitable to represent very small are very large numbers

FLOATING POINT

Not Just Integers

- Real numbers can be represented using:
  - Fixed point
  - Floating point

- Fixed point notation is where the decimal point is fixed and numbers to the right of decimal point are the fraction portion and to the left is the integer portion.
  - Limited by the digits used
  - Not suitable to represent very small are very large numbers

- Programming languages support fraction called **floating point** numbers
  - Example: 3.14159265… (π); 2.71828… (e)
  - Data type used float , double

Fixed Point Example

Fixed Point Example

- Represent 6.75 using 4 integer bits and 4 fraction bits:
  - 6 => 0110 ($2^2+2^{1)}$
  - 0.75 => 0.1100 (2^{-1} + 2^{-2})
  - 6.75 => 0110.1100

Fixed Point Example

- Represent 6.75 using 4 integer bits and 4 fraction bits:
  - 6 => 0110 ($2^2+2^{1)}$
  - 0.75 => 0.1100 ($2^{-1} + 2^{-2}$)
  - 6.75 => 0110.1100

  - Here binary point is implied and the number of bits used is decided before hand
  - Fixed point
  - Floating point

Fixed Point Example

- Represent 6.75 using 4 integer bits and 4 fraction bits:
  - 6 => 0110 (2^2+2^1)
  - 0.75 => 0.1100 ($2^{-1} + 2^{-2}$)
  - 6.75 => 0110.1100

  - Here binary point is implied and the number of bits used is decided before hand
  - Fixed point
  - Floating point

- Represent -7.5 using 4 integer and 4 fraction bits
  - +7.5 => 0111.1000
  - 2's complement -7.5 => 1000.1000

FLOATING POINT

Fixed Point Example

Fixed Point Example

- Perform the following operation: 7.5 – 0.625 => 7.5 + (-0.625)
  - 7.5 => 0111.1000
  - -0.625 => 111.0110 (2'scomplement)
  - 0111.1000 + 1111.0110 = 0110.1110 (6.875)

FLOATING POINT

Fixed Point Example

- Perform the following operation: 7.5 – 0.625 => 7.5 + (-0.625)
  - 7.5 => 0111.1000
  - -0.625 => 111.0110 (2'scomplement)
  - 0111.1000 + 1111.0110 = 0110.1110 (6.875)

- The range and accuracy is very limited.
  - Ex: 8.9375 + 8.3125 = 17.2495
  - 8.9375 => 1000.1111
  - 8.3125 => 1000.0101
  - Add: 0001.0100 (1.25) which is the result of limited range and limited accuracy

Fixed Point Example

- Perform the following operation: 7.5 – 0.625 => 7.5 + (-0.625)
  - 7.5 => 0111.1000
  - -0.625 => 111.0110 (2'scomplement)
  - 0111.1000 + 1111.0110 = 0110.1110 (6.875)

- The range and accuracy is very limited.
  - Ex: 8.9375 + 8.3125 = 17.2495
  - 8.9375 => 1000.1111
  - 8.3125 => 1000.0101
  - Add: 0001.0100 (1.25) which is the result of limited range and limited accuracy

- How to increase the range and improve the accuracy?
  - Go for Floating Point Representation

Floating Point

Not Just Integers

FLOATING POINT

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers

FLOATING POINT

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers
- We use scientific notation to represent these numbers
  - $\pm\, d.f_1f_2f_3\ldots \times 10^{\pm e_1 e_2}$
  - $\pm\, M \times B^{\pm E}$

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers
- We use scientific notation to represent these numbers
  - $\pm\, d.f_1f_2f_3\ldots \times 10^{\pm e_1e_2}$
  - $\pm\, M \times B^{\pm E}$
- This representation is to include very small numbers like 1.0×10^{-23} and very larger numbers like 9.546×10^{12}

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers
- We use scientific notation to represent these numbers
  - $\pm\, d.f_1 f_2 f_3 \ldots \times 10^{\pm e_1 e_2}$
  - $\pm\, M \times B^{\pm E}$
- This representation is to include very small numbers like 1.0×10^{-23} and very larger numbers like 9.546×10^{12}
- Floating point numbers should be ***normalized***
  - Use one non-zero digit as integer
  - In decimal it will be from 1 to 9
  - In binary this should be 1

FLOATING POINT

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers

- We use scientific notation to represent these numbers
  - $\pm\, d.f_1f_2f_3\ldots\ x\ 10^{\pm\, e_1 e_2}$
  - $\pm\, M \text{ x } B^{\pm E}$

- This representation is to include very small numbers like $1.0 \text{ x } 10^{-23}$ and very larger numbers like $9.546 \text{ x } 10^{12}$

- Floating point numbers should be ***normalized***
  - Use one non-zero digit as integer
  - In decimal it will be from 1 to 9
  - In binary this should be 1
  - Ex:

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers
- We use scientific notation to represent these numbers
  - $\pm\, d.f_1f_2f_3\ldots\ x\ 10^{\pm\, e_1 e_2}$
  - $\pm\, M \times B^{\pm E}$
- This representation is to include very small numbers like 1.0×10^{-23} and very larger numbers like 9.546×10^{12}
- Floating point numbers should be ***normalized***
  - Use one non-zero digit as integer
  - In decimal it will be from 1 to 9
  - In binary this should be 1
  - Ex:

    Normalised floating point: 2.234×10^{3} or 1.101×2^{-4}

FLOATING POINT

Not Just Integers

- Floating point notation is used to represent real numbers which are from small to large numbers
- We use scientific notation to represent these numbers
  - $\pm\ d.f_1f_2f_3\ldots \times 10^{\pm e_1 e_2}$
  - $\pm\ \mathrm{M} \times \mathrm{B}^{\pm \mathrm{E}}$
- This representation is to include very small numbers like 1.0×10^{-23} and very larger numbers like 9.546×10^{12}
- Floating point numbers should be ***normalized***
  - Use one non-zero digit as integer
  - In decimal it will be from 1 to 9
  - In binary this should be 1
  - Ex:

    Normalised floating point: 2.234×10^{3} or 1.101×2^{-4}

    Non-normalized floating point: 0.0234×10^{5} or 110.1×2^{-6}

Floating Point

IEEE 754-2008 Standard

IEEE 754-2008 Standard

- IEEE Standard defines structure of floating point number representation

IEEE 754-2008 Standard

- IEEE Standard defines structure of floating point number representation
- Developed in response to divergence of representations and arithmetic operations
    - Portability issues for scientific code
    - Universally adopted

IEEE 754-2008 Standard

- IEEE Standard defines structure of floating point number representation
- Developed in response to divergence of representations and arithmetic operations
  - Portability issues for scientific code
  - Universally adopted
- Defines four representations:
  - Single Precision (32-bits)
  - Double Precision (64-bits)
  - Extended Double Precision 10 bytes (80-bits)
  - Quadruple Precision 16 bytes(128-bits)

FLOATING POINT

IEEE 754-2008 Standard

- IEEE Standard defines structure of floating point number representation
- Developed in response to divergence of representations and arithmetic operations
  - Portability issues for scientific code
  - Universally adopted
- Defines four representations:
  - Single Precision (32-bits)
  - Double Precision (64-bits)
  - Extended Double Precision 10 bytes (80-bits)
  - Quadruple Precision 16 bytes(128-bits)
- Real Number is represented in IEEE 754-2008 standard as three parts:
  - Sign bit
  - Exponent bits
  - Mantissa bits or Significand bits

Floating Point

IEEE 754-2008 Standard (Single Precision)

Floating Point

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						

1 bit   8 bits   23 bits

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						

1 bit   8 bits   23 bits

- Sign bit 0 indicates positive and 1 indicates negative number

Floating Point

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						
1 bit	8 bits								23 bits																						

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						
1 bit	8 bits								23 bits																						

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number
- Exponent represents range of the numbers that shall be represented

Floating Point

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						

1 bit — 8 bits — 23 bits

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number
- Exponent represents range of the numbers that shall be represented
- General form: $\pm\ 1.\text{Mantissa} \times 2^{\text{Exponent}}$

Floating Point

IEEE 754-2008 Standard (Single Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						
1 bit	8 bits								23 bits																						

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number
- Exponent represents range of the numbers that shall be represented
- General form: $\pm$ **1.Mantissa x 2**$^{\text{Exponent}}$
- For Single precision (32-bits) representation:
  - Biased Exponent is 8-bits
  - Mantissa is 23 bits

Floating Point

IEEE 754-2008 Standard (Double Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent												fraction																		

1 bit 11 bits 20 bits

fraction (continued)

32 bits

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number
- Exponent represents range of the numbers that shall be represented
- General form: **± 1.Mantissa x 2$^{\text{Exponent}}$**

Floating Point

IEEE 754-2008 Standard (Double Precision)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent											fraction																			
1 bit	11 bits											20 bits																			

fraction (continued)
32 bits

- Sign bit 0 indicates positive and 1 indicates negative number
- Mantissa represents fraction and signifies accuracy of the number
- Exponent represents range of the numbers that shall be represented
- General form: $\pm$ **1.Mantissa x 2^{Exponent}**
- For Double precision (64-bits) representation:
  - Biased Exponent is 11-bits
  - Mantissa is 52 bits

Biased Exponent

Biased Exponent

- Biased Exponent, BE = Bias + Exponent

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.
- Bias = 127 => $(2^{n-1} - 1)$

Floating Point

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.
- Bias = 127 => $(2^{n-1} - 1)$
- Therefore Range of Actual exponent that could be represented is:
  - Min = 1 – 127 = -126
  - Max = 254 – 127 = 127
  - So range is from -126 to +127

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.
- Bias = 127 => ($2^{n-1} - 1$)
- Therefore Range of Actual exponent that could be represented is:
  - Min = 1 – 127 = -126
  - Max = 254 – 127 = 127
  - So range is from -126 to +127
- FP Representation:

Floating Point

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.
- Bias = 127 => $(2^{n-1} - 1)$
- Therefore Range of Actual exponent that could be represented is:
  - Min = 1 – 127 = -126
  - Max = 254 – 127 = 127
  - So range is from -126 to +127
- FP Representation:

$$N = (-1)^{s} * (1+.M)*2^{(BE-Bias)}$$

Floating Point

Biased Exponent

- Biased Exponent, BE = Bias + Exponent
- Actual Exponent, E = Biased Exponent – Bias
- Recall for Single precision Biased Exponent is 8-bits (Range: 0 to 255)
- BE = 0 and BE = 255 are reserved for special use
- So, BE = 1 to 254 are used for normalized floating point numbers.
- Bias = 127 => $(2^{n-1} – 1)$
- Therefore Range of Actual exponent that could be represented is:
  - Min = 1 – 127 = -126
  - Max = 254 – 127 = 127
  - So range is from -126 to +127
- FP Representation:

$$N = (-1)^s * (1+.M)*2^{(BE-Bias)}$$

Bias = 127 for SP
Bias = 1023 for DP

FP Example

FP Example

What is the value of the following number:

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

Sign = 0; Positive number

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

Sign = 0; Positive number

Biased Exponent = $(1000\ 0110)_2$ = 134;

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

Sign = 0; Positive number

Biased Exponent = $(1000\ 0110)_2$ = 134;

Actual Exponent = 134 – 127 = 7

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

Sign = 0; Positive number

Biased Exponent = $(1000\ 0110)_2$ = 134;

Actual Exponent = 134 – 127 = 7

Mantissa = $(1.\ 1100\ 10\ldots000)_2$ = 1.78125 (1. is implicit)

FP Example

What is the value of the following number:

0	100 0011 0	110 0100 0000 0000 0000 0000

In Hexadecimal this is represented as **0x43640000**

Solution:

Sign = 0; Positive number

Biased Exponent = $(1000\ 0110)_2$ = 134;

Actual Exponent = 134 – 127 = 7

Mantissa = $(1.\ 1100\ 10...000)_2$ = 1.78125 (1. is implicit)

So, the value of the decimal = 1.1100100×2^7 = 11100100 = **228**

FP Example

FP Example

What is the value of the following number:

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

Sign = 1;

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

Sign = 1;

Biased Exponent = $(0111\ 1100)_2$ = 124;

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

Sign = 1;

Biased Exponent = $(0111\ 1100)_2$ = 124;

Actual Exponent = 124 – 127 = -3

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

Sign = 1;

Biased Exponent = $(0111\ 1100)_2$ = 124;

Actual Exponent = 124 – 127 = -3

Mantissa = $(1.\ 0100\ 00…000)_2$ = 1.25 (1. is implicit)

FP Example

What is the value of the following number:

1	011 1110 0	010 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xBE200000**

Solution:

Sign = 1;

Biased Exponent = $(0111\ 1100)_2$ = 124;

Actual Exponent = 124 – 127 = -3

Mantissa = $(1.\ 0100\ 00...000)_2$ = 1.25 (1. is implicit)

So, the value of the decimal = -1.25×2^{-3} = - 0.15625

FLOATING POINT

FP Example

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

FP Example

Write -58.25$_{10}$ in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

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**

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

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	100 0010 0	110 1001 0000 0000 0000 0000

FP Example

Write -58.25_{10} in Single Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

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	100 0010 0	110 1001 0000 0000 0000 0000

In Hexadecimal this is represented as **0xC2690000**

FP Example

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

   58.25_{10} = **111010.01_2**

2. Write in normalized scientific notation:

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

3. Fill in fields:

   Sign bit: **1** (negative)

   11 exponent bits: (1023 + 5) = 1028 = $\mathbf{100\ 0000\ 0100_2}$

   52 fraction bits: **1101 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000**

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

3. Fill in fields:

   Sign bit: **1** (negative)

   11 exponent bits: $(1023 + 5) = 1028 =$ $\mathbf{100\ 0000\ 0100_2}$

   52 fraction bits: **1101 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000**

1	100 0000 0100	1101 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

FP Example

Write -58.25_{10} in Double Precision Floating Point (IEEE 754)

1. Convert decimal to binary:

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

2. Write in normalized scientific notation:

   $\mathbf{1.1101001 \times 2^5}$ (1. is implicit)

3. Fill in fields:

   Sign bit: **1** (negative)
   11 exponent bits: (1023 + 5) = 1028 = **$100\ 0000\ 0100_2$**
   52 fraction bits: **1101 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000**

1	100 0000 0100	1101 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

In Hexadecimal this is represented as **0xC04D200000000000**

Floating Point

Smallest and Largest Normalised FP value

Smallest and Largest Normalised FP value

Smallest and Largest Normalised FP value

Single Precision FP:

Smallest and Largest Normalised FP value

Single Precision FP:

Exponents 00000000 and 11111111 are reserved

Smallest and Largest Normalised FP value

Single Precision FP:

Exponents 00000000 and 11111111 are reserved

Smallest value

Exponent: 00000001

$\Rightarrow$ Actual Exponent = 1 – 127 = –126

Fraction: 000…00 $\Rightarrow$ significand = 1.0

$\pm 1.0 \times 2^{-126} \approx \pm 1.17549\ldots \times 10^{-38}$

Smallest and Largest Normalised FP value

Single Precision FP:

Exponents 00000000 and 11111111 are reserved

Smallest value

Exponent: 00000001

$\Rightarrow$ Actual Exponent = 1 – 127 = –126

Fraction: 000…00 $\Rightarrow$ significand = 1.0

$\pm 1.0 \times 2^{-126} \approx \pm 1.17549\ldots \times 10^{-38}$

Largest value

Biased Exponent: 11111110

$\Rightarrow$ Actual Exponent = 254 – 127 = +127

Fraction: 111…11 $\Rightarrow$ significand ≈ 2.0

$\pm 2.0 \times 2^{+127} = 2^{-128} \approx \pm 3.4028\ldots \times 10^{+38}$

FLOATING POINT

Special Cases

Special Cases

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

Floating Point

Special Cases

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

*NaN is Not a Number

Special Cases

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

*NaN is Not a Number

Ex: ÷ by zero, √-ve no.

FLOATING POINT
Special Cases

Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

Source: Computer Organisation & Design by
Patterson & Hennessy, Morgan Kaufmann

Rounding Modes

Rounding Modes

- **Overflow:** number too large to be represented

Rounding Modes

- **Overflow:** number too large to be represented
- **Underflow:** number too small to be represented

Rounding Modes

- **Overflow:** number too large to be represented
- **Underflow:** number too small to be represented
- **Rounding modes:**
  - Down
  - Up
  - Toward zero
  - To nearest

Rounding Modes

- **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)

Floating Point

Think about it

- What are the largest normalized double precision FP numbers?
  - Hint: double precision exponent is 11 bits and mantissa is 52 bits
- What is the relative precision in terms of decimal fractional digits that single precision and double precision offer?
  - Hint: mantissa bits
- An example to represent denormalized valid floating point number?
  - Hint: Biased Exponent = 0 & Mantissa = Nonzero

THANK YOU

Sudarshan T S B. Ph.D.,
Department of Computer Science & Engineering

sudarshan@pes.edu

+91 80 6666 3333 Extn 215