IEEE Ploating Point Standard 754

G.E. Antoniou

MSU ... Computer Science Department

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation
- Floating Point Numbers (FPN) are processed in the Floating Point Unit (FPU)

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation
- Floating Point Numbers (FPN) are processed in the Floating Point Unit (FPU)
- The "decimal" decimal point in FPN is variable or is automatically adjusted or it **floats**

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation
- Floating Point Numbers (FPN) are processed in the Floating Point Unit (FPU)
- The "decimal" decimal point in FPN is variable or is automatically adjusted or it **floats**
- ....

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation
- Floating Point Numbers (FPN) are processed in the Floating Point Unit (FPU)
- The "decimal" decimal point in FPN is variable or is automatically adjusted or it **floats**
- ....
- Initially a FPN is defined as,

IEEE Floating Point Standard 754

Established in 1985 as homogeneous standard for floating point arithmetic

- In scientific calculations the range of the numbers can be very large or very small ...
- These numbers can be expressed with Floating Point Notation
- Floating Point Numbers (FPN) are processed in the Floating Point Unit (FPU)
- The "decimal" decimal point in FPN is variable or is automatically adjusted or it **floats**
- ....
- Initially a FPN is defined as,
- $$FPNumber = Fraction \times Base^{Exponent}$$

Example: Decimal Numbers

$$\begin{aligned}
12345 &= 12345.0 \times 10^0 \\
&= 1234.5 \times 10^1 \\
&= 123.45 \times 10^2 \\
&= 12.345 \times 10^3 \\
&= 1.2345 \times 10^4
\end{aligned}$$

Example: Decimal Numbers

$$
\begin{aligned}
12345 &= 12345.0 \times 10^0 \\
&= 1234.5 \times 10^1 \\
&= 123.45 \times 10^2 \\
&= 12.345 \times 10^3 \\
&= 1.2345 \times 10^4
\end{aligned}
$$

- The decimal number:

Example: Decimal Numbers

$$\begin{aligned} 12345 &= 12345.0 \times 10^0 \\ &= 1234.5 \times 10^1 \\ &= 123.45 \times 10^2 \\ &= 12.345 \times 10^3 \\ &= 1.2345 \times 10^4 \end{aligned}$$

- The decimal number:
- $$+1234.567_{10} = +.1234567 \times 10^4$$

Example: Decimal Numbers

$$\begin{aligned} 12345 &= 12345.0 \times 10^0 \\ &= 1234.5 \times 10^1 \\ &= 123.45 \times 10^2 \\ &= 12.345 \times 10^3 \\ &= 1.2345 \times 10^4 \end{aligned}$$

- The decimal number:
- $$+1234.567_{10} = +.1234567 \times 10^4$$
- In general

Example: Decimal Numbers

$$
\begin{aligned}
12345 &= 12345.0 \times 10^0 \\
&= 1234.5 \times 10^1 \\
&= 123.45 \times 10^2 \\
&= 12.345 \times 10^3 \\
&= 1.2345 \times 10^4
\end{aligned}
$$

- The decimal number:
- $$+1234.567_{10} = +.1234567 \times 10^4$$
- In general
- $$FPNumber = F \times 10^{Exponent}$$

Example: Binary Numbers

Example: Binary Numbers

- 101_2

Example: Binary Numbers

- 101_2
- 101.0×2^0

Example: Binary Numbers

- 101_2
- 101.0×2^0
- 10.1×2^1

Example: Binary Numbers

- 101_2
- 101.0×2^0
- 10.1×2^1
- 1.01×2^2

Example: Binary Numbers

- 101_2
- 101.0×2^0
- 10.1×2^1
- 1.01×2^2
- ....
- Therefore,

Example: Binary Numbers

- 101_2
- 101.0×2^0
- 10.1×2^1
- 1.01×2^2
- ....
- Therefore,
-

$$FPN_2 = F \times 2^{Exponent}$$

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative
- Therefore our FPN formula becomes,

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative
- Therefore our FPN formula becomes,
- $$FPN = (-1)^S \times F \times 2^{Exponent}$$

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative
- Therefore our FPN formula becomes,
- $$FPN = (-1)^S \times F \times 2^{Exponent}$$
- where,

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative
- Therefore our FPN formula becomes,
- $$FPN = (-1)^S \times F \times 2^{Exponent}$$
- where,
- S = Sign

Signed (Positive–Negative) Numbers

Signed binary numbers ... the leftmost bit is:

- 0 – the number is positive
- 1 – the number is negative
- Therefore our FPN formula becomes,
- $$FPN = (-1)^S \times F \times 2^{Exponent}$$
- where,
- S = Sign
- F = Fraction

FPN Normalization

FPN Normalization

- Normalization ensures a unique floating–point representation of each number

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:
- Not Normalized number

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:
- Not Normalized number
  - $0.1_{10} \times 10^{-6}$

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:
- Not Normalized number
  - $0.1_{10} \times 10^{-6}$
- Normalized number

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:
- Not Normalized number
  - $0.1_{10} \times 10^{-6}$
- Normalized number
  - $1.0_{10} \times 10^{-9}$

FPN Normalization

- Normalization ensures a unique floating–point representation of each number
- Normalized number: A number in scientific notation that **has no leading zeros**
- ....
- EXAMPLES:
- Not Normalized number
  - $0.1_{10} \times 10^{-6}$
- Normalized number
  - $1.0_{10} \times 10^{-9}$
  - $1.0_{2} \times 2^{-1}$

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)
- ....

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)
- ....
- To have one extra representation bit we can do the following:

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)
- ....
- To have one extra representation bit we can do the following:
- Shift left by one bit

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)
- ....
- To have one extra representation bit we can do the following:
- Shift left by one bit
- Leading 1 is discarded

Packing, Hidden 1 Principle

To pack (include) more bits ... the **IEEE 754 standard** makes the leading 1 bit of the normalized binary numbers implicit.

- **Hidden 1 principle**
- ....
- Using the normalization, the leading bit is always nonzero or 1
- Since the leading bit is always 1, why carry it ?
- (There is no need to store it)
- ....
- To have one extra representation bit we can do the following:
- Shift left by one bit
- Leading 1 is discarded
- To get back the initial number .... “ put back the 1 ”.

Updated Formula

To represent the “hidden 1” and the “Fraction” use the formula:

Updated Formula

To represent the "hidden 1" and the "Fraction" use the formula:

- $$F = 1 + significant$$

Updated Formula

To represent the "hidden 1" and the "Fraction" use the formula:

- $$F = 1 + significant$$
- $$F = 1.significant$$

Updated Formula

To represent the "hidden 1" and the "Fraction" use the formula:

- $$F = 1 + significant$$
- $$F = 1.significant$$
- Using the above Hidden 1 principle our FPN formula now becomes:

Updated Formula

To represent the "hidden 1" and the "Fraction" use the formula:

- $$F = 1 + significant$$
- $$F = 1.significant$$
- Using the above Hidden 1 principle our FPN formula now becomes:
- $$FPN = (-1)^S \times (1.significant) \times 2^{Exponent}$$

Updated Formula

To represent the "hidden 1" and the "Fraction" use the formula:

- $$F = 1 + significant$$
- $$F = 1.significant$$
- Using the above Hidden 1 principle our FPN formula now becomes:
- $$FPN = (-1)^S \times (1.significant) \times 2^{Exponent}$$
- $$FPN = (-1)^S \times (1 + significant) \times 2^{Exponent}.$$

Precision Formats: 32-bit

IEEE–754 Floating Point Standard

Precision Formats: 32-bit

IEEE–754 Floating Point Standard

- Single (32-bit) Precision Format (Occupies one four byte word)
  - S = 1
  - E = 8
  - F = 23

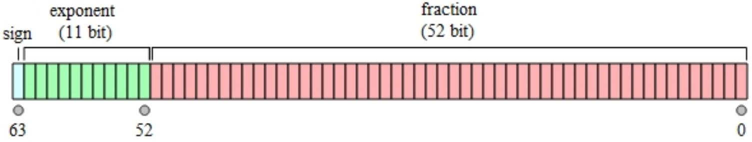
Precision Formats: 64-bit

IEEE–754 Floating Point Standard

Precision Formats: 64-bit

IEEE–754 Floating Point Standard

- Double (64-bit) Precision Format
  - S = 1
  - E = 11
  - F = 52



Biased Notation

Biased Notation

- Biasing the exponent improves accuracy with very small numbers

Biased Notation

- Biasing the exponent improves accuracy with very small numbers
- We represent ....

Biased Notation

- Biasing the exponent improves accuracy with very small numbers
- We represent ....
- the most negative exponent as: 00.....00

Biased Notation

- Biasing the exponent improves accuracy with very small numbers
- We represent ....
- the most negative exponent as: 00.....00
- the most positive as: 11....11

Biased Notation

- Biasing the exponent improves accuracy with very small numbers
- We represent ....
- the most negative exponent as: 00.....00
- the most positive as: 11....11
- The bias is needed to represent a negative exponent without allocating a sign bit.

IEEE–754 Standard; (Single Precision)

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$

- Solving for the Exponent yields,

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$
- Solving for the Exponent yields,
- $$Exponent = E - 127$$

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$
- Solving for the Exponent yields,
- $$Exponent = E - 127$$
- Therefore our new FPN formula takes the form:

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$
- Solving for the Exponent yields,
- $$Exponent = E - 127$$
- Therefore our new FPN formula takes the form:
- $$FPN = (-1)^S \times (1 + significand) \times 2^{E-127}$$

IEEE–754 Standard; (Single Precision)

1. The IEEE 754 standard specifies the exponent in the excess–127 format or bias for Single Precision
2. In this format the number 127 is added to the value of the actual exponent so that,

- $$E = Exponent + 127$$
- Solving for the Exponent yields,
- $$Exponent = E - 127$$
- Therefore our new FPN formula takes the form:
- $$FPN = (-1)^S \times (1 + significand) \times 2^{E-127}$$
- $$FPN = (-1)^S \times (1.significand) \times 2^{E-127}.$$

IEEE–754 Standard; (Double Precision)

IEEE–754 Standard; (Double Precision)

1. The IEEE 754 standard specifies the exponent in the excess–1023 format or bias for Double Precision

IEEE–754 Standard; (Double Precision)

1. The IEEE 754 standard specifies the exponent in the excess–1023 format or bias for Double Precision
2. In this format the number 1023 is added to the value of the actual exponent so that,

IEEE–754 Standard; (Double Precision)

1. The IEEE 754 standard specifies the exponent in the excess–1023 format or bias for Double Precision
2. In this format the number 1023 is added to the value of the actual exponent so that,

- $$FPN = (-1)^S \times (1 + significand) \times 2^{E-1023}$$

IEEE–754 Standard; (Double Precision)

1. The IEEE 754 standard specifies the exponent in the excess–1023 format or bias for Double Precision
2. In this format the number 1023 is added to the value of the actual exponent so that,

- $$FPN = (-1)^S \times (1 + significand) \times 2^{E-1023}$$
- $$FPN = (-1)^S \times (1.significand) \times 2^{E-1023}.$$

Largest and Smallest Values

Largest and Smallest Values

1. Single Precision

Largest and Smallest Values

1. Single Precision
   - Largest normalized value: $2^{127} = \pm 3.4 \times 10^{38}$

Largest and Smallest Values

1. Single Precision
   - Largest normalized value: $2^{127} = \pm 3.4 \times 10^{38}$
   - Smallest normalized value: $2^{-126} = \pm 1.18 \times 10^{-38}$

Largest and Smallest Values

1. Single Precision
   - Largest normalized value: $2^{127} = \pm 3.4 \times 10^{38}$
   - Smallest normalized value: $2^{-126} = \pm 1.18 \times 10^{-38}$
2. Double Precision

Largest and Smallest Values

1. Single Precision
   - Largest normalized value: $2^{127} = \pm 3.4 \times 10^{38}$
   - Smallest normalized value: $2^{-126} = \pm 1.18 \times 10^{-38}$
2. Double Precision
   - Largest normalized value: $2^{1023} = \pm 2.2250738585072020 10^{-308}$

Largest and Smallest Values

1. Single Precision
   - Largest normalized value: $2^{127} = \pm 3.4 \times 10^{38}$
   - Smallest normalized value: $2^{-126} = \pm 1.18 \times 10^{-38}$
2. Double Precision
   - Largest normalized value: $2^{1023} = \pm 2.225073858507202010^{-308}$
   - Smallest normalized value: 2^{-1022}

Examples ...

Illustrative examples follow ...

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$
- $S = 1$

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$
- $\mathrm{S} = 1$
- $E = 127 + 2 = 129_{10}$

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$
- $\mathrm{S} = 1$
- $E = 127 + 2 = 129_{10}$
- The binary equivalent of 129_{10} is 10000001

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$
- $\mathrm{S} = 1$
- $E = 127 + 2 = 129_{10}$
- The binary equivalent of 129_{10} is 10000001
- Therefore

Example: -5_{10}

1. Given the Decimal Number: -5_{10}. Find the FPN representation in single precision notation (127 bias).

- $5 = 101 = 1.01 \times 2^2$
- Exponent $= 2$
- $S = 1$
- $E = 127 + 2 = 129_{10}$
- The binary equivalent of 129_{10} is 10000001
- Therefore
- $$\overbrace{1}^{1Bits}\quad\overbrace{10000001}^{8Bits}\quad\overbrace{01\,\underbrace{000000000000000000000}_{21Bits}}^{23Bits}$$

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28 = 11100 = 1.11 \times 2^4$

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28 = 11100 = 1.11 \times 2^4$
- $E = 127 + 4$

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28 = 11100 = 1.11 \times 2^4$
- $E = 127 + 4$
- The binary equivalent of 131_{10} is 10000011

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28 = 11100 = 1.11 \times 2^4$
- $E = 127 + 4$
- The binary equivalent of 131_{10} is 10000011
- Therefore

Example: -28_{10}

2. Given the Decimal Number: -28_{10}. Find the FPN representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28 = 11100 = 1.11 \times 2^4$
- $E = 127 + 4$
- The binary equivalent of 131_{10} is 10000011
- Therefore
- $$1 \;\; 10000011 \;\; 11\underbrace{000000000000000000000}_{21\ Bits}$$

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

- First let us find the binary equivalent of 0.75_{10} ...

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

- First let us find the binary equivalent of 0.75_{10} ...
- $0.75 \times 2 = 1.50$

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

- First let us find the binary equivalent of 0.75_{10} ...
- $0.75 \times 2 = 1.50$
- $0.50 \times 2 = 1.00$

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

- First let us find the binary equivalent of 0.75_{10} ...
- $0.75 \times 2 = 1.50$
- $0.50 \times 2 = 1.00$
- $0.00 \times 2 = 0.00$ $[stop]$

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN representation in single precision notation (127 bias).

- First let us find the binary equivalent of 0.75_{10} ...
- $0.75 \times 2 = 1.50$
- $0.50 \times 2 = 1.00$
- $0.00 \times 2 = 0.00$ [*stop*]
- Therefore, top–bottom .... the answer is: $(0.11)_2$

Read more: http://www.exploringbinary.com/binary-converter/

Example: 0.75_{10}

Example: 0.75_{10}

- $0.75_{10} = 0.11_2 = 1.1 \times 2^{-1}$

Example: 0.75_{10}

- $0.75_{10} = 0.11_2 = 1.1 \times 2^{-1}$
- $E = 127 - 1$

Example: 0.75_{10}

- $0.75_{10} = 0.11_2 = 1.1 \times 2^{-1}$
- $E = 127 - 1$
- The binary equivalent of 126_{10} is: 01111110

Example: 0.75_{10}

- $0.75_{10} = 0.11_2 = 1.1 \times 2^{-1}$
- $E = 127 - 1$
- The binary equivalent of 126_{10} is: 01111110
- Therefore

Example: 0.75_{10}

- $0.75_{10} = 0.11_2 = 1.1 \times 2^{-1}$
- $E = 127 - 1$
- The binary equivalent of 126_{10} is: 01111110
- Therefore
- $$0 \;\; 01111110 \;\; 1\underbrace{0000000000000000000000}_{Bits\ 22}$$

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

- $1 = 1.0_2 \times 2^0$

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

- $1 = 1.0_2 \times 2^0$
- $E = 127 - 0$

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

- $1 = 1.0_2 \times 2^0$
- $E = 127 - 0$
- The binary equivalent of 127_{10} is: 01111111

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

- $1 = 1.0_2 \times 2^0$
- $E = 127 - 0$
- The binary equivalent of 127_{10} is: 01111111
- Therefore

Example: 1_{10}

4. Given the Decimal Number: 1_{10}. Find the FPN representation in single precision notation (127 bias).

- $1 = 1.0_2 \times 2^0$
- $E = 127 - 0$
- The binary equivalent of 127_{10} is: 01111111
- Therefore
- $$0 \quad 01111111 \quad 0\underbrace{0000000000000000000000}_{Bits\ 22}$$

SPARC Workstation (32-bits) – FP operations

SPARC Workstation (32-bits) – FP operations

- FP operations are performed within a special unit called; Floating Point Unit (FPU).

SPARC Workstation (32-bits) – FP operations

- FP operations are performed within a special unit called; Floating Point Unit (FPU).
- Communication via the CPU and the FPU is done by special FP registers.

SPARC Workstation (32-bits) – FP operations

- FP operations are performed within a special unit called; Floating Point Unit (FPU).
- Communication via the CPU and the FPU is done by special FP registers.
- The CPU and FPU can work in parallel on different data.

SPARC Workstation (32-bits) – FP operations

- FP operations are performed within a special unit called; Floating Point Unit (FPU).
- Communication via the CPU and the FPU is done by special FP registers.
- The CPU and FPU can work in parallel on different data.
- (instruction-level parallelism)

SPARC Workstation (32-bits) – FP operations

- FP operations are performed within a special unit called; Floating Point Unit (FPU).
- Communication via the CPU and the FPU is done by special FP registers.
- The CPU and FPU can work in parallel on different data.
- (instruction-level parallelism)
- Today FPU and CPU can be in the same chip

The FPU has 32 registers

- Q: How can we access the registers ?
- A: Use *ld* (Load) and *st* (Store) instructions,

```
ld      [%fp + 64], %f2
set     var, %o0
st      %f0, [%o0]
```

Floating-Point in Java, C and C++

Floating-Point in Java, C and C++

- Java, C and C++ have two kinds of floating-point numbers (IEEE–754):

Floating-Point in Java, C and C++

- Java, C and C++ have two kinds of floating-point numbers (IEEE–754):
  - Float (32-bit; 4-bytes)

Floating-Point in Java, C and C++

- Java, C and C++ have two kinds of floating-point numbers (IEEE–754):
  - Float (32-bit; 4-bytes)
  - Double (64-bit; 8-bytes)