IEEE Standard for Floating-Point Arithmetic: 754

G.E. Antoniou

Floating Point Standard: IEEE 754–1985/2008

Established in 1985 (2008) as homogeneous standard for binary 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).

It floats ...

- The “decimal” point in FPN is variable or is automatically adjusted or it **floats**
- A Floating Point Number (FPN) is defined as,
- $$FPN = Fraction \times Base^{Exponent}.$$

Example: Decimal Numbers

- $12345 = 12345 \times 10^0$
- $= 1234.5 \times 10^1$
- $= 123.45 \times 10^2$
- $= 12.345 \times 10^3$
- $= 1.2345 \times 10^4$
- and ...
- $$1234.567_{10} = 0.1234567 \times 10^4$$
- General decimal Floating Point Number formula:
- $$FPN_{10} = F \times 10^{Exponent}.$$

Example: Binary Numbers

- 101_2
- $101_2 \times 2^0$
- $10.1_2 \times 2^1$
- $1.01_2 \times 2^2$
- ....
- Therefore the binary Floating Point Number formula is:
- $$FPN_2 = F \times 2^{Exponent}.$$

Signed (positive and negative) Numbers

The leftmost bit of a signed binary number determines the sign of the number:

- If the leftmost bit is 0 :
- ... [The number is: Positive]
- ....
- If the leftmost bit is 1:
- ... [The number is: Negative]
- ....
- Therefore our FPN formula becomes:
- $$FPN = (-1)^S \times F \times 2^{Exponent}$$
- where,
- S = Sign
- F = Fraction.

Problem of Uniqueness

A binary number can be represented as:

- 101_2
- $101_2 \times 2^0$
- $10.1_2 \times 2^1$
- $1.01_2 \times 2^2$
- ....
- **There is no unique representation ...**
- For a unique representation (IEEE—754 standard):
- Use only one digit to the left of the binary–point = **Normalization**.

FPN Normalization

- FPM 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 numbers
  - $0.54_{10} \times 10^{0}$
  - $54_{10} \times 10^{-2}$
  - $0.0054_{10} \times 10^{2}$
  - ... ...
  - $101_{2} \times 2^{0}$
  - $10.1_{2} \times 2^{1}$
- Normalized numbers
  - $5.4_{10} \times 10^{-1}$ (Decimal)
  - ... ...
  - $1.01_{2} \times 2^{2}$ (Binary)

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

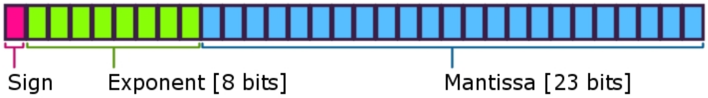
To represent the "hidden 1" and the "Fraction" use the formula:

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

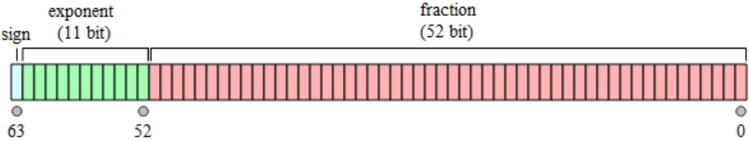
- Single (32-bit) Precision Format (Occupies one four byte word)
  - S = 1
  - E = 8
  - F = 23



Precision Formats: 64-bit

IEEE–754 Floating Point Standard

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



Biased Notation

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

Ordered Binary numbers (Exponent). Biased 127

Exponent	Real Exponent (Biased 127 Exponent)
0000 0000	-127_{10} (Reserved)
0000 0001	-126_{10} (1-127)
0000 0010	-125_{10} (2-127)
0111 1110	-1_{10} (126-127)
0111 1111	0_{10} (127-127)
1000 0000	$+1_{10}$ (128-127)
1000 0001	$+2_{10}$ (129-127)
1111 1110	$+127_{10}$ (254-127)
1111 1111	$+128_{10}$ (Reserved)

- For a 32–bit single-precision number, an exponent in the range: $-126_{10}, \ldots, +127_{10}$ is biased by adding 127 (bias) to get a value in the range $1, \ldots, 254$ (0 and 255 are reserved).

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 Unbiased Exponent so that

- Biased Exponent:

$$E = Unbiased\ Exponent + 127$$

- Solving for the Unbiased Exponent yields,
-

$$Unbiased\ Exponent = E - 127$$

- Therefore our final FPN_{32} formula takes the form:
-

$$FPN_{32} = (-1)^S \times (1 + significand) \times 2^{E-127}.$$

Final IEEE-754 standard formula for FPN_{32}

- $$FPN_{32} = (-1)^S \times (1 + significand) \times 2^{E-127}$$
- $$FPN_{32} = (-1)^S \times (1.significand) \times 2^{E-127}.$$

Example Exponents

1. Unbiased Exponent 2 is stored as $(127 + 2) = 129 = 1000\ 0001_2$ (Biased Exponent)
2. Unbiased Exponent -2 is stored as $(127 + (-2) = 125 = 0111\ 1101_2$ (Biased Exponent).

IEEE–754 Standard; FPN_{64}, (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,

- $$FPN_{64} = (-1)^S \times (1 + significand) \times 2^{E-1023}$$
- $$FPN_{64} = (-1)^S \times (1.significand) \times 2^{E-1023}$$

Largest and Smallest Values

1. Single Precision, FPN_{32}
   - 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, FPN_{64}
   - Largest normalized value:
   - $2^{1023} = \pm 2.2250738585072020 10^{-308}$
   - Smallest normalized value:
   - 2^{-1022}

Examples ...

Illustrative examples follow ...

- Decimal to FPN_{32}
- FPN_{32} to Decimal

Decimal to FPN_{32}

- Given: -5_{10}
- Find: $\overbrace{1}^{1Bit}\ \overbrace{\color{blue}{10000001}}^{8Bits}\ \overbrace{\color{red}{01}\,\underbrace{000000000000000000000}_{21Bits}}^{23Bits}$
- Why?

Example-1: -5_{10}

Given the Decimal Number: -5_{10}. Find the FPN_{32} representation in single precision notation (127 bias).

- $5_{10} = 101_2$
- $5_{10} = 101_2 = 1.01 \times 2^2$
- Unbiased Exponent $= 2$
- Biased Exponent: $E = 127 + 2 = 129_{10}$
- The unsigned binary equivalent of 129_{10} is 10000001_2
- Since, -5_{10}, is negative: $\rightarrow$ S $= 1$
- Therefore:
- $$\overbrace{1}^{1Bit}\quad \overbrace{10000001}^{8Bits}\quad \overbrace{01\,\underbrace{000000000000000000000}_{21Bits}}^{23Bits}$$

Example-2: -28_{10}

Given the Decimal Number: -28_{10}. Find the FPN_{32} representation in single precision notation (127 bias).

- -28_{10} in single precision notation (127 bias)
- $-28_{10} = 11100_2$
- $-28_{10} = 11100_2 = 1.11 \times 2^4$
- Biased Exponent: $E = 127 + 4 = 131_{10}$
- The unsigned binary equivalent of 131_{10} is 10000011_2
- S = 1
- Therefore:
-
$$1 \;\; 10000011 \;\; 11\underbrace{000000000000000000000}_{21 \; Bits}$$

Example-3: 0.75_{10}

3. Given the Decimal Number: 0.75_{10}. Find the FPN_{32} 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-4: 0.75_{10}

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

Example-5: 1_{10}

Given the Decimal Number: 1_{10}. Find the FPN_{32} representation in single precision notation (127 bias).

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

$$0 \quad 01111111 \quad 0\underbrace{0000000000000000000000}_{Bits\ 22}$$

FPN_{32} to Decimal

- Given: 1 10000010 11 $\underbrace{000000000000000000000}_{21\ Bits}$
- Find: -14_{10}
- Why?

Example-1: FPN_{32}

- Given the Floating Point Number:
- $1\ \ 10000010\ \ 11\underbrace{000000000000000000000}_{21\ Bits}$
- Find the Decimal representation.
- The sign (S) is: 1
- The biased Exponent (E) is: $10000010_2 = 130_{10}$
- Fraction: 11000000000000000000000
- FPN Formula: $FPN = (-1)^S \times (1.significand) \times 2^{E-127}$
- Result $= (-1)^1 \times (1.11) \times 2^{130-127} = (-1)^1 \times 1.11 \times 2^3$
- Result $= -(1.11) \times 2^3$
- Result $= -1110_2$
- Result$= -14_{10}$

2 minutes

- Given: $\overbrace{1}^{1 Bit}\;\overbrace{\color{blue}{10000001}}^{8 Bits}\;\overbrace{\color{red}{01}\,\underbrace{000000000000000000000}_{21 Bits}}^{23 Bits}$
- Decimal number?

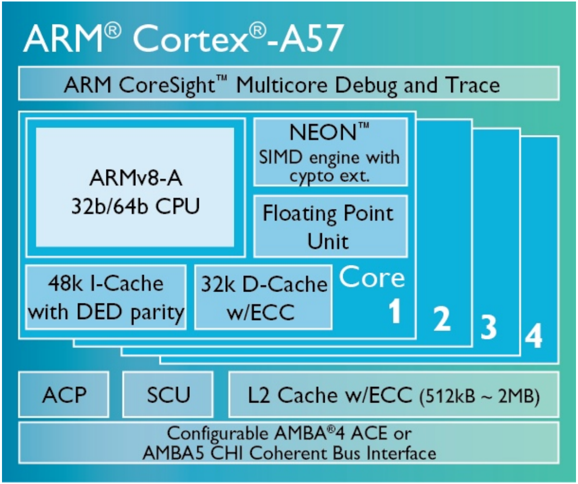
Example-2: FPN_{32}

- Given the Floating Point Number:
- $1\ 10000001\ 01\underbrace{000000000000000000000}_{21\ Bits}$
- Find the Decimal representation.
- The sign (S) is: 1
- The biased Exponent (E) is: $10000001_2 = 129_{10}$
- Fraction: 01000000000000000000000
- FPN Formula: $FPN = (-1)^S \times (1.significand) \times 2^{E-127}$
- Result $= (-1)^1 \times (1.01) \times 2^{129-127} = (-1)^1 \times 1.01 \times 2^2$
- Result $= -(1.01) \times 2^2$
- Result $= -101_2$
- Result $= -5_{10}$

ARM Cortex RISC CPU – A57 – 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

ARM Cortex RISC CPU – A57

- 

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)

- (end)