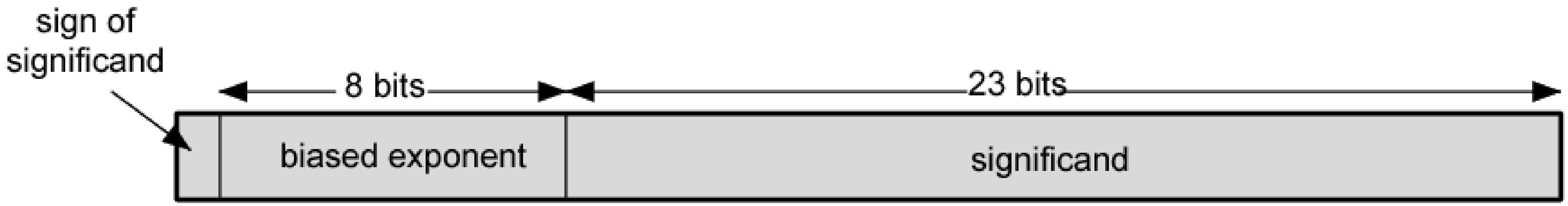
Computer Organization (IS F242)

Lect 12: Floating point

Floating Point – IEEE 754 Standard



(a) Format

- Use equivalent of “scientific notation”

  $$+/-\ .\text{significand} \times 2^{\text{exponent}}$$

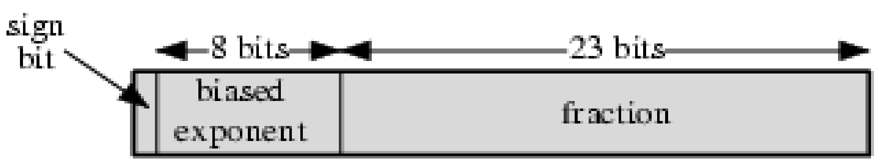
- Need to represent F (*fraction*), E (*exponent*), and sign.
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

Floating-Point Representation

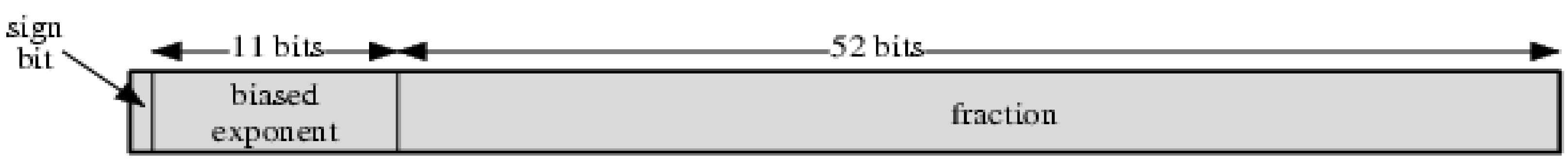
- IEEE 754 floating point standard
  - Single precision
    - 1 bit sign, 8 bit exponent and 23 bit fraction
  - Double precision
    - 1 bit sign, 11 bit exponent and 52 bit fraction

IEEE 754 Floating Point Representation

- Single precision 4 bytes
- Double Precision 8 bytes
- Extended Double 10 bytes
- Quadruple Precision 16 bytes



(a) Single format



(b) Double format

IEEE 754 Floating-Point (32-bits)

Single Precision

$$N = (-1)^S \times 1.\text{fraction} \times 2^{\text{exponent} - 127},\ 1 \le \text{exponent} \le 254$$

$$N = (-1)^S \times 0.\text{fraction} \times 2^{-126},\quad \text{exponent} = 0$$

Double Precision

$$N = (-1)^S \times 1.\text{fraction} \times 2^{\text{exponent} - 1023},\ 1 \le \text{exponent} \le 2046$$

$$N = (-1)^S \times 0.\text{fraction} \times 2^{-1023},\quad \text{exponent} = 0$$

Bias value for SP is 127 and for DP it is 1023

Exponent for Floating point Number

- Exponent is in excess or biased notation
  - 8 bits (in single precision) to represent exponent
  - -128 to +127 OR 0 to 255 (256 values) can be represented
  - Not really interested in representing negative number (avoid complications)
  - How will we manage?
    - Add a bias value so that all Negative values will become positive.
    - Bias in Single precision is +127. Why?
    - Bias in Double precision is +1023.

Normalization

- Floating point Numbers are usually normalized
  - Exponent is adjusted so that leading bit (MSB) of the mantissa is 1
  - Since MSB is always 1, No need to store it

Standard 32 bit Floating point representation

$$N = (-1)^S \times 1.\text{fraction} \times 2^{\text{exponent}-127},\ 1 \le \text{exponent} \le 254$$

$$N = (-1)^S \times 0.\text{fraction} \times 2^{-126},\ \text{exponent} = 0$$

Floating Point Example

- Single-precision IEEE floating point number

  1 01111110 10000000000000000000000

  sign *exponent* *fraction*

  - Sign is 1 →number is negative.
  - Exponent field is 01111110 = 126 (decimal).
  - Fraction is 0.100000000000… = 0.5 (decimal).

Value = $-1.1 \times 2^{(126-127)} = -1.1 \times 2^{-1} = -0.11$

Decimal Equivalent: -0.75.

Floating Point Example

Represent 1/8 (0.125) in IEEE 754 format?

Binary equivalent of 0.125 is 0.001 or 1.0 x 2^{-3} (Normalized)

N = $(-1)^s$ X 1.fraction X $2^{exponent-127}$

Sign bit = 0 (Number is positive)

exponent - 127 = -3 i.e. exponent = 124

Binary equivalent of 124 = 01111100

Fraction = 00000000000000000000000

Final representation of 1/8 in IEEE 754 format is

00111110000000000000000000000000

Floating Point Example

Represent 2^{-131} in IEEE 754 format?

Binary equivalent of 0.00001x 2^{-126}

If exponent is 0 then

$N = (-1)^s \times 0.\text{fraction} \times 2^{-126}$

Sign bit = 0 (Number is positive)

exponent = 0

Fraction = 00001000000000000000000

Final representation of 2^{-131} in IEEE 754 format is

00000000000001000000000000000000

Denormalized Numbers

- Used to handle exponent underflow i.e. exponent is too small to represent
  - How to fit exponent in representable range???
  - Shift fraction to the right and increase exponent accordingly
- Is it really beneficial? If Yes. How?
- Representation
  - Exponent of zero with non zero fraction
  - Bit to the left to the binary point is zero
  - True exponent is -126

Floating point Representation

- What is the largest positive number we can represent by using a floating point representation?
  - 0 11111110 11111111111111111111111 ~ 2^{128}
- What is the smallest positive number we can represent by using a floating point representation?
  - **0 00000000 00000000000000000000001** 2^{-149}

Exercises

- 0.0101×2^{67}
- 01110.1010×2^{-7}
- -127.625
- 0.0011×2^{-137}
- 0

Exceptional cases

- exponent is 0, fraction is non-zero
  - + or – denormalized number
- exponent is 0, fraction is zero
  - ZERO
- exponent is 255(2047), fraction is zero
  - + or – infinity

Exceptional cases

- exponent is 255(2047), fraction is non-zero
  - NaN (Not a Number)
  - Sign bit is 0/1
  - Biased exponent is 255
  - Mantissa is non zero
  - x 11111111 axxxxxxxxxxxxxxxxxxxxxx

Exceptional cases

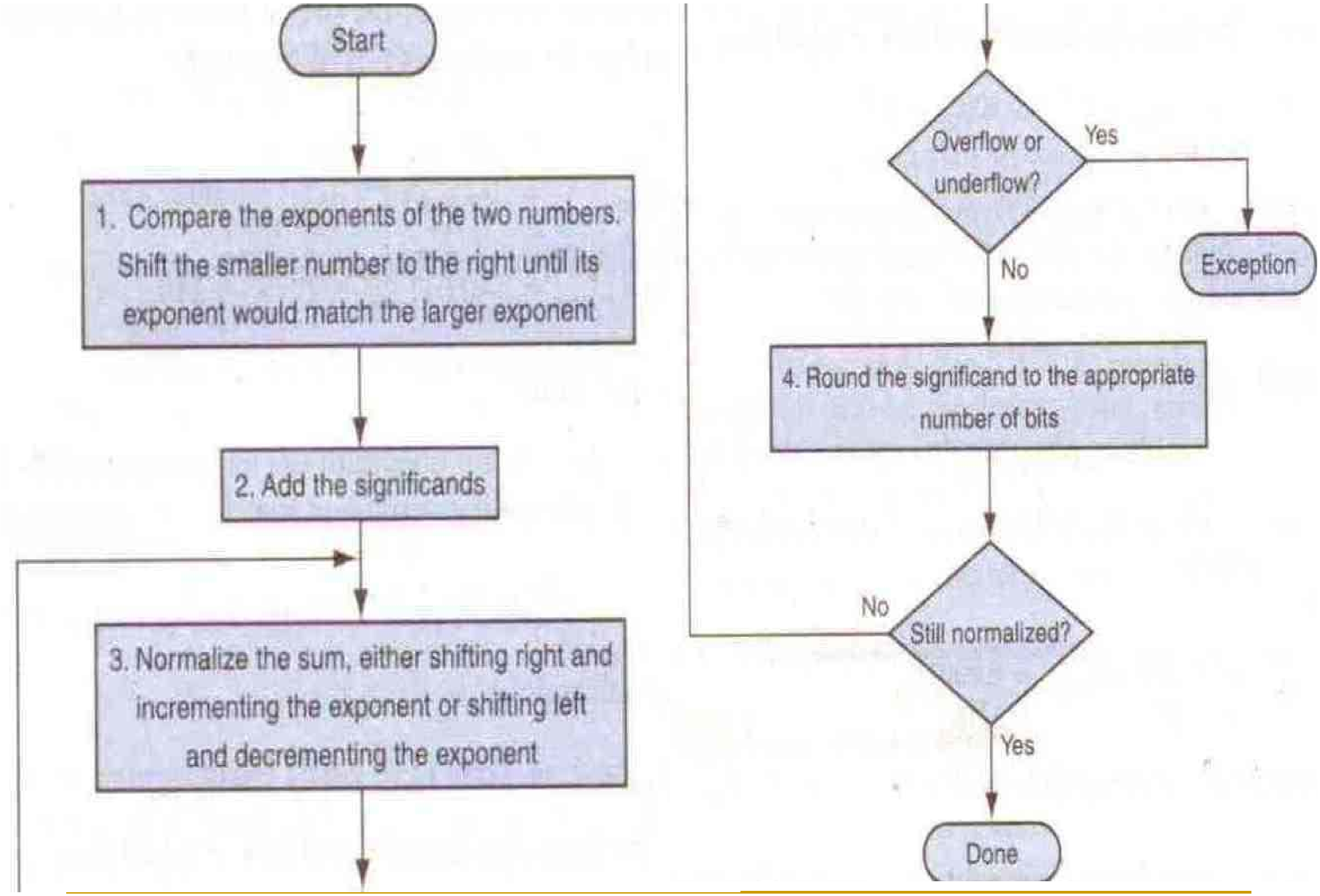
- Signaling NaN (sNaN or NaNS)
  - If a = 0 then Signaling NaN (sNaN)
    - Example: Divided by Zero, Square root of Negative Number, logarithm of a negative number, tangent of an odd multiple of 90 degrees (or π/2 radians), inverse sine or cosine of a number which is less than -1 or greater than +1
  - Signaling NaN signals an invalid operation exception
- Quiet NaN (qNaN or NanQ)
  - If a = 1 then quiet NaN (qNaN)
    - Example: Any operation on signaling NaN, 0/0, ∞/∞, $\infty/-\infty$, $-\infty/\infty$, $-\infty/-\infty$, $0\times\infty$, $0\times-\infty$, The power 1^{∞}, $\infty + (-\infty)$, $(-\infty) + \infty$ and equivalent subtractions
  - qNaN propagates through without signaling an exception

Overflow and Underflow

- Overflow
  - A positive exponent becomes too large to fit in the exponent field
- Underflow
  - A negative exponent becomes too large to fit in the exponent field

Arithmetic Operations

- Addition & Subtraction
  - Check for Zeros
  - Align the Mantissas
  - Add or Subtract the Mantissas
  - Normalize the result
  - Example
    - $X = 0.3 * 10^2 \quad Y = 0.2 * 10^3$
    - $X = (0.1 * 2^0)_2 \; Y = (-0.0111 * 2^0)_2$
    - $12.5 \times 10^1 + 346 \times 10^{-3}$


Start
1. Compare the exponents of the two numbers. Shift the smaller number to the right until its exponent would match the larger exponent
2. Add the significands
3. Normalize the sum, either shifting right and incrementing the exponent or shifting left and decrementing the exponent
Overflow or underflow?
Yes
Exception
No
4. Round the significand to the appropriate number of bits
Still normalized?
No
Yes
Done

FP Adder Hardware