Computer Systems
Lecture 18

Overview

- Floating Point Numbers
- Floating Point Formats
- Excess-*n* notation
- Normalisation of floating point numbers
- Floating point in binary
- IEEE standard 754

Floating Point Numbers

- It is not always possible to express numbers in integer form.
- Real, or **floating point** numbers are used in a computer when:
  - The number to be expressed is outside of the integer range of the computer, like 5.375×10^{25}
  - Or, when the number contains a fraction, like 345.0256

Exponential notation (base 10)

- In general, this notation represents numbers in a form $\boldsymbol{a}\times\mathbf{10^b}$.
- Example: $12345 =$

$$12345 \times 10^0 =$$

$$0.12345 \times 10^5 =$$

$$1234500 \times 10^{-2}$$

Components of exponential notation

- The **sign** of the number.
- The **magnitude** of the number, known as the **mantissa**.
- The sign of the **exponent**.
- The magnitude of the exponent.
-
- Two additional pieces of information:
  - The **base** of the exponent (e.g., 10 or 2).
  - The location of the **decimal point**.

Floating point formats

- Any format for floating point numbers should specify how the components of an exponential notation are stored (in a word, or several words).

- The **base of the exponent** and the **location of the binary point** are standardised as part of the format and, therefore, **do not have to be stored** at all.

Floating Point Formats (cont.)

- **Example:** Suppose, that the standard code consists of space for seven digits and a sign:

SEEMMMMM

  - So, we have two digits for the **exponent** and 5 digits for the **mantissa**.
- Trade-off: precision (mantissa) vs. range (exponent).
- Most commonly, the mantissa is stored using **sign-magnitude** format.
- What about the sign of the exponent?

Excess-*n* notation for the exponent

- Excess-50 notation for the 2-digit decimal representation of the exponent:

Representation	$0 \ldots \quad 49$	$50 \ldots 99$
Exponent being represented	$-50 \ldots -1$	$0 \ldots \quad 49$

- **Offset** the value of the exponent by a chosen amount (here it is 50).
- It is simpler to use for exponents than the complementary form.

Floating point formats

- Thus, 5-digit excess-50 notation allows us a magnitude range of

$$0.00001 \times 10^{-50} < \ldots < 0.99999 \times 10^{+49}$$

  – We assume that the decimal point is located at the beginning of five-digit mantissa.

Normalisation of floating point numbers

- To maximise the precision for a given number of digits, numbers are, usually, stored **with no leading zeros**.
- Process of transformation the numbers into such a form is called **normalisation**.
- Example.
  - A number: 0.00123×10^7
  - Its normalised form: 1.23×10^4

Floating point in the computer: binary representation

- Typically, 4, 8 or 16 bytes are used to represent a floating point number.
- Typical floating point format: 32 bits are used to provide a range of approximately 10^{-38} to 10^{+38}.
  - 1 bit is used for sign of mantissa.
  - 8 bits are used to store the exponent in excess-128 notation.
  - 23 bits are used for mantissa.

Floating point in binary

- Assuming **normalised** representation one can omit the storage of most significant bit (it is always 1 !).
  - So, 23 bits provide 24 bits of precision.
- Binary point should be specified.
  - Most common choice is **after the most significant bit, i.e. 1.???**.
  - Notice, this bit itself is not stored!

Example (excess-128)

- Consider the code:

  0 10000001 11001100000000000000000

- Sign of mantissa is '+' (leftmost bit is 0)
- Mantissa is 1.11001100000000000000000

  Assumed

- Exponent is 00000001 (=10000001-$\underbrace{10000000}_{128}$)
- The number represented is +11.10011000…000

Another example (excess-128)

- Consider the code:

  1 10000100 10001111000000000000000

- Sign of mantissa is '-' (leftmost bit is 1)
- Mantissa is 1.10001111000000000000000

  Assumed

- Exponent is 00000100 (=10000100-$\underbrace{10000000}_{128}$)
- The number is -11000.1111000…000

One more example (excess-128)

- Consider the code:

  1 01111110 10101010101010101010101

- Sign of mantissa is ‘-’ (leftmost bit is 1)
- Mantissa is 1.10101010101010101010101

  Assumed

- Exponent is -00000010 (=01111110-$\underbrace{10000000}_{128}$)
- The number represented is -0.0110101000…000

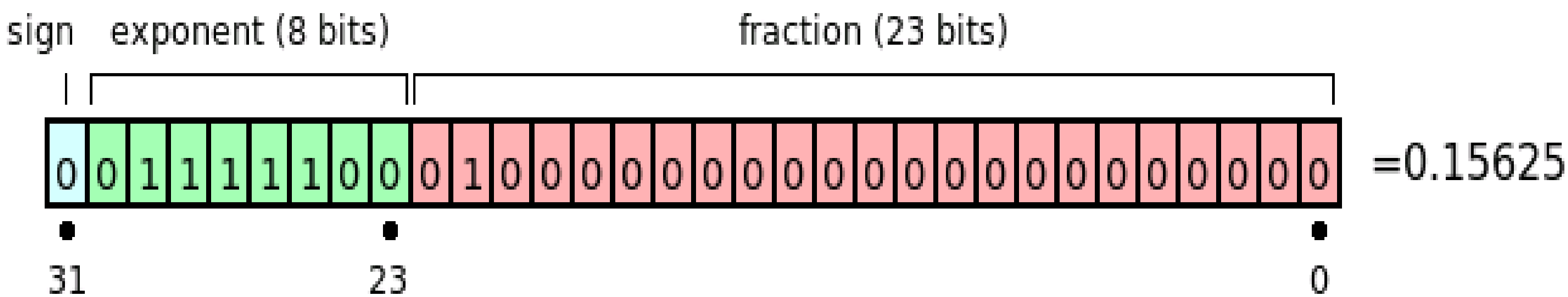
IEEE standard 754

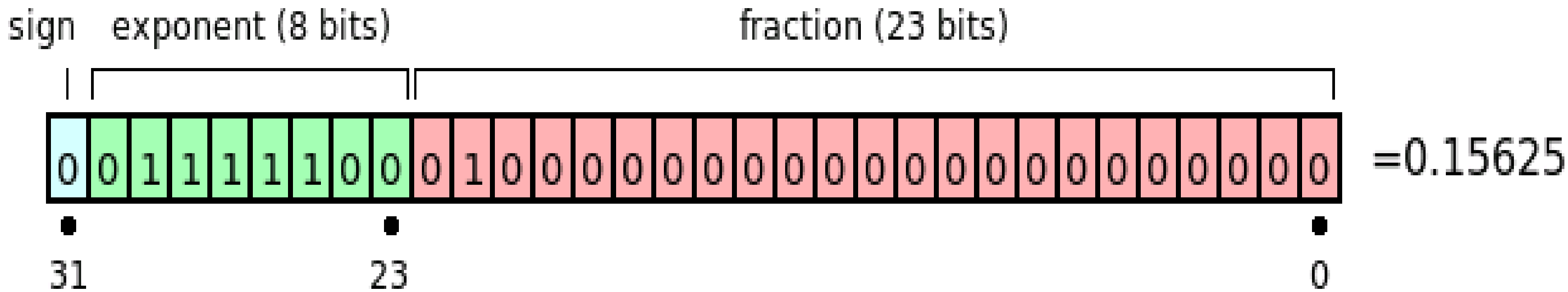
- **Single-precision floating point format**:
  - Almost the same format as we have just described, with some exceptions.
  - 32-bits: 1 bit for sign, 8 bits of exponent, 23 of mantissa.
  - The exponent is formatted using **excess-127** notation.
  - Overall, the standard allows approximately 7 decimal digit precision and approximate value range 10^{-45} to 10^{38}.

Exponent biasing

- The exponent is biased by 2^{8-1}-1, that is, biased by 127.
- Exponents in the range -127 to +127 are representable.
- e=128 reserved for NaN, infinity

Single-precision 32 bit IEEE 754





- The represented number has value v:

$$v = s \times 2^{e} \times m, \text{ where}$$

  - $s = +1$ (positive number) when the sign bit is 0;
  - $s = -1$ (negative number) when the sign bit is 1;
  - e = exponent – 127;
  - m = 1.fraction in binary. (The leading '1' is not stored.) Therefore, $1 \leq m < 2$.
- In the above example, where $s = 1$, $e = -3$, $m = 1.01$ (in binary, which is 1.25 in decimal).
- The represented number is therefore $+1.01 \times 2^{-3}$ (in binary), which is +0.15625.

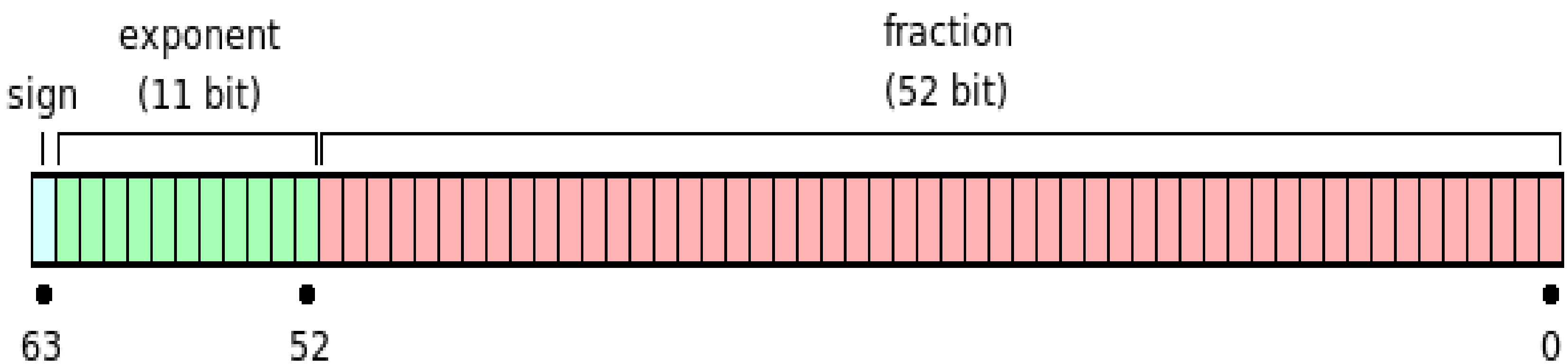
Special cases (e = 128)

- If exponent is 0 and fraction is 0, the number is ± 0 (depending on the sign bit).
- If exponent = $2^8 - 1$ and fraction is 0, the number is $\pm$infinity (again depending on the sign bit).
- If exponent = $2^8 - 1$ and fraction is not 0, the number being represented is not a number (NaN).

IEEE standard 754 (cont.)

- **Double-precision floating point format**:
  - 64-bits: 1 bit for sign, 11 bits of exponent, 52 of mantissa.
  - The exponent is formatted using **excess-1023** notation.
  - Overall, the standard allows approximately 15 decimal digit precision and approximate value range 10^{-324} to 10^{308}.

Double-precision 64 bit IEEE 754



- Q. Under the IEEE 754 standard..
  - how many bits are required to specify the sign of the magnitude?
  - how many bits are required to specify the sign of the exponent?

- Q. Under the IEEE 754 standard, how many bits are required to specify the decimal point position?

- Q. Does IEEE standard 754 provide a specification for NaN?

- Q. Under the IEEE 754 standard for single-precision floating point format, what type of excess notation is used for exponent specification?

- Q. Under the IEEE 754 standard for double-precision floating point format, what type of excess notation is used for exponent specification?

Readings

- [Wil06] Section 5.6.
- Wikipedia article on floating point systems:
  - http://en.wikipedia.org/wiki/Floating_point
- Wikipedia article on IEEE 754 standard:
  - http://en.wikipedia.org/wiki/IEEE_754