MA5233 Computational Mathematics

Lecture 1: Machine Numbers

Simon Etter

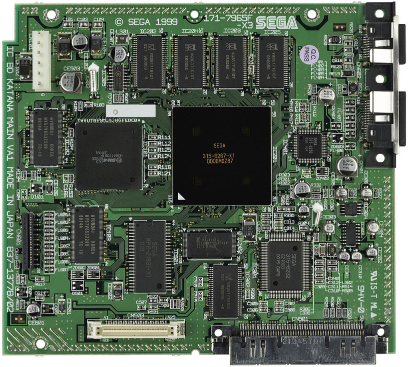
National University of Singapore

2019/2020

Machine Numbers

What is a computer?

- Memory: vector $\mathtt{mem} \in \{0, 1\}^{\mathtt{M}}$
- Processor: `read`, `compute`, `write`



---

https://commons.wikimedia.org/wiki/File:Sega-Dreamcast-Motherboard-Top.jpg

Machine Numbers

Binary numbers

$$d_{n-1} \dots d_0 = \sum_{k=0}^{n-1} d_k \, 2^k \qquad \text{with} \qquad d_k \in \{0, 1\}.$$

Examples

$$\begin{array}{rcrcr} \text{bin.} & & \text{dec.} & & \text{conversion} \\ 10 & \hat{=} & 2 & = & 1 \times 2^1 + 0 \times 2^0 \\ 1011 & \hat{=} & 11 & = & 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \end{array}$$

Machine Numbers

Fixed-size variables

Problem

- We distinguish "10, 11" from "1011" using spaces.
- `mem` allows only 0 & 1, no spaces.

Solution: fix number of digits in numbers.

$$10 \to 0010 \qquad 101 \to 0101$$

Example

- Assume number of digits per number is fixed to four.
- Then "00100101" unambiguously represents the two numbers "0010" and "0101".

Machine Numbers

Consequences of fixed-size variables

- There is a largest representable integer.
- Wrap-around behaviour: $11 + 01 = 00$, $00 - 01 = 11$.

Example

- Julia `UInt64` type represents unsigned 64-bit integer.
- Largest representable number: $2^{64} - 1 \approx 1.8 \times 10^{19}$

Machine Numbers

Negative integers (two's complement)
Flip sign of leading digit:

$$d_{n-1} \dots d_0 = -d_{n-1} \times 2^{n-1} + \sum_{k=0}^{n-2} d_k \, 2^k$$

Rationale: signed and unsigned `+`,`-`,`*` map bits identically.

Examples

$$\begin{array}{rcrcl} \text{s. bin.} & & \text{dec.} & & \text{conversion} \\ 0010 & \hat{=} & 2 & = & -0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ 1010 & \hat{=} & -6 & = & -1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \end{array}$$

Note: wrap-around behaviour remains!

Example

- Julia `Int64` type represents signed 64-bit integer.
- Range of representable numbers: -2^{63} to $2^{63} - 1$.

Machine Numbers

IEEE floating-point numbers

$$s \times f \times 2^e \quad \text{where} \quad \begin{cases} s \in \{-1, 1\} & \text{sign} \\ f := 1.f_0 f_1 \ldots f_p & \text{significand/mantissa/fraction} \\ e \in \{e_{min}, \ldots, e_{max}\} & \text{exponent} \end{cases}$$

Example: -1.01×2^2 represents $-1.25 \times 4 = -5$

Name	Julia	p	2^{-p}	e_{min}	e_{max}	$2^{e_{max}}$
single	`Float32`	23	1.2×10^{-7}	-126	127	1.7×10^{38}
double	`Float64`	52	2.2×10^{-16}	-1022	1023	9.0×10^{307}

Special values

- $f = 0$, $e = e_{min} - 1$: ± 0
- $f = 0$, $e = e_{max} + 1$: $\pm$`Inf`
- $f \neq 0$, $e = e_{max} + 1$: `NaN`

Machine Numbers

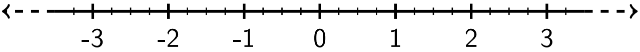
Remarks on floating-point numbers

- Signed zero ± 0 is useful for branch cuts:
  - `sqrt(-1.0 + 0.0im) -> +1.0im`
  - `sqrt(-1.0 - 0.0im) -> -1.0im`
- $\pm$`Inf` represents well-defined limit:
  - `1.0 / 0.0 -> Inf`
  - `1.0 / -0.0 -> -Inf`
  - `1 + Inf -> Inf`
  - `2 * Inf -> Inf`
  - `Inf + Inf -> Inf`
  - `Inf * Inf -> Inf`
  - `Inf == Inf -> true`
- `NaN` represents ill-defined limit:
  - `0.0/0.0 -> NaN`
  - `Inf - Inf -> NaN`
  - `0*Inf -> NaN`
  - `NaN == NaN -> false` (use `isnan()`)

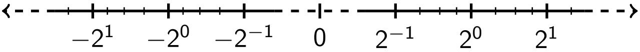
Machine Numbers

Remarks on floating-point numbers

- Accuracy is relative!
  - `x = 1.0; nextfloat(x) - x -> 2.2e-16`
  - `x = 1e16; nextfloat(x) - x -> 2`
- There is an `eps(T) > 0` such that `1 + x == 1` for all `x < eps(T)`.
- Floating-point numbers don't represent this



but this



- Floats represent numbers which ints don't, and vice versa.
  - `x = 2.0^64; Int(x) -> InexactError()`
  - `x = 2^62-2^8; Int(Float64(x)) - x -> 256`

```
            bitstring(x) -> 0011111111111111111111111111111111111111111111111111111100000000
bitstring(Int(Float64(x))) -> 0100000000000000000000000000000000000000000000000000000000000000
```

Machine Numbers

Remarks on floating-point numbers

- `+,-,*,/,sqrt` are all approximate.
- Many mathematical identities are violated:
  - `(a+b)+c != a+(b+c)`
  - `(a+b)*c != a*c + b*c`
  - `b/a != 1/a*b`

Fixed-point numbers (FixedPointNumbers.jl)

$$\mathtt{a} \times 2^{-p} \text{ where } \mathtt{a::Int} \text{ and } p \text{ some fixed number.}$$

- Accuracy is absolute.
- `+,-` are exact (up to overflow).
- Hardly used in practice, but there are applications:
  - bank accounts,
  - time keeping.

Machine Numbers

Maiden launch of Ariane 5 rocket

- Ariane 5: more powerful successor to Ariane 4.
- Largely same software as Ariane 4.
- Horizontal velocity `vx` was stored as `Float64`, but occasionally converted to `Int16`.
- Flight trajectory of Ariane 5 led to `vx` exceeding the range of `Int16`.
- This resulted in software failure and loss of vehicle.
- `https://youtu.be/gp_D8r-2hwk`

Lessons to be learnt

- Think carefully about machine numbers!
- Always test your code!

Machine Numbers

References and further reading

- IEEE standard
  https://doi.org/10.1109/IEEESTD.2008.4610935
- Doubles
  https:
  //en.wikipedia.org/wiki/Double-precision_floating-point_format
- Two's complement
  https://en.wikipedia.org/wiki/Two%27s_complement
- Ariane 5 explosion
  http://www-users.math.umn.edu/~arnold/disasters/ariane.html