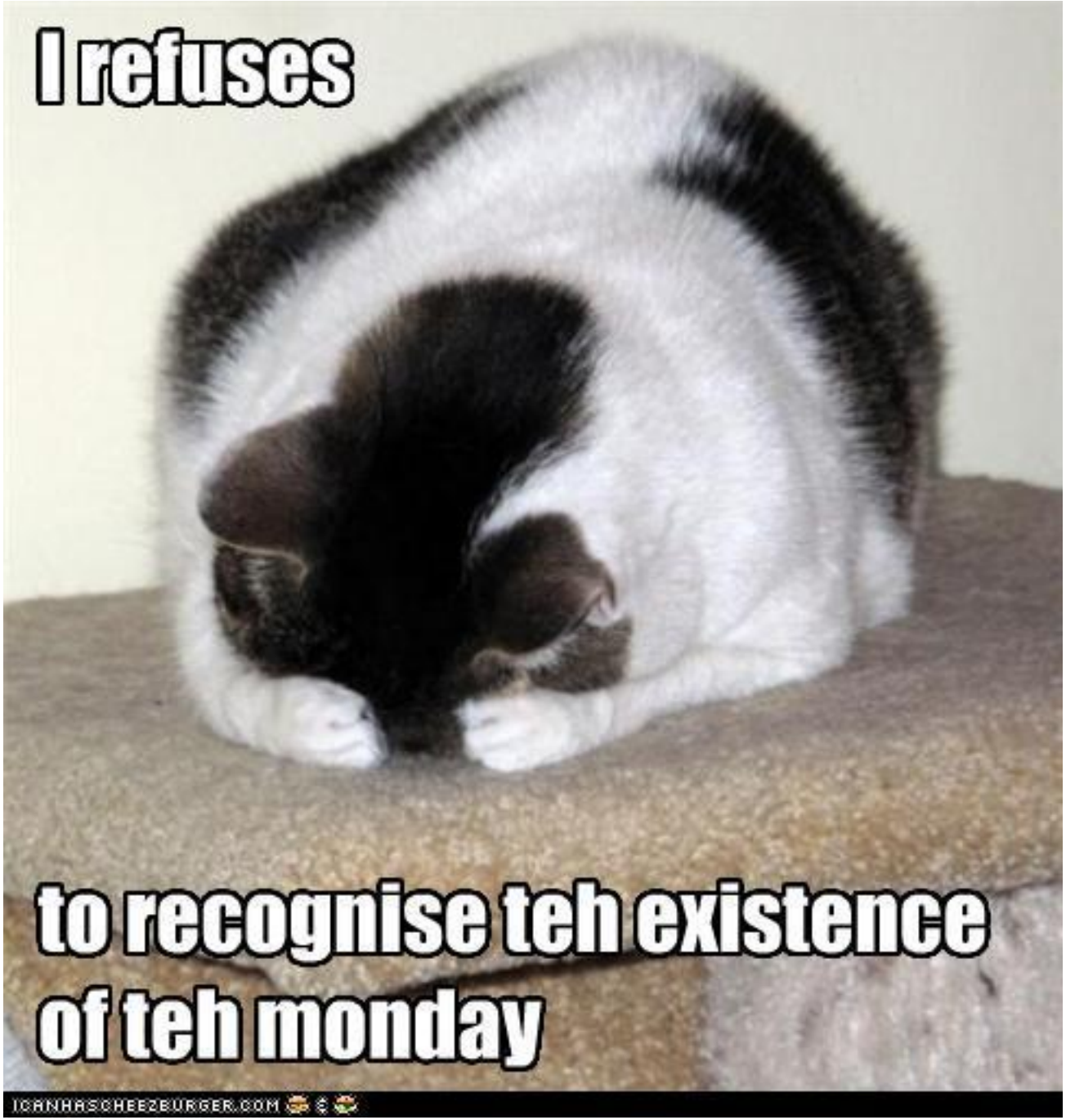

I refuses
to recognise teh existence
of teh monday
ICANHASCHEEZBURGER.COM

- Floating point standard: machine not compiler
  - GPUs

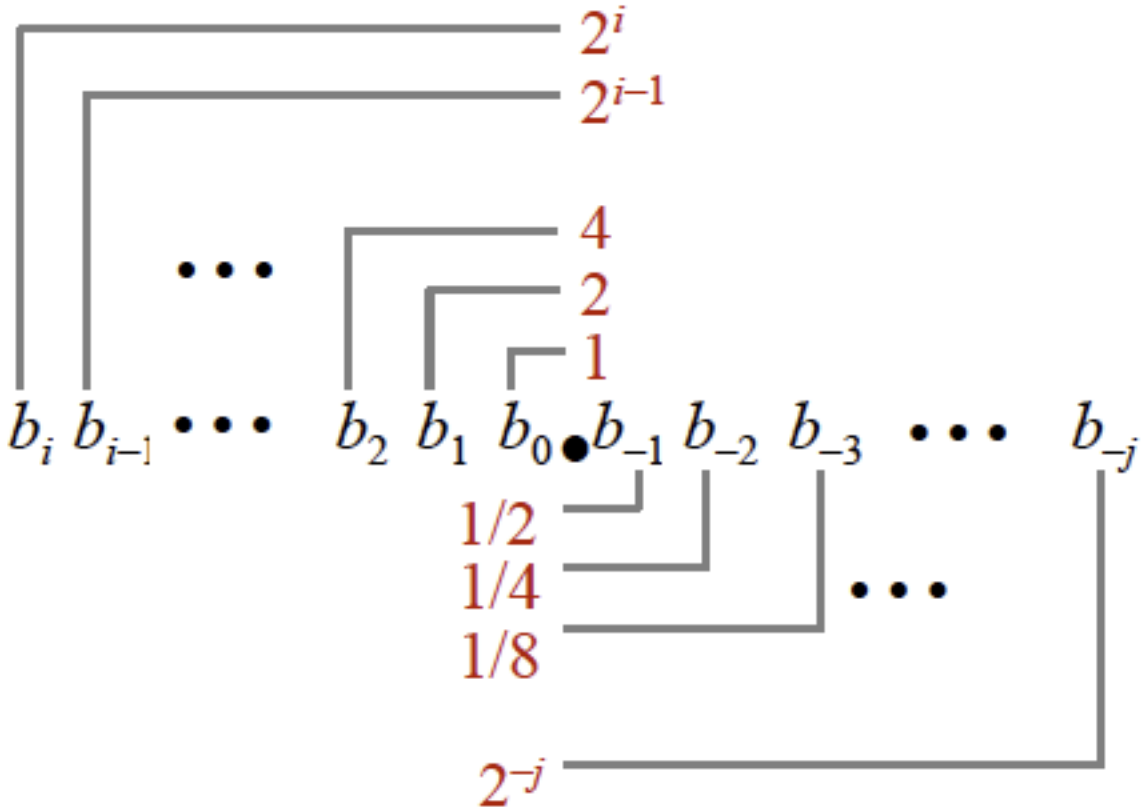
- Practice problem:
  - 2.34

(Shift practice)

Decimal Numbers

- 1234.5678

Fractional Binary Numbers



- **Representation**
  - Bits to right of "binary point" represent fractional powers of 2
  - Represents rational number: $\sum_{k=-j}^{i} b_k \cdot 2^k$

Fractional Binary Numbers: Examples

Value	Representation
5-3/4	101.11_2
2-7/8	10.111_2
63/64	0.111111_2

- **Observations**
  - Divide by 2 by shifting right
  - Multiply by 2 by shifting left
  - Numbers of form $0.111111\ldots_2$ are just below 1.0
    - $1/2 + 1/4 + 1/8 + \ldots + 1/2^i + \ldots \rightarrow 1.0$
    - Use notation $1.0 - \varepsilon$

- Limitations?

Representable Numbers

- **Limitation**
  - Can only exactly represent numbers of the form $x/2^k$
  - Other rational numbers have repeating bit representations

- **Value** **Representation**

Value	Representation
1/3	$0.0101010101[01]\ldots_2$
1/5	$0.001100110011[0011]\ldots_2$
1/10	$0.0001100110011[0011]\ldots_2$

Example: Patriot Missile in 1st Gulf War

Heard of them?

- 6-bit IEEE-like format

IEEE Floating Point

- **IEEE Standard 754**
  - Established in 1985 as uniform standard for floating point arithmetic
    - Before that, many idiosyncratic formats
  - Supported by all major CPUs
- **Driven by numerical concerns**
  - Nice standards for rounding, overflow, underflow
  - Hard to make fast in hardware
    - Numerical analysts predominated over hardware designers in defining standard

Floating Point Representation

- **Numerical Form:**

$$(-1)^s\, M\ \ 2^E$$

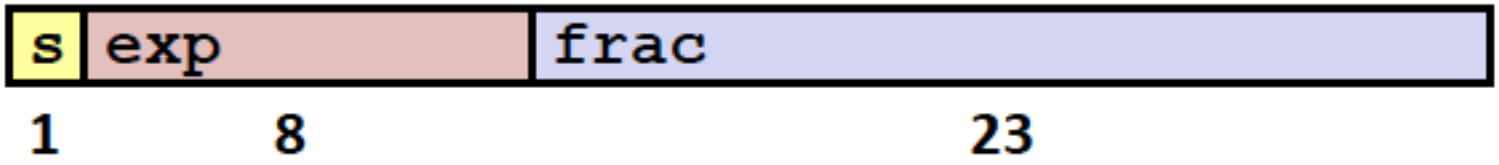
  - **Sign bit** ***s*** determines whether number is negative or positive
  - **Significand** ***M*** normally a fractional value in range [1.0,2.0).
  - **Exponent** ***E*** weights value by power of two

- **Encoding**
  - MSB **s** is sign bit ***s***
  - **`exp`** field encodes ***E*** (but is not equal to E)
  - **`frac`** field encodes ***M*** (but is not equal to M)

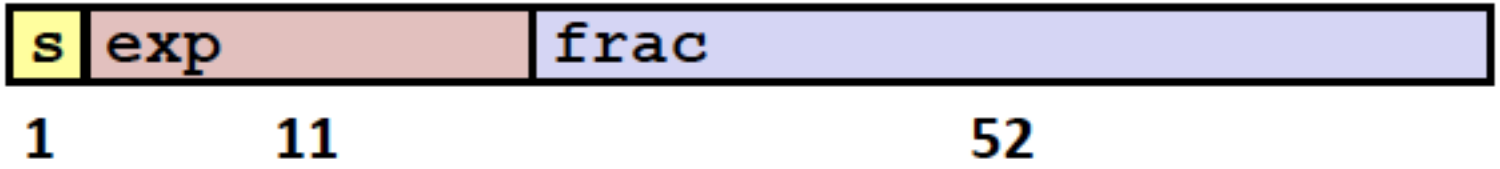
s	exp	frac

Precisions

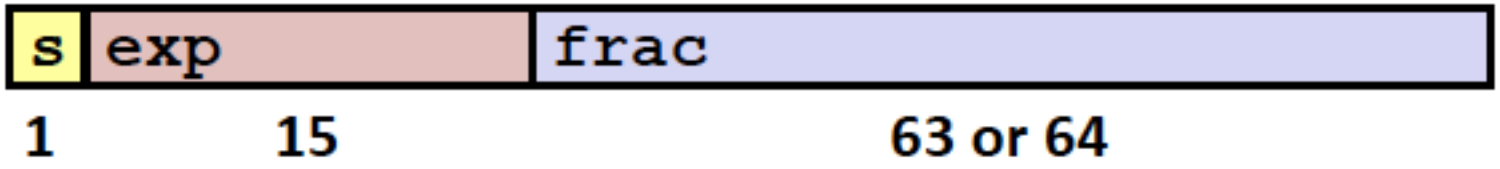
- Single precision: 32 bits



- Double precision: 64 bits



- Extended precision: 80 bits (Intel only)



Condition 1

Normalized Values

- **Condition: `exp` ≠ `000…0` and `exp` ≠ `111…1`**

- **Exponent coded as *biased* value: *E* = *Exp* – *Bias***
  - *Exp*: unsigned value **`exp`**
  - *Bias* = 2^{e-1} - 1, where e is number of exponent bits
    - Single precision: 127 (*Exp*: 1…254, *E*: -126…127)
    - Double precision: 1023 (*Exp*: 1…2046, *E*: -1022…1023)

- **Significand coded with implied leading 1: *M* = `1.xxx…x`$_2$**
  - **`xxx…x`**: bits of **`frac`**
  - Minimum when **`000…0`** (*M* = 1.0)
  - Maximum when **`111…1`** (*M* = 2.0 – ε)
  - Get extra leading bit for "free"

Normalized Encoding Example

- **Value:** `Float F = 15213.0;`
  - $15213_{10} = 11101101101101_2$
    = [redacted]

- **Significand**
  M = [redacted]
  `frac` = [redacted]

- **Exponent**
  E = [redacted]
  Bias =
  Exp =

- **Result:**

0	[redacted]	[redacted]
s	exp	frac

Tiny Floating Point Example

s	exp	frac
1	4	3

- **8-bit Floating Point Representation**
  - the sign bit is in the most significant bit.
  - the next four bits are the exponent, with a bias of 7.
  - the last three bits are the **`frac`**

- **Same general form as IEEE Format**
  - normalized, denormalized
  - representation of 0, NaN, infinity