Lesson 7) Intro & Deep study of binary8 + transistor into Lesson 7

Recall: we will "study" five separate floating point formats in this class (and use four of these in MATLAB):

Formal Names	Nickname
binary8	quarter-precison
binary16	half-precision
binary32	single precision
binary64	double precison
binary80	extendend precison

Generalized floating-point Anatomy for all 5 data types

Total # of bits	Formal Name	Nickname	# bits for sign	# bits for exponent	# bits of significand	biased excess-K value
8	binary8	quarter-precision	1	3	4	$K = 2^{3-1} - 1$ $= 2^2 - 1$ $= 4 - 1 = 3$ (excess-3)
16	binary16	half-precision	1	5	10	$K = 2^{5-1} - 1$ $= 2^4 - 1$ $= 16 - 1 = 15$ (excess-15)
32	binary32	single-precision	1	8	23	$K = 2^{8-1} - 1$ $= 2^7 - 1$ $= 128 - 1 = 127$ (excess-127)
64	binary64	double-precision	1	11	52	$K = 2^{11-1} - 1$ $= 2^{10} - 1$ $= 1024 - 1 = 1023$ (excess-1023)
80*	binary80	Intel-based extended-precision	1	15	64	$K = 2^{15-1} - 1$ $= 2^{14} - 1$ $= 16,383$

* Special Note about binary80

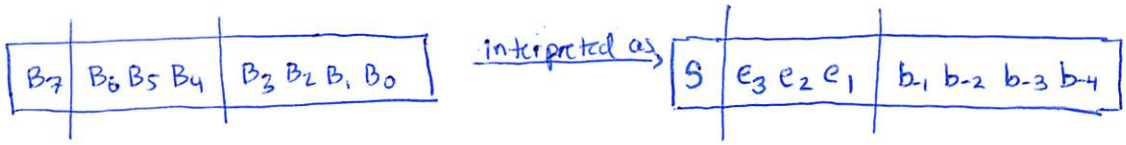
Microprocessors from the well-known hardware company Intel implement Floating-Point arithmetic using 80-bit registers with one bit for the sign, 15 bits for the exponent field and 64 bits for the significand. In Intels extended binary80 format, the leading bit for normalized & subnormal numbers is explicitly stored in memory (1)

we begin w/ a deep dive into binary8:

$$X = \text{binary8}(B)$$

$$\Rightarrow X = \text{binary8}(B_7\, B_6\, B_5\, B_4\, B_3\, B_2\, B_1\, B_0)$$

Visual Representation



make map explicit:

▫ sign field:

$$\underbrace{\boxed{S}}_{\text{value}} = \underbrace{\boxed{B_7}}_{\text{raw bit(s)}} = \begin{cases} 0 & \text{if } X > 0 \\ 1 & \text{if } X < 0 \end{cases}$$

▫ exponent field

$$\underbrace{\boxed{e}}_{\text{value}} = u - K = \text{uint3}(e_2\, e_1\, e_0) - K$$

$$= \text{uint}(B_6\, B_5\, B_4) - K$$

$$= \text{uint}(B_6\, B_5\, B_4) - 3$$

(Note: with some very special assumptions on the range of values $001 \le B_6\, B_5\, B_4 \le 110$ and boundary value patterns 000 and 111 reserved for special encodings)

☐ significand field

If $001 \le B_6 B_5 B_4 \le 110$, then the significand field

$001 \le B_6 B_5 B_4 \le 110$: range of exponent codes that produce normalized floating-point numbers

$B_3 B_2 B_1 B_0$ interpreted as $\boxed{1}.\ b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}$ (hidden bit: implied but not explicitly stored)

where $b_{-1} = B_3$, $b_{-2} = B_2$, $b_{-3} = B_1$, $b_{-4} = B_0$

$\Rightarrow$ If $001 \le B_6 B_5 B_4 \le 110$ and $x = \text{binary8}(B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0)$, then

$$x = \pm (1.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}) \times 2^{e}$$

$$\Rightarrow x = (-1)^{s} \cdot (1.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}) \times 2^{e}$$

hint: think scientific notation

$$\Rightarrow x = (-1)^{s} \cdot (1 + 0.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}) \times 2^{u-K}$$

$K = 2^{3-1} - 1$

$u = \text{uint3}(e_2 e_1 e_0) - \boxed{3}$

$$\Rightarrow x = (-1)^{s} \cdot (1 + 0.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}) \times 2^{u}$$

$$\Rightarrow \boxed{x = (-1)^{s} \cdot (1 + f) \times 2^{e}}$$ ← math object (takes thought to understand)

$$\Rightarrow x = (-1)^{B_7} \cdot \left(1 + \frac{B_3 B_2 B_1 B_0}{2^4}\right) \times 2^{\text{uint3}(B_6 B_5 B_4) - K}$$

← math object that is closer to comp science world

$$\Rightarrow \quad 1 + f = 1 + \frac{B_3\, B_2\, B_1\, B_0}{2^4}$$

f: Explicitly stored significand

$$\Rightarrow \quad f = \frac{B_3\, B_2\, B_1\, B_0}{2^4} = \frac{B_3\, B_2\, B_1\, B_0}{10000} = 0.b_{-1}\, b_{-2}\, b_{-3}\, b_{-4}$$

$$\Rightarrow \quad 0.b_{-1}\, b_{-2}\, b_{-3}\, b_{-4} = \frac{B_3\, B_2\, B_1\, B_0}{2^4}$$

4: # bits in significand field

$$\Rightarrow \quad B_3\, B_2\, B_1\, B_0 = 2^4 \cdot (0.b_{-1}\, b_{-2}\, b_{-3}\, b_{-4})$$

$$\Rightarrow \quad \boxed{B_3\, B_2\, B_1\, B_0 = 2^4 \cdot f}$$

Recall that the hidden bit only appears in the special case of Normalized floating-point numbers in binary 8

$$x = \text{binary8}(B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0)$$

hidden leading bit

$$\Rightarrow \quad x = (-1)^s \cdot (i + f) \times 2^e \qquad \text{where}$$

- $s = B_7 = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x < 0 \end{cases}$

- $f = 0.b_{-1} b_{-2} b_{-3} b_{-4}$ with

  $b_{-1} = B_3,\ b_{-2} = B_2,\ b_{-3} = B_1,\ b_{-4} = B_0$

- $e = u - K = \text{uint3}(B_6 B_5 B_4) - 3$

  $= e_3 e_2 e_1 - 011$

  with $e_3 = B_6,\ e_2 = B_5,\ e_1 = B_4$ and

  $001 \le e_3 e_2 e_1 \le 110$

  $\Leftrightarrow \quad 1 \le \text{uint3}(B_6 B_5 B_4) \le 6$

  $\Leftrightarrow \quad 1 \le u \le 6$

  $\Leftrightarrow \quad 1 - 3 \le u - 3 \le 6 - 3$

  $\Leftrightarrow \quad -2 \le \underbrace{u - K} \le 3$

  $\Leftrightarrow \quad -2 \le e \le 3$ ← range for normalized floating point numbers

Remember that the exponent codes on the boundaries deserve special attention and are NOT normalized floating-point numbers:

lower boundary (subnormals)

□ Subnormal numbers $\Leftrightarrow$ $B_6 B_5 B_4 = 000 \Rightarrow \boxed{e = -2} \neq u - K$

$\Leftrightarrow$ $x = (-1)^s \cdot (0 + f) \times 2^{\boxed{-2}}$

$\Leftrightarrow$ $x = (-1)^s \cdot (0.b_{-1} b_{-2} b_{-3} b_{-4}) \times 2^{-2}$

upper boundary (infinity, NaN, overflow)

□ Infinity and NaN $\Leftrightarrow$ $B_6 B_5 B_4 = 111$ (this bit string gets extra special treatment and doesn't necessarily correspond to a value $e \in \mathbb{Z}$)

$\Leftrightarrow$ $x = \begin{cases} +\infty & \text{if } s=0 \text{ and } B_3 = B_2 = B_1 = B_0 = 0 \\ -\infty & \text{if } s=1 \text{ and } B_3 = B_2 = B_1 = B_0 = 0 \\ \text{NaN} & \text{otherwise} \end{cases}$

NaN stands for "Not a Number" and is not a number at all but instead a special error pattern triggered only when the exponent field $B_6 B_5 B_4$ has bit pattern 111 and at least one $B_K \neq 0$ for $K = 0, 1, 2, 3$.

Now, look back at our tables of all possible binary8 floats and ask fun pattern recognition questions:

- ☐ what is the Δ between sequential floating point numbers for each exponent code?
- ☐ how does this Δ change as we increment exponent codes?
- ☐ consider a specific significand field bit string pattern say $B_3 B_2 B_1 B_0 = 1101$. What happens to mapped value as we vary through range of exponent codes?
- ☐ how is each exponent code related to fixed point format $ufixed5_{2(\ell,f)}(B)$?
- ☐ How complete of a number line visual (in the spirit of the number line on page 35) can you imagine for binary8? Can you visualize all values $x \in \mathbb{Q}_I$ for all exponent codes? If so, what does this look like?

Lesson 7: IEEE 754 Formats (single & double)

Let's begin by getting MATLAB running and open MATLAB documentation on the 16 fundamental (native) data types in MATLAB

(Remember: 10 of these are for numbers)

Let's actually study IEEE 754 single/double by hand.

single-precision
double-precision

$$\boxed{\begin{matrix} x32 \\ \\ x64 \end{matrix}} = \text{binary}32(B) = \text{binary}32(B_{31}\cdots B_1 B_0)$$

$$= \text{binary}64(B) = \text{binary}64(B_{63}\,B_{62}\cdots B_1 B_0)$$

$\boxed{x32}$ / $\boxed{x64}$ ↑ interpreted value(s)

B ↑ raw, uninterpreted string of bits

Let's start by analyzing binary32 in the same spirit as we analyzed binary8

$$B = \begin{array}{|c|c|c|} \hline B_{31} & B_{30}\,B_{29}\cdots B_{24}\,B_{23} & B_{22}\,B_{21}\cdots B_2\,B_1\,B_0 \\ \hline \end{array}$$

exponent field | significand

sign bit | 8-bits | 23-bits

$$= \begin{array}{|c|c|c|} \hline s & e_7\,e_6\cdots e_2\,e_1\,e_0 & b_{-1}\,b_{-2}\cdots b_{-22}\,b_{-23} \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|} \hline s & E & f \\ \hline \end{array}$$

In this case, we have the following explicit map for the relationship between the exponent codes and the interpreted exponent values:

	Exponent code: if the raw, uninterpreted 8-bit exponent word $e_7 \cdots e_0 = E$ is	exponent value: then the excess-K value of $e = u - K$ in decimal is	and the interpreted numerical value of x is
0	0000 0000	Special case of subnormals	$x = \pm (0.b_{-1} \cdots b_{-23}) \times 2^{-126}$
1	0000 0001	$e = \text{uint8}(0000\,0001) - K = 1 - 127 = -126$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-126}$
2	0000 0010	$e = \text{uint8}(E) - K = 2 - 127 = -125$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-125}$
3	0000 0011	$e = \text{uint8}(E) - K = 3 - 127 = -124$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-124}$
4	0000 0100	$e = \text{uint8}(E) - K = 4 - 127 = -123$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-123}$
⋮	⋮	⋮	⋮
125	0111 1101	$e = \text{uint8}(E) - K = 125 - 127 = -2$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-2}$
126	0111 1110	$e = \text{uint8}(E) - K = 126 - 127 = -1$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{-1}$
127	0111 1111	$e = \text{uint8}(E) - K = 127 - 127 = 0$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{0}$
128	1000 0000	$e = \text{uint8}(E) - K = 128 - 127 = 1$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{1}$
129	1000 0001	$e = \text{uint8}(E) - K = 129 - 127 = 2$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{2}$
⋮	⋮	⋮	⋮
252	1111 1100	$e = \text{uint8}(E) - K = 252 - 127 = 125$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{125}$
253	1111 1101	$e = \text{uint8}(E) - K = 253 - 127 = 126$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{126}$
254	1111 1110	$e = \text{uint8}(E) - K = 254 - 127 = 127$	$x = \pm (1.b_{-1} \cdots b_{-23}) \times 2^{127}$
255	1111 1111	Special case of $\pm\infty$ or NaN (could be called "overflow" but is a more nuanced case)	

Range for normalized floats in binary32 (single precision)

If $0000\ 0001 \le E \le 1111\ 1110$ and $x = \text{binary32}(B)$, then

$$x = \pm (1.b_{-1}b_{-2}\cdots b_{-23}) \times 2^{e}$$

$$= (-1)^{s} \cdot (1.b_{-1}b_{-2}\cdots b_{-23}) \times 2^{e}$$

$$= (-1)^{s} \cdot (1.b_{-1}b_{-2}\cdots b_{-23}) \times 2^{u-k} \qquad \text{since } e = u-k$$

where $u = \text{uint8}(E)$

and $k = 2^{8-1} - 1$

$$= (-1)^{s} \cdot (1 + 0.b_{-1}b_{-2}\cdots b_{-23}) \times 2^{\text{uint8}(E)-127}$$

$$= (-1)^{s} \cdot (1 + f) \times 2^{e}$$

$$= (-1)^{s} \cdot \left(1 + \frac{B_{22}B_{21}\cdots B_1 B_0}{2^{23}}\right) \times 2^{e}$$

Example 7.1) Convert $x \in \mathbb{Q}_I$ given by

$$x = \frac{1305}{32}$$ ← this numerator showed up on problem 7 of exam 1 ☺

into both single and double format by hand. Then check our work in MATLAB.

(Hint: $1305 = 1024 + 256 + 16 + 8 + 1$)

Solution:

$$x = \frac{1305}{32}$$

$$= \frac{2^{10} + 2^8 + 2^4 + 2^3 + 2^0}{2^5}$$

$$= \frac{10100011001}{100000}$$

$$= \underbrace{101000.11001}_{5} \times \left[2^{-5} \times 2^5\right]$$

$$= 1.0100011001 \times 2^5$$

$$= +(1 + 0.0100011001) \times 2^5$$

Example 7.1, continued)

Note : $e = 5 = u - K$

$\Rightarrow e = 5 = u - 127$

In binary 32
$K = \boxed{2^{8-1} - 1}$

$\Rightarrow u = e + K$

$= 5 + 127$

$= 128 + 4$

$= 132$

$= 2^7 + 2^2$

$\Rightarrow$ Uint8 $(e_7\ e_6\ e_5\ e_4\ e_3\ e_2\ e_1\ e_0) = 132$

$\Rightarrow B = \underbrace{10000100} = B_{30}\ B_{29} \cdots B_{23}$

binary32$\left(\frac{1305}{32}\right)$

0	10000100	01000110010000000000000

Raw bit string

0100	0010	0010	0011	0010	0000	0000	0000
4	2	2	3	2	0	0	0

hexadecimal

Example 7.1, continued...) Now let's store in binary64:

$$x = 1.0100011001 \times 2^{5}$$

$$e = 5 = u - K$$

11-bits for exponent

$$K = 2^{11-1} - 1$$

$$= 2^{10} - 1$$

$$= 1024 - 1$$

$$= 1023$$

$$\Rightarrow u = 5 + 1023$$

$$\Rightarrow u = 1024 + 4 = 2^{10} + 2^{2} = 1028$$

$$\Rightarrow \text{uint11}(B_{62}\, B_{61} \cdots B_{53}\, B_{52}) = 1028$$

$$\Rightarrow 10000000100$$

Raw bit string

↳ 0 100 0000 0100 0100 0110 0100 0...0

hex → 4 0 4 4 6 4 0 0 00000000

Now let's get into the major foundational ideas of numerical analysis... Not all $x \in \mathbb{R}$ are necessarily $x \in \mathbb{Q}_I$.

But binaryM only encodes $x \in \mathbb{Q}_I$ exactly. All other encodings must be approximate.

Example 7.2) Let $x \in \mathbb{R}$ be equal to $x = 9.4$

$$\Rightarrow \quad x = 9 + 0.4$$

$$= 8 + 1 + 0.4$$

$$= 2^3 + 2^0 + 0.4$$

Note:

- $0.4 = \frac{2}{5}$: what does that mean?

Let's convert this into an binary fraction.

binary fraction step	0.4	Bit string
0		
1	$2 \cdot 0.4 = 0 + 0.8$	0
2	$2 \cdot 0.8 = 1 + 0.6$	1
3	$2 \cdot 0.6 = 1 + 0.2$	1

4	$2 \cdot 0.2 = 0 + 0.4$	0
5		0
6		1
7		1
8		0

$$9.4 = 1001.0110\ 0110\ 0110\ 0110\ \overline{0110}$$

$$= +(1.0010110\ 0110\ 0110\ \overline{0110})\ 2^{3}$$

$$e = 3 = u - K$$

$$\Rightarrow u = 3 + 1023$$

$$\Rightarrow u = 1026 = 1024 + 2$$

$$\Rightarrow E = 10000000010$$

$$= + 1.0010|1100|1100|1100|1100|1100|1100|1100$$
$$1100|1100|1100|1100|1100|1100|1100|1101$$

$$= 0100\ 0000\ 0010$$

$$= 4\ 0\ 2\ 2\ c\ c\ c\ c\ c\ c\ c\ c\ c\ c\ d$$