Types of C variables:

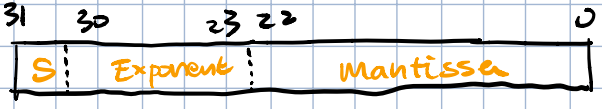
- for the ARM processor:
  - char c; byte (8 bit)
  - short k; half word (16 bits)
  - int i; word (32 bits)
- all of our programs so far have used only integer data, like the above. But if we want use real numbers (eg. 5.75) Then we need:

  float f; word (32 bits. called single precision)

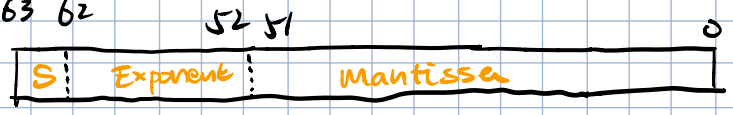
  double d; double-word (64 bits, called double precision)

IEEE floating Point Format

* float



double



* float $f = \pm 1.m \times 2^E$

Example: $f = 100.75$

$$100 = 64+32+4 = 2^6+2^5+2^2 = (1100100)_2$$

$$0.75 = \frac{1}{2}+\frac{1}{4} = 2^{-1}+2^{-2} = 0.11$$

$\therefore$ $100.75 = 1100100.11$

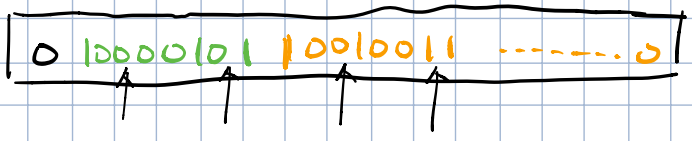
(normalize the number)

$$= 1.1001011 \times 2^6$$

- in the IEEE format, the leading 1 is always present, and so not stored. The "1001011" is the mantissa.

We don't store the 6 as the exponent; Instead, we store $6+127 = 133$ (called excess-127 format)

Here, $133 = 128+4+1 = 2^7+2^2+2^0 = 10000101$

$\therefore f =$ 

$= 0x42C9800$

Example:

$f = -24.15625$

$24 = 16+8 = 2^4+2^3 = 11000$

0.15625
× 2
0.31250
× 2
0.62500
× 2
1.25000
× 2
0.50000
× 2
1.00000

$0.b_1b_2b_3b_4\ldots\ldots$
$\times 2$
$b_1.b_2b_3b_4\ldots\ldots$
$\times 2$
$b_2.b_3b_4\ldots\ldots$

$\therefore -24.15625$
$= -11000.00101001\ldots0$
$= -1.1000010100\ldots0 \times 2^4$

$\therefore E = 4+127 = 131 = 10000011$

$\therefore f = 11000001110000101000\ldots0$

Note: $\pi = 01000000010010010000111\ldots$

$= 0x40490FDB$

Special values:

0: 00000000 0000 ... 00 +/- 0

∞: 11111111 0000 ... 00 +/- ∞

(not a #) NaN: 11111111 !=0

"de-normalized": 00000000 !=0

Range of values:

closest to 0: | % | 00000001 | 000 ..... 00 | +/- 1.0×2^{-126}

largest: | % | 1111 1110 | 1111 ... 11 | +/- $1.11\ldots1 \times 2^{127}$

Example: convert float to decimal value.

f = 0x3F200000 = 0 0111 1110 0100 0000 0

S=+ve

$E = 2^6+2^5+2^4+2^3+2^2+2^1$

$= 126 - 127 = -1$

M = 0100 0

$\therefore f = 1.010\ldots \times 2^{-1} = 0.1010\ldots$

$f = \frac{1}{2} + \frac{1}{8} = 0.5 + 0.125 = 0.625$

Double precision: 64 bits

11 bits exponent (excess 1023)

52 mantissa (more precision)