Floating Point Representations

• The goal of floating point representation is represent a large range of numbers
• Floating point in decimal representation looks like:
  +3.0 \times 10^3, \ 4.5647 \times 10^{-20}, \ etc
• In binary, sample numbers look like:
  -1.0011 \times 2^4, \ 1.10110 \times 2^{-3}, \ etc
• Our binary floating point numbers will always be of the general form:
  (\text{sign}) \ 1.mmmmmm \times 2^{\text{exponent}}
• The sign is positive or negative, the bits to the right of decimal point is the mantissa or \textit{significand}, exponent can be either positive or negative. The numeral to the left of the decimal point is ALWAYS 1 (normalized notation).
Floating Point Encoding

• The number of bits allocated for exponent will determine the maximum, minimum floating point numbers (range)
  \[ 1.0 \times 2^{-\text{max}} \text{ (small number)} \text{ to } 1.0 \times 2^{\text{+max}} \text{ (large number)} \]

• The number of bits allocated for the significand will determine the precision of the floating point number

• The sign bit only needs one bit (negative: 1, positive: 0)
Single Precision, IEEE 754

Single precision floating point numbers using the IEEE 754 standard require 32 bits:

<table>
<thead>
<tr>
<th>1 bit</th>
<th>8 bits</th>
<th>23 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>exponent</td>
<td>significand</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>23</td>
</tr>
</tbody>
</table>

Exponent encoding is bias 127. To get the encoding, take the exponent and add 127 to it.

If exponent is –1, then exponent field = -1 + 127 = 126 = 7 Eh
If exponent is 10, then exponent field = 10 + 127 = 137 = 89h
Smallest allowed exponent is –126, largest allowed exponent is +127. This leaves the encodings 00H, FFH unused for normal numbers.
Convert Floating Point Binary Format to Decimal

1 10000001 010000........0

<table>
<thead>
<tr>
<th>S</th>
<th>exponent</th>
<th>significand</th>
</tr>
</thead>
</table>

What is this number?

Sign bit = 1, so negative.

Exponent field = 81h = 129.

Number is:

-1 \times (01000...000) \times 2^2
-1 \times (0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} \ldots + 0) \times 4
-1 \times (0 + 0.25 + 0 + \ldots + 0) \times 4
-1.25 \times 4 = -5.0.
Convert Decimal FP to binary encoding

What is the number -28.75 in Single Precision Floating Point?

1. Ignore the sign, convert integer and fractional part to binary representation first:
   a. 28 = 1Ch = 0001 1100
   b. .75 = .5 + .25 = 2^{-1} + 2^{-2} = .11

   -28.75 in binary is -00011100.11 (ignore leading zeros)

2. Now NORMALIZE the number to the format 1.mmmm x 2^{exp} Normalize by shifting. Each shift right add one to exponent, each shift left subtract one from exponent:
   
   -11100.11 x 2^{0} = -1110.011 x 2^{1}
   = -111.0011 x 2^{2}
   = -1.110011 x 2^{4}
Convert Decimal FP to binary encoding (cont)

Normalized number is: \(-1.110011 \times 2^4\)

Sign bit = 1

Significand field = 110011000...000

Exponent field = \(4 + 127 = 131 = 83h = 1000\ 0011\)

Complete 32-bit number is:

\[
\begin{array}{ccc}
1 & 10000011 & 110011000...000 \\
S & \text{exponent} & \text{significand}
\end{array}
\]
Overflow/Underflow, Double Precision

- Overflow in floating point means producing a number that is too big or too small (underflow)
  - Depends on Exponent size
  - Min/Max exponents are $2^{-126}$ to $2^{+127}$
    is $10^{-38}$ to $10^{+38}$.
- To increase the range, need to increase number of bits in exponent field.
- Double precision numbers are 64 bits - 1 bit sign bit, 11 bits exponent, 52 bits for significand
- Extra bits in significand gives more precision, not extended range.
Special Numbers

Min/Max exponents are $2^{-126}$ to $2^{+127}$. This corresponds to exponent field values of 1 to 254.

The exponent field values 0 and 255 are reserved for special numbers. Special Numbers are zero, +/- infinity, and NaN (not a number)

Zero is represented by ALL FIELDS = 0.

+/- Infinity is Exponent field = 255 = FFh, significand = 0. +/− Infinity is produced by anything divided by 0.

NaN (Not A Number) is Exponent field = 255 = FFh, significand = nonzero. NaN is produced by invalid operations like zero divided by zero, or infinity − infinity.
Comments on IEEE Format

• Sign bit is placed is in MSB for a reason – a quick test can be used to sort floating point numbers by sign, just test MSB

• If sign bits are the same, then extracting and comparing the exponent fields can be used to sort Floating point numbers. A larger exponent field means a larger number since the ‘bias’ encoding is used.

• All microprocessors that support Floating point use the IEEE 754 standard. Only a few supercomputers still use different formats.