marc-oliver pahl

Hornerschema/ IEEE754

Das Beispiel zeigt noch einmal die Umwandlung einer Zahl von Dezimaldarstellung nach Binärrepräsentation nach dem Hornersschema und die Darstellung einer Zahl in IEEE-754-single und -double precision, sowie die Sonderdarstellungen für 0, positiv Unendlich etc.

18.2.2004

Das Hornerschema (>0)

2

marc-oliver pahl

2342=

0+2*(**1**+2*(**1**+2*(**0**+2*(**0**+2*(**1**+2*(**0**+2*(**0**+2*(**1**+2*(**0**+2*(**0**+2*(**0**+2***1**)))))))))))

```
2342 : 2 = 1171 Rest 0
1171 : 2 =  585 Rest 1
 585 : 2 =  292 Rest 1
 292 : 2 =  146 Rest 0
 146 : 2 =   73 Rest 0
  73 : 2 =   36 Rest 1
  36 : 2 =   18 Rest 0
  18 : 2 =    9 Rest 0
   9 : 2 =    4 Rest 1
   4 : 2 =    2 Rest 0
   2 : 2 =    1 Rest 0
   1 : 2 =    0 Rest 1
```

...

$2342_{10}=100100100110_2$

Das Hornerschema (<0)

3

marc-oliver pahl

0,625=0,5*(**1**+0,5*(**0**+0,5*(**1**+0,5***0**)))

```
0,625 * 2 = 1,25
0,250 * 2 = 0,5
0,500 * 2 = 1,0
0,000 * 2 = 0
```

$0{,}625_{10}=0.101_2$

IEEE-754

4

marc-oliver pahl

$2342{,}625_{10}=100100100110.101_2$

$=(-1)^0*1.00100100110101_2*(2_{10})^{11}$

single precision

| 0 \| 10001010 \| 00100100110101 |

```
 01111111 (bias)
+00001011
=10001010 (exponent)
```

double precision

| 0 \| 10000001010 \| 00100100110101000000000000000000000000000000000 |

```
 01111111111 (bias)
+00000001011
=10000001010 (exponent)
```

There are also special patterns which don't represent normal numbers.
The full IEEE 754 layout is given below:

Single precision		Double precision		Represents
Exponent (8 bits)	Significand (mantissa) (23 bits)	Exponent (11 bits)	Significand (mantissa) (52 bits)	
0	0	0	0	0
0	nonzero	0	nonzero	+/- denormalised number
1-254	anything	1-2046	anything	+/- normalised floating point number
255	0	2047	0	+/- infinity
255	nonzero	2047	nonzero	NaN (Not a Number)

IEEE 754/854 Floating Point layout

A *denormalised* number is a way of allowing very small values (which don't have a 1 immediately after the binary point) and is used in specialised operations.

The two representations for + and -infinity mean that a division by zero can be dealt with *without* having to cause a run-time hardware error. NaN values result from attempts to divide zero by zero, or subtract infinity from itself.

src: http://turing.cs.camosun.bc.ca/comp112/resources/floatingpoint.html