IBM System 360/370 Floating Point

sign	exponent	mantissa
s	e	f
	← 7 bits →	← 24 bits →

- value = $(-1)^s\ 0.f \times 16^{e-64}$
  - Sign bit is 0 for positive number, 1 for negative number
  - Mantissa represented as a fractional value, using signed magnitude
    - Normalized to have leading zero
  - Exponent represented using excess 64 (base assumed to be 16)
- $7.6875_{10} = 111.1011_2$
  $= 0.01111011_2 \times 16^1$
  $= 0.01111011_2 \times 16^{65-64}$
  $= 0\ \ 1000001\ \ 011101100000\ldots0_{\text{IBM 360 fp}}$

DEC PDP 11 & Vax Floating Point

sign	exponent	mantissa
s	e	f
	← 8 bits →	← 23 bits →

- value = $(-1)^s\ 0.1f \times 2^{e-128}$
  - Sign bit is 0 for positive number, 1 for negative number
  - Mantissa represented as a fractional value, using signed magnitude
    - Normalized to begin with 0.1
    - Leading "1" is assumed, and is not explicitly stored (called a *hidden bit*)
  - Exponent represented using excess 128 (base assumed to be 2)
- $7.6875_{10} = 111.1011_2$
  $= 0.1111011_2 \times 2^3$
  $= 0.1111011_2 \times 2^{131-128}$
  $= 0\ \ 10000011\ \ 11101100000\ldots0_{\text{DEC fp}}$

IEEE 754 Floating Point Standard

sign	exponent	mantissa
s	e	f
	← 8 bits →	← 23 bits →

- value = $(-1)^s\ 1.f \times 2^{e-127}$
  - Sign bit is 0 for positive number, 1 for negative number
  - Mantissa represented as a fractional value, using signed magnitude
    - Normalized to begin with 1.xxxxx
    - Leading "1" is assumed, and is not explicitly stored (called a *hidden bit*)
  - Base assumed to be 2
  - Exponent represented using excess 127
- $7.6875_{10} = 111.1011_2$
  $= 1.111011_2 \times 2^2$
  $= 1.111011_2 \times 2^{129-127}$
  $= 0\ \ 10000001\ \ 11101100000\ldots0_{\text{IEEE fp}}$

Table 8.2 The IEEE 754 and DEC floating point formats

s	*e*	*f*	IEEE	DEC
1	1 … 11	1 … 11	NaN	numbers
⋮	⋮	⋮		
1	1 … 11	0 … 01		
1	1 … 11	0 … 00	−∞	
1	1 … 10	1 … 10	numbers	
⋮	⋮	⋮		
1	0 … 01	1 … 01		
1	0 … 00	1 … 11	denormals	reserved
⋮	⋮	⋮		
1	0 … 00	0 … 01		
1	0 … 00	0 … 00	−0	
0	0 … 00	0 … 00	+0	0
0	0 … 00	0 … 01	denormals	
⋮	⋮	⋮		
0	0 … 00	1 … 11		
0	0 … 01	anything	numbers	numbers
⋮	⋮			
0	1 … 10			
0	1 … 11	0 … 00	+∞	
0	1 … 11	0 … 01	NaN	
⋮	⋮	⋮		
0	1 … 11	1 … 11		

DEC vs. IEEE Floating Point

- In the DEC format, we can represent
  - Zero — exponent of 0…0
  - Positive nums w/ exponents –127 to +127
  - Negative nums w/ exponents –127 to +127
- In the IEEE format, we can represent
  - Zero — expon. of 0…0, mantissa of 0…0
  - Positive nums w/ exponents –126 to +127
    - Positive infinity with exponent +128, mantissa of 0…0
    - NAN with exponent +128, mantissa other than 0…0
  - Negative nums w/ exponents –126 to +127
    - Negative infinity with exponent +128, mantissa of 1…1
    - NAN with exponent exponent +128, mantissa other than 1…1
  - Denormals (numbers close to zero) with exponent of 0…0

IEEE 754 Floating Point Standard (cont.)

sign	exponent	mantissa
s	e	f
	← 8 bits →	← 23 bits →

- Additional support for representing numbers very close to zero (*denormals*)
  - value = $(-1)^s\ 0.f \times 2^{-126}$
  - Sign bit as before
  - Exponent assumed to be –126, resented as 0…0 (base assumed to be 2)
  - Mantissa need not be normalized
    - Value such as 1.0×2^{-130} can be represented as 0.0001×2^{-126}

IEEE 754 Floating Point Standard (cont.)

sign	exponent	mantissa
s	e	f
	← 10 bits →	← 53 bits →

- A *double precision* format, shown above, is also available
  - value = $(-1)^s\ 1.f \times 2^{e-1023}$
  - Sign bit as in single precision
  - Mantissa in single precision
    - Normalized to begin with 1.x
    - Leading "1" is assumed, and is not explicitly stored (called a *hidden bit*)
  - Exponent represented using excess 1023 (base assumed to be 2)

Homework #5 — Due 11/9/98 (Part 2)

2. Convert to IEEE 754 floating point single precision format, showing your work as you convert between decimal and binary:

   –18.375