CS330 Architecture and Organization
Assignment Chapter 3

Jingbo Wang

Problem 1 — (8 points) Convert the following binary values to decimal.

(a) 0.101

(b) 110.101

(c) 1 0000 0000.0000 0000 1

(d) 111 1111.1111 111

Answer:

(a)

$$\begin{aligned} 2^{-1} + 2^{-3} &= 0.5 + 0.125 \\ &= 0.625 \end{aligned}$$

(b)

$$\begin{aligned} 2^{1} + 2^{2} + 2^{-1} + 2^{-3} &= 2 + 4 + 0.5 + 0.125 \\ &= 6.625 \end{aligned}$$

(c)

$$\begin{aligned} 2^{8} + 2^{-9} &= 256 + 0.001953125 \\ &= 256.001953125 \end{aligned}$$

(d)

$$\begin{aligned} (2^{7} - 1) + (2^{7} - 1) \cdot 2^{-7} &= (2^{7} - 1) + (1 - 2^{-7}) \\ &= 2^{7} - 2^{-7} \\ &= 127.992188 \end{aligned}$$

Problem 2 — (8 points) Convert the following decimal values to binary.

(a) 0.75

(b) 16.0625

(c) 0.1

(d) 12.12

Answer:
(a)

$$\begin{aligned}
0.75 &= \frac{75}{100} \\
&= \frac{50}{100} + \frac{25}{100} \\
&= 0.5 + 0.25 \\
&= 2^{-1} + 2^{-2} \\
&= 0.11
\end{aligned}$$

(b)

$$\begin{aligned}
16.0625 &= 16 + \frac{625}{10000} \\
&= 2^4 + \frac{1}{16} \\
&= 2^4 + 2^{-4} \\
&= 1\,0000.0001
\end{aligned}$$

(c)

$$\begin{array}{lcl}
0.1 \cdot 2 = 0.2 & \rightarrow & 0 \\
0.2 \cdot 2 = 0.4 & \rightarrow & 0 \\
0.4 \cdot 2 = 0.8 & \rightarrow & 0 \\
0.8 \cdot 2 = 1.6 & \rightarrow & 1 \text{ and subtract one} \\
0.6 \cdot 2 = 1.2 & \rightarrow & 1 \text{ and subtract one} \\
0.2 \cdot 2 = 0.4 & \rightarrow & 0 \\
\vdots & &
\end{array}$$

So, $0.1 = 0.0001\,1001\,1\ldots.$

(d)

$$\begin{aligned}
12 &= 2^3 + 2^2 \\
&= 1100
\end{aligned}$$

$$
\begin{aligned}
0.12 \cdot 2 = 0.24 \quad &\rightarrow \quad 0 \\
0.24 \cdot 2 = 0.48 \quad &\rightarrow \quad 0 \\
0.48 \cdot 2 = 0.96 \quad &\rightarrow \quad 0 \\
0.96 \cdot 2 = 1.92 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.92 \cdot 2 = 1.84 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.84 \cdot 2 = 1.68 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.68 \cdot 2 = 1.36 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.36 \cdot 2 = 0.72 \quad &\rightarrow \quad 0 \\
0.72 \cdot 2 = 1.44 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.44 \cdot 2 = 0.88 \quad &\rightarrow \quad 0 \\
0.88 \cdot 2 = 1.76 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.76 \cdot 2 = 1.52 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.52 \cdot 2 = 1.04 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.04 \cdot 2 = 0.08 \quad &\rightarrow \quad 0 \\
0.08 \cdot 2 = 0.16 \quad &\rightarrow \quad 0 \\
0.16 \cdot 2 = 0.32 \quad &\rightarrow \quad 0 \\
0.32 \cdot 2 = 0.64 \quad &\rightarrow \quad 0 \\
0.64 \cdot 2 = 1.24 \quad &\rightarrow \quad 1 \text{ and subtract one} \\
0.24 \cdot 2 = 0.48 \quad &\rightarrow \quad 0 \\
&\vdots
\end{aligned}
$$

So, $12.12 = 1100.\ 0001\ 1110\ 1011\ 1000\ 0100\ 1111\ 0101\ 1100\ 001\ldots$

Problem 3 — (8 points) Normalize the following fractional binary numbers. Give your answers in scientific notation, expressing the mantissa in binary and the exponents in decimal.

(a) 100.0

(b) 0.1110 0001

(c) 1100 1010.01

(d) 0.0001 01

Answer:

(a)

$$100.0 = 1 \times 2^2$$

(b)

$$0.1110\ 0001 = 1.110\ 0001 \times 2^{-1}$$

(c)

$$1100\ 1010.01 = 1.1001\ 0100 \times 2^7$$

(d)

$$0.0001\ 01 = 1.01 \times 2^{-4}$$

Problem 4 — (8 points) Show how the following binary numbers would be represented in single precision floating point. In each number, the mantissa is given in binary and the exponent is given in decimal. Present your answers in binary.

(a) 1.01×2^0

(b) $-1.1000\ 1100\ 0010 \times 2^{10}$

(c) $1.0100\ 1100\ 0010 \times 2^{-10}$

(d) $1.1101\ 0001\ 1 \times 2^{-127}$

Answer:

(a) $s = 0, E = 0 + 127, m = 0.01$

0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

(b) $s = 1, E = 10 + 127, m = 0.1000\ 1100\ 0010$

1	1	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

(c) $s = 0, E = -10 + 127, m = 0.0100\ 1100\ 0010$

0	0	1	1	1	0	1	0	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

(d) $s = 0, E = -127 + 127, m = 0.1101\ 0001\ 1$

0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Problem 5 — (8 points) Convert the following IEE754 single precision bit patterns to decimal values, using regular or scientific notation as you believe appropriate.

(a) `0x BAC0 0000`

(b) `0x C0D0 0000`

(c) `0x F020 1000`

(d) `0x 7FFF FFFF`

Answer:

(a)

1011 1010 1100 0000 0000 0000 0000 0000

1 0111 0101 1000 0000 0000 0000 0000 000

0111 0101 = 117

$s = 1, e = 117 - 127 = -10, m = 0.1$

So, it is $-1.5 \times 2^{-10} = -0.00146\ldots$

(b)

1100 0000 1101 0000 0000 0000 0000 0000

1 1000 0001 1010 0000 0000 0000 0000 000

$1000\ 0001 = 129$

$s = 1, e = 129 - 127 = 2, m = 0.101$

So, it is $-1.625 \times 2^2 = -6.5$

(c)

1111 0000 0010 0000 0001 0000 0000 0000

1 1110 0000 0100 0000 0010 0000 0000 000

$1110\ 0000 = 224$

$s = 1, e = 224 - 127 = 97, m = 0.0100\ 0000\ 001$

So,it is $(1 + 2^{-2} + 2^{-11}) \times 2^{97} = 2^{97} + 2^{95} + 2^{86}$

(d)

+NaN, although technically the sign bit is not meaningful for not-a-number, so NaN is equally (or arguably more) correct.

Problem 6 — (5 points) The Python programming language has arbitrary integer arithmetic and double precision floating point arithmetic. Consider the following calculations:

```
>>> 2**53
9007199254740992

>>> float(2**53)
9007199254740992.0

>>> 1+2**53
9007199254740993

>>> float(1+2**53)
9007199254740992.0

>>> 3+2**53
9007199254740995

>>> float(3+2**53)
9007199254740996.0
```

Use your knowledge of floating point to explain what's going on. In your explanation, explain what happens for $5 + 2^{53}$ and $6 + 2^{53}$.

Answer:

They are all exactly correct. Without the `float()`, it could calculate the arbitrary precision integer. When we have `float()`, they are converted to double precision floating point numbers. It has 52 stored mantissa bits. So, these integers would all require 53 bits of mantissa to store.

`float(1+2**53)` is rounded down to `float(0+2**53)`, and `float(3+2**53)` is rounded up to `float(4+2**53)`.

For $5+2^{53}$ and $6+2^{53}$,`float(5+2**53)` is rounded down to `float(4+2**53)`, and `float(6+2**53)` can be expressed exactly as a double precision floating point number, so it is `float(6+2**53)`.