1. Other bases

(1) Both octal (base 8) and hexadecimal (base 16) are very common in computers. Octal uses 8 digits, 0 through 7 = 8 \( - 1 \), while hexadecimal requires 16 different digits (we use 0–9 normally, use a or A as the digit meaning (10)\(_{10}\), use b or B as the digit meaning (11)\(_{10}\), and so on to digit f or F meaning (15)\(_{10}\)). Express each of the following numbers in decimal, binary, octal and hexadecimal:

\((7)_{10}, (15)_{10}, (1C)_{16}, (10100101)_2\)

(2) Devise an algorithm for converting the binary representation of a number to the hexadecimal representation of the number (without doing any math). (Hint: \(16 = 2^4\), so consider blocks of 4 binary digits.) Devise similar algorithms to convert hexadecimal to binary, binary to octal, and octal to binary.

(3) Prove that your binary to hexadecimal conversion algorithm is correct.

2. Floating Point Systems

(1) Let \(\text{size}(FP)\) = number of non-zero values in the floating point system FP. Consider the following normalized floating point systems (NFPSs).

\(\text{FP1}: \beta = 4, t = 5, e_{\text{min}} = -4, e_{\text{max}} = +3\)
\(\text{FP2}: \beta = 8, t = 5, e_{\text{min}} = -4, e_{\text{max}} = +3\)
\(\text{FP3}: \beta = 10, t = 5, e_{\text{min}} = -4, e_{\text{max}} = +3\)

(a) For FP1, FP2, and FP3 (where possible) find a base 2 normalized floating point system (NFPS), FP, that represents the given normalized floating point systems using as few extra values as possible.

i.e. find NFPSs where:
\[\forall FP' \in \{FP1, FP2, FP3\}, \exists FP \in NFPS, \forall x \in FP', x = FP(x) \text{ and size}(FP) - \text{size}(FP') \text{ is minimized.} \]
(Hint: for FP3 try to represent (0.1)\(_{10}\) in base 2)

(b) For part (a), what three factors from the original fp system force the choice of the values \(t, e_{\text{min}}\) and \(e_{\text{max}}\) in the base 2 fp system?

(c) Find a base 2 normalized floating point system (NFPS), FP, that approximates the given normalized floating point systems (from (a)) without using extra values.

i.e. find NFPSs where:
\[\forall FP' \in \{FP1, FP2, FP3\}, \exists FP \in NFPS, \sum_{x \in FP'} |x - FP(x)|/|x| \text{ is minimized and size}(FP) \leq \text{size}(FP')\].

(2) Consider a normalized floating point system with base \(\beta = 10, t = 3, e_{\text{min}} = -4\) and \(e_{\text{max}} = +2\).

(a) Evaluate this polynomial, \(3.12x^3 - 2.11x^2 + 4.01x + 10.2\), with \(x = 1.32\). Proceed from left to right, using round to nearest. Show your work.

(b) Evaluate the polynomial proceeding from right to left, using round to nearest. Show your work.

(c) Which calculation is more stable? Explain your answer.

3. Arithmetic using Floating Point Representations

(1) Does the addition of numbers exactly representable in a floating-point system always produce numbers exactly representable in the system? Justify.

(2) Does the subtraction of numbers exactly representable in a floating-point system always produce numbers exactly representable in the system? Justify.

(3) Does the multiplication of numbers exactly representable in a floating-point system always produce numbers exactly representable in the system? Justify.

(4) Does the division of numbers exactly representable in a floating-point system always produce numbers exactly representable in the system? Justify.
Answers

1. Other bases

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>28</td>
<td>11100</td>
<td>34</td>
<td>1C</td>
</tr>
<tr>
<td>165</td>
<td>10100101</td>
<td>245</td>
<td>A5</td>
</tr>
</tbody>
</table>

(2) Count off blocks of 4 binary digits from the right. Each block of 4 digits is a number between 0 and 15, which corresponds to a single hexadecimal digit. Write these hexadecimal digits to complete the conversion. Similar ideas work for the other algorithms.

(3) omitted

2. Floating Point Systems

(1) Let \( \text{size}(FP) \) = number of non-zero values in the floating point system \( FP \). Consider the following normalized floating point systems (NFPSs).

- \( FP_1: \beta = 4, t = 5, e_{\min} = -4, e_{\max} = +3 \)
- \( FP_2: \beta = 8, t = 5, e_{\min} = -4, e_{\max} = +3 \)
- \( FP_3: \beta = 10, t = 5, e_{\min} = -4, e_{\max} = +3 \)

(a) For \( FP_1, FP_2, \) and \( FP_3 \) (where possible) find a base 2 normalized floating point system (NFPS), \( FP \), that represents the given normalized floating point systems using as few extra values as possible.

i.e. find NFPSs where:

\[ \forall FP' \in \{ FP_1, FP_2, FP_3 \}, \exists FP \in NFPS, \forall x \in FP', x = FP(x) \text{ and } \text{size}(FP) - \text{size}(FP') \text{ is minimized.} \]

- \( FP_1: \beta = 4, t = 5, e_{\min} = -4, e_{\max} = +3 \)
  \[ \text{size}(FP_1) = (3 \cdot 4^4) \cdot 8 = 3 \cdot 2^8 \cdot 4^4 = 3 \cdot 2^{11} = 1.5 \cdot 2^{12} \leq 2^{13} \]
- \( FP_1': \beta = 2, t = 10, e_{\min} = -8, e_{\max} = +7 \)
  \[ \text{size}(FP_1') = (2^9) \cdot 16 = 2^9 \cdot 2^4 = 2^{13} \]
- \( FP_2: \beta = 8, t = 5, e_{\min} = -4, e_{\max} = +3 \)
  \[ \text{size}(FP_2) = (7 \cdot 8^3) \cdot 8 = 7 \cdot 8^3 \cdot 2 = 7 \cdot 2^{15} \leq 2^{18} \]
- \( FP_2': \beta = 2, t = 15, e_{\min} = -12, e_{\max} = +11 \)
  \[ \text{size}(FP_2') = (2^{14}) \cdot 24 = 2^{14} \cdot 2^3 \cdot 3 = 3 \cdot 2^{18} \]
- \( FP_3: \beta = 10, t = 5, e_{\min} = -4, e_{\max} = +3 \)

Note: \((0.1)_{10} = (1 \cdot 10^{-1})_{10} \in FP_3\)

Try to write in base 2: \((0.1)_{10} = 1 \cdot 2^{-4} + 3/80 = (0.0001)_2 + 3/80 \]
\[ = 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1/160 = (0.00011)_2 + 1/160 \]
\[ = 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1 \cdot 2^{-8} + 3/1280 = (0.00011001)_2 + 3/1280 \]
\[ = (0.00011001)_2 + 1 \cdot 2^{-9} + 1/2560 = (0.000110011)_2 \]

It turns out that \((.01)_{10} = (0.000110)_2 \) i.e. we can prove:
\[ 0.1 = 2^{-2} \sum_{i=0}^{\infty} 2^{4i-3} + 2^{-4i-2} \]

Thus it is not possible to find a base 2 floating point system to exactly represent a base 10 system when negative exponents are allowed.

(b) For part (a), what three factors from the original fp system force the choice of the values \( t, e_{\min}, \text{and } e_{\max} \) in the base 2 fp system?

The number in the original system with the largest number of significant digits after being converted to base two and the smallest and largest numbers in the entire original system.

(c) Find a base 2 normalized floating point system (NFPS), \( FP \), that approximates the given normalized floating point systems (from (a)) without using extra values.

i.e. find NFPSs where:

\[ \forall FP' \in \{ FP_1, FP_2, FP_3 \}, \exists FP \in NFPS, \sum_{x \in FP'} |x - FP(x)|/|x| \text{ is minimized and } \text{size}(FP) \leq \text{size}(FP'). \]
For 1 and 2: Because we are trying to minimize the relative error we can just take our systems from part(a) that capture all the numbers and approximate them using fewer digits in the mantissa. For 3 we just try to fit as many numbers as we can while using less than size(FP3) values and while very closely representing the largest and smallest values.

FP1: \( \beta = 4, t = 5, e_{\min} = -4, e_{\max} = +3 \)
size(FP1) = \((3 \cdot 4^1) \cdot 8 = 3 \cdot 2^1 1 = 1.5 \cdot 2^1 2 \geq 2^1 2 \)
FP1': \( \beta = 2, t = 9, e_{\min} = -8, e_{\max} = +7 \)
size(FP1') = \((2^8) \cdot 16 = 2^8 \cdot 2^4 = 2^1 2 \)
FP2: \( \beta = 8, t = 5, e_{\min} = -4, e_{\max} = +3 \)
size(FP2) = \((7 \cdot 8^4) \cdot 8 = 7 \cdot 8^5 = 7 \cdot 2^1 5 = 3.5 \cdot 2^1 6 \)
FP2': \( \beta = 2, t = 14, e_{\min} = -12, e_{\max} = +11 \)
size(FP2') = \((2^1 3) \cdot 24 = 2^1 3 \cdot 2^3 \cdot 3 = 3 \cdot 2^1 6 \leq size(FP2) \)
FP3: \( \beta = 10, t = 3, e_{\min} = -4, e_{\max} = +3 \)
size(FP3) = \((9 \cdot 10^2) \cdot 8 = 72 \cdot 100 = 7200 \)
Because 10 is not a power of two we just want to try to match the density of numbers in the range
FP3': \( \beta = 2, t = 9, e_{\min} = -14, e_{\max} = +13 \)
size(FP3') = \((2^9) \cdot 28 = 2^9 \cdot 2^2 \cdot 7 = 2^1 0 \cdot 7 = 1024 \cdot 7 = 7188 \)

(2) Consider a normalized floating point system with base \( \beta = 10, t = 3, e_{\min} = -4 \) and \( e_{\max} = +2 \).

(a) \( x \) is represented as \( 1.32 \times 10^0 \)
\( x^2 = 1.32 \cdot 1.32 = 1.7424 \) is represented as \( 1.74 \times 10^0 \)
\( x^3 = 1.32 \cdot 1.74 = 2.2968 \) is represented as \( 2.30 \times 10^0 \)
\( 3.12 \cdot x^2 = 3.12 \cdot 2.3 = 7.176 \) is represented as \( 7.18 \times 10^0 \)
\( -2.11 \cdot x^2 = -2.11 \cdot 1.74 = -3.6714 \) is represented as \( -3.67 \times 10^0 \)
\( 4.01 \cdot x = 4.01 \cdot 1.32 = 5.2932 \) is represented as \( 5.29 \times 10^0 \)
\( 10.2 \) is represented as \( 1.02 \times 10^1 \)
\( 3.12 \cdot x^3 = 3.12 \cdot 7.18 = 22.358 \) is represented as \( 22.36 \times 10^0 \)
\( 3.12 \cdot x^3 = 3.12 \cdot x^2 + 4.01 \cdot x = 3.51 + 5.29 = 8.8 \) is represented as \( 8.8 \times 10^0 \)
\( 3.12 \cdot x^3 = 3.12 \cdot x^2 - 4.01 \cdot x + 10.2 = 8.8 + 10.2 = 19 \) is represented as \( 1.9 \times 10^1 \)

(b) \( 4.01 \cdot x + 10.2 = 5.29 + 10.2 = 15.49 \) is represented as \( 1.55 \times 10^1 \)
\( -2.11 \cdot x^2 + 4.01 \cdot x + 10.2 = -3.67 + 5.29 = 1.62 \) is represented as \( 1.62 \times 10^0 \)
\( 3.12 \cdot x^3 = 3.12 \cdot x^2 + 4.01 \cdot x + 10.2 = 7.18 + 1.18 \times 10^1 = 18.98 \) is represented as \( 1.90 \times 10^1 \)

(c) For this value of \( x \), both calculations have the same result. Therefore, the result has the same relative error, and the calculations are equally stable.

3. Arithmetic Using Floating Point Representations

Answer: No for all. Easy to find counterexamples.

(1) Overflow
(2) Underflow
(3) Rounding Error, Overflow, Underflow
(4) Rounding Error, Overflow, Underflow