Lab 03 – Number Systems

Enrique Saracho Felix
100406980
CPSC 1150
28/05/2023

Using http://www.cleavebooks.co.uk/scol/calnumba.htm number systems calculator.

Base 10	Base 2
26	11010

Q1- Number Systems Calculator

$(7777)_8$

Base 8	Base 2	Base 10	Base 16
7777	111111111111	4095	FFF

1. What happens if you type 7777 into box for base 2? Or base 5? Or even base 7?
   The calculator displays a message of invalid input.

2. Explain why.
   Because the digit 7 is not part of any of those base systems.
   - Base 2 system goes from 0 – 1.
   - Base 5 system goes from 0 – 4.
   - Base 7 system goes from 0 – 6.

3. Give examples of valid and invalid values:

Base	Valid Examples	Invalid Examples
2	1001101	102 10010.5
8	115	8 79
10	77	A3 F
16	AD	T54 1H3

Q2 – Converting to decimal.

1. $(10101010)_2$
   $0 * 2^0 = 0$
   $1 * 2^1 = 2$
   $0 * 2^2 = 0$
   $1 * 2^3 = 8$
   $0 * 2^4 = 0$
   $1 * 2^5 = 32$
   $0 * 2^6 = 0$
   $1 * 2^7 = 128$
   $(\mathbf{170})_{10}$

2. $(136)_8$
   $6 * 8^0 = 6$
   $3 * 8^1 = 24$
   $1 * 8^2 = 64$
   $(\mathbf{94})_{10}$

3. $(1AF)_{16}$
   $15 * 16^0 = 15$
   $10 * 16^1 = 160$
   $1 * 16^2 = 256$
   $(\mathbf{431})_{10}$

4. $(1C7)_{13}$
   $7 * 13^0 = 7$
   $12 * 13^1 = 156$
   $1 * 13^2 = 169$
   $(\mathbf{332})_{10}$

Q3 – Converting from decimal.

1. $(483)_{10}$ to $(?)_2$
   483 / 2 = 241 remaining 1
   241 / 2 = 120 remaining 1
   120 / 2 = 60 remaining 0
   60 / 2 = 30 remaining 0
   30 / 2 = 15 remaining 0
   15 / 2 = 7 remaining 1
   7 / 2 = 3 remaining 1
   3 / 2 = 1 remaining 1
   1 / 2 = 0 remaining 1
   $(\mathbf{111100011})_2$

2. $(483)_{10}$ to $(?)_8$
   483 / 8 = 60 remaining 3
   60 / 8 = 7 remaining 4
   7 / 8 = 0 remaining 7
   $(\mathbf{743})_8$

3. $(483)_{10}$ to $(?)_{16}$
   483 / 16 = 30 remaining 3
   30 / 16 = 1 remaining 14
   1 / 16 = 0 remaining 1
   $(\mathbf{1E3})_{16}$

4. $(483)_{10}$ to $(?)_7$
   483 / 7 = 69 remaining 0
   69 / 7 = 9 remaining 6
   9 / 7 = 1 remaining 2
   1 / 7 = 0 remaining 1
   $(\mathbf{1260})_7$

Q4 – Bits for one hundred things

To represent 100 different things, we need a total of **7** bits. With one bit we can represent two different things ($2^1 = 2$), 1 and 0. with two, four ($2^2 = 4$). With 6 we can only represent 64 ($2^6 = 64$), but with 7 we can represent up to 128 different things ($2^7 = 128$).

Q5 – One byte of memory

1. How many numbers can be represented?
   256 numbers

2. What is the range of these numbers if they are non-negative?
   From 0 to 255

3. What is the range of these numbers if including negatives? (2's complement)
   From -128 to 127

4. Part 3 but with x bits rather than 8?
   The range is always going to be the number of possible representations (2^x). But half of this number will represent the negative values and the other half will represent 0 plus the positive values. So, the negative values will always be 1 more than the positives.

Q6 – Overflow

What happens when adding $10_{(10)}$ and $120_{(10)}$ (byte numbers) in Java.

- The result is 130_{10} or 10000010_2.
- But, since the range is only from -128 to 127, and Java is using 2's complement to represent the numbers, the leftmost bit is representing a minus sign. Then $10000010_{(2)}$ is actually $-126_{(10)}$.

- This is an error known as **overflow error**, and it happens when the data type used to store the value is not large enough.

Q7 – Radix points

$0.1_{(10)}$ to $?_{(2)}$

0.1 * 2 = **0**.2
0.2 * 2 = **0**.4
0.4 * 2 = **0**.8
0.8 * 2 = **1**.6
0.6 * 2 = **1**.2
0.2 * 2 = **0**.4
0.4 * 2 = **0**.8
0.8 * 2 = **1**.6
0.00011001$_{(2)}$

The value can't be stored with 100% precision because the radix points go on infinitely while the program will use a data type with a limited amount of radix points. This is a **rounding error** and the program will only be able to store and represent an approximate of the real value.