

WEB3CLUBS FOUNDATION LIMITED

Course Instructor: DR. Cyprian Omukhwaya Sakwa

PHONE: +254723584205 Email: cypriansakwa@gmail.com

Foundational Mathematics for Web3 Builders

Implemented in RUST

Lecture 31

July 10, 2024

Number Bases

- In this section we consider various number systems and see how to convert from one system to the other.
- The system in every day use is the decimal (denary) system which uses digits $0, 1, 2, 3, 4, \dots, 9$ and has a base or radix of 10.
- Computers are based on a binary system. The binary system uses the digits 0 and 1 only and has a base or radix of 2.
- We convert a number from base 10 to any other base with the use of the Division Algorithm.
- For instance, we convert a denary number to binary by repeatedly dividing it by 2 and noting the remainder at every stage. This continues until the quotient is 0.

Example 9

53 Integer

Convert a decimal number 53 to binary.

Solution

Using division algorithm, we get

0.53_{10}

$$53 = \underline{26}(2) + 1 \checkmark$$

$$26 = \underline{13}(2) + 0 \checkmark$$

$$\underline{13} = \underline{6}(2) + 1 \checkmark$$

$$\underline{6} = \underline{3}(2) + 0$$

$$\underline{3} = \underline{1}(2) + 1 \checkmark$$

$$\underline{1} = \underline{0}(2) + 1 \checkmark$$

110101

Writing the remainders from bottom going up gives 110101₂. Thus $53_{10} = 110101_2$.

Subscripts tell the base the number is in.

110101

Fractional denary numbers can be converted to binary by repeatedly multiplying by 2 till 1.0 is obtained.

$$\overline{53.8125}_{10}$$

Example 10

Convert 0.8125_{10} to binary.

base 2

.59

Solution

$$\underline{0.8125} \times 2 = \textcircled{1}.625$$

$$\underline{0.625} \times 2 = \textcircled{1}.25$$

$$\underline{0.25} \times 2 = \textcircled{0}.5$$

$$\underline{0.5} \times 2 = \textcircled{1}.0$$

$$\begin{array}{r} 1.625 \\ 1.25 \\ 0.5 \\ 1.0 \end{array}$$

$$\underline{0.1101}_2$$

Writing the circled from up going down gives 1101.

Thus $0.8125_{10} = 0.1101_2$.

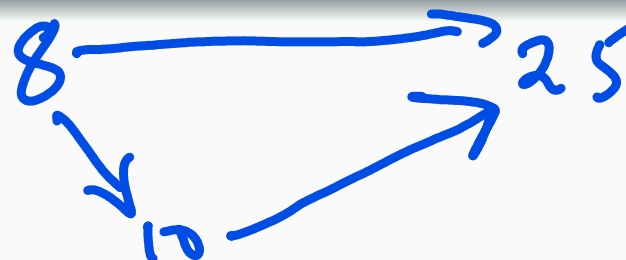
$$\underline{110101.1101}_2$$

$$0.8125$$

$$\underline{53}$$

$$\underline{53.25}$$

$$53 \quad 0.25$$



The following Rust code is used to convert a number from any base to another base. We use it in example 9 and returns same result as above.

0 - 9 ✓

0, 1, 2, ... 9 A B C D E F 16

1A5C.2₁₆ ⇒

```

1 fn main() {
2     let number = "1A5C.2"; // The number to be converted
3     let from_base = 16; // Base of above number
4     let to_base = 10; // Base to convert to
5
6     match convert_base(number, from_base, to_base) {
7         ✓ Some(result)
8         =>
9         println!("{}", in_base, {} is {}, in_base, {}, number, from_base, result, to_base),
10        None => println!("Invalid input."),
11    }
12 }
13
14 fn convert_base(number: &str, from_base: u32, to_base: u32) -> Option<String> {
15     let decimal_value = convert_to_decimal(number, from_base)?;
16     Some(convert_from_decimal(decimal_value, to_base))
17 }
18
19 fn convert_to_decimal(number: &str, base: u32) -> Option<f64> {
20     let parts: Vec<&str> = number.split('.').collect();
21
22     let integer_part = parts.get(0)?;
23     let fractional_part = parts.get(1).unwrap_or(& "");
24 }

```

111 (-2)

1110 110

8

53.025

```

25     let integer_value = convert_integer_part(integer_part, base)?;
26     let fractional_value = convert_fractional_part(fractional_part, base)?;
27
28     Some(integer_value + fractional_value)
29 }
30
31 fn convert_integer_part(part: &str, base: u32) -> Option<f64> {
32     i64::from_str_radix(part, base).ok().map(|v| v as f64)
33 }
34
35 fn convert_fractional_part(part: &str, base: u32) -> Option<f64> {
36     let mut value = 0.0;
37     let mut base_power = base as f64;
38
39     for digit in part.chars() {
40         let digit_value = digit.to_digit(base)? as f64;
41         value += digit_value / base_power;
42         base_power *= base as f64;
43     }
44
45     Some(value)
46 }
47
48 fn convert_from_decimal(number: f64, base: u32) -> String {
49     let integer_part = number.trunc() as u64;
50     let fractional_part = number.fract();
51
52     let integer_str = convert_integer_to_base(integer_part, base);
53     let fractional_str = convert_fractional_to_base(fractional_part, base);
54
55     format!("{}.{}", integer_str, fractional_str)
56 }

```



```

57
58 fn convert_integer_to_base(mut number: u64, base: u32) -> String {
59     let mut result = String::new();
60
61     while number > 0 {
62         let remainder = number % base as u64;
63 result.push(std::char::from_digit(remainder as u32, base).unwrap());
64         number /= base as u64;
65     }
66
67     if result.is_empty() {
68         result.push('0');
69     } else {
70         result = result.chars().rev().collect();
71     }
72
73     result
74 }
75
76 fn convert_fractional_to_base(mut number: f64, base: u32) -> String {
77     let mut result = String::new();
78     let mut count = 0;
79     let precision = 10;
80     // Define the desired precision for the fractional portion.
81
82     while number > 0.0 && count < precision {
83         number *= base as f64;
84         let digit = number.trunc();
85 result.push(std::char::from_digit(digit as u32, base).unwrap());
86         number = number.fract();
87         count += 1;
88     }

```

```
89  
90     result  
91 }
```

Understanding the Rust code

1. Main Function:

- Defines the number to be converted, the base of this number (from_base), and the base to convert to (to_base).
- Calls convert_base with these parameters.
- Prints the result if the conversion is successful; otherwise, it prints "Invalid input."

2. convert_base Function:

- Takes the number as a string and the from_base and to_base as unsigned integers.
- Converts the number from the from_base to a decimal value using convert_to_decimal.
- Converts this decimal value to the to_base using convert_from_decimal.
- Returns the result as an Option<String>.

Understanding the Rust code (conti...)

3. `convert_to_decimal` Function:

- Splits the number into integer and fractional parts.
- Converts each part to decimal separately using `convert_integer_part` and `convert_fractional_part`.
- Adds the integer and fractional values and returns the sum as an `Option<f64>`.
- ✓ Note that `<f64>` represents a 64-bit optional floating-point value that is used to handle cases where a value might be present (`Some(f64)`) or absent (`None`).

4. `convert_integer_part` Function:

- Converts the integer part of the number from the given base to decimal using `i64::from_str_radix`.
- Returns the result as `f64`.

Understanding the Rust code (conti...)

5. `convert_fractional_part` Function:

- Converts the fractional part of the number from the given base to decimal.
- Iterates through each digit, converting it to decimal and summing the values.
- Returns the result as `f64`.

6. `convert_from_decimal` Function:

- Converts the decimal number to the target base.
- Splits the number into integer and fractional parts.
- Converts each part using `convert_integer_to_base` and `convert_fractional_to_base`.
- Combines the converted parts into a single string and returns it.

Understanding the Rust code (conti...)

7. `convert_integer_to_base` Function:

- Converts the integer part of the decimal number to the target base.
- Iteratively divides the number by the base and collects remainders.
- Converts the remainders to the appropriate characters and constructs the result string.

8. `convert_fractional_to_base` Function:

- Converts the fractional part of the decimal number to the target base.
- Iteratively multiplies the number by the base and collects integer parts.
- Converts the integer parts to the appropriate characters and constructs the result string.
-


The Octal number system uses the digits 0, 1, 2, 3, \dots , 7 only and has a base or radix of 8

Example 11

Convert 686_{10} to octal.

Solution

By division algorithm, we have

$$\begin{aligned} 686 &= 85(8) + 6 \\ 85 &= 10(8) + 5 \\ 10 &= 1(8) + 2 \\ 1 &= 0(8) + 1 \end{aligned}$$


Writing the remainders from bottom going up gives 1256_8 .

Thus $686_{10} = 1256_8$

Fractional denary numbers can be converted to octal numbers by repeatedly multiplying by 8 till a whole number is obtained.

Example 12

Convert 0.978515625_{10} to octal.

Solution

$$0.978515625 \times 8 = \textcircled{7}.828125$$

$$0.828125 \times 8 = \textcircled{6}.625$$

$$0.625 \times 8 = \textcircled{5}.0$$

Writing the circled from up going down gives 765.


Thus $0.978515625_{10} = 0.765_8$.

Example 13

Convert the base 10 number 1292 to a base 7 number.

Solution

Using division algorithm, we get

$$\begin{aligned}\underline{1292} &= \underline{184}(7) + 4 \checkmark \\ 184 &= \underline{26}(7) + 2 \checkmark \\ 26 &= \underline{3}(7) + 5 \checkmark \\ 3 &= \underline{0}(7) + 3 \checkmark\end{aligned}$$


Handwritten blue annotations include underlines under 1292, 184, 26, and 3, and checkmarks after each remainder (4, 2, 5, 3). A blue arrow points upwards from the bottom equation to the top one.

Thus $1292_{10} = 3524_7$

15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

Hexadecimal number system uses the digits $0, 1, 2, 3, \dots, 9, A, B, C, D, E, F$, only where A corresponds to 10 and B to 11 and so on. It has a base or radix of 16. We convert a denary number to Hexadecimal number by repeatedly dividing it by 16 and noting the remainder in every stage.

AB

Example 14

1011

Convert 43928_{10} to Hexadecimal.

Solution

$$43928 = 2745(16) + 8 \checkmark$$

$$2745 = 171(16) + 9 \checkmark$$

$$171 = 10(16) + 11 = B \checkmark$$

$$10 = 0(16) + 10 = A \checkmark$$

Thus $43928_{10} = AB89_{16}$

AB89₁₆

Fractional denary numbers can be converted to hexadecimal numbers by repeatedly multiplying by 16 till a whole number is obtained.

Example 15

Convert 0.478759765625_{10} to Hexadecimal.

Solution

$$0.478759765625 \times 16 = \textcircled{7}.66015625$$

$$0.66015625 \times 16 = \textcircled{10}.5625$$

$$0.5625 \times 16 = \textcircled{9}.0$$

Writing the circled from up going down gives 7A9.

Thus $0.478759765625_{10} = 0.7A9_{16}$.

0.7A9
16

To convert a number from another base to base 10 use place-value notation, multiply each digit of the number by the base raised to the power of its position, starting from the rightmost position and moving left. Then sum up all these products. Recall that in 723.56, the 7 represents 7×10^2 , the 2 represents 2×10^1 , the 3 represents 3×10^0 , the 5 represents 5×10^{-1} and the 6 represents 6×10^{-2} so that

$$723.56 = 7 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1} + 6 \times 10^{-2}$$

Example 16

Convert the binary number 11101.11 to decimal.

Solution

$$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 29.75$$

Example 17

Convert the octal number 157.6 to decimal.

Solution

$$1 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 6 \times 8^{-1} = 111.75$$

$$\begin{array}{r} 64 \\ + 40 \\ + 7 \\ + \frac{6}{8} = 0.75 \\ \hline 111.75 \end{array}$$

$$\begin{array}{r} 64 \\ 40 \\ \hline 104 \\ 7 \\ \hline 111.75 \end{array}$$

Example 18

Convert 3523₇ to base 10 number.

Solution

$$3523 = 3 \cdot 7^3 + 5 \cdot 7^2 + 2 \cdot 7^1 + 3 \cdot 7^0 = 3 \cdot 343 + 5 \cdot 49 + 2 \cdot 7 + 3 \cdot 1 = 1291_{10}.$$

Example 19

Convert $15A_{16}$ to decimal.

Solution

$$1 \times 16^2 + 5 \times 16^1 + 10 \times 16^0 = 346_{10}$$

Example 20

Convert $1A5C.2_{16}$ to denary.

Solution

$$1 \times 16^3 + 10 \times 16^2 + 5 \times 16^1 + 12 \times 16^0 + 2 \times 16^{-1} = 6748.125_{10}.$$

Thus $1A5C.2_{16} = 6748.125_{10}$.