Heaven's Light is Our Guide

Rajshahi Universiy of Engineering and Technology

Course Code

ECE 2214

Course Title

Numerical Methods and Discrete Mathematics Sessional

Experiment Date: October 9, 2023

Submission Date: October 14, 2023

Lab Report 5: Modular Exponentiation and Binary Arithmetic in Python

Submitted to	**Submitted by**
Md. Nahiduzzaman	Md. Tajim An Noor
Lecturer	Roll: 2010025
Dept of ECE, Ruet	

Contents

Modular Exponentiation and Binary Arithmetic in Python.

Md Tajim An Noor

1 Introduction

1.1 Modular Exponentiation

Modular arithmetic is the branch of arithmetic mathematics related with the "mod" functionality. Basically, modular arithmetic is related with computation of "mod" of expressions. Expressions may have digits and computational symbols of addition, subtraction, multiplication, division or any other. Here we will discuss briefly about all modular arithmetic operations.
The result of $(a \wedge b \, mod \, m)$ is the modular exponentiation. [1]

1.2 Binary Arithmetic

Binary number system uses only two digits, 0 & 1. The basic arithmetic operations like addition and subtraction are known as binary arithmetic. Binary arithmetic starts from the least significant bit of a binary number and gradually goes towards the most significant bit.

2 Tools Used

- Python
- VS Code - for running python code
- MacTeX -LaTeXcompiler
- VS Code with LaTeXworkshop extension as a text editor

3 Process

3.1 Code:

3.1.1 Modular Exponentiation

```python
base = int(input("Base: "))
expo = int(input("Exponent: "))
mod = int(input("Mod: "))


# function to find modular exponent
def bin_exp(base, exponent, mod):
    ans = 1
    while exponent:
        if exponent % 2 == 1:
            ans = (ans * base) % mod
        base = (base * base) % mod
        exponent = int(exponent) / 2
    return ans


print("Modular exponent: " + str(bin_exp(base, expo, mod)))
```

3.1.2 Binary Arithmetic

```python
a = input("Binary A: ")
b = input("Binary B: ")

# bionary addition function
def bin_add(a, b):
    max_len = max(len(a), len(b))
    a = a.zfill(max_len)
    b = b.zfill(max_len)
    result = ""
    carry = 0
```

```python

    for i in range(max_len - 1, -1, -1):
        r = carry
        r += 1 if a[i] == "1" else 0
        r += 1 if b[i] == "1" else 0
        result = ("1" if r % 2 == 1 else "0") + result
        # carry
        carry = 0 if r < 2 else 1

    if carry != 0:
        result = "1" + result
    return result.zfill(max_len)


# subtraction function, useing 2's compliment
def bin_sub(a, b):
    max_len = max(len(a), len(b))
    a = a.zfill(max_len)
    b = b.zfill(max_len)
    ch = ""

    c = compliment(b)
    result = bin_add(a, c)

    if len(result) > max_len:
        if result[0] == "1":
            compliment(result)
            for i in range(1, len(result)):
                ch += result[i]
        return ch
    else:
        return result


# finding 2's compliment of binary number
def compliment(a):
    newa = ""
    for i in range(0, len(a)):
```

```
        if a[i] == "1":
            newa += "0"
        else:
            newa += "1"
    return bin_add(newa, "1")


print("Sum of A & B = " + bin_add(a, b))
print("Difference of A & B = " + bin_sub(a, b))
```

3.1.3 Integrated

```
# importing the previuosly defined dunctions from other files
from amodularExpo import bin_exp
from bbinaryAddSub import bin_add, bin_sub

base = int(input("Base: "))
expo = int(input("Exponent: "))
mod = int(input("Mod: "))
a = input("Binary A: ")
b = input("Binary B: ")
# using the functions directly
print("Modular exponent: " + str(bin_exp(base, expo, mod)))
print("Sum of A & B = " + bin_add(a, b))
print("Difference of A & B = " + bin_sub(a, b))
```

3.2 Output

Figure 1: Outputs for Modular Exponentiation

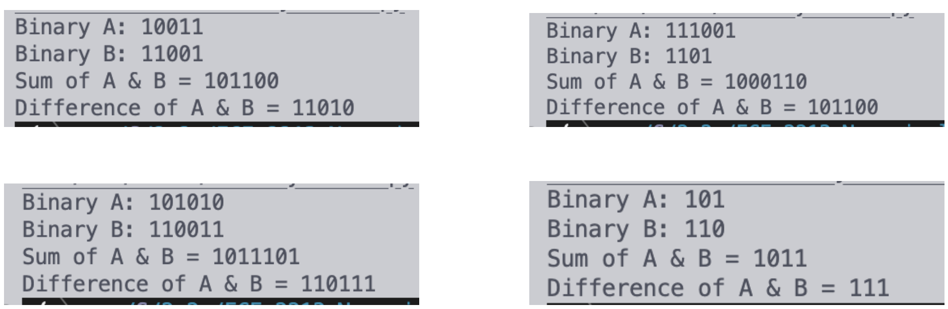

Figure 2: Outputs for Contraposition

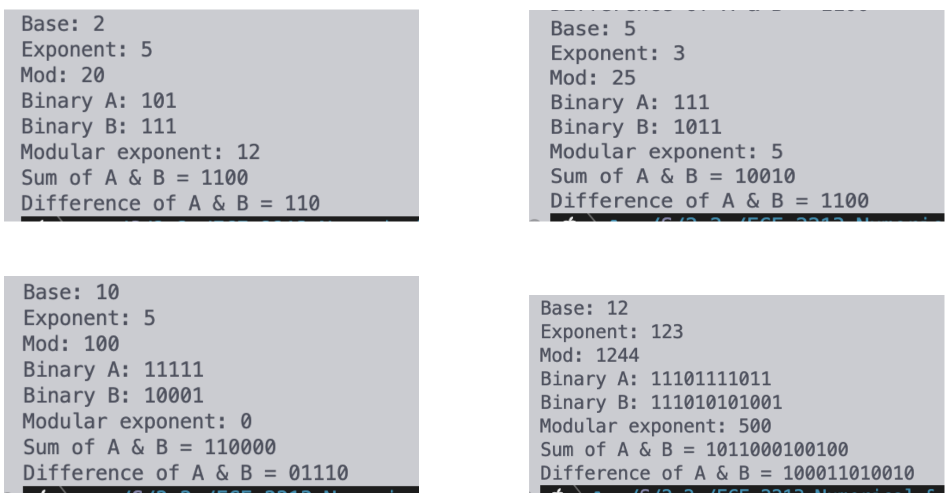

Figure 3: Outputs for Logical Equivalence

4 Discussion

In the first code, the modular exponentiation, 3 inputs are taken: base, exponent & mod. The base is the number that exponent will be done upon, exponent is the power for the base. Using an algorithm, getting way too large number can be avoided. The result is the base to the power of exponent modded by mod.

For the binary arithmetic operations; when adding two binary number, the iteration is done from LSB to MSB. If there is a carry, its calculated accordingly.

For the subtracting operation the 2nd number is converted to its 2's compliment equivalent. And then it was added using the adding function. Then after the addition was done, it was checked if there was any carry bit. If there were any, the result was converted to its equivalent 2's compliment and shown.

References

[1] "Modular Arithmetic," Apr. 2023, [Online; accessed 14. Oct. 2023]. [Online]. Available: https://www.geeksforgeeks.org/modular-arithmetic