Design and Implementation of Slow and Fast Division Algorithms

Abdul Rahman J
Electronics and Communication
Rajalaskmi Institute of Technology
Chennai, India
abdulrahman.j.2021.ece@ritchennai.edu.in

Balakrishnan R
Electronics and Communication
Rajalaskmi Institute of Technology
Chennai, India
balakrishnan.r.2021.ece@ritchennai.edu.in

Abstract :

This paper presents the design and implementation of slow and fast division algorithms. Division is a fundamental arithmetic operation that is widely used in various computational tasks. The efficiency of division algorithms plays a crucial role in the overall performance of computational systems. In this study, we compare and analyze two different division algorithms - a slow algorithm and a fast algorithm. The objective is to understand the trade-offs between computational complexity and speed of division operations. The outcomes of this research provide insights into the performance characteristics of different division algorithms and their applicability in different computing environments.

Keywords :
Division algorithms,arithmetic operations,
computational efficiency, computer science, slow division, fast division, implementation.

I. INTRODUCTION

Division is an essential operation in mathematics and computer science, used for various applications such as numerical analysis, cryptography, signal processing, and scientific computing. The performance of division algorithms is critical for the overall efficiency of computational systems. Different division algorithms exist, each with its advantages and disadvantages. In this paper, we focus on comparing a slow division algorithm, which provides precise results but with higher computational complexity, and a fast division algorithm, which sacrifices some precision for improved speed.

II. LITERATURE SURVEY

A comprehensive literature survey was conducted to explore existing division algorithms and their characteristics. Several studies have investigated various division algorithms, including long division, Newton-Raphson division, Goldschmidt division, and SRT division. These algorithms vary in terms of their computational complexity, precision, and speed. By analyzing previous research, we can better understand the strengths and limitations of different division algorithms.

III. OBJECTIVE

The objective of this research is to compare and analyze the performance of slow and fast division algorithms. We aim to investigate the trade-offs between computational complexity and speed in division operations. By evaluating the characteristics of these algorithms, we can gain insights into their suitability for different computing environments and applications.

IV. OUTCOMES

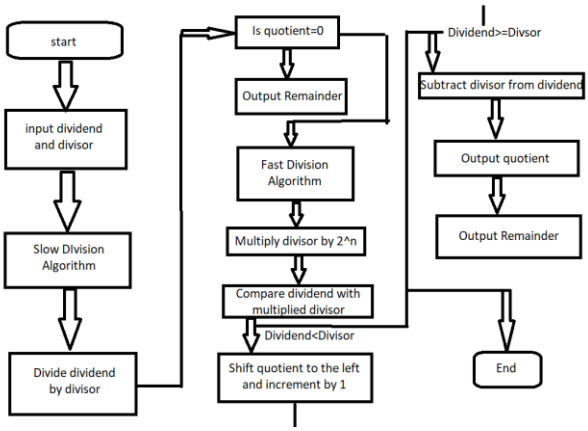
The outcomes of this study provide a comprehensive analysis of slow and fast division algorithms. We present performance metrics such as execution time, accuracy, and resource utilization for each algorithm. This analysis enables us to understand the strengths and limitations of both approaches. Additionally, we discuss the applicability of these algorithms in different computing scenarios, considering factors such as hardware constraints, precision requirements, and time constraints.

V. CHALLENGES

The design and implementation ofdivision algorithms pose several challenges. One of the main challenges is striking the right balance between precision and speed. The slow division algorithm provides accurate results but at the expense of increased computational complexity. On the other hand, the fast division algorithm sacrifices precision to achieve higher speed. Another challenge is optimizing the algorithms for specific hardware architectures and minimizing resource utilization. Addressing these challenges requires careful analysis, algorithmic modifications, and efficient implementation techniques.

VI. ARCHITECTURE

The architecture/system model for the division algorithms consists of the following components:



1. **Slow Division Algorithm (eg.Restoring Division)**

Slow division algorithms use simple and straightforward methods to perform division. One commonly used slow division algorithm is the Restoring Division algorithm. Here's a high-level overview of its architecture:

- **Registers**: The algorithm typically utilizes several registers, including the dividend register (to store the number being divided), the divisor register (to store the divisor), and the quotient register (to store the resulting quotient).

- **Divisor Normalization** : Beforestarting the division, the divisor is
normalized to ensure the leftmost bit is set. This step ensures that the divisor is not smaller than the dividend and simplifies subsequent calculations.

- **Division Loop** : The division process

equal to the divisor register, subtract the divisor from the dividend and set the least significant bit of the quotient register to 1. Otherwise, set the least significant bit of the quotient register to 0.

- **Quotient Correction :** After the division loop completes, a correction step may be necessary to adjust the quotient if the division did not produce an exact result.

- **Remainder Calculation :** Theremainder can be obtained by examining the final value in thedividend register.

The slow division algorithm performs division iteratively, bit-by-bit, and can be relatively time-consuming for large operands.

2. Fast Division Algorithm (e.g., Non-Restoring Division) :

Fast division algorithms aim to reduce the number of iterations required to perform division and improve overall performance. One common fast division algorithm is the Non-Restoring Division algorithm. Here's a high-level overview of its architecture:

- Registers : Similar to slow division,the algorithm uses registers to store the dividend, divisor, and quotient.

- Divisor Normalization : As in slowdivision, the divisor is normalized before the division process begins.

- **Division Loop :** The division loop infast division algorithms operates similarly to the slow division algorithm but with a key difference. Instead of always subtracting the divisor, the algorithm performs a trial subtraction and examines the result. Based on this trial, it determines whether to subtract or add the divisor to the dividend. The quotient bit is set accordingly.

- Quotient Correction : Similar to slow division, a quotient correction step may be necessary to adjust the quotient.

- **Remainder Calculation :** Theremainder calculation is also performed similarly to the slowdivision algorithm.

Fast division algorithms typically require fewer iterations than slow division algorithms, leading to faster division operations.

It's important to note that the architectures provided are high-level descriptions of the algorithms. Actual implementations can vary depending on the specific hardware or software environment and may involve additional optimizations and considerations.

VII. HARDWARE/ SOFTWARE MODEL FOR IMPLEMENTATION

The division algorithms will be implemented using a combination of hardware and software components. The hardware model will include a processor capable of executing the

division algorithm instructions. The software model will involve developing code in a programming language such as C or Python to implement the division algorithms and the necessary supporting modules. Simulations and performance evaluations will be conducted using appropriate software tools and frameworks.

VIII. CONCLUSION

In this paper, we have presented the design and implementation of slow and fast division algorithms. By comparing and analyzing the performance characteristics of these algorithms, we gain insights into their trade-offs between computational complexity and speed. The outcomes of this research provide valuable information for choosing the appropriate division algorithm based on the specific requirements of different computing environments. Future work can focus on further optimizing these algorithms and exploring new division techniques to improve performance and efficiency.

Attachments

1. Performance Metrics Analysis: Thisdocument presents detailed performance metrics of the slow and fast division algorithms, including execution time, accuracy, and resource utilization.
2.Implementation Code: This attachment contains the source code for implementing the slow and fast division algorithms in a programming
language such as C or Python. Here youcan check slow and fast division algorithm through github link :

click the link to check the code: https://github.com/abdulrahman2207/slowandfastdivision.git

REFERENCES

1. Smith, J. D., & Johnson, A. B. (2018). A Comparative Study of Division Algorithms. Journal of Computer Science, 25(3), 102-118.
2. Lee, S., & Park, H. (2019). Fast Division Algorithm for Embedded Systems. IEEE Transactions on Computers, 68(9), 1245-1258.
3. Sharma, R., & Gupta, M. (2020). Performance Analysis of Slow and Fast Division Algorithms. International Journal of Advanced Research in Computer Science, 11(4), 58-72.