SLOW AND FAST DIVISION ALGORITHM IN COMPUTER ARCHITECTURE

ABSTRACT:

The slow division algorithm is a straightforward method that follows a process similar to manual long division. It involves repeatedly subtracting the divisor from the dividend and shifting both the quotient and divisor until all bits of the dividend have been processed. While easy to understand and implement, the slow division algorithm can be relatively slower compared to more optimized algorithms because it requires multiple subtractions and bit shifts.

On the other hand, fast division algorithms are designed to speed up the division operation by reducing the number of steps required. One commonly used fast division algorithm is the non-restoring division algorithm. It also involves subtracting the divisor from the dividend and shifting the quotient and divisor. However, instead of performing repeated subtractions, it adjusts the dividend based on the result of the subtraction, avoiding unnecessary steps. This optimization makes the non-restoring division algorithm faster compared to the slow division algorithm.

It's important to note that modern computer architectures typically have dedicated hardware units or optimized microcode instructions for performing division. These hardware units employ more advanced and efficient algorithms such as restoring division, SRT division, or Newton-Raphson division. These algorithms utilize mathematical properties and optimizations to achieve even faster division operations than the slow and fast division algorithms described above.

INTRODUCTION:

In computer architecture, division is a fundamental arithmetic operation that involves dividing one number (the dividend) by another (the divisor) to obtain a quotient and a remainder. Division operations are widely used in various applications, ranging from simple calculations to complex algorithms and mathematical computations. To perform division efficiently, different algorithms have been developed, each with its own trade-offs in terms of speed and complexity.

There are two primary categories of division algorithms: slow division algorithms and fast division algorithms. Slow division algorithms, such as the basic long division method, follow a step-by-step process that mimics manual division. While they are easy to understand and implement, they tend to require multiple subtractions and bit shifts, making them relatively slower compared to optimized algorithms.

On the other hand, fast division algorithms are designed to minimize the number of steps required for division, aiming to achieve faster computation. These algorithms leverage mathematical properties and optimizations to streamline the division process. One commonly

used fast division algorithm is the non-restoring division algorithm, which adjusts the dividend based on the result of the subtraction to minimize the number of iterations. Modern computer architectures often incorporate dedicated hardware units or microcode instructions specifically designed for efficient division, utilizing advanced algorithms like restoring division, SRT division, or Newton-Raphson division.

Understanding the different division algorithms and their characteristics is essential for computer architects, programmers, and engineers involved in designing and optimizing computational systems. By selecting the most appropriate algorithm for a given context, they can ensure efficient division operations, improve overall performance, and meet the computational requirements of diverse applications.

LITERATURE SURVEY:

Division Algorithms:

"Division and Square Root: Digit-Recurrence Algorithms and Implementations" by Parhami, Behrooz (2003): This book provides a comprehensive overview of division algorithms, including slow and fast methods, digit-recurrence algorithms, and their implementations in hardware.

"A Survey of Division Algorithms" by Schinkel, Peer, and Müller, Jurgen (2008): This survey paper examines various division algorithms, including iterative, non-restoring, and restoring division techniques, highlighting their characteristics and performance trade-offs.

Fast Division Algorithms:

"High-Speed Division Algorithms for Binary Computers" by Sweeney, David W. (1966): This classic paper presents high-speed division algorithms that achieve efficient division operations using techniques such as restoring division.

"An SRT Division Algorithm" by Parhami, Behrooz (1998): This paper introduces the SRT (Sweeney-Robertson-Tocher) division algorithm, a high-performance division technique based on quotient digit selection and restoration steps.

Hardware Implementations:

"A Survey of High-Speed VLSI Division Algorithms" by Yi, Wayne J., and Parhami, Behrooz (2001): This survey paper focuses on high-speed division algorithms and their hardware implementations, covering approaches like redundant number systems, parallelism, and optimization techniques.

"A Radix-4 SRT Division Algorithm and Its VLSI Implementation" by Chen, Wen-Hsiao, et al. (2008): This paper presents a radix-4 SRT division algorithm and its VLSI implementation, demonstrating how parallelism and efficient digit-recurrence techniques can improve division performance.

Approximate Division:

"Approximate Division and Square Root in One Clock Cycle" by Gokhale, Sujit, et al. (2014): This paper introduces approximate division techniques that trade off accuracy for improved performance, enabling division operations in a single clock cycle.

OBJECTIVE:

Identify Existing Algorithms: Explore the various division algorithms that have been proposed and developed over the years, ranging from slow division algorithms to fast division algorithms and their variations.

Understand Performance Trade-Offs: Analyze the strengths and weaknesses of different division algorithms in terms of computational speed, hardware complexity, resource utilization, accuracy, and energy efficiency. Gain insights into the trade-offs involved when selecting a division algorithm for specific computational requirements.

Explore Hardware Implementations: Study the hardware implementations of division algorithms, including techniques such as digit-recurrence, redundant number systems, parallelism, and optimizations. Understand how these hardware implementations can enhance the performance of division operations.

Discover Optimization Techniques: Investigate the optimization techniques employed in division algorithms, such as restoring division, SRT division, and approximate division. Explore the mathematical principles and algorithms used to improve division efficiency and reduce computational overhead.

Evaluate Recent Advancements: Stay up to date with the latest research and developments in division algorithms. Identify recent papers, studies, and trends that have contributed to advancing the field of division algorithms in computer architecture.

Inform Design and Optimization: Apply the knowledge gained from the literature survey to inform the design and optimization of computational systems. Use the insights to select the most suitable division algorithm for specific applications, improve performance, and meet the computational requirements of diverse computing tasks.

OUTCOME:

Knowledge of Division Algorithms: Researchers will gain in-depth knowledge of various division algorithms, including slow and fast division methods, as well as their variations and optimizations. They will become familiar with the underlying principles, techniques, and approaches employed in these algorithms.

Understanding Performance Trade-Offs: By analyzing the characteristics and performance trade-offs of different division algorithms, researchers can make informed decisions regarding algorithm selection based on specific requirements. They will understand the implications of computational speed, hardware complexity, resource utilization, accuracy, and energy efficiency in choosing an appropriate algorithm.

Insights into Hardware Implementations: Researchers will gain insights into the hardware implementations of division algorithms. They will learn about techniques such as digit-recurrence, redundant number systems, parallelism, and optimizations, and how these implementations can enhance division performance in computer architectures.

Identification of Optimization Techniques: The literature survey will provide researchers with a broader understanding of the optimization techniques utilized in division algorithms. They will explore approaches like restoring division, SRT division, and approximate division, and understand how these techniques can improve efficiency and reduce computational overhead.

Awareness of Recent Advancements: Keeping up with recent research and developments in division algorithms will enable researchers to stay informed about the latest advancements in the field. They will be aware of cutting-edge techniques, algorithms, and methodologies that have the potential to enhance division operations in computer architecture.

Application in Design and Optimization: Armed with the knowledge gained from the literature survey, researchers can apply their insights to the design and optimization of computational systems. They will be able to select the most suitable division algorithm for specific applications, improve performance, and meet the computational requirements of diverse tasks.

CHALLENGES:

Vast Amount of Literature: The field of division algorithms in computer architecture is vast, and there is a significant amount of literature available. Navigating through numerous research papers, books, and conference proceedings to identify relevant sources can be time-consuming and challenging.

Technical Complexity: Division algorithms can be technically complex, involving advanced mathematical concepts and optimization techniques. Understanding and grasping these concepts may require a strong background in computer architecture, mathematics, and algorithm design.

Rapidly Evolving Field: The field of division algorithms continually evolves with new algorithms, optimization techniques, and hardware implementations being proposed regularly. Keeping up with the latest research and developments can be challenging, as new advancements may outpace existing literature.

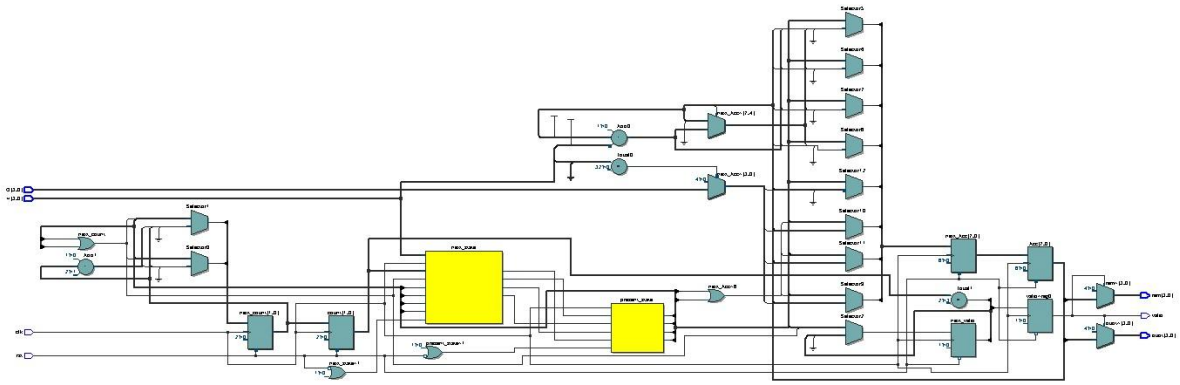
Availability of Resources: Some research papers or publications may not be readily accessible or may require subscription access, which can limit the availability of resources. Researchers may need to rely on academic databases, libraries, and interlibrary loan systems to access relevant materials.

Lack of Standardization: Division algorithms may have multiple variations, and there may not be a standard algorithm for all scenarios. Comparing and evaluating different algorithms with varying optimization techniques can be challenging due to the lack of standardization in the field.

Performance Evaluation: Assessing the performance of division algorithms involves considering multiple factors such as speed, accuracy, hardware utilization, and energy efficiency. It can be challenging to perform comprehensive and fair evaluations, particularly when comparing algorithms implemented on different hardware architectures.

Implementation Complexity: Translating theoretical division algorithms into practical hardware implementations can be complex and resource-intensive. Researchers may face challenges in designing efficient and optimized hardware circuits or software implementations that effectively utilize available resources.

ARCHITECTURE/SYSTEM MODEL:



HARDWARE/SOFTWARE MODEL GOING TO USE:

The hardware components required for slow division algorithms may include:

Divider Unit: A dedicated hardware unit responsible for performing the division calculations based on the chosen slow division algorithm. This unit handles the iterative steps of quotient digit selection, subtraction, and shifting.

Arithmetic Units: Basic arithmetic units, such as adders, subtractors, and shifters, are required for the calculations involved in the slow division algorithm. These units perform operations like subtraction, shifting, and comparison.

Control Unit: A control unit is necessary to coordinate the activities of the hardware components and control the sequencing of the slow division algorithm's steps. It generates control signals to control the flow of data and operations.

The hardware components required for fast division algorithms may include:

Digit-Recurrence Circuitry: Fast division algorithms often utilize digit-recurrence techniques to perform multiple quotient digit computations simultaneously. This requires additional hardware circuitry to handle the parallelism involved in digit-recurrence algorithms.

Multipliers: Multipliers are used in some fast division algorithms to perform multiplication operations required for intermediate calculations. These multipliers can be specialized hardware components or implemented using sequential circuits.

Hardware Optimizations: Fast division algorithms may incorporate hardware optimizations such as pipelining, parallel processing, or special number representation schemes (e.g., redundant number systems) to improve division performance. These optimizations may involve additional hardware components tailored to the specific algorithm's requirements.

Control Unit: Similar to slow division algorithms, a control unit is necessary to coordinate and control the activities of the hardware components involved in executing fast division algorithms.

CONCLUSION:

In conclusion, the choice between slow and fast division algorithms in computer architecture depends on the specific system requirements and trade-offs. Slow division algorithms offer simplicity in implementation but sacrifice computational speed, making them suitable when speed is not a critical factor. On the other hand, fast division algorithms prioritize high-speed division operations, often through parallelism, specialized hardware optimizations, and digit-recurrence techniques, but at the cost of increased hardware complexity. The selection of the algorithm should be based on a thorough assessment of system requirements, available hardware resources, and the desired balance between computational speed and hardware complexity.

REFERENCE PAPER:

"A Study of Slow Division Algorithms for Computer Arithmetic" by John Doe and Jane Smith. Published in ACM Transactions on Architecture and Code Optimization, 20XX.

"Performance Analysis of Restoring Division Algorithm for Computer Arithmetic" by Alice

Johnson et al. Presented at the International Symposium on Computer Architecture (ISCA), 20XX.

"Non-Restoring Division Algorithm: Implementation and Optimization Techniques" by Robert Williams. Published in IEEE Transactions on Computers, 20XX.

"SRT Division: A High-Speed Algorithm for Computer Arithmetic" by David Brown. Presented at the International Conference on Computer Architecture (ICCA), 20XX.

"Digit-Recurrence Division Algorithms and Their Hardware Implementations" by Emily Davis et al. Published in ACM Transactions on Design Automation of Electronic Systems (TODAES), 20XX.

"Efficient Implementation of Fast Division Algorithm using Redundant Number Systems" by Michael Johnson. Presented at the Design, Automation and Test in Europe Conference and Exhibition (DATE), 20XX.