A New Variant of the Barrett Algorithm Applied to Quotient Selection

Niall Emmart*, Fangyu Zheng†‡, Charles Weems*
*College of Information and Computer Sciences, University of Massachusetts, Amherst, MA 01003-4610, USA
†State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China
‡Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
Email: nemmart@yrrid.com, fyzheng@is.ac.cn, weems@cs.umass.edu

***Abstract*—Quotient Selection (QS) is a key step in the classic $O(n^2)$ multiple precision division algorithm. On processors with fast hardware division, it is a trivial problem, but on GPUs, division is quite slow. In this paper we investigate the effectiveness of Brent and Zimmermann's variant as well as our own novel variant of Barrett's algorithm. Our new approach is shown to be suitable for low radix (single precision) QS. Three highly optimized implementations, two of the Brent and Zimmerman variant and one based on our new approach, have been developed and we show that each is many times faster than using the division operation built in to the compiler. In addition, our variant is on average 22% faster than the other two implementations. We also sketch proofs of correctness for all of the implementations and our new algorithm.**

***Index Terms*—multiple precision division, quotient selection, Barrett reduction**

I. Introduction

Quotient Selection (QS) is a simple computation that arises in the implementation of multiple precision division. QS computes the following:

$$q = \left\lfloor \frac{a_1\beta + a_0}{d} \right\rfloor$$
return $\min(q, \beta - 1)$

where β is a power of two (typically 2^w, where w is the machine word size), and $0 \le a_0, a_1 < \beta$, and d is a normalized divisor $\beta/2 \le d < \beta$. On processors with a hardware divide instruction, this is a trivial operation. However, it can be quite slow and often a performance limiter on simpler processors, such as GPU cores, vector processors, DSPs, or embedded processors. Since the same divisor is used repeatedly, a common solution to the problem is to use Barrett's algorithm to replace the division with multiplications. The contributions of this paper are two-fold: (1) we present a novel variant of Barrett's algorithm that allows for faster a implementation of QS; (2) we present three highly optimized implementations of Barrett-based QS (BQS), and show that the implementations are correct. These efficient implementations of BQS, using C and assembly language, are non-trivial and we believe they will be of interest to developers working on similar processors.

The rest of this paper is organized as follows. Section I-A presents background information. Section I-B covers prior work on Barrett's algorithm. Section I-C discusses the hardware requirements to support our BQS implementations. Section II covers Brent and Zimmermann's variant ([4] §2.4.1) of the Barrett algorithm and presents the first two implementations of BQS, which are based on it. Section III covers our proposed variant of Barrett's algorithm and the third BQS implementation. Our experiments and results are presented in Section IV, which show that all three BQS implementations are much faster than the built-in compiler-generated long division on NVIDIA GPUs, and that the third implementation of BQS, based on our variant of Barrett's algorithm, is on average 22% faster than the other two variants. Section V gives our conclusions.

A. Background

In multiple precision (MP) division, we wish to compute $\lfloor X/Q \rfloor$, where X and Q are represented using a fixed radix number system, i.e.,

$$X = \sum_{i=0}^{n-1} x_i\beta^i \qquad \text{and} \qquad Q = \sum_{i=0}^{m-1} q_i\beta^i$$

where β is the radix, $0 \le x_i, q_i < \beta$, and $n \ge m$. MP division algorithms can be classified as either slow quadratic algorithms (such as the grade school long division algorithm) or fast sub-quadratic algorithms. The slow algorithms typically compute a fixed number of quotient bits on each iteration. The fast algorithms include Newton-Raphson, Goldschmidt division [9] and Burnikel-Ziegler's divide and conquer algorithm [5]. The fast algorithms typically rely on grade school division, for example, Newton-Raphson uses grade school division to generate an initial seed and Burnikel-Ziegler uses grade school division for the base cases. Thus a good MP library requires an efficient grade school implementation and one of the key steps of the grade school algorithm is Quotient Selection. For more information about division algorithms and QS see [4], [12], [15].

B. Prior Work on Barrett's Algorithm

In 1984, Barrett [1], [2] introduced an algorithm for the modular reduction operation. His basic idea is replacing the expensive division with a multiplication by a pre-computed constant which approximates the inverse of the modulus. Thus

978-1-5386-2613-9/18/$31.00