Cairo University
Electronics and Communications Department

Computer Arithmetic: One *fast* unit for $\div$ and $\sqrt{}$

Hossam A. H. Fahmy

Multiplicative division

Since $\frac{a}{b} = a \times \frac{1}{b}$, find the reciprocal and then multiply to get the division.

- Use series expansion, $\frac{1}{b} = \frac{1}{1+x} = 1 - x + x^2 - x^3 + \cdots$
- Define $f(x)$ such that $f(x) = 0$ when $x = \frac{1}{b}$ and use the method of Newton-Raphson to get x.

Binomial series

- For this we use $\frac{1}{b} = \frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 - x^5 \cdots = (1-x)(1+x^2)(1+x^4)(1+x^8)\cdots$
- The two's complement of $b = 1 + x$ is $2 - b = 2 - 1 - x = 1 - x$ and $(1+x)(1-x) = 1 - x^2$.
- Similarly, $2 - (1 - x^2) = 1 + x^2$ and $(1-x^2)(1+x^2) = 1 - x^4$.
- We can continue to produce better approximations of $\frac{1}{b}$ by introducing a new factor at each iteration.
- If $0.5 \leq b < 1$ we get $-0.5 \leq x < 0$ which means that x^{2i} starts at position $-2i$ and only affects the following bits. (pre-scale for the IEEE range and correct at the end)
- Doubles the precision each iteration (quadratic convergence).

Starting with a good estimate

Let us use a starter table to find the first exact 8 bits of $\frac{1}{b}$.

- We got $(1-x)(1+x^2)(1+x^4) + \epsilon_0$ with $|\epsilon_0| < 2^{-8}$.
- Now, we multiply by $b = (1+x)$ to get $1 - x^8 + b\epsilon_0$ which yields $1 + x^8 - b\epsilon_0$ after the two's complement.
- The multiplication gives
$$(1 - x^8 + b\epsilon_0)(1 + x^8 - b\epsilon_0) = 1 - x^{16} + 2x^8 b\epsilon_0 - b^2\epsilon_0^2$$
- The new error is $\epsilon_1 = 2x^8 b\epsilon_0 - b^2\epsilon_0^2 = 2(b-1)^8 b\epsilon_0 - b^2\epsilon_0^2$.
- With $\frac{1}{2} \leq b < 1$ and $\epsilon_0 < 2^{-8}$ we get $\epsilon_1 < 2^{-16}$. The error decreases with the rate of increase of quotient bits.

How big is the table?

- Let us assume that we want the first exact m bits of $\frac{1}{b}$, i.e. $\epsilon_0 < 2^{-m}$. $\Rightarrow$ each entry in the table is m bits.

- We use the first n bits of b to access the table. Hence, each entry covers the range $\frac{1}{b} \to \frac{1}{b-2^{-n}}$.

We want

$$
\begin{aligned}
\left|\frac{1}{b} - \frac{1}{b-2^{-n}}\right| &< 2^{-m} \\
\left|\frac{-2^{-n}}{b(b-2^{-n})}\right| &< 2^{-m} \\
2^{-n} &< 2^{-m}b^2 - 2^{-n-m}b \\
2^{-n} &< 2^{-m}b^2
\end{aligned}
$$

Since $\frac{1}{2} \leq b < 1$, we get $2^{-n} < 2^{-m-2}$, i.e. $n > m+2$.

Optimization of the table

- We can treat $b = \frac{1}{2}$ as a special case outside the table.

- Notice now that for all the other cases $b = 0.1b_2b_3b_4\cdots$, the leading bit is always 1.

- Hence, we can reduce n by one bit.

To conclude, for this case, the table has 2^{m+2} entries each m bits wide to give $\epsilon_0 < 2^{-m}$.

Newton-Raphson

We define a function $f(x) = 0$ at $\frac{1}{b}$ and attempt to find its root.

The iteration in Newton-Raphson method is:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

- For $f(x) = \frac{1}{b} - x$ we get $x_{i+1} = \frac{1}{b}$ which requires us to calculate the reciprocal by some other way!

- For $f(x) = \frac{1}{x} - b$ we get

$$x_{i+1} = x_i - \frac{\frac{1}{x} - b}{\frac{-1}{x^2}} = x_i(2 - bx_i).$$

What is the error term?

If $x_i = \frac{1}{b} - \epsilon_i$ then

$$
\begin{aligned}
x_{i+1} &= (\frac{1}{b} - \epsilon_i)(2 - b(\frac{1}{b} - \epsilon_i)) \\
&= \frac{1}{b}(1 - b\epsilon_i)(1 + b\epsilon_i) \\
&= \frac{1}{b}(1 - b^2\epsilon_i^2) \\
&= \frac{1}{b} - b\epsilon_i^2
\end{aligned}
$$

Hence, $\epsilon_{i+1} = b\epsilon_i^2$. The error decreases quadratically.

A simple example

Example 1 Find $\frac{1}{b}$ to at least three decimal digits where $b = 0.75$. Include a calculation of the error $= \epsilon$.

Solution: We start by $X_0 = 1$ and iterate.

$X_0 =$	$= 1$	$\epsilon_1 = 0.333334$
$X_1 = 1(2 - 0.75)$	$= 1.25$	$\epsilon_2 = 0.083334$
$X_2 = 1.25\,(2 - (1.25 \times 0.75))$	$= 1.328125$	$\epsilon_3 = 0.005208$
$X_3 = X_2\,(2 - (1.328125 \times 0.75))$	$= 1.333313$	$\epsilon_4 = 0.000021$

- Obviously, we can get a better starting estimate by using a small table.
- If we start with $x_0 = 1$ in NR we get the same expansion as the binomial:

$$\begin{aligned} x_1 &= 2 - b = (1 - (b - 1)) \\ x_2 &= (2 - b)(2 - b(2 - b)) = (1 - (b - 1))(1 + (b - 1)^2) \end{aligned}$$

The NR method and the binomial are different ways of viewing the same problem with a slight difference in the implementation.

Implementation issues

- Size of the table:
  - For IEEE, an initial estimate with 13 bits is optimal: in two iterations: $13 \rightarrow 26 \rightarrow 52$.
  - However, that means a table that has about $2^{15} = 32k$ entries.
- The NR iteration needs two *sequential* multiplies.
- The binomial series needs two multiplies as well but they can be in parallel. (i.e. $(1 + x^4)(1 - x^4)$ in parallel with $(1 - x)(1 + x^2)(1 + x^4)$)
- The product $(1 - x^8)(1 + x^8)$ is just an 8×8 multiplication. It is possible to use small multipliers to optimize the hardware.
- Instead of iterating, we can duplicate the hardware (several multipliers in sequence, this is a large area!) and pipeline it.

Remainder issues

- Both algorithms may produce non-terminating sequences for terminating quotients, $1/0.8 = 1.2499999\cdots$
- Both algorithms produce the quotient but not the remainder which is required according to IEEE to be positive.
  - At the end of the iterations, get $1 - bq$ (bq is a $2n$ wide multiplication) to find the remainder.
  - If that value is negative, correct q.

Getting the square root as well

If we use $f(x) = b - x^2$ the NR iteration becomes $x_{i+1} = \frac{x_i}{2} + \frac{b}{2x_i}$. This involves a division which is slow.

- Find the reciprocal of the square root ($\frac{1}{\sqrt{b}}$) and then multiply by b to get the square root.
- Often, it is the reciprocal square root that is needed.

The reciprocal square root

With $f(x) = b - \frac{1}{x^2}$ we have $f'(x) = \frac{2}{x^3}$ and

$$x_{i+1} = \frac{x_i}{2}(3 - bx_i^2)$$

This converges quadratically. It needs three multiplications per iteration and no divisions.

If the needed function is indeed $\sqrt{b}$, then a final multiplication is required.

Do we need $\sqrt{}$ in hardware?

- The square root and its reciprocal are even less frequent than division. (About 9 times less frequent.)
- With some minimal support in the hardware, the time latency of the square root can be better than nine times that of division and it will not cause a deterioration to the system performance.

Conclusions

1. It is possible to implement very fast dividers in a large area!
2. The multiplication, division, and square root operations can share the same unit if the effect on the system performance is studied carefully.
3. We need to provide accurate results according to the standard.

What to expect?

- Improvements to the FP addition.
- Recent research ($\approx$ 10 years) in the field.
- Other functions ($e^x, \log x, \sin x, \cdots$)