RECIPEOCAL, DIVISION, RECIPEOCAL SQUARE ROOT AND SQUARE ROOT BY ITERATIVE APPROXIMATION

- AN INITIAL APPROXIMATION OF A FUNCTION ITERATIVELY IMPROVED
- BASED ON MULTIPLICATIONS AND ADDITIONS (vs. only additions and shifts)
- CONVERGES TOWARDS THE RESULT WITH A QUADRATIC OR LINEAR RATE
- QUOTIENT: RECIPROCAL OF THE DIVISOR × THE DIVIDEND
- SQUARE ROOT: INVERSE SQUARE ROOT × THE OPERAND
- ROUNDDING HARDER THAN FOR THE DIGIT-RECURRENCE METHOD
- VARIATIONS TO OBTAIN DIRECTLY QUOTIENT AND SQUARE ROOT
NEWTON-RAPHSON’S METHOD FOR RECIPROCAL APPROXIMATION

- BASED ON A GENERAL METHOD TO OBTAIN THE ZERO OF A FUNCTION (THE VALUE OF $x$ FOR WHICH $f(x) = 0$)
- $x[j]$ AN APPROXIMATION OF THE ZERO
- A BETTER APPROXIMATION IS
  
  $$x[j + 1] = x[j] - \frac{f(x[j])}{f'(x[j])}$$

  $f'(x[j])$ EVALUATED AT $x[j]$

- APPLY TO RECIPROCAL FUNCTION $f(R) = 1/R - d$
  (whose zero is $1/d$)

- RECURRENCE
  
  $$R[j + 1] = R[j](2 - R[j]d)$$

- INITIAL APPROXIMATION $R[0]$
Figure 7.1: Newton-Raphson iteration for finding reciprocal.
• EACH ITERATION REQUIRES TWO MULTIPLICATIONS AND
  ONE SUBTRACTION
• QUADRATIC CONVERGENCE
• RELATIVE ERROR $\epsilon[j]$

\[
\epsilon[j] = 1 - dR[j]
\]

\[
R[j + 1] = \left(\frac{1 - \epsilon[j]}{d}\right) \left(2 - (1 - \epsilon[j])\right)
\]

\[
= \frac{1 - \epsilon[j]^2}{d}
\]

\[
\implies \epsilon[j + 1] = 1 - dR[j + 1] = \epsilon[j]^2
\]
• NUMBER OF ITERATIONS DEPENDS ON INITIAL APPROXIMATION

\[ \epsilon[0] \leq 2^{-k} \]

• TO GET AN ERROR

\[ \epsilon[m] \leq 2^{-n} \]

THE NUMBER OF ITERATIONS IS

\[ m = \left\lceil \log_2 \left( \frac{n}{k} \right) \right\rceil \]
EXAMPLE

RECIPIROCAL OF $d = 5/8$

$R[0] = 1$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$1$</td>
<td>$5 \times 2^{-3}$</td>
<td>$11 \times 2^{-3}$</td>
<td>$11 \times 2^{-3}$</td>
<td>0.14</td>
</tr>
<tr>
<td>1</td>
<td>$11 \times 2^{-3}$</td>
<td>$55 \times 2^{-6}$</td>
<td>$73 \times 2^{-6}$</td>
<td>$803 \times 2^{-9} = 1.5683594$</td>
<td>0.020</td>
</tr>
<tr>
<td>2</td>
<td>$803 \times 2^{-9}$</td>
<td>$4015 \times 2^{-12}$</td>
<td>$4177 \times 2^{-12}$</td>
<td>$3354131 \times 2^{-21} = 1.5993743\ldots$</td>
<td>0.00039</td>
</tr>
</tbody>
</table>

EXACT RESULT: $1/d = 8/5 = 1.6$
MULTIPLICATIVE NORMALIZATION METHOD

\[ R = \frac{1}{d} = \frac{1}{dp[0]p[1]} \cdots \frac{p[m]}{dp[m]} = \frac{R[m]}{d[m]} \]

\[ R = R[m] \text{ if } d[m] = 1 \]

- DEFINE APPROXIMATION \( R[j] = \prod_{i=0}^j p[i] \) AND \( d[j] = dR[j] \)

- IMPROVE APPROXIMATION BY

\[ R[j + 1] = R[j]p[j + 1] \]
MULTIPlicative NORMALIZATION: REciproCAL

Figure 7.2: Illustration of iterations in the multiplicative normalization method.
DETERMINATION OF $P[j]$ FOR QUADRATIC CONVERGENCE

- DEFINE

$$d[j] = d^{j-1} \prod_{i=0}^{j-1} P[i]$$

- OBTAIN THE RECURRENCE

$$d[j] = d[j - 1]P[j - 1]$$

- FOR QUADRATIC CONVERGENCE, IF

$$d[j - 1] = 1 - \epsilon[j - 1]$$

THEN

$$d[j] = 1 - \epsilon[j - 1]^2$$

- CONSEQUENTLY,

$$P[j - 1] = 1 + \epsilon[j - 1]$$

AND

$$d[j - 1] + P[j - 1] = 1 - \epsilon[i - 1] + 1 + \epsilon[i - 1] = 2$$

SO THAT

$$P[j - 1] = 2 - d[j - 1]$$
1. Obtain approximation $P[0]$ to $1/d$
2. $d[0] = dP[0];\ R[0] = P[0]$
3. For $j = 0, 1, 2, 3, \ldots, m - 2$ do
   \[
   P[j + 1] = 2 - d[j] \\
   \]
4. $P[m] = 2 - d[m - 1];\ R[m] = R[m - 1]P[m]$
Figure 7.3: Multiplicative normalization for reciprocal: (a) Implementation with a 2-stage multiplier. (b) Timing diagram.
INITIAL APPROXIMATION

1. PERFORM A TABLE LOOK-UP BASED ON TRUNCATED $d$
   - GOOD FOR RELATIVELY LOW PRECISION INITIAL APPROXIMATION
   - PIECEWISE LINEAR APPROXIMATION IF TABLE TOO LARGE
     \[
     d = d_t 2^{-k} + d_p 2^{-p} + d_r 2^{-n}
     \]
     MS $k$ bits of $d$ used to access the table to get coefficients $a$ and $b$. Then
     \[
     R[0] = a + b d_p 2^{-p}
     \]
   - requires a table look-up and a small multiplication

2. BIPARTITE METHOD: OBTAIN TWO VALUES FROM TABLES AND PERFORM AN ADDITION
   - uses larger tables and adder
IMPLEMENTATION AND EXECUTION TIME

- MODULE TO COMPUTE THE INITIAL APPROXIMATION
- MULTIPLIER
- WIDTH OF PRODUCTS:

\[ R[j] \quad R[j]d \quad R[j+1] = R[j](2-R[j]d) \]

<table>
<thead>
<tr>
<th>j=0</th>
<th>a</th>
<th>a+n</th>
<th>2a+n</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=1</td>
<td>2a+n</td>
<td>2a+2n</td>
<td>4a+3n</td>
</tr>
<tr>
<td>j=2</td>
<td>4a+3n</td>
<td>4a+4n</td>
<td>8a+7n</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- AT ITERATION \( j \) APPROXIMATION HAS A PRECISION OF \( 2^j a \) BITS
  \( \implies \) OK TO KEEP PRODUCTS AT THIS PRECISION
  \( \implies \) NEED NOT PERFORM MULTIPLICATIONS AT FULL PRECISION
ALTERNATIVES

1. USE A FLOATING-POINT MULTIPLIER PRODUCING A ROUNDED PRODUCT

2. USE A RECTANGULAR MULTIPLIER
   - A SEQUENCE OF MULTIPLICATIONS AS PRECISION INCREASES
   - RECTANGULAR MULTIPLIER SMALLER AND FASTER THAN THE SQUARE MULTIPLIER
COMPARISON OF NUMBER OF CYCLES FOR FULL AND RECTANGULAR MULTIPLIER ALTERNATIVES

- RECIPROCAL OF 54 BITS STARTING WITH \( r[0] \) ACCURATE TO 8 BITS
- MULTIPLIER IN SCHEME A STANDARD FLOATING-POINT MULTIPLIER
- MULTIPLIER IN SCHEME B A DEDICATED MULTIPLIER
- OPERATION REQUIRES AT LEAST THREE ITERATIONS

EACH CONSISTING OF TWO CONSECUTIVE MULTIPLICATIONS

IGNORE THE DELAY OF OBTAINING \( 2 - R[i]d \)
1. SCHEME A: Full multiplier $55 \times 55 \rightarrow 55$ (rounded);
   3 cycles per multiply; total: $1 + 3 \times 2 \times 3 = 19$ cycles

2. SCHEME B: Rectangular multiplier $55 \times 16 \rightarrow 55$;
   1 cycle per multiply; total: $1 + 2 + 2 + 4 = 9$ cycles

- $R[1] = R[0](2 - dR[0])$ to 16 bits we use $55 \times 16$ multiplier twice (2 cycles);
- $R[2] = R[1](2 - dR[1])$ to 32 bits we use $55 \times 16$ multiplier twice (2 cycles);
- $R[3] = R[2](2 - dR[2])$ to 54 bits we use $55 \times 16$ multiplier four times (4 cycles).
- A COMPLEMENTER (2's OR 1s')
DIVISION

- TO GET QUOTIENT

\[ Q = R[m]x \]
EXAMPLE OF IMPLEMENTATION: AMD-K7 FLPT UNIT

- DIVISION (20 CYCLES) AND SQUARE ROOT (27 CYCLES)
- DOUBLE PRECISION (53 bits); INTERNAL PRECISION: 76 bits (FOR EXTENDED FORMAT)
- USES 4-STAGE PIPELINED MULTIPLIER: 76 \times 76 PRODUCING 152 BITS
- RADIX-8 MULTIPLIER RECODING WITH \{-4, \ldots, 4\}
- INITIAL APPROXIMATION: BIPARTITE TABLE LOOKUP (69K BITS + ADDER)
Figure 7.4: Block diagram of a division/square-root unit (Adapted from Oberman 1999)
1. [Initialize]
   \[ P[0] \leftarrow \text{RECIP}(\hat{d}) \]
   \[ d[0] \leftarrow d; \ q[0] \leftarrow x \]

2. [Iterate]
   \[ \text{for } j = 0, 1 \]
   \[ d[j + 1] \leftarrow d[j] \times p[j]; \ q[j + 1] \leftarrow q[j] \times p[j] \]
   \[ p[j + 1] = \text{CMPL}(d[j + 1]) \]
   \[ \text{end for} \]

3. [Terminate]
   \[ REM \leftarrow d \times q[3] - x \]
   \[ q \leftarrow \text{ROUND}(q[3], REM, \text{mode}) \]

where

- \text{RECIP} produces the initial approximation of \( 1/d \) in three cycles.
- \text{CMPL}(a) performs bit complementation of \( a \).
- \text{REM} is a negated remainder.
- \text{ROUND} produces a quotient rounded according to the specified \text{mode}