CHAPTER 5

Review: Basic Integer Division

\[ \frac{x}{d} = q + w \]

Dividend: \( 2n \) digits
Divisor: \( n \) digits

\[ x = dq + w \]
\[ 0 \leq |w| < |d| \]
\[ \text{sign}(w) = \text{sign}(x) \]

For a quotient of \( n \) digits

\( \text{Dividend: } 2n \) digits
\( \text{Divisor: } n \) digits

Integer Division

- Use \( n \) iterations of the following recurrence
  \[ w[0] = x \]
  \[ w[j+1] = rw[j] - d^r q_{n-j} \]
  \[ j = 0 \ldots n-1 \]

where \( q = \sum_{j=0}^{n-1} q_j r^j \)

\( d = dr^n \)

Divisor is aligned with most significant half of residual

Example: 721/3

\[ x = 721 = 000721 \]
\[ d = 3 = 003 \]

\[ w[0] - x \]
\[ w[j+1] = rw[j] - d^r q_{n-j} \]
\[ j = 0 \ldots n-1 \]

\[ q = 240, \ \text{rem} = 001 \]

\( (240)(3) + 1 = 721 \)
Integer Division

- In base 10, find q digits by guessing!
- Or by repeated subtraction
- Basic quotient digit set is \{0,1,2,3,4,5,6,7,8,9\}

- To get residual bound of \( q \leq d^* \) use
  \[ q_{n-1} = k \text{ if } d^* k \leq r w[j] < d^*(k+1) \]
  \( 0 \leq k \leq r - 1 \)

Quotient Bit Selection

- Keep subtracting to compute \( q_{n-1} \) until result is negative
  - Then add \( d^* \) back in to get correct residual

\[
\begin{align*}
10 \mod w[0] &= 007210 & \text{div} \ w[0] = 007210 \\
&= 003000 & d^* (1) = 003000 \\
&= 004210 & d^* (2) = 000000 \\
&= 001210 \\
\text{Add } d^* & \text{ back} \ 003000 & \text{q=2, residual = 001210}
\end{align*}
\]

Quotient Bit Selection

- Messy for higher bases
  - Easier for binary – add 1, then fix if negative

- Compute tentative residual
  \[ \bar{w}[j+1] = 2w[j] - d^* \] \( (\text{assume } q=1) \)

  if \( \bar{w}[j+1] \geq 0 \Rightarrow q_{n-1} = 1, \ w[j+1] = \bar{w}[j+1] \)

  else \( q_{n-1} = 0, \ w[j+1] = 2w[j] = \bar{w}[j+1] + d^* \)

  This is Restoring Division

Restoring Division

- Combine the restoration with the next iteration
  - \( w[j] < 0 \Rightarrow q_{n-1} = 0 \)
  - \( w[j] \geq 0 \Rightarrow q_{n-1} = 1 \)

  \[ w[j+1] = 2w[j] - d^* \]

  then \( w[j+1] = 2w[j] - d^* \),

  \[ w[j+1] = 2(\bar{w}[j] + d^*) - d^* \]

  \[ = 2(\bar{w}[j] + d^*) - d^* \]

  \[ = 2\bar{w}[j] + d^* \]

  - So – residuals can be positive or negative
  - Bound is \( \bar{w}[j] < d^* \)

  - May need final restoring step to get positive rem
Nonrestoring Division

Restoring vs. Nonrestoring

- Can think of them as
  - Restoring = q digit set = \{0, 1\}
  - Nonrestoring = q digit set = \{-1, 1\}

  for purposes of residual computation
  \[ q_{n-j} = 1 \text{ if } w[j] \geq 0 \text{ else } -1 \]

- Compute residual directly
  \[ w[j+1] = 2w[j] - d'q_{n-j} \]

Speed up Division!

- Unroll the iterations and do them concurrently
  - Worked for multiplication!

- Unfortunately, each iteration depends on a result from the previous iteration
  - Inherently sequential!
Speed up Division!

- How about using a carry-save adder for the residual computation?
  - Should make the iteration shorter...

- BUT, we need the sign of w[j] at each iteration to tell what to do!
  - How do you tell the sign of a CS result?
  - Add the ws and wc vectors together to convert to conventional form
  - You're back to CPA-delay!

Speed up Division!

- How about using higher-radix divisor to compute more than one bit per iteration?
  - Fewer iterations...

  - How do you choose the quotient bits?
    - With radix-2 there are only two choices
      - Guess 1 and fix if wrong (restoring),
        or determine 1 or -1 each time (nonrestoring)
    - With radix-4 there are four choices
      - Back to guessing!
    - Or compute them all and compare with w[j]

This is what Chapter 5 is all about...

- Increase radix
  - Do more than one bit per iteration, but increases complexity of q digit selection and divisor multiples
- Quotient Digit Set
  - Use redundant (CS) digit set to simplify q digit selection
    - Higher redundancy ρ reduces complexity of q digit selection but increases complexity of divisor multiples

 Representation of residual
- If in redundant (CS) form, can use CS adders at each step
- But increases the q digit selection again...
- Increases number of registers (wc, ws)
- Complicates sign detection for termination step
- q digit selection function
  - Is a BIG piece of the puzzle...
Fractional Division

\[ x = q \cdot d + \text{rem} \]
\[
\text{rem} < |d| \cdot \text{ulp} \quad \text{and} \quad \text{sign}(\text{rem}) = \text{sign}(x)
\]

DIVIDEND ~
DIVISOR \( d \)
QUOTIENT \( q \)
REMAINDER \( \text{rem} \)

- INTEGER QUOTIENT. \( q_{\ell} = 1 \).
- FRACTIONAL QUOTIENT. \( q_{r} = r^{-\ell} \)

TWO TYPES OF DIVISION OPERATION:
1. INTEGER DIVISION, WITH INTEGER OPERANDS AND RESULT
   USUALLY REQUIRES AN EXACT REMAINDER
2. FRACTIONAL DIVISION
   TO AVOID QUOTIENT OVERFLOW. \( x < d \)
   QUOTIENT ROUNDED, WHICH CAN RESULT IN A NEGATIVE REMAINDER.

Fractional Division Assumptions

- OPERANDS/RESULT IN SIGN-AND-MAGNITUDE FORMAT \( \Rightarrow \) CONSIDER MAGNITUDES ONLY

\[
1/2 \leq d < 1: \quad x < d: \quad 0 < q < 1
\]
\[
d = 0.1\ldots
\]

Quotient

\[
q = q[\ell] + \sum_{j=1}^{\ell} q[j] r^{-j}
\]
\[
q = q_\ell q_{\ell-1} q_{\ell-2} q_{\ell-3} \ldots
\]

Basic Division Recurrence

- Based on error bounds (we want to end up with error less than 1 ulp)
  \[
w[j + 1] = rw[j] - dq_{j+1}
  \]
  with \(w[0] = x - dq[0]\)
  \((q[0] \text{ usually } 0\ldots)\)
- The recurrence is performed so \(w[j+1]\) is bounded by \(|w[j]| \leq \rho d\)

Quotient Digit Selection

- Select quotient digits with a selection function (defined later...)
  \[
  q_{j+1} = \text{SEL}(w[j], d)
  \]
- But – using a redundant digit set allows you to be a little wrong – you can fix it on the next step!
- So, you can use truncated versions...
  - Truncated \( w[j] \quad \rightarrow \quad \hat{w}[j] \)
  - Truncated \( d = \hat{d} \)
  - \( q_{j+1} = \text{SEL}(\hat{w}[j], \hat{d}) \)

Division Step

1. ONE DIGIT ARITHMETIC LEFT-SHIFT OF \( w[i] \) TO PRODUCE \( rw[i] \):
2. DETERMINATION OF THE QUOTIENT Digit \( q_{j+1} \)
   BY THE QUOTIENT-DIGIT SELECTION FUNCTION; \( q_{j+1} = \text{SEL}(w[i], d) \)
3. GENERATION OF THE DIVISOR MULTIPLE \( d \times q_{j+1} \) AND
4. SUBTRACTION of \( dq_{j+1} \) from \( rw[i] \):
  \[
w[j + 1] = rw[j] - dq_{j+1}
  \]
5. UPDATE OF THE QUOTIENT \( q[j] \) TO \( q[j + 1] \) BY THE ON-THE-FLY
   CONVERSION
  \[
  q[j + 1] = \text{COPY}(q[j], q_{j+1})
  \]
### Division Hardware

**Initialization Details**

- Needs to satisfy initial value of residual, and assure convergence
  - \( w[0] - x = dq[0] \) and \( |w[0]| \leq pd \). Options:
    - Make \( q[0] = 0 \) and
      - For \( \rho = 1 \) we make \( w[0] = x/2 \).
      - For \( 1/2 < \rho < 1 \) we make \( w[0] = x/4 \)
    - Compensated in the termination step
  - Make \( q[0] = 1 \) and \( w[0] = x - d \). Applicable for \( \rho < 1 \) because \( q > 1 + \rho \) not allowed.

**Iteration Details**

- Radix-2
  - \( N \) iterations for \( n \) bits
  - 1 or 2 extra for scaling bits (\( w[0] = x/2 \) or \( x/4 \))
  - 1 extra for rounding (if needed – i.e. FP)
  - Reduce iterations by factor \( k \) when going from radix \( r \) to \( r^k \)
    - Ex: 53 bit, radix-8, \( \rho = 1 \), \( w[0] = x/2 \)
    - \( N = \left\lfloor \frac{53 + 1 + 1}{3} \right\rfloor = 19 \)

**Termination Details**

- Might get a negative remainder (final residual \( w[N] \))
  - If so, fix it in a termination step
  - Requires sign-detection on final (CS) residual...

\[
q = \begin{cases} 
q[N] & \text{if } w[N] \geq 0 \\
q[N] - 2^{-N} & \text{if } w[N] < 0 
\end{cases}
\]

**Quotient is too big resulting in negative residual, so fix by subtracting oneulp**

---

### Whole Thing

- Initialize
  - Set initial values of \( w[0], q_0 \)
- Perform the iterations of the recurrence
  - \( n \) iterations for radix-2, \( n/2 \) for radix-4 etc.
  - Perhaps a few more to take account of scaling and rounding...
- Termination step
  - If you end up with a negative remainder, you need to adjust things
Termination Details

- If dividend was shifted in initialization, shift the quotient back
- Also, for most floating point implementations, detect the zero-remainder condition
- Determine exact quotient for rounding

Review

- Initialization
  - Fixed shift (1 or 2 positions) of dividend
  - Or computation of $x - d$
- Iterations
  - Execute $N$ cycles of the recurrence
- Termination
  - Sign detection of residual
  - Decrement quotient if final residual is negative
  - Fixed shift to correct for initialization

Example...

- REDUNDANT RESIDUAL \( w[j] = (WC[j], W \cdot [j]) \)

1. [Initialization]
   \( W[j] = 0; WC[j] = 0; q_0 = 0; q_{-1} = 0 \)
2. [Recurrence]
   for \( j = 0 \ldots n + 1 \) (\( n + 2 \) iterations because of initialization and guard bit)
   \( q_{j+1} = \text{SEL}(j); WC[j+1], WC[j+1] = \text{CSADD}(2WC[j], 2W[j]), -q_{j+1};\)
   \( q[j] = \text{CONVERT}(q[j-1], q[j]); \)
   \( w[j+1] = rw[j] - dq_{j+1} \)
   end for
3. [Terminate]
   if \( w[j+2] < 0 \) then \( q = 2 \cdot \text{CONVERT}(q[j+1], q_{j+1} - 1) \)
   else \( q = 2 \cdot \text{CONVERT}(q[j+1], q_{j+1}) \)

Example...

- $n$ is the precision in bits,
- SEL is the quotient-digit selection function:
  \[ q_{j+1} = \text{SEL}(j) = \begin{cases} 
  1 & \text{if } 0 \leq j \leq 3/2 \\
  0 & \text{if } j = -1/2 \\
  -1 & \text{if } -5/2 \leq j \leq -1 
  \end{cases} \]

The estimate \( j \) has four bits (three integer bits and one fractional bit) of the shifted residual in carry-save form,
- CSADD is carry-save addition
- \(-q_{j+1}\) is in 2's complement form, and
- CONVERT on-the-fly quotient conversion function

Example...

\[
\begin{align*}
\text{Dividend } x &= (0.10011111), \text{ divisor } d = (0.11000001), \text{ scaled residual } \\
2w[0] &= 2(x/2) = x, \text{ computed } q/2
\end{align*}
\]

- 4 bits of output (8 bit dividend), \( n=4 \)
- \( N=n+2 \) because of scaling and rounding
- \( w[j+1] = rw[j] - dq_{j+1} \)
- Termination
  - if \( w[j] < 0 \) then \( q = 2(q[j] - 2^4) \)
  - else \( q = 2(q[j]) \)
Example

Dividend \( x = (0.10011111) \), divisor \( d = (0.11000001) \), scaled residual

\[
2n[y] = 2(y/2) = x \mod \text{quotiented} = y/2
\]

\[
\begin{align*}
2V[S][0] & = 000.1001111 \\
2V[C][0] & = 110.0100000 \quad \ast \theta[0] = 0.5 \quad q_0 = 1 \\
\text{J=0} & \quad \text{mod} = 11.0011100 \\
\hline
2V[S][1] & = 000.0110100 \quad \theta[1] = -1 \quad q_1 = -1 \\
2V[S][2] & = 111.0000000 \quad \theta[2] = 0 \quad q_2 = 1 \\
\text{J=1} & \quad \text{mod} = 11.0011100 \\
2V[S][3] & = 000.0110100 \quad \theta[3] = 0 \quad q_3 = 1 \\
2V[S][4] & = 110.0110000 \quad \theta[4] = -1.5 \quad q_4 = -1 \\
2V[S][5] & = 111.1101111 \\
\text{J=2} & \quad \text{mod} = 11.0011100 \\
2V[S][6] & = 000.1001111 \\
\text{J=3} & \quad \text{mod} = 11.0011100 \\
\end{align*}
\]
Example

\[ \begin{align*}
\text{Dividend } x & = 0.62109375, \quad \text{Divisor } d = 0.76953125, \text{ scaled residual } \\
2x[1] = 2x &= x_{\text{corrected}} = x/2 \quad \text{\textbullet Did it work?} \\
.011001 &\text{ is final corrected quotient} \\
\text{Quotient is } 2^0 [a] \text{ because of dividend scaling} \\
\therefore .11001 &= .78125 \\
\text{Rem } = .10100011, \text{ but needs to be scaled to } 2^{-6} \text{ because} \\
\text{that's theulp, then multiplied by 2 for the scaling.} = .0000010100011 = .1989764_{10} \\
\therefore (.78125)(.76953125) + .01989746 &= .62109375 \text{ (yay!)}
\end{align*} \]

On-the-Fly Conversion

\[ \begin{align*}
\therefore \text{The } q \text{-bits use } \{-1,0,1\} \text{ digit set} \\
\therefore \text{The native version of } q \text{ was } .111110 \\
\therefore \text{Would require post-pass conversion...} \\
\therefore \text{On-the-Fly conversion converts } q \text{ into } \{0,1\} \text{ digit by} \\
\therefore \text{digit as the iterations occur}
\end{align*} \]

\[ \begin{align*}
&Q[4] = \begin{cases} 
Q[3] + q_{4}2^{-4} & \text{if } q_{4} \geq 0 \\
Q[3] - 2^{-3} + (r - |q_{4}|)2^{-4} & \text{if } q_{4} < 0
\end{cases} \\
\therefore \text{UPDATE} \\
&Q[j+1] = Q[j] + q_{j+1}r^{-j-1} \\
&Q[j+1] = Q[j] + q_{j+1}r^{-j-1} \\
&\text{\textbullet Since } q_{j+1} \text{ can be negative:} \\
&Q[j+1] = \begin{cases} 
Q[j] + q_{j+1}r^{-j-1} & \text{if } q_{j+1} \geq 0 \\
Q[j] - r^{-j-1} + (r - |q_{j+1}|)r^{-j-1} & \text{if } q_{j+1} < 0
\end{cases} \\
&\text{\textbullet Disadvantage: Subtraction } Q[j] - r^{-j} \\
&\text{requires the propagation of a borrow - slow}
\end{align*} \]

On-the-Fly Conversion

\[ \begin{align*}
&Q[j] = \begin{cases} 
Q[j] + q_{j}2^{-j} & \text{if } q_{j} \geq 0 \\
Q[j] - 2^{-j} + 2^{-j} & \text{if } q_{j} < 0
\end{cases} \\
&\text{\textbullet J MS DIGITS OF CONVERTED QUOTIENT} \\
&Q[j] = \frac{1}{2} q_{j}r^{-j} \\
&\therefore \text{UPDATE} \\
&Q[j+1] = Q[j] + q_{j+1}r^{-j-1} \\
&\text{\textbullet Since } q_{j+1} \text{ can be negative:} \\
&Q[j+1] = \begin{cases} 
Q[j] + q_{j+1}r^{-j-1} & \text{if } q_{j+1} \geq 0 \\
Q[j] - r^{-j-1} + (r - |q_{j+1}|)r^{-j-1} & \text{if } q_{j+1} < 0
\end{cases} \\
&\text{\textbullet Disadvantage: Subtraction } Q[j] - r^{-j} \\
&\text{requires the propagation of a borrow - slow}
\end{align*} \]

On-the-Fly Conversion

\[ \begin{align*}
&Q[j] = \begin{cases} 
Q[j] + q_{j}2^{-j} & \text{if } q_{j} \geq 0 \\
Q[j] - 2^{-j} + 2^{-j} & \text{if } q_{j} < 0
\end{cases} \\
&\text{\textbullet J MS DIGITS OF CONVERTED QUOTIENT} \\
&Q[j] = \frac{1}{2} q_{j}r^{-j} \\
&\therefore \text{UPDATE} \\
&Q[j+1] = Q[j] + q_{j+1}r^{-j-1} \\
&\text{\textbullet Since } q_{j+1} \text{ can be negative:} \\
&Q[j+1] = \begin{cases} 
Q[j] + q_{j+1}r^{-j-1} & \text{if } q_{j+1} \geq 0 \\
Q[j] - r^{-j-1} + (r - |q_{j+1}|)r^{-j-1} & \text{if } q_{j+1} < 0
\end{cases} \\
&\text{\textbullet Disadvantage: Subtraction } Q[j] - r^{-j} \\
&\text{requires the propagation of a borrow - slow}
\end{align*} \]

On-the-Fly Conversion

\[ \begin{align*}
&Q[j] = \begin{cases} 
Q[j] + q_{j}2^{-j} & \text{if } q_{j} \geq 0 \\
Q[j] - 2^{-j} + 2^{-j} & \text{if } q_{j} < 0
\end{cases} \\
&\text{\textbullet J MS DIGITS OF CONVERTED QUOTIENT} \\
&Q[j] = \frac{1}{2} q_{j}r^{-j} \\
&\therefore \text{UPDATE} \\
&Q[j+1] = Q[j] + q_{j+1}r^{-j-1} \\
&\text{\textbullet Since } q_{j+1} \text{ can be negative:} \\
&Q[j+1] = \begin{cases} 
Q[j] + q_{j+1}r^{-j-1} & \text{if } q_{j+1} \geq 0 \\
Q[j] - r^{-j-1} + (r - |q_{j+1}|)r^{-j-1} & \text{if } q_{j+1} < 0
\end{cases} \\
&\text{\textbullet Disadvantage: Subtraction } Q[j] - r^{-j} \\
&\text{requires the propagation of a borrow - slow}
\end{align*} \]
On-the-Fly Conversion

Now we need to keep $Q_M[j]$ updated
- Without propagating a carry!

$$Q_M[j+1] = Q[j+1] - r^{-(j+1)}$$

$$Q_M[4] = Q[4]-2^4 = q_2^4 + q_3 2^3 + q_4 2^2 - 2^4$$

- What happens if $q=-1$ or $1$?


On-the-Fly Conversion

Now all additions are concatenations!
- Take $j$ bits, concatenate one more bit

$$Q[j+1] = \begin{cases} (Q[j], q_{j+1}) & \text{if } q_{j+1} \geq 0 \\ (Q_M[j], (r-|q_{j+1}|)) & \text{if } q_{j+1} < 0 \end{cases}$$

$$Q_M[j+1] = \begin{cases} (Q[j], q_{j+1} - 1) & \text{if } q_{j+1} \geq 0 \\ (Q_M[j], ((r-1) - |q_{j+1}|)) & \text{if } q_{j+1} < 0 \end{cases}$$

INITIAL CONDITIONS $Q[0] = Q_M[0] = 0$ (for a positive quotient)

Conversion Example

<table>
<thead>
<tr>
<th>$j$</th>
<th>$q_j$</th>
<th>$Q[j]$</th>
<th>$Q_M[j]$</th>
<th>$Q[j+1]$</th>
<th>$Q_M[j+1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.101</td>
<td>0.101</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.101</td>
<td>0.101</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.1001</td>
<td>0.1001</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.1000</td>
<td>0.1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Conversion Implementation

$$Q_M[j+1] = \begin{cases} (Q[j], q_{j+1}) & \text{if } q_{j+1} \geq 0 \\ (Q_M[j], (r-|q_{j+1}|)) & \text{if } q_{j+1} < 0 \end{cases}$$

$$Q_M[j+1] = \begin{cases} (Q[j], q_{j+1} - 1) & \text{if } q_{j+1} \geq 0 \\ (Q_M[j], ((r-1) - |q_{j+1}|)) & \text{if } q_{j+1} < 0 \end{cases}$$
Example from previous example

\[ \begin{array}{cccc}
 j & y & q(j) & QM(j) \\
 0 & 0.1 & 0.1 & QM(0) = 0.1 \\
 1 & 1.0 & 0 & QM(1) = 0.0 \\
 2 & 0.1 & 0.0 & QM(2) = 0.0 \\
 3 & 1 & 0 & QM(3) = 0.0 \\
 4 & 0.1 & 0 & QM(4) = 0.0 \\
 5 & 1 & 0 & QM(5) = 0.0 \\
 \end{array} \]

Converted Quotient

\[ Q(j) = \begin{cases} \frac{Q(j)}{QM(j)} & \text{if } q(j) = 0 \\ \frac{Q(j)}{QM(j)} & \text{if } q(j) = -1 \\ \frac{Q(j)}{QM(j)} - \frac{1}{Q(j)} & \text{if } q(j) = 1 \\ \frac{Q(j)}{QM(j)} - \frac{1}{Q(j)} & \text{if } q(j) = -2 \\ \frac{Q(j)}{QM(j)} - \frac{1}{Q(j)} & \text{if } q(j) = 0 \\ \end{cases} \]

Implementation

Example (repeated)

\[ J = 0.5 \]

\[ \begin{align*}
 J = 3 & \quad W[4] = 0.110000010 \\
 & \quad Z[4] = 1.101100100001, \quad q_4 = -1.5, \quad q_5 = 0 \\
 & \quad q_6 = 0.000000000 \\
 W[5] = 1.101100111 \quad W[5] = 0.001000000 \\
 \end{align*} \]

Converted Quotient

\[ q = 0.0110101010 \\
 \text{rem} = 0.000000000 \\
 \text{0.01100011 final quotient} \]

Additional Complications

- **Termination**
  - Require sign of remainder and zero remainder for termination step
  - Could convert to conventional notation
  - Requires CPA
  - Or can try to extract it directly from CS notation
  - The book has some tricks

Example

- **Was it worth it?**
  - Faster iterations because of CS adders
  - More complex q digit selection logic
  - But were able to use truncated w[j] through some magic
  - Extra work in termination stage for final restore
Going Further...

- Can we add higher-radix techniques?
  - Reduce number of iterations
  - Complicates q-digit selection
  - Other stuff is relatively similar...

Radix-4 Division

1. QUOTIENT DIGIT SET (-2, 1, 0, 1, 2) \( r = 4, \rho = 2/3 \)
2. \( \rho < 1 \) initialize \( W[S][i] = \rho i \)
3. THE NEXT RESIDUAL
   \( W[S][j+1] = CSADD(x^3 W[S][j], \rho i W[S][j], -q_{j,i} d) \)
4. QUOTIENT DIGIT SELECTION DEPENDS ON ESTIMATES OF SHIFTED RESIDUAL AND DIVISOR described in terms of SELECTION CONSTANTS \( m_k(i) \)
   \[ q_{j+1} = k \text{ if } m_k(i) \leq y < m_{k+1}(i) \]
5. FINAL QUOTIENT = \( 4 \times \) OBTAINED QUOTIENT

Quotient Digit Selection \( (r = 4, \rho = 2/3) \)

- \( q_{j+1} = k \text{ if } m_k(i) \leq y < m_{k+1}(i) \)
- \( i = 16 \) and divisor truncated to the 4th fractional bit and \( y \) is in \( [y] \) truncated to the 4th fractional bit.

### Example

**Dividend** \( x = 0.1(\text{bin}) = 0.1100(\text{dec}) \)
**Divisor** \( d = 0.1(\text{bin}) = 0.0101(\text{dec}) \)

\[ n = \frac{4 + 1}{2} = 3 \text{ iterations} \]

\[ q = 1 \]

\[ W[S][j+i] = \rho i W[S][j] \]

\[ \begin{array}{c|cccc}
   i & 0 & 1 & 2 & 3 \\
   \hline
   m_0(i) & 0 & 4 & 4 & 4 \\
   m_1(i) & 1 & 4 & 4 & 4 \\
   m_2(i) & 2 & 4 & 4 & 4 \\
   m_3(i) & 3 & 4 & 4 & 4 \\
   m_4(i) & 4 & 4 & 4 & 4 \\
   m_5(i) & 5 & 4 & 4 & 4 \\
   m_6(i) & 6 & 4 & 4 & 4 \\
   m_7(i) & 7 & 4 & 4 & 4 \\
   m_8(i) & 8 & 4 & 4 & 4 \\
   m_9(i) & 9 & 4 & 4 & 4 \\
   m_{10}(i) & 10 & 4 & 4 & 4 \\
   m_{11}(i) & 11 & 4 & 4 & 4 \\
   m_{12}(i) & 12 & 4 & 4 & 4 \\
   m_{13}(i) & 13 & 4 & 4 & 4 \\
   m_{14}(i) & 14 & 4 & 4 & 4 \\
   m_{15}(i) & 15 & 4 & 4 & 4 \\
   \end{array} \]

\[ q_j = 1 \text{ real value } \text{ shown value/16} \]

\[ \sum_{i=0}^{3} q_i \times 2^{-i} = 0.1011 \times 2^{-1} = 0.1011 \]

\[ \begin{array}{c|cccc}
   i & 0 & 1 & 2 & 3 \\
   \hline
   x = \sum_{i=0}^{3} q_i \times 2^{-i} = 0.1011 \times 2^{-1} = 0.1011 \\
   \end{array} \]
Example

\[ q_{j+1} = k \text{ if } m_j(i) \leq y < m_{k+1}(i) \]

On the Fly Conversion

Example

\[ q_{j+1} = k \text{ if } m_j(i) \leq y < m_{k+1}(i) \]

Did it Work?

\[ q_0 = 0.001110 \]
- multiply by 4 because of init to \( x/4 \)
- \( q = 0.1110 = 0.32_{10} = 0.875_{10} \)
- \( r = 0.00101010 \)
- \( x = 0.875_{10} \times 4 \) (init) = 0.00000010101010 \( x = 0.0125390625_{10} \)

\[ 0.68359375 \quad 0.76953125 \]
Dividend \( x = 0.10101111_{2}, \) divisor \( d = 0.11000000_{2} \) (i.e. 160(0.1000)_2 = 12)

scaled residual \( 4x = [x/4] = x \times \text{sign}(x) = q/4 \)

\[ q = 0.68359375 \times 4 = 2.734375_{10} \]
\[ 4x/4 = 0.000000010101010 \]
\[ q = 0.68359375 \]
\[ \text{Did it Work?} \]

\[ q_0 = 0.001110 \]
- multiply by 4 because of init to \( x/4 \)
- \( q = 0.1110 = 0.32_{10} = 0.875_{10} \)
- \( r = 0.00101010 \)
- \( x = 0.875_{10} \times 4 \) (init) = 0.00000010101010 \( x = 0.0125390625_{10} \)

\[ 0.68359375 \quad 0.76953125 \]
Dividend \( x = 0.10101111_{2}, \) divisor \( d = 0.11000000_{2} \) (i.e. 160(0.1000)_2 = 12)

scaled residual \( 4x = [x/4] = x \times \text{sign}(x) = q/4 \)

\[ q = 0.68359375 \times 4 = 2.734375_{10} \]
\[ 4x/4 = 0.000000010101010 \]
\[ q = 0.68359375 \]
\[ \text{Did it Work?} \]
Hardware – Radix-4 Division

Performance Comparison

<table>
<thead>
<tr>
<th>element</th>
<th>delay</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>q-digit selection</td>
<td>6.8</td>
<td>56</td>
</tr>
<tr>
<td>buffers</td>
<td>1.8</td>
<td>5</td>
</tr>
<tr>
<td>MUX</td>
<td>1.4</td>
<td>160</td>
</tr>
<tr>
<td>CSA</td>
<td>1.0</td>
<td>260</td>
</tr>
<tr>
<td>registers (3)</td>
<td>4.0</td>
<td>600</td>
</tr>
<tr>
<td>Convert &amp; Round</td>
<td>1.3</td>
<td>1300</td>
</tr>
</tbody>
</table>

Radix-2 Stage
Radix-4 Stage

(TC means delay not on critical path…)

Total Performance Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>r2</th>
<th>r4</th>
<th>r8</th>
<th>r16 (overlapped)</th>
<th>r312</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle-time factor</td>
<td>1.0</td>
<td>1.3</td>
<td>1.6</td>
<td>1.6</td>
<td>1.7</td>
</tr>
<tr>
<td>Number of cycles</td>
<td>57</td>
<td>20</td>
<td>20</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.0</td>
<td>1.5</td>
<td>1.8</td>
<td>2.4</td>
<td>3.4</td>
</tr>
<tr>
<td>Area factor</td>
<td>1.0</td>
<td>1.1</td>
<td>1.5</td>
<td>1.7</td>
<td>3.9</td>
</tr>
</tbody>
</table>

Includes times/area for schemes talked about in the book that are not presented in these slides… (53-bit quotient)

Radix-4 looks like a sweet spot in terms of speedup/area

Integer Division

- Can use the same fractional division hardware
- Normalize divisor
- Compute a few more bits to satisfy residual bound
- Align quotient with the integer position
- Basically shift a few things and keep track of what you did so that you can shift back to get the right result…

Quotient Digit Selection

Very Tricky Stuff!!
(just ask Intel…)

Quotient digit set:

\[ q_{k+1} \in D_k = \{ -a, -a+1, \ldots, -1, 0, 1, \ldots, a-1, a \} \]

Redundancy factor:

\[ \mu = \frac{a}{r-1} \quad \frac{1}{2} \leq \mu \leq 1 \]

- Two fundamental conditions for q selection
- Containment – must guarantee bounded residual
- Continuity – there must exist a valid choice of \( q_{k+1} \) in the range of shifted residual

Quotient Digit Selection

- Residual recurrence

\[ w[j+1] + r = w[j] - dq_{k+1}, \quad |w[j]| \leq rd \]

\[ \rho = a/(r-1) \quad -a \leq q_{k} \leq a \]

- Selection intervals
  - If \( rw[j] \in [U_k, U_k] \) then \( q_{k+1} = k \) makes \( w[j+1] \) bounded

\[ L_k \leq rw[j] \leq U_k \Rightarrow \quad rd \leq w[j+1] = rw[j] - k \cdot \rho \leq rd \]

- Expressions for selection intervals

\[ U_k = (k+\rho)d \quad L_k = (k-\rho)d \]

\( k \) is the quotient digit you’re selecting
Robertson’s Diagram


Vertical axis is the bound on \( w[j+1] \) that keeps that keeps things in range

\[ w[j+1] - rw[j] - q_{j+1} |rw[j]| \leq \rho \]

\[ \rho = \alpha / (p - 1) \]

Horiz axis is the size of the current \( rw[j] \) which tells you how large \( q \) should be to keep the next \( w[j+1] \) in bound

Continuity and Overlap

- \( q_{j+1} = \text{SEL}(w[j], d) \)
- \( \text{SEL} \) represented by the set \( \{q_k\}, \alpha \leq k \leq a, \)
- \( q_{j+1} = k \) if \( q_k \leq rw[j] \leq q_{k+\rho} \)
- \( q_k \)'s defined as the minimum value of \( rw[j] \) for which \( q_{j+1} = k \)
- CONTINUITY: \( L_k \leq q_k \leq U_k \)
- CONTINUITY: \( q_{j+1} = k - 1 \) for \( rw[j] = q_k - \rho d \leq U_k - 1 \)
- OVERLAP \( L_{j+1} - L_j = (k - 1 + \rho)d - (k - \rho)d = (2\rho - 1)d \)

RESULTING IN

\[ \rho \geq 2^{-1} \]

- REDUNDANCY IN DEC DIGIT SET — OVERLAP BETWEEN SELECTION INTERVALS - SIMPLER SELECTION

Overlap – Robertson Diagram

Overlap – P-D Diagram

Selection Using Constants

- USE CONSTANTS \( M_0 \), INDEPENDENT OF DIVISOR

\[ \max(L_k) \leq M_0 \leq \min(U_{k-1}) + \rho d \]

max and min for the range \( 2^{-1} \leq d < 1 \)

- For \( k > 0 \)

\[ (k - \rho) \leq M_0 \leq (k - 1 + \rho)2^{-1} + \rho d \]

which requires \( \rho \geq \frac{k + 1}{3} \)

- For \( k \leq 0 \)

\[ (k - \rho)2^{-1} \leq k + 1 + \rho \]

which requires \( \rho \geq \frac{(-k) + 2}{3} \)
Radix-2 Example

- \( r = 2 \), \( q \) digits = \{-1,0,1\}, \( \rho = 1 \)

- \( w[j+1] = rw[j] - d_jq_j \), \( |w[j]| \leq \rho d \)

- \( w[j] \) ranges from \(-d\) to \(d\)

- \( \rho = \alpha(r-1), -\alpha \leq q_j \leq \alpha \)

- \( rw[j] \in [L_k, U_k] \)

- \( U_k = (k+\rho)d \), \( L_k = (k-\rho)d \)

Larger Radices

Staircase Selection Function

- For \( r > 2 \), \( m_d \) depends on divisor

- Divide range of divisor into intervals \([d_i, d_{i+1}]\) with

  \[ d_0 = \frac{1}{2}, \quad d_{i+1} = d_i + 2^{-\delta} \]

- MS fractional bits of divisor represent the interval

- For each interval, there is a set of selection constants \( m_d(i) \)

  for \( d \in [d_i, d_{i+1}) \), \( q_{i+1} = k \) if \( m_d(i) \leq rw[j] \leq m_{d+1}(i) - wp \)

Range for Selection Constants

\[ \max(U_d(d_i), U_d(d_{i+1})) \leq m_d(i) \leq \min(U_{d+1}(d_i), U_{d+1}(d_{i+1})) + wp \]
Reduced Precision Selection Const.

\[ m_j(i) = A_j(i)2^{-k} \]

WHERE \( A_j(i) \) IS INTEGER

- CAN USE TRUNCATED RESIDUAL IN COMPARISONS WITH SELECTION CONSTANTS
- RESIDUAL MUST BE IN 2's COMPLEMENT
- SELECTION CONDITIONS

\[
\begin{align*}
\text{for } k > 0 & \quad L_0(d_k + 2^{-k}) \leq A_0(i)2^{-k} \leq U_0(d_k) \\
\text{for } k \leq 0 & \quad L_0(d_k) \leq A_0(i)2^{-k} \leq U_0(d_k + 2^{-k}) + wip
\end{align*}
\]

(5.1)

Example: Radix-4

Non-Redundant residual...

- Known as Robertson's division
- \( g_j \in \{-2, -1, 0, 1, 2\} \)
- \( L_k = \{ \frac{1}{2} + k \} d \)
- \( U_k = \{ -\frac{1}{2} + k \} d \)

BOUND ON \( \delta \)

\[ 2^{-\delta} < \frac{2p - 1}{2(n - p)} = \frac{1}{8} \]

= 3 fractional bits

\[ \begin{align*}
\text{Bound on } r_{w[j]} & = 4 r_{\text{est}} - n_{\text{est}} d \\
\text{Bound on } r_{w[j]+1} & = 4 r_{\text{est}} - 2d \\
\text{Bound on } r_{\text{est}} & = \frac{2d}{3}
\end{align*} \]
Example: Radix-4

Finding the Quotient Ranges

- To find the quotient digit range for $q$, substitute $-2\alpha d$ and $+2\alpha d$ for $r_{i-1}$ in the remainder recurrence relation.

- Example, find the range for $q = +2$.
  
  
  $-2\alpha d = 4r_i - 2d \Rightarrow 4r_{(\min)} = 4/3d$
  
  $+2\alpha d = 4r_i - 2d \Rightarrow 4r_{(\max)} = 8/3d$

  Hence, for $4/3d \leq 4r_i \leq 8/3d$, use $q = +2$.

- Find the range for $q = +1$.
  
  
  $-2\alpha d = 4r_i - d \Rightarrow 4r_{(\min)} = 1/3d$
  
  $+2\alpha d = 4r_i - d \Rightarrow 4r_{(\max)} = 5/3d$

Quotient Ranges

Overlap Regions

P-D Plot – Overlap regions

Select either $q_i = -2$ or $q_i = -1$
Select either $q_i = +2$ or $q_i = +1$
Select either $q_i = -1$ or $q_i = 0$
Select either $q_i = +1$ or $q_i = 0$
**How Many Bits Needed?**

- **Uncertainty Regions**
  - Region of uncertainty if df are 0.3 and 4. as 0.11
  - Other regions

- **Radix-4 Q-Digit selection Table**
  - Note: need analogous entries for negative partial remainders (i.e., where the mod of $a_i = 1$).
Radix-4 Q-Digit selection Table

<table>
<thead>
<tr>
<th>$d$</th>
<th>Range of $D$</th>
<th>$G$</th>
<th>Range of $G$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3 - 12</td>
<td>0</td>
<td>0 - 12</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 - 12</td>
<td>1</td>
<td>0 - 12</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0 - 12</td>
<td>2</td>
<td>0 - 12</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0 - 12</td>
<td>3</td>
<td>0 - 12</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0 - 12</td>
<td>4</td>
<td>0 - 12</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0 - 12</td>
<td>5</td>
<td>0 - 12</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0 - 12</td>
<td>6</td>
<td>0 - 12</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0 - 12</td>
<td>7</td>
<td>0 - 12</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>0 - 12</td>
<td>8</td>
<td>0 - 12</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0 - 12</td>
<td>9</td>
<td>0 - 12</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>0 - 12</td>
<td>10</td>
<td>0 - 12</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>0 - 12</td>
<td>11</td>
<td>0 - 12</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>0 - 12</td>
<td>12</td>
<td>0 - 12</td>
<td>12</td>
</tr>
</tbody>
</table>

What About Carry-Save Residual?

- SO FAR, COMPUTE $r_u(j)$ IN FULL PRECISION, TRUNCATE, AND COMPARISON WITH LOW-PRECISION CONST.
- FULL-PRECISION ADDITION: significant portion of the cycle time
- OVERLAP BETWEEN SELECTION INTERVALS
- COULD USE AN ESTIMATE OF $r_u(j)$

ERROR IN ESTIMATE:

$$\epsilon_{\text{max}} \leq y - \bar{y} \leq \epsilon_{\text{max}}$$

BASIC CONSTRAINT: if we choose $g_{j+1} = k$ for an estimate $\bar{y}$ then

$$\bar{y} \in [\bar{y} + \epsilon_{\text{max}}, \bar{y} + \epsilon_{\text{max}}]$$

$$\bar{L}_0 = \bar{L}_k - \epsilon_{\text{max}}$$

$$\bar{U}_0 = \bar{U}_k - \epsilon_{\text{max}}$$

Constraints on Selection Const.

$$\max(L_0(d_k), L_2(d_{k+1})) \leq m_{\text{const}} \leq \min(U_0(d_k), U_2(d_{k+1}))$$
### Pentium fdiv Bug

#### Pentium from 1994
- SRT divider in FP unit (53 bits?)
- First time Intel tried SRT...
- Radix-4 SRT with carry-save residuals
- Q-bit lookup based on 7-bits of scaled residual and 4-bits of divisor
- Implemented as PLA
- Five entries incorrect
  - Should have been +2, but because the entries were left out of the PLA, returned 0 in those cases
Pentium fdiv bug

Conclusions

- Division is a pain!
- Luckily, you don’t do it very often…
- But, you still have to divide occasionally
- Naturally iterative
- Reduce iteration time with carry-save residuals
- Reduce iterations with higher-radix
- Means all sorts of complications in picking quotient digits
- Don’t forget about init and terminate steps too!