

Lecture 13

Performance Improvements at Register Level - Hardware algorithms for fixed point
Division

Quotient

Division $\overline{) \text{ Dividend}}$

Remainder

Example

Divide 7 by 2.

$7 \div 2$

$$\begin{array}{r} 3 \\ 2 \overline{) 7} \\ \underline{6} \\ 1 \end{array}$$

result

quotient 3

remainder 1

Example

Binary division

Divide 0111 by 0010

$0111 \div 0010$

$$\begin{array}{r} 0011 \\ 10 \overline{) 0111} \\ \underline{10} \\ 11 \\ \underline{10} \\ 1 \end{array}$$

Result

Quotient 0011

Remainder 1

Will write down what we did

Subtract divisor from the dividend

Shift

Now Will take two large numbers - dividend 8 bits and divisor 4 bits

Example

Divide 1001010 by 1000

$$\begin{array}{r}
 1001 \\
 1000 \overline{) 1001010} \\
 \underline{-1000} \\
 10 \\
 101 \\
 1010 \\
 \underline{-1000} \\
 10 \\
 0
 \end{array}$$

Result

Quotient 1001

Remainder 10

write down what We did

Subtracted the divisor from the dividend

If the remainder is less than zero

We put quotient bit as zero

Restored the remainder

Take one more bit from the dividend

Subtract the divisor from the dividend

If the remainder is greater than zero

We put quotient bit as one

Take one more bit from the dividend

Keep doing that for all the bits in dividend

Hardware units needed to perform this computation

Need

a place to store the divisor

a place to store the dividend

Remainder will remain in the dividend register

a place to store the quotient

a adder/subtractor perform subtraction

Specifically we need

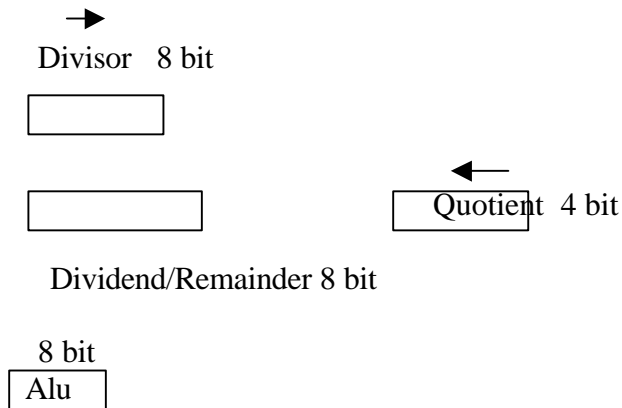
4 bit register for quotient

8 bit register for dividend

8 bit register for divisor

left half of the divisor register will have divisor 4bits

an adder/subtractor and control unit



Control unit

When we multiply 4-bit number by 4-bit number we get an 8-bit product

Similarly, when we divide a 8-bit number by 4-bit number we get a 4-bit number as quotient

Now let us do the division by using this hardware

Example:

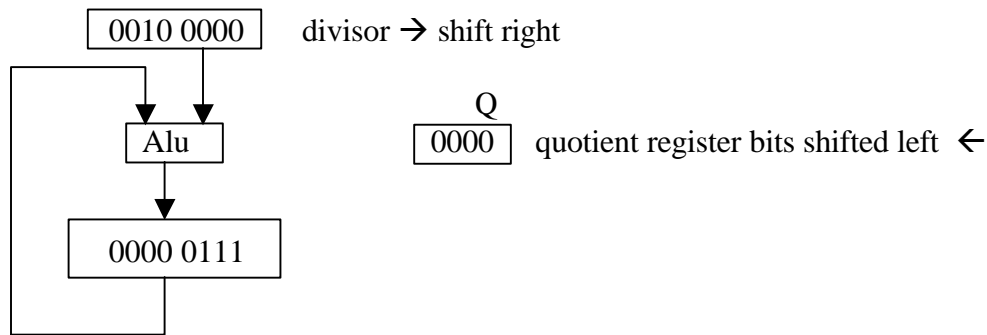
4-bit divisor

$7 \div 2$

$0111 \div 0010$

1st step

initialize registers



control

Control signals to divisor, quotient and ALU

Steps to follow

Rem = Rem - Divisor

If Rem < 0

(1) Shift Q left, $Q_0 = 0$

(2) Restore Remainder

(3) Shift Divisor Right

if Rem > 0

(1) Shift Q left, $Q_0 = 1$

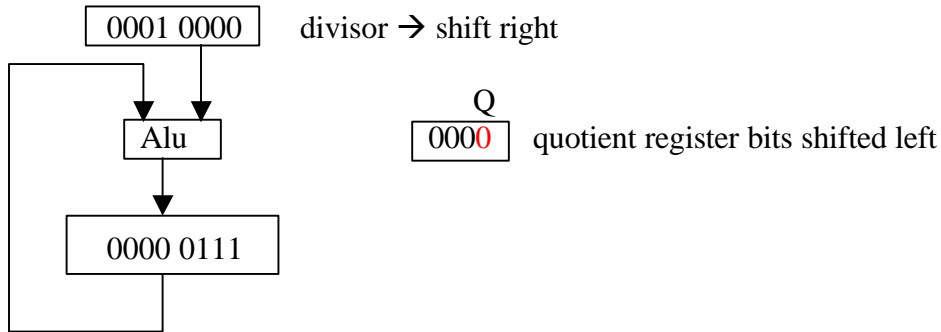
(2) Shift Divisor Right

Step 1

Rem = Rem - Divisor (this time divisor is larger so result is negative)

Rem < 0

- (1) Shift Q left, $Q_0 = 0$
- (2) Restore Remainder
- (3) Shift Divisor Right



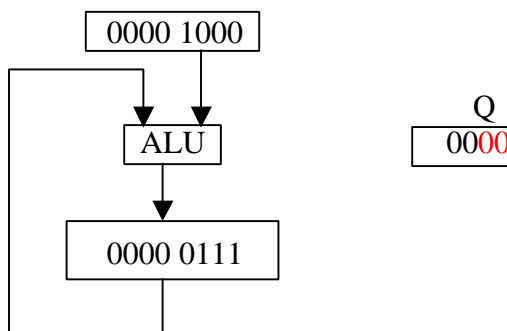
control

2nd step

Rem = Rem - Divisor (divisor is larger so result is negative)

Rem < 0

- (1) Shift Q left, $Q_0 = 0$
- (2) Restore Remainder
- (3) Shift Divisor Right

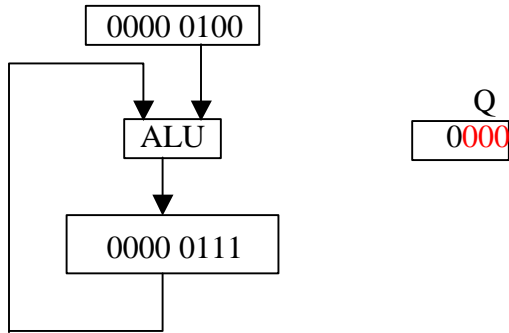


3rd step

Rem = Rem - Divisor (this time divisor is larger so result is negative)

Rem < 0

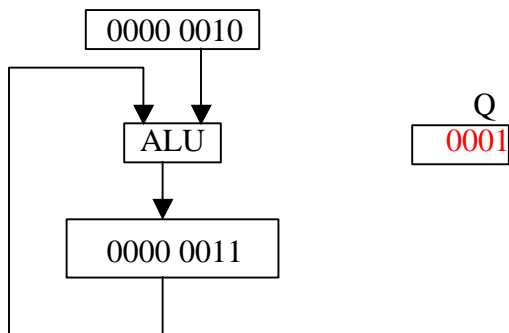
- (1) Shift Q left, $Q_0 = 0$
- (2) Restore Remainder
- (3) Shift Divisor Right

4th step

Rem = Rem - Divisor

Now Rem > 0

- (1) Shift Q left, $Q_0 = 1$
- (2) Shift Divisor Right



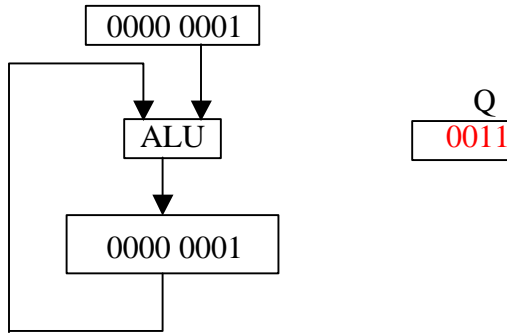
5th step

Rem = Rem - Divisor

Now Rem > 0

(1) Shift Q left, $Q_0 = 1$

(2) Shift Divisor Right



4-bit divisor needs 5 iterations

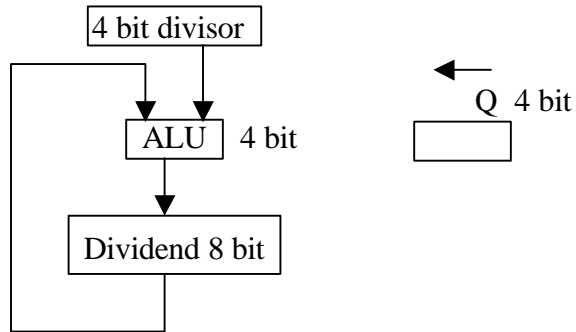
8-bit divisor needs 9 iterations

32-bit divisor needs 33 iterations

2nd version

Reduce the hardware costs in the implementation 1
 8-bit divisor register → 4 bit divisor register
 8-bit adder → 4 bit adder

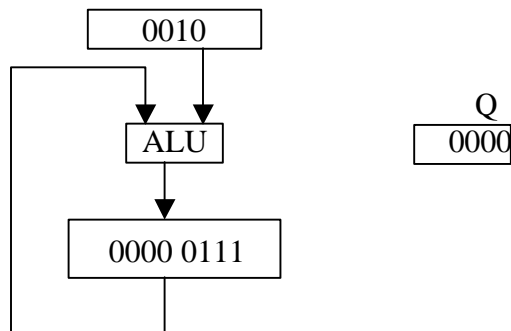
Shift dividend to left
 No shift in divisor



Shift remainder/dividend ←
 No shift in Divisor

1st step

Initialize the registers



2nd step

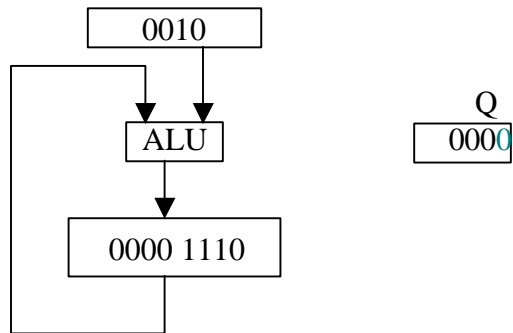
Shift remainder left

Rem = rem-div

Rem < 0

Restore the remainder

Shift left Q, Q0 = 0

3rd step

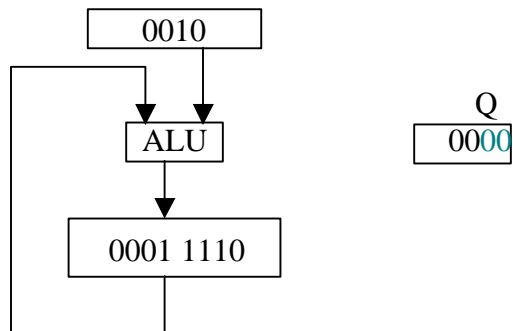
Shift remainder left

Rem = rem-div

Rem < 0

Restore the remainder

Shift left Q, Q0 = 0



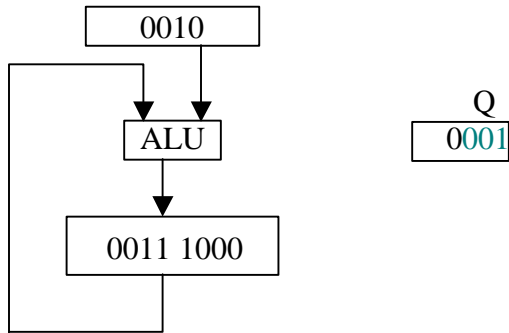
4th step

Shift remainder left

Rem = rem-div

Rem ≥ 0

Shift left Q, Q0 = 1



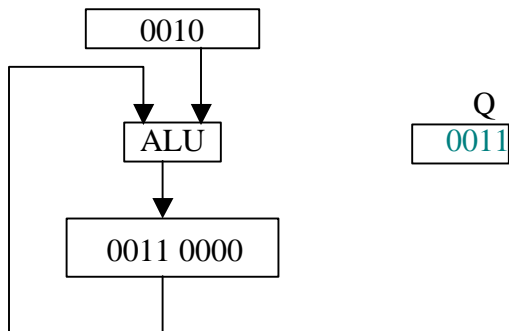
5th step

Shift remainder left

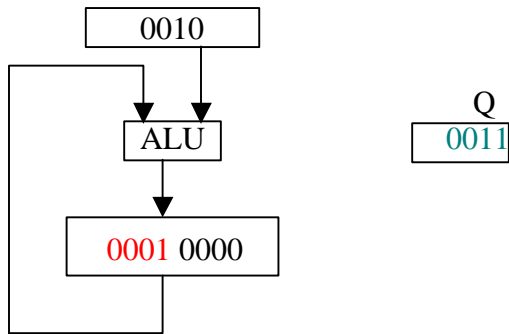
Rem = rem-div

Rem ≥ 0

Shift left Q, Q0 = 1



Final snapshot



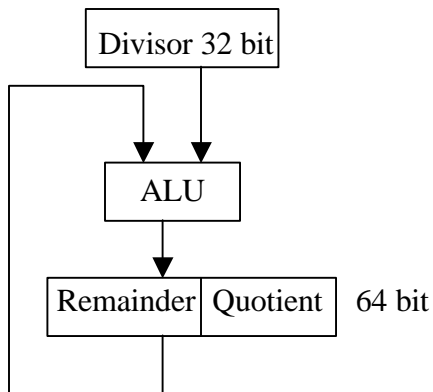
3rd version of Division algorithm

Eliminate the quotient register

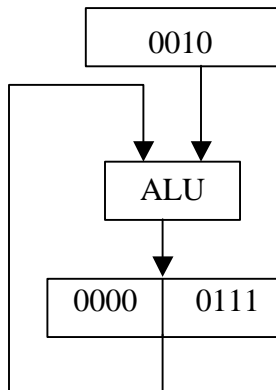
Put the quotient bit to the right most bit position

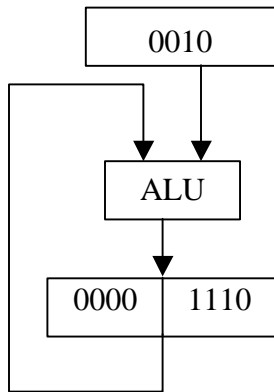
Shift the dividend register one bit left in each iteration

New hardware organization - Data path and control path



Initialize



Shift remainder left

Step 1

Rem = Rem - Div

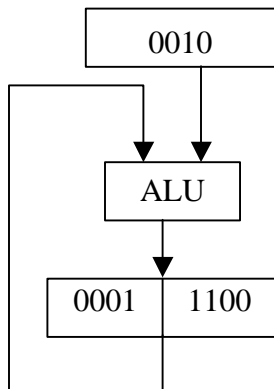
Rem < 0

Restore

Shift left Remainder register

$R_0 = 0$

Step 1

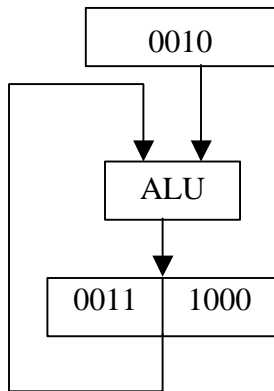


Step 2

Rem = Rem - Div

Rem > 0

Shift left Remainder register

 $Q_0 = 1$ **new content**

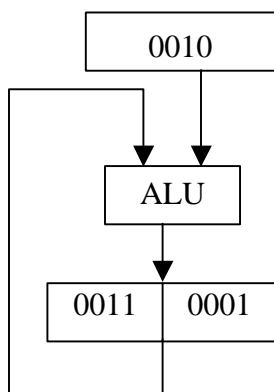
0001 1001 of remainder register

Step 3

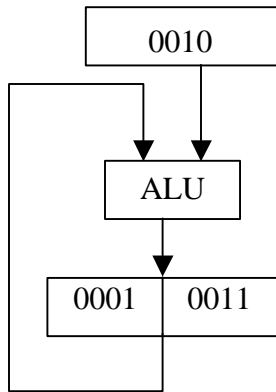
Rem = Rem - Div

Rem > 0

Shift left Remainder register

 $Q_0 = 1$ 

Step 4



0011 - quotient

0001 - remainder