

Multiplication Example

Multiplicand

Multiplier

$$\begin{array}{r} 1000_{\text{ten}} \\ \times 1001_{\text{ten}} \\ \hline \end{array}$$

$$\begin{array}{r} 1000 \\ 0000 \\ 0000 \\ 1000 \\ \hline \end{array}$$

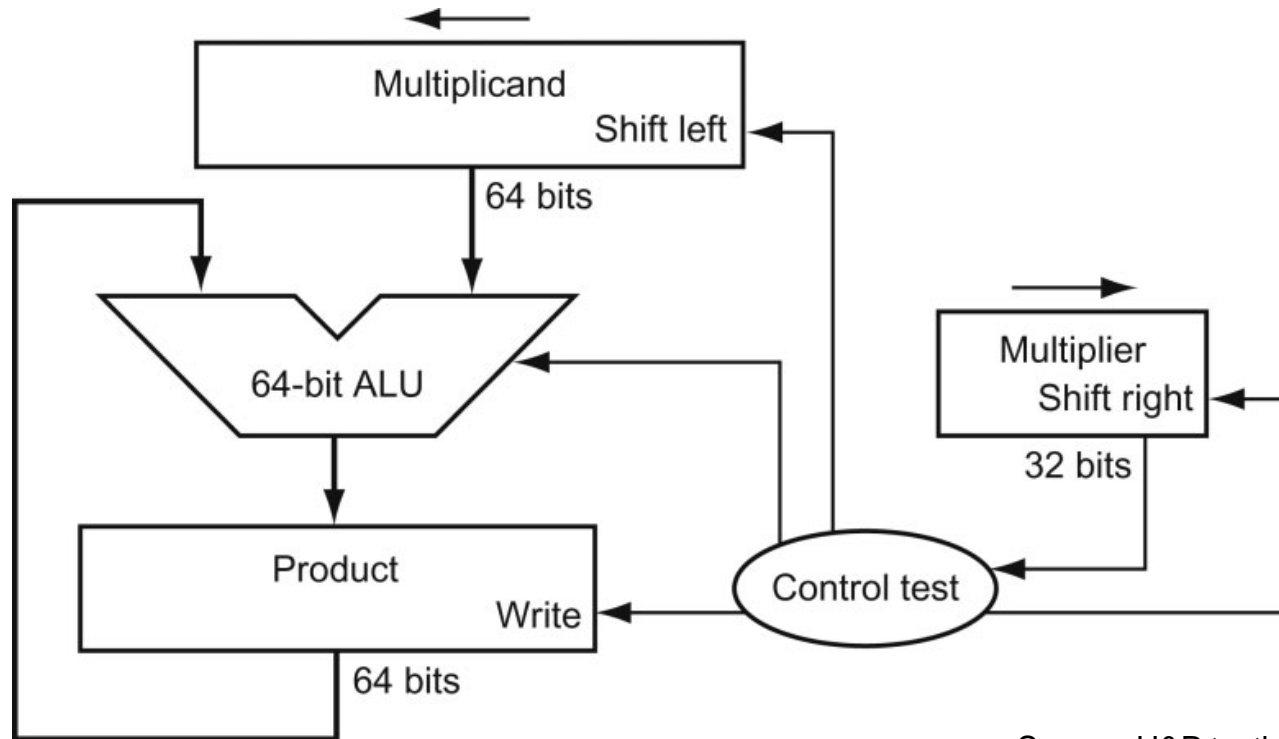
Product

$$1001000_{\text{ten}}$$

In every step

- multiplicand is shifted
- next bit of multiplier is examined (also a shifting step)
- if this bit is 1, shifted multiplicand is added to the product

HW Algorithm 1

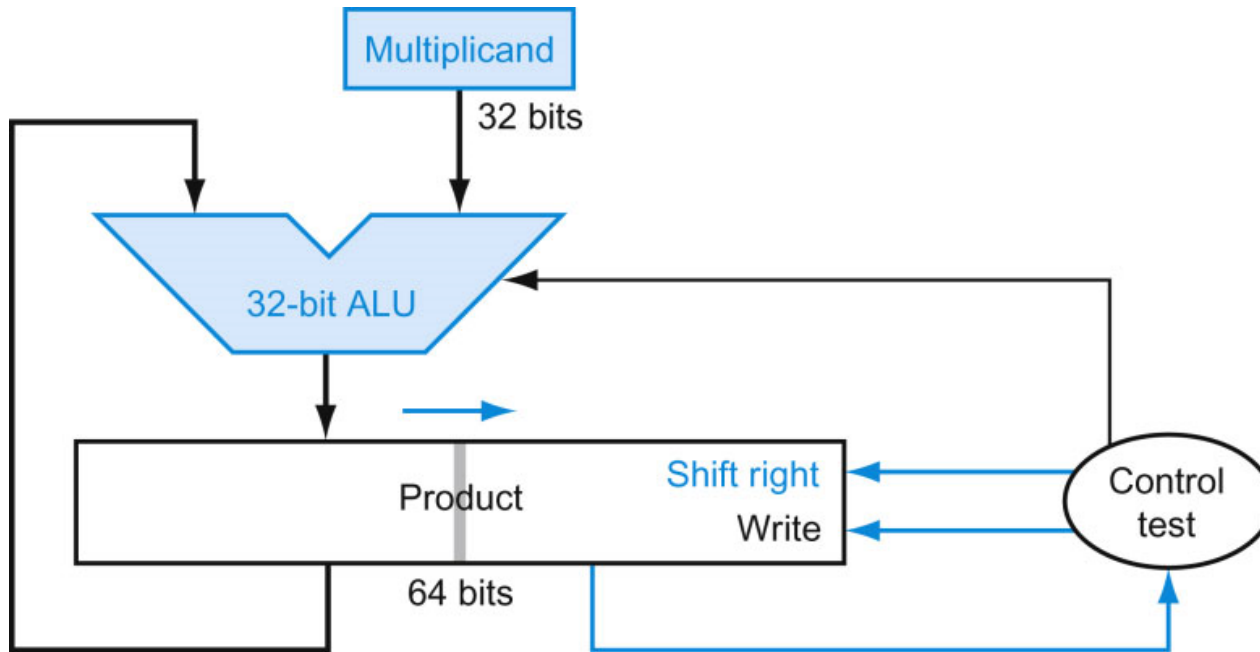


Source: H&P textbook

In every step

- multiplicand is shifted
- next bit of multiplier is examined (also a shifting step)
- if this bit is 1, shifted multiplicand is added to the product

HW Algorithm 2



Source: H&P textbook

- 32-bit ALU and multiplicand is untouched
- the sum keeps shifting right
- at every step, number of bits in product + multiplier = 64, hence, they share a single 64-bit register

Notes

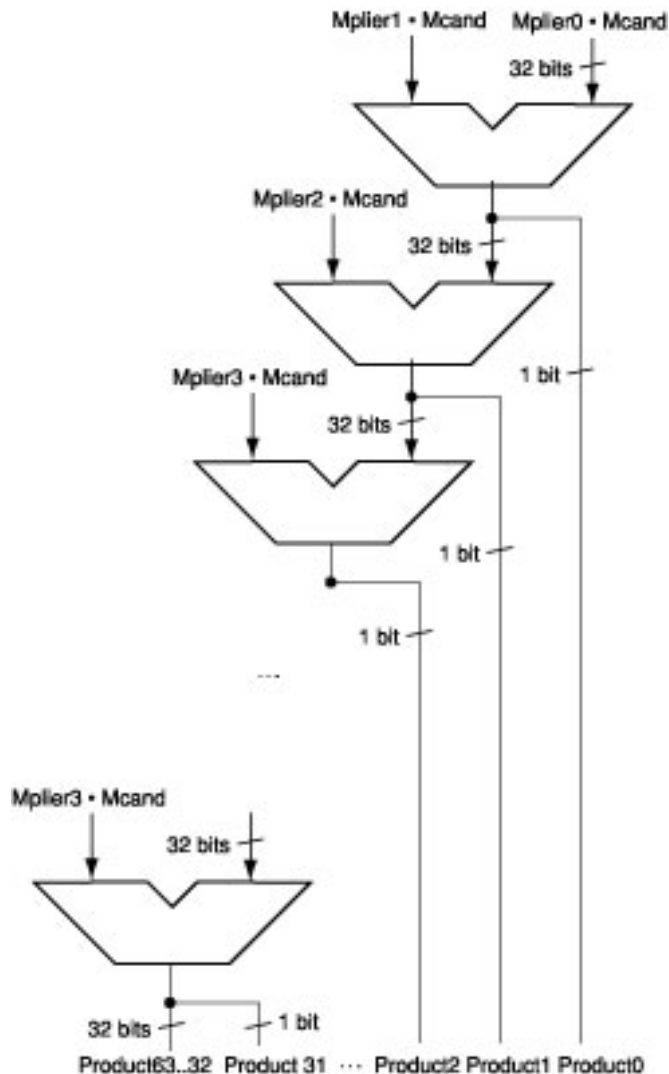
- The previous algorithm also works for signed numbers (negative numbers in 2's complement form)
- We can also convert negative numbers to positive, multiply the magnitudes, and convert to negative if signs disagree
- The product of two 32-bit numbers can be a 64-bit number -- hence, in MIPS, the product is saved in two 32-bit registers

MIPS Instructions

mult	\$s2, \$s3	computes the product and stores it in two “internal” registers that can be referred to as hi and lo
mfhi	\$s0	moves the value in hi into \$s0
mflo	\$s1	moves the value in lo into \$s1

Similarly for multu

Fast Algorithm



- The previous algorithm requires a clock to ensure that the earlier addition has completed before shifting
 - This algorithm can quickly set up most inputs – it then has to wait for the result of each add to propagate down – faster because no clock is involved
- Note: high transistor cost

Division

		$\begin{array}{r} 1001_{\text{ten}} \\ \hline 1000_{\text{ten}} \overline{) 1001010_{\text{ten}}} \\ \underline{-1000} \\ 10 \\ 101 \\ 1010 \\ \underline{-1000} \\ 10_{\text{ten}} \end{array}$	Quotient Dividend
Divisor	1000_{ten}		Remainder

At every step,

- shift divisor right and compare it with current dividend
- if divisor is larger, shift 0 as the next bit of the quotient
- if divisor is smaller, subtract to get new dividend and shift 1 as the next bit of the quotient

Division

		$\begin{array}{r} 1001_{\text{ten}} \\ \hline 1001010_{\text{ten}} \end{array}$	Quotient Dividend
Divisor 1000_{ten}			
	0001001010 $100000000000 \rightarrow$ Quo: 0	0001001010 $0001000000 \rightarrow$ 000001	0000001010 $0000100000 \rightarrow$ 0000010
		0000001010 0000001000 000001001	

At every step,

- shift divisor right and compare it with current dividend
- if divisor is larger, shift 0 as the next bit of the quotient
- if divisor is smaller, subtract to get new dividend and shift 1 as the next bit of the quotient

Divide Example

- Divide 7_{ten} ($0000\ 0111_{\text{two}}$) by 2_{ten} (0010_{two})

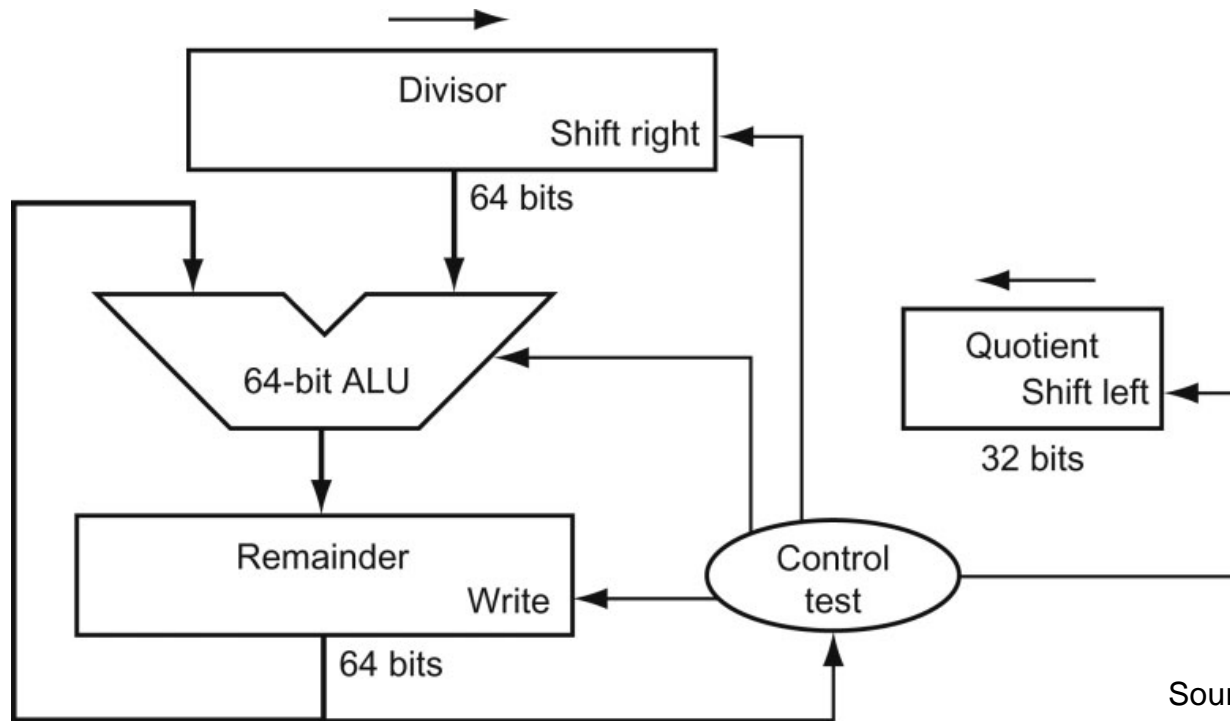
Iter	Step	Quot	Divisor	Remainder
0	Initial values			
1				
2				
3				
4				
5				

Divide Example

- Divide 7_{ten} ($0000\ 0111_{\text{two}}$) by 2_{ten} (0010_{two})

Iter	Step	Quot	Divisor	Remainder
0	Initial values	0000	0010 0000	0000 0111
1	Rem = Rem – Div	0000	0010 0000	1110 0111
	Rem < 0 → +Div, shift 0 into Q	0000	0010 0000	0000 0111
	Shift Div right	0000	0001 0000	0000 0111
2	Same steps as 1	0000	0001 0000	1111 0111
		0000	0001 0000	0000 0111
		0000	0000 1000	0000 0111
3	Same steps as 1	0000	0000 0100	0000 0111
4	Rem = Rem – Div	0000	0000 0100	0000 0011
	Rem >= 0 → shift 1 into Q	0001	0000 0100	0000 0011
	Shift Div right	0001	0000 0010	0000 0011
5	Same steps as 4	0011	0000 0001	0000 0001

Hardware for Division

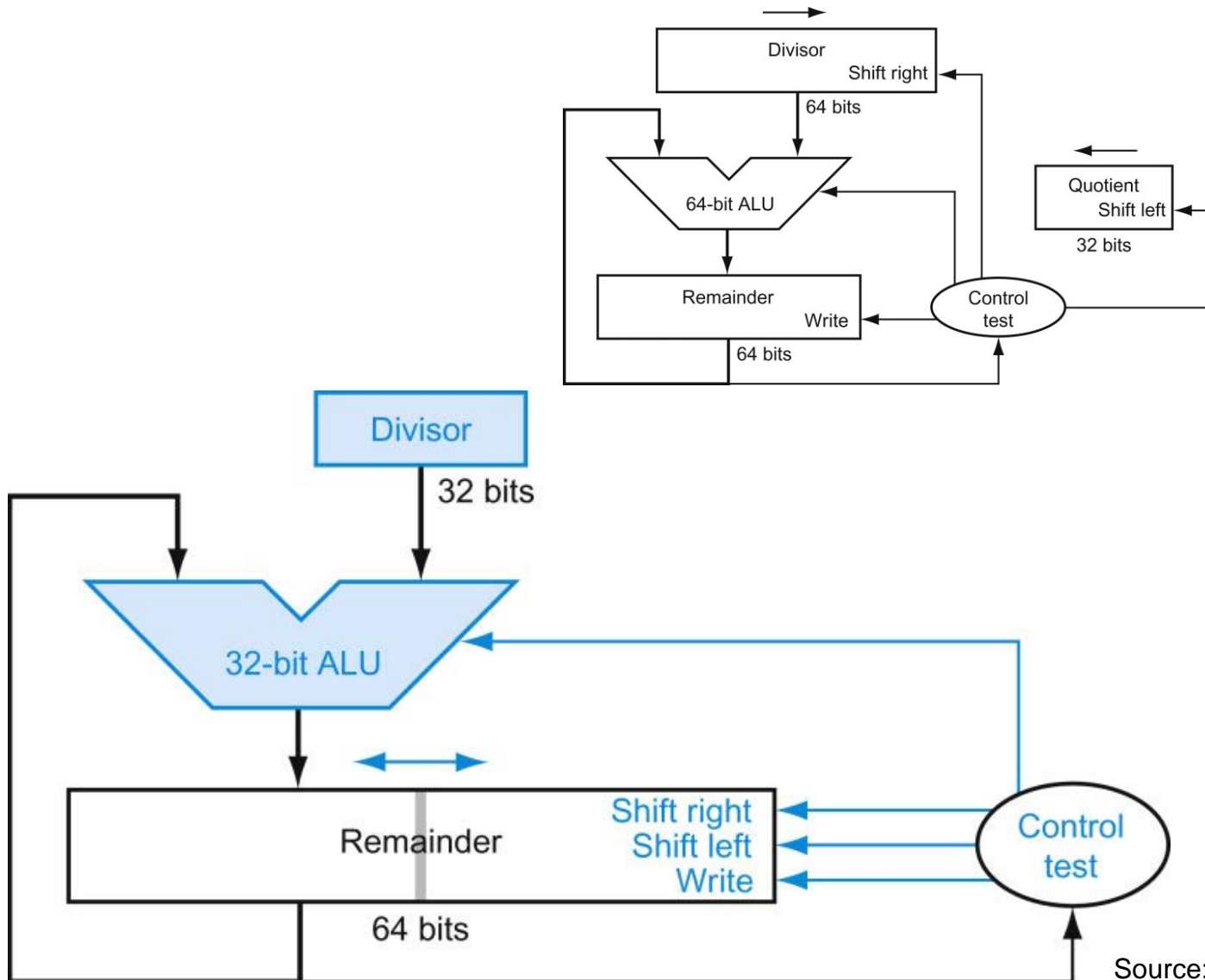


Source: H&P textbook

A comparison requires a subtract; the sign of the result is examined; if the result is negative, the divisor must be added back

Similar to multiply, results are placed in Hi (remainder) and Lo (quotient)

Efficient Division



Divisions Involving Negatives

- Simplest solution: convert to positive and adjust sign later
- Note that multiple solutions exist for the equation:
Dividend = Quotient x Divisor + Remainder

+7	div	+2	Quo =	Rem =
-7	div	+2	Quo =	Rem =
+7	div	-2	Quo =	Rem =
-7	div	-2	Quo =	Rem =

Divisions involving Negatives

- Simplest solution: convert to positive and adjust sign later
- Note that multiple solutions exist for the equation:
Dividend = Quotient x Divisor + Remainder

+7	div	+2	Quo = +3	Rem = +1
-7	div	+2	Quo = -3	Rem = -1
+7	div	-2	Quo = -3	Rem = +1
-7	div	-2	Quo = +3	Rem = -1

Convention: Dividend and remainder have the same sign
Quotient is negative if signs disagree
These rules fulfil the equation above