

Computer Organization and Architecture

Under Graduate Course
(B. Tech-Information Technology, 2nd Semester)
Jan 2020-July 2020

By

Dr. Satish Kumar Singh



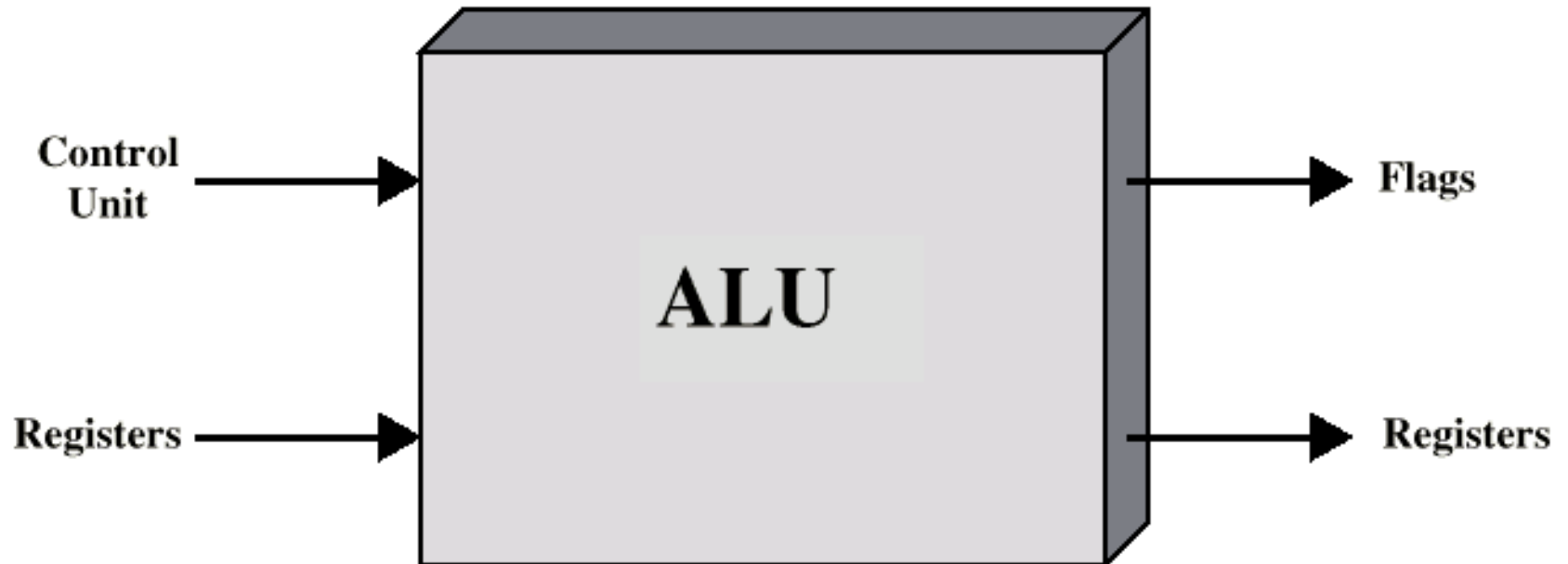
Associate Professor
Indian Institute of Information Technology, Allahabad
Email: sk.singh@iiita.ac.in

Computer Arithmetic

Arithmetic & Logic Unit

- Does the calculations
- Everything else in the computer is there to service this unit
- Handles integers
- May handle floating point (real) numbers
- May be separate FPU (maths co-processor)
- May be on chip separate FPU (486DX +)

ALU Inputs and Outputs



Integer Representation

- Only have 0 & 1 to represent everything
- Positive numbers stored in binary
 - e.g. $41 = 00101001$
- No minus sign
- No period
- Sign-Magnitude
- Two's complement

Sign-Magnitude

- Left most bit is sign bit
- 0 means positive
- 1 means negative
- $+18 = 00010010$
- $-18 = 10010010$
- Problems
 - Need to consider both sign and magnitude in arithmetic
 - Two representations of zero (+0 and -0)

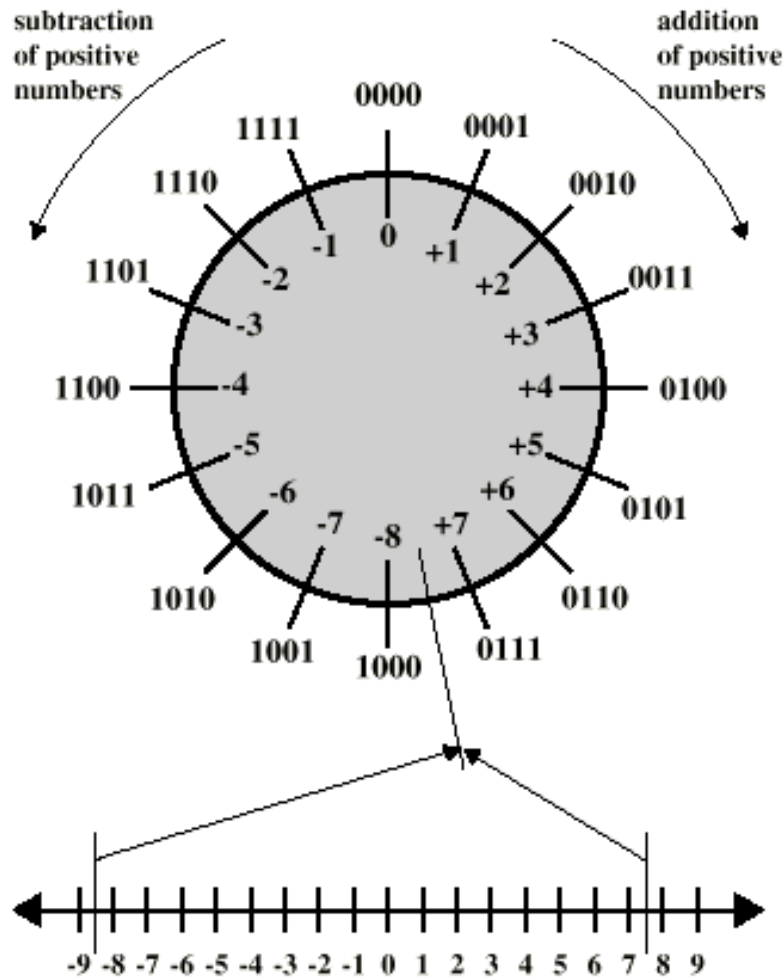
Two's Complement

- $+3 = 00000011$
- $+2 = 00000010$
- $+1 = 00000001$
- $+0 = 00000000$
- $-1 = 11111111$
- $-2 = 11111110$
- $-3 = 11111101$

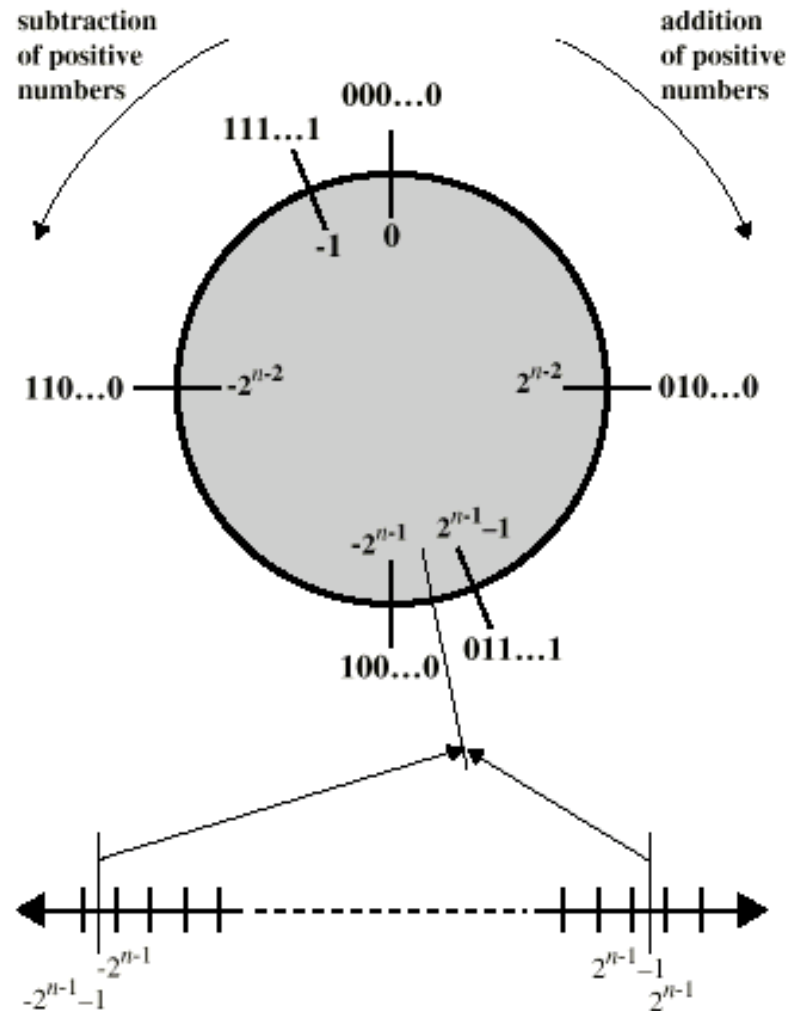
Benefits

- One representation of zero
- Arithmetic works easily (see later)
- Negating is fairly easy
 - $3 = 00000011$
 - Boolean complement gives 11111100
 - Add 1 to LSB 11111101

Geometric Depiction of Twos Complement Integers



(a) 4-bit numbers



(b) n-bit numbers

Negation Special Case 1

- $0 =$ 00000000
- Bitwise not 11111111
- Add 1 to LSB +1
- Result 1 00000000
- Overflow is ignored, so:
- $-0 = 0 \checkmark$

Negation Special Case 2

- $-128 = 10000000$
- bitwise not 01111111
- Add 1 to LSB $+1$
- Result 10000000
- So:
- $-(-128) = -128 \quad X$
- Monitor MSB (sign bit)
- It should change during negation

Range of Numbers

- 8 bit 2s compliment

- +127 = 01111111 = $2^7 - 1$

- -128 = 10000000 = -2^7

- 16 bit 2s compliment

- +32767 = 01111111 11111111 = $2^{15} - 1$

- -32768 = 10000000 00000000 = -2^{15}

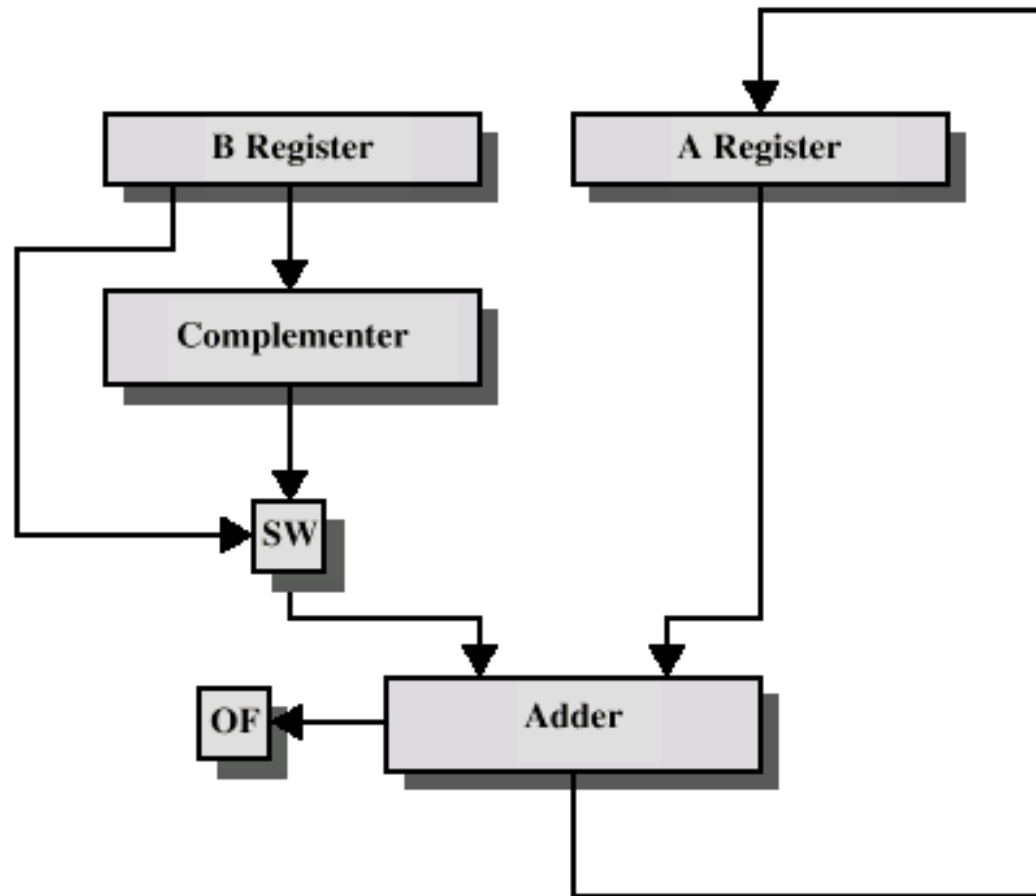
Conversion Between Lengths

- Positive number pack with leading zeros
- $+18 = \quad\quad\quad 00010010$
- $+18 = 00000000\ 00010010$
- Negative numbers pack with leading ones
- $-18 = \quad\quad\quad 10010010$
- $-18 = 11111111\ 10010010$
- i.e. pack with MSB (sign bit)

Addition and Subtraction

- Normal binary addition
- Monitor sign bit for overflow
- Take two's complement of subtrahend and add to minuend
 - i.e. $a - b = a + (-b)$
- So we only need addition and complement circuits

Hardware for Addition and Subtraction



OF = overflow bit

SW = Switch (select addition or subtraction)

Multiplication

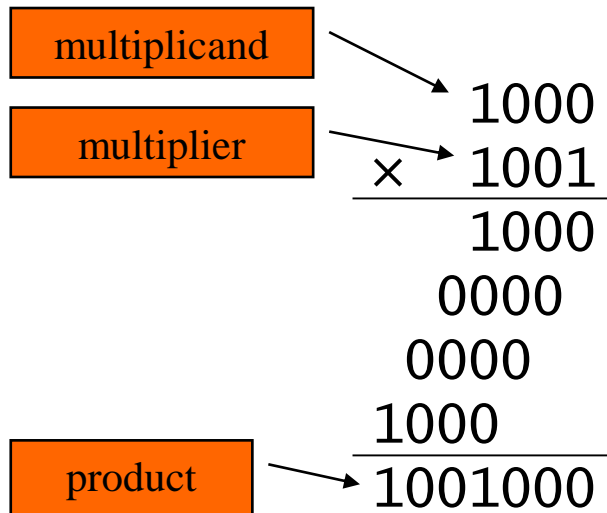
- Complex
- Work out partial product for each digit
- Take care with place value (column)
- Add partial products

Multiplication Example

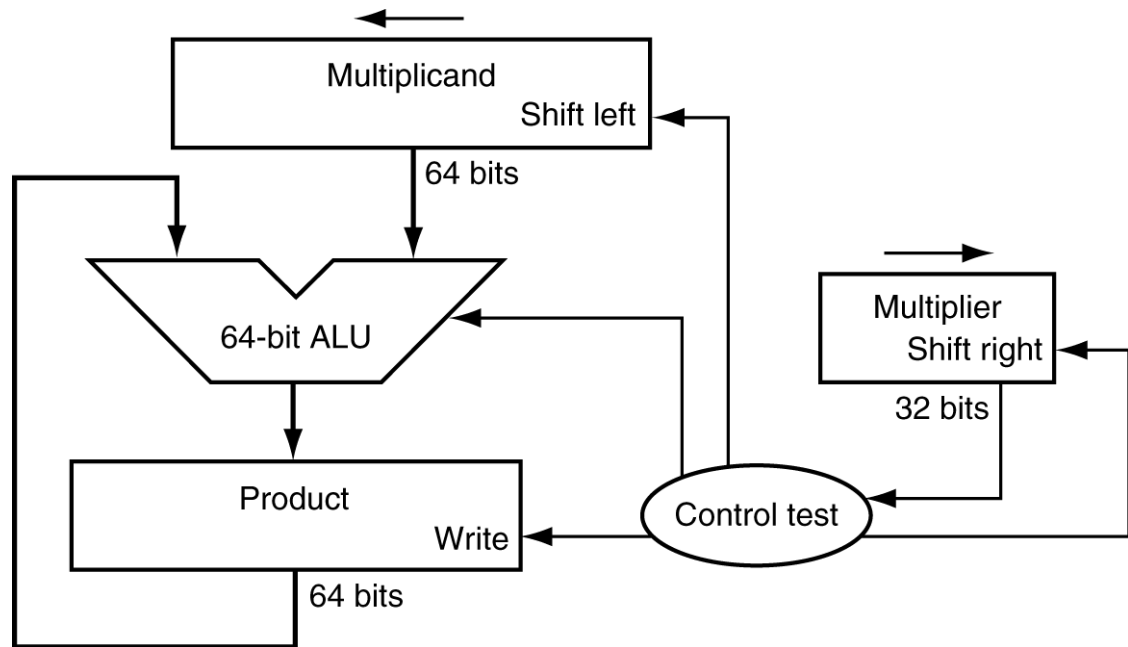
- 1011 Multiplicand (11 dec)
- x 1101 Multiplier (13 dec)
- 1011 Partial products
- 0000 Note: if multiplier bit is 1 copy
- 1011 multiplicand (place value)
- 1011 otherwise zero
- 10001111 Product (143 dec)
- Note: need double length result

Multiplication

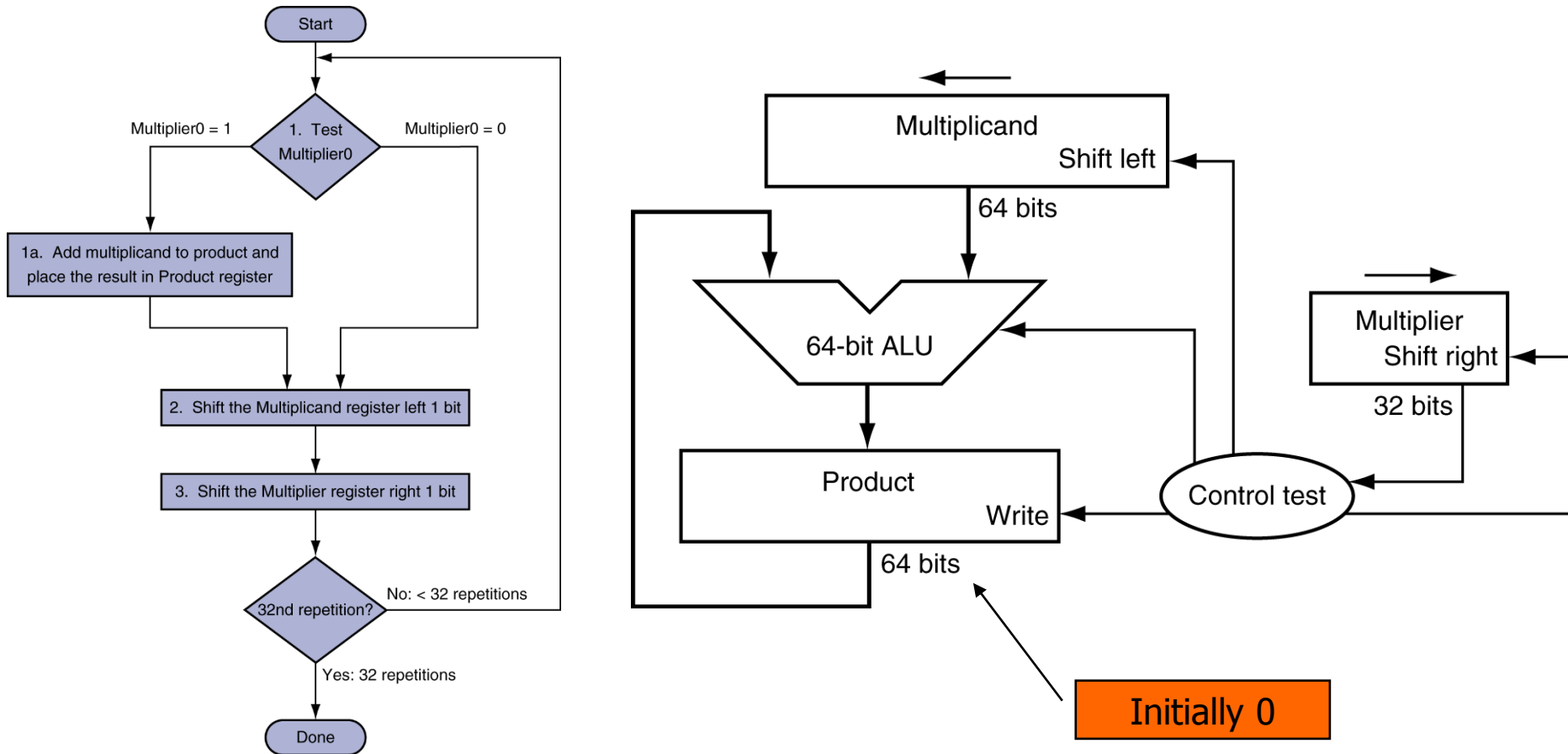
- Start with long-multiplication approach



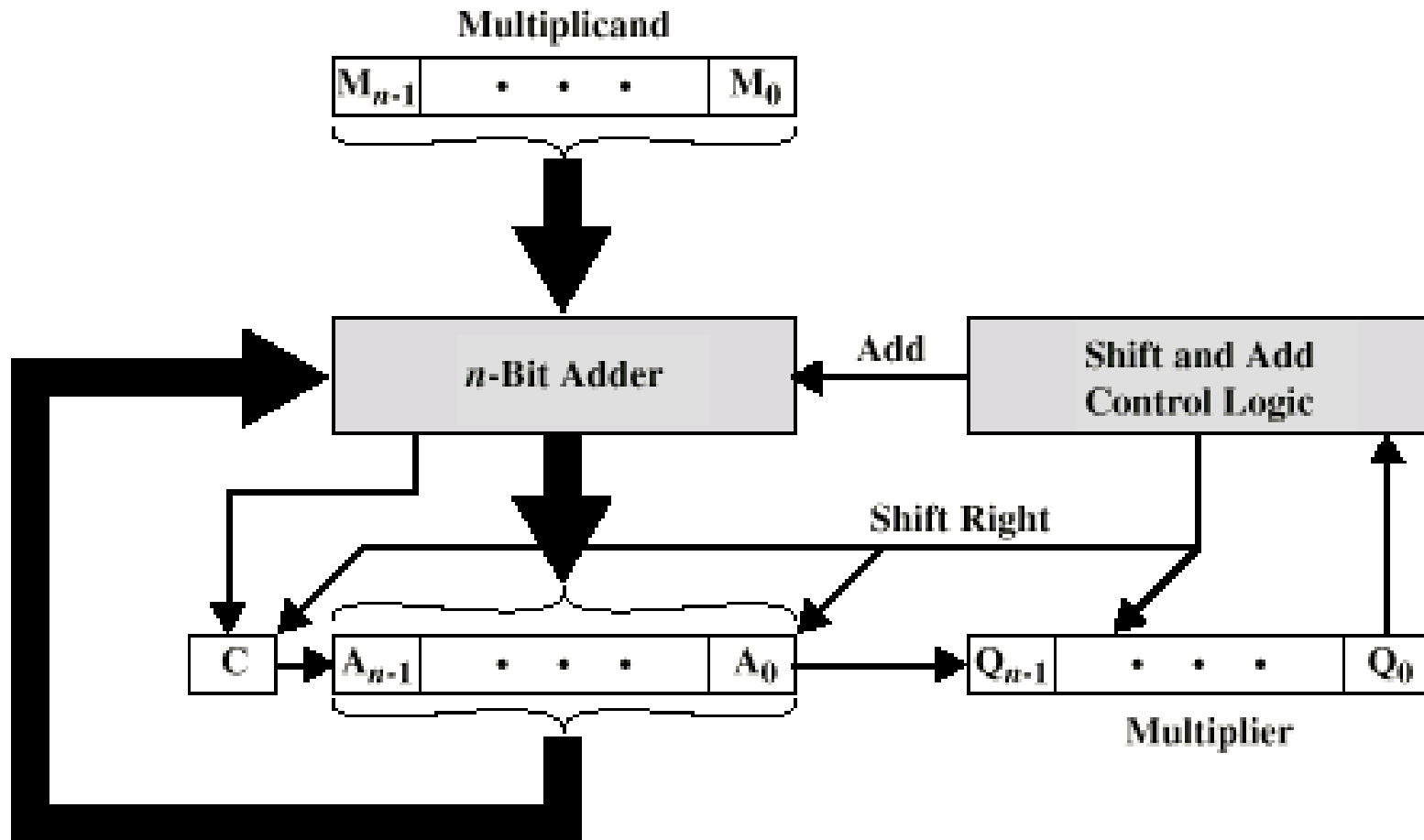
Length of product is the sum of operand lengths



Multiplication Hardware



Unsigned Binary Multiplication

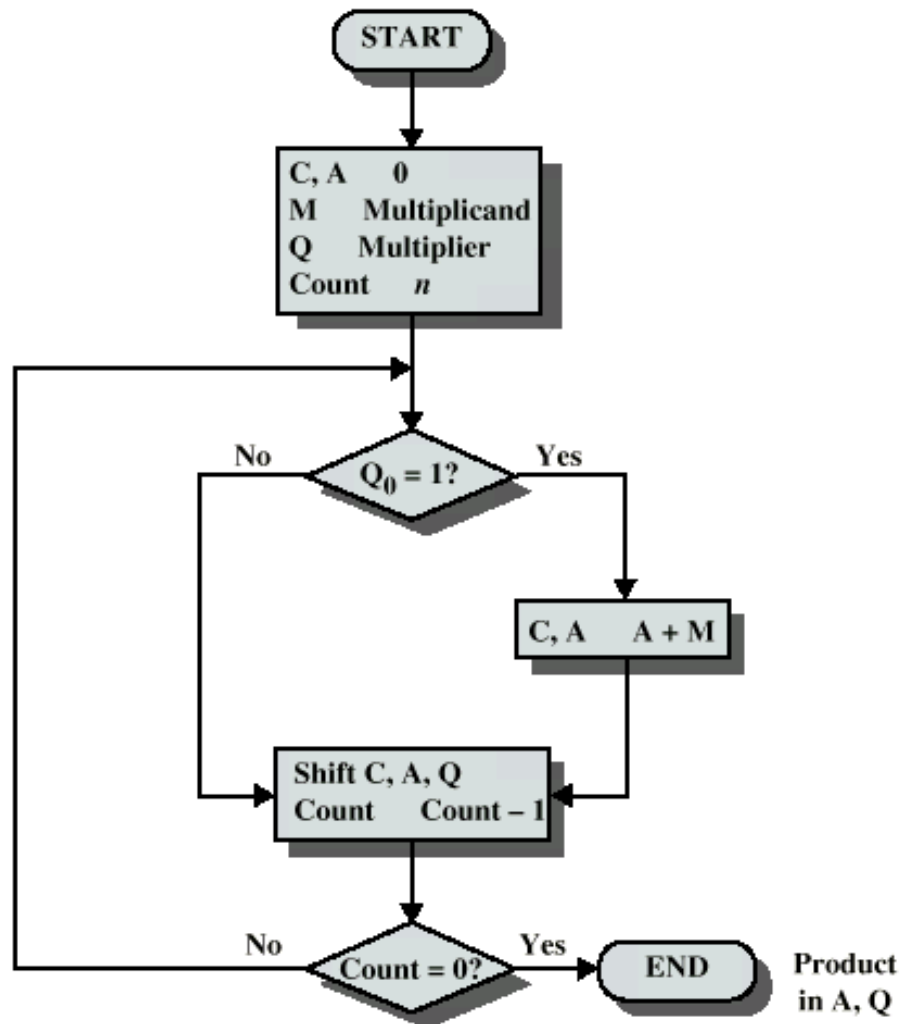


(a) Block Diagram

Execution of Example

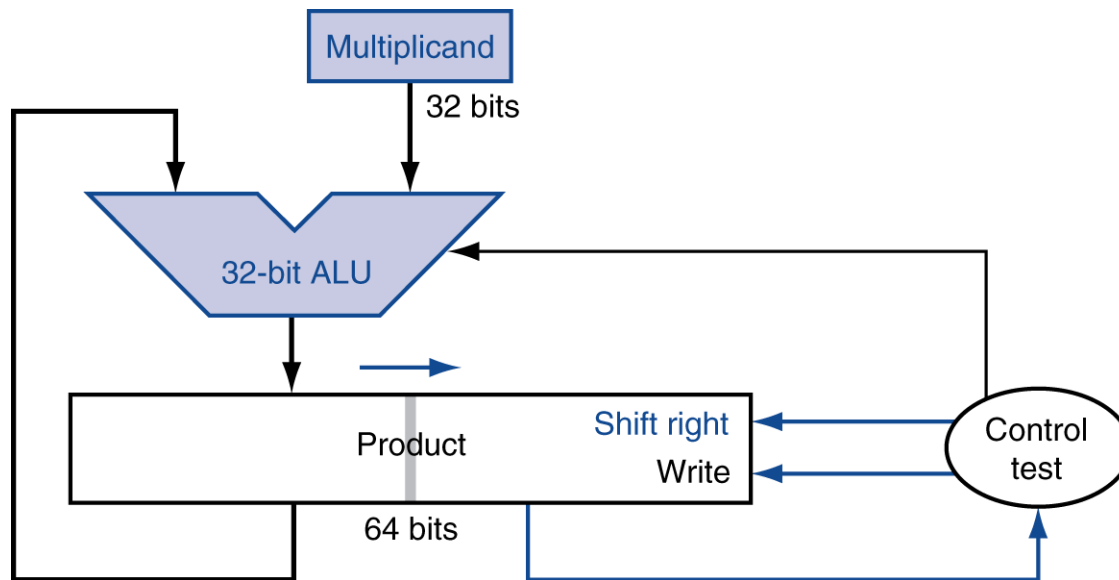
C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	} First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	} Second Cycle
0	1101	1111	1011	Add	
0	0110	1111	1011	Shift	} Third Cycle
1	0001	1111	1011	Add	
0	1000	1111	1011	Shift	} Fourth Cycle

Flowchart for Unsigned Binary Multiplication



Optimized Multiplier

- Perform steps in parallel: add/shift

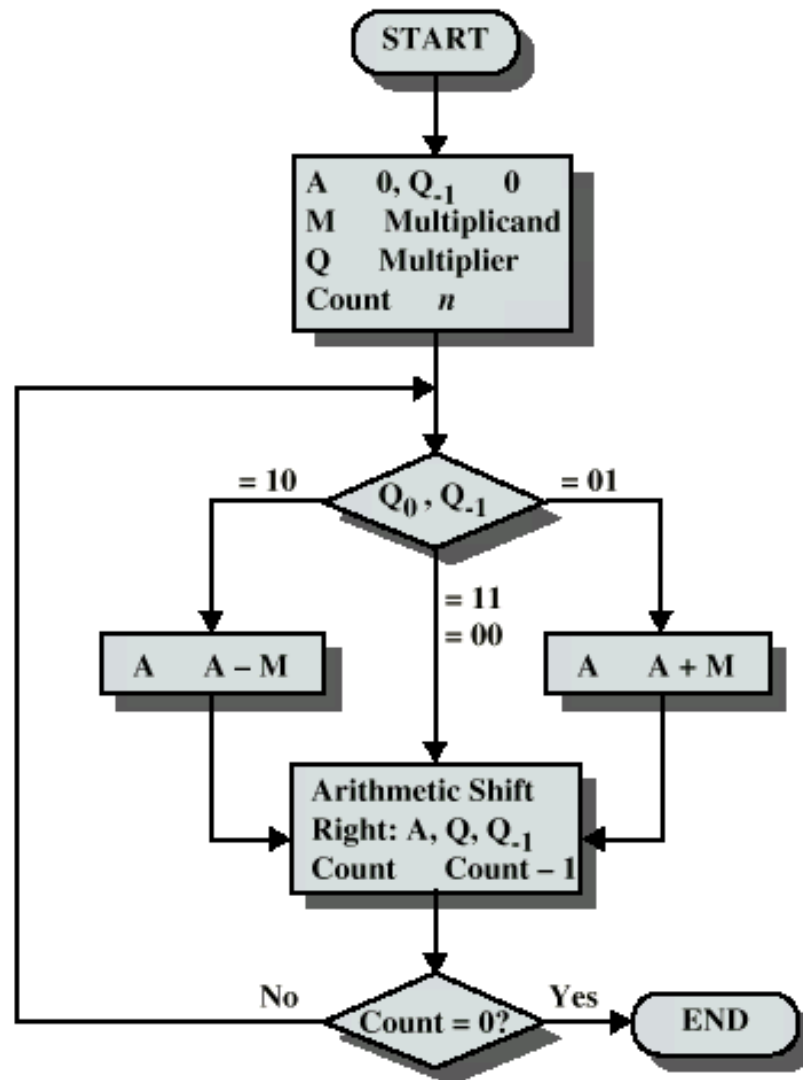


- One cycle per partial-product addition
 - That's ok, if frequency of multiplications is low

Multiplying Negative Numbers

- This does not work!
- Solution 1
 - Convert to positive if required
 - Multiply as above
 - If signs were different, negate answer
- Solution 2
 - Booth's algorithm

Booth's Algorithm

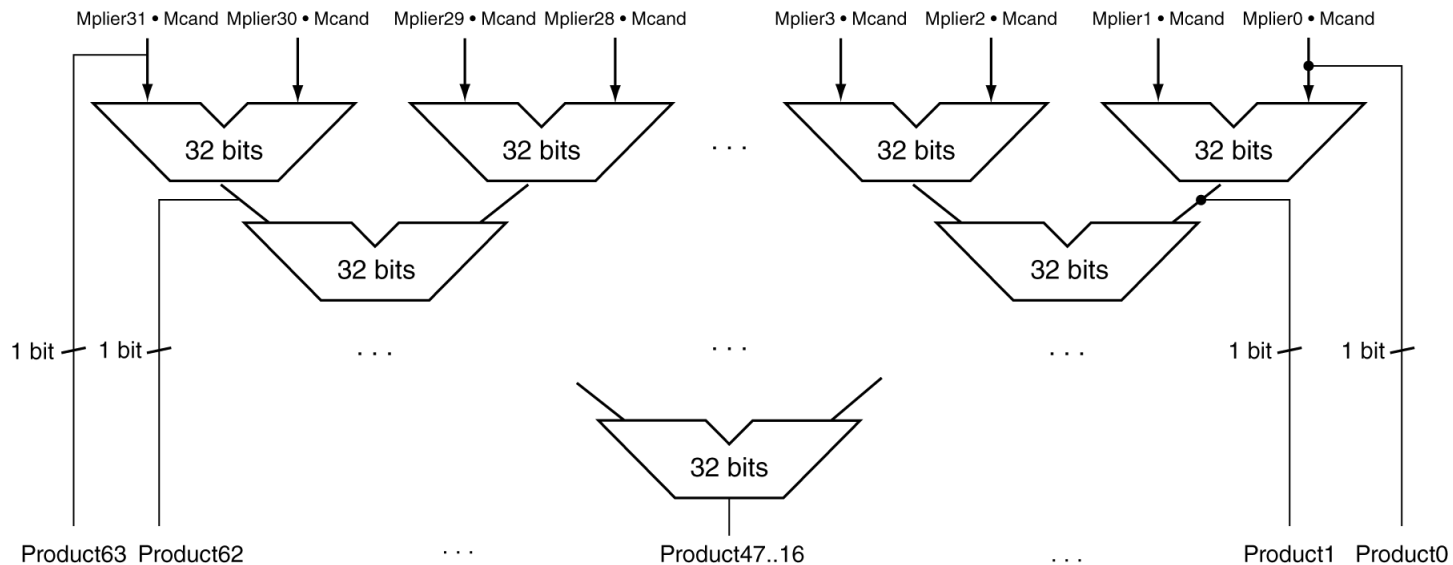


Example of Booth's Algorithm

A	Q	Q ₋₁	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A	} First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	} Second Cycle
0101	0100	1	0111	A	
0010	1010	0	0111	Shift	} Third Cycle
0001	0101	0	0111	Shift	
					} Fourth Cycle

Faster Multiplier

- Uses multiple adders
 - Cost/performance tradeoff

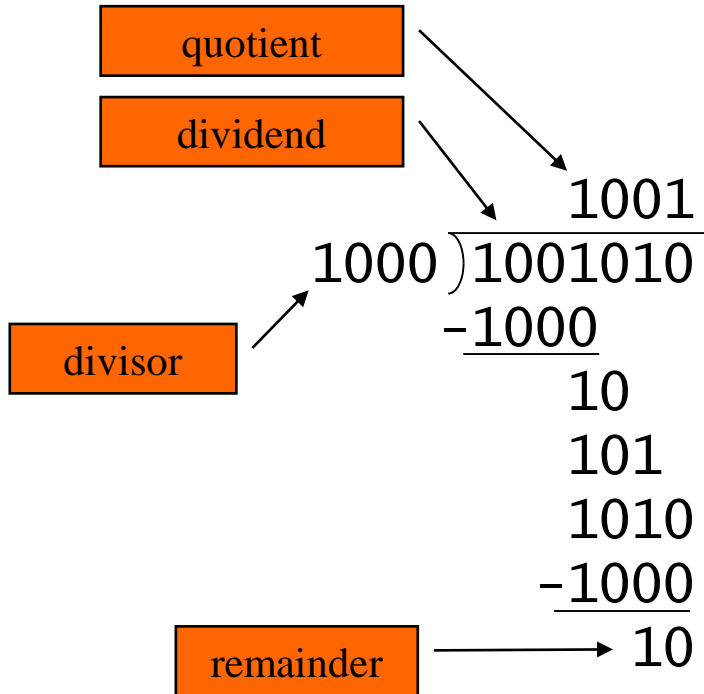


- Can be pipelined
 - Several multiplication performed in parallel

Division

- More complex than multiplication
- Negative numbers are really bad!
- Based on long division

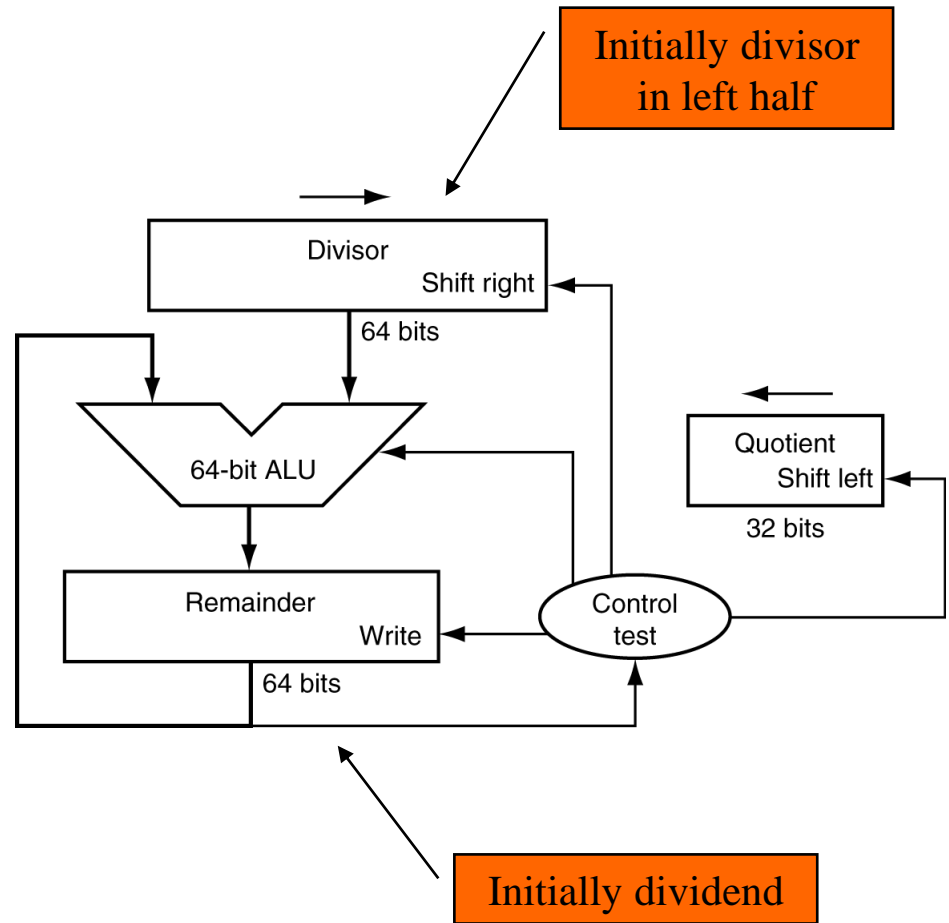
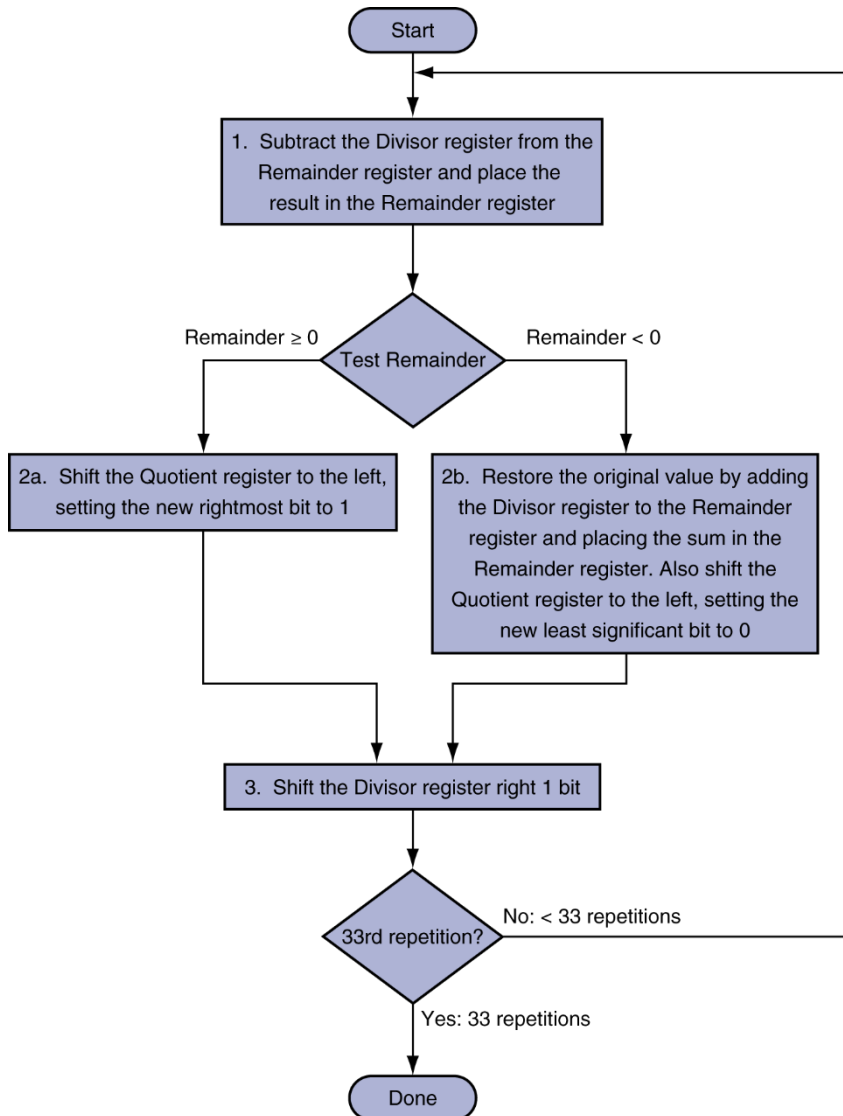
Division of Unsigned Binary Integers



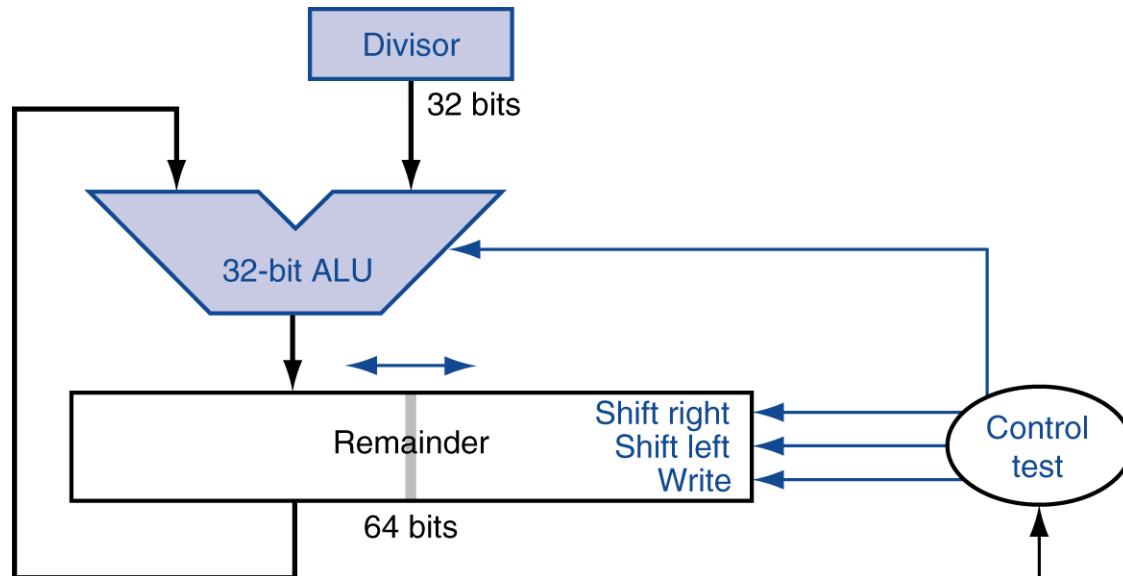
n-bit operands yield *n*-bit quotient and remainder

- Check for 0 divisor
- Long division approach
 - If divisor \leq dividend bits
 - 1 bit in quotient, subtract
 - Otherwise
 - 0 bit in quotient, bring down next dividend bit
- Restoring division
 - Do the subtract, and if remainder goes < 0 , add divisor back
- Signed division
 - Divide using absolute values
 - Adjust sign of quotient and remainder as required

Division Hardware

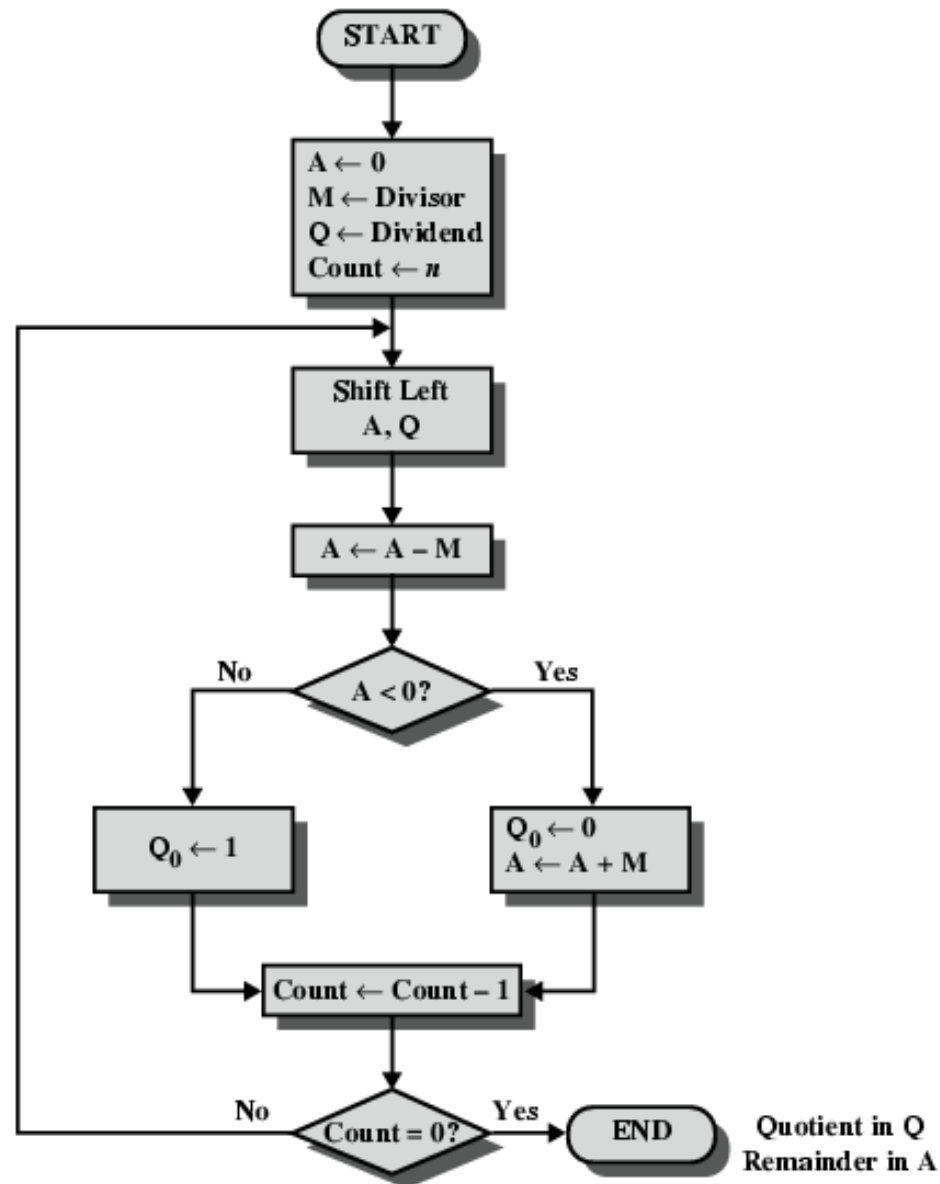


Optimized Divider



- One cycle per partial-remainder subtraction
- Looks a lot like a multiplier!
 - Same hardware can be used for both

Flowchart for Unsigned Binary Division



Required Reading

- Stallings Chapter 9
- IEEE 754 on IEEE Web site