

L13-CS8421-10-6-2008

Multiplication & Division

CS8421

Computing Systems

Dr. Ken Hoganson

Class

Will

Start

Momentarily...

Contents:

- Unsigned Binary Integer Multiplication
- Unsigned Binary Integer Division

Recall that it was claimed you could build a machine that implements: Add, Complement, AND as a minimal set (not fast).

- Complement & AND make NAND = complete set of boolean ops.
- Subtract is Complement & Add
- Mult & Division need SHIFT

- 4-bit number example

(11)	1011	Multiplicand
	x 1101	Multiplier (13)

	1011	Partial Products
	0000	
	1011	Notice the shift!
	1011	

	10001111	Product (143)

The multiplication process

- Multiplication involves the generation of partial products, which are then summed.
- The partial products are easy to see – depends on whether the bit in the multiplier is 0 or 1. When it's a 0, add in zero, when it is a 1, add in the multiplicand in the correct place value alignment.
- Multiplying two n -bit integers can result in a product of up to $2n$ bits.

- Multiplication Flowchart

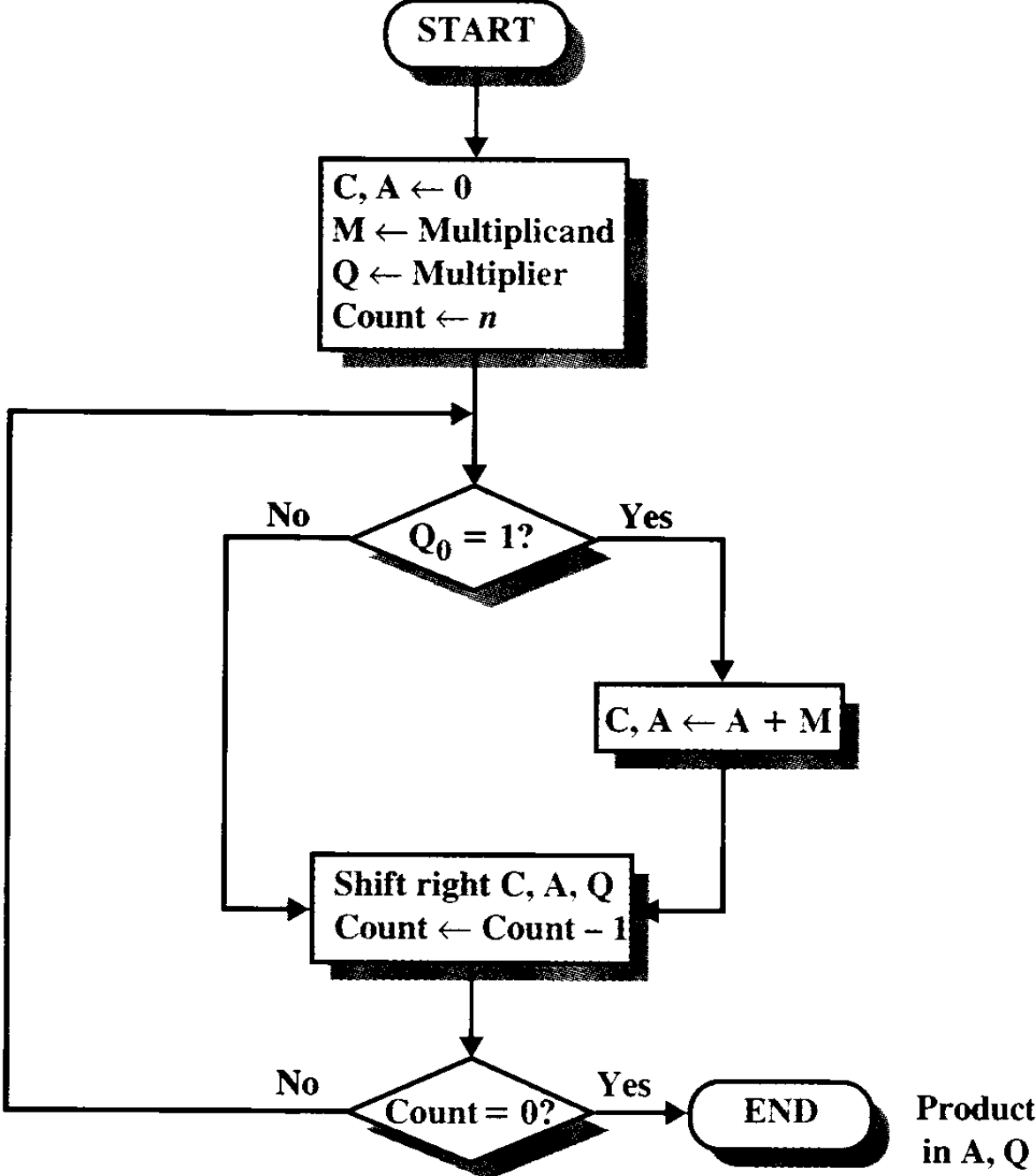


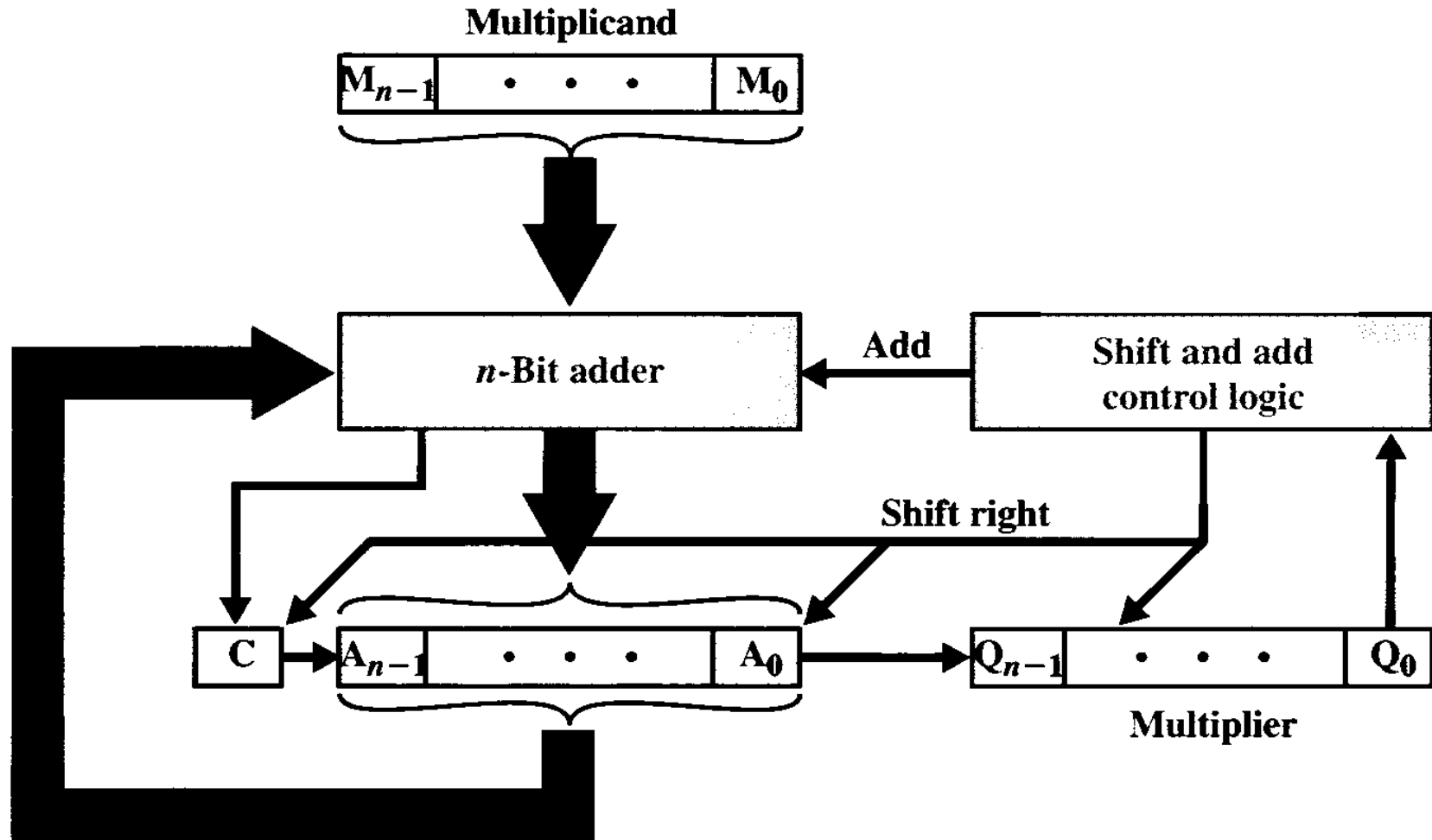
Figure 9.9 Flowchart for Unsigned Binary Multiplication

Add only if
 $Q_0 = 0$

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
		1111			

result

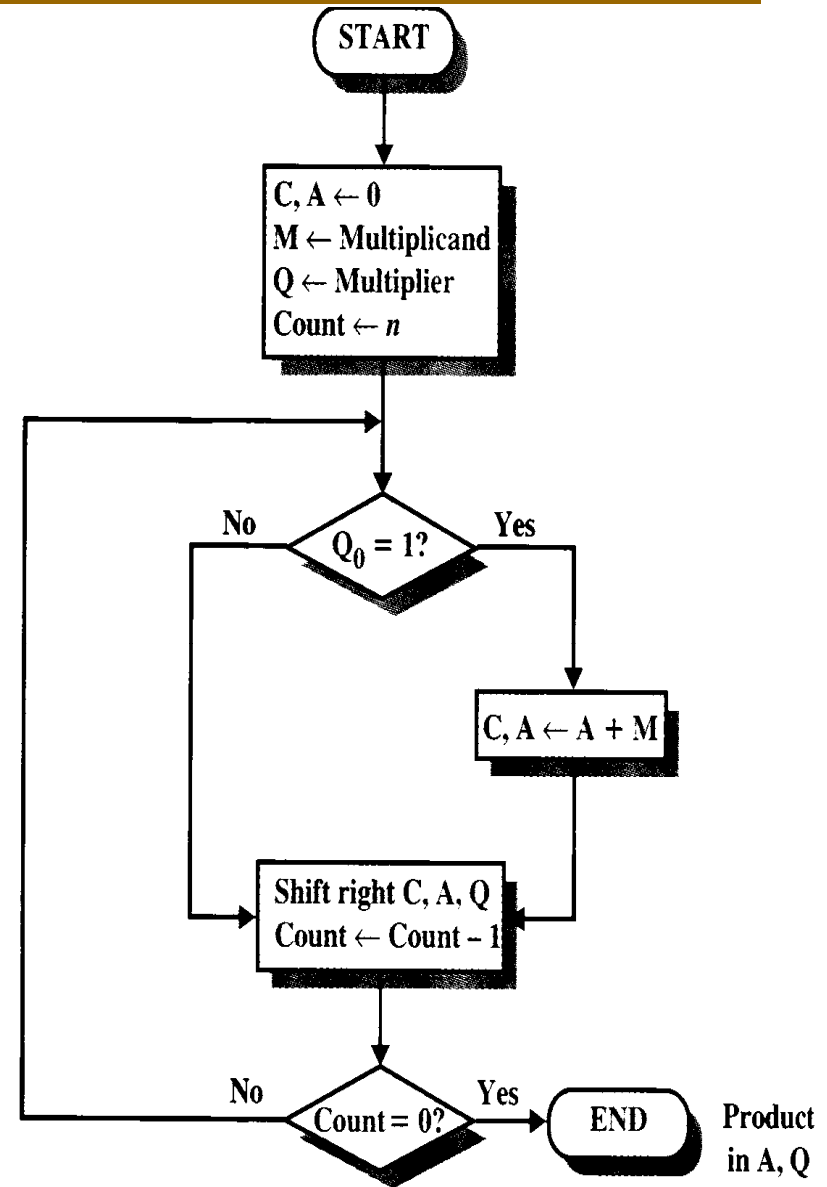
(b) Example from Figure 9.7 (product in A, Q)



(a) Block diagram

Multiply $14 * 5$

Count	C	A	Q	M
4	0	0000	0101	1110



Copyright © 2004, 2006 **Figure 9.9** Flowchart for Unsigned Binary Multiplication

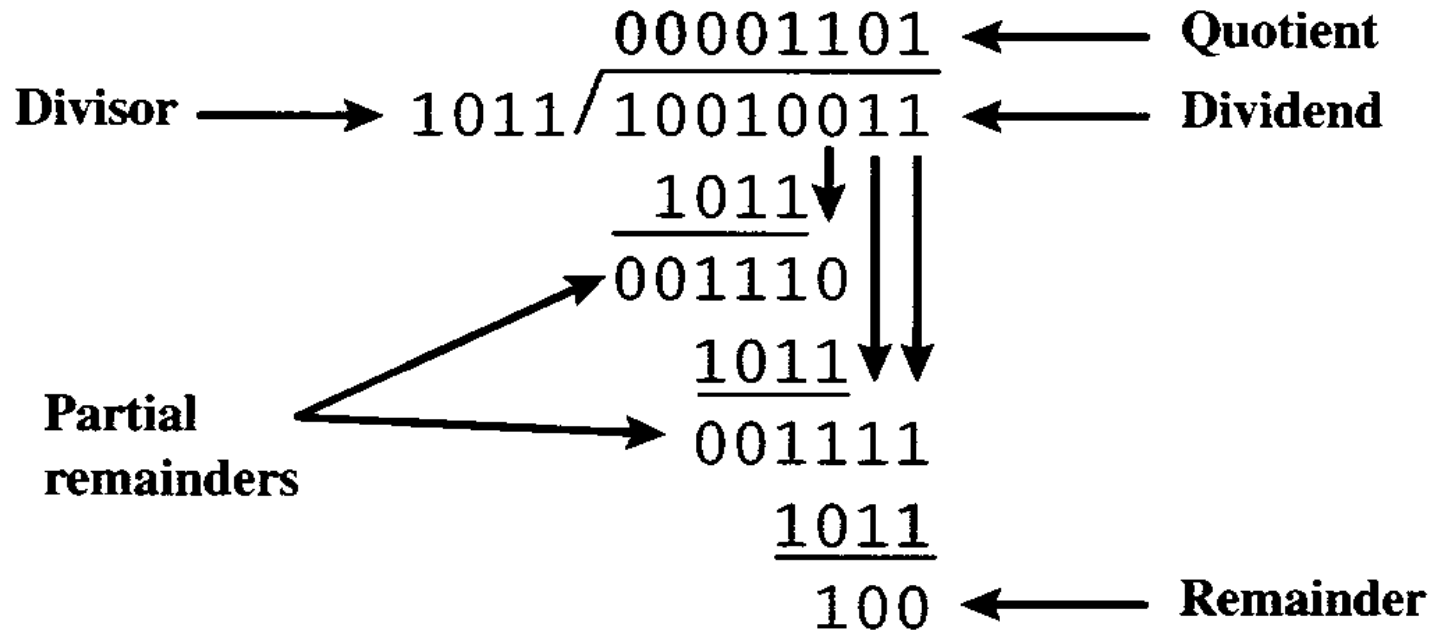


Figure 9.15 Example of Division of Unsigned Binary Integers

Division Example: 14/6

A	Q	M	count
0000	1110	0110	4
0001	110X	0110	4
0001	1100	0110	4
			3
0011	100X	0110	3
0011	1000	0110	3
			2
0111	000X	0110	2
0001	0001	0110	2
			1
0010	001X	0110	1
0010	0010	0110	1

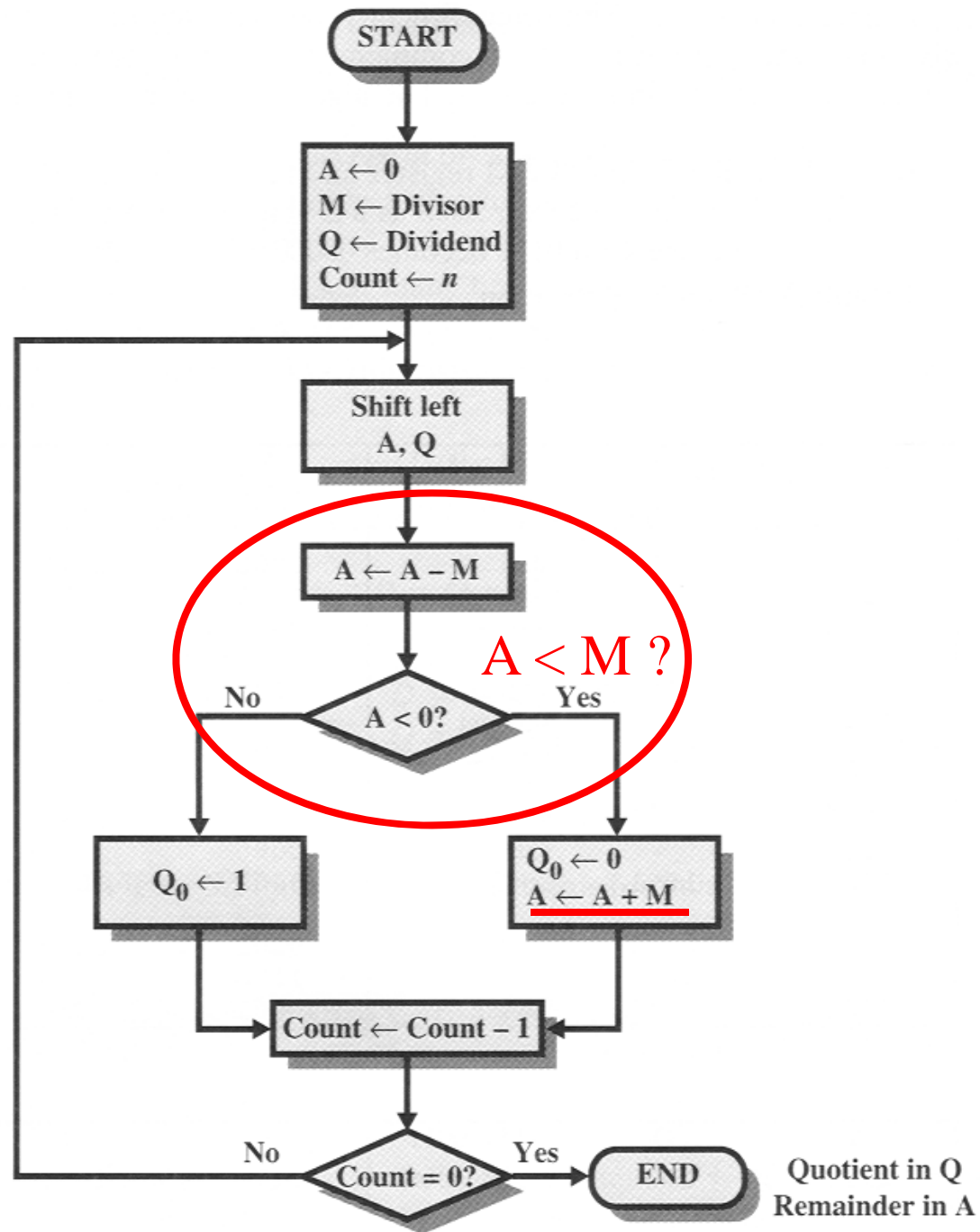
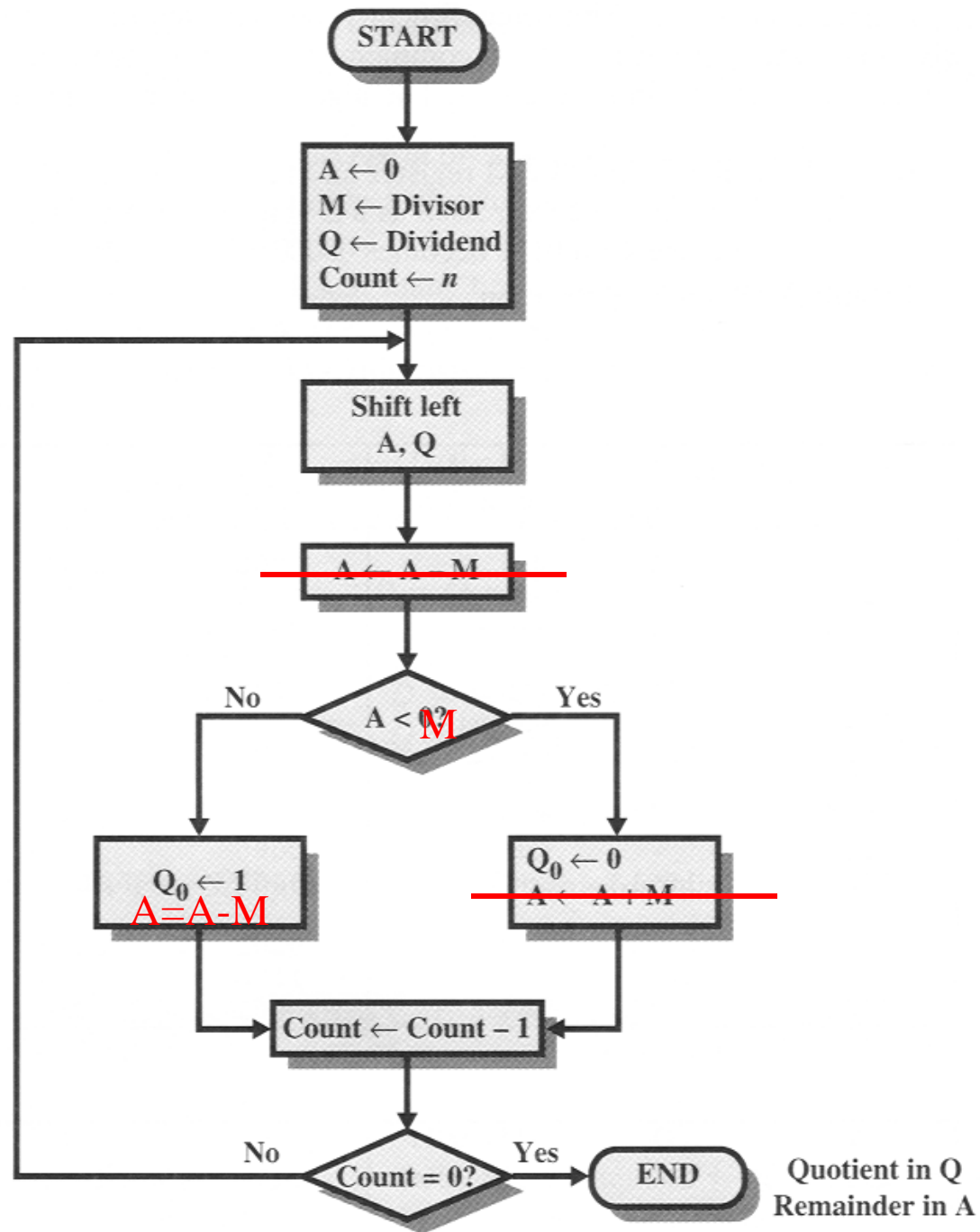


Figure 9.16 Flowchart for Unsigned Binary Division

Why $A = A - M$, compare, then add M back?

Alternative in RED

IF implemented with a compare that internally does a subtraction, in order to set flag register bits. May be more more efficient in that case.



2nd Example 12/3

A	Q	M	count
0000	1100	0011	4

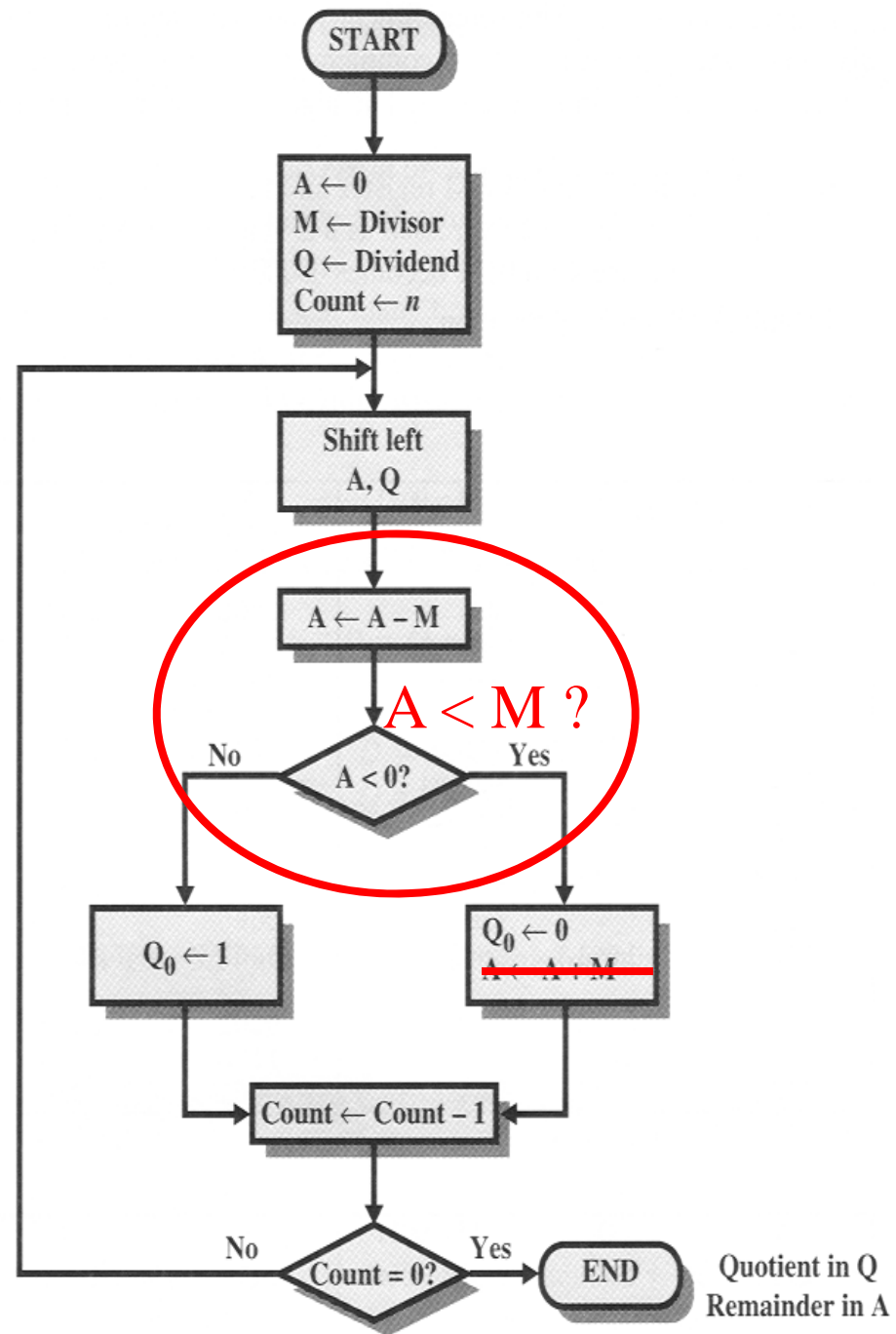


Figure 9.16 Flowchart for Unsigned Binary Division

**End
Of
Today's
Lecture.**

