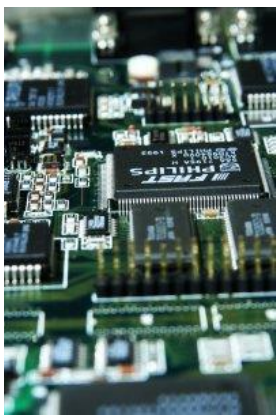
CSE 321b

Computer Organization (2)

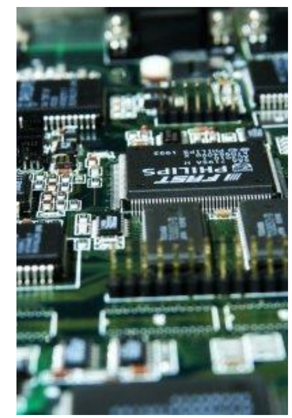
تنظيم الحاسب (2)



3rd year, Computer Engineering

Winter 2017

Lecture #8



Dr. Hazem Ibrahim Shehata
Dept. of Computer & Systems Engineering

Credits to Dr. Ahmed Abdul-Monem Ahmed for the slides

Adminstrivia

- Assignment #2:
  - Due: Thursday, April 13, 2017
- Midterm:
  - Date: Saturday, April 15, 2017
  - Time: 10:30am – 12:00pm
  - Location: classroom #27309
  - Coverage: lectures #1 ➔ #6

Website: http://hshehata.github.io/courses/zu/cse321b/
Office hours: TBA

Chapter 10. Computer Arithmetic (*Cont.*)

Outline

- Integer Representation
  - Sign-Magnitude, Two’s Complement, Biased
- Integer Arithmetic
  - Negation, Addition, Subtraction
  - Multiplication, Division
- Floating-Point Representation
  - IEEE 754
- Floating-Point Arithmetic
  - Addition, Subtraction
  - Multiplication, Division
  - Rounding

Multiplication Example

```
       1011     Multiplicand (11)
     × 1101     Multiplier (13)
   --------
       1011   ⎫
      0000    ⎪
     1011     ⎬ Partial products
    1011      ⎭
   --------
   10001111     Product (143)
```

المضروب Multiplicand (11)

المضاعِف Multiplier (13)

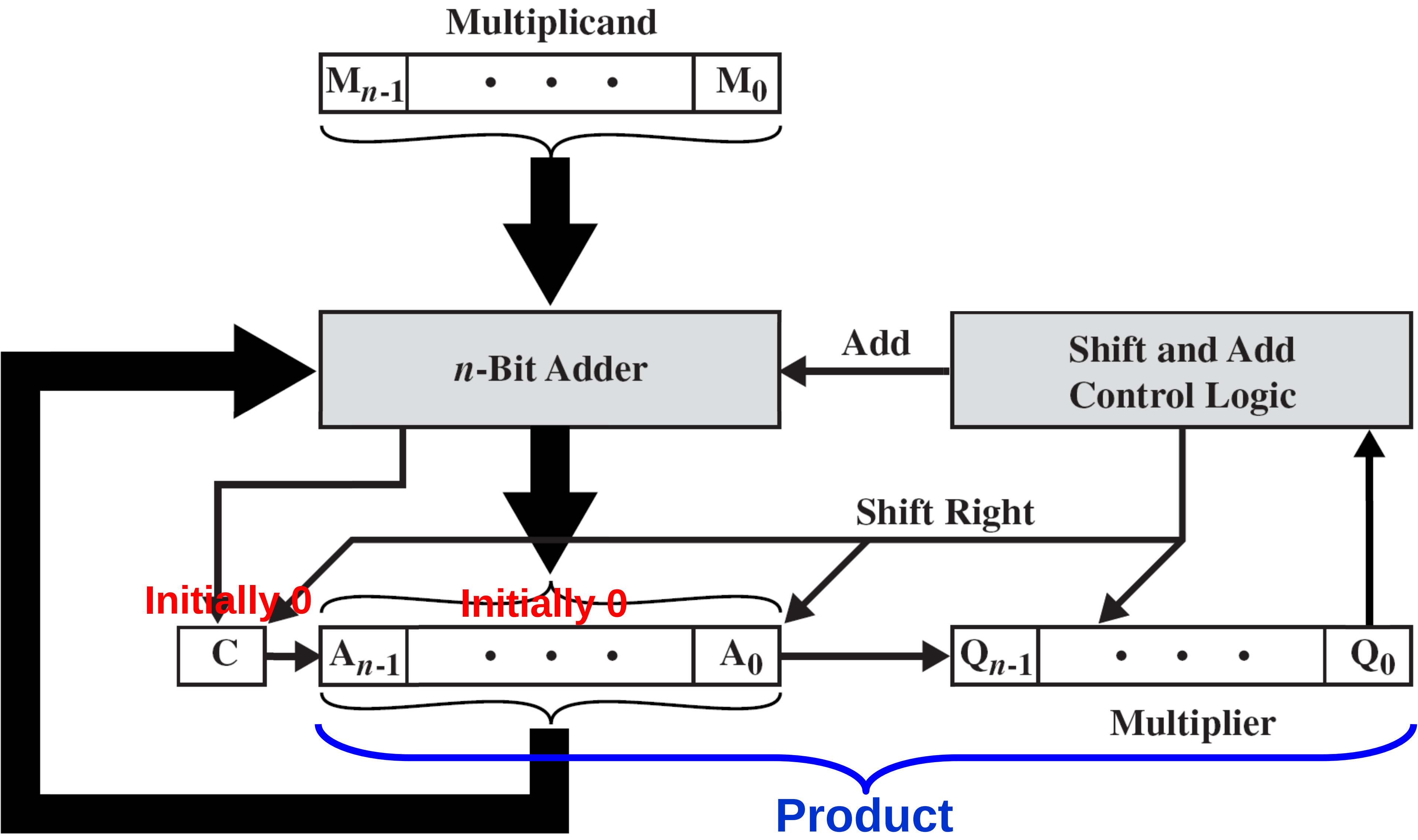
Partial products

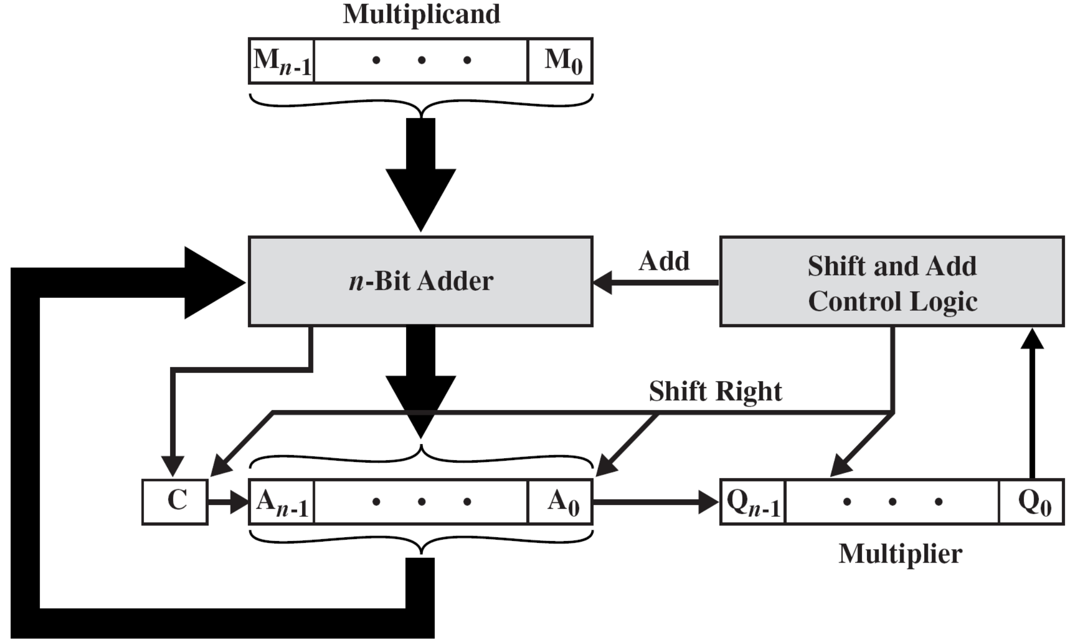
الناتج Product (143)

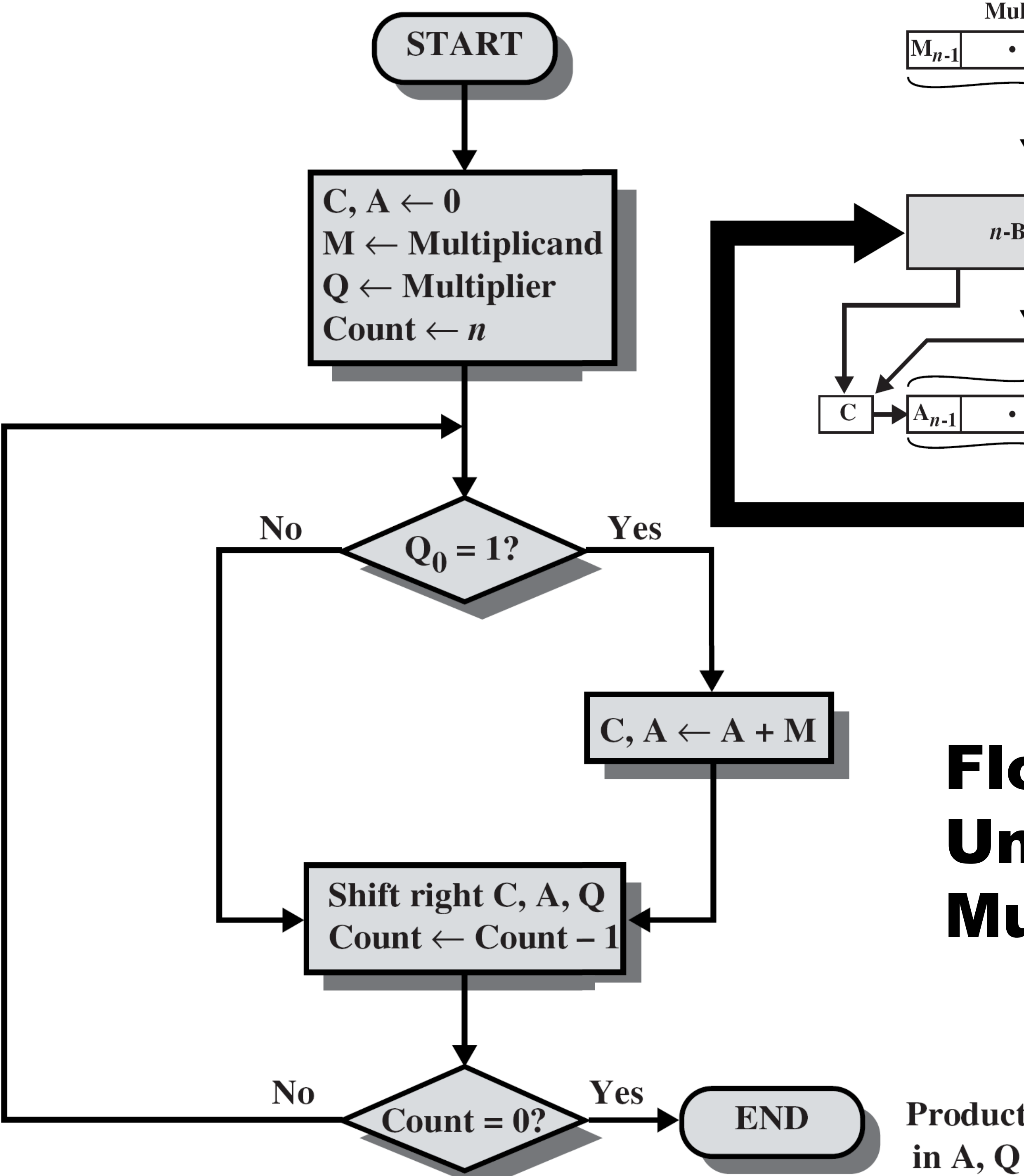
Running addition.
1 ➔ add & shift
0 ➔ shift only

- Complex (relative to addition)!!
  - Work out a partial product for each digit.
  - Shift the partial product appropriately.
  - Add partial products.
  - Generate double-length result.

Unsigned Binary Multiplication







Flowchart for Unsigned Binary Multiplication

Execution of Example

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

Product (143)

Signed Binary Multiplication

- The straight forward multiplication algorithm doesn't work with signed numbers!!
- Evidence: In the previous example, suppose that M & Q are interpreted as signed numbers:
  - M = $(1011)_2$ which represents $(-5)_{10}$
  - Q = $(1101)_2$ which represents $(-3)_{10}$
  - Applying the algorithm results in a product value of $(1000\ 1111)_2$ which represents $(-113)_{10}$
  - This result is wrong! Correct value is supposed to be $(+15)_{10}$!!!!

Signed Multiplication Algorithm #1

1. Convert multiplicand (M) & multiplier (Q) to their absolute (positive) values |M| & |Q|.
2. Run the unsigned multiplication algorithm on |M| & |Q| to obtain the final product (P).
3. Adjust the sign of P (by 2's complementation where needed) according to the following rule:
   - sign(P) = sign(M) X sign(Q)

Signed Multiplication Algorithm #2 (Booth's Algorithm)

	0	1	0	1	1	0	1
x	0	0	1	1	1	1	0

→

Multiplier

	0	1	0	0	0	0	0
–	0	0	0	0	0	1	0

→

Multiplier

0	+1	0	0	0	–1	0

							0	1	0	1	1	0	1
							0	0	+1	+1	+1	+1	0
							0	0	0	0	0	0	0
						0	1	0	1	1	0	1	
					0	1	0	1	1	0	1		
				0	1	0	1	1	0	1			
			0	1	0	1	1	0	1				
		0	0	0	0	0	0	0					
	0	0	0	0	0	0	0						
0	0	0	1	0	1	0	1	0	0	0	1	1	0

Booth's Algorithm – Example

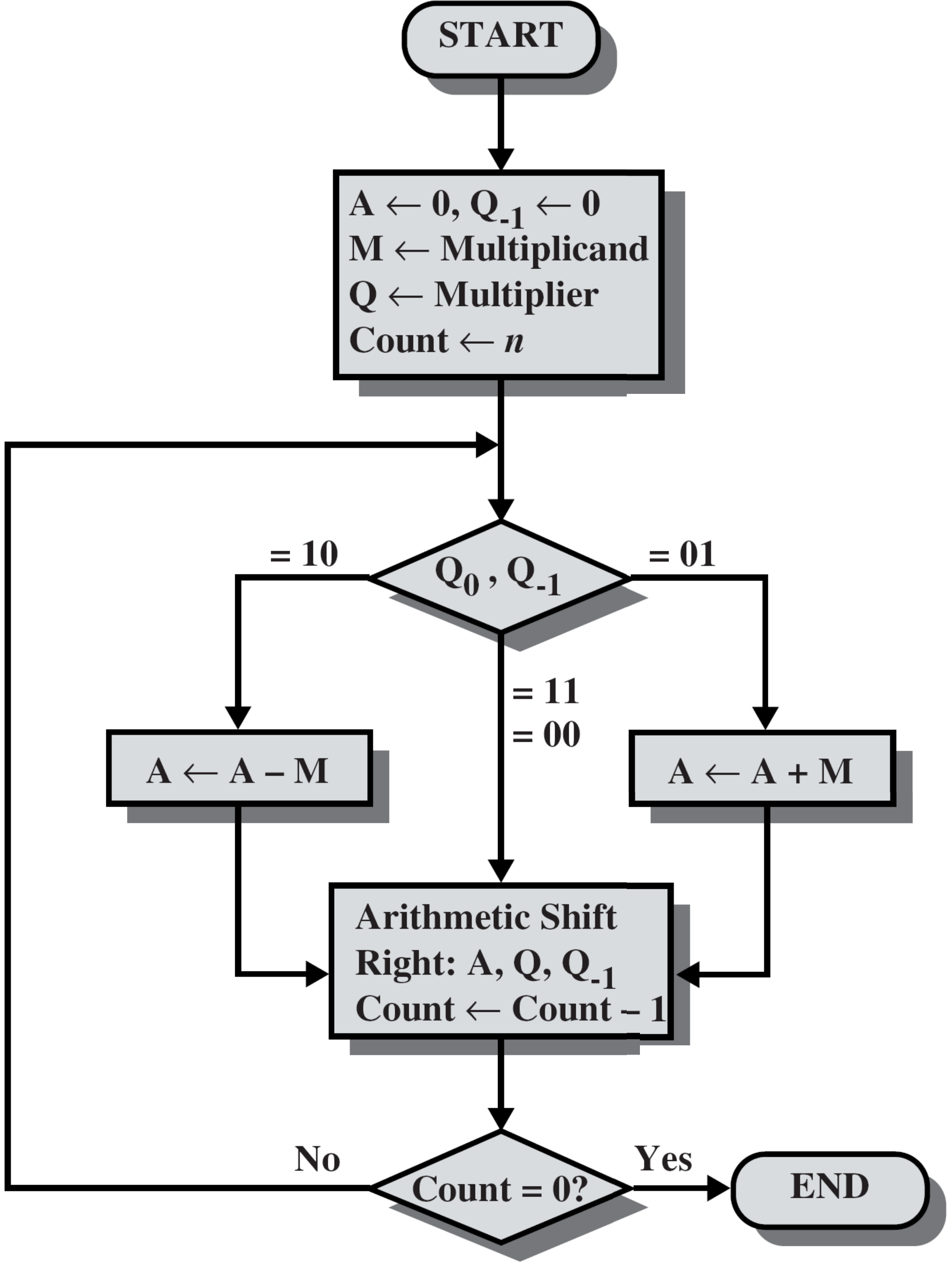
							0	1	0	1	1	0	1	
							0	+1	0	0	0	−1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	0	1	0	0	1	1		2's complement of the multiplicand
0	0	0	0	0	0	0	0	0	0	0	0			
0	0	0	0	0	0	0	0	0	0	0				
0	0	0	0	0	0	0	0	0	0					
0	0	0	1	0	1	1	0	1						
0	0	0	0	0	0	0	0							
0	0	0	1	0	1	0	1	0	0	0	1	1	0	

0	0	1	0	1	1	0	0	1	1	1	0	1	0	1	1	0	0
								⇓									
0	+1	−1	+1	0	−1	0	+1	0	0	−1	+1	−1	+1	0	−1	0	0

Booth's Algorithm – Rule

Multiplier Bit i	 Bit i–1	Version of multiplicand selected by bit i
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

Booth's Algorithm Flowchart



Example on Booth's Algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0		$A \leftarrow A - M$	First Cycle
1100	1001	1		Shift	
1110	0100	1		Shift	Second Cycle
0101	0100	1		$A \leftarrow A + M$	Third Cycle
0010	1010	0		Shift	
0001	0101	0		Shift	Fourth Cycle

Product (21)

Booth's Algorithm, –ve Multiplier

	0	1	1	0	1	(+13)
×	1	1	0	1	0	(−6)

					0	1	1	0	1	
					0	−1	+1	−1	0	
0	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	0	0	1	1		
0	0	0	0	1	1	0	1			
1	1	1	0	0	1	1				
0	0	0	0	0	0					
1	1	1	0	1	1	0	0	1	0	(−78)

Booth's Algorithm - Cases

Worst-case Multiplier	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1

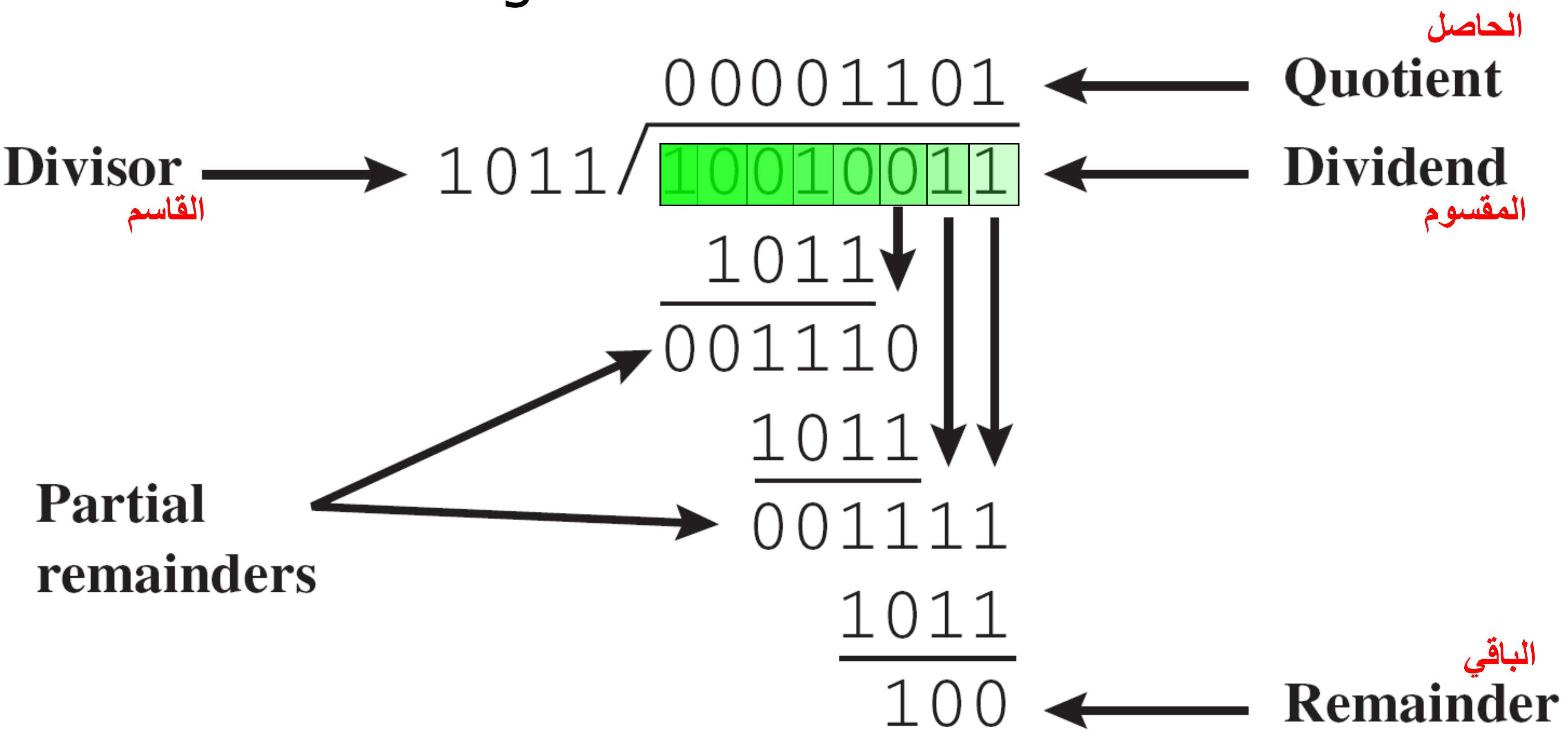
Ordinary Multiplier	1	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0
	0	-1	0	0	+1	-1	+1	0	-1	+1	0	0	0	-1	0	0

Good Multiplier	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1
	0	0	+1	0	0	0	0	-1	0	0	0	0	+1	0	0	-1

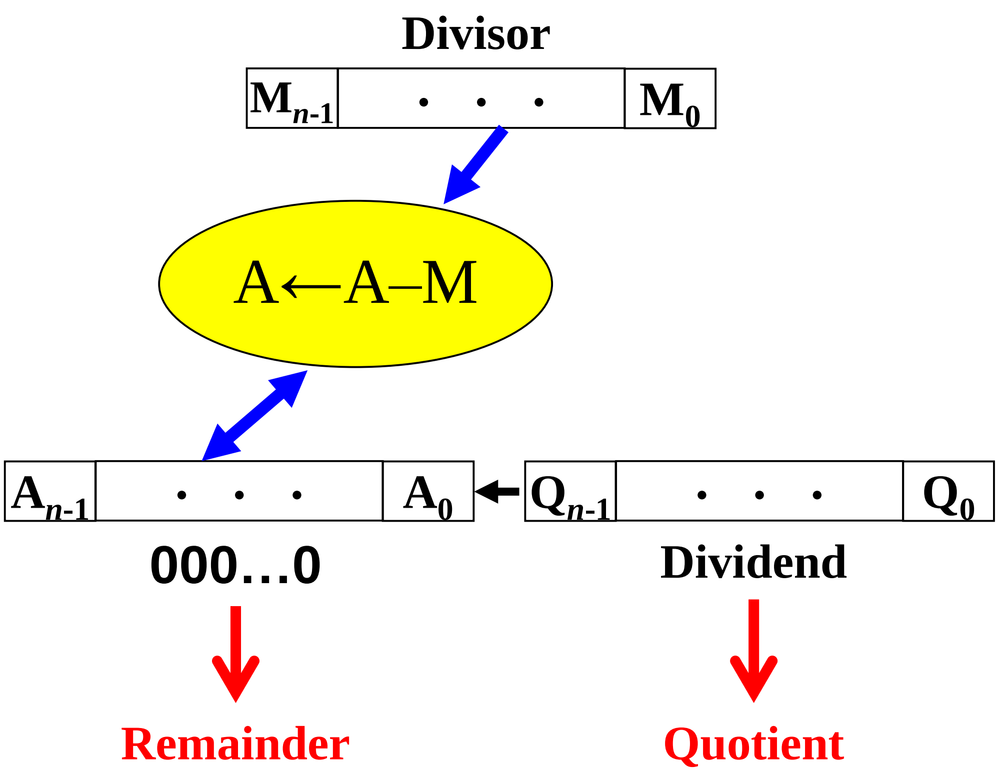
Booth's Algorithm – Pros:
- Treats +ve and –ve multipliers uniformly.
- Use fewer additions if the multiplier has large blocks of 1's.
- On average, has the same efficiency as the normal algorithm.

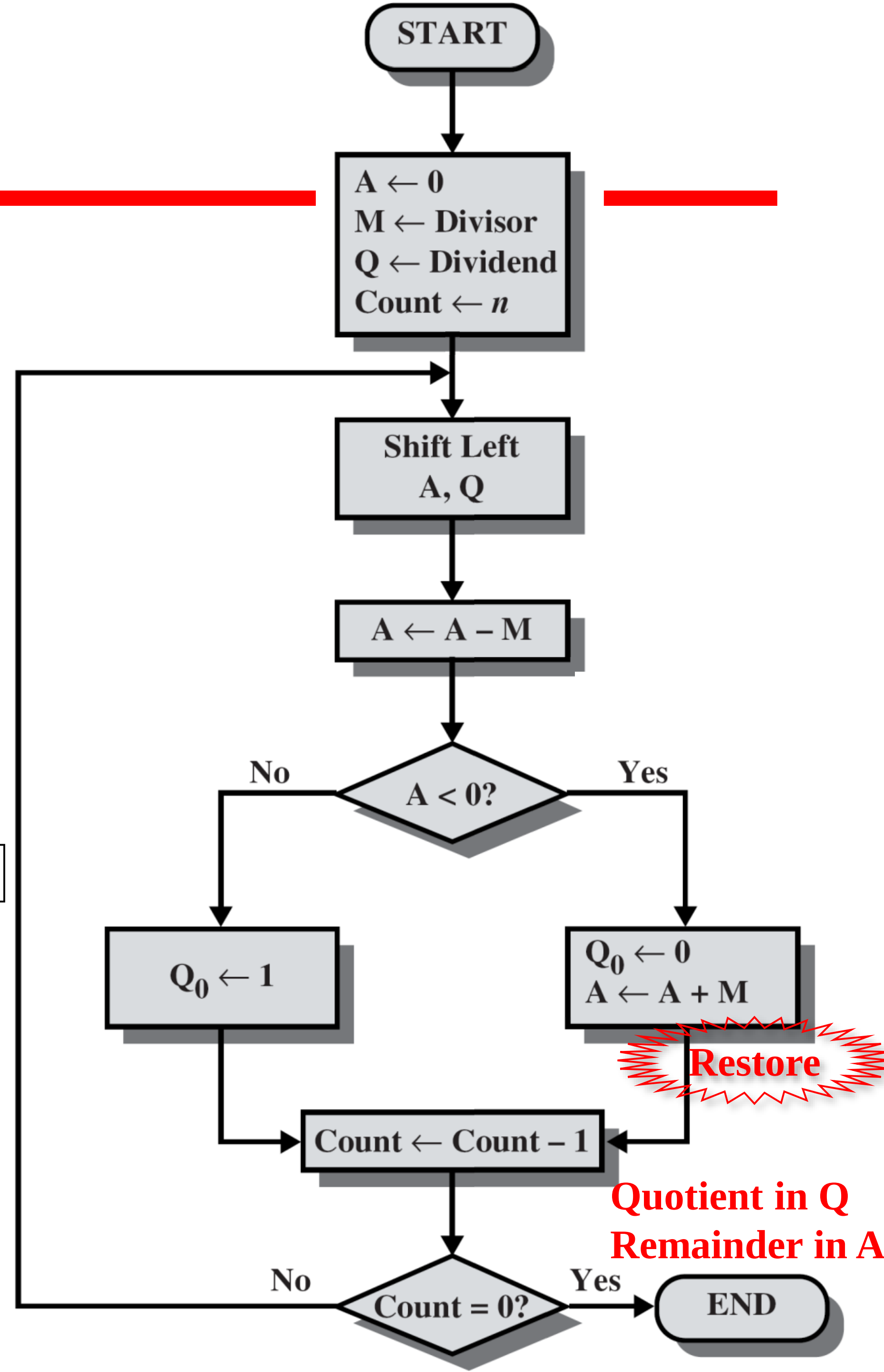
Division

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



Restoring Division (Positive Integers)



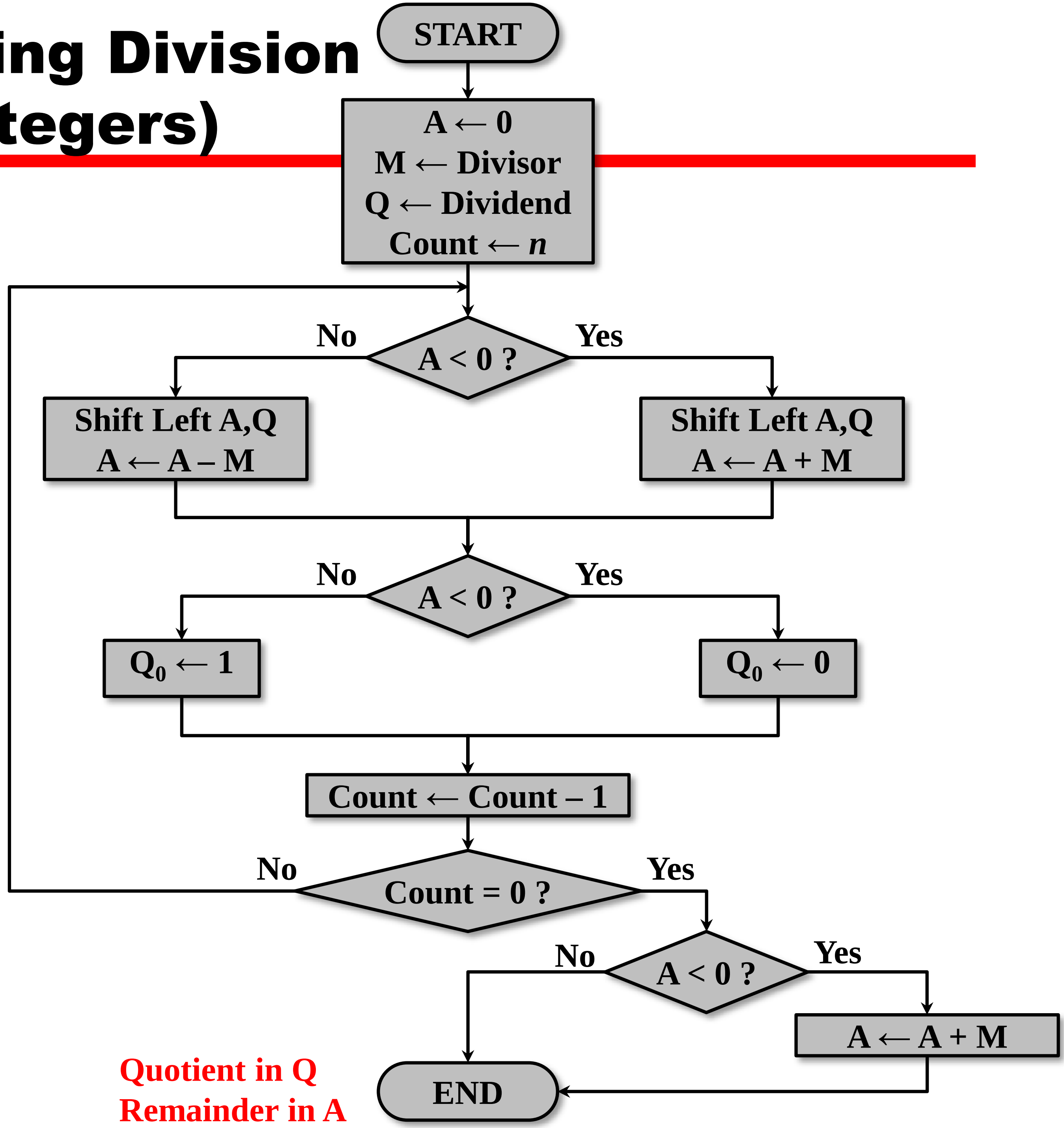


Restoring Division Example

7/3

A	Q	M = 0011	
0000	0111	Initial Value	
0000	1110	Shift	First cycle
1101		Subtract	
0000	1110	Restore	
0001	1100	Shift	Second cycle
1110		Subtract	
0001	1100	Restore	
0011	1000	Shift	Third cycle
0000		Subtract	
0000	1001	Set $Q_0 = 1$	
0001	0010	Shift	Fourth cycle
1110		Subtract	
Remainder 0001	Quotient 0010	Restore	

Non-Restoring Division (Positive Integers)



Quotient in Q
Remainder in A

Non-Restoring Division Example

7/3

A	**Q**	**M = 0011**	
0000	0111	Initial Values	
0000	111?	Shift	First cycle
1101	111?	Subtract	
1101	111**0**	$Q_0 \leftarrow 0$	
1011	11**0**?	Shift	Second cycle
1110	11**0**?	Add	
1110	11**00**	$Q_0 \leftarrow 0$	
1101	1**00**?	Shift	Third cycle
0000	1**00**?	Add	
0000	1**001**	$Q_0 \leftarrow 1$	
0001	**001**?	Shift	Fourth cycle
1110	**001**?	Subtract	
1110	**0010**	$Q_0 \leftarrow 0$	
0001	**0010**	Add	
Remainder	Quotient		

Dealing with Signed Integers

- Given a dividend (D) and divisor (V) where both are signed integers in the 2's complement representation.
- Division can be carried out as follows:
  1. Convert D & V to their absolute (+ve) values |D| & |V|.
  2. Run either restoring or non-restoring division on |D| & |V| to obtain the quotient (Q) and the remainder (R).
  3. Adjust the sign of Q and R (by 2's complementation where needed) according to the following rules:
     - sign(Q) = sign(D) X sign(V)
     - sign(R) = sign(D)

Reading Material

- Stallings, Chapter 10:
  - Pages 331 – 341