Computer Organization and Architecture CEC 470

Module 02 : Arithmetic and Logic Part 2(Multiplication & Division)

(Ch 10, Ch 11, Ch 12)

Last week

- Number system (decimal, binary, hexadecimal)
- Signed and unsigned representation
  - Two’s compliment
- Half adder, full adder, carry look ahead adder, subtraction circuit

Multiplication

- More complicated than addition
  - A straightforward implementation will involve shifts and adds
- More complex operation can lead to
  - More area (on silicon) and/or
  - More time (multiple cycles or longer clock cycle time)
- Let’s begin from a simple, straightforward method

In-class activity Q1

Use Twos compliment to represent the following decimal numbers

a) -7
b) -8

Multiplication: pencil & paper approach

	5	(Multiplicand)
X	3	(Multiplier)
	15	(Product)

Repeated Addition

5+5+5 = 15

Is it efficient? NO (Very Slow)

E.g. 255 x 255 would require 255 additions or steps

	5	$(0\ 1\ 0\ 1)_2$
X	3	$(0\ 0\ 1\ 1)_2$
	15	0 1 0 1

```
      0 1 0 1
    0 1 0 1
  0 0 0 0
0 0 0 0
-------------
0 0 0 1 1 1 1
```

15_{10}

Is it efficient? YES (FAST)

How many steps? 4 Steps for 4 bit numbers

Multiplication: pencil & paper approach

```
          01010010  (multiplicand)
        x 01101101  (multiplier)
      ----------------------------
          01010010
         00000000
        01010010
       01010010
      00000000
     01010010
    01010010
   00000000
  --------------------------------
   0100010111101010
```

Partial product

How many steps for 8 bits numbers? 8 steps as compared to 255 steps using repeated addition

Implementation point of view:

1) Fast
2) Is it complex ? No (we are not at all multiplying)
3) No. of steps = No. of bits in the multiplier
4) Examine the bit of the multiplier (LSB)
    1) If bit is 1, the partial product is multiplicand itself
    2) If bit is 0, the partial product is 0
5) Add all the partial products

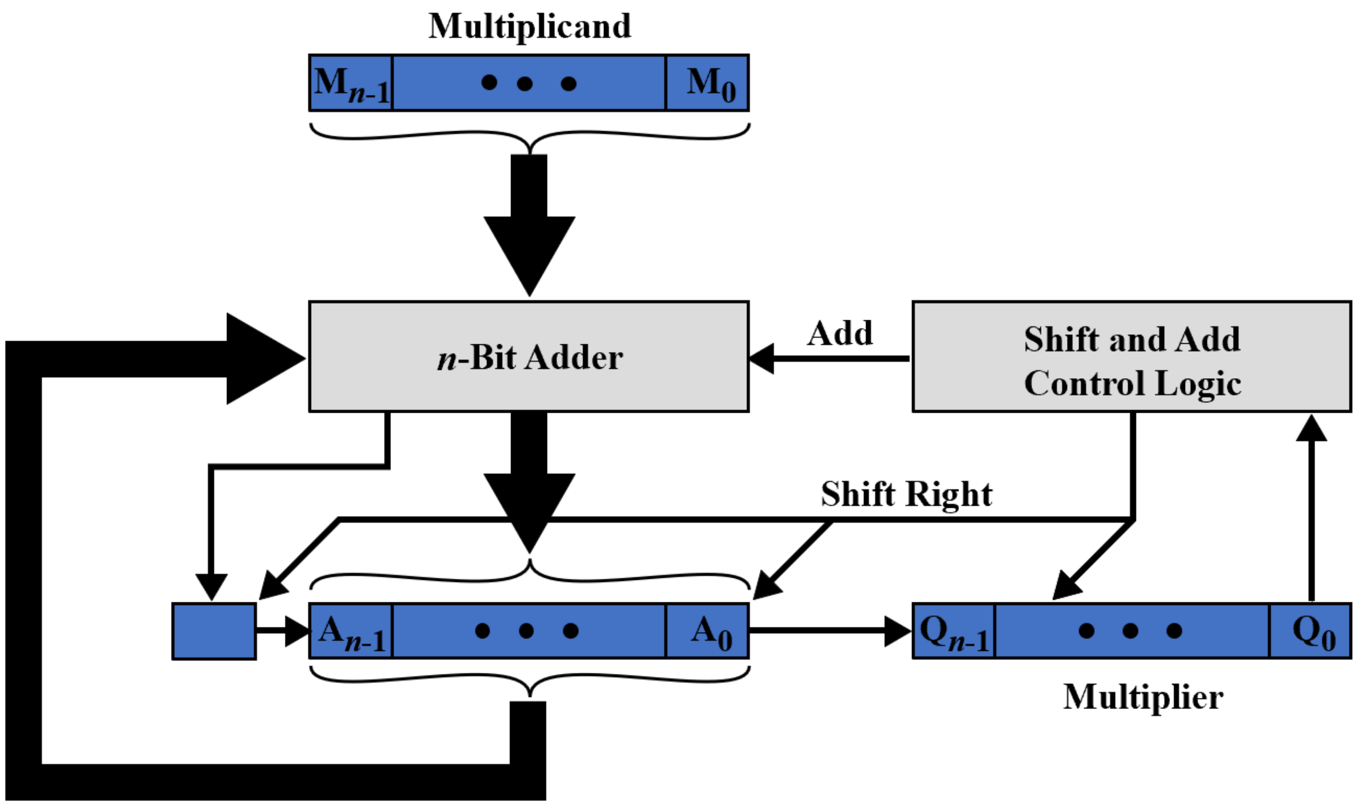
No multiplication
Only arithmetic done is ADD

Multiplication: computerized multiplication

- Running sum on the partial product
- For each 1 on the multiplier: add and shift
- For each 0 on the multiplier : shift

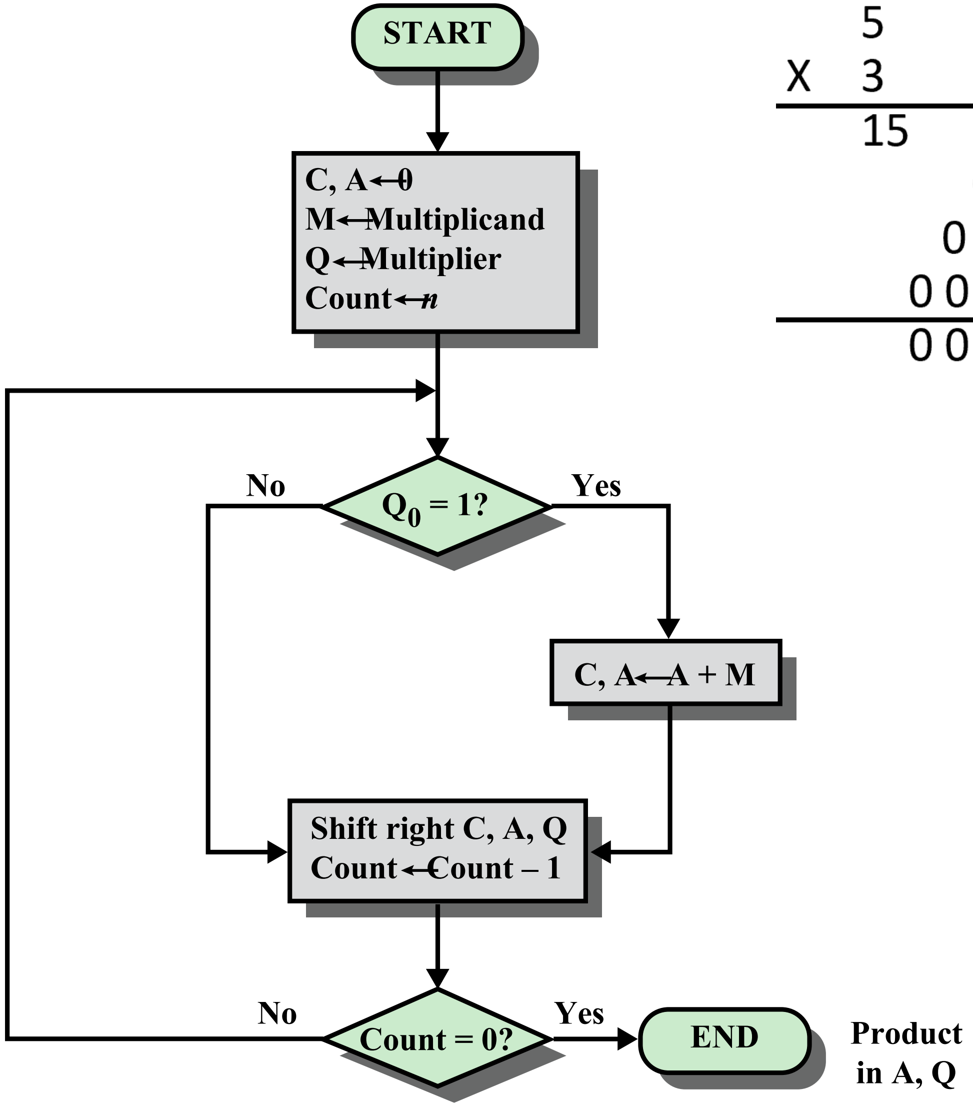
2

4-bit number, No. of values = 2^4
4-bit X 4 bit = 2^4 x 2^4 = 2^8



(a) Block Diagram

Multiplication: computerized multiplication



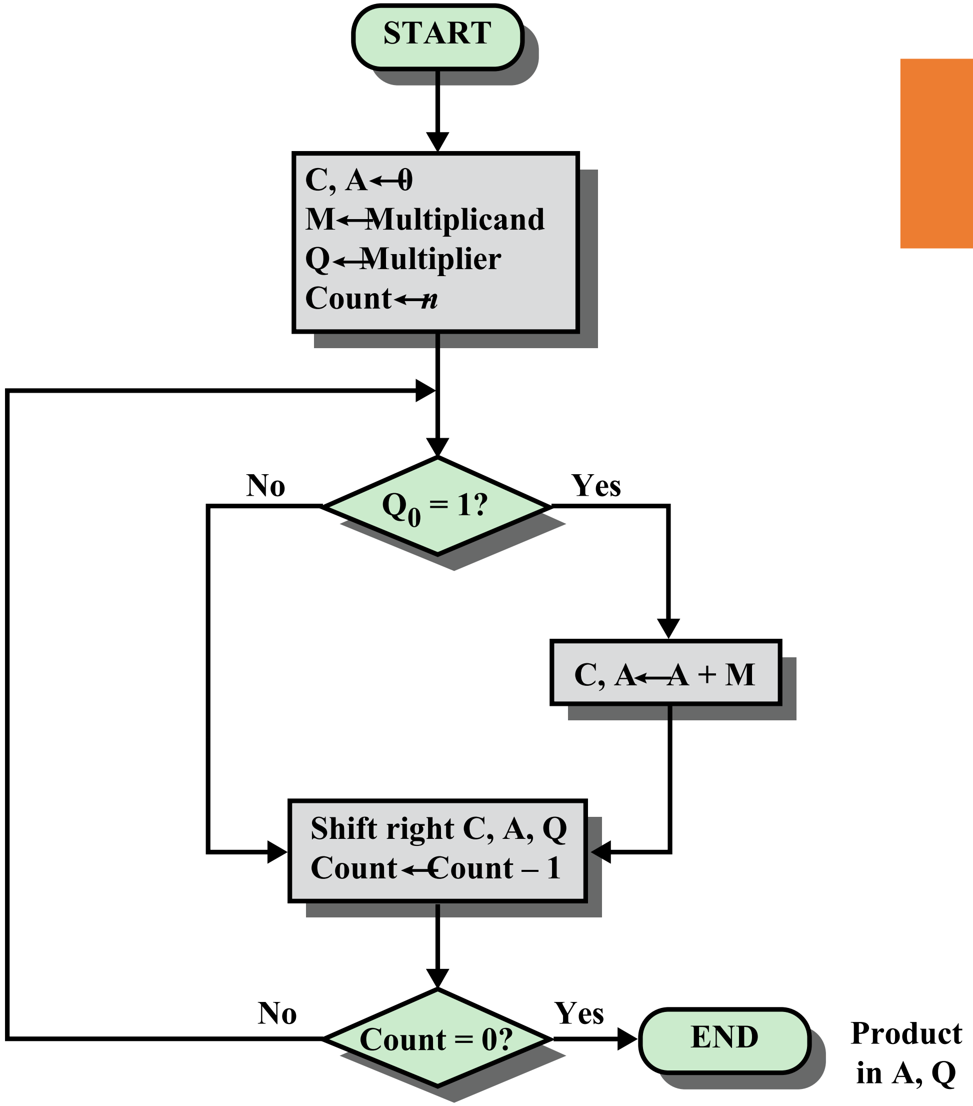
	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1

Count = 4

M: 0 1 0 1

C	A	Q
0	0 0 0 0	0 0 1 1

Multiplication: computerized multiplication



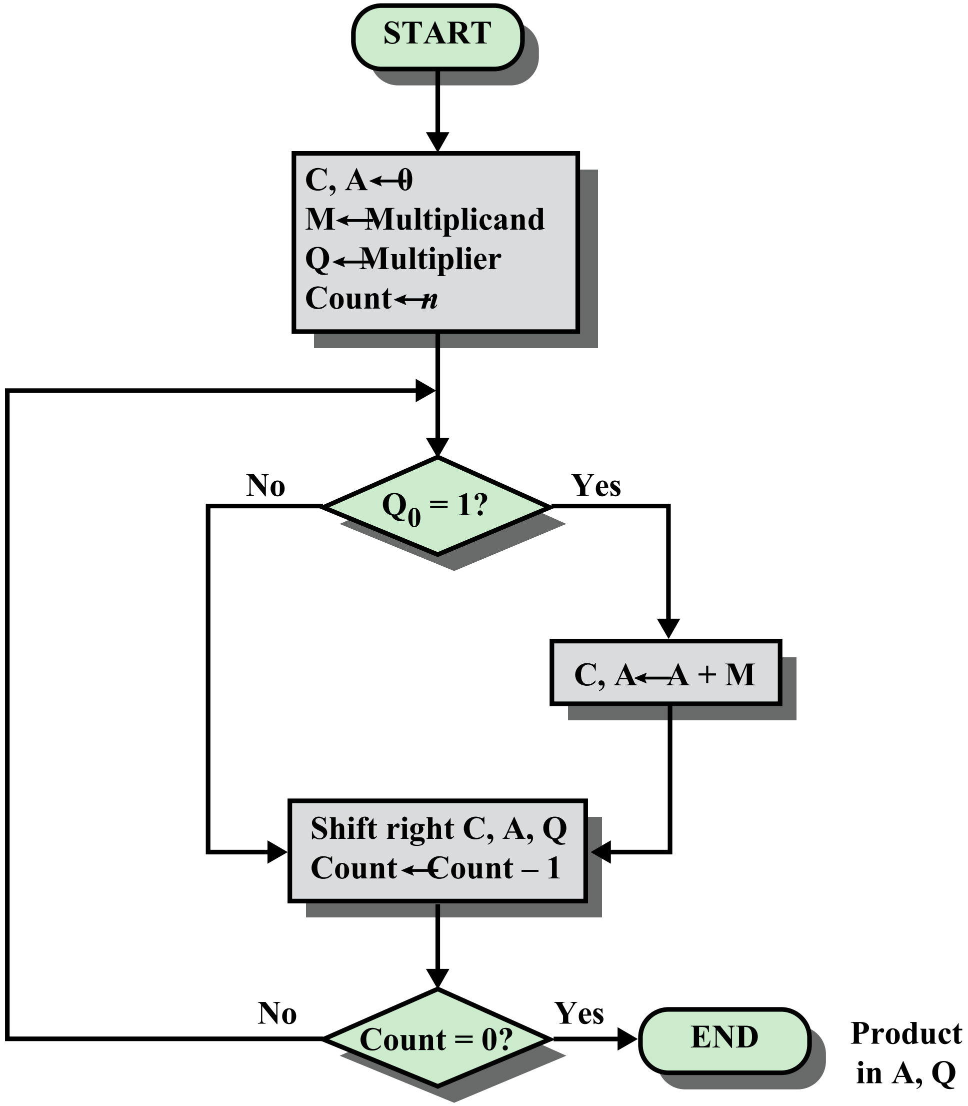
0 1 0 1
0 0 0 0
0 1 0 1

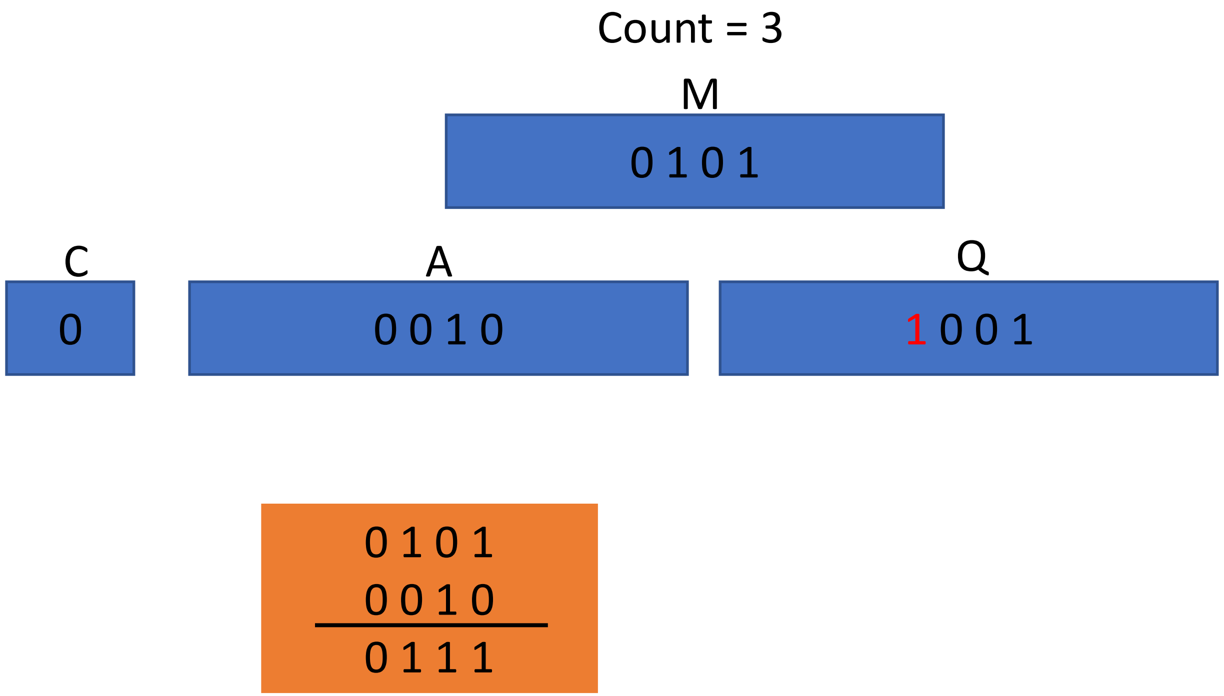
Count = 3

M: 0 1 0 1

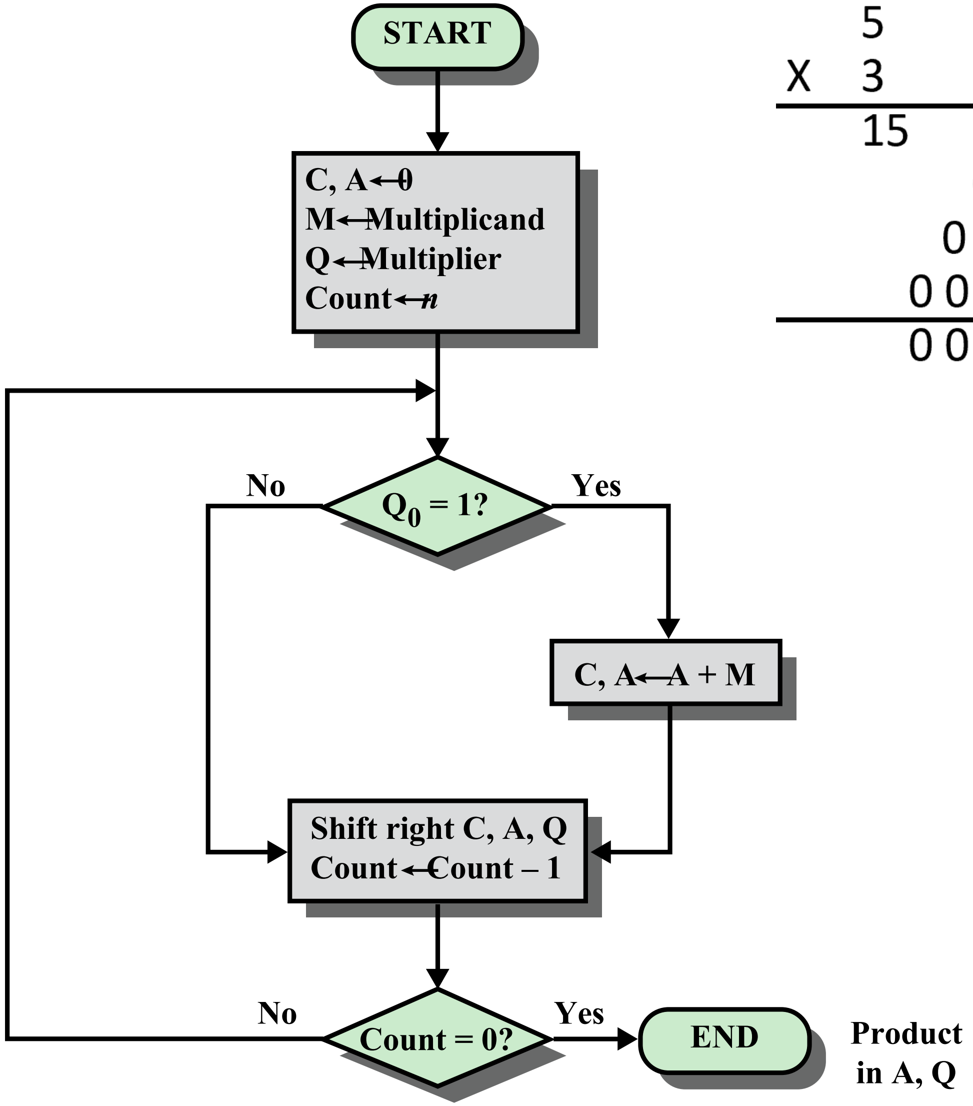
	C	A	Q
Add	0	0 1 0 1	0 0 1 1
Shift Right	0	0 0 1 0	1 0 0 1

Multiplication: computerized multiplication

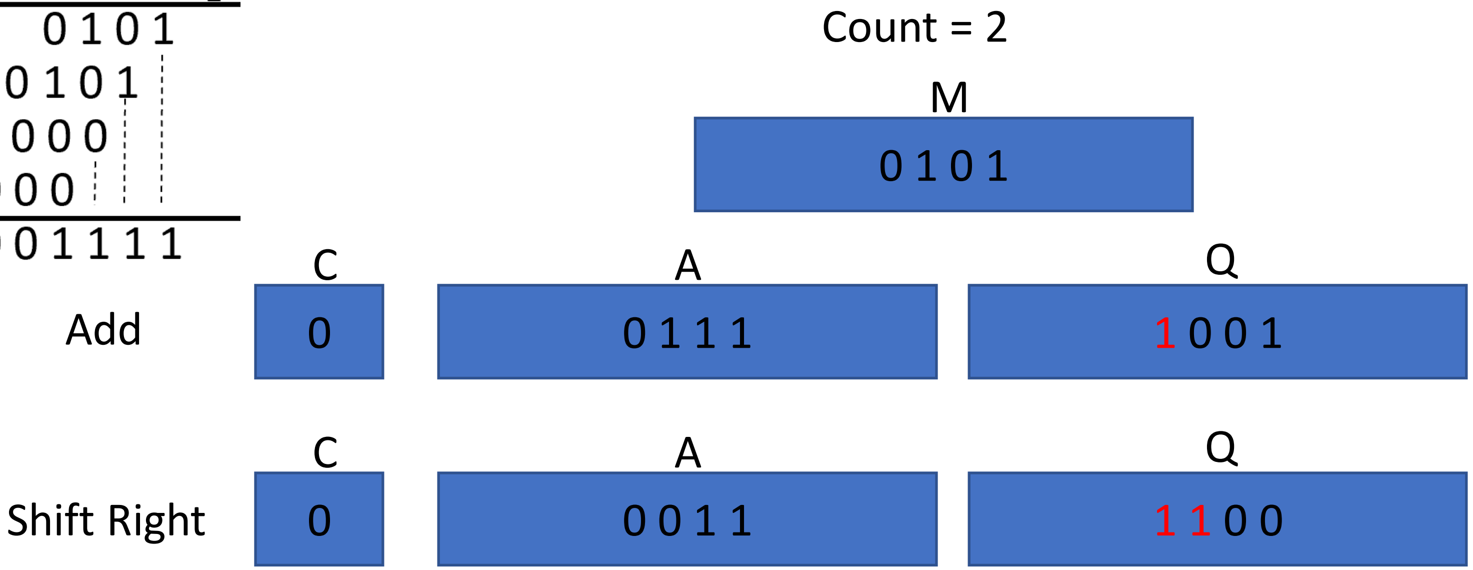




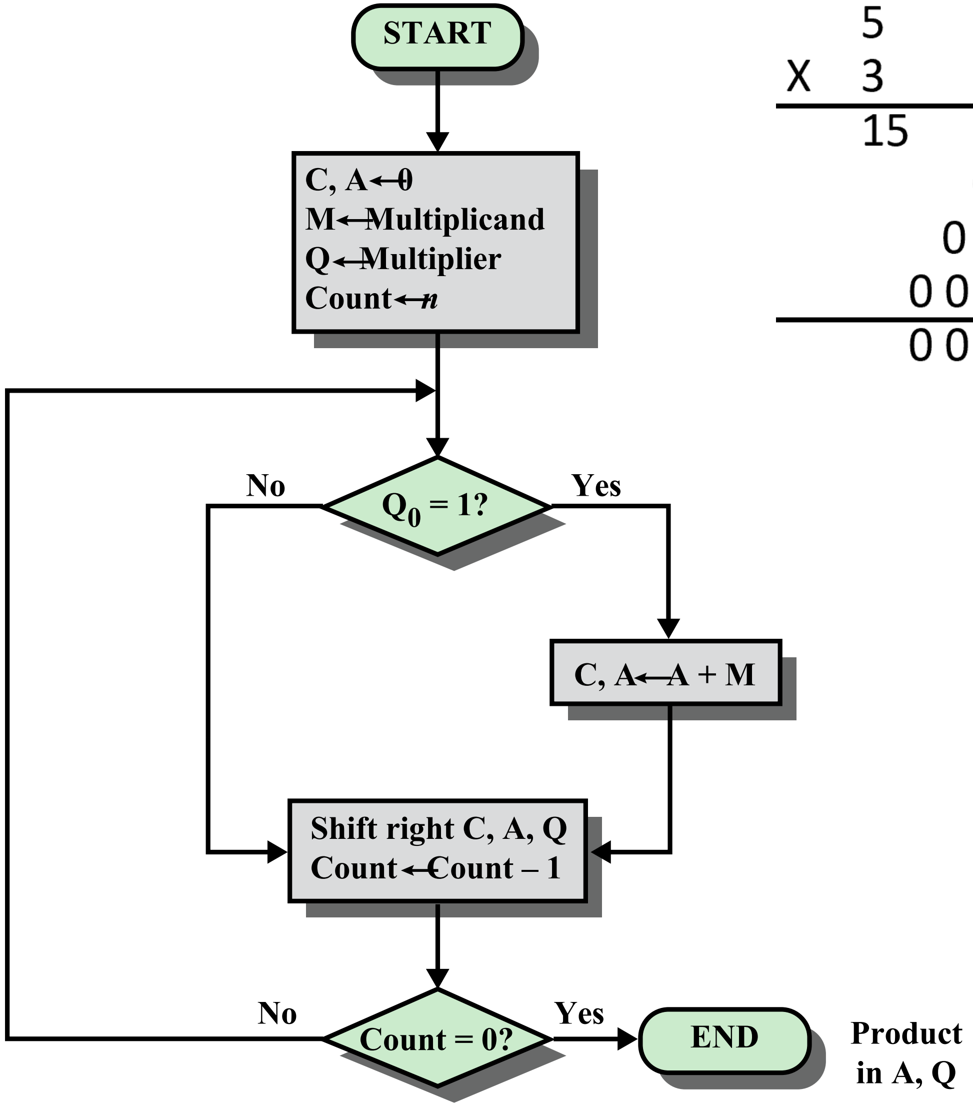
Multiplication: computerized multiplication



	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1

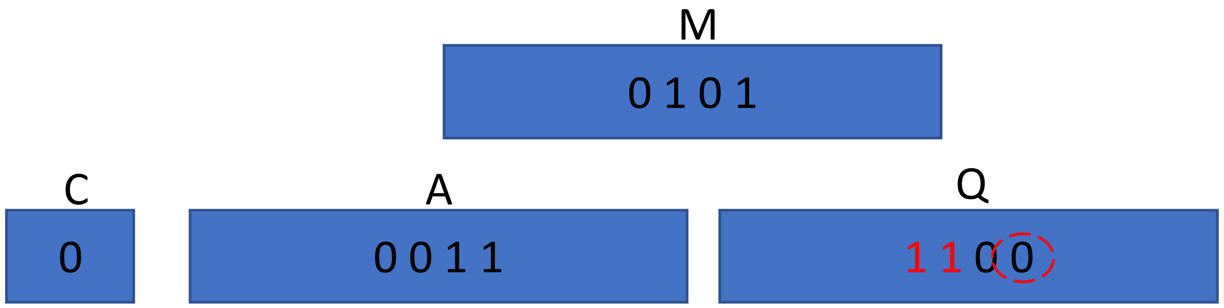


Multiplication: computerized multiplication

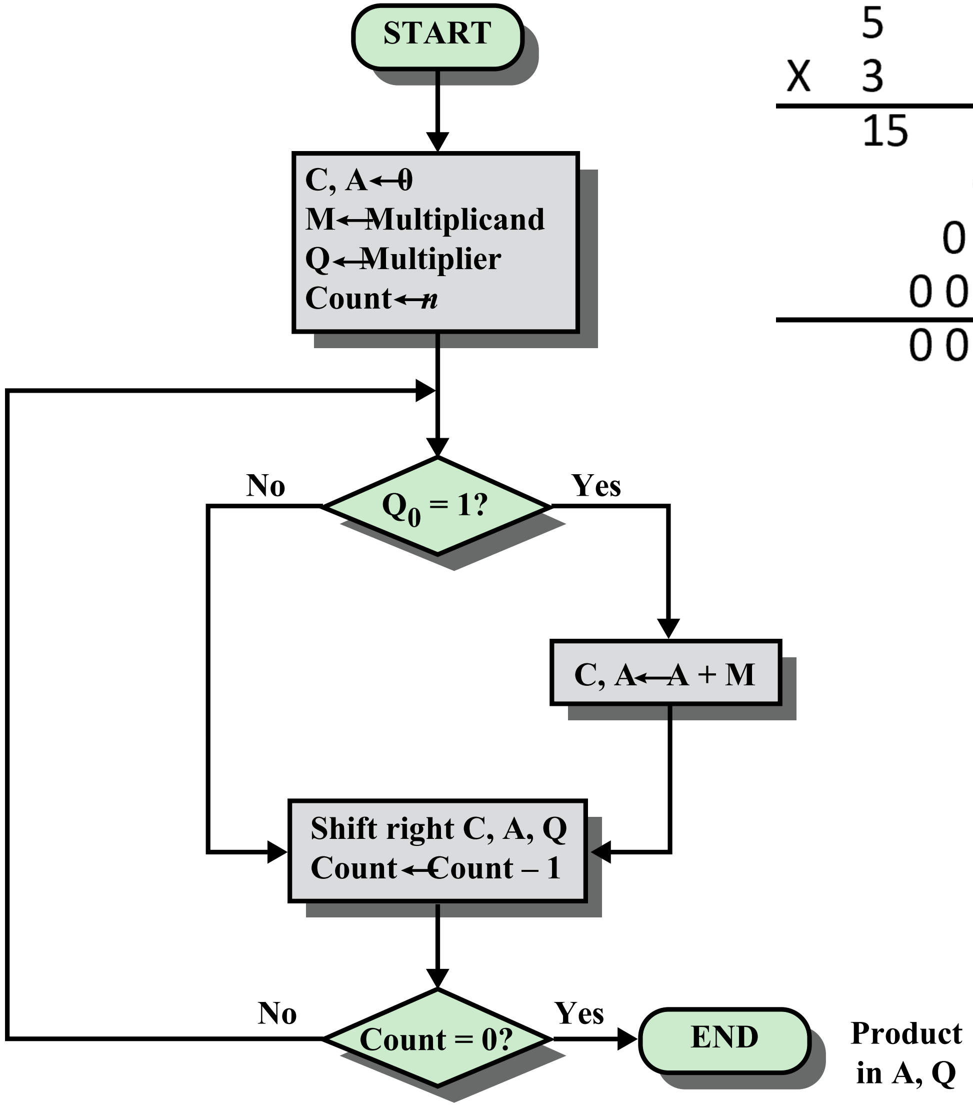


	5	$(0\ 1\ 0\ 1)_2$
X	3	$(0\ 0\ 1\ 1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1

Count = 2

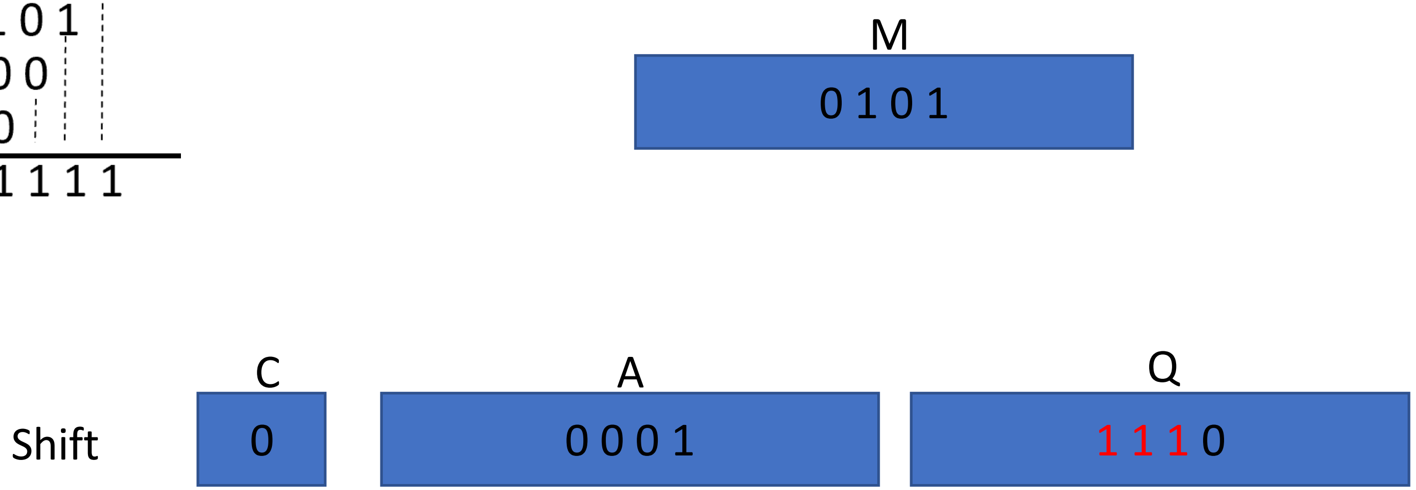


Multiplication: computerized multiplication

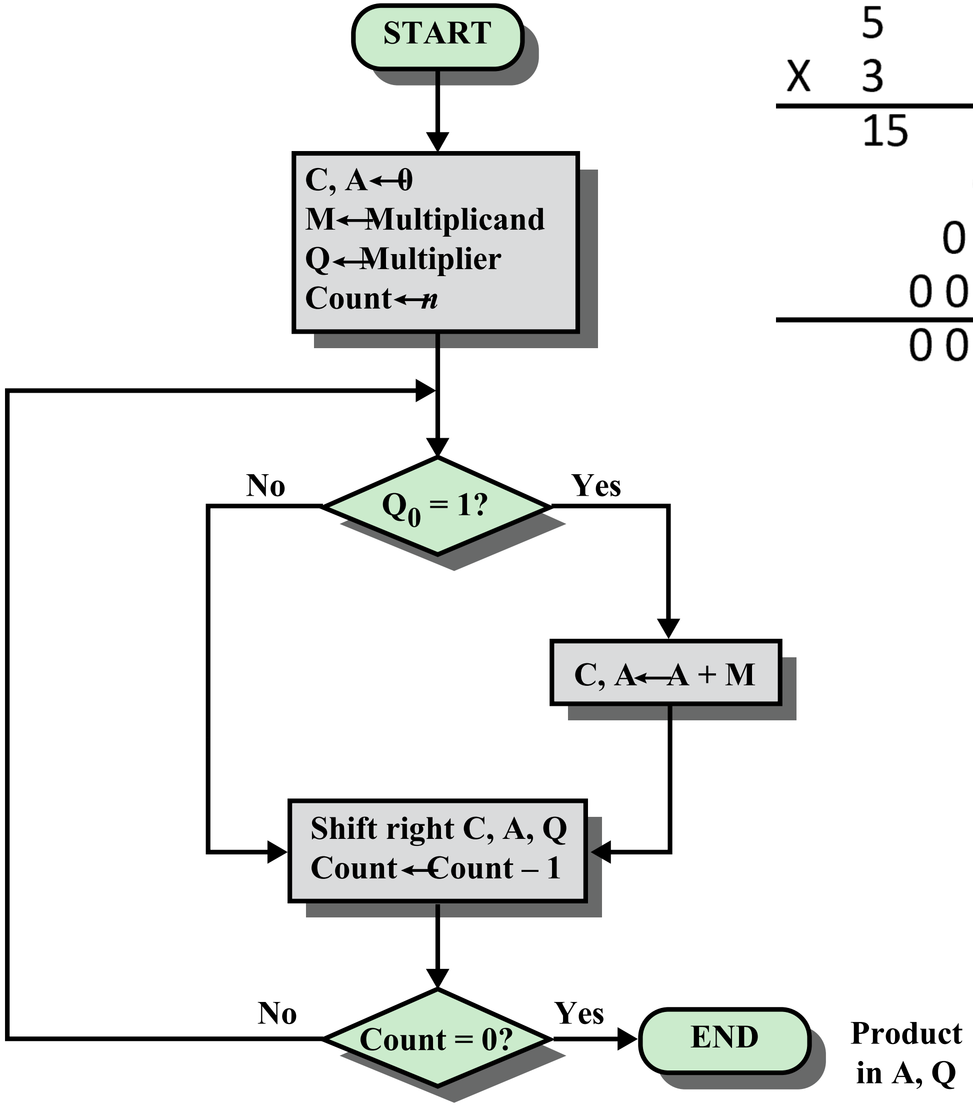


	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1

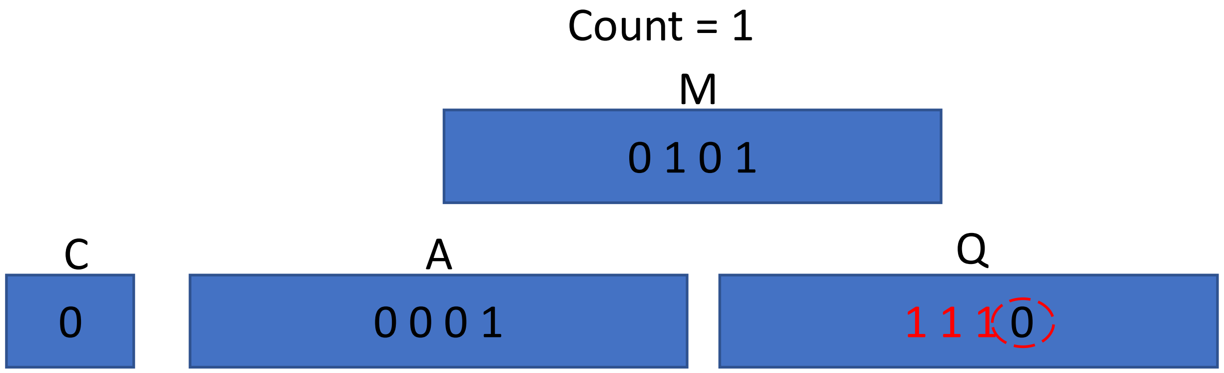
Count = 1



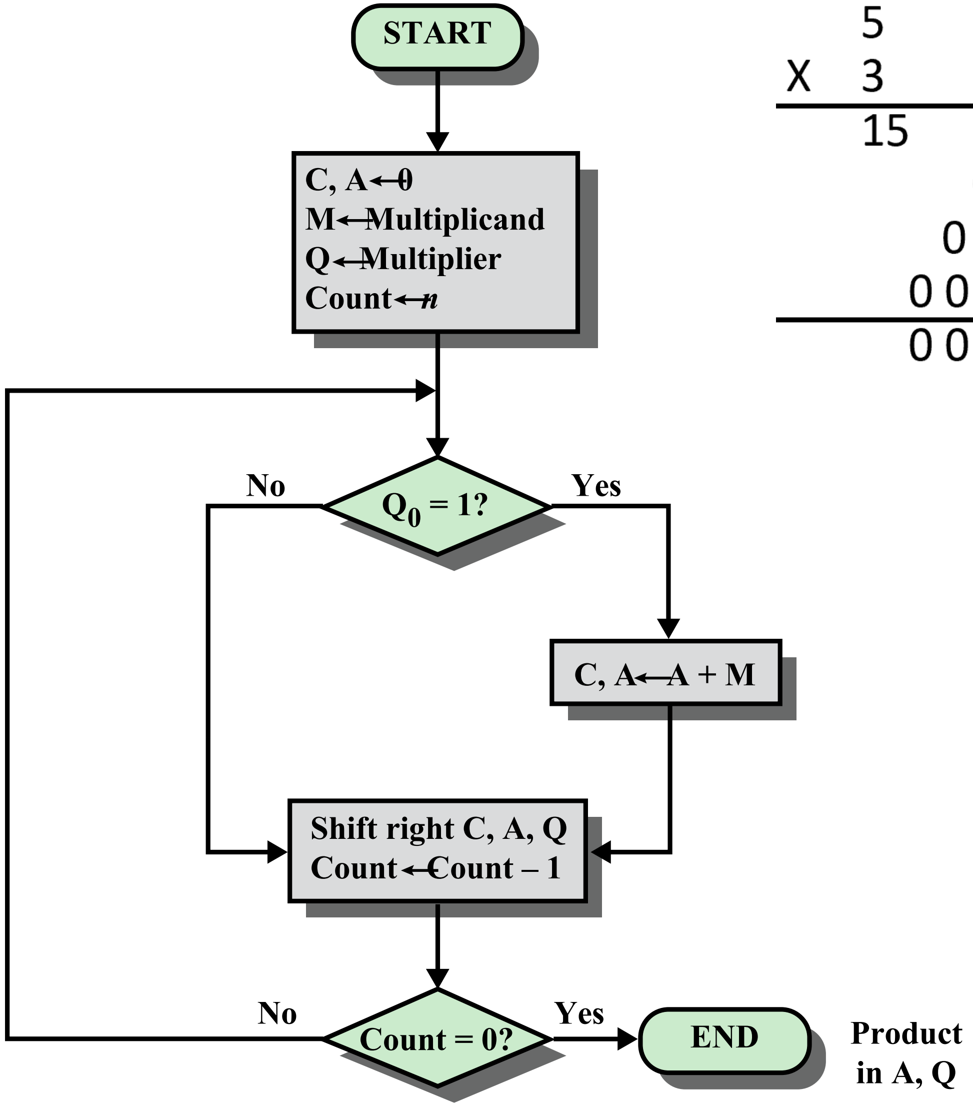
Multiplication: computerized multiplication



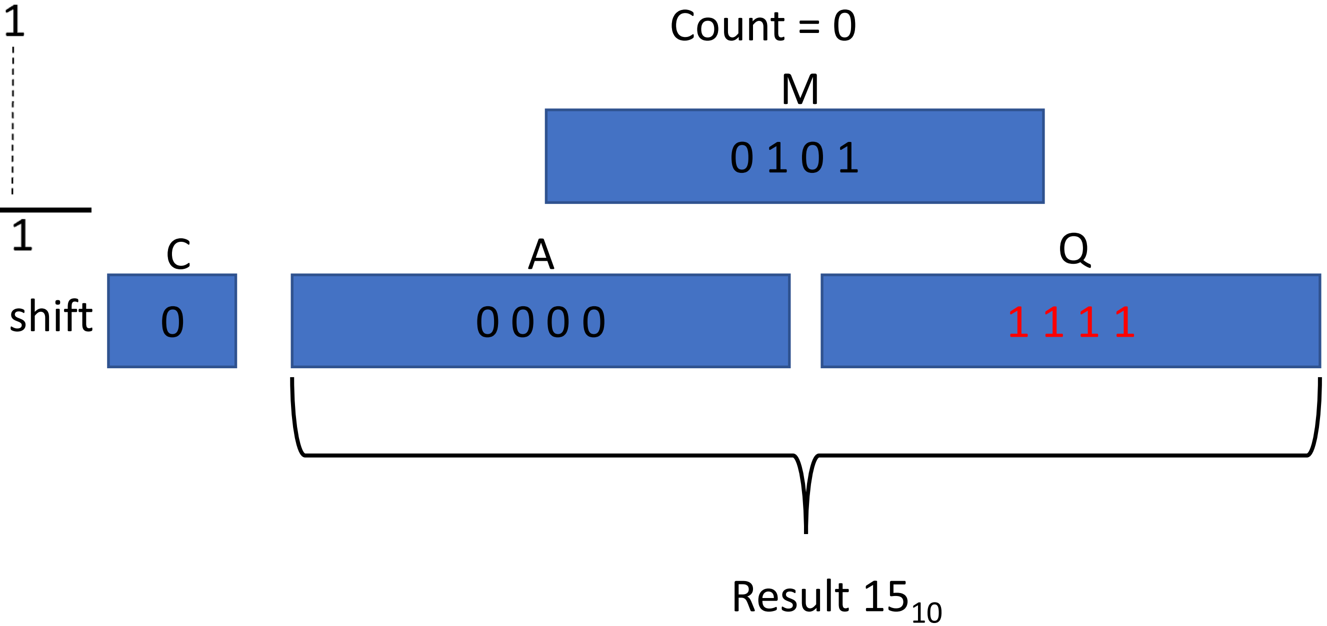
	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1



Multiplication: computerized multiplication



	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1



Drawback?

	5	$(0\,1\,0\,1)_2$
X	3	$(0\,0\,1\,1)_2$
	15	0 1 0 1
		0 1 0 1
		0 0 0 0
		0 0 0 0
		0 0 0 1 1 1 1

- **This method works for only unsigned numbers (strictly positive)**

FAILS for Signed Numbers

	5	$(0\,1\,0\,1)_2$
X	- 3	$(1\,1\,0\,1)_2$
	-15	0 1 0 1
		0 0 0 0
		0 1 0 1
		0 1 0 1
		1 0 0 0 0 0 1

?

Solution…

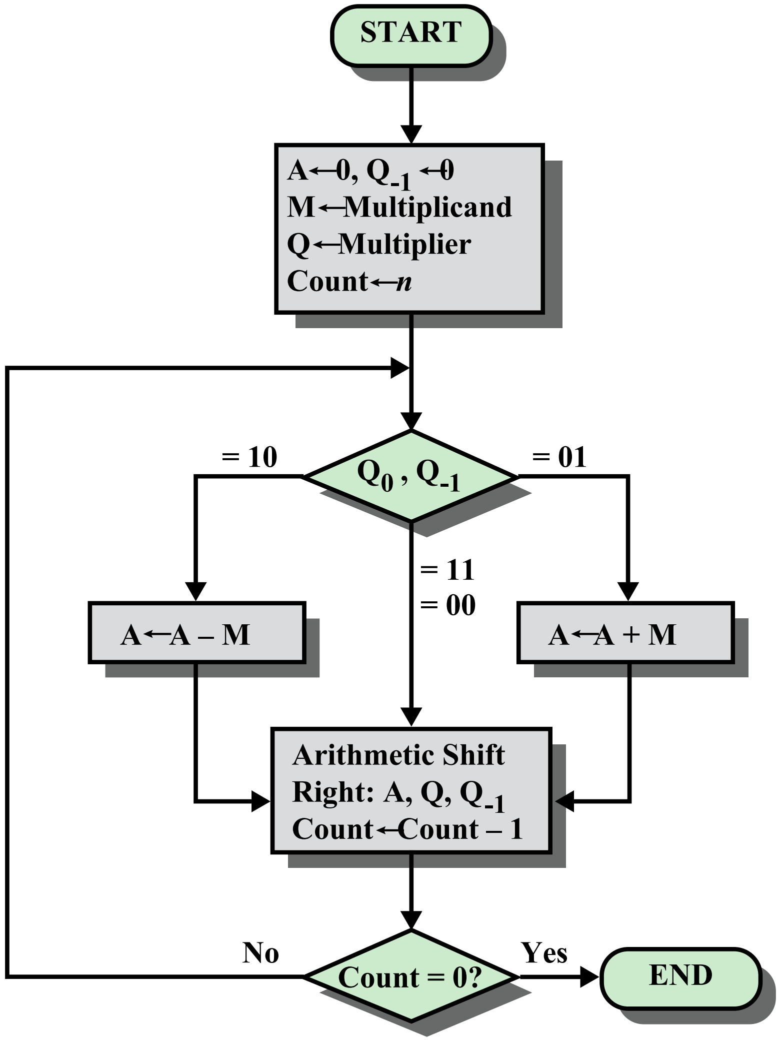
- Convert multiplier and multiplicand to unsigned integers
- Multiply
- If original signs differed, negate result
- But there are more efficient ways

Booths algorithm for signed multiplication

- Bits of the multiplier are scanned one at a a time (the current bit Q_0)
- As bit is examined the bit to the right is considered also (the previous bit Q_{-1})
- Then:

  00: Middle of a string of 0s, so no arithmetic operation.
  01: End of a string of 1s, so add the multiplicand to the left half
  of the product (A).
  10: Beginning of a string of 1s, so subtract the multiplicand from
  the left half of the product (A).
  11: Middle of a string of 1s, so no arithmetic operation.

- Then shift A, Q, bit Q_{-1} right one bit using an arithmetic right shift
- In an **arithmetic shift, the msb remains unchanged**



Right shift vs arithmetic right shift

Example1 :	1 0 0 1
Right shift :	0 1 0 0
Ar. Right shift:	1 1 0 0

Example2 :	0 1 0 1
Right shift :	0 0 1 0
Ar. Right shift:	0 0 1 0

In-class activity Q2

Write the one-bit *right shift and arithmetic right shift* for the following:

1. 11111111
2. 10000001

Example of booths algorithm

Multiply : 7 X 3
Multiplicand = 7 = 0 1 1 1
Multiplier = 3 = 0 0 1 1

A 0 0 0 0	Q(3) 0 0 1 1	Q_{-1} 0	M (7) 0 1 1 1	Initial values

Example of booths algorithm

Multiply : 7 X 3
Multiplicand = 7 = 0 1 1 1
Multiplier = 3 = 0 0 1 1

A	Q(3)	Q_{-1}	M (7)	Initial values
0000	0011	0	0111	
1001 1100	0011 1001	0 1	0111 0111	A<--A - M Shift First Cycle

Example of Booths Algorithm

Multiply : 7 X 3
Multiplicand = 7 = 0 1 1 1
Multiplier = 3 = 0 0 1 1

A 0000	Q(3) 0011	Q_{-1} 0	M (7) 0111	Initial values
1001 1100	0011 1001	0 1	0111 0111	A<--A - M Shift First Cycle
1110	0100	1	0111	Shift Second Cycle

Example of Booths Algorithm

Multiply : 7 X 3
Multiplicand = 7 = 0 1 1 1
Multiplier = 3 = 0 0 1 1

A 0000	Q(3) 0011	Q_{-1} 0	M (7) 0111	Initial values	
1001 1100	0011 1001	0 1	0111 0111	A<--A - M Shift	First Cycle
1110	0100	1	0111	Shift	Second Cycle
0101 0010	0100 1010	1 0	0111 0111	A <--A + M Shift	Third Cycle

Example of Booths Algorithm

Multiply : 7 X 3 = 21

Multiplicand = 7 = 0 1 1 1

Multiplier = 3 = 0 0 1 1

A 0000	Q(3) 0011	Q_{-1} 0	M (7) 0111	Initial values
1001 1100	0011 1001	0 1	0111 0111	A<--A - M Shift — First Cycle
1110	0100	1	0111	Shift — Second Cycle
0101 0010	0100 1010	1 0	0111 0111	A <--A + M Shift — Third Cycle
0001	0101	0	0111	Shift — Fourth Cycle

$\underbrace{0001\ 0101}_{21_{10}}$

In-class activity Q3

Use Booths Algorithm:

-3 x 2