Integer Arithmetic

Muhammad HabibUllah
habib.wattoo@nu.edu.pk

Book Chapter

- “Computer Organization and Architecture”
- Author “William Stallings”
- 8th Edition
- Chapter 9
  - Section 9.1
  - Section 9.2
  - Section 9.3

Arithmetic and Logic Unit (ALU)

- Performs the calculations
- Everything else in the computer is there to serve this unit
- Handles integers
- May handle floating point numbers
- May have a separate FPU

ALU Inputs and Outputs

Integer Representation

- Positive integers are stored in computer in binary format
  - e.g. 43 = 00101011
- Minus sign and period cannot be stored in binary format
- Signed numbers can be represented by
  - Sign-Magnitude Representation
  - Twos Complement Representation

Sign-Magnitude Representation

- MSB represents the sign of the integer
  - 0 for positive integers
  - 1 for negative integers
- Remaining bits represent the magnitude of the number

1001 1101 → MSB = 1 so number is negative

001 1101 = $2^4+2^3+2^2+2^0$ = 16+8+4+1 = 29

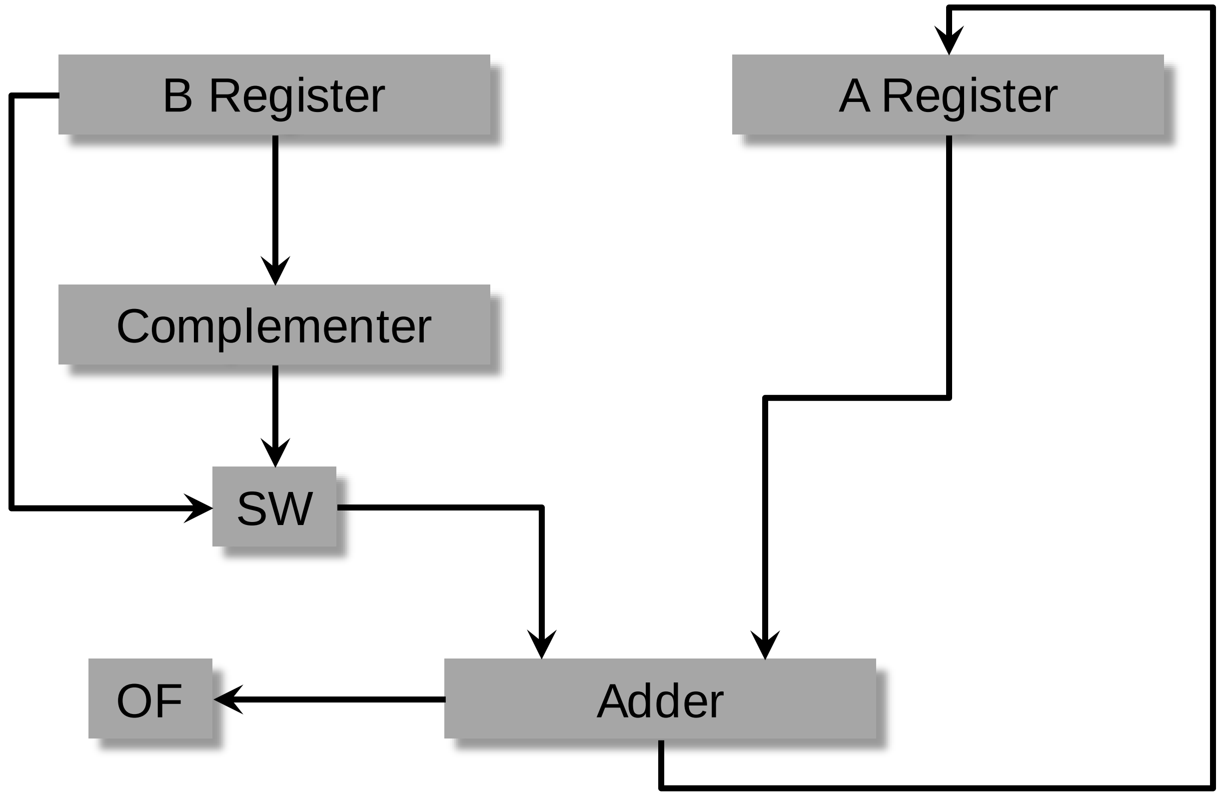
Since MSB = 1, so 1001 1101 represents –29

Addition and Subtraction

- Normal binary addition
- Monitor sign bit for overflow

- For subtraction, take twos complement of the subtrahend and add to other operand
- We can perform subtraction by using the addition operation

Block Diagram for Adder/Subtractor



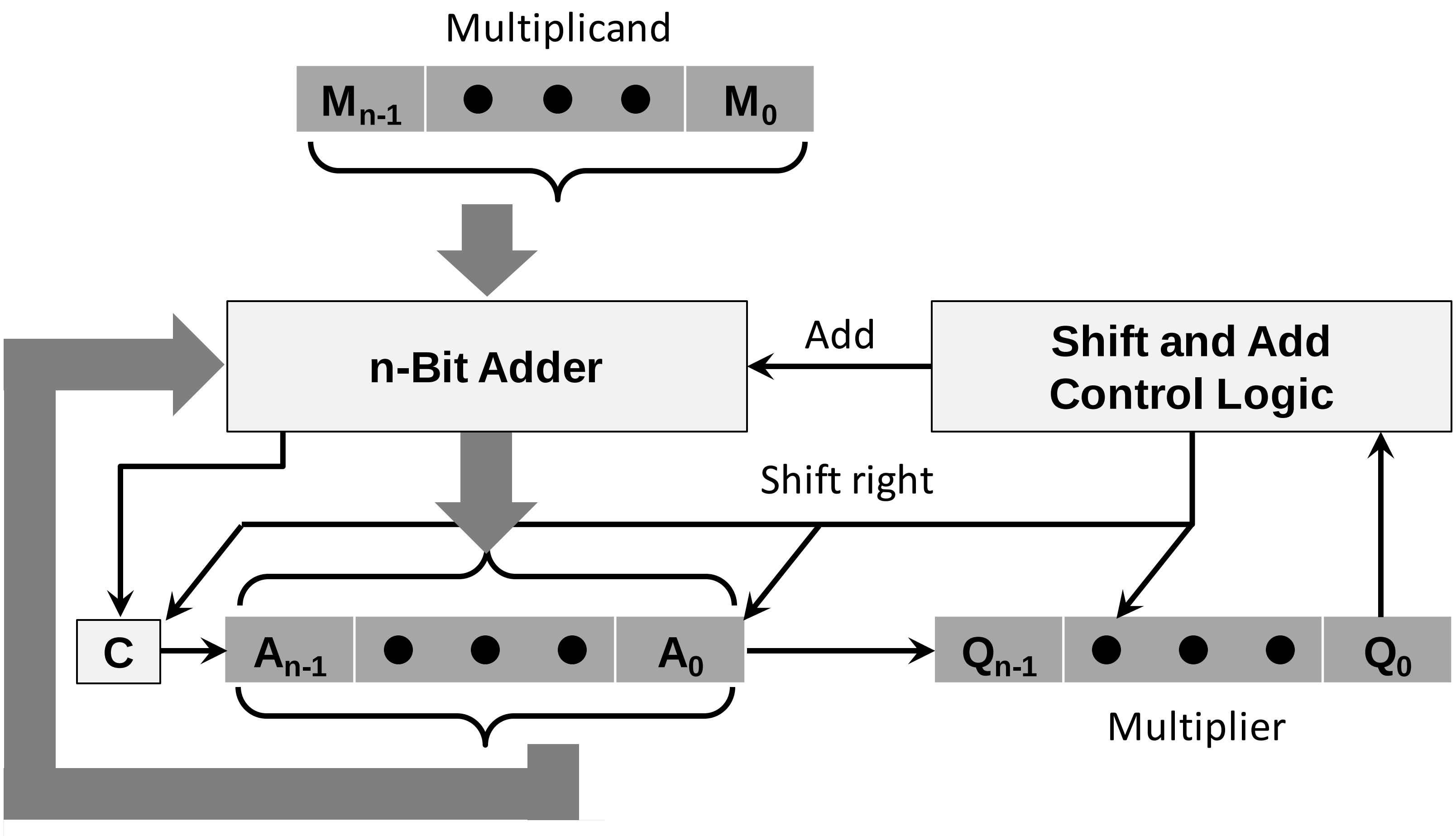
Multiplication

- Complex operation as compared to addition/subtraction
- In simple paper and pencil approach
  - Work out partial product for each digit
  - Take care of place of values in partial product
  - Add partial products to get the final product

Unsigned Binary Multiplication (1/3)

```
        1 0 1 1    ←  Multiplicand (11)
      X 1 1 0 1    ←  Multiplier (13)
      ---------
        1 0 1 1
      0 0 0 0
    1 0 1 1        ←  Partial Product
  1 0 1 1
---------------
1 0 0 0 1 1 1 1    ←  Product (143)
```

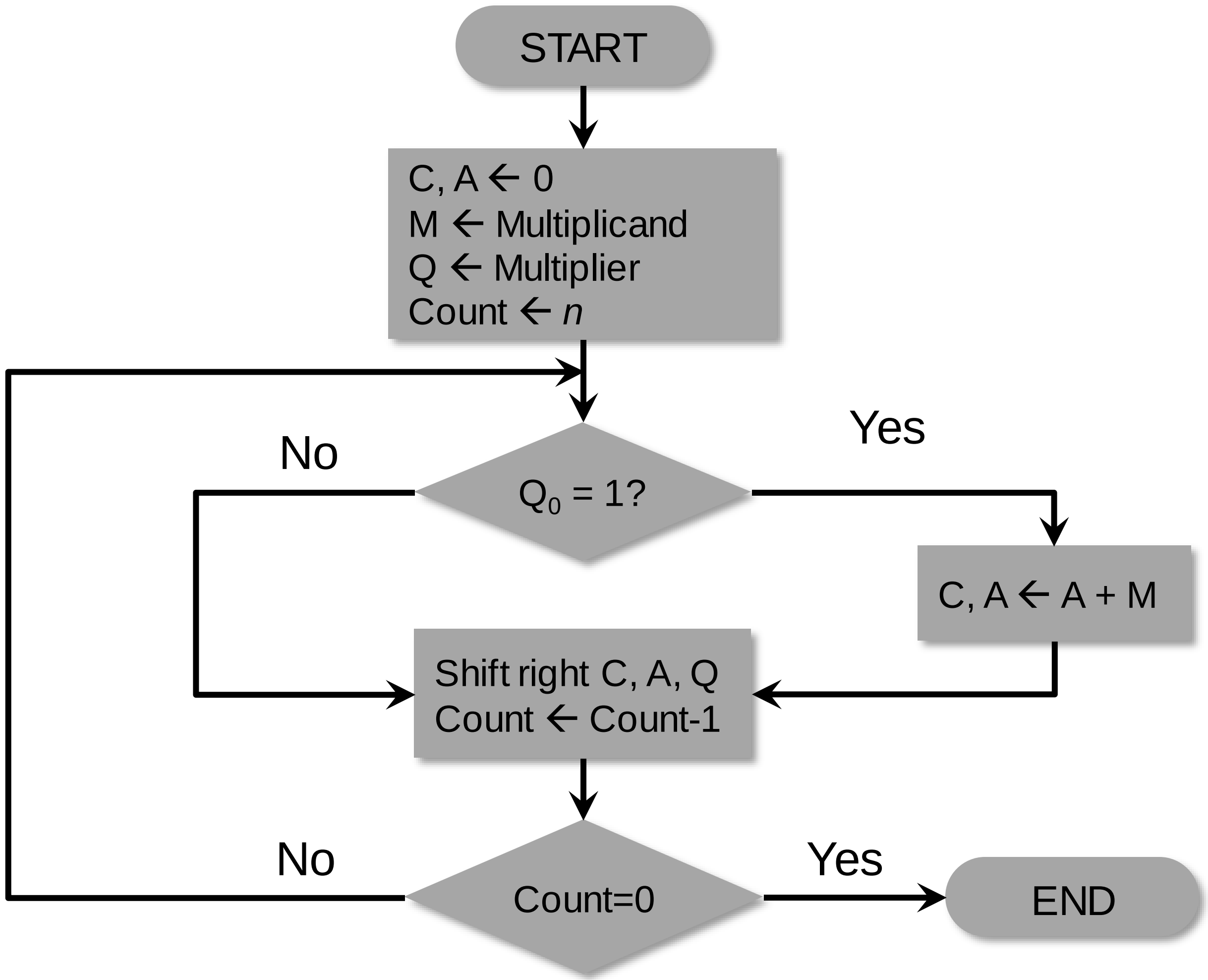
Unsigned Binary Multiplication (2/3)



Unsigned Binary Multiplication (3/3)

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	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

Flowchart for Unsigned Binary Mul.



Multiply Negative Numbers

- Three possibilities to multiply negative numbers
- Solution 1
  - Convert to positive if required
  - Multiply as previous slides like paper and pencil method
  - If signs of original numbers were different, negate answer
- Solution 2
  - Twos complement multiplication
- Solution 3
  - Booth's algorithm

Two's Complement Multiplication

- In binary multiplication, multiplicand is multiplied either by 1 or 0
- Multiplication of a binary number by 2^n means shifting the multiplicand n bits left
- Partial product is written as a 2n-bit number
- Sign-bit of partial product is extended till end
- Will not work if multiplier is negative

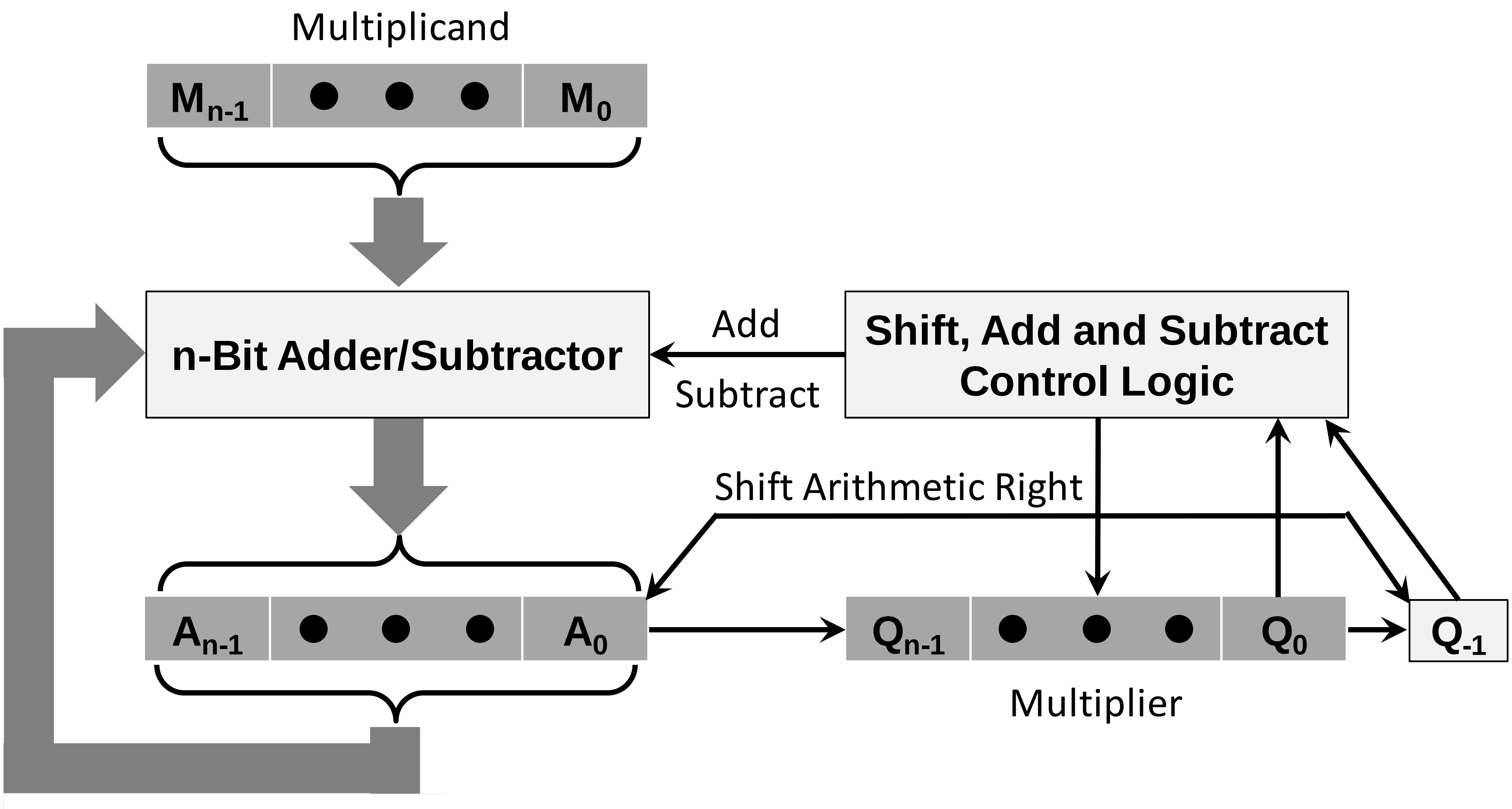
Two's Complement Multiplication

1 0 1 1	←	Multiplicand (not 11 but –5)
X 0 1 0 1	←	Multiplier (+5)
1 1 1 1 1 0 1 1		
0 0 0 0 0 0 0 0	←	Partial Product
1 1 1 0 1 1 0 0		
1 1 1 0 0 1 1 1	←	Product (not 55 but -25)

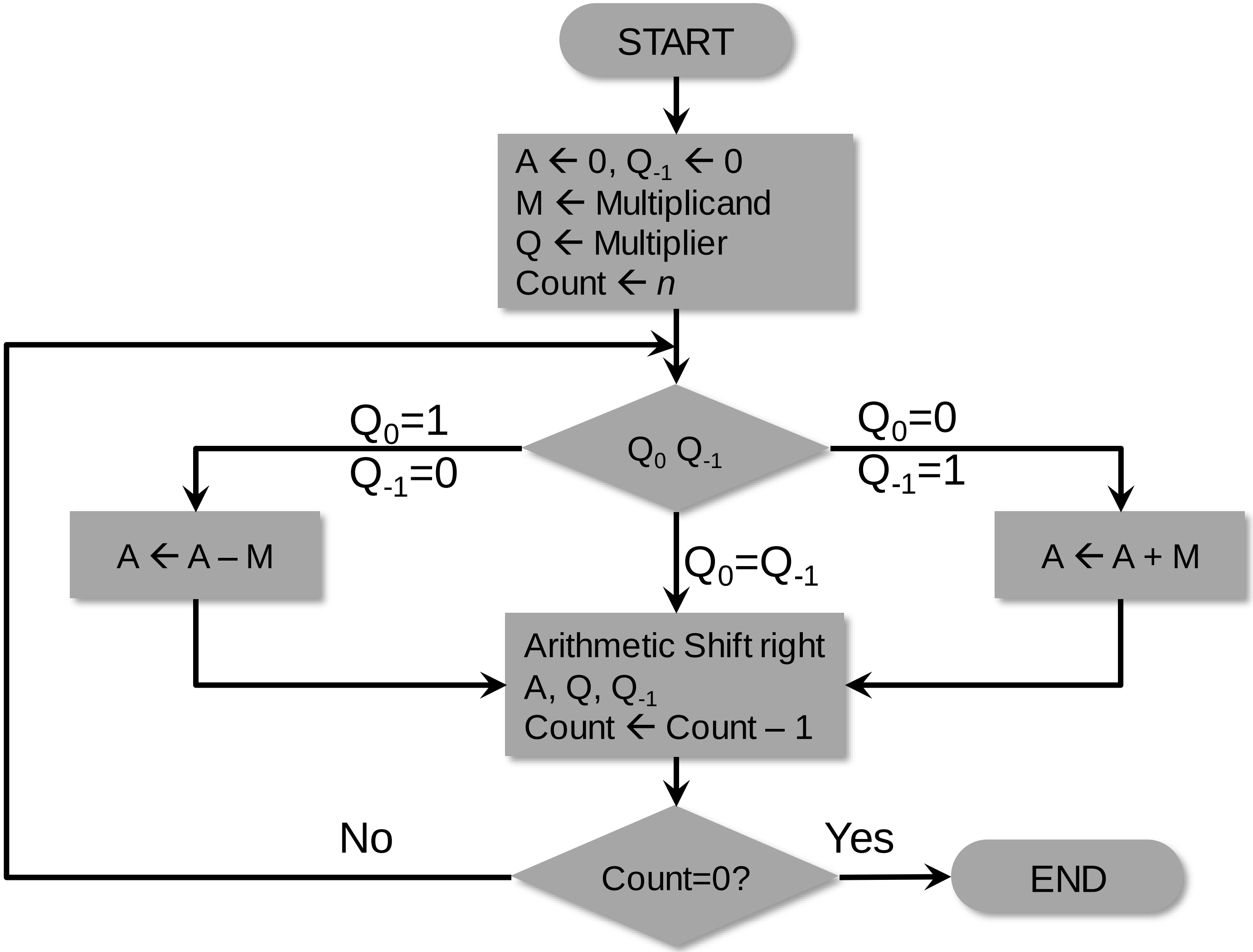
Booth's Algorithm (1/2)

- Works with any combination of positive and negative numbers
- Efficient as compared to previously discussed methods
- Blocks of 1s and 0s are skipped over and just shifting is performed

Booth's Algorithm (2/2)



Flowchart of Booth's Algorithm



Booth's Algorithm Example

A	Q	Q_{-1}	M		
0000	1101	0	1011	Initial Values	
0101	1101	0	1011	A = A–M	First Cycle
0010	1110	1	1011	SAR	
1101	1110	1	1011	A = A+M	Second Cycle
1110	1111	0	1011	SAR	
0011	1111	0	1011	A = A–M	Third Cycle
0001	1111	1	1011	SAR	
0001	1111	1	1011	Do Nothing	Fourth Cycle
0000	1111	1	1011	SAR	