DIGITAL DESIGN & COMPUTER ORGANISATION

Division

Sudarshan T S B., Ph.D.
Department of Computer Science & Engineering

DIGITAL DESIGN & COMPUTER ORGANISATION

Division

Sudarshan T S B., Ph.D.
Department of Computer Science & Engineering

DIVISION

Course Outline

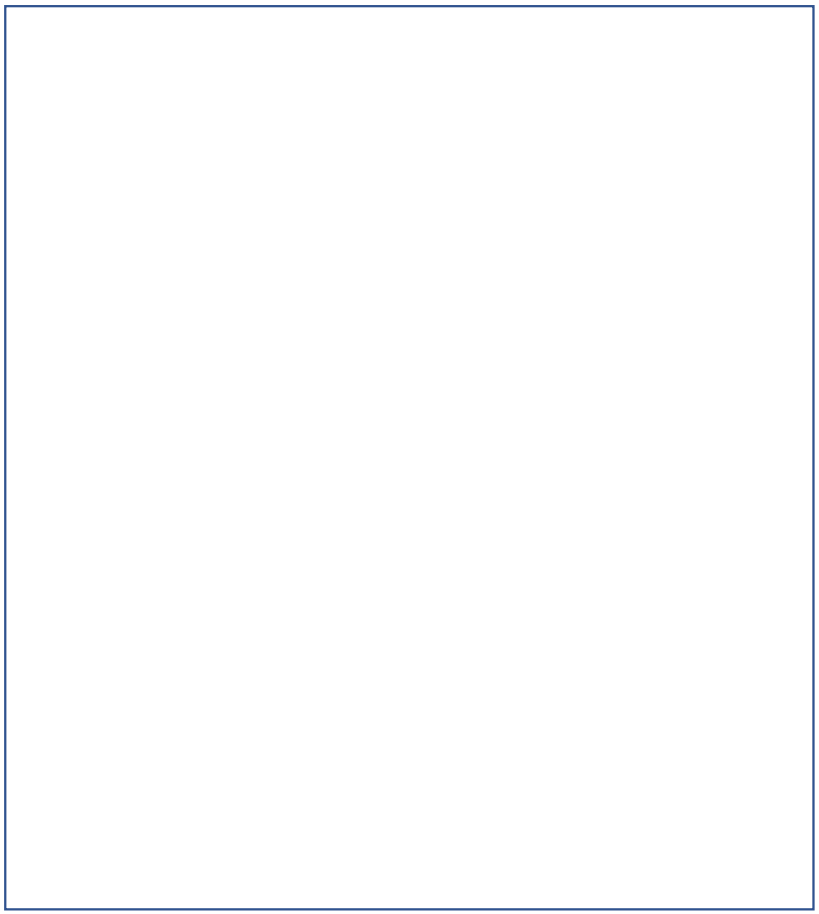
- Digital Design
  - Combinational logic design
  - Sequential logic design
    - ★ Divider
- Computer Organisation
  - Architecture (microprocessor instruction set)
  - Microarchitecture (microprocessor operation)

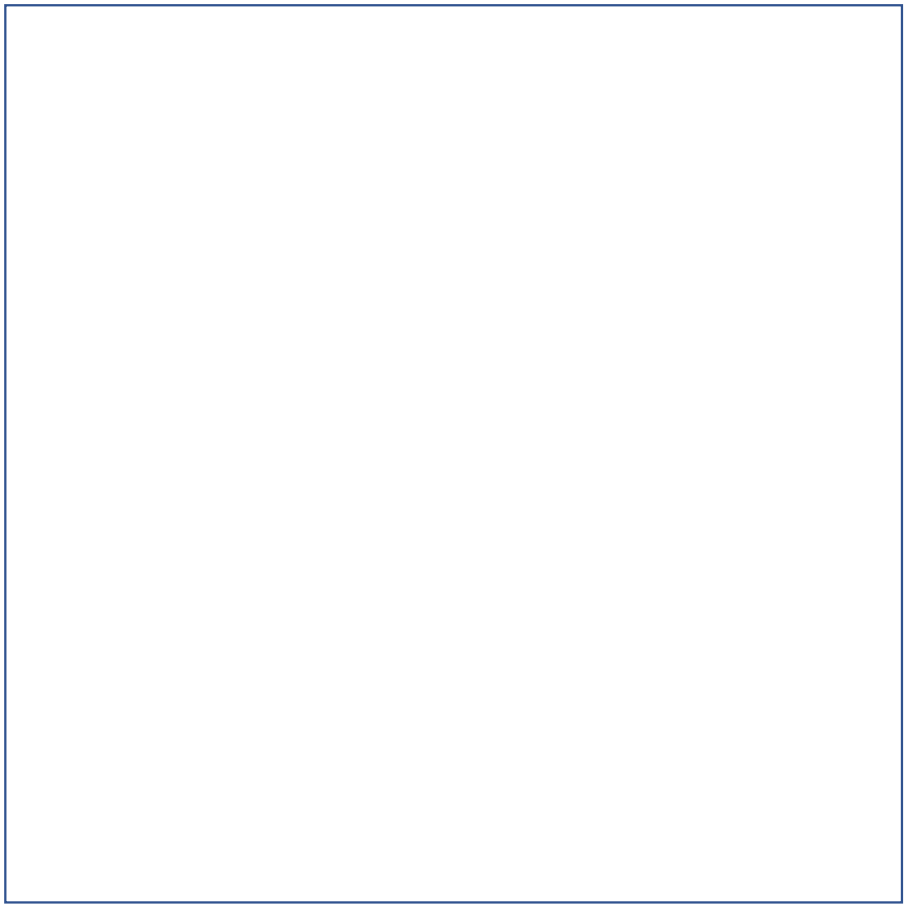
Concepts covered

- Binary Integer Division

DIVISION

Long Hand Method





DIVISION

Long Hand Method

- Manual method of division of decimal number

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

  (13) Divisor, M. => 1101

  (274) Dividend, Q => 100010010

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
    021
13) 274
    26
    14
    13
    ---
     1
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
    ---
     1
```

```
        0
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
    ---
     1
```

```
      00
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
    ---
     1
```

```
      000
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
  021
13) 274
    26
    14
    13
    ---
     1
```

```
      0000
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
    ---
     1
```

```
      00001
1101) 100010010
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
  021
13) 274
    26
    14
    13
     1
```

```
     00001
1101) 100010010
      1101
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
  021
13) 274
    26
    14
    13
     1
```

```
     00001
1101) 100010010
      1101
       100
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      00001
1101) 100010010
      1101
      ----
       100
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      00001
1101) 100010010
      1101
      1000
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010
1101) 100010010
      1101
      1000
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010
1101) 100010010
      1101
      1000
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010
1101) 100010010
      1101
      10000
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      0000101
1101) 100010010
      1101
      10000
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      0000101
1101) 100010010
       1101
       10000
        1101
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      0000101
1101) 100010010
      1101
      10000
       1101
         11
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      0000101
1101) 100010010
       1101
       10000
        1101
          11
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      0000101
1101) 100010010
       1101
       10000
        1101
        111
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
        00001010
1101) 100010010
       1101
       10000
        1101
         111
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
  ---
13) 274
    26
    --
    14
    13
    --
     1
```

```
        00001010
      ----------
1101) 100010010
      1101
      ----
      10000
       1101
       ----
        111
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
        00001010
1101) 100010010
        1101
        10000
         1101
         1110
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
  ---
13) 274
    26
   ---
    14
    13
   ---
     1
```

```
        000010101
      ------------
1101) 100010010
        1101
      ------
        10000
         1101
        -----
         1110
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
           1
```

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
        000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
           1
```

Quotient = 21 => 10101; Remainder = 1

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
           1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
           1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend
- If the remainder is zero or positive quotient bit is 1, the remainder is extended by another bit of the dividend to repeat the process

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
      1101
      10000
      1101
      1110
      1101
         1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend
- If the remainder is zero or positive quotient bit is 1, the remainder is extended by another bit of the dividend to repeat the process
- If the remainder is negative, quotient is 0 the dividend is restored by adding back the divisor.

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
      1101
      10000
       1101
       1110
       1101
          1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend
- If the remainder is zero or positive quotient bit is 1, the remainder is extended by another bit of the dividend to repeat the process
- If the remainder is negative, quotient is 0 the dividend is restored by adding back the divisor.
- Repeat the process until all the digits in the dividend are considered

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
       1101
       10000
        1101
        1110
        1101
           1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend
- If the remainder is zero or positive quotient bit is 1, the remainder is extended by another bit of the dividend to repeat the process
- If the remainder is negative, quotient is 0 the dividend is restored by adding back the divisor.
- Repeat the process until all the digits in the dividend are considered
- This is called **Restoring Division** algorithm

DIVISION

Long Hand Method

- Manual method of division of decimal number
- Binary division also can be performed in a similar way
- Repetitive Subtraction: Mark 1 for success and 0 for failure

(13) Divisor, M. => 1101

(274) Dividend, Q => 100010010

```
   021
13) 274
    26
    14
    13
     1
```

```
      000010101
1101) 100010010
      1101
      10000
       1101
       1110
       1101
          1
```

Quotient = 21 => 10101; Remainder = 1

- A subtraction of divisor is performed with dividend
- If the remainder is zero or positive quotient bit is 1, the remainder is extended by another bit of the dividend to repeat the process
- If the remainder is negative, quotient is 0 the dividend is restored by adding back the divisor.
- Repeat the process until all the digits in the dividend are considered
- This is called **Restoring Division** algorithm
- The other method to be familiar with is **Non-Restoring Division**

Restoring Division

DIVISION

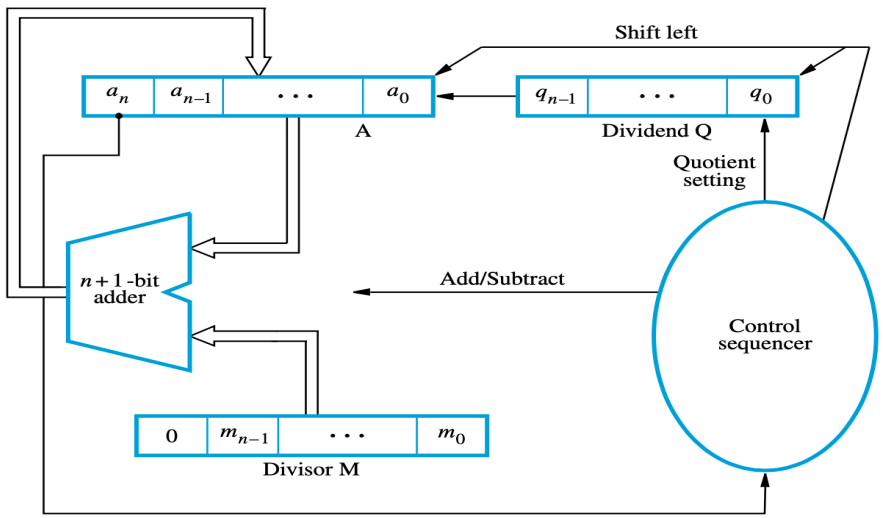
Restoring Division

- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub

DIVISION

Restoring Division

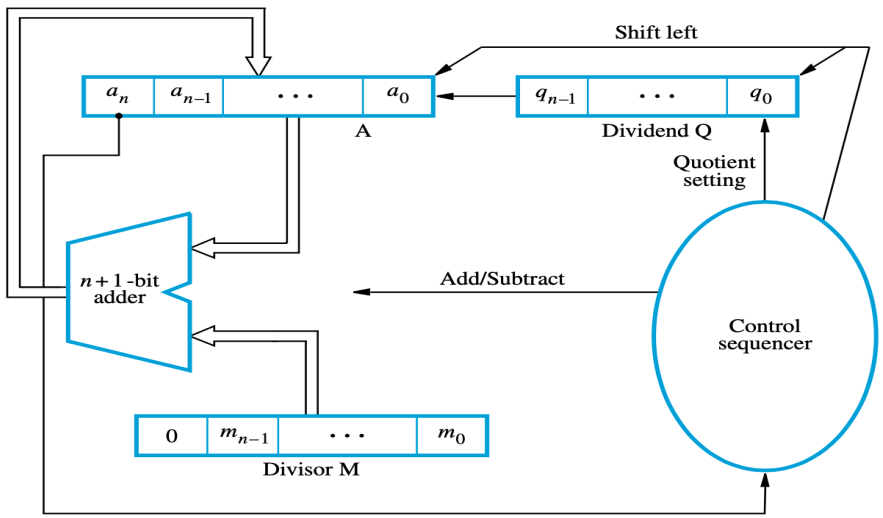
- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub



DIVISION

Restoring Division

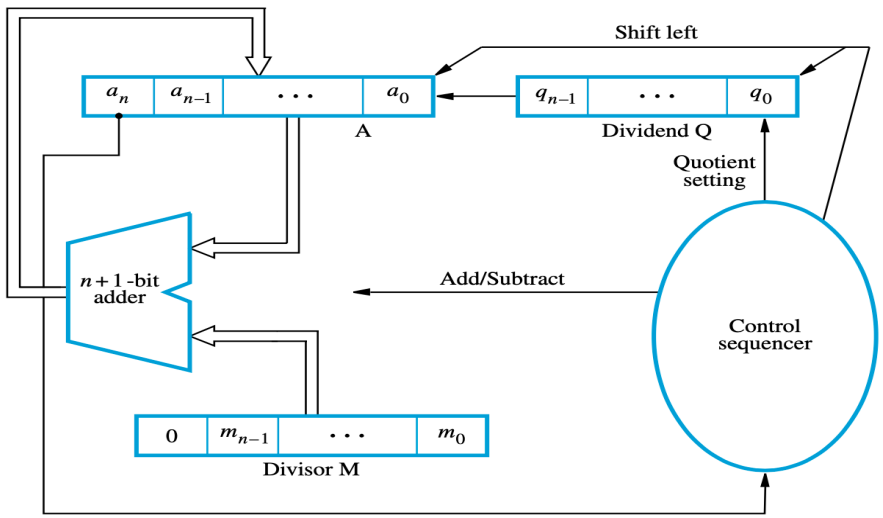
- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub

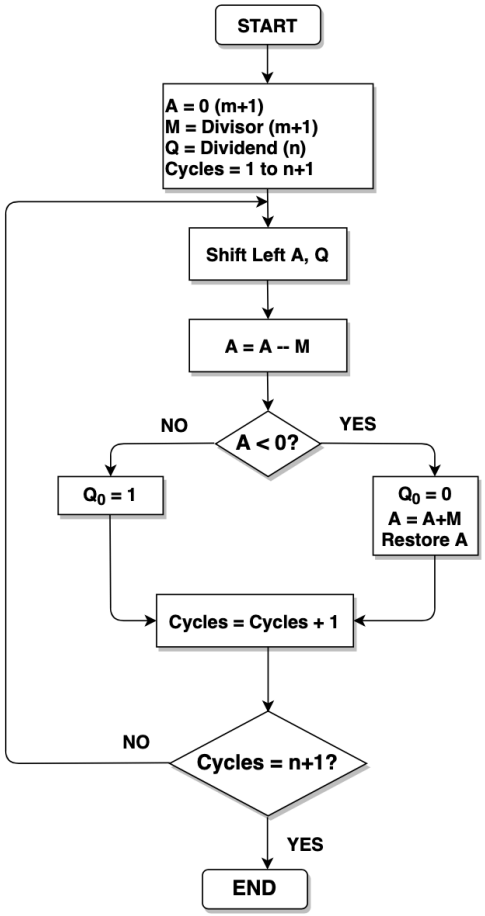


DIVISION

Restoring Division

- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub





DIVISION

Restoring Division

Consider:

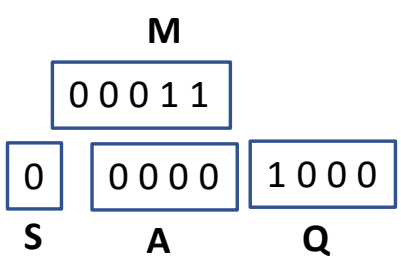
Divisor => 00011

2's Comp=> 11101

Dividend => 1000

DIVISION

Restoring Division



Consider:

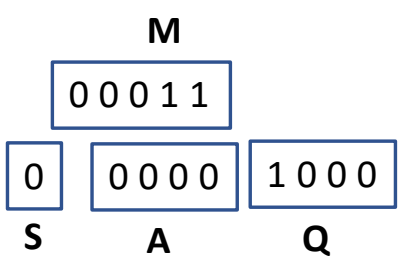
Divisor => 00011

2's Comp=> 11101

Dividend => 1000

DIVISION

Restoring Division

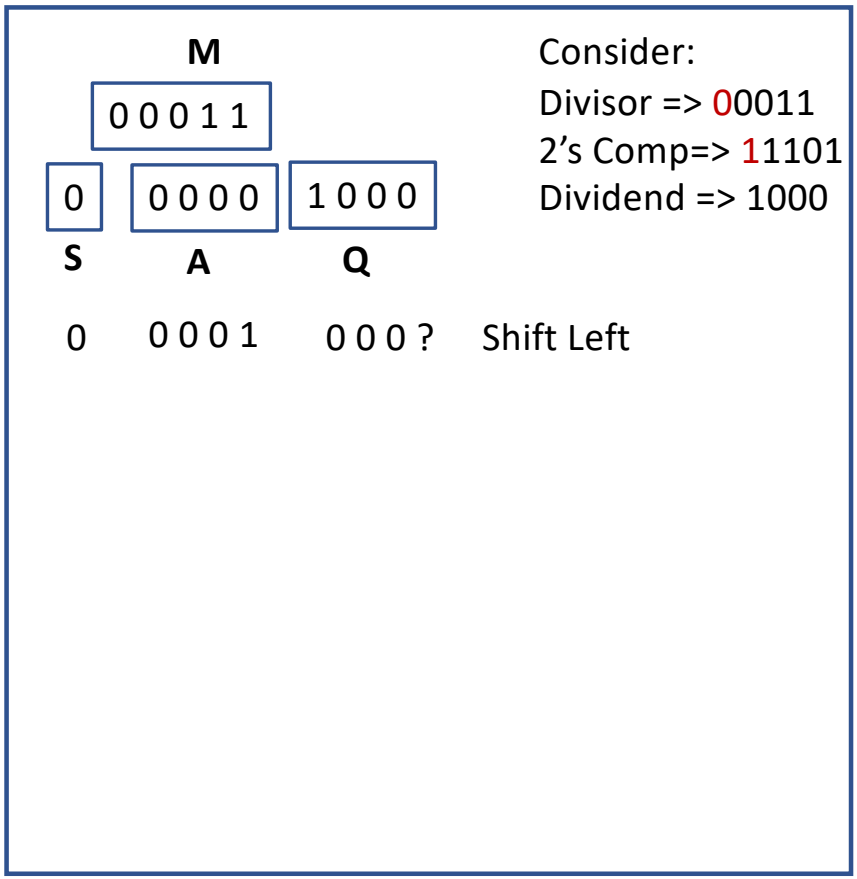


Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

Shift Left

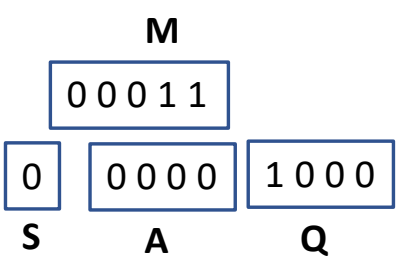
DIVISION

Restoring Division



DIVISION

Restoring Division

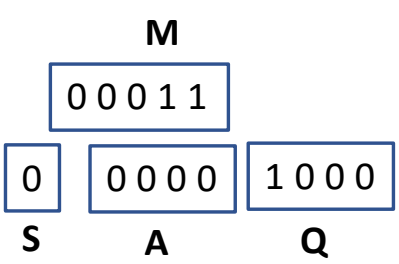


Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 0 1	0 0 0 ?	Shift Left
			Sub M (2's comp)

DIVISION

Restoring Division

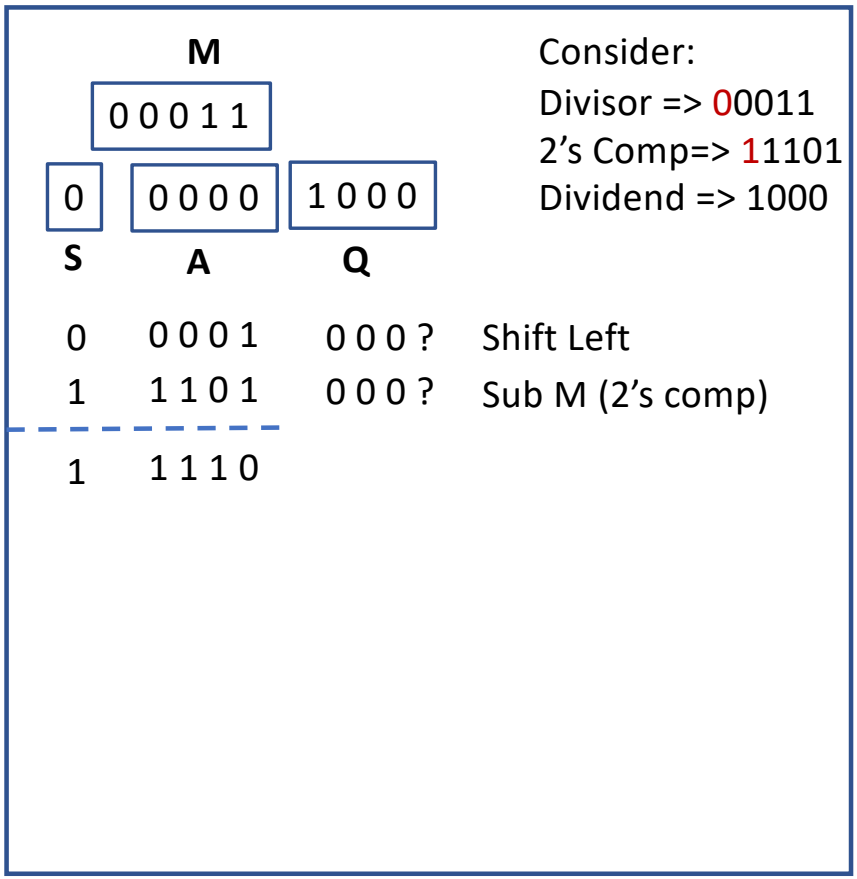


Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)

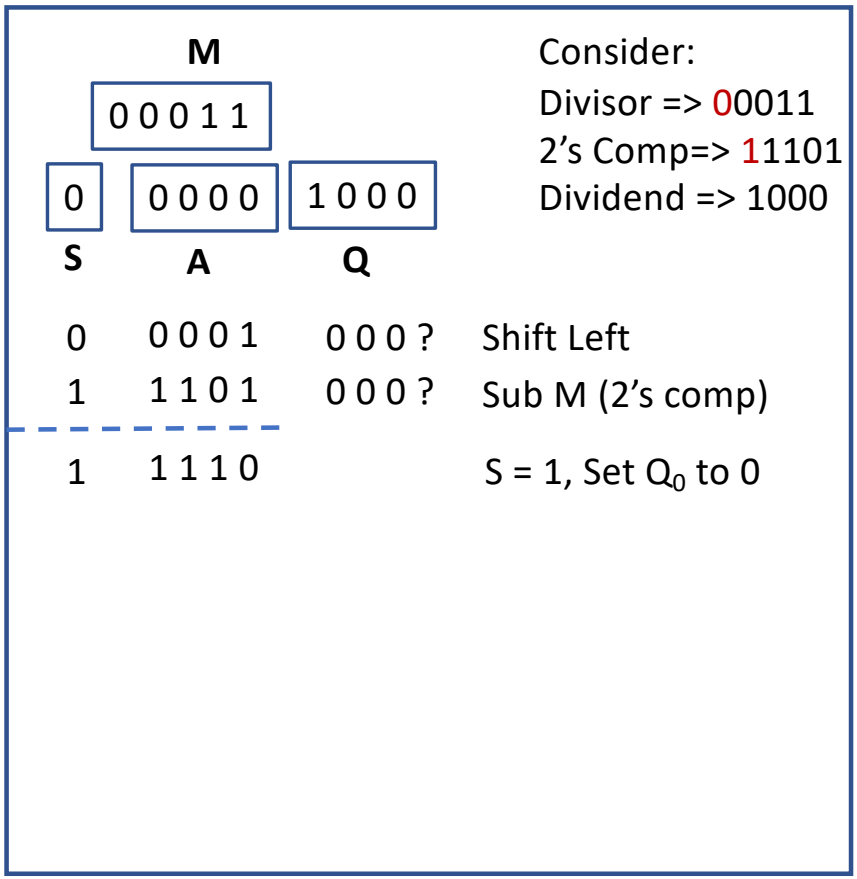
DIVISION

Restoring Division



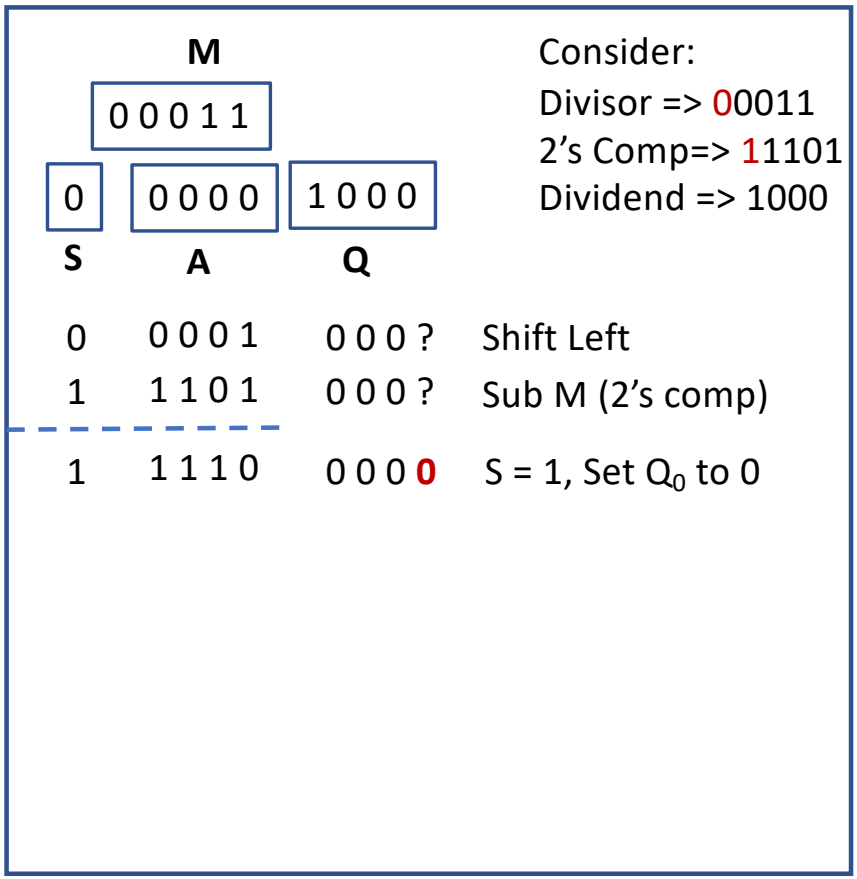
DIVISION

Restoring Division



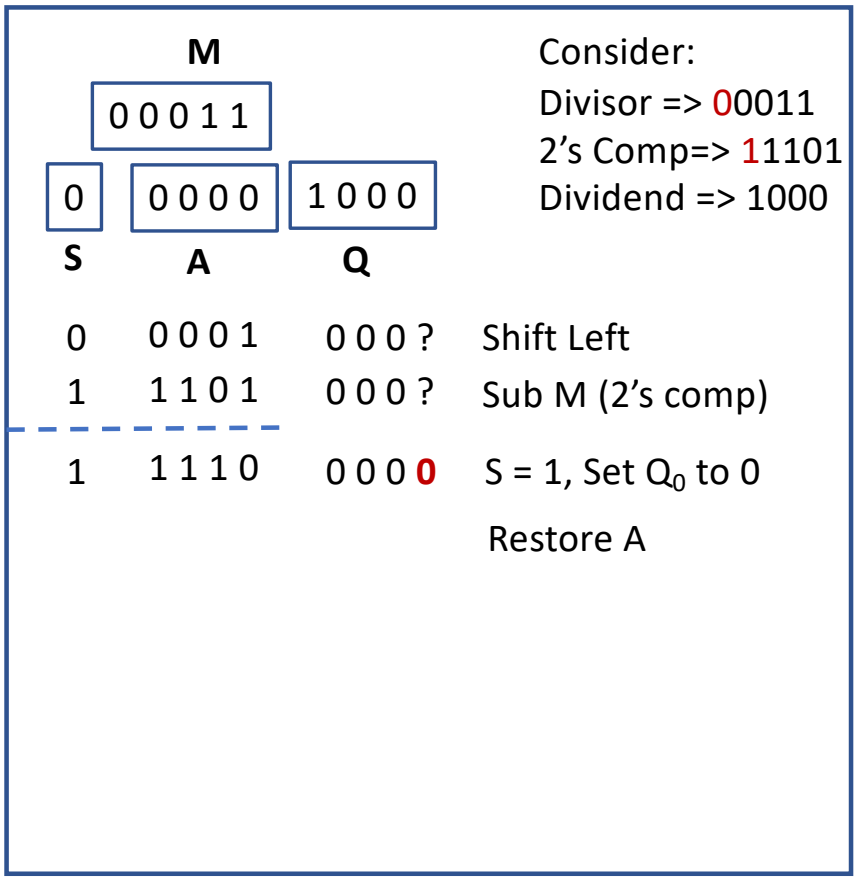
DIVISION

Restoring Division



DIVISION

Restoring Division



DIVISION

Restoring Division

M			
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

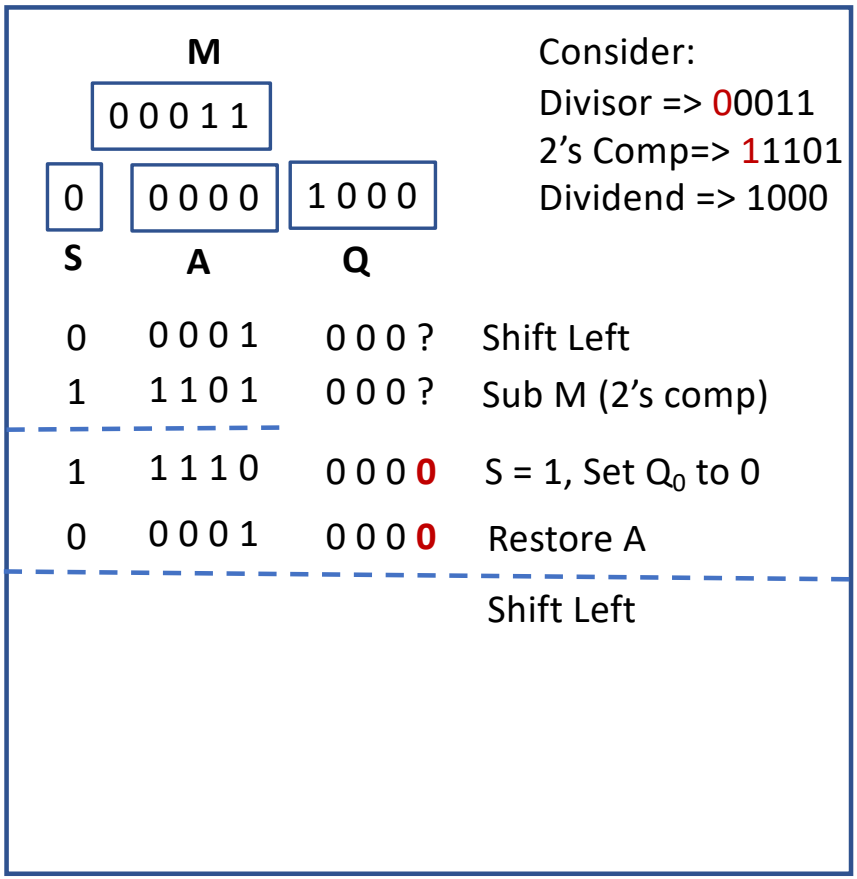
Restoring Division

M			
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

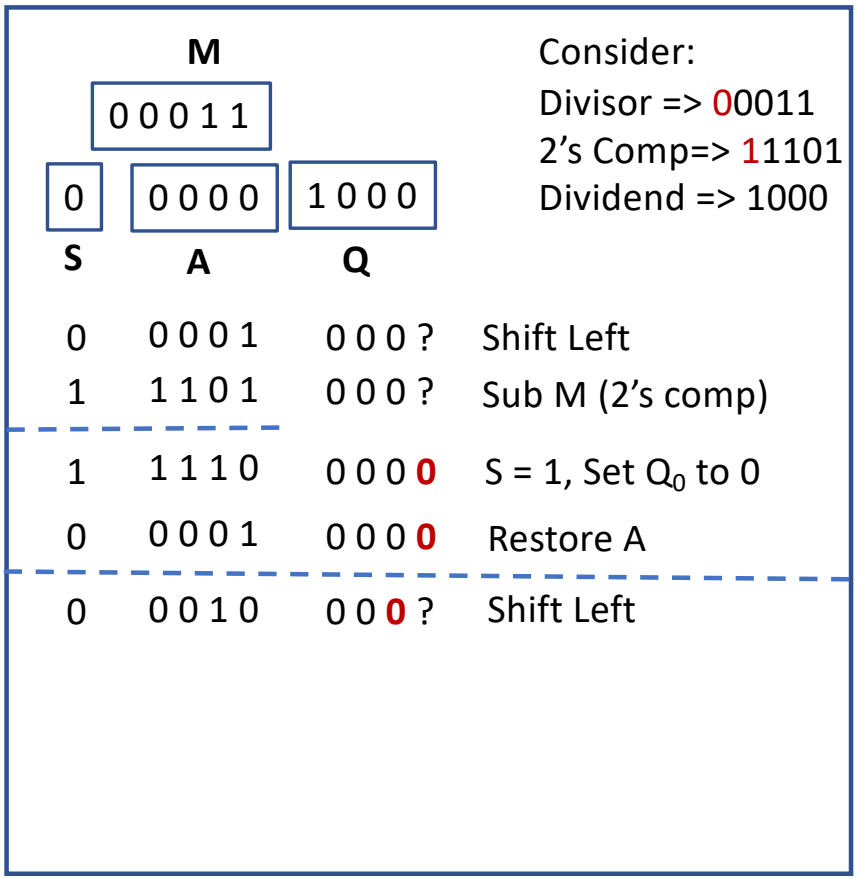
DIVISION

Restoring Division



DIVISION

Restoring Division



DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
			Sub M (2's comp)

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1		

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1		S =1, Set Q_0 to 0

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0 Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

DIVISION

Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011 2's Comp=> 11101 Dividend => 1000
0	0 0 0 0	1 0 0 0	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

DIVISION

Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

S	A	Q
0	0 0 1 0	0 0 **0 0**

DIVISION

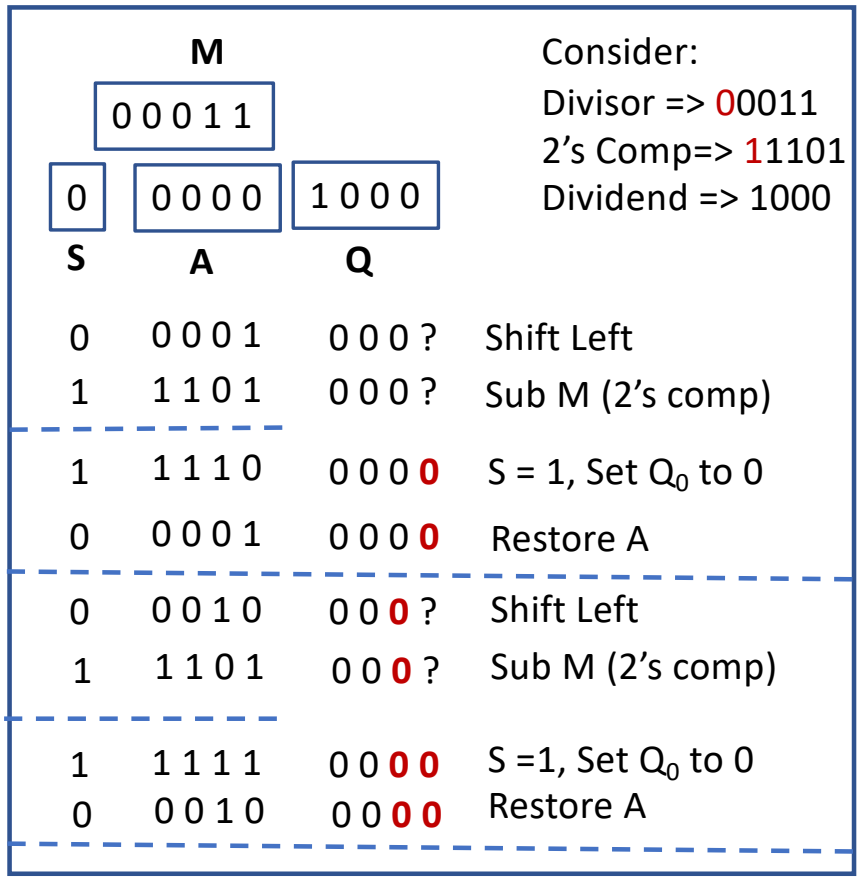
Restoring Division

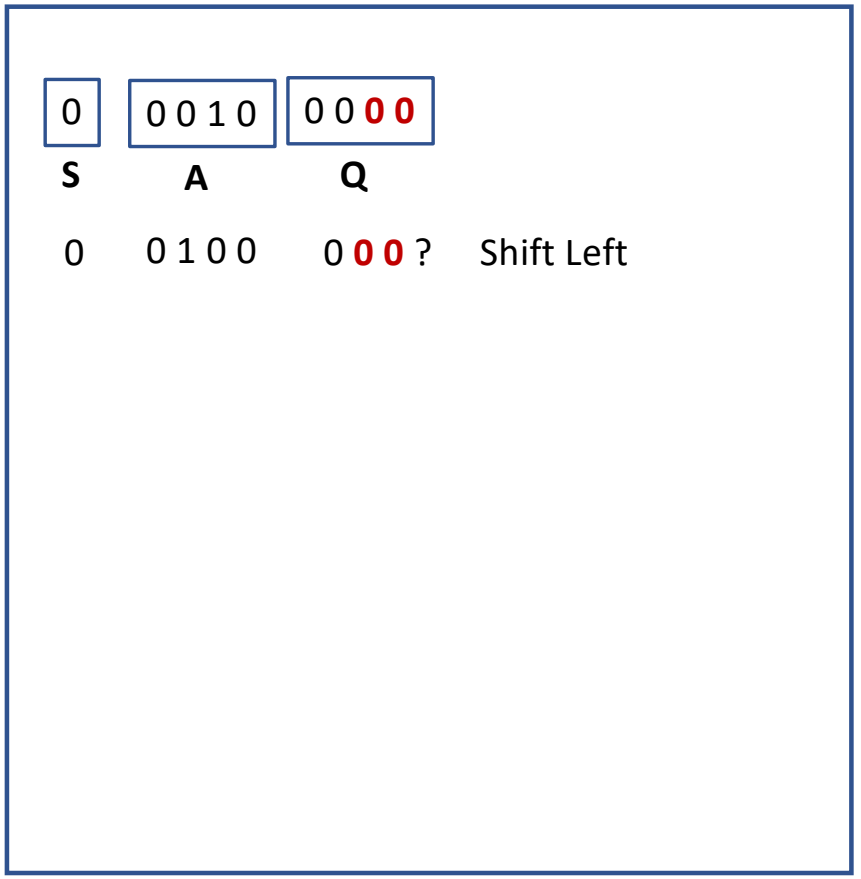
S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

S	A	Q	
0	0 0 1 0	0 0 **0 0**	Shift Left

DIVISION

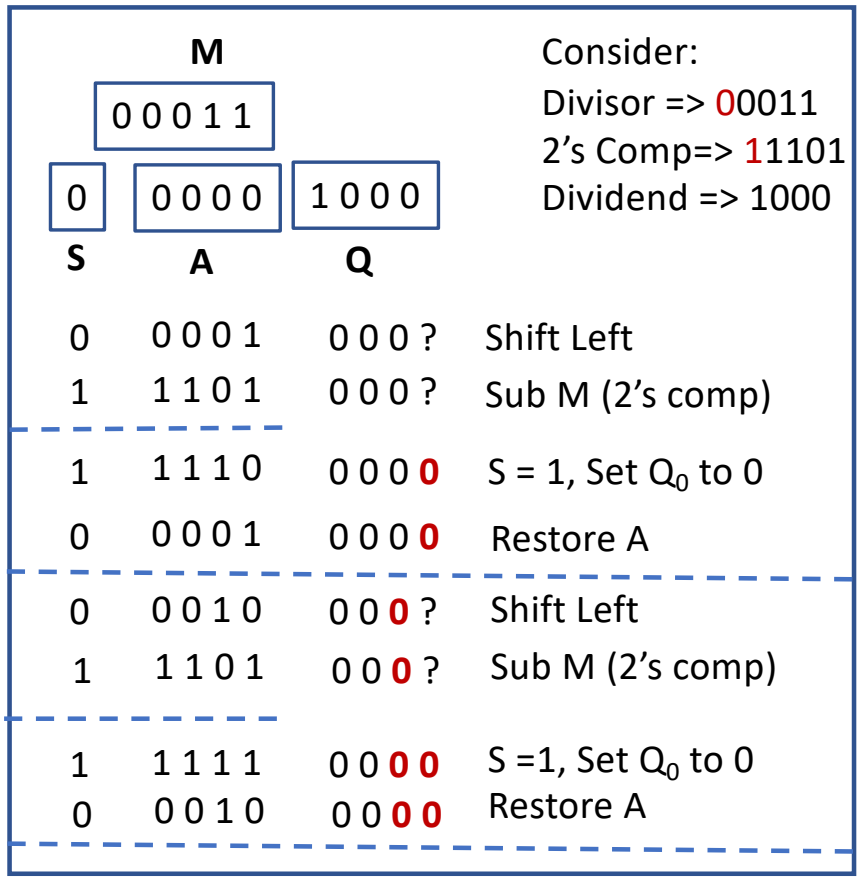
Restoring Division





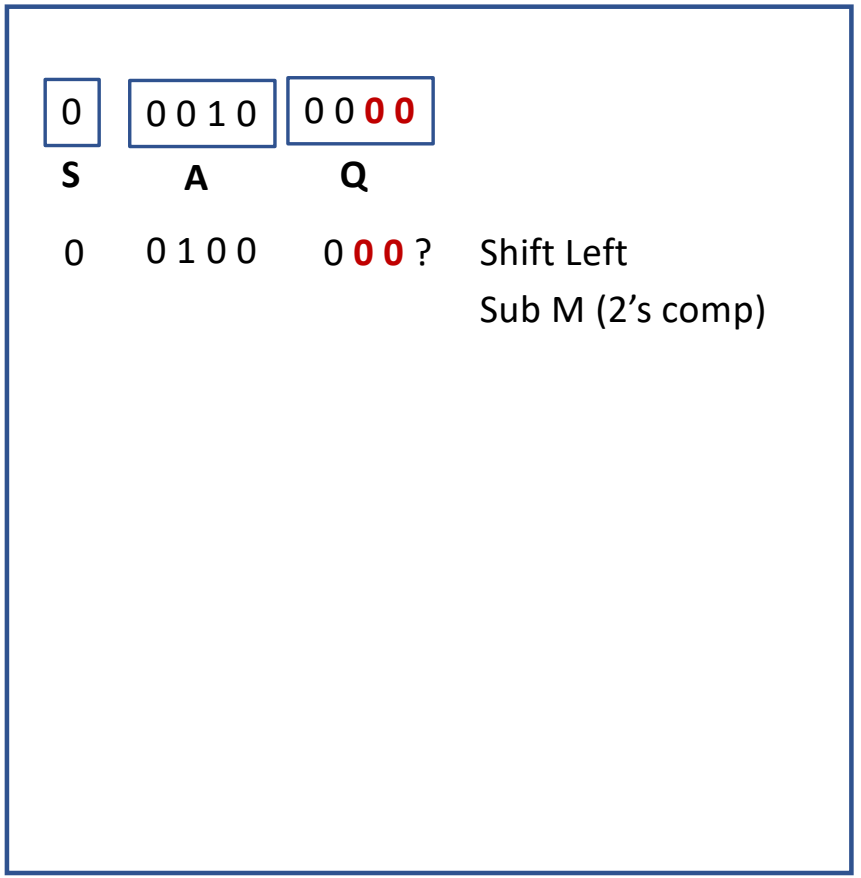
DIVISION

Restoring Division



	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

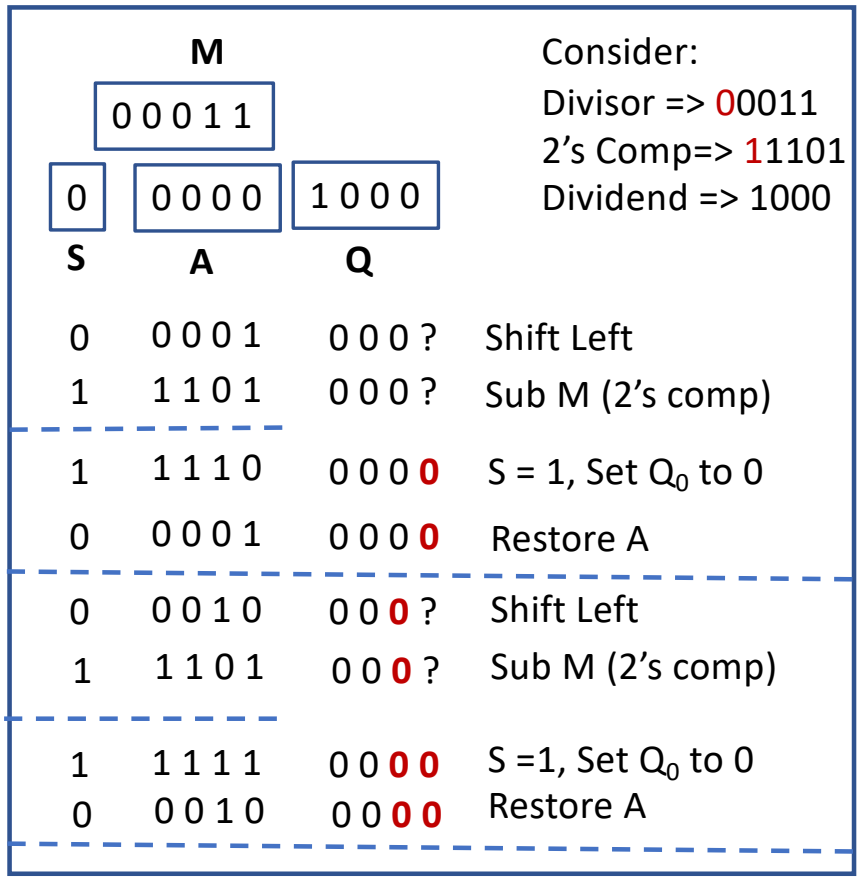
Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

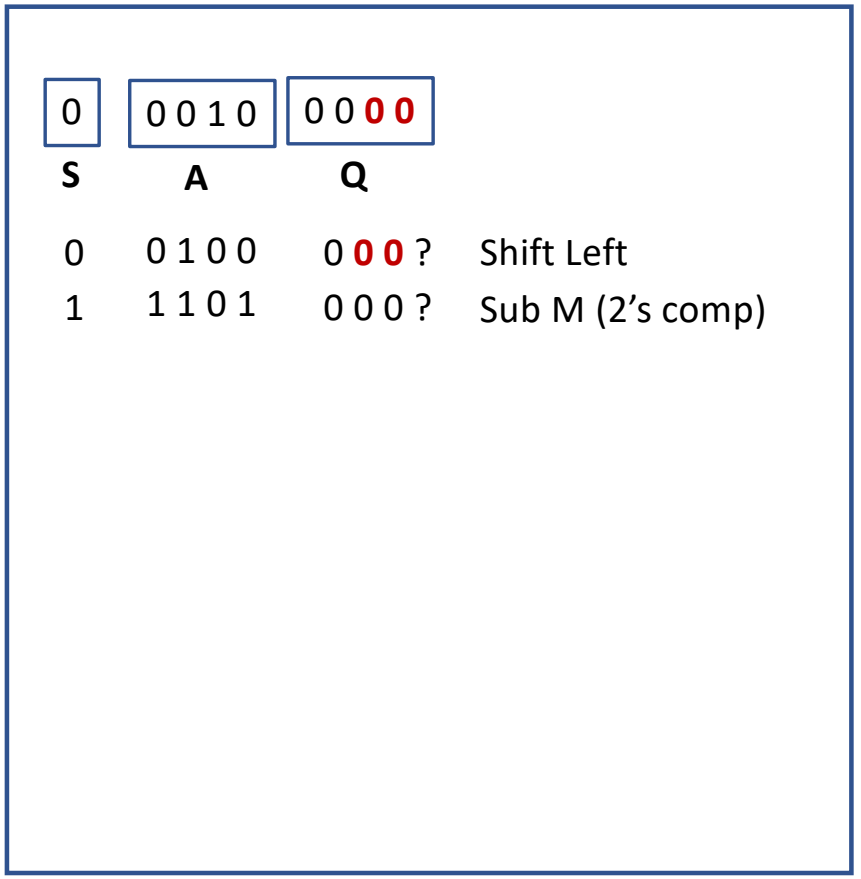


0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
			Sub M (2's comp)

DIVISION

Restoring Division





DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1		

DIVISION

Restoring Division

M			
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1		S = 0, Set Q_0 to 1

DIVISION

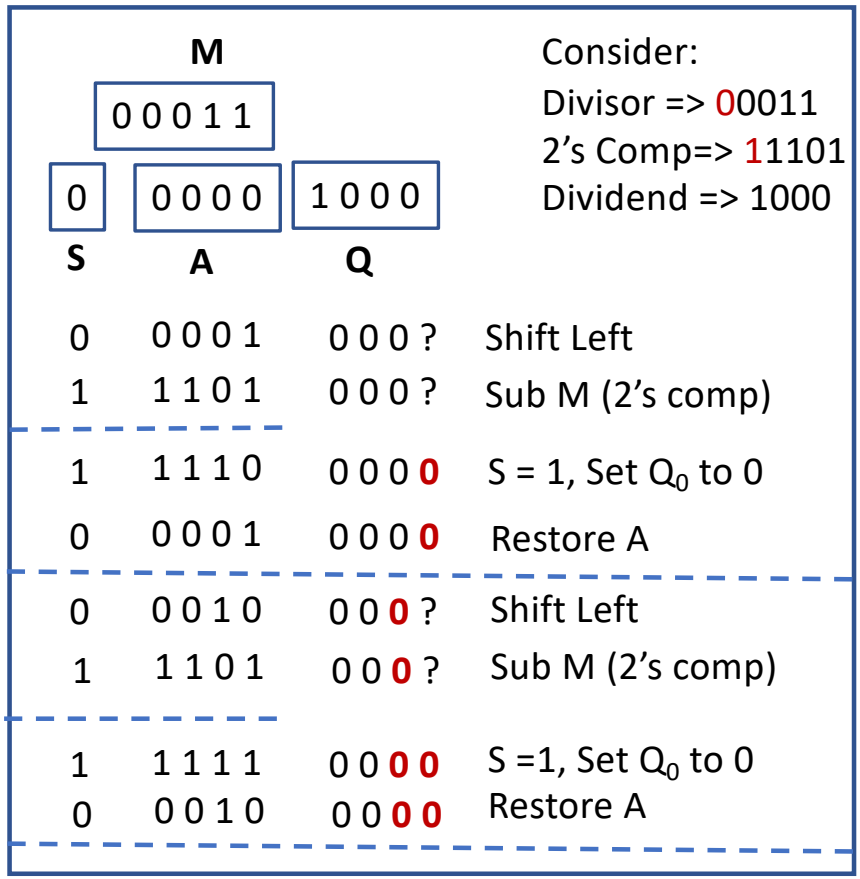
Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

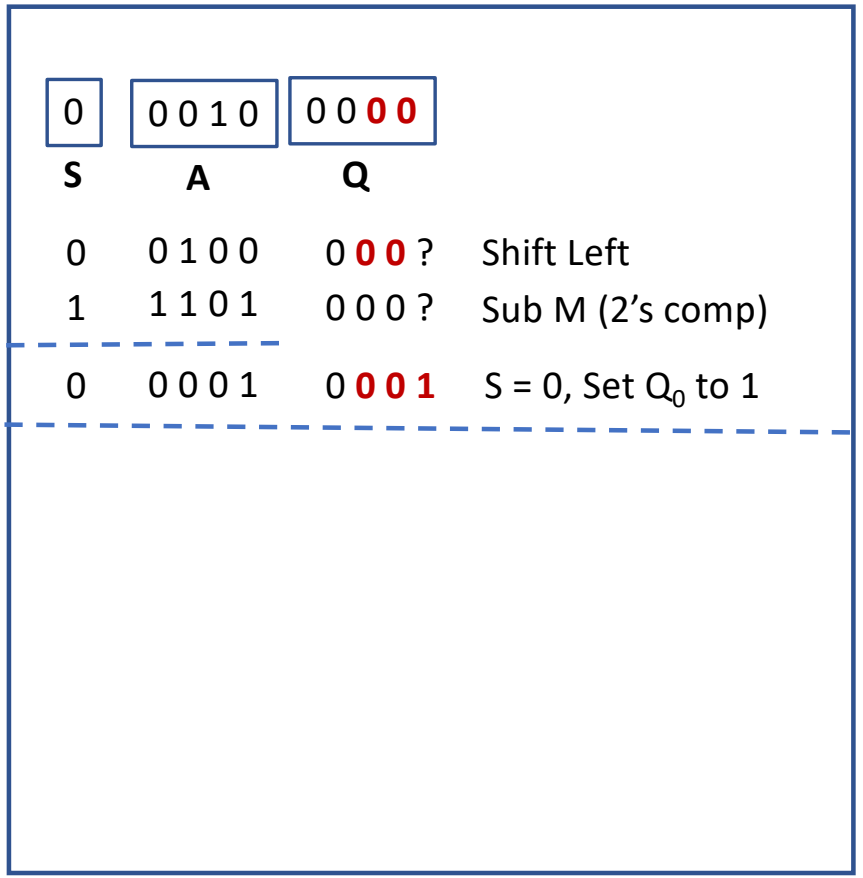
S	A	Q	
0	0 0 1 0	0 0 **0 0**	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1

DIVISION

Restoring Division



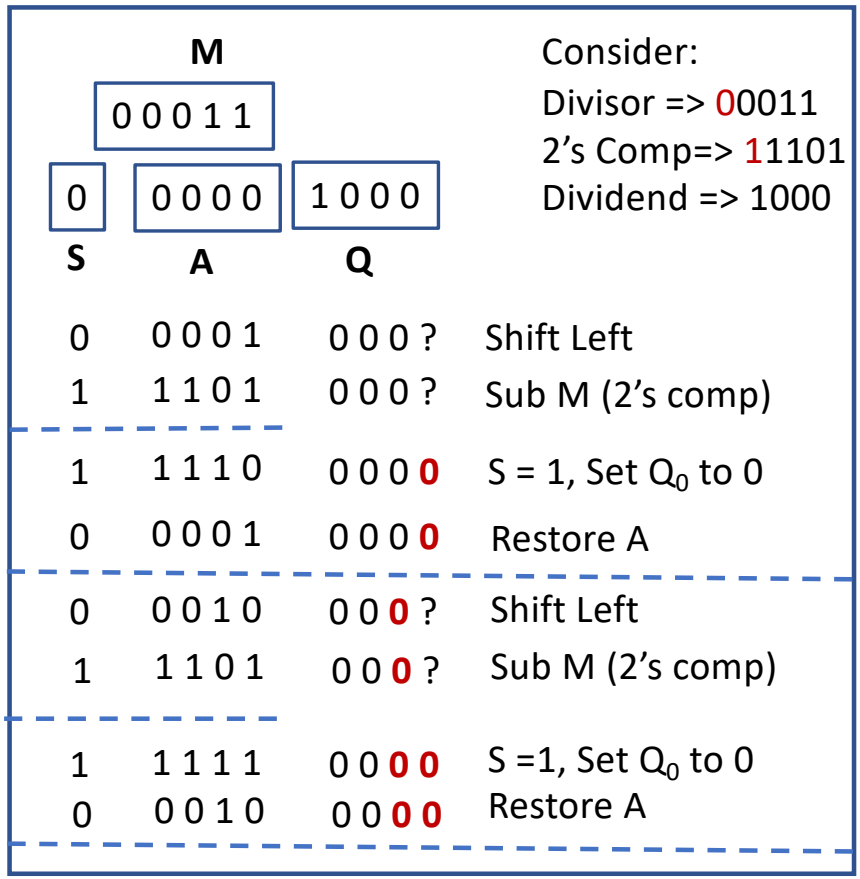
	M		
	0 0 0 1 1		
S	A	Q	
0	0 0 0 0	1 0 0 0	Consider: Divisor => 00011
			2's Comp=> 11101
			Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

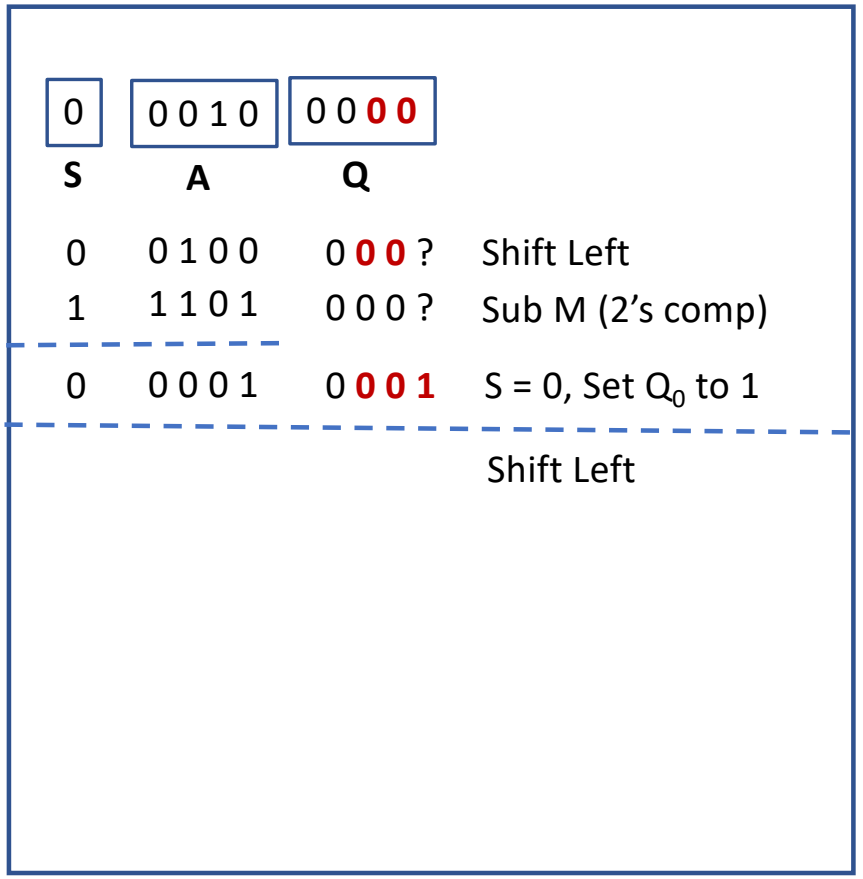


S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1

DIVISION

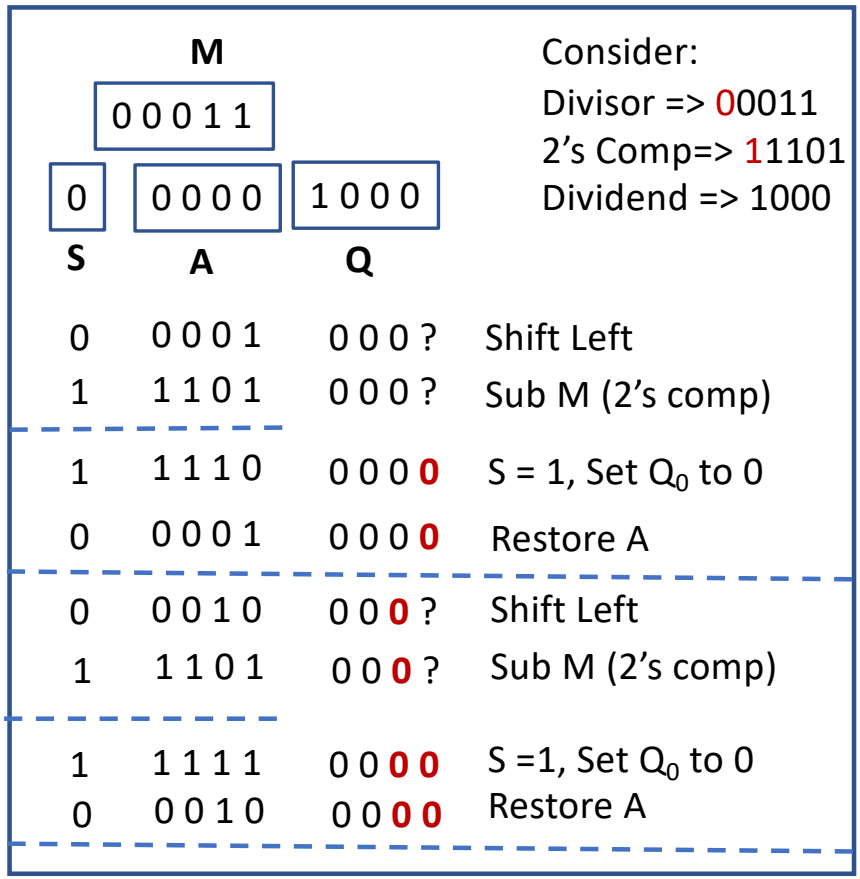
Restoring Division

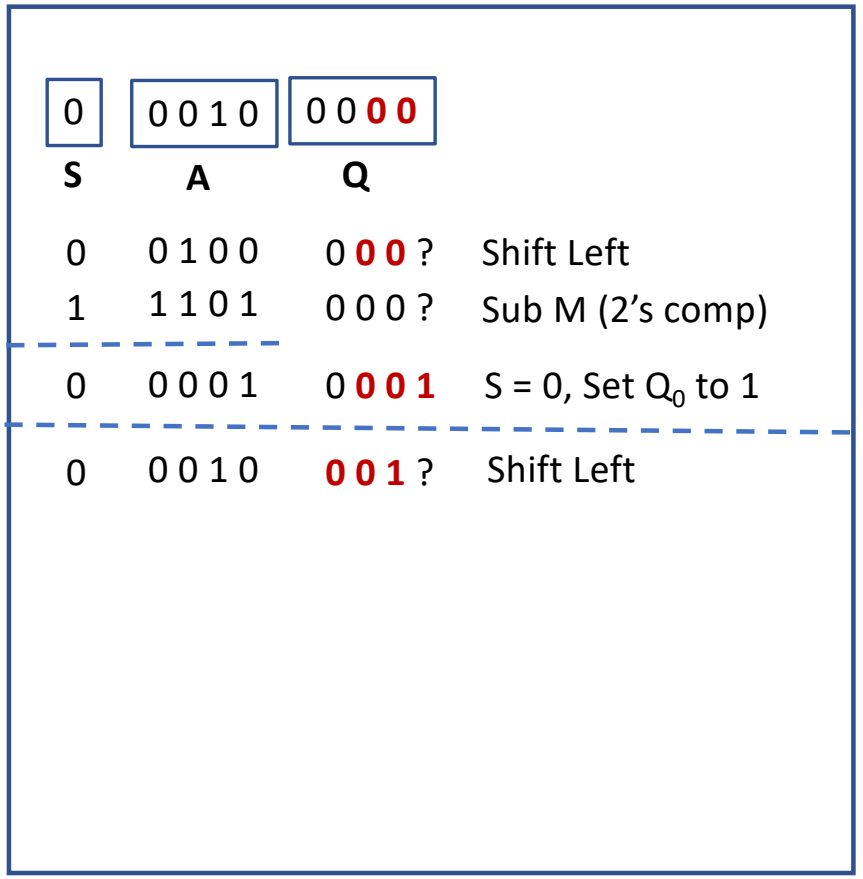




DIVISION

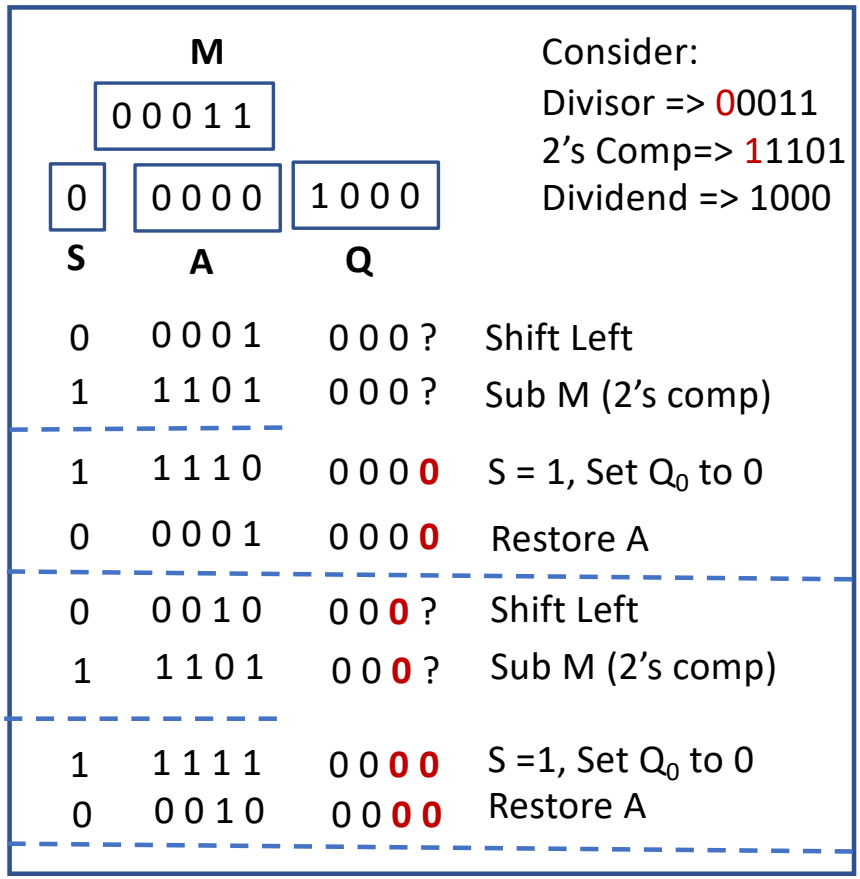
Restoring Division

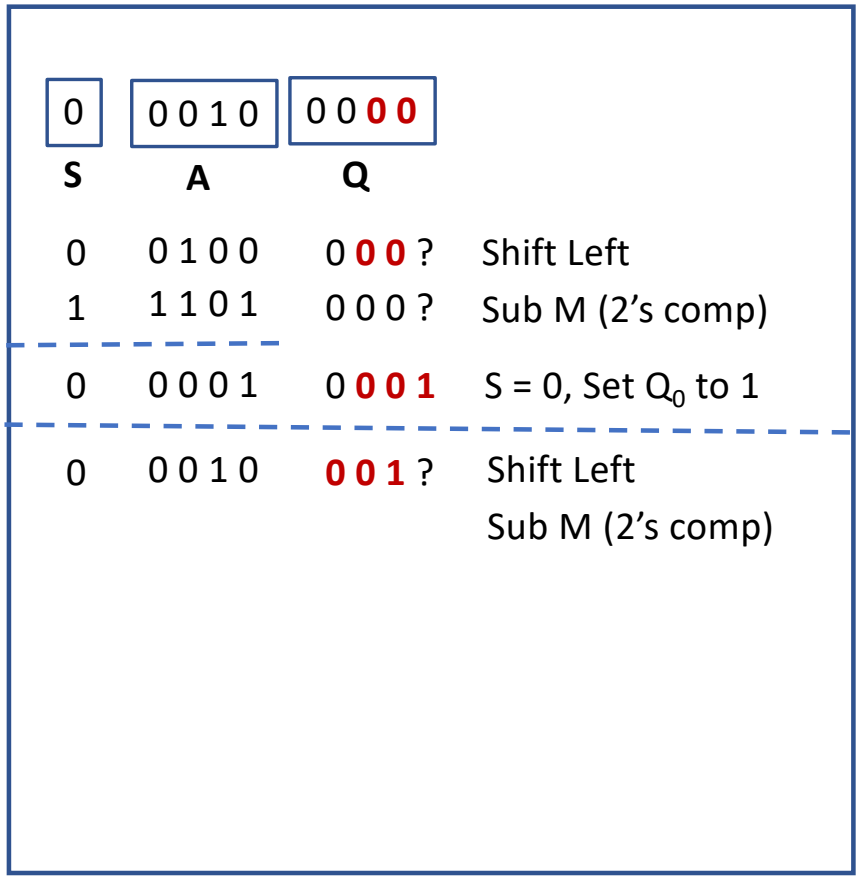




DIVISION

Restoring Division





DIVISION

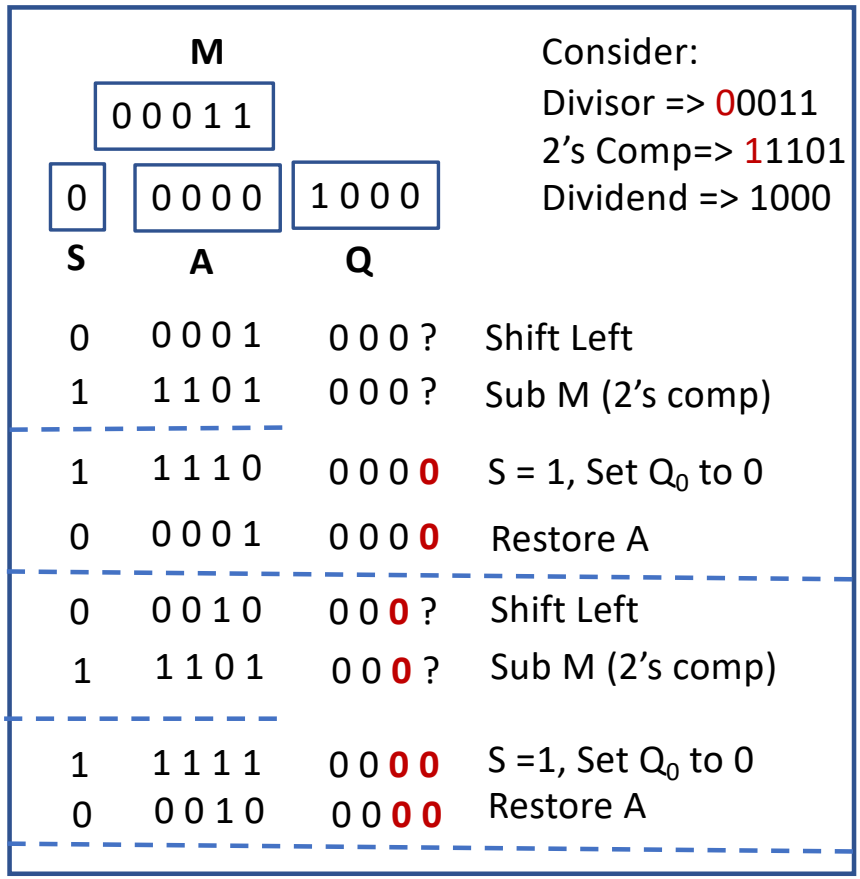
Restoring Division

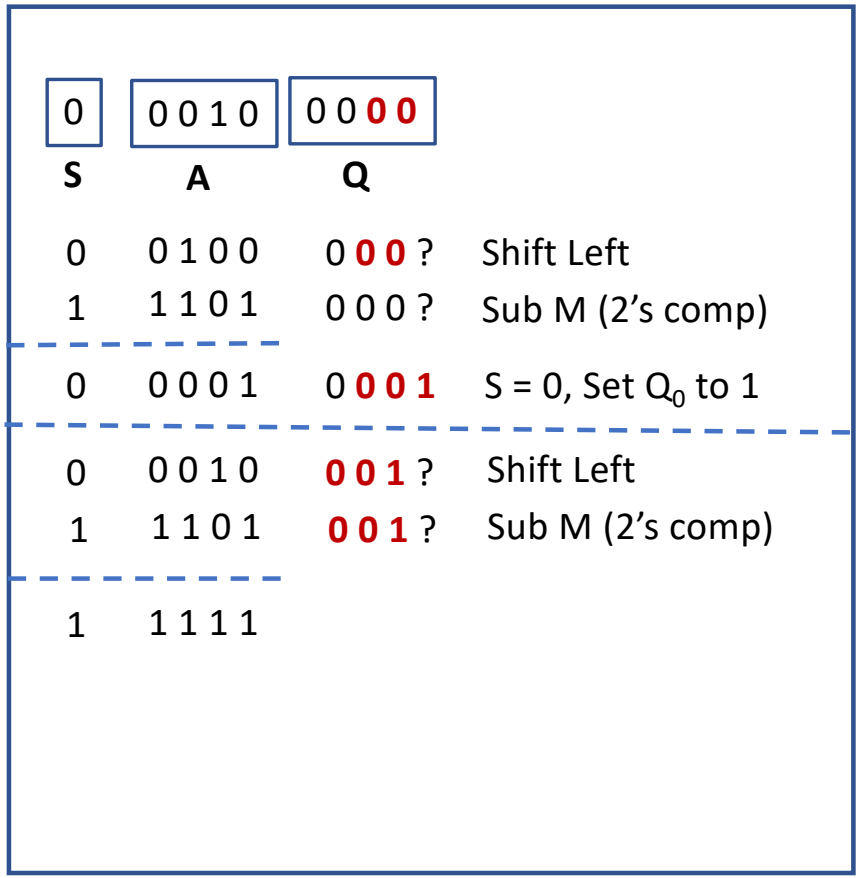
S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

S	A	Q	
0	0 0 1 0	0 0 **0 0**	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)

DIVISION

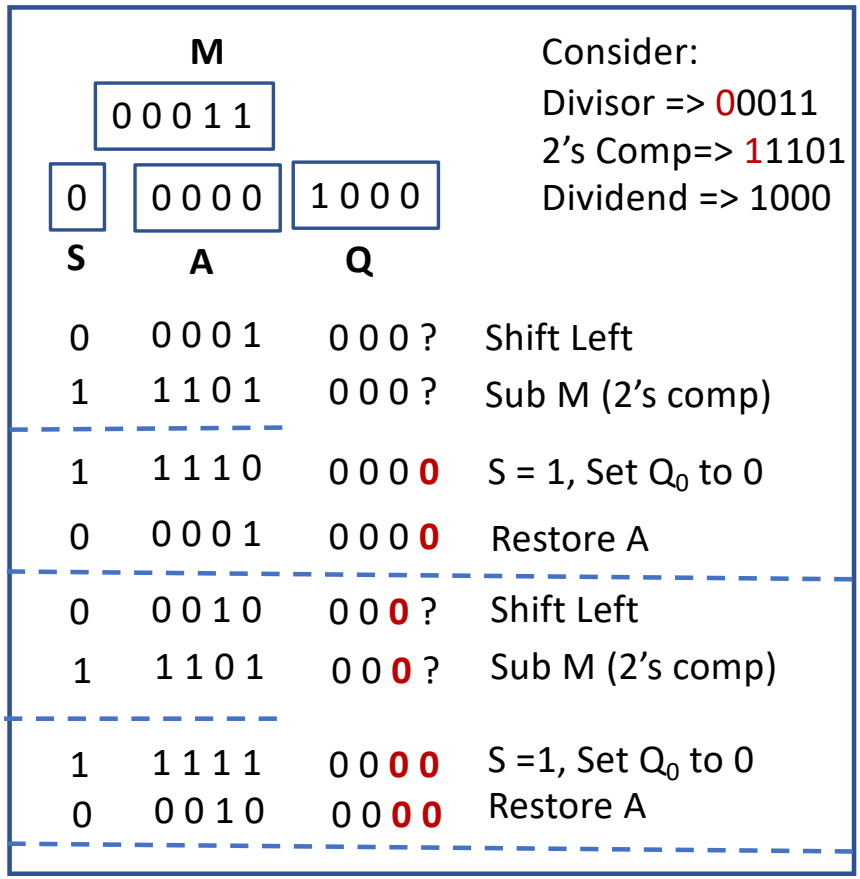
Restoring Division

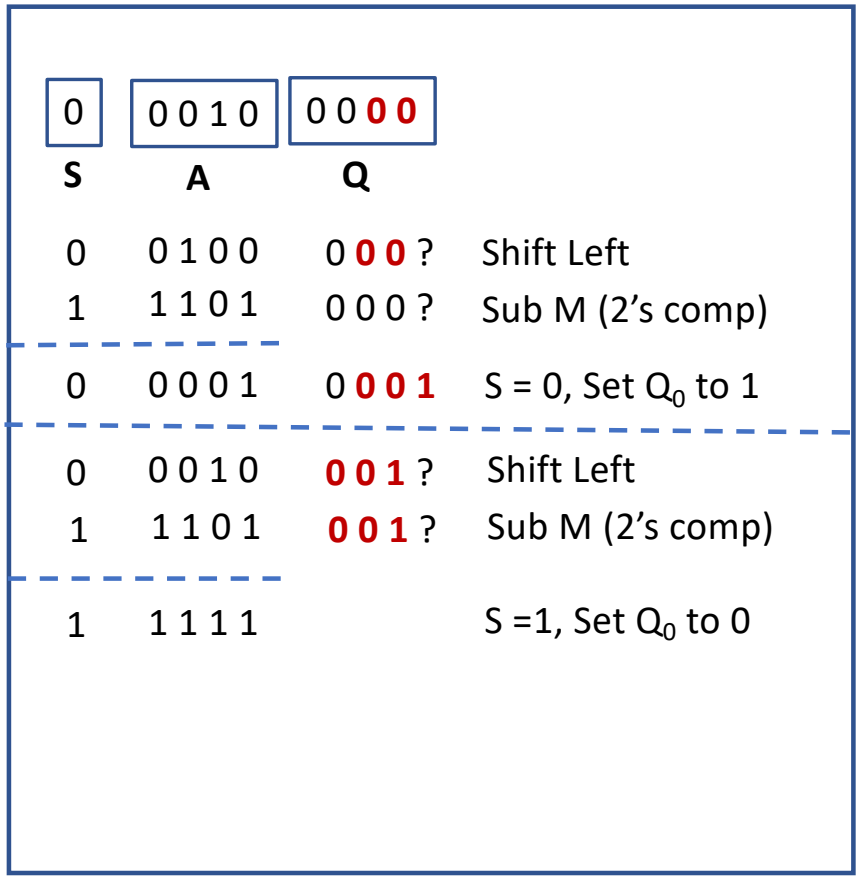




DIVISION

Restoring Division





DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0

DIVISION

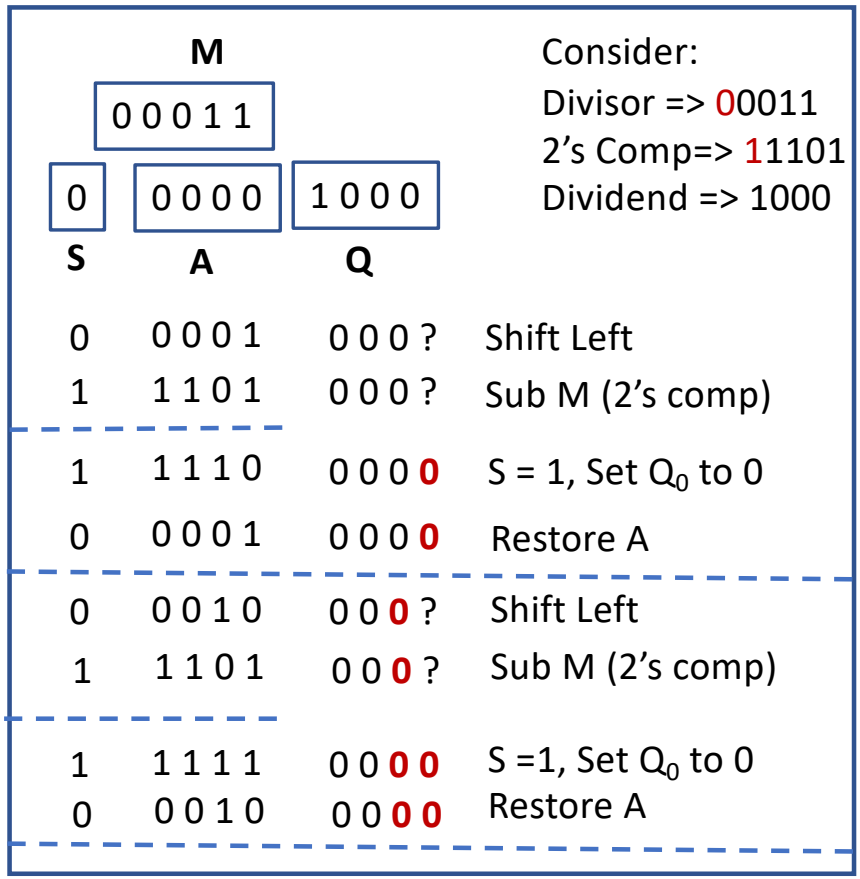
Restoring Division

M				
	0 0 0 1 1			
S	A	Q		
0	0 0 0 0	1 0 0 0		Consider: Divisor => 00011 2's Comp=> 11101 Dividend => 1000
0	0 0 0 1	0 0 0 ?		Shift Left
1	1 1 0 1	0 0 0 ?		Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**		S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**		Restore A
0	0 0 1 0	0 0 **0** ?		Shift Left
1	1 1 0 1	0 0 **0** ?		Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**		S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**		Restore A

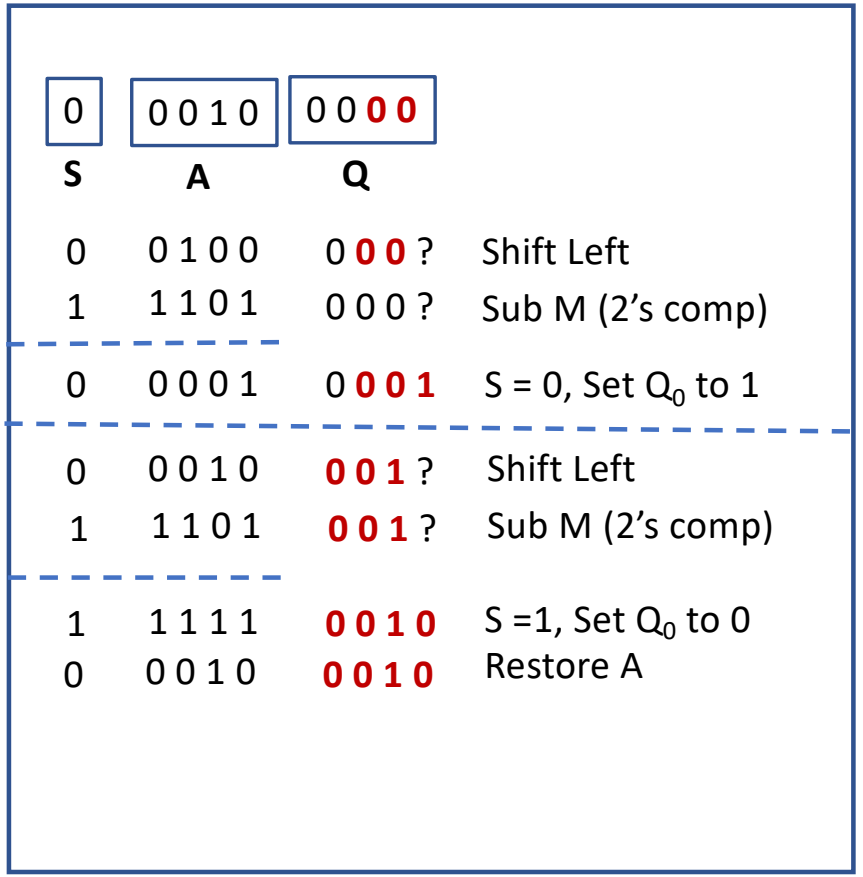
S	A	Q	
0	0 0 1 0	0 0 **0 0**	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0 Restore A

DIVISION

Restoring Division



	M		
	0 0 0 1 1		
S	A	Q	
0	0 0 0 0	1 0 0 0	Consider: Divisor => 00011
			2's Comp=> 11101
			Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A



S	A	Q	
0	0 0 1 0	0 0 **0 0**	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0
0	0 0 1 0	**0 0 1 0**	Restore A

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0
0	0 0 1 0	**0 0 1 0**	Restore A

DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

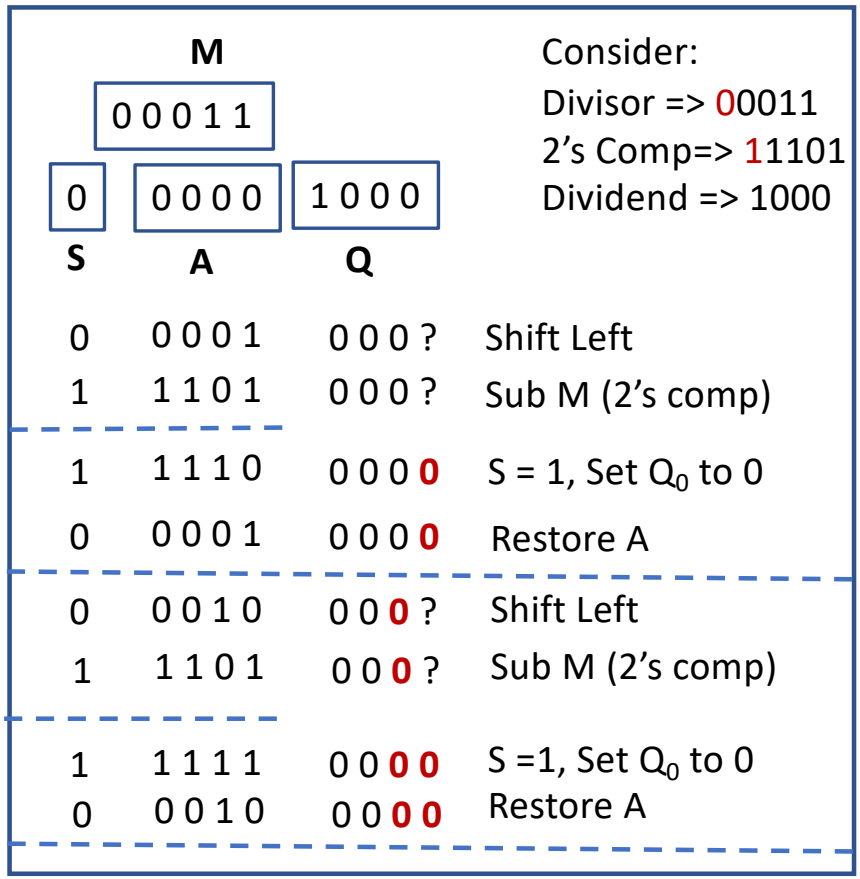
Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

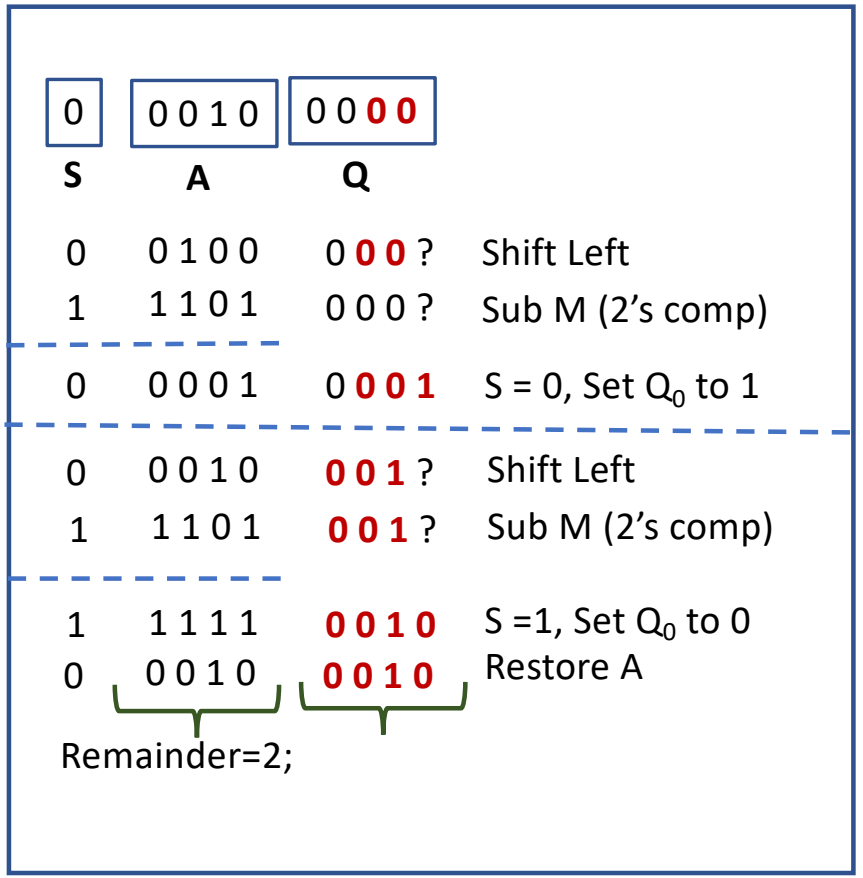
0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0
0	0 0 1 0	**0 0 1 0**	Restore A

Remainder=2;

DIVISION

Restoring Division





DIVISION

Restoring Division

	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

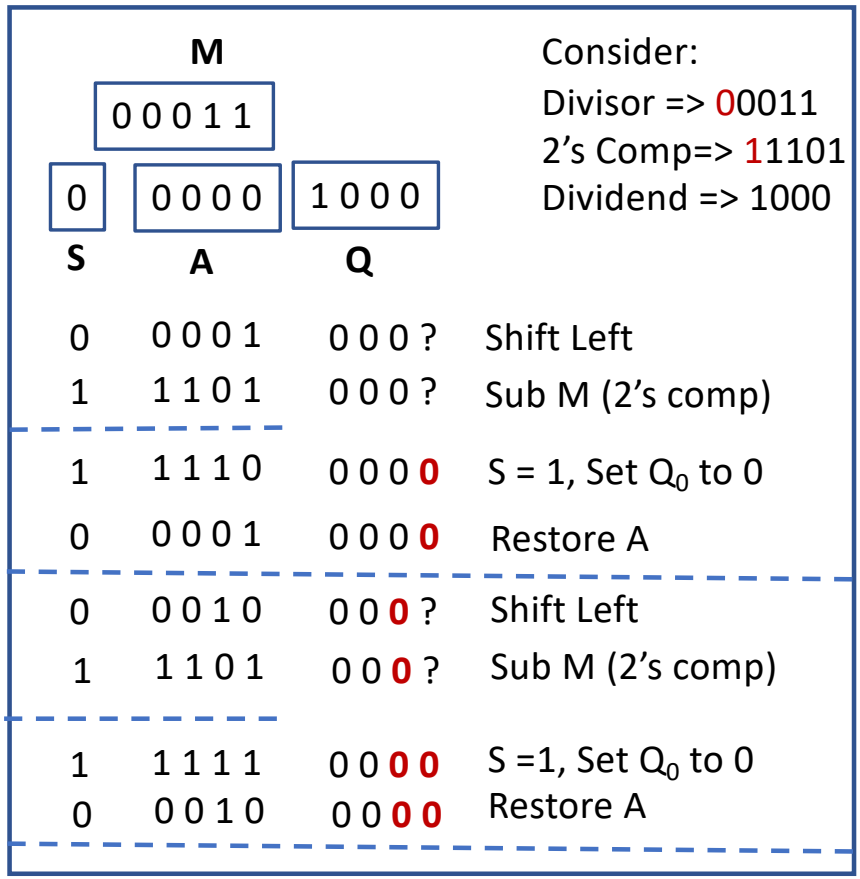
Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0
0	0 0 1 0	**0 0 1 0**	Restore A

Remainder=2; Quotient=2

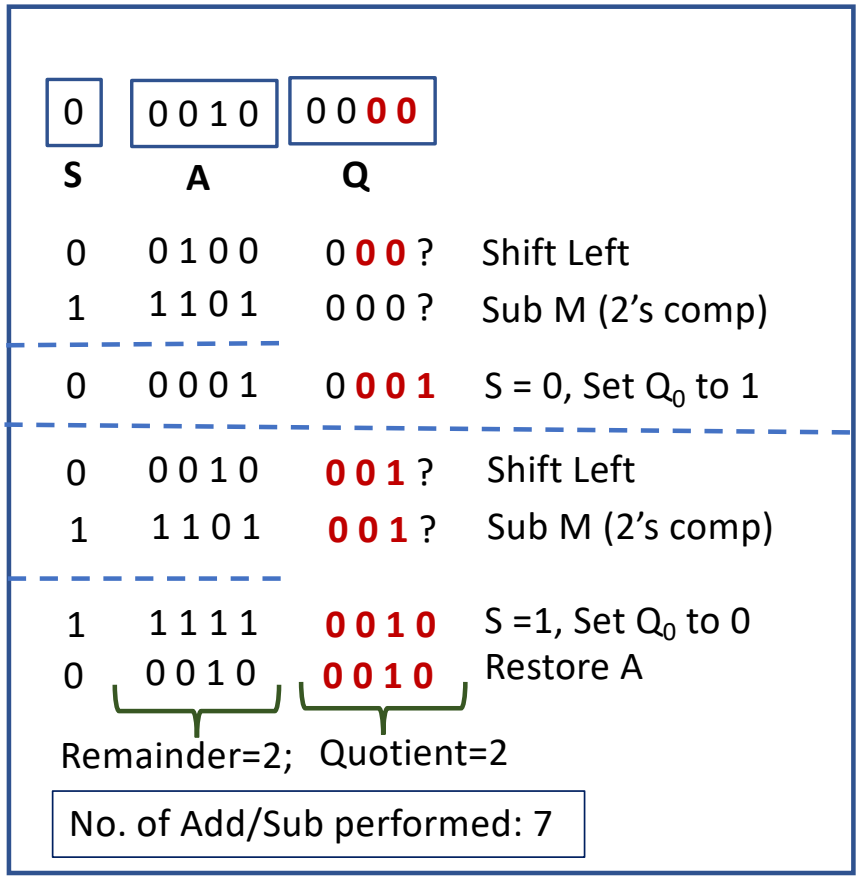
DIVISION

Restoring Division



	M		
	0 0 0 1 1		
0	0 0 0 0	1 0 0 0	
S	A	Q	
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
0	0 0 0 1	0 0 0 **0**	Restore A
0	0 0 1 0	0 0 **0** ?	Shift Left
1	1 1 0 1	0 0 **0** ?	Sub M (2's comp)
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
0	0 0 1 0	0 0 **0 0**	Restore A

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000



0	0 0 1 0	0 0 **0 0**	
S	A	Q	
0	0 1 0 0	0 **0 0** ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S =1, Set Q_0 to 0
0	0 0 1 0	**0 0 1 0**	Restore A

Remainder=2; Quotient=2

No. of Add/Sub performed: 7

DIVISION

Restoring Division

DIVISION

Restoring Division

Observations:

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.
- Avoid the need to restore after an unsuccessful subtraction

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.
- Avoid the need to restore after an unsuccessful subtraction
- If A is positive, we shift left and subtract M => 2A-M

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.
- Avoid the need to restore after an unsuccessful subtraction
- If A is positive, we shift left and subtract M => 2A-M
- If A is negative, we restore A by performing A+M and then shift left and subtract M => 2(A+M)-M = 2A+M

DIVISION

Restoring Division

Observations:

- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.
- Avoid the need to restore after an unsuccessful subtraction
- If A is positive, we shift left and subtract M => 2A-M
- If A is negative, we restore A by performing A+M and then shift left and subtract M => 2(A+M)-M = 2A+M
- 2A+M, represents shift left and add M, when A is negative.

Restoring Division

Observations:

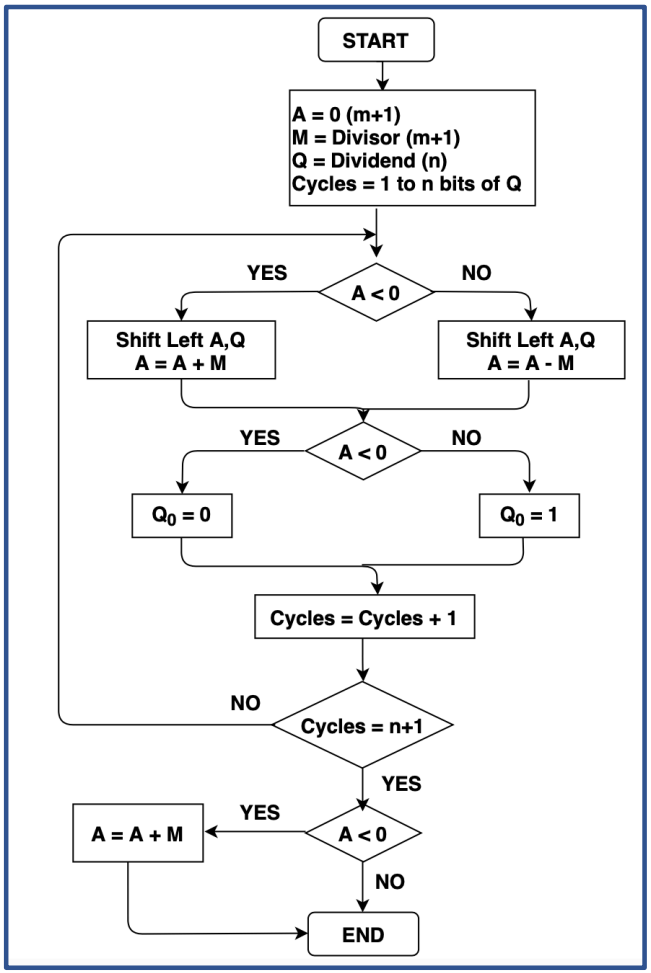
- This method is used for unsigned binary numbers division or positive signed binary numbers division
- There are significant number of addition and subtraction required due to restoration process.
- Avoid the need to restore after an unsuccessful subtraction
- If A is positive, we shift left and subtract M => 2A-M
- If A is negative, we restore A by performing A+M and then shift left and subtract M => 2(A+M)-M = 2A+M
- 2A+M, represents shift left and add M, when A is negative.
- This leads to **Non-Restoring division** algorithm

DIVISION

Non-Restoring Division

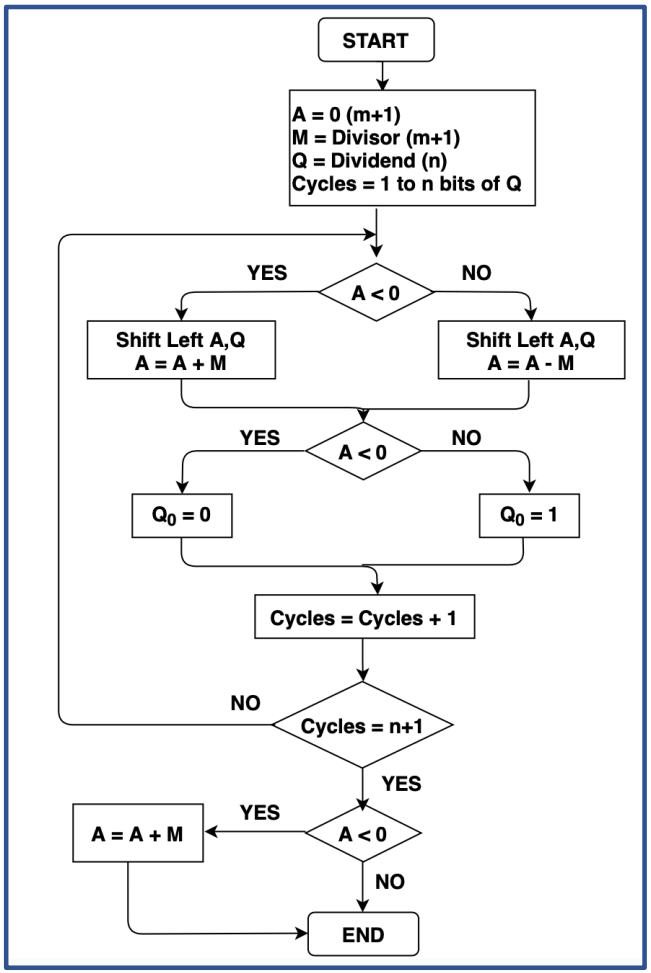
DIVISION

Non-Restoring Division



DIVISION

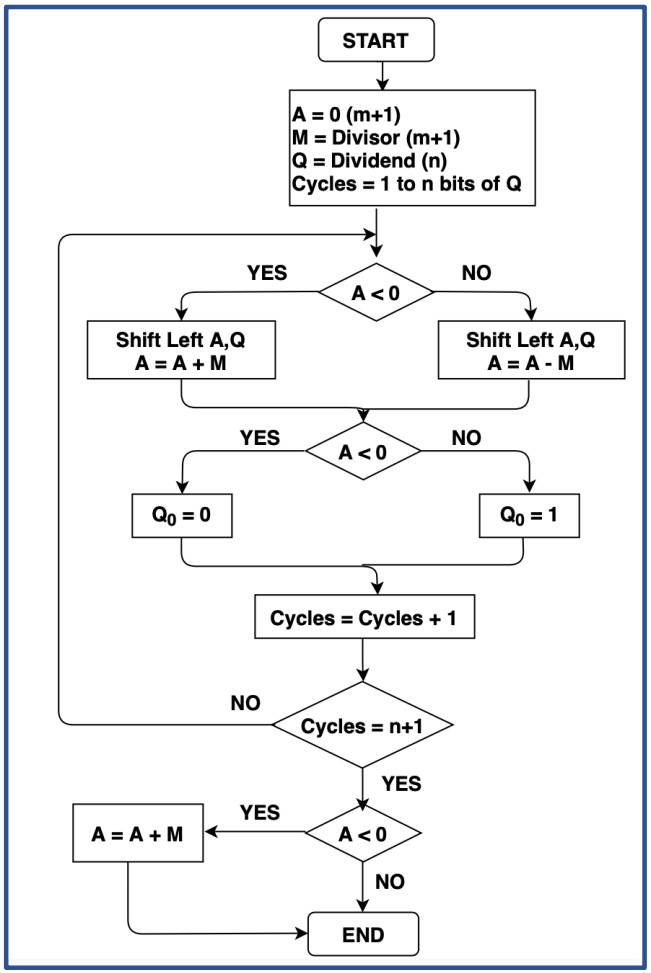
Non-Restoring Division



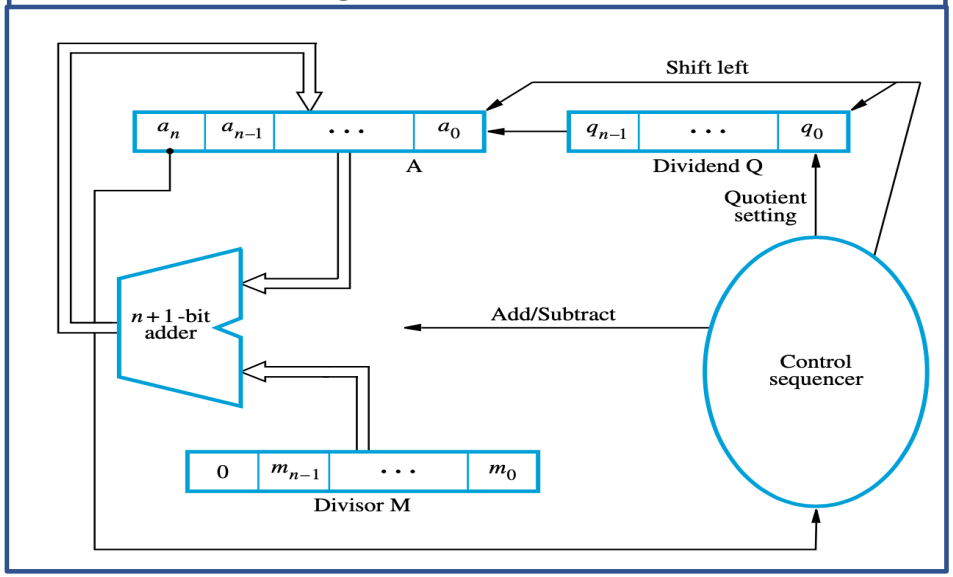
- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub

DIVISION

Non-Restoring Division



- We Require:
  - Accumulator register – A (m+1) bits (Remainder)
  - Dividend register – Q (Quotient) n-bits
  - Divisor register – M (m+1 bit => Sign)
  - n+1-bit Adder
  - Control signals for Shift left and Add/Sub



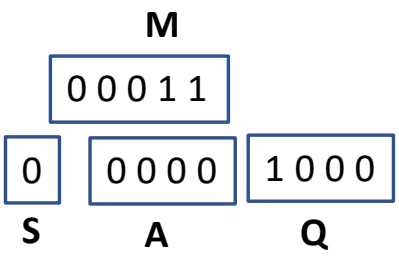
DIVISION

Non-Restoring Division

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

DIVISION

Non-Restoring Division



Consider:

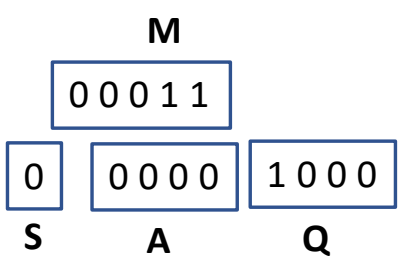
Divisor => 00011

2's Comp=> 11101

Dividend => 1000

DIVISION

Non-Restoring Division



Consider:

Divisor => 00011

2's Comp=> 11101

Dividend => 1000

Shift Left

DIVISION

Non-Restoring Division

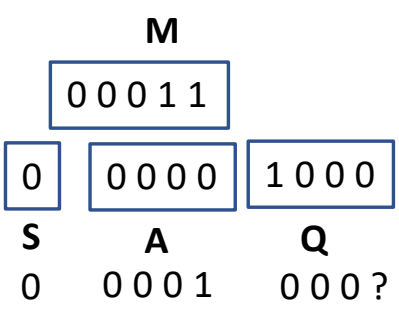
M		
	0 0 0 1 1	
0	0 0 0 0	1 0 0 0
S	A	Q
0	0 0 0 1	0 0 0 ?

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

Shift Left

DIVISION

Non-Restoring Division



Consider:

Divisor => 00011

2's Comp=> 11101

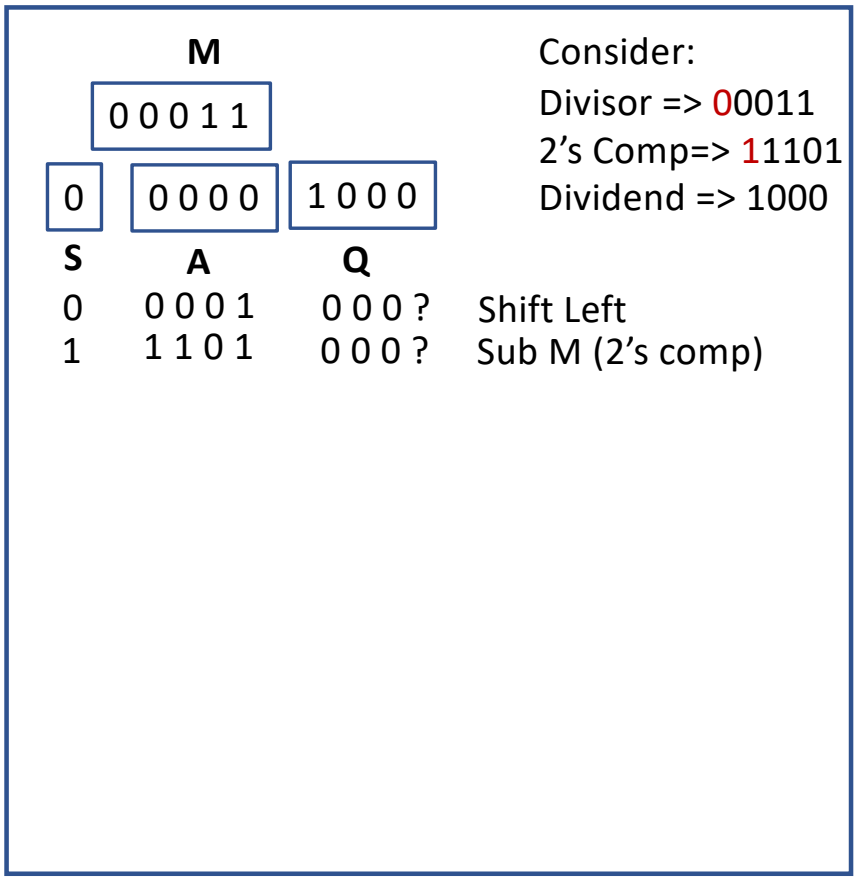
Dividend => 1000

Shift Left

Sub M (2's comp)

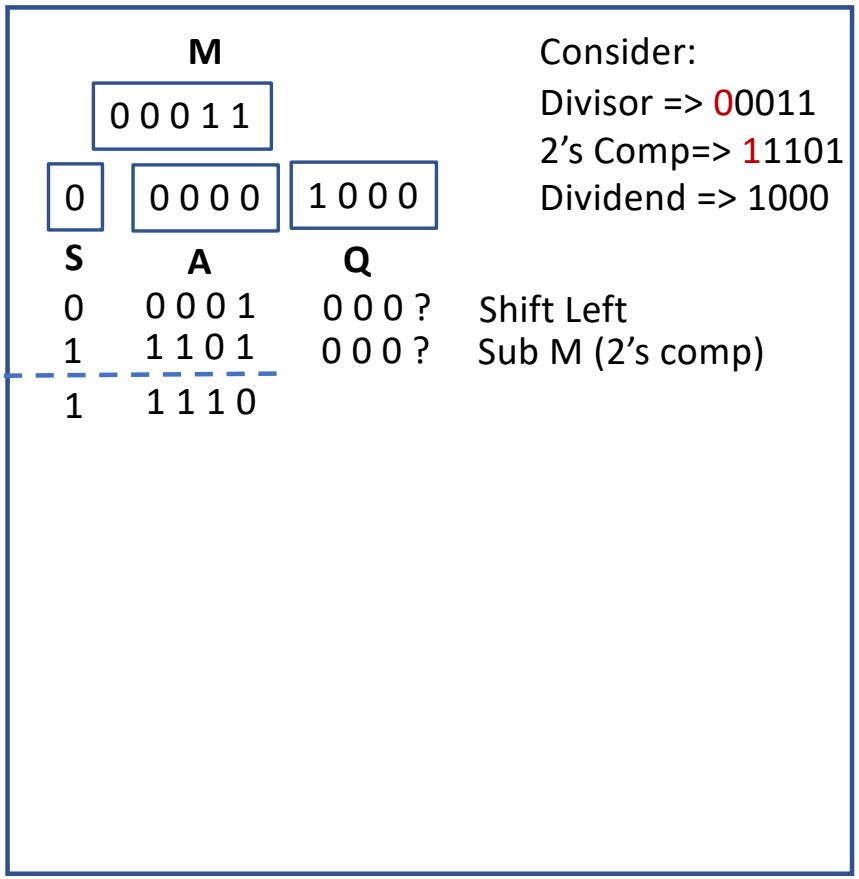
DIVISION

Non-Restoring Division



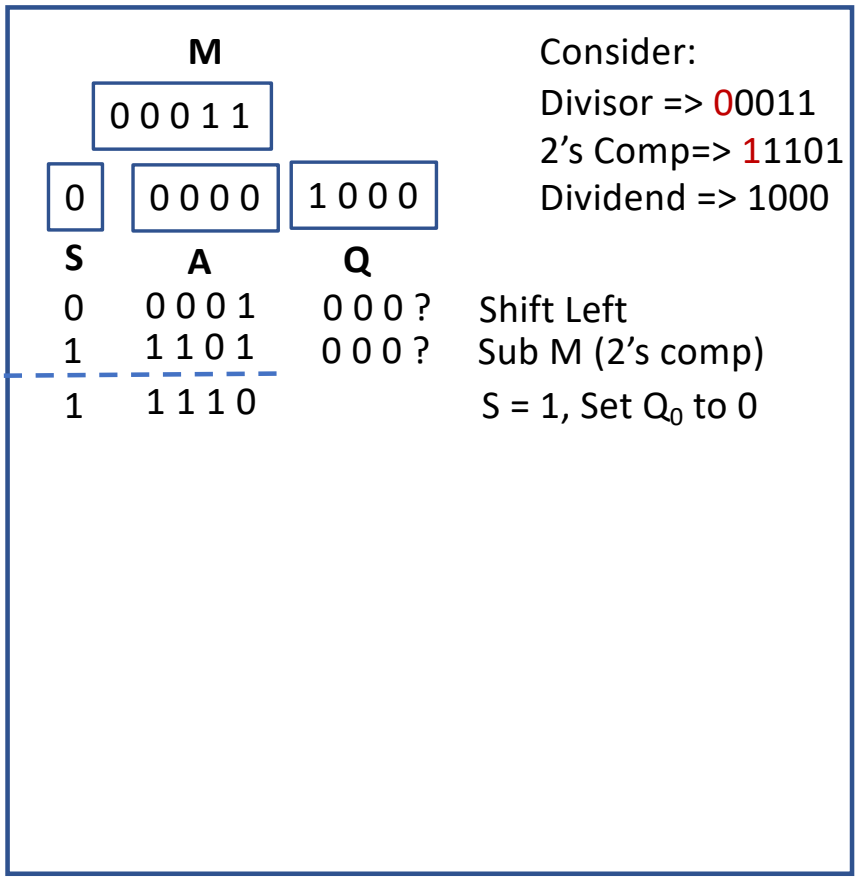
DIVISION

Non-Restoring Division



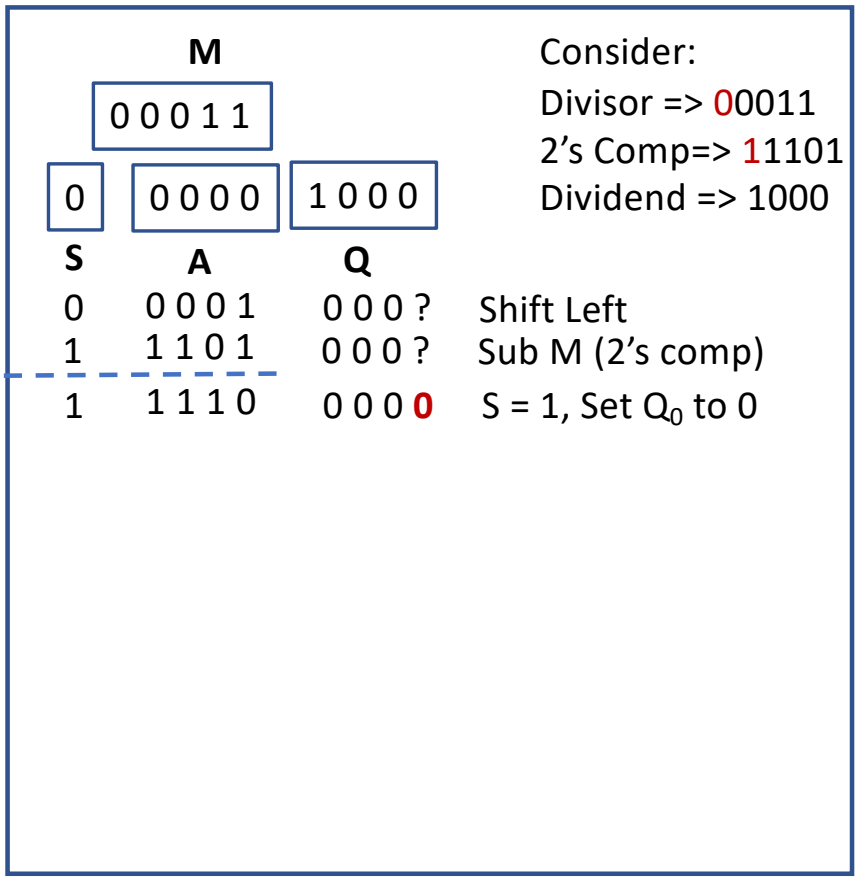
DIVISION

Non-Restoring Division



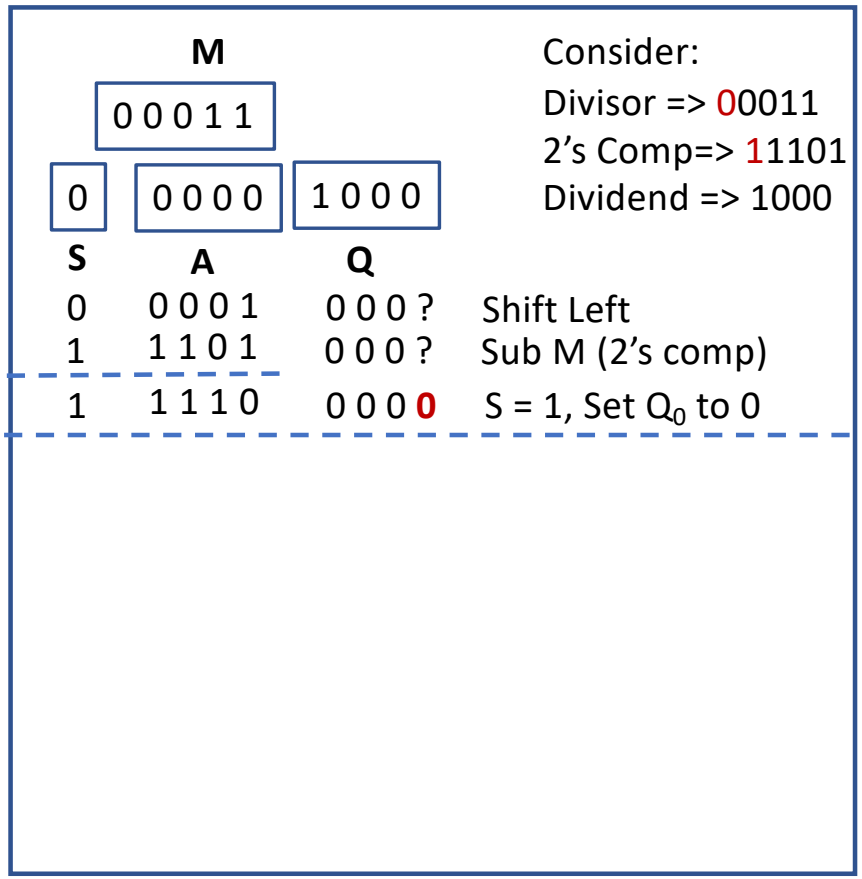
DIVISION

Non-Restoring Division



DIVISION

Non-Restoring Division



Consider:

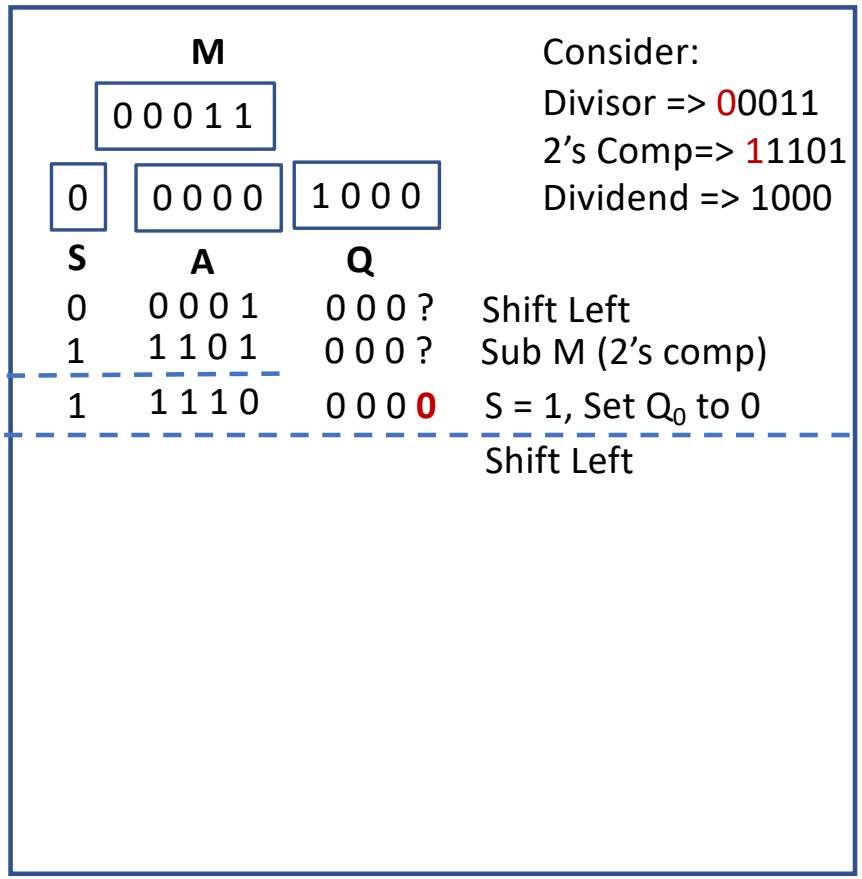
Divisor => 00011

2's Comp=> 11101

Dividend => 1000

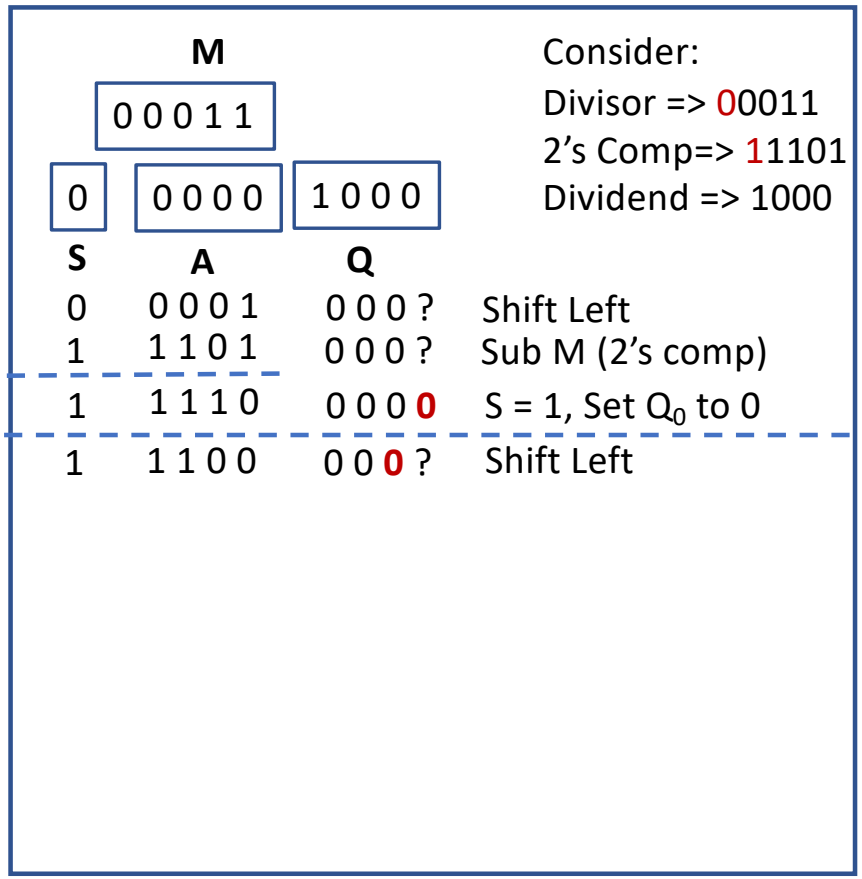
DIVISION

Non-Restoring Division



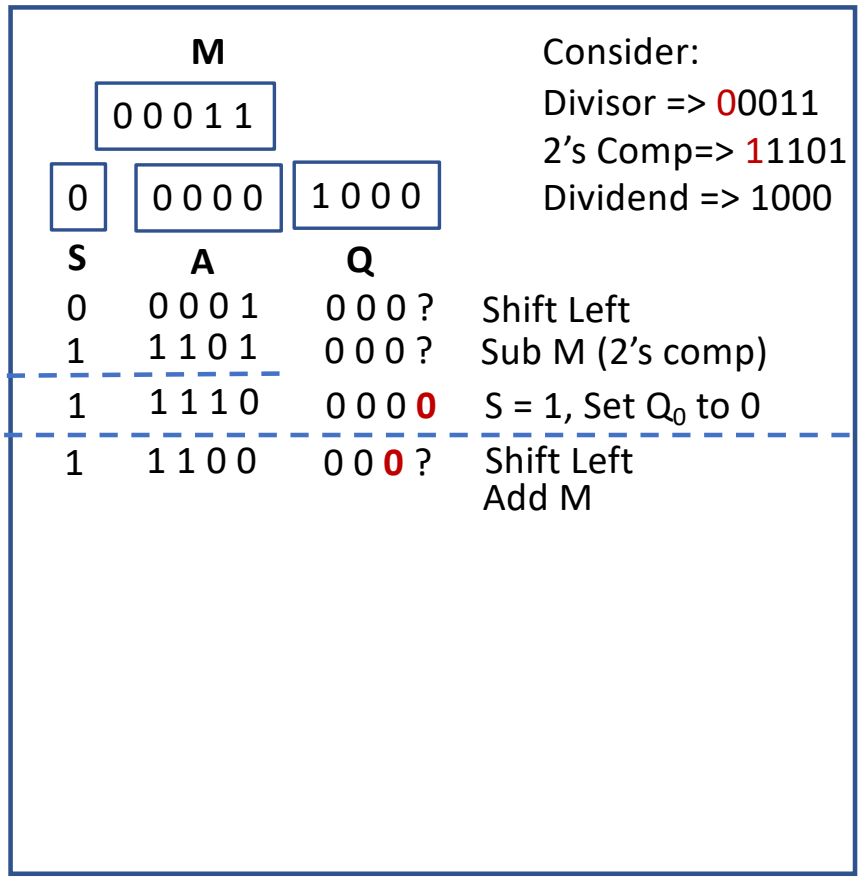
DIVISION

Non-Restoring Division



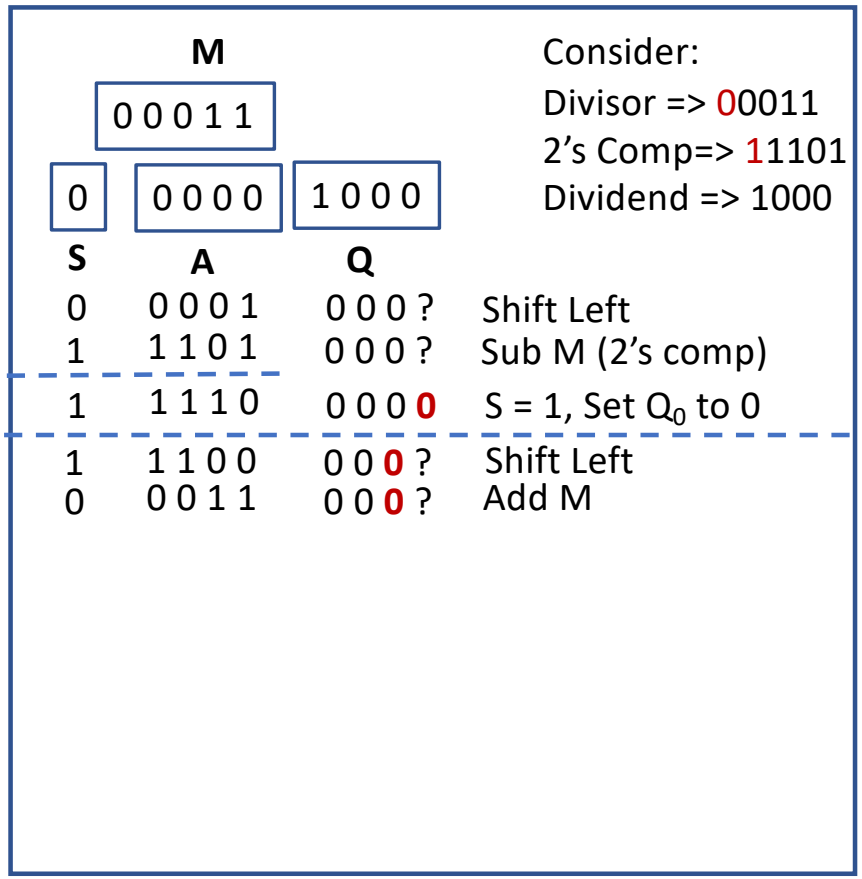
DIVISION

Non-Restoring Division



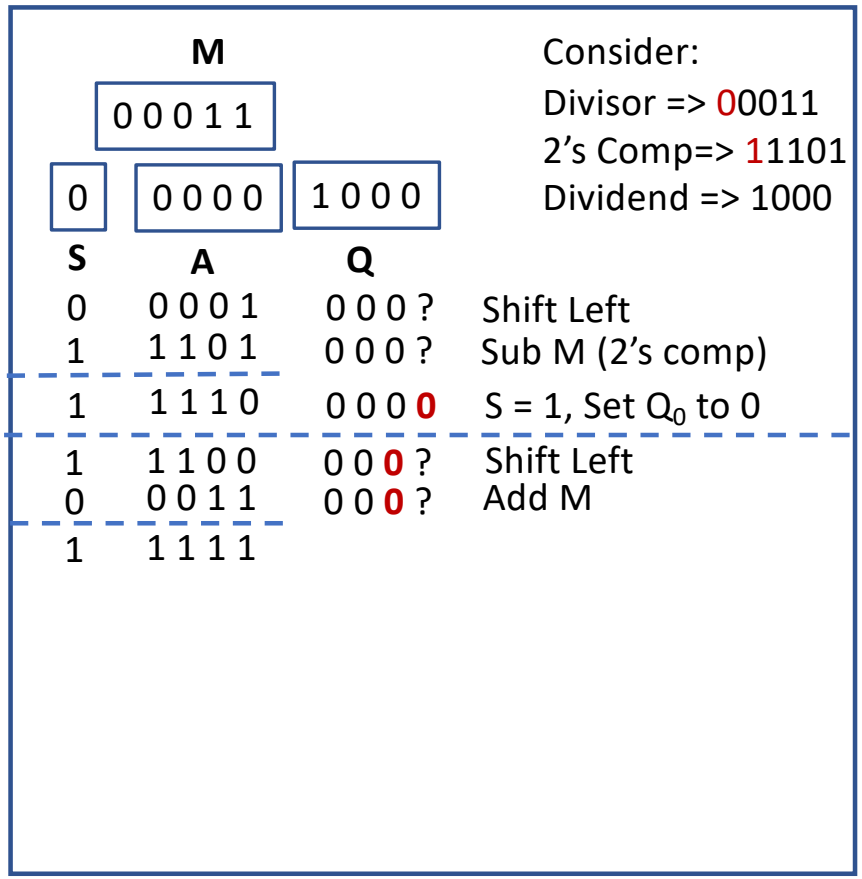
DIVISION

Non-Restoring Division



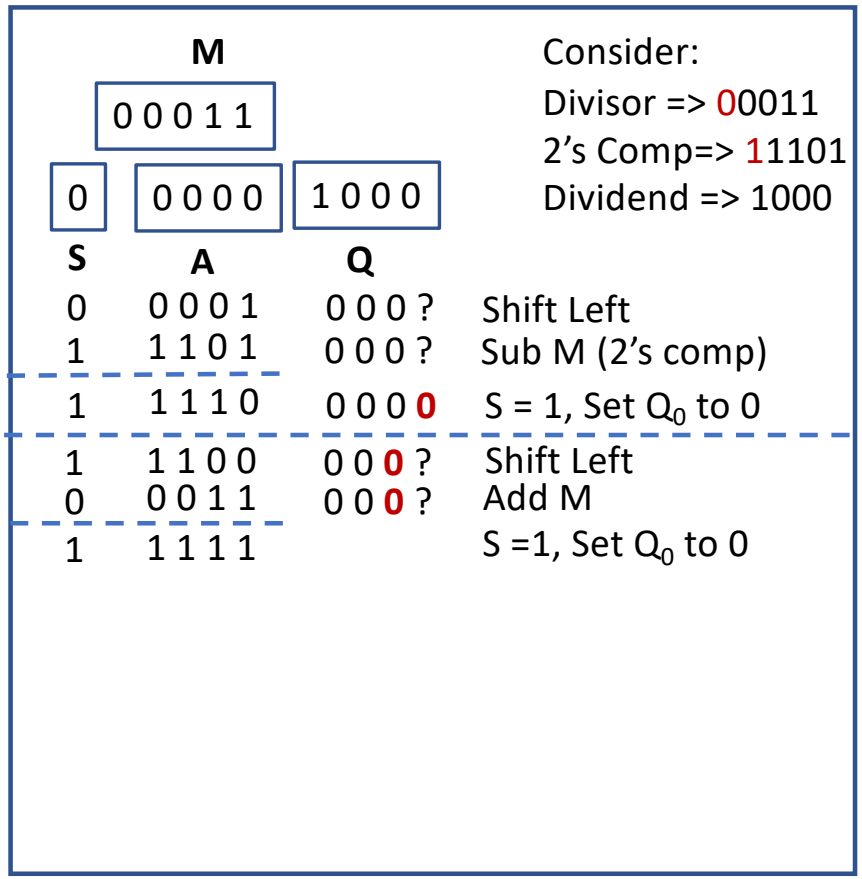
DIVISION

Non-Restoring Division



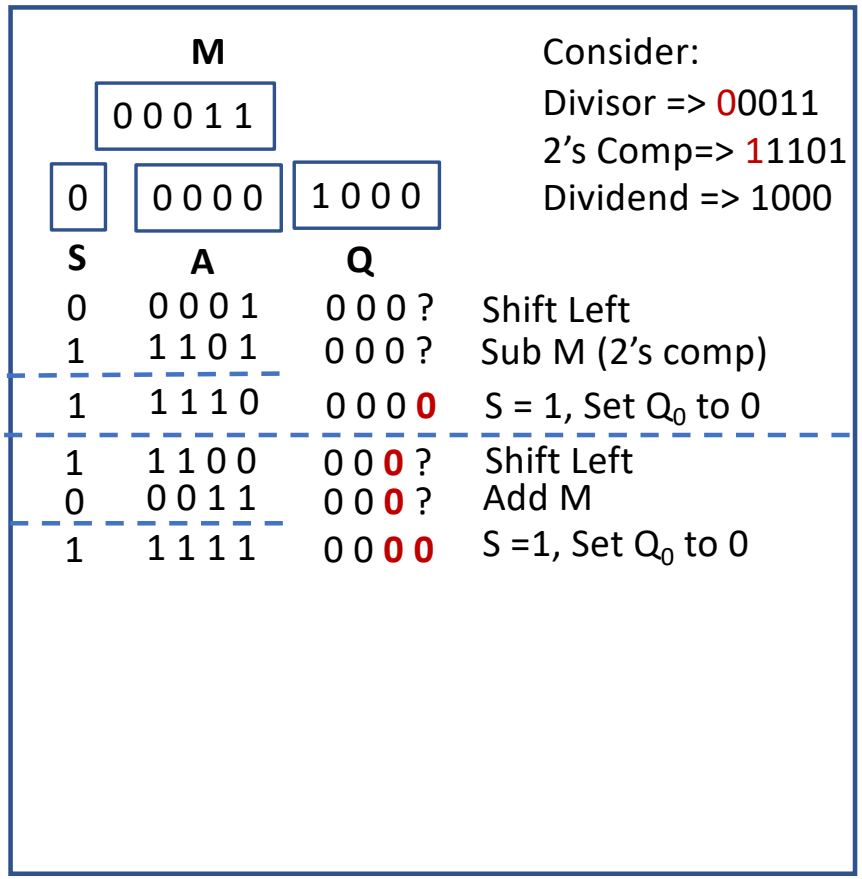
DIVISION

Non-Restoring Division



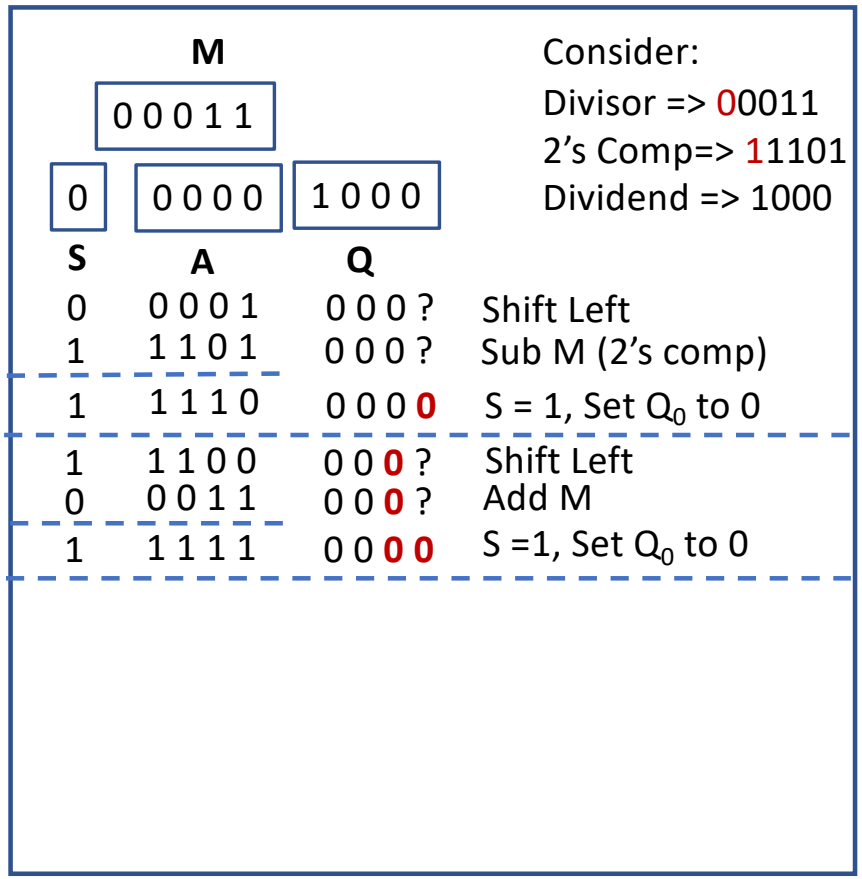
DIVISION

Non-Restoring Division



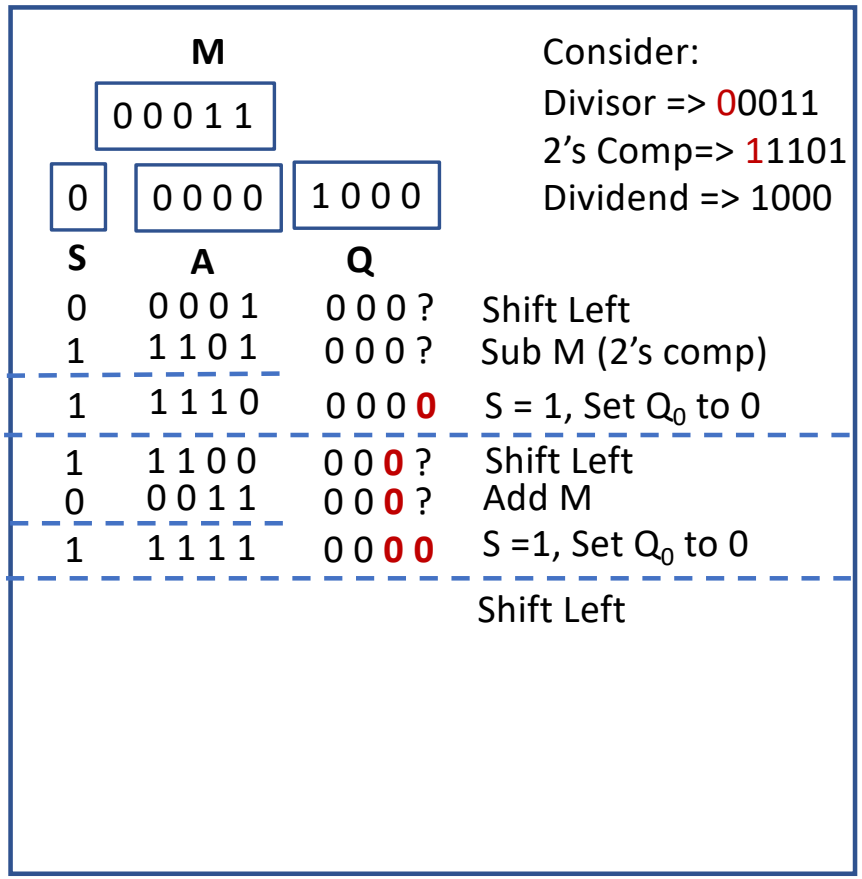
DIVISION

Non-Restoring Division



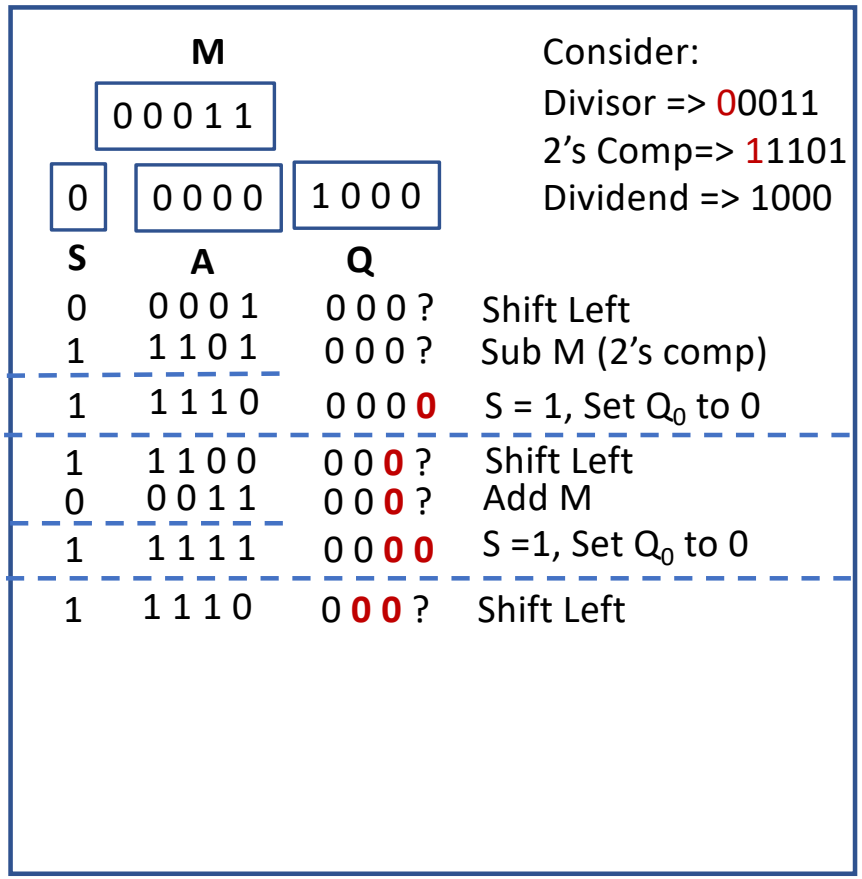
DIVISION

Non-Restoring Division



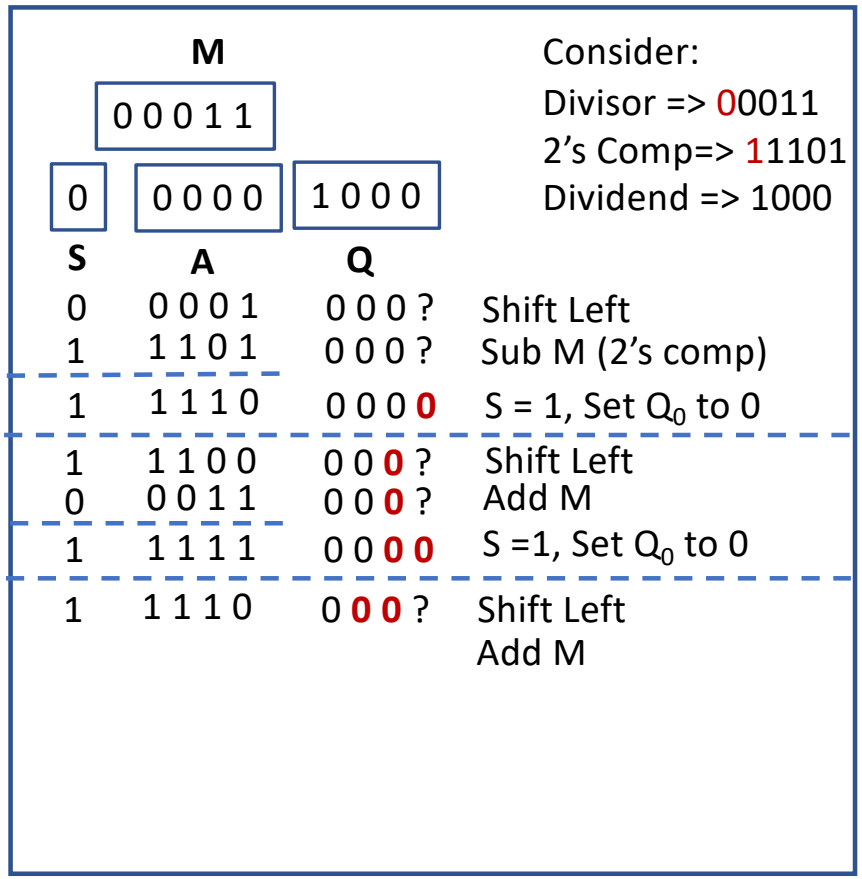
DIVISION

Non-Restoring Division



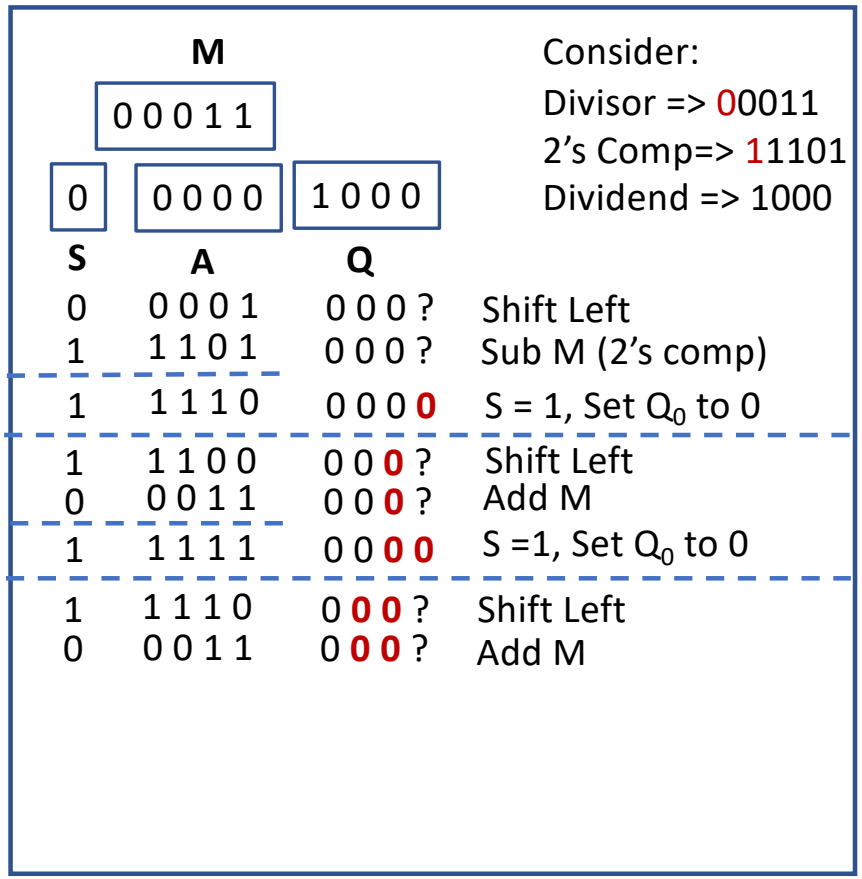
DIVISION

Non-Restoring Division



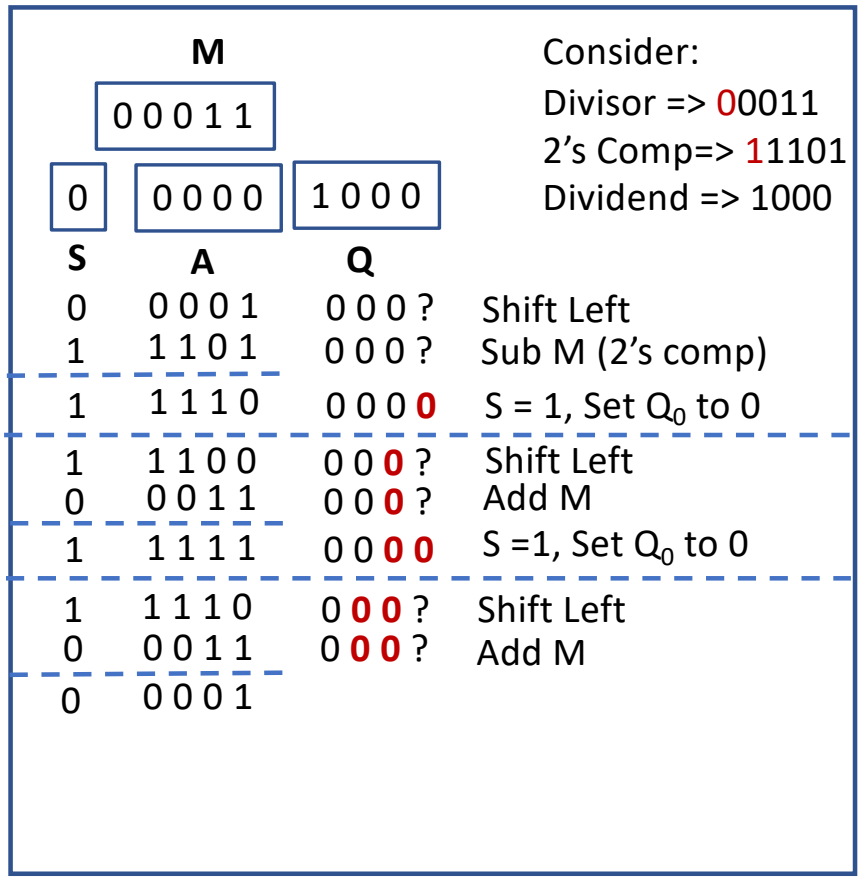
DIVISION

Non-Restoring Division



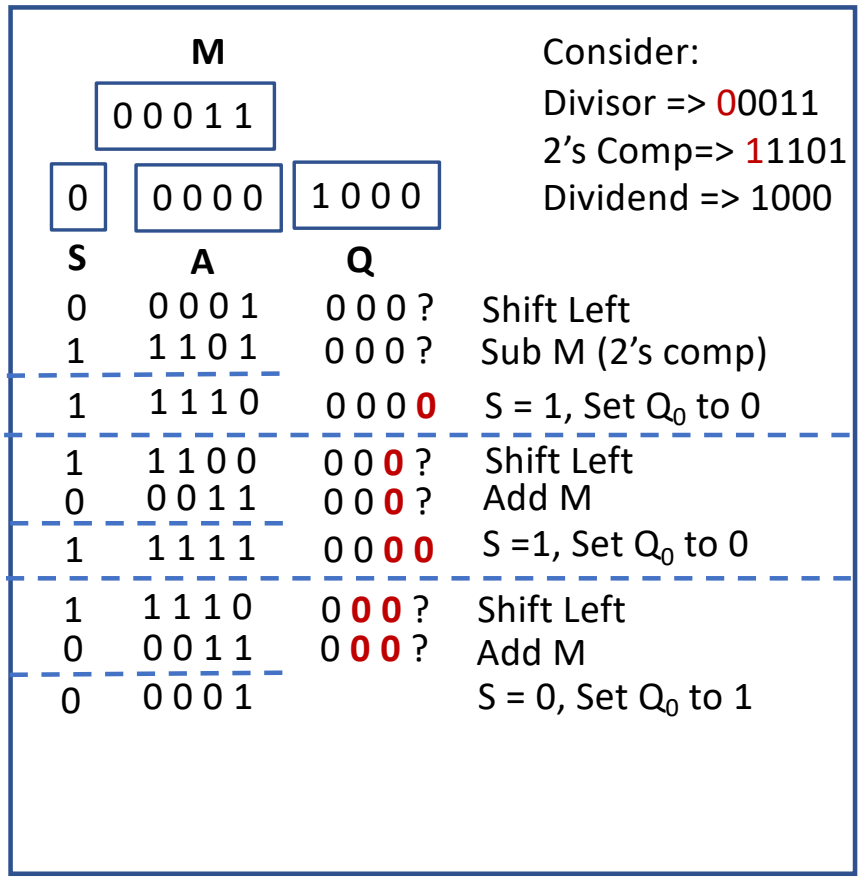
DIVISION

Non-Restoring Division



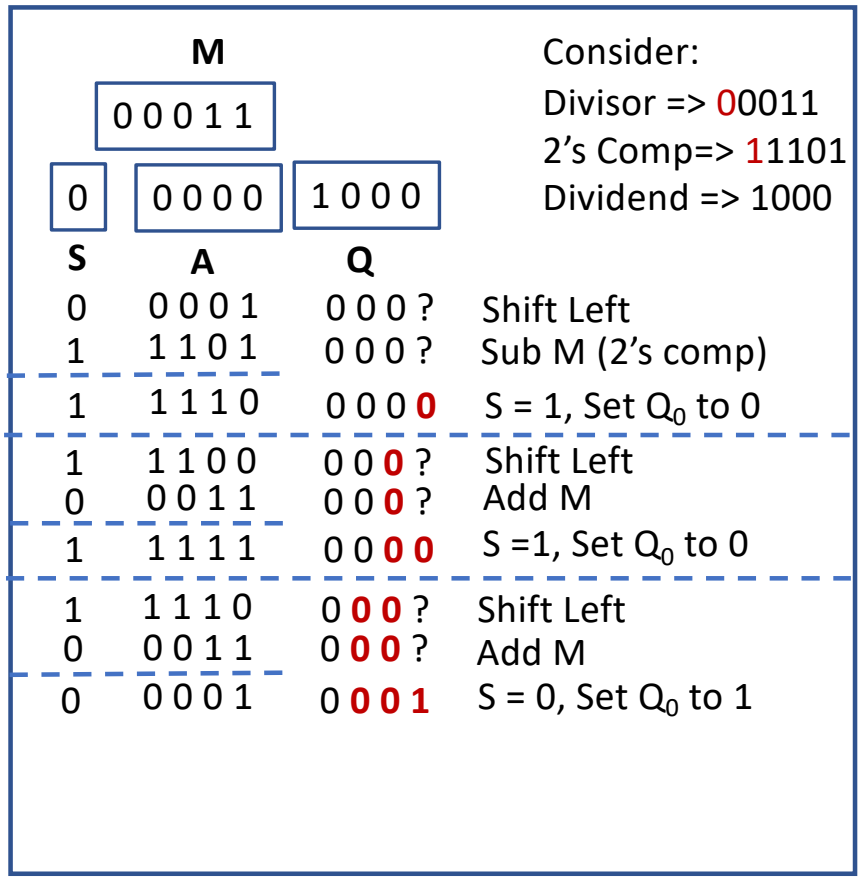
DIVISION

Non-Restoring Division



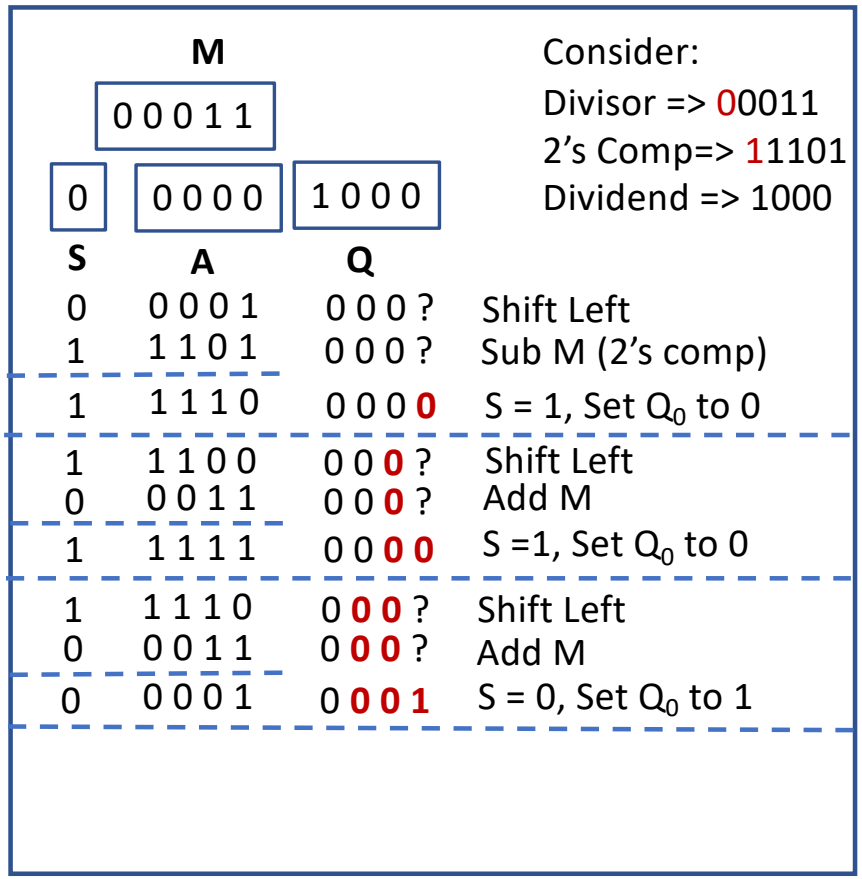
DIVISION

Non-Restoring Division



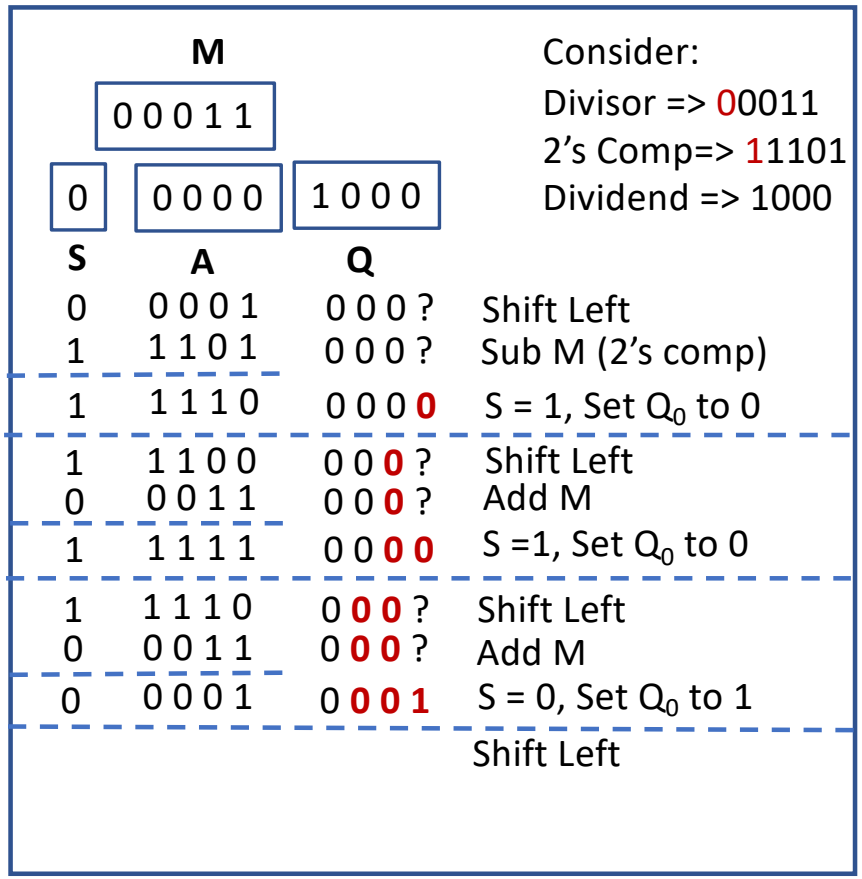
DIVISION

Non-Restoring Division



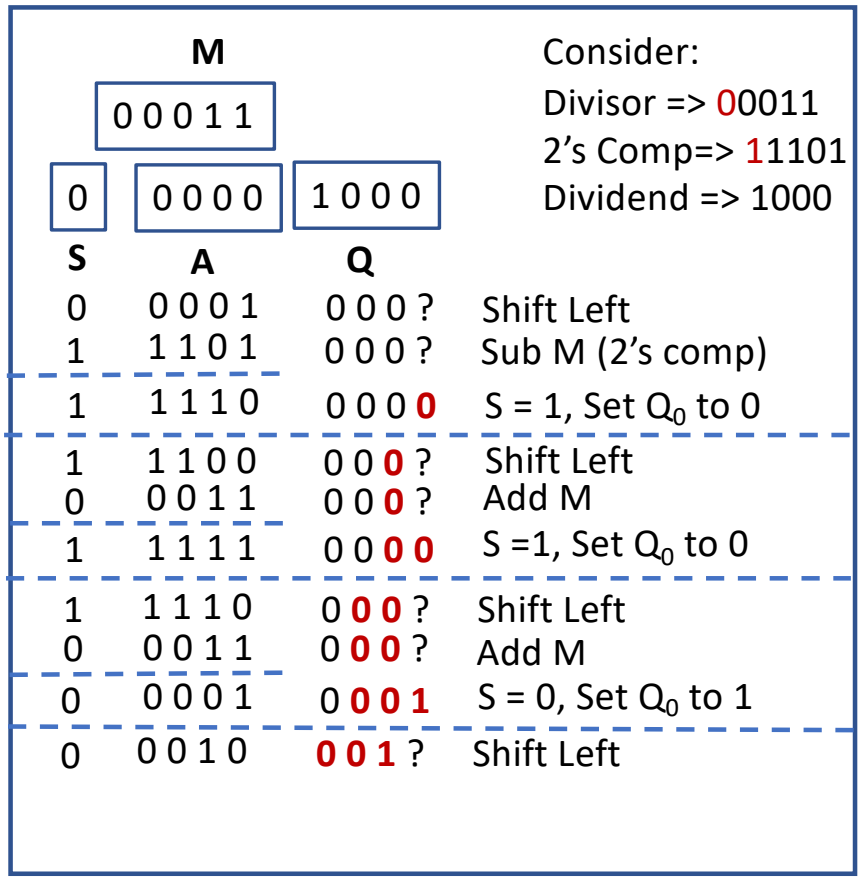
DIVISION

Non-Restoring Division



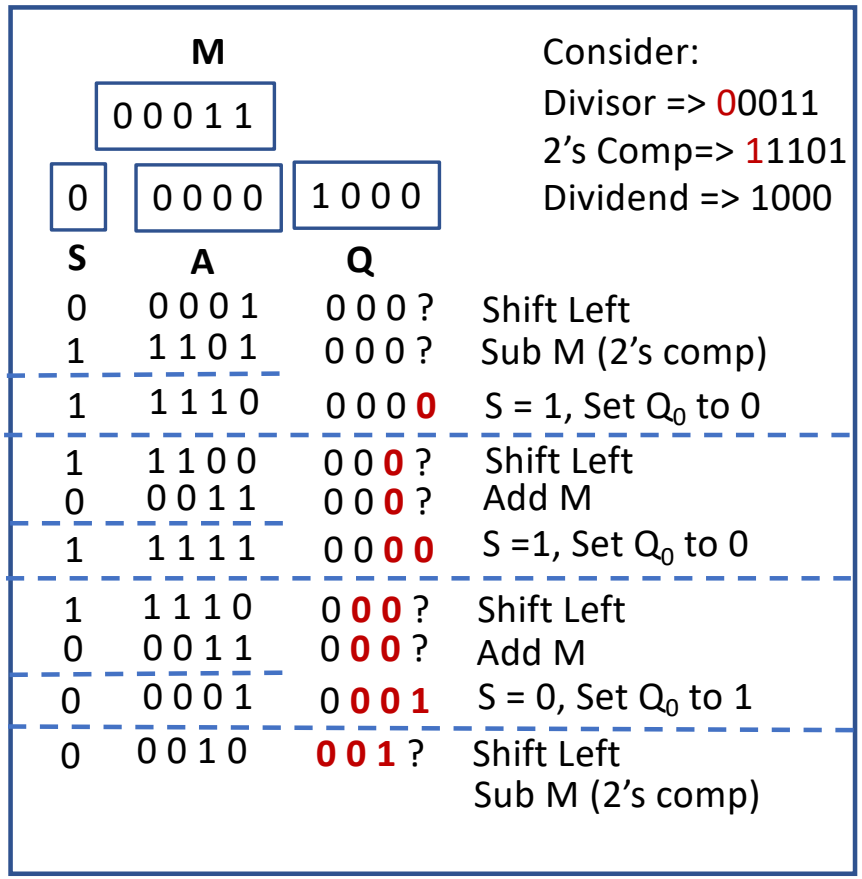
DIVISION

Non-Restoring Division



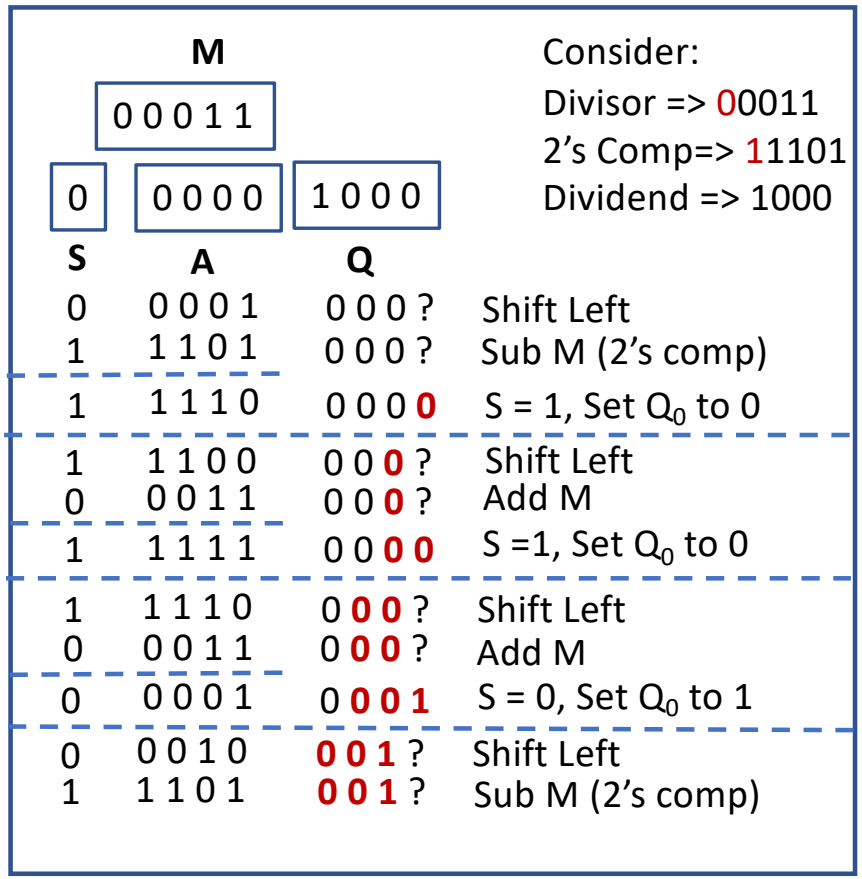
DIVISION

Non-Restoring Division



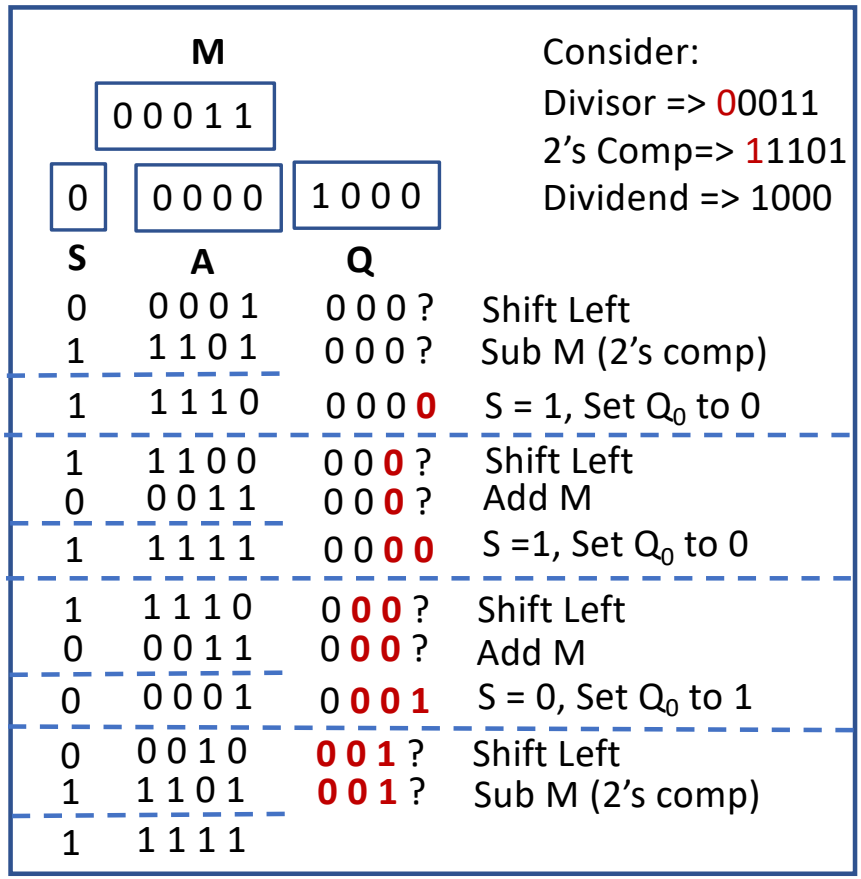
DIVISION

Non-Restoring Division



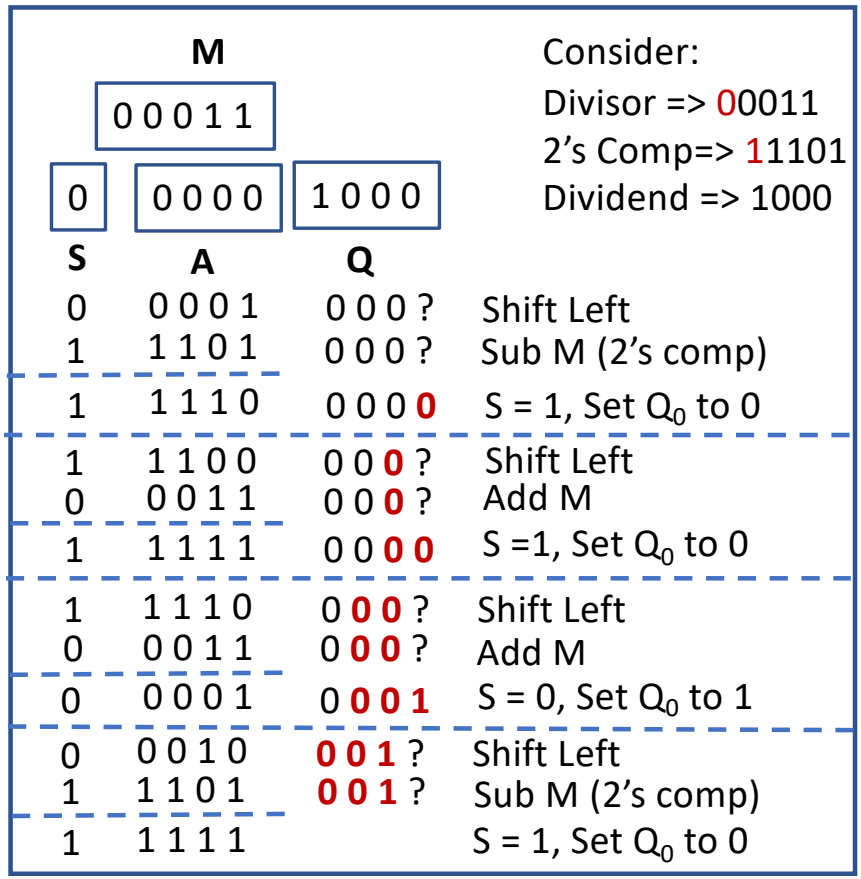
DIVISION

Non-Restoring Division



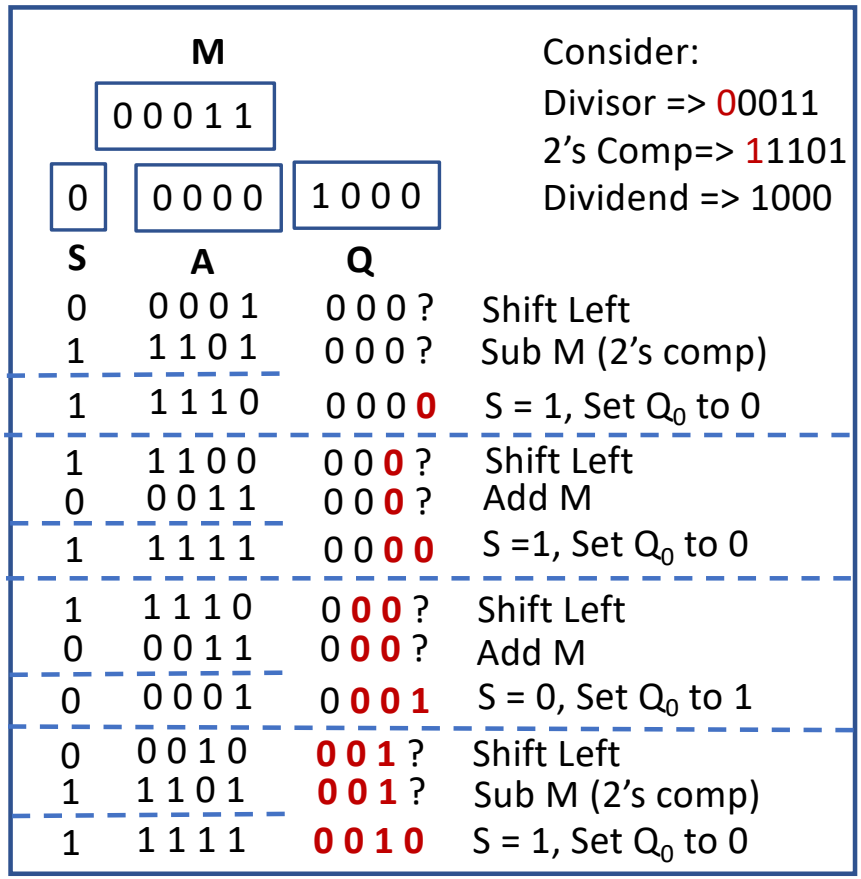
DIVISION

Non-Restoring Division



DIVISION

Non-Restoring Division



DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q
1	1 1 1 1	**0 0 1 0**

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q
1	1 1 1 1	**0 0 1 0**
1	1 1 1 1	**0 0 1 0**

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
1	1 1 1 1	**0 0 1 0**	Restore A

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1		Restore A

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

DIVISION

Non-Restoring Division

M				
	0 0 0 1 1			
0	0 0 0 0	1 0 0 0		
S	A	Q		
0	0 0 0 1	0 0 0 ?	Shift Left	
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)	
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0	
1	1 1 0 0	0 0 **0** ?	Shift Left	
0	0 0 1 1	0 0 **0** ?	Add M	
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0	
1	1 1 1 0	0 **0 0** ?	Shift Left	
0	0 0 1 1	0 **0 0** ?	Add M	
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1	
0	0 0 1 0	**0 0 1** ?	Shift Left	
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)	
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0	

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

1	1 1 1 1	**0 0 1 0**	
S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

DIVISION

Non-Restoring Division

	M			
	0 0 0 1 1			
0	0 0 0 0	1 0 0 0		Consider:
S	A	Q		Divisor => 00011
0	0 0 0 1	0 0 0 ?	Shift Left	2's Comp=> 11101
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)	Dividend => 1000
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0	
1	1 1 0 0	0 0 **0** ?	Shift Left	
0	0 0 1 1	0 0 **0** ?	Add M	
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0	
1	1 1 1 0	0 **0 0** ?	Shift Left	
0	0 0 1 1	0 **0 0** ?	Add M	
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1	
0	0 0 1 0	**0 0 1** ?	Shift Left	
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)	
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0	

1	1 1 1 1	**0 0 1 0**	
S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

Remainder=2;

DIVISION

Non-Restoring Division

	M			
	0 0 0 1 1			
0	0 0 0 0	1 0 0 0		Consider:
S	A	Q		Divisor => 00011
0	0 0 0 1	0 0 0 ?	Shift Left	2's Comp=> 11101
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)	Dividend => 1000
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0	
1	1 1 0 0	0 0 **0** ?	Shift Left	
0	0 0 1 1	0 0 **0** ?	Add M	
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0	
1	1 1 1 0	0 **0 0** ?	Shift Left	
0	0 0 1 1	0 **0 0** ?	Add M	
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1	
0	0 0 1 0	**0 0 1** ?	Shift Left	
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)	
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0	

1	1 1 1 1	**0 0 1 0**	
S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

Remainder=2;

DIVISION

Non-Restoring Division

S	A	Q	M = 00011
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

Consider:
Divisor => 00011
2's Comp=> 11101
Dividend => 1000

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

Remainder=2; Quotient=2

DIVISION

Non-Restoring Division

S	A	Q	
	M = 0 0 0 1 1		Consider: Divisor => 00011
			2's Comp=> 11101
0	0 0 0 0	1 0 0 0	Dividend => 1000
0	0 0 0 1	0 0 0 ?	Shift Left
1	1 1 0 1	0 0 0 ?	Sub M (2's comp)
1	1 1 1 0	0 0 0 **0**	S = 1, Set Q_0 to 0
1	1 1 0 0	0 0 **0** ?	Shift Left
0	0 0 1 1	0 0 **0** ?	Add M
1	1 1 1 1	0 0 **0 0**	S =1, Set Q_0 to 0
1	1 1 1 0	0 **0 0** ?	Shift Left
0	0 0 1 1	0 **0 0** ?	Add M
0	0 0 0 1	0 **0 0 1**	S = 0, Set Q_0 to 1
0	0 0 1 0	**0 0 1** ?	Shift Left
1	1 1 0 1	**0 0 1** ?	Sub M (2's comp)
1	1 1 1 1	**0 0 1 0**	S = 1, Set Q_0 to 0

S	A	Q	
1	1 1 1 1	**0 0 1 0**	
0	0 0 1 1	**0 0 1 0**	Restore A
0	0 0 1 0	**0 0 1 0**	

Remainder=2; Quotient=2

No. of Add/Sub performed: 5

Non-Restoration Division

Points to be noted :

- Both Restoration and Non Restoration Division works for unisigned and positive signed numbers
- For signed numbers, convert the numbers to positive and adjust the sign later
- Simple solution Sign of the result will be XOR function of signs of Divisor and Dividend
- Ex: $\mathbf{7 \div 2 = 3\ and\ the\ remainder\ is\ 1}$

  $\mathbf{-7 \div 2 = -3\ and\ the\ remainder\ is\ -1}$

DIVISION

Think about it

- Compare the number of addition / subtraction for the following operation using restoration and non-restoration methods:

  111110 ÷ 0111

  1000010 ÷ 0101

- For n-bits division what will be the worst case in terms of number of additions and subtractions for restoration and non-restoration division?

THANK YOU

Sudarshan T S B. Ph.D.,
Department of Computer Science & Engineering

sudarshan@pes.edu

+91 80 6666 3333 Extn 215