COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code : CSE 2151

Credits : 04

MULTIPLICATION: UNSIGNED V/S SIGNED

1001	(9)
× 0011	(3)
00001001	1001 × 2^0
00010010	1001 × 2^1
00011011	(27)

(a) Unsigned integers

1001	(−7)
× 0011	(3)
11111001	(−7) × 2^0 = (−7)
11110010	(−7) × 2^1 = (−14)
11101011	(−21)

(b) Twos complement integers

Figure 10.11 Comparison of Multiplication of Unsigned and Twos Complement Integers

MULTIPLICATION: BOOTH'S ALGORITHM

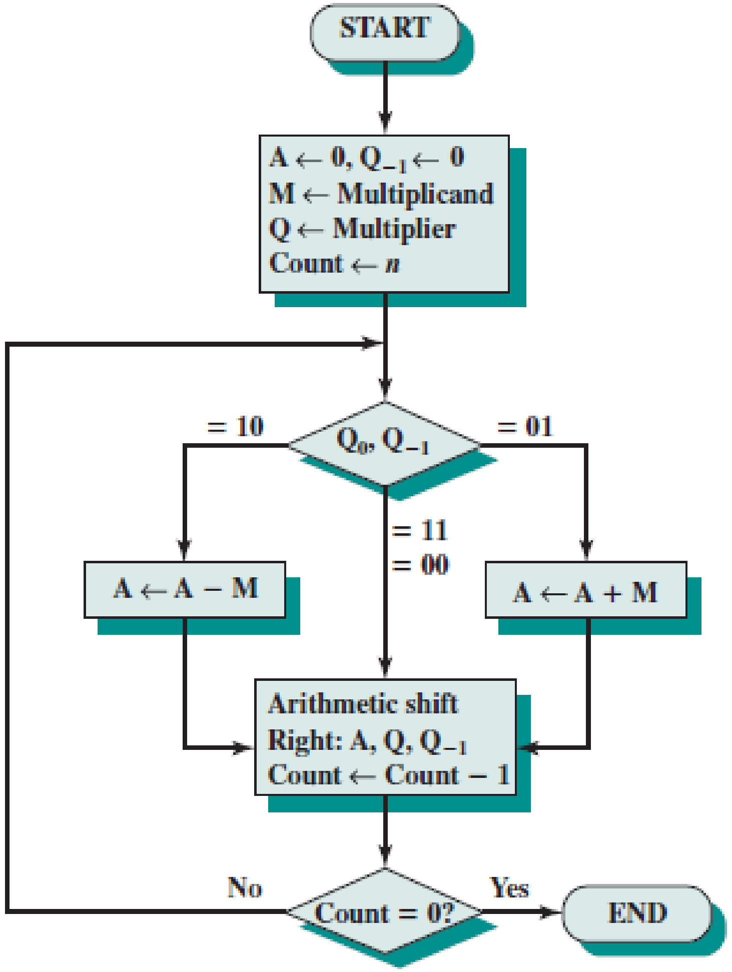

Figure 10.12 Booth's Algorithm for Twos Complement Multiplication

MULTIPLICATION: BOOTH'S ALGORITHM

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A ← A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Figure 10.13 Example of Booth's Algorithm (7 × 3)

BOOTH'S ALGORITHM: 13X-6

- M=13=01101 Q=-6=11010 -M=10011

A	Q	Q_{-1}	M	
00000	11010	0	01101	Initial values
00000	01101	0	01101	Shift Right-1st cycle
10011 10011 *11001*	 01101 *10110*	 0 *1*	 01101 01101	A=A-M [Subtract M from A(Adding 2's complement of M)] *Shift Right- 2nd cycle*
01101 00110 *00011*	 10110 *01011*	 1 *0*	 01101	A=A+M Shift Right- 3rd cycle
10011 10110 *11011*	 01011 *00101*	 0 *1*	 01101 01101	A=A-M [Subtract M from A(Adding 2's complement of M)] Shift Right- 4th cycle
11101	**10010**	1	01101	Shift Right 5th cycle
Taking 2's complement	**0001001110→78** **Product=-78**			

BOOTH'S ALGORITHM: 23X29

- M=23=010111 Q=29=011101 -M=101001

A	Q	Q_{-1}	M	
000000	011101	0	010111	Initial values
101001 101001 110100	 011101 101110	 0 1	 010111	 A=A-M Shift Right-1st cycle
010111 001011 000101	 101110 110111	 1 0	 010111	 A=A+M *Shift Right- 2nd cycle*
101001 101110 110111	 110111 011011	 0 1	 010111	 A=A-M Shift Right- 3rd cycle
111011	101101	1	010111	Shift Right- 4th cycle
111101	110110	1	010111	Shift Right 5th cycle
010111 010100 001010	 110110 011011	 1 0	 010111	 A=A+M *Shift Right- 6th cycle*
	001010011011→667 **Product=667**			

MULTIPLICATION: BOOTH'S ALGORITHM

```
    0111
  × 0011      (0)
--------
11111001      1−0
0000000       1−1
000111        0−1
--------
00010101      (21)
```

(a) (7) × (3) = (21)

```
    0111
  × 1101      (0)
--------
11111001      1−0
0000111       0−1
111001        1−0
--------
11101011      (−21)
```

(b) (7) × (−3) = (−21)

```
    1001
  × 0011      (0)
--------
00000111      1−0
0000000       1−1
111001        0−1
--------
11101011      (−21)
```

(c) (−7) × (3) = (−21)

```
    1001
  × 1101      (0)
--------
00000111      1−0
1111001       0−1
000111        1−0
--------
00010101      (21)
```

(d) (−7) × (−3) = (21)

Figure 10.14 Examples Using Booth's Algorithm

HOW BOOTH'S ALGORITHM WORKS: +VE MULTIPLIER

- Consider the case of a positive multiplier consisting of one block of 1s surrounded by 0s

$$M * (00011110) = M * (2^4 + 2^3 + 2^2 + 2^1) = M * (16 + 8 + 4 + 2)$$
$$= M * 30$$
$$M * (00011110) = M * (2^5 - 2^1) = M * (32 - 2)$$
$$= M * 30$$

- In general, $2^n + 2^{n-1} + \cdots + 2^{n-K} = 2^{n+1} - 2^{n-K}$ **(10.3)**

- the product can be generated by one addition(Adding the content of 2^5 place value) and one subtraction (Subtracting the content of 2^1 place value) of the multiplicand.
- Booth's algorithm conforms to this scheme by performing a subtraction when the first 1 of the block is encountered (1–0) and an addition when the end of the block is encountered (0–1).

$$M * (01111010) = M * (2^6 + 2^5 + 2^4 + 2^3 + 2^1)$$
$$= M * (2^7 - 2^3 + 2^2 - 2^1)$$

HOW BOOTH'S ALGORITHM WORKS: -VE MULTIPLIER

- Let X be a negative number in twos complement notation: $X = \{1x_{n-2}x_{n-3} \ldots.. x_1x_0\}$
- Then the value of X can be expressed as follows:

$$X = -2^{n-1} + (x_{n-2} \times 2^{n-2}) + (x_{n-3} \times 2^{n-3}) + \cdots + (x_1 \times 2^1) + (x_0 \times 2^0) \quad \textbf{(10.4)}$$

- The leftmost bit of X is 1, because X is negative. Assume that the leftmost 0 is in the k^{th} position. Then,

$$X = \{111 \ldots 10x_{k-1}x_{k-2} \ldots x_1x_0\} \quad \textbf{(10.5)}$$

- And the value of X is: $X = -2^{n-1} + 2^{n-2} + \cdots + 2^{k+1} + (x_{k-1} \times 2^{k-1}) + \cdots + (x_0 \times 2^0)$ **(10.6)**

- From Equation $2^n + 2^{n-1} + \cdots + 2^{n-K} = 2^{n+1} - 2^{n-K}$ **(10.3)**
- we can say that: $2^{n-2} + 2^{n-3} + \ldots + 2^{k+1} = 2^{n-1} - 2^{k+1}$
- Rearranging: $-2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^{k+1} = -2^{k+1}$ (10.7)
- Substituting Equation (10.7) into Equation (10.6), we have

$$X = -2^{k+1} + (x_{k-1} * 2^{k-1}) + \ldots + (x_0 * 2^0) \quad (10.8)$$

HOW BOOTH'S ALGORITHM WORKS: -VE MULTIPLIER

- Consider the multiplication of some multiplicand by (-6). In twos complement representation, using an 8-bit word, (-6) is represented as 11111010.
- $-6 = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1$
- $M * (11111010) = M * (-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1)$
- $M * (11111010) = M * (-2^3 + 2^1)$ which is equivalent to
  - $M * (11111010) = M * (-2^3 + 2^2 - 2^1)$ [right to left:0-1→ $- 2^1$, 1-0→ $+2^2$, 0-1→-2^3]

DIVISION

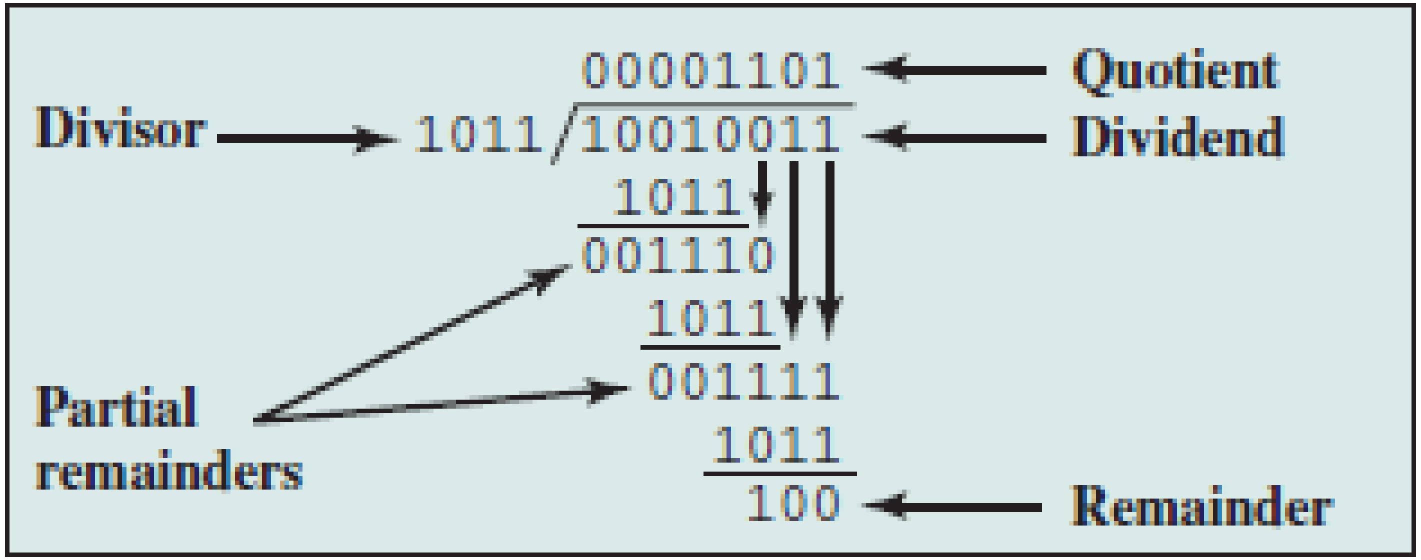

Figure 10.15 Example of Division of Unsigned Binary Integers

DIVISION

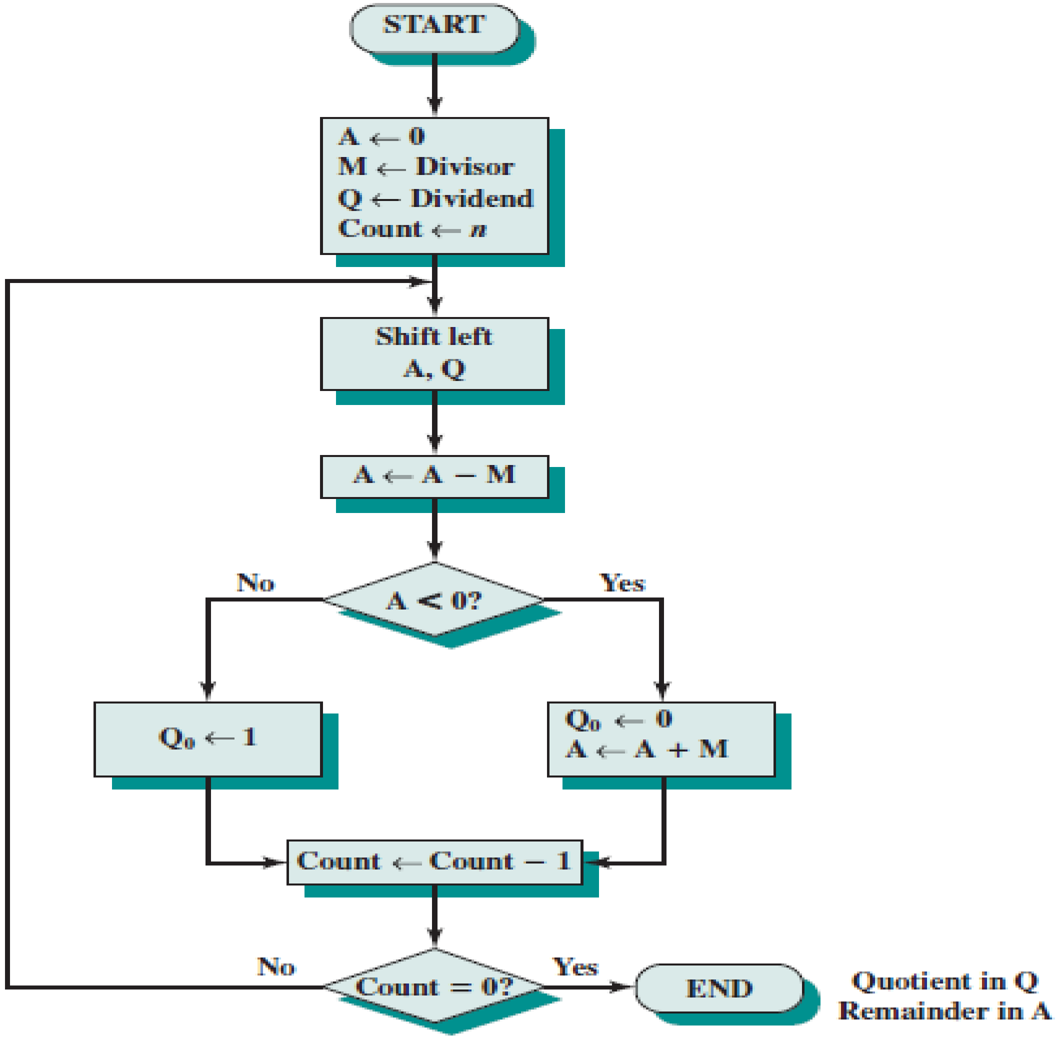

Figure 10.16 Flowchart for Unsigned Binary Division

DIVISION

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Figure 10.17 Example of Restoring Twos Complement Division (7/3)

DIVISION: ALGORITHM

- Assumption: divisor V and the dividend D are positive and that $|V| < |D|$.
- If $|V| = |D|$, then the quotient =1 and the remainder=0.
- If $|V| > |D|$, then Q=0 and R=D. The algorithm can be summarized as follows:
  1. Load the twos complement of the divisor into the M register; that is, the M register contains the negative of the divisor. Load the dividend into the A, Q registers. The dividend must be expressed as a 2n-bit positive number. Thus, for example, the 4-bit 0111 becomes 00000111.
  2. Shift A, Q left 1 bit position.
  3. Perform A=A-M. This operation subtracts the divisor from the contents of A.
  4.
     a. If the result is nonnegative (most significant bit of A=0), then set Q0=1
     b. If the result is negative (most significant bit of A=1), then set Q0=0, and restore the previous value of A.
  5. Repeat steps 2 through 4 as many times as there are bit positions in Q.
  6. The remainder is in A and the quotient is in Q.

DIVISION: EXAMPLE

- Divide 8 by 3; M=-3=11101; Q=01000

A	Q	
00000	01000	Initial values
00000 11101 11101 + 00011 00000	10000 10000 10000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 1st Cycle
00001 11101 11110 + 00011 00001	00000 00000 00000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 2nd Cycle
00010 11101 11111+ 00011 00010	00000 00000 00000	Shift Left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 3rd Cycle
00100 11101 00001	00000 00001	Shift Left Subtract Set Q0=1 4th cycle
00010 11101 11111 + 00011 00010	00010 00010 00010	Shift left Subtract Divisor(Add 2's complement of divisor) Set Q0=0 Restore 5th Cycle

DIVISION

- Consider the following examples of integer division with all possible combinations of signs of D and V:

D = 7	V = 3	→	Q = 2	R = 1
D = 7	V = -3	→	Q = -2	R = 1
D = -7	V = 3	→	Q = -2	R = -1
D = -7	V = -3	→	Q = 2	R = -1

- (-7)/(3) and (7)/(-3) produce different remainders.
- The magnitudes of Q and R are unaffected by the input signs
- The signs of Q and R are easily derivable from the signs of D and V.
  - sign(R) = sign(D)
  - sign(Q) = sign(D) * sign(V).
- One way to do twos complement division is to convert the operands into unsigned values and, at the end, to account for the signs by complementation where needed.
- This is the method of choice for the restoring division algorithm.

EXERCISE

1. Given x and y in twos complement notation i.e., x=0101 and y=1010,compute the product p=x*y with Booth's algorithm
2. Use the Booth algorithm to multiply 23 (multiplicand) by 29 (multiplier), where each number is represented using 6 bits
3. Divide 145 by 13 in binary twos complement notation, using 12-bit words. Use the restoring division algorithm

TOPICS COVERED FROM

- Textbook 2:
  - Chapter 10: 10.3