ICS 233
COMPUTER ARCHITECTURE

Integer Arithmetic
Division

Lecture 14

Integer Arithmetic

- **Division**

  - **Pencil and Paper Method - Long Division (Unsigned number division)**

  - **Two's complement Division Method**

Integer Arithmetic

❑ Long Division of Unsigned numbers

❖ **Method :**

- First, the bits of the dividend are examined from left to right, until the set of bits examined represents a number greater than or equal to the divisor; this is referred to as the divisor being able to divide the number.
- Until this event occurs, 0s are placed in the quotient from left to right.
- When the event occurs, a 1 is placed in the quotient and the divisor is subtracted from the partial dividend. The result is referred to as partial remainder
- From this point on, the division follows a cyclic pattern.
- At each cycle, additional bits from the dividend are appended to the partial remainder until the result is greater than or equal to the divisor
- The divisor is subtracted from this number to produce a new partial remainder.
- The process continues until all the bits of the dividend are exhausted.

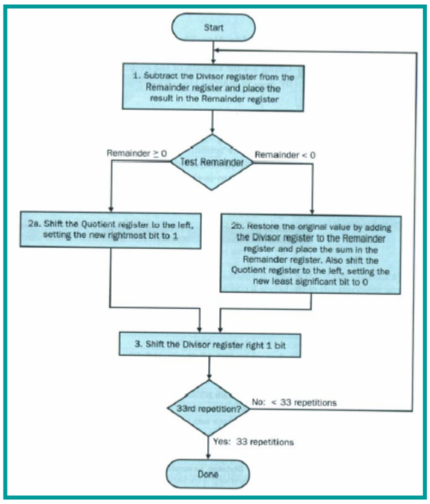
Integer Arithmetic

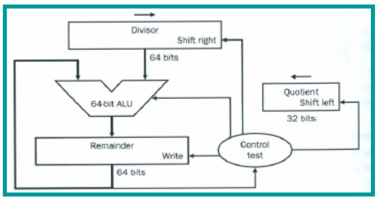
❑ Example : Compute 10010011 ÷ 1011

Dividend: 10010011, Divisor: 1011

```
              00001101   ← Quotient
         ______________________
  1011  |  10010011
            1011
          ----------
           001110
              1011
        -------------
              001111
                 1011
              ---------
                 0100    ← Remainder
```

First version of Division Algorithm



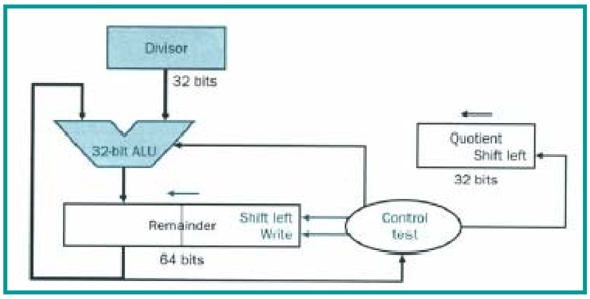


First version of Division Algorithm

Example : Divide 7 by 3 = 0111 / 0011 D= 0011, R=0111

Iteration Counter	Action	Quotient Q	Divisor D	Remainder (Dividend) R
5	Initialization	0000	0011 0000	0000 0111
4	R← R – D R < 0, Restore R Shift Left Q, Q0=0 Shift Right D	0000	0001 1000	1101 0111 0000 0111
3	R← R – D R < 0, Restore R Shift Left Q, Q0=0 Shift Right D	0000	0000 1100	1110 1111 0000 0111
2	R← R – D R < 0, Restore R Shift Left Q, Q0=0 Shift Right D	0000	0000 0110	1111 1011 0000 0111
1	R← R – D, R>0 Shift Left Q, Q0=1 Shift Right D	0001	0000 0011	0000 0001
0	R← R – D R < 0, Restore R Shift Left Q, Q0=0 Shift Right D	0010 **Quotient**	0000 0001	1111 1110 **0000 0001** **Remainder**

Second version of Division Algorithm



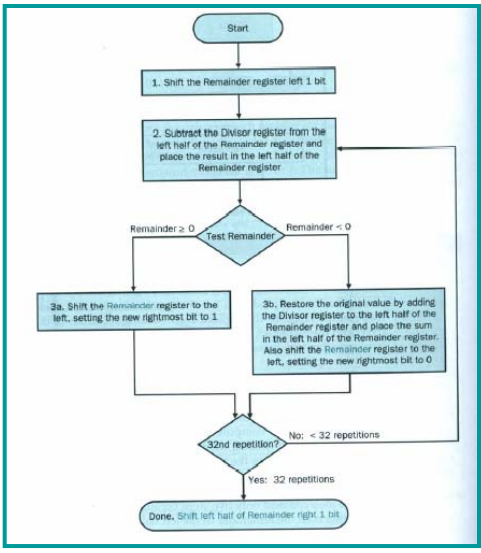
Second version of Division Algorithm

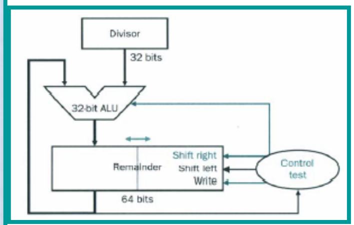
Example : Divide 7 by 3 = 0111 / 0011 D= 0011, R=0111

Iteration Counter	Action	Quotient Q	Divisor D	Remainder (Dividend) R
4	Initialization	0000	0011	0000 0111
3	Shift Left R R← Lefthalf(R) – D R < 0, Restore R Shift Left Q, Q0=0	0000	0011	0000 1110 1101 1110 0000 1110
2	Shift Left R R← Lefthalf(R) – D R < 0, Restore R Shift Left Q, Q0=0	0000	0011	0001 1100 1110 1100 0001 1100
1	Shift Left R R← Lefthalf(R) – D, R > 0 Shift Left Q, Q0=1	0001	0011	0011 1000 0000 1000
0	Shift Left R R← Lefthalf(R) – D R < 0, Restore R Shift Left Q, Q0=0	0010	0011	0001 0000 1110 0000 **0001** 0000

Quotient = 2, Remainder = 1

Third version of Division Algorithm





Third version of Division Algorithm

Example : Divide 7 by 3 = 0111 / 0011 D= 0011, R=0111

Iteration Counter	Action	Divisor D	Remainder (Dividend) R
4	Initialization	0011	0000 0111
3	Shift Left R R← Lefthalf(R) – D R < 0, Restore R, R0=0	0011	0000 1110 1101 1110 0000 1110
2	Shift Left R R← Lefthalf(R) – D R < 0, Restore R, R0=0	0011	0001 1100 1110 1100 0001 1100
1	Shift Left R R← Lefthalf(R) – D, R > 0, R0=1	0011	0011 1000 0000 1001
0	Shift Left R R← Lefthalf(R) – D R < 0, Restore R, R0=0	0011	0001 0010 1110 0010 0001 0010 R Q

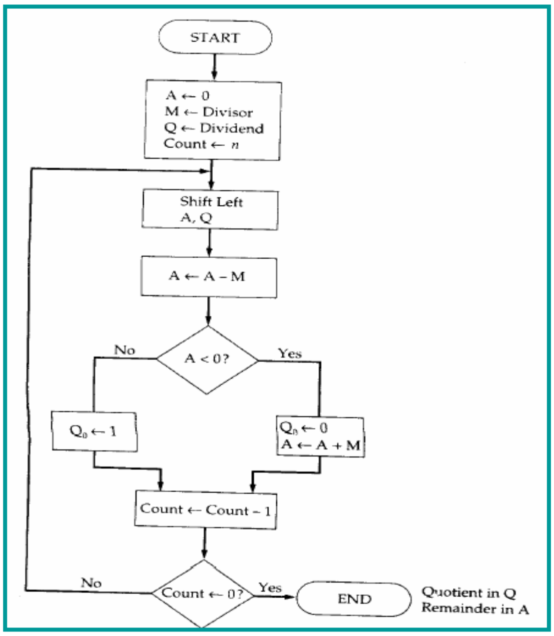
Quotient = 2, Remainder = 1

Integer Arithmetic

❑ Algorithm for Long Division (Restoring Division Technique)

- **The divisor is placed in the M register**
- **The dividend is placed in the Q register**

- **At each step, the A and Q registers together are shifted to the left 1 bit**
- **M is subtracted from A to determine whether A divides the partial remainder.**
- **If it does, then Q0 gets a 1 bit, otherwise Q0 gets a 0 bit and M must be added back to A to restore the previous value.**
- **The count is then decremented and the process continues for n steps.**
- **At the end, the quotient is in the Q register and the remainder is in the A register.**



Integer Arithmetic

- **Example : Compute 147/11 (i.e., 10010011 ÷ 1011)**

A = 00000000 Q = 10010011 M = 00001011

A	Q	Counter	Operation
00000000	10010011	8	Initialization
00000001	00100110		Shift left A, Q
11110110	00100110		A ← A – M
00000001	00100110	7	**Restore A, Q0=0**
00000010	01001100		Shift left A, Q
11110111	01001100		A ← A – M
00000010	01001100	6	**Restore A, Q0=0**
00000100	10011000		Shift left A, Q
11111001	10011000		A ← A – M
00000100	10011000	5	**Restore A, Q0=0**
00001001	00110000		Shift left A, Q
11111110	00110000		A ← A – M
00001001	00110000	4	**Restore A, Q0=0**

Integer Arithmetic

- **Example : Compute 147/11 (i.e., 10010011 ÷ 1011)**

A	Q	Counter	Operation
00010010	01100000		Shift left A, Q
00000111	01100000		A ← A – M
00000111	01100001	3	**Q0=1**
00001110	11000010		Shift left A, Q
00000011	11000010		A ← A – M
00000011	11000011	2	**Q0=1**
00000111	10000110		Shift left A, Q
11111100	10000110		A ← A – M
00000111	10000110	1	**Restore A, Q0=0**
00001111	00001100		Shift left A, Q
00000100	00001100		A ← A – M
00000100	**00001101**	0	**Q0=1**

Remainder=4 Quotient=13

Integer Arithmetic

❑ Division of Two's complement numbers

❖ Algorithm

➢ **Step 1 :**
- Load the divisor into the M register and the dividend into the A, Q registers.
- The dividend must be expressed as a 2n-bit two's complement number.
- For example, the 4-bit 0111 becomes 00000111 and 1001 becomes 11111001.

➢ **Step 2 :**
- Shift A, Q left 1 bit position

➢ **Step 3 :**
- If M and A have the same signs, perform A ← A – M; otherwise, A ← A + M

Integer Arithmetic

❑ Division of Two's complement numbers

❑ Algorithm (Continued)

➢ **Step 4**
- The above operation is successful if the sign of A is the same before and after the operation
- If the operation is successful or (A=0 AND Q=0), then set Q0 ← 1
- If the operation is unsuccessful and (A ≠ 0 OR Q ≠ 0), then set Q0 ← 0 and restore the previous value of A

➢ **Step 5**
- Repeat steps 2 through 4 as many times as there are bit positions in Q

➢ **Step 6**
- **The remainder is in A.**
- **If the signs of the divisor and dividend were the same, then the quotient is in Q, otherwise, the correct quotient is the two's complement of Q.**

Integer Arithmetic

- **Example : Compute 7 ÷ 3 i.e., 0111 ÷ 0011**

A = 0000 Q = 0111 M = 0011

A	Q	Counter	Operation
0000	0111	4	Initialization
0000	1110		Shift left
1101	1110		A ← A – M
0000	1110	3	**Restore A, Q0=0**
0001	1100		Shift left
1110	1100		A ← A – M
0001	1100	2	**Restore A, Q0=0**
0011	1000		Shift left
0000	1000		A ← A – M
0000	1001	1	**Q0 = 1**
0001	0010		Shift left
1110	0010		A ← A – M
0001	**0010**	**0**	**Restore A, Q0 = 0**
Remainder=1	**Quotient=2**		

Integer Arithmetic

- **Example : Compute 7 ÷ (-3) i.e., 0111 ÷ 1101**

A = 0000 Q = 0111 M = 1101

A	Q	Counter	Operation
0000	0111	**4**	Initialization
0000	1110		Shift left
1101	1110		A ← A + M
0000	1110	3	**Restore A, Q0=0**
0001	1100		Shift left
1110	1100		A ← A + M
0001	1100	2	**Restore A, Q0=0**
0011	1000		Shift left
0000	1000		A ← A + M
0000	1001	1	**Q0 = 1**
0001	0010		Shift left
1110	0010		A ← A + M
0001	**0010**	**0**	**Restore A, Q0 = 0**
	1110		**Two's Complement of Q**
Remainder= 1	**Quotient = - 2**		

Integer Arithmetic

- **Example : Compute (- 7) ÷ 3 i.e., 1001 ÷ 0011**

A = 1111 Q = 1001 M = 0011

A	Q	Counter	Operation
1111	1001	4	Initialization
1111	0010		Shift left
0010	0010		A ← A + M
1111	0010	3	**Restore A, Q0=0**
1110	0100		Shift left
0001	0100		A ← A + M
1110	0100	2	**Restore A, Q0=0**
1100	1000		Shift left
1111	1000		A ← A + M
1111	1001	1	**Q0 = 1**
1111	0010		Shift left
0010	0010		A ← A + M
1111	0010	0	**Restore A, Q0 = 0**
	1110		**Two's Complement of Q**

Remainder= - 1 **Quotient = - 2**

Integer Arithmetic

- **Example : Compute (-7) ÷ (-3) i.e., 1001 ÷ 1101**

A = 1111 Q = 1001 M = 1101

A	Q	Counter	Operation
1111	1001	**4**	Initialization
1111	0010		Shift left
0010	0010		A ← A - M
1111	0010	3	**Restore A, Q0=0**
1110	0100		Shift left
0001	0100		A ← A - M
1110	0100	2	**Restore A, Q0=0**
1100	1000		Shift left
1111	1000		A ← A - M
1111	1001	1	**Q0 = 1**
1111	0010		Shift left
0010	0010		A ← A - M
1111	0010	0	**Restore A, Q0 = 0**

Remainder= - 1 **Quotient = 2**

Divider - HARDWARE IMPLEMENTATION