Chapter 2
Data Representation and Arithmetic Algorithms

CE – SE – Digital Logic & Computer Organization and Architecture

Prof. Kalyani N Pampattiwar
Asst. Prof.
Dept. of Computer Engineering,
SIES Graduate School of Technology

Binary Arithmetic

Binary Addition

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary subtraction

A	B	DIFF.	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- **Binary Multiplication**

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1

- **Binary Division**

0 / 1 = 0

1 / 1 = 1

1) Represent (-17) 10 and (+17)10 in :

a) Sign magnitude

b) One's complement

c) Two's complement

Use 8 bit representation.

Binary Addition

- When we add two signed numbers then the cases are:

1) Both +ve ---- addition produces binary answer

2) +ve and smaller –ve

- Take 2's complement of smaller –ve number
- If carry is generated after 8^{th} bit , ignore it

3) Both –ve numbers

- Add 2's complement of both the numbers, ignore 8^{th} final carry
- We get answer in 2's complement
- Therefore, take 2's complement -1= 1's complement , Binary number

4) +ve and large –ve

+ve number

\+ 2's compleent of –ve

Ans in 2's complement

- Take 2's complement -1= 1's complement , Binary number

Examples for Binary Addition

1) Add 14 with 9
2) 14 + (-9)
3) -14+9
4) -14-9

Binary Subtraction

1) **Both +ve**

Binary No.
+ 2's complement of 2^{nd}

ANS

If carry is generated then ignore it

2) +ve and small –ve

Binary No.
+ Binary No.

Binary No.

3) +ve and large –ve

Binary No.
\+ Binary No.

Binary No

4) Both –ve Numbers

2's complement of Binary number
.+ Binary number

ANS

If carry is generated then ignore it

Examples for Binary Subtraction

1) 14-9
2) 14-(-9)
3) 9-(-14)
4) -9-(-14)

Binary Multiplication

```
1)    011          2)   1110
    X 110              X 110
  ___________                  _________
       000               0000
  +    0110            +    11100
  +   01100        +  111000
  ___________      ___________
      10010         1010100
```

Binary Division

1)

```
          100100              Quotient
110  |
     |     11011011           Dividene
     |     -110
           ______
Divisor    000110
            -     110
           ______
                 00011        Remainder
```

2) 1110101 / 1001

Binary subtraction

1's complement

1) Take 1's complement B
2) Add A
   \+ 1's complement of B

   Sum

3) C=1 Add to LSB
   C=0 Find 1's complement of sum

2's complement

1) Take 2's complement of B
2) Add A
   \+ 2's complement of B

   Sum

3) C=1 Ignore it
   C=0 Find 2's complement of sum

Binary subtraction using 1's and 2's complement

1) $(111110)_2 - (011101)_2$

2) $(101101)_2 - (01111)_2$

Octal subtraction using 7's and 8's complement

7's complement

1) Take 7's complement
2) Add A

   \+ 7's complement of B

   Sum

3) C=1 Add to LSB

   C=0 Find 7's complement of sum

8's complement

1) Take 8's complement of B
2) Add A

   \+ 8's complement of B

   Sum

3) C=1 Ignore it

   C=0 Find 8's complement of sum

Octal subtraction using 7’s and 8’s complement

1) $(727)_8 - (143)_8$

2) $(727.24)_8 - (143.4)_8$

Direct method

1) $(727.24)_8 + (143.4)_8 = (1072.64)_8$
2) $(727.24)_8 - (143.4)_8 = (563.64)_8$
3) $(135.7)_8 - (67.7)_8 = (46.0)_8$
4) $(241)_8 - (176)_8 = (43)_8$

Hexadecimal Addition

- Sum of two hexadecimal digits is the same as their equivalent decimal sum, provided the decimal equivalent is less than 16.
- If decimal sum is 16 or grater than 16 then subtract 16 to obtain the hexadecimal digit.
- A carry of 1 is produced when the decimal sum is corrected

Examples

1) $(3)_{16} + (9)_{16} = (C)_{16}$
2) $(3F8)_{16} + (5B3)_{16} = (9AB)_{16}$
3) $(4FB)_{16} + (75D)_{16} + (A12)_{16} + (C39)_{16} = (22A3)_{16}$

Hexadecimal subtraction using 15's and 16's complement

15's complement

1) Take 15's complement
2) Add A
   + 15's complement of B

Sum

3) C=1 Add to LSB
   C=0 Find 15's complement of sum

16's complement

1) Take 16's complement of B
2) Add A
   + 16's complement of B

Sum

3) C=1 Ignore it
   C=0 Find 16's complement of sum

Examples

1) (3B7) 16-(854)16=(49D)16
2) (B02)16-(98F)16=(173)16
3) (CB2)16-(972)16=(340)16

Direct Method

1) $(B02)_{16}-(98F)_{16}=(173)_{16}$
2) $(C14)_{16}-(69B)_{16}=(579)_{16}$
3) $(B92)_{16}-(98F)_{16}=(203)_{16}$
4) $(A2C.6A)_{16}-(8BB.7C)_{16}=(170.EE)_{16}$

For Any Radix

1) $(24)_6-(15)_6=(05)_6$
2) $(321.2)_4-(33.3)_4=(221.3)_4$
3) $(23.4)_{10}-(19.8)_{10}=(3.6)_{10}$

BCD Arithmetic

1) Add two BCD numbers using binary addition.
2) If four bit sum is equal to or less than 9 no correction is needed
3) If four bit sum is greater than 9, or if carry is generated from four bit sum, sum is invalid.
4) To correct it, add $(0110)_2$ to sum. If carry results from this addition, add it to next higher order BCD digits

SIES Graduate School of Technology
RISE WITH EDUCATION

Examples

1) 8 + 9 = 17

2) 24 + 18= 42

3) 48 + 58= 106

4) 175 + 326 = 501

1) 8 + 9=17
1000 BCD of 8
+ 1001 BCD of 9

____

10001 (sum>9)
0110 (add 6)

____

0001 0111
1   7

2) 2) 24 + 18
0010 0100
+ 0001 1000

____

0011 1100 12>9
+ 0110 Add 6
_______1____________
0011 10010
1 1 1   propogation

____

0100 0010
4 2

BCD subtraction using 9's complement

1) Find 9's complement of negative number
2) Add two numbers using BCD addition
3) If carry = 1 Add carry to result

   carry = 0 Find 9's complement of result

Examples

1) $79 - 26 = 53$
2) $89 - 54 = 35$

BCD Subtraction using 10's Complement

1) Find 10's complement of negative number (9's + 1)
2) Add two numbers using BCD addition
3) If carry = 1   Ignore it

   carry = 0   Find 10's complement of sum or result

Examples

1) 28-13=15
2) 79 – 26 = 53

Virtual Lab setup for Practicals

http://vlabs.iitkgp.ernet.in/coa/#

Hexadecimal Product

1) (AFC4) 16 * (B9C) 16 = (7F88770)16
2) (FC2)16 * (DE)16

Multiplication

- **Computerized multiplication can be made more efficient using following ways:**

1) We can perform a running addition on the partial products rather than waiting until the end. This eliminates the need for storage of all the partial products, fewer registers are needed.
2) We can save some time on the generation of partial products. For each 1 on the multiplier, an add and a shift operation are required; but for each 0, only shift is required.

Unsigned binary multiplication

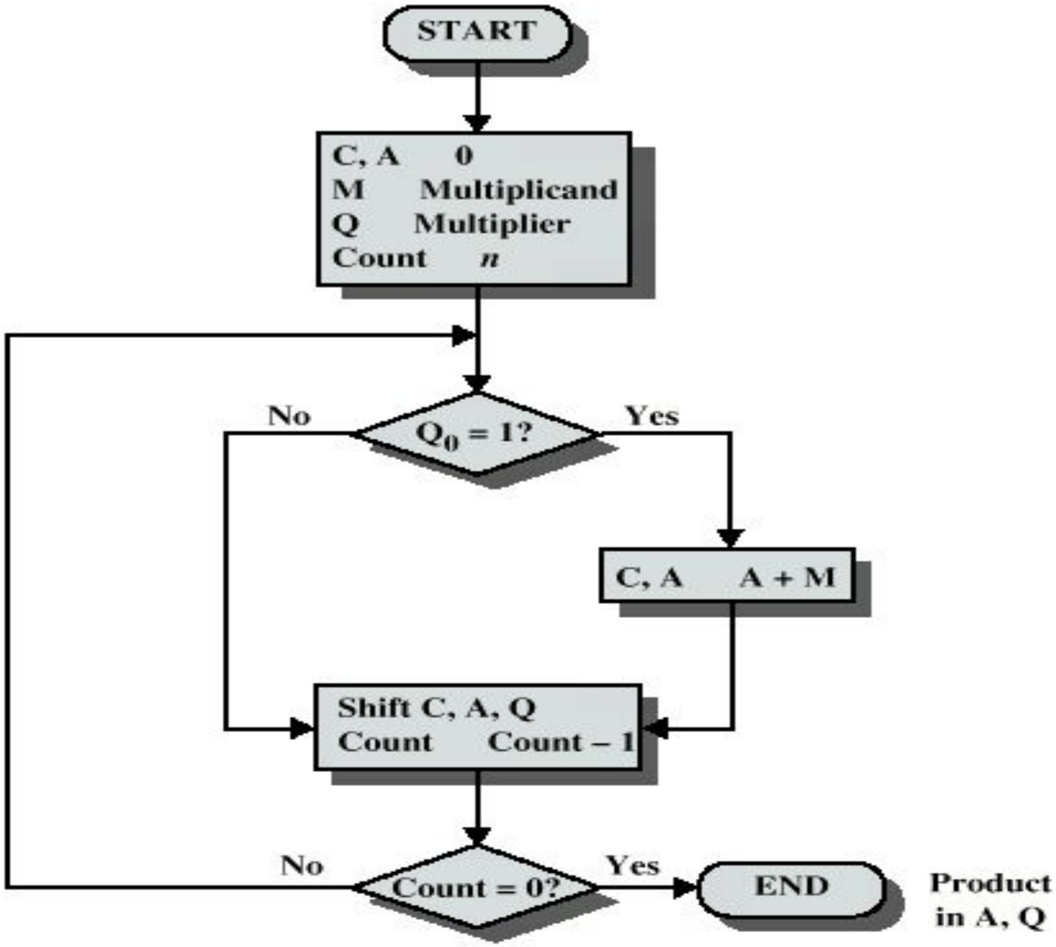


Figure 8.9 Flowchart for Unsigned Binary Multiplication

Unsigned binary multiplication

1) 11 x 13=143

C	A	Q (Multiplier)	M (Multiplicand)		
0	0000	1101	1011	Initial values	
0	1011	1101	1011	Add M to A	First cycle
0	0101	1110	1011	Shift C,A, Q right	
0	0010	1111	1011	Shift	Second cycle
0	1101	1111	1011	Add A+M	Third cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth cycle
0	1000	1111	1011	Shift	

ANS

Graduate School of Technology
SIES
RISE WITH EDUCATION

- For –ve numbers we can perform addition and subtraction using 2’s complement method but this scheme will not work for multiplication
- For e.g. when we multiply -5 * -3 we get –113 (10001111)

2’s complement of -5= 1011

2’s complement of -3= 1101

```
        1011
X       1101
________________
        1011
+     101100
+    1011000
________________
    10001111
```

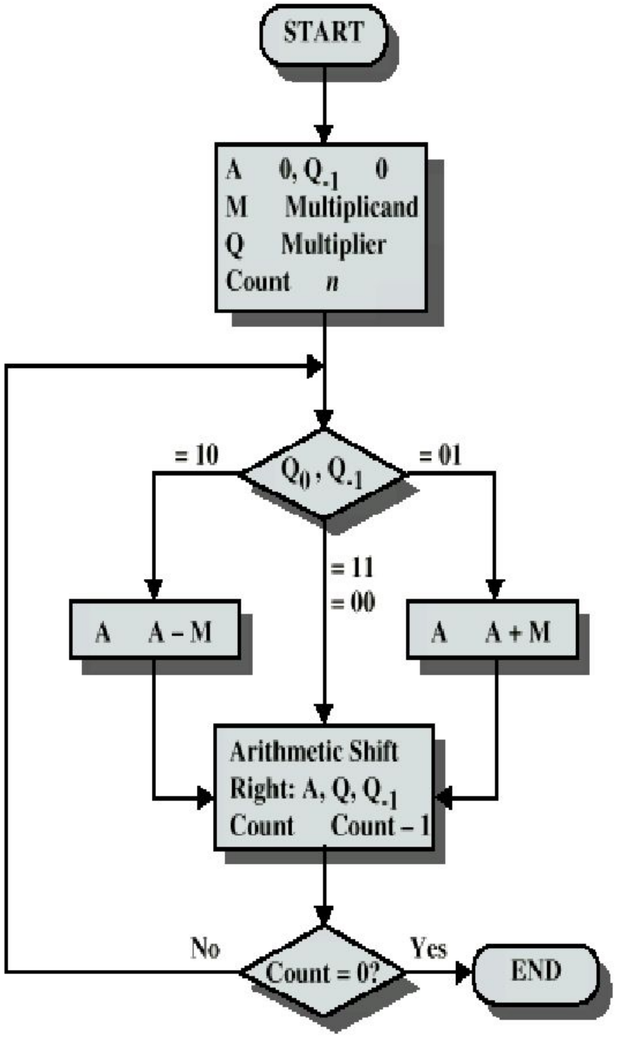
Take 2’s complement of answer

10001111= > 11110001= -113

The solution to the problem are:

1) Convert both multiplier and multiplicand to the +ve numbers and then perform the multiplication and then take 2’s complement of the result iff the sign of the two original numbers differed.
2) Implementers have preferred to use techniques that do not require this final transformation step.
3) One of the most common of these is Booth’s algorithm. This algorithm has the benefit of speeding up the multiplication process. **Used for signed binary multiplication.**

Booth's algorithm for 2's Complement Multiplication



1) Multiply 7 X 3 using Booth's algorithm

A	Q (MULTIPLIER)	Q-1 (1 BIT REG.)	M (MULTIPLICAND)		
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**

ANS

2) -7 X 3

3) -13 X -20

4) -9 X 7

Note: Numbers from **0 to 7 need 4 bits** for multiplication

Numbers from **8 to 15 need 5 bits** for multiplication

Numbers from **16 to 31 need 6 bits** for multiplication & so on

Restoring Division Algorithm

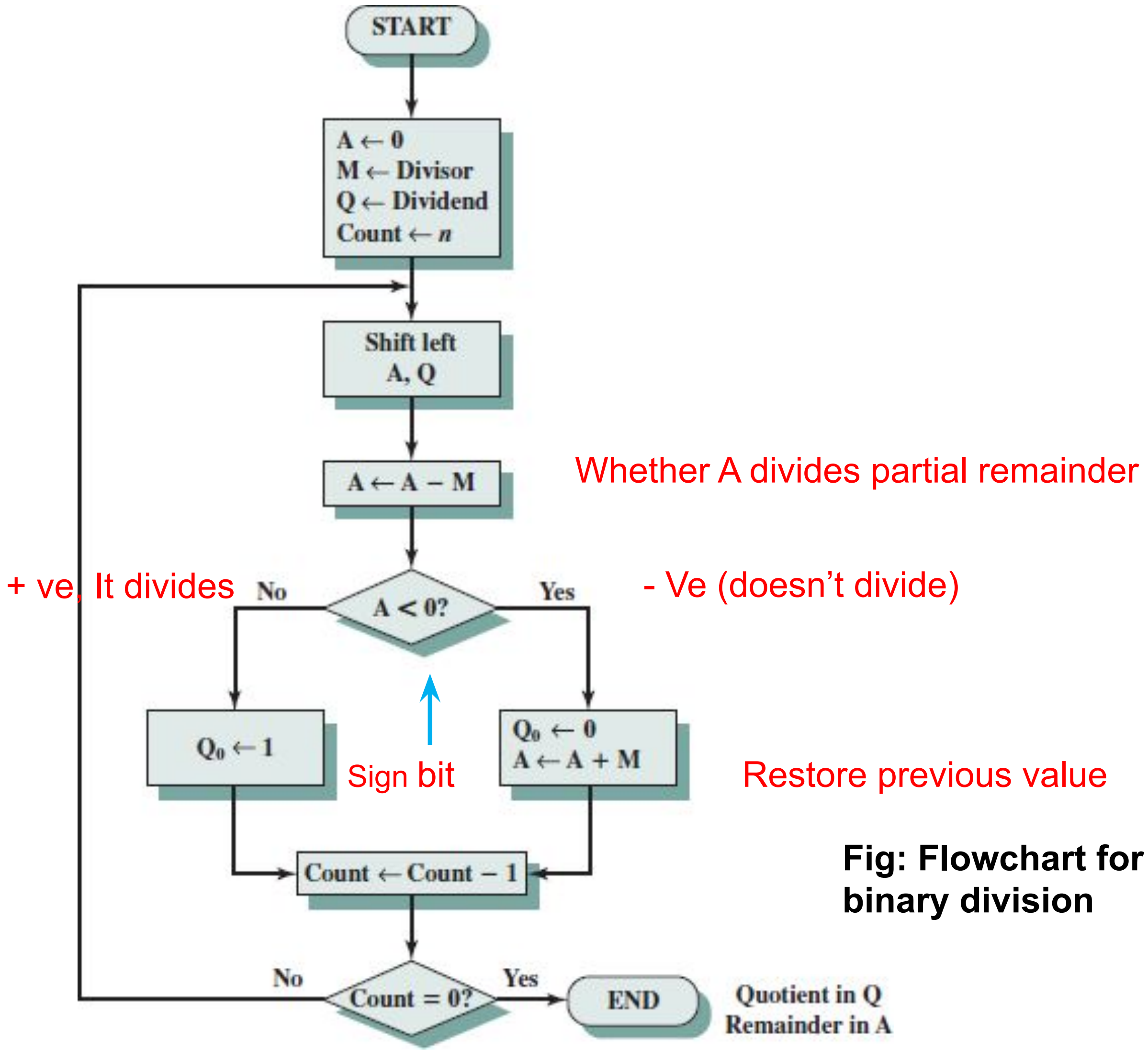


Fig: Flowchart for unsigned binary division

1) 7 / 3

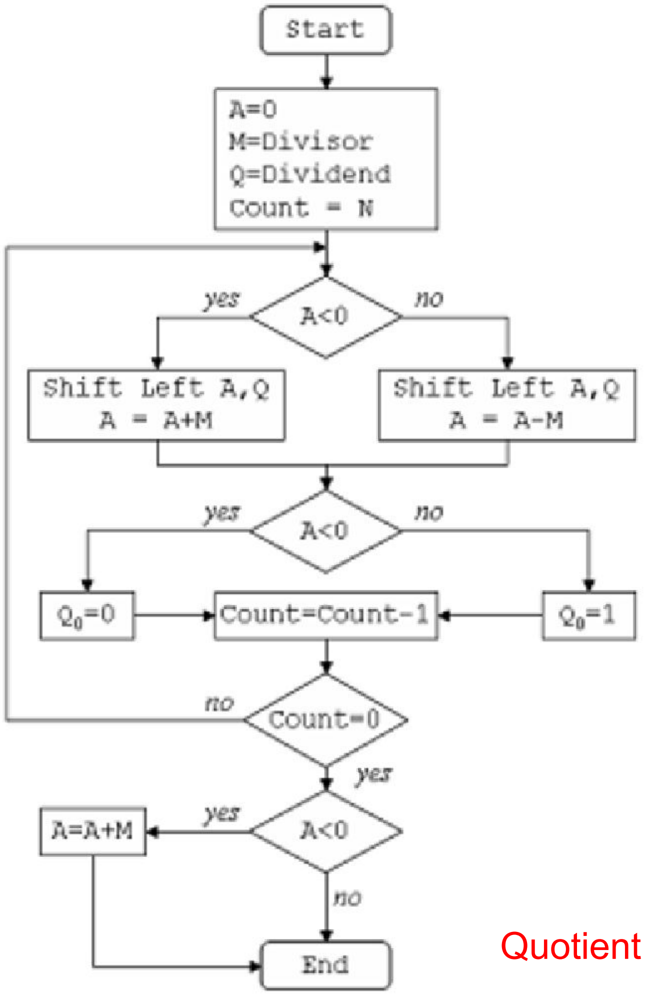
A	Q(Dividend)	
0000	0111	Initial value
0000	1110	Shift
1101		Use 2’s complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, Set Q0=0
0001	1100	shift
1101		
1110		Subtract
0001	1100	Restore, Set Q0=0
0011	1000	Shift
1101		
0000	1001	Subtract , set Q0=1
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, Set Q0=0
Remainder	**Quotient**	

2) Dividend=17 and Divisor=03

3) Dividend=1010 and Divisor=0011

4) Dividend= 1001 and Divisor=0101

Non Restoring Division



Quotient in Q and Remainder in A

1) Dividend= 1010 , Divisor=0011

A	Q	
0000	1010	Initial values
0001	0100	Shift left A,Q
1101		A!=0, Subtract M
1110	0100	
1110	0100	Set Q0=0
1100	1000	Shift
0011		
1111	1000	
1111	1000	Set Q0=0
1111	0000	shift
0011		Add M
10010	0000	
0010	0001	Set Q0=1
0100	0010	Shift
1101		
10001	0010	
0001	0011	set Q0=1

2) Dividend=1011 Divisor= 0101

3) Dividend=19. Divisor=4

Difference Between Restoring and Non Restoring Division Algorithm

Restoring Division	Non Restoring Division
1. Needs restoring of register A if the result of subtraction is -ve	1. Does not need restoring
2. In each cycle content of register A is first shifted left and then divisor is subtracted from it	2. In each cycle content of register A is first shifted left and then divisor is added or subtracted with the content of register A depending on the sign of A
3. Does not need restoring of remainder	3. Needs restoring of remainder if remainder is -Ve
4. Slower algorithm	4. Faster algorithm

Restoring Division Algorithm for Signed Integers

1. Dividend (2n) is placed in AQ register. The divisor is placed in M register.
2. Shift A,Q left 1 bit position. Vacant Q positions are used to store quotient bits.
3. If M & A have same signs, perform A<- A-M. Otherwise A<- A + M
4. The preceding operation is successful if the sign of A is the same before and after the operation.
   a) If the operation is successful or A=0, then set Q0<-1
   b) If the operation is unsuccessful & A!=0, then set Q0 <- 0 & restore the previous value of A.
5. Repeat step 2 through 4 as many times as there are bit positions in Q
6. The remainder is in A. If the signs of the divisor and dividend are the same, then the quotient is in Q otherwise the correct quotient is the 2’s complement of Q.

Example 1.

1) -10/4 10 --□ 00001010 □2's complement= > 11110101+1 = 11110110

```
     Dividend(2n)  Divisor
     A    Q        M
1111       0110    0100 Initial values
---------------------------------------------------------------
  _1110    1100    0100 Shift A, Q left
   0100                 M & A diff. signs , so add A & M
-------------
 1 0010    1100    0100
1110       1100    0100 if operation is unsuccessful (4b) restore
---------------------------------------------------------------------------------
  1101     1000    0100 Shift A, Q left
   0100                 M & A diff. signs , so add A & M
----------
 1 0001    1000    0100 if operation is unsuccessful (4b) restore,
  1101     1000    0100  Q0 <-0
------------------------------------------------------------------------------------------
  1011     0000    0100  Shift A, Q left
  0100                  M & A diff. signs , so add A & M
----------
  1111     0000    0100 same sign
  1111     0001    0100 4a, Q0 <- 1
------------------------------------------------------------------------------------------
```

1110 0010 0100 Shift A,M
0100 Add
----
1 0010 Diff. sign , restore (4b)

1110 0010 0100
Remainder 2’s complement
1110 1110
-2 -2

Dividend= Quotient X Divisor + Remainder
-10= -2 X 4-2
-10=-10

2) 7/ -3