ASSIGNMENT -3

ECE-585: Computer Organization and Design.

1. What is 6FD4 - 273B when these values represent signed 16-bit hexadecimal numbers stored in sign-magnitude format? The result should be written in hexadecimal. You MUST show your work.

answer: Given 6FD4 - 273B

each hexadecimal number is represented by 4-bits.

$(6FD4)_{16} = (0110\ 1111\ 1101\ 0100)_2$.

$(273B)_{16} = (0010\ 0111\ 0011\ 1011)_2$

6FD4 → 0110 1111 1101 0100

- 273B → 0010 0111 0011 1011

0100 1000 1001 1001

$(6FD4 - 273B)_{16} = (4899)_{16}$

2. Assume 174 and 85 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 174 + 85. Is there overflow, underflow or neither? You MUST show your work.

answer:

$(174)_{10}$ is in decimal sign magnitude format.

Converting in binary,

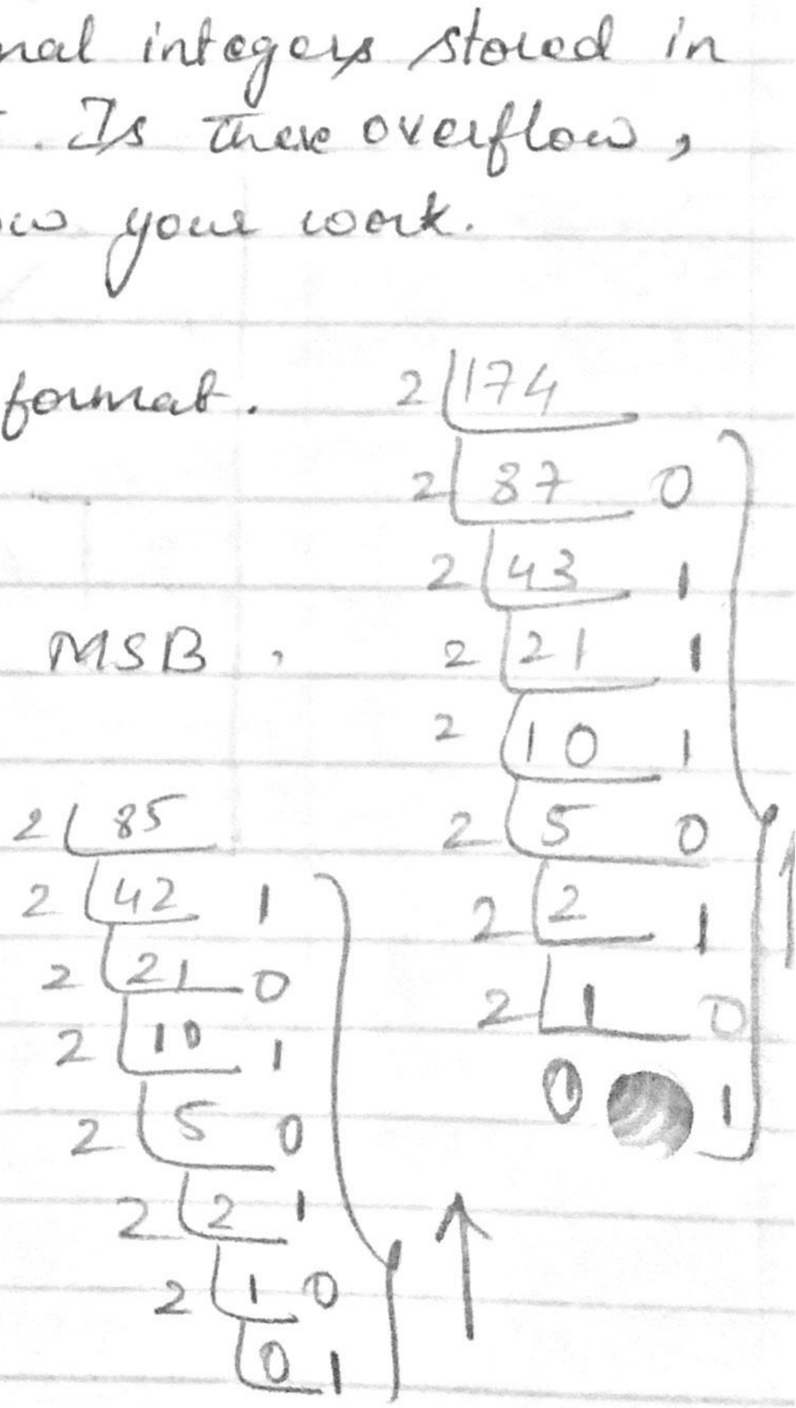
$(174)_{10} = (10101110)_2$.

In sign magnitude representation, the MSB describes the sign of number.

i.e., if MSB = 1, negative no.

MSB = 0, positive no.

& $(85)_{10} = (10101010)_2$

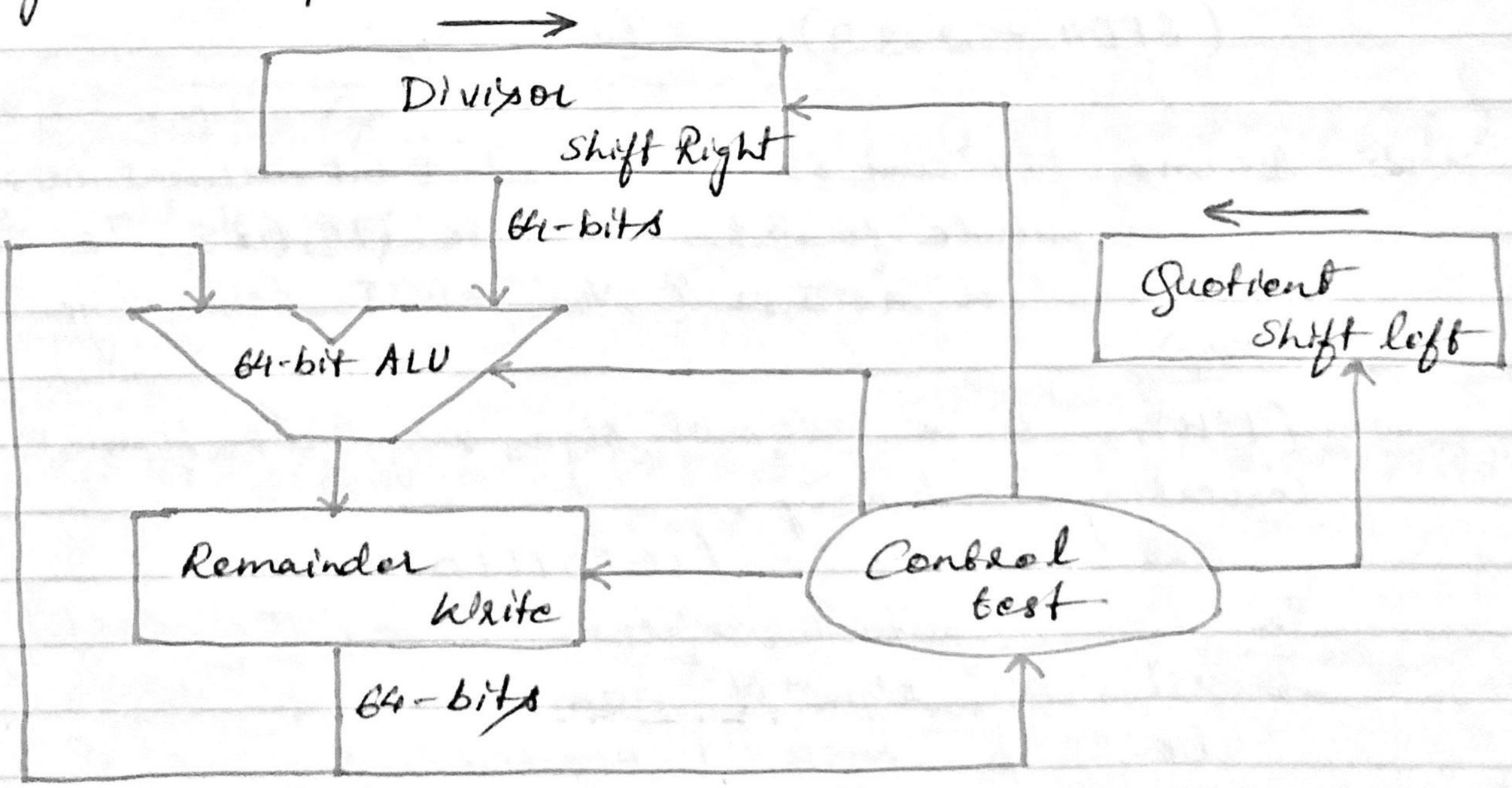


```
                  (1)(1)(1)(1)(1)(0)
  174             1010  1110
+ 85              0101  0101
                 ------------------
                (1)0000  0011
```

$\therefore (10000\ 0011)_2 = (259)_{10}$.

2. Using table similar to the below table-1. Calculate 60 divide by 22 by using the hardware described in fig-1 You should show the contents of each register on each step. Assume both inputs are unsigned 6-bit integers. NOT 32-bit Integers, and modify the division hardware shown in fig-1.

<u>answer:</u> Consider the hardware shown below, the following algorithm represents the hardware described.



D1

→ Division Algorithm:

- Initialize the 32-bit Quotient Register with 0
- Place the divisor in the left half of the 64-bit Divisor Register.
- Set the Remainder Register with the dividend.
- For each Iteration in the algorithm
  - Remainder = Remainder − Divisor.
  - If Remainder < 0 (sign bit is 1)
    - Remainder = Remainder + Divisor.
    - left shift the Quotient Register by 1-bit position
    - Set LSB to 0.
  - If Remainder >= 0 (sign bit is 0)
    - left shift the Quotient Register by 1-bit position.
    - Set LSB to 1.
  - Right Shift the Divisor Register by One bit.

→ Division can be calculated using the above algorithms as follows:

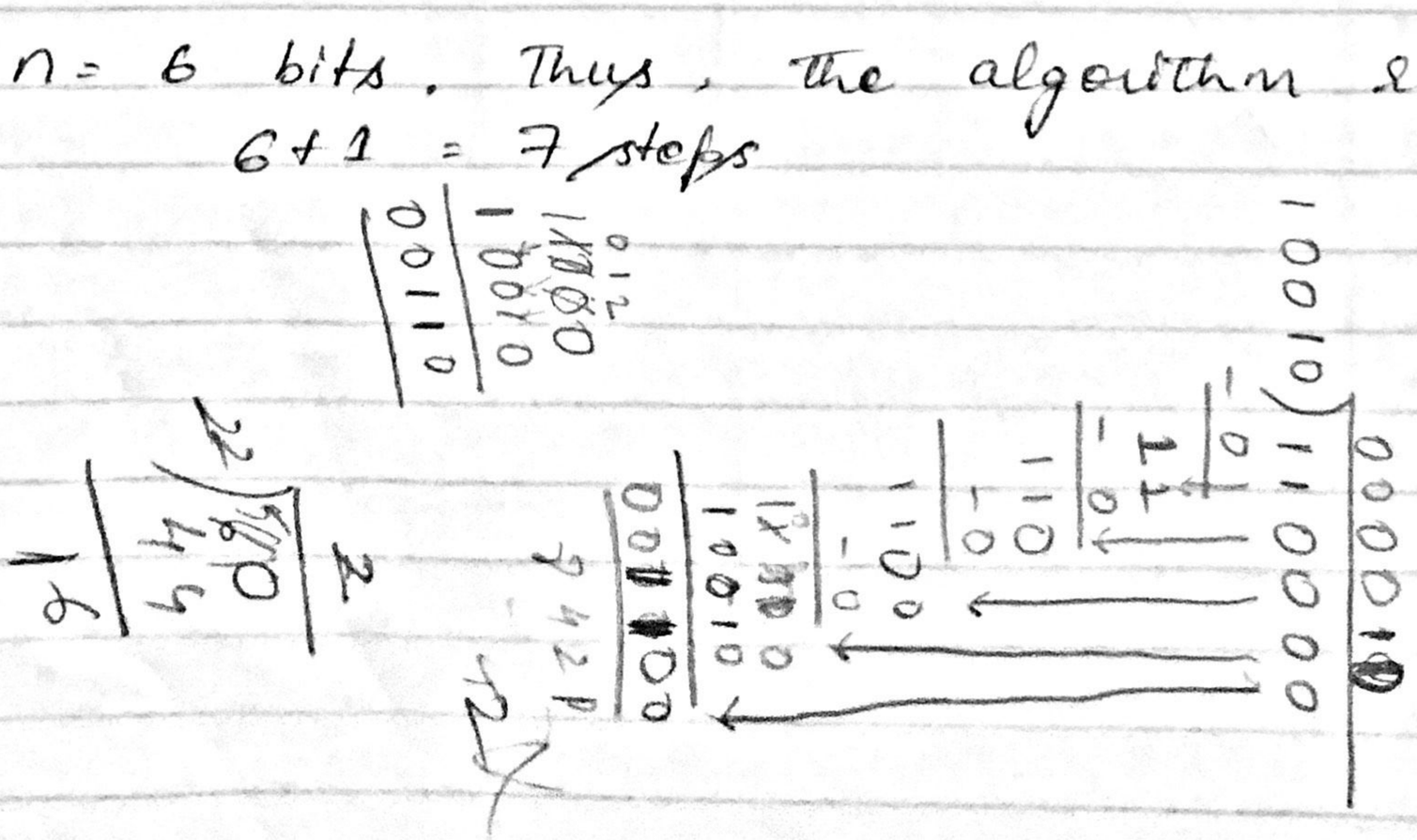
Consider,

Dividend = $(60)_8$ = $(110000)_2$ = $(48)_{10}$

Divisor = $(22)_8$ = $(10010)_2$ = $(18)_{10}$

2|22 2|60
2|11 0 2|30 0
2|5 1 2|15 0
2|2 1 2|7 1
2|1 0 2|3 1
0 1 2|1 1
0 1

Here, n = 6 bits. Thus, the algorithm runs 6+1 = 7 steps



[IGNORE]

Steps	Actions	Quotient (Q)	Divisor (Div)	Remainder (Rem)
0.	Initial Values	000000	010010 000000	000000 110000
1.	✓ Rem = Rem - Div	000000	010010 000000	10110 110000
	✓ if Rem < 0 → Rem = Rem + Div	000000	010010 000000	000000 110000
	Shift left Q, Q0 = 0			
	✓ Shift Div Right	000000	001001 000000	000000 110000
2.	✓ Rem = Rem - Div	000000	001001 000000	10111 110000
	✓ if Rem < 0 → Rem = Rem + Div	000000	001001 000000	000000 110000
	Shift left Q, Q0 = 0			
	✓ Shift Div Right	000000	000100 100000	000000 110000
3.	✓ Rem = Rem - Div	000000	000100 100000	11100 010000
	✓ if Rem < 0 → Rem = Rem + Div	000000	000100 100000	000000 110000
	Shift left Q, Q0 = 0			
	✓ Shift Div Right	000000	000010 010000	000000 110000
4	✓ Rem = Rem - Div	000000	000010 010000	11000 011000
	✓ if Rem < 0 → Rem = Rem + Div	000000	000010 010000	000000 011000
	Shift left Q, Q0 = 0			
	✓ Shift Div Right	000000	000001 001000	000000 011000
5.	✓ Rem = Rem - Div	000000	000001 001000	
	✓ if Rem < 0 → Rem = Rt Div			
6.				
7.				

CONTINUED IN PAGE 7 & 8

4. Write down the binary representation of the decimal number 56.93 assuming the IEEE-754 double precision format.

answer: $(56)_{10} = (111000)_2$.

$(0.93)_{10} = (111\ 0111100000\ 101000\ \ldots)_2$

$\therefore (56.93)_{10} = (111000.1110111100000\ 101000\ \ldots)_2$

Representing in scientific Notation,

$1.11000\ 1110\ 1111\ 0000\ 10100\ \ldots \times 2^5$

We have,

$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - 1023)}$

- Sign = 0 (positive number)
- Exponent = 1023 + 5 = 1028

  $(1028)_{10} = (10000000100)_2$.
- Mantissa = 11 0001 1101 1110 0001

  0100 0111 1010 1110 0000 1010 0

  0111 1010 1111

$(0.93)_{10} = (\quad)_2$

0.93×2	1.86	1	2 | 56
0.86×2	1.72	1	2 | 28 0
0.72×2	1.44	1	2 | 14 0
0.44×2	0.88	0	2 | 7 0
0.88×2	1.76	1	2 | 3 1
0.76×2	1.52	1	2 | 1 1
0.52×2	1.04	1	0 1
0.04×2	0.08	0	
0.08×2	0.16	0	
0.16×2	0.32	0	
0.32×2	0.64	0	
0.64×2	1.28	1	
0.28×2	0.56	0	
0.56×2	1.12	1	
0.12×2	0.24	0	
0.24×2	0.48	0	
0.48×2	0.96	0	
0.96×2	1.92	1	
0.92×2	1.84	1	
⋮	⋮	⋮	

2 | 1028	0	
2 | 514	0	
2 | 257	0	
2 | 128	1	
2 | 64	0	
2 | 32	0	
2 | 16	0	
2 | 8	0	
2 | 4	0	
2 | 2	0	
2 | 1	0	

The binary representation assuming IEEE-754 double precision format is,

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	1	0	1	0

1 bit — 11-bits — 20 bits

0 01111 0101 1100 0010 1000 1111 0101 11

32-bits

5. IEEE 754-2008 standard contains a half precision that is only 16-bits wide. Leftmost bit which is the MSB, is still the sign bit, the exponent is 5-bit wide & has a bias of 15 and the fraction is 10 bits long. A hidden 1 is assumed. Write down the bit pattern to represent -1.585×10^{-1} assuming a version of this format, which uses an excess-16 format (exponent bias = 15) to store the exponent. Comment on how the range & accuracy of this 16-bit floating point format compares to the single precision IEEE 754 std.

answer: In IEEE 754-2008 floating point representation, binary no. is converted to 3-sections i.e., a) Sign bit b) Exponent c) Mantissa

Given $= -1.585 \times 10^{-1}$

$= (-0.1585)_{10} = -(0.00101000100100011\cdots)_2$

Scientific Notation;

$-(1.0100010010011) \times 2^{-3}$

$\therefore (-1)^3 \times (1 + \text{Fraction}) \times 2^{\text{Exponent}-15}$

0.1585×2	0.317	0
0.317×2	0.634	0
0.634×2	1.268	1
0.268×2	0.536	0
0.536×2	1.072	1
0.072×2	0.144	0
0.144×2	0.288	0
0.288×2	0.576	0
0.576×2	1.152	1
0.152×2	0.304	0
0.304×2	0.608	0
0.608×2	1.216	1
0.216×2	0.432	0
0.432×2	0.864	0
0.864×2	1.728	1
0.728×2	1.456	1
⋮	⋮	⋮

- Sign = 1
- Exponent = $15 - 3 = 12 = (01100)_2$
- Mantissa = $(0100010010)_2$

$\therefore$ IEEE Representation,

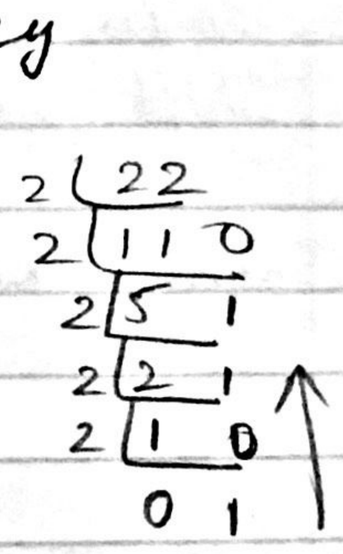
1011000100010010

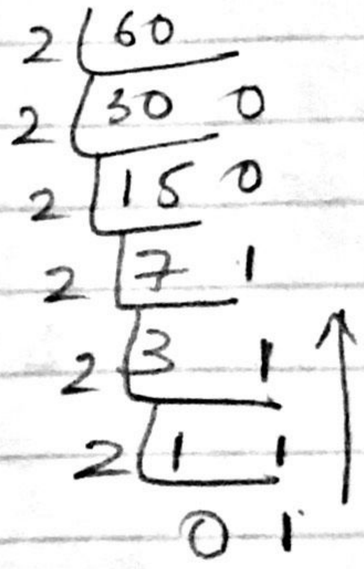
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	1	0	0	0	1	0	0	1	0

↓ S-bit — exponent — fraction.

3. Consider the 2 decimal values are 60 & 22.
Convert decimal value 60 into binary

$(60)_{10} = (0111100)_2$
$(22)_{10} = (0101100)_2$.





Refer the division example using the algorithm in the question.

Now the following steps are used to calculate the ~~for~~ 60 divided by 22 using above binary values.

0: Take initial binary values as Divisor $22 = (010110)_2$ & Remainder $(60)_{10} = (0111100)_2$
Next, Shift the binary Remainder left at a place
That means 0111100 ⟶ 1111000

1: Perform the remainder register (REM) with value of dividend (DIV) and shift remainder register left.
Do the operation REM = REM - DIV.

2: a) If REM > 0, Subtract the divisor register (DIV) from the remainder register & the result is stored in the remainder register REM = REM - DIV.
b) Otherwise, set the add operation for the divisor register (DIV) & the remainder register (REM)
c) The result value is stored in the remainder register REM = REM + DIV

3. Shift the remainder register left
4. If REM > 0, set $R_0 = 1$. If REM < 0, set $R_0 = 0$
5. Repeat the step for no of bits representing divisor n
6. After 'n' steps the remainder is in the left part & quotient is in the right part of Remainder register.

000000 111100

22/60/2
22 60 60
22 44 22
16 2

$(60)_{10} = (0111100)_2$
$(22)_{10} = (010110)_2$

→ Divisor
→ Dividend

IGNORE

1100 | 0010
R | Q

Iteration	Steps	Divisor	Remainder
	Initial Values	010110	001100
0	Shift Remainder left		011000
	Rem = Rem + Div	010110	MSB
1			

	Steps	Quotient	Divisor	Remainder
0	Init:	000000	010110 000000	000000 111100
				R Q
1.	Rem = Rem-Div	000000	010110 000000	101010 111100
	Shift left	000000	010110 000000	000000 111100
			001011 000000	000000 111100
2.	Rem = Rem-Div	000000	001011 000000	110101 111100
	Shift left	000000	001011 000000	000000 111100
			000101 100000	000000 111100
3	Rem = Rem-D	000000	000101 100000	111011 011100
	Shift left	000000	000101 100000	000000 111100
			000010 110000	000000 111100
4	Rem = Rem-Div	000000	000010 110000	111110 001100
	left	000000	000010 110000	000000 111100
			000001 011000	000000 111100
5.	Rem = Rem-Div	0000000	000001 011000	111111 100100
	left	000000	000001 011000	000000 111100
			000000 101100	000000 111100
6.	Rem = Rem-Div	000000	000000 101100	000000 010000
	Rem ≥ 0 $q_0 = 1$	000001	000000 101100	000000 010000
		000010	000000 010110	000000 010000
7				

$(60)_{10} = (0111100)_2$.

$(22)_{10} = (010110)_2$

Iteration	Steps	Quotient	Divisor	Remainder.
0	Initial Values	000000	010110 000000	000000 111100
1.	• Rem = Rem - Div.	000000	010110 000000	101010 111100
	• if R<0 ; R = R+Div shift left Q, Q_0 = 0	00000<u>0</u>	010110 000000	000000 111100
	• Shift Div ~~Register~~ left		001011 000000	000000 111100
2.	• Rem = Rem - Div	000000	001011 000000	110101 111100
	• If R<0 : R = R+Div	00000<u>0</u>	001011 000000	000000 111100
	Shift left Q, Q_0 = 0			
	• Shift Div ~~Register~~ left		000101 100000	000000 111100
3	• Rem = Rem - Div	00000 0	000101 100000	111011 011100
	• If R<0 : R = R+Div	00000<u>0</u>	000101 100000	000000 111100
	Shift left Q, Q_0 = 0			
	• Shift Div ~~Register~~ left		000010 110000	000000 111100
4	• Rem = Rem - Div	000000	000010 110000	111110 001100
	• If R<0 : R = R+Div	00000<u>0</u>	000010 110000	000000 111100
	Shift left Q, Q_0 = 0			
	• Shift Div ~~Register~~ left		000001 011000	000000 111100
5.	• Rem = Rem - Div	000000	000000 101100	111111 100100
	• If R<0 : R = R+Div	00000<u>0</u>	000000 101100	000000 111100
	Shift left Div Reg Q_0 = 0			
	• Shift Div ~~Register~~ left		000000 010110	000000 111100
6.	• Rem = Rem - Div	000000	000000 010110	000000 010000
	• if R>/0 : R = Rem	00000<u>1</u>	000000 010110	000000 010000
	Shift Left Q, Q_0 = 1			
	• ~~Shift Div Register~~ left		000000 010110	000000 010000
7	• Rem = Rem - Div	000001	000000 010110	111010
	• If R<0 : R = R+Div	00001<u>0</u>	000000 010110	000000 010000
	Shift left Q_0 = 0			
	• Shift Div ~~Register~~ left		000000 010110	000000 010000

2

16

Thus the Quotient is $= (000010)_2 = (2)_{10}$
and the remainder is $= (010000)_2 = (16)_{10}$

Hence $(60)_{10}$ divide by $(22)_{10}$ gives Quotient $(2)_{10}$
and remainder is $= (16)_{10}$