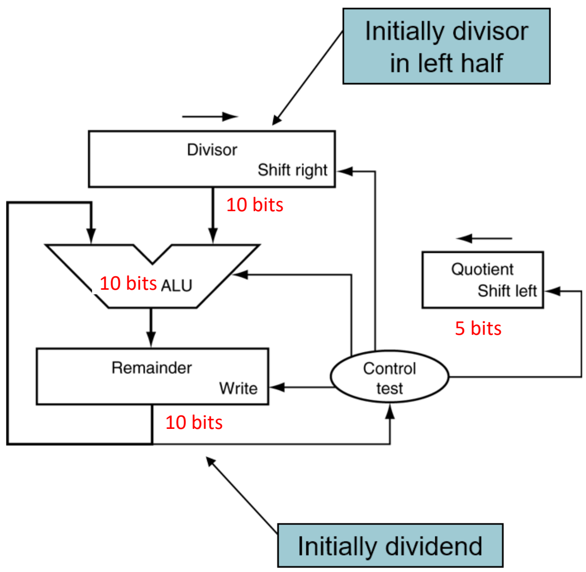
HW#8 (CSC 390): SOLUTION

Q1. The following Circuit performs the division of two 32-bit unsigned binary numbers and produces the Quotient and Remainder in the respective registers. Figure 1 shows the three steps division algorithm that the circuit performs to produce the result. Suppose, you are assigned to design a 5-bit division circuit that will perform the division of the two 5-bit unsigned numbers A=$(11011)_2$ and B=$(00101)_2$ and produce the Quotient and Remainder of the division. Draw the necessary hardware and show the results produced in each iteration of your algorithm. Hints: see figure 3.10 (page 192) of our text book.



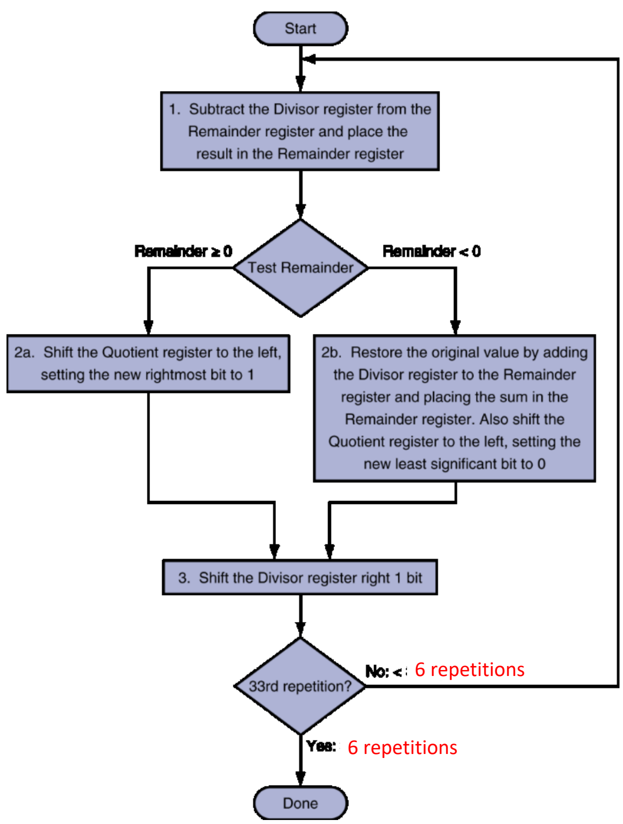


Figure 1

Iteration	Step	Quotient	Divisor	Remainder
0	**Initial Value**	**00000**	**0010100000**	**0000011011**
1	1: Rem = Rem - Div	**00000**	**0010100000**	**1101111011**
	2b : Rem < 0 => +Div, sll Q, Q0 = 0	**00000**	**0010100000**	**0000011011**
	3: Shift Div right	**00000**	**0001010000**	**0000011011**
2	1: Rem = Rem - Div	**00000**	**0001010000**	**1111001011**
	2b : Rem < 0 => +Div, sll Q, Q0 = 0	**00000**	**0001010000**	**0000011011**
	3: Shift Div right	**00000**	**0000101000**	**0000011011**
3	1: Rem = Rem - Div	**00000**	**0000101000**	**1111110011**
	2b : Rem < 0 => +Div, sll Q, Q0 = 0	**00000**	**0000101000**	**0000011011**
	3: Shift Div right	**00000**	**0000010100**	**0000011011**
4	1: Rem = Rem - Div	**00000**	**0000010100**	**0000000111**
	2a : Rem ≥ 0 => sll Q, Q0 = 1	**00001**	**0000010100**	**0000000111**
	3: Shift Div right	**00001**	**0000001010**	**0000000111**
5	1: Rem = Rem - Div	**00001**	**0000001010**	**1111111101**
	2b : Rem < 0 => +Div, sll Q, Q0 = 0	**00010**	**0000001010**	**0000000111**
	3: Shift Div right	**00010**	**0000000101**	**0000000111**
6	1: Rem = Rem - Div	**00010**	**0000000101**	**0000000010**
	2a : Rem ≥ 0 => sll Q, Q0 = 1	**00101**	**0000000101**	**0000000010**
	3: Shift Div right	**00101**	**0000000010**	**0000000010**

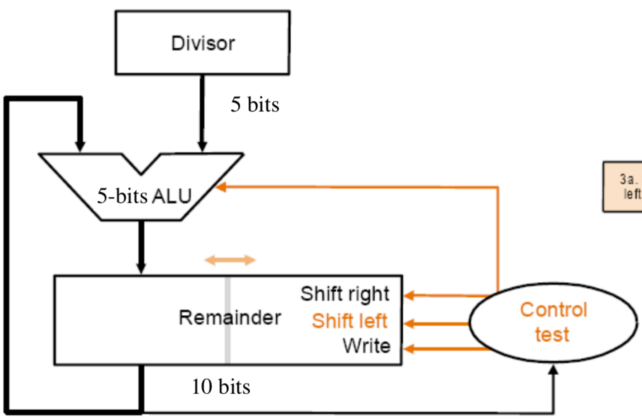
A=$(11011)_2$ ➔ 27 ; B=$(00101)_2$ = 5 ; A/B = 27/5 ➔

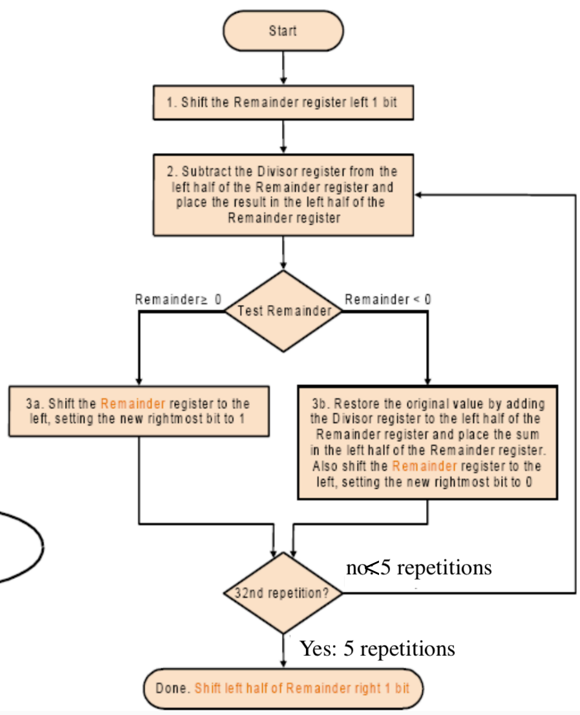
Quotient = 5 → **000101** and

Remainder = 2 → **00010**

Q2.

Interation	Divisor	Hardware design	
		Step	Remainder
0	00101	Initial value	00000 11011
		Shift remainder left by 1	00001 10110
1	00101	Remainder = remainder -division	11100 10110
		Remainder < 0 => +division, shift left; r0 = 0	00011 01100
2	00101	Remainder = remainder -division	11110 01100
		Remainder < 0 => +division, shift left; r0 = 0	00110 11000
3	00101	Remainder = remainder -division	00001 11000
		Remainder > 0 => shift left; r0 = 1	00011 10001
4	00101	Remainder = remainder -division	11110 10001
		Remainder < 0 => +division, shift left; r0 = 0	00111 00010
5	00101	Remainder = remainder -division	00010 00010
		Remainder > 0 => shift left; r0 = 1	00100 00101
Done	00101	Shift “left half of remainder” right by 1	00010 00101





Quotient: 00101 Remainder 00010

Q3.

- $(6.725)_{10} = 110.101110011001100110011_2 = 1.10101110011001100110011_2 \text{ x } 2^2$

So, we represent $(6.725)_{10}$ as

$(-1)^0 \text{ x } (1+ 10101110011001100110011_2) \text{ x } (2)^2$

S = 0
fraction = 10101110011001100110011
exponent = 2 + bias = 2+127 = 129_{10} = 10000001_2

=> Floating point format:

$0100\ 0000\ 1101\ 0111\ 0011\ 0011\ 0011\ 0011_2$ = **$40d73333_{16}$**

- $-0.3125_{10} = 0.0101_2 = 1.01_2 \text{ x } 2^{-2}$

So, we represent -0.3125_{10} as

$(-1)^{-1} \text{ x } (1+ 0.01_2) \text{ x } 2^{-2}$

S = 1
fraction = 01000000000000000000000
exponent = -2 + bias = -2 +127 = 125_{10} = 01111101_2

=> Floating point format:

$1011\ 1110\ 1010\ 0000\ 0000\ 0000\ 0000\ 0000_2$ = **$BEA00000_{16}$**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x40d73333	0xbea00000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000