Q1

a.

28(10) = 00011100(2)

91(10)= 01011011(2)

```
    00011100
 +  01011011
 =  01110111 = 119(10)
```

b.

102(10) =01100110(2)

75(10)=01001011(2)

-75(10)=10110100(2)+1(2)=10110101(2)

```
    01100110
 +  10110101
 =  00011011   =27(10)
```

c.

12(10)=00001100(2)

5(10)=00000101(2)

-5(10)=1111011(2)

```
          00001100
        *  11111011
      ____________________
     000101111000100
```

=11000100(2) last 8 bits =-60(10)

d.

240(10)=11110000(2)

10(10)=00001010(2)

11110000 / 00001010 = 00011000 r 00000000 = 24(10)

Q2

The range of 5-bit 2's complement should be -16 to 15

So

A+B = -22 overflow does occur

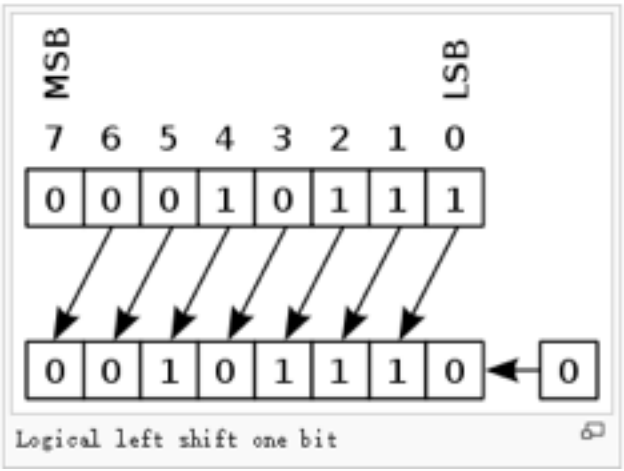
A-B = -2 overflow does not occur

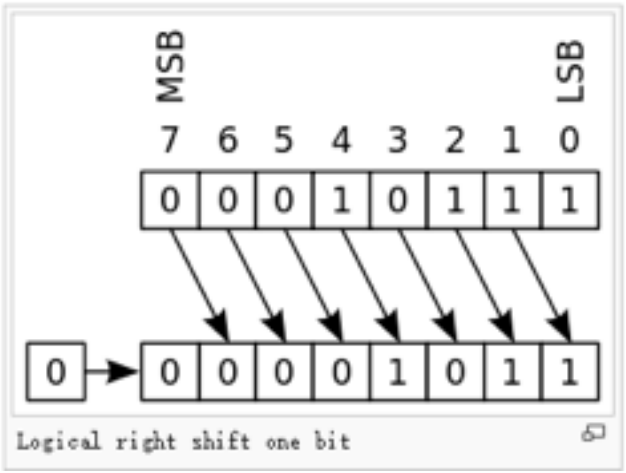
-A+B = 2 overflow does not occur

-A-B = 22 overflow does occur

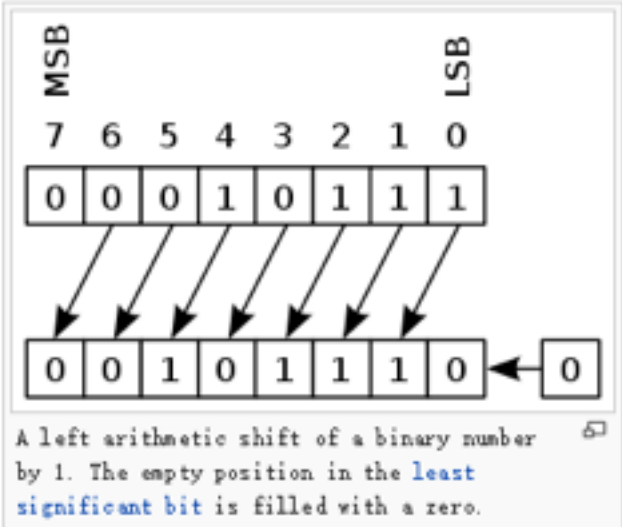
Q3

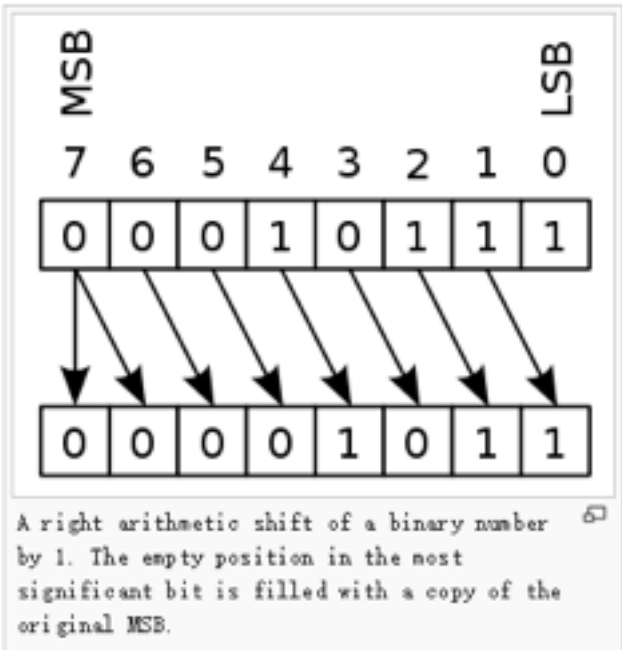
a.



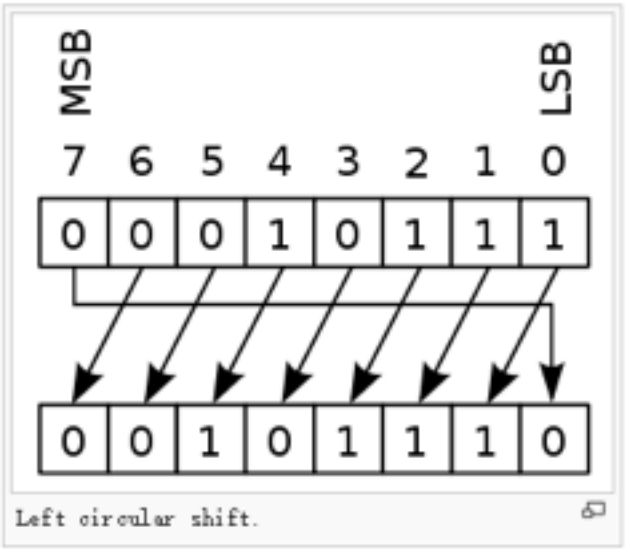


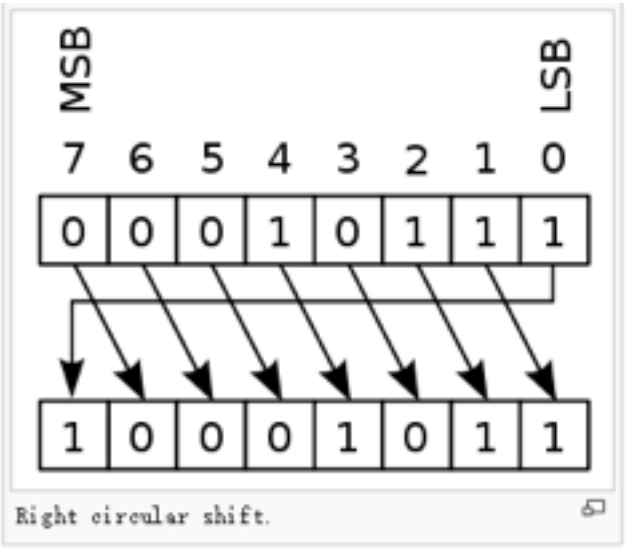
https://en.wikipedia.org/wiki/Logical_shift





https://en.wikipedia.org/wiki/Arithmetic_shift





https://en.wikipedia.org/wiki/Circular_shift

b.

Arithmetic left shifts are, with two exceptions, identical in effect to logical left shifts.

Exception one is the minor trap that arithmetic shifts may trigger arithmetic overflow whereas logical shifts do not. Obviously, that exception occurs in real world use cases only if a trigger signal for such an overflow is needed by the design it is used for. Exception two is the MSB is preserved. Processors usually do not offer logical and arithmetic left shift operations with a significant difference, if any.

https://en.wikipedia.org/wiki/Arithmetic_shift

c.

For division arithmetic Shift-Right is unpredictable.