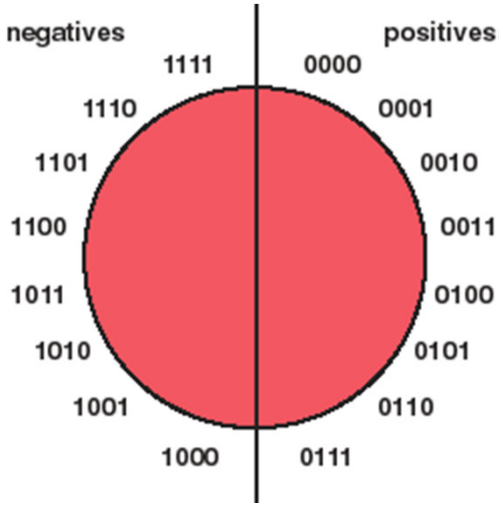
Ying….Yang…1….-1

CSCI 255



http://cs.furman.edu

Signed Nums/Shifting/Arithmetic Ops

- In mathematics, negative integers exists -> forced to do it binary
- In a binary string, the MSB is sacrificed as the signed bit
  - 0 => Positive Value
  - 1 => Negative Vale
- Two different ways were developed:
  - One's complement
  - Two's complement
- Shifting LEFT/RIGHT
- Adding/Subtracting
- Multiplying/Dividing

- Let's start with stuff that is already known: ADDITION
- Single bit additions looks like this:

0 + 0 = 0

0 + 1 = 1

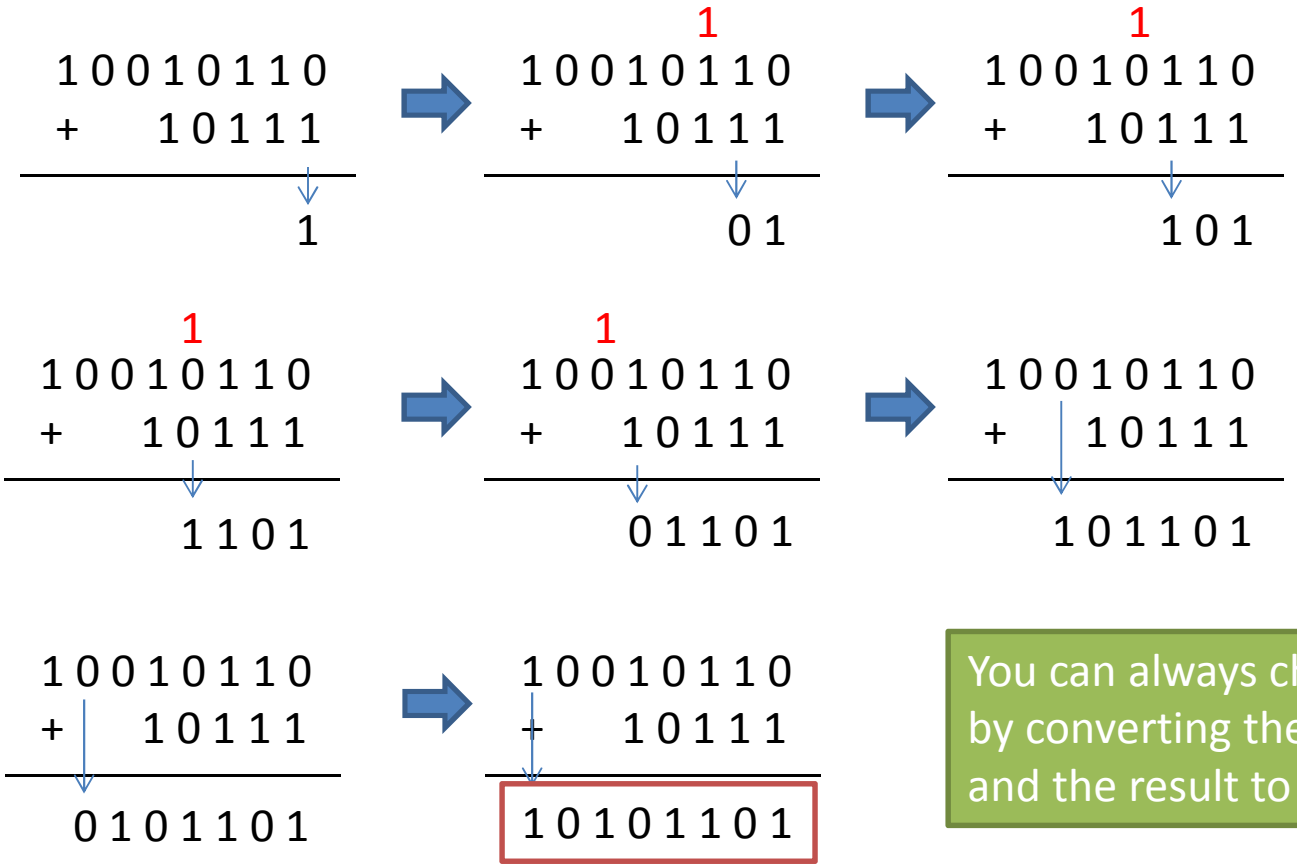
1 + 0 = 1

1 + 1 = 0, carry of 1

These are not OR operations!!

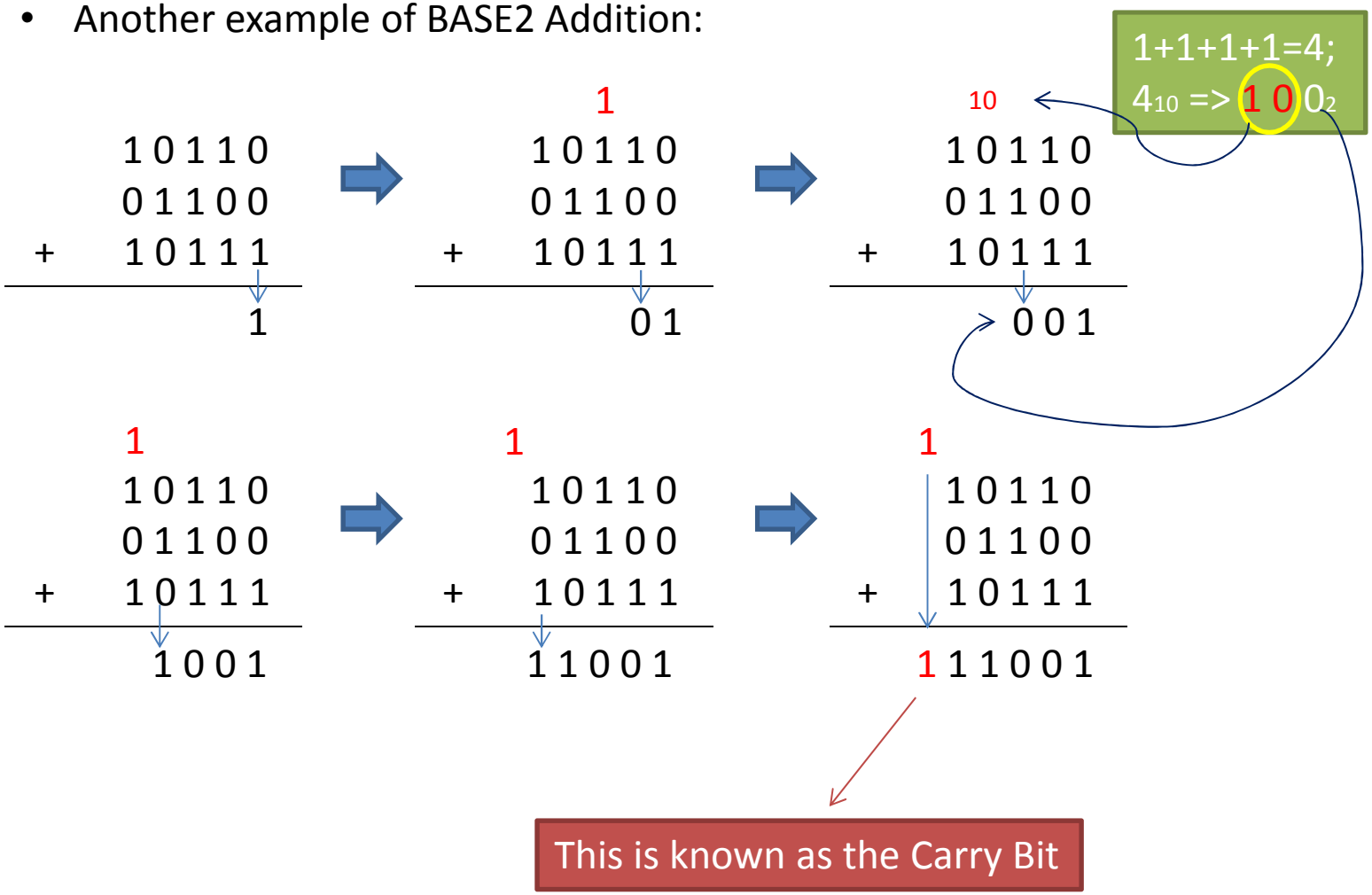
We are actually adding….

- A quick example of BASE2 Addition:



You can always check the result by converting the two numbers and the result to BASE10

- Another example of BASE2 Addition:



- If we can ADD, then we can SUBTRACT
- Before we move to that… We must learn the ways of the machines
- It's a simple trick: If you just add the first number by the negative representation of the second number, you end up subtracting
- This is known as "Subtracting by Adding":

$$\begin{array}{r} 9 \\ -\ 7 \\ \hline 2 \end{array} \quad = \quad \begin{array}{r} 9 \\ +\ -7 \\ \hline 2 \end{array}$$

- So, we must learn in how to turn Binary numbers to its negative representation

- There are two ways:
  - 1's Complement
  - 2's Complement

- As previously mentioned, the MSB is used as the SIGN BIT:
  - 0 => Positive Value
  - 1 => Negative Vale

- Ex:

  0000 1010 => +value (+10)

  1001 0011 => -value (-108)

- 1's complement: First and quick way to represent negation
- All you do: Complement all the bits
- Ex:

0 0 0 0 0 1 0 1 => + 5

Applying the NOT operation

1 1 1 1 1 0 1 0 => - 5

How to know it is negative 5?
- MSB/Sign bit is 1 (negative)
- The zero placements represents the signed value (1+4=5)

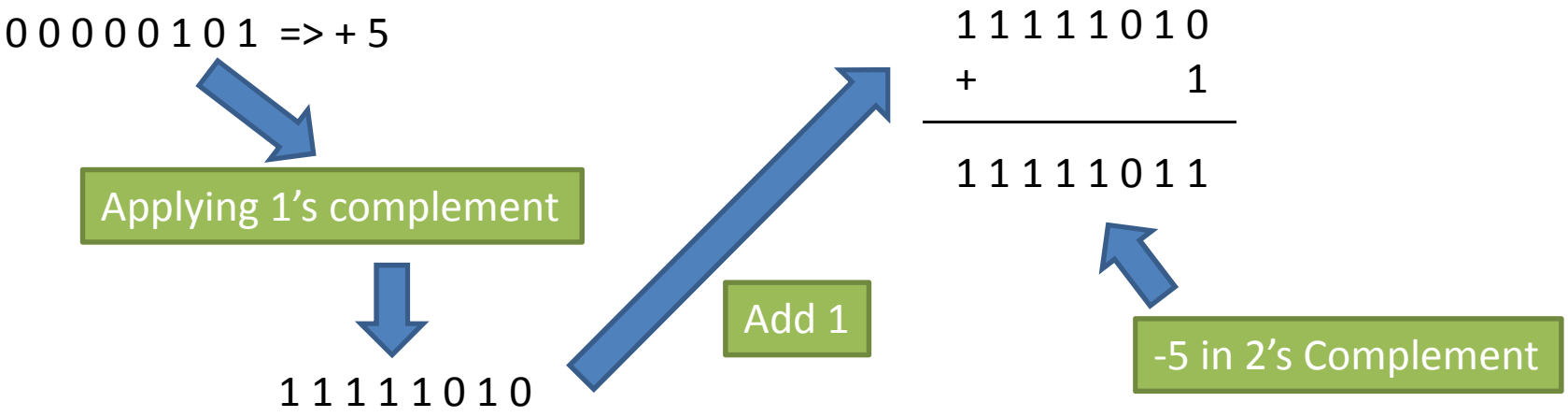
- The trend/sequence of negative values using 1's complement is:

Negative binary	Signed value
1 1111	0
1 1110	-1
1 1101	-2
1 1100	-3
1 1011	-4
1 1010	-5
1 1001	-6
1 1000	-7

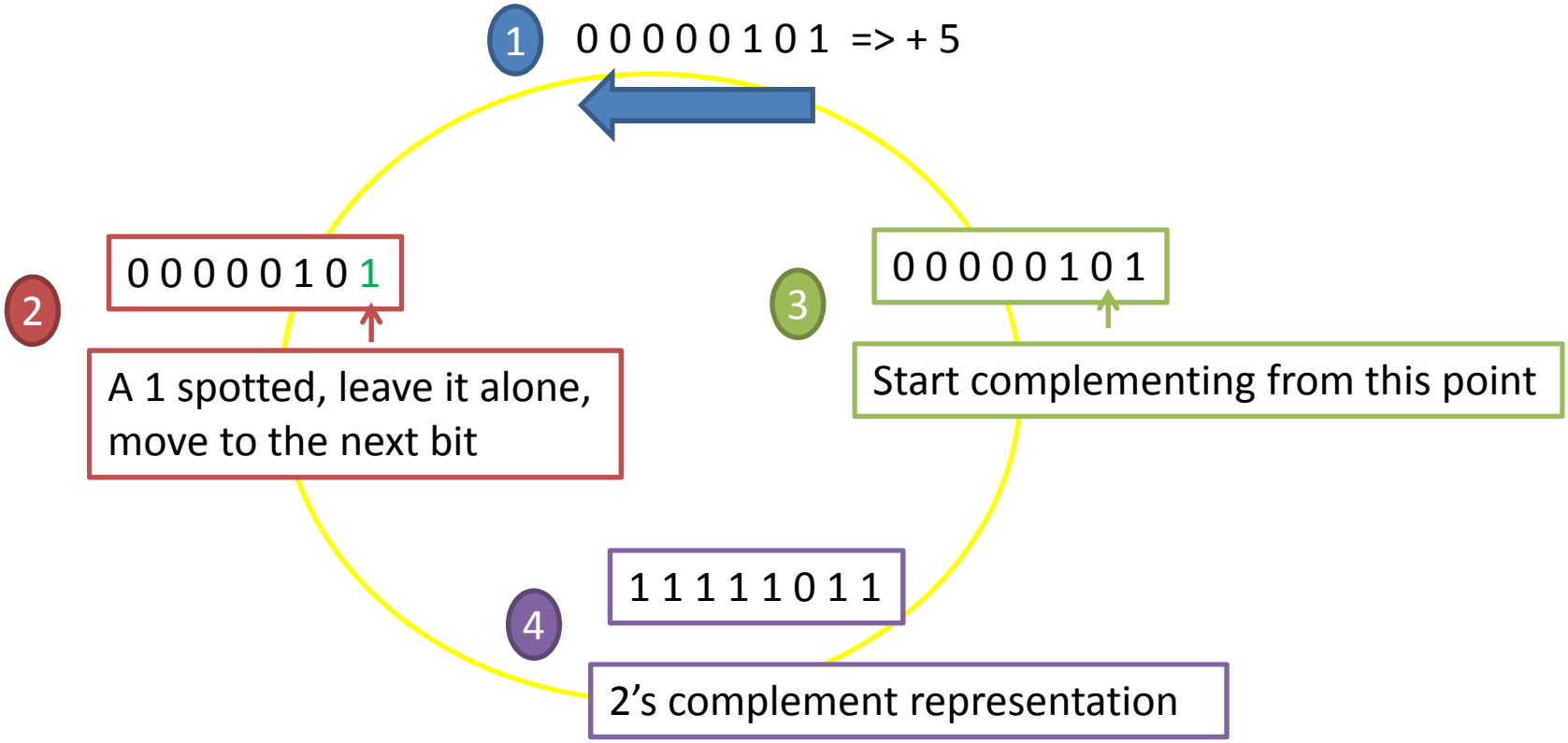
Negative binary	Signed value
1 0111	-8
1 0110	-9
1 0101	-10
1 0100	-11
1 0011	-12
1 0010	-13
1 0001	-14
1 0000	-15

- 2's complement: Improvement of 1's complement
- What's the improvement? It's a great midterm question….
- Two ways to perform 2's complement

First way: perform 1's complement and add 1



2nd way: From right-to-left, spot the first logical 1 and complement the rest of the bits



- The trend/sequence of negative values using 2's complement is:

Negative binary	Signed value
1 1111	-1
1 1110	-2
1 1101	-3
1 1100	-4
1 1011	-5
1 1010	-6
1 1001	-7
1 1000	-8

Negative binary	Signed value
1 0111	-9
1 0110	-10
1 0101	-11
1 0100	-12
1 0011	-13
1 0010	-14
1 0001	-15
1 0000	-16

- Let's continue with stuff that is already known: SUBTRACTION
- Single bit subtraction looks like this:

0 - 0 = 0

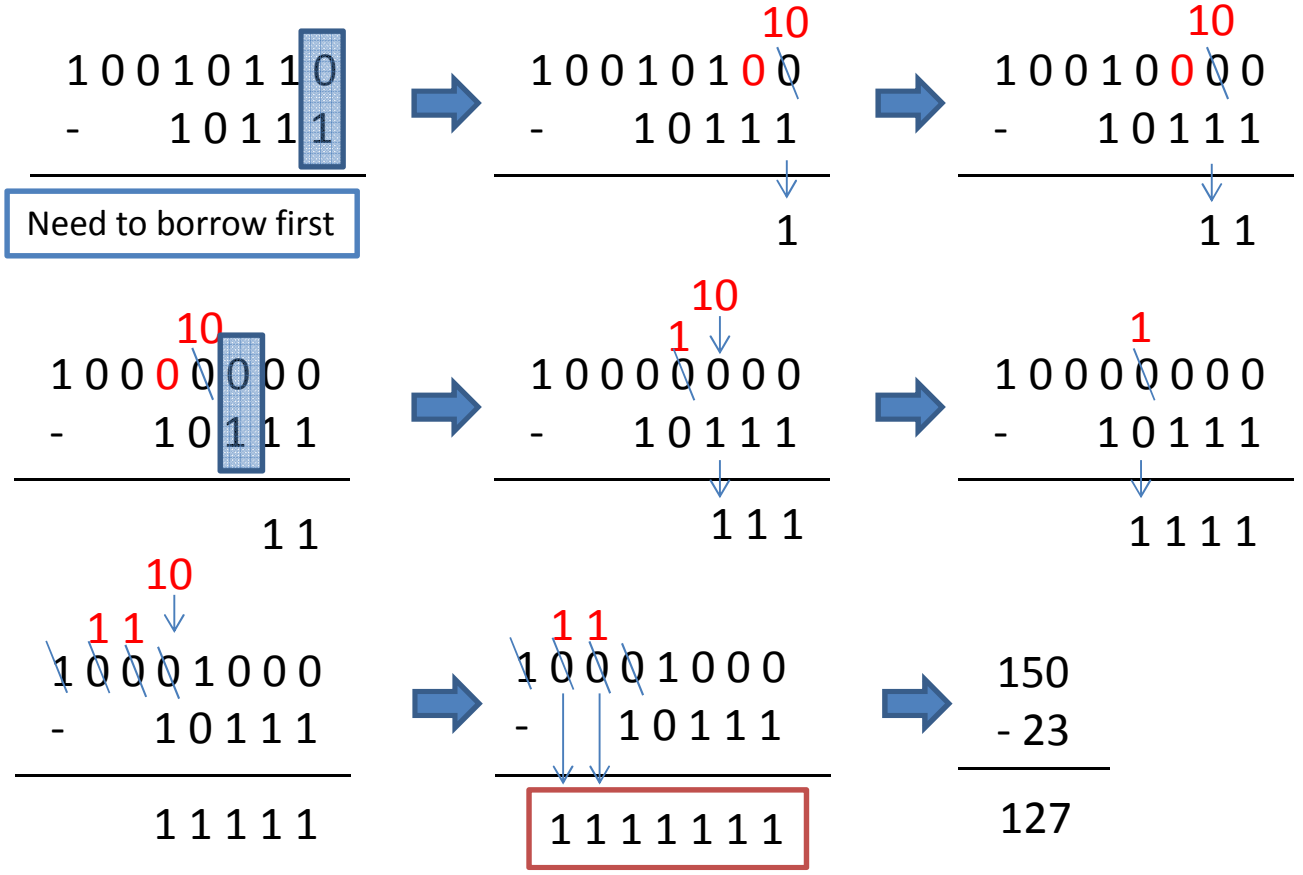
0 - 1 = 1, with borrow

1 - 0 = 1

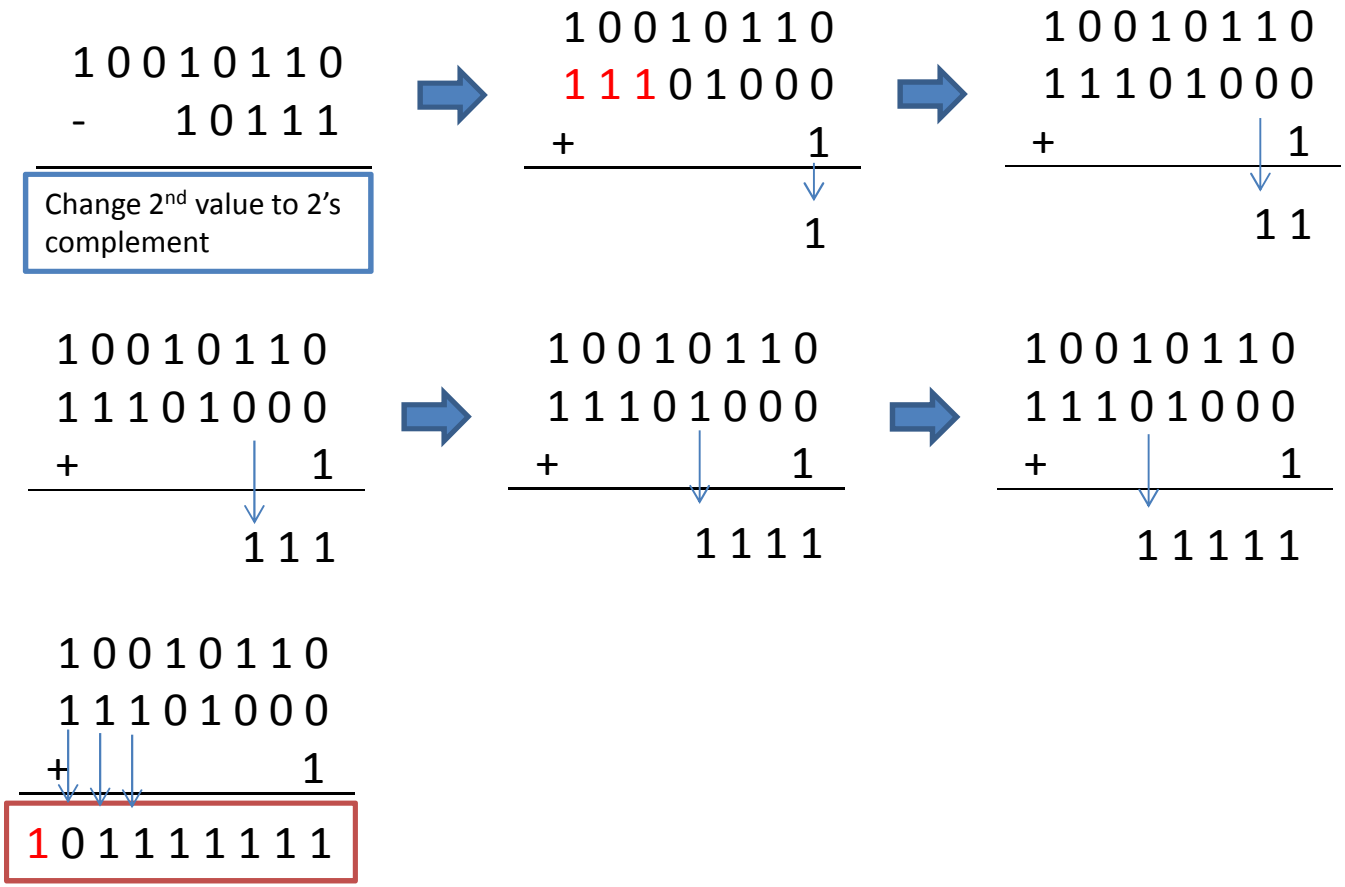
1 - 1 = 0

These are not 1's or 2's Complements

- A quick example of BASE2 Subtraction:



- A quick example of 2's complement Subtraction:



- Stuff that is already known: MULTIPLICATION
- Single bit multiplication looks like this:

0 ● 0 = 0

0 ● 1 = 0

1 ● 0 = 0

1 ● 1 = 1

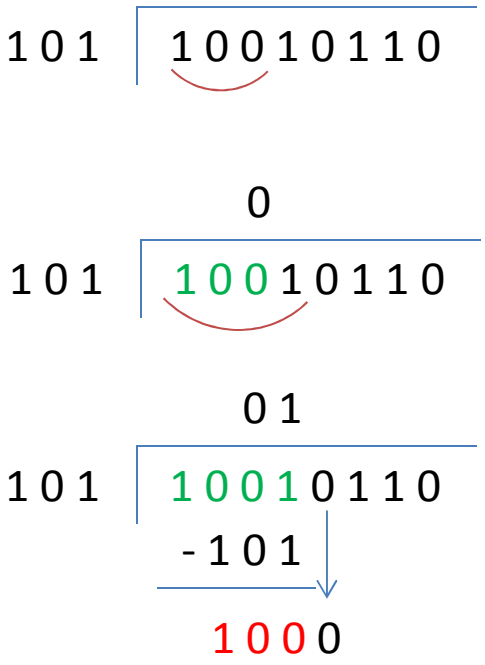
These are not AND operations!!

But it acts like AND logic

- A quick example of BASE2 Multiplication:

```
    1 0 0 1 0 1 1 0
    x     1 0 1 1 1
  -------------------
    1 0 0 1 0 1 1 0
  1 0 0 1 0 1 1 0
1 0 0 1 0 1 1 0
+ 1 0 0 1 0 1 1 0
-------------------
1 1 0 1 0 1 1 1 1 0 1 0
```

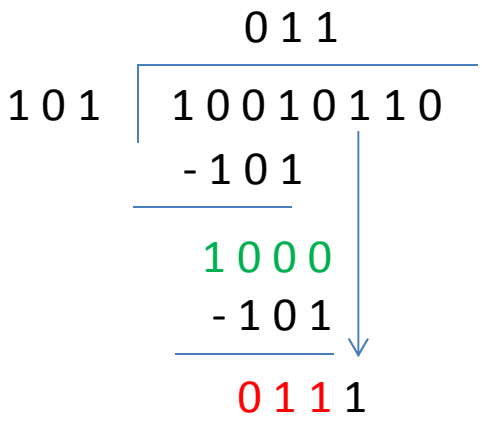
- A quick example of BASE2 Division:



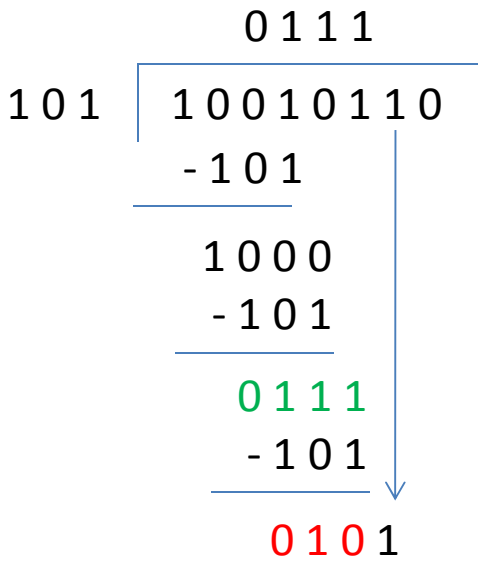
Since the divisor is a 3-bit number =>
From left-to-right, observe the first 3-bits

4 < 5; Quotient = 0. Add next bit

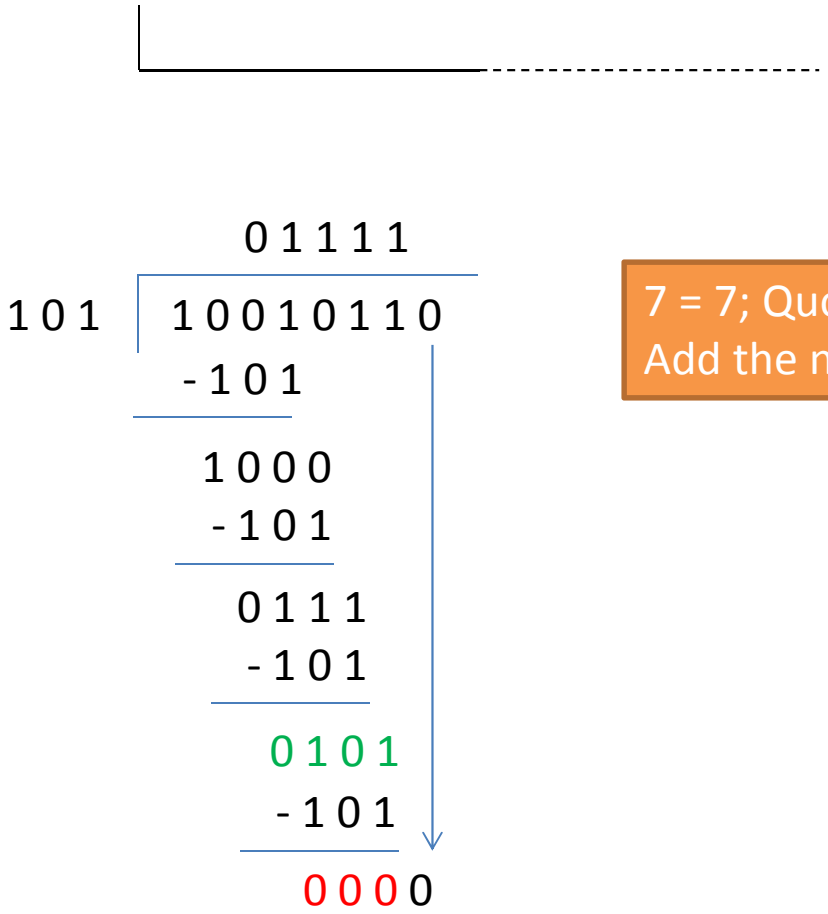
9 > 5; Quotient = 1. Multiply/Subtract.
Add the next bit.



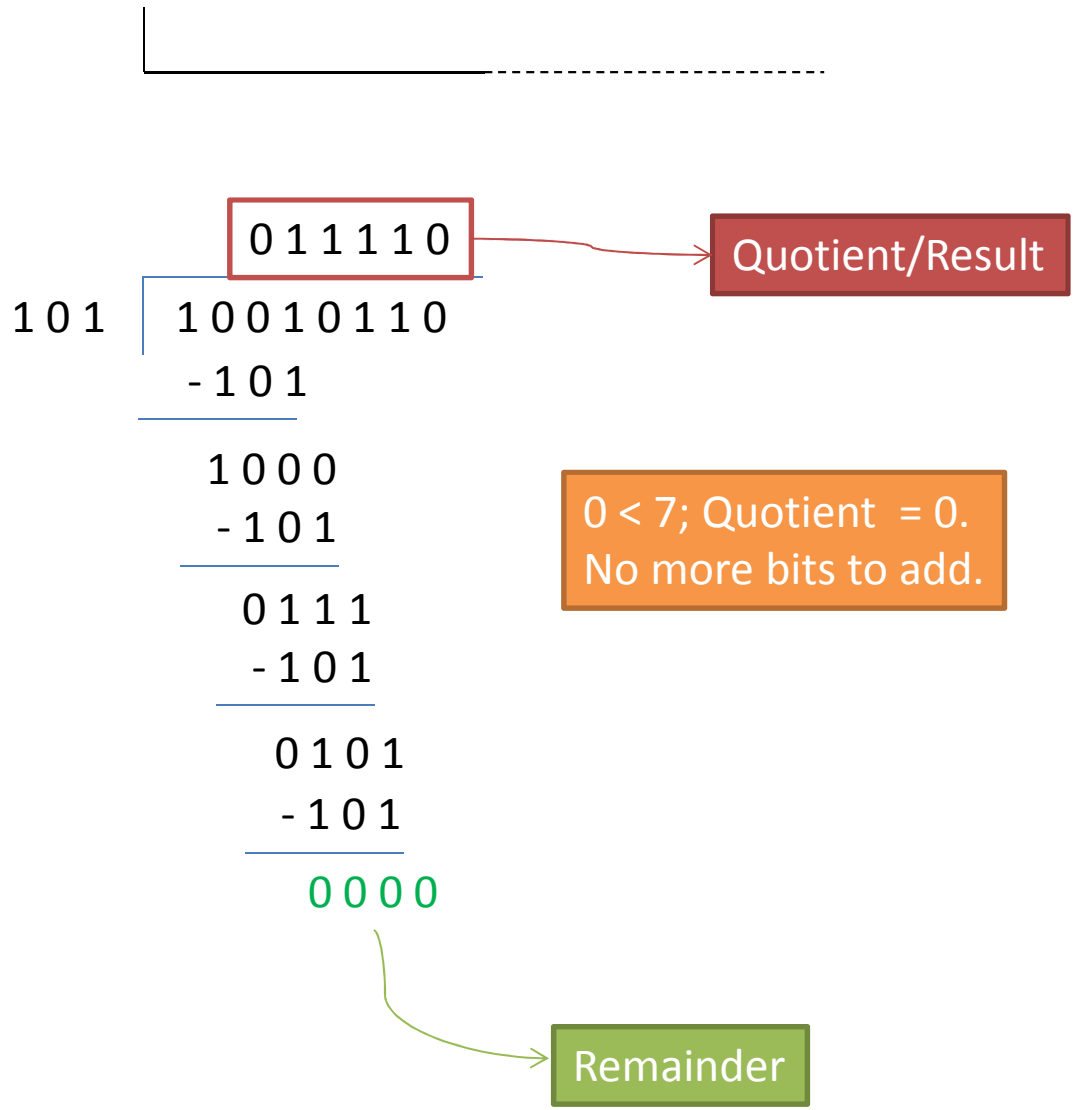
8 > 5; Quotient = 1. Multiply/Subtract. Add the next bit.



7 > 5; Quotient = 1. Multiply/Subtract. Add the next bit.

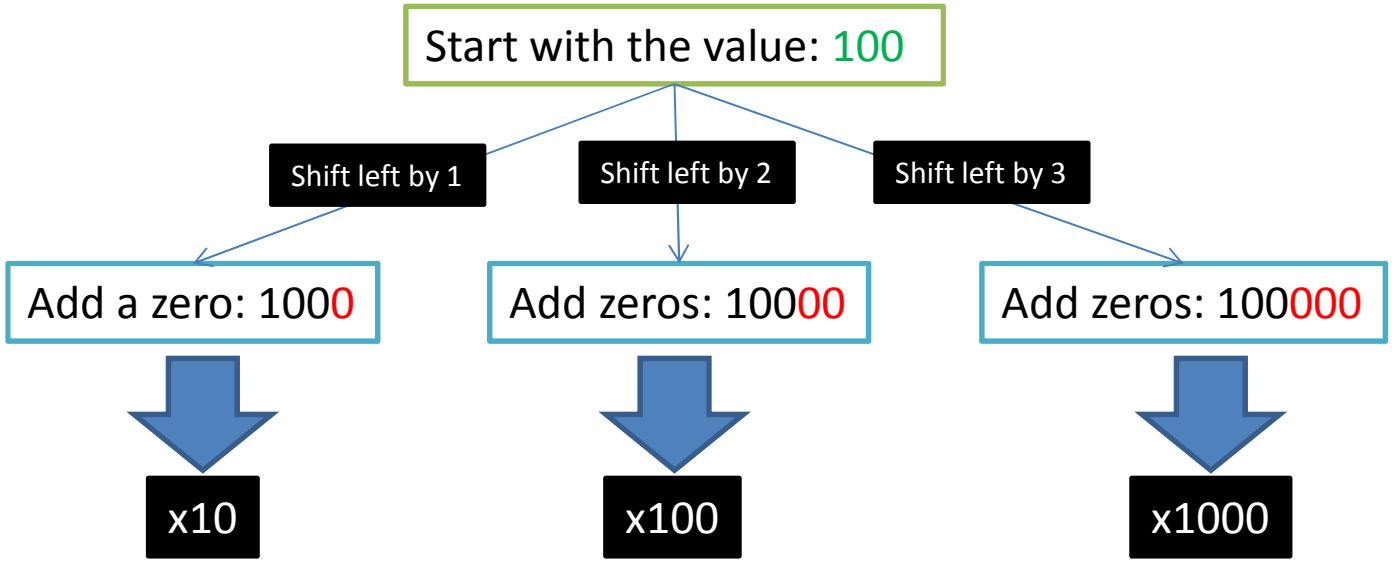


7 = 7; Quotient = 1. Multiply/Subtract. Add the next bit.

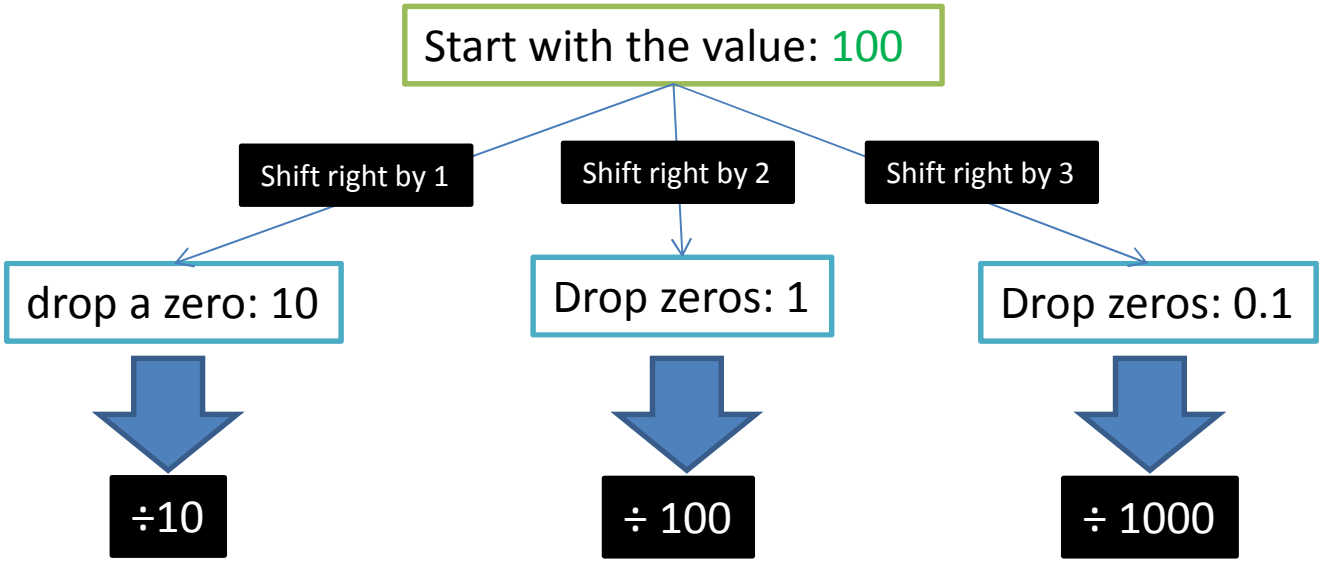

011110
Quotient/Result
101
10010110
-101
1000
-101
0111
-101
0101
-101
0000
0 < 7; Quotient = 0.
No more bits to add.
Remainder

- As seen on the multiplication and division examples…it takes many steps to generate an output
- The more steps = more clock cycles on the machine/board
- Instructions like SHIFTING binary values helps reduce the # of steps
- Only two ways to shift: Left or Right
- When shifting by 1 to the left/right: the digits move one place on that direction
- You can shift by more than 1

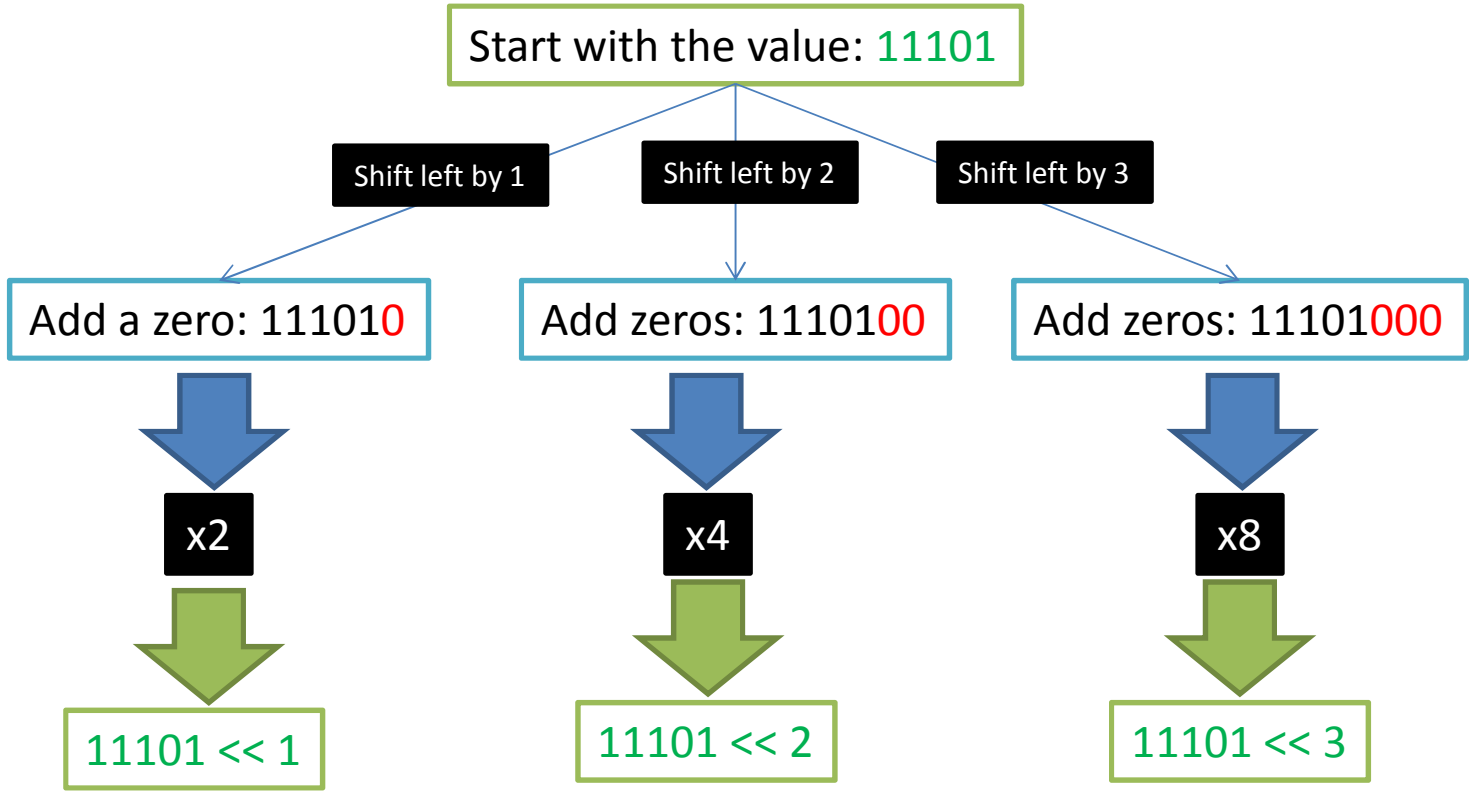
- Example of shifting a DECIMAL value: SHIFT LEFT



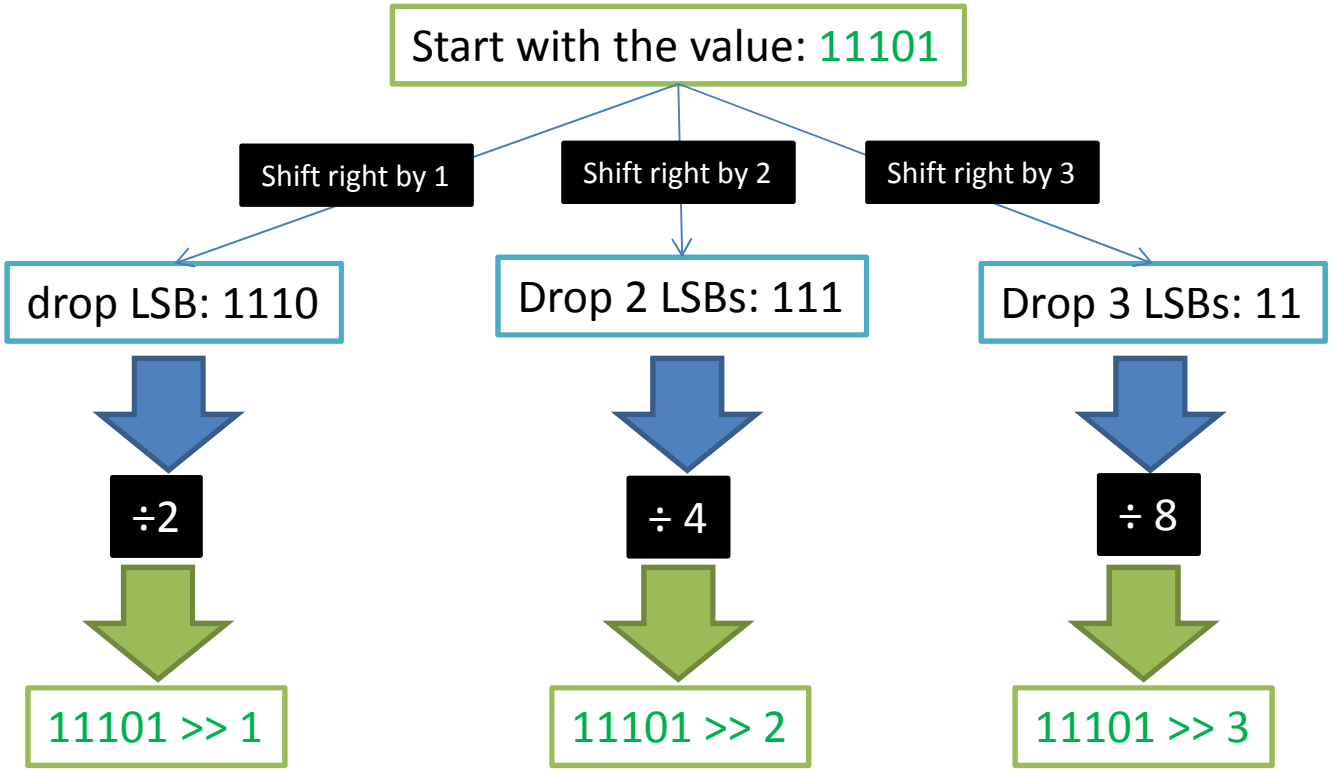
- Example of shifting a DECIMAL value: SHIFT RIGHT



- Example of shifting a BINARY value: SHIFT LEFT



- Example of shifting a DECIMAL value: SHIFT RIGHT



Signed Nums/Shifting/Arithmetic Ops

http://thebeausejourpulpit.files.wordpress.com/2012/05/end-of-the-world.jpg