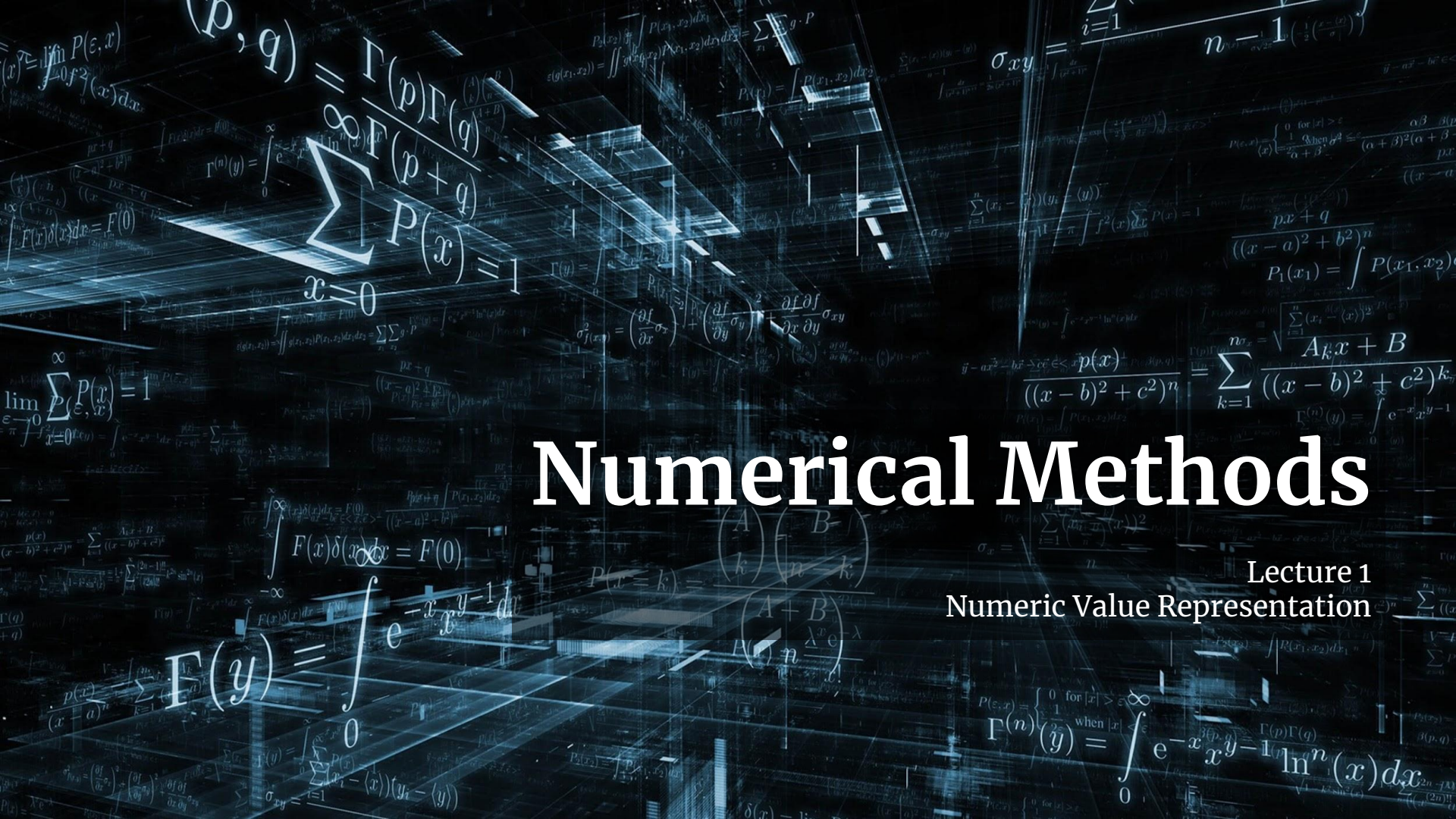


Numerical Methods

Lecture 1
Numeric Value Representation

Overview

- Number System
- Number Conversion
- Representation of Numbers
- Computer Arithmetic
- Errors in Arithmetic

Number System

Number System

- Non-positional Number System
  - Roman Numbering
- Positional Number System
  - Decimal
  - Binary
  - Hexadecimal

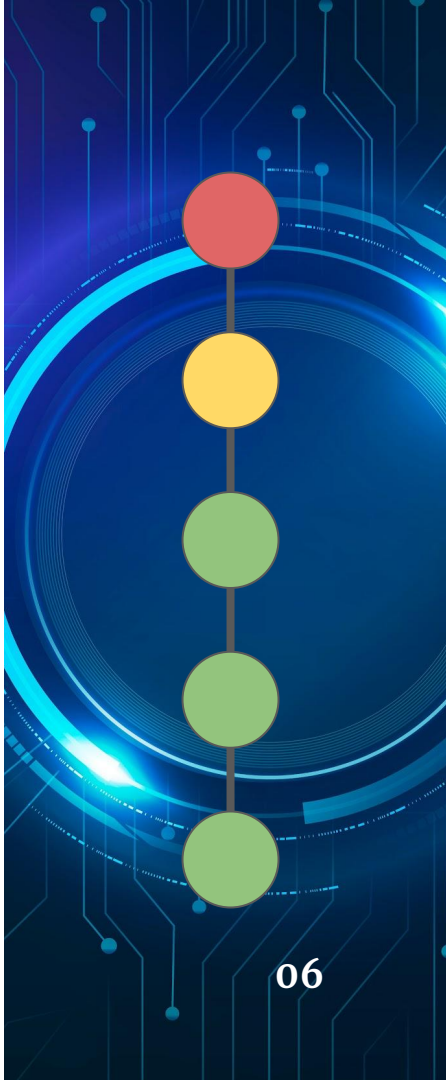
Number Conversion

Number Conversion

- Decimal to a number of base *b*

```
d ← the decimal number
b ← the base of the number to be converted to
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

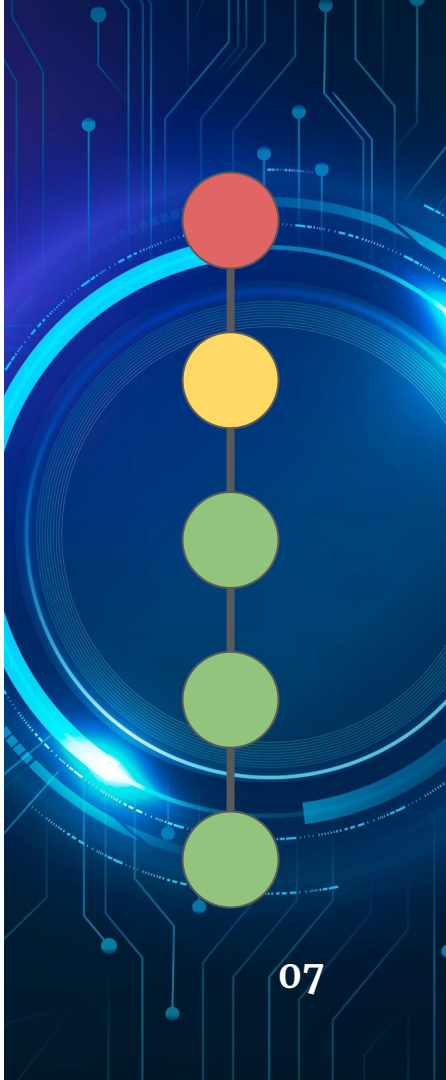


Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← the decimal number
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

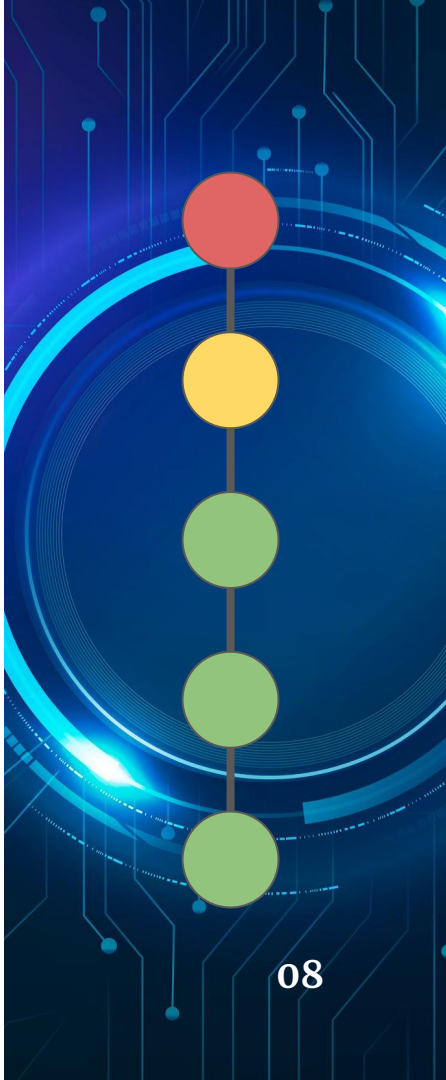


Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```



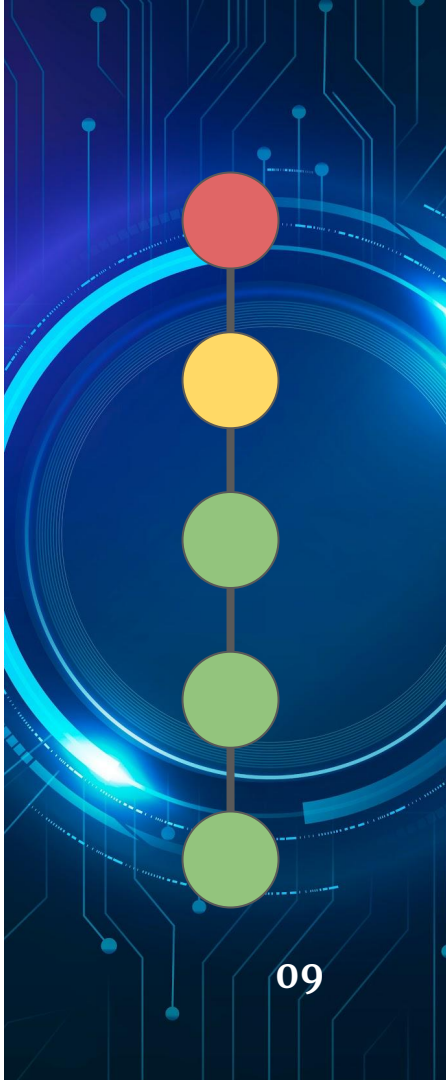
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 1
d = 9
s = “1”



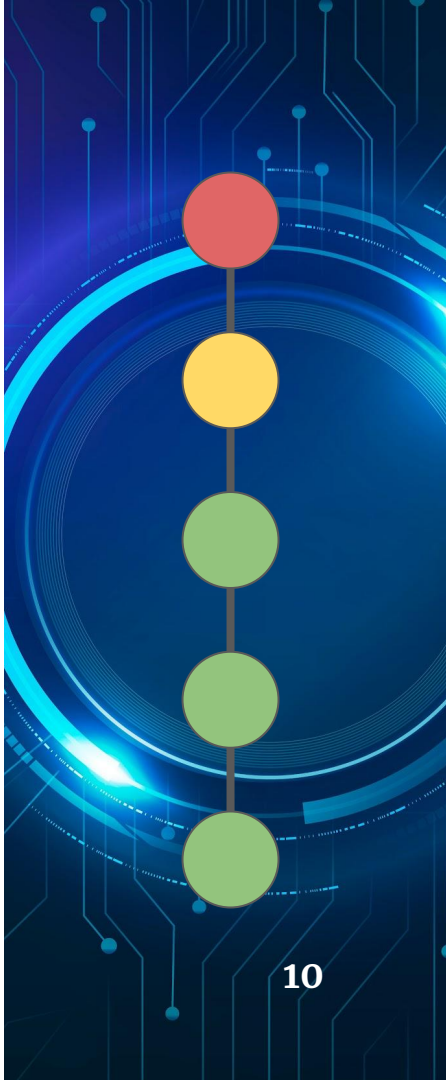
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 1
d = 4
s = “11”



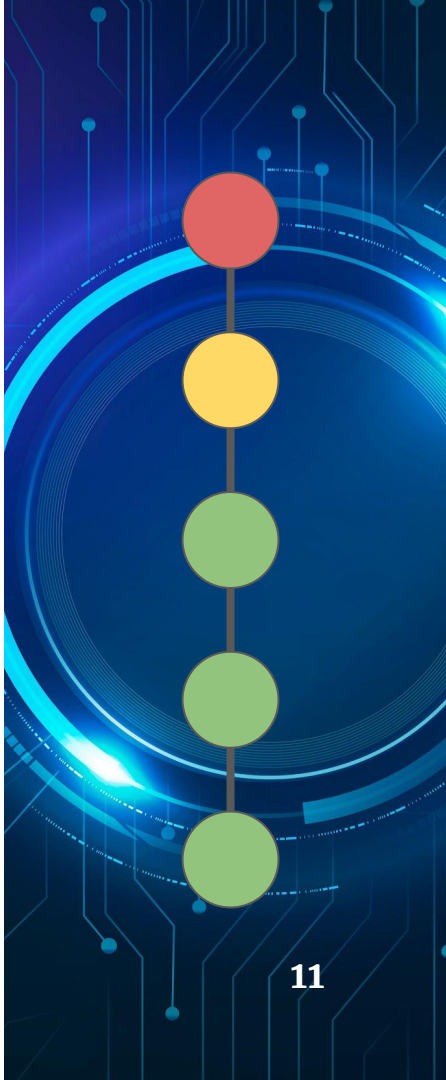
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 0
d = 2
s = “110”



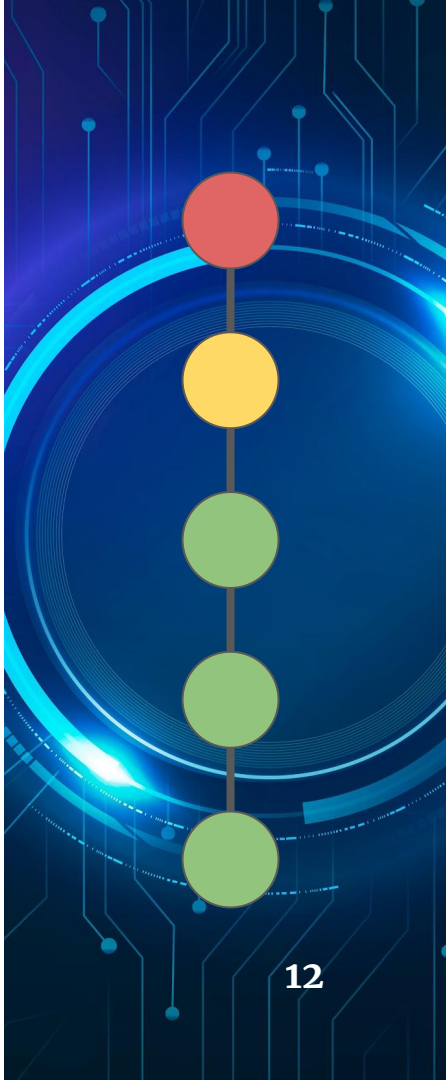
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 0
d = 1
s = “1100”



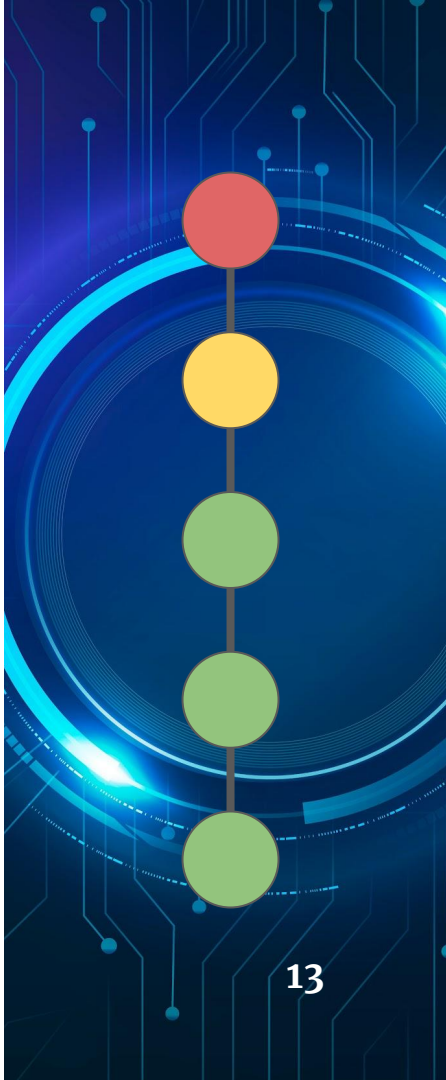
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 1
d = 0
s = “11001”



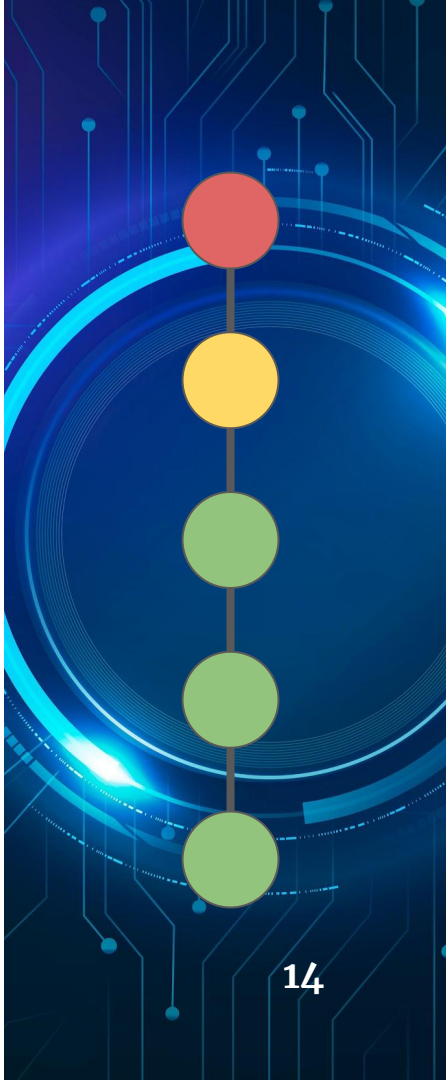
Number Conversion

- Decimal to a number of base 2 (Binary)

```
d ← 19
b ← 2
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

s = “10011”

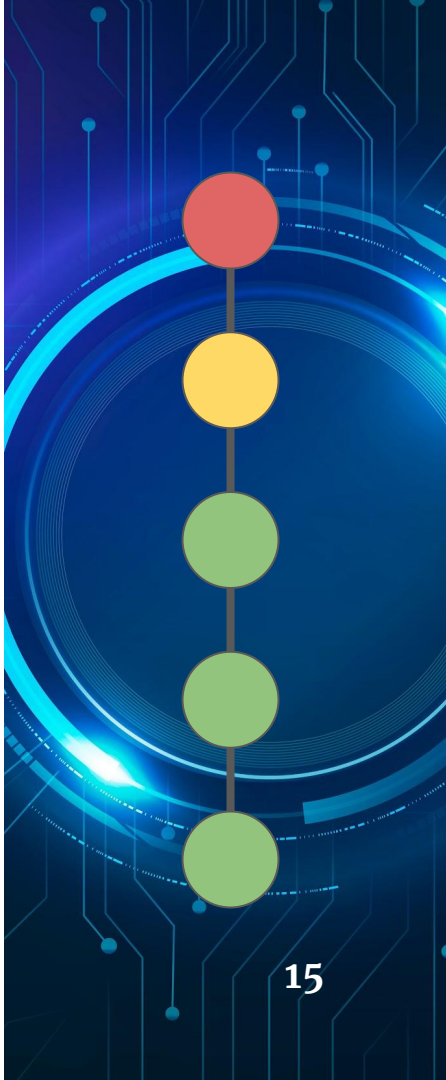


Number Conversion

- Decimal to a number of base *16* (Hexadecimal)

```
d ← the decimal number
b ← 16
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```



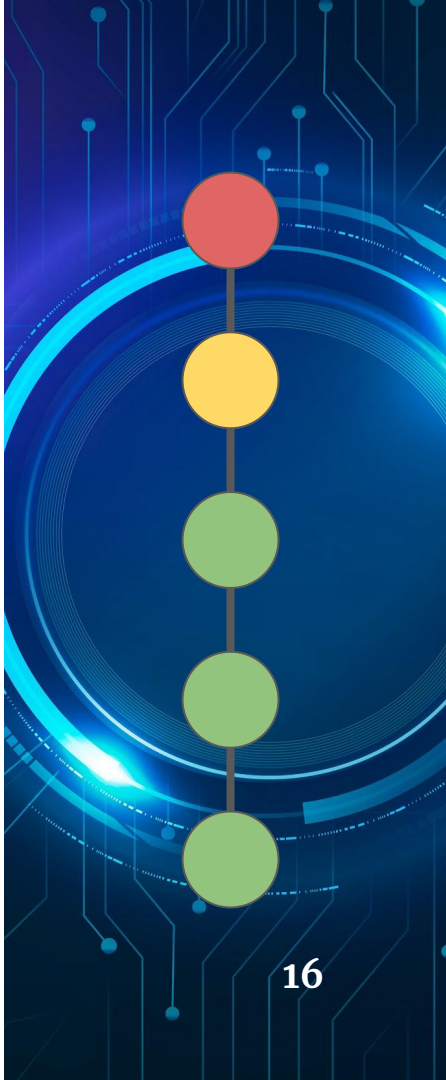
Number Conversion

- Decimal to a number of base *16* (Hexadecimal)

```
d ← 26
b ← 16
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 10
d = 1
s = “A”



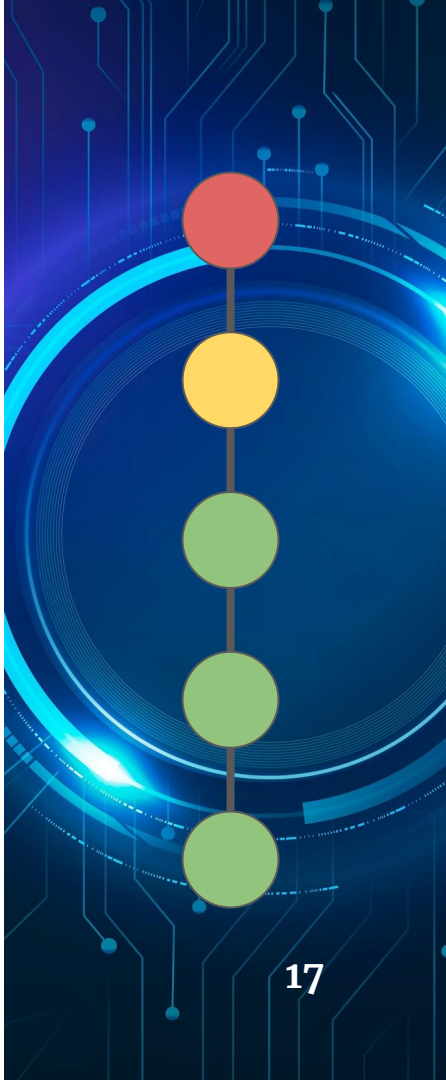
Number Conversion

- Decimal to a number of base *16* (Hexadecimal)

```
d ← 26
b ← 16
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

q = 1
d = 0
s = “A1”



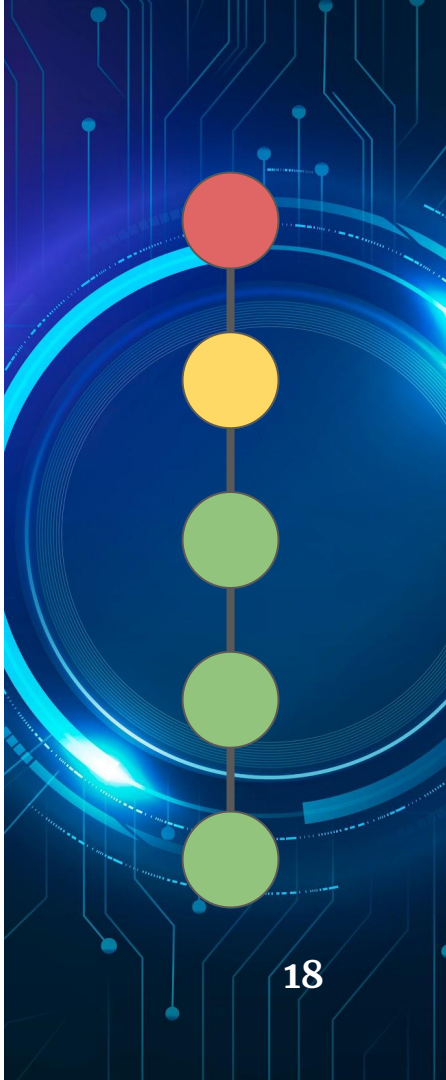
Number Conversion

- Decimal to a number of base *16* (Hexadecimal)

```
d ← 26
b ← 16
s ← an empty string

While d > 0
        q ← d%b
        d ← d/b
        s ← s + correspondingSymbol(q)
s ← reverse(s)
print → s
```

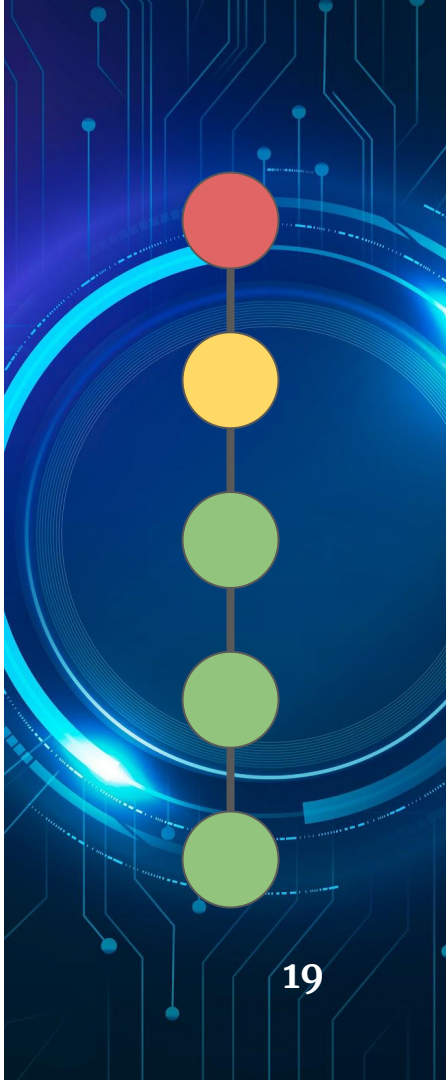
s = “1A”



Number Conversion

- A number of base *b* to Decimal

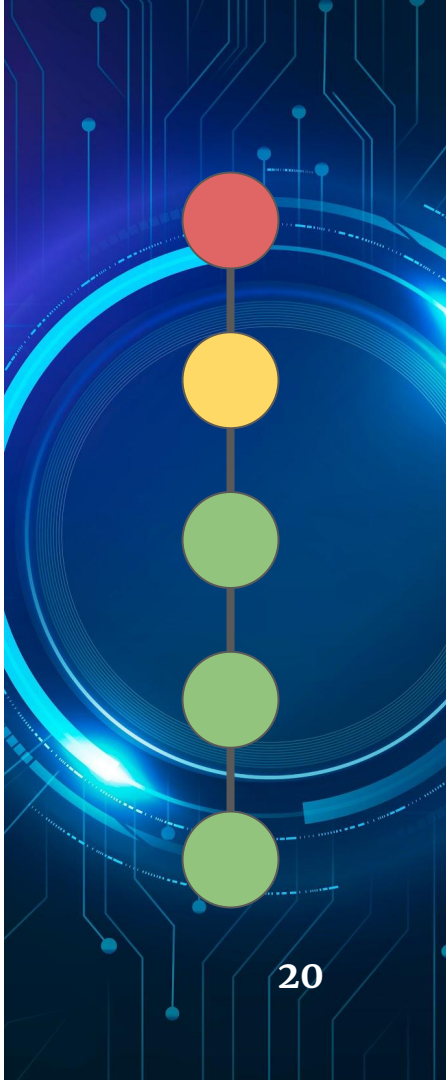
```
n ← the base b number
b ← the base of the number to convert from
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```



Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

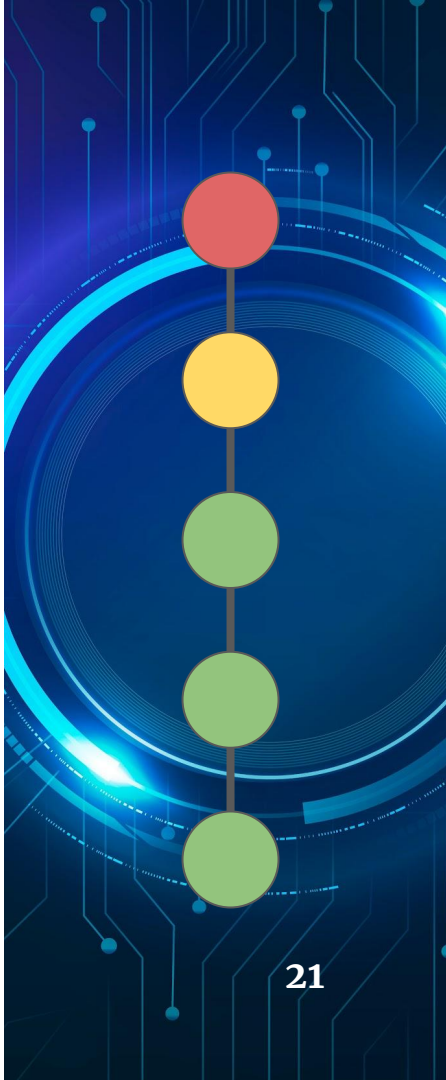


Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

n = 10011
q = 1
$q = 1 * 2^0 = 1$
n = 1001
s = 1
i = 1

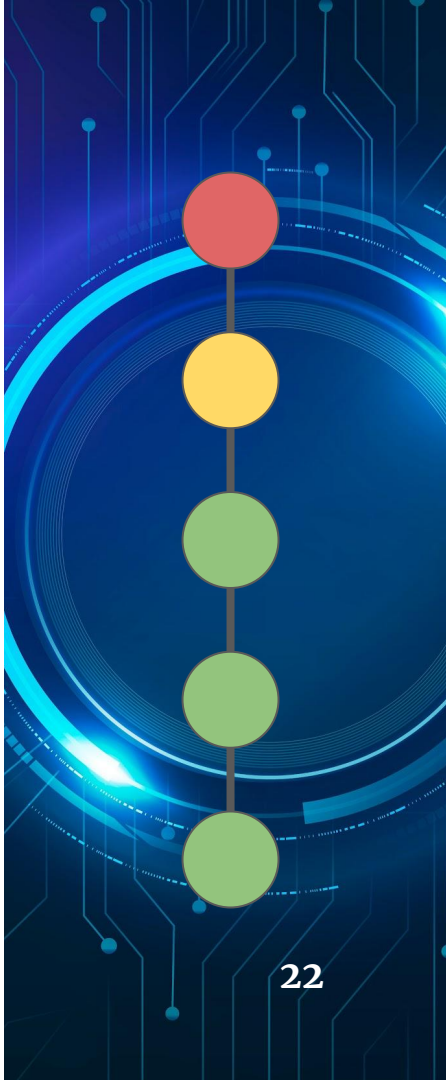


Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

n = 1001
q = 1
$q = 1 * 2^1 = 2$
n = 100
s = 3
i = 2

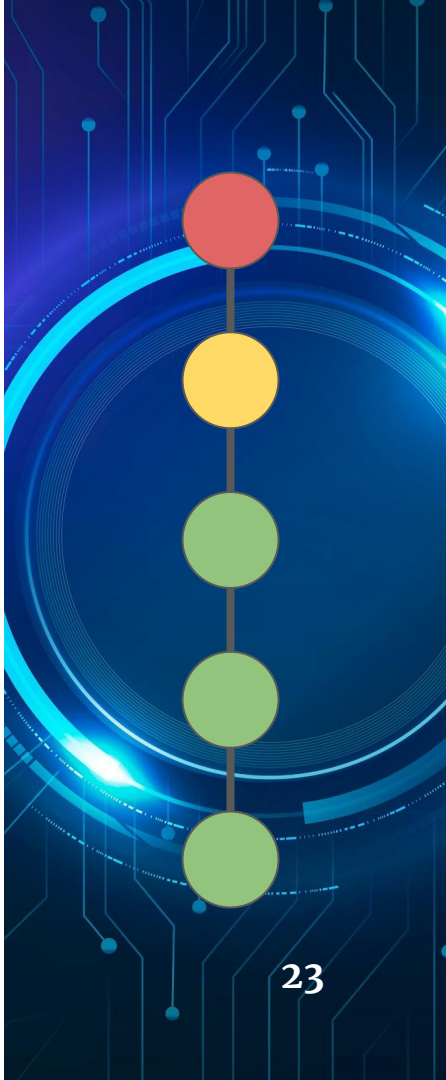


Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

n = 100
q = 0
$q = 0 * 2^2 = 0$
n = 10
s = 3
i = 3

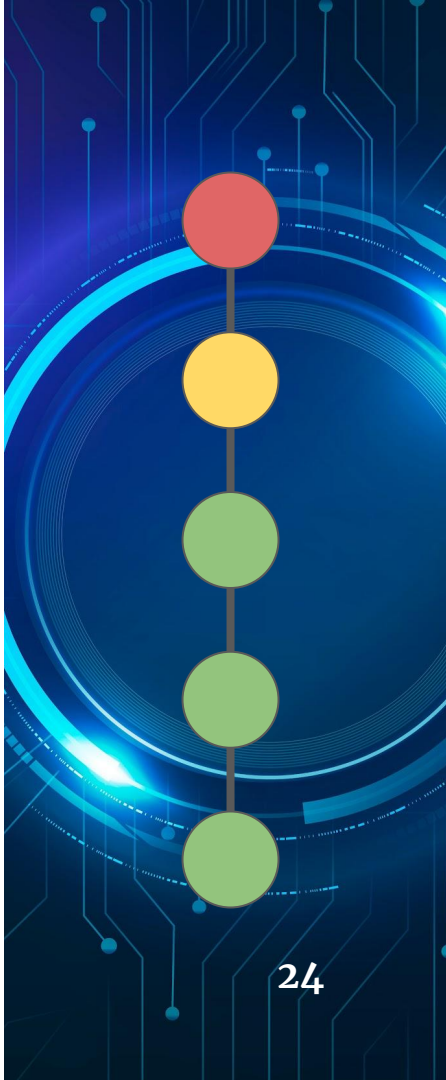


Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

n = 10
q = 0
$q = 0 * 2^3 = 0$
n = 1
s = 3
i = 4

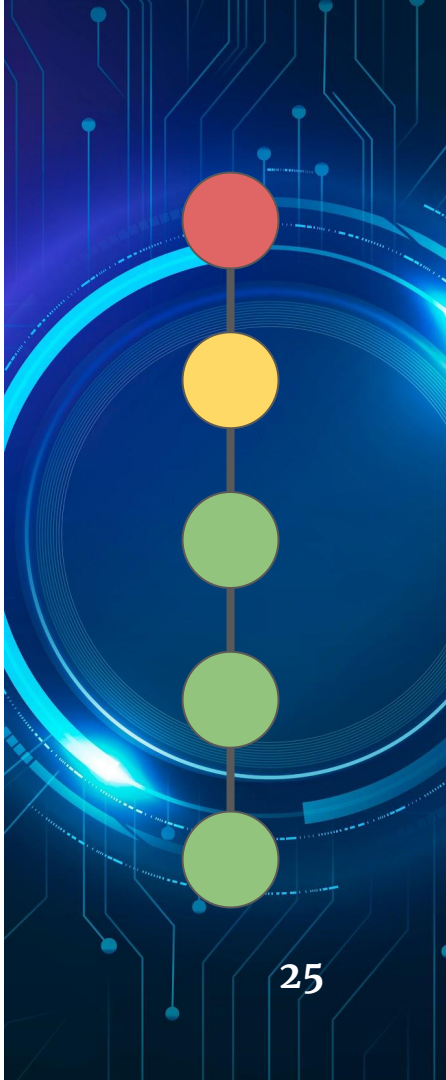


Number Conversion

- A number of base 2 (Binary) to Decimal

```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

n = 1
q = 1
$q = 1 * 2^4 = 16$
n =
s = 19
i = 5

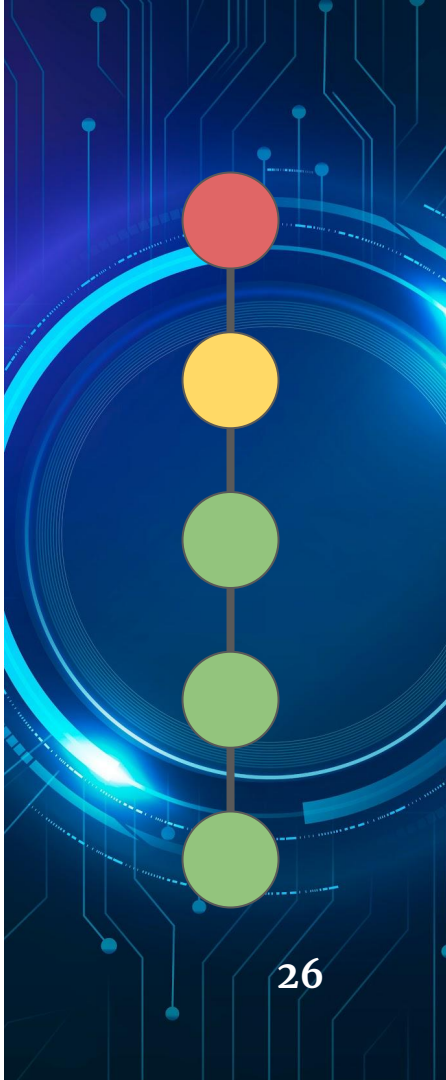


Number Conversion

- A number of base 2 (Binary) to Decimal

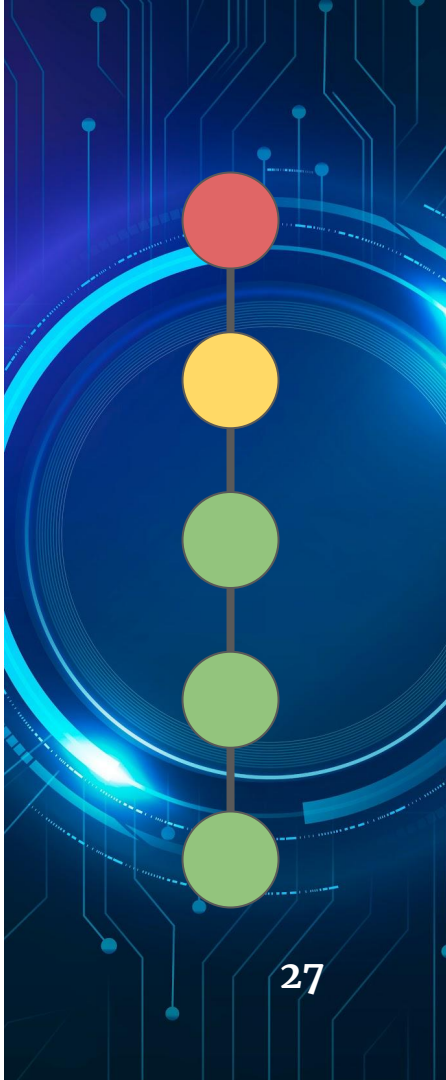
```
n ← 10011
b ← 2
s ← 0
i ← 0
While n is not empty
        q ← last digit of n
        q ← correspondingSymbol(q)*b^i
        n ← remove last digit from n
        s ← s + q
        i += 1
print → s
```

s = 19



Number Conversion

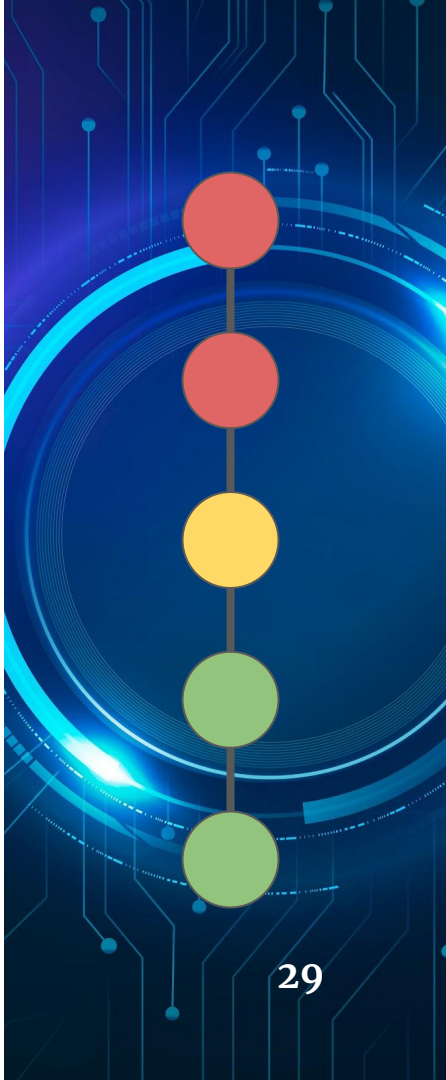
- Binary to Hexadecimal and vice-versa
  - Remember that $2^4 = 16$
  - 2 is the base of binary number
  - 16 is the base of hexadecimal number
  - 4 binary digits help equate 1 digit of hexadecimal
  - 1 hexadecimal digit help equate 4 binary digits



Representation of Numbers

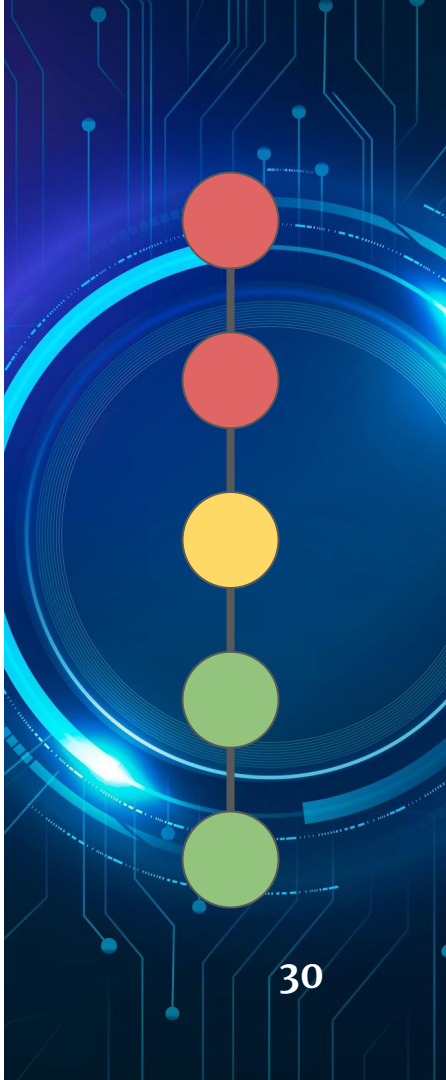
Representation of Numbers

- Dependent on words
- Words are a number of bits
- Word size varies from machine to machine
- Computers process using binary
- Humans easily understand decimal
- Hexadecimal expresses more in less digits



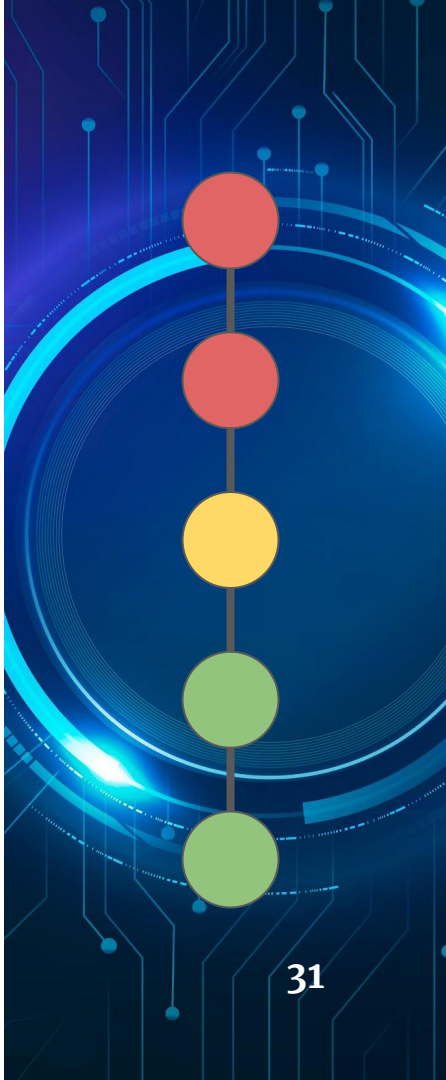
Representation of Numbers

- Integer Representation
  - For word size n, the maximum possible number is $2^n - 1$
  - Negative numbers are stored as 2's complements
    - The use of sign bit makes the highest possible number $2^{n-1} - 1$
    - First toggle all the bits of the number
    - Add 1 to the whole number
    - Use the sign bit as it is



Representation of Numbers

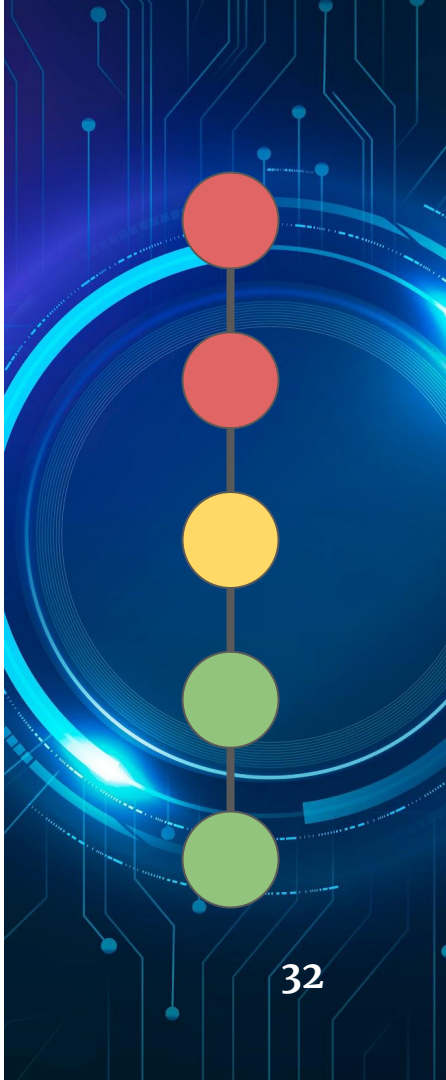
- Floating Point Representation
  - Exponential Form: $x = f \times 10^{-E}$
  - f is mantissa and E is exponent
  - The entire memory location is divided into three parts
  - Typically, in a 32 word environment
    - 24 bits for mantissa
    - 7 bits for exponent
    - 1 bit for sign



Representation of Numbers

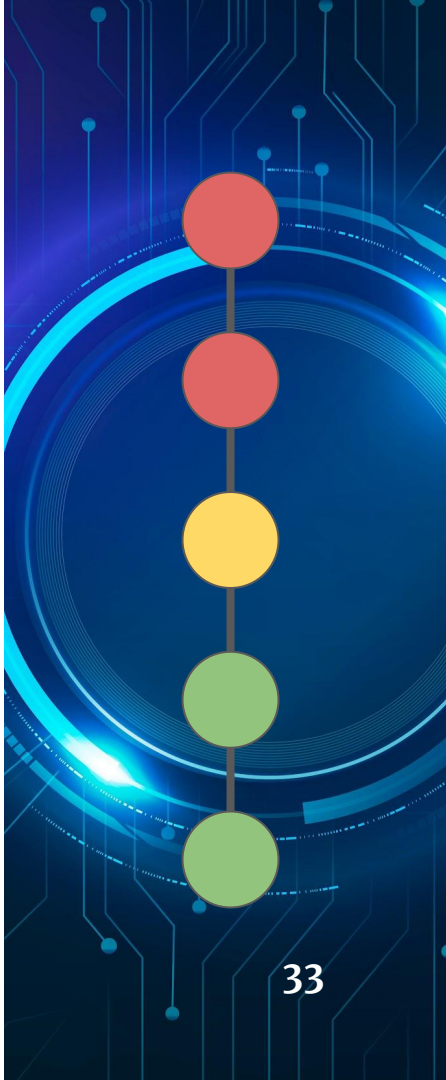
- Floating Point Representation

SIGN	Exponent	Mantissa
1 bit	7 bits	24 bits



Representation of Numbers

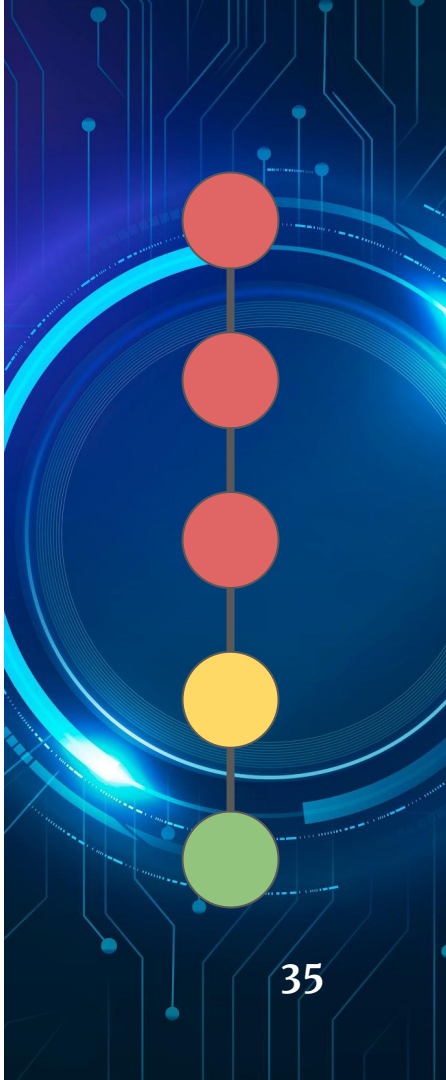
- Floating Point Representation
  - Shifting of decimal points ⇨ Normalization
  - Numbers in normalized form ⇨ Normalised Floating Point Numbers
  - Can be expressed as both 0.593×10^{-9} and .593E-9
  - Conditions for mantissa
    - For positive numbers: $0.1 \leq f < 1.0$
    - For negative numbers: $-0.1 \geq f > -1.0$
    - In general: $0.1 \leq |f| < 1.0$



Computer Arithmetic

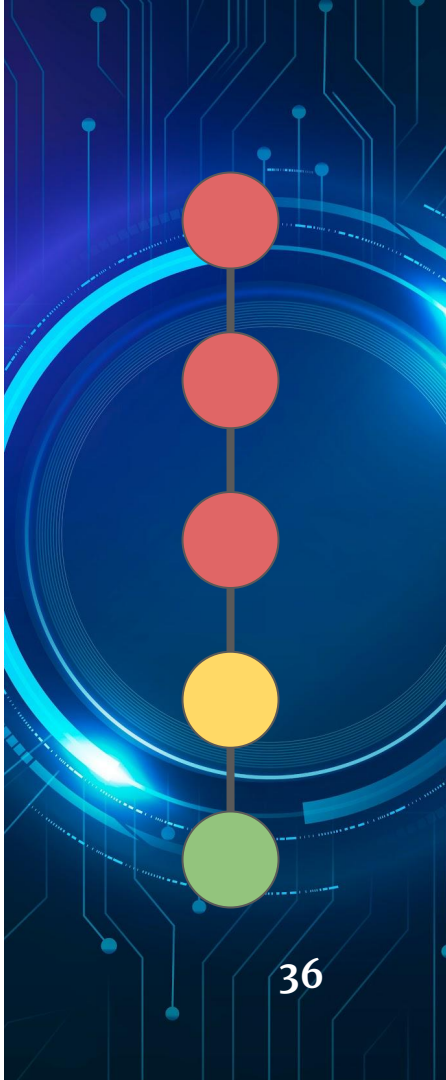
Computer Arithmetic

- Integer Arithmetic
  - Always result in integer
  - Be careful about division operation
  - For integer operation some formula can be false
    - $\frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$



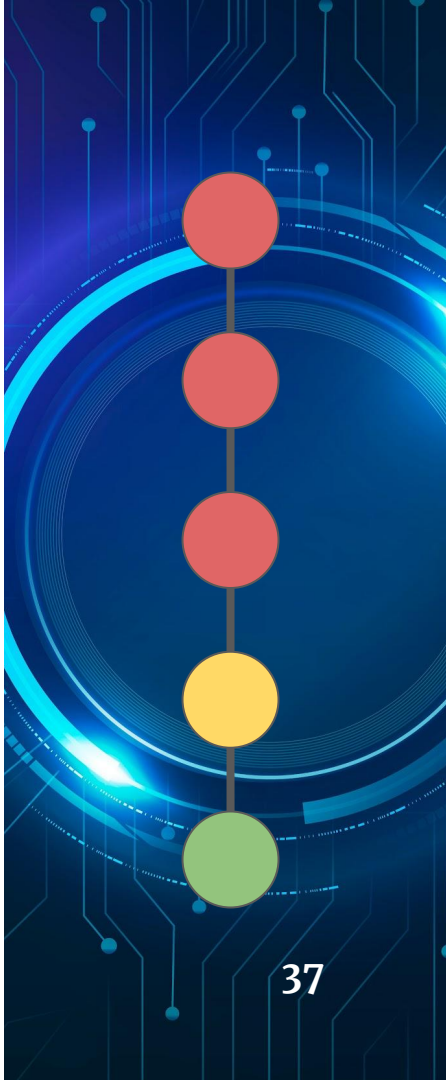
Computer Arithmetic

- Floating Point Arithmetic: Addition
  - Let x and y be added to result in z;
  - The fractional parts: f_x, f_y, f_z
  - The exponent parts: E_x, E_y, E_z;
  - Algorithm:
    - Find the larger number
    - Make the exponents of the number equal
    - Add the mantissa and normalize if necessary



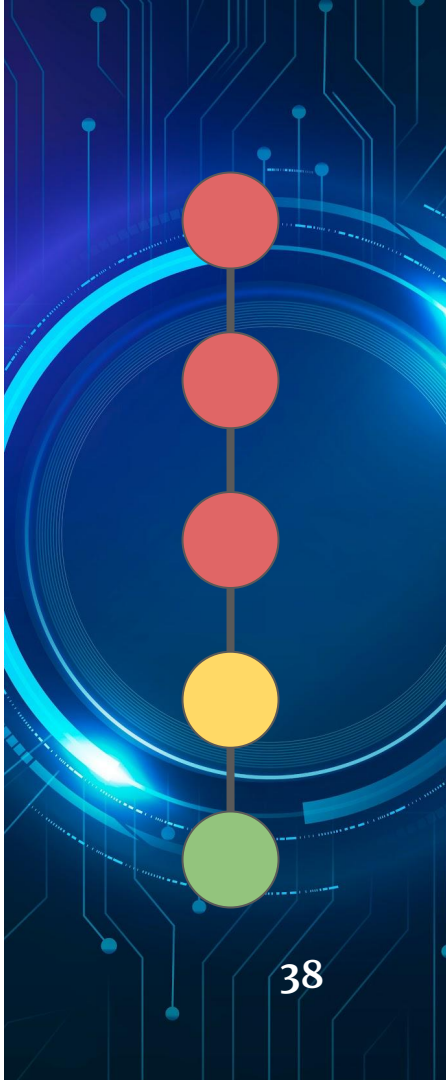
Computer Arithmetic

- Floating Point Arithmetic: Subtraction
  - Same as addition with different sign
  - Subtract the mantissa
  - Normalize if necessary



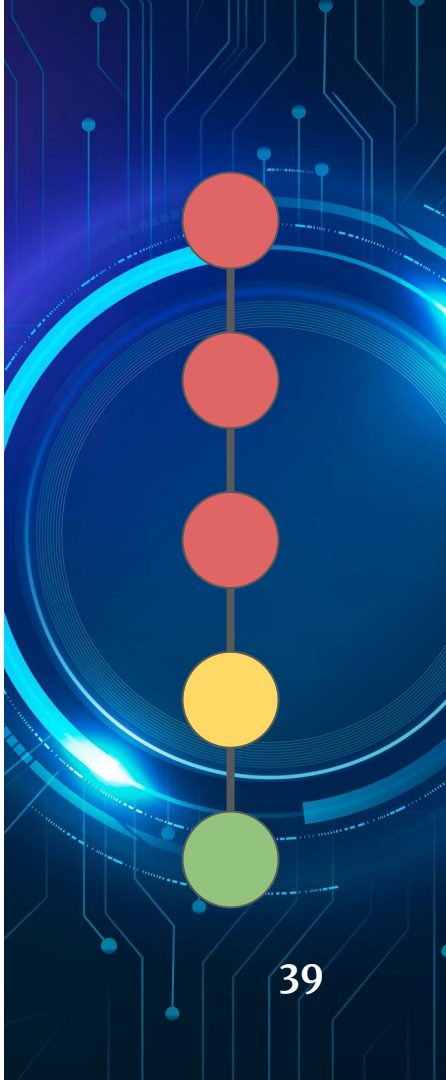
Computer Arithmetic

- Floating Point Arithmetic: Multiplication
  - Multiply mantissa
  - Add exponents
  - Normalize if necessary



Computer Arithmetic

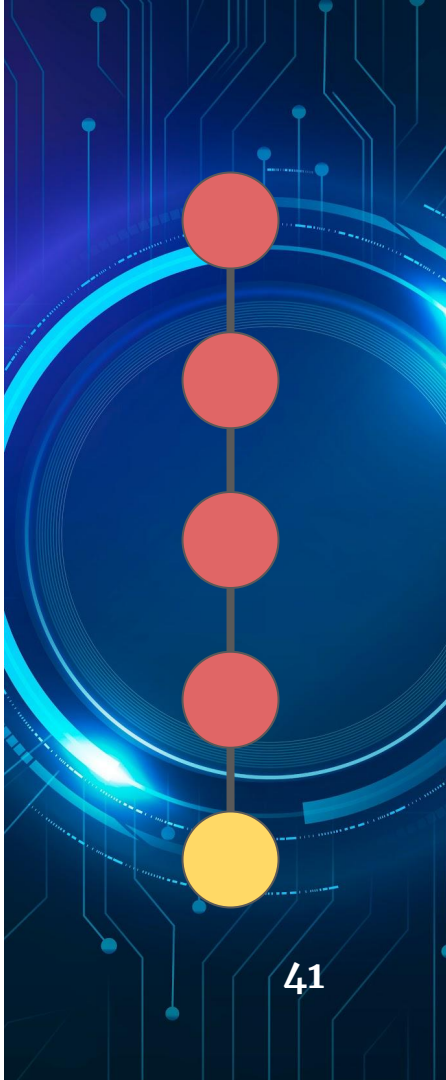
- Floating Point Arithmetic: Division
  - Divide mantissa
  - Subtract exponents
  - Normalize if necessary



Errors in Arithmetic

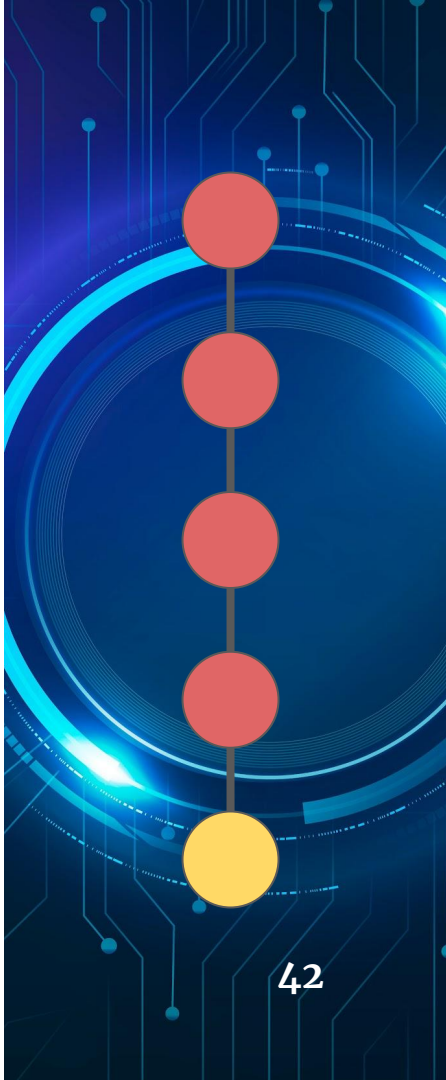
Errors in Arithmetic

- Points after decimal may induce errors
- Memory constraints induce underflow and overflow problem
- Sometimes violation of law of arithmetic



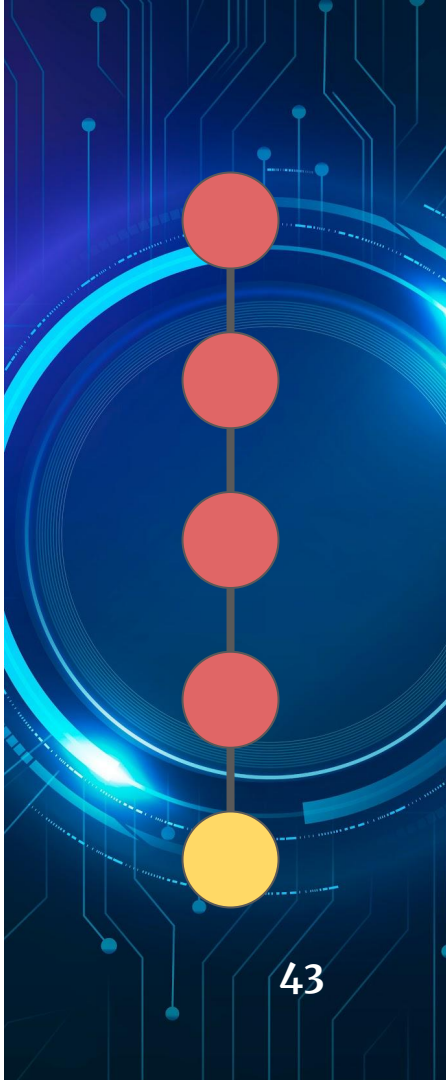
Errors in Arithmetic

- Error
  - $e = x_t - x_a$
- Absolute Error
  - $e_a = |x_t - x_a|$
- Relative Error
  - $e_r = \frac{|x_t - x_a|}{|x_t|}$



Errors in Arithmetic

- Error Propagation
- Conditioning and Stability
  - Condition Number = $\frac{\text{Relative Error in } f(x)}{\text{Relative Error in } x}$
  - Relative Error in f(x), $e_{r,f} = \frac{\Delta f}{f(x)} = \frac{\frac{\Delta f}{\Delta x}\Delta x}{f(x)} = \frac{f'(x)\Delta x}{f(x)}$
  - Relative Error in x, $e_{r,x} = \frac{\Delta x}{x}$
  - Condition Number = $\frac{x f'(x)}{f(x)}$
  - If condition number is large, it is called ill-conditioned



Thank You