

# Base Conversion

Between Powers of 2: 2, 8, 16

Examples: [Rechunk](#)

1. Convert each digit to binary: [Lookup Table](#) or from memory!
2. Merge then rechunk the bits
3. Convert each chunk to the appropriate digit

Base  $N$  to 10

- Use Expanded Notation, or
- For each digit: multiply by  $N$ , and then add the value of the digit

Base 10 to Base  $N$

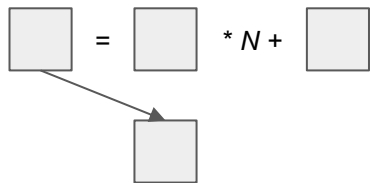
- Successively divide the number by  $N$
- The concatenation of the remainders produce the final value
- Consider the examples via the spreadsheet: [Base Conversion](#)

# Base $N$ to Base 10

Algorithm: multiply, add, and shift

- set  $v = 0$
- For each digit (from left to right)
  - $v = v * \text{base}$ ; # Multiple by the base
  - $v = v + \text{digit}_{10}$ ; # Add the next digit
- emit  $v$

Consider:  $16\# 5a2 == 1442$



$16\# 5a2$

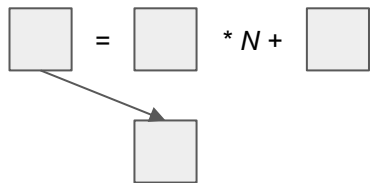
| v    |   | v  | base   | digit |   |
|------|---|----|--------|-------|---|
| 5    | = | 0  | * 16 + | 5     | 5 |
| 90   | = | 5  | * 16 + | 10    | a |
| 1442 | = | 90 | * 16 + | 2     | 2 |
|      |   |    |        |       |   |
|      |   |    |        |       |   |
|      |   |    |        |       |   |

# Base $N$ to Base 10

Algorithm: multiply, add, and shift

- set  $v = 0$
- For each digit (from left to right)
  - $v = v * \text{base}$ ; # Multiple by the base
  - $v = v + \text{digit}_{10}$ ; # Add the next digit
- print  $v$

Consider:  $2\# 10110101 == 181$



$2\# 10110101$

| v   |   | v  | base  | digit |   |
|-----|---|----|-------|-------|---|
| 1   | = | 0  | * 2 + | 1     | 1 |
| 2   | = | 1  | * 2 + | 0     | 0 |
| 5   | = | 2  | * 2 + | 1     | 1 |
| 11  | = | 5  | * 2 + | 1     | 1 |
| 22  | = | 11 | * 2 + | 0     | 0 |
| 45  | = | 22 | * 2 + | 1     | 1 |
| 90  | = | 45 | * 2 + | 0     | 0 |
| 181 | = | 90 | * 2 + | 1     | 1 |

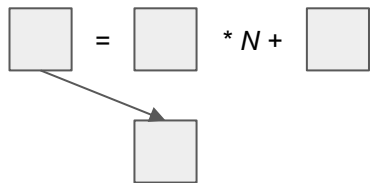
# Base $N$ to Base 10

8# 453

## Algorithm: multiply, add, and shift

- set  $v = 0$
- For each digit (from left to right)
  - $v = v * \text{base};$  # Multiple by the base
  - $v = v + \text{digit}_{10};$  # Add the next digit
- print  $v$

Consider:  $8\# == ?$



| v   |   | v  | base  | digit |   |
|-----|---|----|-------|-------|---|
| 4   | = | 0  | * 8 + | 4     | 4 |
| 37  | = | 4  | * 8 + | 5     | 5 |
| 299 | = | 37 | * 8 + | 3     | 3 |
|     |   |    |       |       |   |
|     |   |    |       |       |   |
|     |   |    |       |       |   |
|     |   |    |       |       |   |
|     |   |    |       |       |   |

# Base Conversion of Real Numbers

## Base 10 to Base $N$

- The whole portion is divided by the new base, repeatedly
  - $\text{Dividend} / \text{Divisor} = (\text{Quotient}, \text{Remainder})$
  - The concatenation of the Remainders provide you with the final digits
- The fraction portion is multiplied by the new base, repeatedly
  - $\text{Multiplier} * \text{Multiplicand} = (\text{Overflow}, \text{Product})$
  - The concatenation of the Overflows provide you with the final digits
- Consider the examples via the spreadsheet: [Base Conversion](#)

# Decimal Real to Binary Real

1. Split the number at the radix point: *whole . fractional*

2. With the whole part,

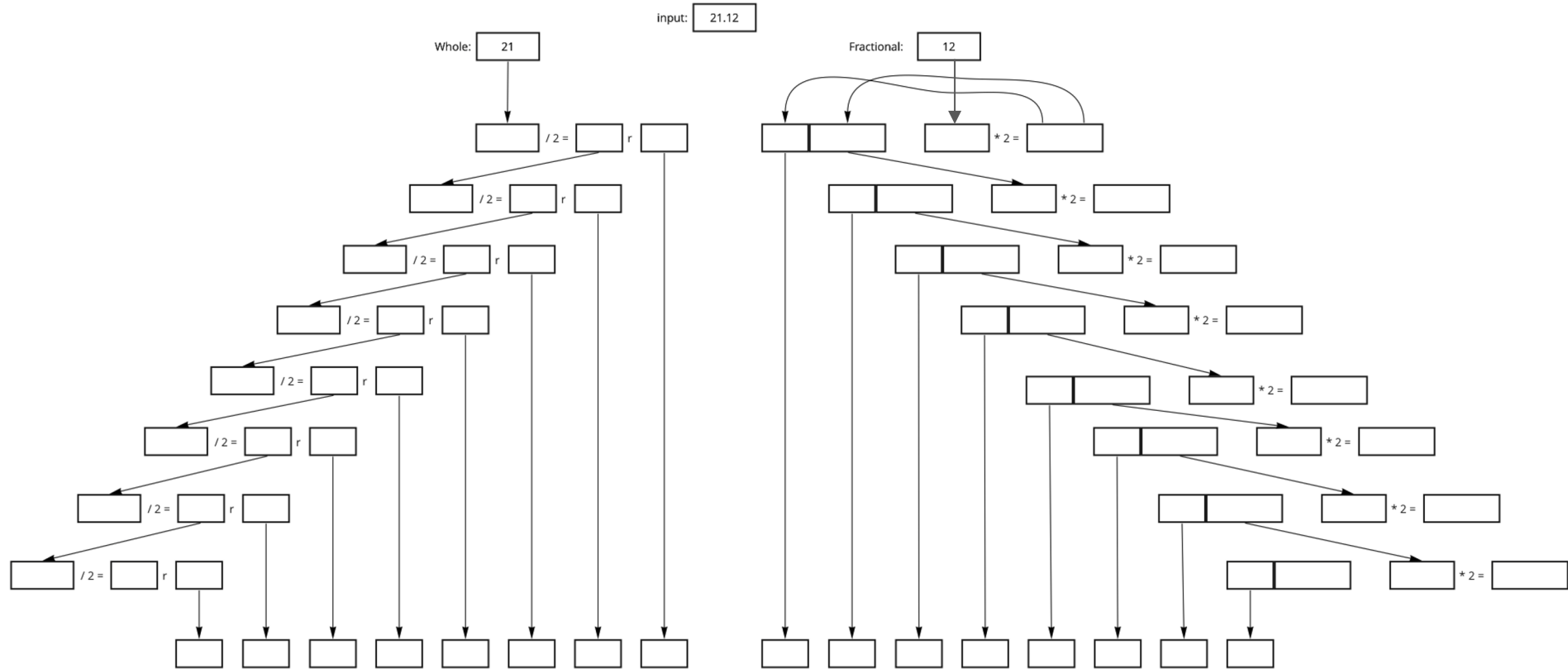
```
number = whole;
while (number != 0 ){
    push ( number % 2 )
    number = number / 2
}
pop_all();
```

3. With the fractional part

```
max = 10 ** stringlength(fractional);
number = fractional
while (number != 0 ) {
    number = number * 2
    if ( number > max ) {
        emit 1
        number = number - max
    } else {
        emit 0
    }
}
```

4. Put the two pieces together

## Real: Decimal to Binary



# Example: - 39.234

## Whole Part

1. Convert the base 10 number into base 2
2. Split the number at the radix point: - 39 . *fractional*

2. With the whole part,

```
number = whole
while (number != 0 ){
    push ( number % 2 )
    number = number / 2
}
pop_all();
```

```
number: 39
number = 39 / 2    → 19, 1
push ( 39 % 2 )    → 1

number = 19 / 2    → 9, 1
push( 19 % 2 )     → 1

number = 9 / 2     → 4, 1
push( 9 % 2 )      → 1

number = 4 / 2     → 2, 0
push( 4 % 2 )      → 0

number = 2 / 2     → 1, 0
push ( 2 % 2 )     → 0

number = 1 / 2     → 0, 1
push ( 1 % 2 )     → 1
```

4. Put the two pieces together

|   |        |   |
|---|--------|---|
| - | 100111 | . |
|---|--------|---|

|   |
|---|
| 1 |
| 0 |
| 0 |
| 1 |
| 1 |
| 1 |



# Example: - 39.234

## Fractional Part

1. Convert the base 10 number into base 2
2. Split the number at the radix point: *whole . 234*

$\text{max} = 10^{**} |234| == 1,000$

$\text{number} = 234$

$\text{number} = 234 * 2 = 0,468$

$\text{number} = 468 * 2 = 0,936$

$\text{number} = 936 * 2 = 1,872 - 1000 = 872$

$\text{number} = 872 * 2 = 1,744 - 1000 = 744$

$\text{number} = 744 * 2 = 1,488 - 1000 = 488$

$\text{number} = 488 * 2 = 0,976$

$\text{number} = 976 * 2 = 1,952 - 1000 = 952$

3. With the fractional part

```
max = 10 ** strlen(fractional);
number = fractional
while (number != 0) {
    ← number = number * 2
    if ( number > max ) {
        emit 1
        number = number - max
    } else {
        emit 0
    }
}
```

4. Put the two pieces together

|   |         |
|---|---------|
| . | 0011101 |
|---|---------|

# Example: - 39.234

## Both Parts

1. Split the number at the radix point: *whole . fractional*

2. With the whole part,

```
number = whole
while (number != 0 ){
    push ( number % 2 )
    number = number / 2
}
pop_all();
```

3. With the fractional part

```
max = 10 ** stringlength(fractional);
number = fractional
while (number != 0 ) {
    number = number * 2
    if ( number > max ) {
        emit 1
        number = number - max
    } else {
        emit 0
    }
}
```

4. Put the two pieces together

|   |        |   |         |
|---|--------|---|---------|
| - | 100111 | . | 0011101 |
|---|--------|---|---------|

# Example: 45.45

## Whole Part

1. Split the number at the radix point: *whole . fractional*

2. With the whole part,

```
number = whole;
while (number != 0 ){
    push ( number % 2 )
    number = number / 2
}
pop_all();
```

```
number: 45
number = 45 / 2    → 22, 1
push ( 1 )
number = 22 / 2    → 11, 0
push ( 0 )
number = 11 / 2    → 5, 1
push ( 1 )
number = 5 / 2     → 2, 1
push ( 1 )
number = 2 / 2     → 1, 0
push ( 1 )
number = 1 / 2     → 0, 1
push ( 1 )
```

---

number = 0 / 0

1  
0  
1  
1  
0  
1

4. Put the two pieces together

|        |   |
|--------|---|
| 101101 | . |
|--------|---|

# Example: 45.45

## Fractional Part

1. Split the number at the radix point: *whole* . 45

```
max = 10 ** |45| == 100
number = 45
  number = number * 2 = 90
  number = number * 2 = 180 - 100 = 80
  number = 80 * 2 = 160 - 100 = 60
  number = 60 * 2 = 120 - 100 = 20
  number = 20 * 2 = 40
  number = 40 * 2 = 80
  number = 80 * 2 = 160 = 100 = 60
```

3. With the fractional part

```
max = 10 ** strlen(fractional);
number = fractional
while (number != 0 ) {
  number = number * 2
  if ( number > max ) {
    emit 1
    number = number - max
  } else {
    emit 0
  }
}
```

4. Put the two pieces together

|   |         |
|---|---------|
| . | 0111001 |
|---|---------|

# Example: 45.45

1. Split the number at the radix point: *whole . fractional*

2. With the whole part,

```
number = whole;
while (number != 0 ) {
    push ( number % 2 )
    number = number / 2
}
pop_all();
```

3. With the fractional part

```
max = 10 ** stringlength(fractional);
number = fractional
while (number != 0 ) {
    number = number * 2
    if ( number > max ) {
        emit 1
        number = number - max
    } else {
        emit 0
    }
}
```

4. Put the two pieces together

|        |   |         |
|--------|---|---------|
| 101101 | . | 0111001 |
|--------|---|---------|