Definition: **BIT**

- **Binary Digit**
- Smallest possible unit of information
- Two values only: 0 or 1
- Represent a single Yes or No question
- Can encode any two-valued system — Yes/No, True/False, Up/Down, On/Off, In/Out, etc.
- Easy to build hardware to encode bits.

**Bits and Patterns**

- 1 Bit gives $2^1 = 2$ patterns: 0 or 1
- 2 Bits gives $2^2 = 4$ patterns: 00, 01, 10, 11
- 3 Bits gives $2^3 = 8$ patterns: 000, 001, 010, 011, 100, 101, 110, 111
- Each new bit doubles the number of patterns
- Therefore: $N$ Bits gives $2^N$ Distinct patterns.

**What About 8 Bits?**

- 00000000 = 0
- 00000001 = 1
- 00000010 = 2
- 00000011 = 3
- 00000100 = 4
- 00000101 = 5
- ... 
- 11111110 = 254
- 11111111 = 255

**Definition: Byte**

- Packet of 8 Bits (French word is “octet”)
- Typical unit of computer memory / storage
- Used to represent one standard character
- Values range from 00000000 ... 11111111
- $2^8 = 256$ Distinct patterns
- Can encode any integer between 0 and 255

**Unsigned Integers**

- Pick storage size of $N$ bits (8, 16, 32, 64, etc.)...
- ...therefore $2^N$ distinct patterns are available.
- Smallest value is all zeroes (decimal value 0),
- Largest value is therefore $2^N-1$.
- Results less than zero are “underflow” errors,
- Results greater than max are “overflow” errors
- Each computer architecture has a fixed $N$. 
Signed Integers

- Pick N, there are still $2^N$ patterns.
- Consider half the patterns to be negative.
  - Half of $2^N = 2^{N/2} = 2^{N-1}$.
- Remaining patterns are zero and above.
- Signed range is therefore $-2^{N-1} \ldots +2^{N-1} - 1$.
  - Zero is considered positive.

Example for N=8

- $2^8 = 256$ patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^8 - 1 = 255$
- Signed Range
  - Minimum: $-2^{8-1} = -2^7 = -128$
  - Maximum: $+2^{8-1} = +2^7 = +128 = +127$

Example for N=16

- $2^{16} = 65536$ patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^{16} - 1 = 65535$
- Signed Range
  - Minimum: $-2^{16-1} = -2^{15} = -32768$
  - Maximum: $+2^{16-1} = +2^{15} = +32768 - 1 = +32767$

Example for N=32

- $2^{32} = 4,294,967,296$ patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^{32} - 1 = 4,294,967,295$
- Signed Range
  - Minimum: $-2^{32-1} = -2^{31} = -2,147,483,648$
  - Maximum: $+2^{32-1} = +2^{31} = +2,147,483,647$
  - Nine (and a little more) significant digits

What about Real numbers?

- Approaches:
  - Rational
  - Fixed-Point
  - Floating-Point
- All require re-interpreting how bits are used.

Rational Numbers

- For N bits, divide into two $N/2$ bit sections:
  - First section is numerator
  - Second section is denominator
- Numbers like $1/3$, $1/7$, $3/7$, $1/10$, $355/113$ are easy
- Reduce to lowest form (e.g., $3/4$ goes to $1/2$)
- Many redundant patterns:
  - low information density
  - Not efficient use of bits
Fixed-Point Numbers

- Set virtual decimal point to middle of bits:
  - Half the bits are integer
  - Half the bits are fraction
- All bit patterns are useful
- Easy to add, subtract, multiply, divide in binary
- Trades off range of values for fraction support.
  - For N=16, max signed value is only
    \( +127.99609375 \)
- Still not an efficient use of bits

Floating-Point Numbers

- Binary version of Scientific Notation
  - Decimal: \( +3.4024\times10^{15} \)
  - Binary: \( +1.00101001\times2^{1001} \)
- Use one bit for sign (0=plus, 1=minus)
- Use some of the N bits for exponent
- Use remaining bits for mantissa (significand)
- Trades off precision for dynamic range

Floating-Point Precision

- Single Precision
  - N=32 bits (1 sign, 8 exponent, 23 mantissa)
  - Dynamic Range: \( \pm10^{38} \)
  - Significant Figures: 5-6 Decimal Digits
    - (Remember 32 bit integers have about 9 sig. figs.)
- Double Precision (Used by Excel)
  - N=64 bits (1 sign, 11 exponent, 52 mantissa)
  - Dynamic Range \( \pm10^{308} \)
  - Significant Figures: 15-16 Decimal Digits

But long fractions get rounded off:

- Expected loss of precision:
  - Numbers with naturally long but finite fractions,
  - Rationals that repeat forever (\( 1/3 = 0.33333333... \)),
  - Irrationals (e, \( \pi \), \( v_2 \), \( v_3 \), \( v_5 \), etc.).
- Unexpected loss of precision: Well-behaved decimal fractions that are ill-behaved in binary
  \( (1/10)_{10} = 0.0001101100110011... \)

Aside: Proof that \( v_2 \) is Irrational

- \( v_2 = 1.414213562... \)
- Remember:
  - Even \times Even = Even (4 \times 6 = 24)
  - Even \times Odd = Even (4 \times 7 = 28)
  - Odd \times Odd = Odd (5 \times 7 = 35)
- Assume \( v_2 \) is Rational: \( v_2 = \frac{p}{q} \)
- Assume Lowest Form: \( p, q \) aren’t both even
  - (if both were even, we can repeatedly divide both \( p \) and \( q \) by 2 until at least one is odd)

Aside: Proof that \( v_2 \) is Irrational

- Square both sides: \( 2 = \frac{p^2}{q^2} \)
- Multiply by \( q^2 \): \( 2q^2 = p^2 \)
- Conclusion #1: \( p^2 \) is even, thus \( p \) is even
- Divide by 2: \( Q^2 = \frac{p^2}{2} = P \times \frac{p}{2} \)
- Conclusion #2: \( Q^2 \) is even, thus \( Q \) is even
- Contradiction:
  - Initial assertion was \( p, q \) aren’t both even, proof says both are even, thus assumption that \( v_2 = \frac{p}{q} \) is false. No such rational number exists.
The Biggest Dirty Secret of Computing

- Most of the interesting numbers in the Universe are irrational.
- Numbers on computers have a fixed and finite number of bits.
- Therefore, most values get rounded off.
- Most numerical results are approximations.
- More bits means more precision, but only forestalls and does not eliminate the problem.

Complex Numbers

- The Real number line extends from \(-\infty\) to \(+\infty\),
- Use of space above and below the line gives us more computational expressive power.
- Negation then becomes a rotation of 180°:

![Complex Numbers](image1)

Complex Numbers

- Rotation of +1 by 90° leaves it in space above the zero center. Call that number \(i\):

![Complex Numbers](image2)

Complex Numbers

- Multiplying a number by \(i\) twice equals negation,
- Thus \(i^2 = -1\), and then \(i = \sqrt{-1}\)
- \(i\) is called “imaginary”

Complex Numbers

- A complex number is then a pair of numbers:
  - A value along the Real axis,
  - A value along the Imaginary axis.
  - Written with the Real part first, then Imaginary.
- Examples:
  - \(2+3i, 5-7i, -3+2i, -4-6i, 6.7+5.9i\), etc.
  - 7 (same as 7+0i)

Complex Math

- Add/Subtract: treat components separately:
  - \(2+6i + 5-2i = (2+5) + (6-2)i = 7+4i\)
  - \(2+6i - 5-2i = (2-5) + (6-(-2))i = -3+8i\)
- Multiplication uses FOIL method:
  - \(2+6i \times 5-2i = \)
  - \((2 \times 5) + (2 \times -2i) + (6i \times 5) + (6i \times -2i) = \)
  - \((10) + (-4i) + (30i) + (-12i^2) = \)
  - \((10 + 12) + (-4 + 30)i = 22+26i\)
Complex Math

- Division uses two complex multiplications to eliminate imaginary component in denominator,
- Multiply both numerator and denominator by complex conjugate of denominator.
- Example:
  - \( \frac{22 + 26i}{2 + 6i} = \)
  - Numerator: \( 22 + 26i \times 2 - 6i = 200 - 80i \)
  - Denominator: \( 2 + 6i \times 2 - 6i = 4 + 36 = 40 \)
  - \( 200 - 80i \div 40 = 5 - 2i \)

Complex Math

- Used in math, engineering, physics, etc.
- Supported by early language FORTRAN,
- Supported by modern language Python,
- Supported (badly) by Excel 2007 and later.
- Mostly Double-Precision Floats,
- Subject to same round-off errors as other floating-point numbers.