Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Representing Integers

Quick Recap of Relevant Topics

IIT Bombay

- Architecture of a simple computer
- Bits and bytes of information

Overview of This Lecture

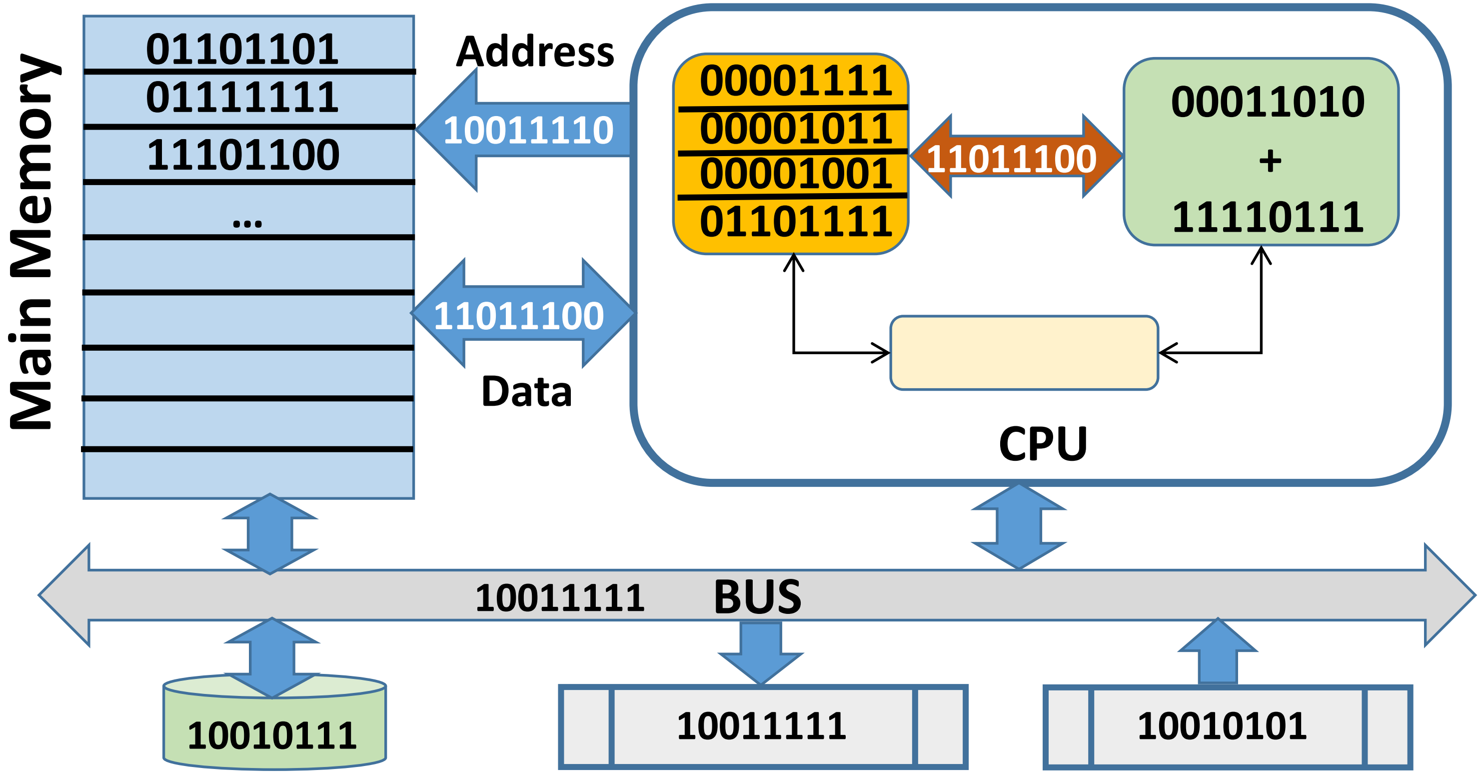
IIT Bombay

- Computer’s internal representation of numbers
  - Integers
- C++ declaration of integer variables

Recap from Earlier Lecture

IIT Bombay

- Snapshot:



- How do we represent integers like 56 or -37 in a computer?

Representing Integers

IIT Bombay

- Integers
  - Decimal representation: base 10, needs numerals 0 - 9

    $$233 = 2 \times 10^2 + 3 \times 10^1 + 3 \times 10^0$$

  - Binary representation: base 2, need numerals 0 – 1

    $$110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

- Any sequence of ‘0’s and ‘1’s can be thought of as representing an integer

Binary To Decimal

IIT Bombay

- What is 1011 in decimal?
  - Number of bits: 4; maximum power of 2 is (4-1) = 3
  - 1011 $= 1 \times 2^3 + \ldots$
  - 1011 $= 1 \times 2^3 + 0 \times 2^2 + \ldots$
  - 1011 $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + \ldots$
  - 1011 $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

Most Significant Bit (MSB) 1011 **Least Significant Bit (LSB)**

- MSB multiplied with highest power of 2, LSB with 2^0

Decimal To Binary

IIT Bombay

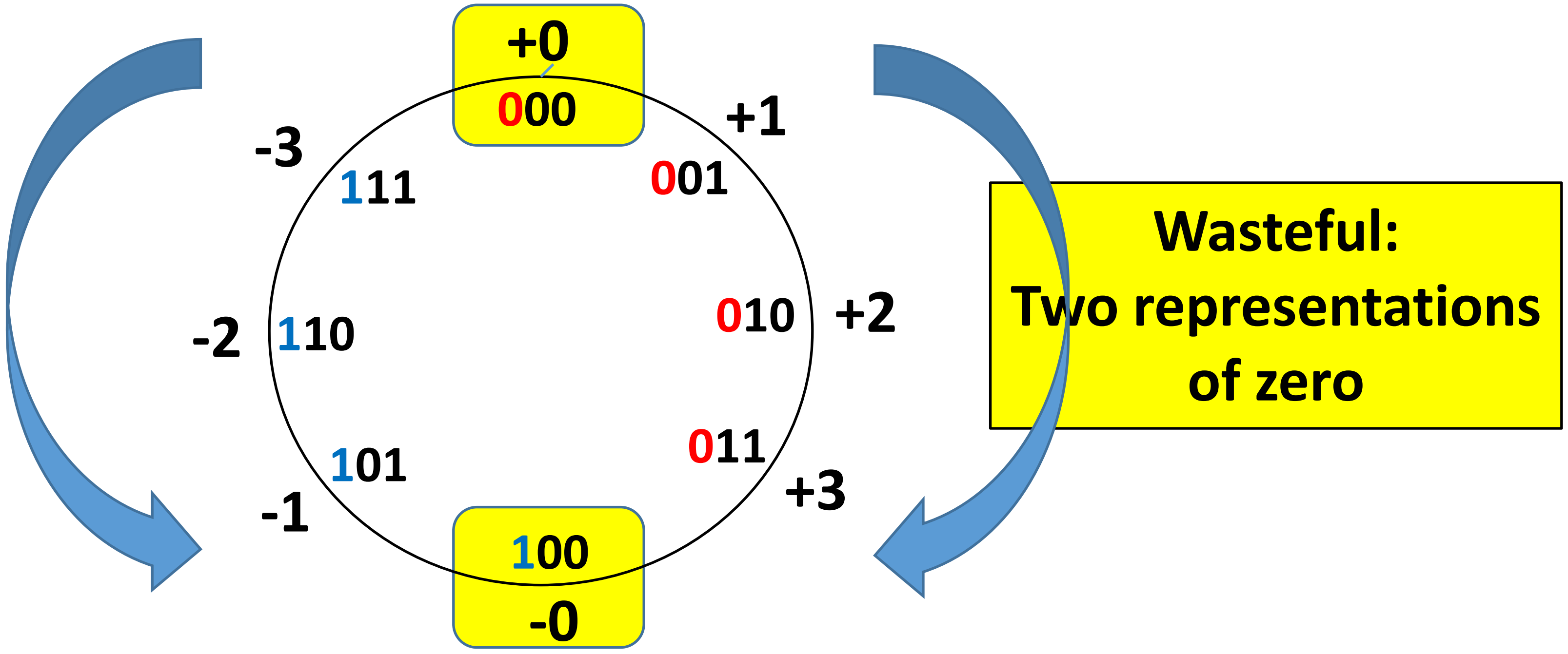
- What is 23 (written in decimal) in binary ?
  - 23/2: Quotient = 11, Remainder = 1 (Least Significant Bit)
    $23 = 11 \times 2 + 1 \times 2^0$
  - 11/2: Quotient = 5, Remainder = 1
    $23 = (5 \times 2 + 1) \times 2 + 1 \times 2^0$
    $= 5 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  - 5/2 : Quotient = 2, Remainder = 1
    $23 = (2 \times 2 + 1) \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
    $= 2 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  - 2/2 : Quotient = 1, Remainder = 0
    $23 = (1 \times 2 + 0) \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
    $= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  - 1/2 : **Quotient = 0**, STOP condition Remainder = 1 (Most Significant Bit)

Answer:
10111

Representing Signed Integers

IIT Bombay

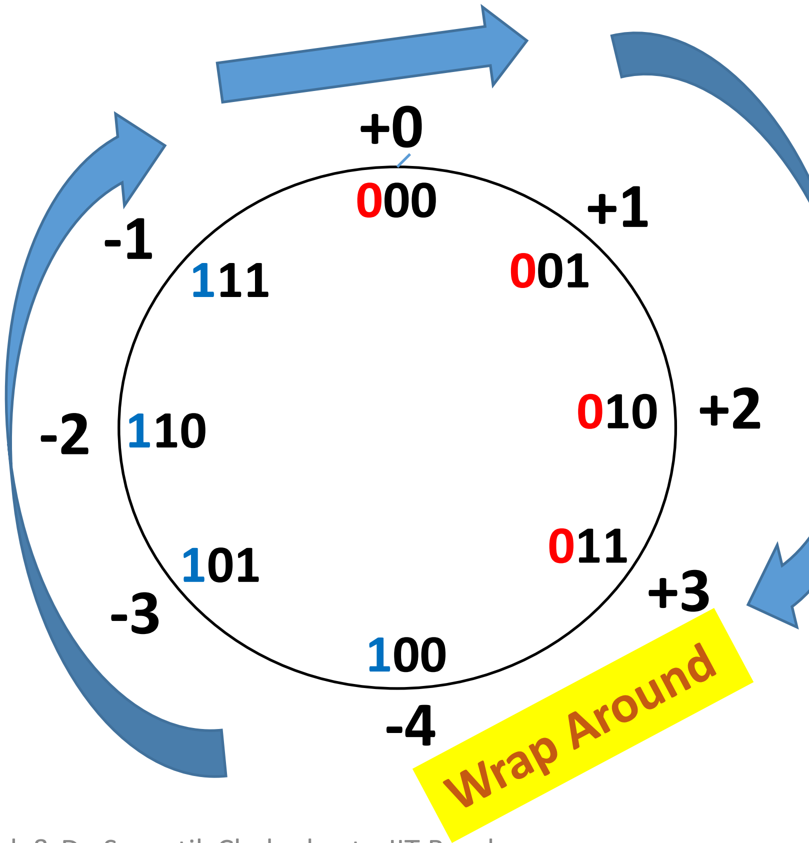
- How do we represent signed integers? -1, -23, …
- Treat MSB as sign bit: negative if MSB is 1, positive if MSB is 0
  - Consider integers represented using 3 bits



Representing Signed Integers

IIT Bombay

- Better representation: two's complement
  - MSB still represents sign



8 numbers represented:
-4 through +3

Only one representation of 0

Two's Complement Representation

IIT Bombay

- Is there an easy way to figure out what 10111 represents in 2's complement?
  - 10111 has MSB 1: Negative integer
  - To get absolute value of 10111
    - Ignore MSB: 10111
    - Flip every bit in 0111: 1000 (decimal 8)
    - Add 1: 1001 (decimal 9)
    - Absolute value is 9
  - Answer: -9

Integers in C++

IIT Bombay

- int data type
- Different kinds of integers allowed in C++
  - short int , long int, long long int, unsigned long int…
  - Number of bytes used to store value
    - short: 2 bytes, standard integer: 4 bytes, long: 4-8 bytes, long long: 8 bytes
  - Signed (2's complement) unless specified explicitly as unsigned
- Maximum, minimum values depend on number of bytes and signed/unsigned interpretation
  - standard integer (signed, 4 bytes): -2^{31} through $+(2^{31} - 1)$
  - unsigned long long (8 bytes): 0 through $2^{64} - 1$
- C++ declarations: int myMarks; unsigned short int numStudents;

Integers in C++

IIT Bombay

- Integer constants can be specified in
  - Decimal (base/radix 10): optional sign followed by digits from 0-9
    - 2, +21, -3097, 0
    - Must begin with non-zero except when value is 0
  - Octal (base/radix 8): ‘0’ (zero) followed by digits from 0-7
    - 0372 is $3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 = 250$ in decimal, 011111010 in binary
  - Hexadecimal (base/radix 16): ‘0x’ (zero x) followed by 0-9 and a-f
    - a = 10, b = 11, c = 12, d = 13, e = 14, f = 15
    - 0x4cf3 is $4 \times 16^3 + 12 \times 16^2 + 15 \times 16^1 + 3 \times 16^0 = 19699$ in decimal, 0100110011110011 in binary
- C++ integer constant declaration:
  - const int scaleFactor = 0x1f;
  - Value of scaleFactor cannot be changed during program execution

Summary

IIT Bombay

- Binary representation of integers
  - Conversion to and from decimal
  - Two's complement representation
  - C++ declarations

IIT Bombay

Computer Programing

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Representing Floating Point Numbers

Quick Recap of Relevant Topics

- Architecture of a simple computer
- Bits and bytes of information

Overview of This Lecture

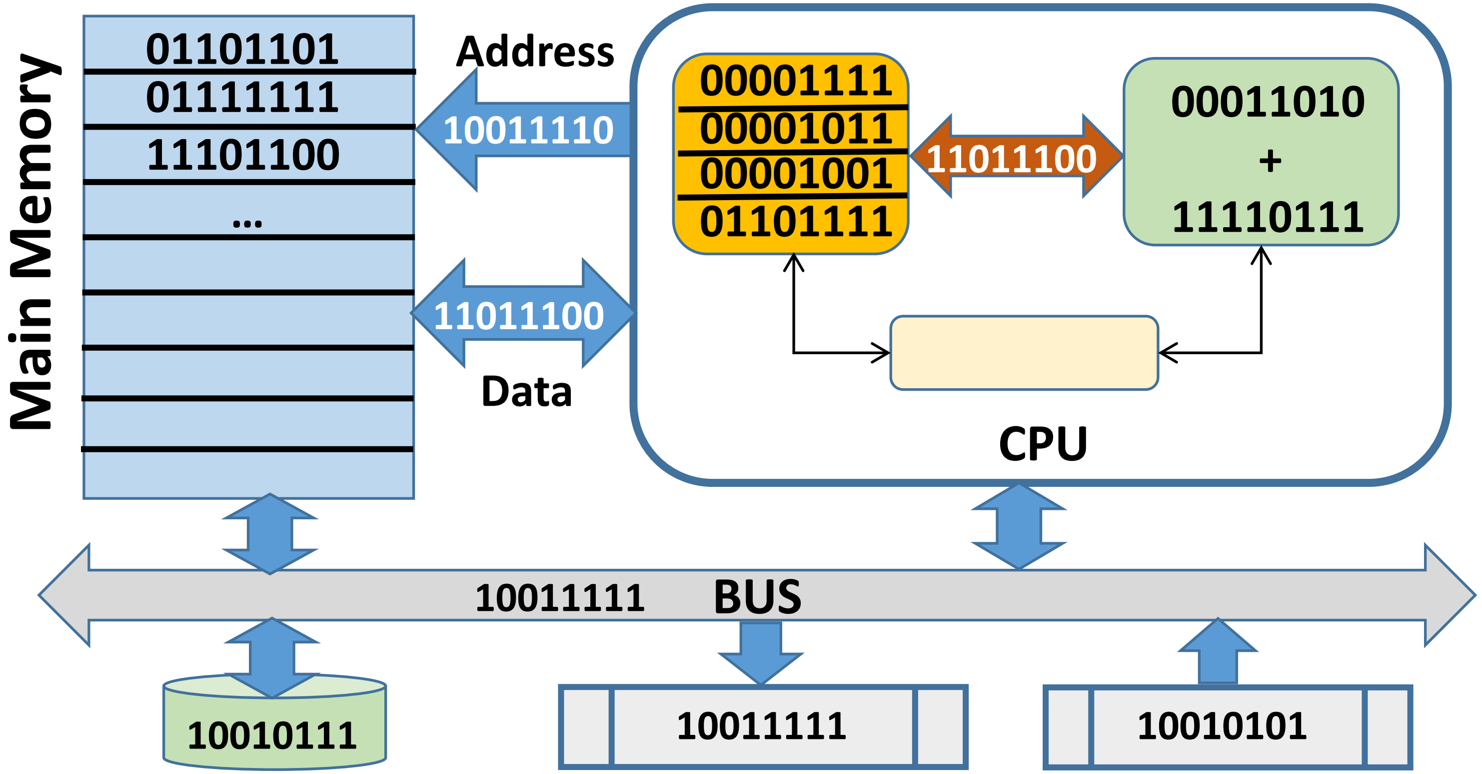
IIT Bombay

- A computer's internal representation of integers
- C++ declarations of integer variables

Recap from Earlier Lecture

IIT Bombay

- Snapshot:



- How do we represent integers like 56 or -37 in a computer?

Representing Floating Point Numbers

IIT Bombay

- Numbers with fractional values, very small or very large numbers cannot be represented as integers
- Floating point number
  - Decimal: $- \underbrace{3.123}_{\text{Mantissa}} \times 10^{-11}$

    Sign

    Exponent

    Base/Radix
    - Mantissa = $-(1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3})$
  - Binary: -1.1101×2^{110}
    - Mantissa = $-(1 \times 2^{0} + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}) = -1.8125$
    - Exponent = $(1 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0}) = 6$

Representing Floating Point Numbers

IIT Bombay

- Normalized mantissa: single non-0 digit to left of radix point
  - $0.02345 \times 10^{12} = 2.345 \times 10^{10}$
  - $110.101 \times 2^{110} = 1.10101 \times 2^{1000}$
  - Binary: Implicit 1 always on left of radix point; need not be stored
- Floating point numbers represented by allocating fixed number of bits for mantissa and exponent
  - Cannot represent all real numbers
  - Finite precision artifacts
    - What is $0.101 \times 2^{111} + 1$ if we have only 3 bits to represent mantissa?

Floating Point Numbers in C++

IIT Bombay

- float and double data types
- float
  - 32 bits (4 bytes): 1 sign, 8 exponent, 23 mantissa
  - Approximate range of magnitude: $10^{-44.85}$ to $10^{34.83}$
- double
  - 64 bits (8 bytes): 1 sign, 11 exponent, 52 mantissa
  - Approximate range of magnitude: $10^{-323.3}$ to $10^{308.3}$
- Special bit patterns reserved for 0, infinity, NaN (not-a-number: result of 0/0), …
- C++ declarations: float temperature; double verticalSpeed;

Floating Point Numbers in C++

IIT Bombay

- Floating point constants can be specified in C++ programs as
  - 23.572 (can have non-normalized mantissa in programs)
  - 2357.2e-2 or 2357.2E-2
    - 2357.2×10^{-2} (base/radix is 10)
- C++ constant floating point declaration
  - const float pi = 3.1415
  - const double e = 2.7183
  - Values of pi and e cannot change during program execution

Summary

IIT Bombay

- Binary representation of integers
  - Conversion to and from decimal
  - Two’s complement representation
  - C++ declarations
- Binary representation of floating point numbers
  - Sign, mantissa and exponent
  - C++ declarations