MATH50003 Numerical Analysis

II.1 Integers

Dr Sheehan Olver

Part II: Representing Numbers

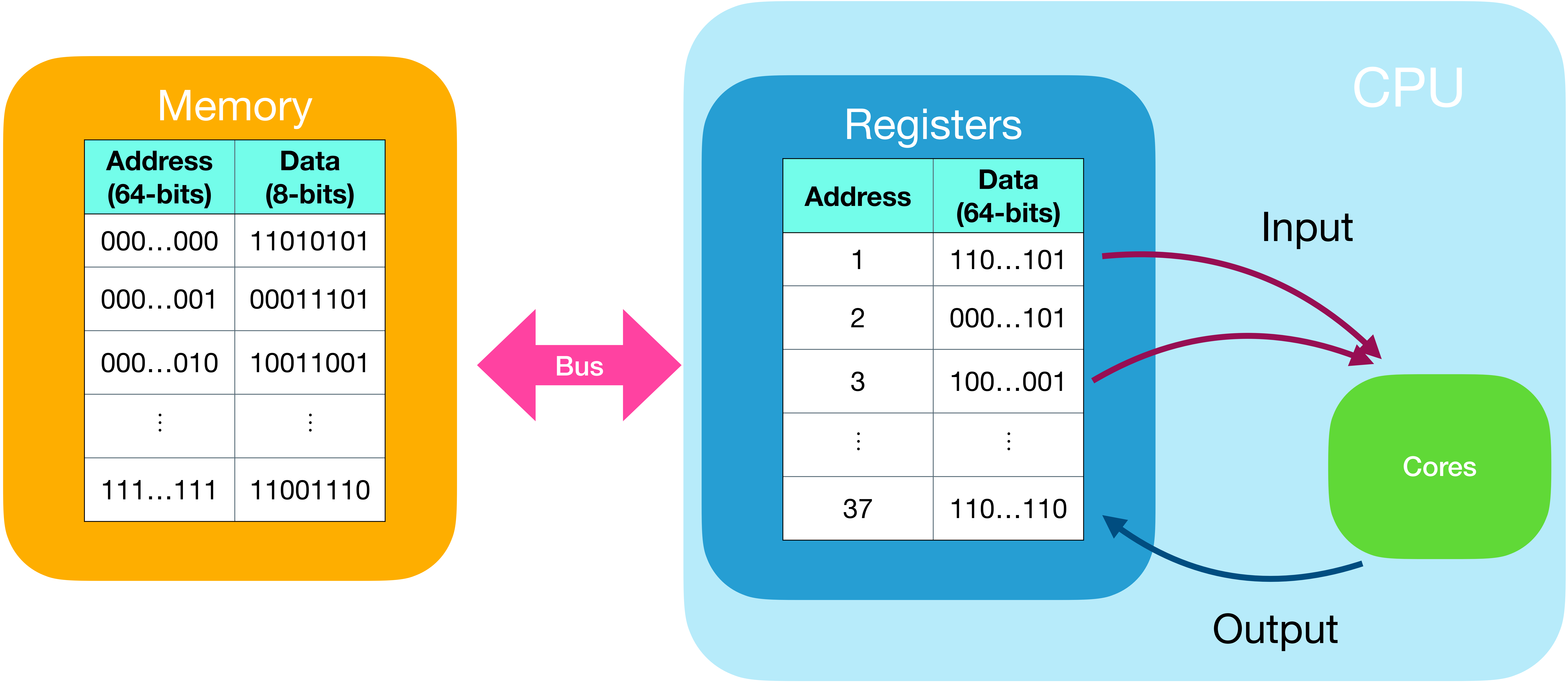
How do computers compute with numbers?

Why are there errors, eg. in divided differences?

Can we understand and bound these errors?

Simplified Model of a Computer

How do computers compute?



Mathematical model

CPUs work on *p*-bits at a time

Cores take (1x or 2x) *p*-bits
and return p-bits.

Operations are

$$f : \mathbb{Z}_{2^p} \to \mathbb{Z}_{2^p} \qquad \text{or}$$
$$f : \mathbb{Z}_{2^p} \times \mathbb{Z}_{2^p} \to \mathbb{Z}_{2^p}$$

for $\mathbb{Z}_m := \{0,1,\ldots,m-1\}$

But how to handle ∞-cardinality sets
integers/reals?

Limitations

- Memory is finite
- All operations work on *p*-bits at a time
- No such thing as throwing an error
- Any operation that manipulates more than *p*-bits must be a composition of simpler functions

Part II

Representing Numbers

1. Integers via modular arithmetic
2. Reals via floating point
3. Floating point arithmetic and bounding errors
4. Interval arithmetic for rigorous computations

II.1 Integers via modular arithmetic

How do we represent integers using only *p*-bits?

Definition 4 (binary format)**.** For $B_0, \ldots, B_p \in \{0, 1\}$ denote an integer in *binary format* by:

$$\pm(B_p \ldots B_1 B_0)_2 := \pm \sum_{k=0}^{p} B_k 2^k$$

Example 3 (integers in binary)

II.1.1 Unsigned (non-negative) integers

Use *p*-bits as first *p* digits of a non-negative number

Arithmetic operations are replaced with modular analogues, where $m = 2^p$:

$$x \oplus_m y := (x + y) \pmod{m}$$
$$x \ominus_m y := (x - y) \pmod{m}$$
$$x \otimes_m y := (x * y) \pmod{m}$$

Example 4 (arithmetic with 8-bit unsigned integers).

Example 5 (overflow with 8-bit unsigned integers)

II.1.2 Signed integers via Two's complement

How do we deal with negative numbers?

If the first bit is 0 the number is interpreted the same as an unsigned integer.

If the first bit is 1 the number is treated as negative. This is done by subtracting 2^p .

Definition 5 (Shifted mod)**.** Define for $y = x \pmod{2^p}$

$$x \ (\mathrm{mod}^{\mathrm{s}}\ 2^p) := \begin{cases} y & 0 \leq \ y \leq \ 2^{p-1} - 1 \\ y - 2^p & 2^{p-1} \leq \ y \leq \ 2^p - 1 \end{cases}$$

Example 7 (addition of 8-bit signed integers)

Example 8 (signed overflow with 8-bit signed integers)

Example 9 (multiplication of 8-bit signed integers)

II.1.3 Hexadecimal format

Numbers are sometimes printed in base-16

Unsigned integers are printed in Hexadecimal/base-16.

This is because base-16 is a power of 2: each digit corresponds to 4 bits.

Hex Digits

Digit	Value
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
a	10
b	11
c	12
d	13
e	14
f	15

Example 10 (Hexadecimal number)

Let’s explore in code.