COA

| Data Representation

Prof. Dr. Mostafa Elhosseini

https://youtube.com/drmelhosseini

AGENDA

- Binary numbers
- Two's complement
- Fixed-point numbers
- Floating-point numbers
- ASCII and Unicode
- Bit fields and packed data
- **Floating-point numbers**

Data Representation

- Microprocessors perform **operations** on data, such as arithmetic, logic, and control operations.
- To perform these operations, microprocessors must represent data in a specific format.
- Data representation in microprocessors is typically done using binary number systems, where data is represented as sequences of bits (binary digits), either 0 or 1.

Data Representation

Here are some common data representation formats in microprocessors:

- Binary numbers
- Two's complement
- Fixed-point numbers
- Floating-point numbers
- ASCII and Unicode
- Bit fields and packed data

Data Representation

Binary numbers

- These are used to represent integers, where each bit position corresponds to a power of 2. For example, the binary number 1011 represents the decimal number 11 $(1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)$.

Two's complement

- This is a method for representing signed integers (positive and negative numbers) in binary form.

Data Representation

Fixed-point numbers

- Represent real numbers by reserving a fixed number of bits for the integer part and a fixed number of bits for the fractional part.

Floating-point numbers

- It uses a scientific notation-like format, where a number is represented by a sign bit, an exponent part, and a mantissa part.
- The IEEE 754 standard is the most widely used floating-point representation in microprocessors.

Data Representation

ASCII and Unicode

- These are character encoding standards used to represent text in microprocessors. **ASCII** is an older, 7-bit standard, whereas **Unicode** is a more modern, variable-width standard that can represent a much larger range of characters, including those from non-Latin scripts.

Data Representation

Bit fields and packed data

- Microprocessors can also represent multiple smaller pieces of data in a single word or register by using **bit fields**. For example, a 32-bit word could store four 8-bit data items, with each 8-bit item occupying a specific position within the 32-bit word.

Floating-point numbers

- Floating-point numbers are a way to represent real numbers in computers, allowing for a wide range of values and decimal points. The IEEE 754 standard is the most widely used floating-point representation in microprocessors.
- It has two common formats: **single-precision** (32-bit) and double-precision (64-bit).

Floating-point numbers

- A single-precision floating-point number is represented using 32 bits, divided into three parts:
  - **Sign bit** (1 bit): Indicates the sign of the number (0 for positive, 1 for negative).
  - **Exponent** (8 bits): Represents the exponent in a **biased** form, with a bias value of 127.
  - **Mantissa** (23 bits): Represents the fractional part of the number's significand in a normalized form.

Example - 03

- Now let's convert a binary single-precision floating-point **number back to a real number**. Let's use the following binary representation as an example:

  01000010001101101000000000000000

Advantage of Single-precision floating-point

Single-precision floating-point representation (often referred to as "float" or "single") has several advantages over other number representation systems, such as fixed-point or double-precision floating-point representation. Some of the advantages are:

- Storage and memory efficiency
- Faster computation
- Wide dynamic range

Advantage of Single-precision floating-point

Storage and memory efficiency

- Single-precision floating-point numbers require only 32 bits of storage, making them more memory-efficient than double-precision floating-point numbers, which require 64 bits. This can be advantageous when dealing with large datasets, where using single-precision numbers can save significant memory

Advantage of Single-precision floating-point

Faster computation

- Due to their smaller size, single-precision floating-point numbers can be processed more quickly by some hardware, leading to faster arithmetic operations, such as addition, subtraction, multiplication, and division. This can be particularly beneficial in real-time applications, where quick calculations are essential

Advantage of Single-precision floating-point

Wide dynamic range

- Single-precision floating-point representation allows for a wide range of numbers, both very large and very small, to be represented with reasonable accuracy. This is useful in applications where values can span multiple orders of magnitude