Project 1: Floating-point Operations in Programming Language

1. Project Summary

In the project, students are supposed to explore how programming languages handle floating-point arithmetic, the nuances of precision, and performance optimizations.

2. Topic Recommendation

Here are some topic recommendations that you can explore, but these are not exhaustive lists. Feel free to delve into any topics related to floating-point operations that interest you:

- **Exploration of Anomalies in POW Function Execution Time**
  1) Interpret results, identify discrepancies, and explain the root causes of differences in behavior and performance.
- **Cross-Language Floating-Point Precision Benchmark and Analysis**
  1) Benchmark floating-point arithmetic across multiple programming languages (e.g., C, Python, Java) with different compilers and settings (e.g., gcc -ffast-math).
  2) Interpret results, identify discrepancies, and explain the root causes of differences in behavior or performance.
- **Profiling Floating-Point Performance on Different Architectures**
  1) Profile the performance of floating-point operations on different hardware architectures (e.g., x86 vs. ARM) using a programming language like C or C++.
  2) Use profiling tools (like gprof, perf, or Valgrind) to measure performance differences.
  3) Analyze the impact of different hardware capabilities on floating-point operations.
- **Decimal Floating-point Arithmetic**
  1) Explore decimal floating-point arithmetic in programming languages (e.g., Python decimal module, Java BigDecimal class)
- **Custom Floating-Point Format and Emulator**
  1) Design and implement your own custom floating-point number format (different from IEEE 754), including defining how numbers are represented (e.g., bit-length for sign, exponent, and mantissa) and then create a software emulator to perform basic arithmetic operations.

3. Submission Requirements

- **Deadline: November 10, 2024 by 23:59.**
- **Late submission policy**
  1) Within 24 hours after the submission is due: 50% point deduction;
  2) More than 24 hours late: 100% point deduction.
- **Submission Checklist**
  1) Technical report, code, and demonstration video (optional).
- **Code Submission Requirements**
  1) Source code files, release version executable files.

2) Please remove unnecessary intermediate files from the project.
3) Please include program instructions, development environment, usage conditions, and run command instructions.

4. References

- IEEE 754-visualization: https://bartaz.github.io/ieee754-visualization/
- Base Convert: IEEE 754 Floating Point: https://baseconvert.com/ieee-754-floating-point
- Float Toy: https://evanw.github.io/float-toy/
- Float Exposed: https://float.exposed/0x40490fdb
- IEEE-754 Floating Point Converter: https://www.h-schmidt.net/FloatConverter/IEEE754.html
- What Every Programmer Should Know About Floating-Point Arithmetic: https://floating-point-gui.de
- The result of 0.1 + 0.2 in various programming languages: https://0.30000000000000004.com