CPU Benchmark

Project realised by *Lethal Company*

Students:

Rareș BECHIR
Nicola Karina Alexandra BOATĂ
Adela-Elena ȘERBULESCU

Timișoara
May, 2024

Chapter 1

Introduction

1.1 Context

- Our benchmark app is testing the CPU performance for the execution of fixed-point and floating-point arithmetic operations.

1.2 Motivation

- The CPU is the brain of any computing device, orchestrating the execution of instructions and managing the flow of data within the system. Its performance is critical in determining the overall speed and efficiency of the system, impacting a wide range of applications from everyday tasks to high-performance computing.
- Our project is a combination between two existing benchmarks, namely testing the performance of the CPU for performing the Bubble Sort algorithm and genarating the digits of number π using Chudnovsky's algorithm and the BPP formula.

Chapter 2

State of the art

The Standard Performance Evaluation Corporation (SPEC) CPU Benchmark Suite is a well-known toolset used to evaluate the performance of CPU and memory subsystems in computing devices. SPEC CPU focuses on measuring the processor's performance through a variety of intensive tasks, including computational workloads that test sorting algorithms and mathematical computations.

Linpack is a widely-used benchmark tool designed to measure a system's floating-point computing power. It is particularly effective for evaluating performance in scientific computing environments and is part of the High-Performance Linpack (HPL) benchmark, often used for ranking supercomputers in the TOP500 list.

When it comes to subble sort testing, Linpack does not natively include a bubble sort benchmark, but its framework can be adapted to implement and measure bubble sort performance. By using the Linpack testing environment, users can write custom test cases to evaluate how well the CPU handles the bubble sort algorithm. On the other hand, our app specifically features bubble sort testing.

In regards to generating the digits of π, Linpack's primary focus is on solving linear equations, which involves intensive floating-point operations. The tool can be configured to execute and measure the performance of algorithms designed for π digit generation, providing insights into the CPU's floating-point computation capabilities. However, Linpack does not specifically include algorithms for generating the digits of π. As mentioned before, it is primarily designed to measure a system's performance through solving dense systems of linear equations. Our benchmark has features testing for the BPP algorithm and the Churdnovsky algorithm.

Chapter 3

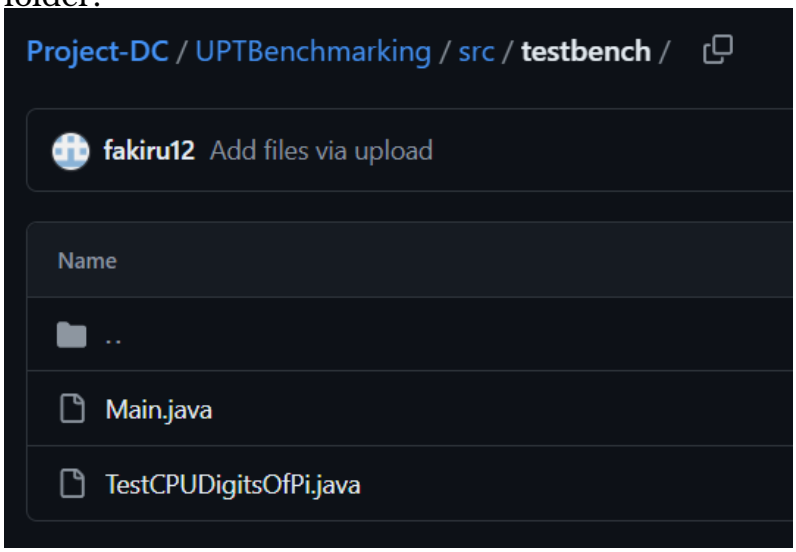
Design and implementation

Our benchmark measures the time for the CPU to perform bubble sorting and the two algorithms for digits of π. The project is structured in four independent packages with different functionalities: bench (which contains the implementations of the algorithms we test), logging (which processes the desired unit of time measurement and displays the results), testbench (which contains the main program for testing bubble sort and a separate one for digits of π) and timing (which implements the stopwatch for our program).

Chapter 4

Usage

- As mentioned before, our program has two separate features for testing bubble sort and digits of π.
- Thus, the user can test the performance for bubble sort by running Main.java and for digits of π, TestCPUDigitsOfPi.java from the testbench folder:



Chapter 5

Results

Here are the results for running the TestCPUDigitsOfPi.java:

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706 8
Chudnovsky algorithm took 1350800 MICROSECONDS
3.14159265358979323846264341193698910358553831488168319519876238741892896977719039977597115285887614 0
BBP algorithm took 5095300 MICROSECONDS
```

For the bubble sort algorithm for an array of 100 elements, it takes about 100k us to execute.

Chapter 6

Conclusions

- As someone who is pretty new to Java, I found the project pretty challenging for me and thus, I didn't contribute enough for the programs themselves and I focused on documenting the project. However, I did grasp some concepts of benchmarking by reviewing the code for the documentation. As for grading myself, I would give myself a 6 at most and more for my colleagues. As for the project meetings, I have nothing to complain about. (Adela Șerbulescu)
- We learned the basics of benchmarking. I would give myslef at least a 5. I mostly contributed with the coding, especially the digits of pi benchmarking. I also think my colleagues deserve at least a 5 or 6. The beginning and debugging was hard, we hated debugging, we liked and enjoyed testing multiple values, especially the ones above 100000. I think the project meetings were fine. (Rareș Bechir)
- We learned the basics of benchmarking and some benchmarking programs. I would give myslef at least a 5. I mostly contributed with the coding, especially the beginning and the bubble sort benchmarking. I also think my colleagues deserve at least a 5 or 6. The beginning and debugging was hard, we hated debugging, we liked and enjoyed when the program worked. I think the project meetings weren't bad. (Nicola Boată)