PC BENCHMARK

Student: Lazar Vlad Adrian

Group: 30433

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA
ROMANIA

Contents

1. Introduction

1.1 Project Requirements

Develop a benchmark application for personal computers, which will measure the following performances of the system under test:

- The type of the processor
- The frequency of the processor
- The dimension of the memory
- The transfer speed of a data block
- The execution speed of arithmetic and logic instructions

1.2 Project Context

Computer benchmarks provide a standard and fair way to test different components of the whole computer system. They vary from graphic benchmarking, logic and arithmetic operations benchmarking, memory block transfer speed, etc.

The proposed PC benchmark application is only a prototype, which is meant for a university project. It could be used, by any PC owner wanting to stress test the power of his machine's processor, with a series of performed operations.

Also, the benchmark will provide a short description of the underlying hardware, so the user will have access to information like frequency, memory size, processor type, so it could also be used by students who want to learn to correlation of the hardware and structure of the PC to the performance it provides.

1.3Objectives

My goal is to implement a basic computer benchmark for a PC, which will address all the main aspects stated in the requirements. Also, the benchmark will state following information about the underlying hardware when opened. For implementations, I will use either C++ or Java

programming languages, and the user interface will be designed either in Qt or JavaFX, depending on the language I will chose for carrying out the development of the project.

The main characteristics of a good benchmark , that I will aim to follow, are:

- Relevance : How close the benchmark behaviour relates to the ones of interest of the consumers
- Reproducibility: produce similar results when the test is run with the same configuration
- Fairness: Allowing different test configurations to compete on their merits (very important in industry)
- Verifiability: Providing confidence that the test results is accurate.
- Usability: avoiding to much dependency on other software & hardware that the users need to run the benchmark.

Weekly plan for project development

WEEK #1 (7TH FEBRUARY)	○ Choose the project
WEEK #2 (21TH MARCH)	○ Establish: 1. Project requirements – what aspects will the benchmark address. 2. Study what is the context of benchmarking on PCs 3. Time table with plan for the future weeks 4. Study some research articles about benchmarking
WEEK #3 (4TH APRIL)	○ Decide on a language for implementations, after finding what C++ and Java can bring to the project. ○ Design a structure of the tests for each of the aspects in the requirements. ○ Begin implementation: on start, gather information about underlying hardware
WEEK #4 (2ND MAY)	○ Begin writing the Implementation Chapter of the documentation

	○ Start the implementation of other types of tests (for speed of data transfer, speed of arithmetic & logic operation execution).
WEEK #5 (16TH APRIL)	○ Analyse the results of more tests, compare them, come up with a formula to compute the score. Save the tests in some files to be able to create a history of the tests on the machine it runs on ○ Design a user interface for the benchmark.
WEEK #6 (30TH MAY)	○ Refine the interface and solve bugs if they appear during testing. Also, complete the testing and validation of the system. Possibly run on another architecture, import the tests, and compare them. ○ Write the "Test and validation" chapter.
WEEK #7 (13TH JUNE)	○ Complete the "Conclusions" chapter in the documentation.

2. Bibliographic research

Benchmarking is a performance measuring technique used in industry to compare certain products or methodologies by measuring their performance in different aspects.

As detailed in [1], beside their main functionality, benchmarks need to be standardized so to cover a wide area from the products in the same domain. Because of this, they are developed by consortia from multiple companies to assure the fairness of the benchmark tool. The most important consortia of this type on the market in the last decades are SPEC (System Performance Evaluation Consortium) and TCP.

There are two main types of benchmarks in industry, namely the specification-based benchmark and kit based benchmark. A hybrid between these two, is also accepted as basically a third type.

Specification based benchmarks begin with the specification of the business problem for which the benchmark is build, hence they may allow the use of some innovative software to build a very specific benchmark for that business.

For kit based benchmarks, the specification is used only as a design guide to building the kit, but the benchmark is not built solely on the specification, to address only a certain business are.

The workload, also briefly explained in [1], is an essential term when discussing about benchmarks. It is a "task", that the SUT (system under test) must perform, on which the performance of the system are evaluated. A good workload should always be fair and should be generally usable, meaning it should be able to be carried out on almost every CPU present on the market, or else, the benchmark using it would fall out the class of standard benchmarks.

In our case, the workload will be composed of a series of stress tests [2]. The capacities of the monitored CPU will be stressed through complex operations, and stats like cycles needed to perform the operations will be monitored.

3. Analysis

As mentioned in the previous section, stress tests are required to be carried out in order to provide an overview of the CPU performance. So, naturally, the issue of choosing good stress tests comes in mind.

Literature research and the testimonials in [2], have inspired me to run 3 kind of tests on the CPU:

Integer operation testing

Integer testing consists of operations involving large integer numbers. For this, my choice as a stress test is computing the nth Fibonacci number, as it performs operations on big numbers if the n parameter is given as a big value. Also, I will look for multiplication/division stress tests.

The Fibonacci Sequence

1,1,2,3,5,8,13,21,34,55,89,144,233,377...

1+1=2	13+21=34
1+2=3	21+34=55
2+3=5	34+55=89
3+5=8	55+89=144
5+8=13	89+144=233
8+13=21	144+233=377

Floating point operations testing

To stress the pc by demanding floating point operations, approximating the value of PI seems the best choice. There is more than one approach on approximating the famous number, as mentioned both in [2] and [3], but my choice is the one involving the Gregory-Leibniz formula:

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots$$

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

The variation parameter here will be the number of terms from the Leibniz series that will be used for computation.

Data block transfer testing

Optionally, if time allows, I will implement testing of the data block transfer speed, by reading and writing large chunks of data in files and monitoring the time taken. Here, I will vary the size of the block transferred.

4. Project Design

The design of the structure should allow, as requested by the requirements, that a test is stored in memory to create a test history, and compare tests one with another.

For this, my approach is to create C++ classes for all types of tests:

- IntegerOperationTest
- FloatingPointOperationTest
- DataTransferOperationTest

A wrapper class, of an all around test, will contain a list of tests for each of the types, along with their results and varying parameters. This structure is also helpful to then put the information in some tables, in a Qt interface.

Beside the functional tests, my plan is to show a system information summary, upon launching the application. This will be a separate System information class, which will be associated with the AllAroundTest, creating a TestRun. The system information will be gathered with the infoware c++ library [4].

Initially, my goal is to display the results of a test run on the interface, to store the results of a history in a file or database, and if time allows, to create graphics to compare 2 tests.

5. Implementation

6. Testing and validation

7. Conclusion

Bibliography

[1] R. Longbottom, UK Government – Stress testing programs (technical report)

https://www.researchgate.net/publication/321868288_Stress_Testing_Programs

[2] Jóakim von Kistowski, University of Wuerzburg – How to Build a benchmark

https://www.researchgate.net/publication/273133047_How_to_Build_a_Benchmark

[3] Approximations for PI (wikipedia articles)

https://en.wikipedia.org/wiki/Approximations_of_%CF%80

[4] Infoware C++ library

https://github.com/ThePhD/infoware?fbclid=IwAR2noupqmHE-p8TuQF7DXxbKxnP0O5ZSfNnfDUSlZKCb-vNXzbY0kMsGRg4