CPU Benchmark

Project realised by *Sveps Team*

Students:

Timotei BADILA
Bogdan BOLOȘ
Aron BUTAR
Raul CAIA
Șerban-Ioan CRAȘOVAN
Andrei-Raul IONESCU

Timișoara

May, 2022

Chapter 1

Read Me

You can remove this file (and chapter) after you finish working on the documentation. Consider it as a cheat sheet for your (probably first) documentation and also as a guideline for using LaTeX and Overleaf. I am sure you will use LaTeX even for other occasions (like writing a CV in a more professional way, writing your diploma project, writing an article, a review and so on). Besides this file, the others can be kept because I expect you to follow the guidelines and the structure I have created for you. You just have to fill in information under some sections and answer to a couple of questions.

The most important thing I am expecting from you while writing this documentation is to learn and make an idea of how such a tool works. You will not receive a lower mark if the documentation doesn't look the best, but I expect you to **use Overleaf**.

Remember:

- In order to save your changes, you have to recompile the project from the 'Recompile' green button.
- A disadvantage of using Overleaf (free edition) in big teams is that the owner of the project can share the project with only one editor, so the other teammates won't be able to make changes to the files. But you can split your documentation into tasks and you can combine your work in the end.
- Each one of you will have to upload the documentation on Campus. I will read only one documentation from each team (they have to be the same) and I will read only the answers to the **conclusions** chapter (I expect them to be unique because I assume each one of you has a different opinion).
- You documentation should not be longer than 12 pages. I know for some of you it's harder to come up with a lot of words, but remember that the documentation also has **tables**, **code sequences**, **figures**.

1.1 How to insert code?

Did you know you can list code in LaTeX? You just have to copy-paste it from the IDE inside a **lstlisting** element. You can provide a caption which explains that sequence of code. The caption doesn't have to be long, but please make it explanatory enough. Please do not copy the entire code, but we'll discuss about this later.

I've attached a short snip of code from one of my previous projects. You can see that the caption is pretty clear and it is longer than just a couple of words, which is good for the reader, who has to understand not only what the code does, but ***why*** you have included a specific code sequence.

```
radioGroup = findViewById(R.id.rol);
registerButton = findViewById(R.id.registerButton);
radioGroup.setOnCheckedChangeListener((group, checkedId) -> {
    RadioButton checkedRadioButton = radioGroup.findViewById(checkedId);
    boolean isChecked = checkedRadioButton.isChecked();
    if(!isChecked){
        setRegisterButtonStyle(false, "#ffe082");
    } else {
        setRegisterButtonStyle(true, "#ffb300");
    }
});
```

Listing 1.1: Checking if the user has selected the role in the application checking a radioButton according to his role (doctor or patient). If any radioButton has been selected, the register button changes is colour and becomes enabled.

Also, we can choose to add a list of all code sequences, a list of all tables and also a list with all figures, but this kind of lists are recommended for more elaborate documents. I remember you the fact that **you shouldn't write a documentation longer than maximum 12 pages**.

1.2 How to insert equations?

You can also list equations (probably the score's formula, right?).

$$z = (x - u)/s \tag{1.1}$$

1.3 How to insert tables?

Nr	CPU	Cores	Clock	RAM	OS	...	Score
1	Core i5 7600K	4/4	3.50	4096	Win10 Home	...	1205
2	Ryzen 7 1700X	8/16	3.00	8192	Win8.1 Pro	...	1622

Table 1.1: Your caption describing the table.

You definitely don't have to worry about learning or understanding details about how tables work in Overleaf, the syntax used to generate a table and other things. Just use a generator (I suggest you to use [1]) and your job will be a lot easier. The table 1.1 is pretty good.

1.4 How to insert images?

You can set a lot of parameters to the pictures you choose to insert in the documentation. You can find out more about them here [2]. For figure 1.1 I chose the width and also I wanted tot keep the aspect ratio unchanged. Inside the folder called **figures** you can store the figures you want to use. Don't delete the ones that are there now because they are used on the first page.

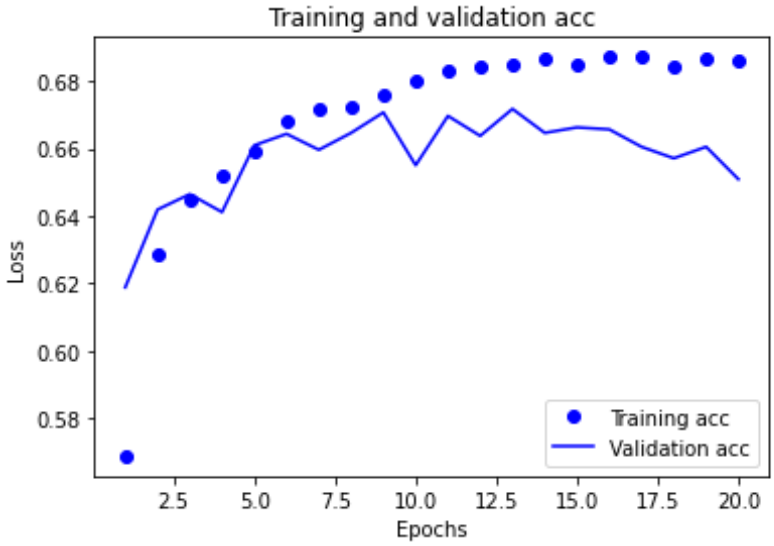


Figure 1.1: Monitoring the training and validation accuracy. In this figure we can see that the model starts overfitting at around the 10th epoch.

Chapter 2

Introduction

2.1 Context

- We decided to test the CPU component through our program by computing various mathematical algorithms. Afterwards, scores will be displayed in the app.
- Our CPU testing application's functionality is computing the Digits of PI up to a specific digit. We did this by computing the Digits of PI in four different ways and then coming up with a final score.

2.2 Motivation

- The CPU is one of the main components of any computer. The more powerful and updated your processor, the faster your computer can complete its tasks. Consequently, we chose to test the CPU because of its importance to any computer and we think everyone should run test on their CPU.
- CPU benchmarks have been implemented for some time, hence our version of testing it is not completely new. However, given the significance of this component, we wanted to do our own design of the application.

Chapter 3

State of the art

- Two benchmarks which compute the Digits of PI and inspired us are Super Pi benchmark and y-cruncher benchmark.
- Both algorithms are popular within the overclocking crowd. Y-cruncher and Super Pi benchmarks have become very used methods for stress-testing applications.
- Super PI is a computer program that calculates pi to a specified number of digits after the decimal point—up to a maximum of 32 million. It uses Gauss–Legendre algorithm and is a Windows port of the program used by Yasumasa Kanada in 1995 to compute pi to 232 digits. Super Pi is much slower than other programs and it utilizes inferior algorithms to them.
- For larger numbers, first implementation of Nilakantha is faster and the better option. On the other hand, the second implementation of Nilakantha is faster for smaller numbers.
- Some disadvantages include: voltage, clock speed.

Chapter 4

Design and implementation

- The benchmarks we provide are able to calculate the n digits of Pi in 4 different methods.
- The 4 formulas we used for calculating the digits of Pi are: Spigot Algorithm of Rabinowitz and Wagon, an Arithmetic-geometric method and 2 Nilakantha series (2 different methods for the same algorithm).
- Nilakantha attributes the series to an earlier Indian mathematician, Madhava of Sangamagrama, who lived c. 1350 – c. 1425.[74] Several infinite series are described, including series for sine, tangent, and cosine, which are now referred to as the Madhava series.

```
    // accuracy parameters set to 1000 arbitrarily
    for(int i=2;i<1000;i+=4)
    {
        denominator1 =  BigDecimal.ONE;
        denominator1 = denominator1.multiply(new BigDecimal(i));
        denominator1 = denominator1.multiply(new BigDecimal(i+1));
        denominator1 = denominator1.multiply(new BigDecimal(i+2));

        denominator2 =  BigDecimal.ONE;
        denominator2 = denominator2.multiply(new BigDecimal(i+2));
        denominator2 = denominator2.multiply(new BigDecimal(i+3));
        denominator2 = denominator2.multiply(new BigDecimal(i+4));
        term1 = new BigDecimal("4").divide(denominator1, 1000, RoundingMode.HALF_UP);
        term2 = new BigDecimal("-4").divide(denominator2, 1000, RoundingMode.HALF_UP);

        pi = pi.add(term1);
        pi = pi.add(term2);
    }

    pi = pi.add(new BigDecimal (3));

    pi = pi.setScale(size, BigDecimal.ROUND_HALF_UP);
    pi =  BigDecimal.ZERO;
}
```

Another approach:

```
      public void run() {
    size -= 1;
    BigDecimal pi = new BigDecimal(3);
    BigDecimal divisorBegin = new BigDecimal(2);
    BigDecimal divisor;
    BigDecimal one = new BigDecimal(1);
    BigDecimal two = new BigDecimal(2);
    BigDecimal four = new BigDecimal(4);
    BigDecimal minusFour = new BigDecimal(-4);
    MathContext mc = new MathContext(size);
    for (long i = 0; i < size; ++i) {
        divisor = one;
        for (long j = 0; j < 3; ++j) {
            divisor = divisor.multiply(new BigDecimal(j).add(divisorBegin, mc), mc);
        }
        pi = pi.add(i % 2 == 0 ? four.divide(divisor, mc) : minusFour.divide(divisor, mc),
             mc);
        divisorBegin = divisorBegin.add(two, mc);
    }
}
```

The Arithmetic-geometric method it is a direct consequence of Gauss' arithmetic-geometric mean, the traditional method for calculating elliptic integrals, and of Legendre's relation for elliptic integrals. The error analysis shows that its rapid convergence doubles the number of significant digits after each step. The new formula is proposed for use in a numerical computation of Pi.

```
     private static MathContext con1024;
private static final  BigDecimal bigTwo = new BigDecimal(2);
private static final BigDecimal bigFour = new BigDecimal(4);

private static BigDecimal bigSqrt(BigDecimal bd, MathContext con) {
    BigDecimal x0 = BigDecimal.ZERO;
    BigDecimal x1 = BigDecimal.valueOf(Math.sqrt(bd.doubleValue()));
    while (!Objects.equals(x0, x1)) {
        x0 = x1;
        x1 = bd.divide(x0, con).add(x0).divide(bigTwo, con);
    }
    return x1;
}

@Override
public void run() {
    BigDecimal a = BigDecimal.ONE;
    BigDecimal g = a.divide(bigSqrt(bigTwo, con1024), con1024);
    BigDecimal t;
    BigDecimal sum = BigDecimal.ZERO;
    BigDecimal pow = bigTwo;
    while (!Objects.equals(a, g)) {
        t = a.add(g).divide(bigTwo, con1024);
        g = bigSqrt(a.multiply(g), con1024);
        a = t;
        pow = pow.multiply(bigTwo);
        sum = sum.add(a.multiply(a).subtract(g.multiply(g)).multiply(pow));
    }
    BigDecimal pi = bigFour.multiply(a.multiply(a)).divide(BigDecimal.ONE.subtract(sum),
        con1024);
}
```

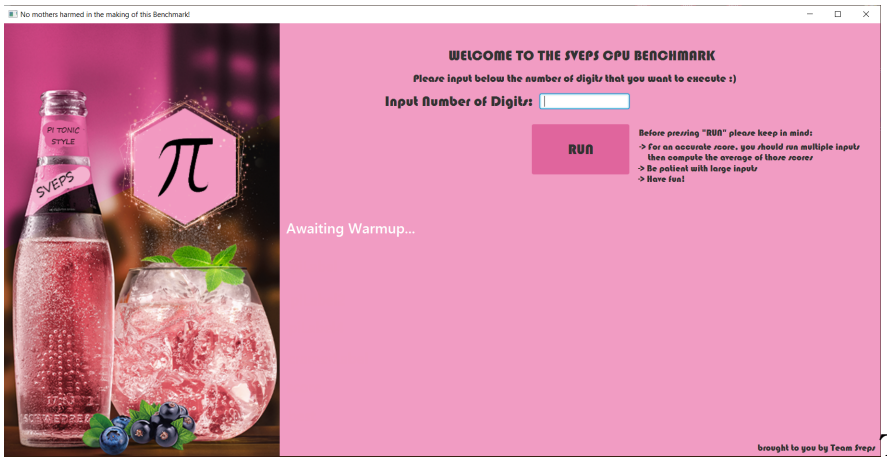
Cramer's rule is an explicit formula for the solution of a system of linear equations with as many equations as unknowns, valid whenever the system has a unique solution.

```
 public void run() {
    double[][] tmp=new double[mat.length][mat.length];
    double[] x=new double[mat.length];
```

```
        double detA;
        long timeout = System.currentTimeMillis();
        detA = computeDeterminant(mat);
        for(int m=0, n=mat.length; m<n; m++) {
            for(int i=0; i<n; i++)
                System.arraycopy(mat[i], 0, tmp[i], 0, n);
            for(int j=0; j<n; j++)
                tmp[j][m]=coef[j];
            x[m]=computeDeterminant(tmp)/detA;
        }
    }
```

Chapter 5

Usage



This is the main view of our application. It is pretty straightforward : after inserting in the box the desired number of digits to be computed, the user can press the run button which will start computing the pi number.

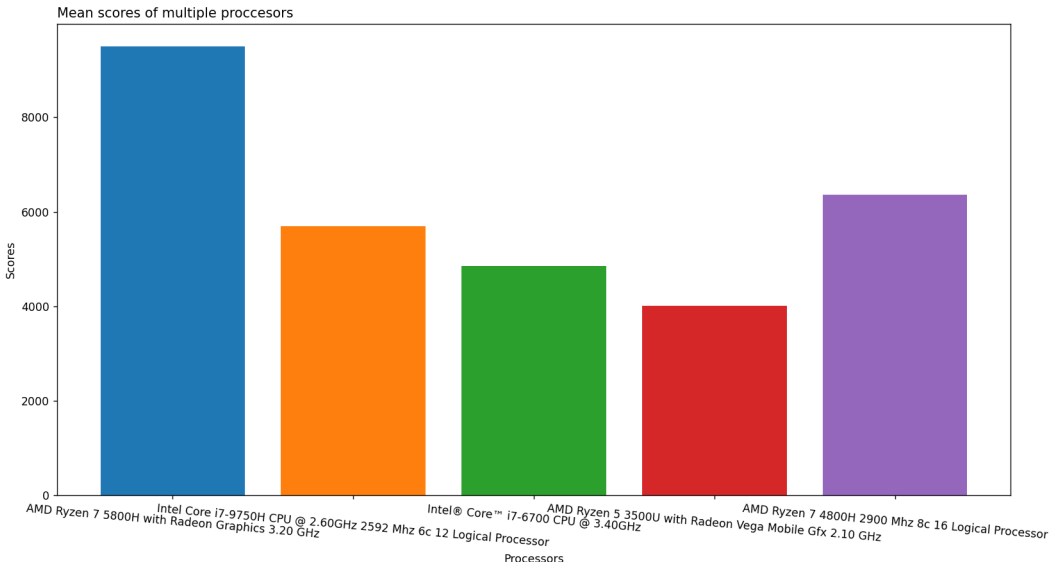
As shown in the picture above the result is the time it took for each algorithm to calculate up to the desired digit of pi.The way we grade our CPU's is by doing the average between the number of nanoseconds it took for all the algorithm's to finish computing divided by the number of desired digits.

Chapter 6

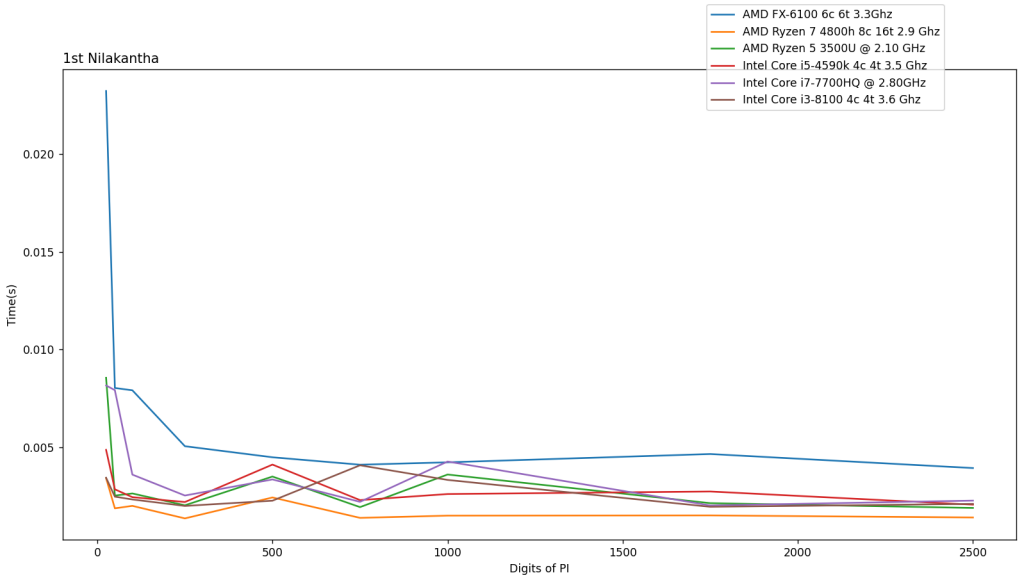
Results

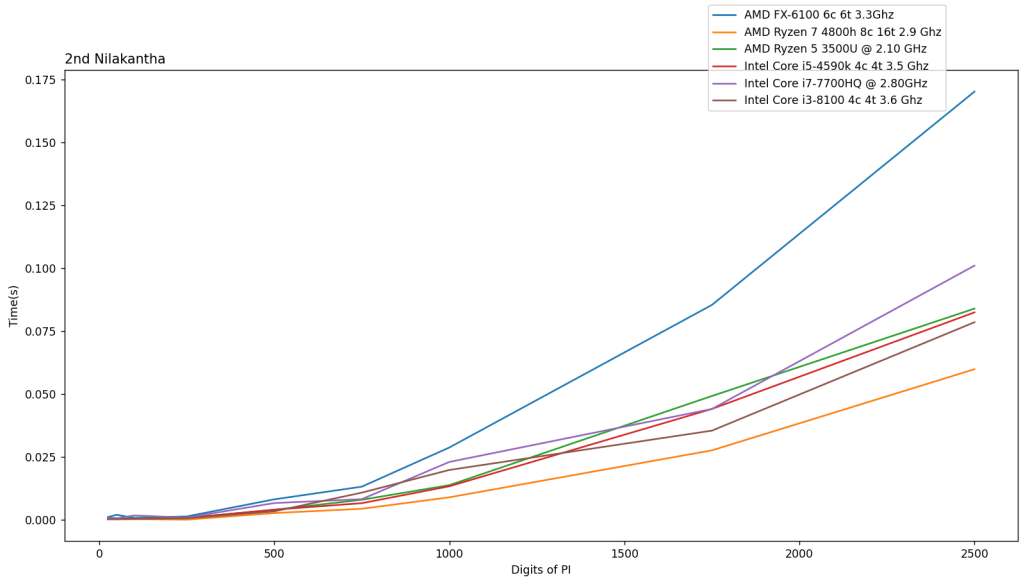
This chapter is again very important in what means **analysing**. If in chapter 2, the focus is on comparing methods, here, the focus is on comparing and on interpreting results.

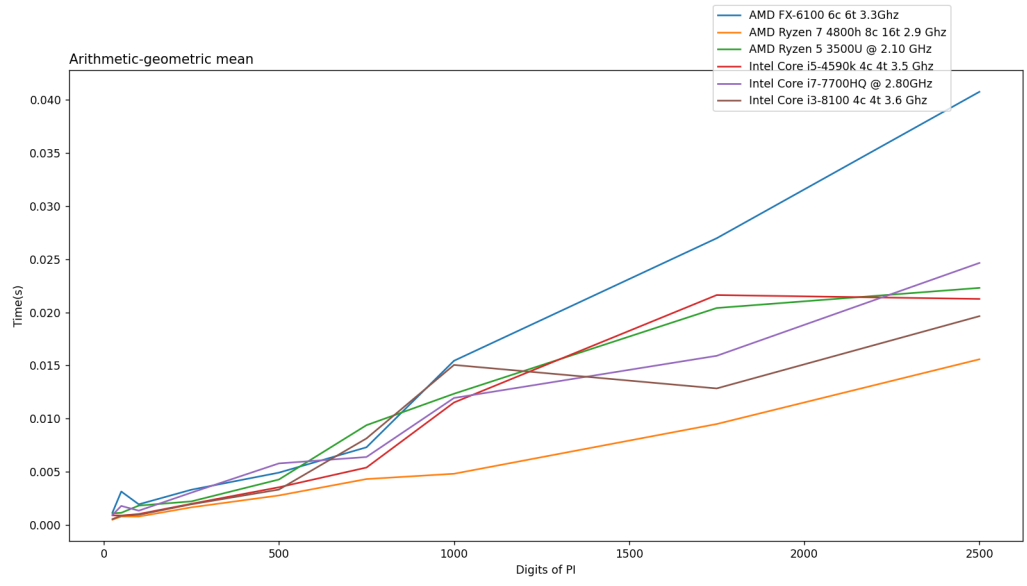
- We run the benchmark of different processors and stored the results. We have two kind of measurements: (1) simple measurements - the processor, the number of digits and the score it got - and (2) complete measurements - the time it took each algorithm to run is stored as well.
- For each processor, we run the benchmark for multiple values of digits (25, 50, 100, 250, 750, 1000, 1750, 2500).
- The following plot shows the average score for 5 processors (simple measurements).

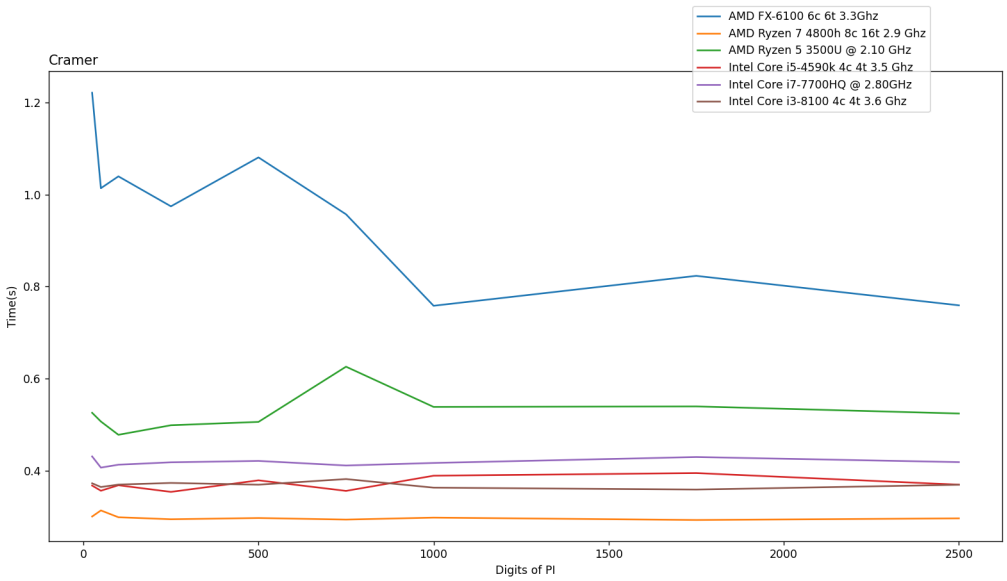


- The following set of plots shows how 6 processors performed of the 4 algorithms used to compute the digits of PI. (complete measurements).

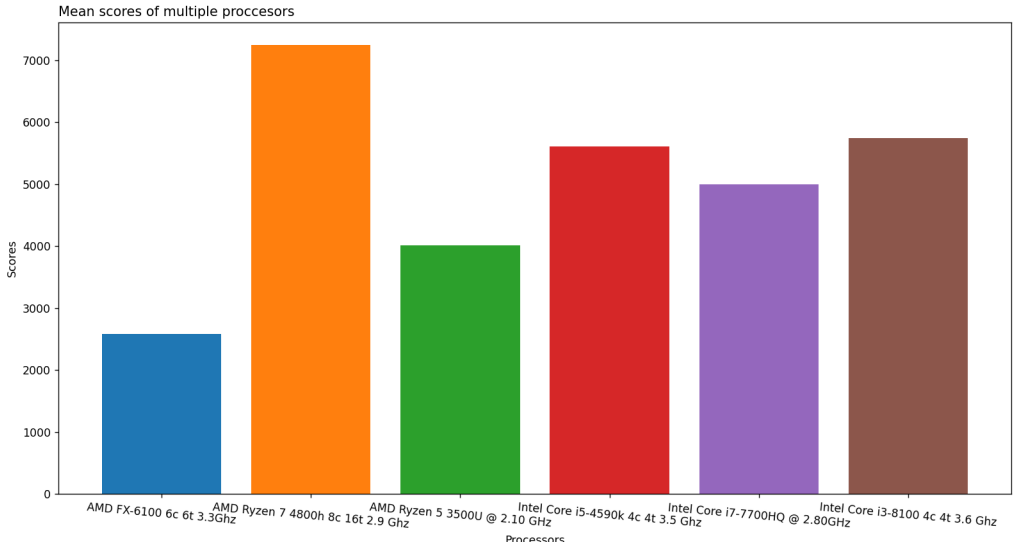

1st Nilakantha
AMD FX-6100 6c 6t 3.3Ghz
AMD Ryzen 7 4800h 8c 16t 2.9 Ghz
AMD Ryzen 5 3500U @ 2.10 GHz
Intel Core i5-4590k 4c 4t 3.5 Ghz
Intel Core i7-7700HQ @ 2.80GHz
Intel Core i3-8100 4c 4t 3.6 Ghz
Time(s)
0.020
0.015
0.010
0.005
0
500
1000
1500
2000
2500
Digits of PI


2nd Nilakantha
AMD FX-6100 6c 6t 3.3Ghz
AMD Ryzen 7 4800h 8c 16t 2.9 Ghz
AMD Ryzen 5 3500U @ 2.10 GHz
Intel Core i5-4590k 4c 4t 3.5 Ghz
Intel Core i7-7700HQ @ 2.80GHz
Intel Core i3-8100 4c 4t 3.6 Ghz
Time(s)
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
500
1000
1500
2000
2500
Digits of PI


Arithmetic-geometric mean
AMD FX-6100 6c 6t 3.3Ghz
AMD Ryzen 7 4800h 8c 16t 2.9 Ghz
AMD Ryzen 5 3500U @ 2.10 GHz
Intel Core i5-4590k 4c 4t 3.5 Ghz
Intel Core i7-7700HQ @ 2.80GHz
Intel Core i3-8100 4c 4t 3.6 Ghz
Time(s)
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
500
1000
1500
2000
2500
Digits of PI



- And here you can see the average scores for these 6 processors.



Chapter 7

Conclusions

Please also focus on this chapter! I know that sometimes while writing a document, conclusions are the last thing to write, but remember that they are also the last thing to read and might change the reader's overall perspective of the document. I would want you to answer to the following questions and, why not, even some added by yourself.

- **What did we learn?**

  (e.g.) During this semester, we learned what benchmarks are and how to make and use them. We learned how to interpret the results we get from the benchmark.

- **What mark would I give to me? How much did I contribute to the team work (project, application, presentation)? What mark do I think my colleagues deserve?**

- **What was hard, what did we enjoy, what did we hate, what did we like and dislike about the CO project?**

- **How would you change (improve) the project meetings for the next generation of students?**

Bibliography

[1] *Table Generator for LaTeX*. https://www.tablesgenerator.com, 2020. [Online; Accesat: 07.05.2021].

[2] *Inserting images in Overleaf*. https://www.overleaf.com/learn/latex/Inserting_Images, 2020. [Online; Accesat: 07.05.2021].