Kaitlyn Stumpf
CSC231, Fall 2015
10/2/2015

Problem 1: When performing arithmetic, how does Javascript score in terms of speed of execution, compared to Java and Python, as illustrated in the page on Nasm, Java, Python that we discussed in class?

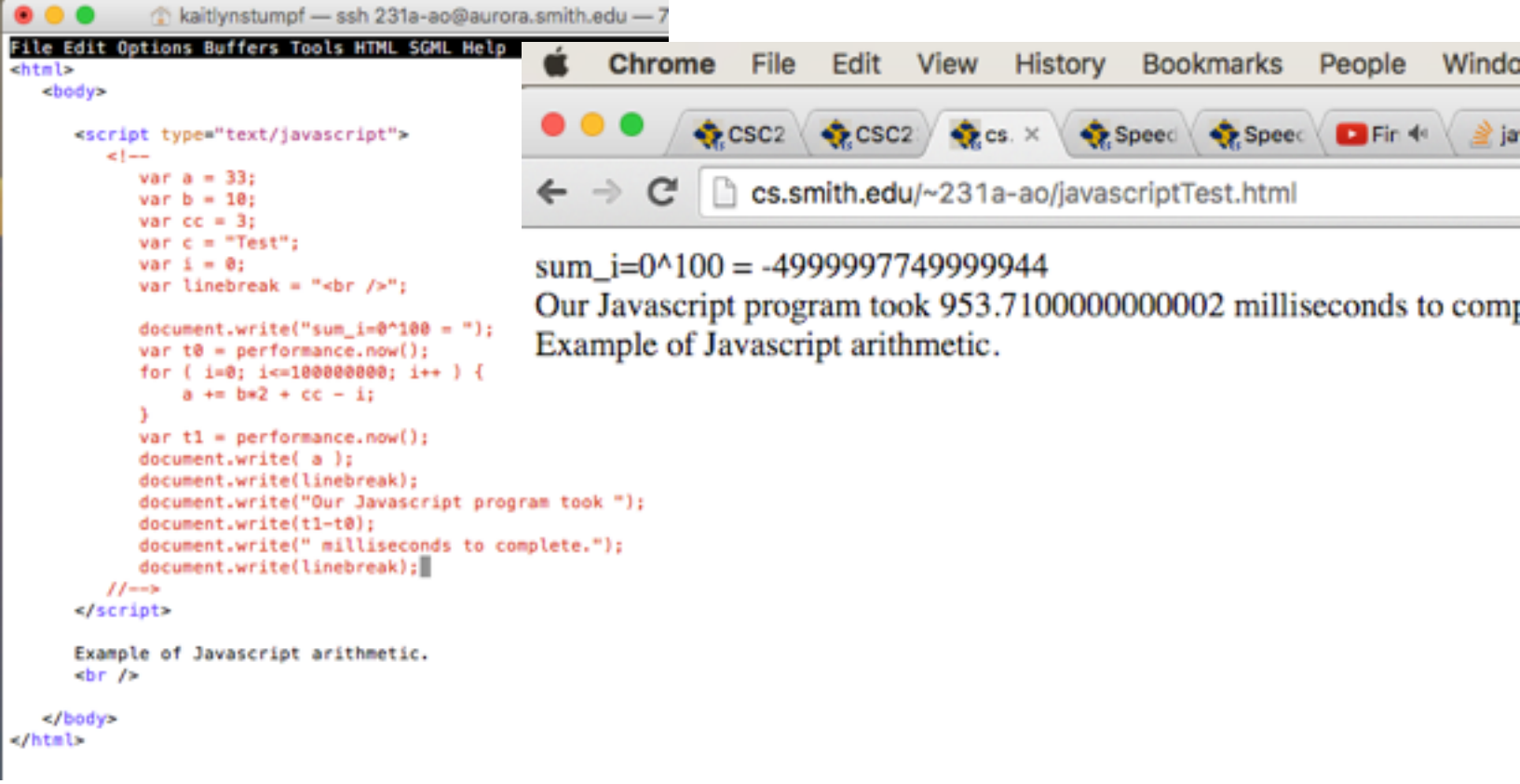
When we compared the speed of execution of Java, C++, Nasm, and Python when executing arithmetic, there were definite differences in performance. Let's paint a picture of each language's performance by first talking about Java. Java took 104ms to perform the given loop, or 1ns per iteration, and returned an overflowed value for its answer to the arithmetic problem. According to the Wikipedia page on Java performance, "because compiled Java programs run on the Java Virtual Machine rather than directly on the computer's processor like C and C++ programs do", Java is often considered slow compared to other compiled languages. Because this is a concern, and because many businesses switched to Java after it gained popularity, optimization has been a main priority ever since. At this point, Java relies on JIT (Just-In-Time) compilation, which is significantly faster than previous generations because it supports better code analysis and because the Java Virtual Machine has been optimized.

This above research explains why the optimized C++ code took 0.7ns per iteration, and was thus 30% faster than the Java code. Because C++ runs directly on the processor, it operates more quickly than Java. C++, however also returned an overflowed value for its answer. Because both Java and C++ are compiled languages, the variable type remains static and it is trusted that whatever variable type a value is stored in will be able to hold this value. When, as is the case with this example code, the value becomes too large for the variable type to hold, the variable overflows, and its MSB is set to 1, hence an overflowed value having a negative sign.

When we look at optimized Nasm code we see that it is better than Java performance, but not as impressive as the optimized C++ code. This is because, as explained in class, the Nasm compiler does not necessarily compile in the most efficient way. It is done automatically by the computer, and so may not be as inventive or frugal as a human would when it comes to code optimization. It's most important job is making sure the code gets compiled properly, with optimization coming in as a second priority.

Python code is a whole other beast. As an interpreted language, it compiles each line as it runs, and so catches the overflowed variable and switches to a variable type that can adequately house all of the value. This constant interpretation as the program runs comes at a cost, however, making optimized C++ 300 times faster than Python.

Now that we understand how Java, C++, Nasm, and Python all compare, let's take a look at Javascript. Our Javascript program was run on the same computer (beowulf2) as all the other programs, so we need not worry about a discrepancy here. To avoid any other discrepancies, I updated the Javascript code to mirror the arithmetic performed when comparing the other languages. Below is a copy of my code and a screenshot of Javascript's performance.



Shockingly, Javascript took a whopping 953.7ms to complete, compared to Java's 104ms. We could assume that without a firm grasp on Javascript optimization practices, I did not properly optimize the code written above. However, the only thing being timed is the for-loop, and I followed W3Schools' advice to reduce activity in my loop. A Wikipedia search reveals that Javascript does run slightly slower on Google Chrome, but this doesn't explain the huge drop in performance speed. This resource (https://cycling74.com/forums/topic/java-200-times-faster-almost-as-easy-to-use-so-why-javascript/) holds the answer. It states that Javascript is between 200 and 400 times slower than Java, and this is because Javascript is an interpreted language, so naturally it is doing more work as it reads and compiles each line of the code it is given. As a warning, other resources praise Javascript as being faster than Java or just about the same speed (as both were worked on by a programmer named Lars Bak), and likely biased all resources are biased toward the language the author prefers. It is important that rather than taking anyone's word on this, we use reliable calculations to compare performances between languages.