# Benchmarking

We decided to implement a simple merge sort to test our VM. With merge sort we could test how the VM would perform with a very CPU and Memory heavy program. We ran the tests on several different versions of thee VM, the one we had at the presentation which didn’t have a GC, our current implementation with the GC turned off but with a very big start heap size, with the GC turned off but a small start heap size so it would have to expand the heap to accommodate the program and last with GC turned on.

The first thing to note is that the current implementation is slower, this was not that surprising since we have added features to the VM and changed the way it handle Handles and not done a single thing to optimize it. It is good to see that the GC version performs almost as well as the Non-GC version. Finally the Non-GC version with the low start heap size has to do a lot of expanded to be able to contain all the generated memory allocations to its predictably slow.

We also tested our VM against other systems, C# and JavaScript, to see how well it would perform compared to those. The results we got was however incomparable. While our VM took almost ten minutes to sort 100.000 elements C# and JavaScript sorted 1.000.000 elements in half a second and two seconds respectively.

We knew it would be slow but maybe not that slow. Part of the reason is that many of the basic instructions such as Add, Subtract and so on are external library functions which are expectably slower than if we had implemented them direct as opcodes. It would have been interesting to see how fast we could have made it run with more basic opcodes and optimizations such as inline-caching, JIT. But as noted in our plan we never attempted to focus on performance, we were more interested in building a general VM since we never done that before. That said it would have been nice if it had been a bit faster.