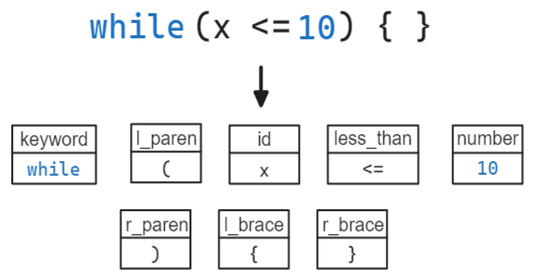
BRANCHLESS AND PARALLEL TOKENIZATION

To what extent can SIMD instructions accelerate lexical analysis?

1. INTRODUCTION

Lexical analysis, also known as lexing, is the first stage of compiling, traslating source code into a series of tokens.



Single instruction, multiple data (SIMD) is a type of parallel processing that enables computers to perform operations on vectors instead of scalar (single) values.

A lexer can be broken down into multiple sub-lexers identifying groups of characters with structural similarities, enabling the usage of SIMD instructions.

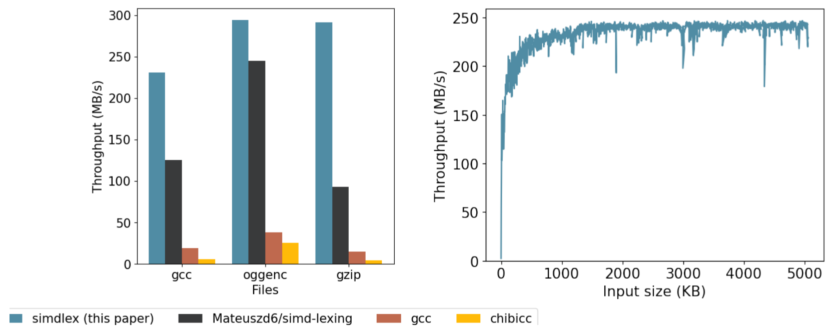
2. METHODOLOGY

The SIMD lexer traverses source code as a string literal in batches of 32 bytes. Within each batch, individual sub-lexers classify each byte either as the beginning or as part of a token body. These results are then combined using a prioritized scheme of sub-lexers to find the final tokens.

Experiments:

• Using large single compilation-unit C programs [1], time the SIMD lexer against existing solutions such as the lexer of GCC [2] and of a toy compiler CHIBICC [3] and another SIMD lexer made by the GitHub user Mateuszd6 [4].

3. RESULTS



First plot shows throughput (MB/s) of different lexers for various files on an AMD Zen 3 processor (3.3 GHz) using the CLANG 18 [5] compiler (with the -O3 flag). This plot provides insight into the performance of the lexer under real-world conditions **(higher is better)**.

Second plot presents lexings speed (MB/s) for synthetic files of various sizes. This analysis allows us to understand how the lexing performance scales with file size **(higher is better)**.

Both experiments are replicable [6].

4. CONCLUSIONS

Given the results from our experiment significant speeds ups can be seen even over mainstream lexers such as that employed by GNU GCC (by an average **13x** speed up).

While results are promising, it is important to keep in mind that the proposed architecture is more complex than what is currently used by other lexers.

5. FUTURE WORK

Looking ahead, future efforts could focus on parallelizing lexing using Multiple Instruction Multiple Data (MIMD) techniques, such as multithreading.

This research was limited to only using x86 SIMD instructions. Extending these optimizations to ARM would be beneficial to cover the most common architectures found in hardware.

REFERENCES

[1] https://people.csail.mit.edu/smcc/projects/single-file-programs
[2] https://gcc.gnu.org/
[3] github.com/rui314/chibicc
[4] https://github.com/Mateuszd6/simd-lexing
[5] https://clang.llvm.org/
[6] https://github.com/alexbolfa/simd-lexer-experiments

Author: Alexandru Bolfa a.bolfa@student.tudelft.nl **Responsible Professor:** Soham Chakraborty **Supervisor:** Dennis Sprokholt TUDelft