Solozhenkin Sevastyan

1st year MIPT student

CONTACTS

sevsol2512@gmail.com
+7(903)729-59-70
t.me/sevsol
github.com/sevaphasol

EDUCATION

University
Moscow Institute of Physics and Technology

Department
Phystech School of Radio Engineering and Computer Technology

Specialization
Informatics and Computation Technology

Average grade	**9.3/10.0**
Informatics grade	**10/10**

School
Kurchatov school (prev. 1189)

Achievements

- Winner of the Kurchatov Physics Olympiad
- Prize winner of 4 Physics Olympiads
- Prize winner of 3 Mathematics Olympiads
- Graduated from school with gold medal

HARD SKILLS

Programming languages
C, Python, asm x86-64

Tools
Git, GDB, EDB, IDA, Makefile, Graphviz, Google Test, SFML

SOFT SKILLS

- Determined
- Persistent
- Hardworking
- Fast learning

PROJECTS

SIMD optimization

github.com/sevaphasol/mandelbrot

Using SIMD instructions (intrinsics) to optimize a program visualizing the Mandelbrot set. The visualization is implemented using the SFML library. Analyzing the compiler-generated assembly code with various compilation flags.

Printf implementation

github.com/sevaphasol/printf

Implementation of a simplified version of printf from libC. Supported output specifiers include floating-point numbers (%f), various integer data types (%d, %x, %o, %b), character (%c), string (%s), unique specifier for colored text output. Unit tests are implemented using the GoogleTest library.

Crack

github.com/sevaphasol/crack

Using a disassembler to analyze an executable file. Exploiting a buffer overflow vulnerability in the program. Implementing a patcher with a graphical interface using the SFML library.

Virtual machine

github.com/sevaphasol/SPU

Implementation of a virtual machine with its own assembler, RAM, and video memory. Additionally, for this task, a stack was implemented with various data protection mechanisms, including safeguards against issues related to multithreading.

Language

github.com/sevaphasol/language

Implementation of a compiler for a custom programming language with a backend in assembly for the virtual machine from the task above. Support for recursive calls is implemented using stack frames. Additionally, a reverse frontend and cross-compilation with another compiler are implemented (support for a unified IR standard).