BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
ENGLISH SPECIALIZATION

DIPLOMA THESIS

Design, implementation and comparision of different software emulating methods

Supervisor
Lect. dr. Mihai Andrei

Author
Oniga Andrei-Mihai

2025

ABSTRACT

Abstract: un rezumat în limba engleză cu prezentarea, pe scurt, a conținutului pe capitole, punând accent pe contribuțiile proprii și originalitate

Contents

1 Introduction

1.1 About

Introducere: obiectivele lucrării și descrierea succintă a capitolelor, prezentarea temei, prezentarea contribuției proprii, respectiv a rezultatelor originale și menționarea (dacă este cazul) a sesiunii de comunicări unde a fost prezentată sau a revistei unde a fost publicată.

1.2 Related work

CHIP8 Applications in engineering [1]. Brainfuck in reinforcement learning [6]. Esoteric languages list with BF in it [3]. Brainfuck conceptual [5]. Brainfuck hardware [4]. Brainfuck self interpreter [2].

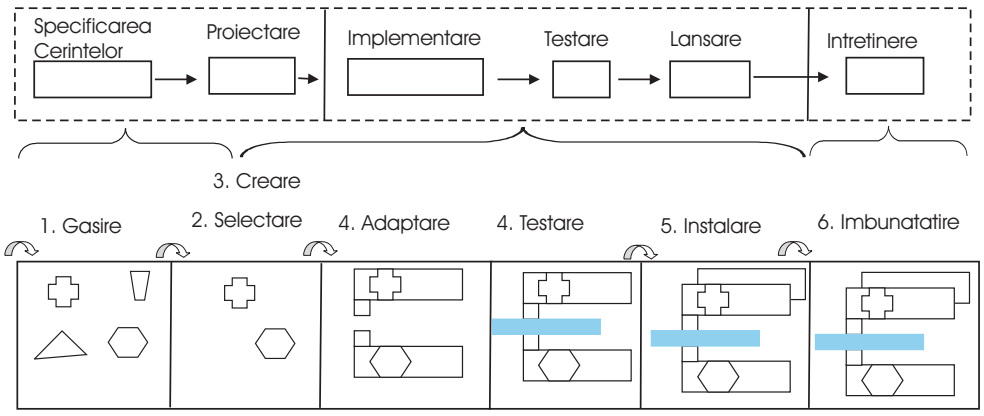

Figure 1.1: Ciclul de dezvoltare al sistemelor bazate pe componente adaptat modelului cascadã

Inserarea și Referirea la Tabelul 1.1.

Nume algoritm	Toate soluțiile	Soluția optimã
Nume 1	20	5
Nume 2	20	2

Table 1.1: Soluții obținute

Adaugarea și Referirea la o Ecuație 1.1.

$$ws_N4 = w_{14} * N1 + W_{24} + N2 + w_{34} * N3 \tag{1.1}$$

2 BF

Brainfuck, or as it is usually shortened for academical purposes, BF, is an esoteric programming language designed to be minimalistic and intentionally difficult to program in.

It was created in 1993 by Swiss student Urban Müller, primarily as a challenge for programmers.

Despite its simplicity, Brainfuck is Turing-complete, meaning it has the computational power to perform any calculation that can be computed by a modern computer, given sufficient resources.

The language operates on an array of memory cells, each initially set to zero, and uses a series of eight commands to manipulate the data and control program flow.

Brainfuck's syntax consists of only eight commands, these being instructions to move the data pointer, modify the value of the current memory cell, input or output data, and control loops.

The language's simplicity—combined with its deliberately difficult syntax—forces programmers to think creatively about how to implement basic operations like arithmetic or data manipulation.

As a result, Brainfuck is more often used in programming challenges and competitions rather than for practical software development.

The language has been made intentionally difficult to use for practical application, which has made it a popular subject of study within the community of esoteric programming languages (esolangs).

Brainfuck often serves as an example of the power of minimalism in programming, demonstrating that even with extremely limited resources, a fully functional computational system can be constructed.

It challenges conventional programming paradigms, as the programmer is forced to manage memory and control flow manually, akin to working in assembly or machine code.

2.1 Machine specification

+	Increments the value at the current position that the machine points to.
-	Decrements the value at the current position that the machine points to.
<	Moves the pointer one cell to the left.
>	Moves the pointer one cell to the right.
[	Jumps after the corresponding closed bracket when the value at the current cell is 0.
]	Jumps after the corresponding open bracket when the value at the current cell is not 0.
.	Outputs the value at the current cell.
,	Read a value to be placed at the current cell.

Table 2.1: BF commands and their descriptions

2.1.1 Programming in the language

For us to properly understand how to use the language in a useful manner, let's take a look at a well known sample used for printing 'Hello World!' to the screen:

```
>++++++++[<+++++++++>-]<.
>++++[<+++++++>-]<+.
+++++++..+++.
>>++++++[<+++++++>-]<++.
------------.
>++++++[<+++++++++>-]<+.
<.
+++.
------.
--------.
>>>++++[<++++++++>-]<+.
```

I took the liberty of splitting the code by each character that is outputted to the screen such that it can be properly observed how the language works.

Let's take a look at the code, line by line:

```
>++++++++[<+++++++++>-]<.
```

This instruction sequence sets the value of the cell at index #1 to 8 and in a continous loop decrements that value and increments cell #2 nine times for each decrement. Thus it sets cell #2 to value 72, ASCII for the character 'H'.

Similar steps are done for all of the other characters except the ones that repeat multiple times, like 'l' which is printed twice by using two dot instructions in the third line, or reusing the value of 'o' from 'Hello' in 'World'.

This sequence constitutes a good introduction to the matter at hand as, by coding in the language, common programming patterns can be observed such as the multiplication pattern:

```
(any number of pluses)[(direction change)(any number of
↪  pluses)(reversed direction change)-]
```

There are also other commonly used patterns, such as:

```
The cell clearing pattern, that sets a cell to 0:
[-]
```

```
The cell moving pattern, that moves a value to a cell relative to
↪  it:
[(any number of moves)+(same amount of reversed moves)-]
```

```
3 To be noted that this assumes initial cell value 0, if that value
  ↪ is not 0 it will be added instead.
4 If you need it to be specifically moved, you can combine the
  ↪ previous clear pattern with this one.
```

2.2 Emulator implementation

Because of the language's simplicity, there have been many emulators made for it in all kinds of programming languages. As such, the focus will be put on the techniques utilised in the creation of optimized emulators rather than the emulators themselves, as they have been a rather exhausted subject.

2.2.1 Emulating yourself?

2.2.2 Simple interpreter

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.2.3 Static compilation

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.3 Applying optimizations

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.3.1 Precalculating jumps

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nos-

trud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.3.2 Consolidating operations

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.3.3 Sequence matching for common patterns

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

2.4 Testing

3 CHIP8

3.1 About

CHIP8, sometimes spelled as CHIP-8, is a programming language and virtual machine specification developed by Joseph Weisbecker on the 1802 processor of the COSMAC VIP computer in the mid-1970s.

It was meant to be an educational tool mainly designed around creating simple video games with much more ease and less resources than conventional programming languages of the time such as BASIC.

Even today it is widely used as an introduction for people that are taking up software emulation as a programming hobby because of its simplicty and ease of implementation.

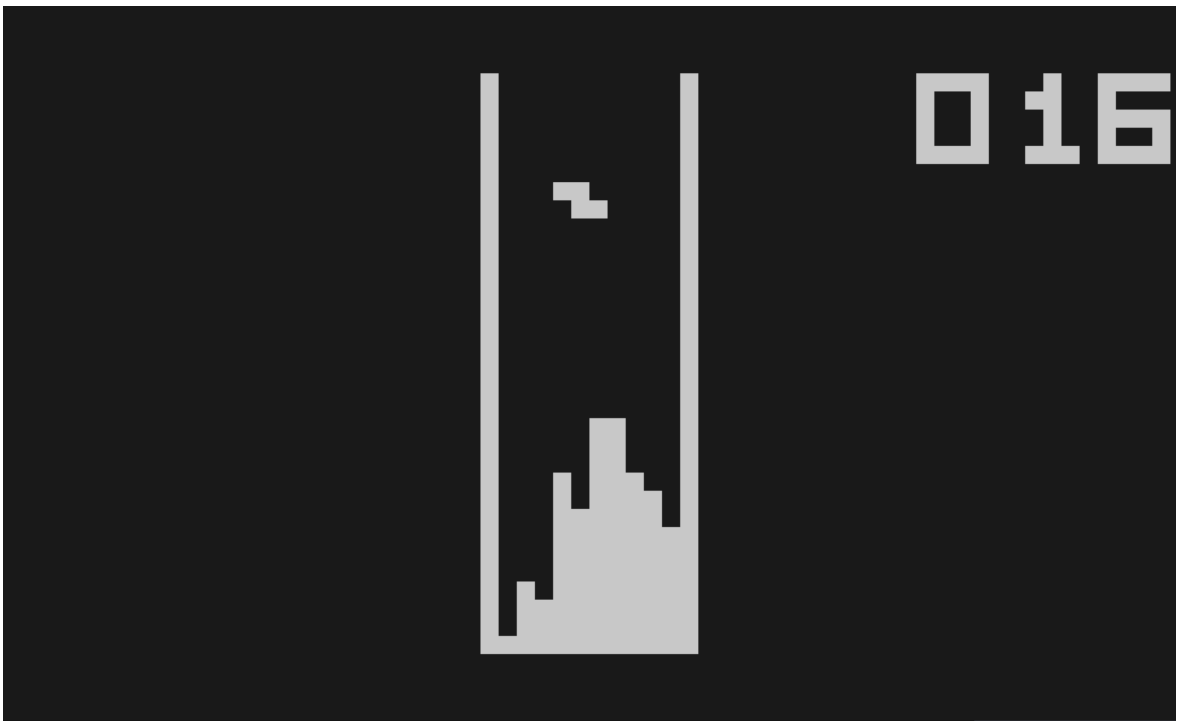

Figure 3.1: Tetris clone written in CHIP8 by Fran Dachille in 1991 running in Edra.

3.2 Virtual machine description

Memory

The CHIP8 virtual machine has 4KB of accessible memory, as is the limit when using 16-bit memory registers. An exception occurs for the I register used mostly in drawing and array management opcodes, which is a 12-bit register instead of 16-bit. Multi-byte data is stored in big-endian format, meaning that the most significant byte of the instruction is stored first.

Historically, the virtual machine's space overlapped the interpreters', and as such, programs were usually loaded from address 0x200 instead. Nowadays, modern emulators use this space to store font data as they run in a separate memory space to the code they actually run. Also, the uppermost 256 bytes (addresses 0xF00-0xFFF) were used for display data and bellow that (addresses 0xEA0-0xEFF) were reserved by the interpreter for its' own internal use such as the call stack and other variables.

Registers

The machine also hosts 16 general use registers, adnotated V0 to VF. The register VF should not be used in normal circumstances as it is used as a flag in some operations and is also set whenever an collision occurs in the drawing of the sprites.

Display

The resolution of the display is 64x32 with a monochrome color. Sprites are drawn to the screen by XOR-ing themselves to the previously drawn data, where, the flag VF is set to 1 in the case of a collision (both pixels are set), like so:

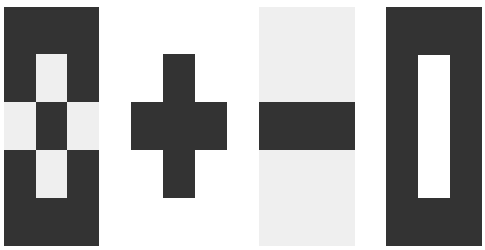

Table 3.1: Illustration of sprite XOR-ing, transforming an 8 (left) into a 0 (right), VF is set in this case to 1

The machine also has two timers, called the delay and sound timer, respectivelly. They both tick down at a rate of 60hz, and when the sound timer is non-zero, a sound is played. The specifics of the sound are not specifically defined, so it is left up to the developer of the emulator to implement however they want, the most preffered way of implementing it is by making it be a buzzer.

Input

Input is done with a 16-key keyboard, with keys having values from 0 to F, an usual input mapping to modern hardware being done like so:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

Table 3.2: Original input layout on the COSMAC VIP

1	2	3	4
Q	W	E	R
A	S	D	F
Z	X	C	V

Table 3.3: Commonly used layout on modern keyboards

Instructions

Opcode	Mnemonic	Description
00E0	CLS	Clear the display.
00EE	RET	Return from a subroutine.
1NNN	JP addr	Jump to location NNN.
2NNN	CALL addr	Call subroutine at NNN.
3XNN	SE Vx, byte	Skip next instruction if Vx is equal to NN.
4XNN	SNE Vx, byte	Skip next instruction if Vx is not equal to NN.
5XY0	SE Vx, Vy	Skip next instruction if Vx is equal to Vy.
6XNN	LD Vx, byte	Set Vx = NN.
7XNN	ADD Vx, byte	Set Vx = Vx + NN.
8XY0	LD Vx, Vy	Set Vx = Vy.
8XY1	OR Vx, Vy	Set Vx = Vx OR Vy.
8XY2	AND Vx, Vy	Set Vx = Vx AND Vy.
8XY3	XOR Vx, Vy	Set Vx = Vx XOR Vy.
8XY4	ADD Vx, Vy	Set Vx = Vx + Vy, set VF = carry.
8XY5	SUB Vx, Vy	Set Vx = Vx - Vy, set VF = NOT borrow.
8XY6	SHR Vx , Vy	Set Vx = Vx SHR 1.
8XY7	SUBN Vx, Vy	Set Vx = Vy - Vx, set VF = NOT borrow.

Opcode	Mnemonic	Description
8XYE	SHL Vx , Vy	Set Vx = Vx SHL 1.
9XY0	SNE Vx, Vy	Skip next instruction if Vx is not equal to Vy.
ANNN	LD I, addr	Set I = NNN.
BNNN	JP V0, addr	Jump to location NNN + V0.
CXNN	RND Vx, byte	Set Vx = random byte AND NN.
DXYN	DRW Vx, Vy, nibble	Display n-byte sprite starting at memory location I at (Vx, Vy), set VF = collision.
EX9E	SKP Vx	Skip next instruction if key with the value of Vx is pressed.
EXA1	SKNP Vx	Skip next instruction if key with the value of Vx is not pressed.
FX07	LD Vx, DT	Set Vx = delay timer value.
FX0A	LD Vx, K	Wait for a key press, store the value of the key in Vx.
FX15	LD DT, Vx	Set delay timer = Vx.
FX18	LD ST, Vx	Set sound timer = Vx.
FX1E	ADD I, Vx	Set I = I + Vx.
FX29	LD F, Vx	Set I = location of sprite for digit Vx.
FX33	LD B, Vx	Store BCD representation of Vx in memory locations I, I+1, and I+2.
FX55	LD [I], Vx	Store registers V0 through Vx in memory starting at location I.
FX65	LD Vx, [I]	Read registers V0 through Vx from memory starting at location I.

The original CHIP8 implementation featured 35 instructions, of which only 31 were initially documented in the specification as opcodes 8XY3, 8XY6, 8XY7 and 8XYE were not mentioned.

This is probably because all of the 8000 opcodes were part of the original CPU, the 1802's Arithmetic Logic Unit and thus were not intended to be part of the instruction set as they were not part of the interpreter itself.

Future extensions will add and modify the way in which these functions operate, such as in the case of instruction FX55 and FX65 which modify the I register by incrementing it for each copied byte, while the SCHIP extensions does not do that.

3.3 Emulator implementation

The main CHIP8 library, cchip8, has been split in several modules: the state, the interpreter and the static compiler. Only the runners require the state, nothing else depends on eachother.

The state is responsible for handling the way in which the executor is supposed to communicate with the outside by storing fucntion pointers to user procedures and other miscellanious data, such as the registers.

```
1  struct chip8_state
2  {
3      bool draw_flag;
4      ... mode;
5      uint16_t pc, i, stack[0x100];
6      uint8_t display_width, display_height;
7      uint8_t v[0x10], sp, dt, st, last_key;
8      uint16_t lowres_font_address, hires_font_address;
9
10     // Callbacks.
11     chip8_read_b_f read_b;
12     chip8_read_w_f read_w;
13     chip8_write_b_f write_b;
14     chip8_draw_sprite_f draw_sprite;
15     chip8_clear_f clear_screen;
16     chip8_key_status_f get_key_status;
17     chip8_random_f get_random;
18     chip8_resize_f resize;
19     chip8_scroll_f scroll;
20
21     void* aux_arg;  // Used for passing data to the callbacks.
22 };
```

3.3.1 Interpreter

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

3.3.2 Just-in-time compiler

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

3.4 Quirks and extensions

3.4.1 Modern Super CHIP8

Figure 3.2: Octogon by John Earnest running in Edra's modern SCHIP mode.

3.5 Testing

4 The final application

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nos-

trud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

4.1 Design

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

4.2 Implementation

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

4.3 Usage

4.3.1 Command line

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

4.3.2 GUI

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

5 Conclusions

Concluzii ...

Bibliography

[1] N. Cruz, M. R. Ferrández, J. Redondo, and J. Alvarez. Applications of chip-8, a virtual machine from the late seventies, in current degrees in computer engineering. *11th International Conference on Education and New Learning Technologies, Palma de Mallorca, Spain*, 2019. doi:10.21125/edulearn.2019.0501.

[2] O. Mazonka and D. Cristofani. A very short self-interpreter. https://www.researchgate.net/publication/1881452_A_Very_Short_Self-Interpreter, 12 2003. Online; accessed 04 April 2025.

[3] S. Morr. Esoteric programming languages: An introduction to brainfuck, intercal, befunge, malbolge, and shakespeare. https://blinry.org/esolangs/esolangs.pdf. Online; accessed 04 April 2025.

[4] J. Sang-Woo. 50,000,000,000 instructions per second: Design and implementation of a 256-core brainfuck computer. https://people.csail.mit.edu/wjun/papers/sigtbd16.pdf. Online; accessed 04 April 2025.

[5] D. Temkin. Language without code: Intentionally unusable, uncomputable, or conceptual programming languages. *Journal of Science and Technology of the Arts*, 9:83, 12 2017. doi:10.7559/citarj.v9i3.432.

[6] L. Xiaoting, L. Xiao, C. Lingwei, P. Rupesh, and W. Dinghao. Alphaprog: Reinforcement generation of valid programs for compiler fuzzing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. doi:10.1609/aaai.v36i11.21527.