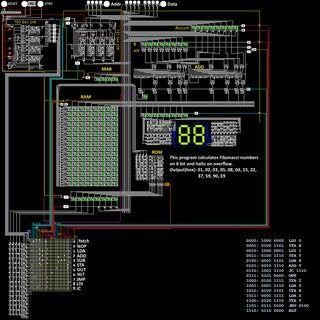
Simple Lang Compiler

## Programmed to work with 8 Bit CPU

TABLE OF CONENTS:-

* Introduction
* Language Declaration
* Importance of 8-Bit CPU Compiler
* Features of 8-bit CPU Compiler
* Architecture Support
* Key Decisions
* Design a Simple High-Level Language (SimpleLang)
* Code Optimization Techniques
* Debugging and Testing
* Using the compiler
* Future Trends and Innovations
* Conclusion

## INTRODUCTION

* SimpleLang is a minimalistic high-level language designed to run on an 8-bit CPU. It includes basic constructs such as variable declarations, assignments, arithmetic operations, and conditional statements, but it does not include loops. This language aims to be easy to understand and implement for educational purposes.
* This typically allows for the total of 256 different values to be represented from 0 to 255.
* A compiler is a computer program that helps in translating the computer code from one programming language into another language. Basically, it translates the program written in the source language to the machine language.

### Language Declaration:-

**Variable Declarations:**

**Assignments:**

Syntax: int variable\_name;

Syntax: variable\_name = expression; Example: int x;

Example: x = 5;

Semantics: Declare a variable of type int. The variable is initialized to 0 by default.Semantics: Assign the result of the expression to the variable.

**Arithmetic Operations: Conditionals:**

Syntax: variable\_name = expression;

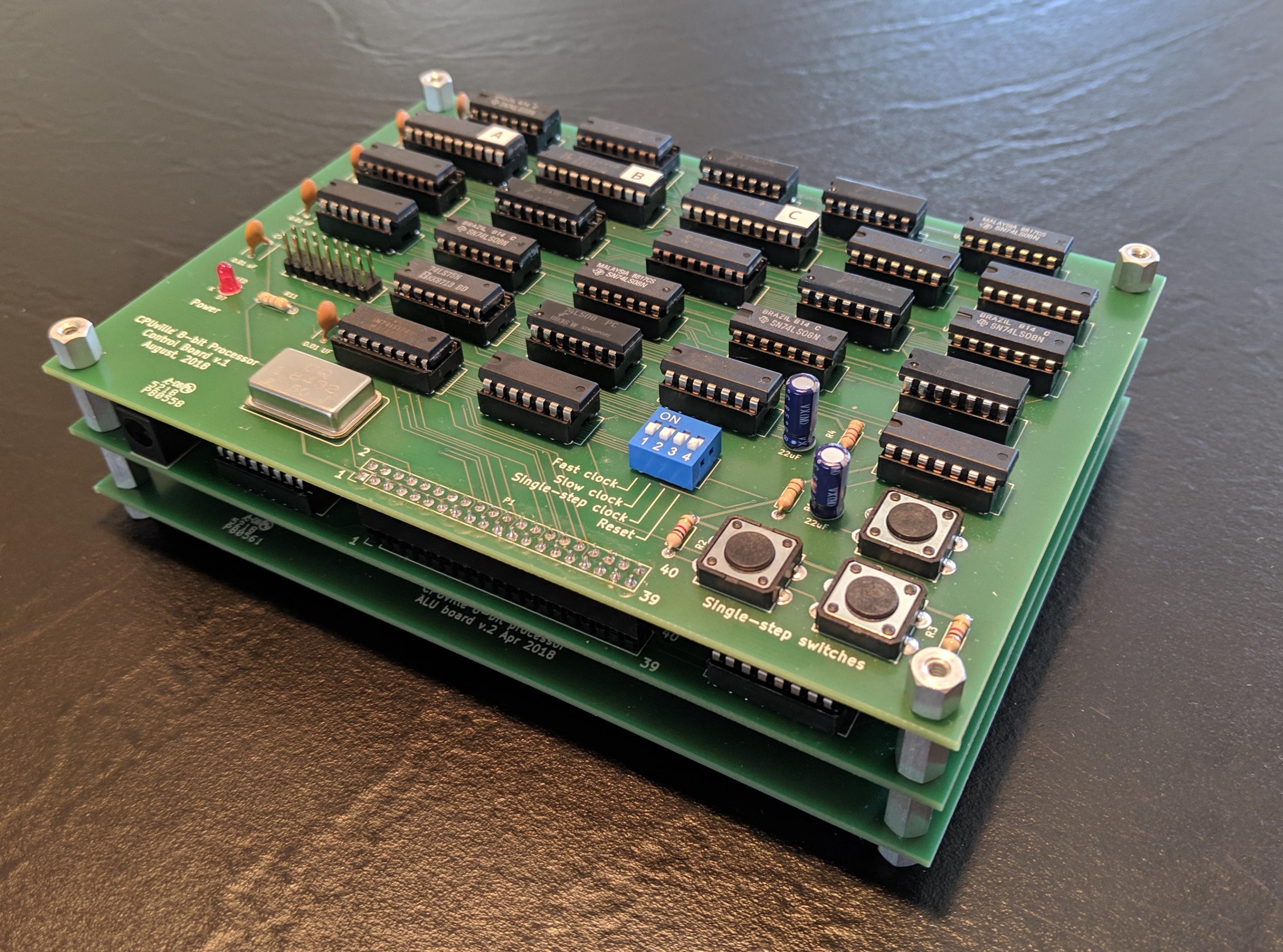
Syntax: if (expression) { statements }

Example: x = a + b;

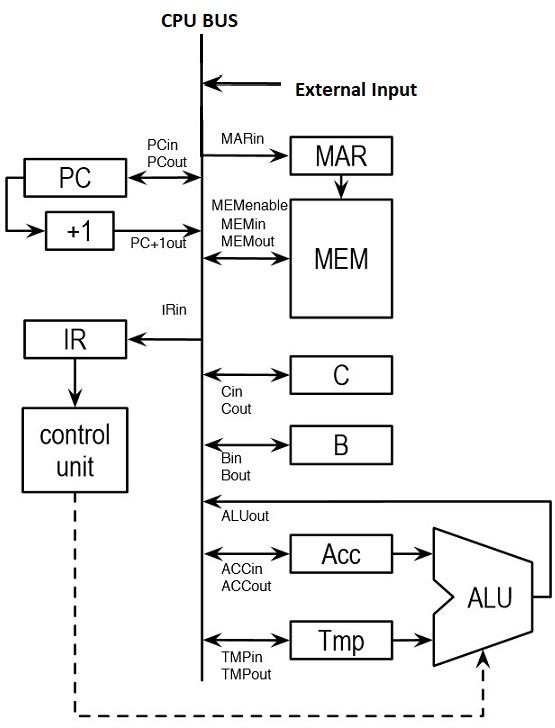
Example: if (a==b) { a -10 = b } Supported Operations: +, -, \*, /

**Print:**

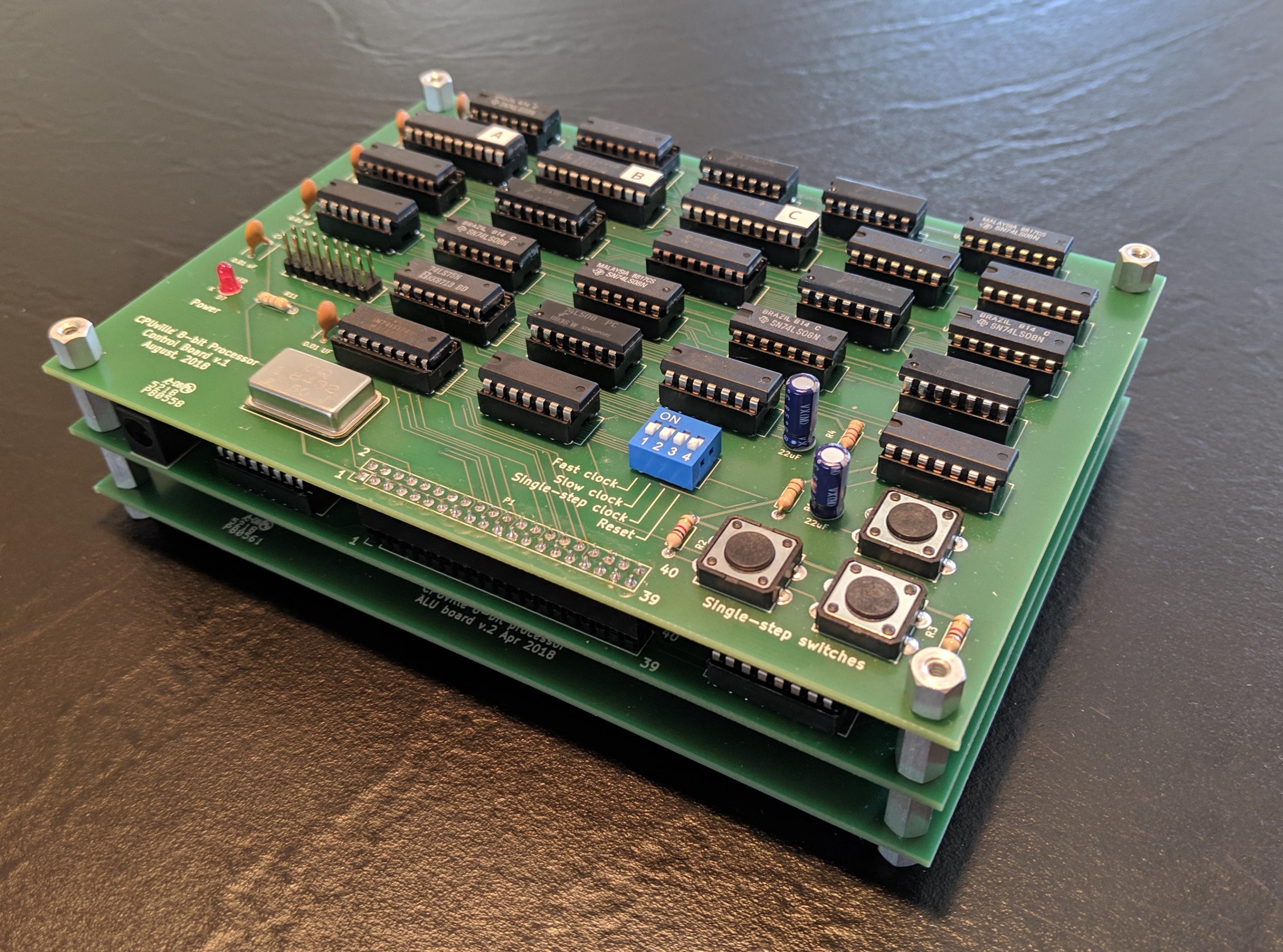
# Importance of 8-bit CPU Compilers:-

* 8-bit CPUs are commonly used in embedded systems, IoT devices, and retro computing projects.
* O p t i m i z e d c o m p i l e r s c a n h e l p maximize t he performance and efficiency of these systems.
* Efficient code generation is crucial for m i n i m i z i n g m e m o r y u s a g e a n d maximizing battery life in resource- constrained devices.

# Features of 8-bit CPU Compiler:-

* + Support for low-level programming constructs such as bit manipulation and direct memory access.
  + Optimization techniques tailored to the constraints of 8-bit architectures, such as code size and execution speed.
  + I n t e g r a t i o n w i t h d e v e l o p m e n t environments and toolchains specific to 8-bit processors.

# Architecture Support:-

* 8-bit CPU compilers are available for a variety of architectures, including popular ones like AVR, PIC, and 6502.
* Each architecture may have its own set of instructions and memory models that the compiler must support.
* Cross- compilation t ools enable developers to write code on a more powerful machine and compile it for the target 8-bit architecture.

# Key Decisions:-

* + Struct-Based Approach: Using structures (Variable, Compiler) to encapsulate related data and operations, promoting clarity and maintainability.
  + Assembly Output Management: Using fixed-size arrays (output, data) to store assembly instructions and data declarations, ensuring simplicity and straightforward management.
  + Error Handling: Basic error handling through standard error output (fprintf(stderr, ...)), mainly for unsupported conditions or expressions.
  + Flexibility and Limitations: The compiler supports a defined set of language features (variable declarations, assignments, conditionals, and printing), adhering strictly to specified syntax and semantics.
  + Efficiency Considerations: While the compiler is straightforward and suitable for educational or small-scale projects, optimizations for performance or advanced features (like functions, loops, or type checking) are not implemented in this version.

# Design a Simple High-Level Language (SimpleLang):-

* Define Syntax and Semantics Define constructs for:
* Variable Declarations: int <var>;
* Assignments: <var> = <expression>;
* Arithmetic Operations: +, -
* Conditionals: if (<condition>) { <statements> }
* Document Language Constructs Example:

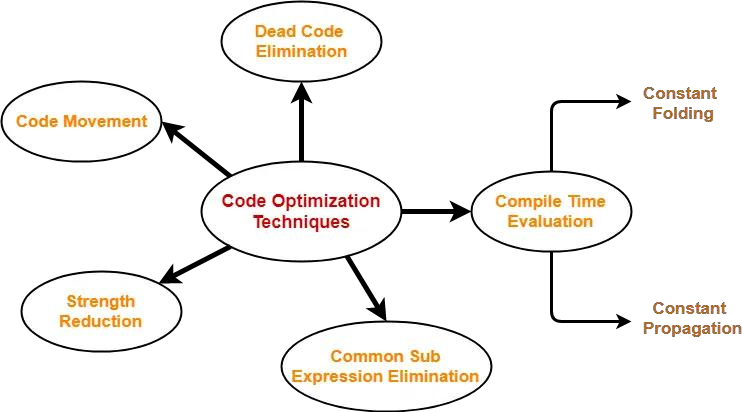
int a;

a = 5 + 3;

if (a == 8) { a = a - 1;

}

# Code Optimization Techniques:-

* Common optimization techniques for 8-bit CPU compilers include loop unrolling, inline functions, and constant propagation.
* Register allocation strategies are crucial for minimizing the number of memory accesses and improving performance.
* Tailoring optimization passes to the specific characteristics of 8 - bit architectures can lead to significant code improvements.

# Debugging and Testing:-

* Debugging tools integrated into the compiler environment can help developers identify and resolve issues in their code.
* Simulation tools allow for testing code on a virtual 8-bit CPU before deploying it to the actual hardware.
* Emulation platforms provide a more accurate representation of the target architecture for testing and debugging purposes.

### Using the compiler:-

First we call the compiler with the source code. By ./scc input.sl > output.asm

We get output which is assembly,

We can use the assembler given with the 8-bit computer to further compile code into machine code

Once we have the machine code, the assembler automatically loads it into the cpu memory,

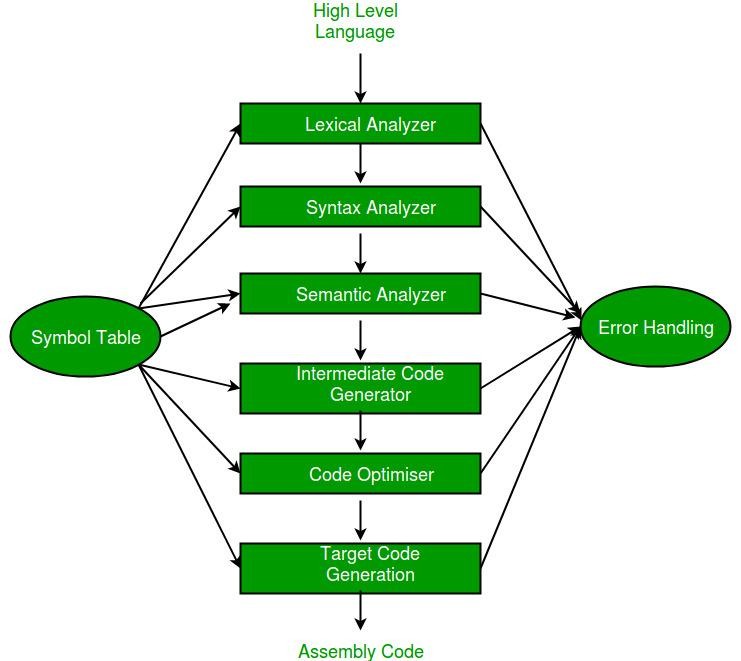
Which is the memory.list file,

Finally we can make build and make run, to get output from our 8-bit Computer.

# Future Trends and Innovations:-

* Continued advancements in compiler technology are likely to further optimize code generation for 8-bit CPUs.
* Integration of machine learning techniques could enable more intelligent optimization strategies tailored to specific applications.
* Support for emerging 8-bit architectures and new features will drive innovation in the development of 8-bit CPU compilers.

# Conclusion:-

* 8-bit CPU compilers play a crucial role in enabling efficient software development for embedded systems and other applications.
* By leveraging optimization techniques and tailored features, developers can maximize the performance of 8-bit CPUs.
* Continued collaboration and innovation in the field of 8-bit CPU compilers will drive progress and expand the capabilities of these legacy processors.

Thank-you

The link to the project, [https://github.com/Sireeshanarayansetti/r](https://github.com/jvsiddharth/compiler-8bitasm)ecruit-vicharak.git,

As instructed the repository is private and "recruit-vicharak" has been invited, Hopefully everything is to your liking,

Please do contact me for any further discussion, Have a good day!

Best Regards, Narayanasetti Sireesha [nsireesha2004@gmail.com](mailto:b4ujasiddhu@gmail.com)