

# **Ghulam Ishaq Khan Institute of Engineering Sciences and Technology**



## **COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE** **(CE322)**

Team Members

Muazzam Shah

2022312

# 16-Bit Processor Simulator

## Introduction

The 16-bit processor simulator is a feature-rich educational tool designed for understanding and exploring processor operations, including memory manipulation, register-based computations, and input/output functionalities. Built using Python and PyQt5, the simulator offers an interactive GUI with support for real-time instruction execution, animations, and file-based program management.

## Features of the Simulator

### 1. Memory:

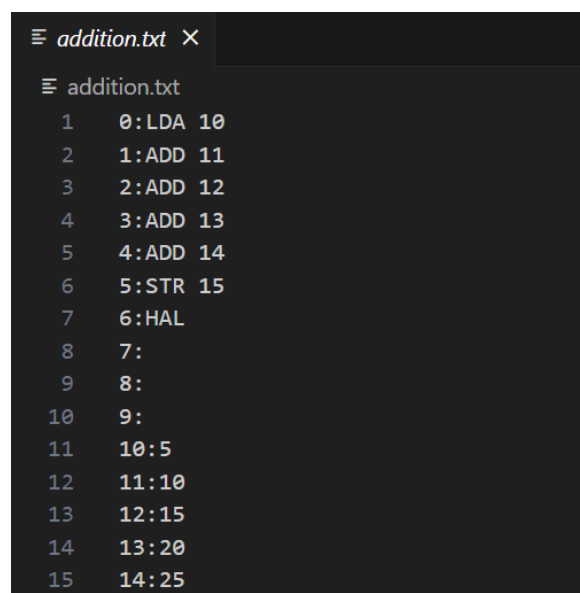
The architecture of this simulator is designed to support up to 2,096 memory addresses, with each address capable of holding a 16-bit word. However, to maintain simplicity and focus on the core functionality of the simulator, the memory has been scaled down to 32 addressable locations. This smaller memory size is sufficient to effectively demonstrate the simulator's operations, including instruction execution, data storage, and manipulation. By limiting the memory, the simulator ensures a manageable environment for users to learn and experiment with the processor's features while retaining the essence of a larger architecture.

- **32 Addressable Memory Cells:**

- Each memory cell can store instructions or data in the form of mnemonics
- Each cell can individually be edited

- **File-Based Program Storage:**

- Programs and data can be saved to and loaded from .txt files
- File format:
  - *Address : Instruction*



```
addition.txt X
addition.txt
1  0:LDA 10
2  1:ADD 11
3  2:ADD 12
4  3:ADD 13
5  4:ADD 14
6  5:STR 15
7  6:HAL
8  7:
9  8:
10 9:
11 10:5
12 11:10
13 12:15
14 13:20
15 14:25
```

- **Memory Visualization:**
  - Users can edit memory cells directly in the GUI.
  - Animations highlight data transfer between memory and registers.

## 2. Central Processing Unit (CPU)

The simulator's CPU incorporates essential registers that drive its functionality and emulate a real processor's operations. These registers work together to facilitate arithmetic computations, memory access, and control flow within the simulated environment. The following registers form the core components of the simulator's CPU:

- **Accumulator (AC):** Serves as the primary register for arithmetic and logical operations, storing intermediate results and acting as the central hub for processing data.
- **Program Counter (PC):** Keeps track of the memory address of the next instruction to be executed, ensuring smooth sequential execution or branching based on control instructions.
- **Instruction Register (IR):** Temporarily holds the instruction fetched from memory, preparing it for decoding and execution.
- **Address Register (AR):** Stores the address of the memory location being accessed, enabling precise control during data retrieval and storage operations.
- **Flag Register (E):** Manages special conditions such as overflow and carry, ensuring correct handling of operations that exceed standard limits.

## 3. Instruction Set Architecture

- **Mnemonic to Binary Conversion:** Each mnemonic has a corresponding binary opcode. By enabling a checkbox in the simulator, users can view the binary equivalent of each instruction, bridging the gap between human-readable commands and machine-level code.
- **16-Bit Instruction Word:**
  - Bits 0–10: Address of the operand.
  - Bits 11–14: Opcode.
  - Bit 15 (I): Addressing mode (0 for direct, 1 for indirect).

## 4. Addressing Modes

- **Direct Addressing:** The operand directly specifies the memory location.
- **Indirect Addressing:** The operand refers to a memory location that contains the address. Indicated by the "I" bit in instructions. Example: LDA I 15. This loads data indirectly from the memory address stored at location 15.

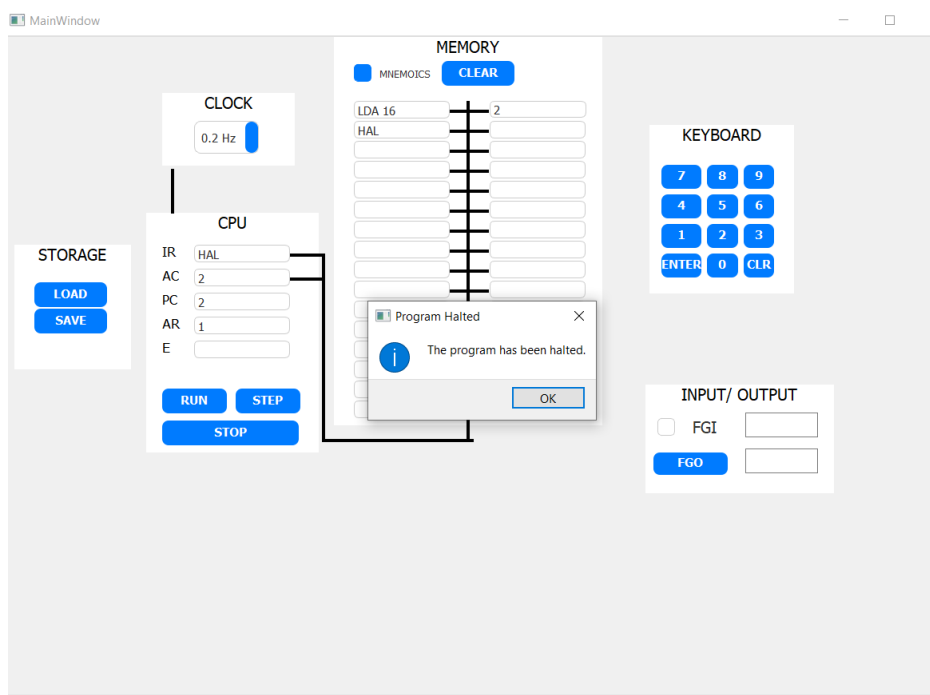
## 5. Input/Output (I/O) System

The simulator features a robust I/O system with keyboard integration, enabling real-time input for testing interrupt-driven operations. Users can provide input directly via the keyboard, while outputs are displayed within the simulator's GUI or on the console, ensuring seamless interaction. To manage

the flow of data, the system utilizes input/output flags: the FGI (Input Flag) indicates when the system is ready to receive input, and the FGO (Output Flag) signals when output is ready to be processed. This design ensures efficient handling of I/O operations and provides a realistic simulation of interrupt-based workflows.

## 6. Execution Control

- **Run, Step, and Stop:**
  - Execute programs continuously or step through instructions.
  - Stop execution at any point.
- **Interactive Popups:**
  - Notifications for program halts (HAL) or critical operations.

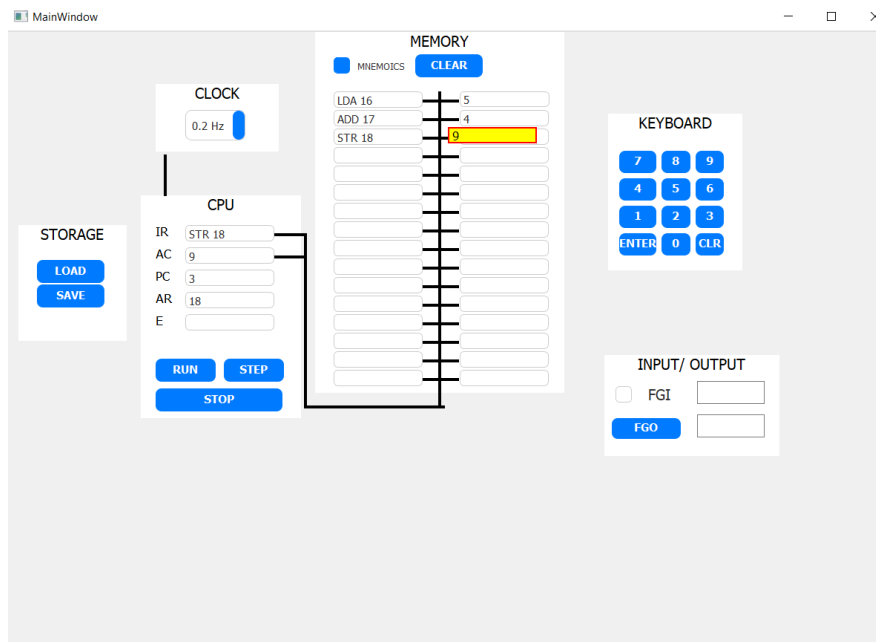


## 7. Arithmetic and Logical Operations

The simulator supports an extensive range of arithmetic and logical operations, providing the flexibility needed to perform complex computations. Key operations include addition, subtraction, and multiplication, as well as logical operations such as AND, OR, and XOR. These operations are carried out using the Accumulator (AC), which serves as the central register for data manipulation. Additionally, the simulator incorporates robust handling for overflow conditions. When a calculation exceeds the capacity of the 16-bit word, the Flag Register (E) is updated to reflect the overflow, ensuring accurate results and enabling users to account for exceptional scenarios during computation. This comprehensive support for arithmetic and logic makes the simulator a practical and versatile tool for exploring processor functionality.

## 8. Visual Animations

- Animations depict data movement between memory and registers:
  - Memory-to-AC operations.
  - AC-to-memory operations.
  - Instruction fetching to IR.



## 9. Error Handling

The simulator is equipped with robust error-handling mechanisms to ensure smooth operation and provide valuable feedback during execution. It includes comprehensive checks to prevent invalid memory access or attempts to execute unsupported operations, safeguarding the system from unexpected crashes. When errors occur, the simulator generates clear and descriptive error messages, accompanied by logs to assist users in identifying and debugging issues. This feature enhances the reliability of the simulator while providing an educational platform for understanding common pitfalls in processor operations.

---

## Working of the Simulator

### Program Workflow

1. **Load Program:**
  - Programs can be entered manually or loaded from .txt files.
2. **Fetch-Decode-Execute Cycle:**
  - The simulator fetches the next instruction using the PC.

- Decodes the operation based on opcode and addressing mode.
- Executes the instruction, updating registers, memory, or both.

### Instruction Execution

- Execution progresses one instruction at a time, with visual and log-based feedback.
- Users can switch between step and run modes dynamically.

### I/O Interrupts

- Input is handled via keyboard with flag updates for readiness.
- Outputs are processed directly to a specified device or console.

### Halting

- The HAL instruction stops the program gracefully with a popup notification.

## Instruction Set Summary

### Memory reference instructions

Opcode	Mnemonic	Description
0000	LDA	Load data from memory to AC
0001	STR	Store data from a register to memory
0010	JMP	Jump to a specified memory location
0011	JZE	Jump if the AC is Zero
0100	JSA	Jump and save return address
0101	AND	Perform bitwise AND operation
0110	OR	Perform bitwise OR operation
0111	XOR	Perform bitwise XOR operation
1000	ADD	Add values in AC
1001	SUB	Subtract values in AC
1010	MUL	Multiply values in registers
1011	INC	Increment and skip if zero
1100	DEC	Decrement and skip if zero

### Register reference instructions

Binary code	Mnemonic	Description
01111 10000000000	CRA	Clear register AC
01111 01000000000	CRE	Clear flag E
01111 00100000000	CTA	Complement AC
01111 00010000000	CTE	Complement E
01111 00001000000	CPA	Compare accumulator with a value
01111 00000100000	INA	Increment AC
01111 00000010000	SKP	Skip if AC > 0
01111 00000001000	SKN	Skip if AC
01111 00000000100	CRA	Circulate right Shift AC and E
01111 00000000010	CLA	Circulate left Shift AC and E
01111 00000000001	HAL	Halt the processor.

### Input/Output Instructions

Opcode	Mnemonic	Description
11111 10000000000	INP	Input character to AC
11111 01000000000	OUT	Output character from AC
11111 00100000000	SFI	Skip on input flag
11111 00010000000	SFO	Skip on output flag
11111 00001000000	PUT	Output data to a specified I/O device.
11111 00000100000	OPT	Perform an operation on the I/O device.
11111 00000010000	SPI	Send data to the serial peripheral interface.
11111 00000001000	SPO	Receive data from the serial peripheral interface.
11111 00000000100	SIE	Set/Reset interrupt enable flag.

### Conclusion

The 16-bit processor simulator is a comprehensive tool for learning and prototyping processor operations. Its support for mnemonic-to-binary conversion, addressing modes, visualizations, and a flexible GUI makes it suitable for a wide range of use cases, including educational purposes and low-level system design experimentation.