Project 1

Luca Guerini and Tijana Minic

Seton Hall University

Introduction

- `Components:`

Introduction

- `Components`:
- Registers: AF, BC, DE, HL, PC, SP

Introduction

- `Components`:
- Registers: AF, BC, DE, HL, PC, SP
- Instruction Execution: Decoding and execution of Z80 instructions

Registers and Flags

- `Registers`
  - → AF, BC, DE, HL: 16-bit registers for various operations.
  - → PC (Program Counter): Keeps track of the instruction being executed.
  - → SP (Stack Pointer): Manages program stack.

Registers and Flags

- `Registers`
  - → AF, BC, DE, HL: 16-bit registers for various operations.
  - → PC (Program Counter): Keeps track of the instruction being executed.
  - → SP (Stack Pointer): Manages program stack.
- `Flags`:
  - → c (Carry)
  - → h (Half Carry)
  - → n (Add/Subtract)
  - → z (Zero)

Implementation details

- `Data Structures:`
  - → Registers Class: Implemented using data classes and mutable mapping.
  - → Opcode Parsing: Utilizes `opcode-parser` for instruction decoding.

Implementation details

- `Data Structures:`
  - → Registers Class: Implemented using data classes and mutable mapping.
  - → Opcode Parsing: Utilizes `opcode-parser` for instruction decoding.
- `Instruction Decoding`:
  - → Execute Method: Executes Z80 instructions based on opcode patterns.
  - → Error Handling: Raises InstructionError for unsupported instructions.

IExecution Process

- `Decode and Execute Loop:`
  - → Memory Addressing: Retrieves instructions from memory using program counter.
  - → Decoding: Uses the decoder to interpret opcodes and fetch corresponding instructions.
  - → Execution: Modifies CPU state based on the decoded instruction.

IExecution Process

- `Decode and Execute Loop:`
  - → Memory Addressing: Retrieves instructions from memory using program counter.
  - → Decoding: Uses the decoder to interpret opcodes and fetch corresponding instructions.
  - → Execution: Modifies CPU state based on the decoded instruction.
- `Continual Loop`: Runs indefinitely, simulating continuous instruction execution.