Basic CPU Components

Objective:

Implement the fundamental components of a CPU, including the ALU, general-purpose registers, and control mechanisms like the program counter and instruction register.

Tasks and Implementation Steps:

1. Build the ALU (Arithmetic Logic Unit) :

The ALU performs arithmetic and logical operations as dictated by the CPU instructions.

Key Functions:

- Addition, subtraction, multiplication, and division.
- Logical operations (AND, OR, XOR, NOT).
- Comparisons (greater than, less than, equal to).

Python Code for ALU:

```
class ALU:
    def operate(self, opcode, operand1, operand2=None):
        if opcode == "ADD":
            return operand1 + operand2
        elif opcode == "SUB":
            return operand1 - operand2
        elif opcode == "MUL":
            return operand1 * operand2
        elif opcode == "DIV":
            return operand1 // operand2 if operand2 != 0
        elif opcode == "AND":
            return operand1 & operand2
        elif opcode == "OR":
            return operand1 | operand2
        elif opcode == "XOR":
            return operand1 ^ operand2
        elif opcode == "NOT":
            return ~operand1
        elif opcode == "CMP":
            return operand1 == operand2
        else:
            raise ValueError(f"Invalid opcode: {opcode}")
```

2. Implement General-Purpose Registers

Registers store temporary data used by the CPU during execution. Each register is identified by a unique name (e.g., `R0`, `R1`).

- **Key Features**:
  - Read and write functionality.
  - A fixed number of registers (e.g., 8 general-purpose registers).

Python Code for Registers:

```
class Registers:
    def __init__(self, num_registers=8):
        self.registers = [0] * num_registers

    def read(self, register_index):
        return self.registers[register_index]

    def write(self, register_index, value):
        self.registers[register_index] = value
```

3. Create the Program Counter and Instruction Register:

These components manage the flow of program execution.

- **Program Counter (PC)**:
  - Points to the memory address of the next instruction to execute.
  - Increments after each instruction fetch.
- **Instruction Register (IR)**:
  - Stores the current instruction being executed.

Python Code for PC and IR:

```python
class ControlUnit:
    def __init__(self):
        self.program_counter = 0
        self.instruction_register = None

    def fetch(self, memory):
        self.instruction_register = memory[self.program_counter]
        self.program_counter += 1

    def get_instruction(self):
        return self.instruction_register

    def set_program_counter(self, value):
        self.program_counter = value
```

How It Works Together

1. ALU handles all computation tasks like arithmetic and logical operations.
2. Registers store data required for operations and instructions.
3. Program Counter (PC) ensures instructions are fetched in sequence.
4. Instruction Register (IR) holds the current instruction for decoding and execution.

These components form the backbone of a basic CPU, enabling instruction fetching, execution, and data processing.