Simulator Description

This is the simulator for the given ISA. The input to the simulator is a binary file (the format is the same as the format of the binary file generated by the assembler).

The simulator is having the following distinct components:

1. Memory (MEM): MEM takes in an 7-bit address and returns a 16-bit value as the data. The MEM stores 256 bytes, initialized to 0s.
2. Program Counter (PC): The PC is a 7-bit register which points to the current instruction.
3. Register File (RF): The RF takes in the register name (R0, R1, ..., R6, or FLAGS) and returns the value stored at that register.
4. Execution Engine (EE): The EE takes the address of the instruction from the PC, uses it to get the stored instruction from MEM, and executes the instruction by updating the RF and PC.

The simulator is loading the binary in the system memory at the beginning, and then it start executing the code at address 0. The code is executed until `hlt` is reached. After execution of each instruction, the simulator outputs one line containing a 7-bit number denoting the program counter. This is followed by 8 space-separated 16-bit binary numbers denoting the values of the registers (R0, R1, ..., R6, and FLAGS).

```
<PC (7 bits)> <R0 (16 bits)> ...  <R6 (16 bits)> <FLAGS (16 bits)>
```

The output is written to STDOUT. Similarly, the input must be read from assembly_output.txt.

After the program is halted, print the memory dump of the whole memory. This should be 128 lines, each having a 16-bit value.

```
<16 bit data>
<16 bit data>
...
<16 bit data>
```

Simulator Pseudocode

```
initialize(MEM); // Load memory from stdin
PC = 0; // Start from the first instruction
halted = false;

while(not halted)
{
    Instruction = MEM.fetchData(PC); // Get current instruction
```

```
    halted, new_PC = EE.execute(Instruction); // Update RF and compute new_PC
    PC.dump(); // Print PC
    RF.dump(); // Print RF state
    PC.update(new_PC); // Update PC
}

MEM.dump() // Print the complete memory
```