Name: Shorena K. Anzhilov
Email: anzhilov.s@northeastern.edu
Spring, 2025

1. **Instruction Address:** 0x300. **Instruction:** `PUSH 0x800`
    a. The stack pointer (SP) moves from `0x118` to `0x114` (decreasing by 4 bytes).
    b. The value `0x800` is stored at memory address `0x114`.
    c. The program counter (PC) changes to `0x304` for the next instruction.
2. **Instruction Address:** 0x304. **Instruction:** `PUSH *(0x804)`
    a. The stack pointer (SP) moves from `0x114` to `0x110`.
    b. The value at address `0x804` (which is `200`) is pushed onto the stack at address `0x110`.
    c. The program counter (PC) updates to `0x308`.
3. **Instruction Address:** 0x308. **Instruction:** `CALL 0x400`
    a. The instruction pushes the return address `0x30C` onto the stack, moving SP to `0x10C`.
    b. The program counter (PC) jumps to `0x400`.
4. **Instruction Address:** 0x400. **Instruction:** `MOV *(SP+8) → EAX`
    a. The value at `0x114` (which is `0x800`) is moved into the EAX register.
    b. The program counter (PC) moves to `0x404`.
5. **Instruction Address:** 0x404. **Instruction:** `MOV SP → *EAX`
    a. The value of SP (`0x10C`) is stored at address `0x800`, saving the old stack pointer.
    b. The program counter (PC) moves to `0x408`.
6. **Instruction Address:** 0x408. **Instruction:** `MOV *(SP+4) → EAX`
    a. The value at `0x110` (which is `200`) is moved into EAX.
    b. The program counter (PC) moves to `0x40C`.
7. **Instruction Address:** 0x40C. **Instruction:** `MOV EAX → SP`
    a. The stack pointer (SP) is updated to `0x200` (switching to the new stack).
    b. The program counter (PC) moves to `0x410`.
8. **Instruction Address:** 0x410. **Instruction:** `RET`
    a. The return address `0x30C` is popped from the stack and loaded into the program counter (PC).
    b. The stack pointer (SP) updates to `0x204`.
    c. The program counter (PC) goes back to `0x30C`.
9. **Instruction Address:** 0x30C. **Instruction:** `ADD 8 → SP`
    a. The stack pointer (SP) increases by 8, changing from `0x204` to `0x20C`.
    b. The program counter (PC) moves to `0x500`.
10. **Instruction Address:** 0x500. **Instruction:** `POP EAX`
    a. The value `50` is popped from the stack and stored in EAX.
    b. The stack pointer (SP) updates from `0x20C` to `0x208`.
    c. The program counter (PC) moves to `0x504`.
11. **Instruction Address:** 0x504. **Instruction:** `POP EBX`
    a. The value `70` is popped from the stack and stored in EBX.
    b. The stack pointer (SP) updates from `0x208` to `0x20C`.
    c. Execution reaches the final point marked as `DONE`.