CS 13 Virtual Machine Project

Implement the a virtual machine for the instructions listed in the chart on page two
Your VM class should have the following fields:

Field	Description	Function
pc	Program Counter; type int	Contains the address of the next instruction to fetch
memory	Array of type int	Contains instructions (a program) and data
stack	Operand and CPU stack	Used for processing arithmetic instructions and storing return address

The pc should be initialized to 0 so the first instruction is fetched from memory location zero. The memory should contain a sample machine language program. This program could be loaded from a text file, or you can directly initialize the memory array when it is created. The stack is the integer stack discussed in part 1 of this assignment. The stack should be properly initialized by its constructor.

To execute the program in memory you will need a method that:
1. Fetches the instruction at the address contained in the pc
2. Increments the pc
3. Decodes the instruction. This is done with a switch statement.
4. Executes the instruction.
5. Repeats this process until a HALT is executed.

Instruction	Opcode	Description	Operation
CONST n	8	Load constant	push(n)

LOAD a	9	Load variable from address a	d = memory[a]
			push(d)

STO a	10	Store variable in address a	d = pop()
			memory[a] = d

ADD	11	Add top two operands and push the result	y = pop()
			x = pop()
			d = x + y
			push(d)

SUB	12	Subtract top two operands and push the result	y = pop()
			x = pop()
			d = x - y
			push(d)

MUL	13	Multiply top two operands and push the result	y = pop()
			x = pop()
			d = x * y
			push(d)

DIV	14	Divide top two operands and push the result	y = pop()
			x = pop()
			d = x / y
			push(d)

EQL	15	Check if equal	y = pop()
			x = pop()
			if x == y push(1) else push(0)

LSS	16	Check if less	y = pop()
			x = pop()
			if x < y push(1) else push(0)

GTR	17	Check if greater	y = pop()
			x = pop()
			if x > y push(1) else push(0)

JMP a	18	Jump to address a	pc = a
FJMP a	19	Jump if false	if (pop() == 0) pc = a
READ	20	Read an integer	d = nextInt(); push(d)
WRITE	21	Write an integer	println(pop())
CALL a	22	Call subprogram	push(pc + 1); pc = a
RET	23	Return from subprogram	pc = pop()
HALT	0	Terminate execution	System.exit(0);

Virtual Machine Test Program

Directions

Assemble this program by hand and use code for testing in your virtual machine.

Test Program

```
        read
        sto     num1
        read
        sto     num2

        load    num1
        load    num2
        mul

        const   6

        add

        sto temp0

        load    temp0
        const   20
        gtr
        fjmp done

        load    temp0
        const   4
        sub
        sto     num3


done:
        halt
```

Address	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	