Stack-based JOI-Virtual Machine Specification

Memory Model

1. Stack: Used for operation execution and local variables
2. Heap: Used for dynamic memory allocation
3. Code: Stores the bytecode instructions

Registers

- SP: Stack Pointer
- FP: Frame Pointer
- HP: Heap Pointer
- PC: Program Counter

Instruction Set Architecture

Stack Operations

- PUSH <value>: Push a value onto the stack
- POP: Pop a value from the stack
- DUP: Duplicate the top value on the stack
- SWAP: Swap the top two values on the stack

Arithmetic Operations

- ADD: Add the top two values on the stack
- SUB: Subtract the top value from the second top value
- MUL: Multiply the top two values on the stack
- DIV: Divide the second top value by the top value
- MOD: Modulo operation on the top two values

Comparison Operations

- EQ: Equal
- NE: Not Equal
- LT: Less Than
- LE: Less Than or Equal
- GT: Greater Than

- GE: Greater Than or Equal

Control Flow

- JMP <addr>: Unconditional jump
- JZ <addr>: Jump if zero
- JNZ <addr>: Jump if not zero
- CALL <addr>: Call a function
- RET: Return from a function

Memory Operations

- LOAD: Load a value from memory onto the stack
- STORE: Store a value from the stack into memory
- ALLOC <size>: Allocate memory on the heap
- FREE: Free allocated memory

Object-Oriented Operations

- NEW <class_id>: Create a new object
- GETFIELD <offset>: Get object field
- SETFIELD <offset>: Set object field
- VCALL <method_id>: Call a virtual method

Instruction Format

Each instruction is 32 bits long:
- 8 bits: Opcode
- 24 bits: Operand (if applicable)

Example Usage

To create an object and call a method:

```
1. NEW 1        // Create an object of class 1
2. DUP         // Duplicate the object reference
3. PUSH 10      // Push an argument
4. VCALL 2      // Call virtual method 2
5. POP         // Clean up the stack
```

To perform arithmetic:

1. PUSH 5
2. PUSH 3
3. ADD
4. PUSH 2
5. MUL
6. PUSH 7
7. SUB

This VM design provides a simple yet flexible foundation for translating object-oriented code to RISC-V assembly. The stack-based nature simplifies the translation process, while the heap allows for dynamic memory allocation needed in OOP languages.