X-machine Bytecodes API

Bytecode	***Description***	***Example***
HALT	Halt the execution of a program	HALT
POP	POP n: pop the top n levels of the runtime stack	POP 5 POP 0
FALSEBRANCH	FALSEBRANCH <label> pop the top of the stack; if it is false (0) then branch to <label> else execute the next bytecode	FALSEBRANCH xyz<<3>>
GOTO	GOTO <label>	GOTO zyx<<3>>
STORE	STORE N <id> - pop the top of the stack; store the value into the offset n from the start of the frame; <id> is used as a comment and for dumping, it's the variable name where the data is stored.	STORE 3 i STORE 2
LOAD	LOAD n <id> ; push the value in the slot which is offset n from the start of the frame onto the top of the stack; <id> is used as a comment and for dumping, it's the variable name where the data is loaded.	LOAD 3 LOAD 2 i
LIT	LIT n – load the literal value n LIT 0 i – this form of Lit was generated to load 0 on the stack to initialize the variable i to the value 0 and reserve space on the runtime stack for i.	LIT 5 LIT 0 i

ARGS	ARGS n ; Used prior to calling a function. n = # of args this instruction is immediately followed by the CALL instruction; the function has n args so ARGS n instructs the interpreter to set up a new frame n down from the top of the runtime stack.	ARGS 4 ARGS 0 ARGS 2
	This will include the arguments in the new frame for the function.	
CALL	CALL <funcname> - transfer control to the indicated function. Make sure to save where the function should return to.	CALL f CALL f<<3>> Note: CALL f and Call f<<3>> are executed in the same way.
RETURN	RETURN <funcname>; Return from the current function; <funcname> is used as a comment to indicate the current function. RETURN is generated for intrinsic functions.	RETURN f<<2>> RETURN Note: returns with labels functions EXECUTE THE same as returns without labels.
BOP	BOP <binary op> , pop top 2 levels of the stack and perform the indicated operation , operations are + - / * == != <= > >= < | & | and & are logical operators not bitwise operators. Lower level is the first operand: Eg: <second-level> + <top-level>	BOP + BOP – BOP /
READ	READ; Read an integer; prompt the user for input and push the value to the stack. Make sure the input is validated.	READ

WRITE	WRITE; Write the value of the top of the stack to output. Leave the value on the top of the stack	WRITE
LABEL	LABEL <label>; target for branches;(FALSEBRANCH, GOTO and CALL)	LABEL xyz<<3>> LABEL Read
DUMP	This bytecode is used to set the state of dumping in the virtual machine. When dump is on, after the execution of each bytecode, the state of the runtime stack is dumped to the console.	DUMP ON DUMP OFF