Technical Documentation Computron

Describe how to interact with the existing backend logic of the virtual machine (Computron VM), what functions are available, and how the user interface should interact with the VM state.

1. Overview of Computron

Computron is an educational virtual machine. The backend is fully implemented in Java and consists of interconnected classes:

Computron	GUI container, interaction: register/memory display, input-output
CvmRegisters	Storage of global VM state: PC, SP, A, X, R, RH, RL, memory M[]
CvmALUnit	Execution of a single instruction (ALU + jump logic, operations)
CvmControl	Execution mode control (step, run)
CvmDigital	Displaying registers and memory in the GUI
CvmSwitching	Switching the active register
CvmFloatTransform	Conversion between float and 2×16 bits

2. Global Model

2.1 Registers (from CvmRegisters)

Register	**Type**	**Meaning**
PC	int	Program Counter - address of the current instruction
SP	int	Stack Pointer
A	int	Accumulator
X	int	Index register
R	float	Float accumulator
RH / RL	int	Upper/lower 16-bit parts of a float variable
M[]	int[65536]	16-bit memory

2.2 VM Flags

- running: true - in program execution mode
- cpuError: 0 - ok, 1 - unknown instruction, 2 - PC out of bounds

3. Public API

3.1 Display / Update methods

displayPC(int value)	Updates PC in UI
displaySP(int value)	Updates SP
displayA(int value)	Updates A

displayX(int value)	Updates X
displayRH(int value)	RH
displayRL(int value)	RL
displayR(float value)	Float accumulator
displayM(int value)	Value of the current memory cell
displayMR(float value)	Float value of the pair (M[PC], M[PC+1])

3.2 Screen I/O

These methods provide input/output through the frontend UI.

Output

sendCToScreen(char c)	display a symbol
sendIToScreen(int i)	output an integer
sendRToScreen(float f)	display a floating point number

Input

receiveCFromScreen()	character request
receiveIFromScreen()	request for an integer number
receiveRFromScreen()	float request

VM calls these methods in a blocking manner, so the front end must return data **immediately** (or organize an input queue).

3.3 Program Loading/Saving

PDLoad.load(computron)	Loads bytecode from file → memory M[]
PDSave.save(computron)	Saves memory to a file
setEndPC(int val)	Set the end of the program
getBeginPC()	Obtain the initial program address
getEndPC()	Obtain the latest address

4. How frontend should integrate with backend

- obtain the last address display the values of all registers at each step
- display the memory status of the selected cell
- display OUT/OUTC/OUTR on the “screen”
- provide input for INP/INPC/INPR
- run execStep() when “Step” is clicked
- run execComputron() in a separate thread when “Run” is selected
- use: setCode(), setInstr(), setAttr() for reference/help
- display processor errors (cpuError != 0)