# Design Notes

## Overview

The assembler is structured in a modular and straightforward manner, employing a two-pass assembly approach to convert assembly source code into machine code.

## Key Classes

1. MachineController

CPU Logic

**Responsibility:** Executes the instruction cycle and enforces all rules of the Instruction Set Architecture.

|  |  |
| --- | --- |
| Key Function | Description |
| performIPL(path) | Executes the Initial Program Load process: resets the machine, loads the LoadFile into memory, and sets the PC to the program's starting address. |
| singleStep() | The primary function that runs one complete Fetch-Decode-Execute cycle. Decodes the IR fields and delegates execution to the correct instruction handler. |
| calculateEA(ix, i, address) | Effective Address Computation. Calculates the final memory address, incorporating indexing (IXR) and indirection (I). This method is also responsible for checking and triggering all Machine Faults (MFR) related to illegal memory access. |
| handleLDR, handleSTR, handleMLT, etc. | Instruction Handlers. Modular functions implementing the logic for all LDR, STR, Transfer, Arithmetic, and Logical opcodes. |

1. Machine State:

Hardware Model

**Responsibility:** Stores the current state of all registers and memory. This is the **data model** of the C6461 machine.

|  |  |
| --- | --- |
| Key Function | Description |
| initialize() | Clears all registers (GPRs, IXRs, PC, MAR, MFR, CC) and sets all 2048 memory locations to zero during a machine reset. |
| get/setPC(), get/setMAR() | Manages the 12-bit PC and MAR values. |
| get/setGPR(), get/setIXR() | Manages the 16-bit General Purpose and Index registers. |
| get/setMemory() | Reads and writes the contents of the main memory array. |

1. SimulatorUI

**Responsibility:** Provides the visual representation of the machine state and handles user input events.

|  |  |
| --- | --- |
| Key Function | Description |
| initializeUI() | Sets up the Swing GUI layout, creating all register fields, buttons, and output areas. |
| updateDisplays() | Reads the current state from MachineState and updates all register fields (PC, GPRs, MBR, etc.) using the correct octal formatting (%04o or %06o). |
| handleIPL() | Opens the file chooser dialog and triggers the MachineController to begin the load process. |
| handleRegisterLoad() | Processes clicks on the small Load buttons next to registers, reading the OCTAL INPUT and depositing the value into the specified register. |

|  |  |  |
| --- | --- | --- |
| Flowchart Step | Description | Purpose in Simulation |
| Start / IPL | The user initiates the Initial Program Load (IPL) in the SimulatorUI. | Resets the hardware and begins the loading process. |
| Load File | The LoadFile is read line-by-line, populating Memory (MachineState). | Transfers the assembled program code into the simulated main memory. |
| Fetch-Decode-Execute by Machine Controller | The MachineController runs the core instruction cycle, fetching the instruction at PC. | Executes the primary logic of the program (arithmetic, transfer, etc.). |
| calculateEA (Security Gate) | The calculateEA function is called to determine the final memory address, simultaneously checking for Reserved Memory Faults and Bounds Violations. | Ensures architectural compliance and halts execution if an illegal memory access is detected. |
| Printer Area Output | The results of the execution (register changes, MFR status, instruction trace) are logged to the PrinterArea. | Provides real-time debugging and verification of the CPU's operation. |
| End | The process terminates on a HLT instruction or a Machine Fault. | Concludes the execution thread and returns control to the GUI. |

## Structural Flow Diagram:

A diagram of a computer program

AI-generated content may be incorrect.