A blue and white logo

Description automatically generated

**CSE3215 DIGITAL LOGIC DESIGN TERM PROJECT**

Yiğit Mesut Ak – 150120006

Leen I. A. Shaqalaih – 150121921

Yusuf İbrahim Matur – 524123004

Yulduzkhon Abdurakhmonova – 150120942

**INSTRUCTION SET**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | [17:14] opcode | [13:10] | [9:6] | [5:4] | [3:0] |
| **ADD** | 0000 | DR | SR1 | 00 | SR2 |
| **AND** | 0001 | DR | SR1 | 00 | SR2 |
| **NAND** | 0010 | DR | SR1 | 00 | SR2 |
| **NOR** | 0011 | DR | SR1 | 00 | SR2 |
| **ADDI** | 0100 | DR | SR | imm6 | |
| **ANDI** | 0101 | DR | SR | imm6 | |
| **LD** | 0110 | DR | address10 | | |
| **ST** | 0111 | SR | address10 | | |
| **CMP** | 1000 | 0000 | OP1 | 00 | OP2 |
| **JUMP** | 1001 | 0000 | address10 | | |
| **JE** | 1010 | 0000 | address10 | | |
| **JA** | 1011 | 0000 | address10 | | |
| **JB** | 1100 | 0000 | address10 | | |
| **JAE** | 1101 | 0000 | address10 | | |
| **JBE** | 1110 | 0000 | address10 | | |

**REGISTERS**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **R0** | 0000 |  | **R4** | 0100 |  | **R8** | 1000 |  | **R12** | 1100 |
| **R1** | 0001 |  | **R5** | 0101 |  | **R9** | 1001 |  | **R13** | 1101 |
| **R2** | 0010 |  | **R6** | 0110 |  | **R10** | 1010 |  | **R14** | 1110 |
| **R3** | 0011 |  | **R7** | 0111 |  | **R11** | 1011 |  | **R15** | 1111 |

**ASSEMBLER IMPLEMENTATION DETAILS**

**Static Variables**

To fetch the operation code of each instruction accordingly, the hash map instructionMap is used in the program. When "ADD" is given as a key, the operation of code of the ADD instruction is returned as result. registerMap is used to represent the registers in binary form. *output.txt* is stated as the default output file as a static variable.

**Main Method**

The program starts by setting the hash maps. asking input file’s name. The instructions in the input file are read by using the method getInstructionsFromFile. Instructions that the method returned are stored in the array list instructions. Each instruction in instructions is transformed into binary by using instructionToBinary method and stored in binaryInstructions array list. In another loop, binary representations are transformed into hexadecimal representations by using binaryToHex method and stored in hexadecimalInstructions array list. Finally, hexadecimal values are written into the output file.

**instructionToBinary** **method**

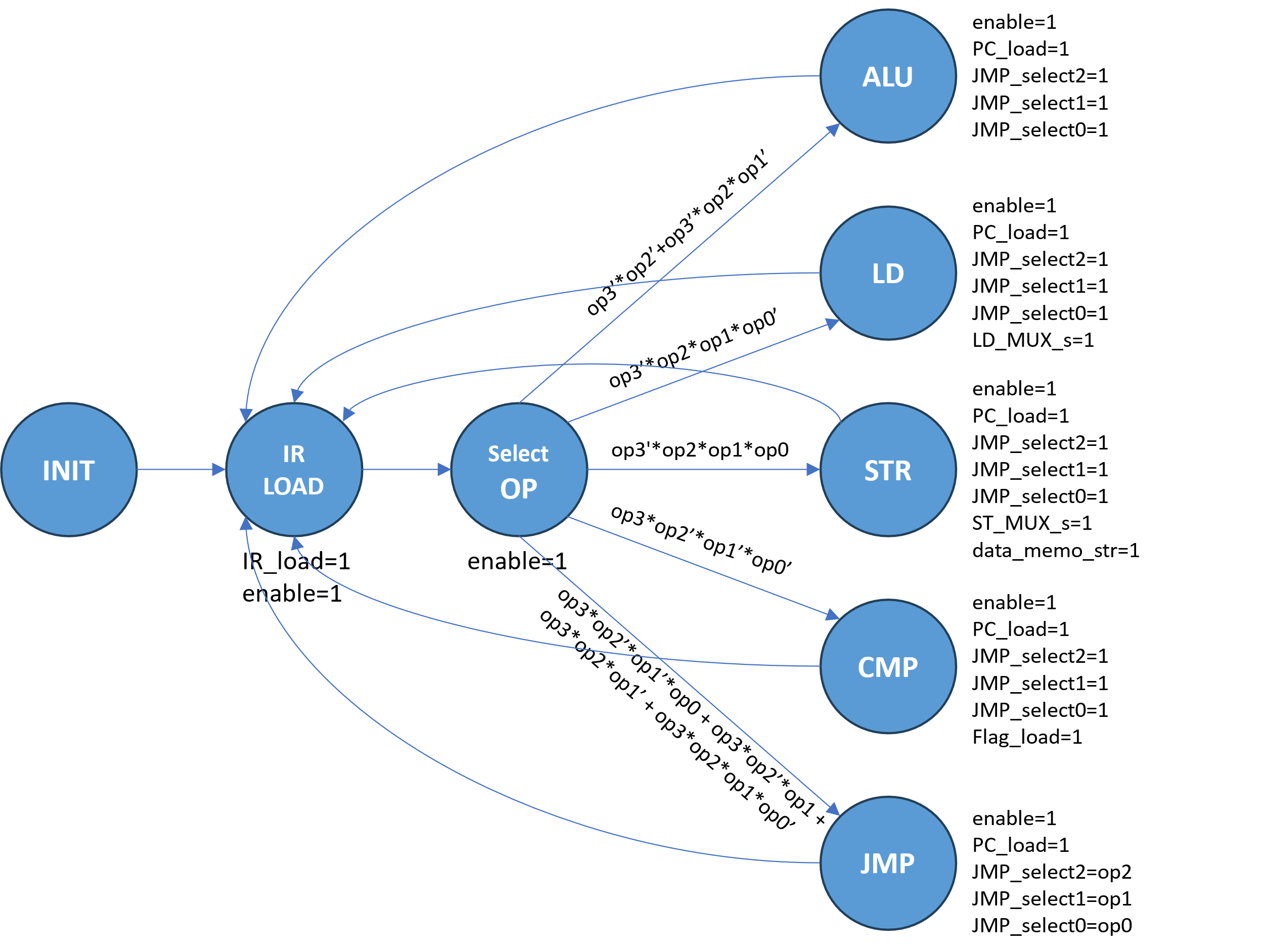
The method gets the instruction string as an input. By splitting the instruction name from the string, it calls the appropriate method according to the instruction name by using swtch-case.

**Instruction methods**

The operation to convert assembly instruction to the binary form is held by implementing each instruction by their names. Each method gets the instruction string as a parameter and splits the string to identify instruction names, registers, and immediate values.

Binary representations of instructions and registers are reached by using the hash maps. Immediate and address values are converted from decimal to binary by using predefined Java methods.

**CIRCUIT IMPLEMENTATION DETAILS**



*Finite state machine of the controller*