Gebze Technical University
Computer Engineering

CSE 331
Computer Organization

PROJECT 3 REPORT

SEDA KANIK
151044068

FATMA NUR ESİRCİ

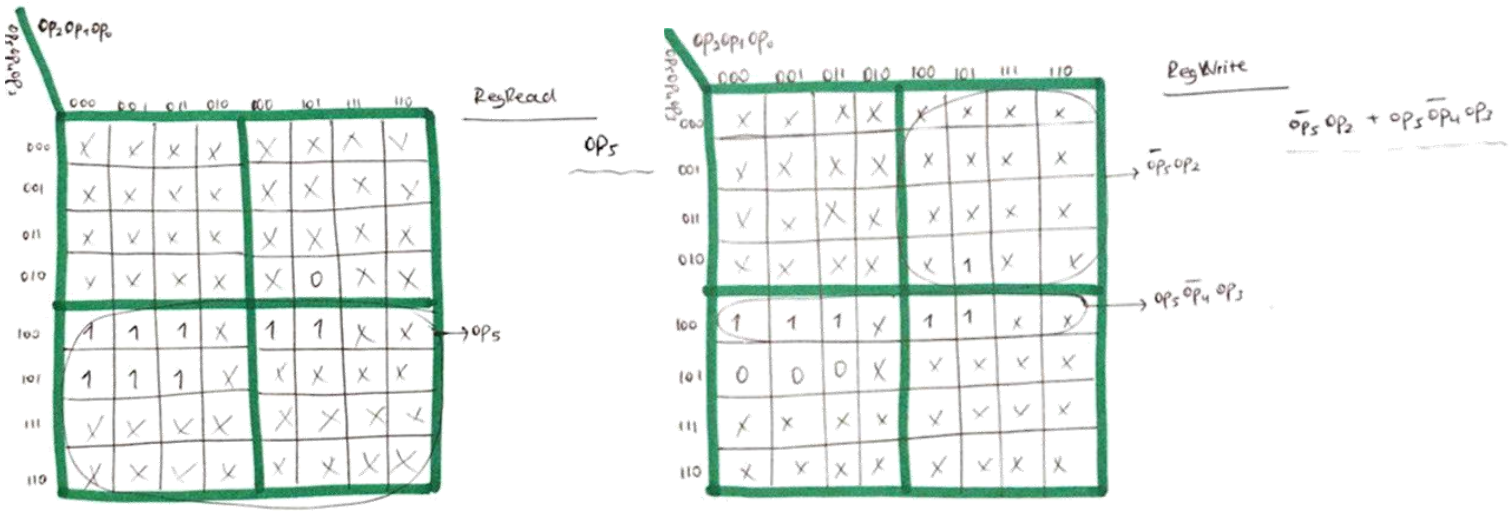
Problem Definition and Solution

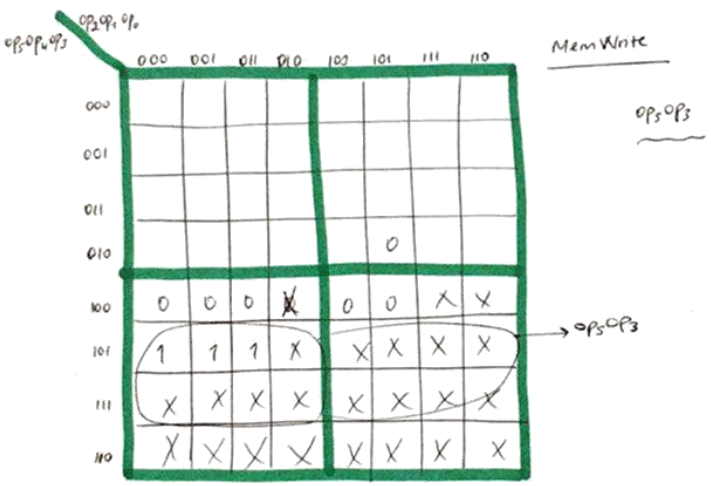
In this homework, ask us to design Datapath especially 9 of I type instructions. To design these instructions, I draw two datapath, one for load and other one for store instructions. This project includes 9 modules also does not include any ALU, because of me, my design. If I will do this project correctly, I will use adder to do all the calculating operations inside of extender.v file. Also I added sign_extender and zero_extender files even I could not use them. I could not use time efficiently.

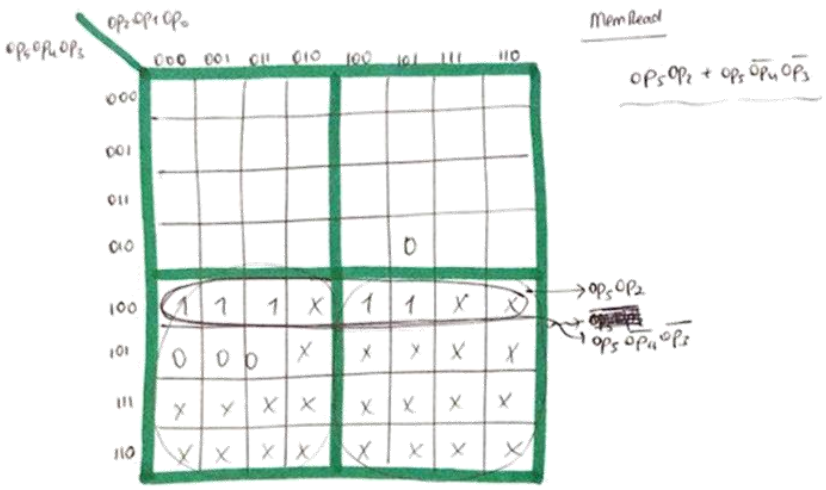
Modules and Definition

- mips32_testbench : creates clock and determines delays every instructions and reads files to arrays
- mips32 : the main file, parses instruction respect their opcode, rs, rt, immediate fields, calculates PC, calls other modules and writes some information and result on screen
- instruction_memory_block : takes instructions and PC, gives new instruction according to index which is PC
- control_unit : takes signals and calculates their value respect opcode of the instruction
- extender : takes instruction and calculates the extended value, in fact, this function should be structural with sign_extender and zero_extender files, but I could not do this type, I got some errors while calling sign_extender and zero_extender files
- sign_extender : the file added project but does not used
- zero_extender : the file added project but does not used
- data_memory_block : adds data.mem file if store instructions, reads from data.mem file if load instructions
- registers_block : updates registers.mem file if load instructions, hold the register contents if store instructions

Design of control_unit:







Test Cases:

PS: THE .MEM FILES ARE IN MIPS32/SIMULATION/MODELSIM

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/mips32_testbench/test/data_memory/data_mem
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000111
00000000000000000000000000001011
00000000000000000000000000000111
00000000000000000000000000001011
00000000000000000000000000000010
00000000000000000000000000001011
00000000000000000000000000000010
00000000000000000000000000001011
00000000000000000000000000000010
00000000000000000000000000001110
00000000000000000000000000000010
00000000000000000000000000000101
00000000000000000000000000000010
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000000010
00000000000000000000000000011001
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000000010
00000000000000000000000000000000
```

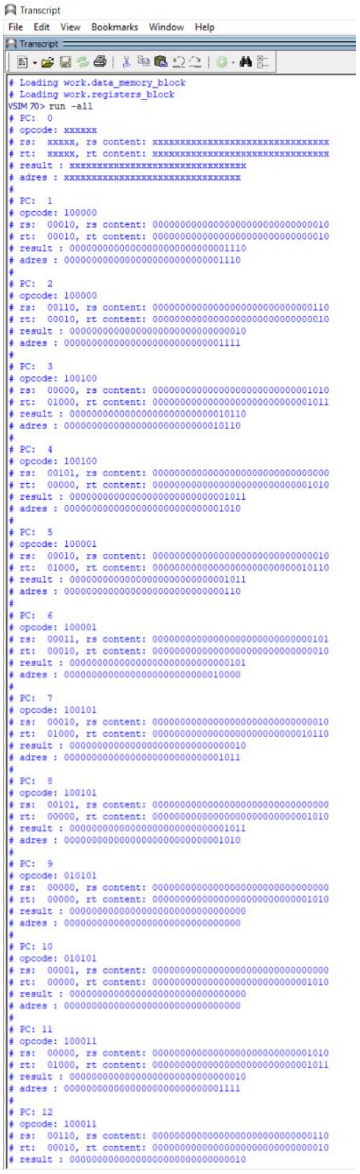
data.mem file has 1224 line

```
10000000010000100000000000001100
10000000110000100000000000001001
10010000000010000000000000001100
10010000101000000000000000001010
10000100010010000000000000000100
10000100011000100000000000001011
10010100010010000000000000001001
10010100101000000000000000001010
01010100000000000000000000000000
01010100001000000000000000000000
10001100000010000000000000000101
10001100110000100000000000001001
10100000011000100000000000000100
10100000010000100000000000001011
10100100110010000000000000000100
10100100010000110000000000001010
10101100110010000000000000000110
10101100010000100000000000001001
```

instruction.mem file has 18 line, 2 examples for each instruction

```
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000001010
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000101
00000000000000000000000000000100
00000000000000000000000000000000
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001011
00000000000000000000000000000101
00000000000000000000000000000010
00000000000000000000000000000101
00000000000000000000000000000000
00000000000000000000000000001001
00000000000000000000000000000100
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000011
```

registers.mem file has 21 line



Here is the test results.

Parsed fields, rs and rt register's contents, the result of all the instructions, and their address in data.mem file are shown on the left side of the screen.

PS: THE .MEM FILES ARE IN MIPS32/SIMULATION/MODELSIM