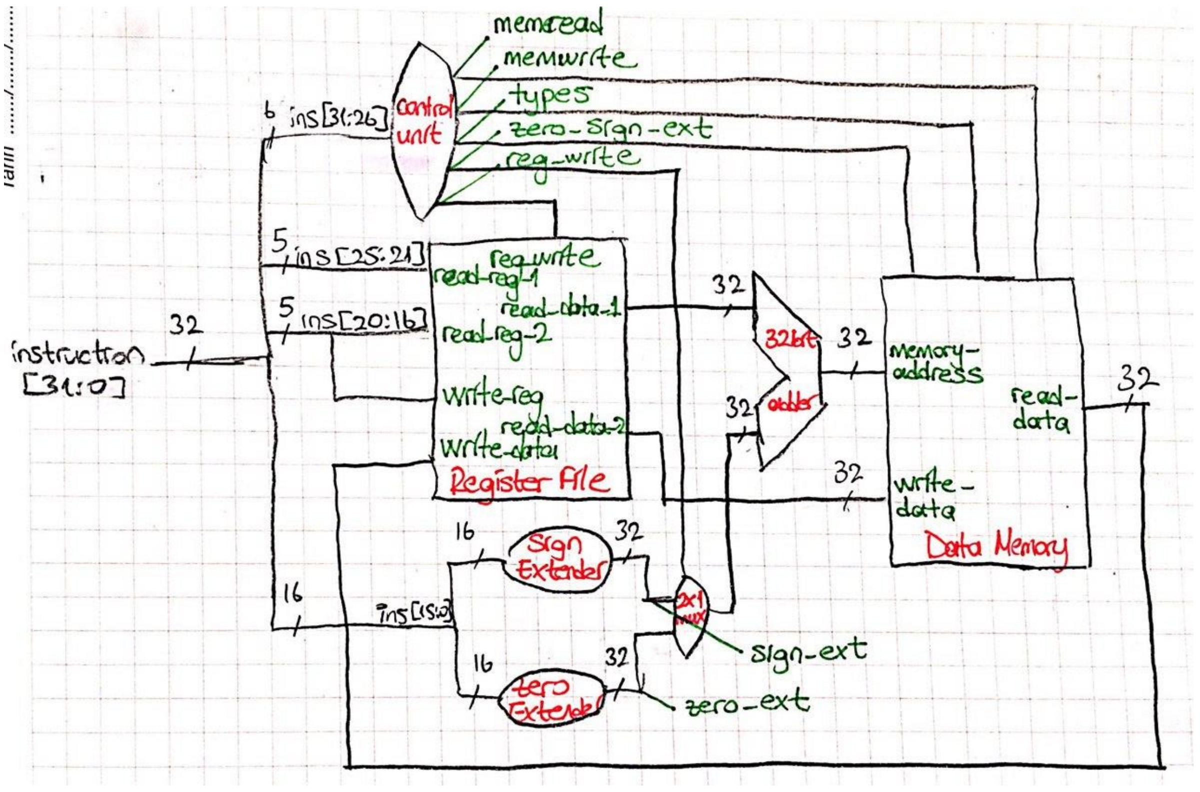
# CSE331 COMPUTER ORGANIZATION HW3 REPORT

NOTE: I designed this datapath using 32 bit adder instead of 32 bit alu.

NOTE: After making changes to the "data memory" and "registers", the new data are written to the "reg_out.txt" and "mem_out.txt files.

NOTE: I put the truth table and the logical expressions of the all control unit signals at appropriate pdfs which they are in the same folder with that pdf. Please check them out!

- I designed mips32 module using 1 control_unit module, 1 mips_registers module, 1 zero_extender32 module, 1 sign_extender32 module, 1 mux2x1_32 module, 1 adder32 module, 1 data_memory module, 1 concatenator module.
- I designed mux2x1 module using 1 not gate, 2 and gates, 1 or gate.
- I designed zero_extender32 and sign_extender32 module using "buf" keyword for fill the appropriate variables.
- I designed half_adder module using 1 xor gate and 1 and gate.
- I designed full_adder module using 2 half_adder module and 1 or gate.
- I designed adder32 module using 32 full_adder module.
- I designed control_unit module using 4 or gates, 3 and gates, 6 not gates
- I designed mips_registers module compatible with 32 32 bit data without clock.
- I designed data_memory module compatible with 256 32 bit data without clock.
- I designed mux4x1_32 module using 3 mux2x1_32 modules.
- I designed concatenator module using "buf" keyword for fill the appropriate variables and 1 mux4x1_32 module.



- The concatenator turn the memory's read_data, register's write_data with using 4x1mux_32 module. This module gets the M[R[rs] + ExtImm] data at the content parameter and turns that data 4 forms which load types got. The mux's inputs are that forms and select bit is a signal which is "types".

**<u>TESTBENCHES</u>**

**- mips_registers_testbench**

inputs:  write_data = 32'b11111111111111111111111111111111;

read_reg_1 = 5'b00001;

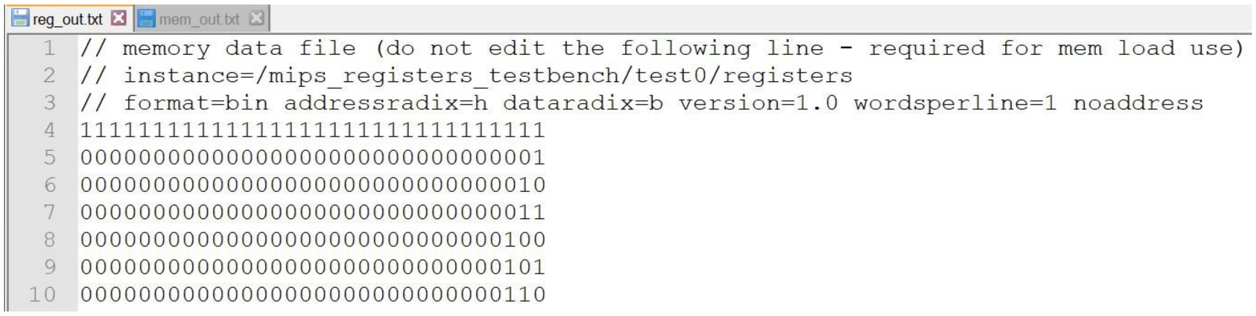
read_reg_2 = 5'b00010;

write_reg = 5'b00000;

reg_write = 1'b1;

monitored output: R[rs] = 00000000000000000000000000000001, R[rt] = 00000000000000000000000000000010

reg_out.txt file after that test (LINE 4):



```
 1 // memory data file (do not edit the following line - required for mem load use)
 2 // instance=/mips_registers_testbench/test0/registers
 3 // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
 4 11111111111111111111111111111111
 5 00000000000000000000000000000001
 6 00000000000000000000000000000010
 7 00000000000000000000000000000011
 8 00000000000000000000000000000100
 9 00000000000000000000000000000101
10 00000000000000000000000000000110
```

**- data_memory_testbench**

test1:

inputs:  address = 32'b00000000000000000000000000000001;

write_data = 32'b11111111111111111111111111111111;

memwrite = 1'b1;

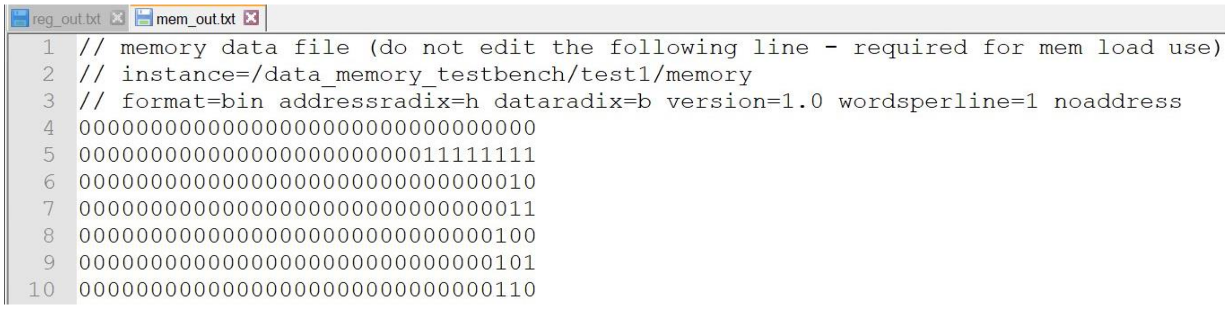
memread = 1'b0;

load_types = 2'b00; (STORE BYTE)

monitored output:

read data = 00000000000000000000000000010100, write data = 11111111111111111111111111111111

mem_out.txt file after the test (LINE 5):



```
 1 // memory data file (do not edit the following line - required for mem load use)
 2 // instance=/data_memory_testbench/test1/memory
 3 // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
 4 00000000000000000000000000000000
 5 00000000000000000000000011111111
 6 00000000000000000000000000000010
 7 00000000000000000000000000000011
 8 00000000000000000000000000000100
 9 00000000000000000000000000000101
10 00000000000000000000000000000110
```

test2:

inputs: address = 32'b00000000000000000000000000000001;

write_data = 32'b11111111111111111111111111111111;

memwrite = 1'b1;

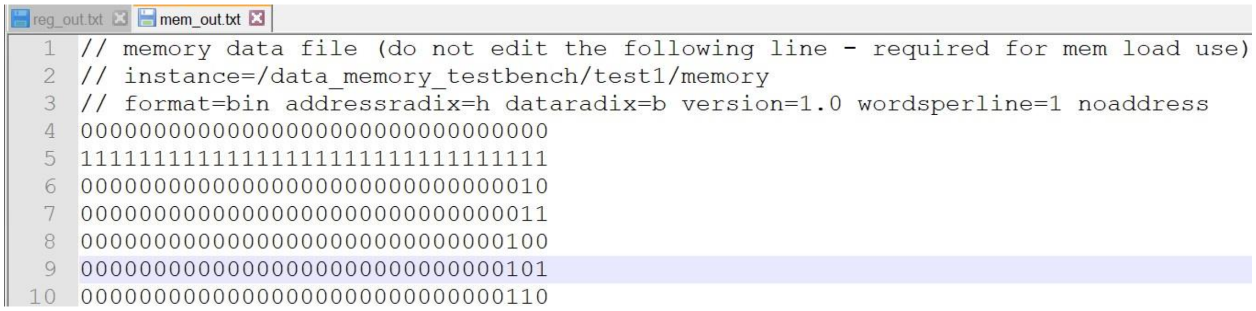
memread = 1'b0;

load_types = 2'b11; (STORE WORD)

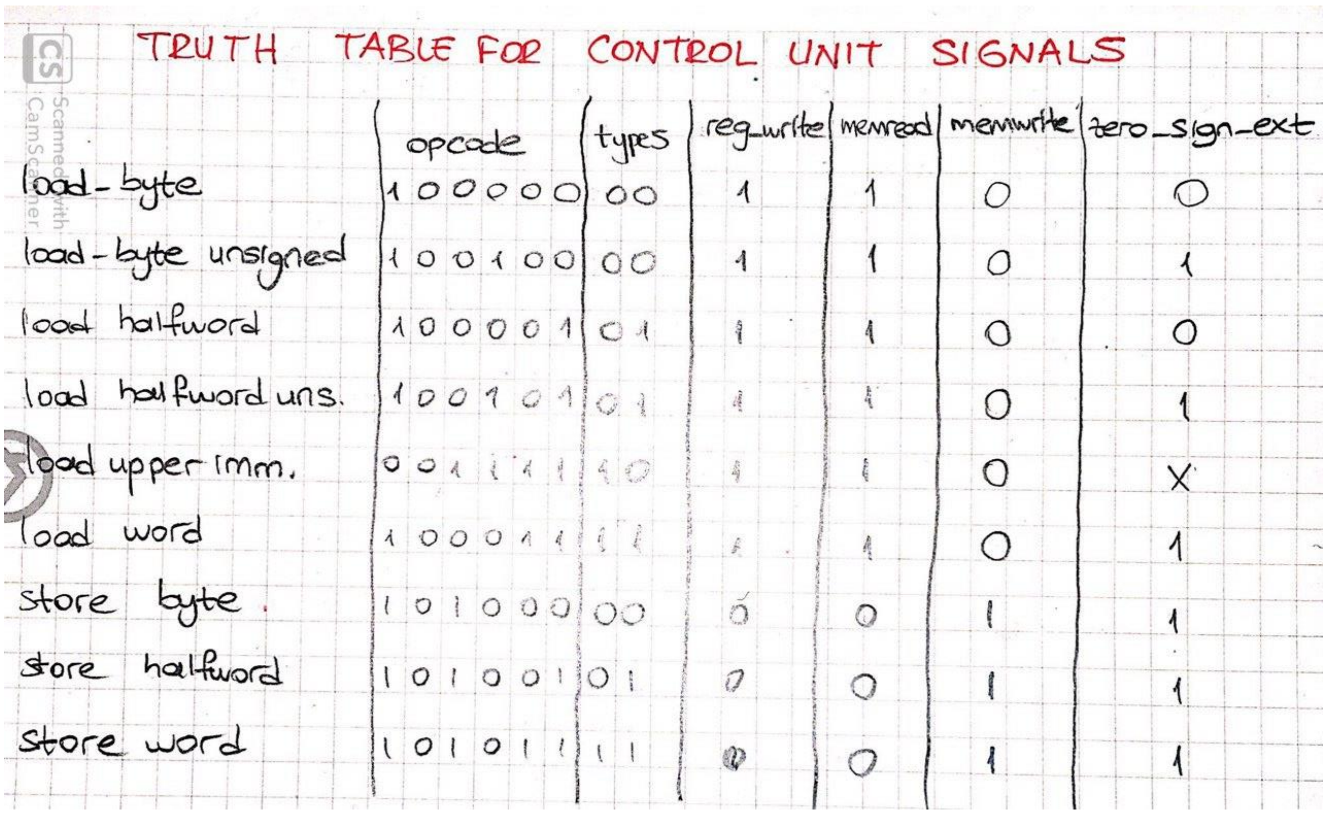
monitored output:

read data = 00000000000000000000000000010100, write data = 11111111111111111111111111111111

mem_out.txt file after the test (LINE 5):



```
 1 // memory data file (do not edit the following line - required for mem load use)
 2 // instance=/data_memory_testbench/test1/memory
 3 // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
 4 00000000000000000000000000000000
 5 11111111111111111111111111111111
 6 00000000000000000000000000000010
 7 00000000000000000000000000000011
 8 00000000000000000000000000000100
 9 00000000000000000000000000000101
10 00000000000000000000000000000110
```



TRUTH TABLE FOR CONTROL UNIT SIGNALS

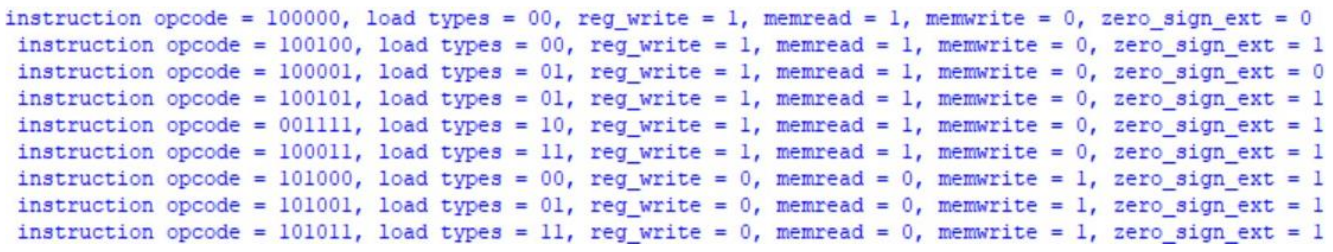
| | opcode | types | reg_write | memread | memwrite | zero_sign_ext |
|---|---|---|---|---|---|---|
| load-byte | 1 0 0 0 0 0 | 0 0 | 1 | 1 | 0 | 0 |
| load-byte unsigned | 1 0 0 1 0 0 | 0 0 | 1 | 1 | 0 | 1 |
| load halfword | 1 0 0 0 0 1 | 0 1 | 1 | 1 | 0 | 0 |
| load halfword uns. | 1 0 0 1 0 1 | 0 1 | 1 | 1 | 0 | 1 |
| load upper imm. | 0 0 1 1 1 1 | 1 0 | 1 | 1 | 0 | X |
| load word | 1 0 0 0 1 1 | 1 1 | 1 | 1 | 0 | 1 |
| store byte | 1 0 1 0 0 0 | 0 0 | 0 | 0 | 1 | 1 |
| store halfword | 1 0 1 0 0 1 | 0 1 | 0 | 0 | 1 | 1 |
| store word | 1 0 1 0 1 1 | 1 1 | 0 | 0 | 1 | 1 |

**-control_unit_testbench**

inputs:

```
instruction_opcode = 6'b100000; // lb
instruction_opcode = 6'b100100; // lbu
instruction_opcode = 6'b100001; // lh
instruction_opcode = 6'b100101; // lhu
instruction_opcode = 6'b001111; // lui
instruction_opcode = 6'b100011; // lw
instruction_opcode = 6'b101000; // sb
instruction_opcode = 6'b101001; // sh
instruction_opcode = 6'b101011; // sw
```

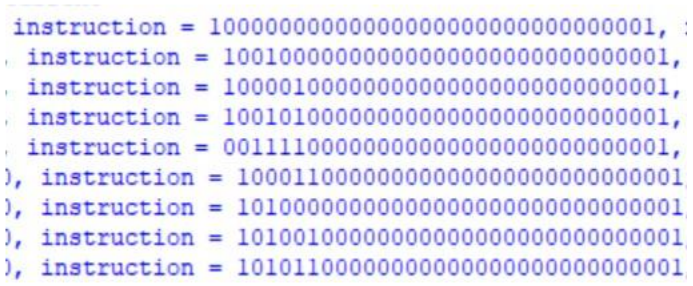
monitoring outputs:



```
instruction opcode = 100000, load types = 00, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 0
 instruction opcode = 100100, load types = 00, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 1
 instruction opcode = 100001, load types = 01, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 0
 instruction opcode = 100101, load types = 01, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 1
 instruction opcode = 001111, load types = 10, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 1
 instruction opcode = 100011, load types = 11, reg_write = 1, memread = 1, memwrite = 0, zero_sign_ext = 1
 instruction opcode = 101000, load types = 00, reg_write = 0, memread = 0, memwrite = 1, zero_sign_ext = 1
 instruction opcode = 101001, load types = 01, reg_write = 0, memread = 0, memwrite = 1, zero_sign_ext = 1
 instruction opcode = 101011, load types = 11, reg_write = 0, memread = 0, memwrite = 1, zero_sign_ext = 1
```

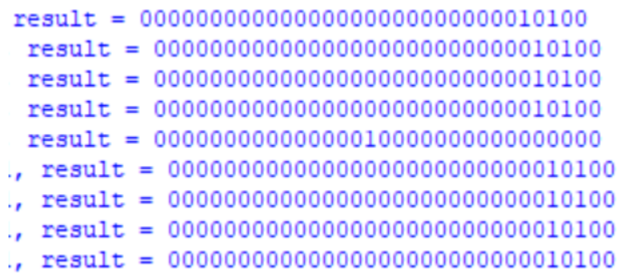
-**mips32_testbench**

inputs:



```
 instruction = 10000000000000000000000000000001,
, instruction = 10010000000000000000000000000001,
, instruction = 10000100000000000000000000000001,
, instruction = 10010100000000000000000000000001,
, instruction = 00111100000000000000000000000001,
), instruction = 10001100000000000000000000000001
), instruction = 10100000000000000000000000000001
), instruction = 10100100000000000000000000000001
), instruction = 10101100000000000000000000000001
```

outputs:



```
result = 00000000000000000000000000010100
 result = 00000000000000000000000000010100
 result = 00000000000000000000000000010100
 result = 00000000000000000000000000010100
 result = 00000000000000010000000000000000
., result = 00000000000000000000000000010100
., result = 00000000000000000000000000010100
., result = 00000000000000000000000000010100
., result = 00000000000000000000000000010100
```