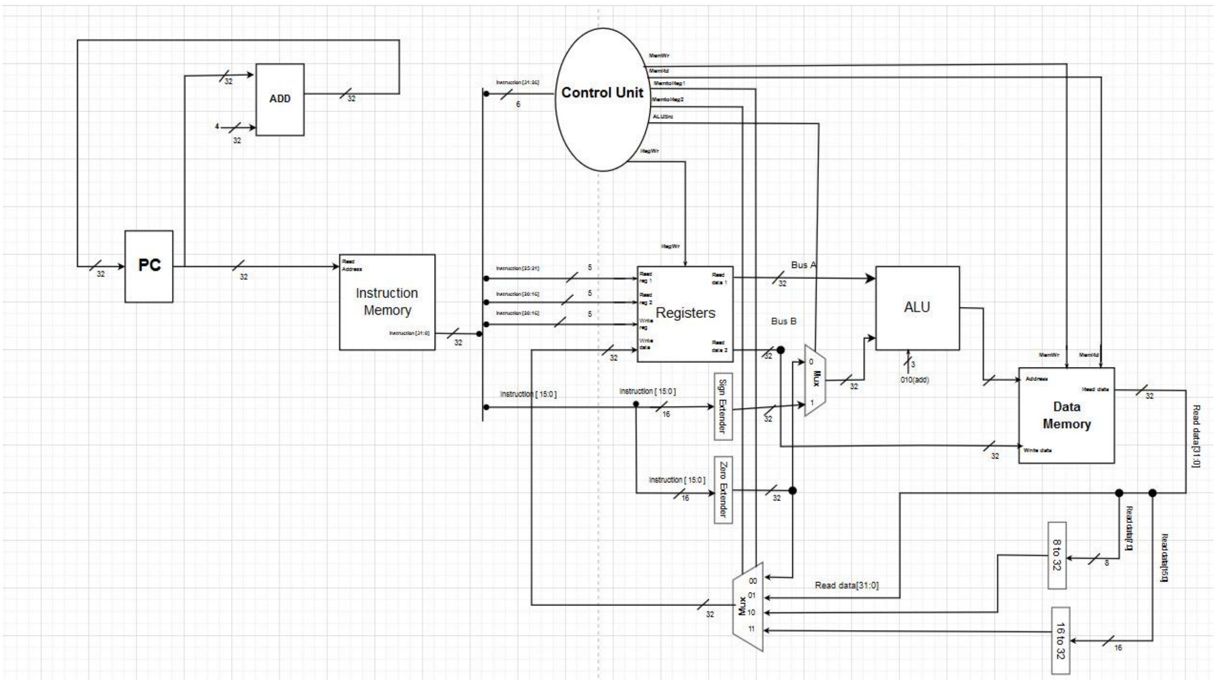
CSE 331 – COMPUTER ORGANIZATION HOMEWORK 2 REPORT

Name – Surname : Furkan OZEV

Number : 161044036

DATAPATH:



CYCLE AND MODULES:

1. Clock is triggered by the program counter.
2. The mips_instr_mem module reads the instruction shown by the program counter from the file.
3. This instruction, which is read from the file, is divided into certain parts with the help of inst_parser module. These are: opcode [31:26], rs [26:22], rt [21:16], immediate [15: 0].
4. mips_registers module can read or modify the data of the corresponding registers from the register file. This module reads each time independently of the clock. Write to the register array is done in clock posedge and write to file is in clock negedge.
5. The control_unit module allows the selection of control or selection signals in the data path. This module takes the instruction's opcode and processes it and generates the signals.
   a.) MemRead : If this signal is set to 1, it indicates that the memory needs to be read.
      ⇨ opcode[3]
   b.) MemWrite: If this signal is set to 1, it indicates that the memory needs to be write.
      ⇨ opcode[5].opcode[3]

   c.) RegWrite: If this signal is set to 1, it indicates that the register needs to be write.
      ⇨ not MemWrite
   d.) Sign: The instructions must be extended in 2 different ways to operate. If the signal is set to 1, sign extend is used, and if the signal is 0, zero extend is used.
      ⇨ (opcode[1]+ opcode[2]+ opcode[3]). opcode[5]
   e.) SignData[1:0]: The data to be written to register can be 4 different ways. Therefore, these signals will be used as selection bits of 4 mux.
6. It is sent to the immediate ALU with the value read from rs register. In ALU these 2 values are added.
7. The memory module can read or modify data from memory according to the corresponding signals.
8. When the cycle is finished, the program counter is increased by 1 to move to the next instruction.

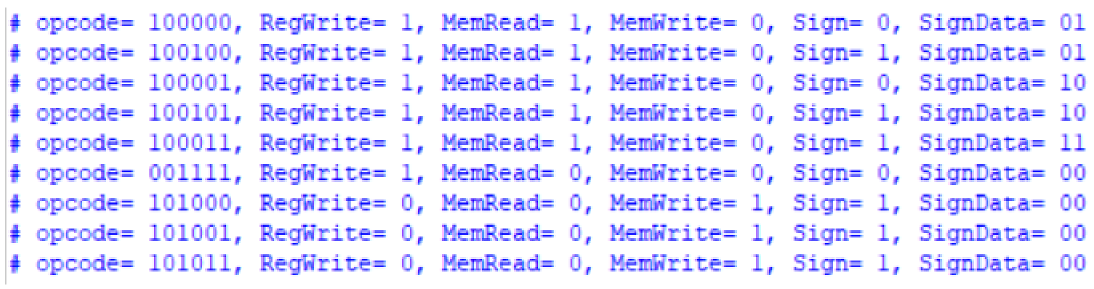
SOME INFORMATIONS:

1. This was done so that the processor would work correctly within the given 9 instruction types.
2. Testbenches were prepared for each module and their tests were completed successfully.
3. Register.mem was used for register, data.txt was used for memory, instruction.mem was used for instruction. You can add the instructions or data you want to try to the file.
4. Since there are 256 data in the file, a maximum of 255 addresses can be used in memory. You should pay attention to this when running the instructions or replace the memory file with a larger file.
5. Running testbenches can cause memory and register files to change. Each study can produce different results. In such cases you can replace the original files in the folder.

TEST BENCHS:

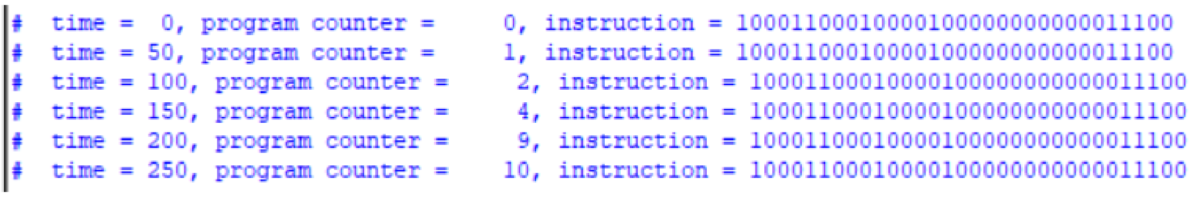
Testbenches of all modules used were prepared. You can access the test of modules that have not been visualized using Modelsim.

Control Unit:



```
# opcode= 100000, RegWrite= 1, MemRead= 1, MemWrite= 0, Sign= 0, SignData= 01
# opcode= 100100, RegWrite= 1, MemRead= 1, MemWrite= 0, Sign= 1, SignData= 01
# opcode= 100001, RegWrite= 1, MemRead= 1, MemWrite= 0, Sign= 0, SignData= 10
# opcode= 100101, RegWrite= 1, MemRead= 1, MemWrite= 0, Sign= 1, SignData= 10
# opcode= 100011, RegWrite= 1, MemRead= 1, MemWrite= 0, Sign= 1, SignData= 11
# opcode= 001111, RegWrite= 1, MemRead= 0, MemWrite= 0, Sign= 0, SignData= 00
# opcode= 101000, RegWrite= 0, MemRead= 0, MemWrite= 1, Sign= 1, SignData= 00
# opcode= 101001, RegWrite= 0, MemRead= 0, MemWrite= 1, Sign= 1, SignData= 00
# opcode= 101011, RegWrite= 0, MemRead= 0, MemWrite= 1, Sign= 1, SignData= 00
```

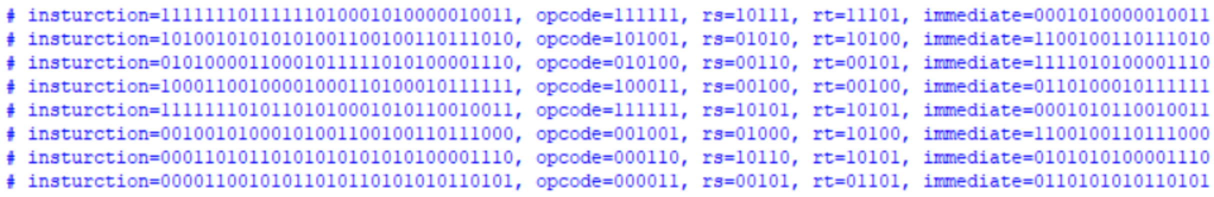
Instruction Memory:



```
#  time =  0, program counter =     0, instruction = 10001100010000100000000000011100
#  time = 50, program counter =     1, instruction = 10001100010000100000000000011100
#  time = 100, program counter =     2, instruction = 10001100010000100000000000011100
#  time = 150, program counter =     4, instruction = 10001100010000100000000000011100
#  time = 200, program counter =     9, instruction = 10001100010000100000000000011100
#  time = 250, program counter =    10, instruction = 10001100010000100000000000011100
```

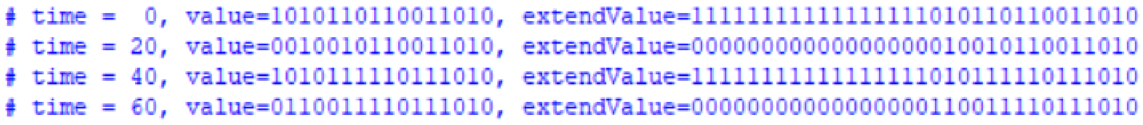
Reads the instruction shown by the program counter.

Instruction Parser:



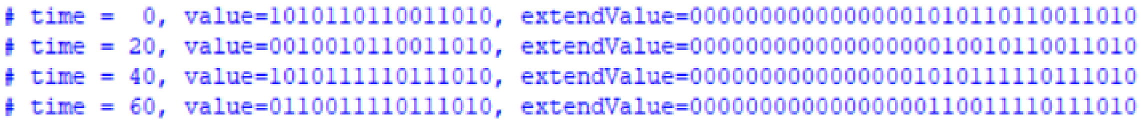
```
# insturction=11111110111111010001010000010011, opcode=111111, rs=10111, rt=11101, immediate=0001010000010011
# insturction=10100101010101001100100110111010, opcode=101001, rs=01010, rt=10100, immediate=1100100110111010
# insturction=01010000110001011111010100001110, opcode=010100, rs=00110, rt=00101, immediate=1111010100001110
# insturction=10001100100001000110100010111111, opcode=100011, rs=00100, rt=00100, immediate=0110100010111111
# insturction=11111110101101010001010110010011, opcode=111111, rs=10101, rt=10101, immediate=0001010110010011
# insturction=00100101000101001100100110111000, opcode=001001, rs=01000, rt=10100, immediate=1100100110111000
# insturction=00011010110101010101010100001110, opcode=000110, rs=10110, rt=10101, immediate=0101010100001110
# insturction=00001100101011010110101010110101, opcode=000011, rs=00101, rt=01101, immediate=0110101010110101
```

Sign Extend:



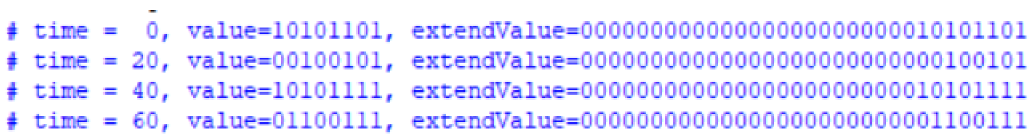
```
# time =  0, value=1010110110011010, extendValue=11111111111111111010110110011010
# time = 20, value=0010010110011010, extendValue=00000000000000000010010110011010
# time = 40, value=1010111110111010, extendValue=11111111111111111010111110111010
# time = 60, value=0110011110111010, extendValue=00000000000000000110011110111010
```

Zero Extend:



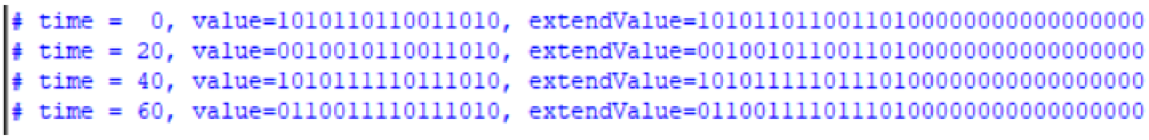
```
# time =  0, value=1010110110011010, extendValue=00000000000000001010110110011010
# time = 20, value=0010010110011010, extendValue=00000000000000000010010110011010
# time = 40, value=1010111110111010, extendValue=00000000000000001010111110111010
# time = 60, value=0110011110111010, extendValue=00000000000000000110011110111010
```

Zero Extend 8 Bit:



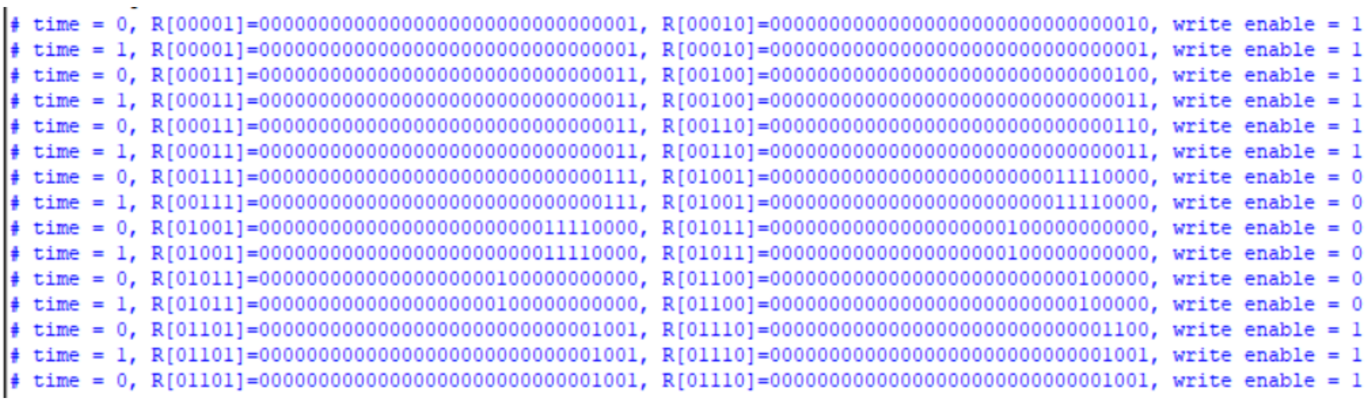
```
# time =  0, value=10101101, extendValue=00000000000000000000000010101101
# time = 20, value=00100101, extendValue=00000000000000000000000000100101
# time = 40, value=10101111, extendValue=00000000000000000000000010101111
# time = 60, value=01100111, extendValue=00000000000000000000000001100111
```

Zero Extend Immediate:



```
# time =  0, value=1010110110011010, extendValue=10101101100110100000000000000000
# time = 20, value=0010010110011010, extendValue=00100101100110100000000000000000
# time = 40, value=1010111110111010, extendValue=10101111101110100000000000000000
# time = 60, value=0110011110111010, extendValue=01100111101110100000000000000000
```

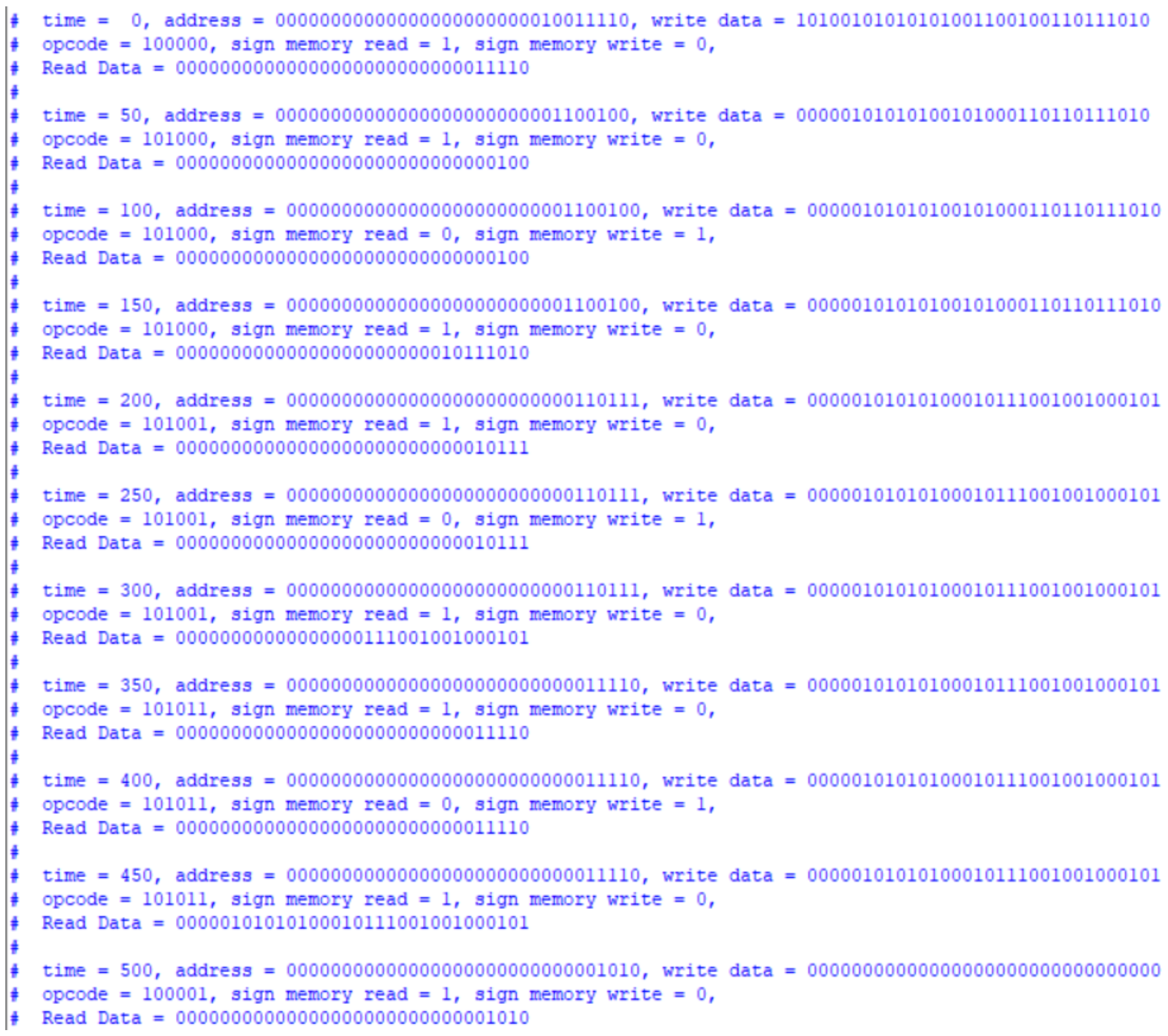
Mips Register:



```
# time = 0, R[00001]=00000000000000000000000000000001, R[00010]=00000000000000000000000000000010, write enable = 1
# time = 1, R[00001]=00000000000000000000000000000001, R[00010]=00000000000000000000000000000001, write enable = 1
# time = 0, R[00011]=00000000000000000000000000000011, R[00100]=00000000000000000000000000000100, write enable = 1
# time = 1, R[00011]=00000000000000000000000000000011, R[00100]=00000000000000000000000000000011, write enable = 1
# time = 0, R[00011]=00000000000000000000000000000011, R[00110]=00000000000000000000000000000110, write enable = 1
# time = 1, R[00011]=00000000000000000000000000000011, R[00110]=00000000000000000000000000000011, write enable = 1
# time = 0, R[00111]=00000000000000000000000000000111, R[01001]=00000000000000000000000011110000, write enable = 0
# time = 1, R[00111]=00000000000000000000000000000111, R[01001]=00000000000000000000000011110000, write enable = 0
# time = 0, R[01001]=00000000000000000000000011110000, R[01011]=00000000000000000000100000000000, write enable = 0
# time = 1, R[01001]=00000000000000000000000011110000, R[01011]=00000000000000000000100000000000, write enable = 0
# time = 0, R[01011]=00000000000000000000100000000000, R[01100]=00000000000000000000000000100000, write enable = 0
# time = 1, R[01011]=00000000000000000000100000000000, R[01100]=00000000000000000000000000100000, write enable = 0
# time = 0, R[01101]=00000000000000000000000000001001, R[01110]=00000000000000000000000000001100, write enable = 1
# time = 1, R[01101]=00000000000000000000000000001001, R[01110]=00000000000000000000000000001001, write enable = 1
# time = 0, R[01101]=00000000000000000000000000001001, R[01110]=00000000000000000000000000001001, write enable = 1
```

In this image, register rs on the left and register rt on the right. If Clock 1 and write signal 1, rs value is written to rt register.

Mips Memory:

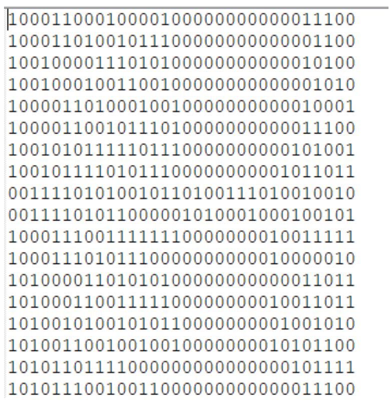


```
#  time =  0, address = 00000000000000000000000010011110, write data = 10100101010101001100100110111010
#  opcode = 100000, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000000011110
#
#  time = 50, address = 00000000000000000000000001100100, write data = 00001010101001010001101101111010
#  opcode = 101000, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000000000100
#
#  time = 100, address = 00000000000000000000000001100100, write data = 00001010101001010001101101111010
#  opcode = 101000, sign memory read = 0, sign memory write = 1,
#  Read Data = 00000000000000000000000000000100
#
#  time = 150, address = 00000000000000000000000001100100, write data = 00001010101001010001101101111010
#  opcode = 101000, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000010111010
#
#  time = 200, address = 00000000000000000000000000110111, write data = 00001010101000101110010010000101
#  opcode = 101001, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000000010111
#
#  time = 250, address = 00000000000000000000000000110111, write data = 00001010101000101110010010000101
#  opcode = 101001, sign memory read = 0, sign memory write = 1,
#  Read Data = 00000000000000000000000000010111
#
#  time = 300, address = 00000000000000000000000000110111, write data = 00001010101000101110010010000101
#  opcode = 101001, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000111001001000101
#
#  time = 350, address = 00000000000000000000000000011110, write data = 00001010101000101110010010000101
#  opcode = 101011, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000000011110
#
#  time = 400, address = 00000000000000000000000000011110, write data = 00001010101000101110010010000101
#  opcode = 101011, sign memory read = 0, sign memory write = 1,
#  Read Data = 00000000000000000000000000011110
#
#  time = 450, address = 00000000000000000000000000011110, write data = 00001010101000101110010010000101
#  opcode = 101011, sign memory read = 1, sign memory write = 0,
#  Read Data = 00001010101000101110010010000101
#
#  time = 500, address = 00000000000000000000000000001010, write data = 00000000000000000000000000000000
#  opcode = 100001, sign memory read = 1, sign memory write = 0,
#  Read Data = 00000000000000000000000000001010
```

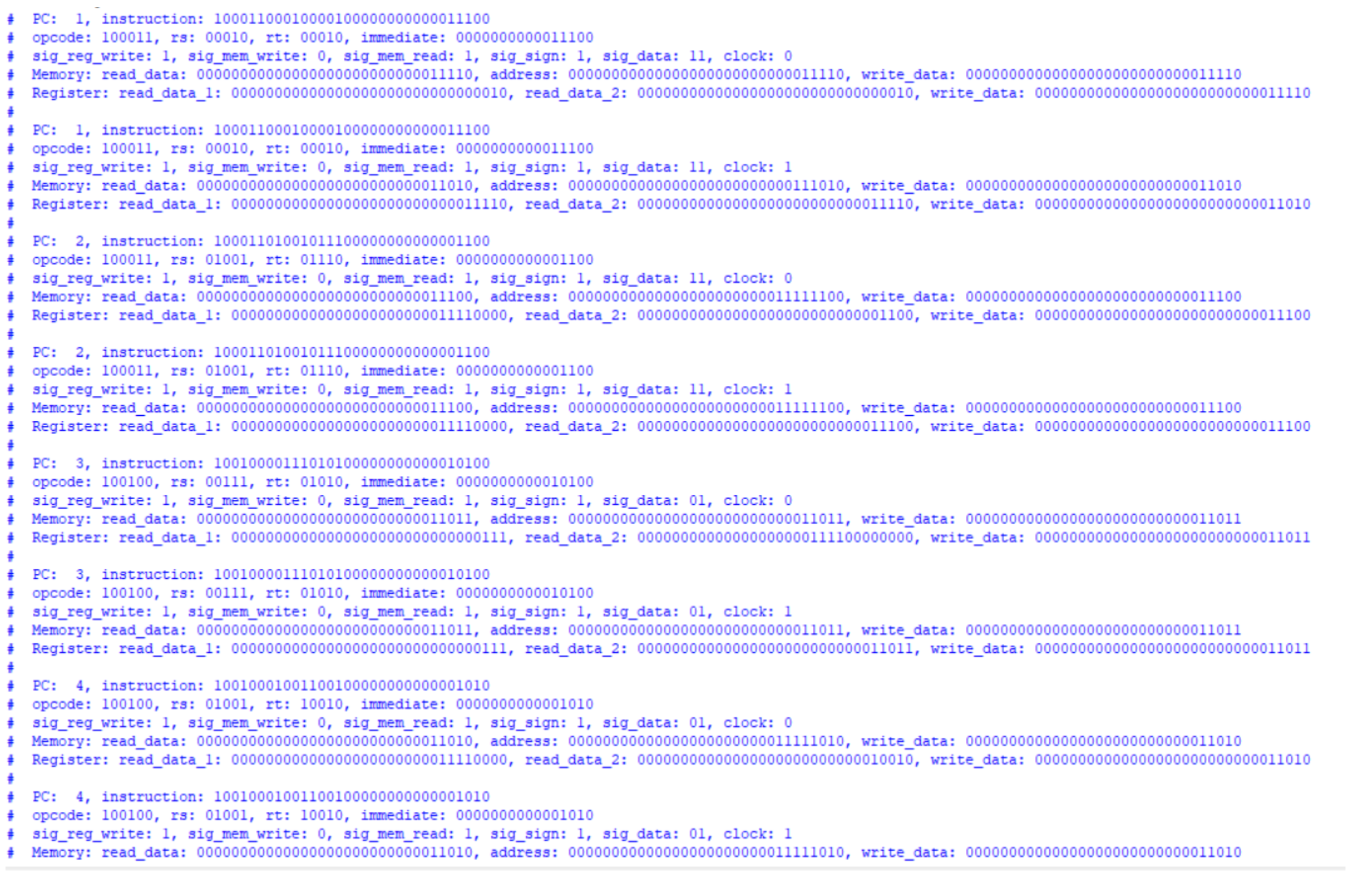
If the read signal is 1, it reads the data at the given memory address. If the write signal is 1, the data to be written is written to the given memory address.

Mips Processor:

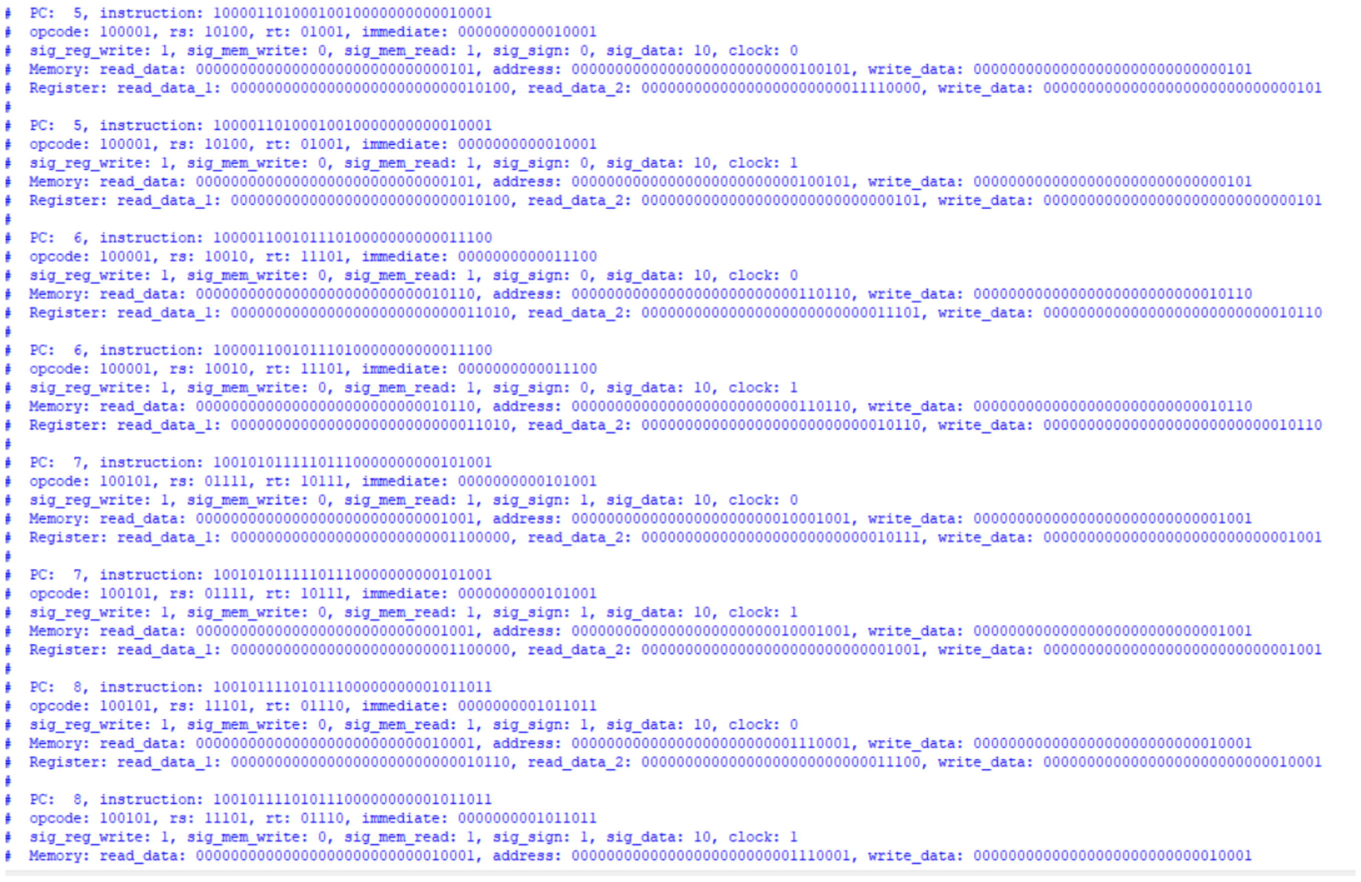
Instruction Fıle:



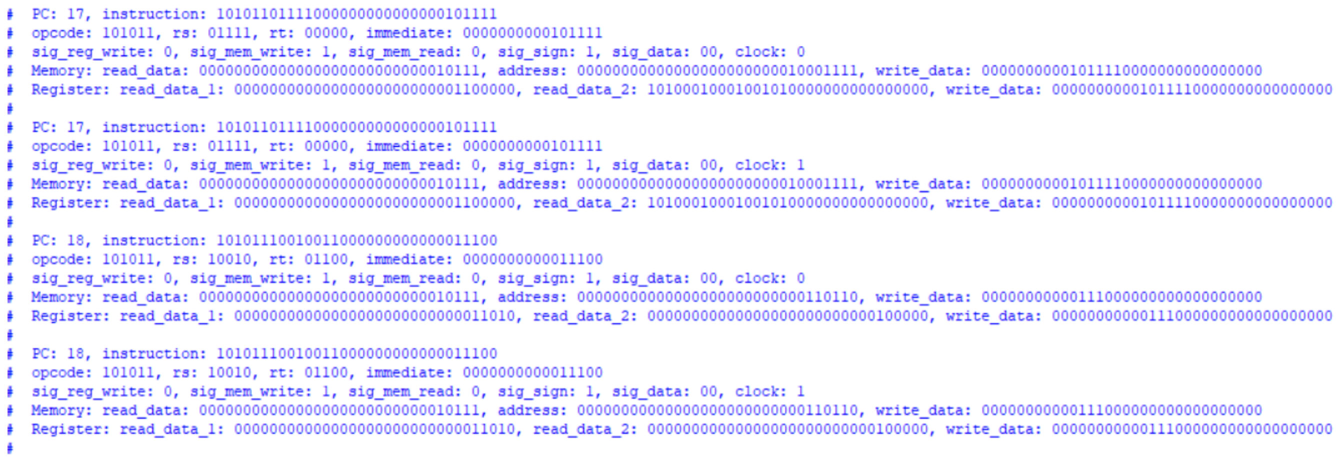
```
10001100010000100000000000011100
10001101001011100000000000001100
10010000111010100000000000010100
10010001001100100000000000001010
10000110100010010000000000010001
10000110010111010000000000011100
10010101111101110000000000101001
10010111101011100000000001011011
00111101010010110100111010010010
00111101011000001010001000100101
10001110011111110000000010011111
10001110101110000000000010000010
10100001101010100000000000011011
10100011001111100000000010011011
10100101001010110000000001001010
10100110010010010000000010101100
10101101111000000000000000101111
10101110010011000000000000011100
```



```
#  PC:  1, instruction: 10001100010000100000000000011100
#  opcode: 100011, rs: 00010, rt: 00010, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 0
#  Memory: read_data: 00000000000000000000000000011110, address: 00000000000000000000000000011110, write_data: 00000000000000000000000000011110
#  Register: read_data_1: 00000000000000000000000000000010, read_data_2: 00000000000000000000000000000010, write_data: 00000000000000000000000000011110
#
#  PC:  1, instruction: 10001100010000100000000000011100
#  opcode: 100011, rs: 00010, rt: 00010, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 1
#  Memory: read_data: 00000000000000000000000000011010, address: 00000000000000000000000000111010, write_data: 00000000000000000000000000011010
#  Register: read_data_1: 00000000000000000000000000011110, read_data_2: 00000000000000000000000000011110, write_data: 00000000000000000000000000011010
#
#  PC:  2, instruction: 10001101001011100000000000001100
#  opcode: 100011, rs: 01001, rt: 01110, immediate: 0000000000001100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 0
#  Memory: read_data: 00000000000000000000000000011100, address: 00000000000000000000000011111100, write_data: 00000000000000000000000000011100
#  Register: read_data_1: 00000000000000000000000011110000, read_data_2: 00000000000000000000000000001100, write_data: 00000000000000000000000000011100
#
#  PC:  2, instruction: 10001101001011100000000000001100
#  opcode: 100011, rs: 01001, rt: 01110, immediate: 0000000000001100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 1
#  Memory: read_data: 00000000000000000000000000011100, address: 00000000000000000000000011111100, write_data: 00000000000000000000000000011100
#  Register: read_data_1: 00000000000000000000000011110000, read_data_2: 00000000000000000000000000011100, write_data: 00000000000000000000000000011100
#
#  PC:  3, instruction: 10010000111010100000000000010100
#  opcode: 100100, rs: 00111, rt: 01010, immediate: 0000000000010100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 01, clock: 0
#  Memory: read_data: 00000000000000000000000000011011, address: 00000000000000000000000000011011, write_data: 00000000000000000000000000011011
#  Register: read_data_1: 00000000000000000000000000000111, read_data_2: 00000000000000000000111100000000, write_data: 00000000000000000000000000011011
#
#  PC:  3, instruction: 10010000111010100000000000010100
#  opcode: 100100, rs: 00111, rt: 01010, immediate: 0000000000010100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 01, clock: 1
#  Memory: read_data: 00000000000000000000000000011011, address: 00000000000000000000000000011011, write_data: 00000000000000000000000000011011
#  Register: read_data_1: 00000000000000000000000000000111, read_data_2: 00000000000000000000000000011011, write_data: 00000000000000000000000000011011
#
#  PC:  4, instruction: 10010001001100100000000000001010
#  opcode: 100100, rs: 01001, rt: 10010, immediate: 0000000000001010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 01, clock: 0
#  Memory: read_data: 00000000000000000000000000011010, address: 00000000000000000000000011111010, write_data: 00000000000000000000000000011010
#  Register: read_data_1: 00000000000000000000000011110000, read_data_2: 00000000000000000000000000010010, write_data: 00000000000000000000000000011010
#
#  PC:  4, instruction: 10010001001100100000000000001010
#  opcode: 100100, rs: 01001, rt: 10010, immediate: 0000000000001010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 01, clock: 1
#  Memory: read_data: 00000000000000000000000000011010, address: 00000000000000000000000011111010, write_data: 00000000000000000000000000011010
```



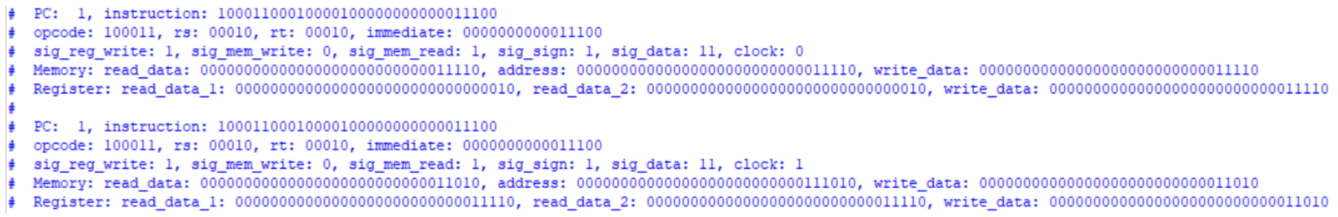
```
#  PC:  5, instruction: 10000110100010010000000000010001
#  opcode: 100001, rs: 10100, rt: 01001, immediate: 0000000000010001
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 0, sig_data: 10, clock: 0
#  Memory: read_data: 00000000000000000000000000000101, address: 00000000000000000000000000100101, write_data: 00000000000000000000000000000101
#  Register: read_data_1: 00000000000000000000000000010100, read_data_2: 00000000000000000000000011110000, write_data: 00000000000000000000000000000101
#
#  PC:  5, instruction: 10000110100010010000000000010001
#  opcode: 100001, rs: 10100, rt: 01001, immediate: 0000000000010001
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 0, sig_data: 10, clock: 1
#  Memory: read_data: 00000000000000000000000000000101, address: 00000000000000000000000000100101, write_data: 00000000000000000000000000000101
#  Register: read_data_1: 00000000000000000000000000010100, read_data_2: 00000000000000000000000000000101, write_data: 00000000000000000000000000000101
#
#  PC:  6, instruction: 10000110010111010000000000011100
#  opcode: 100001, rs: 10010, rt: 11101, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 0, sig_data: 10, clock: 0
#  Memory: read_data: 00000000000000000000000000010110, address: 00000000000000000000000000110110, write_data: 00000000000000000000000000010110
#  Register: read_data_1: 00000000000000000000000000011010, read_data_2: 00000000000000000000000000011101, write_data: 00000000000000000000000000010110
#
#  PC:  6, instruction: 10000110010111010000000000011100
#  opcode: 100001, rs: 10010, rt: 11101, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 0, sig_data: 10, clock: 1
#  Memory: read_data: 00000000000000000000000000010110, address: 00000000000000000000000000110110, write_data: 00000000000000000000000000010110
#  Register: read_data_1: 00000000000000000000000000011010, read_data_2: 00000000000000000000000000010110, write_data: 00000000000000000000000000010110
#
#  PC:  7, instruction: 10010101111101110000000000101001
#  opcode: 100101, rs: 01111, rt: 10111, immediate: 0000000000101001
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 10, clock: 0
#  Memory: read_data: 00000000000000000000000000001001, address: 00000000000000000000000010001001, write_data: 00000000000000000000000000001001
#  Register: read_data_1: 00000000000000000000000001100000, read_data_2: 00000000000000000000000000010111, write_data: 00000000000000000000000000001001
#
#  PC:  7, instruction: 10010101111101110000000000101001
#  opcode: 100101, rs: 01111, rt: 10111, immediate: 0000000000101001
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 10, clock: 1
#  Memory: read_data: 00000000000000000000000000001001, address: 00000000000000000000000010001001, write_data: 00000000000000000000000000001001
#  Register: read_data_1: 00000000000000000000000001100000, read_data_2: 00000000000000000000000000001001, write_data: 00000000000000000000000000001001
#
#  PC:  8, instruction: 10010111101011100000000001011011
#  opcode: 100101, rs: 11101, rt: 01110, immediate: 0000000001011011
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 10, clock: 0
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000000000001110001, write_data: 00000000000000000000000000010001
#  Register: read_data_1: 00000000000000000000000000010110, read_data_2: 00000000000000000000000000011100, write_data: 00000000000000000000000000010001
#
#  PC:  8, instruction: 10010111101011100000000001011011
#  opcode: 100101, rs: 11101, rt: 01110, immediate: 0000000001011011
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 10, clock: 1
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000000000001110001, write_data: 00000000000000000000000000010001
#  PC:  9, instruction: 00111101010010110100111010010010
#  opcode: 001111, rs: 01010, rt: 01011, immediate: 0100111010010010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000100111010101101, write_data: 01001110100100100000000000000000
#  Register: read_data_1: 00000000000000000000000000011011, read_data_2: 00000000000000000000100000000000, write_data: 01001110100100100000000000000000
#
#  PC:  9, instruction: 00111101010010110100111010010010
#  opcode: 001111, rs: 01010, rt: 01011, immediate: 0100111010010010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000100111010101101, write_data: 01001110100100100000000000000000
#  Register: read_data_1: 00000000000000000000000000011011, read_data_2: 01001110100100100000000000000000, write_data: 01001110100100100000000000000000
#
#  PC: 10, instruction: 00111101011000001010001000100101
#  opcode: 001111, rs: 01011, rt: 00000, immediate: 1010001000100101
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010001, address: 01001110100100101010001000100101, write_data: 10100010001001010000000000000000
#  Register: read_data_1: 01001110100100100000000000000000, read_data_2: 00000000000000000000000000000000, write_data: 10100010001001010000000000000000
#
#  PC: 10, instruction: 00111101011000001010001000100101
#  opcode: 001111, rs: 01011, rt: 00000, immediate: 1010001000100101
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010001, address: 01001110100100101010001000100101, write_data: 10100010001001010000000000000000
#  Register: read_data_1: 01001110100100100000000000000000, read_data_2: 10100010001001010000000000000000, write_data: 10100010001001010000000000000000
#
#  PC: 11, instruction: 10001110011111110000000010011111
#  opcode: 100011, rs: 10011, rt: 11111, immediate: 0000000010011111
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 0
#  Memory: read_data: 00000000000000000000000000010010, address: 00000000000000000000000010110010, write_data: 00000000000000000000000000010010
#  Register: read_data_1: 00000000000000000000000000010011, read_data_2: 00000000000000000000000000011111, write_data: 00000000000000000000000000010010
#
#  PC: 11, instruction: 10001110011111110000000010011111
#  opcode: 100011, rs: 10011, rt: 11111, immediate: 0000000010011111
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 1
#  Memory: read_data: 00000000000000000000000000010010, address: 00000000000000000000000010110010, write_data: 00000000000000000000000000010010
#  Register: read_data_1: 00000000000000000000000000010011, read_data_2: 00000000000000000000000000010010, write_data: 00000000000000000000000000010010
#
#  PC: 12, instruction: 10001110101110000000000010000010
#  opcode: 100011, rs: 10101, rt: 11000, immediate: 0000000010000010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010010111, write_data: 00000000000000000000000000010111
#  Register: read_data_1: 00000000000000000000000000010101, read_data_2: 00000000000000000000000000011000, write_data: 00000000000000000000000000010111
#
#  PC: 12, instruction: 10001110101110000000000010000010
#  opcode: 100011, rs: 10101, rt: 11000, immediate: 0000000010000010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010010111, write_data: 00000000000000000000000000010111

#  PC: 13, instruction: 10100001101010100000000000011011
#  opcode: 101000, rs: 01101, rt: 01010, immediate: 0000000000011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000000100100, write_data: 00000000000110110000000000000000
#  Register: read_data_1: 00000000000000000000000000001001, read_data_2: 00000000000000000000000000011011, write_data: 00000000000110110000000000000000
#
#  PC: 13, instruction: 10100001101010100000000000011011
#  opcode: 101000, rs: 01101, rt: 01010, immediate: 0000000000011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000000100100, write_data: 00000000000110110000000000000000
#  Register: read_data_1: 00000000000000000000000000001001, read_data_2: 00000000000000000000000000011011, write_data: 00000000000110110000000000000000
#
#  PC: 14, instruction: 10100011001111100000000010011011
#  opcode: 101000, rs: 11001, rt: 11110, immediate: 0000000010011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010110100, write_data: 00000000100110110000000000000000
#  Register: read_data_1: 00000000000000000000000000011001, read_data_2: 00000000000000000000000000011110, write_data: 00000000100110110000000000000000
#
#  PC: 14, instruction: 10100011001111100000000010011011
#  opcode: 101000, rs: 11001, rt: 11110, immediate: 0000000010011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010110100, write_data: 00000000100110110000000000000000
#  Register: read_data_1: 00000000000000000000000000011001, read_data_2: 00000000000000000000000000011110, write_data: 00000000100110110000000000000000
#
#  PC: 15, instruction: 10100101001010110000000001001010
#  opcode: 101001, rs: 01001, rt: 01011, immediate: 0000000001001010
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000001001111, write_data: 00000000010010100000000000000000
#  Register: read_data_1: 00000000000000000000000000000101, read_data_2: 01001110100100100000000000000000, write_data: 00000000010010100000000000000000
#
#  PC: 15, instruction: 10100101001010110000000001001010
#  opcode: 101001, rs: 01001, rt: 01011, immediate: 0000000001001010
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000001001111, write_data: 00000000010010100000000000000000
#  Register: read_data_1: 00000000000000000000000000000101, read_data_2: 01001110100100100000000000000000, write_data: 00000000010010100000000000000000
#
#  PC: 16, instruction: 10100110010010010000000010101100
#  opcode: 101001, rs: 10010, rt: 01001, immediate: 0000000010101100
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000011000110, write_data: 00000000101011000000000000000000
#  Register: read_data_1: 00000000000000000000000000011010, read_data_2: 00000000000000000000000000000101, write_data: 00000000101011000000000000000000
#
#  PC: 16, instruction: 10100110010010010000000010101100
#  opcode: 101001, rs: 10010, rt: 01001, immediate: 0000000010101100
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000011000110, write_data: 00000000101011000000000000000000
```



```
#  PC: 17, instruction: 10101101111000000000000000101111
#  opcode: 101011, rs: 01111, rt: 00000, immediate: 0000000000101111
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010001111, write_data: 00000000010111110000000000000000
#  Register: read_data_1: 00000000000000000000000001100000, read_data_2: 10100010001001010000000000000000, write_data: 00000000010111110000000000000000
#
#  PC: 17, instruction: 10101101111000000000000000101111
#  opcode: 101011, rs: 01111, rt: 00000, immediate: 0000000000101111
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010001111, write_data: 00000000010111110000000000000000
#  Register: read_data_1: 00000000000000000000000001100000, read_data_2: 10100010001001010000000000000000, write_data: 00000000010111110000000000000000
#
#  PC: 18, instruction: 10101110010011000000000000011100
#  opcode: 101011, rs: 10010, rt: 01100, immediate: 0000000000011100
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000000110110, write_data: 00000000000111000000000000000000
#  Register: read_data_1: 00000000000000000000000000011010, read_data_2: 00000000000000000000000000100000, write_data: 00000000000111000000000000000000
#
#  PC: 18, instruction: 10101110010011000000000000011100
#  opcode: 101011, rs: 10010, rt: 01100, immediate: 0000000000011100
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000000110110, write_data: 00000000000111000000000000000000
#  Register: read_data_1: 00000000000000000000000000011010, read_data_2: 00000000000000000000000000100000, write_data: 00000000000111000000000000000000
#
```

You can check the files to see if the tests and the processor are working correctly. For example, when a load instruction is executed, you will see that the data changes as requested in the instruction by looking at the corresponding address in the register file or the corresponding address in the memory file when the store instructions are running.

Analyze 1:



```
#  PC:  1, instruction: 10001100010000100000000000011100
#  opcode: 100011, rs: 00010, rt: 00010, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 0
#  Memory: read_data: 00000000000000000000000000011110, address: 00000000000000000000000000011110, write_data: 00000000000000000000000000011110
#  Register: read_data_1: 00000000000000000000000000000010, read_data_2: 00000000000000000000000000000010, write_data: 00000000000000000000000000011110
#
#  PC:  1, instruction: 10001100010000100000000000011100
#  opcode: 100011, rs: 00010, rt: 00010, immediate: 0000000000011100
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 1, sig_sign: 1, sig_data: 11, clock: 1
#  Memory: read_data: 00000000000000000000000000011010, address: 00000000000000000000000000111010, write_data: 00000000000000000000000000011010
#  Register: read_data_1: 00000000000000000000000000011110, read_data_2: 00000000000000000000000000011110, write_data: 00000000000000000000000000011010
```

This is load byte instruction.

Firstly look at opcode, rs, rt, immediate value. Also, signal.

Clock = 0, Look at register for data

Rs content(read_data_1): 00000000000000000000000000000010

Rt content(read_data_2): 00000000000000000000000000000010

In memory, read_data_1 and immediate data were added according to the rule set by the instruction. Memory read_data is also shown.

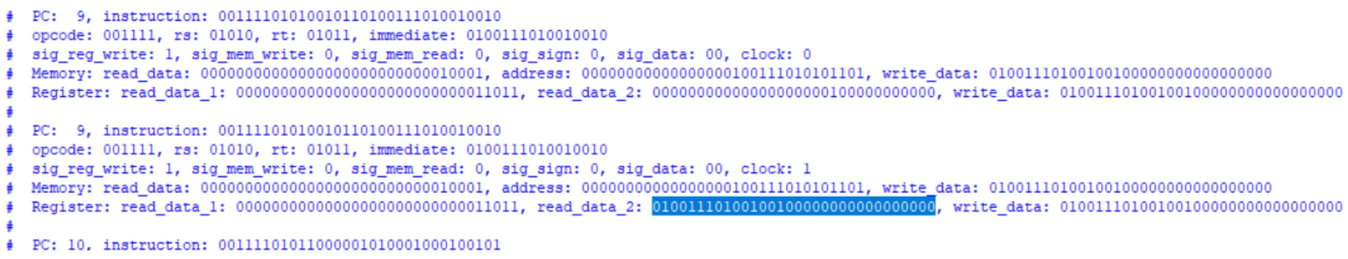
Data: 00000000000000000000000000011110

Clock = 1,

Data from memory is written to rt address in register via mux according to related signals.

! For this test, rt and rs addres are same.

Analyze 2:



```
#  PC:  9, instruction: 00111101010010110100111010010010
#  opcode: 001111, rs: 01010, rt: 01011, immediate: 0100111010010010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000100111010101101, write_data: 01001110100100100000000000000000
#  Register: read_data_1: 00000000000000000000000000011011, read_data_2: 00000000000000000000100000000000, write_data: 01001110100100100000000000000000
#
#  PC:  9, instruction: 00111101010010110100111010010010
#  opcode: 001111, rs: 01010, rt: 01011, immediate: 0100111010010010
#  sig_reg_write: 1, sig_mem_write: 0, sig_mem_read: 0, sig_sign: 0, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010001, address: 00000000000000000100111010101101, write_data: 01001110100100100000000000000000
#  Register: read_data_1: 00000000000000000000000000011011, read_data_2: 01001110100100100000000000000000, write_data: 01001110100100100000000000000000
#
#  PC: 10, instruction: 00111101011000001010001000100101
```

This is load upper immediate instruction.

Firstly look at opcode, rs, rt, immediate value. Also, signal.

Clock = 0, Look at register for data

Rs content(read_data_1): 00000000000000000000000000011011

Rt content(read_data_2): 00000000000000000000100000000000

Immediate Data(write_data): 01001110100100100000000000000000

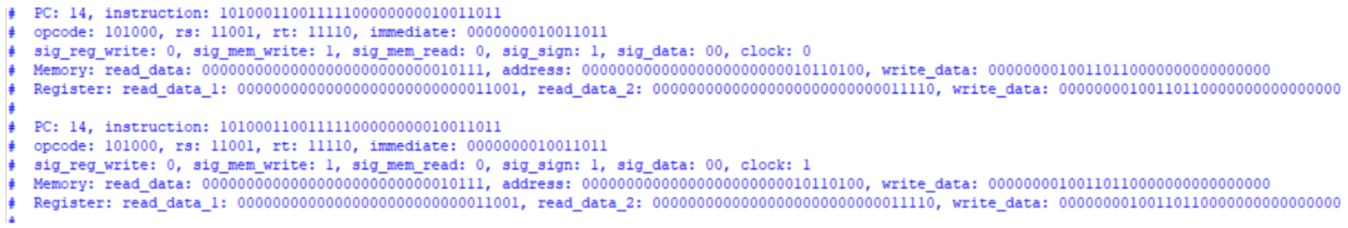
Clock = 1,

Immediate data is extended 16 bits to the right. Then this data reaches the mux and reaches the write data section of the register by the signal coming from the control.

Write data is written to the rt register.

Rt content(read_data_2): 01001110100100100000000000000000

Analyze 3:



```
#  PC: 14, instruction: 10100011001111100000000010011011
#  opcode: 101000, rs: 11001, rt: 11110, immediate: 0000000010011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 0
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010110100, write_data: 00000000100110110000000000000000
#  Register: read_data_1: 00000000000000000000000000011001, read_data_2: 00000000000000000000000000011110, write_data: 00000000100110110000000000000000
#
#  PC: 14, instruction: 10100011001111100000000010011011
#  opcode: 101000, rs: 11001, rt: 11110, immediate: 0000000010011011
#  sig_reg_write: 0, sig_mem_write: 1, sig_mem_read: 0, sig_sign: 1, sig_data: 00, clock: 1
#  Memory: read_data: 00000000000000000000000000010111, address: 00000000000000000000000010110100, write_data: 00000000100110110000000000000000
#  Register: read_data_1: 00000000000000000000000000011001, read_data_2: 00000000000000000000000000011110, write_data: 00000000100110110000000000000000
#
```

This is store by instruction.

Firstly look at opcode, rs, rt, immediate value. Also, signal.

Clock = 0, Look at register for data

Rs content(read_data_1): 00000000000000000000000000011001

Rt content(read_data_2): 00000000000000000000000000011110

Immediate Data: 0000000010011011

[R[rs]+SıgnExtendImmediate] (memory address): 00000000000000000000000010110100

R[rt] (memory write_data): 00000000100110110000000000000000

Clock = 1,

It collects Rs and immediate data according to the rule of instruction. This data specifies the memory address. Writes the data of the Rt register to this memory address.

data_mem[address] <= rt_data[7:0];

data_mem[address]: 00000000000000000000000000011110

Fınally instruction Result:

M[R[rs]+SıgnExtendImmediate]: 00000000000000000000000000011110

M[00000000000000000000000010110100]: 00000000000000000000000000011110

To fully understand the test, see the values of the corresponding addresses in the files. It is better if you look at the changed version again after the test.