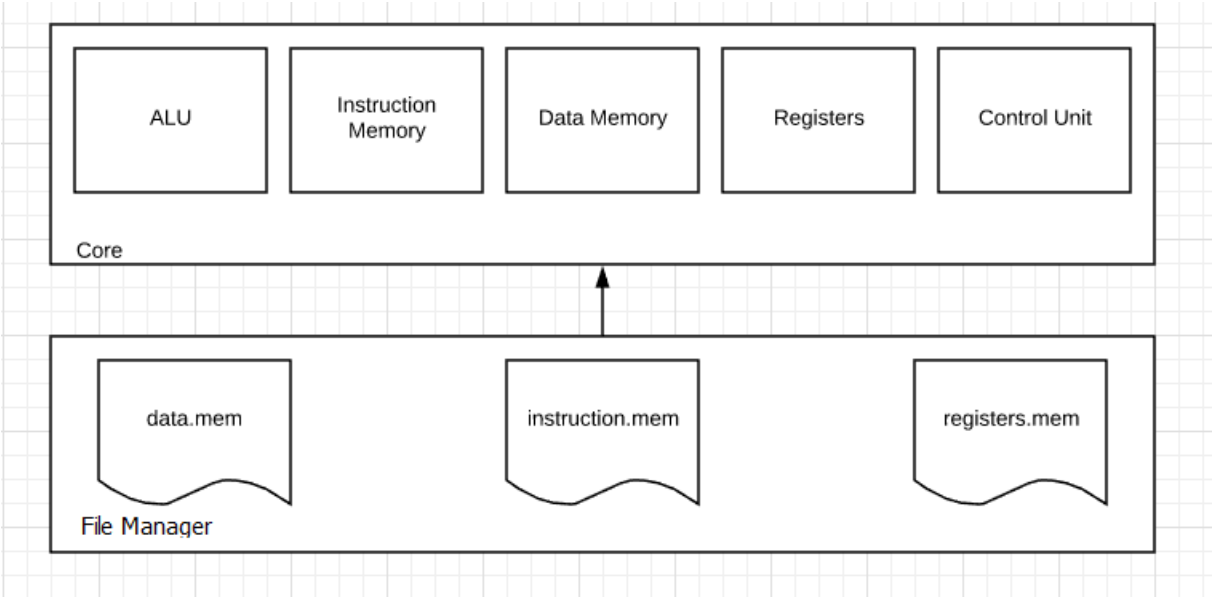
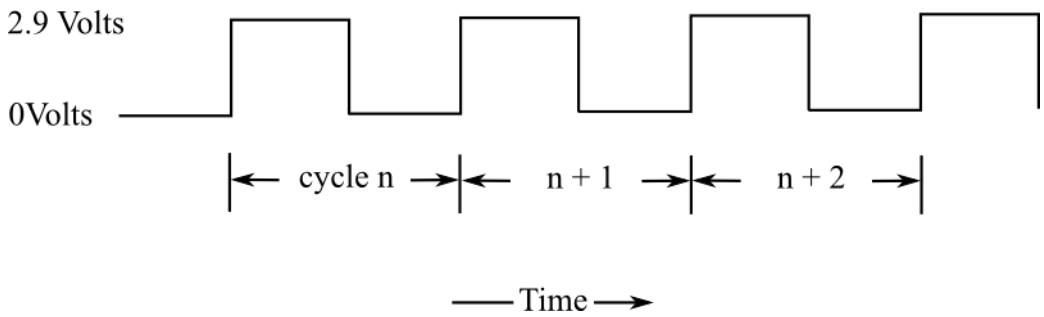
141044076
TANER DURKUT

BİL 331 – BİLGİSAYAR ORGANİZASYONU FİNAL PROJESİ RAPORU

1. GİRİŞ



Bir instruction'ın yaşam döngüsü



İnstruction, instruction.mem dosyasından okunur, mips_core modülü içinde parçalarına ayrılır. Opcode ve function control unit modülüne gönderilir ve bu değerlere göre sinyaller belirlenir. Daha sonra ALU modülüne registers.mem dosyasından okunan source register'larının değerleri gönderilir. ALU'dan çıkan sonuç destination register'a yazılır. Clock 0 dan 1 e değiştiği zaman (posedge) PC değeri 1 artar ve bir sonraki instruction okunur.

Testbench dosyasında #50 clock = ~clock; #50 clock = ~clock; yapmamızın sebebi, 1 cycle'ı tamamlamaktır. Clock 0'dan başlıyor ve 50 birim zaman sonra 1 oluyor, ve 50 birim zaman sonra yeniden 0 oluyor ve yeniden 1 ve yeniden 0 şeklinde devam ediyor. Yukarıdaki görselde de görüldüğü gibi 1 clock cycle, 0'dan 1 e anında başlıyor ve yine 0'dan 1 e anında tamamlanıyor.

141044076
TANER DURKUT

Eksikler

Final projemde, zamanım yetmediği için maalesef J type, I type instruction'ları çalıştıramadım.

2. Metod

Mips_registers modülü:

Outputs:

Read_data_1: rs contenti

Read_data_2: rt contenti

Inputs:

write_data: yazılacak data

read_reg_1: okunacak register (rs)

read_reg_2: okunacak register (rt)

write_reg: yazılacak register (rd for R types)

signal_reg_write: registera yazma sinyali

clock: clock girişi

Bu modülde registers.mem dosyası okunur rs ve rt registerlarının contentleri output olarak verilir. Ayrıca, signal_reg_write sinyali 1 ise ALU'da hesaplanan ve writedataya gelen değer rd register'ına yazılır.

Mips_instr_mem modülü:

inputs:

clock: clock girişi

program_counter: okunacak instruction'ın adresini belirten değer

outputs:

instruction: program_counter adresinden okunan instruction

Bu modülde program counter'ın gösterdiği yerdeki instruction okunur ve output olarak iletilir.

Mips_data_mem modülü:

inputs:

clock: clock girişi

mem_address: değerin okunacağı adres

write_data: memory'e yazılacak data

sig_mem_read: data memory okuma sinyali 1 ise okumaya izin vardır, 0 ise okuma izni yoktur.

sig_mem_write: data memory yazma sinyali 1 ise yazmaya izin vardır, 0 ise yazma izni yoktur.

Outputs:

141044076
TANER DURKUT

Read_data: memory'den okunan data

Bu modülde data memory'i ifade eden data.mem dosyası okunur. Eğer yazma izni varsa (yani sinyal 1 ise) write_data daki değer memory'ye yazılır. Eğer okuma izni varsa (yani sinyal 1 ise) mem_address in gösterdiği yerdeki değer okunur ve read_data ile dışarı output olarak iletilir.

Mips_control_unit modülü:

inputs:

opcode: okunan instruction'ın ilk 6 biti

func: okunan instruction'ın son 6 biti

Outputs:

RegDst: sonucun rt'ye mi rd'ye mi yazılacağını belirten sinyal (0: rt, 1:rd)

Branch: instruction branch instruction'ı ise 1 olan sinyal

MemRead: memory'den okuma izninin olup olmadığını ifade eden sinyal

MemtoReg: memory'den register'a yazma işlemini temsil eden sinyal

ALUOp: ALU'nun hangi işlemi yapacağını belirten 4 bitlik değer

MemWrite: memory'ye yazma izninin olup olmadığını ifade eden sinyal

ALUSrc: 1 ise source2 signextend imm olarak seçilir, 0 ise rt

RegWrite: register bloğuna yazma izninin olup olmadığını belirten sinyal

Bu modülde opcode veya func koduna göre sinyaller ayarlanır.

Instr.	Reg Des	ALUSrc	Memto Reg	Reg Wr	Mem Rd	Mem Wr	Branch	ALUop1	ALUop0
R-type	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Table: Control Lines Determined by the opcode

Mips_alu modülü:

inputs:

ALUOp: hangi işlemin yapılacağını belirten 4 bitlik değer

Content1: rs değeri

141044076
TANER DURKUT

Content2: rt değeri

Outputs:

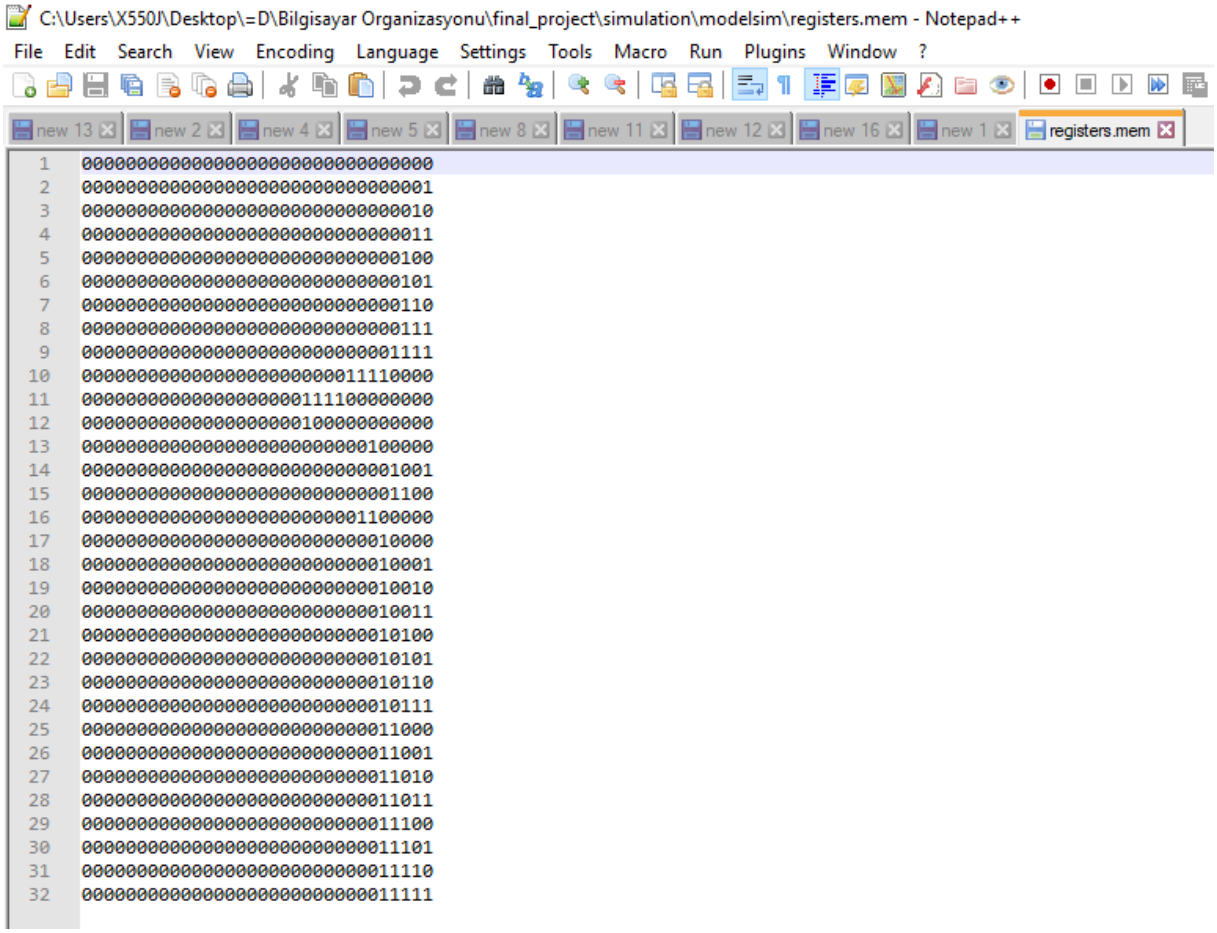
Result: işlemin sonucunu dışarı ileten birim

Signal_zero: branch instruction'ı durumunda 1 ileten sinyal

Bu modülde gelen işleme göre contentler kullanılarak sonuç belirlenir.

3.Sonuç

Registers BEFORE



Ss1.png

Registers AFTER

141044076
TANER DURKUT

C:\Users\X550J\Desktop\=D\Bilgisayar Organizasyonu\final_project\simulation\modelsim\res_registers.mem - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

new 13 | new 2 | new 4 | new 5 | new 8 | new 11 | new 12 | new 16 | new 1 | registers.mem | res_registers.mem

```
1  // memory data file (do not edit the following line - required for mem load use)
2  // instance=/mips_testbench/test/registers/registers
3  // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
4  00000000000000000000000000110110
5  00000000000000000000000000000001
6  00000000000000000000000000000010
7  00000000000000000000000000000011
8  00000000000000000000000000000100
9  00000000000000000000000000000101
10 00000000000000000000000000000110
11 00000000000000000000000001101100
12 00000000000000000000000001110000
13 00000000000000000000000011110000
14 00000000000000000000111100000000
15 00000000000000000000000000101001
16 00000000000000000000000000100000
17 00000000000000000000000000001001
18 00000000000000000000000000001100
19 00000000000000000000000001100000
20 00000000000000000000000000010000
21 00000000000000000000000000010001
22 00000000000000000000000000100001
23 00000000000000000000000000010011
24 00000000000000000000000000010100
25 00000000000000000000000000010101
26 00000000000000000000000000010110
27 00000000000000000000000000010111
28 00000000000000000000000000011000
29 00000000000000000000000000011001
30 00000000000000000000000000011010
31 00000000000000000000000000011011
32 00000000000000000000000000011100
33 00000000000000000000000000011101
34 00000000000000000000000000011110
35 00000000000000000000000000011111
36
```

Ss2.png

Testbench results:

ModelSim ALTERA STARTER EDITION 10.1d

File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help

```
sim:/mips_testbench/read_reg_1 \
sim:/mips_testbench/read_reg_2 \
sim:/mips_testbench/result \
sim:/mips_testbench/shamt \
sim:/mips_testbench/write_reg
VSIM 12> step -current
# ================================================================
#  time:  0 instruction: 00000001111101110001110000100000 result:        108
#  RS: 01111 RT: 01110 RD: 00111 Read_data_1: 00000000000000000000000001100000, Read_data_2: 00000000000000000000000000001100,  RegWrite signal: 1, RD adress: 00111
# ================================================================
#  time: 50 instruction: 00000010000011111010000000100000 result:        112
#  RS: 10000 RT: 01111 RD: 01000 Read_data_1: 00000000000000000000000000010000, Read_data_2: 00000000000000000000000001100000,  RegWrite signal: 1, RD adress: 01000
# ================================================================
#  time: 150 instruction: 00000010000100011001000000100000 result:         33
#  RS: 10000 RT: 10001 RD: 10010 Read_data_1: 00000000000000000000000000010000, Read_data_2: 00000000000000000000000000010001,  RegWrite signal: 1, RD adress: 10010
# ================================================================
#  time: 250 instruction: 00100010011100110000000000000001 result:         38
#  RS: 10011 RT: 10011 RD: 00000 Read_data_1: 00000000000000000000000000010011, Read_data_2: 00000000000000000000000000010011,  RegWrite signal: 1, RD adress: 00000
# ================================================================
#  time: 350 instruction: 00000001010010101011100000101010 result:         41
#  RS: 10100 RT: 10101 RD: 01011 Read_data_1: 00000000000000000000000000010100, Read_data_2: 00000000000000000000000000010101,  RegWrite signal: 1, RD adress: 01011
# ================================================================
#  time: 450 instruction: 00101010110011000000000000010000 result:         54
#  RS: 10110 RT: 01100 RD: 00000 Read_data_1: 00000000000000000000000000010110, Read_data_2: 00000000000000000000000000100000,  RegWrite signal: 1, RD adress: 00000
# ================================================================
#  time: 550 instruction: 00000010000100011001000000100010 result:         33
#  RS: 10000 RT: 10001 RD: 10010 Read_data_1: 00000000000000000000000000010000, Read_data_2: 00000000000000000000000000010001,  RegWrite signal: 1, RD adress: 10010
# ================================================================
#  time: 650 instruction: 10001100101001110000000000001010 result:        113
#  RS: 00101 RT: 00111 RD: 00000 Read_data_1: 00000000000000000000000000000101, Read_data_2: 00000000000000000000000001101100,  RegWrite signal: 1, RD adress: 00000

VSIM 13>
```

Ss3.png