CSE 331

HW4-RAPOR

Moduller:

-hw3 ‘deki mips32 modülü yerine mips32_single_cycle modülü gelmiştir.Bu modül r type 7 başka instruction ile çalışabilir.

-zero_extended ve sign_extended modülleri instruction’nın immedite kısmını 32 bite genişletmek için kullanılır.

-başka bir yeni modül ise data_memory modülüdür.bu modül verilen adres bilgisine göre data.mem dosyasındaki(256 satır) 32 bitlik verileri okuyabilir ve yine aynı dosyaya yazabilir.

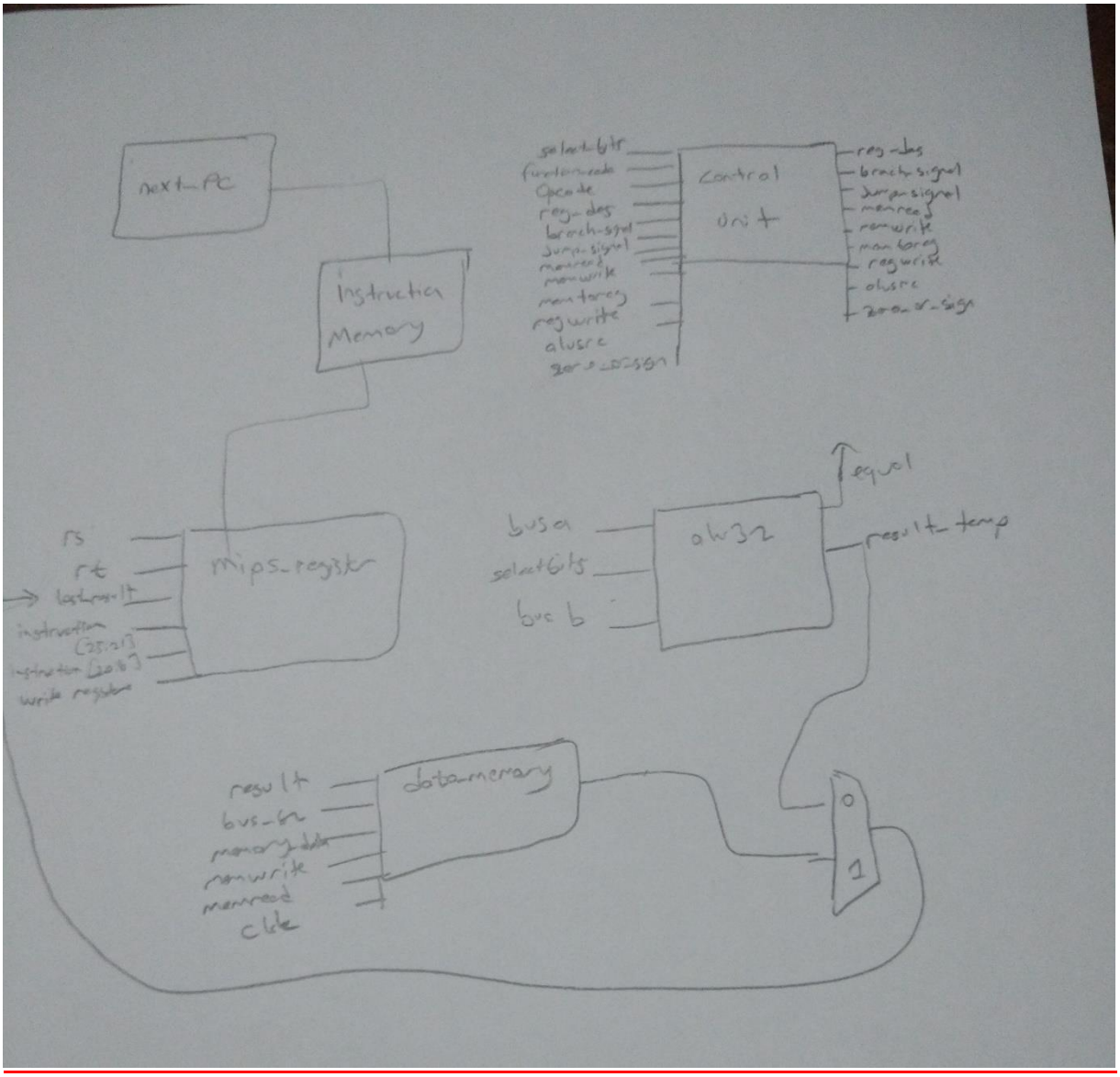
-başka yeni bir modül ise instruction_memory modülüdür.Bu modül PC counterinin gösterdiği değere göre instruction.mem dosyasında instruction okur.instruction.mem dosyasında toplam 30 instruction vardır!

-next_PC modülü bir sonraki instructionu işleyebilmek için PC counteri arttırır.arttırma işlemini eğer instruction branch ise PC+1+signextended(immediate) olarak ,jump ise PC[31:26] ::instruction[25:0] concanate ederek elde ettiği atar .Eğer ikiside değilse PC+1 olarak arttırır.eğer instruction dosyasının sonuna geldiyse yani 30. Adreste ise program sonlanır.

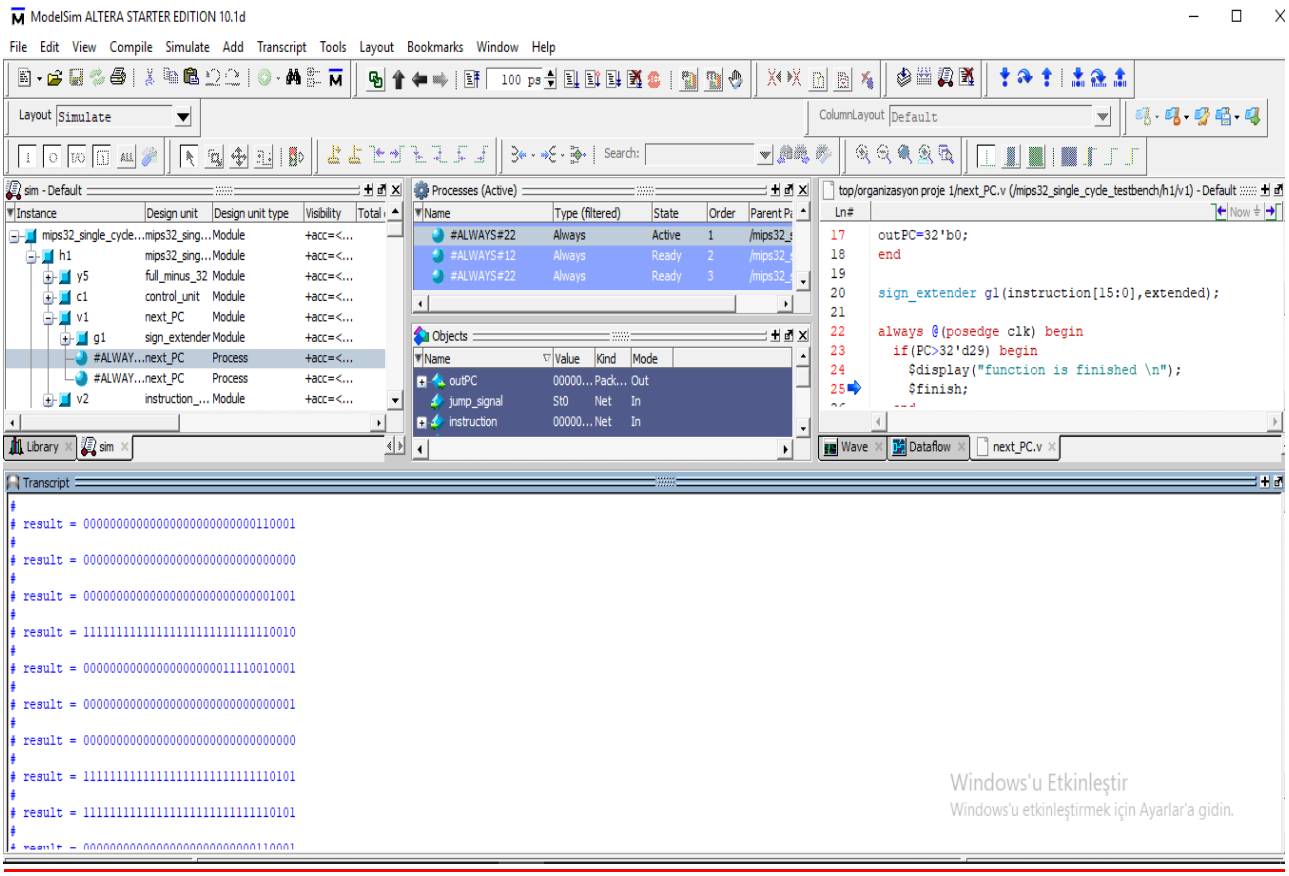
-Diğer modüller hw3 deki modüllerin üzerine inşa edilmişlerdir.

-NOT:jump ve branch instructionlarınında saçmada olsa sonuçları konsolda basılmaktadır lütfen fazladan result basılmış diye düşünmeyin sırayı takip ettiğiniz de sonuçların doğru olduğunu göreceksiniz.

Modul çizimleri:



Simülasyon sonuçları:



Konsolun yakınlaşmış hali:

```
Transcript
add wave -position insertpoint  \
sim:/mips32_single_cycle_testbench/result \
sim:/mips32_single_cycle_testbench/clk2 \
sim:/mips32_single_cycle_testbench/clk
VSIM 5> step -current
# result = 11100000111100001100000011111000
#
# result = 00000000000000000000000000110001
#
# result = 00000000000000000000000000000000
#
# result = 00000000000000000000000000001001
#
# result = 11111111111111111111111111110010
#
# result = 00000000000000000000011110010001
#
# result = 00000000000000000000000000000001
#
# result = 00000000000000000000000000000000
#
# result = 11111111111111111111111111110101
#
# result = 11111111111111111111111111110101
#
# result = 00000000000000000000000000110001
```

```
Transcript
#
# result = 00000000000000000000000000110001
#
# result = 00000000000000000000000000000010
#
# result = 00000000000000000000011110010001
#
# result = 00000000000000000000000000001101
#
# result = 01110010111100000000011111111111
#
# result = 00000000000000000000000000000000
#
# result = 00000000000000000000000000110001
#
# result = 00000000000000000000000000000000
#
# function is finished
#
# ** Note: $finish    : C:/Users/ibrah/Desktop/organizasyon proje 1/next_PC.v(25)
#    Time: 3600 ps  Iteration: 1  Instance: /mips32_single_cycle_testbench/h1/v1
# 1
# Break in Module next_PC at C:/Users/ibrah/Desktop/organizasyon proje 1/next_PC.v line 25

VSIM 6>
```

İnstruction memory dosyası ,data memory dosyası ve register memory dosyası:

mips32_single_cycle.v | Compilation Report - proje9 | simulation/modelsim/instruction.mem | mips_registers.v

```
1   //10 instructions with their definition.
2
3
4
5
6   //shift left logical instruction rs=1.reg ,rt=2.reg ,rd=3.reg  shamt=3
7   00000000001000100001100011000000
8   //add instruction rs=4.reg, rt=5.reg, rd=6.reg, shamt=x
9   00000000100001010011000000100000
10  //and instruction rs=7.reg, rt=8.reg, rd=9.reg, shamt=x
11  00000000111010000100100000100100
12  //addu instrouction rs=10.reg, rt=11.reg, rd=12.reg, shamt=x
13  00000001010010110110000000100001
14  //nor instruction rs=13.reg, rt=14.reg, rd=15.reg, shamt=x
15  00000001101011100111100000100111
16  //or instruction rs=16.reg, rt=17.reg, rd=18.reg, shamt=x
17  00000010000100011001000000100101
18  //sltu instruction rs=19.reg, rt=20.reg, rd=21.reg shamt=x
19  00000010011101001010100000101011
20  //shift right logical instruction rs=22.reg, rt=23.reg, rd=24.reg shamt=7
21  00000010110101111110000011100010
22  //sub instruction rs=25.reg, rt=26.reg, rd=27.reg shampt=x
23  00000011001110101101100000100010
24  //subu instruction rs=28.reg, rt=29.reg, rd=30.reg shamt=x
25  00000011100111011111000000100011
26
```

```
//subu instruction rs=28.reg, rt=29.reg, rd=30.reg shamt=x
00000011100111011111000000100011

//addiu rs=4.reg rt=31.reg
00100100100111110000000000101101
//andi rs=7.reg rt=30.reg
00110000111111101111111111000111
//ori rs=16.reg rt=12.reg
00110110000011000000000000010001

//store word rs 7.reg rt=8.reg immediate=11
101011_00111_01000_0000000000001011
//load word rs=7.reg rt=18.reg immediate=9
100011_00111_10010_0000000000001001

//jump target adress=19
000010_00000_00000_00000_00000_010100

//jump instruction skip these 4 instruction it jump 20. instruction(21)
00000011100111011111000000100011
00000011100111011111000000100011
00000011100111011111000000100011
00000011100111011111000000100011

//branch rs=6.reg rt=31.reg immediate=8
```

```
100011_00111_10010_0000000000001001

//jump target adress=19
000010_00000_00000_00000_00000_010100

//jump instruction skip these 4 instruction it jump 20. instruction(21)
00000011100111011111000000100011
00000011100111011111000000100011
00000011100111011111000000100011
00000011100111011111000000100011

//branch rs=6.reg rt=31.reg immediate=8
000100_00110_11111_0000000000001000

//branch instruction skip 8 of these 9 instruction it points 9 '8+1' after it.
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

Register-memory dosyası:

```
1  // memory data file (do not edit the following line - required for mem load use)
2  // instance=/mips32_single_cycle_testbench/h1/f1/registers
3  // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
4  00000000000000000000000000000000
5  00000000000000000000000000001101
6  11111100000111100001100000011111
7  11100000111100001100000011111000
8  00000000000000000000000000000100
9  00000000000000000000000000101101
10 00000000000000000000000000110001
11 00000000000000000000000000000010
12 00000000000000000000000000001101
13 00000000000000000000000000000000
14 00000000000000000000000000000100
15 00000000000000000000000000000101
16 00000000000000000000011110010001
17 00000000000000000000000000001001
18 00000000000000000000000000001100
19 11111111111111111111111111110010
20 00000000000000000000011110000000
21 00000000000000000000000000010001
22 01110010111110000000011111111111
23 00000000000000000000000000010011
24 00000000000000000000000000010100
25 00000000000000000000000000000001
26 00000000000000000000000000000000
27 00000000000000000000000000010111
```

```
10 00000000000000000000000000110001
11 00000000000000000000000000000010
12 00000000000000000000000000001101
13 00000000000000000000000000000000
14 00000000000000000000000000000100
15 00000000000000000000000000000101
16 00000000000000000000011110010001
17 00000000000000000000000000001001
18 00000000000000000000000000001100
19 11111111111111111111111111110010
20 00000000000000000000011110000000
21 00000000000000000000000000010001
22 01110010111110000000011111111111
23 00000000000000000000000000010011
24 00000000000000000000000000010100
25 00000000000000000000000000000001
26 00000000000000000000000000000000
27 00000000000000000000000000010111
28 00000000000000000000000000000000
29 00000000000000000000000000001111
30 00000000000000000000000000011010
31 11111111111111111111111111110101
32 00000000000000000000000000001111
33 00000000000000000000000000011010
34 00000000000000000000000000000010
35 00000000000000000000000000110001
36
```

-data memory dosyası(tüm satırlarda aynı sayı vardır sw instruction'nının nın eriştiği satır hariç.

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/mips32_single_cycle_testbench/h1/t1/MEMORY
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
00000000000000000000000000001101
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
01110010111100000000011111111111
```

-toplamda 259.satıra kadar gitmektedir bu dosya aynı şekilde.

-