CSE331 - Computer Organization
FINAL PROJECT

Proje için yeni tasarladığım modüller:

Control Unit : Bu modülde ALU için select bitleri gerekli sadeleştirme yaparak buldum. Eski projeye ek olarak yeni instructionlar geldiği için değiştirdim.
S[2] = (F[1] + (F[5] + F[0])')&ALU_OP[1] + ALU_OP[0]
S[1] = (F[1] XOR F[2])'&ALU_OP[1] + ALU_OP[4] +(ALU_OP[4]'& ALU_OP[3]'&ALU_OP[2]'& ALU_OP[41]'& ALU_OP[0]')
S[0] = (F[2]&F[0] + F[5]'&F[1]&F[0]')&ALU_OP[1] + ALU_OP[3]

SignExtend : addiu, lw, sw, instructionlarında kullanmak için tasarladım. Instructionun [15:0] i alınır. Bu 16 bitlik sayı 32 bite çıkartılır. İlk 16 bit instructionun 16 biti ile aynıdır. Kalan kısımlar instruction[15] ile aynidir.

ZeroExtend : andi, ori instructionlarında kullanmak için tasarladım. Instructionun [15:0] i alınır. Bu 16 bitlik sayı 32 bite çıkartılır. İlk 16 bit instructionun 16 biti ile aynıdır. Kalan kısımlar 0 ile doldurulur.

Mips_Control_Unit : Gelen OpCode yani instruction[5:0] ' a gore gerekli sadeleştirmeler yapılıp ALU_OP, ALU_SRC, RegDst, MemtoReg, MemWrite, Branch, Jump, MemRead, RegWrite sinyalleri oluşturulur. Bu sinyaller top_moduledeki muxlar için select bitler olarak kullanılır.

Is_Equal : Bu modülü branch instructionu için tasarladım. Branch instructionu eğer iki sayı eşitse 1 değilse 0 gösterir. Bu durumda bu modülün içindeki 32' de 0 ilse 1 sinyali eğer hepsi 0 değilse 0 sinyali verir.

Mux_2_1_5bit : Bu mux' u önceden tasarlamış olduğum mux_2_1 modulunu kullanarak tasarladım. Buradaki mux un görevi gelen bite göre rt ya da rd yi seçmektir. RegDest sinyali 0 gelirse rt 1 gelirse rd seçer.

Instruction_Memory : mips32_testbench modülündeki yapmış olduğum $readmemb(".\\instruction.mem", EXE.instruction_MEMORY.instruction_array); okuma ile instruction_memory modülündeki instruction_array 'i dosyadaki instructionlar ile doldurdum.

Data_Memory : mips32_testbench modülündeki yapmış olduğum $readmemb(".\\data.mem",EXE.data_MEMORY.data_array); okuma ile sig_mem_read sinyali geldiği takdirde data_array'inin içinden read_data' yi çekiyorum bu işlem lw için. sig_mem_write sinyali geldiğinde ise data_array'inin içinde yazılacak wire_data 'yi yazıyorum bu işlem sw için.

Program_Counter : Bu modülde yaptığım şeyler program counter' i 1 arttırmaktır. JUMP ve BRANCH için gerekli program counteri arttırma işlemlerini yapamadım.

Mips32_Single_Cycle : Bu modülde yaptığım işlemler şu şekildedir.

instruction_memory modülü ile instruction' un instruction_array' inden aldım.

mips_control_unit modülü ile gerekli sinyalleri ürettim.

control_unit modülü ile ALU için gerekli select sinyalleri ürettim.

zero_extend modülü ile zero_extend_out 32 bitlik sayıyı oluşturdum.

Signextend modülü ile sign_extend_out 32 bitlik sayıyı oluşturudum.

or or_gate(select_signal_1,ALU_OP[2],ALU_OP[3]); ile sign_extend mi zero_extend mi olduğuna karar verecek select_signal_1 i oluşturdum.

mux_2_1_32bit sign_or_zero mux'u ile hangi extend olduğunu buldum.

mux_2_1_5bit rt_or_rd mux'u ile rt mi rd mi olduğunu buldum.

mips_registers mips_REG ile register üzerindeki işlemleri yaptım.

concatenate_32 ile shamt i 32 bite çıkardım.

mux_2_1_32bit rt_or_shamt, mux_2_1_32bit rs_or_rt ve

mux_2_1_32bit rt_or_shamt_OR_sign_or_zero ile grekli seçimleri yaptım.

alu32 ALU ile ALU daki işlemleri yaptırdım.

is_equal modulunde ALU dan gelen sonucun 0 mi değil mi olduğuna baktım. Branch için.

program_counter modülünü çağardım.

data_memory için ALU ' dan çıkan sonuç adresimiz oluyor. Bu sayede gelen sinyale göre adrese ya yazma ya da adresten veri alma işlemi yapıyorum.

```
Alu_Control = 010
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000000001000100001100000100000
----$rs----     = 00000000000000000000000000000001
----$rt----     = 00000000000000000000000000000010
zeroextend_out  = 00000000000000000001100000100000
signextend_out  = 00000000000000000001100000100000
---Result---    = 00000000000000000000000000000011
Program_counter = 00000000000000000000000000000000
*********************************************************

Alu_Control = 010
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000000100001010011000000100001
----$rs----     = 00000000000000000000000000000100
----$rt----     = 00000000000000000000000000000101
zeroextend_out  = 00000000000000000011000000100001
signextend_out  = 00000000000000000011000000100001
---Result---    = 00000000000000000000000000001001
Program_counter = 00000000000000000000000000000001
*********************************************************

Alu_Control = 100
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000000111010000100100000100010
----$rs----     = 00000000000000000000000000000111
----$rt----     = 00000000000000000000000000001111
zeroextend_out  = 00000000000000000100100000100010
signextend_out  = 00000000000000000100100000100010
---Result---    = 11111111111111111111111111111000
Program_counter = 00000000000000000000000000000010
*********************************************************




Alu_Control = 100
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000001010010110110000000100011
----$rs----     = 00000000000000000000111100000000
----$rt----     = 00000000000000000000100000000000
zeroextend_out  = 00000000000000000110000000100011
signextend_out  = 00000000000000000110000000100011
---Result---    = 00000000000000000000011100000000
Program_counter = 00000000000000000000000000000011
*********************************************************

Alu_Control = 000
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000001101011100111100000100100
----$rs----     = 00000000000000000000000000001001
----$rt----     = 00000000000000000000000000001100
zeroextend_out  = 00000000000000000111100000100100
signextend_out  = 00000000000000000111100000100100
---Result---    = 00000000000000000000000000001000
Program_counter = 00000000000000000000000000000100
*********************************************************

Alu_Control = 001
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000010000100011001000000100101
----$rs----     = 00000000000000000000000000010000
----$rt----     = 00000000000000000000000000010001
zeroextend_out  = 00000000000000000001000000100101
signextend_out  = 11111111111111111001000000100101
---Result---    = 00000000000000000000000000010001
Program_counter = 00000000000000000000000000000101
*********************************************************
```

```
Alu_Control = 111
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000010011101001010100000100111
----$rs----     = 00000000000000000000000000010011
----$rt----     = 00000000000000000000000000010100
zeroextend_out  = 00000000000000000010100000100111
signextend_out  = 11111111111111111010100000100111
---Result---    = 11111111111111111111111111101000
Program_counter = 00000000000000000000000000000110
************************************************************

Alu_Control = 110
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000010110101111100000011000000
----$rs----     = 00000000000000000000000000010110
----$rt----     = 00000000000000000000000000010111
zeroextend_out  = 00000000000000000100000011000000
signextend_out  = 11111111111111111100000011000000
---Result---    = 00000000000000000000000010111000
Program_counter = 00000000000000000000000000000111
************************************************************

Alu_Control = 101
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000011001110101101100001000010
----$rs----     = 00000000000000000000000000011001
----$rt----     = 00000000000000000000000000011010
zeroextend_out  = 00000000000000000101100001000010
signextend_out  = 11111111111111111101100001000010
---Result---    = 00000000000000000000000000001101
Program_counter = 00000000000000000000000000001000
************************************************************



Alu_Control = 100
MIPS_Control : RegDst = 1, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00000011100111011111000000101011
----$rs----     = 00000000000000000000000000011100
----$rt----     = 00000000000000000000000000011101
zeroextend_out  = 00000000000000000111000000101011
signextend_out  = 11111111111111111111000000101011
---Result---    = 00000000000000000000000000000001
Program_counter = 00000000000000000000000000001001
************************************************************

Alu_Control = 000
MIPS_Control : RegDst = 0, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00110000001000100101010101010101
----$rs----     = 00000000000000000000000000000001
----$rt----     = 00000000000000000000000000000010
zeroextend_out  = 00000000000000000101010101010101
signextend_out  = 00000000000000000101010101010101
---Result---    = 00000000000000000000000000000001
Program_counter = 00000000000000000000000000001010
************************************************************

Alu_Control = 001
MIPS_Control : RegDst = 0, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00110100100001010011001100110011
----$rs----     = 00000000000000000000000000000100
----$rt----     = 00000000000000000000000000000101
zeroextend_out  = 00000000000000000011001100110011
signextend_out  = 00000000000000000011001100110011
---Result---    = 00000000000000000011001100110111
Program_counter = 00000000000000000000000000001011
************************************************************
```

```
Alu_Control = 010
MIPS_Control : RegDst = 0, MemtoReg = 0, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 1
---Instruction  = 00100100111010001000010000100001
----$rs----     = 00000000000000000000000000000111
----$rt----     = 00000000000000000000000000001111
zeroextend_out  = 00000000000000000000010000100001
signextend_out  = 11111111111111111000010000100001
---Result---    = 11111111111111111000010000101000
Program_counter = 00000000000000000000000000001100
********************************************************

Alu_Control = 010
MIPS_Control : RegDst = 0, MemtoReg = 1, MemWrite = 0, Branch = 0, Jump = 0, MemRead = 1, RegWrite = 1
---Instruction  = 10001100111010000000000000000001
----$rs----     = 00000000000000000000000000000111
----$rt----     = 11111111111111111000010000101000
zeroextend_out  = 00000000000000000000000000000001
signextend_out  = 00000000000000000000000000000001
---Result---    = 00000000000000000000000000001000
Program_counter = 00000000000000000000000000001101
********************************************************

Alu_Control = 010
MIPS_Control : RegDst = 0, MemtoReg = 0, MemWrite = 1, Branch = 0, Jump = 0, MemRead = 0, RegWrite = 0
---Instruction  = 10101101111100000000000000000010
----$rs----     = 00000000000000000000000000001000
----$rt----     = 00000000000000000000000000010000
zeroextend_out  = 00000000000000000000000000000010
signextend_out  = 00000000000000000000000000000010
---Result---    = 00000000000000000000000000001010
Program_counter = 00000000000000000000000000001110
********************************************************

** Note: $finish    : D:/altera13.1/workspace/hw4/151044058_restored/mips32_testbench.v(38)
```

YUKARDAKI GORSELLERDEKİ INSTRUCTIONLARIN SIRASI

ADD --- ADDU ---- SUB ---- SUBU ---- AND

OR --- NOR --- SLL --- SRL ---SLTU----ANDI

ORI ---- ADDIU ---LW ---- SW

BEQ ve J instructionlarinin sinyallerini üretebiliyorum fakat program counterı ayarlayamadığım için yapamadım.

ALİ HAYDAR KURBAN 151044058