COMPUTER ORGANIZATION

HW4

161044063

ZAFER ALTAY

1-) Components

Bu ödev için hazırladığım ve testlerini yaptığım tüm componentler:

- One Bit 4x1 Mux
- Thirty Two Bit 4x1 Mux
- One Bit Alu
- One Bit Alu MostSignificantBit
- Thirty Two Bit Alu
- Sign Extend
- One Bit 2x1 Mux
- Five Bit 2x1 Mux
- Sixteen Bit 2x1 Mux
- Thirty Two Bit 2x1 Mux
- Zero Extend
- One Bit Full Adder
- ThirtyTwo Bit Full Adder
- Restrict 32 to 18
- One Bit Comparator
- Thirty Two Bit Comparator
- Selector
- JumpBox
- Rtype Jr
- For Lui 16 to 32
- Control Unit
- Alu Control
- Registers
- Data Memory
- Instruction Memory
- Program counter

2-)Test Bench Result

1-) One Bit 4x1 Mux

```
# Select Bits: 00 and Result: 1
# Select Bits: 01 and Result: 1
# Select Bits: 10 and Result: 1
# Select Bits: 11 and Result: 1
# Select Bits: 10 and Result: 0
```

2-) Thirty Two Bit 4x1 Mux

```
# Select Bits: 00 and Result:  Result: 11111111111111000000000000000000
# Select Bits: 01 and Result:  Result: 00000000000000000000000000000001
# Select Bits: 10 and Result:  Result: 00000000000000000000000000000010
# Select Bits: 11 and Result:  Result: 00000000000000000000000000000011
```

3-) One Bit ALU

```
# OPCODE: 000 Input A: 0 InputB: 0 Result: 0, Cout:0
# OPCODE: 000 Input A: 1 InputB: 0 Result: 0, Cout:0
# OPCODE: 000 Input A: 0 InputB: 1 Result: 0, Cout:0
# OPCODE: 000 Input A: 1 InputB: 1 Result: 1, Cout:1
# OPCODE: 001 Input A: 0 InputB: 0 Result: 0, Cout:0
# OPCODE: 001 Input A: 1 InputB: 0 Result: 1, Cout:0
# OPCODE: 001 Input A: 0 InputB: 1 Result: 1, Cout:0
# OPCODE: 001 Input A: 1 InputB: 1 Result: 1, Cout:1
# OPCODE: 010 Input A: 0 InputB: 0 Result: 0, Cout:0
# OPCODE: 010 Input A: 1 InputB: 0 Result: 1, Cout:0
# OPCODE: 010 Input A: 0 InputB: 1 Result: 1, Cout:0
# OPCODE: 010 Input A: 1 InputB: 1 Result: 0, Cout:1
# OPCODE: 110 Input A: 0 InputB: 0 Result: 0, Cout:1
# OPCODE: 110 Input A: 1 InputB: 0 Result: 1, Cout:1
# OPCODE: 110 Input A: 0 InputB: 1 Result: 1, Cout:0
# OPCODE: 110 Input A: 1 InputB: 1 Result: 0, Cout:1
# OPCODE: 111 Input A: 0 InputB: 0 Result: 0, Cout:1
# OPCODE: 111 Input A: 1 InputB: 0 Result: 1, Cout:1
# OPCODE: 111 Input A: 0 InputB: 1 Result: 1, Cout:0
# OPCODE: 111 Input A: 1 InputB: 1 Result: 0, Cout:1
```

4-) ThirtyTwo Bit ALU

```
# OPCODE: 000
# A: 11111111111111110000000000000000
# B: 00000000000000001111111111111111
# Result: 00000000000000000000000000000000, Zero: 1
# OPCODE: 001
# A: 11111111111111110000000000000000
# B: 00000000000000001111111111111111
# Result: 11111111111111111111111111111111, Zero: 0
# OPCODE: 010
# A: 11111111111111110000000000000000
# B: 00000000000000001111111111111111
# Result: 11111111111111111111111111111111, Zero: 0
# OPCODE: 110
# A: 00000000000000001111111111111111
# B: 00000000000000001111111111111111
# Result: 00000000000000000000000000000000, Zero: 1
# OPCODE: 111
# A: 00000000000000001111111111111111
# B: 00001111000011111111000011110000
# Result: 00001111000011110000111100001111, Zero: 0
```

5-) Sign Extender

```
# Input Bits: 0111101110010101 and Result: 00000000000000000111101110010101
# Input Bits: 1111101110010101 and Result: 11111111111111111111101110010101
```

6-) Zero Extender

```
# Input Bits: 0111101110010101 and Result: 00000000000000000111101110010101
# Input Bits: 1111101110010101 and Result: 00000000000000001111101110010101
```

7-) One Bit 2x1 Mux

```
Select Bits: 0 and Result: 0
Select Bits: 1 and Result: 1
Select Bits: 0 and Result: 1
Select Bits: 1 and Result: 0
```

8-) Five Bit 2x1 Mux

```
Select Bits: 0 and Result:  Result: 00001
Select Bits: 1 and Result:  Result: 11110
```

9-) Sixteen Bit 2x1 Mux

```
Select Bits: 0 and Result: 0000000000000000
Select Bits: 1 and Result: 1111111111111111
Select Bits: 0 and Result: 1111111111111111
Select Bits: 1 and Result: 0000000000000000
```

10-) ThirtyTwo Bit 2x1 Mux

```
# Select Bits: 0 and Result:  Result: 00001111000011110000111100001111
# Select Bits: 1 and Result:  Result: 11110000111100001111000011110000
```

11-) Restrict 32 to 18

```
Output Bits: 01111111111111111 and Ä°nputBits: 10100010101111011111111111111111
Output Bits: 100001000010001000 and Ä°nputBits: 00000000000000100001000010001000
```

11-) Selector

```
#  Rtype : 0,Gt : 0,eq 0,lt 0,sel1: 0,sel0: 0
#
#  Rtype : 1,Gt : 1,eq 0,lt 0,sel1: 1,sel0: 1
#
#  Rtype : 1,Gt : 0,eq 1,lt 0,sel1: 0,sel0: 1
#
#  Rtype : 1,Gt : 0,eq 0,lt 1,sel1: 1,sel0: 0
#
```

12-) Control Unit

```
# opcope : 100011 ,RegDest: 1 ,SignExtend: 1,Brn: 0, Bne: 0, lui: 0, MemW : 0, MemRead : 1, MemtoReg :1,AluOp1: 0, AluOp0: 0,AluSrc: 1, RegWr: 1 ,RegWr: 0,Jmp : 0,Jal : 0
# opcope : 101011 ,RegDest: 0 ,SignExtend: 1,Brn: 0, Bne: 0, lui: 0, MemW : 1, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 0,AluSrc: 1, RegWr: 0 ,RegWr: 0,Jmp : 0,Jal : 0
# opcope : 000000 ,RegDest: 0 ,SignExtend: 0,Brn: 0, Bne: 0, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 1, AluOp0: 0,AluSrc: 0, RegWr: 1 ,RegWr: 1,Jmp : 0,Jal : 0
# opcope : 001111 ,RegDest: 1 ,SignExtend: 0,Brn: 0, Bne: 0, lui: 1, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 0,AluSrc: 0, RegWr: 1 ,RegWr: 0,Jmp : 0,Jal : 0
# opcope : 001110 ,RegDest: 1 ,SignExtend: 0,Brn: 0, Bne: 0, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 0,AluSrc: 1, RegWr: 1 ,RegWr: 0,Jmp : 0,Jal : 0
# opcope : 000010 ,RegDest: 0 ,SignExtend: 0,Brn: 0, Bne: 0, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 0,AluSrc: 0, RegWr: 0 ,RegWr: 0,Jmp : 1,Jal : 0
# opcope : 000011 ,RegDest: 0 ,SignExtend: 0,Brn: 0, Bne: 0, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 0,AluSrc: 0, RegWr: 0 ,RegWr: 1,Jmp : 1,Jal : 1
# opcope : 000100 ,RegDest: 0 ,SignExtend: 1,Brn: 1, Bne: 0, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 1,AluSrc: 0, RegWr: 0 ,RegWr: 0,Jmp : 0,Jal : 0
# opcope : 000101 ,RegDest: 0 ,SignExtend: 1,Brn: 0, Bne: 1, lui: 0, MemW : 0, MemRead : 0, MemtoReg :0,AluOp1: 0, AluOp0: 1,AluSrc: 0, RegWr: 0 ,RegWr: 0,Jmp : 0,Jal : 0
```

13-)ALU Control

```
# ALU OP: 00 , Function Field: 100101, AluControl: 010
# ALU OP: 00 , Function Field: 111111, AluControl: 010
# ALU OP: 00 , Function Field: 000000, AluControl: 010
# ALU OP: 11 , Function Field: 100101, AluControl: 001
# ALU OP: 01 , Function Field: 100101, AluControl: 110
# ALU OP: 10 , Function Field: 100101, AluControl: 001
# ALU OP: 10 , Function Field: 100110, AluControl: 111
# ALU OP: 10 , Function Field: 100000, AluControl: 010
# ALU OP: 10 , Function Field: 100010, AluControl: 110
# ALU OP: 10 , Function Field: 100100, AluControl: 000
```

14-)RtypeJr

```
# JR: 0
#
# JR: 1
#
# JR: 0
#
```

15-)One Bit Comparator

```
# a: 0, b: 0 ,gt: 0,eq: 1,lt:0
# a: 1, b: 0 ,gt: 1,eq: 0,lt:0
# a: 0, b: 1 ,gt: 0,eq: 0,lt:1
# a: 1, b: 1 ,gt: 0,eq: 1,lt:0
```

16-)ThirtyTwo Bit Comparator

```
# gt 1, eq: 0 , lt: 0
# A: 01000000000000000000000000000000
# B: 00111111111111111111111111111111
#
# gt 0, eq: 0 , lt: 1
# A: 01000000000000000000000000000000
# B: 01111111111111111111111111111111
#
# gt 0, eq: 0 , lt: 1
# A: 01111111000000000000000110000010
# B: 01111111000000000000000110000100
#
# gt 1, eq: 0 , lt: 0
# A: 01111111000000000000000110000001
# B: 01111111000000000000000110000000
#
```

17-)Jump Box

```
# OUTput : 11111111111111111111111111111111

# OUTput : 00000000000000000000000000000000
```

17-)One Bit Full Adder

```
# Input A: 0 ,InputB: 0 ,Carry In: 0 ,Result: 0, Cout:0
# Input A: 1 ,InputB: 0 ,Carry In: 0 ,Result: 1, Cout:0
# Input A: 0 ,InputB: 1 ,Carry In: 0 ,Result: 1, Cout:0
# Input A: 1 ,InputB: 1 ,Carry In: 0 ,Result: 0, Cout:1
# Input A: 0 ,InputB: 0 ,Carry In: 1 ,Result: 1, Cout:0
# Input A: 1 ,InputB: 0 ,Carry In: 1 ,Result: 0, Cout:1
# Input A: 0 ,InputB: 1 ,Carry In: 1 ,Result: 0, Cout:1
# Input A: 1 ,InputB: 1 ,Carry In: 1 ,Result: 1, Cout:1
```

18-)ThirtyTwo Bit Full Adder

```
# reg1: 00000000000000000000000000000000
# B: 11111111111111111111111111111111
# Result: 11111111111111111111111111111111
# reg1: 11111111111111111111111111111111
# B: 11111111111111111111111111111111
# Result: 11111111111111111111111111111110
# reg1: 01010101010101010101010101010101
# B: 10101010101010101010101010101010
# Result: 11111111111111111111111111111111
# reg1: 01111111111111111111111111111111
# B: 00000000000000000000000000000001
# Result: 10000000000000000000000000000000
```

18-)Program Counter

```
# clk =1, inpc=00000000000000000000000000000001, pc=00000000000000000000000000000001
# clk =0, inpc=00000000000000000000000000000001, pc=00000000000000000000000000000001
# clk =1, inpc=00000000000000000000000011000001, pc=00000000000000000000000011000001
# clk =0, inpc=00000000000000000000000011000001, pc=00000000000000000000000011000001
# clk =1, inpc=00000000000000000000001111000001, pc=00000000000000000000001111000001
# clk =0, inpc=00000000000000000000001111000001, pc=00000000000000000000001111000001
# clk =1, inpc=00000000000000000000000011111111, pc=00000000000000000000000011111111
# clk =0, inpc=00000000000000000000000011111111, pc=00000000000000000000000011111111
# clk =1, inpc=00000000000000000000000000000000, pc=00000000000000000000000000000000
```

19-)For Lui

```
Input Bits: 1111111111111111 and Result: 11111111111111110000000000000000
Input Bits: 0000000000000000 and Result: 00000000000000000000000000000000
Input Bits: 1111000011110000 and Result: 11110000111100000000000000000000
```

20-)Instruction Memory

Before Inst Memory

```
 1  00000000000000001000100000100000
 2  00000000000000000000000000011110
 3  00000000000000000000000111111110
 4  11111111111111111111111111111111
 5  00000000000000000000000000000000
 6  00000000000000000000000000000000
 7  00000000000000000000000000000000
 8  00000000000000000000000000000000
 9  00000000000000000000000000000000
10  00000000000000000000000000000000
11  00000000000000000000000000000000
12  00000000000000000000000000000000
13  00000000000000000000000000000000
14  00000000000000000000000000000000
```

Test:

```
ReadAddress=00000000000000000000000000000001 instruction=00000000000000000000000000011110
ReadAddress=00000000000000000000000000000010 instruction=00000000000000000000000111111110
ReadAddress=00000000000000000000000000000011 instruction=11111111111111111111111111111111
```

After:

```
 1  00000000000000001000100000100000
 2  00000000000000000000000000011110
 3  00000000000000000000000111111110
 4  11111111111111111111111111111111
 5  00000000000000000000000000000000
 6  00000000000000000000000000000000
 7  00000000000000000000000000000000
 8  00000000000000000000000000000000
 9  00000000000000000000000000000000
10  00000000000000000000000000000000
11  00000000000000000000000000000000
12  00000000000000000000000000000000
13  00000000000000000000000000000000
14  00000000000000000000000000000000
15  00000000000000000000000000000000
```

21-)Data Memory

Before:

```
 1  00000000000000000000000000000000
 2  00000000000000000000000000000000
 3  00000000000000000000000000000000
 4  00000000000000000000000000000000
 5  00000000000000000000000000000000
 6  00000000000000000000000000000000
 7  00000000000000000000000000000000
 8  00000000000000000000000000000000
 9  00000000000000000000000000000000
10  00000000000000000000000000000000
11  00000000000000000000000000000000
12  00000000000000000000000000000000
13  00000000000000000000000000000000
14  00000000000000000000000000000000
15  00000000000000000000000000000000
16  00000000000000000000000000000000
17  00000000000000000000000000000000
18  00000000000000000000000000000000
19  00000000000000000000000000000000
20  00000000000000000000000000000000
21  00000000000000000000000000000000
22  00000000000000000000000000000000
23  00000000000000000000000000000000
24  00000000000000000000000000000000
25  00000000000000000000000000000000
26  00000000000000000000000000000000
27  00000000000000000000000000000000
28  00000000000000000000000000000000
29  00000000000000000000000000000000
```

Test:

```
# MemWrite: 1,MemRead: 0 ,address : 0000000000000010,ReadData : xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#
# MemWrite: 0,MemRead: 1 ,address : 0000000000000010,ReadData : 11111110111111011111101111110111
#
# MemWrite: 1,MemRead: 0 ,address : 0000000000000100,ReadData : 11111110111111011111101111110111
#
# MemWrite: 0,MemRead: 1 ,address : 0000000000000100,ReadData : 00000000000000000011111111111111
#
```

After:

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/DataMemory_testbench/test/memory
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000000
11111110111111011111101111110111
00000000000000000000000000000000
00000000000000000011111111111111
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

22-)Register File

Before:

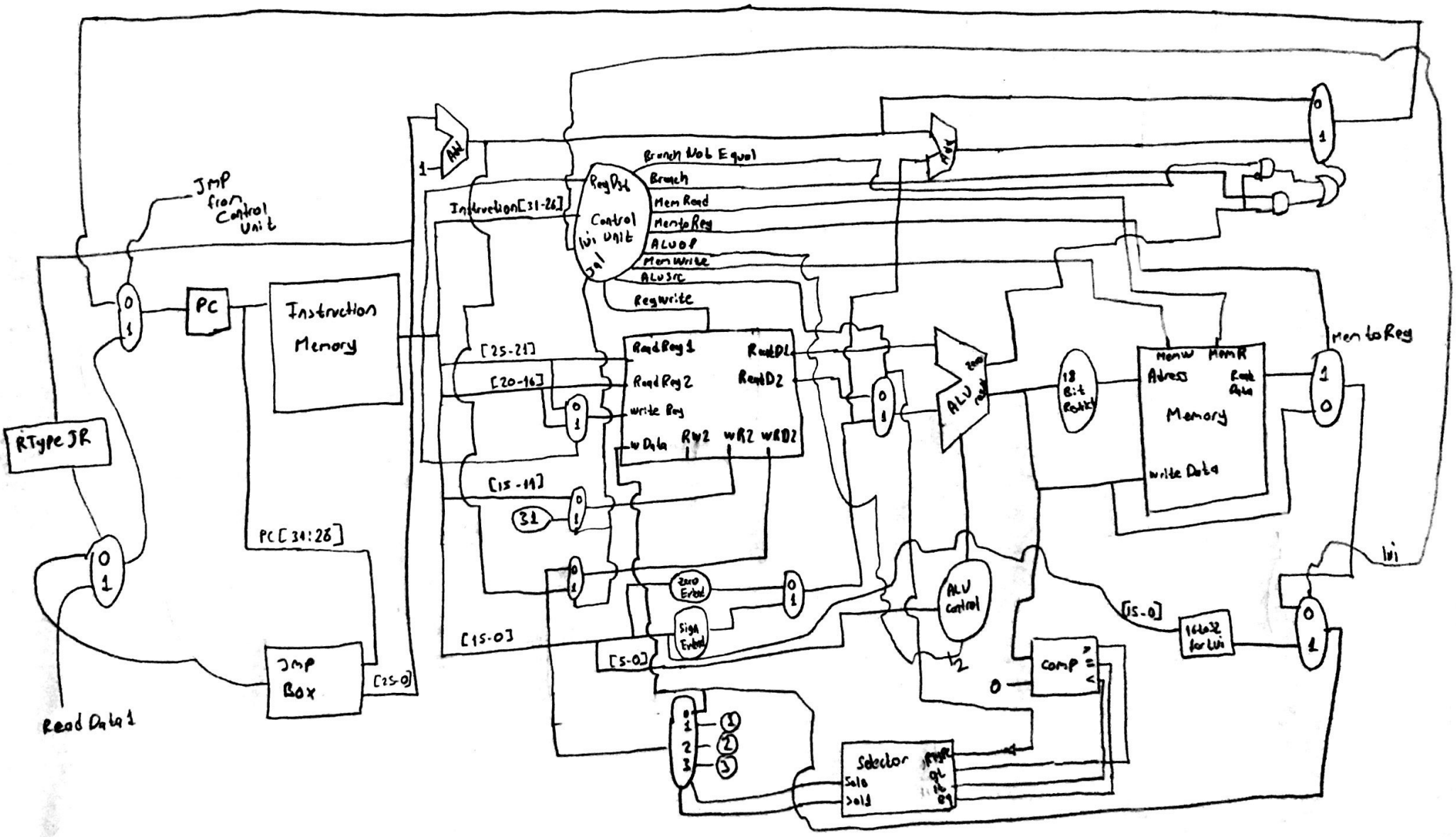
```
00000000000000000000000000000011
00000000000000000000000000001111
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

Test:

```
# Regwrite :0, Rgewrite2: 1, ReadReg1=00000 , ReadReg2=00001 , ReadData1=00000000000000000000000000000011 , ReadData2=00000000000000000000000000001111 ,WriteData=11111110111111011111101111110111 , WriteReg1 =xxxxx,
WriteReg2= 00111
#
# Regwrite :1, Rgewrite2: 0, ReadReg1=00000 , ReadReg2=00001 , ReadData1=00000000000000000000000000000011 , ReadData2=00000000000000000000000000001111 ,WriteData=11111110111111011111101111110111 , WriteReg1 =00010,
WriteReg2= 00111
```

After:

```
1  // memory data file (do not edit the following line - required for mem load use)
2  // instance=/Registers_testbench/test/registers
3  // format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
4  00000000000000000000000000000011
5  00000000000000000000000000001111
6  00000000000000000000000000000000
7  00000000000000000000000000000000
8  00000000000000000000000000000000
9  00000000000000000000000000000000
10 00000000000000000000000000000000
11 11111110111111101111110111110111
12 00000000000000000000000000000000
13 00000000000000000000000000000000
14 00000000000000000000000000000000
15 00000000000000000000000000000000
16 00000000000000000000000000000000
17 00000000000000000000000000000000
18 00000000000000000000000000000000
```



* Datapath'ime ek olarak eklediğim componentler:

1) RTypeJR = Eger Instruction Memory'den gelen Singal jR ise Program Counter'a ReadData1 yani rs content'ini secmek için 1 sinyali üretir. Diger durumlarda 0'olur.

2) JmpBox = Jump Edilmesi istenen instruction için PC'nin ilk 6 bibi ile Jump instructionunun son 26 bitini birleştirir.

3) Comparator = ALU'dan cikan sonucu 0 ile karsilastirir. 32 bit'tir. 32 tane 1 bitlik comparator ile ürettim.

4) Selector = Comparatordan gelen sinyal ve ALUsrc' sinyalinin degilini input olarak 2 tane Select sinyali üretir. Eger secim instruction rtype degilse 0, rtype'ise 0 ile karsilastirilmasina göre sinyal üretir.

5) 4x1 Mux = Selectordan aldigi sinyalleri kullanarak Register'a yazilacak datayi secer. RType degilse normal data, r type ise 1, 2 veya 3'ü output verir.

6) 18 Bit Restrict = Memory 18 bit kabul ettigi için 32 bit'i 18 bit'e düşürür.

7) 16 to 32 farlui = Lui instructionu için son 16 biti 0, ilk 16 biti ise aldigi input yapan 32 bit output üretir.

8) ALU = Toplama, cikarma, and, or, xor islemlerini yapar.

9) Register : Register normalinden farklı olarak 2 tane yazmak için data alır.

Çünkü Rtype instructionlarda rs ve rd'ye yazma gerçekleştiriliyor. Böylece böyle

2 tane register adresi, 2 tane data, 2 tane reg write sinyali alır.

10) 32 bit Full Adder : 32 tane 1 bit Full Adder'den oluşur.

11) ALU Control = ALUOp ve Function Field'den aldığı sinyallere göre ALU için işlem bitleri üretir.