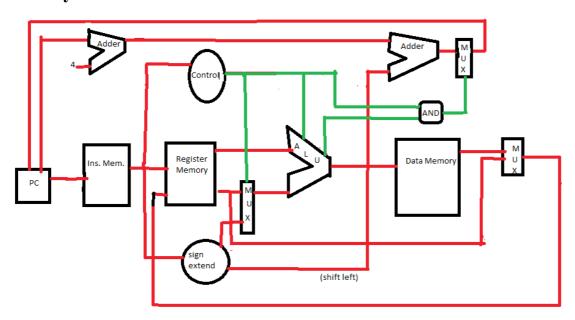
## 1. GİRİŞ

## 1.1 Büyük Resim



#### 1) mips instruction modülü

PC'deki değere göre instruction.mem dosyasında istenen instructionu fetch eder.

#### 2) mux 5bit modülü

write reg register'ının hangi register olacağına karar veriyor.

#### 3) mips registers modülü

Instructionların parse edilmesiyle gelen register'daki değerleri read\_data\_1 ve read\_data\_2'ye atıyor. Ayrıca write-back stage'inde RegDest sinyaline göre write\_reg'e ilgili veriyi yazıyor.

#### 4) mux 32bit modülü

Alu\_src sinyaline göre read\_data\_2'nin mips\_registers'dan gelen sayıya mı yoksa sign\_extend edilmiş sayıya mı eşit olacağını belirler.

#### 5) control modülü

Instruction'ların parse edilmesiyle çıkan opcode'a göre yapılacak işlemi anlıyor ve datapath'e gerekn sinyalleri gönderiyor.

#### 6) ALU modülü

Opcode'a göre, gelen datalar üzerinde gereken işlemi yapıyor. Result'ı buluyor.

#### 7) mips\_data modülü

mem\_read ya da mem\_write sinyali geldiğinde ALU'dan gelen result değerindeki adres üzerinde işlem yapar. (okuma ya da yazma işlemi)

#### 8) mux 32bit modülü

memToReg sinyaline göre memory'den gelen mi yoksa ALU'dan gelen değerin mi register.mem dosysına yazılacağını belirler.

#### 9) extend module

Gelen sayıyı extend eder.

#### 1.2 Life cycle of an instruction

- -mips instr mem modülü
- -control modülü
- -mips registers modülü

```
-extend modülü
-mux_32bit modülü
-alu modülü
-mips_data_mem modülü
-mux_32bit modülü
-mux_5bit modülü
-mips_registers modülü
-mux_PC modülü
```

#### 1.3

Jump ederken PC'yi değiştirmek için kullandığımız PC = {{PC+4[31:28]},jump\_address,2'b0}; sürekli hata verdi bu sebeple jump'la ilgili olan her şeyi silmek zorunda kaldım.

ModelSim'de bazı outputlarda x çıkıyor. Bunu da düzeltemedim.

#### 2. Metod

Instruction fetch edildikten sonra opcode'u control'e gider ve opcode'a göre datapath'te yapılacak işlem belirlenir. Instruction'ın kalan kısımları da fetch edildikten sonra hangi register'daki değerin kullanıcağını anlamak için ve sign extend edilemek için kullanılır. İstenen register'daki datalar (ya da sign extend edilmiş data) gerekli işlemi yapmak üzere ALU modülüne gönderilir. Orada gerekli işlem yapıldıktan sonra eğer branch edilmesi gerekiyorsa branch yapılır, memory'yle ilgili bir işlem varsa Result oraya gönderilir, bunların ikisi de yoksa write back yapmak için register modülüne gönderilir.

En sonunda bulunan sonuç ilgili registera yazılır ya da istenen memory adresine yazılır ya da jump işlemi yapılır. PC'nin yeni değerinin gösterdiği yerdeki instruction alınarak aynı döngü program bitene kadar tekrarlanır.

#### 2.1

```
1) mips_instr_mem (instruction, PC)
input = PC
output = instruction
PC'nin değerine göre istenen instruction'ı verir.
```

2) control (opcode, sig\_reg\_dest,signal\_reg\_write, sig\_branch, sig\_mem\_read, sig\_mem\_write, sig\_alu, sig\_memToReg, sig\_alu\_src, alu\_op)
input = opcode
output = sig\_reg\_dest,signal\_reg\_write, sig\_branch, sig\_mem\_read, sig\_mem\_write, sig\_alu, sig\_memToReg, sig\_alu\_src, alu\_op

Gelen opcode'a göre datapath'in sitenen islemi yapması için gereken sinyalleri üretir.

### 3) mips registers

```
(read_data_1,temp,write_data,read_reg_1,read_reg_2,write_reg,signal_reg_write,clock, zero)
output = read_data_1, read_data_2
input = write_data,read_reg_1,read_reg_2,write_reg,signal_reg_write,clock, zero
Gelen sinyallere göre gelen register'daki değeri verir ya da gelen veriyi istenen register'a
yazar.
```

```
4) mux_32bit (read_data_2, sign_ex, temp, sig_alu_src)
output = out,
input = in1, in2, signal
Gelen sinyale göre in1 ya da in2'yi output olarak verir.
```

5) alu (read\_data\_1, read\_data\_2, opcode, shamt, zero, result)

output = result, zero, sig disable

**input** = read data 1, read data 2, opcode, shamt, resul, zero, sig disable;

Gelen opcode'a göre gelen datalar üzerinde işlem yapar ve result olarak output verir.

Branch durumunda zero sinyalini 1 yapar.

### 6) mips\_data\_mem (read\_data, result, write\_data, sig\_mem\_read, sig\_mem\_write, zero)

output = read data

input = result, write\_data, sig\_mem\_read, sig\_mem\_write, zero

sig\_mem\_read ve sig\_mem\_write sinyallerine göre okuma ya da yazma yapar okunan değer olarak read data'yı verir, yazılacak değer için write data'yı ve adresi alır.

#### 7) mux 5bit (write reg, read reg 2, rd, sig reg dest)

**output** = write reg

**input** = read reg 2, rd, sig reg dest)

Gelen sinyale göre read\_reg\_2 ya da rd'yi output olarak verir.

## 8) mux\_PC (temp2, temp, sign\_ex, zero, sig\_branch, clock);

output = PC,

**input** = temp, sign ex, zero, sig branch, clock, sig disable

Normal durumda PC'yi 1 artırır

Zero 1'ken yani branch durumunda sign extend'i 2 kere shift edip değeri PC'ye atar.

## 9) extend(out, in)

input = in

output = out

Gelen sayıyı 16 bitten 32 bite extend eder.

# 3. Sonuç

#### 3.1

