Gebze Technical University
Computer Engineering

CSE 331

ASSİGNMENT 3 REPORT

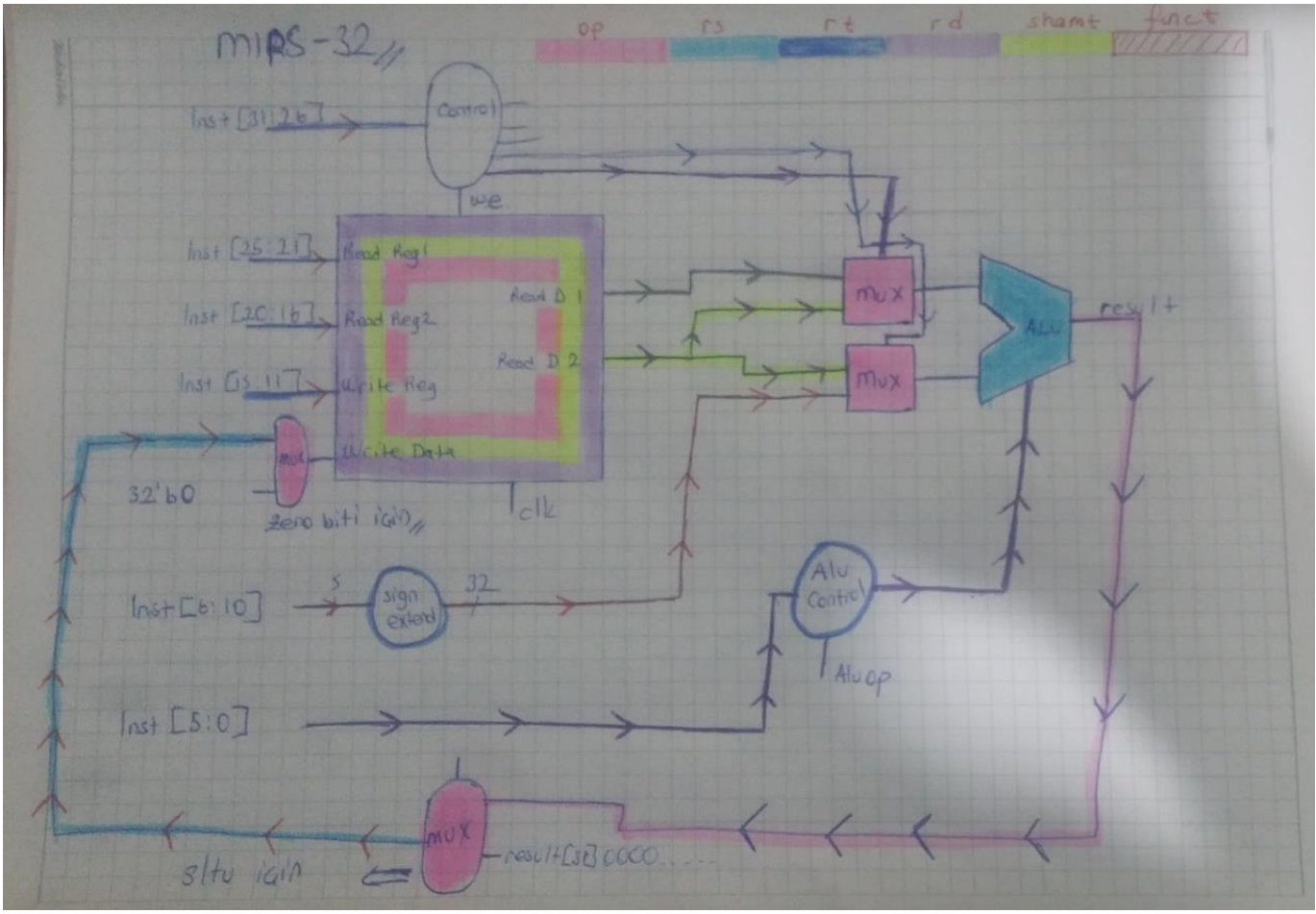
AKIN ÇAM
151044007

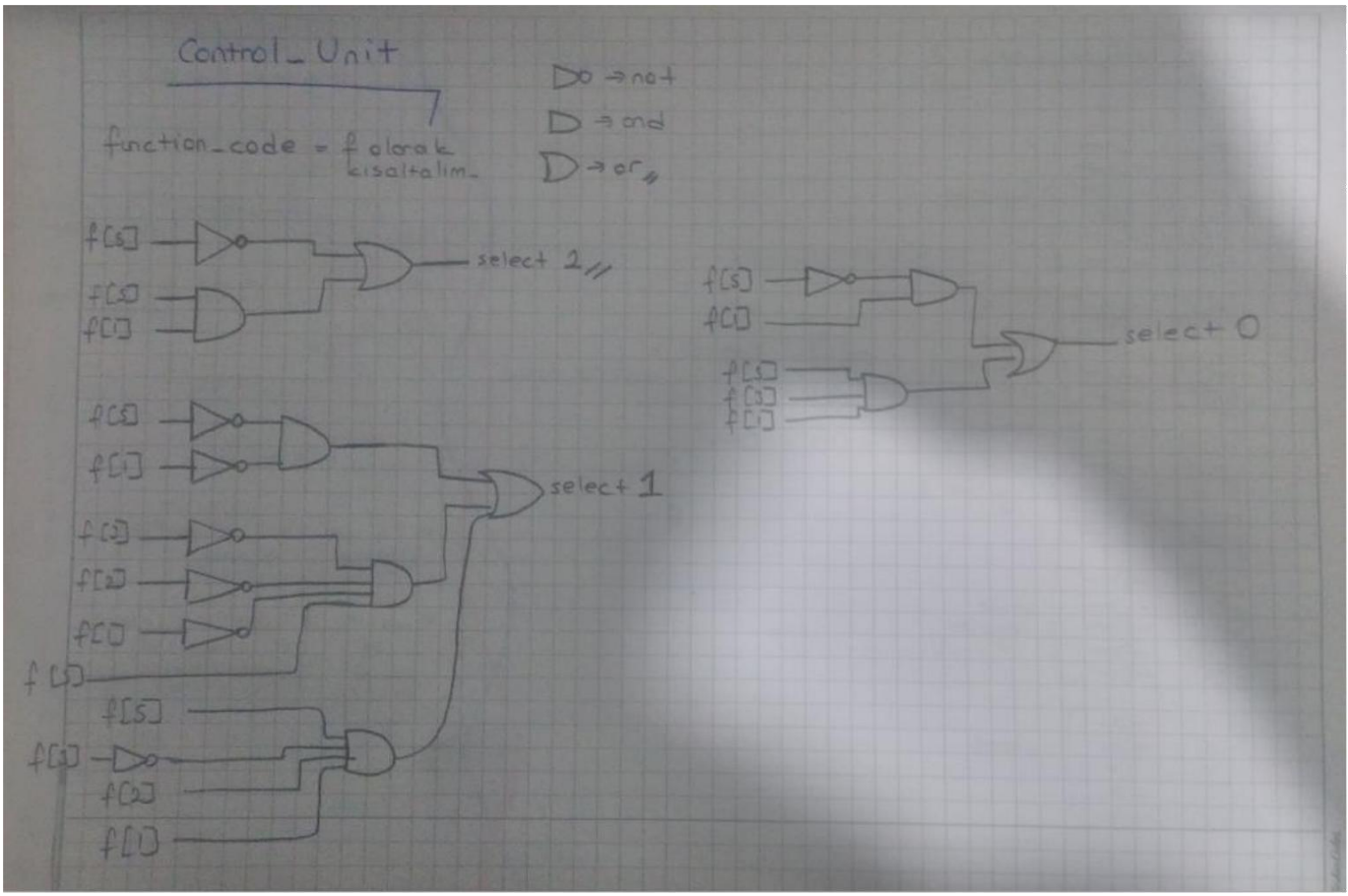
Course Assistant:

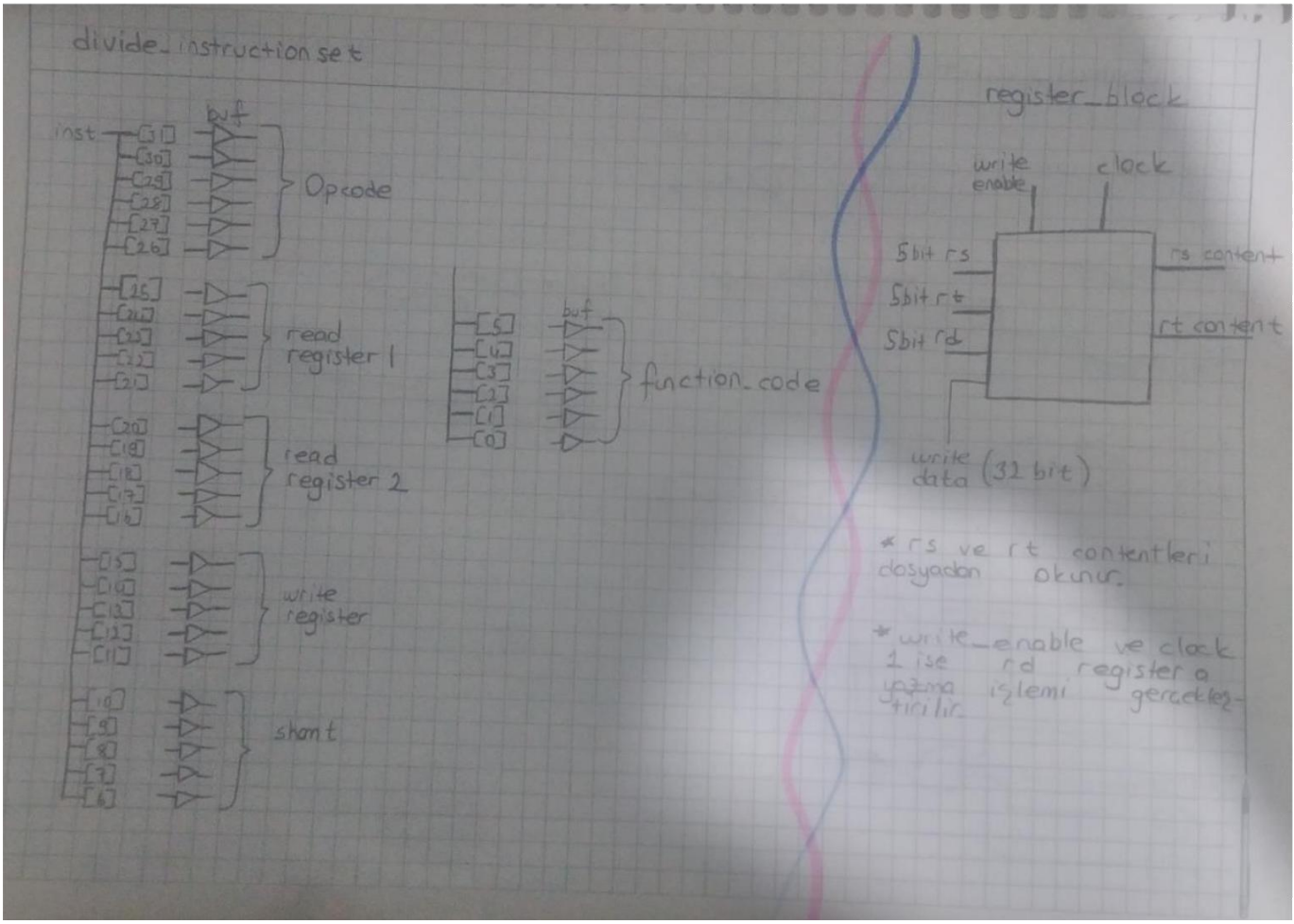
1) Schematic Designs for All Modules

R-type instructionlar için çalışan bir single cycle processor tasarlanmıştır.

Zero registerına hiçbir şeyin yazılmaması,shift durumunda rs yerine rt; rt yerine 32 bite genişletilmiş shampt kullanılmıştır. Sltu için sub kullanılıp daha sonra küçük ise rd registera (most significant sub result ve 00.. ile) 000….01 değil ise 00…..00 yazılır.




Control_Unit
function_code = f olarak kisaltalim
D → not
D → and
D → or
f[5]
f[3]
f[0]
select 2
select 0
select 1
f[4]
f[2]
f[1]


divide_instruction set
inst
buf
Opcode
read register 1
read register 2
write register
shant
function_code
register_block
write enable
clock
5bit rs
5bit rt
5bit rd
rs content
rt content
write data (32 bit)
* rs ve rt contentleri dosyadan okunur.
* write_enable ve clock 1 ise rd register a yazma islemi gerceklestirilir

MIPS_32:
Verilen instruction ilk önce opcode,rs,rt,rd,shamt,function_code olarak ayrılır.(divide_instructionset) rs ve rt contentleri okunur.(register_block) Select inputları belirlenir(control_unit). Verilen function_code 000000 000010 ise yani shift ise alu ya rs yerine rt, rt yerine shamt ın 32 bitlik hali gönderilir(mux ile)(function_code 000000 ve 0000010 ile ayrı ayrı olarak her index için ilk önce xor yapılır daha sonra xor sonucu not işlemine konulur. Bu işlem function code un tüm indexleri için uygulanır(6 defa) Daha sonra bu indexler için and işlemi yapılır. Daha sonra bu iki kod için çıkan sonuçlar or işlemine konulur ve bu mux a gönderilir. Mux 1 ise 32bit haline getirilmiş shiftamount kullanılır aksi takdirde normal işleme devam edilir). Verilen function_code sltu ise alu da çıkarma işlemi yapılır ve sonucun most significant biti ve diğer kısımlar sıfır olacak şekilde rd belirlenir(mux ile seçim yapılır)(shampt işleminde kullanılan xor not ve and işlemi kullanılır).
Eğer verilen rd zero register ise rd ye yazma işlemi gerçekleştirilmez(rd content ilk önce 0 daha sonra bir önceki işlem sonucu ile or işlemi yapılır. Bu select mux'a gönderilir .
İlk başta write_enable ve clock 0 dır. Daha sonra clock 1 ve write_enable 1 durumunda yazmak için register_block çağrılır.
REGISTER_BLOCK:
Bir registers(her indexte 32 bitlik sayı alabilen array size ı 32 olan bir değişken) değişkenine dosyadan contentler okunur. Read_data1 ve read_data2 ye verilen read_reg1 ve read_reg2 indexleri kullanılarak contentler okunur.
Eğer write_enable ve clk 1 ise rd register ına yazma işlemi gerçekleştirilir.
CONTROL_UNIT:
Select inputları belirlenir. function_code a göre alunun select bitleri belirlenir. Select bitleri için 1 olduğu durumlar belirlenmiş ve bu durumlar için ilgili gate ler ile bir yapı oluşturulmuştur.
DIVIDE_INSTRUCTIONSET:
verilen instruction ilk önce opcode,rs,rt,rd,shamt,function_code olarak ayrılır. opcode 6 bit, rs,rt,rd,shamt 5bit,,function_code 6 bit.

2) Modelsim Simulation Results:

3.1 Register Contentleri:

```
 4  00000000000000000000000000000000
 5  00000000000000000000000011111110
 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
30  00000000000000000000000000000000
31  00000000000000000000000000000000
32  00000000000000000000000000000000
33  00000000000000000000000000000000
34  00000000000000000000000000000000
35  00000000000000000000000000000000
36
```

3.2 Sonuçlar:

```
****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010000000000100000 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 00000 | shamt= 00000 | function_code= 100000
 alu result= 00000000000000000000000011111100
 Rd content after write--> 00000000000000000000000000000000


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010001100000100001 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 00011 | shamt= 00000 | function_code= 100001
 alu result= 00000000000000000000000011111100
 Rd content after write--> 00000000000000000000000011111100


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010010000000100100 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 00100 | shamt= 00000 | function_code= 100100
 alu result= 00000000000000000000000011111110
 Rd content after write--> 00000000000000000000000011111110


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010010100000100111 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 00101 | shamt= 00000 | function_code= 100111
 alu result= 11111111111111111111111100000001
 Rd content after write--> 11111111111111111111111100000001
```

```
****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010011000000100101 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 00110 | shamt= 00000 | function_code= 100101
 alu result= 00000000000000000000000011111110
 Rd content after write--> 00000000000000000000000011111110


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000000000001001110000101011 | opcode= 000000 | rs= 00000 | rt= 00001 | rd= 00111 | shamt= 00000 | function_code= 101011
 alu result= 00000000000000000000000000000001
 Rd content after write--> 00000000000000000000000000000001


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010100000011000000 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 01000 | shamt= 00011 | function_code= 000000
 alu result= 00000000000000000000011111110000
 Rd content after write--> 00000000000000000000011111110000


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010100100011000010 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 01001 | shamt= 00011 | function_code= 000010
 alu result= 00000000000000000000000000011111
 Rd content after write--> 00000000000000000000000000011111


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000000000001010100000100010 | opcode= 000000 | rs= 00000 | rt= 00001 | rd= 01010 | shamt= 00000 | function_code= 100010
 alu result= 11111111111111111111111100000010
 Rd content after write--> 11111111111111111111111100000010


****************
 Rd content before write--> 00000000000000000000000000000000

 instruction_set= 00000000001000010101100000100011 | opcode= 000000 | rs= 00001 | rt= 00001 | rd= 01011 | shamt= 00000 | function_code= 100011
 alu result= 00000000000000000000000000000000
 Rd content after write--> 00000000000000000000000000000000
```

3.3 Instruction lardan sonra contentlerin bulundugu dosya:

```
 4  00000000000000000000000000000000
 5  00000000000000000000000011111110
 6  00000000000000000000000000000000
 7  00000000000000000000000111111100
 8  00000000000000000000000011111110
 9  11111111111111111111111100000001
10  00000000000000000000000011111110
11  00000000000000000000000000000001
12  00000000000000000000011111110000
13  00000000000000000000000000011111
14  11111111111111111111111100000010
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
30  00000000000000000000000000000000
31  00000000000000000000000000000000
32  00000000000000000000000000000000
33  00000000000000000000000000000000
34  00000000000000000000000000000000
35  00000000000000000000000000000000
36
```