CSE 331 PROJECT #4

Gökhan Has – 161044067

NOT : Dosyalardan okuma yapan modüllerde dosya yolu (path) sorun yaratabilmektedir. Eğer hata alınırsa dosya yolunun kontrol edilerek düzeltilmesi gerekmektedir.

1 BITLIK ALU :

```
# time = 0,     Array =000,     Ai =0,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =20,     Array =000,     Ai =1,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =1,        result =1
# time =40,     Array =000,     Ai =1,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =60,     Array =000,     Ai =0,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =80,     Array =001,     Ai =0,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =100,    Array =001,     Ai =1,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =1,        result =1
# time =120,    Array =001,     Ai =1,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =140,    Array =001,     Ai =0,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =160,    Array =010,     Ai =0,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =180,    Array =010,     Ai =0,  Bi =0,  Ci =1,  Lessi =0,       Ci+1 =0,        result =1
# time =200,    Array =010,     Ai =0,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =220,    Array =010,     Ai =0,  Bi =1,  Ci =1,  Lessi =0,       Ci+1 =1,        result =0
# time =240,    Array =010,     Ai =1,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =260,    Array =010,     Ai =1,  Bi =0,  Ci =1,  Lessi =0,       Ci+1 =1,        result =0
# time =280,    Array =010,     Ai =1,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =1,        result =0
# time =300,    Array =010,     Ai =1,  Bi =1,  Ci =1,  Lessi =0,       Ci+1 =1,        result =1
# time =320,    Array =110,     Ai =0,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =340,    Array =110,     Ai =0,  Bi =0,  Ci =1,  Lessi =0,       Ci+1 =1,        result =0
# time =360,    Array =110,     Ai =0,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =0,        result =0
# time =380,    Array =110,     Ai =0,  Bi =1,  Ci =1,  Lessi =0,       Ci+1 =0,        result =1
# time =400,    Array =110,     Ai =1,  Bi =0,  Ci =0,  Lessi =0,       Ci+1 =1,        result =0
# time =420,    Array =110,     Ai =1,  Bi =0,  Ci =1,  Lessi =0,       Ci+1 =1,        result =1
# time =440,    Array =110,     Ai =1,  Bi =1,  Ci =0,  Lessi =0,       Ci+1 =0,        result =1
# time =460,    Array =110,     Ai =1,  Bi =1,  Ci =1,  Lessi =0,       Ci+1 =1,        result =0
```

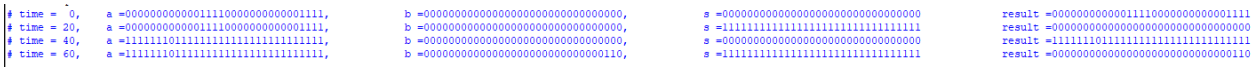
32 BIT ALU:

```
# time = 0,     ALUopsArray =000,       AiArrays =00110000110000000000000000000001,     BiArrays =00110000110000000000000000000111,     ResultArray =00110000110000000000000000000001
# time =20,     ALUopsArray =000,       AiArrays =01111111111111111111111111111110,     BiArrays =01111111111111111111111111111111,     ResultArray =01111111111111111111111111111110
# time =40,     ALUopsArray =000,       AiArrays =01110000110011110000000011110001,     BiArrays =00110000110000000000000000000111,     ResultArray =00110000110000000000000000000001
# time =60,     ALUopsArray =000,       AiArrays =01111111000001111000011111110001,     BiArrays =01111111111111111111111111111111,     ResultArray =01111111000001111000011111110001
# time =80,     ALUopsArray =001,       AiArrays =00110000110000000000000000000001,     BiArrays =01111111111111111111111111111111,     ResultArray =01111111111111111111111111111111
# time =100,    ALUopsArray =001,       AiArrays =01111111111111111111111111110001,     BiArrays =01111111111111111111111111111111,     ResultArray =01111111111111111111111111111111
# time =120,    ALUopsArray =001,       AiArrays =00110000110011110000000011110001,     BiArrays =01111111111111111111111111000111,     ResultArray =01111111111111111111111111111111
# time =140,    ALUopsArray =001,       AiArrays =01110000111111110000111111110001,     BiArrays =01111111111111111111111111000000,     ResultArray =01111111111111111111111111110001
# time =160,    ALUopsArray =010,       AiArrays =00000000000000000000000000000000,     BiArrays =00000000000000000000000000000001,     ResultArray =00000000000000000000000000000001
# time =180,    ALUopsArray =010,       AiArrays =00000000000000000000000000000011,     BiArrays =00000000000000000000000000000111,     ResultArray =00000000000000000000000000001010
# time =200,    ALUopsArray =110,       AiArrays =00000000000000000000000000000011,     BiArrays =00000000000000000000000000000010,     ResultArray =00000000000000000000000000000001
# time =220,    ALUopsArray =110,       AiArrays =01111111111111111111111111110001,     BiArrays =00001111111111111111111111111111,     ResultArray =01101111111111111111111111110010
# time =240,    ALUopsArray =111,       AiArrays =00000000000000000000000000000000,     BiArrays =00000000000000000000000000000001,     ResultArray =00000000000000000000000000000001
# time =260,    ALUopsArray =111,       AiArrays =00000000000000000000000000000001,     BiArrays =00000000000000000000000000000000,     ResultArray =00000000000000000000000000000000
```

MUX 5 BITLIK:

```
# Input1 = 00000,       Input2 = 11111, Selection = 0,  Result = 00000
# Input1 = 00000,       Input2 = 11111, Selection = 1,  Result = 11111
```

MUX 32 BITLIK: Bu 32 bit girişleri olan muxlarda selection biti 32 olacak şekilde tasarladım.



Bit1to32 : Tek biti 32 bite çevirir. 32 bitlik muxların sinyal biti 32 olduğu için şarttır.

```
# time = 0,      Input Bit = 0,          Result = 00000000000000000000000000000000
# time =20,      Input Bit = 1,          Result = 11111111111111111111111111111111
```

Mux 1 BITLIK:

```
# time =  0,    a =0,   b =0,   c =0,   d =0    s0 =0   s1 =0   result =0
# time = 20,    a =1,   b =0,   c =0,   d =0    s0 =0   s1 =0   result =1
# time = 40,    a =0,   b =0,   c =0,   d =0    s0 =0   s1 =1   result =0
# time = 60,    a =0,   b =1,   c =0,   d =0    s0 =0   s1 =1   result =1
# time = 80,    a =0,   b =0,   c =0,   d =0    s0 =1   s1 =0   result =0
# time = 100,   a =0,   b =0,   c =1,   d =0    s0 =1   s1 =0   result =1
# time = 120,   a =0,   b =0,   c =0,   d =0    s0 =1   s1 =1   result =0
# time = 140,   a =0,   b =0,   c =0,   d =1    s0 =1   s1 =1   result =1
```

XOR:

```
# time =  0, a =0, b=0, result=0
# time = 20, a =1, b=0, result=1
# time = 40, a =0, b=1, result=1
# time = 60, a =1, b=1, result=0
```

ZeroExtend:

```
time = 0,     Immediate = 1111111111111111,          Result = 00000000000000001111111111111111
time =20,     Immediate = 1111111100000001,          Result = 00000000000000001111111100000001
time =40,     Immediate = 0000000000001111,          Result = 00000000000000000000000000001111
time =60,     Immediate = 0000000000000000,          Result = 00000000000000000000000000000000
```

SignExtend:

```
# time = 0,     Immediate = 1111111111111111,          Result = 11111111111111111111111111111111
# time =20,     Immediate = 1111111100000001,          Result = 11111111111111111111111100000001
# time =40,     Immediate = 0000000000001111,          Result = 00000000000000000000000000001111
# time =60,     Immediate = 0000000000000000,          Result = 00000000000000000000000000000000
```

ShiftLeft32 : 16 to 32

```
# time = 0,     InputArr = 11111111111111111111111111111111,          Result = 11111111111111111111111111111100
# time =20,     InputArr = 11111111000000011111111100000001,          Result = 11111100000001111111110000000100
# time =40,     InputArr = 00000000000011110000000000001111,          Result = 00000000001111000000000000111100
# time =60,     InputArr = 00000000000000000000000000000000,          Result = 00000000000000000000000000000000
```

ShiftLeft28: 26 to 28. (Jump instructionları için)

```
# time = 0,     InputArr = 11111111111111111111111111,        Result = 11111111111111111111111100
# time =20,     InputArr = 11111111000000011111111100,        Result = 11111111000000011111111110000
# time =40,     InputArr = 00000000000011110000000001,        Result = 00000000000011110000000000100
# time =60,     InputArr = 00000000000000000000000000,        Result = 00000000000000000000000000000
```

ALU CONTROL:

```
# ALUop = 00,   Function Field = xxxxxx,        Operation = 010
# ALUop = 01,   Function Field = xxxxxx,        Operation = 110
# ALUop = 10,   Function Field = 100000,        Operation = 010
# ALUop = 10,   Function Field = 100010,        Operation = 110
# ALUop = 10,   Function Field = 100100,        Operation = 000
# ALUop = 10,   Function Field = 100101,        Operation = 001
# ALUop = 10,   Function Field = 101010,        Operation = 111
```

CONTROL UNIT:

```
# Opcode = 000000, regDst = 1, Jump = 0, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =1,  ALUop =10
# Opcode = 100011, regDst = 0, Jump = 0, Branch = 0, MemRead = 1, MemtoReg = 1, MemWrite = 0, ALUsrc = 1, RegWrite =1,  ALUop =00
# Opcode = 101011, regDst = 0, Jump = 0, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 1, ALUsrc = 1, RegWrite =0,  ALUop =00
# Opcode = 000000, regDst = 1, Jump = 0, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =1,  ALUop =10
# Opcode = 000100, regDst = 0, Jump = 0, Branch = 1, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =0,  ALUop =01
# Opcode = 000010, regDst = 0, Jump = 1, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =0,  ALUop =00
# Opcode = 000011, regDst = 0, Jump = 1, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =0,  ALUop =00
# Opcode = 000101, regDst = 0, Jump = 0, Branch = 0, MemRead = 0, MemtoReg = 0, MemWrite = 0, ALUsrc = 0, RegWrite =0,  ALUop =00
```

Instruction Memory:

```
#  time =  0, Insruction: 10000000011000100000000000001010
#  time = 20, Insruction: 10000000010000111110101000110000
#  time = 40, Insruction: 10010000101001100011000100110001
#  time = 60, Insruction: 10010000110001011111000110110001
```

MIPS32:

Ana modül yani tüm datapath buradadır. Tüm hesaplamaları bu modül yapar.

R TYPE

add	RD = RS + RT
xor	RD = RS xor RT
slt	RD = (RS < RT) ? 1:0
sub	RD = RS – RT
and	RD = RS and RT
or	RD = RS or RT

sra	RD = RT shift h (h = number)
srl	RD = RT >> Shamt

I TYPE

xori	RT = RS + ZEROEXTEND
sltiu	RT = RS + SIGNEXTEND ? 1:0
lw	RT = MEM[RS + SIGNEXTEND]
lh	RT = {16'b0, MEM[RS + ZEROEXTEND](15:0)}
lb	RT = {24'b0, MEM[RS + ZEROEXTEND](7:0)}
sw	MEM[RS + SIGNEXTEND] = RT
sb	MEM[RS + SIGNEXTEND(7:0)] = RT(7:0)
beq	(RS == RT) ? 1: PC = PC + 4 + BRANCHADDER
bne	(RS == RT) ? 0: PC = PC + 4 + BRANCHADDER

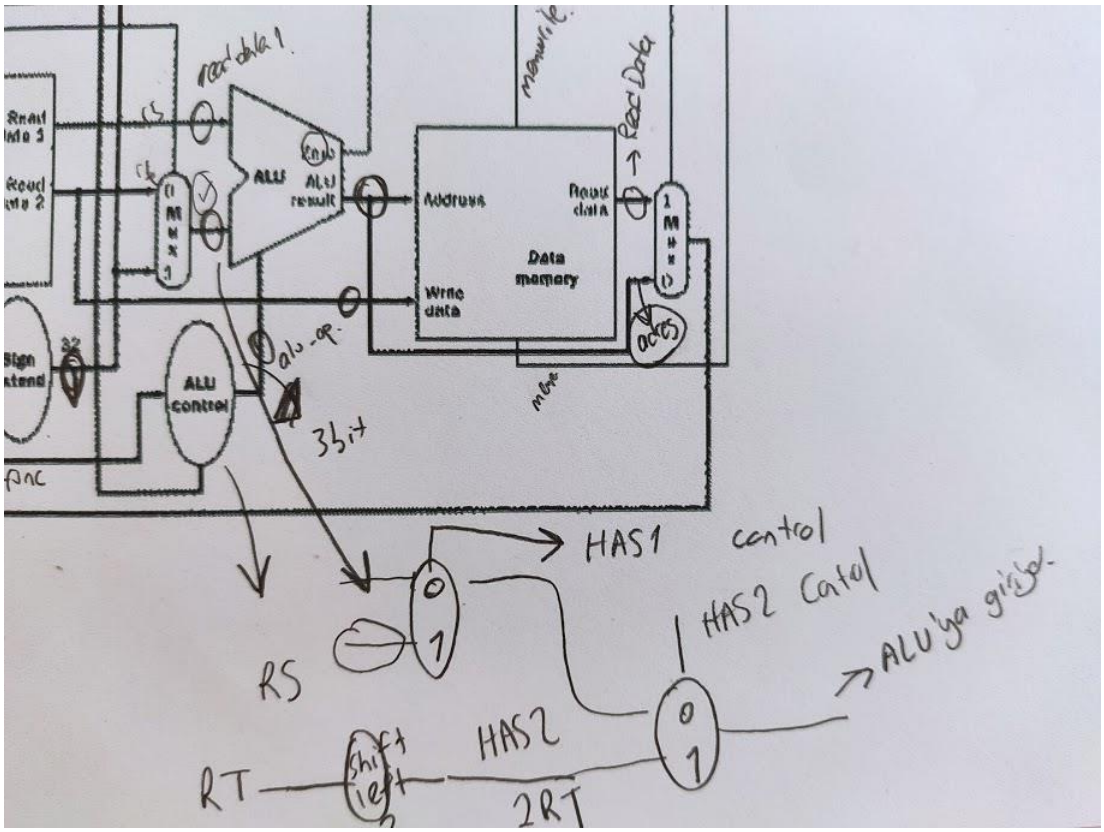
J TYPE

j	PC = JUMP-ADDER
jal	R[31] = PC + 8, PC = JUMP-ADDER
jr	PC = RS

Ekstra iki tane R-Type instruction ekledim. R-Type oldukları için opcode'ları 0 bit olmalı.

Has1	RD <- RS + RS
Has2	RD <- RS + RT + RT

Function fieldleri toplama işlemi yapılacağı için 100000 olarak ayarlandı. Insruction datapat'i aşağıdaki gibidir. Resimdeki RT ve RS ler, içeriğidir. $RS ve $RT'dir.



Aşşağıdaki ssler mips32 testbench modülü instructionları na aittir. Ek olarak tüm ssler bir klasörde toplanmıştır. Oradan detaylı bakılabilir.

```
# time:  0,
# PC: 00000000000000000000000000000000,
# instruction: 10101111111101111000000100000000,
# Opcode= 101011 ,
# rs = 11111 ,
# rt = 01111 ,
# rd = 00000,
# shamt = 00100,
# function = 000000,
# immediate = 0000000100000000,
# Mem_read = 0 ,
# Reg_Write = 0,
# Mem_Write = 1,
# RegDst = 0,
# Jump = 0,
# Branch = 0,
# MemToReg = 0,
# ALUop = 00,
# ALUsrc = 1,
# WriteData = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,
# Read_Data_1 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,
# Read_Data_2 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#
#
# time: 50,
# PC: 00000000000000000000000000000001,
# instruction: 00000000010000010001100000100000,
# Opcode= 000000 ,
# rs = 00010 ,
# rt = 00001 ,
# rd = 00011,
# shamt = 00000,
# function = 100000,
# immediate = 0001100000100000,
# Mem_read = 0 ,
# Reg_Write = 1,
# Mem_Write = 0,
# RegDst = 1,
# Jump = 0,
# Branch = 0,
# MemToReg = 0,
# ALUop = 10,
# ALUsrc = 0,
# WriteData = 00000000000000000000000000011110,
# Read_Data_1 = 00000000000000000000000000011111,
# Read_Data_2 = 11111111111111111111111111111111
#
```

```
time: 150,
PC: 00000000000000000000000000000010,
instruction: 00000000011001000010100000100001,
Opcode= 000000 ,
rs = 00011 ,
rt = 00100 ,
rd = 00101,
shamt = 00000,
function = 100001,
immediate = 0010100000100001,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000001,
Read_Data_1 = 00000000000000000000000000000010,
Read_Data_2 = 00000000000000000000000000000001


time: 250,
PC: 00000000000000000000000000000011,
instruction: 00000000101001100011100000100100,
Opcode= 000000 ,
rs = 00101 ,
rt = 00110 ,
rd = 00111,
shamt = 00000,
function = 100100,
immediate = 0011100000100100,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000100,
Read_Data_1 = 00000000000000000000000000011110,
Read_Data_2 = 00000000000000000000000000000100
```

```
time: 350,
PC: 00000000000000000000000000000100,
instruction: 00000000111010000100100000100101,
Opcode= 000000 ,
rs = 00111 ,
rt = 01000 ,
rd = 01001,
shamt = 00000,
function = 100101,
immediate = 0100100000100101,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000111,
Read_Data_1 = 00000000000000000000000000000001,
Read_Data_2 = 00000000000000000000000000000110


time: 450,
PC: 00000000000000000000000000000101,
instruction: 00000001010010010101100000100111,
Opcode= 000000 ,
rs = 01010 ,
rt = 01001 ,
rd = 01011,
shamt = 00000,
function = 100111,
immediate = 0101100000100111,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 11111111111111111111111111110100,
Read_Data_1 = 00000000000000000000000000000100,
Read_Data_2 = 00000000000000000000000000001111
```

```
time: 550,
PC: 00000000000000000000000000000110,
instruction: 00000011111011000110100100000000,
Opcode= 000000 ,
rs = 11111 ,
rt = 01100 ,
rd = 01101,
shamt = 00100,
function = 000000,
immediate = 0110100100000000,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000010101,
Read_Data_1 = 00000000000000000000000000001110,
Read_Data_2 = 00000000000000000000000000000111


time: 650,
PC: 00000000000000000000000000000111,
instruction: 00000001010011100111100010000010,
Opcode= 000000 ,
rs = 01010 ,
rt = 01110 ,
rd = 01111,
shamt = 00010,
function = 000010,
immediate = 0111100010000010,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 11111111111111111111111111111111,
Read_Data_1 = 00000000000000000000000000011111,
Read_Data_2 = 00000000000000000000000000100000
```

```
time: 750,
PC: 00000000000000000000000000001000,
instruction: 00000011111100001000100000100010,
Opcode= 000000 ,
rs = 11111 ,
rt = 10000 ,
rd = 10001,
shamt = 00000,
function = 100010,
immediate = 1000100000100010,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000010,
Read_Data_1 = 00000000000000000000000000011110,
Read_Data_2 = 00000000000000000000000000001100


time: 850,
PC: 00000000000000000000000000001001,
instruction: 00000011110100011001000000100011,
Opcode= 000000 ,
rs = 11110 ,
rt = 10001 ,
rd = 10010,
shamt = 00000,
function = 100011,
immediate = 1001000000100011,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000000,
Read_Data_1 = 00000000000000000000000000011111,
Read_Data_2 = 00000000000000000000000000010000
```

```
time: 950,
PC: 00000000000000000000000000001010,
instruction: 00000010010111011010000000101010,
Opcode= 000000 ,
rs = 10010 ,
rt = 11101 ,
rd = 10100,
shamt = 00000,
function = 101010,
immediate = 1010000000101010,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000000,
Read_Data_1 = 00000000000000000000000000011110,
Read_Data_2 = 00000000000000000000000000000010


time: 1050,
PC: 00000000000000000000000000001011,
instruction: 00000010111010101101010000101011,
Opcode= 000000 ,
rs = 10110 ,
rt = 10101 ,
rd = 10101,
shamt = 00000,
function = 101011,
immediate = 1010100000101011,
Mem_read = 0 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 1,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 10,
ALUsrc = 0,
WriteData = 00000000000000000000000000000001,
Read_Data_1 = 00000000000000000000000000000000,
Read_Data_2 = 00000000000000000000000000011101
```

```
time: 1150,
PC: 00000000000000000000000000001100,
instruction: 00100010101101110000000000010010,
Opcode= 001000 ,
rs = 10101 ,
rt = 10111 ,
rd = 00000,
shamt = 00000,
function = 010010,
immediate = 0000000000010010,
Mem_read = 0 ,
Reg_Write = 0,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 00,
ALUsrc = 0,
WriteData = 00000000000000000000000000101011,
Read_Data_1 = 00000000000000000000000000010110,
Read_Data_2 = 00000000000000000000000000010101


time: 1250,
PC: 00000000000000000000000000001101,
instruction: 00110010111111001000000000011011,
Opcode= 001100 ,
rs = 10111 ,
rt = 11001 ,
rd = 00000,
shamt = 00000,
function = 011011,
immediate = 0000000000011011,
Mem_read = 0 ,
Reg_Write = 0,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 00,
ALUsrc = 0,
WriteData = 00000000000000000000000000011000,
Read_Data_1 = 00000000000000000000000000000001,
Read_Data_2 = 00000000000000000000000000010111
```

```
time: 1350,
PC: 00000000000000000000000000001110,
instruction: 00100111001110100000000000100011,
Opcode= 001001 ,
rs = 11001 ,
rt = 11010 ,
rd = 00000,
shamt = 00000,
function = 100011,
immediate = 0000000000100011,
Mem_read = 0 ,
Reg_Write = 0,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 00,
ALUsrc = 0,
WriteData = 00000000000000000000000000110000,
Read_Data_1 = 00000000000000000000000000010111,
Read_Data_2 = 00000000000000000000000000011001


time: 1450,
PC: 00000000000000000000000000001111,
instruction: 00110111010111011000000000011001,
Opcode= 001101 ,
rs = 11010 ,
rt = 11011 ,
rd = 00000,
shamt = 00000,
function = 011001,
immediate = 0000000000011001,
Mem_read = 0 ,
Reg_Write = 0,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 00,
ALUsrc = 0,
WriteData = 00000000000000000000000000110011,
Read_Data_1 = 00000000000000000000000000011001,
Read_Data_2 = 00000000000000000000000000011010
```

```
time: 1550,
PC: 00000000000000000000000000010000,
instruction: 00101011101111000000000000011101,
Opcode= 001010 ,
rs = 11101 ,
rt = 11100 ,
rd = 00000,
shamt = 00000,
function = 011101,
immediate = 0000000000011101,
Mem_read = 0 ,
Reg_Write = 0,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 0,
ALUop = 00,
ALUsrc = 0,
WriteData = 00000000000000000000000000110101,
Read_Data_1 = 00000000000000000000000000011010,
Read_Data_2 = 00000000000000000000000000011011


time: 1650,
PC: 00000000000000000000000000010001,
instruction: 10001111110111110000000001000000,
Opcode= 100011 ,
rs = 11110 ,
rt = 11111 ,
rd = 00000,
shamt = 00001,
function = 000000,
immediate = 0000000001000000,
Mem_read = 1 ,
Reg_Write = 1,
Mem_Write = 0,
RegDst = 0,
Jump = 0,
Branch = 0,
MemToReg = 1,
ALUop = 00,
ALUsrc = 1,
WriteData = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,
Read_Data_1 = 00000000000000000000000000011101,
Read_Data_2 = 00000000000000000000000000011100
```