HOMEWORK 4

Yunus Emre Geyik – 1801042635

TEST BENCHES SEPERATLY

ALU CONTROL TEST BENCH

```
Transcript
VSIM 6> step -current
# Time= 0, ALUOp:100, Func:000, Output:110
# Time=20, ALUOp:100, Func:001, Output:000
# Time=40, ALUOp:100, Func:010, Output:010
# Time=60, ALUOp:100, Func:011, Output:001
# Time=80, ALUOp:100, Func:100, Output:101
# Time=100, ALUOp:100, Func:101, Output:111
# Time=120, ALUOp:000, Func:xxx, Output:000
# Time=140, ALUOp:001, Func:xxx, Output:110
# Time=160, ALUOp:111, Func:xxx, Output:111
# Time=180, ALUOp:011, Func:xxx, Output:101
# Time=200, ALUOp:010, Func:xxx, Output:010
# Time=240, ALUOp:101, Func:xxx, Output:100
# Time=260, ALUOp:000, Func:xxx, Output:000

VSIM 7>
```

COMPERATOR TEST BENCH

```
Transcript
#
# Top level modules:
#       comparator_testbench
ModelSim> vsim work.comparator_testbench
# vsim work.comparator_testbench
# Loading work.comparator_testbench
# Loading work.comparator
add wave -position insertpoint  \
sim:/comparator_testbench/_input \
sim:/comparator_testbench/Equal \
sim:/comparator_testbench/NotEqual
VSIM 6> step -current
# time =  0, _input =00000000000000000000000000001111, Equal=0, NotEqual=1
# time = 20, _input =00000000000000000000000000000000, Equal=1, NotEqual=0

VSIM 7>
```

INSTRUCTION MEMORY TEST BENCH

```
Transcript
# -- Compiling module instruction_memory_testbench
#
# Top level modules:
#       instruction_memory_testbench
ModelSim> vsim work.instruction_memory_testbench
# vsim work.instruction_memory_testbench
# Loading work.instruction_memory_testbench
# Loading work.instruction_memory
add wave -position insertpoint  \
sim:/instruction_memory_testbench/_pc \
sim:/instruction_memory_testbench/_instruction
VSIM 6> step -current
# Time= 0, _pc:00000000000000000000000000000000, _instruction:0000001000010000
# Time=20, _pc:00000000000000000000000000000001, _instruction:0000010001011000

VSIM 7>
```

EXTENDER TEST BENCH

```
Transcript
# -- Compiling module extender_testbench
#
# Top level modules:
#       extender_testbench
ModelSim> vsim work.extender_testbench
# vsim work.extender_testbench
# Loading work.extender_testbench
# Loading work.extender
add wave -position insertpoint  \
sim:/extender_testbench/_input \
sim:/extender_testbench/_output
VSIM 6> step -current
# time =  0, _input =110111, _output=11111111111111111111111111110111
# time = 20, _input =001011, _output=00000000000000000000000000001011

VSIM 7>
```

BEFORE REGISTER TEST BENCH

```
mem_register.mem - Not Defteri
Dosya  Düzen  Biçim  Görünüm  Yardım
// memory data file (do not edit the following line - required for mem load use)
// instance=/register_testbench/atb/_registers
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

REGISTER TEST BENCH

```
Transcript
sim:/register_testbench/Rt \
sim:/register_testbench/Rd_or_Rt \
sim:/register_testbench/WriteData \
sim:/register_testbench/RegWrite \
sim:/register_testbench/Read_Rs_Data \
sim:/register_testbench/Read_Rt_Data \
sim:/register_testbench/clock
VSIM 6> step -current
# Time= 0, Rs:110, Rt:100, Rd_or_Rt:xxx, WriteData:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, RegWrite:0, Read_Rs_Data:00000000000000000000000000000110, Read_Rt_Data:00000000000000000000000000000100
# Time=20, Rs:010, Rt:001, Rd_or_Rt:011, WriteData:00000000000000000000000011110101, RegWrite:1, Read_Rs_Data:00000000000000000000000000000010, Read_Rt_Data:00000000000000000000000000000001
# ** Note: $finish    : D:/Emre/OneDrive - GTÜ/Masaüstü/HW4/register_testbench.v(31)
#    Time: 60 ps  Iteration: 0  Instance: /register_testbench
# 1
# Break in Module register_testbench at D:/Emre/OneDrive - GTÜ/Masaüstü/HW4/register_testbench.v line 31

VSIM 7>
```

AFTER REGISTER TESTBENCH

mem_register_output.mem - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/register_testbench/atb/_registers
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000011110101
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

BEFORE THE DATA MEMORY TEST BENCH

mem_memory.mem - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/DataMemory_testbench/atb/DataMem
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000001000
00000000000000000000000000000011
00000000000000000000000000001001
00000000000000000000000000011010
00000000000000000000000000000000
00000000000000000000000001010101
00000000000000000000000000101010
00000000000000000000000000010010
00000000000000000000000000000010
00000000000000000000000000000101
00000000000000000000000000011100
00000000000000000000000000000111
```

DATA MEMORY TEST BENCH

Transcript

```
#        DataMemory_testbench
ModelSim> vsim work.DataMemory_testbench
# vsim work.DataMemory_testbench
# Loading work.DataMemory_testbench
# Loading work.DataMemory
add wave -position insertpoint  \
sim:/DataMemory_testbench/Adress \
sim:/DataMemory_testbench/WriteData \
sim:/DataMemory_testbench/ReadData \
sim:/DataMemory_testbench/MemoryRead \
sim:/DataMemory_testbench/MemoryWrite
VSIM 6> step -current
# Time= 0, Adress:00000000000000000000000000000100, WriteData:00000000000000000000000000001001, MemoryRead:1, MemoryWrite:0, ReadData:00000000000000000000000000001001
# Time=20, Adress:00000000000000000000000000000010, WriteData:00000000000000000000000000001100, MemoryRead:0, MemoryWrite:1, ReadData:00000000000000000000000000001001

VSIM 7>
```

AFTER THE DATA MEMORY TEST BENCH

mem_memory.mem - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/DataMemory_testbench/atb/DataMem
// format=bin addressradix=h dataradix=b version=1.0 wordsperline=1 noaddress
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000001100
00000000000000000000000000000011
00000000000000000000000000001001
00000000000000000000000000011010
00000000000000000000000000000000
00000000000000000000000001010101
00000000000000000000000000101010
00000000000000000000000000010010
00000000000000000000000000000010
```

PROGRAM COUNTER TESTBENCH

Transcript

```
# vsim work.program_counter_testbench
# Loading work.program_counter_testbench
# Loading work.program_counter
add wave -position insertpoint  \
sim:/program_counter_testbench/_input \
sim:/program_counter_testbench/_output \
sim:/program_counter_testbench/_clock
VSIM 6> step -current
# Time= 0, _input =00000000000000000000000000000000, _output=00000000000000000000000000000000, _clock=0
# Time= 5, _input =00000000000000000000000010011111, _output=00000000000000000000000010011111, _clock=1
# ** Note: $finish    : D:/Emre/OneDrive - GTÜ/Masaüstü/HW4/program_counter_testbench.v(31)
#    Time: 10 ps  Iteration: 0  Instance: /program_counter_testbench
# 1
# Break in Module program_counter_testbench at D:/Emre/OneDrive - GTÜ/Masaüstü/HW4/program_counter_testbench.v line 31

VSIM 7>
```

SHIFTLEFT2 TEST BENCH

Transcript

```
# -- Compiling module shiftleft2_testbench
#
# Top level modules:
#       shiftleft2_testbench
ModelSim> vsim work.shiftleft2_testbench
# vsim work.shiftleft2_testbench
# Loading work.shiftleft2_testbench
# Loading work.shiftleft2
add wave -position insertpoint  \
sim:/shiftleft2_testbench/_input \
sim:/shiftleft2_testbench/_output
VSIM 6> step -current
# Time: 0 _input:00000000000000000000000000001111, _output: 00000000000000000000000000111100
# Time:20 _input:00000000000000000000000000001101, _output: 00000000000000000000000000110100

VSIM 7>
```

MUX 3x1 BİT TEST BENCH

```
Transcript
#        mux1bit
ModelSim> vsim work.mux3bit_testbench
# vsim work.mux3bit_testbench
# Loading work.mux3bit_testbench
# Loading work.mux3bit
# Loading work.mux1bit
add wave -position insertpoint  \
sim:/mux3bit_testbench/_input1 \
sim:/mux3bit_testbench/_input2 \
sim:/mux3bit_testbench/_output \
sim:/mux3bit_testbench/selectBit
VSIM 7> step -current
# time =  0, _input1 =111, _input2=001, _output=001, selectBit=1
# time = 20, _input1 =011, _input2=101, _output=011, selectBit=0

VSIM 8>
```

CONTROL UNIT TEST BENCH

```
Transcript
sim:/Control_Unit_testbench/ALUSrc \
sim:/Control_Unit_testbench/ALUOp \
sim:/Control_Unit_testbench/BranchNot
VSIM 6> step -current
# Time= 0, Opcode=0000, RegWrite=1, RegDest=1, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=0, ALUOp:100, BranchNot:0
# Time=20, Opcode=0001, RegWrite=1, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=1, ALUOp:000, BranchNot:0
# Time=40, Opcode=0010, RegWrite=1, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=1, ALUOp:001, BranchNot:0
# Time=60, Opcode=0011, RegWrite=1, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=1, ALUOp:111, BranchNot:0
# Time=80, Opcode=0100, RegWrite=1, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=1, ALUOp:011, BranchNot:0
# Time=100, Opcode=0101, RegWrite=0, RegDest=0, MemtoReg=0, Branch:1, MemRead=0, MemWrite=0, ALUSrc=0, ALUOp:010, BranchNot:0
# Time=120, Opcode=0110, RegWrite=0, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=0, ALUOp:010, BranchNot:1
# Time=140, Opcode=0111, RegWrite=1, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=0, ALUSrc=1, ALUOp:101, BranchNot:0
# Time=160, Opcode=1000, RegWrite=1, RegDest=0, MemtoReg=1, Branch:0, MemRead=1, MemWrite=0, ALUSrc=1, ALUOp:000, BranchNot:0
# Time=180, Opcode=1001, RegWrite=0, RegDest=0, MemtoReg=0, Branch:0, MemRead=0, MemWrite=1, ALUSrc=1, ALUOp:000, BranchNot:0

VSIM 7>
```

MY INSTRUCTIONS

	Rd = Rs Rt	Rs	Rt	Rd
AND	$2 $1, $0	001	000	010
	$3, $2, $1	010	001	011
ADD	$5, $4, $3	100	011	101
	$7, $6, $5	110	101	111
SUB	$5, $4, $3	100	011	101
	$7, $6, $5	110	101	111
XOR	$3, $5, $6	101	110	011
	$4, $7, $2	111	010	100
NOR	$2, $1, $0	001	000	010
	$3, $2, $1	010	001	011
OR	$3, $5, $6	101	110	011
	$4, $7, $2	111	010	100

I - Type

opcode	Rs	Rt	IM
			000 000

ADDI $Rt = Rs + IM$

$5, $4, 4	→ 100	101	000100
$6, $3, 18	→ 011	110	010010

ANDI

$3, $2, 5	→ 010	011	000101
$4, $1, 6	→ 001	100	000110

ORI

$6, $7, 7	→ 111	110	000111
$1, $4, 3	→ 100	001	000011

NORI

$7, $0, 10	→ 000	111	001010
$4, $3, 16	→ 011	100	010000

BEQ

$4, $3, Rosto	→ 011	100	011001
$6, $5, Porfotol	→ 101	110	001101

BNE

$3, $1, Flma	→ 001	011	101000
$5, $7, porfotol	→ 111	101	111001

SLTI

$4, $5, 12	→ 101	100	001100
$1, $6, 15	→ 110	001	001111

LW

$5, 32 ($3)	→ 011	101	100000
$2, 16 ($4)	→ 100	010	010000

SW

$4, 12 ($6)	→ 110	100	001100
$1, 20 ($7)	→ 111	001	010100

ALU CONTROL BITS

ALU Control Bits

Instruction Opcode	ALUop	$F_2F_1F_0$ Function	Desired ALU Action	ALU Control
AND	100	000	and	110
ADD	100	001	add	000
SUB	100	010	subtract	010
XOR	100	011	xor	001
NOR	100	100	nor	101
OR	100	101	or	111
ADDI	000	xxx	add	000
ANDI	001	xxx	and	110
ORI	111	xxx	or	111
NORI	011	xxx	nor	101
BEQ	010	xxx	subtract	010
BNE	010	xxx	subtract	010
SLTI	101	xxx	set on less than	100
LW	000	xxx	add	000
SW	000	xxx	add	000

THE TRUTH TABLE FOR THE 3 ALU CONTROL BITS

The Truth Table for the 3 ALU Control Bits

A	B	C	D	E	F			
(A_2)	(A_1)	(A_0)						
ALUOp2	ALUOp1	ALUOp0	F_2	F_1	F_0	OP_2	OP_1	OP_0
1	0	0	0	0	0	1	1	0
1	0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	0
1	0	0	0	1	1	0	0	1
1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	1	1	1
0	0	0	x	x	x	0	0	0
0	0	1	x	x	x	1	1	0
1	1	1	x	x	x	1	1	1
0	1	1	x	x	x	1	0	1
0	1	0	x	x	x	0	1	0
0	1	0	x	x	x	0	1	0
1	0	1	x	x	x	1	0	0
0	0	0	x	x	x	0	0	0
0	0	0	x	x	x	0	0	0

$$OP_2 = A_2 A_1' F_2 F_1' + A_2 A_1' F_1 F_0' + A_0$$

$$OP_1 = A_2 A_1' A_0' F_2' F_0' + A_2 A_1' A_0 F_2 F_1' F_0 + A_2' A_1' A_0 + A_2 A_1 A_0 + A_2' A_1 A_0'$$

$$OP_0 = A_2 A_1' A_0' F_2' F_1 F_0 + A_2 A_1' A_0' F_2 F_1' + A_1 A_0$$

RTL -> CONTROL

RTL → Control

Instr.	RegW	RegDst	MemtoReg	Branch	MR	MW	ALUSrc	3-bit ALUOp
AND	1	1	0	0	0	0	0	100
ADD	1	1	0	0	0	0	0	100
SUB	1	1	0	0	0	0	0	100
XOR	1	1	0	0	0	0	0	100
NOR	1	1	0	0	0	0	0	100
OR	1	1	0	0	0	0	0	100
ADDI	1	0	0	0	0	0	1	000
ANDI	1	0	0	0	0	0	1	001
ORI	1	0	0	0	0	0	1	111
NORI	1	0	0	0	0	0	1	011
BEQ	0	X	X	1	0	0	0	010
BNE	0	X	X	1	0	0	0	010
SLTI	1	0	0	0	0	0	1	101
LW	1	0	1	0	1	0	1	000
SW	0	X	X	0	0	1	1	000

TRUTH TABLE FOR THE CONTROL UNIT

Truth Table for the Control Unit

OpCode:	ADDI 0001	ANDI 0010	ORI 0011	NORI 0100	BEQ 0101	BNE 0110	SLTI 0111	LW 1000	SW 1001	(R Types) 0000
RegW	1	1	1	1	0	0	1	1	0	1
RegDst	0	0	0	0	X	X	0	0	X	1
MemtoReg	0	0	0	0	X	X	0	1	X	0
Branch	0	0	0	0	1	0	0	0	0	0
MR	0	0	0	0	0	0	0	1	0	0
MW	0	0	0	0	0	0	0	0	1	0
ALUSrc	1	1	1	1	0	0	1	1	1	0
ALUOp2	0	0	1	0	0	0	1	0	0	1
ALUOp1	0	0	1	1	1	1	0	0	0	0
ALUOp0	0	1	1	1	0	0	1	0	0	0
BranchNot	0	0	0	0	0	1	0	0	0	0

RegW = ADDI + ANDI + ORI + NORI + SLTI + LW + R Types

RegDst = R Types

MemtoReg = LW

Branch = BEQ + BNE

MR = LW

MW = SW

ALUSrc = ADDI + ANDI + ORI + NORI + SLTI + LW + SW

ALUOp2 = ORI + SLTI + R Types

ALUOp1 = ORI + NORI + BEQ + BNE

ALUOp0 = ANDI + ORI + NORI + SLTI

MINI MIPS

AND OPERATION

```
# clock =1, Instruction=00000010000010000
#  ProgramCounter=00000000000000000000000000000000, Opcode=0000, ALUOp=100, ALUControl=110, Rs=001, Rt=000, Rd=010, Func=000
#  AluResult=00000000000000000000000000000000, Rs_Data=00000000000000000000000000000001, Rt_Data=00000000000000000000000000000000, ExtendedNum=00000000000000000000000000010000
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=00000100001011000
#  ProgramCounter=00000000000000000000000000000001, Opcode=0000, ALUOp=100, ALUControl=110, Rs=010, Rt=001, Rd=011, Func=000
#  AluResult=00000000000000000000000000000000, Rs_Data=00000000000000000000000000000010, Rt_Data=00000000000000000000000000000001, ExtendedNum=00000000000000000000000000011000
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
```

ADD OPERATION

```
#
# clock =1, Instruction=00001000011101001
#  ProgramCounter=00000000000000000000000000000010, Opcode=0000, ALUOp=100, ALUControl=000, Rs=100, Rt=011, Rd=101, Func=001
#  AluResult=00000000000000000000000000000100, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000000, ExtendedNum=11111111111111111111111111101001
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=00001101011111001
#  ProgramCounter=00000000000000000000000000000011, Opcode=0000, ALUOp=100, ALUControl=000, Rs=110, Rt=101, Rd=111, Func=001
#  AluResult=00000000000000000000000000001010, Rs_Data=00000000000000000000000000000110, Rt_Data=00000000000000000000000000000100, ExtendedNum=11111111111111111111111111111001
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
```

SUB OPERATION

```
clock =1, Instruction=00001000011101010
 ProgramCounter=00000000000000000000000000000100, Opcode=0000, ALUOp=100, ALUControl=010, Rs=100, Rt=011, Rd=101, Func=010
 AluResult=00000000000000000000000000000100, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000000, ExtendedNum=11111111111111111111111111101010
 RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
clock =0, Instruction=00001101011111010
 ProgramCounter=00000000000000000000000000000101, Opcode=0000, ALUOp=100, ALUControl=010, Rs=110, Rt=101, Rd=111, Func=010
 AluResult=00000000000000000000000000000010, Rs_Data=00000000000000000000000000000110, Rt_Data=00000000000000000000000000000100, ExtendedNum=11111111111111111111111111111010
 RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
```

XOR OPERATION

```
#
# clock =1, Instruction=0000101110011011
#  ProgramCounter=00000000000000000000000000000110, Opcode=0000, ALUOp=100, ALUControl=001, Rs=101, Rt=110, Rd=011, Func=011
#  AluResult=00000000000000000000000000000010, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000110, ExtendedNum=00000000000000000000000000011011
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=0000111010100011
#  ProgramCounter=00000000000000000000000000000111, Opcode=0000, ALUOp=100, ALUControl=001, Rs=111, Rt=010, Rd=100, Func=011
#  AluResult=00000000000000000000000000000000, Rs_Data=00000000000000000000000000000010, Rt_Data=00000000000000000000000000000010, ExtendedNum=11111111111111111111111111100011
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
```

NOR OPERATION

```
#
# clock =1, Instruction=0000001000010100
#  ProgramCounter=00000000000000000000000000001000, Opcode=0000, ALUOp=100, ALUControl=101, Rs=001, Rt=000, Rd=010, Func=100
#  AluResult=11111111111111111111111111111110, Rs_Data=00000000000000000000000000000001, Rt_Data=00000000000000000000000000000000, ExtendedNum=00000000000000000000000000010100
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=0000010001011100
#  ProgramCounter=00000000000000000000000000001001, Opcode=0000, ALUOp=100, ALUControl=101, Rs=010, Rt=001, Rd=011, Func=100
#  AluResult=00000000000000000000000000000000, Rs_Data=11111111111111111111111111111110, Rt_Data=00000000000000000000000000000001, ExtendedNum=00000000000000000000000000011100
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
```

OR OPERATION

```
#
# clock =1, Instruction=0000101110011101
#  ProgramCounter=00000000000000000000000000001010, Opcode=0000, ALUOp=100, ALUControl=111, Rs=101, Rt=110, Rd=011, Func=101
#  AluResult=00000000000000000000000000000110, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000110, ExtendedNum=00000000000000000000000000011101
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=0000111010100101
#  ProgramCounter=00000000000000000000000000001011, Opcode=0000, ALUOp=100, ALUControl=111, Rs=111, Rt=010, Rd=100, Func=101
#  AluResult=11111111111111111111111111111110, Rs_Data=00000000000000000000000000000010, Rt_Data=11111111111111111111111111111110, ExtendedNum=11111111111111111111111111100101
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
```

ADDI OPERATION

```
#
# clock =1, Instruction=0001100101000100
#  ProgramCounter=00000000000000000000000000001100, Opcode=0001, ALUOp=000, ALUControl=000, Rs=100, Rt=101, Rd=000, Func=100
#  AluResult=00000000000000000000000000000010, Rs_Data=11111111111111111111111111111110, Rt_Data=00000000000000000000000000000010, ExtendedNum=00000000000000000000000000000100
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
#
# clock =0, Instruction=0001011110010010
#  ProgramCounter=00000000000000000000000000001101, Opcode=0001, ALUOp=000, ALUControl=000, Rs=011, Rt=110, Rd=010, Func=010
#  AluResult=00000000000000000000000000011000, Rs_Data=00000000000000000000000000000110, Rt_Data=00000000000000000000000000000110, ExtendedNum=00000000000000000000000000010010
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
```

ANDI OPERATION

```
#
# clock =1, Instruction=0010010011000101
#  ProgramCounter=00000000000000000000000000001110, Opcode=0010, ALUOp=001, ALUControl=110, Rs=010, Rt=011, Rd=000, Func=101
#  AluResult=00000000000000000000000000000100, Rs_Data=11111111111111111111111111111110, Rt_Data=00000000000000000000000000000100, ExtendedNum=00000000000000000000000000000101
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
#
# clock =0, Instruction=0010001100000110
#  ProgramCounter=00000000000000000000000000001111, Opcode=0010, ALUOp=001, ALUControl=110, Rs=001, Rt=100, Rd=000, Func=110
#  AluResult=00000000000000000000000000000000, Rs_Data=00000000000000000000000000000001, Rt_Data=11111111111111111111111111111110, ExtendedNum=00000000000000000000000000000110
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
```

ORI OPERATION

```
#
# clock =1, Instruction=0000101110011101
#  ProgramCounter=00000000000000000000000000001010, Opcode=0000, ALUOp=100, ALUControl=111, Rs=101, Rt=110, Rd=011, Func=101
#  AluResult=00000000000000000000000000000110, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000110, ExtendedNum=00000000000000000000000000011101
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=0000111010100101
#  ProgramCounter=00000000000000000000000000001011, Opcode=0000, ALUOp=100, ALUControl=111, Rs=111, Rt=010, Rd=100, Func=101
#  AluResult=11111111111111111111111111111110, Rs_Data=00000000000000000000000000000010, Rt_Data=11111111111111111111111111111110, ExtendedNum=11111111111111111111111111100101
#  RegWrite=1, RegDest=1, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
```

NORI OPERATION

```
#
# clock =1, Instruction=0100000111001010
#  ProgramCounter=00000000000000000000000000010010, Opcode=0100, ALUOp=011, ALUControl=101, Rs=000, Rt=111, Rd=001, Func=010
#  AluResult=11111111111111111111111111110101, Rs_Data=00000000000000000000000000000000, Rt_Data=11111111111111111111111111110101, ExtendedNum=00000000000000000000000000001010
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
#
# clock =0, Instruction=0100011100010000
#  ProgramCounter=00000000000000000000000000010011, Opcode=0100, ALUOp=011, ALUControl=101, Rs=011, Rt=100, Rd=010, Func=000
#  AluResult=11111111111111111111111111101011, Rs_Data=00000000000000000000000000000100, Rt_Data=00000000000000000000000000000000, ExtendedNum=00000000000000000000000000010000
#  RegWrite=1, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=1, BranchNot=0
#
#
```

BEQ OPERATION

```
#
# clock =1, Instruction=0101011100011001
#  ProgramCounter=00000000000000000000000000010100, Opcode=0101, ALUOp=010, ALUControl=010, Rs=011, Rt=100, Rd=011, Func=001
#  AluResult=00000000000000000000000000011001, Rs_Data=00000000000000000000000000000100, Rt_Data=11111111111111111111111111101011, ExtendedNum=00000000000000000000000000011001
#  RegWrite=0, RegDest=0, MemtoReg=0, Branch=1, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
#
# clock =0, Instruction=0101101110001101
#  ProgramCounter=00000000000000000000000000010101, Opcode=0101, ALUOp=010, ALUControl=010, Rs=101, Rt=110, Rd=001, Func=101
#  AluResult=11111111111111111111111111111011, Rs_Data=00000000000000000000000000000010, Rt_Data=00000000000000000000000000000111, ExtendedNum=00000000000000000000000000001101
#  RegWrite=0, RegDest=0, MemtoReg=0, Branch=1, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=0
#
#
```

BNE OPERATION

```
#
# clock =0, Instruction=01100010111101000
#  ProgramCounter=00000000000000000000000000010110, Opcode=0110, ALUOp=010, ALUControl=010, Rs=001, Rt=011, Rd=101, Func=000
#  AluResult=11111111111111111111111111111111, Rs_Data=00000000000000000000000000000011, Rt_Data=00000000000000000000000000000100, ExtendedNum=11111111111111111111111111101000
#  RegWrite=0, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=1
#
#
#
# clock =1, Instruction=01100010111101000
#  ProgramCounter=00000000000000000000000000010110, Opcode=0110, ALUOp=010, ALUControl=010, Rs=001, Rt=011, Rd=101, Func=000
#  AluResult=11111111111111111111111111111111, Rs_Data=00000000000000000000000000000011, Rt_Data=00000000000000000000000000000100, ExtendedNum=11111111111111111111111111101000
#  RegWrite=0, RegDest=0, MemtoReg=0, Branch=0, MemRead=0, MemWrite=0, ALUSrc=0, BranchNot=1
#
#
```