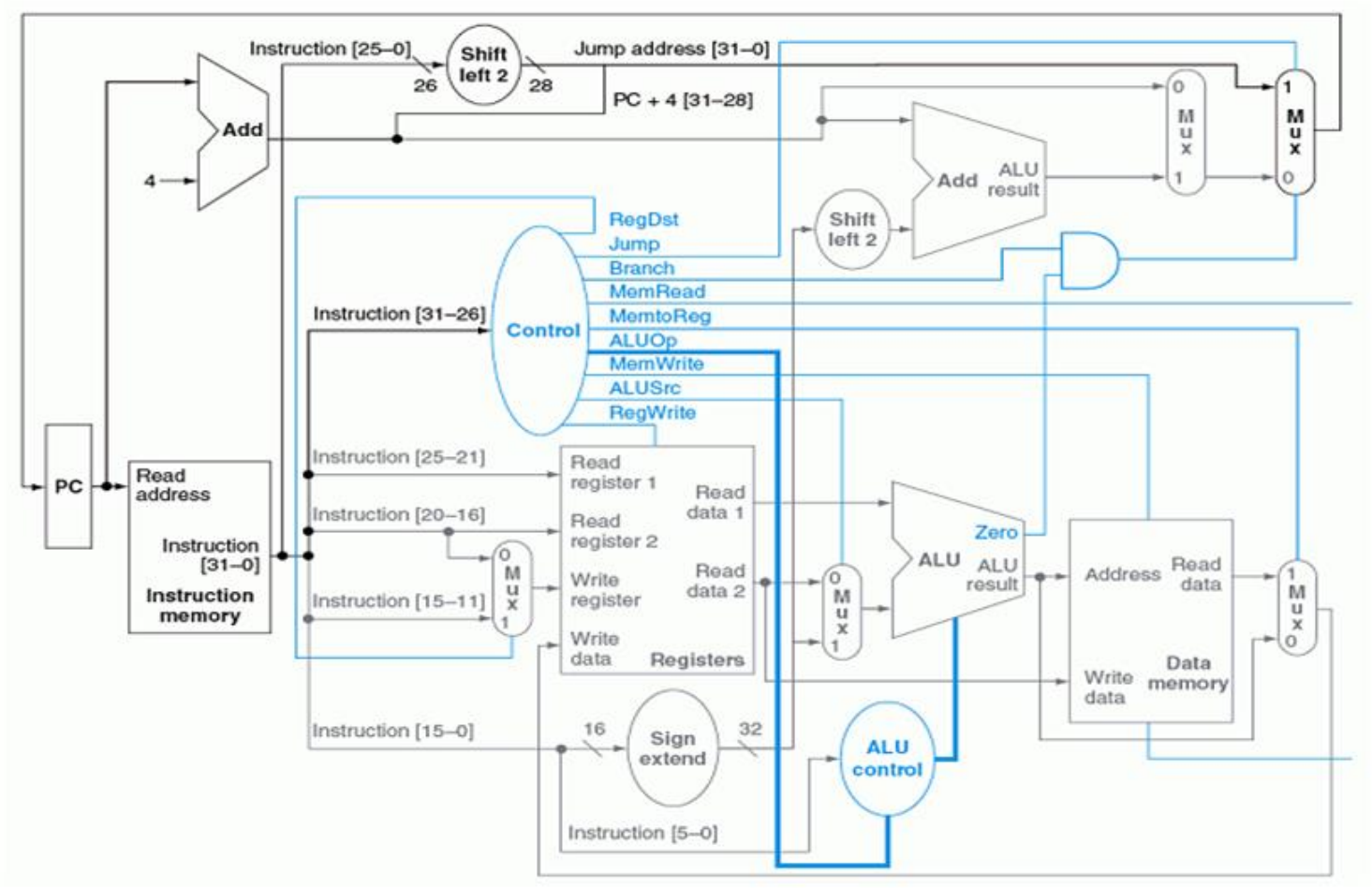
CSE 331/503-COMPUTER ORGANIZATION

#HW3

Atakan ALTIN

1801042668

Mips Processor



- In this homework, mips processor which is shown in the image is implemented.
- Jump operation does not supported and it takes 16 bit instruction so it is called mini mips processor.
- All asked instructions is implemented and works correctly.

Truth Table for Control Unit

Control Unit	0000 R-TYPE	0001 ADDI	0010 ANDI	0011 ORI	0100 NORI	0101 BEQ	0110 BNE	0111 SLTI	1000 LW	1001 SW
regDest	1	0	0	0	0	0	0	0	0	0
ALUSrc	0	1	1	1	1	0	0	1	1	1
memToReg	0	0	0	0	0	0	0	0	1	0
regWrite	1	1	1	1	1	0	0	1	1	0
memRead	0	0	0	0	0	0	0	0	1	0
memWrite	0	0	0	0	0	0	0	0	0	1
Branch	0	0	0	0	0	1	0	0	0	0
BranchNot	0	0	0	0	0	0	1	0	0	0
ALUOp_2	0	0	1	1	1	0	0	1	0	0
ALUOp_1	1	0	1	1	0	1	1	0	0	0
ALUOp_0	1	0	0	1	1	0	0	0	0	0
ALUOP	R-TYPE	add	and	or	nor	sub	sub	Set Less than	add	add
ZeroExtend	0	0	1	1	1	0	0	0	0	0

ALU Control Signals

ALU Control

Instruction	ALUOP	Function Field	Action	Alu Control
AND	011	000	AND	110
ADD	011	001	ADD	000
SUB	011	010	SUB	010
XOR	011	011	XOR	001
NOR	011	100	NOR	101
OR	011	101	OR	111
ADDI	000	X	ADD	000
ANDI	110	X	AND	110
ORI	111	X	OR	111
NORI	101	X	NOR	101
BEQ	010	X	SUB	010
BNE	010	X	SUB	010
SLTI	100	X	SLT	100
LW	000	X	ADD	000
SW	000	X	ADD	000

Testbench Results

1. Main Control Testbench

```
# time=  0,
# opcode= 0000,
# RegDst= 1,
# ALUsrc= 0,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 011
#
# time= 20,
# opcode= 0001,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 000
#
```

```
# time= 40,
# opcode= 0010,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 1,
# ALUop= 110
#
# time= 60,
# opcode= 0011,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 1,
# ALUop= 111
#
```

```
# time= 80,
# opcode= 0100,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 1,
# ALUop= 101
#
# time= 100,
# opcode= 0101,
# RegDst= 0,
# ALUsrc= 0,
# MemtoReg= 0,
# RegWrite= 0,
# MemRead= 0,
# MemWrite= 0,
# Branch= 1,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 010
#
```

```
# time= 120,
# opcode= 0110,
# RegDst= 0,
# ALUsrc= 0,
# MemtoReg= 0,
# RegWrite= 0,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 1,
# zeroExt= 0,
# ALUop= 010
#
# time= 140,
# opcode= 0111,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 1,
# MemRead= 0,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 100
#
```

```
# time= 160,
# opcode= 1000,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 1,
# RegWrite= 1,
# MemRead= 1,
# MemWrite= 0,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 000
#
# time= 180,
# opcode= 1001,
# RegDst= 0,
# ALUsrc= 1,
# MemtoReg= 0,
# RegWrite= 0,
# MemRead= 0,
# MemWrite= 1,
# Branch= 0,
# Branch_not= 0,
# zeroExt= 0,
# ALUop= 000
#
```

2. ALU Control Testbench

```
# time=  0,AluOp= 011,func= 000,AluControl= 110
#
# time= 20,AluOp= 011,func= 001,AluControl= 000
#
# time= 40,AluOp= 011,func= 010,AluControl= 010
#
# time= 60,AluOp= 011,func= 011,AluControl= 001
#
# time= 80,AluOp= 011,func= 100,AluControl= 101
#
# time= 100,AluOp= 011,func= 101,AluControl= 111
#
# time= 120,AluOp= 110,func= 101,AluControl= 110
#
# time= 140,AluOp= 111,func= 101,AluControl= 111
#
# time= 160,AluOp= 101,func= 101,AluControl= 101
#
# time= 180,AluOp= 010,func= 101,AluControl= 010
#
# time= 220,AluOp= 100,func= 101,AluControl= 100
#
# time= 240,AluOp= 000,func= 101,AluControl= 000
#
```

3. **Registers Testbench**

```
VSIM 40> step -out -current
# time =  0, clk = 1 read_reg_1 =   001, read_reg_2 =   010, read_data_1 = 00000000000000000000000000000001, read_data_2 = 00000000000000000000000000000010, write_register = 011 ,write_data = 00000000111110000000111100001110,regWrite = 0
# time = 20, clk = 0 read_reg_1 =   011, read_reg_2 =   100, read_data_1 = 00000000000000000000000000000011, read_data_2 = 00000000000000000000000000000100, write_register = 011 ,write_data = 00000000111110000000111100001110,regWrite = 1
# time = 40, clk = 1 read_reg_1 =   011, read_reg_2 =   100, read_data_1 = 00000000111110000000111100001110, read_data_2 = 00000000000000000000000000000100, write_register = 011 ,write_data = 00000000111110000000111100001110,regWrite = 1
# time = 60, clk = 0 read_reg_1 =   000, read_reg_2 =   011, read_data_1 = 00000000000000000000000000000000, read_data_2 = 00000000111110000000111100001110, write_register = 000 ,write_data = 10111010111101001001111111111110,regWrite = 1
# time = 80, clk = 1 read_reg_1 =   000, read_reg_2 =   011, read_data_1 = 00000000000000000000000000000000, read_data_2 = 00000000111110000000111100001110, write_register = 000 ,write_data = 10111010111101001001111111111110,regWrite = 1

VSIM 41>
```

- Time 0-20 simply read data from register and try to write register when regWrite signal is 0.
- Time 20-40 simply read data from register and try to write register when regWrite signal is 1 and it succeeds.
- Time 60-80 try to write data to $zero register and it fails as it's supposed to be.

4. **Data Memory Testbench**

```
VSIM 9> step -out -current
# time =  0, address = 00000000000000000000000000000001, read_data = 00000000000000000000000000010100, write_data = 00000000000000000000000000011111, read_signal = 1, write_signal = 0, clk = 1
# time = 20, address = 00000000000000000000000000000001, read_data = 00000000000000000000000000010100, write_data = 00000000000000000000000000011111, read_signal = 1, write_signal = 1, clk = 0
# time = 40, address = 00000000000000000000000000000001, read_data = 00000000000000000000000000011111, write_data = 00000000000000000000000000011111, read_signal = 1, write_signal = 1, clk = 1
# time = 60, address = 00000000000000000000000000000010, read_data = 00000000000000000000000000000010, write_data = 11011110111100000001001000111110, read_signal = 1, write_signal = 1, clk = 0
# time = 80, address = 00000000000000000000000000000010, read_data = 11011110111100000001001000111110, write_data = 11011110111100000001001000111110, read_signal = 1, write_signal = 1, clk = 1
# time = 100, address = 00000000000000000000000000000111, read_data = 00000000000000000000000000000111, write_data = 11001101101000110100001000011100, read_signal = 1, write_signal = 1, clk = 0
# time = 120, address = 00000000000000000000000000000111, read_data = 11001101101000110100001000011100, write_data = 11001101101000110100001000011100, read_signal = 1, write_signal = 1, clk = 1

VSIM 10>
```

- You can see in the time 0-20 interval write data does not work because write_signal is 0. But in the time 20-40 data is updated because write_signal is 1.

5. **Instruction Memory Testbench**

```
initial begin
instMem.instruction_memory[0]=16'b0000000000000000;
instMem.instruction_memory[4]=16'b0000111111110000;
instMem.instruction_memory[8]=16'b1111000000000000;
```

```
# time=  0, read_address= 00000000000000000000000000000000, instruction= 0000000000000000, clk= 1
# time= 20, read_address= 00000000000000000000000000000000, instruction= 0000000000000000, clk= 0
# time= 40, read_address= 00000000000000000000000000000100, instruction= 0000111111110000, clk= 1
# time= 60, read_address= 00000000000000000000000000000100, instruction= 0000111111110000, clk= 0
# time= 80, read_address= 00000000000000000000000000001000, instruction= 1111000000000000, clk= 1
```

- You can see the instructions read correctly.

6. **Sign Extender Testbench**

```
VSIM 22> step -out -current
# time=  0, imm6= 000101, R= 00000000000000000000000000000101
# time= 20, imm6= 100101, R= 11111111111111111111111111100101
# time= 40, imm6= 110000, R= 11111111111111111111111111110000
```

- Sign extend is needed for addi,lw,sw etc.

7. **Zero Extender Testbench**

```
VSIM 27> step -out -current
# time=  0, imm6= 000101, R= 00000000000000000000000000000101
# time= 20, imm6= 110011, R= 00000000000000000000000000110011
# time= 40, imm6= 111111, R= 00000000000000000000000000111111
```

- Zero extend is needed for andi,ori etc.

8. **Mux 2x1 3b Testbench**

```
VSIM 31> step -out -current
# time =  0 ,a=001,b=010, Selection=0 ,Result=001
# time = 20 ,a=111,b=100, Selection=1 ,Result=100
```

- This mux is needed for register destination selection in MiniMips.

9. **Shift Left_32b 2 bit Testbench**

```
VSIM 35> step -out -current
# time=  0,     input= 11111111111111111111111111111111, output= 11111111111111111111111111111100
# time= 20,     input= 11110000111111111111111100000001, output= 11000011111111111111110000000100
# time= 40,     input= 00000000000000000000000000001111, output= 00000000000000000000000000111100
# time= 60,     input= 00000000000000000000000000000000, output= 00000000000000000000000000000000
```

- This is needed for branch operations as seen in the Mips diagram.

ModelSim Similation Results

Initial Datas:

data.txt - Notepad

File Edit Format View Help

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
```

Initial Registers:

registers.mem - Notepad

File Edit Format View Help

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Instructions:

```
test.instMem.instruction_memory[0]=16'b0110100101000010; // bne $4 $5 2 +
test.instMem.instruction_memory[4]=16'b0000001010011000; // and $3 = $2 & $1 ->does not work cause of bne
test.instMem.instruction_memory[8]=16'b0000001010001001; // add $1 = $2 + $1
test.instMem.instruction_memory[12]=16'b0110000000000010; // bne $0 $0 2 -
test.instMem.instruction_memory[16]=16'b0000100001011001; // add $3 = $4 + $1
test.instMem.instruction_memory[20]=16'b0101001010000010; // beq $1 $2 2 -
test.instMem.instruction_memory[24]=16'b0101000000000010; // beq $0 $0 2 -
test.instMem.instruction_memory[28]=16'b0000001010011000; // and $3 = $2 & $1 ->does not work cause of beq
test.instMem.instruction_memory[32]=16'b0000110111100000; // and $4 = $6 & $7
test.instMem.instruction_memory[36]=16'b0000011100010000; // and $2 = $3 & $4
test.instMem.instruction_memory[40]=16'b0000100011101010; //sub $5 = $4 - $3
test.instMem.instruction_memory[44]=16'b0000001111110010; //sub $6 = $1 - $7
test.instMem.instruction_memory[48]=16'b0000010011111011; //xor $7 = $2 xor $3
test.instMem.instruction_memory[52]=16'b0000100101001011; //xor $1 = $4 xor $5
test.instMem.instruction_memory[56]=16'b0000000001010100; // $2 = $0 nor $1
test.instMem.instruction_memory[60]=16'b0000101111011100; // $3 = $5 nor $7
test.instMem.instruction_memory[64]=16'b0000010101100101; // $4 = $2 or $5
test.instMem.instruction_memory[68]=16'b0000010000101101; // $5 = $2 or $0
test.instMem.instruction_memory[72]=16'b0001001110000101; // addi $6 = $1 + 5
test.instMem.instruction_memory[76]=16'b0001010111001001; // addi $7 = $2 + 9
test.instMem.instruction_memory[80]=16'b0010010001000110; // andi $1 = $2 & 6
test.instMem.instruction_memory[84]=16'b0010011010000111; // andi $2 = $3 & 7
test.instMem.instruction_memory[88]=16'b0011000011000001; // ori $3 = $0 || 1
test.instMem.instruction_memory[92]=16'b0011010100000101; // ori $4 = $2 || 5
test.instMem.instruction_memory[96]=16'b0100000101000001; // nori $5 = $0 ~|| 1
test.instMem.instruction_memory[100]=16'b0100010110000101; // nori $6 = $2 ~|| 5
test.instMem.instruction_memory[104]=16'b0111010111000101; // slti $7 = $2 <? 5
test.instMem.instruction_memory[108]=16'b0111011001001001; // slti $1 = $3 <? 9
test.instMem.instruction_memory[112]=16'b1000000010000001; // lw $2 = M[$0 + 1]
test.instMem.instruction_memory[116]=16'b1000010011000101; // lw $3 = M[$2 + 5]
test.instMem.instruction_memory[120]=16'b1001000111000010; // sw Mem[$0+2] = $7
test.instMem.instruction_memory[124]=16'b1001000110000011; // sw Mem[$0+3] = $6
```

- Indexes are incremented by 4 because program counter and shift left works according to this logic.

1. **bne $4 $5 2**

```
# time=  0, clock= 1, PC= 00000000000000000000000000000000, instruction= 0110100101000010,
# opcode= 0110, rs= 100, rt= 101, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000100, read_data_2= 00000000000000000000000000000101,
# write_data= 11111111111111111111111111111111,
# ALUop= 010, ALUcontrol= 010, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 1, extend_type= 0
#
#
# time= 20, clock= 0, PC= 00000000000000000000000000001000, instruction= 0110100101000010,
# opcode= 0110, rs= 100, rt= 101, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000100, read_data_2= 00000000000000000000000000000101,
# write_data= 11111111111111111111111111111111,
# ALUop= 010, ALUcontrol= 010, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 1, extend_type= 0
#
#
# time= 40, clock= 1, PC= 00000000000000000000000000001000, instruction= 0000001010001001,
# opcode= 0000, rs= 001, rt= 010, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000001, read_data_2= 00000000000000000000000000000010,
# write_data= 00000000000000000000000000000011,
# ALUop= 011, ALUcontrol= 000, ALUresult= 00000000000000000000000000000011, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000000010
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 60, clock= 0, PC= 00000000000000000000000000001100, instruction= 0000001010001001,
# opcode= 0000, rs= 001, rt= 010, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000001, read_data_2= 00000000000000000000000000000010,
# write_data= 00000000000000000000000000000011,
# ALUop= 011, ALUcontrol= 000, ALUresult= 00000000000000000000000000000011, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000000010
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

- You can see the program counter change. Instructions[4] are passed.

2. **add $1 = $2 + $1**

```
#
# time= 60, clock= 0, PC= 00000000000000000000000000001100, instruction= 0000001010001001,
# opcode= 0000, rs= 001, rt= 010, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000001, read_data_2= 00000000000000000000000000000010,
# write_data= 00000000000000000000000000000011,
# ALUop= 011, ALUcontrol= 000, ALUresult= 00000000000000000000000000000011, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000000010
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

3. **bne $0 $0 2 , add $3 = $4 + $1**

```
#
# time= 80, clock= 1, PC= 00000000000000000000000000001100, instruction= 0110000000000010,
# opcode= 0110, rs= 000, rt= 000, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000000,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000000
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 1, extend_type= 0
#
#
# time= 100, clock= 0, PC= 00000000000000000000000000010000, instruction= 0110000000000010,
# opcode= 0110, rs= 000, rt= 000, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000000,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000000
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 1, extend_type= 0
#
#
# time= 120, clock= 1, PC= 00000000000000000000000000010000, instruction= 0000100001011001,
# opcode= 0000, rs= 100, rt= 001, rd= 011, funct= 001, imm6= 011001
# read_data_1= 00000000000000000000000000000100, read_data_2= 00000000000000000000000000000011,
# write_data= 00000000000000000000000000000111,
# ALUop= 011, ALUcontrol= 000, ALUresult= 00000000000000000000000000000111, ALUzero= 0, resultExtended= 00000000000000000000000000011001, mux_result= 00000000000000000000000000000011
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 140, clock= 0, PC= 00000000000000000000000000010100, instruction= 0000100001011001,
# opcode= 0000, rs= 100, rt= 001, rd= 011, funct= 001, imm6= 011001
# read_data_1= 00000000000000000000000000000100, read_data_2= 00000000000000000000000000000011,
# write_data= 00000000000000000000000000000111,
# ALUop= 011, ALUcontrol= 000, ALUresult= 00000000000000000000000000000111, ALUzero= 0, resultExtended= 00000000000000000000000000011001, mux_result= 00000000000000000000000000000011
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

- You can see the program counter change. Because 0=0 , program counter does not jump and it does add operation.

4. beq $1 $2 2 , beq $0 $0 2, and $4 = $6 & $7

```
#
# time= 160, clock= 1, PC= 00000000000000000000000010100, instruction= 0101001010000010,
# opcode= 0101, rs= 001, rt= 010, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000011, read_data_2= 00000000000000000000000000000010,
# write_data= 00000000000000000000000000000001,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000010
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 1, Branch_not= 0, extend_type= 0
#
#
# time= 180, clock= 0, PC= 00000000000000000000000011000, instruction= 0101001010000010,
# opcode= 0101, rs= 001, rt= 010, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000011, read_data_2= 00000000000000000000000000000010,
# write_data= 00000000000000000000000000000001,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000010
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 1, Branch_not= 0, extend_type= 0
#
#
# time= 200, clock= 1, PC= 00000000000000000000000011000, instruction= 0101000000000010,
# opcode= 0101, rs= 000, rt= 000, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000000,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000000
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 1, Branch_not= 0, extend_type= 0
#
#
# time= 220, clock= 0, PC= 00000000000000000000000100000, instruction= 0101000000000010,
# opcode= 0101, rs= 000, rt= 000, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000000,
# ALUop= 010, ALUcontrol= 010, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000000
# RegDst= 0, ALUsrc= 0, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 0, Branch= 1, Branch_not= 0, extend_type= 0
#
#
# time= 240, clock= 1, PC= 00000000000000000000000100000, instruction= 0000110111100000,
# opcode= 0000, rs= 110, rt= 111, rd= 100, funct= 000, imm6= 100000
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 11111111111111111111111111100000, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 260, clock= 0, PC= 00000000000000000000000100100, instruction= 0000110111100000,
# opcode= 0000, rs= 110, rt= 111, rd= 100, funct= 000, imm6= 100000
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 11111111111111111111111111100000, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

- First beq does nothing because registers are not equal. Second beq performs jump adress because zero registers are equal. So in jumped address program does and operation.

5. and $2 = $3 & $4

```
time= 280, clock= 1, PC= 00000000000000000000000100100, instruction= 0000011100010000,
opcode= 0000, rs= 011, rt= 100, rd= 010, funct= 000, imm6= 010000
read_data_1= 00000000000000000000000000000111, read_data_2= 00000000000000000000000000000110,
write_data= 00000000000000000000000000000110,
ALUop= 011, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000010000, mux_result= 00000000000000000000000000000110
RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0


time= 300, clock= 0, PC= 00000000000000000000000101000, instruction= 0000011100010000,
opcode= 0000, rs= 011, rt= 100, rd= 010, funct= 000, imm6= 010000
read_data_1= 00000000000000000000000000000111, read_data_2= 00000000000000000000000000000110,
write_data= 00000000000000000000000000000110,
ALUop= 011, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000010000, mux_result= 00000000000000000000000000000110
RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
```

6. sub $5 = $4 - $3

```
#
# time= 320, clock= 1, PC= 00000000000000000000000101000, instruction= 0000100011101010,
# opcode= 0000, rs= 100, rt= 011, rd= 101, funct= 010, imm6= 101010
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 11111111111111111111111111111111,
# ALUop= 011, ALUcontrol= 010, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 11111111111111111111111101010, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 340, clock= 0, PC= 00000000000000000000000101100, instruction= 0000100011101010,
# opcode= 0000, rs= 100, rt= 011, rd= 101, funct= 010, imm6= 101010
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 11111111111111111111111111111111,
# ALUop= 011, ALUcontrol= 010, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 11111111111111111111111101010, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

7. sub $6 = $1 - $7

```
# time= 360, clock= 1, PC= 00000000000000000000000000101100, instruction= 0000001111110010,
# opcode= 0000, rs= 001, rt= 111, rd= 110, funct= 010, imm6= 110010
# read_data_1= 00000000000000000000000000000011, read_data_2= 00000000000000000000000000000111,
# write_data= 11111111111111111111111111111100,
# ALUop= 011, ALUcontrol= 010, ALUresult= 11111111111111111111111111111100, ALUzero= 0, resultExtended= 11111111111111111111111111110010, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 380, clock= 0, PC= 00000000000000000000000000110000, instruction= 0000001111110010,
# opcode= 0000, rs= 001, rt= 111, rd= 110, funct= 010, imm6= 110010
# read_data_1= 00000000000000000000000000000011, read_data_2= 00000000000000000000000000000111,
# write_data= 11111111111111111111111111111100,
# ALUop= 011, ALUcontrol= 010, ALUresult= 11111111111111111111111111111100, ALUzero= 0, resultExtended= 11111111111111111111111111110010, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

8. xor $7 = $2 xor $3

```
# time= 400, clock= 1, PC= 00000000000000000000000000110000, instruction= 0000010011111011,
# opcode= 0000, rs= 010, rt= 011, rd= 111, funct= 011, imm6= 111011
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 00000000000000000000000000000001,
# ALUop= 011, ALUcontrol= 001, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 11111111111111111111111111111011, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 420, clock= 0, PC= 00000000000000000000000000110100, instruction= 0000010011111011,
# opcode= 0000, rs= 010, rt= 011, rd= 111, funct= 011, imm6= 111011
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000111,
# write_data= 00000000000000000000000000000001,
# ALUop= 011, ALUcontrol= 001, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 11111111111111111111111111111011, mux_result= 00000000000000000000000000000111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
```

9. xor $1 = $4 xor $5

```
# time= 440, clock= 1, PC= 00000000000000000000000000110100, instruction= 0000100101001011,
# opcode= 0000, rs= 100, rt= 101, rd= 001, funct= 011, imm6= 001011
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111111,
# write_data= 11111111111111111111111111111001,
# ALUop= 011, ALUcontrol= 001, ALUresult= 11111111111111111111111111111001, ALUzero= 0, resultExtended= 00000000000000000000000000001011, mux_result= 11111111111111111111111111111111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 460, clock= 0, PC= 00000000000000000000000000111000, instruction= 0000100101001011,
# opcode= 0000, rs= 100, rt= 101, rd= 001, funct= 011, imm6= 001011
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111111,
# write_data= 11111111111111111111111111111001,
# ALUop= 011, ALUcontrol= 001, ALUresult= 11111111111111111111111111111001, ALUzero= 0, resultExtended= 00000000000000000000000000001011, mux_result= 11111111111111111111111111111111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
```

10.nor $2 = $0 $1

```
# time= 480, clock= 1, PC= 00000000000000000000000000111000, instruction= 0000000001010100,
# opcode= 0000, rs= 000, rt= 001, rd= 010, funct= 100, imm6= 010100
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111001,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 101, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000010100, mux_result= 11111111111111111111111111111001
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 500, clock= 0, PC= 00000000000000000000000000111100, instruction= 0000000001010100,
# opcode= 0000, rs= 000, rt= 001, rd= 010, funct= 100, imm6= 010100
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111001,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 101, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000010100, mux_result= 11111111111111111111111111111001
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
```

11. nor $3 = $5 $7

```
# time= 520, clock= 1, PC= 00000000000000000000000000111100, instruction= 0000101111011100,
# opcode= 0000, rs= 101, rt= 111, rd= 011, funct= 100, imm6= 011100
# read_data_1= 11111111111111111111111111111111, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000000000,
# ALUop= 011, ALUcontrol= 101, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000011100, mux_result= 00000000000000000000000000000001
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 540, clock= 0, PC= 00000000000000000000000001000000, instruction= 0000101111011100,
# opcode= 0000, rs= 101, rt= 111, rd= 011, funct= 100, imm6= 011100
# read_data_1= 11111111111111111111111111111111, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000000000,
# ALUop= 011, ALUcontrol= 101, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000011100, mux_result= 00000000000000000000000000000001
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

12.or $4 = $2 $5

```
# time= 560, clock= 1, PC= 00000000000000000000000001000000, instruction= 0000010101100101,
# opcode= 0000, rs= 010, rt= 101, rd= 100, funct= 101, imm6= 100101
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111111,
# write_data= 11111111111111111111111111111111,
# ALUop= 011, ALUcontrol= 111, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 11111111111111111111111111100101, mux_result= 11111111111111111111111111111111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 580, clock= 0, PC= 00000000000000000000000001000100, instruction= 0000010101100101,
# opcode= 0000, rs= 010, rt= 101, rd= 100, funct= 101, imm6= 100101
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111111,
# write_data= 11111111111111111111111111111111,
# ALUop= 011, ALUcontrol= 111, ALUresult= 11111111111111111111111111111111, ALUzero= 0, resultExtended= 11111111111111111111111111100101, mux_result= 11111111111111111111111111111111
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

13.or $5 = $2 $0

```
# time= 600, clock= 1, PC= 00000000000000000000000001000100, instruction= 0000010000101101,
# opcode= 0000, rs= 010, rt= 000, rd= 101, funct= 101, imm6= 101101
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 111, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 11111111111111111111111111101101, mux_result= 00000000000000000000000000000000
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 620, clock= 0, PC= 00000000000000000000000001001000, instruction= 0000010000101101,
# opcode= 0000, rs= 010, rt= 000, rd= 101, funct= 101, imm6= 101101
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000110,
# ALUop= 011, ALUcontrol= 111, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 11111111111111111111111111101101, mux_result= 00000000000000000000000000000000
# RegDst= 1, ALUsrc= 0, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

14.addi $6 = $1 + 5

```
# time= 640, clock= 1, PC= 00000000000000000000000001001000, instruction= 0001001110000101,
# opcode= 0001, rs= 001, rt= 110, rd= 000, funct= 101, imm6= 000101
# read_data_1= 11111111111111111111111111111001, read_data_2= 11111111111111111111111111111100,
# write_data= 11111111111111111111111111111110,
# ALUop= 000, ALUcontrol= 000, ALUresult= 11111111111111111111111111111110, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 660, clock= 0, PC= 00000000000000000000000001001100, instruction= 0001001110000101,
# opcode= 0001, rs= 001, rt= 110, rd= 000, funct= 101, imm6= 000101
# read_data_1= 11111111111111111111111111111001, read_data_2= 11111111111111111111111111111100,
# write_data= 11111111111111111111111111111110,
# ALUop= 000, ALUcontrol= 000, ALUresult= 11111111111111111111111111111110, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

15.addi $7 = $2 + 9

```
# time= 680, clock= 1, PC= 00000000000000000000000001001100, instruction= 0001010111001001,
# opcode= 0001, rs= 010, rt= 111, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000001111,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000001111, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000001001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 700, clock= 0, PC= 00000000000000000000000001010000, instruction= 0001010111001001,
# opcode= 0001, rs= 010, rt= 111, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000110, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000001111,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000001111, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000001001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

16.andi $1 = $2 & 6

```
# time= 720, clock= 1, PC= 00000000000000000000000001010000, instruction= 0010010001000110,
# opcode= 0010, rs= 010, rt= 001, rd= 000, funct= 110, imm6= 000110
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111001,
# write_data= 00000000000000000000000000000110,
# ALUop= 110, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000000110, mux_result= 00000000000000000000000000000110
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 740, clock= 0, PC= 00000000000000000000000001010100, instruction= 0010010001000110,
# opcode= 0010, rs= 010, rt= 001, rd= 000, funct= 110, imm6= 000110
# read_data_1= 00000000000000000000000000000110, read_data_2= 11111111111111111111111111111001,
# write_data= 00000000000000000000000000000110,
# ALUop= 110, ALUcontrol= 110, ALUresult= 00000000000000000000000000000110, ALUzero= 0, resultExtended= 00000000000000000000000000000110, mux_result= 00000000000000000000000000000110
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

17.andi $2 = $3 & 7

```
# time= 760, clock= 1, PC= 00000000000000000000000001010100, instruction= 0010011010000111,
# opcode= 0010, rs= 011, rt= 010, rd= 000, funct= 111, imm6= 000111
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000110,
# write_data= 00000000000000000000000000000000,
# ALUop= 110, ALUcontrol= 110, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000111, mux_result= 00000000000000000000000000000111
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 780, clock= 0, PC= 00000000000000000000000001011000, instruction= 0010011010000111,
# opcode= 0010, rs= 011, rt= 010, rd= 000, funct= 111, imm6= 000111
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000110,
# write_data= 00000000000000000000000000000000,
# ALUop= 110, ALUcontrol= 110, ALUresult= 00000000000000000000000000000000, ALUzero= 1, resultExtended= 00000000000000000000000000000111, mux_result= 00000000000000000000000000000111
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

18.ori $3 = $0 || 1

```
# time= 800, clock= 1, PC= 00000000000000000000000001011000, instruction= 0011000011000001,
# opcode= 0011, rs= 000, rt= 011, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000001,
# ALUop= 111, ALUcontrol= 111, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 820, clock= 0, PC= 00000000000000000000000001011100, instruction= 0011000011000001,
# opcode= 0011, rs= 000, rt= 011, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000000001,
# ALUop= 111, ALUcontrol= 111, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

19.ori $4 = $2 || 5

```
# time= 840, clock= 1, PC= 00000000000000000000000001011100, instruction= 0011010100000101,
# opcode= 0011, rs= 010, rt= 100, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111111,
# write_data= 00000000000000000000000000000101,
# ALUop= 111, ALUcontrol= 111, ALUresult= 00000000000000000000000000000101, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 860, clock= 0, PC= 00000000000000000000000001100000, instruction= 0011010100000101,
# opcode= 0011, rs= 010, rt= 100, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111111,
# write_data= 00000000000000000000000000000101,
# ALUop= 111, ALUcontrol= 111, ALUresult= 00000000000000000000000000000101, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

20.nori $5 = $0 ~| | 1

```
# time= 880, clock= 1, PC= 00000000000000000000000001100000, instruction= 0100000101000001,
# opcode= 0100, rs= 000, rt= 101, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000110,
# write_data= 11111111111111111111111111111110,
# ALUop= 101, ALUcontrol= 101, ALUresult= 11111111111111111111111111111110, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 900, clock= 0, PC= 00000000000000000000000001100100, instruction= 0100000101000001,
# opcode= 0100, rs= 000, rt= 101, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000110,
# write_data= 11111111111111111111111111111110,
# ALUop= 101, ALUcontrol= 101, ALUresult= 11111111111111111111111111111110, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

21.nori $6 = $2 ~| | 5

```
# time= 920, clock= 1, PC= 00000000000000000000000001100100, instruction= 0100010110000101,
# opcode= 0100, rs= 010, rt= 110, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111110,
# write_data= 11111111111111111111111111111010,
# ALUop= 101, ALUcontrol= 101, ALUresult= 11111111111111111111111111111010, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
#
# time= 940, clock= 0, PC= 00000000000000000000000001101000, instruction= 0100010110000101,
# opcode= 0100, rs= 010, rt= 110, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111110,
# write_data= 11111111111111111111111111111010,
# ALUop= 101, ALUcontrol= 101, ALUresult= 11111111111111111111111111111010, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 1
#
```

22.slti $7 = $2 <? 5

```
# time= 960, clock= 1, PC= 00000000000000000000000001101000, instruction= 0111010111000101,
# opcode= 0111, rs= 010, rt= 111, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000001111,
# write_data= 00000000000000000000000000000001,
# ALUop= 100, ALUcontrol= 100, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 980, clock= 0, PC= 00000000000000000000000001101100, instruction= 0111010111000101,
# opcode= 0111, rs= 010, rt= 111, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000001111,
# write_data= 00000000000000000000000000000001,
# ALUop= 100, ALUcontrol= 100, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

23.slti $1 = $3 <? 9

```
# time= 1000, clock= 1, PC= 00000000000000000000000001101100, instruction= 0111011001001001,
# opcode= 0111, rs= 011, rt= 001, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000001, read_data_2= 00000000000000000000000000000110,
# write_data= 00000000000000000000000000000001,
# ALUop= 100, ALUcontrol= 100, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000001001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 1020, clock= 0, PC= 00000000000000000000000001110000, instruction= 0111011001001001,
# opcode= 0111, rs= 011, rt= 001, rd= 001, funct= 001, imm6= 001001
# read_data_1= 00000000000000000000000000000001, read_data_2= 00000000000000000000000000000110,
# write_data= 00000000000000000000000000000001,
# ALUop= 100, ALUcontrol= 100, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000001001, mux_result= 00000000000000000000000000001001
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 1,MemRead= 0, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

24.lw $2 = M[$0 + 1]

```
# time= 1040, clock= 1, PC= 00000000000000000000000001110000, instruction= 1000000010000001,
# opcode= 1000, rs= 000, rt= 010, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000010100,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 1, RegWrite= 1,MemRead= 1, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 1060, clock= 0, PC= 00000000000000000000000001110100, instruction= 1000000010000001,
# opcode= 1000, rs= 000, rt= 010, rd= 000, funct= 001, imm6= 000001
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000000,
# write_data= 00000000000000000000000000010100,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000001, ALUzero= 0, resultExtended= 00000000000000000000000000000001, mux_result= 00000000000000000000000000000001
# RegDst= 0, ALUsrc= 1, MemtoReg= 1, RegWrite= 1,MemRead= 1, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

25. lw $3 = M[$2 + 5]

```
#
# time= 1080, clock= 1, PC= 00000000000000000000000001110100, instruction= 1000010011000101,
# opcode= 1000, rs= 010, rt= 011, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000010100, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000011001,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000011001, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 1, RegWrite= 1,MemRead= 1, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 1100, clock= 0, PC= 00000000000000000000000001111000, instruction= 1000010011000101,
# opcode= 1000, rs= 010, rt= 011, rd= 000, funct= 101, imm6= 000101
# read_data_1= 00000000000000000000000000010100, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000011001,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000011001, ALUzero= 0, resultExtended= 00000000000000000000000000000101, mux_result= 00000000000000000000000000000101
# RegDst= 0, ALUsrc= 1, MemtoReg= 1, RegWrite= 1,MemRead= 1, MemWrite= 0, Branch= 0, Branch_not= 0, extend_type= 0
#
```

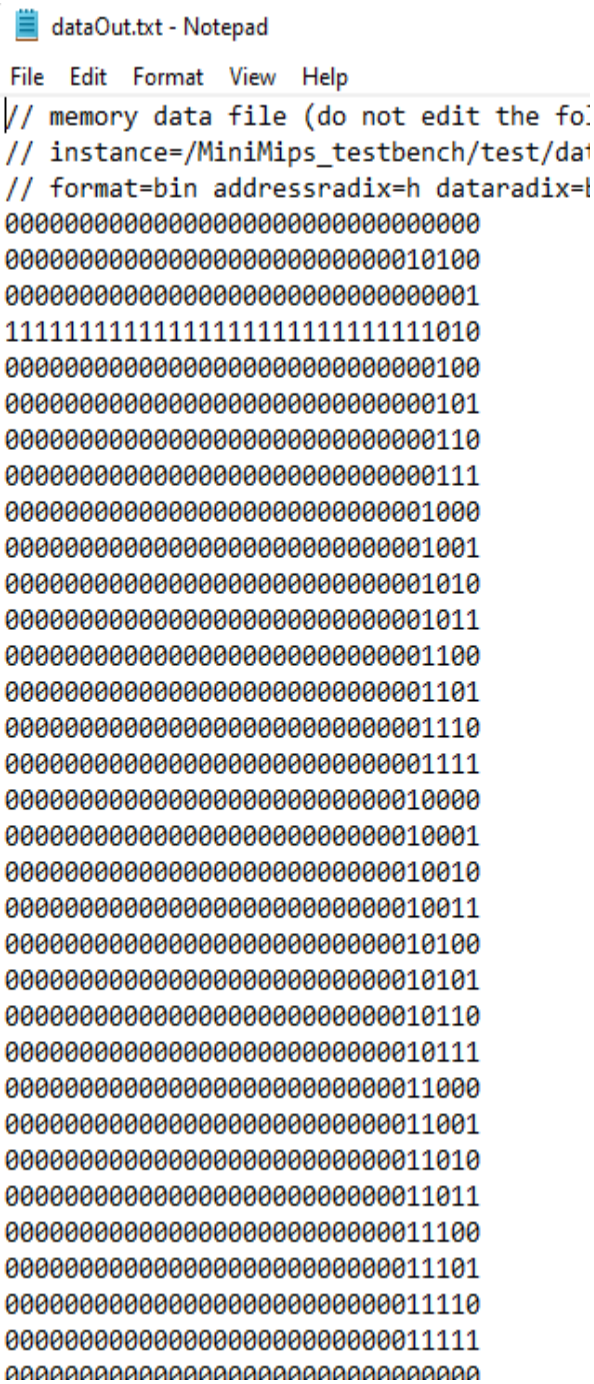
26.sw Mem[$0+2] = $7

```
#
# time= 1120, clock= 1, PC= 00000000000000000000000001111000, instruction= 1001000111000010,
# opcode= 1001, rs= 000, rt= 111, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000000010,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000010, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000010
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 1, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 1140, clock= 0, PC= 00000000000000000000000001111100, instruction= 1001000111000010,
# opcode= 1001, rs= 000, rt= 111, rd= 000, funct= 010, imm6= 000010
# read_data_1= 00000000000000000000000000000000, read_data_2= 00000000000000000000000000000001,
# write_data= 00000000000000000000000000000010,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000010, ALUzero= 0, resultExtended= 00000000000000000000000000000010, mux_result= 00000000000000000000000000000010
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 1, Branch= 0, Branch_not= 0, extend_type= 0
#
```

27.sw Mem[$0+3] = $6

```
# time= 1160, clock= 1, PC= 00000000000000000000000001111100, instruction= 1001000110000011,
# opcode= 1001, rs= 000, rt= 110, rd= 000, funct= 011, imm6= 000011
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111010,
# write_data= 00000000000000000000000000000011,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000011, ALUzero= 0, resultExtended= 00000000000000000000000000000011, mux_result= 00000000000000000000000000000011
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 1, Branch= 0, Branch_not= 0, extend_type= 0
#
#
# time= 1180, clock= 0, PC= 00000000000000000000000010000000, instruction= 1001000110000011,
# opcode= 1001, rs= 000, rt= 110, rd= 000, funct= 011, imm6= 000011
# read_data_1= 00000000000000000000000000000000, read_data_2= 11111111111111111111111111111010,
# write_data= 00000000000000000000000000000011,
# ALUop= 000, ALUcontrol= 000, ALUresult= 00000000000000000000000000000011, ALUzero= 0, resultExtended= 00000000000000000000000000000011, mux_result= 00000000000000000000000000000011
# RegDst= 0, ALUsrc= 1, MemtoReg= 0, RegWrite= 0,MemRead= 0, MemWrite= 1, Branch= 0, Branch_not= 0, extend_type= 0
#
```

Final Data Output:



Final Register Output: