CSE331
HOMEWORK #4

BİLAL YALÇINKAYA
1901042643

Control Unit: **This module runs according to jpeg i attached which is karnaugh maps of opCodes. But in jpeg it goes op3op2op1op0 while in code it goes op0op1op2op3 so i hope its not gonna make a confusion.**

Deconstruct Instruction: **This module takes 16 bit instruction and outputs its opcode,rs,rt,rd,func,imm.**

Instruction Memory: **At start it reads all instructions from dat file and records it to reg array and at every negedge of clock reads next instruction from reg and outputs it.**

IsEqual: **Takes 2 numbers, Xnors every pair and 'and' them to find out is numbers are equal.**

Memories: **Holds 64 of 32bit numbers at reg. At start instantiates variables by assign all 0. if MemRead is positive gets data from wanted memory. If clock is positive and MemWrite is 1 then writes data to wanted Address.**

PC Adder: **This is for incrementing Program Counter 1 or if BEQ or BNE comes increment according to Immidiate number. If branch and isequal is both 1 then output PC + 1 + extendedImm else PC + 1.**

Registers: **Holds 8 of 32bit numbers at reg. instantiate variables respectively 0,1,2,3,4,5,6,7. Reads data at every posedge or negedge to R[rs] and R[rt] to outputs. If clock is positive, writeRegister is not 0 because R[0] is zero register(can't be changed) and RegWrite is 1 then write data to wanted register.**

Sign Extender: **Extends 6 bit Immidiate to 32 bit number signed number.**

ALU: **Takes Rs and Rt(or imm) and calculates all possible results of opCode(or Func) and mux all of them according to opCode for output.**

CONTROL UNIT

Opcode	ALUSRC	Branch	Mem Read	Mem write	MemtoReg	Reg Wrt	RegDes
0000	0	0	0	0	0	1	1
0001	1	0	0	0	0	1	0
0010	1	0	0	0	0	1	0
0011	1	0	0	0	0	1	0
0100	1	0	0	0	0	1	0
0101	0	1	0	0	x	0	x
0110	0	1	0	0	x	0	x
0111	1	0	0	0	0	1	0
1000	1	0	1	0	1	1	0
1001	1	0	0	1	x	0	x

Op_1Op_0 \ Op_3Op_2	00	01	11	10
00	0	1	x	1
01	1	0	x	1
11	1	1	x	x
10	1	0	x	x

$$\Rightarrow ALUSrc = \bar{3}\bar{2}0 + \bar{3}\bar{2}1 + \bar{3}10 + 2\bar{1}\bar{0} + 3\bar{2}\bar{1}$$

$$Branch = \bar{3}2\bar{1}0 + \bar{3}21\bar{0}$$

$$Memread = 32\bar{1}\bar{0}$$

$$Mem\ write = 32\bar{1}0$$

Op_1Op_0 \ Op_3Op_2	00	01	11	10
00	1	1	x	1
01	1	0	x	0
11	1	1	x	x
10	1	0	x	x

$$\Rightarrow RegWrite = \bar{3}\bar{2} + \bar{1}\bar{0} + \bar{3}10$$

$$MemtoReg = 3\bar{2}\bar{1}\bar{0}$$

$$RegDes = \bar{3}2\bar{1}\bar{0}$$

TESTBENCHES

Control Unit

```
VSIM 17> run
# opCode = 0000, RegDst = 1, ALUSrc = 0, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 0001, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 0010, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 0011, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 0100, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 0101, RegDst = 0, ALUSrc = 0, MemtoReg = 0, RegWrite = 0, MemRead = 0, MemWrite = 0, Branch = 1
# opCode = 0110, RegDst = 0, ALUSrc = 0, MemtoReg = 0, RegWrite = 0, MemRead = 0, MemWrite = 0, Branch = 1
# opCode = 0111, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 1, MemRead = 0, MemWrite = 0, Branch = 0
# opCode = 1000, RegDst = 0, ALUSrc = 1, MemtoReg = 1, RegWrite = 1, MemRead = 1, MemWrite = 0, Branch = 0
# opCode = 1001, RegDst = 0, ALUSrc = 1, MemtoReg = 0, RegWrite = 0, MemRead = 0, MemWrite = 1, Branch = 0
```

Memories

```
VSIM 77> run -all
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000000000, ReadData = 00000000000000000000000000000000, WriteData = 00000000000000000000000000000000
# MemWrite = 1, MemRead = 0, Address = 00000000000000000000000000001111, ReadData = 00000000000000000000000000000000, WriteData = 00000000111100000111100111011001
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000001111, ReadData = 00000000111100000111100111011001, WriteData = 00000000000000000000000000000000
# MemWrite = 1, MemRead = 0, Address = 00000000000000000000000000000011, ReadData = 00000000111100000111100111011001, WriteData = 00000000000000010001001001011001
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000000011, ReadData = 00000000000000010001001001011001, WriteData = 00000000000000000000000000000000
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000000100, ReadData = 00000000000000000000000000000000, WriteData = 00000000000000000000000000000000
# MemWrite = 1, MemRead = 0, Address = 00000000000000000000000000001101, ReadData = 00000000000000000000000000000000, WriteData = 00000000000000001110000010010101
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000001101, ReadData = 00000000000000001110000010010101, WriteData = 00000000000000000000000000000000
# MemWrite = 0, MemRead = 1, Address = 00000000000000000000000000000111, ReadData = 00000000000000000000000000000000, WriteData = 00000000000000000000000000000000
```

Registers

```
VSIM 85> run -all
# clock = 0, RegWrite = 0, writeData = 00000000000000000000010010000100, writeRegister = 000, R[000] = 00000000000000000000000000000000, R[111] = 00000000000000000000000000000111
# clock = 1, RegWrite = 0, writeData = 00000000000000000000010010000100, writeRegister = 000, R[000] = 00000000000000000000000000000000, R[111] = 00000000000000000000000000000111
# clock = 0, RegWrite = 1, writeData = 00000000000000000000010010000100, writeRegister = 001, R[001] = 00000000000000000000000000000001, R[000] = 00000000000000000000000000000000
# clock = 1, RegWrite = 1, writeData = 00000000000000000000010010000100, writeRegister = 001, R[001] = 00000000000000000000010010000100, R[000] = 00000000000000000000000000000000
# clock = 0, RegWrite = 0, writeData = 00000000000000000000100011001011, writeRegister = 010, R[010] = 00000000000000000000000000000010, R[001] = 00000000000000000000010010000100
# clock = 1, RegWrite = 0, writeData = 00000000000000000000100011001011, writeRegister = 010, R[010] = 00000000000000000000000000000010, R[001] = 00000000000000000000010010000100
# clock = 0, RegWrite = 1, writeData = 00000000000000000000100011001011, writeRegister = 011, R[011] = 00000000000000000000000000000011, R[010] = 00000000000000000000000000000010
# clock = 1, RegWrite = 1, writeData = 00000000000000000000100011001011, writeRegister = 011, R[011] = 00000000000000000000100011001011, R[010] = 00000000000000000000000000000010
# clock = 0, RegWrite = 0, writeData = 00000000000000000001100110001110, writeRegister = 100, R[100] = 00000000000000000000000000000100, R[011] = 00000000000000000000100011001011
# clock = 1, RegWrite = 0, writeData = 00000000000000000001100110001110, writeRegister = 100, R[100] = 00000000000000000000000000000100, R[011] = 00000000000000000000100011001011
# clock = 0, RegWrite = 0, writeData = 00000000000000000001100110001110, writeRegister = 100, R[100] = 00000000000000000000000000000100, R[011] = 00000000000000000000100011001011
```

MAIN TESTBENCH

Transcript

```
#           AT START
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000101
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
run
# Rs = 101 = , Rt = 111, Rd = 111, Func = 000 | AND 101 with 111 = 101 | write it to register 111
# ALUResult: 00000000000000000000000000000001,Instruction: 0000011101101000,opcode: 0000, func: 000, rs: 011, rt: 101, rd: 101, imm: 101000
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000101
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000001
run
# Rs = 011 = , Rt = 101, Rd = 101, Func = 000 | AND 011 with 101 = 001 | write it to register 101
# ALUResult: 00000000000000000000000000000010,Instruction: 0000001001001001,opcode: 0000, func: 001, rs: 001, rt: 001, rd: 001, imm: 001001
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000010
run
# Rs = 001 = , Rt = 001, Rd = 001, Func = 001 | ADD 001 with 001 = 010 | write it to register 001
# ALUResult: 00000000000000000000000000000100,Instruction: 0000001010001001,opcode: 0000, func: 001, rs: 001, rt: 010, rd: 001, imm: 001001
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000010
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000011
```

```
# Rs = 001, Rt = 010, Rd = 001, Func = 001 | ADD 010 with 010 = 100 | write it to register 001
# ALUResult: 00000000000000000000000000000010,Instruction: 0000001010001010,opcode: 0000, func: 010, rs: 001, rt: 010, rd: 001, imm: 001010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000100
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000100
run
# Rs = 001, Rt = 010, Rd = 001, Func = 010 | SUB 100 with 010 = 010 | write it to register 001
# ALUResult: 00000000000000000000000000000000,Instruction: 0000001001001010,opcode: 0000, func: 010, rs: 001, rt: 001, rd: 001, imm: 001010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000010
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000101
run
# Rs = 011, Rt = 101, Rd = 101, Func = 010 | SUB 011 with 101 = 001 | write it to register 101
# ALUResult: 00000000000000000000000000000101,Instruction: 0000011110011011,opcode: 0000, func: 011, rs: 011, rt: 110, rd: 011, imm: 011011
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000011
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000110
# Rs = 011, Rt = 110, Rd = 011, Func = 011 | XOR 011 with 110 = 101 | write it to register 011
# ALUResult: 00000000000000000000000000000011,Instruction: 0000011110011011,opcode: 0000, func: 011, rs: 011, rt: 110, rd: 011, imm: 011011
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000100
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0000111
```

```
# Rs = 100, Rt = 010, Rd = 100, Func = 011 | XOR 100 with 010 = 110 | write it to register 100
# ALUResult: 00000000000000000000000000000100,Instruction: 0000100010100011,opcode: 0000, func: 011, rs: 100, rt: 010, rd: 100, imm: 100011
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0001000
run
# Rs = 000, Rt = 001, Rd = 001, Func = 100 | NOR 0 with 0 = 1 | write it to register 001
# ALUResult: 00000000000000000000000000000000,Instruction: 0000000001001100,opcode: 0000, func: 100, rs: 000, rt: 001, rd: 001, imm: 001100
# 0: 00000000000000000000000000000000
# 1: 11111111111111111111111111111111
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0001001
run
# Rs = 001, Rt = 001, Rd = 001, Func = 100 | NOR 1 with 1 = 0 | write it to register 001
# ALUResult: 11111111111111111111111111111111,Instruction: 0000001001001100,opcode: 0000, func: 100, rs: 001, rt: 001, rd: 001, imm: 001100
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000101
# PC: 0001010
run
# Rs = 100, Rt = 111, Rd = 111, Func = 101 | OR 110 with 101 = 111 | write it to register 111
# ALUResult: 00000000000000000000000000000111,Instruction: 0000100111111101,opcode: 0000, func: 101, rs: 100, rt: 111, rd: 111, imm: 111101
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0001011
# Rs = 101, Rt = 110, Rd = 001, Func = 101 | OR 001 with 110 = 111 | write it to register 001
# ALUResult: 00000000000000000000000000000111,Instruction: 0000101110001101,opcode: 0000, func: 101, rs: 101, rt: 110, rd: 001, imm: 001101
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000111
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0001100
run
# Opcode = 0001, Rs = 111, Rt = 001, imm = 001111 | ADDI 111 with 1111 = 10110 | write it to register 001
# ALUResult: 00000000000000000000000000010110,Instruction: 0001111001001111,opcode: 0001, func: 111, rs: 111, rt: 001, rd: 001, imm: 001111
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 00000000000000000000000000000001
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0001101
run
# Opcode = 0001, Rs = 000, Rt = 101, imm = 101010 | ADDI 0 with 11111111111111111111111111101010 = 11111111111111111111111111101010 | write it to register 101
# ALUResult: 11111111111111111111111111101010,Instruction: 0001000101101010,opcode: 0001, func: 010, rs: 000, rt: 101, rd: 101, imm: 101010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000000110
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0001110
run
# Opcode = 0010, Rs = 001, Rt = 100, imm = 010101 | ANDI 10110 with 010101 = 10100 | write it to register 100
# ALUResult: 00000000000000000000000000010100,Instruction: 0010001100010101,opcode: 0010, func: 101, rs: 001, rt: 100, rd: 010, imm: 010101
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000101
# 4: 00000000000000000000000000010100
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0001111
```

```
# Opcode = 0010, Rs = 111, Rt = 011, imm = 101100 | ANDI 111 with 101100 = 000100 | write it to register 011
# ALUResult: 00000000000000000000000000000100,Instruction: 0010111011101100,opcode: 0010, func: 100, rs: 111, rt: 011, rd: 101, imm: 101100
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000010100
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010000
run
# Opcode = 0011, Rs = 101, Rt = 100, imm = 010101 | ORI 11111111111111111111111111101010 with 010101 = 1 | write it to register 100
# ALUResult: 11111111111111111111111111111111,Instruction: 0011101100010101,opcode: 0011, func: 101, rs: 101, rt: 100, rd: 010, imm: 010101
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000000010
# 3: 00000000000000000000000000000100
# 4: 11111111111111111111111111111111
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010001
run
# Opcode = 0011, Rs = 001, Rt = 010, imm = 000001 | ORI 10110 with 000001 = 010111 | write it to register 010
# ALUResult: 00000000000000000000000000010111,Instruction: 0011001010000001,opcode: 0011, func: 001, rs: 001, rt: 010, rd: 000, imm: 000001
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 11111111111111111111111111111111
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010010
run
# Opcode = 0100, Rs = 100, Rt = 100, imm = 111111 | NORI 1 with 1 = 0 | write it to register 100
# ALUResult: 00000000000000000000000000000000,Instruction: 0100100100111111,opcode: 0100, func: 111, rs: 100, rt: 100, rd: 111, imm: 111111
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 11111111111111111111111111101010
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010011
# Opcode = 0100, Rs = 101, Rt = 101, imm = 111111 | NORI 11111111111111111111111111101010 with 11111111111111111111111111111111 = 0 | write it to register 101
# ALUResult: 00000000000000000000000000000000,Instruction: 0100101101111111,opcode: 0100, func: 111, rs: 101, rt: 101, rd: 111, imm: 111111
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000010110
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010100
run
# Opcode = 0111, Rs = 000, Rt = 001, imm = 000000 | SLTI 000000 with 000000 = 0 | write it to register 001
# ALUResult: 00000000000000000000000000000000,Instruction: 0111000001000000,opcode: 0111, func: 000, rs: 000, rt: 001, rd: 000, imm: 000000
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000000
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010101
run
# Opcode = 0111, Rs = 001, Rt = 001, imm = 000010 | SLTI 000001 with 000010 = 1 | write it to register 001
# ALUResult: 00000000000000000000000000000001,Instruction: 0111001001000010,opcode: 0111, func: 010, rs: 001, rt: 001, rd: 000, imm: 000010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010110
run
# Opcode = 0101, Rs = 000, Rt = 111, imm = 000111 | BEQ 00000 with 000111 = 0 | don't jump to PC + 1 + 000111
# ALUResult: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,Instruction: 0101000111000111,opcode: 0101, func: 111, rs: 000, rt: 111, rd: 000, imm: 000111
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0010111
```

```
# Opcode = 0101, Rs = 001, Rt = 001, imm = 000001 | BEQ 011 with 101 = 001 | jump to PC + 1 + 000001
# ALUResult: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,Instruction: 0101001001000001,opcode: 0101, func: 001, rs: 001, rt: 001, rd: 000, imm: 000001
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011001
run
# Opcode = 0110, Rs = 000, Rt = 000, imm = 000111 | BNE 0 with 0 = 0 | don't jump to PC + 1 + 000111
# ALUResult: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,Instruction: 0110000000000111,opcode: 0110, func: 111, rs: 000, rt: 000, rd: 000, imm: 000111
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011010
run
# Opcode = 0110, Rs = 001, Rt = 100, imm = 000001 | BNE 001 with 100 = 1 | jump to PC + 1 + 000001
# ALUResult: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,Instruction: 0110001100000001,opcode: 0110, func: 001, rs: 001, rt: 100, rd: 000, imm: 000001
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011100
# Opcode = 1001, Rs = 001, Rt = 111, imm = 000000 | SW register 111 to memory 001 + 000000
# ALUResult: 00000000000000000000000000000001,Instruction: 1001001111000000,opcode: 1001, func: 000, rs: 001, rt: 111, rd: 000, imm: 000000
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011101
# M[1]: 00000000000000000000000000000111
# M[2]: 00000000000000000000000000000000
run
# Opcode = 1001, Rs = 000, Rt = 110, imm = 000010 | SW register 110 to memory 000 + 000010
# ALUResult: 00000000000000000000000000000010,Instruction: 1001000110000010,opcode: 1001, func: 010, rs: 000, rt: 110, rd: 000, imm: 000010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000001
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011110
# M[1]: 00000000000000000000000000000111
# M[2]: 00000000000000000000000000000110
run
# Opcode = 1000, Rs = 001, Rt = 001, imm = 000000 | LW memory 001 + 000000 to register 001
# ALUResult: 00000000000000000000000000000111,Instruction: 1000001001000000,opcode: 1000, func: 000, rs: 001, rt: 001, rd: 000, imm: 000000
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000111
# 2: 00000000000000000000000000010111
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0011111
# M[1]: 00000000000000000000000000000111
# M[2]: 00000000000000000000000000000110
run
# Opcode = 1000, Rs = 000, Rt = 010, imm = 000010 | LW memory 000 + 000010 to register 010
# ALUResult: 00000000000000000000000000000010,Instruction: 1000000010000010,opcode: 1000, func: 010, rs: 000, rt: 010, rd: 000, imm: 000010
# 0: 00000000000000000000000000000000
# 1: 00000000000000000000000000000111
# 2: 00000000000000000000000000000110
# 3: 00000000000000000000000000000100
# 4: 00000000000000000000000000000000
# 5: 00000000000000000000000000000000
# 6: 00000000000000000000000000000110
# 7: 00000000000000000000000000000111
# PC: 0100000
# M[1]: 00000000000000000000000000000111
# M[2]: 00000000000000000000000000000110
```