Computer Architecture - CS F342

A PROJECT REPORT
ON

Design, Implementation and Testing of a 16-bit Multi-Cycle RISC Processor

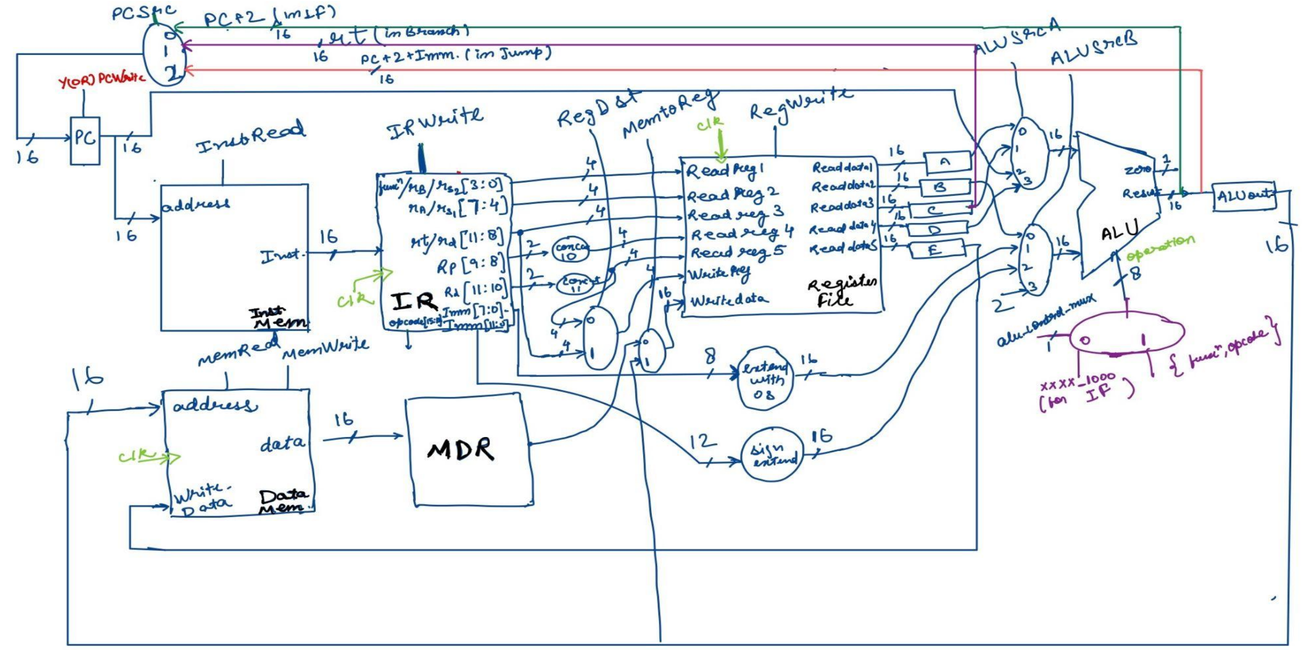
By

Mehul Kavdia (2018A8PS0860P)
Anubhav Srivastava (2018A8PS0030P)

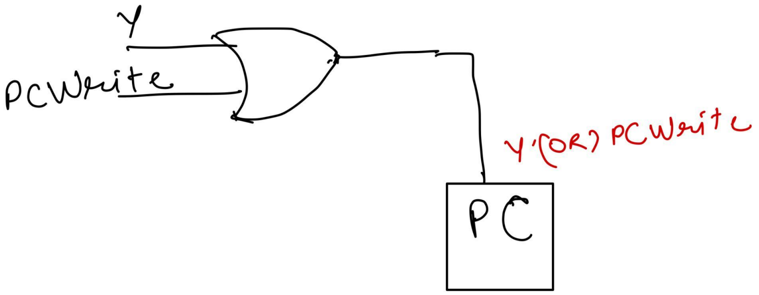
April, 2021

RISC processor design - 16 bits

Datapath:



For write signal of PC:



	PCWriteCond1	PCWriteCond2	zero	Y
B.E	1	0	1	1
	1	0	0	0
BNE	0	1	1	0
	0	1	0	1
(other cases) Not a branch inst.	x	x	x	0

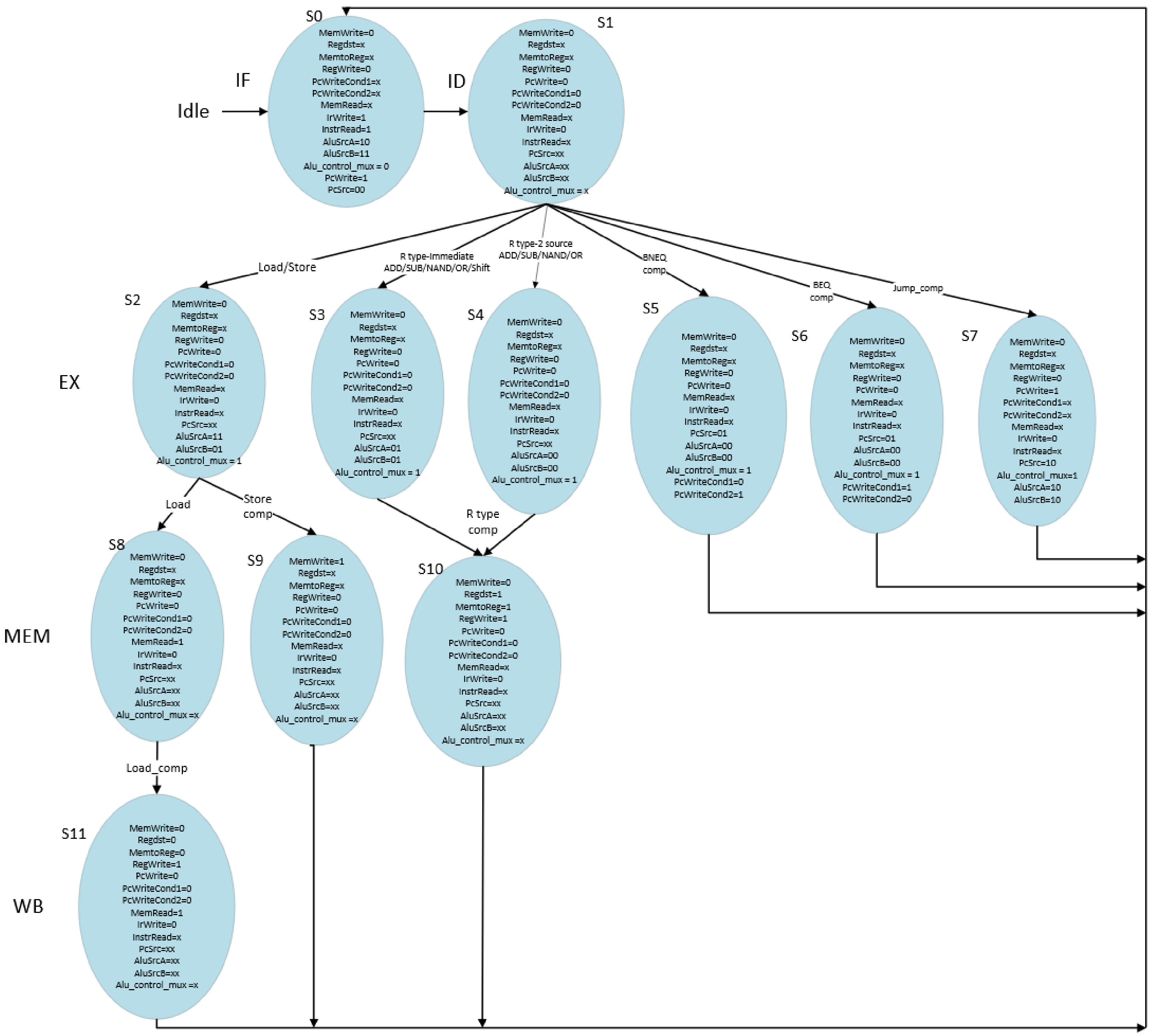
Where Y will depend on PCWriteCond1, PCWriteCond2(control signals) and zero (output from ALU), writing the truth table and k-map:

zero \ Cond1, Cond2	00	01	11	10
0	0	1	0	0
1	0	0	0	1

We get Y =(~pcwritecond1) && pcwritecond2 && (~zero) || pcwritecond1 && (~pcwritecond2) && zero

Control Unit:

Control Unit – State Diagram



Module prototype:

module multi_cycle(result_is_zero,address_invalid,clk,reset):
Top level module - contains various submodules

Reset Signal:
Before we give the clock pulse, we set the reset input of the processor. It brings the control path to idle state, also, pcout becomes 16'd0, to fetch the first instruction from instruction memory in the first clock cycle.

Address_invalid Flag:
All instructions should be placed at even addresses in instruction memory. This is because instructions in instruction memory are byte aligned, starting from address 16'd0. So, whenever, the instruction address is odd (i.e. the LSB of address is 1), the address_invalid flag becomes 1.

Result_is_zero Flag:
When ALU calculation results in zero, result_is_zero flag becomes 1.

Demo Program with instructions and their cycles

Instruction format
IF(State_number)
ID(State_number)
EX(State_number)
MEM(State_number)
WB (State_number)
Refer the Control unit - state diagram for values of control signals for each state

// All values in hex format
// Initially, in data memory(data.dat) 0080 at 4 and 0100 at 6.

//Register initial values
R0=xxxx, R1=xxxx, R2=xxxx, R3=xxxx, R4=xxxx, R5=xxxx, R6=xxxx, R7=xxxx, R8=xxxx, R9=xxxx, R10=xxxx, R11=xxxx, R12=xxxx, R13=xxxx, R14=xxxx, R15=xxxx

//0000
ADD R2 R0 R0 : 8200
IF(S0) - PC = PC +2 = 0002, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1000
EX(S4) - Addition is performed - Aluout = R0 + R0 = 0000
MEM(S10) - Writing in register R2
R2 = 0000

//0002
ADD R8 R0 R0 : 8800
IF(S0) - PC = PC +2 = 0004, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1000
EX(S4) - Addition is performed - Aluout = R0 + R0 = 0000
MEM(S10) - Writing in register R8
R8 = 0000

//0004
Load R12 R8 02 : 1002
IF(S0) - PC = PC +2 = 0006, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0001
EX(S2) - Load instruction address calculation - Aluout = R8 + 0004
MEM(S8) - Loads data into memory data register from data memory at the address location calculated in EX (0004)
WB(S11) - Write in register R12
R12 = 0080

//0006
Load R13 R8 03 : 1403
IF(S0) - PC = PC +2 = 0008, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0001
EX(S2) - Load instruction address calculation - Aluout = R8 + 0006
MEM(S8) - Loads data into memory data register from data memory at the address location calculated in EX (0006)
WB(S11) - Write in register R13
R13 = 0100

//0008
ADDi2 R2 14 : A214
IF(S0) - PC = PC +2 = 000A, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1010
EX(S3) - Addition is performed - Aluout = R2 + 0014 = 0014
MEM(S10) - Writing in register R2
R2 = 0014

//000A
ADD R1 R12 R13 : 81CD
IF(S0) - PC = PC +2 = 000C, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1000
EX(S4) - Addition is performed - Aluout = R12 + R13 = 0180
MEM(S10) - Writing in register R1
R1 = 0180

//000C
SUBi2 R1 80 : E180
IF(S0) - PC = PC +2 = 000E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1110
EX(S3) - Subtraction is performed - Aluout = R1 - 0080 = 0100
MEM(S10) - Writing in register R1
R1 = 0100

//000E
BEQ R2 R1 R0 : 4210
IF(S0) - PC = PC +2 = 0010, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0100
EX(S6) - Check if R1==R0, Aluout = R1 - R0 = FF00 != 0
Condition False - no branching

//0010
SRL R1 1 : 0112
IF(S0) - PC = PC +2 = 0012, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Logical Right Shift is performed - Aluout = 0080
MEM(S10) - Writing in register R1
R1 = 0080

//0012
JUMP FF8 : 3FF8
IF(S0) - PC = PC +2 = 0014, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0011
EX(S7) - Calculate jump address, PC = PC + 2 + immediate = 0014 + FFF8 = 000C
Jump to PC = 000C

//000C
SUBi2 R1 80 : E180
(explained above)
PC = 000E
R1 = 0000, result_is_zero = 1

//000E
BEQ R2 R1 R0 : 4210
(explained above)
PC = 0010
R1==R0, Condition true - branch to address stored in R2 (0014)
PC=0014

//0014

SRL R13 2 : 0D22
IF(S0) - PC = PC +2 = 0016, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Logical Right Shift is performed - Aluout = 0040
MEM(S10) - Writing in register R13
R13 = 0040

//0016
ADDi2 R8 8 : A808
IF(S0) - PC = PC +2 = 0018, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1010
EX(S3) - Addition is performed - Aluout = R8 + 0008 = 0008
MEM(S10) - Writing in register R8
R8 = 0008

//0018
BNEQ R2 R13 R8 : 52D8
IF(S0) - PC = PC +2 = 001A, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0101
EX(S5) - Check if R13!=R8, Aluout = R13 - R8 = FFC8 != 0
Condition True- Branch to address stored in R2 (0014)

//0014
SRL R13 2 : 0D22
(explained above)
PC = 0016
R13 = 0010

//0016
ADDi2 R8 8 : A808
(explained above)
PC = 0018
R18 = 0010

//0018
BNEQ R2 R13 R8 : 52D8
(explained above)
PC = 001A
R13 == R8, Condition False - no branching

//001A
OR R1 R12 R13 : F1CD
IF(S0) - PC = PC +2 = 001E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1111

EX(S4) - OR operation is performed - Aluout = R12 OR R13 = 0090
MEM(S10) - Writing in register R1
R1 = 0090

//001C
ADD R9 R0 R0 : 8900
IF(S0) - PC = PC +2 = 001E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1000
EX(S4) - Addition is performed - Aluout = R0 + R0 = 0000
MEM(S10) - Writing in register R9
R9 = 0000

//001E
ADDi2 R9 4 : A904
IF(S0) - PC = PC +2 = 0020, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1010
EX(S3) - Addition is performed - Aluout = R9 + 0004 = 0004
MEM(S10) - Writing in register R9
R9 = 0004

//0020
Store R13 R9 2 : 2502
IF(S0) - PC = PC +2 = 0022, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0010
EX(S2) - Store instruction address calculation - Aluout = R9 + 0004
MEM(S9) - Writes data into data memory at the address location calculated in EX (0008)

//0022
ADD R14 R1 R0 : 8E10
IF(S0) - PC = PC +2 = 0024, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1000
EX(S4) - Addition is performed - Aluout = R1 + R0 = 0090
MEM(S10) - Writing in register R1
R9 = 0090

//0024
Store R14 R9 3 : 2903
IF(S0) - PC = PC +2 = 0026, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0010
EX(S2) - Store instruction address calculation - Aluout = R9 + 0006
MEM(S9) - Writes data into data memory at the address location calculated in EX (0010)

//0026
NANDi R8 00H :**7800H**

IF(S0) - PC = PC +2 = 0028, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0111
EX(S3) - NAND operation is performed - Aluout = R8(NAND)00 = 1111
MEM(10) - Writing in register R8 = Aluout = 1111

//0028
NANDi R8 FFH :78FFH
IF(S0) - PC = PC +2 = 002A, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0111
EX(S3) - NAND operation is performed - Aluout = R8(NAND)FF = 0000
MEM(S10) - Writing in register R8
R8 = 1111

//002A
ADDi1 R8 AA :98AA
IF(S0) - PC = PC +2 = 002C, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1001
EX(S3) - Add immediate operation is performed - Aluout= R8 + (FF)AA = FFAA
MEM(S10) - Writing in register R8
R8 = FFAA

//002C
//**ADDi1 R8 AB** : **98AB**
IF(S0) - PC = PC +2 = 002E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1001
EX(S3) - Add immediate operation is performed - Aluout= R8+FFAB=FF55
MEM(S10) - Writing in register R8
R8 = FF55

//002E
NANDi R1 00H :**7100**
IF(S0) - PC = PC +2 = 0030, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0111
EX(S3) - NAND operation is performed - Aluout = R1(NAND)0000 = FFFF
MEM(S10) - Writing in register R1
R1 = FFFF

//0030
NANDi R1 FFH : 71FF
IF(S0) - PC = PC +2 = 0032, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0111
EX(S3) - NAND operation is performed - Aluout = R1(NAND)FFFF = 0000
MEM(S10) - Writing in register R1
R1 = 0000

//0032
ORi R2 FFH : 62FF
IF(S0) - PC = PC +2 = 0034, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0110
EX(S3) - OR operation is performed - Aluout = R2(OR)FFFF = FFFF
MEM(S10) - Writing in register R2

R2 = FFFF

//0034
NANDi R2 FFH: 72FF
IF(S0) - PC = PC +2 = 0036, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0111
EX(S3) - NAND operation is performed - Aluout = R2(NAND)FFFF = 0000
MEM(S10) - Writing in register R2
R2 = 0000

//0036
ADDi2 R1 64 : A164
IF(S0) - PC = PC +2 = 002E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1010
EX(S3) - Add immediate operation is performed - Aluout= R1+0064=0064
MEM(S10) - Writing in register R1
R1 = 0064

//0038
ADDi2 R2 8c : A28C
IF(S0) - PC = PC +2 = 003A, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1010
EX(S3) - Add immediate operation is performed - Aluout= R2+008C=008C
MEM(S10) - Writing in register R2 = Aluout = 008C

//003A
SUB R3 R1 R2 : C312
IF(S0) - PC = PC +2 = 003C, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1100
EX(S4) - Subtraction operation is performed - Aluout=0064-008C=FFD8
MEM(S10) - Writing in register R3 = Aluout = FFD8

//003C
SUB R3 R2 R1 : C321
IF(S0) - PC = PC +2 = 003E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1100
EX(S4) - Subtraction operation is performed - Aluout= -0064+008c=0028
MEM(S10) - Writing in register R3 = Aluout = 0028

//003E
SUBi1 R1 8C : D18C
IF(S0) - PC = PC +2 = 0040, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1101
EX(S3) - Subtraction immediate operation is performed - Aluout=0064-FF8c=00D8
MEM(S10) - Writing in register R1 = Aluout =00D8

//0040
SUBi1 R1 D8 : D1D8
IF(S0) - PC = PC +2 = 0042, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1101
EX(S3) - Subtraction immediate operation is performed - Aluout=00D8-FFD8=FF00

MEM(S10) - Writing in register R1 = Aluout =FF00

//0042
SUBi2 R3 27 : E327
IF(S0) - PC = PC +2 = 0044, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1110
EX(S3) - Subtraction immediate operation is performed - Aluout=0028-0027=0001
MEM(S10) - Writing in register R3
R3 = 0001

//0044
SLL R2 8 : 0281
IF(S0) - PC = PC +2 = 0046, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Shift logic left operation is performed - Aluout=008C <<8 = 8C00
MEM(S10) - Writing in register R2
R2 = 8C00

//0046
SLL R1 8 : 0181
IF(S0) - PC = PC +2 = 0048, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Shift logic left operation is performed - Aluout=FF00 <<8 = 0000
MEM(S10) - Writing in register R1
R1 = 0000

//0048
SAR R2 8 : 0283
IF(S0) - PC = PC +2 = 004A, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Shift arithmetic right operation is performed -Aluout=0283H/8c00H>>>8= FF8c
MEM(S10) - Writing in register R2
R2 = FF8C

//004A
SAR R3 Imm : 0383
IF(S0) - PC = PC +2 = 004C, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0000
EX(S3) - Shift arithmetic right operation is performed -Aluout=0383H //0001H>>>8 = 0000
MEM(S10) - Writing in register R3
R3 = 0000

//004C
NAND R1 R1 R2 : B121
IF(S0) - PC = PC +2 = 004E, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1011
EX(S4) - NAND operation is performed - Aluout = R1(NAND)R2 = FFFF
MEM(S10) - Writing in register R1
R1 = FFFF

//004E

OR R2 R2 R3 : F223
IF(S0) - PC = PC +2 = 0050, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 1111
EX(S4) - OR operation is performed - Aluout = R2(OR)R3 = FF8C
MEM(S10) - Writing in register R2
R2 = FF8C

//0050
Jump 053 : 3001
IF(S0) - PC = PC + 2 = 0052, read from instruction memory (instr.dat)
ID(S1) - Instruction decoded with opcode 0011
EX(S7) - Calculate jump address, PC = PC + 2 + immediate = 0052 + 0001 = 0053
Jump to PC = 0053
Illegal address (LSB bit of address must be 0), address_invalid = 1, user should stop the execution