ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO : 2

DUE DATE : 24.05.2021

GROUP NO : G24

GROUP MEMBERS:

150180056 : BEYZA OZAN

150180090 : MİHRİBAN NUR KOÇAK

150180099 : MEHMET YİĞİT BALIK

Contents

1 INTRODUCTION

In this project we designed a hardwired control unit for the processor system that we have implemented in the previous project. Hardwired design controls the system and sends appropriate signals to the system in order to achieve 15 different operation. There operations are BRA, LD, ST, MOV, AND, OR, NOT, AND, SUB, LSL, LSR, PUL, PSH, INC, DEC and BNE. In order to recognize and execute these operations properly, a Sequential Counter circuit, Fetch and Decode phases are implemented.
File organization:

- `HWcontrolUnitProject2.circ`
  - Component: `WholeSystem(Part4).circ`
    - Component: `8-bit_GeneralRegisterFile(Part2a).circ`
      - Component: `8-bit_Register(Part1a).circ`
    - Component: `8-bit_AdressRegisterFile(Part2b).circ`
      - Component: `8-bit_Register(Part1a).circ`
    - Component: `16-bit_InstructionRegister(Part1b).circ`
      - Component: `16-bit_Adder_Subtractor.circ`
      - Component: `16-bit_D-FlipFlop.circ`
    - Component: `ALUSystem(Part3).circ`
  - Component: `8-bit_Register(Part1a).circ`

Only circuit file that belongs to this project is `HWcontrolUnitProject2.circ`. All the other circuits are designed in the project 1.

2 PROJECT PARTS

2.1 CPU & Sequential Counter

In this part we used the Whole System from the Project-1 (Part4). For the timing conditions we created Sequential Counter.

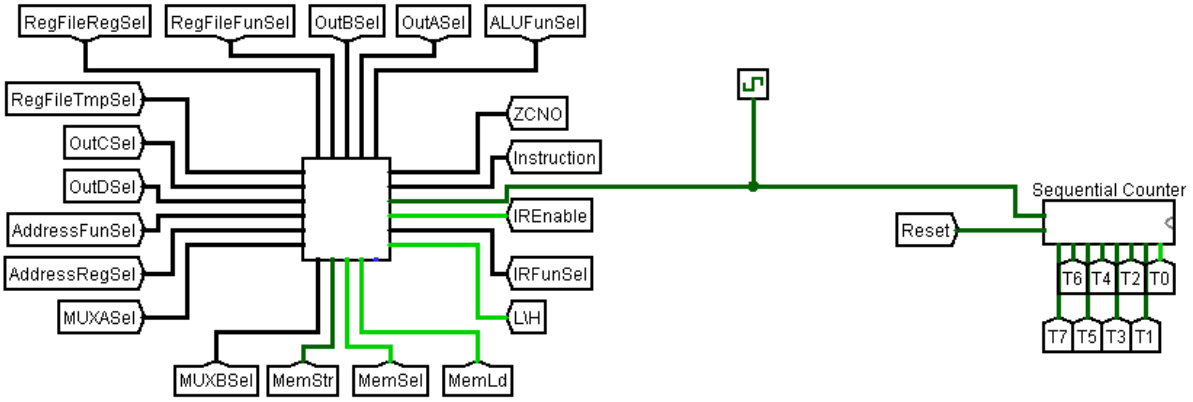

Figure 1: Abstract Circuits of CPU and Sequential Counter

16-Bit instruction and ZCNO outputs are obtained from the CPU to be used in the later parts such as Fetch and Decode. All other inputs are obtained from the Hard Wired Control Unit and send to CPU as can be seen in Figure 1.

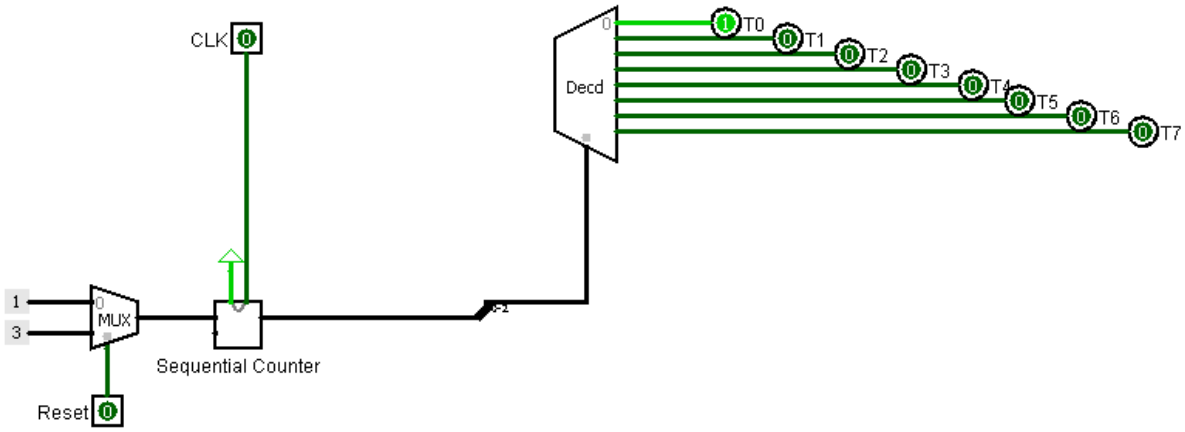

Figure 2: Logisim Circuit of Sequential Counter

Sequence Counter circuit is created to control each different execute operations according to their time occupation. For instance BRA(x00) operation occupies just 1 clock cycle, without fetch, and ADD operation can occupy up to 3 clock cycles. By creating the Sequence Counter we aimed to solve these timing issues.
In the Sequence Counter

- 1 to 2 Multiplexer is used to select functionality of the 8-bit register. If the reset signal is 0, then functionality of the register is always increment mode. If the reset signal 1, then the functionality becomes clear to set counter to T0.
- 8 bit Register is used to store how many clock cycles passed from the beginning of the fetch phase.

- 3 to 8 Decoder is used to represent the timing signals. If the instruction cycle just started, signal T0 becomes 1 and others become 0.

This design can count up to T7, but in this project the worst case of the clock cycle occupation is 5 (T4).

2.2 Fetch

In the fetch phase, the instructions in the memory are fetched according to little endian order because memory has 8-bit output and instructions are consist of 16-bit. It is obvious that the fetch phase should occupy first 2 clock cycles, T0 and T1. When the T0 and T1 clock signals are on, control unit must open the way of memory output to the instruction register and all other registers must be disabled to prevent any unwanted data changes.

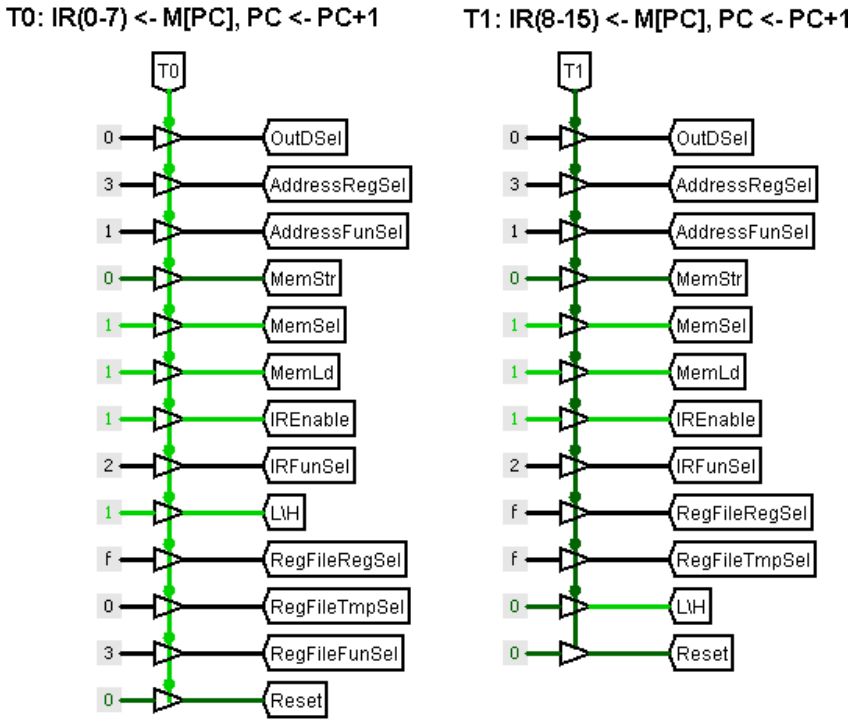

Figure 3: Control Unit of Fetch

According to the Figure-3:

T0: $IR(0-7) \leftarrow M[PC], PC <- PC+1$

- OutDSel = 00 (x0). Sends PC to address input of the memory.
- AddressRegSel = 011 (x3). Enables PC to increment.
- AddressFunSel = 01 (x1). Increments PC.
- MemStr = 0. Disables storing into memory.

- MemSel = 1. Enables memory chip select.
- MemLd = 1. Loads memory content to the memory output.
- IREnable = 1. Enables IR for data loading.
- IRFunSel = 10. Load data into instruction register.
- L-H = 1. Stores data into the least significant 8 bit.
- RegFileRegSel = 1111 (xf). Disables general purpose registers.
- RegFileTmpSel = 0000 (x0). Enables all the temporary registers to clear data in them.
- RegFileFunSel = 11 (x3). Clears data in the temporary registers.
- Reset = 0. Sends increment signal to the Sequential Counter and makes T1 clock signal 1.

T1: $IR(8-15) \leftarrow M[PC], PC < -PC + 1$

- OutDSel = 00 (x0). Sends PC to address input of the memory.
- AddressRegSel = 011 (x3). Enables PC to increment.
- AddressFunSel = 01 (x1). Increments PC.
- MemStr = 0. Disables storing into memory.
- MemSel = 1. Enables memory chip select.
- MemLd = 1. Loads memory content to the memory output.
- IREnable = 1. Enables IR for data loading.
- IRFunSel = 10. Load data into instruction register.
- L-H = 0. Stores data into the least significant 8 bit.
- RegFileRegSel = 1111 (xf). Disables general purpose registers.
- RegFileTmpSel = 1111 (xf). Enables all the temporary registers to clear data in them.
- Reset = 0. Sends increment signal to the Sequential Counter and makes T2 clock signal 1.

2.3 Decode

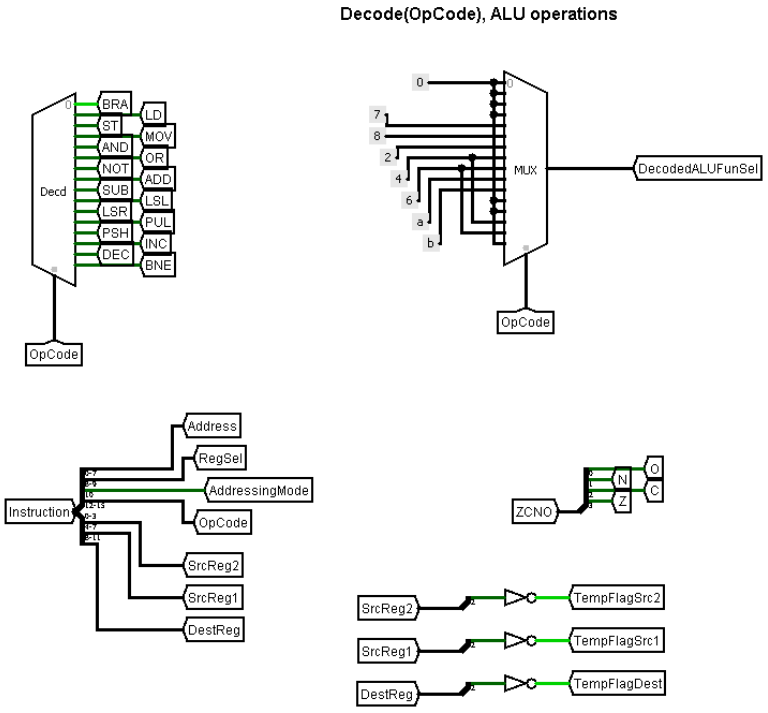

Figure 4: Control Unit of Decode

In the decode stage, our design selects which operation is going to be executed, which functionality of the ALU is going to be used. To determines mentioned the Instruction must be broken into pieces, as can be seen in the Figure-4.
Breaking Instruction into pieces:

- Address Referenced Instruction
  - Address = Instruction(0-7)
  - RegSel = Instruction(8-9)
  - AddressingMode = Instruction(10)
  - OpCode = Instruction(12-15)
- Non-Address Referenced Instruction
  - SrcReg2 = Instruction(0-3)
  - SrcReg1 = Instruction(4-7)
  - DestReg = Instruction(8-11)
  - OpCode = Instruction(12-15)

According to the OpCode, the operation, which is going to executed, and which functionality of the ALU is determined. If the operation is;

- BRA, LD, ST, MOV, PUL or PSH; DecodedALUFunSel set to 0 (Pass input A).
- AND, DecodedALUFunSel set to 7 (A & B).
- OR, DecodedALUFunSel set to 8 (A — B).
- NOT, DecodedALUFunSel set to 2 (not A).
- ADD, DecodedALUFunSel set to 4 (A + B).
- SUB, DecodedALUFunSel set to 6 (A - B).
- LSL, DecodedALUFunSel set to a (LSL(A)).
- LSR, DecodedALUFunSel set to b (LSR(A)).
- INC, DecodedALUFunSel set to 4 (A + B (B is going to be 1)).
- DEC, DecodedALUFunSel set to 6 (A - B (B is going to be 1)).

ZCNO output of the CPU is broken into pieces also to use Z in the BNE operation.
If an arithmetic operation is going to held with the Address Register File (ARF) registers we need to store its value in the temporary registers of the Register File. Inorder to determine it is ARF register or not, 3 flags are declared: TempFlagSrc2: 1 if SrcReg2 is a ARF register. TempFlagSrc1: 1 if SrcReg1 is a ARF register. TempFlagDest: 1 if DestReg is a ARF register.

2.4 BRA(x00), BNE(x0F)

BRA : PC ← Value
BNE: IF Z=0 THEN PC ← Value

After fetch and decode operations, if IR(15-12) (Opcode) is equal to 0x00, it corresponds to BRA operation and if IR(15-12) (Opcode) is equal to 0x0F, it corresponds to BNE operation. BRA and BNE operations are implemented together because these two operations are almost identical except for one small difference. The difference is that BNE operation is only performed when Z is equal to 0. BRA and BNE operations use the address reference format instructions which has already written to IR at T0 and T1 moments. BRA and BNE operations work only in Immediate addressing mode and according to that, the main purpose of them are transferring the value in the Address Field of instruction (IR(7-0)) to the Program Counter (PC).

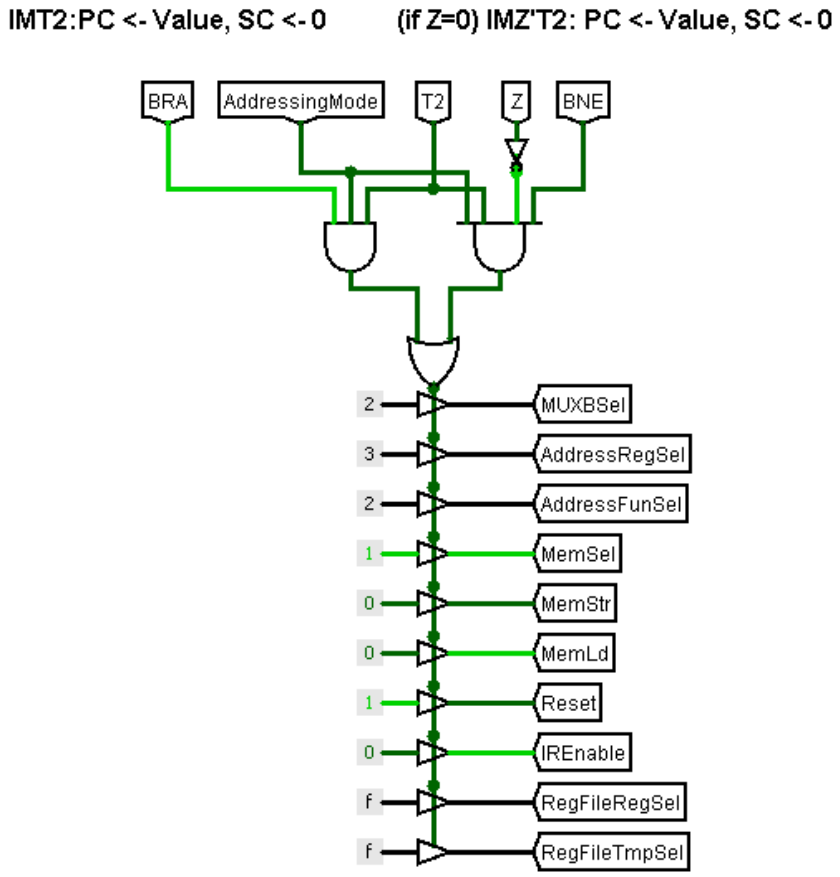

Figure 5:

If IR(10) (addressing mode) is equal to 1 and opcode is equal to x00, BRA operation is done with Immediate addressing mode or if IR(10) (addressing mode) is equal to 1 and Z is equal to 0 and opcode is equal to x0F, BNE operation is done with Immediate addressing mode. To perform one of these operations at T2 moment, we have given following inputs according to the Figure 5:

T2: $PC \leftarrow ADDRESSField, SC \leftarrow 0$

- MUXBSel = 10 (x2): Selects input going to the Adress Register File as Instruction Register's output's least significant 8- bit (IR(0-7)).
- AddressRegSel = 011 (x3): Enables only PC.
- AddressFunSel = 10 (x2): Load input data to enabled register.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables storing into memory.
- MemLd = 0: Disables loading memory content to the memory output.

- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111 (xF): Disables all the registers of Register File.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.

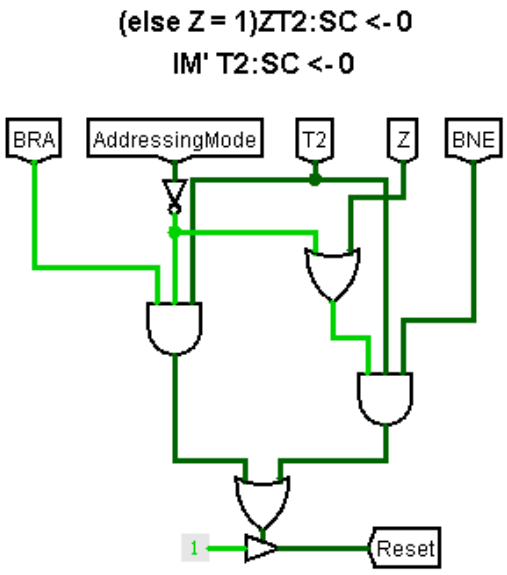

Figure 6:

If IR(10) (addressing mode) is equal to 0 and opcode is equal to x00, BRA operation isn't done or if IR(10) (addressing mode) is equal to 0 or and Z is equal to 1 and opcode is equal to x0F, BNE operation isn't done. So that only sequence counter is reseted for the purpose of turning back to T0 moment after this operation. To perform that at T2 moment, we have given following input according to the Figure 6:

T2: $SC \leftarrow 0$

- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.

2.5 PSH(0x0B)

In this operation, first we write the desired register value (Rx) to the memory location of SP (Stack Pointer) address register then we decrement the SP value by 1:

$$M[SP] \leftarrow Rx, \quad SP \leftarrow SP - 1$$

PSH operation is done in one step, so we used 1 time signal (T2).

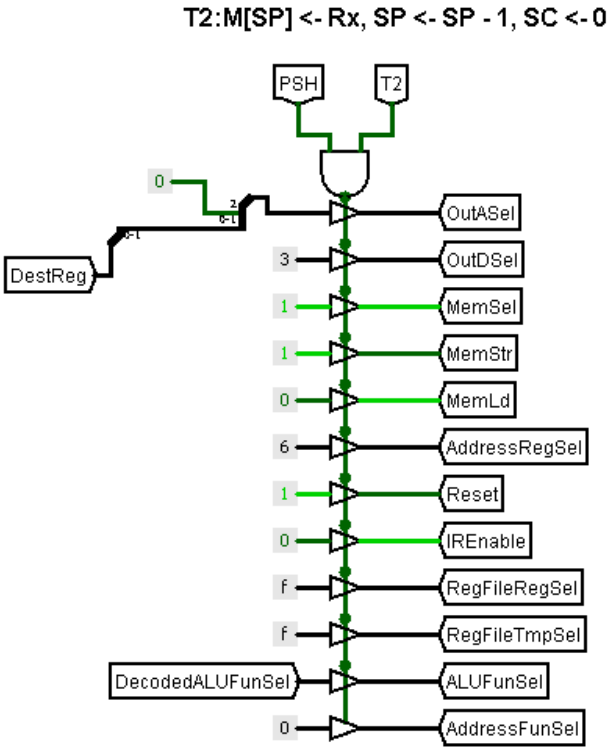

Figure 7: Circuit of PSH operation

First, we send the T2 and PSH tunnel to an AND gate so that the following operations run while both light up.

T2: $M[SP] \leftarrow Rx, \quad SP \leftarrow SP - 1$

- OutASel = Least significant two bits of DestReg and 0 for msb. Reads DESTREG value from Register File for ALU.
- OutDSel = 11(x3): Selects the SP register of Address Register File in order to send value to the memory as a address.
- MemSel = 1: Enables memory chip select.
- MemStr = 1: Enables writing into given address to the memory.

- MemLd = 0: Disables loading value from the memory.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- AddressRegSel = 110(x6): Enables only the SP register of Address Register File.
- AddressFunSel = 00(x0): Decreases the SP value by 1 in ARF.
- RegFileRegSel = 1111(xf): Disables all the registers R1, R2, R3, R4 of Register File.
- RegFileTmpSel = 1111(xf): Disables all the temporary registers of Register File
- ALUFunSel = DecodedALUFunSel = 0000: It provides to send input A directly to output.

2.6 PUL(0x0C)

In this operation, first we increment the SP value by 1 then, we read the value in the SP address from memory and write this value to the desired Rx register:

$$SP \leftarrow SP + 1, Rx \leftarrow M[SP]$$

PUL operation is done in two step, so we used 2 time signal (T2-T3).

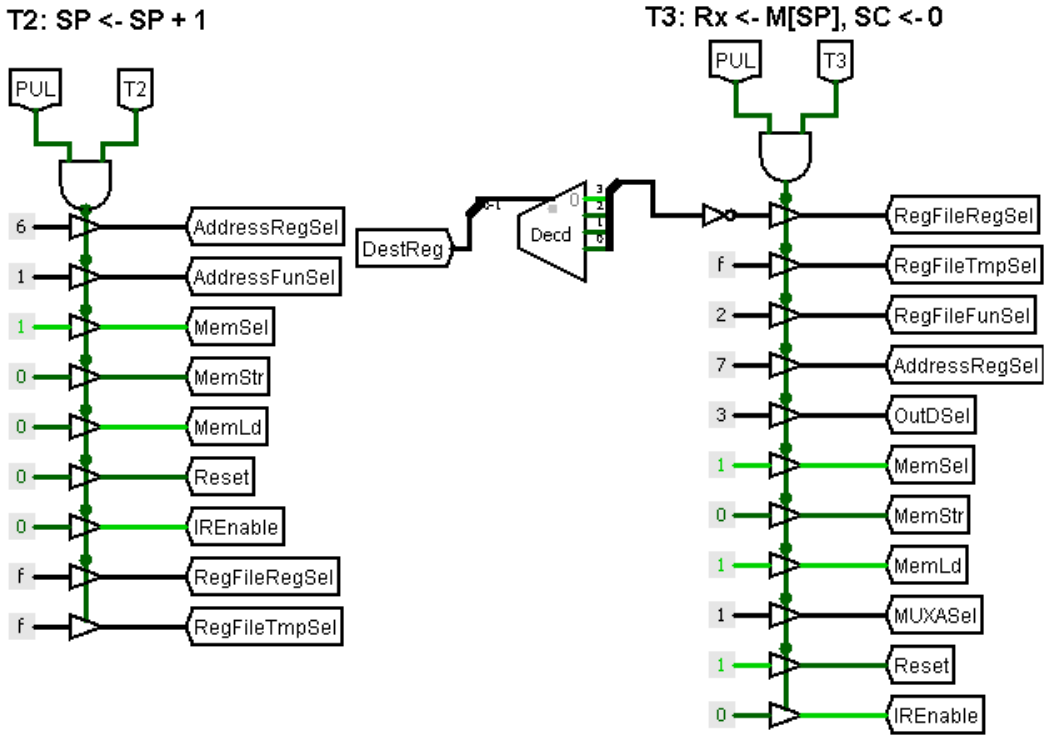

Figure 8: Circuit of PUL operation

First, we send the T2 and PUL tunnel to an AND gate so that the following operations run while both light up.

T2: $SP \leftarrow SP + 1$

- AddressRegSel = 110(x6): Enables only the SP register of Address Register File.
- AddressFunSel = 01(x1): Increases the SP value by 1 in ARF.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing value into the memory.
- MemLd = 0: Disables loading value from the memory.
- Reset = 0:
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all the registers R1, R2, R3, R4 of Register File.
- RegFileTmpSel = 1111(xf): Disables all the temporary registers of Register File

We send the T3 and PSH tunnel to an AND gate so that the following operations run while both light up.

T3: $Rx \leftarrow M[SP], \quad SC \leftarrow 0$

- RegFileRegSel = Selects one of the R1, R2 ,R3 or R4 registers of Register File as Rx. We use 2:4 decoder and NOT gate using 2-bit DestReg which is part of the instruction as a selection input for decoder.
- RegFileTmpSel = 1111(xf): Disables all the temporary registers of Register File
- RegFileFunSel = 10(x2): LOAD..
- AddressRegSel = 111(x7): Disables all registers of Address Register File.
- OutDSel = 11(x3): Selects the SP register of Address Register File in order to send value to the memory as a address.
- MUXASel = 01(x1): Selects the output of the memory for Register File.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into given address to the memory.
- MemLd = 1: Enables loading value at SP from the memory.

- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.

2.7 LD(0x01)

LD : Rx ← Value

After fetch and decode operations, if IR(15-12) (Opcode) is equal to 0x01, it corresponds to LD operation. LD operation uses the address reference format instructions which has already written to IR at T0 and T1 moments. The main purpose of LD operation is deciding the value according to addressing mode and then transfer this value to the desired register destination. LD operation can be performed with two different addressing modes. If IR(10) (addressing mode) is equal to 0, LD operation is done with Direct addressing mode.If IR(10) (addressing mode) is equal to 1, LD operation is done with Immediate addressing mode.

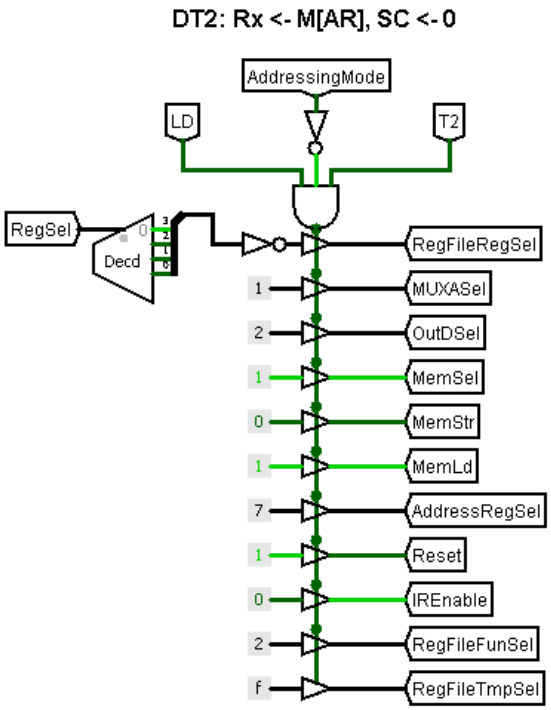

Figure 9: LD Operation with Direct Addressing Mode

If IR(10) (addressing mode) is equal to 0, LD operation is done with Direct addressing mode. Direct addressing mode corresponds to receiving the value located at the address of the memory according to the address value in the AR. To perform that at T2 moment, we have given following inputs according to the Figure 9:

T2: $Rx \leftarrow M[AR], SC \leftarrow 0$

- RegFileRegSel: Selects registers of Register File as Rx (destination register) using 2:4 decoder and not gate according to 2-bit RegSel part of the instruction. Decoder and not gate creates RegFileRegSel input as followings:
  - RegSel = 00: Data from decoder is 0001, using not gate it is converted to 1110 which makes RegFileRegSel = 1110. Now, RegFileRegSel input corresponds to selection of register R1.
  - RegSel = 01: Data from decoder is 0010, using not gate it is converted to 1101 which makes RegFileRegSel = 1101. Now, RegFileRegSel input corresponds to selection of register R2.
  - RegSel = 10: Data from decoder is 0100, using not gate it is converted to 1011 which makes RegFileRegSel = 1011. Now, RegFileRegSel input corresponds to selection of register R3.
  - RegSel = 11: Data from decoder is 1000, using not gate it is converted to 0111 which makes RegFileRegSel = 0111. Now, RegFileRegSel input corresponds to selection of register R4.
- MUXASel = 01 (x1): Selects input going to the Register File as output data coming from Memory.
- OutDSel = 02 (x2): Sends data in AR to address input of the memory.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables storing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel = 111 (x7): Disables all registers of Address Register File.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileFunSel = 10 (x2): Load data to enabled register.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.

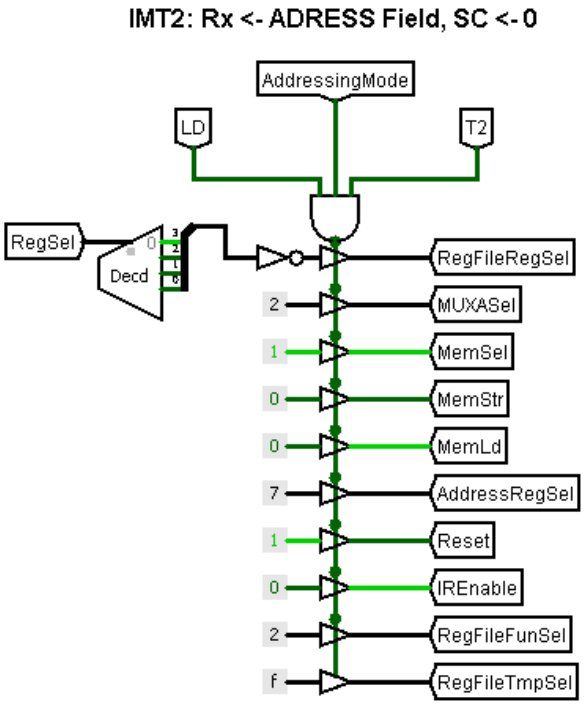

Figure 10: LD Operation with Immediate Addressing Mode

If IR(10) (addressing mode) is equal to 1, LD operation is done with Immediate addressing mode. Immediate addressing mode corresponds to getting value directly from the 8-bit ADDRESS field of the instruction. To perform that at T2 moment, we have given following inputs according to the Figure 10:

T2: $Rx \leftarrow ADDRESSField, SC \leftarrow 0$

- RegFileRegSel: Selects registers of Register File as Rx (destination register) using 2:4 decoder and not gate according to 2-bit RegSel part of the instruction. Decoder and not gate creates RegFileRegSel input as followings:
  - RegSel = 00: Data from decoder is 0001, using not gate it is converted to 1110 which makes RegFileRegSel = 1110. Now, RegFileRegSel input corresponds to selection of register R1.
  - RegSel = 01: Data from decoder is 0010, using not gate it is converted to 1101 which makes RegFileRegSel = 1101. Now, RegFileRegSel input corresponds to selection of register R2.
  - RegSel = 10: Data from decoder is 0100, using not gate it is converted to 1011 which makes RegFileRegSel = 1011. Now, RegFileRegSel input corresponds to selection of register R3.

  - RegSel = 11: Data from decoder is 1000, using not gate it is converted to 0111 which makes RegFileRegSel = 0111. Now, RegFileRegSel input corresponds to selection of register R4.

- MUXASel = 10 (x2): Selects input going to the Register File as Instruction Register's output's least significant 8- bit (IR(0-7)).
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables storing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel = 111 (x7): Disables all registers of Address Register File.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileFunSel = 10 (x2): Load data to enabled register.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.

2.8 ST(0x02)

ST : Value ← Rx

After fetch and decode operations, if IR(15-12) (Opcode) is equal to 0x02, it corresponds to ST operation. ST operation uses the address reference format instructions which has already written to IR at T0 and T1 moments. The main purpose of ST operation is transferring the data in the desired register source to the address of the memory according to the address value in the AR which corresponds to Direct addressing mode. ST operation can be performed with only Direct addressing mode.

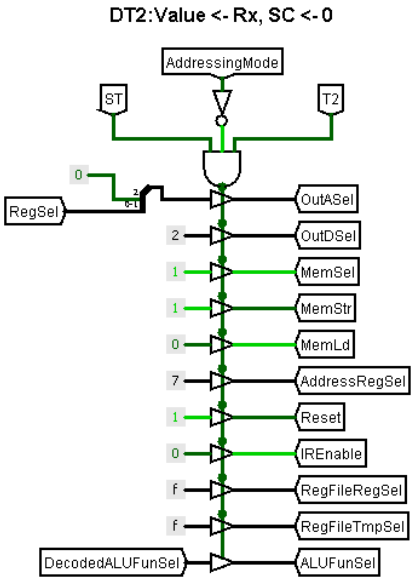

Figure 11: ST Operation with Direct Addressing Mode

If IR(10) (addressing mode) is equal to 0, ST operation is done with Direct addressing mode. Direct addressing mode corresponds to writing to the address of the memory according to the address value in the AR. To perform that at T2 moment, we have given following inputs according to the Figure 11:

T2: $M[AR] \leftarrow Rx, SC \leftarrow 0$

- OutASel: Sends data in selected register according to the A input of the ALU. The register selection is done according to RegSel as followings:
  - RegSel = 00: using splitter and constant value 0, OutASel becomes 000. OutASel = 000 corresponds to value in the register R1.
  - RegSel = 01: using splitter and constant value 0, OutASel becomes 001. OutASel = 001 corresponds to value in the register R2.
  - RegSel = 10: using splitter and constant value 0, OutASel becomes 010. OutASel = 010 corresponds to value in the register R3.
  - RegSel = 11: using splitter and constant value 0, OutASel becomes 011. OutASel = 011 corresponds to value in the register R4.
- OutDSel = 02 (x2): Sends data in AR to address input of the memory.
- MemSel = 1: Enables memory chip select.
- MemStr = 1: Enables writing into given address of the memory.
- MemLd = 0: Disables loading memory content to the memory output.

- AddressRegSel = 111 (x7): Disables all registers of Address Register File.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111 (xF): Disables all the registers of Register File.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- ALUFunSel = DecodedALUFunSel = 0000: For ST operation, DecodedALUFunSel = 0000 which corresponds to giving input A directly as output.

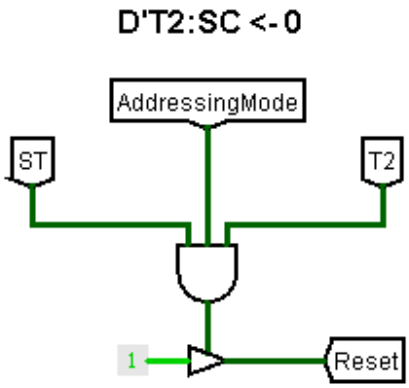

Figure 12: ST Operation with Not Direct Addressing Mode

If IR(10) (addressing mode) is equal to 1, ST operation isn't done so that only sequence counter is reset for the purpose of turning back to T0 moment after this operation. To perform that at T2 moment, we have given following input according to the Figure 12:

T2: $SC \leftarrow 0$

- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.

2.9 MOV(0x03)

After the fetch and decode operations, if Opcode is equal to 0x03, it corresponds to MOV operation. MOV operation moves data in the SRCREG to the DESTREG in just one clock cycle. In this operation, there are 4 possible data movement which are:

- From ARF to ARF
- From ARF to RF

- From RF to RF
- From RF to ARF

According to these 4 different movement possibilities, 4 different control units are designed.
Even though there are 4 different control units, common operation handled in one clock cycle.

T2: $DESTREG \leftarrow SRCREG1, SC \leftarrow 0$

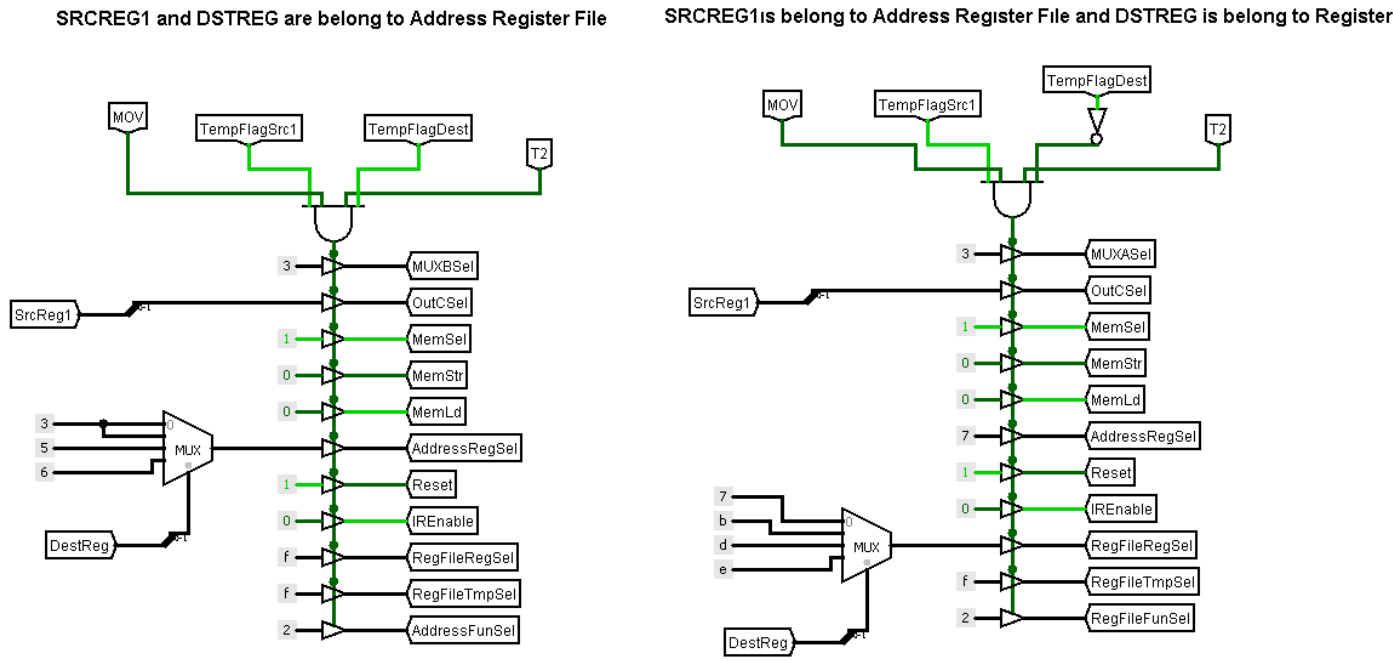

Figure 13: SRCREG1 belongs to ARF

If the SRCREG is belongs to ARF:

- If DESTREG belongs to ARF, then MUXBSel = 11 (x3). If DESTREG belongs to RF, then MUXASel = 11 (x3). Sends SRCREG data (from ARF) to ARF or RF, respectively.
- OutCSel = SrcReg(0-1). Selects C output of the ARF according to the least significant 2 bits of the SrcReg1.
- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- If DESTREG belongs to ARF, then it is set by the output of MUX (with the DestReg(0-1) selector input), it enables PC, AR or SP according to DestReg(0-

1). If DESTREG belongs to RF, then AddressRegSel = 111 (x7) disables all the registers of the ARF.

- Reset = 1. Sends reset signal to the Sequential Counter and makes T0 clock signal 1.
- IREnable = 0. Disables Instruction Register.
- If DESTREG belongs to ARF, then RegFileRegSel = 1111 (xf), disables all the registers of the Register File. If DESTREG belongs to RF then RegFileRegSel is set by the output of MUX (with the DestReg(0-1)). It enables R1, R2, R3 or R4 according to DestReg(0-1).
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- If DESTREG belongs to ARF, then AddressFunSel = 10 (x2). Loads data into register of ARF. If DESTREG belongs to RF, then RegFileFunSel = 10 (x2). Loads data into register of RF.

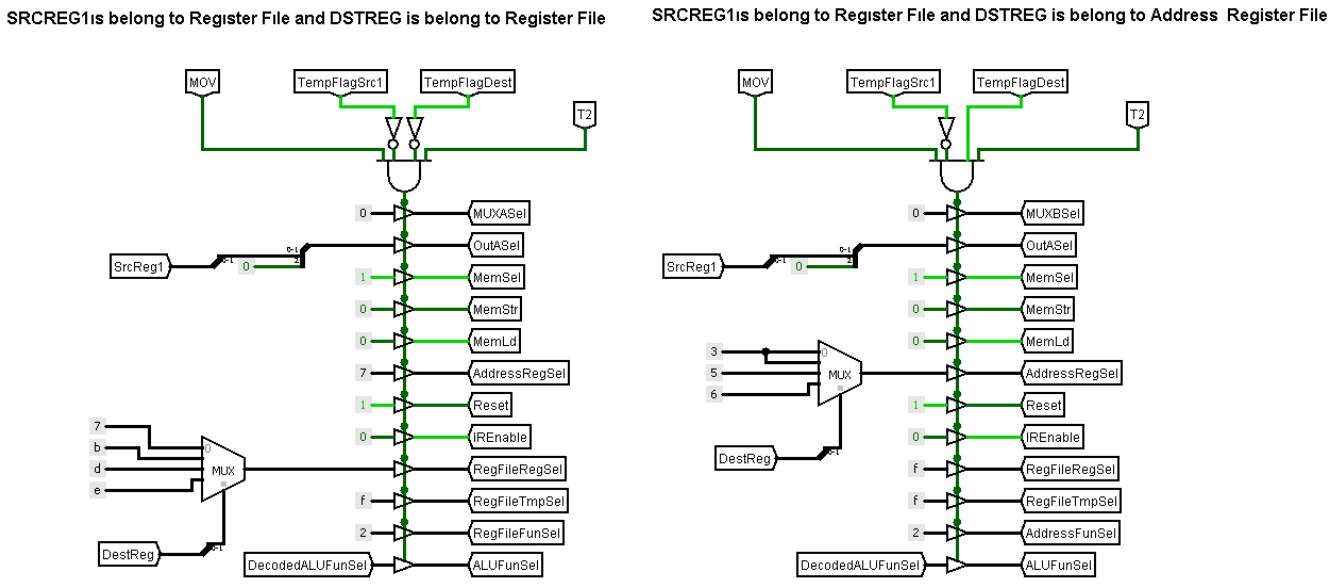

Figure 14:

If the SRCREG is belongs to RF:

- If DESTREG belongs to ARF, then MUXBSel = 11 (x3). If DESTREG belongs to RF, then MUXASel = 11 (x3). Sends SRCREG data (from ARF) to ARF or RF, respectively.
- OutASel = SrcReg(0-1). Selects A output of the RF according to the least significant 2 bits of the SrcReg1.

- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- If DESTREG belongs to ARF, then it is set by the output of MUX (with the DestReg(0-1) selector input), it enables PC, AR or SP according to DestReg(0-1). If DESTREG belongs to RF, then AddressRegSel = 111 (x7) disables all the registers of the ARF.
- Reset = 1. Sends reset signal to the Sequential Counter and makes T0 clock signal 1.
- IREnable = 0. Disables Instruction Register.
- If DESTREG belongs to ARF, then RegFileRegSel = 1111 (xf), disables all the registers of the Register File. If DESTREG belongs to RF then RegFileRegSel is set by the output of MUX (with the DestReg(0-1)). It enables R1, R2, R3 or R4 according to DestReg(0-1).
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- If DESTREG belongs to ARF, then AddressFunSel = 10 (x2). Loads data into register of ARF. If DESTREG belongs to RF, then RegFileFunSel = 10 (x2). Loads data into register of RF.
- ALUFunSel = DecodedALUFunSel. Sets ALUFunSel with right functionality (x0(PASS A) in this case).

2.10 NOT(x06),LSL(0x09),LSR(0x0A)

DESTREG ← NOT SRCREG1
DESTREG ← LSL SRCREG1
DESTREG ← LSR SRCREG1

After fetch and decode operations, if IR(15-12) (Opcode) is equal to 0x06, it corresponds to NOT operation, if IR(15-12) (Opcode) is equal to 0x09, it corresponds to LSL operation, if IR(15-12) (Opcode) is equal to 0x0A, it corresponds to LSR operation. All three operations use the instruction format without an address reference which has already written to IR at T0 and T1 moments. These three operations can be implemented together since the only difference between their implementation path is the operation performed by ALU. They need exactly same inputs except only the ALUFunSel which decides the operation to be performed by ALU. The main purpose of NOT operation is getting the value from the source register and perform NOT operation on this value

thanks to ALU then transfer this value to the desired destination register. The main purpose of LSL operation is getting the value from source register and perform LSL operation on this value thanks to ALU then transfer this value to the desired destination register. The main purpose of LSR operation is getting the value from source register and perform NOT operation on this value thanks to ALU then transfer this value to the desired destination register. Due to the fact that both SRCREG1 and DESTREG can be one of the register from Register File or Address Register File, there occurs 4 possibilities:

- DESTREG (**ARF**)$\leftarrow$ $\lhd$ SRCREG1 (**ARF**)
- DESTREG (**ARF**)$\leftarrow$ $\lhd$ SRCREG1 (**RF**)
- DESTREG (**RF**)$\leftarrow$ $\lhd$ SRCREG1 (**ARF**)
- DESTREG (**RF**)$\leftarrow$ $\lhd$ SRCREG1 (**RF**)

($\lhd$: represents one of the three operations which are NOT, LSL, LSR,
ARF: Address Register File, RF: Register File)

These four possibilities can be examined in two groups:

SRCREG1 belongs to Address Register File :

When SRCREG1 belongs to Address Register File, the operation cannot be done directly with only one clock cycle since Adress Register File's output doesn't have direct link to ALU. Therefore, regardless of DESTREG, before performing operation with ALU, the data should be transferred to one of the Temporary Registers of Register File (we prefered using T1 register).

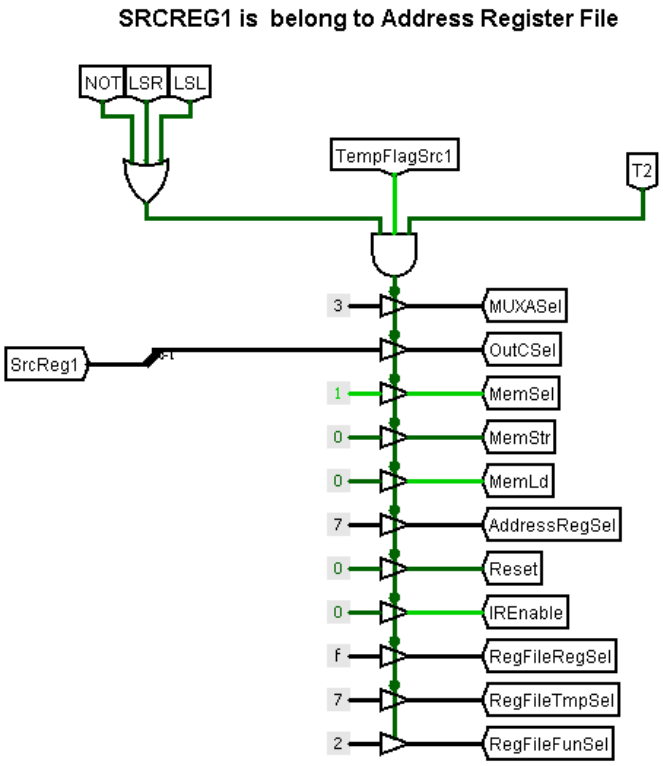

Figure 15: NOT, LSR and LSL operations' T2 moment while SRCREG1 is belong to Address Register File

To perform that at T2 moment, we have given following inputs according to the Figure 21:

T2: $T1 \leftarrow SRCREG1$

- MUXASel = 11 (x3): Selects input going to the Register File as OutC output of Adress Register File.
- OutCSel: Selects OutC output according to least significant 2-bit of SrcReg1 (IR(7-4)) as followings:
  - SrcReg1(1-0) = 00 or SrcReg1(1-0) = 01 : Data in PC is given as OutC output.
  - SrcReg1(1-0) = 10: Data in AR is given as OutC output.
  - SrcReg1(1-0) = 11: Data in SP is given as OutC output.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel = 111 (x7): Disables all registers of Address Register File.

- Reset = 0: Sends increment signal to the Sequential Counter and makes T3 clock signal 1.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111 (xF): Disables all the registers of Register File.
- RegFileTmpSel = 0111 (x7): Enables only temporary register T1.
- RegFileFunSel = 10 (x2): Load data to enabled registers.

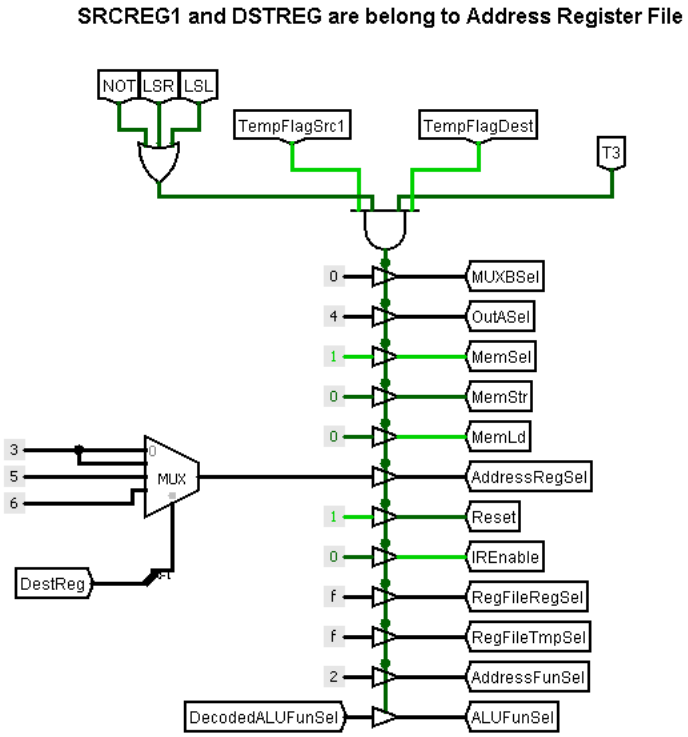

Figure 16: NOT, LSR and LSL operations' T3 moment while SRCREG1 and DESTREG are belong to Address Register File

After transferring SRCREG1's data to the T1 register, now the operation can be performed on data by ALU and then obtained data can be transferred to desired destination register. In this moment, destination register's location become important.

If DESTREG belongs to Address Register File, data manipulated by ALU is transferred to desired register of Address Register File. To perform that at T3 moment, we have given following inputs according to the Figure 35:

T3: $DESTREG \leftarrow T1, SC \leftarrow 0$

- MUXBSel = 00 (x0): Selects input going to the Address Register File as output of ALU.
- OutASel = 100: (x4) Sends data in register T1 to the input A of the ALU.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel: Enables desired destination register according to least significant 2-bit of DestReg (IR(11-8)) using 4x1 MUX as followings:
  - DestReg(1-0) = 00 or DestReg(1-0) = 01 : AddressRegsel = 011 (x3): Enables PC.
  - DestReg(1-0) = 10: AddressRegsel = 101 (x5): Enables AR.
  - DestReg(1-0) = 11: AddressRegsel = 110 (x6): Enables SP.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111 (xF): Disables all the registers of Register File.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- AdressFunSel = 10 (x2): Loads data to enabled registers.
- ALUFunSel = DecodedALUFunSel: For NOT operation, DecodedALUFunSel = 0010 and for LSL operation, DecodedALUFunSel = 1010 and for LSR operation, DecodedALUFunSel = 1011.

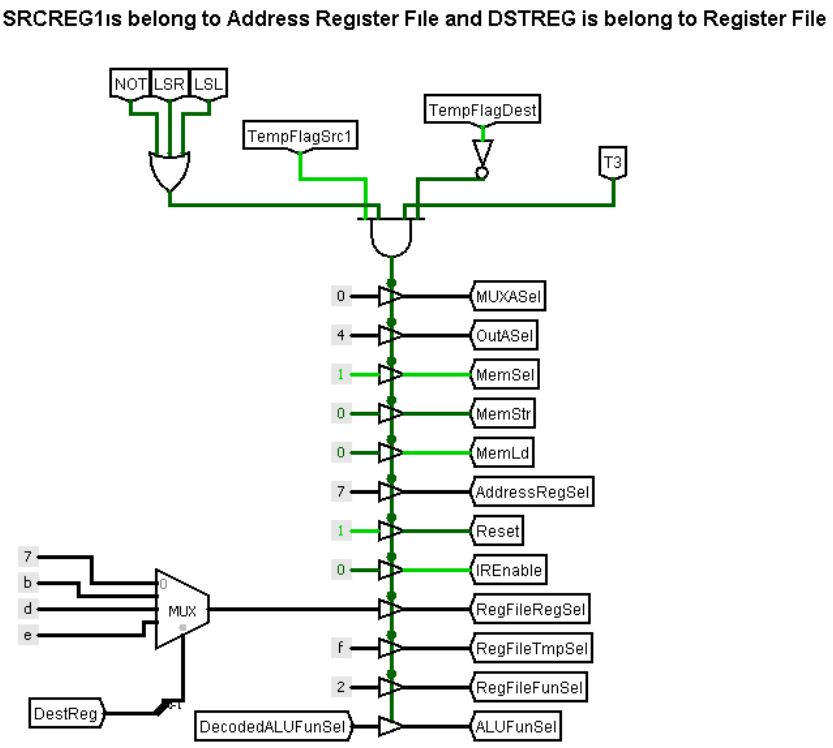

Figure 17: NOT, LSR and LSL operations' T3 moment while SRCREG1 is belong to Address Register File and DESTREG is belong to Register File

If DESTREG belongs to Register File, data manipulated by ALU is transferred to desired register of Register File. To perform that at T3 moment, we have given following inputs according to the Figure 23:

T3: $DESTREG \leftarrow T1, SC \leftarrow 0$

- MUXASel = 00 (x0): Selects input going to the Register File as output of ALU.
- OutASel = 100: (x4) Sends data in register T1 to the input A of the ALU.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel = 111 (x7): Disables all the registers of Address Register File.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel: Enables desired destination register according to least significant 2-bit of DestReg (IR(11-8)) using 4x1 MUX as followings:

  - DestReg(1-0) = 00: AddressRegsel = 0111 (x7): Enables R1.
  - DestReg(1-0) = 01: AddressRegsel = 1011 (xB): Enables R2.
  - DestReg(1-0) = 10: AddressRegsel = 1101 (xD): Enables R3.
  - DestReg(1-0) = 11: AddressRegsel = 1110 (xE): Enables R4.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- RegFileFunSel = 10 (x2): Loads data to enabled registers.
- ALUFunSel = DecodedALUFunSel: For NOT operation, DecodedALUFunSel = 0010 and for LSL operation, DecodedALUFunSel = 1010 and for LSR operation, DecodedALUFunSel = 1011.

SRCREG1 belongs to Register File :

When SRCREG1 belongs to Register File, the operation can be done directly with only one clock cycle since Register File's output has direct link to ALU.

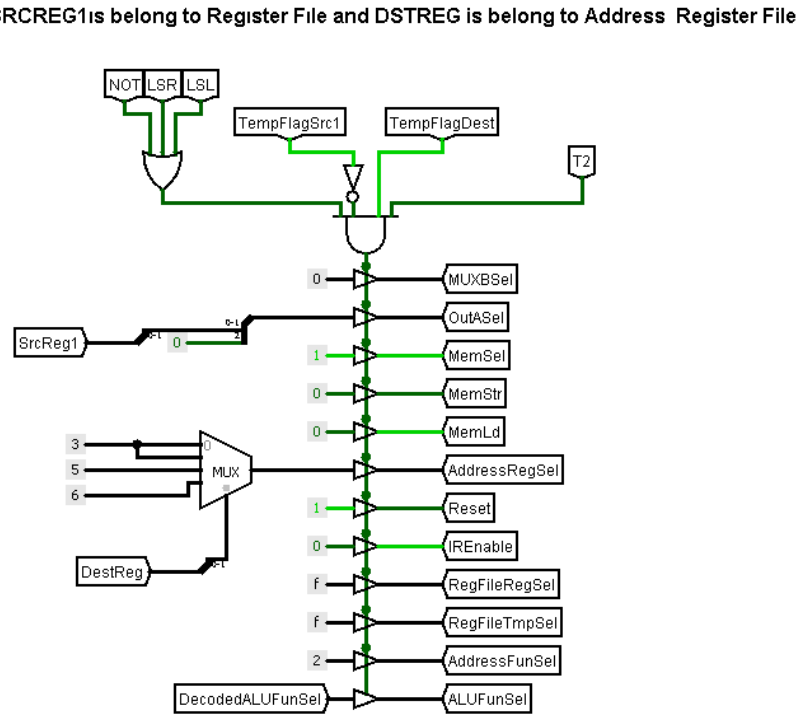

Figure 18: NOT, LSR and LSL operations' T2 moment while SRCREG1 is belong to Register File and DESTREG is belong to Address Register File

If DESTREG belongs to Address Register File, data is taken from Register file then data is manipulated by ALU and then transferred to desired register of Address Register File. To perform that at T2 moment, we have given following inputs according to the Figure 18:

T3: $DESTREG \leftarrow SRCREG1, SC \leftarrow 0$

- MUXBSel = 00 (x0): Selects input going to the Address Register File as output of ALU.
- OutASel: Selects output data going to input A of ALU according to least significant 2-bit of SrcReg1 (IR(7-4)) using a splitter and constant 0 as followings:
  - SrcReg1(1-0) = 00 : OutASel = 000: Data in R1 is given as OutA output so that sent to input A of ALU.
  - SrcReg1(1-0) = 01 : OutASel = 001: Data in R2 is given as OutA output so that sent to input A of ALU.
  - SrcReg1(1-0) = 10 : OutASel = 010: Data in R3 is given as OutA output so that sent to input A of ALU.
  - SrcReg1(1-0) = 11 : OutASel = 011: Data in R4 is given as OutA output so that sent to input A of ALU.
- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel: Enables desired destination register according to least significant 2-bit of DestReg (IR(11-8)) using 4x1 MUX as followings:
  - DestReg(1-0) = 00 or DestReg(1-0) = 01 : AddressRegsel = 011 (x3): Enables PC.
  - DestReg(1-0) = 10: AddressRegsel = 101 (x5): Enables AR.
  - DestReg(1-0) = 11: AddressRegsel = 110 (x6): Enables SP.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111 (xF): Disables all the registers of Register File.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- AdressFunSel = 10 (x2): Loads data to enabled registers.
- ALUFunSel = DecodedALUFunSel: For NOT operation, DecodedALUFunSel = 0010 and for LSL operation, DecodedALUFunSel = 1010 and for LSR operation, DecodedALUFunSel = 1011.

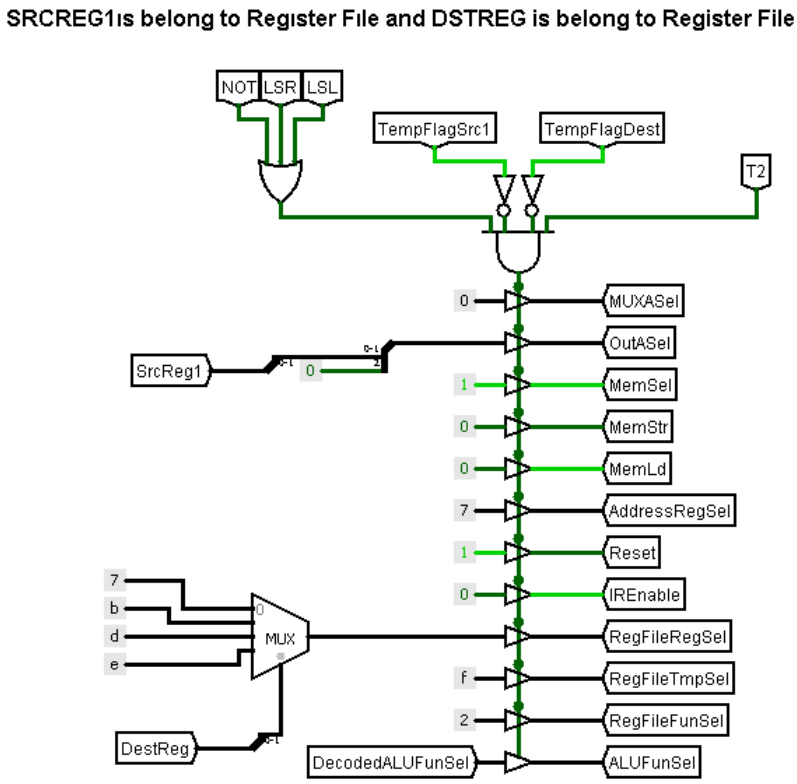

Figure 19: NOT, LSR and LSL operations' T2 moment while SRCREG1 is belong to Register File and DESTREG is belong to Register File

If DESTREG belongs to Address Register File, data is taken from Register file then data is manipulated by ALU and then transferred to desired register of Register File. To perform that at T2 moment, we have given following inputs according to the Figure 19:

T3: $DESTREG \leftarrow SRCREG1, SC \leftarrow 0$

- MUXASel = 00 (x0): Selects input going to the Register File as output of ALU.
- OutASel: Selects output data going to input A of ALU according to least significant 2-bit of SrcReg1 (IR(7-4)) using a splitter and constant 0 as followings:
  - SrcReg1(1-0) = 00 : OutASel = 000: Data in R1 is given as OutA output so that sent to input A of ALU.
  - SrcReg1(1-0) = 01 : OutASel = 001: Data in R2 is given as OutA output so that sent to input A of ALU.
  - SrcReg1(1-0) = 10 : OutASel = 010: Data in R3 is given as OutA output so that sent to input A of ALU.

  - SrcReg1(1-0) = 11 : OutASel = 011: Data in R4 is given as OutA output so that sent to input A of ALU.

- MemSel = 1: Enables memory chip select.
- MemStr = 0: Disables writing into memory.
- MemLd = 0: Disables loading memory content to the memory output.
- AddressRegSel = 111 (x7): Disables all the registers of Address Register File.
- Reset = 1: Sends reset signal to the Sequential Counter to reset SC for the purpose of turning back to T0 moment after this operation.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel: Enables desired destination register according to least significant 2-bit of DestReg (IR(11-8)) using 4x1 MUX as followings:
  - DestReg(1-0) = 00: AddressRegsel = 0111 (x7): Enables R1.
  - DestReg(1-0) = 01: AddressRegsel = 1011 (xB): Enables R2.
  - DestReg(1-0) = 10: AddressRegsel = 1101 (xD): Enables R3.
  - DestReg(1-0) = 11: AddressRegsel = 1110 (xE): Enables R4.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- RegFileFunSel = 10 (x2): Loads data to enabled registers.
- ALUFunSel = DecodedALUFunSel: For NOT operation, DecodedALUFunSel = 0010 and for LSL operation, DecodedALUFunSel = 1010 and for LSR operation, DecodedALUFunSel = 1011.

2.11 INC(0x0D), DEC(0x0E)

At the first stage of the fetch phase temporary registers of the Register File were reset. For the increment and decrement operation Temporary register $T4$ is going to be used. In order to use it 1 must be loaded. The initial value stored in the $T4$ is 0. 1 can be loaded by using the increment functionality of the register itself.

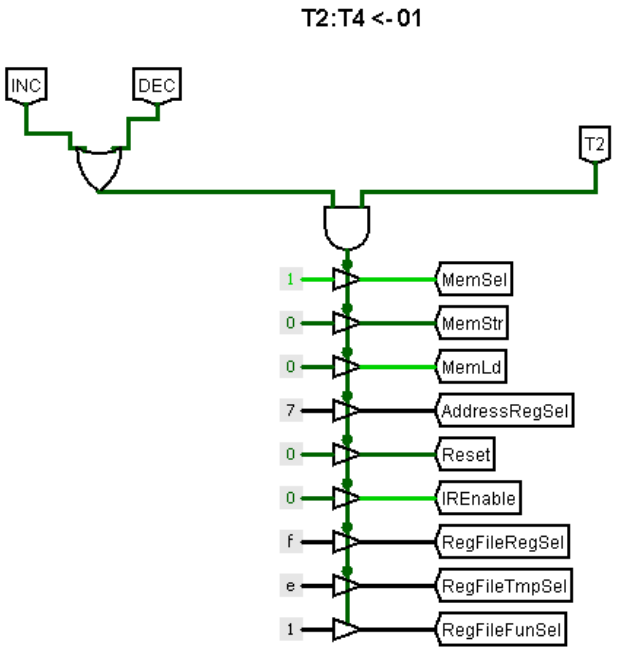

Figure 20: Load 1 into Temporary Register 4

T2: $T4 \leftarrow T4 + 1$

- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- AddressRegSel = 111 (x7). Disables all the registers of ARF.
- Reset = 0. Sends increment signal to the Sequential Counter and makes T3 clock signal 1.
- IREnable = 0. Disables Instruction Register.
- RegFileRegSel = 1111(xf). Disables all the registers of the Register File.
- RegFileTmpSel = 1110(xe). Enables just $T4$ to load 1.
- RegFileFunSel = 01(x1). Increments $T4$ (initially 0). $T4$ stores 1 now.

At this point, There are four possibilities of this operation:

- Both SRCREG1 and DESTREG can be belong to the Register File.
- SRCREG1 can be belong to the Register File and DESTREG can be belong to the ARF.
- Both SRCREG1 and DESTREG can be belong to the ARF.

- SRCREG1 can be belong to the ARF and DESTREG can be belong to the Register File.

According to these possibilities different actions must be taken.

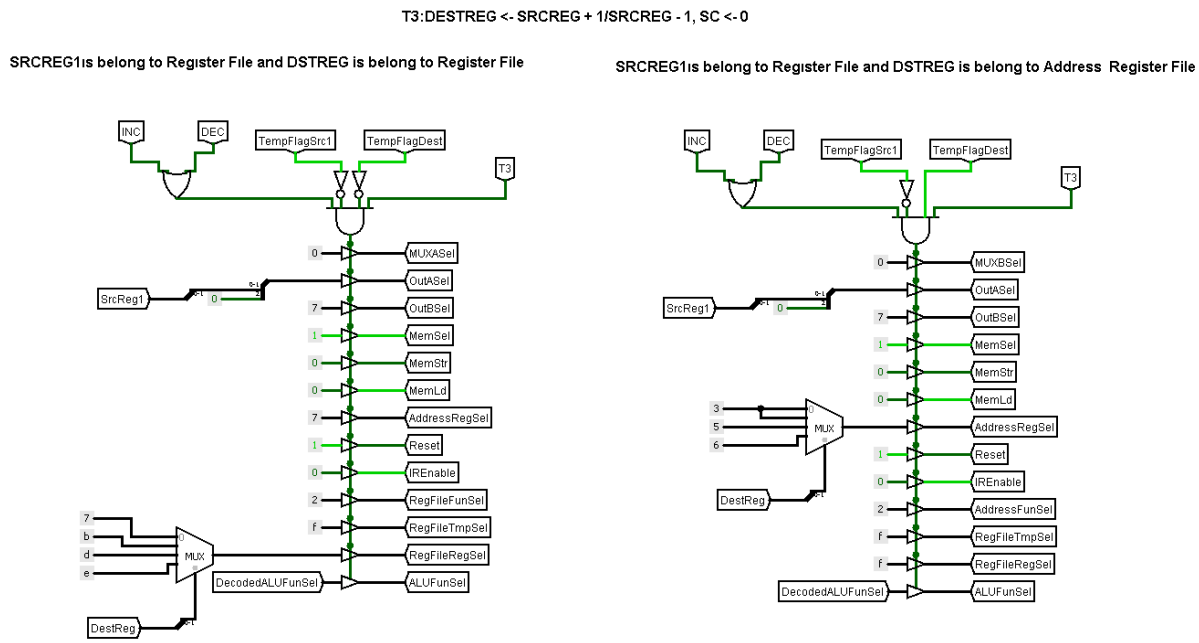

Figure 21: SRCREG1 belongs to Register File

If SRCREG1 belongs to Register File, increment or decrement and directly load to the DESTREG:

T3: $DESTREG \leftarrow SRCREG + 1/SRCREG - 1$**, SC ¡- 0**

- If DSTREG belongs to RF, MUXASel = 0. If belongs to ARF MUXBSel = 0. In order to transfer the incremented or decremented data to the correct destination.
- OutASel = 0 concatinated with SrcReg1(0-1). Selects the SRCREG1 register.
- OutBSel = 111 (x7). Passes $T4$ which stores 1 to B input of the ALU.
- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- If DESTREG belongs to RF, then AddressRegSel = 111 (x7) disables all the registers of the ARF. If DESTREG belongs to ARF, then it is set by the output of MUX (with the DestReg(0-1) selector input). It enables PC, AR or SP according to DestReg(0-1).
- Reset = 1. Sends reset signal to the Sequential Counter and makes T0 clock signal 1.

- IREnable = 0. Disables Instruction Register.
- If DESTREG belongs to RF, then RegFileFunSel = 10 (x2). Loads data into register of RF. If DESTREG belongs to ARF, then AddressFunSel = 10 (x2). Loads data into register of ARF.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- If DESTREG belongs to RF then RegFileRegSel is set by the output of MUX (with the DestReg(0-1)). It enables R1, R2, R3 or R4 according to DestReg(0-1). If DESTREG belongs to ARF, then RegFileRegSel = 1111 (xf), disables all the registers of the Register File.
- ALUFunSel = DecodedALUFunSel. Sets ALUFunSel with right functionality (x4(ADD) or x6 (SUB) in this case).

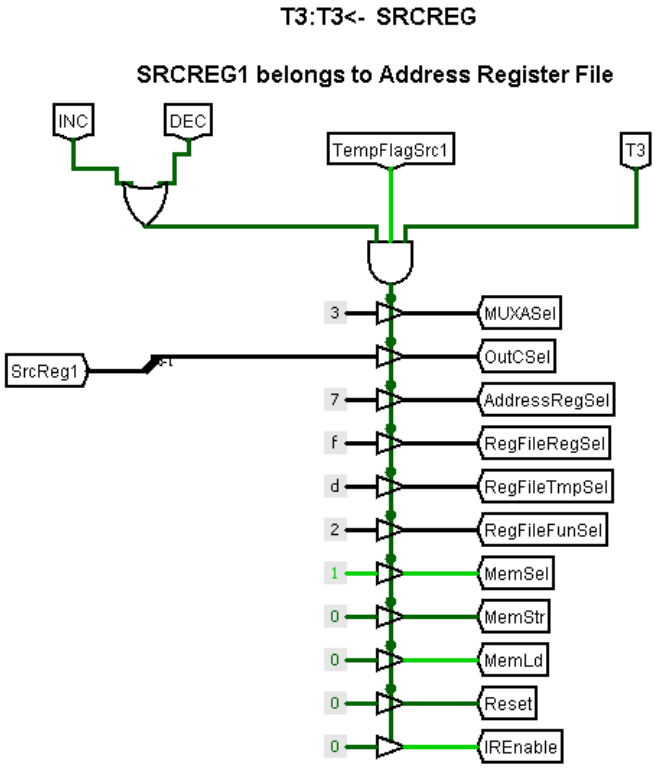

Figure 22: SRCREG1 belongs to ARF(T3)

If SRCREG1 is belong to ARF. Then corresponding data in the register must be carried to a temporary register in the RF in order to be operand of the ALU. In this design $T3$ is used to store data.

T3: $T3 \leftarrow SRCREG1$

- MUXASel = 11 (x3). Sends ARF register data to the RF.

- OutCSel = SrcReg(0-1). Selects the operand register data by the least significant 2 bits of the SrcReg.
- AddressRegSel = 111 (x7). Disables all the registers of ARF.
- RegFileRegSel = 1111 (xf). Disables all the registers of the Register File.
- RegFileTmpSel = 1101 (xd). Enables just $T3$ to load data.
- RegFileFunSel = 10(x2). Loads input data into $T3$ (initially 0).
- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- Reset = 0. Sends increment signal to the Sequential Counter and makes T4 clock signal 1.
- IREnable = 0. Disables Instruction Register.

Figure 23: SRCREG1 belongs to ARF(T4)

Finally, desired operation can be done and stored in the DESTREG.

T4: $DESTREG \leftarrow T3 + 1/T3 - 1, SC < -0$

- If DSTREG belongs to RF, MUXASel = 0. If belongs to ARF MUXBSel = 0. In order to transfer the incremented or decremented data to the correct destination.
- OutASel = 6 sends $T3$ to the A input of the ALU.
- OutBSel = 111 (x7). Sends $T4$ which stores 1 to B input of the ALU.
- If DESTREG belongs to RF, then AddressRegSel = 111 (x7) disables all the registers of the ARF. If DESTREG belongs to ARF, then it is set by the output of MUX (with the DestReg(0-1) selector input). It enables PC, AR or SP according to DestReg(0-1).

- If DESTREG belongs to RF then RegFileRegSel is set by the output of MUX (with the DestReg(0-1)). It enables R1, R2, R3 or R4 according to DestReg(0-1). If DESTREG belongs to ARF, then RegFileRegSel = 1111 (xf), disables all the registers of the Register File.
- If DESTREG belongs to RF, then RegFileFunSel = 10 (x2). Loads data into register of RF. If DESTREG belongs to ARF, then AddressFunSel = 10 (x2). Loads data into register of ARF.
- RegFileTmpSel = 1111 (xF): Disables all the temporary registers of Register File.
- MemSel = 1. Memory chip selection.
- MemStr = 0. Disables storing into memory.
- MemLd = 0. Disables read from memory.
- Reset = 1. Sends reset signal to the Sequential Counter and makes T0 clock signal 1.
- IREnable = 0. Disables Instruction Register.
- ALUFunSel = DecodedALUFunSel. Sets ALUFunSel with right functionality (x4(ADD) or x6 (SUB) in this case).

2.12 AND(x04), OR(x05),ADD(x07),SUB(x08)

After fetch and decode operations, we can consider and combine operations together where Opcodes are equal to 0x04 (AND), 0x05(OR), 0x07(ADD) and 0x08(SUB). This is because all these operations have two inputs and all they are done in the ALU. We do an operation with SRCREG1 and SRCREG2 and then the result is assigned to the DESTREG. The operations can be seen below:

- DESTREG ← SRCREG2 AND SRCREG1 (Opcode = 0x04)
- DESTREG ← SRCREG2 OR SRCREG1 (Opcode = 0x05)
- DESTREG ← SRCREG2 + SRCREG1 (Opcode = 0x07)
- DESTREG ← SRCREG2 - SRCREG1 (Opcode = 0x08)

SRCREG1, SRCREG2 and DESTREG can be belong to Address Register File or Register File. Therefore, for these 4 operations, there are 8 possibilities:

- (1) DESTREG (**ARF**)← SRCREG2 (**RF**) * SRCREG1 (**RF**)

- (2) DESTREG (**RF**)← SRCREG2 (**RF**) * SRCREG1 (**RF**)
- (3) DESTREG (**RF**)← SRCREG2 (**ARF**) * SRCREG1 (**RF**)
- (4) DESTREG (**ARF**)← SRCREG2 (**ARF**) * SRCREG1 (**RF**)
- (5) DESTREG (**RF**)← SRCREG2 (**RF**) * SRCREG1 (**ARF**)
- (6) DESTREG (**ARF**)← SRCREG2 (**RF**) * SRCREG1 (**ARF**)
- (7) DESTREG (**RF**)← SRCREG2 (**ARF**) * SRCREG1 (**ARF**)
- (8) DESTREG (**ARF**)← SRCREG2 (**ARF**) * SRCREG1 (**ARF**)

(*: represents one of the four operations which are AND, OR, ADD, SUB,
ARF: Address Register File, RF: Register File)

One clock cyle is enough when SRCREG's belong to the Register File.

T2: DESTREG (ARF)← SRCREG2 (RF) * SRCREG1 (RF), SC ← 0 (1)

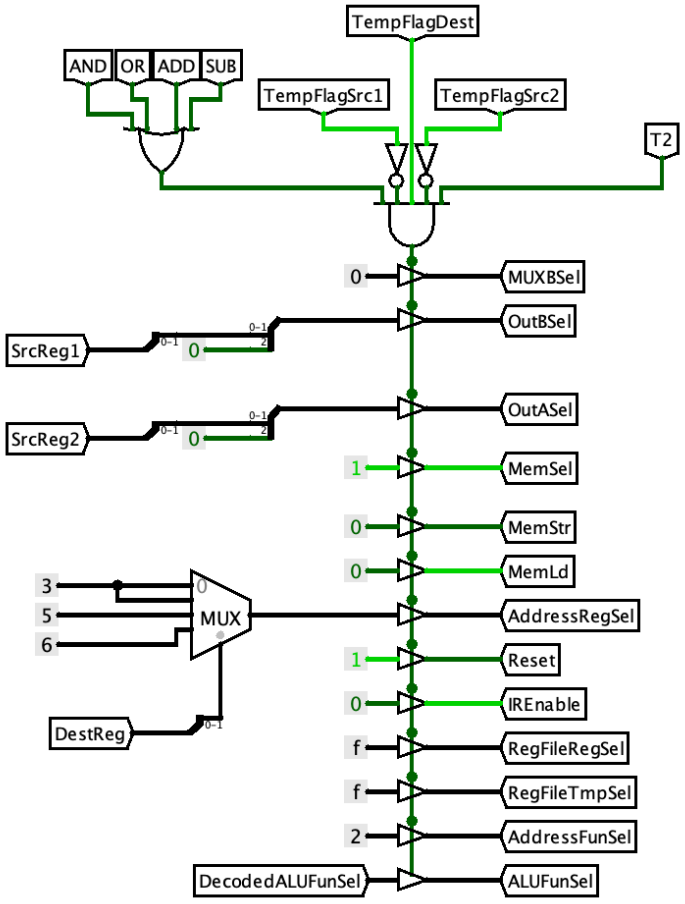

Figure 24: SRCREG1 and SRCREG2 belong to RF, DESTREG belongs to ARF

- MUXBSel = 00: Selects output of ALU for ARF.
- OutBSel = Least significant two bits of SrcReg1 and 0 for msb. Reads SRCREG1 value from Register File
- OutASel = Least significant two bits of SrcReg2 and 0 for msb. Reads SRCREG1 value from Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of PC, AR or SP according to DestReg(0-1).
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all registers(R1, R2, R3, R4) of Register File
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File
- AddressFunSel = 10(x2): Loads data into a register of Address Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

T2: DESTREG (RF)← SRCREG2 (RF) * SRCREG1 (RF), SC ← 0 (2)

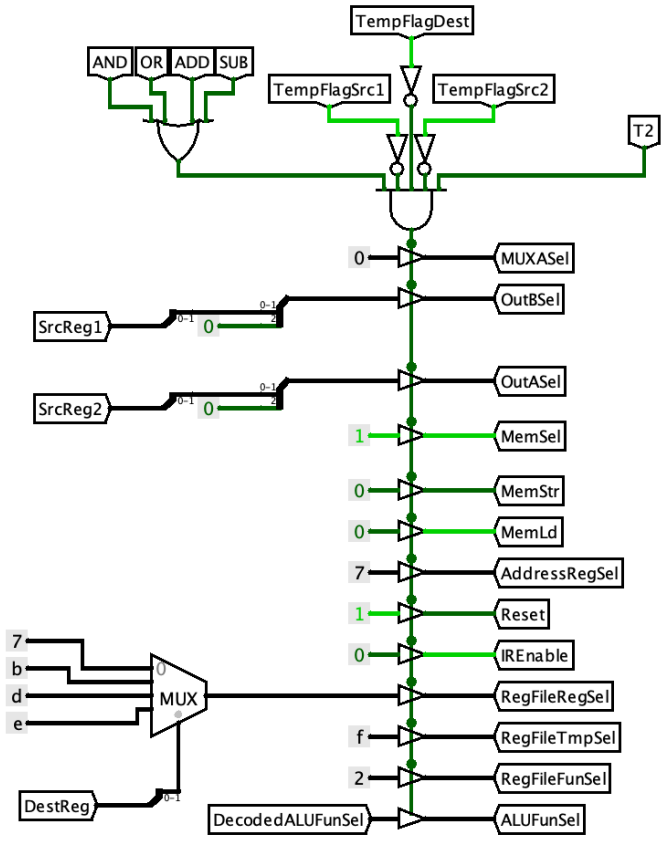

Figure 25: SRCREG1,SRCREG2 and DESTREG belong to RF

- MUXASel = 00: Selects output of ALU for RF.
- OutBSel = Least significant two bits of SrcReg1 and 0 for msb. Reads SRCREG1 value from Register File
- OutASel = Least significant two bits of SrcReg2 and 0 for msb. Reads SRCREG2 value from Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of R1, R2, R3 or R4 according to DestReg(0-1).

- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File
- RegFileFunSel = 10(x2): Loads data into a register of Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

T2: T2 ← SRCREG2 (ARF), SRCREG1(RF), DESTREG(RF/ARF) (3-4)

These processes, in T2 time signal, are done in both cases where DESTREG belongs to either RF or ARF. In short, T2 time signal processes are done when SRCREG1 belongs to RF and SRCREG2 belongs to ARF, independently of DESTREG.

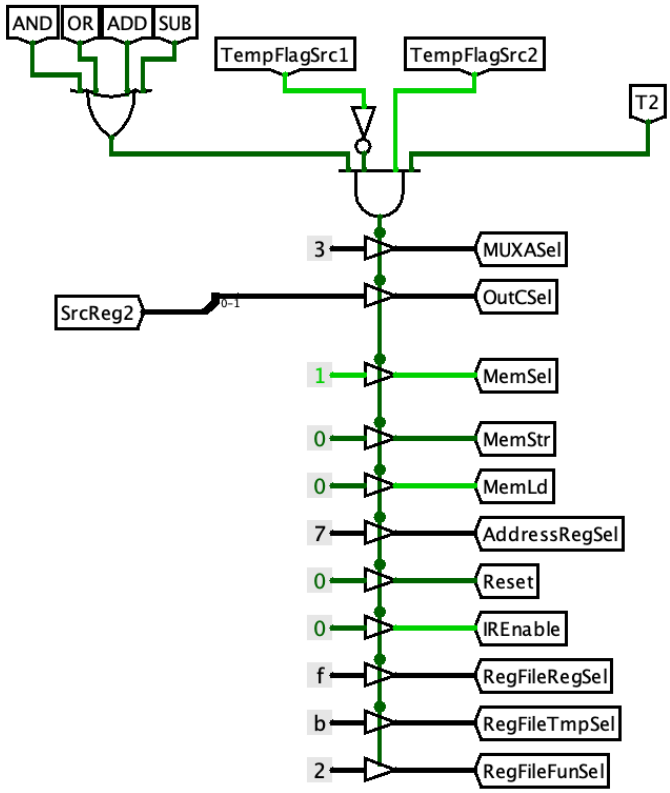

Figure 26: SRCREG1 belongs to RF and SRCREG2 belongs to ARF

- MUXASel = 11(x3): Selects OutC of ARF.
- OutCSel = Least significant two bits of SrcReg2. Reads SRCREG2 value from Address Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory

- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 0: Disables to reset a current clock signal.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all registers(R1, R2, R3, R4) of Register File
- RegFileTmpSel = 1011(xb): Enables just T2 temporary register of Register File in order to load data.
- RegFileFunSel = 10(x2): Loads data into a register of Register File.

If DESTREG belongs to Register File:

T3: DESTREG (RF)← SRCREG2 (ARF) * SRCREG1 (RF), SC ← 0 (3)

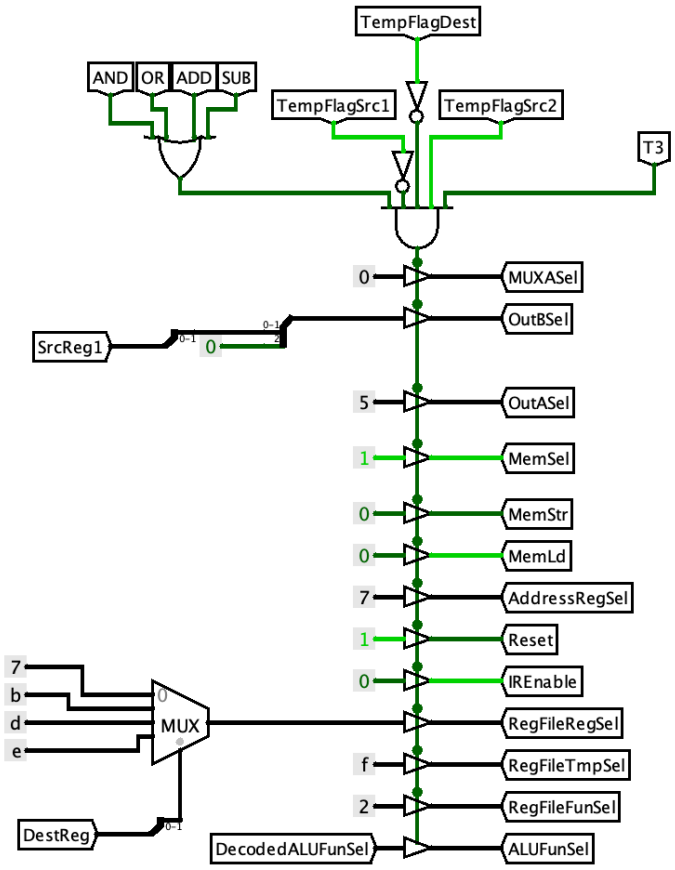

Figure 27: DESTREG and SRCREG1 belong to RF, and SRCREG2 belongs to ARF

- MUXASel = 00: Selects output of ALU for RF.
- OutBSel = Least significant two bits of SrcReg1 and 0 for msb. Reads SRCREG1 value from Register File

- OutASel = 101(x5): Enables to read only T2 register from Register File and use it for ALU operations.
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of R1, R2, R3 or R4 according to DestReg(0-1) for DESTREG.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File
- RegFileFunSel = 10(x2): Loads data into a register of Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

If DESTREG belongs to Address Register File:

T3: DESTREG (ARF)← SRCREG2 (ARF) * SRCREG1 (RF), SC ← 0 (4)

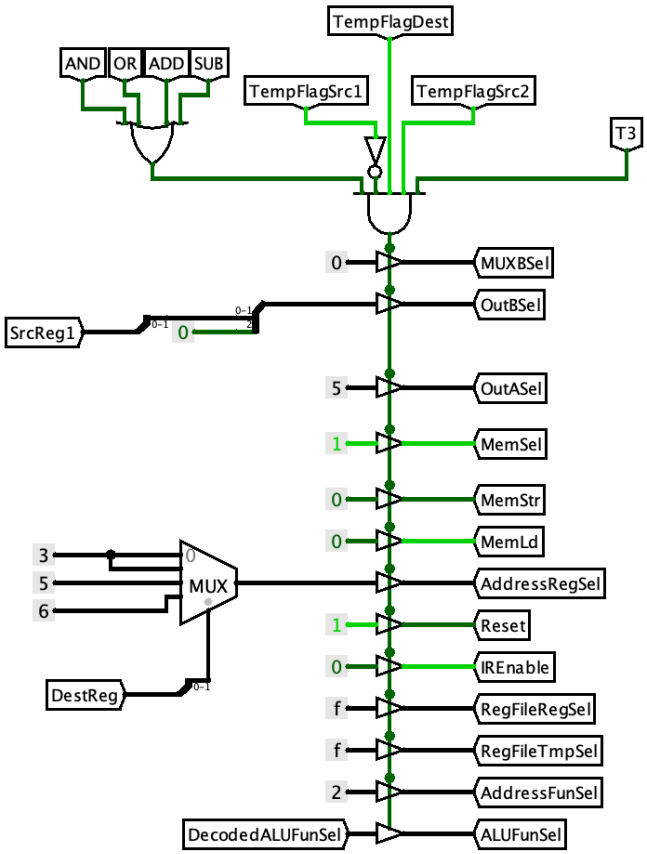

Figure 28: SRCREG1 belongs to RF, DESTREG and SRCREG2 belong to ARF

- MUXBSel = 00: Selects output of ALU for ARF.
- OutBSel = Least significant two bits of SrcReg1 and 0 for msb. Reads SRCREG1 value from Register File
- OutASel = 101(x5): Enables to read only T2 register from Register File and use it for ALU operations.
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of PC, AR or SP according to DestReg(0-1) for DESTREG.
- Reset = 1: Resets the clock signal as T0.

- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all R1, R2, R3, R4 registers of Register File.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File.
- AddressFunSel = 10(x2): Loads data into a temporary register of Address Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

In cases (5) and (6), SRCREG1 belongs to ARF and SRCREG2 belongs to RF. Unlike cases (3) and (4), the locations(ARF or RF) of these SRC inputs change so that we use temporary register T1 for SRCREG1 instead of T2 since it is already used for SRCREG2.

T2: T1 ← SRCREG1 (ARF), SRCREG2(RF), DESTREG(RF/ARF) (5-6)

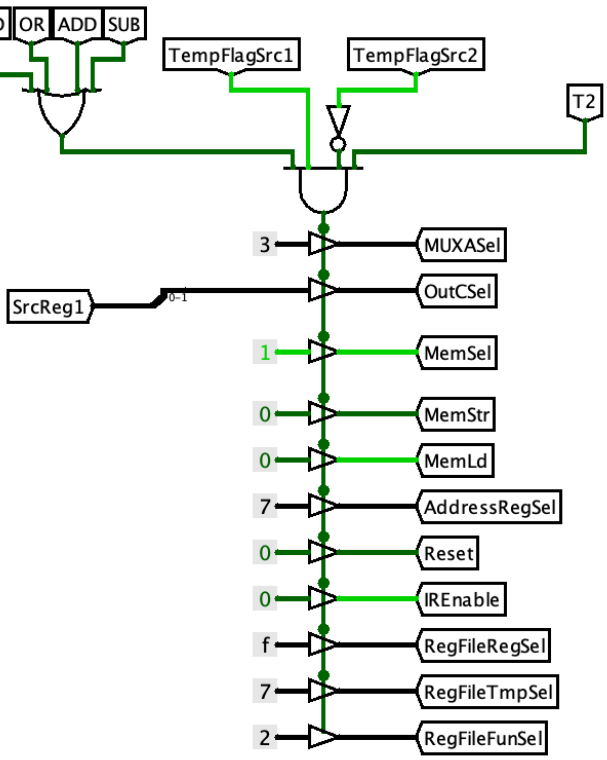

Figure 29: DESTREG and SRCREG1 belong to RF, and SRCREG2 belongs to ARF

- MUXASel = 11(x3): Selects OutC of ARF.
- OutCSel = Least significant two bits of SrcReg1. Reads SRCREG1 value from Address Register File

- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 0: Disables to reset a current clock signal.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all registers(R1, R2, R3, R4) of Register File
- RegFileTmpSel = 0111(x7): Enables just T1 temporary register of Register File in order to load data.
- RegFileFunSel = 10(x2): Loads data into a register of Register File.

If DESTREG belongs to Register File:

T3: DESTREG (RF)← SRCREG2 (RF) * SRCREG1 (ARF), SC ← 0 (5)

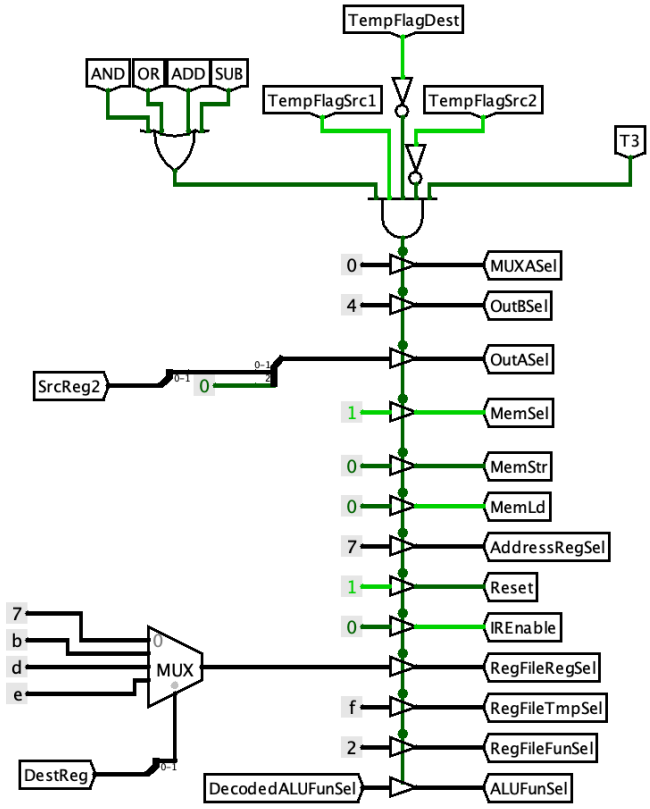

Figure 30: DESTREG and SRCREG1 belong to RF, and SRCREG2 belongs to ARF

The only difference from case (3) is OutASel and OutBSel.

- MUXASel = 00: Selects output of ALU for RF.
- OutBSel = 100(x4): Enables to read only T1 register from Register File and use it for ALU operations.
- OutASel = Least significant two bits of SrcReg2 and 0 for msb. Reads SRCREG2 value from Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of R1, R2, R3 or R4 according to DestReg(0-1) for DESTREG.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File
- RegFileFunSel = 10(x2): Loads data into a register of Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

If DESTREG belongs to Address Register File:

T3: DESTREG (ARF)← SRCREG2 (RF) * SRCREG1 (ARF), SC ← 0 (6)

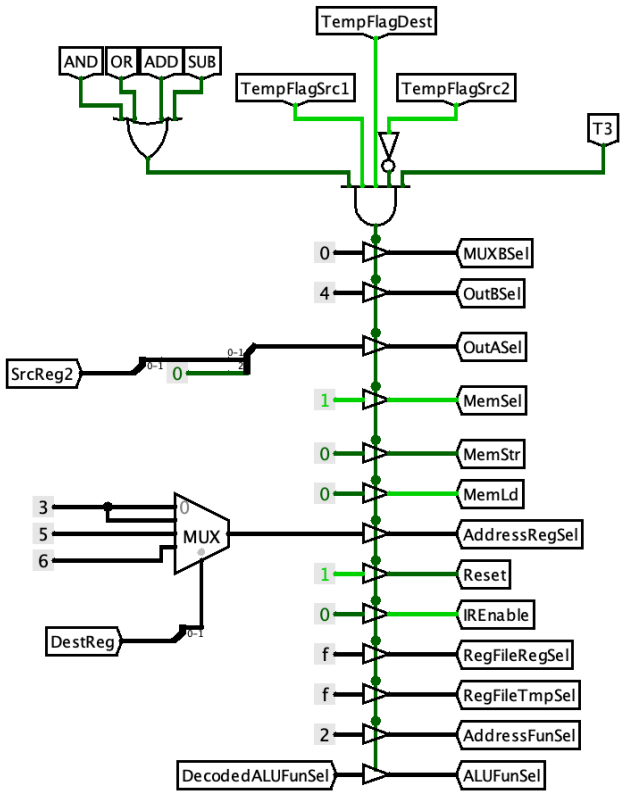

Figure 31: DESTREG and SRCREG1 belong to RF, and SRCREG2 belongs to ARF

Again, the only difference from case (4) is OutASel and OutBSel.

- MUXBSel = 00: Selects output of ALU for ARF.
- OutBSel = 100(x4): Enables to read only T1 register from Register File and use it for ALU operations.
- OutASel = Least significant two bits of SrcReg2 and 0 for msb. Reads SRCREG2 value from Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of PC, AR or SP according to DestReg(0-1) for DESTREG.
- Reset = 1: Resets the clock signal as T0.

- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all R1, R2, R3, R4 registers of Register File.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File.
- AddressFunSel = 10(x2): Loads data into a temporary register of Address Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

For case (7) and (8), we used 3 different time signal. Since both SRCREG1 and SRCREG2 belong to Address Register File, we assign SRCREG1 value to T1 register at T2 clock signal and assign SRCREG2 value to T2 register at T3 clock signal. Finally, according to whether the DESTREG belongs to AR or ARF, we performed the operations at T4 clock signal.

T2: T1 ← SRCREG1(ARF), SRCREG2(ARF), DESTREG(RF/ARF) (7-8)

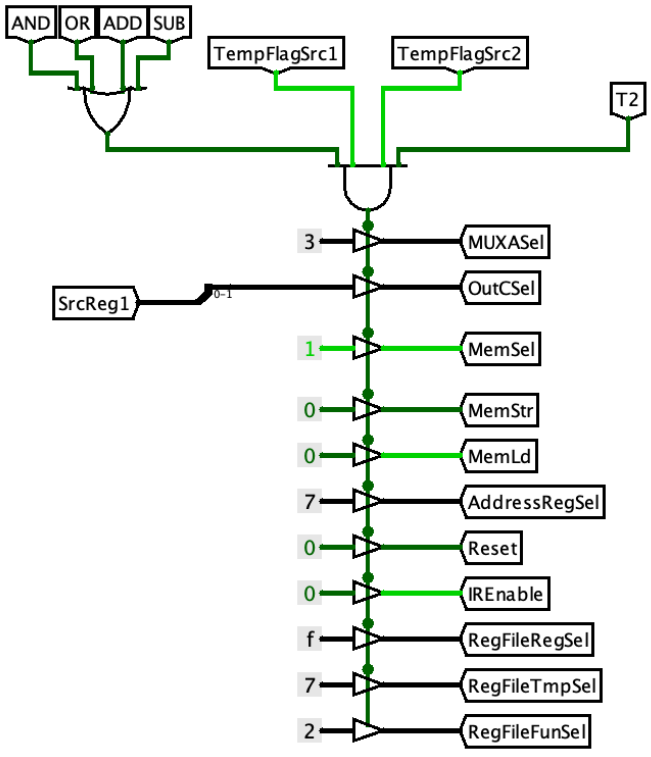

Figure 32: SRCREG1 belongs to ARF

This operation is the same as in Figure 29 (case 5-6) which means that we assign SRCREG1 to T1 for both of them. Therefore, we are not repeating same things here.

T3: T2 ← SRCREG2(ARF), SRCREG1(ARF), DESTREG(RF/ARF) (7-8)

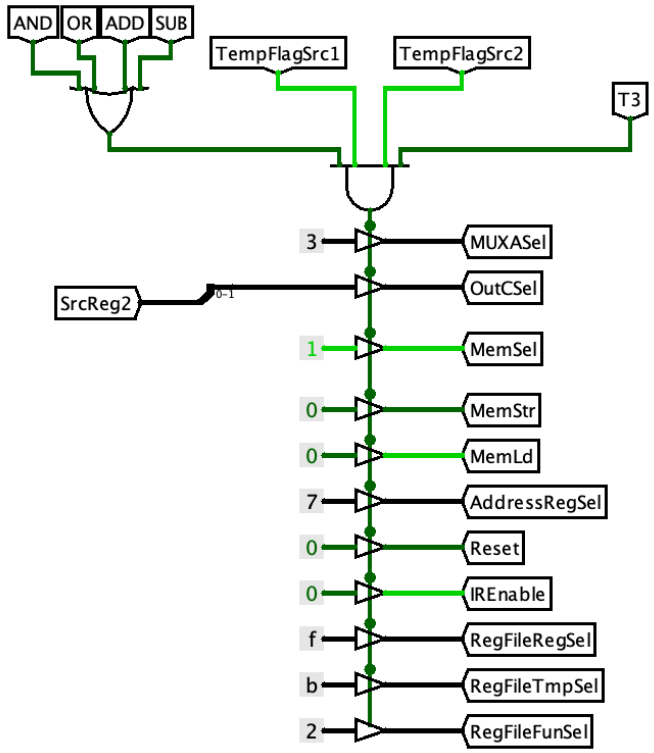

Figure 33: SRCREG2 belongs to ARF

Unlike Figure 32, we just change OutCSel as SrcReg2 instead SrcReg1 and RegFileTmpSel as 1011 instead 0111.

- MUXASel = 11(x3): Selects OutC of ARF.
- OutCSel = Least significant two bits of SrcReg2. Reads SRCREG2 value from Address Register File
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 0: Disables to reset a current clock signal.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all registers(R1, R2, R3, R4) of Register File

- RegFileTmpSel = 1011(xb): Enables just T2 temporary register of Register File in order to load data.
- RegFileFunSel = 10(x2): Loads data into a register of Register File.

If DESTREG belongs to Register File:

T4: DESTREG (RF)← SRCREG2 (ARF) * SRCREG1 (ARF), SC ← 0 (7)

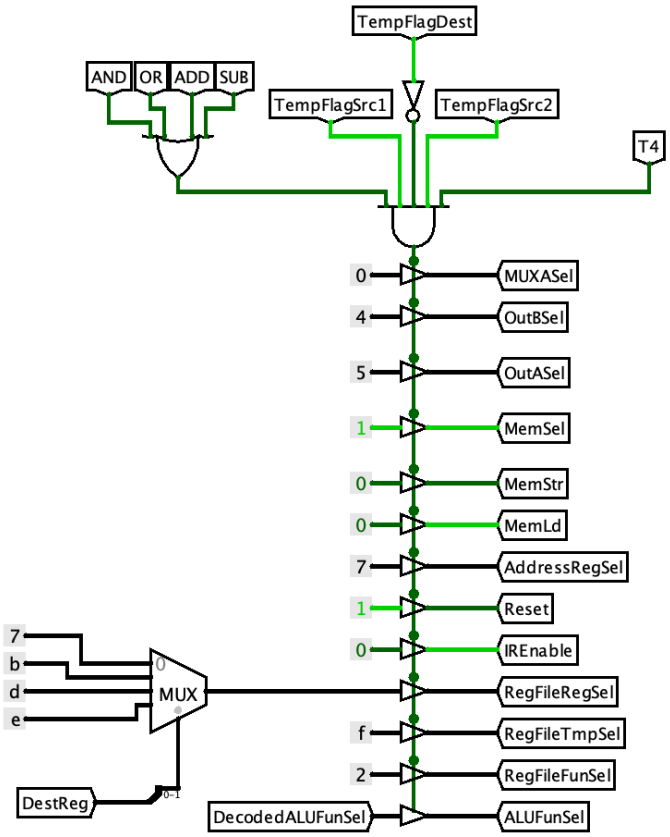

Figure 34: SRCREG1 and SRCREG2 belong to ARF, and DESTREG belongs to RF

- MUXASel = 00: Selects output of ALU for RF.
- OutBSel = 100(x4): Enables to read only T1 register from Register File for OutB and use it for ALU operations.
- OutASel = 101(x5): Enables to read only T2 register from Register File for OutA and use it for ALU operations.
- MemSel = 1
- MemStr = 0: Disables writing to memory

- MemLd = 0: Disables reading from memory
- AddressRegSel = 111(x7): Disable all registers of Address Register File.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of R1, R2, R3 or R4 according to DestReg(0-1) for DESTREG.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File
- RegFileFunSel = 10(x2): Loads data into a register of Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

If DESTREG belongs to Address Register File:

T4: DESTREG (ARF)← SRCREG2 (ARF) * SRCREG1 (ARF), SC ← 0 (8)

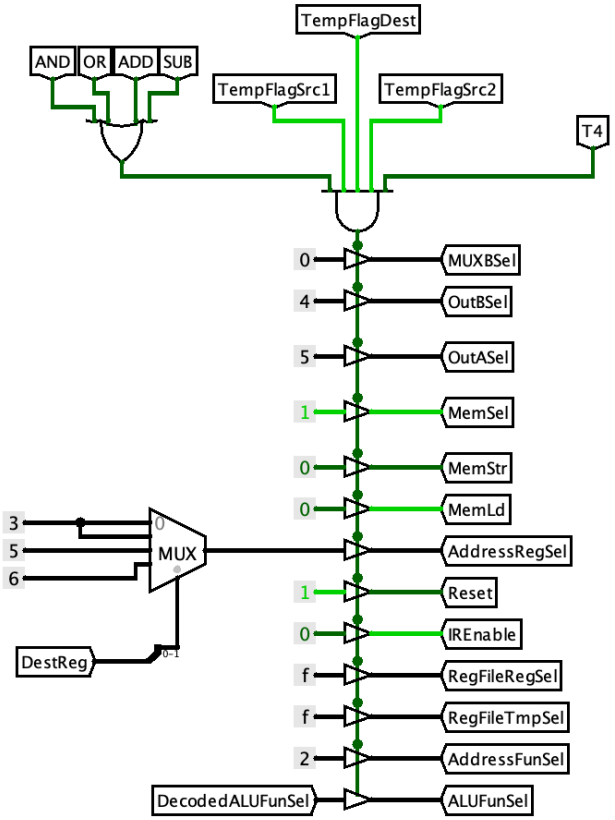

Figure 35: DESTREG, SRCREG1 and SRCREG2 belong to ARF

- MUXBSel = 00: Selects output of ALU for ARF.
- OutBSel = 100(x4): Enables to read only T1 register from Register File for OutB and use it for ALU operations.
- OutASel = 101(x5): Enables to read only T2 register from Register File for OutA and use it for ALU operations.
- MemSel = 1
- MemStr = 0: Disables writing to memory
- MemLd = 0: Disables reading from memory
- AddressRegSel = The output of a MUX with the DestReg(0-1) selector input. It choose one of PC, AR or SP according to DestReg(0-1) for DESTREG.
- Reset = 1: Resets the clock signal as T0.
- IREnable = 0: Disables Instruction Register.
- RegFileRegSel = 1111(xf): Disables all R1, R2, R3, R4 registers of Register File.
- RegFileTmpSel = 1111(xf): Disables all temporary registers of Register File.
- AddressFunSel = 10(x2): Loads data into a temporary register of Address Register File.
- ALUFunSel = DecodedALUFunSel: Sets ALUFunSel with correct functionality using MUX with opcode value.

3 RESULTS

We have tested our design with some test cases.

3.1 Case-1

The first case was given in the project description. According to this case, data in the memory addresses 0xA0, 0xA1, 0xA2, 0xA3 and 0xA4 must be added and written into memory address 0xA6.

Listing 1: Meaningful Instruction of the Desired Process:

```
BRA 0x20            (PC <- 20)
```

```
        LD R1 IM 0x05     (R1 <- 05)
        LD R2 IM 0x00     (R2 <- 00)
        LD R3 IM 0xA0     (R3 <- A0)
        MOV AR R3         (AR <- R3 = A0)
LABEL:  LD R3 D           (R3 <- M[AR])
        ADD R2 R2 R3      (R2 <- R2 + R3)
        INC AR AR         (AR <- AR + 1)
        DEC R1 R1         (R1 <- R1 - 1))
        BNE IM LABEL      (If Z = 0 return LABEL)
        INC AR AR         (AR <- AR + 1)
        ST R2 D           (M[AR] <- R2)
```

Listing 2: Memory Format for Instructions:

```
v2.0 raw
20 4 30*0 5 14 0 15 a0
16 60 32 0 12 56 75 20
d2 40 e4 28 f4 20 d2 4
21 106*0 1 2 3 4 5
```

In this memory format 1, 2, 3, 4 and 5 is written into the addresses which are going to be added. The result must be 0x0f (1 + 2 + 3 + 4 + 5 = 15 in decimal).
After executing the all the instructions, the desired result 0x0f is written to desired location 0xA6.

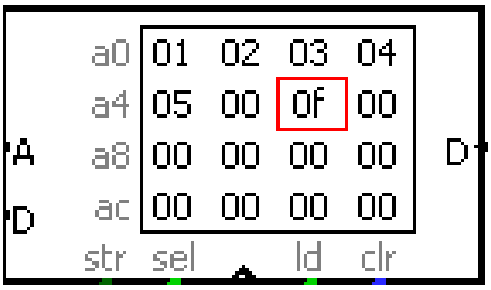

Figure 36: Result of the first test case

3.2 Case-2

The second case tests all the (source and destination register) combinations of NOT, LSL, LSR operations.

Listing 3: Meaningful Instruction of the Desired Process:

```
LD R1 IM 0x01     (R1 <- 01)
```

```
MOV AR R1          (AR <- R1 = 01)
LSL AR AR          (AR <- LSL(AR) = 02)
LSL R2 AR          (R2 <- LSL(AR) = 04)
LSR R3 R2          (R3 <- LSR(R2) = 02)
NOT AR R3          (AR <- NOT(R3) = fd)
```

Listing 4: Memory Format for Instructions:

```
v2.0 raw
1 14 40 32 20 92 20 95
50 a6 60 62
```

3.3 Case-3

The last case tests all the (source and destination register) combinations of ADD, SUB, AND, OR operations. Meaningful instruction of the desired process:

Listing 5: Meaningful Instruction of the Desired Process:

```
LD R1 IM 0x5c      (R1 <- 5c)
LD R2 IM 0x17      (R2 <- 17)
OR R3 R1 R2        (R3 <- R1 || R2 = 0x5f)
AND AR R1 R2       (AR <- R1 && R2 = 0x14)
ADD R1 R3 AR       (R1 <- R3 + AR = 0x73)
SUB SP AR R3       (SP <- R3 - AR = 0x4b)
SUB SP AR SP       (SP <- SP - AR = 0x37)
SUB R2 AR SP       (R2 <- SP - AR = 0x23)
```

Listing 6: Memory Format for Instructions:

```
v2.0 raw
5c 14 17 15 45 56 45 42
62 74 26 83 23 83 23 85
```

4 DISCUSSION

In this project, we have designed a hardwired control unit for given architecture using the structure that we had designed in Part 4 of Project 1. We have preferred to group operations according to their behaviours then implemented them as groups and preferred to use tunnels to represent connections rather than wires to improve intelligibility of our design. We have connected all our structures to single clock generator to synchronise all

organization in one hand.

First of all, we have used the structure from Part 4 of Project 1 as our CPU, so that we have given inputs to it and got outputs from it. In addition to this structure, we have also needed a Sequence Counter to arrange clock signals.

After doing this, we have designed Fetch and Decode cycles due to the fact that we need them at the beginning of all operations. At Fetch cycle, since every address of our memory can store only 8-bits, the instruction is read and written to Instruction Register part by part in two clock cycles. At Decode cycle, we have obtained many flags to make our job easier during design of operations in following clock cycles. Firstly, since we have two types of instructions, we have divided Instruction input into parts according to given formats for both instruction types separately, e.g. Opcode. Secondly, according to 4-bit Opcode, we have obtained flags for every operation using 4:16 Decoder, e.g. if Opcode is 0000 only BRA flag is 1 others are 0. Thirdly, according to 4-bit Opcode, we have obtained 4-bit DecodedALUFunsel input considering every operation's behaviour using 16x1 MUX, e.g. DecodedALUFunsel = 0010 if Opcode is 0110 (NOT). Fourthly, we have obtained Z C N O flags separately by dividing ZCNO output. Fifthly, due to the fact that there is no direct connection between Address Register's output and ALU's input, we have obtained flags to know if a Temporary Register is need or not according to sources' and destination's locations which should be considered while using instruction without address reference. To do that we have used only 3rd bit of DestReg, SrcReg1 and SrcReg2, e.g. TempFlagDest is 1 if DestReg's 3rd bit is 0.

Finally, using all of these flags, inputs, constants, three state buffers, logical gates and plexers we implemented all operations, some of them as groups some of them individually to be able to perform their operations in following clock signals. Different inputs, different number of clock signals, different instruction types are used for every group of operations considering their behaviour carefully. We have tested our implementations for various Instruction inputs so we believe our architecture works as desired.

5 CONCLUSION

In this project, we learned lots of things about a hardwired control unit of basic computer system. We implemented many instructions in Logisim and we became more proficient in using Logisim. Everything we learned in the lesson started to make sense while we were making the designs piece by piece. After we finished the design, we tested some cases

to check if our design worked correctly, then we made some corrections. Thanks to the tests, every process about the implementation of basic central processing unit has become clearer and more understandable for us.