ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION
PROJECT 2 REPORT

PROJECT NO : 2

PROJECT DATE : 23.05.2023

GROUP NO : G71

GROUP MEMBERS:

150220901 : MOHAMAD CHAHADEH

150190068 : OZGUR SEFEROGLU

150200913 : FITNETE GUNI

SPRING 2023

Contents

1 INTRODUCTION

In this project, we implement a hardwired control unit for basic processor we designed in project1. This hardwired control unit consists of sequence counter, combinational control unit, 2 decoders and ALUSystem. Sequence counter, combinational control unit, 2 decoders recognize the instructions and create appropriate control signal for ALUSystem. Then ALUSystem performs operations such AND, OR, NOT, ADD, SUB, LSR, LSL, PUL, PSH, INC, DEC, LD, ST, MOV, BRA and BNE according to control signals.

2 INSTRUCTIONS

- OPCODE (HEX) : **0x00**

    - Symbol : AND

    - Addressing Mode : N/A

    - Description : DSTREG ← SREG1 AND SREG2, we make AND operation between source register 1 and 2 and put the result of it to destination register.

- OPCODE (HEX) : **0x01**

    - Symbol : OR

    - Addressing Mode : N/A

    - Description : DSTREG ← SREG1 OR SREG2, we make OR operation between source register 1 and 2 and put the result of it to the destination register.

- OPCODE (HEX) : **0x02**

    - Symbol : NOT

    - Addressing Mode : N/A

    - Description : DSTREG ← NOT SREG1, we take complement of source register 1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x03**

    - Symbol : ADD

    - Addressing Mode : N/A

    - Description : DSTREG ← SREG1 + SREG2, we add source register2 to source register1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x04**

- Symbol : SUB

- Addressing Mode : N/A

- Description : DSTREG ← SREG1 - SREG2, we subtract source register2 from source register1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x05**

  - Symbol : LSR

  - Addressing Mode : N/A

  - Description : DSTREG ← LSL SREG1, we make logical shift left operation to source register1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x06**

  - Symbol : LSL

  - Addressing Mode : N/A

  - Description : DSTREG ← LSR SREG1, we make logical shift right operation to source register1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x07**

  - Symbol : INC

  - Addressing Mode : N/A

  - Description : DSTREG ← SREG1 + 1, we increase source register1 by 1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x08**

  - Symbol : DEC

  - Addressing Mode : N/A

  - Description : DSTREG ← SREG1 - 1, we decrease source register1 by 1 and put the result of it to the destination register.

- OPCODE (HEX) : **0x09**

  - Symbol : BRA

  - Addressing Mode : IM

  - Description : PC ← Value, we directly put the value of IR[7-0] to the PC due to addressing mode is immediate.

- OPCODE (HEX) : **0x0A**

  - Symbol : BNE

  - Addressing Mode : IM

  - Description : IF Z = 0 THEN PC ← Value, if Z is equal to zero then we put IR[7-0] to the PC (due to its addressing mode is immediate).

- OPCODE (HEX) : **0x0B**

  - Symbol : MOV

  - Addressing Mode : N/A

  - Description : DSTREG ← SREG1, we move source register 1 to destination register.

- OPCODE (HEX) : **0x0C**

  - Symbol : LD

  - Addressing Mode : IM, D

  - Description : Rx ← Value, we put the value of M[AR] or IR[7-0] (due to its addressing mode is both D and IM) to the Rx to make the load.

- OPCODE (HEX) : **0x0D**

  - Symbol : ST

  - Addressing Mode : D

  - Description : Value ← Rx, we put Rx to the M[AR] to make the store operation.

- OPCODE (HEX) : **0x0E**

  - Symbol : PUL

  - Addressing Mode : N/A

  - Description : M[SP] ← Rx; SP ← SP - 1, we put Rx to the M[SP] and decrease SP by 1.

- OPCODE (HEX) : **0x0F**

  - Symbol : PSH

  - Addressing Mode : N/A

  - Description :SP ← SP + 1; Rx ← M[SP], firstly we increase SP by 1 and then put M[SP] to the Rx.

3 IMPLEMENTATION

3.1 Hardwired Control Unit

Our hardwired control unit has multiple modules. There are ALUSystem that we created in project1, Sequence Counter and Combinational Control Unit . ALUSystem fetches instructions from memory to the Instruction Register. A decoder with 4 bit selection inputs decodes Instruction Registers 15-12 bits which is the OpCode part and sends output signals to combinational control unit. Also there is a sequence counter, which count from 0 to 7, in order to perform operation at appropriate clock signals. A decoder with 3 bit selection inputs, decodes T coming from sequence counter and sends T0, T1, T2, ..., T7 signals to combinational control unit. Inside combinational control unit, all of the control signals that will be sent to ALUSystem is determined according to T, OpCode, SRCREG1, SRCREG2, DESTREG signals. Lastly, outputs of combinational circuit is sent to ALUSystem to perform instructions.
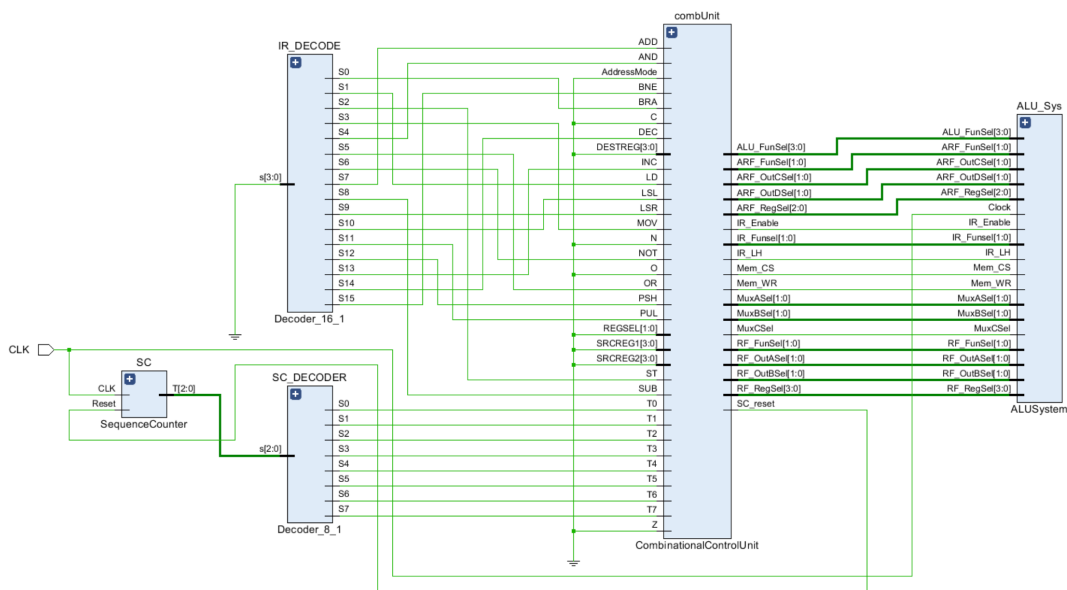

Figure 1: Hardwired Control Unit

3.2 Sequence Counter

Sequence counter generates timing signals which will control clock cycle for micro-operations. Timing signals will be sent to combinational circuit and used while arranging the control signals for ALUSystem. When an instruction is completed. Reset signal is sent to sequence counter and timing signal reset to T0.
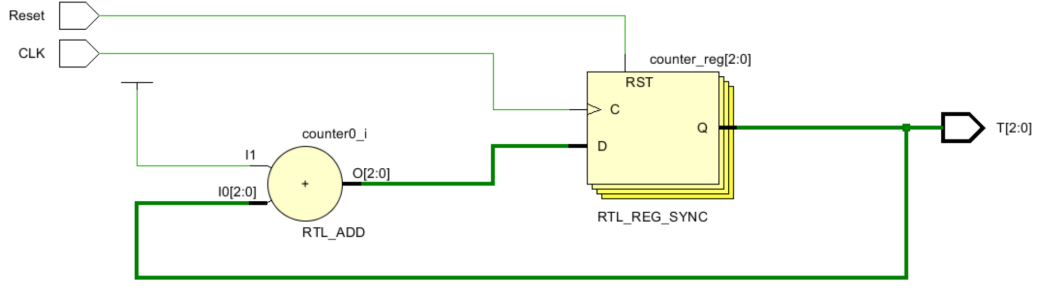
4

Figure 2: Sequence Counter

3.3 Combinational Control Unit

Combinational control unit takes T signals, OpCode signal from decoder, RSEL, DSTREG, SREG1, SREG2, AddressMode from instruction register of ALUSystem and Z, C, N, O from ZCNO flag register of ALUSystem as inputs. Then according to these inputs , it sets control signals for corresponding instructions and sends them to ALUSystem.

3.3.1 ALU operations (AND, OR, NOT, ADD, SUB, LSR, LSL)

These instructions will be completed in 1 clock cycle, but SUB instruction will be completed in 3 clock cycles. In these operations, we make destination register enable thanks to DESTREG. Then, we sent the value in the source register to the appropriate outputs. We routed these outputs to the A and B inputs of the ALU according to each operation. When DESTREG[2] is 0, we set ARF-FunSel to 10. When DESTREG[2] is 1, we set RF-FunSel to 10. Finally, we reset the clock cycle thanks to SC-reset.

3.3.2 INC, DEC

INC and DEC operation will be completed in two clock cycles. When the fist clock cycle(T2) comes, the value of SRCREG is send to DESTREG. When the second clock cycle(T3) comes, the value of DESTREG is increased. In these operations, we make destination register enable thanks to DESTREG. Then, we sent the value in the source register to the appropriate output. For loading, we set the funsel of register to 10 (if DESTREG[2]=0, ARF-FunSel, else if DESTREG[2]=1, RF-FunSel). At the next clock signal, we set the funsel of register to 00(Decrement) or 10(Increment) for decrement or increment. Finally, we reset the clock cycle thanks to SC-reset. Also, we carried the data after incrementation or decrementation to ALU in order to check if flags in ZCNO register changes.

5

3.3.3 BRA, BNE

BRA and BNE operations is completed in 1 clock cycle. In these instructions, we set MuxBSel to 00 at time T2 so that IROut(0-7) comes as Include bits of Address Register File. Moreover, we set ARFFunSel to 10, and ARFRegSel to 011 in order to load to PC at time T2. we make other registers disable. Finally, we reset the clock cycle thanks to SC-reset. Even though both instructions perform same operations, there is a difference. For BNE instruction, operations explained in the paragraph are performed only when Z flag inside ZCNO register is 0. Otherwise BNE instruction does nothing.

3.3.4 MOV

MOV operation is completed in 1 clock cycle. In this operaiton, we make destination register enable thanks to DESTREG. Then, we sent the value in the source register to the appropriate output. For loading, we set the funsel of register to 10(if DESTREG[2s=0, ARF-FunSel,else if. When SRCREG[2] is 0, we set MuxASel or MuxBSel to 10, and 11 respectively. On the other condition, we set MuxASel or MuxBSel to 11, and 11 respectively. We set FunSel of register file or Address register file to load mode according to DESTREG. Finally, we reset the clock cycle thanks to SC-reset.

3.3.5 LD

LD operation is completed in 1 clock cycle. In this operation, when address mode of IR is 0, we set ARF-OutDSel to 10 so that AR comes as Address bits of Memory. In addition, we set Mem-WR to 0, and Mem-CS to 0 for reading mode of memory. We set MuxASel to 01 at time T2 so that Memory Output comes as Include bits of Register File. When address mode of IR is 1, we set MuxASel to 00 at time T2 so that IROut(0-7) comes as Include bits of Register File. In both situations, We must set RF-FunSel to 10 for the loading. Also we choose the register thanks to REGSEL bits of IR. Finally, we reset the clock cycle thanks to SC-reset.

3.3.6 ST

ST operation is completed in 1 clock cycle. In this operation, We did all the operations at the time of T2. We sent the value of the selected register from the OutB of the register file and passed it over to the ALU to make it go through the memory. Thus, we set RF-OutBSel to REGSEL bits of IR, and ALU-FunSel to 0001. In order to writing the incoming value to M[AR], we set ARF-OutDSel to 10. Moreover, we set Mem-WR to 1, and Mem-CS to 0 for writing mode of memory.Finally, we reset the clock cycle thanks to SC-reset.

3.3.7 PUL

This instruction will be completed in 2 clock cycles. At T2, we increment the SP register. For incrementation, we arrange ARFRegSel as 110 and ARFFunSel as 01. Since we don't have any work to do with IR, RF and memory, we disable them by giving IREnable 0, RFRegSel 1111 and MemCS 1. Also reset input for sequence counter is 0, because the instruction is not completed. For T3, we give ARFOutDSel 11 because SP needs to be given as address to memory, rMemWR 0 and rMemCS to be able to read from memory. Then we arrange RFRegSel and ARFRegSel according to DESTREG. If DESTREG[2] is 1, it means that destination register is in RF. Thus we give ARFRegSel 111, RFFunSel 10 (because we will load), MuxASel 01 (because data coming from memory output) and defined RFRegSel according to DESTREG[1:0]. If DESTREG[2] is 0, it means that destination register is in ARF. Thus we give RFRegSel 1111, ARFFunSel 10 (because we will load), MuxBSel 10 (because data coming from memory output)and defined ARFRegSel according to DESTREG[1:0]. Since the instruction will be completed at the end of T3, we give SCReset 1.

3.3.8 PSH

PSH instruction can be completed in 1 cycle. At T2, we make MemCS 0 and MemWR 1 to write the data to memory, ARFOutDSel 11 to give SP to memory as address, ARFRegSel 110 and ARFFunSel 00 to increment the SP register. Since giving SP from ARF to memory and incrementing SP will be at the same cycle, the address given to the memory will be SP before the incrementation. Also, if SRCREG1 is a register inside ARF, we give MuxCSel 0, ARFOutCSel SRCREG1[1:0] and ALUFunSel 0000, because data needs to be carry from ARF to MuxC, from MuxC to ALU and from ALU to memory. if SRCREG1 is a register inside RF, we give MuxCSel 1, RFOutASel SRCREG1[1:0] and ALUFunSel 0000, because data needs to be carry from RF to MuxC, from MuxC to ALU and from ALU to memory.

3.3.9 RESET

When our basic computer is initiated, all of our registers are empty and we need to reset them. At the initiation, sequence counter starts counting from T7. That's because we use 1 clock cycle in order to reset all of the registers. Then it becomes T0. Since we don't reach T7 in any instruction in our basic computer, sequence counter sends T7 signal only 1 time.

3.3.10 FETCH

Fetch operation is completed in 2 clock cycles because the data we take from memory is 8 bit but our instructions are consist of 16 bits. At T0 MemCS and MemWR are set to 0 in order to read from memory. IR is enabled and IRLH 1 becuase we load the data from memory to IR(7-0). Then we increment the PC by 1. At T1, MemCS and MemWR are set to 0 and IR is enabled again but this time IRLH is 0 because we load the data from memory to IR(15-8) and complete the fetch operation. Each time we load data from memory to instruction register, we increment PC by 1. After fetch operation, data inside instruction register is decoded and sent to combinational control unit.

4 DISCUSSION AND RESULTS

While implementing project, first of all, we design the SequenceCounter module for clock cycle. Then, we decode the result of SequenceCounter with 1 to 8 decoder. Also, we use a 1 to 16 decoder in order to decode the IR. Secondly, we design the CombinationalControlUnit module. This module is responsible for determining the inputs of the ALU system, which we made in the previous project, according to the clock signal and OP-CODE. Finally, we design the HardwiredControlUnit module. In this module, we make all connections between the ALU system, SequenceCounter, CombinationalControlUnit and decoders.

Unfortunately, I was not able to get the correct testbench result, I tried everything, Tested each component individually, rewrote some implementations, and changed the memory, All in vain. I think the problem was from the ALUSystem implementation of the previous project which had some fault and so these faults showed up again in this project as well.
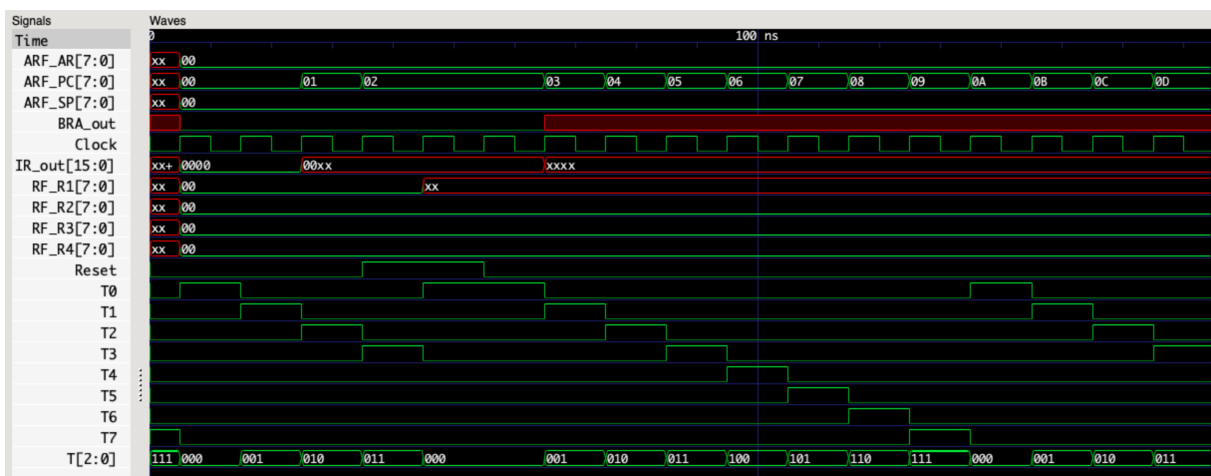

Figure 3: Testbench of the final implementation

5 CONCLUSION

In this Project, we have learnt how a basic computer works and performs various instructions. Even though we weren't able to get the correct output at the end, we still learned how to implement important Computer components such as IR, RF, ARF, ALU unit and so on. We also learned how to access and manipulate memory and do operations like read and write from and to it.