ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E
COMPUTER ORGANIZATION
PROJECT 2 REPORT

CRN : 21334 - 21336

LECTURER : Gökhan İnce - Mustafa Kamaşak

GROUP MEMBERS:

150200060 : Ömer Yıldız

150190052 : Utku Kadir Somer

SPRING 2024

Contents

1 INTRODUCTION

In this project, we designed a hardwired control unit for the architecture developed in Part 4 of Project 1. Our goal was to create a control unit capable of managing the execution of a set of instructions stored in memory, operating within a framework that divides these instructions into two categories: those with address references and those without. This design aims to ensure synchronous and efficient execution of operations within a computer system.

Our control unit interfaces with the modules developed in the previous project, managing inputs such as selection signals, function selections, and ALU inputs. By implementing a timing unit, we ensured that all components of our system function correctly and in harmony.

1.1 Task Distributions

All the tasks, including fetching, decoding, writing the memory file, and designing and implementing the executions for instructions with and without address references, were done in collaboration together. We worked as a team, ensuring collaborative effort and shared insights across all tasks.

2 MATERIALS AND METHODS

2.1 Cycles

- **T0 (Fetch cycle 1)**: `IR(7-0) ← M[PC], PC ← PC + 1`
  Control: `Write = 1, IR LH = 0, Mem CS = 0, Mem WR = 0, ARF FunSel = 1, ARF RegSel = 011, ARF OutDSel = 0`

- **T1 (Fetch cycle 2)**: `IR(15-8) ← M[PC], PC ← PC + 1`
  Control: `Write = 1, IR LH = 1, Mem CS = 0, Mem WR = 0, ARF FunSel = 1, ARF RegSel = 011, ARF OutDSel = 0`

- **T2 (Decode cycle)**: `D0...D33 ← IR(15-10), AR ← IR(7-0), REGSEL ← IR(9-8), DESTREG ← IR(8-6), SRCREG1 ← IR(5-3), SRCREG2 ← IR(2-0) S← IR(9)`
  Control: `MuxBSel = 11, ARF RegSel = 101, ARF FunSel = 100`

- **D0 T3 (Load from PC)**: `S1 ← PC`
  Control: `MuxASel = 01, ARF OutCSel = 00, RF ScrSel = 0111, RF FunSel = 010`

- **D0 T4**: S2 ← IR(7-0)
  Control: MuxASel = 11, RF ScrSel = 1011, RF FunSel = 010

- **D0 T5**: PC ← S1 + S2
  Control: Alu FunSel = 00100, RF OutASel = 100, RF OutBSel = 101 ,ARF RegSel = 011, ARF FunSel = 010, MuxBSel = 00

- **D1 T3 Z (Load from PC)**: S1 ← PC
  Control: MuxASel = 01, ARF OutCSel = 00, RF ScrSel = 0111, RF FunSel = 010

- **D1 T4 Z**: S2 ← IR(7-0)
  Control: MuxASel = 11, RF ScrSel = 1011, RF FunSel = 010

- **D1 T5 Z**: PC ← S1 + S2
  Control: Alu FunSel = 00100, RF OutASel = 100, RF OutBSel = 101 ,ARF RegSel = 011, ARF FunSel = 010, MuxBSel = 00

- **D2 T3 Z' (Load from PC)**: S1 ← PC
  Control: MuxASel = 01, ARF OutCSel = 00, RF ScrSel = 0111, RF FunSel = 010

- **D2 T4 Z'**: S2 ← IR(7-0)
  Control: MuxASel = 11, RF ScrSel = 1011, RF FunSel = 010

- **D2 T5 Z'**: PC ← S1 + S2
  Control: Alu FunSel = 00100, RF OutASel = 100, RF OutBSel = 101 ,ARF RegSel = 011, ARF FunSel = 010, MuxBSel = 00

- **D3 T3**: Rx ← M[SP], SP ← SP + 1
  Control: RF FunSel = 101, ARF OutDSel = 11, MemCS = 0, MemWR = 0 ,RF RegSel = SEL, MuxASel = 10

- **D3 T4**: SP ← SP + 1
  Control: ARF FunSel = 001, ARF RegSel = 110

- **D4 T3**: M[SP] ← Rx, SP ← SP - 1
  Control: ARF FunSel = 000, ARF RegSel = 110, RF OutASEL = SEL, ALU FUNSEL = 00000, MemCS = 0, Mem WR = 1, ARF OUTDSEL = 11

- **D4 T4**: SP ← SP - 1
  Control: ARF FunSel = 000, ARF RegSel = 110

  For the next 7 cycle (D5 - D11) there are 4 possibilities.

- SRC is RF, Dest is RF **D5 T3**: `ALUOUT ← RF`
  Control: `RF OUTASEL = SCREG1, ALU FUNSEL = 10001`

- **D5 T4**: `RF ← ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC is RF, Dest is ARF **D5 T3**: `ALUOUT ← RF`
  Control: `RF OUTASEL = SCREG1, ALU FUNSEL = 10001`

- **D5 T4**: `ARF ← ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX BSEL = 00`

- SRC is ARF, Dest is RF **D5 T3**: `S1 ← ARF`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010`

- **D5 T4**: `RF ← S1`
  Control: `RF OUTASEL = 100, ALU FUNSEL = 10001, RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC is ARF, Dest is ARF **D5 T3**: `S1 ← ARF`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010`

- **D5 T4**: `ARF ← S1`
  Control: `RF OUTASEL = 100, ALU FUNSEL = 10001, RF REGSEL = DESTREG, RF FUNSEL = 010, MUX BSEL = 00`

  For the D6 - D11 instructions have the same logic but only the ALU Funsel values are changing according to instruction.

  There are multiple possibilities for instructions between D12 - D16 and D21 - D29. SRC1 is RF, SRC2 is RF, DEST is RF

- SRC1 is RF, SRC2 is RF, DEST is RF **D12 T3**: `OUTA ← Rx1, OUTB ← Rx2, ALUOUT ← OUTA AND OUTB`
  Control: `RF OUTASEL = SCREG1, RF OUTBSEL = SCREG2, ALU FUNSEL = 10111`

- **D12 T4**: `RF ← ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC1 is RF, SRC2 is RF, DEST is ARF **D12 T3**: `OUTA ← Rx1, OUTB ← Rx2, ALUOUT ← OUTA AND OUTB`
  Control: `RF OUTASEL = SCREG1, RF OUTBSEL = SCREG2, ALU FUNSEL = 10111`

- **D12 T4**: `ARF` ← `ALUOUT`
  Control: `ARF REGSEL = DESTREG, ARF FUNSEL = 010, MUX BSEL = 00`

- SRC1 is ARF, SRC2 is RF, DEST is RF **D12 T3**: `S1` ← `ARF, OUTB` ← `Rx2`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010, RF OUTBSEL = SCREG2`

- **D12 T4**: `ALUOUT` ← `OUTA AND OUTB, OUTA` ← `S1`
  Control: `ALU FUNSEL = 10111, RF OUTASEL = 100`

- **D12 T5**: `RF` ← `ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC1 is RF, SRC2 is ARF, DEST is RF **D12 T3**: `S2` ← `ARF, OUTA` ← `Rx1`
  Control: `ARF OUTCSEL = SCREG2, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010, RF OUTASEL = SCREG1`

- **D12 T4**: `ALUOUT` ← `OUTA AND OUTB, OUTB` ← `S2`
  Control: `ALU FUNSEL = 10111, RF OUTBSEL = 101`

- **D12 T5**: `RF` ← `ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC1 is RF, SRC2 is ARF, DEST is ARF **D12 T3**: `S2` ← `ARF, OUTA` ← `Rx1`
  Control: `ARF OUTCSEL = SCREG2, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010, RF OUTASEL = SCREG1`

- **D12 T4**: `ALUOUT` ← `OUTA AND OUTB, OUTB` ← `S2`
  Control: `ALU FUNSEL = 10111, RF OUTBSEL = 101`

- **D12 T5**: `ARF` ← `ALUOUT`
  Control: `ARF REGSEL = DESTREG, ARF FUNSEL = 010, MUX BSEL = 00`

- SRC1 is ARF, SRC2 is RF, DEST is ARF **D12 T3**: `S1` ← `ARF, OUTB` ← `Rx2`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010, RF OUTBSEL = SCREG2`

- **D12 T4**: `ALUOUT` ← `OUTA AND OUTB, OUTA` ← `S1`
  Control: `ALU FUNSEL = 10111, RF OUTASEL = 100`

- **D12 T5**: `ARF` ← `ALUOUT`
  Control: `ARF REGSEL = DESTREG, ARF FUNSEL = 010, MUX BSEL = 00`

- SRC1 is ARF, SRC2 is ARF, DEST is RF **D12 T3**: `S1 ← ARF`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010`

- **D12 T4**: `S2 ← ARF`
  Control: `ARF OUTCSEL = SCREG2, MUX ASEL = 01, RF SCRSEL = 1011, RF FUNSEL = 010`

- **D12 T5**: `OUTA ← S1, OUTB ← S2, ALUOUT ← OUTA AND OUTB`
  Control: `RF OUTBSEL = 101, RF OUTASEL = 100, ALU FUNSEL = 10111`

- **D12 T6**: `RF ← ALUOUT`
  Control: `RF REGSEL = DESTREG, RF FUNSEL = 010, MUX ASEL = 00`

- SRC1 is ARF, SRC2 is ARF, DEST is ARF **D12 T3**: `S1 ← ARF`
  Control: `ARF OUTCSEL = SCREG1, MUX ASEL = 01, RF SCRSEL = 0111, RF FUNSEL = 010`

- **D12 T4**: `S2 ← ARF`
  Control: `ARF OUTCSEL = SCREG2, MUX ASEL = 01, RF SCRSEL = 1011, RF FUNSEL = 010`

- **D12 T5**: `OUTA ← S1, OUTB ← S2, ALUOUT ← OUTA AND OUTB`
  Control: `RF OUTBSEL = 101, RF OUTASEL = 100, ALU FUNSEL = 10111`

- **D12 T6**: `ARF ← ALUOUT`
  Control: `ARF REGSEL = DESTREG, ARF FUNSEL = 010, MUX BSEL = 00`

  For the D12-D16 and D21-D29 instructions are in the same logic but only ALU Funsel value changes according to instruction. From D24 to D29, S value is equal to 1, that's why flags will change.

- **D17 T3**: `DestReg[15:8] ← IR [7-0]`
  Control: `MUXASEL = 11, RF REGSEL = SEL, RF FUNSEL = 110`

- **D18 T3**: `Rx ← M[AR], AR ← AR + 1`
  Control: `ARF OUTDSEL = 10, RF REGSEL = SEL, RF FUNSEL = 101, MemCS = 0, MemWR = 0, MUX ASEL = 10, ARF FUNSEL = 001, ARF REGSEL = 101`

- **D18 T4**: `Rx ← M[AR]`
  Control: `ARF OUTDSEL = 10, RF REGSEL = SEL, RF FUNSEL = 110, MemCS = 0, MemWR = 0, MUX ASEL = 10`

- **D19 T3**: M[AR] ← Rx, AR ← AR + 1
  Control: RF OUTASEL = SEL, ALU FUNSEL = 00000, MEMCS = 0, MEMWR = 1, ARF OUTDSEL = 10, MUX CSEL = 0, ARF FUNSEL = 001, ARF REGSEL = 101

- **D19 T4**: M[AR] ← Rx
  Control: RF OUTASEL = SEL, ALU FUNSEL = 10000, MEMCS = 0, MEMWR = 1, ARF OUTDSEL = 10, MUX CSEL = 1

- **D20 T3**: DestReg[7:0] ← IR [7-0]
  Control: MUXASEL = 11, RF REGSEL = SEL, RF FUNSEL = 101

- **D30 T3**: S1 ← PC
  Control: ARF OUTCSEL = 00, ARF SCRSEL = 0111, MUX ASEL = 01, ARF FUNSEL = 010

- **D30 T4**: M[SP] ← S1, SP ← SP + 1
  Control: OUT ASEL = 100, ALU FUNSEL = 10000, OUT DSEL = 11, MUX CSEL = 0, MEMWR = 1, MEMCS = 0, ARF REGSEL = 110, ARF FUNSEL = 001

- **D30 T5**: M[SP] ← S1
  Control: OUT ASEL = 100, ALU FUNSEL = 10000, OUT DSEL = 11, MUX CSEL = 1, MEMWR = 1, MEMCS = 0

- **D30 T6**: PC ← Rx
  Control: OUT ASEL = SEL, ALU FUNSEL = 10000, MUX BSEL = 00, ARF FUNSEL = 010, ARF REGSEL = 011

- **D31 T3**: PC ← M[SP]
  Control: OUT DSEL = 11, MUX BSEL = 10, ARF REGSEL = 011, ARF FUNSEL = 010

- **D31 T4**: SP ← SP + 1
  Control: ARF REGSEL = 110, ARF FUNSEL = 001

- **D31 T5**: PC ← M[SP]
  Control: OUT DSEL = 11, MUX BSEL = 10, ARF REGSEL = 011, ARF FUNSEL = 010

- **D32 T3**: Rx ← IR[7:0]
  Control: MUX ASEL = 11, RF REGSEL = SEL, RF FUNSEL = 010

- **D33 T3**: S1 ← AR
  Control: ARF OUTCSEL = 10, RF SCRSEL = 0111, MUX ASEL = 01, RF FUNSEL = 010

- **D33 T4**: `S2 ← IR[7:0]`
  Control: `MUX ASEL = 11, RF SCRSEL = SEL, RF FUNSEL = 010`

- **D33 T5**: `AR ← S1 + S2`
  Control: `ALU FUNSEL = 10100, RF OUTASEL = 100, RF OUTBSEL = 101, ARF REGSEL = 101, ARF FUNSEL = 010, MUX BSEL = 00`

- **D33 T6**: `M[AR] ← Rx`
  Control: `ARF OUTDSEL = 10, MEMCS = 0, MEMWR = 1, RF OUTASEL = SEL, ALU FUNSEL = 10000, MUX CSEL = 0`

2.2 General layout

The overall layout of our hardwired control unit was designed to ensure efficient processing and execution of instructions. We organized the control signals systematically to manage the flow of data and control information throughout the system. This structured approach facilitated easier debugging and testing of individual components.

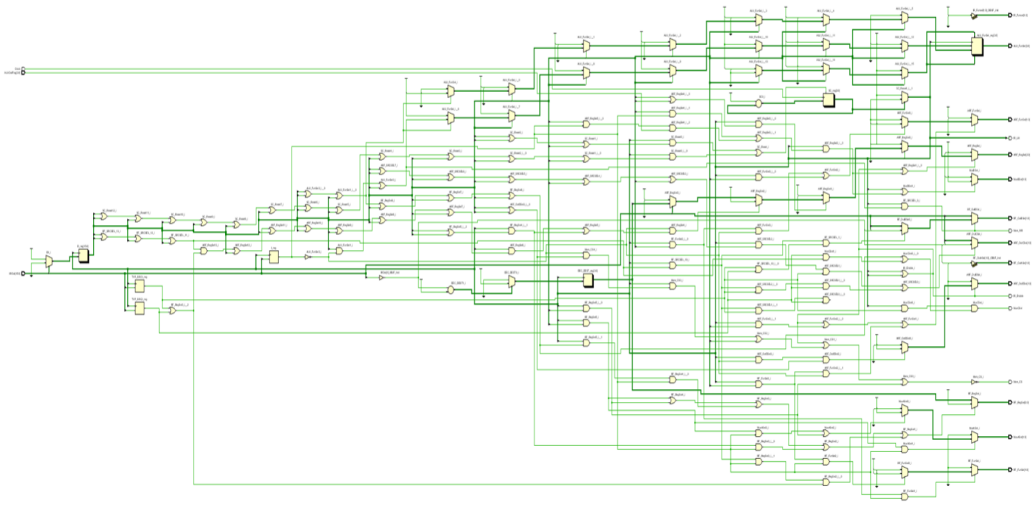

Figure 1: Control Unit Design

2.3 Timing Unit

The timing unit is a fundamental component of our control unit design, ensuring that all operations within the system are executed in a precise and orderly manner. The timing

unit is responsible for generating control signals at specific intervals to synchronize the fetching, decoding, and execution phases of the instruction cycle.

In accordance with the project requirements, our timing unit initiates the instruction cycle by fetching the instruction in two clock cycles. During the first clock cycle (T0), the least significant byte (LSB) of the instruction is loaded from memory address A into IR(7-0). In the second clock cycle (T1), the most significant byte (MSB) of the instruction is loaded from memory address A+1 into IR(15-8). After these two cycles, the instruction is fully loaded into the Instruction Register (IR), and the execution phase begins.

By dividing the instruction fetching process into these two cycles, we ensure that our system can handle instructions stored in little-endian format, as specified in the project guidelines. The timing unit's precise control over these cycles is crucial for maintaining synchronization and efficient processing within the control unit.

3 FETCHING AND DECODING

The fetching and decoding phases are critical to the operation of our control unit. The fetching phase involves retrieving the instruction from memory and loading it into the Instruction Register (IR) in two cycles, as detailed in the Timing Unit section.

Once the instruction is fetched, the decoding phase (T2) begins. During this phase, the control unit interprets the opcode and other relevant fields within the instruction to determine the specific operation to be performed. This involves decoding the 6-bit OPCODE, the 2-bit RSEL, and the 8-bit ADDRESS for instructions with address references, or decoding the 6-bit OPCODE, the 1-bit S, the 3-bit DSTREG, the 3-bit SREG1, and the 3-bit SREG2 for instructions without address references.

4 RESULTS

In this section, we present the results of our hardwired control unit design, highlighting the successful implementation and validation of its functionality.

4.1 Execution of Instructions

Our control unit was tested using a series of instructions as described in the project requirements. We implemented both instructions with address references and those without. The following are some of the key results observed during the execution of these instructions:

- **Fetch Cycle**:

  - T0 (Fetch cycle 1): Successfully loaded the LSB of the instruction from memory to IR(7-0) and incremented the PC by 1.
  - T1 (Fetch cycle 2): Successfully loaded the MSB of the instruction from memory to IR(15-8) and incremented the PC by 1.

- **Decode and Execute Cycles**:
  - T2 (Decode cycle): Correctly decoded the opcode and other fields, setting up the appropriate control signals for execution.
  - Execution cycles (T3 and beyond): Correctly performed the operations specified by the instructions, such as data transfer, arithmetic, and logical operations.

4.2 Timing and Synchronization

Our timing unit successfully synchronized the various components of the control unit, ensuring that each instruction was fetched, decoded, and executed in the correct sequence. The precise control over the clock cycles allowed for smooth transitions between the stages of instruction processing.

4.3 Memory Access

For instructions involving memory access, such as LDR (load) and STR (store), our control unit correctly managed the memory addresses and data transfers. The memory operations were performed efficiently, with the control signals accurately directing the data flow between the memory and the registers.

4.4 Control Signal Validation

Throughout the testing phase, we closely monitored the control signals generated by our control unit. The signals were validated to ensure they matched the expected values for each stage of the instruction cycle. This validation confirmed the correctness of our design and its ability to handle a variety of instruction types.

4.5 Summary of Observations

- The control unit successfully managed the fetching, decoding, and execution of instructions, demonstrating the effectiveness of our design.

- The timing unit played a crucial role in maintaining synchronization, ensuring that all components operated harmoniously.
- Memory access operations were handled efficiently, with accurate control over data transfers.
- Control signals were validated and found to be correct, reinforcing the reliability of our design.

5 DISCUSSION

The design and implementation of our hardwired control unit posed several challenges, which we successfully addressed through a collaborative effort. One of the primary challenges was ensuring precise synchronization across the various stages of instruction execution, from fetching and decoding to actual execution and memory access.

Our control unit design adheres to the project specifications by categorizing instructions into those with and without address references. This distinction allowed us to tailor our control signals and data paths accordingly, ensuring efficient handling of each instruction type. For instance, instructions with address references, such as LDR and STR, required careful management of memory addresses and data registers, while instructions without address references, like AND, ORR, and XOR, focused on register operations.

Furthermore, the project required that the instruction register be filled in two clock cycles due to the 8-bit output constraint of the RAM. Our design effectively accommodates this by splitting the fetching process into two distinct cycles, ensuring that both the LSB and MSB of the instruction are correctly loaded into the IR.

Through rigorous testing and simulation, we verified the correctness and efficiency of our control unit design. The control signals generated by our timing unit and decoding logic were validated to ensure proper execution of all instruction types, as outlined in the project guidelines.

6 CONCLUSION

In conclusion, we have successfully designed and implemented a hardwired control unit that meets the specifications outlined in Project 2. Our control unit efficiently manages the fetching, decoding, and execution of instructions, ensuring synchronous and reliable operation of the computer system. The timing unit's precise control over instruction cycles and the effective handling of different instruction types are key highlights of our design.

The collaborative effort as a team was instrumental in overcoming the challenges encountered during this project. The insights gained from this experience have enhanced our understanding of control unit design and its critical role in computer architecture. Future work could explore further optimizations in control signal generation and the incorporation of additional instruction types to expand the system's capabilities.

Overall, our project demonstrates a successful application of hardwired control unit principles, resulting in a robust and efficient design that aligns with the project requirements.