# Instruction Sequencer FSM

By Sidhardh Burre

CS/ECE 2330 Digital Logic Design

March 20, 2021

# Table of Contents

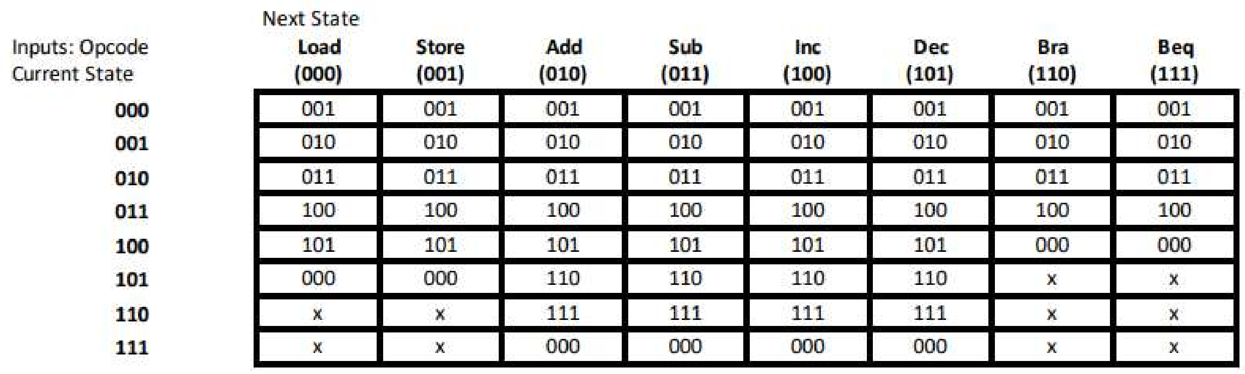
## Contents

# Table of Figures

## 1 Problem Statement

Design a circuit to generate the instruction sequencing signals used to ensure the proper sequencing of the control signals generation for instruction fetch and execute, completing the control unit design.

The circuit in question is a Finite State Machine (FSM) with 3 inputs (the 3-bit opcode) and requires 3 sequential elements to store the current state. A state transition table for the circuit is shown in Table 1, each 'x' indicates a "don't care". Additionally, a state transition diagram is shown in Figure 1.



| Inputs: Opcode | Next State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Current State | **Load (000)** | **Store (001)** | **Add (010)** | **Sub (011)** | **Inc (100)** | **Dec (101)** | **Bra (110)** | **Beq (111)** |
| **000** | 001 | 001 | 001 | 001 | 001 | 001 | 001 | 001 |
| **001** | 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 |
| **010** | 011 | 011 | 011 | 011 | 011 | 011 | 011 | 011 |
| **011** | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| **100** | 101 | 101 | 101 | 101 | 101 | 101 | 000 | 000 |
| **101** | 000 | 000 | 110 | 110 | 110 | 110 | x | x |
| **110** | x | x | 111 | 111 | 111 | 111 | x | x |
| **111** | x | x | 000 | 000 | 000 | 000 | x | x |

**Table 1: Simple CPU Sequencer FSM State Transition Table**

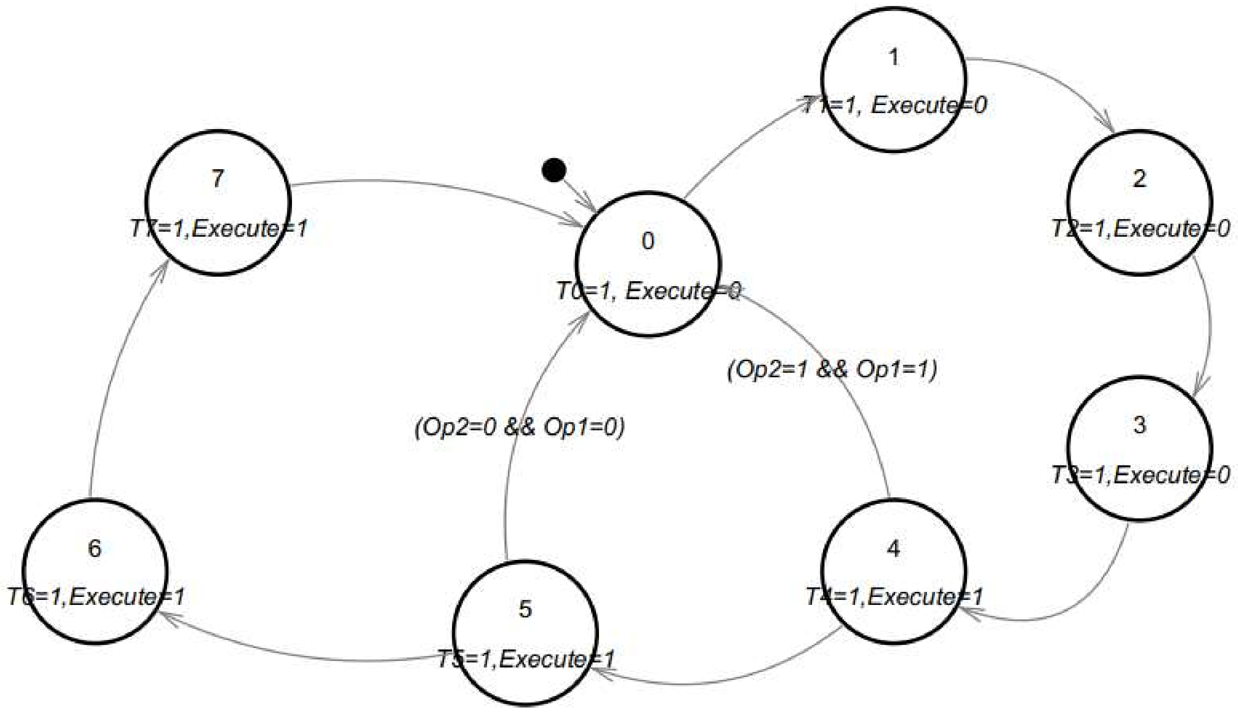


**Figure 1: State Transition Diagram**

The purpose of this circuit is to generate the 8 timing pulses that are used to sequence the instruction fetch and execute operations. A typical set of outputs is shown in Figure 2, for an instruction that requires 8 steps to complete. The output table for the timing pulses is shown in Table 2 for 8 instructions.

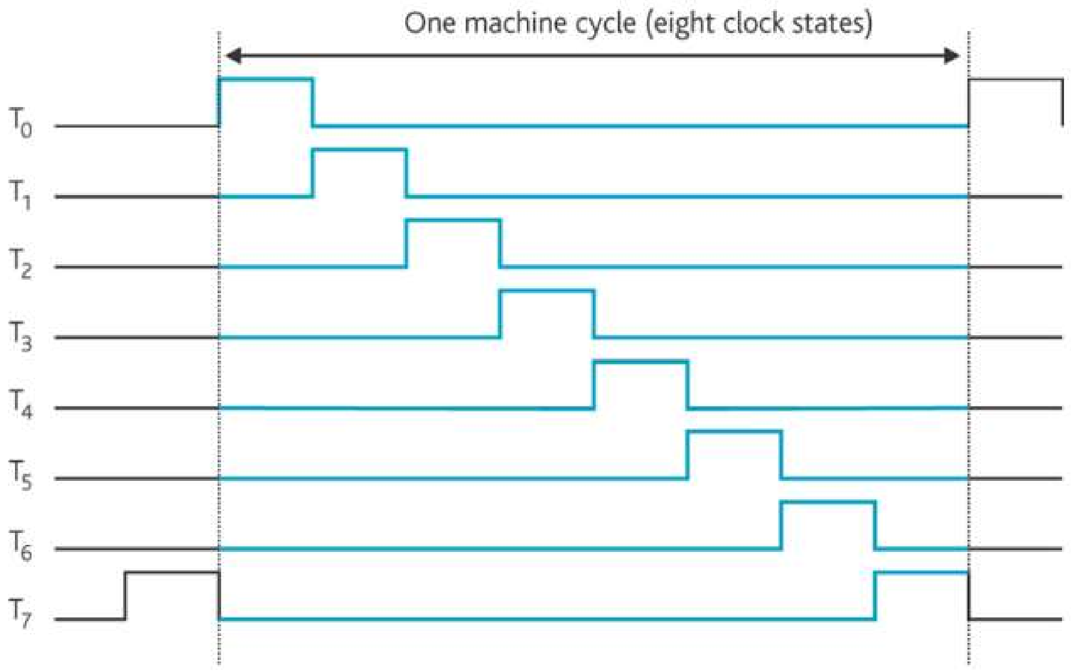
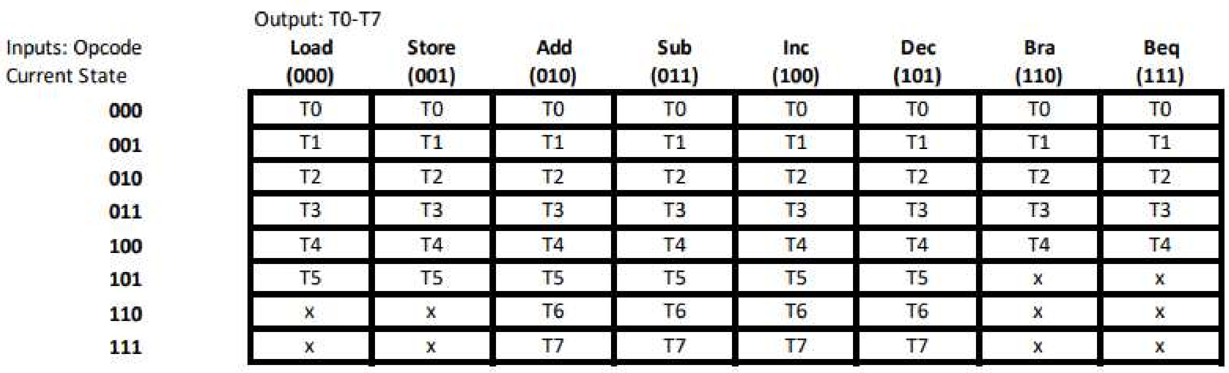


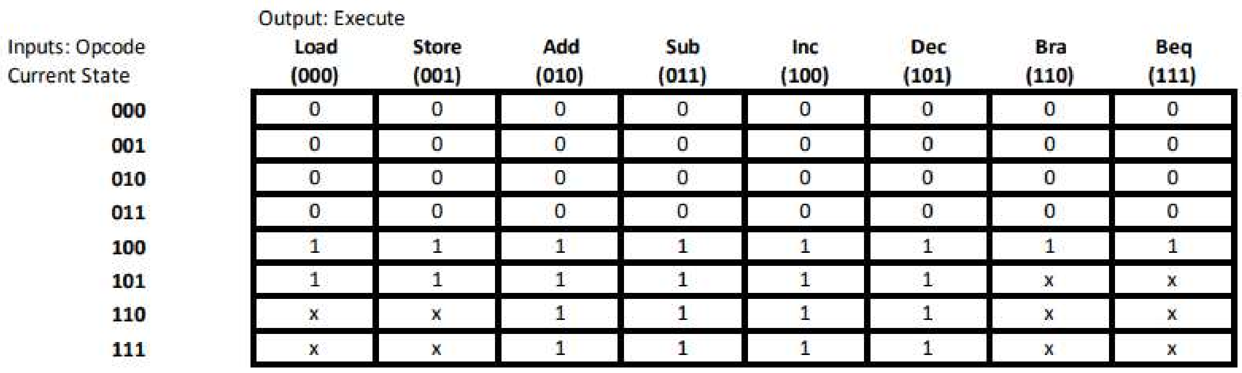
**Figure 2: Timing Pulses for One Instruction Cycle**



| Inputs: Opcode<br>Current State | Output: T0-T7<br>Load<br>(000) | Store<br>(001) | Add<br>(010) | Sub<br>(011) | Inc<br>(100) | Dec<br>(101) | Bra<br>(110) | Beq<br>(111) |
|---|---|---|---|---|---|---|---|---|
| **000** | T0 | T0 | T0 | T0 | T0 | T0 | T0 | T0 |
| **001** | T1 | T1 | T1 | T1 | T1 | T1 | T1 | T1 |
| **010** | T2 | T2 | T2 | T2 | T2 | T2 | T2 | T2 |
| **011** | T3 | T3 | T3 | T3 | T3 | T3 | T3 | T3 |
| **100** | T4 | T4 | T4 | T4 | T4 | T4 | T4 | T4 |
| **101** | T5 | T5 | T5 | T5 | T5 | T5 | x | x |
| **110** | x | x | T6 | T6 | T6 | T6 | x | x |
| **111** | x | x | T7 | T7 | T7 | T7 | x | x |

**Table 2: Output Table for Timing Pulses**

Additionally, the circuit must generate a signal to indicate when an instruction is being fetched and when it is being executed. The output table for Execute is shown in Table 3.

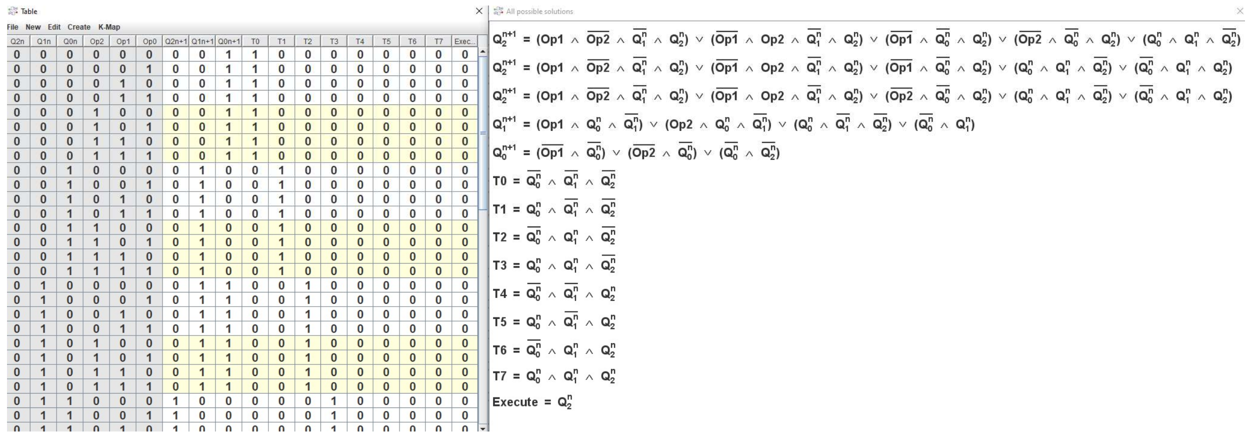


| Inputs: Opcode<br>Current State | Output: Execute<br>Load<br>(000) | Store<br>(001) | Add<br>(010) | Sub<br>(011) | Inc<br>(100) | Dec<br>(101) | Bra<br>(110) | Beq<br>(111) |
|---|---|---|---|---|---|---|---|---|
| **000** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **001** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **010** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **011** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **100** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **101** | 1 | 1 | 1 | 1 | 1 | 1 | x | x |
| **110** | x | x | 1 | 1 | 1 | 1 | x | x |
| **111** | x | x | 1 | 1 | 1 | 1 | x | x |

**Table 3: Output Table for Execute**

## 2 Analytical Design

The truth table for the Instruction Sequencer FSM is shown in Figure 3 along with the logical equations that allow the input to be converted into the corresponding signal.



Table

File New Edit Create K-Map

| Q2n | Q1n | Q0n | Op2 | Op1 | Op0 | Q2n+1 | Q1n+1 | Q0n+1 | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | Exec... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

All possible solutions

$$Q_2^{n+1} = (Op1 \wedge \overline{Op2} \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op1} \wedge Op2 \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op1} \wedge \overline{Q_0^n} \wedge Q_2^n) \vee (\overline{Op2} \wedge \overline{Q_0^n} \wedge Q_2^n) \vee (Q_0^n \wedge Q_1^n \wedge \overline{Q_2^n})$$

$$Q_2^{n+1} = (Op1 \wedge \overline{Op2} \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op1} \wedge Op2 \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op1} \wedge \overline{Q_0^n} \wedge Q_2^n) \vee (Q_0^n \wedge Q_1^n \wedge \overline{Q_2^n}) \vee (\overline{Q_0^n} \wedge Q_1^n \wedge Q_2^n)$$

$$Q_2^{n+1} = (Op1 \wedge \overline{Op2} \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op1} \wedge Op2 \wedge \overline{Q_1^n} \wedge Q_2^n) \vee (\overline{Op2} \wedge \overline{Q_0^n} \wedge Q_2^n) \vee (Q_0^n \wedge Q_1^n \wedge \overline{Q_2^n}) \vee (\overline{Q_0^n} \wedge Q_1^n \wedge Q_2^n)$$

$$Q_1^{n+1} = (Op1 \wedge Q_0^n \wedge \overline{Q_1^n}) \vee (Op2 \wedge Q_0^n \wedge \overline{Q_1^n}) \vee (Q_0^n \wedge \overline{Q_1^n} \wedge \overline{Q_2^n}) \vee (\overline{Q_0^n} \wedge Q_1^n)$$

$$Q_0^{n+1} = (\overline{Op1} \wedge \overline{Q_0^n}) \vee (\overline{Op2} \wedge \overline{Q_0^n}) \vee (\overline{Q_0^n} \wedge \overline{Q_2^n})$$

$$T0 = \overline{Q_0^n} \wedge \overline{Q_1^n} \wedge \overline{Q_2^n}$$

$$T1 = Q_0^n \wedge \overline{Q_1^n} \wedge \overline{Q_2^n}$$

$$T2 = \overline{Q_0^n} \wedge Q_1^n \wedge \overline{Q_2^n}$$

$$T3 = Q_0^n \wedge Q_1^n \wedge \overline{Q_2^n}$$

$$T4 = \overline{Q_0^n} \wedge \overline{Q_1^n} \wedge Q_2^n$$

$$T5 = Q_0^n \wedge \overline{Q_1^n} \wedge Q_2^n$$

$$T6 = \overline{Q_0^n} \wedge Q_1^n \wedge Q_2^n$$

$$T7 = Q_0^n \wedge Q_1^n \wedge Q_2^n$$

$$Execute = Q_2^n$$

**Figure 3: Truth Table and Logical Expressions for Instruction Sequencer FSM**

The logical expressions were analyzed to form the circuit schematic shown in Figure 4.

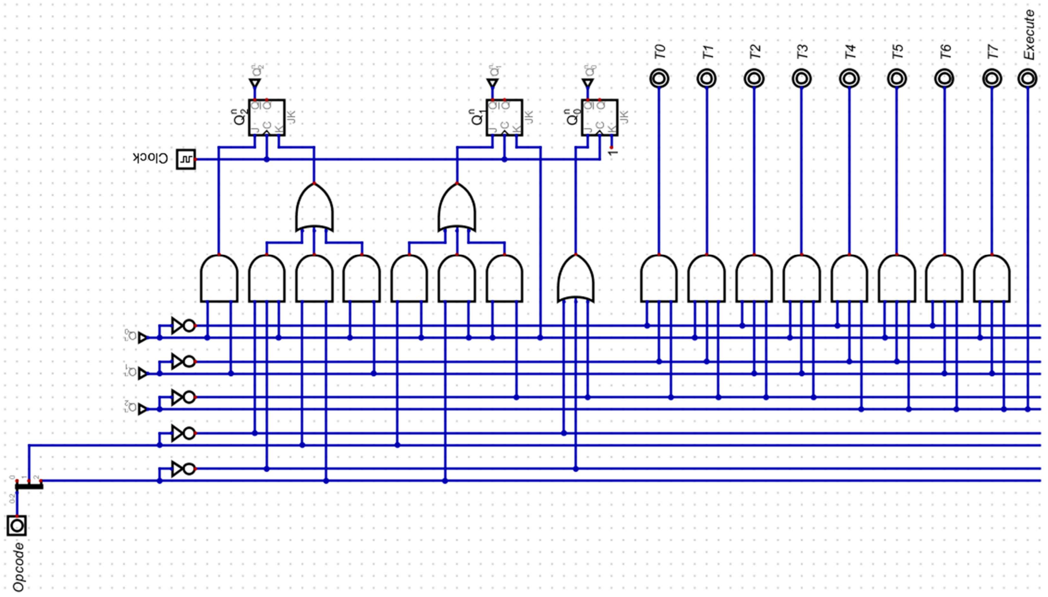
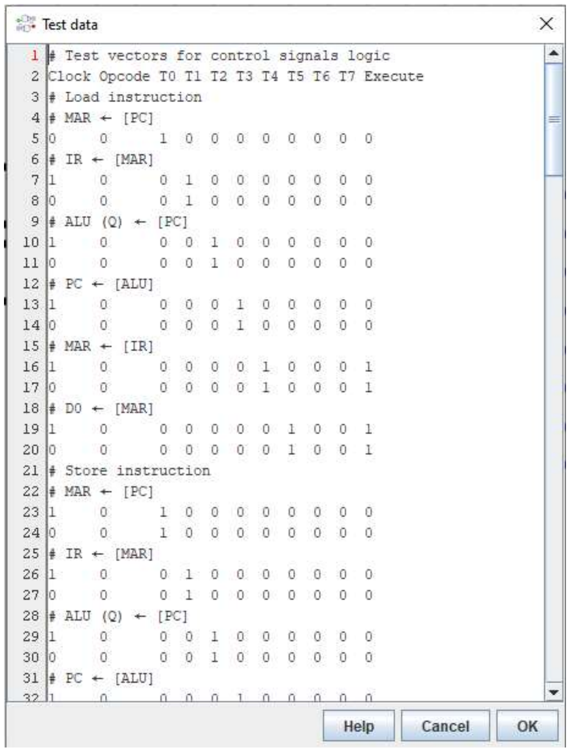


**Figure 4: Circuit for Instruction Sequencer FSM**

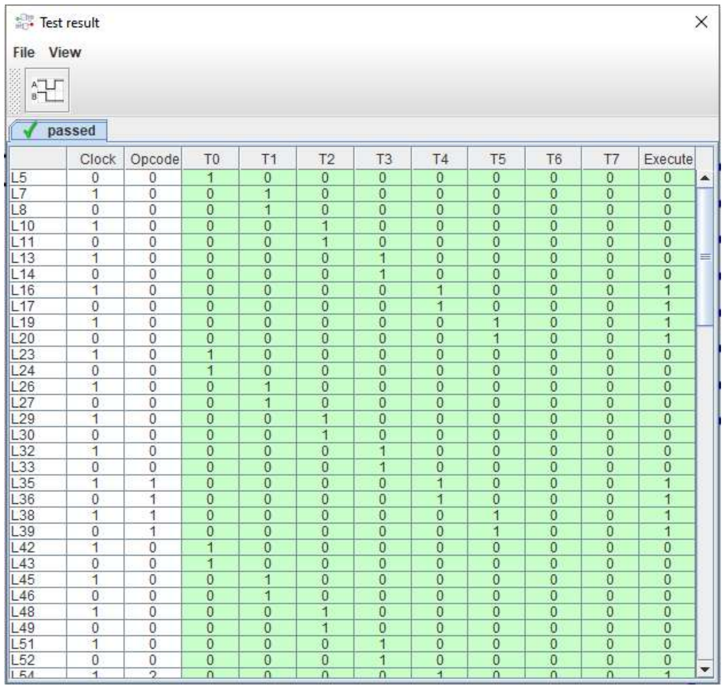
## 3 Numerical Verification

Shown in Figure 5, Figure 6, and Figure 7 are the numerical verification results, where an extensive set of input test vectors (Figure 5) were applied to the circuit.



```
# Test vectors for control signals logic
Clock Opcode T0 T1 T2 T3 T4 T5 T6 T7 Execute
# Load instruction
# MAR ← [PC]
0     0      1  0  0  0  0  0  0  0  0
# IR ← [MAR]
1     0      0  1  0  0  0  0  0  0  0
0     0      0  1  0  0  0  0  0  0  0
# ALU (Q) ← [PC]
1     0      0  0  1  0  0  0  0  0  0
0     0      0  0  1  0  0  0  0  0  0
# PC ← [ALU]
1     0      0  0  0  1  0  0  0  0  0
0     0      0  0  0  1  0  0  0  0  0
# MAR ← [IR]
1     0      0  0  0  0  1  0  0  0  1
0     0      0  0  0  0  1  0  0  0  1
# D0 ← [MAR]
1     0      0  0  0  0  0  1  0  0  1
0     0      0  0  0  0  0  1  0  0  1
# Store instruction
# MAR ← [PC]
1     0      1  0  0  0  0  0  0  0  0
0     0      1  0  0  0  0  0  0  0  0
# IR ← [MAR]
1     0      0  1  0  0  0  0  0  0  0
0     0      0  1  0  0  0  0  0  0  0
# ALU (Q) ← [PC]
1     0      0  0  1  0  0  0  0  0  0
0     0      0  0  1  0  0  0  0  0  0
# PC ← [ALU]
1     0      0  0  0  1  0  0  0  0  0
```

**Figure 5: Input Test Vectors**


Test result
File View
passed

| | Clock | Opcode | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | Execute |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L7 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L8 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L10 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L11 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L13 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L14 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L16 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| L17 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| L19 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| L20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| L23 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L24 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L26 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L27 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L29 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L30 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L32 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L33 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L35 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| L36 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| L38 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| L39 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| L42 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L43 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L45 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L46 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L48 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L49 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| L51 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L52 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| L54 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

**Figure 6: Test Results (Table)**

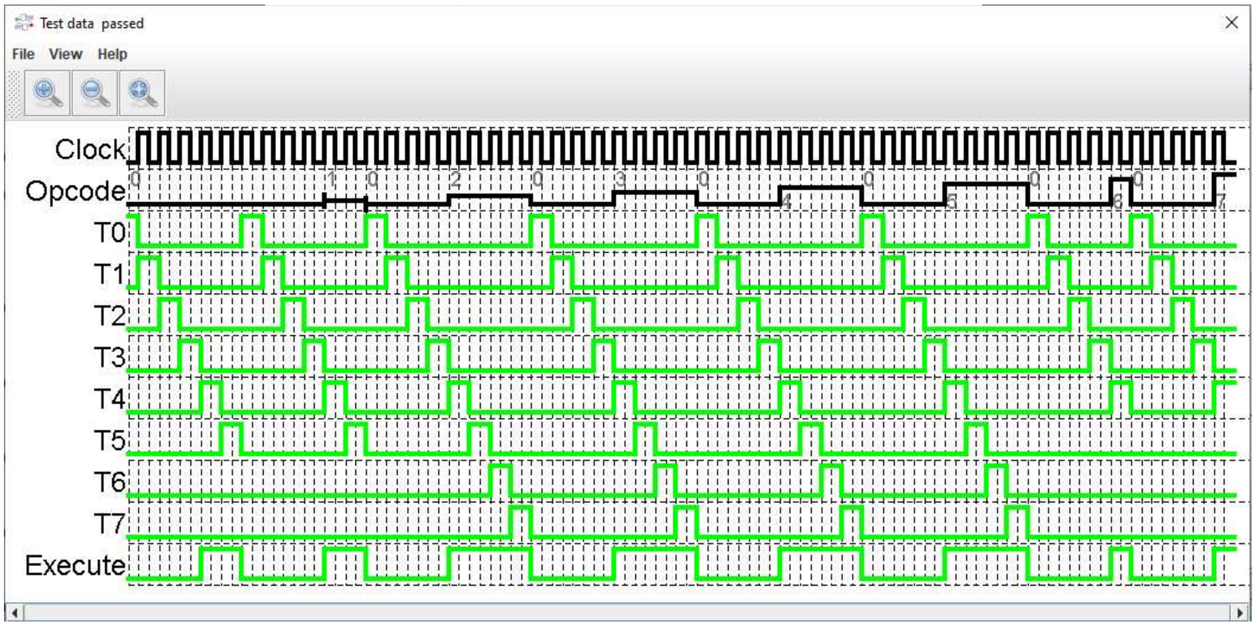


**Figure 7: Test Results (Graph)**

## 4 Summary

The problem requirement for this assignment was to design an Instruction Sequencer FSM circuit to generate the sequencing signals used to ensure the proper sequencing of the control signals generation for fetch and execute. To do so, the circuit must perform the dual function of generating the 8 timing pulses to sequence the fetch and execute operations as well as to generate the signal to indicate when the instruction is being fetched and executed.

The truth table and corresponding logical equations were essential to defining functionality and developing a logic expression, that was then used to specify a digital circuit solution. The digital circuit was tested with an extensive set of input vectors and produced the expected output for each test vector. Therefore, the specified digital circuit design solution satisfies the problem requirements.