Bangladesh University of Engineering and Technology
January 2018 CSE 404
Report on 4-bit PC Design and Simulation

Submitted by:

Anik Sarker (1405001)
Md. Masum Mushfiq (1405002)
Tahmid Hasan (1405004)
Md. Ashiqur Rahman (1405024)
Md. Sadiqur Rahman Sohel (1405030)

Date of Submission:
August 3, 2018

Contents

1 Introduction

In this assignment, we designed the architecture of a 4-bit PC with 8-bit addressing range. This PC includes 25 basic macroinstructions with several arithmatic and logical operations. It includes one 4-bit input port, one 4-bit output port, one arithmatic unit, three data registers, namely *Acc, B, Temp*, one program counter, one stack pointer, one RAM, one address register for RAM, one data register for RAM, one instruction register, and one controller sequencer for generating microinstructions. We also simulated this design using **Proteus** simulation software.

2 Instruction Set

Our design had provision for 25 basic intructions which included input, output, arithmatic, logical, register, memory, stack, jump, and flag operations,

Instruction	**Description**
1. STA address	$Memory[address] \leftarrow Acc$
2. MOV B, Acc	$B \leftarrow Acc$
3. ADC address	$Acc \leftarrow Acc + Memory[address] + Carry$
4. ADC Immediate	$Acc \leftarrow Acc + Immediate + Carry$
5. SBB address	$Acc \leftarrow Acc - Memory[address] - Carry$
6. SBB Immediate	$Acc \leftarrow Acc - Immediate - Carry$
7. DEC	$Acc \leftarrow Acc - 1$
8. IN	$Acc \leftarrow Input\_port$
9. OUT	$Output\_port \leftarrow Acc$
10. PUSH	*Pushes the content of the Accumulator to the stack*
11. POP	*Pops off top element of stack to Accumulator*
12. AND Immediate	$Acc \leftarrow Acc.Immediate$
13. XOR address	$Acc \leftarrow Acc \oplus Memory[address]$
14. XOR Immediate	$Acc \leftarrow Acc \oplus Immediate$
15. XCHG	$Acc \Longleftrightarrow B$
16. NOT	$Acc \leftarrow !Acc$
17. CMP B	*Accumulator will be unchanged. Set flags according to* $A - B$
18. SHL	$Acc \leftarrow Acc << 1, Carry \leftarrow Acc[MSB], Acc[LSB] \leftarrow 0$
19. ROL	$Acc \leftarrow Acc << 1, Carry \leftarrow Acc[MSB], Acc[LSB] \leftarrow Acc[MSB]$
20. JMP address	*Jump to the address*
21. JE address	*Jump if equal*
22. JG address	*Jump if greater*
23. CLC	*Clears the Carry flag*
24. CLZ	*Clears the Zero flag*
25. HLT	*Halts execution*

Table 1: Instruction Set

3 Block Diagram

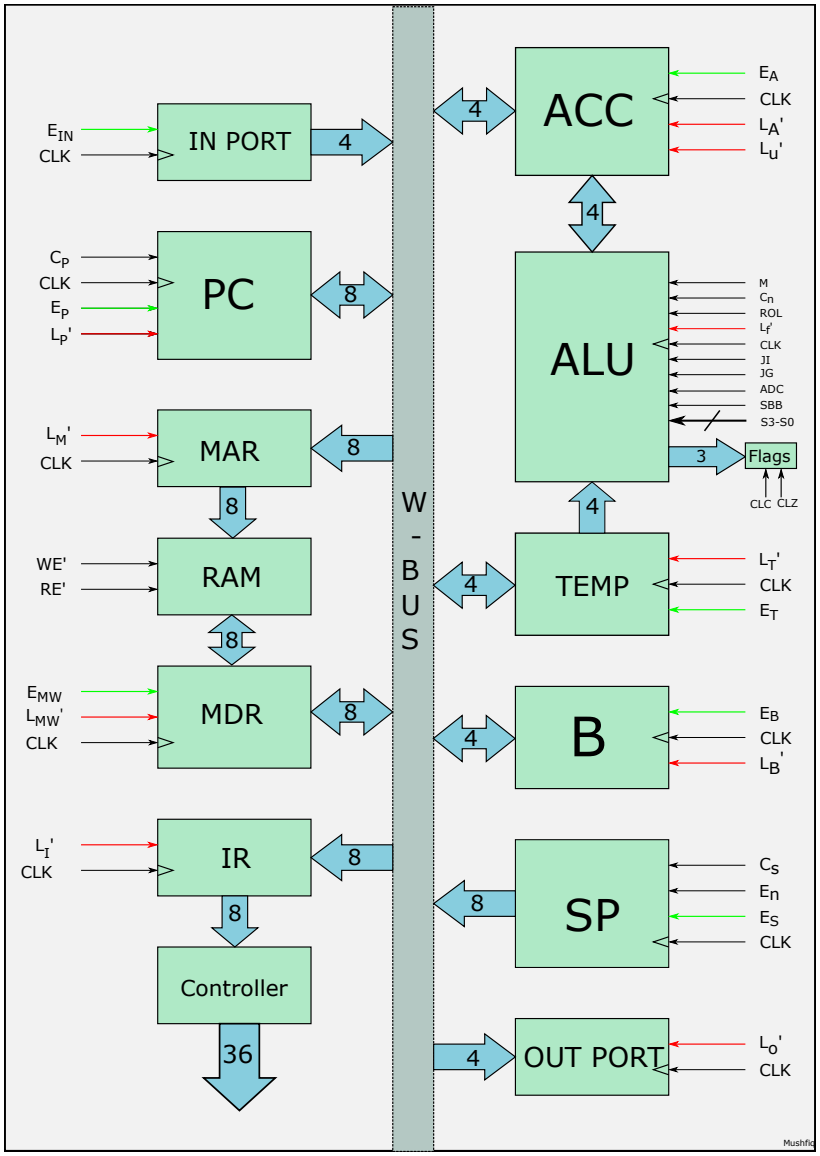

Figure 1: Block Diagram of 4-bit PC

4 Brief Description of Blocks

4.1 Input Port

Input port contains a 4-bit register. When it gets a positive clock edge, input data is loaded to register. When E_{IN} signal is high, data is emitted to bus through a tristate buffer.

4.2 Output Port

Ouput port also has a 4-bit register which shows the output to a 7-segment display. When L'_O is low, data is loaded into the register through a tristate buffer. When the register receives a negative clock pulse, it displays the output.

4.3 Accumulator

Accumulator contains a 4-bit bidirectional register connected to both bus and ALU. When L'_A is low and the register receives a negative clock pulse, it stores the data of the bus through a tristate buffer. Accumulator always pushes its data to ALU irrespective of the clock signal. When L'_U is low, ALU loads the resultant of the operatoin to accumulator through another tristate buffer. When E_A is high, Accumulator emits its data to bus.

4.4 Temp Register

Temp is a temporary 4-bit register connected to bus and ALU. ALU connection is combinational while bus connection is controlled by clock. When L'_T is low, the data from the bus is loaded on a negative clock pulse. When E_T is high, data is emitted to bus.

4.5 B Register

B is a 4-bit register connected to bus only. When E_B is high, data is loaded to bus. When L'_B is low and it receives a negative clock edge, data is loaded from bus.

4.6 PC

PC is an 8-bit register which contains the address of the current instruction. When the program starts, its value is 1. When C_P is high and it gets a positive clock edge, its value is incremented by 1 through a counter. When E_P is high, PC value is loaded to bus. When L'_P is low, data is loaded from bus on a negative clock edge.

4.7 SP

SP is the stack pointer which points it its top address where value will be stored on push operation. It has no load operation. Its initial value is $0xFF$. When E_N is enbled, it can only then count up or down using a counter. When C_S is low, it counts down. When high, it counts up. When E_S is enbled, data is loaded to bus.

4.8 RAM

Our 4-bit PC has a $2^8 * 8$ RAM which is connected to MAR and MDR. When RE' is low, it loads the data to MDR from the memory location pointed by MAR. When WE' goes from 1 to 0, data from MDR is written to the memory location pointed by MAR.

4.9 MAR

MAR is an 8-bit address register. When L'_M is low, address is loaded from address bus to MAR on negative clock edge. The connection from MAR to RAM is combinational i.e. MAR always points to the memoey location indicated by its value.

4.10 MDR

MDR is an 8-bit data register. When we are dealing with address, we use all 8 bits. When we use it to load or store data, we only take into account the least significant 4 bits. When L'_{MW} is low, it loads the data from bus on negative clock edge. When E_{MW} is high, data is loaded to bus.

4.11 IR

IR is the instruction register. It contains the opcode for the current instruction. When L'_I is low, it loads the instruction from bus on negative edge and sends it to controller for decoding.

4.12 Controller

The controller is a sequencer which generates the control signals to execute each microinstruction. Our 4-bit PC is microprogrammed with 75 control words each containing 39 control signals. These control words are stored in 5 ROMs each with 8 bit range. Which control word is selected next is controlled by 2 mode signals which select whether to read from the location pointed by IR or increment the counter value to read control word.

5 Circuit Diagram

Diagram of the main circuit along with its subcircuits is attached.

6 Cycle Descriptions

Instruction(OP Code, #T states)	Microinstructions	Active Signals ALU in Brackets	CW
0. Fetch (0, 3)	***1.*** $MAR \leftarrow PC$	L'_M, E_P	$666940D122$
	2. $MDR \leftarrow RAM[MAR], PC{+}{+}$	C_P, RE'	$3A6940D122$
	3. $IR \leftarrow MDR$	E_{MW}, L'_I	$F26140D121$
1. STA address (3, 8)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $MAR \leftarrow MDR$	E_{MW}, L'_M	$E26940D122$
	7. $MDR \leftarrow Acc$	E_A, L'_{MW}	$727840D122$
	8. $RAM[MAR] \leftarrow MDR$	W'_E	$526940D122$
2. MOV B, Acc (8, 4)	***4.*** $B \leftarrow Acc$	E_A, L'_B	$727940D122$
3. ADC address (9, 9)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $MAR \leftarrow MDR$	E_{MW}, L'_M	$E26940D122$
	7. $MDR \leftarrow RAM[MAR]$	RE'	$326940D122$
	8. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	9. $Acc \leftarrow ALU$	$L'_u, L'_F, ADC(011001)$	$722966D140$
4. ADC Immediate (15, 7)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	7. $Acc \leftarrow ALU$	$L'_u, L'_F, ADC(011001)$	$722966D140$
5. SBB address (19, 9)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $MAR \leftarrow MDR$	E_{MW}, L'_M	$E26940D122$
	7. $MDR \leftarrow RAM[MAR]$	RE'	$326940D122$
	8. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	9. $Acc \leftarrow ALU$	$L'_u, L'_F, SBB(000110)$	$722958D180$
6. SBB Immediate (25, 7)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	7. $Acc \leftarrow ALU$	$L'_u, L'_F, SBB(000110)$	$722958D180$
7. DEC (29, 4)	***4.*** $Acc \leftarrow ALU$	$L'_U, L'_F(000000)$	$722940D100$
8. IN (30, 4)	***4.*** $Acc \leftarrow Input\_port$	E_{IN}, L'_A	$734940D120$
9. OUT (31, 4)	***4.*** $Output\_port \leftarrow Acc$	E_A, L'_O	$727940C120$
10. PUSH (32, 6)	***4.*** $MAR \leftarrow SP$	E_S, L'_M	$626940D922$
	5. $MDR \leftarrow Acc, SP--$	E_A, L'_{MW}, E_N, C'_S	$727840D522$
	6. $Temp \leftarrow MDR$	L'_T, E_{MW}	$526940D120$
11. POP (35, 7)	***4.*** $SP{+}{+}$	E_N, C_S	$726940D722$
	5. $MAR \leftarrow SP$	E_S, L'_M	$626940D922$
	6. $MDR \leftarrow RAM[MAR]$	R'_E	$326940D122$
	7. $Acc \leftarrow MDR$	E_{MW}, L'_A	$F24940D120$
12. AND Immediate (39, 7)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	7. $Acc \leftarrow ALU$	$L'_u, L'_F(101110)$	$72295DD100$
13. XOR address (43, 9)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $MAR \leftarrow MDR$	E_{MW}, L'_M	$E26940D122$

	7. $MDR \leftarrow RAM[MAR]$	RE'	$326940D122$
	8. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	9. $Acc \leftarrow ALU$	$L'_u, L'_F(101001)$	$722965D100$
14. XOR Immediate (49, 7)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $Temp \leftarrow MDR$	L'_T, E_{MW}	$F26900D122$
	7. $Acc \leftarrow ALU$	$L'_u, L'_F(101001)$	$722965D100$
15. XCHG (53, 6)	***4.*** $Temp \leftarrow Acc$	E_A, L'_T	$727900D122$
	5. $Acc \leftarrow B$	E_B, L'_A	$72C940D122$
	6. $B \leftarrow Temp$	E_T, L'_B	$7269C0D020$
16. NOT (56, 4)	***4.*** $Acc \leftarrow ALU$	$L'_U, L'_F(100000)$	$722941D100$
17. CMP B (57, 5)	***4.*** $Temp \leftarrow B$	E_B, L'_T	$72E900D122$
	5. (SUB)	$L'_F(010110)$	$72695AD100$
18. SHL (59, 5)	***4.*** $Temp \leftarrow Acc(ADD)$	E_A, L'_T	$727900D122$
	5. $Acc \leftarrow ALU$	$L'_U, L'_F(010110)$	$722964D100$
19. ROL (61, 5)	***4.*** $Temp \leftarrow Acc(ADD)$	E_A, L'_T	$727900D122$
	5. $Acc \leftarrow ALU$	$L'_U, L'_F, ROL(010110)$	$722964F100$
20. JMP address (63, 6)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR]$	RE'	$326940D122$
	6. $PC \leftarrow MDR$	E_{MW}, L'_P	$F06940D120$
21. JE address (66, 6)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $BUS \leftarrow MDR, ENABLE\ JE$	E_{MW}, JE	$F26B40D120$
22. JG address (69, 6)	***4.*** $MAR \leftarrow PC$	E_P, L'_M	$666940D122$
	5. $MDR \leftarrow RAM[MAR], PC{+}{+}$	RE', C_P	$3A6940D122$
	6. $BUS \leftarrow MDR, ENABLE\ JG$	E_{MW}, JG	$F26D40D120$
23. CLC (72, 4)	***4.*** $ENABLE\ CLC$	CLC	7269409120
24. CLZ (73, 4)	***4.*** $ENABLE\ CLZ$	CLZ	7269405120
25. HLT (74, 4)	***4.*** $ENABLE\ HLT$	HLT	$726940D132$

Table 2: Complete cycle descriptions of all instructions

7 Timing Diagram

The timing diagrams of the five instructions **STA address, SBB address, POP, XOR Immediate, JG** for all T states are given below.

7.1 STA address

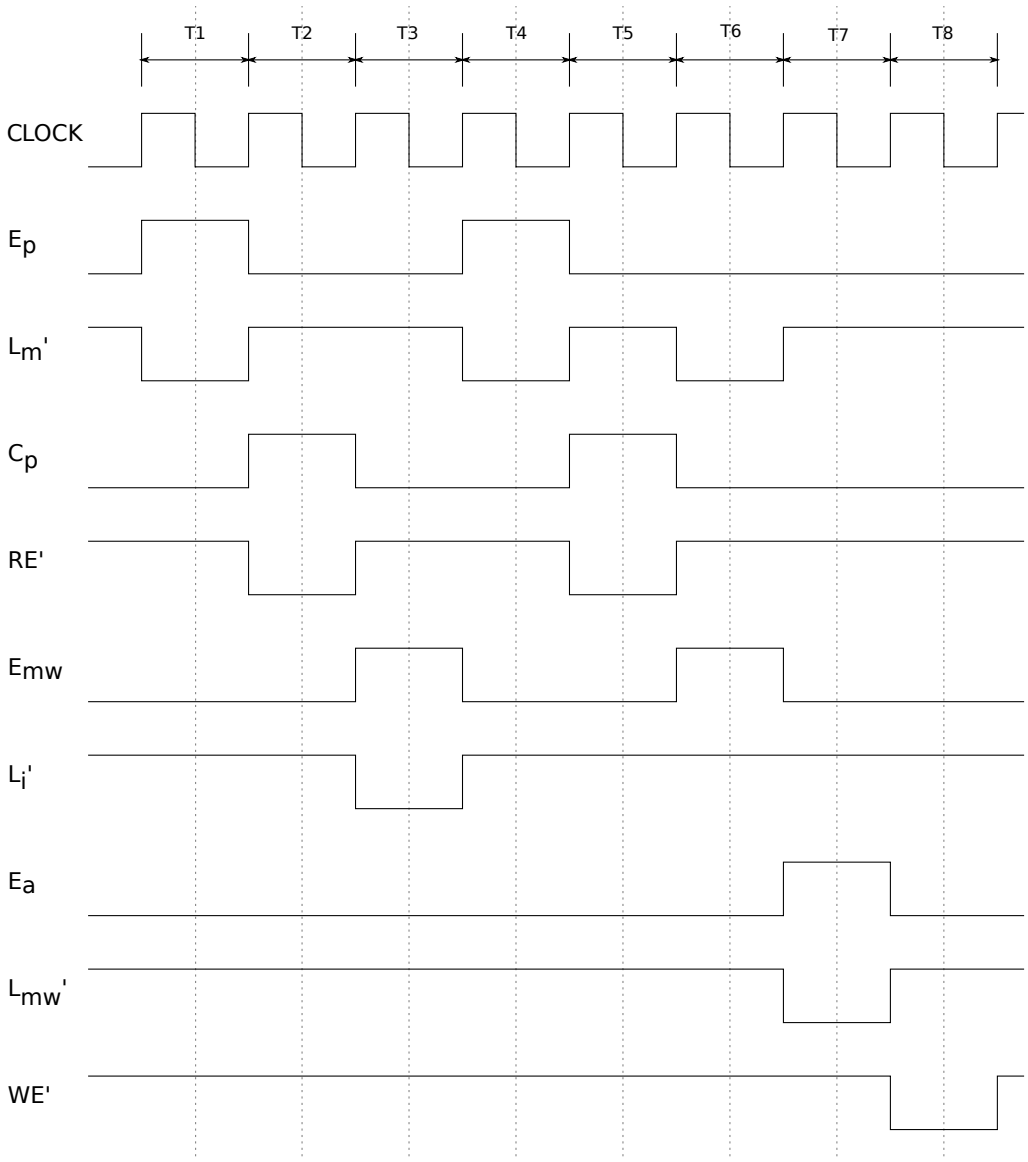

Figure 2: Timing Diagram of **STA address**

7.2 SBB address

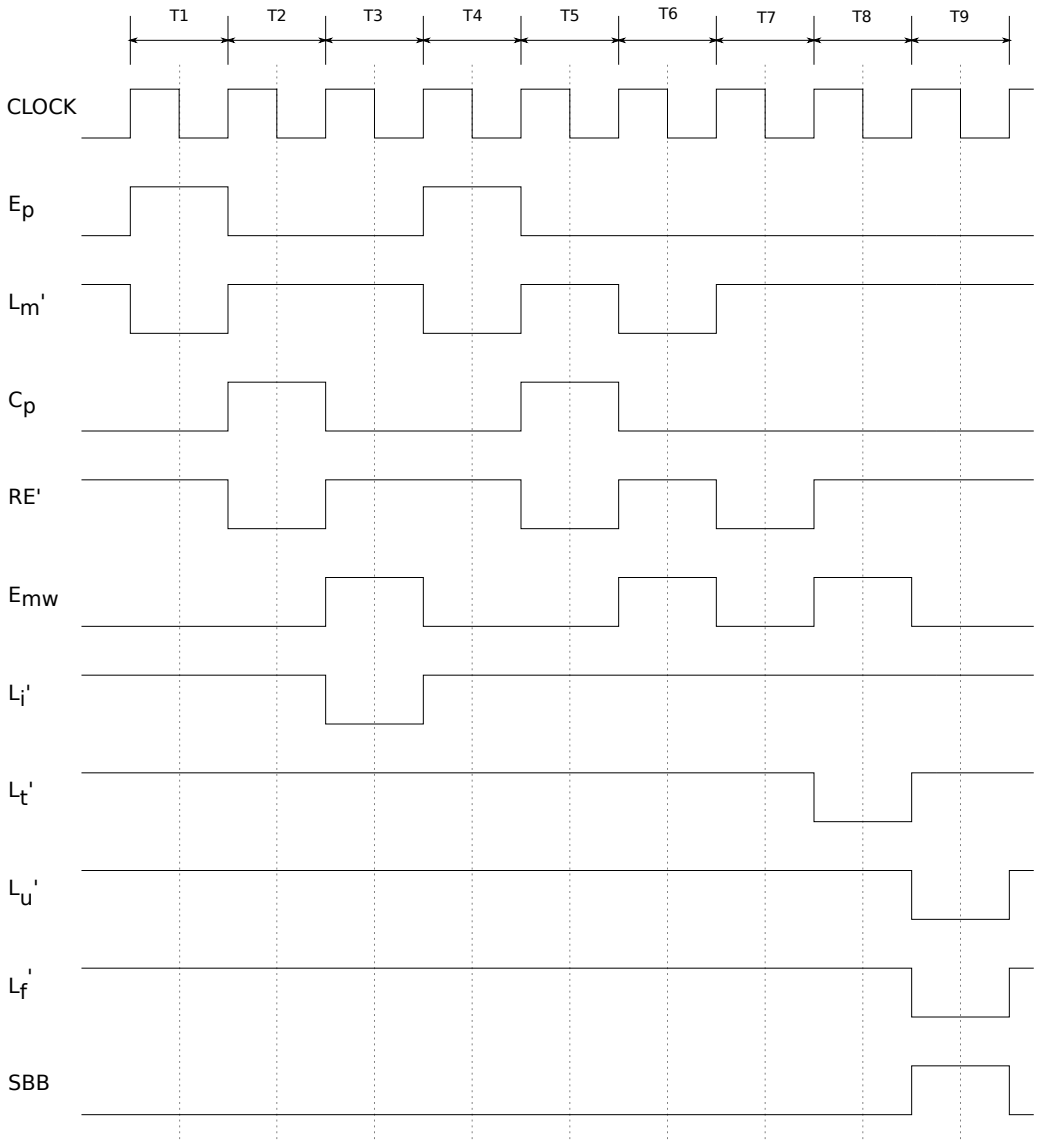

Figure 3: Timing Diagram of **SBB address**

7.3 POP

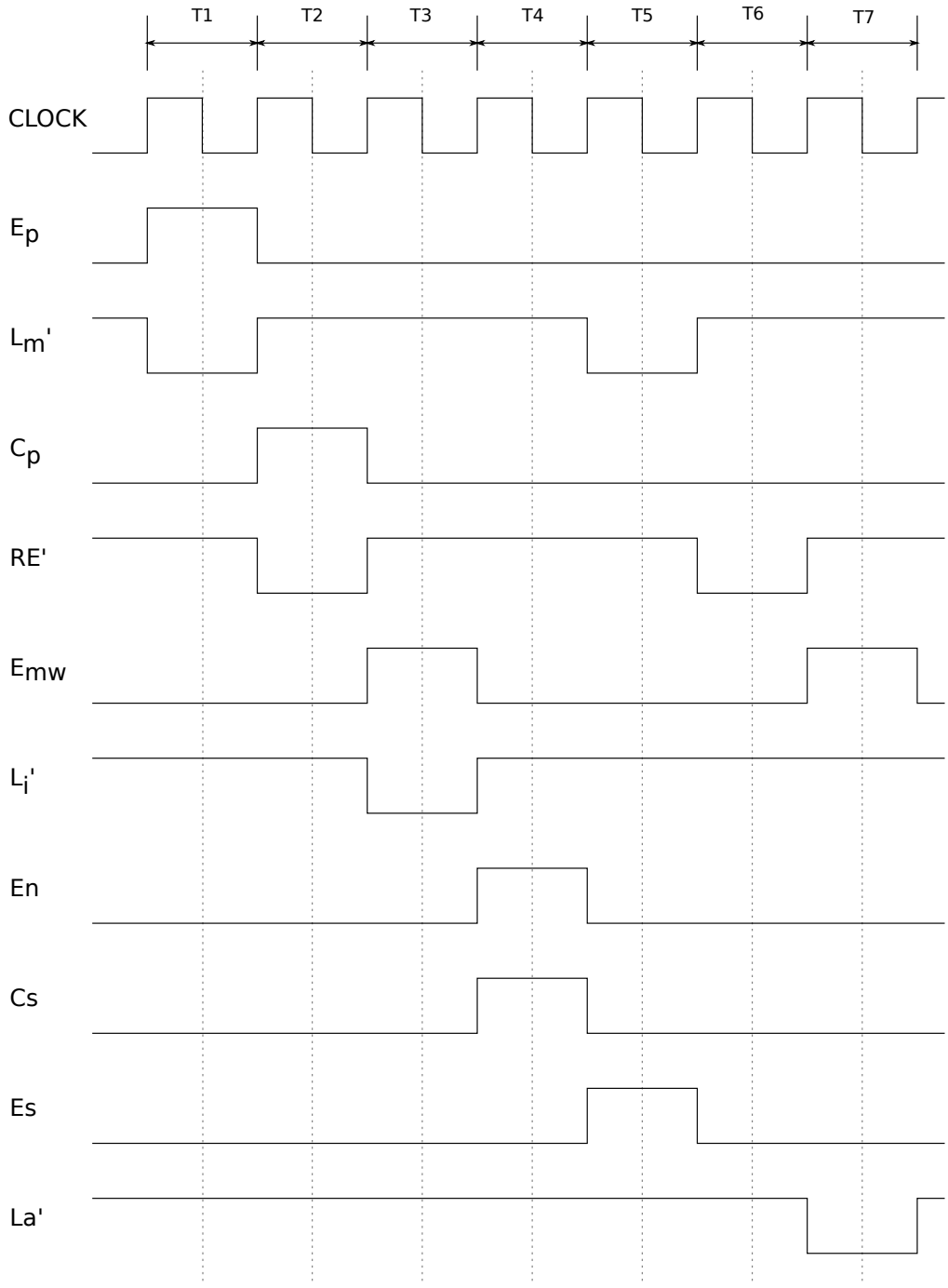

Figure 4: Timing Diagram of **POP**

7.4 XOR Immediate

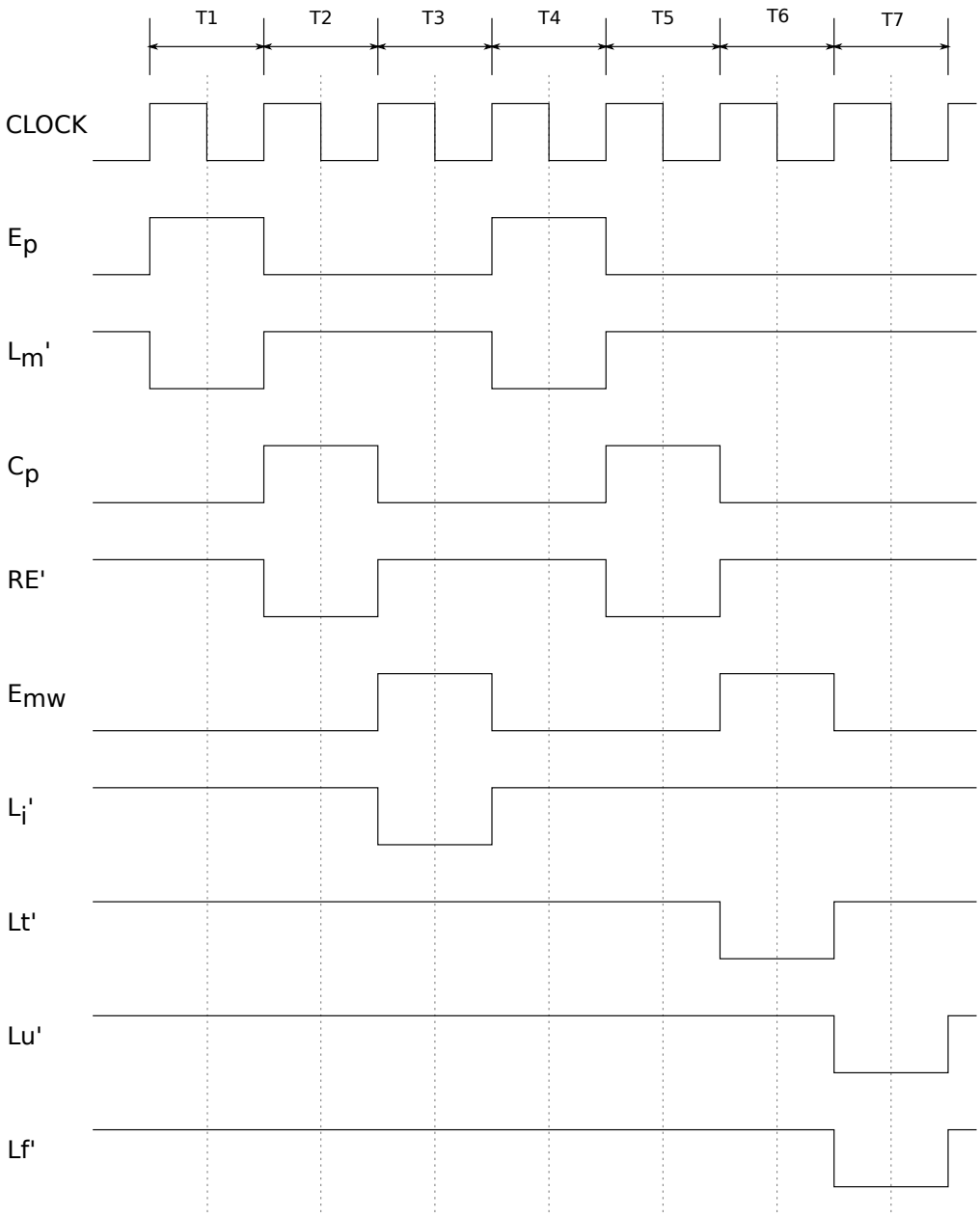

Figure 5: Timing Diagram of **XOR Immediate**

7.5 JG

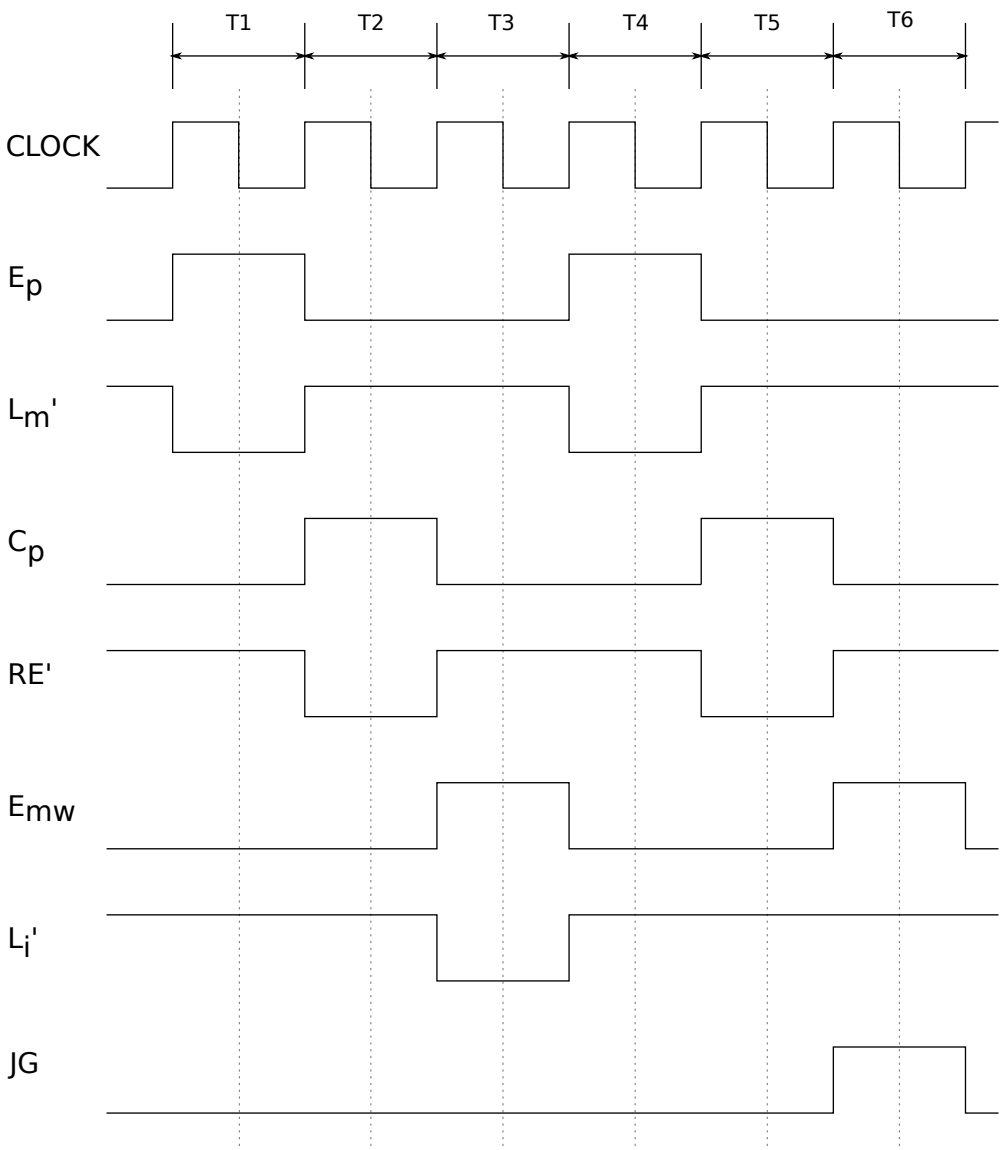

Figure 6: Timing Diagram of **JG**

8 Writing and Executing Program

A separate ROM is used to store the code in HEX format. A counter generated the address sequentially. This data is then transferred to RAM and written to the same address as the RAM. This module is active until the counter counts upto 255. Then it is switched off by switching off the clock dedicated to this module only. The code is then executed starting from its first address by toggling the initial clock and setting the PC to 1. Execution is stopped by the halting instruction which stops the clock.

9 Equipments Used

Name	IC Number	Quantity
7 Segment BCD	N/A	28
NOT Gate	7404	49
AND Gate	7408	8
OR Gate	7432	2
MUX 4:1	74153	2
MUX 2:1	74157	6
D Flip-flop	74173	13
ALU	74181	1
Tristate Buffer	74244	31
Memory(ROM)	2732	6
XNOR Gate	4077	1
2K*8 Static RAM	6116	1
AND_8	N/A	1
CLOCK	N/A	2
Counter_8	N/A	4
Logicprobe	N/A	81
Logicstate	N/A	10

Table 3: List of Equipments

10 Discussion

This project was, perhaps, the most challenging assignment of our undergraduate studies. It was the only single project where each and every single team member gave his fullest effort. We initially made some mistakes that took away a bulk amount of time. The ALU IC description provided to us contained some wrong signals. We solved this by searching the main datasheet on the internet. It was unbeknownst to us that WE' signal was edge triggered instead to level triggered like all other signals. We also had trouble uploading the code from ROM to RAM. We had to increment the program counter by 1 beforehand for fully functioning operation. We used two seperate clocks: one for code uploading and one for regular operation. We accidentally fed the code uploading clock to our main module which resulted in wrong operations. But despite all these mistakes and shortcomings we were finally able to complete the project by dint of our combined efforts and instructors' advice.