4 bit Computer Design and Implementation

CSE 404N
Digital System Design Sessional

Group #1
Student No.: **0205002**
0205003
0205004
0205014
0205026
0005069

Table of Contents

1. Introduction:

The design and implementation of a 4 – bit computer poses a really challenging, yet wonderful design and implementation concern to the novice hardware designers like us. It requires a good understanding of various design techniques and a complete realization of the computer architecture. It also provides a fragrance of a large scale design challenge of designing a whole computer in a small scale version imposed by the 4-bit data size and a restricted instruction set.

We, the hardware designers often tend to make a language out of the different hardware components. Whenever we think of logical ANDing, ORing or XORing, the respective logic gates readily provides us with such functions. We always think of branching in terms of a multiplexer. A for loop or a while loop often gives us the illusion of using a counter or a flip-flop. These little things form the vocabulary of a hardware design. An instruction uses this vocabulary to do some definite task. Thus an instruction set defines the complete specification of the assembly language lying beneath the hardware.

In our computer design, we implemented an instruction set consisting of 28 instructions as suggested by our course teachers. Our computer has a 4 bit data length and an 8 bit address length. However, the design is easily extendible to a more powerful computer.

In the following sections, we shall describe the design of our computer under various titles. Section 2 gives the instruction set. In section 3 and 4, we presented the block diagram and the description of different blocks in that diagram. Section 5 and 6 list and describes the Control Words and signals used. In section 7, we have given a detailed pin-diagram specification of the circuit. Section 8 addresses the timing diagrams for all the instructions while section 9 describes the working techniques of these instructions. Section 10 is a little user manual to show how to program and run our Computer. Section 11 lists some special features of our Computer. Finally in section 12, all the ICs used with count are listed.

2. Instruction Set:

Instruction	Description
LDA address	Acc ← Memory [address]
STA address	Memory [address] ← Acc
MOV Acc, B	Acc ← B
MOV B, Acc	B ← Acc
MOV Acc, immediate	Acc ← immediate
In	Acc ← input_port
Out	Output_port ← Acc
ADD B	Acc ← Acc + B
ADC B	Acc ← Acc + B + C (Contents of Carry Flag)
SUB B	Acc ← Acc – B
SBB B	Acc ← Acc - B - Bo(Contents of Carry Flag)
ADD immediate	Acc ← Acc + immediate
SUB [address]	Acc ← Acc - Memory [address]
CMP B	Set Flags according to “SUB B”
TEST B	Set Flags according to “AND B”
JC address	Jumps to the address if the Carry Flag is set
JE address	Jumps to the address if equal i.e. the Zero Flag is set
PUSH	Pushes the contents of the accumulator to the stack
POP	Pops off the top of the stack to the accumulator
CALL address	Calls a subroutine at the address unconditionally
RET	Returns from the current subroutine to the caller
JMP address	Jumps unconditionally to the address
HLT	Halts execution
NOP	No Operation
SHL	Acc ← Acc << 1, C ←Acc [MSB], Acc [LSB] ←0
SHR	Acc ← Acc >> 1, C ←Acc [LSB], Acc [MSB] ←0
AND B	Acc ← Acc. B
XOR immediate	Acc ← Acc xor immediate

3. Block Diagram:

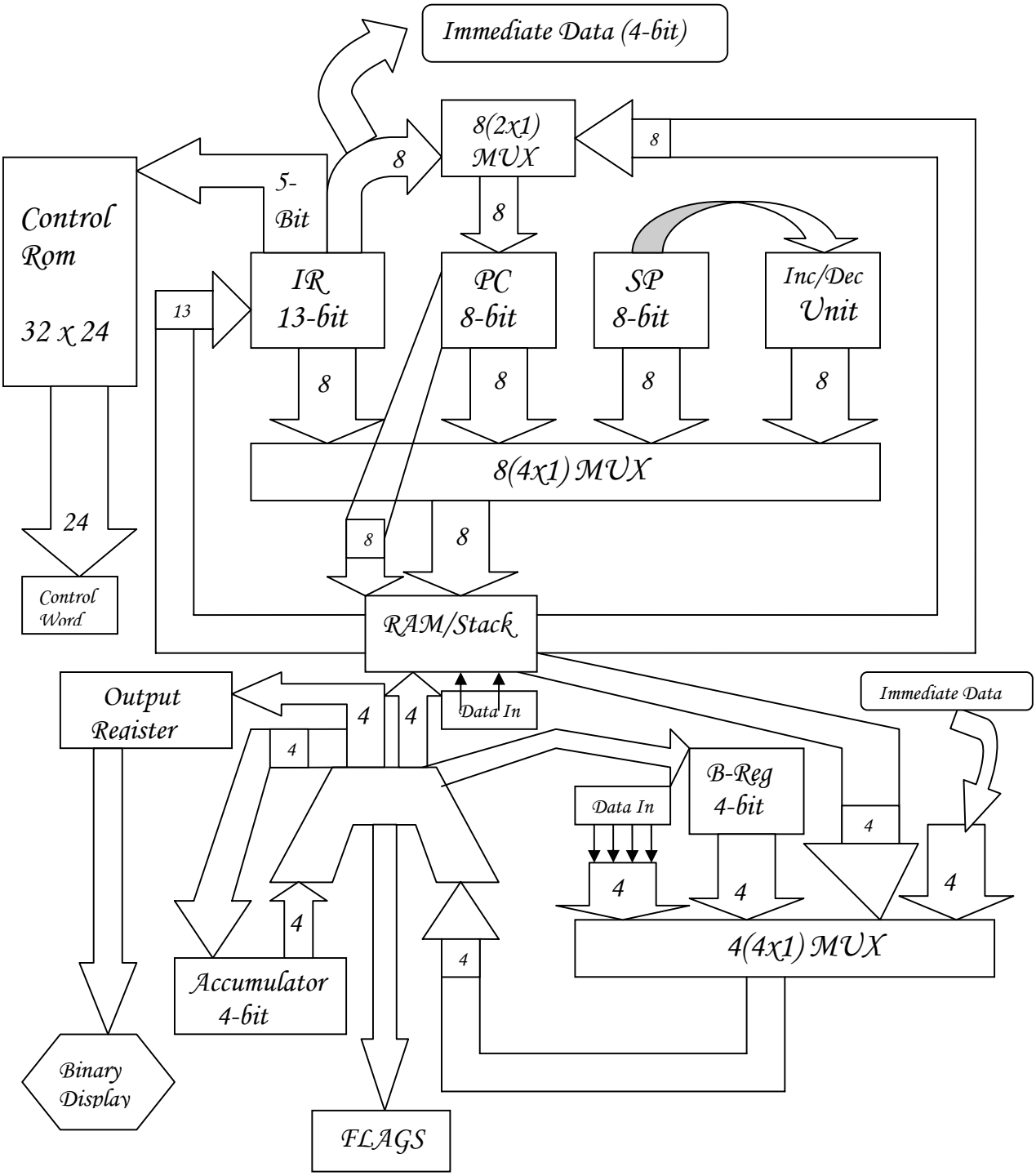


Fig: *Block Diagram of 4-bit Computer*

4. Components of the Computer:

4.1. Accumulator:

Accumulator is a register that stores intermediate results during a computer run. In arithmetic and logic operations, one operands must be the accumulator and the result of the operation is also stored here.

The PUSH, POP, IN, OUT and MOV instructions also involves Accumulator register. In the block diagram, Accumulator output goes to the A input of the ALU and the output of the ALU unit comes as the input of the Accumulator.

4.2. B register:

B is another register that is used in various two-operand arithmetic and logic operations. its output goes to the B MUX and its input is provided from the result of the ALU unit

4.3. ALU unit:

ALU unit provides the necessary arithmetic and logic operations. One of its input comes from the output of the Accumulator and the other input comes from the B MUX. Its output goes to the input of Accumulator, B register, Output register and RAM/Stack.

4.4. Output Register and Binary Display:

After the execution of OUT command, the value of Accumulator is stored in the output register and is displayed in the binary display consisting of4 LEDs.

4.5. RAM/Stack:

RAM stores the Opcode, immediate data / address and also provides the space for the DATA segment for the user. Stack gives the user the provision of stacking various data and later popping them. It also saves PC value pointing the instruction to be executed next on the Caller subroutine so that it can be retrieved after the execution of the RET instruction in the Callee subroutine.

RAM and Stack are placed together and they are differentiated through the unused MSB address bits. Also, Control ROM provides the signal /Mem-Stack to specify whether RAM or Stack should be addressed.

RAM/Stack input comes from three sources: ALU result, PC output and Input switches. Its output goes to B MUX as well as to the IR input. It is addressed by the Address MUX.

4.6. Program Counter (PC):

PC is a counter that points to the memory address that stores the next instruction to be executed. At the start of the execution, PC is always reset to 0. During the normal sequential flow, after every fetch cycle, PC value is incremented by 1. When a jump is required, PC value is loaded on the rising edge of the execution clock cycle by the value provided from PC MUX. Its output goes to the Address MUX so that it can provide the addressing to the RAM during fetch cycle.

4.7. Instruction Register (IR):

IR stores the Opcode and immediate data/address of the instruction that is currently being executed. When its value is 0, it specifies that fetch cycle is being performed. At the start of the computer run, it is reset to 0. During fetch cycle, it is loaded from the RAM location addressed by the PC value. After every execution cycle, it is once again reset to 0. Its output provides the necessary addressing to the Control ROM so that it can provide the Control Signal appropriate to the Opcode currently being executed. Its output also goes to the Address MUX to provide immediate addressing of the RAM as needed.

4.8. Control ROM:

Control ROM is addressed by the Opcode currently being stored in IR and it provides the 24 – bit Control Signal to run the computer correctly. The Control Signals are described in section 5 and 6.

4.9. SP and the SP-Decrement Unit:

SP is an up-down counter that points to the top of the Stack. Initially it is reset to 0. After the execution of a PUSH or CALL instruction, its value is incremented by 1 and after the execution of a POP or RET instruction, its value is decremented by 1. Its output goes to the Address MUX to provide addressing of the stack during push or CALL instruction.

The output of SP also goes to the SP-Decrement unit to provide decremented value of SP a-priori of the execution of the POP and RET instruction via the Address MUX.

4.10. Address MUX:

The Address MUX provides the proper addressing of the RAM / Stack among several addresses: PC, IR, SP, SP-1, Input address switches. The Control Signals AddS1 and AddS0 and the RUN-/PROG input make up the selectors of this MUX.

4.11. B MUX:

The B MUX provides the B input of the ALU unit among 4 options: B register, RAM, input port and immediate data from IR. The Control Signals BS1 and BS0 make up the selectors of this MUX.

4.12. PC MUX:

The PC MUX provides the input of the PC between 2 options: Stack (during RET) and immediate data from IR. The Control Signal PCS makes up the selector of this MUX.

4.13. Input Port:

Input port provides the input data during the execution of IN instruction via the B MUX.

4.14. Input Switches:

These Switches provides the addressing, Opcode and immediate data/address during the program mode of the computer.

5. Opcodes and Control Words:

OpCode (Decimal)	OpCode(Binary)	Command	Control Word (Binary)																								Hexadecimal Control Word
			1				2				3				4				5				6				
			PCinU	PCinC	PCinZ	PCS	_Mem/Stk	_Inc/Dec	R/_W	ALU0	AddS1	AddS0	BS1	BS0	ALU1	ALU2	ALU3	ALU4	ALU5	AccS1	AccS0	Bin	Flagin	OUTin	_IRR	HLT	
00	00000	Fetch	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	02C002
01	00001	LDA address	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	03A8E0
02	00010	STA address	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	018D80
03	00011	MOV Acc, B	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0318E0
04	00100	MOV B, Acc	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0	030D90
05	00101	MOV Acc, imm	0	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0338E0
06	00110	IN	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0308E0
07	00111	OUT	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	0	030D84
08	01000	ADD B	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0214E8
09	01001	ADC B	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	0	1	0	0	0	0216E8
10	01010	SUB B	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	1	1	0	1	0	0	0	031168
11	01011	SBB B	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	031368
12	01100	ADD imm	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1	0	0	0	0234E8
13	01101	SUB [addr]	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	0	03A168
14	01110	CMP B	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	031108
15	01111	TEST B	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0	031E88

OpCode (Decimal)	OpCode(Binary)	Command	Control Word (Binary)																								Hexadecimal Control Word
			1				2				3				4				5				6				
			PCinU	PCinC	PCinZ	PCS	_Mem/Stk	_INc/Dec	R/_W	ALU0	AddS1	AddS0	BS1	BS0	ALU1	ALU2	ALU3	ALU4	ALU5	AccS1	AccS0	Bin	Flagin	OUTin	_IRR	HLT	
16	10000	Fetch	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	02C002
17	10001	JC address	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	520000
18	10010	JE address	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	320000
19	10011	PUSH	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	094D80
20	10100	POP	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0F28E0
21	10101	CALL address	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	984000
22	10110	RET	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8E0000
23	10111	JMP address	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	920000
24	11000	HLT	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	020003
25	11001	NOP	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	020000
26	11010	SHL	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	1	1	0	0	1	0	0	0	030DC8
27	11011	SHR	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	030DA8
28	11100	AND B	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	1	0	1	0	0	0	031EE8
29	11101	XOR imm	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1	1	0	1	1	0	1	0	0	0	033B68
30	11110	NOP	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	020000
31	11111	NOP	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	020000

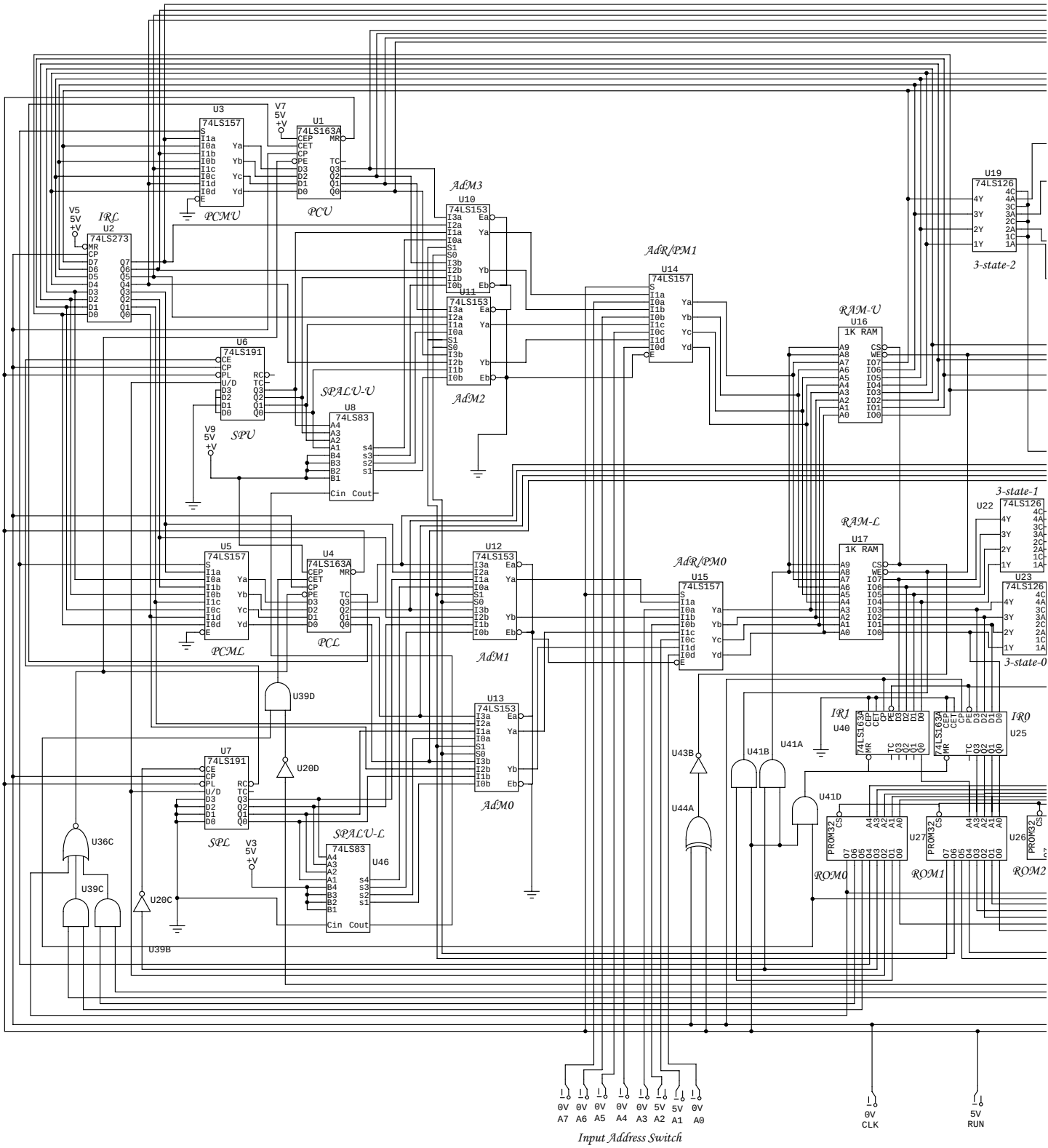
6. Control Signals:

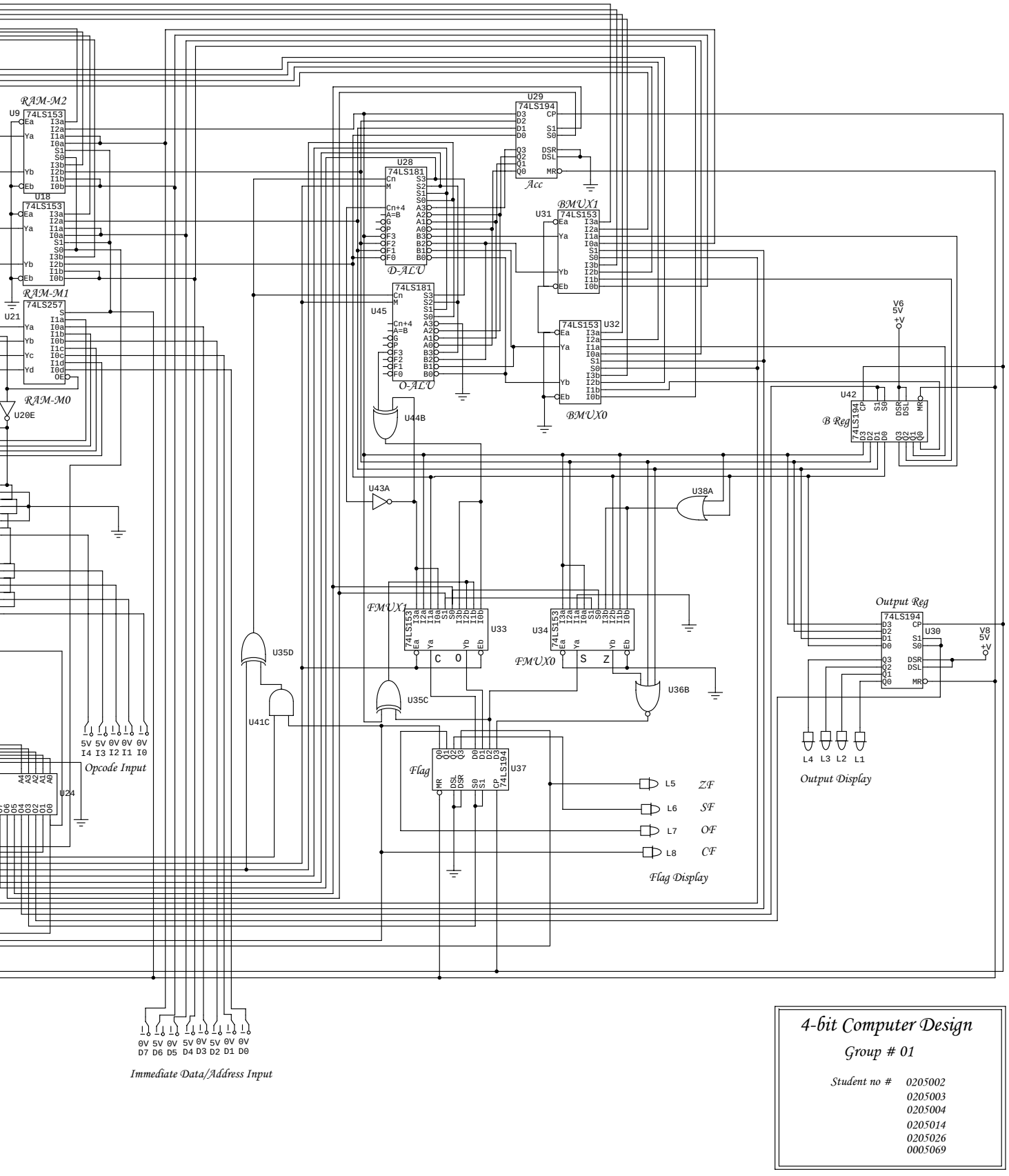
Signal	Description		Comment	Instructions
PCinU	1	Jump unconditionally		**JMP, CALL, RET**
	0	No unconditional jump		
PCinC	1	Jump if C=1		**JC**
	0	Do not jump on C		
PCinZ	1	Jump if Z=1		**JE**
	0	Do not jump on Z		
PCS	0	Popped address from Stack output	Selects the PCMUX ; Significant only if PCinU=1	**JMP,CALL,RET**
	1	Immediate address from IR		
____ Mem/Stack	0	RAM	Acts as count enable of SP as well	0 **RAM operations, fetch**
	1	Stack		1 **PUSH,POP,CALL,RET**
__ Inc/Dec	0	SP ←SP + 1	Significant only if Stack=1	0 **PUSH,CALL**
	1	SP ← SP – 1		1 **POP,RET**
__ R/W	0	Write Memory/Stack		**RAM and Stack operations, fetch**
	1	Read Memory/Stack		
ALU0-5	S0=ALU2 S1=ALU0 S2=ALU4 S3=ALU5 Cin=ALU2 xor (ALU3.C) M=ALU1		Selects ALU Operational Mode	**All instructions working on data**
AddS1-0	00	SP – 1	Select the Address MUX	**All instructions of RAM and Stack**
	01	SP		
	10	IR		
	11	PC		
BS1-0	00	Input port	Select the B MUX	**All operations involving data**
	01	B register		
	10	RAM/Stack		
	11	Immediate data (IR)		
AccS1-0	00	No change	Select Accumulator operations	**All operations involving data**
	01	Shift Right		
	10	Shift Left		
	11	Load		
Bin	1	Load B		**MOV B, Acc**
	0	Do not load B		
Flagin	1	Load Flags		**All operations where Flags are affected**
	0	Do not load Flags		

Signal	Description		Comment	Instructions
Outin	1	Load Output		**OUT**
	0	Do not load Output		
___ IRR	0	Reset IR		**Fetch, HLT**
	1	Do not reset IR		
HLT	1	Halt		**HLT**
	0	**Do not halt**		

7. Circuit Diagram:

Attached at the end of the report


U3
74LS157
U1
74LS163A
V7
5V
+V
PCMU
PCU
V5
5V
+V
IRL
U2
74LS273
AdM3
U10
74LS153
U11
74LS153
AdM2
AdR/PM1
U14
74LS157
U19
74LS126
3-state-2
RAM-U
U16
1K RAM
U6
74LS191
SPALU-U
U8
74LS83
V9
5V
+V
SPU
U5
74LS157
U4
74LS163A
PCML
PCL
U12
74LS153
AdM1
AdR/PM0
U15
74LS157
RAM-L
U17
1K RAM
U22
74LS126
3-state-1
U23
74LS126
3-state-0
U39D
U13
74LS153
AdM0
IR1
U40
74LS163A
IR0
U25
U43B
U41B
U41A
U44A
U41D
U7
74LS191
U20D
SPL
SPALU-L
U46
74LS83
V3
5V
+V
U36C
U39C
U20C
U39B
PROM32
U21
PROM32
U26
PROM32
ROM0
ROM1
ROM2
0V A7
0V A6
0V A5
0V A4
0V A3
5V A2
5V A1
0V A0
Input Address Switch
0V
CLK
5V
RUN



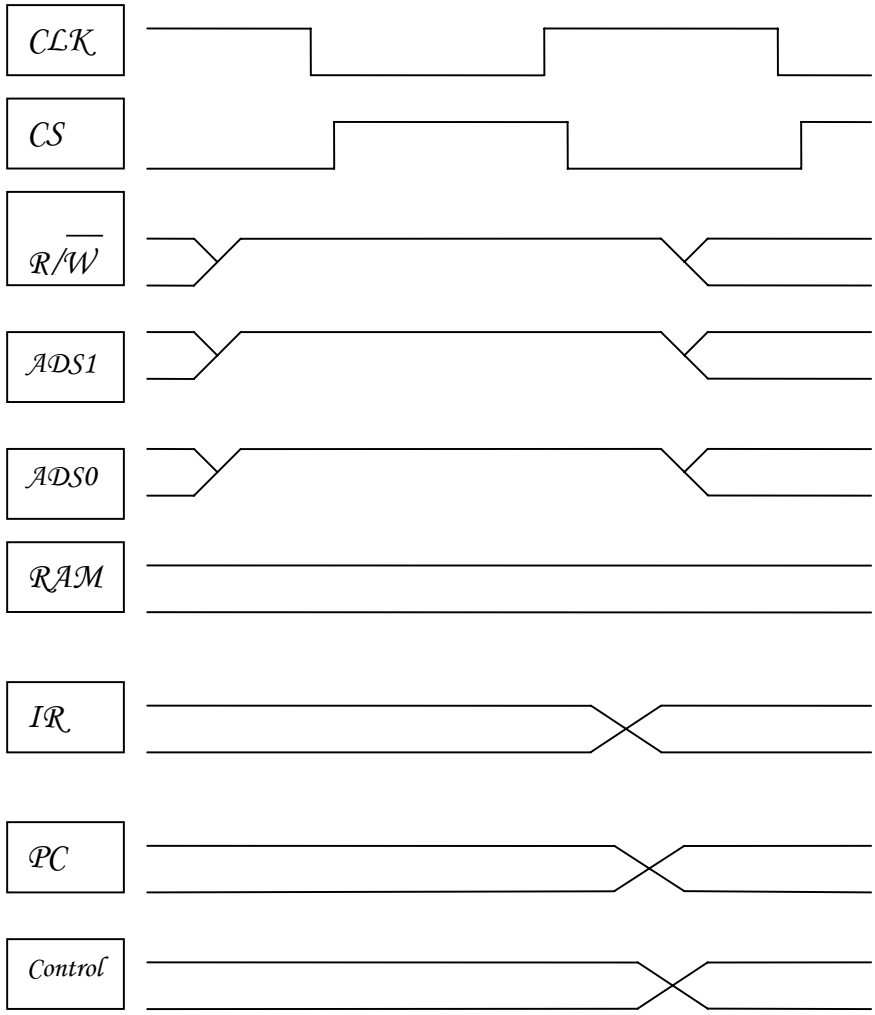
4-bit Computer Design

Group # 01

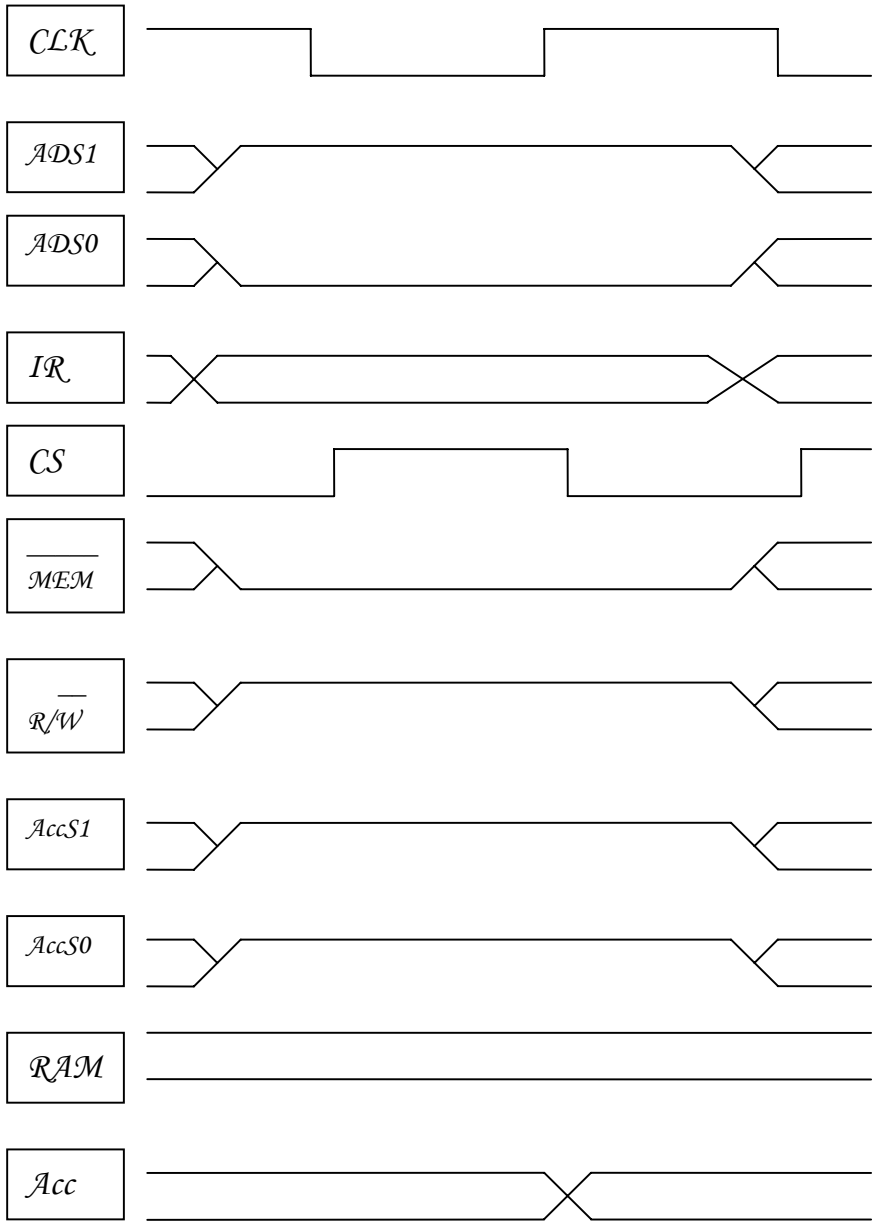
Student no # 0205002
0205003
0205004
0205014
0205026
0005069

8. Timing Diagrams:

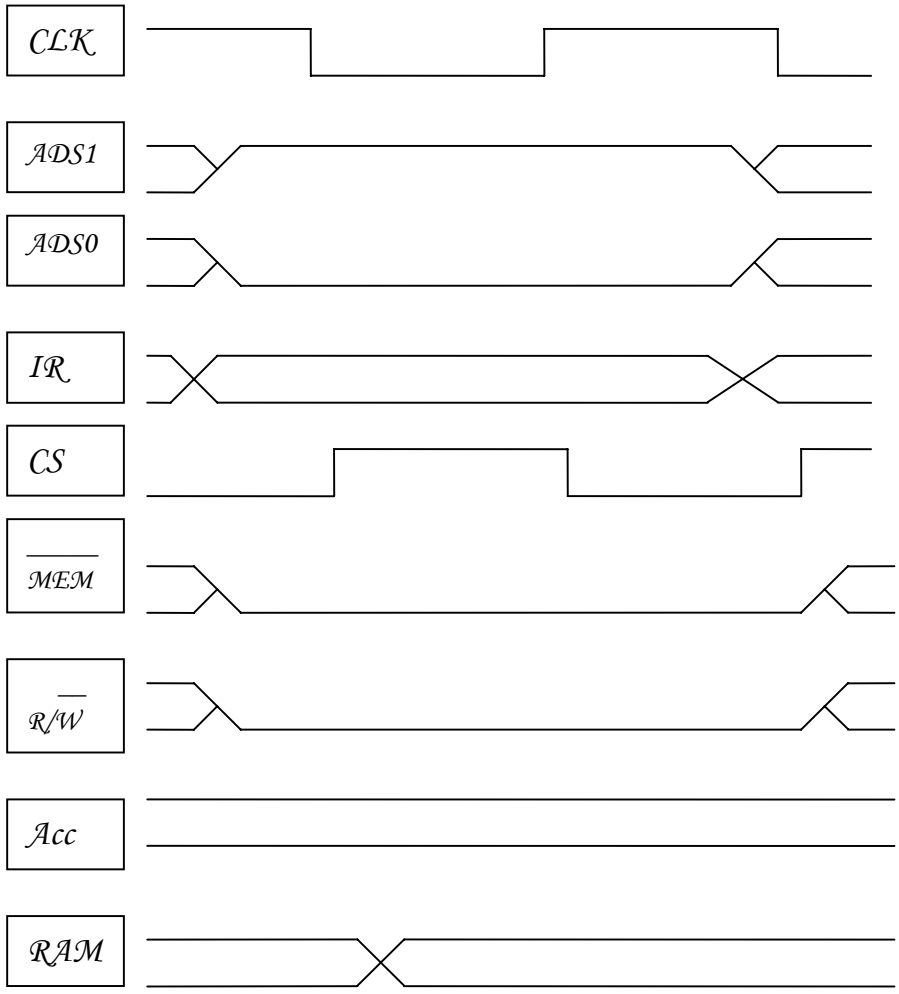
8.1. Fetch:



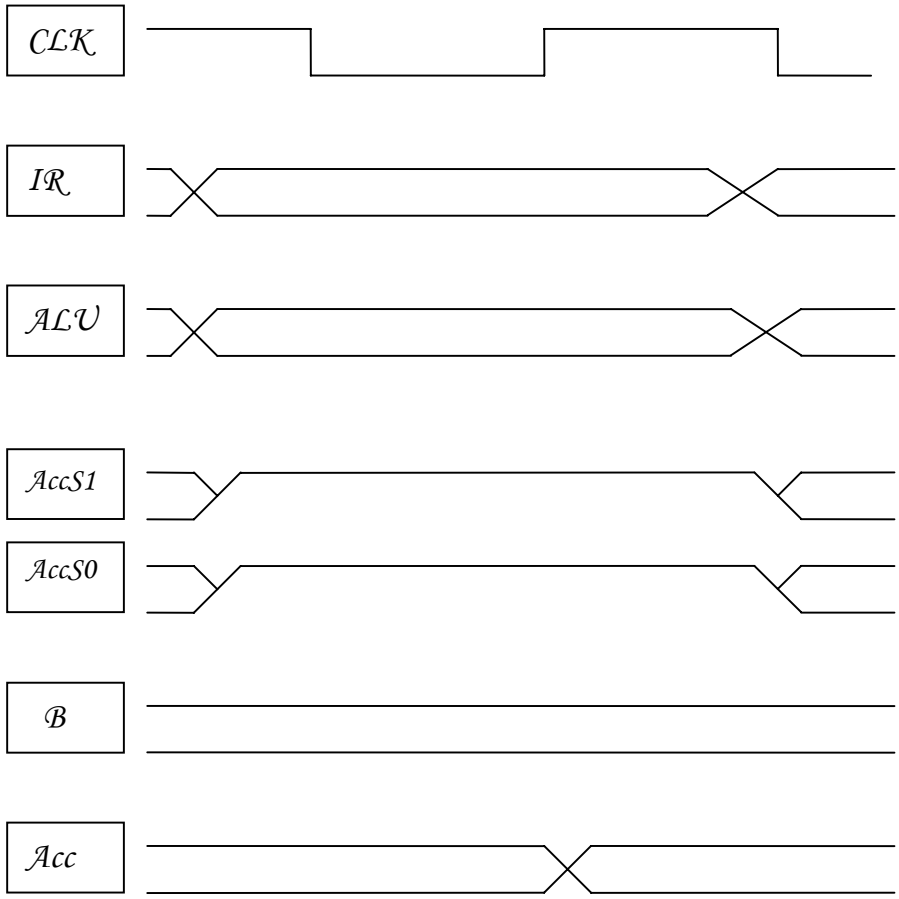
8.2. LDA address:



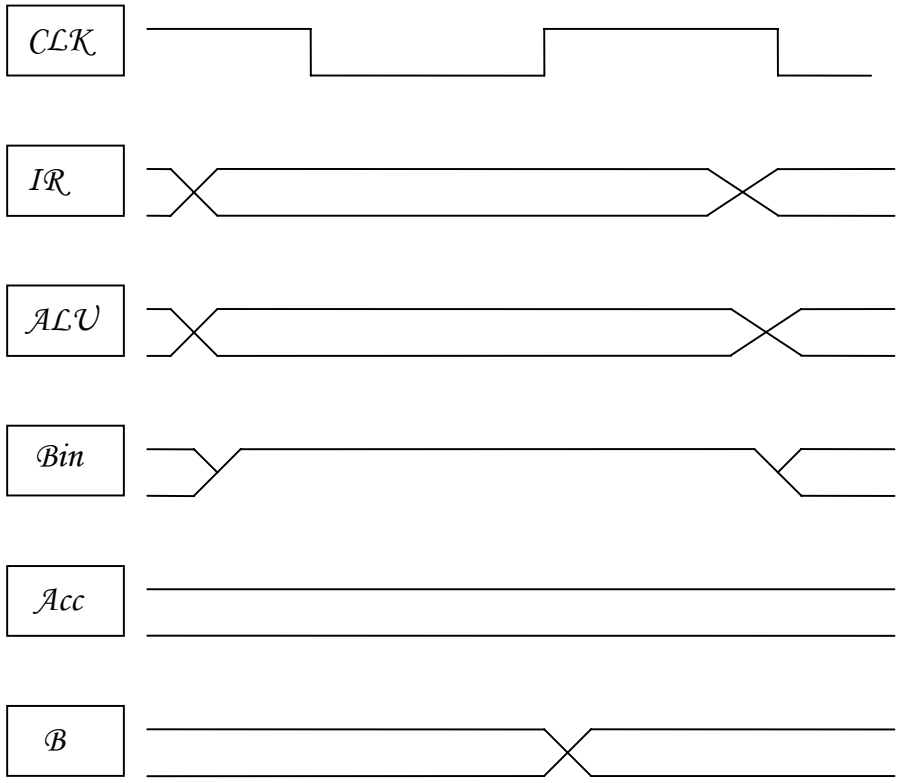
8.3. STA address:



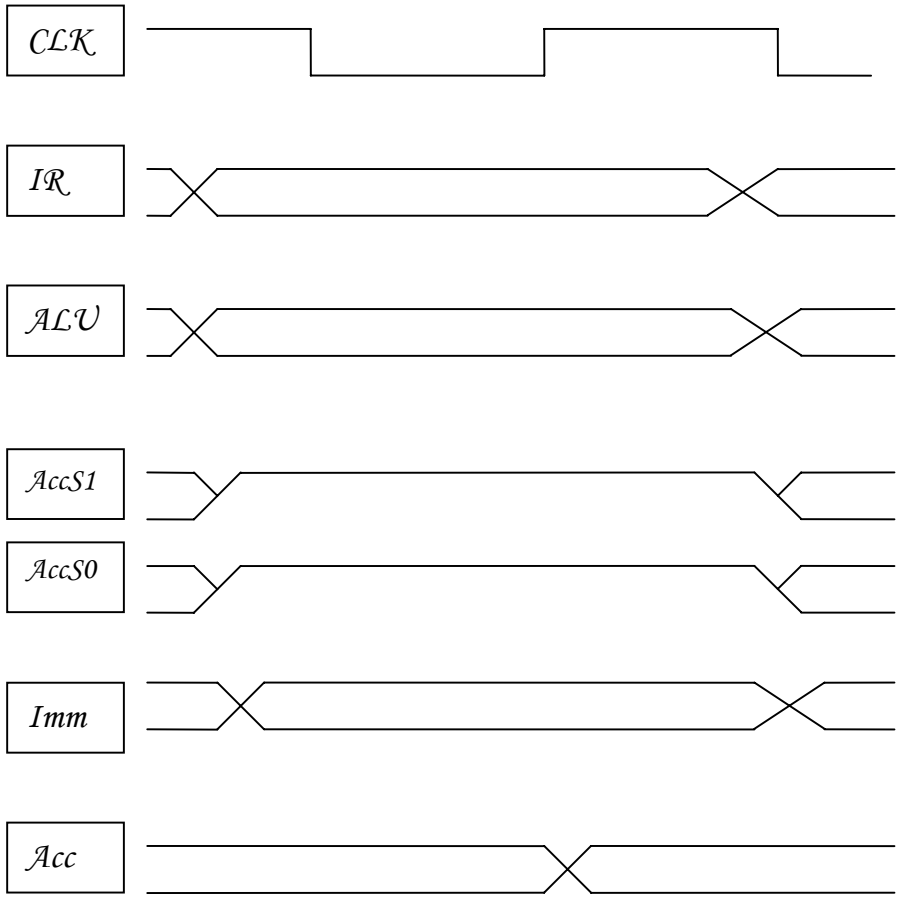
8.4. MOV Acc, B:



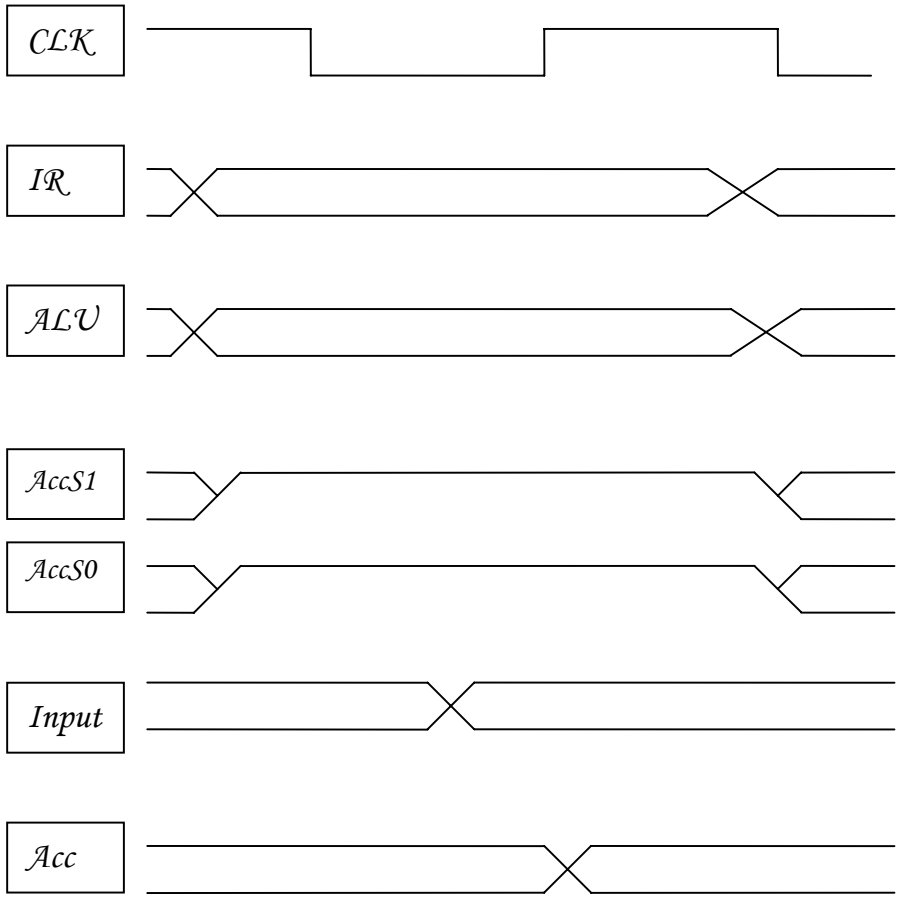
8.5. MOV B, Acc:



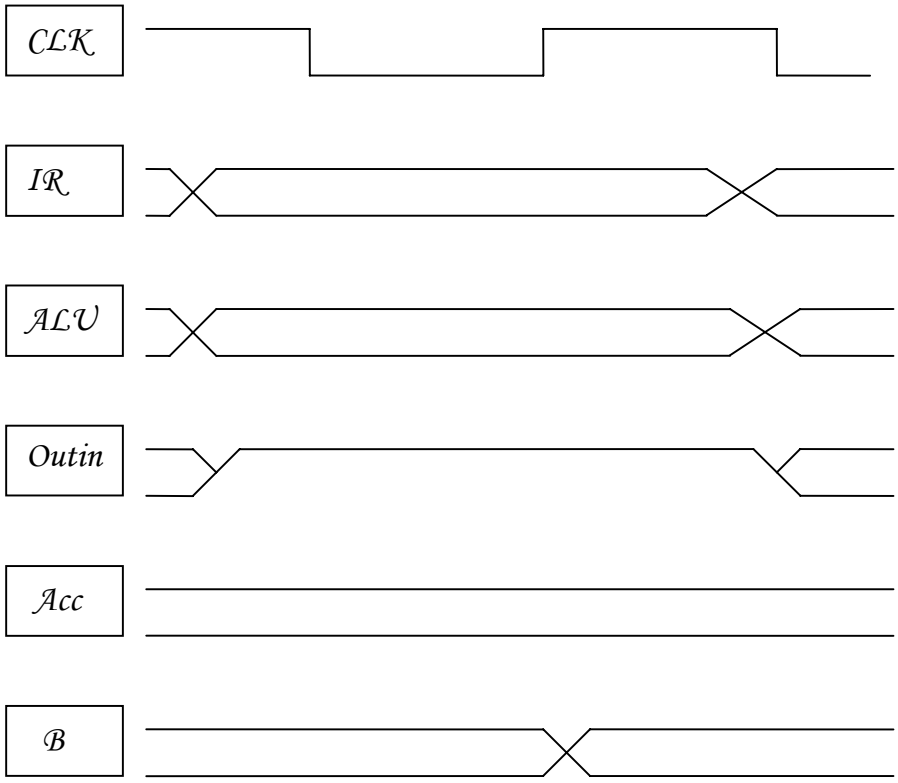
8.6. MOV Acc, immediate:



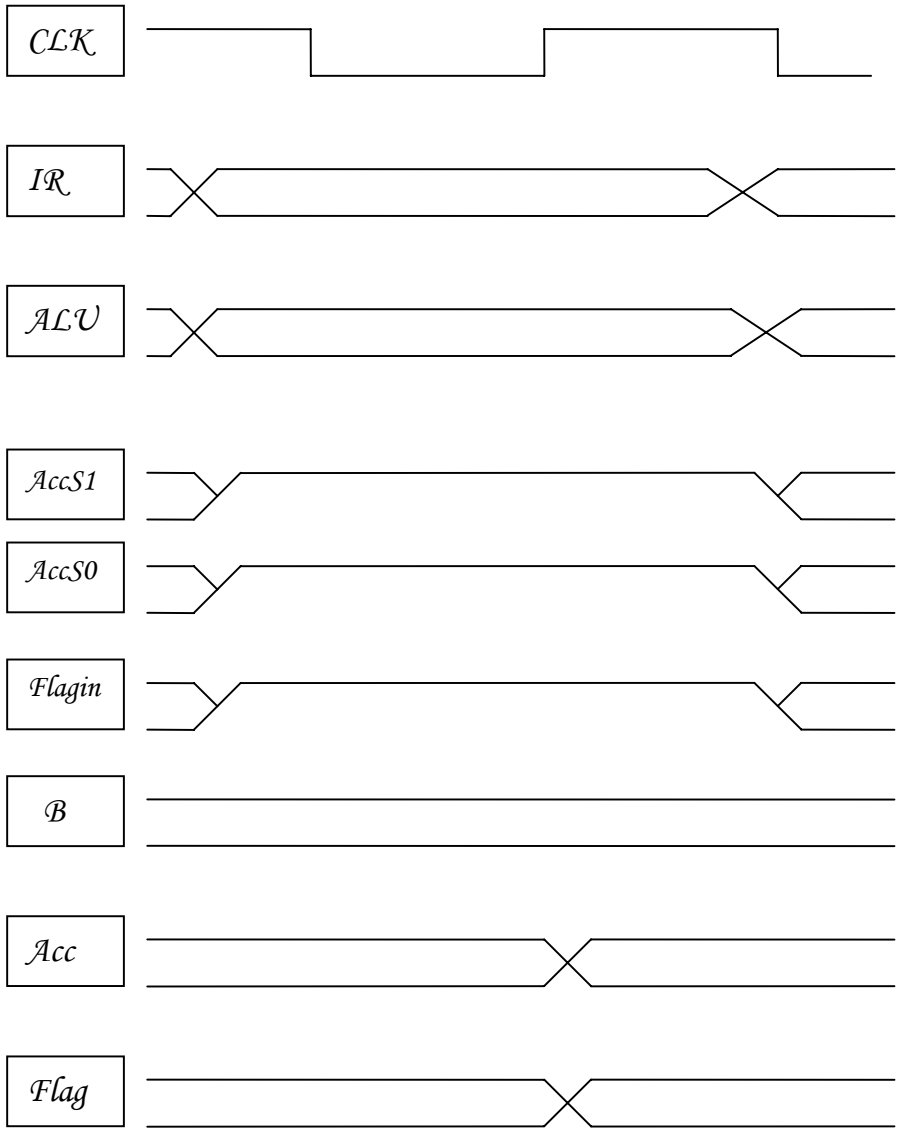
8.7. IN:



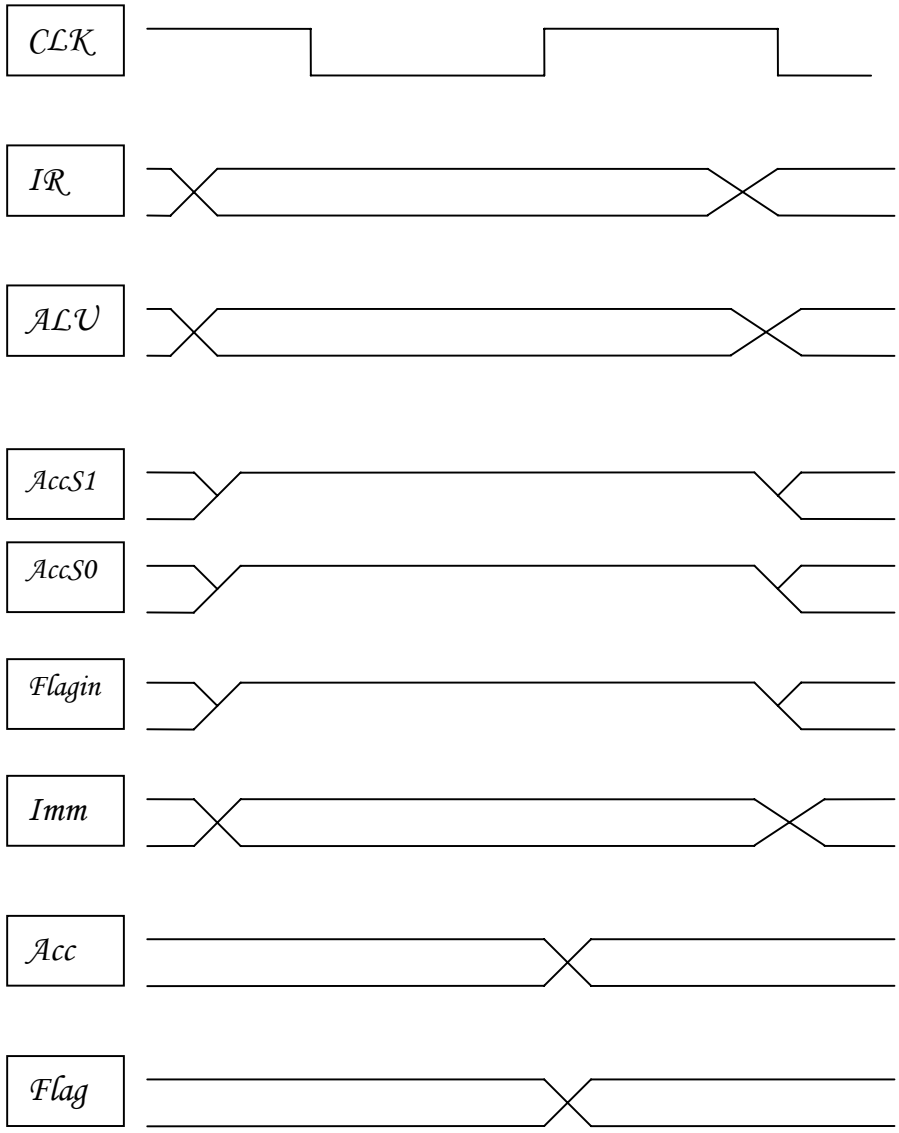
8.8. OUT:



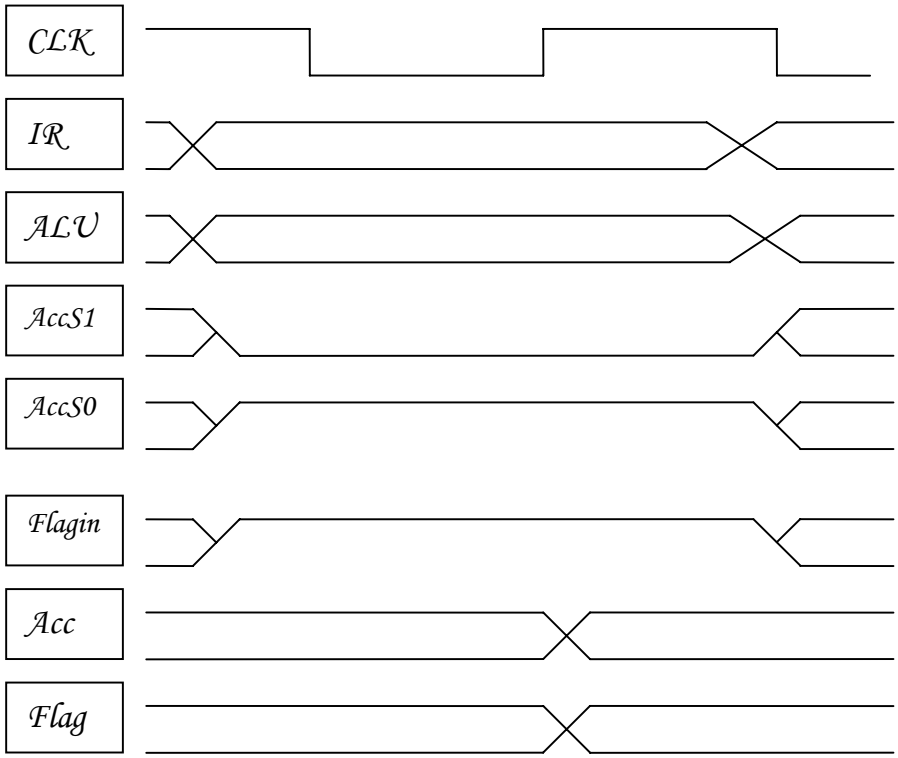
8.9. ADD B / ADC B / SUB B / SBB B / AND B:



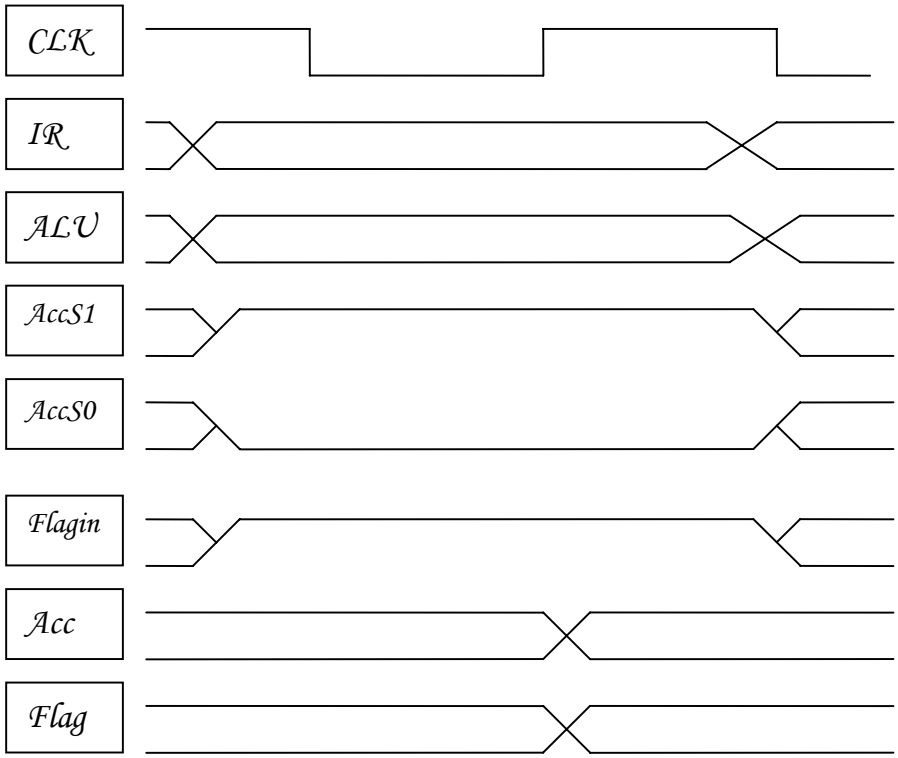
8.10. ADD immediate / XOR immediate:



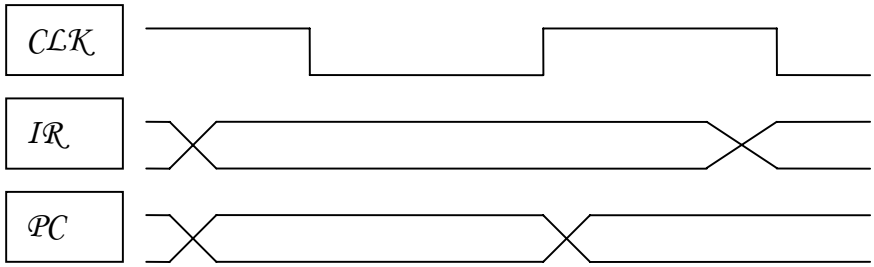
8.11. SHL:



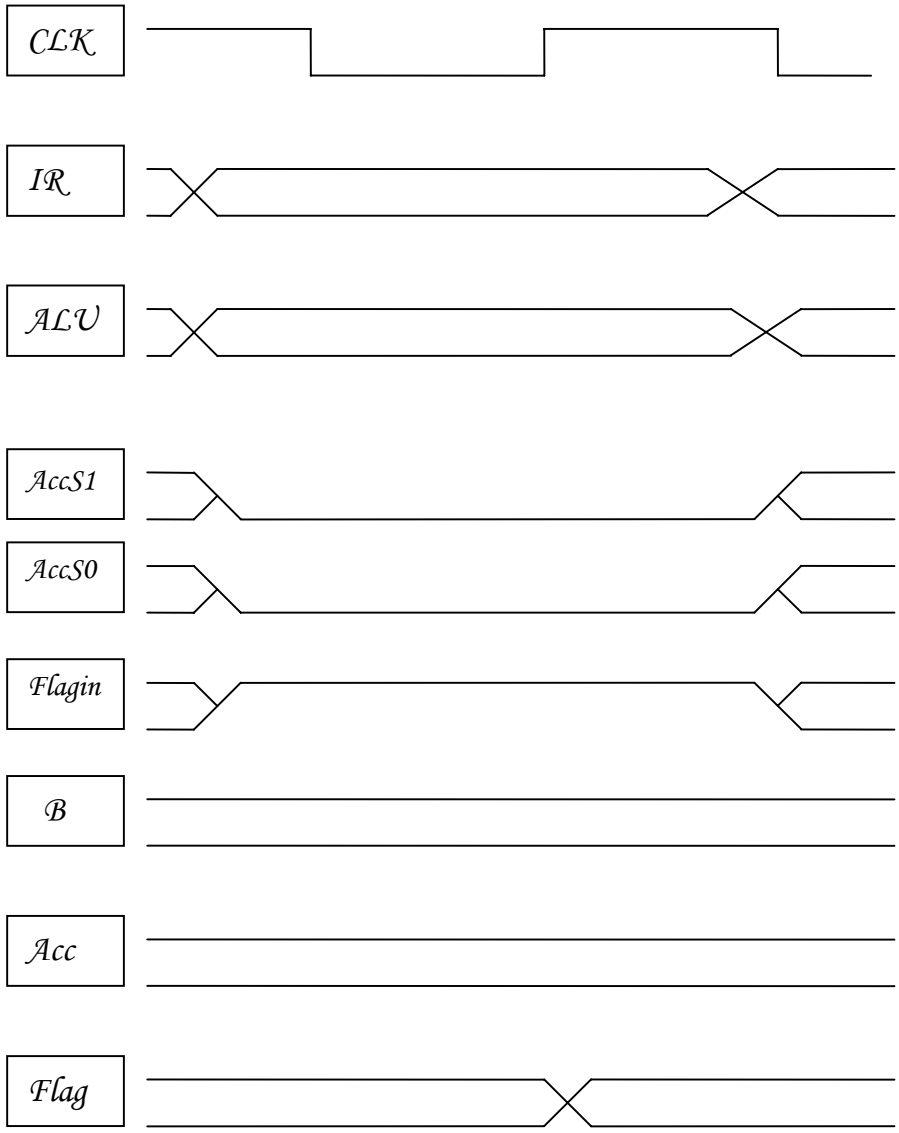
8.12. SHR:



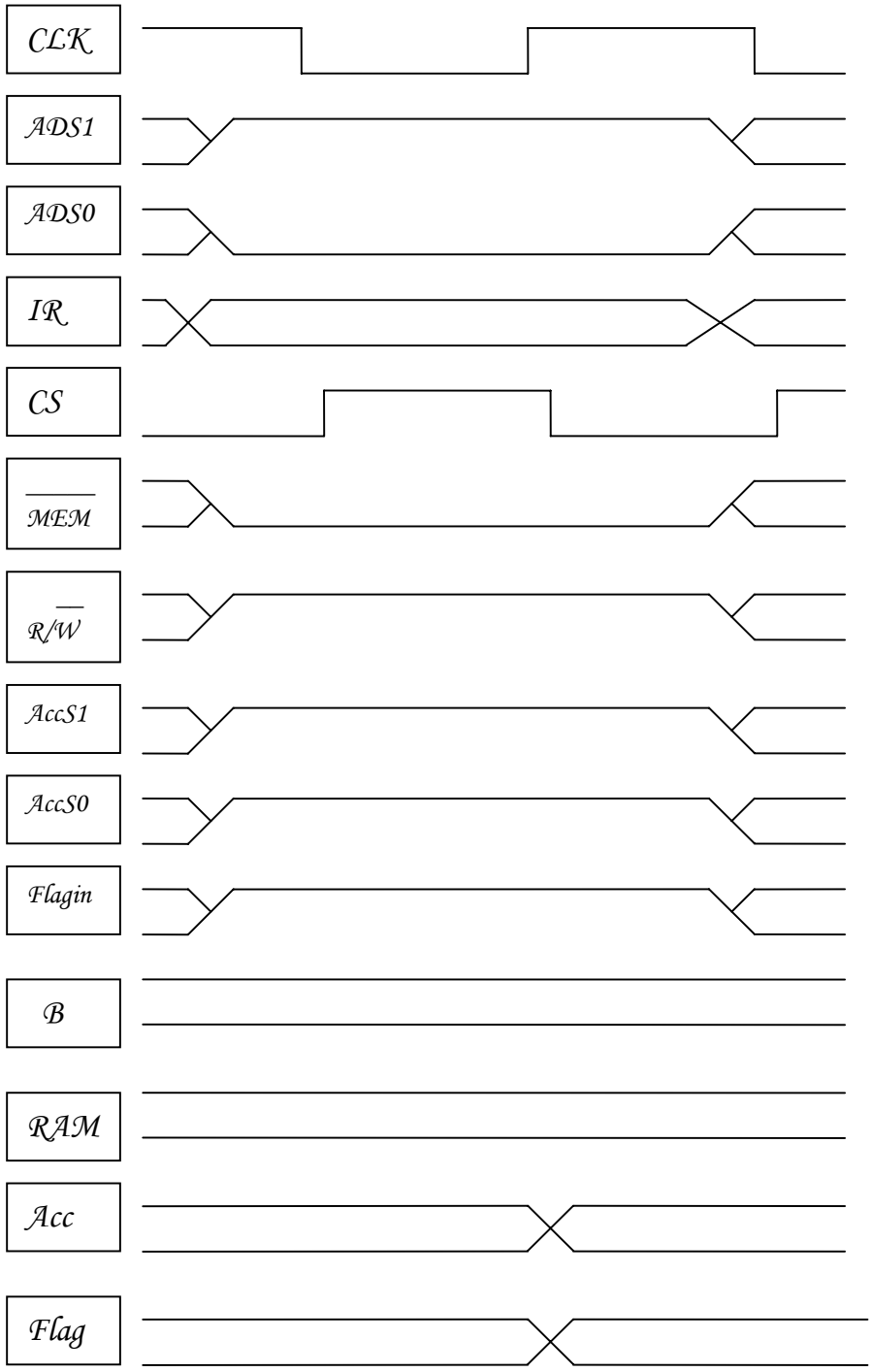
8.13. NOP:



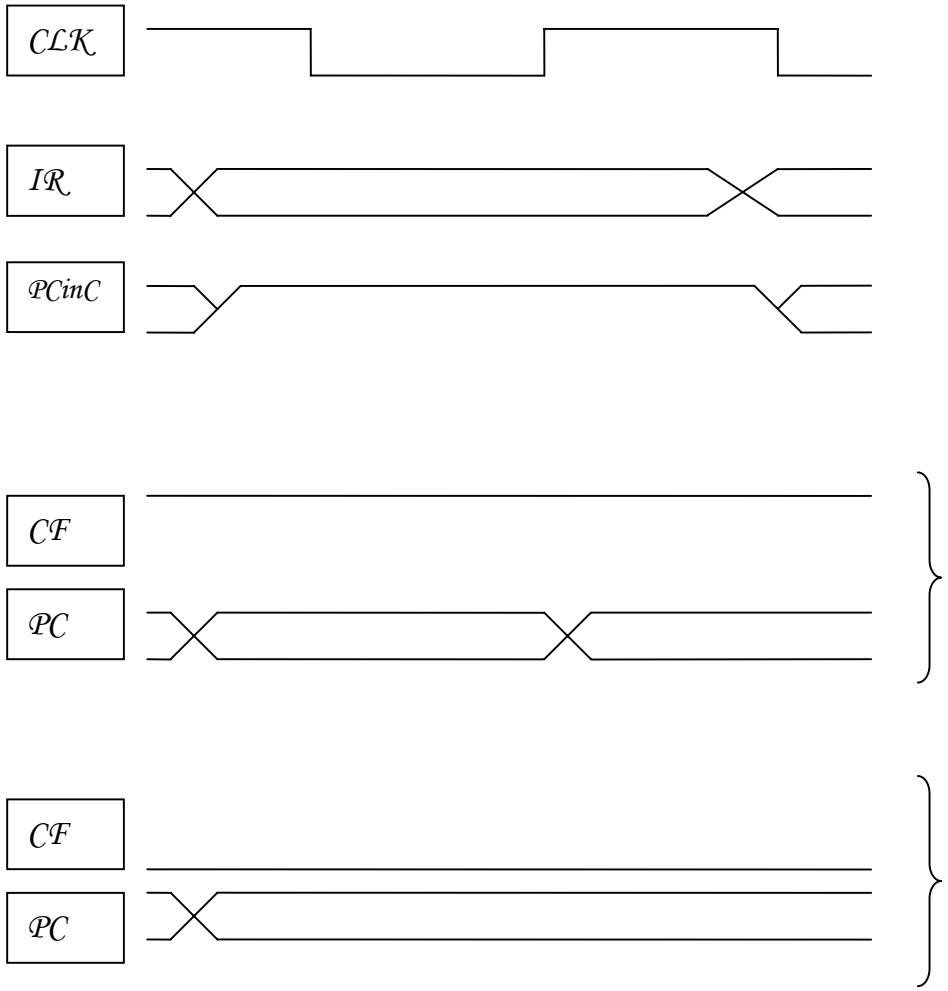
8.14. CMP B / TEST B:



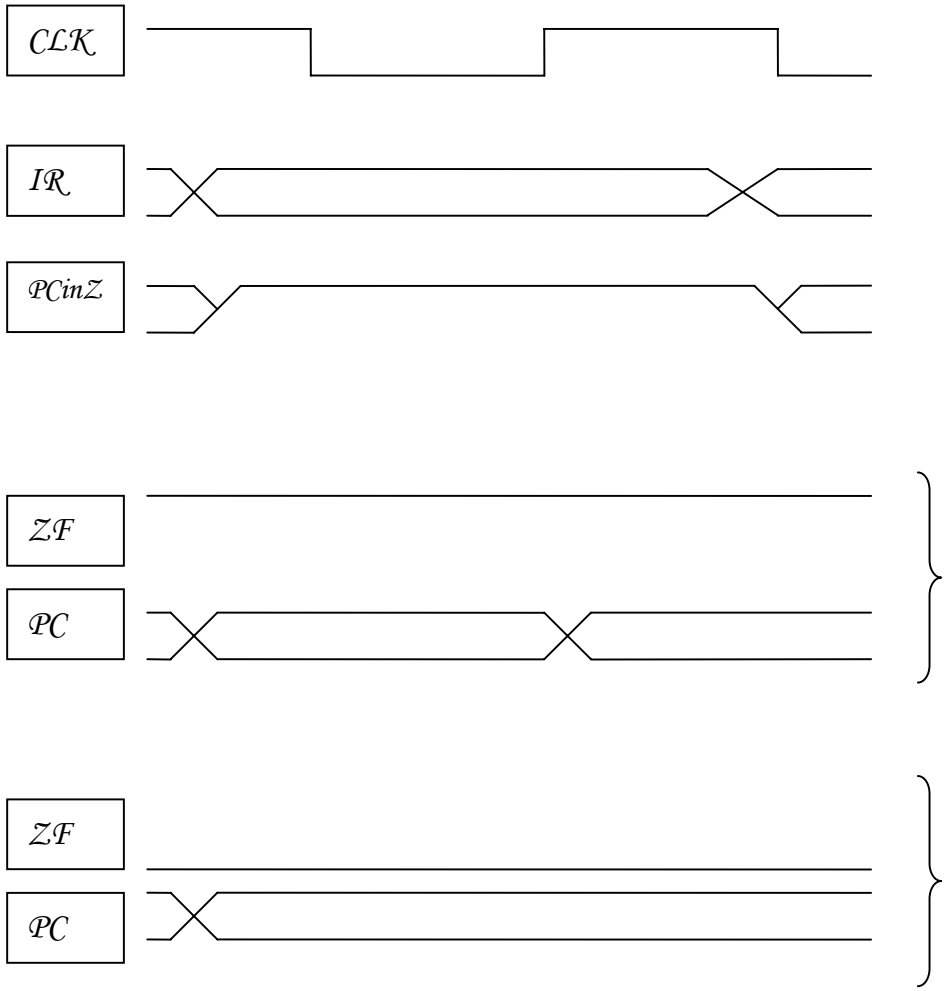
8.15. SUB [address]:



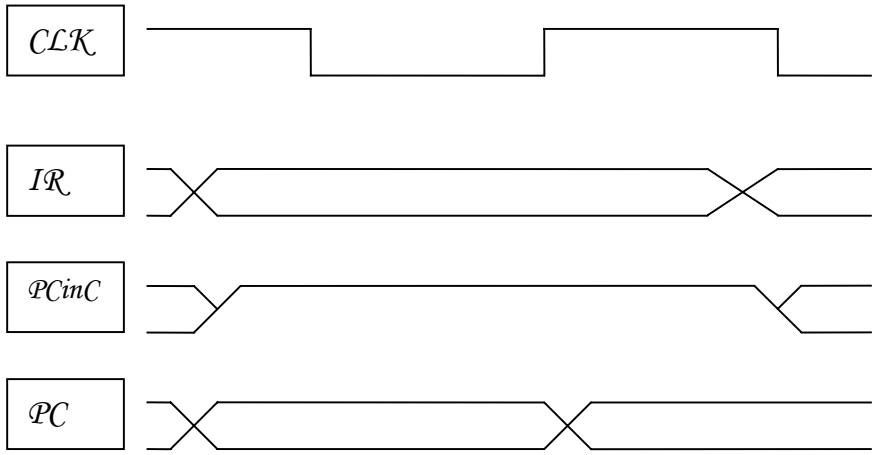
8.16. JC address:



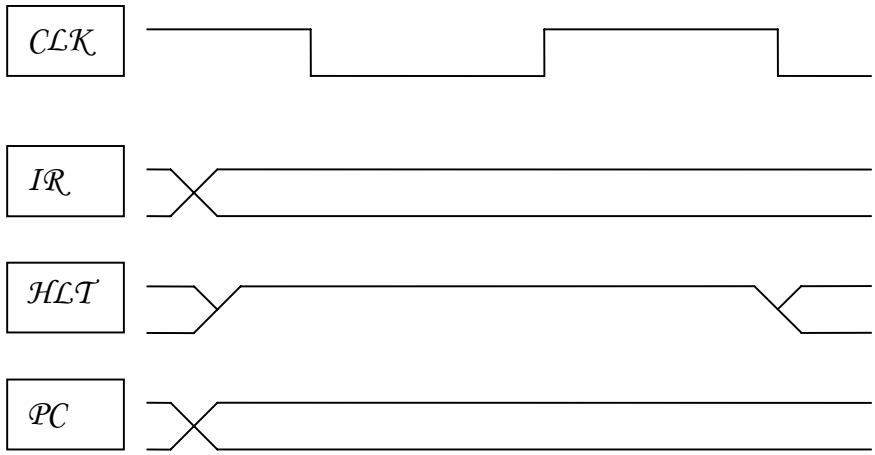
8.17. JE address:



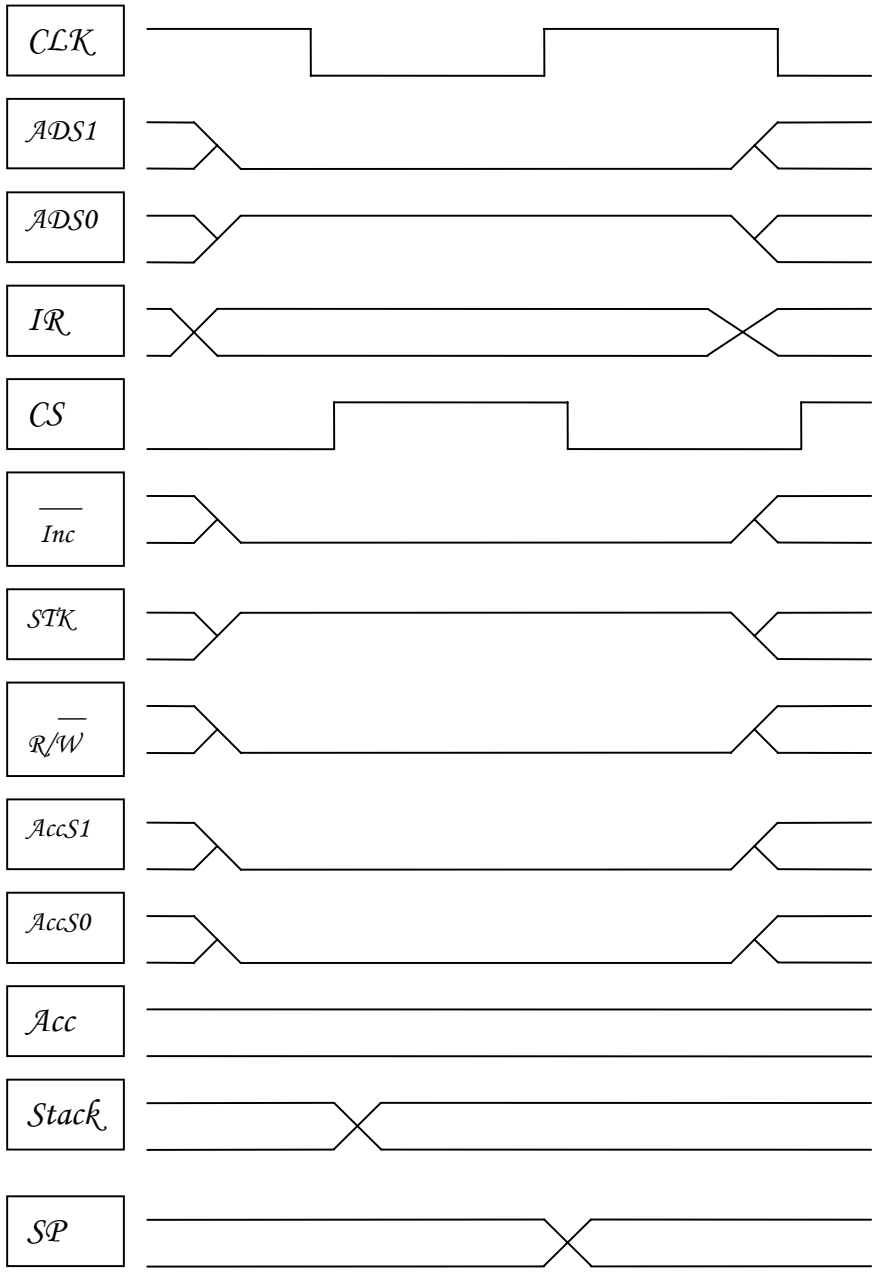
8.18. JMP address:



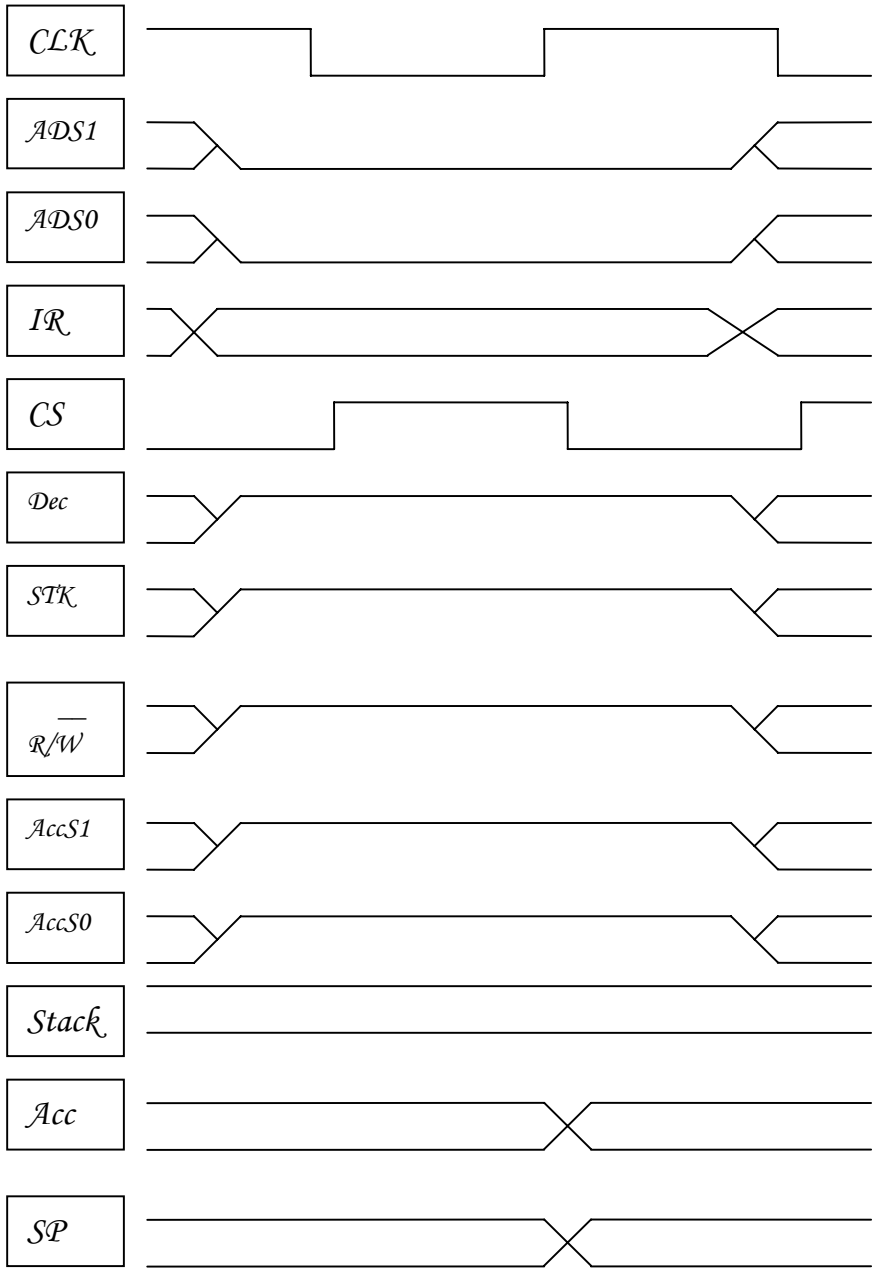
8.19. HLT address:



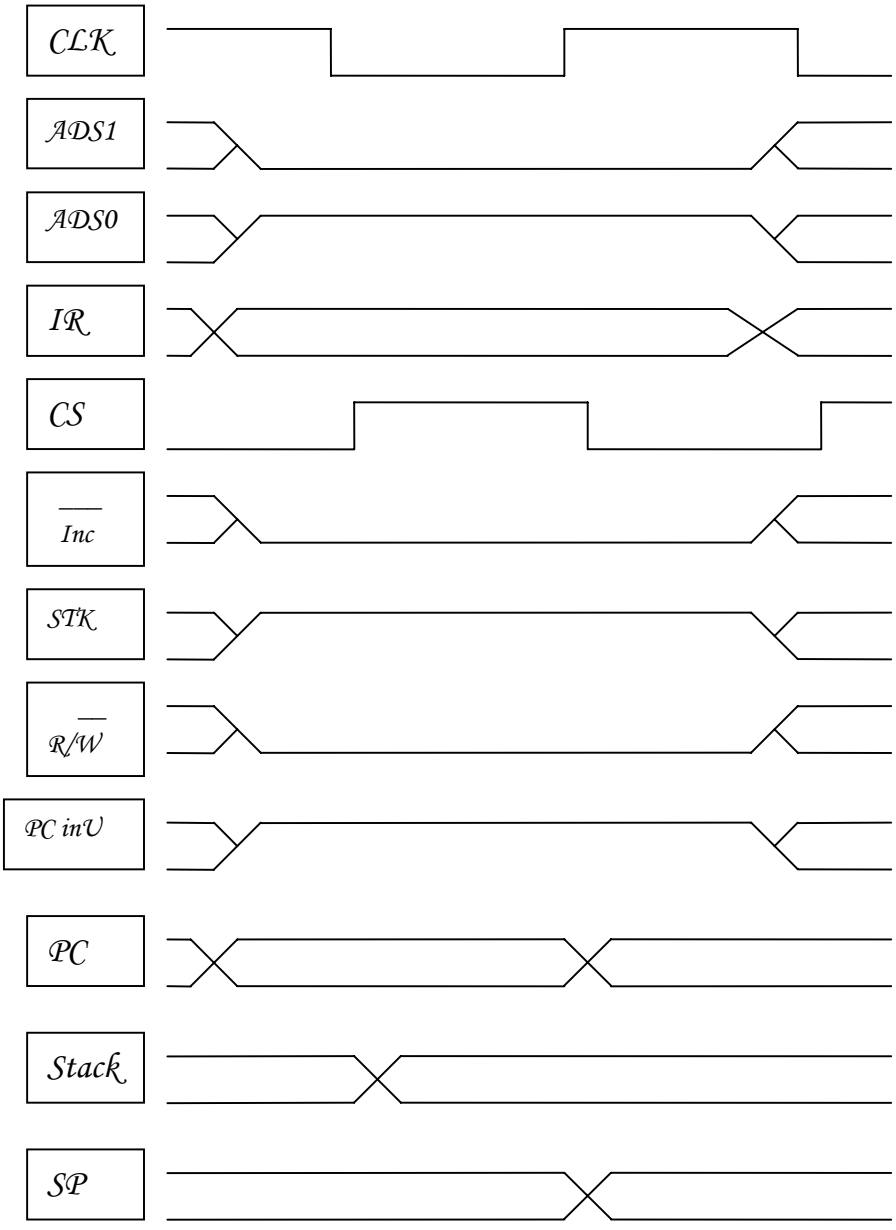
8.20. PUSH:



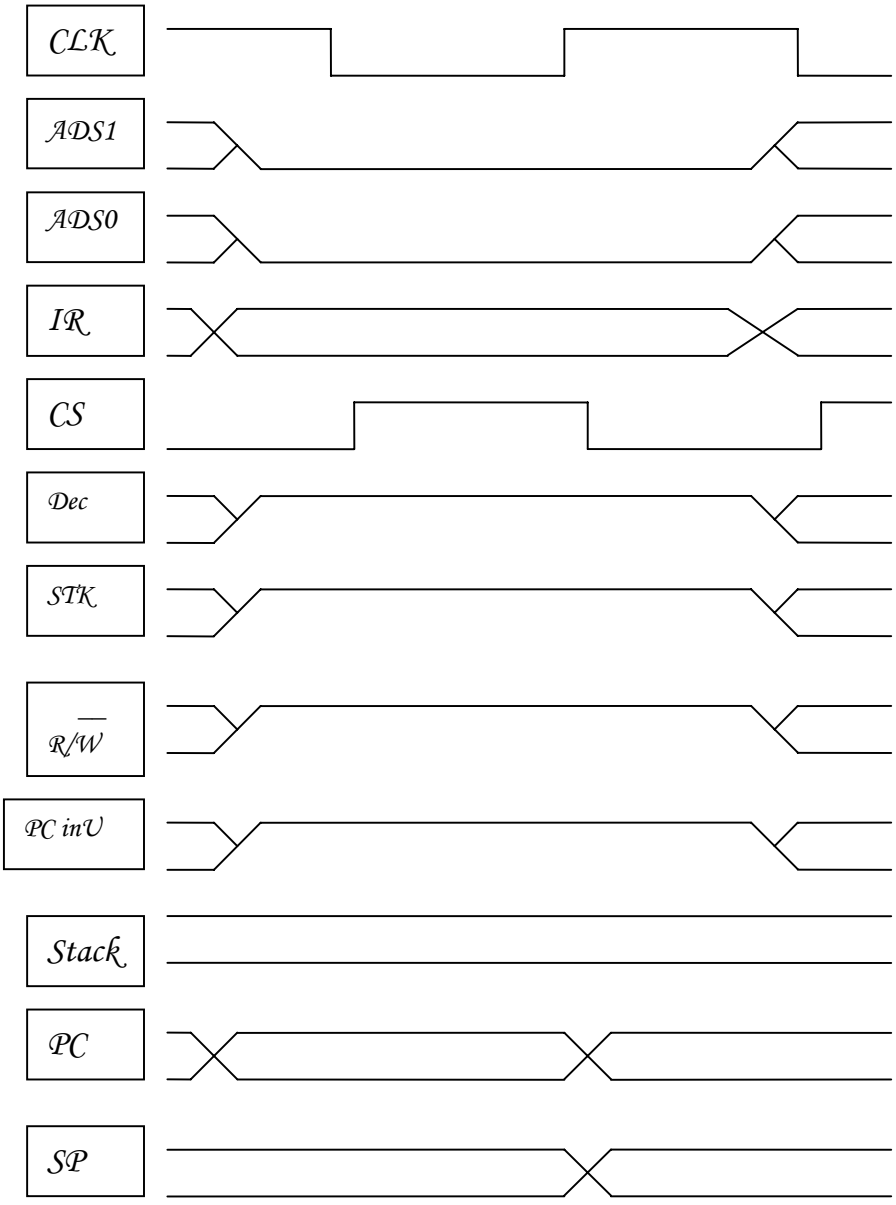
8.21. POP:



8.22. CALL address:



8.23. RET:



9. Explanation of Instructions:

9.1. Fetch:

During fetch cycle, Address MUX selects the PC output to address RAM. Also, Control signals select RAM in reading mode. At the end of fetch the next instruction's Opcode and immediate data/address from the RAM is loaded into IR and the corresponding Control Signals are generated. Finally, the PC value is incremented before the end of the fetch cycle.

9.2. Execution:

During execution cycle Control signals corresponding to the instruction being executed controls the data and control flow of the computer according to the instruction. For every execution cycle except that of HLT, /IRR = 0 is generated and as a result, at the end of the cycle, IR is reset and the Control Signal for the fetch cycle is generated by the Control ROM

9.2.1. LDA address:

During the execution cycle of this instruction, RAM read operation is selected and RAM is addressed by immediate address from IR through the Address MUX. BMUX selects the RAM data to be selected on the B input of the ALU unit. This data is transferred through the ALU unit with the help of ALU0-5 Control Signals and is stored in the Accumulator.

9.2.2. STA address:

For this instruction, RAM write operation is selected and RAM is addressed by the immediate address from IR. The Accumulator data is transferred through ALU unit with necessary ALU0-5 Control Signals and is loaded into the RAM.

9.2.3. MOV Acc, B / MOV Acc, imm / IN:

B MUX selects B / immediate / input port data to go to the B in put of the ALU unit. This data is transferred through ALU and is loaded into Accumulator.

9.2.4. MOV B,Acc / OUT:

The Accumulator data is passed through the ALU and is loaded into the B / Output register.

9.2.5. ADD B / ADC B / SUB B / SBB B / ADD imm / AND B / XOR imm:

BMUX selects B register output / immediate data from IR to pass to the B input of ALU unit. On the A input of the ALU unit, Accumulator data is given. By means of the Control Signals ALU0-5, the corresponding ALU operation is selected and the result is loaded into the Accumulator. Flagin is also set to 1 and for all 4 flags (ZF, SF, OF and CF), the necessary input is generated by the hardware depending on the result of the ALU operation. These values are then loaded into the Flags.

9.2.6. CMP B / TEST B:

BMUX selects the B register output to be on the B input of the ALU unit. On the A input, Accumulator value is supplied and the necessary ALU operation (SUB for CMP and AND for TEST) is selected through ALU0-5. Depending on the result the Flag inputs are generated via hardware and are loaded into the Flag register by setting Flagin into 1. But the ALU output is not loaded into the Accumulator; rather the Accumulator data remains unchanged.

9.2.7. SUB [address]:

RAM read operation is selected and immediate address from IR addresses RAM through Address MUX. The output of the RAM is passed onto the B input of the ALU unit. On the A input, the Accumulator data resides and SUB operation is selected by ALU0-5. The ALU result is then loaded into the Accumulator and depending on the result necessary flag input is generated and loaded into the Flag register.

9.2.8. SHL/SHR:

AccS1 and AccS0 is so set that shift left / shift right operation is selected. Moreover, Accumulator data is transferred through ALU unit and from its output the corresponding Flag input is generated and loaded into the Flag register.

9.2.9. JC address:

PCinC Control signal is set to 1 and as such it enables the parallel loading in the PC counter provided that CF = 1. PCS is also set to 1 so that immediate address from IR is selected through PC MUX. Hence, PC value is loaded with the immediate address when CF = 1 and effectively the control flow jumps to the corresponding address.

9.2.10. JE address:

PCinZ Control signal is set to 1 and as such it enables the parallel loading in the PC counter provided that ZF = 1. PCS is also set to 1 so that immediate address from IR is selected through PC MUX. Hence, PC value is loaded with the immediate address when ZF = 1 and effectively the control flow jumps to the corresponding address.

9.2.11. JMP address:

PCinU Control signal is set to 1 and as such it enables the parallel loading in the PC counter unconditionally. PCS is also set to 1 so that immediate address from IR is selected through PC MUX. Hence, PC value is loaded with the immediate address and effectively the control flow jumps to the corresponding address.

9.2.12. PUSH:

Stack write operation is selected and Stack is addressed by SP with the help of Address MUX. The Accumulator data is transferred through ALU unit and the result is loaded into the Stack. At the end of the cycle, SP is incremented by 1 since /Inc-Dec is kept at 0.

9.2.13. POP:

Stack read operation is selected and Stack is addressed by (SP-1) with the help of Address MUX. This stack output is transferred through ALU unit and the result is loaded into the Accumulator. At the end of the cycle, SP is decremented by 1 since /Inc-Dec is kept at 1.

9.2.14. CALL address:

Stack write operation is selected and Stack is addressed by SP with the help of the Address MUX. PC value is Stored into the Stack during the negative edge of the clock. Meanwhile PCinU is set to 1 so that PC is loaded unconditionally. PCS = 1 selects the immediate address to be loaded into the PC. This address is loaded during the Rising edge of the clock and effectively transfers the control flow at the start of the subroutine addressed by the immediate address. At the end of the cycle SP is incremented by 1.

9.2.15. RET:

Stack read operation is selected and Stack is addressed by (SP -1) with the help of the Address MUX. Meanwhile PCinU is set to 1 so that PC is loaded unconditionally. PCS = 0 selects the Popped address from the Stack to be loaded into the PC. This effectively transfers the control flow to the caller subroutine at the point where it left off. At the end of the cycle SP is decremented by 1.

9.2.16. NOP:

The RAM is in the read mode and for every register. Load is disabled by the Control signal. So, no effective operation is done.

9.2.17. HLT:

HLT signal is generated that stops the PC counting and /IRR is also set to 1 so that IR is not reset to be in the fetch cycle. So, effectively, the computer just halts because it can not change its state.

11. How to Load / Write Programs:

For writing and loading a program into the computer, The RUN-/PROG switch is hold at 0. Then the RAM address is selected via input address switches, the Opcode is given by the input Opcode switches and the immediate data/address is provided by the input data / address switches. Then at each clock pulse one instruction or one data is loaded into the memory

12. How to Run / Execute Programs:

Each instruction in our computer takes only two cycle to run: one cycle for being fetched and the other for being executed. When the RUN-/PROG input is at 1 level, the computer is in running mode. At the start of the running mode, the PC value is 0. Then, at each pair of clock pulse, one instruction is executed until a HLT instruction is executed.

13. Special Features:

1. The Computer runs each instruction in only two clock cycles: one for fetching and one for executing
2. RAM needs 200ns for its read/write. Hence, 250 ns for each half clock cycle is a safe estimate. This fact ensures that the computer can run as fast as
3. 1 / 2 * 250ns= 2 MHz
4. Data length is 4 bit. for unsigned operations data range is 0 to 15 and for signed operation the range is-8 to +7
5. 8 bit addressing makes up 256 blocks of available memory
6. Practically infinite stack. Stack is also addressed by 8 bits. So, 256 blocks of stack space is provided for 256 blocks of memory, which is normally more than enough.
7. 4 Flag is kept to provide the user with the information regarding the status of the computer. These 4 Flags are Sign Flag (SF), Zero Flag (ZF), Overflow Flag (OF) and Carry Flag (CF).

14. ICs used with Count:

IC number	Description	Count
7404	Hex Inverter	**2**
7408	Quad 2-input AND	**2**
7427	Tri 3-inout NOR	**1**
7432	Quad 2 input OR	**1**
7486	Quad 2-input XOR	**2**
7483	4-bit full Adder	**2**
74126	Quad 4-bit buffer	**3**
74153	Dual 4x1 MUX	**10**
74157	Quad 2x1 MUX	**4**
74163	4 bit Up Counter with Synchronous reset and load	**4**
74181	4-bit ALU	**2**
74191	4-bit Up-down Counter with Asynchronous load	**2**
74194	4-bit universal shift register with synchronous load and asynchronous reset	**4**
74257	Quad 2x1 tri-state MUX	**1**
74273	Octal D flip-flop with asynchronous reset	**1**
6116	2K x 8-bit RAM	**2**
2716	2K x 8-bit EPROM	**3**
Total		**46**

15. Discussion:

The design and implementation challenges were wonderful, excellent although sometimes they created almost hopeless situations. We thoroughly enjoyed the task and furthermore this assignment gave us the satisfaction of facing some challenges worthy of taking in real life as a hardware designer. However, it has peeped into our mind that as a hardware designer, it might have been enough to do the design with necessary simulation. The implementation challenges are sometimes unnecessarily imposed. In spite of that, we were thrilled in every bit of the making of the computer.