GUJARAT TECHNOLOGICAL UNIVERSITY

Government Engineering College, Bhavnagar

Subject: **Microprocessor & Interfacing(ASS)**

B.E. C.E. Semester-6th

(Computer Branch)

Submitted By:

Name: Vankani Arjun BakulBhai

Enrollment: 180210107060

Prof. Hardi Sanghavi
(Faculty Guide)

Prof. KARSHAN KANDORIYA
(Head of the Department)

Academic Year (2020-21)

SR No.	(ASSIGNMENT work)	Page No
1	Data Transfer Instructions	03
2	Arithmetic Instructions	04
3	Logical Instructions	05
4	Branching Instructions	06
5	Stack & I/O Instructions	08
6	Assembly Language Program 1	9
7	Assembly Language Program 2	12
8	Timing Diagram	17

Data Transfer Instructions

Instruction Description	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags
1) MOV	**Move**	MOV MOV => R1,R2 – Register~ MOV B,C| R,M- Indirect ~ MOV H,M | M,R Direct~ MOV M,L	1 Byte(8bit)	None
		The MOV instruction copies the contents of the source register or contents stored at location pointed by HL pair (M) into destination register or location pointed by the HL pair (M).		
2) MVI	**Move immediate**	MVI MVI => R - immediate register~ MVI H,60H | M immediate memory~ MVI M,60H	2 Byte(16bit)	None
		The MVI instruction is used to store 8-bit data into destination register or memory location, which is pointed by HL pair.		
3) LXI	**Load immediate 16-bit**	LXI LXI => R,D Immediate ~ LXI H,2841 [H] <-28, [L] <- 45H	3 Byte(24bit)	None
		This instruction loads a 16-bit immediate data in the memory location pointed by the HL pair (M).		
4) LDA	**Load Accumulator 16-bit**	LDA LDA =>16-bit address - Direct Addressing Mode~ LDA 2813H	3 Byte(24bit)	None
		The LDA instruction used to copy the contents from the address to accumulator.		
5) LDAX	**Load HL register**	LDAX LDAX => B/D register pair - Register Indirect ~ LDAX B | LDAX D	3 Byte(24bit)	None
		The contents of the mentioned register pair point to a memory location. LDAX instruction copies the contents of that particular memory location into the accumulator register.		
6) LHLD	**Load HL register pair**	LHLD LHLD =>16-bit address - LHLD Memory ~ LHLD 2187H	3 Byte(24bit)	None
		The LHLD instruction copies the contents of a specified memory location pointed out by a 16-bit address into register L. The contents of the next memory location are copied into register H.		
7) STA	**Store Accumulator**	STA STA => 16-bit address – Direct addressing Mode ~ STA 2186H	3 Byte(24bit)	None
		The STA instruction used to copy the contents of accumulator to the memory location, which is specified by operand		
8) STAX	**Store Accumulator 16-bit**	STAX STAX => Register pair – Indirect addressing Mode ~ STAX B | STAX D	1 Byte(8bit)	None
		The contents of the accumulator register are copied into the memory specified by the register pair in the operand.		
9) SHLD	**Store HL Register pair**	SHLD SHLD => 16-bit address – Direct addressing Mode ~ SHLD 2186H	3 Byte(24bit)	None
		The first 8-bit contents of the register L is stored into the memory location specified by the 16-bit address. The next 8-bit contents of the register H are stored into the subsequent memory location.		
10) XCHG	**Exchange HL with DE**	XCHG None - Implicit Addressing Mode ~ XCHG	3 Byte(24bit)	None
		The contents of register pair H-L are exchanged with the contents of the register-pair D-E. The information stored in register H is exchanged with that of D; similarly, that in register L is exchanged with the contents of the register E		

Arithmetic Instructions

	Instruction	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags	Description
1) ADD	Addition	ADD	ADD => R – Register ~ ADD B | M - Indirect ~ ADD M	1 Byte(8bit)	ALL(Z,S,P,C,AC)	The contents of any register or memory location are added to the contents of the accumulator register. The corresponding sum is then saved in the accumulator. If the second addend is a memory, its address is specified by the H-L pair.
2) ADC	Add with Carry	ADC	ADC => R – Register ~ ADC B | M indirect ~ ADC M	1 Byte(8bit)	ALL(Z,S,P,C,AC)	The contents of any register or memory location are added to the contents of the accumulator register. The corresponding sum is then saved in the accumulator. If the second addend is a memory, its address is specified by the H-L pair.
3) ADI	Add immediate	ADI	ADI => 8-bit immediate - Immediate addressing mode~ ADI 12H	2 Byte(16bit)	ALL(Z,S,P,C,AC)	The contents of the accumulator register are added with immediate 8-bit data using the ADI instruction. The sum is thus stored in the accumulator
4) ACI	Add immediate Carry	ACI	ACI =>8-bit data - Immediate Addressing Mode ~ ACI 16H	2 Byte(16bit)	ALL(Z,S,P,C,AC)	ACI is similar to the ADC instruction. The immediate value of length 8-bit is added to the contents of the accumulator register along with the carry value. Thus, the Carry flag plays an important role and becomes the third addend.
5) DAD	Add register pair with HL	DAD	DAD => register pair - Register Addressing Mode ~ DAD B	1 Byte(8bit)	Only carry Flag	The 16-bit contents of any 8085 register pair are added to the contents of the H-L register pair. The result is also stored in the H-L pair. The carry flag is also updated with the result of the addition operation
6) SUB	Subtraction	SUB	SUB => R – Register Addressing Mode ~ SUB B | M - Indirect ~ SUB M	1 Byte(8bit)	ALL(Z,S,P,C,AC)	The contents of any register or memory location are added to the contents of the accumulator register. The corresponding sum is then saved in the accumulator. If the second addend is a memory, its address is specified by the H-L pair.
7) SBB	Subtraction with borrow	SBB	SBB => =>R – Register Addressing Mode ~ SBB B | M - Indirect ~ SBB M	1 Byte(8bit)	ALL(Z,S,P,C,AC)	The contents of any register or memory location are added to the contents of the accumulator register. The corresponding sum is then saved in the accumulator. If the second addend is a memory, its address is specified by the H-L pair.
8) SUI	Sub immediate	SUI	SUI => 8-bit immediate - Immediate addressing mode~ SUI 12H	2 Byte(16bit)	ALL(Z,S,P,C,AC)	The contents of the accumulator register are subtracted with immediate 8-bit data using the SUI instruction. The answer is thus stored in the accumulator.
9) SBI	Sub immediate with borrow	SBI	SBI => 8-bit immediate - Immediate addressing mode ~ SBI 56H	2 Byte(16bit)	ALL(Z,S,P,C,AC)	The contents of the accumulator register are subtracted with immediate 8-bit data AND the borrow value (specified by the CY flag) using the SBI instruction. The answer is thus stored in the accumulator.
10) INR	Increment instruction	INR	INR => R – Register ~ INR B | M indirect ~ INR M	1 Byte(8bit)	All except carry flag	The INR instruction is used to increment/increase the contents of a register or of the data at the location pointed by HL pair (M) by one. The result is stored in the same location itself. In the case of the operand being M, the value at the memory location given by the HL pair is incremented by one
11) INX	Increment register pair by one	INX	INX => R – Register Addressing Mode ~ INR B	1 Byte(8bit)	None	The INX instruction increments the contents of a register-pair by the value of one. The result is stored in the same place itself.
12) DCR	Decrement instruction	DCR	DCR => R – Register ~ DCR B | M indirect ~ DCR M	1 Byte(8bit)	All except carry flag	The DCR command is used to decrement/decrease the contents of a register or of a memory location by the value of one. The result after this operation is stored in the same location itself.
13) DCX	Decrement register pair by one	DCX	DCX => R – Register Addressing Mode ~ DCX B	1 Byte(16bit)	None	The DCX instruction decrements the contents of the register-pair by the value of one. The result is stored in the same place itself.

Logical Instructions

	Instruction Description	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags
1) CMP	Compare	CMP	CMP => R – Register ~ CMP B | M - Indirect ~ CMP M	1 Byte(8bit)	Carry, Zero Flag
	The CMP instruction compares the contents of accumulator and the content of register H or location pointed by HL pair. If the contents of the accumulator are < R/M → C=1, == R/M → Z=1, > R/M → C=0 and Z=0.				
2) CPI	Compare immediate	CPI	CPI => 8-bit immediate – Immediate addressing mode ~ CPI 2184H	1 Byte(8bit)	Carry, Zero Flag
	CPI instruction immediate compares the content of accumulator and 8-bit immediate data.I f the contents of the accumulator are < R/M → C=1, == R/M → Z=1, > R/M → C=0 and Z=0.				
3) ANA	Logical AND	ANA	ANA => R – Register ~ ANA B | M - Indirect ~ ANA M	1 Byte(8bit)	ALL(Z,S,P,C,AC)
	The contents of a register or any memory location are logically AND with the contents stored in the accumulator register. The resulting answer is saved in the accumulator. If the operand happens to be a memory location, then its address is mentioned by the contents of the H-L pair.				
4) ANI	Logical AND immediate	ANI	ANI =>8-bit data - Immediate Addressing Mode ~ ANI 16H	2 Byte(16bit)	ALL(Z,S,P,C,AC)
	The ANI instruction works exactly like the ANA instruction but performs logical AND of 8-bit immediate value with the contents of the accumulator register.				
5) XOR	Logical XOR	XRA	XRA => R – Register ~ XRA B | M - Indirect ~ XRA M	1 Byte(8bit)	ALL(Z,S,P,C,AC)
	The contents of a register or any memory location are logically XOR with the contents stored in the accumulator register. The resulting answer is saved in the accumulator. If the operand happens to be a memory location, then its address is mentioned by the contents of the H-L pair.				
6) XRI	Logical XOR immediate	XRI	XRI => 8-bit immediate - Immediate addressing mode~ XRI 16H	1 Byte(8bit)	ALL(Z,S,P,C,AC)
	The XRI instruction works exactly like the XRA instruction but performs logical XOR of 8-bit immediate value with the contents of the accumulator register.				
7) ORA	Logical OR	ORA	ORA =>R – Register Addressing Mode ~ ORA B | M - Indirect ~ ORA M	1 Byte(8bit)	ALL(Z,S,P,C,AC)
	The contents of a register or any memory location are logically OR with the contents stored in the accumulator register. The resulting answer is saved in the accumulator. If the operand happens to be a memory location, then its address is mentioned by the contents of the H-L pair.				
8) ORI	Logical OR immediate	ORI	ORI => 8-bit immediate - Immediate addressing mode~ ORI 12H	2 Byte(16bit)	ALL(Z,S,P,C,AC)
	The ORI instruction works exactly like the ORA instruction but performs logical OR of 8-bit immediate value with the contents of the accumulator register				
9) RLC	Rotate Accumulator Left	RLC	RLC => Implicit Addressing Mode ~ RLC	1 Byte(8bit)	Carry Flag
	The RLC instruction causes each binary bit in the accumulator register to be rotated by one position to its left. The MSB value is shifted to the LSB as well as the Carry Flag in the PSW. The other PSW bits, such as S, Z, P, or AC, are not affected by this operation.				
10) RAL	Rotate Accumulator Left with carry	RAL	RAL => Implicit Addressing Mode ~ RAL	1 Byte(8bit)	All except carry flag
	The RRC instruction causes each binary bit in the accumulator register to be rotated by one position to its right. The LSB value is shifted to the MSB as well as the Carry Flag in the PSW. The other PSW bits, such as S, Z, P, or AC, are not affected by this operation.				
11) RRC	Rotate Accumulator Right	RRC	RRC => Implicit Addressing Mode ~ RRC	1 Byte(8bit)	Carry Flag
	The RAL instruction causes each binary bit in the accumulator register to be rotated by one position to its left through the carry flag as well. The MSB value is shifted to the Carry Flag, and the Carry Flag in the PSW is shifted to the LSB. The other PSW bits, such as S, Z, P, or AC, are not affected by this operation.				
12) RAR	Rotate Accumulator Right with Carry	RAR	RAR => Implicit Addressing Mode ~ RAR	1 Byte(8bit)	Carry Flag
	The RAR instruction causes each binary bit in the accumulator register to be rotated by one position to its right through the carry flag as well. The LSB value is shifted to the Carry Flag, and the Carry Flag in the PSW is shifted to the MSB.				
13) CMA	Complement Accumulator	CMA	CMA => Implicit Addressing Mode ~ CMA	1 Byte(16bit)	None
	The CMA instruction is used to compliment accumulator.				
14) CMC	Complement Accumulator with Carry	CMC	CMC => Implicit Addressing Mode ~ CMC	1 Byte(8bit)	Carry Flag
	The CMC instruction complements the Carry Flag bit.				
15) STC	Set Carry Flag	STC	STC => Implicit Addressing Mode ~ STC	1 Byte(16bit)	None
	The STC instruction sets the Carry Flag bit to 1				

Branch Instructions ~1

	Instruction Description	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags
1) JMP	**Unconditional** Jumps to the address.	JMP	JMP => 16-bit address – Immediate Addressing Mode~ JMP 2185H	3 Byte(24bit)	None
2) JC	**Jump on carry** Jumps to the address if carry flag is 1	JC	JC => 16-bit address – Immediate Addressing Mode~ JC 1425H	3 Byte(24bit)	None
3) JNC	**Jump on no carry** Jumps to the address if carry flag is 0	JNC	JNC => 16-bit address – Immediate Addressing Mode~ JNC 7120H	3 Byte(24bit)	None
4) JZ	**Jump on zero** Jumps to the address if zero flag is 1	JZ	JZ => 16-bit address – Immediate Addressing Mode~ JZ 2000H	3 Byte(24bit)	None
5) JNZ	**Jump on no zero** Jumps to the address if zero flag is 0	JNZ	JNZ => 16-bit address – Immediate Addressing Mode~ JNZ 2471H	3 Byte(24bit)	None
6) JP	**Jump on plus** Jumps to the address if sign flag is 0	JP	JP => => 16-bit address – Immediate Addressing Mode~ JP 3415H	3 Byte(24bit)	None
7) JM	**Jump on minus** Jumps to the address if sign flag is 1	JM	JM => 16-bit address – Immediate Addressing Mode~ JM 1792H	3 Byte(24bit)	None
8) JPE	**Jump on even parity** Jumps to the address if parity flag is 1	JPE	JPE=> 16-bit address – Immediate Addressing Mode~ JPE 4701H	3 Byte(24bit)	None
9) JPO	**Jump on odd parity** Jumps to the address if parity flag is 0	JPO	JPO => 16-bit address – Immediate Addressing Mode~ JPO 7013H	3 Byte(24bit)	None
10) CALL	**Unconditional call** Unconditionally call	CALL	CALL => 16-bit address – Immediate | Register Addressing Mode ~CALL 5410H	3 Byte(24bit)	None
11) CC	**Call on carry** Call if carry flag is 1	CC	CC => 16-bit address – Immediate | Register Addressing Mode ~CC 4008H	3 Byte(24bit)	None
12) CNC	**Call on no carry** Call if carry flag is 0	CNC	CNC => 16-bit address – Immediate | Register Addressing Mode ~CNC 8142H	3 Byte(24bit)	None
13) CZ	**Call on zero** Call if zero flag is 1	CZ	CZ => 16-bit address – immediate | Register Addressing Mode ~ CZ 1142H	3 Byte(24bit)	None
14) CNZ	**Call on no zero** Call if zero flag is 0	CNZ	CNZ => 16-bit address – immediate | Register Addressing Mode ~CNZ 4102H	3 Byte(24bit)	None
15) CP	**Call on plus** Call if sign flag is 0	CP	CP=> 16-bit address – immediate | Register Addressing Mode~ CP 2H	3 Byte(24bit)	None

Branch Instructions ~2

Instruction	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags
16) CM	Call on minus	CM CM => 16-bit address – Immediate | Register Addressing Mode ~ CM 1185H	3 Byte(24bit)	None
	Call if sign flag is 1			
17) CPE	Call on even parity	CPE CPE => 16-bit address – Immediate | Register Addressing Mode~ CPE 4125H	3 Byte(24bit)	None
	Call if parity flag is 1			
18) CPO	Call on odd parity	CPO CPO => 16-bit address – Immediate | Register Addressing Mode ~ CPO 2019H	3 Byte(24bit)	None
	Call if parity flag is 0			
19) RET	Unconditional return	RET RET => Indirect Addressing Mode~ RET	1 Byte(8bit)	None
	Return from the subroutine unconditionally.			
20) RC	Return on carry	RC RC => Indirect Addressing Mode~ RC	1 Byte(8bit)	None
	Return from the subroutine if carry flag is 1			
21) RNC	Return on no carry	RNC RNC => Indirect Addressing Mode~ RNC	1 Byte(8bit)	None
	Return from the subroutine if carry flag is 0			
22) RZ	Return on zero	RZ RZ => Indirect Addressing Mode ~ RZ	1 Byte(8bit)	None
	Return from the subroutine if zero flag is 1			
23) RNZ	Return on no zero	RNZ RNZ=> Indirect Addressing Mode ~ RNZ	1 Byte(8bit)	None
	Return from the subroutine if zero flag is 0			
24) RP	Return on plus	RP RP=> Indirect Addressing Mode ~ RP	1 Byte(8bit)	None
	Return from the subroutine if sign flag is 0			
25) RM	Return on minus	RM RM=> Indirect Addressing Mode ~ RM	1 Byte(8bit)	None
	Return from the subroutine if sign flag is 1			
26) RPE	Return on even parity	RPE RPE=> Indirect Addressing Mode ~ RPE	1 Byte(8bit)	None
	Return from the subroutine if parity flag is 1			
27) RPO	Return on odd parity	RPO RPO=> Indirect Addressing Mode ~ RPO	1 Byte(8bit)	None
	Return from the subroutine if parity flag is 0			

Input-Output Instructions

Instruction	Mnemonic	Opcode => Operand - Addressing Mode~ Example	Size	Flags	Description

1) PUSH

PUSH instruction PUSH PUSH =>R Register~ Indirect Addressing Mode – PUSH H|PUSH D 1 Byte(8bit) None

The PUSH command pushes the contents of the register onto the stack in 8085, which saves it as a temporary copy.

2) PUSH PSW

PUSH PSW instruction PUSH PSW PUSH => PSW - Indirect Addressing Mode ~ PUSH PSW 1 Byte(8bit) None

The PUSH PSW command pushes the contents of the PSW (Program status word or flag register) onto the stack in 8085, which saves it as a temporary copy.

3) POP

POP instruction 16-bit POP POP => R Register~ Indirect Addressing Mode – POP H|POP B 1 Byte(8bit) None

The POP command pops the contents stored onto the stack (where it might have been saved as a temporary copy) to a register.

4) POP PSW

POP PSW instruction POP PSW POP => PSW - Indirect Addressing Mode ~ POP PSW 1 Byte(8bit) None

The data at the memory location pointed by the stack pointer are copied to the flag register. The stack pointer is incremented by one, and the contents of that location are copied to the accumulator. The stack pointer is again incremented by one.

5) XTHL

Exchange Stack & HL register pair XTHL XTHL - Indirect Addressing Mode ~ XTHL 1 Byte(8bit) None

Contents of L register are exchanged with top of the stack. The stack pointer is incremented, and the contents of the next location of the stack are exchanged with H register.

6) SPHL

Move content HL to SP register SPHL SPHL - Register Addressing Mode ~ SPHL 1 Byte(8bit) None

Contents of L register are exchanged with top of the stack. The stack pointer is incremented, and the contents of the next location of the stack are exchanged with H register.

7) IN

Input contents at port shifted to the accumulator IN IN => 8-bit port address– Direct addressing Mode ~ IN 12 2 Byte(16bit) None

The current 8-bit contents available at the PORT# mentioned will be shifted to the Accumulator register.

8) OUT

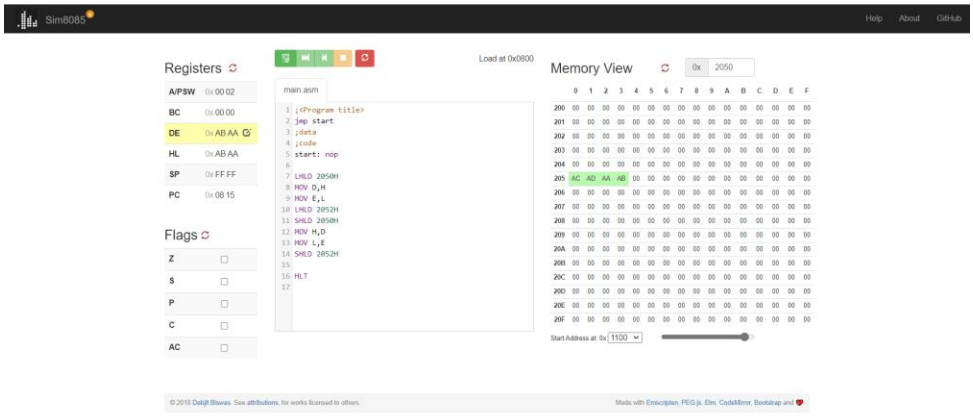
Contents of the accumulator shifted to port OUT OUT => 8-bit port address– Direct addressing Mode ~ OUT 27 2 Byte(16bit) None

The 8-bit value stored in the accumulator register gets copied on to the PORT#.

8085 Assembly Language Programs (Assignment -06)

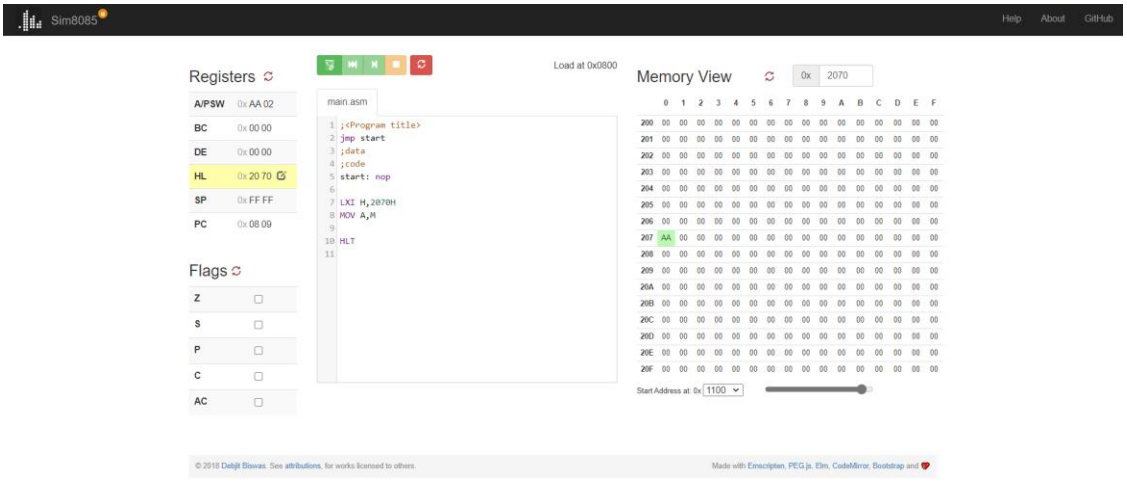
Design an 8085 Assembly Language Program for the following definitions:

A) Write an 8085 Assembly language program to interchange 16-bit data stored in memory locations 2050, 2051, 2052, and 2053 without XCHG instruction.

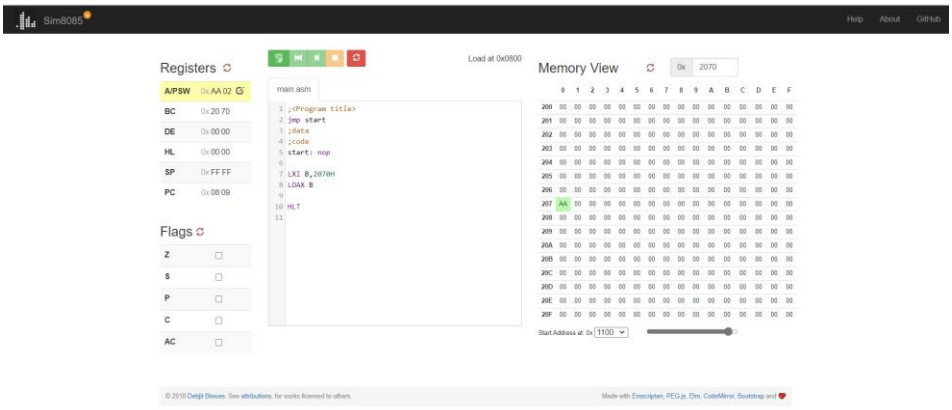


B) The memory location 2070H holds the data byte F2H.Write instructions to transfer the data byte to the accumulator using three different Opcode: MOV, LDAX, and LDA.

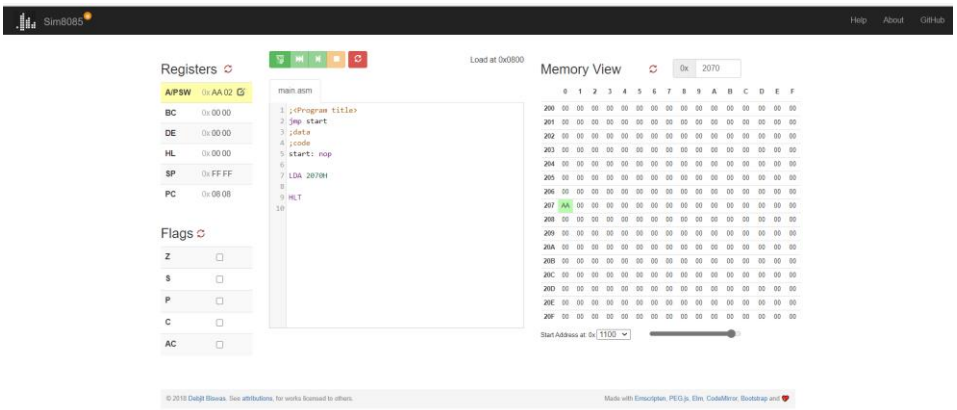
i. MOV:



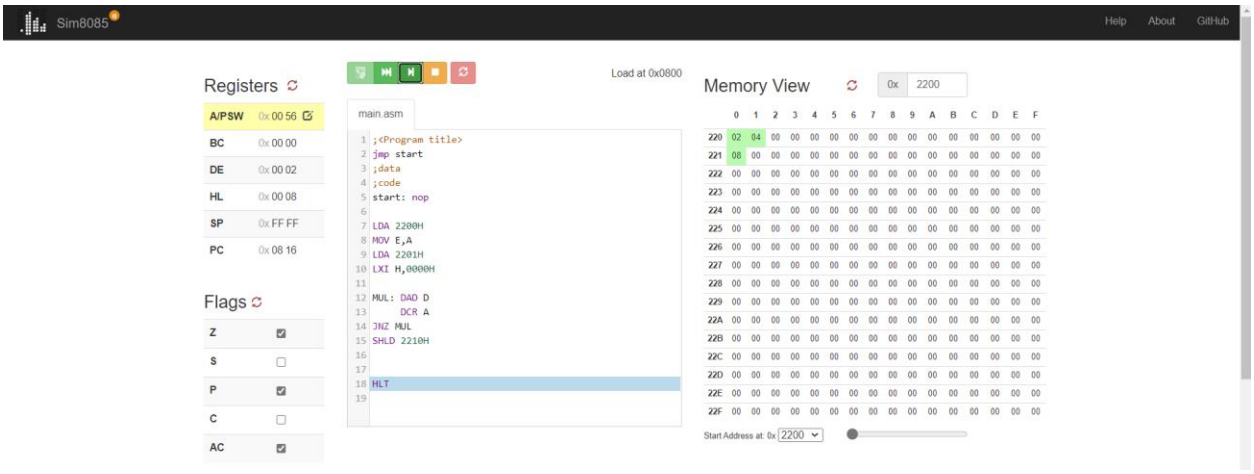
ii. LDAX:



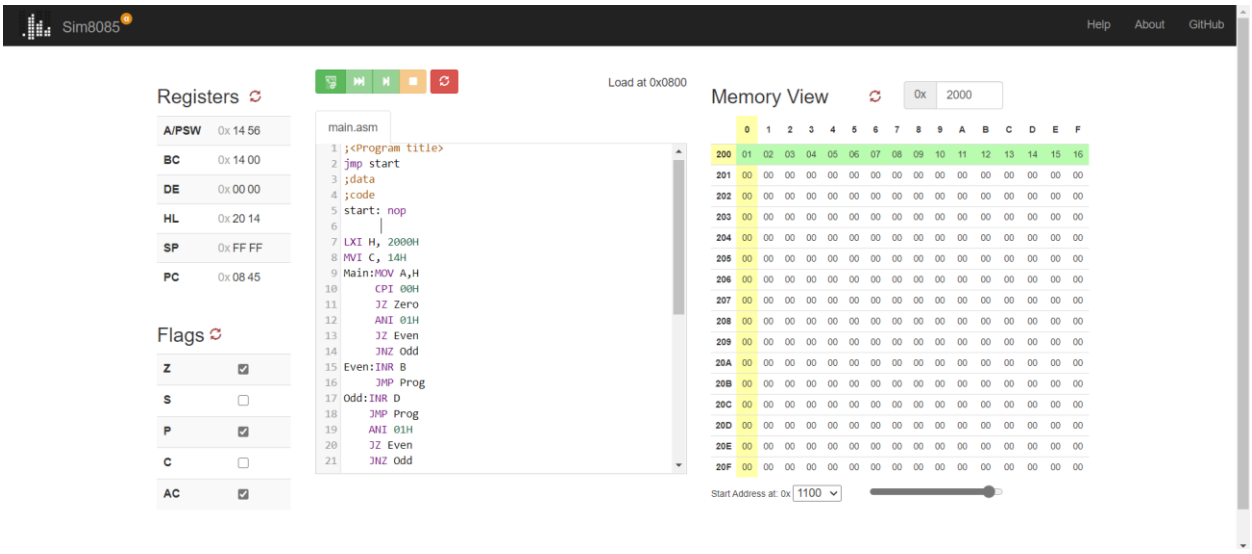
iii. LDA:

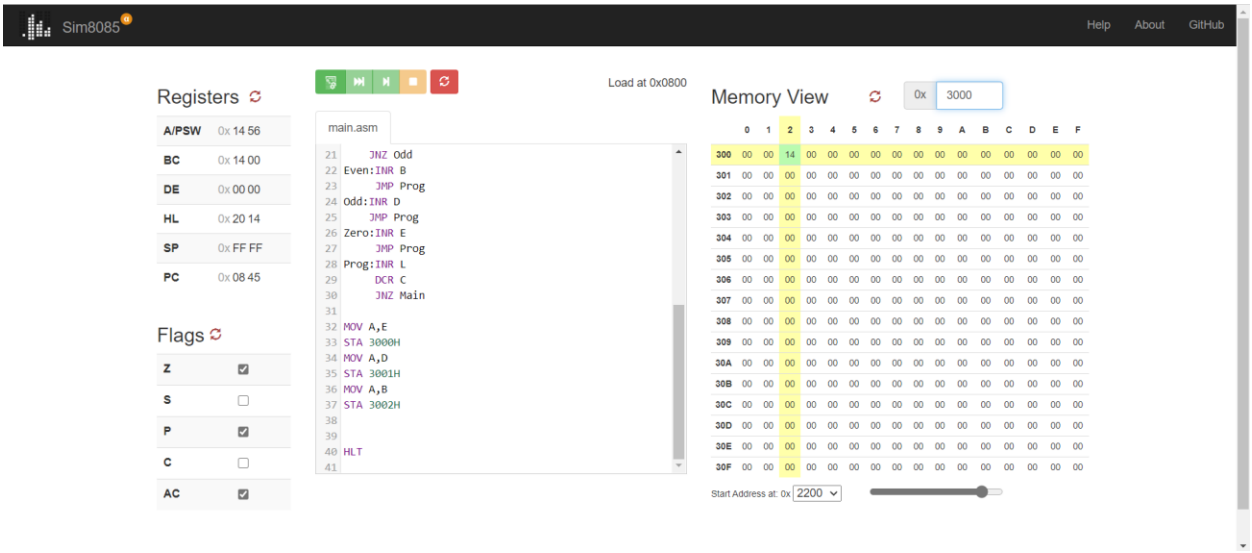


C) Write assembly language program to do multiplication of two numbers. Specify the memory location of each and every instruction and also draw flowchart.



D) An array of twenty data bytes is stored on memory locations 2000H onwards. Write an 8085-assembly language program to count the number of zeros, odd numbers and even numbers and store them on memory locations 3000H, 3001H and 3002H, respectively.





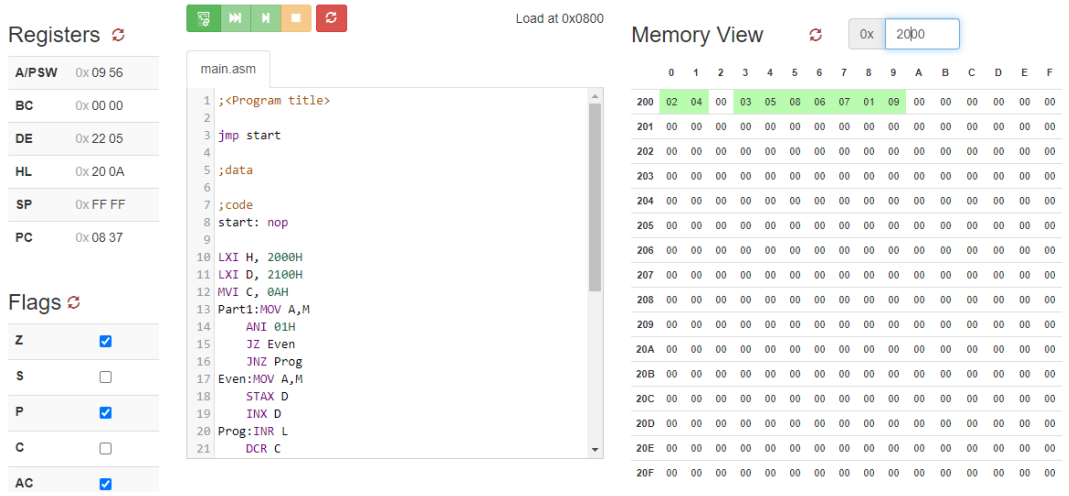
8085 Assembly Language Programs

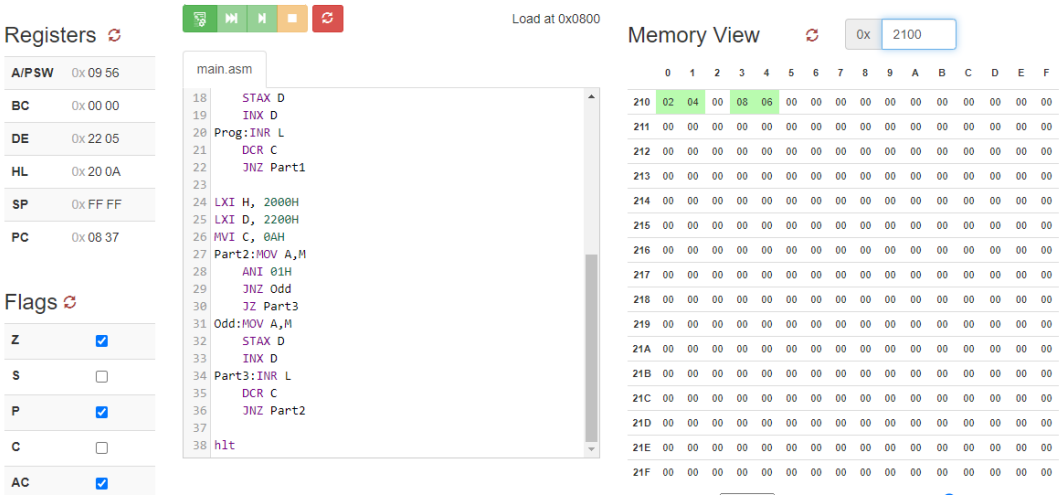
(Assignment -07)

- **Design an 8085 Assembly Language Program for the following definitions:**

A) An array of ten numbers is stored from memory location 2000H onwards. Write an 8085-assembly language program to separate out and store the EVEN and ODD numbers on new arrays from 2100H and 2200H, respectively.

Output:





Memory View 0x 2200

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
220	03	05	07	01	09	00	00	00	00	00	00	00	00	00	00	00
221	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
222	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
223	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
225	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
226	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
227	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
228	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
229	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
22F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

B) Write an 8085-assembly language program to count the number of bytes that are greater than 2010 and lesser than 4010 from an array of ten bytes stored on memory locations 2000H onwards. Store such numbers on memory locations 3000H onwards.

Output:

Registers

A/PSW	0x 00 56
BC	0x 00 00
DE	0x 30 06
HL	0x 20 09
SP	0x FF FF
PC	0x 08 6D

Flags

Z	☑
S	☐
P	☑
C	☐
AC	☑

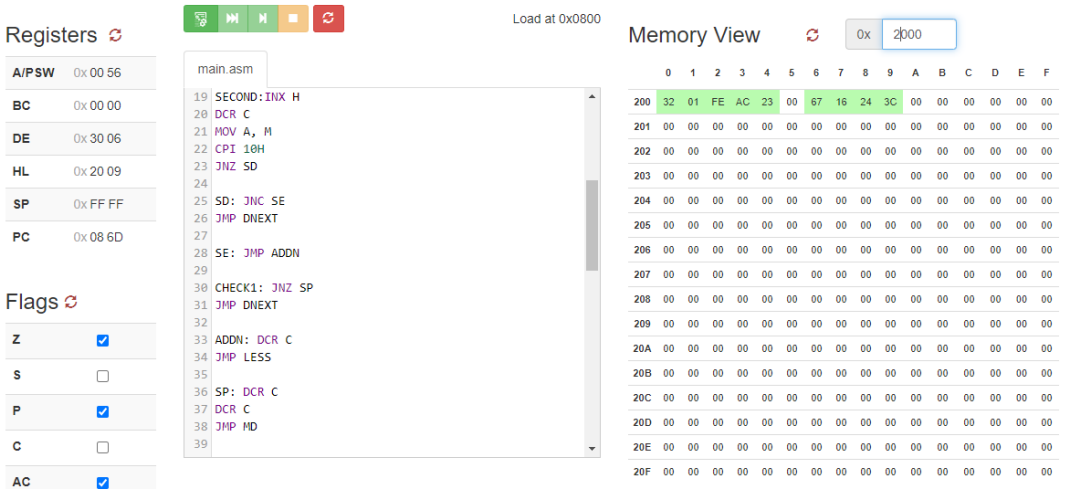
Load at 0x0800

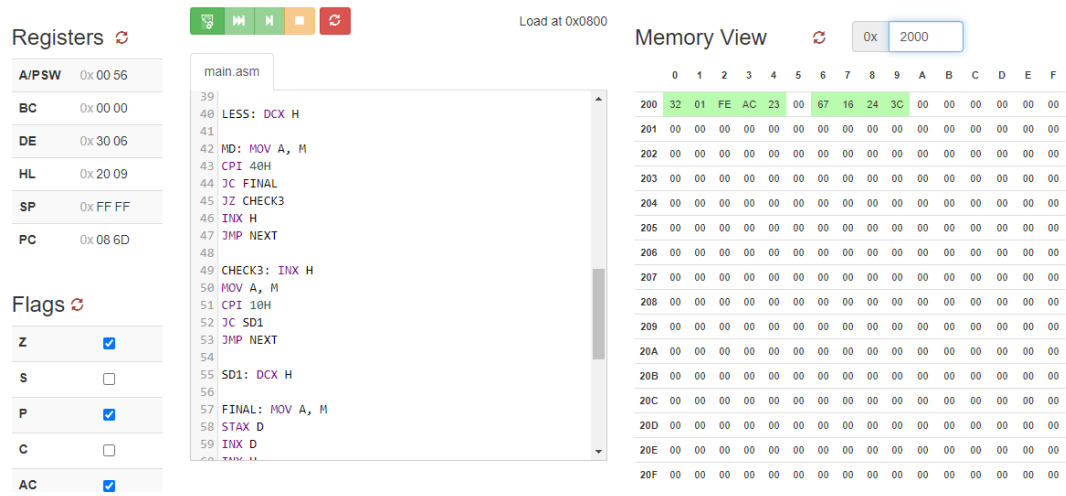
main.asm

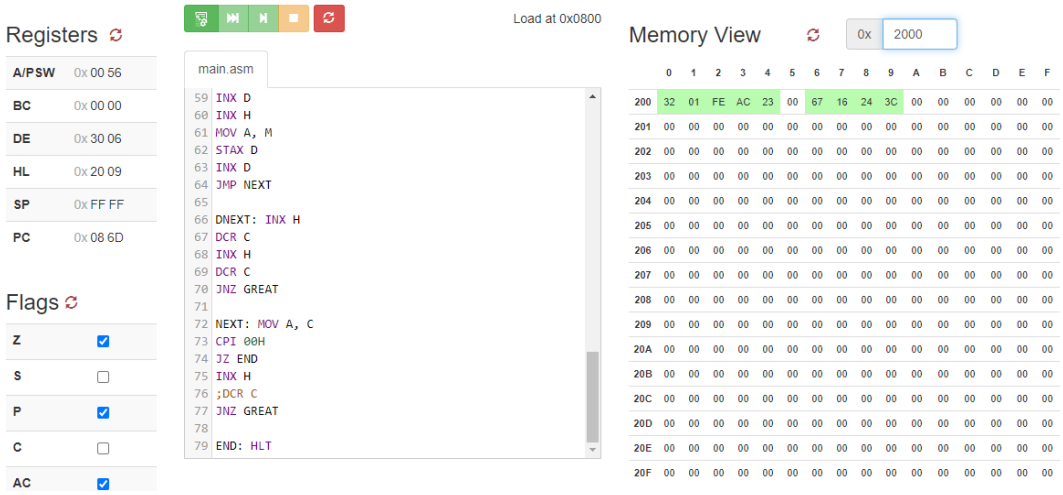
```
1  ;<Program title>
2
3  jmp start
4
5  ;data
6
7  ;code
8  start: nop
9  LXI H, 2000H
10 LXI D, 3000H
11 MVI C, 0AH
12
13 GREAT:MOV A, M
14 CPI 20H
15 JZ SECOND
16 JNC CHECK1
17 JMP DNEXT
18
19 SECOND:INX H
20 DCR C
21 MOV A, M
```

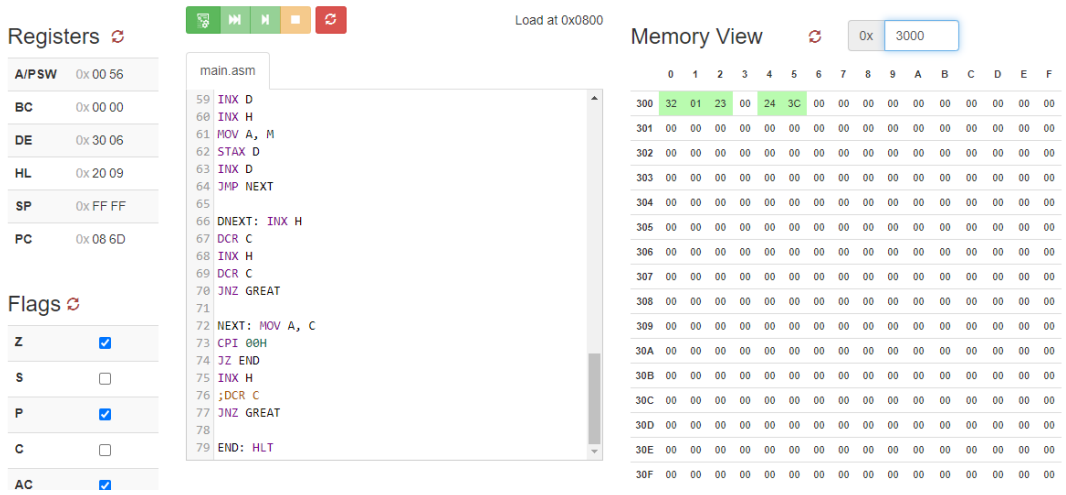
Memory View 0x 2000

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
200	32	01	FE	AC	23	00	67	16	24	3C	00	00	00	00	00	00
201	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
202	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
203	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
204	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
205	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
206	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
207	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
209	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00


Registers
A/PSW 0x 00 56
BC 0x 00 00
DE 0x 30 06
HL 0x 20 09
SP 0x FF FF
PC 0x 08 6D
Flags
Z
S
P
C
AC
Load at 0x0800
main.asm
19 SECOND:INX H
20 DCR C
21 MOV A, M
22 CPI 10H
23 JNZ SD
24
25 SD: JNC SE
26 JMP DNEXT
27
28 SE: JMP ADDN
29
30 CHECK1: JNZ SP
31 JMP DNEXT
32
33 ADDN: DCR C
34 JMP LESS
35
36 SP: DCR C
37 DCR C
38 JMP MD
39
Memory View
0x 2000
0 1 2 3 4 5 6 7 8 9 A B C D E F
200 32 01 FE AC 23 00 67 16 24 3C 00 00 00 00 00 00
201 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
202 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
203 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
204 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
205 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
206 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
207 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
208 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
209 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

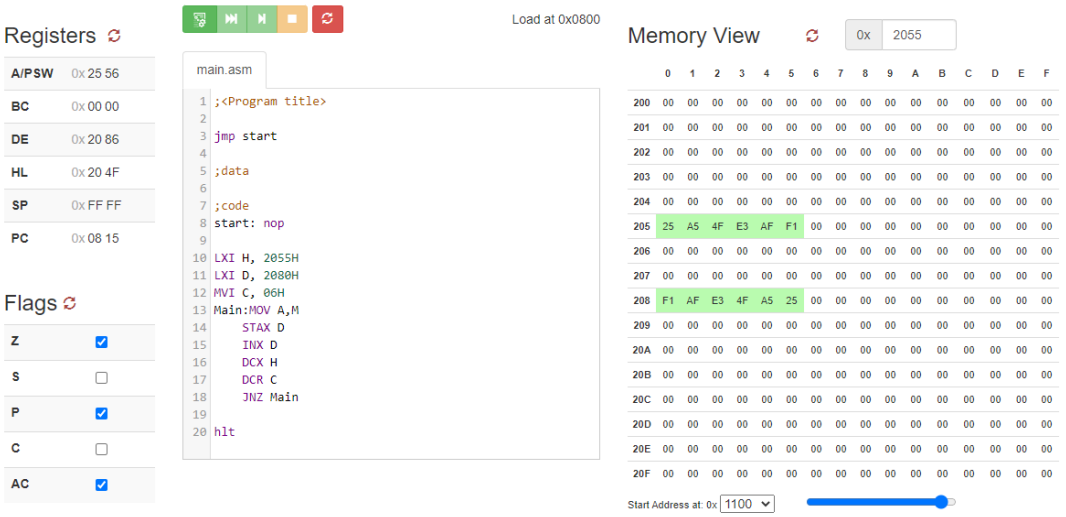

Registers
A/PSW 0x 00 56
BC 0x 00 00
DE 0x 30 06
HL 0x 20 09
SP 0x FF FF
PC 0x 08 6D
Flags
Z
S
P
C
AC
Load at 0x0800
main.asm
39
40 LESS: DCX H
41
42 MD: MOV A, M
43 CPI 40H
44 JC FINAL
45 JZ CHECK3
46 INX H
47 JMP NEXT
48
49 CHECK3: INX H
50 MOV A, M
51 CPI 10H
52 JC SD1
53 JMP NEXT
54
55 SD1: DCX H
56
57 FINAL: MOV A, M
58 STAX D
59 INX D
Memory View
0x 2000
0 1 2 3 4 5 6 7 8 9 A B C D E F
200 32 01 FE AC 23 00 67 16 24 3C 00 00 00 00 00 00
201 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
202 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
203 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
204 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
205 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
206 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
207 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
208 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
209 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


Registers
A/PSW 0x 00 56
BC 0x 00 00
DE 0x 30 06
HL 0x 20 09
SP 0x FF FF
PC 0x 08 6D
Flags
Z
S
P
C
AC
Load at 0x0800
main.asm
59 INX D
60 INX H
61 MOV A, M
62 STAX D
63 INX D
64 JMP NEXT
65
66 DNEXT: INX H
67 DCR C
68 INX H
69 DCR C
70 JNZ GREAT
71
72 NEXT: MOV A, C
73 CPI 00H
74 JZ END
75 INX H
76 ;DCR C
77 JNZ GREAT
78
79 END: HLT
Memory View
0x 2000
0 1 2 3 4 5 6 7 8 9 A B C D E F
200 32 01 FE AC 23 00 67 16 24 3C 00 00 00 00 00 00
201 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
202 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
203 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
204 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
205 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
206 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
207 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
208 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
209 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



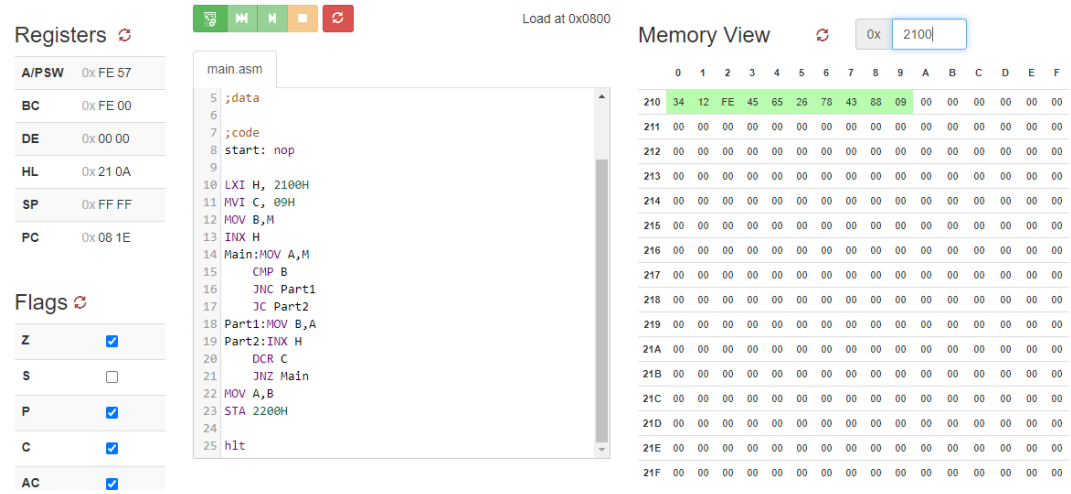
C) The following block of data is stored in the memory locations from 2050H to 2055H. Write an 8085 Assembly language program to Transfer the data to the locations 2080H to 2085H in the reverse order. DATA(H) 25, A5, 4F, E3, AF, F1

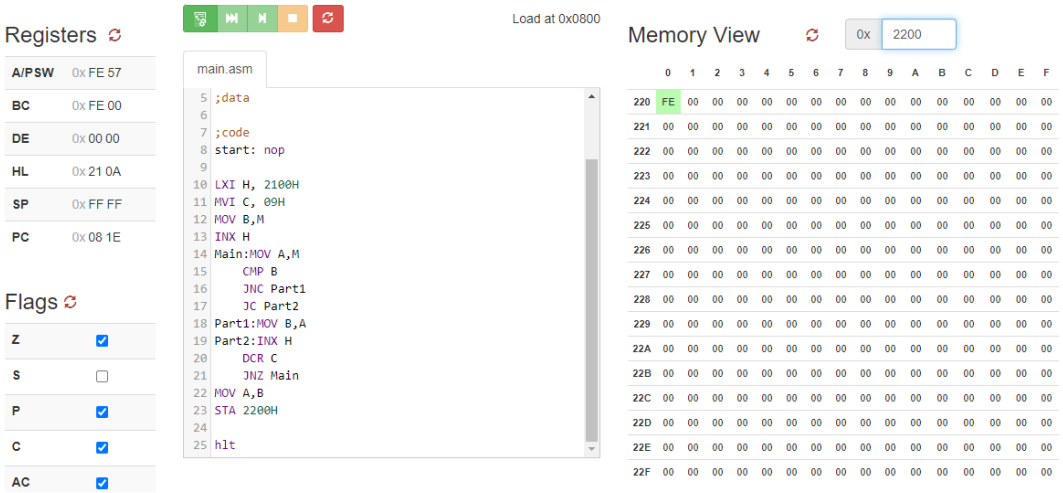
Output:



D) An array of ten data bytes is stored on memory locations 2100H onwards. Write an 8085-assembly language program to find the largest number and store it on memory location 2200H.

Output:





Microprocessor and Interfacing

→ 18021010760

→ ARJUN Vankani

DATE: / / PAGE

(Assignment - 8)

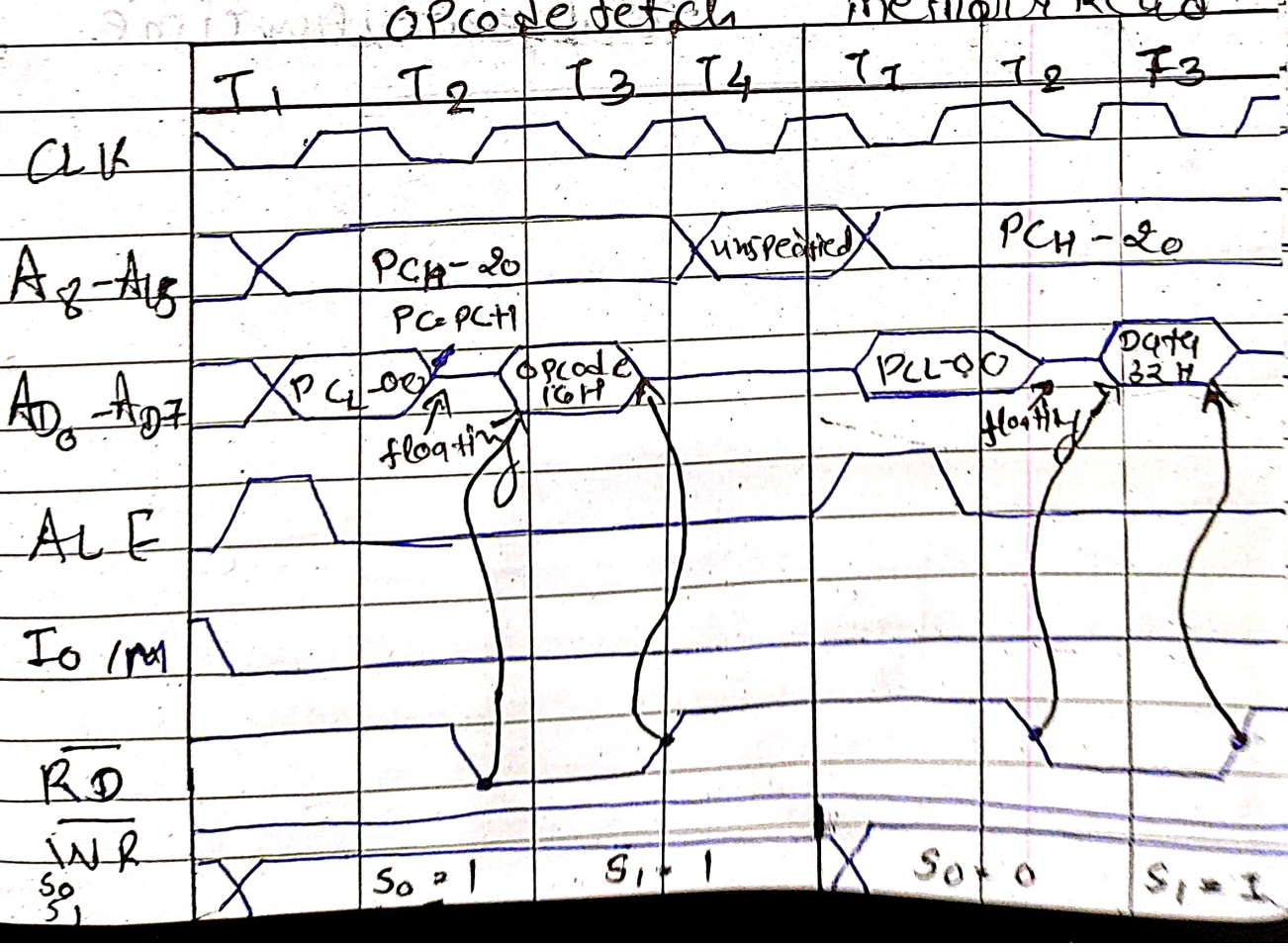
Timing Diagram

Q-1 Draw the timing diagram for the following given instruction.

I) MVI A, 32H

→ Number of required machine cycles and T-states:

- Opcode fetch : 4 T-states
- memory Read : 3 T-states



DATE : / / PAGE

→ Clock frequency of 8085 $f = 3\,MHz$

→ T-State Clock Period $= 1/f = 0.333\,\mu s$

⇒ Total execution time ⇒

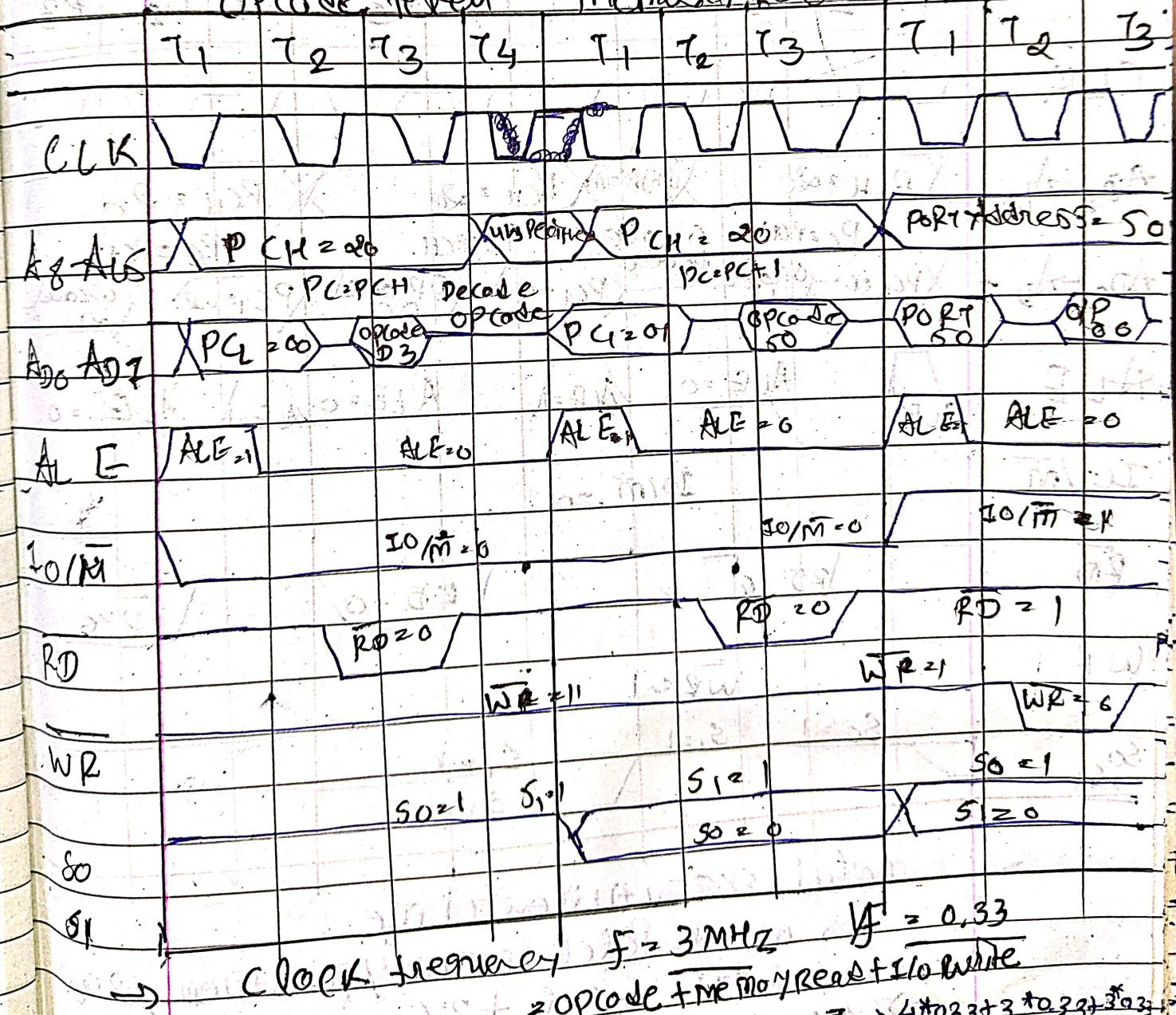
Machine cycle	No. of T-state	Clock Period
OPcode fetch	4T	$4 * 0.33 = 1.32$
Memory Read	3T	$3 * 0.33 = 0.99$
~~Memory Read~~	3T	$3 * 0.33 = 0.99$
	Total execution time	3.33 μs

DATE: / / PAGE

(Q2)

OUT 50H ⇒ Number of required Machine cycles and T-states.

- Opcode fetch : 4 T-states
- memory Read : 3-states
- I/O write : 3-states



→ Clock frequency $f = 3\,MHz$ $\Rightarrow T = 0.33$

Total execution = Opcode + memory Read + I/O write

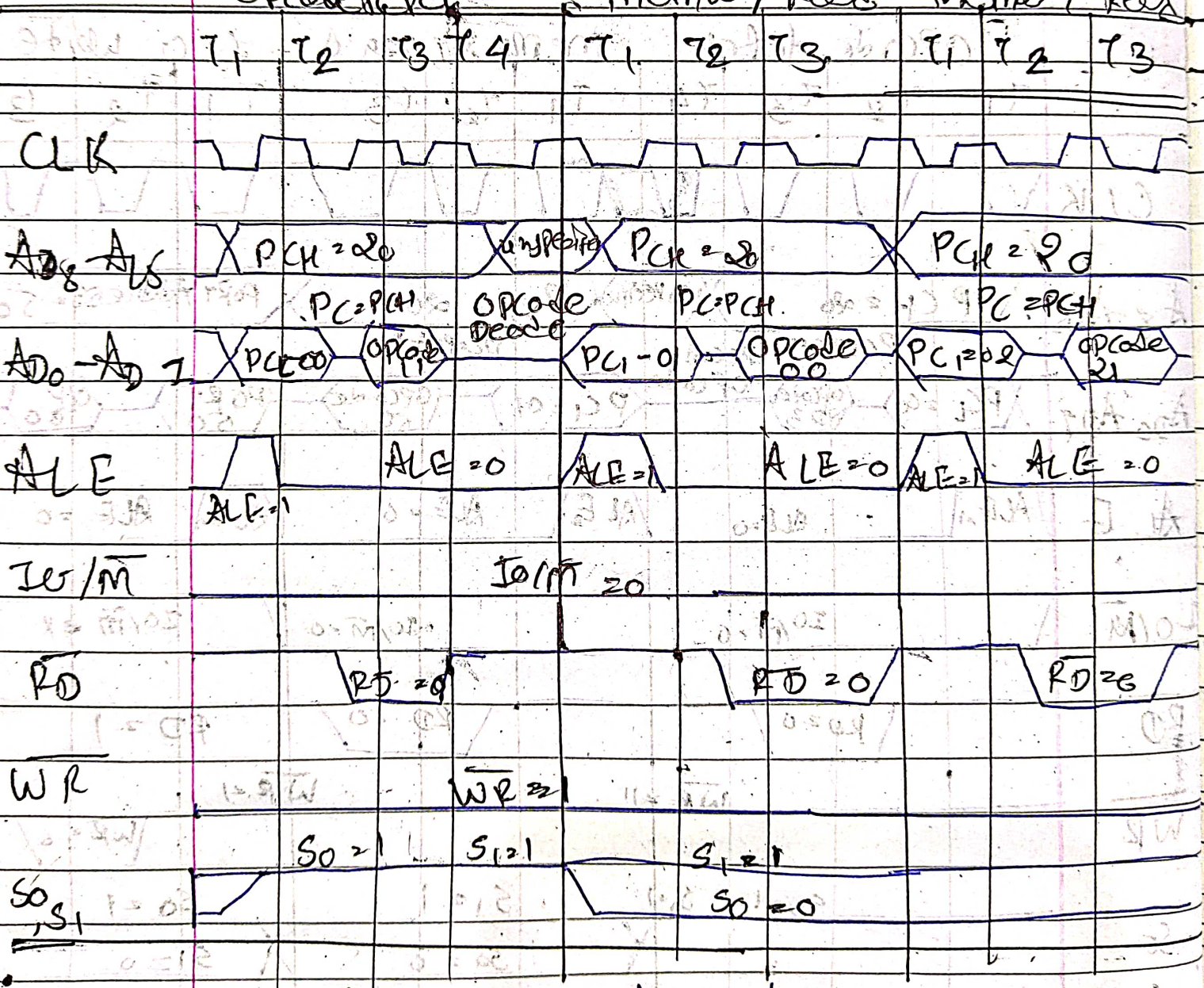
$= 4T + 3T + 3T \rightarrow 4*0.33 + 3*0.33 + 3*0.33$

$\Rightarrow$ **3.33 μs**

DATE : / / PAGE

(3) LXI B, 2100H, ⇒ Number of required Machine cycles and T-states

- Opcode fetch : 4 T-states
- memory Read : 3 T-states
- memory Reas : 3 T-states



⇒ Total execution time

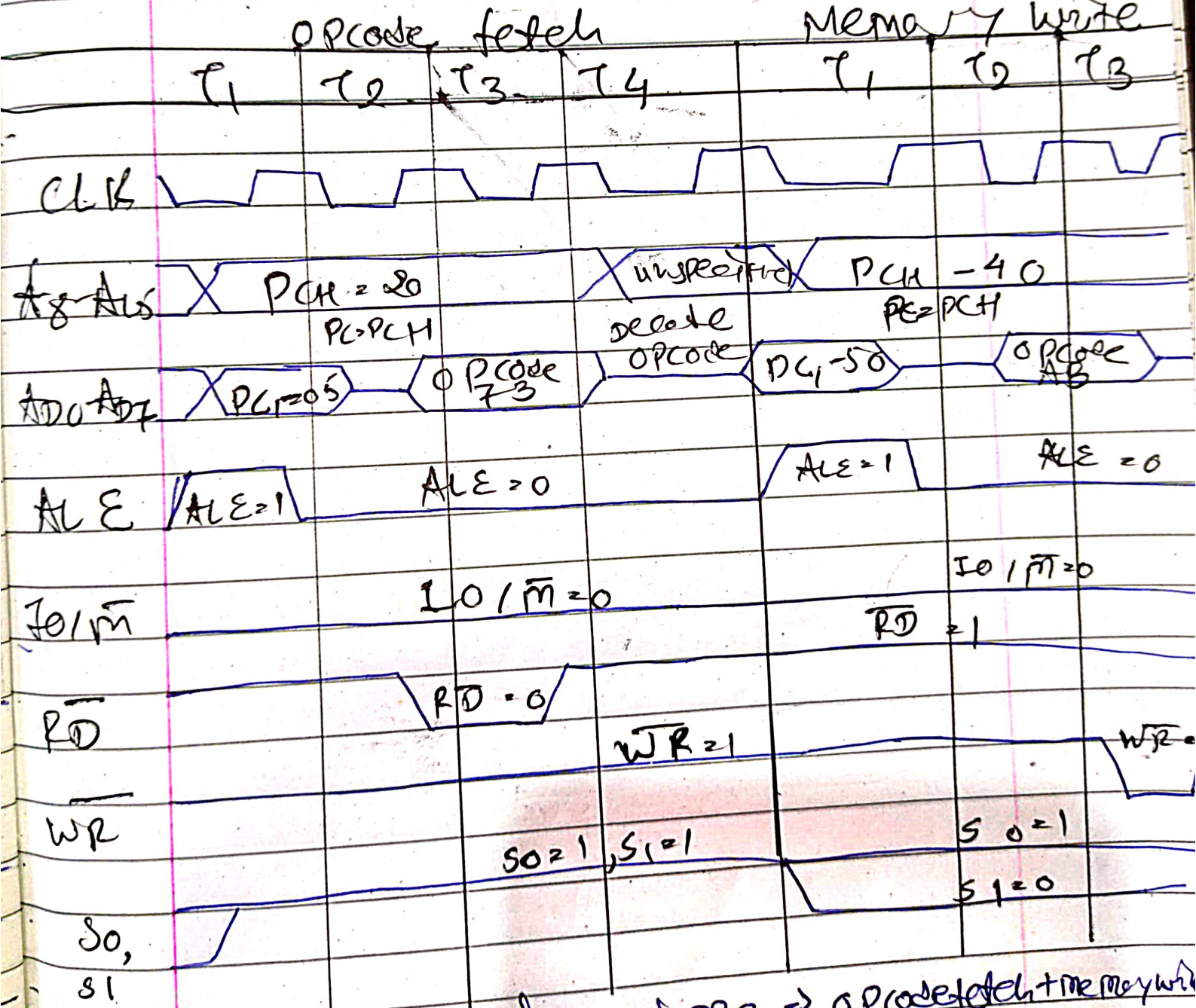
⇒ opcode fetch + memory Read + Memory Reas

⇒ 4T + 3T + 3T

⇒ 4 * 0.33 + 3 * 0.33 + 3 * 0.3 ⇒ 3.33 μs

4/ MOV M, B → Number of required Machine cycles and T-states

→ → • Opcode fetch : 4 T-States
• Memory Write : 3 -States



⇒ Total Execution Time ⇒ Opcode fetch + Memory write
→ 4T + 3T
⇒ 4*0.33 + 3*0.33
⇒ 2.31 μs