Vishwakarma Government Engineering College, Chandkheda

Computer Engineering Department

Semester-VI

3160712-Microprocessor and Interfacing

Practical List

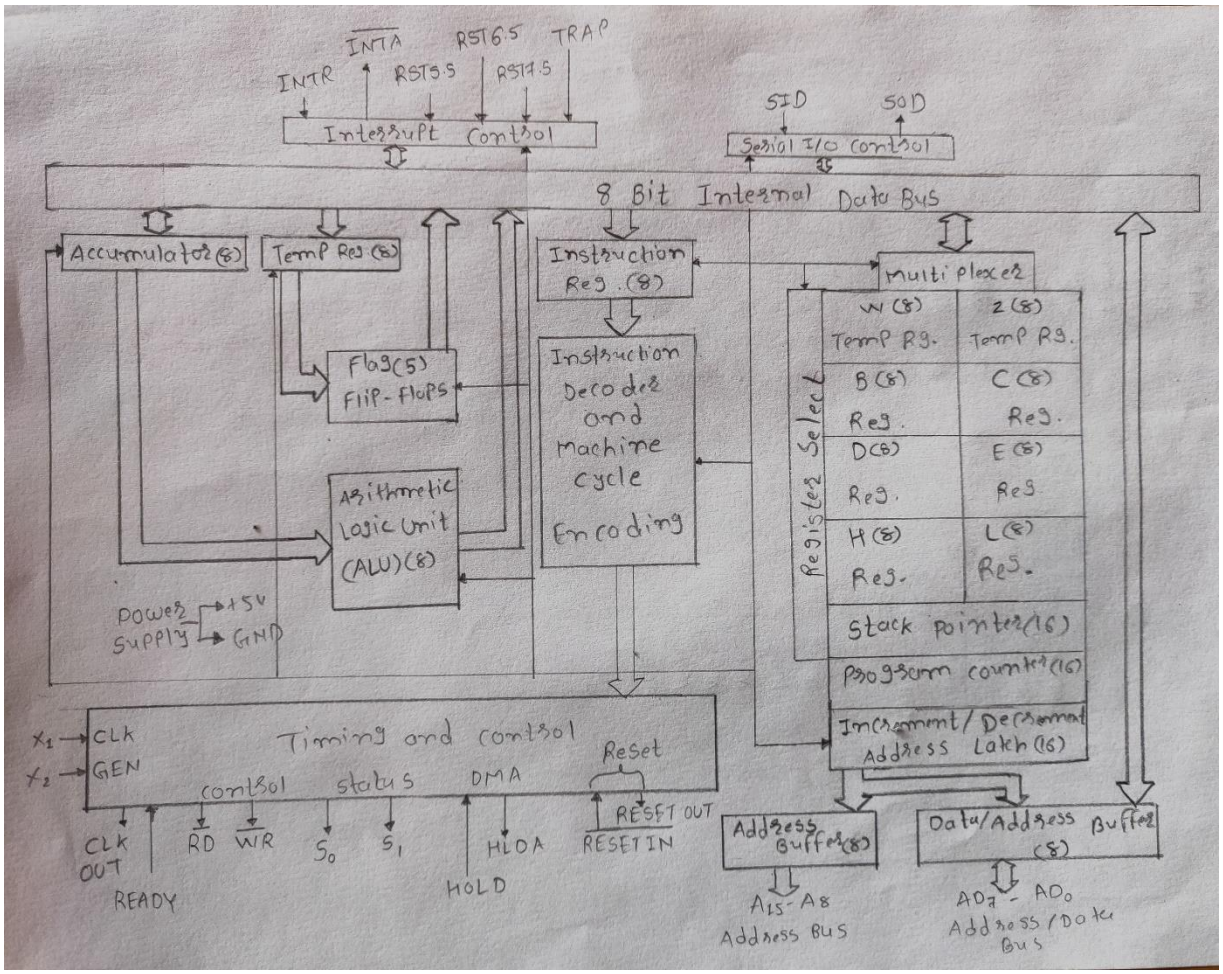
Practical No.	AIM	CO
1	Study of 8085 microprocessor architecture, Pin diagram and bus organization.	1
2	Study of 8085 Instruction set. a) Data transfer group b) Arithmetic group c) Logic group d) Branch group e) Stack, I/O and machine control group	2
3	a. Write a program for addition of two 16-bit numbers using carry using 8085. b. Write a program for subtraction of two 8-Bit Numbers. (Display of Borrow) using 8085.	3
4	a. Write a program for Multiplication of Two 8-Bit Numbers By Bit Rotation Method using 8085. b. Division of Two 8-Bit Numbers by Repeated Subtraction Method using 8085.	3
5	a. Write a Program using 8085 for Finding Square-Root of a Number. b. Write a program to find a factorial of a given number using 8085.	3
6	a. Write a program to move a block of data using 8085. b. Write a Program to Arrange Number in Ascending Order Using 8085.	3
7	a. Write a Program to Find GCD Of Two Numbers Using 8085. b. Write a Program to Add ‘N’ Two Digit BCD Numbers Using 8085.	3
8	a. Write a program for 8085to reverse the block of 8-bit values using stack. The length of the block is given in memory location 2050H and block starts from 2051H. b. Write a Program to compare two 16-bit numbers stored in memory Using 8085.	3
9	Study of 8255A Programmable Peripheral Interface Architecture, ports and their modes.	4
10	Study of 8086 microprocessor architecture and its register organization.	5

Practical – 1

Aim: Study of 8085 microprocessor basic architecture, pin diagram and bus organization.

- 8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology. It is used in washing machines, microwave ovens, mobile phones, etc.
- The maximum clock frequency is 3MHz while minimum frequency is 500 kHz.
- It provides 16 address lines, therefore capable of addressing 2^{16}= 64K of memory.
- It works on 5-Volt DC power supply.
- It is a single chip N MOS device with 40 pins.
- It provides accumulator, 5 flag register, 6 general purpose registers and 2 special purpose registers (SP, PC).
- It has two 16 bit registers named program counters (PC) and stack pointer (SP).
- It generates 8 bit I/O address so it can access 2^8 = 256 input ports.

8085 microprocessor architecture:



8085 consists of the following functional units −

1. **Accumulator**

   It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU. Result of operation is stored in Accumulator.

2. **Arithmetic and logic unit**

   As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

3. **General purpose register**

   There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data. These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

4. **Program counter**

   It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

5. **Stack pointer**

   It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

6. **Temporary register**

   It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

7. **Flag register**

   It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

   - These are the set of 5 flip-flops −

     - Sign (S): Set (1) if 7th bit of result is 1; otherwise reset (0).
     - Zero (Z): Set (1) when result is zero; otherwise reset (0).
     - Auxiliary Carry (AC): Set (1) when carry bit is generated by 3rd bit & passed to. bit 4th bit.

- Parity (P): Set (1) if result has even no. of 1's & Reset (0) if result has odd no. of 1's.
- Carry (CY): Set (1) if arithmetic operation results in carry; otherwise reset (0).

8. Instruction register and decoder

It is an 8-bit register. When an instruction is fetched from memory then it is stored in the Instruction register. Instruction decoder decodes the information present in the Instruction register.

9. Timing and control unit

It provides timing and control signal to the microprocessor to perform operations. Following are the timing and control signals, which control external and internal circuits

- Control Signals: READY, RD', WR', ALE
- Status Signals: S0, S1, IO/M'
- DMA Signals: HOLD, HLDA
- RESET Signals: RESET IN, RESET OUT

10. Interrupt control

As the name suggests it controls the interrupts during a process. When a microprocessor is executing a main program and whenever an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

There are 5 interrupt signals in 8085 microprocessor: INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

11. Serial Input/output control

It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

12. Address buffer and address-data buffer

The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

13. Address bus

Whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices. Group of 16 unidirectional lines generally identified as A0 to A15. I.e. bits flow from microprocessor

to peripheral devices. 16 address lines are capable of addressing 65536 memory locations. So, 8085 has 64K memory locations.

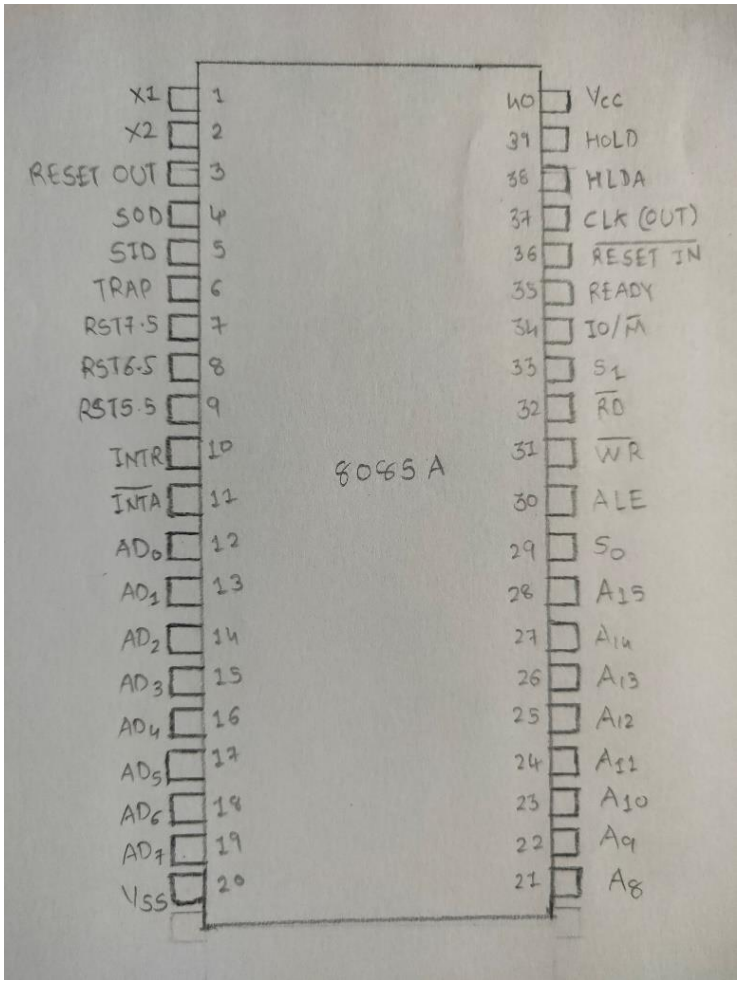
14. data bus

Data bus carries the data to be stored. It is bidirectional. Group of 8 lines identified as D0 to D7. 8 data lines enable microprocessor to manipulate data ranging from 00H to FFH (2^8=256 numbers). Largest number appear on data bus is 1111 1111 => $(255)_{10}$. As Data bus is of 8-bit, 8085 is known as 8-bit Microprocessor.

15. multiplexer

A multiplexer pulls out the right group of bits, depending on the instruction.

8085 Pin Diagram:



The pins of an 8085 microprocessor can be classified into seven groups −

- **Address bus**

  A15-A8 are unidirectional and used to carry high-order address of 16-bit address, it carries the most significant 8-bits of memory/IO address (Pin 28 – 21).

- **Data bus**

  AD7-AD0, it carries the least significant 8-bit (low-order address) address and data bus. The low-order address bus can be separate from these signals by using a latch (ALE) (Pin 19 - 12).

- **Control and status signals**

  These signals are used to identify the nature of operation. There are 3 control signal and 3 status signals.

  Three control signals are RD, WR & ALE.

  1. $\overline{\textbf{RD}}$ − This signal indicates that the selected IO or memory device is to be read and is ready for accepting data available on the data bus (Pin – 32).
  2. $\overline{\textbf{WR}}$ − This signal indicates that the data on the data bus is to be written into a selected memory or IO location (Pin – 31).
  3. **ALE** – (Address Latch Enable) it is a positive going pulse generated when a new operation is started by the microprocessor. When the pulse goes high, it indicates address (ALE ← 1, then Address bus). When the pulse goes down it indicates data (ALE ← 0, then Data bus) (Pin – 30).

  Three status signals are IO/M, S0 & S1.

  - **IO/ $\overline{\textbf{M}}$**

    This signal is used to differentiate between IO and Memory operations, i.e. when it is **high** indicates **IO operation** and when it is **low** then it indicates **memory operation** (Pin 34).

  - **S1 & S0**

    These signals are used to identify the type of current operation (Pin 30 & 29).

S_1	S_0	Mode
0	0	HLT
0	1	WRITE

1	0	READ
1	1	OPCODE FETCH

- **Clock signals**

  There are 3 clock signals, i.e. X1, X2, CLK OUT.

  - **X1, X2** − A crystal (RC, LC N/W) is connected at these two pins and is used to set frequency of the internal clock generator. This frequency is internally divided by 2. Therefore, to operate a system at 3MHz, the crystal should have a frequency of 6MHz (Pin 1 & 2).
  - **CLK (OUT)** − this signal is used as the system clock for devices connected with the microprocessor (Pin 37).

- **Interrupts & externally initiated signals**

  Interrupts are the signals generated by external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. We will discuss interrupts in detail in interrupts section.

  - $\overline{\textbf{INTA}}$ **(Output)** − It is an interrupt acknowledgment signal.
  - **INTR (Input) -** Interrupt Request It is used for general purpose interrupt.
  - $\overline{\textbf{RESET IN}}$ − This signal is used to reset the microprocessor by setting the program counter to zero.
  - **RESET OUT** − this signal is used to reset all the connected devices when the microprocessor is reset.
  - **RST7.5, RST6.5, RST5.5 (Input)** → Restart Interrupts. These are vector interrupts that transfer the program control to specific memory locations. RST7.5, RST6.5, RST5.5 have higher priorities than INTR interrupt. Among these 3 interrupts, the priority order (higher to lower) is RST7.5, RST6.5, and RST5.5 respectively.
  - **TRAP (Input) -** This is a non-mask able interrupt & has the highest priority.
  - **READY** − this signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.
  - **HOLD** − This signal indicates that another master is requesting the use of the address and data buses.
  - **HLDA (HOLD Acknowledge)** − It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed.

- **Serial I/O signals**

  There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.

  - **SOD** (Serial output data line) − The output SOD is set/reset as specified by the SIM instruction.
  - **SID** (Serial input data line) − the data on this line is loaded into accumulator whenever a RIM instruction is executed.

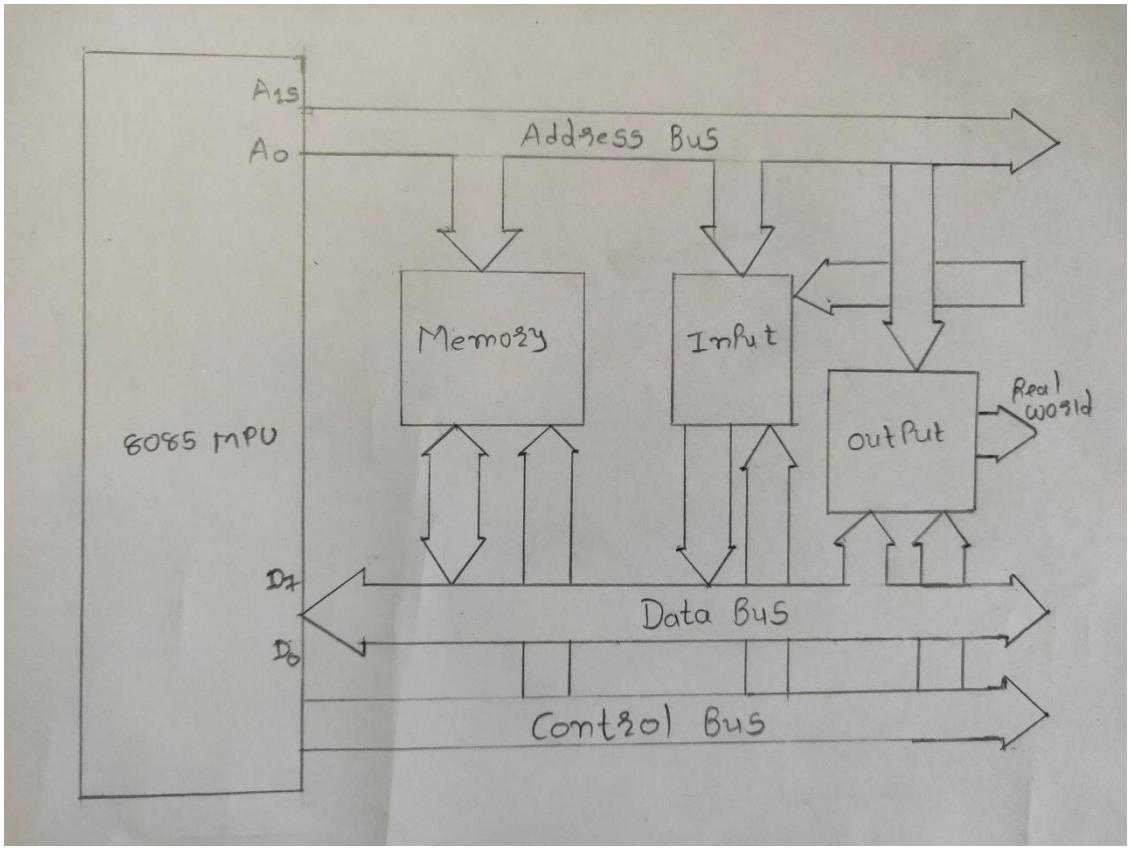
- **Power Supply & Frequency Signal**

  - V_{cc} → Pin 40, +5V Supply.
  - V_{ss} → Pin 20, Ground Reference

Control & Status :

IO/ $\overline{M}$	**$\overline{RD}$**	**$\overline{WR}$**	**Operation**
0	0	0	HLT
0	0	1	$\overline{MEMR}$
0	1	0	$\overline{MEMW}$
0	1	1	Opcode fetch
1	0	0	HLT
1	0	1	$\overline{IOR}$
1	1	0	$\overline{IOW}$
1	1	1	NOP

Bus Organization of 8085



1. Address bus

Whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices. Group of 16 unidirectional lines generally identified as A0 to A15. I.e. bits flow from microprocessor to peripheral devices. 16 address lines are capable of addressing 65536 memory locations. So, 8085 has 64K memory locations.

2. data bus

Data bus carries the data to be stored. It is bidirectional. Group of 8 lines identified as D0 to D7. 8 data lines enable microprocessor to manipulate data ranging from 00H to FFH (2^8=256 numbers). Largest number appear on data bus is 1111 1111 => $(255)_{10}$. As Data bus is of 8-bit, 8085 is known as 8-bit Microprocessor.

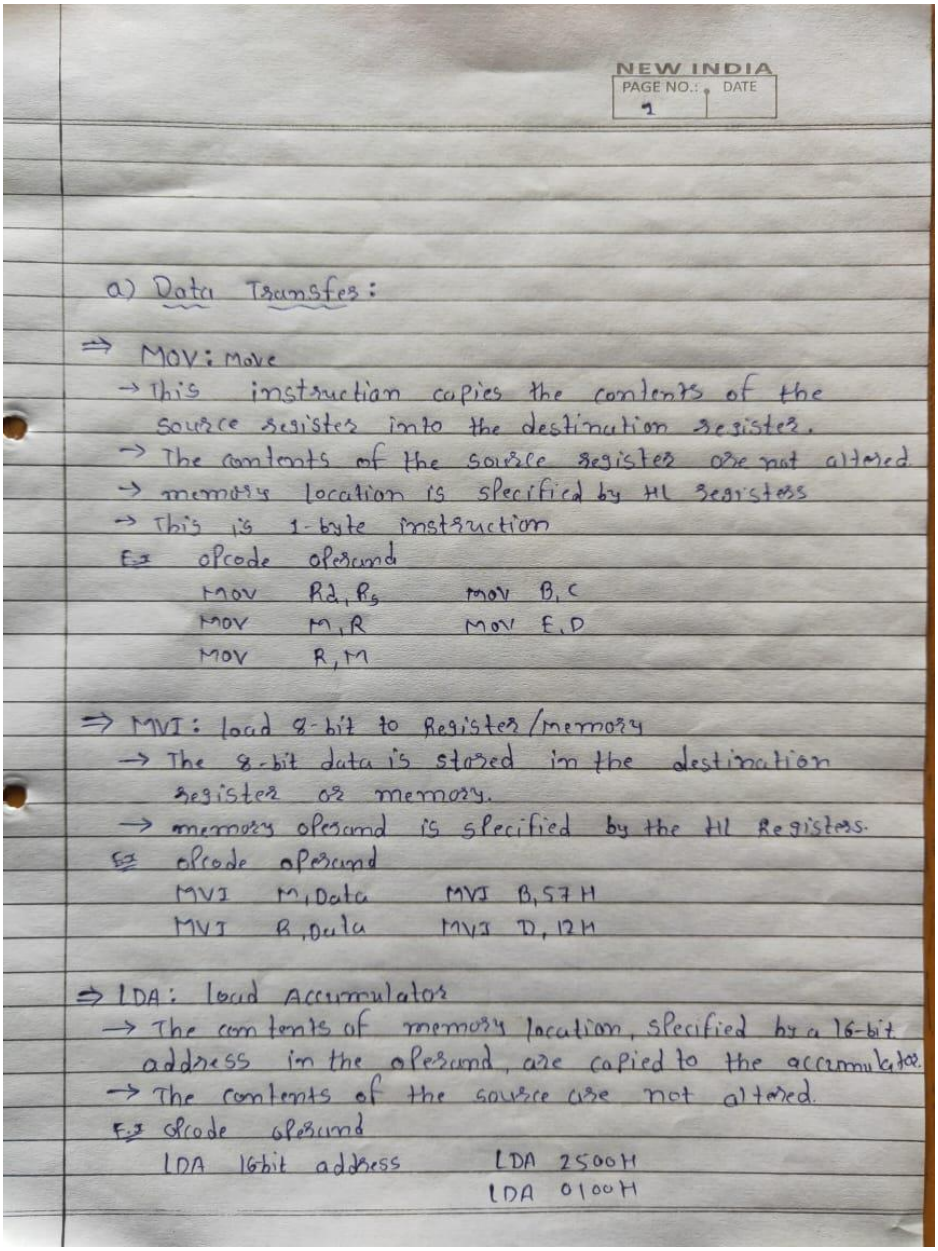
3. Control bus

It comprises of various single lines that carry synchronization, timing & control signals. These signals are used to identify a device type with which MPU intends to communicate. Some control signals are Read, Write and Opcode fetch etc.

Practical – 2

Aim: Study of 8085 instruction set.

a) Data transfer group
b) Arithmetic group
c) Logic group
d) Branch group
e) Stack, I/O and machine control group



NEW INDIA
PAGE NO.: 1 DATE

a) Data Transfer:

⇒ MOV: Move
→ This instruction copies the contents of the source register into the destination register.
→ The contents of the source register are not altered
→ memory location is specified by HL registers
→ This is 1-byte instruction
Ex opcode operand
MOV R2, R5 MOV B, C
MOV M, R MOV E, D
MOV R, M

⇒ MVI: load 8-bit to Register/memory
→ The 8-bit data is stored in the destination register or memory.
→ memory operand is specified by the HL Registers.
Ex opcode operand
MVI M, Data MVI B, 57 H
MVI R, Data MVI D, 12H

⇒ LDA: load Accumulator
→ The contents of memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
→ The contents of the source are not altered.
Ex opcode operand
LDA 16bit address LDA 2500H
LDA 0100H

NEW INDIA
PAGE NO.: 2 DATE

⇒ LDAX: load the accumulator indirect
→ The memory location pointed by extended register pair denoted as 'Rp'.
→ The contents of either the register pair or the memory location are not altered.
→ This instruction copies the contents of that memory location into the accumulator.
Ex opcode operand
LDAX Rp B/D

MVI B, 00H
MVI C, 06H
LDAX B

⇒ LXI: load the register pair immediate
→ The instruction loads 16-bit data in the register pair designated in the operand.
Ex opcode operand
LXI Rp 16-bit Data LXI H, 2034H
→ This is 3-byte instruction.

⇒ STA: store Accumulator
→ The contents of the accumulator are copied into the memory location specified by the operand.
Ex opcode operand
STA 16-bit address

MVI A, 09H
STA 0002H
→ This is 3-byte instruction

⇒ STAX: The contents of the accumulator are copied into the memory location specified by the contents of the register pair.
Ex
STAX Rp

STAX B

NEW INDIA
PAGE NO.: 3 DATE

⇒ LHLD : load H and L registers direct
→ The instruction copies the contents of the 16-bit memory location into the HL register pair.
Ex opcode operand LHLD 2050H
LHLD 16-bit address

⇒ SHLD: store H and L registers direct
→ The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand.
Ex opcode operand
SHLD 16-bit SHLD 0002H

⇒ XCHG: Exchange H and L with D and E
→ The contents of register H are exchanged with contents of register D, and the contents of register L are exchanged with the contents of register E

Ex opcode operand

		D	A2	03	E			D	D3	08	E
XCHG	None	H	D3	08	L		XCHG	H	A2	03	L

⇒ SPHL: Copy H and L registers to the stack pointer
→ The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.
Ex opcode operand.
SPHL None

NEW INDIA
PAGE NO.: 4 DATE

⇒ XTHL: Exchange H and L with top of stack
→ The contents of L register exchanged with location pointed by stack Pointer register. The contents of the H register are exchanged with next stack location (SP+1).
Ex: XTHL None

⇒ PUSH: Push the Register Pair onto the stack
→ The contents of the register Pair designated in the operand are copied onto the stack.
→ The SP register is decremented and high order register (B,D,H) copied into that location
→ The SP register is decremented and low-order register (C,E,L) are copied into that location.
Ex: Opcode Operand
PUSH Rp PUSH B

⇒ Pop: Pop off stack to the register Pair
→ The contents of the memory location pointed by the stack Pointer register are copied to the low-order register (C,E,L) of the operand.
→ The SP Pointer is incremented and content of memory location copied to the high-order register (B,D,H) of the operand.
→ The SP register is again incremented by 1.

⇒ OUT: output from Accumulator to 8-bit Port
Ex: OUT 8 bit Port address

⇒ IN: Input data to accumulator from a Port with 8-bit address
Ex: IN 8-bit Port address

B) Arithmetic Instruction

NEW INDIA
PAGE NO.: 5 DATE

⇒ ADD: Add register or memory, to the accumulator
→ The contents of the operand are added to the contents of the accumulator and the result is stored in the accumulator
→ opcode operand ADD B ; A = A + B
ADD M/R

⇒ ADC: Add register to the accumulator with carry
→ The contents of the operand and the carry flag are added to the contents of the accumulator and result is stored in the accumulator
→ ADC M/R ADC B; A = A + B + CY

⇒ ADI: Add the immediate to the accumulator
→ The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator.
→ ADI 8-bit data ADI 03; A = A + 03h

⇒ ACI: Add the immediate to the accumulator with carry
→ The 8-bit data and the carry flag are added to the content of the accumulator and stored in there.
→ ACI 8-bit data ACI 03; A = A + 03h + CY

⇒ DAD: Add the register pair to H and L registers
→ The 16-bit contents of the specified register pair are added to contents of the HL register and the sum is stored in the HL register.
→ DAD B/D DAD B

NEW INDIA
PAGE NO.: 6 DATE

⇒ SUB: Subtract the register or the memory from the accumulator
→ The contents of the operand (R/m) are subtracted from the contents of the accumulator, and the result is stored in the accumulator.
→ All flags are modified to reflect the result of the subtraction.
→ SUB R/M SUB B; A=A-B

⇒ SBB: Subtract the source and borrow from the AC.
→ The contents of the operand (R/M) and the Borrow flag are subtracted from the contents of the AC and the result is placed in the AC.
→ All flags are modified to reflect the result.
→ SBB R/M SBB B; A=A-B-CY

⇒ SUI: Subtract the immediate from the accumulator
→ The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator.
→ All flags are modified to reflect the result.
→ SUI 8-bit data SUI 08h; A=A-08h

⇒ SBI: Subtract the immediate from the accumulator with borrow.
→ The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result stored in AC.
→ All flags are modified to reflect the result of subtraction
→ SBI 8-bit data SBI 08h; A=A-(08h+CY)

NEW INDIA
PAGE NO.: 7 DATE

⇒ INR: Increment the register or the memory by 1.
→ The contents of the designated register or memory are incremented by 1 and result is stored in the same place.
→ INR R/M INR B

⇒ INX: Increment Register Pair by 1
→ The contents of the designated ~~R/M~~ register pair is incremented by 1 and the result is stored in the same place.
→ INX Rp

⇒ DCR: Decrement the register or the memory by 1
→ The contents of the designated register or memory are decremented by 1 and the result is stored in the same place.
→ DCR R/M DCR B

⇒ DCX: Decrement register Pair by 1
→ The contents of the designated register pair are decremented by 1 and their result is stored at the same place.
→ DCX Rp DCX B; decrements BC

⇒ DAA: Decimal adjust accumulator
→ The contents of the accumulator are changed from a binary value to two 4-bit BCD digits.
→ if lower-order 4 bits in the accumulator > 9 or AC flag = 1
Then DAA adds 6 to the low-order bits.
→ if higher-order 4 bits in the accumulator > 9 or CY flag = 1
Then DAA adds 6 to the high-order bits.

c) Branching x Instruction

NEW INDIA
PAGE NO.: 8 DATE

⇒ JMP: Jump unconditionally
→ The program sequence is transferred to the memory address given in the operand
→ JMP 16-bit address JMP 2030 H

⇒ Jump conditionally

Opcode	Operand	Description	Example
JC	16 bit address	Jump on carry, flag CY = 1	JC 2030H
JNC		Jump on NO carry, flag CY = 0	JNC 2030H
JZ		Jump on zero, flag Z = 1	JZ 2030H
JNZ		Jump on NO zero, flag Z = 0	JNZ 2030H
JP		Jump on Positive, flag S = 0	JP 2030H
JM		Jump on minus, flag S = 1	JM 2030H
JPE		Jump on Parity Even, flag P = 1	JPE 2030H
JPO		Jump on Parity odd, flag P = 0	JPO 2030H

⇒ CALL conditionally

Opcode	Operand	Description	Example
CC	16 bit address	Call on carry, flag CY = 1	CC 2030H
CNC		Call on No carry, flag CY = 0	CNC 2030H
CZ		Call on zero, flag Z = 1	CZ 2030H
CNZ		call on no zero, flag Z = 0	CNZ 2030H
CP		Call on Positive, S = 0	CP 2030H
CM		Call on minus, flag S = 1	CM 2030H
CPE		Call on Parity Even, flag P = 1	CPE 2030H
CPO		Call on Parity Odd, flag P = 0	CPO 2030H

NEW INDIA
PAGE NO.: 9 DATE

⇒RET: Return from subroutine unconditionally
→ The program sequence is transferred from the subroutine to the calling program.
→ RET 16-bit address

⇒ RET conditionally

Opcode	Operand	Description	Example
RC	16bit address	Return on carry, CY=1	RC 2030H
RNC		Return on no carry, CY=0	RNC 2030H
RZ		Return on zero, Z=1	RZ 2030H
RNZ		Return on NO zero, Z=0	RNZ 2030H
RP		Return on Positive, S=0	RP 2030H
RM		Return on minus, S=1	RM 2030H
RPE		Return on Parity Even, P=1	RPE 2030H
RPO		Return on Odd Parity, P=0	RPO 2030H

⇒ PCHL: Load the program counter with HL contents
→ The contents of registers H&L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
→ PCHL None PCHL

⇒ RST: Restart
→ The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations

NEW INDIA
PAGE NO.: 16 DATE

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

Ex:

Opcode	Operand
RST	0-7

→ The 8085 has additionally 4 interrupts.

Opcode	Operand	Description	Example
TRAP	None	It restart from address 0024H	TRAP
RST 5.5	None	It restart from address 002CH	RST 5.5
RST 6.5	None	It restart from address 0034H	RST 6.5
RST 7.5	None	It restart from address 003CH	RST 7.5

d) Logical Instruction

⇒ CMP: Compare the registers or memory with the accumulator

Note: The flag register in the 8085 is called the Program status word (PSW) register.

→ The contents of operand (r/m) are compared with the contents of the accumulator.

if (A) < (reg/mem): CY=1, carry flag is set

if (A) = (reg/mem): Z = 1, zero flag is set

if (A) > (reg/mem): CY=0, Z=0, carry and zero flags are reset

NEW INDIA
PAGE NO.: 11 DATE

⇒ CPI: Compare immediate with the accumulator
→ The second byte data is compared with contents of the accumulator. The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < data: CY = 1
if (A) = data; Z = 1
if (A) > data : CY=0, Z=0
Ex CPI 8-bit data

⇒ ANA: Logical AND register or memory with the accumulator
→ The contents of the accumulator are logically ANDed with the contents of the operand and the result is placed in the accumulator
→ S, Z, P are modified to reflect the result of the operation
→ CY is reset, AC is set Ex ANA R/M

⇒ ANI: Logical AND immediate with the accumulator
→ the contents of the accumulator are logically ANDed with the 8-bit data and the result is placed in the accumulator.
→ S, Z, P are modified to reflect the result.
→ CY is reset, AC is set. Ex ANI 8-bit data

⇒ XRA: Exclusive OR register or memory with the accumulator
→ The contents of the accumulator are Exclusive ORed with the contents of the operand (R/M) and the result is placed in the accumulator.
→ S, Z, P are modified to reflect the result
→ CY is reset, AC is set Ex XRA M/R

NEW INDIA
PAGE NO.: 12 DATE

XRI: Exclusive OR immediate with the accumulator
→ The contents of the accumulator are logically XORed with the 8 bit data (operand) and the result is placed in the accumulator.
→ S, Z, P are modified to reflect the result of the operation.
→ CY is reset, AC is set. Ex XRI 02H

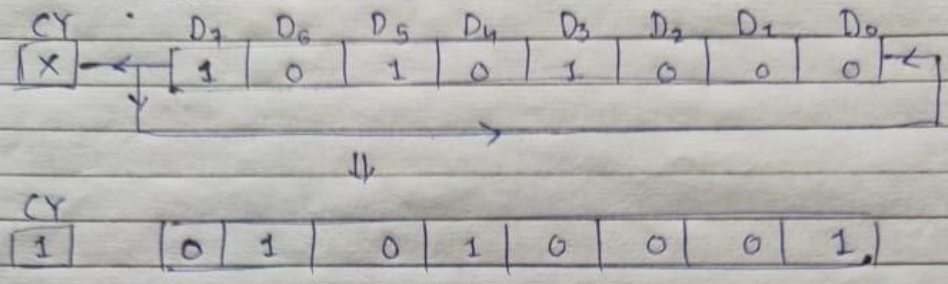
ORA: Logical OR register or memory with the accumulator
→ The contents of the accumulator are logically ORed with the contents of the operand (R/M) and the result placed in the accumulator.
→ S, Z, P are modified to reflect the result and CY is reset, AC is set
→ ORA B

⇒ ORI: logical OR immediate with the accumulator
→ The contents of the accumulator are logically ORed with the 8-bit data and the result is placed in the accumulator.
→ S, Z, P are modified to reflect the result the operation.
→ CY is reset, AC is set.
→ ORI 02H

⇒ RLC: Rotate the accumulator left
→ Each binary bit of the accumulator is rotated left by one position.
→ Bit D_7 is placed in the position of D_0 as well as in the Carry flag.
→ CY is modified according to bit D_7.
→ S, Z, P, AC are not affected.

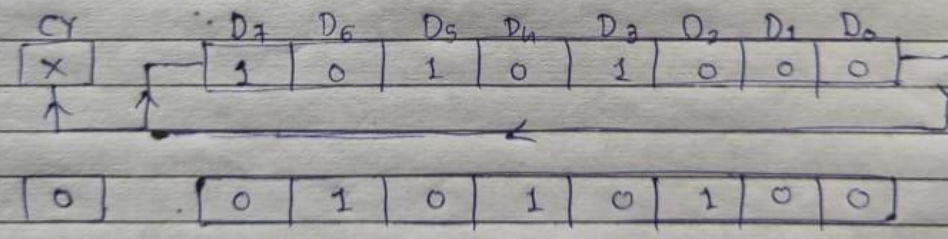
NEW INDIA
PAGE NO.: 13 DATE

RCL None
accumulator



⇒ RRC : Rotate the accumulator right.
→ Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the carry flag.
→ CY is modified according to bit D0
→ S, Z, P, AC are not affected.

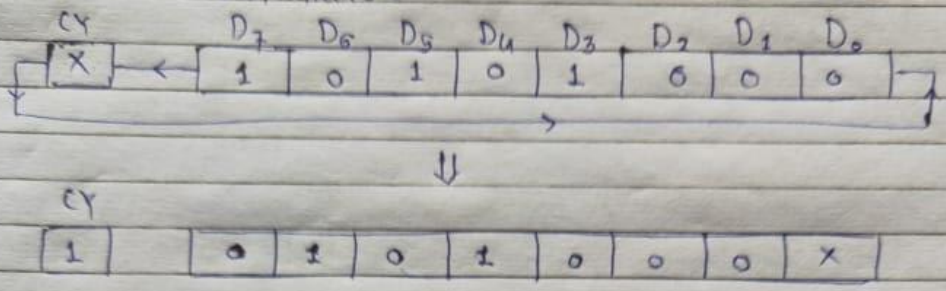
RRC None
accumulator



⇒ RAL: Rotate the accumulator left through carry
→ Each binary bit of the accumulator is rotated left by one position through the carry flag.
→ Bit D7 is placed in the carry flag, and the carry flag is placed in the least significant position D0.
→ CY is modified according to bit D7.
→ S, Z, P, AC are not affected.

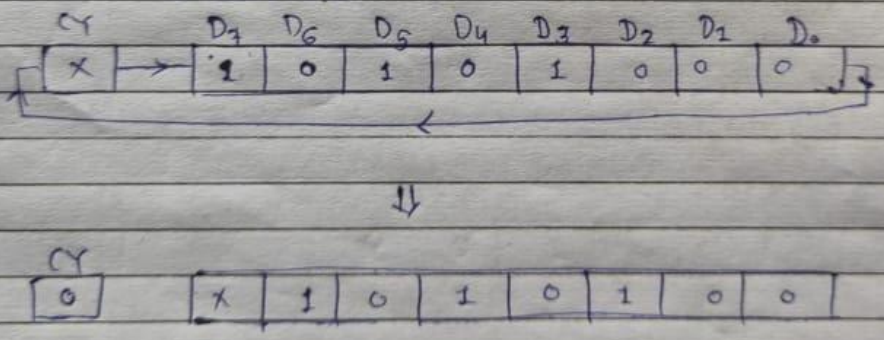
NEW INDIA
PAGE NO.: 14 DATE

RAL None
accumulator



⇒ RAR: Rotate the accumulator right through carry
→ Each binary bit of the accumulator is rotated right by one position through the carry flag.
→ Bit D0 is placed in the carry flag, and the carry flag is placed in the most significant position D7
→ CY is modified and S, Z, P, AC are not affected

RAR None
accumulator



⇒ CMA: complement accumulator
→ CMA None A ← [A']

⇒ CMC: complement carry
→ CMC None CY ← [CY']

⇒ STC: Set carry
→ STC CY = 1

NEW INDIA
PAGE NO.: 15 DATE

e) Control instructions:

⇒ NOP:
→ No operation is performed. The instruction is fetched and decoded However no operation is executed.
→ It is used to increase Processing time of execution.
→ 1 CPU cycle is wasted to execute a NOP instruction

⇒ HLT:
→ The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.

⇒ DI:
→ The interrupt enable flip-flop is reset and all the interrupts expect the ~~TA~~ TRAP are disabled. No flags are effected.

⇒ EI:
→ The interrupt enable flip-flop is set and all interrupts are enabled.
→ No flags are affected.
→ After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset thus disabling the interrupts.
→ This instruction is necessary to re enable the interrupts (except TRAP).

NEW INDIA
PAGE NO.: 16 DATE

⇒ SIM: This is a multipurpose instruction used to

Practical – 3

Aim: Write a program for addition of two 16-bit numbers using carry using 8085.

Mnemonic	Operand	Comment
LDA	2000	Load Accumulator with lower bits of first no
MOV	B,A	copy Accumulator to Reg B
LDA	2002	Load Accumulator with lower bits of second no
ADD	B	Add Reg B to Accumulator
MOV	E,A	Copy Accumulator to Reg E
LDA	2001	Load Accumulator with higher bits of first no
MOV	B,A	Copy Accumulator to Reg B
LDA	2003	Load Accumulator with higher bits of second no
ADC	B	Add Reg B with Carry to Accumulator
MOV	D,A	Copy Accumulator to Reg D
HLT		stop
# ORG	2000H	
# DB	10H,34H,05H,64H	

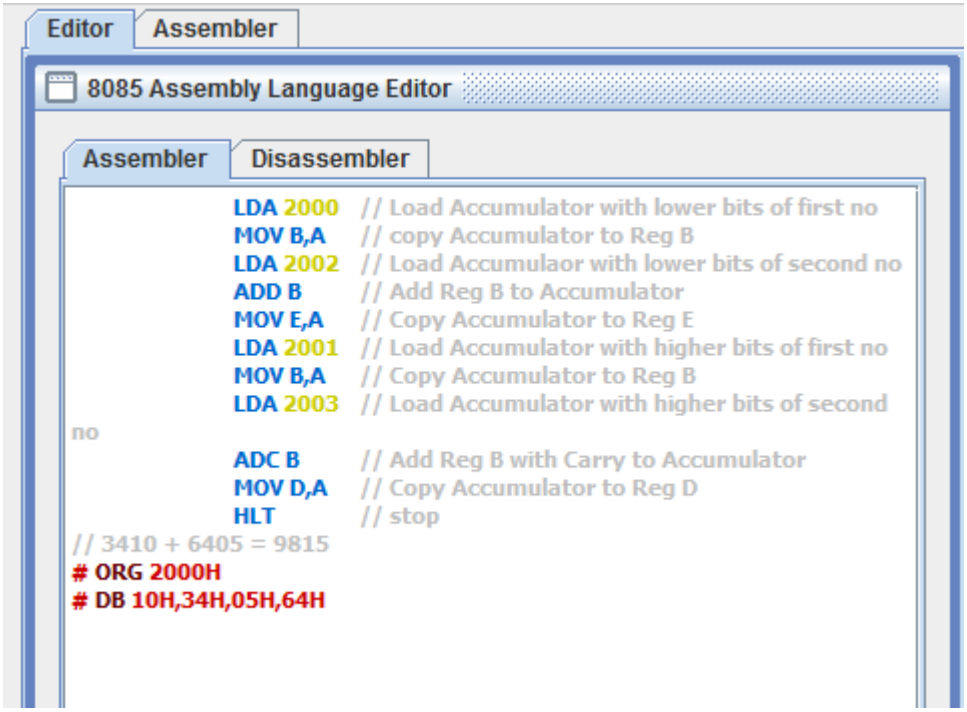
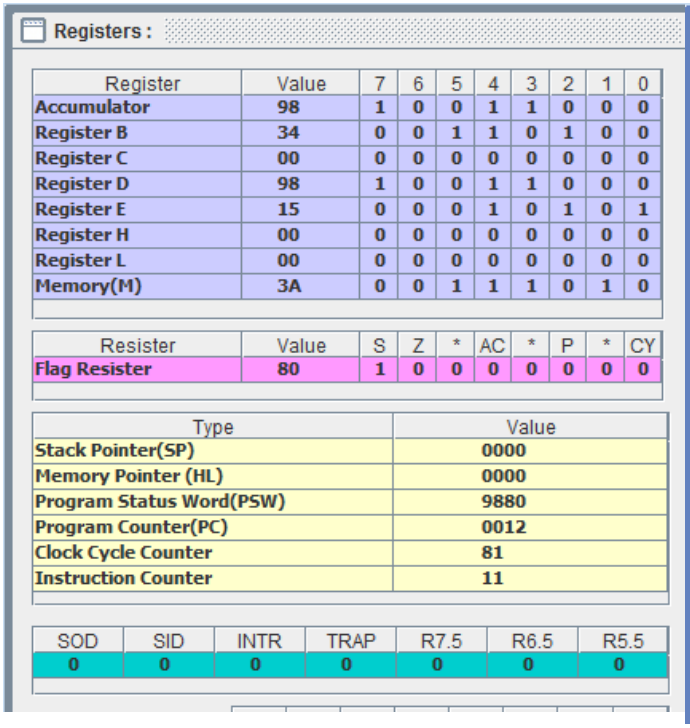


Fig3.1 – sum of two 16 bit



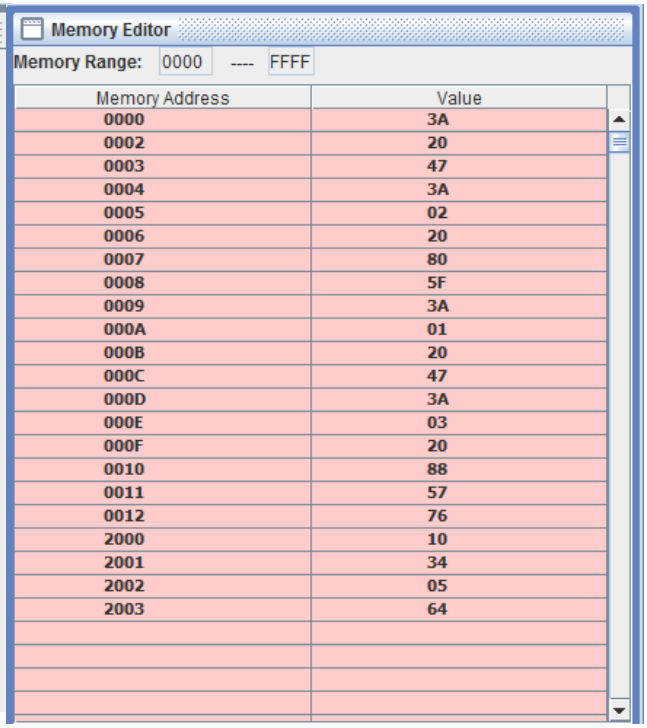
Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	98	1	0	0	1	1	0	0	0
Register B	34	0	0	1	1	0	1	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	98	1	0	0	1	1	0	0	0
Register E	15	0	0	0	1	0	1	0	1
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	3A	0	0	1	1	1	0	1	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	80	1	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	9880
Program Counter(PC)	0012
Clock Cycle Counter	81
Instruction Counter	11

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0



Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
0000	3A
0002	20
0003	47
0004	3A
0005	02
0006	20
0007	80
0008	5F
0009	3A
000A	01
000B	20
000C	47
000D	3A
000E	03
000F	20
0010	88
0011	57
0012	76
2000	10
2001	34
2002	05
2003	64

Fig3.2 – Registers

Fig3.3 – Memory Address

b. Write a program for subtraction of two 8-Bit Numbers. (Display of Borrow) using 8085.

Label	Mnemonic	Operand	Comment
	MVI	C,00	Initialize C to 00
	MVI	A,A9	Initialize A to A9h
	MVI	B,AB	Initialize B to ABh
	SUB	B	A=A-B
	JNC	SKIP	Jump on no carry
	INR	C	Increment value in Reg C
	CMA		Complement Accumulator contents
	INR	A	Increment value in Accumulator
SKIP:	STA	2000	Store the value of Accumulator to memory
	MOV	A,C	Move content of Reg C to Accumulator
	HLT		Terminate the program

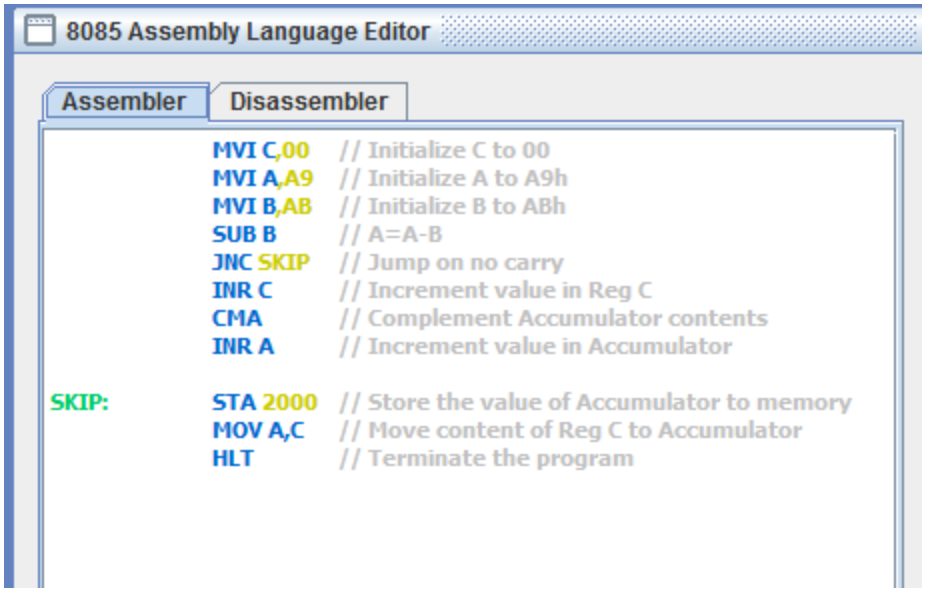


Fig3.4 – Subtraction of 8 bit

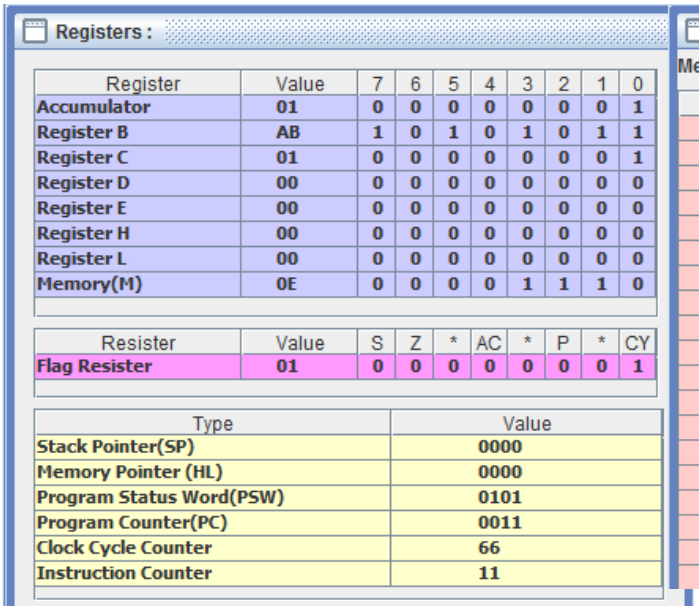


Fig3.5 – Registers

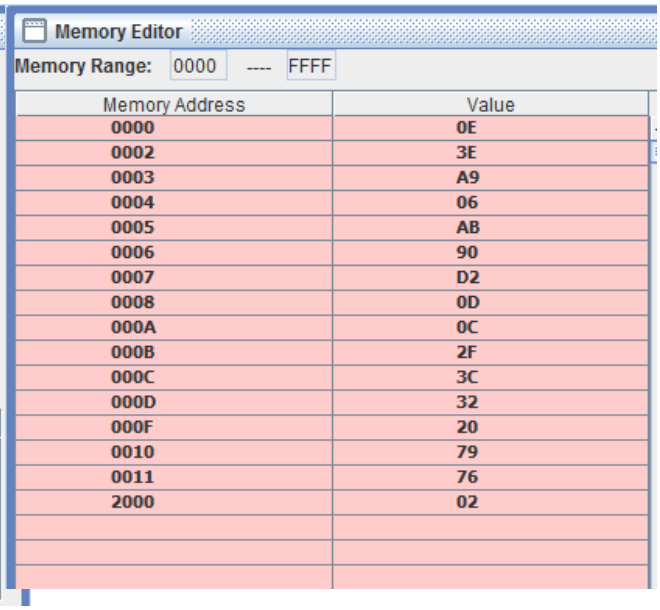


Fig3.6 – Memory Address

Practical – 4

Aim: Write a program for Multiplication of Two 8-Bit Numbers By Bit Rotation Method using 8085.

Label	Mnemonic	Operand	Comment
	LXI	H,2000	// Load HL pair
	MOV	D,M	// Load first number in Reg D
	INX	H	// increment Hl pair
	MOV	B,M	// Load second number in Reg B
	MVI	C,00	// initialize Reg C to 00
START:	MOV	A,D	// Copy Reg D into Accumulator
	ANI	FF	// And operation with Accumulator content
	JZ	CNT	// Check multiplicand is 0 than stop
	RRC		// Rotate Accumulator Right
	JNC	SKIP	// if ANI MSB bit is 0 than skip
	MOV	A,C	// Copy Reg c to Accumulator
	ADD	B	// Add Reg B to Accumulator
	MOV	C,A	// Copy Accumulator to Reg C
SKIP:	MOV	A,D	// Copy Reg D to Accumulator
	STC		// set Carry flag
	CMC		// compliment carry
	RAR		// Do right shift
	MOV	D,A	// Copy Accumulator to Reg D
	MOV	A,B	// Copy Reg B to Accumulator
	STC		// set Carry flag
	CMC		// compliment carry
	RAL		// Do left shift
	MOV	B,A	// copy Accumulator to Reg B
	JMP	START	// JMP to start
CNT:	MOV	A,C	// Copy Reg C to Accumulator
		HLT	// stop
# ORG 2000H			
# DB 05H,04H			

Assembler | Disassembler

```
                LXI H,2000 // Load HL pair
                MOV D,M    // Load first number in Reg D
                INX H      // increment Hl pair
                MOV B,M    // Load second number in Reg B
                MVI C,00   // initialize Reg C to 00

START:          MOV A,D    // Copy Reg D into Accumulator
                ANI FF     // And operation with Accumulator content
                JZ CNT     // Check multipicant is 0 than stop
                RRC        // Roatate Accumulator Right
                JNC SKIP   // if ANI MSB bit is 0 than skip
                MOV A,C    // Copy Reg c to Accumulator
                ADD B      // Add Reg B to Accumulator
                MOV C,A    // Copy Accumulator to Reg C

SKIP:           MOV A,D    // Copy Reg D to Accumulator
                STC        // set Carry flag
                CMC        // compliment carry
                RAR        // Do right shift
                MOV D,A    // Copy Accumulator to Reg D
                MOV A,B    // Copy Reg B to Accumulator
                STC        // set Carry flag
                CMC        // compliment carry
                RAL        // Do left shift
                MOV B,A    // copy Accumulator to Reg B
                JMP START          // JMP to start

CNT:            MOV A,C    // Copy Reg C to Accumulator
                HLT        // stop
# ORG 2000H
# DB 05H,04H
```

Fig4.1 Two 8-bit Multiplication

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	14	0	0	0	1	0	1	0	0
Register B	20	0	0	1	0	0	0	0	0
Register C	14	0	0	0	1	0	1	0	0
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	20	0	0	1	0	0	0	0	0
Register L	01	0	0	0	0	0	0	0	1
Memory(M)	04	0	0	0	0	0	1	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	54	0	1	0	1	0	1	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	2001
Program Status Word(PSW)	1454
Program Counter(PC)	0023
Clock Cycle Counter	331
Instruction Counter	64

Fig4.2 Registers

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
0000	21
0002	20
0003	56
0004	23
0005	46
0006	0E
0008	7A
0009	E6
000A	FF
000B	CA
000C	22
000E	0F
000F	D2
0010	15
0012	79
0013	80
0014	4F
0015	7A
0016	37
0017	3F
0018	1F
0019	57
001A	78
001B	37
001C	3F
001D	17

Fig4.3 – Memory Address

b. Division of Two 8-Bit Numbers by Repeated Subtraction Method using 8085.

Label	Mnemonic	Operand	Comment
	LXI	H,2000	Load HL pair with 2000
	MOV	A,M	Copy Dividend to Accumulator
	INX	H	Increment HL pair
	MOV	B,M	Copy Divisor to Reg B
	MVI	C,00	Initialize Reg C to 00
L1:	CMP	B	Compare Reg B with Accumulator
	JC	CNT	if Z=1;Jump to CNT
	SUB	B	Subtract Reg B from Accumulator
	INR	C	Increment Reg C
	JMP	L1	Jump to L1
CNT:		HLT	Stop
# ORG 2000H			
# DB 0DH,04H			

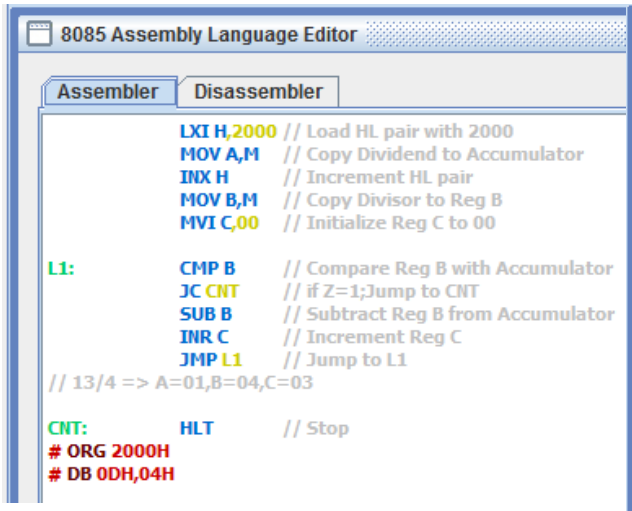


Fig4.3 Two 8-bit Division

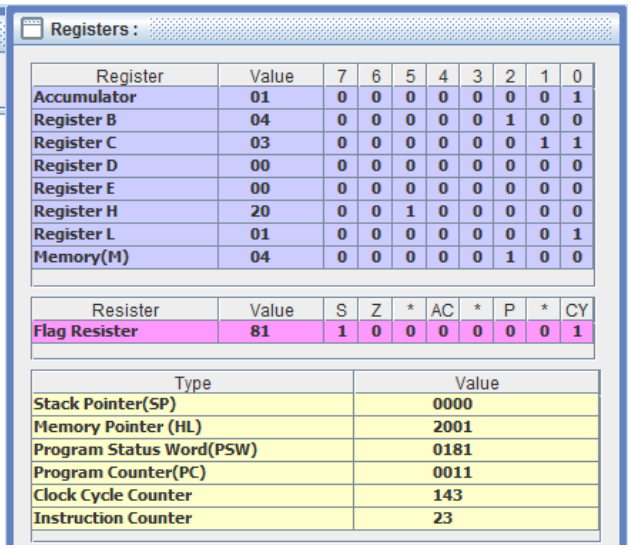


Fig4.4 Registers

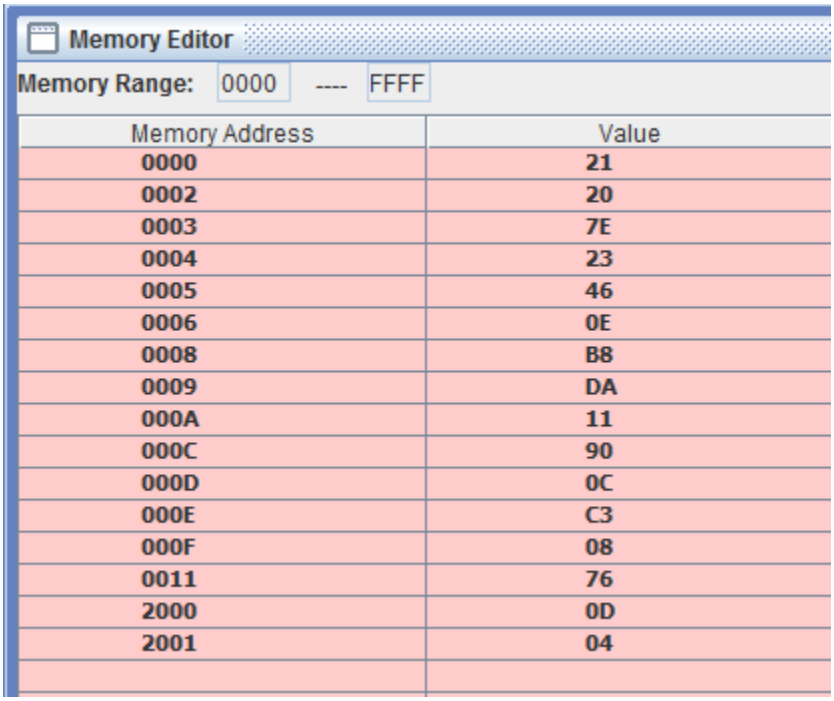


Fig4.5 Memory Address

Practical – 5

Aim: Write a Program using 8085 for Finding Square-Root of a Number.

Label	Mnemonic	Operand	Comment
	MVI	B,01	Initialize Reg B to 01
	MVI	C,01	initialize Reg C to 01
	LDA	2000	load the number to Accumulator
START:	SUB	B	Subtract Reg B from Accumulator
	JZ	CNT	if Z=1;Accumulator content is 0 than CNT
	INR	B	Increment Reg B
	INR	B	Increment Reg B
	INR	C	Increment Reg C
	JMP	START	Jump to start
CNT:	MOV	A,C	Copy Reg C to Accumulator
	HLT		Stop
# ORG 2000H			
# DB 09H			

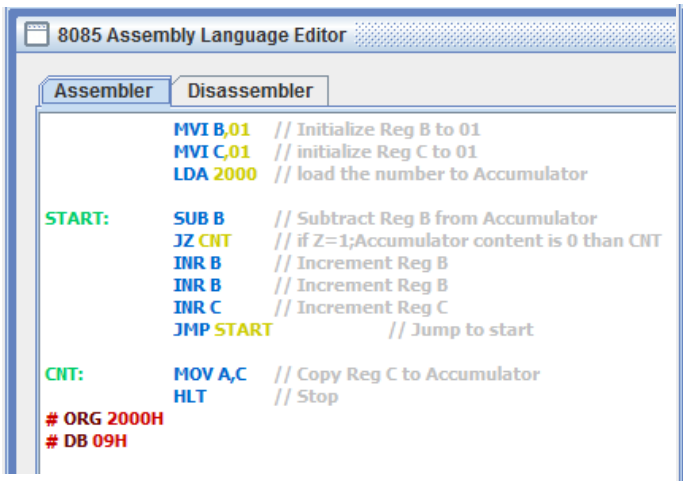


Fig5.1 – Square-root of number

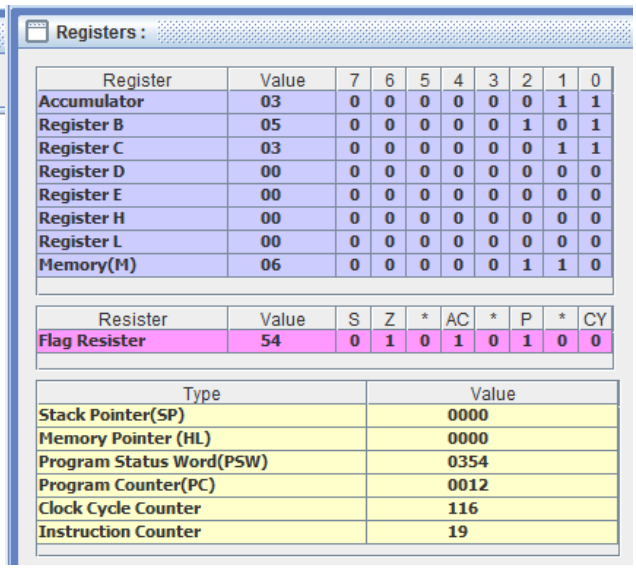


Fig5.2 – Registers

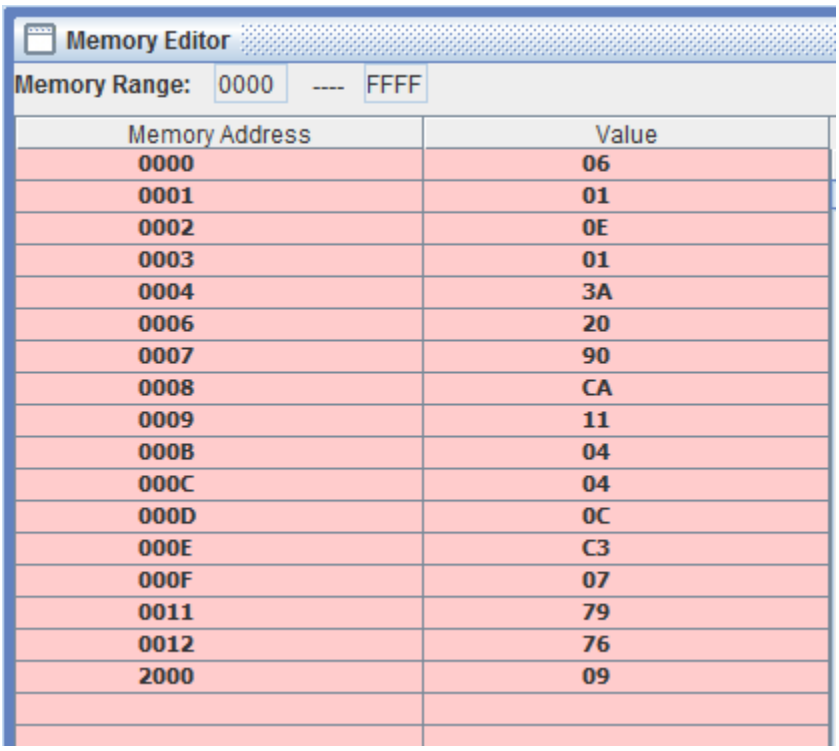


Fig5.3 – Memory Address

b. Write a program to find a factorial of a given number using 8085.

Label	Mnemonic	Operand	Comment
	LXI	H,2000	Load Hl pair with 2000
	MOV	B,M	Copy Memory to Reg B
	MVI	A,00	Initialize Accumulator with 00
	MOV	D,B	Copy Reg B to Reg D
	DCR	B	Decrement Reg B
START:	JZ	CNT	if Z=1; Jump to CNT
	MOV	E,B	Copy Reg B to Reg E
MUL:	ADD	D	Add Reg D to Accumulator
	DCR	E	Decrement Reg D
	JNZ	MUL	if Z=0; Jump to MUL
	MOV	D,A	Copy Accumulator to Reg D
	MVI	A,00	Initialize Accumulator to 00
	DCR	B	Decrement Reg B
	JMP	START	Jump to START
CNT:	MOV	A,D	Copy Reg D to Accumulator
	HLT		Stop
# ORG 2000H			
# DB 05H			

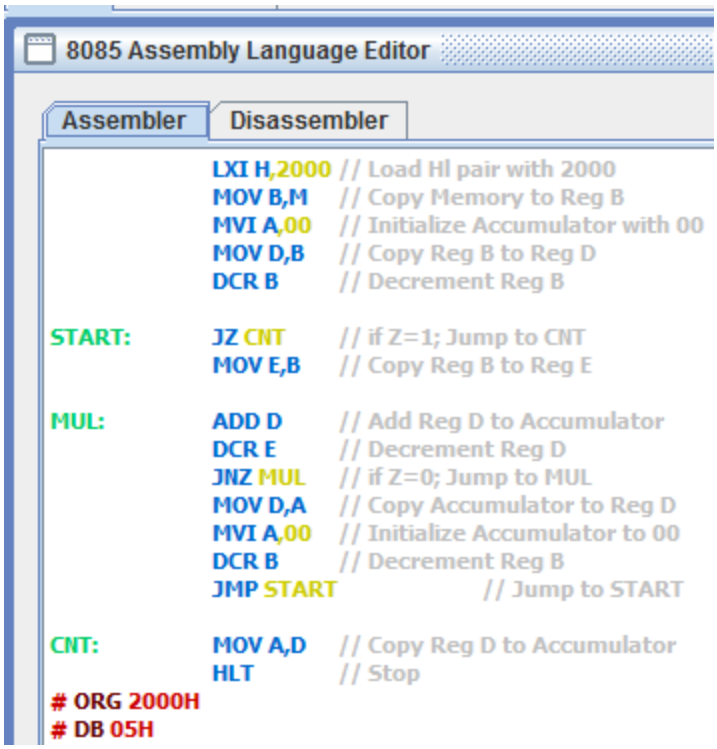


Fig5.4 Factorial of number

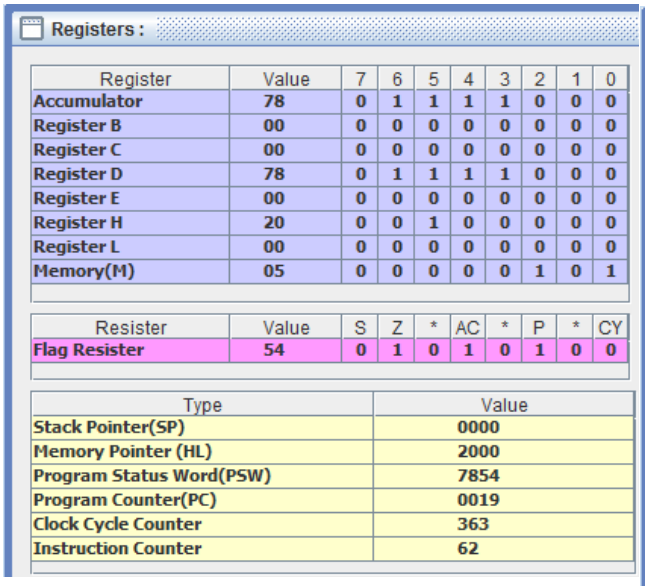


Fig5.5 Registers

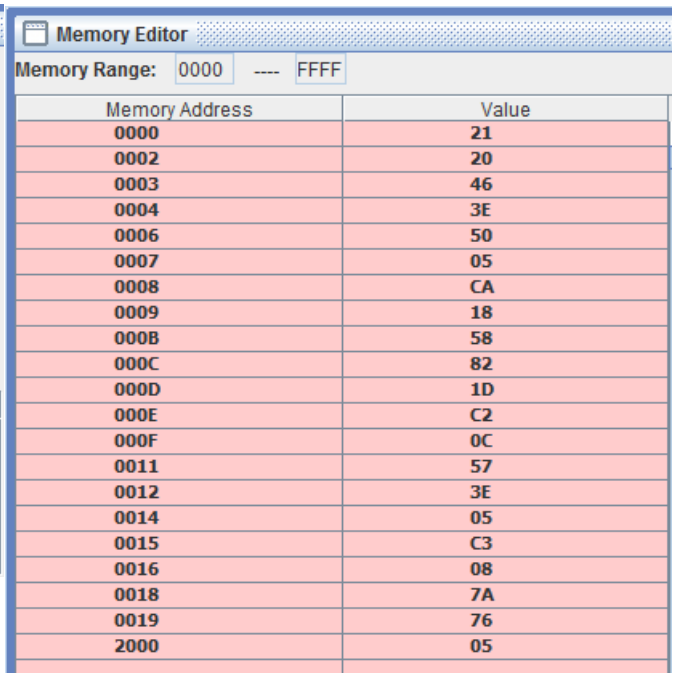


Fig5.6 Memory Address

Practical – 6

Aim: Write a program to move a block of data using 8085.

Label	Opcode	Operand	Comment
	LXI	B,2050	// Load BC pair with 2050H
	LXI	D,3050	// Load DE pair with 3050H
	MVI	H,05	// Initialize Reg H to 05
START:	LDAX	B	// Load Accumulator Direct from BC pair
	INX	B	// Increment BC pair
	STAX	D	// Store Accumulator Direct to DE pair
	INX	D	// increment DE pair
	DCR	H	// Decrement Reg H
	JNZ	START	// Jump on no Zero to START
	HLT		// stop
# ORG 2050H			
# DB 05H,06H,02H,0AH,04H			

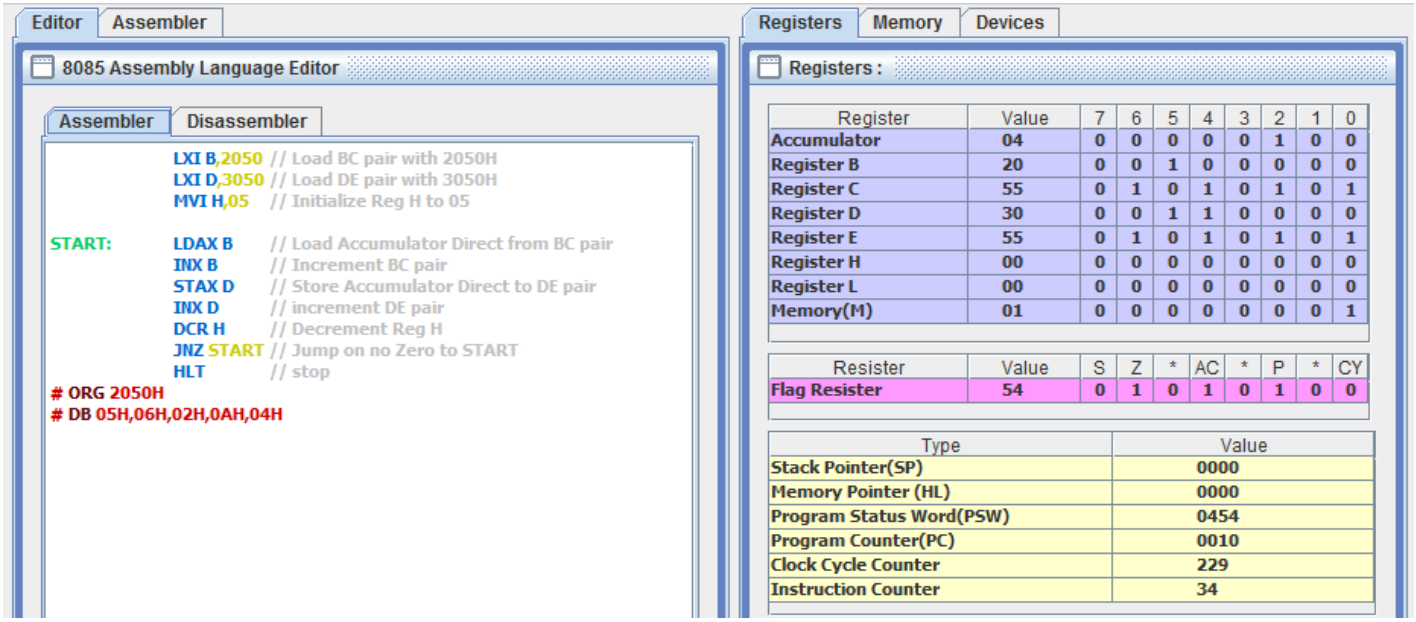
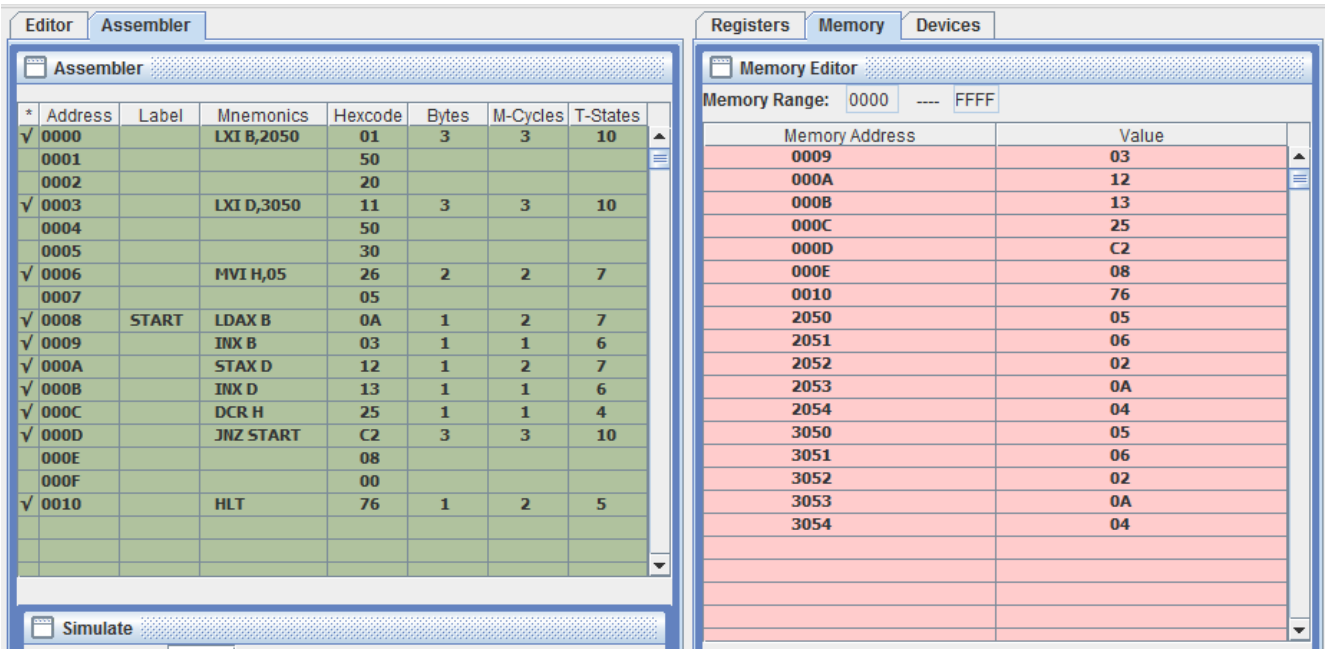


Fig6.1 - Editor Register



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		LXI B,2050	01	3	3	10
	0001			50			
	0002			20			
√	0003		LXI D,3050	11	3	3	10
	0004			50			
	0005			30			
√	0006		MVI H,05	26	2	2	7
	0007			05			
√	0008	START	LDAX B	0A	1	2	7
√	0009		INX B	03	1	1	6
√	000A		STAX D	12	1	2	7
√	000B		INX D	13	1	1	6
√	000C		DCR H	25	1	1	4
√	000D		JNZ START	C2	3	3	10
	000E			08			
	000F			00			
√	0010		HLT	76	1	2	5

Memory Address	Value
0009	03
000A	12
000B	13
000C	25
000D	C2
000E	08
0010	76
2050	05
2051	06
2052	02
2053	0A
2054	04
3050	05
3051	06
3052	02
3053	0A
3054	04

Fig6.2 – Assembler Memory

b. Write a Program to Arrange Number in Ascending Order Using 8085.

Label	**Opcode**	**Operand**	**Comment**
	MVI	B,05	// Initialize Reg B with count
START:	MOV	C,B	// Copy Reg B to Reg C
	LXI	H,2000	// Load HL pair with 2000
LOOP:	MOV	A,M	// Copy Memory content to Accumulator
	INX	H	// Increment HL pair
	CMP	M	// Compare Accumulator Content with Memory Content
	JC	SKIP	// Jump on Carry(CY=1) to SKIP ;(A<M)
	MOV	D,M	// Copy Memory content to Reg D
	MOV	M,A	// Copy Accumulator to Memory
	DCX	H	// Decrement Hl pair
	MOV	M,D	// Copy Reg D to Memory
	INX	H	// Increment HL pair
SKIP:	DCR	C	// Decrement Reg C
	JNZ	LOOP	// Jump on Zero(Z=1) to LOOP
	DCR	B	// Decrement Reg B
	JNZ	START	// Jump on no Zero(Z=0) to START
	HLT		// stop
# ORG 2000H			
# DB 05H,03H,06H,02H,08H			

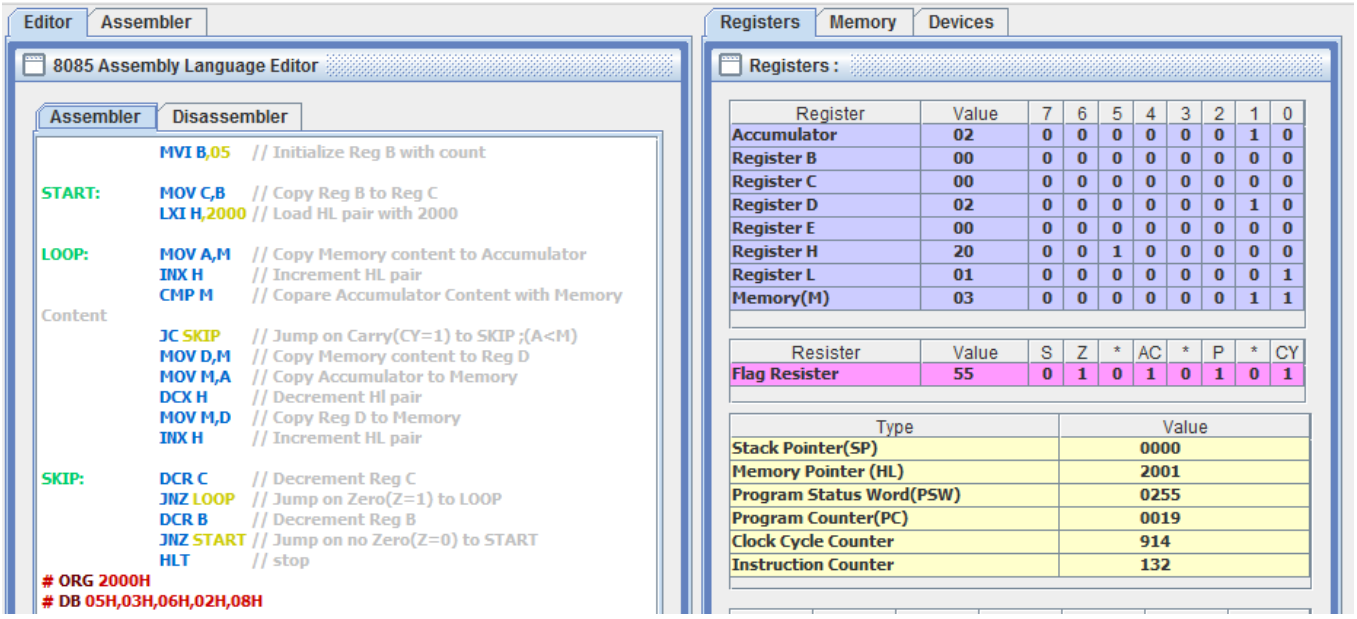


Fig6.3 - Editor Register

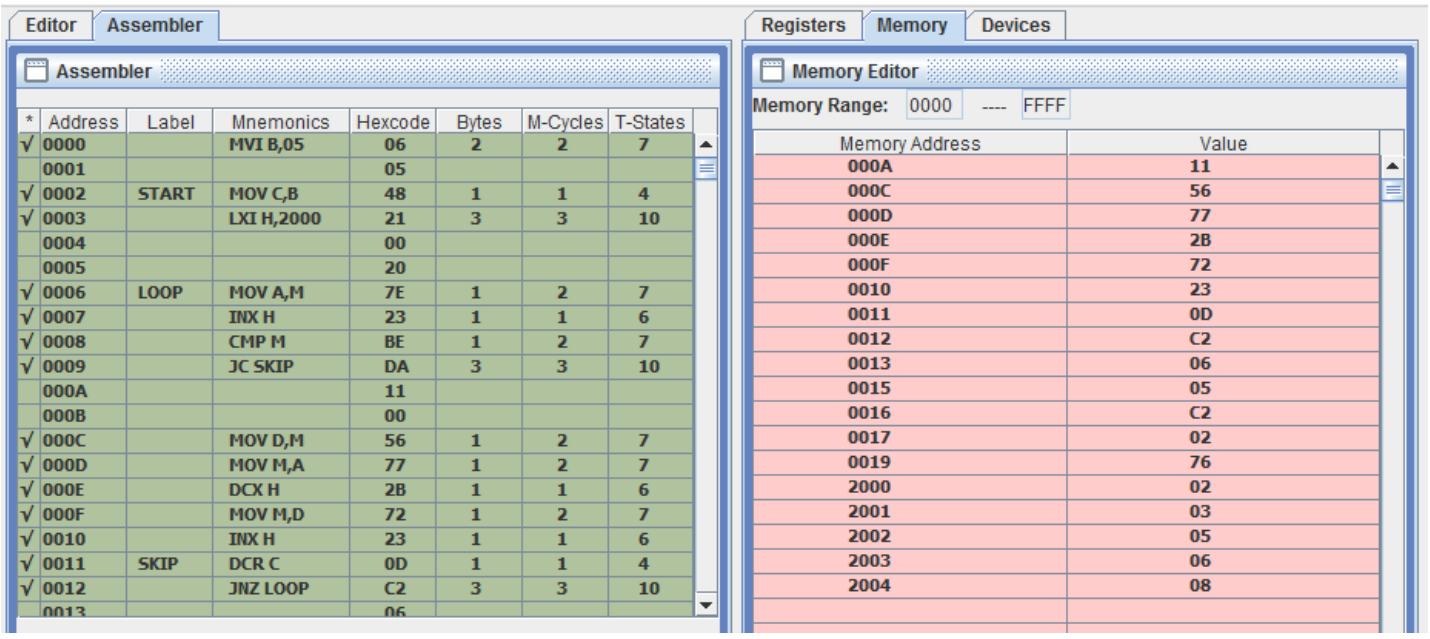


Fig6.4 – Assembler Memory

Practical – 7

A. Write a Program to Find GCD Of Two Numbers Using 8085.

LABEL	OPCODE	OPERAND	COMMENT
	MVI	A,3C	// Load no1 to Reg A
	MVI	B,24	// Load no2 to Reg B
START:	CMP	B	// compare Reg B with Accumulator
	JZ	CNT	// jump to CNT if Z=1
	JC	L1	// jump to L1 if C=1
	SUB	B	// Subtract Reg B from Reg A
	JMP	START	// jump to START
L1:	MOV	C,B	// Copy Reg B to Reg C
	MOV	B,A	// Copy Reg A to Reg B
	MOV	A,C	// Copy Reg C to Reg A
	JMP	START	// jump to START
CNT:	HLT	HLT	// stop

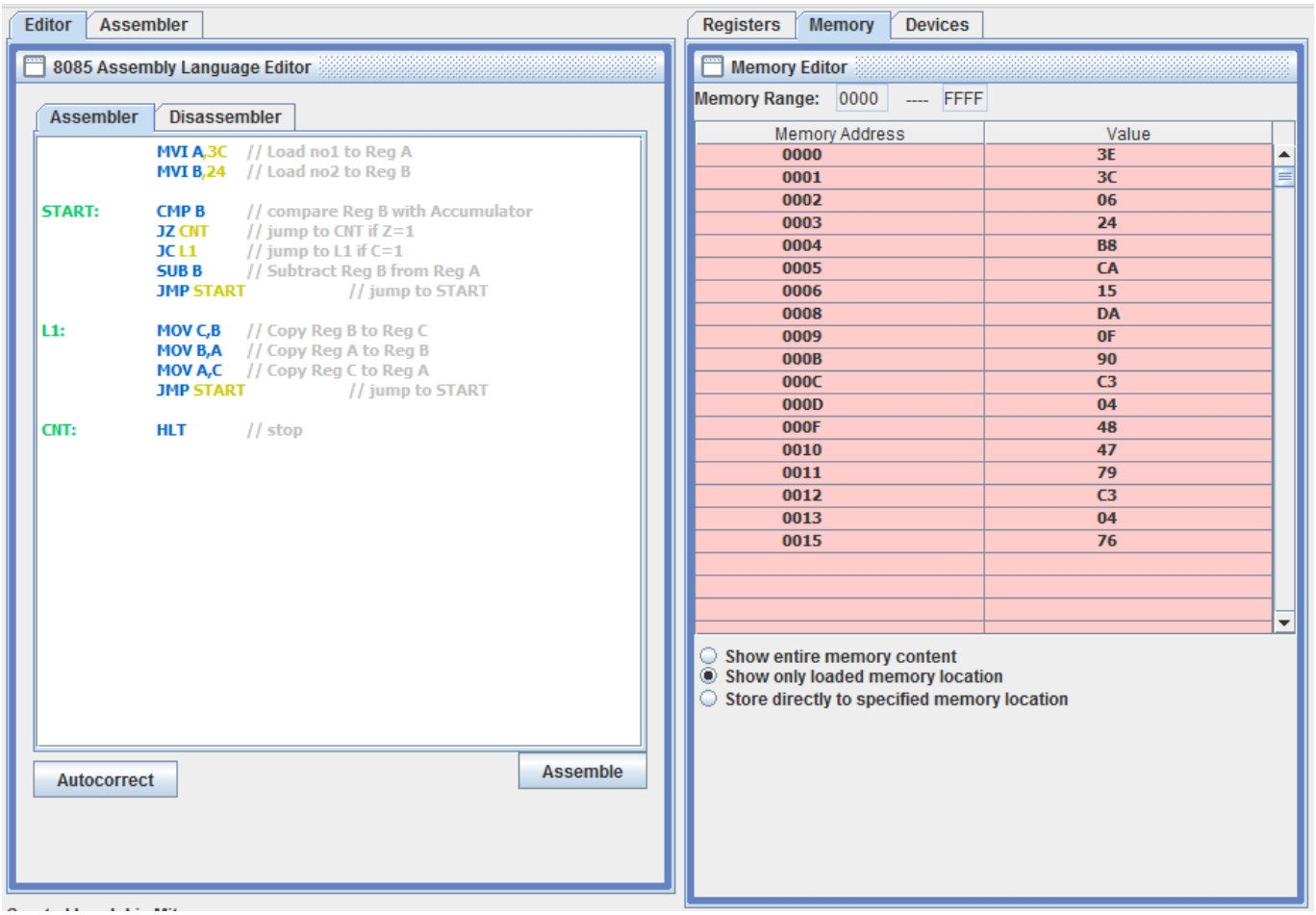


Fig7.1

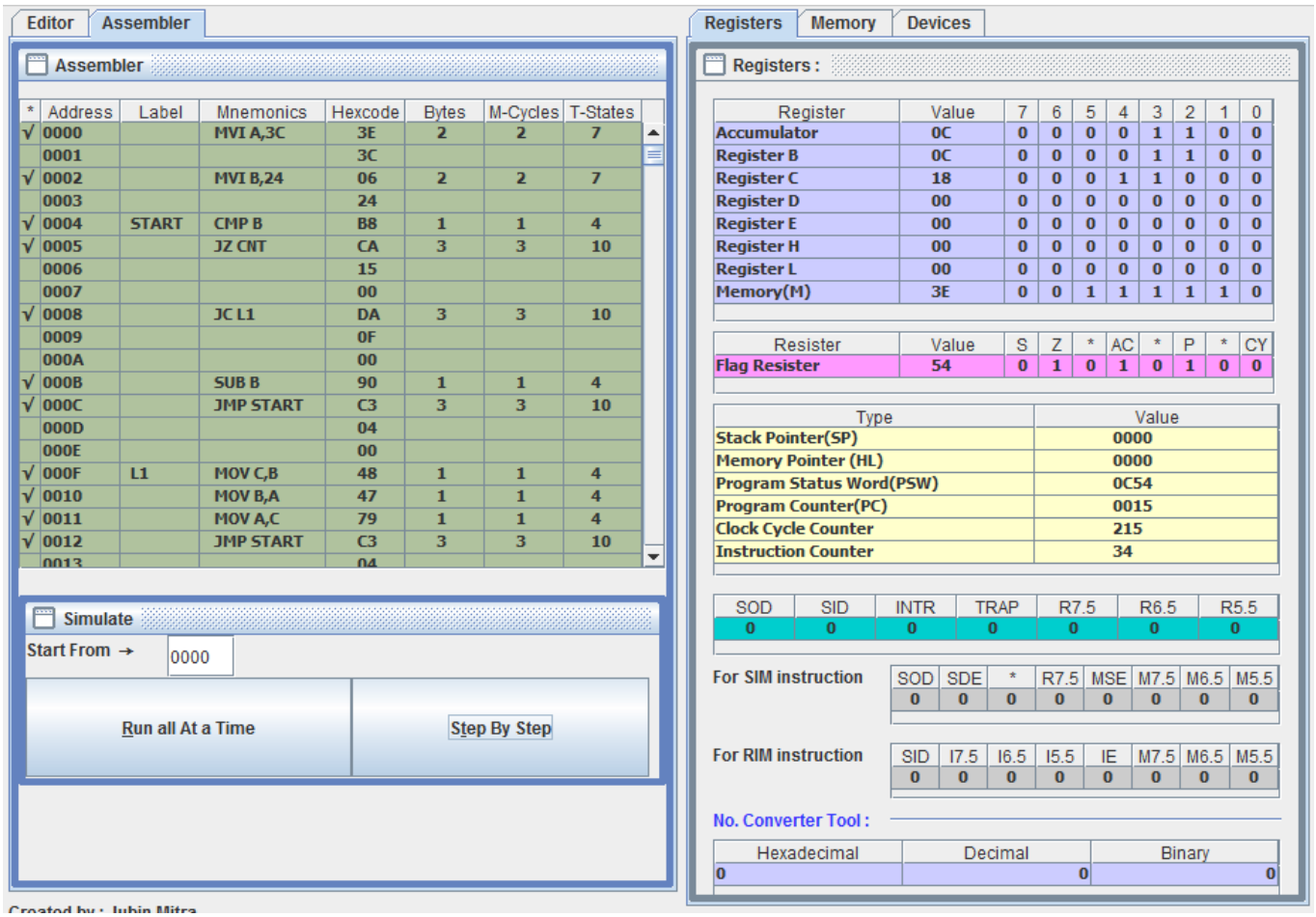

Editor
Assembler
Assembler
* Address Label Mnemonics Hexcode Bytes M-Cycles T-States
√ 0000 MVI A,3C 3E 2 2 7
0001 3C
√ 0002 MVI B,24 06 2 2 7
0003 24
√ 0004 START CMP B B8 1 1 4
√ 0005 JZ CNT CA 3 3 10
0006 15
0007 00
√ 0008 JC L1 DA 3 3 10
0009 0F
000A 00
√ 000B SUB B 90 1 1 4
√ 000C JMP START C3 3 3 10
000D 04
000E 00
√ 000F L1 MOV C,B 48 1 1 4
√ 0010 MOV B,A 47 1 1 4
√ 0011 MOV A,C 79 1 1 4
√ 0012 JMP START C3 3 3 10
0013 04
Simulate
Start From → 0000
Run all At a Time
Step By Step
Created by : Jubin Mitra
Registers
Memory
Devices
Registers :
Register Value 7 6 5 4 3 2 1 0
Accumulator 0C 0 0 0 0 1 1 0 0
Register B 0C 0 0 0 0 1 1 0 0
Register C 18 0 0 0 1 1 0 0 0
Register D 00 0 0 0 0 0 0 0 0
Register E 00 0 0 0 0 0 0 0 0
Register H 00 0 0 0 0 0 0 0 0
Register L 00 0 0 0 0 0 0 0 0
Memory(M) 3E 0 0 1 1 1 1 1 0
Resister Value S Z * AC * P * CY
Flag Resister 54 0 1 0 1 0 1 0 0
Type Value
Stack Pointer(SP) 0000
Memory Pointer (HL) 0000
Program Status Word(PSW) 0C54
Program Counter(PC) 0015
Clock Cycle Counter 215
Instruction Counter 34
SOD SID INTR TRAP R7.5 R6.5 R5.5
0 0 0 0 0 0 0
For SIM instruction
SOD SDE * R7.5 MSE M7.5 M6.5 M5.5
0 0 0 0 0 0 0 0
For RIM instruction
SID I7.5 I6.5 I5.5 IE M7.5 M6.5 M5.5
0 0 0 0 0 0 0 0
No. Converter Tool :
Hexadecimal Decimal Binary
0 0 0

Fig7.2

B. Write a Program to Add ‘N’ Two Digit BCD Numbers Using 8085.

LABEL	OPCODE	OPERAND	COMMENT
	LXI	H,2051	// Load hl pair with 2050
	MVI	C,04	// Initialize Reg C
	MVI	D,00	// Clear reg D
	DCR	C	// Decrement Reg C
	MOV	A,M	// Copy memory eontent to Reg A
START:	INX	H	// Increment Reg pair
	ADD	M	// Add memory contetnt to Reg A
	DAA		// Decimal Adjust After Addition
	JNC	CNT	// Jump to CNT if CY=0
	INR	D	// Increment Reg D
CNT:	DCR	C	// Decrement Reg C
	JNZ	START	// Jump to START if Z=0
	HLT		// stop
# ORG 2051H			
# DB 22H,05H,54H,61H			

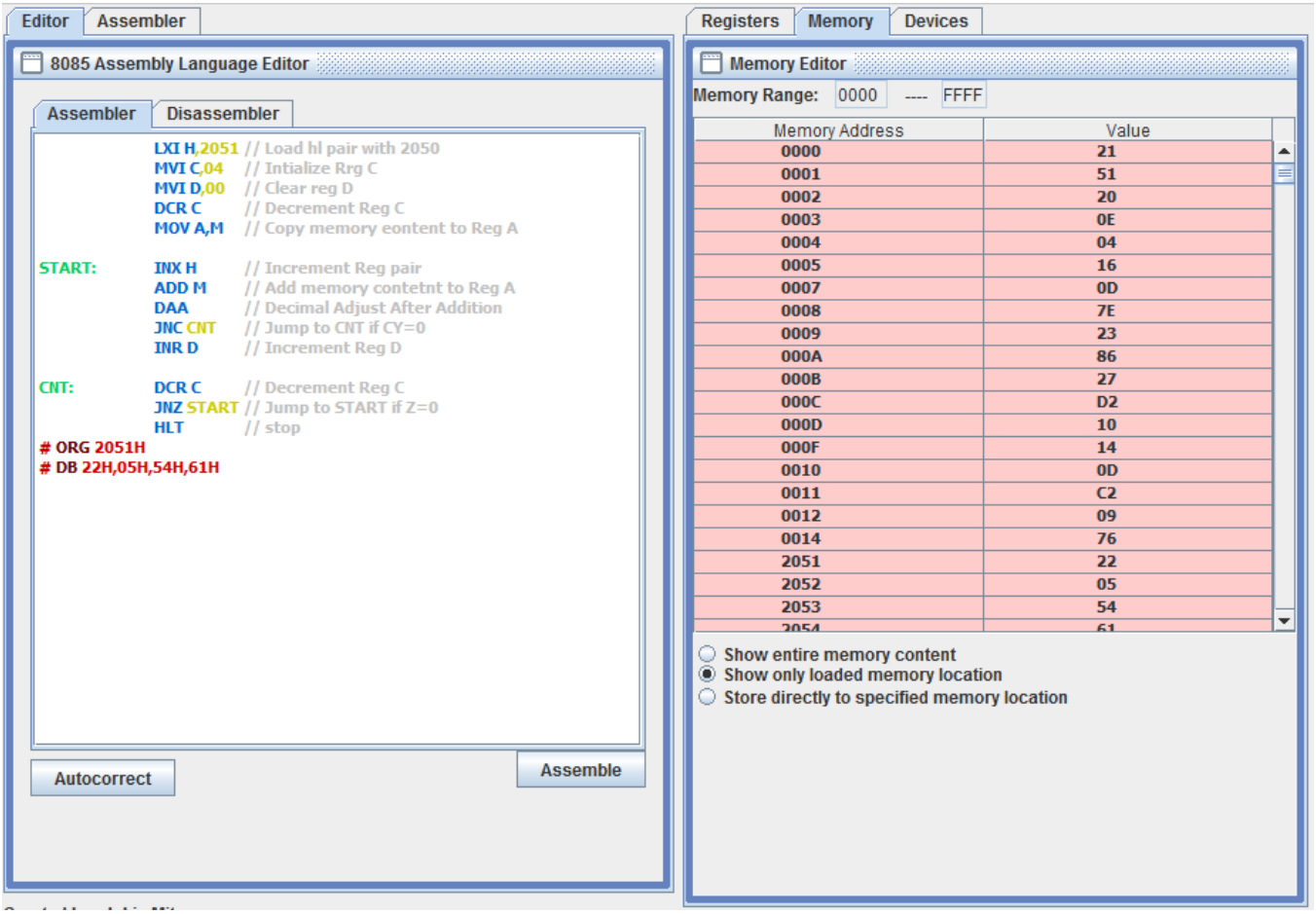


Fig7.3

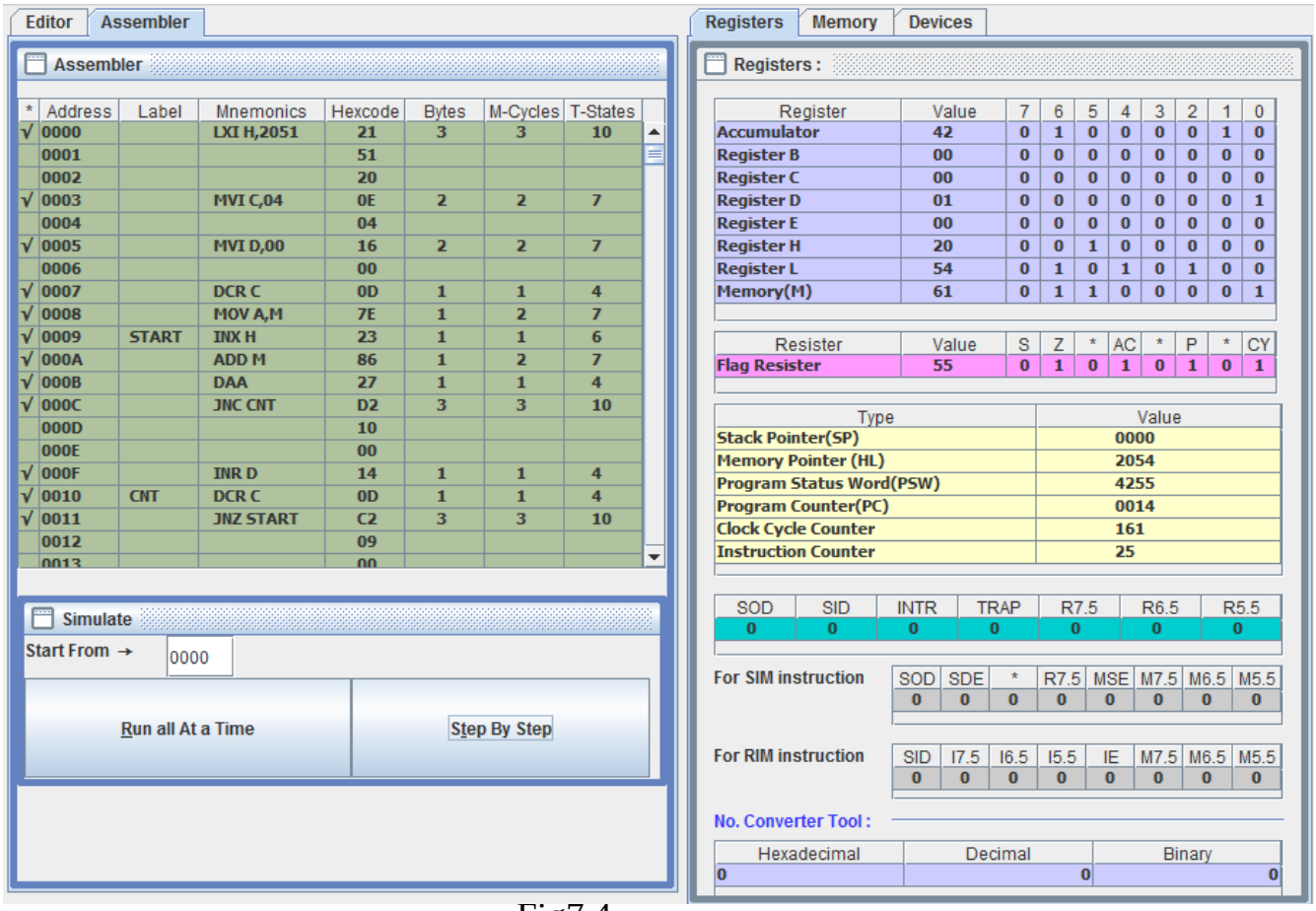

Editor
Assembler
Assembler
Simulate
Start From →
0000
Run all At a Time
Step By Step
Registers
Memory
Devices
Registers :
No. Converter Tool :

Fig7.4

Practical – 8

A. Write a program for 8085to reverse the block of 8-bit values using stack. The length of the block is given in memory location 2050H and block starts from 2051H.

LABEL	OPCODE	OPERAND	COMMENT
	LXI	H,2050	// load hl pair
	MOV	A,M	// copy mwmory content to Reg A
	MOV	D,A	// Copy Reg A to Reg D
	MOV	E,D	// Copy Reg D to Reg E
	RRC		// Rotate Accumulator right
	JC	PRE	// Jump to PRE if CY=1
	JNC	POST	// jump to POST if CY=0
POST:	INX	H	// Increment Hl pair
	MOV	B,M	// COpy memory content to Reg B
	DCR	D	// Decrement Reg D
	INX	H	// Increment Hl pair
	MOV	C,M	// Copy memory content to Reg C
	DCR	D	// Decrement Reg D
	JMP	START	// Jump to start
PRE:	MVI	B,00	// Initialize Reg B to 00
	INR	E	// Increment Reg E
	INX	H	// Increment Hl pair
	MOV	C,M	// Copy memory content to Reg C
	DCR	D	// Decrement Reg D
START:	PUSH	B	// push Reg pair B to stack
	JZ	REVERSE	// Jump to REVERSE if Z=1
	INX	H	// Increment Reg pair HL
	MOV	B,M	// COpy memory content to Reg B
	INX	H	// increment hl pair
	MOV	C,M	// Copy memory content to Reg C
	DCR	D	// Decrement Reg D
	DCR	D	
		START	// Jump to START
REVERSE:	LXI	H,2051	
LOOP:	POP	B	// Pop stack content to Reg pair BC
	MOV	M,C	// Copy Reg C to memory
	INX	H	// Increment Reg pair HL

	MOV	M,B	// Copy Reg B to Memory
	INX	H	// Increment Reg pair HL
	DCR	E	// Decrement Reg R
	DCR	E	
	JNZ	LOOP	// Jump to LOOP if Z=0
	HLT		// stop
# ORG 2050H			
# DB 05H,01H,02H,03H,04H,05H			

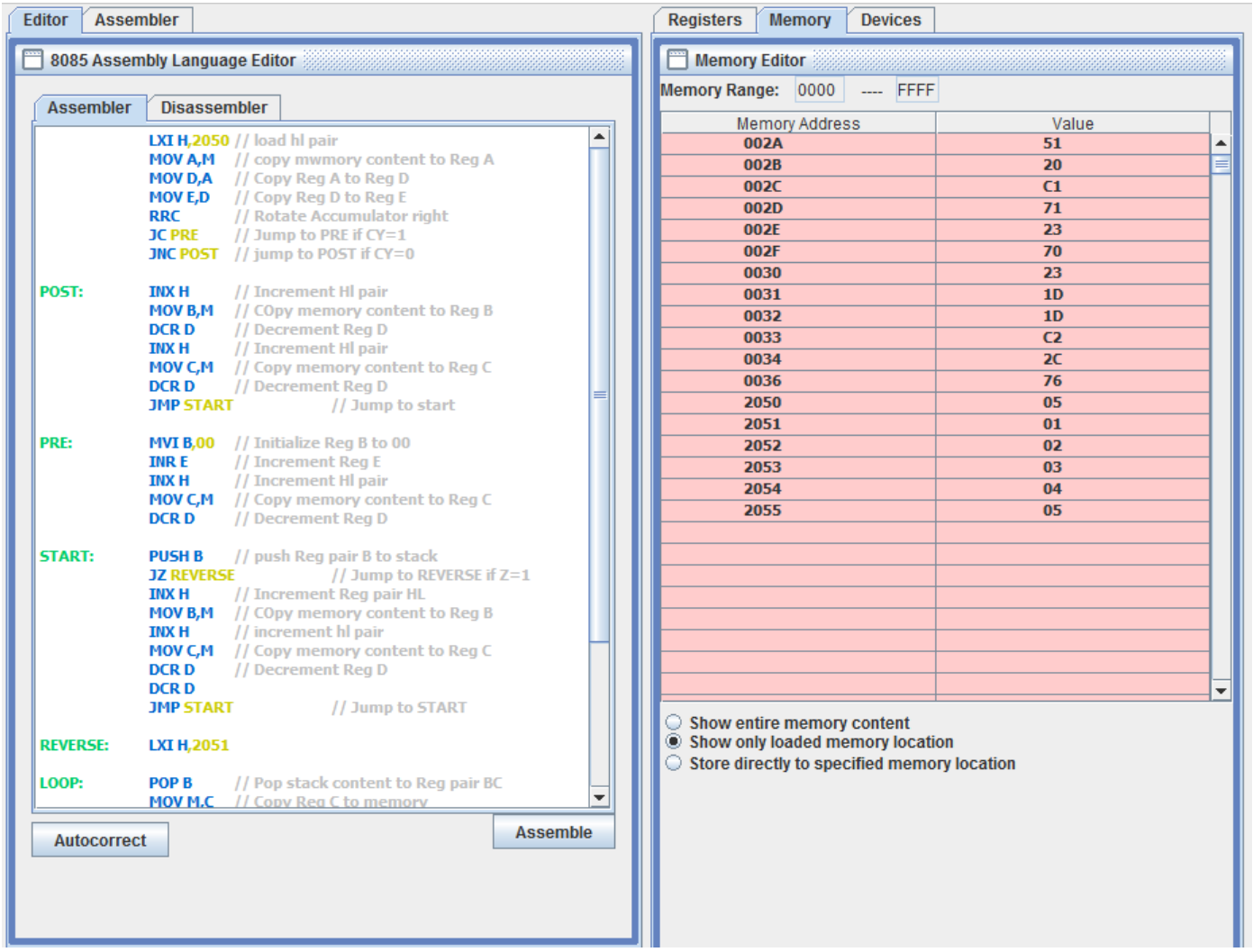
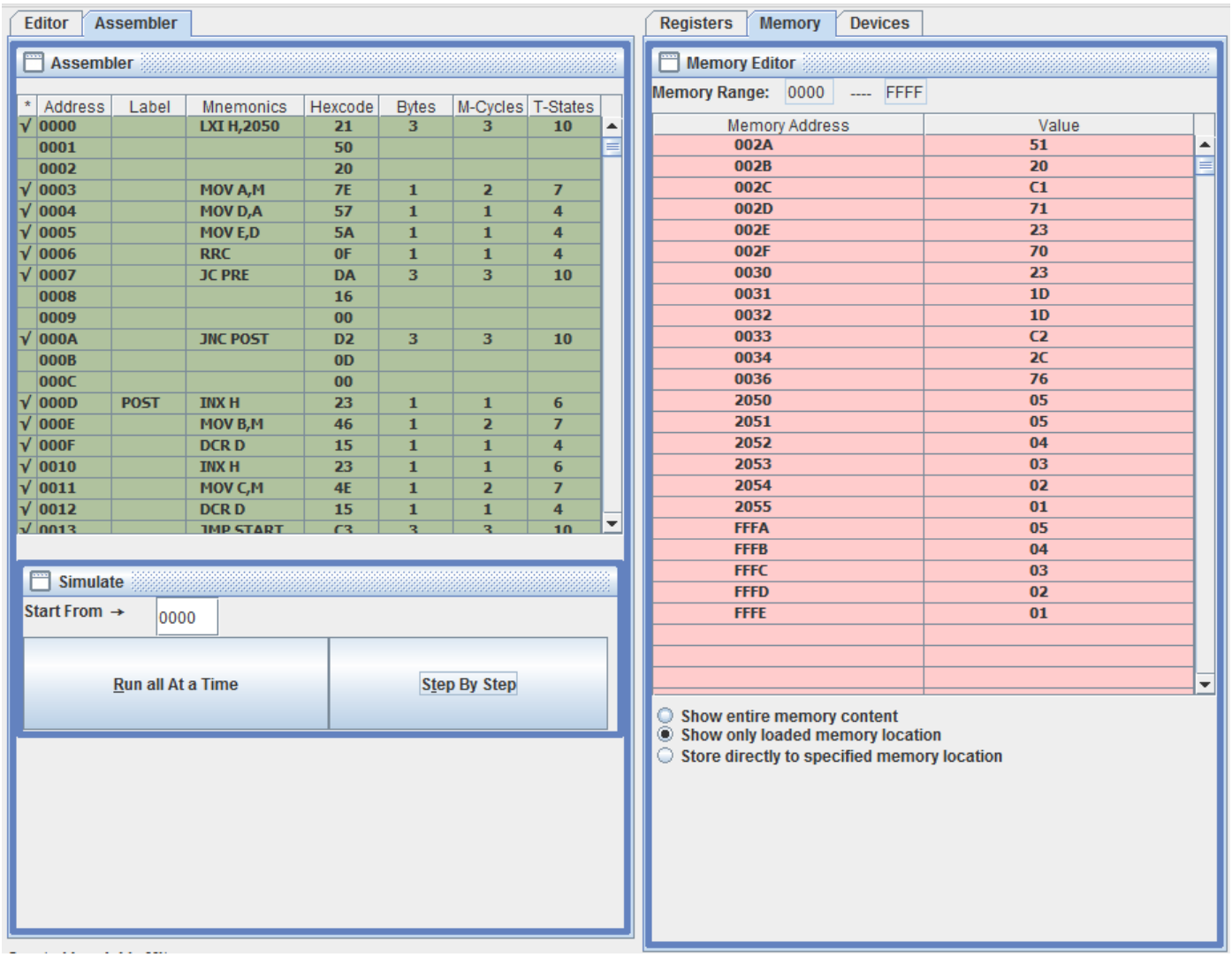


Fig8.1



*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		LXI H,2050	21	3	3	10
	0001			50			
	0002			20			
√	0003		MOV A,M	7E	1	2	7
√	0004		MOV D,A	57	1	1	4
√	0005		MOV E,D	5A	1	1	4
√	0006		RRC	0F	1	1	4
√	0007		JC PRE	DA	3	3	10
	0008			16			
	0009			00			
√	000A		JNC POST	D2	3	3	10
	000B			0D			
	000C			00			
√	000D	POST	INX H	23	1	1	6
√	000E		MOV B,M	46	1	2	7
√	000F		DCR D	15	1	1	4
√	0010		INX H	23	1	1	6
√	0011		MOV C,M	4E	1	2	7
√	0012		DCR D	15	1	1	4
√	0013		JMP START	C3	3	3	10

Memory Address	Value
002A	51
002B	20
002C	C1
002D	71
002E	23
002F	70
0030	23
0031	1D
0032	1D
0033	C2
0034	2C
0036	76
2050	05
2051	05
2052	04
2053	03
2054	02
2055	01
FFFA	05
FFFB	04
FFFC	03
FFFD	02
FFFE	01

Fig8.2

B. Write a Program to compare two 16-bit numbers stored in memory Using 8085.

LABEL	OPCODE	OPERAND	COMMENT
	LXI	H,2052	// load hl pair, second no
	LXI	D,2050	// load de pair, first no
	LDAX	D	// load higher 8 bits of first no
	CMP	M	// compare memory content with Reg A
	JC	LESS	// Jump to LESS if CY=1
	JNZ	BIG	// jump to BIG if Z=0
	INX	H	// increment hl reg pair
	INX	D	// increment de reg pair
	LDAX	D	// load lower 8 bits of first no
	CMP	M	// compare memory content with Reg A
	JC	LESS	
	JNZ	BIG	
	MVI	A,01	// Initialize Reg A with 01
	JMP	CNT	// jump to CNT
LESS:	MVI	A,00	
	JMP	CNT	
BIG:	MVI	A,02	
CNT:	HLT		// stop
// 02 => no1 > no2			
// 01 => no1 = no2			
// 00 => no1 < no2			
# ORG 2050H			
# DB 23H,51H,23H,51H			

Editor | Assembler

8085 Assembly Language Editor

Assembler | Disassembler

```
                LXI H,2052 // load hl pair, second no
                LXI D,2050 // load de pair, first no
                LDAX D     // load higher 8 bits of first no
                CMP M      // compare memory content with Reg A
                JC LESS    // Jump to LESS if CY=1
                JNZ BIG    // jump to BIG if Z=0
                INX H      // increment hl reg pair
                INX D      // increment de reg pair
                LDAX D     // load lower 8 bits of first no
                CMP M      // compare memory coontent with Reg A
                JC LESS
                JNZ BIG
                MVI A,01   // Initialize Reg A with 01
                JMP CNT    // jump to CNT

LESS:           MVI A,00
                JMP CNT

BIG:            MVI A,02

CNT:            HLT        // stop
// 02 = no1 > no2
// 01 = no1 = no2
// 00 = no1 < no2
# ORG 2050H
# DB 23H,51H,23H,51H
```

Autocorrect | Assemble

Registers | Memory | Devices

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
0015	C2
0016	22
0018	3E
0019	01
001A	C3
001B	24
001D	3E
001F	C3
0020	24
0022	3E
0023	02
0024	76
2050	23
2051	51
2052	23
2053	51

- ( ) Show entire memory content
- (•) Show only loaded memory location
- ( ) Store directly to specified memory location

Editor | Assembler

Assembler

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	0000		LXI H,2052	21	3	3	10
	0001			52			
	0002			20			
√	0003		LXI D,2050	11	3	3	10
	0004			50			
	0005			20			
√	0006		LDAX D	1A	1	2	7
√	0007		CMP M	BE	1	2	7
√	0008		JC LESS	DA	3	3	10
	0009			1D			
	000A			00			
√	000B		JNZ BIG	C2	3	3	10
	000C			22			
	000D			00			
√	000E		INX H	23	1	1	6
√	000F		INX D	13	1	1	6
√	0010		LDAX D	1A	1	2	7
√	0011		CMP M	BE	1	2	7
√	0012		JC LESS	DA	3	3	10
	0013			1D			

Simulate

Start From → 0000

Run all At a Time | Step By Step

Registers | Memory | Devices

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	01	0	0	0	0	0	0	0	1
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	20	0	0	1	0	0	0	0	0
Register E	51	0	1	0	1	0	0	0	1
Register H	20	0	0	1	0	0	0	0	0
Register L	53	0	1	0	1	0	0	1	1
Memory(M)	51	0	1	0	1	0	0	0	1

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	54	0	1	0	1	0	1	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	2053
Program Status Word(PSW)	0154
Program Counter(PC)	0024
Clock Cycle Counter	110
Instruction Counter	15

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

For RIM instruction

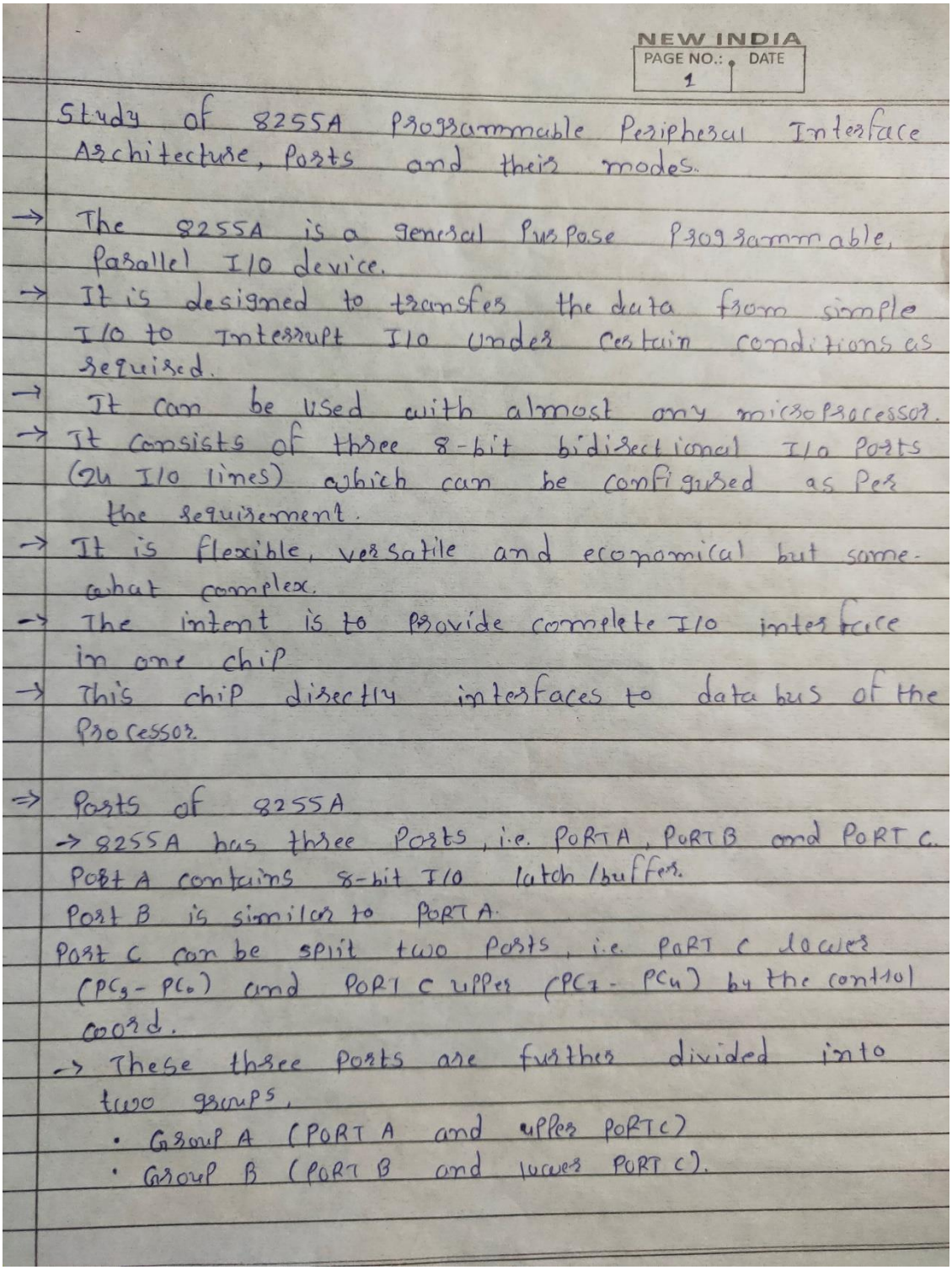
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

No. Converter Tool :

Hexadecimal	Decimal	Binary
0	0	0

Practical – 9

Aim: Study of 8255A Programmable Peripheral Interface Architecture, ports and their modes

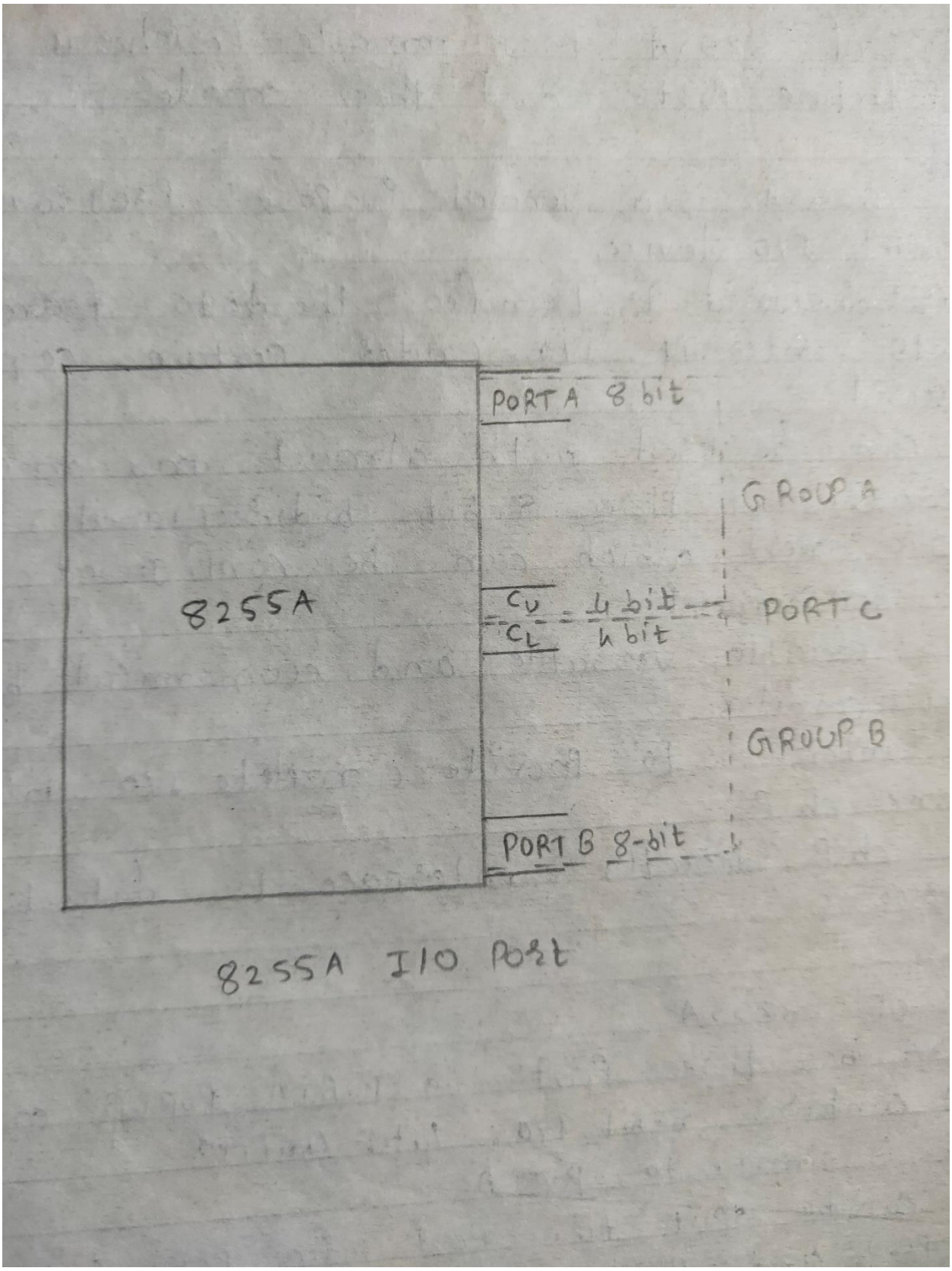


NEW INDIA
PAGE NO.: 1 DATE

Study of 8255A Programmable Peripheral Interface Architecture, Ports and their modes.

→ The 8255A is a General Purpose Programmable, Parallel I/O device.
→ It is designed to transfer the data from simple I/O to Interrupt I/O under certain conditions as required.
→ It can be used with almost any microprocessor.
→ It consists of three 8-bit bidirectional I/O ports (24 I/O lines) which can be configured as per the requirement.
→ It is flexible, versatile and economical but somewhat complex.
→ The intent is to provide complete I/O interface in one chip
→ This chip directly interfaces to data bus of the processor.

⇒ Ports of 8255A
→ 8255A has three ports, i.e. PORT A, PORT B and PORT C.
Port A contains 8-bit I/O latch/buffer.
Port B is similar to PORT A.
Port C can be split two ports, i.e. PORT C lower (PC_3 - PC_0) and PORT C upper (PC_7 - PC_4) by the control word.
→ These three ports are further divided into two groups,
- Group A (PORT A and upper PORT C)
- Group B (PORT B and lower PORT C).



8255A I/O Port

NEW INDIA
PAGE NO.: 2 DATE

→ 8255A Architecture: Control logic

RD (READ): This is an active low signal, that enables Read operation. when signal is low MPU reads data from selected I/O Port of 8255A

WR (WRITE): This is an active low signal, that enables write operation. when signal is low MPU writes data into selected I/O Port or control register.

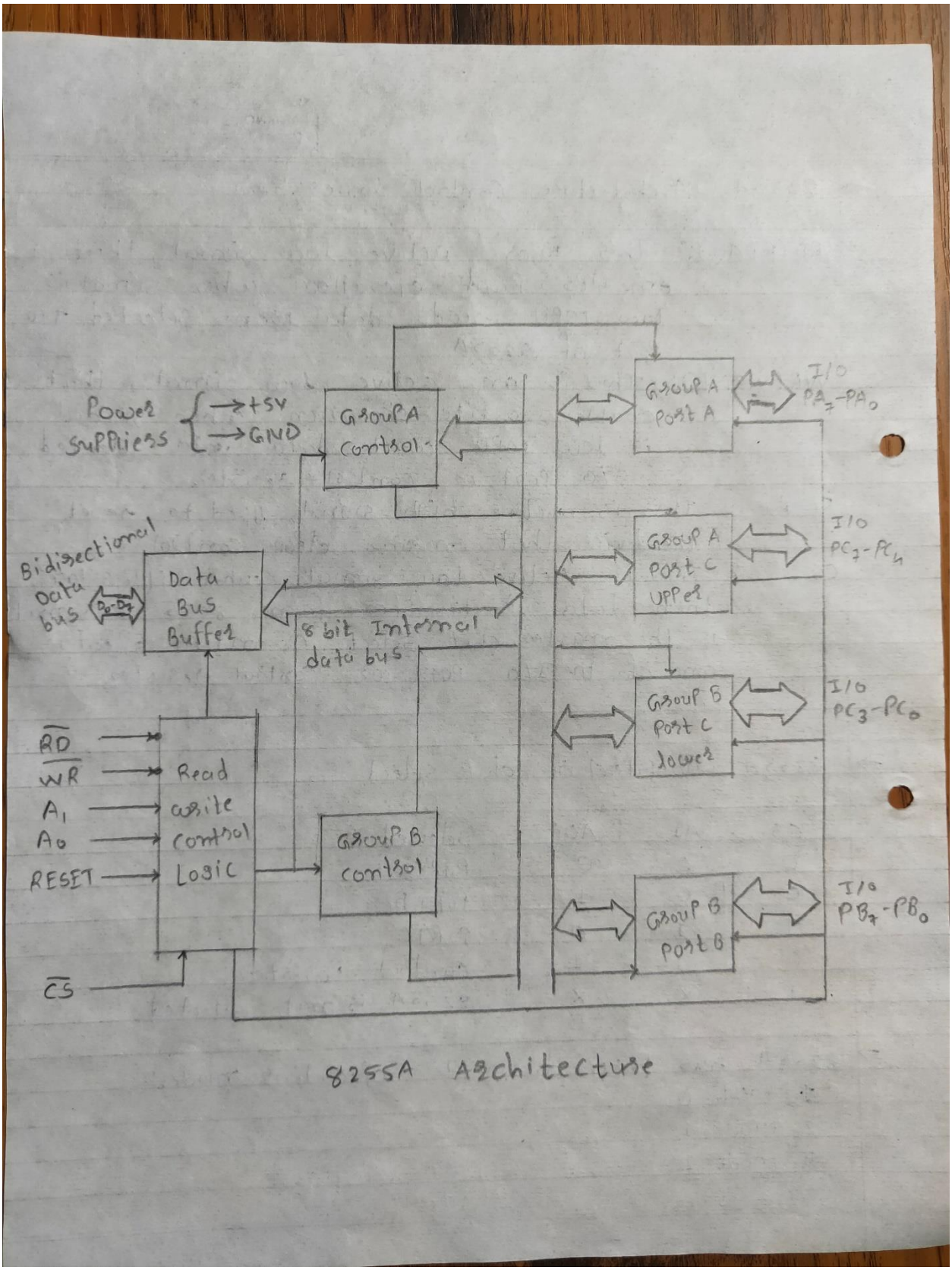
RESET : This is active high signal, used to reset the device. That means clear control.

CS : This is Active Low signal when it is low, then data is transfer from 8085 CS signal is the master chip select A_0 and A_1 specify one of the I/O ports or control register.

→ 8255A Architecture: chip select

CS	A1	A0	selected
0	0	0	PORTA
0	0	1	PORT B
0	1	0	PORTC
0	1	1	control Register
1	X	X	8255A is not selected

→ 8255A has three different operating modes:
1. mode 0
2. mode 1
3. mode 2



8255A Architecture

NEW INDIA
PAGE NO.: 3 DATE

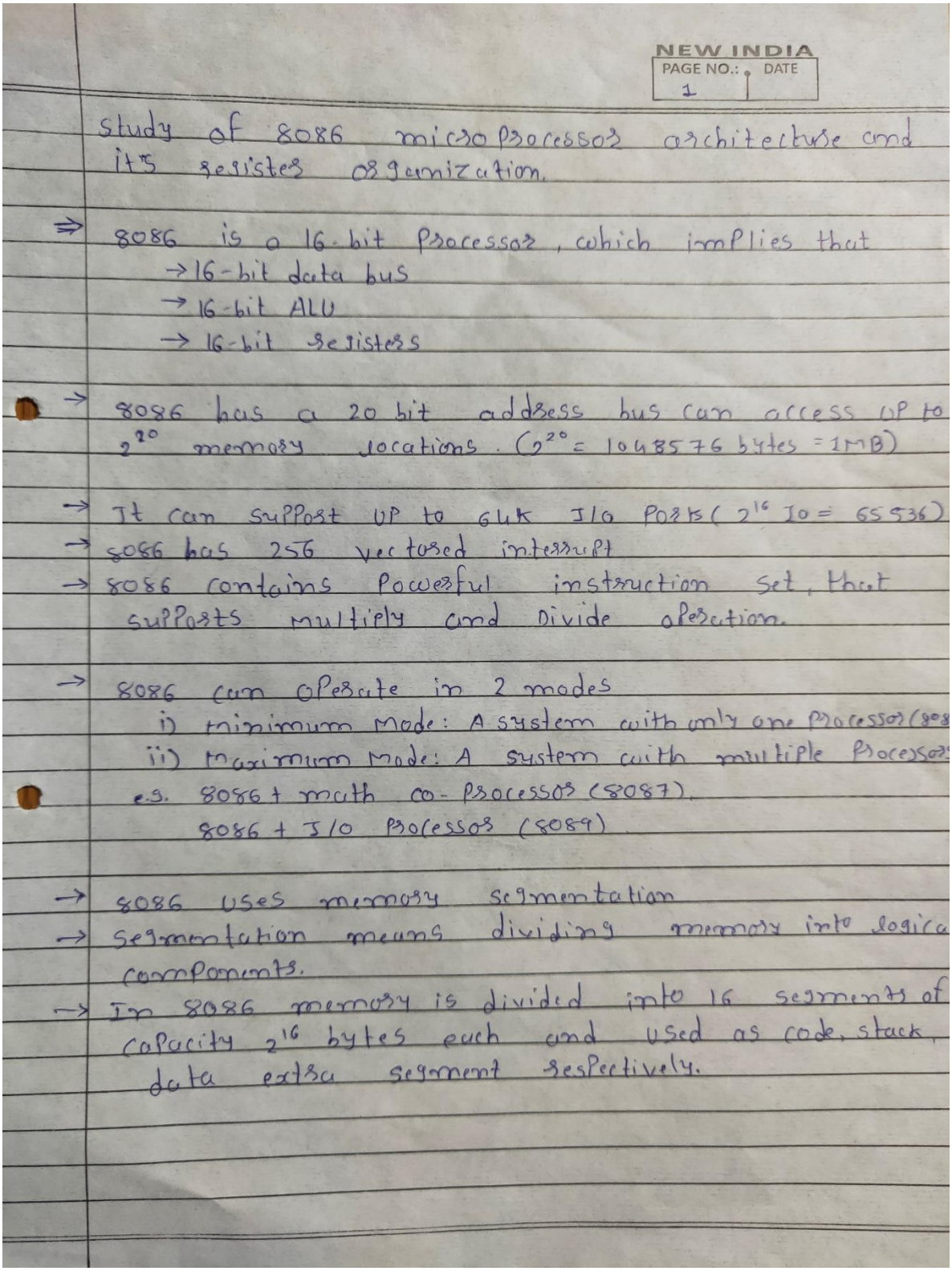
⇒ Mode 0
→ simple I/O for Port A, B and C
→ In this mode, Port A and B is used as two 8-bit Ports and Port C as two 4-bit Ports.
→ Each Port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched
→ Ports do not have handshake or interrupt capability.

⇒ Mode 1: Input or output with Handshake
→ Handshake signal are exchanged between MPU and Peripheral prior to data transfer.
→ In this mode, Port A and B is used as 8-bit I/O Ports.
→ mode 1 is a handshake mode whereby Ports A and/or B use bits from Port C as handshake signals.
→ In the handshake mode, two types of I/O data transfer can be implemented: status check and interrupt.

⇒ Mode 2:
→ In this mode, Port A can be configured as the bidirectional Port and Port B either in Mode 0 or mode 1.
→ Port A uses five signals from port C as handshake signals for data transfer.
→ The remaining three signals from Port C can be used either as simple I/O or as handshake for Port B.

Practical – 10

Aim: Study of 8086 microprocessor architecture and its register organization.



NEW INDIA
PAGE NO.: 1 DATE

Study of 8086 microprocessor architecture and its registers organization.

⇒ 8086 is a 16-bit processor, which implies that
→ 16-bit data bus
→ 16-bit ALU
→ 16-bit registers

→ 8086 has a 20 bit address bus can access upto 2^{20} memory locations. (2^{20} = 1048576 bytes = 1MB)

→ It can support upto 64K I/O ports (2^{16} IO = 65536)
→ 8086 has 256 vectored interrupt
→ 8086 contains powerful instruction set, that supports multiply and divide operation.

→ 8086 can operate in 2 modes
i) minimum mode: A system with only one processor (8086
ii) maximum mode: A system with multiple processors
e.g. 8086 + math co-processor (8087).
8086 + I/O processor (8089)

→ 8086 uses memory segmentation
→ Segmentation means dividing memory into logical components.
→ In 8086 memory is divided into 16 segments of capacity 2^{16} bytes each and used as code, stack, data extra segment respectively.

NEW INDIA
PAGE NO.: 2 DATE

⇒ 8086 Architecture:

→ In 8086 CPU is divided into two independent functional parts BIU and EU two units speeds up the processing.

⇒ components of BIU (Bus Interface Unit):

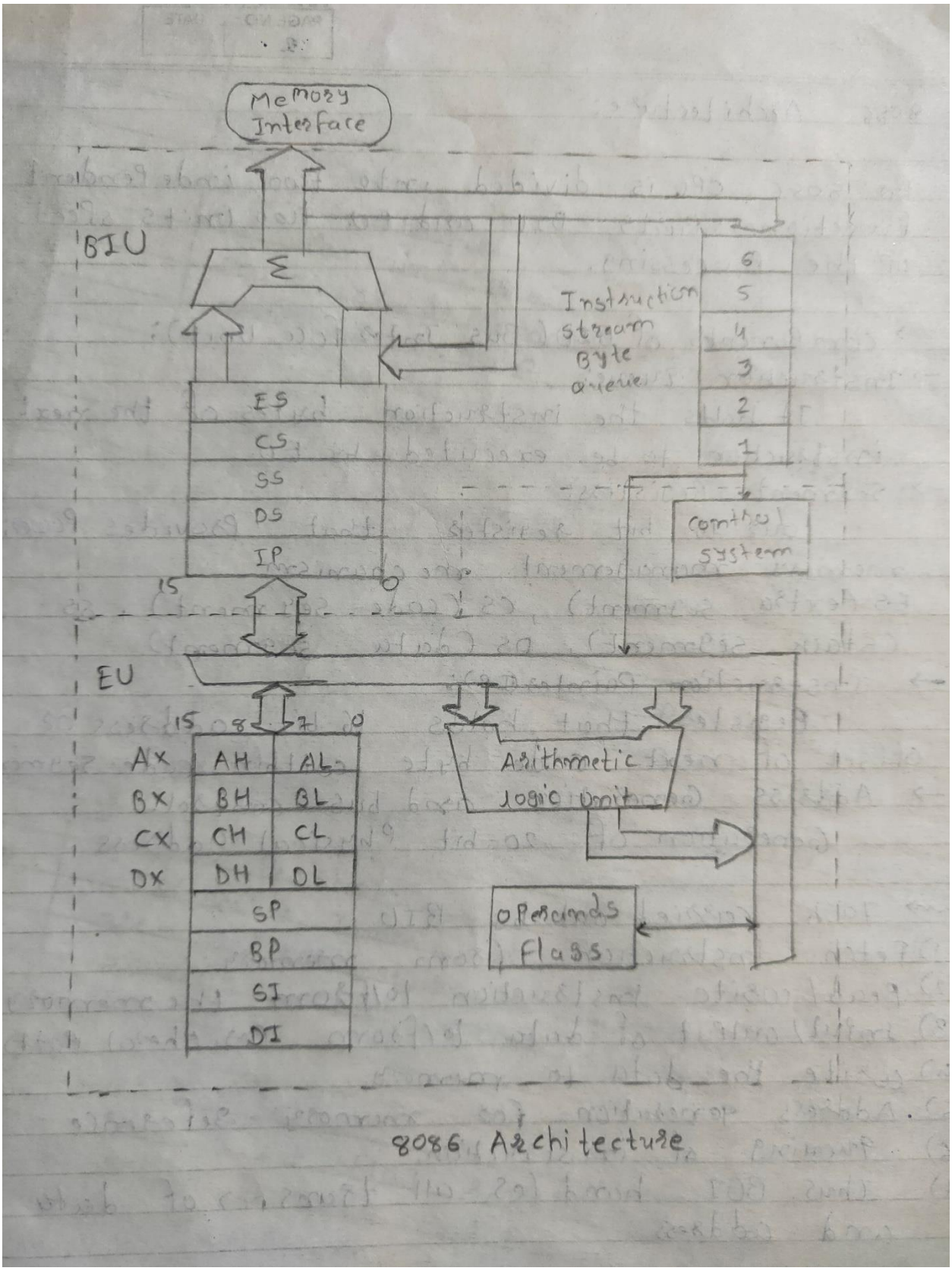
→ Instruction queue:
It holds the instruction bytes of the next instruction to be executed by EU.

→ Segment Registers:
four 16 bit registers that provides powerful memory management mechanism
ES (extra segment), CS (code segment), SS (stack segment), DS (data segment).

→ Instruction Pointer (IP):
Register that holds 16-bit address or offset of next code byte within code segment

→ Address Generation and bus control:
Generation of 20-bit Physical address

⇝ Task carried out by BIU
1) Fetch instruction from memory.
2) Read/write instruction to/from the memory.
3) Input/output of data to/from Peripheral ports
4) write the data to memory.
5) Address generation for memory reference.
6) queuing of instruction.
7) Thus, BOI handles all transfer of data and address.



8086 Architecture

NEW INDIA
PAGE NO.: 3 DATE

⇒ Components of EU:

→ ALU (Arithmetic logic Unit):
Contains 16-bit ALU, that performs add, subtract increment, decrement, compliement, shift binary numbers, AND, OR, XOR etc.

→ CU (control unit): Directs internal operation

→ General purpose Registers (GPR):
EU has 4 general purpose 16-bit registers AX, BX, CX, DX each register is the combination of two 8-bit registers.

→ Pointer registers:
Stack Pointer: It always points to the top of the stack, used for sequential access of stack segment.
Base Pointer: is used when we need to pass Parameter through stack.

→ Index Registers:
SI (source Index) and DI (Destination Index): are used for string related operation and for moving block of memory from one location to the other

→ Flag Register:
The 16-bit flag register of 8086 contains 9 active flags (6 conditional & 3 control flags) other 7 flags are undefined.

1. CF - Carry Flag
2. PF - Parity Flag
3. AF - Auxiliary Flag
4. ZF - Zero Flag
5. SF - sign flag
6. TF - Trap Flag
7. IF - Interrupt Flag
8. DF - Direction Flag
9. OF - overflow flag

PAGE NO.: 4 DATE

⇒ Task carried out by EU:
→ Decodes the instruction
→ It executes decoded instructions
→ Tells BIU from where to fetch the instruction
→ EU takes care of performing operation on the data
→ EU is also known as execution heart of the processor.

⇒ Types of segment registers are as follows:
1. Code segment (CS): executable program is stored in CS

2. Data segment (DS): The DS contains most data used by program. Data are accessed in the Data segment by an offset address or the content of other register that holds the offset address.

3. Stack segment (SS): SS defined the to area of memory used for the stack.

4. Extra segment (ES): ES is additional data segment.

⇒ Flag Register:

→ The 16-bit flag register of 8086, 16bit flag is a

Flag register

↓ control flag (TF, IF, DF)

status Flag (OF, ZF, SF, AF, PF, CF)

NEW INDIA
PAGE NO.: 5 DATE

→ 9 active flags (6 conditional & 3 control flag)
→ 7 flags are undefined.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

U - Undefined
OF - overflow flag
DF - direction flag
IF - Interrupt flag
TF - Trap flag
CF - Carry flag
PF - Parity flag
AF - Auxiliary
ZF - zero flag
SF - sign flag

→ Interrupt Flag:
IF=0 ; Disable maskable interrupt
IF=1 ; Enable maskable interrupt

→ Trap Flag: is used for single step control.
TF = 0; whole program will be executed.
TF = 1; program will run in single step mode

→ overflow flag:
It occurs when signed numbers are added as subtracted.
OF=0; result has not exceeded the capacity of machine.
OF=1; result has exceeded the capacity of machine

→ Direction Flag:
DF=0 ; string bytes are accessed from lower to higher memory address.
DF=1 ; string bytes are accessed from higher to lower memory address.

NEW INDIA
PAGE NO.: 6 DATE

→ Carry Flag (CF): This flag indicates an overflow condition for unsigned integer arithmetic

→ Auxiliary Flag (AF): If an operation performed in ALU generates a carry / borrow from lower nibble (i.e. D0 - D3) to upper nibble (i.e. D4 - D7) the AF flag is set.

→ Parity Flag (PF): This flag is used to indicate the Parity of result. if lower-order 8-bits of the result contains even number of 1's, the Parity flag is set and for odd number of 1's the Parity flag is reset.

→ zero flag (ZF): It is set; if the result of arithmetic or logical operation is zero else it is reset.

→ sign Flag (SF): In sign magnitude format the sign of number is indicated by MSB bit. if the result of operation is negative, sign flag is set.