Page No.

Expt. No.

Architecture of 8085 Microprocessor

- 8085 is an 8-bit general purpose Microprocessor It consist of following functional units:-

1) Arithmetic Logical Unit (ALU) : It is used to perform mathematical operation like: Addition, Subtraction, division & multiplication etc.

2) Flag Register: It is an 8-bit register that stores either 0 or 1 depending upon which value is stored in accumulator.

3) Accumulator: It is used to perform I/O, Arithmetic & logical operation It is connected to ALU & internal data Bus.

4) General Purpose Registers: There are 6 general purpose register which can hold 8-bit values. These are B, C, D, E, H, L & In pair they work as 16-bit register e.g BC, DE, HL.

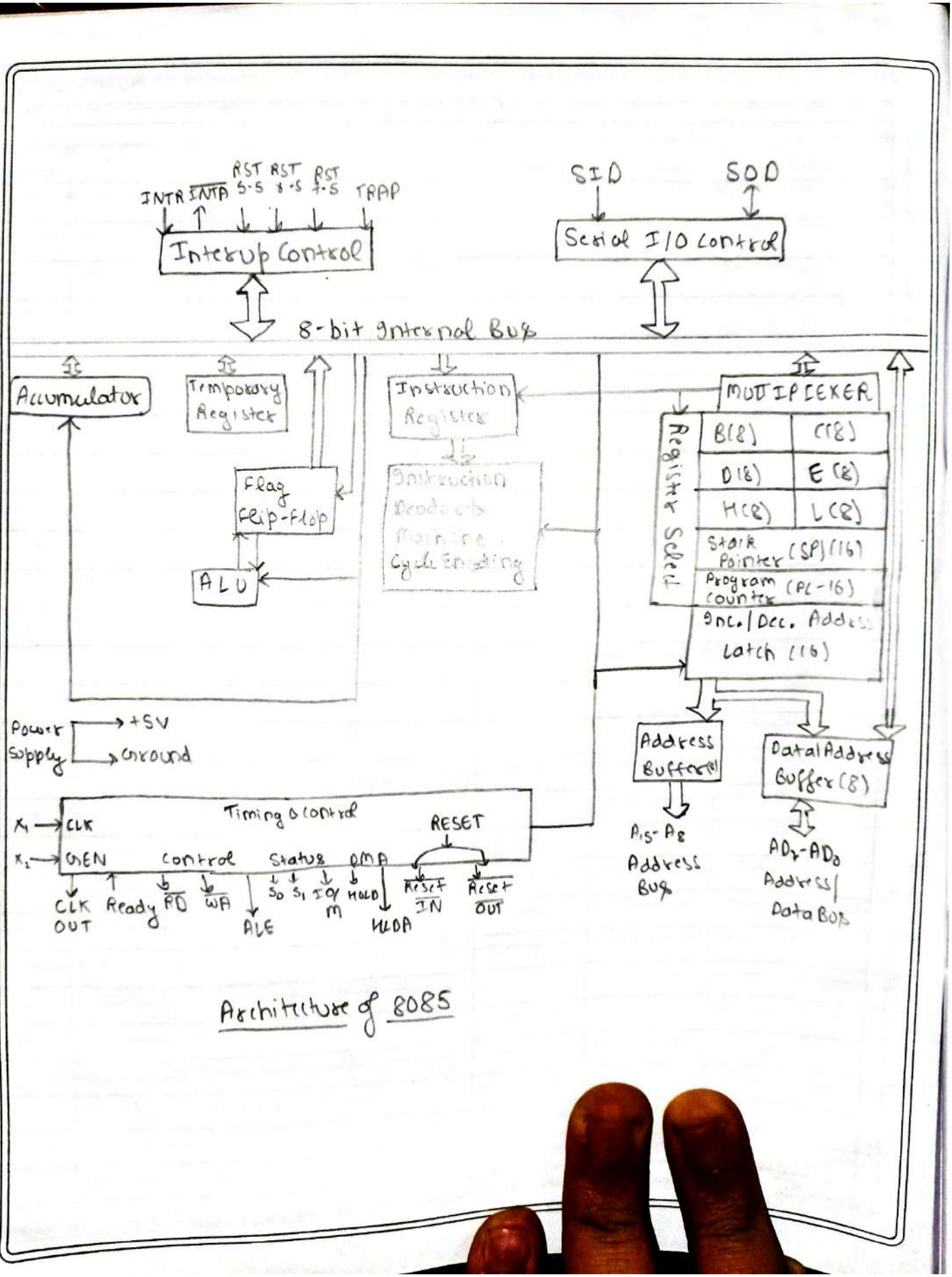
5) Program Counter: It holds the address value of the memory to the next instruction that is to be executed It is a 16-bit register.

6) Stack Pointer: It work like stack, in stack the content of register is stored that is later used in program. It is a 16-bit special register

7) Temporary Register: It is 8 bit register that hold data values

NOOTAN

Teacher's Signature



Architecture of 8085

during Arithmetic & logical operations

8) Instruction register & decoder :- 8bit register holds instruction code that is being decoded.

9) Timing & Control Unit :- It controls flow of data from CPU to other devices & also used to control operation performed by microprocess & devices connected to it.

10) Interrupt control :- When a microprocessor is executing a main program & if suddenly can interrupt occurs, microprocessor shift control from main program to process incoming request. There are 5 interrupt in 8085 → INTR, TRAP, RST 7.5 RST 6.5, RST 5.5.

11) Address/Data bus :- It is bidirection & carries the data which is to be stored. Address bus is unidirection & carries location where data is to be stored.

12) Serial I/O control :- It control serial data comm. by using Serial input & Serial output data.

Page No.

Expt. No.

Instruction Set of 8085

1) Data Transfer Instruction :-

a) **MOV** :- Copy content of source register to designated register
e.g MOV B, C

b) **MVI** :- 8-bit data stored in designated register. e.g MVI A, 20H

c) **LDA** :- content of 16-bit address copied to Accumulator.
e.g LDA 2000H

d) **LDAX** :- Content of designated register pair point to memory location & copy content of that memory to Accumulator. e.g :- LDAX B.

e) **LXI** :- Load 16 bit data in register pair. e.g :- LXI H, 2500H

f) **LHLD** :- copy content of specified 16-bit address in register L & content of next memory to register H. e.g LHLD 2500H

g) **STA** :- Copy content of accumulator to specified 16 bit address
e.g STA 2050H

h) **STAX** :- Copy content of accumulator to address specified by register pair e.g :- STAX B.

i) **SHLD** :- content of L register store into specified 16 bit address & content of H register stored in next 16 bit address.
e.g SHLD 50 H

j) **XCHG** :- Content of H & L register exchange with D & E register respectively. e.g XCHG

k) **SPHL** :- Load content of H & L register in Stack Pointer Register
e.g SPHL

l) **XTHL** :- Content of location (16-bit address) specified by stack pointer copied to L Register & (SP+1) address content copied to H Register e.g :- XTHL

NOOTAN

Teacher's Signature

Expt. No. Page No.

m) PUSH:- Content of register pair specified copied into stack pointer. As:- SP is decremented & high order store in it & decremented again & lower order data stored in that location. e.g PUSH B.

n) POP:- Content of SP location store to low order Register & Increment by 1 to store the content to high order Register & SP increment again. e.g POP H.

(o) OUT:- Content of accumulator copied into specified I/O Port e.g OUT F8H.

(p) IN:- Content of I/O Port specified are read & reloaded to Accumulator e.g → IN 8CH.

- ARITHMATIC INSTRUCTION:-

a) ADD:- Content of operand are added to content of accumulator & result store in Accumulator. e.g ADD B.

b) ADC:- Content of operand & carry added to Accumulator & result store in Accumulator e.g → ADC B.

(c) ADI:- 8 bit data added to content of Accumulator & result store in Accumulator e.g → ADI 08H

d) ACI:- 8 bit data & content of carry Added to Accumulator & result stored in Accumulator e.g → ACI 45H

e) DAD:- 16 bit Content of specified register pair added to HL pair & result stored in HL pair. e.g → DAD H

f) SUB:- Operand content subtracted from accumulator & result store in Accumulator. e.g SUB B.

g) SBB:- Operand & Borrow subtracted from accumulator & result

Expt. No. Page No.

stored in Accumulator. e.g → SBB B.

h) SUI :- 8bit data subtract from accumulator & result in accumulator
e.g SUI 10H

i) SBI :- 8bit data & Borrow flag subtracted from content of accumulator & result stored in accumulator eg → SBI 10H.

j) INR :- Content of designate Register increment by 1 e.g INR B.

k) INX :- Content of designated Register Pair increment by 1 e.g INX B.

l) DCR :- Content of designated Register decrement by 1 e.g DCR B

m) DCX :- Content of designated Register pair decrement by 1 e.g DCX B

n) DAA :- Content of accumulator change from binary to 4 bit BCD.
If value of low-order 4 bits is >9 or if AC=1 then add 6 to low order 4 bits
If value of high-order 4 bits is >9 & C=1, add 6 to high order 4 bits e.g DAA.

=> LOGICAL INSTRUCTION:

a) CMP :- Content of operand compare with Accumulator.
A < operand (CY=1), A = Operand (Z=1), A > Operand (CY=0, Z=0)
e.g CMP B.

b) ANA :- Content of Accumulator Logically ANDed with operand & result store in Accumulator e.g ANA B

c) CPI :- 8bit data compare with Accumulator e.g CPI 10H

d) ANI :- Content of Accumulator ANDed with 8 bit data eg → ANI 10H

e) XRA :- Content of Accumulator Exclusively ORed with operand & result stored in Accumulator e.g → XRA B.

f) XRI :- Content of Accumulator Exclusive ORed with 8bit data & result store in Accumulator e.g → XRI 80H

NOOTAN

Teacher's Signature

Expt. No. Page No.

g) ORA:- content of Accumulator ORed with operand e.g. ORA B
h) ORI:- content of Accumulator ORed with 8 bit data e.g. ORI 10H
i) RLC:- Each binary bit of Accumulator rotated left by 1 Position without carry e.g RLC
j) RRC:- Each binary bit of Accumulator rotated Right by 1 Position with carry e.g RRC
k) RAL:- Each binary bit of Accumulator rotated left by 1 Position with carry e.g RAL
l) RAR:- Binary bit of Accumulator rotated right by 1 Position with carry e.g RAR
m) CMA:- Accumulator content complemented e.g CMA
n) CMC:- Carry flag is complemented e.g CMC
o) STC:- Carry flag set to 1 e.g STC

BRANCHING INSTRUCTION:

a) JMP:- Program Sequence is transfer to memory location specify by 16-bit address given in operand. e.g JMP 2030H.

Conditionally Jump:-
- JC → Jump on carry
- JNC → Jump on no carry
- JP → Jump on positive
- JM → Jump on minus
- JZ → Jump on zero
- JNZ → Jump on No zero
- JPE → Jump on Parity even (P=1)
- JPO → Jump on Parity odd. (P=0)

b) CALL:- Program Sequence transfer to memory location specify by 16-bit address given in operand. e.g CALL 2030H

Conditionally CALL:-
- CC → CALL on carry (CY=1)
- CNC → CALL on No Carry (CY=0)

NOOTAN

Teacher's Signature

- CP → Call on +ve (S=0) • CM → Call on Minus (S=1) • CZ → Call on Zero (Z=1)
- CNZ → Call on No Zero (Z=0) • CPE → Call on Parity Even (P=1)
- CPO → Call on Parity odd (P=0)

c) RET:- Program sequence transfer from subroutine to calling program.
e.g → RET

conditional →

- RC → Return on Carry • RNC → Return on No Carry
- RZ → Return on Zero • RNZ → Return on No Zero
- RP → Return on Positive • RM → Return on Minus
- RPE → Return on Parity Even • RPO → Return on Parity odd.

d) PCHL:- Content of HL copied into PC. e.g PCHL.

e) RST:- It is 1 byte instruction used in conjuction with interrupts & insert using ext hardware

CONTROL INSTRUCTION

- NOP:- No operation is performed. e.g → NOP
- HLT:- CPU finish instruction & halt any further e.g HLT
- DI:- Interrupt enable is reset & all interrupt except TRAP disable e.g DI
- EI:- Interrupt enable reset & all interrupt enabled e.g → EI
- RIM:- Instruction to read interrupt status & read Serial data input bit e.g → RIM
- SIM:- It is used to implement 8085 interrupt & Serial data output e.g SIM.

Page No.

Expt. No. 1

Experiment - 1

Aim:- Write a program in 8085 to add two 8bit Number:-
i) Immediate addressing
ii) Direct addressing
iii) Indirect addressing

Algorithm:- 1) Start Microprocessor
2) Initialize carry as zero
3) First 8 bit into accumulator
4) copy content of accumulator in B
5) load second 8bit into accumulator
6) Add B to accumulator & check for carry
7) Jump if No carry
8) Increment carry if any
9) Store result in memory
10) Move carry to Accumulator & store it in next memory location
11) Stop Execution of Program.

Program code:-

- Immediate Addressing

```
jmp start
start: nop
MVI C,00H       ; Store carry
MVI A,07H       ; First Number
MVI B,05H       ; Second Number
Add B           ; Add content of Bin accumulator
```


Teacher's Signature

```
        JNC LOOP        ; Jump on LOOP if No carry
        INR C           ; Increment C if any carry
LOOP:   STA 2500H       ; Store result in 2500H
        MOV A, C        ; Move carry in Accumulator
        STA 2501H       ; Store carry in 2501H
        HLT             ; Terminate Program.
```

⇒ Direct Addressing

```
        jmp start
start:  nop
        MVI C, 00H      ; Store carry
        LDA 2500H       ; First No. to Accumulator
        MOV B, A        ; Move Accumulator to B
        LDA 2501H       ; Move Second Number to Accumulator
        ADD B           ; Add B to Accumulator
        JNC LOOP        ; Jump to LOOP if No carry
        INR C           ; Increment carry if any
LOOP:   STA 2502H       ; Store result in 2502H
        MOV A, C        ; Move carry to Accumulator
        STA 2503H       ; Store carry in 2503H
        HLT             ; Terminate The Program.
```

⇒ Indirect Addition:

```
        jmp start
start:  nop
        MVI C, 00H      ; Store carry
```

Expt. No. _____ Page No. _____

```
LXI H, 2500H    ; Store Address of 1st number in HL Pair
MOV B, M        ; Move first number from HL Pair location to B
INX H           ; Increment HL Pair
MOV A, M        ; Move Second number in Accumulator
ADD B           ; Add B to Accumulator
JNC LOOP        ; JUMP to LOOP if no carry
INR C           ; Increment C if carry
LOOP: STA 2502H ; Store Result in 2502H
      MOV A, C  ; Move carry in Accumulator
      STA 2503H ; Store carry in 2503H.
HLT             ; Terminate The Program.
```

RESULT:- Addition of two 8bit Number are done successfully & following results will appear:-

1) Immediate:- Input → A → 07H , B → 05H
OUTPUT → 2500H → 12H , 2501H → 00H (No carry)

2) Direct Address → Input → 2500H → 254 (FEH)
2501H → 255 (FFH)
OUTPUT → 2502H → 253 (FDH)
2503H → 01 (carry)

3) Indirect Addressing:- Input → 2500H: 09H , 2501H → 03H
OUTPUT → 2502H :- 12 2503H → 00.

NOOTAN

Teacher's Signature _____

Page No

Expt. No 2

Experiment-2

Aim :- Write a program for Multiplication & division of two 8-bit numbers stored in consecutive Memory location

=) FOR MULTIPLICATION

Algorithm :- 1) Start Microprocessor
2) Get first number in B
3) Get Second number in C
4) Initialize Accumulator as zero
5) Initialize carry as zero
6) Add B to Accumulator
7) Jump if no carry
8) Increment carry by 1 if any
9) Decrement C & repeat from step 6 till C = 0
10) Store multiplied value in accumulator to memory
11) Move carry to Accumulator & store in memory
12) Stop execution of program.

Code :-

```
jmp start
start: nop
LXI H, 2500H
MOV B, M        ; Move first number to register B
INX H           ; Increment HL Reg pair
MOV C, M        ; Move Second Number to Register C (count)
LOOP: ADD B     ; Add Content of Register B to Accumulator
      DCR C     ; Decrement C to reduce count.
```

NOOTAN

Teacher's Signature

```
JNZ LOOP        ; Jump to LOOP if NO zero
STA 2502H       ; Store Result in 2502H
HLT             ; Terminate the Program
```

Result:- Multiplication of 2 (8 bit Number) done Successfully:-

Input :→ 2500H → 10 (First No.)
2501H → 2 (Second No.)

OUTPUT → 2502H → 20.

FOR DIVISION:-

1) Start Microprocessor
2) Initialize Quotient as zero
3) load first 8bit data
4) Copy content of Accumulator to B
5) Load second 8 bit data
6) Compare both value
7) Jump if divisor greater that dividend
8) Subtract dividend value by divisor value
9) Increment Quotient
10) Jump to Step 7 till dividend become zero
11) Store result in accumulator & Store in Memory
12) Move remainder to accumulator & Store in Memory
13) Stop Program.

Expt. No. _____ Page No. _____

Code:-

```
jmp start
start: nop
LXI H, 2501H    ; Address of Dividend is store to HL Pair
MOV B, M        ; Move value of Dividend to B Register
MVI C, 00       ; Clear C for Quotient
DCX H           ; Decrement H to get Pointed to divisor location
MOV A, M        ; Move divisor to Accumulator
NEXT: CMP B     ; Compare Accumulator with Register B
      JC LOOP   ; Jump to LOOP if carry
      SUB B     ; Subract Accumulator from B
      INR C     ; Increment C
      JMP NEXT  ; Jump to Next

LOOP: STA 2503H ; Store remainder in 2503H
      MOV A, C  ; Move content of C to Accumulator
      STA 2502H ; Store Quotient in 2502H
```

Result:- Program for Division of two 8-bit no. written succesfully

Input → 2500H → 10 (Dividend)
2501H → 3 (Divisor)
OUTPUT → 2502H → 3 (Quotient)
2503H → 1 (Remainder)

NOOTAN

Teacher's Signature _____

Expt. No 3 Page No

Experiment-3

Aim: Write a program to sort a series of 10 data bytes in Ascending / Descending order.

Theory / Algorithm :-

FOR ASCENDING ORDER :-

1) Initialize HL pair as memory pointer
2) Get count into C register (for bubble sort (N-1))
3) Copy it in D-register.
4) Get first value in Accumulator
5) Compare it with next location
6) If out of order exchange content of Accumulator & Memory.
7) Decrement D-register count by 1
8) Repeat 5 & 7 till D = 0
9) Decrement C-register by 1
10) Repeat 3 to 9 till C become zero
11) Stop Execution

Code :-

```
jmp start
start: nop
LXI, 2500 H      ; Array length address
MOV C, M         ; Array length to C
DCR C            ; (Array length - 1) to C.
```

NOOTAN

Teacher's Signature _____

Expt. No_____ Page No_____

```
REPEAT: MOV D,C      ; Move content of C in D
        LXI H, 2501H ; Store address of first Element in HL

LOOP:   MOV A,M      ; Copy content of memory to Accumulator
        INX H        ; Increment HL Pair
        CMP M        ; Compare Accumulato with Memory
        JC SKIP      ; Jump to SKIP if carry
        MOV B,M      ; Copy from memory to B
        MOV M,A      ; Move Accumulator to memory
        DCX H        ; Decrement HL Pair
        MOV M,B      ; Move B to Memory
        INR H        ; Increment HL Pair

SKIP:   DCR D        ; Decrement D
        JNZ LOOP     ; Jump to LOOP if no zero
        DCR C        ; Decrement count
        JNZ REPEAT   ; Jump to REPEAT if no zero

        HLT          ; Close Program.
```

RESULT: Program for Ascending order Sorting written successfully

Input :		OUTPUT	
2500H → 10	2506H → 19	2500H → 1	2506H → 15
2501H → 8	2507H → 50	2502H → 2	2507H → 19
2502H → 4	2508H → 43	2503H → 4	2508H → 43
2503H → 6	2509H → 100	2504H → 6	2509H → 50
2504H → 2	250AH → 15	2505H → 8	250AH → 100
2505H → 1			

NOOTAN

Teacher's Signature _____

Expt. No. _____ Page No. _____

For descending ORDER

- **Algorithm:-** i) Initialize HL pair as memory pointer
ii) Get count in C (N-1) for bubble sort)
iii) Copy it in D register
iv) Get first value in Accumulator
v) Compare with next location
vi) If out of order Exchange Accumulator & Memory
vii) Decrement D reg. by 1
viii) Repeat (v) to (iii) till D-reg. become zero.
ix) Decrement C reg. by 1
x) Repeat step (iii) to (ix) till C reg become zero.

Code:-

```
jmp start
start: nop
LXI H, 2500H        ; Set pointer for array
MOV C, M            ; Load count
DCR C               ; Decrement count (N-1)

REPEAT: MOV D, C    ; Move C reg to D
        LXI H, 2501H ; Starting address of data array

LOOP: MOV A, M      ; Copy content of Memory to Accumulator
      INX H         ; Increment HL pair
      CMP M         ; Compare Memory with Accumulator
      JNC SKIP      ; If no carry Jump to SKIP
```


Teacher's Signature _____

Expt. No. _____ Page No. _____

```
MOV B,M     ; Content of Memory to B-reg.
MOV M,A     ; Content of Accumulator to memory
DCX H       ; Decrement HL Pair
MOV M,B     ; Content of B-reg to Memory
INX H       ; Increment HL Pair

SKIP: DCR D     ; Decrement D-reg
      JNZ LOOP  ; If no zero Jump to LOOP
      DCR C     ; Decrement C-reg
      JNZ REPEAT ; If no zero Jump to REPEAT

HLT         ; Terminate
```

Result: Program for descending order written successfully

Input: 2500H → 10 2506H → 19
2501H → 8 2507H → 50
2502H → 4 2508H → 43
2503H → 6 2509H → 100
2504H → 2 250AH → 15
2505H → 1

OUTPUT: 2501H → 100 2506H → 8
2502H → 50 2507H → 6
2503H → 43 2508H → 4
2504H → 19 2509H → 2
2505H → 15 250AH → 1

Teacher's Signature _____

Expt. No. 4 Page No.

Experiment- 4

Aim:- Find the largest element from an array of n element.

Algorithm:-

1) Load address of first element in HL pair
2) Move count to B-reg
3) Increment Pointer
4) Get first data in A-reg
5) Decrement Count
6) Increment Pointer
7) Compare content of Accumulator with memory.
8) If Carry = 0 go to 10th else go to 9
9) Move content of Memory to Accumulator
10) Decrement Count
11) If Not Zero go to step 6 & if Zero go to next
12) Store result in another memory.
13) Terminate Program.

Code:-

```
jmp start
start: nop

LXI H, 2500H        ; No of Element location
MOV B, M            ; Store Element count in B
INX H               ; Increment HL pair
MOV A, M            ; Move first Element to Accumulator
DCR B               ; Decrement Count (B)
```

NOOTAN

Teacher's Signature

Expt. No_____ Page No_____

```
LOOP2: INX H      ; Increment HL Pair
       CMP M      ; Compare Memory with Accumulator
       JNC LOOP   ; Jump to LOOP if No Carry (Accumulator > M)
       MOV A, M   ; Move greater value to Accumulator

LOOP1: DCR B      ; Decrement B
       JNZ LOOP2  ; Jump to LOOP2 until count=0
       STA 250AH  ; Store result in 250AH

HLT               ; Terminate Program.
```

RESULT: Program for finding largest element is written successfully.

INPUT: 2500H → 5
2501H → 100
2502H → 255
2503H → 40
2504H → 90
2505H → 10

OUTPUT → 250AH → 255

Teacher's Signature _____

Expt. No 5 Page No

Experiment - 5

Aim: Write a program in 8086 to perform following operation on two 16-bit numbers:-

i) Addition ii) Subtraction iii) Multiplication

FOR Addition

Algorithm:

1) Initialize the data segment
2) Get first no. in AX
3) Get Second no in BX
4) Initialize reg CL = 0
5) Add two number
6) Check for carry
7) Increment CL if carry
8) Display result
9) Display carry
10) Stop

Code:

```
code segment
Assume CS: code
Start:
        MOV AX, 0000H
        MOV BX, AX
        MOV DX, AX
        MOV SI, 3000h
        MOV AX, [SI]
        inc SI
        INC SI
        MOV BX, [SI]
```

Expt. No_____ Page No_____

```
inc si
inc si
add ax, bx
mov [si], ax
jc l1
inc si
inc si
mov [si], dx
int 3
l1: inc dx
inc si
inc si
mov [si], dx
int 3
code ends
end start
```

Result :- Program for Addition of 16 bit no in 8086 is written successfully

Input :- 3000h :- 12
3001h :- b6
3002h :- 12
3003h :- A6

OUTPUT :- 3004h → 24
3005h → 68
3006h → 01 (Carry)
3007h → 00

NOOTAN

Teacher's Signature _____

Expt. No_____ Page No_____

For Subtraction

Algorithm:

i) Initialize data segment
ii) Get first no in AX
iii) Get second no in BX
iv) Subtract two numbers
v) check for borrow
vi) Display result
vii) Stop

Code:

```
code segment
assume cs: code
start:
        mov ax, 0000h
        mov bx, ax
        mov dx, ax
        mov si, 3000h
        mov ax, [si]
        inc si
        inc si
        mov bx, [si]
        inc si
        inc si
        sub ax, bx
        mov [si], ax
        Jc l1
        inc si
```

```
        inc si
        mov [si], dx
        int 3
    l1: inc dx
        inc si
        inc si
        mov [si], dx
        int 3
code ends
end start
```

RESULT:- Program for subtraction of two 16 bit numbers done successfully.

Input:-		OUTPUT:-	
3000H → 12		3004H :- 00	
3001H → bc		3005H :- 10	
3002H → 12		3006H :- 00	
3003H → 9c		3007H :- 00	

FOR MULTIPLICATION

Algorithm:- i) Initialize data segment
ii) Get first no in AX
iii) Get second no in BX as counter
iv) Initialize Result = 0
v) Result = Result + First Number
vi) Decrement BX
vii) If count (BX != 0) go to step 5
viii) Display Result then stop

Teacher's Signature _____

Expt. No_____ Page No_____

Code:

```
code segment
assume cs: code
start:  mov ax, 0000h
        mov bx, ax
        mov dx, ax
        mov cx, ax
        mov si, 3000h
        mov ax, [si]
        mov cx, ax
        inc si
        inc si
        mov bx, [si]
        dec bx
        inc si
        inc si
    u:  add ax, cx
        dec bx
        jnz u
        mov [si], ax
        int 3
code ends
end start
```

Result:- Code for Multiplication Ran Successfully

Input: 3000h → c5 3001h → 16 3002h → 08 3003h → 00

OUTPUT: 3004h → 28 3005h → DE

NOOTAN

Teacher's Signature _____

Expt. No 6 Page No

Experiment-6

Aim:- Write a program in 8086 to generate Fibonacci Series for 10 terms

Algorithm:-

i) Initialize CX with no of terms
ii) Initialize AX = 01h BX = 01h
iii) Move AX to resultant series
iv) Increment series pointer
v) Move BX to Memory
vi) Decrement CX by 1
vii) Increment memory to next location
viii) Add AX & BX
ix) Move AX to memory
x) Move BX to DX
xi) Move AX to BX
xii) Move DX to AX
xiii) Decrement CL by 1
xiv) If CL != 0 go to (viii)
xv) End

Code:-

```
code Segment
    assume cs: code
    start
        mov ax, 0000h
        mov bx, 0001h
```

NOOTAN Teacher's Signature

Expt. No_____ Page No_____

```
mov cx, 0005h
mov si, 3000h
mov [si], ax
inc si
mov [si], bx
dec cx
L1: add ax, bx
inc si
mov [si], ax
mov ax, bx
mov bx, [si]
dec cx
jnz L1
int 3
code ends
end start
```

Result :- Program for fibonacci executed successfully.

Input :→ ax → 0000h
bx → 0001h
cx → 0005h

OUTPUT :- 3000h → 00 3005h → 05
3001h → 01 3006h → 08
3002h → 01 3007h → 00
3003h → 02 3008h → 15
3004h → 03 3009h → 22.

NOOTAN

Teacher's Signature _____

Expt. No 7 Page No

Experiment-7

Aim:- Write a program in 8086 to generate factorial of a 8 bit number.

Algorithm:-

1) load no. whose factorial it to be calculated in AX
2) copy content of AX into CX
3) Decrement CX by 1
4) Multiply CX and AX
5) Compare CX with 01H
6) If Zero=1 repeat step 4
7) When Z=1 mov content to AX to memory
8) End program.

Code:-

```
code Segment
Assume CS: code
start:
        mov si,3000h
        mov Ax,[Si]
        mov Bx, Ax
        dec Bx
l1:     Mul Bx
        dec Bx
        Jnz l1
        Inc Si
        mov [Si], ax
```

NOOTAN

Teacher's Signature

Expt. No_____ Page No_____

int 3

code ends

end start

RESULT:- code for factorial executed successfully.

Input :→ 3000H → 06

OUTPUT :→ 3001H → 00

3002H → 02

NOOTAN

Teacher's Signature _____

Expt. No_____ Page No_____

Architecture of 8086

8086 consist of two main Sections -
1) Bus interface unit
2) Execution Unit.

⇒ 8086 is a 16 bit microprocessor having 20 address lines & 16 data line that provide upto 1MB storage, It is available in 3 version based on :-
i) 8086 - 5 MHz ii) 8086 - 2 → 8MHz iii) 8086 - 1 → 10MHz

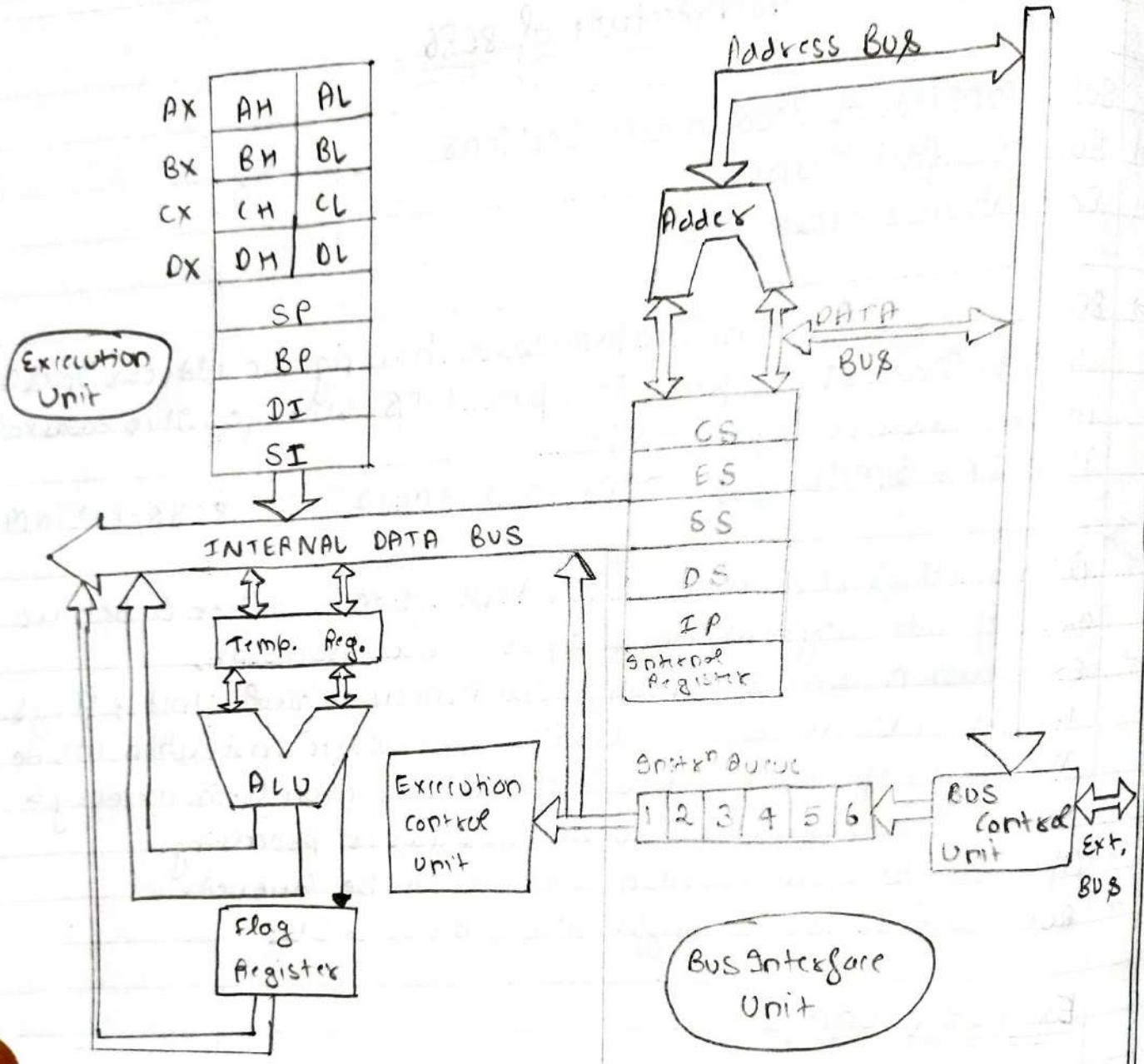
⇒ Both units of 8086 are independent of each other & behave as seperate asynchronous operational processor.
→ Execution units contains Arithmetic logical Unit, flags when EU decode instruction or execute instruction inside the microprocessor, BIU Prefetch instruction from memory & store them in instr^n queue for faster processing.
→ Upto 6 byte of instruction stream can be queued.
→ Queue act as FIFO buffer b/w BIU & EU.

Execution unit.

→ It consist of four 16 bit General purpose data register can also be use as 8-bit registers.
→ four 16 bit pointer & base register & one 16 bit flag register.

NOOTAN

Teacher's Signature _____



8086 Architecture

General Purpose data Register :-

- It can be divided in two parts such as higher & lower part to store 8-bit data.
- **AX Register :-** It is used as a 16-bit accumulator where as AL is used as a 8-bit accumulator. It is used in division, Multiplication & shift & Rotate instruction.
- **BX Register :-** It can be used as a memory pointer in DS. It can be used for based, based Index or Register Indirect addressing modes.
- **CX Register (CL) :-** It can be used as a counter in string manipulation & shift/ rotate instⁿ & default counter in loop instⁿ
- **DX Register :-** It is used in DIV instⁿ to hold higher ward of 32-bit operand & remainder after division, used in multiplic to hold higher ward of 32 bit result.

⇒ **Pointers & Base Register :-** Four 16-bit pointer, base Registers.

- **Stack Pointer (SP) :-** Pointing to Program stack.
- **Base Pointer (BP) :-** Pointing to data in SS.
- **Source Index (SI) :-** Used for index, Based-Indexed & register indirect addressing, as well as source data address in string manipulation instⁿ

Teacher's Signature _____

Expt. No_____ Page No_____

- Destination Index (DI) : Used for Indexed, Indexed register Indirect addressing as well as destination data address in string manipulation instruction.

⇒ Flag Registers :- Only 1, 16-bit flag Register & set of 16-independent flipflops (FFs) (6 status & 3 control flags)

Status flag :-

i) Zero flag (Z) :- Result is zero or not, Z=1 for zero & 0 for not zero.

ii) Carry flag (Cy) :- carry/Borrow generated out of MSB.

iii) Sign flag (S) :- Result is +ve/-ve 0 for +ve, 1 for -ve.

iv) Parity flag (P) :- odd parity = 0, Even parity = 1

v) Auxilliary Carry flag (AC) :- Convert Binary to BCD.

vi) Overflow flag (OF) :- If signed result is of more bits than the destination operand then this flag will be set other wise it will be cleared.

Control flag :-

(i) Trap flag (TF) :- If TF=1 Single step interrupt occurs after execution of next inst^n.

iv) Interrupt Enable (IF) :- Mask or Unmask Maskable interrupts.

v) Direction Flag (DF) : Used in string related operation.
auto decrement [SI or DI] = 1
auto increment = 0

→ BUS INTERFACE UNIT :-

- Segment Registers : Memory of 8086 is of 1MB which is divided into segments, at a time access four segments [CS, SS, DS, ES].
→ Each segment is upto 64 KB long
→ Each segment is independent & seperately addressable unit.
& if assigned a Base address which is its starting location in memory
→ Segment may be adjacent, disjoint, partially overlapped or fully overlaped.
→ These register are used with 16-bit pointer, index & base register to generate the 20-bit physical address.
→ Segmented architecture was used in 8086 to keep compatibility with earlier processor 8085.

a) Code Segment (CS) :- 16-bit register, store & base address of 64 KB, & microprocessor instⁿ or program.
- IP is default register used by µp to access instⁿ from CS
- Can't change directly.
- During execution of for JUMP, for CALL & For Return instⁿ CS reg. auto updated.

Teacher's Signature

Expt. No_____ Page No_____

b) Stack Segment (SS) : 16 bit register containing offset add. of 64 KB used as stack memory (LIFO)

- By default, SP & BP are the pointer register.
- PUSH & POP are main instⁿ.
- can't be initialize by loading immediate value in SS.
- can change directly using POP.

(c) Data Segment (DS) : hold logical add of 64 KB long DS, & used to store data.

- Default register are → AX BX CX DX & Index reg (SI, DI)
- can't be initialize by loading immediate value is DS, but can be change directly using POP & LDS reg instⁿ.

(d) Extra Segment (ES) : Used to store data, it is by default is the destⁿ location for string data which are always pointed by DI reg.

- can't initialize ES by immediate data, can be change using POP & LES instⁿ.

- Instruction Pointer (IP) & address Summation:-

IP contains the offset or logical add. of next byte to be need from code Segment.

Teacher's Signature _____