**EXP-1**

**Using DAD instruction:**

LHLD 0000H ; Get 1st 16-bit number in H-L pair

XCHG ; Save 1st 16-bit number in DE by exchanging with H-L pair

LHLD 0002H ; Get 2nd 16-bit number in H-L

DAD D ; Add DE and HL

SHLD 0004H ; Store 16-bit result in memory locations 0004H and 0005H

HLT ; Stop

**Using ADD & ADC instruction:**

LHLD 0000H ; Get 1st 16-bit number in H-L pair

XCHG ; Save 1st 16-bit number in DE

LHLD 0002H ; Get 2nd 16-bit number in H-L pair

MOV A, E ; Get lower byte of the 1st number

ADD L ; Add lower byte of the 2nd number

MOV L, A ; Store result in L-register

MOV A, D ; Get higher byte of the 1st number

ADC H ; Add higher byte of the 2nd number with CARRY

MOV H, A ; Store result in H-register

SHLD 0004H ; Store 16-bit result in memory locations 0004H and 0005H HLT ; Stop

**EXP-2**

LHLD 0000H ; Get first 16-bit number in HL   
XCHG ; Save first 16-bit number in DE  
LHLD 0002H ; Get second 16-bit number in HL   
MOV A, E ; Get lower byte of the first number   
SUB L ; Subtract lower byte of the second number   
MOV L, A ; Store lower byte subtraction result in L register   
MOV A, D ; Get higher byte of the first number   
SBB H ; Subtract higher byte of second number with borrow   
MOV H, A ; Store higher byte subtraction result in H register   
SHLD 0004H ; Store l6-bit result in memory locations 0004H and 0005H.   
HLT ; Terminate program execution

**EXP-3**

Assembly Language program for unconditional Jump instruction:  
  
MVI A, 07H ; get the value 07 H into accumulator

MVI B, 05H ; get the value 05H into B register

ADD B ; Add the content of accumulator with B reg and store in acc

JMP NEXT ; Jump unconditionally to label NEXT

INR A ; Increment content of accumulator. **Note**: this step is not executed

NEXT: HLT ; Stop the execution

Assembly Language program for Jump instruction when condition is not satisfied:

MVI A, 07H ; get the value 07 H into accumulator

MVI B, 05H ; get the value 05H into B register

ADD B ; Add the content of accumulator with B reg and store in acc

JC NEXT ; Jump if carry flag is set (Cy =1) to label NEXT

INR A ; Increment content of accumulator. **Note**: this step is executed

INR A ; Increment content of accumulator. **Note**: this step is executed

INR A ; Increment content of accumulator. **Note**: this step is executed

INR A ; Increment content of accumulator. **Note**: this step is executed

NEXT: HLT ; Stop the execution

Assembly Language program for Jump instruction when condition is satisfied:  
  
MVI A, 07H ; get the value 07 H into accumulator

MVI B, 05H ; get the value 05H into B register

ADD B ; Add the content of accumulator with B reg and store in acc

JNC NEXT ; Jump if carry flag is not set ( CY =0 )to label NEXT

INR A ; Increment content of accumulator. **Note**: this step is executed

INR A ; Increment content of accumulator. **Note**: this step is executed

NEXT: HLT ; Stop the execution

**EXP-4**

To write an assembly language program to perform one byte BCD addition.

MVI C, 00H

LHLD 0000H

MOV A, L

ADD H

DAA

JNC LOOP

INR C

LOOP: STA 0002H

MOV A, C

STA 0003H

HLT

**EXP-5**

To write an assembly language program to move block of data from one offset address to another offset address.

MVI C,05H

LXI H,1000H

LXI D,2000H

LOOP: MOV A,M

STAX D

INX H

INX D

DCR C

JNZ LOOP

HLT

**EXP-6**

Write a program to add two 8 bit numbers stored at internal Ram locations 20H and 21H.   
Store Sum at 22H and Carry at 23H.

org 0

start:

MOV R2, #00H

MOV A, 30H

ADD A, 31H

JNC Skip

INC R2

Skip:

MOV 32H, A

MOV 33H, R2

here: SJMP here

end

Write a program to subtract two 8 bit numbers stored at internal Ram locations 20H and 21H. Store Result at 22H and Borrow at 23H.

Org 0

start:

MOV R2, #00H

MOV A, 20H

CLR C

SUBB A, 21 H ; perform the subtraction

JNC Skip

INC R2 ; only when there is a borrow

Skip:

MOV 22H, A ; store difference from A to 22

MOV 23H, R2 ; store borrow from R2 to 23

Here: SJMP here

end

Write a program to multiply two 8 bit numbers stored at internal Ram locations 20H and 21H. Store Result at 22H and 23H.

org 0

start:

MOV A, 20H

MOV B, 21H

MUL AB

MOV 22H,A

MOV 23H,В

here: SJMP here

end

Write a program for 8 bit division. Dividend at internal Ram location 20H and Divisor at 21H. Store Quotient at 22H and Remainder at 23H.

org 0

start:

MOV A,60H

MOV b,61H

DIV AB

MOV 62H,A

MOV 63H,b

here: SJMP here

end

**EXP-7**

**Steps to generate a square wave using 8051**

1. Initially clear port 0.7
2. Set the output of any port on the 8051 to logic low.
3. Wait for some time.
4. Set the output of the same port to logic high.
5. Again wait for the same amount of time as earlier.
6. Loop around the same.

CLR P0.7

Back: MOV A,#00H

MOV P1,A

CALL DELAY

MOV A,#0FFH

MOV P1,A

CALL DELAY

LJMP Back

DELAY:MOV R2,#02FH

Here: DJNZ R2,Here

RET

**EXP-8**

org 0

start:

MOV A,#0E5H

MOV P1,A

End

org 0

main: MOV P1,P2

Jmp main

End

org 00h

Back: MOV A,#0FEH ; Data

MOV P1,A ; Output Data to LEDs

Acall Delay ; call delay

MOV A,#0FFH ; Data

MOV P1,A ; Ouput Data to LEDs

Acall Delay ; call delay

SJMP Back ; do it again

Delay: MOV R0,#0FFH ; outer loop

Again: MOV R1,#0FFH ; inner loop

Here: Djnz R1,Here ; jump if R1 not zero

Djnz R0,Again ; jump if R0 not zero

Ret

End

**EXP-9**

**Program to display Number 1 on LCD 3**

Org 00h

select: setb P0.7

setb P3.3

setb P3.4

start: mov A,#0f9h

mov P1,A

end

**Program to display Number 5 on LCD 2**

Org 00h

select: setb P0.7

Clr P3.3

setb P3.4

start: mov A,#92h

mov P1,A

end

MOV 30H,#'H'

MOV 31H,#'E'

MOV 32H,#'L'

MOV 33H,#'L'

MOV 34H,#'0'

MOV 35H,#20H

MOV 36H,#'W'

MOV 37H,#'O'

MOV 38H,#'R'

MOV 39H,#'L'

MOV 3AH,#'D'

MOV 3BH,#0H

**;end of data marker**

**;initialize the display**

**;see instruction set for details**

CLR P1.3 **; clear RS- indicates that instructions are being sent to the module**

**;function set**

CLR P1.7

CLR P1.6

SETB P1.5

CLR P1.4 ; high nibble set

SETB P1.2

CLR P1.2 ; negative edge on E

CALL DELAY ; wait

SETB P1.2

CLR P1.2 ; negative edge on E

SETB P1.7 ; low nibble set

SETB P1.2

CLR P1.2 ;negative edge on E

CALL DELAY

; entry mode set

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4 ; high nibble set

SETB P1.2

CLR P1.2 ;negative edge on E

SETB P1.6

SETB P1.5 ; low nibble set

SETB P1.2

CLR P1.2 ; high nibble set

CALL DELAY ; wait

CLR P1.7

CLR P1.6

CLR P1.5

CLR P1.4

SETB P1.2

CLR P1.2

SETB P1.7

SETB P1.6

SETB P1.5

SETB P1.4

SETB P1.2

CLR P1.2

CALL DELAY

SETB P1.3 ; clear RS

MOV R1,#30H ; data to be sent to LCD is stored in 8051 RAM , starting at location 30 H

LOOP: MOV A,@R1 ; move data pointed to by R1 to A

JZ FINISH ; if A is 0, then end of data has been reached, jump out of loop

CALL SendChar ; send Data in A to LCD module

INC R1

JMP LOOP

FINISH:

JMP $

SendChar: MOV C,ACC.7

MOV P1.7,C

MOV C,ACC.6

MOV P1.6,C

MOV C,ACC.5

MOV P1.5,C

MOV C,ACC.4

MOV P1.4,C

SETB P1.2

CLR P1.2

MOV C,ACC.3

MOV P1.7,C

MOV C,ACC.2

MOV P1.6,C

MOV C,ACC.1

MOV P1.5,C

MOV C,ACC.0

MOV P1.4,C ; low nibble set

SETB P1.2

CLR P1.2 ; negative edge on E

CALL DELAY

DELAY: MOV R0,#50H

DJNZ R0,$

RET

**EXP-10**

**Program to rotate DC Motor**

Org 00h

Main: SETB P3.1

ACALL rotate

rotate: MOV R1, #02H

Delay: DJNZ R1,Delay

JZ Loop

RET

Loop : CLR P3.1

JMP Main