Semester- III Paper- III

DSC -1005 C: Electronics Communication and Microprocessor 8085

Section II: Microprocessor 8085

UNIT 4: Programming with 8085 Microprocessor

Presented By:

Dr. C. B. Patil

Vivekanad College (Autonomous), Kolhapur

Syllabus:

- Programs of Addition (8 and 16 bit), Subtraction, Multiplication,Division, Block Transfer and Exchange, Masking, ascending and descending order, Time delay generation using register and register pair, Detection of odd and even numbers.

Addition of Two 8 Bit Numbers : (Indirect addressing mode)

Statement: Add the contents of memory location 6001H into the memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H
7004		INX H	; HL point's 6001H
7005		ADD M	; Add second operand in first
7006		INX H	; HL points 6002H
7007		MOV M, A	; Store result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	08H	08H
6001H	04H	04H
6002H	XXH	0CH

Addition of Two 8 Bit Numbers : (Direct addressing mode)

Statement: Add the contents of memory location 6001H into the memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LDA 6000H	; Load ACC from memory location 6000H
7003		MOV B, A	; move first operand from ACC
7004		LDA 6001H	; Load ACC from memory location 6001H
7007		ADD B	; Add second operand in first
7008		STA 6002H	; store the result at 6002H
700B		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	08H	08H
6001H	04H	04H
6002H	XXH	0CH

Addition of Two 8 Bit Numbers : (Immediate addressing mode)

Statement: Add the given two numbers and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI A ,08H	; get first operand into ACC
7002		MVI B, 04H	; get second operand into reg. B
7004		ADD B	; Add second operand in first
7005		STA 6002H	; store the result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6002H	XXH	0CH

Subtraction of Two 8 Bit Numbers :

Statement: Subtract the contents of memory location 6001H from the contents of memory location 6000H and place the result in memory location 6002H.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H
7004		INX H	; HL point's 6001H
7005		SUB M	; Subtract operand from first operand
7006		INX H	; HL points 6002H
7007		MOV M, A	; Store result at 6002H
7008		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	0FH	0FH
6001H	04H	04H
6002H	XXH	0BH

Multiplication of Two 8 Bit Numbers

Statement: Multiply the contents of memory location 6001H by the contents of memory location 6000H and place the result in memory location 6002H.

Microprocessor 8085 has no direct multiplication instruction and the multiplication is done by using successive addition method. e.g 5x2=5+5

Here the number which is to be multiply is loaded in B and C register. Clear the accumulator(A) and add the contents of C in A repeatedly until contents of B becomes Zero.(for every addition, B is decremented by 1).

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV B, M	; Get first operand from location 6000H in B register
7004		INX H	; HL point's 6001H
7005		MOV C,M	; Get Second operand from location 6001H in C register
7006		MVI A, 00H	;Clear the contents of the accumulator(A)
7008	HERE:	ADD C	;Add the contents of reg .C in accumulator
7009		DCR B	;Decrement contents of B by 1
700A		JNZ HERE	; If not zero, repeat
700D		INX H	; HL points 6002H
700E		MOV M, A	; Store result at 6002H
700F		HLT	; Terminate program execution

Division of Two 8 Bit Numbers

Statement: Divide the contents of memory location 6001H by the contents of memory location 6000H and place the result with quotient in memory location 6002H and remainder in memory location 6003H. Microprocessor 8085 has no direct division instruction. So the division is done by using successive subtraction method. e.g. 5/2 then 5-2-2=1 (Reminder) two times subtraction means quotient is 2.

During each successive subtraction, we need to monitor the remainder should not be less than the quotient. if the remainder is less than the divisor then we need to stop the successive division. This condition is monitor by CMP B instruction. If accumulator data is larger than the data in register B then carry flag is reset and hence the successive division will continue. The number of repetitions will give the value of quotient and Accumulator holds the remainder.

Memory Address	Label	Mnemonics	Comments
7000	START:	LXI H, 6000H	; HL point's memory location 6000H
7003		MOV A, M	; Get first operand from location 6000H in A register
7004		INX H	; HL point's 6001H
7005		MOV B,M	; Get Second operand from location 6001H in B register
7006		MVI C,00H	;Clear the contains of the register C
7008	HERE:	SUB B	;Subtract contents of reg . B from A
7009		INR C	;Increment contents of C by 1
700A		CMP B	; Compare B with A
700B		JNC HERE	; If carry not generated then repeat
700E		INX H	; HL points 6002H
700F		MOV M, C	; Store result (quotient) at 6002H
7010		INX H	; HL points 6003H
7011		MOV M, A	; Store result (Remainder) at 6003H
7012		HLT	; Terminate program execution

Block Transfer

Statement: Write a program to transfer a block of 10 numbers from memory location 6000H into the 8000H.

Block transfer means to copy a block of data and store it a new location. To perform the block transfer operation two data pointers are required one is at source and second is at the destination. Here in the program HL pointer reads the data from the source and DE pointer stores the data at the destination. After coping a byte of data both pointers are incremented by one till all the data is transferred. Register C is configured as a counter for the data transfer which decrements with each data transfer.

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI C, 0AH	; Initialize counter
7002		LXI H, 6000H	; HL point's memory location 6000H
7005		LXI D, 8000H	; DE point's memory location 8000H
7008	BACK:	MOV A,M	;Get no. from memory location pointed by HL pair
7009		STAX D	;Store no. into memory location pointed DE pair
700A		INX H	; HL point's next memory location
700B		INX D	; DE point's next memory location
700C		DCR C	; Decrement counter by 1
700D		JNZ BACK	: If not zero, repeat
7010		HLT	; Terminate program execution

4.7 BLOCK EXCHANGE

Statement: Write a program to exchange the memory block of 10 numbers between memory location 6000H and the 8000H

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI C, 0AH	; Initialize counter
7002		LXI H, 6000H	; HL point's memory location 6000H
7005		LXI D, 8000H	; DE point's memory location 8000H
7008	BACK:	MOV B,M	;Get first block number into B
		LDAX D	;Get second block number into A
7009		MOV M,A	;Store second block number into first block
700A		MOV A,B	; Get first block number into A from B
700B		STAX D	; Store first block number into second block
700C		INX H	; HL point's next memory location
700D		INX D	; DE point's next memory location
700E		DCR C	; Decrement counter by 1
700F		JNZ BACK	: If not zero, repeat
7012		HLT	; Terminate program execution

4.3 16 BIT ADDITION

Statement: Add the 16-bit number in memory locations 6000H and 6001H to the 16-bit number in memory locations 6002H and 6003H. The most significant eight bits of the two numbers to be added are in memory locations 6001H and 6003H. Store the result in memory locations 6004H and 6005H with the most significant byte in memory location 6005H. Also store the carry at 6006H

Memory Address	Label	Mnemonics	Comments
7000	START:	LHLD 6000H	; Get first I6-bit number in HL pair
7003		XCHG	; Save first I6-bit number in DE
7004		LHLD 6002H	;Get second I6-bit number in HL
7007		DAD D	; Add DE and HL
7008		SHLD 6004H	;Store 16-bit result in memory locations 4004H and 4005H
700B		RAL	;Rotate ACC left through CY flag
700C		ANI 01H	;mask bits D7-D1
700E		STA 6006H	;store the carry if generated to 6006H
7011		HLT	;Terminate program execution

Result:-

Memory Location	Before Execution	After Execution
6000H	11H	11H
6001H	22H	22H
6002H	33H	33H
6003H	44H	44H
6004H	XXH	44H
6005H	XXH	66H
6006H	XXH	00H

4.10 To find whether an 8 bits number is even or odd :

Sol: Given number is EVEN number if its lower bit is 0 i.e. low otherwise number is ODD.

To check whether the number is odd or even, we basically perform AND operation with 01 by using ANI instruction. If number is even then we will get 00 otherwise 01 in accumulator.

We use 11 to represent odd number and 22 to represent even number.

Algorithm:

Load the accumulator with the first data.

Perform AND operation with 01 on first data using ANI Instruction.

Check if zero flag is set then set the value of accumulator to 22 otherwise 11 to accumulator.

Now load the result value in memory location.

Memory Address	Label	Mnemonics	Comments
7000	START:	LDA, 6000H	; get number into ACC from memory location 6000H
7003		ANI 01H	; AND given number with 01H
7004		JZ EVEN	; If Z=1, go to display Even Number
7005		MVI A,11H	; otherwise load 11H into ACC
7006		JMP STORE	; Jump to store code 11H as given no is Odd
	EVEN:	MVI A, 22H	;Load 22H into Acc
7007	STORE:	STA 6050H	; Store result at 6002H
7008		HLT	;Terminate program execution

4.10 To find total number of even and odd numbers in an array :

Memory Address	Label	Mnemonics	Comments
7000		MVI C, 05H	; load count into C
7002		MVI D, 00H	; clear D
7004		MVI E, 00H	; clear E
7006		LXI H, 6000H	; Load HL to point First no.
7009	NEXT:	MOV A,M	; get no. from memory
700A		ANI 01H	;AND with 01H
700C		JZ EVEN	; if Z=1, go to increment even no. count
700F		INR E	;else increment odd no. count
7010		JMP DEC	; go to decrement count
7013	EVEN:	INR D	; increment even no. count
7014	DEC:	INX H	;increment memory pointer
7015		DCR C	;decrement count
7016		JNZ NEXT	;if all nos. Are not verified, go to next memory location
7019		MOV A,D	;get even count from D into ACC
701A	STORE:	STA 6060H	; store even count to 6060H
701D		MOV A,E	; ;get odd count from D into ACC
701E		STA 6061H	;store odd count to 6061H
7021		HLT	; Terminate the program

4.10 To find total number of even and odd numbers in an array :

Memory Address	Label	Mnemonics	Comments
7000		MVI C, 05H	; load count into C
7002		MVI D, 00H	; clear D
7004		MVI E, 00H	; clear D
7006		LXI H, 6000H	; Load HL to point First no.
7009	NEXT	MOV A,M	; get no. from memory
700A		RAR	;rotate ACC right through carry flag
700B		JC odd	; if CY=1, go to increment odd no. count
700E		INR D	;else increment even no. count
700F		JMP DEC	; go to decrement count
7012	odd:	INR E	; increment even no. count
7013	DEC	INX H	;increment memory pointer
7014		DCR C	;decrement count
7015		JNZ NEXT	;if all nos. Are not verified, go to next memory location
7018		MOV A,D	;get even count from D into ACC
7019	STORE:	STA 6060H	; store even count to 6060H
701C		MOV A,E	; ;get odd count from D into ACC
701D	STORE:	STA 6061H	;store odd count to 6061H
7020		HLT	; terminate program

4.8 ARRANGE NUMBERS IN ASCENDIG ORDER:

Statement: Write a program to sort given 10 numbers from memory location 6000H in the ascending order.

Memory Address	Label	Mnemonics	Comments
7000		MVI B, 09	; Initialize counter 1
7002	START:	LXI H, 6000H	; Initialize memory pointer (HL point’s memory location 6000H)
7005		MVI C, 09H	; Initialize counter 2
7007	BACK:	MOV A, M	; Get the number from memory location pointed by HL pair
7008		INX H	; Increment memory pointer
7009	HERE:	CMP M	; Compare number with next number
700A		JC SKIP	; If less, don't interchange
700D		JZ SKIP	; If equal, don't interchange
7010		MOV D, M	;otherwise interchange
7011		MOV M, A	two numbers
7012		DCX H	using reg.D
7013		MOV M, D	and
7014		INX H	memory pointer
7015	SKIP:	DCR C	: Decrement counter 2
7016		JNZ BACK	: If not zero, repeat
7019		DCR B	; Decrement counter 1
701A		JNZ START	; if all passes are not completed then go back and repeat
701D		HLT	; Terminate program execution

4.9 ARRANGE NUMBERS IN DECENDING ORDER:

Statement: Write a program to sort given 10 numbers from memory location 6000H in the descending order.

Memory Address	Label	Mnemonics	Comments
7000		MVI B, 09	; Initialize counter 1
7002	START:	LXI H, 6000H	; Initialize memory pointer (HL point's memory location 6000H)
7005		MVI C, 09H	; Initialize counter 2
7007	BACK:	MOV A, M	; Get the number from memory location pointed by HL pair
7008		INX H	; Increment memory pointer
7009		CMP M	; Compare number with next number
700A		JNC SKIP	; If large, don't interchange
700D		JZ SKIP	; If equal, don't interchange
7010		MOV D, M	;otherwise interchange
7011		MOV M, A	two numbers
7012		DCX H	;decrement memory pointer
7013		MOV M, D	;copy no. to memory location
7014		INX H	; increment memory pointer
7015	SKIP:	DCR C	; Decrement counter 2
7016		JNZ BACK	; If not zero, repeat
7019		DCR B	; Decrement counter 1
701A		JNZ START	; if all passes are not completed then go back and repeat
701D		HLT	; Terminate program execution

Masking :

Mask upper nibble of given number :

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI A, 25H	; get number into ACC
7002		ANI 0FH	; AND given number with 0FH
7004	STORE:	STA 6050H	; Store result at 6050H
7007		HLT	;Terminate program execution

Result: 6050H - 05H

Mask upper nibble of given number :

Memory Address	Label	Mnemonics	Comments
7000	START:	MVI A, 25H	; get number into ACC
7002		ANI F0H	; AND given number with F0H
7004	STORE:	STA 6050H	; Store result at 6050H
7007		HLT	;Terminate program execution

Result: 6050H - 20H

4.10 SUBROUTINE FOR TIME DELAY

4.10.1 Using single register:

	Mnemonics	T States
	MVI B FF H	7
AGAIN:	DCR B	4
	JNZ AGAIN	10/7
	RET	3

In above program register B is loaded with FF (255) and it is decremented by one. The content of register B is checked for zero and again decremented until it will become zero.

T states (No of clock pulses required to execute instruction) are given by Intel.

Total clock pulses for above program:

MVI	1 * 7	=	7	; since it is executed by one time
DCR	255 * 4	=	1020	
JNZ	254 * 10	=	2540	
JNZ (No jump)	1 * 7	=	7	
RET	1*3	=	3	
Total clock pulses		**=**	**3577**	

If crystal connected to processor is of 6 MHz, internal frequency is 3 MHz and time required for one clock pulse is 1/T= 1/3MHz = $0.33 * 10^{-6}$ second = 0.33 µS

Hence total time delay produced by above subroutine is,

3577 * 0.33µS = 1180.41 µS = 1.18ms.

Total time delay (T_D) = Time Delay outside the loop (T_0) + Time delay inside Loop (T_L)

Time delay outside loop is due to instructions which are executed only one time.
In above program it is MVI B and RET.
$T_0 = (7+3) \times 0.33 \times 10^{-6} = 3.3 \times 10^{-6} = 3.3\mu S$

The loop instructions are : DCR B and JNZ with total loop T-states 14 (4+10)

$$\mathbf{T_L = (T * Loop\ T\text{-}states * N_{10})}$$

Where T = System clock period

N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay register

T_L = 0.33*14*255 = 1178.1µS = 1.1781ms

Total time delay (T_D)= Time Delay outside the loop (T_O) + Time delay inside Loop (T_L)
= 3.3 µS + 1178.1 µS
= 1181.4 µS

= 1.1814 ms

4.10.2 Time delay using register pair

	Mnemonics		T States
	MVI B 64 H	(REG.B=64H = 100_{10})	7
AGAIN:	MVI C FF H		7
BACK:	DCR C		4
	JNZ C BACK		10/7
	DCR B		4
	JNZ AGAIN		10/7
	RET		

In this program, register C is decremented 255 times, then register B is decremented by one. Again C is loaded 255 times. This process repeated 100 times. In this way reg. C is loaded and decremented 255 X 100 = 25500 times.

Calculations of time delay:

	No. of T states:
MVI B 64 H	7
DCR B	4
JNZ AGAIN	10/7
RET	10

T_D= Time Delay outside the loop (T_0) + Time delay inside Loop (T_L)

Time delay outside loop is due to instructions which are executed only one time. In above program these are MVI B and RET

T_0= (7+3) X 0.33 X 10^{-6} =10 X0.33 X10^{-6} =9.9 µS

T_L for Loop1=T_{L1}

```
BACK    DCR C                          4
        JNZ C BACK                     10
```

T_L= (T* Loop T-states * N_{10})

Where, T = System clock period

N_{10} = Equivalent decimal number of the hexadecimal count loaded in the delay register

T_{L1} = 0.33 µS *14*255

= 3531.33 µS

= 3.53mS

T_L for Loop2=T_{L2}

The loop 2 is executed 100 times because the count (64H) in register B.

```
Loop 2=   AGAIN:  MVI C FF          7
                  DCR B             4
                  JNZ AGAIN         10/7
```

TL2 = 100(TL1+21 T-states*0.33) µS

= 100(3531.33+20.79) µS

= 3552.12*100 µS

= 355212.00 µS

= 355.21 mS

In this way time delay can be increased by using register pairs.

T_L= Loop 1= BACK DCR C (Repeated 255 X 10 (REG. C) times)

JNZ C BACK

= count X Loop T states X Clock Period

= (255 X 10) X (7 For DCR C + 10 For JNZ) X 0.99 µS

= 2550 X 17 X 0.99 µS

= 42916 µS

Loop 2= AGAIN: MVI C FF

DCR B

JNZ AGAIN

REG.B is loaded with 64 H (100) . Hence,

= count X Loop T states X Clock Period

= 100 X (7 for DCR B + 10 For JNZ+ 7 for MVI C) X 0.99 µS

= 100 x 27 x 0.99 µS

= 2773 µS

Hence T_L= 42916 +2773= 45589 µS

Total Delay = (45589 +9.9) µS

= 45598.9 µS

= 45.598 ms

- **Long answer questions (8 Marks)**

1. Write an ALP (Assembly Language Program) to add two 8 bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after addition.
2. Write an ALP (Assembly Language Program) to subtract two 8 bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after subtraction.
3. Write an ALP(Assembly Language Program) to multiply two8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after multiplication.
4. Write an ALP(Assembly Language Program) to divide two 8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after division.
5. Write an ALP(Assembly Language Program) to transfer a block of data from 6000H and store it at 8000H. copy 10 number of bytes in sequential manner.

1. Write an ALP (Assembly Language Program) Add the 16-bit number in memory locations 6000H and 6001H to the 16-bit number in memory locations 6002H and 6003H. The most significant eight bits of the two numbers to be added are in memory locations 6001H and 6003H. Store the result in memory locations 6004H and 6005H with the most significant byte in memory location 6005H. Also store the carry at 6006H.
2. Write an ALP(Assembly Language Program) to exchange a block of data between 6000H and 8000H. copy 10 number of bytes in sequential manner.
3. Write an ALP(Assembly Language Program) to multiply two8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after multiplication.
4. Write an ALP(Assembly Language Program) to divide two 8-bit numbers. The numbers are stored 6000 and 6001H memory location. Store the result at 6002H memory location after division.
5. Write an ALP(Assembly Language Program) to transfer a block of data from 6000H and store it at 8000H. copy 10 number of bytes in sequential manner.