EXPERIMENT NO. – 1

OBJECTIVE: Write a program to add two hexadecimal & decimal numbers.

APPARATUS REQUIRED: -

8085 Microprocessor programming kit, instruction coding sheet. Power supply A.C (230V Mains)

DESCRIPTION/ALGORITHM:-

Hexadecimal Addition:

The program takes the content of 2009, adds it to 200B & stores the result back at 200C.

Steps:

1. Initialize HL Reg. pair with address where the first number is lying.

2. Store the number in accumulator.

3. Get the second number.

4. Add the two numbers and store the result in 200B.

5. Go back to Monitor.

PROGRAM:-

```
LXI H, 2009 ;     Point 1st no.
MOV A, M ;      Load the acc.
INX H ;              Increase Pointer
ADD M ;           ADD 2nd NO.
INX H ;            Increase Pointer
MOV M, A ;     Store Result
HLT
```

Decimal Addition:

Steps:
1. Initialize HL Reg. pair with address where the first number is lying.
2. Store the number in accumulator.
3. Get the second number.
4. Add the two numbers and store the result in 200B.
5. Go back to Monitor

PROGRAM:-

```
 LXI H, 2009 ;   Point 1st no.
 MOV A, M ;      Load the acc.
 INX H ;                     Increase Pointer
 ADD M ;         ADD 2nd NO.
 DAA ;           Adjust the decimal
 INX H ;                     Inccrease Pointer
 MOV M, A ;      Store Result
 HLT
```

REULTS:-

Thus the numbers at 2009H and at memory are added.

CONCLUSION:-

Thus the program to add two 8-bit numbers was executed.

EXPERIMENT NO. – 2

OBJECTIVE: - Write a program using 8085 Microprocessor for addition and subtraction of two BCD numbers.

APPARATUS REQUIRED: -

8085 Microprocessor programming kit, instruction coding sheet. Power supply A.C (230V Mains)

DESCRIPTION/ALGORITHM:-

Steps for addition:

1. Initialize HL Reg. pair with address where the first number is lying.
2. Store the number in accumulator.
3. Get the second number.
4. Add the two numbers and store the result in 200B.
5. Go back to Monitor

Steps for subtraction:-

7. Initialize HL Reg. pair with address where the first number is lying.
8. Store the number in accumulator.
9. Get the second number.
10. Subtract second no from acc and store the result in 200B.
11. Adjust the decimal
12. Go back to Monitor

PROGRAM:-

```
 LXI H, 2009 ;          Point first number.
 MOV A, M ;             Load the acc.
 INX H ;                        Increase Pointer
 SUB M ;                Subtract second number.
 DAA ;                  Adjust the decimal
 INX H ;                        Increase Pointer
 MOV M, A ;             Store Result
 HLT                    We can use RST 5 also to end the program.
```

REULTS:-

The BCD numbers at 2009H and memory are added and substracted.

CONCLUSION:-

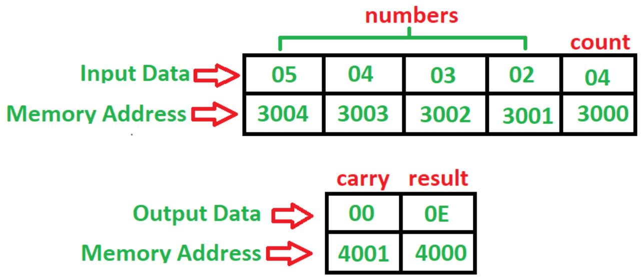
Thus the substraction and addition of BCD operation is taken out using assembly language.

EXPERIMENT NO. –3

OBJECTIVE: - Write a program to find the sum of a series where series starts from **3001** memory address and count of series is at **3000** memory address where starting address of the given program is **2000** store result into **4000** memory address.

APPARATUS REQUIRED: -

8085 Microprocessor programming kit, instruction coding sheet. Power supply A.C (230V Mains)



	numbers				count
Input Data	05	04	03	02	04
Memory Address	3004	3003	3002	3001	3000

	carry	result
Output Data	00	0E
Memory Address	4001	4000

ALGORITHM:-

1. Move 00 to register B immediately for carry
2. Load the data of memory [3000] into H immediately
3. Move value of memory into register C
4. Decrease C by 1
5. Increase H-L pair by 1
6. Move value of memory into accumulator
7. Increase H-L pair by 1
8. Add value of memory with accumulator
9. Jump if no carry to step 11
10. Increase value of register B by one
11. Decrease register C by 1
12. Jump if not zero to step-7
13. Store content of accumulator into memory [4000] (**result**)
14. Move content of register B into accumulator
15. Store content of accumulator into memory [4001] (**carry**)
16. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MVI	B, 00	[B] <- 00
2002	LXI	H, [3000]	[H-L] <- [3000]

Memory	Mnemonics	Operands	Comment
2005	MOV	C, M	[C] <- [M]
2006	DCR	C	[C] <- [C] – 1
2007	INX	H	[H-L] <- [H-L] + 1
2008	MOV	A, M	[A] <- [M]
2009	INX	H	[H-L] <- [H-L] + 1
200A	ADD	M	[A] <- [A] + [M]
200B	JNC	200F	jump if no carry
200E	INR	B	[B] <- [B] + 1
200F	DCR	C	[C] <- [C] – 1
2010	JNZ	2009	jump if not zero
2013	STA	[4000]	result
2016	MOV	A, B	[A] <- [B]
2017	STA	[4001]	carry
201A	HLT		Stop

EXPERIMENT NO. –4

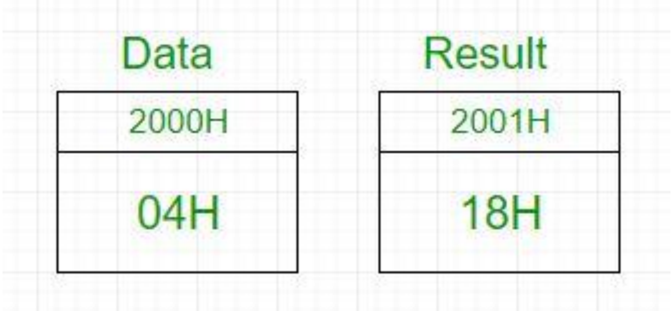
Problem – Write an assembly language program for calculating the factorial of a number using 8085 microprocessor.

Example –

Input : 04H

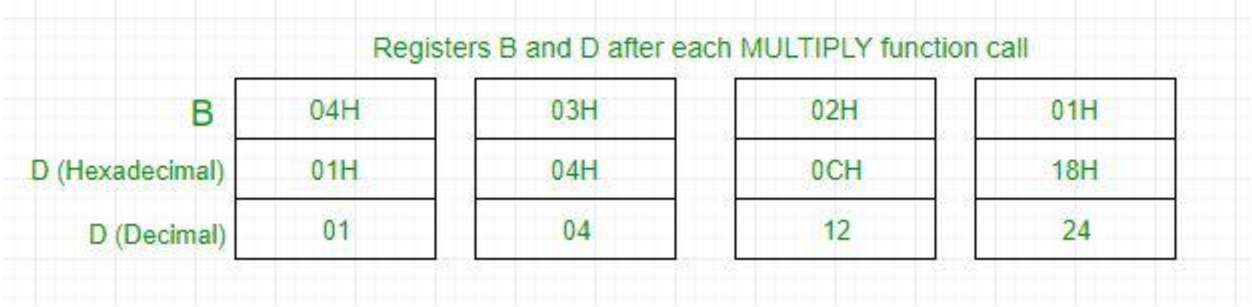
Output : 18H

as 04*03*02*01 = 24 in decimal => 18H



In 8085 microprocessor, no direct instruction exists to multiply two numbers, so multiplication is done by repeated addition as 4×3 is equivalent to 4+4+4 (i.e., 3 times).

Load 04H in D register -> Add 04H 3 times -> D register now contains 0CH -> Add 0CH 2 times -> D register now contains 18H -> Add 18H 1 time -> D register now contains 18H -> Output is 18H


Registers B and D after each MULTIPLY function call

B	04H	03H	02H	01H
D (Hexadecimal)	01H	04H	0CH	18H
D (Decimal)	01	04	12	24

Algorithm –

1. Load the data into register B
2. To start multiplication set D to 01H
3. Jump to step 7
4. Decrements B to multiply previous number
5. Jump to step 3 till value of B>0
6. Take memory pointer to next location and store result
7. Load E with contents of B and clear accumulator
8. Repeatedly add contents of D to accumulator E times
9. Store accumulator content to D
10. Go to step 4

Address	Label	Mnemonic	Comment
2000H	Data		Data Byte

Address	Label	Mnemonic	Comment
2001H	Result		Result of factorial
2002H		LXI H, 2000H	Load data from memory
2005H		MOV B, M	Load data to B register
2006H		MVI D, 01H	Set D register with 1
2008H	FACTORIAL	CALL MULTIPLY	Subroutine call for multiplication
200BH		DCR B	Decrement B
200CH		JNZ FACTORIAL	Call factorial till B becomes 0
200FH		INX H	Increment memory
2010H		MOV M, D	Store result in memory
2011H		HLT	Halt
2100H	MULTIPLY	MOV E, B	Transfer contents of B to C
2101H		MVI A, 00H	Clear accumulator to store result
2103H	MULTIPLYLOOP	ADD D	Add contents of D to A
2104H		DCR E	Decrement E
2105H		JNZ MULTIPLYLOOP	Repeated addition

Address	Label	Mnemonic	Comment
2108H		MOV D, A	Transfer contents of A to D
2109H		RET	Return from subroutine

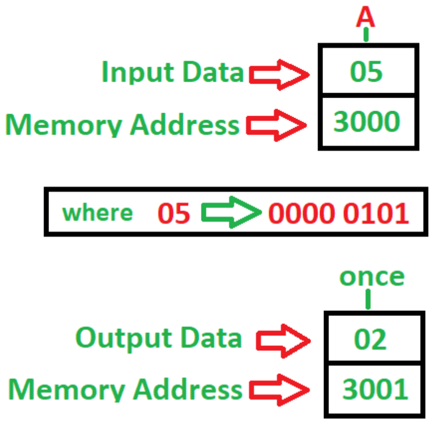
Explanation –

1. First set register B with data.
2. Set register D with data by calling MULTIPLY subroutine one time.
3. Decrement B and add D to itself B times by calling MULTIPLY subroutine as 4*3 is equivalent to 4+4+4 (i.e., 3 times).
4. Repeat the above step till B reaches 0 and then exit the program.
5. The result is obtained in D register which is stored in memory

EXPERIMENT NO. - 5

Problem – Write a program to count number of ones in the given 8-bit number use register B to display the count of ones where starting address is **2000** and the number is stored at **3000** memory address and store result into **3001** memory address.

Example –



Algorithm –

1. Move 00 to register B immediately for count
2. Move 08 to register C immediately for shifting
3. Load the data of memory [3000] into accumulator
4. Rotate ‘A’ right with carry
5. Jump if no carry to step-7
6. Otherwise increase register B by 1
7. Decrease register C by 1
8. Jump if not zero to step-4
9. Move content of register B into accumulator
10. Store content of accumulator into memory [3001] (**number of count**)
11. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MVI	B, 00	[B] <- 00
2002	MVI	C, 08	[C] <- 08
2004	LDA	[3000]	[A] <- [3000]
2007	RAR		rotate ‘A’ right with carry

Memory	Mnemonics	Operands	Comment
2008	JNC	200C	jump if no carry
200B	INR	B	[B] <- [B] + 1
200C	DCR	C	[C] <- [C] – 1
200D	JNZ	2007	jump if not zero
2010	MOV	A, B	[A] <- [B]
2011	STA	[3001]	number of ones
2014	HLT		Stop

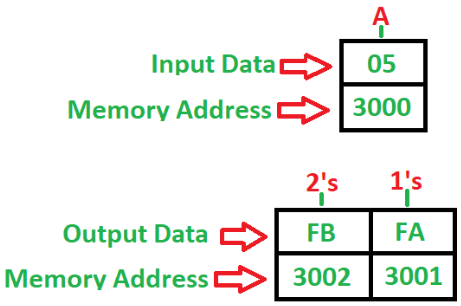
Explanation – Registers A, B and C are used for general purpose.

1. **MVI** is used to load an 8-bit given register immediately (2 Byte instruction)
2. **LDA** is used to load accumulator direct using 16-bit address (3 Byte instruction)
3. **MOV** is used to transfer the data from accumulator to register(any) or register(any) to accumulator (1 Byte)
4. **RAR** is used to shift ‘A’ right with carry (1 Byte instruction)
5. **STA** is used to store data from accumulator into memory direct using 16-bit address (3 Byte instruction)
6. **INR** is used to increase given register by 1 (1 Byte instruction)
7. **JNC** is used to jump to the given step if there is no carry (3 Byte instruction)
8. **JNZ** is used to jump to the given step if there is not zero (3 Byte instruction)
9. **DCR** is used to decrease given register by 1 (1 Byte instruction)
10. **HLT** is used to halt the program

EXPERIMENT NO. –6

Problem – Write a program to find 1’s and 2’s complement of 8-bit number where starting address is **2000** and the number is stored at **3000** memory address and store result into **3001** and **3002** memory address.

Example –



Algorithm –

1. Load the data from memory 3000 into A (accumulator)
2. Complement content of accumulator
3. Store content of accumulator in memory 3001 (1’s complement)
4. Add 01 to Accumulator content
5. Store content of accumulator in memory 3002 (2’s complement)
6. Stop

Program –

Memory	**Mnemonics**	**Operands**	**Comment**
2000	LDA	[3000]	[A] <- [3000]
2003	CMA		[A] <- [A^]
2004	STA	[3001]	1’s complement
2007	ADI	01	[A] <- [A] + 01
2009	STA	[3002]	2’s complement
200C	HLT		Stop

Explanation –

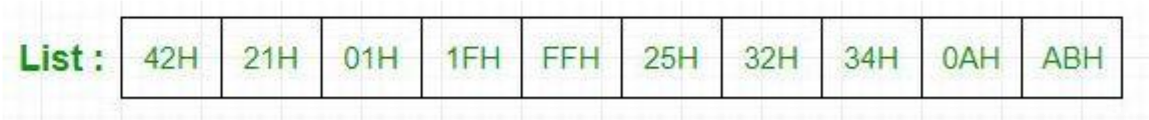
1. **A** is an 8-bit accumulator which is used to load and store the data directly

2. **LDA** is used to load accumulator direct using 16-bit address (3 Byte instruction)
3. **CMA** is used to complement content of accumulator (1 Byte instruction)
4. **STA** is used to store accumulator direct using 16-bit address (3 Byte instruction)
5. **ADI** is used to add data into accumulator immediately (2 Byte instruction)
6. **HLT** is used to halt the program

EXPERIMENT NO. –7

Problem – Write an assembly language program in 8085 microprocessor to find maximum and minimum of 10 numbers.

Example –



List :	42H	21H	01H	1FH	FFH	25H	32H	34H	0AH	ABH

Minimum: 01H, **Maximum:** FFH

In CMP instruction:
If Accumulator > Register then carry and zero flags are reset
If Accumulator = Register then zero flag is set
If Accumulator < Register then carry flag is set

Assumption – List of numbers from 2050H to 2059H and output at 2060H and 2061H.

Algorithm –

1. Maximum number is stored in B register and minimum in C register
2. Load counter in D register
3. Load starting element in Accumulator, B and C register
4. Compare Accumulator and B register
5. If carry flag is not set then transfer contents of Accumulator to B. Else, compare Accumulator with C register, if carry flag is set transfer contents of Accumulator to C
6. Decrement D register
7. If D>0 take next element in Accumulator and go to point 4
8. If D=0, store B and C register in memory
9. End of program

Program-

Address	Label	Instruction	Comment
2000H		LXI H, 2050H	Load starting address of list
2003H		MOV B, M	Store maximum
2004H		MOV C, M	Store minimum
2005H		MVI D, 0AH	Counter for 10 elements
2007H	LOOP	MOV A, M	Retrieve list element in Accumulator
2008H		CMP B	Compare element with maximum number

Address	Label	Instruction	Comment
2009H		JC MIN	Jump to MIN if not maximum
200CH		MOV B, A	Transfer contents of A to B as A > B
200DH	MIN	CMP C	Compare element with minimum number
200EH		JNC SKIP	Jump to SKIP if not minimum
2011H		MOV C, A	Transfer contents of A to C if A < minimum
2012H	SKIP	INX H	Increment memory
2013H		DCR D	Decrement counter
2014H		JNZ LOOP	Jump to LOOP if D > 0
2017H		LXI H, 2060H	Load address to store maximum
201AH		MOV M, B	Move maximum to 2060H
201BH		INX H	Increment memory
201CH		MOV M, C	Move minimum to 2061H
201DH		HLT	Halt

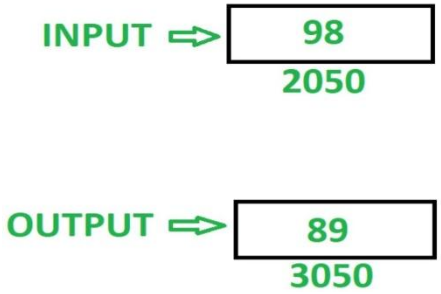
Explanation –

1. One by one all elements are compared with B and C register.
2. Element is compared with maximum, if it greater than maximum then it is stored in B register. Else, it is compared with minimum and if it is less than minimum then it stored in C register.
3. Loop executes 10 number of times.
4. At the end of 10 iterations, maximum and minimum are stored at 2060H and 2061H respectively.

EXPERIMENT NO. –8

Problem: Write an assembly language program in 8085 microprocessor to reverse 8-bit numbers.

Example:



Assume that number to be reversed is stored at memory location 2050, and reversed number is stored at memory location 3050.

Algorithm –

1. Load content of memory location 2050 in accumulator A
2. Use **RLC** instruction to shift the content of A by 1 bit without carry. Use this instruction 4 times to reverse the content of A
3. Store content of A in memory location 3050

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LDA 2050	A <- M[2050]
2003	RLC	Rotate content of accumulator left by 1 bit
2004	RLC	Rotate content of accumulator left by 1 bit
2005	RLC	Rotate content of accumulator left by 1 bit
2006	RLC	Rotate content of accumulator left by 1 bit
2007	STA 3050	M[2050] <- A
200A	HLT	END

Explanation – Register A used:

1. **LDA 2050:** load value of memory location 2050 in Accumulator A.
2. **RLC:** Rotate content of accumulator left by 1 bit
3. **RLC:** Rotate content of accumulator left by 1 bit
4. **RLC:** Rotate content of accumulator left by 1 bit
5. **RLC:** Rotate content of accumulator left by 1 bit
6. **STA 3050:** store content of A in memory location 3050.
7. **HLT:** stops executing the program and halts any further execution.

Another Alternative and simple way to execute the above program and the magic of 8085 rotate instructions :

This can be done by using 4 times RRC instruction.

```
LDA 2050H
RRC
RRC            // 4 RRC do same work as 4 RLC. So we can use anyone alternatively.
RRC
RRC           //After this 4th RRC instruction our 8 bit number is reversed and stored in accumulator.
STA 3050H
HLT
```

Step-by-Step and Bit-by-Bit explanation of the Above Program with an example :

Example :

98H in Binary Written as :

1001 1000

RLC 1st Time : 0011 0001 {Carry Flag = 1}

RLC 2nd Time : 0110 0010 {Carry Flag = 0}

RLC 3rd Time : 1100 0100 {Carry Flag = 0}

RLC 4th Time : 1000 1001 { Carry Flag = 1}

Converted Number after 4th RLC : 1000 1001 [89H]

Hence our number is reversed from 98H to 89H.

For Example : 98H in Binary Written as :

1001 1000

RRC 1st Time : 0100 1100 {Carry Flag = 0}

RRC 2nd Time : 0010 0110 {Carry Flag = 0}

RRC 3rd Time : 0001 0011 { Carry Flag = 0}

RRC 4th Time : 1000 1001 { Carry Flag = 1}

Converted Number after 4th RRC : 1000 1001 [89H]

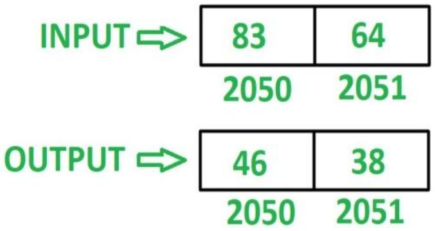
Hence our number is reversed from 98H to 89H.

Hence Instead of 4 RLC, we can also use 4 RRC instructions in our code alternatively.

EXPERIMENT NO. –9

Problem – Write an assembly language program in 8085 microprocessor to reverse 16 bit number.

Example – Assume 16 bit number is stored at memory location 2050 and 2051.



Algorithm –

1. Load contents of memory location 2050 in register L and contents of memory location 2051 in register H
2. Move contents of L in accumulator A
3. Reverse the contents of A by executing **RLC** instruction 4 times
4. Move the contents of A in L
5. Move the contents of H in A
6. Reverse the contents of A by executing **RLC** instruction 4 times
7. Move the contents of L in H
8. Move the contents of A in L
9. Store the content of L in memory location 2050 and contents of H in memory location 2051

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
2000	LHLD 2050	L <- M[2050], H <- M[2051]
2003	MOV A, L	A <- L
2004	RLC	Rotate accumulator content left by 1 bit without carry
2005	RLC	Rotate accumulator content left by 1 bit without carry
2006	RLC	Rotate accumulator content left by 1 bit without carry
2007	RLC	Rotate accumulator content left by 1 bit without carry
2008	MOV L, A	L <- A

2009	MOV A, H	A <- H
200A	RLC	Rotate accumulator content left by 1 bit without carry
200B	RLC	Rotate accumulator content left by 1 bit without carry
200C	RLC	Rotate accumulator content left by 1 bit without carry
200D	RLC	Rotate accumulator content left by 1 bit without carry
200E	MOV H, L	H <- L
200F	MOV L, A	L <- A
2010	SHLD 2050	M[2050] <- L, M[2051] <- H
2013	HLT	END

Explanation – Registers A, H, L are used for general purpose.

1. **LHLD 2050:** loads contents of memory location 2050 in L and 2051 in H.
2. **MOV A, L:** moves content of L in A.
3. **RLC:** shift the content of A left by one bit without carry. Repeat the current instruction 4 times so that contents of A get reversed.
4. **MOV L, A:** moves the content of A in L.
5. **MOV A, H:** moves the content of H in A.
6. **RLC:** shift the content of A left by one bit without carry. Repeat the current instruction 4 times so that contents of A get reversed.
7. **MOV H, L:** moves the content of L in H.
8. **MOV L, A:** moves the content of A in L.
9. **SHLD 2050:** stores the content of L in 2050 and H in 2051.
10. **HLT:** stops executing the program and halts any further execution.

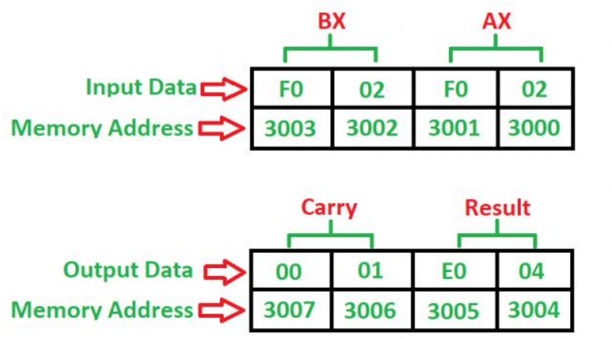
EXPERIMENT NO. –10

Problem – Write a program to add two 16-bit numbers where starting address is **2000** and the numbers are at **3000** and **3002** memory address and store result into **3004** and **3006** memory address.

Example –

Algorithm –

1. Load 0000H into CX register (for carry)
2. Load the data into AX(accumulator) from memory 3000
3. Load the data into BX register from memory 3002
4. Add BX with Accumulator AX
5. Jump if no carry
6. Increment CX by 1
7. Move data from AX(accumulator) to memory 3004
8. Move data from CX register to memory 3006
9. Stop



Input Data	F0	02	F0	02
Memory Address	3003	3002	3001	3000

Output Data	00	01	E0	04
Memory Address	3007	3006	3005	3004

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	CX, 0000	[CX] <- 0000
2003	MOV	AX, [3000]	[AX] <- [3000]
2007	MOV	BX, [3002]	[BX] <- [3002]
200B	ADD	AX, BX	[AX] <- [AX] + [BX]
200D	JNC	2010	Jump if no carry

Memory	Mnemonics	Operands	Comment
200F	INC	CX	[CX] <- [CX] + 1
2010	MOV	[3004], AX	[3004] <- [AX]
2014	MOV	[3006], CX	[3006] <- [CX]
2018	HLT		Stop

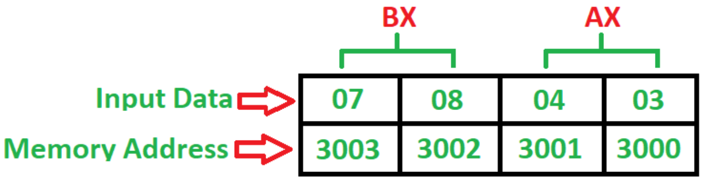
Explanation –

1. **MOV** is used to load and store data.
2. **ADD** is used to add two numbers where their one number is in accumulator or not.
3. **JNC** is a 2-bit command which is used to check whether the carry is generated from accumulator or not.
4. **INC** is used to increment an register by 1.
5. **HLT** is used to stop the program.
6. **AX** is an accumulator which is used to load and store the data.
7. **BX, CX** are general purpose registers where BX is used for storing second number and CX is used to store carry.

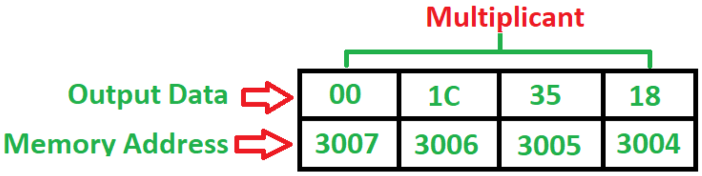
EXPERIMENT NO. –11

Problem – Write a program to multiply two 16-bit numbers where starting address is **2000** and the numbers are at **3000** and **3002** memory address and store result into **3004** and **3006** memory address.

Example –



Input Data	07	08	04	03
Memory Address	3003	3002	3001	3000



Output Data	00	1C	35	18
Memory Address	3007	3006	3005	3004

Algorithm –

1. First load the data into AX(accumulator) from memory 3000
2. Load the data into BX register from memory 3002
3. Multiply BX with Accumulator AX
4. Move data from AX(accumulator) to memory
5. Move data from DX to AX
6. Move data from AX(accumulator) to memory
7. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	AX, [3000]	[AX] <- [3000]
2004	MOV	BX, [3002]	[BX] <- [3002]
2008	MUL	BX	[AX] <- [AX] * [BX]
200A	MOV	[3004], AX	[3004] <- AX
200E	MOV	AX, DX	[AX] <- [DX]

Memory	Mnemonics	Operands	Comment
2010	MOV	[3006], AX	[3006] <- AX
2014	HLT		Stop

Explanation –

1. **MOV** is used to load and store data.
2. **MUL** is used to multiply two 16-bit numbers.
3. **HLT** is used to stop the program.
4. **AX** is an accumulator which is used to store the result.
5. **BX, DX** are general purpose registers where BX is used for multiplication and DX is used for result.

EXPERIMENT NO. –12

Problem – Write a program in 8086 microprocessor to find out the sum of series of even numbers, where numbers are stored from starting offset 500 and store the result at offset 600.

Example –

Input Data ⇨	04	30	15	07	08
Memory Address(offset) ⇨	500	501	502	503	504

Output Data ⇨	38
Memory Address(offset) ⇨	600

Algorithm –

1. Assign 500 to SI
2. Load data from offset SI to register CL (count) and assign 00 to register CH inc. SI by 1
3. Load data from offset SI and apply TEST with 01, if result is non zero jump to step 5
4. Add the offset data with register AL
5. Increase offset by 1
6. LOOP to step 3
7. Store the result (content of register AL) to offset 600
8. Stop

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV SI, 500	SI<-500
403	MOV CL, [SI]	CL<-[SI]
405	INC SI	SI<-SI+1
406	MOV CH, 00	CH<-00
408	MOV AL, 00	AL<-00

MEMORY ADDRESS	MNEMONICS	COMMENT
40A	MOV BL, [SI]	BL<-[SI]
40C	TEST BL, 01	BL AND 01
40F	JNZ 413	JUMP IF NOT ZERO
411	ADD AL, BL	AL<-AL+BL
413	INC SI	SI<-SI+1
414	LOOP 40A	JUMP TO 40A IF CX NOT ZERO
416	MOV [600], AL	AL->[600]
41A	HLT	END

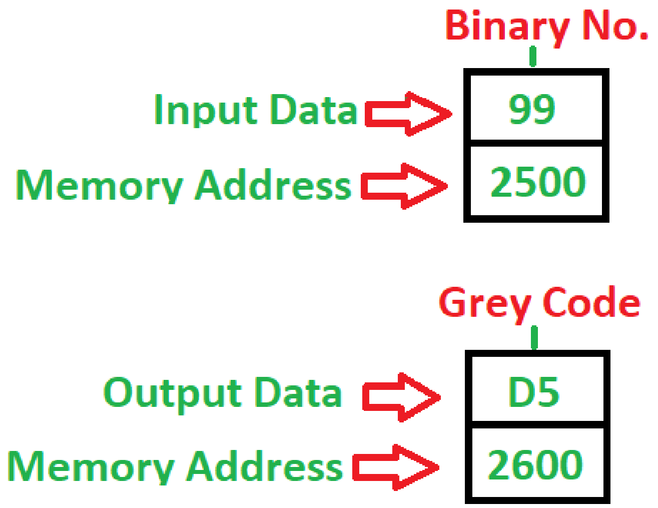
Explanation –

1. **MOV SI, 500:** assign 500 to SI
2. **MOV CL, [SI]:** load data from offset SI to register CL
3. **INC SI:** increase value of SI by 1
4. **MOV CH, 00:** assign 00 to register CH
5. **MOV AL, 00:** assign 00 to register AL
6. **MOV BL, [SI]:** load data from offset SI to register BL
7. **TEST BL, 01:** AND register BL with 01
8. **JNZ 413:** jump to address 413 if not zero
9. **ADD AL, BL:** add contents of register AL and BL
10. **INC SI:** increase value of SI by 1
11. **LOOP 40A:** jump to 40A if CX not zero and CX=CX-1
12. **MOV [600], AL:** store the value of register AL to offset 600
13. **HLT:** end.

EXPERIMENT NO. –13

Problem – Write a program to convert Binary number to Gray code 8-bit number where starting address is **2000** and the number is stored at **2500** memory address and store result into **2600** memory address.

Example –



Algorithm –

1. Move value at [2500] into AL
2. Move AL into BL
3. Logical shift right AL one time
4. XOR BL with AL (Logically) and store into BL
5. Move content of BL into 2600
6. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	AL, [2500]	[AL] <- [2500]
2004	MOV	BL, AL	[BL] <- [AL]
2006	SHR	AL, 01	Shift Right one time
2008	XOR	BL, AL	[BL] <- [BL] @ AL

Memory	Mnemonics	Operands	Comment
200A	MOV	[2600], BL	[2600] <- [BL]
200E	HLT		Stop

Explanation – Registers AL, BL are used for general purpose

1. **MOV** is used to transfer the data
2. **SHR** is used to shift right (logically) up to counter is not zero
3. **XOR** is used to exclusive-or of two values (logically)
4. **HLT** is used to halt the program

EXPERIMENT NO. –14

Problem – Write a program to convert ASCII to BCD 8-bit number where starting address is **2000** and the number is stored at **2050** memory address and store result into **3050** memory address.

Example-
Input : location: 2050
Data : 37
Output : location: 3050
Data : 07

Algorithm –

1. Move value at [2050] into AL
2. Perform AND operation on AL with 0F
3. Move content of accumulator AL into 3050
4. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	AL, [2050]	[AL] <- [2050]
2004	AND	AL, 0F	[AL] <- ([AL] AND 0F)
2006	MOV	[3050], AL	[3050] <- [AL]
200A	HLT		Stop

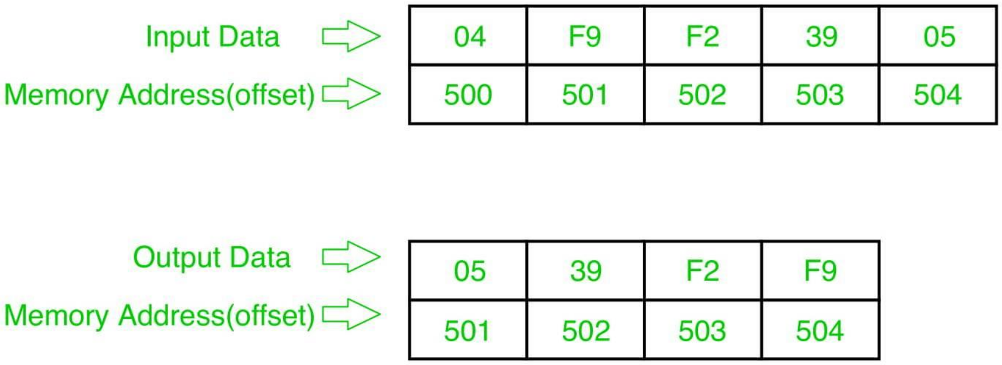
Explanation – Registers AL is used for general purpose

1. **MOV** is used to transfer the data
2. **AND** is used for multiplication (logically)
3. **HLT** is used to halt the program

EXPERIMENT NO. –15

Problem – Write a program in 8086 microprocessor to sort numbers in ascending order in an array of n numbers, where size “n” is stored at memory address 2000 : 500 and the numbers are stored from memory address 2000 : 501.

Example –



Input Data ⇨	04	F9	F2	39	05
Memory Address(offset) ⇨	500	501	502	503	504

Output Data ⇨	05	39	F2	F9
Memory Address(offset) ⇨	501	502	503	504

Example explanation:

Pass-1:
F9 F2 39 05
F2 F9 39 05
F2 39 F9 05
F2 39 05 F9 (1 number got fix)

Pass-2:
F2 39 05 F9
39 F2 05 F9
39 05 F2 F9 (2 number got fix)

Pass-3:
39 05 F2 F9
05 39 F2 F9 (sorted)

Algorithm –

1. Load data from offset 500 to register CL (for count).

2. Travel from starting memory location to last and compare two numbers if first number is greater than second number then swap them.

3. First pass fix the position for last number.

4. Decrease the count by 1.

5. Again travel from starting memory location to (last-1, by help of count) and compare two numbers if first number is greater than second number then swap them.

6. Second pass fix the position for last two numbers.

7. Repeated.

Program –

MEMORY ADDRESS	**MNEMONICS**	**COMMENT**
400	MOV SI, 500	SI<-500
403	MOV CL, [SI]	CL<-[SI]
405	DEC CL	CL<-CL-1
407	MOV SI, 500	SI<-500
40A	MOV CH, [SI]	CH<-[SI]
40C	DEC CH	CH<-CH-1
40E	INC SI	SI<-SI+1
40F	MOV AL, [SI]	AL<-[SI]
411	INC SI	SI<-SI+1
412	CMP AL, [SI]	AL-[SI]

MEMORY ADDRESS	MNEMONICS	COMMENT
414	JC 41C	JUMP TO 41C IF CY=1
416	XCHG AL, [SI]	SWAP AL AND [SI]
418	DEC SI	SI<-SI-1
419	XCHG AL, [SI]	SWAP AL AND [SI]
41B	INC SI	SI<-SI+1
41C	DEC CH	CH<-CH-1
41E	JNZ 40F	JUMP TO 40F IF ZF=0
420	DEC CL	CL<-CL-1
422	JNZ 407	JUMP TO 407 IF ZF=0
424	HLT	END

Explanation –

1. **MOV SI, 500**: set the value of SI to 500.
2. **MOV CL, [SI]**: load data from offset SI to register CL.
3. **DEC CL**: decrease value of register CL BY 1.
4. **MOV SI, 500**: set the value of SI to 500.
5. **MOV CH, [SI]**: load data from offset SI to register CH.
6. **DEC CH**: decrease value of register CH BY 1.
7. **INC SI**: increase value of SI BY 1.

8. **MOV AL, [SI]**: load value from offset SI to register AL.

9. **INC SI**: increase value of SI BY 1.

10. **CMP AL, [SI]**: compares value of register AL and [SI] (AL-[SI]).

11. **JC 41C**: jump to address 41C if carry generated.

12. **XCHG AL, [SI]**: exchange the contents of register AL and SI.

13. **DEC SI**: decrease value of SI by 1.

14. **XCHG AL, [SI]**: exchange the contents of register AL and SI.

15. **INC SI**: increase value of SI by 1.

16. **DEC CH**: decrease value of register CH by 1.

17. **JNZ 40F**: jump to address 40F if zero flat reset.

18. **DEC CL**: decrease value of register CL by 1.

19. **JNZ 407**: jump to address 407 if zero flat reset.

20. **HLT**: stop.

EXPERIMENT NO. –16

Problem: Write an assembly level program to print a given string .

Note:

The program cannot be run on an online editor, please use MASM to run the program and use dos box to run MASM, you might use any 8086 emulator to run the program.

Explanation:

1. Create a string
2. Load the effective address of the string in dx using LEA command
3. Print the string by calling the interrupt with 9H in AH
4. The string must be terminated by '$' sign

Program

```
.MODEL SMALL
.STACK 100H
.DATA
;The string to be printed
STRING DB 'This is a sample string', '$'
.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
; load address of the string
LEA DX,STRING
;output the string
;loaded in dx
MOV AH,09H
INT 21H
;interrupt to exit
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN
```

EXPERIMENT NO. –17

Problem – Write an assembly language program in 8086 to transfer a string from one memory location to another in assembly language (Copy String).

```
READ MACRO MSG
   MOV AH,0AH
   LEA DX,MSG
   INT 21H
ENDM

PRINT MACRO MSG
   MOV AH,09H
   LEA DX,MSG
   INT 21H
ENDM

DATA SEGMENT
   CR EQU 0DH
   LF EQU 0AH
   MSG1 DB "ENTER STRING:$"
   MSG2 DB "CONTENTS ARE TRANSFERRED $"
   BUFF DB 255
          DB 0
          DB 255 DUP ('$')
   DEST DB 255 DUP ('$')
DATA ENDS

CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
      MOV DS,AX
      PRINT MSG1
      READ BUFF
      MOV SI,OFFSET BUFF+2
      MOV DI,OFFSET BUFF
      MOV CL,BYTEPTR[SI+1]
      MOV CH,00H
      CLD

REPEAT: MOVSB
      PRINT MSG2
      PRINT DEST
      MOV AH,4CH
      INT 21H
```

```
CODE ENDS

END START
```

EXPERIMENT NO. –18

Problem Write ALP to read a text from keyboard, convert into uppercase and display on cleared screen.

```
.model small
.stack 100
.data
String label byte
Maxlen db 20
Actlen db ?
Str db 20 dup(?)
.code
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX
LEA DX, STRING
MOV AH, 0AH
INT 21H
CALL CLR_SCR
MOV CL, ACTLEN
MOV CH, 00H
MOV BX, 00H
STEP: MOV DL, STR[BX]
CMP DL, 'a'
JB N1
CMP DL, 'z'
JA N1
SUB DL, 20H
MOV AH, 02H
N1: INT 21H
INC BX
LOOP STEP
MOV AX, 4C00H
INT 21H
MAIN ENDP
CLR_SCR PROC NEAR
MOV AH, 0600H
MOV BH, 07H
MOV CX, 0000H
MOV DX, 184FH
INT 10H
RET
CLR_SCR ENDP
END MAIN
```

EXPERIMENT NO. –19

Problem Write ALP 8086 program to reverse a string.
Explanation:

1. Create a string
2. Traverse through the string
3. Push the characters in the stack
4. Count the number of characters
5. Load the starting address of the string
6. POP the top character of the stack until count is not equal to zero
7. Put the character and reduce the count and increase the address
8. Continue until the count is greater than zero
9. Load the effective address of the string in dx using LEA command
10. Print the string by calling the interrupt with 9H in AH
11. The string must be terminated by ‘$’ sign

```
.MODEL SMALL
.STACK 100H
.DATA

; The string to be printed
STRING DB 'This is a sample string', '$'

.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX

; call reverse function
CALL REVERSE

; load address of the string
LEA DX,STRING

; output the string
; loaded in dx
MOV AH, 09H
INT 21H

; interrupt to exit
MOV AH, 4CH
INT 21H
```

```
MAIN ENDP
REVERSE PROC
        ; load the offset of
        ; the string
        MOV SI, OFFSET STRING

        ; count of characters of the;
        ;string
        MOV CX, 0H

        LOOP1:
        ; compare if this is;
        ;the last character
        MOV AX, [SI]
        CMP AL, '$'
        JE LABEL1

        ; else push it in the;
        ;stack
        PUSH [SI]

        ; increment the pointer;
        ;and count
        INC SI
        INC CX

        JMP LOOP1

        LABEL1:
        ; again load the starting;
        ;address of the string
        MOV SI, OFFSET STRING

                LOOP2:
                ;if count not equal to zero
                CMP CX,0
                JE EXIT

                ; pop the top of stack
                POP DX

                ; make dh, 0
                XOR DH, DH
        ; put the character of the;
        ;reversed string
        MOV [SI], DX

        ; increment si and;
        ;decrement count
        INC SI
        DEC CX

        JMP LOOP2


    EXIT:
    ; add $ to the end of string
    MOV [SI],'$ '
    RET

REVERSE ENDP
END MAIN
```

EXPERIMENT NO. –20

Problem: Write an 8086 program to check whether a given string is palindrome or not.
Explanation:

1. Create a string
2. Traverse to the end of the string
3. Get the address of the end of the string, DI
4. Load the starting address of the string, SI
5. Compare the value stored at the address
6. Increment the pointer, SI
7. Decrements the pointer, DI
8. Compare again the value stored at si and di
9. Repeat the steps until SI<=DI
10. If all the characters match print string is palindrome else print not palindrome

Program

```
.MODEL SMALL
.STACK 100H
.DATA

; The string to be printed
STRING DB 'abba', '$'
STRING1 DB 'String is palindrome', '$'
STRING2 DB 'String is not palindrome', '$'

.CODE
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX

; check if the string is;
;palindrome or not
CALL Palindrome

;interrupt to exit
MOV AH, 4CH
INT 21H
MAIN ENDP
Palindrome PROC

; load the starting address
; of the string
```

```
MOV SI,OFFSET STRING

; traverse to the end of;
;the string
LOOP1 :
    MOV AX, [SI]
    CMP AL, '$'
    JE LABEL1
    INC SI
    JMP LOOP1

;load the starting address;
;of the string
LABEL1 :
    MOV DI,OFFSET STRING
    DEC SI

    ; check if the string is palindrome;
    ;or not
    LOOP2 :
    CMP SI, DI
    JL OUTPUT1
    MOV AX,[SI]
    MOV BX, [DI]
    CMP AL, BL
    JNE OUTPUT2

    DEC SI
    INC DI
    JMP LOOP2

OUTPUT1:
    ;load address of the string
    LEA DX,STRING1

    ; output the string;
    ;loaded in dx
    MOV AH, 09H
    INT 21H
    RET

OUTPUT2:
    ;load address of the string
    LEA DX,STRING2
```

```
        ; output the string
        ; loaded in dx
        MOV AH,09H
        INT 21H
        RET

Palindrome ENDP
END MAIN
```