Laboratory Report | 2020

Embedded System Project
(EET 4100)

Name: ANSHUMAN DASH.....

Registration No:1741016004.....

Branch: ECE.....Section: E.....

Department of Electronics and Communication Engineering
Institute Of Technical Education & Research (Faculty Of Engineering)
Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha

CONTENTS

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 01

Movement of data between register, memory and stack using 8051

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I. Objectives

1.Write an Assembly language program (ALP) using 8051 to store 5 different 8-bit numbers in ROM memory locations starting from 40H onwards.

2.Write an ALP using 8051 to read two 8-bit numbers stored in 40H and 41H, add them and store the result in 45H.

3.Write an ALP using 8051 to use registers of bank 3, and load the same value 05H in the registers R0 to R3.

4.Write an ALP using 8051 to store the R6 and R7 register of Bank 2 to the stack.

5.Write an ALP using 8051 to store the data 10H and 20H in memory address 40H and 41H, respectively. After that using stack swap the contents of memory address 40H and 41H.

6.Write an ALP using 8051 to store 01H in R0 of Bank 0, 02H in R0 of Bank 1, 03H in R0 of Bank 2 and 04H in R0 of Bank 3.

II. Lab Component

Requirements:

1) PC
2) EDSim51DI or Keil μVision software

Theory :

The MOV instruction copies data from one location to another. This instruction tells the CPU to move (in reality, COPY) the source operand to the destination operand.

SYNTAX:- MOV destination, source; copy source to dest.

The ADD instruction tells the CPU to add the source byte to register A and put the result in register A.

III. Programs& Outputs

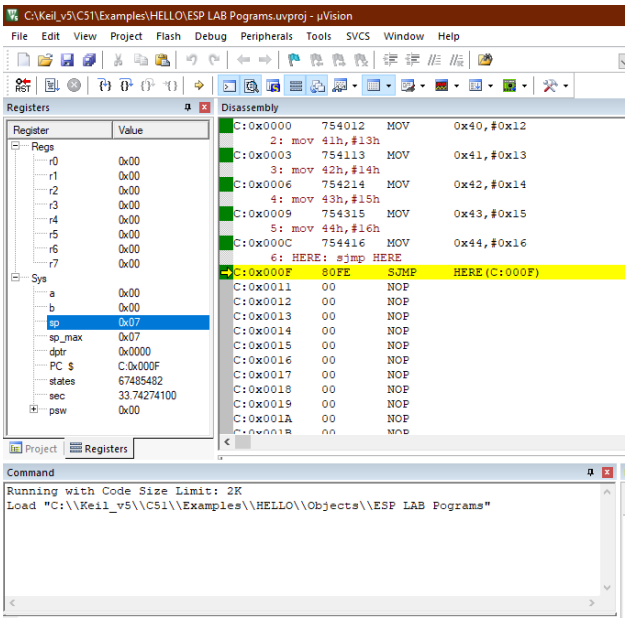
Complete the following using EDSim51DI/ Keil μVision software

(1)

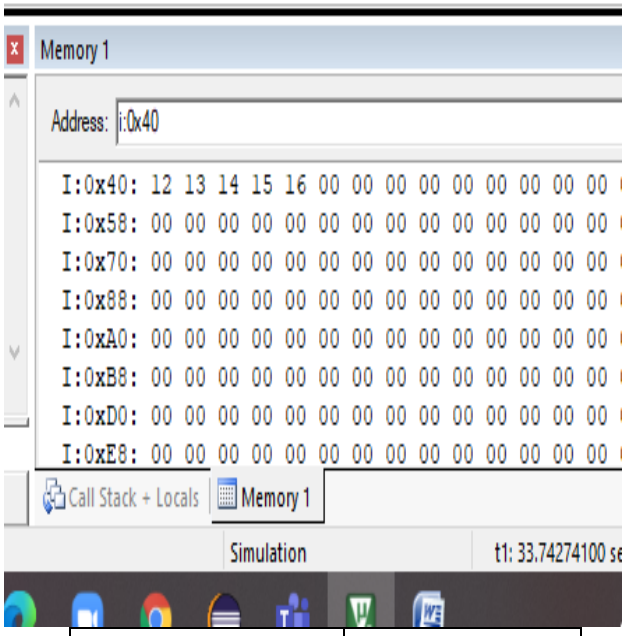
Memory Location	Label	Opcode	Mnemonics	Comments
0000H		75	MOV 41H,#13H	Moves data 13 to memory location 41H
0002H		75	MOV42H,#14H	Moves data 14 to memory location 42H
0004H		75	MOV 43H,#15H	Moves data 15 to memory location 43H
0006H		75	MOV44H,#16H	Moves data 16 to memory location 44H
0008H	HERE	80	SJMP HERE	Short jump to 0008H
000AH			END	

RESULTS:-

Memory Location	Input Data
41H	13H
42H	14H
43H	15H
44H	16H



Memory Location	Output Data
41H	13H
42H	14H
43H	15H
44H	16H



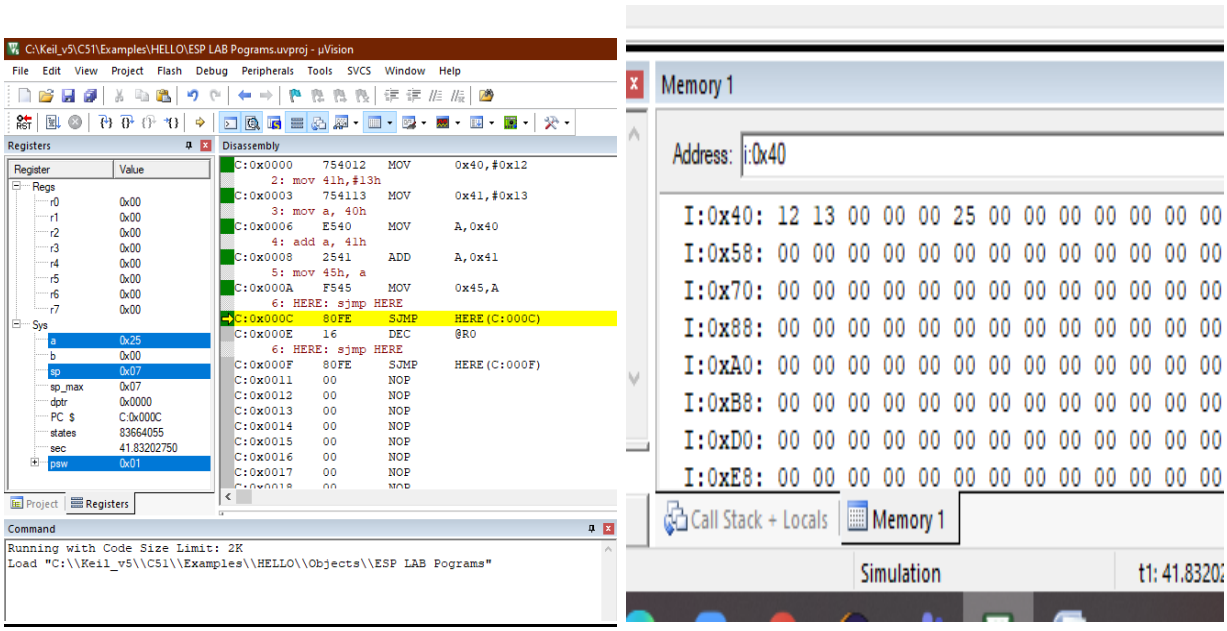
(2)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		75	MOV 40H,#12H	Moves data 12 to memory location 40h
0002H		75	MOV 41H,#13H	Moves data 13 to memory location 42h
0004H		E5	MOV A, 40H	Moves data of 40hto register A
0006H		25	ADD A, 41H	Add A with data of 45h
0008H		F5	MOV 45H, A	Moves data of A to memory location 45h
000AH	HERE	80	SJMP HERE	Short jump to 0008
000CH			END	

RESULTS:-

Memory Location	Input Data
40H	12H
41H	13H

Memory Location	Output Data
45H	25H



(3)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		D2	SETB PSW.4	Set PSW.4
0002H		D2	SETB PSW.3	Set PSW.3
0004H		78	MOV R0, #05H	Moves data 12 to memory location 40h
0008H		79	MOV R1, #05H	Moves data 13 to memory location 42h
0008H		7A	MOV R2, #05H	Moves data of 40h to register A
0008H		7B	MOV R3, #05H	Moves data of A to memory location 45h
000AH	HERE	80	SJMP HERE	Short jump to 0008
000CH			END	

RESULTS:-

Register	Input Data
R0, R1	05H
R2,R3	05H

Register	Output Data
R0,R1	05H
R2,R3	05H

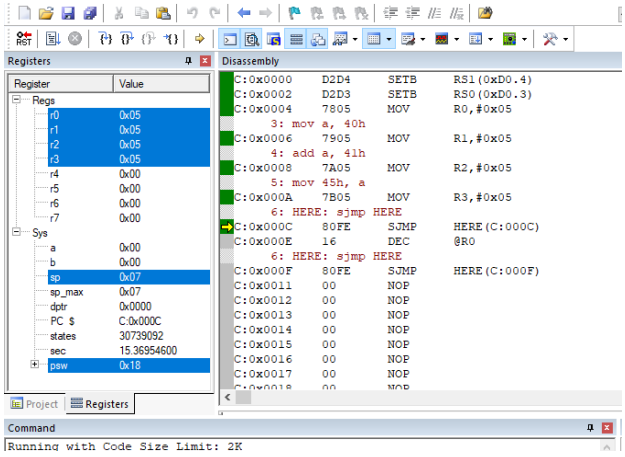
(4)

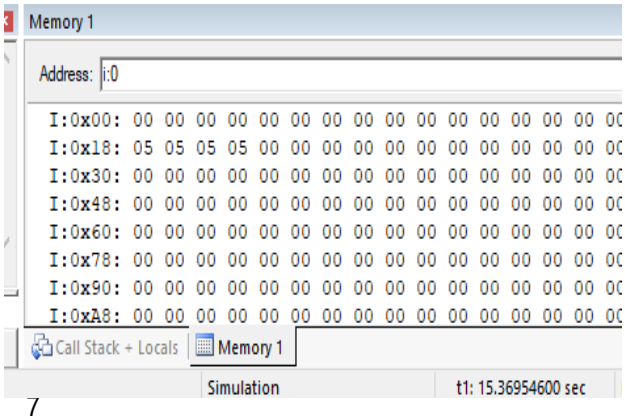
Memory Location	Label	Opcode	Mnemonics	Comments
0000H		D2	SETB PSW.4	Set PSW.4
0002H		C2	CLR PSW.3	Clear PSW.3
0004H		7E	MOV R6, #05H	Moves data of 05h to register R6
0006H		7F	MOV R7, #A0H	Moves data of 0Ah to register R7
0008H	HERE	80	SJMP HERE	Short jump to 0008
000AH			END	

RESULTS:-

Register	Input Data
R6	05H
R7	0AH

Register	Output Data
R6	05H
R7	0AH

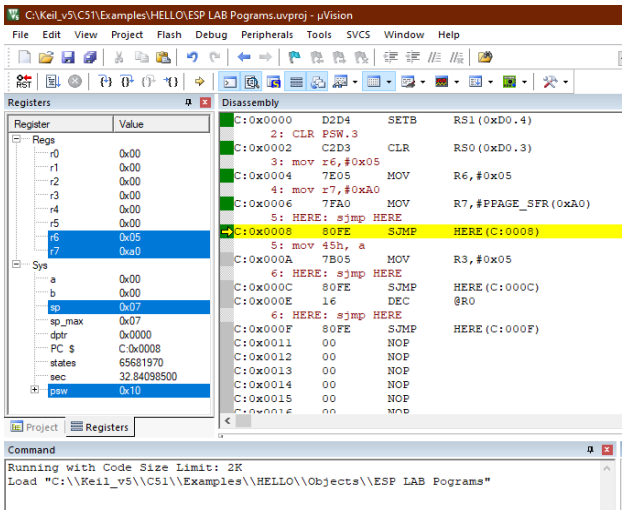




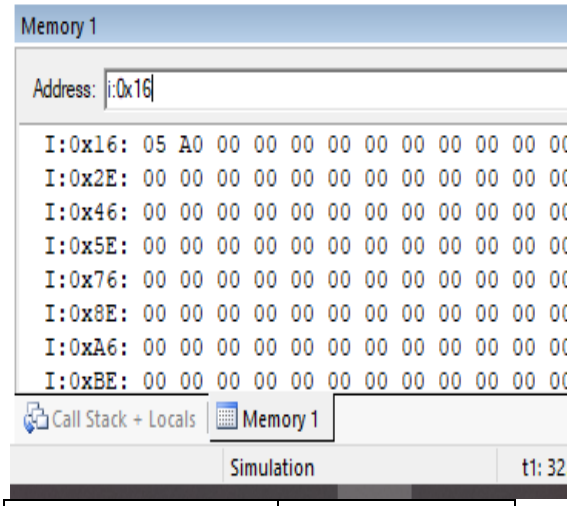
(5)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		75	MOV 40H,#12H	Moves data of 12H to memory location 40H
0002H		75	MOV 41H,#FAH	Moves data of FAH to memory location 41H
0004H		C0	PUSH 40H	Push 40H
0006H		C0	PUSH 41H	Push 41H
0008H		D0	POP 40H	Pop 40H
000A		D0	POP 41H	Pop 41H
000CH	HERE	80	SJMP HERE	Short jump to 0008
000EH			END	

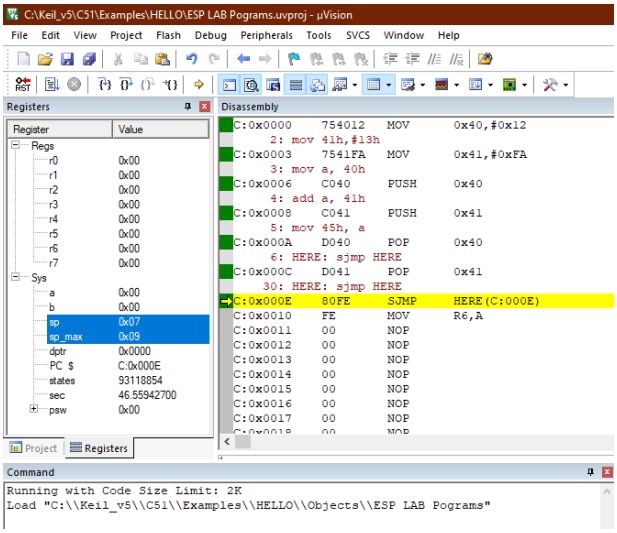
RESULTS:-

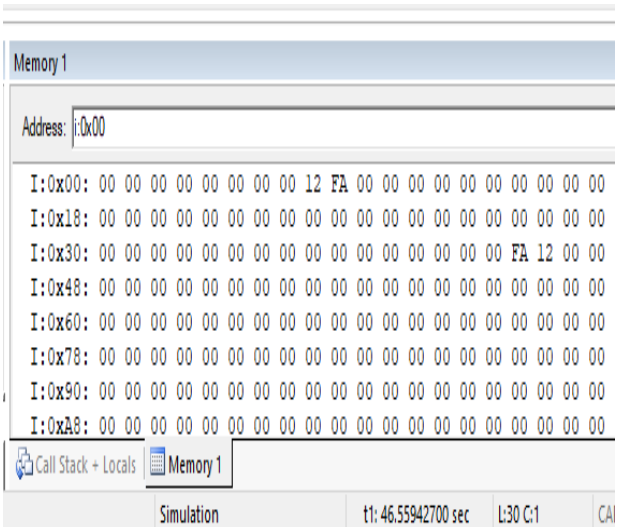


Register	Input Data
R6	05H
R7	0AH



Register	Output Data
R6	05H
R7	0AH





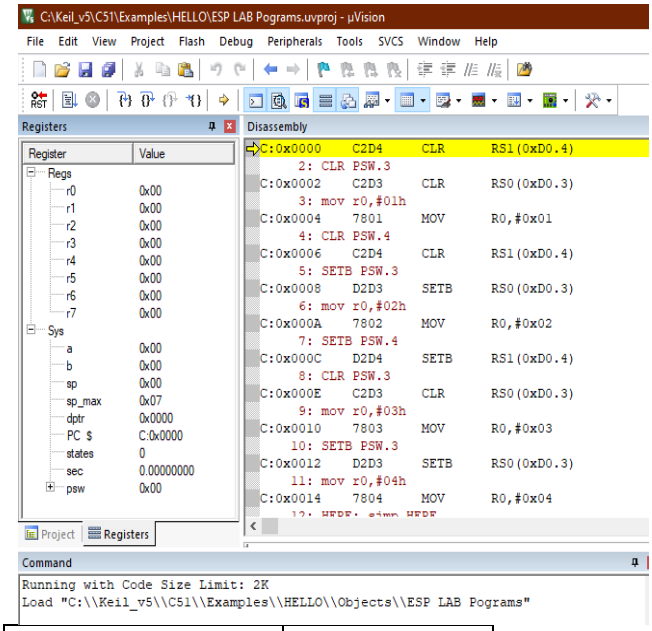
(6)

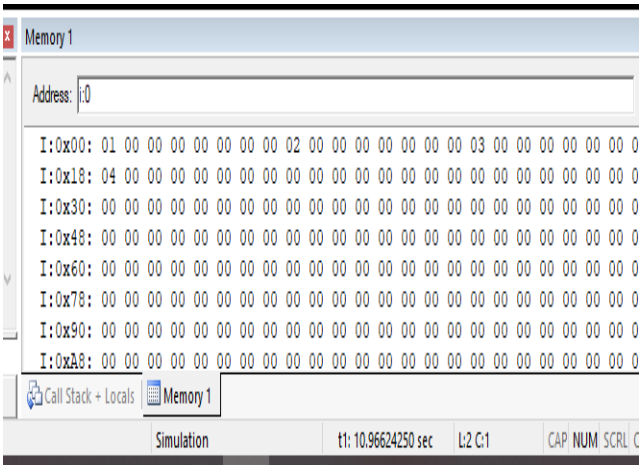
Memory Location	Label	Opcode	Mnemonics	Comments
0000H		C2	CLR PSW.4	Clear PSW.4
0002H		C2	CLR PSW.3	Clear PSW.3
0004H		78	MOV R0, #01H	Moves data 01h to R0 of bank0
0006H		C2	CLR PSW.4	Clear PSW.4
0008H		D2	SETB PSW.3	Set PSW.3
000AH		78	MOV R0, #02H	Moves data 02h to R0 of bank1
000CH		D2	SETB PSW.4	Set PSW.4
000EH		C2	CLR PSW.3	Clear PSW.3
0010H		78	MOV R0, #03H	Moves data 03h to R0 of bank2
0012H		D2	SETB PSW.3	Set PSW.3
0014H		78	MOV R0, #04H	Moves data 04h to R0 of bank3
0016H	HERE	80	SJMP HERE	Short jump to 0008
0018H			END	

RESULTS:-

Register	Input Data
R0	01H

Register	Output Data
R0	01H





R0	02H
R0	03H
R0	04h

R0	02H
R0	03H
R0	04H

IV-Conclusion

Thus, the entire programs for movement of data between register, memory and stack using 8051 were executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 02

Branching and Iterative Programming using 8051

Department of Electronics & Communication Engineering

Siksha 'O' AnusandhanDeemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1.Write an Assembly language program using 8051 to find the number of occurrences of a specific data in an array of numbers at the internal memory locations 40H-4FH. Assume the data to be searched is C0H and store the result in a memory location.

2.Write an Assembly language program using 8051 to reverse the bits in a byte.

3.Write an Assembly language program using 8051 to check whether given number is palindrome or not. If palindrome store FFH in accumulator else store 00H in accumulator.

4.Write an Assembly language program using 8051 to count number of ones and zeros in an eight-bit number.

5.Write an Assembly language program using 8051 to complement the content of accumulator 1217 times.

6.Write an Assembly language program using 8051 to fill a block of memory in internal RAM with a specific data.

II.Lab Component

Requirements:

3) PC
4) EDSim51DI or Keil μVision software

III.Programs& Outputs

Complete the following using EDSim51DI/ Keil μVision software

(All the programs must be written in the following table.)

1.

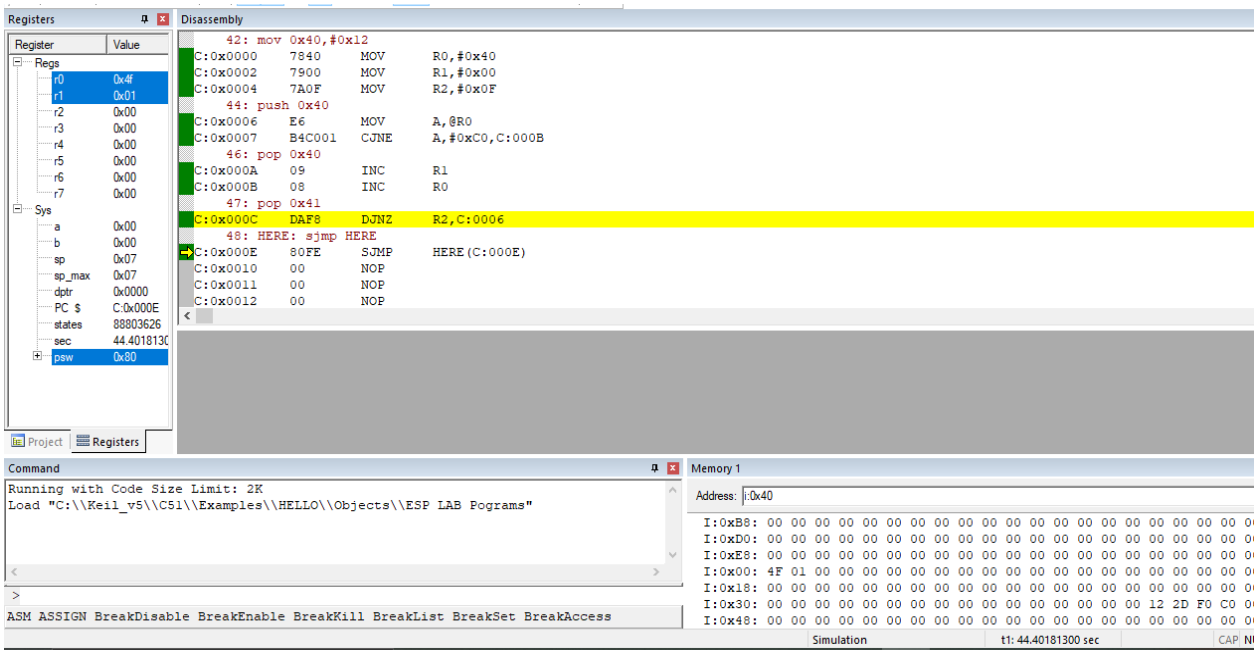
Memory Location	Label	Opcode	Mnemonics	Comments
0000H			ORG 000H	Origin
0001H		78	MOV R0, #40H	Moves data 40H to R0
0003H		79	MOV R1, #00H	Moves data 00H to R1
0005H		7A	MOV R2, #0FH	Moves data 0FH to R2
0007H	AGAIN:	E6	MOV A, @R0	Moves data of R0 to A
0009H		B4	CJNE A, #0CH, NEXT	Compare and jump if A not equal to 0CH
000CH		09	INC R1	Increment R1
000DH	NEXT:	08	INC R0	Increment R0

000EH		DA	DJNZ R2, AGAIN	Decrement and jump if R2 not zero
0010H	HERE:	80	SJMP HERE	Short jump to here
0012H			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
R0	C0H

Memory Location	Output Data
R1	01H



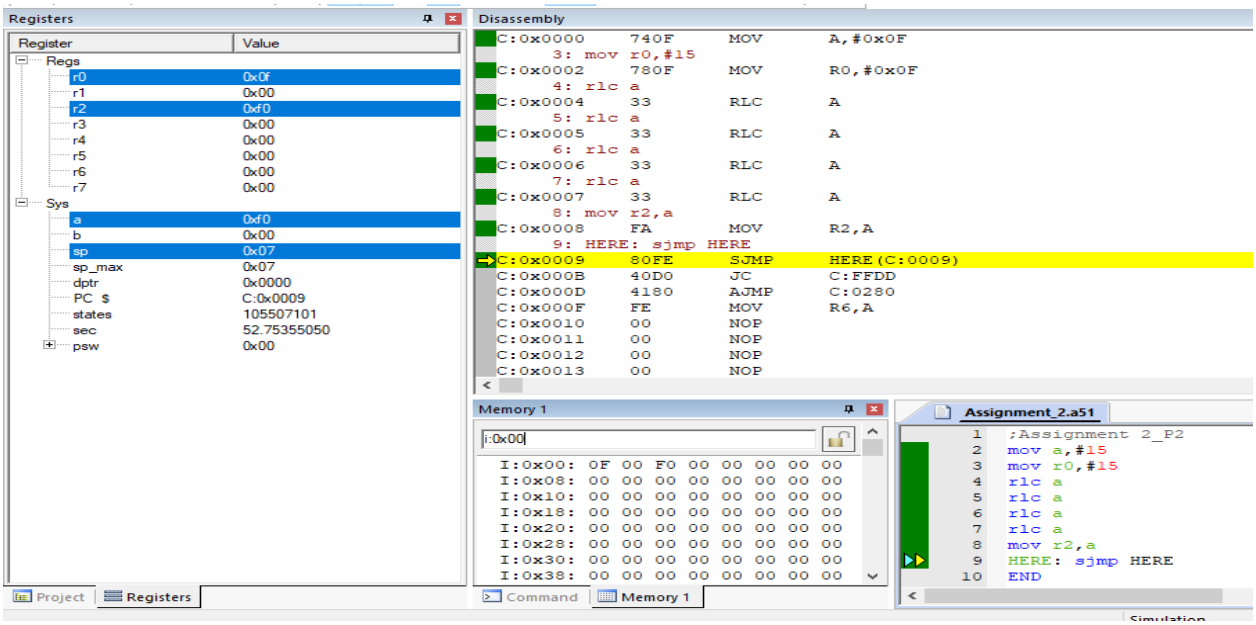
2.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		74	MOV A, #15H	Moves data 15H to A
0002H		78	MOV R0, #15H	Moves data 15H to R0
0004H		33	RLC A	Rotate left data of A throughcarry
0005H		33	RLC A	Rotate left data of A through carry
0006H		33	RLC A	Rotate left data of A through carry
0007H		33	RLC A	Rotate left data of A through carry
0008H		FA	MOV R2,A	Moves data of A to R2
000AH	HERE	80	SJMP HERE	Short jump to here
000CH			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
0000H	0FH

Memory Location	Output Data
0002H	F0H



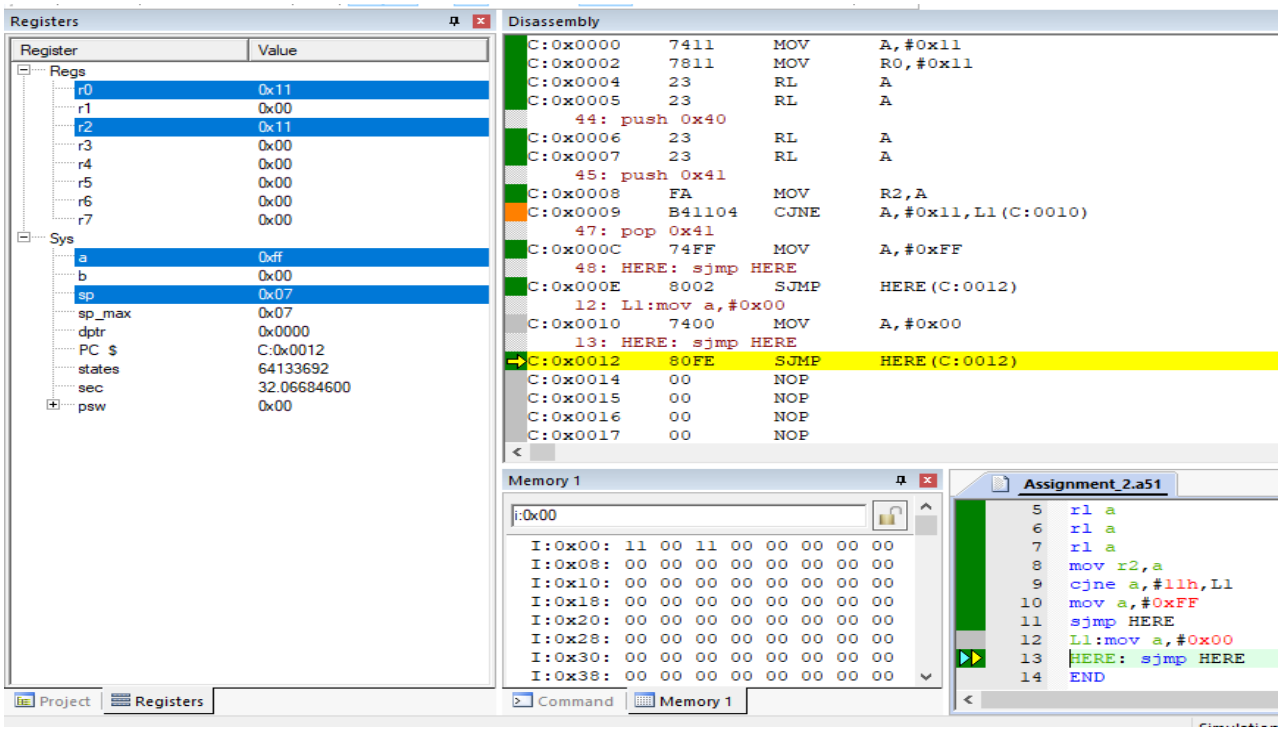
3.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		74	MOV A, #20H	Moves 20H to A
0002H		78	MOV R0, #20H	Moves 20H to R0
0004H		23	RL A	Rotate left A
0005H		23	RL A	Rotate left A
0006H		23	RL A	Rotate left A
0007H		23	RL A	Rotate left A
0008H		FA	MOV R2,A	Moves data of A to R2
000AH		B4	CJNE A,#20H,L1	Compare and jump if A not equal to
000DH		74	MOV A,#0XFF	Moves FFH to A
000FH		80	SJMP HERE	Short jump to here
0011H	L1	74	MOV A,#0x00H	Moves 00H to A
0013H	HERE	80	SJMP HERE	Short jump to here
0015H			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
0000H	11H

Memory Location	Output Data
0002H	01H



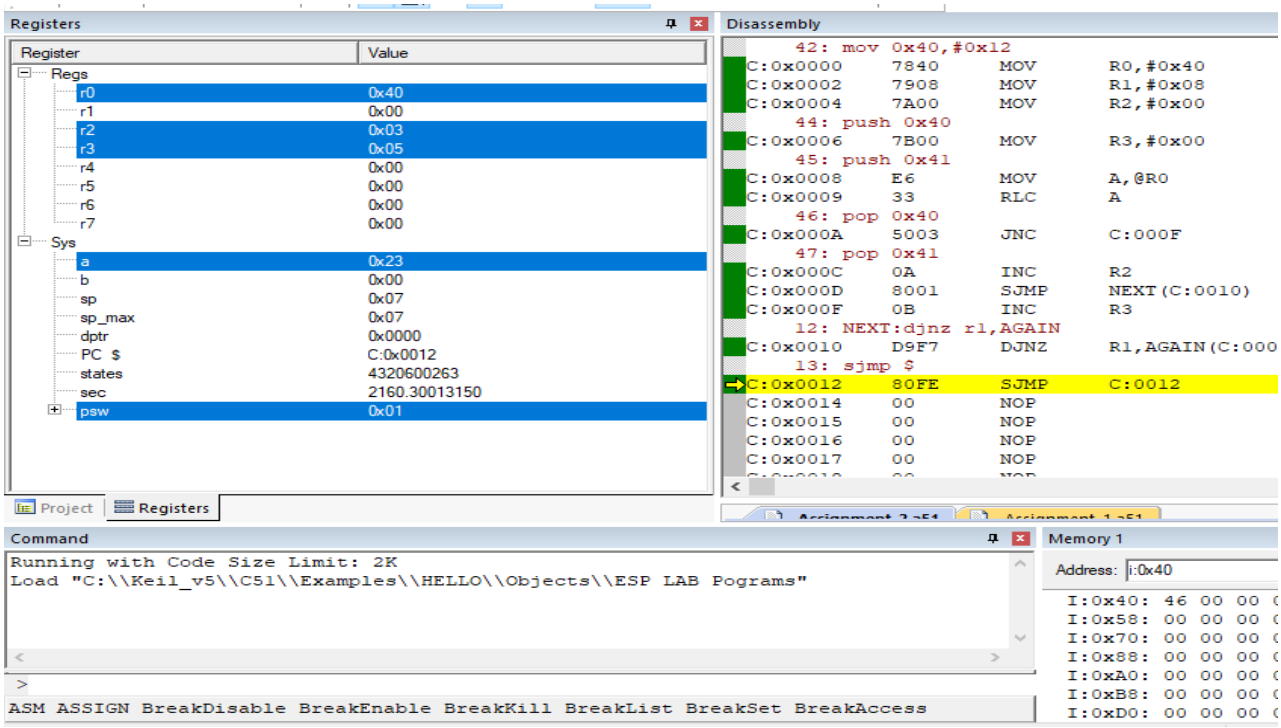
4.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		78	MOV R0,#0x40	Moves data 40H to R0
0002H		79	MOV R1, #0x08	Moves data 08H to R1
0004H		7A	MOV R2, #0x00	Moves data 00H to R2
0006H		7B	MOV R3, #0x00	Moves data 00H to R3
0008H		E6	MOV A, @R0	Moves data of R0 to A
000AH	AGAIN:	33	RLC A	Rotate A left through carry
000BH		50	JNC NEXTT	Jump if not carry
000DH		0A	INC R2	Increment carry
000EH		80	SJMP NEXT	Short jump to here
0010H	NEXTT:	0B	INC R3	Increment R3
0011H	NEXT:	D9	DJNZ R1, AGAIN	Decrement and jump if not equal to zero
0013H	HERE:	80	SJMP HERE	Short jump to here
0015H			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
4000H	46H

Memory Location	Output Data
40002H	01H



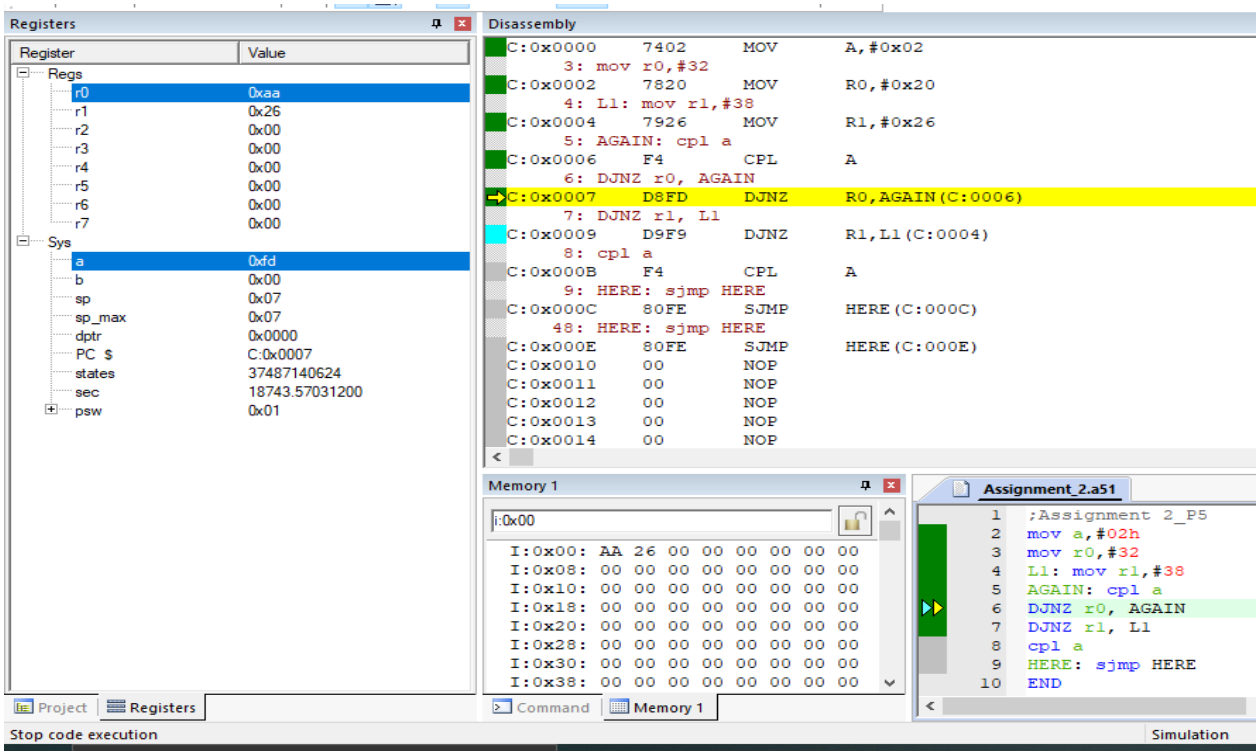
5.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		74	MOV A, #02H	Moves data 02H to A
0002H		78	MOV R0, #32H	Moves data 32H to R2
0004H	L1:	79	MOV R1, #38H	Moves data 38H to R3
0006H	AGAIN:	F4	CPL A	Complements A
0007H		D8	DJNZ R0, AGAIN	Decrement and jump if not equal to zero
0009H		D9	DJNZ R1, L1	Decrement and jump if not equal to zero
000BH		F4	CPL A	Complements A
000CH	HERE:	80	SJMP HERE	Short jump to here
000EH			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
0000H	AAH

Memory Location	Output Data
0001H	26H



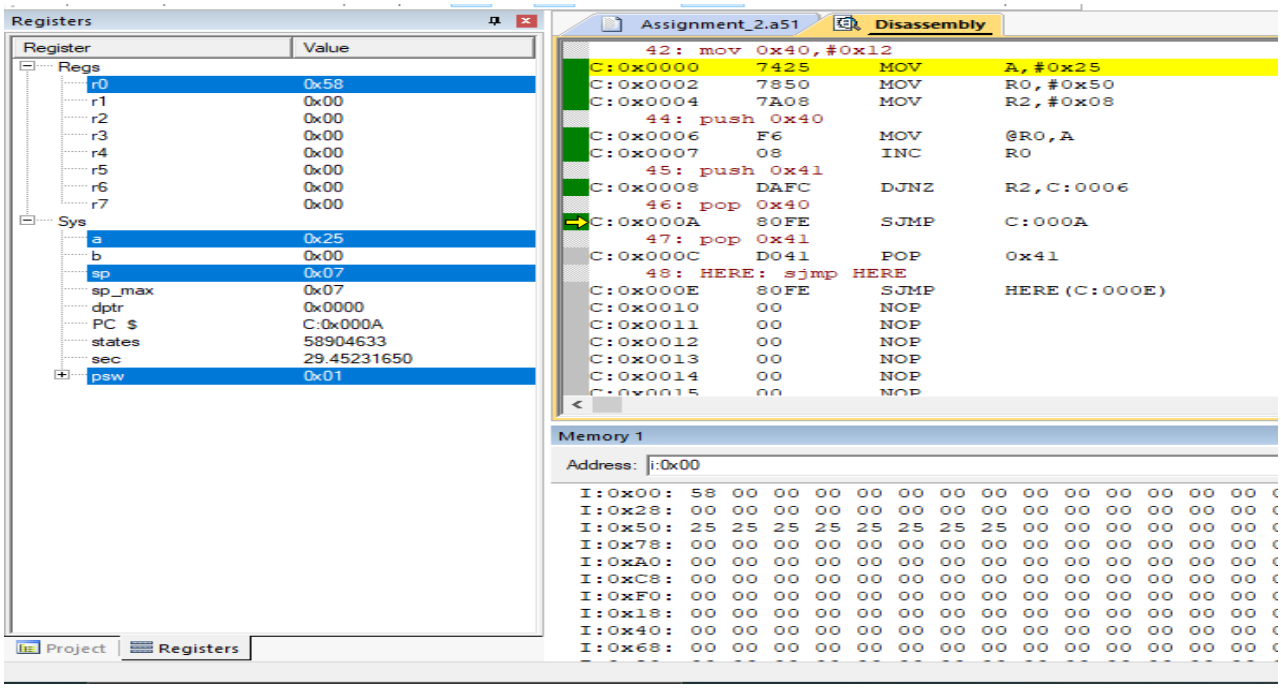
6.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		74	MOV A, #25H	Moves data 25H to A
0002H		78	MOV R0, #50H	Moves data 50H to R0
0004H		7A	MOV R2, #08H	Moves data 08H to R2
0006H	AGAIN:	F6	MOV @R0, A	Moves data of A to R0
0008H		08	INC R0	Increment R0
0009H		DA	DJNZ R2, AGAIN	Decrement and jump if not equal to zero
000BH		80	SJMP $	Short jump to here
000DH			END	

The results should be mentioned in a table like the following.

Memory Location	Input Data
0000H	58H

Memory Location	Output Data
0050H-0057H	25H



IV. Conclusion

Thus, the entire programs for movement of data between register, memory and stack using 8051 were executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 03

8051 I/O Port Programming

Department of Electronics & Communication Engineering

Siksha 'O' AnusandhanDeemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1. Write an ALP to toggle the pin P1.7 continuously using 8051 instructions.

2. Write a program to perform the following

(a) Keep monitoring the P1.2 bit until it becomes high.

(b) When P1.2 becomes high, write value 45H to port 0.

(c) Send a high-to-low (H-to-L) pulse to P2.3.

3. A switch is connected to P1.7. Write a program to check the status of SW and perform the following,

(a) If SW=0, send letter 'N' to P2

(b) If SW=1, send letter 'Y' to P2

Use the carry flag to check the switch status.

4.A switch is connected to pin P1.0 and an LED to pin P2.7. Write a program to get the status of the switch and send it to the LED.

5.Write an ALP to create a square wave of 50 percent duty cycle on bit 4 of port 1.

6.Write an ALP to get 8-bit data from P1 and send it to all other ports P0, P2 and P3.

II.Lab Component

Requirements:

5) PC
6) EDSim51DI or Keil μVision software

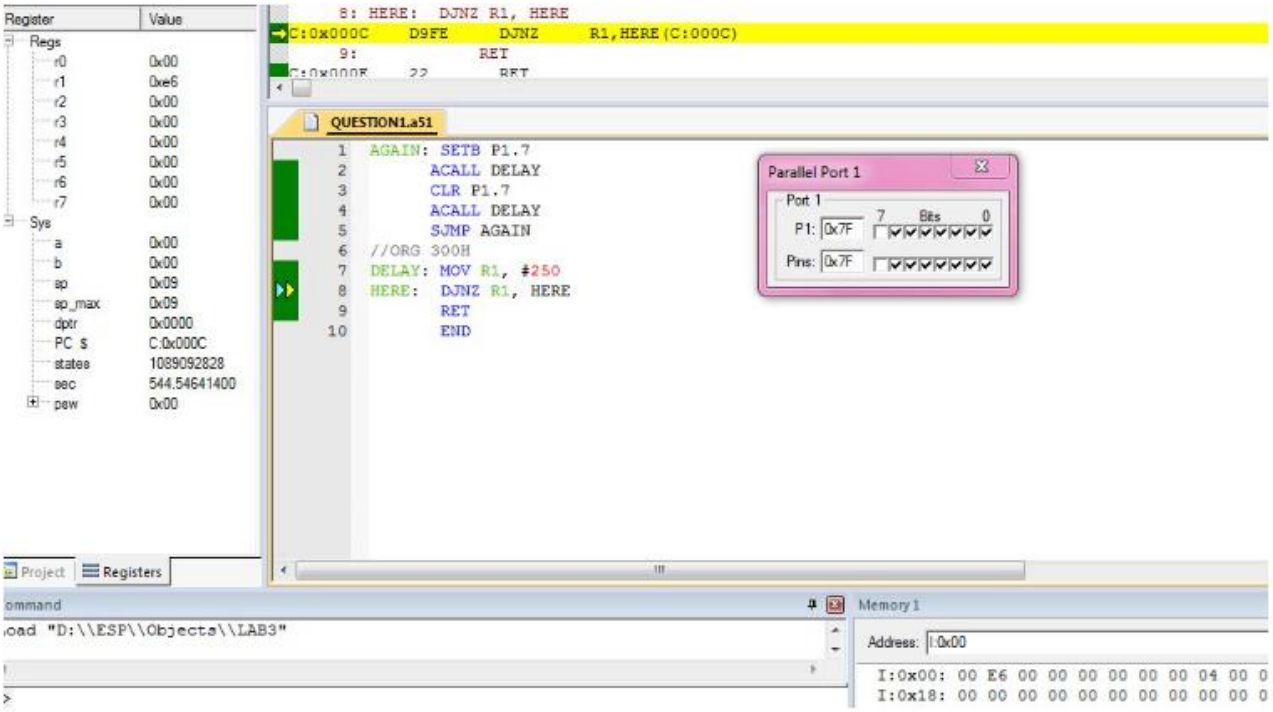
III.Programs& Outputs

1.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H	AGAIN:	D2	SETB P1.7	Sets the bit 7 of port 1 to value of 1
0002H		11	ACALL DELAY	Absolute call to DELAY subroutine
0004H		C2	CLR P1.7	Clear the bit 7 of port 1
0006H		31	ACALL DELAY	Absolute call to DELAY subroutine
0008H		80	SJMP AGAIN	Short Jump to AGAIN
000AH	DELAY:	79	MOV R1, #250	Move data 250 to R1
000CH	HERE:	D9	DJNZ R1, HERE	Decrement R1 and jump to HERE if not zero

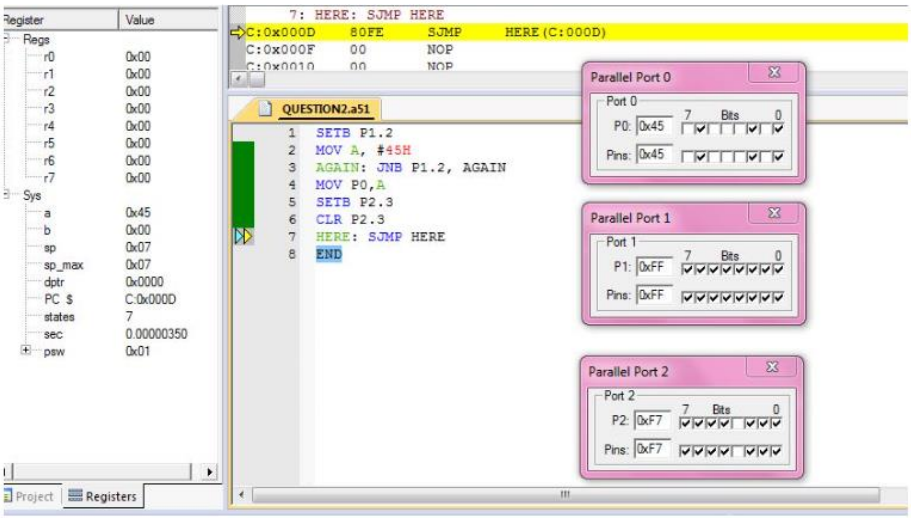
000EH		22	RET	Return to main routine
000FH			END	End

Output:



2.

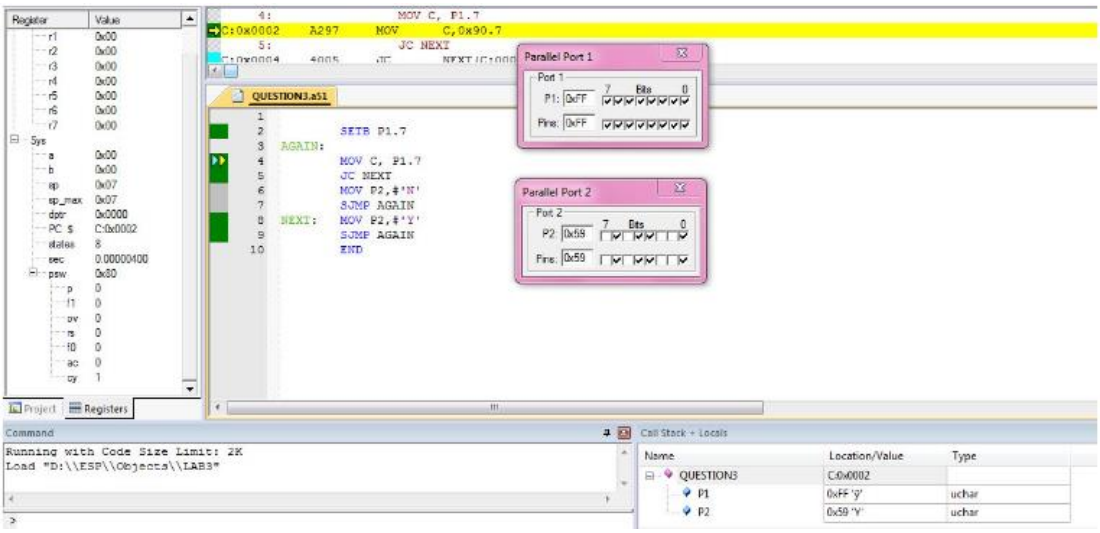
Memory Location	Label	Opcode	Mnemonics	Comments
0000H		D2	SETB P1.2	Sets the bit 2 of port 1 to value of 1
0002H		74	MOV A, #45H	Move data 45 to A
0004H	AGAIN:	30	JNB P1.2, AGAIN	Jump to AGAIN if bit is not set
0007H		92	MOV P0, A	Move value of A to P0
0009H		D2	SETB P2.3	Sets the bit 3 of port 2 to value of 1
000BH		C2	CLR P2.3	Clear the bit 3 of port 2
000DH	HERE:	80	SJMP HERE	Short jump to HERE
000FH			END	End



Output:

3.

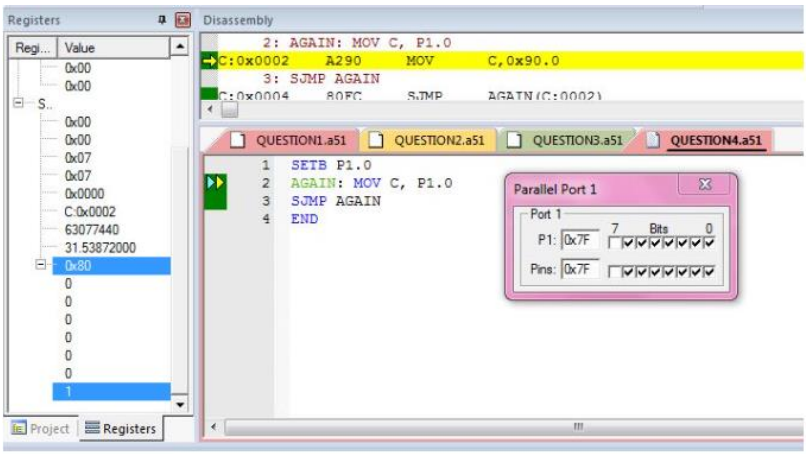
Memory Location	Label	Opcode	Mnemonics	Comments
0000H		D2	SETB P1.7	Sets the bit 7 of port 1 to value of 1
0002H	AGAIN:	A2	MOV C, P1.7	Move the value of P1.7 to C
0004H		40	JC NEXT	Jump to NEXT if carry flag is set
0006H		92	MOV P2, #'N'	Move the data 'N' to P2
0008H		80	SJMP AGAIN	Short jump to AGAIN
000AH	NEXT:	92	MOV P2, #'Y'	Move the data 'Y' to P2
000CH		80	SJMP AGAIN	Short jump to AGAIN
000EH			END	End



Output:

4.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		D2	SETB P1.0	Sets the bit 0 of port 1 to value of 1
0002H	AGAIN:	A2	MOV C, P1.0	Move the value of P1.0 to C
0004H		80	SJMP AGAIN	Short jump to AGAIN
0006H			END	End

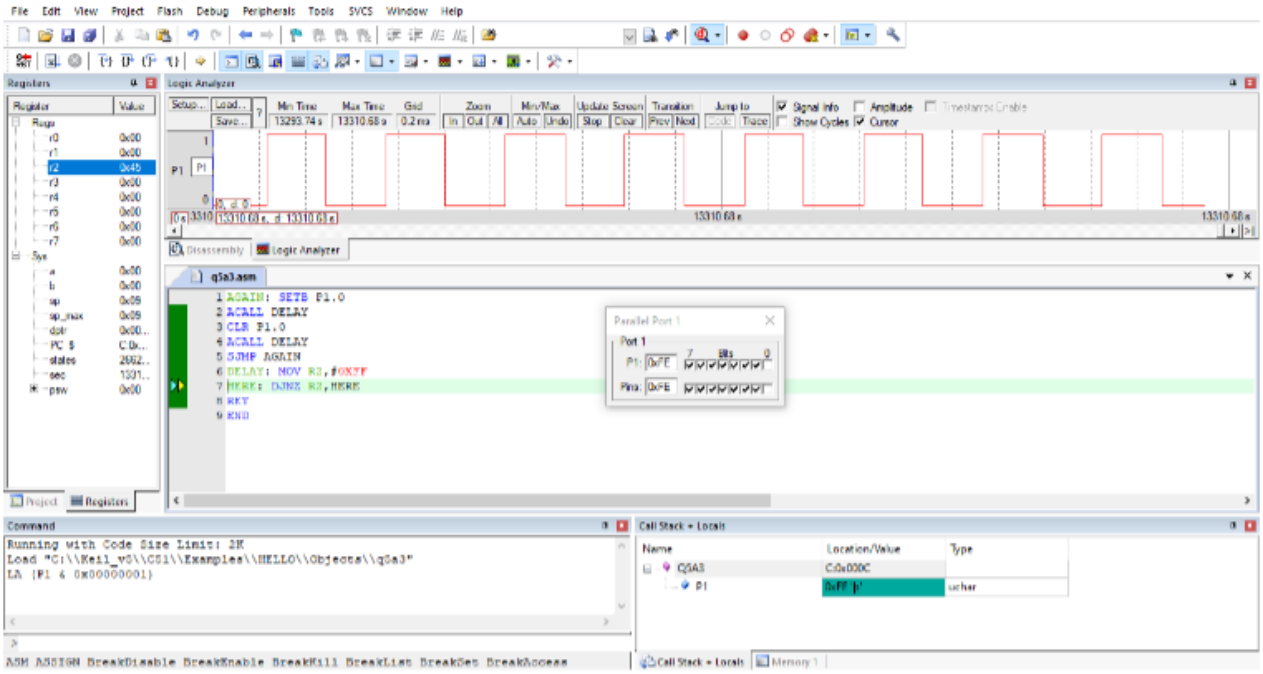


Output:

5.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H	HERE:	D2	SETB P1.7	Sets the bit 7 of port 7 to value of 1
0002H		12	LCALL DELAY	Long call to DELAY
0005H		C2	CLR P1.7	Clear the bit 7 of port 1
0007H		12	LCALL DELAY	Long call to DELAY
000AH		80	SJMP HERE	Short jump ton HERE
000CH	DELAY:	7B	MOV R3, #0FFH	Move data FFH to R3
000EH	AGAIN:	DB	DJNZ R3, AGAIN	Decrement R3 and jump to AGAIN if not zero
0010H		22	RET	Return to main routine
0011H			END	End

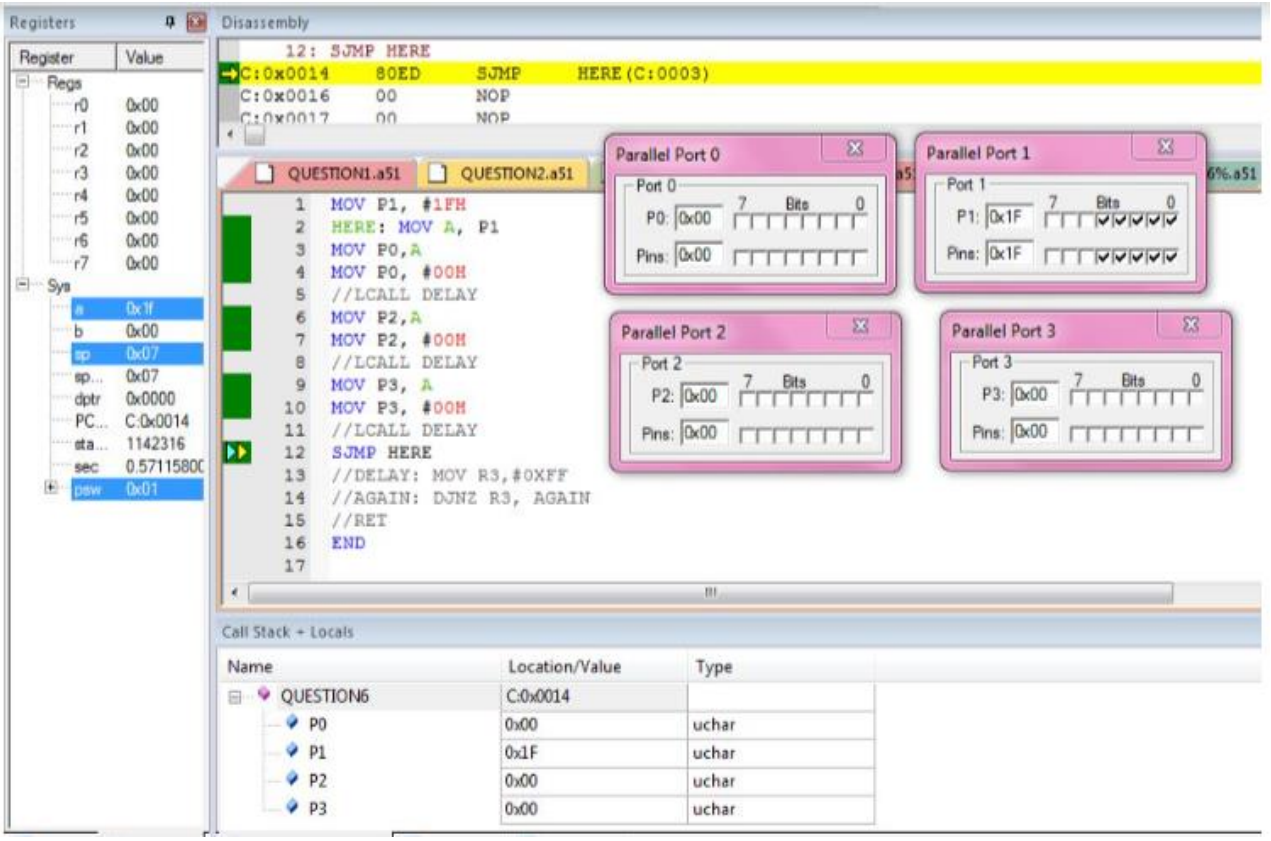
Output:



6.

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		92	MOV P1, #1FH	Move data 1h to P1
0002H	HERE	E5	MOV A, P1	Move data of P1 to A
0004H		92	MOV P0, A	Move the value of A to P0
0006H		92	MOV P0, #00H	Move data 0 to P0
0008H		92	MOV P2, A	Move data of A to P2
000AH		92	MOV P2, #00H	Move data 0 to P2
000CH		92	MOV P3, A	Move the value of A to P3
000EH		92	MOV P3, #00H	Move the data 0 to P3
0010H		80	SJMP HERE	Short jump to HERE
0012H			END	End

Output:



IV.Conclusion

The programs for I/O port programming using 8051 was successfully executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 04

Arithmetical and Logical way of Data Processing in 8051

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1.Write an Assembly language program (ALP) for 8051 to add 5 different 8-bit numbers stored in consecutive locations 40H onwards.

2.Write an ALP for 8051 to add 5 different 8-bit numbers stored in consecutive locations 40H onwards. Find the average of the numbers.

3.Write an ALP for 8051 to add 5 different 8-bit numbers stored in consecutive locations 40H onwards. The result must be in BCD.

4.Write an ALP for 8051 to add 2 16-bit numbers 3CE7h and 3B8Dh. Place the higher byte of result in R7 and lower byte in R6.

5.Write an ALP for 8051 to find the cube of a given 8–bit number.

6.Write an ALP for 8051 to determine whether the number is odd or even. Place the result suitably.

7.Write an ALP for 8051 to find the number of ones in a number.

8.Write an ALP for 8051 to (a) mask upper nibble (b) mask lower nibble.

9.Write an ALP for 8051 to interchange lower and upper nibble.

10.Write an ALP using 8051 to find 2's complement of a number.

11.Write an ALP using 8051 to find the Grey code equivalent of a given number.

12. Write an ALP using 8051 to read a number from P1 and convert it to decimal equivalent. Place the result 40H onwards with lowest significant digits at 40H.

II.Lab Component

Requirements:

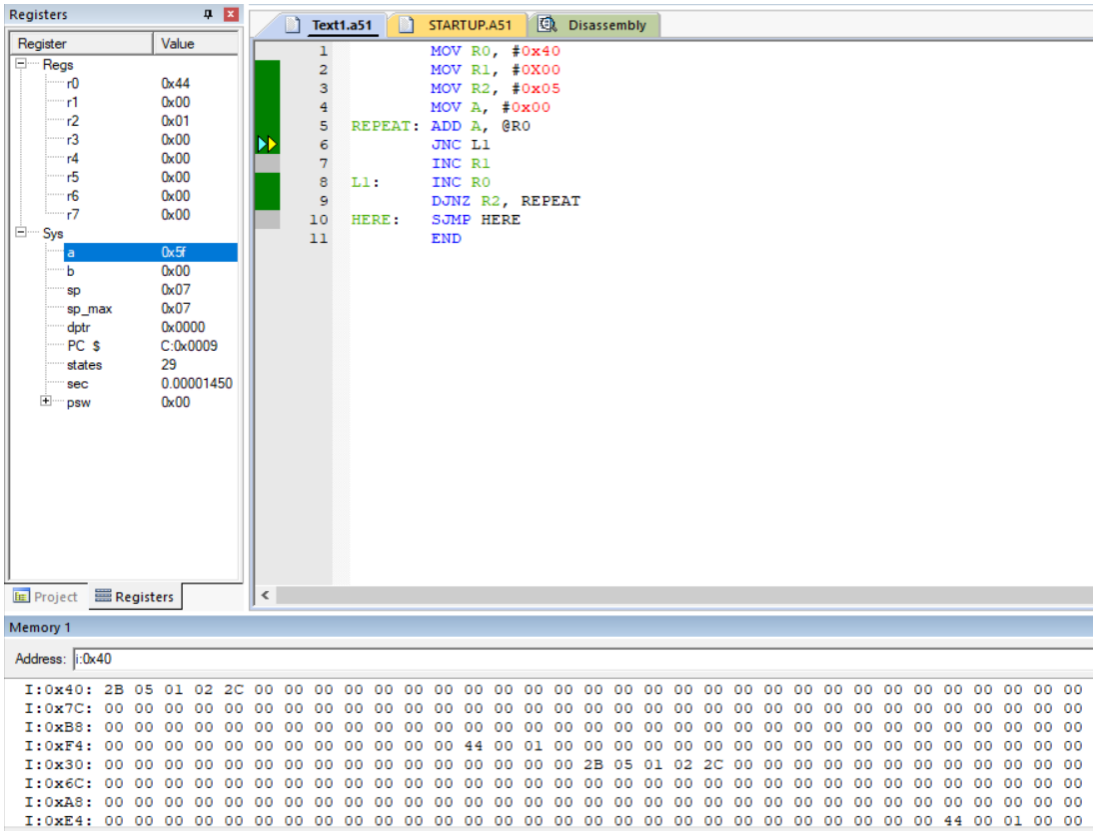
7) PC
8) Keil μVision software

III.Programs& Outputs

(1)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		A8	MOV R0, #0x40	Moves 40H to register R0
0002H		A9	MOV R1, #0x00	Moves 00H to register R1
0004H		AA	MOV R2, #0x05	Moves 05H to register R2
0006H		E5	MOV A, #0x00	Moves 00H to register A
0008H	REPEAT:	26	ADD A, @R0	Add data inAccumulator with content of R0
0009H		50	JNC L1	Jump to L1 if no carry
000BH		09	INC R1	Increment R1
000CH	L1:	08	INC R0	Increment R0
000DH		DA	DJNZ R2, REPEAT	Don't Jump if not zero
000FH	HERE:	80	SJMP HERE	Short Jump to HERE
0011H			END	

Program Output:



Results:

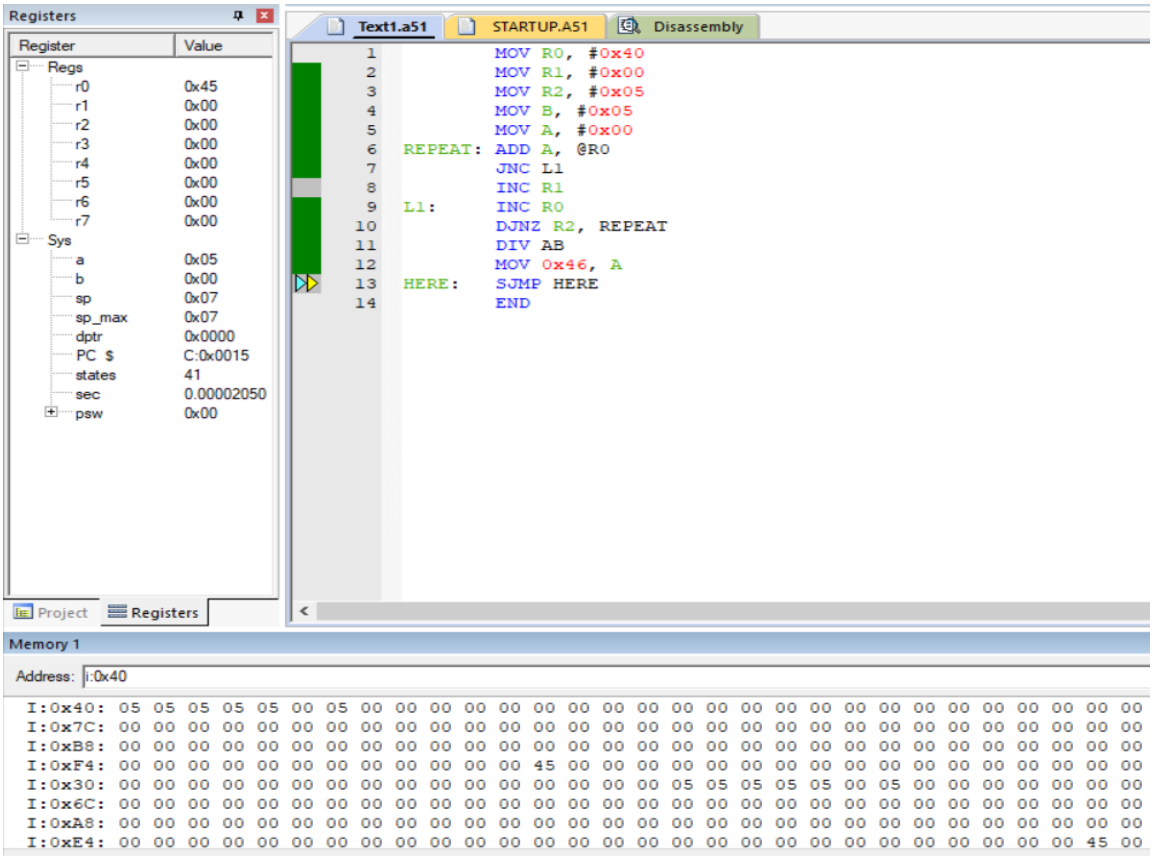
Memory Location	Input Data
0x40	2BH
0x41	05H
0x42	01H
0x43	02H

Memory Location	Output Data
0x44	2CH

(2)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		A8	MOV R0, #0x40	Moves 40H to register R0
0002H		A9	MOV R1, #0x00	Moves 00H to register R1
0004H		AA	MOV R2, #0x05	Moves 05H to register R2
0007H		A2	MOV B, #0x05	Moves 05H to register B
0008H		E5	MOV A, #0x00	Moves 00H to register A
000AH	REPEAT:	26	ADD A, @R0	Add data in Accumulator with content of R0
000BH		50	JNC L1	Jump to L1 if no carry
000DH		09	INC R1	Increment R1
000EH	L1:	08	INC R0	Increment R0
000FH		DA	DJNZR2, REPEAT	Don't Jump if not zero
0011H		84	DIV AB	Divide value in A by B
0012H		F5	MOV 0x46, A	Moves data in Accumulator to 0x46
0014H	HERE:	80	SJMP HERE	Short Jump to HERE
0016H			END	

Program Output:



Results:

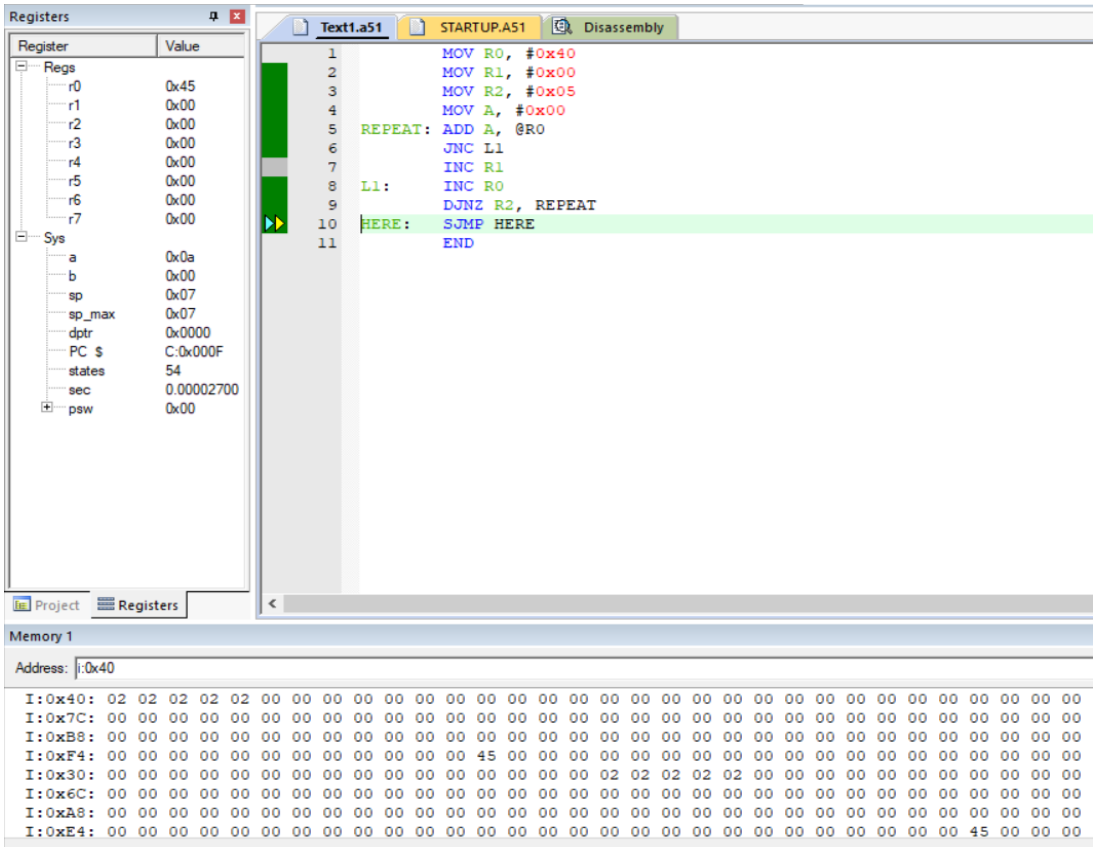
Memory Location	Input Data
0x40	05H
0x41	05H
0x42	05H
0x43	05H
0x44	05H

Memory Location	Output Data
0x46	05H

(3)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		A8	MOV R0, #0x40	Moves 40H to register R0
0002H		A9	MOV R1, #0x00	Moves 00H to register R1
0004H		AA	MOV R2, #0x05	Moves 05H to register R2
0007H		E5	MOV A, #0x00	Moves 00H to register A
0008H	REPEAT:	26	ADD A, @R0	Add data in Accumulator with content of R0
0009H		50	JNC L1	Jump to L1 if no carry
000BH		09	INC R1	Increment R1
000CH	L1:	08	INC R0	Increment R0
000DH		DA	DJNZ R2, REPEAT	Don’t Jump if not zero
000FH	HERE:	80	SJMP HERE	Short Jump to HERE
0011H			END	

Program Output:



Results:

Memory Location	Input Data
0x40	02H
0x41	02H
0x42	02H
0x43	02H
0x44	02H

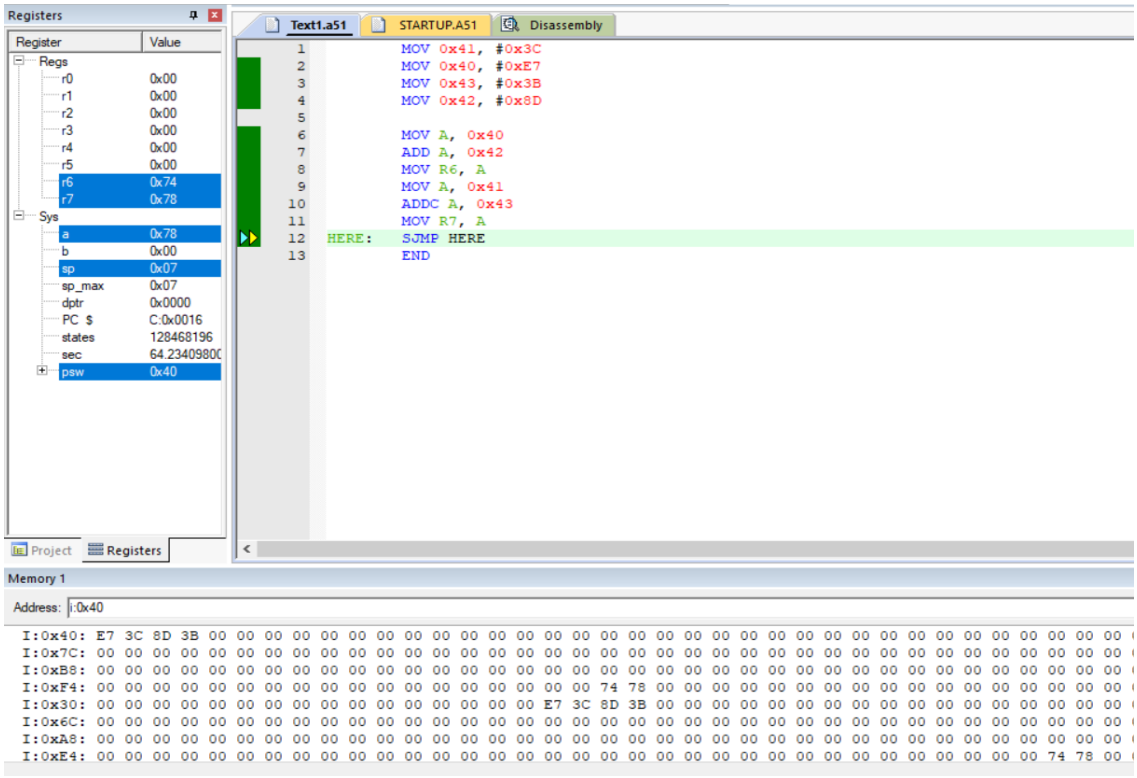
Memory Location	Output Data
0x06	45H

(4)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		75	MOV 0x41, #0x3C	Moves data 3CH to address 0x41
0002H		75	MOV 0x40, #0xE7	Moves data E7H to address 0x40
0004H		75	MOV 0x43, #0x3B	Moves data 3BH to address 0x43
0007H		75	MOV 0x42, #0x8D	Moves data 8DH to address 0x42
0008H		E5	MOV A, 0x40	Moves data in address 0x40 to Accumulator
000AH		25	ADD A, 0x42	Add both data in Accumulator and 0x42
000CH		FE	MOV R6, A	Move data in accumulator to R6
000EH		E5	MOV A, 0x41	Move data in 0x41 to Accumulator
0010H		35	ADDC A, 0x43	Add data in 0x43 and accumulator, if complement

0012H		FF	MOV R7, A	Move Accumulator to R7
0014H	HERE:	80	SJMP HERE	Short Jump to HERE
0016H			END	

Program Output:



Results:

Memory Location	Input Data
0x40	E7H
0x41	3CH
0x42	8DH
0x43	3BH

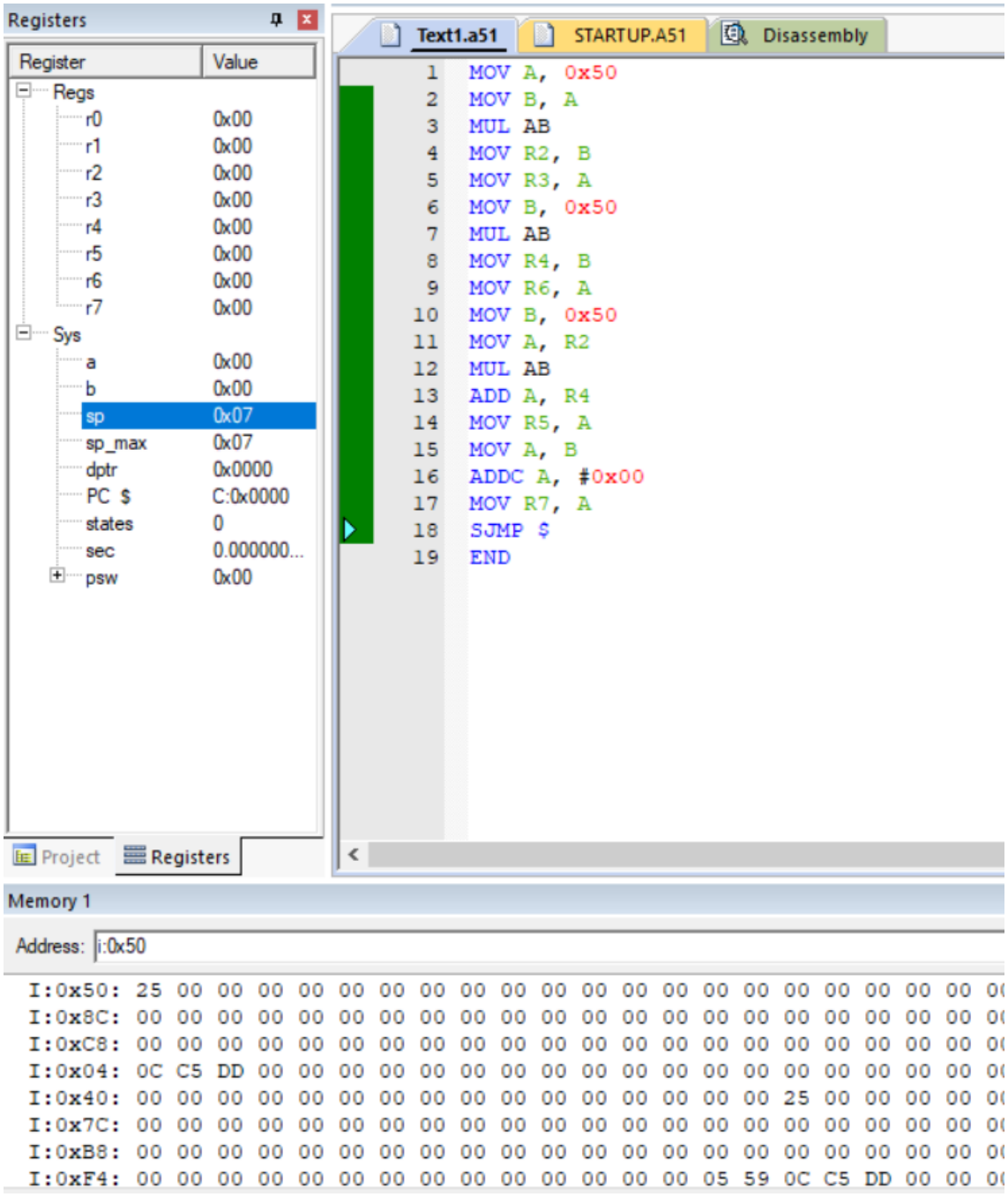
Memory Location	Output Data
0x0C	74
0x0D	78

(5)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x50	Moves data from 0x50 to Accumulator
0002H		F5	MOV B, A	Moves data from Accumulator to B
0004H		A4	MUL AB	Multiply data in Accumulator with data in B
0005H		AA	MOV R2, B	Moves data from B to R2

0007H		FB	MOV R3, A	Moves data from Accumulator to R3
0009H		85	MOV B, 0x50	Moves data from 0x50 to B
000BH		A4	MUL AB	Multiply data in Accumulator with data in B
000CH		AC	MOV R4, B	Moves data from B to R4
000EH		FE	MOV R6, A	Moves data from Accumulator to R6
0010H		85	MOV B, 0x50	Moves data from 0x50 to B
0012H		EA	MOV A, R2	Moves data in R2 to Accumulator
0014H		A4	MUL AB	Multiply data in Accumulator with data in B
0015H		2C	ADD A, R4	Adds values in Accumulator and R4
0016H		FD	MOV R5, A	Moves data from accumulator to R5
0018H		E5	MOV A, B	Moves data from B to Accumulator
001AH		34	ADDC A, #0x00	Adds values of Accumulator and 00H, and moves to Accumulator; if complement
001CH		FF	MOV R7, A	Moves data from Accumulator to R7
001EH		80	SJMP $	Short Jump to HERE
0020H			END	

Program Output:



Results

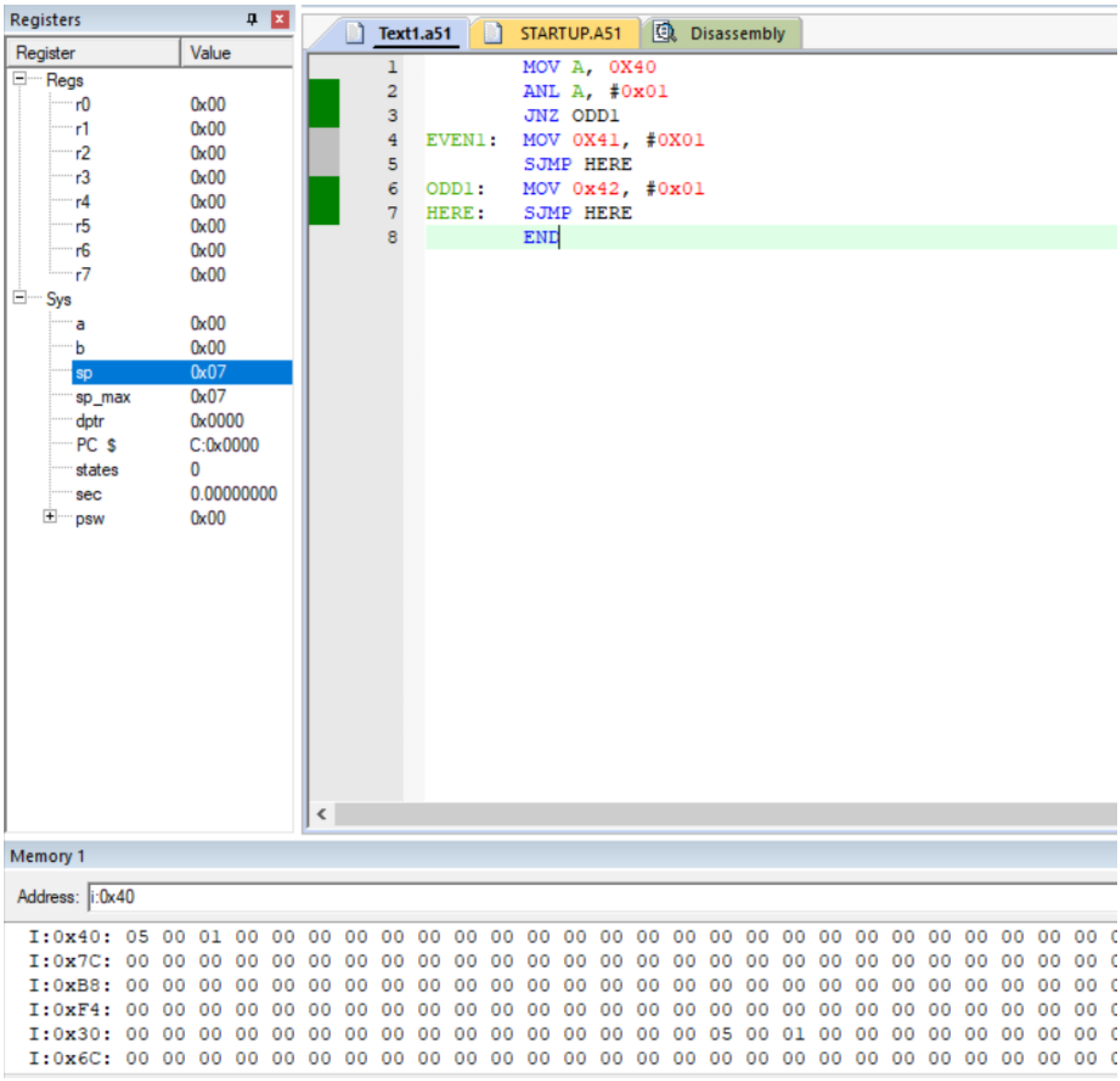
Memory Location	Input Data
0x50	25H

Memory Location	Output Data
0x04	0CH
0x05	C5H
0x06	DDH

(6)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x40	Moves data from 0x40 to Accumulator
0002H		54	ANL A, #0x01	Bitwise logical AND operation occurs, which is then stored in Accumulator
0004H		70	JNZ ODD1	Jump to ODD1 if not zero
0006H	EVEN1:	75	MOV 0x41, #0x01	Moves 01H to memory address 0x41
0008H		80	SJMP HERE	Short Jump to HERE
000AH	ODD1:	75	MOV 0x42, #0x01	Moves 01H to memory address 0x42
000CH	HERE:	80	SJMP HERE	Short Jump to HERE
000EH			END	

Program Output:



Results

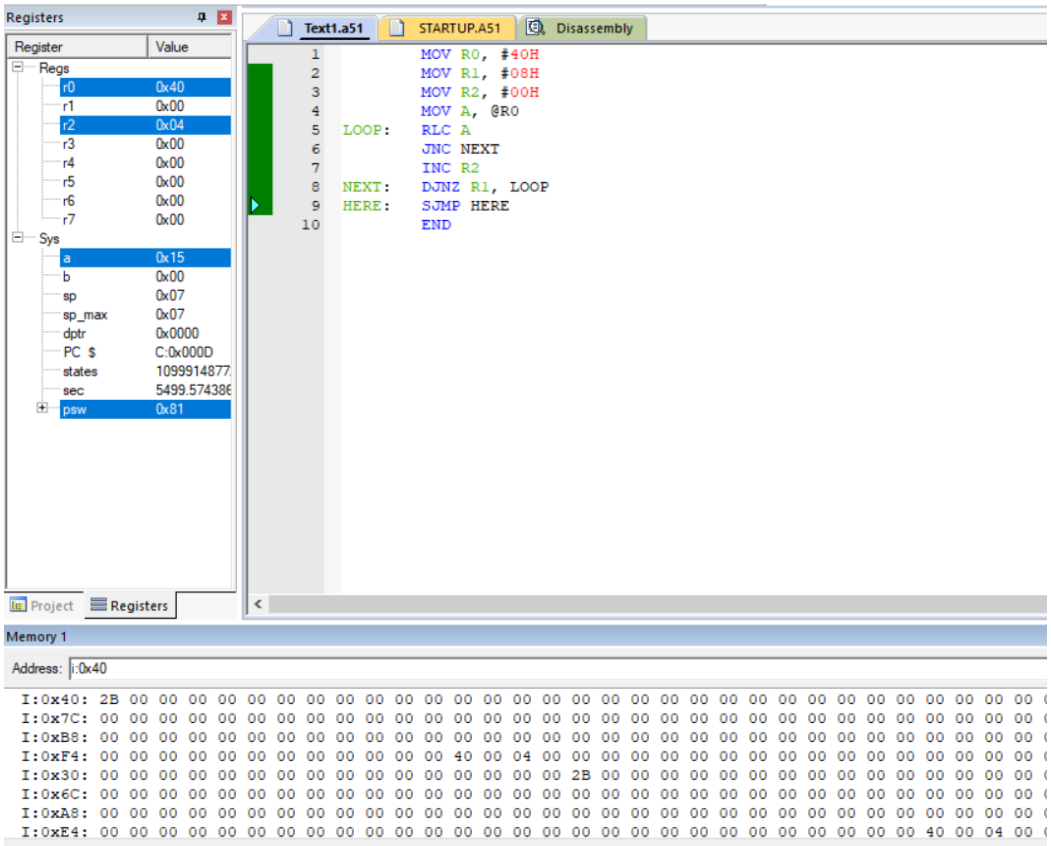
Memory Location	Input Data
0x40	05H

Memory Location	Output Data
0x42	01H

(7)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		78	MOV R0, #40H	Moves 40H to R0
0002H		79	MOV R1, #08H	Moves 08H to R1
0004H		7A	MOV R2, #00H	Moves 00H to R2
0006H		E6	MOV A, @R0	Indirect addressing
0008H	LOOP:	33	RLC A	Rotate Accumulator left through carry
0009H		50	JNC NEXT	Jump to NEXT if no carry
000BH		0A	INC R2	Increment R2
000CH	NEXT:	D9	DJNZ R1, LOOP	Don't Jump if Not Zero
000EH	HERE:	80	SJMP HERE	Short Jump to HERE
0010H			END	

Program Output:



Results

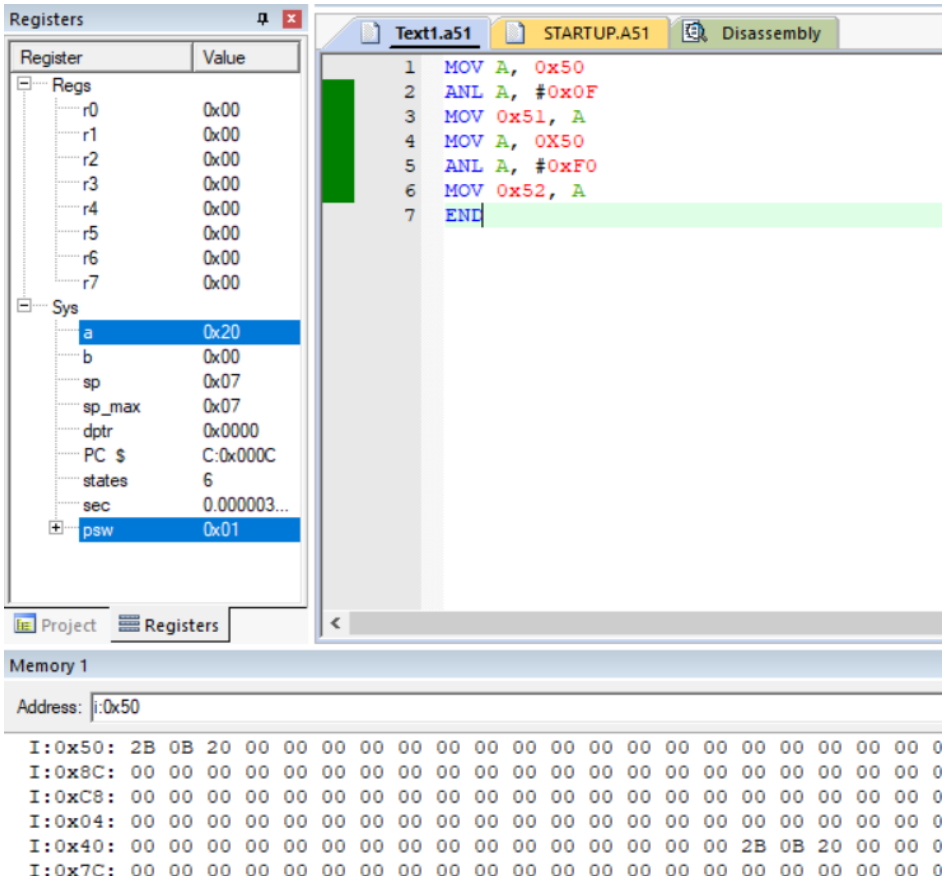
Memory Location	Input Data
0x40	2BH

Memory Location	Output Data
0x08	04H

(8)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x50	Moves data from 0x50 to Accumulator
0002H		54	ANL A, #0x0F	Bitwise logical AND operation occurs with 0FH, and is then stored in Accumulator
0004H		F5	MOV 0x51, A	Moves data from Accumulator to 0x51
0006H		E5	MOV A, 0x50	Moves data from 0x50 to Accumulator
0008H		54	ANL A, #0xF0	Bitwise logical AND operation occurs with F0H, and is then stored in Accumulator
000AH		F5	MOV 0x52, A	Moves data from Accumulator to 0x52
000CH			END	

Program Output:



Results

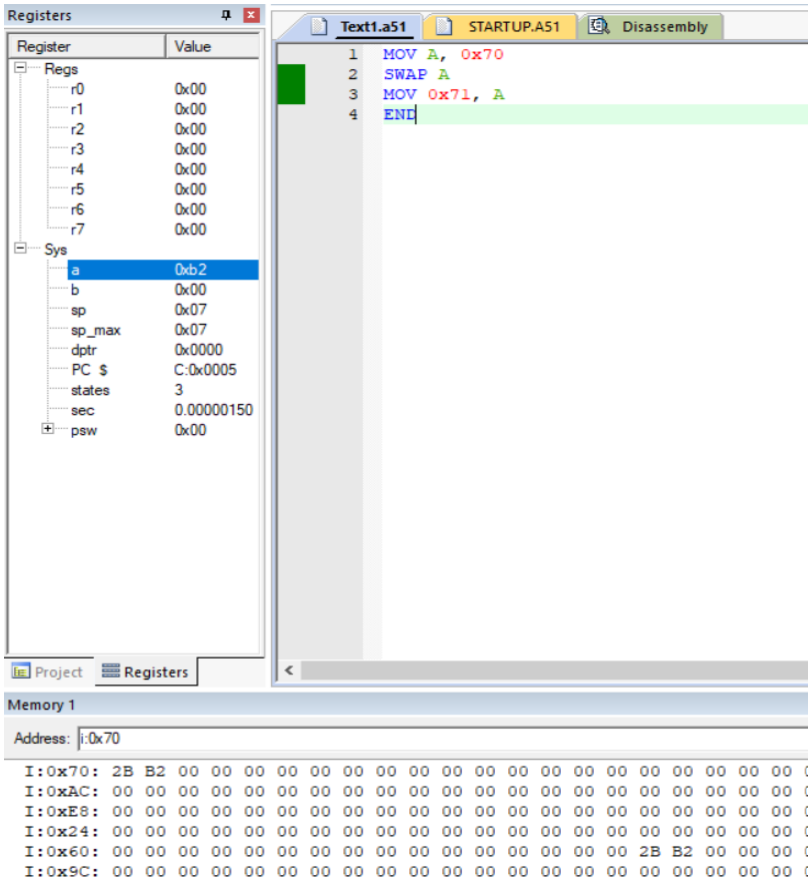
Memory Location	Input Data
0x50	2BH

Memory Location	Output Data
0x51	0BH
0x52	20H

(9)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x70	Moves data from memory address 0x70 to Accumulator
0002H		C4	SWAP A	Swaps the content in Accumulator
0003H		F5	MOV 0x71, A	Moves the data from Accumulator to memory address 0x71
0005H			END	

Program Output:



Results

Memory Location	Input Data
0x70	2BH

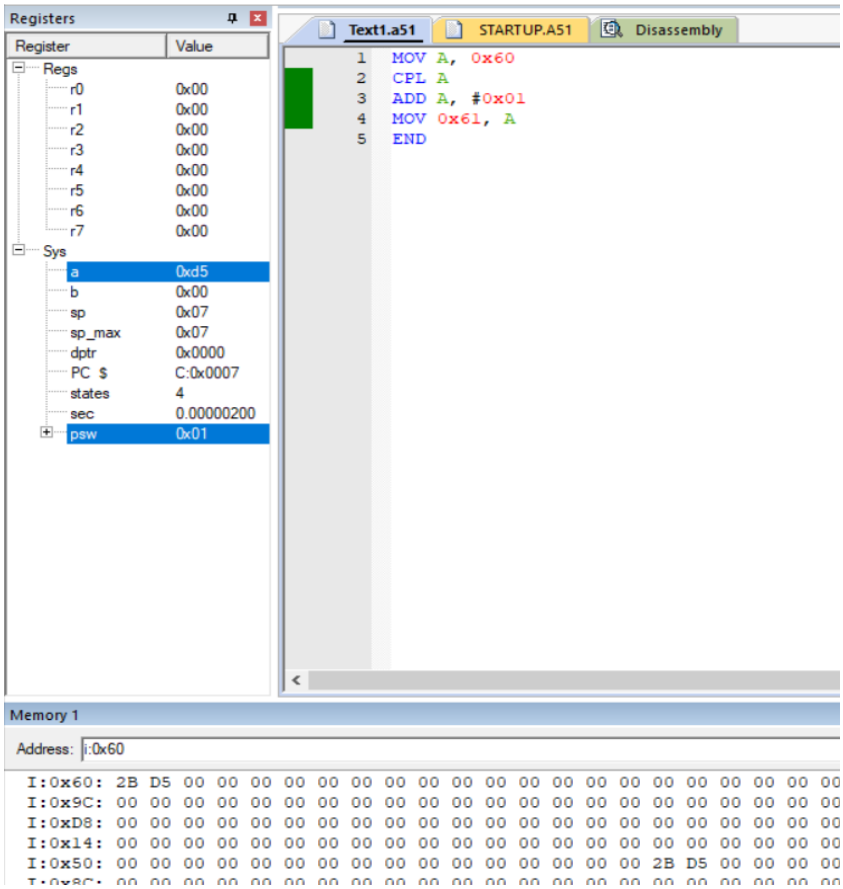
Memory Location	Output Data
0x71	B2H

(10)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x60	Moves data from 0x60 to Accumulator
0002H		F4	CPL A	Complement Accumulator
0003H		24	ADD A, #0x01	Adds data in Accumulator and 01H
0005H		F5	MOV 0x61, A	Moves data from Accumulator to memory address

				0x61
0007H			END	

Program Output:



Results

Memory Location	Input Data
0x60	2BH

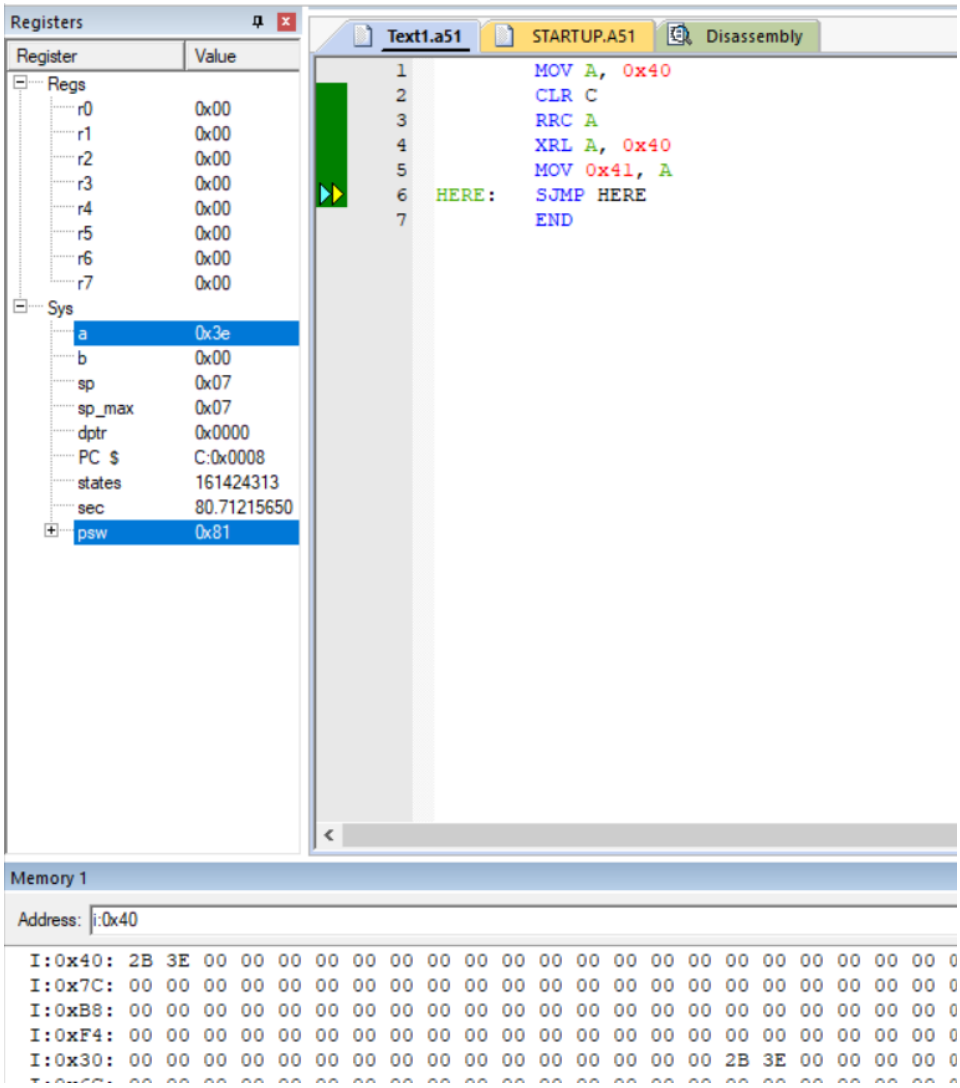
Memory Location	Output Data
0x61	D5H

(11)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x40	Moves data from memory address 0x40 to Accumulator
0002H		C3	CLR C	Clears the content of C
0003H		13	RRC A	Rotate Accumulator Right through Carry
0004H		65	XRL A, 0x40	Logical exclusive OR operation occurs with content in 0x40, and is then stored in Accumulator
0006H		F5	MOV 0x41, A	Moves data from Accumulator to memory address

				0x41
0008H	HERE:	80	SJMP HERE	Short Jump to HERE
000AH			END	

Program Output:



Results

Memory Location	Input Data
0x40	2BH

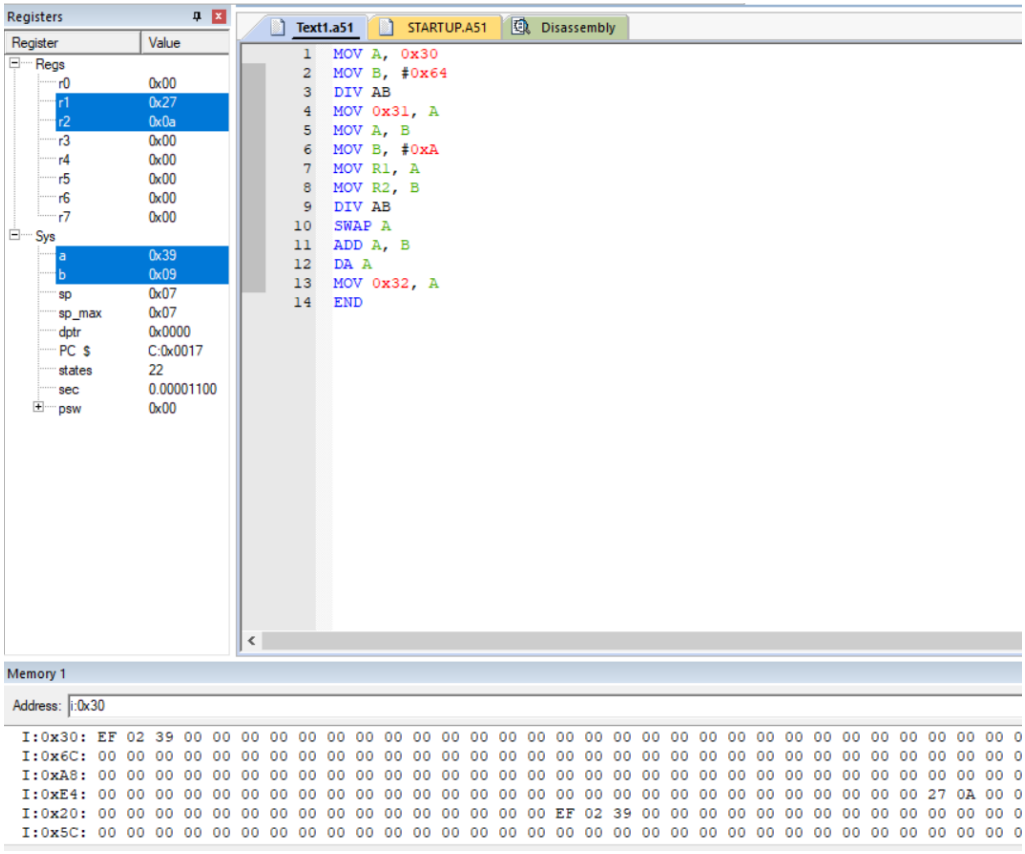
Memory Location	Output Data
0x41	3EH

(12)

Memory Location	Label	Opcode	Mnemonics	Comments
0000H		E5	MOV A, 0x30	Moves data from memory address 0x30 to Accumulator
0002H		75	MOV B, #0x64	Moves 64H to B
0004H		84	DIV AB	Divides A by B

0005H		F5	MOV 0x31, A	Moves data from Accumulator to memory address 0x31
0007H		E5	MOV A, B	Moves data from B to A
0009H		75	MOV B, #0xA	Moves 0AH to B
000BH		F9	MOV R1, A	Moves data from Accumulator to R1
000DH		AA	MOV R2, B	Moves data from B to R2
000FH		84	DIV AB	Divides A by B
0010H		C4	SWAP A	Swaps the content in Accumulator
0011H		25	ADD A, B	Adds the values in Accumulator and B
0012H		D4	DA A	Adjust content in accumulator to BCD
0013H		F5	MOV 0x32, A	Moves data from Accumulator to memory address 0x32
0015H			END	

Program Output:



Results:

Memory Location	Input Data
0x30	EFH
0x31	02H

Memory Location	Output Data
0x32	39H

IV.Conclusion

The programs for arithmetical and logical way of data processing in 8051 was successfully executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 05

8051 programming in C

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1.Write an 8051 C program to toggle all the bits of P1 and P3 continuously with a 250 ms delay.

2.Write an 8051 C program to toggle only bit P1.6 continuously without changing the rest bits of the same port.

3.Write an 8051 C program to get bit P1.6 and send it to P2.6 after complementing it.

4.Write an 8051 C program to send values -4 to +4 to port P2.

5.Write an 8051 C program to get a byte of data from P1. If it is less than 100, send it to P0; otherwise, send it to P2.

6.Write an 8051 C program to convert packed BCD 0x29 to ASCII and display the bytes on P1 and P2.

II.Lab Component

Requirements:

9) PC
10) EDSim51DI or Keil μVision software

III.Programs

Complete the following using EDSim51DI/ Keil μVision software

(All the programs must be written in the following table.)

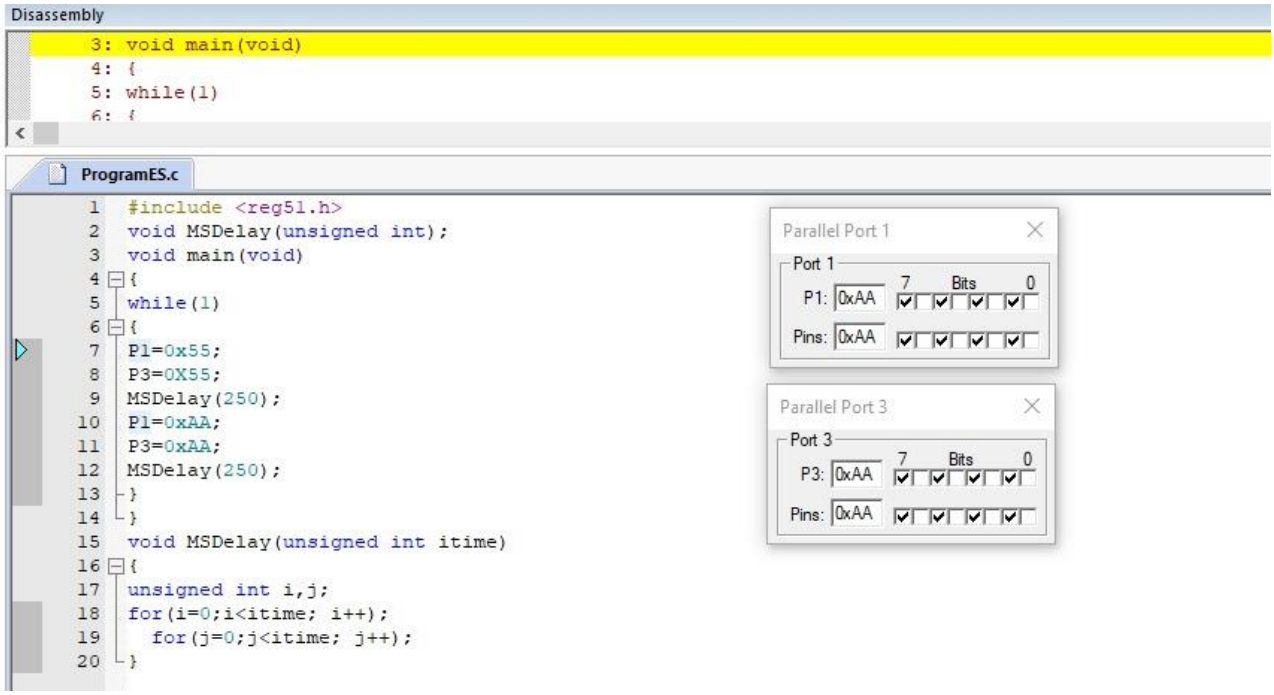
1.

Code

```
#include <reg51.h>

void MSDelay(unsigned int);

void main(void)

{

        while(1)

        {

                P1=0x55;
```

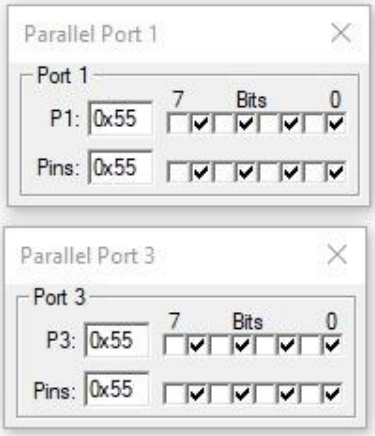
```
            P3=0X55;
            MSDelay(250);
            P1=0xAA;
            P3=0xAA;
            MSDelay(250);
        }
}
void MSDelay(unsigned int itime)
{
        unsigned int i,j;
        for(i=0;i<itime; i++);
            for(j=0;j<itime; j++);
}
```

Result



gramES.c

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while(1)
{
P1=0x55;
P3=0X55;
MSDelay(250);
P1=0xAA;
P3=0xAA;
MSDelay(250);
}
}
void MSDelay(unsigned int itime)
{
unsigned int i,j;
for(i=0;i<itime; i++);
  for(j=0;j<itime; j++);
}
```



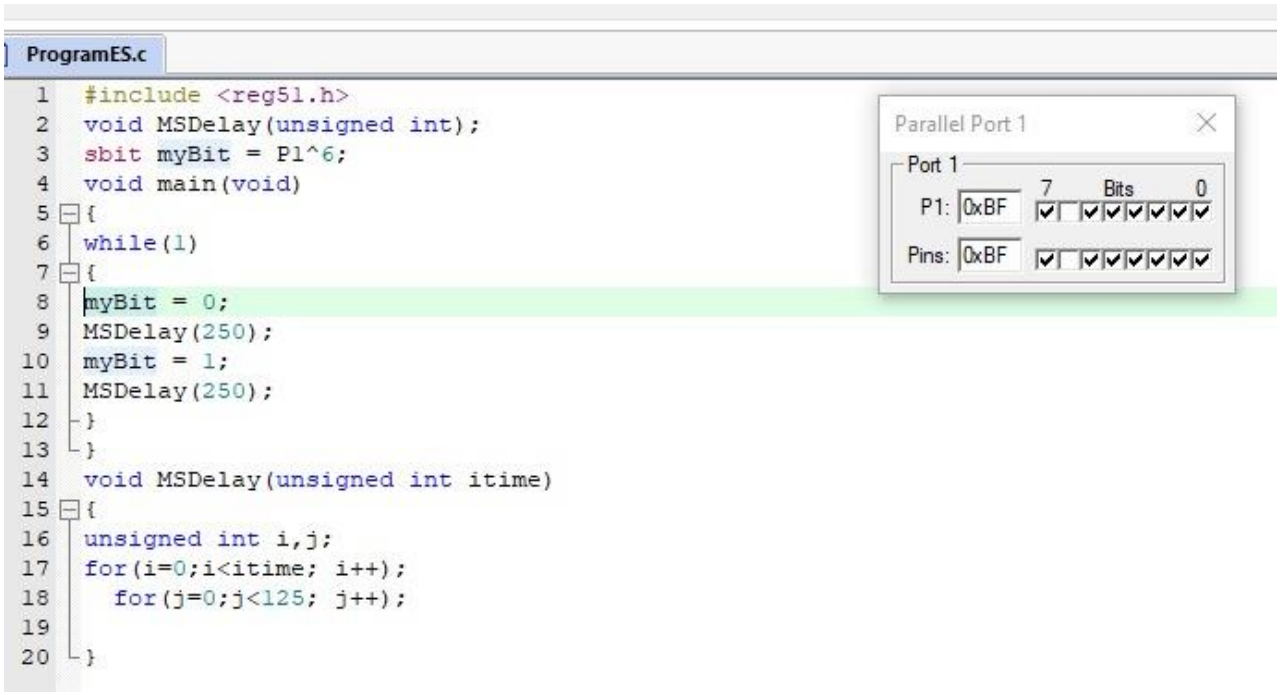
2.

Code

```
#include <reg51.h>

void MSDelay(unsigned int);

sbit myBit = P1^6;

void main(void)

{

        while(1)

        {

                myBit = 0;

                MSDelay(250);

                myBit = 1;

                MSDelay(250);

        }
```

```
}

void MSDelay(unsigned int itime)

{

        unsigned int i,j;

        for(i=0;i<itime; i++);

                for(j=0;j<125; j++);

}
```

Result



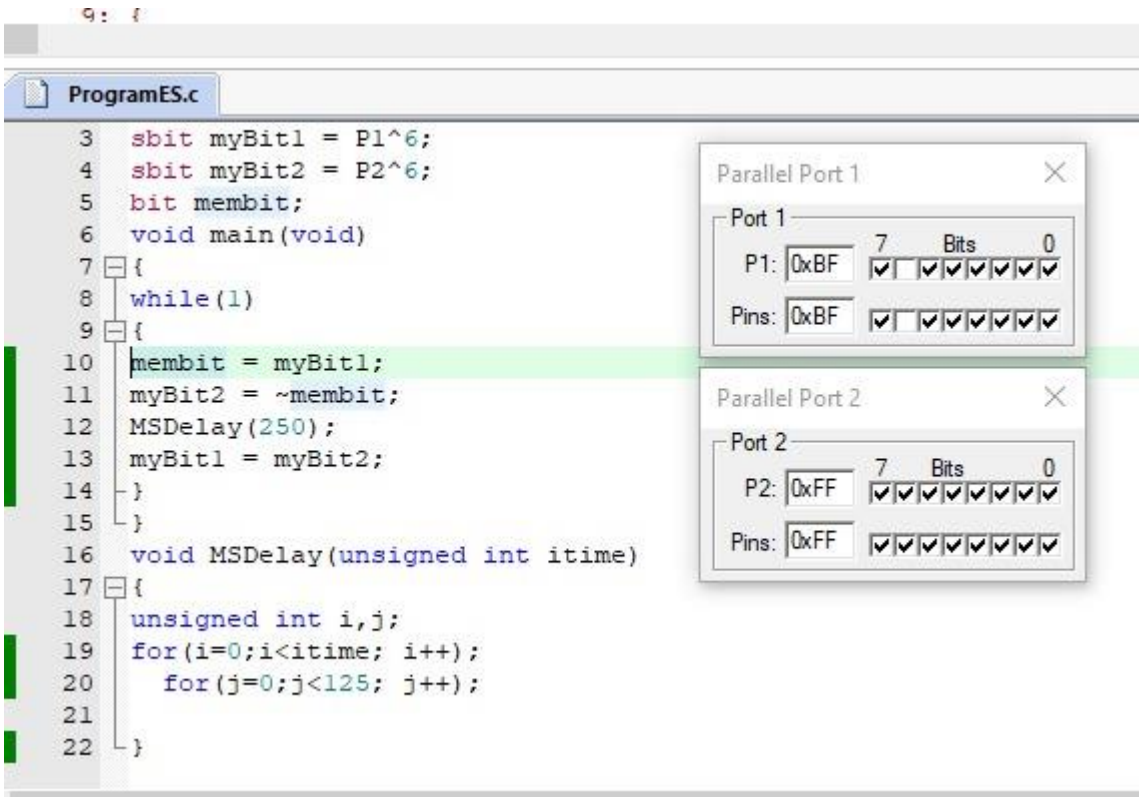
3.

<u>Code</u>

```
#include <reg51.h>

void MSDelay(unsigned int);

sbit myBit1 = P1^6;

sbit myBit2 = P2^6;
```

```
bit membit;

void main(void)

{

        while(1)

        {

                membit = myBit1;

                myBit2 = ~membit;

                MSDelay(250);

                myBit1 = myBit2;

        }

}

void MSDelay(unsigned int itime)

{

        unsigned int i,j;

        for(i=0;i<itime; i++);

                for(j=0;j<125; j++);

}
```

Result



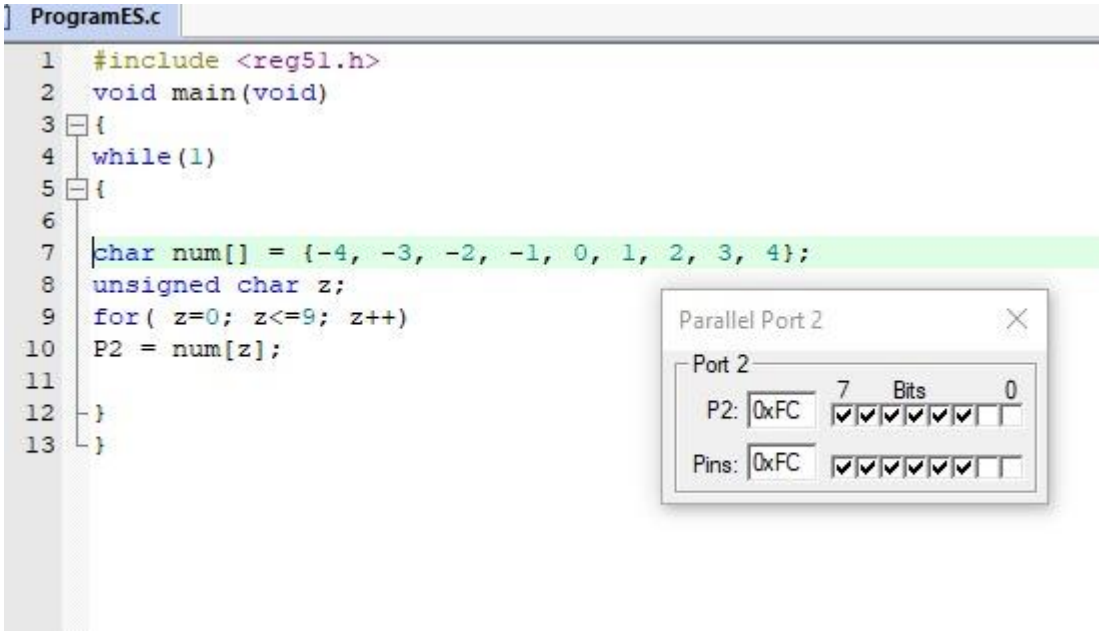
4.

Code

```
#include <reg51.h>

void main(void)

{

        while(1)

        {

                char num[] = {-4, -3, -2, -1, 0, 1, 2, 3, 4};

                unsigned char z;

                for( z=0; z<=9; z++)

                        P2 = num[z];

        }
```

}

Result



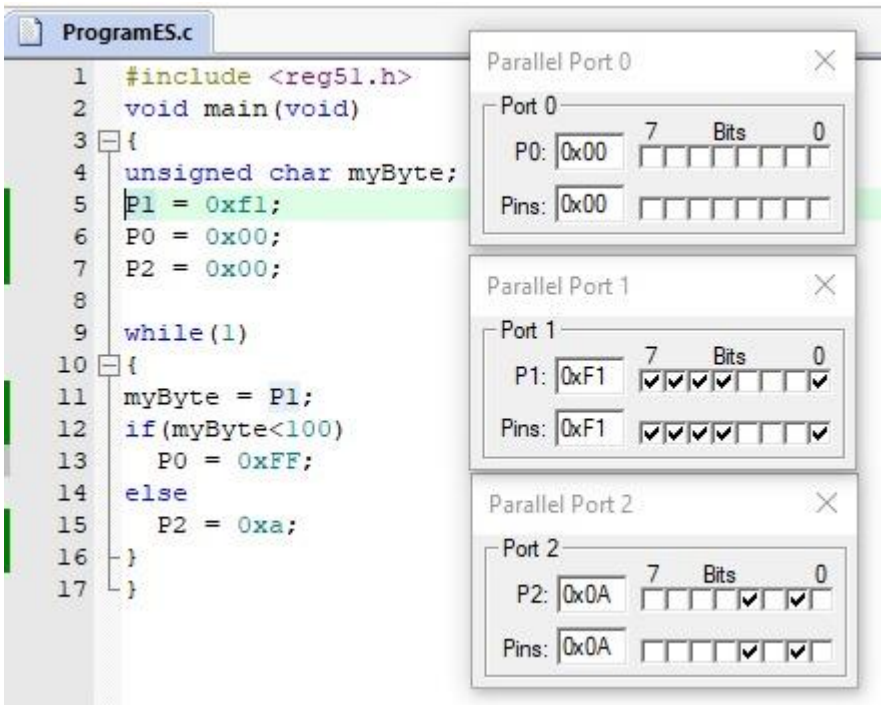
5.

Code

```
#include <reg51.h>

void main(void)

{

        unsigned char myByte;

        P1 = 0xf1;

        P0 = 0x00;

        P2 = 0x00;



        while(1)

                {
```

```
                myByte = P1;
                if(myByte<100)
                        P0 = 0xFF;
                else
                        P2 = 0xa;
        }
}
```

Result



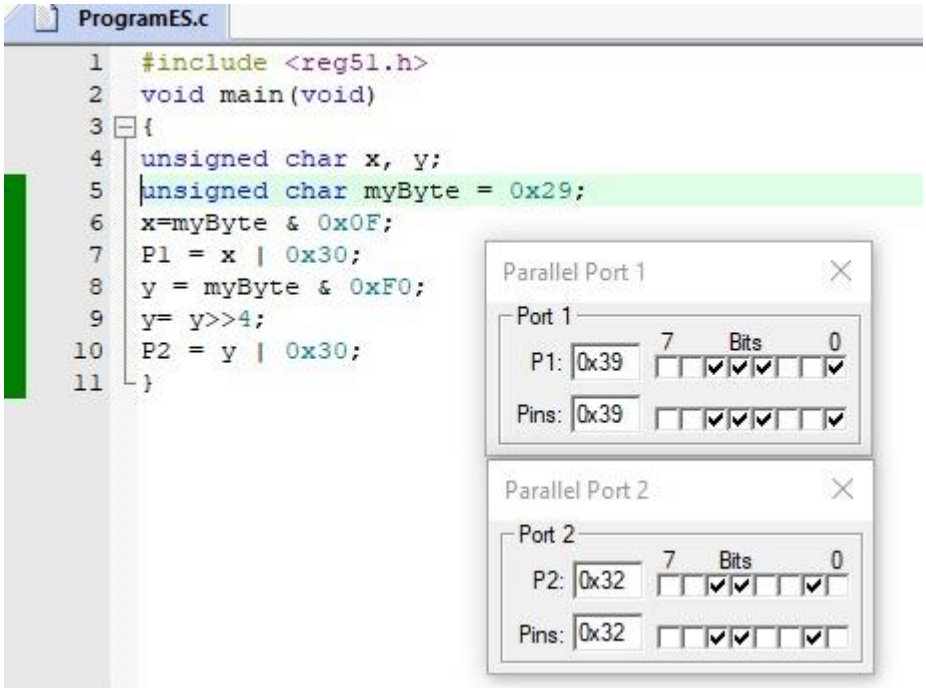
6.

Code

```
#include <reg51.h>
void main(void)
{
        unsigned char x, y;
```

```
unsigned char myByte = 0x29;

x=myByte & 0x0F;

P1 = x | 0x30;

y = myByte & 0xF0;

y= y>>4;

P2 = y | 0x30;
}
```

Result



IVConclusion

Thus all the 8051 programs were completed using C programming.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 06

Data transfer among register, memory, and segments in 8086

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1. Write an Assembly language program (ALP) using 8086 to clear 100 memory locations in the data segment.
2. Write an ALP using 8086 to read the numbers from one array and add 7 to each value and store the results in another array.
3. Write an ALP program using 8086 to read the 8-bit numbers from two arrays with one from each array add the data and put the result in a third array. Assume that no carry is produced during the addition.
4. Write an ALP using 8086 to exchange the content of two registers using stack.
5. Write an ALP using 8086 to move one-hundred-word type data from the offset address 1000H to the offset address 3000H in the segment 5000H.
6. Write an ALP using 8086 to move one hundred bytes of data from the offset address 2000H to the offset address 3000H in the segment 4000H.

II.Lab Component

Requirements:

11) PC
12) EMU8086

Theory:

Need for Segmentation –

The Bus Interface Unit (BIU) contains four 16 bit special purpose registers (mentioned below) called as Segment Registers.

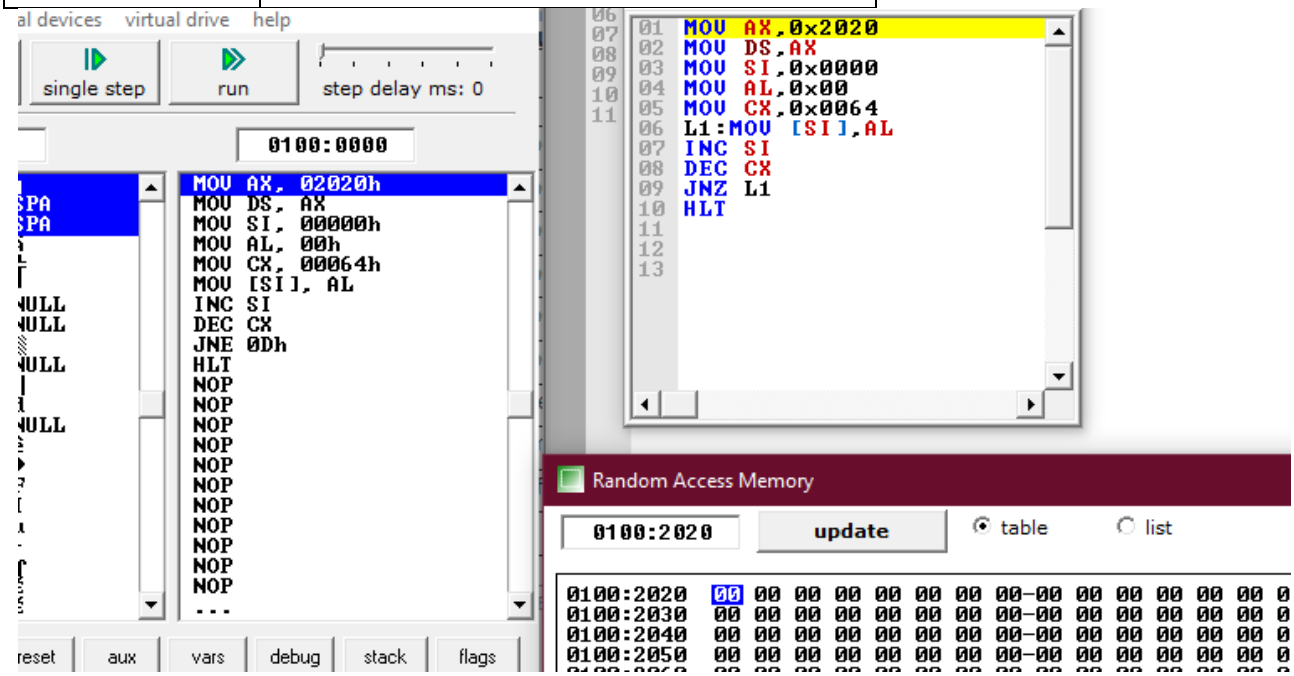
- **Code segment register (CS):** is used for addressing memory location in the code segment of the memory, where the executable program is stored.
- **Data segment register (DS):** points to the data segment of the memory where the data is stored.
- **Extra Segment Register (ES):** also refers to a segment in the memory which is another data segment in the memory.
- **Stack Segment Register (SS):** is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

IV.Programs& Outputs

(1)

Mnemonics	Comments
MOV AX,0x2020	load the value 2020h into AX register
MOV DS,AX	load the value of AX register into DS
MOV SI,0x0000	load the value 0000h into offset SI
MOV AL,0x00	load the value 0000h into AL
MOV CX,0x0064	load the value 100h into CX
L1:MOV [SI],AL	load the value AL into SI
INC SI	Increment the offset SI by 1

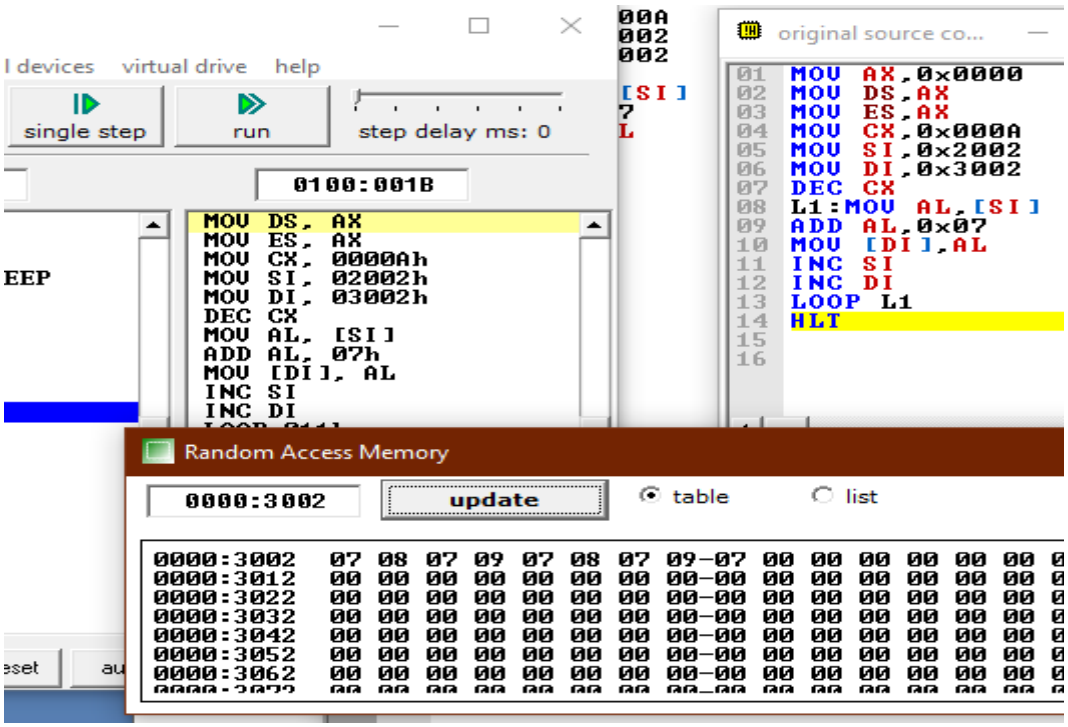
DEC CX	Decrement CX by 1
JNZ L1	Jump if Not equal to zero
HLT	Halt



(2)

Mnemonics	Comments
MOV DS,AX	load the value of AX register into DS
MOV ES,AX	load the value of AX register into ES
MOV CX,0x000A	load the value 000Ah into CX
MOVSI,0x2002	load the value 2002h into offset SI.
MOV DI,0x3002	load the value 3002h into offset DI.
DEC CX	Decrement CX by 1
L1:MOV AL,[SI] ADD AL,0x07	load the value SI into AL Add 07h to AL
MOV [DI],AL	load the value AL into DI
INC SI	Increment the offset SI by 1
INC DI	Increment the offset DI by 1
LOOP L1	Loop

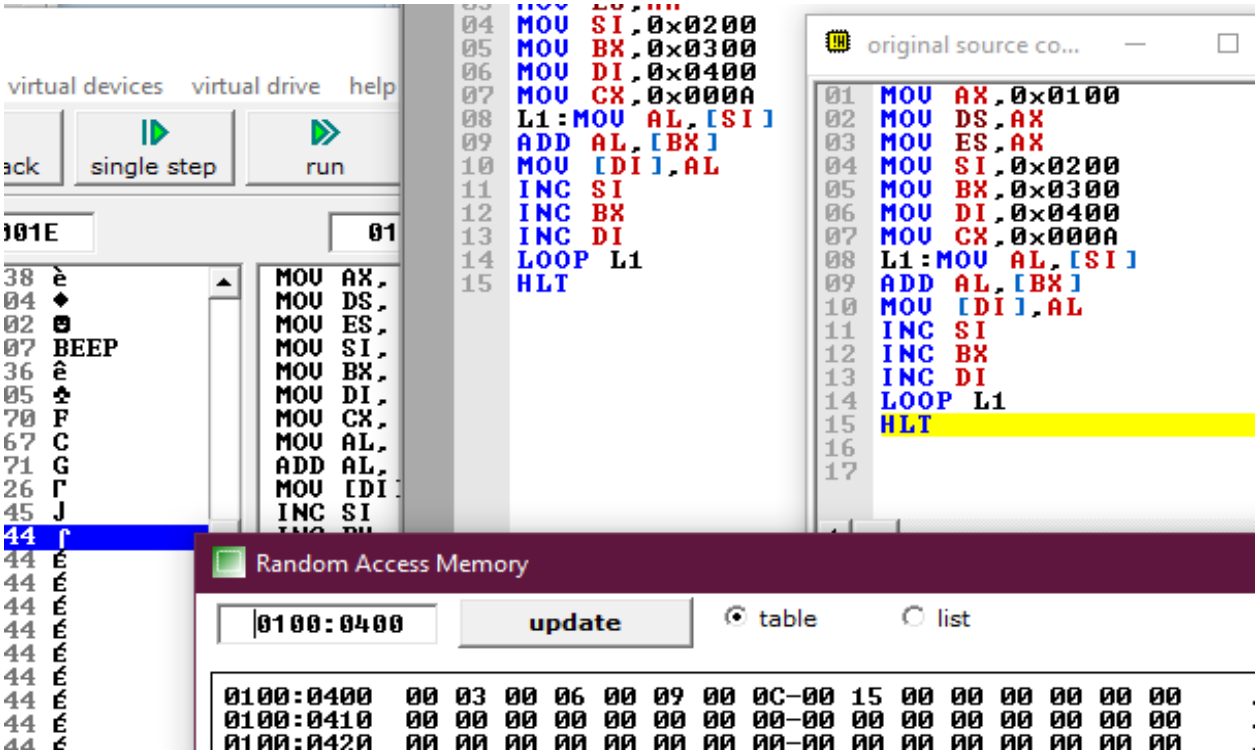
HLT	Halt



(3)

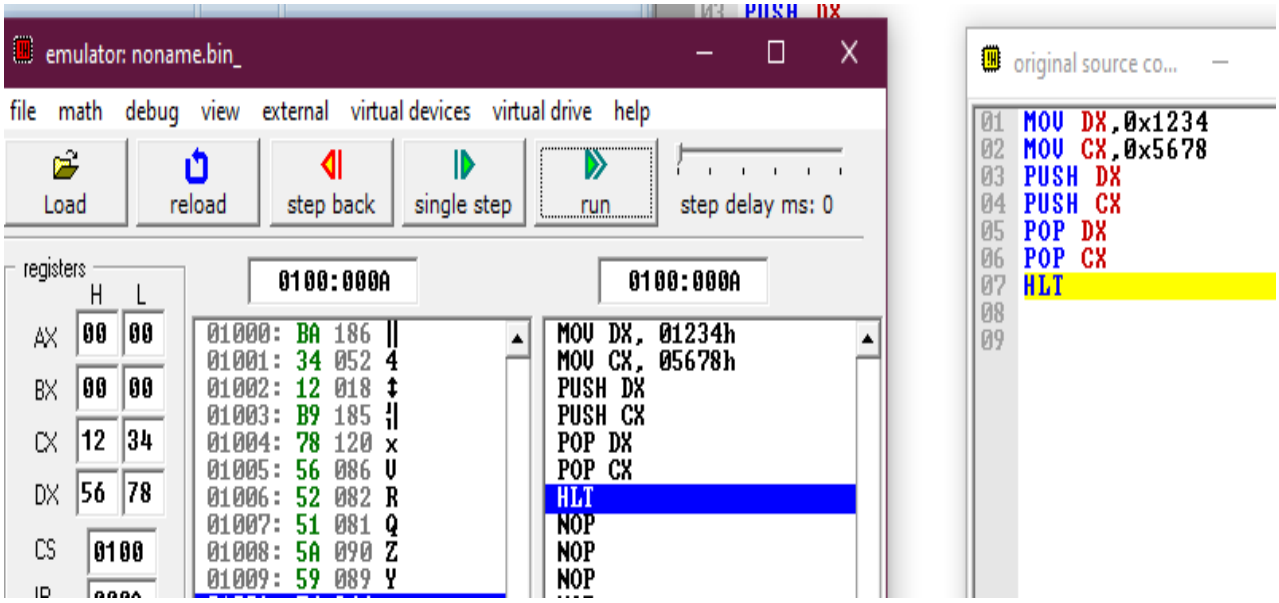
Mnemonics	Comments
MOV AX,0x0100	load the value 100h to AX register
MOV DS,AX	load the value of AX register into DS
MOV ES,AX	load the value of AX register into ES
MOV SI,0x0200	load the value 0200h into offset SI.
MOV BX,0x0300	load the value 0300h into BX
MOV DI,0x0400	load the value 0400h into offset DI.
MOV CX,0x000A	load the value 000Ah into CX
L1:MOV AL,[SI]	load the value SI into AL
ADD AL,[BX]	AL=AL+[BX]
MOV [DI],AL	load the value AL into DI
INC SI	Increment the offset SI by 1
INC BX	Increment the BX by 1

INC DI	Increment the offset DI by 1
LOOP L1	Loop
HLT	Halt



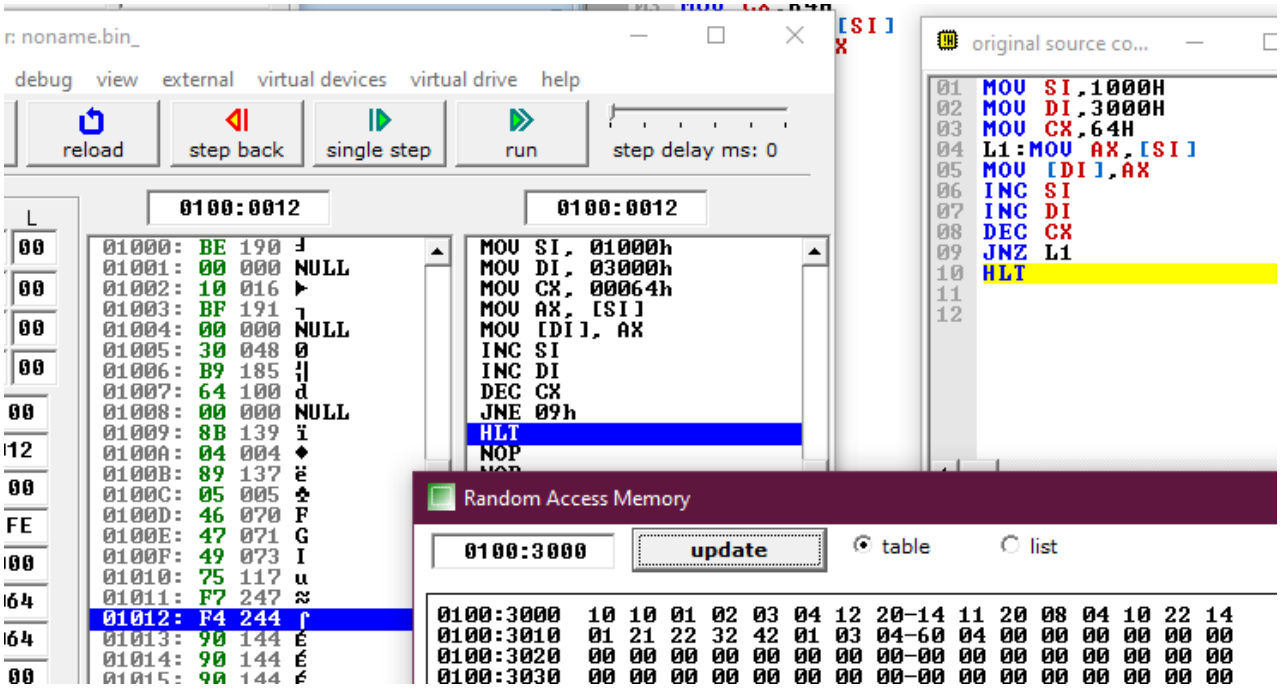
(4)

Mnemonics	Comments
MOV DX,0x1234	Loads 1234h to DX
MOV CX,0x5678	Loads 5678h to DX
PUSH DX	----------
PUSH CX	----------
POP DX	----------
POP CX	----------
HLT	Halt



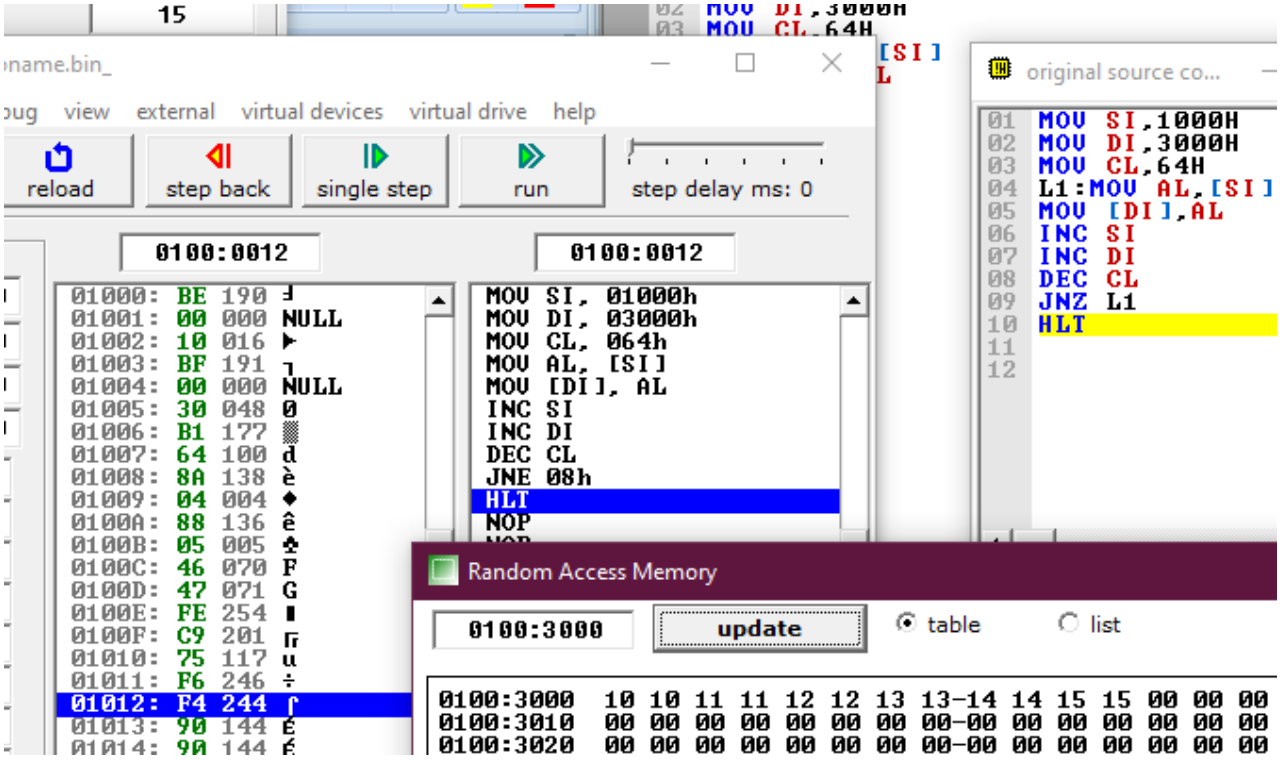
(5)

Mnemonics	Comments
MOV AX,0x5000	load the value 5000h to AX register
MOV DS,AX	load the value of AX register into DS
MOV ES,AX	load the value of AX register into ES
MOV SI,1000H	load the value 1000h into offset SI.
MOV DI,3000H	load the value 3000h into offset DI
MOV CX,64H	load the value 64h into CX
L1:MOV AX,[SI]	load the value SI into AL
MOV [DI],AX	load the value AX into DI
INC SI	Increment the offset SI by 1
INC DI	Increment the offset DI by 1
DEC CX	Decrement CX by 1
JNZ L1	Jump if Not equal to zero
HLT	Halt



(6)

Mnemonics	Comments
MOV AX,0x5000	load the value 5000h to AX register
MOV DS,AX	load the value of AX register into DS
MOV ES,AX	load the value of AX register into ES
MOV SI,1000H	load the value 1000h into offset SI.
MOV DI,3000H	load the value 3000h into offset DI
MOV CL,64H	load the value 64h into CL
L1:MOV AL,[SI]	load the value SI into AL
MOV [DI],AL	load the value AL into DI
INC SI	Increment the offset SI by 1
INC DI	Increment the offset DI by 1
DEC CL	Decrement CL by 1
JNZ L1	Jump if Not equal to zero
HLT	Halt



IV.Conclusion

Thus, all the programs were executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 07

Arithmetic and logical operation related programmingusing 8086

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives

1. Develop a short sequence of instructions that adds AX, BX,CX,DX and SP. Save the sum in DI register
2. Select the correct instruction to perform each of the following task
    a. Rotate all the bits of AL left three places.
    b. Move all bits in AL left one place, making sure that 0 movesinto the rightmost bit position.
    c. Rotate carry right one place through EDX.
3. If DL=0F3H and BH= 72H, list the difference after BH is subtracted from DL and show the contents of the ag register.
4. Write a sequence of instructions that cube the 8-bit number found in DL. Load DL with 05 initially, and make sure that your result is a 16-bit number.
5. Write a sequence of instructions that divides the number in BL by the number in CL and then multiplies the result by 2. The final answer must be a 16-bit number stored in the DX register.
6. Develop a short sequence of instructions that clears the three leftmost bits of DH without changing the remainder of DH and stores the result in BH.

II.Lab Component

Requirements:

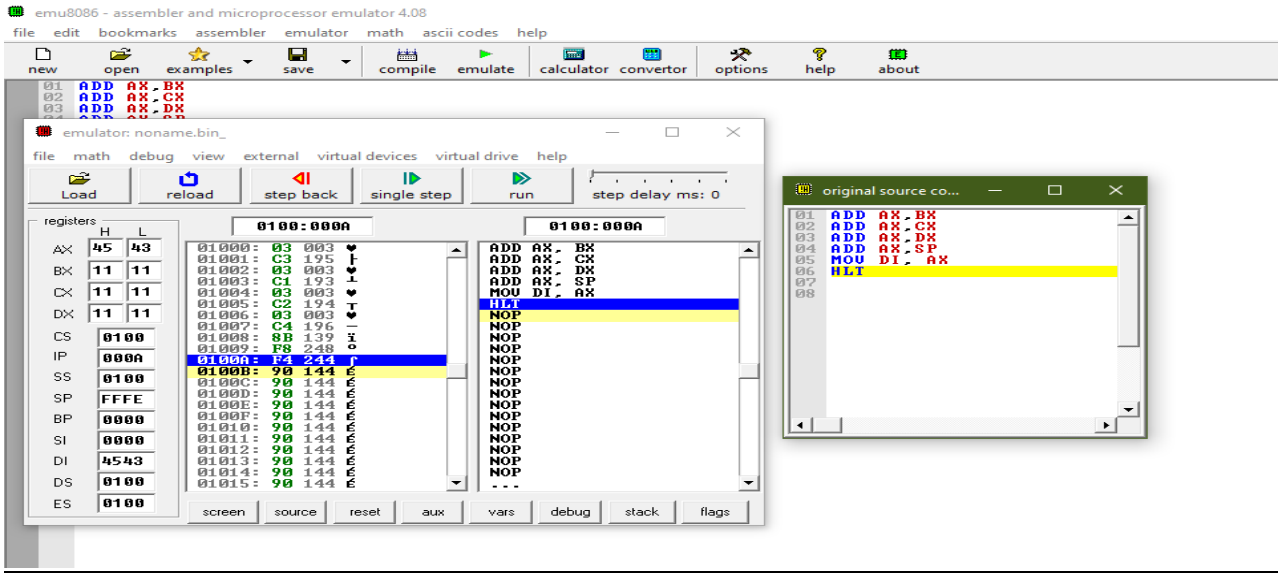
13) PC
14) EMU8086

III.Programs& Outputs

1.

Mnemonics	Comments
ADD AX, BX	Adding contents in AX and BX
ADD AX, CX	Adding contents in AX and CX
ADD AX, DX	Adding contents in AX and DX
ADD AX, SP	Adding contents in AX and SP
MOV DI, AX	Moves data in AX to DI
HLT	Halt

OUTPUT:-



2.

(a) ROL AL, 0x0003

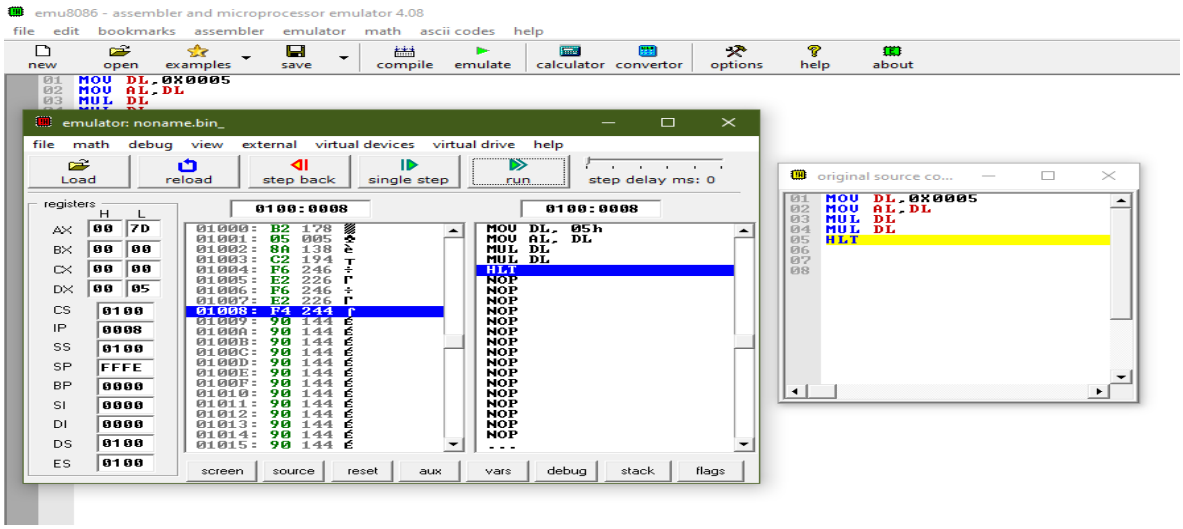
(b) SHL AL, 0x0001

(c) RCR EDX, 0x0001

4.

Mnemonics	Comments
MOV DL, 0x0005	Moves data in memory address 0x0005 to DL
MOV AL, DL	Moves data in DL to AL
MUL DL	Unsigned Multiplication at DL
MUL DL	Unsigned Multiplication at DL
HLT	Halt

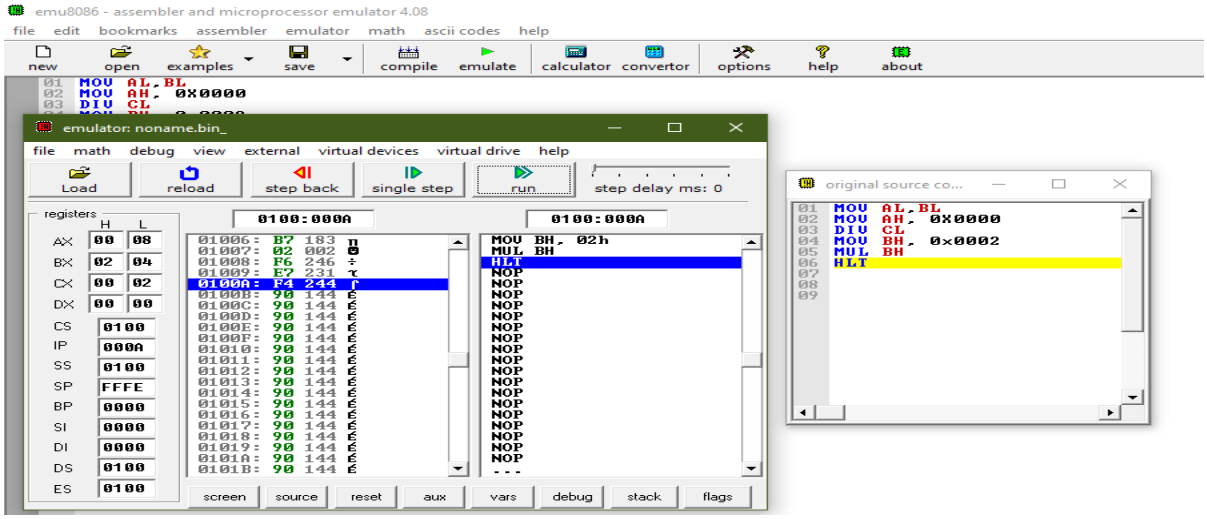
OUTPUT:-



5.

Mnemonics	Comments
MOV AL, BL	Moves data from BL to AL
MOV AH, 0x0000	Moves data in memory address 0x0000 to AH
DIV CL	Unsigned Division at CL
MOV BH, 0x0002	Moves data in memory address 0x0002 to BH
MUL BH	Unsigned multiplication at BH
HLT	Halt

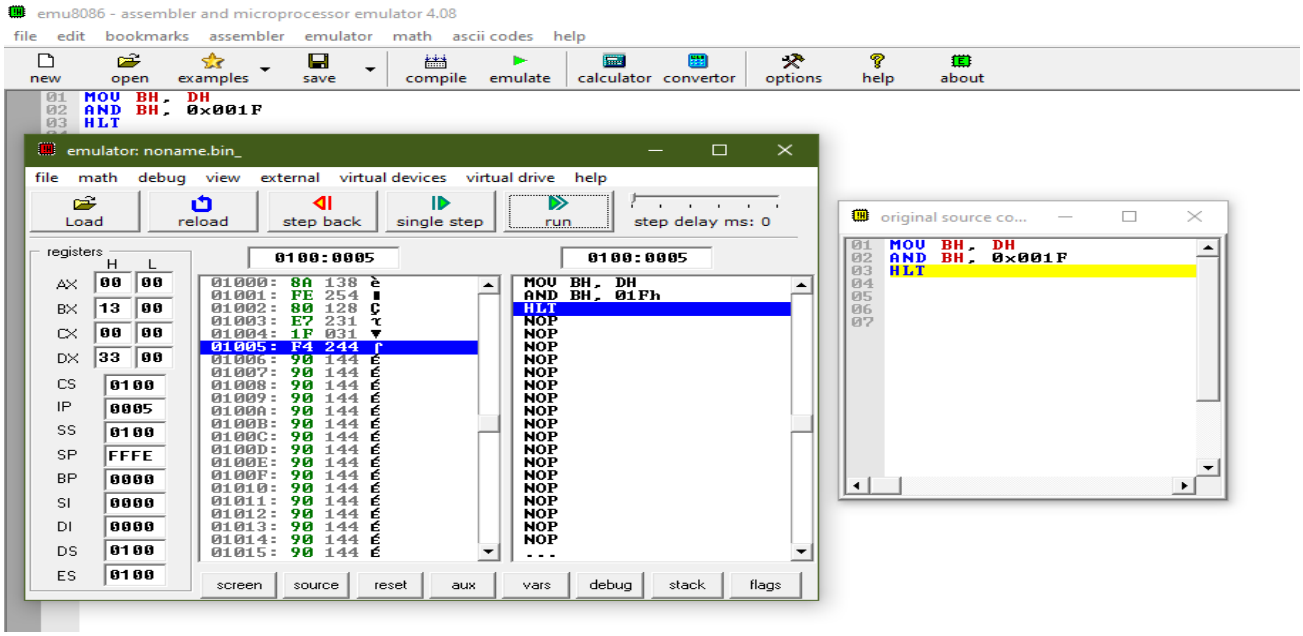
OUTPUT:-



6.

Mnemonics	Comments
MOV BH, DH	Moves data from DH to BH
ANDBH, 0x001F	Performs logical AND operation on data in BH and 0x001F
HLT	Halt

OUTPUT:-



IV. Conclusion

In this given lab experiment arithmetic and logical operationrelated programming using 8086 was successfully executed.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 08

8086 Branching Operation Programming-I

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University
Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives (Minimum any three)

1. Write an Assembly language program (ALP) for 8086 to find the average of N 16-bit numbers.
2. Write an ALP for 8086 to sort an array in ascending/descending order.
3. Write an ALP for 8086 to sort an array into odd/even numbers.
4. Write an ALP for 8086 to find the sum of odd numbers in a series of N numbers.
5. Write an ALP for 8086 to determine the sum of corresponding elements of two arrays.
6. Write an ALP for 8086 to find the product of corresponding elements of two arrays.

II.Lab Component

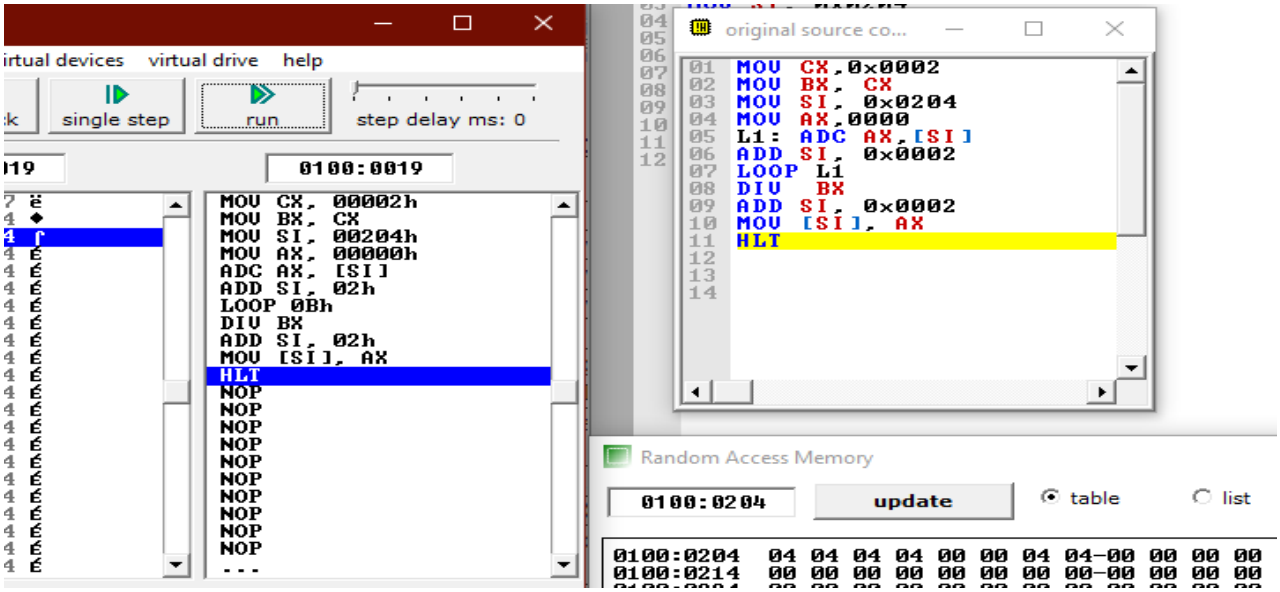
Requirements:

15) PC
16) EMU8086

III.Programs& Outputs

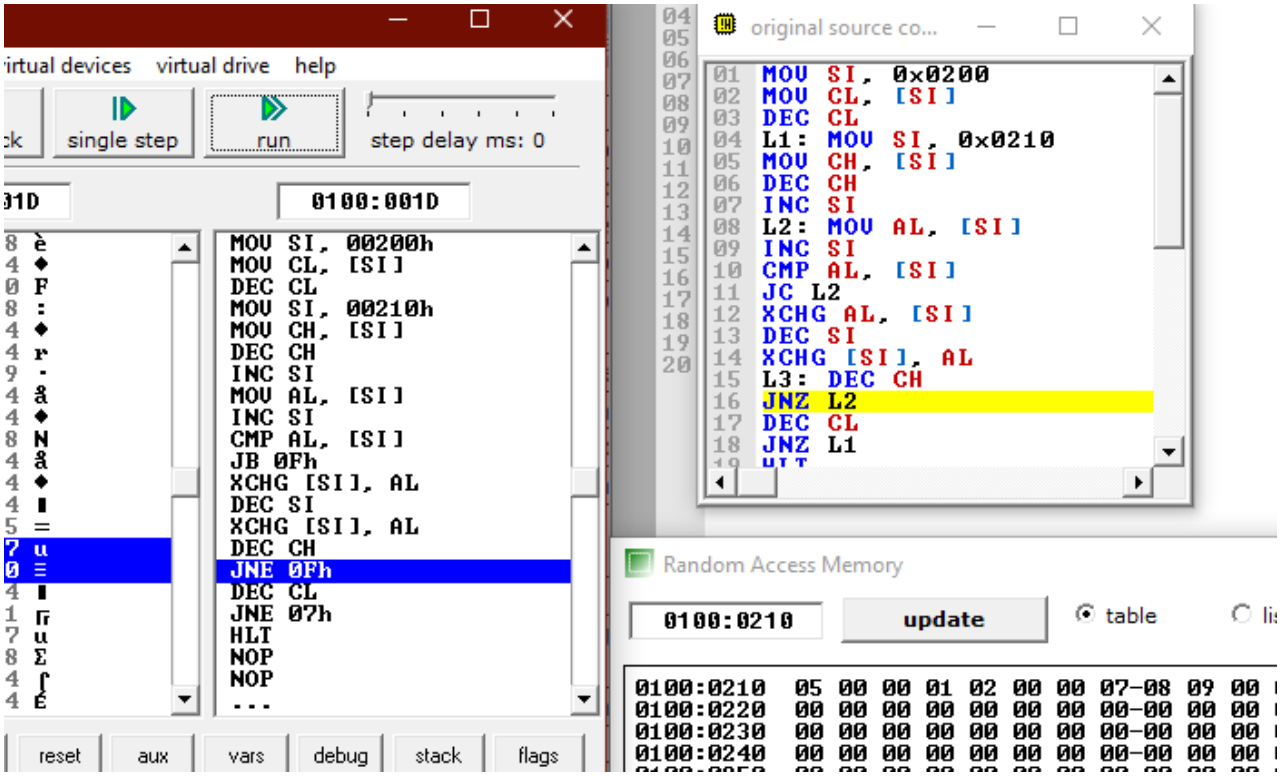
(1)

Mnemonics	Comments
MOV CX,0x0002	Data moved from 0×0202 to CX
MOV BX, CX	Data moved from CX to BX
MOV SI, 0×0204	Data moved from 0×0204 to SI
MOV AX,0000	AX initialized to zero
L1: ADC AX,[SI]	Add along with carry with that of data in [SI] & store it in AX
ADD SI, 0×0002	SI is moved to next memory location
LOOP L1	Jump to L1
DIV BX	Data in AX divided with that of BX
ADD SI, 0×0002	SI made to go to next memory location
MOV [SI], AX	Data moved from AX to [SI]
HLT	End of program



(2)

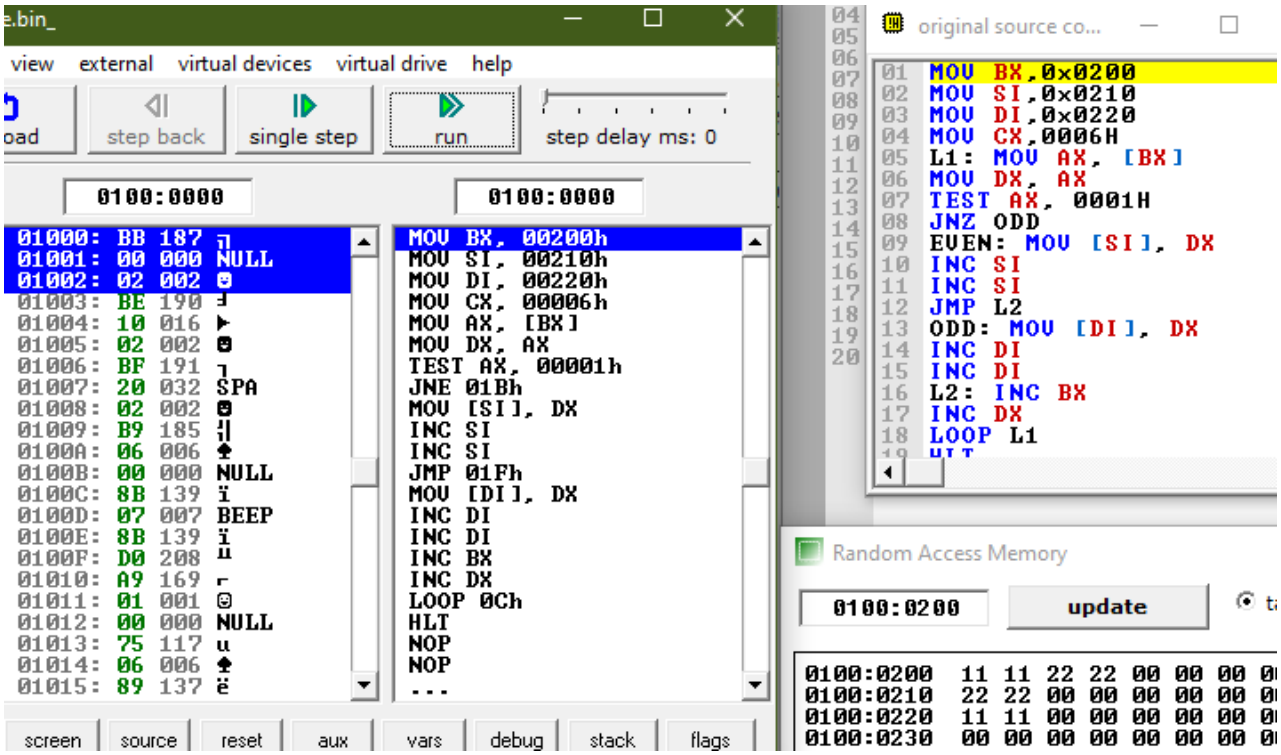
Mnemonics	Comments
MOV SI, 0×0200	0×0200 data moved to SI
MOV CL, [SI]	Data moved from memory location [SI] to CL
DEC CL	CL decreased by 1
L1: MOV SI, 0×0300	0×0200 data moved to SI
MOV CH, [SI]	Data moved from memory location stored in SI to CH.
DEC CH	CH decreased by 1.
INC SI	SI increased by 1.
L2: MOV AL, [SI]	Data moved from memory location SI to AL
INC SI	SI increased by 1
CMP AL, [SI]	AL compared with data stored in memory location address stored in SI.
JC L3	Check if carry flag is set.
XCHG AL, [SI]	Exchange AL with data stored in memory location address stored in SI.
DEC SI	SI increased by 1.
XCHG [SI], AL	
L3: DEC CH	CH decreased by 1.
JNZ L2	Jump if zero flag is reset.
DEC CL	CL decreased by 1.
JNZ L1	Jump if carry flag is reset.
HLT	End of program.



(3)

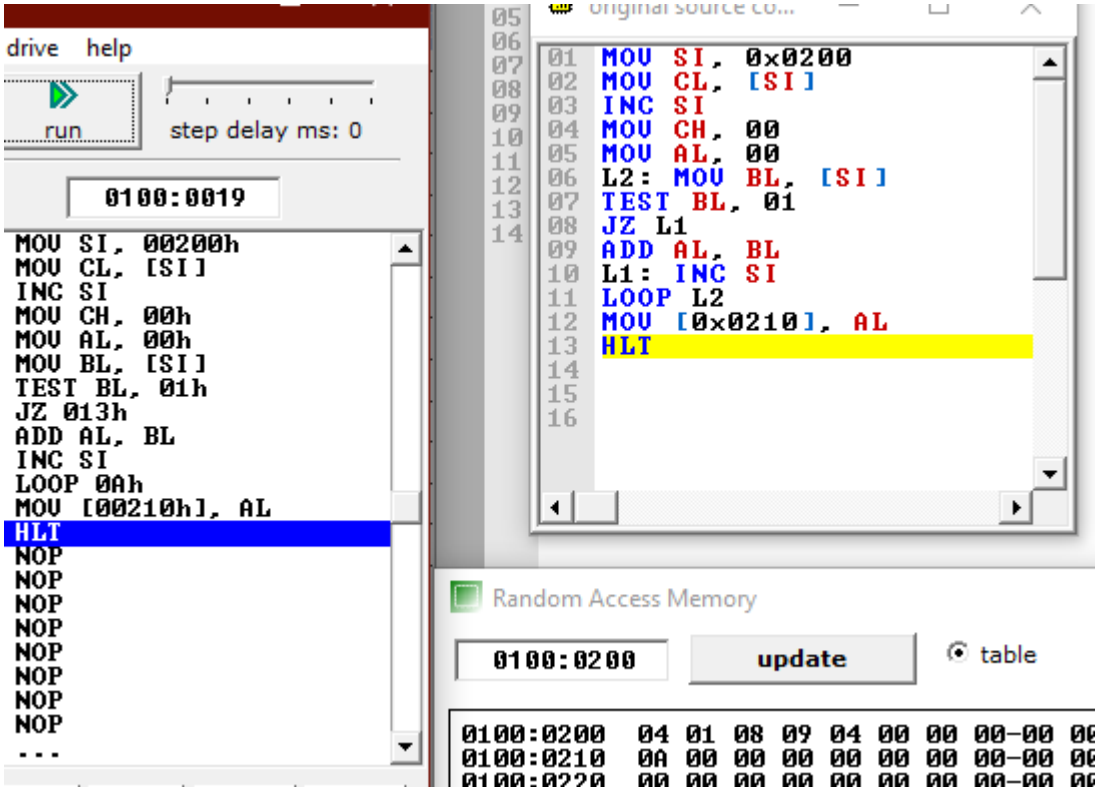
Mnemonics	Comments
MOV BX, 2000H	2000H data moved to BX.
MOV SI, 2010H	2010H data moved to SI.
MOV DI, 2020H	2020H data moved to DI.
MOV CX, 0006H	0006H data moved to CX.
L1: MOV AX, [BX]	Data moved from memory location address stored in BX to AX.
MOV DX, AX	Copy data in AX to BX.
TEST AX, 0001H	
JNZ ODD	Jump if carry flag in reset.
EVEN: MOV [SI], DX	Data moved from DX to memory location address stored in SI.
INC SI	SI increased by 1.
INC SI	SI increased by 1.
JMP L2	Jump to the label.
ODD: MOV [DI], DX	Data moved from DX to memory location address stored in DI.
INC DI	DI increased by 1.
INC DI	DI increased by 1.
L2: INC BX	BX increased by 1.
INC DX	DX increased by 1.
LOOP L1	Jump to L1.

HLT	End of program.



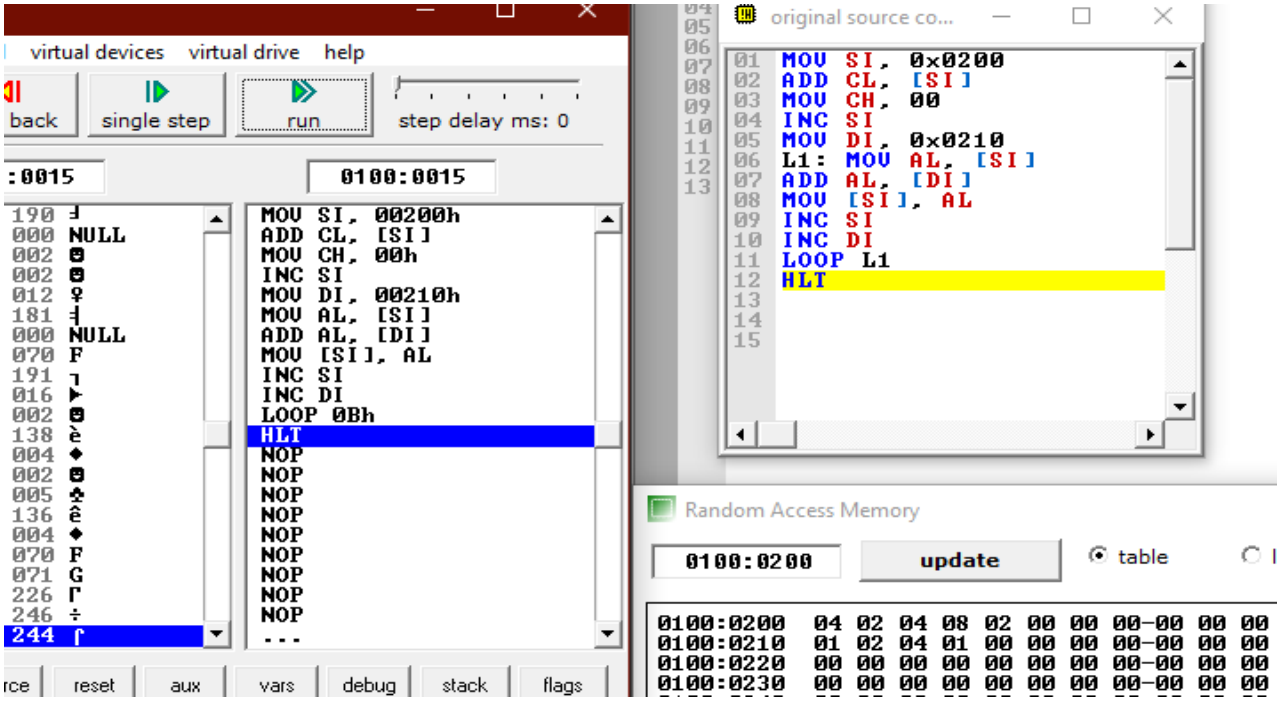
(4)

Mnemonics	Comments
MOV SI, 0×0200	0×0200 data moved to SI.
MOV CL, [SI]	Data moved from memory location address stored in SI to CL.
INC SI	SI increased by 1.
MOV CH, 00	Zero is stored in CH.
MOV AL, 00	Zero is stored in AL.
L2: MOV BL, [SI]	Data moved from memory location address stored in SI to BL.
TEST BL, 01	
JZ L1	Jump to label if zero flag is set.
ADD AL, BL	AL= AL+BL
L1: INC SI	SI increased by 1.
LOOP L2	Jump to L2.
MOV [0×0210], AL	Data moved from AL to memory location 0×0300
HLT	End of program.



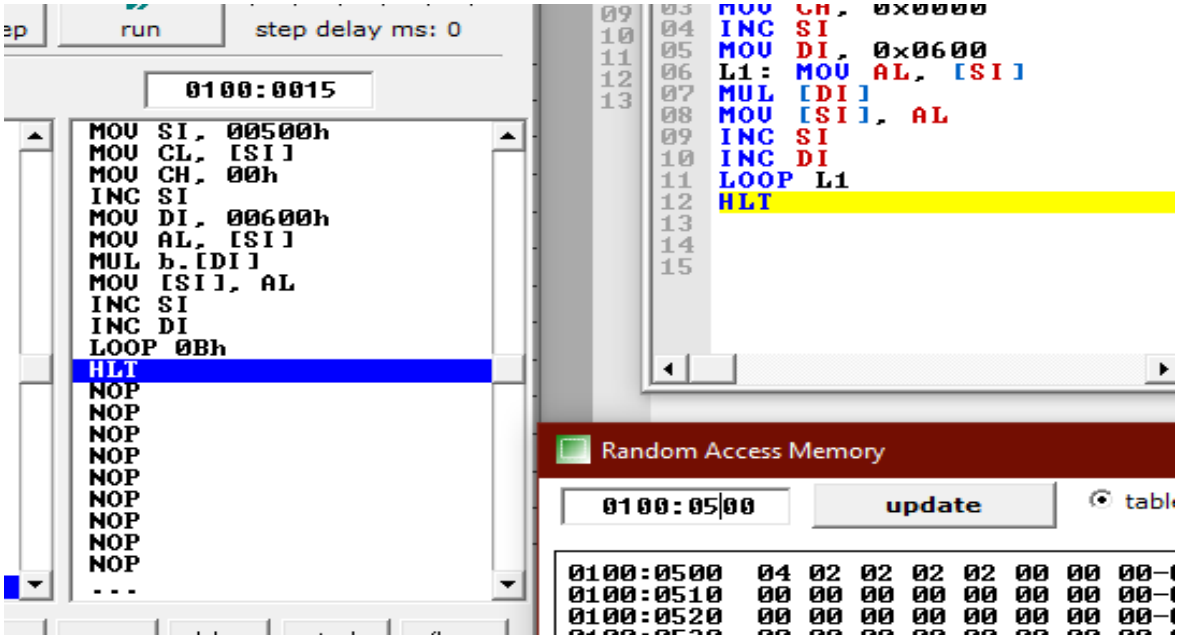
(5)

Mnemonics	Comments
MOV SI, 0×0200	0×0202 data moved to SI.
ADD CL, [SI]	Data moved from memory location SI to CL.
MOV CH, 00	Zero is stored in CH.
INC SI	SI increased by 1.
MOV DI, 0×0210	0×0302 data moved to DI.
L1: MOV AL, [SI]	Data moved from memory location address stored in SI to AL.
ADD AL, [DI]	AL= AL+ Data stored in memory location address in DI
MOV [SI], AL	Data moved from AL to memory location address stored in SI.
INC SI	SI increased by 1.
INC DI	DI increased by 1.
LOOP L1	Jump to L1.
HLT	End of program.



(6)

Mnemonics	Comments
MOV SI, 0×0500	0×0500 data moved to SI.
MOV CL, [SI]	Data moved from memory location address stored in SI to CL.
MOV CH, 0×0000	Zero is stored in CH.
INC SI	SI increased by 1.
MOV DI, 0×0600	0×0600 data moved to DI.
L1: MOV AL, [SI]	Data moved from memory location address stored in SI to AL.
MUL [DI]	AL=AL×[Data in memory location address stored in DI]
MOV [SI], AL	Data in AL copied to memory location address stored in SI.
INC SI	SI increased by 1.
INC DI	DI increased by 1.
LOOP L1	Jump to L1.
HLT	End of program.



IV. Conclusion

Thus, from this lab experiment we dealt with programs involving conditional statements in 8086. We also got an idea about the way to deal with arrays in 8086. The mistakes committed and the corrections made with due help of my professor were really helpful in learning new things.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 09

8086 Branching Operation Programming-II

Department of Electronics & Communication Engineering

Siksha 'O' Anusandhan Deemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives (Minimum any three)

1. Write an ALP for 8086 to generate the Fibonacci series.
2. Write an ALP for 8086 to generate A.P series of N numbers.
3. Write an ALP for 8086 to generate G.P series of N numbers.
4. Write an ALP for 8086 to find the square root of a perfect square number.
5. Write an ALP for 8086 to print the table of an input integer.
6. Write an ALP for 8086 to find the factorial of any number up to 12.

II.Lab Component

Requirements:

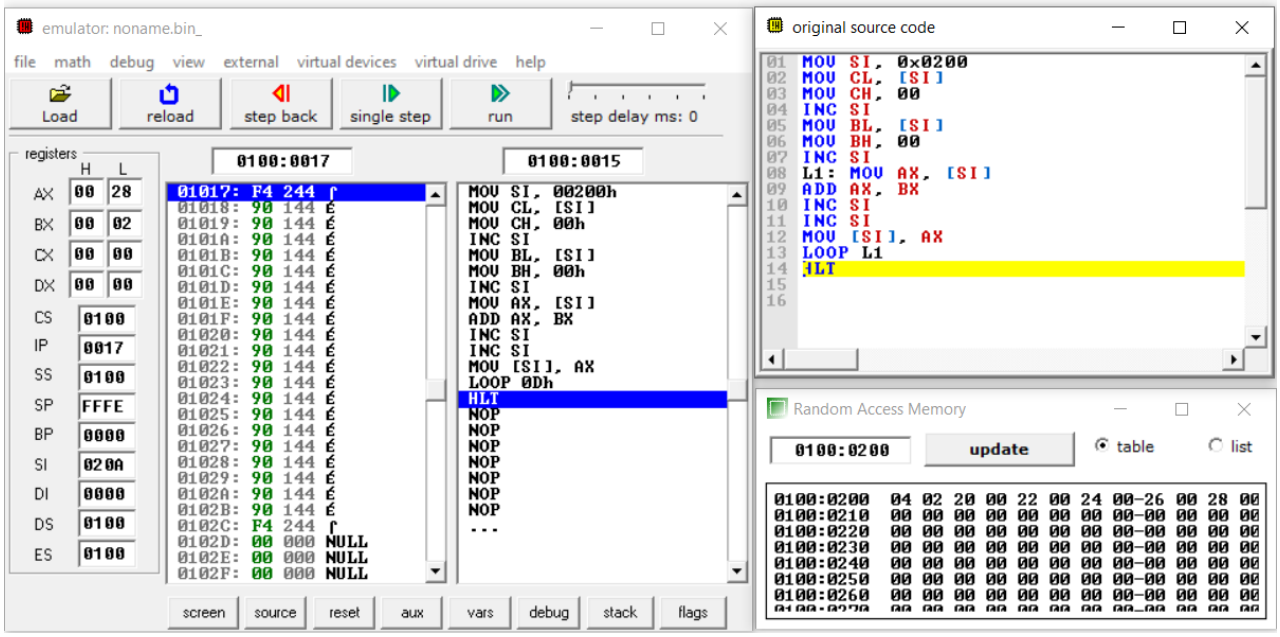
17) PC
18) EMU8086

III.Programs& Outputs

(2)

Mnemonics	Comments
MOV SI, 0x0200	Moves data to SI from memory location 0x0200
MOV CL, [SI]	Moves data from SI to CL
MOV CH, 00	Moves 00H data to CH
INC SI	Data in SI is incremented
MOV BL, [SI]	Moves data from SI to BL
MOV BH, 00	Moves 00H data to BH
INC SI	Data in SI is incremented
L1: MOV AX, [SI]	Moves data from SI to AX
ADD AX, BX	Adds data in AX and BX
INC SI	Data in SI is incremented
INC SI	Data in SI is incremented
MOV [SI], AX	Moves data in SI from AX
LOOP L1	Runs loop to L1
HLT	Halt

Program Output:



Results:

Memory Location / Register	Input Data
0x0200	04
0x0201	02
0x0202	20

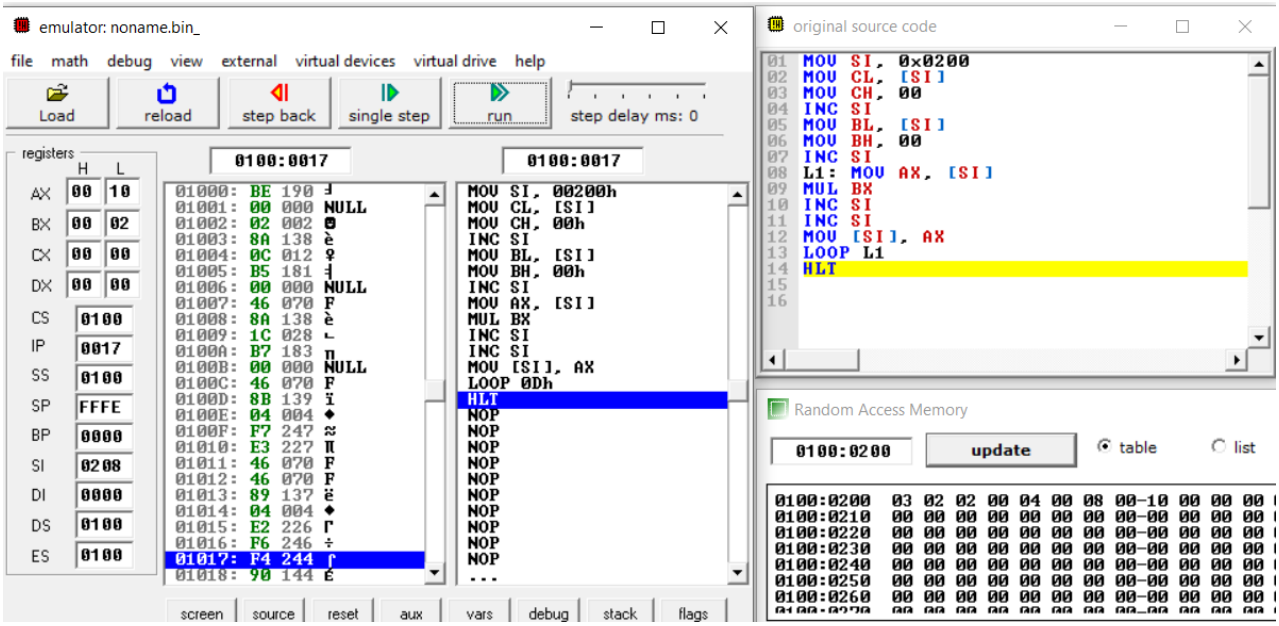
Memory Location / Register	Output Data
0x0204	22
0x0206	24
0x0208	26
0x020A	28

(3)

Mnemonics	Comments
MOV SI, 0x0200	Moves data to SI from memory location 0x0200
MOV CL, [SI]	Moves data from SI to CL
MOV CH, 00	Moves 00H data to CH
INC SI	Data in SI is incremented
MOV BL, [SI]	Moves data from SI to BL
MOV BH, 00	Moves 00H data to BH
INC SI	Data in SI is incremented
L1: MOV AX, [SI]	Moves data from SI to AX
MUL BX	Multiplies data in BX
INC SI	Data in SI is incremented
INC SI	Data in SI is incremented
MOV [SI], AX	Moves data in SI from AX
LOOP L1	Runs loop to L1

HLT	Halt

Program Output:



Results:

Memory Location / Register	Input Data
0x0200	03
0x0201	02
0x0202	02

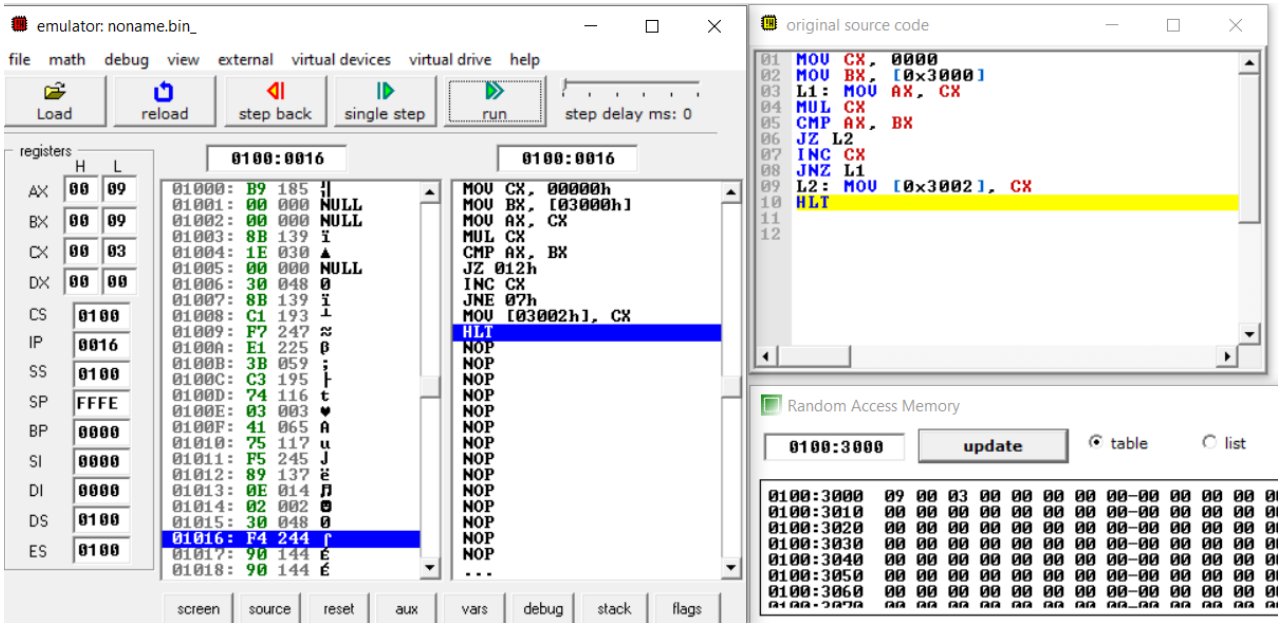
Memory Location / Register	Output Data
0x0204	04
0x0206	08
0x0208	10

(4)

Mnemonics	Comments
MOV CX, 0000	Moves data 00H to CX
MOV BX, [0x3000]	Moves data in memory location 0x3000 to BX
L1: MOV AX, CX	Moves data in CX to AX
MUL CX	Multiplies data in CX
CMP AX, BX	Compare data in AX and BX
JZ L2	Jump to L2 if equals to zero
INC CX	Increments the value of CX
JNZ L1	Jump to L1 if not equal to zero
L2:MOV [0x3002], CX	Moves data from CX to memory location 0x3002

HLT	Halt

Program Output:



Results:

Memory Location / Register	Input Data
0x3000	09

Memory Location / Register	Output Data
0x3002	3

IV. Conclusion

Programs related branching operations are executed successfully.

EMBEDDED SYSTEM PROJECT (EET4100)

Lab 10

String data type programming in 8086

Department of Electronics & Communication Engineering

Siksha 'O' AnusandhanDeemed to be University Bhubaneswar

Branch: ECE Section: E		
Name	**Registration No.**	**Signature**
ANSHUMAN DASH	**1741016004**	Anshuman Dash

I.Objectives(Minimum any three)

1. Write an Assembly language program (ALP) for 8086 to move a string of 10 bytes from 0100:0200h to 0100:0500h
2. Write an ALP for 8086 to find the length of a string.
3. Write an ALP for 8086 to reverse a string of 10 bytes.
4. Write an ALP for 8086 to search for a data byte in a string of 10 bytes.
5. Write an ALP for 8086 to compare two data strings.
6. Write an ALP for 8086 to search for a data byte in a string of 10 bytes using SCASB/SCASW instruction.

II.Lab Component

Requirements:

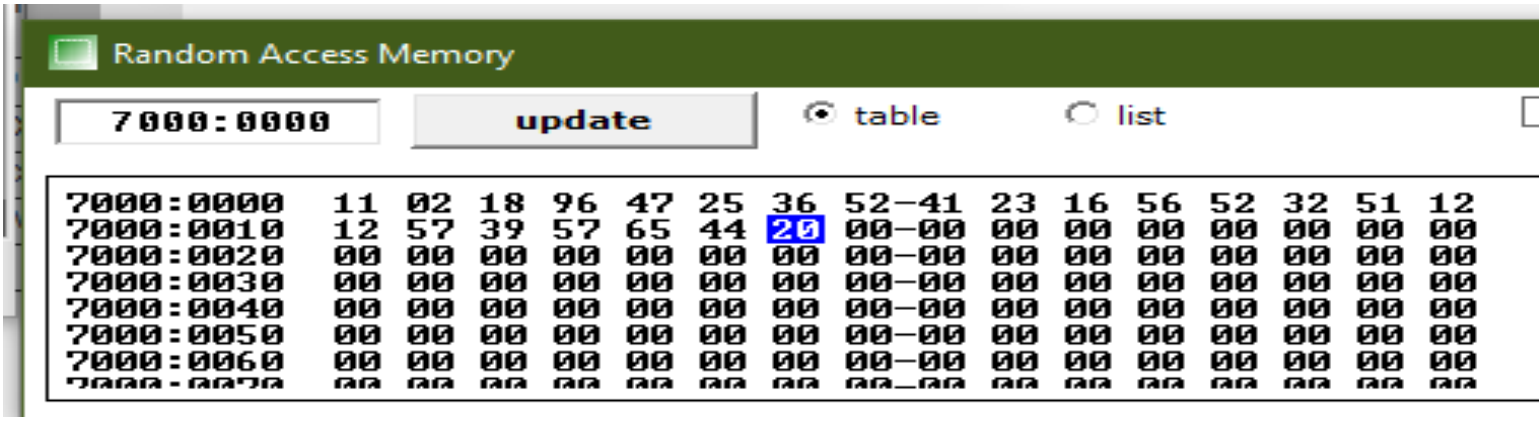
1)PC

2)EMU8086

III.Programs& Outputs

(1)

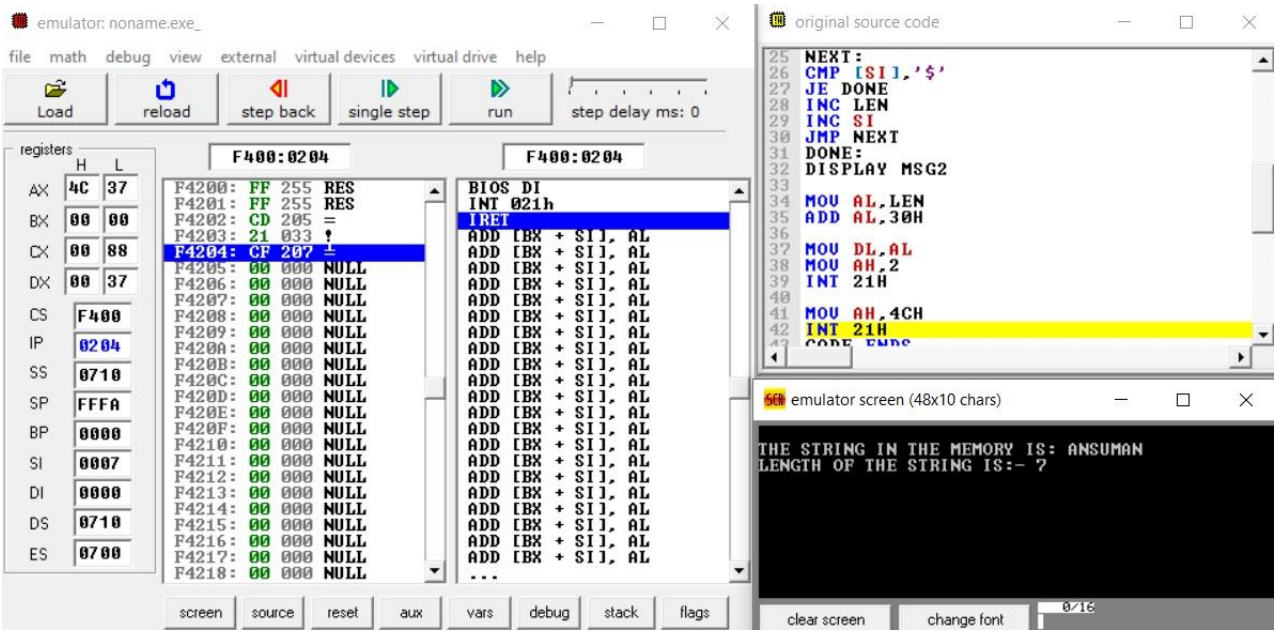
Mnemonics	Comments
MOV AX,0x5000	Moves data to AX from memory location 0x5000
MOV DS,AX	Moves data from AX to DX
MOV AX,0x7000	Moves data to AX from memory location 0x0200
MOV ES,AX	Moves data from AX to EX
MOV SI,0x0000	Moves data to SI from memory location 0x0000
MOV DI,0x0000	Moves data to DI from memory location 0x0000
MOV EX,0x0020	Moves data to EX from memory location 0x020
CLD	Clears the direction flag
REP: MOVSB	Move the byte/word from one string to another.
HLT	Halt



(2)

```
DATA SEGMENT
STR DB ‘ANSHUMAN$’
MSG1 DB 10,13,’THE STRING IN THE MEMORY IS : $’
MSG2 DB 10,13,’LENGTH OF THE STRING IS :- $’
LEN DB 0H
DATA ENDS

DISPLAY MACRO MSG
MOV AH,9
LEA DX,MSG
INT 21H
ENDM

CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX

DISPLAY MSG1

DISPLAY STR

LEA SI,STR
NEXT:
CMP [SI],'$'
JE DONE
INC LEN
INC SI
JMP NEXT
DONE:
DISPLAY MSG2

MOV AL,LEN
ADD AL,30H
MOV DL,AL
MOV AH,2
INT 21H
```

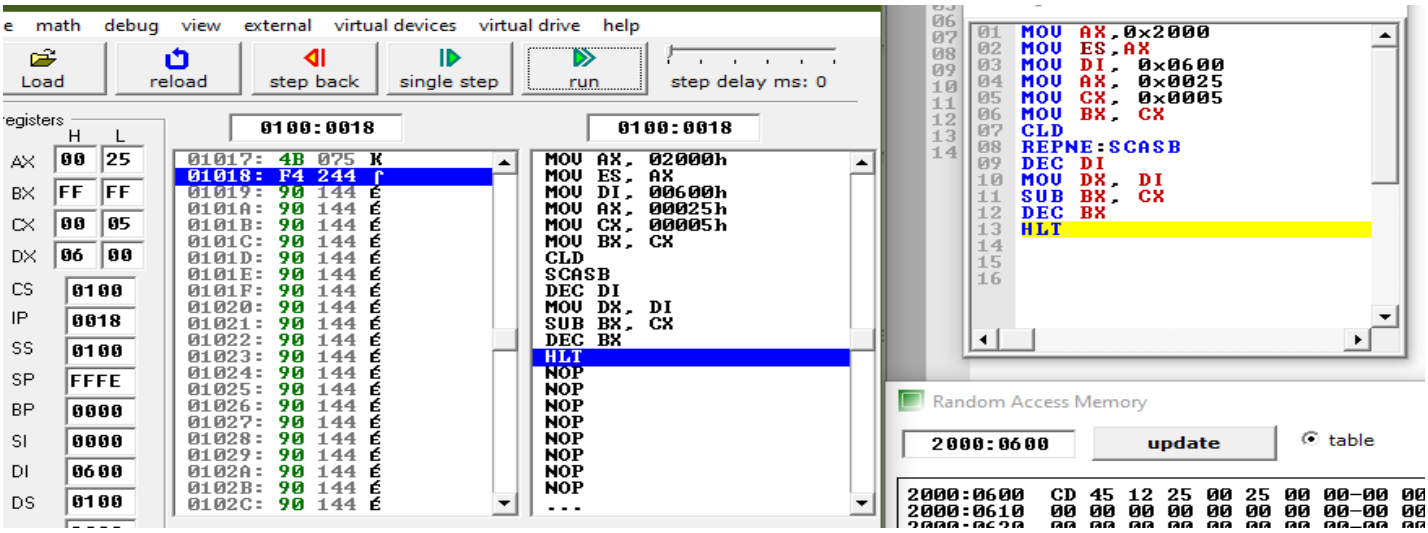
```
MOV AH,4CH
INT 21H
CODE ENDS

END START
```



(6)

Mnemonics	Comments
MOV AX,0x2000	Moves data to AX from memory location 0x2000
MOV ES,AX	Moves data from AX to EX
MOV DI, 0x0600	Moves data to DI from memory location 0x0600
MOV AX, 0x0025	Moves data 25h to AX
MOV CX, 0x0005	Moves data 05h to CX
MOV BX, CX	Moves data CX to BX
CLD	Clears the direction Flag
REPNE :SCASB	Scans the data
DEC DI	Decrement DI by 1
MOV DX, DI	Moves DI to DX
SUB BX, CX	BX=BX-CX

DEC BX	Decrement BX by 1
HLT	Halt



IV. Conclusion

All the programs have been successfully executed. In this lab we have done the operation on string using 8086.