

School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

<u>LAB – 01: 8086 – Arithmetic Operations (Addition</u> <u>and Subtraction)</u>

LABTASK – 1: 8 – BIT ADDITION

AIM:

To write 8086 Assembly language program to add two 8-bit number in AL, BL registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

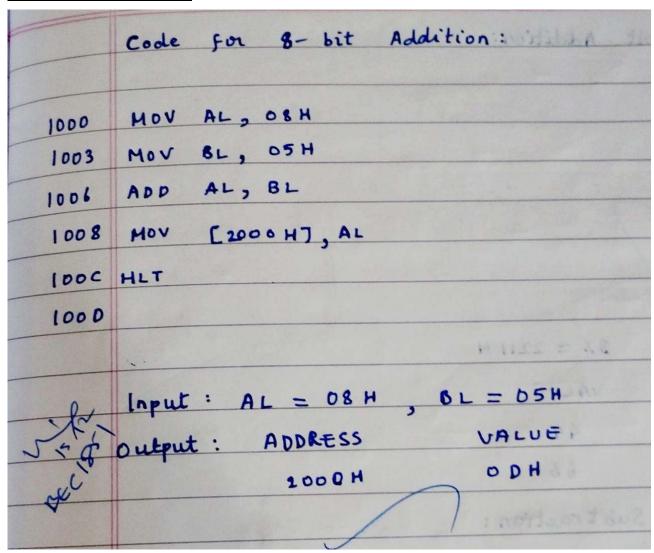
- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL,08H	Move data 08H to AL register
1003	MOV BL,05H	Move data 05H to BL register
1006	ADD AL, BL	Add AL and BL content
1008	MOV [2000H], AL	Move AL to the memory location 2000H

100C	ИΙТ	Halt the program
1000	ПLI	Halt the program

Address	Value
2000H	0DH

OUTPUT VERIFICATION:



INFERENCE:

The output is found out to be "0DH" and stored in the address 2000H.

RESULT:

Thus the 8086 Assembly language program to add two 8-bit number in AL, BL registers and store the result in memory location 2000H is written and executed successfully.

LABTASK - 2: 16 - BIT ADDITION

AIM:

To write 8086 Assembly language program to add two 16-bit number in AX, BX registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AX,4433H	Move data 4433H to AX register
1004	MOV BX,2211H	Move data 2211H to BX register
1008	ADD AX, BX	Add AX and BX content
100A	MOV [2000H], AX	Move AX to the memory location 2000H
100E	HLT	Halt the program

OUTPUT:

Address	Value
2000Н	44

2001H	66

OUTPUT VERIFICATION:

2	Code fo	16- bit	Addition:
	19810881		Contract to the same
1000	MOV AX	,4433 H	
1004	Mov 8x	, 2211 H	
(008	S ADD	AX, BX	
LODA		2000 H], AX	
100E	HLT		
100F		/	
0/1	Input:	AX = 44 33 H	8 × = 2211 H
ring.	Output:	AX = 44 33 H ADDRESS 2000H	VALUE
3kg		2000H	44
		2001 H	66 114 0

INFERENCE:

The output is found out to be "6644H" and lower byte is stored in the address 2000H and higher byte is stored in the address 2001H.

RESULT:

Thus the 8086 Assembly language program to add two 16-bit number in AX, BX registers and store the result in memory location 2000H is written and executed successfully.

LABTASK – 3: 8 – BIT SUBTRACTION

AIM:

To write 8086 Assembly language program to subtract two 8-bit number in AL, BL registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL,08H	Move data 08H to AL register
1003	MOV BL,05H	Move data 05H to BL register
1006	SUB AL, BL	Subtract AL and BL content
1008	MOV [2000H], AL	Move AL to the memory location 2000H
100C	HLT	Halt the program

OUTPUT:

Address	Value
2000H	03H

OUTPUT VERIFICATION:

	2 code for 8-bit Subtraction:
1000	MOV AL, 08H MOV BL, 05H
1006	FSUB AL, BL
1008	MOV (2000 H), AL
(00 D	H ILES = X 8
<:600 85	Input: AL = 08H, BL = 05H Output: ADDRESS VALUE
usuch	2000 H 03H

INFERENCE:

The output is found out to be "03H" and it is stored in the address "2000H".

RESULT:

Thus the 8086 Assembly language program to subtract two 8-bit number in AL, BL registers and store the result in memory location 2000H is written and executed successfully.

LABTASK - 4: 16 - BIT SUBTRACTION

AIM:

To write 8086 Assembly language program to subtract two 16-bit number in AL, BL registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AX,4433H	Move data 4433H to AX register
1004	MOV BX,2211H	Move data 2211H to BX register
1008	SUB AX, BX	Subtract AX and BX content
100A	MOV [2000H], AX	Move AX to the memory location 2000H
100E	HLT	Halt the program

OUTPUT:

Address	Value
2000H	44

2001H	66

OUTPUT VERIFICATION:

-	code	Jon	16 - bit	Subtra	ction:		
1000	Mov	A ×	4422H		,		
			4433H				1
1008	SUB	AX,					1
loo A	MOV	(20	00 H] , AX				
looE	}	LT		/			. 4
100F							
10	Input	: Ax	= 44331 ADDRESS 2000H		B x =	2211 H	l
vilves!	Outpu	t:	ADDRESS		, , ,	ALVE	-
Jec.			2000H		• • • • • • • • • • • • • • • • • • • •	22	9.77
			2001H		1 4	22	4 2

INFERENCE:

The output is found out to be "2222H" and the lower byte is stored in the address 2000H and higher byte is stored in the address 2001H.

RESULT:

Thus the 8086 Assembly language program to subtract two 16-bit number in AL, BL registers and store the result in memory location 2000H is written and executed successfully.



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

<u>LAB – 02: 8086: Arithmetic Operations</u> (<u>Multiplication and Division</u>)

LABTASK - 1: 8 - BIT MULTIPLICATION

AIM:

To write 8086 Assembly language to multiply two 8-bit number in AL, BL registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL,08H	Move data 08H to AL register
1003	MOV BL,05H	Move data 05H to BL register
1006	MUL BL	Multiply AL and BL content

1008	MOV [2000H], AX	Move AX to memory location 2000H
100C	HLT	Halt the program

Address	Value
2000Н	28H
2001H	00H

OUTPUT VERIFICATION:

	8 - bit Multiplication	
1000	MOV AL, OHH	12.2
1003	MOV BL, OSH	0.7
1006	MUL BL	
1008	MOV [2000H], AL AX	* 1 *
1000	HLT	
100D		
20	Input : AL = 08H , BL = 05H	, by
San og	Output: ADDRESS VALUE	
sec 85)	2000H 28	
Υ	2001H 00	

INFERENCE:

The output is found out to be "28H" and stored in the address 2000H.

RESULT:

Thus the 8086 Assembly language program to multiply two 8-bit number in AL, BL registers and store the result in memory location 2000H is written and executed successfully.

LABTASK-2: 16 – BIT MULTIPLICATION

AIM:

To write 8086 Assembly language to multiply two 16-bit number in AX, BX registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AX,4433H	Move data 08H to AX register
1004	MOV BX,2211H	Move data 05H to BX register
1008	MUL BX	Multiply AX and BX content
100A	MOV [2000H], AX	Move AX to memory location 2000H
100E MOV [2002H], DX Move DX to memory location 2002H		Move DX to memory location 2002H
1012	HLT	Halt the program

OUTPUT:

Address	Value
2000H	63H
2001H	4DH
2002H	13H
2003H	09H

OUTPUT VERIFICATION:

						1.13
16- b	it Multipl	iration				
MoV	AX, 4433	Н				1
MoV	Bx, 2211	H				
MUL	Вх					
Mov	[2000H],	ΑX				
MOV	[2002 H],	Νď				
HLT					, k	, 1
lneu	t: AY =	l. /. 20 H	B V - 00 I		χ	0
Outo			VALUE	П		
\$	20	00 H	63	10	1 1	
	20	01 H	40			Laa
	20	02 H	13		١,	
	مه	03 H	99		11	
	MOV MOV MOV HLT	MOV AX, 4433 MOV BX, 2211 MUL BX MOV [2000H], MOV [2002H], HLT Input: AX = Output: & ADD 200 200	MOV AX, 4433 H MOV BX, 2211 H MUL BX MOV [2000H], AX MOV [2002 H], DX HLT Input: AX = 4433 H	MOV BX, 2211 H MUL BX MOV [2000H], AX MOV [2002H], DX HLT Input: AX = 4433H, BX = 2211 Output: B ADDRESS VALUE 2000 H 4D 2002 H 13	MOV AX, 4433 H MOV BX, 2211 H MUL BX MOV [2000H], AX MOV [2002H], DX HLT Input: AX = 4433 H, BX = 2211 H Output: & ADDRESS VALUE 2000 H 4D 2002 H 13	MOV AX, 4433 H MOV BX, 2211 H MUL BX MOV [2000H], AX MOV [2002H], DX HLT Input: AX = 4433 H, BX = 2211H Output: B ADDRESS VALUE 2000 H 4D 2002 H 13

INFERENCE:

The output is found out to be "0913 4D63H" and lower byte of AX is stored in the address 2000H, higher byte of AX is stored in the address 2001H, lower byte of DX is stored in the address 2002H and higher byte of DX is stored in the address 2003H.

RESULT:

embly language progration 2		

LABTASK - 3: 8 - BIT DIVISION

AIM:

To write 8086 Assembly language to divide two 8-bit number in AL, BL registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

A – Line Assembler

GO - Execution

SB – To view output

U – Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL,08H Move data 08H to AL register	
1003	MOV BL,05H	Move data 05H to BL register
1006	1006 DIV BL Divide AL and BL content	
1008	MOV [2000H], AX	Move AX to memory location 2000H
100C	OC HLT Halt the program	

OUTPUT:

Address	Value

2000H	01H
2001H	03H

OUTPUT VERIFICATION:

8-	bit	Division
-	1700	0 , , , , , , , , , , , , , , , , , , ,

1000	Mov	AL, 08H
1003	Mov	BL, 05 H
1006	DIV	BL
1008	MOV	[2000H], A X
1000	HLT	

Input: AL = D8H, BL = 05H

Output: ADDRESS YALUF

2000H

INFERENCE:

The quotient is found out to be "01H" and stored in the address 2000H and the remainder is found out to be "03H" and stored in the address 2001H.

RESULT:

Thus the 8086 Assembly language program to divide two 8-bit number in AL, BL registers and store the result in memory location 2000H is written and executed successfully.

LABTASK - 4: 16- BIT DIVISION

AIM:

To write 8086 Assembly language to divide two 16-bit number in AX, BX registers and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

PROGRAM:

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL,4433H	Move data 08H to AL register
1004	MOV BL,2211H	Move data 05H to BL register
1008	DIV BX	Multiply AL and BL content
100A	MOV [2000H], AX	Move AX to memory location 2000H
100E	MOV [2002H], DX	Move DX to memory location 2002H
1012	HLT	Halt the program

OUTPUT:

Address	Value
2000H	02H
2001H	00H
2002H	11H
2003H	00H

OUTPUT VERIFICATION:

 10-	bit	Division	

1000	Mov	AX,	4433 H
1004	Mov	В×,	2211 H
1008	DIV	Bx	

1012 HLT

> Input , Ax = 4433 H BX = 2211 H

Output: ADDRESS VALUE 2000 H 02

2001H

2002 H

00

2003 00

INFERENCE:

The lower byte of quotient is found out to be "02H", higher byte of quotient is found out to be "00H" and stored in the address 2000H and 2001H respectively. The lower byte of remainder is found out to be "11H", higher byte of remainder is found out to be "00H" and stored in the address 2002H and 2003H respectively. **RESULT:** Thus the 8086 Assembly language program to divide two 16-bit number in AX, BX registers and store the result in memory location 2000H is written and executed successfully.

BECE204P - Microprocessors and Microcontrollers Lab

Page No: 11

[21BEC1851]



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

LAB – 03: 8086: Sum and Average of "N" 8-bit Numbers, Factorial and Celcius to Fahrenheit

LABTASK – 1: SUM OF "N" 8-BIT NUMBER

AIM:

To write 8086 Assembly language to find sum of "N" 8-bit number and store the result in memory location 0600H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV AX, 0000	Clear AX register
100C	MOV CL, [SI]	Load the block size (Value of N)
100E	INC SI	Increment SI to point next memory location
100F	ADD AL, [SI]	Add AL and data pointed by SI register
1011	ADC AH, 00	Add AH and 00H along with carry
1014	INC SI	Increment SI to point next memory location
1015	DEC CL	Decremet CL value by 1
1017	JNZ 100F	If Z=0, jump to BACK label
1019	MOV [DI], AX	Store the result of the division into the memory
		location pointed by DI
101B	HLT	Stop the execution

Address	Value
0600H	15H
0601H	00H

OUTPUT VERIFICATION:

	Lab-1: Sum	-
1000	MOV SI, 5000H 0500H	
1004	MOV DJ, 6000 H 0600 H	
008	MOV AX, 0000	
1006	MOV CL, [SI]	
100 1	INC SI	
100 F	ADD AL, [SI]	
1011	ADC AH, OD	
1014	INC SI	
1015	DEC CL	
10 [7	JN\$ 100F	
1019	MOV [DI], AX	
10 l B	HLT	
1018		
1018	HLT INPUT ADDRESS	VALUE
1018	Input	VALUE OSH
1018	Input ADDRESS	
1018	ADDRESS 0500 H	05H
1018	ADDRESS 0500 H 0501 H	05H ⁰ 4H ⁰ 2 H
	ADDRESS 0500 H 0501 H 0502 H	05H 04H
	ADDRESS 0500 H 0501 H 0502 H	05H 04H 02 H 09H
	ADDRESS 0500 H 0501 H 0502 H	05H 04H 02 H 09H 01H
1018	ADDRESS 0500 H 0501 H 0502 H	05H 04H 02 H 09H 01H
	ADDRESS 0500 H 0501 H 0502 H 0503 H 0504 H	05H 04H 02 H 09H 01H

INFERENCE:

The output is found out to be "15H" and stored in the address 0600H.

RESULT:

Thus the 8086 Assembly language program to find the sum of "N" 8-bit number is written and executed successfully.

LABTASK – 2: AVERAGE OF "N" 8-BIT NUMBER

AIM:

To write 8086 Assembly language to find average of "N" 8-bit number and store the result in memory location 0600H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV AX, 0000H	Clear AX register
100C	MOV CL, [SI]	Load the block size (Value of N)
100E	MOV BL, CL	Also store into BL
1010	INC SI	Increment SI to point next memory location
1011	ADD AL, [SI]	Add AL and data pointed by SI register
1013	ADC AH, 00H	Add AH and 00H along with carry
1016	INC SI	Increment SI to point next memory location
1017	DEC CL	Decremet CL value by 1
1019	JNZ BACK	If Z=0, jump to BACK label
101B	DIV BL	Otherwise divide it with BL
101D	MOV [DI], AX	Store the result of the division into the memory
		location pointed by DI
101F	HLT	Stop the execution

Address	Value
0600H	04H
0601H	01H

OUTPUT VERIFICATION:

	Averag	ı:		
1000		SI, 05	DO H	
1004		DI, 06		
1008	MOV			
100C	Mov	AX CL		
100€.	MOV	اک و ∟8		
(olo	INC	31		
1011	ADD	AL,[SI]	
1013	A DC	Αн,	00 H	
1016	1 NC	s I		
1017	DEC	CL		
1019	JNZ	В		
1018	DIN	BL		
lolD	Mov	(DI),	Ax	
locF	HLT			
የዕ ታር				
	Input:		Value	
	DE an H		05 H	

	Input:	Value		
	D500 H	054		
	0501 4	044		
5/12	0502 H	02H		
2 1 48	05034	/ 09 H		
350 45	05044	/ 01 H		
	0505H	054		
	Output:			
	0601 H	04H	(a)	
	060 2 H	ØLH	(R)	

INFERENCE:

The output is found out to be "04H" (Quotient) and "01H" (Remainder) stored in the address 0600H and 0601H respectively. RESULT: Thus the 8086 Assembly language program to find the average of "N" 8-bit number is written and executed successfully.

<u>LABTASK – 3: FACTORIAL</u>

AIM:

To write 8086 Assembly language to find the factorial and to store in the memory location 0400H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV CX, 04H	Load number whose factorial is to be found I CX
1004	MOV AX, 0001H	Load AX with 0001H
1008	MOV BX, AX	Copy AX into BX
100A	INC BX	Increment BX value by 1
100B	MUL BX	Multiply $AX*BX = DX:AX$
100D	CMP BX, CX	Compare BX with CX i.e BX-CX
100F	JNZ BACK	If Z is not zero, jump to BACK Label
1011	MOV [4000H], AX	Store AX register content to memory location 4000H
1015	HLT	Stop the execution

Address	Value
4000H	18H
4001H	00H

OUTPUT VERIFICATION:

	Factorial:
1000	MOV CX, 04H
1004	MOV AX, ODDIH
1008	MOV BX AX
100A	INC BX
100 B	MUL BX
1000	CMP BX, CX
100F	JNZ POA
1011	MOV [4000H], AX
1015	HLT
1016	
	Input : Cx = 04H
, ,	output:
J. 6/23	4000 H 18H
05 1/1 V	4001 H 00H
NEE 187	•
V	

INFERENCE:

The value is found out to be "18H" and stored in the address 4000H.

RESULT:

Thus the 8086 Assembly language program to find factorial is written and executed successfully.

LABTASK – 4: CELCIUS TO FAHRENHEIT

AIM:

To write 8086 Assembly language to convert from celcius to fahreheit and store the result in memory location 2000H.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV AL, 19H	Load 25 degree (19 H) Celsius value into AL register
1003	MOV BL, 09H	Load BL with 09H
1006	MUL BL	Multiply BL with 25 degree celsius value and store the
		result in AX
1008	MOV CL, 05H	Move 05H to CL register
100B	DIV CL	Divide AX by CL, Quotient in AL and remainder in
		AH.
100D	MOV DL, 20H	Move 20H to DL register
1010	ADD AL, DL	Add DL along with AL (Quotient)

1012	MOV [2000H], AL	Store the final result available in AL into memory
		location 2000H
1016	HLT	Stop the execution

Address	Value
2000H	4DH

OUTPUT VERIFICATION:

	Celcius to Fahrenheit:	
1000	MOV AL, 19H	
1003	MOV BL, O9H	
1006	MUL BL	
8001	MOV CL, OSH	
100 B	DIV CL	
100 D	MOV DL, 10H	
1010	ADD AL,DL	
1012	MOV [2000H], AL	
1016	HLT	
101 7		
J1257	Input: Ax = 19H	
5 45	Output:	
50	2000 H : 4D	

INFERENCE:

The output "4DH" is stored in the memory location 2000H.

RESULT:

Thus the 8086 Assembly language program to convert from celcius to fahrenheit is written and executed successfully.



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

LAB – 04: 8086: Smallest and Largest Number in an Array, Sorting an array in ascending and decending order

<u>LABTASK – 1: SMALLEST NUMBER IN AN ARRAY</u>

AIM:

To write 8086 Assembly language to find smallest number in an array.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV CL, [SI]	Load the block size (Value of N)
100A	INC SI	Increment SI to point next memory location
100B	MOV AL, [SI]	Add AL and data pointed by SI register
101D	DEC CL	Decrement CL value by 1
101F	INC SI	Increment SI to point next memory location
1010	MOV BL, [SI]	Get next byte of the array in BL register
1012	CMP AL, BL	Compare smallest data AL with next byte BL of the
		array
1014	JC 1018	If C=1, jump to 1018
1016	MOV AL, BL	Move BL to AL
1018	DEC CL	Decrement CL value by 1
101A	JNZ 100F	If Z=0, then go to 100F otherwise go to next step
101C	MOV [DI], AL	Move smallest data AL to DI
101E	HLT	Stop the execution

Address	Value
0600H	01H

OUTPUT VERIFICATION:

ab Task 1 1	Smallest Number	
ddress	Memonics	
0 00	MOV SI, 0500H	
1004	MOV DI, OGOOH	
1008	MOV CL, [SI]	
100 A	INC SI	
1008	MOV AL, [SI]	
100 D	DEC CL	
100F	INC SI	
1010	MOV BL, [51]	
1012	CMP AL, BL	
1014	JC 1018	
1016	MOV AL, BL	
1018	DEC CL	
LOIA	TN 2 100F	
101 C	MOV [DI], AL	
IDIE	HLT	
1015		
Input "		
ADDRESS	Válue	
0 500 H	05	
0501 H	63	
0502H	01	
0503M	0.2	
05044	04	
0505 H		
output,	VALUE J. DE	,
ADDRESS	VALUE 7.00	1851
0 60 0 4	ol William	

INFERENCE:

The lowest value in the given array is "01H" which is stored in the memory 0600H.

RESULT:

Thus, the smallest number in an array code is written and executed successfully.

LABTASK - 2: LARGEST NUMBER IN AN ARRAY

AIM:

To write 8086 Assembly language to find largest number in an array.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV CL, [SI]	Load the block size (Value of N)
100A	INC SI	Increment SI to point next memory location
100B	MOV AL, [SI]	Add AL and data pointed by SI register
101D	DEC CL	Decrement CL value by 1
101F	INC SI	Increment SI to point next memory location
1010	MOV BL, [SI]	Get next byte of the array in BL register
1012	CMP AL, BL	Compare smallest data AL with next byte BL of the
		array
1014	JNC 1018	If C=0, jump to 1018
1016	MOV AL, BL	Move BL to AL
1018	DEC CL	Decrement CL value by 1
101A	JNZ 100F	If Z=0, then go to 100F otherwise go to next step
101C	MOV [DI], AL	Move smallest data AL to DI
101E	HLT	Stop the execution

Address	Value
0600H	05H

```
Lab Task-2: Largest Number
 Address
 1000
                MOV SI, 0500H
                 MOV DI, OLOOH
 1004
 1008
                 MOV CL, [S]
 LODA
                 INC 31
  1008
                  MOV AL, [SI]
  1000
                  DEC CL
  100F
                  INC SI
  1010
                  MOV BL, [51]
   1012
                  INC 1018
   1014
                  MOV AL, BL
   1016
                  DEC CL
   1018
                   TNZ 100F
   IDIA
                  MOV [DI], AL
    101 C
                  HLT
    IDIE
    JOIF
     Input :
                                             VALUE
                   Value
      ADDRESS
                                            1 05
                    05
                                 0 600 H
      0500 H
                    05
      0502H
                    01
      0503H
                    02
      05044
      0505 H
```

INFERENCE:

The lowest value in the given array is "05H" which is stored in the memory 0600H.

RESULT:

The largest value in an array code is written and executed successfully.

<u>LABTASK – 3: SORTING AN ARRAY IN ASCENDING ORDER</u>

AIM:

To write 8086 Assembly language to sort the array in the ascending order.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Set SI as a pointer for array
1004	MOV DI, 0600H	Set CL register as count for N-1 comparison (outer
		loop)
1006	DEC CL	Decrement CL register by 1
1008	MOV SI, 0500H	Initialize SI register as array pointer
100C	MOV CH, [SI]	Load CH and the value of SI
100E	DEC CH	Decrement CH value by 1
1010	INC SI	Increment the array pointer
1011	MOV AL, [SI]	Get the first element of array in AL register
1013	CMP AL, [SI]	Compare the next element of the array with AL

1016	JC 101D	Check carry flag if C = 1 go to 101D otherwise go to
		next step
1018	XCHG AL, [SI]	Exchange the content of memory pointer by SI and the
		content of memory location i.e. AL
101A	XCHG AL, [SI-1]	Exchange the content of memory pointer by SI - 1 and
		the content of memory location i.e. AL
101D	DEC CH	Decrement the count for comparison (CH register)
101F	JNZ 1011	Check zero flag if $Z = 0$ go to 1011 otherwise go to
		next step
1021	DEC CL	Decrement the count for repetition
1023	JNZ 1008	Check the zero flag if $Z = 0$, go to 1008 Otherwise go
		to next step
1025	HLT	Halt the execution

Address	Value
0500H	05H
0501H	01H
0502H	02H
0503H	03H
0504H	04H
0505H	05H

```
Lab Task 3: Sorting an array in ascending order
 Address
               Memonics
  1000
               MOV SI, 1100H
  1004
               MOV CL, [SI]
   1006
               DEC CL
                MOV 51, 1100 H
   100 $
   1000
                MOV
                     CH, [SI]
   ODE
                DEC CH
    1010
                INC
                     AL, [SI]
    1011
                MOV
    1013
                INC SI
    1014
                CMP AL, [SI]
                JC ANEAD HOLD TOLD
    1016
                XCHG AL, [SI]
    1018
                 XCHG AL, [SI-1]
    101A
     10 10
                 DEC CH
     IOIF
                 JN2 $ 1011
                                                 7 . 112
     1021
                       CL
                 DEC
     1023
                 TNZ
                      1008
      1025
                 HLT
     10 26
                      output!
   Input:
                     - 0500 : 05
    0500 : 05
                       0504: 01
    0501 :03
                      05021 02
    0502 105
                      0503:03
    0503 :01
                       0504: 04
    0504 :02
                       0505:05
    0505 : 24
```

INFERENCE:

The array is sorted and stored in the address 0500H.

RESULT: Thus the 8086 Assembly language program to sort the array in ascending order is written and executed successfully.

<u>LABTASK – 4: SORTING AN ARRAY IN DESCENDING ORDER</u>

AIM:

To write 8086 Assembly language to sort the array in the descending order.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Set SI as a pointer for array
1004	MOV DI, 0600H	Set CL register as count for N-1 comparison (outer
		loop)
1006	DEC CL	Decrement CL register by 1
1008	MOV SI, 0500H	Initialize SI register as array pointer
100C	MOV CH, [SI]	Load CH and the value of SI
100E	DEC CH	Decrement CH value by 1
1010	INC SI	Increment the array pointer
1011	MOV AL, [SI]	Get the first element of array in AL register
1013	CMP AL, [SI]	Compare the next element of the array with AL

1016	JNC 101D	Check carry flag if C = 0 go to 101D otherwise go to
		next step
1018	XCHG AL, [SI]	Exchange the content of memory pointer by SI and the
		content of memory location i.e. AL
101A	XCHG AL, [SI-1]	Exchange the content of memory pointer by SI - 1 and
		the content of memory location i.e. AL
101D	DEC CH	Decrement the count for comparison (CH register)
101F	JNZ 1011	Check zero flag if $Z = 0$ go to 1011 otherwise go to
		next step
1021	DEC CL	Decrement the count for repetition
1023	JNZ 1008	Check the zero flag if $Z = 0$, go to 1008 Otherwise go
		to next step
1025	HLT	Halt the execution

Address	Value
0500H	05H
0501H	05H
0502H	04H
0503H	03H
0504H	02H
0505H	01H

```
Lab Task 4 . Sorting an array in descending order
```

```
Address
            Memonics
1000
            MOV SI, 1100 H
 1004
            MOV CL, [SI]
 1006
             DEC CL
             MOV SI, 1100H
   1008
  100 (
             MOV CH, [SI]
             DEC CH
  LOOE
             INC SI
  1010
             MOV AL, [SI]
   1011
             INC SI
   1013
              CMP AL, [Si]
   1014
    1016
              INC 101D
              XCHG AL, [SI]
    1018
              XCHG AL, [SI-1]
    10 1 A
              DEC CH
    1010
              INZ 1011
    IDIF
     1021
               DEC CL
     1023
               JNZ 1008
               HLT
      1025
      1026
```

(. 1851 851

```
Input: Output:

0500:05

0500:05

0500:05

0501:05

0501:05

0502:04

0502:04

0503:03

0504:02

0505:01
```

INFERENCE:

The array is sorted and stored in the address 0500H.

RESULT:

Thus the 8086 As	sembly language program to sort the array in decending order is w	ritten and
executed successfully.		
[21BEC1851]	BECE204P - Microprocessors and Microcontrollers Lab	Page No: 15



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

LAB – 05: 8086: Smallest and Largest Number in an Array, Sorting an array in ascending and decending order

LABTASK - 1: SMALLEST NUMBER IN AN ARRAY

AIM:

To write 8086 Assembly language to find smallest number in an array.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

Important terms:

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV CL, [SI]	Load the block size (Value of N)
100A	INC SI	Increment SI to point next memory location
100B	MOV AL, [SI]	Add AL and data pointed by SI register
101D	DEC CL	Decrement CL value by 1
101F	INC SI	Increment SI to point next memory location
1010	MOV BL, [SI]	Get next byte of the array in BL register
1012	CMP AL, BL	Compare smallest data AL with next byte BL of the
		array
1014	JC 1018	If C=1, jump to 1018
1016	MOV AL, BL	Move BL to AL
1018	DEC CL	Decrement CL value by 1
101A	JNZ 100F	If Z=0, then go to 100F otherwise go to next step
101C	MOV [DI], AL	Move smallest data AL to DI
101E	HLT	Stop the execution

Address	Value
0600H	01H

ab Task 1 1	Smallest Number	
ddress	Memonics	
0 00	MOV SI, 0500H	
1004	MOV DI, 0600 H	
1008	MOV CL, [SI]	
100 A	INC SI	
1008	MOV AL, [SI]	
100 D	DEC CL	
100F	INC SI	
1010	MOV BL, [51]	*
1012	CMP AL, BL	
1014	JC 1018	
1016	MOV AL, BL	
1018	DEC CL	
LOIA	TN Z 100F	
101 C	MOV [DI], AL	
IDIE	HLT	
1015		
Input :		
ADDRESS	Value	
0 500 H	05	
0501 M	63	
0501H	05	
0503M	02	
05044	04	
0505 H		
output,	4.	1.688 1851 21862.1851
ADDRESS	VALUE	1.00
0 60 0 4	01	UBU

INFERENCE:

The lowest value in the given array is "01H" which is stored in the memory 0600H.

RESULT:

Thus, the smallest number in an array code is written and executed successfully.

LABTASK - 2: LARGEST NUMBER IN AN ARRAY

AIM:

To write 8086 Assembly language to find largest number in an array.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Load 0500H into Source Index Register
1004	MOV DI, 0600H	Load 0600H into Destination Index Register
1008	MOV CL, [SI]	Load the block size (Value of N)
100A	INC SI	Increment SI to point next memory location
100B	MOV AL, [SI]	Add AL and data pointed by SI register
101D	DEC CL	Decrement CL value by 1
101F	INC SI	Increment SI to point next memory location
1010	MOV BL, [SI]	Get next byte of the array in BL register
1012	CMP AL, BL	Compare smallest data AL with next byte BL of the
		array
1014	JNC 1018	If C=0, jump to 1018
1016	MOV AL, BL	Move BL to AL
1018	DEC CL	Decrement CL value by 1
101A	JNZ 100F	If Z=0, then go to 100F otherwise go to next step
101C	MOV [DI], AL	Move smallest data AL to DI
101E	HLT	Stop the execution

Address	Value
0600H	05H

```
Lab Task-2: Largest Number
 Address
 1000
                MOV SI, 0500H
                 MOV DI, OLOOH
 1004
 1008
                 MOV CL, [S]
 LODA
                 INC 31
  1008
                  MOV AL, [SI]
  1000
                  DEC CL
  100F
                  INC SI
  1010
                  MOV BL, [51]
   1012
                  INC 1018
   1014
                  MOV AL, BL
   1016
                  DEC CL
   1018
                   TNZ 100F
   IDIA
                  MOV [DI], AL
    101 C
                  HLT
    IDIE
    JOIF
     Input :
                                             VALUE
                   Value
      ADDRESS
                                            1 05
                    05
                                 0 600 H
      0500 H
                    05
      0502H
                    01
      0503H
                    02
      05044
      0505 H
```

INFERENCE:

The lowest value in the given array is "05H" which is stored in the memory 0600H.

RESULT:

The largest value in an array code is written and executed successfully.

<u>LABTASK – 3: SORTING AN ARRAY IN ASCENDING ORDER</u>

AIM:

To write 8086 Assembly language to sort the array in the ascending order.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Set SI as a pointer for array
1004	MOV DI, 0600H	Set CL register as count for N-1 comparison (outer
		loop)
1006	DEC CL	Decrement CL register by 1
1008	MOV SI, 0500H	Initialize SI register as array pointer
100C	MOV CH, [SI]	Load CH and the value of SI
100E	DEC CH	Decrement CH value by 1
1010	INC SI	Increment the array pointer
1011	MOV AL, [SI]	Get the first element of array in AL register
1013	CMP AL, [SI]	Compare the next element of the array with AL

1016	JC 101D	Check carry flag if C = 1 go to 101D otherwise go to
		next step
1018	XCHG AL, [SI]	Exchange the content of memory pointer by SI and the
		content of memory location i.e. AL
101A	XCHG AL, [SI-1]	Exchange the content of memory pointer by SI - 1 and
		the content of memory location i.e. AL
101D	DEC CH	Decrement the count for comparison (CH register)
101F	JNZ 1011	Check zero flag if $Z = 0$ go to 1011 otherwise go to
		next step
1021	DEC CL	Decrement the count for repetition
1023	JNZ 1008	Check the zero flag if $Z = 0$, go to 1008 Otherwise go
		to next step
1025	HLT	Halt the execution

Address	Value
0500H	05H
0501H	01H
0502H	02H
0503H	03H
0504H	04H
0505H	05H

```
Lab Task 3: Sorting an array in ascending order
 Address
               Memonics
  1000
               MOV SI, 1100H
  1004
               MOV CL, [SI]
   1006
               DEC CL
                MOV 51, 1100 H
   100 $
   1000
                MOV
                     CH, [SI]
   ODE
                DEC CH
    1010
                INC
                     AL, [SI]
    1011
                MOV
    1013
                INC SI
    1014
                CMP AL, [SI]
                JC ANEAD HOLD TOLD
    1016
                XCHG AL, [SI]
    1018
                 XCHG AL, [SI-1]
    101A
     10 10
                 DEC CH
     IOIF
                 JN2 $ 1011
                                                 7 . 112
     1021
                       CL
                 DEC
     1023
                 TNZ
                      1008
      1025
                 HLT
     10 26
                      output!
   Input:
                     - 0500 : 05
    0500 : 05
                       0504: 01
    0501 :03
                      05021 02
    0502 105
                      0503:03
    0503 :01
                       0504: 04
    0504 :02
                       0505:05
    0505 : 24
```

INFERENCE:

The array is sorted and stored in the address 0500H.

RESULT: Thus the 8086 Assembly language program to sort the array in ascending order is written and executed successfully.

<u>LABTASK – 4: SORTING AN ARRAY IN DESCENDING ORDER</u>

AIM:

To write 8086 Assembly language to sort the array in the descending order.

PROCEDURE:

- 1. Connect the power cord and keyboard with the kit.
- 2. Switch on the power supply.
- 3. Press reset in the kit.
- 4. Type "A" in the keyboard and press enter. "Line assembler" will be displayed.
- 5. Starting address will be displayed in the kit. Type "1000" as the starting address.
- 6. Type the program and note the address of each line of the code till HLT.
- 7. Press reset in the kit.
- 8. Type "GO" [space] starting address (e.g. 1000) for execution.
- 9. Press enter in the keyboard.
- 10. "executing" message will be displayed.
- 11. Press reset in the kit.
- 12. Give "SB" [space] memory location (e.g. 2000) for execution.
- 13. Output will be displayed.

<u>Important terms:</u>

- A Line Assembler
- GO Execution
- SB To view output
- U Disassembly

ADDRESS	MEMONICS	COMMENTS
1000	MOV SI, 0500H	Set SI as a pointer for array
1004	MOV DI, 0600H	Set CL register as count for N-1 comparison (outer
		loop)
1006	DEC CL	Decrement CL register by 1
1008	MOV SI, 0500H	Initialize SI register as array pointer
100C	MOV CH, [SI]	Load CH and the value of SI
100E	DEC CH	Decrement CH value by 1
1010	INC SI	Increment the array pointer
1011	MOV AL, [SI]	Get the first element of array in AL register
1013	CMP AL, [SI]	Compare the next element of the array with AL

1016	JNC 101D	Check carry flag if $C = 0$ go to 101D otherwise go to
		next step
1018	XCHG AL, [SI]	Exchange the content of memory pointer by SI and the
		content of memory location i.e. AL
101A	XCHG AL, [SI-1]	Exchange the content of memory pointer by SI - 1 and
		the content of memory location i.e. AL
101D	DEC CH	Decrement the count for comparison (CH register)
101F	JNZ 1011	Check zero flag if $Z = 0$ go to 1011 otherwise go to
		next step
1021	DEC CL	Decrement the count for repetition
1023	JNZ 1008	Check the zero flag if $Z = 0$, go to 1008 Otherwise go
		to next step
1025	HLT	Halt the execution

Address	Value
0500H	05H
0501H	05H
0502H	04H
0503H	03H
0504H	02H
0505H	01H

```
Lab Task 4 . Sorting an array in descending order
```

```
Address
            Memonics
1000
            MOV SI, 1100 H
 1004
            MOV CL, [SI]
 1006
             DEC CL
             MOV SI, 1100H
   1008
  100 (
             MOV CH, [SI]
             DEC CH
  LOOE
             INC SI
  1010
             MOV AL, [SI]
   1011
             INC SI
   1013
              CMP AL, [Si]
   1014
    1016
              INC 101D
              XCHG AL, [SI]
    1018
              XCHG AL, [SI-1]
    10 1 A
              DEC CH
    1010
              INZ 1011
    IDIF
     1021
               DEC CL
     1023
               JNZ 1008
               HLT
      1025
      1026
```

(. 185) 185c 1851

```
Input: Output:

0500:05

0500:05

0500:05

0501:05

0501:05

0502:04

0502:04

0503:03

0504:02

0505:01
```

INFERENCE:

The array is sorted and stored in the address 0500H.

RESULT:

Thus the 8086 As	sembly language program to sort the array in decending order is w	ritten and
executed successfully.		
[21BEC1851]	BECE204P - Microprocessors and Microcontrollers Lab	Page No: 15



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

LAB – 06: INTRODUCTION TO KEIL IDE & ASSEMBLY PROGRAMMING WITH ARITHMETIC INSTRUCTION OF 8051

LABTASK-1: SOLVING THE MATHEMATICAL EQUATION

AIM:

To write an 8051 assembly language program to solve the given mathematical equation, W = (Y+3Z-6X)/6D

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV A, #12H

MOV B, #03H

MUL AB

MOV 50H, A

MOV A, #02H

MOV B, #06H

MUL AB

MOV 51H, A

MOV A, #00H

ADD A, #25H

SUBB A, 51H

MOV 52H, A

MOV A, #03H

MOV B, #06H

MUL AB

MOV 53H, A

MOV A, 52H

MOV B, 53H

DIV AB

MOV 33H, A

MOV 34H, B

END

OUTPUT:

Address	Value
D:33H	04H
D:34H	07H

()

ORGI DOODH

MOV A, # 12H

MOV B, # 03H

MUL AB

MOV 50H, A

MOV A, # 02 H

MOUB, # 06H

MOV MUL AB

MOV 5H, A

MOV A, # 00H

日 ADD A, # 25H

SUBB A, 51H

MOV 52 H, A

MOU A, # 03H

MOV B, # 06 H

MUL AB

MOV 53H, A

MOV A, 52H

MOV B, 53 H

DIV AB

MOV 33H, A

MOV 341 H, B

ight of the state of the state

END

Out put:

33H : 04

34H; 07

INFERENCE:		
	r the given mathematical equation is a 16-bit value and it is stored in 33	H and 34H
memory locations.	i die green mannemanen equanton is a 10 on variae and 10 is stored in 55	
RESULT:		
	Assembly language program to solve the given mathematical equation	is written
and executed successf		
and encoured successi		
[21BEC1851]	BECE204P - Microprocessors and Microcontrollers Lab	Page No: 5

LABTASK-2: SOLVING THE MATHEMATICAL EQUATION

AIM:

To write an 8051 assembly language program to solve the given mathematical equation, $(a-b)^2 = (a^2 + b^2 - 2ab)$

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV 55H, #18H

MOV 56H, #51H

MOV A, 55H

MOV B, 55H

MUL AB

MOV 20H, A

MOV 21H, B

MOV A, 56H

MOV B, 56H

MUL AB

MOV 22H, A

MOV 23H, B

MOV A, #02H

MOV B, 55H

MUL AB

MOV B, 56H

MUL AB

MOV 24H, A

MOV 25H, B

MOV A, 20H

MOV B, 22H

ADD A, B

MOV B, 24H

 $SUBB\;A,\,B$

MOV 58H, A

MOV A, 21H

MOV B, 23H

ADDC A, B

MOV B, 25H

SUBB A, B

MOV 57H, A

END

OUTPUT:

Address	Value
D:57H	0СН
D:58H	B1H

2

```
output:
ORG
    0000 H
                       Address : Value
MOV
     55H ,#18H
                        57H : OCH
    56H, #51H
                         58H : BIH :
MOV
MOV
     A , 55H
     B, 55 H
MOV
MUL AB
      20A, A
 MOV
      21H, B
 MOV
      A, 56H
 MOV
 MOV B, 56H
       AB
 MUL
      22H, A
  NOV
       23H, B
  Mov
      A, # 02 H
  Mov
       B, 55 H
  MOV
       AB
  MUL
        B, 56 H
   MOV
        AB
   MUL
   MOV
         24H, A
        25H, B
   MOV
         A, 20 H
   MOV
         B, 22 H
   Mov
         A, B
   ADD
         B, 24H
   MOV
         A,B
   SUBB
         58H, A
   MOV
        A,21H
    MOV
        B, 23H
   MOV
   ADDC A, B
    MOV B, 25 H
    SUBB A,B
         STH, A
    Mov
   END
```

<u>INFERENCE:</u>		
	the given mathematical equation is a 16-bit value and it is stored in	57H and 58H
memory locations.	the given mathematical equation is a 10-oit value and it is stored in	3/11 and 3011
RESULT:		:: :
	Assembly language program to solve the given mathematical equation	on is written
and executed successful	ııy.	
[21BEC1851]	BECE204P - Microprocessors and Microcontrollers Lab	Page No: 9



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

LAB – 07: ASSEMBLY PROGRAMMING WITH BIT-ORIENTED AND PROGRAM CONTROL INSTRUCTIONS OF 8051

CHALLENGING TASK-1: SUM OF FIRST 10 NATURAL NUMBERS

AIM:

To write an 8051 assembly language program to find sum of first 10 natural numbers.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV R2, #0AH

MOV A, #00H

MOV R0, #00H

AGAIN: INC R0

ADD A, R0

DJNZ R2, AGAIN

MOV 46H, A

END

OUTPUT:

Address	Value
D:46H	37H

ORG 0000 H MOV RZ, #OAH MOV A, # OOH MOV RO, # OOH AGIAIN : INC RO ADD A, RO DJNZ R2 , A GAIN MOV 46H, A END OUTPUT:

T.608 BEC1851

8

46H: 37

INFERENCE:

The sum of the first 10 natural numbers is stored in the memory location 46H.

RESULT:

Thus the 8051 Assembly language program to find the sum of first 10 natural numbers is written and executed successfully.

CHALLENGING TASK- 2: COMPARISON OF TWO 8- BIT NUMBERS

AIM:

To write an 8051 assembly language program to compare two 8- bit number in external memory location 8000H and 8001H respectively and reflect your result as,

- If NUM1 < NUM2, Set LSB of data RAM location 2FH.
- If NUM1 > NUM2, Set MSB of data RAM location 2FH.
- If NUM1 = NUM2, the CLR both LSB and MSB of 2FH.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV DPTR, #8000H

MOVX A, @DPTR

MOV R0, A

INC DPTR

MOVX A, @DPTR

MOV R1, A

MOV A, R0

SUBB A, R1

JZ EQUAL

JNC BIG

SETB 78H

SJMP END1

BIG: SETB 7FH

SJMP END1

EQUAL: CLR 78H

CLR 7FH

END1: NOP

END

OUTPUT 1: (NUM1 > NUM2)

Address	Value
D:2FH	80H

OUTPUT 2: (NUM1 = NUM2)

Address	Value
D:2FH	00Н

ORG DOODH

MOV DPTR, # 8000 H MOVX A, @ DPTR MOV RO, A INC DPTR MOV A, @ DPTR MOV RI, A MOV A, RO JZ EQUAL 1 SETB 78H SIMP ENDI. BIG: SETB 7FH SIMP ENDY EquAL: CLR 784 CLR THH END ENDT NOP-

INFERENCE:

If the NUM1 > NUM2, the output is 80H, if NUM1 < NUM2 the output is 01H and if the NUM1 =		
NUM2 the output is 00H.		
RESULT:		
Thus the 8051 Assembly language program to compare the numbers is written and executed		
successfully.		

LABTASK-1: STORING DATA IN EXTERNAL RAM

AIM:

To write an 8051 assembly language program to move a block of five data starting from RAM 40H to external RAM 3000H onwards and perform complement operation before storing to RAM.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV R0, #40H

MOV DPTR, #3000H

MOV R2, #05H

LOOP: MOV A, @R0

CPL A

MOVX @DPTR, A

INC R0

INC DPTR

DJNZ R2, LOOP

END

OUTPUT:

Address	Value
X:3000H	FE
X:3001H	FD
X:3002H	FC
X:3003H	FB
X:3004H	FA

Task - 1 Lab ORG H 0000 AO, # 40H MOV DPTR , # 3000 H MOV Mov R2, # 05H A, @RO LOOP: MOV CPL A @ DPTR, A MOV X INC RO INC DPTR DINZ R2, LOOP

INFERENCE:

The data is stored external memory 3000H to 3004H after complementing.

RESULT:

Thus the 8051 Assembly language program to complement and store in external RAM is written and executed successfully.

LABTASK-2: SETTING THE PORT

AIM:

To write an 8051 assembly language program to set the port as high if 24H is high and set the port low if 24H is low.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV P1, #00H

AGAIN: MOV C, 24H

JNC NO

SETB P1.0

SJMP AGAIN

NO: CLR P1.0

SJMP AGAIN

END

OUTPUT:

Address	Value
D:24H	00H

Lab Task-2

ORG DODOH

MOV PI, # 00H

AGAIN: MOV C, 24H

JNC NO

SETB PI.D

SJMP AGAIN

NO : CLR PI.O

SIMP AGAIN

END

INFERENCE:

Thus, the port is set as high if 24H is high and set the port low if 24H is low.

RESULT:

Thus the 8051 Assembly language program to set port low and high is written and executed successfully.

LABTASK-3: COMPARE THE VALUES AND STORE IN REGISTER

AIM:

To write an 8051 assembly language program to compare the temperature values and store it in R2 or R3 register according to the following conditions.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

MOV A, 55H

CJNE A, #30H, OVER

SJMP EXIT

OVER: JNC NEXT

MOV R1, A

SJMP EXIT

NEXT: MOV R2, A

EXIT: NOP

END

OUTPUT 1: (T < 30H)

Register	Value
R1	25H
A	25H

OUTPUT 2: (T > 30)

Register	Value
R2	35H
A	25H

ORG ODDOH

MOV A, 55H

CINE A, #30H, OVER

SJMP EXIT

OVER: JNC NEXT

MOV RI, A

SJMP EXIT

NEXT: MOV R2, A

EXIT: NOP

OR END

INFERENCE:

Thus, the temperature value is compared and if T = 30H, A = 30H, if T < 30H, R1 = T and if T > 30H then R2 = T.

RESULT:

Thus the 8051 Assembly language program to compare the temperature value and store in register is written and executed successfully.



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By

21BEC1851 - Rahul Karthik S

Submitted To

Dr. Prakash V

<u>LAB – 08: I/O PORT PROGRAMMING IN 8051</u>

CHALLENGING TASK-1:

AIM:

To write an 8051 assembly language program to toggle the status of the LEDs connected at port 1 pins 0 for every 1851 ms.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

CLR P1.0

BACK: SETB P1.0

ACALL DELAY

CLR P1.0

SJMP BACK

DELAY: MOV R1, #13

LOOP1: MOV R2, #255

LOOP2: MOV R3, #255

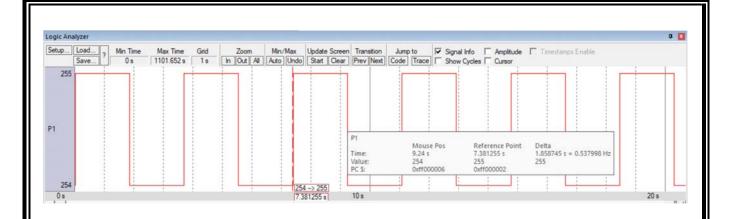
LOOP3: DJNZ R3, LOOP3

DJNZ R2, LOOP2

DJNZ R1, LOOP1

RET

OUTPUT:



208H 851

OUTPUT VERIFICATION:

Challenging Task - 1

ORG DOOOH

CLR PLO

BACK : SETB PI.O

ACALL DELAY

CLR PI.O

ACALL DELAY

SIMP BACK

DELAY : MOU RI, # 13

LOOP1 : MOV R2, # 255

LOOP2 : MOV R3 # 255

LOOP 3: POINT R3, LOOPS NOP

NOP

LOOPS: QTNZ R3, LOOPS

DINZ RZ, LOOP2

DJNZ RI, LOOPI

(85) I RET

Challenging Task-1

Crystal Freq = 11.0592

Time Period =
$$\frac{1}{0.9216} = 1.085 \mu s$$

No. of cycles to create delay =
$$\frac{1851000}{2.17}$$

$$R_1 = \frac{852996}{255} = 3345.1 \approx 3346$$

$$R_2 = \frac{3346}{255} = 13 \cdot 12 \approx \frac{13}{255} = R_3$$

In loop 3: 255 x 553.35 + 255 (1+2) x 1085
In loop 2: 255 x 553.35 + 255 (1+2) x 1085
=
$$\frac{13}{11}$$
 = $\frac{1}{11}$ = $\frac{1$

$$= 13 \times (1+2) \times 1.085$$

$$= 141934.213$$
In wop 1: 13 x 141935 + 13 x $1(+2)$ x 1.085
+ 3×1.085

INFERENCE:		
The 8051 Assembly language program is written to generate a time delay of 1.8 seconds.		
RESULT:		
Thus the 8051 Assembly language program to toggle the status of the LEDs connected at port 1 pins 0 for		
every 1851 ms is written and executed		

LAB TASK-1:

AIM:

To write an 8051 assembly language program to toggle the status of the LEDs connected at port 1 pins 0 for every 1 s.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

CLR P1.0

BACK: SETB P1.0

ACALL DELAY

CLR P1.0

SJMP BACK

DELAY: MOV R1, #5

LOOP1: MOV R2, #200

LOOP2: MOV R3, #250

LOOP3: NOP

NOP

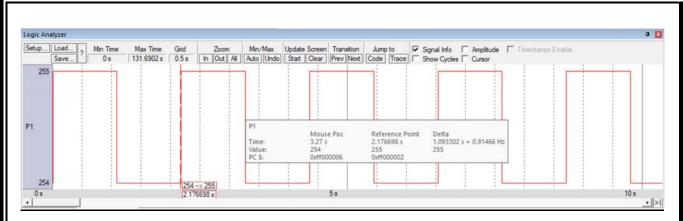
DJNZ R3, LOOP3

DJNZ R2, LOOP2

DJNZ R1, LOOP1

RET

OUTPUT:



OUTPUT VERIFICATION:

```
3.0
    Task-1:
Lab
                              ORG
   0000 H
                               CLR PI.O
                        at a transfer of the
BACK : SETB P. 10
ACALL DELAY
CLR PID
ACALL DELAY
                              that they are a
SIMP BACK
                              - 1 to 1115
DELAY
DELAY : MOV RI, #5
LOOPI : MOV R2, # 200
                        ; IMC
                         IMC
LOOP2 : MOV R3, # 250
                        ; I MC
L0093 : NOP
                     ; I MC
        NOP
       DINZ R3, LOOP3 ) 2 MC
        DINZ RZ, LOOPZ
                        ; 2 MC
        DINZ RI, LOOP!
                     ; 2 MC
                       ; 2 MC
        RET
```

INFERENCE:

The 8051 Assembly language program is written to generate a time delay of 1 seconds.

Thus the 8051 Assembly language program to toggle the status of the LEDs conn	ected at port 1 pins 0 for
every 1 s is written and executed	

LAB TASK 2:

AIM:

To write an 8051 assembly language program to continuously get the status of the switch and send it to the LED.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

SETB P1.0

CLR P1.1

HERE: MOV C, P1.0

JC LEDON

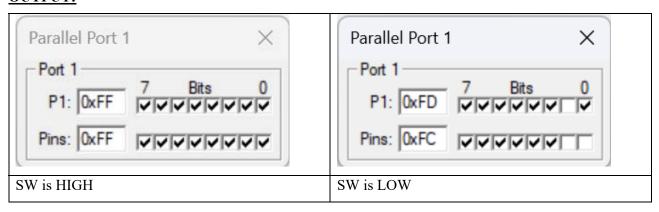
CLR P1.1

SJMP HERE

LEDON: SETB P1.1

SJMP HERE

OUTPUT:



SETB PI.D

CLR PI.I

HERE: MOV C, PI.O

TC LEDON

CLR PI.I

SIMP HERE

LEDON: SETB PI.I

SIMP HERE

LEDON: SETB PI.I

SIMP HERE

INFERENCE:

The 8051 Assembly language program is written to get the switch and send it to LED.

RESULT:

Thus the 8051 Assembly language program to continuously get the status of the switch and send it to the LED.



School of Electronics Engineering (SENSE)

BECE204P – MICROPROCESSORS AND MICROCONTROLLERS LAB RECORD

Submitted By
21BEC1851 – Rahul Karthik S

Submitted To

Dr. Prakash V

<u>LAB – 11: INTERUPT PROGRAMMING & LCD</u> <u>INTERFACING WITH 8051</u>

LABTASK-1:

AIM:

To write a program using interrupts that continuously get 8-bit date from P0 and sends it to P1 while simultaneously creating a square wave of 200 μ s period on P2.1. Use timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

LJMP MAIN

ORG 0080H

CPL P2.1

RETI

ORG 0030H

MAIN: MOV P0, #0FFH

MOV P1, #00H

CLR P2.1

MOV P2.1

MOV TMOD. #02H

MOV TH0, #0AH

MOV IE, #82H

SETB TR0

BACK: MOV A, P0

MOV P1, A

SJMP BACK

END

OUTPUT VERIFICATION:

```
Lab Task-1:
  ORG
        0000 H
  LJMP
        MAIN
  ORG
       000 BH
       P2,1
  CPL
                                    TANTON LINES
  RETI
                                         Strain type
  MAIN : ORG DOSOH
                                       100 to 100 a ...
   MOV PO, # OFFH
   MOV PI, # 00H
               MINORNE DE NO COM : $1 HO, DAYS
   P CLR PZ.1
   MOV TMOD, # 02H
   MOV THO, # 6A4H
                               2790 + 1 W ( A 2 ) 6 4
   MOV 1E, # 82 H
    SETS TRO
                                      todal .. var
    BACK : MOV
    MOV PI, A
    ST MP BACK
    END
```

INFERENCE:

From this we can know that, P0 is an input port whereas P1 and P2 are input port.

RESULT:

Thus the 8051 Assembly language program to use interrupts that continuously get 8-bit date from P0 and sends it to P1 while simultaneously creating a square wave of 200 μs period on P2.1 is written and executed successfully.

LABTASK-2:

AIM:

To turn on LED when external interrupt is low, turn on the LED and to perform toggle operation in P1.5 with the delay of 500 ms.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0000H

LJMP MAIN

// ISR for INT1

ORG 0013H

SETB P1.3

RETI

ORG 0030H

MAIN: SETB P3.3

CLR P1.3

CLR P1.5

MOV IE, #10000100B

HERE: CLR P1.3

SETB P1.5

ACALL DELAY

CLR P1.5

ACALL DELAY

SJMP HERE

// DELAY OF 500 ms

DELAY: MOV R2, #04H

HERE3: MOV R1, #0FFH

HERE2: MOV R0, #0FFH

HERE1: DJNZ R0, HERE1

DJNZ R1, HERE1

DJNZ R2, HERE2

RET

END

OUTPUT VERIFICATION:

INFERENCE:

The interrupt is given externally from the ESA 8051 kit, and LED is turning on and off based on the interrupt.

RESULT:

Thus the 8051 Assembly language program to turn on LED when external interrupt is low, turn on the LED and to perform toggle operation in P1.5 with the delay of 500 ms is written and executed successfully.

LABTASK-3:

AIM:

To write an 8051 assembly language program to display the message "21BEC1851", on LCD display using DPTR.

PROCEDURE:

- 1. Create a Project (.uvproj)
- 2. Create and write an assembly program (.a51)
- 3. Build your project to check errors.
- 4. Select debug mode.
- 5. Run your code to verify the output.

PROGRAM:

ORG 0030H

MOV DPTR, #MYCOM

C1: CLR A

MOVC A, @A+DPTR

ACALL COMNWRT

ACALL DELAY

INC DPTR

JZ SEND_DAT

SJMP C1

SEND_DAT: MOV DPTR, #MYDATA

D1: CLR A

MOVC A,@A+DPTR

ACALL DATAWRT

ACALL DELAY

INC DPTR

JZ AGAIN

SJMP D1

AGAIN: SJMP AGAIN

COMNWRT: MOV P2, A

CLR P3.7

CLR P3.6

SETB P3.5

ACALL DELAY

CLR P3.5

SETB P3.5

ACALL DELAY

CLR P3.5

RET

DATAWRT: MOV P2, A

SETB P3.7

CLR P3.6

SETB P3.5

ACALL DELAY

CLR P3.5

RET

DELAY: MOV R3, #250

HERE2: MOV R4, #255

HERE: DJNZ R4, HERE

DJNZ R3, HERE2

RET

ORG 0300H

MYCOM: DB 38, 0EH, 01, 06, 84H, 0

MYDATA: DB "21BEC1851", 0

END

```
Lab Task-3:
  ORG 0000 H
  LIMP MAIN
  ORG 0030H
  MAIN: MOU DOTE, # MYCOM
   MOVC . A, QA + DPTR
   ACALL COMNWRT
    INC' DPTR
    JZ SEND-DAT
    SIMP CI
    SEND_DATE: MOU DATE, #MYDATA
 DI:CLR A
    MOVEEA, QA+ DATE
    ACALL DATAWRE
    ACALL DELAY
    INC DATE
    JZ AGAIN
    STMP ADI
AGAIN: STMP AGAIN
COMN WET : MOV P2, A
       CLR P3.7
        CLR P3.6
        SETB P3.5
        ACALL DELAY
        CLR P3.5
         RET
DATA WET : MOU PZ, A
         SETB P3.7
         CLR 93.6
         SETB P3.5
         ACALL DELAY
         ELR PS.S
DELAY: MOU R3 , # 25$
```

HERE2 : MOV R4, # 255

HERE : DINZ R4, THERE

DINZ R3, HERE2

RET

01.708

OR6 - 0300 H

MYCOM: DB 38H, OEH, 01, 06, 844,0

MY DATA : DB '21BEC1851', 0

END

INFERENCE:

Thus, the register number, "21BEC1851" is printed one by one in LCD display.

RESULT:

Thus, an 8051 assembly language program to display the message "21BEC1851", on LCD display using DPTR is written and executed successfully.