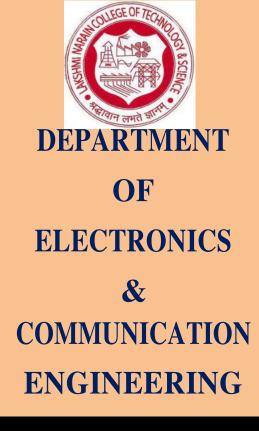
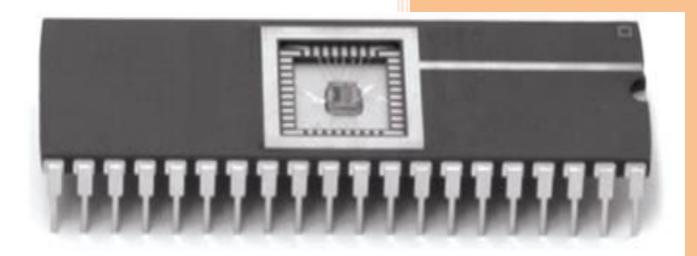


Name of Student: _	
Enrollment No.: _	
Class: _	
Section:	
Session: _	



Microprocessor & Its Applications [EC-501] Lab Manual



Prepared By: Dr. Aparna Gupta

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE

Kalchuri Nagar, Raisen Road Bhopal (MP) 462023



Vision and Mission of the Department

Vision

To be world-wide recognized for adopting and keeping innovation and entrepreneurship mindset as abreast of learning to produce professionals as valuable, ethical and moral resource for industry and society.

Mission

- To establish an ecosystem where students could grow with innovative practices followed in communication engineering.
- To adopt the global approaches to transform the young aspirant into most acceptable engineering professional catering the modern challenges of the society in the most ethical with full of patriotic zeal.
- To facilitate and felicitate all the learners to have close interactions with the industry experts and researchers for keeping them updated of the current and future needs of the society.
- To develop the mindset of being innovative and entrepreneurial in the future engineering professionals.



ELETRONICS AND COMMUNICATION ENGINEERING

PSO

- 1. To analyze, design and develop solutions of real time problems and industry needs.
- 2. Ability of effectively communicating with the professionals and preparation of reports, documents and presentation while working in teams.
- 3. Knowledge and understanding of latest developments in the field of VLSI, Embedded system, Networking, Matlab and other major tools necessary for keeping pace with the industry.
- 4. Ability of solving complex engineering problems with ethical and law full approach to prevent the society and environment from adverse impacts.

DEPARTMENT OF ELECTRONIC AND COMMUNICATION ENGINEERING PEOs

- 1. The graduate will have the knowledge and skills of analog and digital communication in providing necessary solutions to the real world problems.
- 2. The graduate will be able to design, develop, analyze and implement the modern tools and systems involving principles of electronics and telecommunication engineering.
- 3. The graduate will be following the ethical practices of the core industry and supporting software industry in providing most acceptable solution to the society.
- 4. The graduate will have the innovative mindset of learning and implementing the latest technological advancements and research outcomes in the electronic hardware and software to keep pace with the rapid developments in socio economic world.



GROUP OF COLLEGES TOWARDS BEING THE BEST" LAKSHMI NARAIN COLLEGE OF TECHNOLOGY AND SCIENCE, BHOPAL



INDEX

 Name of Student:
 Enrollment No.:

S.No.	. Title of the Experiment	Date of Experiment	Date of Experiment	Remark
1	Installation and Study of 8086 assemblers/ compilers			
2	Perform addition and subtraction of 8- bit numbers.			
3	Perform addition, subtraction, multiplication and division of 16-bit numbers			
4	Perform Logical Operation using 8086			
5	Perform Universal Gate Operation using 8086			
6	Find 1's and 2's complements			
7	Find the average of numbers			
8	Exchanging the data of registers			
9	Finding factorial of a number			
10	Stepper Motor Rotation			



Code of Conducts for the Laboratory

- > All bags must be left at the indicated place.
- > The lab timetable must be strictly followed.
- > Be **PUNCTUAL** for your laboratory session.
- > Noise must be kept to a minimum.
- ➢ Workspace must be kept clean and tidy at all time.
- ➤ Handle the experiment kit and interfacing kits with care.
- All students are liable for any damage to the accessories due to their own negligence.
- Students are strictly **PROHIBITED** from taking out any items from the laboratory.
- Students are NOT allowed to work alone in the laboratory without the Lab Supervisor
- Report immediately to the Lab Supervisor if any malfunction of the accessories, is there.
- Before leaving the lab Place the stools properly.
- Please check the laboratory notice board regularly for updates.



Experiment -1 Installation and Study of 8086 assemblers/compilers

Aim: Installation and study of 8086 assemblers/compilers.

(a) Debug

(b) Emu86

(c) NASM

Theory:

To understand the architecture of 8086 microprocessor.

Enable to write small assembly language programs using DEBUG utility program.

To understand the design flows of NASM and Emu8086.

Introduction:

Assembly language programs resemble a compiled language programs, like PASCAL. In compiled languages the user first creates a source file, which is a text file of entire program. The compiler changes it into machine language instructions. In the compiled language program the entire program is transformed into machine language at once. Assembly Language is converted into executable machine code by a utility program referred to as an assembler, the conversion process is referred to as assembly, or assembling the code.

Microprocessor:

A silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably. A Microprocessor (sometimes abbreviated μ P) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles. A **microprocessor** incorporates the functions of a computer's CPU on a single integrated circuit, or at its most a few integrated circuits. All modern CPUs are microprocessors making the micro-prefix redundant.

Central Processing Unit(CPU) on a single silicon chip (called microchip) that can be 'Soft Wired' by using different programming instructions. When coupled with memory and input-output devices, a microprocessor becomes a microcomputer. INTEL 4004, one of the most earliest microprocessors, had a 4-bit data bus and could address 4.5 kilobytes of memory. The first mass produced Microprocessor was Intel 8080 than in 1973 cost about \$400.

Three basic characters differentiate microprocessors: -

- \rightarrow INSTRUCTION SET: The set of the instructions that microprocessor can execute.
- \rightarrow BANDWIDTH: The number of bits processed in a single instruction.
- → CLOCK SPEED: Given in Megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.



In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).



Two basic characteristics differentiate microprocessors:

Instruction set: The set of instructions that the microprocessor can execute.

Bus width: The number of bits processed in a single instruction.

8086 Architecture:

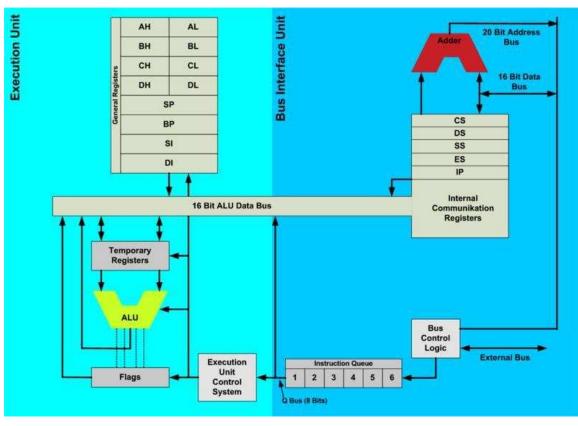


Figure 1. 8086 Architecture

• 8086 is a 16 bit µp.



- 8086 has a 20 bit address bus can access upto 2²⁰ memory locations (1 MB).
- It can support up to 64K I/O ports.
- It provides 14, 16-bit registers.
- It has multiplexed address and data bus AD0-AD15 and A16–A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- 8086 is designed to operate in two modes, Minimum and Maximum.
- It can prefetches up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.
- It requires +5V power supply.
- A 40-pin dual in line package.

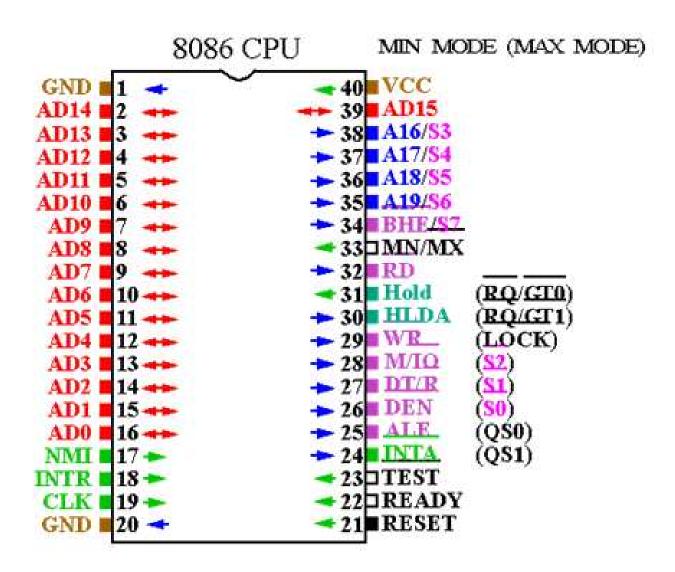


Figure 2. 8086 Pin Diagram





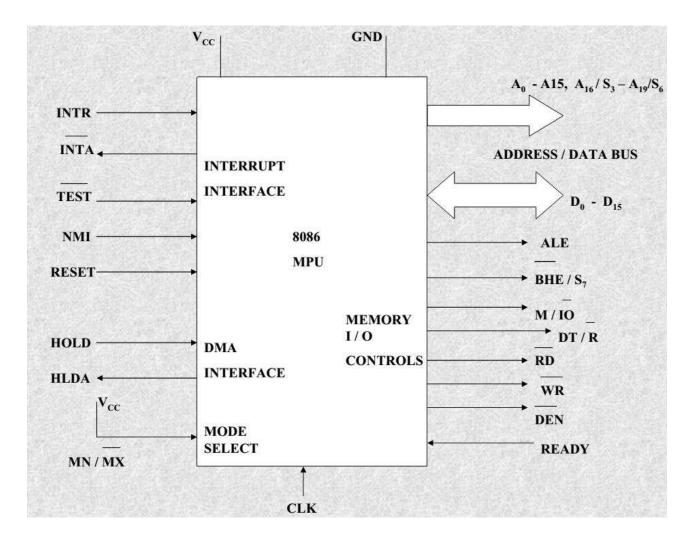


Figure 3. 8086 Signal Diagram





Debug for 8086:

C:\WINDOWS\system32\debug.exe	- 🗆 ×
-a 8AE7:8100	

Figure4. Debug- DOS based emulator for Windows

A debugger displays the contents of memory and lets us view the registers and variables as they change. We can step through a program one line at a time called tracing, making it easier to find logic errors.

Debugger Functions

Some of the most rudimentary functions that a debugger canperform are the following:

- Assemble short programs
- View a program's source code, along with its machine code
- View the CPU registers and flags
- Trace or execute a program, watching variables for changes
- Enter new values into memory
- Search for binary or ASCII values in memory
- Move a block of memory from one location to another
- Fill a block of memory
- Load and write disk files

Debugger Commands

- A Assemble a program using instruction mnemonics
- R Display the contents of registers and flags
- T Trace a single instruction
- U Disassemble memory into assembler mnemonics
- D Dump (display) the contents of memory
- E Enter bytes into memory
- Q Quit DEBUG and return to DOS
- L Load data from disk
- W Write data from memory to disk
- N Create a filename for use by the L and W commands

EMU8086:

The integrated 8086 assembler can generate console programs that can be executed on any computer that runs x86 machine code (Intel/AMD architecture)

The architecture of the 8086 Intel microprocessor is called **"Von Neumann architecture"** after the mathematician who conceived of the design.



A CPU can interpret the contents of memory as either instructions or data; there's no difference in the individual bytes of memory, only the way in which they're arranged. Because of this, it's even possible for programs to re-write their own instructions, then execute the instructions they've changed.

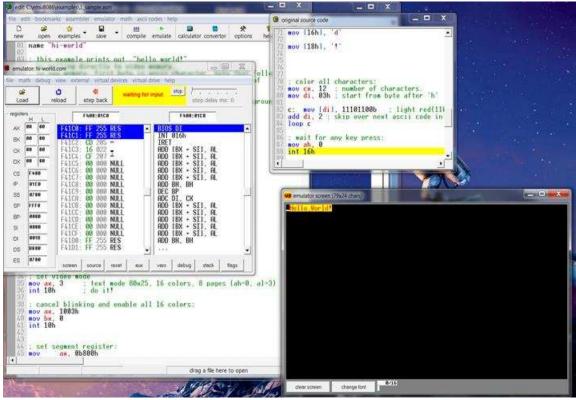


Figure 5.emu 8086 compiled output



NASM: The Netwide Assembler

The Netwide Assembler, NASM, is an 80x86 and x86–64 assembler designed for portability and modularity. It supports a range of object file formats, including Linux and *BSD a.out, ELF, COFF, Mach–O, Microsoft 16–bit OBJ, Win32and Win64. It will also output plain binary files. Its syntax is designed to be simple and easy to understand, similar to Intel's but less complex. It supports all currently known x86 architectural extensions, and has strong support for macros.

To assemble a file, you issue a command of the form **nasm -f <format><filename> [-o <output>]** For example, **nasm -f elf myfile.asm** will assemble myfile.asminto an ELFobject file myfile.o. And **nasm -f bin myfile.asm -o myfile.com** will assemble myfile.asminto a raw binary file myfile.com.

To produce a listing file, with the hex codes output from NASM displayed on the left of the original sources, use the –loption to give a listing file name, for example: nasm –f coff myfile.asm –l myfile.lst

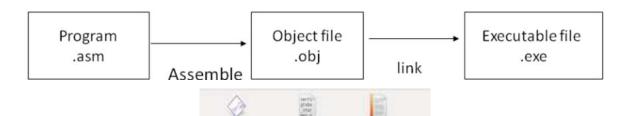


Figure 6. Flow of program development

008	ayoush@ubuntu: -/coding/asm
start:	
	nov eax, 4
	nov ebx, 1
	nov eck, string
	nov edx, length
	Int 80h
	nov eax, 1
	nov ebx, 8
	int Boh
section	.data
	string: db 'Hello Ayoush', ðah
	length:equ 13
	ubuntu:-/coding/asm5 nasm -f elf -o hello.o hello.asm ubuntu:-/coding/asm5 ls -l hello.o
	r 1 ayoush ayoush 672 Jun 22 13:18 hello.o
	ubuntu:-/coding/asm5 ld hello.o -o hello
	ning: cannot find entry symbol _start; defaulting to 8080808088848058
	ubuntu:-/coding/asmS ./hello
Hello A	
	ubuntu:-/coding/asm5

Figure 7. Snapshot of NASM Assembler



Summary:

The native language is machine language (using 0,1 to represent the operation). A single machine instruction can take up one or more bytes of code. Assembly language is used to write the program using alphanumeric symbols (or mnemonic), eg. ADD, MOV, PUSH etc. The program will then be assembled (similar to compiled) and linked into an executable program. The executable program could be .com, .exe, or .bin files. An **assembly language** is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions.

<u>Result:</u> We have studied the installation & assembling of 8086 compilers.



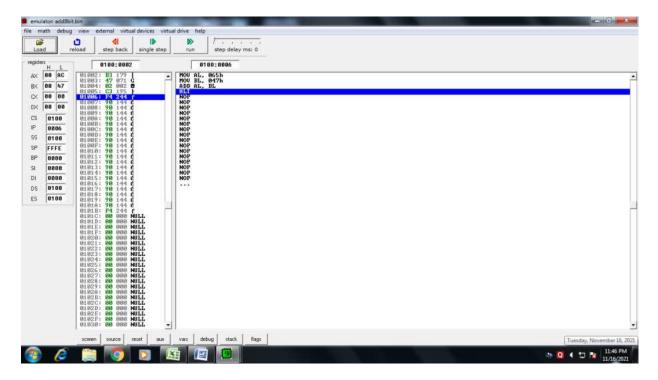
Experiment -2 Perform addition and subtraction of 8-bit numbers.

Aim: Write a program in 8086 to perform addition and subtraction of 8-bit numbers.

Software Used: EMU8086

Programs:

a) Addition: MOV Al, 65H; MOV Bl, 47H; ADD Al, Bl; HLT;



b) Subtraction:

MOV Al, 45H; MOV Bl, 43H; SUB Al, Bl; HLT;





ee, bea	reload step back single step	nun step delay ms: 0	
isters H L	0100:0006	0100:0006	
× 00 02	01000: B0 176 B	100 AL, 845h 100 BL, 843h	
× 88 43	01002: B3 179 S	UB AL, BL	
× 08 80	81884: 28 842 W	0P 0P	
× 00 00	81886: F4 244 F	IOP IOP	
S 0100	81888: 98 144 £	KOP NOP	
8886	81886: 98 144 €		
5 0100	0188C: 98 144 £	10P	
FFFE	0180E: 90 144 C		
P 0000	81018: 98 144 C	IOP	
8888	01011: 90 144 É 01012: 90 144 É	iop Iop Iop	
8888	81814: 90 144 E	10P	
5 0100 S 0100	B1B16: 98 144 É	60P 60P 60P	
	1111111: 198:144:6 144:6 11111: 198:144:6 144:6 11111: 198:144:6 144:6 11111: 198:144:6 144:6 11111: 198:144:6 144:6 11111: 198:144:6 11111.6 11111: 198:144:6 11111.6 11111: 198:144:6 11111.6 11111: 198:144:6 11111.6 11111: 198:146:1 11111.6 11111: 198:146:1 11111.6 11111: 198:146:1 11111.6	LT DD [EK + S1], AL DD [EK + S1], AL	

<u>Result:</u> We have performed addition and subtraction of 8-bit numbers.



Experiment - 3 Perform addition, subtraction, multiplication and division of 16-bit numbers.

Aim: Write a program in 8086 to perform:

- a) addition of two 16-bit numbers
- b) subtraction of two 16-bit numbers
- c) multiplication of two 16-bit numbers
- d) division of two 16-bit numbers

Software Used: Emu8086

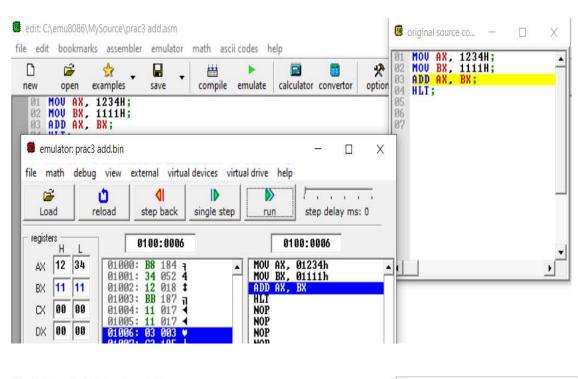
Programs:

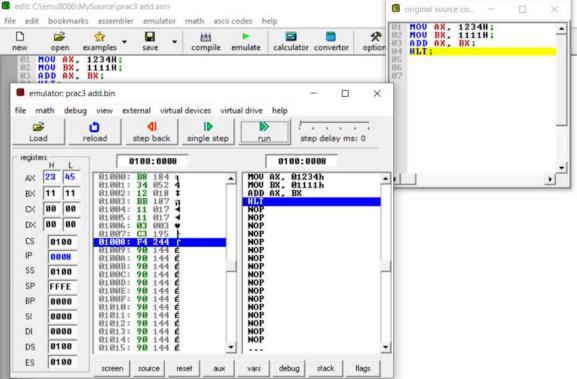
a) Addition:

MOV AX, 1234H; MOV BX, 1111H; ADD AX, BX; HLT;







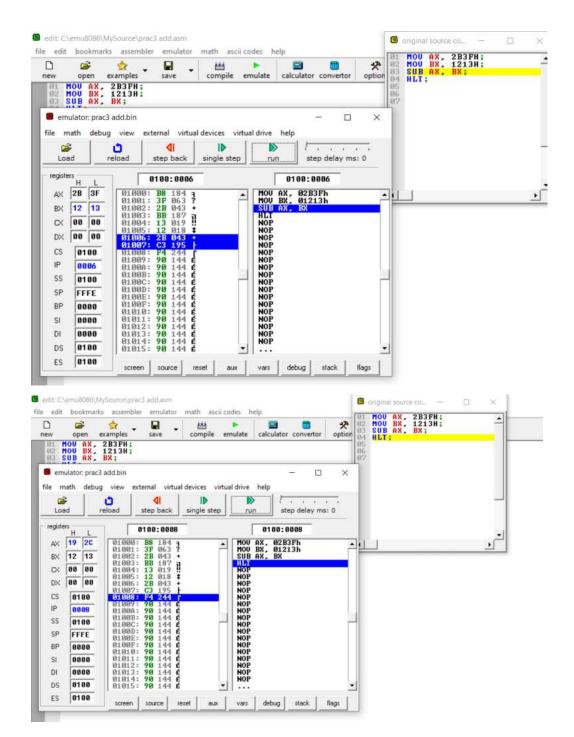


b) Subtraction:

MOV AX, 2B3FH; MOV BX, 1213H; SUB AX, BX; HLT;

GROUP OF COLLEGES VORKING TOWARDS BEING THE BEST" LAKSHMI NARAIN COLLEGE OF TECHNOLOGY AND SCIENCE, BHOPAL



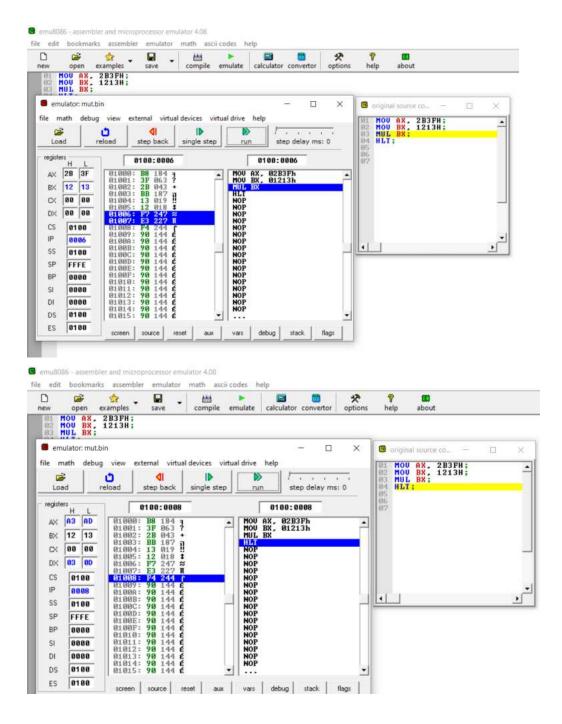


c) Multiplication:

MOV AX, 2B3FH; MOV BX, 1213H; MUL BX; HLT;







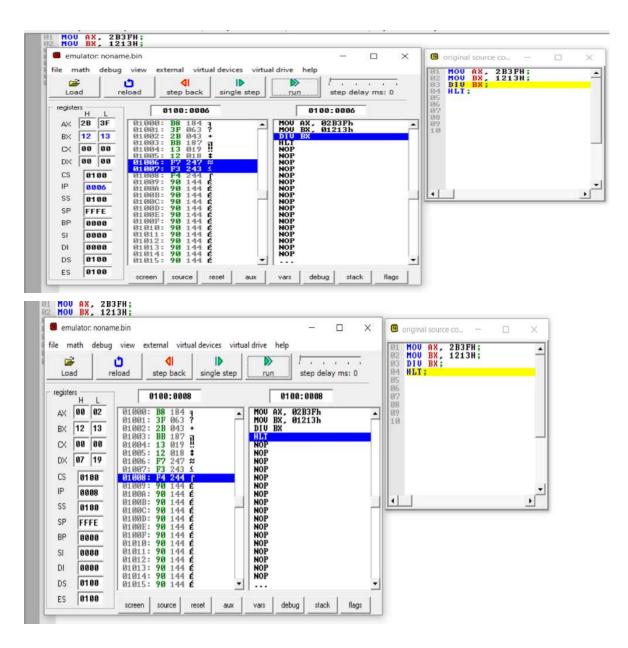
d) Division:

MOV AX, 2B3FH; MOV BX, 1213H; DIV BX; HLT;



GROUP OF COLLEGES





Result: We have performed addition, subtraction, multiplication and division of 16-bit numbers





Experiment - 4 Perform Logical Operation using 8086

Aim: Write a program in 8086 to perform various logical operation.

Software Used: EMU8086

Theory:

a) Not Operation

- 1. Start debug utility program on windows xp or Linux.
- 2. Take operand-1 into register AX.
- 3. Use logical operation instruction NOT for Not operation.
- 4. Store the result into register AX.
- **5.** Program halted

b) AND Operation

- 1. Start debug utility program on windows xp or Linux.
- 2. Take operand-1 into register AX.
- 3. Take operand-2 into register BX.
- 4. Use logical operation instruction AND for AND operation.
- 5. Store the result into register AX.
- 6. Program halted

c) OR Operation

- 1. Start debug utility program on windows xp or Linux.
- 2. Take operand-1 into register AX.
- 3. Take operand-2 into register BX.
- 4. Use logical operation instruction OR for OR operation.
- 5. Store the result into register AX.
- 6. Program halted\

d) XOR Operation

- 1. Start debug utility program on windows xp or Linux.
- 2. Take operand-1 into register AX.
- 3. Take operand-2 into register BX.
- 4. Use logical operation instruction XOR for XOR operation.
- 5. Store the result into register AX.
- 6. Program halted

Programs:

i) NOT operation:



MOV AX, 1234H; NOT AX; HLT;

Loa		r	d eload	step	di back	single s	tep	ru	n ste	, , , p delay m	; ; s: 0
giste	ers — H	L		0100	:0005]	0100:0	005	
×	ED	CB	0100		184 =		-	MOU		}4h	
3X	00	00	0100	: 34	052 4 018 3	1		NOT HLT	AX		
x	00	00	0100	: F7		5		NOP NOP			
	-		0100	F4	244	r		NOP			
XC	00	00	01000	: 90	144			NOP			
CS	01	00	0100	: 90	144 E			NOP			
P	_	20000 L	0100	: 90	144			NOP			
	00	05	0100	1: 90 1: 90	144 E			NOP			
ŝS	01	00	0100		144			NOP			
SΡ	FF	FE	0100		144			NOP			
3P	-		0100		144 E			NOP NOP			
P	00	00	01010	: 90	144			NOP			
51	00	00	0101	: 90	144			NOP NOP			
Я	00	00	0101		144			NOP			
e e	100		0101		144			NOP			

ii) AND operation: MOV AX, 1567H;



MOV BX, 0A893H; AND AX, BX; HLT;

emulator: noname.bin_

– 🗆 X

file math debug view external virtual devices virtual drive help

Lo		r	oload ste	dl p back	single step	run		step delay	, , ms: 0
registe	ers — H	L	010	0:0008	-	Г	01	00:0008	
AX	00	03	01000: B				AX,	01567h	
ВX	A8	93	01001: 6 01002: 1	5 021 §		AND	BX, AX,	ØA893h BX	
CX	00	00	01003: B 01004: 9	3 147 8		HLT NOP			
DX	00	00	01005: A 01006: 2	3 035 #		NOP			
CS	01	00	01007: C			NOP			
IP	00	08	01009: 9 0100A: 9	0 144 É 0 144 É		NOP			
SS	01	00	0100B: 9 0100C: 9			NOP			
SP	FF	FE	0100D: 9 0100E: 9	0 144 É		NOP			
ΒP	00	00	0100F: 9	0 144 É		NOP			
SI	00	00	01010: 9	0 144 É		NOP			
DI	00	00	01012: 9	0 144 É		NOP			
DS	01	00	01014: 9	0 144 É 0 144 É	*	NOP			
ES	01	00	1	1	eset aux	vars	de	bug stack	flags

GROUP OF COLLEGES WORKING TOWARDS BEING THE BEST" LAKSHMI NARAIN COLLEGE OF TECHNOLOGY AND SCIENCE, BHOPAL



MOV BX, 0A893H; OR AX, BX;

HLT;

le m	nath debug	g view external virtual d	levices virtual drive help
Loa	513	reload step back	single step
registe	rs H L	0100:0008	0100:0008
AX	AD 73	01000: B8 184 7	MOU AX, 08573h
ВX	A8 63	01001: 73 115 s 01002: 85 133 à	MOU BX, ØA863h OR AX, BX
сx	00 00	П 01003: BB 187 П 01004: 63 099 с	HLT NOP
DX	00 00	01005: A8 168 ¿ 01006: 0B 011 ð	NOP
		01007: C3 195	NOP
CS	0100	01008: F4 244 r	NOP
IP	0008	01009: 90 144 É 0100A: 90 144 É	NOP
SS	0100	0100B: 90 144 É	NOP
SP	FFFE	0100C: 90 144 É 0100D: 90 144 É	NOP NOP
	-	0100E: 90 144 É 0100F: 90 144 É	NOP NOP
ΒP	0000	01010: 90 144 É	NOP
SI	0000	01011: 90 144 É	NOP
DI	0000	01012: 90 144 É 01013: 90 144 É	NOP NOP
DS	0100	01014: 90 144 É 01015: 90 144 É	- NOP

GROUP OF COLLEGES UNCOUNT OF COLLEGES LAKSHMI NARAIN COLLEGE OF TECHNOLOGY AND SCIENCE, BHOPAL



🗰 em	nulato		BX, 0A893H; R AX, BX; ne.bin_	- 🗆 X
ile n		debug I	view external virtual devices virtu:	al drive help
Loa		r	toadstep backsingle step	run step delay ms: 0
registe	ers — H	L	0100:0008	0100:0008
AX	FØ	EA	01000: B8 184 7	MOU AX, 05879h
В×	A8	93	01001: 79 121 y 01002: 58 088 X	MOU BX, ØA893h XOR AX, BX
cx	00	00	01003: BB 187 01004: 93 147 0	HLT NOP
DX	00	00	01005: A8 168 ¿ 01006: 33 051 3	NOP NOP
CS	์ ดา	00	01007: C3 195 - 01008: F4 244 c	NOP
IP	-	08	01009: 90 144 É 0100A: 90 144 É	NOP NOP
SS	1	00	0100B: 90 144 É 0100C: 90 144 É	NOP
SP	-	FE	0100D: 90 144 É	NOP
BP		00	0100E: 90 144 É 0100F: 90 144 É	NOP
SI	- Andrews	00	01010: 90 144 É 01011: 90 144 É	NOP NOP
DI	-	00	01012: 90 144 É 01013: 90 144 É	NOP NOP
DS	-	00	01014: 90 144 É 01015: 90 144 É	NOP
ES	01	00	screen source reset aux	vars debug stack flags

<u>Result:</u> We have performed various logical operations in 8086.

GROUP OF COLLEGES UNCOUNT OF COLLEGES LAKSHMI NARAIN COLLEGE OF TECHNOLOGY AND SCIENCE, BHOPAL



Perform Universal Gate Operation using 8086

Aim: Write a program in 8086 to perform universal gate operation.

Software Used: EMU8086

Programs:

i) NAND operation: MOV AX, 1567H; MOV BX, 0A893H; AND AX, BX; NOT AX; HLT;

Loa	192	reload	d step back	single step	run step delay ms:
gister	rs H L		01 00 : 000A		0100:000A
AX BX CX DX CS IP SS SP SP SI DI	FD FE 12 03 00 00 00 00 0100 00 0100 000A 0100 FFFE 00000 0000 00000 0000	01000 01001 01002 01003 01004 01005 01006 01007 01008 01007 01008 01007 01008 01007 01008 01007 01008 01007 01008 010010 01010 01013 01013 01013 01013 01013 01013	61 097 a 23 035 # BB 187 a 03 003 ¥ 12 018 4 23 035 # C3 195 a F7 247 a F7 247 a F7 247 a F7 247 a F9 144 6 90 144 6		MOU AX, 02361h MOU BX, 01203h AND AX, BX NOT AX HLT NOP NOP NOP NOP NOP NOP NOP NOP NOP NOP
DS ES	0100	01015	: 90 144 É	<u> </u>	1

- ii) NOR operation:
 - MOV AX, 1567H; MOV BX, 0A893H;



OR AX, BX;
NOT AX;
HLT:

NC.	1	. 1	1/	۷,	
Ŧ	Т	۰.			

Loa		r	oloadste	⊲l p backsin		N run	1 N.V.	yms:0
giste	ers H	L	010	10:000A		01	00:000A	
4X	18	80	01000: B		- M	OV AX,	06350h	
ВХ	A5	7F	01001: 50	0 080 P 3 099 c		OV BX, R AX, I	ØA57Fh RX	
	-		01003: B	B 187 a	N	OT AX		
X	00	00	01004: 71 01005: A			LT		
DХ	00	00	01005: A			OP OP		
	·		01007: C	3 195 -	N	OP		
CS	01	00	01008 : F	7 247 ≈ 0 208 ¤		OP OP		
P	00	OA	01009 E	4 244 r		OP		
SS	04	00	0100B: 9	0 144 É	N	OP		
	01	00	0100C: 90		N	OP		
SP	FF	FE	0100D: 90 0100E: 90			OP OP		
BP	0.0	00	0100F: 9			ŎP		
	-		01010: 9			OP		
SI	00	00	01011: 90			OP OP		
DI	66	00	01013: 9			ÖP		
			01014: 9			OP		
DS	01	00	01015: 9		211 Contra			

Result: Universal Gates has been performed in 8086.

Signature of Faculty

Experiment - 6 Find 1's and 2's complements



Aim: Write a program to find one's and two's complement of a number in 8086.

Program: -

- 1. One's complement: MOV AX, 1472H;
 - NOT AX;
 - HLT;

Loa		r	toadstep back	single step	run step delay ms: 0	
registe	ers H	L	0100:0005		0100:0005	
AX	EB	8D	01000: B8 184 7		MOU AX, 01472h	7
ВX	00	00	01001: 72 114 r 01002: 14 020 ¶	ti 🛛 🚺	NOT AX	
CX	00	00	01003: F7 247 ≈ 01004: D0 208 □		NOP	
DX	00	00	01005: F4 244 f 01006: 90 144 É		NOP NOP	
CS	01	00	01007: 90 144 É 01008: 90 144 É		NOP NOP	
IP	00	05	01009: 90 144 É 0100A: 90 144 É		NOP NOP	
SS	01	00	0100B: 90 144 É 0100C: 90 144 É	_	NOP	
SP	FF	FE	0100D: 90 144 É 0100E: 90 144 É		NOP	
ΒP	00	00	0100F: 90 144 É		NOP	
SI	00	00	01011: 90 144 É		NOP NOP	
DI	00	00	01012: 90 144 É 01013: 90 144 É		NOP NOP	
DS	01	00	01014: 90 144 É	-		
ES	01	00	screen source r		vars debug stack flags	

- 2. Two's complement:
 - a) MOV AX,1523H; NOT AX;

GROUP OF COLLEGES UNCERTING TOWARDS BEING THE BEST"



ADD AX,0001H; HLT;

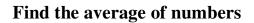
Lo			reload	⊲I step back	single step	ru		step dela	y ms: 0	
egiste	ers — H	L		0100:000	18		01	00:0008		
AX	EA	DD	01000:	B8 184		MOU	AX,	01523h		
ВX	00	00	01001:	23 035 15 021	# §	- NOT ADD		00001h		ſ
cx	00	00	01003:	F7 247 DØ 208	ж	HLT NOP		CADADADAN 1990		
	-	_	01005:	05 005	٠	NOP				
DX	00	00	01006:	01 001		NOP				
CS	01	00	01007	00 000 F4 244	NULL	NOP				
	1.000		01009:	90 144	É	NOP				
IP	196	08	0100A:	90 144	É	NOP				
SS	01	00	01008:	90 144	E F	NOP NOP				Ť
SP	lee.	FE	0100D:	90 144	Ĕ	NOP				
	FF	FE.	0100E:	90 144	é	NOP				
ΒP	00	00	0100F:	90 144	E	NOP NOP				
SI	66	00	01011:	90 144	É	NOP				
	-		01012:	90 144	é	NOP				
DI	00	00	01013:	90 144 90 144	É	NOP				
DS	01	00	DIDT.1.	10 144	0 5 3					- 38

b) MOV AX,1523H; NEG AX;



	1000	lebug vie	w extern		1		help	
کی Load		reload	sti	⊲I ep back	single step	run	step del	ay ms: 0
gisters	s H	L	01	00:0005	5	Γ	0100:0005	
жſ				8 184	1 🔺	MOU A	X, Ø1523h	
×「	00	00 00	002: 1	3 035 5 021	# §	NEG A	IX .	
хĺ	00	00 01	004: D	8 216	× ±	NOP NOP		
× [00	00 Ø1			É	NOP NOP		
s	010	01 0 01			É	NOP		
•	000	— Q1		Ø 144 Ø 144	É	NOP		
s	010	. 01	00B: 9	0 144	é _	NOP		
Р	FFF	F 01	00D: 9	0 144	é 👘	NOP		
Р	000	6 Ø	00F: 9	0 144	É	NOP NOP		
1	000	0 01	011: 9	0 144	É	NOP NOP		
	1.0.0.0	01	012: 9		É	NOP NOP		

Result: one's and two's complement of a number in 8086 has been evaluated.



Aim: Write a program to find average of 3 numbers in 8086.

Program: -

MOV AX, 154AH; MOV BX, 2312H; MOV CX, 0A152H; ADD AX, BX; ADD AX, CX; MOV DX, 0003H; DIV DX; HLT;

		nath	debug	View	external virti	ual devices virtu	al drive	help	
H L 0100.0009 0100.0009 AX 15 4A 01000.0009 MOU AX. 0154Ah BX 23 12 01001.0009 01002.0009 MOU AX. 0154Ah BX 23 12 01001.0001 BB 184 1 A CX A1 52 01001.0005 BB 187 A ADD AX. 02312h CX A1 52 01001.0005 B9 185 ADD AX. 0X ADD CX 04 00061.005 23 035 H MOU DX. 00003h DX 08 00 01005 B9 185 H H H CS 0100 010065 B9 165 H HUT NOP NOP 010082 C1 193 H NOP NOP SS 0100 010082 C1 193 NOP NOP NOP SP FFFE 010002 BA 186 NOP NOP NOP SI </th <th>1000</th> <th></th> <th></th> <th>and the second se</th> <th>and the second second</th> <th>and the second se</th> <th>1.000</th> <th>step dela</th> <th></th>	1000			and the second se	and the second	and the second se	1.000	step dela	
AX 15 4A 01000: B8 184 1 ▲ BX 23 12 01000: B8 184 1 ▲ MOU AX. 0154Ah BX 23 12 01002: SB 187 1 ▲ MOU BX. 02312h BX 23 12 01003: BB 187 1 ▲ MOU AX. 02312h CX A1 52 01004: 12 018 ± ▲ ADD AX. 03 DX 08 00 01005: 23 035 ± ■ ADD AX. 03 DX 08 00 01006: B9 185 ↓ ■ BDIU DX. 00003h DX 0100 01009: 03 003 ∨ ■ ■ HLI NOP SS 0100 01009: 03 003 ∨ ■ NOP NOP NOP SF FFFE 010001: BA 186 Ⅲ ■ NOP NOP NOP SI 0000 01010: F7 247 ≈ ■ NOP NOP NOP SI 0000 01010: F7 247 ≈ ■ NOP NOP NOP 01010: 90 01013: 90<	egiste	H su	L	1 [0100:000	9	Γ	0100:0009	
BX 23 12 01002: 15 021 § CX A1 52 01003: BB 187 1 CX A1 52 01003: BB 187 1 DX 00 01005: 23 035 # 0U 0X, 000003h DX 00 01006: B9 185 1 HOU DX, 000003h DX 00 01006: B9 185 1 HU DX, 000003h CS 0100 01009: 03 053 V HU DX P 0009 01008: 03 053 V NOP NOP SS 0100 01008: 03 063 V NOP NOP SP FFFE 0100E: 03 003 V NOP NOP SI 0000 01010: F7 247 X NOP NOP SI 0000 01010: F7 247 NOP NOP 010000 01013: 90 144 K NOP NOP 01 0009 01013: 90 144	AX	(and the second	4A		0: B8 184	1 -	MOU	AX. 0154Ah	3
CX A1 52 01004: 12 018 1 ADD AX, CX 0X 08 00 01005: 23 035 1 MOU DX, 00003h 0X 08 00 01006: B9 185 1 DIU DX 0X 08 00 01006: B9 185 1 DIU DX 01000 01008: A1 161 1 1 NOP 00009 01008: A3 003 ♥ NOP SS 0100 01000: BA 186 NOP SP FFFE 01000: BA 186 NOP BP 0000 01005: F7 247 ≈ NOP SI 0000 01011: F2 242 ≥ NOP DI 0000 01013: 90 144 € NOP	BX	23	12	0100	2: 15 021		MOU	CX, ØA152h	
0X 00 01005: 23 035 # MOU DX, 00003h 0X 01006: B9 185 // 01006: B9 185 // DIU DX 01000 01008: A1 161 1 NOP 0F 01009 01008: A3 195 + 0F 01008: A3 195 + NOP NOP 01008: A3 195 + NOP SS 0100 01008: A3 003 ♥ SP FFFE 01000: BA 186 NOP 01008: P7 247 ≈ SI 0000 01011: F2 242 ≥ DI 0009 01012: F4 244 € DI 01009 01013: 90 144 €	CX	A1	52				ADD I	AX, CX	
B1887: 52 082 R HLT B1908: A1 161 1 NOP IP 0009 01009: 03 003 NOP SS 0100 01008: 013 NOP SS 0100 01008: 013 NOP SP FFFE 01008: B1008: NOP BP 0000 01008: F7 247 NOP SI 0000 01011: F2 242 NOP DI 0009 01013: 90 144 NOP		00	00			# {	MOU	DX, 00003h	
IP 0009 0100A: C3 195 NOP SS 0100 0100B: 03 003 ♥ NOP SP FFFE 0100D: BA 186 NOP SP 0100E: 03 003 ♥ NOP BP 0000 0100F: 00 000 NULL NOP SI 0000 0101: F2 242 ≥ NOP DI 0000 0101: F4 244 € NOP		-	_	8188 0188	7: 52 082 8: A1 161	1	HLT NOP		
55 0100 0100C: C1 193 ⊥ NOP SP FFFE 0100D: BA 186 II NOP BP 0000 0100F: 00 0003 ♥ NOP SI 0000 01011: F2 247 ≈ NOP DI 0000 01012: F4 244 € NOP DI 0000 01013: 90 144 € NOP	IP	00	09	0.2.0.0		ř.			
SP FFFE 0100D: BA 186 NOP BP 0000 0100F: 00 0003 ♥ NOP BP 0000 0100F: 00 0000 NULL NOP SI 0000 01011: F2 247 ≈ NOP DI 0000 01012: F4 244 NOP DI 0000 01013: 90 144 NOP	SS	01	88	10.0.0707					
BP 06000 0100F: 00 0000 NOP SI 0000 01011: F2 247 ≈ NOP DI 0000 01012: F4 244 f NOP DI 0000 01013: 90 144 f NOP	SP	FF	FE	0100	D: BA 186		NOP		
SI 0000 01011: F2 242 NOP DI 0000 01013: 90 144 NOP DI 01014: 90 144 NOP	BP	00	00	0100	F: 88 089	NULL	NOP		
DI 0000 01012: F4 244 C NOP 01013: 90 144 E NOP 01014: 90 144 E	SI	00	00	0101	1: F2 242		NOP		
81814: 90 144 É		0.0	00	0101	2: F4 244 3: 90 144	٤ ا			
		-					100000		

Department of Electronics & Communication Engineering

GROUP OF COLLEGES

1	OLLEGE OF TECH	
	ARE-132	1331
WISHMI		SCIENC
- 🔇		1
	गण लमत	

Loa	8	oload Step back	single step	n step dela	ay ms: 0
egiste	rs — L	0100:0012		0100:0012	-
AX	D9 AE	01012: F4 244 f	A HLT		
ВX	23 12	01013: 90 144 É 01014: 90 144 É	- NOP NOP		
cx	A1 52	01015: 90 144 É 01016: 90 144 É	NOP		
DX	00 03	01017: 90 144 É 01018: 90 144 É	NOP		
CS	0100	01019: 90 144 É 0101A: 90 144 É	NOP		
IP	0012	0101B: 90 144 É 0101C: 90 144 É	NOP		
SS	0100	0101D: 90 144 É 0101E: 90 144 É	- NOP NOP		
SP	FFFE	0101F: 90 144 É	NOP		
BP	0000	01020: 90 144 É 01021: 90 144 É	NOP		
SI	0000	01022: 90 144 É 01023: 90 144 É	NOP		
DI	0000	01024: 90 144 É 01025: 90 144 É	NOP		
2.	0100	01026: 90 144 É			

Result: Average of three numbers has been evaluated.



Experiment – 8 Exchanging the data of registers

Aim: Write a program in 8086 to exchange the data of the registers.

Program: -

MOV AX, 0004H; MOV BX, 0003H; MOV DX, BX; MOV BX, CX; MOV CX, AX; MOV AX, DX; HLT;

Load reload step back single step run step delay	2		g view			1			8 51	7 13	1
AX 06 04 01000: B8 B8 184 1 MOU AX, 000004h BX 00 03 01001: 04 004 • MOU BX, 00003h BX 00 03 01002: 00 0000 NULL MOU CX, 000002h CX 09 02 01004: 03 03 • MOU BX, CX, 000002h DX 00 00 01005: 00 0000 NULL MOU BX, CX DX 00 00 01007: 02 002 00 MOU AX, DX DX 00 00 01007: 02 002 00 MOU AX, DX MOU BX, CX 01006: B9 185 1 MOU AX, DX MOU AX, DX 01007: 02 002 00 MOU AX, DX MOU AX, DX 01006: B9 185 1 NOP NOP SS 0100 01007: B8 139 NOP NOP SF FFFE 0100E: C2 194 NOP NOP SI 0000 01012: 90 144 NOP NOP NOP		0	reload	step back	1.5	ep	run	ste	p dela	y ms: I	0
BX 00 03 01001: 04 • MOU BX, 00003h BX 00 03 01002: 00 000 NULL MOU BX, 00002h CX 00 02 01003: BB 187 1 MOU DX, 00002h DX 00 02 01004: 03 003 • MOU DX, 00002h DX 00 00 01005: 00 0000 NULL MOU DX, 00 DX 00 00 01007: 02 002 C MOU AX, DX DX 01007: 02 002 C MOU AX, DX MOU AX, DX DX 01007: 02 002 C MOU AX, DX MOU AX, DX IP 00099 01008: 00 000 NULL NOP NOP SS 0100 01001: BB 139 I NOP NOP SI 0000 01011: F4 244 MOP NOP	egisters	s H L		0100:0009				0100:0	009	-	
BX 00 03 01002: 00 000 NULL MOU CX 00002h CX 00 02 01003: BB 187 1 MOU DX 03 DX 00 02 01005: 00 000 NULL MOU DX BX DX 00 00 01005: 00 000 NULL MOU DX BX DX 00 00 01005: 00 000 NULL MOU DX DX DX 00 00 01007: 02 002 C MOU AX DX DX 01007: 02 002 C MOU AX DX DX 01007: 02 002 C MOU AX DX BY 01008: 00 000 NULL NOP NOP NOP SS 0100 0100E: BS 139 I NOP NOP SI 0000 01011: F4 244 <t< th=""><th>ax [</th><th>00 04</th><th></th><th></th><th>1</th><th></th><th>IOU AX</th><th>, 0000</th><th></th><th></th><th>- 1</th></t<>	ax [00 04			1		IOU AX	, 0000			- 1
CX 00 02 01004::03:003 * MOU BX, CX DX 00 00 01005::00:000 NULL MOU MOU CX, AX DX 00 00 01006::B9:185;1 MOU AX, DX CS 0100 01007::02:002 02 MOU AX, DX P 0009 01008::00:000 NULL NOP SS 0100 01008::00:000 NULL NOP SS 0100 01008::88:139:1 NOP NOP SF FFFE 0100E::88:139:1 NOP NOP SP FFFE 0100E::88:139:1 NOP NOP SI 0000 01011::F4:244 NOP NOP DI 0000 01013::90:144 NOP NOP DI 0000 01014::90:144 144 NOP NOP	BX [00 03	0100	2: 00 000			IOU CX	, 0000			
DX 00 00 01005: 00 000 NULL MOU CX, AX 01006: B9 185 1 MOU AX, DX 01007: 02 002 B MOU AX, DX 01007: 02 002 B HLT NOP 01007: 02 000 NULL NOP 01009: 8B 139 1 NOP SS 0100 0100E: D9 217 NOP SP FFFE 0100E: CS 2002 NOP NOP 0100E: SB 139 NOP NOP SP FFFE 0100E: CS 200 NOP NOP SI 0000 0101F: SB 139 NOP NOP SI 0000 01011: F4 244 NOP NOP DI 0000 010112: 90 144 NOP NOP 01014: 29 144 144 NOP NOP	cxΪ	00 02]		ON BX	, BX , CX	iolendiae		
CS 0100 01008: 00 000 NULL NOP IP 0009 01009: 8B 139 ï NOP SS 0100 01008: 8B 139 ï NOP SF 0100 0100E: D9 217 j NOP SP FFFE 0100E: C8 200 Ľ NOP BP 0000 0100F: 8B 139 ï NOP SI 0000 01011: F4 244 ſ NOP DI 0000 01013: 90 144 € NOP DI 01014: 90 144 € NOP		00 00	0100	6: B9 185	1	P P	IOU CX	, AX			
IP 0009 0100A: D3 211 H NOP SS 0100 0100B: 8B 139 I NOP SP FFFE 0100C: D9 217 J NOP SP FFFE 0100D: 8B 139 I NOP BP 0000 0100F: 8B 139 I NOP SI 0000 01010: C2 194 T NOP SI 0000 01011: F4 244 I NOP DI 0000 01013: 90 144 É NOP 01014: 90 144 É NOP NOP	CS	0100	tor do tor tor								
SS 0100B: 8B 139 NOP SP FFFE 0100C: D9 217 NOP SP FFFE 0100D: 8B 139 NOP BP 0000 0100F: 8B 139 NOP SI 0000 01011: F4 244 NOP DI 0000 01013: 90 144 NOP NOP 01014: 90 144 NOP	IP	0009	0100		ï						
SP FFFE 0100D: 8B 139 I NOP BP 0000 0100F: 8B 139 I NOP SI 0000 01010: C2 194 T NOP SI 0000 01011: F4 244 NOP NOP DI 0000 01013: 90 144 NOP NOP	SS	0100	0100	B: 8B 139	ï	- N	IOP				
BP 0000 0100E: C8 200 E NOP 0100F: 8B 139 I NOP 01010: C2 194 T NOP SI 0000 01011: F4 244 f NOP DI 0000 01013: 90 144 é NOP 01014: 90 144 é NOP	SP		0100	D: 8B 139		N	IOP				
SI 01010: C2 194 T NOP 01011: F4 244 r NOP			0100	F: 8B 139		N	IOP				
DI 01012: 90 144 é NOP DI 01000 01013: 90 144 é NOP 01014: 90 144 é NOP NOP		The strength of the strength of the		the second se	Ţ						
01014: 90 144 6						N N	IOP				
						-					
ES 0100 arrean lacura l			1					1911			2

GROUP OF COLLEGES

1	OLLEGE OF TECH	6
	ALT:	SOLES
ESA	The second	CENC.
	गणन लमते जाग	/

Loa	2	r	D eload	step	dl back		▶ e step	ru	> n	step del	, , , lay ms: 0	
egiste	ers H	Ľ] [0100):001	1			01	00:0011		
AX	00	03	0100		184	1	-	MOU	AX,	00004h		
ВX	00	02	0100	2: 00	004	NULL		MOU	BX, CX,	00002h		
cx	00	84	0100		187 003	₽.		MOU	DX, BX,	BX CX		
DX	00	03	0100		000 185	NULL		MOU	CX, AX,	AX		
CS	ß1	00	0100		002 000	ÖNULL		HLT NOP				
IP		11	0100	9: 8B	139	ï		NOP				
SS	-	00	0100	B: 8B	139	j		NOP				
SP	FF	FE	0100	D: 8B	139	ï		NOP				
ΒP	00	00	0100	F: 8B	139	ï		NOP				
SI	00	00	0101 0101	1: F4	244	Ţ		NOP				
DI	00	00	0101	3: 90	144 144	É		NOP NOP				
DS	01	00	0101	4: 90	144	É	-					

Result: Ex-changing the data of the registers has been performed.



Experiment – 9 Finding factorial of a number

Aim: Write a program to find factorial of a number in 8086.

Program: -

MOV AX,0001H; MOV BX,0005H; MUL BX; DEC BX: JNZ L1; HLT;

Loa	₽ ad	r	o eload s	⊲I tep back		step	ru 🔊	n step delay ms:	0
registe	ers H	L	01	00:000	В		1	0100:000B	
AX	00	78		B8 184	1		MOU	AX, 00001h	_
в×	00	00			© NULL		MOU	BX, 00005h BX	
cx	00	00		BB 187 05 005	ิส		DEC	BX Ø6 h	
	-		01005:	000 000	NULL		HLT	207	
DX	00	00		F7 247 E3 227	х т		NOP		
CS	01	00		4B 075	K		NOP		
IP	00	OB		75 117 FB 251	u		NOP		
SS	10000	00	0100B:	F4 244	ſ		NOP		
	-		0100C: 0100D:	90 144 90 144	É		NOP NOP		
SP	FF	FE		90 144	Ĕ		NOP		
BP	00	00		90 144 90 144	É		NOP		
SI	00	00	01011:	90 144 90 144	É		NOP		
DI		00	the set of the set of	90 144 90 144	É		NOP		
	00			90 144	Ē		INVI		

Result: Factorial of a number in 8086 has been evaluated.



Experiment – 10 Stepper Motor Rotation

Aim: Write a program to rotate stepper motor using 8051

Program: -0000 MOV TMOD, #50H ; put timer 1 in event counting mode 0003 SETB TR1 ; start timer 1 0005 MOV DPL, #LOW(LEDcodes) ; | put the low byte of the start address of the ; 7-segment code table into DPL 0008 MOV DPH, #HIGH(LEDcodes) ; put the high byte into DPH 000B CLR P3.4 ; **CLR P3.3** | enable Display 0 000D again: 000F CALL setDirection ; set the motor's direction 0011| MOV A, TL1 ; move timer 1 low byte to A CJNE A, #10, skip ; if the number of 0013 revolutions is not 10 skip next instruction CALL clearTimer 0016 ; if the number of revolutions is 10, reset timer 1 skip: MOVC A, @A+DPTR ; | get 7-segment code 0018 from code table - the index into the table is ; | decided by the value in A ; | (example: the data pointer points to the start of the ; | table - if there are two revolutions, then A will contain two, ; | therefore the second code in the table will be copied to A) MOV C, FØ ; move motor direction value 0019 to the carry



; | and from there to ACC.7 001B MOV ACC.7, C (this will ensure Display 0's decimal point ; | will indicate the motor's direction) ; | move (7-seg code for) 001D MOV P1, A number of revolutions and motor direction ; | indicator to Display 0 001F JMP again ; do it all again setDirection: 0021 PUSH ACC ; save value of A on stack 0023 PUSH 20H ; | save value of location 20H (first bit-addressable ; | location in RAM) on stack CLR A MOV 20H, #0 ; clear A 0025 ; clear location 20H 0026 0029| MOV C, P2.0 MOV ACC.0, C ; put SW0 value in carry 002B ; then move to ACC.0 002D MOV C, F0 ; move current motor direction in carry 002F MOV 0, C ; and move to LSB of location 20H (which has bit address 0) 0031 CJNE A, 20H, changeDir ; | compare SW0 (LSB of A) with F0 (LSB of 20H) ; | - if they are not the same, the motor's direction needs to be reversed ; if they are the same, 0034 JMP finish motor's direction does not need to be changed changeDir: 0036 CLR P3.0 ; | stop motor 0038 CLR P3.1 ; reset timer 1 003A CALL clearTimer (revolution count restarts when motor direction changes) MOV C, P2.0; move SW0 value to carry 003C

Department of Electronics & Communication Engineering



003E MOV F0, C ; and then to F0 - this is the new motor direction 0040 MOV P3.0, C ; move SW0 value (in carry) to motor control bit 1 CPL C 0042 ; invert the carry MOV P3.1, C 0043 ; and move it to motor control bit 0 (it will therefore have the opposite ; | value to control bit 1 and the motor will start ; | again in the new direction) finish: POP 20H 0045 ; get original value for location 20H from the stack 0047 POP ACC ; get original value for A from the stack 0049 : return from subroutine RFT clearTimer: 004A CLR A ; reset revolution count in A to zero 004B CLR TR1 ; stop timer 1 004D MOV TL1, #0 ; reset timer 1 low byte to zero SETB TR1 ; start timer 1 0050 0052 ; return from subroutine RET LEDcodes: ; this label points to the start address; of the 7-segment code table which is ; | stored in program memory using the DB command below DB 11000000B, 11111001B, 10100100B, 10110000B, 10011001B, 10010010B, 10000010B, 11111000B, 10000000B, 1001000B







Result: Stepper motor rotation has been performed using 8051 programming.