



## LABORATORY MANUAL

**B.Tech. Semester- IV**

**MICROCONTROLLER LAB**

**Subject code: LC-ECE-214G**

**Prepared by:**

Mrs. Monika Thakur

**Checked by:**

Mrs. Dimple Saproo

**Approved by:**

Name : Prof. (Dr.) Isha Malhotra

**Sign.: .....**

**Sign.: .....**

**Sign.: .....**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING  
DRONACHARYA COLLEGE OF ENGINEERING  
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)**

## **Table of Contents**

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objective (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (Cos)
8. CO-PO and CO-PSO Mapping
9. Course Overview
10. List of Experiments
11. Dos and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Details of Conducted Experiments
16. Lab Experiments

## **Vision and Mission of the Institute**

### **Vision:**

To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and Managers competent to match the expanding expectations of the Corporate World has been our ever enlarging vision extending to new horizons since the inception of Dronacharya College of Engineering.

### **Mission:**

- M1.** To prepare students for full and ethical participation in a diverse society and encourage lifelong learning by following the principle of 'Shiksha evam Sahayata' i.e. Education & Help.
- M2.** To impart high-quality education, knowledge and technology through rigorous academic programs, cutting-edge research, & Industry collaborations, with a focus on producing engineers & managers who are socially responsible, globally aware, & equipped to address complex challenges.
- M3.** Educate students in the best practices of the field as well as integrate the latest research into the academics.
- M4.** Provide quality learning experiences through effective classroom practices, innovative teaching practices and opportunities for meaningful interactions between students and faculty.
- M5.** To devise and implement programmes of education in technology that are relevant to the changing needs of society, in terms of breadth of diversity and depth of specialization.

## **Vision and Mission of the Department**

### **VISION**

To be a Centre of Excellence for producing high quality engineers and scientists capable of providing sustainable solutions to complex problems and promoting cost effective indigenous technology in the area of Electronics, Communication & Control Engineering for Industry, Research Organizations, Academia and all sections of society.”

### **MISSION OF THE DEPARTMENT**

- M1** To frame a well-balanced curriculum with an emphasis on basic theoretical knowledge as well the requirements of the industry.
- M2.** To motivate students to develop innovative solutions to the existing problems for betterment of the society.
- M3.** Collaboration with the industry, research establishments and other academic institutions to bolster the research and development activities in department.
- M4.** To provide infrastructure and support for culmination of novel ideas into useful prototypes.
- M5.** To promote research in emerging and interdisciplinary areas and act as a facilitator for knowledge generation and dissemination through Research, Institute – Industry and Institute-Institute interaction.

## **Programme Educational Objectives (PEOs)**

**PEO1:** To practice the profession of engineering using a systems perspective and analyze, design, develop, optimize & implement engineering solutions and work productively as engineers, including supportive and leadership roles on multidisciplinary teams

**PEO2:** To Continue their education in leading graduate programs in engineering & interdisciplinary areas to emerge as researchers, experts, educators & entrepreneurs and recognize the need for, an ability to engage in continuing professional development and life-long learning

**PEO3:** To Engineers, guided by the principles of sustainable development and global interconnectedness, will understand how engineering projects and affect society and the environment.

**PEO4:** To Promote Design, Research and implementation of products and services in the field of Engineering through strong Communication and Entrepreneurial skills.

**PEO5:** To Re-learn and innovate in ever-changing global economic and technological environments on the 21<sup>st</sup> century.

## Programme Outcomes (POs)

**PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and software tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Programme Specific Outcomes (PSOs)**

**PSO1.** Equip themselves to potentially rich & employable field of Engineering. Analyse and design electronic systems for signal processing, communications and other applications.

**PSO2.** Pursue higher studies in the contemporary Technologies and multidisciplinary fields with an inclination towards continuous learning. area of Electronics, Telecommunication ,VLSI or Instrumentation.

**PSO3.** Design, implement and evaluate processes, components and/or programs using modern techniques, skills and tools of core Information Technologies to effectively integrate effective communication-based solutions into using Electronic components.

**PSO4.** Develop impactful solutions by using research-based knowledge and methods in the fields of integration and implementation, alongside Meeting the requirements of the Industrial standard.

## **University Syllabus**

1. To study the architecture of 8086 microprocessor and 8086 microprocessor kit.
2. Write a program to add the contents of the memory location to the content of other memory location and store the result in 3rd memory location.
3. Write a program to add 16 bit number using 8086 instruction set.
4. Write a multiplication of two 16 bit numbers using 8086 instruction set.
5. Write a program for division of two 16 bit numbers using 8086 instruction set.
6. Write a program factorial of a number.
7. Write a Program to transfer a block of data with & without overlap.
8. Write a program to find the average of two numbers.
9. Write a Program to check whether data byte is odd or even
10. Write a program to find maximum number in the array of 10 numbers.
11. Write a program to find the sum of the first 'n' integers.
12. Write a program to generate a square wave.
13. Write a program to generate a rectangular wave.
14. Write a program to generate a triangular wave.



### Course Outcomes (COs)

Upon successful completion of the course, the student will be able to:

CO1. Do assembly language programming of 8086.

CO2. Do assembly language programming of 8086 for interfacing of peripherals.

#### CO-PSO Mapping

	PSO1	PSO2	PSO3	PSO4
CO1	2	2	3	2
CO2	2	2	3	2
Average	2	2	3	2

#### CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	1	2	2	1	1		2		1	2	1
CO2	1	1	2	1	2	3				3	2	2
Average	2	1	2	1.5	1.5	2		1		2	2	1.5

## Course Overview

The Microprocessors and Microcontrollers Lab course is designed to provide students with hands-on experience in working with microprocessors and microcontrollers and to enhance their understanding of embedded systems. This lab-based course complements the theoretical knowledge gained in the Microprocessors and Microcontrollers theory course, allowing students to apply their knowledge in practical applications. The course typically covers the topics: Introduction,overview of microprocessors and microcontrollers and their applications. It gives differentiating information between microprocessors and microcontrollers based on their characteristics and applications. It gives information about writing assembly language programs to perform basic arithmetic operations and control flow. Practicing instruction sets, addressing mode s, and assembly language programming techniques specific to the microprocessors being used. It also gives information about microcontroller Programming,overview of microcontroller architecture and programming model. It gives information about interfacing microprocessors and microcontrollers with peripheral devices such as switches, LEDs, LCDs, and sensors. Implementing different communication protocols (UART, SPI, I2C) to establish communication with external devices. Configuring and utilizing timers, interrupts, and other peripheral modules for efficient device control and data transfer.

**List of Experiments mapped with COs**

<b>S. No.</b>	<b>Name of the Experiments</b>	<b>Course Outcome</b>
1	1. To study the architecture of 8086 microprocessor and 8086 microprocessor kit.	CO1,
2	Write a program to add the contents of the memory location to the content of other memory location and store the result in 3rd memory location.	CO1
3	Write a program to add 16 bit number using 8086 instruction set.	CO1
4	Write a multiplication of two 16 bit numbers using 8086 instruction set.	CO1
5	Write a program for division of two 16 bit numbers using 8086 instruction set.	CO1
6	Write a program factorial of a number.	CO1
7	Write a Program to transfer a block of data with & without overlap.	CO1
8	Write a program to find the average of two numbers.	CO1
9	Write a Program to check whether data byte is odd or even	CO1
10	Write a program to find maximum number in the array of 10 numbers.	CO1
11.	Write a program to find the sum of the first 'n' integers.	CO1,CO2
12.	Write a program to generate a square wave.	CO1,CO2
13.	Write a program to generate a rectangular wave.	CO1,CO2
14	Write a program to generate a triangular wave.	CO1,CO2

### Dos and DONT's

#### Dos

1. Do follow the lab rules and guidelines provided by your instructor or lab supervisor.
2. Do wear appropriate safety gear, such as safety glasses or gloves, when working with electronic components or soldering equipment.
3. Do handle microprocessors and other electronic components with care to avoid damage.
4. Do organize your workspace and keep it clean and tidy to prevent accidents and confusion.
5. Do double-check your connections and circuit configurations before applying power to avoid short circuits or other electrical issues.
6. Do ask questions and seek clarification from your instructor or lab assistants if you are unsure about any aspect of the lab experiment or procedure.
7. Do backup your work regularly, especially if you are working on programming or design projects.
8. Do document your work, including circuit diagrams, code snippets, and observations, for future reference and troubleshooting.
9. Do collaborate and engage in discussions with your lab partners or classmates, as it can enhance your understanding and problem-solving skills.
10. Do explore additional resources, such as textbooks, online tutorials, or datasheets, to deepen your knowledge of microprocessors and related concepts.

#### DON'Ts

1. Don't eat, drink, or smoke in the lab, as it can pose a safety hazard and damage the equipment.
2. Don't leave your workspace unattended while your experiment is running or when equipment is powered on.
3. Don't attempt to repair or modify equipment without proper authorization and guidance.
4. Don't rush or take shortcuts in your experiments or projects. Take your time to understand the concepts and follow the correct procedures.
5. Don't ignore warning signs, such as abnormal smells, smoke, or overheating of equipment. Report any issues to your instructor or lab supervisor immediately.
6. Don't rely solely on your lab partners' work or solutions. Each student should actively participate and contribute to the lab activities.
7. Don't hesitate to seek help if you encounter difficulties or challenges during the lab. Your instructor or lab assistants are there to assist you.

## **General Safety Precautions**

### **Precautions (In case of Injury or Electric Shock)**

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

### **Precautions (In case of Fire)**

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

**Emergency: Reception**

**Security : Main Gate**

## **Guidelines to students for report preparation**

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

- 1) All files must contain a title page followed by an index page. **The files will not be signed by the faculty without an entry in the index page.**
- 2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.
- 3) For each experiment, the record must contain the following
  - (i) Aim/Objective of the experiment
  - (ii) Apparatus required with specification and Name plate details
  - (iii) Program, flowchart, memory locations.
  - (v) Results/ output

### **Note:**

1. Students must bring their lab record along with them whenever they come for the lab.
2. Students must ensure that their lab record is regularly evaluated.

### Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
<b>AC1: Pre-Lab written work (this may be assessed through viva)</b>	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
<b>AC2: Flow Chart / Connection</b>	Flow chart must be neatly drawn and / should be sequentially connected	Flow chart drawn and connection given	Circuit diagram and connection to be given as per directions.	Unable to draw flow chart connection as per circuit diagram.
<b>AC3: Identification of problems in running the equipment and note down the reading</b>	Able to identify the mistakes while running the machine and note down the reading accurately by varying all the related parameters	Able to identify the mistakes while running the machine and note down the reading by varying the parameters	Only few readings are taken and varying parameter is not proper	Unable to identify the mistakes
<b>AC4: Final Demonstration and Execution</b>	All variants of input /output are measured, experiment is well demonstrated and implemented concept is clearly explained	All variants of input /output are not measured, experiment is demonstrated and implemented concept is clearly explained	Only few variants are measured, experiment is demonstrated and implemented concept is not clearly explained	Not well demonstrated and not explained the concept
<b>AC5: Lab Record Assessment</b>	All the readings are properly recorded and model calculations properly executed and performance analysis- results are plotted with graph (if necessary)	70 % calculations are done results and performance analysis are plotted with graph	Less than 70 % calculations are done results are plotted with graph if necessary	Not completed

# LAB EXPERIMENTS



## **LAB EXPERIMENT No. 1**

**AIM:-** To study the architecture of 8086 microprocessor and 8086 microprocessor kit.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

### **THEORY:-**

The 8086 is a 16-bit, N-channel, HMOS microprocessor. The term HMOS is used for “high-speed MOS”. The 8086 uses 20 address lines and 16 data lines. It can directly address up to  $2^{20} = 1\text{Mbytes}$  of memory. The 16-bit data word is divided into a low-order byte and a high-order byte. The 20 address lines are time multiplexed lines. The 16 low-order address lines are time multiplexed with data, and the 4 high-order address lines are time multiplexed with status signals

### **OPERATING MODES OF 8086**

There are two modes of operation for Intel 8086, namely the minimum mode and the maximum mode. When only one 8086 CPU is to be used in a microcomputer system the 8086 is used in the minimum mode of operation. In this mode the CPU issues the control signals required by memory and I/O devices. In case of maximum mode of operation control signals are issued by Intel 8288 bus controller which is used with 8086 for this very purpose. When MN/MX' is high the CPU operates in the minimum mode. When it is low the CPU operates in the maximum mode.

### **Pin Description For Minimum Mode**

For the minimum mode of operation the pin MN/MX' is connected to 5V d.c supply. The description of the pins from 24 to 31 for the minimum mode is as follows:

**INTA(Output):** Pin no. 24 Interrupt acknowledge. On receiving interrupt signal the processor issues an interrupt acknowledge signal. It is active LOW.

**ALE(Output) :** Pin no. 25 Address latch enable. It goes HIGH during T1. The microprocessor sends this signal to latch the address into the Intel 8282/8283 latch.

**DEN(Output) :** Pin no. 26 Data enable. When Intel 8286/8287 octal bus transceiver is used this signal acts as an output enable signal. It is active LOW.

## MICROCONTROLLERS LAB MANUAL PCC-ECE-214G

**DT/R' (Output)** : Pin no. 27 Data Transmit/Receive. When Intel 8286/8287 octal bus transceiver is used this signal controls the direction of data flow through the transceiver. When it is High data are sent out. When it is LOW data are received.

**M/IO(Output)** : Pin no. 28.Memory or I/O access. When it is HIGH the CPU wants to access memory. When it is LOW the CPU wants to access I/O device.

**WR (Output)** : Pin no. 29. Write. When it is LOW the CPU performs memory or I/O write Operation.

**HLDA (Output)** : Pin no. 30.HOLD acknowledge. It is issued by the processor when it receives HOLD signal. It is active HIGH signal. When HOLD request is removed HLDA goes LOW.

**HOLD (Output)** : Pin no. 31.Hold. when another device in microcomputer system wants to use the address and data bus, it sends a HOLD request to CPU through this pin. It is an active HIGH signal.

### Pin Description For Maximum Mode

For the maximum mode of operation the pin MN/MX is made  $\overline{LOW}$ . It is grounded. The description of the pins from 24 to 31 is as follows:

**QS1, QS0(Output)**: Pin no. 24,25 Instruction Queue status. Logic are given below:

QS1	QS0	
0	0	No operation
0	1	1 <sup>st</sup> byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

**$\overline{S0}, \overline{S1}, \overline{S2}$  (Output)** : Pin nos. 26,27,28.status signals. These signals are connected to the bus controller Intel 8288.The bus controller generates memory and I/O access control signals. Table for status signals is :

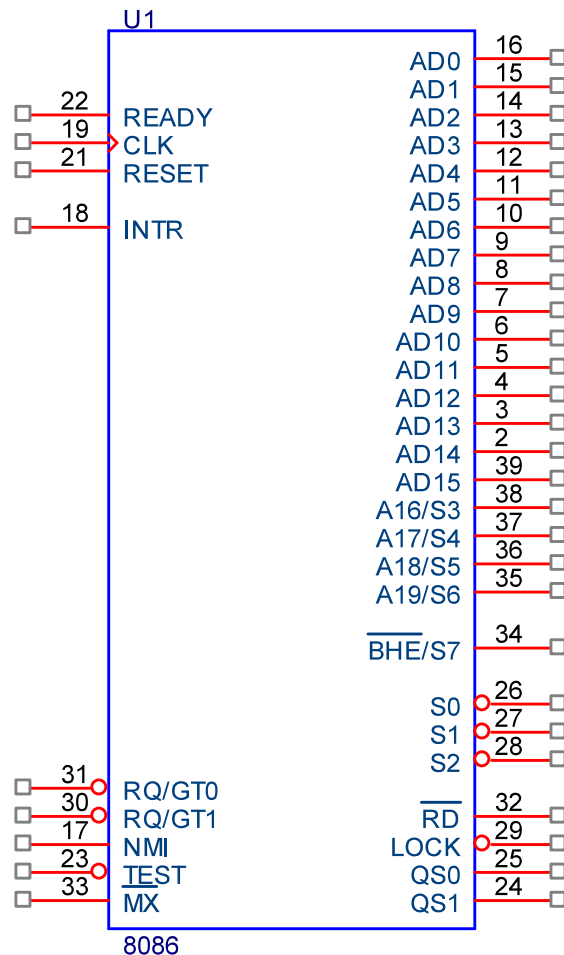
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	0	0	Interrupt acknowledge
0	0	1	Read data from I/O port
0	1	0	Write data into I/O port
0	1	1	Halt
1	0	0	Opcode fetch

## MICROCONTROLLERS LAB MANUAL PCC-ECE-214G

1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state.

**$\overline{\text{LOCK}}$ (Output)** : Pin no. 29. It is an active LOW signal. When it is LOW all interrupts are masked and no HOLD request is granted. In a multiprocessor system all other processors are informed by this signal that they should not ask the CPU for relinquishing the bus control.

**$\overline{\text{RQ}} / \overline{\text{GT}}_1, \overline{\text{RQ}} / \overline{\text{GT}}_0$  (Bidirectional)** : Pin no. 30,31. Local bus Priority control. Other processors ask the CPU through these lines to release the local bus.  $\overline{\text{RQ}} / \overline{\text{GT}}_1$  has higher priority than  $\overline{\text{RQ}} / \overline{\text{GT}}_0$



PIN DIAGRAM OF 8086

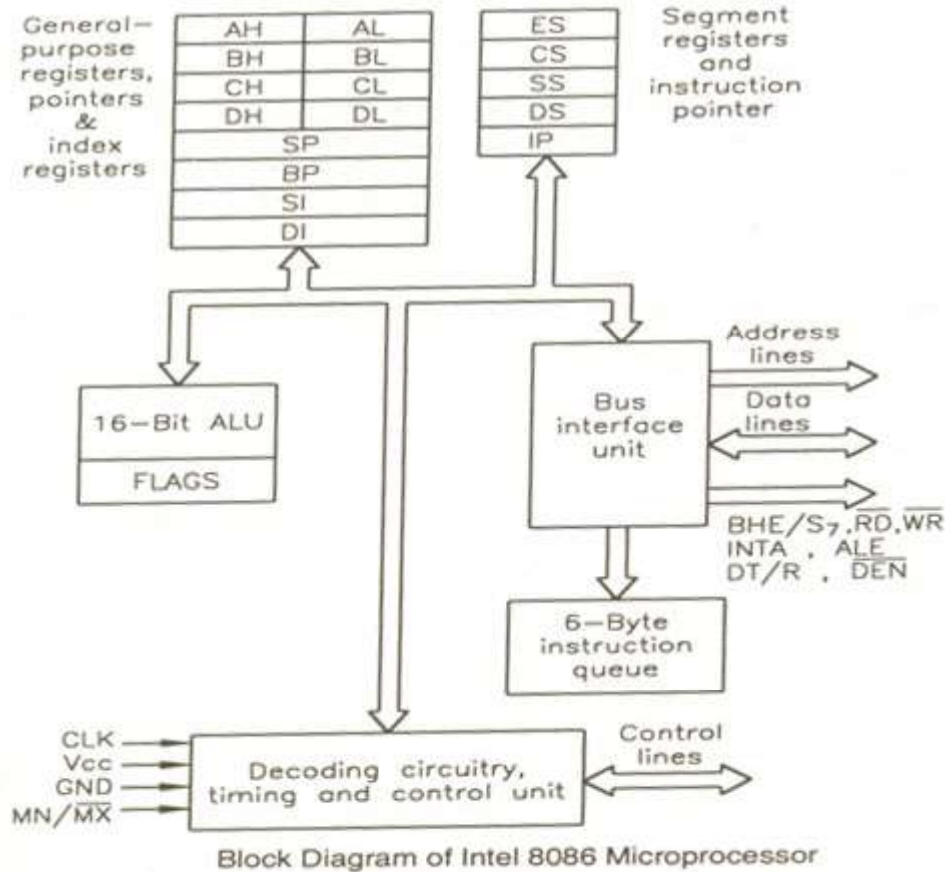
## FUNCTIONAL UNITS OF 8086 :

The 8086 contains two functional units: a bus interface unit (BIU) and an execution unit(EU).

The general purpose registers, stack pointer, base pointer and index registers, ALU, Flag Register (FLAGS), instruction decoder and timing and control unit constitute execution unit(EU).

The segment registers, instruction pointer and 6-byte instruction queue are associated with the bus interface unit(BIU).

## BLOCKDIAGRAM OF 8086:



**REGISTERS OF 8086 :** The Intel 8086 contains the following registers:

- General Purpose Register
- Pointer and Index Registers
- Segment Registers
- Instruction Registers
- Status Flags

## PROCEDURE:-

About Kit NV5586A

## General Description:

NV5586A is a single board microprocessor training/development kit configured around the Intel's 16 bit Microprocessor 8086. This kit can be used to train engineers, to control any industrial process and to develop software for 8086 systems.

The kit has been designed to operate in the max. mode. Co-processor 8087 and I/O Processor 8089 can be added on board.

The Kit communicates with the outside world through an IBM PC compatible Keyboard with 20x2 LCD Display. The kit also has the capacity of interacting with PC.

NV5586A is packed up with powerful monitor in 128K Bytes of factory programmed EPROMS and 32K Bytes of Read/Write Memory. The total memory on the board is 144K Bytes. The system has 72 programmable I/O lines. The serial I/O Communication is made possible through 8251.

For control applications, three 16 bit Timer/Counters are available through 8253. For real time applications, the 8 level of interrupt are provided through 8259. NV5586A provides onboard battery back up for onboard RAM. This saves the user's program in case of power failure.

The onboard resident system monitor software is very powerful. It provides various software commands like BLOCK MOVE, SINGLE STEP, EXECUTE, FILL etc. Which are helpful in debugging/developing software. An onboard line assembler provides user to write program in assembling language.

## System Specifications

**Central Processor :** 8086, 16 bit Microprocessor operating in max. mode.

**Co-Processor Support :** Support 8087 Numeric Data Processor.

**I/O Processor Support :** Support 8089 I/O Processor.

**EPROM :** 128K Bytes of EPROM Loaded with monitor program.

**RAM :** 32K bytes of CMOS RAM with Battery Backup using 3.6V Ni-Cd Battery.

**Parallel :** 72 I/O lines using three nos. of 8255.

**Serial :** RS-232-C Interface using 8251.

**Interrupt :** 8 different level interrupt using 8259.

**Timer/Counter :** Three 16 bit Timer/Counter using 8253.

**Keyboard & Display :** 105 IBM PC Keyboard & 20x2 LCD Display.

**BUS :** All address, data and control signals (TTL Compatible) available at 50 Pin & 20 Pin FRC Connector.

**Power Supply :** 5V/ 2 Amps,  $\pm 12V/250mA$

**Physical Size :** 32.6cm x 25.2cm

**Operating Temp. :** 0 to 50°C.

## Keyboard Mode:

1. Examine/Modify the memory byte locations.
2. Examine/Modify the contents of any of internal register of 8086.
3. Move a block of Data/Program from one location to another location.
4. Fill a particular memory area with a constant.
5. To execute the program in full clock speed.
6. To execute program in single instruction execution.

### **System Installation:**

To install NV5586A the following additional things are required.

1. Connect the External SMPS Power Supply to AC Power and 5 pin connector to the left side on NV5586A Kit.
2. Switch on the Power Supply at the rear end of SMPS supply.
3. A message – **NV5586A 8086 Mic. Tr.** will come on display (PRESS RESET if you do not get - NV5586A 8086 Mic. Tr.
4. Now NV5586A Kit is ready for the user's experiments for Keyboard Mode commands.

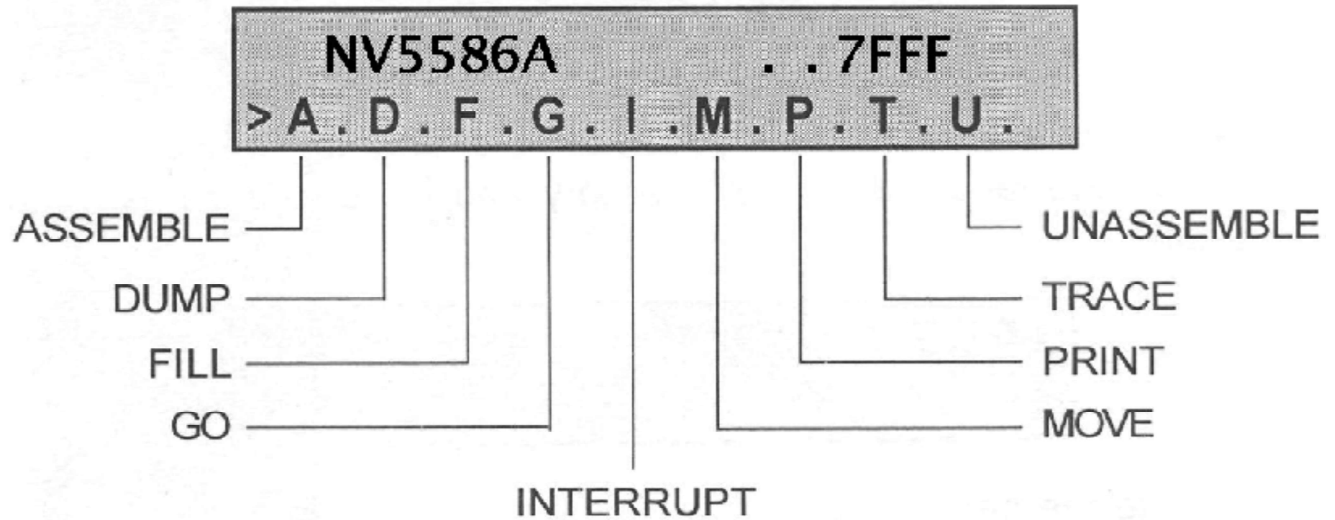
### **Operating Commands:**

After power ON the system, it will display as follows:



```
NV5586A 8086 Mic.Tr.  
ENTER RETURN KEY...
```

After pressing <Enter>, the operating commands will be displayed:



**A – Assemble:**

This command is used to convert the input Assemble Language to the Machine Language in the memory. Once

under this command, first set the address which is similar to the command “D” followed by an Enter or an Arrow Down key to go to a new step. However, only a maximum of 35 words are allowed for input.

**D - Display or modify the RAM’s Hexadecimal:**

A.D.U. is the important commands in the compiling. The effective address or both the effective address and Segment base can be used during input. When the cursor is placed at the beginning, the key will immediately show “F000” as the Segment base and the Effective address next.

**Syntax:**

**D** (If no input, press Enter key or ARROW UP/DOWN key would allow the built-in address to be used)

**F - Fill data into the RAM:**

By setting the starting, ending address and the details, an <Enter> key will allow the data to enter the RAM.

**G - Proceed to the address for execution:**

The GO command, which causes the machine language statements to be executed. This command executes the

loaded program and allows the user to specify the addresses at which program execution will stop.

**I – Interrupt:**

Three Interrupts (Effective address) can be set in for the program execution, the CPU will continuously make a

single-step subprogram for checking IP values. When the IP register has the same value as the Interrupt’s address, it will enter the Interrupt’s subprogram. Enter command “I” will interrupt the program

### **Display Register:**

Command “R” displays the content in the register. This command allows the user to examine the content of the register in the CPU. Each time during display, 4 registers will be shown.

### **M - Moving Data:**

The command MOVE is used to move data in the memory from a specified address to another address by input the starting address, the ending address and the desire address. A RETURN key is then used to execute the changes.

### **T - Trace Program (an N-step designed command):**

This command is used for program execution. TRACE will enter the INTERRUPT subprogram every time the program execute. N has a decimal range from 1-99 with 10 as the rounding off number, and only operate if N is not 0; other-wise it will clear the function.

### **U – Unassemble:**

The UNASSEMBLE command decodes the value of a group memory location mnemonics, and display on the displayed. Once enter this command, input the proper design address.

### **Memory Section: Address Purposes**

<b>Address</b>	<b>Purposes</b>
0000:0000	RAM AREA
0000:7FFF	{ODD RAM & EVENRAM }
F000:0000	ROM MONITOR AREA
F000:FFFF	{ODD ROM & EVEN ROM}

### **I/O Address:**

The addresses of the various chips in I/O mapped in NV5586A are as follows:



---

**MICROCONTROLLERS LAB MANUAL PCC-ECE-214G**

---

Device No.	Port No.	Selected Device
8255-I		PPI
	70	Port A
	72	Port B
	74	Port C
	76	Control Word
8255-II		PPI
	80	Port A
	82	Port B
	84	Port C
	86	Control Word
8255-III		PPI
	10	Port A
	12	Port B
	14	Port C
	16	Control Word
8253		PIT
	00	Counter 0
	02	Counter 1
	04	Counter 2
	06	Counter 3
8259	30	Interrupt controller Data Word
	32	Command/Status Word
8251	50	Data Register
	52	Controller Word Register

## MICROCONTROLLERS LAB MANUAL PCC-ECE-214G

---

### RAM Memory:

Address	Purposes
0000:0000	Interrupt Vector Section (INT1, INT2, INT3 have arranged the interrupt section and stack segment)
	Stack Segment
0000:0390	BUFFER
0000:039B	SYSTEM DATA
0000:93E0	BUFFER (Only if needed)
0000:0400 to 0000:7FFF	USER'S RAM AREA

### System Data of RAM:

0000:039B - Store 9B, will stop at the subprogram exit next to the WAIT command each time it leave the interrupt display subprogram, waiting for F2 to continue execution (is used in TRACE to convert to single-step hardware).

0000:039C

0000:039D } TRACE Buffer

0000:039F - Flags, function of each byte is as followed:

#### BIT:

- 0 : Enter NMI as 1, otherwise as 0
  - 1 : After the "G" key, will be set to as 0, 'SHIFT' + 'F7'
  - 2 : During subprogram, is set to as 1
  - 3 : Set to 1 after entering INTERRUPT
  - 4 : Use in interrupt system
  - 5 : Use in interrupt System
  - 6 : Set 0 to INTERRUPT, and set 1 to TRACE
  - 7 : Set TRACE or INTERRUPT as TF flags, timer 1
- |           |  |   |
|-----------|--|---|
| 0000:03A0 |  | Buffer of Interrupt setting                             |
| 0000:03A5 |  |   |
| 0000:03AE |  | Preserved battery to test bit                           |
| 0000:03AF |  |   |
| 0000:039E |  | Flags, use the command "A"                              |
| 0000:03B0 |  |   |
| 0000:03D8 |  | Data stored in the register monitor during interruption |

**Note:** Address 4350 to 4900 is used for internal operation of trainer and this area is not user accessible.

### NVIS 8086

#### (1) Program Writing Steps

Press Enter

Select A

0000:0400(Program Address)

Write Program up to HLT.

#### (2) Data Writing Steps

F<sub>7</sub>

D 0000:0200(Data Address)

#### (3) Execution Steps

F<sub>7</sub>

I 0000:0000:0000

(FA):(IA):(LA)

F<sub>7</sub>

G 0000:0400

F<sub>7</sub>

D 0000:0300(Result Address)

Or

R (To check result in Register)

**RESULT:-** The 8086 microprocessor architecture, features and understanding of NVIS NV5586 have been studied and ready to perform experiments.

### **PRECAUTION:**

Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Questions / Answers:**

1. Question: What is the architecture of the 8086 microprocessor? Answer: The 8086 microprocessor has a 16-bit architecture with a 16-bit data bus, 16-bit address bus, and a segmented memory model.
2. Question: What are the main registers of the 8086 microprocessor? Answer: The main registers of the 8086 microprocessor are AX, BX, CX, DX, SI, DI, BP, and SP.
3. Question: What is the purpose of the instruction pointer (IP) register? Answer: The instruction pointer (IP) register holds the offset address of the next instruction to be executed in the current code segment.
4. Question: What is the maximum memory capacity that can be addressed by the 8086 microprocessor? Answer: The 8086 microprocessor can address up to 1MB ( $2^{20}$  bytes) of memory.

5. Question: What is the difference between real mode and protected mode in the 8086 microprocessor? Answer: In real mode, the 8086 operates with 16-bit registers and can directly access up to 1MB of memory. In protected mode, the 8086 can access memory beyond 1MB, utilizes a 32-bit register set, and provides memory protection and multitasking features.

**LAB EXPERIMENT No. 2**

**AIM:-** Write a program to add the contents of the memory location to the contents of other memory location and store the result in 3<sup>rd</sup> memory location.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Input Data:

0000:0410 (DS:Offset): 1<sup>st</sup> No. 8-bit

0000:0411(DS:Offset): 2<sup>nd</sup> No. 8-bit

Output Data:

0000:0412-0413: Result 16-bit

CS (Code Segment)	IP (Instruction Pointer)	Mnemonics	Comments
0000	0400	MOV AL, [410]	Copy 1 <sup>st</sup> no. in AL
0000		ADD AL, [411]	Add two No.'s
0000		MOV [412], AL	Store 8-bit LSB of Result into 0412 Address
0000		MOV AL,00	Initialize AL
0000		ADC AL,00	Add CF to AL
0000		MOV [413], AL	Store MSB of Result into 0413 location
0000		HLT	

Data Set Example:

Input Data:

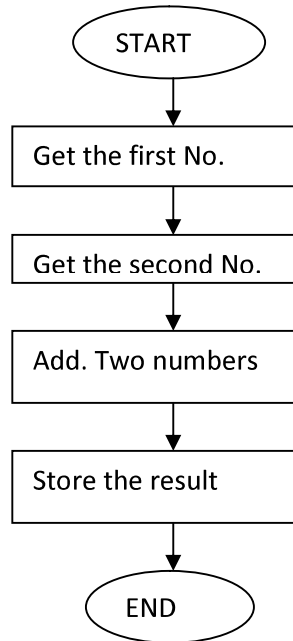
0000:0410 (DS:Offset): 5A

0000:0411(DS:Offset): 54

Output Data:

0000:0412-0413: 00 AE

### FLOW CHART:-



### PROCEDURE:-

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Addition of two numbers is executed.

### **PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Questions /Answers**

Question: How do you copy a data byte in the microprocessor?

Answer: To copy a data byte in the microprocessor, you can use the MOV instruction to move the value from the source memory location or register to the destination memory location or register.

**LAB EXPERIMENT No. 3**

**AIM:-** Write a program to add 16 bit number using 8086 instruction set.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Input Data:

0000:0414-15 (DS:SI): 1<sup>st</sup> No. 16-bit

0000:0416-17(DS:SI+2): 2<sup>nd</sup> No. 16-bit

Output Data:

0000:0418-041A: Result 24-bit

CS (Code Segment)	IP (Instruction Pointer)	Mnemonics	Comments
0000	041B	MOV SI,0414	Initialize SI Pointer
0000		MOV BX, [SI]	Load 1st No. in BX
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		ADD BX, [SI]	Add CF to AL
		INC SI	Increment Pointer
		INC SI	Increment Pointer
		MOV [SI], BX	Save 16-bit LSB of Result
0000		MOV AL,00	Initialize AL with 00
0000		ADC AL,00	Add AL with CF with immediate 00
0000		INC SI	Increment Pointer
		MOV [SI], AL	Store MSB of Result
		HLT	

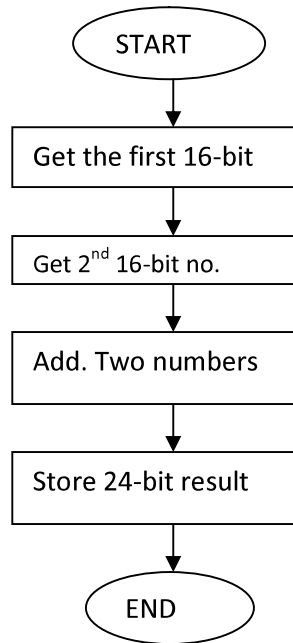
Data Set Example:    Input Data:



0000:0414-15 (DS:SI): 3E 0B    0000:0416-17(DS:SI): 54 A2

Output Data:        0000:0418-041A: 00 92 AD

**FLOW CHART:-**



**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using ‘Assemble-A’.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using ‘Unassemble-U’ note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using ‘Dump-D’ command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using ‘Dump-D’ and verify the result.

**RESULT:** Addition of two 16-bit numbers is performed.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Question / Answers:**

Question 1: How can you perform addition with carry using the 8086 microprocessor?

Answer: To perform addition with carry in the 8086 microprocessor, you can use the ADC (Add with Carry) instruction instead of the ADD instruction. The ADC instruction adds the source operand, along with the carry flag (CF), to the destination operand.

Question 2: How can you add an immediate value to a register using the 8086 microprocessor?

Answer: To add an immediate value to a register in the 8086 microprocessor, you can use the ADD instruction with the desired register.

Question 3: How can you add two 8-bit numbers using the 8086 microprocessor?

Answer: To add two 8-bit numbers in the 8086 microprocessor, you can use the ADD instruction with 8-bit registers.

**LAB EXPERIMENT No. 4**

**AIM:-** Write a program to multiply two 16-bit numbers using 8086 instruction set.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Input Data:

0000:0414-15 (DS:SI): 1<sup>st</sup> No. 16-bit

0000:0416-17(DS:SI+2): 2<sup>nd</sup> No. 16-bit

0000:0418-041B: Result 32-bit

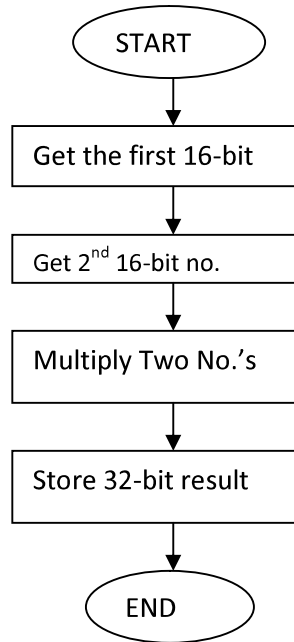
CS (Code Segment)	IP (Instruction Pointer)	Mnemonics	Comments
0000	0420	MOV SI,0414	Initialize SI Pointer at 1 <sup>st</sup> No. location
0000		MOV AX, [SI]	Load 1st No. in AX
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		MOV BX, [SI]	Load 2 <sup>nd</sup> No. in BX
		MUL BX	Multiply two 16-bit no's
		INC SI	Increment Pointer
		INC SI	Increment Pointer
		MOV [SI], AX	Save 16-bit LSB of Result
		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		MOV [SI], DX	Save 16-bit MSB of Result
		HLT	

Data Set Example: Input Data:

0000:0414-15 (DS:SI): 00 FF    0000:0416-17(DS:SI): 00 FF

Output Data:        0000:0418-041A: 00 FE 01

**FLOW CHART:-**



**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using ‘Assemble-A’.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using ‘Unassemble-U’ note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using ‘Dump-D’ command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using ‘Dump-D’ and verify the result.

**RESULT:** Multiplication of two 16-bit numbers is performed.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### Question / Answers

Question 1: How can you multiply two 8-bit numbers using the 8086 microprocessor?

Answer: The 8086 microprocessor does not have a dedicated instruction for multiplying two 8-bit numbers. However, you can use a combination of instructions to perform multiplication.

Question 2: How can you multiply two 16-bit numbers using the 8086 microprocessor?

Answer: The 8086 microprocessor provides the MUL instruction for multiplying two 16-bit numbers.

Question 3: How can you multiply a 16-bit number with an 8-bit number using the 8086 microprocessor?

Answer: The 8086 microprocessor provides the IMUL instruction for signed multiplication.

Question 4: How can you perform multiplication with a constant value using the 8086 microprocessor?

Answer: To multiply a register with a constant value, you can use the MUL instruction with the immediate value.

**LAB EXPERIMENT No. 5**

**AIM:-** Write a program for division of two 16-bit numbers using 8086 instruction set.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Input Data:

0000:0440-0443 (DS:SI): 1<sup>st</sup> No. 32-bit      (Store in DX-AX)

0000:0444-0445(DS:SI+4): 2<sup>nd</sup> No. 16-bit      (Store in CX)

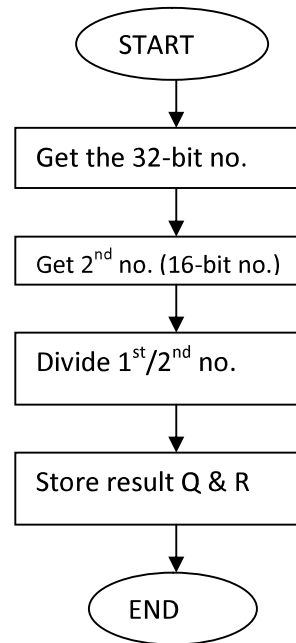
0000:0446-0449: Result 32-bit      (AX = Quotient, DX=Remainder)

CS (Code Segment)	IP (Instruction Pointer)	Mnemonics	Comments
0000	0400	MOV SI, 0440	Initialize SI Pointer at 1 <sup>st</sup> No. location
0000		MOV AX, [SI]	Load LSB of 1 <sup>st</sup> 32-bit No. in AX
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		MOV DX, [SI]	Load MSB of 1 <sup>st</sup> 32-bit No. in DX
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		MOV CX, [SI]	Load 2 <sup>nd</sup> No. in CX
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer
0000		DIV CX	Divide DX.AX / CX
0000		MOV [SI], AX	Save 16-bit Quotient of Result
0000		INC SI	Increment Pointer
0000		INC SI	Increment Pointer

## MICROCONTROLLERS LAB MANUAL PCC-ECE-214G

0000		MOV [SI] , DX	Save 16-bit Remainder of Result
0000		HLT	

### FLOW CHART



### PROCEDURE:-

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Division of two 16-bit numbers is performed.

### **PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Questions/ Answers**

Question 1: How can you divide a 16-bit number by an 8-bit number using the 8086 microprocessor?

Answer: The 8086 microprocessor provides the DIV instruction for unsigned division.

Question 2: How can you divide a 32-bit number by a 16-bit number using the 8086 microprocessor?

Answer: The 8086 microprocessor performs division on 16-bit operands. Therefore, to divide a 32-bit number by a 16-bit number, you will need to perform multiple division operations.

Question 3: How can you perform signed division using the 8086 microprocessor?

Answer: The 8086 microprocessor provides the IDIV instruction for signed division.

Question 4: How can you divide a 16-bit number by a constant value using the 8086 microprocessor?

Answer: To divide a register by a constant value, you can use the DIV instruction with the immediate value.



**LAB EXPERIMENT No. 6**

**AIM:-** Write a program to find the factorial of a number.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

**Assumptions:**

- Starting address of program: 0400
- Input memory location: 0500
- Output memory location: 0600 and 0601

**Point to be noted:**

If the Given Number is a 16-bit number, the AX register is automatically used as the second parameter and the product is stored in the DX:AX register pair. This means that the DX register holds the high part and the AX register holds the low part of a 32-bit number.

**Algorithm:**

1. Input the Number whose factorial is to be find and Store that Number in CX Register  
(Condition for LOOP Instruction)
2. Insert 0001 in AX(Condition for MUL Instruction) and 0000 in DX
3. Multiply CX with AX until CX become Zero(0) using LOOP Instruction
4. Copy the content of AX to memory location 0600
5. Copy the content of DX to memory location 0601
6. Stop Execution

ADDRESS	MNEMONICS	COMMENTS
0400	MOV CX, [0500]	CX ← [0500]
0404	MOV AX, 0001	AX ← 0001
0407	MOV DX, 0000	DX ← 0000
040A	MUL CX	DX:AX ← AX * CX

## **MICROCONTROLLERS LAB MANUAL PCC-ECE-214G**

---

040C	LOOP 040A	Go To [040A] till CX->00
0410	MOV [0600], AX	[0600]←AX
0414	MOV [0601], DX	[0601]←DX
0418	HLT	Stop Execution

### **PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Factorial of 8-bit number is performed.

### **PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Questions/Answers**

Question 1: What is a factorial?

Answer: Factorial is a mathematical operation denoted by the exclamation mark (!). It is used to calculate the product of an integer and all the positive integers below it.

Question 2: How is factorial calculated?

Answer: Factorial is calculated by multiplying the given number by all the positive integers less than it down to 1. For example, 5! (read as "5 factorial") is calculated as  $5 \times 4 \times 3 \times 2 \times 1 = 120$ .

Question 3: How can you calculate factorial using a loop in programming?

Answer: Factorial can be calculated using a loop in programming, such as a for loop or a while loop.

.

Question 4: Is there a limit to the factorial calculation?

Answer: Yes, there is a limit to the factorial calculation due to the limited range of the data type used to store the result..

**LAB EXPERIMENT No. 7**

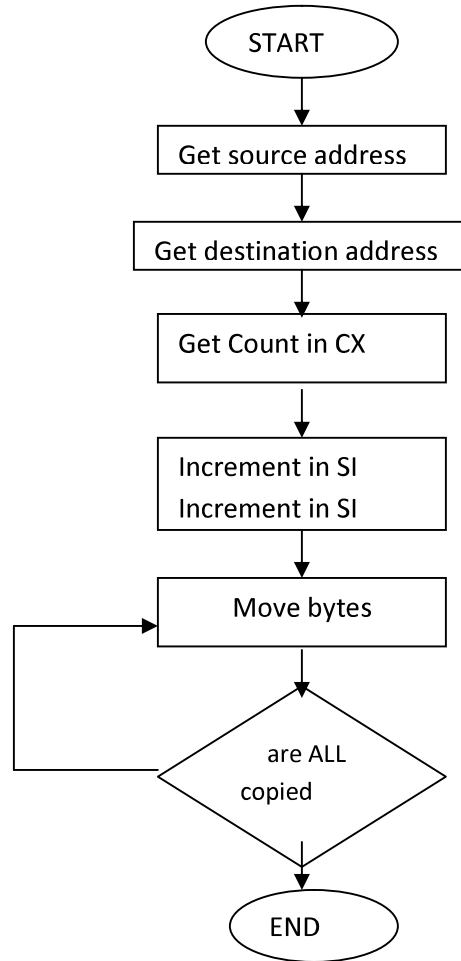
**AIM:-** Write a program to transfer a block of data with and without overlap.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
0500		FC	CLD		Clear direction flag DF
		BE,00,03	MOV	SI,0400	Source address in SI
		BF,02,02	MOV	DI,0600	Destination address in DI
		8B,0C	MOV	CX,[SI]	Count in CX
		46	INC	SI	Increment SI
		46	INC	SI	Increment SI
	BACK	A4	MOV	SB	Move byte
		E2,FD	LOOP	BACK	Jump to BACK until CX becomes zero
		CC	INT		Interrupt program

**FLOW CHART:**



**INPUT DATA**

0300 : 0B  
0301 : 00  
0302 : 03  
0303 : 04  
0304 : 05  
0305 : 06  
0306 : 15  
0307 : 07

0308 : 12  
0309 : 08  
030A : 09  
030B : 0A  
030C : 0B  
030D : 0E

**OUTPUT DATA**

0202 : 03  
0203 : 04  
0204 : 05  
0205 : 06  
0206 : 15  
0207 : 07  
0208 : 12  
0209 : 08  
020A : 09  
020B : 0A  
020C : 0B  
020D : 0E

**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.

- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Transfer of block of data numbers is performed.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

**LAB EXPERIMENT No. 8**

**AIM:-** Write a program to find average of two numbers.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Input Data	→	03	45	32	8A
Offset Address	→	500	501	502	503

Output Data	→	40	01
Offset Address	→	600	601

**Algorithm –**

1. Assign value 500 in SI and 600 in DI
2. Move the contents of [SI] in CL
3. Move 0000 in AX
4. Move the contents of CL to BL
5. Increment the value of SI by 1
6. Add the contents of AL and [SI]
7. Add 00 to AH with previous carry
8. Increment the value of SI by 1
9. Decrements the value of CL by 1
10. If Zero Flag (ZF) is not set go to step 6 else go to step 11
11. Divide the contents of AX by BL
12. Move the contents of AX in [DI]
13. Halt the program



**Program –**

Memory Address	Mnemonics	Comments
400	MOV SI, 500	SI <- 500
403	MOV DI, 600	DI <- 600
406	MOV AX, 0000	AX = 0000
409	MOV CL, [SI]	CL <- [SI]
40B	MOV BL, CL	BL <- CL
40D	INC SI	SI = SI + 1
40E	ADD AL, [SI]	AL = AL + [SI]
410	ADC AH, 00	AH = AH + 00 + cy
412	INC SI	SI = SI + 1
413	DEC CL	CL = CL – 1
415	JNZ 40E	JUMP if ZF = 0
417	DIV BL	AX = AX / BL
419	MOV [DI], AX	[DI] <- AX
41B	HLT	Stop

**Explanation –**

1. **MOV SI, 500** is used to move offset 500 to Starting Index (SI).
2. **MOV DI, 600** is used to move offset 600 to Destination Index (DI).
3. **MOV AX, 0000** is used to move data 0000 to AX.
4. **MOV CL, [SI]** is used to move the contents of [SI] to CL.
5. **MOV BL, CL** is used to copy contents of CL to BL.

6. **INC SI** is used to increment contents of SI by 1.
7. **ADD AL, [SI]** is used to add contents of [SI] to AL.
8. **ADC AH, 00** is used to 00 along with previous cy to AH.
9. **INC SI** is used to increment contents of SI by 1.
10. **DEC CL** is used to decrement contents of CL by 1.
11. **JNZ 40E** is used to jump to offset 40E if value of ZF = 0.
12. **DIV BL** is used to multiply contents of AX by BL.
13. **MOV [DI], AX** is used to move the contents of AX to [DI].
14. **HLT** stops executing the program and halts any further execution.

### **PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Average of data numbers is performed.

### **PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

### **Question / Answers**

Question 1: What is the average?

Answer: The average, also known as the arithmetic mean, is a measure of central tendency that represents the sum of a set of values divided by the number of values in the set.

Question 2: How is the average calculated?

Answer: The average is calculated by summing up all the values in a set and dividing the sum by the total number of values. Mathematically, it can be represented as:  $\text{Average} = \frac{\text{Sum of Values}}{\text{Number of Values}}$ .

**LAB EXPERIMENT No. 9**

**AIM:-** Write a program to check whether the data byte is odd or even.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

Assembler Directive/Mnemonic

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

MSG DB 10,13,'ENTER A NUMBER = \$'

MSG1 DB 10,13,'NUMBER IS EVEN \$'

MSG2 DB 10,13,'NUMBER IS ODD \$'

DATA ENDS

*CODE SEGMENT*

START:

MOV BX,DATA

MOV DS,BX

PRINT MACRO MESSAGE

LEA DX,MESSAGE

MOV AH,09H

INT 21H

ENDM

PRINT MSG

MOV AH,01H

INT 21H

SAR AL,01

JC ODD

PRINT MSG1

JMP TERMINATE

ODD:

PRINT MSG2

TERMINATE:

MOV AH,4CH

INT 21H

CODE ENDS

END START

### **PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Odd or even data numbers are checked.

### **PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

**Question / Answers**

Question: What is parity?

Answer: no.of 1's in data.

**LAB EXPERIMENT No. 10**

**AIM:-** Write a program to find maximum number in an array of 10 numbers.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

ADDRESS	OPCODE	MNEMONIC	COMMENTS
0400	BE 00 05	MOV SI, 0500	
0403	B9 10 00	MOV CX, 0010	
0406	B4 00	MOV AH, 00	
0408	3A 24	CMP AH, [SI]	AH-DATA OF [SI]
040A	73 02	JAE 040E	Jump Above or equal, Jump at CF=0
040C	8A 24	MOV AH, [SI]	
040E	46	INC SI	
040F	E0 F7	LOOPNE 0408	Decrement Counter, if not zero, continue the loop.
0411	88 24	MOV [SI], AH	Max. No. in 0510 adress.
0413	F4	HLT	

Result will be stored at 0510: largest no. ; no.'s are stored from 0500 to 050F.

**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.

- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** The largest numbers from array is found.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.



**LAB EXPERIMENT No. 11**

**AIM:-** Write a program to find the sum of the first ‘n’ integers.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A)

**THEORY/PROGRAM:-**

ADDRESS	Label	MNEMONIC	COMMENTS
0400		MOV SI, 0600	Initialize Pointer
		MOV CX, [SI]	Load Count for ‘n’ numbers stored from 0602 onwards each of 16-bit
	ABOVE	INC SI	
		INC SI	Increment Pointer to indicate 1 <sup>st</sup> no. in array of n-numbers
		MOV BL,00	Initialize BL to maintain carry during addition
		MOV AX,0000	Initialize AX for the following iteration
		ADD AX, [SI]	Add and update Sum
		JNC BELOW	Jump if no carry
		INC BL	Increment/Update Carry maintenance in BL
	BELOW	DEC CX	
		JNZ ABOVE	Repeat the iteration till ZF=1
		MOV [SI], AX	Store 16-bit LSB of sum of ‘n’ numbers after array
		INC SI	
		INC SI	
		MOV [SI], BL	Store 8-bit MSB of sum of ‘n’ numbers
		HLT	

**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using ‘Assemble-A’.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.

- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Sum of n' integer numbers is performed.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

**LAB EXPERIMENT No. 12**

**AIM:-** Write a program to generate a square wave.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A),

**THEORY/PROGRAM:-**

Assembler Directive/Mnemonic	Comments
CODE SEGMENT	
ASSUME CS: CODE, DS: CODE	
PROG PROC FAR	
ORG 400H ;	
Step-1	
MOV AL, 80H	;0400 B0 80
OUT 66H, AL	;0402 E6 66 for inbuilt
8255 change 66H to 76H	
;Step-2	
START: MOV AL, 55H	;0404 B0 55
;Step-3	
OUT 60H, AL	;0406 E6 60 for inbuilt
8255 change 60H to 70H	
;Step-4	
OUT 62H, AL	;0408 E6 62 for inbuilt
8255 change 62H to 72H	
;Step-5	
OUT 64H, AL	;040A E6 64 for inbuilt
8255 change 64H to 74H	
CALL DELAY1	;040C E8 0D 00
MOV AL, 0AAH	;040F B0 AA
;Step-6	
OUT 60H, AL	;0411 E6 60 for inbuilt

8255 change 60H to 70H

;Step-7

OUT 62H, AL ;0413 E6 62 for inbuilt

8255 change 62H to 72H

;Step-8

OUT 64H, AL ;0415 E6 64 for inbuilt

8255 change 64H to 74H

CALL DELAY1 ;0417 E8 02 00

;Step-9

JMP START ;041A EB E8

DELAY1: MOV CX, 0000H ;041C B9 0000

DL1: DEC CX ;041F 49

JNZ DL1 ;0420 75 FD

RET ;0422 C3

PROG ENDP

CODE ENDS

END

### **PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.
- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

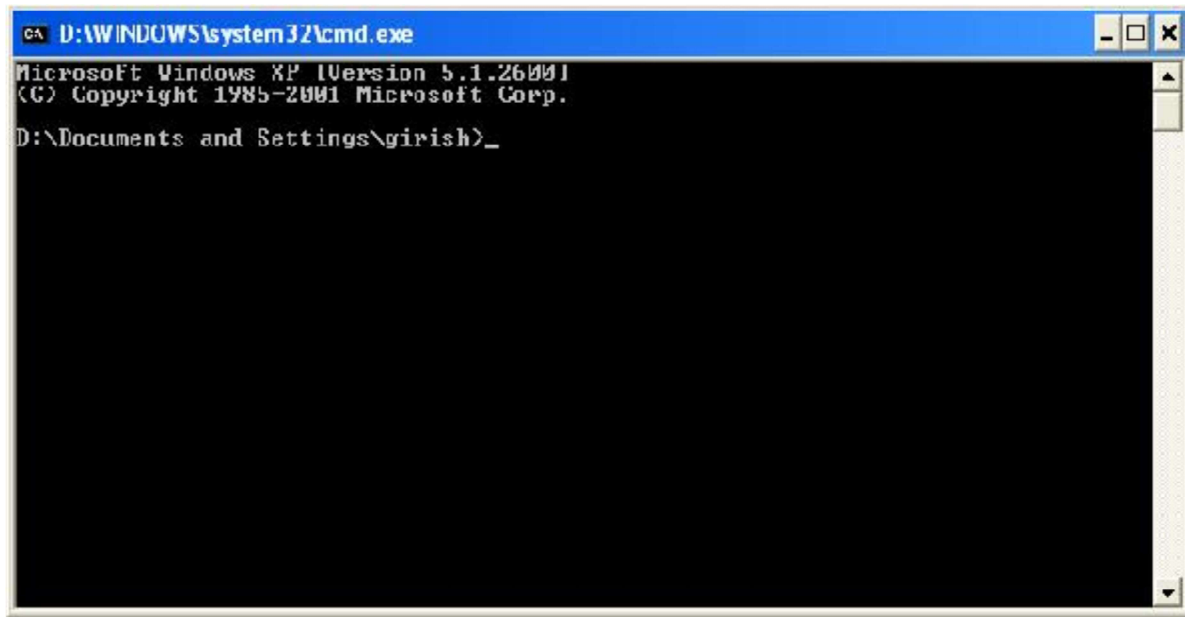
**Write any instructions with initial address Org 400H program**

**NOTE:** We can write above file in any window or dos editor directly. And change extension to .ASM

**Step2**>save above file suppose file name **ABC.ASM** in same folder where the assembler is saved.

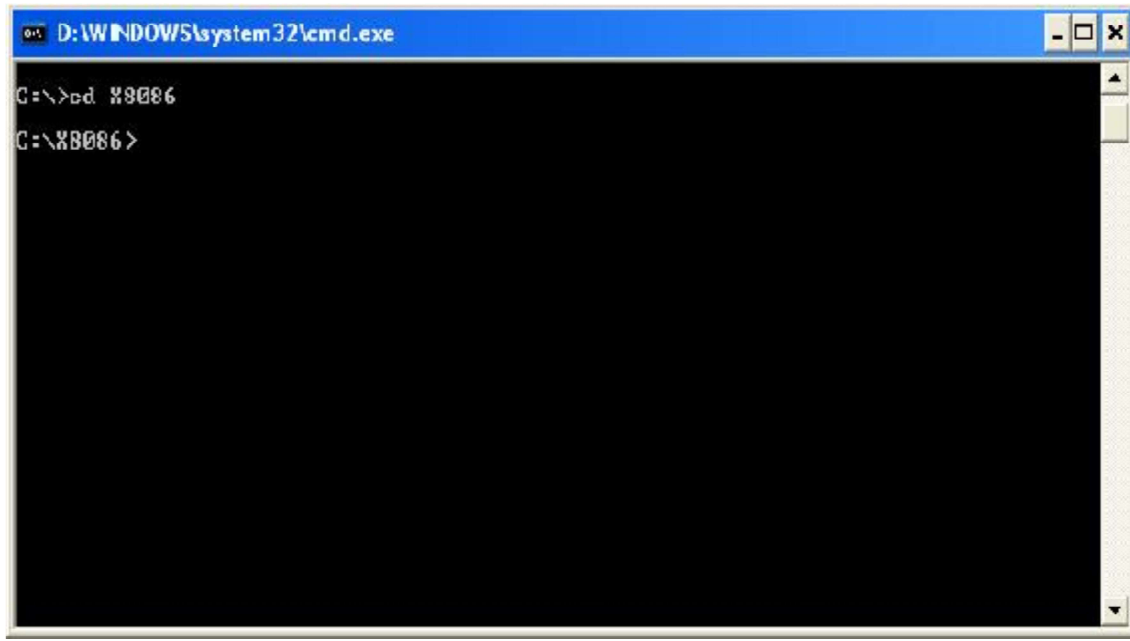
**Step3**>Now do the following steps

Open command prompt window by typing cmd in run command a window appears as shown below



```
GA D:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600.1
(C) Copyright 1985-2001 Microsoft Corp.
D:\Documents and Settings\girish>_
```

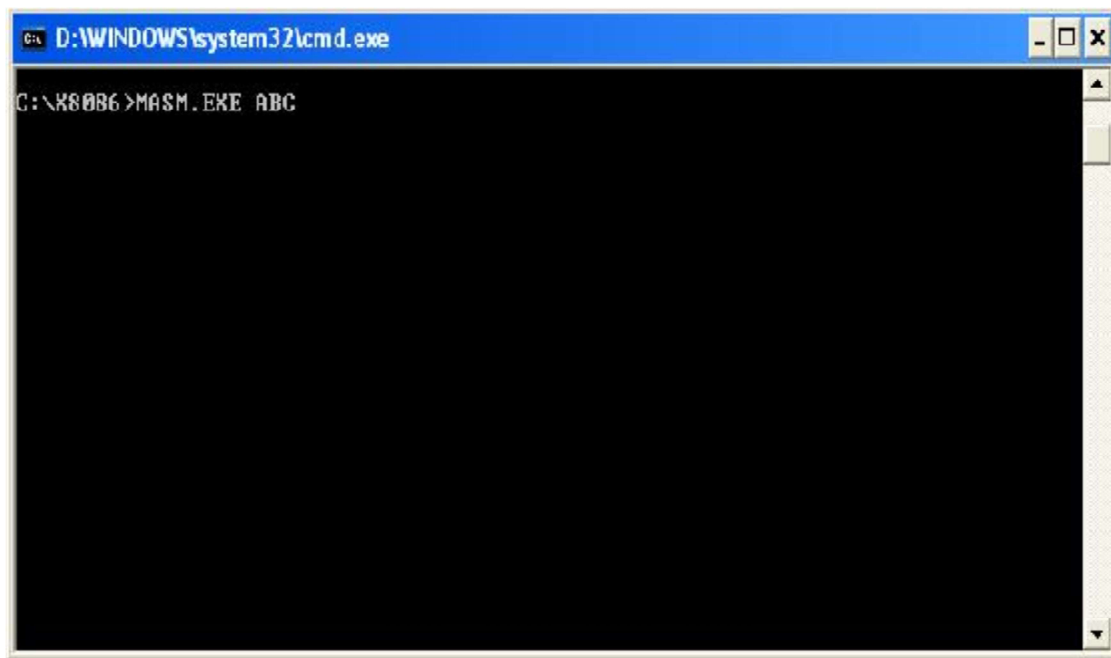
Now enter into the cross assembler directory for my system it is in C: drive Run MASM



A screenshot of a Windows command prompt window. The title bar reads "D:\WINDOWS\system32\cmd.exe". The command prompt shows the following sequence of commands and output:

```
C=>cd %8086
C:\%8086>
```

Run MASM file in below mention format



A screenshot of a Windows command prompt window. The title bar reads "D:\WINDOWS\system32\cmd.exe". The command prompt shows the following command and output:

```
C:\%8086>MASM.EXE ABC
```

Then press “ENTER” key of PC keyboard four times .The PC screen will display  
SUCCESSFUL 0 WARNING & 0 SERVE ERROR

**Note:** The ASM file should be in same folder where MASM assembler is placed

**Step4**>Run LINK file in below mention format

C :> LINK.exe ABC (No need to give any extension.)

Then press “ENTER” key of PC keyboard four times PC screen will display  
SUCCESSFUL 0 WARNING TO STACK SEGMENT.

**Step 5**>Run EXE2BIN file in below mention format.

C:>EXE2BIN.exe ABC (No need to give any extension)

Then ENTER key of PC keyboard once.

It will generate binary file of your program

**Step 6**>Now run Ascbin.exe in the below mention format

C:>Ascbin.exe

then ENTER(from PC keyboard)

This is a Program to convert ascii file to bin file & bin file to ascii file

After pressing ENTER it will show

Press <ESC> to exit program

Press <s> & <enter> to exit now

Press <c> & <Enter> to continue

Now Press C for continues.

**Step 7**> Window will open asking for

1) BIN to ASCII

2) ASCII to BIN

Select no 1 for BIN to ASCII then ENTER (from PC keyboard)

**Step 8**>Again window will open asking for

Enter the BIN filename?

Enter the filename with extension ABC.bin

**Enter the BIN filename? ABC.bin**

After pressing ENTER it will again prompt for ASCII filename

**Enter the ASCII filename? ABC.ASC**

**Note: In these fields extensions are necessary**

**Enter origin? 0000:0400** then ENTER (from PC keyboard)

Our ASCII file will be generated named **ABC.asc** down load this file into M8086—02

Connect HyperTerminal with kit then select transfer then send text file.

Our file will be down loaded into kit.

**RESULT:** Square wave is generated.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.



**LAB EXPERIMENT No. 13**

**AIM:-** Write a program to generate a triangular wave.

**APPARATUS REQUIRED:-** 8086 Microprocessor Kit (NV5586A), simulator

**THEORY/PROGRAM:-**

The following program will generate a Ramp (a triangular waveform) output at Xout and Yout.

**PROGRAM TO GIVE RAMP OUTPUT AT X-OUT & Y-OUT**

Address	Op-codes	Label	Mnemonics	Comments
0400	B0 80		MOV AL,80H	
0402	BA FE FF		MOV DX,0FFFEH	INITIALIZE 8255
0405	EE		OUT DX,AL	PA & PB AS OUTPUT
0406	B0 00		MOV AL,0H	SELECT A AS DATA 00
0408	BA F8 FF	START	MOV DX,0FFF8H	
040B	EE		OUT DX,AL	OUT AT X-OUT
040C	BA FA FF		MOV DX,0FFFAH	
040F	EE		OUT DX,AL	OUT AT Y-OUT
0410	FE C0		INC AL	INCREMENT A
0412	EB F4		JMP START	JUMP TO START

**PROCEDURE:-**

- i) Select the starting address for Code Segment and IP as per program using 'Assemble-A'.
- ii) Insert the assembly language program using NVIS 8086 Kit.
- iii) Note down the next IP address in Assembly Language Memory Address Column till HLT/INT instruction.
- iv) Using 'Unassemble-U' note down the relevant Hex codes for all instructions in the program into the Assembly Language Table.
- v) Insert the Input data by using 'Dump-D' command at concerned Data Segment and Offset address.
- vi) After completion of Input Data Filling and Assembly Program follow the procedure listed from step vii onwards.
- vii) Select Interrupt Flag-I option and fill the address where you want to stop the microprocessor to stop. You can stop the processor maximum 3 times in a program depending upon the inside addresses of the program.

- viii) Select Go command by G option and fill the starting address of CS:IP register combination and press Enter.
- ix) You can toggle between main menu and inside other options using F7 Key pressing.
- x) When program become executed, check the output data using 'Dump-D' and verify the result.

**RESULT:** Triangular wave is successfully generated.

**PRECAUTIONS:-**

1. Make sure that all the machine codes should be as per specified in the program.
2. Keep your workspace clean, tidy, and free from clutter to avoid accidents and confusion.

**Questions / Answers**

Question 1: How can you generate a triangular waveform using the 8086 microprocessor?

Answer: To generate a triangular waveform with the 8086 microprocessor, you can utilize the Digital-to-Analog Converter (DAC) and a counter. The counter is incremented or decremented repeatedly to produce a series of values that create a triangular waveform when passed through the DAC.

Question 2: How does the counter help in generating a triangular waveform?

Answer: The counter is used to generate a sequence of values that represent the ascending and descending portions of the triangular waveform. By incrementing the counter and converting its value to an analog voltage using the DAC, the ascending part of the waveform is generated. Then, by decrementing the counter and converting its value again, the descending part of the waveform is generated.

Question 3: How is the rate of increment or decrement determined for the counter?

Answer: The rate of increment or decrement for the counter determines the frequency or period of the triangular waveform. By controlling the speed at which the counter is incremented or decremented, you can adjust the frequency of the waveform. This can be achieved by configuring a timer or using specific delay loops in the code.

Question 4: What is the role of the DAC in triangular waveform generation?

Answer: The DAC converts the digital values produced by the counter into analog voltage levels. These analog voltage levels correspond to the values required for the triangular waveform. By feeding these analog voltages into an output device, such as a speaker or an oscilloscope, the triangular waveform can be observed and utilized for various applications.

### **Viva-Voce Questions**

1. Expand PPI?
2. Where do we prefer the serial communication?
3. What is the function of instruction pointer (IP) register?
4. What is the difference between IN and OUT instructions?
5. What is MODEM?
6. What sort of pipelining concept is available in 8086/8088  $\mu\text{p}$ ?
7. What is the purpose of segment registers in 8086/8088  $\mu\text{p}$ ?
8. How physical address is generated in case of 8086/8088  $\mu\text{p}$ ?
9. Explain segment overlapping. What is the use of LOCK' signal in 8086?
10. What do you know about relocatability of program?
11. Explain Division instruction of 8086  $\mu\text{p}$ ?
12. Define any three directives of 8086 assembler.
13. What is difference between LOOP, REP and JMP in  $\mu\text{p}$  8086?
14. What are the utility of conditional and control flags in  $\mu\text{p}$  8086 & name them in above category?
15. What is the difference between Real & Protected mode of  $\times 86$  family  $\mu\text{p}$ 's?
16. How  $\mu\text{p}$  8086 reads or write from or in to memory 2 bytes of data in one cycle & at what address it check from memory whether even or odd for this?

**SHORT Questions/ Answers:-**

Q.1 Explain MOV R1, R2 ?

A.1 Copy the content of register to register

Q.2 How many machine cycles and T-states are in MOV instruction?

A.2 One Machine cycle for Even address (16-bit operation), while two for Odd address.

Q.3 Which of the flag/s is/are affected in MOV instruction?

A.3 No Flags affected.

Q.4 What is XCHG ?

A.4 Swap the contents of the registers/Memory location data.

Q.6 Explain the addressing mode of XCHG?

A.6 Register Direct/Indirect modes are possible.

Q.7 What is ADD R1, R2 ?

A.7 Add the contents at operand register/memory operand and save result at destination operand.

Q.8 What is ADC R1, R2 and how it is different from ADD instruction?

A.8 Add the contents at operand register/memory operand with carry flag and save result at destination operand.

This lab manual has been updated by

Er. Monika Rani

([monika.rani@ggnindia.dronacharya.info](mailto:monika.rani@ggnindia.dronacharya.info))

**Cross checked by**

HOD ECE