**SCHEME : K**

Name : _____

Roll No. : _____ Year : 20___ 20___

Exam Seat No. : _____

# LABORATORY MANUAL FOR
# DIGITAL TECHNIQUES AND MICROPROCESSORS (313305)

**COMPUTER GROUP**

**MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI**
**(Autonomous) (ISO 9001: 2015) (ISO/IEC 27001:2013)**

## VISION

To ensure that the Diploma level Technical Education constantly matches the latest requirements of Technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

## MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the challenging technological & environmental challenges.

## QUALITY POLICY

We, at MSBTE, are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programmes.

## CORE VALUES

MSBTE believes in the following

- Skill development in line with industry requirements
- Industry readiness and improved employability of Diploma holders
- Synergistic relationship with industry
- Collective and Cooperative development of all stake holders
- Technological interventions in societal development
- Access to uniform quality technical education

A Laboratory manual for

# Digital Techniques and Microprocessor

## (313305)

## Semester – III

## (BD/IF/IH)

## Maharashtra State Board of Technical Education, Mumbai

**(Autonomous) (ISO 9001:2015) (ISO/IEC 27001:2013)**

**Maharashtra State Board of Technical Education, Mumbai**

**(Autonomous) (ISO 9001:2015) (ISO/IEC 27001:2013)**

**4th Floor, Government Polytechnic Building,**
**49, Kherwadi, Bandra (East), Mumbai- 400051.**
**(Printed on July 2024)**

# MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI

## Certificate

This is to certify that Mr./Ms. …… Roll No. ……. of Third Semester of Diploma in ……..of Institute …… …… (Code: ……) has completed the term work satisfactorily in course **Digital Techniques and Microprocessors (313305)** for the academic year 20……to 20…….. as prescribed in the curriculum.

**Place: ……..** **Enrollment No. :…….**

**Date: ……** **Exam Seat No.: ……**

**Subject Teacher** **Head of department** **Principal**

Seal of Institute

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI

# Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative ‘K’ Scheme curricula for engineering diploma programmes with outcome-based education as the focus and accordingly, a relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work, making each teacher, instructor and student realize that every minute of the laboratory time needs to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome- based curriculum, every practical course has been designed to serve as a ***'vehicle'*** to develop this industry identified competency in every student. The practical skills are difficult to develop through "chalk and duster" activity in the classroom situation. Accordingly, the "K” scheme laboratory manual development team designed the practical to focus on the outcomes, rather than the traditional age old practice of conducting practical to 'verify the theory" (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the predetermined outcomes. It is expected. from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through the procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student- centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

The basic aim of this course is that, the student must learn the basic concepts, rules and laws of electric and magnetic circuits and practical thereof. The basic concepts of electrical engineering in this course will be very useful for understanding electrical circuits.

Although best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome.

## Program Outcomes (POs) to be achieved through Practicals of this Course

Following programme outcomes are expected to be achieved through the practical of the course.

**PO 1. Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, sciences and engineering fundamentals and engineering specialization to solve the engineering problems.

**PO 2. Problem analysis:** Identify and analyze well-defined engineering problems using codified standard methods.

**PO 3. Design/ development of solutions:** Design solutions for well-defined technical problems and assist with the design of system components or processes to meet specified needs.

**PO 4. Engineering tools:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

**PO 5. Engineering practices for Society, Sustainability and Environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

**PO 6. Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

**PO 7. Life-Long Learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

# List of Industry Relevant Skills

The following industry relevant skills of the identified of competency “Test digital systems by applying principles of digital techniques and microprocessors." are expected to be developed in the student by undertaking the laboratory work as given in laboratory manual

1. Analyze problem definition.
2. Build digital circuits.
3. Develop assembly programs for Real life applications.
4. Ability to solve application-level problems.

## Practical-Course outcome matrix

**COURSE LEVEL LEARNING OUTCOMES (COS)**

CO1 - Test logic gates and digital systems.

CO2 - Use basic combinational and sequential logic circuits employing digital ICs.

CO3 - Perform operations on registers using 8086 instructions.

CO4 - Use 8086 microprocessor environment to build and execute assembly language programs.

CO5 - Develop assembly language programming in 8086 to implement loops and branching instructions.

| Sr. No. | Title of the Practical | CO1 | CO2 | CO3 | CO4 | CO5 |
|---|---|---|---|---|---|---|
| 1 | * Verification of truth table of basic logic gates, special logic gates and Identify various Logic gate ICs. | ✓ | | | | |
| 2 | Implementation and verification of expression using universal logic gate ICs | ✓ | | | | |
| 3 | Verification of De-Morgan's theorems using basic logic gates | | ✓ | | | |
| 4 | * Conversion of expression to Sum-of- Product (SOP) and Product-of-Sum (POS) | | ✓ | | | |
| 5 | * Implement Multiplexer and Demultiplexer logic (The practical may be performed using virtual lab) | | ✓ | | | |
| 6 | Implementation of Latch | | ✓ | | | |
| 7 | * Verification of contents of general purpose, segment registers, flags and memory locations of different segments during execution of the program | | | ✓ | | |
| 8 | * Assembly language programming for addition and subtraction for hexadecimal numbers | | | | ✓ | |
| 9 | Apply assembly language programming logic for addition, subtraction and multiplication for BCD numbers. | | | | ✓ | |
| 10 | * Assembly language programming for multiplication and division | | | | ✓ | |
| 11 | Assembly language programming to find smallest /largest hexadecimal numbers | | | | ✓ | |
| 12 | * Assembly language programming for sorting of data | | | | | ✓ |
| 13 | Assembly language programming for transfer of block of data | | | | | ✓ |
| 14 | Count the occurrence of a given number from a block of data | | | | | ✓ |
| 15 | * Implement shift and rotate instructions on given data | | | | | ✓ |

## Guidelines to Teachers

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain prior concepts to the students before starting of each practical.
3. Involve students in the performance of each experiment.
4. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
5. Teachers should give opportunities to students for hands-on experience after the demonstration.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected of the students by the industry.
8. Finally give practical assignments and assess the performance of students based on tasks assigned to check whether it is as per the instructions.
9. Teacher is expected to refer complete curriculum document and follow guidelines for implementation
10. At the beginning of the practical, which is based on the simulation, teacher should make the students acquainted with any simulation software environment.

## Instructions for Students

1. Listen carefully to the lecture given by the teacher about the subject, curriculum, learning structure, skills to be developed.
2. Organize the work in the group and record all programs.
3. Student shall develop maintenance skills as expected by industries.
4. Student shall attempt to develop related hand-on skills and gain confidence.
5. Student shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual
6. Student shall refer to technical magazines.
7. Student should develop the habit of submitting the practicals on date and time.
8. Student should prepare well while submitting a write-up of exercise.
9. Attach/paste separate papers wherever necessary.

## Content Page

### List of Practical and Progressive Assessment Sheet

| Sr. No. | Title of the Practical | Page no. | Date of Performance | Date of Submission | Assessment Marks (25) | Dated sign. of Teacher | Remarks (If any) |
|---|---|---|---|---|---|---|---|
| 1 | * Verification of truth table of basic logic gates, special logic gates and Identify various Logic gate ICs. | | | | | | |
| 2 | Implementation and verification of expression using universal logic gate ICs | | | | | | |
| 3 | Verification of De-Morgan's theorems using basic logic gates | | | | | | |
| 4 | * Conversion of expression to Sum-of- Product (SOP) and Product-of-Sum (POS) | | | | | | |
| 5 | * Implement Multiplexer and Demultiplexer logic (The practical may be performed using virtual lab) | | | | | | |
| 6 | Implementation of Latch | | | | | | |
| 7 | * Verification of contents of general purpose, segment registers, flags and memory locations of different segments during execution of the program | | | | | | |
| 8 | * Assembly language programming for addition and subtraction for hexadecimal numbers | | | | | | |
| 9 | Apply assembly language programming logic for addition, subtraction and multiplication for BCD numbers. | | | | | | |
| 10 | * Assembly language programming for | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | multiplication and division | | | | | | |
| 11 | Assembly language programming to find smallest /largest hexadecimal numbers | | | | | | |
| 12 | * Assembly language Programming for sorting of data | | | | | | |
| 13 | Assembly language programming for transfer of block of data | | | | | | |
| 14 | Count the occurrence of a given number from a block of data | | | | | | |
| 15 | * Implement shift and rotate instructions on given data | | | | | | |
| **Total** | | | | | | | |

**Note: Out of above suggestive LLOs –**

- '*' Marked Practicals (LLOs) Are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

## Practical No.1: Verification of truth table of basic logic gates, special logic Gates and Identify various Logic gate ICs.

### I. Practical Significance

Logic gates are the basic building blocks of complex logic circuits and all types of digital systems. Logic gates are used in all digital circuits such as switches, memories, microprocessors and embedded systems. Knowledge of functions of logic gates will help the students to build the digital circuits and the implementation of logic circuits with a minimum number of logic gates.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve and demonstrate the following COs on completion of course based learning

Test logic gates and digital systems.

### IV. Laboratory Learning Outcome(s)

Identify various logic gate ICs.

Verify truth tables of basic logic gates (AND-7408, OR- 7432, NOT-7404) using breadboard

Verify truth tables of universal gates (NAND-7400, NOR-7402).

Verify truth tables of special logic gates EX-OR-7486, EX-NOR-74266

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI. Relevant Theoretical Background

A logic gate is an electronic circuit which makes logical decisions. It has only one output and one or more inputs. In digital logic design only two voltage levels or states are allowed and these states are generally referred to as logic “1” or High represented by +5V and logic “0” or Low represented by 0V.

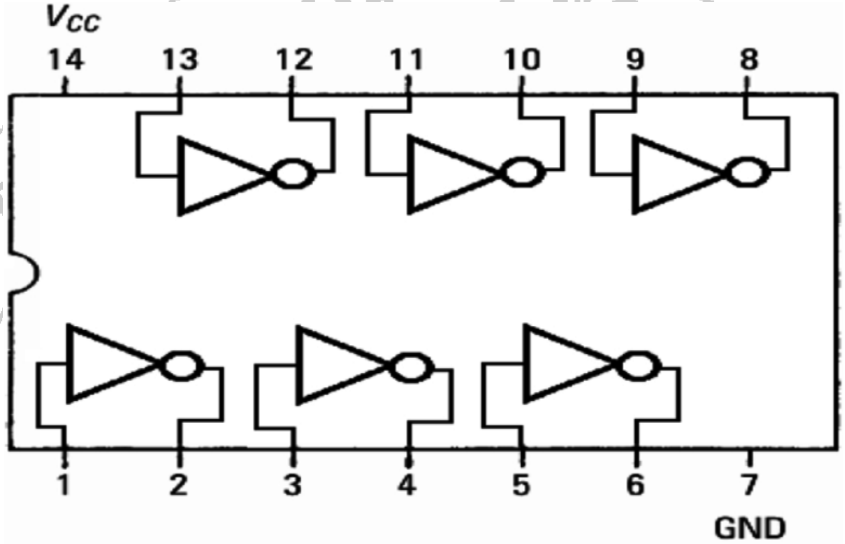
Digital systems are said to be constructed by using logic gates like AND, OR, NOT and EX-OR gates. These gates are verified using Truth Tables which help to understand the behavior of logic gates.

**Classification of Logic gates**

**Logic Gates**

| Basic Gates | Universal Gates | Special Purpose Gates |
|---|---|---|
| NOT, AND & OR Gate | NAND & NOR Gate | EX-OR & EX-NOR Gate |

## VII. Circuit diagram/Layout of Laboratory




$Y=\overline{A}$

| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

**Fig 1.1 NOT Gate IC 7404 and its Truth Table**

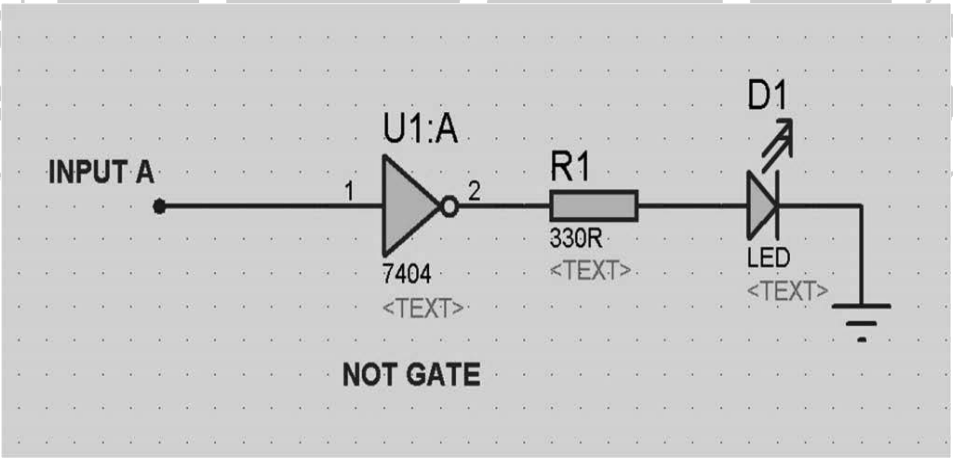



**Fig 1.2 NOT Gate Connection**

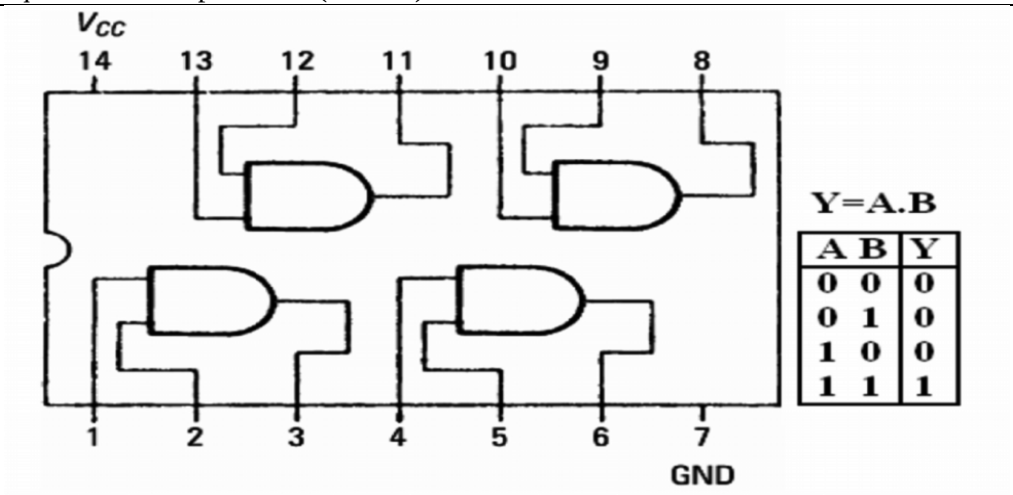



**Fig 1.3 AND Gate IC 7408 and its Truth Table**

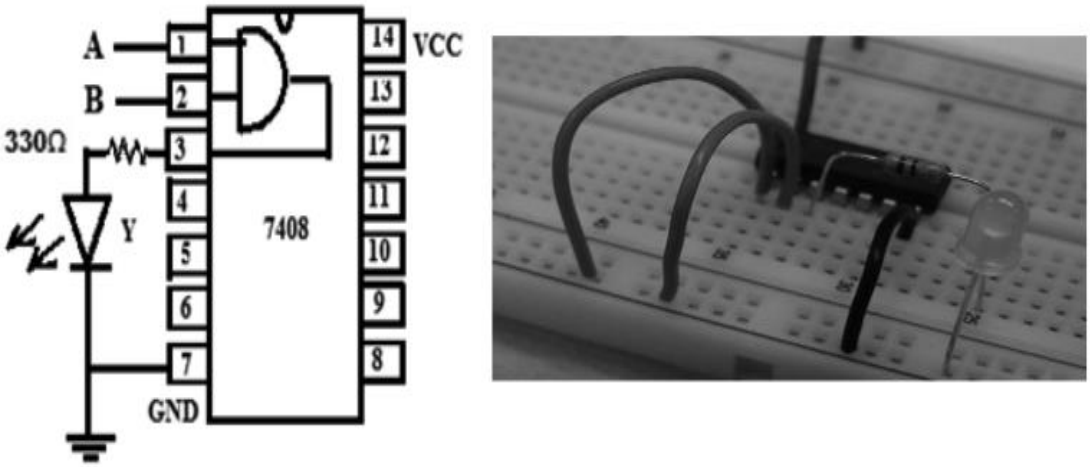



**Fig 1.4 AND Gate IC 7408 Sample Circuit**

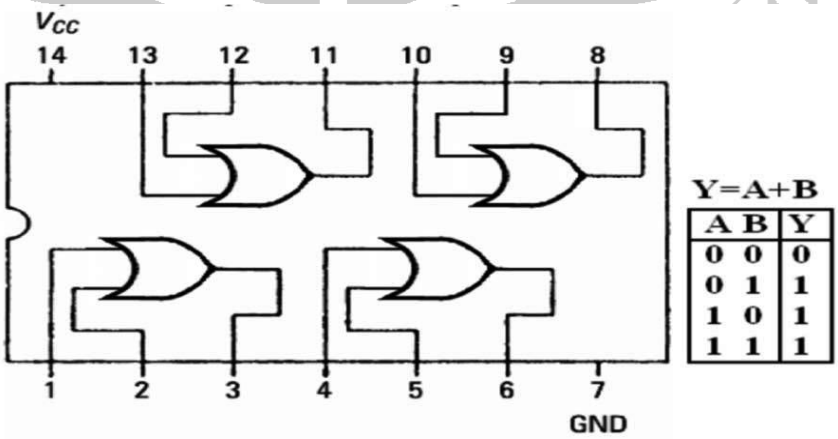
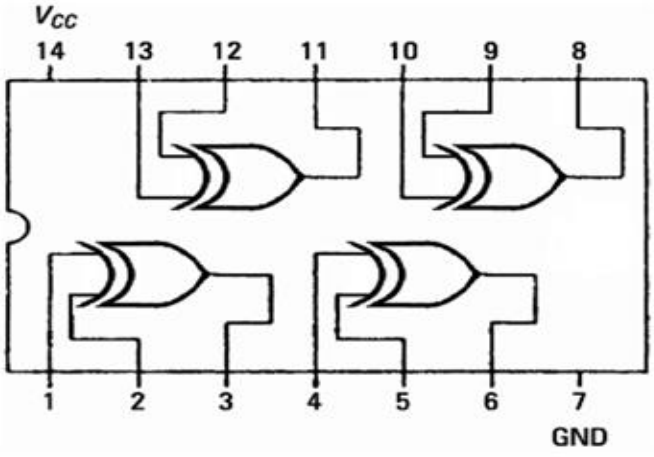


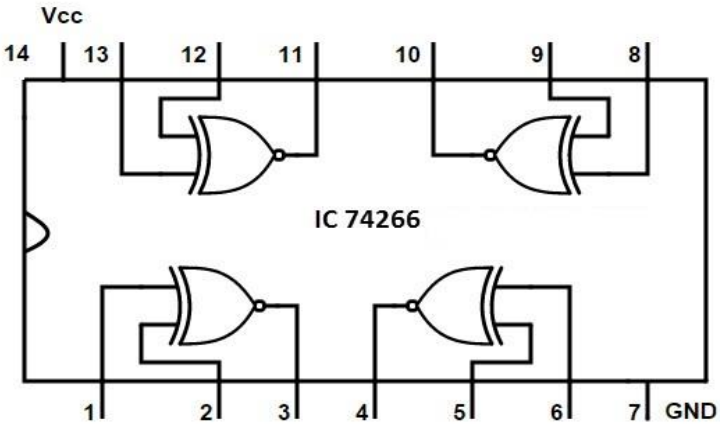

**Fig 1.5 OR Gate IC 7432 and its Truth Table**




Y=A⊕B

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Fig 1.6 EX- OR Gate IC 7486 and its Truth Table**




| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Boolean Expression

output $Y = \overline{A \oplus B}$

**Fig 1.7 EX- NOR Gate IC 74266 and its Truth Table**

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Digital Multimeter | 3 and 1/2 digit | **1** |
| **2** | DC Regulated Power Supply | 2 x 0-30 V; 0-2 Automatic Overload (Current Protection) Constant Voltage and Constant Current Operation Digital Display for Voltage and Current Adjustable Current Limiter Excellent Line and Load Regulation | **1** |
| **3** | Basic logic gates (AND-7408, OR- 7432, NOT- 7404) EX-OR- 7486, EX-NOR-74266 | -- | **1 Each** |
| **4** | Bread board | 5.5 cm X 17 cm | **1** |
| **5** | Connecting wires | Single strand 0.6mm Teflon coating | **As required** |

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **6** | IC Tester | Digital IC Tester | **1** |
| **7** | LEDs | Red/Green/Yellow 5mm | **5** |
| **8** | Resistors | 330 Ω /0.25 W | **1** |
| **9** | Stripper | -- | **1** |
| **10** | Digital IC's Data sheets of ICs used in Lab | | |

## IX. Precautions to be followed

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply (0-5 V/ 500mA)

## X. Procedure

1. Make the connections as per the circuit diagram of logic gates and give the supply voltage to relevant pin.
2. Connect the inputs from the source to logic gates as per the logic levels.
3. Observe the output on LED for each combination of input as per truth table.
4. Verify the truth table.
5. Repeat the process for all other logic gates.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……...

…….…

..……..…

……..

……...

…….…

## XIII. Observations and Calculations

| Inputs | | 7404(NOT) | | 7408(AND) | | 7432(OR) | | 7486(EX-OR) | | 74266(EX-NOR) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | LED Status ON/OFF | Output Voltage | LED Status ON/OFF | Output Voltage | LED Status ON/OFF | Output Voltage | LED Status ON/OFF | Output Voltage | LED Status ON/OFF | Output Voltage |
| **0(0V)** | **0(0V)** | | | | | | | | | | |
| **0(0V)** | **1(5V)** | | | | | | | | | | |
| **1(5V)** | **0(0V)** | | | | | | | | | | |
| **1(5V)** | **1(5V)** | | | | | | | | | | |

## XIV. Result(s)

……...

..……..…

..……..…

## XV. Interpretation of results

……..

.....……

……..

## XVI. Conclusion and recommendation

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.**

1. Write down voltage at logic level 0 and 1.
2. List the functions of pin 7 and 14 in IC 7432.
3. What will happen if pin number 14 is connected to ground and pin number 7 is connected to VCC?
4. List the number of NOT gates available in IC 7404.
5. Write the name of manufacturers of Digital IC used in your Practical Lab.
6. State the need for a resistor connected in series with LED. Write its value.

**[Space for Answers]**

## XVIII. References / Suggestions for further reading


1. https://academo.org/demos/logic-gate-simulator
2. https://www.youtube.com/watch?v=AT_GjUjNFpo
3. https://www.youtube.com/watch?v=EBlgoycFNJ8
4. https://www.youtube.com/watch?v=WGYEpZQnRE8
5. http://www.ti.com/lit/ds/symlink/sn74ls00.pdf


## XIX. Assessment Scheme

| | **Performance Indicators** | **Weightage** |
|---|---|---|
| | ProcessRelated:15 Marks | 60 % |
| 1 | Handling of the components | 10% |
| 2 | Identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | Working in teams | 10% |
| | **ProductRelated:10 Marks** | **40%** |
| 5 | Calculated the or ethical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| | **Total(25 Marks)** | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

**Practical No.2: Implementation and verification of expression using universal logic gate ICs**

## I. Practical Significance

NAND and NOR universal gates that can implement basic gates and any Boolean function. Any basic gate AND,OR and NOT gates can be implemented using universal gates which means any digital circuit can be implemented using universal gates.

## II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

## III. Course Level Learning Outcome(s)

Test logic gates and digital systems.

## IV. Laboratory Learning Outcome(s)

Design a circuit for a given logical expression using the universal gates (NAND)
Design a circuit for a given logical expression using the universal gates (NOR).

## V. Relevant Affective Domain related outcome(s)

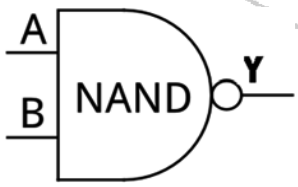
1. Handle IC and equipment carefully.
2. Follow safe practices

## VI. Relevant Theoretical Background

NOR and NAND gates have the property that they individually can be used to hardware-implement a logic circuit corresponding to any given Boolean expression. That is, it is possible to use either only NAND gates or only NOR gates to implement any Boolean expression. The Basic gates AND, OR, NOT can be realized from it. The NAND gate is AND gate succeeded by NOT gate. The NOR gate is OR gate succeeded by NOT gate.

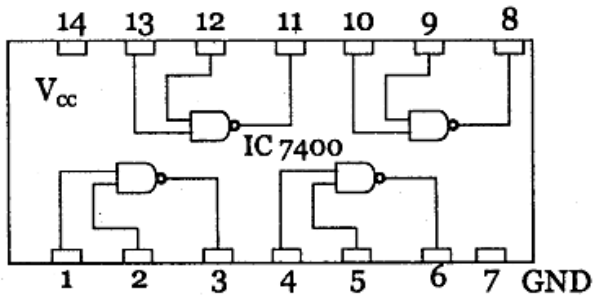
## VII. Circuit diagram

**Layout of Laboratory**




| Input | | Output |
|---|---|---|
| A | B | Y= $\overline{A.B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Figure 2.1 Symbol and Truth Table of NAND gate**



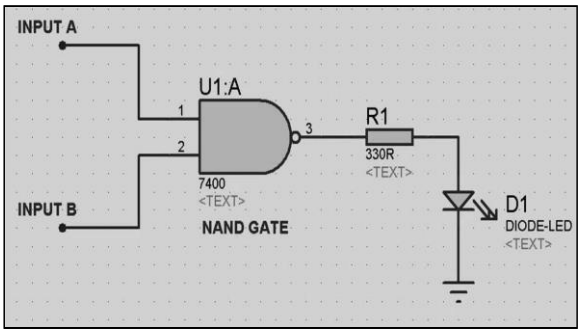
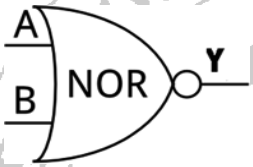
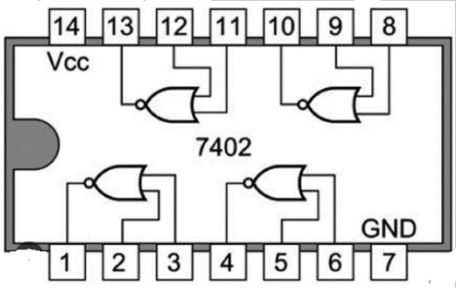




**Figure No 2.2 NAND gate using IC 7400**




| Input | | Output |
|---|---|---|
| A | B | $\overline{A+B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Figure 2.3 Symbol and Truth Table of NOR gate**



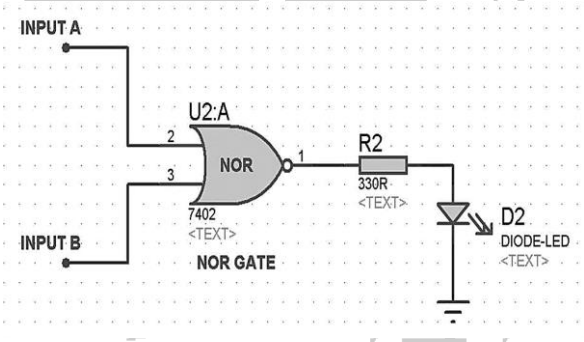




**Figure No 2.4: NOR gate using IC 7402**

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Digital Multimeter | Digital Multimeter: 3 1/2 digit display. | 2 |
| **2** | Breadboard | General Purpose Breadboard | 1 |
| **3** | DC power supply | +5 V/500mA Regulated power supply | 1 |
| **4** | IC | 7400, 7402 | 1 Each |
| **5** | LED, Resistors | Red color 5 mm ,330Ohms | 1 Each |
| **6** | Connecting wires | Single strand 0.6 mm Teflon coating | LS |
| **7** | Digital IC tester | Tests a wide range of Digital IC's such as 74 Series, 40/45 Series of CMOS Ic's | 1 |

## IX. Precautions to be followed

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply

## X. Procedure

1. Make the connection as per circuit diagram and give supply voltage to relevant pin
2. Connect the inputs from source to logic gates as per logic level.
3. Observe the output on LED for each combination of input as per truth table.
4. Verify the truth table.
5. Repeat the process for other universal logic gate.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure

……

……

……

……

## XIII. Observations and Calculations (use blank sheet provided if space not sufficient)

| Inputs | | 7400(NAND) | | 7402(NOR) | |
|---|---|---|---|---|---|
| A | B | LED Status (ON/OFF) | Output voltage | LED Status (ON/OFF) | Output voltage |
| 0(0V) | 0(0V) | | | | |
| 0(0V) | 1(5V) | | | | |
| 1(5V) | 0(0V) | | | | |
| 1(5V) | 1(5V) | | | | |

## XIV. Result(s)

……

……

……

……

## XV. Interpretation of results

……

……

……

## XVI. Conclusion and recommendation

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Construct basic gates NOT, AND, OR using NAND gate. Write necessary outputs.
2. List number of NOR gates used in IC 7402.
3. Construct basic gates using NOR gates only.
4. How many NAND gates are there in a single 7400 NAND gate IC?
5. Design a logical circuit for the expression Y=AB+CD

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References/Suggestions for further reading


1. https://youtu.be/TQ1DgsdUe5A?feature=shared
2. https://youtu.be/uUOIV4DqFHc?feature=shared

## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No.3: Verification of De-Morgan's theorems using basic logic gates

### I. Practical Significance

Logic gates are the basic units used to implement complex logic circuits that are constructed using various combinations of gates known as combinational logic circuits. It requires the use of two or more gates to form useful complex functions. These functions begin with Boolean equations. De Morgan's Theorem is used to simplify Boolean expressions and hence in turn the complex logic circuits.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve & demonstrate the following COs on completion of course based learning
Use basic combinational and sequential logic circuits employing digital ICs.

### IV. Laboratory Learning Outcome(s)

Verify the truth table of De-Morgan's first theorem using basic logic gates
Verify the truth table of De-Morgan's second theorem using basic logic gates.

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

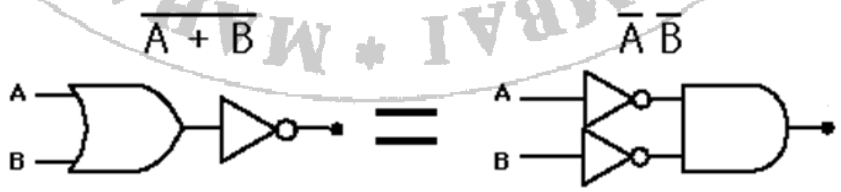
### VI. Relevant Theoretical Background

De Morgan's Theorem is used to simplify Boolean expressions and Digital circuits.

**De-Morgan's First theorem**

De-Morgan's First theorem states that for any two elements A and B in a Boolean algebra the complement of the Sum is equal to Product of complements.
In other words a NOR gate is equivalent to a bubbled AND gate.

**NOR gate = Bubbled AND gate**




**Logically it is given by expression-**

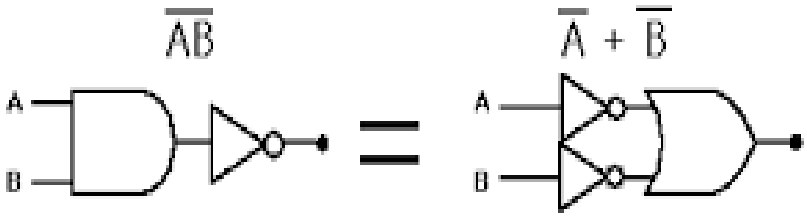
$$\overline{A+B} = \overline{A}.\overline{B}$$

**De-Morgan's Second theorem**

De-Morgan's Second theorem states that for any two elements A and B in a Boolean algebra the complement of the Product is equal to Sum of complements.

In other words a NAND gate is equivalent to a bubbled OR gate.

**NAND gate = Bubbled OR gate**




**Logically it is given by expression-**

$$\overline{AB} = \overline{A} + \overline{B}$$

## VII. Circuit diagram/Layout of Laboratory

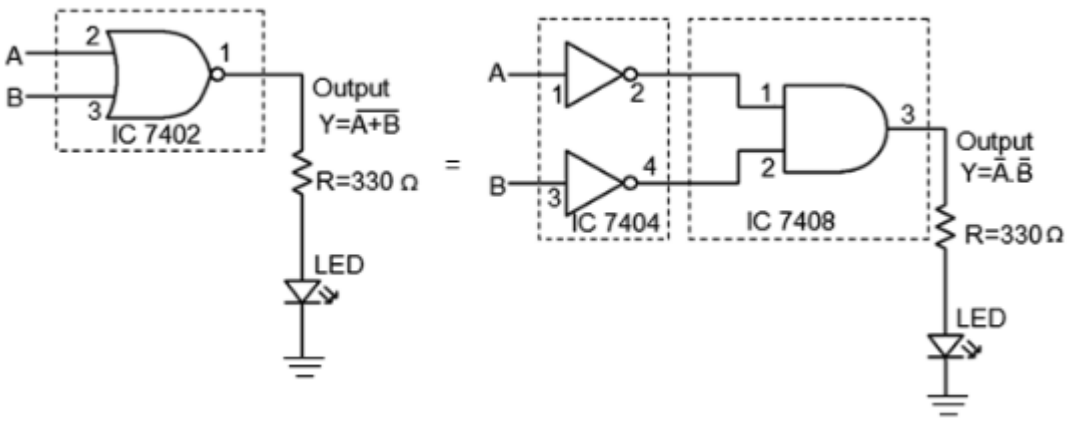



**Fig 3.1 De-Morgan's First Theorem**

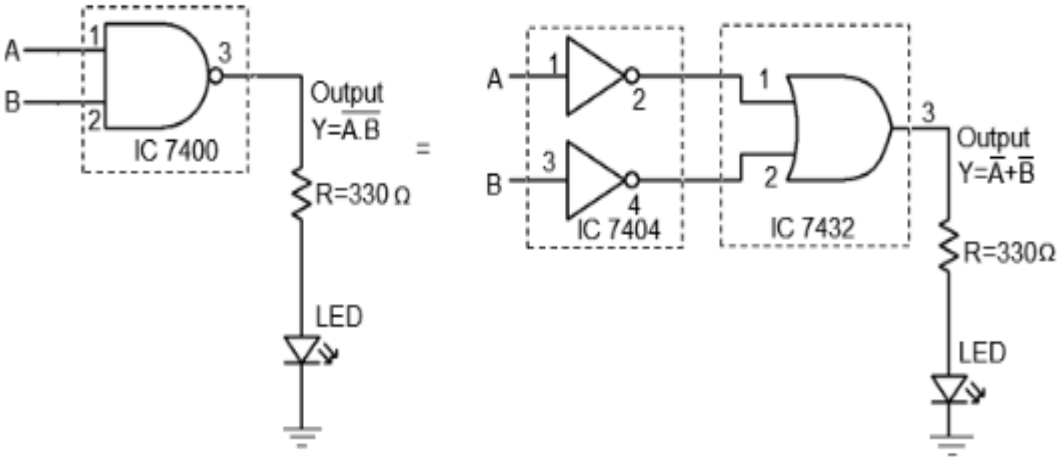



**Fig 3.2 De-Morgan's Second Theorem**

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Digital Multimeter | 3 and 1/2 digit | 1 |
| 2 | DC Regulated Power Supply | 2 x 0-30 V; 0-2 A Automatic Overload (Current Protection) Constant Voltage and Constant Current Operation Digital Display for Voltage and Current Adjustable Current Limiter Excellent Line and Load Regulation | 1 |
| 3 | Basic logic gates (NOT- 7404, AND-7408, OR- 7432) Universal gates (NAND-7400, NOR-7402) | -- | 1 Each |
| 4 | Bread board | 5.5 cm X 17 cm | 1 |
| 5 | Connecting wires | Single strand 0.6mm Teflon coating | As required |
| 6 | IC Tester | Digital IC Tester | 1 |
| 7 | LEDs | Red/Green/Yellow 5mm | 5 |
| 8 | Resistors | 330 Ω /0.25 W | 1 |
| 9 | Stripper | -- | 1 |
| 10 | Digital IC's Data sheets of ICs used in Lab | | |

## IX. Precautions to be followed

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply (0-5 V/ 500mA)

## X. Procedure

1. Make the connections as per the circuit diagram of De-Morgan's First theorem (Fig 3.1) and give the supply voltage to the relevant pin.
2. Connect the inputs from the source to logic gates as per the logic levels.
3. Observe the output on LED for each combination of input as per truth table.
4. Verify the truth table.
5. Repeat the process for De-Morgan's second theorem. (Fig. 3.2)

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……

……

……

……

……

……

## XIII. Observations and Calculations

**De-Morgan's First theorem Observation:**

| Input | | Outputs | |
|---|---|---|---|
| A | B | LHS= $\overline{A+B}$ | RHS =$\overline{A}.\overline{B}$ |
| **0** | **0** | | |
| **0** | **1** | | |
| **1** | **0** | | |
| **1** | **1** | | |

**De-Morgan's Second theorem Observation:**

| Input | | Outputs | |
|---|---|---|---|
| A | B | LHS = $\overline{A.B}$ | RHS = $\overline{A}+\overline{B}$ |
| **0** | **0** | | |
| **0** | **1** | | |
| **1** | **0** | | |
| **1** | **1** | | |

**XIV. Result(s)**

……..……..……..……

..……..……..……

..……

**XV. Interpretation of results**

……..……

.....……..……..……

……

**XVI. Conclusion and recommendation**

……

……..……

……..……

**XVII. Practical related questions**

**Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.**

1. Name the ICs required for De-Morgan's First Theorem
2. Name the ICs required for De-Morgan's Second Theorem
3. Realize Ex-OR gate using NAND gate.
4. Realize Ex-NOR gate using NOR gate
5. Draw 3 input NOR gate using 2 input NOR gate.
6. Simplify the following equation using De-Morgan's Theorem

$$Y = \overline{\overline{AB} + \bar{A} + AB}$$

**[Space for Answers]**

...……..……

……

...……..……

……

...……..……

……

...……..……

……

...……..……

## XVIII. References & suggestions for further reading


1. https://www.allaboutcircuits.com/textbook/digital/chpt-7/demorgans-theorems/
2. https://www.youtube.com/watch?v=W7YTfLaPWRY
3. http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/DeMorgan.html


## XIX. Assessment Scheme

| **Performance Indicators** | | **Weightage** |
|---|---|---|
| Process Related: 15 Marks | | **60 %** |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total (25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No.4: * Conversion of expression to Sum-of-Product (SOP) and Product-of-Sum (POS)

### I. Practical Significance

The standard forms of Boolean functions help the logic circuit designer by simplifying the derivation of the function to be implemented. The goal of logic expression minimization is to find an equivalent of an original logic expression that has fewer variables per term, has fewer terms and needs less logic to implement. The minimization will result in reduction of the number of gates (resulting from less number of terms) and the number of inputs per gate (resulting from less number of variables per term) the minimization will reduce cost, efficiency and power consumption.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Use basic combinational and sequential logic circuits employing digital ICs.

### IV. Laboratory Learning Outcome(s)

Design and test the circuit for converting expression into Sum-of- Product (SOP)
Design and test the circuit for converting expression into Product-of-Sum (POS).

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and equipment carefully.
2. Follow safe practices

### VI. Relevant Theoretical Background

SOP stands for Sum of Product which is sum of minterms or number of 1s whereas POS stands for Product of Sum which is product of maxterms or number of 0s. These provide a systematic way of expressing and simplifying logical expressions. SOP form combines multiple OR operations applied to AND ed terms, whereas POS form involves AND operations applied to ORed terms. Examples of SOP and POS equations are as below-

SOP Equation-
$Y = AB + BC + AC$
Can be implemented by using AND-OR Logic

POS Equation-
$Y =(A+B).(B+C).(A+C)$
Can be implemented by using OR-AND Logic

**Table 4.1 Expressions for minterm and maxterm.**

| Variables | | | Min terms | Max terms |
|---|---|---|---|---|
| A | B | C | $m_i$ | $M_i$ |
| 0 | 0 | 0 | A' B' C' = m 0 | A + B + C = M 0 |
| 0 | 0 | 1 | A' B' C = m 1 | A + B + C' = M 1 |
| 0 | 1 | 0 | A' B C' = m 2 | A + B' + C = M 2 |
| 0 | 1 | 1 | A' B C = m 3 | A + B' + C' = M 3 |
| 1 | 0 | 0 | A B' C' = m 4 | A' + B + C = M 4 |
| 1 | 0 | 1 | A B' C = m 5 | A' + B + C' = M 5 |
| 1 | 1 | 0 | A B C' = m 6 | A' + B' + C = M 6 |
| 1 | 1 | 1 | A B C = m 7 | A' + B' + C' = M 7 |

## VII. Truth Table and Circuit diagram Layout of Laboratory

**Table 4.2 Truth table of a sample digital circuit**

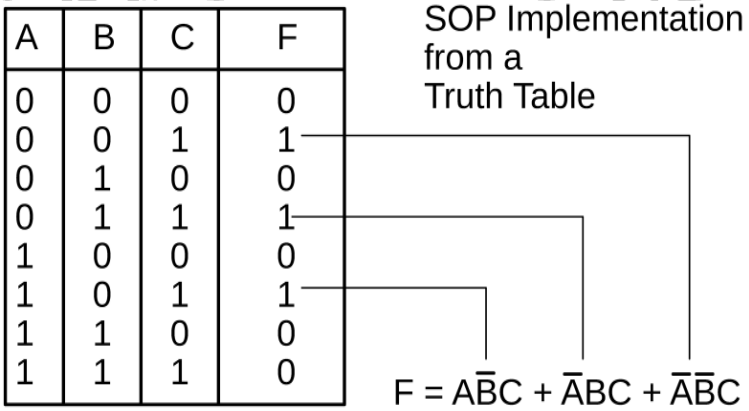
| Inputs | | | Output |
|---|---|---|---|
| **A** | **B** | **C** | |
| **0** | **0** | **0** | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

For SOP expression by referring to table no 4.2

$$f = m1 + m3 + m5$$

$$f = \sum m\ (1,3,5)$$

**Table No 4.3 SOP implementation from a truth table**



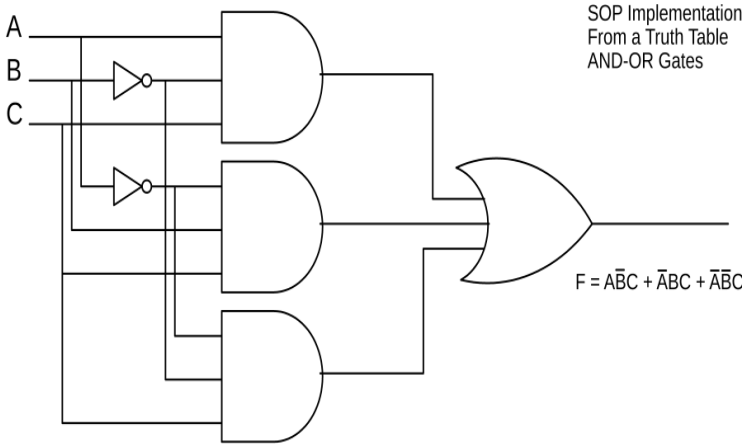



**Fig 4.1 Truth Table and Logic circuit for SOP form**

For POS expression by referring table 4.2

f=M0.M2.M4.M6.M7
f=∏M (0,2,4,6,7)

**Table 4.4 POS implementation from a truth table**

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

POS implementation
from a Truth Table

$F = (\bar{A}+\bar{B}+\bar{C})(A+\bar{B}+\bar{C})(A+\bar{B}+C)(A+B+\bar{C})(A+B+C)$

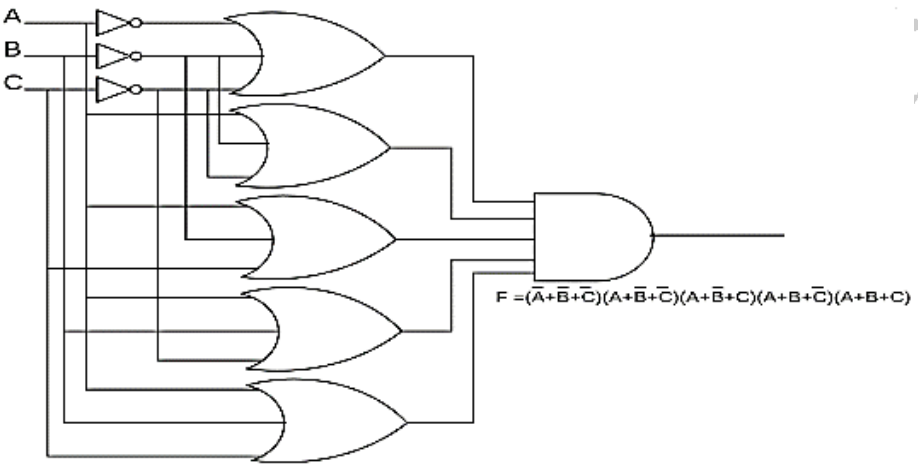



**Fig 4.2 Truth Table and Logic circuit for SOP form**

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Digital Multimeter | Digital Multimeter: 3 ½ digit display. | 2 |
| **2** | Breadboard | General Purpose Breadboard | 1 |
| **3** | DC power supply | +5 V Fixed power supply | 1 |
| **4** | IC | 7404,7408,7432 | 1 Each |
| **5** | LED, Resisters | Red color 5 mm ,330R | 1 Each |
| **6** | Connecting wires | Single strand 0.6 mm Teflon coating | LS |
| **7** | Digital IC Tester | Tests a wide range of Digital IC's such as 74 Series, 40/45 Series of CMOS IC's | 1 |

## IX. Precautions to be followed

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply

## X. Procedure

1. Make connections as shown in the respective circuit diagram in **Fig No.2** using breadboard
2. Connect +5 V to pin 14 and connect ground to pin no.7 of all IC's used.
3. Apply inputs as shown in observation table No.2 and observe the output on LED.
4. Note down the output in the observation table No.2.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure

……..…….

……..……

..…….……

……..

……..……

……..……

..……..……

**XIII. Observations and Calculations (use blank sheet provided if space not sufficient)**

| Inputs | | | Output | |
|---|---|---|---|---|
| **A** | **B** | **C** | **Output of SOP circuit** | **Output of POS circuit** |
| **0** | **0** | **0** | | |
| **0** | **0** | **1** | | |
| **0** | **1** | **0** | | |
| **0** | **1** | **1** | | |
| **1** | **0** | **0** | | |
| **1** | **0** | **1** | | |
| **1** | 1 | 0 | | |
| 1 | 1 | 1 | | |

**XIV. Result(s)**

……

……

……

……

**XV. Interpretation of results**

……

……

……

**XVI. Conclusion and recommendation**

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Define Min Term and Max Term.
2. Give the definition of Canonical form
3. Explain Significance of SOP
4. Standardize following Boolean expression into SOP form.

$$Y = AC + BC$$

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

**Practical No.5: Implement Multiplexer and Demultiplexer logic (The practical may be performed using virtual lab)**

## I. Practical Significance

In most of the electronic systems, the digital data is available on more than one line. It is necessary to route this data over a single line. Under such circumstances we require a circuit which selects one of the many inputs at a time. This circuit is a multiplexer (Mux), which has many inputs, one output and some select inputs. Multiplexer improves the reliability of the digital system because it reduces the number of external wired connections.

A Demultiplexer (or De-mux) is a device taking a single input and selecting one of the many data output lines, which is connected to the single input. An electronic demultiplexer can be considered as a single-input, multiple-output switch. De-multiplexers are mainly used in Boolean function generators and decoder circuits.

## II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

## III. Course Level Learning Outcome(s)

Students will be able to achieve & demonstrate the following COs on completion of course based learning

Use basic combinational and sequential logic circuits employing digital ICs.

## IV. Laboratory Learning Outcome(s)

Design a Combinational Circuit using Multiplexer IC-74LS153 (4:1 MUX).

Design a Combinational Circuit using Demultiplexer IC -74139.

## V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

## VI. Relevant Theoretical Background

**Multiplexer:**

Multiplexer is a combinational circuit which accepts multiple analog signals or digital signals and selects one signal and transmits over a shared medium. It has maximum of $2^n$ data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines. Multiplexer is also called as Mux. Since there are 'n' selection lines, there will be $2^n$ possible combinations of zeros and ones.

**Types of Multiplexer**

a. 2-to-1 (1 select line)
b. 4-to-1 (2 select lines)
c. 8-to-1 (3 select lines)
d. 16-to-1 (4 select lines)

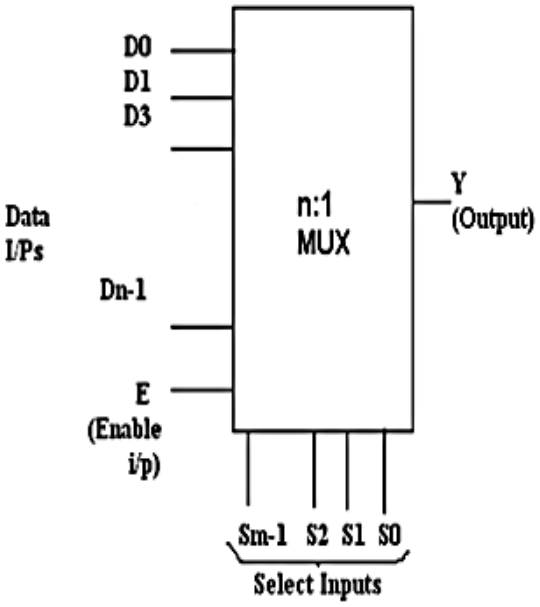



**Fig. 5.1 N:1 Multiplexer Block Diagram**

**Table No 5.1: Truth table of 8:1 MUX**

| Select Data Inputs | | | Output |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | $D_4$ |
| 1 | 0 | 1 | $D_5$ |
| 1 | 1 | 0 | $D_6$ |
| 1 | 1 | 1 | $D_7$ |

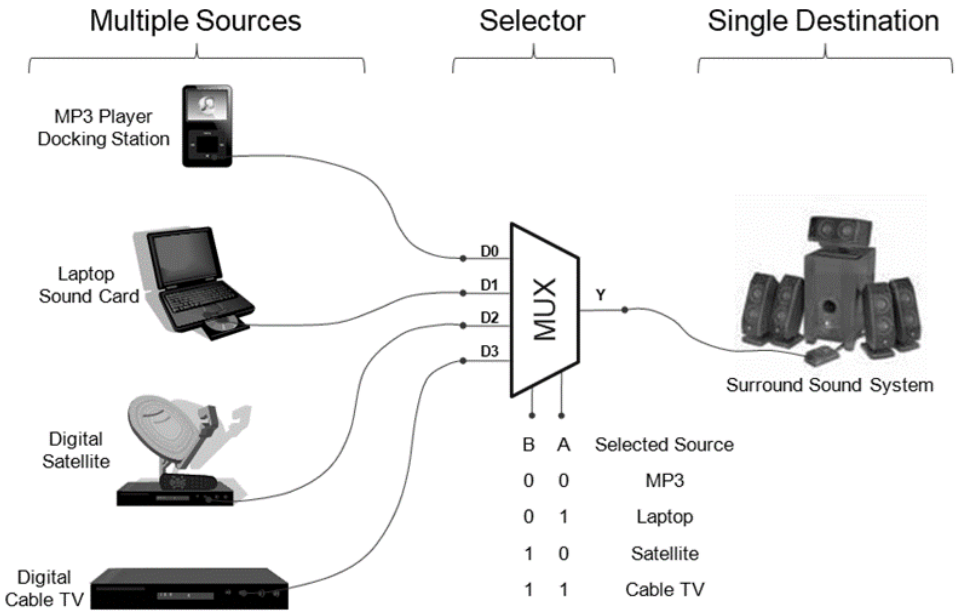



**Fig 5.2 Application of Multiplexer**

**Applications of Multiplexer-**

a. Multiplexers are used in Communication Systems like telephone networks, Satellite systems, Telemetry.
b. Broadcasting of Radio and Television signals would have been impossible without multiplexers.
c. Multiplexer is also used in data routing within the computer.
d. Multiplexers are widely used in computer memory to fetch data from specified memory locations.
e. Multiplexer is used as a switch setting Comparator and Function Generator**.**

**Multiplexer IC 74LS153**

74LS153 MUX has two separate 4:1 MUXs with separate enable signals on it**.**

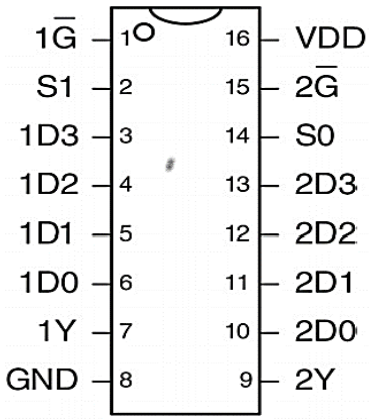



**Fig 5.3 Pin Diagram of IC 74LS153 4:1 Multiplexer**

**Table No 5.2: Available Multiplexer ICs**

| Sr. No | IC No. | Function | Output State |
|---|---|---|---|
| 1 | 74157 | Quad 2:1 MUX | Output is same as input given |
| 2 | 74158 | Quad 2:1 MUX | Output is inverted input |
| 3 | 74153 | Dual 4:1 MUX | Output is same as input given |
| 4 | 74352 | Dual 4:1 MUX | Output is inverted input |
| 5 | 74151A | 8:1 MUX | Both outputs are available |
| 6 | 74151 | 8:1 MUX | Output is inverted input |
| 7 | 74150 | 16:1 MUX | Output is inverted input |

**Demultiplexer:**

A Demultiplexer is a digital switch with a single input (source) and multiple outputs (destinations). The select lines determine which output the input is connected to. It has maximum of 2n data outputs, 'n' selection lines and single input line. One of these data outputs will be connected to the input based on the values of selection lines. Demultiplexer is also called as De-Mux. Since there are 'n' selection lines, there will be 2n possible combinations of zeros and ones.

**Types of Demultiplexer**

a. 1-to-2 (1 select line)
b. 1-to-4 (2 select lines)
c. 1-to-8 (3 select lines)
d. 1-to-16 (4 select lines)

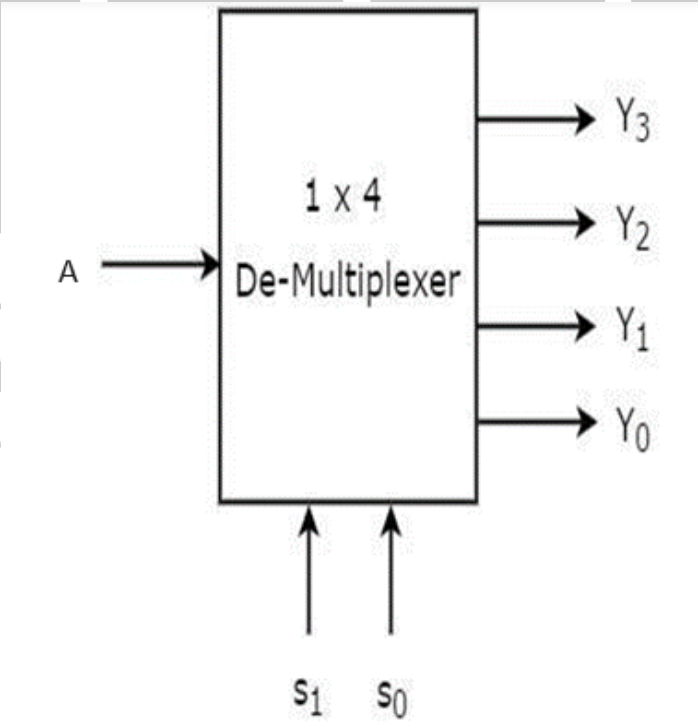



**Fig. 5.4 1:4 Demultiplexer Block Diagram**

**Table No 5.3 Truth table of 1:4 Demultiplexer**

| INPUTS | | Output | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 0 | 0 | 0 | 0 | 0 | A |
| 0 | 1 | 0 | 0 | A | 0 |
| 1 | 0 | 0 | A | 0 | 0 |
| 1 | 1 | A | 0 | 0 | 0 |

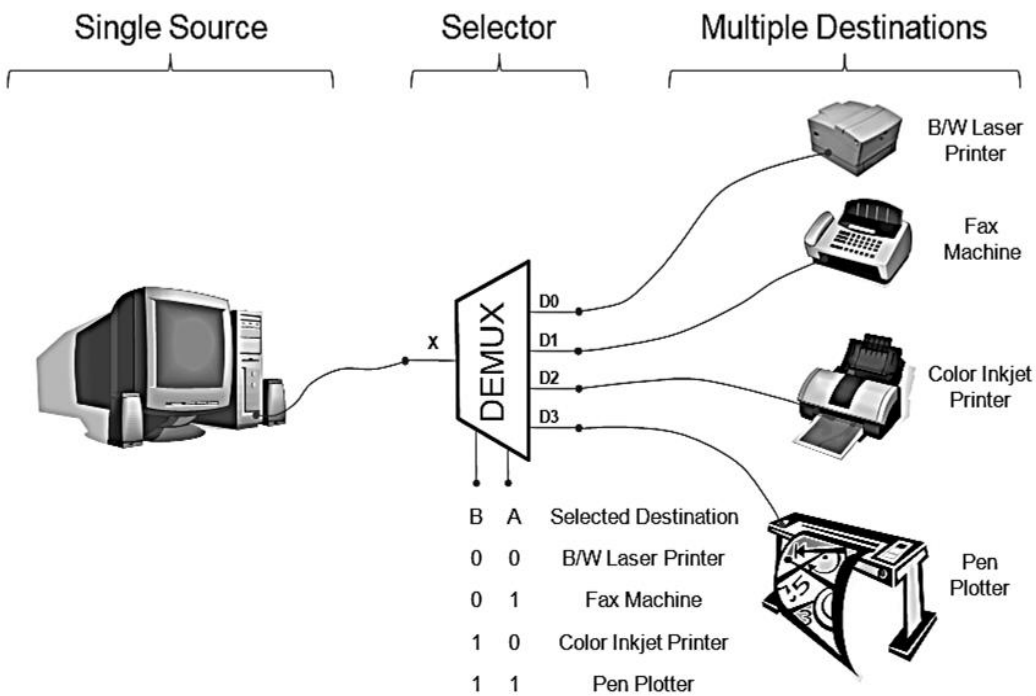



**Fig 5.5 Application of Demultiplexer**

**Applications of Demultiplexer:**

a. De-multiplexers are used in several input and output devices for data routing.
b. De-multiplexers are also employed for data transmission in synchronous systems.
c. De-multiplexers are also utilized in data acquisition systems.
d. De-multiplexers can be used for generating Boolean functions.
e. De-multiplexers can be used in serial to parallel converters.
f. De-multiplexers are used for broadcasting of ATM packets.

**Demultiplexer IC SN74HC139 (74139**)

The SN74HC139 (74139) IC provides two individual 2-line to 4-line decoders in a single package. The decoders take as input a two digit binary number 1 thru 4 (00, 01, 10, 11) and output by selecting one of four lines.

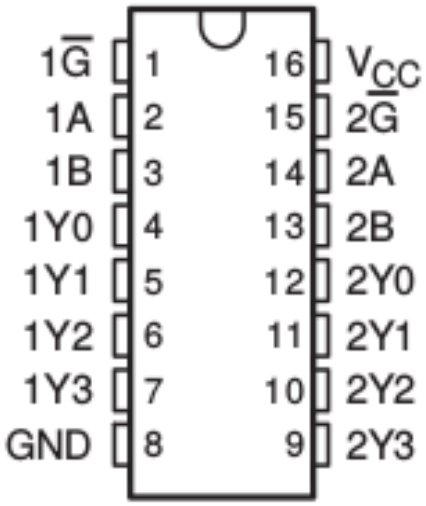



**Fig 5.6 Pin Diagram of IC 74LS139 1:4 Demultiplexer**

**Table No 5.4 Available Demultiplexer ICs**

| S. No. | IC No | Function | Output State |
|---|---|---|---|
| 1 | 74139 | Dual 1:4 demux | Output is inverted input |
| 2 | 74156 | Dual 1:4 demux | Output is open collector |
| 3 | 74138 | 1:8 demux | Output is inverted input |
| 4 | 74154 | 1:16 demux | Output is inverted input |
| 5 | 74159 | 1:16 demux | Output is open collector and same as input |

## VII. Circuit Diagram/Pin Diagram

(The students are expected to draw circuit diagram using above Multiplexer and Demultiplexer to verify the truth table)

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Digital Multimeter | 3 and 1/2 digit | **1** |
| **2** | DC Regulated Power Supply | 2 x 0-30 V; 0-2 Automatic Overload (Current Protection) Constant Voltage and Constant Current Operation Digital Display for Voltage and Current Adjustable Current Limiter Excellent Line and Load Regulation | **1** |
| **3** | MUX IC 74LS153, DEMUX IC 74139 | -- | **1 Each** |
| **4** | Bread board | 5.5 cm X 17 cm | **1** |
| **5** | Connecting wires | Single strand 0.6mm Teflon coating | **As required** |
| **6** | IC Tester | Digital IC Tester | **1** |
| **7** | LEDs | Red/Green/Yellow 5mm | **5** |
| **8** | Resistors | 330 Ω /0.25 W | **1** |
| **9** | Stripper | -- | **1** |
| **10** | Digital IC's Data sheets of ICs used in Lab | | |

## IX. Precautions to be followed

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply (0-5 V/ 500mA)

## X. Procedure

1. Make the connections as per the circuit diagram of multiplexer and give the supply voltage to relevant pin.
2. Connect the inputs from the source to input pins as per the logic levels.
3. Observe the output on LED for each combination of input as per truth table.
4. Verify the truth table.
5. Repeat the process for Demultiplexer.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

**Link of V Lab – Operation of 4:1 Multiplexer & 1:4 Demultiplexer**

Students can perform and study the operation of 4:1 Multiplexer & 1:4 Demultiplexer using following Virtual Lab experiment:

https://de-iitr.vlabs.ac.in/exp/multiplexer-demultiplexer/simulation.html

## XIII. Observations and Calculations

**Truth table for 4:1 Multiplexer (IC 74153)**

| Inputs | | | | Output |
|---|---|---|---|---|
| Strobe | Data input | Select Inputs | | Y |
| G' | Dn | S1 | S0 | |
| 0 | | 0 | 0 | |
| 0 | | 0 | 1 | |
| 0 | | 1 | 0 | |
| 0 | | 1 | 1 | |
| 1 | | X | X | |

**Truth table for 1:4 Demultiplexer (IC 74139)**

| Strobe | Data input | Select inputs | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| G' | Din | S1 | S0 | Y3 | Y2 | Y1 | Y0 |
| 0 | | 0 | 0 | | | | |
| 0 | | 0 | 1 | | | | |
| 0 | | 1 | 0 | | | | |
| 0 | | 1 | 1 | | | | |
| 1 | | X | X | | | | |

## XIV. Result(s)

……

……

……

## XV. Interpretation of results

……

……

……

## XVI. Conclusion and recommendation

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. State the function of enable input in a Multiplexer IC?
2. Give the applications of MUX and DEMUX.
3. Determine the output of IC 74153 if S0=1, S1=0
4. Implement the following functions using demultiplexer.
   i. $f1 = \Sigma m (0, 2, 4, 6)$
   ii. $f2 = \Sigma m (1, 3, 5)$

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References/Suggestions for further reading


1. https://www.youtube.com/watch?v=FKvnmxte98A
2. https://www.youtube.com/watch?v=t3Ed13z9uz8&t=5s
3. https://www.tutorialspoint.com/demultiplexers-and-their-applications
4. https://www.youtube.com/watch?app=desktop&v=kWPqbW28zAo
5. https://www.geeksforgeeks.org/difference-between-multiplexer-and-demultiplexer/


## I. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **ProcessRelated:15Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | Identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | Working in teams | 10% |
| **ProductRelated:10Marks** | | **40%** |
| 5 | Calculated the or ethical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total(25Marks)** | | **100 %** |

| Marks Obtained | | | Dated signature of Teacher |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No 6: Implementation of Latch

### I. Practical Significance

Latches are primarily used to store digit values within a circuit until they are required. They are often used in combination with other digital circuits to implement sequential circuits, Latches are widely used in digital systems for various applications, including data storage, control circuits, and flip-flop circuits.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Use basic combinational and sequential logic circuits employing digital ICs.

### IV. Laboratory Learning Outcome(s)

Verify states of the Latch using IC 74373.

### V. Relevant Affective Domain related outcome(s)

a. Follow precautionary measures.

b. Demonstrate working as a leader/ a team member

c. Follow ethical practices.

### VI. Relevant Theoretical Background

The 74HC373 high speed octal D-type latches utilize advanced silicon-gate CMOS technology. They possess the high noise immunity and low power consumption of standard CMOS integrated circuits, as well as the ability to drive 15 LS-TTL loads. Due to the large output drive capability and the 3-STATE feature, these devices are ideally suited for interfacing with bus lines in a bus organized system.

When the LATCH ENABLE input is HIGH, the Q outputs will follow the D inputs. When the LATCH ENABLE goes LOW, data at the D inputs will be retained at the outputs until LATCH ENABLE returns HIGH again. When a high logic level is applied to the OUTPUT CONTROL input, all outputs go to a high impedance state, regardless of what signals are present at the other inputs and the state of the storage elements. All inputs are protected from damage due to static discharge by internal diode clamps to Vcc and ground. In high impedance it does not draw current.

## VII. Actual Circuit diagram used in a laboratory

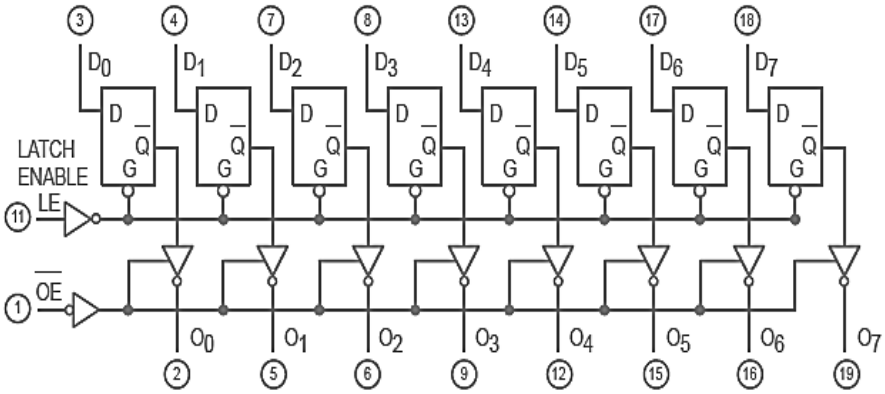



**Figure 6.1: Internal Logic Diagram of IC 74373**

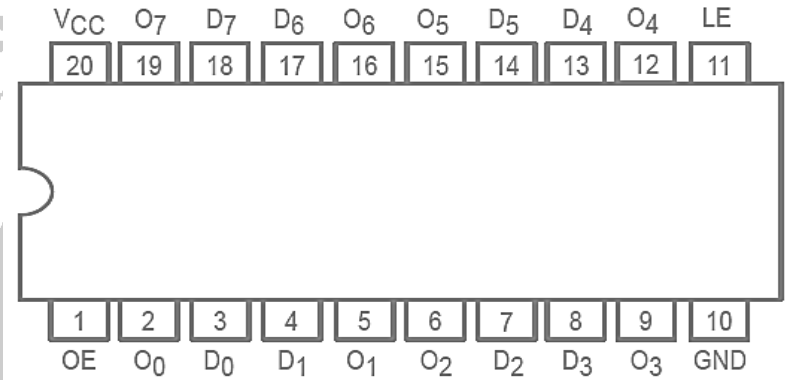



**Figure 6.2: Pin diagram of IC 74373**

**Table 6.1 Truth table of IC 74373**

| D | LE | $\overline{OE}$ | Qn |
|---|---|---|---|
| H | H | L | H |
| L | H | L | L |
| X | L | L | Qo |
| X | X | H | Z* |

D0-D7 are data inputs.
Q0-Q7 are data outputs
OE –Output Enable
LE-Latch Enable
H=HIGH Voltage Level
L=LOW Voltage Level
X= Don't Care
Z=High Impedance

**VIII. Required Resources/apparatus/equipment with specifications**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Digital Multimeter | 3 and 1/2 digit | **1** |
| **2** | DC Regulated Power Supply | 2 x 0-30 V; 0-2 Automatic Overload (Current Protection) Constant Voltage and Constant Current Operation Digital Display for Voltage and Current Adjustable Current Limiter Excellent Line and Load Regulation | **1** |
| **3** | LATCH IC 74LS373 | 20 pin IC | **1** |
| **4** | Bread board | 5.5 cm X 17 cm | **1** |
| **5** | Connecting wires | Single strand 0.6mm Teflon coating | **As required** |
| **6** | IC Tester | Digital IC Tester | **1** |
| **7** | LEDs | Red/Green/Yellow 5mm | **5** |
| **8** | Resistors | 330 Ω /0.25 W | **1** |
| **9** | Stripper | -- | **1** |
| **10** | Digital IC's Data sheets of ICs used in Lab | | |

**IX. Precautions to be followed**

1. Test the IC using digital IC tester before conducting the experiment
2. Check Circuit connections before switch on the power supply
3. Give suitable power supply (0-5 V/ 500mA)

**X. Procedure**

1. Make the connections as per the circuit diagram of multiplexer and give the supply voltage to relevant pin.
2. Connect the inputs from the source to input pins as per the logic levels.
3. Observe the output on LED for each combination of input as per truth table.
4. Verify the truth table.

**XI. Resources used**

| Sr. No. | Name of Resource | Specifications | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| Sr. No. | Name of Resource | Specifications | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure

……

……

……

……

## XIII. Observation Table

| Sr. No. | D input | LE | $\overline{OE}$ | Qn |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | |
| 2 | 0 | 1 | 0 | |
| 3 | X | 0 | 0 | |
| 4 | X | x | 1 | |

## XIV. Result(s)

……

……

……

……

## XV. Interpretation of results

……

……

……

……

## XVI. Conclusion and recommendation

……

……

……

……

**XVII. Practical related questions**

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Give advantages and disadvantages of latches?
2. Draw latch using NAND gates only.
3. Draw latch using NOR gates only.

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References/Suggestions for further reading

1. https://youtu.be/W4g0CLsD6bQ?feature=shared
2. https://youtu.be/MEC7iiK6nV4?feature=shared

## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| Marks Obtained | | | Dated signature of Teacher |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 7: Verification of contents of general purpose, segment registers, flags and memory locations of different segments during execution of the program

### I. Practical Significance

A Microprocessor is an important part of a computer architecture without which one will not be able to perform anything on computer. It is a programmable device that takes in input, performs some arithmetic and logical operations over it and produces the desired output. In simple words, a Microprocessor is a chip that can fetch instructions from memory, decode and execute them, and give results. It is important to study and know the use of different registers and memory available in 8086 microprocessors so that programmers can make efficient use of these while programming microprocessor based systems.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve & demonstrate the following COs on completion of course based learning

Perform operations on registers using 8086 instructions.

### IV. Laboratory Learning Outcome(s)

Develop an assembly language program to verify the contents of general purpose, Segment registers, flags and contents of memory locations of segments

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI. Relevant Theoretical Background

There are 8 general-purpose registers in the 8086 microprocessor. All general purpose registers of the 8086 microprocessor can be used for arithmetic and logic operations. General-purpose registers are used to store temporary data within the microprocessor.

The 8086 microprocessor has segmented memory architecture, which means that memory is divided into segments that are addressed using both a segment register and an offset. The segment registers points to the start of a segment, while the offset specifies the location of a specific byte within the segment

The flag register is an important component of the 8086 microprocessor because it is used to determine the behavior of many conditional jump and branch instructions. The various flags in the flag register are set or cleared based on the result of arithmetic, logic, and other instructions executed by the processor.

Following diagram shows the general organization of different registers in 8086 and flag register of 8086.

.

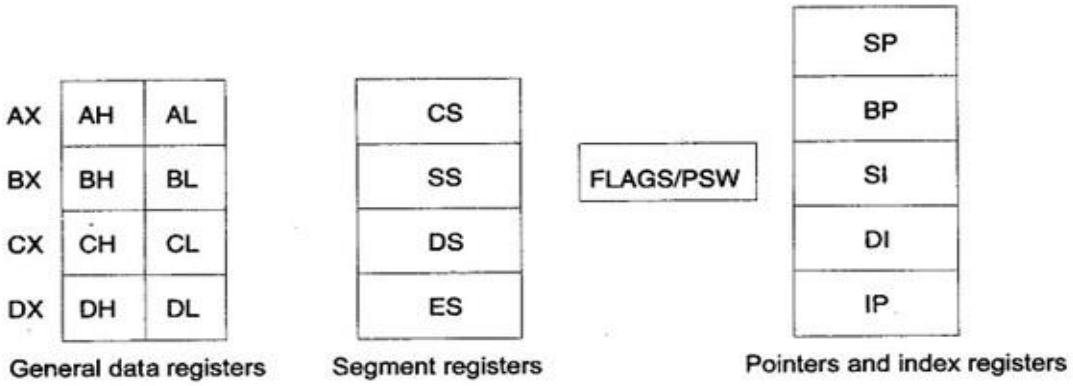



**Fig. 7.1 Register organization of 8086 Microprocessor**

**Functions of general purpose and segment registers-**

General purpose registers can be used for some specific functions apart from temporary data storage.

**Table No 7.1 General purpose registers**

| Sr. No | Name of the register | Function |
|---|---|---|
| 1 | **AX Accumulator** | Accumulator can be used for I/O operations and string manipulation |
| 2 | **BX Base Register** | This register usually contains a data pointer used for based, based indexed or register indirect addressing. |
| 3 | **CX Count Register** | This register can be used in Loop, shift/rotate instructions and as a counter in string manipulation, |
| 4 | **DX Data Register** | This register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number. |

**Segment Registers-**

There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers. Segment registers are used to hold the starting address (Base address) of the segment defined by the user.

**Table No 7.2 Segment registers**

| Sr. No | Name of the register | Function |
|---|---|---|
| 1 | **CS Code Segment** | The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. |
| 2 | **DS Data Segment** | All data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. |
| 3 | **SS Stack Segment** | All data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment |
| 4 | **ES Extra Segment** | The Destination Index (DI) register references the ES segment in string manipulation instructions. |

**Flag Registers:**

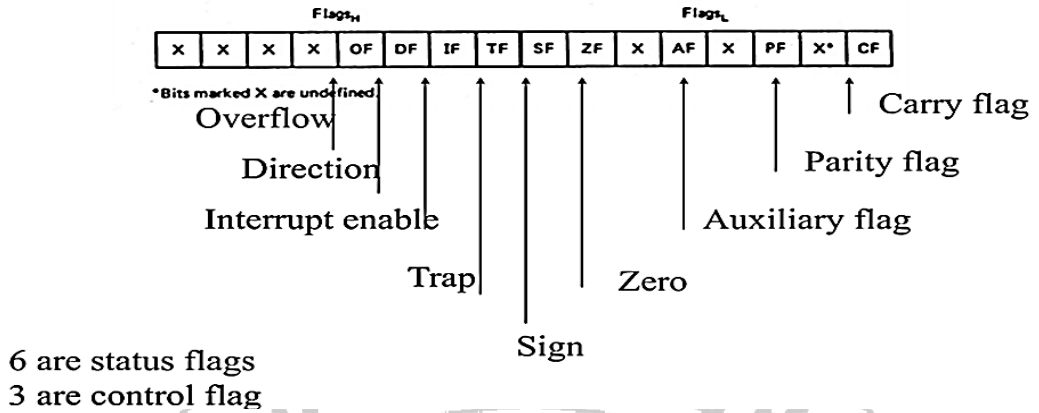



**Fig. 7.2 Flag register/Program Status Word (PSW)**

**Table No 7.3 Functions of different flags in flag register-**

| Sr. No | Name of the flag | Function |
|---|---|---|
| **1** | Overflow Flag (OF) | Set if the result is too large positive number, or is too small negative number to fit into the destination operand. |
| **2** | Direction Flag (DF) | If set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented |
| **3** | Interrupt-enable Flag (IF) | Setting this bit enables maskable interrupts |
| **4** | Single-step/Trap Flag (TF) | If set then a single-step interrupt will occur after the next instruction. |
| **5** | Sign Flag (SF) | Set if the most significant bit of the result is set otherwise reset. |
| **6** | Zero Flag (ZF) | Set if the result of operation is zero otherwise reset. |
| **7** | Auxiliary carry Flag (AF) | Set if there is a carry from or borrow to bits 0-3 in the AL register otherwise reset. |
| **8** | Parity Flag (PF) | Set if parity (the number of "1" bits) in the low-order byte of the result is even otherwise reset. |
| **9** | Carry Flag (CF) | Set if there is a carry from or borrow to the most significant bit during last result calculation otherwise reset. |

## VII. Algorithm/Flowchart

**Add two 8-bit numbers in order to understand the use of different registers and memory location in 8086**

1. Initialize the data segment with numbers on which operation is to be performed.
2. Initialize necessary variables to store the number and the result generated after operation.
3. Perform arithmetic operations by using appropriate instruction.
4. Use proper instructions to store the result in memory.
5. See the status of different registers and memory location after final result.

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | As per batch size |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and its peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of a given program. (Use blank space provided or attach more pages if needed)
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, hen type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……..……

……

..……...

…….

…….

## XIII. Program code with comments- Sample program with explanation

Students should refer to the below sample program and try to write and execute simple programs and observe the content of different general-purpose registers, segment registers, flags and memory locations after executing the program in TASM.

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 0B6H | Declaration of variable |
| | NO2 DB 7CH | Declaration of variable |
| | SUM DB ? | Declaration of variable |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register AL |
| | MOV BL, NO2 | Second number in register BL |
| | ADD AL, BL | Add second number to first number |
| | MOV SUM, AL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

```
+=========================================================================+
 Power Programming Tools 64 Bit By Sagar Taware :Computer Dep. NESGOI
+=========================================================================+

Type Proper Command To Perform the Desired Action

Command                .       Action
                       .
Edit          -        Open MS-DOS Editor
TASM          -        Compilation
tlink         -        Perform Linking
td            -        Launch Turbo Debugger
Exit          -        Exit Tasm 1.2

        For Compiling your files tasm "yourfilename".asm
e.g for compiling st.asm command is : tasm st.asm
For Linking and debugging same as 32 bit : tlink,td. tlink st.obj
   For RUN Use st1.exe   **Use Alt + ENTER for Maximize Window**
   Complink,DPMIload and TasmX also available using 32bit commands

C:\TASM>
```

**Fig No 7.3 TASM Window**

```
File  Edit  Search  View  Options  Help
                C:\TASM\ADD.ASM
DATA SEGMENT
NO1 DB 0B6H
NO2 DB 7CH
SUM DB  ?
DATA ENDS

CODE SEGMENT
Start:ASSUME CS:CODE, DS:DATA
MOV DX, DATA
MOV DS, DX
MOV AL, NO1
MOV BL, NO2
ADD AL, BL
MOV SUM, AL
MOV AH, 4CH
CODE ENDS
END START

F1=Help                                  Line:1    Col:1
```

**Fig No 7.4 View of Editor (TASM)**

```
 TASM          -       Compilation
 tlink         -       Perform Linking
 td            -       Launch Turbo Debugger
 Exit          -       Exit Tasm 1.2
 ____________________________________________________________________
        For Compiling your files tasm "yourfilename".asm
e.g for compiling st.asm command is : tasm st.asm
For Linking and debugging same as 32 bit : tlink,td. tlink st.obj
   For RUN Use st1.exe   **Use Alt + ENTER for Maximize Window**
   Complink,DPMIload and TasmX also available using 32bit commands
 ____________________________________________________________________

C:\TASM>edit

C:\TASM>tasm add.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   add.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  476k


C:\TASM>
```

**Fig No 7.5 View of Assembler**

```
≡ File  Edit  View  Run  Breakpoints  Data  Options  Window  Help        READY
[■]=CPU 80486                    ds:0000 = B6          1=[↑][↓]
cs:0000 BAAD48        mov    dx,48AD        ax 0032    c=1
cs:0003 8EDA          mov    ds,dx          bx 007C    z=0
cs:0005 A00000        mov    al,[0000]      cx 0000    s=0
cs:0008 8A1E0100      mov    bl,[0001]      dx 48AD    o=0
cs:000C 02C3          add    al,bl          si 0000    p=0
cs:000E A20200        mov    [0002],al      di 0000    a=1
cs:0011 B44C          mov    ah,4C          bp 0000    i=1
cs:0013 0000          add    [bx+si],al     sp 0000    d=0
cs:0015 0000          add    [bx+si],al     ds 48AD
cs:0017 0000          add    [bx+si],al     es 489D
cs:0019 0000          add    [bx+si],al     ss 48AC
cs:001B 0000          add    [bx+si],al     cs 48AE
cs:001D 0000          add    [bx+si],al     ip 0011
ds:0000 B6 7C 32 00 00 00 00 00 ╢¦2
ds:0008 00 00 00 00 00 00 00 00
ds:0010 BA AD 48 8E DA A0 00 00 ‖¦HÄ┌á        ss:0002 6474
ds:0018 8A 1E 01 00 02 C3 A2 02 è▲☺ ☻│ó☻     ss:0000►0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

**Fig No 7.6 View of TD (Turbo Debugger)**

## XIV. Observations

Addition operation will be performed in the following way in a microprocessor during which an auxiliary carry and carry is generated which can be seen in AF and CF, Parity of number in AL

register is odd hence PF = 0. MSB is zero hence SF = 0, result is non zero hence ZF = 0. Data and results can be seen in DS.

| | | | AF |
|---|---|---|---|
| **Carry** | | | **1111 1** |
| **Operation** | **B6H** | | **1011 0110** |
| | **+ 7CH** | | **0111 1100** |
| **1** | **32H** | **1** (CF) | **0011 0010** **Content of AL register** |

Different registers and memory locations can be observed as below after execution of the program in Turbo Debugger.

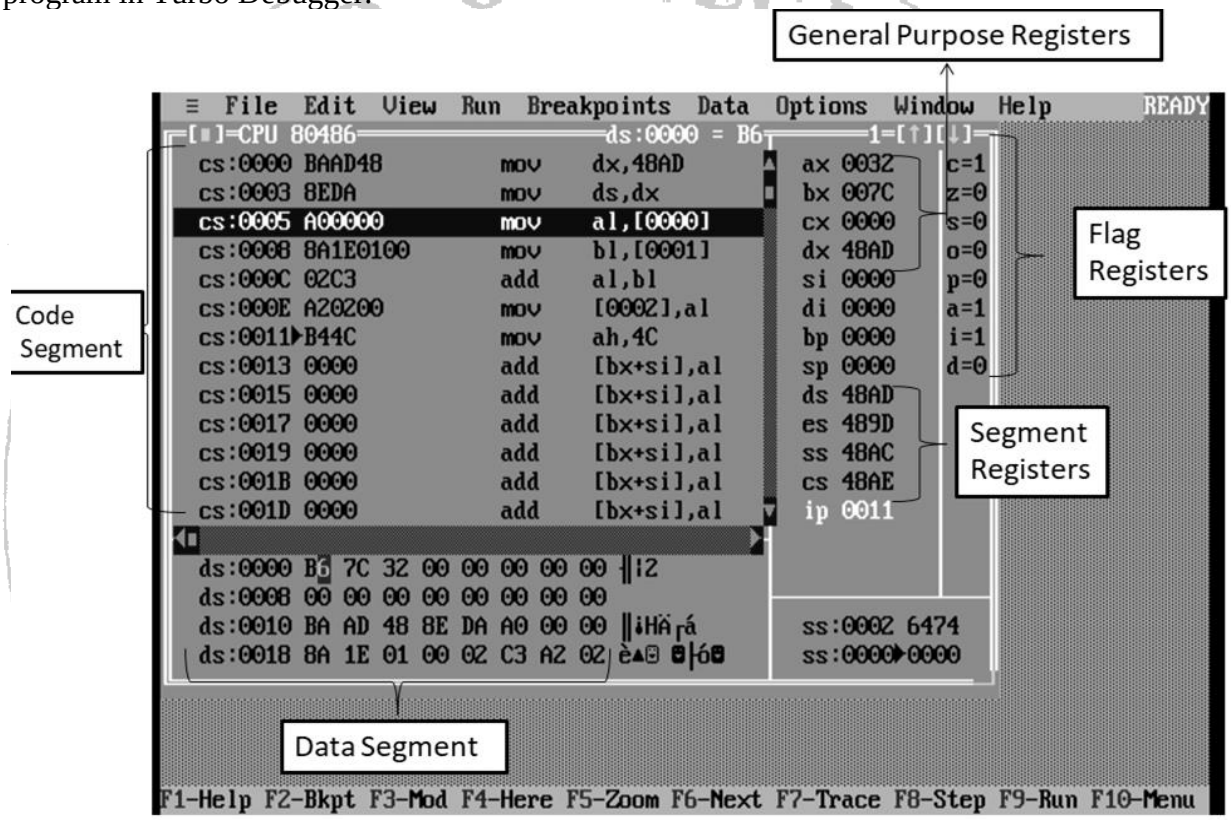



**Fig No 7.7- Output Window showing all Registers, Code and Data Segment**

## XV. Result(s)/Output of the program

……..…

…….…

……..…

## XVI. Conclusion and recommendation

……..…

……..……

……..……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write an ALP to add 16-Bit numbers and check the content of different registers.
2. Write the content of AL register and flag after execution of following code-

```
MOV AL, 99
ADD AL,01
```

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References /Suggestions for further reading

1. https://www.geeksforgeeks.org/architecture-of-8086/?ref=lbp
2. https://www.geeksforgeeks.org/general-purpose-registers-8086-microprocessor/?ref=lbp

## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| Marks Obtained | | | Dated signature of Teacher |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

**Practical No. 8: * Assembly language programming for addition and subtraction for hexadecimal numbers.**

**I. Practical Significance**

Assembly language programming is useful to be aware of the programming environment and development of code and debugging and execution skills. It is easier to understand and saves a lot of time for the programmer and code optimization.

**II. Industry/Employer Expected Outcome(s)**

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

**III. Course Level Learning Outcome(s)**

Students will be able to achieve and demonstrate the following COs on completion of course based learning

Use an 8086 microprocessor environment to build and execute assembly language programs.

**IV. Laboratory Learning Outcome(s)**

Develop an assembly language program to add 8 bit and 16-bit signed/ unsigned hexadecimal numbers.

Develop an assembly language program to Subtract two 8-bit and 16-bit signed/ unsigned hexadecimal numbers.

**V. Relevant Affective Domain related outcome(s)**

1. Handle IC and Equipment carefully.
2. Follow safe practices.

**VI. Relevant Theoretical Background**

**ADD / ADC destination, source**

The ADD instruction adds a number from source to a number from destination. The ADC instruction adds the carry flag into the result of addition. The source may be an immediate number, a register, or a memory location as specified by any 24 addressing modes. The destination may be a register or a memory. The source and destination must be of the same type and cannot both be memory locations. Destination should not be an immediate number.

**Flag affected:** OF, CF, PF, AF, SF, ZF

**Syntax & Operation:**

1. **ADD <DEST> ,<SRC>** Destination <— destination + source
2. **ADC <DEST>,<SRC>** Destination <— destination + source **+** CF

**SUB / SBB destination, source**

The SUB instruction is used to subtract the data in source from the data in destination and the stores result in destination. The SBB instruction is used to subtract the source operand and the borrow [CF], which may reflect from the result of the previous operations, from the destination operand, and the result is stored in destination operand. Source must be a register or memory location or immediate data and the destination must be a register or a memory location. The destination operands should not be immediate data and the source and destination both should not be memory operands.

**Flag affected:** OF, CF, PF, AF, SF, and ZF.

**Syntax & Operation:**

1. **SUB <DEST> ,<SRC>**
   Destination <— destination - source
2. **SBB <DEST> ,<SRC>**
   Destination <— destination + source - CF

## VII. Algorithm/Flow chart

**Algorithm for program to arithmetic operation.**

1. Initialize the data segment with numbers on which Arithmetic operations to be performed.
2. Initialize necessary variables to store the number and the result generated after operation.
3. Perform arithmetic operations by using appropriate instruction.
4. Store the final result.

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | As per batch size |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and their peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of given program
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.

5. Once the assembly language program is created, then type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……..

……

……

……..

## XIII. Program code with comments

**Program for addition of 8 bit numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 26H | Declaration of variable |
| | NO2 DB 54H | |
| | NO3 DB ? | |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register |
| | ADD AL, NO2 | Add second number to first |
| | MOV NO3, AL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

**Program for subtraction of 8 bit numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 26H | Declaration of variable |
| | NO2 DB 54H | |
| | NO3 DB ? | |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register |
| | SUB AL, NO2 | Add second number to first |
| | MOV NO3, AL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

## XIV. Observations

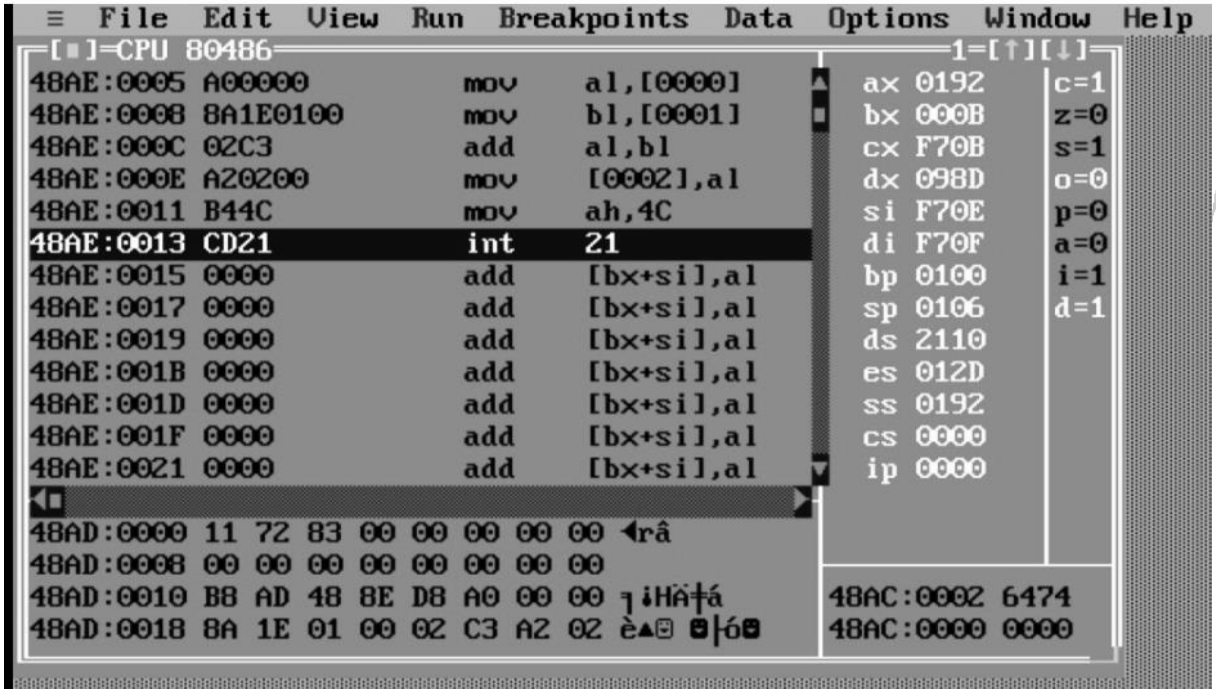



**Figure 8.1: Program output after adding two 8 bit numbers.**

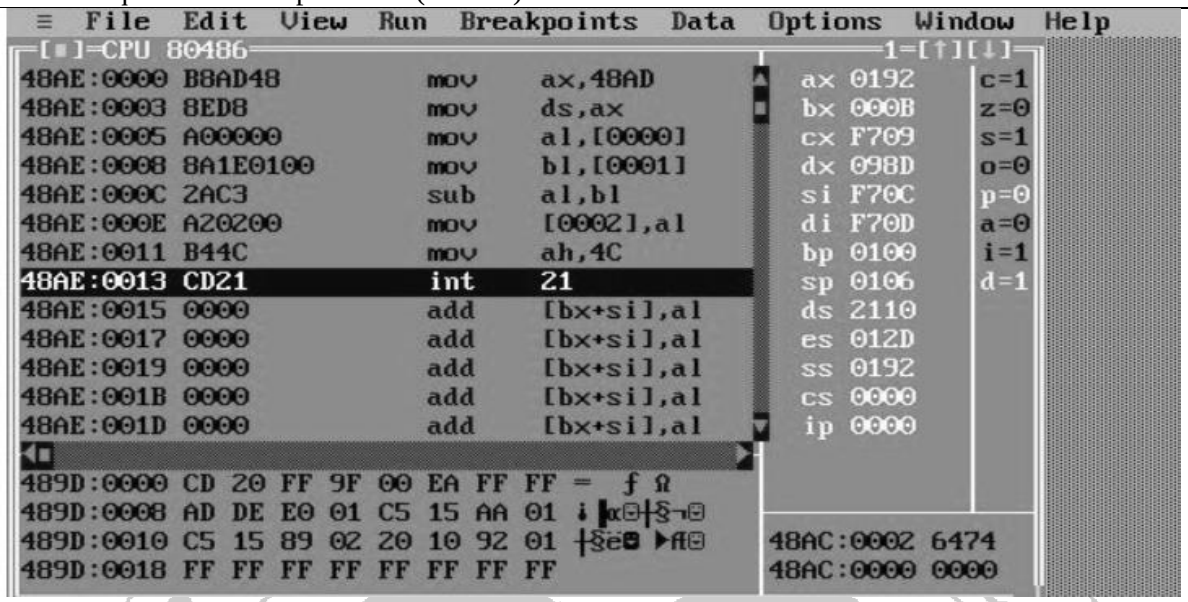



**Figure 8.2: Program output after subtracting two 8 bit numbers.**

**Student Activity:**

Observe and write the contents of registers, memory location in Code and Data Segment using debugger TD or Debug after the execution of the program.

| Registers | | | Flag Register | | |
|---|---|---|---|---|---|
| | After | Before | | | |
| AX | | | Carry Flag | CF | |
| BX | | | Zero Flag | ZF | |
| CX | | | Sign Flag | SF | |
| DX | | | Overflow Flag | OF | |
| SI | | | Parity Flag | PF | |
| DI | | | Auxiliary Carry Flag | AF | |
| BP | | | Interrupt Flag | IF | |
| SP | | | Direction Flag | DF | |
| DS | | | | | |
| ES | | | | | |
| SS | | | | | |
| CS | | | | | |
| IP | | | | | |

| **Address** | **Contents** | **Address** | **Contents** |
|---|---|---|---|
| CS:0000 | | CS:0008 | |
| CS:0001 | | CS:0009 | |
| CS:0002 | | CS:000A | |
| CS:0003 | | CS:000B | |
| CS:0004 | | CS:000C | |
| CS:0005 | | CS:000D | |
| CS:0006 | | CS:000E | |
| CS:0007 | | CS:000F | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

**XV. Result(s)/Output of the program**

……..

..……

..……

**XVI. Conclusion and recommendation**

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write down the function of flags used in arithmetic operation.
2. Explain the instruction ADC and SBB used in the program.
3. Complete the following table after execution of above programs.

| **For 8 bit/16 bit addition** | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| No 1 | 48H | 1234H |
| No 2 | 19H | 5678H |
| Sum result | | |

| **For 8 bit/16 bit Subtraction** | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| No 1 | 48H | 1234H |
| No 2 | 19H | 5678H |
| Sub | | |

4. Write the program for 16 bit addition
5. Write the program for 16 bit subtraction.

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

**XVIII. References /Suggestions for further reading**


1. https://www.tutorialspoint.com/assembly_programming/
2. https://mysc.altervista.org/biginners-guide-8086/
3. https://www.geeksforgeeks.org/8086-program-add-2-bcd-numbers/
4. https://www.geeksforgeeks.org/8086-program-subtract-two-16-bit-bcd-numbers/


**XIX. Assessment Scheme**

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 9: Apply assembly language programming logic for Addition, Subtraction and Multiplication for BCD numbers.

### I. Practical Significance

In high level language programming, the decimal numbers system is used to perform arithmetic operations. However in microprocessors all arithmetic operations are performed on Hexadecimal or Binary numbers. Hence Binary Coded Decimal (BCD) representation of decimal number system which is easy to encode and decode helps to understand microprocessor based systems.

### II. Industry/Employer Expected Outcome (s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome (s)

Students will be able to achieve and demonstrate the following COs on completion of course based learning
Use an 8086-microprocessor environment to build and execute assembly language programs.

### IV. Laboratory Learning Outcome(s)

Develop an assembly language program to add 8-bit and 16-bit BCD numbers
LLO 9.2 Develop an assembly language program to subtract two 8-bit and 16-bitBCD numbers

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI. Relevant Theoretical Background

In assembly language program special instructions are required to convert arithmetic operation result of decimal numbers to appropriate result in BCD format. It is require using DAA, DAS and AAM instructions to perform Addition, Subtraction and Multiplication operations on BCD numbers. The operation of these instructions is explained as below-

#### 1. DAA ( Decimal Adjust Accumulator)

This instruction is used to convert the result of the addition of two packed BCD numbers to a valid BCD number. The result has to be only in the AL register. DAA instruction should be used after ADD/ADC instruction. DAA instruction affects AF, CF, PF and ZF. OF is undefined.

a. After addition if the lower nibble is > 9 or AF = 1, then AL = AL+ 06H
b. After addition, if the upper nibble is > 9 or CF = 1, then AL = AL+ 60H
c. If both the above conditions are satisfied, then AL= AL+ 66H

Unpacked BCD uses one byte (eight bits) to represent each decimal digit, while packed BCD uses four bits to represent each decimal digit. Packed BCD is more space-efficient but requires additional processing to convert to and from unpacked BCD.

**2. DAS (Decimal Adjust After Subtraction)**

This instruction is used to convert the result of the subtraction of two packed BCD numbers to a valid BCD number. The result has to be only in the AL register. DAS instruction should be used after SUB/SBB instruction. DAS instruction affects AF, CF, PF and ZF. OF is undefined.

. After subtraction if the lower nibble is > 9 or AF = 1, then AL = AL - 06H
a. After addition, if the upper nibble is > 9 or CF = 1, then AL = AL - 60H
b. If both the above conditions are satisfied , then AL= AL - 66H

**3. AAM (ASCII Adjust After Multiplication)**

This instruction is used to convert the product in AL after the multiplication into unpacked BCD format. The higher nibble of multiplication operands is filled with zeros. The instruction should be used after MUL and the result is placed in the AX register.

The binary number in AL register is divided by 10 and quotient is stored in the register AH, Remainder in AL. Operation Performed:--

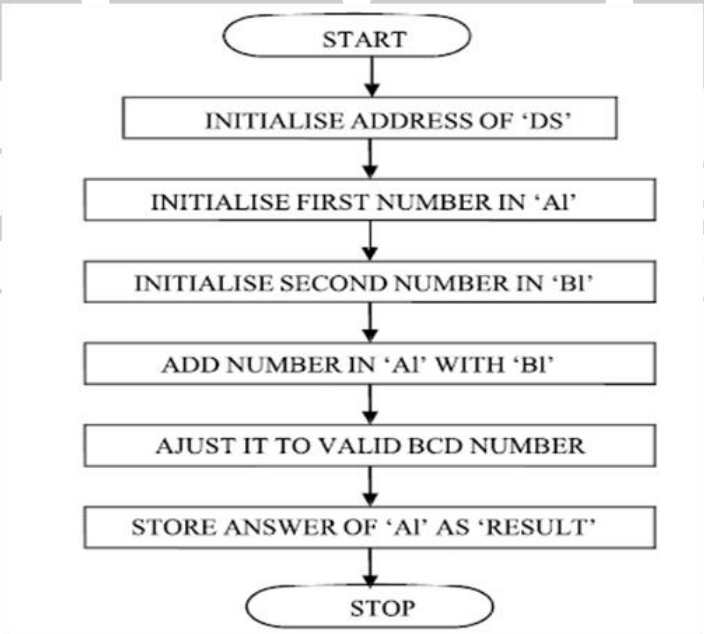
AL = AL MOD 10
AH = AL / 10

## VII. Algorithm/Flow chart

**Algorithm for program to Add/Sub/Multiply two BCD numbers in order to understand BCD arithmetic operation.**

1. Initialize the data segment with numbers on which BCD operations to be performed.
2. Initialize necessary variables to store the number and the result generated after operation.
3. Perform arithmetic operations by using appropriate instruction.
4. Use proper instruction to convert the result into BCD.
5. Store the final result.

**Flowchart for Addition of two 8 bit BCD numbers**



**VIII. Resources Required**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | **As per batch size** |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

**IX. Precautions to be followed**

1. Handle computer systems and their peripherals properly.
2. Shut down computers properly.

**X. Procedure**

1. Write an algorithm and draw a flowchart of a given program. (Use blank space provided or attach more pages if needed)
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, hen type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

**XI. Resources Used**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quanti ty |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……..

…….

..……

……..

## XIII. Program code with comments

**Sample program for BCD addition of 8 bit numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 26H | Declaration of variable |
| | NO2 DB 54H | |
| | NO3 DB ? | |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment and code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register |
| | ADD AL, NO2 | Add second number to first |
| | DAA | Decimal adjust |
| | MOV NO3, AL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

**Note:** Students should refer to the above program and write for other BCD operations like Subtraction and Multiplication.

……..

..……

..……

……..

..……

..……

……..

..……

..……

……..

..……

..……

## XIV. Observations

Observe and write the content of Register, memory locations in Code and Data Segment using debugger TD or Debug after the execution of the program.

| Types of registers | Registers | | | Flag Registers | | |
|---|---|---|---|---|---|---|
| | | Before | After | Carry flag | CF | |
| General Purpose register | AX | | | Zero flag | ZF | |
| | BX | | | Sign flag | SF | |
| | CX | | | Overflow flag | OF | |
| | DX | | | Parity flag | PF | |
| Index register | SI | | | Auxiliary Carry flag | AF | |
| | DI | | | Interrupt flag | IF | |
| Base Pointer | BP | | | Direction flag | DF | |
| Stack Pointer | SP | | | | | |
| Segment Register | DS | | | | | |
| | ES | | | | | |
| | SS | | | | | |
| | CS | | | | | |
| Instruction Register | IP | | | | | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

## XV. Result(s)/Output of the program

……..

..……

..……

## XVI. Conclusion and recommendation

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write down the function of flags used for BCD arithmetic operation.
2. Explain the instruction DAA and DAS used in the program.
3. Write some applications of BCD numbers.
4. Complete the following table after execution of above programs.

| **For 8 bit/16 bit BCD addition** | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| BCD No 1 | 48H | A5C4H |
| BCD No 2 | 19H | 6E78H |
| Sum result before DAA | | |
| Sum result after DAA | | |

| **For 8 bit/16 bit BCD Subtraction** | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| BCD No 1 | 48H | 9845H |
| BCD No 2 | 19H | A372H |
| Sub result before DAS | | |
| Sub result after DAS | | |

| **For 8 bit/16 bit BCD Multiplication** | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| BCD No 1 | 04H | 2500H |
| BCD No 2 | 06H | 3000H |
| Result before AAM | | |
| Result after AAM | | |

**[Space for Answers]**

..……

……

……

……

……

……

……

……

## XVIII. References /Suggestions for further reading


1. https://www.tutorialspoint.com/assembly_programming/
2. https://www.geeksforgeeks.org/8086-program-add-two-16-bit-bcd-numbers-carry/
3. https://www.geeksforgeeks.org/8086-program-subtract-two-16-bit-bcd-numbers/
4. https://www.geeksforgeeks.org/8086-program-multiply-two-16-bit-numbers/


## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related: 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total (25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

**Practical No. 10: * Assembly language programming for multiplication and division**

**I. Practical Significance**

Multiplication is developed for equal group situations in advanced computations. Various real-world problems can be solved with multiplication. Divide means to split, separate, distribute, share or make groups of equal items.

**II. Industry/Employer Expected Outcome(s)**

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

**III. Course Level Learning Outcome(s)**

Students will be able to achieve and demonstrate the following COs on completion of course based learning

Use 8086 microprocessor environment to build and execute assembly language programs.

**IV. Laboratory Learning Outcome(s)**

Develop assembly language programming for multiplication and division.

**V. Relevant Affective Domain related outcome(s)**

1. Handle IC and Equipment carefully.
2. Follow safe practices.

**VI. Relevant Theoretical Background**

**MUL SOURCE 8/16**

This instruction is used to multiply an unsigned byte (8 bits) by a byte (8 bits) or to multiply two unsigned words (16 bits).

**a) Byte Multiplication**:

a. The 8-bit Multiplicand should be in the AL register.
b. The 8-bit Multiplier should be loaded in an 8 bit register or a memory location.
c. After multiplication,

AX←Product (16 bits).

**b) Word Multiplication:**

a. A 16-bit multiplicand must be loaded in the AX register.
b. The 16 Multiplier must be loaded in a 16 bit register or a memory location.

After multiplication
DX ← 16 bit Most Significant Word of the product
AX ← bit Least Significant Word of the product.

**IMUL SOURCE8/16:** This instruction is used to multiply 8 bit signed number in source register or memory location to an 8 bit signed number in AL register. Similar to MUL instruction, the 16-bit signed result is available in AX register.

The 32-bit product for 16-bit signed multiplication is available in DX and AX registers.

**DIV/IDIV source**

**DIV source:** divides an unsigned word by an unsigned byte during 16/8 division, and to divide unsigned double word i.e., 32-bits by an unsigned word during 32/16 division.
The word (dividend) must be in the AX register and a byte (divisor) may be in any 8-bit register or memory location during the division of a word by a byte.
After the division, 8-bit quotient will be stored in AL register and 8-bit remainder will stored in AH register

**IDIV source**: This instruction is used to divide signed word by a signed byte or to divide signed double word by a single signed word.
**Flag affected:** None and OF, CF, PF, AF, SF, ZF are undefined.
**Operation**

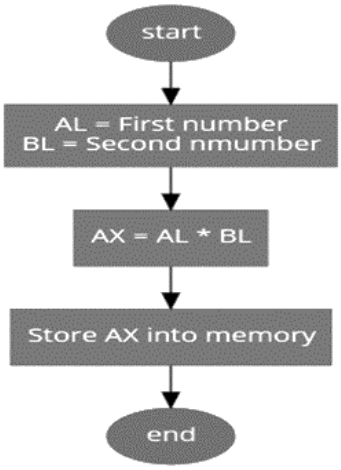
a. If source is byte, then
   AL← AL/unsigned 8-bit source (Quotient)
   AH← AL MOD unsigned 8-bit source (Remainder)
b. If source is word then
   AX←DX:AX/unsigned 16-bit source (Quotient)
   DX← DX MOD unsigned 16-bit source (Remainder)

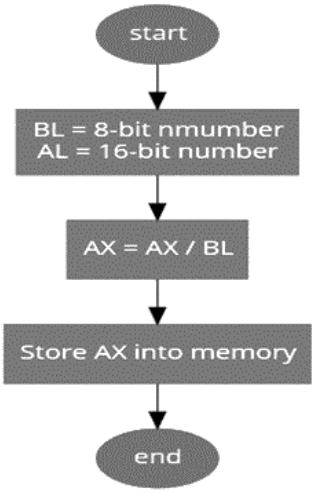
## VII. Algorithm/Flow chart

**Multiply/Divide two numbers in order to understand hexadecimal arithmetic operation.**

1. Initialize the data and code segment
2. Read the 8-bit multiplicand/Divisor
3. Read the 8-bit multiplier/Divident
4. Use MUL/DIV instruction to multiply the two 8-bit numbers.
5. Store the product.
6. End

**Flow Chart for multiplication and division:**



## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | **As per batch size** |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and their peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of given program
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, then type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……..

……

……

……..

……

……

……..

……

……

……..

## XIII. Program code with comments

**Multiplication of 8-bit numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 12H | Declaration of variable |
| | NO2 DB 34H | |
| | NO3 DW ? | |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register |
| | MOV BL, NO2 | Second number in register |
| | MUL BL | Multiply two numbers |
| | MOV NO3, AX | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

**Division of 8 bit numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 26H | Declaration of variable |
| | NO2 DB 54H | |
| | NO3 DB ? | |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV AL, NO1 | First number in register |
| | MOV BL, NO2 | Second number in register |

| | | |
|---|---|---|
| | DIV BL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

## XIV. Observations

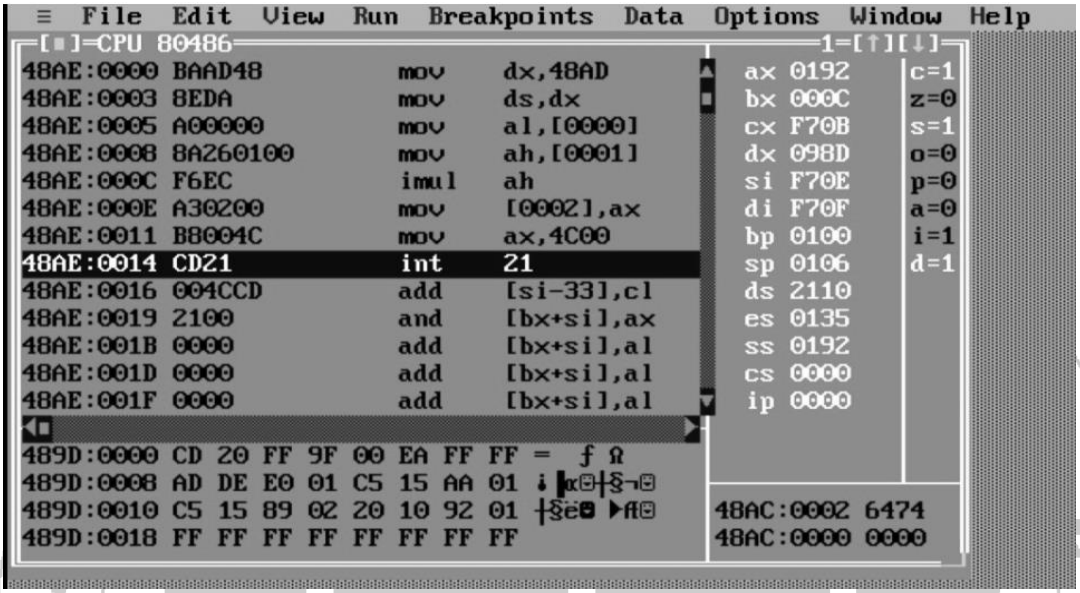



**Fig No 10.1: Output Window**

Observe and write the contents of Registers, memory location in Code Segment and Data Segment using debugger TD or Debug after the execution of the program.

| Registers | | | Flag Register | | |
|---|---|---|---|---|---|
| | After | Before | | | |
| AX | | | Carry Flag | CF | |
| BX | | | Zero Flag | ZF | |
| CX | | | Sign Flag | SF | |
| DX | | | Overflow Flag | OF | |
| SI | | | Parity Flag | PF | |
| DI | | | Auxiliary Carry Flag | AF | |
| BP | | | Interrupt Flag | IF | |
| SP | | | Direction Flag | DF | |
| DS | | | | | |
| ES | | | | | |
| SS | | | | | |
| CS | | | | | |
| IP | | | | | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| CS:0000 | | CS:0008 | |
| CS:0001 | | CS:0009 | |
| CS:0002 | | CS:000A | |
| CS:0003 | | CS:000B | |
| CS:0004 | | CS:000C | |
| CS:0005 | | CS:000D | |
| CS:0006 | | CS:000E | |
| CS:0007 | | CS:000F | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

**XV. Result(s)/Output of the program**

……..

……

……

**XVI. Conclusion and recommendation**

……

……

……

**XVII. Practical related questions**

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write the program for 16 bit multiplication.
2. Write the program for 16 bit division.
3. Write down the function of flags used in arithmetic operation.
4. Complete the following table after execution of above programs.

| For 8 bit/16 bit multiplication | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| No 1 | 25H | 2345H |
| No 2 | 18H | 6789H |
| Result | | |

| For 8 bit/16 bit Division | | |
|---|---|---|
| | 8 bit numbers | 16 bit numbers |
| No 1 | 5CH | 5678H |
| No 2 | 2AH | 1234H |
| Result | | |

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References /Suggestions for further reading


1. https://www.tutorialspoint.com/assembly_programming/
2. https://mysc.altervista.org/biginners-guide-8086/
3. https://www.geeksforgeeks.org/8086-program-divide-two-16-bit-bcd-numbers/
4. https://www.geeksforgeeks.org/8086-program-multiply-two-16-bit-numbers/

## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 11: Assembly language programming to find smallest/largest Hexadecimal numbers

### I Practical Significance

In assembly language programming, flags are affected after Compare instruction. The status of the flags can be used to make decisions about smaller or greater numbers. Students will be able to use the Compare instruction and decision making instruction to find the smallest and largest number.

### II Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

### III Course Level Learning Outcome(s)

Students will be able to achieve & demonstrate the following COs on completion of course based learning

Use 8086 microprocessor environment to build and execute assembly language programs

.

### IV Laboratory Learning Outcome(s)

Develop assembly language programming for finding smallest /largest hexadecimal numbers.

### V Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI Relevant Theoretical Background

Array is the set of N numbers i.e., byte or word. Hence, memory pointer and counter is required to read or write numbers from or to memory location in the array.

To find the smallest/largest number from the array, the numbers in the array must be compared with each other. Array may consist of 8 bit numbers i.e. byte or 16 bit numbers i.e. word, so a memory pointer is required to read numbers from the array. Also, one counter called a byte or word counter which indicates how many numbers are there in the array, is required in the program to read and compare only desired numbers from the array. In 8086, the CMP instruction is used to numeric data fields.

**CMP destination, source**

The CMP instruction compares a byte/word from the specified source with a byte/word from the specified destination. The source and the destination can be a register, immediate data or memory location

It subtracts the source operand from the destination but does not store the result anywhere.

The flags (OF, CF, PF, AF, SF, ZF) are affected depending on the result of subtraction.

Source and destination both cannot be memory locations.

**Operation Performed: --**

- If destination > source then CF = 0, ZF = 0, SF = 0
- If destination < source then CF = 1, ZF = 0, SF = 1
- If destination = source then CF = 0, ZF = 1, SF = 0

**Conditional Jump Instruction** is used after Compare instruction to check the flag status and then the desired result can be obtained about the number is smaller, greater or equal. There are many conditional Jump instructions such as-

| Instruction | Description | Condition/ Flag affected |
|---|---|---|
| JC | Jump if carry | Carry = 1 |
| JNC | Jump if no carry | Carry = 0 |
| JE/JZ | Jump if equal or Jump if zero | Zero = 1 |
| JNE/JNZ | Jump if not equal or Jump if not zero | Zero = 0 |
| JA/JNBE | Jump if Above or Not Below/Equal | CF, ZF |
| JAE/JNB | Jump if Above/Equal or Not Below | CF |
| JB/JNAE | Jump if Below or Not Above /Equal | CF |
| JBE/JNA | Jump if Below/Equal or Not Above | AF, CF |
| JG/JNLE | Jump if Greater or if not Less/Equal | OF, SF, ZF |
| JGE/JNL | Jump if Greater/Equal or if not Less | OF, SF |
| JL/JNGE | Jump if Less or if not Greater/Equal | OF, SF |
| JLE/JNG | Jump if Less /Equal or if not Greater | OF, SF, ZF |

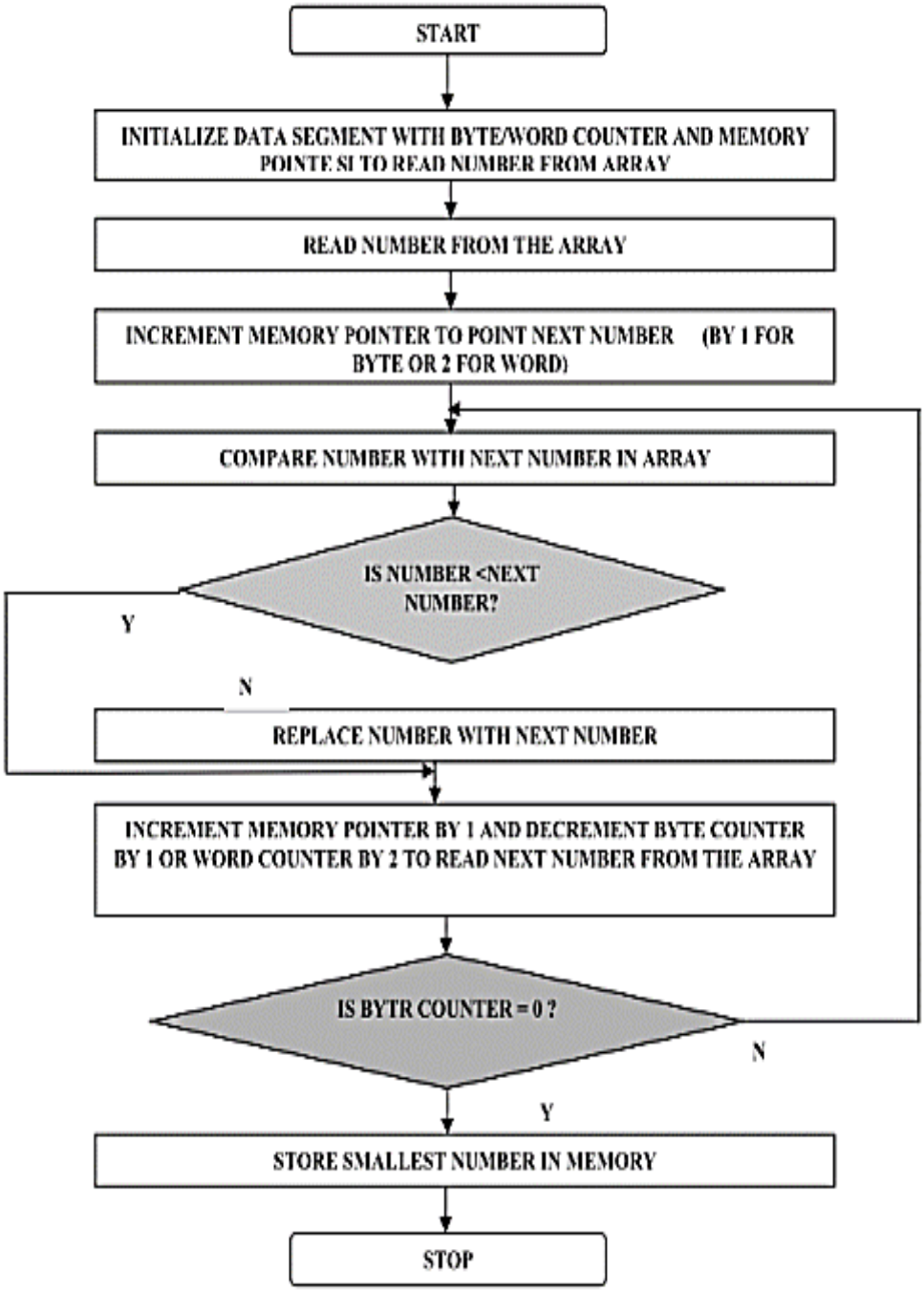
**Conditional Jump Instruction** is used to jump to a certain location/memory address after the condition is satisfied.

## VII Algorithm/Flow chart

**Algorithm for program to find smallest number**

1. Initialize the data segment with an array of numbers from which the smallest number is to be found.
2. Initialize necessary variables to store the number and the result generated after operation.
3. Perform comparison of 1st number from array with second number check which one is smaller.
4. Compare the next number with the smaller number from the previous 2 numbers.
5. Again check which one is smaller and repeat the procedure with all numbers in the array until the smallest number is found.
6. Store the final result.

**Flowchart for finding smallest number-**



**VIII Resources Required**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | **As per batch size** |
| **2** | Any Editor to write/edit programs | EDIT/ Notepad | |
| **3** | Turbo/Macro Assembler | (TASM / MASM) | |
| **4** | Turbo Linker | (TLINK/LINK5) | |
| **5** | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

**IX Precautions to be followed**

1. Handle computer system and its peripherals properly.
2. Shut down computers properly.

**X Procedure**

1. Write an algorithm and draw a flowchart of a given program. (Use blank space provided or attach more pages if needed)
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. once the assembly language program is created, hen type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

**XI Resources Used**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

**XII Actual Procedure followed**

……

……

……

……

……

……

**XIII Program code with comments**

**Sample program for finding smallest number from array of 5 numbers:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | ARRAY DB 25H,08H,37H,03H,64H | Declaration of ARRAY |
| | SMALLEST DB 00H | Declaration of variable for result |
| | DATA ENDS | |
| | | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | MOV CL, 04H | Load Counter |
| | MOV SI, OFFSET ARRAY | Load offset address of 1$^{ST}$ No of array to SI |
| | MOV AL, [SI] | Load 1$^{st}$ number to AL |
| UP | INC SI | Increment SI |
| | CMP AL, [SI] | Compare 1$^{st}$ no with 2$^{nd}$ number |
| | JC NEXT | Jump if carry to next |
| | MOV AL, [SI] | |
| NEXT | DEC CL | |
| | JNZ UP | |
| | MOV SMALLEST, AL | Store final result in memory |
| | MOV AH, 4CH | |
| | CODE ENDS | |
| | END START | |

Student should refer the above program and write an assembly language program for finding greatest number.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XIV Observations

**Content of memory location and AL register while finding smallest number**

| Address | Original Contents | Loop 1 | Loop 2 | Loop 3 | Loop 4 | Loop 5 |
|---|---|---|---|---|---|---|
| DS:0000 | **25** | **AL = ____** | **AL = ____** | **AL = ____** | **AL = ____** | **AL = ____** |
| DS:0001 | **08** | | | | | |
| DS:0002 | **37** | | | | | |
| DS:0003 | **03** | | | | | |
| DS:0004 | **64** | | | | | |

**Content of memory location and AL register while finding Greatest number**

| Address | Original Contents | Loop 1 | Loop 2 | Loop 3 | Loop 4 | Loop 5 |
|---|---|---|---|---|---|---|
| DS:0000 | **25** | **AL = ____** | **AL = ____** | **AL = ____** | **AL = ____** | **AL = ____** |
| DS:0001 | **08** | | | | | |
| DS:0002 | **37** | | | | | |
| DS:0003 | **03** | | | | | |
| DS:0004 | **64** | | | | | |

## XV Result(s)/Output of the program

……

……

……

## XVI Conclusion and recommendation

……

……

……

**XVII Practical related questions**

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write down the function of Compare instruction.
2. State the conditional jump instructions used for finding the greatest number.

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII References /Suggestions for further reading


1. https://www.tutorialspoint.com/assembly_programming/
2. https://ankurm.com/8086-assembly-program-to-find-smallest-number-from-given-numbers/
3. https://www.tutorialspoint.com/8086-program-to-find-the-min-value-in-a-given-array
4. https://www.geeksforgeeks.org/8086-program-find-min-value-given-array/


## XIX Assessment Scheme

| **Performance Indicators** | | **Weightage** |
|---|---|---|
| Process Related : 15 Marks | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

**Practical No. 12: * Assembly language programming for sorting of data**

**I. Practical Significance**

Sorting is a progress that organizes a collection of data into either ascending or descending order. This operation requires comparison of data and exchanging the position of data. Students will be able to use XCHG or MOV instruction while implementing sorting algorithms.

**II. Industry/Employer Expected Outcome(s)**

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

**III. Course Level Learning Outcome(s)**

Students will be able to achieve and demonstrate the following COs on completion of course based learning

Develop assembly language programming in 8086 to implement loops and branching instructions.

**IV. Laboratory Learning Outcome(s)**

Develop an assembly language program to Sort numbers of given arrays in ascending order.

Develop an assembly language program to Sort numbers of a given array in descending order.

**V. Relevant Affective Domain related outcome(s)**

a. Follow precautionary measures.
b. Demonstrate working as a leader/ a team member.
c. Follow ethical practices

**VI. Relevant Theoretical Background**

If numbers in an array are arranged such that every nth number is greater than $(n-1)^{th}$ number, then that array is in ascending order. If numbers in an array are arranged such that every nth number is smaller than $(n-1)^{th}$ number, then that array is in descending order. There are many sorting algorithms such as Selection sort, Insertion sort, Bubble sort, Merge sort, Quick sort. Arranging numbers involves different operations such as comparing numbers, swapping numbers depending on result of comparison, repeating comparison operation for all numbers in an array

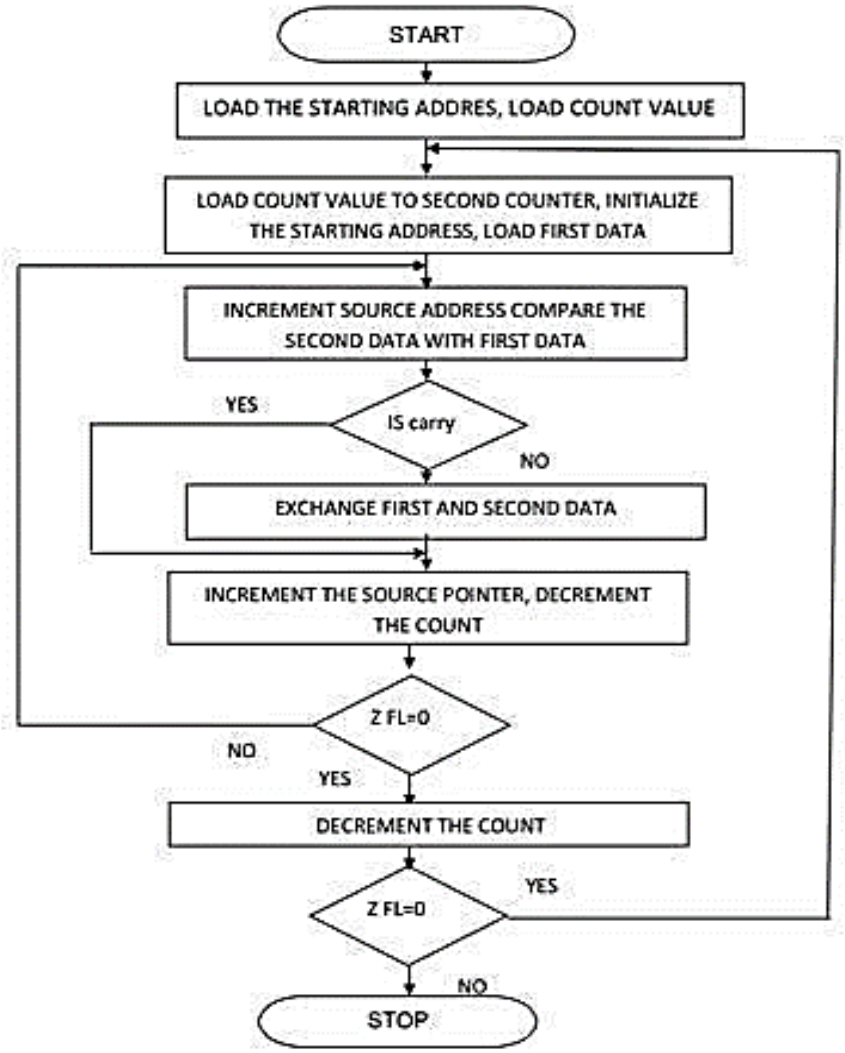
**XCHG destination, source**

This instruction exchanges the contents of a register with the contents of another register or memory location. The instruction cannot directly exchange the contents of two memory locations. A memory location can be specified as the source or as the Destination. The source and destination should both be words or they must both be byte. The segment register cannot be used in this instruction

Operation performed by XCHG instruction:

Destination ⟷ Source

## VII. Algorithm/Flow chart




## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | As per batch size |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and their peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of given program
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, then type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step.
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……

……

……

……

……

……

……

……

……

……

**XIII. Program code with comments**

Program for ascending order:

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | ARRAY DB 15H,05H,08H,78H,56H | Declaration of Variables. |
| | DATA ENDS | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE,DS:DATA | Initialization of Data segment and code segment |
| | MOV DX,DATA | |
| | MOV DS,DX | |
| | MOV BL,05H | Load counter in BL register |
| STEP1: | MOV SI,OFFSET ARRAY | |
| | MOV CL,04H | |
| STEP: | MOV AL,[SI] | Get the first number into AL register |
| | CMP AL,[SI+1] | Compare the first number with next number |
| | JC DOWN | If the number is smaller keep it as it is. |
| | XCHG AL,[SI+1] | Exchange the numbers |
| | XCHG AL,[SI] | |
| DOWN: | ADD SI,1 | Take the next number |
| | LOOP STEP | Repeat the process. |
| | DEC BL | Decrement the counter |
| | JNZ STEP1 | |
| | MOV AH,4CH | Terminate the program. |
| | INT 21H | |
| | CODE ENDS | |
| | END START | |

**Student's activity:**

Program for descending order of numbers:

| Label | Instruction code | Comments |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

**XIV. Observations**

Observe and write the contents of Registers, memory location in Code Segment and Data Segment using debugger TD or Debug after the execution of the program.

| **Registers** | | | **Flag Register** | | |
|---|---|---|---|---|---|
| | **After** | **Before** | | | |
| AX | | | Carry Flag | CF | |
| BX | | | Zero Flag | ZF | |
| CX | | | Sign Flag | SF | |
| DX | | | Overflow Flag | OF | |
| SI | | | Parity Flag | PF | |
| DI | | | Auxiliary Carry Flag | AF | |
| BP | | | Interrupt Flag | IF | |
| SP | | | Direction Flag | DF | |
| DS | | | | | |
| ES | | | | | |
| SS | | | | | |
| CS | | | | | |
| IP | | | | | |

| **Address** | **Contents** | **Address** | **Contents** |
|---|---|---|---|
| CS:0000 | | CS:0008 | |
| CS:0001 | | CS:0009 | |
| CS:0002 | | CS:000A | |
| CS:0003 | | CS:000B | |
| CS:0004 | | CS:000C | |
| CS:0005 | | CS:000D | |
| CS:0006 | | CS:000E | |
| CS:0007 | | CS:000F | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

**XV. Result(s)/Output of the program**

……

……

……

**XVI. Conclusion and recommendation**

……

……

……

## XVII. Practical related questions

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. If numbers in an array are 07H,02H,09H,10H,06H, write the array contents in each pass while arranging numbers in ascending order
2. If numbers in an array are07H,02H,09H,10H,06H, write the array contents in each pass while arranging numbers in descending order.
3. List different sorting algorithms?
4. Describe the use of XCHG instruction?

**[Space for Answers]**

……

……

……

## XVIII. References /Suggestions for further reading


1. https://www.tutorialspoint.com/assembly_programming/
2. https://www.geeksforgeeks.org/8086-program-ascnding order/
3. https://www.geeksforgeeks.org/8086-program-descending order/


## XIX. Assessment Scheme

| XIX. | **Performance Indicators** | **Weightage** |
|---|---|---|
| | **Process Related : 15 Marks** | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| | **Product Related: 10 Marks** | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| | **Total ( 25 Marks)** | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 13: Assembly language programming for transfer of block of data

### I. Practical Significance

In operating system programs such as video device drivers, memory management modules are normally written in assembly language where a memory block of large data is to be transferred from main memory to video memory continuously to display steady video on screen. In this practical students will be able to use MOV and MOVS instruction for data transfer operation in assembly language programs. Block transfer helps in improving performance and to allow prefetching of data for processing in microprocessor based systems.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve and demonstrate the following COs on completion of course based learning
Develop assembly language programming in 8086 to implement loops and branching instructions.

### IV. Laboratory Learning Outcome(s)

Develop assembly language programming for transfer of block of data

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI. Relevant Theoretical Background

Block transfer operation is transferring of a block from source memory locations to destination locations. Counter is required to perform a block transfer operation which is equal to the length of the data block. On each transfer of data from source to destination, the counter must be decremented by one and memory pointer must be incremented by one or two depending on byte or word transfer. This process is repeated till the counter becomes zero.

**Before Block Transfer**

| Source Block | | | Destination Block | |
|---|---|---|---|---|
| Memory location | Data | | Memory Location | Data |
| DS:0000H | 52H | | DS:0000H | 72H |
| DS:0000H | 79H | | DS:0000H | 39H |
| DS:0000H | 4AH | | DS:0000H | C4H |
| DS:0000H | 90H | | DS:0000H | 8DH |
| DS:0000H | F3H | | DS:0000H | 25H |

**After Block Transfer**

| Source Block | | | Destination Block | |
|---|---|---|---|---|
| Memory location | Data | | Memory Location | Data |
| DS:0000H | 52H | | DS:0000H | 52H |
| DS:0000H | 79H | | DS:0000H | 79H |
| DS:0000H | 4AH | | DS:0000H | 4AH |
| DS:0000H | 90H | | DS:0000H | 90H |
| DS:0000H | F3H | | DS:0000H | F3H |

If the number of bytes or words in block is 5, then initialize this as byte counter or word counter in CX register. Then two memory pointers are required to point source block and destination block. Hence use of SI and DI registers respectively as source and destination memory pointers. The block can be transferred from source to destination either using string instruction i.e. MOVS/ MOVSB/ MOVSW or without string instructions such as simple MOV instruction. For MOVSB/MOVSW instruction, the default memory pointer for source and destination blocks are DS:SI and ES: DI respectively. Two arrays must be declared in the array where in one array contains actual numbers and another array must be empty. To declare an empty array, we can use the DUP directive for example 5 dup (0) statements allocates five memory locations and initializes them with 0.

**VII. Algorithm/Flow chart**

**Algorithm for block transfer program**

1. Initialize the data segment with address of source and destination block.
2. Load effective address of BLOCK 1 to SI and BLOCK 2 to DI.
3. Enter data in the source block.
4. Initialize counter in CX register
5. Transfer content from source location to destination location.
6. Decrement counter, increment SI and DI
7. If counter is not equal to 0 then go to step no 5
8. Display the content and stop.

**Flow chart for Block transfer** (student should draw flow chart by referring above algorithm)

## VIII. Resources Required

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | **As per batch size** |
| **2** | Any Editor to write/edit programs | EDIT/ Notepad | |
| **3** | Turbo/Macro Assembler | ASM / MASM) | |
| **4** | Turbo Linker | LINK/LINK5) | |
| **5** | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and its peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of a given program. (Use blank space provided or attach more pages if needed)
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, then type tasm filename. asm on the command prompt and press Enter Key to create filename.obj file
6. Type Tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename .exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step.
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr.No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**XII. Actual Procedure followed**

……

……

……

……

……

……

……

……

……

……

……

**XIII. Program code with comments**

**Sample program for block transfer using simple counter:**

| Label | Instruction code | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | NO1 DB 12H,24H,45H,67H,89H | Declaration of variable |
| | NO2 DB 05H DUP(0) | |
| | DATA ENDS | |
| | CODE SEGMENT | |
| START: | ASSUME CS:CODE, DS:DATA | Initialization of data segment & code segment |
| | MOV DX, DATA | |
| | MOV DS, DX | |
| | LEA SI, NO1 | Load source memory pointer |
| | LEA DI, NO2 | Load destination memory pointer |
| | MOV CL, 05H | Load Counter in CL register |
| UP: | MOV AL, [SI] | Move 1st number from source memory to AL register |
| | MOV [DI], AL | Move 1st number from AL register to destination memory |
| | INC SI | Increment source memory pointer |
| | INC DI | Increment destination memory pointer |
| | DEC CL | Decrement counter |
| | JNZ UP | Jump up if counter is not zero |
| | MOV AH, 4CH | |
| | INT 21H | |
| | CODE ENDS | |
| | END START | |

Student should refer the above program and write for block transfer using Loop, String instructions

……..

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

**XIV. Observations**

Observe and write the content of Register, memory locations in Data Segment using debugger TD or Debug after the execution of the program.

| Types of registers | Registers | | | Flag Registers | | |
|---|---|---|---|---|---|---|
| | | Before | After | Carry flag | CF | |
| General Purpose register | AX | | | Zero flag | ZF | |
| | BX | | | Sign flag | SF | |
| | CX | | | Overflow flag | OF | |
| | DX | | | Parity flag | PF | |
| Index register | SI | | | Auxiliary Carry flag | AF | |
| | DI | | | Interrupt flag | IF | |
| Base Pointer | BP | | | Direction flag | DF | |
| Stack Pointer | SP | | | | | |
| Segment Register | DS | | | | | |
| | ES | | | | | |
| | SS | | | | | |
| | CS | | | | | |
| Instruction Register | IP | | | | | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

**XV. Result(s)/Output of the program**

…….

..……

..……

**XVI. Conclusion and recommendation**

……

……

……

**XVII. Practical related questions**

**Note: Below given are a few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifying CO.**

1. Write an operation of instruction used for initializing memory pointers.
2. State the use of Loop instruction in block transfer program.
3. Explain the instruction MOVSB and MOVSW used in the string operation.
4. Write some applications of Block transfer program.
5. Write an ALP to Block transfer in reverse order.
6. Write an ALP to Block transfer in overlapping order.

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

## XVIII. References /Suggestions for further reading

1. https://www.tutorialspoint.com/assembly_programming/
2. https://www.geeksforgeeks.org/8086-program-add-2-bcd-numbers/
3. https://www.geeksforgeeks.org/8086-program-subtract-two-16-bit-bcd-numbers/
4. https://www.geeksforgeeks.org/8086-program-multiply-two-16-bit-numbers/

## XIX. Assessment Scheme

| | Performance Indicators | Weightage |
|---|---|---|
| | **Process Related : 15 Marks** | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| | **Product Related: 10 Marks** | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| | **Total ( 25 Marks)** | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 14: Count the occurrence of a given number from a block of data

### I. Practical Significance

Tracking occurrences of any event can help in quality control and manufacturing, market analysis, research, language processing and monitoring etc. This can reduce the redundancy of any system. It can easily identify repentance of any occurrence.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:
Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve and demonstrate the following COs on completion of course based learning
Develop assembly language programming in 8086 to implement loops and branching instructions.

### IV. Laboratory Learning Outcome(s)

Apply assembly language programming logic for counting the Occurrence of a given number.

### V. Relevant Affective Domain related outcome(s)

a. Follow precautionary measures.
b. Demonstrate working as a leader/ a team member
c. Follow ethical practices.

### VI. Relevant Theoretical Background

**CMP (Compare)**

Syntax: CMP operand1, operand2

Description: The CMP training performs a subtraction among operand1 and operand2, but it does not save the result. It only updates the flags primarily based on the result of the comparison.

**Example: CMP AX, BX – Compares the content.**

**Conditional Jumps:**

**i) JC : Stands for 'Jump if Carry'**

It checks whether the carry flag is set or not. If yes, then jump takes place, that is: If CF = 1, then jump.

**ii) JNC : Stands for 'Jump if Not Carry'**

It checks whether the carry flag is reset or not. If yes, then jump takes place, that is: If CF = 0, then jump.

**iii) JE / JZ : Stands for 'Jump if Equal' or 'Jump if Zero'**

It checks whether the zero flag is set or not. If yes, then jump takes place, that is: If ZF = 1, then jump.

### VII. Algorithm/Flow chart

1. The array contains the block of data in which we want to count occurrences.

2. Array length calculates the length of the array.
3. Search value holds the number we want to count occurrences of.
4. Count is a variable to store the count of occurrences.
5. The algorithm loops through each element of the array using a loop counter (CX) and a pointer (SI).
6. Inside the loop, it compares each element with the search value. If it finds a match, it increments the count variable.
7. After looping through all elements, it displays the count using DOS interrupt 21H
8. Finally, it exits the program using DOS interrupt 21H function 4CH.

**VIII. Resources Required**

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| **1** | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | As per batch size |
| **2** | Any Editor to write/edit programs | EDIT/ Notepad | |
| **3** | Turbo/Macro Assembler | (TASM / MASM) | |
| **4** | Turbo Linker | (TLINK/LINK5) | |
| **5** | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

**IX. Precautions to be followed**

1. Handle computer systems and their peripherals properly.
2. Shut down computers properly.

**X. Procedure**

1. Write an algorithm and draw a flowchart of given program
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. Once the assembly language program is created, then type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file.
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……
……
……
……
……
……
……
……
……
……
……
……
……

## XIII. Program code with comments

| Label | Mnemonic | Comments |
|---|---|---|
| | DATA SEGMENT | |
| | ARRAY DB 10H,20H,30H,40H,50H,20H,60H ,70H,20H,80H | Declaration of variables. |
| | LEN DB 10 | Length of the array |
| | NUM DB 20H | Number to be searched for |
| | COUNT DB 0 | Count of occurrences |

| Label | Mnemonic | Comments |
|---|---|---|
| | DATA ENDS | |
| | CODE SEGMENT | |
| | ASSUME DS:DATA,CS:CODE | Initialization of Data and Code Segment |
| START | MOV AX,DATA | |
| | MOV DS,AX | Initialize data segment |
| | MOV CX,LEN | Load length of array |
| | MOV SI,OFFSET ARRAY | Load offset of array |
| | MOV AL,NUM | Load number to be searched for |
| | XOR BL,BL | Clear bl to use as a counter |
| NEXT ELEMENT: | CMP AL,[SI] | Compare al with current element |
| | JNE NOT_EQUAL | If not equal, jump to not equal |
| | INC BL | Increment counter |
| NOT_EQUAL | INC SI | Move to the next element |
| LOOP | NEXT_ELEMENT | Loop until cx becomes zero |
| | MOV COUNT,BL | Store the result in count |
| | MOV AH,4CH | Terminate the program |
| | INT 21H | |
| | CODE ENDS | |
| | END START | |

## XIV. Observations

Observe and write the contents of Registers, memory location in Code Segment and Data Segment using debugger TD or Debug after the execution of the program

| Registers | | | Flag Register | | |
|---|---|---|---|---|---|
| | **After** | **Before** | | | |
| AX | | | Carry Flag | CF | |
| BX | | | Zero Flag | ZF | |
| CX | | | Sign Flag | SF | |
| DX | | | Overflow Flag | OF | |
| SI | | | Parity Flag | PF | |
| DI | | | Auxiliary Carry Flag | AF | |
| BP | | | Interrupt Flag | IF | |
| SP | | | Direction Flag | DF | |
| DS | | | | | |
| ES | | | | | |
| SS | | | | | |
| CS | | | | | |
| IP | | | | | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| CS:0000 | | CS:0008 | |
| CS:0001 | | CS:0009 | |
| CS:0002 | | CS:000A | |
| CS:0003 | | CS:000B | |
| CS:0004 | | CS:000C | |
| CS:0005 | | CS:000D | |
| CS:0006 | | CS:000E | |
| CS:0007 | | CS:000F | |

| Address | Contents | Address | Contents |
|---|---|---|---|
| DS:0000 | | DS:0008 | |
| DS:0001 | | DS:0009 | |
| DS:0002 | | DS:000A | |
| DS:0003 | | DS:000B | |
| DS:0004 | | DS:000C | |
| DS:0005 | | DS:000D | |
| DS:0006 | | DS:000E | |
| DS:0007 | | DS:000F | |

**XV. Result(s)/Output of the program**

……

……

……

**XVI. Conclusion and recommendation**

……

……

……

……

**XVII. Practical related questions**

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. List conditional and unconditional Jump instructions used in 8086 microprocessors.
2. Write a program to count the number of 1's in a given byte.
3. List comparison instructions used in 8086.

**[Space for Answers]**

……

……

……

……

……

……

……

……

……

……

## XVIII. References /Suggestions for further reading


1. https://www.tutorialspoint.com/assembly_programming/
2. https://mysc.altervista.org/biginners-guide-8086/
3. https://www.geeksforgeeks.org/8086-program/


## XIX. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| **Marks Obtained** | | | **Dated signature of Teacher** |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |

## Practical No. 15: Implement shift and rotate instructions on given data

### I. Practical Significance

Any machine (system) works on machine language, which consists of binary numbers. In the 8086 microprocessor, we have 16-bit registers to handle our data. Sometimes, the need to perform some necessary shift and rotate operations on our data may occur according to the given condition and requirement. So, for that purpose, we have various Shift and Rotate instructions present in the 8086 microprocessor.

Rotate instructions in 8086 can be used to manipulate data in a single register, without requiring additional memory space. This can save memory space and improve overall performance. Rotate instruction can be used to swap the nibble.

Shift instructions can be used to multiply or divide a binary number with powers of 2.

### II. Industry/Employer Expected Outcome(s)

This course aims to help the student to attain the following industry identified outcomes through various teaching learning experiences:

Test digital systems by applying principles of digital techniques and microprocessors.

### III. Course Level Learning Outcome(s)

Students will be able to achieve & demonstrate the following COs on completion of course based learning

Develop assembly language programming in 8086 to implement loops and branching instructions.

### IV. Laboratory Learning Outcome(s)

Develop an assembly language program to shift given hex number to the left /right (with and without carry).

Develop an assembly language program to rotate given hex number to the left /right (with and without carry).

### V. Relevant Affective Domain related outcome(s)

1. Handle IC and Equipment carefully.
2. Follow safe practices.

### VI. Relevant Theoretical Background

**Shift** instructions move a bit string (or operand treated as a bit string) to the **right** or **left**, with excess bits discarded (although one or more bits might be preserved in flags). In **arithmetic shift left** or **logical shift left** zeros are shifted into the low-order bit. In **arithmetic shift right** the sign bit (most significant bit) is shifted into the high-order bit. In **logical shift right** zeros are shifted into the high-order bit.

**Rotate** instructions are similar to shift instructions, except that rotate instructions are circular, with the bits shifted out one end returning on the other end. Rotates can be to the left or right. Rotates can also employ an extended bit for multi-precision rotates.

## VII. Algorithm/Example

There are four Shift and Rotate instructions as stated below-

1. SHR: Shift Logical Right
2. SAR : Shift Arithmetic Right
3. SHL : Shift Logical Left
4. SAL : Shift Arithmetic Left
5. ROL : Rotate without Carry Left
6. ROR : Rotate without Carry Right
7. RCL : Rotate with Carry Left
8. RCR : Rotate with Carry Right

**Example: --**

Use of SHL instruction for Multiplication: -

MOV CL, 03; Load CL register for the count

SHL BH, CL; Shift the contents of BH register by 3 towards left

If CF = 0, BH = 04H

BH = 20H (32D) [ 04 * $2^3$ = 32 D, 20H = 32D]

Note: -- SHL can be used to multiply a number with powers of 2.

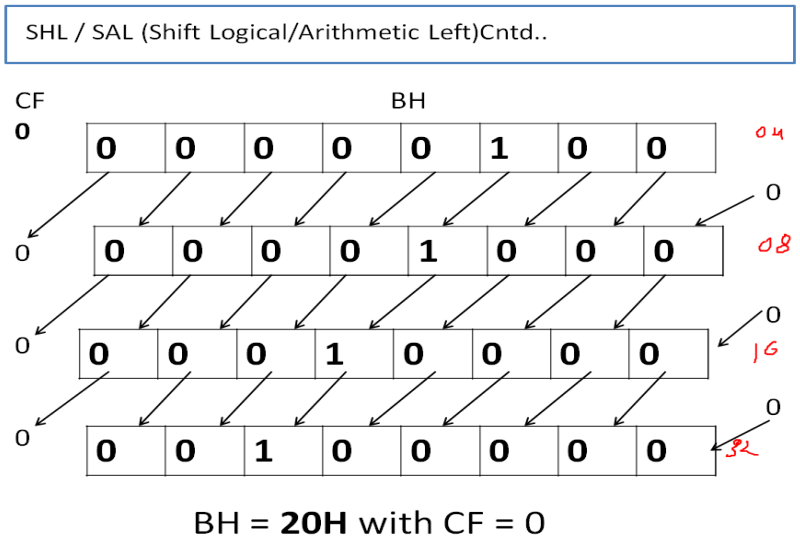



**Fig 15.1 SHL Execution**

## VIII. Resources Required

| Sr. No | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| 1 | Personal Computer | Intel Pentium Onwards Minimum 2GB RAM. 500GbyteHDD) installed with Windows 2000 onwards | As per batch size |
| 2 | Any Editor to write/edit programs | EDIT/ Notepad | |
| 3 | Turbo/Macro Assembler | (TASM / MASM) | |
| 4 | Turbo Linker | (TLINK/LINK5) | |
| 5 | Turbo Debugger | (ID/Debug), (DOSBOX utility for higher-end operating systems) | |

## IX. Precautions to be followed

1. Handle computer systems and its peripherals properly.
2. Shut down computers properly.

## X. Procedure

1. Write an algorithm and draw a flowchart of a given program. (Use blank space provided or attach more pages if needed)
2. Double click on the DOSBOX TASM 1.4 icon.
3. Type edit filename.asm on DOS prompt and press Enter Key
4. Type the program and save on disk.
5. once the assembly language program is created, hen type tasm filename.asm on the command prompt and press Enter Key to create filename.obj file
6. Type tlink filename.obj or tlink filename on command prompt and press Enter Key to create filename.exe file.
7. Finally, type debug filename.exe or td filename.exe on the command prompt and press Enter Key to debug your program step by step
8. Observe the contents of registers, memory location used and status of flags.

## XI. Resources Used

| Sr. No. | Name of Resource | Suggested Broad Specification | Quantity |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## XII. Actual Procedure followed

……

……

……

……

……

……

……

……

……

……

……

……

**XIII. Program code with comments**

Students should refer to the above example and try the following examples using Shift and Rotate instructions. Observe the content of destination register and carry flag after executing the below instructions in TASM and justify the answer using the above method.

Examples:-

1. If CF = 0, BX = E6D3H
   SAR BX, 1
2. If CF = 1, BX = C7A2H
   SHR BX, 1
3. If CF = 1, DX = 98F4H
   ROR DX, 2
4. If CF = 1, CX = B586H
   ROL CX, 2
5. If CF = 0, DX = E749H
   RCR DX, 2
6. If CF = 0, AX = 15D2H
   RCL AX, 2

**XIV. Observations:** Students should observe and write the result from destination registers

| Sr No. | Name of Register | Result |
|---|---|---|
| 1. | BX | |
| 2. | BX | |
| 3. | DX | |
| 4. | CX | |
| 5. | DX | |
| 6. | AX | |

**XIV. Result(s)/Output of the program**

……

……

……

**XV. Conclusion and recommendation**

……

……

……

**XVI. Practical related questions**

**Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.**

1. Write an instruction to rotate register BH left 4 times with carry.
2. Write an instruction to arithmetically shift the content of register AH right 5 times.
3. Write an ALP to check whether the number is POSITIVE or NEGATIVE.
4. Rotate the contents of DX to write 2 times without carry.
5. Write an ALP to check whether the number is ODD or EVEN.

**[Space for Answers]**

## XVII. References /Suggestions for further reading


1. https://www.includehelp.com/embedded-system/shift-and-rotate-instructions-in-8086-microprocessor.aspx
2. https://www.youtube.com/watch?v=0c8iKrco6nI
3. https://www.youtube.com/watch?v=b2fbAnfsBvs

## XVIII. Assessment Scheme

| Performance Indicators | | Weightage |
|---|---|---|
| **Process Related : 15 Marks** | | 60 % |
| 1 | Handling of the components | 10% |
| 2 | identification of components | 20% |
| 3 | Measuring value using suitable instrument | 20% |
| 4 | working in teams | 10% |
| **Product Related: 10 Marks** | | **40%** |
| 5 | Calculated theoretical values of given component | 10% |
| 6 | Interpretation of result | 05% |
| 7 | Conclusion | 05% |
| 8 | Practical related questions | 15% |
| 9 | Submitting the journal in time | 05% |
| **Total ( 25 Marks)** | | **100 %** |

| Marks Obtained | | | Dated signature of Teacher |
|---|---|---|---|
| **Process Related (15)** | **Product Related (10)** | **Total (25)** | |
| | | | |