SCHEME : K

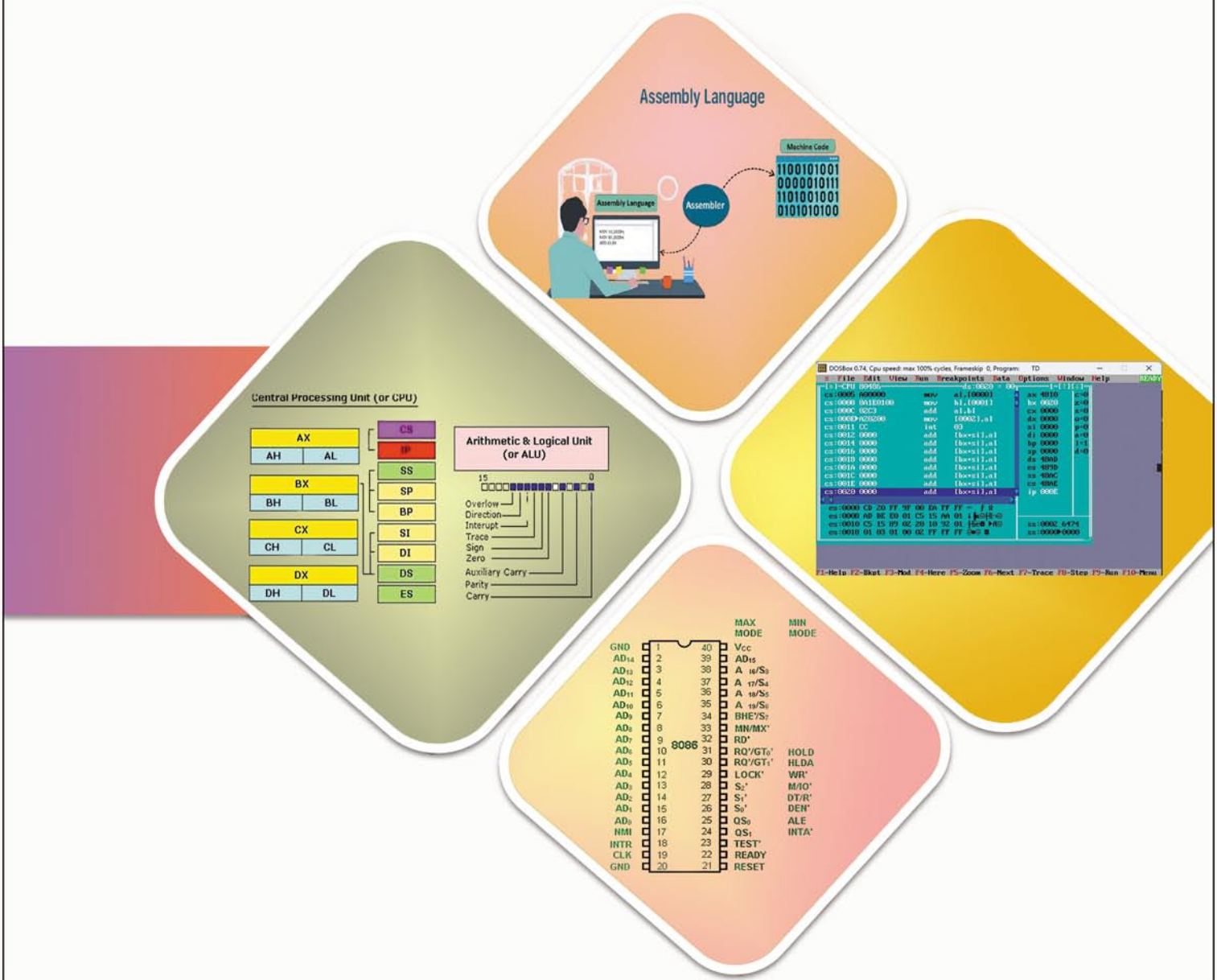
Name : _____

Roll No. : _____ Year : 20___ 20___

Exam Seat No. : _____

LABORATORY MANUAL FOR

MICROPROCESSOR PROGRAMMING -314321



COMPUTER ENGINEERING 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 Practical Manual
for

Microprocessor Programming

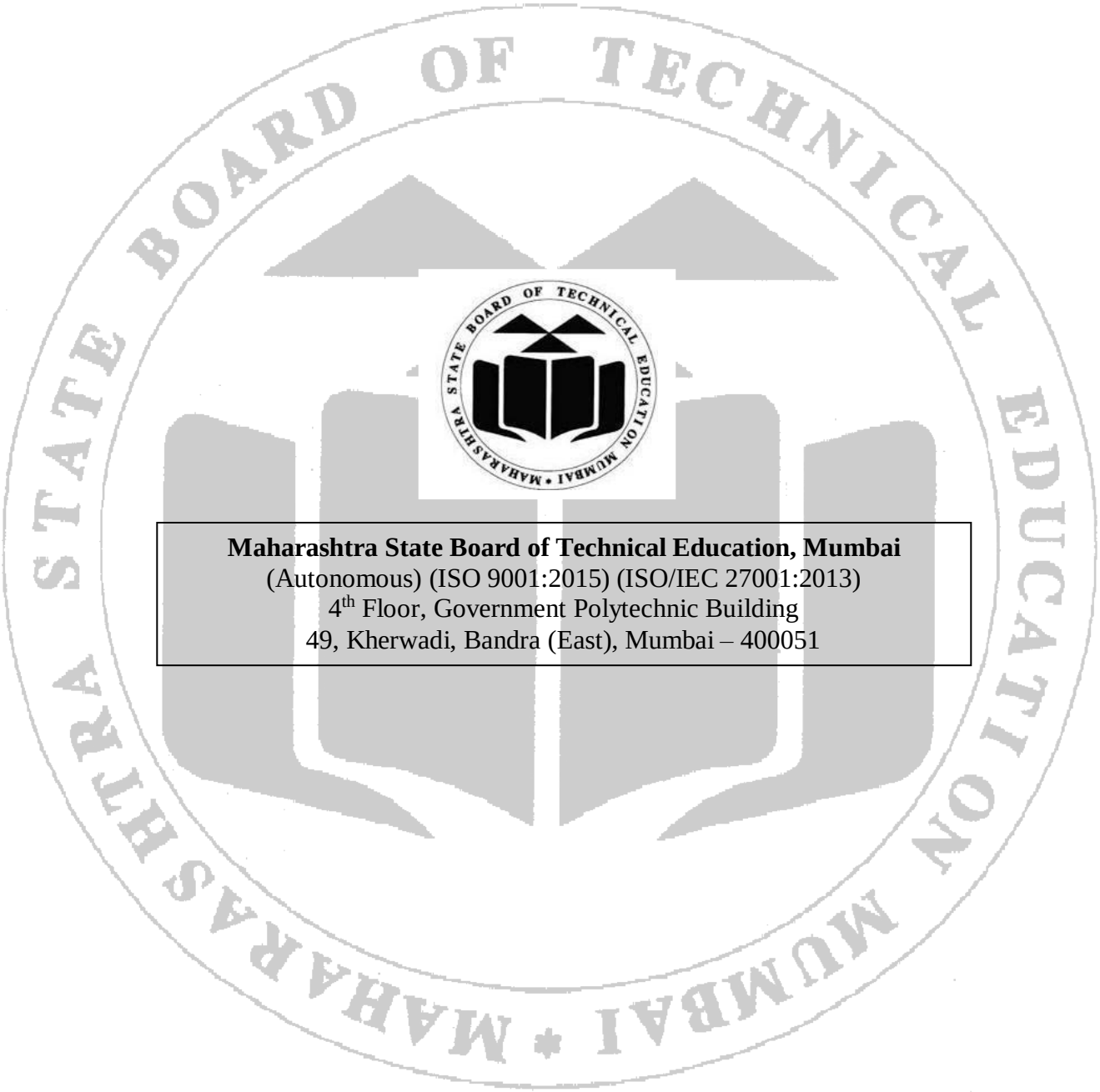
(314321)

Semester-IV

Diploma in Engineering and Technology
(CO, CM, CW, AN, AI, DS, HA)

Maharashtra State Board of Technical Education, Mumbai
(Autonomous) (ISO 9001:2015) (ISO/IEC 27001:2013)

'K' Scheme Curriculum



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

Maharashtra State Board of Technical Education

Certificate

This is to certify that Mr./Ms. …… Roll No…… of the Fourth Semester of Diploma in .....Engineering/Technology (Program Code - .....4K) of the Institute …… (Inst. Code……) has completed the practical work satisfactorily for the course Microprocessor Programming (Course Code: 314321) for the academic year 20….. – 20… .....as prescribed in the curriculum.

Place …… Enrollment No……

Date: ..... Exam Seat No. ……..

Course Teacher **Head of the Department** **Principal**

Seal of the Institute

Preface

The primary objective of any engineering laboratory or fieldwork in the technical education system is to develop essential industry-relevant competencies and skills. In line with this goal, MSBTE introduced the innovative 'K' Scheme curricula for engineering diploma programs, emphasizing outcome-based education. A significant amount of time is allocated for practical work, underscoring the importance of laboratory activities. This ensures that every teacher, instructor, and student recognizes the need to effectively utilize every minute in the lab to develop these outcomes, rather than engaging in mundane activities. Practical skills, which are difficult to acquire through traditional classroom methods, are a key focus. Hence, the 'K' scheme laboratory manual emphasizes outcomes rather than the traditional practice of conducting practical's merely to 'verify the theory,' which may become a secondary benefit.

This laboratory manual is crafted to assist all stakeholders—students, teachers, and instructors—in achieving the predetermined outcomes. Students are expected to thoroughly read the relevant practical procedure and understand the necessary theoretical background at least a day in advance. Each practical exercise in this manual starts by identifying the competency, industry-relevant skills, course outcomes, and practical outcomes, serving as key focal points. Students will become aware of the skills they will acquire through the provided procedures and necessary precautions, which will help them solve real-world problems in their professional lives.

The manual also offers guidelines for teachers and instructors to effectively facilitate student-centered lab activities. This involves arranging and managing necessary resources so that students can systematically follow procedures and precautions, ensuring the achievement of desired outcomes.

A microprocessor is a general-purpose system used in various specialized processing devices built using digital logic. Many items that were not traditionally computer-related now include microprocessors, such as household appliances, cars, car keys, tools, test instruments, and toys. A microprocessor control program can be easily customized to meet different product line needs, allowing performance upgrades with minimal product redesign. Students will learn to write assembly language code, optimize critical sections of high-level programs, implement loops at the microprocessor level using jump instructions, and utilize microprocessors to receive input from keyboards and mice through interrupts. They will also understand how a machine interprets instructions at a low level and the rationale behind memory segmentation in a process. While every effort has been made to ensure accuracy in this laboratory manual, perfection cannot be guaranteed, especially as this is the first edition. Any errors and suggestions for improvement are welcome and can be brought to our attention.

Program Outcomes (POs) to be achieved through Practical:

PO1	Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
PO2	Problem analysis: Identify and analyses well-defined engineering problems using codified standard methods.
PO3	Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
PO4	Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
PO5	Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.
PO6	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.
PO7	Life-long learning: Ability to analyses individual needs and engage in updating in the context of technological changes.

List of Relevant Skills

Following skills are crucial for students to develop a strong foundation in microprocessor technology, enabling them to apply their knowledge in real-world scenarios and in various industrial applications.

1. Analyze the functional block diagram of 8086 or x86 based processor.
2. Develop an assembly language program using assembler for given problem.
3. Use procedures and macros in assembly language programs.

Practical Course Outcome Matrix

Course Outcomes (COs)

CO1	Analyse the functional block diagram of 8086 microprocessor.
CO2	Use program development tools and assembler directives.
CO3	Use instructions in different addressing modes.
CO4	Develop an assembly language program for a given task using assembler.
CO5	Use procedures and micros to develop an assembly language program for a given problem.

Sr. No.	Title of the Experiment	CO1	CO2	CO3	CO4	CO5
1	* Identification of various blocks in 8086 microprocessor architecture.	✓				
2	* Use assembly language programming (ALP) tools and directives.		✓			
3	* ALP to perform addition and subtraction of two given numbers.			✓		
4	ALP for multiplication of two signed and unsigned numbers.			✓		
5	ALP to perform division of two unsigned and signed numbers.			✓		
6	ALP to add, subtract, multiply and divide two BCD numbers.			✓		
7	*ALP to perform block transfer operation.				✓	
8	ALP to find sum of series.				✓	
9	*ALP to find smallest and largest number from array of numbers.				✓	
10	ALP to arrange numbers in an array in ascending or descending order.				✓	
11	*ALP to find the length of string and concatenate two strings				✓	
12	ALP for string operations such as string reverse and string copy.				✓	
13	ALP to compare two strings.				✓	
14	* ALP to check a given number is odd or even.				✓	
15	ALP to check a given number is positive or negative.				✓	
16	ALP to count number of' ‘0' and '1's in a given number.				✓	
17	* ALP to perform arithmetic operations on given numbers using procedure.					✓
18	ALP to perform arithmetic operations on given numbers using macro.					✓

Guidelines to Teachers

1. Teachers should align the explanation of the topic to teaching learning outcome (TLOs).
2. Refer to laboratory learning outcome (LLOs) for the execution of the practical to focus on the defined objectives.
3. Promote life-long learning by training the students to equip themselves with essential knowledge, skills and attitudes.
4. If required, provide demonstration for the practical emphasizing on the skills that the student should achieve.
5. Teachers should give opportunity to the students for exhibiting their skills after the demonstration.
6. Provide feedback and/or suggestions and share insights to improve effectiveness.
7. Assess students' skill achievement related to COs of each unit.

Instructions for Students

1. 100% attendance is compulsory for all practical sessions.
2. Students must adhere to ethical practices.
3. Plagiarism is strictly prohibited.
4. Students should accomplish the requisites of Teamwork, Collaboration and Group Dynamics during the practical sessions.
5. Conscious practice to develop professional communication on your own in and out of class is essential to achieve the course objectives.
6. All the students must follow the schedule of practical sessions, complete the assigned work/activity and submit the assignment in stipulated time as instructed by the course teacher.
7. Follow formal attire and maintain personal appearance.

Content Page

List of Practical and Formative Assessment Sheet

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (25)	Teacher's Sign	Remark
1	* Identification of various blocks in 8086 microprocessor architecture.					
2	* Use assembly language programming (ALP) tools and directives.					
3	* ALP to perform addition and subtraction of two given numbers.					
4	ALP for multiplication of two signed and unsigned numbers.					
5	ALP to perform division of two unsigned and signed numbers.					
6	ALP to add, subtract, multiply and divide two BCD numbers.					
7	*ALP to perform block transfer operation.					
8	ALP to find sum of series.					
9	*ALP to find smallest and largest number from array of numbers.					
10	ALP to arrange numbers in an array in ascending or descending order.					
11	*ALP to find the length of string and concatenate two strings					
12	ALP for string operations such as string reverse and string copy.					
13	ALP to compare two strings.					
14	*ALP to check a given number is odd or even.					
15	ALP to check a given number is positive or negative.					
16	ALP to count number of' '0' and '1's in a given number.					

17	* ALP to perform arithmetic operations on given numbers using procedure.					
18	ALP to perform arithmetic operations on given numbers using macro.					
				Total		

***Total marks to be transferred to proforma published by MSBTE**

Note:

- '*' 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: Identification of various blocks in 8086 microprocessor

I Practical Significance

By identifying the various blocks within the 8086 microprocessor, students gain a clear understanding of how different components of the microprocessor function and interact. Knowing the functions of various blocks helps in optimizing the performance of the microprocessor in different applications. Students can learn to enhance system efficiency by leveraging specific blocks for certain tasks.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO1 - Analyze the functional block diagram of 8086 microprocessors.

IV Laboratory Learning Outcome(s)

LLO 1.1 Identify the functions of various blocks in 8086 architectures.

LLO 1.2 Identify the use of registers of 8086.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

The Bus Interface Unit (BIU):

It provides the interface of 8086 to external memory and I/O devices via the System Bus. It performs various machine cycles such as memory read, I/O read, etc. to transfer data between memory and I/O devices.

BIU performs the following functions are as follows:

- It generates the 20-bit physical address for memory access.
- It fetches instructions from the memory.
- It transfers data to and from the memory and I/O.
- Maintains the 6-byte pre-fetch instruction queue (supports pipelining).

The Execution Unit (EU):

The main components of the EU are General purpose registers, the ALU, Special purpose registers, the Instruction Register and Instruction Decoder, and the Flag/Status Register.

- Fetches instructions from the Queue in BIU, decodes, and executes arithmetic and logic operations using the ALU.
- Sends control signals for internal data transfer operations within the microprocessor. (Control Unit)
- Sends request signals to the BIU to access the external module.
- It operates with respect to T-states (clock cycles) and not machine cycles.

VII Required Resources:

Sr. No.	Name of the Resources	Specifications	Qty
1.	Chart	8086 Microprocessor Block diagram.	1 No.

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.
3. Do not insert pen drives into the laboratory computers.

IX Conclusion:

……

……

……

……

X Practical related questions

1. State the functions of ALU.

……

……

……

……

……

2. Draw flag register format of 8086.

……

……

……

……

……

3. Draw the functional block diagram of 8086 microprocessor

XI References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XII Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Identify the different components of block diagram of 8086.	30%
2. Identify the use of registers of 8086.	30%
Product related (10 Marks)	**40%**
3. Practical related questions	30%
4. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 2: Use assembly language programming (ALP) tools and directives

I Practical Significance

Assembly language is used to write program in the form of mnemonics that is the short form of operations i.e. for addition *add* and operands, which may be registers or memory location. In operating system, system program is normally written in assembly language using tools like assembler, linker and for debugging debugger. Hence, students will be able to use various such tools required for assembly language programming.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 2- Use program development tools and assembler directives.

IV Laboratory Learning outcome(s)

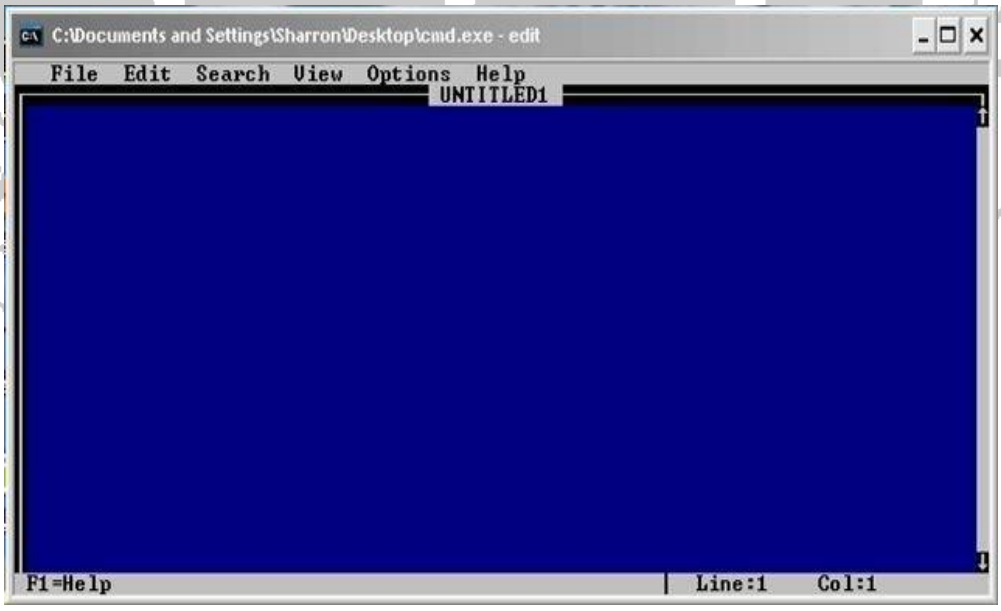
LLO.2.1. Identify the function of given assembly language tool.
LLO.2.2. Use assembler directives in a given situation.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

1. **Editor:** An editor is a program, which is used to construct assembly language program in appropriate format so that the assembler will translate it correctly to machine language. Therefore, you can type your program called as source program using editor. The **DOS** based editor such as **EDIT** can be used to type your program.

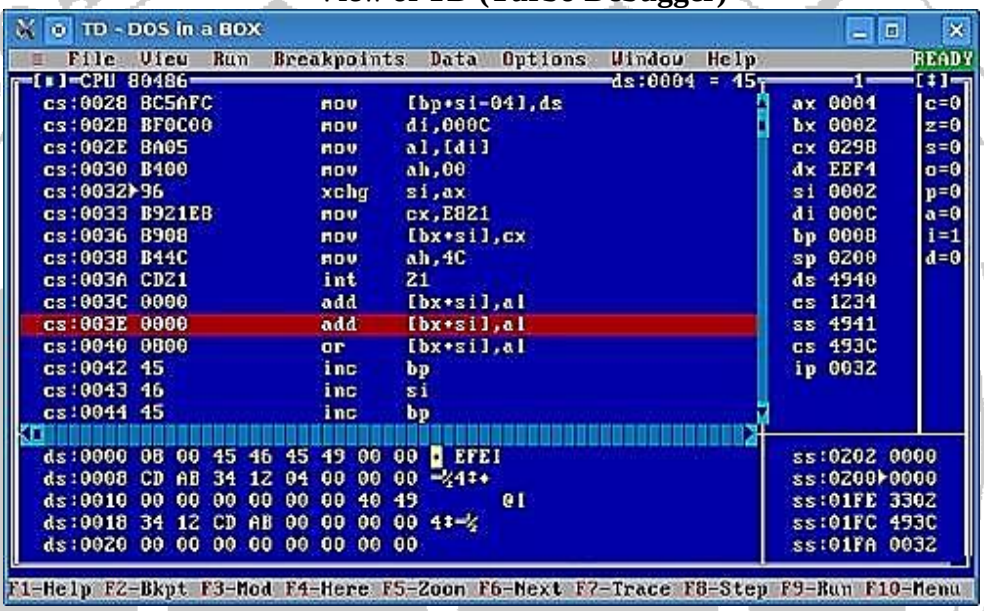


2. **Assembler:** An assembler is a program that translate assembly language program to the appropriate binary code for each instruction in program i.e. machine code and generate the file

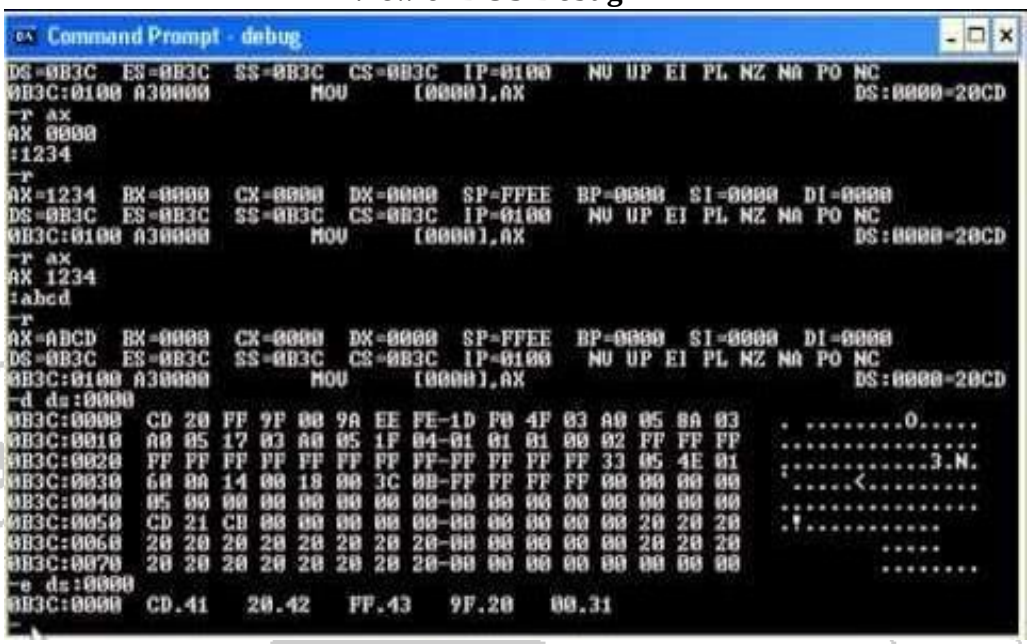
called as object file with extension **.obj**. Assembler may be TASM Borland's Turbo Assembler and MASM Microsoft Macro Assembler etc.

3. **Linker:** A linker is a program that combines, if requested, more than one separately assembled program module into one executable program and generate .exe module, and initializes it with special instructions to enable its subsequent loading the execution. Linker may be **TLINK** Borland's Turbo Linker and **LINK** Microsoft's Linker
4. **Debugger:** Debugger is a program is used to execute program in single step mode under the control of the user. The process of locating and correcting errors using a debugger is known as debugging. Some examples of debugger are DOS **Debug** command, Borland's turbo Debugger **TD**, Microsoft Debugger known as Code View **CV** etc.

View of TD (Turbo Debugger)



View of DOS Debug



VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Install DOSBOX TASM 1.4 or above.
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 2.1: Contents of Registers

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

2. Observe and write the contents of memory location in Code Segment using debugger TD or Debug

Table 2.2: Contents of memory location in Code Segment

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	

3. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 2.3: Contents of memory location in Data Segment

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	

XI Conclusion:

……

……

XII Practical related Questions

1. Write the assembly language tools used in your lab in Table 2.4.

Table 2.4: Tools Used

Sr. No.	Tools Used	Name of Tool	Version
1	Editor		
2	Assembler		
3	Linker		
4	Debugger		

2. List the files extensions that are created by the Assembler and Linker used.

……

……

3. List the program development step for assembly language programming.

……

……

……

……

4. List the assembler directives of 8086.

……

……

……

……

……

5. Describe how an assembly language program is developed and debugged using system tools.

……

……

……

……

……

……

XIII References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XIV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

Marks Obtained		
Process Related (15)	**Product Related (10)**	**Total (25)**

Practical No. 3: ALP to perform addition and subtraction of two given numbers.

I Practical Significance

In assembly language, ADD/ADC and SUB/SBB instructions are used for performing addition and subtraction operations. In operating systems, where efficiency and direct hardware access are paramount, system programs like device drivers and memory management modules are often written in assembly. Here, the ability to utilize these instructions becomes essential for implementing arithmetic operations efficiently. Therefore, familiarizing students with these instructions in assembly language programming equips them with foundational skills necessary for system-level development and optimization.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO3 - Use instructions in different addressing modes.

IV Laboratory Learning Outcome(s)

LLO 3.1 Use different addressing mode instructions in program.

LLO 3.2 Write an assembly language program for addition and subtraction using different addressing mode instruction.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

ADD *I* 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:

ADD <DEST>, <SRC>

Destination = destination + source

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

barrow [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 an immediate data and the source and destination both should not be memory operands.

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

Syntax & Operation:

SUB <DEST>, <SRC>

Destination = destination - source

SBB <DEST>, <SRC>

Destination = destination - source - CF

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double click on DOSBOX TASM 1.4 icon.
3. Type editfilename.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 tasmfilename.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 debugfilename.exe or tdfilename.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.

X Observations:

Observe and write the contents of Register using debugger TD or Debug after the execution of program 16 bit addition of two numbers.

Table 3.1: Contents of Registers

Registers			Flag Register		
	Before	**After**			
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					

Table 3.2: Contents of memory location in Code Segment

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	

Table 3.3: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……

……

……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Write an ALP for subtraction of two 16-bit numbers.

……

……

……

……

……

……

……

……

2. Write the content of AL register and status of flags after execution of following code.

```
MOY AL, 99
ADD AL, 01
```

……

……

……

……

3. Write the difference between ADD and ADC.

……

……

……

……

4. Write ALP to perform addition of two 8 bit numbers.

……

……

……

……

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 4: ALP for multiplication of two signed and unsigned numbers

I Practical Significance

In high-level language programming, the mathematical sign for multiplication (×) is used to perform arithmetic operation. However, in assembly language the mnemonics are used to perform arithmetic operation such MUL for unsigned multiplication and IMUL for signed multiplication. In operating system, system program such as device drivers, memory management modules are normally written in assembly language where addition and subtraction is required. Hence, students will be able to use MUL instruction for unsigned and IMUL instruction for signed numbers in assembly language program.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 3- Use instructions in different addressing modes.

IV Laboratory Learning outcome(s)

LLO.4.1. Write an assembly language program for multiplication of two 16 bit unsigned numbers.
LLO.4.2. Write an assembly language program for multiplication of two 16 bit signed numbers.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

MUL source

MUL is used to multiply an **unsigned** byte/word from source with an **unsigned** byte/word in the AL/AX register. The source must be any register or a memory location. When a byte is multiplied with the byte in AL, then the result is stored in AX because the result of multiplication is maximum 16 bits. When a word is multiplied with the word in AX, then the MSW of result is stored in DX and LSW of result in AX register because the result of multiplication is maximum 32-bits. If the MSB or MSW of the result is zero, then CF and OF both will be set.

Flag affected by an instruction: OF, CF and PF, AF, SF, ZF are undefined.

Operation

(a) If source is byte then AX ← AL × unsigned 8 bit source.
(b) If source is word then DX**:** AX ← AX × unsigned 16 bit source.

Examples

MUL DL	Multiply AL by DL, result in AX.
MUL BX	Multiply AX by BX, result in DX**:** AX.

IMUL source

IMUL instruction is used to multiply a **signed** byte/word from source with a **signed** byte/word in the AL/AX register. The source must be a register or a memory location. When a byte is multiplied with the byte in AL, then the result is stored in AX because the result of multiplication is maximum 16 bits. When a word is multiplied with the word in AX, then the MSB result is stored in DX and LSB in AX register because the result of multiplication is maximum 32-bits. If the magnitude of the product does not requires all the bits of the destination, the unused bits are filled with the copy of the sign bit.

Flag affected by instruction: OF, CF and PF, AF, SF, ZF are undefined.

Operation

(a) If source is byte then AX ← AL × signed 8 bit source.

(b) If source is word then DX: AX ← AX × signed 16 bit source.

Examples

IMUL DL Multiply AL by DL, result in AX.

IMUL BX Multiply AX by BX, result in DX**:** AX.

Example of multiplication of signed byte with signed word.

MOV BX, multiplier Load signed word multiplier in BX.

MOV AL, multiplicand Load signed byte multiplicand in AL.

CBW Convert Byte to Word i.e. extends sign of AL into AH.

IMUL BX Word multiplies, result in DX**:** AX

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 4.1: Contents of Registers

Registers			Flag Register		
	Before	After			
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					

2. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 4.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. Write the names of result registers of multiplication of 8/16-bits unsigned and signed numbers

……

……

2 Which instruction you have used to extend the sign of 8-bit negative number for 8bit x 16-bit multiplication.

……

……

3 State the flag affected by IMUL instruction

……

……

4 State the difference between MUL and IMUL

……

……

……

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 5: ALP to perform division of two unsigned and signed numbers.

I Practical Significance

In assembly language, the **DIV** and **IDIV** instructions are used to perform unsigned and signed division operations, respectively. These operations provide precise control over hardware resources, which is essential for system-level programming. By directly manipulating CPU registers and handling low-level operations, these instructions are crucial for developing efficient device drivers, memory management modules, and other key components of an operating system. Mastery of DIV and IDIV ensures optimized performance and reliability in system software development, where resource constraints and execution speed are critical.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO3 - Use instructions in different addressing modes.

IV Laboratory Learning Outcome(s)

LLO 5.1 Write an assembly language program for division of two unsigned numbers.

LLO 5.2 Write an assembly language program for division of two signed numbers.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

DIV source:

DIV/IDIV instruction divides an **unsigned/signed** word by an **unsigned/signed** byte during 16/8 division, and to divide **unsigned/signed** double word i.e. 32-bits by an **unsigned/signed** word during 32/16 division. The word (dividend) must be in the AX register and a byte (divisor) may 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.

Flag affected: None and OF, CF, PF, AF, SF, ZF are undefined.

Operation

a) If source is byte then

AL+- AL *I* 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: AX **MOD** unsigned 16 bit source. (Remainder)

Examples

DIV BL ; Divide word in AX by byte in BL, quotient in AL and remainder in AH. DIV NUM [BX]; Divide word in AX by byte in memory location pointer by [BX].

IDIV source:

This instruction divides a **signed** word by a **signed** byte during 16/8 division, and to divide **signed** double word i.e. 32-bits by a **signed** word during 32/16 division.

During the division of a word by a byte, the word (dividend) must be in the AX register and a byte (divisor) may in any 8-bit register or memory location. After the division operation, 8-bit quotient will be available in AL register and 8-bit remainder will available in AH register During the division of double word by word, the dividend must be in DX: AX for double word or AX for word, but source of the divisor should be a word or byte register or a memory location.

When we want to divide a byte by a byte, we must first store dividend byte in AL and fill all bits in AH with sign bit of AL using CBW instruction. When we want to divide a word by a word, we must first store dividend word in AX and fill all bits in DX with sign bit of AX using CWD instruction.

Flag affected: None and OF, CF, PF, AF, SF, ZF are undefined.

Operation

a) If source is byte then

AL+- AL *I* signed 8 bit source (Quotient)

AH +-AL **MOD** signed 8 bit source. (Remainder)

b) If source is word then

AX+- DX: AX/ signed 16 bit source (Quotient)

DX+- DX: AX **MOD** signed 16 bit source. (Remainder)

Examples

IDIV BL ; Divide a signed word in AX by a signed byte in BL, quotient in AL and remainder in AH.

IDIV NUM [BX]; Divide a signed word in AX by a signed byte in memory location pointer by [BX].

Example of division of signed byte with signed byte.

```
MOY BL,     divisor Load signed byte divisor in     BL. MOY
AL,         dividend Load signed byte dividend in AL. CBW
            Extend sign of AL into AH.
```

IDIV BH Byte division, remainder in AH and quotient in AL.

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

Observe and write the contents of Register using debugger TD or Debug after the execution of 16/8-bit unsigned division program.

Table 5.1: Contents of Registers

Registers			Flag Register		
	Before	**After**			
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					

Table 5.2: Contents of memory location in Code Segment

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	

Table 5.3: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……

……

……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Write an ALP for signed division of two 16-bit numbers.

……

……

……

……

……

……

……

……

2. Write the result of division of signed numbers you have taken in program**.**

……

……

……

……

3. Write the difference between DIV and IDIV instructions.

……

……

……

……..

4. Write an ALP to divide 16-bit signed number by 8-bit signed number

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 6: ALP to add, subtract, multiply and divide two BCD numbers

I. Practical Significance

In high-level language programming, decimal numbers system is used to perform arithmetic operation. However, microprocessor performs all arithmetic operation on binary i.e. hexadecimal numbers. In assembly language program, special instructions are required to convert arithmetic operation result of decimal numbers to appropriate result in BCD format. Hence, students will be able to used DAA and DAS instruction to perform arithmetic operation on decimal (BCD) numbers in assembly language program.

II. Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III. Course Level Learning outcome(s)

CO 3- Use instructions in different addressing modes.

IV. Laboratory Learning outcome(s)

LLO.6.1. Use DAA and DAS instructions to perform arithmetic operations on BCD numbers.
LLO.6.2. Write an ALP to perform arithmetic operations on BCD numbers.

V. Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

DAA (Decimal adjust accumulator)

DAA instruction is used to convert the result of the addition of two packed BCD numbers into a packed BCD number. DAA only works on AL register. So, DAA instruction should be used after the ADD/ADC instruction. The ADD/ADC instruction adds the two BCD number in hexadecimal format and DAA instruction convert this hexadecimal result to BCD result.

Flag affected: CF, PF, AF, SF, ZF and OF is undefined

Operation

1. If lower nibble of AL > 9 or AF = 1(Set), then AL = AL + 06.
2. If higher nibble of AL > 9 or CF = 1 (Set) , then AL = AL + 60.
3. If both above conditions are satisfied, then AL = AL + 66.

DAS (Decimal adjust after subtraction)

DAS instruction is used to convert the result of the subtraction of two packed BCD numbers to a packed BCD number. DAS instruction only works on AL register. So, DAS instruction must be used after the SUB/SBB instruction. The SUB/SBB instruction subtracts the two BCD number in hexadecimal format and DAS instruction convert this hexadecimal result to BCD result. The working of DAS instruction is given below.

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

Operation

1. If lower nibble of AL > 9 or AF = 1 then AL = AL – 06.
2. If higher nibble of AL > 9 or CF = 1 then AL = AL – 60
3. If both above conditions are satisfied then AL = AL – 66

VII. Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII. Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX. Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X. Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 6.1: Contents of Registers

Registers			Flag Register		
	Before	After			
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					

2. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 6.2: Contents of memory location in Data Segment

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	

XI. Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

……

……

……

XII. Conclusion:

……

……

XIII. Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. Write the flags used for BCD arithmetic operation.

   ……

   ……

2. Write the instructions that converts the result of addition and subtraction in unpacked decimal digits.

   ……

   ……

3. Write an ALP to multiply the two BCD numbers stored in BL and CL register

   ……

   ……

4. Write an output of DAA instruction in AL register of following code after the execution and also the status of CF and AF.

```
MOV AL, 99 H
MOV BL, 01 H
ADD AL, BL
DAA
```

   ……

   ……

   ……

   ……

   ……

   ……

5. Write an output of DAS instruction in AL register of following code after the execution and also the status of CF and AF.

```
MOV AL, 03 H
MOV BL, 07 H
SUB AL, BL
DAA
```

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 7: ALP to perform block transfer operation.

I Practical Significance

In assembly language programming, block transfer operations are essential for moving data blocks between memory locations. These operations are crucial in systems programming, particularly in scenarios demanding high efficiency, such as embedded systems or operating system kernels. Operating systems frequently utilize block transfers to manage memory effectively. Additionally, block transfers facilitate communication with peripheral devices. In high-performance applications, such as graphics processing or scientific computing, optimizing block transfers can enhance cache utilization and memory bandwidth, thereby boosting overall system performance.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO4 - Develop an assembly language program for a given task using assembler.

IV Laboratory Learning Outcome(s)

LLO 7.1 Implement loop in assembly language program.

LLO 7.2 Use string instruction to perform block transfer operation.

LLO 7.3 Write an ALP to perform block transfer data without using string instruction.

LLO 7.4 Write an ALP to perform block transfer data with using string instruction

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

Block transfer operation is nothing, but it is transferring of block of date from source memory locations to destination memory locations. Counter is required to perform block transfer operation which is equal to length of data block. On each transfer of data from source to destination 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	56H		DS:0005H	15H
DS:000IH	7BH		DS:0006H	49H
DS:0002H	62H		DS:0007H	F7H
DS:0003H	23H		DS:000SH	C9H
DS:0004H	AAH		DS:0009H	55H

After Block Transfer

Source Block	
Memory Location	Data
DS:0000H	56H
DS:000IH	7BH
DS:0002H	62H
DS:0003H	23H
DS:0004H	AAH

Destination block	
Memory Location	Data
DS:0005H	56H
DS:0006H	7BH
DS:0007H	62H
DS:000SH	23H
DS:0009H	AAH

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 SI and DI registers respectively as source and destination memory pointers. The block can be transfer from source to destination either using string instruction i.e. MOVS/MOVSB/MOVSW or without using string instruction such as simple MOY 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 empty array, we can use **DUP** directive. For example, 5 dup (0) statements allocates five memory location and initialize them with 0.

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double click on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

Observe and write the contents of Register using debugger TD or Debug after the execution of program.

1) Table 7.1: Observe and write the contents of Source and destination block memory location before transfer

Source Memory Block		Destination Memory Block	
Address	**Contents**	**Address**	**Contents**
DS:0000		DS:0005	
DS:0001		DS:0006	
DS:0002		DS:0007	
DS:0003		DS:0008	
DS:0004		DS:0009	

2) Table 7.2: Observe and write the contents of Source and destination block memory location after transfer

Source Memory Block		Destination Memory Block	
Address	**Contents**	**Address**	**Contents**
DS:0000		DS:0005	
DS:0001		DS:0006	
DS:0002		DS:0007	
DS:0003		DS:0008	
DS:0004		DS:0009	

Table 7.3: Contents of memory location in Code Segment

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	

Table 7.4: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……

……

……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Write an ALP for Transfer of block of 16-bit numbers.

……..

……..

……..

……..

2. State the meaning of movsb/movsw instruction.

……..

……..

……..

3. Write the instructions you have used to initialize memory pointer for source and destination block of data.

……..

……..

……..

4. Write an ALP to perform block transfer in reverse order.

……..

……..

……..

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 8: ALP to find sum of series

I Practical Significance

In some industrial applications of assembly language programming, it is required to repeat group of instructions for specific number of times such as providing time delay while generating waves such as square, triangular, saw tooth etc. Students will be able to implement loop by using variants of Jump instructions.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 4- Develop an assembly language program for a given task using assembler.

IV Laboratory Learning outcome(s)

LLO.8.1. Implement loop in assembly language program to find sum of series.

LLO.8.2 Write an assembly language program to find sum of series of n Hexadecimal numbers

LLO.8.3. Write an assembly language program to find sum of series of n BCD numbers.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

The addition of the numbers in the series or array of n numbers which are stored in the memory is called as sum of series. So, byte or word counter which indicate length of series is required to read numbers from the series one by one. The result of addition may be greater than either 8 bit or 16 bit depending on numbers stored in the array.

Loop Instructions

Instruction	Action
LOOP Label	CX= CX-1 ; if (CX < > 0) jump to label
LOOPZ/LOOPE Label	CX= CX-1 ;if (CX <> 0) AND (ZF=1) jump to target
LOOPNZ/LOOPNE Label	CX= CX-1 ;if (CX <> 0) AND (ZF=0) jump to target
JCXZ label	CX= CX-1; if CX=0 jump to target

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available

4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 8.1: Contents of Registers

Registers			Flag Register		
	Before	**After**			
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					

2. Observe and write the contents of memory location in Data Segment after the execution of program.

Table 8.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions?

(Use blank space provide for answers or attached more pages if needed)

1. Note down the registers used as memory pointer and counter in a program

……

……

2. State the use of INC instruction in your program.

……

……

3. What is the condition to terminate loop formed using LOOP instruction.

……

……

4. Write applications where loop instruction can be used?

……

……

5. Which register is used as a counter to store count for a LOOP instruction?

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 9: ALP to find smallest and largest number from array of numbers.

I Practical Significance

In assembly language programming, flags are affected after compare instruction. These flags can then be used to determine whether a number is smaller or greater. By using the CMP instruction along with decision-making instructions, students can learn to find the smallest and largest numbers in an array.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO4 - Develop an assembly language program for a given task using assembler.

IV Laboratory Learning Outcome(s)

LLO 9.1 Implement loop in assembly language program to find smallest and largest number from the array of n numbers.

LLO 9.2 Use decision making branching instruction to find smallest or largest number.

LLO 9.3 Write an assembly language program to find smallest number from the array of n numbers.

LLO 9.4 Write an assembly language program to find largest number from the array of n numbers.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical 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 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 memory pointer is required to read numbers from the array. Also, one counter called as 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 compare two numeric data fields.

CMP destination, source

The CMP instruction compares a byte/word from the specified source and a byte/word from the specified destination. The source and destination can be an immediate data, a register or a memory location. However, the source and the destination should not both be memory locations. The comparison is actually done by non-destructive subtraction of the source byte or word from the destination byte or word i.e. the source and the destination will not change, but the flags will affect to specify the results of the comparison.

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

Condition	CF	ZF	SF	Meaning of flag status
AX=BX	0	1	0	Source and destination operands are equal
AX>BX	0	0	0	Destination operand is greater than source operand
AX<BX	1	0	1	Destination operand is smaller than source

Conditional Jump instruction is used to jump to certain location/memory address, after condition is satisfied

Symbol/ Instruction	Description	Flags affected
JE/JZ	Jump if Equal or Jump if Zero	ZF
JNE/JNZ	Jump if not Equal or Jump if Not Zero	ZF
JA/JNBE	Jump if Above or Jump if Not Below/Equal	CF,ZF
JAE/JNB	Jump if Above/Equal or Jump if Not Below	CF
JB/JNAE	Jump if Below or Jump if Not Above/Equal	CF
JBE/JNA	Jump if Below/ Equal or Jump if Not Above	AF,CF
JG/JNLE	Jump if Greater or Jump if not Less/Equal	OF,SF,ZF
JGE/JNL	Jump if Greater/Equal or Jump if not Less	OF,SF
JL/JNGE	Jump if Less or Jump if not Greater/Equal	OF,SF
JLE/JNG	Jump if Less/Equal or Jump if not Greater	OF,SF,ZF
JC	Jump if Carry	CF
JNC	Jump if not Carry	CF

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.

2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

Observe and write the contents of Register using debugger TD or Debug after the execution of program

Table 9.1: Contents of memory location and AL register while finding smallest number

Address	Original Contents	Loop 1	Loop 2	Loop 3	Loop4	Loop 5
DS:0000	**12**	**AL=**	**AL=**	**AL=**	**AL=**	**AL=**
DS:0001	**07**					
DS:0002	**25**					
DS:0003	**18**					
DS:0004	**02**					

Table 9.2: Contents of memory location and AL register while finding largest number

Address	Original Contents	Loop 1	Loop 2	Loop3	Loop 4	Loop 5
DS:0000	**12**	**AL=**	**AL=**	**AL=**	**AL=**	**AL=**
DS:0001	**07**					
DS:0002	**25**					
DS:0003	**18**					
DS:0004	**02**					

XI Results (Program code with output)

XII Conclusion:

……
……
……
……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Which instructions are used to make decision to find smallest/largest number in the program.
   ……
   ……
   ……
   ……
   ……
   ……
2. Show flag status after comparisons of following operands.

Nl	N2	CMPN1,N2			Nl	N2	CMPN2,Nl		
		CF	ZF	SF			CF	ZF	SF
25	45				75	36			
75	43				23	87			
234	234				100	100			

3. Provide examples of conditional jump instructions used to find the smallest and largest numbers in an array and explain how they work.
   ……
   ……
   ……
   ……
   ……
   ……
4. Write ALP to find smallest number from array of 5 16 bit numbers using loop instruction.
   ……
   ……
   ……

……
……
……

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 10: ALP to arrange numbers in an array in ascending or descending order

I Practical Significance

Sorting is a process that organizes a collection of data into either ascending or descending order. This operation requires comparison of data and exchange the position of data depending on result of comparison. There are different algorithms for sorting data. Students will be able to use XCHG or MOV instruction while implementing sorting algorithms.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 4- Develop an assembly language program for a given task using assembler.

IV Laboratory Learning outcome(s)

LLO.10.1. Apply iterative method to arrange numbers in array in ascending or descending order.
LLO.10.2. Write an assembly language program to arrange numbers in array in ascending order.
LLO.10.3. Write an assembly language program to arrange numbers in array in descending order.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

If numbers in an array are arranged such that every n^{th} 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 n^{th} 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 word or they must both be byte. The segment register cannot be used in this instruction.

Operation performed by XCHG instruction: Destination ↔ Source

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Install DOSBOX TASM 1.4 or above.
2. Double clip on 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.

X Observations

1. Observe and write the contents of memory location after the execution of program.

Table 10.1: Contents of memory location in ascending order operation

Address	Original Contents	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
DS:0000	**10**					
DS:0001	**06**					
DS:0002	**23**					
DS:0003	**15**					
DS:0004	**01**					

2. Observe and write the contents of memory location after the execution of program.

Table 10.2: Contents of memory location in descending order operation

Address	Original Contents	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5
DS:0000	**10**					
DS:0001	**06**					
DS:0002	**23**					
DS:0003	**15**					
DS:0004	**01**					

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. Explain use of XCHG instruction

……

……

2. Which sorting algorithm is used in your program ?

……

……

3. If numbers in an array are 08H,03H,20H,16H,01H write the array contents in each pass while arranging numbers in ascending order.

……

……

……

……

……

……

4. If numbers in an array are 08H, 03H, 20H, 16H ,01H write the array contents in each pass while arranging numbers in descending order. .

……
……
……
……
……
……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 11: ALP to find the length of string and concatenate two strings

I Practical Significance

A string is a sequence of characters enclosed in quotes. In various applications, it is required to display messages, obtain input from users, search for specific characters or words within a string, arrange characters in a particular order, and combine different strings. By learning to perform these operations, students will be able to manipulate strings efficiently and effectively in various programming contexts.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO4 - Develop an assembly language program for a given task using assembler.

IV Laboratory Learning Outcome(s)

LLO 11.1 Write an assembly language program to find length of string

LLO 11.2 Write an assembly language program to concatenate two strings.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

The string consists of either numbers or characters. In assembly language programming, the string must be declared in single quotes i.e. ' ' and must end with '$' sign. The data type of the string is always byte because assembler store 8 bit ASCII value of every character of string in memory.

For Example

Dept db 'Computer Engineering$'

Assembler stores string characters in memory at consecutive memory locations. Hence to perform any string related operation such as comparison, length, reverse etc., the memory pointer and byte counter is required.

Without byte counter, the string operations are possible. For that, you have to read character from string array and compare it with '$'. If character is not '$ ', then character is string character. If character is '$', then it is end of string.

Length of String:

To find the length of the string, we need one length counter and initialize memory pointer to read character from the string. Read character from the array and compare it with '$' which indicate end of the string. If the character is not '$' then increment length counter else stop reading character from the string.

Concatenation of two Strings:

The concatenation of two strings means merging of second string in first string. For example, suppose, 'Computer' and 'Department' are two separate strings, after concatenation string will become 'Computer Department'.

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

Table 11.1: Length of string

	Example 1	Example 2
Input String Taken	**Microprocessor**	----------
Length of string		

Table 11.2: String concatenation

	Example 1	Example 2
Input string 1 taken	**Microprocessor**	_____
Input string 2 taken	**Programming**	_____
Output string		

XI Results (Program code with output)

XII Conclusion:

……
……
……
……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. State the name of register which is used as counter for length of string?

……
……
……

2. State the registers that are used as memory pointers in string concatenation program.

……
……
……

3. Write an ALP to perform string copy operation.

……
……
……
……
……

……

……

……

……

4. State the meaning of '$'sign in string.

……

……

……

……

……

……

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 12: ALP for string operations such as string reverse and string copy

I Practical Significance

String is a sequence of characters enclosed in quotes. In various applications it is required to display messages to get input from user, search particular character/word in string, arrange characters in string, combine different strings. Student will be able to perform different operations on string.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 4- Develop an assembly language program for a given task using assembler.

IV Laboratory Learning outcome(s)

LLO.12.1. Write an assembly language program to copy string.
LLO.12.2. Write an assembly language program to copy string in reverse order.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

The string consists of either numbers or characters. In assembly language programming, the string must be declared in single quotes i.e. ' ' and must end with '$' sign. The data type of the string is always byte because assembler store 8 bit ASCII value of every character of string in memory.

For Example

```
dept    db    'MSBTE$'
```

Assembler stores string characters in memory at consecutive memory locations. Hence to perform any string related operation such as comparison, length, reverse etc., the memory pointer and byte counter is required

Without byte counter, the string operations are possible. For that, you have to read character from string array and compare it with '$'. If character is not '$', then character is string character. If character is '$', then it is end of string.

String in reverse order:

The memory pointer and length counter is required to read string and then copy string in another blank string variable in reverse order. To reverse the string, first find out the length of the source string, then add this value to memory pointer register to point last character of the source string. Now copy last character from source string to first character position of destination blank string. Perform this operation continuously till first character of the source string gets transfer to destination string by decrementing memory pointer for source string and incrementing memory pointer for destination string

String copy

Source string copies of each character to the destination string. The null character is used to determine the end of string. Copy first character from source string to first character position of destination blank string. Perform this operation continuously till last character of the source string

gets transfer to destination string by decrementing memory pointer for source string and incrementing memory pointer for destination string.

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Install DOSBOX TASM 1.4 or above.
2. Double clip on 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.

X Observations

Table 12.1: Reverse of a string

	Example 1	Example 2
Input string taken	**COMPUTER**	_____
Reverse string		

Table 12.2: Copy of string

	Example 1	Example 2
Input string taken	**Microprocessor**	_____
Copy of string		

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. How to reverse a string in Assembly language 8086?

……

……

2. State the registers that are used as memory pointers in string reverse program

……

……

3. State the role of direction flag while using string instructions?

……

……

4. What is advantage of using string instructions over normal instructions.

……

……

……

……

……

……

5. Write an ALP to find the string is palindrome or not.

……

……

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 13: ALP to compare two strings.

I Practical Significance

Comparing two strings in assembly language typically involves checking each character in both strings to determine if they are identical or if one is greater than the other. This process can be done using various assembly instructions to iterate through the strings, compare each character, and handle the results.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO4 - Develop an assembly language program for a given task using assembler.

IV Laboratory Learning Outcome(s)

LLO 13.1 Write an assembly language program to compare two strings without string instruction.

LLO 13.2 Write an assembly language program to compare two strings using string instruction.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

The string consists of numbers or characters. In assembly, the string must be declare in single quotes i.e. ' ' and end with '$' sign. The data type of the string is always byte type because assembler store ASCII value of each and every character of string in memory, as ASCII codes are 8 bit.

For Example: name db 'MSBTE$'

Assembler stores string characters in memory at consecutive memory locations. Hence to perform any string related operation such as comparison, length, reverse etc., the memory pointer and byte counter is required as same as used in block transfer program. For comparison, the string instruction is CMPSB as data type of string is byte. Simple CMP instruction also can be used to compare two strings. First, we will have to find the length of both strings, if lengths are equal, then strings may be equal or unequal. Then, we will have to compare both strings character by character to find equality.

String stored at Consecutive Memory locations

Source Block	
Memory Location	Data
DS:0000H	M
DS:000lH	S
DS:0002H	B
DS:0003H	T
DS:0004H	E
DS:0005H	'$'

After the comparison of two string, we need DOS function 09H of interrupt 21H to display string such as "Strings are same$" or "Strings are not same$".

Function: 09H of INT 21H (Display Strings on Console) Function Call with
Example: AH= 09H
DS: DX = Segment: Offset of string
MOV AH,09H
MOV DX, offset STR
INT 21H

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

Table 13.1: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……
……
……
……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Write the instructions you have used to initialize memory pointer for source and destination strings.

……
……
……

2. Write the string instruction used for string comparison in your program.

……
……
……

3. State the use of REP prefix instruction

……
……
……

4. Write the flag which are used to know whether strings are equal or unequal

……
……
……

5. Write an assembly language program to compare two strings using string instruction.

……
……
……
……
……
……

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 14: ALP to check a given number is odd or even

I Practical Significance

Decimal or hexadecimal numbers consists of Odd as well as Even numbers. Most of the times, it is required to check number is odd or even such as odd or even parity used in serial communication. Hence, students will be able to check and count odd and even numbers in array using assembly language program.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086.

III Course Level Learning outcome(s)

CO 4- Develop an assembly language program for a given task using assembler.

IV Laboratory Learning outcome(s)

LLO.14.1. Use div and rotate instructions to check the given number is odd or even.

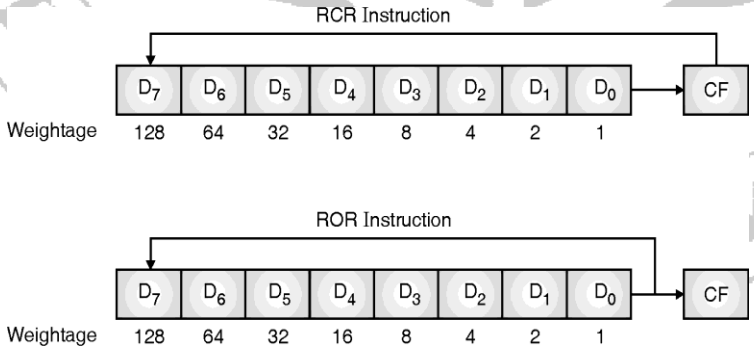
LLO.14.2. Write an assembly language program to count odd and even from the array of n numbers.

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

In 8 bit or 16-bit number, the D_0 bit is used to decide the given number is odd or even because the weightage of D_0 bit in decimal is 1 i.e. odd value and the weightage of D_1, D_2 …. D_{15} bits are 2, 4, 8… i.e. even value in 8 bit or 16-bit number. When two even or odd numbers are added, then result is always even, but when odd number is added with even number, then result is always odd. Hence, when D_0 bit of any number is 1, then that number is odd and if 0 then number is even. To test any number for odd or even, check D_0 bit of that number. To check D_0 bit of any number, rotate the bits of that number toward left by 1 bit using rotate instruction i.e. ROR or RCR as shown as follows:



Then D_0 bit goes to the carry flag, hereafter by checking carry flag, number can be tested for odd or even.

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 14.1: Contents of Registers

Registers			Flag Register		
	Before	After			
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			

2. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 14.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. Write the flag used to check whether the number is ODD or EVEN.

……

……

2 Which bit of 8/16-bit number is used to decide if number is odd or even?

……

……

3 Write an ALP to count odd as well as even numbers in array of 10 numbers.

……

……

……

……

……

……

……

……

4 Write an ALP to add the all even numbers in array of 10 numbers.

……

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 15: ALP to check a given number is positive or negative.

I Practical Significance

In the second complement format, a signed hexadecimal number's most significant bit (MSB) indicates its sign. If the MSB is 0, the number is positive; if the MSB is 1, the number is negative. Consequently, students can easily determine or count the positive and negative numbers in an array by using an assembly language program. This approach enables efficient processing and analysis of numerical data within arrays, leveraging the inherent characteristics of the second complement format for accurate sign determination.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO4 - Develop an assembly language program for a given task using assembler.

IV Laboratory Learning Outcome(s)

LLO 15.1 Use rotate instructions to check the given number is positive or negative.

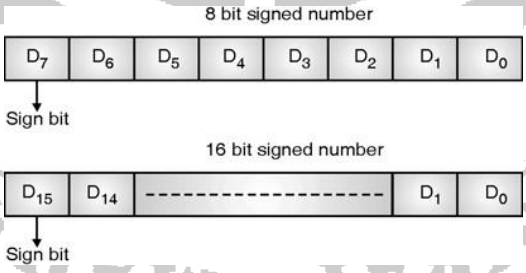
LLO 15.2 Write an assembly language program to count positive and negative numbers in given array.

V Relevant Affective Domain related Outcomes

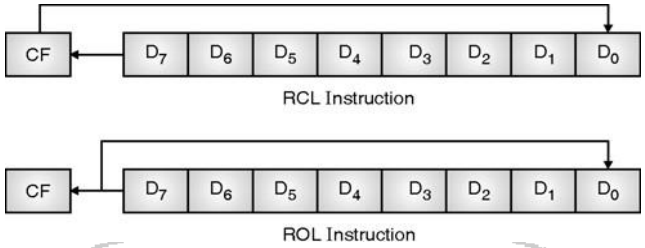
1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

The most significant bit (MSB) i.e. D_7 or D_{15} in 8 bit or 16-bit signed magnitude number indicate sign of the number i.e. D_7 or D_{15} as shown Fig. given below



Hence, by checking most significant bit, we can find out a byte or word is positive or negative number. Most significant bit i.e. D_7 or D_{15} for byte or word can be checked using either ROL or RCL instruction as given **in** Fig. given below.



The program for checking odd or even number can be used by replacing **ROR** or **RCR** instruction with **ROL or RCL** instruction to check either number is positive of negative.

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

1. Observe and write the contents of Register using debugger TD or Debug after the execution of program.

Table 15.1: Contents of Registers

Registers			Flag Register		
	Before	After			
AX			Carry Flag	CF	
BX			Zero Flag	ZF	
EX			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					

2. Observe and write the contents of memory location in Data Segment after the execution of program.

Table 15.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……
……
……
……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Write the flag which is used to check whether the number is Positive or Negative.

……
……
……

2. Which bit of 8-bit and 16-bit number is used to decide if the number is Positive or Negative?

……
……
……

3. Write an ALP to count Positive as well as Negative numbers in array of 10 numbers.

……
……
……

4. Write an ALP to add the all Positive numbers in array of 10 numbers.

……
……
……

5. Differentiate between shift and rotate instructions.

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

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 16 : ALP to count number of' '0' and '1's in a given number

I Practical Significance

In microprocessor based automation, the sensors output is connected to ports of microprocessor based system. Each sensor gives output on the corresponding pin of the input port. Microprocessor reads the contents of port i.e. all pins and copy it into the internal register. So, each sensor output can be checked by rotating the content of register toward left or right and find out the status of sensor connected to port pin. Hence, students will be able to check or count '0's and '1's in given numbers using assembly language program.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 4- Develop an assembly language program for a given task using assembler.

IV Laboratory Learning outcome(s)

LLO.16.1. Use rotate instructions to count '0' and '1' in the given number.

LLO.16.2. Write an assembly language program to count number of' '0' and '1's in a given number

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

The total numbers of 1's or 0's can be count in any number by rotating that number toward right or left by either 8 times for 8-bit number or 16 times for 16-bit number.

ROR or RCR or RCL or ROL instruction can be used to rotate any number to check how many ones or zeros are in the numbers.

When we rotate number once to left or right, corresponding bit i.e. D_0 or D_7 initially goes to carry flag, then we can check carry flag by using JNC or JC to count numbers of ones or zeros.

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 16.1: Contents of Registers

Registers			Flag Register	
	Before	**After**		
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				

2. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 16.2: Contents of memory location in Data Segment

Number 8-bit/16-bit in Hexadecimal	Nos. of '1's	Nos. of '0's
AA		
55		
88		
99		

FFFF		
AA55		
F0F0		
9898		

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. Write the flag used to count '1's and '0's

……

……

2. Write the instructions used in your program to rotate and check numbers of '0' or '1'.

……

……

3. Explain ROR and RCL in detail

……

……

……

……

4. Explain JNC and JC in detail

……

……

……

5 Write appropriate instruction for Rotate the content of DX to write 2 times without carry.

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 17: ALP to perform arithmetic operations on given numbers using

I Practical Significance

In assembly language programming, repetitive groups of instructions can be encapsulated into subprograms, subroutines, or procedures. This technique not only promotes code reuse but also enhances the program's structure and readability. Procedures enable programmers to write cleaner, more efficient code, making it easier to maintain and update. Consequently, students will be able to write and utilize procedures in their assembly language programs, fostering a deeper understanding of modular programming principles and improving their overall coding proficiency. This modular approach also facilitates collaboration among multiple developers, as each can work on separate modules without interfering with others' code.

II Industry / Employer Expected Outcome(s)

1. Develop assembly language programs using 8086.

III Course Level Learning Outcome(s)

CO5 - Use procedures and micros to develop an assembly language program for a given problem.

IV Laboratory Learning Outcome(s)

LLO 17.1 Use CALL and RET instructions to call procedures using different parameter passing methods.

LLO 17.2 Use assembler directives: PROC and ENDP to write the procedure.

LLO 17.3 Write an assembly language program using procedure to perform for addition, subtraction, multiplication and division.

LLO 17.4 Write an assembly language program using procedure to solve equation such as Z =(A+B) *(C+D).

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices

VI Relevant Theoretical Background

A procedure is a set of the program statements that can be processed independently and reuse again and again. Here are the four steps that need to be accomplished in order to call and return from a procedure.

1. Save return address
2. Call the procedure
3. Execute procedure
4. Return to calling program

The assembler directives PROC and ENDP are required to define a procedure. The directive PROC specifies the beginning of the procedure and the directive ENDP specifies the end of the procedure to the assembler. The directive PROC and ENDP must enclose the procedure code which defines the subroutine. The procedures must be defined within the code segment only.

Syntax:

procedure_ name PROC [NEAR/FAR]

,:

RET
ENDP

The CALL instruction is used to transfer program control to a subprogram or a procedure by storing the return address on the stack. The call can be of two types

1. Inter-Segment or near call
2. Intra-Segment or far call

A near call refers to a procedure call which is in the same code segment as the CALL instruction and a far call refers to a procedure call which is in the different code segment from that of the CALL instruction.

Example:

CALL fact

The instruction RET is used to transfer program control from the procedure back to the calling program i.e. main program or procedure following the CALL. The RET instruction are of two types:

1. Near RET or inter segment return.
2. Far RET or intra segment return.

If a procedure is declared as near, the execution of the RET replaces the IP with a word from the top of the stack which contains the offset address of the instruction following the CALL instruction. Hence such return is called as near return because transfer of the control is within the segment. If a procedure is defined as far, the execution of RET instruction pops two words from the stack and places them into the registers IP and CS to transfer control to the calling program.

VII Required Resources:

S. No.	Instrument /Object	Specification	Quantity	Remarks
1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No./Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No./Group	Whichever is available
3.	Assembler	MASM or TASM	1 No./Group	Whichever is available
4.	Linker	LINK or TLINK	1 No./Group	Whichever is available
5.	Debugger	Debug or TD	1 No./Group	Whichever is available

VIII Precautions to be followed

1. Follow safety and operational guidelines while using Laboratory.
2. Handle computer system and peripherals with care.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed)
2. Double clik on DOSBOX TASM 1.4 icon.
3. Type *editfilename.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 *tasmfilename.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 *debugfilename.exe* or *tdfilename.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.

X Observations:

1. Observe and write the contents of Register using debugger TD or Debug after the execution of program.

Table 17.1: Contents of Registers

Registers			Flag Register		
	Before	After			
AX			Carry Flag	CF	
BX			Zero Flag	ZF	
ex			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					

2. Observe and write the contents of memory location in Data Segment after the execution of program.

Table 17.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

XII Conclusion:

……
……
……
……

XIII Practical related questions

(Use blank space provided for answers or attach more pages if needed)

1. Which procedure have been used in your program (Near or Far)?

……
……
……

2. Write the content of Instruction pointer IP before and after the execution of CALL instruction.……
……
……
……

3. What are the advantages of using procedure?

……
……
……

4. Write an ALP to find smallest number from the array of 10 numbers using procedure.

……
……
……
……

5. Differentiate between near and far proceudre.

……
……
……

……
……
……
……
……
……
……
……
……
……
……

XIV References/Suggestions for further reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/
2. http://mysc.altervista.org/beginners-guide-8086/
3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

Practical No. 18: ALP to perform arithmetic operations on given numbers using macro

I Practical Significance

In assembly language programs, small program codes of the same pattern are frequently occurring at different places of the program which perform the same operation on the different data of the same data type. Such repeated code can be written separately as a macro. The process of defining macros and using them to simplify the programming process is known as macros programming. Hence, students will be able to use macro in assembly language program.

II Industry/Employer Expected outcome(s)

Develop assembly language programs using 8086

III Course Level Learning outcome(s)

CO 5- Use procedures and micros to develop an assembly language program for a given problem.

IV Laboratory Learning outcome(s)

LLO.18.1. Use assembler directives MACRO and ENDM to write the macros using parameters.
LLO.18.2. Write an assembly language program using macro to perform for addition, subtraction, multiplication and division.

LLO.18.3. Write an assembly language program using macro to solve equation such as Z = (A+B)* (C+D).

V Relevant Affective Domain Related Outcomes

1. Follow precautionary measures.
2. Demonstrate working as a leader/ a team member.
3. Follow ethical practices.

VI Relevant Theoretical Background

When assembler encounters a macro name later in the source code, the block of code associated with the macro name is substituted or expanded at the point of call, known as macro expansion. Hence macro is called as open subroutine. Macros should be used when its body has a few program statements; otherwise, the machine code of the program will be large on account of the same code being repeated in the position where macros are used. The directive MACRO and ENDM must enclose the definition, declarations, or a small part of the code which have to be substituted at the invocation of the macro. The macro should be start with directive MACRO and end with ENDM directive.

Syntax:

macro_name MACRO *[macro variables separated by colon*]
:
:
:
ENDM

VII Required resources

S. No.	Instrument /Object	Specification	Quantity	Remarks

1.	Desktop PC	Pentium IV or above with Keyboard, Mouse, Monitor	1 No. /Group	Whichever is available
2.	Editor	MS-DOS EDIT or Notepad	1 No. /Group	Whichever is available
3.	Assembler	MASM or TASM	1 No. /Group	Whichever is available
4.	Linker	LINK or TLINK	1 No. /Group	Whichever is available
5.	Debugger	Debug or TD	1 No. /Group	Whichever is available

VIII Precautions to be followed

1. Handle computer system and peripherals with care.
2. Follow safety practices.

IX Procedure

1. Write algorithm and draw flow-chart for given program (Use blank space provided or attach more pages if needed).
2. Double clip on 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.

X Observations

1. Observe and write the contents of Register using debugger TD or Debug

Table 18.1: Contents of Registers

Registers			Flag Register		
	Before	After			
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					

2. Observe and write the contents of memory location in Data Segment using debugger TD or Debug

Table 18.2: Contents of memory location in Data Segment

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	

XI Results (Program code with output)

(Note: Write a program and output assigned by teacher)

……

……

……

……

……

……

……

……

……

……

……

……

XII Conclusion:

……

……

XIII Practical related Questions

(Use blank space provide for answers or attached more pages if needed)

1. State the advantages and disadvantages using macro.

……

……

2. State the function of directive MACRO and ENDM.

……

……

3. Write an ALP to perform $\mathbf{y=a^2+b^2+c^2}$ using macro to compute square**.**

……

……

……

……

……

……

……

……

……

……

4. Write ALP using macro to perform multiplication of two 8 bit unsigned numbers.

……

……

……

……

……

……

5. Compare Procedure and Macro.

……

……

……

……

……

……

XIV References / Suggestions for further Reading

1. https://www.elprocus.com/8086-assembly-language-programs-explanation/

2. http://mysc.altervista.org/beginners-guide-8086/

3. https://www.tutorialspoint.com/assembly_programming/

XV Assessment Scheme (25 Marks)

Performance Indicators	Weightage
Process related (15 Marks)	**60%**
1. Use editor to create assembly language program file.	20%
2. Use assembler and linker to create .exe file	20%
3. Use debugger in single step mode to locate/trace the errors and correcting the errors	20%
Product related (10 Marks)	**40%**
4. Practical related questions	15%
5. Expected Output/Observation	15%
6. Completion and submission of practical in time	10%
Total (25 Marks)	**100%**

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

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI