LIVE TO LEARN & LEARN TO SHARE

MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

DEPARTMENT OF EMERGING TECHNOLOGIES

M A N U A L

Name: .....

Roll No: .....**Branch:** .....

Year: .....**Sem:** ..... **Academic Year:**.....

Subject: .....

CSE EMERGING TECHNOLOGIES

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

DEPARTMENT OF EMERGING TECHNOLOGIES

CERTIFICATE

Certified that this is the Bonafide Record of the Work Done by Mr./Ms ..... Roll.No..... …of B.Tech …...….. Year ..... Semester, Specialization …….for Academic year 20..… - 20…... in ……Laboratory.

Date: Faculty Incharge HOD

Internal Examiner External Examiner

INDEX

S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

MRCET CAMPUS

(AutonomousInstitution–UGC,Govt. of India)

(Approved by AICTE-Accredited by NBA (Tier-1) &NAAC– 'A' Grade ISO 9001:2015Certified)

Maisammaguda, Dhulapally, Hyderabad–500 100,Telangana State,India

DEPARTMENT OF EMERGING TECHNOLOGIES

VISION

- "To be at the forefront of Emerging Technologies and to evolve as a Centre of Excellence in Research, Learning and Consultancy to foster the students into globally competent professionals useful to the Society."

MISSION

The department of CSE (Emerging Technologies) is committed to:

- To offer highest Professional and Academic Standards in terms of Personal growth and satisfaction.
- Make the society as the hub of emerging technologies and thereby capture opportunities in new age technologies.
- To create a benchmark in the areas of Research, Education and Public Outreach.
- To provide students a platform where independent learning and scientific study are encouraged with emphasis on latest engineering techniques.

Quality Policy

- To pursue continual improvement of teaching learning process of Undergraduate and Post Graduate programs in Engineering & Management vigorously.
- To provide state of art infrastructure and expertise to impart the quality education and research environment to students for a complete learning experiences.
- Developing students with a disciplined and integrated personality.
- To offer quality relevant and cost effective programmes to produce engineers as per requirements of the industry need.

For more information: www.mrcet.ac.in

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

- To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1. Fundamentals and critical knowledge of the Computer System:- Able to understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

EMBEDDED SYSTEMS DESIGN

MANUAL

B.TECH (R-20 Regulation)

(III YEAR–I SEM)

(2022-23)

DEPARTMENT OF CSE

(EMERGING TECHNOLOGIES)

(INTERNET OF THINGS)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

INDEX

S.NO	List of experiments	Page No
	Cycle 1: Using 8086 Processor Kits and/or Assembler	
1	Programs for 16 bit arithmetic operations for 8086 (using Various addressing modes)	2
2	Program for sorting an array for 8086	6
3	Program for String manipulation	10
4	Program to perform logical operations	13
	Cycle 2: Using 8051 Microcontroller Kit	
1	Program to perform arithmetic & logical operations	19
2	Serial Communication from / to 8051 to / from I/O devices	23
	Cycle 3: Interfacing I/O Devices to Arduino	
1	Program for interface LED	26
2	Program for interface LCD to Arduino	27
3	Program to interface soil Moisture sensor to Arduino	28
4	Serial communication programming	29

Cycle 1: Using 8086 Processor Kits and/or Assembler

PART-A
INTRODUCTION TO MASM

EDITOR

An editor is a program, which allows you to create a file containing the assembly language statements for your program. As you type in your program, the editor stores the ASCII codes for the letters and numbers in successive RAM locations. When you have typed in all of your programs, you then save the file on a floppy of hard disk. This file is called source file. The next step is to process the source file with an assembler. In the MASM /TASM assembler, you should give your source file name the extension, .ASM

ASSEMBLER

An assembler program is used to translate the assembly language mnemonics for instructions to the corresponding binary codes. When you run the assembler, it reads the source file of your program from the disk, where you saved it after editing on the first pass through the source program the assembler determines the displacement of named data items, the offset of labels and pails this information in a symbol table. On the second pass through the source program, the assembler produces the binary code for each instruction and inserts the offset etc that is calculated during the first pass. The assembler generates two files on floppy or hard disk. The first file called the object file is given the extension. OBJ. The object file contains the binary codes for the instructions and information about the addresses of the instructions. The second file generated by the assembler is called assembler list file. The list file contains your assembly language statements, the binary codes for each instructions and the offset for each instruction. In MASM/TASM assembler, MASM/TASM source file name ASM is used to assemble the file. Edit source file name LST is used to view the list file, which is generated, when you assemble the file.

LINKER

A linker is a program used to join several object files into one large object file and convert to an exe file. The linker produces a link file, which contains the binary codes for all the combined

modules. The linker however doesn't assign absolute addresses to the program, it assigns is said to be reloadable because it can be put anywhere in memory to be run. In MASM/TASM, LINK/TLIN9K source filename is used to link the file.

DEBUGGER

A debugger is a program which allows you to load your object code program into system memory, execute the program and troubleshoot are debug it the debugger allows you to look at the contents of registers and memory locations after your program runs. It allows you to change the contents of register and memory locations after your program runs. It allows you to change the contents of register and memory locations and return the program. A debugger also allows you to set a break point at any point in the program. If you inset a breakpoint the debugger will run the program up to the instruction where the breakpoint is set and stop execution. You can then examine register and memory contents to see whether the results are correct at that point. In MASM/TASM, td filename is issued to debug the file.

DEBUGGER FUNCTIONS:

1. Debugger allows looking at the contents of registers and memory locations.
2. We can extend 8-bit register to 16-bit register which the help of extended register option.
3. Debugger allows setting breakpoints at any point with the program.
4. The debugger will run the program up to the instruction where the breakpoint is set and then stop execution of program. At this point, we can examine registry and memory contents at that point.
5. With the help of dump we can view register contents.
6. We can trace the program step by step with the help of F7.
7. We can execute the program completely at a time using F8

The DOS -Debugger:

The DOS "Debug" program is an example of simple debugger that comes with MS-DOS. Hence it is available on any PC .it was initially designed to give the user the capability to trace logical errors in executable file.

Below, are summarized the basic DOS - Debugger commands

COMMAND	SYNTAX
Assemble	A [address]
Compare	C range address
Dump	D [range]
Enter	E address [list]
Fill	F range list
Go	G [=address] [addresses]
Hex	H value1 value2
Input	I port
Load	L[address] [drive][first sector][number]
Move	M range address
Name	N[pathname][argument list]
Output	O port byte
Proceed	P [=address][number]
Quit	Q
Register	R[register]
Search	S range list
Trace	T [=address][value]
Unassembled	u [range]
Write	W[address][drive][first sector][number]

MS-MASM:

Microsoft's Macro Assembler (MASM) is an integrated software package Written by Microsoft Corporation for professional software developers. It consists of an editor, an assembler, a linker and a debugger (Code View). The programmer's workbench combines these four parts into a user-friendly programming environment with built in on line help. The following are the steps used if you are to run MASM from DOS

MICROPROCESSOR LAB EXECUTION PROCEDURE

STEP1: Opening the DOS prompt

Click start menu button and click on Run and then type cmd at command prompt immediately DOS window will be appeared

STEP2: Checking the masm installation

To know MASAM is installed or not simply type masm at the command prompt upon that it replies masm version vendor (Microsoft), etc... If you get any error there is no masm in that PC

STEP3: Directory changing (create a folder with your branch and no in D drive)

Change the current directory to your won directory suppose your folder in D drive type the following commands to change the directory at command prompt type D: hit enter now you are in D drive type cd folder name hit the enter

Example: D cd ece10

Now we are in folder cse10

STEP4: writing the program

At the command prompt type the edit programname.asm

Example. Edit add.asm

Immediately editor window will open and there you have to write the program. Type the program in that window after completion save the Program, to save the program Go to file opt in the menu bar and select save opt now your code is ready to Assemble.

STEP5: Assembling, Linking and executing the program

Go to file opt click exit opt now DOS prompt will be displayed to assemble the program type the following commands at the DOS prompt

Masm Program Name, Program Name, Program Name, Program Name hit the enter

Example: Masm add, add, add, add enter

OR

Example: Masm add.asm

If there are any errors in the program assembler reports all of them at the command promptwith line no's, if there are now bugs your ready to link the program. To link the program type the following line at command prompt Link program name**,,,,,** (5 commas)

Example: Link add,,,,,OR

Example: link add.obj

After linking you are ready to execute the program. To execute the program type the followingcommand

Debug program name.exe hit the enter

Example: Debug add.exe

Now you entered into the execution part of the program here you have to execute the program instruction by instruction (debugging) first of all press the ***r*** key(register) hit the enter key it'll displays all the registers and their initial values in HEXDECIMAL note down the values of all the register which are used in the program. To execute the next instruction press **t** key (TRACE) hit the enter it'll execute that instruction and displays the contents of all the register. You have to do this until you reach the last instruction of the program. After execution you have to observe the results(in memory or registers based on what you have written in the program).

STEP6: Copying list file (common for all programs):

A list file contains your code starting address and end address along with your program .For every program assembler generates a list file at your folder, programname.lst (ex. Add.lst) you should copy this to your lab observation Opening a list file

Edit program. 1 st Example. Edit add l st

1a. Program to add two 16 bit numbers

```
                                Assume cs:code,ds:data
0000                            data segment                //data segment starts
0000 1204                               n1 dw 1204h         // declaring inputs n1,n2
0002 5698                               n2 dw 5698h
0004 0000                               sum dw ?            //declaring output variable
0006 0001                               carry db 1 dup (0)  //declaring carry
        00
0007                            data ends                   //data segment ends


0000                            code segment                //code segment starts
0000 B8-----R                           mov ax,data         //moving starting address of
                                                            //data segment into ax and then
0003 8E D8                              mov ds,ax           //into data segment register.
0005 A1 0000 R                          mov ax, n1          //loading n1 into accumulator
0008 03 06 0002 R                       add ax, n2          //add n2 to ax result in ax
000C 73 04                              jnc l1              //if no carry jump to label l1
000E FE 06 0006 R                       inc carry           //if carry =1increment carry
0012 A3 0004 R                  l1:     mov sum, ax         //store result in location sum
0015 CC                                 int 3h              //break point interrupt
0016                                    code ends           //code segment ends
                                        End                 //program ends
```

Result: d ds:0000 0006

Input:

Address	value
0B61:0000	04
0B61:0001	12
0B61:0002	98
0B61:0003	56

Output:

Address	value

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. What are the various registers present in 8086 architecture?
3. Explain the addition instruction?
4. What is the difference between carry and Auxiliary carry?
5. What is the difference between ADD and ADC instructions?

1b. Program to subtract two 16 bit numbers

```
                          Assume cs:code,ds:data
0000                      data segment                  //data segment starts
0000  9734                      n1 dw 9734h             // declaring inputs n1,n2
0002  1423                      n2 dw 1423h
0004  0000                      diff dw ?               //declaring output variable
0006  0001                      borrow db 1 dup(0)      //declaring borrow
0007                      data ends                     //data segment ends


0000                      code segment                  //code segment starts
0000  B8 -----R                 mov ax,data             //moving starting address of
0003  8E D8                     mov ds,ax               //data segment into ax and then
                                                        //into data segment register.
0005  A1 0000 R                 mov ax,n1               //loading n1 into accumulator

0008  2B 06 0002 R              sub ax,n2               //subtract n2 from Ax, result in
                                                        Ax
000C  73 04                     jnb l1                  //if no borrow jump to label l1
000E FE 06 0006 R               inc borrow              //if borrow =1increment
                                                              borrow
0012  A3 0004 R           l1:   mov diff,ax             //store result in location diff
0015  CC                        int 3h                  //break point interrupt
0016                      code ends                     //code segment ends
                          End                           //program ends
```

Result: d ds:0000 0006

Input:

Address	value
0B61:0000	34
0B61:0001	97
0B61:0002	23
0B61:0003	14

Output:

Address	value

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. What are the various registers present in 8086 architecture?
3. Explain the Subtraction instruction?
4. What is the difference between carry and Auxiliary carry?
5. What is the difference between SUB and SBB instructions

1c. Program to multiply two 16 bit numbers

```
                                Assume cs:code,ds:data
0000                            data segment
0000  1214                              n1 dw 1214H          //declaring variables n1
                                                                     and n2
0002 3162                               n2 dw 3162H
0004  0000                              lproduct dw ?        // declaring two locations to
                                                             store lower and higher order
                                                                     words of the result
0006 0000                               hproduct dw ?
0008                            data ends
0000                            code segment
0000  B8 -----R                         mov ax,data
0003  8E D8                             mov ds,ax
0005  A1 0000 R                         mov ax,n1            // moving n1 into ax
0008  8B 1E 0002 R                      mov bx,n2            //moving n2 into bx
000C  F7 E3                             mul bx               //multiplying bx with ax
000E  A3 0004 R                         mov lproduct,ax      //store lower word of result
                                                             from ax into lproduct location
0011  89 16 0006 R                      mov hproduct,dx      //store higher word of result
                                                             from dx into hproduct location
0015  CC                                int 3H               //break point interrupt

0016                            code ends
                                End
```

Result: d ds:0000 0007

Input:

Address	value
0B61:0000	14
0B61:0001	12
0B61:0002	62
0B61:0003	31

Output:

Address	value

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. What are the various registers present in 8086 architecture?
3. Explain hexadecimal multiplication by taking an example.
4. Where is the result stored after multiplication?
5. What is the difference between MUL and IMUL instructions?

1d. Program to divide 16 bit number/8 bit number

```
                        Assume cs:code,ds:data
0000                    data segment
0000  89EA                    N1 dw 89EAH        // declaring N1 as 16bit num
0002  98                      N2 db 98H          //declaring N2 as 8 bit
0003  00                      quo db ?           // declaring variables to store
                                                    quotient and remainder
0004  00                      rem db ?
0005                    data ends
0000                    code segment
0000  B8 -----R               mov ax,data
0003  8E D8                   mov ds,ax
0005  A1 0000 R               mov ax,N1          // moving N1 into ax
0008  8A 1E 0002 R            mov bl,N2          //moving N2 into bl register
000C  F6 F3                   div bl             //divide Ax by bl
000E  A2 0003 R               mov quo,al         //storing quotient in al into
                                                 location quo
0011  88 26 0004 R            mov rem,ah         //storing remainder in al into
                                                 location rem
0015  CC                      int 3h
0016                    code ends
                        end
```

Result: d ds:0000 0004

Input:

Address	value
0B61:0000	EA
0B61:0001	89
0B61:0002	98

Output:

Address	value

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. What are the various registers present in 8086 architecture?
3. Explain hexadecimal division by taking an example.
4. Where is the result stored after division?
5. What is the difference between DIV and IDIV instructions?

2a. Program to sort an array in ascending order

```
                                Assume cs:code,ds:data
0000                            data segment
0000  24 32 14 82 16                arr db 24H,32H,14H,82H,16H      //declaring an 8 bit
                                                                    array with 5 elements
= 0004                              count equ 0004H                 //declaring count =4
0005                            data ends
0000                            code segment
0000  B8 -----R                      mov ax,data
0003  8E D8                          mov ds,ax
0005  BB 0004                        mov bx,count                   Initialize bx with count
0008  8B CB                     l1:  mov cx, bx
000A 8D 36 0000 R                    lea si,arr
000E  8A 04                     l2:  mov al,[si]
0010  3A 44 01                       cmp al,[si+1]                  Compare number in al
0013  72 05                          jb l3                          with memory location
0015  86 44 01                       xchg al,[si+1]                 number
0018  88 04                          mov [si],al
001A  46                        l3:  inc si
001B  E2 F1                         loop l2
001D 4B                             dec bx
001E 75 E8                          jnz l1
0020  CC                            int 3h
0021                            code ends
                                End
```

Result: d ds:0000 0004

Input: 24 32 14 82 16
Output:

Theory:

Bubble sort logic: Sorting technique used in the program is the bubble sort. The first number of the series is compared with the second number. If the first number is greater thanthe second number, exchange their positions in the series otherwise leave the positions unchanged. Then compare the second number in the recent series with third and repeat the exchange part that was carried out for the first and the second number and for all the remaining numbers of the series. Repeat this procedure for the complete series (n-1) times. After (n-1) iterations , the largest number moves to the end of the series. Again start from the first address of the series. Repeat the same procedure now for (n-2) iterations so that second highest number is in the last but one place in the series. Continue this until the complete series is arranged in the ascending order.

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. Explain JB, JNZ and LEA instructions?
3. Explain how sorting takes place by taking an example array.
4. Explain CMP and LOOP instructions?
5. What are the differences between conditional and unconditional jumps?

2b. Program to sort an array in descending order

```
                                Assume cs:code,ds:data
000                             data segment
0000  24 32 14 86                     arr db 24h,32h,14h,86h,16h     //declaring an 8 bit
= 0004                                 count equ 04h
0004                            data ends                              //declaring count =4
0000                            code segment
0000  B8 -----R                       mov ax,data
0003  8E D8                           mov ds,ax
0005  33 C3                           xor ax,bx
0007  BA 0004                         mov dx,count                 Initialize dx with count
000A                            l2:
000A 8B CA                            mov cx,dx
000C  8D 36 0000 R                    lea si,arr
0010                            l1:
0010  8A 04                           mov al, [si]
0012  3A 44 01                        cmp al, [si+1]               Compare number in al with memory
0015  77 05                           ja l3                        location number
0017  86 44 01                        xchg [si+1],al
001A  86 04                           xchg [si],al
001C                            l3:
001C  46                              inc si
001D E2 F1                            loop l1
001F  4A                              dec dx
0020  75 E8                           jnz l2
0022  CC                              int 3h
0023                            code ends
                                End
```

Result: d ds:0000 0004

Input: 24 32 14 86 16
Output:

Theory:

Bubble sort logic: Sorting technique used in the program is the bubble sort. The first number of the series is compared with the second number. If the first number is smaller than the second number, exchange their positions in the series otherwise leave the positions unchanged. Then compare the second number in the recent series with third and repeat the exchange part that was carried out for the first and the second number and for all the remaining numbers of the series. Repeat this procedure for the complete series (n-1) times. After (n-1) iterations, the smallest number moves to the end of the series. Again start from the first address of the series. Repeat the same procedure now for (n-2) iterations so that

second highest number is in the last but one place in the series. Continue this until the complete series is arranged in the ascending order.

Sample Viva questions:

1. Explain the assembler directives used in the program?
2. Explain JB, JNZ and LEA instructions?
3. Explain how sorting takes place by taking an example array
4. Explain CMP and LOOP instructions
5. What are the differences between conditional and unconditional jumps?

3. Program to search a character in a given string

```
                                Assume cs:code,ds:data
0000                          data segment
0000   6C 61 62 20 24              str1 db 'lab $'          Declaring input string
=                                  count equ ($-str1)       declaring count
0005
0005   66 6F 75 6E 64 20           str2 db 'found $'        declaring string as
24                                                          "found"

       6E 6F 74 20 66 6F          str3 db 'not found $'     declaring string
000C                                                        as"not found"
          75 6E 64 20 24
0017                               data ends
0000                               code segment
0000   B8-----R                    mov ax,data              moving starting
                                                            address of data
                                                            segment into ax
0002    0003 8E D8                 mov ds,ax                and then into data
                                                            segment register.
0005   8E C0                       mov es,ax                Initializing Extra
                                                            segment register with
                                                            starting address of
                                                            data segment itself.
0007   B9 0004                     mov cx,count-1
000A     8D 3E 0000 R              lea di,str1              moving starting
                                                            address of input
                                                            string into DI register
000E  B0 61                        mov al,'a'               Move character to be
                                                            searched into al
                                                            register
0010   F2/ AE                      repne scasb               scan the string for the
                                                                  character
0012   74 0B                       jz l1                    If the character found
                                                                  jump to l1
0014   B4 09                       mov ah,09H               load the function call
                                                            value 09 into ah
0016   8D 16 000C R                lea dx,str3              load the starting
                                                            address of string
                                                            "not found" into DX
001A   CD 21                       int 21H                  display the string
001C   EB 09 90                    jmp l2
001F   B4 09                l1:    mov ah,09H               load the function call
                                                            value 09 into ah
0021   8D 16 0005 R                lea dx,str2              //load the starting
                                                            address of string
                                                            "found" into dx
0025   CD 21                       int 21H                  display the string
0027   CC                   L2:    int 3H                   Break point interrupt
0028                               code ends
                                   End
```

Result:

Input: character ‘a’ in string “lab”

Output:

4a. Program to reverse the given string

```
                                Assume cs:code,ds:data,es:data
0000                            data segment
0000   6D 70 6D 63 6C 61            str1 db 'mpmclab$'          //Declaring the input
                                                                        string
    62 24
= 0008                              count equ($-str1)
0008    0007                            str2 db 8 dup(0)                //Declaring a string to
                                                                         store the output string
000F                            data ends
0000                            code segment
0000   B8 ---- R                    mov ax,data
0003   8E D8                        mov ds,ax
0005   8E C0                        mov es,ax
0007   B9 0007                      mov cx,count-1
000A   8D 36 0000 R                 lea si,str1                     //load the starting
                                                                    address of string1 intoSI
                                                                    register.
000E   8D 3E 0008 R                 lea di,str2                     //load the starting
                                                                    address of string2 intoDI
                                                                    register.
0012   83 C6 06                     add si,count-2                  // add the count-2 to SI

                                                                    value to go to the endof
                                                                     the string
0015   FD                    l1:     std                            //set direction flag
0016   AC                           lodsb                           //load the contents of
                                                                    string1 into al
0017   FC                           cld                             //clear direction flag
0018   AA                           stosb                           //store the content of al
                                                                    into string2
0019   E2 FA                        loop l1                         //repeat the above loop
                                                                    for count number of
                                                                    times.
001B   B0 24                        mov al,24h                      //move the ascii value
                                                                    of '$' into al register.
001D   AA                           stosb                           //store the value in al
                                                                    into the output string
001E   B4 09                        mov ah,09h                      // use INT 21h
                                                                    functionwith a function
                                                                    callvalue of 09h in ah to
                                                                    display the output string.

0020    8D 16 008R                  lea dx str2
0024    CD 21                       int 21h
0026    CC                          int 3h
0027                                 Code ends
                                    end
```

Result:

Input: mpmclab

Output:

4b. Program to compare the two given strings

```
                                Assume cs:code,ds:data,es:data
0000                            data segment
0000  4D 50 4D 43 6C 61                 str1 db 'MPMClab$'                  //declaring input
                                                                                    string 1
          62 24

= 0008                                  count equ ($-str1)
0008  69 6D 70 6C 61 62                 str2 db 'implab$'                   //declaring input
      24                                                                            string 2

000F  65 71 75 61 6C 24                 str3 db 'equal$'
0015  6E 6F 74 20 65 71                 str4 db 'not equal$'
    75 61 6C 24
001F                            data ends
0000                            code segment
0000  B8 ---- R                         mov ax,data
0003  8E D8                             mov ds,ax
0005  8E C0                             mov es,ax
0007  B9 0007                           mov cx,count-1
000A  8D 36 0000 R                      lea si,str1                 //load the
                                                                    starting address
                                                                    of string1 in SI
000E  8D 3E 0008 R                      lea di,str2                 //load the
                                                                    starting address
                                                                    of string2 in DI
0012  F3/ A6                            repe cmpsb                  //compare the
                                                                    two strings
0014  75 0B                             jne l1                      //if the strings
                                                                    are not equal
                                                                    jump to label l1
0016  B4 09                             mov ah,09h
0018  8D 16 000F R                      lea dx,str3                 //display
                                                                    function
001C  CD 21                             int 21h
001E EB 09 90                           jmp l2
0021  B4 09                     l1:     mov ah,09h
0023  8D 16 0015 R                      lea dx,str4                 //display
                                                                    function
0027  CD 21                             int 21h
0029  CC                        l2:     int 3h                      //breakpoint interrupt
002A                            code ends
                                End
```

Result:
Input: MPMClab, implab

Output:

4c. Program to check if a given string is palindrome or not

```
                                Assume cs:code,ds:data,es:data
0000                            data segment
0000  6C 69 72 69 6C 24                str1 db "liril$"                //declaring input string
= 0006                                  count equ ($-str1)
0006  0006[                             str2 db 6 dup(0)                //declaring string to
                                                                        store the result
          00
                              ]
000C  70 61 6C 69 6E 64                 str3 db "palindrome$"

     72 6F 6D 65 24
0017  6E 6F 74 70 61 6C            str4 db "notpalindrome$"
     69 6E 64 72 6F 6D
     65 24
0025                            data ends
0000                            code segment
0000  B8----- R                         mov ax,data
0003  8E D8                             mov ds,ax
0005  8E C0                             mov es,ax
0007  B9 0005                           mov cx,count-1
000A 8D 36 0000 R                       lea si,str1
000E 8D 3E 0006 R                       lea di,str2
0012    83 C6 04                        add si,count-2
0015                            l1:
0015    FD                              std                     //set the direction flag
0016    AC                              lodsb                   //load the contents of
                                                                the input string pointed
                                                                by SI into al
0017    FC                              cld                     //clear direction flag
0018    AA                              stosb                   //store the content of al
                                                                into output string
                                                                pointed by DI
0019    E2 FA                           loop l1                 //repeat the steps for
                                                                count number of times
001B    8D 36 0000 R                    lea si,str1             //again initiaze SI and
                                                                DI with starting
                                                                addresses of string1 and
                                                                        string2
001F  8D 3E 0006 R                      lea di,str2
0023  B9 0005                           mov cx,count-1
0026  F3/ A6                            repe cmpsb                      //compare the two
                                                                                strings
0028  75 0B                             jne l2                  //if the strings are not
                                                                equal jump to label l2
002A  B4 09                             mov ah,09h
```

```
002C  8D 16 000C R        lea dx,str3        //display "palindrome"
0030  CD 21               int 21h
0032  EB 09 90            jmp l3
0035                l2:
0035  B4 09               mov ah,09h
0037  8D 16 0017 R        lea dx,str4        //display "not palindrome"
003B  CD 21               int 21h
003D                l3:
003D CC                   int 3h
003E                code ends
                    End
```

Result:

Input: liril

Output:

Sample Viva questions:

1. What are DOS function calls?
2. Explain the various string instructions?
3. What is the need for STD and CLD instructions?
4. Explain the Duplicate and EQU assembler directives.
5. Explain the LODSB and STOSB instructions instructions

Cycle 2: Using 8051 microcontroller Kits and/or Assembler

1.Program to perform arithmetic & logical operations

Program to add two 8 bit numbers

Program:

ORG 0000H MOV R0, #52H MOV A, #29H ADD A, R0 END

Source code:

Address	Opcodes	Mnemonics
0000	78 52	MOV R0, #52
0002	74 29	MOV A, #29
0004	28	ADD A, R0

Result:

- **Program to subtract two 8 bit numbers**

Program:

ORG 0000H CLR C
MOV R0, #52H MOV A, #89H SUBB A, R0 END

Source code:

Address	Opcodes	Mnemonics
0000	C3	CLR C
0001	78 52	MOV R0, #52
0003	74 89	MOV A, #89

0005 98 SUBB A, R0

Result:

- **Program to multiply two 8 bit numbers**

Program:

ORG 0000H MOV A, #14H MOV B, #04H MUL AB END

Source code:

Address	Opcodes	Mnemonics
0000	74 14	MOV A, #14
0002	75 F0 04	MOV B, #04
0005	A4	MUL AB

Result:

- **Program to divide two 8 bit numbers**

Program:

MOV A, #80H MOV B, #0EH DIV AB
END

Source code:

Address	Opcodes	Mnemonics
0000	74 80	MOV A,#80
0002	75 F0 0E	MOV B ,#0E
0005	84	DIV AB

Result:

2.Serial Communication from / to 8051 to / from I/Odevices

2a. Program to transfer a character ‘A’:

Program:

```
MOV TMOD, #X020
MOV SCON, #0X50
MOV TH1, #0xfd
SETB TR1
Mov SBUF, #’A’
Again: JNB TI, again
        CLR TR1
```

2b. Program to receive a character:

Program:

```
MOV TMOD, #X020
MOV SCON, #0X50
MOV TH1, #0xfd
SETB TR1
Again: JNB RI, again
        CLR RI
```

Cycle 3: Interfacing I/O Devices to Arduino

1. Program for interface LED to Arduino

1.Program:

```
int led=8;
void setup()
{
Pinmode(led,OUTPUT);
}
Void loop()
{
digitalWrite(led,HIGH);
delay(2000);
digitalWrite(led,LOW);
delay(2000);
}
```

Result:

2. Interfacing LCD to Arduino Uno

1) Program:

```
liquidCrystal lcd(12,11,4,3,2,1);
void setup()
{
Lcd.begin(16,2);
}
Void loop()
{
Lcd.setCursor(0,0);
Lcd.print("welcome to ESD");
}
```

Result:

3.Interfacing Soil Moisture sensor to Arduino

Program:

```
Int s=9;
Void setup()
{
pinMode(s,INPUT);
}
Void loop()
{
int p=digitalRead(s);
if(s==1)
{
Serial.println("soil is dry");
}
else
{
Serial.println("soil is wet");
}
```

Ouput:

4. Serial communication Programming

Program:

1) Transfer a string

```
Void setup()
{
Serial.begin(9600);
}
Void loop()
{
Serial.println("Welcome to IOT");
}
```

Output: