# GOVERNMENT POLYTECHNIC COLLEGE, PERUMBAVOOR

## <u>Vision</u>

Excel as a centre of skill education moulding professionals who sincerely strive for the betterment of society.

### <u>Mission</u>

1. To impart state of the art knowledge and skill to the graduate and moulding them to be competent, committed and responsible for the well being of society.
2. To apply technology in the traditional skills, thereby enhancing the living standard of the community

## DEPARTMENTOFCOMPUTER ENGINEERING

### Vision

Excel as a skill center in Computer Engineering moulding professionals who are adaptive and sincerely strive towards betterment of society.

### Mission

1. To impart state of the art, knowledge, skill and attitude to the graduates ensuring sustainable development.
2. To develop adaptiveness for being competant to acquaint with the technological changes .

## PROGRAM EDUCATIONAL OUTCOMES (PEOs)

1. To produce technically competent diploma holders in engineering with scientific, analytical, mathematical and problem solving skills.

2. To develop the habit of quality, safety, self-learning along with environmental awareness.

3. To equip diploma holders with good management practices, interpersonal skills and entrepreneurial discipline with strong adherence to ethics and values.

## PROGRAM OUTCOMES (POs)

1. An ability to apply knowledge of basic mathematics, science and engineering to solve engineering problems.

2. An ability to apply discipline-specific knowledge to solve core and/or applied engineering problems.

3. An ability to plan and perform experiments and practices and to use the results to solve engineering problems.

4. Apply appropriate technologies and tools with an understanding of the limitations.

5. Demonstrate knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to engineering practice.

6. Understand the impact of the engineering solutions in societal and environmental contexts, and demonstrate the knowledge and need for sustainable development.

7. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

8. Function effectively as an individual, and as a member or leader in diverse/ multidisciplinary teams.

9. An ability to communicate effectively.

10. Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the context of technological changes.

## PROGRAM SPECIFIC OUTCOME(PSO)

1. **Specialization knowledge**: The computer engineering diploma graduate will be able to work in information technology industry in the area of development, implementation, testing and maintenance.

2. **Professional growth:** The computer engineering diploma graduate will be fit for real time software projects.

3. **Entrepreneurship:** A successful career as an entrepreneur with a passion, social commitment and ethical responsibility for real-world applications using optimal resources.

**COURSE OUTCOMES**

| Course Outcome | | Blooms Taxonomy Level | Lab Hours |
|---|---|---|---|
| **CO1** | Outline the Programming environment of x86 | U | 4 |
| **CO2** | Experiment with the Instruction set and Programming Concepts of x86 Processor- Data transfer Instructions- Branch instructions- Arithmetic instructions | U,A | 16 |
| **CO3** | Experiment with the Instruction set and Programming Concepts of x86 Processor- Shift and Rotate Instructions- String Instructions | U,A | 16 |
| **CO4** | Experiment with the Instruction set and Programming Concepts of x86 Processor- Procedures, Macros and Number Format Conversions | U,A | 16 |

**EXP. 1**

# 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/TLIN5K 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* uponthat 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

Ex. **D cd ece10**

Now we are in folder cse10

**STEP4**: **writing the program**

At the command prompt type the **edit programname.asm**

**Ex. 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**: A**ssembling, 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

**Ex: Masm add, add, add, add enter**

**OR**

**Ex: Masm add.asm**

If there are any errors in the program assembler reports all of them at the command prompt with 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

**OR**

**Ex: link add.obj**

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

**Debug program name.exe** hit the enter

**Ex: 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 programname.lst**
**Ex. Edit add.lst**

**EXP NO. 2**

## ODD OR EVEN

### AIM

Write a program to check whether the given number is odd or even.

### OBJECTIVES

- To understand the use of rotate instruction.

### ALGORITHM

**Step I**: Initialize the data segment memory.

**Step II :** Read number to check odd or even

**Step III** : Shift the number one position right

**Step IV**: Check if there is carry

**Step V:** If carry display odd number

**Step VI:** If not carry display even number

**Step V:** Stop

### PROGRAM:

```
.MODEL SMALL
.STACK 100H
.DATA
        MSG1 DB "NUMBER IS ODD$"
        MSG2 DB "NUMBER IS EVEN$"
        READ DB "ENTER A NUMBER $"
.CODE
        MOV AX, @DATA
        MOV DS, AX
        LEA DX, READ
```

```
            MOV AH, 09H
            INT 21H
            MOV AH, 01H
            INT 21H
            SUB AL,30
            SHR AL,01
            JC LABEL1
            LEA DX, MSG2
            MOV AH,09H
            INT 21H
            JMP LAST
LABEL1:     LEA DX, MSG1
            MOV AH,09H
            INT 21H
LAST:       MOV AH,4CH
            INT 21H
```

**OBSERVATIONS**

Enter a number 5

The given number is odd

**RESULT**

Verified the given number is odd or even.

**EXP NO. 3**

## LARGEST OF 10 NUMBERS

**AIM**

Write a program to find largest from the given numbers.

**OBJECTIVES**

- To understand the use of branch instruction.

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Set input numbers in the memory

**Step III** : Set count in a register

**Step IV**: Compare data between memory and one largest

**Step V:** If swapping necessary do it

**Step VI:** Increment memory for next data

**Step V:** Decrement count and repeat step IV if needed

**Step VI:** Store/display largest number

**PROGRAM:**

```
.MODEL SMALL
.STACK 100H
.DATA
        NUM DB 06H,08H,09H,10H,13H,12H,35H,21H,22H,30H
        LARGE DB 0H
.CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV CX, 10
        MOV AL,0H
```

```
            LEA SI,NUM
CHECK:  MOV BL,[SI]
            CMP AL,BL
            INC SI
            INC NOCARRY
            MOV AL,BL
NOCARRY     DEC CX
            JNZ CHECK
            MOV LARGE, AL
            MOV AH,4CH
            INT 21H
            END
```

## OBSERVATIONS

1) Num 06h,08,09,10h,13h,12h,35h,21h,22h,30h
2) large 0

Largest = 35h

## RESULT

Program executed and obtained the result

**EXP NO. 4**

## FACTORIAL

**AIM**

Write a program to find factorial of given number

**OBJECTIVES**

- To understand the use of branch instruction.

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Read data from keyboard

**Step III** : Convert it to decimal

**Step IV: Continues** multiplication from 1 to that number

**Step VI:** Store/display factorial of the given number

**PROGRAM:**

```
.MODEL SMALL
.STACK 100H
.DATA
    MSG DB "ENTER THE NUMBER $"
    FACT DB 0H
.CODE
    MOV AX, @DATA
    MOV DS, AX
    MOV AX,0H
    LEA DX, MSG
    MOV AH,09H
```

```
        INT 21H
        MOV AH, 01H
        INT 21H
        SUB AL, 30H
        MOV AH,0H
        JZ FINAL
        MOV CX, AX
        DEC CX
FACTO:  MUL CX
        DEC CX
        JNZ  FACTO
        MOV AH,4CH
        INT 21H
        END
```

**OBSERVATIONS**

Enter a number : 5

Factorial of given number is  : 120

**RESULT**

Program executed and obtained the result

**EXP NO. 5**

# HEXADECIMAL TO DECIMAL CONVERSION

## AIM

Write a program to convert 16 bit hexadecimal number to decimal number

## OBJECTIVES

- To understand the use of branch , stack related instructions

## ALGORITHM

**Step I**: Initialize the data segment memory.

**Step II :** Move the number into ax register

**Step III** : Set CX as 10

**Step IV**: Divide CX with AX

**Step V:** Push DX value into stack

**Step VI:** Update DX value to 0

**Step V:** Increment count

**Step VI:** Compare AX as 0

**Step VII:** Repeat step 4 – step 8 till CX is zero

**Step VIII:** Pop DX from stack

**Step IX:** Add 30 to DL register

**Step X:** If ZF = 0 repeat step 10- step 11

**Step XI: Stop**

## PROGRAM:

```
.MODEL SMALL
.STACK 100H
.DATA
    NUM DB  OFH
    COUNT DB 0H
.CODE
```

```
        MOV AX, @DATA
        MOV DS, AX
        MOV AX , NUM
        MOV DX ,0H
        MOV CX, 10
REV:    DIV CX
        PUSH DX
        MOV DX,0
        INC COUNT
        CMP AX,0
        JNE REV
DISP:   POP DX
        ADD AL, 30H
        MOV AH,02H
        INT 21H
        DEC COUNT
        JNZ DISP
        MOV AH,4CH
        INT 21H
        END
```

**OBSERVATIONS**

1) Hexa decimal Number = 0f
2) Decimal = 15

**RESULT**

Program executed and obtained the result

**EXP NO. 6**

## COMPARE TWO STRING

**AIM**

Write a program to compare between two string

**OBJECTIVES**

- To understand the use of string instructions

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Initialize first string

**Step III** : Initialize second string

**Step IV**: Initialize message for 'Same' or 'Not same'

**Step V:** String Offset values are stored in the Index register

**Step VI:** Compare string using string instructions

**Step VII:** Repeat step 6 until last charector

**Step VIII:** Check flags

**Step IX:** Display based on flag

**Step X: Stop**

**PROGRAM:**

```
.MODEL SMALL
.STACK 100H
.DATA
    DATA1 DB "WELCOMM"
    DATA2 DB "WELCOME"
    GUD DB "STRING IS SAME $"
    BAD DB "STRINGS ARE NOT SAME $"
```

```
.CODE
            MOV AX , @DATA
            MOV DS, AX
            MOV ES ,AX
            LEA SI, DATA1
            LEA DI , DATA2
            MOV CX , 7
    REPE    CMPSB
            JNZ MSG
            LEA DX,GUD
            JMP DISP
    MSG:    LEA AX, BAD
    DISP:   MOV AH, 09H
            INT 21H
            MOV AH,4CH
            INT 21H
            END
```

## OBSERVATIONS

Strings are: WELCOME and WELCOMM

Strings are not same : BAD

## RESULT

Program executed and obtained the result

**EXP NO. 7**

## PACKED BCD TO UNPACKED BCD

**AIM**

Write a program to convert packed bcd to unpacked bcd

**OBJECTIVES**

- To understand the use of Logical and shift instructions

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Initialize first string

**Step III** : Load the unpacked bcd into AL register

**Step IV**: Copy this data to another Register (BL)

**Step V:**Mask the higher nibble in AL register

**Step VI:** Mask lower nibble of BL register

**Step VII:** Shift BL 4 times right

**Step VIII:** **AL** and BL contain unpacked bcd

**Step IX:** Display

**Step X: Stop**

**PROGRAM:**

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
        MOV AX , @DATA
        MOV DS, AX
        MOV AL,95H
        MOV BL,AL
```

```
AND AL,0FH
AND BL, 0F0H
MOV CL,04H
SHR BL,CL
MOV AH,4CH
INT 21H
END
```

**OBSERVATIONS**

Input data : 45

Output data 05 04

**RESULT**

Program executed and obtained the result

# EXP No. 8
# SQUARE OF A NUMBER

**AIM**: Write a program to find the square root of a number

**OBJECTIVES**

To understand the use of arithmetic procedure and macro instructions

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Display a message for reading data using macro

**Step III** : Read a number using procedure

**Step IV**: Find square

**Step V:**Display message

**Step VI:** Display result

**Step VII: Stop**

**PROGARM CODE:**

```
;square of a number
.model small
.stack 100h
Printstring    macro mes
        mov dx,offset mes
        mov ax,0900h
         int 21h
        endm
.data
    cr equ 0dh
    lf equ 0ah
    mes1 db'Enter the number:$'
     mes2 db,cr,lf,'The square is:$'
     n db ?
    res dw 0
    result db 20 dup(0)
```

```
.code
    mov ax,@data
    mov ds,ax
    printstring mes1
    call read
    mov ch,00h
    mov bl,n
    mov al,n
    mul bl
    call disp
    printstring mes2
    printstring result
    mov ax,4c00h
    int 21h
read proc
        mov ax,0100h
        int 21h
        sub al,30h
        mov bh,0ah
        mov bl,00h
        mul bh
        mov n,al
        mov ax,0100h
        int 21h
        sub al,30h
        add n,al
        ret
read    endp

disp proc
            mov si,offset result
```

```
          mov cx,0000h
          mov bx,000ah
        l7:mov dx,0000h
          div bx
          add dl,30h
          push dx
          inc cx
          cmp ax,bx
          jge l7
          add al,30h
          mov [si],al
        l5:pop ax
          inc si
          mov [si],al
          loop l5
          inc si
          mov al,'$'
          mov [si],al
          ret
disp      endp
          end
```

**Observation**

Enter the number:08
The square is:64

**RESULT** **:**The program has been executed successfully and result is obtained

**EXP NO 9**

**SUM OF ‘N’ NUMBERS**

**AIM**: Write a program to find the sum of ‘n’ numbers

**OBJECTIVE :** Understanding the usage of arithmetic, macro and procedure instructions

**ALGORITHM**

**Step I**: Initialize the data segment memory.

**Step II :** Display a message for reading data using macro

**Step III** : Read a number using procedure

**Step IV**: Find sum of numbers up to that number

**Step V:**Display message

**Step VI:** Display result

**Step VII: Stop**

**PROGRAM CODE**:

```
;sum of natural number upto n
     printstring macro arg
     mov ah,09h
     mov dx,offset arg
     int 21h
     endm
.model small
.data
     n db ?
     outstring db 20 dup(0)
     num db ?
     cr equ 0dh
     lf equ 0ah
     mes1 db "Enter the number$"
     mes2 db,cr,lf,"The sum is $"
```

```
.code
      mov ax,@data
      mov ds,ax
      printstring mes1
      call read
      mov cl,n
      mov ch,00h
      mov bx,0001h
      mov ax,0000h
   l1:add ax,bx
      inc bx
      loop l1
      call disp
      printstring mes2
      printstring outstring
      mov ax,4c00h
      int 21h
read proc
      mov ah,01h
      int 21h
      sub al,30h
      mov bl,0ah
      mul bl
      mov n,al
      mov ah,01h
      int 21h
      sub al,30h
      add n,al
      ret
read endp
```

```
disp proc
    mov si,offset outstring
    mov cx,0000h
    mov bx,000ah
  l2:mov dx,0000h
    div bx
    add dl,30h
    push dx
    inc cx
    cmp ax,000ah
    jge l2
    add al,30h
    mov [si],al
  l3:pop ax
    inc si
    mov [si],al
    loop l3
    inc si
    mov bl,24h
    mov [si],bl
    ret
disp endp
    end
```

**Observation** :

Enter the number03
The sum is 06

**RESULT**: The program has been executed successfully and result is ontained