



### Micro-Project Report

Institute Code: 0078

Academic Year: 2022-23

Course: Microprocessors (MIC)

Semester: 4

Class: SYCM-I

Program: Computer Technology

Course Code: 22415 Scheme: I

Date of Report: 27/03/2023

**Title of Micro-Project:** Write an ALP to write a procedure for performing Basic Arithmetic operations

#### 1.0 Problem Definition:

The objective of this project is to create a modular procedure in 8086 assembly language to perform basic arithmetic operations like addition, subtraction, multiplication, division square, cube and factorial, without taking input from the user. The procedure will be optimized for maximum efficiency and accuracy, and the result will be stored in a register or memory location as defined in the code. The program will include appropriate comments and documentation for future reference and can be used in other programs by simply calling it. The ultimate goal is to create a versatile and efficient procedure for basic arithmetic operations.

#### 2.0 Rationale:

Microprocessor is the main component of a computer where 8086 is the base of all upward developed processors till current processors. This course will cover the basics of 8086 and its architecture along with instruction set, assembly language programming with effective use of procedure and macros. This course also covers the architectural issues such as instruction set program and data types. On top that, the student are also introduced to the increasingly important area of parallel organisation. This subject serves as a basic to develop hardware related projects, This course will enable the students to inculcate assembly language programming concepts and methodology to solve problems.

#### 3.0 Aim /Benefits of Micro-Project:

The Aim/Objectives of the Micro-project is:

- To develop an understanding of assembly language programming and its applications in performing basic arithmetic operations.
- To implement an ALP that can perform operations such as addition, subtraction, multiplication, and division of two numbers.also find square, cube and factorial
- To optimize the ALP to improve the performance and efficiency of arithmetic operations.
- To test and debug the ALP to ensure accurate and reliable results.
- To document the ALP code and provide instructions for its use and modification.
- To explore the potential of the ALP in solving practical problems that require arithmetic operations, such as data processing and scientific calculations.
- To evaluate the limitations and challenges of using assembly language programming for arithmetic operations and compare it with other programming languages.

#### 4.0 Course Outcomes Achieved (COs):

CO.1: Write assembly language program for the given problem.

CO.2: Develop an assembly language program using assembler.

CO.3: Develop assembly language programs using procedures, macros and modular programming approach.

#### 5.0 Literature Review:

The 8086 microprocessor is capable of performing arithmetic operations such as addition, subtraction, multiplication, and division using the Arithmetic and Logic Unit (ALU) and the General Purpose Registers (GPR). The procedure for performing arithmetic operations in 8086 involves loading

the operands into the GPR, selecting the appropriate instruction, and executing the operation using the ALU. The result is stored back into the GPR. Assembly language programming is commonly used to perform arithmetic operations using the 8086 microprocessor.

## **6.0 Actual Methodology followed:**

1. Discussing on various topics and finalized one topic of ALP to write a procedure for performing Basic Arithmetic operations
2. Identify the specific arithmetic operations to be performed, such as addition, subtraction, multiplication, or division. also square,cube and factorial
3. Choose an appropriate assembly language and IDE (Integrated Development Environment) for writing and testing the ALP, such as MS-DOS,TASM,TLINK,TD.
4. Write the ALP code using the appropriate assembly language syntax and instructions to perform the chosen arithmetic operations.
5. Test the ALP code by inputting different sets of data and verifying that the output matches the expected results for each arithmetic operation.
6. Optimize the ALP code for efficiency and speed by reducing the number of instructions and minimizing memory usage.
7. Document the ALP code and its functionality, including input and output formats, data types, and any limitations or constraints.
8. Validate the ALP code by comparing it with existing solutions and verifying that it meets the requirements and specifications of the project.
9. Iterate and refine the ALP code as needed based on feedback and testing results.
10. Testing all the modules and making corrections as per guidelines suggested by Guide.
11. Finalised all modules of ALP to to perform basic operations.
12. Generated Final Output.

## **7.0 Actual Resources used:**

| S. No. | Name of Resource/material | Specifications  | Qty | Remarks                               |
|--------|---------------------------|---|-----|---------------------------------------|
| 1.     | Laptop                    | HP Elitebook 830 g3-Intel(R)<br>Core(TM) i5, 8 GB installed<br>RAM,SSD-256 GB,HDD-1TB | 01  | For project work                      |
| 2.     | Operating System          | Windows 11 Pro x64 bit  | 01  | For running ALP Tools                 |
| 3.     | Editor                    | MS-DOS :-1.4v   | 01  | For Editing and Making Changes of ALP |
| 4.     | Assembler                 | TASM :-1.4v   | 01  | For converting ALP to MLL             |
| 5.     | Linker                    | TLINK :-2.0v  | 01  | For linking different object files    |
| 6.     | Debugger                  | TD :-2.0v   | 01  | For finding and fix errors            |
| 7.     | Other software            | Microsoft Office Word   | 01  | For documentation                     |
| 8.     | Reference book:           | The 8088 & 8086 Microprocessor  | 01  | For studying MIC concepts             |

|    |          |   |    |                                    |
|----|----------|---|----|------------------------------------|
| 9. | Websites | www.geeksforgeeks.org<br>www.studytonight.com<br>www.tutorialspoint.com | 03 | For referring sample ALP & concept |
|----|----------|---|----|------------------------------------|

## 8.0 Outputs of the Micro-Project:

### ❖ CODE-

; Define the data segment

Data Segment

```
a db 08h      ; Input variable a
b db 04h      ; Input variable b
res_add db 00h ; Result of addition
res_sub db 00h ; Result of subtraction
res_mul dw ?   ; Result of multiplication
res_quo db 00h ; Result of division (quotient)
res_rem db 00h ; Result of division (remainder)
res_sqr dw ?   ; Result of squaring a
res_cube dw ?  ; Result of cubing b
res_fact dw ?  ; Result of factorial b
```

;End of Data Segment

Data ends

; Define the code segment

Code Segment

Assume CS:Code, DS:Data

Start:

```
; Data Segment Initialization
mov dx,Data
mov ds,dx
```

; Perform addition of a and b

call add1

; Perform subtraction of b from a

call sub1

; Perform multiplication of a and b

call mul1

; Perform division of a by b

call div1

; Perform squaring of a

call sqr

; Perform cubing of b

call cube1

; Perform factorial of b

```
call FACTORIAL
```

```
; Exit the program
```

```
mov ah,4ch
```

```
int 21h
```

```
; Procedure to add a and b
```

```
add1 PROC NEAR
```

```
    mov al,a      ; Move the value of a to the AL register
```

```
    mov bl,b      ; Move the value of b to the BL register
```

```
    ; Add the values in AL and BL and store the result in AL
```

```
    add al,bl
```

```
    ; Move the result of the addition to the res_add variable
```

```
    mov res_add,al
```

```
    ; Return from the procedure
```

```
    RET
```

```
endp
```

```
; Procedure to subtract b from a
```

```
sub1 PROC NEAR
```

```
    mov al,a      ; Move the value of a to the AL register
```

```
    mov bl,b      ; Move the value of b to the BL register
```

```
    ; Subtract the value of BL from AL and store the result in AL
```

```
    sub al,bl
```

```
    ; Move the result of the subtraction to the res_sub variable
```

```
    mov res_sub,al
```

```
    ; Return from the procedure
```

```
    RET
```

```
endp
```

```
; Procedure to multiply a and b
```

```
mul1 PROC NEAR
```

```
    mov al,a      ; Move the value of a to the AL register
```

```
    mov bl,b      ; Move the value of b to the BL register
```

```
    ; Multiply the values in AL and BL and store the result in AX
```

```
    mul bl
```

```
    ; Move the result of the multiplication to the res_mul variable
```

```
    mov res_mul,ax
```

```
    ; Return from the procedure
```

```
    RET
```

```
endp
```

```
; Procedure to divide a by b
```

```
div1 PROC NEAR
```

```
    mov al,a      ; Move the value of a to the AL register
```

```
    mov bl,b      ; Move the value of b to the BL register
    ; Divide the value of AL by BL and store the quotient in AL and remainder in AH
    div bl
    ; Move the quotient to the res_quo variable
    mov res_quo,al
    ; Move the remainder to the res_rem variable
    mov res_rem,ah
    ; Return from the procedure
    RET
endp
```

; Procedure to find square a

```
sqr PROC NEAR
    mov al,a      ; Move the value of a to AL
    mov bl,a      ; Move the value of a to BL
    mul bl       ; Multiply the values of AL and BL
    mov res_sqr,ax ; Move the result to the variable res_sqr
    RET          ; Return from the procedure
endp
```

; Procedure to find cube b

```
cube1 PROC NEAR
    mov al,b ; Move the value of b to AL
    mov bl,b ; Move the value of b to BL
    mul bl      ; Multiply the values of AL and BL
    mul bl      ; Multiply the values of AL and BL
    mov res_cube,ax ; Move the result to the variable res_cube
    RET          ; Return from the procedure
endp
```

; Procedure to calculate the factorial of a number

```
FACTORIAL PROC NEAR
    mov al, 1      ; Initialize result to 1
    mov bl, b      ; Start with input value
    FACT_LOOP:     ; Loop to perform multiplication
        imul bl      ; Multiply by current value of b
        dec bl       ; Decrement b
        jnz FACT_LOOP ; Continue until b reaches 0
    mov res_fact, ax ; Store result in res_fact
    RET
ENDP
```

; End of code segment

Code ends

end Start

## ❖ OUTPUT-

### ▪ After Addition Result-

File Edit View Run Breakpoints Data Options Window Help READY

[CPU] - CPU 80486

| Address | OpCode   | Instruction   | Value |
|---------|----------|---------------|-------|
| cs:0021 | 8A1E0100 | mov bl,[0001] |       |
| cs:0025 | 02C3     | add al,bl     |       |
| cs:0027 | A20200   | mov [0002],al |       |
| cs:002A | C3       | ret           |       |
| cs:002B | A00000   | mov al,[0000] |       |
| cs:002E | 8A1E0100 | mov bl,[0001] |       |
| cs:0032 | 2AC3     | sub al,bl     |       |
| cs:0034 | A20300   | mov [0003],al |       |
| cs:0037 | C3       | ret           |       |
| cs:0038 | A00000   | mov al,[0000] |       |
| cs:003B | 8A1E0100 | mov bl,[0001] |       |
| cs:003F | F6E3     | mul bl        |       |
| cs:0041 | A30400   | mov [0004],ax |       |
| cs:0044 | C3       | ret           |       |
| cs:0045 | A00000   | mov al,[0000] |       |

ds:0002 = 00

Registers:

- ax 000C c=0
- bx 0004 z=0
- cx 0000 s=0
- dx 48AD o=0
- si 0000 p=1
- di 0000 a=0
- bp 0000 i=1
- sp FFFE d=0
- ds 48AD
- es 489D
- ss 48AC
- cs 48AE
- ip 0027

Memory Dump:

```

es:0000 CD 20 FF 9F 00 EA FF FF = f 
es:0008 AD DE E0 01 C5 15 AA 01 i x0|>0
es:0010 C5 15 89 02 20 10 92 01 +8e0>f0
es:0018 01 03 01 00 02 FF FF FF E0 E0
es:0020 FF FF FF FF FF FF FF FF FF

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

### ▪ After Subtraction Result-

File Edit View Run Breakpoints Data Options Window Help READY

[CPU] - CPU 80486

| Address | OpCode   | Instruction   | Value |
|---------|----------|---------------|-------|
| cs:002E | 8A1E0100 | mov bl,[0001] |       |
| cs:0032 | 2AC3     | sub al,bl     |       |
| cs:0034 | A20300   | mov [0003],al |       |
| cs:0037 | C3       | ret           |       |
| cs:0038 | A00000   | mov al,[0000] |       |
| cs:003B | 8A1E0100 | mov bl,[0001] |       |
| cs:003F | F6E3     | mul bl        |       |
| cs:0041 | A30400   | mov [0004],ax |       |
| cs:0044 | C3       | ret           |       |
| cs:0045 | A00000   | mov al,[0000] |       |
| cs:0048 | 8A1E0100 | mov bl,[0001] |       |
| cs:004C | F6F3     | div bl        |       |
| cs:004E | A20600   | mov [0006],al |       |
| cs:0051 | 88260700 | mov [0007],ah |       |
| cs:0055 | C3       | ret           |       |

ds:0003 = 00

Registers:

- ax 0004 c=0
- bx 0004 z=0
- cx 0000 s=0
- dx 48AD o=0
- si 0000 p=0
- di 0000 a=0
- bp 0000 i=1
- sp FFFE d=0
- ds 48AD
- es 489D
- ss 48AC
- cs 48AE
- ip 0034

Memory Dump:

```

es:0000 CD 20 FF 9F 00 EA FF FF = f 
es:0008 AD DE E0 01 C5 15 AA 01 i x0|>0
es:0010 C5 15 89 02 20 10 92 01 +8e0>f0
es:0018 01 03 01 00 02 FF FF FF E0 E0
es:0020 FF FF FF FF FF FF FF FF

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

▪ After Multiplication Result-

```

File Edit View Run Breakpoints Data Options Window Help READY
[CPU] - CPU 80486 ds:0004 = 0000 1 [ ]
cs:0038 A00000 mov al,[0000]
cs:003B 8A1E0100 mov bl,[0001]
cs:003F F6E3 mul bl
cs:0041 A30400 mov [0004],ax
cs:0044 C3 ret
cs:0045 A00000 mov al,[0000]
cs:0048 8A1E0100 mov bl,[0001]
cs:004C F6F3 div bl
cs:004E A20600 mov [0006],al
cs:0051 88260700 mov [0007],ah
cs:0055 C3 ret
cs:0056 A00000 mov al,[0000]
cs:0059 8A1E0000 mov bl,[0000]
cs:005D F6E3 mul bl
cs:005F A30800 mov [0008],ax
[ ]
es:0000 CD 20 FF 9F 00 EA FF FF = f ?
es:0008 AD DE E0 01 C5 15 AA 01 i |x0|S70
es:0010 C5 15 89 02 20 10 92 01 +Se0 ►R0
es:0018 01 03 01 00 02 FF FF FF 0000
es:0020 FF FF FF FF FF FF FF FF
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000
ss:FFFE►000E

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

▪ After Division Result-

```

File Edit View Run Breakpoints Data Options Window Help READY
[CPU] - CPU 80486 ds:0007 = 00 1 [ ]
cs:0045 A00000 mov al,[0000]
cs:0048 8A1E0100 mov bl,[0001]
cs:004C F6F3 div bl
cs:004E A20600 mov [0006],al
cs:0051 88260700 mov [0007],ah
cs:0055 C3 ret
cs:0056 A00000 mov al,[0000]
cs:0059 8A1E0000 mov bl,[0000]
cs:005D F6E3 mul bl
cs:005F A30800 mov [0008],ax
cs:0062 C3 ret
cs:0063 A00100 mov al,[0001]
cs:0066 8A1E0100 mov bl,[0001]
cs:006A F6E3 mul bl
cs:006C F6E3 mul bl
[ ]
es:0000 CD 20 FF 9F 00 EA FF FF = f ?
es:0008 AD DE E0 01 C5 15 AA 01 i |x0|S70
es:0010 C5 15 89 02 20 10 92 01 +Se0 ►R0
es:0018 01 03 01 00 02 FF FF FF 0000
es:0020 FF FF FF FF FF FF FF
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000
ss:FFFE►0011

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

▪ After Finding Square Result-

The screenshot shows a debugger interface with the CPU window selected. The assembly code in the CPU window is:

```

[CPU] - CPU 80486
ds:0008 = 0000 1 [↑]
cs:0056 A00000 mov al,[0000]
cs:0059 8A1E0000 mov bl,[0000]
cs:005D F6E3 mul bl
cs:005F A30800 mov [0008],ax
cs:0062 C3 ret
cs:0063 A00100 mov al,[0001]
cs:0066 8A1E0100 mov bl,[0001]
cs:006A F6E3 mul bl
cs:006C F6E3 mul bl
cs:006E A30A00 mov [000A],ax
cs:0071 C3 ret
cs:0072 B001 mov al,01
cs:0074 8A1E0100 mov bl,[0001]
cs:0078 F6EB imul bl
cs:007A FECB dec bl

```

The registers and stack information on the right side of the debugger are:

|         |     |
|---------|-----|
| ax 0040 | c=0 |
| bx 0008 | z=0 |
| cx 0000 | s=0 |
| dx 48AD | o=0 |
| si 0000 | p=0 |
| di 0000 | a=0 |
| bp 0000 | i=1 |
| sp FFFE | d=0 |
| ds 48AD |     |
| es 489D |     |
| ss 48AC |     |
| cs 48AE |     |
| ip 005F |     |

The stack dump at the bottom shows memory from es:0000 to es:0020:

```

es:0000 CD 20 FF 9F 00 EA FF FF = f ?
es:0008 AD DE E0 01 C5 15 AA 01 i [x0] S-?
es:0010 C5 15 89 02 20 10 92 01 +S?e? ►f?
es:0018 01 03 01 00 02 FF FF FF ???
es:0020 FF FF FF FF FF FF FF FF FF

```

Registers and stack dump at the bottom:

|              |  |
|--------------|--|
| ss:0006 706C |  |
| ss:0004 6568 |  |
| ss:0002 6474 |  |
| ss:0000 0000 |  |
| ss:FFFE►0014 |  |

Keyboard shortcuts at the bottom:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

▪ After Finding Cube Result-

The screenshot shows a debugger interface with the CPU window selected. The assembly code in the CPU window is:

```

[CPU] - CPU 80486
ds:000A = 0000 1 [↑]
cs:0063 A00100 mov al,[0001]
cs:0066 8A1E0100 mov bl,[0001]
cs:006A F6E3 mul bl
cs:006C F6E3 mul bl
cs:006E A30A00 mov [000A],ax
cs:0071 C3 ret
cs:0072 B001 mov al,01
cs:0074 8A1E0100 mov bl,[0001]
cs:0078 F6EB imul bl
cs:007A FECB dec bl
cs:007C 75FA jne 0078
cs:007E A30C00 mov [000C],ax
cs:0081 C3 ret
cs:0082 0000 add [bx+sil],al
cs:0084 0000 add [bx+sil],al

```

The registers and stack information on the right side of the debugger are:

|         |     |
|---------|-----|
| ax 0040 | c=0 |
| bx 0004 | z=0 |
| cx 0000 | s=0 |
| dx 48AD | o=0 |
| si 0000 | p=0 |
| di 0000 | a=0 |
| bp 0000 | i=1 |
| sp FFFE | d=0 |
| ds 48AD |     |
| es 489D |     |
| ss 48AC |     |
| cs 48AE |     |
| ip 006E |     |

The stack dump at the bottom shows memory from es:0000 to es:0020:

```

es:0000 CD 20 FF 9F 00 EA FF FF = f ?
es:0008 AD DE E0 01 C5 15 AA 01 i [x0] S-?
es:0010 C5 15 89 02 20 10 92 01 +S?e? ►f?
es:0018 01 03 01 00 02 FF FF FF ???
es:0020 FF FF FF FF FF FF FF FF

```

Registers and stack dump at the bottom:

|              |  |
|--------------|--|
| ss:0006 706C |  |
| ss:0004 6568 |  |
| ss:0002 6474 |  |
| ss:0000 0000 |  |
| ss:FFFE►0017 |  |

Keyboard shortcuts at the bottom:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

▪ After Finding Factorial Result-

```

[CPU 80486] ds:000C = 0000 1 [+] READY
File Edit View Run Breakpoints Data Options Window Help
[CPU 80486] ds:000C = 0000 1 [+] READY
cs:0072 B001      mov    al,01
cs:0074 8A1E0100  mov    bl,[0001]
cs:0078 F6EB      imul   bl
cs:007A FECB      dec    bl
cs:007C 75FA      jne    0078
cs:007E A30C00  mov    [000C],ax
cs:0081 C3        ret
cs:0082 0000      add    [bx+sil,al]
cs:0084 0000      add    [bx+sil,al]
cs:0086 0000      add    [bx+sil,al]
cs:0088 0000      add    [bx+sil,al]
cs:008A 0000      add    [bx+sil,al]
cs:008C 0000      add    [bx+sil,al]
cs:008E 0000      add    [bx+sil,al]
cs:0090 0000      add    [bx+sil,al]
es:0000 CD 20 FF 9F 00 EA FF FF = f ?
es:0008 AD DE E0 01 C5 15 AA 01 i |x0|S-0
es:0010 C5 15 89 02 20 10 92 01 |$e| ►f0
es:0018 01 03 01 00 02 FF FF FF 00 00
es:0020 FF FF FF FF FF FF FF FF FF
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000
ss:FFFE►001A

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

## 9.0 Skill Developed / Learning outcome of this Micro-Project:

### Technical Skills:

- Understanding of 8086 microprocessor architecture
- Knowledge of Assembly Language Programming
- Knowledge of arithmetic operations
- Understanding the procedures
- Debugging skills
- Knowledge of software tools

### Soft Skills:

- Leadership and decision making.
- Working in team.
- Time management.
- Presentation and writing skills.

## 10.0 Applications of Micro Project:

- Simple calculator
- Data analysis
- Embedded systems
- Gaming
- Educational purposes

**11.0 Name of Group Members:**

| <b>Enrolment No.</b> | <b>Roll No.</b> | <b>Seat No.</b> | <b>Name of Students</b>         | <b>Student Signature</b> |
|----------------------|-----------------|-----------------|---------------------------------|--------------------------|
| 2100780077           | 06              | 385977          | Baldota Kalash Sachin           |                          |
| 2100780081           | 10              | 385981          | Bhalerao Mukunda Chandrashekhar |                          |
| 2100780086           | 15              | 385986          | Dhakane Aditya Arun             |                          |

Date: 27/03/2023

Evaluated by: Dated Signature of Guide: \_\_\_\_\_

Name of Guide: Ms. P.A.Agrawal