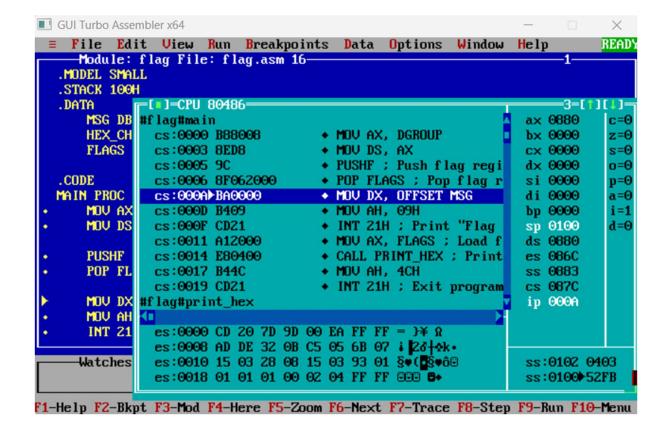# Experiment 6

Name:Pratik Chavan

Div/Batch:A/A1   Roll No.07

.MODEL SMALL

.STACK 100H

.DATA

   MSG DB 'Flag Register: $' *; Message to display*

   HEX_CHARS DB '0123456789ABCDEF' *; Lookup table for hex digits*

   FLAGS DW ? *; Variable to store flag register value*

.CODE

MAIN PROC

   MOV AX, DGROUP

   MOV DS, AX

   PUSHF   *; Push flag register onto the stack*

   POP FLAGS   *; Pop flag register into FLAGS variable*

   MOV DX, OFFSET MSG

   MOV AH, 09H

   INT 21H   *; Print* "Flag Register: "

   MOV AX, FLAGS *; Load flag register value into AX*

   CALL PRINT_HEX *; Print the flag register in hexadecimal format*

   MOV AH, 4CH

```asm
    INT 21H        ; Exit program

MAIN ENDP


; -------------------------
; Print 16-bit Hex Procedure
; -------------------------
PRINT_HEX PROC
    MOV CX, 4      ; We have 4 hex digits (16-bit / 4-bit each)
    MOV BX, 12     ; Bit shift amount (12, 8, 4, 0)


HEX_LOOP:
    MOV DX, AX     ; Copy AX value
    MOV CL, BL     ; Move shift count into CL (Fix for SHR error)
    SHR DX, CL     ; Shift right to isolate one hex digit
    AND DX, 0FH    ; Mask the lower 4 bits
    MOV SI, DX     ; Move index to SI
    MOV DL, [HEX_CHARS + SI] ; Convert to ASCII hex character
    MOV AH, 02H
    INT 21H        ; Print the hex digit
    SUB BX, 4      ; Move to the next hex digit
    LOOP HEX_LOOP  ; Repeat until all digits are printed


    RET
PRINT_HEX ENDP


END MAIN


OUTPUT:
```

≡ File Edit View Run Breakpoints Data Options Window Help   READY

─Module: flag File: flag.asm 16──────────────────────1─

```
.MODEL SMALL
.STACK 100H
.DATA
    MSG DB
    HEX_CH
    FLAGS

.CODE
MAIN PROC
  •   MOV AX
  •   MOV DS

  •   PUSHF
  •   POP FL

  ▶   MOV DX
  •   MOV AH
  •   INT 21
```

─[■]─CPU 80486───────────────────3─[↑][↓]─

```
#flag#main
cs:0000 B88008          ◆ MOV AX, DGROUP
cs:0003 8ED8            ◆ MOV DS, AX
cs:0005 9C              ◆ PUSHF ; Push flag regi
cs:0006 8F062000        ◆ POP FLAGS ; Pop flag r
cs:000A▶BA0000          ◆ MOV DX, OFFSET MSG
cs:000D B409            ◆ MOV AH, 09H
cs:000F CD21            ◆ INT 21H ; Print "Flag
cs:0011 A12000          ◆ MOV AX, FLAGS ; Load f
cs:0014 E80400          ◆ CALL PRINT_HEX ; Print
cs:0017 B44C            ◆ MOV AH, 4CH
cs:0019 CD21            ◆ INT 21H ; Exit program
#flag#print_hex
```

```
ax 0880    c=0
bx 0000    z=0
cx 0000    s=0
dx 0000    o=0
si 0000    p=0
di 0000    a=0
bp 0000    i=1
sp 0100    d=0
ds 0880
es 086C
ss 0883
cs 087C
ip 000A
```

```
es:0000 CD 20 7D 9D 00 EA FF FF = }¥ Ω
es:0008 AD DE 32 0B C5 05 6B 07 ¡ 2δ┤⇕k•
es:0010 15 03 28 08 15 03 93 01 §♥(◘§♥ôʘ
es:0018 01 01 01 00 02 04 FF FF ☺☺☺ ☻♦
```

```
ss:0102 0403
ss:0100▶52FB
```

─Watches─

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

Experiment 3

Name:Pratik Chavan

Div/Batch:A/A1   Roll No.07

```
.MODEL SMALL

.STACK 100H

.DATA

   NUM DB 5        ; Number for factorial (change this value as needed)

   FACT DW 1       ; Variable to store factorial result

   MSG DB 'Factorial: $'  ; Message to display before result


.CODE

MAIN PROC

   MOV AX, @DATA

   MOV DS, AX


   MOV AL, NUM     ; Load number in AL

   CBW             ; Convert AL to AX (sign-extend)

   CALL FACTORIAL   ; Call factorial procedure


   MOV DX, OFFSET MSG

   MOV AH, 09H

   INT 21H          ; Print message


   CALL PRINT_NUM   ; Print the factorial result


   MOV AH, 4CH
```

```
    INT 21H        ; Exit program
MAIN ENDP


; Factorial Procedure
FACTORIAL PROC
    MOV CX, AX      ; Move number to CX for loop counter
    MOV AX, 1       ; Initialize AX = 1 (Factorial starts at 1)


FACTORIAL_LOOP:
    MUL CX          ; AX = AX * CX
    LOOP FACTORIAL_LOOP


    MOV FACT, AX    ; Store the result in FACT
    RET
FACTORIAL ENDP


; Print Number Procedure
PRINT_NUM PROC
    MOV AX, FACT    ; Load factorial result
    MOV CX, 0       ; Clear CX (digit counter)


NEXT_DIGIT:
    MOV DX, 0
    MOV BX, 10
    DIV BX          ; AX / 10 → Quotient in AX, Remainder in DX
    PUSH DX         ; Push remainder (digit) onto stack
    INC CX          ; Increment digit counter
    TEST AX, AX     ; Check if AX is zero
```

```
    JNZ NEXT_DIGIT   ; If not, continue extracting digits


PRINT_LOOP:

    POP DX          ; Get digit from stack

    ADD DL, '0'     ; Convert to ASCII

    MOV AH, 02H

    INT 21H         ; Print digit

    LOOP PRINT_LOOP  ; Repeat for remaining digits


    RET

PRINT_NUM ENDP


END MAIN
```
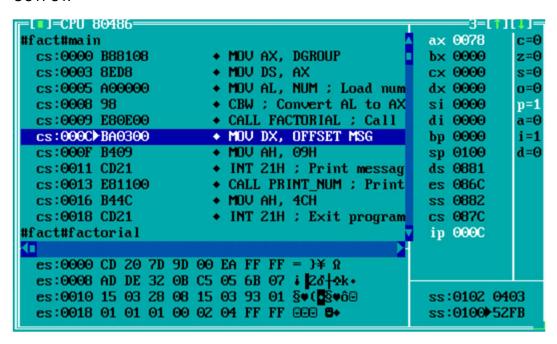
OUTPUT:

Experiment 3

Name:Pratik Chavan

Div/Batch:A/A1   Roll No.07

```
.MODEL SMALL
.STACK 100H
.DATA
    ARRAY DB 10, 25, 15, 40, 5, 30, 50, 20
    LEN EQU $ - ARRAY
    MIN DB 0
    MAX DB 0

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV SI, 0
    MOV AL, ARRAY[SI]
    MOV MIN, AL
    MOV MAX, AL

FIND_MIN_MAX:
    MOV AL, ARRAY[SI]
    CMP AL, MAX
    JG UPDATE_MAX
    CMP AL, MIN
```

```asm
    JL UPDATE_MIN

    JMP NEXT_ELEMENT


UPDATE_MAX:

    MOV MAX, AL

    JMP NEXT_ELEMENT


UPDATE_MIN:

    MOV MIN, AL


NEXT_ELEMENT:

    INC SI

    CMP SI, LEN

    JL FIND_MIN_MAX


    MOV AX, 4C00H

    INT 21H


MAIN ENDP

END MAIN
```

OUTPUT: