CSE 2006

MICROPROCESSOR AND INTERFACING

Task – 2

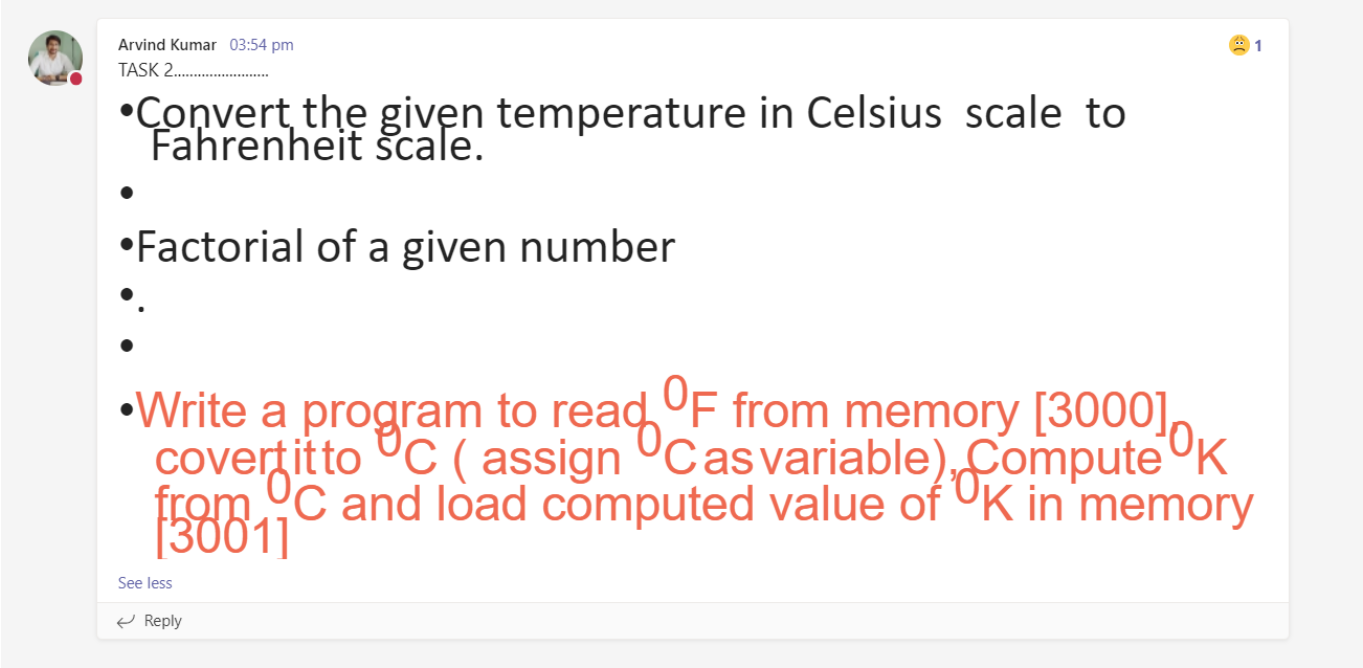
L11+L12 | SJT516

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105



1. **Aim:** To convert the given temperature in Celsius scale to Fahrenheit scale.

   Input: I'm inputting 37°.

   Algorithm: The algorithm to convert from Celsius to Fahrenheit is the temperature in Celsius times 9/5, plus 32

   Input: I'm inputting 37°.

   Code:

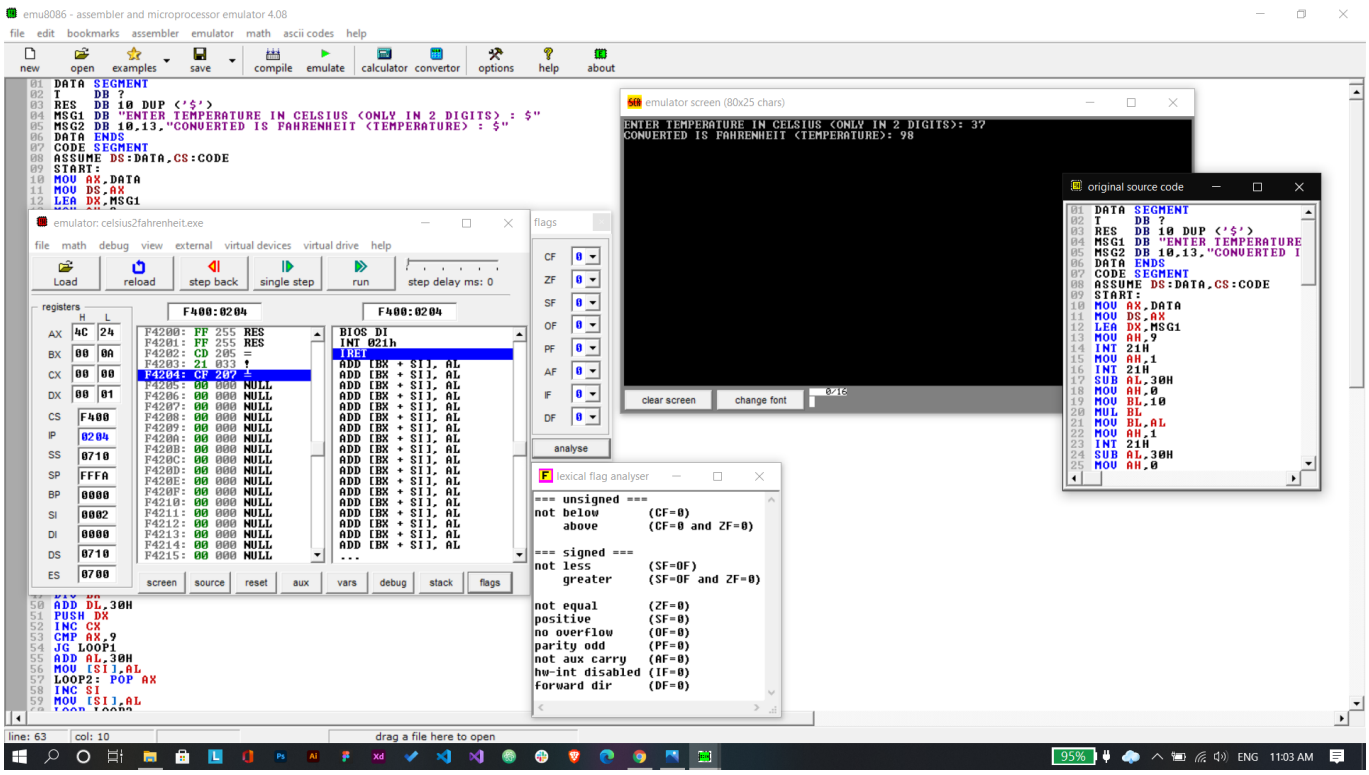
```
DATA SEGMENT
T    DB ?
RES  DB 10 DUP ('$')
MSG1 DB "ENTER TEMPERATURE IN CELSIUS (ONLY IN 2 DIGITS) : $"
MSG2 DB 10,13,"CONVERTED IS FAHRENHEIT (TEMPERATURE) : $"
DATA ENDS
CODE SEGMENT
ASSUME DS:DATA,CS:CODE
START:
MOV AX,DATA
MOV DS,AX
LEA DX,MSG1
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV AH,0
MOV BL,10
MUL BL
MOV BL,AL
MOV AH,1
```

sharadindu.adhikari2019@vitstudent.ac.in

```
INT 21H
SUB AL,30H
MOV AH,0
ADD AL,BL
MOV T,AL
MOV DL,9
MUL DL
MOV BL,5
DIV BL
MOV AH,0
ADD AL,32
LEA SI,RES
CALL HEX2DEC
LEA DX,MSG2
MOV AH,9
INT 21H
LEA DX,RES
MOV AH,9
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
HEX2DEC PROC NEAR
MOV CX,0
MOV BX,10
LOOP1: MOV DX,0
DIV BX
ADD DL,30H
PUSH DX
INC CX
CMP AX,9
JG LOOP1
ADD AL,30H
MOV [SI],AL
LOOP2: POP AX
INC SI
MOV [SI],AL
LOOP LOOP2
RET
HEX2DEC ENDP
END START
```

Output: AX=0068 (in degree C to degree F formula=>F=((C*9)/5)+32, here AX or C=37, therefore F=98

Screenshot along with Output window:



2. **Aim: To find factorial of a given number.**

Algorithm: i. Input the Number whose factorial is to be find and Store that Number in CX Register (Condition for LOOP Instruction).
ii. Insert 0001 in AX(Condition for MUL Instruction) and 0000 in DX Multiply CX with AX until CX become Zero(0) using LOOP Instruction.
iii. Copy the content of AX to memory location 0600.
iv. Copy the content of DX to memory location 0601.
v. Stop Execution.

Input: I'm inputting 10!

Code:

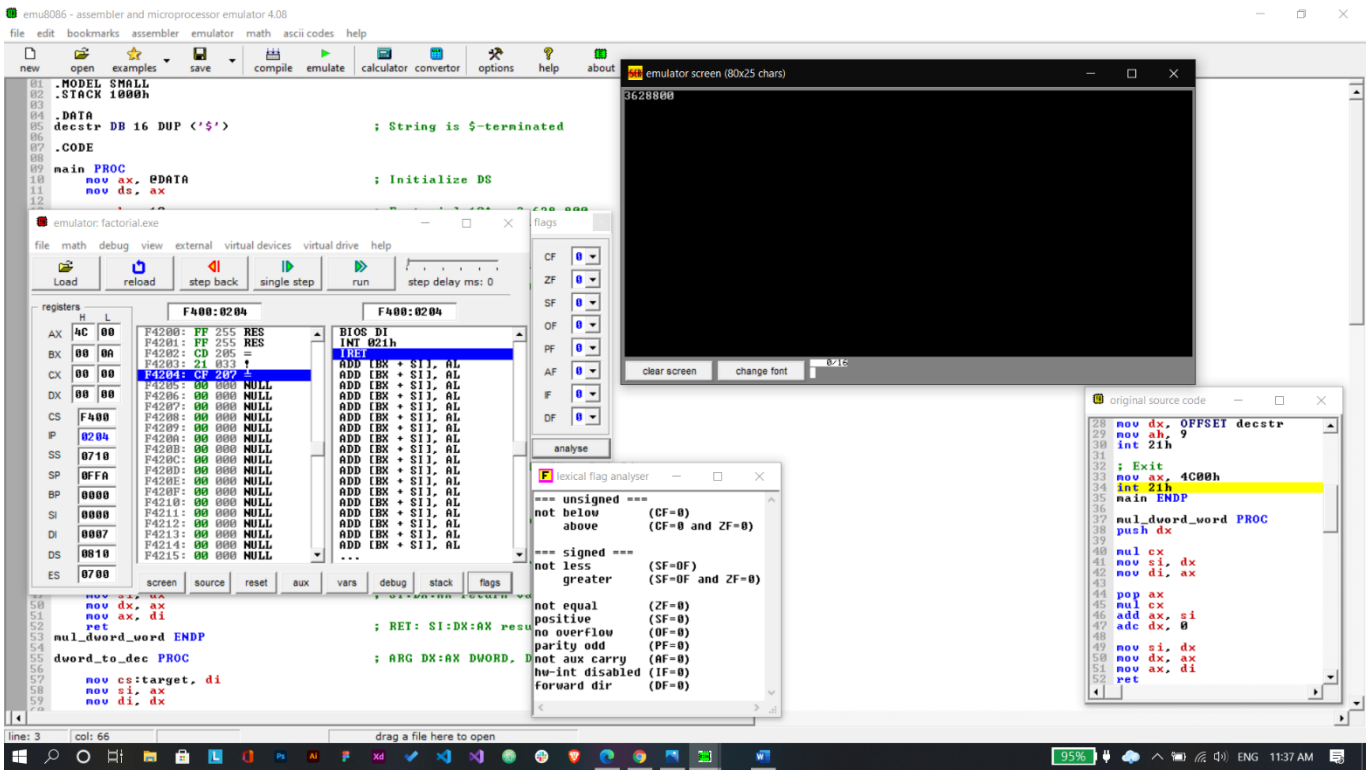
```
.MODEL SMALL
.STACK 1000h

.DATA
decstr DB 16 DUP ('$')                  ; String is $-terminated

.CODE

main PROC
    mov ax, @DATA                       ; Initialize DS
    mov ds, ax
```

```
    mov bx, 10                          ; Factorial 10! = 3.628.800
    xor dx, dx                          ; DX:AX=1 (first multiplicand)
    mov ax, 1                           ; Begin with 1

    ; for (dx:ax = 1, cx = 2; cx <= 10; cx++)
    mov cx, 2                           ; Incrementing multiplicator
    L1:
    call mul_dword_word                 ; DX:AX * CX -> DX:AX
    inc cx
    cmp cx, bx
    jbe L1                              ; While cx <= 10

    ; Print result
    mov di, OFFSET decstr
    call dword_to_dec
    mov dx, OFFSET decstr
    mov ah, 9
    int 21h

    ; Exit
    mov ax, 4C00h
    int 21h
main ENDP

mul_dword_word PROC                     ; DX:AX multiplicand, CX multiplier
    push dx

    mul cx                              ; AX * CX -> DX:AX
    mov si, dx                          ; Store high result
    mov di, ax                          ; Low result won't be changed anymore

    pop ax                              ; High word
    mul cx                              ; AX * CX -> DX:AX
    add ax, si                          ; Add high result from last mul to low resul
t here
    adc dx, 0

    mov si, dx                          ; SI:DX:AX return value
    mov dx, ax
    mov ax, di
    ret                                 ; RET: SI:DX:AX result
mul_dword_word ENDP

dword_to_dec PROC                       ; ARG DX:AX DWORD, DI: offset of string

    mov cs:target, di
    mov si, ax
    mov di, dx

    ; First Loop: get digits and push them
```

```
    mov cs:counter, 0
    mov bx, 10
    LL1:
    inc cs:counter
    xor dx, dx
    mov ax, di                          ; High WORD
    mov cx, ax
    div bx                              ; DX:AX / BX -> AX Remainder DX
    mov di, ax                          ; Store new high word
    mul bx                              ; AX * BX -> DX:AX
    sub cx, ax                          ; sub highest CX-divisible value

    mov dx, cx
    mov ax, si                          ; Low WORD
    div bx                              ; DX:AX / BX -> AX Remainder DX
    or dl, 30h                          ; Convert remainder to ASCII
    push dx                             ; Store remainder
    mov si, ax                          ; Store new low WORD

    or ax, di                           ; Anything more to process?
    jnz LL1                             ; yes: jump to LL1 above

    ; Second Loop: get back digits in reversed order
    mov di, cs:target
    mov cx, cs:counter
    LL2:
    pop ax
    mov [di], al
    inc di
    loop LL2
    mov BYTE PTR [di], '$'              ; Terminator for INT 21h/09h

    ret
    counter dw 0
    target dw 0
dword_to_dec ENDP
```

Output: 3628800 (as is the desired output of 10!)

Screenshot alongside output window:



3. **Aim: To write a program to read °F from memory [3000] and convert it to °C. Further to compute °K from °C and load computed value of K in memory [3001].**

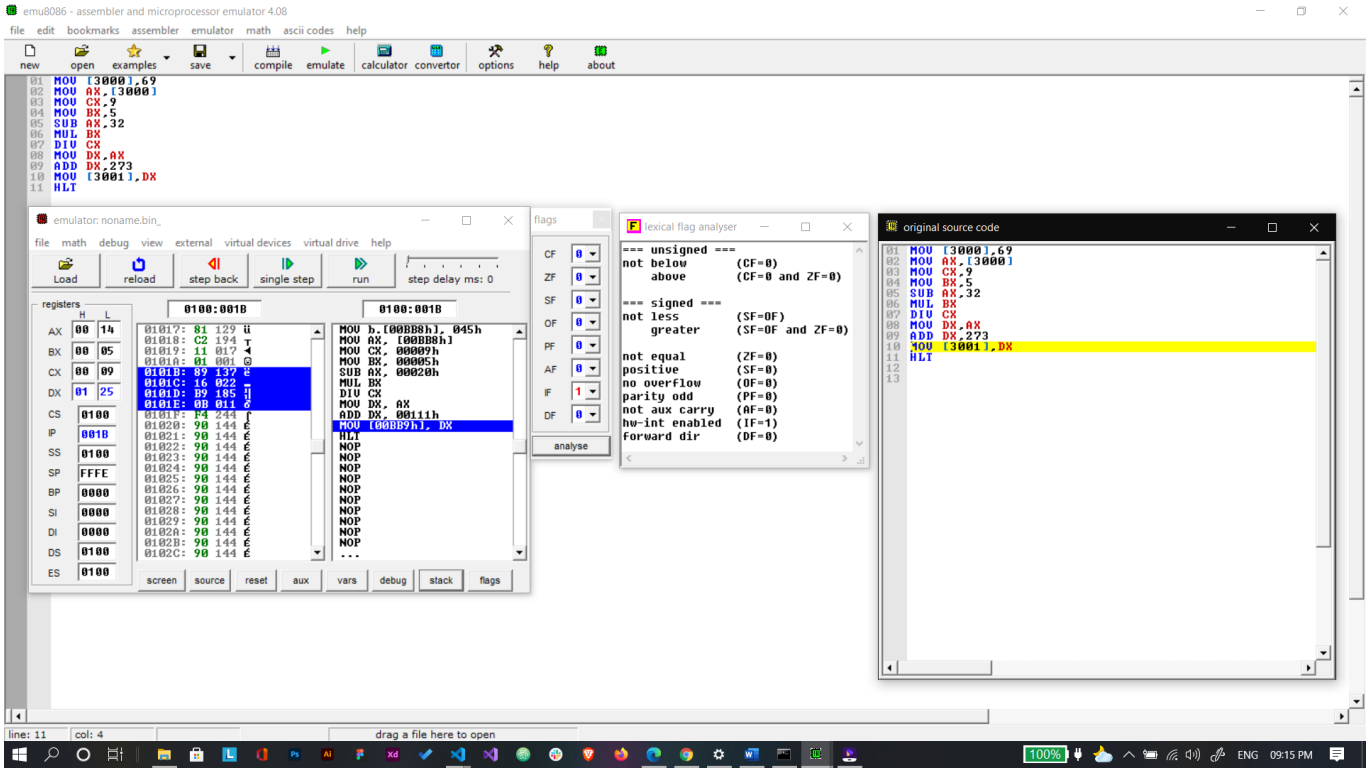
   Algorithm: Same formula, as described in the 1st question. Just gotta store the value, in addition.

   Code:

```
MOV [3000],69
MOV AX,[3000]
MOV CX,9
MOV BX,5
SUB AX,32
MUL BX
DIV CX
MOV DX,AX
ADD DX,273
MOV [3001],DX
HLT
```

   Output: DX=[3001]=0125h=293 (in degree F to degree C formula=>C=((F-32)*5)/9,
   ;here AX or F=69 , therefore C=20.556~20 (i.e 14h in hexadecimal),
   ;and K=C+273=293 (i.e 125h in hexadecimal which is stored in [3001])

Screenshot:



Output: DX=[3001]=0125h=293 (in degree F to degree C formula=>C=((F-32)*5)/9,
; here AX or F=69 , therefore C=20.556~20 (i.e 14h in hexadecimal),
; and K=C+273=293 (i.e 125h in hexadecimal which is stored in [3001])