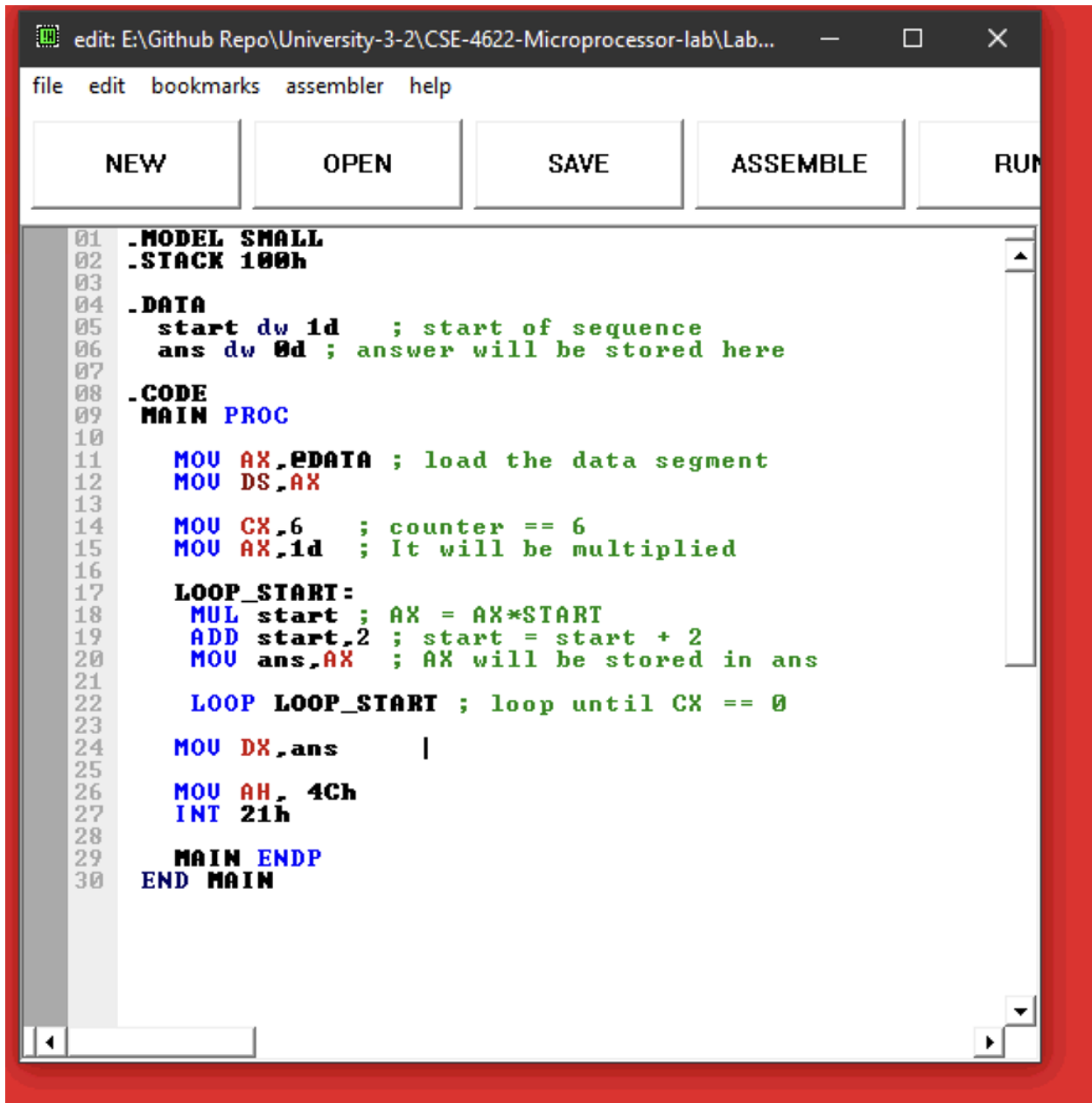


TASK-1

In the first task we had to get the product of the series 1,3,5,7,9,11.



```
01 .MODEL SMALL
02 .STACK 100h
03
04 .DATA
05     start dw 1d    ; start of sequence
06     ans dw 0d      ; answer will be stored here
07
08 .CODE
09 MAIN PROC
10
11     MOV AX,EDATA    ; load the data segment
12     MOV DS,AX
13
14     MOV CX,6        ; counter == 6
15     MOV AX,1d       ; It will be multiplied
16
17     LOOP_START:
18         MUL start    ; AX = AX*START
19         ADD start,2   ; start = start + 2
20         MOV ans,AX    ; AX will be stored in ans
21
22         LOOP LOOP_START ; loop until CX == 0
23
24     MOV DX,ans
25
26     MOV AH, 4Ch
27     INT 21h
28
29     MAIN ENDP
30 END MAIN
```

Here I took a variable named "start", initialized to 1. This is the start of the sequence. We took a counter for the loop and initialized it to 6. Then just kept on looping until the counter gets to zero.

Logic was:

Start = 1

Ans = 0

AX = 1

LOOP

AX = AX*START

START=START+2

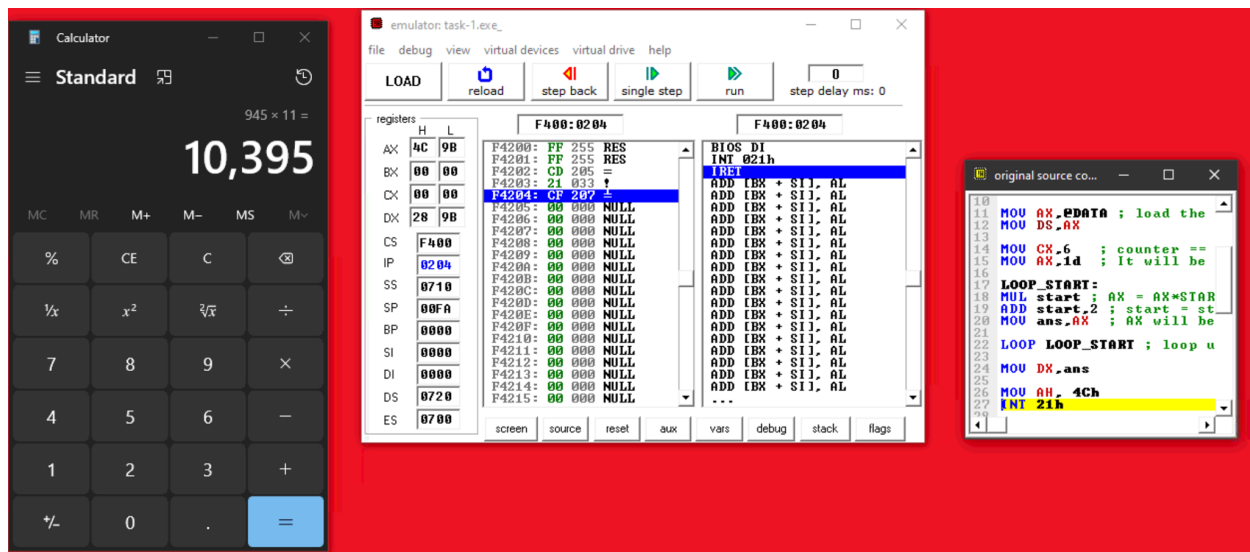
ANS = AX

Final ans is;

TASK-2

So on this problem I had to write two procedures that were called from the main. One to take input and the other to display output.

First the MAIN PROCEDURE:



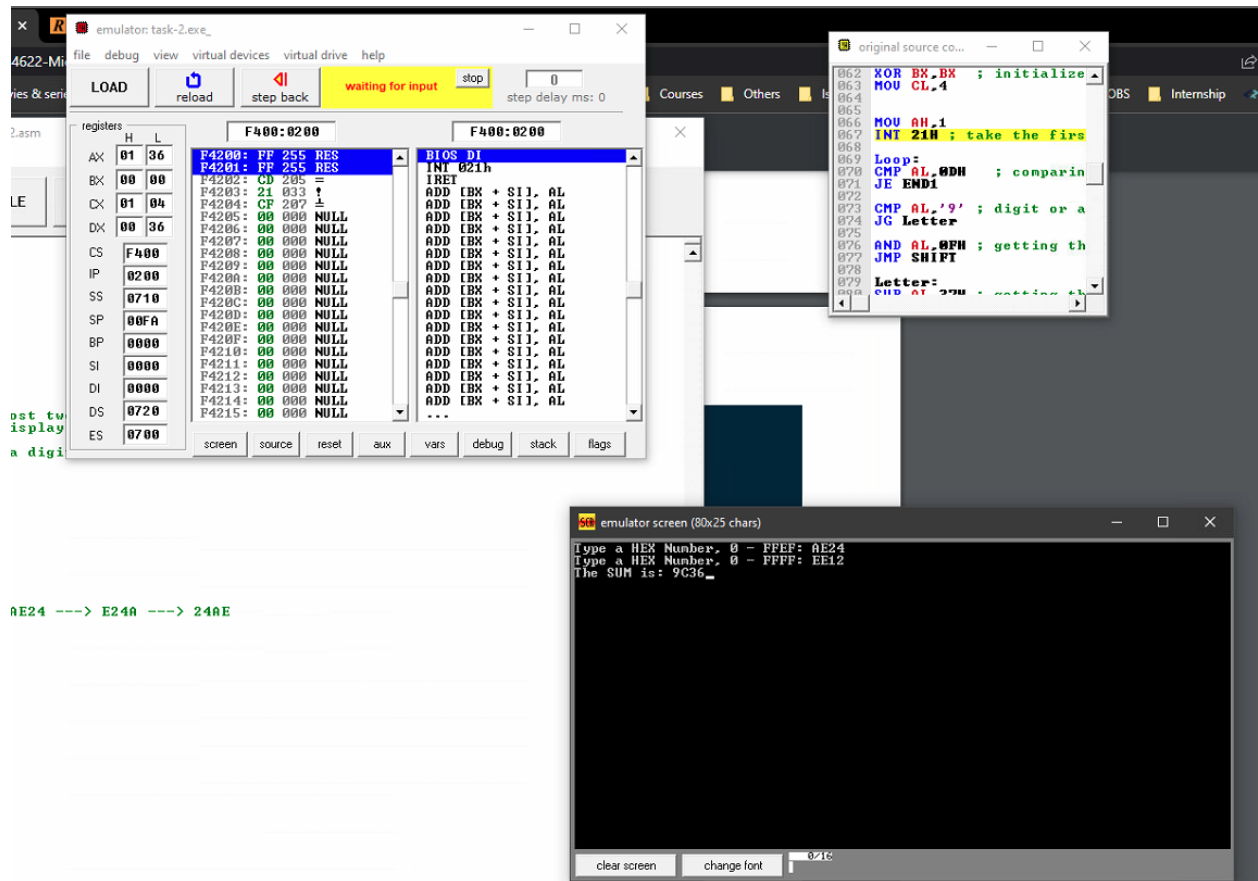
```

0001 .MODEL SMALL
0002
0003 .STACK 100h
0004
0005 .DATA
0006     hex1 DW ?
0007     hex2 DW ?
0008
0009     Prompt1 DB 'Type a HEX Number, 0 - FFEF: $' ; input message 1
0010     Prompt2 DB 'Type a HEX Number, 0 - FFFF: $' ; input message 2
0011     Prompt3 DB 10, 13, 'The SUM is: $' ; output message
0012
0013     counter db 4 ; number of digits in a hexnumber
0014
0015 .CODE
0016
0017 MAIN PROC
0018     MOV AX, @DATA ; loaded the data segment
0019     MOV DS, AX
0020
0021
0022     MOV AH, 9
0023     LEA DX, Prompt1 ; first message print
0024     INT 21h
0025
0026     CALL INHEX ; called the inhex procedure
0027     MOV hex1, BX ; hex1 a BX rakhtesi
0028
0029     ;print Carraige return and new line
0030     MOV AH, 2
0031     MOV DL, 0DH
0032     INT 21h
0033     MOV AH, 2
0034     MOV DL, 0AH
0035     INT 21h
0036
0037
0038     MOV AH, 9
0039     LEA DX, Prompt2 ; same as before
0040     INT 21h
0041
0042
0043     CALL INHEX
0044     MOV hex2, BX ; hex2 a BX rakhtesi
0045
0046
0047     MOV AH, 9
0048     LEA DX, Prompt3
0049     INT 21h
0050
0051
0052
0053     ADD BX, hex1
0054
0055     CALL OUTHEX ; showign result = hex1 + hex2
0056
0057
0058 MAIN ENDP
0059

```

Here we just displayed two prompts and called the inhex procedure to take input. After calculating the sum we called the outhex procedure to show the output.

RESULT:



NOW THE INHEX

```

060
061 IMHEX PROC
062 XOR BX,BX ; initialized zero
063 MOV CL,4
064
065
066 MOV AH,1
067 INT 21H ; take the first input digit
068
069 Loop:
070 CMP AL,0DH ; comparing if it is CR or not
071 JE END1
072
073 CMP AL,'9' ; digit or alphabet?
074 JG Letter
075
076 AND AL,0FH ; getting the hexa decimal value of digit
077 JMP SHIFT
078
079 Letter:
080 SUB AL,37H ; getting the hexa decimal value of letter
081
082 SHIFT:
083 SHL BX,CL ; shifting BX left by 4 bits
084 OR BL,AL ; and putting the latest input in the most right section of BL
085
086 INT 21H ; taking the next input
087
088 JMP Loop
089
090 END1:
091 RET
092
093 IMHEX ENDP
094
095

```

Here we used the BX to store the input. We had to keep track of two types of input: one is digit and the other one is alphabet. And we took the input using a loop and converted it to their respective hexadecimal value and stored it inside BX. We also had to left-shift the BX by 4 bits to take the input one after another.

NOW THE OUTHEX:

```

098 OUTHEX PROC
099
100     MOV CL,4    ; 4 digits to show
101
102 PRINT:
103     MOV DL,BH   ; getting the BH(the righmost two digits) to store inside DL
104     SHR DL,CL   ; Then shifting right to display only one digit
105
106     CMP DL,9    ; comparing to see if its a digit or letter
107     JG ALPHABET
108
109     ADD DL,30H   ; number or digit
110     JMP DIGIT
111
112 ALPHABET:
113     ADD DL,37h   ; letter A,B,C,D,E,F
114
115 DIGIT:
116     MOV AH,2
117     INT 21h
118
119     ROL BX,CL    ; rotating the ans , ans = AE24 ---> E24A ---> 24AE
120     DEC counter
121
122     CMP counter,0
123     JNE PRINT
124
125 RET
126
127 OUTHEX ENDP
128
129
130
131 END MAIN

```

OUTHEX is basically just taking one by one digits/alphabet from the BH and displaying it to the console. The process is almost similar to the INHEX. Here we had to convert the hexadecimal value to ASCII code.