Name:- Sravan Chittupalli

VJTI ID: 181060018

EXPERIMENT - 4

AIM

1. Design an 8086-assembly language program to compute the factorial of a positive integer ‘n’ using a recursive procedure. Integer n is a byte integer.

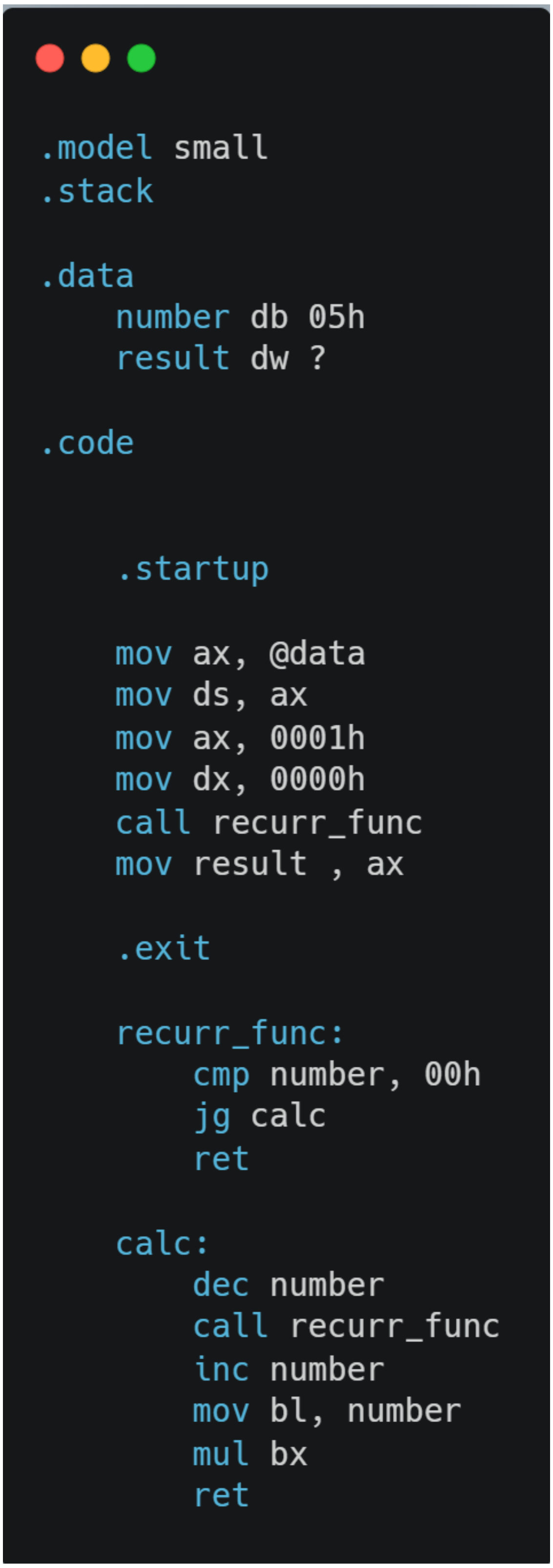
SOFTWARE

- EMU8086 emulator

ALGORITHM

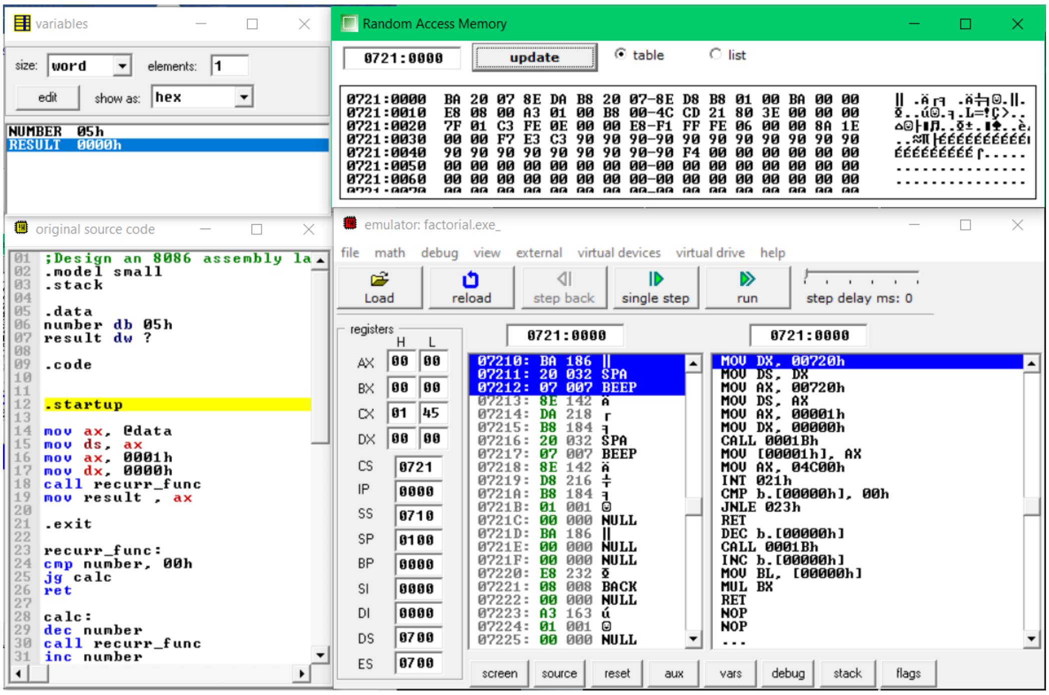
1. Input the Number whose factorial is to be found into variable called ‘number’.
2. Insert 0001 in AX (Condition for MUL Instruction) and 0000 in DX
3. Decrement number
4. Call recur_func till number is not equal to zero
5. Copy the content of AX to ‘result’ variable.
6. Stop Execution

CODE



```
.model small
.stack

.data
    number db 05h
    result dw ?

.code


    .startup

    mov ax, @data
    mov ds, ax
    mov ax, 0001h
    mov dx, 0000h
    call recurr_func
    mov result , ax

    .exit

    recurr_func:
        cmp number, 00h
        jg calc
        ret

    calc:
        dec number
        call recurr_func
        inc number
        mov bl, number
        mul bx
        ret
```

- **INPUT**



variables — size: word, elements: 1, show as: hex

```
NUMBER  05h
RESULT  0000h
```

original source code

```
;Design an 8086 assembly la
.model small
.stack

.data
number db 05h
result dw ?

.code


.startup

mov ax, @data
mov ds, ax
mov ax, 0001h
mov dx, 0000h
call recurr_func
mov result , ax

.exit

recurr_func:
cmp number, 00h
jg calc
ret

calc:
dec number
call recurr_func
inc number
```

Random Access Memory — 0721:0000, ◉ table ○ list

```
0721:0000  BA 20 07 8E DA B8 20 07-8E D8 B8 01 00 BA 00 00   ‖ .Äΰ┐ .Ä╪╕☺.‖..
0721:0010  E8 08 00 A3 01 00 B8 00-4C CD 21 80 3E 00 00 00   Φ..ú☺.╕.L═!Ç>...
0721:0020  7F 01 C3 FE 0E 00 00 E8-F1 FF FE 06 00 00 8A 1E   ⌂☺├■♫..Φ± ■♠..è▲
0721:0030  00 00 F7 E3 C3 90 90 90-90 90 90 90 90 90 90 90   ..≈π├ÉÉÉÉÉÉÉÉÉÉÉ
0721:0040  90 90 90 90 90 90 90 90-90 F4 00 00 00 00 00 00   ÉÉÉÉÉÉÉÉÉ⌠......
0721:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0721:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

emulator: factorial.exe_

```
registers   H   L
AX         00  00
BX         00  00
CX         01  45
DX         00  00
CS         0721
IP         0000
SS         0710
SP         0100
BP         0000
SI         0000
DI         0000
DS         0700
ES         0700
```

0721:0000

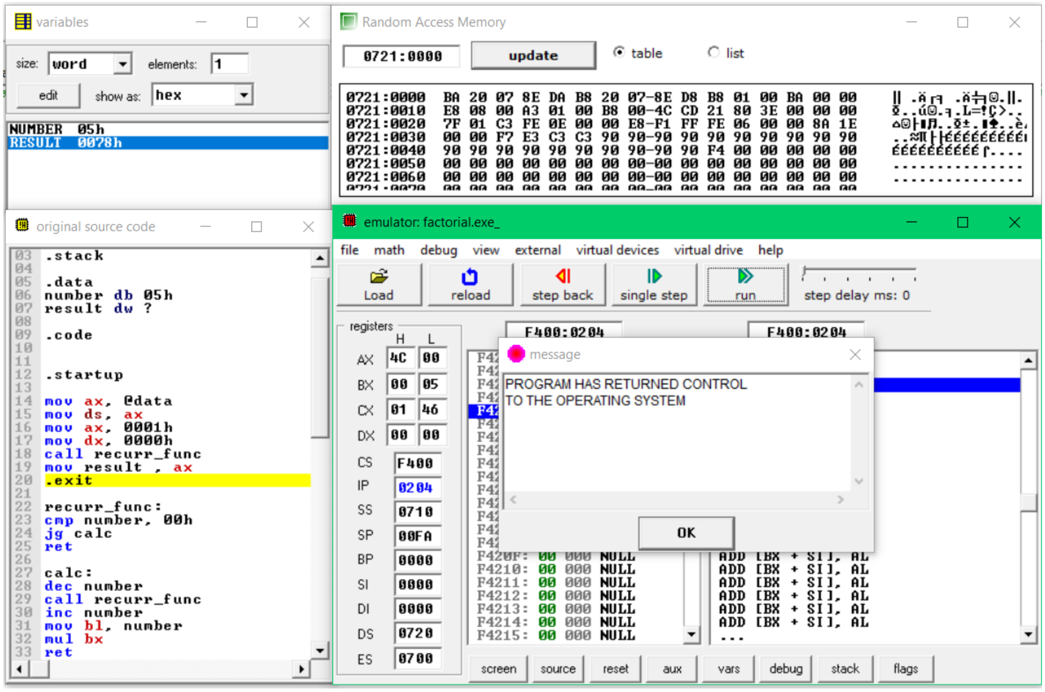
```
07210: BA 186 ‖
07211: 20 032 SPA
07212: 07 007 BEEP
07213: 8E 142 Ä
07214: DA 218 ┌
07215: B8 184 ╕
07216: 20 032 SPA
07217: 07 007 BEEP
07218: 8E 142 Ä
07219: D8 216 ╪
0721A: B8 184 ╕
0721B: 01 001 ☺
0721C: 00 000 NULL
0721D: BA 186 ‖
0721E: 00 000 NULL
0721F: 00 000 NULL
07220: E8 232 Φ
07221: 08 008 BACK
07222: 00 000 NULL
07223: A3 163 ú
07224: 01 001 ☺
07225: 00 000 NULL
```

0721:0000

```
MOV DX, 00720h
MOV DS, DX
MOV AX, 00720h
MOV DS, AX
MOV AX, 00001h
MOV DX, 00000h
CALL 0001Bh
MOV [00001h], AX
MOV AX, 04C00h
INT 021h
CMP b.[00000h], 00h
JNLE 023h
RET
DEC b.[00000h]
CALL 0001Bh
INC b.[00000h]
MOV BL, [00000h]
MUL BX
RET
NOP
NOP
...
```

- **OUTPUT**

5! = (120)$_{10}$ = (78)$_{16}$



variables — size: word, elements: 1, show as: hex

```
NUMBER  05h
RESULT  0078h
```

original source code

```
.stack

.data
number db 05h
result dw ?

.code


.startup

mov ax, @data
mov ds, ax
mov ax, 0001h
mov dx, 0000h
call recurr_func
mov result , ax
.exit

recurr_func:
cmp number, 00h
jg calc
ret

calc:
dec number
call recurr_func
inc number
mov bl, number
mul bx
ret
```

Random Access Memory — 0721:0000, ◉ table ○ list

```
0721:0000  BA 20 07 8E DA B8 20 07-8E D8 B8 01 00 BA 00 00   ‖ .Äΰ┐ .Ä╪╕☺.‖..
0721:0010  E8 08 00 A3 01 00 B8 00-4C CD 21 80 3E 00 00 00   Φ..ú☺.╕.L═!Ç>...
0721:0020  7F 01 C3 FE 0E 00 00 E8-F1 FF FE 06 00 00 8A 1E   ⌂☺├■♫..Φ± ■♠..è▲
0721:0030  00 00 F7 E3 C3 90 90 90-90 90 90 90 90 90 90 90   ..≈π├ÉÉÉÉÉÉÉÉÉÉÉ
0721:0040  90 90 90 90 90 90 90 90-90 F4 00 00 00 00 00 00   ÉÉÉÉÉÉÉÉÉ⌠......
0721:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0721:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

emulator: factorial.exe_

```
registers   H   L
AX         4C  00
BX         00  05
CX         01  46
DX         00  00
CS         F400
IP         0204
SS         0710
SP         00FA
BP         0000
SI         0000
DI         0000
DS         0720
ES         0700
```

F400:0204

```
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

F400:0204

```
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

message

```
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

CONCLUSION

In this program we have learnt how to implement recursion in assembly language. Recursion is a fast method to solve problems but takes a lot of memory