



**FAST-NATIONAL UNIVERSITY OF  
COMPUTER AND EMERGING SCIENCES  
PESHAWAR CAMPUS**

**NAME: Junaid Saeed**

**ROLL NO: 22P-9200**

**SECTION: BS(CS)-3D**

**COURSE: Coal**

**ASSIGNMENT # 3**

**SUBMITTED TO: Dr . Usman Abbasi**

## **Question # 1:**

→ In (question 1) i am using subroutines for add, subtract, multiply and divide and then passing numbers as a parameters in the stack and also storing the results for each of the operation in the variables . after all work then popping back them . and rest of the screenshots and code visualization is given below.

```

[org 0x0100]
    jmp start

operand1: dw 5
operand2: dw 10
sum_result: dw 0
subtraction_result: dw 0
multiplication_result: dw 0
division_result: dw 0

divv:
    push bp
    mov bp, sp
    push ax
    push bx

    mov ax, [bp+6]
    mov bx, [bp+4]

    div bx
    mov [division_result], ax
    pop bx
    pop ax
    pop bp
    ret

mull:
    push bp
    mov bp, sp
    push ax
    push bx

    mov ax, [bp+6]
    mov bx, [bp+4]
    imul ax, bx
    mov [multiplication_result], ax
    pop bx
    pop ax
    pop bp
    ret

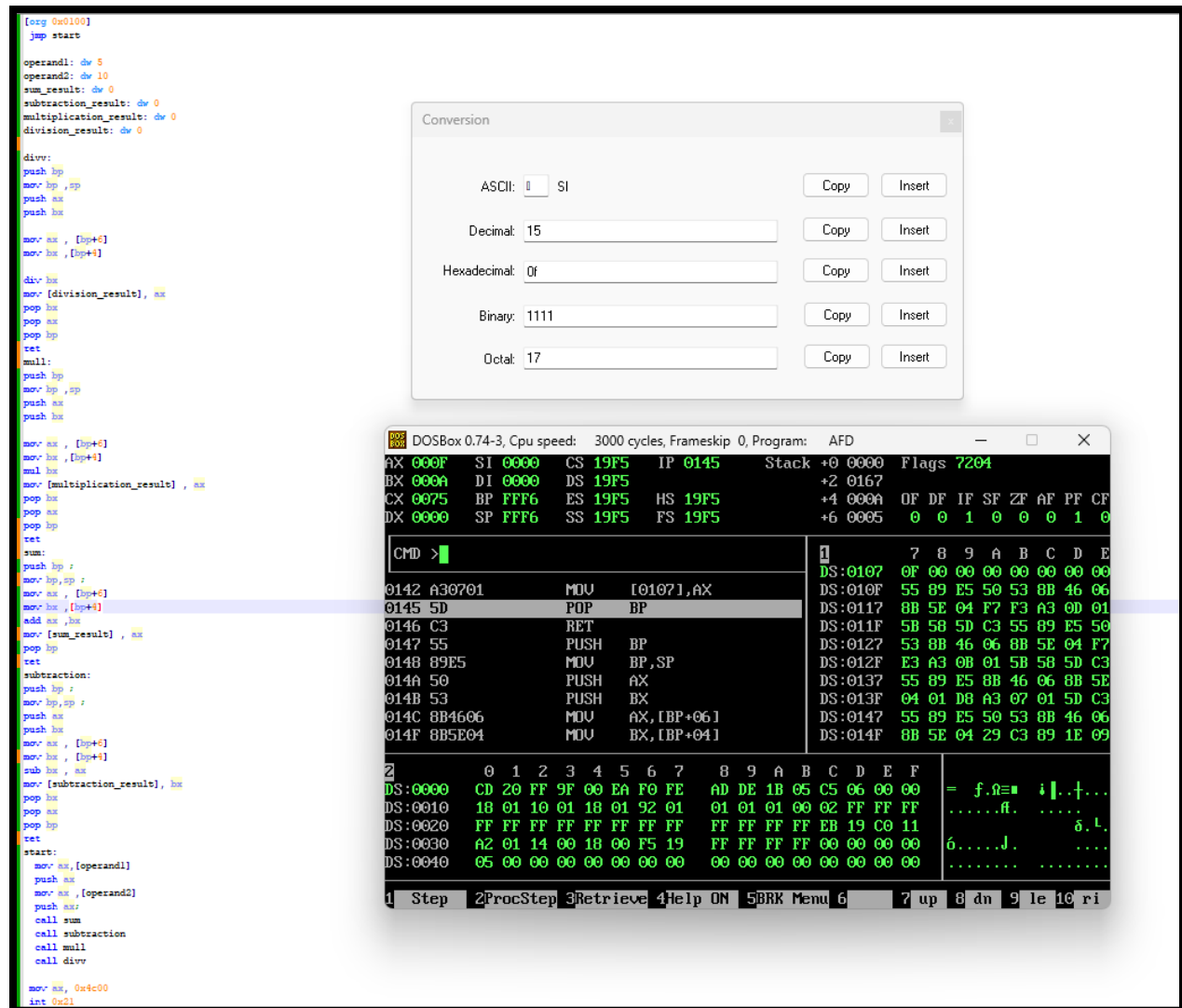
sum:
    push bp ;
    mov bp, sp ;
    mov ax, [bp+6]
    mov bx, [bp+4]
    add ax, bx
    mov [sum_result], ax
    pop bp
    ret

subtraction:
    push bp ;
    mov bp, sp ;
    push ax
    push bx
    mov ax, [bp+6]
    mov bx, [bp+4]
    sub bx, ax
    mov [subtraction_result], bx
    pop bx
    pop ax
    pop bp
    ret

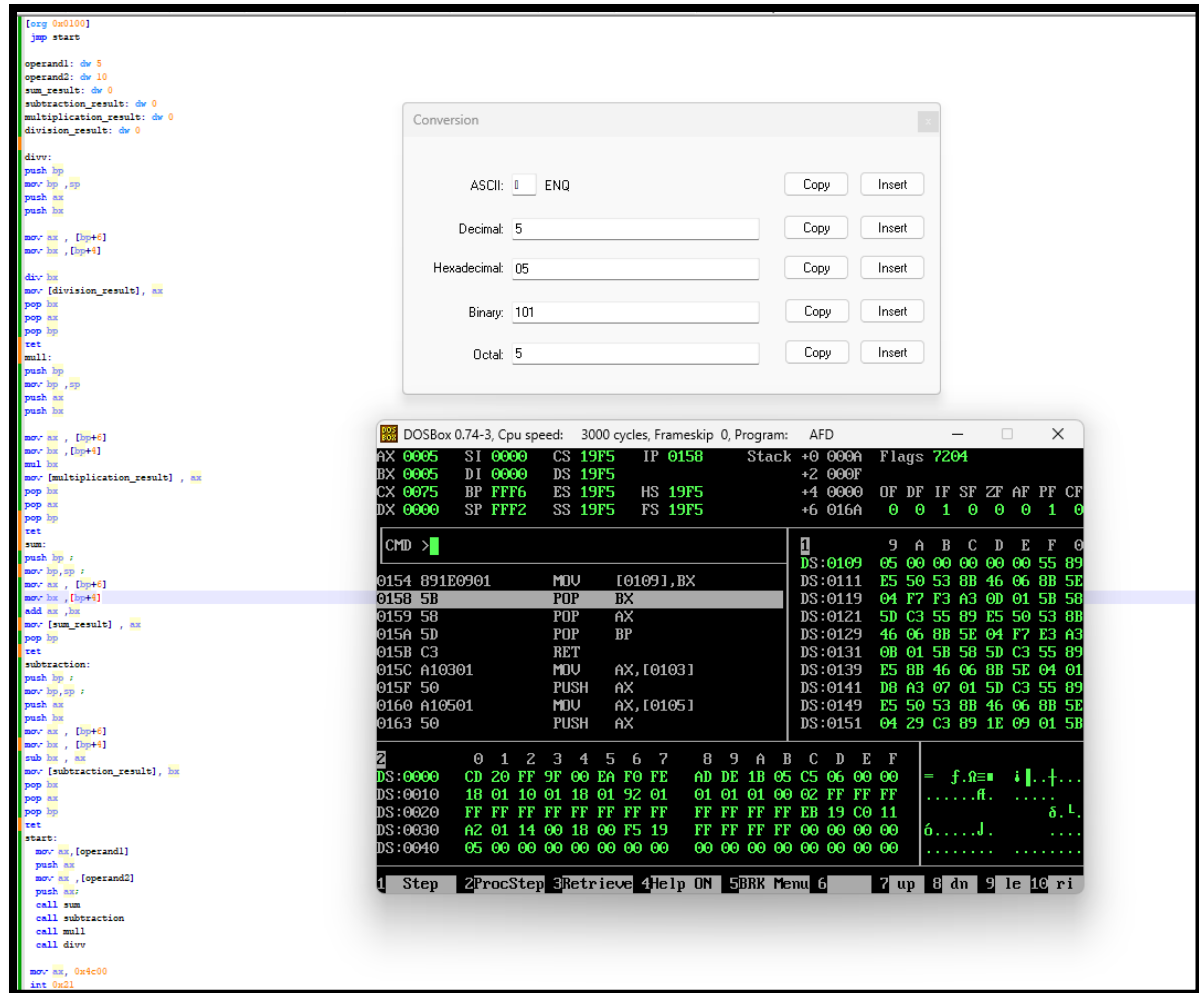
start:
    mov ax, [operand1]
    push ax
    mov ax, [operand2]
    push ax;
    call sum
    call subtraction
    call mull
    call divv

    mov ax, 0x4c00
    int 0x21

```



→ The upper screen shot is of addition.



→ The upper screen shot is of subtraction.

```

[org 0x0100]
jmp start

operand1 dw 5
operand2 dw 10
sum_result dw 0
subtraction_result dw 0
multiplication_result dw 0
division_result dw 0

divr:
push bp
mov bp, sp
push ax
push bx

mov ax, [ip+7]
mov bx, [ip+1]

div bx
mov [division_result], ax
pop bx
pop bp
pop bp
test
mul:
push bp
mov bp, sp
push ax
push bx

mov ax, [ip+7]
mov bx, [ip+1]
imul bx, bx
mov [multiplication_result], ax
pop bx
pop bp
test
sum:
push bp
mov bp, sp
mov ax, [ip+7]
mov bx, [ip+1]
add bx, bx
mov [sum_result], ax
pop bp
test
subtraction:
push bp
mov bp, sp
push ax
push bx

mov ax, [ip+7]
mov bx, [ip+1]
sub bx, ax
mov [subtraction_result], bx
pop bx
pop bp
test
start:
mov ax, [operand1]
push ax
mov ax, [operand2]
push ax
call sum
call subtraction
call mul
call divr

mov ax, 0x400
int 0x21

```

Conversion

ASCII: 2

Copy

Insert

Decimal: 50

Copy

Insert

Hexadecimal: 32

Copy

Insert

Binary: 110010

Copy

Insert

Octal: 62

Copy

Insert

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0032	SI 0000	CS 19F5	IP 0134	Stack +0 0000	Flags 7204
BX 0000	DI 0000	DS 19F5		+2 000F	OF DF IF SF ZF AF PF CF
CX 0079	BP FFF6	ES 19F5	HS 19F5	+4 0000	0F DF IF SF ZF AF PF CF
DX 0000	SP FFF2	SS 19F5	FS 19F5	+6 0171	0 0 1 0 0 0 1 0

CMD >		B	C	D	E	F	0	1	2
0131 A30B01	MOV	[010B], AX	DS:010B	32	00	00	00	55	09
0134 5B	POP	BX	DS:0113	53	8B	46	06	8B	5E
0135 5B	POP	AX	DS:011B	F3	A3	00	01	5B	5D
0136 5D	POP	BP	DS:0123	55	09	E5	50	53	8B
0137 C3	RET		DS:012B	8B	5E	04	0F	AF	C3
0138 55	PUSH	BP	DS:0133	01	5B	5D	5D	C3	55
0139 89E5	MOV	BP, SP	DS:013B	0B	46	06	8B	5E	04
013B 8B4606	MOV	AX, [BP+06]	DS:0143	A3	07	01	5D	C3	55
013E 8B5E04	MOV	BX, [BP+04]	DS:014B	50	53	8B	46	06	8B
			DS:0153	29	C3	89	1E	09	01

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00	= f.0= i.+....
DS:0010	10	01	10	01	10	01	92	01	01	01	01	02	FF	FF	FF	FF	.....f. ....
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	00	11	.....6. t. ....
DS:0030	A2	01	14	00	10	00	F5	19	FF	FF	FF	FF	00	00	00	00	6.....J. ....
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

→ The upper screen shot is of multiplication.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
IX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202
IX 0000 DI 0000 DS 19F5 +2 20CD
IX 0076 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
IX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

100 E95A00 JMP 015D
103 05000A ADD AX,0A00
106 0000 ADD [BX+SI],AL
108 0000 ADD [BX+SI],AL
10A 0000 ADD [BX+SI],AL
10C 0000 ADD [BX+SI],AL
10E 005589 ADD [DI-77],DL
111 E550 IN AX,[50]

1 D E F 0 1 2 3 4
DS:010D 00 00 55 89 E5 50 53 8B
DS:0115 46 06 8B 5E 04 F7 F3 A3
DS:011D 0D 01 5B 58 5D C3 55 89
DS:0125 E5 50 53 8B 46 06 8B 5E
DS:012D 04 0F AF C3 A3 0B 01 5B
DS:0135 58 5D C3 55 89 E5 8B 46
DS:013D 06 8B 5E 04 01 D8 A3 07
DS:0145 01 5D C3 55 89 E5 50 53
DS:014D 8B 46 06 8B 5E 04 29 C3
DS:0155 89 1E 09 01 5B 58 5D C3

S:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω■ ÷|.†...
S:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....ff. ....
S:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 .....δ.L.
S:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
S:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

```

→ The upper screen shot is of division. In this dividing 5 by 10 it gives us 0.5 but in this it will be shown in the output as 0.(not in points).

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202  
 BX 0000 DI 0000 DS 19F5 +2 20CD  
 CX 0000 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF  
 DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

Program terminated OK

0100 E95A00	JMP	015D	DS:010D	00 00 55 89 E5 50 53 8B
0103 05000A	ADD	AX,0A00	DS:0115	46 06 8B 5E 04 F7 F3 A3
0106 000F	ADD	[BX],CL	DS:011D	0D 01 5B 58 5D C3 55 89
0108 0005	ADD	[DI],AL	DS:0125	E5 50 53 8B 46 06 8B 5E
010A 0032	ADD	[BP+SI],DH	DS:012D	04 0F AF C3 A3 0B 01 5B
010C 0000	ADD	[BX+SI],AL	DS:0135	58 5D C3 55 89 E5 8B 46
010E 005589	ADD	[DI-77],DL	DS:013D	06 8B 5E 04 01 D8 A3 07
0111 E550	IN	AX,[50]	DS:0145	01 5D C3 55 89 E5 50 53
			DS:014D	8B 46 06 8B 5E 04 29 C3
			DS:0155	89 1E 09 01 5B 58 5D C3

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
DS:0000	CD	20	FF	9F	00	EA	FF	FF	AD	DE	1B	05	C5	06	00	00	= f.0 i   . + ...
DS:0010	18	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FF	.....ff. ....
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	E4	11	.....J. ....
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00	6.....J. ....
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

→Program terminated sucessfully.

## Question # 2: (now using recursion)



```

[org 0x0100]

jmp start

data:    dw 5
count:   dw 11
result:  dw 0

recursion:
    push bp
    mov bp, sp
    mov ax, [bp + 4]      ; Load the argument from the stack
    add [result], ax
    dec word[count]
    cmp word[count], 0
    jle end_recursion    ; Jump if less than or equal to zero

    ; Recursive call
    push word[result]    ; Save the current result on the stack
    push word[count]     ; Save the current count on the stack
    call recursion       ; Recursive call
    add sp, 4            ; Clean up the stack

end_recursion:
    pop bp
    ret

start:   mov ax, [data]
         push ax
         mov cx, [count]
         push cx
         call recursion

         mov ax, 0x4c00
         int 0x21

```

→steps:

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

IX 0007	SI 0000	CS 19F5	IP 0117	Stack +0 FFE6	Flags 7204
IX 0000	DI 0000	DS 19F5		+2 012A	
IX 000A	BP FFDE	ES 19F5	HS 19F5	+4 0007	OF DF IF SF ZF AF PF CF
IX 0000	SP FFDE	SS 19F5	FS 19F5	+6 001B	0 0 1 0 0 0 1 0

CMD >				1	7	8	9	A	B	C	D	E
0006				DS:0107	22	00	55	89	E5	8B	46	04
M113 FF0E0501 DEC W/[0105]				DS:010F	01	06	07	01	FF	0E	05	01
M117 813E05010000 CMP [0105],0000				DS:0117	81	3E	05	01	00	00	7E	0F
M11D 7E0F JNG 012E				DS:011F	FF	36	07	01	FF	36	05	01
M11F FF360701 PUSH [0107]				DS:0127	E8	DF	FF	81	C4	04	00	5D
M123 FF360501 PUSH [0105]				DS:012F	C3	A1	03	01	50	8B	0E	05
M127 E8DFFF CALL 0109				DS:0137	01	51	E8	CD	FF	B8	00	4C
M12A 81C40400 ADD SP,0004				DS:013F	CD	21	8E	5E	FC	83	7D	0E
M12E 5D POP BP				DS:0147	00	74	09	8B	46	F2	48	3B
M12F C3 RET				DS:014F	46	F6	7E	08	B8	01	00	EB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
S:0010	18	01	10	01	18	01	92	01	01	01	00	FF	00	01	00	
S:0020	01	00	01	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	C0	11	
S:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00
S:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0006 SI 0000 CS 19F5 IP 0123 Stack +0 002B Flags 7204

BX 0000 DI 0000 DS 19F5

CX 000A BP FFD6 ES 19F5 HS 19F5

DX 0000 SP FFD4 SS 19F5 FS 19F5

+2 FFDE

+4 012A 0F DF IF SF ZF AF PF CF

+6 0006 0 0 1 0 0 0 1 0

CMD >

0005

011F FF360701 PUSH [0107]

0123 FF360501 PUSH [0105]

0127 E8DFFF CALL 0109

012A 81C40400 ADD SP,0004

012E 5D POP BP

012F C3 RET

0130 A10301 MOV AX,[0103]

0133 50 PUSH AX

0134 8B0E0501 MOV CX,[0105]

1

7 8 9 A B C D E

DS:0107 28 00 55 89 E5 8B 46 04

DS:010F 01 06 07 01 FF 0E 05 01

DS:0117 81 3E 05 01 00 00 7E 0F

DS:011F FF 36 07 01 FF 36 05 01

DS:0127 E8 DF FF 81 C4 04 00 5D

DS:012F C3 A1 03 01 50 8B 0E 05

DS:0137 01 51 E8 CD FF B8 00 4C

DS:013F CD 21 8E 5E FC 83 7D 0E

DS:0147 00 74 09 8B 46 F2 48 3B

DS:014F 46 F6 7E 08 B8 01 00 EB

2

0 1 2 3 4 5 6 7 8 9 A B C D E F

DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω■ i|..+...

DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 FF 00 01 00 .....ff. ....

DS:0020 01 00 01 FF FF FF FF FF FF FF FF FF EB 19 C0 11 ... δ.L.

DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....

DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....

1 Step

2ProcStep

3Retrieve

4Help ON

5BRK Menu

6

7 up

8 dn

9 le

10 ri

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0003 SI 0000 CS 19F5 IP 011F Stack +0 FFC6 Flags 7200

BX 0000 DI 0000 DS 19F5

CX 000A BP FFBE ES 19F5 HS 19F5

DX 0000 SP FFBE SS 19F5 FS 19F5

+2 012A

+4 0003 0F DF IF SF ZF AF PF CF

+6 0031 0 0 1 0 0 0 0 0

CMD >

0034

011D 7E0F JNG 012E

011F FF360701 PUSH [0107]

0123 FF360501 PUSH [0105]

0127 E8DFFF CALL 0109

012A 81C40400 ADD SP,0004

012E 5D POP BP

012F C3 RET

0130 A10301 MOV AX,[0103]

0133 50 PUSH AX

1

7 8 9 A B C D E

DS:0107 34 00 55 89 E5 8B 46 04

DS:010F 01 06 07 01 FF 0E 05 01

DS:0117 81 3E 05 01 00 00 7E 0F

DS:011F FF 36 07 01 FF 36 05 01

DS:0127 E8 DF FF 81 C4 04 00 5D

DS:012F C3 A1 03 01 50 8B 0E 05

DS:0137 01 51 E8 CD FF B8 00 4C

DS:013F CD 21 8E 5E FC 83 7D 0E

DS:0147 00 74 09 8B 46 F2 48 3B

DS:014F 46 F6 7E 08 B8 01 00 EB

2

0 1 2 3 4 5 6 7 8 9 A B C D E F

DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω■ i|..+...

DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 FF 00 01 00 .....ff. ....

DS:0020 01 00 01 FF FF FF FF FF FF FF FF FF EB 19 C0 11 ... δ.L.

DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....

DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....

1 Step

2ProcStep

3Retrieve

4Help ON

5BRK Menu


6

7 up

8 dn

9 le

10 ri


 DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202  
 BX 0000 DI 0000 DS 19F5 +2 20CD  
 CX 0000 BP 0000 ES 19F5 HS 19F5 +4 9FFF 0F DF IF SF ZF AF PF CF  
 DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

> Program terminated OK

0100	E92D00	JMP	0130
0103	050000	ADD	AX,0000
0106	0037	ADD	[BX],DH
0108	005589	ADD	[DI-77],DL
010B	E58B	IN	AX,[8B]
010D	46	INC	SI
010E	0401	ADD	AL,01
0110	06	PUSH	ES

1

7	8	9	A	B	C	D	E	
DS:0107	37	00	55	89	E5	8B	46	04
DS:010F	01	06	07	01	FF	0E	05	01
DS:0117	81	3E	05	01	00	00	7E	0F
DS:011F	FF	36	07	01	FF	36	05	01
DS:0127	E8	DF	FF	81	C4	04	00	5D
DS:012F	C3	A1	03	01	50	8B	0E	05
DS:0137	01	51	E8	CD	FF	B8	00	4C
DS:013F	CD	21	8E	5E	FC	83	7D	0E
DS:0147	00	74	09	8B	46	F2	48	3B
DS:014F	46	F6	7E	08	B8	01	00	EB

2

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
DS:0000	CD	20	FF	9F	00	EA	FF	FF	AD	DE	1B	05	C5	06	00	00
DS:0010	18	01	10	01	18	01	92	01	01	01	00	FF	00	01	00	00
DS:0020	01	00	01	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	E4	11	00
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

= f.ñ ì |..†...  
 .....f. ....  
 ... δ.Σ.  
 ó.....J. ....  
 .....

1

Step

2ProcStep

3Retrieve

4Help ON

5BRK Menu

6

7 up

8 dn

9 le

10 ri

## →FINAL OUTPUT:

The screenshot displays a DOSBox environment with three main components:

- Assembly Code Editor:** Shows the assembly code for a recursive program. The code initializes data, sets a count of 11, and calls a recursive subroutine to add 5 repeatedly. The final instruction is `int 0x21` to terminate the program.
- Conversion Utility:** A small window showing the decimal value 55 converted to other bases: ASCII (7), Hexadecimal (0037), Binary (110111), and Octal (67).
- DOSBox Memory Dump:** Shows the state of registers and memory. The `AX` register contains `0000`, and the `SI` register contains `0000`. The memory dump shows the program's execution flow, including the recursive call and return sequence.

→In this I am using recursion and also using subroutine to add 5 (11 times) in this I am also passing parameters and saving values.and at the end popping back .and the result as shown in the upper screenshot is 55.

--THANKS--