Assignment No: 3

Fahid Imran

Roll No: 23i-0061

COAL

BSAI B

April 5, 2025

Instructor

Ms. Farheen Tabassum

Fast NUCES Islamabad

Campus

ID = 23i-0061

X = 0 0000 h

Y = 0 0061 h

P = 0 0023 h

Q = 0 2361 h

|----------o(Question No 1)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 000h
0 0002 h -->	Mov Bx, 061h
0 0003 h -->	Push Ax
0 0004 h -->	Call MyProc
0 0005 h -->	Exit
0 0006 h -->	Main EndP
0 0007 h -->	MyProc Proc
0 0008 h -->	Push 3
0 0009 h -->	Push 10
0 000A h -->	Call Exmp
0 000B h -->	Add sp, 4
0 000C h -->	Ret 2
0 000D h -->	Exmp Proc
0 000E h -->	Mov Bp, Sp
0 000F h -->	Push Bp
0 0010 h -->	Mov Bp, Sp
0 0011 h -->	Mov Ax, [Bp + 10]
0 0012 h -->	Mov Bx, [Bp + 6]
0 0013 h -->	Add Ax, Bx
0 0014 h -->	Push Ax

0 0015 h --> Pop Bp

0 0016 h --> Ret 2

Solution:

Stack		
Address	**Data**	
0 0026 h		← SP
0 0024 h		
0 0022 h		
0 0020 h		
0 001E h		
0 001C h		
0 001A h		
0 0018 h		
0 0016 h		
0 0014 h		

Main proc

Mov Ax, 0 0000 h

Mov Bx, 0 0061 h

Push Ax

Call MyProc

Stack		
Address	**Data**	
0 0026 h		
0 0024 h	0 0000 h	
0 0022 h	RA_Main = 0 0005 h	← SP
0 0020 h		

0 001E h	
0 001C h	
0 001A h	
0 0018 h	
0 0016 h	
0 0014 h	

MyProc Proc

Push 3

Push 10

Call Exmp

Stack	
Address	**Data**
0 0026 h	
0 0024 h	0 0000 h
0 0022 h	RA_Main = 0 0005 h
0 0020 h	0 0003 h
0 001E h	0 000a h
0 001C h	RA_MyProc = 0 000B h
0 001A h	
0 0018 h	
0 0016 h	
0 0014 h	

← SP

Exmp Proc

Mov Bp, Sp

Stack	
Address	**Data**

0 0026 h	
0 0024 h	0 0000 h
0 0022 h	RA_Main = 0 0005 h
0 0020 h	0 0003 h
0 001E h	0 000a h
0 001C h	RA_MyProc = 0 000B h
0 001A h	
0 0018 h	
0 0016 h	
0 0014 h	

← SP, BP

Push Bp

Mov Bp, Sp

Stack	
Address	**Data**
0 0026 h	
0 0024 h	0 0000 h
0 0022 h	RA_Main = 0 0005 h
0 0020 h	0 0003 h
0 001E h	0 000a h
0 001C h	RA_MyProc = 0 000B h
0 001A h	0 001C h
0 0018 h	
0 0016 h	
0 0014 h	

← SP, BP

Mov Ax, [Bp + 10]

Mov Bx, [Bp + 6]

Add Ax, Bx

Registers	
Name	**Value**
AX	0 0000 h
BX	0 0003 h
AX	0 0003 h

Push Ax

Stack		
Address	**Data**	
0 0026 h		
0 0024 h	0 0000 h	
0 0022 h	RA_Main = 0 0005 h	
0 0020 h	0 0003 h	
0 001E h	0 000a h	
0 001C h	RA_MyProc = 0 000B h	
0 001A h	0 001C h	← BP
0 0018 h	0 0003 h	← SP
0 0016 h		
0 0014 h		

Pop Bp

Stack	
Address	**Data**
0 0026 h	
0 0024 h	0 0000 h
0 0022 h	RA_Main = 0 0005 h

0 0020 h	0 0003 h	
0 001E h	0 000a h	
0 001C h	RA_MyProc = 0 000B h	
0 001A h	0 001C h	← SP
0 0018 h	poped	
0 0016 h		
0 0014 h		

Ret 2

Error: This will cause an error because ret statement will mov top value to IP and in this case it will mov 0 001C h in IP. Which is not required return location. Therefore, it may cause an error.

Solution: Use following code:

Add SP, 2

Ret

Stack		
Address	**Data**	
0 0026 h		
0 0024 h	0 0000 h	
0 0022 h	RA_Main = 0 0005 h	
0 0020 h	0 0003 h	
0 001E h	0 000a h	← SP
0 001C h	poped	
0 001A h	poped	
0 0018 h	poped	
0 0016 h		
0 0014 h		

Add sp, 4

Stack	
Address	**Data**
0 0026 h	
0 0024 h	0 0000 h
0 0022 h	RA_Main = 0 0005 h
0 0020 h	poped
0 001E h	poped
0 001C h	poped
0 001A h	poped
0 0018 h	poped
0 0016 h	
0 0014 h	

← SP

Ret 2

Stack	
Address	**Data**
0 0026 h	
0 0024 h	poped
0 0022 h	poped
0 0020 h	poped
0 001E h	poped
0 001C h	poped
0 001A h	poped
0 0018 h	poped
0 0016 h	
0 0014 h	

← SP

Exit

Main EndP

|----------o(Question No 2)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 12h
0 0002 h -->	Mov Bx, 28h
0 0003 h -->	Push Ax
0 0004 h -->	Push Bx
0 0005 h -->	Mov Cx, 5
0 0006 h -->	Push Cx
0 0007 h -->	Call MyProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	MyProc Proc
0 000B h -->	L1:
0 000C h -->	Push Bp
0 000D h -->	Mov Bp, Sp
0 000E h -->	Cmp Cx, 0
0 000F h -->	Jz L2
0 0010 h -->	Mov Ax, [Bp + 6]
0 0011 h -->	Mov Bx, [Bp + 4]
0 0012 h -->	Add Ax, Bx
0 0013 h -->	Push Ax
0 0014 h -->	Push Bx
0 0015 h -->	Dec Cx
0 0016 h -->	Push Cx

0 0017 h --> Jmp L1

0 0018 h --> L2:

0 0019 h --> ADD SP, 40

0 001A h --> Ret 6

0 001B h --> MyProc EndP

Solution:

Stack	
Address	**Data**
0 003C h	
0 003A h	
0 0038 h	
0 0036 h	
0 0034 h	
0 0032 h	
0 0030 h	
0 002E h	
0 002C h	
0 002A h	

← SP

Main proc

Mov Ax, 12h

Mov Bx, 28h

Push Ax

Push Bx

Mov Cx, 5

Push Cx

Call MyProc

Stack

Address	Data	
0 003C h	0 0012 h	
0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	← SP
0 0034 h		
0 0032 h		
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

```
MyProc Proc
L1:
Push Bp
Mov Bp, Sp
Cmp Cx, 0
Jz L2
Mov Ax, [Bp + 6]
Mov Bx, [Bp + 4]
Add Ax, Bx
Push Ax
Push Bx
Dec Cx
Push Cx
Jmp L1
```

; When cx = 5

Registers	
Name	**Value**
CX	0 0005 h

AX	0 0028 h
BX	0 0005 h
AX	0 002d h
CX	0 0004 h

Stack		
Address	**Data**	
0 003C h	0 0012 h	
0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	
0 0034 h	Old Value of BP	← BP
0 0032 h	0 002d h	
0 0030 h	0 0005 h	
0 002E h	0 0004 h	← SP
0 002C h		
0 002A h		

; When cx = 4

Registers	
Name	**Value**
CX	0 0004 h
AX	0 002d h
BX	0 0005 h
AX	0 0032 h
CX	0 0003 h

Stack

Address	Data	
0 003C h	0 0012 h	
0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	
0 0034 h	Old Value of BP	
0 0032 h	0 002d h	
0 0030 h	0 0005 h	
0 002E h	0 0004 h	
0 002C h	0 0034 h	← BP
0 002A h	0 0032 h	
0 0028 h	0 0005 h	
0 0026 h	0 0003 h	← SP
0 0024 h		
0 0022 h		
0 0020 h		
0 001E h		
0 001C h		

; When cx = 3

Registers	
Name	**Value**
CX	0 0003 h
AX	0 0032 h
BX	0 0005 h
AX	0 0037 h
CX	0 0002 h

Stack		
Address	**Data**	
0 003C h	0 0012 h	
0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	
0 0034 h	Old Value of BP	
0 0032 h	0 002d h	
0 0030 h	0 0005 h	
0 002E h	0 0004 h	
0 002C h	0 0034 h	
0 002A h	0 0032 h	
0 0028 h	0 0005 h	
0 0026 h	0 0003 h	
0 0024 h	0 002C h	← BP
0 0022 h	0 0037 h	
0 0020 h	0 0005 h	
0 001E h	0 0002 h	← SP
0 001C h		

; When cx = 2

Registers	
Name	**Value**
CX	0 0002 h
AX	0 0037 h
BX	0 0005 h
AX	0 003C h

CX	0 0001 h

Stack		
Address	**Data**	
0 003C h	0 0012 h	
0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	
0 0034 h	Old Value of BP	
0 0032 h	0 002d h	
0 0030 h	0 0005 h	
0 002E h	0 0004 h	
0 002C h	0 0034 h	
0 002A h	0 0032 h	
0 0028 h	0 0005 h	
0 0026 h	0 0003 h	
0 0024 h	0 002C h	
0 0022 h	0 0037 h	
0 0020 h	0 0005 h	
0 001E h	0 0002 h	
0 001C h	0 0024 h	← BP
0 001A h	0 003C h	
0 0018 h	0 0005 h	
0 0016 h	0 0001 h	← SP
0 0014 h		
0 0012 h		
0 0010 h		

0 000E h	
0 000C h	
0 000A h	
0 0008 h	

; When cx = 1

Registers	
Name	**Value**
CX	0 0001 h
AX	0 003C h
BX	0 0005 h
AX	0 0041 h
CX	0 0000 h

Stack	
Address	**Data**
0 003C h	0 0012 h
0 003A h	0 0028 h
0 0038 h	0 0005 h
0 0036 h	RA_Main = 0 0008 h
0 0034 h	Old Value of BP
0 0032 h	0 002d h
0 0030 h	0 0005 h
0 002E h	0 0004 h
0 002C h	0 0034 h
0 002A h	0 0032 h
0 0028 h	0 0005 h
0 0026 h	0 0003 h

0 0024 h	0 002C h	
0 0022 h	0 0037 h	
0 0020 h	0 0005 h	
0 001E h	0 0002 h	
0 001C h	0 0024 h	
0 001A h	0 003C h	
0 0018 h	0 0005 h	
0 0016 h	0 0001 h	
0 0014 h	0 001C h	← BP
0 0012 h	0 0041 h	
0 0010 h	0 0005 h	
0 000E h	0 0000 h	← SP
0 000C h		
0 000A h		
0 0008 h		

; When cx = 0

Stack	
Address	**Data**
0 003C h	0 0012 h
0 003A h	0 0028 h
0 0038 h	0 0005 h
0 0036 h	RA_Main = 0 0008 h
0 0034 h	Old Value of BP
0 0032 h	0 002d h
0 0030 h	0 0005 h
0 002E h	0 0004 h

0 002C h	0 0034 h
0 002A h	0 0032 h
0 0028 h	0 0005 h
0 0026 h	0 0003 h
0 0024 h	0 002C h
0 0022 h	0 0037 h
0 0020 h	0 0005 h
0 001E h	0 0002 h
0 001C h	0 0024 h
0 001A h	0 003C h
0 0018 h	0 0005 h
0 0016 h	0 0001 h
0 0014 h	0 001C h
0 0012 h	0 0041 h
0 0010 h	0 0005 h
0 000E h	0 0000 h
0 000C h	0 0014 h ← SP, BP
0 000A h	
0 0008 h	

L2:

ADD SP, 40

Stack	
Address	**Data**
0 003C h	0 0012 h

0 003A h	0 0028 h	
0 0038 h	0 0005 h	
0 0036 h	RA_Main = 0 0008 h	← SP
0 0034 h	poped	
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h	poped	
0 0024 h	poped	
0 0022 h	poped	
0 0020 h	poped	
0 001E h	poped	
0 001C h	poped	
0 001A h	poped	
0 0018 h	poped	
0 0016 h	poped	
0 0014 h	poped	
0 0012 h	poped	
0 0010 h	poped	
0 000E h	poped	
0 000C h	poped	← BP
0 000A h		
0 0008 h		

Ret 6

Stack		
Address	Data	
0 0042 h		← SP
0 0040 h	poped	
0 003E h	poped	
0 003C h	poped	
0 003A h	poped	
0 0038 h	poped	
0 0036 h	poped	
0 0034 h	poped	
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h	poped	
0 0024 h	poped	
0 0022 h	poped	
0 0020 h	poped	
0 001E h	poped	
0 001C h	poped	
0 001A h	poped	
0 0018 h	poped	
0 0016 h	poped	
0 0014 h	poped	

0 0012 h	poped	
0 0010 h	poped	
0 000E h	poped	
0 000C h	poped	← BP
0 000A h		
0 0008 h		

MyProc EndP

Exit

Main EndP

|----------o(Question No 3)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 1Fh
0 0002 h -->	Mov Bx, 2Ch
0 0003 h -->	Push Ax
0 0004 h -->	Push Bx
0 0005 h -->	Mov Cx, 4
0 0006 h -->	Push Cx
0 0007 h -->	Call MyProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	MyProc Proc
0 000B h -->	L1:
0 000C h -->	Push Bp
0 000D h -->	Mov Bp, Sp
0 000E h -->	Cmp Cx, 0
0 000F h -->	Jz L2
0 0010 h -->	Mov Ax, [Bp + 6]

0 0011 h --> Mov Bx, [Bp + 4]

0 0012 h --> Rol Ax, 1

0 0013 h --> Ror Bx, 1

0 0014 h --> And Ax, Bx

0 0015 h --> Push Ax

0 0016 h --> Push Bx

0 0017 h --> Dec Cx

0 0018 h --> Push Cx

0 0019 h --> Jmp L1

0 001A h --> L2:

0 001B h --> ADD SP, 40

0 001C h --> Ret 6

0 001D h --> MyProc EndP

Solution:

Stack		
Address	**Data**	
0 003C h		← SP
0 003A h		
0 0038 h		
0 0036 h		
0 0034 h		
0 0032 h		
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

Main proc

Mov Ax, 01Fh

Mov Bx, 02Ch

Push Ax

Push Bx

Mov Cx, 4

Push Cx

Call MyProc

Registers	
Name	**Value**
AX	0 001F h
BX	0 002C h
CX	0 0004 h

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 001F h	
0 0038 h	0 002C h	
0 0036 h	0 0004 h	
0 0034 h	RA_Main = 0 0008 h	← SP
0 0032 h		
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

MyProc Proc

L1:

Push Bp

Mov Bp, Sp

Cmp Cx, 0

```
Jz L2
Mov Ax, [Bp + 6]
Mov Bx, [Bp + 4]
Rol Ax, 1
Ror Bx, 1
And Ax, Bx
Push Ax
Push Bx
Dec Cx
Push Cx
Jmp L1
```

; for CX = 4

Registers			
Name	**Value**		
CX	0 0004 h		
AX	0 002C h	0000 0000 0010 1100	
BX	0 0004 h	0000 0000 0000 0100	
AX	0 0058 h	0000 0000 0101 1000	ROL 1
BX	0 0002 h	0000 0000 0000 0010	ROR 1
AX	0 0000 h	0000 0000 0000 0000	AND
CX	0 0003 h		

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 001F h	
0 0038 h	0 002C h	
0 0036 h	0 0004 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP

0 0030 h	0 0000 h	
0 002E h	0 0002 h	
0 002C h	0 0003 h	← SP
0 002A h		

; for CX = 3

Registers			
Name	**Value**		
CX	0 0003 h		
AX	0 0000 h	0000 0000 0000 0000	
BX	0 0002 h	0000 0000 0000 0010	
AX	0 0000 h	0000 0000 0000 0000	ROL 1
BX	0 0001 h	0000 0000 0000 0001	ROR 1
AX	0 0000 h	0000 0000 0000 0000	AND
CX	0 0002 h		

Stack	
Address	**Data**
0 003C h	
0 003A h	0 001F h
0 0038 h	0 002C h
0 0036 h	0 0004 h
0 0034 h	RA_Main = 0 0008 h
0 0032 h	Old value of BP
0 0030 h	0 0000 h
0 002E h	0 0002 h
0 002C h	0 0003 h

0 002A h	0 0032 h	← BP
0 0028 h	0 0000 h	
0 0026 h	0 0001 h	
0 0024 h	0 0002 h	← SP
0 0022 h		
0 0020 h		
0 001E h		
0 001C h		
0 001A h		
0 0018 h		
0 0016 h		
0 0014 h		
0 0012 h		
0 0010 h		

; for CX = 2

Registers			
Name	**Value**		
CX	0 0002 h		
AX	0 0000 h	0000 0000 0000 0000	
BX	0 0001 h	0000 0000 0000 0001	
AX	0 0000 h	0000 0000 0000 0000	ROL 1
BX	0 8000 h	1000 0000 0000 0000	ROR 1
AX	0 0000 h	0000 0000 0000 0000	AND
CX	0 0001 h		

Stack

Address	Data	
0 003C h		
0 003A h	0 001F h	
0 0038 h	0 002C h	
0 0036 h	0 0004 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	
0 0030 h	0 0000 h	
0 002E h	0 0002 h	
0 002C h	0 0003 h	
0 002A h	0 0032 h	
0 0028 h	0 0000 h	
0 0026 h	0 0001 h	
0 0024 h	0 0002 h	
0 0022 h	0 002A h	← BP
0 0020 h	0 0000 h	
0 001E h	0 8000 h	
0 001C h	0 0001 h	← SP
0 001A h		
0 0018 h		
0 0016 h		
0 0014 h		
0 0012 h		
0 0010 h		

; for CX = 1

Registers		

Name	Value		
CX	0 0001 h		
AX	0 0000 h	0000 0000 0000 0000	
BX	0 8000 h	1000 0000 0000 0000	
AX	0 0000 h	0000 0000 0000 0000	ROL 1
BX	0 4000 h	0100 0000 0000 0000	ROR 1
AX	0 0000 h	0000 0000 0000 0000	AND
CX	0 0000 h		

Stack	
Address	**Data**
0 003C h	
0 003A h	0 001F h
0 0038 h	0 002C h
0 0036 h	0 0004 h
0 0034 h	RA_Main = 0 0008 h
0 0032 h	Old value of BP
0 0030 h	0 0000 h
0 002E h	0 0002 h
0 002C h	0 0003 h
0 002A h	0 0032 h
0 0028 h	0 0000 h
0 0026 h	0 0001 h
0 0024 h	0 0002 h
0 0022 h	0 002A h
0 0020 h	0 0000 h
0 001E h	0 8000 h
0 001C h	0 0001 h

0 001A h	0 0022 h	← BP
0 0018 h	0 0000 h	
0 0016 h	0 4000 h	
0 0014 h	0 0000 h	← SP
0 0012 h		
0 0010 h		

; for CX = 0

Stack	
Address	**Data**
0 003C h	
0 003A h	0 001F h
0 0038 h	0 002C h
0 0036 h	0 0004 h
0 0034 h	RA_Main = 0 0008 h
0 0032 h	Old value of BP
0 0030 h	0 0000 h
0 002E h	0 0002 h
0 002C h	0 0003 h
0 002A h	0 0032 h
0 0028 h	0 0000 h
0 0026 h	0 0001 h
0 0024 h	0 0002 h
0 0022 h	0 002A h
0 0020 h	0 0000 h
0 001E h	0 8000 h
0 001C h	0 0001 h

0 001A h	0 0022 h	
0 0018 h	0 0000 h	
0 0016 h	0 4000 h	
0 0014 h	0 0000 h	
0 0012 h	0 001A h	← SP, BP
0 0010 h		

L2:

ADD SP, 40

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 001F h	← SP
0 0038 h	poped	
0 0036 h	poped	
0 0034 h	poped	
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h	poped	
0 0024 h	poped	
0 0022 h	poped	
0 0020 h	poped	

0 001E h	poped	
0 001C h	poped	
0 001A h	poped	
0 0018 h	poped	
0 0016 h	poped	
0 0014 h	poped	
0 0012 h	poped	← BP
0 0010 h		

Ret 6

Note: ret 6 = SP + 2 + 6

Stack		
Address	**Data**	
0 0042 h		← SP
0 0040 h	poped	
0 003E h	poped	
0 003C h	poped	
0 003A h	poped	
0 0038 h	poped	
0 0036 h	poped	
0 0034 h	poped	
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	

0 0026 h	poped	
0 0024 h	poped	
0 0022 h	poped	
0 0020 h	poped	
0 001E h	poped	
0 001C h	poped	
0 001A h	poped	
0 0018 h	poped	
0 0016 h	poped	
0 0014 h	poped	
0 0012 h	poped	← BP
0 0010 h		

Error: Ret 6 will pop 0 001F h from Stack and mov it to IP register. And then add 6 in SP. But this is not required returning point and may cause error.

MyProc EndP

Exit

Main EndP

|----------o(Question No 4)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 000h
0 0002 h -->	Mov Bx, 061h
0 0003 h -->	Mov Dx, 5

```
0  0004 h  -->          Push Ax
0  0005 h  -->          Push Bx
0  0006 h  -->          Push Dx
0  0007 h  -->          Call NestedProc
0  0008 h  -->          Exit
0  0009 h  -->          Main EndP
0  000A h  -->          NestedProc Proc
0  000B h  -->          Push Bp
0  000C h  -->          Mov Bp, Sp
0  000D h  -->          Mov Ax, [Bp + 6]
0  000E h  -->          Mov Bx, [Bp + 4]
0  000F h  -->          Or Ax, Bx
0  0010 h  -->          Shl Ax, 1
0  0011 h  -->          Push Ax
0  0012 h  -->          Call DeepProc
0  0013 h  -->          Pop Ax
0  0014 h  -->          Pop Bp
0  0015 h  -->          Ret 6
0  0016 h  -->          NestedProc EndP
0  0017 h  -->          DeepProc Proc
0  0018 h  -->          Mov Cx, 3
0  0019 h  -->          Loop1:
0  001A h  -->          Shr Ax, 1
0  001B h  -->          Xor Ax, Bx
0  001C h  -->          Push Ax
0  001D h  -->          Loop Loop1
0  001E h  -->          Ret
0  001F h  -->          DeepProc EndP
```

Solution:

Stack

Address	Data
0 003C h	
0 003A h	
0 0038 h	
0 0036 h	
0 0034 h	
0 0032 h	
0 0030 h	
0 002E h	
0 002C h	
0 002A h	

← SP

Main proc

Mov Ax, 000h

Mov Bx, 061h

Mov Dx, 5

Push Ax

Push Bx

Push Dx

Call NestedProc

Registers	
Name	**Value**
AX	0 0000 h
BX	0 0061 h
DX	0 0005 h

Stack	
Address	**Data**

0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	← SP
0 0032 h		
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

```
NestedProc Proc
Push Bp
Mov Bp, Sp
Mov Ax, [Bp + 6]
Mov Bx, [Bp + 4]
Or Ax, Bx
Shl Ax, 1
Push Ax
Call DeepProc
```

Registers			
Name	**Value**		
AX	0 0061 h	0000 0000 0110 0001	
BX	0 0005 h	0000 0000 0000 0101	
AX	0 0065 h	0000 0000 0110 0101	OR
AX	0 00CA h	0000 0000 1100 1010	SHL

Stack

Address	Data	
0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 00CA h	
0 002E h	RA_NestedProc = 0 0013 h	← SP
0 002C h		
0 002A h		

```
DeepProc Proc
Mov Cx, 3
Loop1:
Shr Ax, 1
Xor Ax, Bx
Push Ax
Loop Loop1
; cx = 3
```

Registers			
Name	Value		
CX	0 0003 h		
BX	0 0005 h	0000 0000 0000 0101	
AX	0 00CA h	0000 0000 1100 1010	
AX	0 0065 h	0000 0000 0110 0101	SHR
AX	0 0060 h	0000 0000 0110 0000	XOR

Stack

Address	Data	
0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 00CA h	
0 002E h	RA_NestedProc = 0 0013 h	
0 002C h	0 0060 h	← SP
0 002A h		

; cx = 2

Registers			
Name	Value		
CX	0 0002 h		
BX	0 0005 h	0000 0000 0000 0101	
AX	0 0060 h	0000 0000 0110 0000	
AX	0 0030 h	0000 0000 0011 0000	SHR
AX	0 0035 h	0000 0000 0011 0101	XOR

Stack	
Address	Data
0 003C h	
0 003A h	0 0000 h
0 0038 h	0 0061 h
0 0036 h	0 0005 h

0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 00CA h	
0 002E h	RA_NestedProc = 0 0013 h	
0 002C h	0 0060 h	
0 002A h	0 0035 h	← SP

; cx = 1

Registers			
Name	**Value**		
CX	0 0001 h		
BX	0 0005 h	0000 0000 0000 0101	
AX	0 0035 h	0000 0000 0011 0101	
AX	0 001A h	0000 0000 0001 1010	SHR
AX	0 001F h	0000 0000 0001 1111	XOR

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 00CA h	
0 002E h	RA_NestedProc = 0 0013 h	
0 002C h	0 0060 h	
0 002A h	0 0035 h	

0 0028 h	0 001F h	← SP
0 0026 h		
0 0024 h		

Ret

Error: Ret statement pop top value of stack and mov it in IP register. In this case when ret statement will come then it will pop 0 001F h and mov it in IP register. Which is not required return location. Therefore this will cause an error.

Correction: For correct return we should use a statement before **ret** statement.

ADD SP, 6

Ret

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 00CA h	← SP
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h		

0 0024 h	

DeepProc EndP

Pop Ax

Pop Bp

Registers	
Name	**Value**
AX	0 00CA h
BP	Old value of BP

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0000 h	
0 0038 h	0 0061 h	
0 0036 h	0 0005 h	
0 0034 h	RA_Main = 0 0008 h	← SP
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h		
0 0024 h		

Ret 6

Note: This will mov top value in IP register which is 0 0008 h. And then add 6 to SP.

Stack	
Address	**Data**
0 003C h	
0 003A h	poped
0 0038 h	poped
0 0036 h	poped
0 0034 h	poped
0 0032 h	poped
0 0030 h	poped
0 002E h	poped
0 002C h	poped
0 002A h	poped
0 0028 h	poped
0 0026 h	
0 0024 h	

← SP

NestedProc EndP

Exit

Main EndP

|----------o(Question No 5)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 061h
0 0002 h -->	Mov Bx, 000h

```
0  0003 h  -->          Mov Cx, 3
0  0004 h  -->          Push Ax
0  0005 h  -->          Push Bx
0  0006 h  -->          Push Cx
0  0007 h  -->          Call ComplexProc
0  0008 h  -->          Exit
0  0009 h  -->          Main EndP
0  000A h  -->          ComplexProc Proc
0  000B h  -->          Push Bp
0  000C h  -->          Mov Bp, Sp
0  000D h  -->          Mov Ax, [Bp + 6]
0  000E h  -->          Mov Bx, [Bp + 4]
0  000F h  -->          Add Ax, Bx
0  0010 h  -->          Not Ax
0  0011 h  -->          Push Ax
0  0012 h  -->          Call ShiftProc
0  0013 h  -->          Pop Bp
0  0014 h  -->          Ret 6
0  0015 h  -->          ComplexProc EndP
0  0016 h  -->          ShiftProc Proc
0  0017 h  -->          Mov Dx, 5
0  0018 h  -->          Push Dx
0  0019 h  -->          Mov Cx, 2
0  001A h  -->          Loop2:
0  001B h  -->          Rol Ax, 1
0  001C h  -->          Push Ax
0  001D h  -->          Loop Loop2
0  001E h  -->          Ret
0  001F h  -->          ShiftProc EndP
```

Solution:

Stack	
Address	**Data**
0 003C h	
0 003A h	
0 0038 h	
0 0036 h	
0 0034 h	
0 0032 h	
0 0030 h	
0 002E h	
0 002C h	
0 002A h	

← SP

Main proc

Mov Ax, 061h

Mov Bx, 000h

Mov Cx, 3

Push Ax

Push Bx

Push Cx

Call ComplexProc

Stack	
Address	**Data**
0 003C h	
0 003A h	0 0061 h
0 0038 h	0 0000 h
0 0036 h	0 0003 h

0 0034 h	RA_Main = 0 0008 h	← SP
0 0032 h		
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

ComplexProc Proc

Push Bp

Mov Bp, Sp

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← SP, BP
0 0030 h		
0 002E h		
0 002C h		
0 002A h		

Mov Ax, [Bp + 6]

Mov Bx, [Bp + 4]

Add Ax, Bx

Not Ax

Push Ax

Call ShiftProc

Registers	
Name	**Value**
AX	0 0000 h
BX	0 0003 h
AX	0 0003 h
AX	0 FFFC h

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 FFFC h	
0 002E h	RA_ComplexProc = 0 0013 h	← SP
0 002C h		
0 002A h		

ShiftProc Proc

Mov Dx, 5

Push Dx

Mov Cx, 2

Registers	
Name	**Value**

DX	0 0005 h
CX	0 0002 h

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 FFFC h	
0 002E h	RA_ComplexProc = 0 0013 h	
0 002C h	0 0005 h	← SP
0 002A h		

; when cx = 2

Loop2:

Rol Ax, 1

Push Ax

Loop Loop2

Registers		
Name	**Value**	
CX	0 0002 h	
AX	0 FFFC h	1111 1111 1111 1100
Rol Ax, 1	0 FFF9 h	1111 1111 1111 1001

Stack	
Address	**Data**

0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 FFFC h	
0 002E h	RA_ComplexProc = 0 0013 h	
0 002C h	0 0005 h	
0 002A h	0 FFF9 h	← SP

; when cx = 1

Registers		
Name	**Value**	
CX	0 0001 h	
AX	0 FFF9 h	1111 1111 1111 1001
Rol Ax, 1	0 FFF3 h	1111 1111 1111 0011
CX	0 0000 h	

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 FFFC h	

0 002E h	RA_ComplexProc = 0 0013 h	
0 002C h	0 0005 h	
0 002A h	0 FFF9 h	
0 0028 h	0 FFF3 h	← SP
0 0026 h		
0 0024 h		
0 0022 h		

Ret

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← BP
0 0030 h	0 FFFC h	
0 002E h	RA_ComplexProc = 0 0013 h	
0 002C h	0 0005 h	
0 002A h	0 FFF9 h	← SP
0 0028 h	pop	
0 0026 h		
0 0024 h		
0 0022 h		

Error: Ret statement pop top value of stack and mov it in IP register. In this case when ret statement will come then it will pop 0 FFF3 h and mov it in IP register. Which is not required return location. Therefore this will cause an error.

Correction: For correct return we should use a statement before **ret** statement.

Also use **ret 2** instead of **ret** to pop correct value in BP.

ADD SP, 6

Ret 2

Stack		
Address	**Data**	
0 003C h		
0 003A h	0 0061 h	
0 0038 h	0 0000 h	
0 0036 h	0 0003 h	
0 0034 h	RA_Main = 0 0008 h	
0 0032 h	Old value of BP	← SP, BP
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h		
0 0024 h		
0 0022 h		

ShiftProc EndP

Pop Bp

Ret 6

Note: ret 6 = (SP + 2 + 6)

Registers	
Name	**Value**
BP	Old Address of BP

Stack		
Address	**Data**	
0 003C h		← SP
0 003A h	poped	
0 0038 h	poped	
0 0036 h	poped	
0 0034 h	poped	
0 0032 h	poped	
0 0030 h	poped	
0 002E h	poped	
0 002C h	poped	
0 002A h	poped	
0 0028 h	poped	
0 0026 h		
0 0024 h		
0 0022 h		

ComplexProc EndP

Exit

Main EndP

|----------o(Question No 6)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Eax, 0 0023 h
0 0002 h -->	Mov Ebx, 0 0000 0061h
0 0003 h -->	Push Eax
0 0004 h -->	Push Ebx
0 0005 h -->	Call ProcA
0 0006 h -->	Exit
0 0007 h -->	Main EndP
0 0008 h -->	ProcA Proc
0 0009 h -->	Push Ebp
0 000A h -->	Mov Ebp, Esp
0 000B h -->	Mov Eax, [Ebp + 8]
0 000C h -->	Mov Ebx, [Ebp + 4]
0 000D h -->	Or Eax, Ebx
0 000E h -->	Shr Eax, 2
0 000F h -->	Push Eax
0 0010 h -->	Call ProcB
0 0011 h -->	Pop Ebp
0 0012 h -->	Ret 8
0 0013 h -->	ProcA EndP
0 0014 h -->	ProcB Proc
0 0015 h -->	Mov Ecx, 3
0 0016 h -->	LoopB:
0 0017 h -->	Ror Eax, 1
0 0018 h -->	And Eax, 0FFFFh
0 0019 h -->	Push Eax
0 001A h -->	Loop LoopB
0 001B h -->	Ret

0 001C h --> ProcB EndP

Solution:

Stack		
Address	Data	
0 0000 0064 h		← ESP
0 0000 0060 h		
0 0000 005C h		
0 0000 0058 h		
0 0000 0054 h		
0 0000 0050 h		
0 0000 004C h		
0 0000 0048 h		
0 0000 0044 h		
0 0000 0040 h		

Main proc

Mov Eax, 0 0023 h

Mov Ebx, 0 0000 0061h

Push Eax

Push Ebx

Call ProcA

Registers	
Name	Value
EAX	0 0000 0023 h
EBX	0 0000 0061 h

Stack	
Address	Data

0 0000 0064 h		
0 0000 0060 h	0 0000 0023 h	
0 0000 005C h	0 0000 0061 h	
0 0000 0058 h	RA_Main = 0 0006 h	← ESP
0 0000 0054 h		
0 0000 0050 h		
0 0000 004C h		
0 0000 0048 h		
0 0000 0044 h		
0 0000 0040 h		

```
ProcA Proc
Push Ebp
Mov Ebp, Esp
Mov Eax, [Ebp + 8]
Mov Ebx, [Ebp + 4]
Or Eax, Ebx
Shr Eax, 2
Push Eax
Call ProcB
```

Registers			
Name	**Value**		
EAX	0 0000 0061 h	0000 0000 0000 0000 0000 0000 0110 0001	
EBX	0 0000 0006 h	0000 0000 0000 0000 0000 0000 0000 0110	
EAX	0 0000 0067 h	0000 0000 0000 0000 0000 0000 0110 0111	OR
EAX	0 0000 0019 h	0000 0000 0000 0000 0000 0000 0001 1001	SHR 2

Stack	
Address	**Data**

0 0000 0064 h		
0 0000 0060 h	0 0000 0023 h	
0 0000 005C h	0 0000 0061 h	
0 0000 0058 h	RA_Main = 0 0000 0006 h	
0 0000 0054 h	Old value of EBP	← EBP
0 0000 0050 h	0 0000 0019 h	
0 0000 004C h	RA_ProcA = 0 0000 0011 h	← ESP
0 0000 0048 h		
0 0000 0044 h		
0 0000 0040 h		

```
ProcB Proc
Mov Ecx, 3
LoopB:
Ror Eax, 1
And Eax, 0FFFFh
Push Eax
Loop LoopB
```

Registers			
Name	**Value**		
ECX	0 0000 0003 h		
EAX	0 0000 0019 h	0000 0000 0000 0000 0000 0000 0001 1001	
EAX	0 8000 000C h	1000 0000 0000 0000 0000 0000 0000 1100	ROR 1
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0000 000C h	0000 0000 0000 0000 0000 0000 0000 1100	AND
ECX	0 0000 0002 h		
EAX	0 0000 000C h	0000 0000 0000 0000 0000 0000 0000 1100	
EAX	0 0000 0006 h	0000 0000 0000 0000 0000 0000 0000 0110	ROR 1

	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0000 0006 h	0000 0000 0000 0000 0000 0000 0000 0110	AND
ECX	0 0000 0001 h		
EAX	0 0000 0006 h	0000 0000 0000 0000 0000 0000 0000 0110	
EAX	0 0000 0003 h	0000 0000 0000 0000 0000 0000 0000 0011	ROR 1
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0000 0003 h	0000 0000 0000 0000 0000 0000 0000 0011	AND
ECX	0 0000 0000 h		

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 0000 0023 h	
0 0000 005C h	0 0000 0061 h	
0 0000 0058 h	RA_Main = 0 0000 0006 h	
0 0000 0054 h	Old value of EBP	← EBP
0 0000 0050 h	0 0000 0019 h	
0 0000 004C h	RA_ProcA = 0 0000 0011 h	
0 0000 0048 h	0 0000 000C h	
0 0000 0044 h	0 0000 0006 h	
0 0000 0040 h	0 0000 0003 h	← ESP

Ret

Error: Ret statement pop top value of stack and mov it in IP register. In this case when ret statement will come then it will pop 0 0000 0003 h and mov it in IP register. Which is not required return location. Therefore this will cause an error.

And also use **ret 4** instead of **ret** to pop correct address of BP.

Correction: For correct return we should use a statement before **ret** statement.

ADD SP, 12

Ret 4

Note: Ret 4 will mov top of Stack to IP register and then add 4 to SP.

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 0000 0023 h	
0 0000 005C h	0 0000 0061 h	
0 0000 0058 h	RA_Main = 0 0000 0006 h	
0 0000 0054 h	Old value of EBP	← ESP, EBP
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	

ProcB EndP

Pop Ebp

Registers	
Name	**Value**
EBP	Old value of EBP

Stack	
Address	**Data**

0 0000 0064 h		
0 0000 0060 h	0 0000 0023 h	
0 0000 005C h	0 0000 0061 h	
0 0000 0058 h	RA_Main = 0 0000 0006 h	← ESP
0 0000 0054 h	poped	
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	

Ret 8

Stack		
Address	**Data**	
0 0000 0064 h		← ESP
0 0000 0060 h	poped	
0 0000 005C h	poped	
0 0000 0058 h	poped	
0 0000 0054 h	poped	
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	

ProcA EndP

Exit

Main EndP

|----------o(Question No 7)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Eax, 0 0023 2361h
0 0002 h -->	Mov Ebx, 0 2361 0023 h
0 0003 h -->	Mov Ecx, 4
0 0004 h -->	Push Eax
0 0005 h -->	Push Ebx
0 0006 h -->	Push Ecx
0 0007 h -->	Call MainProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	MainProc Proc
0 000B h -->	Push Ebp
0 000C h -->	Mov Ebp, Esp
0 000D h -->	Mov Eax, [Ebp + 8]
0 000E h -->	Mov Ebx, [Ebp + 4]
0 000F h -->	Xor Eax, Ebx
0 0010 h -->	Shl Eax, 1
0 0011 h -->	Push Eax
0 0012 h -->	Call LogicalProc
0 0013 h -->	Pop Ebp
0 0014 h -->	Ret 8
0 0015 h -->	MainProc EndP
0 0016 h -->	LogicalProc Proc
0 0017 h -->	Mov Edx, 7
0 0018 h -->	Push Edx
0 0019 h -->	LoopLogic:
0 001A h -->	Rol Eax, 2

0 001B h --> Or Eax, 0FFFFh

0 001C h --> Push Eax

0 001D h --> Loop LoopLogic

0 001E h --> Ret

0 001F h --> LogicalProc EndP

Solution:

Stack	
Address	**Data**
0 0000 0064 h	
0 0000 0060 h	
0 0000 005C h	
0 0000 0058 h	
0 0000 0054 h	
0 0000 0050 h	
0 0000 004C h	
0 0000 0048 h	
0 0000 0044 h	
0 0000 0040 h	

← ESP

Main proc

Mov Eax, 0 0023 2361h

Mov Ebx, 0 2361 0023 h

Mov Ecx, 4

Push Eax

Push Ebx

Push Ecx

Call MainProc

Registers

Name	Value
EAX	0 0023 2361 h
EBX	0 2361 0023 h
ECX	0 0000 0004 h

Stack	
Address	**Data**
0 0000 0064 h	
0 0000 0060 h	0 0023 2361 h
0 0000 005C h	0 2361 0023 h
0 0000 0058 h	0 0000 0004 h
0 0000 0054 h	RA_Main = 0 0008 h
0 0000 0050 h	
0 0000 004C h	
0 0000 0048 h	
0 0000 0044 h	
0 0000 0040 h	

← ESP

```
MainProc Proc
Push Ebp
Mov Ebp, Esp
Mov Eax, [Ebp + 8]
Mov Ebx, [Ebp + 4]
Xor Eax, Ebx
Shl Eax, 1
Push Eax
Call LogicalProc
```

Registers			
Name	**Value**		

EAX	0 0000 0004 h	0000 0000 0000 0000 0000 0000 0000 0100	
EBX	0 0000 0008 h	0000 0000 0000 0000 0000 0000 0000 1000	
EAX	0 0000 000C h	0000 0000 0000 0000 0000 0000 0000 1100	XOR
EAX	0 0000 0018 h	0000 0000 0000 0000 0000 0000 0001 1000	SHl 1

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 0023 2361 h	
0 0000 005C h	0 2361 0023 h	
0 0000 0058 h	0 0000 0004 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← EBP
0 0000 004C h	0 0000 0018 h	
0 0000 0048 h	RA_MainProc = 0 0013 h	← ESP
0 0000 0044 h		
0 0000 0040 h		

```
LogicalProc Proc
Mov Edx, 7
Push Edx
LoopLogic:
Rol Eax, 2
Or Eax, 0FFFFh
Push Eax
Loop LoopLogic
```

Registers			
Name	**Value**		

EDX	0 0000 0007 h		
ECX	0 0000 0004 h		
EAX	0 0000 0018 h	0000 0000 0000 0000 0000 0000 0001 1000	
EAX	0 0000 0060 h	0000 0000 0000 0000 0000 0000 0110 0000	ROl 2
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	OR
ECX	0 0000 0003 h		
EAX	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0003FFFC h	0000 0000 0000 0011 1111 1111 1111 1100	ROl 2
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 0003 FFFF h	0000 0000 0000 0011 1111 1111 1111 1111	OR
ECX	0 0000 0002 h		
EAX	0 0003 FFFF h	0000 0000 0000 0011 1111 1111 1111 1111	
EAX	0 000F FFFC h	0000 0000 0000 1111 1111 1111 1111 1100	ROl 2
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 000F FFFF h	0000 0000 0000 1111 1111 1111 1111 1111	OR
ECX	0 0000 0001 h		
EAX	0 000F FFFF h	0000 0000 0000 1111 1111 1111 1111 1111	
EAX	0 003F FFFC h	0000 0000 0011 1111 1111 1111 1111 1100	ROl 2
	0 0000 FFFF h	0000 0000 0000 0000 1111 1111 1111 1111	
EAX	0 003F FFFF h	0000 0000 0011 1111 1111 1111 1111 1111	OR
ECX	0 0000 0000 h		

Stack	
Address	**Data**
0 0000 0064 h	
0 0000 0060 h	0 0023 2361 h

0 0000 005C h	0 2361 0023 h	
0 0000 0058 h	0 0000 0004 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← EBP
0 0000 004C h	0 0000 0018 h	
0 0000 0048 h	RA_MainProc = 0 0013 h	
0 0000 0044 h	0 0000 0007 h	
0 0000 0040 h	0 0000 FFFF h	
0 0000 003C h	0 0003 FFFF h	
0 0000 0038 h	0 000F FFFF h	← ESP
0 0000 0034 h		
0 0000 0030 h		

Ret

Error: Ret statement pop top value of stack and mov it in IP register. In this case when ret statement will come then it will pop 0 000F FFFF h and mov it in EIP register. Which is not required return location. Therefore this will cause an error.

And also use **ret 4** instead of **ret** to pop correct value of BP.

Correction: For correct return we should use a statement before **ret 4** statement.

ADD SP, 16

Ret 4

Note: Ret 4 will mov top of Stack to EIP register and then add 4 to ESP.

Stack	
Address	**Data**
0 0000 0064 h	

0 0000 0060 h	0 0023 2361 h	
0 0000 005C h	0 2361 0023 h	
0 0000 0058 h	0 0000 0004 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← ESP, EBP
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	
0 0000 003C h	poped	
0 0000 0038 h	poped	
0 0000 0034 h		
0 0000 0030 h		

LogicalProc EndP

Pop Ebp

Registers	
Name	**Value**
EBP	Old value of EBP

Stack	
Address	**Data**
0 0000 0064 h	
0 0000 0060 h	0 0023 2361 h
0 0000 005C h	0 2361 0023 h
0 0000 0058 h	0 0000 0004 h

0 0000 0054 h	RA_Main = 0 0008 h	← ESP
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	
0 0000 003C h	poped	
0 0000 0038 h	poped	
0 0000 0034 h		
0 0000 0030 h		

Ret 8

Note: Use Ret 12 instead of ret 8 to clear Stack correctly. Ret 12 first mov Top value to EIP Redister and then add 12 to ESP.

Stack		
Address	**Data**	
0 0000 0064 h		← ESP
0 0000 0060 h	poped	
0 0000 005C h	poped	
0 0000 0058 h	poped	
0 0000 0054 h	poped	
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	

0 0000 003C h	poped
0 0000 0038 h	poped
0 0000 0034 h	
0 0000 0030 h	

MainProc EndP

Exit

Main EndP

|----------o(Question No 8)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Eax, 0 5566 0023 h
0 0002 h -->	Mov Ebx, 0 2361 BCCDh
0 0003 h -->	Mov Ecx, 3
0 0004 h -->	Push Eax
0 0005 h -->	Push Ebx
0 0006 h -->	Push Ecx
0 0007 h -->	Call ComputeProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	ComputeProc Proc
0 000B h -->	Push Ebp
0 000C h -->	Mov Ebp, Esp
0 000D h -->	Mov Eax, [Ebp + 8]
0 000E h -->	Mov Ebx, [Ebp + 4]
0 000F h -->	And Eax, Ebx
0 0010 h -->	Not Eax

```
0  0011 h  -->        Push Eax
0  0012 h  -->        Call ShiftLogic
0  0013 h  -->        Pop Ebp
0  0014 h  -->        Ret 8
0  0015 h  -->        ComputeProc EndP
0  0016 h  -->        ShiftLogic Proc
0  0017 h  -->        Mov Edx, 6
0  0018 h  -->        Push Edx
0  0019 h  -->        Mov Ecx, 2
0  001A h  -->        LoopShift:
0  001B h  -->        Ror Eax, 1
0  001C h  -->        Push Eax
0  001D h  -->        Loop LoopShift
0  001E h  -->        Ret
0  001F h  -->        ShiftLogic EndP
```

Solution:

Stack		
Address	**Data**	
0 0000 0064 h		← ESP
0 0000 0060 h		
0 0000 005C h		
0 0000 0058 h		
0 0000 0054 h		
0 0000 0050 h		
0 0000 004C h		
0 0000 0048 h		
0 0000 0044 h		
0 0000 0040 h		

Main proc

Mov Eax, 0 5566 0023 h

Mov Ebx, 0 2361 BCCDh

Mov Ecx, 3

Push Eax

Push Ebx

Push Ecx

Call ComputeProc

Registers	
Name	**Value**
EAX	0 5566 0023 h
EBX	0 2361 BCCDh
ECX	0 0000 0003 h

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 5566 0023 h	
0 0000 005C h	0 2361 BCCDh	
0 0000 0058 h	0 0000 0003 h	
0 0000 0054 h	RA_Main = 0 0008 h	← ESP
0 0000 0050 h		
0 0000 004C h		
0 0000 0048 h		
0 0000 0044 h		
0 0000 0040 h		

ComputeProc Proc

Push Ebp

Mov Ebp, Esp

Mov Eax, [Ebp + 8]

Mov Ebx, [Ebp + 4]

And Eax, Ebx

Not Eax

Push Eax

Call ShiftLogic

Registers			
Name	Value		
EAX	0 0000 0003 h	0000 0000 0000 0000 0000 0000 0000 0011	
EBX	0 0000 0008 h	0000 0000 0000 0000 0000 0000 0000 1000	
EAX	0 0000 0000 h	0000 0000 0000 0000 0000 0000 0000 0000	AND
EAX	0 FFFF FFFF h	1111 1111 1111 1111 1111 1111 1111 1111	Not

Stack		
Address	Data	
0 0000 0064 h		
0 0000 0060 h	0 5566 0023 h	
0 0000 005C h	0 2361 BCCDh	
0 0000 0058 h	0 0000 0003 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← EBP
0 0000 004C h	0 FFFF FFFF h	
0 0000 0048 h	RA_ComputeProc = 0 0013 h	← ESP
0 0000 0044 h		
0 0000 0040 h		

```
ShiftLogic Proc
Mov Edx, 6
Push Edx
Mov Ecx, 2
LoopShift:
Ror Eax, 1
Push Eax
Loop LoopShift
```

Registers			
Name	**Value**		
EDX	0 0000 0006 h		
ECX	0 0000 0002 h		
EAX	0 FFFF FFFF h	1111 1111 1111 1111 1111 1111 1111 1111	
EAX	0 FFFF FFFF h	1111 1111 1111 1111 1111 1111 1111 1111	ROR 1
ECX	0 0000 0001 h		
EAX	0 FFFF FFFF h	1111 1111 1111 1111 1111 1111 1111 1111	
EAX	0 FFFF FFFF h	1111 1111 1111 1111 1111 1111 1111 1111	ROR 1

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 5566 0023 h	
0 0000 005C h	0 2361 BCCDh	
0 0000 0058 h	0 0000 0003 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← EBP
0 0000 004C h	0 FFFF FFFF h	

0 0000 0048 h	RA_ComputeProc = 0 0013 h	
0 0000 0044 h	0 FFFF FFFF h	
0 0000 0040 h	0 FFFF FFFF h	← ESP
0 0000 003C h		
0 0000 0038 h		
0 0000 0034 h		
0 0000 0030 h		

Ret

Error: Ret statement pop top value of stack and mov it in IP register. In this case when ret statement will come then it will pop 0 FFFF FFFF h and mov it in EIP register. Which is not required return location. Therefore this will cause an error.

And also use **ret 4** instead of **ret** to pop correct value of BP.

Correction: For correct return we should use a statement before **ret 4** statement.

ADD SP, 8

Ret 4

Note: Ret 4 will mov top of Stack to EIP register and then add 4 to ESP.

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 5566 0023 h	
0 0000 005C h	0 2361 BCCDh	
0 0000 0058 h	0 0000 0003 h	
0 0000 0054 h	RA_Main = 0 0008 h	
0 0000 0050 h	Old value of EBP	← ESP, EBP

0 0000 004C h	poped
0 0000 0048 h	poped
0 0000 0044 h	poped
0 0000 0040 h	poped
0 0000 003C h	
0 0000 0038 h	
0 0000 0034 h	
0 0000 0030 h	

ShiftLogic EndP

Pop Ebp

Registers	
Name	**Value**
EBP	Old value of EBP

Stack		
Address	**Data**	
0 0000 0064 h		
0 0000 0060 h	0 5566 0023 h	
0 0000 005C h	0 2361 BCCDh	
0 0000 0058 h	0 0000 0003 h	
0 0000 0054 h	RA_Main = 0 0008 h	← ESP
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	

0 0000 0044 h	poped
0 0000 0040 h	poped
0 0000 003C h	
0 0000 0038 h	
0 0000 0034 h	
0 0000 0030 h	

Ret 8

Note: Use Ret 12 instead of ret 8 to clear Stack correctly. Ret 12 first mov Top value to EIP Redister and then add 12 to ESP.

Ret 12

Stack		
Address	**Data**	
0 0000 0064 h		← ESP
0 0000 0060 h	poped	
0 0000 005C h	poped	
0 0000 0058 h	poped	
0 0000 0054 h	poped	
0 0000 0050 h	poped	
0 0000 004C h	poped	
0 0000 0048 h	poped	
0 0000 0044 h	poped	
0 0000 0040 h	poped	
0 0000 003C h		
0 0000 0038 h		
0 0000 0034 h		
0 0000 0030 h		

ComputeProc EndP

Exit

Main EndP

|----------o(Question No 9)o----------|

Corrected Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 2h
0 0002 h -->	Mov Bx, 4h
0 0003 h -->	Push Ax
0 0004 h -->	Push Bx
0 0005 h -->	Mov Cx, 10
0 0006 h -->	Push Cx
0 0007 h -->	Call loopingProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	loopingProc Proc
0 000B h -->	L1:
0 000C h -->	Push Bp
0 000D h -->	Mov Bp, Sp
0 000E h -->	Mov Ax, [Bp + 6]
0 000F h -->	Mov Bx, [Bp + 4]
0 0010 h -->	Add Ax, Bx
0 0011 h -->	Push Ax
0 0012 h -->	Push Bx
0 0013 h -->	Loop L1
0 0014 h -->	Add SP, 58
0 0015 h -->	Pop BP

0 0016 h --> Ret 6

0 0017 h --> loopingProc EndP

Solution:

Stack	
Address	**Data**
0 0064 h	← SP
0 0062 h	
0 0060 h	
0 005E h	
0 005C h	
0 005A h	
0 0058 h	
0 0056 h	
0 0054 h	
0 0052 h	

Main proc

Mov Ax, 2h

Mov Bx, 4h

Push Ax

Push Bx

Mov Cx, 10

Push Cx

Call loopingProc

Registers	
Name	**Value**

AX	0 0002 h
BX	0 0004 h
CX	0 000a h

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 0002 h	
0 0060 h	0 0004 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	← SP
0 005A h		
0 0058 h		
0 0056 h		
0 0054 h		
0 0052 h		

```
loopingProc Proc
L1:
Push Bp
Mov Bp, Sp
Mov Ax, [Bp + 6]
Mov Bx, [Bp + 4]
Add Ax, Bx
Push Ax
Push Bx
Loop L1
```

Registers	

Name	Value	
CX	0 000a h	
AX	0 0004 h	
BX	0 000a h	
AX	0 000e h	ADD
CX	0 0009 h	
AX	0 0064 h	
BX	0 000e h	
AX	0 0072 h	ADD
CX	0 0008 h	
AX	0 005A h	
BX	0 0072 h	
AX	0 00CC h	ADD
CX	0 0007 h	
AX	0 0054 h	
BX	0 00CC h	
AX	0 0120 h	ADD
CX	0 0006 h	
AX	0 004E h	
BX	0 0120 h	
AX	0 016E h	ADD
CX	0 0005 h	
AX	0 0048 h	
BX	0 016E h	
AX	0 01B6 h	ADD
CX	0 0004 h	

AX	0 0042 h	
BX	0 01B6 h	
AX	0 01F8 h	ADD
CX	0 0003 h	
AX	0 003C h	
BX	0 01F8 h	
AX	0 0234 h	ADD
CX	0 0002 h	
AX	0 0036 h	
BX	0 0234 h	
AX	0 026A h	ADD
CX	0 0001 h	
AX	0 0030 h	
BX	0 026A h	
AX	0 029A h	ADD
CX	0 0000 h	

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0002 h
0 0060 h	0 0004 h
0 005E h	0 000a h
0 005C h	RA_Main = 0 0008 h
0 005A h	Old value of BP (let = 0 0064 h)
0 0058 h	0 000e h

0 0056 h	0 000a h
0 0054 h	0 005A h
0 0052 h	0 0072 h
0 0050 h	0 000e h
0 004E h	0 0054 h
0 004C h	0 00CC h
0 004A h	0 0072 h
0 0048 h	0 004E h
0 0046 h	0 0120 h
0 0044 h	0 00CC h
0 0042 h	0 0048 h
0 0040 h	0 016E h
0 003E h	0 0120 h
0 003C h	0 0042 h
0 003A h	0 01B6 h
0 0038 h	0 016E h
0 0036 h	0 003C h
0 0034 h	0 01F8 h
0 0032 h	0 01B6 h
0 0030 h	0 0036 h
0 002E h	0 0234 h
0 002C h	0 01F8 h
0 002A h	0 0030 h
0 0028 h	0 026A h
0 0026 h	0 0234 h
0 0024 h	0 002A h
0 0022 h	0 029A h

← BP

0 0020 h	0 026A h	← SP

Add SP, 58

Pop BP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0002 h
0 0060 h	0 0004 h
0 005E h	0 000a h
0 005C h	RA_Main = 0 0008 h ← SP
0 005A h	poped
0 0058 h	poped
0 0056 h	poped
0 0054 h	poped
0 0052 h	poped
0 0050 h	poped
0 004E h	poped
0 004C h	poped
0 004A h	poped
0 0048 h	poped
0 0046 h	poped
0 0044 h	poped
0 0042 h	poped
0 0040 h	poped
0 003E h	poped
0 003C h	poped

0 003A h	poped
0 0038 h	poped
0 0036 h	poped
0 0034 h	poped
0 0032 h	poped
0 0030 h	poped
0 002E h	poped
0 002C h	poped
0 002A h	poped
0 0028 h	poped
0 0026 h	poped
0 0024 h	poped
0 0022 h	poped
0 0020 h	poped

Ret 6

Stack	
Address	**Data**
0 0064 h	
0 0062 h	poped
0 0060 h	poped
0 005E h	poped
0 005C h	poped
0 005A h	poped
0 0058 h	poped
0 0056 h	poped
0 0054 h	poped

← SP

0 0052 h	poped
0 0050 h	poped
0 004E h	poped
0 004C h	poped
0 004A h	poped
0 0048 h	poped
0 0046 h	poped
0 0044 h	poped
0 0042 h	poped
0 0040 h	poped
0 003E h	poped
0 003C h	poped
0 003A h	poped
0 0038 h	poped
0 0036 h	poped
0 0034 h	poped
0 0032 h	poped
0 0030 h	poped
0 002E h	poped
0 002C h	poped
0 002A h	poped
0 0028 h	poped
0 0026 h	poped
0 0024 h	poped
0 0022 h	poped
0 0020 h	poped

loopingProc EndP

Exit

Main EndP

|----------o(Question No 10)o----------|

Corrected Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 0 0023 h
0 0002 h -->	Mov Bx, 0 2361 h
0 0003 h -->	Mov Cx, 000Ah
0 0004 h -->	Push Ax
0 0005 h -->	Push Bx
0 0006 h -->	Push Cx
0 0007 h -->	Call FirstProc
0 0008 h -->	Exit
0 0009 h -->	Main EndP
0 000A h -->	FirstProc Proc
0 000B h -->	Push Bp
0 000C h -->	Mov Bp, Sp
0 000D h -->	Mov Dx, [Bp + 6]
0 000E h -->	Mov Ax, [Bp + 4]
0 000F h -->	Rol Ax, 2
0 0010 h -->	Add Ax, Dx
0 0011 h -->	Push Ax
0 0012 h -->	Call SecondProc
0 0013 h -->	Pop BP
0 0014 h -->	Ret 6
0 0015 h -->	FirstProc EndP
0 0016 h -->	SecondProc Proc
0 0017 h -->	Push Bp
0 0018 h -->	Mov Bp, Sp

```
0  0019 h  -->          Mov Bx, [Bp + 4]
0  001A h  -->          Mov Ax, [Bp + 6]
0  001B h  -->          Ror Bx, 3
0  001C h  -->          Sub Ax, Bx
0  001D h  -->          Push Ax
0  001E h  -->          Call ThirdProc
0  001F h  -->          Pop BP
0  0020 h  -->          Ret 2
0  0021 h  -->          SecondProc EndP
0  0022 h  -->          ThirdProc Proc
0  0023 h  -->          Push Bp
0  0024 h  -->          Mov Bp, Sp
0  0025 h  -->          Mov Cx, 5
0  0026 h  -->          Loop1:
0  0027 h  -->          Shl Ax, 1
0  0028 h  -->          Or Ax, 0001h
0  0029 h  -->          Push Ax
0  002A h  -->          Dec Cx
0  002B h  -->          Jnz Loop1
0  002C h  -->          Call FourthProc
0  002D h  -->          Add SP , 10
0  002E h  -->          Pop BP
0  002F h  -->          Ret 2
0  0030 h  -->          ThirdProc EndP
0  0031 h  -->          FourthProc Proc
0  0032 h  -->          Push Bp
0  0033 h  -->          Mov Bp, Sp
0  0034 h  -->          And Ax, Bx
0  0035 h  -->          Xor Ax, 0FFFFh
0  0036 h  -->          Push Ax
0  0037 h  -->          Mov Dx, [Bp + 4]
0  0038 h  -->          ADD SP, 2
```

0 0039 h --> Pop BP

0 003A h --> Ret

0 003B h --> FourthProc EndP

Solution:

Stack	
Address	**Data**
0 0064 h	← SP
0 0062 h	
0 0060 h	
0 005E h	
0 005C h	
0 005A h	
0 0058 h	
0 0056 h	
0 0054 h	
0 0052 h	

Main proc

Mov Ax, 0 0023 h

Mov Bx, 0 2361 h

Mov Cx, 000Ah

Push Ax

Push Bx

Push Cx

Call FirstProc

Registers	
Name	**Value**

AX	0 0023 h
BX	0 2361 h
CX	0 000a h

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 0023 h	
0 0060 h	0 2361 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	← SP
0 005A h		
0 0058 h		
0 0056 h		
0 0054 h		
0 0052 h		

```
FirstProc Proc
Push Bp
Mov Bp, Sp
Mov Dx, [Bp + 6]
Mov Ax, [Bp + 4]
Rol Ax, 2
Add Ax, Dx
Push Ax
Call SecondProc
```

Registers			
Name	**Value**		

DX	0 2361 h		
AX	0 000a h	0000 0000 0000 1010	
AX	0 0028 h	0000 0000 0010 1000	ROL 2
AX	0 2389 h		ADD

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 0023 h	
0 0060 h	0 2361 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP	← BP
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	← SP
0 0054 h		
0 0052 h		

```
SecondProc Proc
Push Bp
Mov Bp, Sp
Mov Bx, [Bp + 4]
Mov Ax, [Bp + 6]
Ror Bx, 3
Sub Ax, Bx
Push Ax
Call ThirdProc
```

Registers		

Name	Value		
BX	0 2389 h	0010 0011 1000 1001	
AX	0 0064 h		
BX	0 2471 h	0010 0100 0111 0001	ROR 3
AX	0 DBF3 h		SUB

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 0023 h	
0 0060 h	0 2361 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	
0 0054 h	0 005A h	← BP
0 0052 h	0 DBF3 h	
0 0050 h	RA_SecondProc = 0 001E h	← SP
0 004E h		
0 004C h		
0 004A h		
0 0048 h		
0 0046 h		
0 0044 h		

ThirdProc Proc

```
Push Bp
Mov Bp, Sp
Mov Cx, 5
Loop1:
Shl Ax, 1
Or Ax, 0001h
Push Ax
Dec Cx
Jnz Loop1
Call FourthProc
```

Registers			
Name	**Value**		
CX	0 0005 h		
AX	0 DBF3 h	1101 1011 1111 0011	
AX	0 B7E6 h	1011 0111 1110 0110	SHL
	0 0001 h	0000 0000 0000 0001	
AX	0 B7E7 h	1011 0111 1110 0111	OR
CX	0 0004 h		
AX	0 B7E7 h	1011 0111 1110 0111	
AX	0 6FCE h	0110 1111 1100 1110	SHL
	0 0001 h	0000 0000 0000 0001	
AX	0 6FCF h	0110 1111 1100 1111	OR
CX	0 0003 h		
AX	0 6FCF h	0110 1111 1100 1111	
AX	0 DF9E h	1101 1111 1001 1110	SHL
	0 0001 h	0000 0000 0000 0001	
AX	0 DF9F h	1101 1111 1001 1111	OR
CX	0 0002 h		

AX	0 DF9F h	1101 1111 1001 1111	
AX	0 BF3E h	1011 1111 0011 1110	SHL
	0 0001 h	0000 0000 0000 0001	
AX	0 BF3F h	1011 1111 0011 1111	OR
CX	0 0001 h		
AX	0 BF3F h	1011 1111 0011 1111	
AX	0 7E7E h	0111 1110 0111 1110	SHL
	0 0001 h	0000 0000 0000 0001	
AX	0 7E7F h	0111 1110 0111 1111	OR
CX	0 0000 h		

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 0023 h	
0 0060 h	0 2361 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	
0 0054 h	0 005A h	
0 0052 h	0 DBF3 h	
0 0050 h	RA_SecondProc = 0 001E h	
0 004E h	0 0054 h	← BP
0 004C h	0 B7E7 h	

0 004A h	0 6FCF h	
0 0048 h	0 DF9F h	
0 0046 h	0 BF3F h	
0 0044 h	0 7E7F h	
0 0042 h	RA_ThirdProc = 0 002B h	← SP
0 0040 h		
0 003E h		

FourthProc Proc

Push Bp

Mov Bp, Sp

And Ax, Bx

Xor Ax, 0FFFFh

Push Ax

Mov Dx, [Bp + 4]

Registers			
Name	**Value**		
AX	0 7E7F h	0111 1110 0111 1111	
BX	0 2471 h	0010 0100 0111 0001	
AX	0 2471 h	0010 0100 0111 0001	AND
	0 FFFF h	1111 1111 1111 1111	
AX	0 DB8E h	1101 1011 1000 1110	XOR
DX	0 7E7F h		

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0023 h

0 0060 h	0 2361 h	
0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	
0 0054 h	0 005A h	
0 0052 h	0 DBF3 h	
0 0050 h	RA_SecondProc = 0 001E h	
0 004E h	0 0054 h	
0 004C h	0 B7E7 h	
0 004A h	0 6FCF h	
0 0048 h	0 DF9F h	
0 0046 h	0 BF3F h	
0 0044 h	0 7E7F h	
0 0042 h	RA_ThirdProc = 0 002B h	
0 0040 h	0 004E h	← BP
0 003E h	0 DB8E h	← SP

ADD SP, 2

Pop BP

Ret

FourthProc EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0023 h
0 0060 h	0 2361 h

0 005E h	0 000a h	
0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	
0 0054 h	0 005A h	
0 0052 h	0 DBF3 h	
0 0050 h	RA_SecondProc = 0 001E h	
0 004E h	0 0054 h	← BP
0 004C h	0 B7E7 h	
0 004A h	0 6FCF h	
0 0048 h	0 DF9F h	
0 0046 h	0 BF3F h	
0 0044 h	0 7E7F h	← SP
0 0042 h	poped	
0 0040 h	poped	
0 003E h	poped	

Add SP , 10

Pop BP

Ret 2

ThirdProc EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0023 h
0 0060 h	0 2361 h
0 005E h	0 000a h

0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	
0 0058 h	0 2389 h	
0 0056 h	RA_FirstProc = 0 0012 h	
0 0054 h	0 005A h	← SP, BP
0 0052 h	poped	
0 0050 h	poped	
0 004E h	poped	
0 004C h	poped	
0 004A h	poped	
0 0048 h	poped	
0 0046 h	poped	
0 0044 h	poped	
0 0042 h	poped	
0 0040 h	poped	
0 003E h	poped	

Pop BP

Ret 2

SecondProc EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 0023 h
0 0060 h	0 2361 h
0 005E h	0 000a h

0 005C h	RA_Main = 0 0008 h	
0 005A h	Old Value of BP = let(0 0064 h)	← SP, BP
0 0058 h	poped	
0 0056 h	poped	
0 0054 h	poped	
0 0052 h	poped	
0 0050 h	poped	
0 004E h	poped	
0 004C h	poped	
0 004A h	poped	
0 0048 h	poped	
0 0046 h	poped	
0 0044 h	poped	
0 0042 h	poped	
0 0040 h	poped	
0 003E h	poped	

Pop BP

Ret 6

FirstProc EndP

Exit

Main EndP

Stack		
Address	**Data**	
0 0064 h		← SP
0 0062 h	poped	
0 0060 h	poped	
0 005E h	poped	

0 005C h	poped
0 005A h	poped
0 0058 h	poped
0 0056 h	poped
0 0054 h	poped
0 0052 h	poped
0 0050 h	poped
0 004E h	poped
0 004C h	poped
0 004A h	poped
0 0048 h	poped
0 0046 h	poped
0 0044 h	poped
0 0042 h	poped
0 0040 h	poped
0 003E h	poped

|----------o(Question No 11)o----------|

Code:

Address	Instruction
0 0000 h -->	Main proc
0 0001 h -->	Mov Ax, 0 2361 h
0 0002 h -->	Mov Bx, 0 0023 h
0 0003 h -->	Push Ax
0 0004 h -->	Push Bx
0 0005 h -->	Call ProcA
0 0006 h -->	Exit

```
0  0007 h  -->          Main EndP
0  0008 h  -->          ProcA Proc
0  0009 h  -->          Push Bp
0  000A h  -->          Mov Bp, Sp
0  000B h  -->          Mov Cx, [Bp + 4]
0  000C h  -->          And Cx, 0FFh
0  000D h  -->          Push Cx
0  000E h  -->          Call ProcB
0  000F h  -->          Add SP, 2
0  0010 h  -->          Pop BP
0  0011 h  -->          Ret 4
0  0012 h  -->          ProcA EndP
0  0013 h  -->          ProcB Proc
0  0014 h  -->          Push Bp
0  0015 h  -->          Mov Bp, Sp
0  0016 h  -->          Mov Dx, [Bp + 4]
0  0017 h  -->          Or Dx, 0F0F0h
0  0018 h  -->          Push Dx
0  0019 h  -->          Call ProcC
0  001A h  -->          Add SP, 2
0  001B h  -->          Pop BP
0  001C h  -->          Ret
0  001D h  -->          ProcB EndP
0  001E h  -->          ProcC Proc
0  001F h  -->          Push Bp
0  0020 h  -->          Mov Bp, Sp
0  0021 h  -->          Mov Ax, 5
0  0022 h  -->          Mov Bx, [Bp + 4]
0  0023 h  -->          LoopC:
0  0024 h  -->          Xor Bx, 0F0Fh
0  0025 h  -->          Push Bx
0  0026 h  -->          Dec Ax
```

```
0  0027 h  -->        Jnz LoopC
0  0028 h  -->        Call ProcD
0  0029 h  -->        Add Sp, 10
0  002A h  -->        Pop BP
0  002B h  -->        Ret
0  002C h  -->        ProcC EndP
0  002D h  -->        ProcD Proc
0  002E h  -->        Push Bp
0  002F h  -->        Mov Bp, Sp
0  0030 h  -->        Mov Bx, [Bp + 4]
0  0031 h  -->        Not Bx
0  0032 h  -->        Push Bx
0  0033 h  -->        Add SP, 2
0  0034 h  -->        Pop BP
0  0035 h  -->        Ret
0  0036 h  -->        ProcD EndP
```

Solution:

Stack	
Address	**Data**
0 0064 h	← SP
0 0062 h	
0 0060 h	
0 005E h	
0 005C h	
0 005A h	
0 0058 h	
0 0056 h	
0 0054 h	

0 0052 h	

Main proc

Mov Ax, 0 2361 h

Mov Bx, 0 0023 h

Push Ax

Push Bx

Call ProcA

Registers	
Name	**Value**
AX	0 2361 h
BX	0 0023 h

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 2361 h	
0 0060 h	0 0023 h	
0 005E h	RA_Main = 0 0006 h	← SP
0 005C h		
0 005A h		
0 0058 h		
0 0056 h		
0 0054 h		
0 0052 h		

ProcA Proc

Push Bp

Mov Bp, Sp

Mov Cx, [Bp + 4]

And Cx, 0FFh

Push Cx

Call ProcB

Registers			
Name	Value		
CX	0 0023 h	0000 0000 0010 0011	
	0 00FF h	0000 0000 1111 1111	
CX	0 0023 h	0000 0000 0010 0011	AND

Stack		
Address	Data	
0 0064 h		
0 0062 h	0 2361 h	
0 0060 h	0 0023 h	
0 005E h	RA_Main = 0 0006 h	
0 005C h	Old value of BP	← BP
0 005A h	0 0023 h	
0 0058 h	RA_ProcA = 0 000F h	← SP
0 0056 h		
0 0054 h		
0 0052 h		

ProcB Proc

Push Bp

Mov Bp, Sp

Mov Dx, [Bp + 4]

Or Dx, 0F0F0h

Push Dx

Call ProcC

Registers			
Name	Value		
DX	0 0023 h	0000 0000 0010 0011	
	0 F0F0 h	1111 0000 1111 0000	
DX	0 F0F3 h	1111 0000 1111 0011	OR

Stack		
Address	Data	
0 0064 h		
0 0062 h	0 2361 h	
0 0060 h	0 0023 h	
0 005E h	RA_Main = 0 0006 h	
0 005C h	Old value of BP	
0 005A h	0 0023 h	
0 0058 h	RA_ProcA = 0 000F h	
0 0056 h	0 005C h	← BP
0 0054 h	0 F0F3 h	
0 0052 h	RA_ProcB = 0 0018 h	← SP

ProcC Proc

Push Bp

Mov Bp, Sp

Mov Ax, 5

Mov Bx, [Bp + 4]

LoopC:

Xor Bx, 0F0Fh

Push Bx

Dec Ax

Jnz LoopC

Call ProcD

Registers			
Name	Value		
AX	0 0005 h		
BX	0 0023 h	0000 0000 0010 0011	
	0 0F0F h	0000 1111 0000 1111	
BX	0 0F2C h	0000 1111 0010 1100	XOR
AX	0 0004 h		
BX	0 0F2C h	0000 1111 0010 1100	
	0 0F0F h	0000 1111 0000 1111	
BX	0 0023 h	0000 0000 0010 0011	XOR
AX	0 0003 h		
BX	0 0023 h	0000 0000 0010 0011	
	0 0F0F h	0000 1111 0000 1111	
BX	0 0F2C h	0000 1111 0010 1100	XOR
AX	0 0002 h		
BX	0 0F2C h	0000 1111 0010 1100	
	0 0F0F h	0000 1111 0000 1111	
BX	0 0023 h	0000 0000 0010 0011	XOR
AX	0 0001 h		
BX	0 0023 h	0000 0000 0010 0011	
	0 0F0F h	0000 1111 0000 1111	
BX	0 0F2C h	0000 1111 0010 1100	XOR
AX	0 0000 h		

Stack		
Address	**Data**	
0 0064 h		
0 0062 h	0 2361 h	
0 0060 h	0 0023 h	
0 005E h	RA_Main = 0 0006 h	
0 005C h	Old value of BP	
0 005A h	0 0023 h	
0 0058 h	RA_ProcA = 0 000F h	
0 0056 h	0 005C h	
0 0054 h	0 F0F3 h	
0 0052 h	RA_ProcB = 0 0018 h	
0 0050 h	0 0056 h	← BP
0 004E h	0 0F2C h	
0 004C h	0 0023 h	
0 004A h	0 0F2C h	
0 0048 h	0 0023 h	
0 0046 h	0 0F2C h	
0 0044 h	RA_ProcC = 0 0024 h	← SP
0 0042 h		
0 0040 h		
0 003E h		
0 003C h		
0 003A h		
0 0038 h		
0 0036 h		

ProcD Proc

Push Bp

Mov Bp, Sp

Mov Bx, [Bp + 4]

Not Bx

Push Bx

Registers			
Name	Value		
BX	0 0F2C h	0000 1111 0010 1100	
BX	0 F0D3 h	1111 0000 1101 0011	NOT

Stack	
Address	Data
0 0064 h	
0 0062 h	0 2361 h
0 0060 h	0 0023 h
0 005E h	RA_Main = 0 0006 h
0 005C h	Old value of BP
0 005A h	0 0023 h
0 0058 h	RA_ProcA = 0 000F h
0 0056 h	0 005C h
0 0054 h	0 F0F3 h
0 0052 h	RA_ProcB = 0 0018 h
0 0050 h	0 0056 h
0 004E h	0 0F2C h
0 004C h	0 0023 h

0 004A h	0 0F2C h	
0 0048 h	0 0023 h	
0 0046 h	0 0F2C h	
0 0044 h	RA_ProcC = 0 0024 h	
0 0042 h	0 0050 h	← BP
0 0040 h	0 F0D3 h	← SP
0 003E h		
0 003C h		
0 003A h		
0 0038 h		
0 0036 h		

Add SP, 2

Pop BP

Ret

ProcD EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 2361 h
0 0060 h	0 0023 h
0 005E h	RA_Main = 0 0006 h
0 005C h	Old value of BP
0 005A h	0 0023 h
0 0058 h	RA_ProcA = 0 000F h
0 0056 h	0 005C h
0 0054 h	0 F0F3 h
0 0052 h	RA_ProcB = 0 0018 h

0 0050 h	0 0056 h	← BP
0 004E h	0 0F2C h	
0 004C h	0 0023 h	
0 004A h	0 0F2C h	
0 0048 h	0 0023 h	
0 0046 h	0 0F2C h	← SP
0 0044 h	poped	
0 0042 h	poped	
0 0040 h	poped	
0 003E h		
0 003C h		
0 003A h		
0 0038 h		
0 0036 h		

Add Sp, 10

Pop BP

Ret

ProcC EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 2361 h
0 0060 h	0 0023 h
0 005E h	RA_Main = 0 0006 h
0 005C h	Old value of BP
0 005A h	0 0023 h
0 0058 h	RA_ProcA = 0 000F h

0 0056 h	0 005C h	← BP
0 0054 h	0 F0F3 h	← SP
0 0052 h	poped	
0 0050 h	poped	
0 004E h	poped	
0 004C h	poped	
0 004A h	poped	
0 0048 h	poped	
0 0046 h	poped	
0 0044 h	poped	
0 0042 h	poped	
0 0040 h	poped	
0 003E h		
0 003C h		
0 003A h		
0 0038 h		
0 0036 h		

Add SP, 2

Pop BP

Ret

ProcB EndP

Stack	
Address	**Data**
0 0064 h	
0 0062 h	0 2361 h
0 0060 h	0 0023 h
0 005E h	RA_Main = 0 0006 h

0 005C h	Old value of BP	← BP
0 005A h	0 0023 h	← SP
0 0058 h	poped	
0 0056 h	poped	
0 0054 h	poped	
0 0052 h	poped	
0 0050 h	poped	
0 004E h	poped	
0 004C h	poped	
0 004A h	poped	
0 0048 h	poped	
0 0046 h	poped	
0 0044 h	poped	
0 0042 h	poped	
0 0040 h	poped	
0 003E h		
0 003C h		
0 003A h		
0 0038 h		
0 0036 h		

Add SP, 2

Pop BP

Ret 4

ProcA EndP

Stack	
Address	**Data**

0 0064 h	
0 0062 h	poped
0 0060 h	poped
0 005E h	poped
0 005C h	poped
0 005A h	poped
0 0058 h	poped
0 0056 h	poped
0 0054 h	poped
0 0052 h	poped
0 0050 h	poped
0 004E h	poped
0 004C h	poped
0 004A h	poped
0 0048 h	poped
0 0046 h	poped
0 0044 h	poped
0 0042 h	poped
0 0040 h	poped
0 003E h	
0 003C h	
0 003A h	

← SP