



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Microprocessors and Interfacing

(CSE – 3002)

LAB TASK – 1

Name: **Vibhu Kumar Singh**

Reg. No: **19BCE0215**

Teacher: **Mr. Konguvel E.**

1A: Perform two data transfer function:

DATA 1 to General Purpose Registers

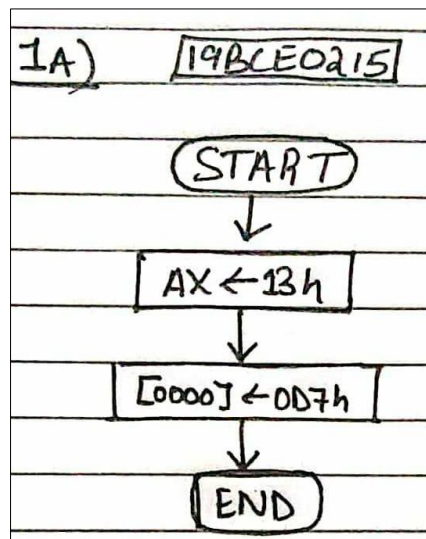
DATA 2 to Any memory location

Ans 1A:

1. Handwritten Assembly Language Program (ALP):

1A)	19BCE0215
1.	MOV AX, 13h
2.	MOV ds: [0000], 0D7h
3.	END

2. Flow Chart:



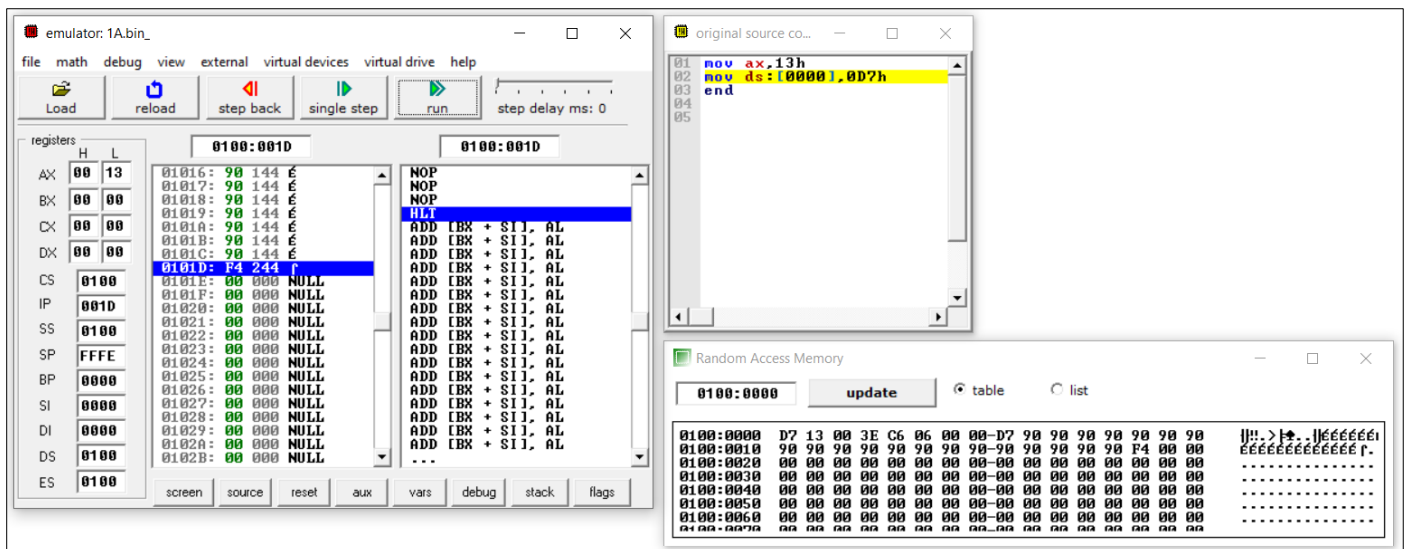
3. Handwritten Calculations for Verification: (N/A)

4. Snapshot of ALP:

```

edit: C:\Users\Vibhu\OneDrive - vit.ac.in\Desktop\Fall Semester 21-22\Micro\ELA\1A.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 mov ax, 13h
02 mov ds: [0000], 0D7h
03 end
  
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.



Reg. No – 19BCE0215

DATA 1 : 19d = 13h

DATA 2 : 0215d = D7h

1B: Perform any two Arithmetic operations using DATA 1 and DATA 2: (ADD/SUB/MUL/DIV)

Store the result in General Purpose Registers

Check the status of Flag register

Ans 1B:

[Addition]

1. Handwritten Assembly Language Program (ALP):

19BLE0215

I. data_new segment

data1 dw 13h

data2 dw 0D7h

data3 dw 1 dup{?}

data_new ends

code segment

assume cs:code, ds:data_new

start:

mov ax, data_new

mov ds, ax

mov ax, data1

mov bx, data2

add ax, bx

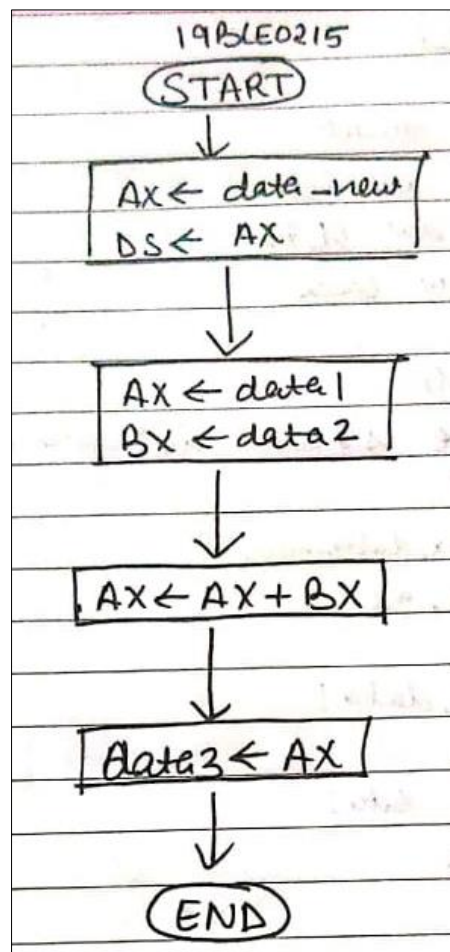
mov data3, ax

code ends

end start

end.

2. Flow Chart:



3. Handwritten Calculations for Verification:

Add
13
+ D7
EA

4. Snapshot of ALP:

```

edit: C:\Users\Vibhu\OneDrive - vit.ac.in\Desktop\Fall Semester 21-22\Micro\ELA\Experiment-1\Addition(16bit).asm
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 data_new segment
02     data1 dw 13h
03     data2 dw 0D7h
04     data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08     assume cs:code,ds:data_new
09     start:
10     mov ax,data_new
11     mov ds,ax
12
13     mov ax,data1
14     mov bx,data2
15     add ax,bx
16     mov data3,ax
17
18     code ends
19 end start
20 end
  
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.

emulator: Addition(16bit).exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

AX	00	EA
BX	00	D7
CX	00	21
DX	00	00
CS	0711	
IP	0025	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

0711:0025

07126:	90	144	E
07127:	90	144	E
07128:	90	144	E
07129:	90	144	E
0712A:	90	144	E
0712B:	90	144	E
0712C:	90	144	E
0712D:	90	144	E
0712E:	90	144	E
0712F:	90	144	E
07130:	90	144	E
07131:	90	144	E
07132:	90	144	E
07133:	90	144	E
07134:	90	144	E
07135:	F4	244	F
07136:	00	000	NULL
07137:	00	000	NULL
07138:	00	000	NULL
07139:	00	000	NULL
0713A:	00	000	NULL
0713B:	00	000	NULL

0711:0025

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

original source co...

01 data_new segment
02 data1 dw 13h
03 data2 dw 0D7h
04 data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08 assume cs:code,ds:data_n
09 start:
10 mov ax,data_new
11 mov ds,ax
12
13 mov ax,data1
14 mov bx,data2
15 add ax,bx
16 mov data3,ax
17
18 code ends
19 end start
20 end

flags

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

analyse

5 | Page

[SUBTRACT]

1. Handwritten Assembly Language Program (ALP):

```
19BLE0215

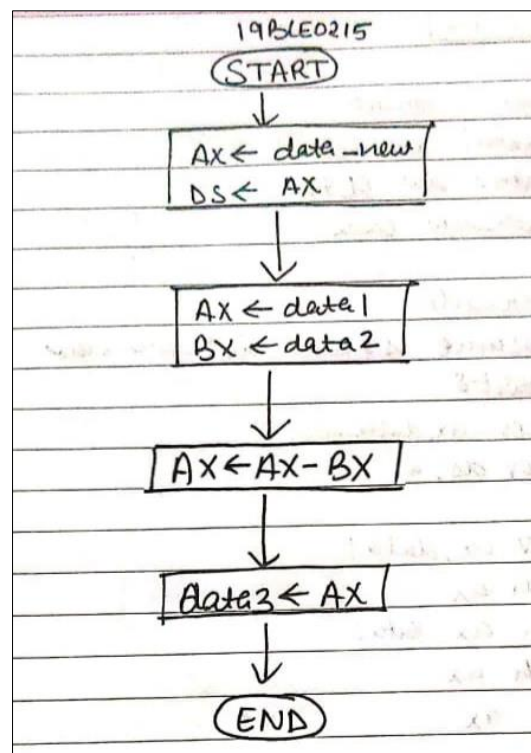
I. data_new segment
    data1 dw 13h
    data2 dw 007h
    data3 dw 1 dup{?}
data_new ends

code segment
    assume cs:code, ds:data_new
start:
    mov ax, data_new
    mov ds, ax

    mov ax, data1
    mov bx, data2
    sub ax, bx
    mov data3, ax

code ends
end start
end.
```

2. Flow Chart:



3. Handwritten Calculations for Verification:

$$\begin{array}{r} \text{Sub} \\ 13 \\ - 07 \\ \hline 3C \end{array}$$

4. Snapshot of ALP:

```
edit: C:\Users\Vibhu\OneDrive - vit.ac.in\Desktop\Fall Semester 21-22\Micro\ELA\Experiment-1\Subtraction(16bit).asm
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 data_new segment
02     data1 dw 13h
03     data2 dw 0D7h
04     data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08     assume cs:code,ds:data_new
09 start:
10     mov ax,data_new
11     mov ds,ax
12
13     mov ax,data1
14     mov bx,data2
15     sub ax,bx
16     mov data3,ax
17
18 code ends
19 end start
20 end
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.

The screenshot displays the emulator interface for 'Subtraction(16bit).exe'. The 'registers' window shows the following values:

Register	H	L
AX	FF	3C
BX	00	07
CX	00	21
DX	00	00
CS	0711	
IP	0025	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

The 'memory' window shows the following values:

Address	Value
07126	90 144 E
07127	90 144 E
07128	90 144 E
07129	90 144 E
0712A	90 144 E
0712B	90 144 E
0712C	90 144 E
0712D	90 144 E
0712E	90 144 E
0712F	90 144 E
07130	90 144 E
07131	90 144 E
07132	90 144 E
07133	90 144 E
07134	90 144 E
07135	04 244 0
07136	00 000 NULL
07137	00 000 NULL
07138	00 000 NULL
07139	00 000 NULL
0713A	00 000 NULL
0713B	00 000 NULL

The 'flags' window shows the following status:

Flag	Status
CF	1
ZF	0
SF	1
OF	0
PF	1
AF	1
IF	1
DF	0

**1C: Perform any two Logical operations using DATA 1 and DATA 2:
(AND/OR/XOR)**

Store the result in General Purpose Registers

Check the status of Flag register

Ans 1C:**1. Handwritten Assembly Language Program (ALP):**

```

19BCE0215

I. data_new segment
    data1 dw 13h
    data2 dw 0D7h
    data3 dw 1 dup{?}
data_new ends

code segment
    assume cs:code, ds:data_new
    start:
    mov ax, data_new
    mov ds, ax

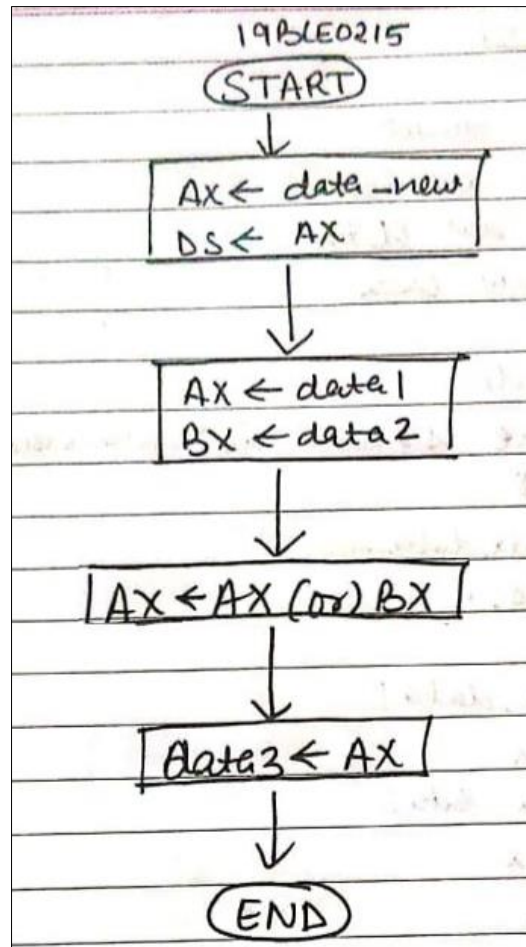
    mov ax, data1
    mov bx, data2
    or ax, bx
    mov data3, ax

    code ends

end start
end.

```


2. Flow Chart:

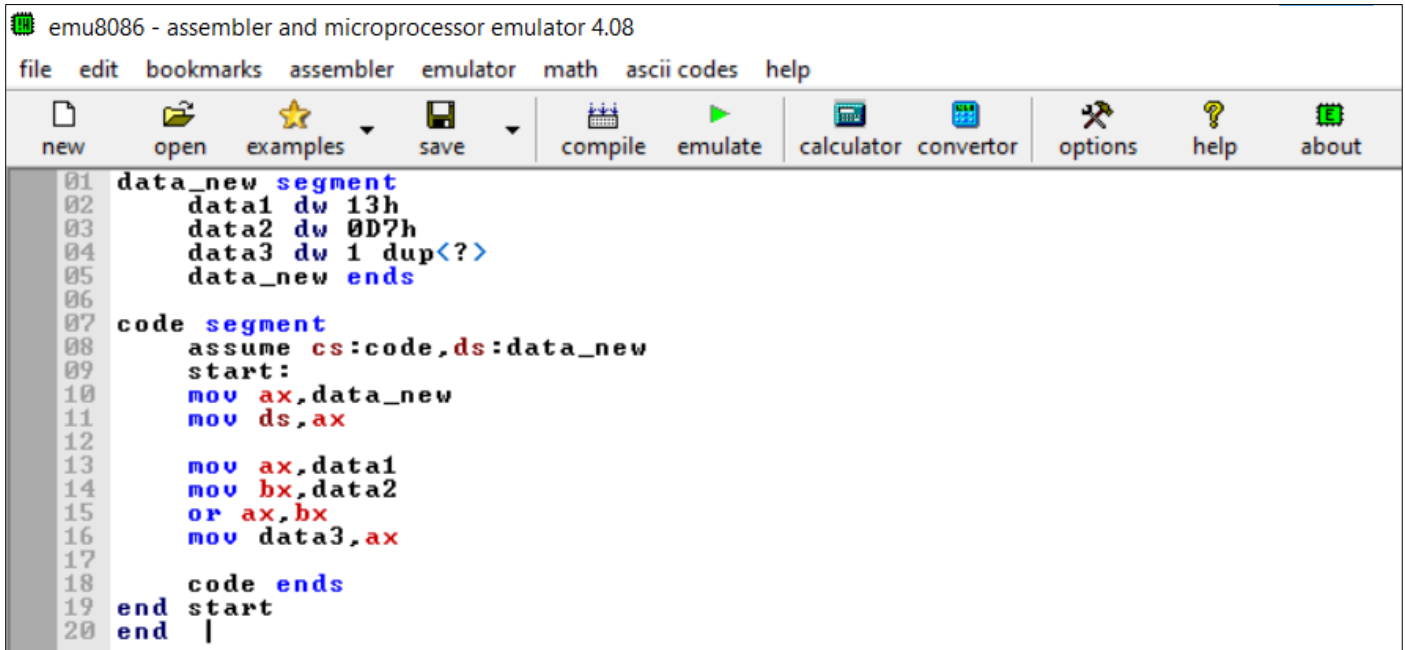


3. Handwritten Calculations for Verification:

Q9

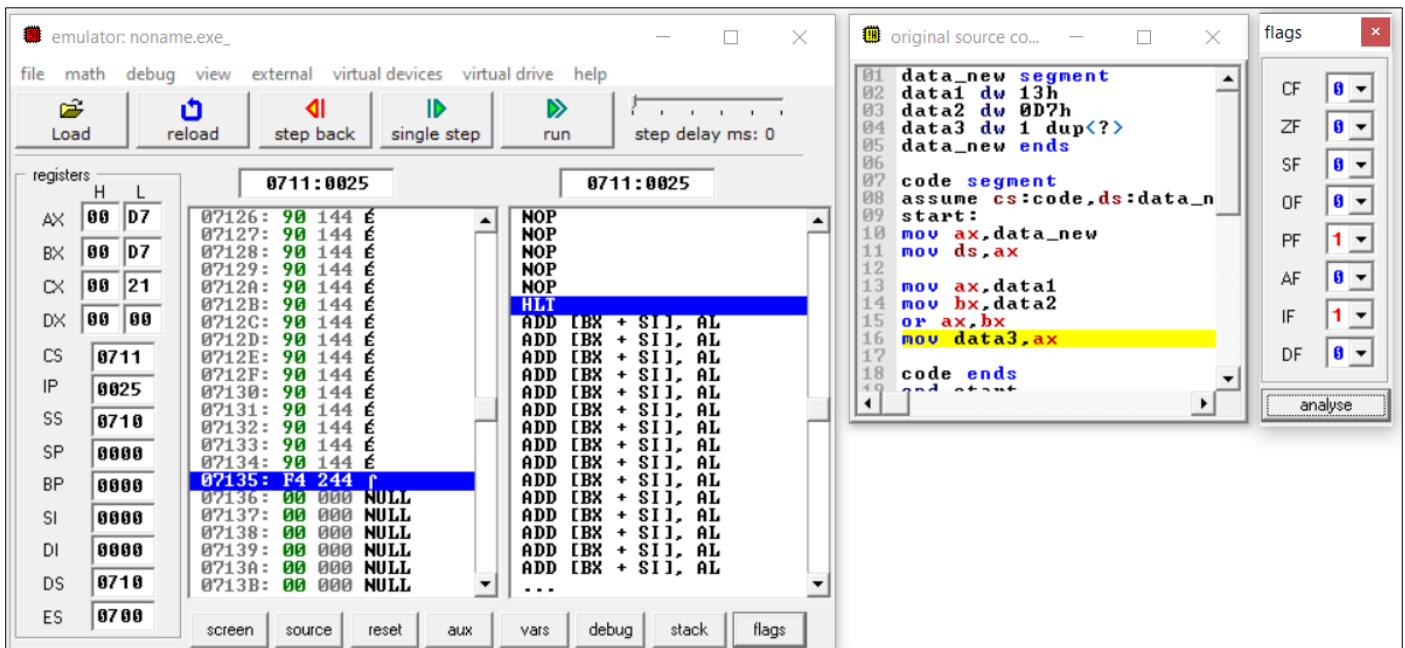
$$13h = (10011)_2$$
$$D7h = (11010111)_2$$
$$\begin{array}{r} 00010011 \\ \text{(or)} \quad 11010111 \\ \hline 11010111 \end{array}$$
$$(11010111)_2 = \boxed{D7h}$$

4. Snapshot of ALP:



```
01 data_new segment
02     data1 dw 13h
03     data2 dw 0D7h
04     data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08     assume cs:code,ds:data_new
09     start:
10     mov ax,data_new
11     mov ds,ax
12
13     mov ax,data1
14     mov bx,data2
15     or ax,bx
16     mov data3,ax
17
18     code ends
19 end start
20 end |
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.



The screenshot displays the emu8086 emulator interface with the following components:

- Registers Panel:** Shows the state of various registers. The CS register is highlighted at 0711, and the IP register is at 0025. The stack pointer (SP) is at 0000.
- Memory Panel:** Displays memory addresses and their contents. The address 0711:0025 is selected, showing a value of 00 00 00.
- Source Code Panel:** Shows the assembly code being executed. The instruction `mov data3,ax` is highlighted in yellow.
- Flags Panel:** Shows the status of various flags. The CF (Carry Flag) is 0, ZF (Zero Flag) is 0, SF (Sign Flag) is 0, OF (Overflow Flag) is 0, PF (Parity Flag) is 1, AF (Auxiliary Flag) is 0, IF (Interrupt Flag) is 1, and DF (Direction Flag) is 0.

[AND]

1. Handwritten Assembly Language Program (ALP):

```
19BLE0215

I. data_new segment
    data1 dw 13h
    data2 dw 0D7h
    data3 dw 1 dup{?}
data_new ends

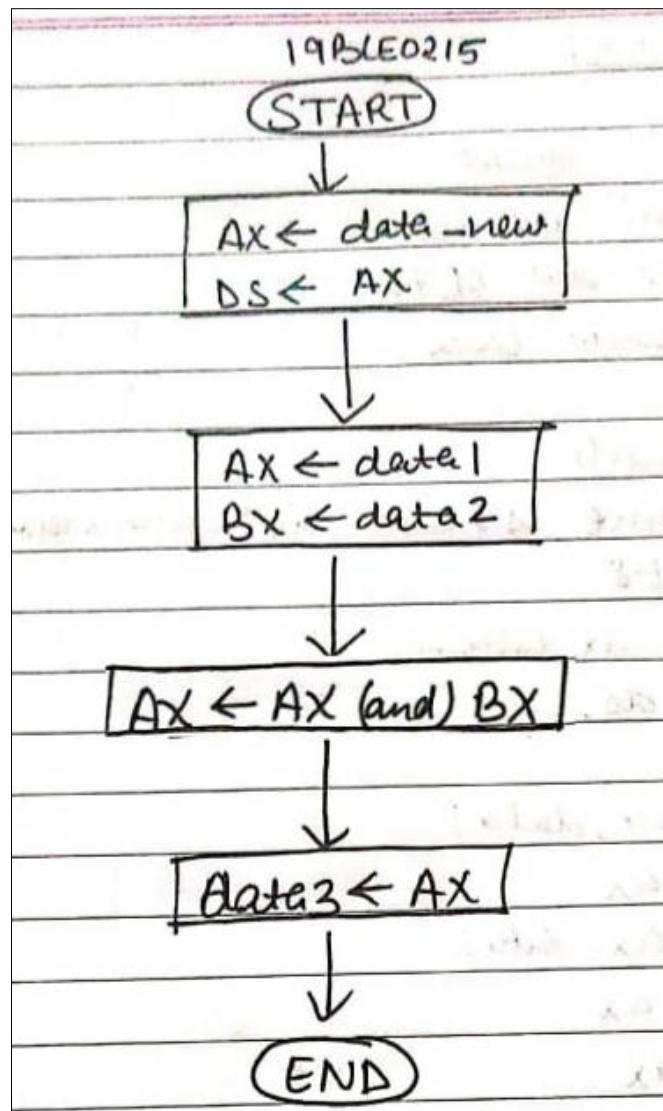
code segment
    assume cs: code, ds: data_new
start:
    mov ax, data_new
    mov ds, ax

    mov ax, data1
    mov bx, data2
    and ax, bx
    mov data3, ax

code ends

end start
end.
```

2. Flow Chart:



3. Handwritten Calculations for Verification:

And

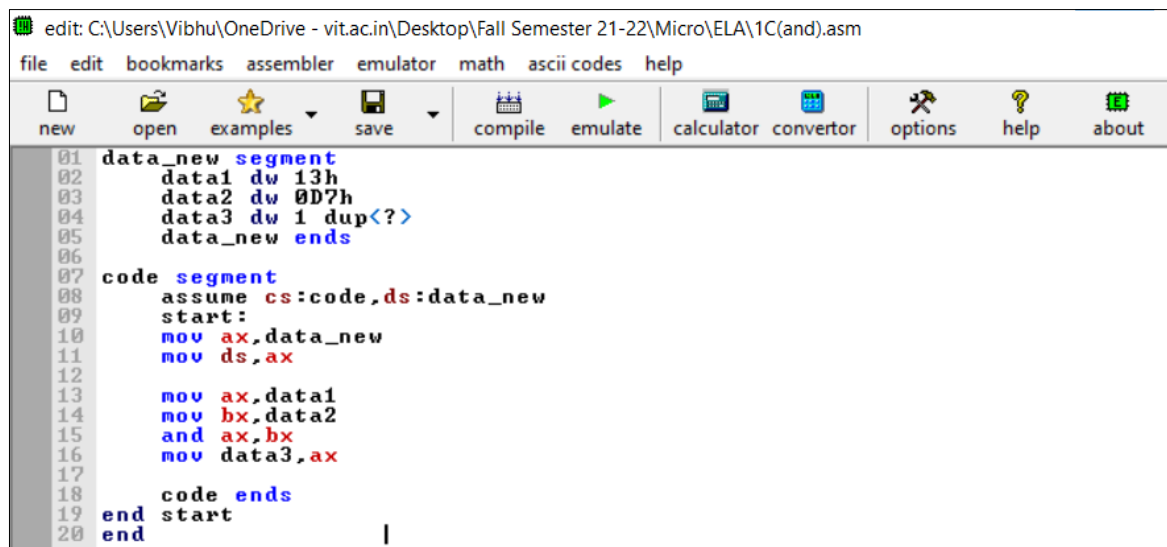
$$13h = (10011)_2$$

$$D7h = (11010111)_2$$

$$\begin{array}{r}
 00010011 \\
 \text{(and)} \quad 11010111 \\
 \hline
 00010011
 \end{array}$$

$$(00010011)_2 = 13h$$

4. Snapshot of ALP:

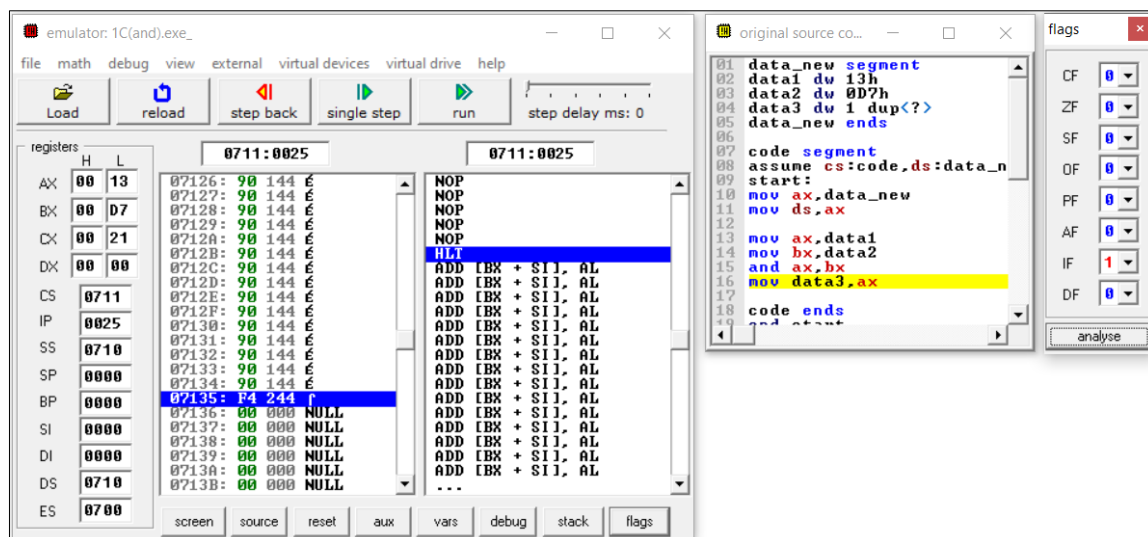


```
edit: C:\Users\Vibhu\OneDrive - vit.ac.in\Desktop\Fall Semester 21-22\Micro\ELA\1C(AND).asm
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 data_new segment
02     data1 dw 13h
03     data2 dw 0D7h
04     data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08     assume cs:code,ds:data_new
09 start:
10     mov ax,data_new
11     mov ds,ax
12
13     mov ax,data1
14     mov bx,data2
15     and ax,bx
16     mov data3,ax
17
18 code ends
19 end start
20 end
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.



emulator: 1C(AND).exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers H L

Register	H	L
AX	00	13
BX	00	D7
CX	00	21
DX	00	00
CS	0711	
IP	0025	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

0711:0025

Address	Hex	ASCII	Instruction
07126	90 144 E		NOP
07127	90 144 E		NOP
07128	90 144 E		NOP
07129	90 144 E		NOP
0712A	90 144 E		NOP
0712B	90 144 E		NOP
0712C	90 144 E		ADD [BX + SI], AL
0712D	90 144 E		ADD [BX + SI], AL
0712E	90 144 E		ADD [BX + SI], AL
0712F	90 144 E		ADD [BX + SI], AL
07130	90 144 E		ADD [BX + SI], AL
07131	90 144 E		ADD [BX + SI], AL
07132	90 144 E		ADD [BX + SI], AL
07133	90 144 E		ADD [BX + SI], AL
07134	90 144 E		ADD [BX + SI], AL
07135	F4 244 F		HLT
07136	00 000 NULL		ADD [BX + SI], AL
07137	00 000 NULL		ADD [BX + SI], AL
07138	00 000 NULL		ADD [BX + SI], AL
07139	00 000 NULL		ADD [BX + SI], AL
0713A	00 000 NULL		ADD [BX + SI], AL
0713B	00 000 NULL		...

original source co...

```
01 data_new segment
02     data1 dw 13h
03     data2 dw 0D7h
04     data3 dw 1 dup<?>
05 data_new ends
06
07 code segment
08     assume cs:code,ds:data_n
09 start:
10     mov ax,data_new
11     mov ds,ax
12
13     mov ax,data1
14     mov bx,data2
15     and ax,bx
16     mov data3,ax
17
18 code ends
19 end start
```

flags

Flag	Value
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

analyse

1D: Push the DATA 1, DATA 2 to Stack and Pop it from Stack.

Ans 1D:

1. Handwritten Assembly Language Program (ALP):

```

1D) [19BCE0215]

data_new segment
    data1 dw 13h
    data2 dw 0D7h
data_new ends

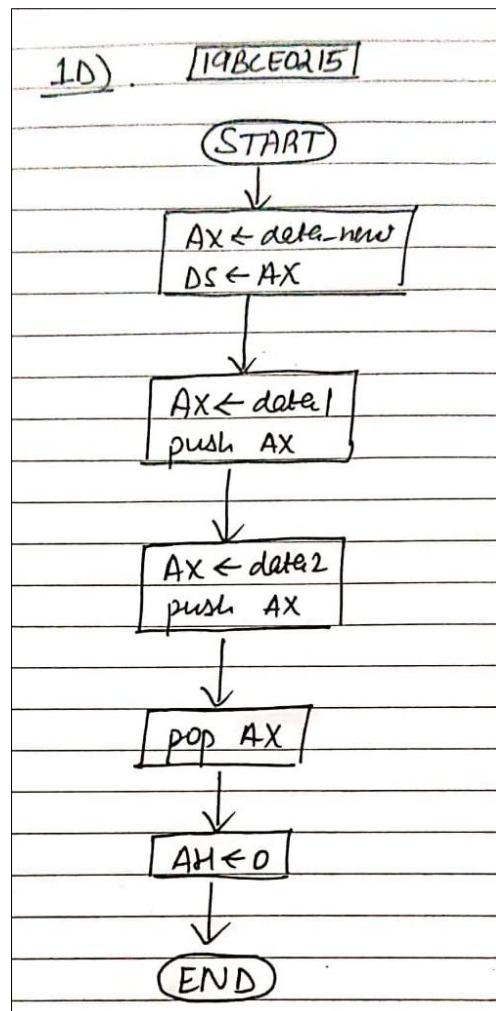
code segment
    assume cs:code, ds:data_new
    start:
        mov ax, data_new
        mov ds, ax

        mov ax, data1
        push ax
        mov ax, data2
        push ax
        pop ax

        mov ah, 0
    end start
code ends
end.

```

2. Flow Chart:



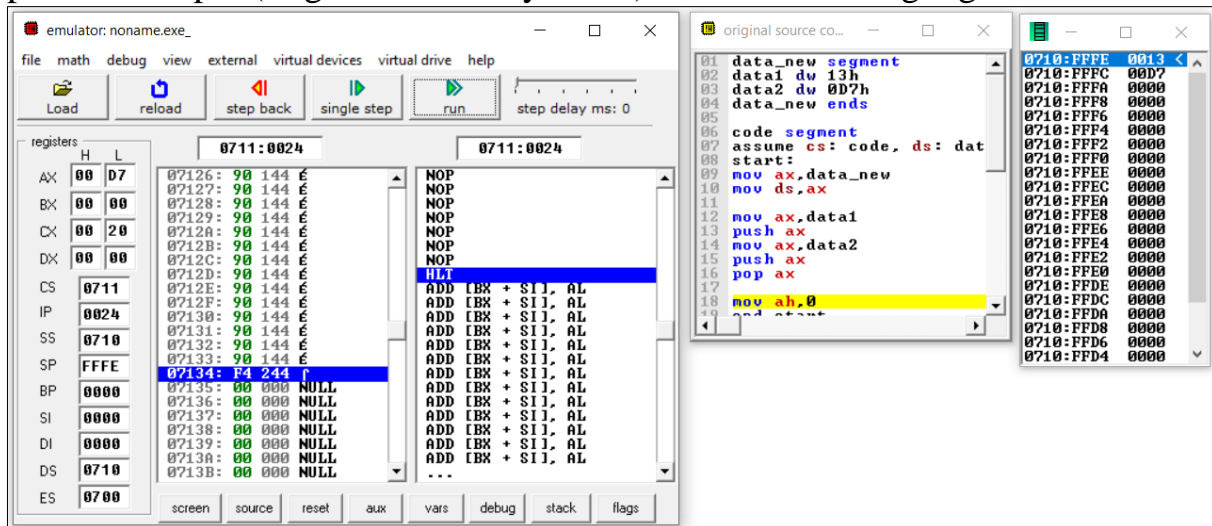
3. Handwritten Calculations for Verification: (N/A)

4. Snapshot of ALP:

```

emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 data_new segment
02 data1 dw 13h
03 data2 dw 0D7h
04 data_new ends
05
06 code segment
07 assume cs: code, ds: data_new
08 start:
09 mov ax, data_new
10 mov ds, ax
11
12 mov ax, data1
13 push ax
14 mov ax, data2
15 push ax
16 pop ax
17
18 mov ah, 0
19 end start
20 code ends
21 ret
  
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.



Reg. No – 19BCE0215

DATA 1 : 19d = 13h

DATA 2 : 0215d = D7h

1E: Perform SHR, SAR, ROL, RCL Bit manipulation operations using DATA 2.

Store the result in General Purpose Registers

Check the status of Flag register

Ans 1E:

1. Handwritten Assembly Language Program (ALP):

```

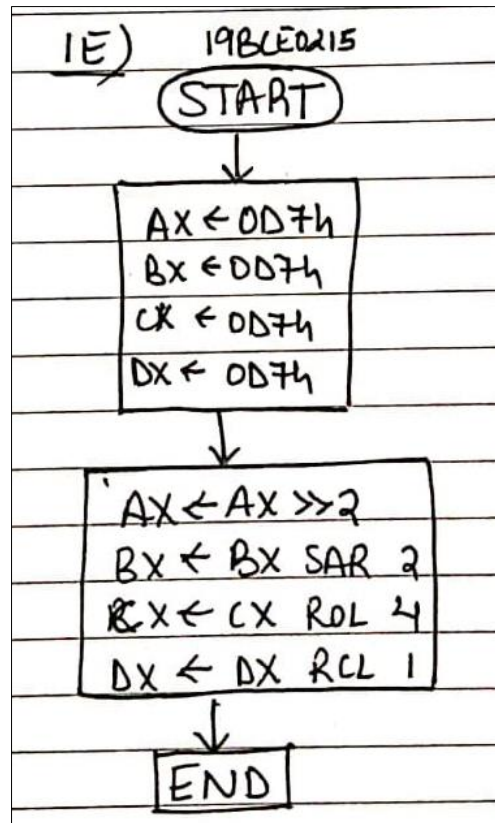
1E) [19BLE0215]

MOV AX, 0D7h
MOV BX, 0D7h
MOV CX, 0D7h
MOV DX, 0D7h

SHR AX, 2
SAR BX, 2
ROL CX, 4
RCL DX, 1

HLT
    
```

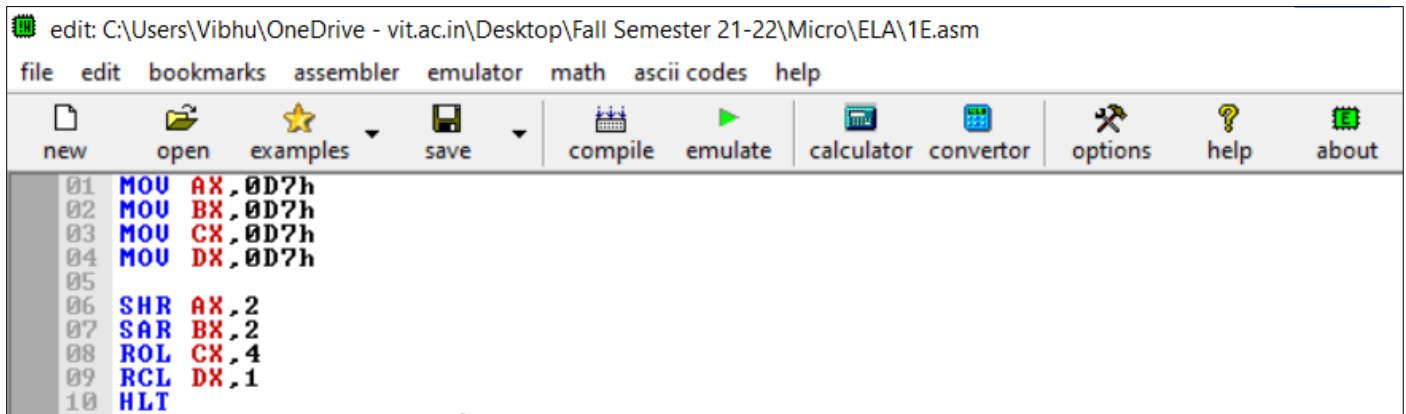
2. Flow Chart:



3. Handwritten Calculations for Verification:

1E)	19BCE0215
SHR, 2	ASR, 2
0D7h = (11010111) ₂	0D7h = (11010111) ₂
(11010111) >> 2	(11010111) >> 2
= (110101) ₂ = 35h	= (110101) ₂ = 35h
ROL, 4	RCL, 1
0D7h = (11010111) ₂	0D7h = (11010111) ₂
(11010111) ROL 4	(11010111) RCL 1
= (110101110000) ₂ = D70h	= (110101110) ₂ = 1AEh

4. Snapshot of ALP:



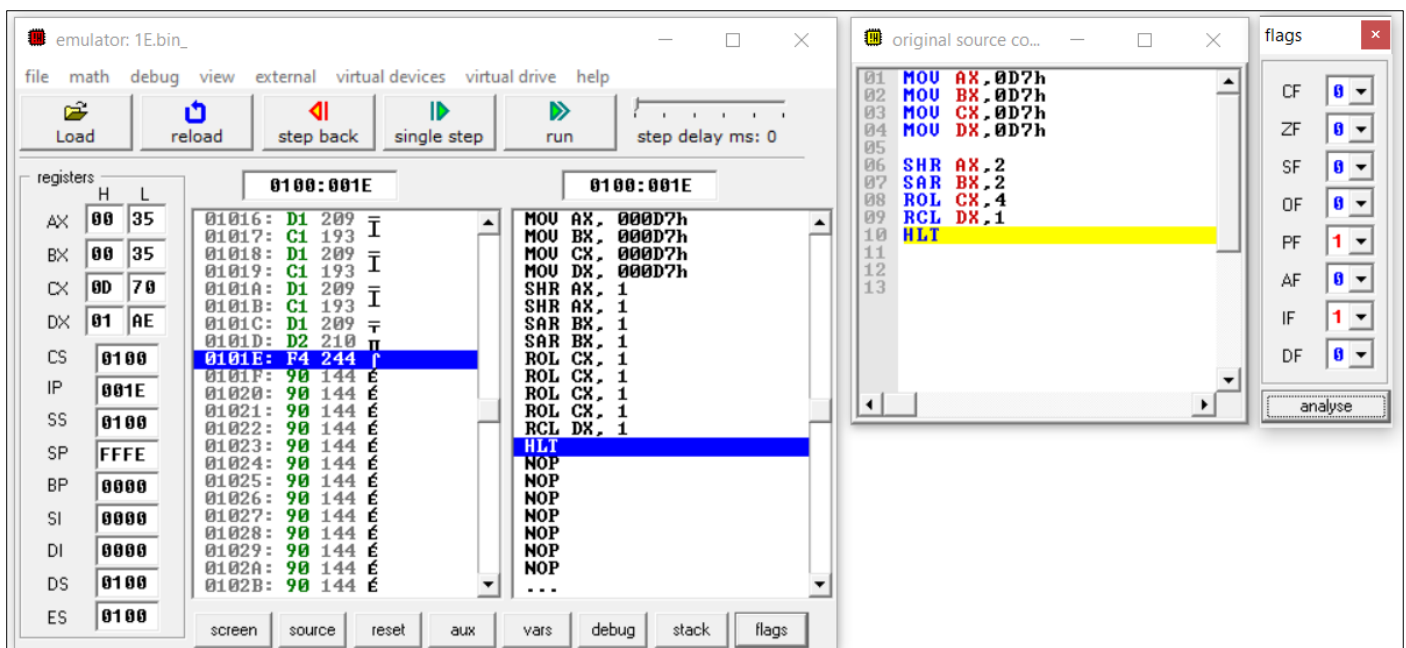
edit: C:\Users\Vibhu\OneDrive - vit.ac.in\Desktop\Fall Semester 21-22\Micro\ELA\1E.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
01 MOV AX,0D7h
02 MOV BX,0D7h
03 MOV CX,0D7h
04 MOV DX,0D7h
05
06 SHR AX,2
07 SAR BX,2
08 ROL CX,4
09 RCL DX,1
10 HLT
```

5. Snapshot of Output (Registers/Memory/Stack) and status of Flag registers.



emulator: 1E.bin_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	35
BX	00	35
CX	00	70
DX	01	AE
CS	0100	
IP	001E	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:001E

Address	Hex	ASCII
01016:	D1 209	I
01017:	C1 193	I
01018:	D1 209	I
01019:	C1 193	I
0101A:	D1 209	I
0101B:	C1 193	I
0101C:	D1 209	I
0101D:	D2 210	I
0101E:	F4 244	I
0101F:	90 144	E
01020:	90 144	E
01021:	90 144	E
01022:	90 144	E
01023:	90 144	E
01024:	90 144	E
01025:	90 144	E
01026:	90 144	E
01027:	90 144	E
01028:	90 144	E
01029:	90 144	E
0102A:	90 144	E
0102B:	90 144	E

0100:001E

```
MOV AX,000D7h
MOV BX,000D7h
MOV CX,000D7h
MOV DX,000D7h
SHR AX,1
SAR BX,1
ROL CX,1
RCL DX,1
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

original source co...

```
01 MOV AX,0D7h
02 MOV BX,0D7h
03 MOV CX,0D7h
04 MOV DX,0D7h
05
06 SHR AX,2
07 SAR BX,2
08 ROL CX,4
09 RCL DX,1
10 HLT
11
12
13
```

flags

CF	0
ZF	0
SF	0
OF	0
PF	1
AF	0
IF	1
DF	0

analyse

screen source reset aux vars debug stack flags