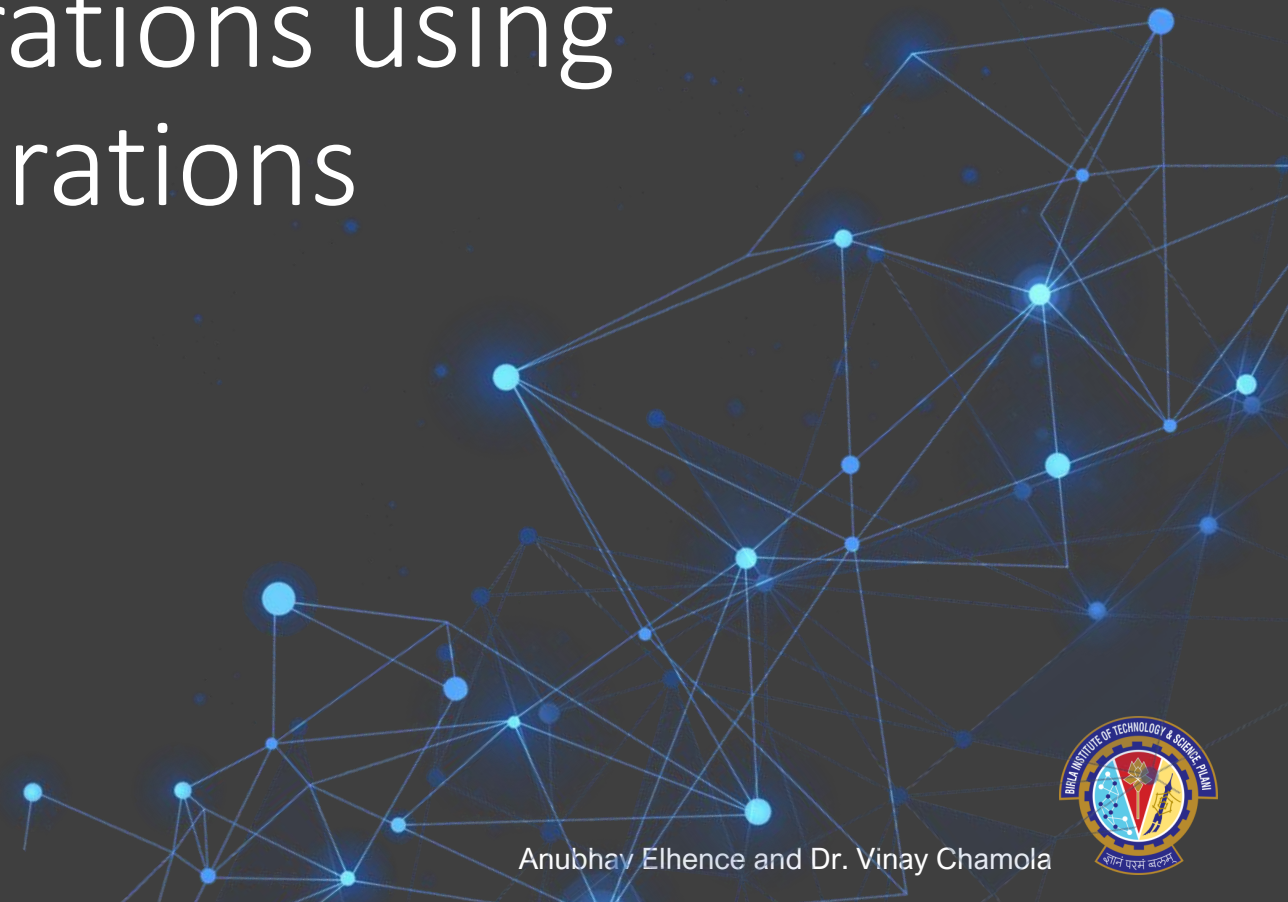
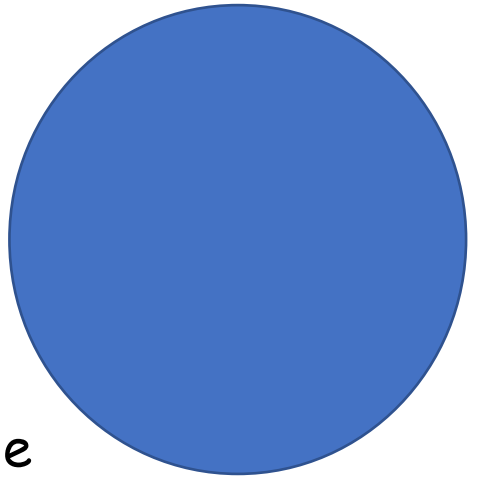
CS/EEE/INSTR F241
Lab 7 – Advanced Operations using Interrupts and File Operations

Anubhav Elhence




Quick Revision

DOS File handle Functions



- In DOS, a file handle is a unique identifier used to access an open file. The DOS file handle functions are a set of interrupts that allow programs to create, open, read from, write to, and close files using file handles. Here are some of the most commonly used DOS file handle functions:

- INT 21h, AH=3Ch: Create File

  This interrupt is used to create a new file and returns a file handle that can be used to access the file.

- INT 21h, AH=3Dh: Open File

  This interrupt is used to open an existing file and returns a file handle that can be used to access the file.

- INT 21h, AH=3Fh: Read From File

  This interrupt is used to read data from an open file using a specified file handle.

- INT 21h, AH=40h: Write To File

  This interrupt is used to write data to an open file using a specified file handle.

- INT 21h, AH=3Eh: Close File

  This interrupt is used to close an open file using a specified file handle.

- INT 21h, AH=44h: Get File Information

  This interrupt is used to retrieve information about a file using a specified file handle.

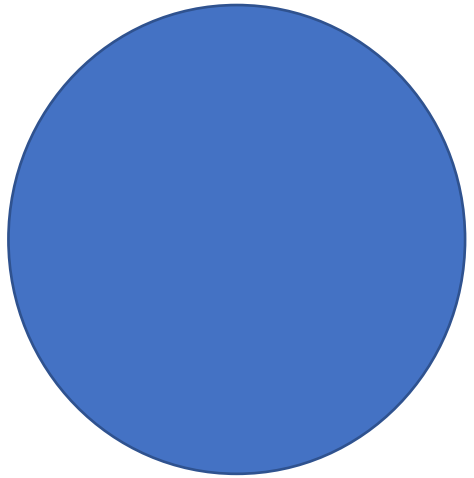
- INT 21h, AH=4Eh: Find First File

  This interrupt is used to search for the first file that matches a specified file pattern and returns a file handle that can be used to access the file.

Anubhav Elhence and Dr. Vinay Chamola

Follow Along example:

- Creating a random file



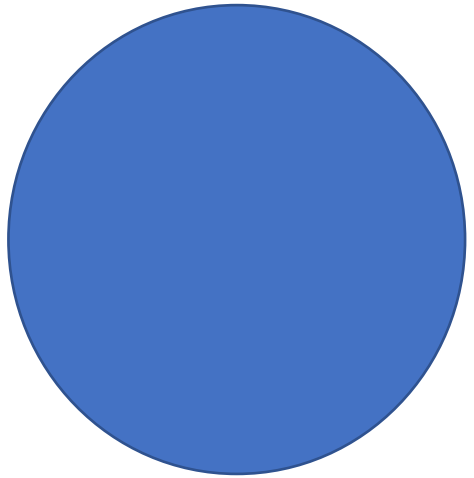
week6_c1.asm > ...

```
1    .model tiny
2    .data
     4 references
3    fname   db 'testing',0
     6 references
4    handle  dw ?
5    .code
6    .startup
7        mov ah, 3ch
8        lea dx, fname
9        mov cl, 1h
10       int 21h
11       mov handle, ax
12   .exit
     2 references
13   end
```


Anubhav Elhence and Dr. Vinay Chamola

Follow Along example 2:

- Writing into a file



week6_c2.asm > ...

```
1    .model tiny
2    .data
     4 references
3    fname   db 'second.txt',0
     6 references
4    handle  dw ?
     3 references
5    msg     db 'MuP rocks!'
6    .code
7    .startup
8
9        ; Create a file if it
10       ; is not existing
11       mov ah, 3ch
12       lea dx, fname
13       mov cl, 1h
14       int 21h
15       mov handle, ax
16
17       ; open file
18       mov ah, 3dh
19       mov al, 1h
20       lea dx, fname
21       int 21h
22       mov handle, ax
23
24       ; write msg to file
25       mov ah, 40h
26       mov bx, handle
27       mov cx, 10
28       lea dx, msg
29       int 21h
30
31       ; close file
32       mov ah, 3eh
33       int 21h
34   .exit
     2 references
35   end
```


Anubhav Elhence and Dr. Vinay Chamola

Follow Along example 3:

- Reading from a file

week6_c3.asm > ...

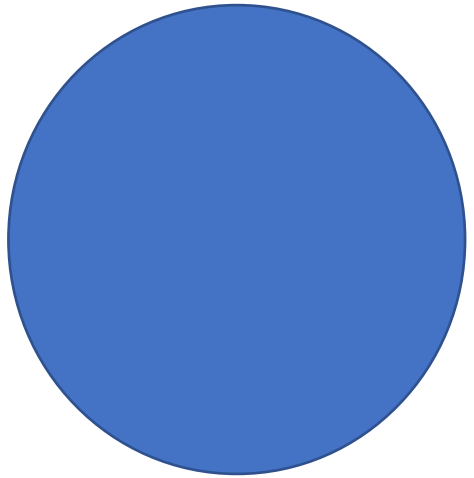
```
1   .model tiny
2   .data
    4 references
3   fname   db 'USER.txt', 0
    6 references
4   handle  dw ?
    3 references
5   msg db 20 dup('$')
6   .code
7   .startup
8       ; open file
9       mov ah, 3dh
10      mov al, 0h
11      lea dx, fname
12      int 21h
13      mov handle, ax
14
15      ; read content into msg
16      mov ah, 3fh
17      mov bx, handle
18      mov cx, 10
19      lea dx, msg
20      int 21h
21
22      ; print msg
23      lea dx, msg
24      mov ah, 09h
25      int 21h
26
27      ; close file
28      mov ah, 3eh
29      int 21h
30  .exit
    2 references
31  end
```


Anubhav Elhence and Dr. Vinay Chamola

Input a string from keyboard (STDIN)

week5_c1.asm > ...

```
1    .model tiny
2    .486
3    .data
     1 reference
4    max1 db 32
     0 references
5    act1 db ?
     0 references
6    inp1 db 32 dup(0)
7    .code
8    .startup
9
10       lea DX, max1
11       mov ah, 0ah
12       int 21h
13
14   .exit
     0 references
15   end
```



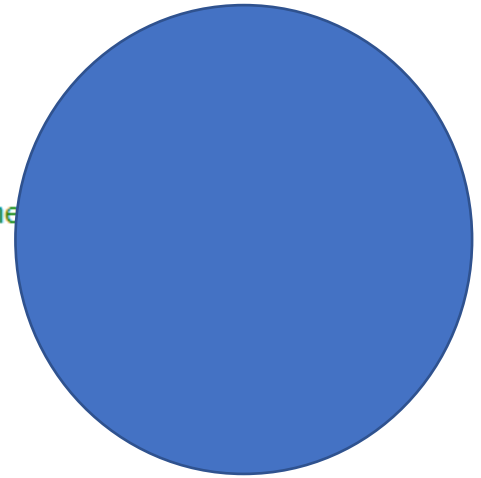
- After the interrupt, act 1 will contain the number of characters read, and the characters themselves will start at inp1 The characters will be terminated by a carriage return (ASCII code 0Dh), although this will not be included in the count (Note: this will not be included in the ACT1 but you have to count Enter also when you are specifying it in max1)

Let's look at a more complicated Example

- Write an ALP that does the following:
  - (1) Display the string "Enter 10 character long User Name" and goes to the next line
  - (2) Takes in the user-entered string of 10 characters and compares with the user name value already stored in memory
  - (3) If there is no match it should exit saying "wrong Username"
  - (4) If there is a match it should display the string "Enter 5 character long Password" and goes to the next line
  - (5) Takes in the password entered by the user and compares it with the password already stored in memory
  - (6) If there is no match it should exit'
  - (7) If there is a match it should display "Hello <Username>" where <Username> is replaced by the actual username of the person.

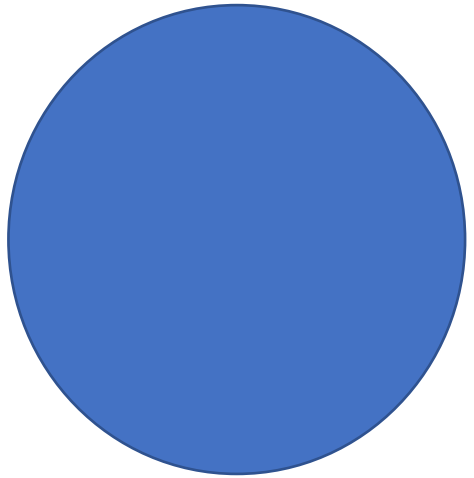
```
-g 01af
enter 10 character long User Name:
anub@g.com
enter 5 character long password:
******
hello anub@g.com
AX=092A BX=0000 CX=0000 DX=0272 SP=FFFE BP=0000 SI=0230 DI=0235
DS=0863 ES=0863 SS=0863 CS=0863 IP=01AF NV UP EI PL NZ NA PO NC
0863:01AF 8D1E0902          LEA     BX,[0209]                   DS:0209=6E65
```

Making the right Data initializations



```
        3 references
8   msg1 db "enter 10 character long User Name: $" ; Message 1: Prompt to enter the username
        2 references
9   usn1 db "anub@g.com$" ; Correct username for comparison
        3 references
10  max1 db 20 ; Maximum length for input
        2 references
11  act1 db ? ; Placeholder for action
        3 references
12  inp1 db 20 dup("$") ; Buffer to store user's input for username
13
        3 references
14  msg2 db "enter 5 character long password:$" ; Message 2: Prompt to enter the password
        1 reference
15  pass1 db "oscar" ; Correct password for comparison
        2 references
16  inp2 db 30 dup("$") ; Buffer to store user's input for password
        1 reference
17  msg3 db "hello $" ; Message 3: Greeting message when both inputs are correct
        1 reference
18  msg4 db "wrong username$" ; Message 4: Wrong username input
        1 reference
19  msg5 db "wrong password$" ; Message 5: Wrong password input
        7 references
20  nline db 0ah, 0dh, "$" ; New line characters
21
```

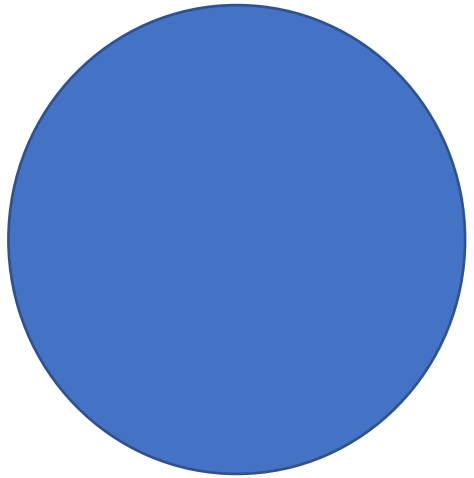
Let's display the first msg on the screen



```
25      ; Display message 1 on the screen and go to the next line.
26
27      lea dx, msg1
28      mov ah, 09h
29      int 21h
30
31      ; Add a new line after the message.
32
33      lea dx, nline
34      mov ah, 09h
35      int 21h
36
```


Anubhav Elhence and Dr. Vinay Chamola

- Time for taking input from the user



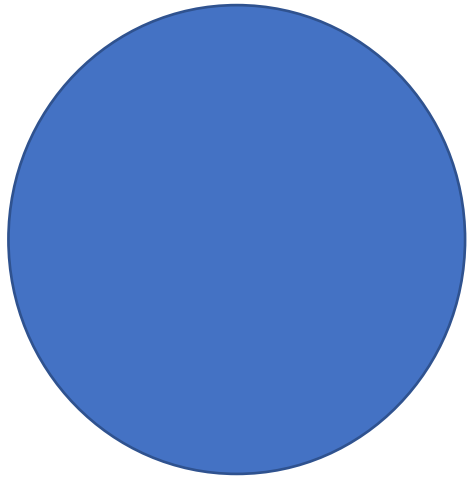
```
37      ; Take input from the user and store it in inp1.
38
39      lea dx, max1
40      mov ah, 0ah
41      int 21h
```

- Time for doing a conditional check… let's compare the entered username with the stored username

```
43      ; Compare the entered username with the stored username.
44
45      cld
46      lea di, inp1
47      lea si, usn1
48      mov cx, 11
49      repe cmpsb
50      jcxz l1
51
```

- If the username is wrong... Then...

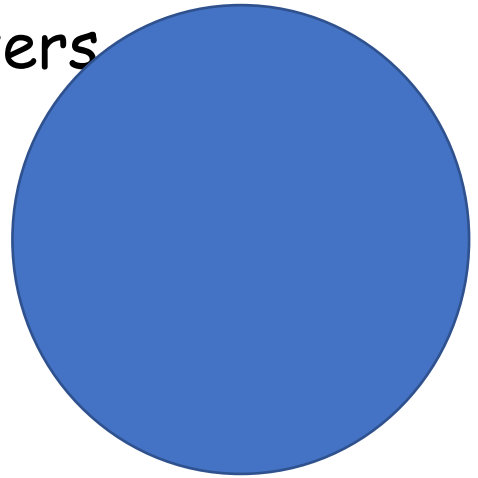
```
52      ; If the username is incorrect, display the "wrong username" message and exit.
53
54      lea dx, nline
55      mov ah, 09h
56      int 21h
57
58      lea dx, msg4
59      mov ah, 09h
60      int 21h
61
62      mov ah, 4ch
63      int 21h
64
```



- If the username is correct... Then...

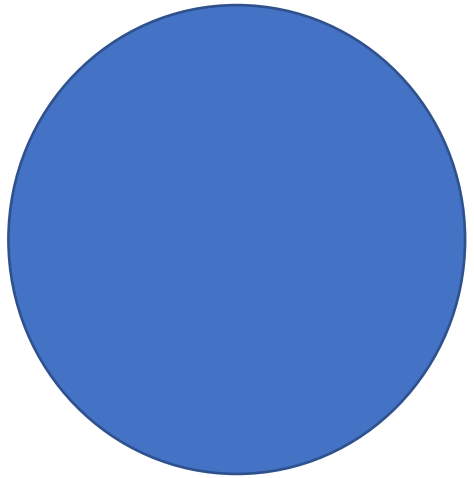
```
65      ; If the username is correct, display the "enter password" message.
66
67      l1:
68      lea dx, nline
69      mov ah, 09h
70      int 21h
71
72      lea dx, msg2
73      mov ah, 09h
74      int 21h
75
76      lea dx, nline
77      mov ah, 09h
78      int 21h
```

- Taking password input from user, while masking the characters



```
80      ; Take password input from the user, masking the characters.
81
82      mov cx, 6
83      lea di, inp2
84      l2:
85      mov ah, 08h
86      int 21h
87      mov [di], al
88      mov dl, '*'
89      mov ah, 02h
90      int 21h
91      inc di
92      dec cx
93      jnz l2
```

- Compare the entered password with the stored password



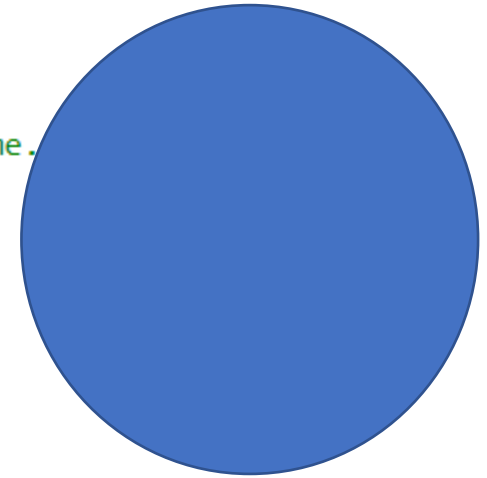
```
; Compare the entered password with the stored password.

cld
mov cx, 6
lea di, inp2
lea si, pass1
repe cmpsb
jcxz l3
```

- If the password is incorrect... Then...

```
104     ; If the password is incorrect, display the "wrong password" message and exit.
105
106     lea dx, nline
107     mov ah, 09h
108     int 21h
109
110     lea dx, msg5
111     mov ah, 09h
112     int 21h
113
114     mov ah, 4ch
115     int 21h
116
```

- If the password is correct... Then...

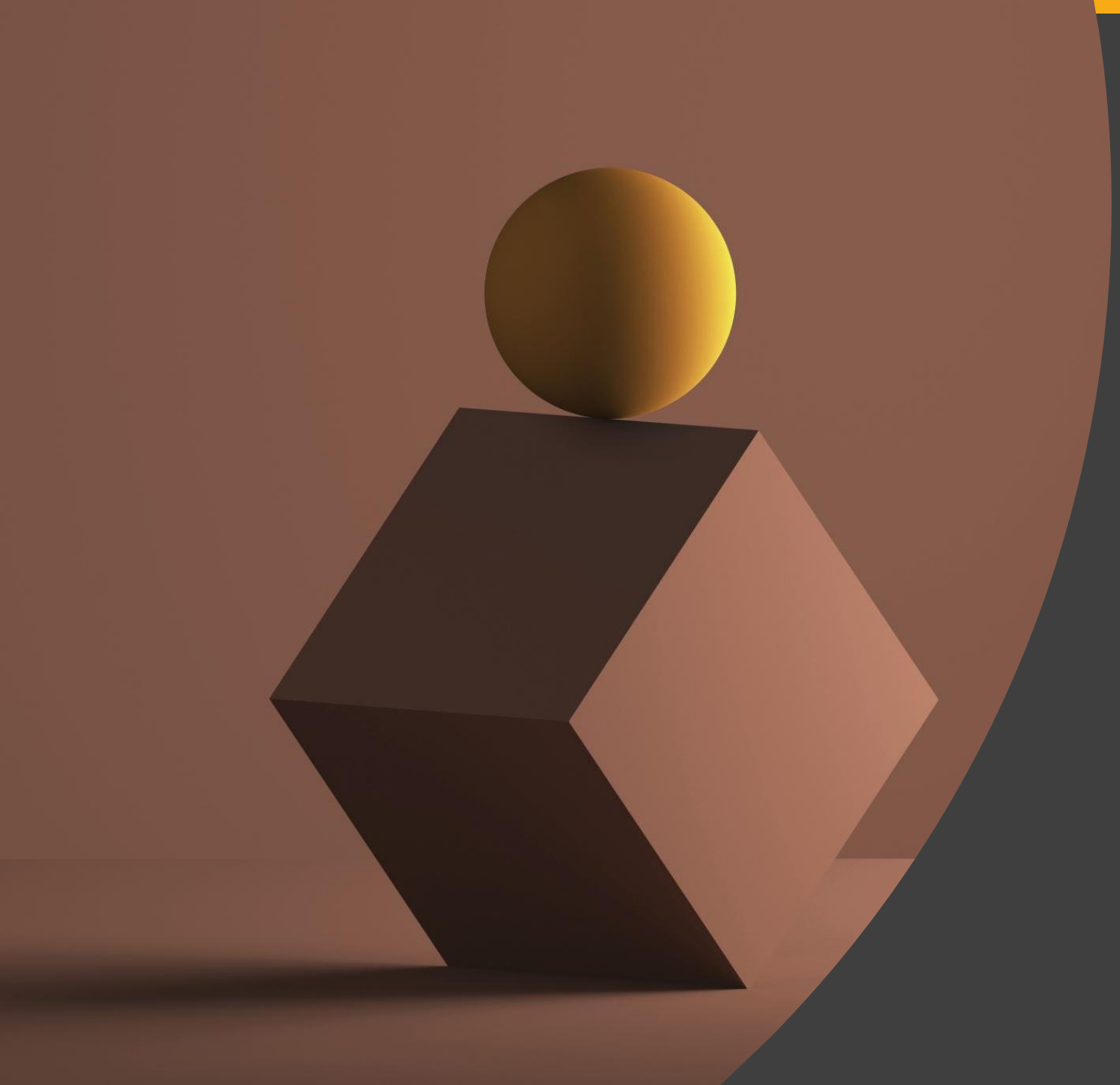


```
117     ; If the password is correct, display the greeting message and the username.
118
119     l3:
120     lea dx, nline
121     mov ah, 09h
122     int 21h
123
124     lea dx, msg3
125     mov ah, 09h
126     int 21h
127
128     lea dx, usn1
129     mov ah,09h
130     int 21h
131
132     lea dx, nline
133     mov ah, 09h
134     int 21h
135
```

```
145     .exit
        3 references
146     end
147
```

Time for Lab Tasks:

Please check the description of this video.



Thankyou

Anubhav Elhence and Dr. Vinay Chamola