```
.MODEL SMALL
.STACK 100H
.DATA
    ; DEFINE YOUR VARIABLES
    ; The .data section is where you define and allocate memory for
    ; your program's data variables. This section typically contains
    ; declarations for variables, constants, and strings that your program uses.
    flag_array db 0,0,0,0,0

    msg db 0ah,0dh, "Enter the password(8-20 length): $"
    invalid_msg db 0ah,0dh, "Invalid Password!$"
    strong_msg db 0ah,0dh, "Strong password!$"
    medium_msg db 0ah,0dh, "Medium password!$"
    weak_msg db 0ah,0dh, "Weak password!$"

    password db 20 DUP<?>

.CODE
    ; The .code section is where you place your program's executable
    ; instructions or code. It contains the actual assembly language
    ; instructions that perform tasks and computations.

    MAIN PROC

        MOV AX, @DATA
        MOV DS, AX   ; DS = 1000

        ; YOUR CODE STARTS HERE
        start:
        ; Display prompt message
        LEA DX, msg
        MOV AH, 9
        INT 21h

        ; Read password and length count
        MOV SI,0
        MOV AH,1
        MOV BL,0

        input_loop:
            INT 21h
            CMP AL,13
            JE input_invalid
            MOV password[SI],AL
```

```
    INC SI
    INC BL
    CMP BL,20
    JE input_valid
loop input_loop

input_invalid:
    CMP BL,8
    JGE input_valid
    LEA DX, invalid_msg
    MOV AH, 9
    INT 21h
    JMP start

input_valid:
    MOV DI,0
    CMP BL,12
    JLE weak_len          ; Bypassing strong password implication in flag array
    MOV flag_array[DI],31h

weak_len:
MOV BH,00
MOV CX,BX
MOV SI,0

check_loop:
MOV DL,password[SI]
INC SI
CMP SI,CX
JG flag_check
CMP DL,33
JL final_invalid_check
JGE sp_ch1_check
num:
    JMP num_check
    sp_ch2:
        JMP sp_ch2_check
        cap:
            JMP cap_check
            sp_ch3:
                JMP sp_ch3_check
                small:
                    JMP small_check
                    sp_ch4:
```

```
                    JMP sp_ch4_check
                    final_invalid:
                       JMP final_invalid_check
loop check_loop


sp_ch1_check:
MOV DI,4
CMP DL,47
JG num
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

num_check:
MOV DI,1
CMP DL,57
JG sp_ch2
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

sp_ch2_check:
MOV DI,4
CMP DL,64
JG cap
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

cap_check:
MOV DI,2
CMP DL,90
JG sp_ch3
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

sp_ch3_check:
MOV DI,4
```

```
CMP DL,96
JG small
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

small_check:
MOV DI,3
CMP DL,122
JG sp_ch4
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

sp_ch4_check:
MOV DI,4
CMP DL,126
JG final_invalid
MOV flag_array[DI],31h

CMP SI,CX
JLE check_loop

final_invalid_check:
LEA DX, invalid_msg
MOV AH, 9
INT 21h
JMP end_code

flag_check:
MOV CX,5
MOV BL,0
MOV SI,0

flag_check_loop:
   MOV BH,flag_array[SI]
   CMP BH,31h
   JNE continue
   INC BL
   continue:
   INC SI
loop flag_check_loop
```

```
    CMP BL,2
    JLE weak
    CMP BL,4
    JLE medium
    JG  strong

    weak:
       LEA DX, weak_msg
       MOV AH, 9
       INT 21h
       JMP end_code
    medium:
       LEA DX, medium_msg
       MOV AH, 9
       INT 21h
       JMP end_code
    strong:
       LEA DX, strong_msg
       MOV AH, 9
       INT 21h
    end_code:

    ; YOUR CODE ENDS HERE

    MOV AX, 4C00H
    INT 21H

MAIN ENDP
END MAIN
```