Experiment 10
Push/pop instructions

Objective:
To learned how to use the stack and write assembly procedures.

Stacks:
POP Instruction
The POP instruction first copies the contents of the stack element pointed to by ESP into a 16- or 32-bit destination operand and then increments ESP. If the operand is 16 bits, ESP is incremented by 2; if the operand is 32 bits, ESP is incremented by 4.

There are two instruction formats:

1. POP r /m16
2. POP r /m32

PUSHFD and POPFD Instructions
The PUSHFD instruction pushes the 32-bit EFLAGS register on the stack, and
POPFD pops the stack into EFLAGS:
pushfd
popf

PUSHAD, PUSHA, POPAD, and POPA
• The PUSHAD instruction pushes all of the 32- bit general-purpose registers on the stack in the following order: EAX, ECX, EDX, EBX, ESP, EBP, ESI, and EDI.
• The POPAD instruction pops the same registers off the stack in reverse order.
• The PUSHA instruction pushes the l6-bit general-purpose registers (AX, CX, DX, BX, SP, BP, SI, DI) on the stack in the order listed.
• The POPA instruction pops the same registers in reverse order.

Note: we will use the following interrupt services:

Service 01h: DOS get character function
mov ah,01h ; returns ASCII code of character to AL
int 21h ; and echo it to the monitor

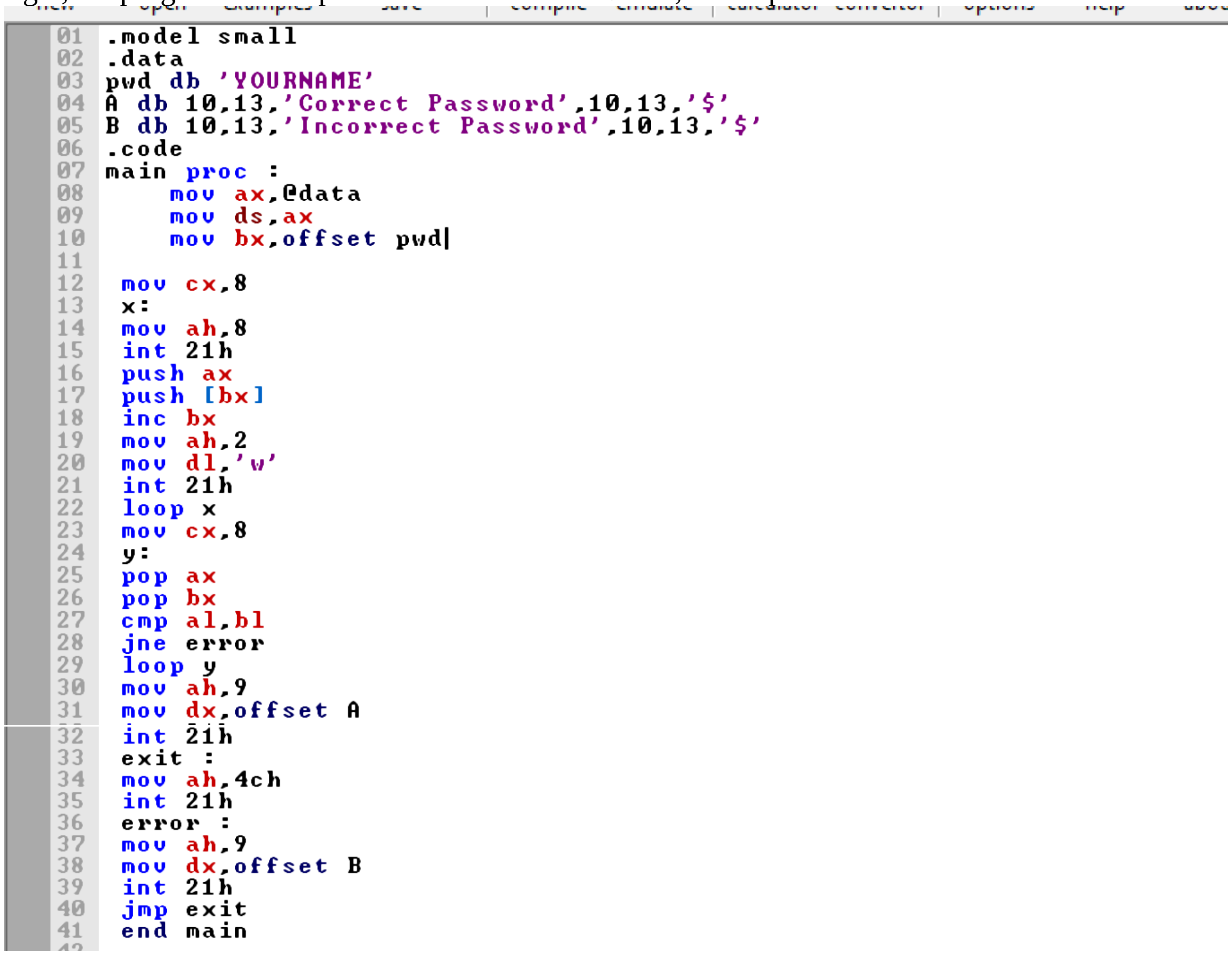
Service 02h: DOS print character function

mov ah,02h
mov dl,ASCII# ; ASCII code of character for print in DL
int 21h

Service 08h: Get character without echo
mov ah,08h ; returns ASCII code of character to AL
int 21h ; but don• t echo it to the monitor

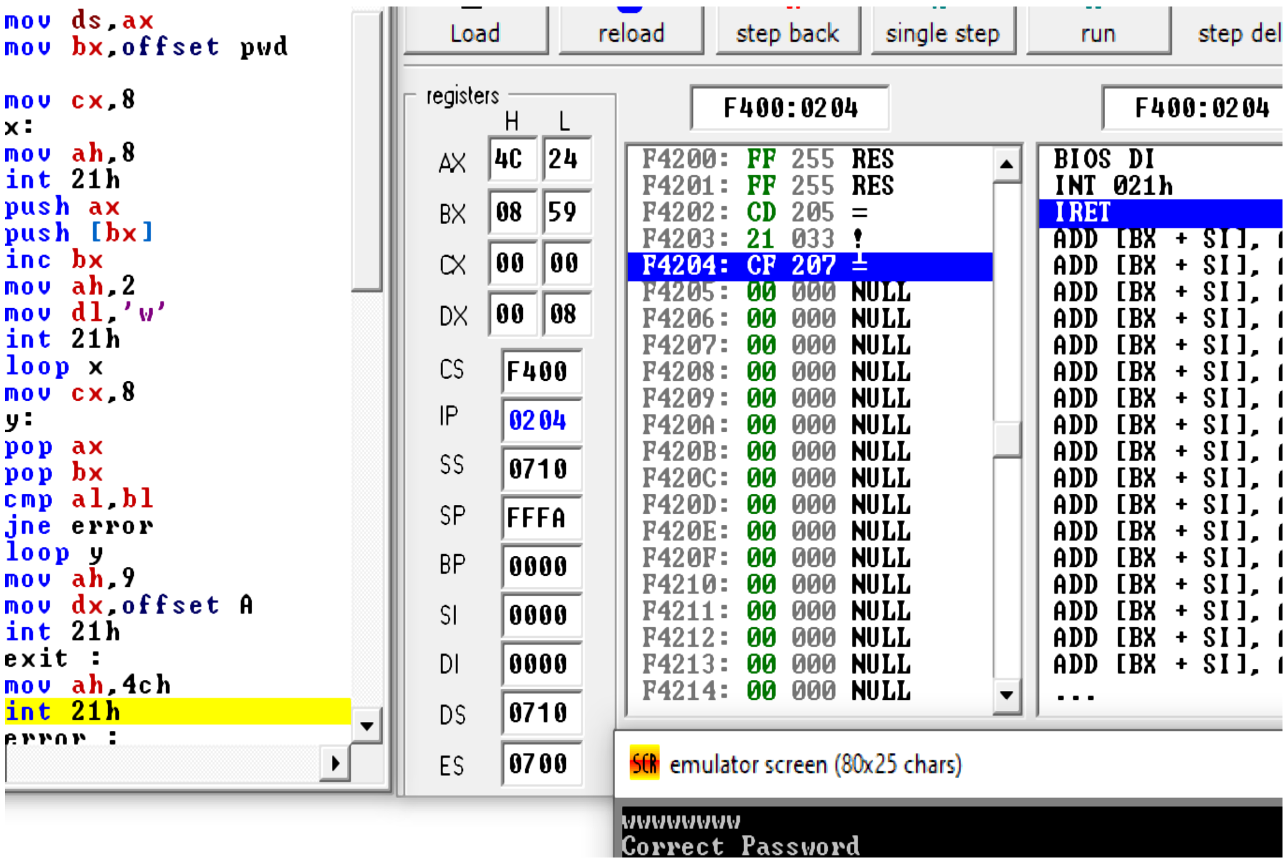
Lab Objective

Task #1: Write an assembly language program that asks the user to enter a password formed from 8 characters. The program prints the password as stars on the screen. If the password is right, the program should print 'Correct Password'. Else, it will print 'Incorrect Password'



```
.model small
.data
pwd db 'YOURNAME'
A db 10,13,'Correct Password',10,13,'$'
B db 10,13,'Incorrect Password',10,13,'$'
.code
main proc :
    mov ax,@data
    mov ds,ax
    mov bx,offset pwd

 mov cx,8
 x:
 mov ah,8
 int 21h
 push ax
 push [bx]
 inc bx
 mov ah,2
 mov dl,'w'
 int 21h
 loop x
 mov cx,8
 y:
 pop ax
 pop bx
 cmp al,bl
 jne error
 loop y
 mov ah,9
 mov dx,offset A
 int 21h
 exit :
 mov ah,4ch
 int 21h
 error :
 mov ah,9
 mov dx,offset B
 int 21h
 jmp exit
 end main
```



```
mov ds,ax
mov bx,offset pwd

mov cx,8
x:
mov ah,8
int 21h
push ax
push [bx]
inc bx
mov ah,2
mov dl,'w'
int 21h
loop x
mov cx,8
y:
pop ax
pop bx
cmp al,bl
jne error
loop y
mov ah,9
mov dx,offset A
int 21h
exit :
mov ah,4ch
int 21h
error :
```

```
AX 4C 24
BX 08 59
CX 00 00
DX 00 08
CS F400
IP 0204
SS 0710
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
```

```
F400:0204                       F400:0204

F4200: FF 255 RES               BIOS DI
F4201: FF 255 RES               INT 021h
F4202: CD 205 =                 IRET
F4203: 21 033 !                 ADD [BX + SI],
F4204: CF 207 ╧                 ADD [BX + SI],
F4205: 00 000 NULL              ADD [BX + SI],
F4206: 00 000 NULL              ADD [BX + SI],
F4207: 00 000 NULL              ADD [BX + SI],
F4208: 00 000 NULL              ADD [BX + SI],
F4209: 00 000 NULL              ADD [BX + SI],
F420A: 00 000 NULL              ADD [BX + SI],
F420B: 00 000 NULL              ADD [BX + SI],
F420C: 00 000 NULL              ADD [BX + SI],
F420D: 00 000 NULL              ADD [BX + SI],
F420E: 00 000 NULL              ADD [BX + SI],
F420F: 00 000 NULL              ADD [BX + SI],
F4210: 00 000 NULL              ADD [BX + SI],
F4211: 00 000 NULL              ADD [BX + SI],
F4212: 00 000 NULL              ADD [BX + SI],
F4213: 00 000 NULL              ADD [BX + SI],
F4214: 00 000 NULL              ...
```

```
wwwwwwww
Correct Password
```