

Lab # 4

Name : Suliman Sharif
Sap id :64620

Computer Organization & Assembly Language

Q1. You need to transfer values between registers and swap the contents of two registers using the MOV and XCHG instructions.

```
include 'emu8086.inc'
```

```
.model small
```

```
.stack 100h
```

```
.data
```

```
.code
```

```
main proc
```

```
    mov ax,'1'
```

```
    mov bx,'2'
```

```
XCHG ax,bx
```

```
    mov dx,ax
```

```
    mov ah,2
```

```
    int 21h
```

```
    mov dx,bx
```

```
    mov ah,2
```

```
    int 21h
```

```
    mov ah,4ch
```

```
    int 21h
```

```
main endp
```

end main

The screenshot shows the emu8086 software interface. At the top is a menu bar with file, edit, bookmarks, assembler, emulator, math, asciicodes, help, new, open, examples, save, compile, emulate, calculator, converter, options, help, and about. Below the menu is a code editor window titled "original source code" containing the following assembly code:

```

81
82
83 include 'emu8086.inc'
84
85 .model small
86 .stack 100h
87 .data
88 .code
89 main proc
10 mov ax,'1'
11 mov bx,'2'
12 XCHG ax,bx
13
14 mov dx,ax
15 mov ah,2
16 int 21h
17 mov dx,bx
18 mov dx,bx
19 mov ah,2
20 int 21h
21 int 21h
22
23 main endp
24
25

```

To the right of the code editor is a window titled "emulator screen (80x25 chars)" showing the assembly code again. Below the code editor is a status bar with "original source code" and "emulator screen (80x25 chars)".

At the bottom left is a window titled "emulator noname.exe" showing the registers and memory dump. The registers window has tabs for F400:0204h and F400:0204. The memory dump window shows memory starting at address F42001.

Registers	H	L
AX	AC	31
BX	00	31
CX	01	17
DX	00	31
CS	F400	
IP	0204	
SS	0710	
SP	00FA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Q2. You want to subtract two numbers stored in registers and store the result in another register using the SUB instruction.

include 'emu8086.inc'

.model small

.stack 100h

.data

.code

main proc

mov bl,9

mov cl,4

sub bl,cl

add bl,48

```
mov dl,bl
```

```
mov ah,2
```

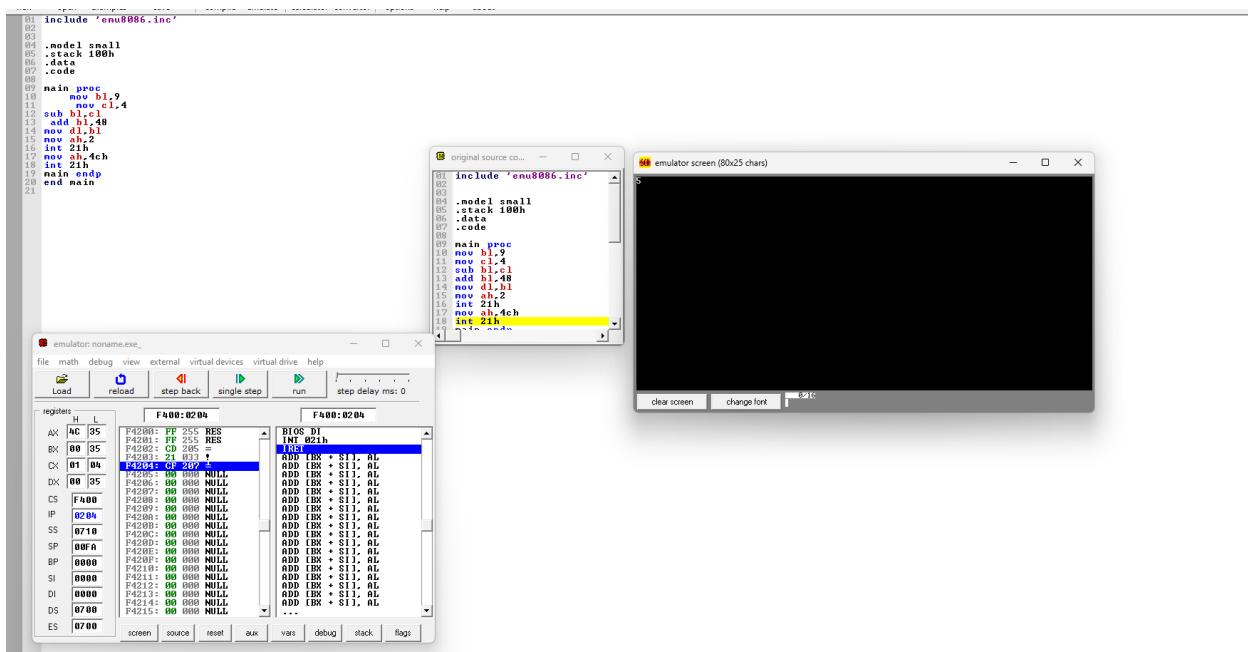
```
int 21h
```

```
mov ah,4ch
```

```
int 21h
```

```
main endp
```

```
end main
```



Q3. You need to compare two values and jump to a different part of the program if they are equal using CMP and JE.

```
include 'emu8086.inc'
```

```
.model small
```

```
.stack 100h
```

```
.data
```

```
.code
```

```
main proc
```

```
    mov al,10
```

```
mov bl,10  
CMP al,bl
```

JE show

```
print 'both are not same '
```

```
mov ah,4ch
```

```
int 21h
```

show:

```
print 'both are same'
```

```
mov ah,4ch
```

```
int 21h
```

```
main endp
```

```
end main
```

The screenshot shows the nmu8086 interface with three main windows:

- Original source code window:** Displays the assembly code for the program.
- Emulator screen window:** Shows the output of the program execution, displaying "both are same".
- Registers window:** Shows the state of CPU registers after the program has run. The registers include AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES, all containing their initial values (0000h).

The assembly code in the source window is:

```
.include 'emu8086.inc'  
.model small  
.stack 100h  
.data  
.code  
1.0 main proc  
2.0     mov al,10  
3.0     mov bl,10  
4.0     CMP al,bl  
5.0     JE show  
6.0     print 'both are not same '  
7.0     mov ah,4ch  
8.0     int 21h  
9.0     show:  
10.0    print 'both are same'  
11.0    mov ah,4ch  
12.0    int 21h  
13.0    main endp  
14.0    end main  
15.0
```

Q4. You need to manipulate bits in two registers. You will:

- 1. Set specific bits using the ORR instruction.**
- 2. Toggle certain bits using the XOR instruction.**

```
include 'emu8086.inc'
```

```
.model small
```

```
.stack 100h
```

```
.data
```

```
.code
```

```
main proc
```

```
    mov ax,10100
```

```
    mov bx,01010
```

```
    or ax,bx
```

```
    xor ax,bx
```

```
    mov dx,ax
```

```
    mov ah,2
```

```
    int 21h
```

```
    mov ah,4ch
```

```
    int 21h
```

```
main endp
```

```
end main
```

