

National University of Computer and Emerging Sciences



Laboratory Manual
for
Computer Organization and Assembly Language Programming
(EL 213)

Course Instructor	Ms. Aatira Anum
Lab Instructor(s)	M. Salman Mubarik Rasaal Ahmad
Section	H
Semester	Fall 2023

Department of Computer Science

FAST-NU, Lahore, Pakistan

In lab Questions

Task 1a: Write, assemble (with listing file) and open the code in AFD.

```
; lab2Task1code

[org 0x0100]

; no code because we just want to look
; at how data is stored in memory

mov ax, 0x4c00 ; termination statements

int 21h

; data declaration

Num1: dd 0A0B0C0Dh

Num2: dw 0102h

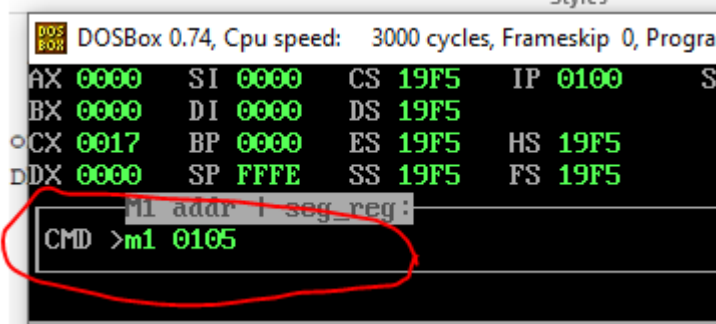
Num3 db 33h

Nums: dw 3456h

      db 99h

my_array: dw 0E0Fh, 0506h, 0708h, 0910h
```

Task1b: See the data in memory view M1 by writing the following statement.



Where 0105 is address at which num1 is pointing.

You can see the address of num1 from listing file, see fig1, just add 0100h to it (base address from where code starts to load)

See how data is stored in little endian form in memory and write the address that points to each byte.

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202
 BX 0000 DI 0000 DS 19F5 +2 20CD
 CX 0017 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
 DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

Address	Disassembly	Comment
0100	B8004C	MOV AX, 4C00
0103	CD21	INT 21
0105	0D0C0B	OR AX, 0B0C
0108	0A02	OR AL, [BP+SI]
010A	0133	ADD [BP+DI], SI
010C	56	PUSH SI
010D	3499	XOR AL, 99
010F	0F	DB 0F

DS:0105 0D 0C 0B 0A 02 01 33 56
 DS:010D 34 99 0F 0E 06 05 08 07
 DS:0115 10 09 C0 89 56 E4 89 46
 DS:011D E6 C7 46 F6 00 00 8B 46
 DS:0125 F6 D1 E0 D1 E0 C5 5E D8
 DS:012D 01 C3 8B 07 8B 57 02 85
 DS:0135 D2 75 04 85 C0 74 1C C7
 DS:013D 46 DC 00 00 8E 5E FC 83
 DS:0145 7D 0E 00 74 09 8B 46 F2
 DS:014D 48 3B 46 F6 7E 08 B8 01

M1: memory view

shows 0105 index now
 num1 is stored in LE form
 0D is at 0105,
 0C is at 0106,
 0B is at 0107 and
 0A is at 0108 address

Task2: Run the following code and see the changes in registers write the values of ax, al and ah after each line.

```
;lab2task2code
[org 0x0100]
;code
    Mov ax, [num1]      ;ax=?
    Mov ax, [num2]      ;ax=?
    Mov ax, [num2+2]    ;ax=?
    Mov ax, [num2+1]    ;ax=?
    Mov al, [num2+3]    ;ax=?
    Mov ah, [num1]      ;ax=?
    Mov ax, [array1]    ;ax=?
    Mov ax, [array1+2]  ;ax=?
    Mov al, [array2]    ;ax=?
    Mov al, [array2+1]  ;ax=?
    Mov ax, [array2]    ;ax=?
mov ax, 0x4c00 ; termination statements
int 21h
; data
    Num1: dw 0A0Bh
    Num2: dd 0C0D0E0Dh
    Array1: dw 0102h , 0304h
    Array2: db 05h , 06h, 07h
```

Task 3: Run the following code and see the changes in memory (in labels you declared)

```
;lab2task2code
[org 0x0100]
;code

    Mov ax, 9876h
    Mov bx, 5432h
    Mov [num1], ax
    Mov [num2], bx
    Mov [num2+2], bx
    Mov [array1], ax
    Mov [array2], bl
    Mov [array2], bl
    Mov word [num1], 0000h
    Mov byte [num1], 01h
    Mov byte [num2+1], 11h
    Mov word [array1+2], 3870h

mov ax, 0x4c00 ; termination statements
int 21h

; data

Num1: dw 0A0Bh
Num2: dd 0C0D0E0Dh
Array1: dw 0102h , 0304h
Array2: db 05h , 06h, 07h
```

Task 4: Identify the problems by running following instructions in AFD and correct them by replacing them with one or two instructions having the same effect.

- a. mov [02], [22]
- b. mov [wordvar], 20
- c. mov bx, al
- d. mov ax, [si+di+100]

Task 5: Convert the following C++ codes in assembly language.

Part a: Calculate the Absolute Difference

```
int x = 8;
int y = 15;
int z = 20;
```

```
int result;

if (x > y) {
    if (x > z) {
        result = x - z;
    } else {
        result = z - x;
    }
} else {
    if (y > z) {
        result = y - z;
    } else {
        result = z - y;
    }
}
```

Part B: Determine the Smallest Number

```
int p = 42;
int q = 18;
int r = 30;
int smallest;

if (p < q) {
    if (p < r) {
        smallest = p;
    } else {
        smallest = r;
    }
} else {
    if (q < r) {
        smallest = q;
    } else {
        smallest = r;
    }
}
```

Part C: Temperature Classification

```
int temperature = 78;
int classification;
```

```
if (temperature < 0) {  
    classification = 1; // Freezing  
} else if (temperature >= 0 && temperature < 25) {  
    classification = 2; // Cold  
} else if (temperature >= 25 && temperature < 70) {  
    classification = 3; // Moderate  
} else {  
    classification = 4; // Hot  
}
```