Assembly Language LAB

Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
2013
ECOM 2125: Assembly Language LAB
Created by: Eng. Ahmed M. Ayash
Modified and Presented By: Eihab S. El-Radie

Lab # 6

Addressing Modes

Objective:

- To know more about Assembly language, such as how CPU can access data in memory.

❖ Addressing Modes:

The addressing modes are different ways to find the memory address (how CPU can access data in memory). There is **five** addressing modes used in assembly programming with 80x86 family, these modes are:

1. Direct Addressing Mode

In this mode the 16 bit Offset is taken directly from the instruction.

```
.data
START DB 23H
mydouble DD 12345678H
.code
mov cl, START
mov START,BL
mov cl, [START+1]
mov CX, [200]
mov al, mydouble      ;error because the operand size do not match
mov al, byte PTR mydouble
mov ax, word PTR mydouble
```

2. Register Indirect: [bx], [si], [di], [bp]

In this mode the Offset is specified in either a **pointer register** or an **index register**. The pointer register can be either base register (BX) or base pointer (BP) and Index register can either be Source index (SI) or Destination index (DI) register.

Example:

```
mov bx,100h
mov ax,[bx]
```

This AM is useful to address an array since we can't use direct addressing to address all the elements in an array.

3. Based Addressing Mode

In this mode EA (effective address or physical address) is obtained by adding a displacement (offset) value in of BX or BP to the segment registers used are DS & SS.
When **Memory is accessed**, the 20 bit physical address is computed from BX and DS. On the other hand, when the **stack is accessed**, the 20 bit physical address is computed from BP and SS.

```
MOV AX, [BX]
MOV AX, [BX+5] ⟺ MOV AX,5[BX]  ⟺ MOV AX, [BX]+5
; Register added to a constant to form the offset
MOV AX,[BP]
MOV AX, [BP+5] ⟺ MOV AX,5[BP]  ⟺MOV AX, [BP]+5
```

4. Indexed Addressing Mode

The same as the Base AM but, the EA (effective address or physical address) is obtained by adding a displacement (offset) value in of SI or DI to the segment registers used are DS.

```
MOV AL, [SI]
MOV AL, [DI+5] ⟺MOV AL, 5[DI] ⟺MOV AL, [DI]+5
MOV AX, [SI]
```

5. Base-indexed:

```
MOV AX, [bx+si]
MOV AX, [bx+si+2]      ; Base-Indexed with displacement
MOV AL, START[bx+si]
```

Instructions that deals with addresses like **mov** takes this address as operand, there is many types of operands as:

- Register operand: mov ax,bx
- Immediate operand: mov ax,20h
- Direct operand: msg DB 10

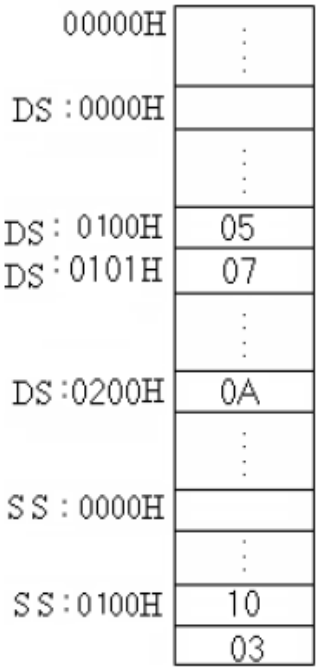
  :

  mov ax,msg

- **Segment Defaults:**
  The default segment is when the offset is used to point to a memory location.
  - bx,si,di ⟹ Data segment
  - bp ⟹ Stack segment
  - Based – indexed operands: mov ax,[bx+si]
    mov ax,[bp+bx] ; error , two base registers
    mov ax,[si+di] ; error , two index registers

Example:

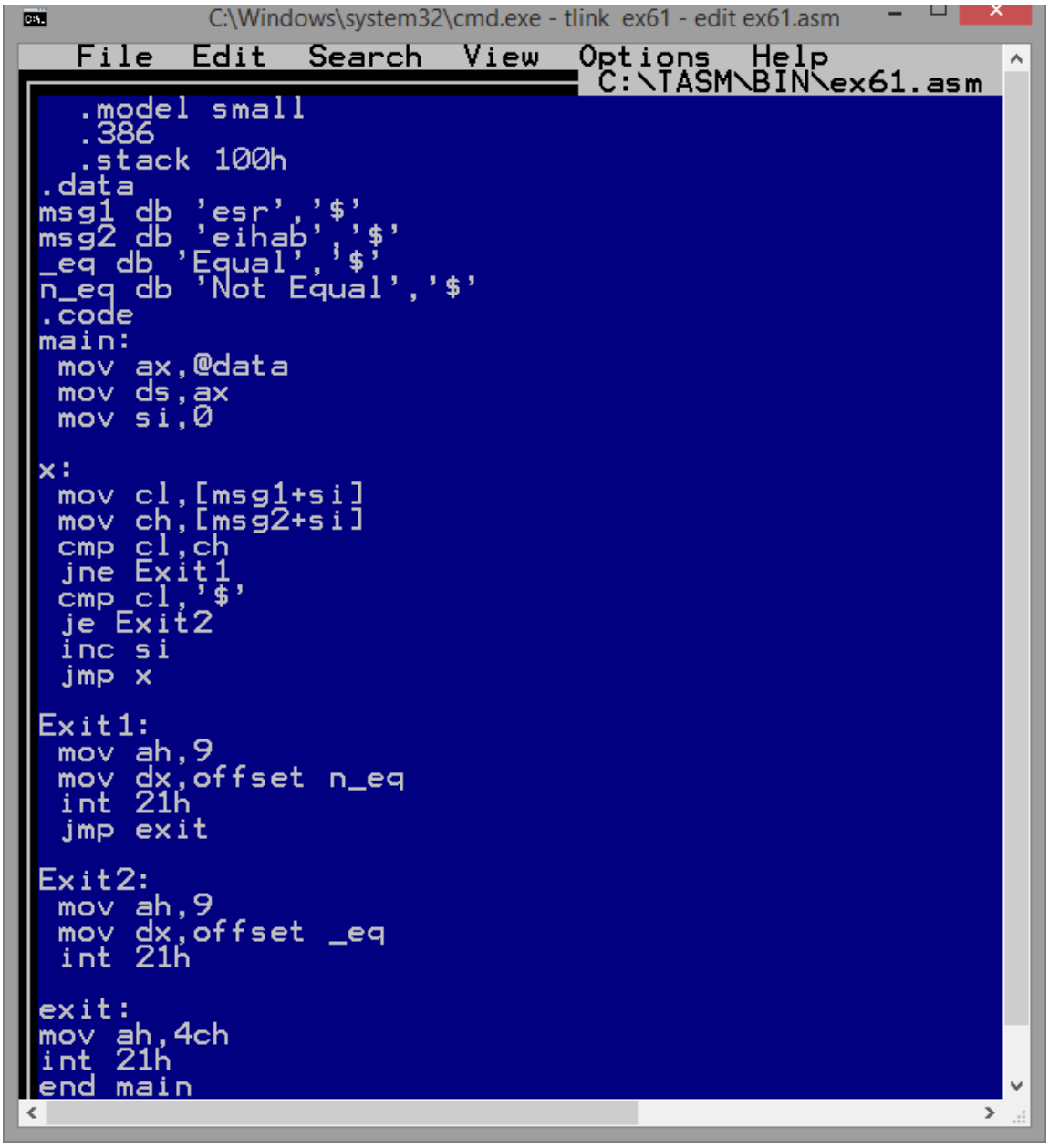
```
Mov SI, 100H
Mov DI, 200H
Mov BP, 100H
Mov BX, 1
MOV AX,[SI]              ; AX=0705H
MOV CL,[DI]              ; CL= 0AH
MOV CH,[BP]              ; CH=10H
MOV CH,DS:[BP]           ; CH=05H
MOV AH,SS:[SI]           ;AH=10H
MOV AL,[SI]+1            ;AL=07
MOV AL,1[SI]             ;AL=07
MOV AL,[SI+1]            ;AL=07
MOV AL,[SI+BX]           ;AL=07
MOV AL,DS:1[BP+SI]       ;AL= the content of
                         ;the memory location
                         ;with the address
                         ;DS:201H
```



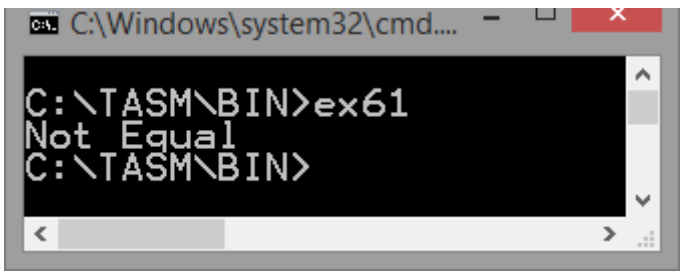
Lab work:

Excercise1:

Write an assembly program that determines if two strings are equal or not.

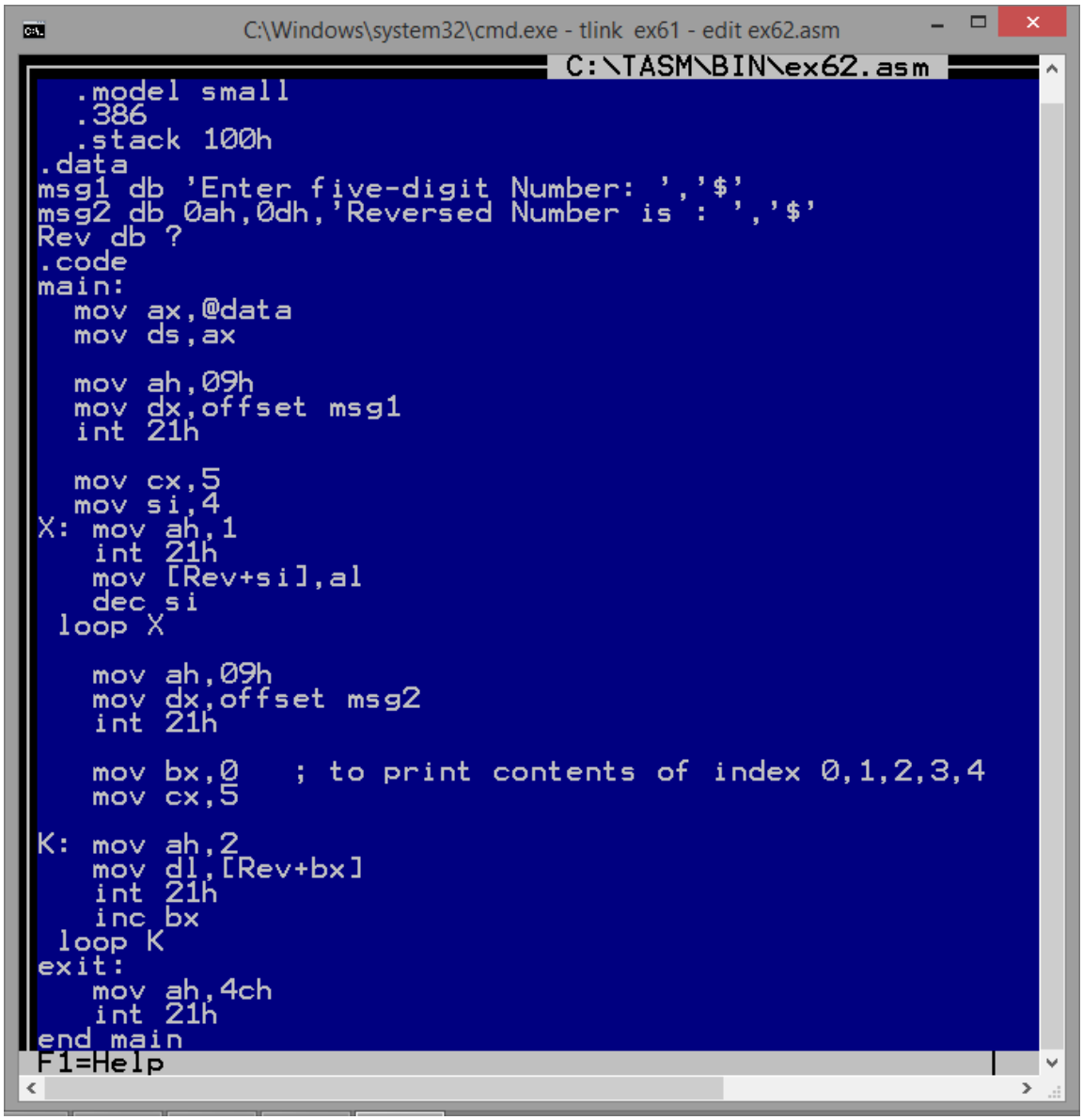


Output:

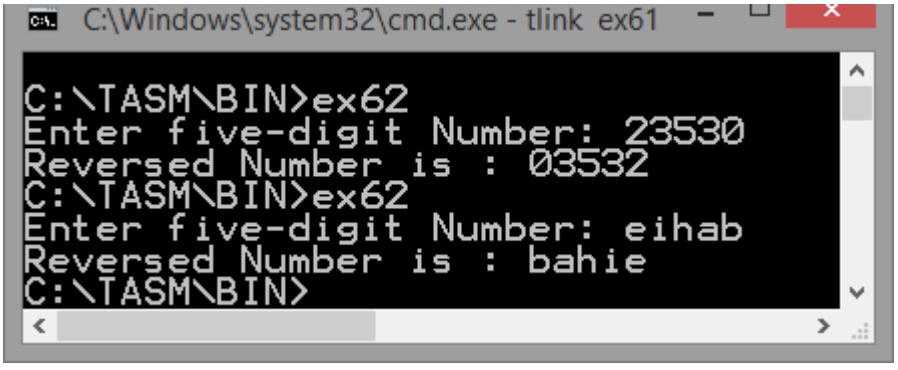


Excercise2:

Write an assembly program that allows the user to enter 5-digits Number, and then prints it in reversed-order digits.



Output:



Homework:

1. Write an assembly program that allows the user to enter two strings and determines if they are equal or not. (Each string must end with $).

2. Improve Exercise 2 code so that the user can only enter decimal numbers. If the user enters anything else, an error message should appear and the program should ask the user again to enter digits.