

# Actividad 4

**Christian Geovany Muñoz Rodríguez**

**Ingeniería en computación**

**Código: 221350605**

**Seminario de Solución de Problemas de  
Traductores de lenguajes – D09 (Lunes y  
Miércoles de 1 a 3)**

**Maestro: José Juan Meza Espinosa**

**Universidad de Guadalajara**

**Centro Universitario de Ciencias Exactas e  
Ingenierías**

**22 de febrero del 2023**



## Código:

```
.model small
```

```
.stack 100h
```

```
.data
```

```
datos db
```

```
80,100,119,135,148,156,160,159,152,142,127,109,90,70,51,33,18,8,1,0,4,12,25,41,6  
0,80,100,119,135,148,156,160,159,152,142,127,109,90,70
```

```
.code
```

```
start:
```

```
    mov ax, @data
```

```
    mov ds, ax
```

```
    mov si, offset datos
```

```
    mov al, [si]
```

```
    mov al, datos[3]
```

```
    mov al, datos+10
```

```
    mov si, offset datos
```

```
    mov bx, 6
```

```
    mov al, [si+bx]
```

```
    mov si, offset datos
```

```
    mov bx, 20
```

```
    mov al, [si+bx+3]
```

```
    mov si, offset datos
```

end

Este código muestra ejemplos de diferentes modos de direccionamiento en ensamblador x86. Primero, se inicializa el segmento de datos con la etiqueta "@data". Luego, se define una matriz de 27 bytes llamada "datos".

A continuación, se muestran los siguientes modos de direccionamiento:

The screenshot shows the emulator interface with the following details:

- Emulator Title Bar:** emulator: Act4.exe
- Menu Bar:** file, math, debug, view, external, virtual devices, virtual drive, help
- Toolbar:** Load, reload, step back, single step, run, step delay ms: 0
- Registers Window:**
  - H L:** 07 50
  - BX:** 00 00
  - CX:** 01 70
  - DX:** 00 00
  - CS:** 0723
  - IP:** 0000
  - SS:** 0710
  - SP:** 0100
  - BP:** 0000
  - SI:** 0000
  - DI:** 0000
  - ES:** 0700
- Register Data Table:**

Register	Value	Comment
AX	07 50	
BX	00 00	
CX	01 70	
DX	00 00	
CS	0723	
IP	0000	
SS	0710	
SP	0100	
BP	0000	
SI	0000	
DI	0000	
ES	0700	
- Register Data Table (Continued):**

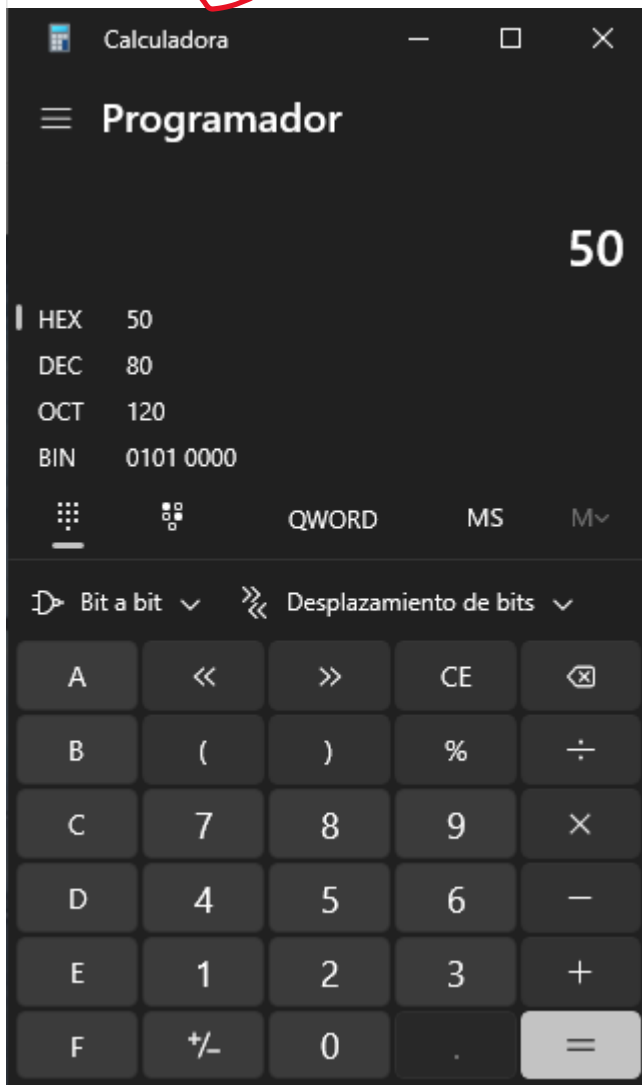
Register	Value	Comment
0723:0000	B8 184	MOV AX, 00720h
0723:0001	20 032	MOV DS, A
0723:0002	07 067	MOV SI, 00000h
0723:0003	BE 142	ISI
0723:0004	D8 216	MOV AL, [00003h]
0723:0005	BE 190	MOV AL, [0000Ah]
0723:0006	00 000	MOV SI, 00000h
0723:0007	00 000	MOV BX, 00000h
0723:0008	8A 138	MOV AL, [BX + SI]
0723:0009	04 004	MOV SI, 00000h
0723:000A	A0 160	MOV BX, 00014h
0723:000B	03 000	MOV AL, [BX + SI] + 03h
0723:000C	00 000	MOV SI, 00000h
0723:000D	A0 160	MOV CX, 00004h
0723:000E	0A 010	MOV AL, [SI] - 01h
0723:000F	00 000	...
- Original Source Code Window:**

```

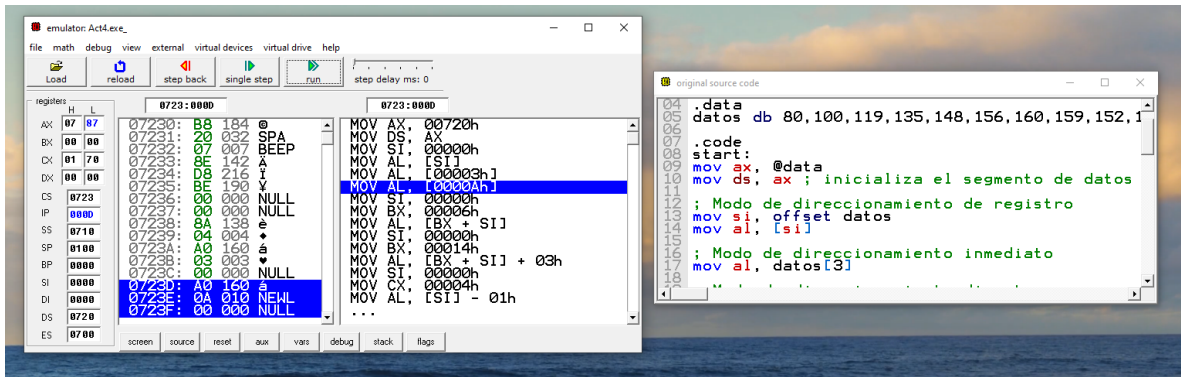
04 .data
05 datos db 80,100,119,135,148,156,160,159,152,1
06
07 .code
08 start:
09 mov ax, @data
10 mov ds, ax ; inicializa el segmento de datos
11
12 ; Modo de direccionamiento de registro
13 mov si, offset datos
14 mov al, [si]
15
16 ; Modo de direccionamiento inmediato
17 mov al, datos[3]

```

```
.model small  
.stack 100h  
  
.data  
datos db 80, 100, 119, 135, 148, 156, 160, 159, 152, 142, 127, 109,
```



**Modo de direccionamiento inmediato:** se carga el byte en la posición de memoria 4 de la matriz directamente en el registro AL.

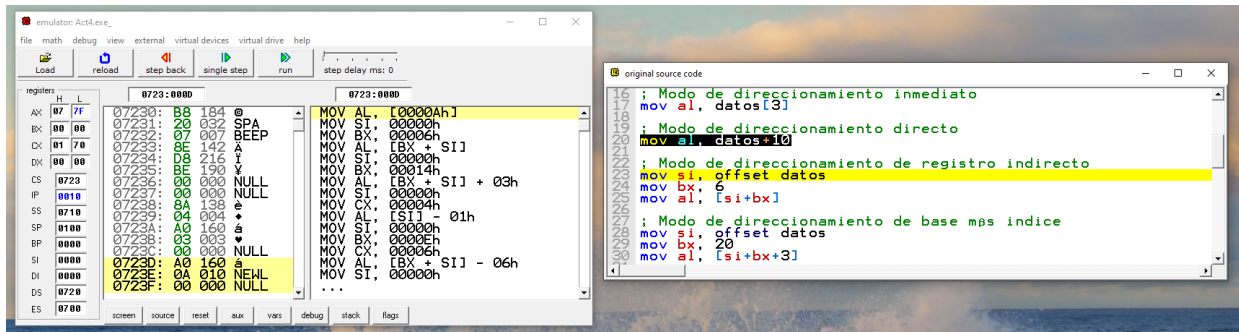


```
.model small
.stack 100h

.data
datos db 80,100,119,135,148,156,160,159,152,142,127,109,
```



**Modo de direccionamiento directo:** se carga el byte en la posición de memoria 10 de la matriz sumando el valor de 10 a la dirección base de la matriz.

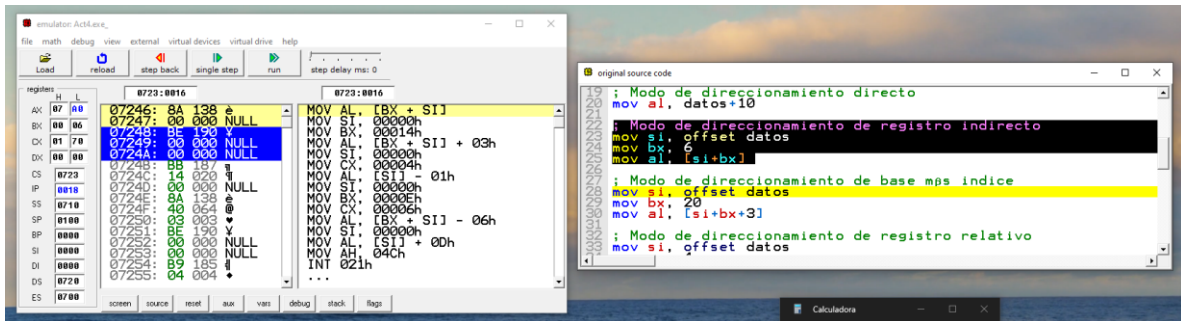


```
.model small
.stack 100h

.data
datos db 80,100,119,135,148,156,160,159,152,142,127,
```

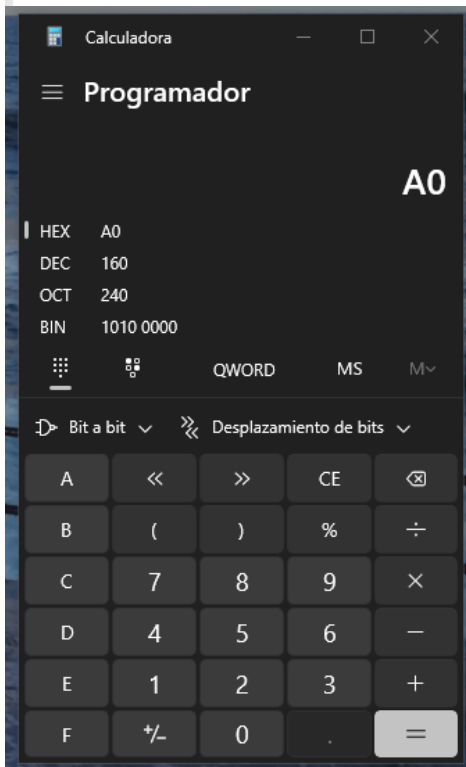


**Modo de direccionamiento de registro indirecto:** se carga la dirección base de la matriz en el registro SI y se carga el sexto byte de la matriz sumando el valor de 6 al registro SI.

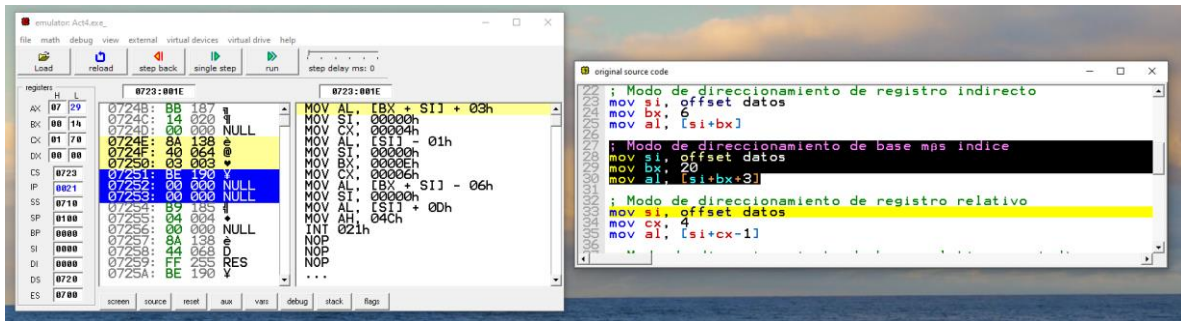


```
.model small
.stack 100h

.data
datos db 80,100,119,135,148,156,160,1
```



**Modo de direccionamiento de base más índice:** se carga la dirección base de la matriz en el registro SI y se carga el byte en la posición de memoria 20 + 3 (23) sumando el valor de 20 y el valor de 3 al registro SI.



```

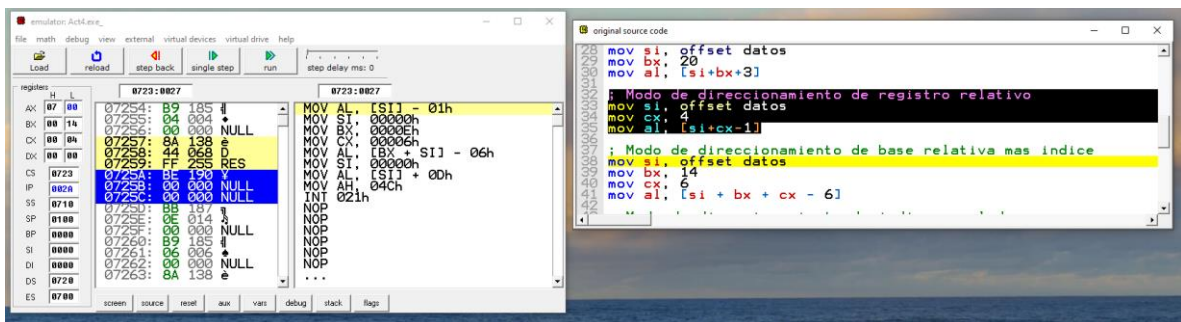
.model small
.stack 100h

.data
datos db 80,100,119,135,148,156,160,159,152,142,127,109,90,70,51,33,18,8,1,0,4,12,25,41,60,1

```

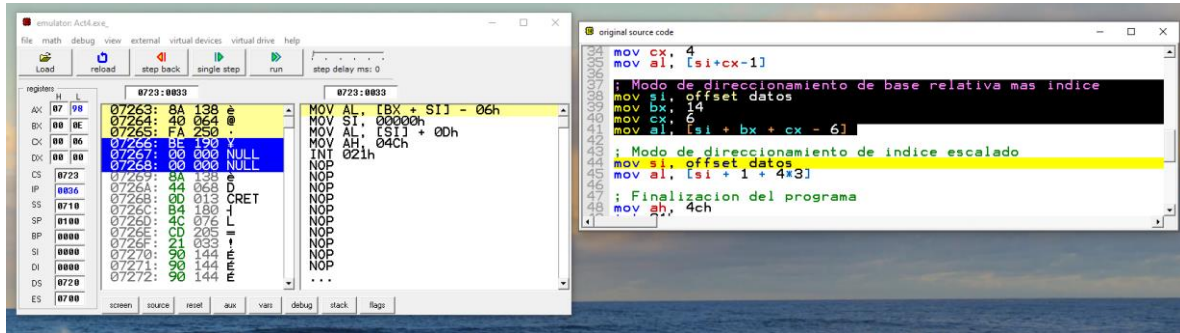


**Modo de direccionamiento de registro relativo:** se carga la dirección base de la matriz en el registro SI y se carga el cuarto byte de la matriz sumando el valor de 4 al registro CX y restando 1 para obtener la posición correcta de memoria.

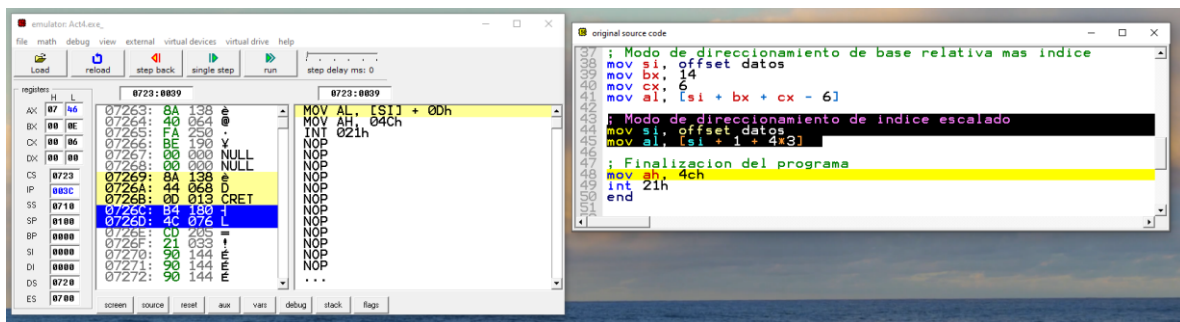




**Modo de direccionamiento de base relativa más índice:** se carga la dirección base de la matriz en el registro SI y se carga el byte en la posición de memoria  $14 + 6$  (20) sumando el valor de 14 y el valor de 6 al registro BX y al registro CX.



**Modo de direccionamiento de índice escalado:** se carga el byte en la posición de memoria  $1 + 4 * 3$  (13) sumando el valor de 1 y el valor de 4 multiplicado por 3 al registro SI.



## Conclusiones

En la programación de computadoras, el direccionamiento es una parte crucial del proceso de diseño y creación de programas. En este programa se demostró una variedad de modos de direccionamiento, cada uno con su propio propósito y aplicación.

Esta práctica me enseñó que el direccionamiento efectivo es esencial para escribir programas eficientes y funcionales. Además, aprendí que diferentes tipos de datos y variables pueden requerir diferentes modos de direccionamiento para acceder a ellos.

Por lo tanto, es importante para cualquier programador tener una comprensión sólida de los diferentes modos de direccionamiento, y cómo y cuándo utilizarlos de manera efectiva. Esto garantiza un código más legible y mantenible, y un rendimiento óptimo del programa. En resumen, esta práctica fue una excelente

oportunidad para mejorar mis habilidades de programación y comprensión del direccionamiento en ensamblador x86.

## **Bibliografía:**

Brey, B. B. (2006). *Microprocesadores Intel : 8086/8088, 80186/80188, 80286, 80386 y 80486, Pentium, procesador Pentium Pro, Pentium II, Pentium III y Pentium 4: arquitectura, programación e interfaces*. Pearson Educación.