Programación en ensamblador i8086

Tema: Algoritmo de búsqueda binaria

Asignatura: Arquitectura de computadoras

Nombre: Lautaro Galante

Profesor: Víctor Teppaz

Fecha: 11 de noviembre de 2023

Como proyecto final de la materia de Arquitectura de computadoras, se nos pidió crear un programa en ensamblador 8086, yo decidí implementar el algoritmo de búsqueda binaria, ya que este me haria pensar un poco sobre como hacerlo y asi poder aprender sobre las distintas instrucciones de la arquitectura.

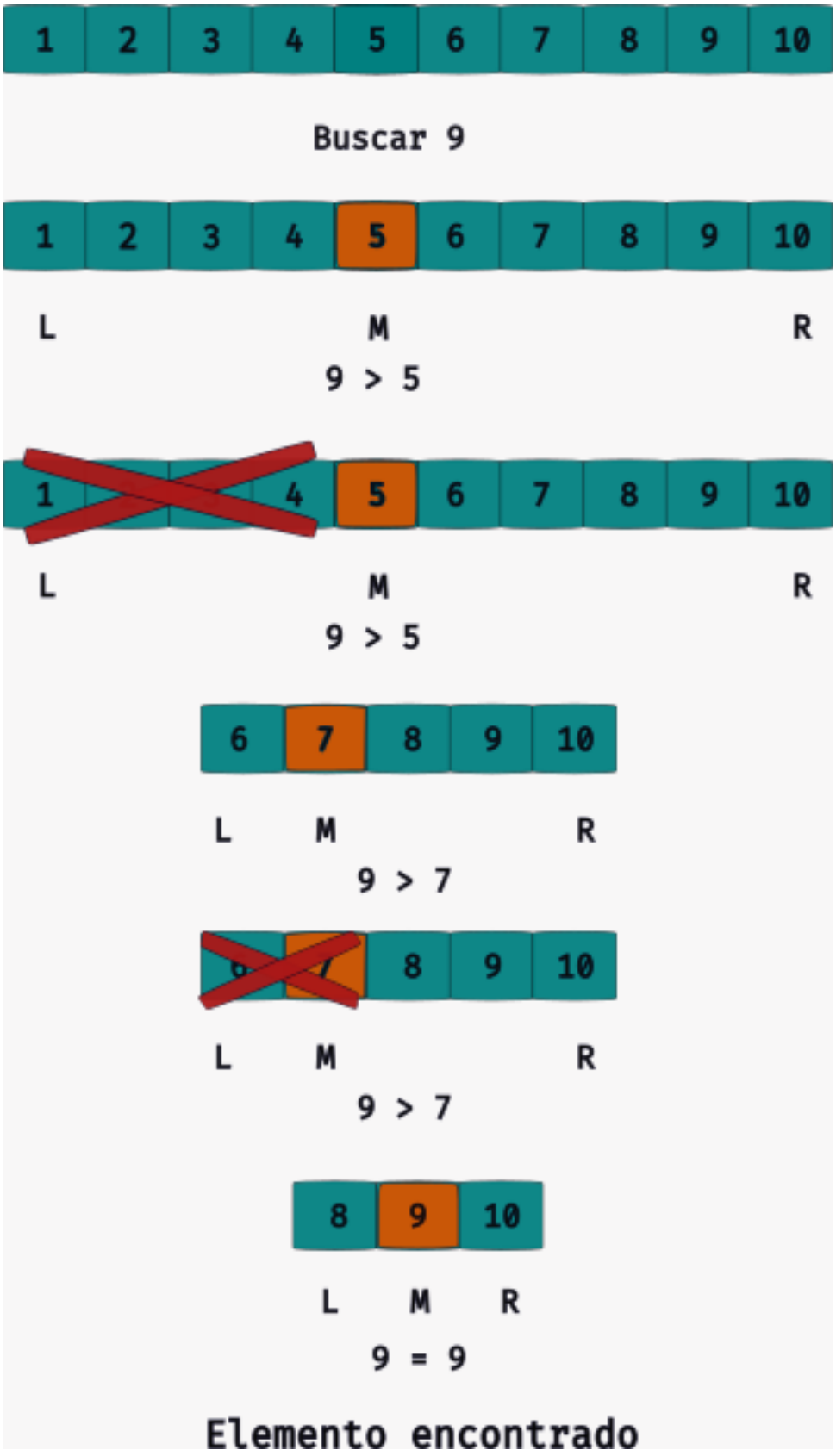
En ciencias de la computación y matemáticas, el algoritmo de búsqueda binaria o binary search en inglés, es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado. Compara el valor con el elemento en el medio del array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre.

La búsqueda binaria es computada en el peor de los casos en un tiempo logarítmico, realizando $O(\log n)$ comparaciones, donde **n** es el número de elementos del arreglo y **log** es el logaritmo. La búsqueda binaria requiere solamente $O(1)$ en espacio, es decir, que el espacio requerido por el algoritmo es el mismo para cualquier cantidad de elementos en el array.

Para que se entienda mejor dejo expresado el algoritmo en pseudocódigo:

```
Function busqueda_binaria(Array, n, T):
    Izquierda := 0
    Derecha := n − 1
    while Izquierda ≤ Derecha do
        medio := (Izquierda + Derecha)/2
        if Array[medio] < T then
            Izquierda := medio + 1
        else
            if Array[medio] > T then
                Derecha := medio − 1
            else
                return medio
            end
        end
    end
    return no se encontro
```

Aqui hay una representación gráfica de los pasos que realiza el algoritmo



A continuación explicare que hace cada parte del código en ensamblador

Bloque 1

```
1       ORG 100h
2
3       .DATA
4           numeros db 1, 2, 3, 4, 5, 6, 7, 8, 9
5           numero db ?
6           mensaje db "Ingrese el numero a buscar: ", "$"
7           resultado db "El indice del valor ingresado es: ", "$"
8           left db 0
9           right db 8
10          middle db ?
11          value db ?
12
```

Al inicio del código en la linea **1** declaro **ORG 100h** que es una directiva del compilador, luego en la linea **3** defino un sector de datos **.DATA** en el cuál tengo un array y distintas variables, el primero es un array **numeros** que es de tipo **db** que tiene una capacidad de **8 bits** es decir **1 byte**, este contiene números del **1** al **9**, en la linea **5** una variable **numero** del mismo tipo la cuál la inicialize con el simbolo **?** que significa que aún no se le a asignado un valor, en la linea **6** una variable string **mensaje** y en la **7** **resultado** para mostrar los mensajes para que se ingrese el número, y para indicar el valor de salida. Luego hay cuatro variables, **left, rigth, middle y value**, **left** guarda el primer indice del array que es **0** y **right** guarda el inidice **8** que es el ultimo elemento del array en este caso es el **9**, en la variable **middle** y **value** aún no le asigne ningún valor, luego mas adelante en el codigo se explicara para que son cada variable.

Bloque 2

```
1       .CODE
2         MAIN:
3           mov ax, numeros
4           mov dx, offset mensaje
5
6           mov ah, 9h
7           int 21h
8
9           mov ah, 01h
10          int 21h
11          sub al, '0'
12          mov numero, al
13
14          call BINARYSEARCH
15          call PRINT
16
```

En el bloque **2** de codigo dentro del sector **.CODE** se encuentra el procedimiento principal **.MAIN** que tiene las primeras instrucciones y las llamadas a otros procedimientos, en la linea **3** se copian los valores del array **numeros** a el registro entero de 16 bits **ax** usando la instrucción **mov**, en la linea **4** se copia la dirección de memoria de la variable **mensaje** utilizando el operador **offset** en el registro **dx**, en la linea **6** se copia el valor hexadecimal **9h** al registro **ah** que es la parte alta de **ax** esta es una función para imprimir una cadena por consola, en la linea **7** se ejecuta la interrupción **int 21h** el **bios** busca el código de función que coincide con la función cargada en el registro **ah** y procede a imprimir la cadena **Ingrese el numero a buscar:** que se encuentra guardada en la variable **mensaje**.

En la linea **9** se copia el valor de la función **01h** en el registro **ah** esta función es para leer un carácter desde el teclado, en la linea **10** se ejecuta la interrupción, en la linea **11** se convierte el valor numérico **ascci** a decimal y se almacena en el registro **al**, luego en la linea **12** se copia el valor del registro **al** en la variable **numero**, en la linea **14** se utiliza el operador **call**

para llamar al procedimiento **BINARYSEARCH** que es donde ocurre toda la lógica del algoritmo, y en la linea **15** se llama al procedimiento **PRINT** que lo utilizo para imprimir los valores.

Bloque 3

```
1       BINARYSEARCH PROC
2           mov ax, 0
3           mov bl, left
4           mov bh, right
5
6           SEARCHLOOP:
7               cmp bl, bh
8               jg SEARCHEND
9
```

En este bloque **3** de código definimos el procedimiento **BINARYSEARCH**, donde dentro en la linea **2** limpiamos el registro de **ax** copiando el valor **0**, en la linea **3** copiamos el valor almacenado en la variable **left** a **bl** la parte baja de **bx**, en la linea **4** copiamos el valor del la variable **right** a **bh** su parte alta.

Luego en la linea **6** definimos un ciclo **SEARCHLOOP**, en el utilizamos la instrucción **cmp** su función es equivalente a la de una resta, ya que resta el segundo operando al el primer operando, pero no guarda el resultado de la resta, sino que cambia las banderas de estado en el registro de banderas, **CF:** se establece en 1 si el primer operando es menor al segundo, En este caso como el segundo operando es mayor al primero, la bandera **CF** se setea en **1**.

En la linea **8** la instrucción **jg** corre la ejecución del programa a donde se encuentre la definición de la etiqueta asignada a su derecha, que en este caso es **SEARCHEND**, esto sucede si el primer operando es mayor al segundo, lo que llevaria a terminar el ciclo ya que no tiene mucho sentido seguir con la búsqueda por que cuando se llega a esta instancia significa que se encontro el indice del valor ingresado.

Bloque 4

```
1            mov al, bh
2            add al, bl
3            mov cl, 2
4            div cl
5
6            mov cx, 0
7            mov cl, numero
8
```

Este bloque **4** de código es una continuación de las instrucciónes que se encuentran dentro del ciclo **SEARCHLOOP**, en la linea **1** copiamos el valor de **bh** en **al**, en la linea **2** con la instrucción **add** sumamos el valor de **bl** a **al**. En la linea **3** copiamos el valor **2** al registro bajo **cl**, y en la linea **4** usamos la intrucción **div** para dividir en dos el valor que se encuentra en **al**, por lo que ahora tendremos dos valores en el registro **ax**, en **ah** se ecuentra el residuo y en **al** se encuentra el cociente.

En la linea **6** limpiamos el valor del registro **cx**, y en la linea **7** copiamos el valor del la variable **numero**, que es el valor que ingresamos teclado.

Bloque 5

```
1            mov ah, 0
2            mov si, ax
3
4            mov middle, al
5            mov al, numeros[si]
6            mov value, al
7
8            cmp value, cl
9            jne second
10           jmp SEARCHEND
11
```

En el bloque **5** de código, en la linea número **1** se sobreescribe con cero el valor del registro alto **ah** para que en el registro entero **ax** solo quede en su parte baja **al** el resultado de haber dividido la sumatoria de los valores **left** y **right** entre **2**, entonces en la linea **2** copiamos el valor de **ax** en **si** (**source index**) que es el índice de la mitad del array **numeros**.

Igualmente necesitamos guardar el valor del índice medio en la variable **middle**, eso ocurre en la linea **4** donde copiamos el valor de el registro **al** ya que necesitamos reescribir ese registro para que en la linea **5**, utilicemos el **si** para iterar dentro del array **numeros** de la siguiente manera **numeros**[**si**], entonces con el índice medio dentro del array accedemos al número que se encuentra en esa posición media y lo copiamos en el registro **al**, luego en la linea **6** copiamos ese número en la variable **value**.

En la linea **8** comparamos ese número medio con el número que se ingreso por teclado que ya lo tenemos almacenado en el registro **cl** que explique en el bloque **4**, por lo que si los números no son iguales la instrucción **jne** salta al procedimiento **second**, sino la instrucción **jmp SEARCHEND** termina el bucle y se retorna ese número ya que son iguales.

Bloque 6

```
1           second:
2               cmp value, cl
3               jg third
4
5               mov dx, 0
6               mov dl, middle
7               inc dl
8               mov bl, dl
9
10              jmp SEARCHLOOP
11
```

En el bloque **6** en la linea **1** se encuentra la definición del procedimiento **second**, dentro del mismo en la linea **2** comparamos el valor medio del array, es decir el número que se encuentra en la mitad con el número ingresado por teclado, en la linea **3** si el primer operando es mayor al segundo se salta al procedimiento **third**, sino se procede con las operaciones que comienzan en la linea **5**, se copia el valor **0** en el registro **dx** para limpiarlo, luego en la linea **6** se copia el indice del valor medio que esta guardado en la variable **middle** en **dl**, luego en la linea **7** se usa la instrucción **inc** para incrementar en uno el registro **dl**.

En la linea **8** se copia el valor incrementado de **dl** en el registro **bl** ya que este corresponde a la parte izquierda del array **numeros**, para recortar la parte de array en donde no se encuentra valor que se ingreso por teclado. Luego en la linea **10** se salta al principio del loop utilizando la instrucción **jmp SEARCHLOOP** para que este se vuelva a iterar.

Bloque 7

```
1              third:
2                  mov dx, 0
3                  mov dl, middle
4                  dec dl
5                  mov bh, dl
6
7                  jmp SEARCHLOOP
8
9          SEARCHEND:
10         RET
11
12     BINARYSEARCH ENDP
13
14 END MAIN
15
```

En el bloque **7** de código en la linea **1** se define el procedimiento **third**, donde en la linea **2** limpiamos el registro **dx**, en la linea **3** copiamos el valor de la variable **middle** en **dl**, y luego en la linea **4** utilizamos la instrucción **dec** para decrementar en **1** el valor que se encuentra en **dl**, ya que en este caso no necesitamos hacer ninguna comparación, por que el programa llego hasta este procedimiento por que la comparación del procedimiento anterior **second** dio que el primer operando era mayor al segundo, entonces en este procedimiento solo se realizan las operaciones necesarias, para que en la linea **5** se copie el nuevo valor de **dl** en **bh** que es la parte derecha del array **numeros**, luego en la linea **7** con la instrucción **jmp SEARCHLOOP** volvemos a iterar hasta que se encuentra el valor que buscamos, cuando la ejecución llega nuevamente a la comparación que ocurre en el bloque de código número **3** y el valor del registro **bl** es mayor que **bh** el bucle **SEARCHLOOP** termina y se retorna en la linea **10** con la instrucción **RET**

Bloque 8

```
1       PRINT:
2           mov dl, 13
3           mov ah, 2
4           int 21h
5
6           mov dl, 10
7           mov ah, 2
8           int 21h
9
10          mov dx, offset resultado
11          mov ah, 9h
12          int 21h
13
14          mov dl, middle
15          add dl, '0'
16          mov ah, 02h
17          int 21h
18
```

Al retornar, el procedimiento que sigue en el orden de llamadas es el procedimiento **PRINT** que sirve para imprimir el índice del número que ingresamos por teclado, que es la posición donde se encuentra ese número en el array. En la linea **2** del bloque **8** de código se copia el valor **13** en el registro **dl**, en la linea **3** se copia el valor **2** en **ah**, en la linea **4** se usa la instrucción **int 21h**, que en conjunto estas tres instrucciónes lo que hacen es aplicar un retorno de carro, para que en la linea **6, 7 y 8** se aplique un salto de linea en la consola, para que el resultado no se imprima pegado a la impresión anterior.

En la linea **10** se copia la dirección de memoria de la variable **resultado** usando la instrucción **offset** en el registro **dx**, en la linea **11** se copia el valor de la función **9h** en **ah**, luego se usa la interrupción **int 21h** para imprimir la cadena que se guarda en la variable resultado que es **El indice del valor ingresado es:** , en la linea **14** se copia el indice del número buscado que esta en **middle** al registro **dl**, en la linea **15** se convierte ese valor **ascci** a decimal, para que en la linea **16 y 17** se imprima el indice del número buscado por consola.