Instrucciones en ensamblador[1]

El lenguaje ensamblador como leguaje de programación no está excluido de estructura. Todas las buenas prácticas (sangrado, legibilidad, documentación) en la escritura de programas se vuelven de una mayor importancia, pues ayudarán al programador, y a terceros, si es el caso, a tener un correcto mantenimiento del código para su posible reúso.

Al hacer operaciones matemáticas en un programa de ensamblador, siempre debemos tener en cuenta el estado o los valores de los registros de memoria antes de la operación, ya que, generalmente, los resultados de las operaciones los recuperaremos en otros registros determinados y no se desea eliminar datos que podríamos usar más adelante en el programa.

Entonces, para solucionar un problema de cómputo recurrimos a estructuras probadas que son comunes en los lenguajes de programación, por ejemplo, los ciclos for, while, la condicional if, la escalera if then else, la estructura case, funciones para la entrada y salida de datos, etc.

En ensamblador estas estructuras no forman directamente parte de las instrucciones básicas –los nemónicos. Pero todas ellas se pueden implementar sin mayor problema a través de instrucciones base.

Para esta unidad aún no se han introducido las interrupciones, que nos permitirían interactuar con el usuario a través del teclado y la pantalla, por lo que todos los ejemplos que desarrollamos hasta el momento hacen operaciones internas que verificaremos en los valores de los registros de la PC.

Ahora que conocemos de manera general los tipos de instrucciones, combinaremos unos y otros para crear procesos cada vez más complejos.

Una primera tarea que se puede realizar es trabajar con arreglos de valores, que en el caso de contener código ASCII les llamamos cadenas. La forma de definir estos arreglos o cadenas como variables es declararlos en el segmento .DATA, definir el tipo de arreglo (DB para bytes, DW para palabras –dos bytes-), definir el tamaño y el contenido, si lo conocemos.

En el caso de las cadenas de texto siempre se terminan con el carácter ‘$’ para que el sistema sepa dónde termina. Para inicializar los arreglos se usa la directiva DUP (valor), con el valor de inicialización; puede usarse DUP(?) si no nos interesa un valor particular inicial.

[1] Documento elaborado por el Dr. Carlos Jiménez Gallegos.

Las variables de tipo arreglo son conjuntos de localidades de memoria contiguas. Esto quiere decir que la dirección de una variable de tipo arreglo simplemente apunta al primer elemento de la variable. Para ir apuntando a la siguiente posición es suficiente con incrementar la dirección en 1, en el caso de arreglos de tipo byte, o incrementar en 2 si se trata de un arreglo del tipo *word.*

No olvide que en las operaciones para direccionar las posiciones de un arreglo debe usarse el registro índice de destino *DI* para arreglos a los que se les va escribir un valor. Y el registro índice fuente *SI* para un arreglo del que estamos leyendo valores. Para escribir un valor usaremos el formato *MOV [DI]*, valor y para leer un valor MOV AH, [SI].

Note en el siguiente ejemplo el uso de la instrucción **LOOP** para realizar repeticiones fijas de bloques de código, que en este caso nos ayudarán a realizar operaciones a lo largo de los arreglos:

```
.MODEL SMALL                ;Modelo pequeño

.STACK 100h                 ;Segmento de pila 256 posiciones

CR EQU 13                   ;Retorno de carro
LF EQU 10                   ;nueva línea

.DATA                       ;Segmento de datos

TEXTO1 DB 'Datos y ciclos  $'  ;Texto
CADENA DB 'Esta es una cadena$'
ARREGLO DB 10 dup(0)        ; Arreglo de 10 posiciones relleno de ceros
Opción DB 'x'               ; Variable opción inicializada con letra 'x'
Casilla: DB ?               ; Variable Casilla sin inicializar
Pos dB 0,0                  ; Arreglo Pos de dos elementos
                                    ; inicializada en [0,0]

.CODE                       ;Código del programa

; **********   Texto de presentación ****
```

```
MOV AX,@DATA                ;Cargar la direccion de datos
MOV DS, AX                  ;Cargar la direccion en el segmento de datos
LEA DX,TEXTO1               ;Carga el texto

MOV AH,9h                   ;imprimir en pantalla
INT 21h
                            ;Llama al sistema operativo

;  ******* AQUI SE INICIA EL PROGRAMA *******

; *** llenar el arreglo con FFh ***

        MOV CX,10           ; 10 ciclos (tamaño de ARREGLO)
        LEA DI, ARREGLO     ; obtiene la direccion de ARREGLO
                            ; (apunta al primer elemento)
        MOV AH,0FFh         ;          valor

llenandoFF:                 ; Inicio del lazo llenadoFF

        MOV [DI], AH        ; poner valor en la posición actual
        INC DI              ; mover DI a la siguiente posicion
        LOOP llenandoFF     ; repetir desde llenadoFF

; *** llenar el arreglo con valores ***

        MOV CX,10           ; 10 ciclos (tamaño de ARREGLO)
        LEA DI, ARREGLO     ; obtiene la direccion de ARREGLO
        MOV AH,0h           ; valor inicial

llenando:                   ; Inicio del lazo llenado

        MOV [DI], AH        ; poner valor en la posicion actual
```

```
        INC DI                  ; mover DI a la siguiente posicion
        INC AH
        LOOP llenando                   ; repetir desde llenado



; *** llenar el arreglo con valores de otro arreglo ***
; *** Copia los primeros 10 caracteres de CADENA en ARREGLO

        MOV CX,10               ; 10 ciclos (tamaño de ARREGLO)
        LEA DI, ARREGLO     ; obtiene la dirección de ARREGLO
                                ; (Destino de los datos)
    LEA SI, CADENA      ; obtiene dirección de cadena
                ; (Fuente de los datos)

Pasando:                        ; Inicio del lazo Pasando

        MOV AH,[SI]             ; copiar en AH el valor CADENA
                                ; en la posición actual
        MOV [DI], AH            ; poner valor en la posición actual
        INC DI                  ; mover DI a la siguiente posicion
        INC SI                  ; mover SI a la siguiente posicion
        LOOP Pasando                    ; repetir desde Pasando

;  ******* AQUI SE FINALIZA EL PROGRAMA *******

MOV AX,4C00h            ; Funcion 4ch
INT 21h

END                 ; Fin del programa
```

Programa 1

Con los comandos de control de flujo y el direccionamiento de vectores se pueden hacer funciones de búsqueda o de sustitución de valores o caracteres. En el siguiente programa se desarrollan tres tareas sobre una cadena:

- Contar las veces que se repite un caracter definido.
- Devolver la posición de la primer ocurrencia de un caracter en una cadena.
- Sustituir un caracter por otro en cada ocurrencia de la cadena.

Note que en la segunda tarea no se necesita recorrer toda la cadena, una vez que obtiene la posición de la primer ocurrencia del caracter se salta el *loop*. Note además el uso de caracteres ASCII extendido para imprimir en la pantalla vocales con acento.

```
.MODEL SMALL                ;Modelo pequeño

.STACK 100h                 ;Segmento de pila 256 posiciones

CR EQU 13                   ;Retorno de carro
LF EQU 10                   ;nueva linea

.DATA                       ;Segmento de datos

; ** uso de caracteres extendidos: 163-> 'u', 162->'ó'
TEXTO1 DB 'B',163,'squeda y Sustituci',162,'n $' ;Texto
CADENA DB 'Esta es una cadena$'
ARREGLO DB 10 dup(0)        ; Arreglo de 10 posiciones relleno de ceros
Opción DB 'x'               ; Variable opcion inicializada con letra 'x'
Casilla: DB ?               ; Variable Casilla sin inicializar
Pos dB 0,0                  ; Arreglo Pos de dos elementos
                            ; inicializada en [0,0]

.CODE                       ;Código del programa

; **********  Texto de presentacion ****
MOV AX,@DATA                ;Cargar la dirección de datos
```

```
MOV DS, AX                  ;Cargar la direccion en el segmento de datos
LEA DX,TEXTO1               ;Carga el texto

MOV AH,9h                   ;imprimir en pantalla
INT 21h
                            ;Llama al sistema operativo


;  ******* AQUI SE INICIA EL PROGRAMA *******

; 1) *** contar cuantas 'a' tiene la cadena ***

        MOV BH,'a'          ; Letra a buscar
        MOV CX,18           ; 10 ciclos (tamaño de CADENA)
        LEA SI,CADENA               ; obtiene la direccion de CADENA
        MOV AL,0            ; Iniciamos el conteo en cero

conteo:                     ; Inicio del conteo

        MOV AH, [SI]        ; poner valor en la posicion actual
        CMP AH,BH           ; Compara el valor actual con 'a'
        JNE noes            ; salta a noes si no son iguales
        INC AL                      ; caso contrario aumenta el conteo
noes:
        INC SI              ; mover SI a la siguiente posicion
        LOOP conteo         ; repetir desde conteo
; * En este momento AL tendrá en número de 'a' en la cadena (4)


; 2) *** encontrar la primer ocurrencia de 'u' ***
; al final AL tiene la posición de la primer ocurrencia
; o cero si no se encontró
```

```
        MOV BH,'u'          ; Letra a buscar
        MOV CX,18           ; 10 ciclos (tamaño de CADENA)
        LEA SI,CADENA               ; obtiene la dirección de CADENA
        MOV AL,0            ; iniciamos con 0 indicando
    MOV DL,1        ; posición inicial
                            ; no encontrado

search:                             ; Inicio de la busqueda

        MOV AH, [SI]        ; poner valor en la posicion actual
        CMP AH,BH           ; Compara el valor actual con 'u'
        JE found            ; salta a found si son iguales
        INC DL                      ; aumentar AL
        INC SI              ; mover SI a la siguiente posicion
        LOOP search         ; repetir desde search
        JMP fin                     ; saltar a fin (no incrementa AL)
found:
        MOV AL,DL           ; Pone en AL la posicion actual
                              ; que es la primer ocurrencia 'u'
                              ; y sale del loop
fin:

; * En este momento AL tendrá un 9


; 3) *** Sustituir 'e' por 'E' en la cadena***

        MOV BH,'e'          ; Letra a buscar
        MOV CX,18           ; 10 ciclos (tamaño de CADENA)
        LEA SI,CADENA               ; obtiene la dirección de CADENA
        LEA DI, CADENA              ; el destino es CADENA
```

```
susti:                          ; Inicio de la búsqueda

        MOV AH, [SI]            ; poner valor en la posicion actual
        CMP AH,BH               ; Compara el valor actual con 'e'
        JNE salta               ; salta si no es 'e'
        MOV [DI],'E'            ; sustituye por E si es 'e'
salta:
        INC SI                  ; mover SI a la siguiente posicion
        INC DI                  ; siguiente posición destino
        LOOP susti              ; repetir desde susti

; en este momento CADENA tiene:
; 'Esta Es una cadEna$'

;  ******* AQUI SE FINALIZA EL PROGRAMA *******

MOV Ax,4c00h                    ;Función 4ch
INT 21h

END                         ;Fin del programa
```

Programa 2