INSTRUCCIONES DE CADENA

Las instrucciones de manejo de cadenas difieren de las demás instrucciones del 8086 en que pueden accesar memoria e incrementar y decrementar un registro apuntador en una sola instrucción.

Como su nombre implica, las instrucciones de cadena son particularmente útiles en la manipulación de cadenas de caracteres. Son también adecuadas para el manejo de arreglos, búfers de datos y todo tipo de cadenas de bytes y palabras. Además, generan menos códigos y son más rápidas que las combinaciones equivalentes de instrucciones normales del 8086, tal como LOOP, INC y MOV.

Las instrucciones de cadena pueden agruparse en los siguientes grupos funcionales: instrucciones usadas para el movimiento de datos (LODS, STOS y MOVS) e instrucciones de cadena para inspección y comparación de datos (SCAS y CMPS).

Instrucción MOVSn

MOVS mueve un byte, palabra o palabra doble desde una localidad en memoria a otra.
Se carga la dirección de los operandos en los registros DI (Apuntador a la cadena receptora) y SI (Apuntador a la cadena emisora)

Operación	Instrucción Básica	Operandos
Mover un byte	MOVSB	ES:DI, DS:SI
Mover una palabra	MOVSW	ES:DI, DS:SI
Mover una palabra doble	MOVSD	ES:DI, DS:SI

No se codifican los operandos, pero la instrucción decrementa o incrementa a DI y SI en 1 para un byte, en 2 para una palabra y en 4 para una palabra doble, dependiendo de la bandera de dirección, para definir el estado de la bandera de dirección (DF) se usan las instrucciones:

CLD DF=0 Recorrido de Izq a Der , incrementa a los apuntadores
STD DF=1 Recorrido de Der a Izq, decrementa a los apuntadores

Para realizar el movimiento repetido de un localidad de memoria a otra, se necesita de la ayuda del prefijo REP antes de la instrucción MOVSn, la instrucción queda asi:

[etiqueta:] REP MOVSn

El prefijo REP proporciona un ejecución repetida con base a un contador inicial que se establece en Cx, haciendo que en cada iteración disminuya Cx y repite la operación hasta que Cx= 0.

Para hacer uso de esta instrucción se necesita que el programa .exe generado inicialice el registro ES (en general, pero no necesario) con la misma dirección de DS, algunas

otras instrucciones de cadena también lo requieren, estas son: STOS, CMPS, SCAS, para obtener esta alineación solo basta con añadir la siguiente línea al protocolo : MOV ES,AX.

La instrucción equivalente a la instrucción REP MOVSB son:

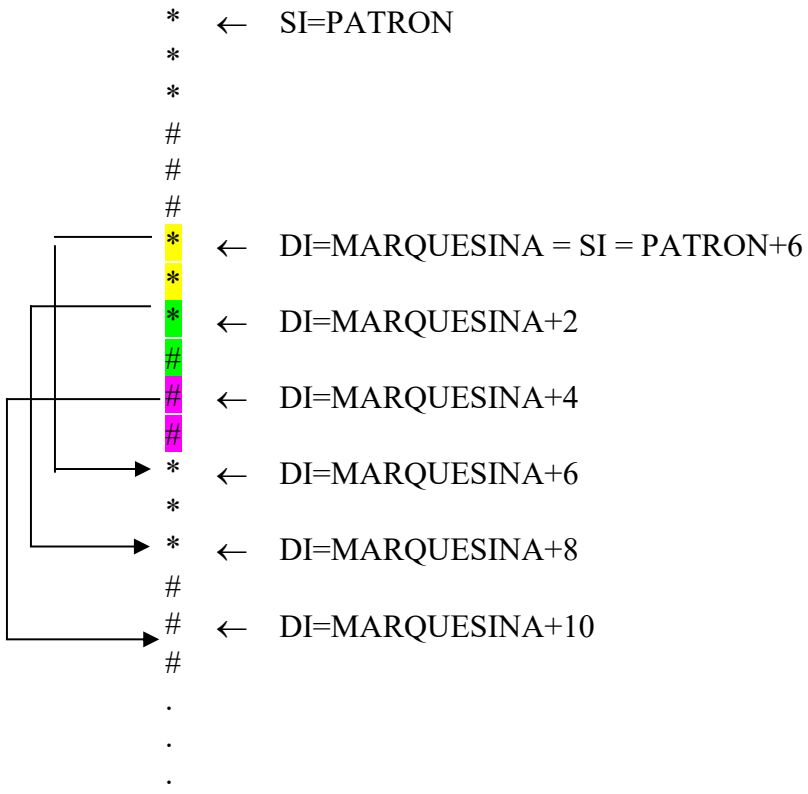
```
COMPARA: CMP CX,0
         JE SAL
         MOV AL,[SI]
         MOV [DI],AL
         INC DI
         INC SI
         DEC CX
         JMP COMPARA
SAL :    . . .
```

```
; COPIA UNA SUBCADENA DE LA CADENA1 A LA 2 DIRECCION IZQ A DER CON
;MOVSW

STACKSG SEGMENT PARA STACK 'STACK'
     DB      100H  DUP(0)
STACKSG ENDS
DATASG SEGMENT PARA 'DATA'

  CAD1      DB   'PROGRAMA QUE IMPRIME UN MENSAJE$'
  CAD2      DB   '                              $'
DATASG ENDS

CODESG SEGMENT PARA 'CODE'
PRINCI PROC FAR
        ASSUME SS:STACKSG, DS:DATASG, CS:CODESG
        PUSH DS
        SUB AX,AX
        PUSH AX
        MOV AX,SEG DATASG
        MOV DS,AX
        MOV ES,AX ; Linea que se añade para el uso de MOVS

      CLD         ;  BANDERA DE DIRECCION DE IZQ A DER
      MOV CX,5
      LEA DI,CAD2
      LEA SI,CAD1
      REP MOVSW
      LEA DX,CAD2
      CALL MENSAJE
      CALL LEE
      CALL FIN
PRINCI ENDP

MENSAJE PROC
PUSH AX
MOV AH,09H
INT 21H
POP AX
RET
MENSAJE ENDP

LEE PROC
PUSH AX
MOV AH,01
INT 21H
POP AX
RET
LEE ENDP

FIN PROC
MOV AH,4CH
INT 21H
RET
FIN ENDP

CODESG ENDS
END PRINCI
```

```
;Copia una subcadena de la cadena1 a la 2 usando MOVSB con
dirección    ;de derecha a izq

STACKSG SEGMENT PARA STACK 'STACK'
     DB      100H  DUP(0)
STACKSG ENDS
DATASG SEGMENT PARA 'DATA'

  CAD1      DB   'PROGRAMA QUE IMPRIME UN MENSAJE$'
  CAD2      DB   '                              $'
 DATASG ENDS

CODESG SEGMENT PARA 'CODE'
PRINCI PROC FAR
        ASSUME SS:STACKSG, DS:DATASG, CS:CODESG
        PUSH DS
        SUB AX,AX
        PUSH AX
        MOV AX,SEG DATASG
        MOV DS,AX
        MOV ES,AX ; LINEA QUE SE AÑADE AL PROTOCOLO


      STD   ;DIRECCIÓN DE DERECHA A IZQUIERDA
      MOV CX,10
      LEA DI,CAD2+9
      LEA SI,CAD1+9

      REP MOVSB
      LEA DX,CAD2
      CALL MENSAJE
      CALL LEE
      CALL FIN
PRINCI ENDP

MENSAJE PROC
PUSH AX
MOV AH,09H
INT 21H
POP AX
RET
MENSAJE ENDP

LEE PROC
PUSH AX
MOV AH,01
INT 21H
POP AX
RET
LEE ENDP

FIN PROC
MOV AH,4CH
INT 21H
RET
FIN ENDP

CODESG ENDS
END PRINCI
```

Repitiendo un patrón con MOVSW

La razón de esta repetición se debe a que ES=DS y al traslapar los segmentos los apuntadores se alinean uno inmediatamente después del otro.



```
;programa para tasm QUE REPITE UN PATRON 7 VECES USANDO MOVSW
; Definicion de stack

.MODEL small
.STACK  100

;DEFINICION DE AREAS DE TRABAJO
.DATA
patron DB '***###'
marquesina db 43 DUP(?)

.CODE
PRINCI PROC FAR
   ;PROTOCOLO
   push ds
   sub ax,ax
   push ax
   MOV AX,@DATA
   MOV DS,AX
   MOV ES,AX

   ;INICIA PROGRAMA

   CLD
   MOV  CX,21
   LEA DI,MARQUESINA
   LEA SI,PATRON
   REP MOVSW
   MOV AL, '$'
   MOV [DI],AL
   LEA DX,MARQUESINA
   MOV AH,09
   INT 21H
   mov ah,01
   int 21h
   RET

PRINCI ENDP

END PRINCI
```

Instrucción LODSn

LODS carga una cadena desde la memoria.
Carga desde memoria un byte en el AL, una palabra en AX o una palabra doble en el EAX.
La dirección de memoria esta sujeta a los registros DS:SI, dependiendo de la bandera de dirección se incrementa o decrementa.

Operación	Instrucción Básica	Operandos
Cargar un byte	LODSB	AL, DS:SI
Cargar una palabra	LODSW	AX, DS:SI
Cargar una palabra doble	LODSD	EAX, DS:SI

Instrucciones equivalentes:

```
MOV AL, [SI]
INC SI / DEC SI
```

Se pude utilizar para recorrer una cadena byte a byte, palabra por palabra o palabra doble por palabra doble, examinándola de forma sucesiva contra un valor particular.

```
;Carga de memoria los valores del dato tabla y los imprime en pantalla
;como caracteres. Uso de LODSB
STACKSG SEGMENT PARA STACK 'STACK'
     DB      100H  DUP(0)
STACKSG ENDS
DATASG SEGMENT PARA 'DATA'

  tabla      DB   1,2,3,4,5,6,7,8,9,10
  conta db 10

DATASG ENDS

CODESG SEGMENT PARA 'CODE'
PRINCI PROC FAR
        ASSUME SS:STACKSG, DS:DATASG, CS:CODESG
        PUSH DS
        SUB AX,AX
        PUSH AX
        MOV AX,SEG DATASG
        MOV DS,AX


CLD
MOV CL,0
MOV SI, OFFSET TABLA
OTRO: LODSB
MOV DL,AL               ; MOV AL,[SI], INC SI
ADD DL,48               ; SUMA PARA CONVERTIR DE BINARIO A ASCII= 30H
CALL ESCRIBE
INC CL
MOV DL,' '
CALL ESCRIBE
CMP CL,CONTA
JNE OTRO
CALL FIN
PRINCI ENDP

MENSAJE PROC
PUSH AX
MOV AH,09H
INT 21H
POP AX
RET
MENSAJE ENDP

LEE PROC
PUSH AX
MOV AH,01
INT 21H
POP AX
RET
LEE ENDP

ESCRIBE PROC
PUSH AX
MOV AH,02
INT 21H
POP AX
RET
```

```
ESCRIBE ENDP

FIN PROC
MOV AH,4CH
INT 21H
RET
FIN ENDP

CODESG ENDS
END PRINCI
```

```
; Programa ejemplo que muestra la cadena del 0 al 9 en forma inversa.
; Uso de LODSB
STACKSG SEGMENT PARA STACK 'STACK'
     DB      100H  DUP(0)
STACKSG ENDS
DATASG SEGMENT PARA 'DATA'

  tabla      DB   '0,1,2,3,4,5,6,7,8,9',13,10,09,'$'
  TABLAB DB 20 DUP(20H)
DATASG ENDS

CODESG SEGMENT PARA 'CODE'
PRINCI PROC FAR
        ASSUME SS:STACKSG, DS:DATASG, CS:CODESG
        PUSH DS
        SUB AX,AX
        PUSH AX
        MOV AX,SEG DATASG
        MOV DS,AX

MOV DX, OFFSET TABLA
CALL MENSAJE
CLD
MOV CX,13H
MOV SI, OFFSET TABLA
LEA DI, TABLAB+18       ;almacenando caracteres, salto de línea y
                              ;tabulador
REGR:   LODSB           ; MOV AL,[SI], INC SI
        MOV [DI],AL
        DEC DI
        LOOP REGR
;---------------------------------------------------------------
  ;DIFERENTES FORMAS PARA ASIGNAR EL DELIMITADOR $ A LA CADENA


     ;------ Direccionamiento indexado directo: DS+SI+cte
      MOV SI,19
      MOV AL, '$'
      MOV TABLAB[SI],AL            ;[SI+xxxx]



        ;-----Direccionamiento de base :DS+bx+cte
      ; MOV BH,00
        ; MOV BL,18                 ;12H
        ; MOV TABLAB[BX+1],'$'           ; [XXXX+BX+1]

      ;LEA BX, TABLAB
      ;MOV AL,'$'
      ;MOV [BX+19],AL               ;[XXXX+18]


       ;----- Direccionamiento indirecto: DS+SI
       ;LEA SI,TABLAB+19            ;13H
       ;MOV AL, '$'
       ;MOV [SI],AL
```

```
       ;LEA SI, TABLAB
       ;MOV AL, '$'
       ;MOV [SI+19], AL            ; [SI+19]
;-------------------------------------------------------------
        LEA DX,TABLAB
        CALL MENSAJE
        CALL LEE
CALL FIN
PRINCI ENDP

MENSAJE PROC
PUSH AX
MOV AH,09H
INT 21H
POP AX
RET
MENSAJE ENDP
```

Instrucción STOSn

Almacena una cadena de caracteres.
Almacena los contenidos del registro AL, AX y EAX en un byte, en una palabra o palabra doble en memoria respectivamente.
La dirección de memoria esta sujeta a los registros ES:DI, dependiendo de la bandera de dirección se incrementa o decrementa el registro DI en 1, 2 o 4 bytes.

Operación	Instrucción Básica	Operandos
Almacenar un byte	STOSB	ES:DI ,AL
Almacenar una palabra	STOSW	ES:DI ,AX
Almacenar una palabra doble	STOSD	ES:DI ,EAX

Para uso práctico de STOS se utiliza con un prefijo REP, esta instrucción ayuda a inicializar el área de datos a cualquier valor especificado, tal como limpiar el área de despliegue a blancos. Puede establecer el número de bytes, palabras o palabras dobles en CX.
Las instrucciones equivalentes a REP STOSB son:

```
COMPARA:  CMP CX,0
          JE SALTA
          MOV [DI], AL ; almacena   AL en memoria
          INC/ DEC DI
          DEC CX
          JMP COMPARA
SALTA:
```

```
;-------------------------------------------------------------------
-
;ALMACENA 10 CARACTERES BLANCOS A LA CADENA MEN1
;-------------------------------------------------------------------
-
.model small
.stack 100h
.data
    MEN1 DB 'ENSAMBLADOR$'

.code
  inicio proc far
     ;PROTOCOLO
      PUSH DS
      SUB AX,AX
      PUSH AX
      MOV AX,@data
```

```
      MOV DS,AX   ; OBSERVESE QUE DS=ES
      MOV ES,AX
   ;***********
     LEA DI,MEN1
     MOV DX,DI
     CALL MENSAJE
     CLD
     MOV AX,2020H
     MOV CX,05
     LEA DI,MEN1
     REP STOSW
     LEA DX,MEN1
     CALL MENSAJE
     CALL FIN
   INICIO endp

     FIN proc
       push ax
       mov ah,4ch
       int 21h
       pop ax
     FIN endp

     MENSAJE PROC
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
     MENSAJE ENDP

end  inicio
```

Instrucción CMPSn

Comparar cadenas. Compara el contenido de una localidad de memoria direccionada con DS:SI con el de otra localidad de memoria direccionada por ES:DI, dependiendo de la bandera de dirección incrementa o decrementa los registros SI, DI en 1, 2 o 4 bytes.
Solo se utiliza para comparaciones alfanuméricas, compara de acuerdo con valores ASCII, no es adecuada para operaciones algebraicas.

Operación	Instrucción Básica	Operandos
Comparar un byte	CMPSB	DS:SI, ES:DI
Comparar una palabra	CMPSW	DS:SI, ES:DI
Comparar una palabra doble	CMPSD	DS:SI, ES:DI

Comúnmente se utiliza con un prefijo: REPE (REPZ) o REPNE (REPNZ)

Instrucción REPE (REPZ)

Repite cuando sea igual, es decir, repite la operación mientras la bandera de cero (ZF) indique igual a 0. Se detiene cuando ZF ≠0 o cuando CX =0.

Instrucción REPNE (REPNZ)

Repite la operación mientras la bandera de cero (ZF) indique diferente a 0. Se detiene cuando ZF =0 o cuando CX =0.

```
 ;-------------------------------------------------------------------
--
; Programa que COMPARA CADENAS,DE DOS EN DOS, USO DE CMPSB Y CMPSW
;-------------------------------------------------------------------
-

.model small
.stack 100h
.data
    MEN1 DB 'ENSAMBLADOR$'
    MEN2 DB  'ENSAMBLADOR$'
    MEN3 DB  'Ensamblador$'
    MEN4 DB 'MEN1 ES IGUAL A MEN2$'
    MEN5 DB 'MEN1 ES IGUAL A MEN3$'
    MEN6 DB 'NO SON IGUALES$'
    SALTA DB 13,10,'$'
    BANDERA1 DB 0
    BANDERA2 DB 0
.code
  inicio proc far
     ;PROTOCOLO
      PUSH DS
      SUB AX,AX
      PUSH AX
      MOV AX,@data
      MOV DS,AX   ; DS=ES
      MOV ES,AX
   ;***********
      CLD
      MOV CX,06
      LEA DI,MEN1
      LEA SI,MEN2
      REPE CMPSW
      JNE OTRA           ;SALTA A COMPARAR CON LA OTRA CADENA
      MOV BANDERA1,01   ; BANDERA ACTIVADA
OTRA:  MOV CX,12
      LEA DI,MEN1
      LEA SI,MEN3
      REPE CMPSB
      JNE EDO1   ; SALTA A COMPARAR ESTADO DE LAS BANDERAS
      MOV BANDERA2,01
EDO1:  CMP BANDERA1,01
      JNE MAL1
      LEA DI,MEN4
      MOV DX,DI
      CALL MENSAJE
      LEA DX,SALTA
      CALL MENSAJE
```

```
      JMP EDO2
MAL1: LEA DI,MEN6
     MOV DX,DI
     CALL MENSAJE
     LEA DX,SALTA
     CALL MENSAJE
EDO2:  CMP BANDERA2,01
      JNE MAL2
      LEA DI,MEN5
      MOV DX,DI
      CALL MENSAJE
      JMP EXIT
MAL2: LEA DX,MEN6
     CALL MENSAJE
EXIT:     CALL FIN
   INICIO endp

     FIN proc
       push ax
       mov ah,4ch
       int 21h
       pop ax
     FIN endp

     MENSAJE PROC
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
     MENSAJE ENDP
end  inicio
```

Instrucción SCASn

Búsqueda en cadenas (Rastrear). Compara el contenido de la localidad de memoria direccionado por ES:DI con el contenido del registro AL, AX o EAX. Depende de la bandera de dirección para incrementar o decrementar DI en 1, 2 o 4 bytes.
Útil para aplicaciones de edición de texto, en la que el programa tiene que buscar signos de puntuación, como puntos, comas y blancos

Operación	Instrucción Básica	Operandos
Rastrear un byte	SCASB	ES:DI, AL
Rastrear una palabra	SCASW	ES:DI, AX
Rastrear una palabra doble	SCASD	ES:DI, EAX

```
;-------------------------------------------------------------------
-
; Busca el caracter 'a' en la cadena Ensamblador y la sustituye por
20h
;-------------------------------------------------------------------
-

.model small
.stack 100h
.data
    MEN1 DB 'ENSAmBLADOR$'
    .code
  inicio proc far
     ;PROTOCOLO
      PUSH DS
      SUB AX,AX
      PUSH AX
      MOV AX,@data
      MOV DS,AX   ; DS=ES
      MOV ES,AX
   ;***********
     CLD
     MOV AL,'A'
     MOV CX,11
     LEA DI,MEN1
     REPNE SCASB
     JNE EXIT
     DEC DI ; DECREMENTAR LA DIRECCION PARA HACER EL REMPLAZO
     MOV BYTE PTR[DI],20H   ;mover un byte a la localidad apuntada por
DI
     LEA DX,MEN1
     CALL MENSAJE
     CALL LEE
EXIT:CALL FIN
   INICIO endp

    LEE proc
       push ax
       mov ah,01h
       int 21h
       pop ax
```

```
       ret
     LEE endp

FIN PROC
       push ax
       mov ah,4ch
       int 21h
       pop ax
FIN ENDP

MENSAJE PROC
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
     MENSAJE ENDP
END INICIO
```

://www.programacion.com.py/escritorio/ensamblador/ejercicios-resueltos-en-ensamblador-8086