1.- Programa que inicializa una cadena con la letra ***A***. En el segmento de datos he creado dos variables de cadenas de caracteres: ***origen1*** y ***origen2*** para hacer la inicialización empleando instrucciones de cadena y sin ellas, respectivamente.

```
dosseg
.model small
.stack 100h
.data
  longitud equ 27
  origen1 db "esta es la cadena a iniciar"
  origen2 db "esta es la cadena a iniciar"
.code
inicio:
   mov ax, @data
   mov ds, ax
   mov es, ax

   ; Con instrucciones de cadena

   cld                ; se pone el flag de dirección a 0
   xor cx, cx
   mov cl, longitud
   lea di, origen1    ; ponemos en ES:DI la dirección de la cadena destino
   mov al, 'A'        ; Ponemos en AL la letra A.
   rep stosb          ; con rep repetimos la operación tantas veces como indique
                      ; el valor contenido en el registro CX



   ; Sin instrucciones de cadena

   xor si, si
   xor cx, cx
   mov cl, longitud
   mov al, 'A'
   Bucle:
     mov origen2[si], al
     inc si
     loop bucle
   Fin:
      mov ah, 4ch
      int 21h
end inicio
```

2.-Programa que busca un carácter en una cadena. En el segmento de datos he creado dos variables: la cadenas de caracteres: ***origen1*** y la variable ***caracter*** que contendrá el elemento a buscar. Se ha programado con instrucciones de cadena y sin ellas.

```
dosseg
.model small
.stack 100h
.data
  longitud equ 34
  origen1 db "esta es la cadena en la que buscar"
  caracter db 'q'
.code
inicio:
   mov ax, @data
   mov ds, ax
   mov es, ax

   ; Con instrucciones de cadena

   cld                ; se pone el flag de dirección a 0
   xor cx, cx
   mov cl, longitud
   mov al, caracter   ; ponesmos en AL el carácter a buscar
   lea di, origen1    ; ponemos en ES:DI la dirección de la cadena destino
   repnz scasb        ; con repnz repetimos la operación tantas veces como
                      ; indique el registro CX o cuando se active el flag Z al haber
                      ; encontrado coincidencia.


   ; Sin instrucciones de cadena

   xor si, si
   xor cx, cx
   mov cl, longitud
   mov al, caracter
   Bucle:
     cmp origen1[si], al
     jz fin
     inc si
     loop bucle
   Fin:
      mov ah, 4ch
      int 21h
end inicio
```

3.- Programa que compara dos cadenas. En el segmento de datos he creado dos variables de cadenas de caracteres: ***origen1*** y ***origen2*** que serán las cadenas a comparar. Se ha programado con instrucciones de cadena y sin ellas.

```
dosseg
.model small
.stack 100h
.data
  longitud equ 28
  origen1 db "esta es la cadena a comparar"
  origen2 db "esta es la cadena comparada."
.code
inicio:
   mov ax, @data
   mov ds, ax
   mov es, ax

   ; Con instrucciones de cadena

   cld                ; se pone el flag de dirección a 0
   xor cx, cx
   mov cl, longitud
   lea si, origen1    ; ponemos en DS:SI la dirección de la cadena origen
   lea di, origen2    ; ponemos en ES:DI la dirección de la cadena destino
   repz cmpsb         ; con repz repetimos la operación tantas veces como
                      ; indique el registro CX o cuando se desactive el flag Z al
                      ; no haber encontrado coincidencia.

   ; Sin instrucciones de cadena

   xor si, si
   xor cx, cx
   mov cl, longitud
   Bucle:
     mov al, origen1[si]
     cmp origen2[si], al
     jnz fin
     inc si
     loop bucle
   Fin:
      mov ah, 4ch
      int 21h
end inicio
```

4.- Programa que copia una cadena en otra. En el segmento de datos he creado tres variables de cadenas de caracteres: ***origen***, ***destino1*** y ***destino2*** que serán las cadenas a copiar en las otras dos. Se ha programado con instrucciones de cadena y sin ellas.

```
dosseg
.model small
.stack 100h
.data
  longitud equ 26
  origen db "esta es la cadena a copiar"
  destino1 db longitud dup(0)
  destino2 db longitud dup(0)
.code
inicio:
   mov ax, @data
   mov ds, ax
   mov es, ax

   ; Con instrucciones de cadena

   cld                ; se pone el flag de dirección a 0
   xor cx, cx
   mov cl, longitud
   lea si, origen     ; ponemos en DS:SI la dirección de la cadena origen
   lea di, destino1   ; ponemos en ES:DI la dirección de la cadena destino
   rep movsb          ; con rep repetimos la operación tantas veces como indique
                      ; el valor contenido en el registro CX



   ; Sin instrucciones de cadena

   xor si, si
   xor cx, cx
   mov cl, longitud
   Bucle:
     mov al, origen[si]
     mov destino2[si], al
     inc si
     loop bucle

   mov ah, 4ch
   int 21h
end inicio
```