## Desarrollar lo siguiente en su cuaderno o computadora:

1. Modifica el código de prueba para que imprima los caracteres en los siguientes incisos utilizando las instrucciones SHL, SHR, ROR y ROL (las 4 para cada inciso). Para cada caso documenta tus resultados. Para tu repositorio solo es necesario que subas los códigos del inciso a.

## a. A

```
1 → section .data
            char db 0 ; Variable para guardar el carácter a imprimir
newline db 10 ; Código ASCII para salto de línea (LF)
       char db 0
 4
 5 → section .text
 6 global _start ; Punto de entrada del programa
 7
 8 _start:
 9
10 ▼ ; === SHL (Shift Left) ===
mov al, 8 ; AL = 00001000 (8 en binario)

shl al, 3 ; Desplaza 3 bits a la izquierda: AL = 01000000 (64)

or al, 1 ; OR con 00000001 → AL = 01000001 (65, que es 'A')

mov [char], al ; Guarda el valor de AL en la variable 'char'

call print_char ; Llama a la subrutina para imprimir el carácter
16
17 → ; === SHR (Shift Right) ===
18 mov al, 130 ; AL = 10000010 (130 en binario)
19 shr al, 1 ; Desplaza 1 bit a la derecha: Al
       shr al, 1 ; Desplaza 1 bit a la derecha: AL = 01000001 (65 = 'A')
mov [char], al ; Guarda AL en 'char'
call print_char ; Imprime el carácter
20
21
22
23 - ; === ROL (Rotate Left) ===
mov al, 65 ; AL = 01000001 ('A')
rol al, 8 ; Rota 8 bits a la izquierda (vuelve al mismo valor)
mov [char], al ; Guarda el resultado en 'char'
call print_char ; Imprime el carácter
28
29 ▼ ; === ROR (Rotate Right) ===
30
           mov al, 130 ; AL = 10000010
          31
32
33
34

; Terminar el programa

mov eax, 1 ; syscall número 1 = exit

xor ebx, ebx ; Código de salida = 0

int 0x80 ; Llamada al sistema (salida del programa)

39
40 ; ===== Subrutina para imprimir un carácter con salto de línea =====
```

```
41 → print_char:
        mov eax, 4
                          ; syscall número 4 = write
42
        mov ebx, 1
43
                           ; descriptor 1 = salida estándar (pantalla)
                           ; puntero al carácter a imprimir
44
        mov ecx, char
                           ; número de bytes a imprimir = 1
        mov edx, 1
45
                           ; llamada al sistema para imprimir
46
        int 0x80
47
48
       ; imprimir salto de línea
                   ; syscall write
49
        mov eax, 4
50
                           ; stdout
       mov ebx, 1
                           ; dirección del salto de línea
51
        mov ecx, newline
                           ; 1 byte
52
        mov edx, 1
                           ; imprimir '\n'
53
        int 0x80
54
        ret
                           ; volver al código principal
```

## **b.** 0

```
1 → section .data
                          ; Variable donde se guarda el carácter
    char db 0
       newline db 10
                          ; Salto de línea (LF)
 4
 5 → section .text
 6 global _start
 7
 8 _start:
 9
10 → ; === SHL (Shift Left) ===
       mov al, 6 ; AL = 00000110
11
                          ; AL = 00110000 (48 = '0')
        shl al, 3
12
                          ; Guardar carácter
13
        mov [char], al
      call print_char
14
                          ; Imprimir
15
16 → ; === SHR (Shift Right) ===
       mov al, 192
                         ; AL = 11000000
17
                          ; AL = 00110000 (48 = '0')
18
       shr al, 2
19
        mov [char], al
20
       call print_char
22 - ; === ROL (Rotate Left) ===
       mov al, 12
                        ; AL = 00001100
24
       rol al, 2
                           ; AL = 00110000 (rota los bits a la izquierda)
25
        mov [char], al
26
      call print_char
27
28 - ; === ROR (Rotate Right) ===
                          ; AL = 00000011
        mov al, 3
        shl al, 6
30
                          ; AL = 11000000
        ror al, 2
                           ; AL = 00110000 (48 = '0')
31
32
       mov [char], al
33
     call print char
34
35
       ; Salir del programa
36
        mov eax, 1
37
        xor ebx, ebx
        int 0x80
38
39
```

```
41 → print_char:
                          ; syscall write
42
       mov eax, 4
                          ; stdout
       mov ebx, 1
43
                          ; dirección del carácter
44
       mov ecx, char
       mov edx, 1
                           ; longitud: 1 byte
45
       int 0x80
46
47
48
       ; salto de línea
49
       mov eax, 4
50
       mov ebx, 1
51
       mov ecx, newline
52
       mov edx, 1
53
       int 0x80
54
       ret
```

## c. g

```
1 → section .data
                           ; Variable donde se guarda el carácter
        char db 0
 3
       newline db 10
                           ; Salto de línea (LF)
 4
 5 → section .text
 6
    global start
 7
 8 _start:
 9
10 → ; === SHL (Shift Left) ===
        mov al, 13 ; AL = 00001101 (13)
11
        shl al, 3
12
                            ; AL = 01101000 (104)
13
        dec al
                            ; AL = 103 \rightarrow 'g'
        mov [char], al
14
15
       call print_char
16
17 ▼ ; === SHR (Shift Right) ===
        mov al, 206 ; AL = 11001110 (206)
19
        shr al, 1
                            ; AL = 01100111 (103 = 'g')
20
        mov [char], al
21
      call print_char
22
23 - ; === ROL (Rotate Left) ===
        mov al, 103
                       ; AL = 'g'
24
25
        rol al, 8
                            ; Rotación completa (valor queda igual)
26
        mov [char], al
27
        call print_char
28
29 ▼ ; === ROR (Rotate Right) ===
                          ; AL = 11001110
        mov al, 206
30
31
        ror al, 1
                           ; AL = 01100111 (103 = 'g')
32
        mov [char], al
33
       call print_char
34
       ; Salida del programa
35
36
        mov eax, 1
37
        xor ebx, ebx
38
        int 0x80
39
```

```
41 → print_char:
                         ; syscall write
42
        mov eax, 4
                          ; stdout
43
        mov ebx, 1
                          ; dirección del carácter
44
        mov ecx, char
45
        mov edx, 1
                          ; longitud: 1 byte
46
        int 0x80
47
48
      ; imprimir salto de línea
49
        mov eax, 4
        mov ebx, 1
50
        mov ecx, newline
51
52
        mov edx, 1
53
        int 0x80
54
        ret
```

```
1 → section .data
2 char db 0
                        ; Variable donde se guarda el carácter
      newline db 10 ; Salto de línea (LF)
3
4
5 → section .text
6 global start
7
8 _start:
9
10 ⋅ ; === SHL (Shift Left) ===
11
     mov al, 15 ; AL = 00001111 (15)
      shl al, 2
12
                        ; AL = 00111100 (60)
                         ; AL = 61 → '='
13
      inc al
    mov [char], al
call print_char
14
15
16
17 ▼ ; === SHR (Shift Right) ===
   mov al, 122 ; AL = 01111010 (122)
19
      shr al, 1
                         ; AL = 00111101 (61 = '=')
     mov [char], al
20
21
    call print_char
22
23 ▼ ; === ROL (Rotate Left) ===
24
     mov al, 61
                     ; AL = 00111101 ('=')
25
       rol al, 8
                         ; Rotar 8 bits (valor no cambia)
26
       mov [char], al
27
    call print_char
28
29 ▼ ; === ROR (Rotate Right) ===
      mov al, 122 ; AL = 01111010
30
      ror al, 1
31
                         ; AL = 00111101 (61 = '=')
32
      mov [char], al
33
    call print_char
34
35
      ; Salir del programa
36
      mov eax, 1
37
       xor ebx, ebx
       int 0x80
38
39
```

```
41 - print_char:
        mov eax, 4 ; syscall write
mov ebx, 1 ; stdout
mov ecx, char ; dirección del carácter
mov edx, 1 ; longitud: 1 byte
42
43
44
 45
46 int 0x80
 47
       ; salto de línea
mov eax, 4
mov ebx, 1
mov ecx, newline
 48
 49
 50
 51
         mov ecx, newline
 52
         mov edx, 1
 53
          int 0x80
54 ret
```