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

```
section .data
    char db 0           ; Variable para guardar el carácter a imprimir
    newline db 10       ; Código ASCII para salto de línea (LF)

section .text
    global _start       ; Punto de entrada del programa

_start:

; === 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

; === SHR (Shift Right) ===
    mov al, 130         ; AL = 10000010 (130 en binario)
    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

; === 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

; === ROR (Rotate Right) ===
    mov al, 130         ; AL = 10000010
    ror al, 1           ; Rota 1 bit a la derecha: AL = 01000001 ('A')
    mov [char], al      ; Guarda AL en 'char'
    call print_char     ; Imprime el carácter

    ; 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)

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

```
print_char:
    mov eax, 4          ; syscall número 4 = write
    mov ebx, 1          ; descriptor 1 = salida estándar (pantalla)
    mov ecx, char       ; puntero al carácter a imprimir
    mov edx, 1          ; número de bytes a imprimir = 1
    int 0x80            ; llamada al sistema para imprimir

    ; imprimir salto de línea
    mov eax, 4          ; syscall write
    mov ebx, 1          ; stdout
    mov ecx, newline    ; dirección del salto de línea
    mov edx, 1          ; 1 byte
    int 0x80            ; imprimir '\n'
    ret                 ; volver al código principal
```

b. 0

```
section .data
    char db 0           ; Variable donde se guarda el carácter
    newline db 10       ; Salto de línea (LF)

section .text
    global _start

_start:

; === SHL (Shift Left) ===
    mov al, 6           ; AL = 00000110
    shl al, 3           ; AL = 00110000 (48 = '0')
    mov [char], al      ; Guardar carácter
    call print_char     ; Imprimir

; === SHR (Shift Right) ===
    mov al, 192         ; AL = 11000000
    shr al, 2           ; AL = 00110000 (48 = '0')
    mov [char], al
    call print_char

; === ROL (Rotate Left) ===
    mov al, 12          ; AL = 00001100
    rol al, 2           ; AL = 00110000 (rota los bits a la izquierda)
    mov [char], al
    call print_char

; === ROR (Rotate Right) ===
    mov al, 3           ; AL = 00000011
    shl al, 6           ; AL = 11000000
    ror al, 2           ; AL = 00110000 (48 = '0')
    mov [char], al
    call print_char

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80

```

```
print_char:
    mov eax, 4          ; syscall write
    mov ebx, 1          ; stdout
    mov ecx, char       ; dirección del carácter
    mov edx, 1          ; longitud: 1 byte
    int 0x80

    ; salto de línea
    mov eax, 4
    mov ebx, 1
    mov ecx, newline
    mov edx, 1
    int 0x80
    ret
```

c. g

```
section .data
    char db 0           ; Variable donde se guarda el carácter
    newline db 10       ; Salto de línea (LF)

section .text
    global _start

_start:

; === SHL (Shift Left) ===
    mov al, 13          ; AL = 00001101 (13)
    shl al, 3           ; AL = 01101000 (104)
    dec al              ; AL = 103 → 'g'
    mov [char], al
    call print_char

; === SHR (Shift Right) ===
    mov al, 206         ; AL = 11001110 (206)
    shr al, 1           ; AL = 01100111 (103 = 'g')
    mov [char], al
    call print_char

; === ROL (Rotate Left) ===
    mov al, 103         ; AL = 'g'
    rol al, 8           ; Rotación completa (valor queda igual)
    mov [char], al
    call print_char

; === ROR (Rotate Right) ===
    mov al, 206         ; AL = 11001110
    ror al, 1           ; AL = 01100111 (103 = 'g')
    mov [char], al
    call print_char

    ; Salida del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80

```

```
print_char:
    mov eax, 4            ; syscall write
    mov ebx, 1            ; stdout
    mov ecx, char         ; dirección del carácter
    mov edx, 1            ; longitud: 1 byte
    int 0x80

    ; imprimir salto de línea
    mov eax, 4
    mov ebx, 1
    mov ecx, newline
    mov edx, 1
    int 0x80
    ret
```

d. =

```
section .data
    char db 0           ; Variable donde se guarda el carácter
    newline db 10       ; Salto de línea (LF)

section .text
    global _start

_start:

; === SHL (Shift Left) ===
    mov al, 15          ; AL = 00001111 (15)
    shl al, 2           ; AL = 00111100 (60)
    inc al              ; AL = 61 → '='
    mov [char], al
    call print_char

; === SHR (Shift Right) ===
    mov al, 122         ; AL = 01111010 (122)
    shr al, 1           ; AL = 00111101 (61 = '=')
    mov [char], al
    call print_char

; === ROL (Rotate Left) ===
    mov al, 61          ; AL = 00111101 ('=')
    rol al, 8           ; Rotar 8 bits (valor no cambia)
    mov [char], al
    call print_char

; === ROR (Rotate Right) ===
    mov al, 122         ; AL = 01111010
    ror al, 1           ; AL = 00111101 (61 = '=')
    mov [char], al
    call print_char

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80

```

```
print_char:
    mov eax, 4          ; syscall write
    mov ebx, 1          ; stdout
    mov ecx, char       ; dirección del carácter
    mov edx, 1          ; longitud: 1 byte
    int 0x80

    ; salto de línea
    mov eax, 4
    mov ebx, 1
    mov ecx, newline
    mov edx, 1
    int 0x80
    ret
```