Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Ingeniería en Software y Tecnologías Emergentes

Materia: Organización de Computadoras

Alumno: Jesus Eduardo Rodríguez Ramírez

Profesor: Jonatan Crespo Ragland

Grupo 932

Trabajo: CÓDIGOS Taller 8

Ensenada, B.C; a 29 de octubre del 2024

1. De acuerdo al código de prueba 1, responde y desarrolla lo siguiente:

a. Al correr el programa de código de prueba. Que imprime y por que? Documenta agregando líneas al código que prueba explicando su funcionamiento.

El código imprimirá "Resultado: N" debido a la conversión del valor 30 a su carácter ASCII correspondiente.

```
section .data
    num1 db 0b11001100 ; 204 en binario
    num2 db 0b10101010 ; 170 en binario
    result db 0
    message db "Resultado: ", 0

section .bss
    buffer resb 4

section .text
    global _start

_start:
    mov al, [num1]    ; Mueve el contenido de num1 al registro AL
    mov bl, [num2]    ; Mueve el contenido de num2 al registro BL
    and al, bl        ; Realiza la operación AND entre AL y BL, el resultado se almacena en AL
    mov [result], al  ; Mueve el resultado de AL a la variable result

    ; Imprimir "Resultado: "
    mov eax, 4        ; syscall número para sys_write
    mov ebx, 1        ; File descriptor 1 es STDOUT
    mov ecx, message  ; Dirección del mensaje
    mov edx, 11       ; Longitud del mensaje
    int 0x80          ; Llamada al kernel

    ; Convertir el resultado a una cadena ASCII para impresión
    mov eax, [result]
    add al, '0'       ; Convertir a carácter ASCII (asumiendo que el resultado es un solo dígito)
    mov [buffer], al

    ; Imprimir el resultado
    mov eax, 4        ; syscall número para sys_write
    mov ebx, 1        ; File descriptor 1 es STDOUT
    mov ecx, buffer   ; Dirección del buffer
    mov edx, 1        ; Longitud del buffer
    int 0x80          ; Llamada al kernel

    ; Salida del programa
    mov eax, 1        ; syscall número para sys_exit
    xor ebx, ebx      ; Código de salida 0
    int 0x80          ; Llamada al kernel
```

b. Investiga el funcionamiento de la instrucción lógica AND en ensamblador x86. Aplicala en un ejemplo de tu elección (puede ser en tu cuaderno o práctica). La instrucción lógica AND en ensamblador x86 realiza una operación lógica AND bit a bit entre dos operandos y almacena el resultado en el primer operando. La operación AND compara cada bit del primer operando con el bit correspondiente del segundo operando: si ambos bits son 1, el bit resultante es 1; de lo contrario, es 0**.**

```
section .data
msg db 'Resultado: ', 0    ; Mensaje que se mostrará antes del resultado
newline db 0xA             ; Carácter de nueva línea

section .bss
res resb 4                 ; Reserva de 4 bytes para almacenar el resultado

section .text
global _start
_start:
    ; Instrucciones aritméticas
    mov eax, 10            ; Coloca el valor 10 en el registro EAX
    mov ebx, 5             ; Coloca el valor 5 en el registro EBX
    add eax, ebx           ; Suma EAX + EBX y almacena el resultado en EAX (10 + 5 = 15)

    ; Instrucción lógica (AND)
    and eax, 0xF           ; Aplica una operación AND con 0xF (15 en decimal), deja EAX en 15

    ; Instrucciones de manipulación de bits
    shl eax, 1             ; Desplaza EAX un bit a la izquierda, equivalente a multiplicar por 2 (15 * 2 = 30)

    ; Guardar el resultado en la sección .bss
    mov [res], eax         ; Almacena el valor de EAX (30) en la variable 'res'

    ; Llamar a la rutina para imprimir el mensaje
    mov eax, 4             ; Syscall para escribir
    mov ebx, 1             ; Usar la salida estándar (pantalla)
    mov ecx, msg           ; Dirección del mensaje a imprimir
    mov edx, 11            ; Longitud del mensaje ('Resultado: ')
    int 0x80               ; Interrupción para imprimir el mensaje

    ; Imprimir el número (resultado almacenado en 'res')
    mov eax, [res]         ; Carga el valor almacenado en 'res' en EAX (30)

    add eax, '0'           ; Convierte el número a carácter ASCII sumando el valor de '0'
    mov [res], eax         ; Almacena el carácter convertido en 'res'

    mov eax, 4             ; Syscall para escribir
    mov ebx, 1             ; Usar la salida estándar
    mov ecx, res           ; Dirección del resultado a imprimir
    mov edx, 1             ; Longitud de 1 carácter
    int 0x80               ; Interrupción para imprimir el número

    ; Imprimir nueva línea
    mov eax, 4             ; Syscall para escribir
    mov ebx, 1             ; Usar la salida estándar
    mov ecx, newline       ; Dirección del carácter de nueva línea
    mov edx, 1             ; Longitud de 1 carácter
    int 0x80               ; Interrupción para imprimir nueva línea

    ; Terminar el programa
    mov eax, 1             ; Syscall para salir
    xor ebx, ebx           ; Código de salida 0
    int 0x80               ; Interrupción para terminar el programa
```

c. Investiga el funcionamiento de SHL y SHR y documenta.

La instrucción SHL (Shift Left) desplaza los bits del operando de destino hacia la izquierda por el número de bits especificado en el oper ando de cuenta. Los bits desplazados más allá del destino se desplaza n primero al flag de carry (CF). Los espacios vacíos se llenan con ceros durante la operación. Esta operación es equivalente a multiplica el operando por 2^n, donde n es el número de bits desplazados.

```
section .data
    msg db 'Resultado: ', 0
    newline db 0xA

section .bss
    res resb 4 ; Espacio para el resultado

section .text
    global _start

_start:
    ; Instrucciones aritméticas
    mov eax, 10
    mov ebx, 5
    add eax, ebx

    ; Instrucción lógica (AND)
    and eax, 0xF

    ; Instrucciones de manipulación de bits
    shl eax, 1

    ; Guardar el resultado en la sección .bss
    mov [res], eax

    ; Llamar a la rutina para imprimir el resultado
    mov eax, 4        ; Syscall para escribir
    mov ebx, 1        ; Usar la salida estándar (pantalla)
    mov ecx, msg      ; Dirección del mensaje a imprimir
    mov edx, 11       ; Longitud del mensaje
    int 0x80          ; Interrupción para imprimir el mensaje

    ; Imprimir el número (resultado almacenado en 'res')
    mov eax, [res]    ; Cargar el resultado en EAX
    add eax, '0'      ; Convertir el número en carácter (ASCII)
    mov [res], eax    ; Almacenar el carácter convertido
    mov eax, 4        ; Syscall para escribir
    mov ebx, 1        ; Usar la salida estándar
    mov ecx, res      ; Dirección del resultado
    mov edx, 1        ; Longitud de 1 carácter
    int 0x80          ; Interrupción para imprimir el número

    ; Imprimir nueva línea
    mov eax, 4        ; Syscall para escribir
    mov ebx, 1        ; Usar la salida estándar
    mov ecx, newline  ; Dirección de la nueva línea
    mov edx, 1        ; Longitud de 1 carácter
    int 0x80          ; Interrupción para imprimir nueva línea

    ; Terminar el programa
    mov eax, 1        ; Syscall para salir
    xor ebx, ebx      ; Código de salida 0
    int 0x80          ; Interrupción para terminar el programa
```

d. Modifica las instrucciones aritméticas, instrucciones lógicas y/o instrucciones de manipulación de bits en el programa de prueba para que ahora imprima lo siguiente (por partes): l (ele minúscula), D, B, 4 y 2. Describe paso a paso o en comentarios lo que se encuentra en los registros conforme avanza tu programa.

```
section .data
    newline db 0xA ; Nueva línea

section .bss
    res resb 1 ; Espacio para un carácter resultado

section .text
    global _start

_start:
    ; Inicialización de EAX con el valor ASCII de 'L'
    mov eax, '1'        ; EAX = 0x6C ('L')

    ; Imprimir 'L'
    mov [res], al       ; Guardar 'L' en res
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, res        ; Dirección del resultado
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir el carácter

    ; Imprimir 'D'
    mov al, 'D'         ; EAX = 0x44 ('D')
    mov [res], al       ; Guardar 'D' en res
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, res        ; Dirección del resultado
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir el carácter

    ; Imprimir 'B'
    mov al, 'B'         ; EAX = 0x42 ('B')
    mov [res], al       ; Guardar 'B' en res
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, res        ; Dirección del resultado
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir el carácter

    ; Imprimir '4'
    mov al, '4'         ; EAX = 0x34 ('4')
    mov [res], al       ; Guardar '4' en res
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, res        ; Dirección del resultado
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir el carácter

    ; Imprimir '2'
    mov al, '2'         ; EAX = 0x32 ('2')
    mov [res], al       ; Guardar '2' en res
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, res        ; Dirección del resultado
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir el carácter

    ; Imprimir nueva línea al final
    mov eax, 4          ; Syscall para escribir
    mov ebx, 1          ; Usar la salida estándar (pantalla)
    mov ecx, newline    ; Dirección de la nueva línea
    mov edx, 1          ; Longitud de 1 carácter
    int 0x80            ; Interrupción para imprimir nueva línea

    ; Terminar el programa
    mov eax, 1          ; Syscall para salir
    xor ebx, ebx        ; Código de salida 0
    int 0x80            ; Interrupción para terminar el programa
```