



**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.

```
1 section .data
2     num1 db 0b11001100 ; 204 en binario
3     num2 db 0b10101010 ; 170 en binario
4     result db 0
5     message db "Resultado: ", 0
6
7 section .bss
8     buffer resb 4
9
10 section .text
11     global _start
12
13 _start:
14     mov al, [num1] ; Mueve el contenido de num1 al registro AL
15     mov bl, [num2] ; Mueve el contenido de num2 al registro BL
16     and al, bl ; Realiza la operación AND entre AL y BL, el resultado se almacena en AL
17     mov [result], al ; Mueve el resultado de AL a la variable result
18
19     ; Imprimir "Resultado: "
20     mov eax, 4 ; syscall número para sys_write
21     mov ebx, 1 ; File descriptor 1 es STDOUT
22     mov ecx, message ; Dirección del mensaje
23     mov edx, 11 ; Longitud del mensaje
24     int 0x80 ; Llamada al kernel
25
26     ; Convertir el resultado a una cadena ASCII para impresión
27     mov eax, [result]
28     add al, '0' ; Convertir a carácter ASCII (asumiendo que el resultado es un solo dígito)
29     mov [buffer], al
30
31     ; Imprimir el resultado
32     mov eax, 4 ; syscall número para sys_write
33     mov ebx, 1 ; File descriptor 1 es STDOUT
34     mov ecx, buffer ; Dirección del buffer
35     mov edx, 1 ; Longitud del buffer
36     int 0x80 ; Llamada al kernel
37
38     ; Salida del programa
39     mov eax, 1 ; syscall número para sys_exit
40     xor ebx, ebx ; Código de salida 0
41     int 0x80 ; Llamada al kernel
42
```

b. Investiga el funcionamiento de la instrucción lógica AND en ensamblador x86. Apicala 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.

```

1  section .data
2  msg db 'Resultado: ', 0      ; Mensaje que se mostrará antes del resultado
3  newline db 0xA              ; Carácter de nueva línea
4
5  section .bss
6  res resb 4                  ; Reserva de 4 bytes para almacenar el resultado
7
8  section .text
9  global _start
10 _start:
11     ; Instrucciones aritméticas
12     mov eax, 10               ; Coloca el valor 10 en el registro EAX
13     mov ebx, 5                ; Coloca el valor 5 en el registro EBX
14     add eax, ebx              ; Suma EAX + EBX y almacena el resultado en EAX (10 + 5 = 15)
15
16     ; Instrucción Lógica (AND)
17     and eax, 0xF              ; Aplica una operación AND con 0xF (15 en decimal), deja EAX en 15
18
19     ; Instrucciones de manipulación de bits
20     shl eax, 1                ; Desplaza EAX un bit a la izquierda, equivalente a multiplicar por 2 (15 * 2 = 30)
21
22     ; Guardar el resultado en la sección .bss
23     mov [res], eax            ; Almacena el valor de EAX (30) en la variable 'res'
24
25     ; Llamar a la rutina para imprimir el mensaje
26     mov eax, 4                ; Syscall para escribir
27     mov ebx, 1                ; Usar la salida estándar (pantalla)
28     mov ecx, msg              ; Dirección del mensaje a imprimir
29     mov edx, 11               ; Longitud del mensaje ('Resultado: ')
30     int 0x80                  ; Interrupción para imprimir el mensaje
31
32     ; Imprimir el número (resultado almacenado en 'res')
33     mov eax, [res]            ; Carga el valor almacenado en 'res' en EAX (30)
34
35     add eax, '0'               ; Convierte el número a carácter ASCII sumando el valor de '0'
36     mov [res], eax            ; Almacena el carácter convertido en 'res'
37
38     mov eax, 4                ; Syscall para escribir
39     mov ebx, 1                ; Usar la salida estándar
40     mov ecx, res              ; Dirección del resultado a imprimir
41     mov edx, 1                ; Longitud de 1 carácter
42     int 0x80                  ; Interrupción para imprimir el número
43
44     ; Imprimir nueva línea
45     mov eax, 4                ; Syscall para escribir
46     mov ebx, 1                ; Usar la salida estándar
47     mov ecx, newline          ; Dirección del carácter de nueva línea
48     mov edx, 1                ; Longitud de 1 carácter
49     int 0x80                  ; Interrupción para imprimir nueva línea
50
51     ; Terminar el programa
52     mov eax, 1                ; Syscall para salir
53     xor ebx, ebx              ; Código de salida 0
54     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 operando de cuenta. Los bits desplazados más allá del destino se desplazan primero al flag de carry (CF). Los espacios vacíos se llenan con ceros durante la operación. Esta operación es equivalente a multiplicar el operando por 2^n , donde n es el número de bits desplazados.

```
1 section .data
2     msg db 'Resultado: ', 0
3     newline db 0xA
4
5 section .bss
6     res resb 4 ; Espacio para el resultado
7
8 section .text
9     global _start
10
11 _start:
12     ; Instrucciones aritméticas
13     mov eax, 10
14     mov ebx, 5
15     add eax, ebx
16
17     ; Instrucción Lógica (AND)
18     and eax, 0xF
19
20     ; Instrucciones de manipulación de bits
21     shl eax, 1
22
23     ; Guardar el resultado en la sección .bss
24     mov [res], eax
25
26     ; Llamar a la rutina para imprimir el resultado
27     mov eax, 4 ; Syscall para escribir
28     mov ebx, 1 ; Usar la salida estándar (pantalla)
29     mov ecx, msg ; Dirección del mensaje a imprimir
30     mov edx, 11 ; Longitud del mensaje
31     int 0x80 ; Interrupción para imprimir el mensaje
32
33     ; Imprimir el número (resultado almacenado en 'res')
34     mov eax, [res] ; Cargar el resultado en EAX
35     add eax, '0' ; Convertir el número en carácter (ASCII)
36     mov [res], eax ; Almacenar el carácter convertido
37     mov eax, 4 ; Syscall para escribir
38     mov ebx, 1 ; Usar la salida estándar
39     mov ecx, res ; Dirección del resultado
40     mov edx, 1 ; Longitud de 1 carácter
41     int 0x80 ; Interrupción para imprimir el número
42
43     ; Imprimir nueva línea
44     mov eax, 4 ; Syscall para escribir
45     mov ebx, 1 ; Usar la salida estándar
46     mov ecx, newline ; Dirección de la nueva línea
47     mov edx, 1 ; Longitud de 1 carácter
48     int 0x80 ; Interrupción para imprimir nueva línea
49
50     ; Terminar el programa
51     mov eax, 1 ; Syscall para salir
52     xor ebx, ebx ; Código de salida 0
53     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.

```

1 * section .data
2 | newline db 0xA ; Nueva línea
3
4 * section .bss
5 | res resb 1 ; Espacio para un carácter resultado
6
7 * section .text
8 | global _start
9
10 * _start:
11 | ; Inicialización de EAX con el valor ASCII de 'l'
12 | mov eax, 'l' ; EAX = 0x6C ('l')
13
14 | ; Imprimir 'l'
15 | mov [res], al ; Guardar 'l' en res
16 | mov eax, 4 ; Syscall para escribir
17 | mov ebx, 1 ; Usar la salida estándar (pantalla)
18 | mov ecx, res ; Dirección del resultado
19 | mov edx, 1 ; Longitud de 1 carácter
20 | int 0x80 ; Interrupción para imprimir el carácter
21
22 | ; Imprimir 'D'
23 | mov al, 'D' ; EAX = 0x44 ('D')
24 | mov [res], al ; Guardar 'D' en res
25 | mov eax, 4 ; Syscall para escribir
26 | mov ebx, 1 ; Usar la salida estándar (pantalla)
27 | mov ecx, res ; Dirección del resultado
28 | mov edx, 1 ; Longitud de 1 carácter
29 | int 0x80 ; Interrupción para imprimir el carácter
30
31 | ; Imprimir 'B'
32 | mov al, 'B' ; EAX = 0x42 ('B')
33 | mov [res], al ; Guardar 'B' en res
34 | mov eax, 4 ; Syscall para escribir
35 | mov ebx, 1 ; Usar la salida estándar (pantalla)
36 | mov ecx, res ; Dirección del resultado
37 | mov edx, 1 ; Longitud de 1 carácter
38 | int 0x80 ; Interrupción para imprimir el carácter
39
40 | ; Imprimir '4'
41 | mov al, '4' ; EAX = 0x34 ('4')
42 | mov [res], al ; Guardar '4' en res
43 | mov eax, 4 ; Syscall para escribir
44 | mov ebx, 1 ; Usar la salida estándar (pantalla)
45 | mov ecx, res ; Dirección del resultado
46 | mov edx, 1 ; Longitud de 1 carácter
47 | int 0x80 ; Interrupción para imprimir el carácter
48
49 | ; Imprimir '2'
50 | mov al, '2' ; EAX = 0x32 ('2')
51 | mov [res], al ; Guardar '2' en res
52 | mov eax, 4 ; Syscall para escribir
53 | mov ebx, 1 ; Usar la salida estándar (pantalla)
54 | mov ecx, res ; Dirección del resultado
55 | mov edx, 1 ; Longitud de 1 carácter
56 | int 0x80 ; Interrupción para imprimir el carácter
57
58 | ; Imprimir nueva línea al final
59 | mov eax, 4 ; Syscall para escribir
60 | mov ebx, 1 ; Usar la salida estándar (pantalla)
61 | mov ecx, newline ; Dirección de la nueva línea
62 | mov edx, 1 ; Longitud de 1 carácter
63 | int 0x80 ; Interrupción para imprimir nueva línea
64
65 | ; Terminar el programa
66 | mov eax, 1 ; Syscall para salir
67 | xor ebx, ebx ; Código de salida 0
68 | int 0x80 ; Interrupción para terminar el programa
69

```