

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA  
FACULTAD DE INGENIERÍA ARQUITECTURA Y DISEÑO

INGENIERÍA EN SOFTWARE Y TECNOLOGÍAS EMERGENTES



ORGANIZACIÓN DE COMPUTADORAS

## Taller 8

ADRIAN BALDERAS ROSAS

Jonatan Crespo Ragland

## a. ¿Qué imprime el programa y por qué? (Con documentación en el código)

Este programa realiza algunas operaciones aritméticas, lógicas y de manipulación de bits. Aquí está el código con comentarios detallados:

OneCompiler

HelloWorld.asm42vu65djg

```
1 section .data
2 msg db 'Resultado: ', 0 ; Mensaje que se imprimirá
3 newline db 0xA ; Carácter de nueva línea (ASCII 0xA)
4
5 section .bss
6 res resb 4 ; Reserva 4 bytes para almacenar el resultado
7
8 section .text
9 global _start
10 _start:
11 ; Instrucciones aritméticas
12 mov eax, 10 ; Carga el valor 10 en el registro EAX
13 mov ebx, 5 ; Carga el valor 5 en el registro EBX
14 add eax, ebx ; Suma EAX y EBX, resultado: EAX = 15
15
16 ; Instrucción Lógica (AND)
17 and eax, 0xF ; Realiza AND lógico entre EAX y 0xF (15 en decimal)
18 ; Como EAX ya es 15, el resultado de AND es 15
19
20 ; Instrucciones de manipulación de bits
21 shl eax, 1 ; Realiza un desplazamiento lógico a la izquierda (SHL) en EAX por 1 bit
22 ; EAX = 15 << 1 (desplazar a la izquierda): resultado EAX = 30
23
24 ; Guardar el resultado en la sección .bss
25 mov [res], eax ; Almacena el valor de EAX (30) en la variable 'res'
26
27 ; Imprimir el mensaje "Resultado: "
28 mov eax, 4 ; Syscall número 4 para escribir
29 mov ebx, 1 ; Salida estándar (pantalla)
30 mov ecx, msg ; Dirección del mensaje a imprimir
31 mov edx, 11 ; Longitud del mensaje (11 caracteres)
32 int 0x80 ; Interrupción para imprimir el mensaje
33
34 ; Imprimir el número (resultado almacenado en 'res')
35 mov eax, [res] ; Carga el valor almacenado en 'res' (30) en EAX
36 add eax, '0' ; Convierte el número en su equivalente ASCII sumando el valor '0'
37 ; 30 + '0' = carácter con valor 30 en ASCII (no imprimible)
38
39 mov [res], eax ; Almacena el carácter ASCII en 'res'
40 mov eax, 4 ; Syscall número 4 para escribir
41 mov ebx, 1 ; Salida estándar
42 mov ecx, res ; Dirección del resultado
43 mov edx, 1 ; Longitud de 1 carácter
44 int 0x80 ; Interrupción para imprimir el carácter
45
46 ; Imprimir una nueva línea
47 mov eax, 4 ; Syscall número 4 para escribir
48 mov ebx, 1 ; Salida estándar
49 mov ecx, newline ; Dirección del carácter de nueva línea
50 mov edx, 1 ; Longitud de 1 carácter
51 int 0x80 ; Interrupción para imprimir nueva línea
52
53 ; Terminar el programa
54 mov eax, 1 ; Syscall para salir
55 xor ebx, ebx ; Código de salida 0
56 int 0x80 ; Interrupción para terminar el programa
57
```

STDIN

Input for the program

Output:

Resultado: N

## ¿Qué imprime?

El programa debería imprimir:

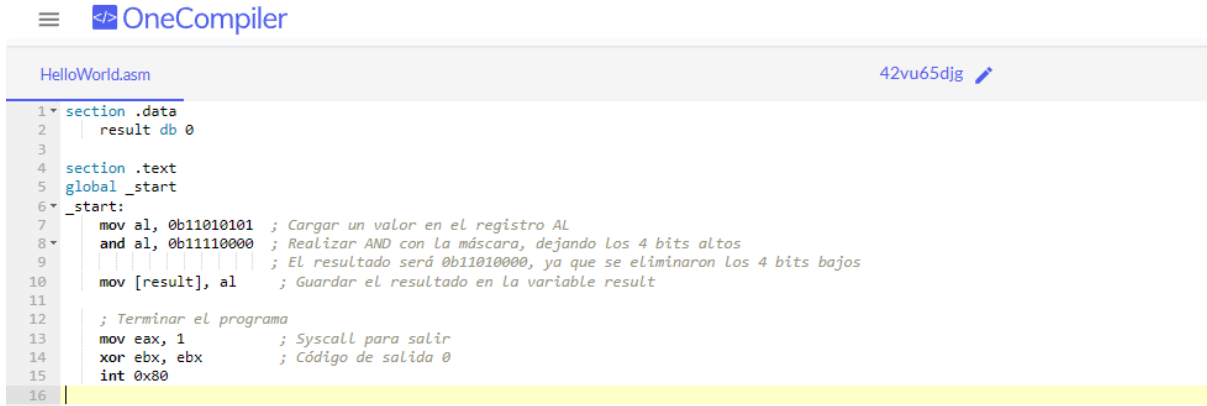
Resultado:

Luego intentará imprimir el carácter correspondiente al número 30 en ASCII. Sin embargo, el valor 30 en ASCII es un carácter de control que no es visible o imprimible. Entonces, aunque el programa parece estar funcionando, lo que verás será el texto "Resultado:" seguido de un carácter no imprimible.

## b. Instrucción lógica AND en ensamblador x86

La instrucción AND realiza una operación de "Y" lógica bit a bit entre dos operandos. El resultado es 1 solo si ambos bits correspondientes en los operandos son 1. Se

usa comúnmente para "enmascarar" bits, es decir, para poner a cero ciertos bits de un valor y dejar otros sin cambios.



```
1 section .data
2     result db 0
3
4 section .text
5 global _start
6 _start:
7     mov al, 0b11010101 ; Cargar un valor en el registro AL
8     and al, 0b11110000 ; Realizar AND con la máscara, dejando los 4 bits altos
9     ; El resultado será 0b11010000, ya que se eliminaron los 4 bits bajos
10    mov [result], al ; Guardar el resultado en la variable result
11
12    ; Terminar el programa
13    mov eax, 1 ; Syscall para salir
14    xor ebx, ebx ; Código de salida 0
15    int 0x80
16
```

En este caso, el valor de `al` pasa de `0b11010101` a `0b11010000` debido al AND con la máscara `0b11110000`.

### c. Instrucciones SHL y SHR

- **SHL (Shift Left):** Desplaza los bits de un operando a la izquierda por un número de posiciones. Los bits vacantes se llenan con ceros. Esto efectivamente multiplica el valor por 2 por cada desplazamiento a la izquierda.

`mov al, 3` ; Cargar el valor 3 en AL (0b00000011)

`shl al, 1` ; Desplaza los bits de AL a la izquierda 1 vez: resultado 6 (0b00000110)

**SHR (Shift Right):** Desplaza los bits de un operando a la derecha por un número de posiciones. Los bits vacantes a la izquierda se llenan con ceros. Esto divide el valor por 2 por cada desplazamiento a la derecha.

`mov al, 6` ; Cargar el valor 6 en AL (0b00000110)

`shr al, 1` ; Desplaza los bits de AL a la derecha 1 vez: resultado 3 (0b00000011)

### d. Modificación para imprimir los caracteres: 1, D, B, 4, 2

Aquí vamos a modificar las operaciones lógicas, aritméticas y de bits para que impriman los caracteres deseados.

Los valores ASCII de estos caracteres son:

- 'l' = 108
- 'D' = 68

- 'B' = 66
- '4' = 52
- '2' = 50

Modificación del programa:

OneCompiler

HelloWorld.asm

42vu65djg

```

1 section .data
2     msg db 'Resultado: ', 0
3     newline db 0xA
4
5 section .bss
6     res resb 1
7
8 section .text
9 global _start
10 _start:
11     ; Imprimir 'L' (ASCII 108)
12     mov al, 108      ; Cargar el valor ASCII de 'L'
13     mov [res], al    ; Guardar en res
14     mov eax, 4       ; syscall para escribir
15     mov ebx, 1       ; Usar salida estándar
16     mov ecx, res     ; Dirección del valor
17     mov edx, 1       ; Longitud de 1 carácter
18     int 0x80         ; Interrupción para imprimir
19
20     ; Imprimir 'D' (ASCII 68)
21     mov al, 68       ; Cargar el valor ASCII de 'D'
22     mov [res], al    ; Guardar en res
23     int 0x80         ; Imprimir
24
25     ; Imprimir 'B' (ASCII 66)
26     mov al, 66       ; Cargar el valor ASCII de 'B'
27     mov [res], al    ; Guardar en res
28     int 0x80         ; Imprimir
29
30     ; Imprimir '4' (ASCII 52)
31     mov al, 52       ; Cargar el valor ASCII de '4'
32     mov [res], al    ; Guardar en res
33     int 0x80         ; Imprimir
34
35     ; Imprimir '2' (ASCII 50)
36     mov al, 50       ; Cargar el valor ASCII de '2'
37     mov [res], al    ; Guardar en res
38     int 0x80         ; Imprimir
39
40     ; Imprimir nueva línea
41     mov eax, 4       ; syscall para escribir
42     mov ebx, 1       ; Usar salida estándar
43     mov ecx, newline ; Dirección de nueva línea
44     mov edx, 1       ; Longitud de 1 carácter
45     int 0x80         ; Imprimir nueva línea
46
47     ; Terminar el programa
48     mov eax, 1       ; syscall para salir
49     xor ebx, ebx     ; Código de salida 0
50     int 0x80         ; Interrupción para terminar el programa
51

```

## e. Alternativas para imprimir los caracteres

Para cada carácter, podrías modificar otras partes del código o utilizar otras instrucciones lógicas y de bits, como usar desplazamientos o ANDs para modificar los valores sin necesidad de cargar el ASCII directo.

Por ejemplo:

- Para 'l' (108),