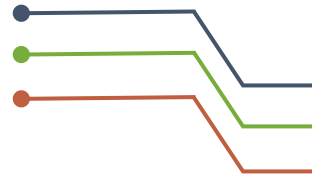
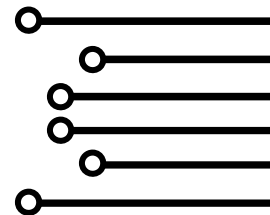




# C Embebido



# Operadores Bitwise



Para realizar operaciones nivel bit.



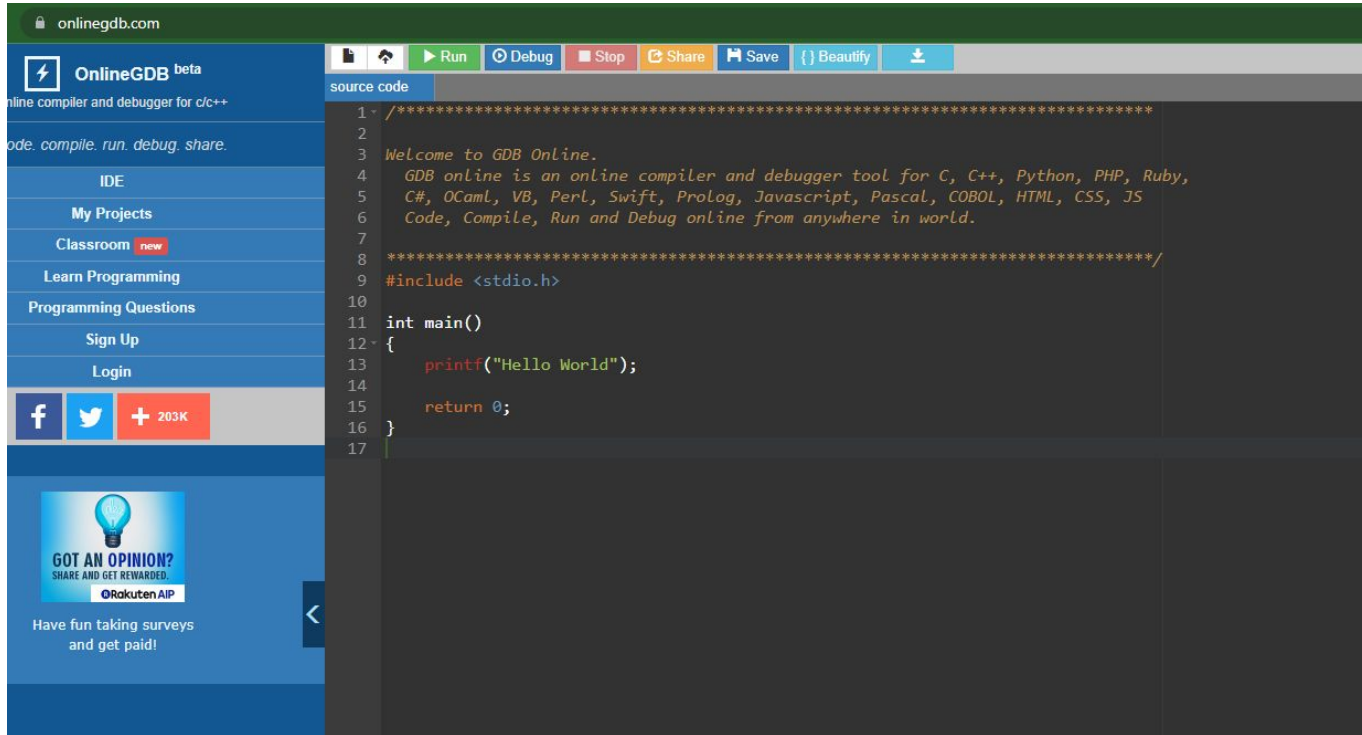
Recordaremos el Bitwise AND y OR. Y conoceremos el XOR y complementario.



También veremos el corrimiento

- Corrimiento a la derecha
- Corrimiento a la izquierda

Wels



The screenshot displays the OnlineGDB website interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (marked as new), Learn Programming, Programming Questions, Sign Up, and Login. Below these are social media icons for Facebook and Twitter, and a button to '+ 203K'. At the bottom of the sidebar is a Rakuten AIP advertisement. The main area shows the 'source code' editor with a C program. The program includes a welcome message and a simple 'Hello World' output.

```
1 - /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,  
5 C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```

# ONLINE GDB

# BITWISE

AND

OR

XOR

Comp

Operadores de bit a bit

# AND

- La salida del Bitwise AND es 1 si los 2 bits operando son 1.
- Si cualquier bit operando es 0, la salida es 0.

25 = 00011001

87 = 01010111

Operación de 25 AND 87

00011001

& 01010111

---

00010001 = 17

Signo: &

## AND

## Enmascaramiento

- Una máscara define cuáles bits quieres mantener y cuales quieres limpiar.

*25 = 00011001*

*Mantener los bits lsb y limpiar los msb*

*input     00011001*

*mask     & 00001111*

*00001001 = 9*

A large brown arrow pointing right with the word "OR" in white text inside it.

# OR

- La salida del Bitwise OR es 1 si sólo 1 bit operando es 1.

*25 = 00011001*

*87 = 01010111*

*Operación de 25 OR 87*

*00011001*

*/ 01010111*

*01011111 = 95*

**Signo: |**

# XOR

- La salida del Bitwise XOR es 1 si los 2 bits operandos son diferentes.
- Si son iguales los bits operandos la salida es 0.

25 = 00011001

87 = 01010111

Operación de 25 XOR 87

00011001

^ 01010111

---

01001110 = 78

Signo: ^



A green icon with the word 'COMP' in white, followed by two green chevrons pointing to the right.

## COMP

- El operador de Complemento bit a bit es un operador unario, sólo funciona en un operando.
- Cambia de 1 a 0 y de 0 a 1.

25 = 00011001

87 = 01010111

Operación de NOT 87

~ 01010111

---

10101000 = -88

Signo: ~

# Operadores Bitwise



Para realizar operaciones nivel bit.



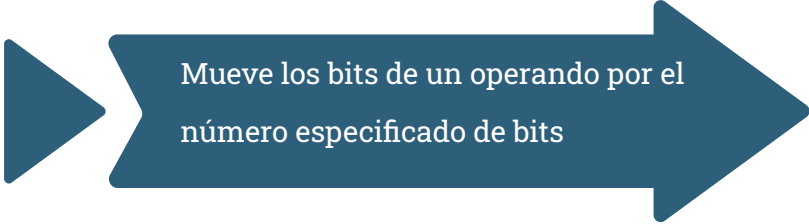
Recordaremos el Bitwise AND y OR. Y conoceremos el XOR y complementario.



También veremos el corrimiento

- Corrimiento a la derecha
- Corrimiento a la izquierda

# Corrimiento a la derecha



Mueve los bits de un operando por el número especificado de bits

operando  $\gg$  n°

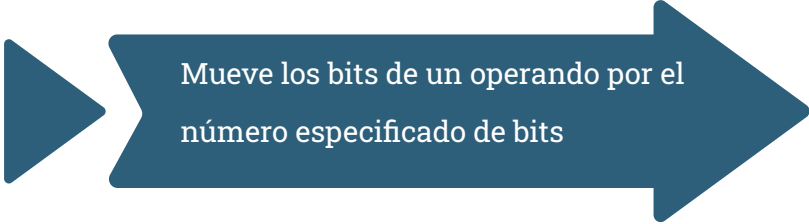
100 = 01100100

Shift right de 1 bit

input      01100100

$\gg 1$       00110010

# Corrimiento a la derecha



Mueve los bits de un operando por el número especificado de bits

operando  $\gg n^\circ$

Si es sin signo, en el corrimiento se rellena con cero.

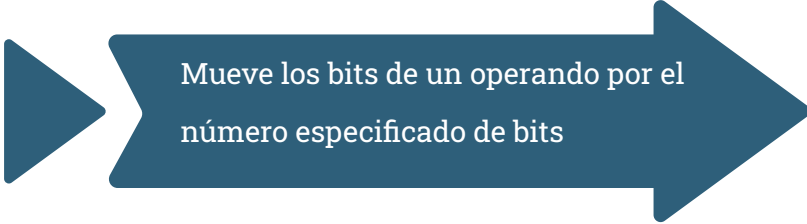
100 = 01100100

Shift right de 1 bit

input      01100100

$\gg 1$       00110010

# Corrimiento a la derecha



Mueve los bits de un operando por el número especificado de bits

operando  $\gg n^\circ$

*100 = 01100100*

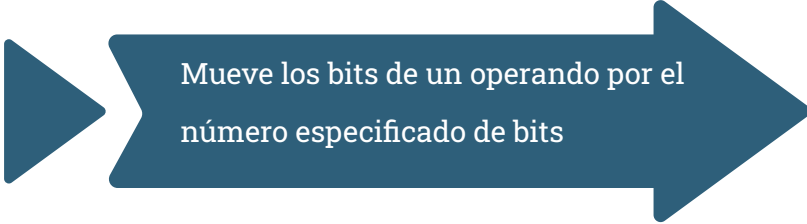
*Shift right de 1 bit*

*input      01100100*

*>>1      00110010*

Si es con signo, en el corrimiento se rellena con bit más significativo antes del turno.

# Corrimiento a la derecha



Mueve los bits de un operando por el número especificado de bits

operando >> n°

```
uint32_t variable = 100;  
uint32_t y = variable >> 4;  
printf("Resultado corrimiento derecho: %d \r\n",y);
```

# Corrimiento a la derecha

Mueve los bits de un operando por el número especificado de bits a la derecha.

$$y = x / 2^n$$



$$y = x \gg n$$

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

$10_{10}$

$\gg$

Right Shift

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

$5_{10}$

# Corrimiento a la izquierda

Mueve los bits de un operando por el número especificado de bits a la izquierda.

operando  $\ll$  n°

*-6 = 11111010*

*Shift Left de 2 bit  
input 11111010*

*$\ll 2$  11101000 = -24*



*Wels*

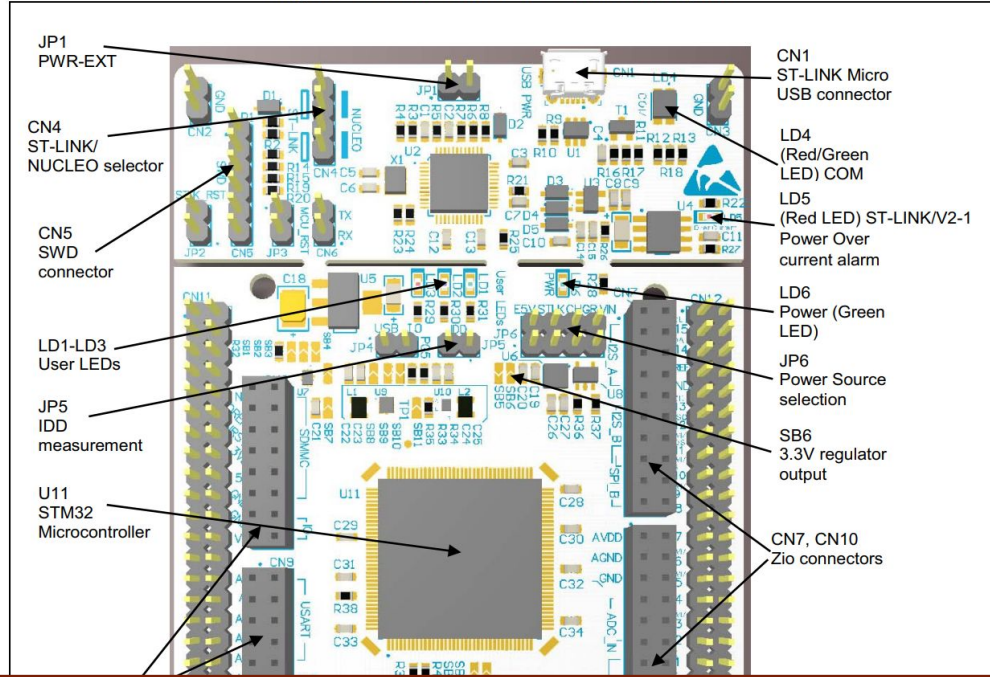


# STM32 CubeIDE



**USAREMOS EL STM32CubeIDE**

Figure 4. STM32 Nucleo-144 board top layout



LED1 - LED3

# GPIOB -> MODER

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

2 bits por cada Pin

**MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

PB0	46	PB0
PB1	47	PB1
PB2	48	PB2
PB3	133	PB3
PB4	134	PB4
PB5	135	PB5
PB6	136	PB6
PB7	137	PB7
PB8	139	PB8
PB9	140	PB9
PB10	69	PB10
PB11	70	PB11
PB12	73	PB12
PB13	74	PB13
PB14	75	PB14
PB15	76	PB15

## Clearing bit n

Es el resultado de aplicar **AND** al valor de una variable almacenada con el complemento (**NOT**)

```
*p_RegModemGPIOB &= 0xFFFFFFFF0;  
*p_RegModemGPIOB |= 0x00000001;
```

```
data &= ~(1 << n);
```

## Setting bit n

Es el resultado de aplicar **OR** al valor de una variable almacenada con corrimiento.

```
*p_RegModerGPIOB &= 0xFFFFFFFF0;  
*p_RegModerGPIOB |= 0x00000001;
```

```
data |= (1 << n);
```

## Flipping bit n

Es el resultado de aplicar **XOR** al valor de una variable almacenada con corrimiento.

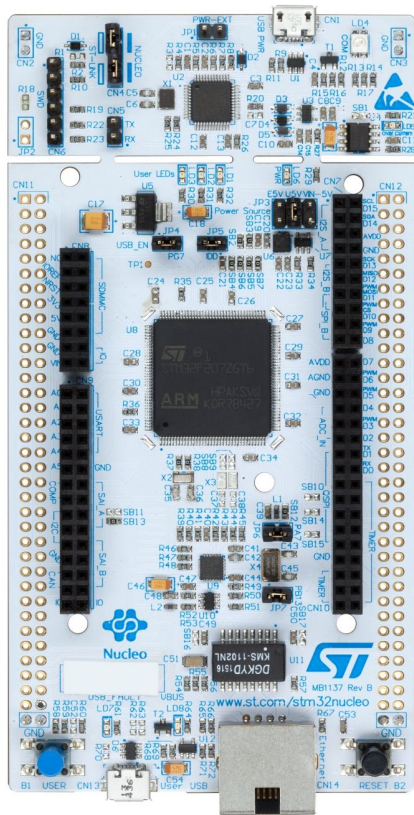
```
*p_RegOdrGPIOB |= 0X00000001;
```

```
data ^= (1 << n);
```

## Checking bit n

Es el resultado de aplicar **AND** al valor de una variable almacenada con corrimiento

```
data &= (1 << n);
```



# Gracias

@welsttheory

[hola@welsttheory.com](mailto:hola@welsttheory.com)

[www.welsttheory.com](http://www.welsttheory.com)