UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Operadores a nivel de bit

Basado en una presentación del
Ing. Marcelo Trujillo

➢ Operadores lógicos

operador	símbolo	operandos	Tipo de operación
✓ *and*	&	2	binaria
✓ *or*	|	2	binaria
✓ *xor*	^	2	binaria
✓ *not*	~	1	unario
✓ *shift izquierda*	<<	2	binaria
✓ *shift derecha*	>>	2	binaria

Las operaciones lógicas a nivel de bit and, or, xor, not, shift a izquierda y shift a derecha, se ejecutan sobre los bits individuales de los operandos intervinientes en las mismas.

➢ Operadores lógicos

- ✓ *La utilización de operadores a nivel de bit tiene sentido practico sobre los tipos de variables enteras*
- ✓ *Todos los ejemplos serán desarrollados sobre variables del tipo **unsigned char,** pero todo lo dicho será valido para cualquiera de los tipos de datos de variable entera.*

➢ *Operador lógico & (and)*

unsigned char A = 0x3b , B = 0x5d , C;

A	*B*	*A AND B*
0	0	0
0	1	0
1	0	0
1	1	1

A	0	0	1	1	1	0	1	1
B	0	1	0	1	1	1	0	1
C	0	0	0	1	1	0	0	1

A	*B*	*A AND B*
0	0	0
0	1	0
1	0	0
1	1	1

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **cero**

Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**

➢ *Operador lógico | (or)*

unsigned char A = 0x3b , B = 0x5d , C;

A	*B*	*A* OR *B*
0	0	0
0	1	1
1	0	1
1	1	1

A	0	0	1	1	1	0	1	1
B	0	1	0	1	1	1	0	1
C	0	1	1	1	1	1	1	1

A	*B*	*A* OR *B*
0	0	0
0	1	1
1	0	1
1	1	1

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**

Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **uno**

➢ Operador lógico ^ (xor)

unsigned char A = 0x3b , B = 0x5d , C;

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	0	0	1	1	1	0	1	1
B	0	1	0	1	1	1	0	1
C	0	1	1	0	0	1	1	0

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**

Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **su complemento**

➢ *Operador lógico ^ (xor)*

unsigned char A = 0x3b, C;

A	*B*	*A XOR B*
0	0	0
0	1	1
1	0	1
1	1	0

Pero además !!!!!!!

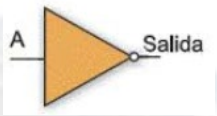
A	*B*	*A XOR B*
0	0	0
0	1	1
1	0	1
1	1	0

Mientras el primer operando sea igual a segundo
El resultado será **cero**

Mientras el primer op. sea distinto al segundo op.
El resultado será **uno**

➢ *Operador lógico ~ (not)*

unsigned char A = 0x3b , C;



A	NOT *A*
0	1
1	0

A	0	0	1	1	1	0	1	1
C	1	1	0	0	0	1	0	0

➢ *Operador lógico << (shift izquierda)*

unsigned char A = 0x3b;

➢ *Operador lógico << (shift izquierda)*

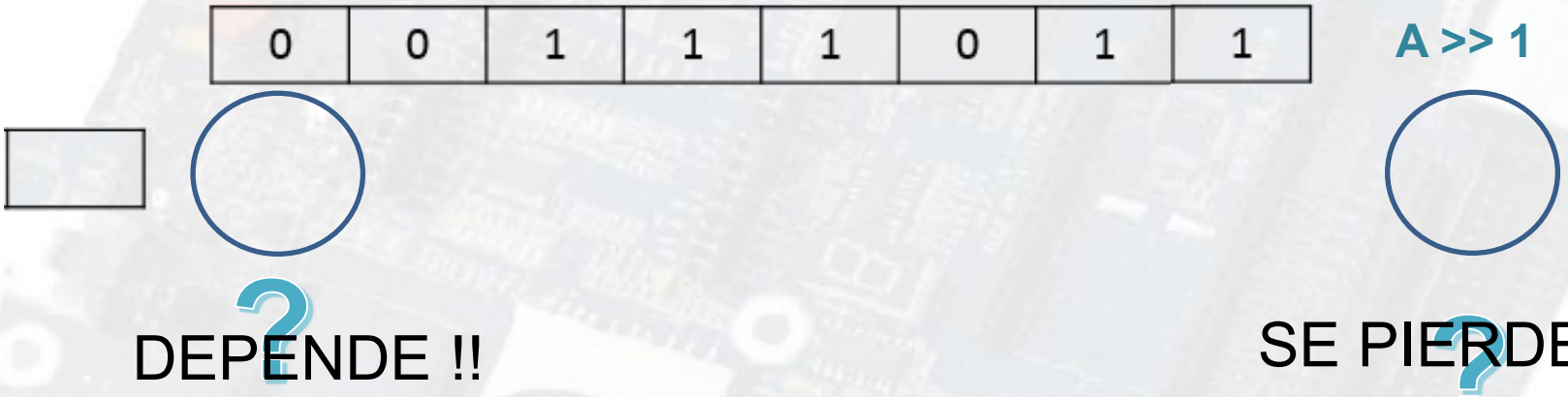
unsigned char A = 0x01;

	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	1	0	A << 1
4	0	0	0	0	0	1	0	0	A << 2
8	0	0	0	0	1	0	0	0	A << 3
16	0	0	0	1	0	0	0	0	A << 4
32	0	0	1	0	0	0	0	0	A << 5
64	0	1	0	0	0	0	0	0	A << 6
128	1	0	0	0	0	0	0	0	A << 7

Desplazar a la izquierda es multiplicar por dos

➢ *Operador lógico >>(shift derecha)*

unsigned char A = 0x3b;



Si la variable es sin signo se completa con ceros

Si la variable es con signo y su valor es positivo se completa con ceros

Si la variable es con signo y su valor es negativo se completa con unos

➢ Mascaras

unsigned char A = 0x3d, B = ? ,C;

&								
A	0	0	1	1	1	1	0	1
B	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**
C	0	0	1	1	0	1	0	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Mientras el primer operando sea igual a segundo
El resultado será **cero**

Mientras el primer op. sea distinto al segundo op.
El resultado será **uno**

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Garantiza CEROS

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **cero**
Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**
Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **uno**

Mascaras

unsigned char A = 0x3d, B = ? ,C;

A	0	0	1	1	1	1	0	1
B	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
C	0	1	1	1	1	1	0	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **cero**
Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Garantiza UNOS

Mientras el primer operando sea
igual al segundo
El resultado será **cero**
Mientras el primer op. sea distinto
El resultado será **uno**

Mientras el primer operando valga cero
No importa cuanto valga el segundo operando
El resultado será **el segundo operando**
Mientras el primer operando valga uno
No importa cuanto valga el segundo operando
El resultado será **uno**

➢ Mascaras

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

```
EstadoPrevio =
    xxxxxxx0
while ( 1 )
{
        EstadoActual =
        xxxxxxx0

        Alarma = EstadoActual  ^ EstadoPrevio ;
        xxxxxxx0

        EstadoPrevio = EstadoActual ;
        xxxxxxx0
        if (Alarma & 0x01) == 0x01 )
}
```

Lectura de Sensor

Lectura de Sensor

ALARMA

➢ Mascaras

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

```
while ( 1 )
{
    Motor = Motor & 0x01 ;
}
```