

# Representação de Dados

## Manipulação de Bits

Noemi Rodriguez  
Ana Lúcia de Moura  
Raúl Renteria  
Alexandre Meslin

<http://www.inf.puc-rio.br/~inf1018>

# Operações Lógicas em C

---

## Operadores **lógicos**: or (||), and (&&) e not (!)

- ✓ tratam qualquer argumento diferente de zero como **true** e igual a zero como **false**
- ✓ operação resulta em 1 (true) ou 0 (false)

<b>Expressão</b>	<b>Resultado</b>
<code>!0x41</code>	<code>0x00</code>
<code>!0x00</code>	<code>0x01</code>
<code>0xaa &amp;&amp; 0x55</code>	<code>0x01</code>
<code>0xaa    0x55</code>	<code>0x01</code>

# Operadores *bit a bit* em C

---

Podem ser aplicados a qualquer dos tipos **inteiros**

Baseados em álgebra booleana (**bit a bit**)

A & B	&	0	1
	0	0	0
	1	0	1

A   B		0	1
	0	0	1
	1	1	1

~A	~	
	0	1
	1	0

A ^ B	^	0	1
	0	0	1
	1	1	0

# Exemplos

---

```
unsigned short a,b,c;
```

```
a = 0xFF00;          /* a = 1111 1111 0000 0000 */
```

```
b = 0xA5A5;          /* b = 1010 0101 1010 0101 */
```

```
c = a | b;           /* c = 0xFFA5 */
```

```
c = a & b;           /* c = 0xA500 */
```

```
c = ~a;              /* c = 0x00FF */
```

```
c = a^b;             /* c = 0x5AA5 */
```

# Deslocamento de bits

---

$x \ll n$

deslocamento (shift) para a esquerda de  $n$  bits

- ✓ bits à direita são preenchidos com 0

$x \gg n$

deslocamento (shift) para a direita de  $n$  bits

- ✓ lógico: bits à esquerda são preenchidos com 0
  - ✓ aritmético: bits à esquerda repetem o bit mais significativo
- 
- ✓ se o operando é ***unsigned***, o compilador usa o shift lógico
  - ✓ se o operando é ***signed***, é livre escolha do compilador
    - ✓ mas em geral o compilador usa o shift aritmético

# Exemplos

---

Operação	x = [01100011]	x = [10010101]
x << 4	0 0 1 1 0 0 0 0	0 1 0 1 0 0 0 0
x >> 4 (lógico)	0 0 0 0 0 1 1 0	0 0 0 0 1 0 0 1
x >> 4 (aritmético)	0 0 0 0 0 1 1 0	1 1 1 1 1 0 0 1