Módulo 9 - Operadores bit a bit

Em linguagem C, os operadores bit a bit permitem a manipulação de valores a nível de bits. Eles são usados para realizar operações bit a bit em inteiros, o que pode ser útil em várias situações, como manipulação de configurações de hardware, compactação de dados e outras operações de baixo nível. Aqui estão os principais operadores bit a bit em C:

1. `&` **(E bit a bit)**:
   - Sintaxe: `a & b`
   - Descrição: Realiza uma operação lógica E bit a bit entre os bits de `a` e `b`. O resultado é 1 se ambos os bits forem 1.
2. `|` **(OU bit a bit)**:
   - Sintaxe: `a | b`
   - Descrição: Realiza uma operação lógica OU bit a bit entre os bits de `a` e `b`. O resultado é 1 se pelo menos um dos bits for 1.
3. `^` **(OU exclusivo bit a bit)**:
   - Sintaxe: `a ^ b`
   - Descrição: Realiza uma operação lógica OU exclusivo (XOR) bit a bit entre os bits de `a` e `b`. O resultado é 1 se os bits forem diferentes.
4. `~` **(Complemento bit a bit)**:
   - Sintaxe: `~a`
   - Descrição: Inverte todos os bits de `a`, trocando 0 por 1 e vice-versa.
5. `<<` **(Deslocamento à esquerda)**:
   - Sintaxe: `a << n`
   - Descrição: Desloca os bits de `a` para a esquerda em `n` posições. Isso é equivalente a multiplicar `a` por 2 elevado a `n`.
6. `>>` **(Deslocamento à direita)**:
   - Sintaxe: `a >> n`
   - Descrição: Desloca os bits de `a` para a direita em `n` posições. Isso é equivalente a dividir `a` por 2 elevado a `n`.

Exemplo de uso:

```
#include <stdio.h>

int main() {
    unsigned int a = 12; // 1100 em binário
    unsigned int b = 25; // 11001 em binário

    unsigned int resultado;

    resultado = a & b; // Operação E bit a bit
    printf("a & b = %u\n", resultado); // Resultado: 8 (1000 em binário)

    resultado = a | b; // Operação OU bit a bit
    printf("a | b = %u\n", resultado); // Resultado: 29 (11101 em binário)

    resultado = a ^ b; // Operação OU exclusivo bit a bit
    printf("a ^ b = %u\n", resultado); // Resultado: 21 (10101 em binário)

    resultado = ~a; // Complemento bit a bit
    printf("~a = %u\n", resultado); // Resultado: 4294967283

    resultado = a << 2; // Deslocamento à esquerda
    printf("a << 2 = %u\n", resultado); // Resultado: 48 (11000 em binário)

    resultado = b >> 1; // Deslocamento à direita
    printf("b >> 1 = %u\n", resultado); // Resultado: 12 (1100 em binário)

    return 0;
}
```

Lembre-se de que os operadores bit a bit geralmente são usados em contextos onde a manipulação de bits é necessária, como programação de sistemas embarcados, manipulação de protocolos de comunicação, etc.