Operadores bit a bit (bitwise)

Em programação de microcontroladores é muito comum se deparar com situações em que se deseja manipular um ou mais determinados bits em um byte. Exemplo disso são os registradores dos microcontroladores, em que o programador pode ter o acesso a um bit ou a todos de uma só vez. A linguagem C permite estas operações. Não se deve confundir com as operações lógicas em que uma expressão deve ser avaliada como verdadeiro ou falso (como exemplo, as condições dos *if*, *while*, etc).

Vale ressaltar que é importante o estudante fazer uma revisão sobre variáveis inteiras (*char*, *int*, *long int*, e os modificadores *unsigned* e *signed*), seus tamanhos em byte no *standard* C, e as faixas de representação (uma variável *char*, por exemplo pode assumir valores dentro de uma faixa de -128 a + 127).

Operações lógicas para usar bits em Linguagem C

Antes de conhecer os operadores, vamos relembrar as operações lógicas em sistemas digitais.

Lógica NÃO (NOT)

Inverte o estado do bit.

Tabela verdade:

A	S
0	1
1	0

Lógica E (AND)

A saída será igual a 1 somente se todos os bits de entrada forem iguais a 1. Se pelo menos um dos bits for 0, a saída será 0.

Tabela verdade:

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Lógica OU (OR)

A saída assumirá nível lógico 1 se pelo menos 1 bit de entrada for igual a 1. Somente assumirá nível lógico 0 se todas as entradas apresentarem nível lógico 0.

Tabela verdade:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Lógica Ou-Exclusivo (XOR)

A saída será igual a 1 somente se um dos valores for 1 (o outro deve ser zero). Falando de outra forma, a saída será igual a 0 se as entradas forem iguais. Caso as entradas forem diferentes a saída assumirá o valor 1.

Tabela verdade:

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Operadores bit a bit em linguagem C

Operador	*Função*
&	**E (AND)**
|	**OU (OR)**
^	**OU-EXCLUSIVO (XOR)**
~	**NOT ou Complemento de 1**
<<	**Deslocamento à esquerda (shift left)**
>>	**Deslocamento à direita (shift right)**

Os operadores de deslocamento fazem o *shift* de bits para esquerda ou direita.

Exemplo:

Vamos assumir que **x** incialmente tenha o valor 0b00000001, valor 1 em decimal. Note que os compiladores C *standard* não têm o prefixo para indicar binário nos números, mas o software para o arduino sim, e este prefixo é 0b (zero b minúsculo)

```
1 x = (x<<2);
```

Após a operação acima, será atribuido a *x* o valor 0b000000100, valor 4 em decimal.

Se em seguida realizarmos uma operação de deslocamento à direita:

```
2 x = (x>>1);
```

Após essa operação será atribuído à variável *x* o valor 0b00000010, valor 2 em decimal.

Note que a operação de deslocamento à esquerda multiplica por dois a cada *shift*. Já o deslocamento à direita, divide por dois a cada *shift*. Dessa forma pode-se também utilizar esses operadores para multiplicação ou divisão por 2, sendo mais eficiente que utilizando o operador de multiplicação. É bom também ressaltar que o deslocamento não é uma rotação e entram zeros na extremidade oposta ao deslocamento. Percebam que pode não ser possível reconstituir os números após dois deslocamentos opostos consecutivos de mesma magnitude.

Usando máscara de bits

Máscaras são sequências de bits que podem ser empregadas para a alteração de bits específicos em um byte (ou conjunto qualquer de bits) com as operações apresentadas. A seguir são exibidas algumas operações com máscaras.

Ativando um bit (colocar 1, setar)

Para ativar um bit especifico de uma variável ou registrador, ou seja, colocar nível lógico 1, preservando os outros sem mudança, utiliza-se o operador | (OU). Por exemplo, desejamos colocar em nível 1 o bit 5 (lembre que a numeração parte da direita para esquerda começando com zero) do PORTB (registrador de 8 bits que espelha terminais específicos), no ATmega328 (microcontrolador usado na construção das placas arduino, se isso ainda não é familiar, pense como uma variável de 8 bits). Para isso fazemos a seguinte operação:

```
1 PORTB = PORTB|0b00100000;
```

```
  X X X X X X X X     -> valor presente no registrador PORTB
| 0 0 1 0 0 0 0 0     -> máscara para alteração do bit 5
  X X 1 X X X X X     -> resultado, apenas o bit 5 é alterado
```

Note que isto funciona porque **1 OU 0 = 1** e **1 OU 1 = 1**, ou seja, **1 OU X = 1**. Já se fizermos **0 OU 0 = 0** e **0 ou 1 = 1**, ou seja, **0 OU X = X**.

A criação da máscara pode ser feita com o operador de deslocamento à esquerda, ficando da seguinte forma:

```
1 PORTB = PORTB | (1<<5);
```

Que é a mesma coisa que fazer **0b00000001 << 5** para a máscara.

Será feita a lógica *OU* com o byte criado na operação (1<<5), que deslocou o valor '1' 5 vezes para a esquerda.

A operação pode ser simplificada, ficando da seguinte forma:

```
1 PORTB |= (1<<5);
```

Limpando um bit (zerando, resetando)

Para limpar um bit específico de uma variável ou registrador, ou seja, colocar nível lógico 0, utiliza-se o operador & (AND). Vamos utilizar o mesmo registrador e bit utilizado no exemplo anterior como exemplo:

```
1 PORTB = PORTB & 0b11011111;
```

```
  X X X X X X X X     -> valor presente no registrador PORTB
& 1 1 0 1 1 1 1 1     -> máscara para alteração do bit 5
  X X 0 X X X X X     -> resultado, apenas o bit 5 é alterado
```

Usando o operador de deslocamento ficará da seguinte forma:

```
1 PORTB = PORTB & (~(1<<5);
```

Note que é necessário inverter os valores para a máscara ficar da forma desejada.

A operação pode ser simplificada, ficando da seguinte forma:

```
1 PORTB &= ~(1<<5);
```

Invertendo o estado de um bit

Para inverter o estado lógico de um bit utiliza-se o operador ^ (*OU-Exclusivo*). Continuamos com o mesmo bit e registrador dos exemplos anteriores:

```
1 PORTB = PORTB^0b00100000;
```

```
      1
  X X / X X X X X     -> valor presente no registrador PORTB
      0
^ 0 0 1 0 0 0 0 0     -> máscara para alteração do bit 5
      0
  X X / X X X X X     -> resultado, o bit 5 é invertido
      1
```

Usando o operador de deslocamento ficará da seguinte forma:

```
1 PORTB = PORTB ^ (1<<5);
```

A operação pode ser simplificada, ficando da seguinte forma:

```
1 PORTB ^= (1<<5);
```

Testando valor de um bit

Para teste de um bit também utilizaremos a lógica **&** (*AND*) para operação com a máscara de bits. Por exemplo, temos uma tecla ligada ao pino 2 do PORTD do Atmega328. Para leitura desse pino utilizamos o registrador PIND. A operação ficará da seguinte forma:

```
1 PIND & (1<<2)
```

```
  X X X X X T X X   -> valor presente no registrador PIND
& 0 0 0 0 0 1 0 0   -> máscara para alteração do bit 2
  0 0 0 0 0 T 0 0   -> resultado, o valor do bit estará presente no resultado
```

Caso o bit de teste tenha o valor 1, o resultado conterá o valor 1 somente na posição do bit, caso contrário o resultado será 0.

Exercícios:

Para os exercícios, considere os registradores PORTB e PORTD tendo 8 bits, e sendo possível a notação com prefixo binário (0b).

1 – Para a sequência em C para arduino abaixo, mostre os valores finais em bits das operações:
PORTB= 0b10011100;
PORTD= 0b00111101;

a) ~PORTB

b) PORTB | PORTD

c) PORTB & PORTD

d) PORTB ^ PORTD

e) (PORTB << 4) & (~(PORTD >>2)

2 – Atribua um valor a PORTB de tal forma que após as operações consecutivas abaixo, PORTB não seja igual ao valor inicial.

PORTB = PORTB >> 2;
PORTB = PORTB << 2;

3 – Mostre as operações com máscaras para que o bit 7 de PORTD seja invertido

4 – Mostre as operações com máscaras e deslocamentos para que os bits 3 e 6 de PORTB sejam trocados.

5 – Mostre as operações com máscaras e deslocamentos para que PORTB tenha os nibbles (conjunto de 4 bits consecutivos) trocados .

6 – Mostre as operações com máscaras e deslocamentos para que PORTB tenha os bits espelhados, ou seja:

$B_7B_6B_5B_4B_3B_2B_1B_0$ para $B_0B_1B_2B_3B_4B_5B_6B_7$