UNIVERSIDADE DO ESTADO DE MINAS GERAIS

Docente: Rildo Afonso de Almeida

Microprocessadores & Microcontroladores

5 - Programação C

5.3 - Operadores

A linguagem C possui uma gama de operadores, sendo possivelmente uma das linguagens com maior número de operadores disponível atualmente.

Podemos classificar os operadores da linguagem C em sete categorias principais: ***atribuição, aritméticos, relacionais, lógicos bit a bit, de memória e outros.***

5 - Programação C

5.3.1 - Atribuição

Em C, o operador de atribuição “=“ é utilizado para atribuir um determinado valor a uma variável. Um exemplo de atribuição:

```
x = 10;
y = x;
```

Podemos verificar no programa anterior duas operações de atribuição: na primeira foi atribuído o valor 10 à variável “x”, na segunda, foi atribuído o valor de “x” (que é 10 à variável “y”.

5 - Programação C

5.3.2 - Aritméticos

São utilizados para determinar ao compilador que efetue determinadas operação matemática em relação a um ou mais dados:

Operador	Ação
+	Adição
-	Subtração ou menos unário
*	Multiplicação
/	Divisão
%	Resto de Divisão Inteira
++	Incremento
- -	Decremento

5 - Programação C

5.3.2 - Aritméticos

Os operadores de adição, subtração, multiplicação e divisão dispensam comentários.

O operador % é utilizado para retornar o resto de um operação de divisão inteira. Vejamos um exemplo:

5 / 2 = 2,5 em uma divisão real , ou 5 / 2 = 2, em uma divisão inteira , sendo o resto igual a 1.

Assim, o resultado de 5 / 2 é 2 e o resultado de 5 % 2 é igual a 1.

5 - Programação C

5.3.2 - Aritméticos

Os operadores de incremento e decremento são utilizados para somar 1 (incremento) ou subtrair 1 (decremento) de uma variável.

A forma geral para utilização destes dois últimos operadores é:

variável ++; ou variável - -;

Ou ainda por meio de uma atribuição:

variável_1 = variavel_2 ++; ou variável_1 = variavel_2 - -;

5 - Programação C

5.3.2 - Aritméticos

Em C é também possível escrever:

variável_1 = ++ variavel_2; ou variável_1 = - - variavel_2;

Neste casos, a operação de incremento/ decremento é realizada antes da atribuição propriamente dita.

Vejamos um exemplo:

```
int x, y, z;
x= 0;
y= x ++;
z = ++ x;
```

5 - Programação C

5.3.2 - Aritméticos

Vejamos um exemplo:

```
int x, y, z;
x= 0;
y= x ++;
z = ++ x;
```

Neste casos, após a execução dos três comandos, o valor da variável “x” será igual a 2, o valor da variável “y” será igual a 1 e o valor da variável “z” será igual a 2.

Obs.: Não é possível utilizar os operadores de incremento ou decremento com variáveis ou tipos de dados complexos, tais como os tipos ponto flutuante.

5 - Programação C

5.3.3 - Relacionais

São utilizados em testes condicionais para determinar a relação existentes entre os dados:

Operador	Ação
>	Maior que
>=	Maior ou Igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
!=	Diferente

5 - Programação C

5.3.4 - Booleanos

Os operadores lógicos ou booleanos são utilizados para realizar conjunções, disjunções ou negações entre elementos em um teste condicional. Os operadores lógicos somente podem resultar em um dos valores: verdadeiro ou falso.

Operador	Ação
&&	AND (E)
||	OR (OU)
!	NOT (NÃO)

UEMG

5 - Programação C

5.3.4 - Booleanos

Os operadores relacionais são elementos de suma importância na construção de testes condicionais. Com esses operadores podemos relacionar diversas condições diferentes em um mesmo teste lógico.

Vejamos um exemplo:

```
int x, y;
x = 10;
if (x>5  &&  x<20)  y = x;
```

5 - Programação C

5.3.4 - Booleanos

Vejamos um exemplo:

```
int x, y;
x = 10;
if (x>5  &&  x<20)  y = x;
```

Como podemos verificar , a variável “y” somente será igual ao valor da variável “x” se o valor de “x” for maior que 5 e “x” for menor que 20. O que nos leva a concluir que ao final da execução do programa “y” será igual a 10.

5 - Programação C

5.3.4 - Booleanos

Um aspecto importante a ser observado é que em C, uma variável com valor igual a zero, será avaliada como falsa e se tiver um valor diferente de zero, será avaliada como verdadeira. Vejamos um outro exemplo:

```
int  teste, teste_2;
teste = 0;
teste_2 = 0;
if (!Teste)  teste_2 ++;
```

5 - Programação C

5.3.4 - Booleanos

Como podemos perceber, a avaliação de “!teste” será verdadeira, já que a variável possui valor zero e o teste verifica a negação da variável. Assim no presente caso, a variável “teste_2“ terminará com valor 1.

```
int  teste, teste_2;
teste = 0;
teste_2 = 0;
if (!Teste)  teste_2 ++;
```

5 - Programação C

5.4 – Lógicos Bit a bit

Os operadores lógicos bit a bit são utilizados para realizar operações lógicas entre elementos ou variáveis. No entanto, ao contrário dos operadores lógicos simples, os operadores lógicos bit a bit podem resultar em um valor da mesma magnitude dos elementos operados.

Operador	Ação
&	AND (E)
|	OR (OU)
^	XOR (OU exclusivo)
~	NOT (complemento de um)
>>	Deslocamento à direita
<<	Deslocamento à esquerda

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.1 – Operador & (And)

A operação lógica AND funciona da mesma forma que o operador booleano AND, mas com a diferença de que a operação é realizada separadamente para cada bit dos operandos. Vejamos um exemplo:

```
int v1, v2;
v1 = 100;
v2 = v1 & 15;
```

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.1 – Operador & (And)

A operação AND representada ocorrerá da seguinte forma:

100 decimal = 01100100
AND (&)
015 decimal = 00001111

Resultado = 00000100

Isto significa que o valor armazenado em “v2” será igual a 4 decimal.

UEMG

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.2 – Operador | (OR)

A operação OR, tal qual o operador AND, também trabalha de maneira similar ao seu equivalente booleano, com a diferença de que também aqui a operação é realizada para cada bit dos operadores. Vejamos um exemplo:

```
int v1, v2;
v1 = 0x20;
v2 = v1 | 0x04;
```

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.2 – Operador | (OR)

A operação OR será realizada da seguinte forma:

20 hexadecimal = 00100000
OR (|)
04 hexadecimal = 00000100

Resultado = 00100100

Isto significa que o valor armazenado em “v2” será igual a 24 hexadecimal.

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.3 – Operador ^ (XOR)

A operação XOR (exclusivamente OU) consiste em uma operação lógica entre dois, na qual o resultado somente será verdadeiro (nível lógico 1) se um e somente um deles for verdadeiro (nível 1). Ou seja, caso os operandos sejam iguais (0, 0 ou 1, 1), o resultado será falso (nível 0).

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.3 – Operador ^ (XOR)

Operadores XOR são muito utilizados em função de comparações de valores: se os bits dos operandos são iguais, o resultado é 0; se forem diferentes, o resultado é 1. Vejamos um exemplo do funcionamento do operador XOR em C:

```
int x , y;
x = 100;
y = x ^ 99;
```

UEMG

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.3 – Operador ^ (XOR)

A operação XOR será realizada da seguinte forma:

100 decimal = 01100100
XOR (^)
99 decimal = 01100011

Resultado = 00000111

Podemos observar que apenas os bits diferentes entre os dois operandos resultaram em um valor 1. Os bits iguais resultaram em um valor 0.

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.4 – Operador ~ (NOT)

O NOT atua como operador de negação, ou em aritmética binária o complemento de um. Isto significa que o operador NOT inverte o estado de cada bit do operando especificado. Vejamos um exemplo:

```
int x , y;
long a, b;
x = 1;
a = 1;
y = ~x;
b = ~ a;
```

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.4 – Operador ~ (NOT)

Vejamos a operação do operador de complemento:

1 decimal = 00000001 (8 bits, variável x)

Resultado = 11111110 (8 bits, variável y)

1 decimal = 0000000000000001 (16 bits, variável a)

Resultado = 1111111111111110 (16 bits, variável a)

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.4 – Operador ~ (NOT)

Concluímos então que a variável “x”, de 8 bits, terminará o programa com o complemento do valor 1, ou seja 254.

Já a variável “b” será atribuído o valor de negação de “a”, de 16 bits, o que resulta no valor 65534.

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.4 – Operador ~ (NOT)

O NOT atua como operador de negação, ou em aritmética binária o complemento de um. Isto significa que o operador NOT inverte o estado de cada bit do operando especificado. Vejamos um exemplo:

```
int x , y;
long a, b;
x = 1;
a = 1;
y = ~x;
b = ~ a;
```

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

O formato geral de uso dos operadores de deslocamento de bits à esquerda e à direita é o seguinte:

- Valor >> número de bits a deslocar à direita ou
- Valor << número de bits a deslocar à esquerda

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

Vejamos um pequeno programa para demonstrar o funcionamento dos operadores de deslocamento:

```
int x, y, z;
x = 10;
y = x << 2;
z = x >> 1;
```

O funcionamento dos operadores de deslocamento é o seguinte:

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

Primeiramente é atribuído à variável “y” o deslocamento de dois bits à esquerda da variável “x”. Vejamos o funcionamento desta operação:

10 decimal = 00001010

<<

00010100

<<

Resultado = 00101000

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

Observe que foram realizadas duas operações de deslocamento, sendo que o primeiro deslocamento resultou em 00010100 binário (20 decimal). Em seguida é realizado outro deslocamento, que resulta em 00101000 binário (40 decimal), sendo este valor atribuído à variável “y”.

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

A próxima linha atribui à variável “z” o valor de “x” deslocando um bit à direita. Vejamos o funcionamento desta operação:

10 decimal = 00001010

>>

z = 00000101

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

Percebemos que o conteúdo da variável “x” (00001010 binário ou 10 decimal) é deslocado um bit à direita, resultado em 00000101 binário ou 5 decimal, sendo este valor atribuído à variável “z”.

5 - Programação C

5.4 – Lógicos Bit a bit

5.4.5 – Operadores de Deslocamento << e >>

Com base nestas operações podemos verificar que no operador de deslocamento à esquerda, cada bit deslocado equivale a multiplicar o primeiro operando por 2. Já no operador de deslocamento à direita, equivale a dividir o operando por 2.

Obs.: Não é possível utilizar os operadores lógicos bit a bit com variáveis ou tipos de dados complexos, tais como os tipos ponto flutuante.

5 - Programação C

5.4.6 – Operador de Memória

Os operadores de memória, também chamados de operadores de ponteiros, são elementos de grande importância na linguagem C. De fato, os ponteiros são considerados um dos pilares da linguagem C, pois permitem o acesso direto a qualquer endereço de memória do sistema. Existem dois operadores complementares para o acesso à memória:

Operador	Ação
&	Endereço do operando
*	Conteúdo do endereço apontado pelo operando

5 - Programação C

5.4.6 – Operador de Memória

5.4.6.1 – Operador &

O & é um operador unário(único) utilizado para retornar o endereço de memória do seu operando. Isto significa que se escrevermos:

endereco_a = &a;

Teremos que a variável “endereco_a” conterá o endereço em que está armazenada a variável “a”.

5 - Programação C

5.4.6 – Operador de Memória

5.4.6.1 – Operador *

Já o * é um operador unário(único) utilizado para retornar o conteúdo da posição de memória endereçada pelo operando que o segue. Vejamos outro exemplo:

a = *endereco_a;

O que fará com que o valor armazenado no local apontado pela variável “endereco_a” seja atribuído à variável “a”.

5 - Programação C

5.4.6 – Operador de Memória

5.4.6.1 – Operador *

Vejamos um pequeno exemplo da utilização de ponteiros:

```
#include <16f628.h>
#use delay (clock=4000000)
#fuses NOWDT, INTRC_IO, PUT, NOPROTECT, BROWNOUT, NOMCLR, NOCPD
#use rs232(baud=2400, xmit=PIN_A5, INVERT )
int  endereco, x, y;
main () {
x = 5;
endereco = &x;
y = *endereco;
Printf("valor x= %d – endereco de x = %1x – valor y = %d", x, endereco, y);
}
```

5 - Programação C

5.4.6 – Operador de Memória

5.4.6.1 – Operador *

Provavelmente a saída impressa deste programa será:

Valor x = 5 – endereco de x = 26 – valor y = 5

O que significa que o valor de “&x” (endereço de “x”) é igual a 0x26, ou seja, a variável “x” está localizada no endereço 0x26 da memória RAM do PIC.

5 - Programação C

5.4.7 – Outros Operadores

Além dos operadores citados anteriormente, podemos encontrar ainda outros operadores não tão conhecidos em C:

Operador	Ação
?	Operador ternário condicional
,	Separador de expressões
.	Separador de estruturas
- >	Ponteiro de elemento de estruturas
(tipo)	Operador de Modelagem de dado
sizeof	Retorna o tamanho da variável

5 - Programação C

5.4.7.1 – Operador ?

O operador ternário “?” é utilizado para substituir uma expressão condicional baseada no comando IF e tem esse nome devido ao fato de ser composto por três expressões. Sua forma geral é:

Variável = Expressão1? Expressão2 : Expressão3

O que significa: avalie a expressão 1 e se for verdadeira, atribui à variável a expressão 2; caso contrário, atribua a expressão 3.

5 - Programação C

5.4.7.1 – Operador ?

Vejamos um exemplo:

```
int x, y;
x = 5 ;
y = x==7 ? 10 : x + 3;
```

5 - Programação C

5.4.7.1 – Operador ?

O funcionamento desse programa é o seguinte: primeiramente é atribuído o valor 5 à variável "x". Em seguida, a expressão condicional é avaliada: se "x" for igual a 7, então "y" será igual a 10; caso contrário, "y" será igual a "x" mais 3. Como sabemos que "x" é igual a 5, teremos que "y" será igual a 8 (5+3)

```
int x, y;
x = 5 ;
y = x==7 ? 10 : x + 3;
```

5 - Programação C

5.4.7.2 – Operador ,

Outro operador pouco conhecido da linguagem C é o vírgula, utilizado para enfileirar duas ou mais expressões. A forma geral de utilização deste operador é:

variável = (expressão 1 , expressão 2 [, expressão x])

5 - Programação C

5.4.7.2 – Operador ,

variável = (expressão 1 , expressão 2 [, expressão x])

Note que as expressões são avaliadas da esquerda para a direita e a variável recebe o valor da última expressão avaliada. Veja o exemplo a seguir:

```
y = (x = 0, x+ 5);
```

UEMG

5 - Programação C

5.4.7.2 – Operador ,

Como podemos perceber, primeiramente a variável “x” assume o valor zero e em seguida, “y” irá assumir o valor de “x” mais 5, ou seja, ao final da avaliação da expressão, “y = 5”.

```
y = (x = 0, x+ 5);
```

5 - Programação C

5.4.7.3 – Operador .

O operador ponto é utilizado em estruturas de dados como separador dos elementos.

5 - Programação C

5.4.7.4 – Operador - >

Também chamado de operador seta, é utilizado para a função de ponteiro para uma estrutura de dados.

5 - Programação C

5.4.7.5 – Operador de Modelagem de Tipo

A linguagem C dispõe de um operador unário destinado especificamente a forçar a conversão do operando especificado em um tipo de dado determinado. A isto, dá-se o nome de modelagem ou typecasting. A forma geral do operador de modelagem é:

(tipo) variável

Onde o “tipo” especifica o novo tipo de dado para o qual o conteúdo atual da variável especificada será convertido.

5 - Programação C

5.4.7.5 – Operador de Modelagem de Tipo

Observe que o conteúdo da variável não é alterado pelo operador, ao invés disso, aloca-se uma região da memória RAM para o armazenamento temporário do valor modificado.

5 - Programação C

5.4.7.6 – Operador Sizeof

O operador sizeof é utilizado para retornar a quantidade de memória utilizada por uma determinada variável ou um determinado tipo de dado.

A principal aplicação deste tipo de operador é permitir ao programador controlar a ocupação de memória pelo programa, auxiliando assim na portabilidade do programa. A forma geral do operador é:

sizeof variável ou
sizeof (tipo de dados)

UEMG

Bibliografia Básica

MIYADAIRA, A. N. *Microcontroladores PIC18: aprenda e programe em Linguagem C* Ed. Érica, 1ª Ed., 2009, São Paulo.

LUZ, C. E. S. *Programando Microcontroladores PIC em Linguagem C com base no PIC4520.* Ed. Ensino Profissional, 2011, São Paulo.

SILVA, R. A. *Programando Microcontroladores PIC.* Ed. Ensino Profissional, 2011, São Paulo.

PEREIRA, F. *Microcontroladores PIC – Programação em C.* Érica: São Paulo, 2003.

UEMG

Bibliografia Complementar

TAUB, H.. Circuitos Digitais e Microprocessadores. McGraw Hill do Brasil, 1984.

ZILLER, Roberto M. *Microprocessadores: Conceitos Importantes*. Edição do Autor, 2000.

DALTRINI, Beatriz M., JINO, M., MAGALHÃES, L. P.. *Introdução à Computação Digital*. Makron Books, 1999.