Arquitectura de Computadores 2008/09

Aula prática 4 – Expressões e Operadores em C

1. Considere:

```
int a[2];

a[0] = 2;
a[1] = 0;
```

Indique o resultado das seguintes expressões em C:

a. `a[0] == a[1]`

b. `!a[0] || a[1]`

c. `*a`

d. `*a+1`

e. `*( a+1 )`

f. `(*a) * (*(a+1))`

g. `!!*a`

h. `a[0] = a[1]`

Valide as suas respostas experimentalmente com um pequeno programa em C, que apresente no ecrã o resultado das diversas expressões.

2. Considere a seguinte declaração:

```
unsigned char b;
```

Escreva expressões, válidas em C, para:

a. Colocar o bit 1 de b a **1**, mantendo os outros inalterados;

b. Colocar os 4 bits mais significativos b a **0**, mantendo os outros inalterados;

c. Colocar o bit 2 de b a **0**, mantendo os outros inalterados;

d. Determinar se o bit 0 de b é **0** ou **1**;

Valide as expressões com um pequeno programa em C, que apresente no ecrã os diversos resultados (**considere `b=0x85`**).

Explique (no papel) os resultados obtidos, convertendo os números para binário.

3. Considere:

```
int n;
```

Baseando-se nos operadores de deslocamento de bits, da linguagem C, e sem utilizar multiplicações (*) nem divisões (/), escreva expressões para:

a. Multiplicar n por 16;

b. Dividir n por 4;

c. Multiplicar n por 12;

Teste as suas expressões através de um programa em C, que pede ao utilizador números (positivos ou negativos) e apresenta no ecrã os resultados.

4. Reescreva a função **`mystrlen`** da aula anterior, de forma a esta ficar o mais compacta possível, utilize para tal a desreferenciação e o incremento de apontadores.

5. Considere as seguintes declarações:

```
int i = 0x11223344;

int *pi = &i;
char *pb = (char *)pi;
```

Utilize-as como base para um programa que determina, experimentalmente, se o seu computador é *big endian* ou *little endian*.

Nota:

- *big endian*, byte mais significativo de um inteiro é armazenado primeiro (no endereço mais baixo);
- *little endian*, byte menos significativo de um inteiro é armazenado primeiro (no endereço mais baixo).

6. Programe a função:

```
void converte_binario( unsigned char b, char s[] )
```

Que dado um byte **b** preenche a *string* **s** com a sua representação em binário.

Utilize os operadores de manipulação de bits do C e assuma que **s**, tem pelo menos 9 posições.

Teste a função através de um programa que pede ao utilizador um número e o converte para binário.

Para ler o byte pode utilizar o seguinte código:

```
unsigned char b;
unsigned int u;

scanf("%x",&u);     // ler em hexadecimal
b = u;
```

7. Programe uma função, que inverte o bit 5 de um byte, mantendo os restantes bits inalterados:

```
void inverte5( unsigned char *b )
```

Sugestão: utilize o **ou exclusivo**.