Arquitetura de Computadores 2017/18

TPC 2

Prazo de entrega: 23:59 de 14 de abril de 2018

Tópicos: *Programação em C. Representação de reais.*

Este trabalho de casa consiste em dois exercícios de programação **individual**. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código deve ser estritamente individual. Todas as resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor.

A entrega será via Mooshak do DI (`http://mooshak.di.fct.unl.pt/~mooshak/`). Os programas serão compilados com o seguinte comando:

```
cc -m32 -Wall -std=c11 -o prog  prog.c
```

Exercício 1

Complete a função abaixo que deve imprimir valores do tipo **`float`**, na forma:

mantissa **`x2^`***expoente*

O sinal e os valores em *mantissa* e *expoente* devem ser obtidos diretamente dos bits na representação **`float`**. A função tem a seguinte assinatura:

```
void printFloat( float val );
```

Exemplo de utilização da função `printFloat`:

```
printFloat( -2.5 );
```

escreve no *output* a seguinte *string* (mudando de linha):

`-1.250000 x2^1`

Note que -2.5 tem a representação: `1 10000000 01000000000000000000000` Como o bit 31 é 1 representa um valor negativo (daí '**-**'); nos bits do expoente temos 10000000_2= 128_{10} logo, depois de subtrair o deslocamento de 127, obtemos o expoente **1**; e na parte da mantissa está representado 1.01_2= 1.25_{10}.
Para obter em C os bits na representação do **`float`**, ou passar para **`float`** qualquer padrão de bits, use as funções fornecidas que permitem obter um **`unsigned int`** com os mesmos bits do **`float`** e *vice versa*. Pode assim manipular essa representação com os operadores aritméticos e de lógica binária.

```
#include <stdio.h>

unsigned F2U( float n ) {  // float to unsigned int
  return *(unsigned*)&n;
}

float U2F( unsigned n ) {  // unsigned int to float
  return *(float*)&n;
}

void printFloat( float val ) {
  unsigned int valbits = F2U( val );
  char sign;   // should be ' ' or '-'
  int exponent;
  float mantissa;
// TODO
```

```
  printf("%c%f x2^%d\n", sign, mantissa, exponent);
}

int main( int argc, char *argv[] ) {
  float x;

  scanf( "%f", &x );
  printFloat(x);
  return 0;
}
```

Exercício 2

Implemente uma função para copiar quaisquer elementos de um vetor `src` para outro `dst`, dada a dimensão de cada elemento do vetor e o número de elementos a copiar (não pode usar qualquer outra função para auxiliar a sua implementação). A assinatura da função será a seguinte:

```
void arrayCopy( void *dst, void *src, int elemSize, int len );
```

Esta deve permitir, *inclusive*, a copia de subvetores dentro do mesmo vetor, **tendo em conta eventuais sobreposições**†. Assuma que nunca se copia para além dos limites dos vetores. Para testar a sua implementação e como exemplo de utilização, pode usar o programa fornecido que deve afixar como output:

```
3 4 5 6 0
1 2 1 2 3 6
uma mensagem!
mensagem!
```

```
#include <stdio.h>
#include <string.h>

void arrayCopy( void *src, void *dst, int elemSize, int num ) {
    // TODO

}

int main() {    // programa de teste
    int v1[6] = { 1, 2, 3, 4, 5, 6 };
    int v2[5] = { 0, 0, 0, 0, 0 };
    char s1[] = "uma mensagem!";
    char s2[100] = "";

    arrayCopy( v1+2, v2, sizeof(int), 4 );
    for ( int i=0; i<5; i++ )
       printf("%d ", v2[i] );
    putchar('\n');

    arrayCopy( v1, v1+2, sizeof(int), 3 );
    for ( int i=0; i<6; i++ )
       printf("%d ", v1[i] );
    putchar('\n');

    arrayCopy( s1, s2, sizeof(char), strlen(s1)+1 );
    puts(s2);

    arrayCopy( s1+4, s1, sizeof(char), strlen(s1)-3 );
    puts(s1);

    return 0;
}
```

† À semelhança de `memmove` da biblioteca do C.