AULA PRÁTICA N.º 9

Objectivos:

- A norma IEEE 754. Representação de números reais (tipos *float* e *double*).
- Programação com a unidade de vírgula flutuante do MIPS. Parte 1.

Guião:

1. Considere o seguinte programa que lê um valor inteiro, multiplica-o por uma constante real e apresenta o resultado.

```
int main(void)
{
    float res;
    int val;

    do
    {
        val = read_int();
        res = (float)val * 2.59375;
        print_float( res );
    } while(res != 0.0);
    return 0;
}
```

a) Traduza o programa para *assembly* do MIPS[1] e teste o seu funcionamento no MARS com diferentes valores de entrada.

b) Determine, manualmente, a representação binária em vírgula flutuante com precisão simples, segundo a norma IEEE 754, do valor 7.78125 (3 * 2.59375). Compare o valor obtido com o calculado pela unidade de vírgula flutuante do MIPS quando o valor de entrada do programa é 3 (certifique-se que a opção "*values displayed in hexadecimal*" do menu "*settings*" do MARS está activa).

2. A função seguinte converte um valor de temperatura em graus Fahrenheit para graus Celsius.

```
double f2c(double ft)
{
    return (5.0 / 9.0 * (ft - 32.0));
}
```

a) Escreva, em linguagem C, a função **`main()`** para teste da função **`f2c()`**. Traduza as duas funções para *assembly* do MIPS e teste o conjunto com diferentes valores de entrada (note que para a impressão do valor no ecrã tem que usar a *system call* **`print_double()`**). Recorde a convenção de utilização dos registos do MIPS no que concerne à passagem de parâmetros para funções e devolução de resultados de tipo float/double.

[1] Tenha em atenção que apenas os registos de índice par da FPU podem ser usados no contexto das instruções.

3. A função **`average()`** calcula o valor médio de um *array* de reais codificados em formato vírgula flutuante, precisão dupla.

```
double average(double *array, int n)
{
   int i = n;
   double sum = 0.0;
   for(; i > 0; i--)
   {
      sum += array[i-1];
   }
   return sum / (double)n;
}
```

A seguinte função **`main()`** serve para teste da função **`average()`**.

```
#define SIZE 10
int main(void)
{
   static double a[SIZE];
   int i;
   for(i = 0; i < SIZE; i++)
   {
      a[i] = (double)read_int();
   }
   print_double( average(a, SIZE) );
   return 0;
}
```

a) Traduza as duas funções para *assembly* do MIPS e teste o conjunto.

4. A função **`max()`** calcula o valor máximo de um *array* de "n" elementos em formato vírgula flutuante, precisão dupla.

```
double max(double *p, unsigned int n)
{
   double max;
   double *u = p+n-1;

   max = *p++;
   for(; p <= u; p++)
   {
      if(*p > max)
         max = *p;
   }
   return max;
}
```

a) Traduza a função **`max()`** para *assembly* do MIPS.

b) Acrescente à função **`main()`** que escreveu no exercício anterior a chamada à função **`max()`** e a impressão no ecrã do valor máximo do array.

Exercícios adicionais

1. A função seguinte calcula a mediana dos valores de um *array* de quantidades reais, codificadas em precisão dupla.

```
#define TRUE 1
#define FALSE 0

double median(double *array, int nval)
{
     int houveTroca, i;
     double aux;

     do
     {
        houveTroca = FALSE;
        for( i = 0; i < nval-1; i++ )
        {
           if( array[i] > array[i+1] )
           {
              aux = array[i];
              array[i] = array[i+1];
              array[i+1] = aux;
              houveTroca = TRUE;
           }
        }
     } while( houveTroca == TRUE );
     return array[nval / 2];
}
```

a) Traduza a função para *assembly* do MIPS. Inclua a sua chamada na função **`main()`** que escreveu anteriormente e acrescente código para visualizar os resultados (*array* ordenado e mediana).