Trabalho prático N.º 2

Objetivos

- Consolidar os conceitos associados à implementação em *assembly* de estruturas de controlo de fluxo de execução.
- Rever os conceitos sobre a manipulação de *arrays* de inteiros com e sem sinal.

Trabalho a realizar

Parte I

1. Traduza para *assembly* do MIPS o programa em linguagem C que se apresenta na página seguinte. Esse programa preenche um *array* de inteiros, efetua a sua ordenação por ordem crescente e apresenta o resultado.
2. Verifique o correto funcionamento do programa para vários conjuntos de valores de entrada.
3. Altere a codificação do programa de ordenação de modo a fazer ordenação por ordem decrescente.
4. Reescreva o programa dado em C de modo a permitir a ordenação de inteiros com sinal e reflita essas alterações na codificação *assembly* (incluíndo as necessárias alterações nos *system calls*). Verifique o correto funcionamento do programa introduzindo, como dados de entrada, inteiros positivos e negativos.

Parte II

O *core* MIPS disponível no microcontrolador PIC32 implementa, no coprocessador 0, um contador crescente de 32 bits (designado por *core timer*) atualizado a cada dois ciclos de relógio do CPU. Na placa DETPIC32 o relógio do CPU está configurado a 40 MHz, pelo que o contador é incrementado a uma frequência de relógio de 20 MHz. Isto significa que o tempo necessário para incrementar o contador desde o valor 0 até 20.000.000 (0x01312D00) é 1 segundo.

A placa DETPIC32 disponibiliza dois *system calls* para interagir com esse contador: ler o valor atual do contador (**`readCoreTimer()`**) e reiniciar a zero o seu valor (**`resetCoreTimer()`**).

1. Usando os *system calls* adequados, escreva em C e traduza para *assembly* do MIPS um programa que permita visualizar, em ciclo infinito, o valor atual do contador.
2. Usando os *system calls* adequados, escreva em C e traduza para *assembly* do MIPS um programa que incremente uma variável (começando em 0) à frequência de 1 segundo. De cada vez que a variável for atualizada, o seu valor deve ser apresentado no ecrã. Verifique o correto funcionamento do programa.
3. Com base no ponto anterior, escreva em C e traduza para *assembly* um programa que implemente um relógio digital no formato HH:MM:SS (começando em valores pedidos ao utilizador no início do programa). Verifique o correto funcionamento do programa.

Elementos de apoio

- Tabela com resumo do conjunto de instruções da arquitectura MIPS, na versão adaptada a Arquitectura de Computadores II (disponível no site da disciplina).
- Slides das aulas teóricas de Arquitectura de Computadores I.
- David A. Patterson, John L. Hennessy, Computer Organization & Design – The Hardware/Software Interface, Morgan Kaufmann Publishers.

```
// ************************************************************************
// AC2 – Programa 1
// ************************************************************************

#define     N_INT 5
#define     TRUE  1
#define     FALSE 0

unsigned int sequentialSort(unsigned int, unsigned int *);
void troca(unsigned int *v1, unsigned int *v2);

void main(void)
{
   static unsigned int lista[N_INT];     // reservado no segmento de dados
   unsigned int i, n_trocas;
   unsigned int *ptr;

   printStr("\nLeitura e ordenacao de inteiros em base 10\n");
   printStr("Introduza 5 Inteiros: ");   // system call

   for( i = 0; i < N_INT; i++ )
   {
      lista[i] = readInt(10);            // system call
      putChar(' ');          // system call
   }

   n_trocas = sequentialSort( N_INT, lista );

   printStr("\nNumero de trocas realizado: ");
   printInt(n_trocas, 10);         // system call

   printStr("\nResultado da ordenacao: ");

   for( ptr = lista; ptr < lista + N_INT; ptr++ )
   {
      printInt(*ptr, 10);          // system call
      putChar(' ');                // system call
   }
}

unsigned int sequentialSort(unsigned int n_val, unsigned int *array)
{
   unsigned int i, j, aux, n_trocas=0;

   for( i = 0; i < n_val - 1; i++ )
   {
      for( j = i + 1; j < n_val; j++ )
      {
         if( array[i] > array[j] )
         {
            troca(&array[i], &array[j]);
            n_trocas++;
         }
      }
   }
   return n_trocas;
}

void troca(unsigned int *v1, unsigned int *v2)
{
   unsigend int aux;
   aux = *v1;
   *v1 = *v2;
   *v2 = aux;
}
```