LSDI 2013/14 – Laboratório 7

Programação em *assembly* MIPS

1. Introdução

Este trabalho laboratorial decorrerá em duas aulas, pretendendo-se completar e validar um programa em *assembly* para o processador MIPS. O programa é constituído por várias rotinas, cada uma realizando uma tarefa específica sobre sequências de números inteiros (vetores) armazenadas em memória. O processo de validação de cada rotina basear-se-à na simulação e análise da execução com a ferramenta MARS.
Como ponto de partida, é fornecido o ficheiro `progLab7.asm` com o código estruturado e comentado do programa incompleto, onde deverão ser acrescentadas as várias rotinas. Uma das facilidades incluídas no programa é uma interface que permite, através de um menu, escolher as várias operações a realizar (i.e., rotinas a executar).

2. Desenvolvimento

A construção e validação das rotinas devem ser feitas como trabalho de casa, antes das aulas laboratoriais, de modo a usar a aula para resolver possíveis problemas de implementação e para apresentar aos docentes a solução encontrada. Para cada uma das rotinas a desenvolver na primeira aula (secção 2.1) utilize os registos que são indicados em `progLab7.asm` no local destinado à inserção do respetivo código fonte, com o significado que lhes é atribuído. Como exemplo, o fragmento seguinte refere-se à rotina `sum` (código a partir da linha 262 do programa), a qual deve usar os registos `$a0`, `$a1` e `$v0`:

```
#-----------------------------------------------------------------------
# sum - cálculo do somatório dos elementos de um vetor
#
# Argumentos:
#   $a0 - dimensão do vetor
#   $a1 - endereço base do vetor
# Valor retornado:
#   $v0 - somatório
#-----------------------------------------------------------------------

sum:    # coloque o seu código a partir daqui...

        jr  $ra    # para retornar ao programa que chamou esta rotina
```

Descrevem-se de seguida as rotinas a completar nas duas aulas laboratoriais.

2.1 Operações sobre um vetor

- Soma: calcula o somatório dos elementos do vetor.
- Média: calcula o valor médio dos elementos do vetor. O resultado a apresentar deve ser arredondado (ex: 7 ÷ 3 ≈ 2, 8 ÷ 3 ≈ 3). Para o efeito, note que o arredondamento implica incrementar o quociente em uma unidade no caso de o dobro do resto ser maior que o divisor.
- Máximo: determina o valor máximo contido no vetor.
- Inverte: inversão da ordem dos elementos do vetor. O primeiro elemento troca de posição com o último, o segundo com o penúltimo, etc.

- Dimensão da maior subsequência crescente: determina o número de elementos da maior subsequência estritamente crescente existente no vetor.
  Por exemplo, o vetor [-2, 3, -4, 14, 14, 17, 18, 18, 0, 3] tem 4 subsequências estritamente crescentes: [-2, 3], [-4, 14], [14, 17, 18] e [0, 3]; a maior delas tem 3 elementos.

2.2 Operações envolvendo dois vetores

As rotinas a seguir descritas envolvem operações com dois vetores. A primeira tarefa a realizar é alterar o programa fornecido de modo a suportar a definição e introdução dos valores de dois vetores, com a mesma dimensão, para que depois possa simular cada uma das rotinas a construir. Sempre que possível, aproveite rotinas já existentes.

- Vetor soma: calcula o vetor resultante da adição dos dois vetores, guardando o resultado em memória num terceiro vetor (a definir).

- Produto interno: calcula o valor do produto interno de dois vetores, assumindo que cada produto é representável em 32 bits.

- Máximo relativo: determina o maior valor contido no vetor (entre dois possíveis) que tem maior valor médio.

- Número de elementos iguais: determina o número de elementos comuns dos vetores. Considere que em cada vetor não há elementos repetidos.