## PARTE I (sem consulta)

- (1,5) 1 Indique qual a convenção usada pelos compiladores do MIPS na preservação de registos quando da invocação de subrotinas. Que vantagens apresenta em relação à convenção "callee save" (preservação dos registos a cargo da subrotina invocada)? Ilustre a resposta com um exemplo.
- (2) **2** A instrução *la* (*load address*) é uma instrução nativa do reportório de instruções do MIPS? No caso de não o ser traduza-a em instruções nativas, indicando o formato do(s) código(s) de instrução. No caso de ser uma instrução nativa indique o formato do respectivo código de instrução.
  - 3 A e B são dois numeros representados no formato IEEE de vírgula flutuante, precisão simples.
- (1,5) **a** Indique qual a representação do produto A\*B (em binário) e qual o respectivo valor (em decimal, sob a forma  $X*2^Y$ ).
- (1) **b** Descreva em C (ou Pascal), ou através de um fluxograma, o algoritmo de multiplicação que utilizou.
- (1,5) c Desenhe o diagrama de um multiplicador série capaz de executar a multiplicação dos significandos (mantissas)
  - 4 A figura da página seguinte representa um "datapath" que executa um subconjunto do reportório de instruções MIPS. Cada instrução é executada em vários ciclos de relógio. Suponha que a instrução a executar é: addi \$8,\$10,-14
- (1) **a** Quantos ciclos de relógio são necessários à execução dos seguintes tipos de instrução:
  - Aritméticas e lógicas
  - Load e store
  - Saltos condicionais
- (1) **b** Indique os valores dos sinais do datapath, nomeadamente de A, B e ALUout, nas diferentes fases de execução da instrução, supondo que o conteúdo dos registos antes da execução da instrução é:
  - $(\$10) = 100000000000000000000000000111 = 0000000F_{16}$
  - $(\$8) = 10000000000000000111100000000111 = 80007807_{16}$
- c Indique os valores dos sinais de controlo (mais claros na figura) nas diferentes fases de execução da instrução
- (1,5) **d** Qual o valor armazenado no registo como resultado da execução da instrução de adição? É o resultado correcto da adição? Se não é, qual a lógica necessária à detecção do erro?

## **PARTE II**

<u>Cotações</u>: 1 - 0.5; 2 - 2; 3 - 2; 4 - 2; 5 - 1.5

**NOTE BEM:** Na resolução da PARTE II do exame pode consultar o anexo às folhas de acompanhamento da parte prática da disciplina, composto por cinco páginas e contendo o *set* de instruções do MIPS. Responda **apenas** ao que é solicitado em cada questão **respeitando estritamente** o que for aí determinado. **Não repita** código que já escreveu em alíneas anteriores. Para todos os efeitos, respeite as convenções adoptadas quanto à utilização e salvaguarda de registos. Respeite igualmente **rigorosamente** os aspectos estruturais e a sequência de instruções indicadas no programa original fornecido na página seguinte, bem como as indicações sobre quais registos usar para cada variável.

- 1 Escreva, em *Assembly* do MIPS, o trecho de código identificado na página seguinte pela letra *A*. Respeite a ordem de declaração das variáveis. Admita para o efeito que a variável **nsmp** reside no registo \$s6 e que as restantes variáveis deverão obrigatoriamente residir em memória, no segmento de dados.
- 2 Escreva, em *Assembly* do MIPS, o trecho de código identificado na página seguinte pela letra *B*. Use, onde necessário, as funções disponibilizadas pelo *Kernel* do PCSPIM. Assegure que a estrutura de controle de fluxo **do/while()** é implementada de forma adequada.
- 3 Escreva, em *Assembly* da <u>máquina real</u> do MIPS, o trecho de código correspondente à subrotina recursiva identificado na página seguinte pela letra *C*. Use, para armazenar a variável **bb**, o registo \$t0.
- **4** Escreva, em *Assembly* do MIPS, o trecho de código correspondente à subrotina **fir**(), identificada na página seguinte pela letra **D**. Use, para cada variável local, os registos que são indicados no código sob a forma de comentário. Note que o registo **\$f20** é do tipo *Calee Saved*. Note também que na evocação da subrotina **prod**(), o 2º parâmetro deve ser passado no registo **\$f12** e que esta subrotina devolve o valor em **\$f0**.
- 5 Escreva, em *Assembly* do MIPS, o trecho de código identificado na página seguinte pela letra E correspondente à subrotina **prod**().

```
char msg1[] = {" No de coeficientes: "};
     char msg2[] = {"\nProxima amostra: "};
char msg3[] = {"\nResultado: "};
     int amostras[80];
Α
     double coef[32];
     int vl, ncoefi;
     void main(void)
                                   // nsmp deve residir em $s6
            int nsmp;
                                          // As restantes variáveis residem na memória
            print_str(msg1);
            ncoefi = read_int();
            ... /* Trecho de código dedicado à leitura dos coeficientes a
В
                   partir de um ficheiro e inicialização do array amostras[] */
            do
            {
                   print_str (msg2);
                   nsmp = read_int();
                   if (nsmp >= 0)
                            shift (amostras, nsmp, ncoefi);
                            vl = fir (amostras, coef, ncoefi);
                            print_str (msg3);
                            print_int (vl);
            \} while (nsmp >= 0);
     void shift (int *ip, int lv, int na)
            int bb;
                                                 // bb Deverá usar o registo $t0
            int *kp;
            na --;
C
            if (na > 0) {
                   bb = *ip;
                   kp = ip + 1;
                   shift (kp, bb, na);
            *ip = lv;
     int fir(int *val, double *cf, int nv)
                                                 // n deverá residir no registo $s0
            int n = 0;
                                                 // tot deverá residir no registo $f20
            double tot = 0.0;
            for (n = 0; n < nv; n++)
D
                   tot = tot + prod (*val, *cf);
                   val ++;
                   cf ++;
            return ((int) tot);
     double prod (int j, double k)
                                                // k é recebido em $f12..$f13
E
            return ((double) j * k);
                                                // Valor a devolver em $f0..$f1
     }
```