Esta ficha foi elaborada para acompanhar o estudo da disciplina de Arquitetura de Computadores I (a2s1) ou para complementar a preparação para os momentos de avaliação finais da mesma. Num segundo ficheiro poderás encontrar um conjunto de propostas de solução aos exercícios que estão nesta ficha. É conveniente relembrar que algum conteúdo destes documentos pode conter erros, aos quais se pede que sejam notificados pelas vias indicadas na página web, e que serão prontamente corrigidos, com indicações de novas versões.

1. Codifica, em código máquina, as seguintes instruções, considerando que as instruções de tipo R têm código de operação 000000, as instruções jump têm código de operação 000010, e branch on not equal 000101:

```
a) add  $2, $8, $3     (considere-se o código de função 100000)
b) addi $2, $8, 100    (considere-se o código de operação 001000)
c) addu $2, $8, $3     (considere-se o código de função 100001)
d) sll  $2, $8, 4      (considere-se o código de função 000000)
e) sw   $2, 200($3)    (considere-se o código de operação 101011)
f) j    fatorial       (considere-se que fatorial se encontra em 0x4004000C)
g) bne  $2, $8, endif  (considere-se endif salta 5 instruções à frente)
h) jr   $ra            (considere-se o código de função 001000)
i) jal  bubblesort     (considere-se o código de operação 000011 e endereço 0x4004000C)
```

2. Qual é a diferença, em termos de arquitetura (a nível físico), entre os registos $s e os registos $t?

3. Como é que se processa a tradução dos programas em Assembly do MIPS, relativamente a instruções que contém etiquetas (labels), como a instrução jump? Por outras palavras, sabendo que nelas se escreve uma etiqueta, a que corresponde esta e como é que é feita a correspondência entre o seu significado real e a identificação de label?

4. Explica o processamento de toda a instrução jal.

5. De que tipo é a instrução jump register? Explica como é que a instrução se processa quando se executa, por exemplo, jr $ra.

6. Considerem-se as variáveis f, g, h, i e j estão atribuídas aos registos $s0-$s4, respetivamente. Consideremos também, que os endereços-base dos arrays A e B estão nos registos $s6 e $s7, respetivamente.

a) Como é que se traduz, para linguagem Assembly do MIPS, a seguinte operação em C?

f = g - A[B[4]];

b) A que corresponde a instrução "`lw  $s0, 4($s6)`", em C?

7. No simulador MARS estão incluídas duas instruções distintas para o cálculo da divisão entre registos (referimo-nos à instruções de `div`). Sendo uma destas uma pseudo-instrução, a qual recebe como registos de escrita e leitura os registos rs, rt e rd, como é que podemos implementar a divisão em instruções nativas? A pseudo-instrução em causa é a seguinte:

```
div    $s0, $s1, $s2          # grava em $s0 o valor de $s1/$s2
```

8. Qual a importância da salvaguarda de valores após a invocação de novas funções? É importante guardar o conteúdo de determinados registos sempre? Se não, em que casos é que é necessário e porquê?

9. Escreve em linguagem Assembly do MIPS, respeitando as convenções de utilização dos registos, uma função `int impar(int x)`, que retorna o valor 1 caso o argumento x seja ímpar, e 0 caso o argumento x seja par.

```
int impar(int x) {
  if (x % 2 == 0)
    return 0;
  else
    return 1;
}
```

10. Escreve em linguagem Assembly do MIPS a seguinte função `int bitCount(unsigned x)`, que conta o número de bits que se encontram a 1 ao longo de uma sequência de bits. Segue as convenções do MIPS, para a realização deste exercício.

```
int bitCount(unsigned x) {
  int bit;
  if (x == 0)
    return 0;
  bit = x & 0x1;
  return bit + bitCount(x >> 1);
}
```