ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Arquitectura de Sistemas e Computadores I

1º Frequência

Licenciatura em Engenharia Informática

27 de Março de 2012

Considere um processador MIPS32 com ordenação de bytes **little endian**.
Registos: zero, at, v0-v1, a0-a3, t0-t7, s0-s7, t8-t9, k0-k1, gp, sp, fp/s8, ra.

1. Considere a função `xpto` apresentada em baixo. Suponha que a função é carregada em memória no endereço `0x00400010` e o array `A` é carregado no endereço `0x10008004`.
   (a) Quantos bytes ocupa este programa em memória? Qual o endereço correspondente à *label* `L2`?
   (b) Analise a função `xpto` e explique qual a sua finalidade (pretende-se que explique o que faz a função como um todo e não uma descrição do que fazem as instruções. Por ex. calcula o factorial, converte string para maiúsculas, etc).
   (c) Escreva uma função `myfunction` que invoque `xpto` passando como argumento o array `A`.
   (d) No caso de ser passado o array `A` como argumento, indique quais as alterações feitas em memória e o valor devolvido pela função.
   (e) Preencha o conteúdo de cada byte na tabela da direita antes e depois da função executar (use um traço caso não conheça o valor).
   (f) Quantas instruções são executadas? Assumindo que cada instrução é executada num ciclo de relógio, e que a frequência de relógio é de 100MHz, quanto tempo demora a execução?
   (g) Optimize o código e calcule o *speedup* (quantas vezes é mais rápido?).

```
    .data
A:  .word 1, -1, 2, -2

    .text

# Args:
#  a0 - array address
#  a1 - number of elements
xpto:
    or   $v0, $zero, $zero
L1: lw   $t0, 0($a0)
    slt  $t0, $t0, $zero
    beq  $t0, $zero, L2
    nop
    sw   $zero, 0($a0)
    addi $v0, $v0, 1
L2: addi $a0, $a0, 4
    addi $a1, $a1, -1
    bne  $a1, $zero, L1
    nop
    jr   $ra
    nop
# end of xpto
```

Endereço	Antes	Depois
0x10008013		
0x10008012		
0x10008011		
0x10008010		
0x1000800f		
0x1000800e		
0x1000800d		
0x1000800c		
0x1000800b		
0x1000800a		
0x10008009		
0x10008008		
0x10008007		
0x10008006		
0x10008005		
0x10008004		
0x10008003		
0x10008002		
0x10008001		
0x10008000		
0x10007fff		

2. Converta as pseudo-instruções seguintes para instruções reais MIPS:
   (a) `li $t0, 0x10008004`
   (b) `bgt $t0, $t1, L1`
   (c) `move $t0, $t1`