

NOME: \_\_\_\_\_

Nº: \_\_\_\_\_

As questões teóricas (1 a 7) devem ser respondidas nesta mesma folha. As questões 1 a 4 são de escolha múltipla e, pelo menos, uma das respostas está correcta, valendo 0,75 valores. Uma resposta errada desconta 0,25 valores. As questões 5 e 6 valem 2 valores cada. A questão 7 vale 1,5 valores.

1. A lei de Amdhal indica que “o maior aumento de desempenho possível introduzindo melhorias numa determinada característica de um sistema é limitado pela percentagem em que essa característica é usada”. Considere um programa com  $T_{exec} = 2.40$  s, sendo 50% operações com inteiros, 30% leituras e escritas em memória e 20% operações com vírgula flutuante. Qual dos seguintes valores representa o tempo de execução se o acesso à memória for 6 vezes mais rápido?

- 0.40 s
- 1.00 s
- 1.80 s
- 2.40 s

2. Considere a seguinte struct em C: `struct {int a; char b; char c; char d; char e} v[800]`; Qual das seguintes instruções em C é implementada pela instrução “`movb 5(%ebx,%ecx,8), %eah`” (*assembly* do IA32). Admita que `%eah` está associado a `char x`, `%ebx` contém o endereço base de `vector` e `%ecx` contém o valor 98?

- `x = v[98].b;`
- `x = v[99].e;`
- `x = v[98].c;`
- `x = v[98].e;`

3. Sabendo que  $T_{exec} = \#I * CPI * T_{cc}$ , ao aumentar 4 vezes a frequência do relógio do processador resulta:

- num  $T_{exec}$  maior, pois frequências mais altas correspondem a maiores tempos de execução.
- em diminuir  $T_{exec}$  para 1/4, pois o ciclo do relógio ( $T_{cc}$ ) diminui 4 vezes.
- em diminuir  $T_{exec}$  para mais de 1/4, devido a atrasos associados a outros componentes, tais como os barramentos e memória.
- em diminuir  $T_{exec}$  para mais de 1/4, devido a um aumento no número de instruções

4. O princípio da localidade temporal, um dos responsáveis pelo sucesso da hierarquia de memória, ...

- consiste em aceder várias vezes a endereços de memória contíguos, num dado intervalo temporal.
- observa-se em programas que acedem aleatoriamente aos elementos das suas estruturas de dados.
- observa-se particularmente na execução de ciclos, quando é repetido um pequeno número de instruções.
- é explorado de forma bastante eficiente na execução de chamadas de funções e procedimentos.

5. Na compilação de uma função, escrita em C, o compilador gera código para criar, para cada invocação uma estrutura designada por *activation record*, constituída por vários campos e que mantém o contexto da função. Descreva esta estrutura e o papel de cada um dos seus campos, tal como é gerada pelo `gcc`. Para facilitar a sua resposta represente-a graficamente juntamente com o apontador que para ela é mantido.


6. Em 1945, von Neumann definiu um conjunto de conceitos sobre a construção de uma máquina de cálculo universal, que se passou a designar por “Máquina de von Neumann”. Esses conceitos podem ser resumidos em 3 contribuições fundamentais: (1) uma organização para esta máquina; (2) uma arquitectura (conjunto de instruções); (3) o conceito de “*stored program*”. Caracterize brevemente cada uma destas contribuições.


7. Indique os vários estágios de execução da instrução `beq $t2, $t3, -12`, explicitando os elementos do datapath single cycle do MIPS utilizados (ver figura anexa), as operações realizadas em cada fase, e indique o valor dos sinais de controlo `RegDst`, `RegWrite`, `ALUSrc`, `PCSrc`, `MemWrite`, `MemRead` e `MemToReg`.


NOME: \_\_\_\_\_

Nº: \_\_\_\_\_

**PRÁTICA**

P8a: \_\_\_\_\_

P8b: \_\_\_\_\_

P8c: \_\_\_\_\_

P8d: \_\_\_\_\_

Σ: \_\_\_\_\_

A questão 8 vale 4 valores, a questão 9 vale 2,5 valores e a questão 10 vale 2 valores.
---

8. Considere o bloco de programa escrito em linguagem C que calcula a nota média numa lista de alunos.

<pre>typedef struct{     char nome[22];     int nota; }aluno;  aluno alunos[128];</pre>	<pre>int NotaMedia(aluno * lstalunos){     int i,soma;     soma=0;      for (i=0;i&lt;128;i++){         soma+=lstalunos[i].nota;     }     return (int)(soma/128); }</pre>
---	--

- (a) Complete o programa em baixo, nas linhas marcadas de 1 a 6, de modo a corresponder a uma compilação válida do programa em C.

<pre>NotaMedia:     pushl %ebp     movl  %esp, %ebp     subl  \$12, %esp     movl  \$0, -8(%ebp)     /*1*/ movl  \$0, _____ .L6:     cmpl  \$127, -4(%ebp)     /*2*/ jg   _____ .L7:     movl  -4(%ebp), %edx     movl  %edx, %eax     sall  \$3, %eax     subl  %edx, %eax</pre>	<pre>    leal  0(,%eax,4), %edx     movl  8(%ebp), %eax     movl  24(%edx,%eax), %edx     leal  -8(%ebp), %eax .L8:     /*3*/ addl  %edx, _____     /*4*/ leal  _____, %eax     incl  (%eax)     /*5*/ jmp   _____ .L9:     /*5*/ movl  _____, %eax     sarl  \$7, %eax     leave     ret</pre>
---	---

- (b) Identifique e explique as instruções responsáveis pelo teste do ciclo `for`.


(c) Identifique e explique as instruções responsáveis pelo cálculo da soma: `soma+=1stalunos[i].nota`


(d) Que alterações seria necessário fazer, no programa assembly, caso a lista de alunos passasse a ter 256 alunos.


NOME: \_\_\_\_\_

Nº: \_\_\_\_\_

9. Considere o seguinte fragmento de código, escrito em assembly do MIPS:

```

slt    $a1, $t0, $t1
ble    $s0, $s1, $L1
move   $fp, $sp
sw     $t2, 12($s1)
lw     $a0, -4($fp)
$L1:
sll    $s0, $s1, 2    # QB
li     $v0, 0xFA00FF

```

P9a: __
P9b: __
P9c: __
P10a: __
P10b: __
Σ: ____


a) Reescreva o código fornecido, usando apenas instruções nativas do MIPS.

b) Converta a instrução assinalada com "# QB" para código máquina do MIPS, apresentando o resultado final em hexadecimal e incluindo na resposta todos os passos intermédios usados.

c) Converta para assembly do MIPS a instrução máquina **0x A171FFFA** (em hexadecimal), apresentando os cálculos intermédios.

**10.** Considere uma máquina com uma frequência do relógio de 1.5 GHz e uma *cache* organizada em blocos de 1 palavra. O tempo de acesso à memória central é composto por uma latência de 20 ns mais 10 ns por cada palavra. A máquina necessita, em média, de 2.58 ciclos para executar cada instrução, sem incluir os acessos à memória central. Esta máquina é usada para executar um programa com  $10^9$  instruções (20% das quais implicam um acesso a dados em memória) e exhibe uma *miss rate* de instruções de 6%.

**a)** Se o tempo de execução deste programa for de 4s, qual a *miss rate* de dados?



**b)** Para tirar partido da localidade espacial, o tamanho dos blocos da *cache* foi aumentado para 4 palavras. Esta alteração resultou numa diminuição da *miss rate* de instruções para 4% e de dados para 6%. Qual o ganho relativamente às condições da alínea anterior? Como justifica este resultado?

