Preencher na correcção
1:_____
2:_____
3:_____
4:_____
5:_____
6:_____
7:_____
8:_____

F:_____

Arquitectura de Computadores I - 2006/07 Exame 1ª chamada – 09.Jan.07

NOME: _____ **Nº:**_____

Estas questões devem ser respondidas na própria folha do enunciado. As questões 1 a 4 são de escolha múltipla, e apenas uma das respostas está correcta, valendo 1 valor. Uma resposta errada desconta 1/3 de valor. As questões 5 e 6 valem 3 valores cada. A questão 7 (2 valores) só deve ser respondida pelos alunos que, justificadamente, não fizeram a componente teórico-prática.

1. Considere o seguinte código em C:

```
struct {int a; char b} lista[1000];
int i=10, x;
x = lista[i].b;  /***/
```

Qual das seguintes sequências de instruções *assembly* do IA32 implementa a instrução C assinalada com asteriscos, sabendo que `%esi` está associado a `x` e `%eax` contem o valor de `i`?

☐ `sall 4, %eax`
`leal lista, %ebx`
`movb 4(%ebx, %eax), %esi`
	☐ `sall 2, %eax`
	`leal lista, %ebx`
	`movb 4(%ebx, %eax), %esi`

☐ `movb lista+4(,%eax,8), %esi`	☐ `leal lista, %ebx` `movb (%ebx,%eax,8), %esi`

2. Relativamente ao *datapath* de ciclo único do MIPS podemos dizer que:

☐ É ineficiente pois todas as instruções necessitam do mesmo período de tempo para executar.

☐ Permite obter o melhor desempenho pois o CPI de qualquer instrução é 1.

☐ É eficiente pois cada instrução apenas necessita dos períodos de tempo associados às várias fases da sua execução.

☐ É eficiente pois cada componente apenas é usado uma vez para a execução de cada instrução.

3. Uma técnica de optimização utilizada pelos compiladores consiste em armazenar as variáveis mais usadas em registos. Esta técnica aumenta o desempenho porque:

☐ Uma vez que os acessos a registos são mais rápidos do que acessos a memória a frequência do relógio pode ser maior.

☐ No IA32 esta técnica resulta em programas com menor número de instruções.

☐ Uma vez que os acessos a registos são mais rápidos do que acessos a memória o CPI deste programa é menor.

☐ A utilização de registos permite processar vários elementos de dados por instrução.

4. Considere uma máquina com um espaço de endereçamento de 32 *bits*, 4 MBytes de *cache*, com linhas de 16 palavras e 32 *bits* por palavra. Sabendo que o mapeamento de endereços na *cache* é 8-*way set associative*, quantos bits são necessários para a *tag*?

☐ 14

☐ 16

☐ 13

☐ 12

NOME: _____ **Nº:**_____

5. Indique, justificando, o valor dos sinais de controlo `RegDst`, `RegWrite`, `ALUSrc`, `PCSrc`, `MemWrite`, `MemRead` e `MemToReg`, para a execução da instrução `lw $s0, 100($t0)`, para o *datapath* de ciclo único do MIPS (ver figura anexa).

NOME: _____ **Nº:**_____

6. O tempo de execução de um programa numa determinada máquina pode ser caracterizado recorrendo a 3 parâmetros: número de instruções executadas (#I), número médio de ciclos por instrução (CPI) e frequência do relógio (f). As técnicas de optimização do código afectam um ou mais destes parâmetros. Indique, justificando, qual o parâmetro afectado pela técnica conhecida como *loop unrolling*.

NOME: _____ **Nº:**_____

Apenas para os alunos que, justificadamente, não fizeram a componente teórico-prática.

7. Os mecanismos de invocação e retorno de procedimentos e funções exigem a execução de um conjunto de passos que garantem a correcta implementação da semântica definida pela linguagem de alto nível utilizada (ex.: passagem de parâmetros, controlo de fluxo, etc.). Enumere e descreva estes passos, indicando ainda como são implementados para a arquitectura IA32 e a linguagem C.

NOME: _____ **Nº:**_____

Estas questões devem ser respondidas em folha separada e valem 8 valores.

8. Considere o código *assembly* do MIPS (metade inferior da tabela) obtido após compilar a função apresentada na metade superior da mesma tabela.

```
int somatorio (int n, int d[])
{
  int i, s=0;

  for (i=0; i < n; i++)
     s += d[i];
  return (s);
}
```

```
somatorio:
    li $t1, 0                # B1
    li $t0, 0
    b teste
ciclo:
    sll $t2, $t0, 2          # B2                    # CONV
    add $t2, $a1, $t2        # B2
    lw $t2, 0($t2)           # B2                    # CONV
    add $t1, $t1, $t2
    addi $t0, $t0, 1         # B3
teste:
    slt $t3, $t0, $a0        # B4
    bne $t3, $zero, ciclo    # B4                    # CONV
    move $v0, $t1            # B5
    jr $ra
```

a) Identifique a funcionalidade dos blocos de instruções etiquetados de **B1 a B5**, associando com o código correspondente em C e descrevendo cada passo.

b) Converta para código máquina as instruções etiquetada com `CONV`. Apresente todos os passos intermédios e converta o resultado final para hexadecimal.

c) Escreva em *assembly* do IA32 o código correspondente à instrução **`s += d[i];`** considerando que o valor de `i` está no registo `%esi` e `s` está associada ao registo `%eax`.

d) Esta função, executada com um valor de `n`=1000 e numa máquina com uma frequência de relógio de 2 GHz, consumiu 10518 ciclos de relógio.
Calcule o tempo de execução.
Qual o número médio de ciclos por instrução (CPI) verificado para esta função nesta máquina?

e) Optimize o código da função `somatorio()` desenrolando o ciclo de forma a processar 4 elementos do vector por iteração. Assuma que o valor de `n` é sempre múltiplo de 4.

f) Assumindo que o CPI desta nova versão é o mesmo que para a versão anterior, qual o tempo de execução da versão optimizada, para `n`=1000, na mesma máquina?