T1:_____
T2:_____
T3:_____
T4:_____
T5:_____
T6:_____
T7:_____
T: _____
P:_____

Arquitectura de Computadores I - 2002/03 Exame 2ª chamada – 22.Jan.03

NOME: _____ **Nº:**_____

TEÓRICA

As 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 2 valores cada. A questão 7 (4 valores) só deve ser respondida pelos alunos que, justificadamente, não fizeram a componente teórico-prática.

1. O conceito *stored program* (John von Neumann, 1945) refere-se à capacidade de:

☐ Armazenar os programas em memória secundária (ex.: discos), onde podem ser acedidos sempre que necessário.

☐ Programar o computador para executar diferentes tarefas carregando as sequências de instruções apropriadas (programas) na memória central, sem necessidade de alterar a estrutura e organização do mesmo.

☐ Armazenar numa memória não volátil o código necessário para gerir, controlar e coordenar os diferentes componentes do computador.

☐ Armazenar os programas mais usados em dispositivos de memória mais rápidos e não voláteis, para que a sua execução seja mais rápida.

2. O mecanismo de interrupções é uma das formas utilizadas de comunicação entre o processador e os controladores de periféricos. Podemos afirmar que:

☐ Este liberta o processador da necessidade de verificar activamente alterações no estado dos controladores.

☐ Os controladores apenas interrompem o processador para comunicar o fim de operações de I/O.

☐ Utilizando este mecanismo o processador fica liberto de qualquer envolvimento activo na transferência de dados entre a memória e os controladores.

☐ Utilizado juntamente com o DMA (D*irect Memory Access*) este mecanismo liberta o processador de qualquer responsabilidade nas operações de I/O.

3. A localidade espacial é uma propriedade exibida pela maioria dos programas. Podemos afirmar que:

☐ A organização da memória tira partido desta característica agrupando acessos a um nível inferior da hierarquia (ex.: acesso da *cache* à memória central) em blocos de palavras, em vez de aceder palavra a palavra.

☐ É responsável pelo efeito acelerador da hierarquia da memória, ao contribuir para que blocos de dados e/ou instruções acedidos repetidamente pelo processador sejam, com grande probabilidade, encontrados na *cache* após o primeiro acesso.

☐ Se traduz pelo facto de determinadas variáveis e/ou porções de código serem repetidamente acedidas pelos processadores.

☐ Se traduz pelo facto de que a grande maioria dos blocos de dados e código acedidos por um programa se situarem num intervalo limitado do espaço de endereçamento.

4. O programa P tem $2*10^8$ instruções (50% das quais implicam um acesso à memória). Quando executado na máquina M, exibe uma *miss rate* de instruções de 5%, de dados de 10% e um CPI_{CPU} de 2 ciclos incluindo o *hit time*. A máquina M tem uma *cache* com linhas de 4 palavras e um acesso à memória central tem uma latência de 10 ns seguida de um tempo de 10 ns por palavra. Sabendo que a frequência do relógio de M é de 2 GHz, qual dos seguintes valores corresponde ao tempo de execução de P em M:

☐ 2,2 seg.

☐ 1,2 seg.

☐ 0,7 seg.

☐ 1,7 seg.

NOME: _____ **Nº:**_____

5. Indique os vários estágios de execução da instrução `beq $s0, $t5, -50`, 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:_____

6. As arquitecturas RISC disponibilizam, ao nível do conjunto de instruções, um leque de modos de endereçamento mais restrito do que as arquitecturas CISC. Indique quais as razões que justificam esta abordagem mais minimalista e de que forma esta se reflecte no código *assembly* gerado pelos compiladores. Use como exemplos o *assembly* do IA32 e do MIPS.

NOME: _____ Nº:_____

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

7. O tempo de execução de um programa num determinado processador pode ser previsto calculando o produto de 3 métricas: número de instruções (#I), ciclos por instrução (CPI) e o período do relógio (Tcc). Discuta os vários factores que influenciam cada uma destas métricas (ex.: algoritmo, compilador, conjunto de instruções, microarquitectura, tecnologia de desenvolvimento do processador).

NOME: _____ **Nº:**_____

PRÁTICA

As questões práticas devem ser respondidas em folha separada. A questão 1 vale 4 valores, as questões 2 e 3 valem 2 valores cada.

1. Considere que o código assembly IA32 (sem optimização) obtido após compilar a função da tabela seguinte é o que se apresenta na coluna direita da mesma tabela.

```
typedef struct {
   char tipo;
   int  p;
   int  s;
   int  f;
   } pauta;

int calcular(char n, pauta *nota)
   {
   int na, j;
   for (j=20, na=0; j > 0; --j)
      if (nota[j].tipo >= n)
         ++na;
      else
         nota[j].f = somar (
            nota[j].p,nota[j].s);
   return (na+1);
   }
```

```
calcular:
   pushl %ebp
   movl  %esp, %ebp
   subl  $24, %esp          # ******
   movl  8(%ebp), %eax         # B 1
   movb  %al, -1(%ebp)         # B 1
   movl  $20, -12(%ebp)        # B 1
   movl  $0, -8(%ebp)          # B 1
.L3:
   cmpl  $0, -12(%ebp)
   jg    .L6
   jmp   .L4
.L6:
   movl -12(%ebp), %eax        # B 2
   movl  %eax, %edx            # B 2
   sall  $4, %edx              # B 2
   movl  12(%ebp), %eax        # B 2
   movb  (%eax,%edx), %al      # B 2
   cmpb  -1(%ebp), %al         # B 2
   jl    .L7                   # B 2
   leal  -8(%ebp), %eax
   incl  (%eax)
   jmp   .L5
.L7:
   subl  $8, %esp
   movl  -12(%ebp), %eax
   movl  %eax, %edx
   sall  $4, %edx
   movl  12(%ebp), %eax
   pushl 8(%eax,%edx)
   movl  -12(%ebp), %eax       # B 3
   movl  %eax, %edx            # B 3
   sall  $4, %edx              # B 3
   movl  12(%ebp), %eax        # B 3
   pushl 4(%eax,%edx)          # B 3
   call  somar                 # B 3
   addl  $16, %esp
   movl  %eax, %ecx            # B 4
   movl  -12(%ebp), %eax       # B 4
   movl  %eax, %edx            # B 4
   sall  $4, %edx              # B 4
   movl  12(%ebp), %eax        # B 4
   movl  %ecx, 12(%eax,%edx)   # B 4
.L5:
   leal  -12(%ebp), %eax       # B 5
   decl  (%eax)                # B 5
   jmp   .L3                   # B 5
.L4:  movl  -12(%ebp), %eax    # B 6
   decl  %eax                  # B 6
   leave                       # B 6
   ret
```

NOME: _____ Nº:_____

a) Identifique a funcionalidade dos blocos **1 a 5** de instruções etiquetados.

b) Identifique 2 erros no bloco **6**. Em vez de 24, qual o valor mínimo que se poderia usar na instrução etiquetada com " # ****** " ?.

c) Escreva em assembly do MIPS o código correspondente à seguinte linha de código. **Notas:** considere que um char ocupa 1 byte, um inteiro ocupa 4 bytes e não precisa escrever a instrução de salto.

if (nota[j].s > na)

Assuma que:

- o registo **$s1** contém a variável inteira **na**
- o registo **$t2** contém o apontador para a variável estruturada **nota**
- o registo **$t3** contém a variável inteira **j**

2. Converta o seguinte programa para assembly do MIPS (apresente todas as constantes em decimal):

```
0x200D0020
0x2002FFF6
0x01A26820
0x1DA0FFFE
```

3. Considere uma máquina com um espaço de endereçamento de 64 *bits*, uma cache com uma capacidade de 512 Kbytes de dados, palavras de 64 *bits*, blocos de 16 palavras e mapeamento directo.

a) Qual a distribuição dos *bits* do endereço pela *tag*, índice, *block offset* e *byte offset* para seleccionar o *byte* correcto na *cache*?

b) Qual a capacidade total da *cache,* contando com os campos de controlo (*tag* e *valid bit*)?

c) Se a organização da *cache* fosse *2-way set associative* a capacidade total da cache seria maior ou menor? Justifique.