Arquitectura de Sistemas e Computadores II
2ª Frequência

Departamento de Informática
Universidade de Évora

17 de Dezembro de 2014

Indique todos os cálculos efectuados

Perguntas rápidas

1. [0,5 valores] Quantos blocos tem cada conjunto de uma cache *3-way set associative*?

2. [0,5 valores] Considere duas caches com o mesmo número de blocos e o mesmo número de palavras por bloco, uma *direct-mapped* e a outra *fully associative*. Em geral, em qual se obterá maior *hit rate*?

3. [0,5 valores] Um processo pode alterar directamente o conteúdo da sua tabela de páginas?

4. [0,5 valores] Que tipo de escalabilidade é estudada quando o mesmo programa é executado variando só o número de processadores utilizados?

Caches

Considere que uma palavra tem 32 bits e que os endereços seguintes são acedidos pela ordem indicada:

```
53 42 54 32 26 21 53
```

5. [3 valores] Simule o funcionamento de uma cache *direct-mapped*, com 4 palavras e blocos de 1 palavra, para a sequência de acessos indicada. Assuma que a cache inicialmente está vazia e, para cada acesso, indique a palavra acedida, o número do bloco a que pertence, o índice da posição que vai ocupar na cache e se há um *hit* ou um *miss*. Apresente o conteúdo final da cache, incluindo o *tag*, e calcule a *miss rate* verificada.

6. [1 valor] Num sistema com a cache da pergunta anterior, quantos ciclos de relógio será atrasada a execução do programa, devido à sequência de acessos apresentada, se o *hit time* for 1 ciclo e a *miss penalty* for 20 ciclos. (Se não respondeu à pergunta 5, utilize o valor 40% para a *miss rate*.)

7. [2 valores] Simule o funcionamento, para os 3 primeiros acessos da sequência apresentada, de uma cache *2-way set associative*, com 2 conjuntos, blocos de 4 palavras e usando a estratégia LRU na substituição de blocos. Assuma que a cache inicialmente está vazia e, para cada acesso, indique o número do bloco a que pertence a palavra acedida, o índice da posição da cache que irá ocupar, se há um *hit* ou um *miss* e, quando aplicável, o número do bloco que será substituído. Apresente o conteúdo final da cache.

Memória virtual

8. [1,5 valores] Qual o espaço ocupado pela tabela de páginas de um processo num sistema com endereços virtuais de 40 bits, páginas de 16 KB, e em que cada entrada da tabela ocupa 4 *bytes*?

9. Num momento da execução de um programa, todas as páginas físicas estão em uso, a página física 20 é a que não é acedida há mais tempo, e o TLB (*direct-mapped*, com 4 blocos de uma tradução) do sistema e a tabela de páginas do programa têm os conteúdos (parcialmente) mostrados:

TLB

	Valid	Dirty	Tag	Pág. física
0	1	0	50	5
1	1	0	33	10
2	1	1	25	8
3	1	1	24	2

Tabela de páginas

	Dirty	Pág. física
	...	
99	0	2
100	1	20
101	0	DISCO
102	1	8
103	0	13
	...	

(a) [2 valores] Qual é o número da página virtual a que corresponde o conteúdo da posição 0 do TLB?

(b) [2,5 valores] Descreva o que acontece se nesta situação for acedido um endereço da página virtual 101. Mostre os conteúdos resultantes do TLB e da tabela de páginas.

Multiprocessamento

10. [2 valores] Um programador tem uma aplicação que quer transformar para tirar proveito de um novo processador com 2 *cores*. A aplicação consiste em cinco tarefas, que demoram os tempos na tabela abaixo:

Tarefa	t_1	t_2	t_3	t_4	t_5
Tempo	1s	5s	3s	4s	5s

Os resultados da tarefa t_1 são usados pelas restantes tarefas, que são independentes entre si.

Se cada tarefa tiver de ser executada num só *core*, como deverá ele distribuir as tarefas pelos *cores* disponíveis, de modo a minimizar o tempo de execução da aplicação?

Com essa distribuição, qual o *speedup* obtido em relação à execução sequencial?

11. Pretende-se implementar um mecanismo para a sincronização de processos num sistema multiprocessador MIPS de memória partilhada, que permita garantir que nenhuma *thread* do programa continua a execução para lá de um ponto de sincronização até todas as *threads* terem alcançado esse ponto. A base desse mecanismo é uma função `espera` que todas as *threads* devem invocar no ponto de sincronização.

As versões C e MIPS (sem *delay slots*) da implementação proposta para a função são apresentadas abaixo. O valor inicial da variável (partilhada) `faltam` é o número de *threads* que devem sincronizar naquele ponto.

```
void espera()
{
  faltam = faltam - 1;

  while (faltam > 0)
    ;
}
```

```
espera: lw $t0, faltam($0)
        addiu $t0, $t0, -1
        sw $t0, faltam($0)
testa:  beq $t0, $0, fim
        lw $t0, faltam($0)
        j testa
fim:    jr $ra
```

(a) [2 valores] Explique a razão por que a implementação proposta pode não funcionar como se pretende. Exemplifique usando duas *threads* e 2 como valor inicial de `faltam`.

(b) [2 valores] Apresente uma versão MIPS que tenha o efeito pretendido.