Licenciatura em Engenharia Informática – Arquitectura de Computadores

Primeiro teste – 28 de Março de 2012 – Duração: 2 horas

Responda às questões no espaço reservado para o efeito após cada alínea, ou na última página caso necessite mais espaço. Apresente todos os cálculos que efectuar, e esclareça qualquer pressuposto que tenha de fazer para resolver a uma questão.

Número		Nome	Resolução

1. [5 valores] Complete a implementação em C da seguinte função **recursiva** para encontrar o elemento mínimo entre os elementos de um array de inteiros apontado por "a" com "n" elementos. Pode pressupor que na invocação inicial n>=1 e que "a" aponta para um array correctamente alocado para albergar os n elementos. A base da recursão é dada no código abaixo, e cada passo da recursão (que terá de implementar) deverá invocar a própria função exactamente duas vezes.

```
int minimo(int *a, int n) {
   int m1,m2;
   if (n==1)
      return *a;
   m1 = minimo(a,n/2);
   m2 = minimo(a+n/2,n/2+n%2);
   if (m1<m2)
      return m1;
   else
      return m2;

}
```

2. [4 valores] Qual a saída (output) do seguinte código em C?

```
int a[3];
int *p1, *p2;
a[0] = 10;
a[1] = 20;
a[2] = 30;
p1 = a;
p2 = p1;
p1++;
printf ("%d %d %d\n",a[0],a[1],a[2]);
(*p2)++;
printf ("%d %d %d\n",a[0],a[1],a[2]);
*p2=*(p1+1);
printf ("%d %d %d\n",a[0],a[1],a[2]);
p1 = &(a[2]);
if (p1 == p2) {
  printf ("Igualdade um\n");
}
if (*p1 == *p2) {
  printf("Igualdade dois\n");
}
```

10 20 30
11 20 30
30 20 30
Igualdade dois

3. [5 valores] Considere os números decimais **-2** e **-4**.
    a. Efectue a soma dos dois números em complemento para dois. Deve apenas utilizar esta representação, não recorrendo a nenhuma tradução ou operações noutras representações, por exemplo decimal. Mostre todos os seus cálculos.

Partindo do pressuposto que vamos usar 8 bits para a representação binária:
-2 = 11111110
-4 = 11111100
---------- (efectuando a aritmética normal para a soma binária)
11111010 = -6

    b. Será que o mesmo método de cálculo poderia ser usado na representação em complemento para um? Justifique usando o mesmo exemplo da alínea anterior.

Não pois a aritmética normal para a soma binária não funcionaria neste caso, como se demonstra pelo exemplo anterior:

-2 = 11111101
-4 = 11111011

11111000 = -7 (resultado errado)

    c. Proponha um método de cálculo alternativo para o caso da alínea anterior e dê uma justificação intuitiva para o motivo pelo qual esse método estará correcto.

O método consiste em fazer uma soma binária normal e, tendo em conta que ambos os números são negativos neste caso, adicionar uma unidade ao resultado final, que ficaria assim igual a 11111001 = -6 em complemento para um.

A justificação intuitiva é que a representação em complemento para um tem dois zeros, e como tal, se dispusermos os números binários numa circunferência em que os números consecutivos correspondem a adicionar uma unidade de acordo com a aritmética normal, ao transitar nesta circunferência da parte positiva para a parte negativa temos de subtrair uma unidade extra para corrigir a existência dos dois zeros. Mas ao somar dois números negativos estamos a incluir esta correcção duas vezes e portanto temos de voltar a somar uma unidade para compensar.

Número: Nome:

4. [6 valores] Um grupo de alunos da cadeira de AC está a pensar na próxima geração da arquitectura MIPS, na qual os endereços e as instruções terão 64 bits, e a arquitectura incluirá 64 registos de 64 bits. Ao longo desta pergunta, mostre e justifique todos os cálculos que efectuar.
    a. Qual o número de Bytes armazenado no conjunto de todos os registos?

64 registos X 64 bits = 2^6 X 2^6 = 2^12 bits = 2^12/8 Bytes = 2^12/2^3 = 2^9 = 512 Bytes

    b. Suponha que tem de desenhar o formato das instruções na nova arquitectura, e decide ter um campo de "opcode" de 8 bits, e que para o formato R são usados os bits necessários para os campos dos registos e do "shift amount", sendos os restantes bits dedicados ao campo da função. Quantas instruções diferentes são suportadas nesta nova arquitectura? (Deve-se referir a tipos de instruções, ou seja, addi $t0,$t1,1 e addi $s0,$s1,2 são a mesma instrução.) Para simplificar a resolução, o resultado final pode ser apresentado sob a forma de uma soma de potências de dois, por exemplo, se o resultado for $2^{50}+2^{40}+1$, pode ser apresentado desta forma, não tendo de calcular 2^{50} nem 2^{40}.

O novo formato R tem 8 bits para o opcode, 6 bits para cada um de Rs, Rt, Rd (de forma a endereçar 64 registos), 6 bits para o shift amount (de forma a deslocar entre 0 e 63 bits); como tal sobram os restantes 64-8-6-6-6-6 = 32 bits para a função.
Logo para as instruções tipo I e J temos 2^8-1 diferentes instruções (ou seja, todos os opcodes menos o zero), e para as instruções tipo R temos 2^32 instruções.
Resposta: 2^32+2^8-1

    c. Será que os criadores da arquitectura MIPS concordariam ou discordariam do aumento do número de instruções da linguagem? Porquê?

Discordariam pois este aumento poderia trazer mais complexidade ao nível do hardware necessário para suportar as novas instruções, e poderia implicar que a execução de cada instrução se tornasse mais lenta, e que houvesse menos espaço, por exemplo, para registos adicionais. Tal contraria a filosofia RISC que serve como base ao MIPS.

    d. Voltando à arquitectura original (32 bits), suponha que cada uma das instruções seguintes está armazenada na localização de memória (em hexadecimal) ABCD1234. Qual o endereço da instrução que será executada de seguida a cada uma destas instruções?
        i. 000000 00001 00010 00011 00000 100000

Opcode = 0 (tipo R), funct = 20hex, instrução add (ver na folha verde do MIPS)
Como tal executa-se a instrução seguinte em memória, no endereço ABCD1238hex

        ii. 000011 11111111111111111111111100

Opcode = 3, instrução jal
Como tal PC salta o endereço: (PC_actual+4)[31:28], campo_endereço, "00". Neste caso: AFFFFFF0hex