Licenciatura em Engenharia Informática – Arquitectura de Computadores

Exame de época normal – 19 de Junho de 2012 – Duração: 2,5 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. [2 valores] Considere o seguinte código em C que adiciona um elemento a uma lista. Escreva a correspondente função para retirar o elemento que está na cabeça da lista, retornando o respectivo valor armazenado. Note que para a reposta estar completamente correcta o programa não pode conter fugas de memória (memory leaks).

```
struct elem {
  int val;
  struct elem *prox;
};
typedef struct elem elem_t;

elem_t *lista; /* var. global
                  inicializada a NULL */

void insere(int n) {
 elem_t *novo = (elem_t *)malloc(sizeof(elem_t));
 novo->prox = lista;  //      (*novo).prox
 novo->val = n;
 lista = novo;
}
```

```
int remover_cabeca_lista() {
    elem_t * ptr = lista;
    lista = lista -> prox;
    free (ptr);
}
```

2. [2 valores] Implemente a função strcmp que compara duas strings. Na versão simplificada desta função que deverá implementar, a função recebe como argumentos dois ponteiros para caracteres que representam duas strings terminadas com o caracter '\0', e retorna o valor 1 se estas forem diferentes ou 0 se forem iguais.

```
int strcmp(char *s1, char *s2) {
    while (1) {
        if (*s1 != *s2) return 1;
        if (*s1 == '\0' && *s2 == '\0') return 0;
        s1++;
        s2++;
    }
}
```

3. [3 valores] Considere a seguinte sequência de 8 bits: 1100 0010

Qual o seu significado se forem interpretados como sendo:

a) Um número em representação de sinal e amplitude (indique a sua resposta em decimal).

$-(2^7+2^6+2^1) = -128-64-2 = -194$

b) Um número em representação de complemento para dois (indique a sua resposta em decimal).

11000010
> 11100001
00111110

$-(2^5+2^4+2^3+2^2) = -62$

c) Um número em representação de vírgula flutuante em que são usados 1 bit para o sinal, seguido de 4 para o expoente e 3 para a fracção, e o enviesamento é de 7 (indique a sua resposta em decimal).

$-1,010_2 \times 2^{8-7} = -10,10_2 = -(2^1+2^{-1}) = -2,5$

d) Uma instrução MIPS, em que estes oito bits ocupam os bits menos significativos da instrução, e os restantes bits (à esquerda destes oito) são zeros.

```
srl  $0, $0, 3
```

(opcode=0, funct=2, rs=rt=0, shamt=3)

Número: Nome:

4. [2.5 valores] Considere o seguinte segmento de código assembly MIPS, em que o registo $s0 armazena o valor da variável x, e o registo $s1 da variável y.

Loop:	slt $t0, $s1, $s0	# if ($s1 < $s0) $t0=1 else $t0=0
	beq $t0, $zero, Fim	# if ($t0 == 0) goto Fim
	addi $s0, $s0, -1	# $s0 --;
	j Loop	# goto Loop
Fim:		

a) No comentário no final de cada linha, indique qual o pseudo-código correspondente a essa instrução assembly. Exemplo do tipo de pseudo-código que deve escrever: **`if (x!=$t0) goto Loop`**

b) Escreva o código C correspondente a este assembly usando o menor número instruções em C possível.

```
while (y< x--) ;
```

5. [2.5 valores] Complete a implementação em MIPS das seguintes funções: a função somaVector que recebe como argumentos um apontador para o início do array e a respectiva dimensão e soma todos os elementos do array, e a função mediaVector com os mesmo argumentos, e que calcula a média dos elementos (invocando para tal a primeira função). Deve usar exactamente os espaços reservados para cada instrução e a instrução deve equivaler ao respectivo comentário.

```
somaVector:
      add $t0, $zero, $zero    # $t0 acumula a soma dos elementos do array
loop: beq $a1, $zero, exit     # se o número de elemtos a percorrer é zero sai
      lw $t1, 0($a0)           # obtem próximo elemento inteiro da memória
      add $t0, $t0, $t1        # soma o valor do elemento ao acumulador
      addi $a0, $a0, 4         # passa para o próximo inteiro no vector
      addi $a1, $a1, -1        # decrementa o número de elementos a percorrer
      j loop
exit: add $v0, $t0, $zero      # retorna o valor contido em $t0
      jr $ra

mediaVector:
      addi $sp, $sp, -8        # armazena dados que serão necessários mais tarde
      sw $a1, 4($sp)
      sw $ra, 0($sp)
      jal somaVector
      lw $a1, 4($sp)           # repõe dados armazenados
      lw $ra, 0($sp)
      addi $sp, $sp, 8
      div $v0, $a1             # calcula o valor de retorno
      mflo $v0
      jr $ra
```

6. [6 valores] Nesta pergunta pretende-se adicionar ao conjunto de instruções do MIPS uma instrução chamada *addmi* que efectua a adição entre o conteúdo de um registo e um valor imediato, armazenando o resultado no endereço de memória contido no registo rt (ao contrário da instrução existente de soma de um registo com um imediato que armazena o resultado num registo).
A especificação em RTL da instrução é a seguinte:

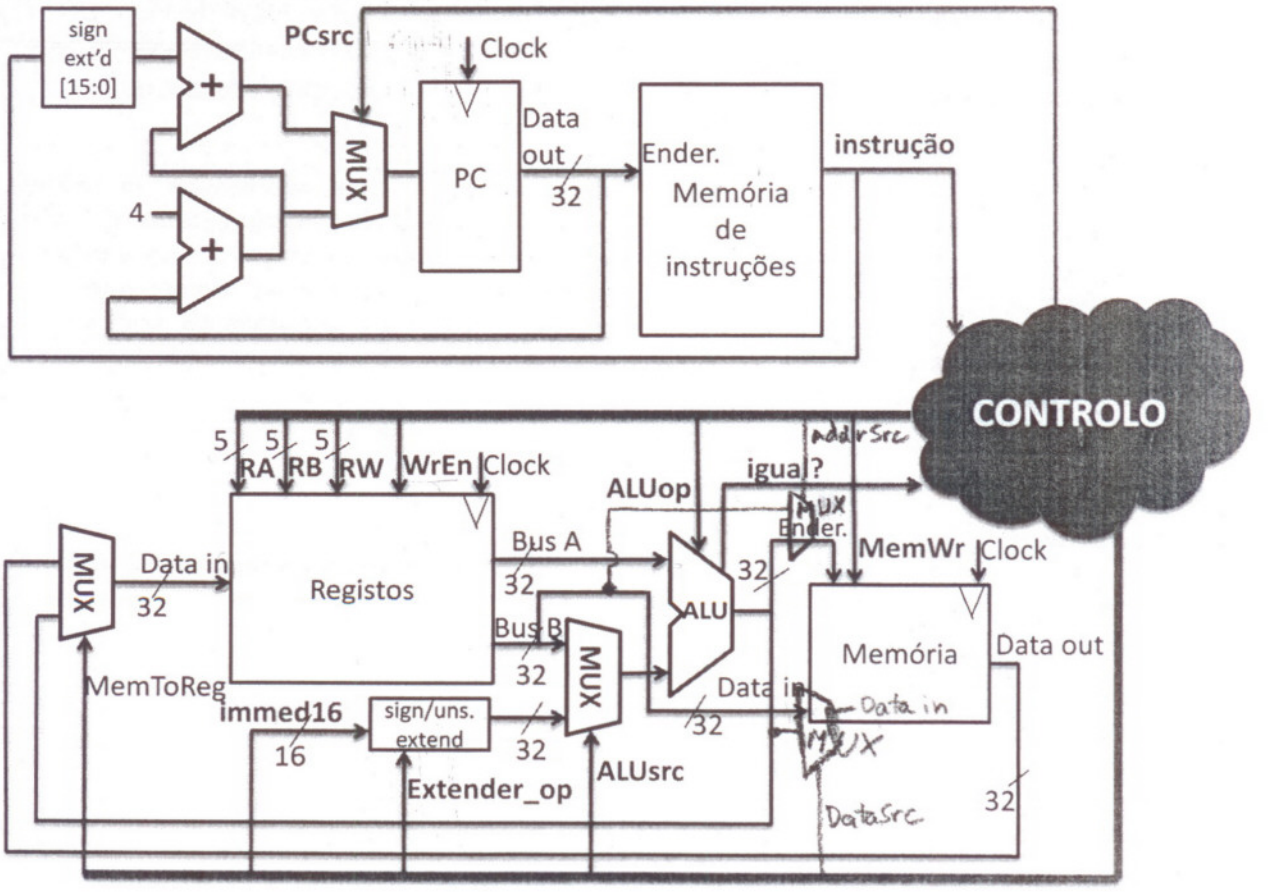
```
$pc ← $pc + 4
MEM[rt] ← R[rs] + ext_sinal(imed)
```

a) Qual o formato (R, I, ou J) da nova instrução em linguagem máquina MIPS? Justifique brevemente.

I, pois os argumentos são rs, rt, e um imediato

b) Para suportar esta instrução, quais as alterações mínimas que são necessárias efectuar ao hardware do CPU de ciclo único sem pipeline que estudámos na cadeira? Deverá em particular indicar quais os componentes novos que têm de ser adicionados ou modificados. Pode adicionar multiplexers, ALUs, somadores, ou fios. Adicionalmente, pode adicionar novas saídas e entradas aos componentes existentes. A sua resposta deve ter estar dividida em três partes:

b.1) Efectue as alterações directamente no circuito seguinte:



Número: Nome:

b.2) Liste os componentes que foram adicionados e modificados, e para cada componente indique claramente a que entradas ou saídas de outros componentes estão ligados:

MUX1 - entradas: saída da ALU e Bus B
- saída: Ender. da Mem.
MUX2 - entradas: mesmas
saída: Data In da Mem.

b.3) Liste as novas linhas de controlo que foram adicionadas:

AddrSrc
DataSrc

c) Qual os valores para todas as linhas de controlo do CPU quando esta instrução estiver a ser executada? Deve indicar os valores da respectiva linha (0 ou 1) ou X no caso de ser indiferente (don't care). Deve também adicionar à tabela abaixo as novas linhas de controlo que teve de adicionar ao CPU.

PCSrc	ALUctrl	WrEn	Extender_op	ALUsrc
0=PC+4+(sign_ext (immed)<<2); 1=PC+4	00=ADD; 01=SUB; 10=OR; 11=COMPARE	0 = no write; 1 = write	0 = zero extend, 1 = sign extend	0=RegB; 1=imediato
1	00	0	1	1

MemWr	MemToReg	AddrSrc	DataSrc		
0 = no write; 1 = write	0 = ALU; 1 = Memory	0 = Bus B 1 = ALU	0 = Bus B 1 = ALU		
1	X	0	1		

d) Suponha que após adicionar esta nova instrução e produzir o respectivo CPU, e ao medir o desempenho do seu programa ao executar no novo CPU, reparou que este se executa mais rapidamente que anteriormente. Indique uma possível explicação para este facto.

o programa faz uso frequente da nova
instrução que executa num ciclo o que
antes demorava dois ciclos de relógio.

e) No entanto, o fabricante teve de diminuir a frequência do CPU para o novo modelo. Indique uma possível explicação para este facto.

Ao adicionar mais lógica (MUXs) temos de aumentar o período do relógio para incluir o atraso adicional da propagação dos sinais lógicos pelos novos circuitos. Maior período ⇒ Menor frequência

f) A resposta da aínea anterior permaneceria válida se o CPU usasse pipelining? Explique justificadamente porquê.

Depende. Se a nova lógica estiver no patamar mais lento vai diminuir. Mas caso contrário, só diminui se ao somar o atraso da nova lógica esse patamar se tornar o mais lento.

7. [2 valores] Admita uma arquitectura de um computador com palavras e endereços de 32 bits.

   a. Qual é o tamanho máximo para o espaço de endereçamento de um programa, ou seja, quanta memória pode um programa endereçar?

2^{32} Bytes.

   b. Dado o seguinte endereço: 0000 0110 1111 0000 1111 1100 1101 1100

   Indique as componentes em que é subdividido - tag, índice (quando necessário) e deslocamento - para cada uma das seguintes arquiteturas de cache:

      i. completamente associativa de 1 MByte com linhas de 64 Bytes (isto é, a dimensão do bloco é de 64 Bytes).

índice → 0 bits (completamente assoc.)

offset → 6 bits (escolhe entre os 64 Bytes do bloco)

tag → 32 − 6 = 26 bits

      ii. associativa com 8 vias, com capacidade de 4 MBytes, e com linhas de 32 Bytes (isto é, a dimensão do bloco é de 32 Bytes).

cada via tem capacidade = 512 kBytes, = 2^{14} linhas

⇒ 14 bits para índice

offset → 5 bits (escolhe entre 32 B do bloco)

tag → 32 − 14 − 5 = 13 bits

$512k = 2^{19}$ Bytes

$2^{19}/32 = 2^{19}/2^5 = 2^{14}$ linhas

Bytes/linha