Exame **Arquitectura de Computadores I** **04-01-2005**

PARTE I (sem consulta)

1. **NOTA**: Use **no máximo 40 palavras** para responder a cada uma das 4 alíneas seguintes:

a) Descreva sucintamente as duas operações básicas que devem ser levadas a cabo pelo *datapath* sempre que ocorre uma excepção.

b) Na norma IEEE 754 existem valores reservados para o expoente que configuram casos particulares de codificação. Indique quais são e o seu significado.

c) Diga o que entende por "*forwarding*" e qual a sua utilidade (dê um exemplo ilustrativo).

d) Identifique o tipo a que pertence a instrução "**lw $2, 0($3)**", os respectivos campos que a compõem e os respectivos valores.

2. Considere o *datapath single-cycle* que foi apresentado nas aulas teóricas. Admita os seguintes atrasos de propagação envolvidos nos vários elementos operativos e de estado:

Memória externa (dados e código): Leitura – 5ns; Escrita – 8ns

File Register: Leitura – 2ns; Escrita – 4ns *Sign Extend*: 1ns

ALU (qualquer operação): 3ns *Somadores*: 1ns *Outros dispositivos*: 0ns

a) Determine o tempo mínimo necessário à execução das seguintes três instruções:

add $s0, $s1, $s2 **lw $t0, 20($a0)** **sw $t9, 0($s0)**

b) Determine, **justificando adequadamente**, a máxima frequência de relógio desta arquitectura.

3. Considere que o conteúdo dos registos **$f6** e **$f4** é respectivamente:

$f6 = 0 00010010 01010000000000000000000

$f4 = 1 01101111 11000000000000000000000

a) Obtenha a representação das quantidades armazenadas naqueles registos (codificadas segunda a norma IEEE 754) na forma $\pm$**0,m * 2^{exp}** , em que "**m**" e "**exp**" estão em decimal.

b) Determine o resultado da instrução **div.s $f0, $f6, $f4**, realizando, em binário e/ou hexadecimal, os passos necessários à sua obtenção. Indique, em binário, qual o conteúdo do registo **$f0** após a execução da instrução.

4. Considere, na **figura 2**, o trecho de código *Assembly* ali apresentado. Admita que o valor presente em **$PC** corresponde ao *label* "**lp1:**", que nesse instante o conteúdo dos registos é o indicado, e que vai iniciar-se o "*instruction fetch*" da próxima instrução. Considere ainda o *datapath* e a unidade de controlo fornecido na próxima página, no pressuposto de que corresponde a uma implementação simplificada do MIPS de execução multi-ciclo sem *pipelining*.

a) Escreva **em hexadecimal**, nas duas colunas do lado direito da fig.2, o endereço em que se encontra armazenada cada uma das instruções do código fornecido, e o respectivo conteúdo, sabendo que a primeira linha corresponde à instrução presente no *label* "**lp1:**".

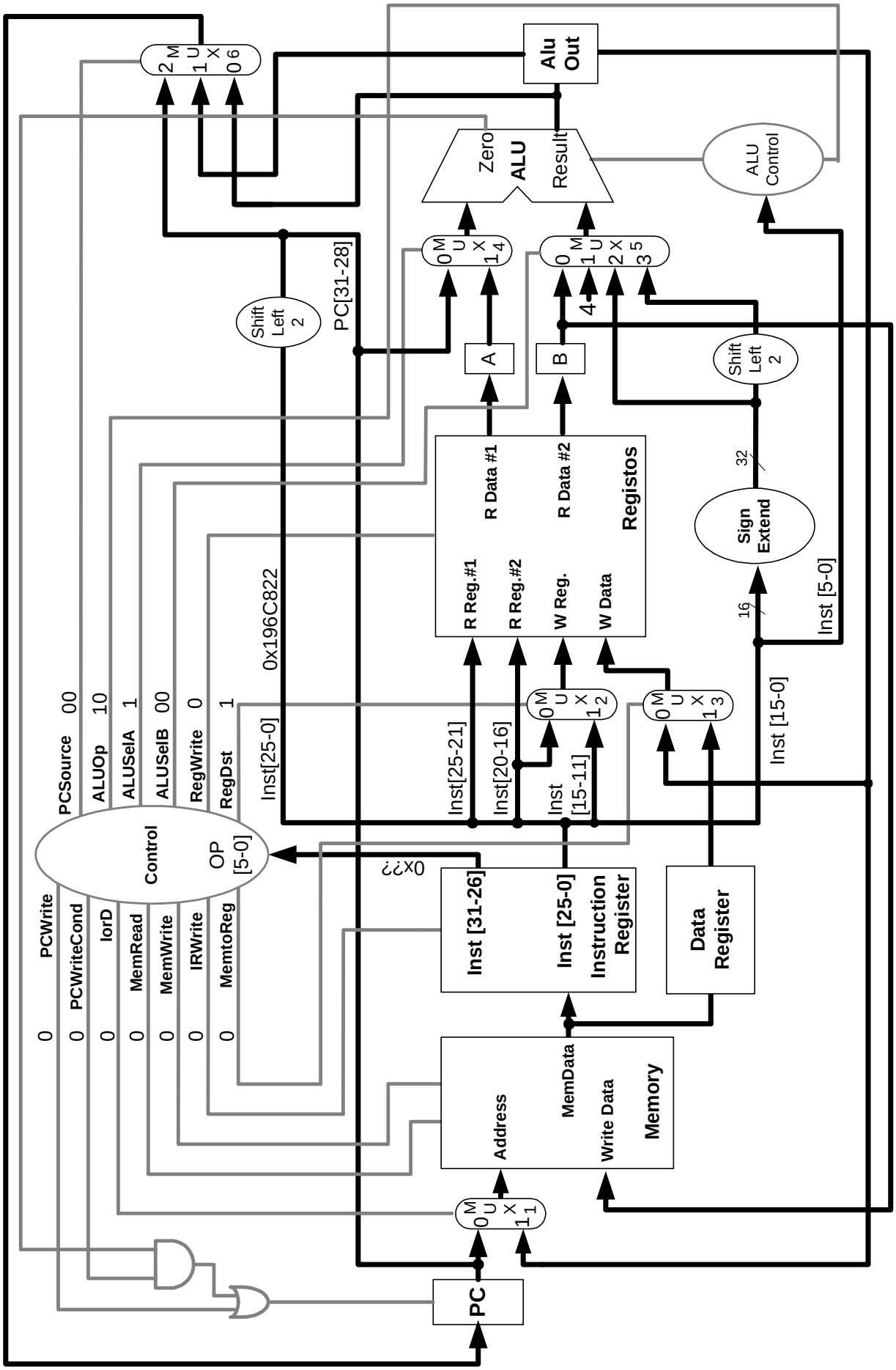
b) Considere que os valores indicados no *datapath* fornecido correspondem à "fotografia" tirada no decurso da execução de uma dada instrução. Identifique qual a instrução em causa, escreva a linha de código correspondente em *Assembly* do MIPS e determine em que fase de execução esta se encontra. **Justifique adequadamente a sua resposta**.

c) Preencha a tabela fornecida em anexo com o nome de cada uma das fases de execução da instrução "**addi $10, $10, -1**" e com o valor que tomam, em cada uma delas, os sinais e valores do *datapath* e os sinais de controlo ali indicados. Admita que o valor lógico "1" corresponde ao estado activo. (considere que $10 contém o valor indicado na tabela da figura 2).

Não se esqueça de preencher o cabeçalho

d) Sabendo que a frequência do relógio do CPU é de 500 MHz, determine o tempo total que demora a executar o código fornecido, desde o instante inicial do *instruction fetch* da instrução presente em "**lp1**:" até ao momento em que vai iniciar-se o *instruction fetch* da instrução presente em "**next:**". **Justifique adequadamente a sua resposta**.

Cotações: 1a) a 1d) – 0,5; 2a) – 1,0; 2b) – 0,5; 3a) – 1,0; 3b) – 1,5; 4a) – 2,0; 4b) – 1,0; 4c) – 2,0; 4d) – 1,0


Alu Out
Zero
ALU
Result
ALU Control
MUX 0 1 2
MUX 0 1 4
MUX 0 1 2 3 5
Shift Left 2
PC[31-28]
4
A
B
Shift Left 2
32
Sign Extend
16
R Data #1
R Data #2
Registos
R Reg.#1
R Reg.#2
W Reg.
W Data
0x196C822
Inst[25-0]
Inst[25-21]
Inst[20-16]
Inst [15-11]
Inst [15-0]
Inst [5-0]
MUX 0 1 2
MUX 0 1 3
PCSource 00
ALUOp 10
ALUSelA 1
ALUSelB 00
RegWrite 0
RegDst 1
Control
OP [5-0]
0x??
PCWrite 0
PCWriteCond 0
IorD 0
MemRead 0
MemWrite 0
IRWrite 0
MemtoReg 0
Inst [31-26]
Inst [25-0]
Instruction Register
Data Register
Address
MemData
Write Data
Memory
MUX 0 1 1
PC

Exame **Arquitectura de Computadores I** **04-01-2005**

Nome:_____

Curso:_____ **Nº Mecanográfico:**_____

```
lp1:   beq  $10, $0, next
       lw   $8, -0x3($9)
       sub  $6, $6, $8
       addi $10, $10, -1
       add  $9, $9, $7
       j    lp1
next:  ...
```

OpCode	Funct	Operação
0	0x20	add
0	0x22	sub
0x02		j
0x04		beq
0x08		addi
0x0D		ori
0x0F		lui
0x23		lw
0x2b		sw

	. . .
reg $6	0x000000FF
reg $7	0xFFFFFFFC
reg $8	0x80020C05
reg $9	0x1001000B
reg $10	0x00000003
	. . .
$PC	0x0040001C

CPU

Endereço	Código
...	...
...	...

Figura 2 (Problema 4)

	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5
Nome da fase					
Datapath					
A					
B					
Data Register					
ALU Out					
ALU Result					
ALU Zero					
Controlo					
ALUSelA					
ALUSelB					
ALUOp					
IorD					
IRWrite					
MemRead					
MemtoReg					
MemWrite					
PCSource					
RegDst					
RegWrite					
PCWrite					
PCWriteCond					

Exame **Arquitectura de Computadores I** **04-01-2005**

PARTE II

NOTE BEM: Leia atentamente todas as questões, comente o código usando a linguagem C e respeite a convenção de passagem de parâmetros e salvaguarda de registos que estudou ☺. Respeite rigorosamente os aspectos estruturais e a sequência de instruções indicadas no código original fornecido, bem como as indicações, quando existirem, sobre quais os registos a usar para cada variável.

Considere o seguinte trecho de código em linguagem C:

```
int main (int argc, char* argv[])
{
      static int lista[10];
      int totalmin;    /* deve residir no registo $s2  */
      float media;     /* deve residir no registo $f20 */
      float varia;

      if ((argc > 0) && (argc < 11))
      {
            totalmin = analise (argc, argv, lista);

            (…)

            varia = variancia (argc, lista, media);

            print_str ("Variância: "); /* system call */
            print_float (varia);       /* system call */
            return 0;
      }
      else
      {
            print_str ("Erro");
            return 1;
      }
}
```

1. Codifique em *Assembly* do MIPS, a função **`main`**, sabendo que os protótipos das funções **`analise`** e **`variancia`** são os seguintes:

```
int analise (int num, char* palavras[], int* lista);

float variancia (int num, int* lista, float media);
```

Defina, no segmento de dados, as *strings* e as variáveis declaradas como `static`.

(v.s.f.f.)

Cotações: **1**-1.5; **2**-2.5; **3**-1.5; **4**-2.5 (Total: 8 valores) DETUA/2005

2. Considerando que a função **`contamin`** tem o seguinte protótipo:

```
int contamin (char* str);
```

traduza para *Assembly* do MIPS, a função **`analise`**.

```
int analise (int num, char* palavras[], int* lista)
{
      int i;
      int min;
      int total = 0;

      for (i = 0; i < num; i++)
      {
            min = contamin (palavras[i]);
            lista[i] = min;
            total = total + min;
      }

      return total;
}
```

3. Traduza para *Assembly* do MIPS, a função **`contamin`**.

```
int contamin (char* str)
{
      int cont = 0;

      while (*str != '\0')
      {
            if ((*str >= 'a') && (*str <= 'z'))
            {
                  cont++;
            }
            str++;
      }

      return cont;
}
```

4. Traduza para *Assembly* do MIPS, a função **`variancia`**.

```
float variancia (int num, int* lista, float media)
{
      int i;
      float dif;
      float varia = 0.0;

      for (i = 0; i < num; i++)
      {
            dif = (float)lista[i] - media;
            varia = varia + (dif * dif);
      }

      return varia / (float)num;
}
```