Recorrência **Introdução à Arquitectura de Computadores** **11-07-2002**

PARTE I (sem consulta)

Nota: – não é permitido o uso de calculadoras -

1. a) Descreva o formato da instrução do MIPS **bne $s0, $t0, next**, indicando os campos que a compõem e o seu significado.

b) Indique o número de modos de endereçamento nativos suportados pelo MIPS e descreva sucintamente cada um deles.

2. a) A mantissa dos valores intermédios resultantes dos cálculos em vírgula flutuante são armazenados com três bits suplementares, tendo cada um deles um finalidade específica. Indique o nome de cada um e descreva sucintamente a sua finalidade.

b) Admita que o conteúdo dos registos **$f10** e **$f12** do co-processador aritmético do MIPS é respectivamente **2,5 x 2^{1}** e **1.34375 x 2^{-2}**. Determine qual o resultado da instrução

add.s $f2, $f10, $f12

indicando os vários passos envolvidos na obtenção desse resultado e fornecendo o valor binário presente em cada um dos registos envolvidos.

3. Observe com atenção a **figura 2** fornecida em anexo. Atente ao conteúdo das posições de memória compreendidas entre os endereços **0x400068** e **0x400074** e, bem assim, o conteúdo dos registos do CPU no exacto momento em que se concluiu a execução da instrução armazenada no endereço **0x00400064**. Considere ainda o *datapath* e a unidade de controle correspondentes a uma versão simplificada do MIPS cujo diagrama é fornecido na figura 1, no pressuposto de que corresponde a uma implementação de execução multi-ciclo sem *pipelining*:

a) Escreva, em Assembly do MIPS, o trecho de código armazenado entre aqueles endereços (quatro instruções). Na(s) instruções de *branch* e (ou) *jump*, determine, e expresse em hexadecimal, o endereço destino.

b) Determine, justificando, o número total de ciclos de relógio necessários à execução daquelas quatro instruções.

c) Considere agora a execução da instrução armazenada em **0x00400068**. Preencha a tabela fornecida em anexo com a seguinte informação: nome de cada uma das fases de sua execução; valor que tomam, em cada uma das fases, os sinais do *datapath* ali indicados; valor que tomam, também para cada uma das fases, os vários sinais de controle. Admita para isso que o valor lógico "1" corresponde ao estado activo dos sinais, correspondendo o valor lógico "0" ao seu estado não activo. **NOTA**: Não se esqueça de preencher o cabeçalho com o seu nome, curso e N.M.

d) Determine o valor armazenado em cada um dos registos do CPU indicados na figura 2, após a conclusão da execução da instrução que se encontra no endereço **0x00400070**.

e) Indique, justificando, qual das quatro instruções não é suportada pelo *datapath* fornecido. Dê mais três exemplos de instruções da máquina real, pertencentes a famílias distintas, que também não sejam suportadas por aquele *datapath*.

f) Descreva, de forma sucinta mas objectiva, quais as alterações que deveria introduzir no *datapath* para poder suportar a instrução **jal.**

Cotações: **1a)–** 1,5; **1b)**– 1.0**; 2a)**– 1.0**; 2b)**– 1,5**; 3a)**– 1,5**; 3b)**– 0,5**; 3c)**– 2**; 3d)**– 1**; 3e)**– 1; **3f)**-1

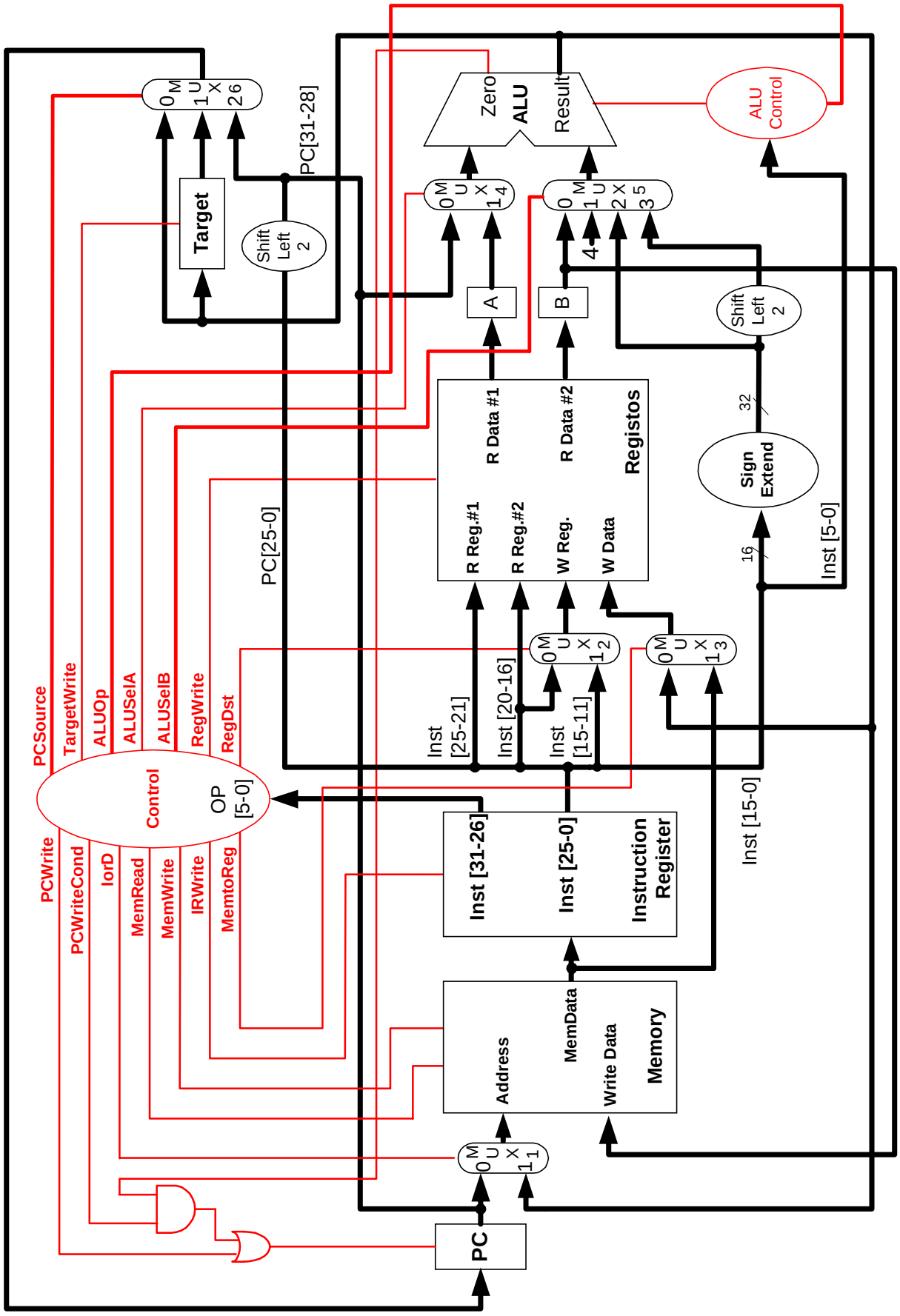


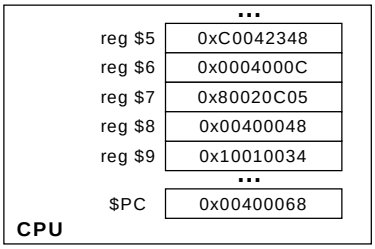
Figura 1 (Problema 3)

Recorrência **Introdução à Arquitectura de Computadores** **11-07-2002**

Nome:_____

Curso:_____ **Nº Mecanográfico:**_____

OpCode	Funct	Operação
0	0x20	add
0	0x22	sub
0x02		j
0x03		jal
0x05		bne
0x08		addi
0x0F		lui
0x23		lw
0x2b		sw
0x31		ori



Memória

0x00400064	...
0x00400068	100011 01000 00101 0000000000100100
0x0040006C	001000 01000 01000 1111111111111100
0x00400070	110001 00110 01001 0000000000000011
0x00400074	000010 00 0001 0000 0000 0000 0001 1001
0x00400078	...

Figura 2 (Problema 3)

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

Recorrência **Introdução à Arquitectura de Computadores** **11.07.02**

PARTE II

Cotações: 1 – 2; **2a** – 1,0; **2b** – 2,5; **2c** – 2.5 (8 valores)

NOTE BEM: Na resolução da PARTE II do exame pode consultar o anexo às folhas de acompanhamento da parte prática da disciplina, composto por cinco páginas e contendo o *set* de instruções do MIPS. Responda **apenas** ao que é solicitado em cada questão **respeitando estritamente** o que for aí determinado. **Não repita** código que já escreveu em alíneas anteriores. Para todos os efeitos, respeite as convenções adoptadas quanto à utilização e salvaguarda de registos. Respeite igualmente **rigorosamente** os aspectos estruturais e a sequência de instruções indicadas no programa original fornecido na página seguinte, bem como as indicações sobre quais registos usar para cada variável.

1- O trecho de código Assembly que se segue utiliza um conjunto de instruções do MIPS que apenas são reconhecidas pela "máquina virtual". Re-escreva aquele código usando apenas instruções da máquina real. Refira os registos do MIPS pelo seu nome real, usando o registo $1 como registo auxiliar no caso de ser necessário. Se for necessário pode igualmente criar novos *labels*.

NOTA: a instrução "**break $0**" gera uma excepção de "*overflow*".

```
Code_1:         ...
                li    $s0, 50
                mulo  $s1, $s0, -50
                abs   $s1, $s1
                ...
```

2- O programa que se segue, escrito em linguagem C, tem como finalidade calcular a média final de curso, dados um conjunto de disciplinas e os respectivos créditos**.**

a) Complete o programa principal **`main()`**, escrevendo os trechos de código identificados pelos **`"???"`**. Note que cada **`"???"`** está numerado e pode corresponder a uma ou mais linhas de código em falta. Observe a necessidade de garantir que a dimensão dos *arrays* não é excedida. Use onde necessário funções disponibilizadas pelo kernel do PCSPIM!

Indique sucintamente, justificando, quais as alterações que deveria introduzir no programa se quisesse transformar as estrutura de controle de fluxo do tipo **while()** numa estrutura do tipo **do/while().**

As rotinas leituraNota e printMedia têm os seguintes protótipos:

```
void leituraNota(int nn, int *p_nota);
void printMedia (int n_disc, int media);
```

b) Codifique em Assembly do MIPS o código correspondente ao programa principal. As variáveis **`notas[]`** e **`creditos[]`** devem ser criadas em memória, no segmento de dados, enquanto as variáveis **`j`**, **`mf`**, **`pn`** e **`pc`** devem fazer uso respectivamente dos registos $s0, $s1, $s2 e $f2. A variável **nd** deve ser implementada no registo $t1. As strings utilizadas no interface com o utilizador devem residir em memória, no segmento de dados.

c) Codifique em Assembly do MIPS o código correspondente à subrotina **`média_final()`**. As variáveis **`i`** e **`media`** devem fazer uso respectivamente dos registos $s5 e $s6, enquanto **`total`** e **`t_cred`** devem usar os registos $f0 e $f2.

```
int   notas[50];
float creditos[50];

void main()
{
      int j, mf, nd;
      int *pn;
      float pc;

      print_string ("Numero de disciplinas: ");
      nd = read_int ();

      ???-1

      while ( ???-2 )
      {
            printf ("Creditos da disciplina nº %d: ", j);
            pc = read_float ();
            leituraNota(j, pn);

            ???-3
      }
      mf = media_final (notas, creditos, nd);
      printMedia (nd, mf);
}

int media_final (int *notas, float *creditos, int n_disc)
{
      int i, media;
      float total, t_cred;

      total = 0.;
      t_cred = 0.;
      for (i = 0; i < n_disc; i++)
      {
            total = total + (*creditos * (float) notas[i]);
            t_cred = t_cred + *creditos;
            creditos++;
      }
      media = (int) (0.5 + (total / t_cred));
      return media;
}
```