Exame **Introdução à Arquitectura de Computadores** **21-06-2002**

PARTE I (sem consulta)

1. "*A simplicidade favorece a regularidade*". Comente esta afirmação, relacionando-a com as características fundamentais de uma arquitectura do tipo RISC.

2. a) Indique uma instrução, do set de instruções do MIPS, que suporte a evocação de subrotinas. Descreva sucintamente o seu funcionamento e os mecanismos que lhe estão associados.

b) Admita agora que o MIPS não dispunha de instruções desse tipo. Sugira um método alternativo, recorrendo apenas às restantes instruções, que lhe permitisse suprir essa falta. Ilustre com um exemplo em *Assembly*.

3. a) Na norma IEEE 754 o expoente é codificado em excesso 2^{n-1}-1 (com ***n*** igual ao número de bits do mesmo). Descreva as razões para a adopção dessa técnica de codificação em detrimento da codificação em complemento para dois.

b) Admita que o registo **$f10** contém a quantidade $\mathbf{-9{,}074833}_{10}$. Indique qual o conteúdo, em binário, desse registo, sabendo que aquela quantidade está codificado em vírgula flutuante, precisão simples, de acordo com a norma IEEE 754.

4. Observe com atenção a **figura 2** fornecida em anexo. Considere o conteúdo das posições de memória compreendidas entre os endereços **0x400034** e **0x400040** e, bem assim, o conteúdo dos registos do CPU no momento em que se concluiu a execução da instrução armazenada no endereço **0x00400030**. Considere ainda o *datapath* e a unidade de controle correspondentes a uma versão simplificada do MIPS cujo diagrama é fornecido na próxima página, 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 (4 instruções).

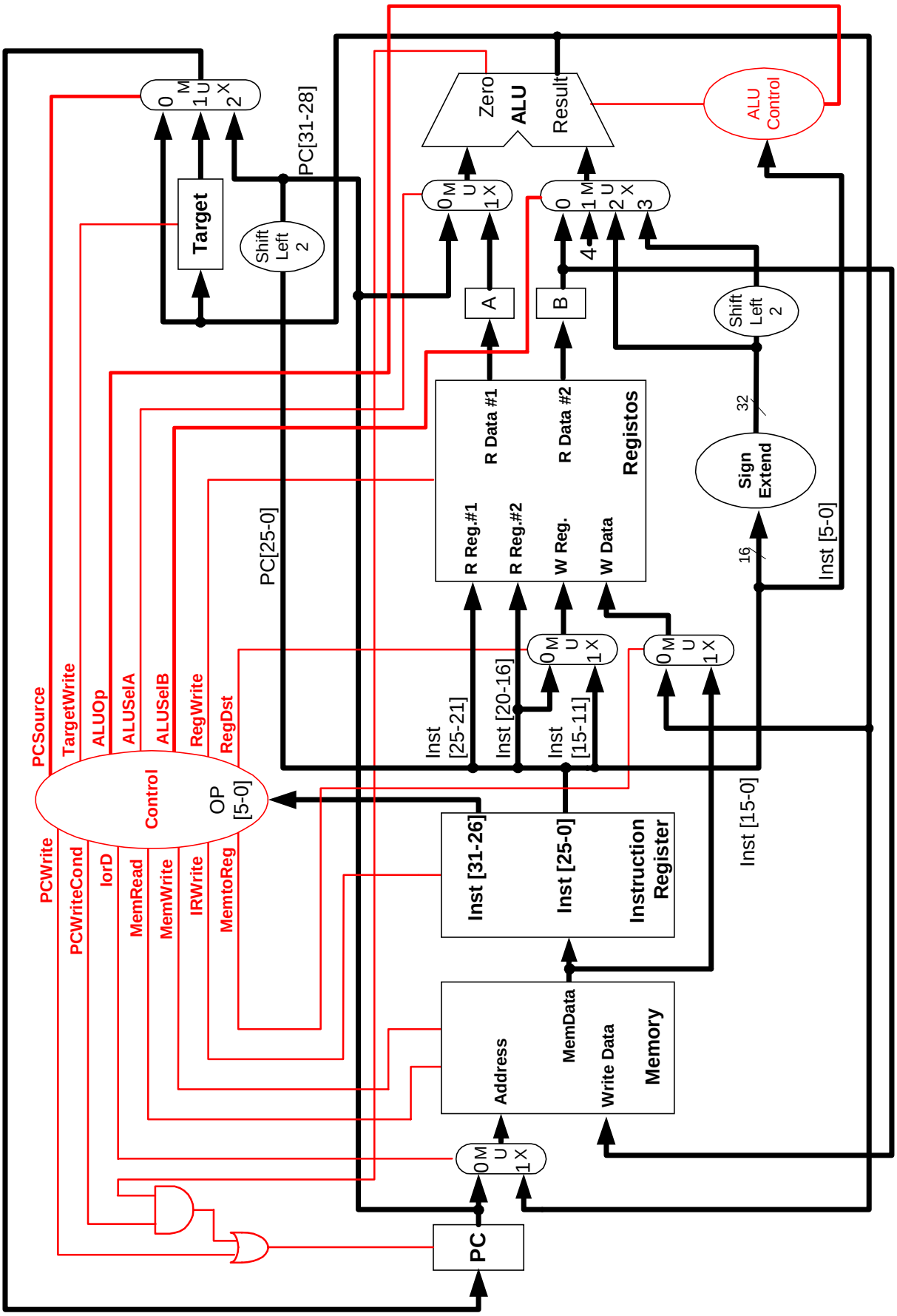
b) Sabendo que a frequência do sinal de relógio é de100MHz determine o tempo total necessário para concluir a execução desse trecho de código. Justifique a sua resposta.

c) Admita agora que o valor armazenado em **$PC** é **0x00400038**. Indique o número de ciclos de relógio (fases) necessárias para completar a execução da instrução respectiva.

d) Considerando a execução da instrução referida na alínea anterior, preencha a tabela fornecida em anexo com a seguinte informação: nome de cada uma das fases de execução da instrução; 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.

e) Determine, finalmente, o valor armazenado em cada um dos registos do CPU indicados na figura 2, após a execução da instrução que se encontra no endereço **0x00400040**. Comente o valor que resulta da execução dessa instrução, sugerindo um método para verificar a sua validade.

Cotações: **1–** 1,0**; 2a)**– 1.5**; 2b)**– 1,0**; 3a)**– 1,0**; 3b)**– 2,0**; 4a)**– 1,0**; 4b)**– 1,0**; 4c)**– 0,5**; 4d)**– 2,0**; 4e)**– 1,0

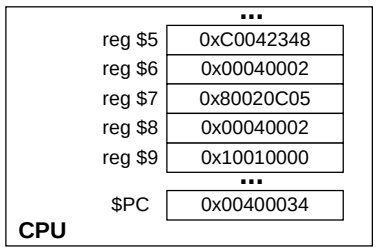

PCSource
TargetWrite
ALUOp
ALUSelA
ALUSelB
RegWrite
RegDst
PCWrite
PCWriteCond
IorD
MemRead
MemWrite
IRWrite
MemtoReg
Control
OP [5-0]
PC
M U X
0
1
2
Target
Shift Left 2
PC[31-28]
PC[25-0]
Zero
ALU
Result
ALU Control
3
4
A
B
R Data #1
R Data #2
Registos
R Reg.#1
R Reg.#2
W Reg.
W Data
Inst [25-21]
Inst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

Exame **Introdução à Arquitectura de Computadores** **21-06-2002**

Nome:_____

Curso:_____ **Nº Mecanográfico:**_____

OpCode	Funct	Operação
0	0x20	add
0	0x22	sub
0x02		j
0x03		jal
0x04		beq
0x05		bne
0x08		addi
0x0F		lui
0x23		lw
0x2b		sw



Memória

0x00400030	...
0x00400034	000100 00110 01000 0000000000000001
0x00400038	101011 01001 00101 0000000001001100
0x0040003C	001000 00101 00101 0000000000000001
0x00400040	000000 00110 00111 01001 00000 100010
0x00400044	...

Figura 2 (Problema 4)

	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					

Exame **Introdução à Arquitectura de Computadores** **21.06.02**

PARTE II

Cotações: 1 – 0,5; **2** – 2,0; **3 –** 2,0; **4** – 2,0; **5 –** 1.5 (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.

Introdução: O Dr. M. é daqueles médicos que cobra um valor variável pelas consultas (leva mais aos mais ricos) o que lhe levanta alguns problemas de contabilidade. Uma vez que lhe ofereceram uma workstation baseada num MIPS, precisa agora de um programa que lhe permita, entre outros pormenores, saber quanto ganha em média por consulta...

1- Escreva, em *Assembly* do MIPS, o trecho de código identificado na página seguinte pela letra ***A*.** Respeite a ordem de declaração das variáveis. Estas variáveis **deverão obrigatoriamente** residir em memória, no segmento de dados.

2- Escreva, em *Assembly* do MIPS, o trecho de código identificado na página seguinte pela letra ***B*.** Use, onde necessário, as funções disponibilizadas pelo *Kernel* do PCSPIM. Respeita as indicações dadas para **num**, **Opcao** e **tipoConsulta**. Se nenhuma indicação é dada, implemente essa(s) variável(eis) da maneira que achar mais conveniente. Admita os seguintes protótipos para as rotinas **NovaConsulta** e **VerMediaActual**:

```
int NovaConsulta();
void VerMediaActual(double media);
```

3- O trecho identificado pela letra ***C*** apenas contem instruções da máquina real. Reescreva-o da **maneira mais simples possível**, tirando partido das instruções virtuais. **Codifique ainda** em C o mesmo código. Esta rotina cria uma tabela de 4 elementos correspondendo aos valores dos 4 níveis de preços praticados.

4- Escreva, em *Assembly* do MIPS, o trecho de código correspondente à subrotina **ActualizaMedia ()**, identificada na página seguinte pela letra ***D*.**

5- Suponha que a variável **int *consultas** contem uma lista (de inteiros) com o nível de preço (1 .. 4) das últimas 100 consultas. Para calcular a percentagem de pacientes ricos a quem o Dr. M deu consulta, implemente o código designado por ***E*** em *Assembly*. A função seria chamada com o seguinte código:

```
PercentagemRicos(0, consultas);
```

A

```
char Mensag1[]= {"0– Sair ; 1– Novo valor ; 2– Ver media ; 3– Reset media"};
char Mensag2[]= {"Opcao inválida"};
int precario[4];
double media= 0.0;
```

B

```
void main(void)
{
      unsigned int num;              // num - deve residir em $s1
      char opcao;                    // opcao - deve residir em $t1
      int tipoConsulta, preco;       // tipoConsulta - deve residir em $t2

      print_string(Mensag1);
      CreatePrecario(precario);

      do{     opcao= read_char();
              switch(opcao){
              '1':   tipoConsulta= NovaConsulta();
                     num++;
                     preco= precario[tipoConsulta];
                     ActualizaMedia(num, preco, &media);
                     break;

              '2':   VerMediaActual(media);
                     break;

              '3':   media= 0;
                     num= 0;
                     break;

         default:  print_string(Mensag2);
      }while(opcao != '0');
}
```

C

```
CreatePrecario:      addi  $v0, $0, 0
                     addi  $t0, $0, 0
                     addi  $t1, $t0, 10
                     addi  $t2, $0, 2
                     addi  $t3, $0, 3
          loop:      sll   $t4, $t0, 2
                     add   $t4, $t4, $a0
                     sw    $t1, 0($t4)
                     mult  $t1, $t2
                     mflo  $t1
                     addi  $t0, $t0, 1
                     slt   $t5, $t3, $t0
                     beq   $t5, $0, loop
                     addi  $v0, $0, 1
                     jr $ra
```

D

```
void ActualizaMedia(unsigned num, int preco, double *p_media)
{
     *p_media= (*p_media * (double)(num – 1) + (double)preco) / (double) num;
}
```

E

```
int PercentagemRicos(int i, int *consultas)
{
      int perc= 0;                   // perc - deve residir em $v0

      if(i == 100)
      {
              return(perc);
      }
      else
      {
              perc = PercentagemRicos(i++, consultas);
              if( consultas[i] == 3 )
                     perc++;
              return (perc);
      }
}
```