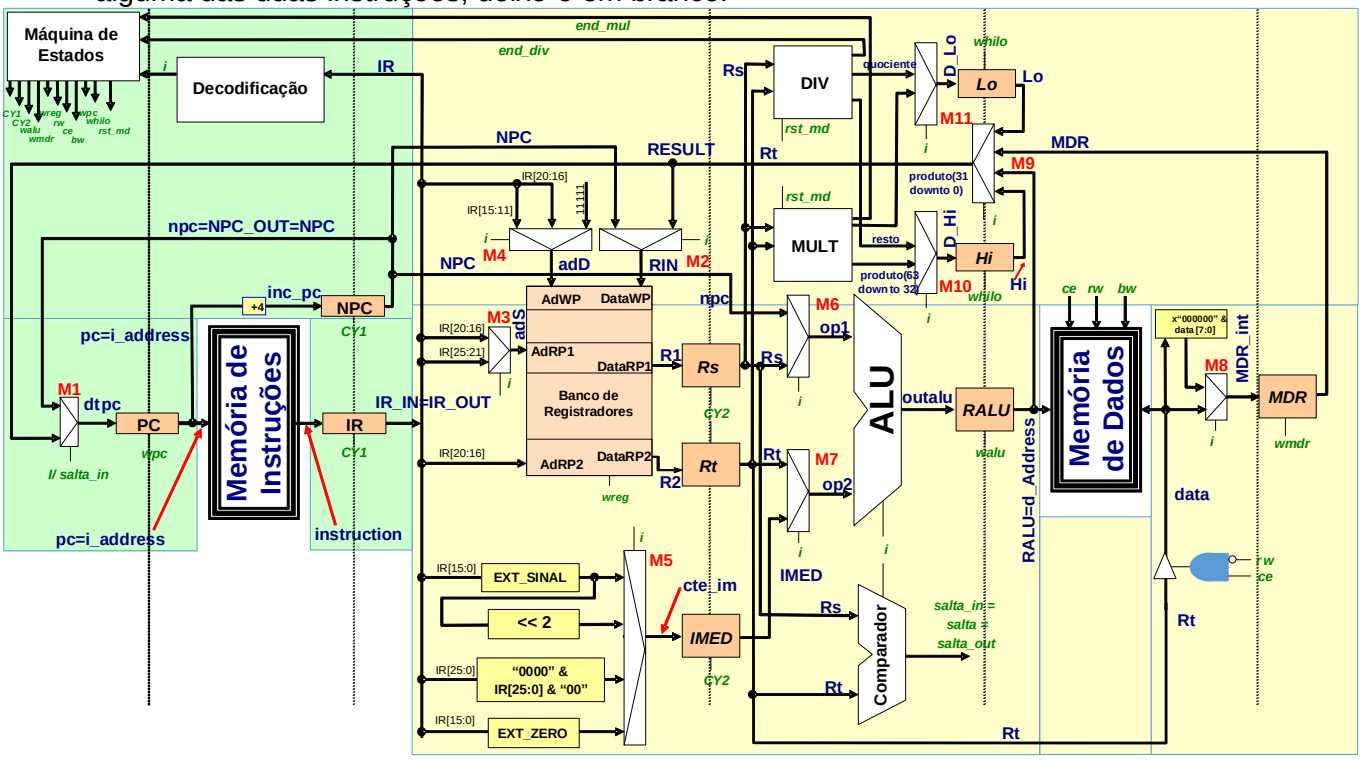
Prova P2 **Disciplina: Organização de Computadores**

Professor: Ney Laert Vilar Calazans

Aluno: 21/junho/2018

1. [3,5 pontos]. Considere o processador multiciclo abaixo (é o mesmo visto em aula):
   a) [1,5 pontos]. Marcar sobre o desenho todos os caminhos que contêm informações úteis à execução da instrução “BLEZ Rt,Rs,rótulo” em algum ciclo de relógio durante a execução (nos multiplexadores, indicar de que entrada a saída do multiplexador vem, ou marque ambas entradas, se a decisão depende dos dados da instrução).
   b) [2 pontos]. Use a tabela abaixo do desenho para explicar sucintamente o que ocorre em cada ciclo de relógio para as duas instruções especificadas na segunda e terceira colunas da tabela. A cada ciclo, dizer que registrador(es) é(são) escrito(s), e com qual conteúdo (semântica do valor), e a operação que a ULA executa, no ciclo relevante. Se algum ciclo não for usado em alguma das duas instruções, deixe-o em branco.



Ciclo de Relógio	Instrução: LBU Rt, Offset(Rs)	Instrução: SLTIU Rs,Rt,imed
1º		
2º		
3º		
4º		
5º		

2. [3,5 pontos]. Dada uma frequência de operação de 1GHz para o processador **MIPS multiciclo**, assuma que a organização original foi alterada para dar suporte a todas as instruções do programa abaixo, mantendo a característica multiciclo. Com estes pressupostos, calcule e diga:
   a) (1,5 pontos). O número de ciclos de relógio que leva p/ executar o programa, com a área de dados fornecida (A pseudo-instrução **la** equivale a **lui+ori** e li equivale a um **addiu**. Assuma que a instrução **syscall** executa em 4 ciclos de relógio);
   b) (0,5 pontos). O tempo de execução do programa em segundos (1ns=10^{-9} segundos);
   c) (1,0 ponto). O que faz este programa, do ponto de vista semântico. Diga o que saída ele gera;
   d) (0,5 pontos). Se este programa possui subrotina(s). Se sim, diga onde esta(s) se encontra(m), definindo o intervalo de linhas que ela(s) ocupa(m).

```
1.        .text
2.        .globl main
3.  main: la     $t0,t1
4.        la     $t1,t2
5.        li     $t2,0
6.        li     $t3,0
7.  l:    lbu    $t4,0($t0)
8.        beq    $t4,$zero,fl
9.        addiu  $t2,$t2,1
10.       sb     $t4,0($t1)
11.       addiu  $t0,$t0,1
12.       addiu  $t1,$t1,1
13.       j      l
14. fl:   addiu  $t1,$t1,1
15.       sb     $zero,0($t1)
16.       la     $t5,s
17.       sw     $t2,0($t5)
18.       li     $v0,10
19.       syscall
20.       .data
21. s:    .word  0
22. t1:   .asciiz      "Teste de Varios Valores: 1, 2, 3"
23. t2:   .space 40
```

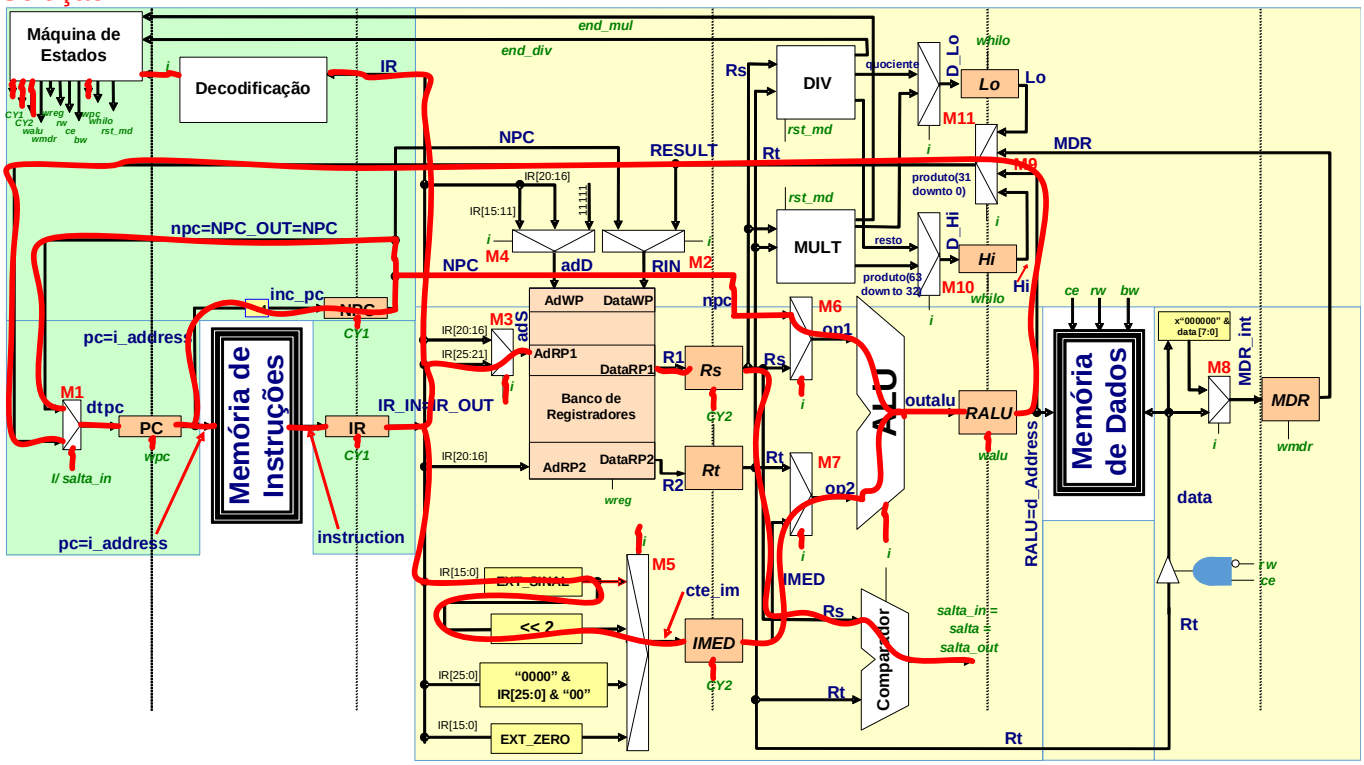
3. [3,0 pontos]. Assuma uma frequência de relógio de 500 MHz para a organização MIPS multiciclo estudada em aula e desenhada acima. Ela foi modificada para reconhecer e executar a instrução LB do MIPS em 5 ciclos de relógio. Faça o que se pede.
   a) O programa abaixo é um teste de execução, idealizado para comparar o efeito das instruções LB e LBU. As posições de memória **position1** e **position2** correspondem respectivamente às duas primeiras posições da área de dados mostrada. Calcule os valores escritos em $t0 e $t1 após a execução do programa até a linha 11, explicando o que ocorre.
   b) Aponte pelo menos **6** módulos do processador MIPS Multiciclo que **NÃO** são utilizados durante a execução das linhas 2 a 11 do programa abaixo. Use os blocos do diagrama visto em aula, reproduzido na questão 1 desta prova. Aqui, define-se como um bloco qualquer figura geométrica presente no diagrama. Por exemplo, qualquer um dos registradores identificados (Rs, PC, IMED, etc) é um bloco, qualquer um dos multiplexadores (M1 a M11) é um bloco, a ULA, o banco de registradores, são blocos, os retângulos correspondendo a hardware de extensão de valores são blocos, etc. Por **"não utilizado"** entende-se aqui um bloco que durante **toda a execução do programa** jamais vai conter ou processar dado útil à execução de pelo menos uma das instruções deste programa.

```
1. Address    Code        Basic                      Source
2. 0x00400000  0x3c01b4b4  lui $1,0xffffb4b4     2    li     $t0,0xb4b4b4b4
3. 0x00400004  0x3428b4b4  ori $8,$1,0x0000b4b4
4. 0x00400008  0x3c015d5d  lui $1,0x00005d5d     3    li     $t1,0x5d5d5d5d
5. 0x0040000c  0x34295d5d  ori $9,$1,0x00005d5d
6. 0x00400010  0x3c011001  lui $1,0x00001001     4    la     $t2,position1
7. 0x00400014  0x342a0000  ori $10,$1,0x00000000
8. 0x00400018  0x3c011001  lui $1,0x00001001     5    la     $t3,position2
9. 0x0040001c  0x342b0004  ori $11,$1,0x00000004
10.0x00400020  0x81480000  lb $8,0x00000000($10) 6    lb     $t0,0($t2)
11.0x00400024  0x91690000  lbu $9,0x00000000($11)7    lbu    $t1,0($t3)
12.Address     Data
13.0x10010000  0xbbbbbbbb  0xeeeeeeee
```

Gabarito

1. [3,5 pontos]. Considere o processador multiciclo abaixo (é o mesmo visto em aula):
   a) [1,5 pontos]. Marcar sobre o desenho todos os caminhos que contêm informações úteis à execução da instrução “BLEZ Rt,Rs,rótulo” em algum ciclo de relógio durante a execução (nos multiplexadores, indicar de que entrada a saída do multiplexador vem, ou marque ambas entradas, se a decisão depende dos dados da instrução).
   b) [2 pontos]. Use a tabela abaixo do desenho para explicar sucintamente o que ocorre em cada ciclo de relógio para as duas instruções especificadas na segunda e terceira colunas da tabela. A cada ciclo, dizer que registrador(es) é(são) escrito(s), e com qual conteúdo (semântica do valor), e a operação que a ULA executa, no ciclo relevante. Se algum ciclo não for usado em alguma das duas instruções, deixe-o em branco.

Solução:



a) O desenho acima foi anotado com os caminhos usados pela instrução. Note que foram salientados todos os sinais de controle efetivamente relevantes para a instrução.

Ciclo de Relógio	Instrução: LBU Rt, Offset(Rs)	Instrução: SLTIU Rt,Rs,imed
1º	O conteúdo do registrador PC é usado para buscar o código objeto da instrução, (**LBU Rt,Offset(Rs)** ou **SLTIU Rs,Rt,imed**) da Memória de Instruções. Este código é carregado no registrador IR. O valor do PC incrementado de 4 é carregado no registrador NPC. O sinal de controle ativado é apenas CY1.	
2º	A instrução é decodificada no BC, que gera sinais de controle p/ sua execução a cada ciclo. Neste segundo ciclo, CY2 faz com que Rs seja carregado com o valor do registrador base, IMED seja carregado com o imediato imed com sinal estendido (EXT_SINAL). O valor carregado em Rt é irrelevante.	A instrução é decodificada no BC, que gera sinais de controle p/ sua execução a cada ciclo. Neste segundo ciclo, CY2 faz com que Rs receba o valor a ser comparado com o dado imediato (imed), IMED recebe o imediato imed com sinal estendido (EXT_SINAL), e o valor carregado em Rt é irrelevante.

3º	A ALU produz o endereço de leitura, somando o valor do registrador base (em Rs) e o offset (em IMED), o que é então carregado em RALU. Sinal ativo: walu.	A ALU produz internamente a constante 0 ou a constante 1 em 32 bits, dependendo do resultado da comparação entre as entradas Rs e IMED como dois números sem sinal. RALU recebe 1 se Rs < IMED ou recebe 0 caso contrário. Sinal ativo: walu.
4º	O conteúdo de RALU endereça a memória de dados, e os sinais ce/rw/bw valem 1/1/x, respectivamente. Eles comandam a leitura de 4 bytes da memória a partir do endereço em RALU. Estes 4 bytes vêm da memória e o byte menos significativo deles é acrescido de 24 0s à esquerda, descartando os 3 bytes mais significativos. Os 32 bits resultantes são carregados no registrador MDR. Sinal ativo: wmdr.	O conteúdo de RALU passa para RESULT e daí para a entrada do banco de registradores na porta de escrita deste. O valor de RALU é então escrito no banco de registradores, no registrador especificado pelos bits 20-16 do código objeto da instrução. Além disto o conteúdo do NPC é copiado para o PC. Sinais ativos: wreg e wpc.
5º	O valor em MDR é escrito no banco de registradores, no registrador especificado pelos bits 20-16 do código objeto da instrução. Além disto o conteúdo do NPC é copiado para o PC. Sinais ativos: wreg e wpc.	-

b) Ver conteúdo da tabela acima.

2. [3,5 pontos]. Dada uma frequência de operação de 1GHz para o processador **MIPS multiciclo**, assuma que a organização original foi alterada para dar suporte a todas as instruções do programa abaixo, mantendo a característica multiciclo. Com estes pressupostos, calcule e diga:
   a) (1,5 pontos). O número de ciclos de relógio que leva p/ executar o programa, com a área de dados fornecida (A pseudo-instrução **la** equivale a **lui+ori** e li equivale a um **addiu**. Assuma que a instrução **syscall** executa em 4 ciclos de relógio);
   b) (0,5 pontos). O tempo de execução do programa em segundos (1ns=10^{-9} segundos);
   c) (1,0 ponto). O que faz este programa, do ponto de vista semântico. Diga o que saída ele gera;
   d) (0,5 pontos). Se este programa possui subrotina(s). Se sim, diga onde esta(s) se encontra(m), definindo o intervalo de linhas que ela(s) ocupa(m).

```
1.          .text
2.          .globl main          # Ciclos     Ação
3.  main:   la      $t0,t1       #   8        Carrega em $t0 o endereço da cadeia
4.          la      $t1,t2       #   8        Carrega em $t1 o endereço da nova cadeia
5.          li      $t2,0        #   4        Carrega em $t2 o valor 0 (contador de elementos)
6.          li      $t3,0        #   4        Carrega em $t3 o valor 0
7.  l:      lbu     $t4,0($t0)   #   5        Carrega em $t4 o valor de elemento da cadeia t1
8.          beq     $t4,$zero,fl #   4        Se fim da cadeia, sai do laço
9.          addiu   $t2,$t2,1    #   4        incrementa o contador de elementos da cadeia
10.         sb      $t4,0($t1)   #   4        copia elemento da cadeia t1 para a cadeia t2
11.         addiu   $t0,$t0,1    #   4        incrementa ponteiro da cadeia t1
12.         addiu   $t1,$t1,1    #   4        incrementa ponteiro da cadeia t2
13.         j       l            #   4        volta a executar laço
14. fl:     addiu   $t1,$t1,1    #   4        aqui, fim da cópia
15.         sb      $zero,0($t1) #   4        coloca caracter fim de cadeia no fim de t2
16.         la      $t5,s        #   8        pega endereço do tamanho da cadeia tratada
17.         sw      $t2,0($t5)   #   4        armazena valor de contagem obtido no laço
18.         li      $v0,10       #   4        prepara-se para terminar
19.         syscall              #   4        e termina o programa
20.         .data
21. s:      .word  0
22. t1:     .asciiz       "Teste de Varios Valores: 1, 2, 3"
23. t2:     .space 40
```

Solução:

a) O programa possui um ciclo apenas, que executa uma vez para cada elemento da cadeia t1, ou seja, 32 vezes, mais a vez para o caracter de fim de cadeia. O laço ocupa as linhas 7-13. Nas 32 execuções, passa-se por todas as linhas do laço. Na última vez passa apenas pelas linhas 7 e 8. As instruções dos trechos das linhas 3-6 e 14-19 são executadas exatamente uma vez,

gastando 24+28=52 ciclos. Cada volta completa do laço gasta 29 ciclos para executar. Logo no laço se gasta 32*29+9=937 ciclos. Assim, o número de ciclos total para executar o programa é 52+937=**989 ciclos**.

b) Como f=1GHz, o período do relógio é T=(1/(10^9))s ou $1x10^{-9}$s. Para executar o programa leva-se então 989*$1x10^{-9}$s=**$9,89x10^{-7}$s**.

c) O programa copia a cadeia t1 para outra cadeia, t2 e no processo calcula o tamanho da cadeia original. As saídas geradas são o preenchimento da cadeia t2 com o mesmo conteúdo da cadeia t1 e a escrita da variável s com o tamanho destas cadeias, ou seja, 32.

d) O programa não possui subrotinas (não usa a instrução jal seguida de jr $ra).

3. [3,0 pontos]. Assuma uma frequência de relógio de 500 MHz para a organização MIPS multiciclo estudada em aula e desenhada acima. Ela foi modificada para reconhecer e executar a instrução LB do MIPS em 5 ciclos de relógio. Faça o que se pede.
   a) O programa abaixo é um teste de execução, idealizado para comparar o efeito das instruções LB e LBU. As posições de memória **position1** e **position2** correspondem respectivamente às duas primeiras posições da área de dados mostrada. Calcule os valores escritos em $t0 e $t1 após a execução do programa até a linha 11, explicando o que ocorre.
   b) Aponte pelo menos **6** módulos do processador MIPS Multiciclo que **NÃO** são utilizados durante a execução das linhas 2 a 11 do programa abaixo. Use os blocos do diagrama visto em aula, reproduzido na Questão 1 desta prova. Aqui, define-se como um bloco qualquer figura geométrica presente no diagrama. Por exemplo, qualquer um dos registradores identificados (Rs, PC, IMED, etc) é um bloco, qualquer um dos multiplexadores (M1 a M11) é um bloco, a ULA, o banco de registradores, são blocos, os retângulos correspondendo a hardware de extensão de valores são blocos, etc. Por **"não utilizado"** entende-se aqui um bloco que durante **toda a execução do programa** jamais vai conter ou processar dado útil à execução de pelo menos uma das instruções deste programa.

```
1. Address     Code        Basic                        Source
2. 0x00400000  0x3c01b4b4  lui $1,0xffffb4b4     2     li      $t0,0xb4b4b4b4
3. 0x00400004  0x3428b4b4  ori $8,$1,0x0000b4b4
4. 0x00400008  0x3c015d5d  lui $1,0x00005d5d     3     li      $t1,0x5d5d5d5d
5. 0x0040000c  0x34295d5d  ori $9,$1,0x00005d5d
6. 0x00400010  0x3c011001  lui $1,0x00001001     4     la      $t2,position1
7. 0x00400014  0x342a0000  ori $10,$1,0x00000000
8. 0x00400018  0x3c011001  lui $1,0x00001001     5     la      $t3,position2
9. 0x0040001c  0x342b0004  ori $11,$1,0x00000004
10.0x00400020  0x81480000  lb $8,0x00000000($10) 6     lb      $t0,0($t2)
11.0x00400024  0x91690000  lbu $9,0x00000000($11)7     lbu     $t1,0($t3)
12.Address     Data
13.0x10010000  0xbbbbbbbb  0xeeeeeeee
```

Solução:

a) O trecho de programa carrega $t0 com 0xB4B4B4B4 (linhas 2-3). Em seguida carrega $t1 com 0x5D5D5D5D (linhas 4-5). Depois carrega em $t2 o endereço de position1 e em $t3 o endereço de position2 (linhas 6-9). Na linha 10, a execução de lb escreve em $t0 0xFFFFFFBB, pois as instruções de leitura da memória sempre escrevem 32 bits no registrador destino. A instrução lb lê da memória apenas um byte (no caso o valor 0xbb), faz a **extensão de sinal** do mesmo e escreve o resultado de 32 bits no registrador destino ($t0). Na linha 11, a execução de lbu escreve em $t1 0x000000EE, pois as instruções de leitura da memória sempre escrevem 32 bits no registrador destino. A instrução lbu lê da memória apenas um byte (no caso o valor 0xee), faz a **extensão de zero** do mesmo e escreve o resultado de 32 bits no registrador destino ($t1).

b) Referenciando-se ao diagrama de blocos do processador na questão 1, os blocos não usados neste trecho de programa são 14: DIV, MULT, os Muxes que geram D_Hi e D_Lo, os registradores Hi e Lo, os blocos de extensão <<2 e "0000"&IR[25:0]&00, o Comparador, o mux M8, o registrador MDR, o bloco de extensão x"000000"&data[7:0], o registrador Rt e o bloco que gera o sinal **data** para escrita na memória. Qualquer lista de seis dos 14 blocos acima é uma resposta correta da questão.