Prova P4/PS **Disciplina: Organização de Computadores-EC**

Professor: Ney Laert Vilar Calazans

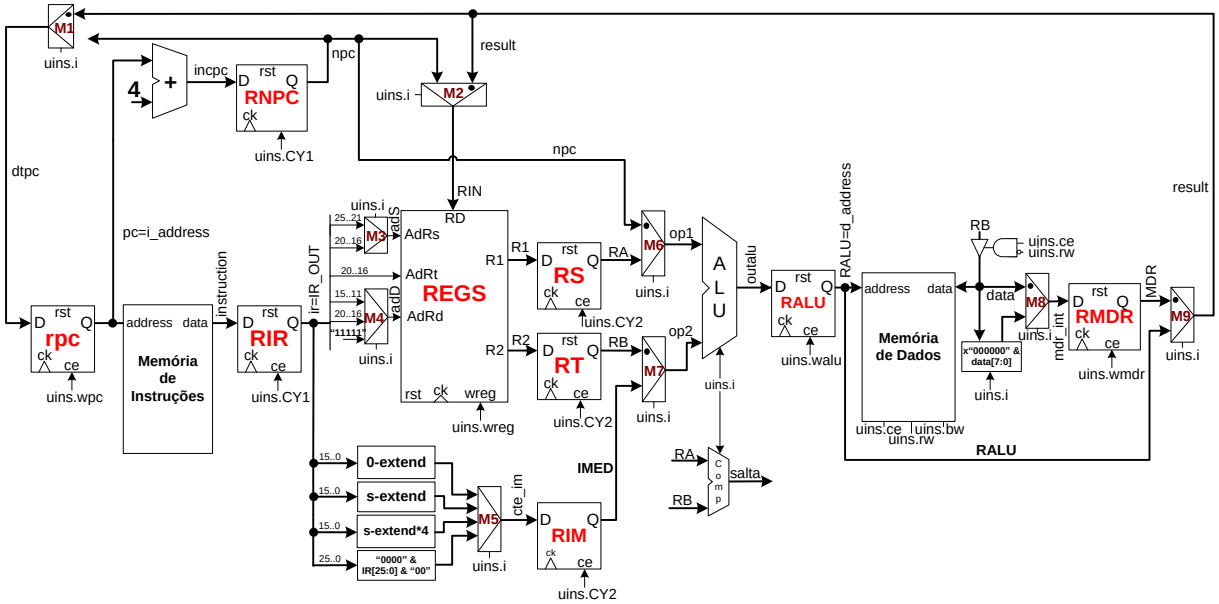
Aluno: 30/novembro/2012

1. (4 pontos) Assumindo uma frequência de relógio de 500 MHz para a organização MIPS multiciclo estudada em aula responda, para o programa abaixo, as seguintes questões:
   a) Qual o número de ciclos de relógio consumidos para a execução do programa? Conside a área de dados fornecida, assumindo que a pseudo-instrução **`la`** leva 8 ciclos de relógio para executar (sendo equivalente a **`lui+ori`**) e que as instruções **`syscall`** e a pseudo-instrução **`li`** (equivalente a um addiu) levam 4 ciclos de relógio para executar cada.
   b) Qual o tempo de execução do programa, em nanossegundos?
   c) O que faz este programa?

```
1.              .text
2.  main:       la          $s0, string
3.              la          $s1, letras
4.  loop:       lb          $t1, 0($s0)
5.              blez        $t1, fim
6.              sw          $t1, 0($s1)
7.              addiu       $s0,$s0, 1
8.              addiu       $s1,$s1, 4
9.              j           loop
10. fim:        li          $v0,10
11.             syscall
12.
13.             .data
14. string:     .asciiz     "teste de byte xy"
15. letras:     .word 0x0
```

2. (3,0 pontos) Verdadeiro ou Falso. Abaixo aparecem 10 afirmativas. Marque com V as afirmativas verdadeiras e com F as falsas. Se não souber a resposta correta, deixe em branco, pois cada resposta correta vale 0,3 pontos, mas cada resposta incorreta desconta 0,2 pontos do total positivo de pontos. Não é possível que a questão produza uma nota menor do que 0,0 pontos.
   a) ( ) Uma arquitetura RISC em geral possui maior abundância de registradores de dados que uma arquitetura CISC.
   b) ( ) O modo de endereçamento imediato, conforme usado no MIPS, obtém o operando constante de 32 bits a usar na instrução sempre acrescentando 16 bits em 0 à esquerda dos 16 bits menos significativos (bits 15 a 0, ou a metade direita) do código objeto da instrução.
   c) ( ) Se um processador usa apenas instruções onde cada código objeto ocupa exatamente 16 bits, o número máximo de instruções distintas que o processador pode ter no seu conjunto de instruções é 65536.
   d) ( ) Os campos de dado imediato em instruções do MIPS como addi, ori e xori permitem especificar apenas um subconjunto próprio dos valores possíveis de serem armazenados em um registrador do banco de registradores do processador.
   e) ( ) O modo de endereçamento pseudo-absoluto no MIPS é usado apenas em instruções de salto incondicional.
   f) ( ) O código objeto **0x110BBFF6** corresponde a uma instrução beq que quando saltar, o faz necessariamente para uma linha do programa anterior à linha onde há o beq.
   g) ( ) A instrução **ori $t0,$t0,0x0001** escreve em $t0 o número natural ímpar igual ou o mais próximo possível do valor que havia anteriormente neste registrador.
   h) ( ) O código objeto **0x0c100014** corresponde a um salto para uma subrotina que inicia necessariamente em uma posição de memória anterior à posição onde se encontra o salto.
   i) ( ) A memória do MIPS é endereçada a byte, mas é possível escrever nela dados de 8, 16, 32 , 64 ou 128 bits usando uma única instrução do processador.
   j) ( ) Não é possível somar duas constantes de 128 bits usando o processador MIPS.

3. (3,0 pontos) Considere a organização do bloco de dados multiciclo abaixo, que acomoda a execução de um subconjunto da arquitetura do conjunto de instruções do processador MIPS. Considere também as modificações que foram necessárias realizar nesta organização para dar suporte às novas instruções solicitadas no trabalho TP3. Em seguida, responda às questões abaixo.



a) [1 ponto] Marcar no desenho acima e/ou descrever todos os caminhos do bloco de dados efetivamente usados pela instrução **BNE Rs, Rt, Rótulo**. Isto significa marcar e/ou descrever em texto todos os caminhos por onde passa informação útil relevante à execução da instrução, ou seja, os dados e endereços que esta realmente necessita manipular.

b) [1 ponto] Diga qual operação é executada pela unidade lógica-aritmética (ALU) no terceiro ciclo de relógio da instrução **BNE Rs, Rt, Rótulo**, justificando sua resposta.

c) [1 ponto] Na implementação desta instrução o comparador desenhado abaixo da ALU na execução da instrução é usado ou não? Explique por quê.

Prova P4/PS **Disciplina: Organização de Computadores-EC**

Professor: Ney Laert Vilar Calazans

Aluno: 30/novembro/2012

Gabarito

1. (4 pontos) Assumindo uma frequência de relógio de 500 MHz para a organização MIPS multiciclo estudada em aula responda, para o programa abaixo, as seguintes questões:

a) Qual o número de ciclos de relógio consumidos para a execução do programa? Considere a área de dados fornecida, assumindo que a pseudo-instrução **`la`** leva 8 ciclos de relógio para executar (sendo equivalente a **`lui+ori`**) e que as instruções **`syscall`** e a pseudo-instrução **`li`** (equivalente a um addiu) levam 4 ciclos de relógio para executar cada.

b) Qual o tempo de execução do programa, em nanossegundos?

c) O que faz este programa?

```
1.              .text
2.  main:       la          $s0, string       # 8 ciclos
3.              la          $s1, letras       # 8 ciclos
4.  loop:       lbu         $t1, 0($s0)       # 5 ciclos
5.              blez        $t1, fim          # 4 ciclos
6.              sw          $t1, 0($s1)       # 4 ciclos
7.              addiu       $s0,$s0, 1        # 4 ciclos
8.              addiu       $s1,$s1, 4        # 4 ciclos
9.              j           loop              # 4 ciclos
10. fim:        li          $v0,10            # 4 ciclos
11.             syscall                       # 4 ciclos
12.
13.             .data
14. string:     .asciiz     "teste de byte xy"
15. letras:     .word 0x0
```

Solução da Questão 1

a) A solução desta questão é simples após se definir a estrutura de execução do programa. As linhas 2 e 3 e 10 e 11 são executadas apenas uma vez, perfazendo um total de 24 ciclos de relógio. As demais linhas executáveis (4 a 9) constituem um único laço. A última vez que se passa por este laço executa-se apenas suas duas primeiras instruções (linhas 4 e 5) gastando um total de 9 ciclos (lembrando que as instruções de leitura da memória [lw e lbu] gastam 5 ciclos para executar na MIPS multiciclo). As demais vezes executa-se todas as instruções do laço, tomando 6*4=25 ciclos. Analisando a estrutura de controle do programa pode-se ver que a condição de saída do laço é $t1 conter 0 ao executar a linha 5. Isto somente vai ocorrer quando a linha anterior ler um valor 0 da memória apontada por $s0, que inicialmente aponta para a cadeia string de caracteres. Ou seja quando o caractere fim de texto (NULL=00h) for alcançado. Como a cadeia possui 16 caracteres e após aparece o caractere fim de texto, o laço executará 17 vezes, visto que o laço percorre a cadeia caractere a caractere. Logo, o número de ciclos para executar este programa com os dados fornecidos é 16+16*25+9+8=433 ciclos.

b) Dado que o relógio é de 500MHz o tempo para executar um ciclo de relógio é $(1/500*10^6)$segundos, ou 2 ns. O tempo de execução então é 433ciclos*2ns = 866ns.

c) O programa recebe uma cadeia de caracteres compactada (1 caractere por byte de memória) e produz uma versão descompactada da mesma cadeia, com um caractere a cada palavra de 32 bits na memória.

Fim da Solução da Questão 1

2. (3,0 pontos) Verdadeiro ou Falso. Abaixo aparecem 10 afirmativas. Marque com V as afirmativas verdadeiras e com F as falsas. Se não souber a resposta correta, deixe em branco, pois cada resposta correta vale 0,3 pontos, mas cada resposta incorreta desconta 0,2 pontos do total positivo de pontos. Não é possível que a questão produza uma nota menor do que 0,0 pontos.
   a) ( ) Uma arquitetura RISC em geral possui maior abundância de registradores de dados que uma arquitetura CISC.
   b) ( ) O modo de endereçamento imediato, conforme usado no MIPS, obtém o operando constante de 32 bits a usar na instrução sempre acrescentando 16 bits em 0 à esquerda dos 16 bits menos significativos (bits 15 a 0, ou a metade direita) do código objeto da instrução.
   c) ( ) Se um processador usa apenas instruções onde cada código objeto ocupa exatamente 16 bits, o número máximo de instruções distintas que o processador pode ter no seu conjunto de instruções é 65536.
   d) ( ) Os campos de dado imediato em instruções do MIPS como addi, ori e xori permitem especificar apenas um subconjunto próprio dos valores possíveis de serem armazenados em um registrador do banco de registradores do processador.
   e) ( ) O modo de endereçamento pseudo-absoluto no MIPS é usado apenas em instruções de salto incondicional.
   f) ( ) O código objeto **0x110BBFF6** corresponde a uma instrução beq que quando saltar, o faz necessariamente para uma linha do programa anterior à linha onde há o beq.
   g) ( ) A instrução **ori $t0,$t0,0x0001** escreve em $t0 o número natural ímpar igual ou o mais próximo possível do valor que havia anteriormente neste registrador.
   h) ( ) O código objeto **0x0c100014** corresponde a um salto para uma subrotina que inicia necessariamente em uma posição de memória anterior à posição onde se encontra o salto.
   i) ( ) A memória do MIPS é endereçada a byte, mas é possível escrever nela dados de 8, 16, 32 , 64 ou 128 bits usando uma única instrução do processador.
   j) ( ) Não é possível somar duas constantes de 128 bits usando o processador MIPS.

Solução da Questão 2

a) (V) Uma arquitetura RISC em geral possui maior abundância de registradores de dados que uma arquitetura CISC.

Justificativa: VERDADEIRO, esta é uma das características fundamentais de máquinas RISC, pensada para evitar a busca frequente de dados na memória externa, um meio tipicamente menos eficiente que buscar informação em registradores.

b) (F) O modo de endereçamento imediato, conforme usado no MIPS, obtém o operando constante de 32 bits a usar na instrução sempre acrescentando 16 bits em 0 à esquerda dos 16 bits menos significativos (bits 15 a 0, ou a metade direita) do código objeto da instrução.

Justificativa: FALSO, este modo opera dependendo da instrução específica. Existem duas formas de gerar os 16 bits mais significativos do operando constante: por extensão de ZERO, como usado nas instruções lógicas como ANDI e ORI e por extensão de SINAL, como ocorre nas instruções ADDI e SLTIU. A extensão de ZERO funciona exatamente como a questão enuncia (acrescentando 16 bits em 0 à esquerda dos 16 bits menos significativos do código objeto da instrução). A extensão de SINAL, por outro lado, acrescenta ou 16 bits em 0 ou dezesseis bits em 1, sendo os 16 bits idênticos ao bit 15 dos 16 bits menos significativos do código objeto da instrução. A justificativa para esta funcionalidade é que a extensão de sinal preserva o sinal da constante de 16 bits original na representação de 32 bits, caso estes sejam interpretados como números expressos na codificação complemento de 2.

c) (V) Se um processador usa apenas instruções onde cada código objeto ocupa exatamente 16 bits, o número máximo de instruções distintas que o processador pode ter no seu conjunto de instruções é 65536.

Justificativa: VERDADEIRO, pois com 16 bits no código o objeto existem 2^{16} = 65536 códigos binários distintos. Se cada um destes corresponder a uma instrução válida do processador, atinge-se este valor máximo. Claro, esta hipótese é improvável de ocorrer na prática, pois isto implicaria que nenhuma das instruções do processador aceitasse qualquer operando.

d) (V) Os campos de dado imediato em instruções do MIPS como addi, ori e xori permitem especificar apenas um subconjunto próprio dos valores possíveis de serem armazenados em um registrador do banco de registradores do processador.

Justificativa: VERDADEIRO, pois o campo de dado imediato possui 16 bits de tamanho, enquanto os registradores do MIPS são de 32 bits, sendo assim impossível especificar qualquer valor de 32 bits como dado imediato de uma única instrução.

e) (V) O modo de endereçamento pseudo-absoluto no MIPS é usado apenas em instruções de salto incondicional.

Justificativa: VERDADEIRO. Ao usar este modo não há espaço no código objeto para armazenar nenhum operando condicional, pois o operando ocupa 26bits e o código da instrução ocupa os seis bits restantes e nenhuma código especifica qualquer tipo de teste implícito.

f) (V) O código objeto **0x110BBFF6** corresponde a uma instrução BEQ que quando saltar, o faz necessariamente para uma linha do programa anterior à linha onde há o BEQ.

Justificativa: VERDADEIRO. Os 6 primeiros bits do código objeto são, em binário, 000100, ou seja 4 em decimal/hexadecimal. Isto corresponde de fato ao código de uma instrução BEQ. Observando os 16 bits inferiores (0xBFF6), nota-se que se trata de um numeral que em complemento de 2 expressa um número negativo menor que -1. Assim, ao somar este valor, depois de estender seu sinal, ao valor do registrador PC durante a execução da instrução, será obtido um valor menor que o endereço da instrução corrente, ou seja, o endereço de uma linha anterior à da instrução BEQ.

g) (V) A instrução **ori $t0,$t0,0x0001** escreve em $t0 o número natural ímpar igual ou o mais próximo possível do valor que havia anteriormente neste registrador.

Justificativa: VERDADEIRO. De fato, o efeito desta instrução é eventualmente escrever ´1´ no bit 0 do registrador $t0, mantendo todos os demais. Como todo número ímpar tem a propriedade de ter o bit 0 em ´1´, isto justifica a primeira parte da afirmação. Como nenhum dos bits restantes de $t0 é alterado pelo ORI, o valor ou é igual ao valor anterior de $t0 (se $t0 já continha um valor ímpar) ou é o número ímpar obtido somando 1 ao número par existente previamente em $t0.

h) (F) O código objeto **0x0c100014** corresponde a um salto para uma subrotina que inicia necessariamente em uma posição de memória anterior à posição onde se encontra o salto.

Justificativa: FALSO. Os 6 primeiros bits do código objeto são, em binário, 000011, ou seja 3 em decimal/hexadecimal. Isto corresponde de fato ao código de uma instrução JAL de salto para subrotina. Contudo, JAL usa modo de endereçamento pseudo-absoluto, o que quer dizer que os 26 bits restantes do código objeto devem ser concatenados, depois de acrescentar dois bits em 0 à direita destes, com os 4 bits mais significativos do registrador PC. Ora, sem saber o valor específico do endereço onde se encontra a instrução JAL, é impossível determinar se o endereço assim obtido está antes ou depois da instrução de salto para subrotina JAL.

i) (F) A memória do MIPS é endereçada a byte, mas é possível escrever nela dados de 8, 16, 32, 64 ou 128 bits usando uma única instrução do processador.

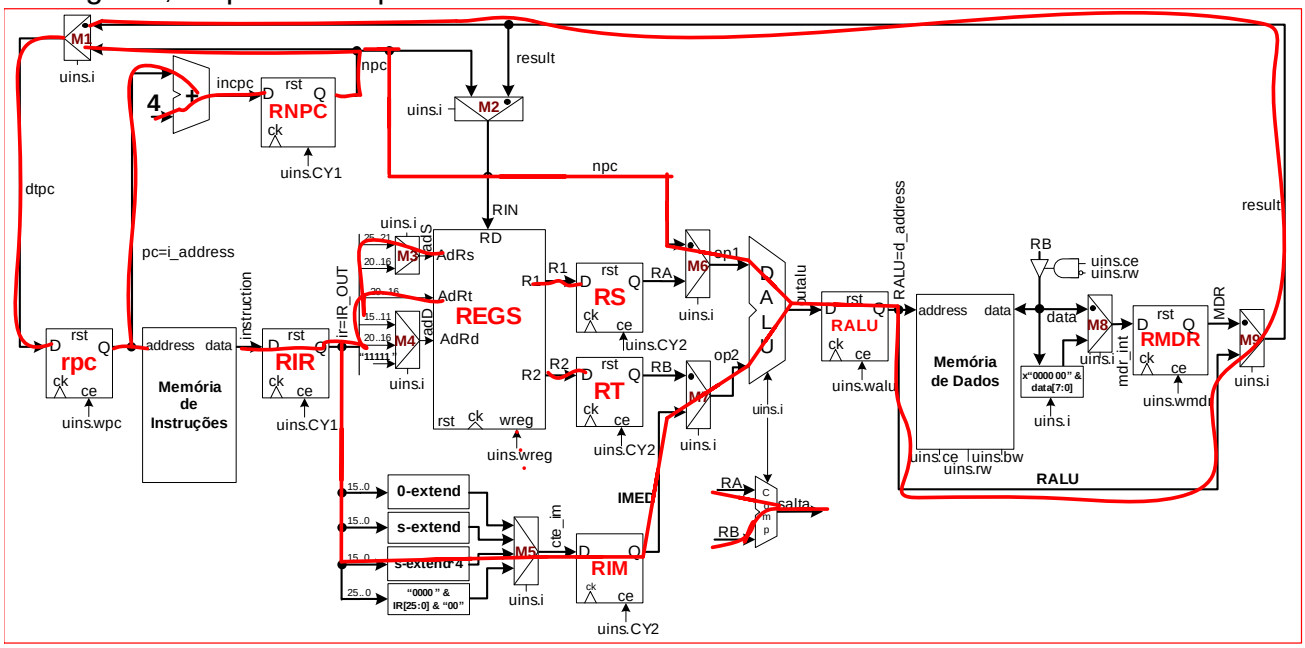
Justificativa: FALSO. Existem instruções que permitem escrever bytes (SB, 8 bits), meias palavras (e.g. SH, 16 bits) e palavras (e.g. SW, 32 bits) na memória. Até existe uma pseudo-instrução capaz de escrever 64 bits na memória (embora não tenhamos estudado esta pseudo-instrução aqui, por tratar-se de instrução específica da unidade de ponto flutuante). Contudo, nenhuma instrução isolada escreve 128 bits na memória.

j) (F) Não é possível somar duas constantes de 128 bits usando o processador MIPS.

Justificativa: FALSO. O processador MIPS executa um software armazenado na memória de instruções. Definitivamente é possível realizar operações de soma sobre valores de 128 bits expressando em software do processador um algoritmo que manipula porções de 32 bits. Logo, é possível somar duas constantes de 128 bits usando o processador MIPS.

Fim da Solução da Questão 2

3. (3,0 pontos) Considere a organização do bloco de dados multiciclo abaixo, que acomoda a execução de um subconjunto da arquitetura do conjunto de instruções do processador MIPS. Considere também as modificações que foram necessárias realizar nesta organização para dar suporte às novas instruções solicitadas no trabalho TP3. Em seguida, responda às questões abaixo.



a) [1 ponto] Marcar no desenho acima e/ou descrever todos os caminhos do bloco de dados efetivamente usados pela instrução **BNE Rs, Rt, Rótulo**. Isto significa marcar e/ou descrever em texto todos os caminhos por onde passa informação útil relevante à execução da instrução, ou seja, os dados e endereços que esta realmente necessita manipular.

b) [1 ponto] Diga qual operação é executada pela unidade lógica-aritmética (ALU) no terceiro ciclo de relógio da instrução **BNE Rs, Rt, Rótulo**, justificando sua resposta.

c) [1 ponto] Na implementação desta instrução o comparador desenhado abaixo da ALU na execução da instrução é usado ou não? Explique por quê.

Solução da Questão 3

a) As linhas desenhadas sobre a figura descrevem o fluxo de dados e de controle da instrução BNE. Como sempre o fluxo de execução começa com a busca da instrução na Memória de Instruções, trazendo para o IR e incrementando o PC e guardando seu novo valor em NPC. Em seguida busca-se os valor de Rs e Rt com os endereços contidos respectivamente nos campos de mesmo nome da instrução e coloca estes nos registrados RS e RT, respectivamente. Em paralelo com esta leitura do banco, os 15 bits inferiores da instrução são acrescentados de dois bits 0 à direita e os 18 bits resultantes são convertidos em 32 via

uma operação de extensão de sinal. O resultado é colocado no registrador IMED. Este valor é em seguida somado ao NPC na ULA e o resultado que é o endereço de salto é armazenado no registrador RALU. Em paralelo com esta operação de soma na ULA se faz a comparação de RS com RT no comparador Comp e o resultado fica disponível no fio de nome salta (‘1’ se os registradores são diferentes, ‘0’ caso contrário). Finalmente, o multiplexador M1 usa o sinal salta para decidir qual dentre os valores NPC e aquele em RALU irá ser copiado para o PC.

b) Como já descrito no item anterior a ULA faz uma soma de suas entradas op1 e op2, para obter o endereço para onde saltar, caso o salto seja executado.
c) Com também já explicado acima, sim, o comparador é usado para comparar se Rs e RT são diferentes ou não. Ele é usado pelo fato de a ULA estar ocupada ao mesmo tempo, realizando o cálculo do endereço de salto.

Fim da Solução da Questão 3