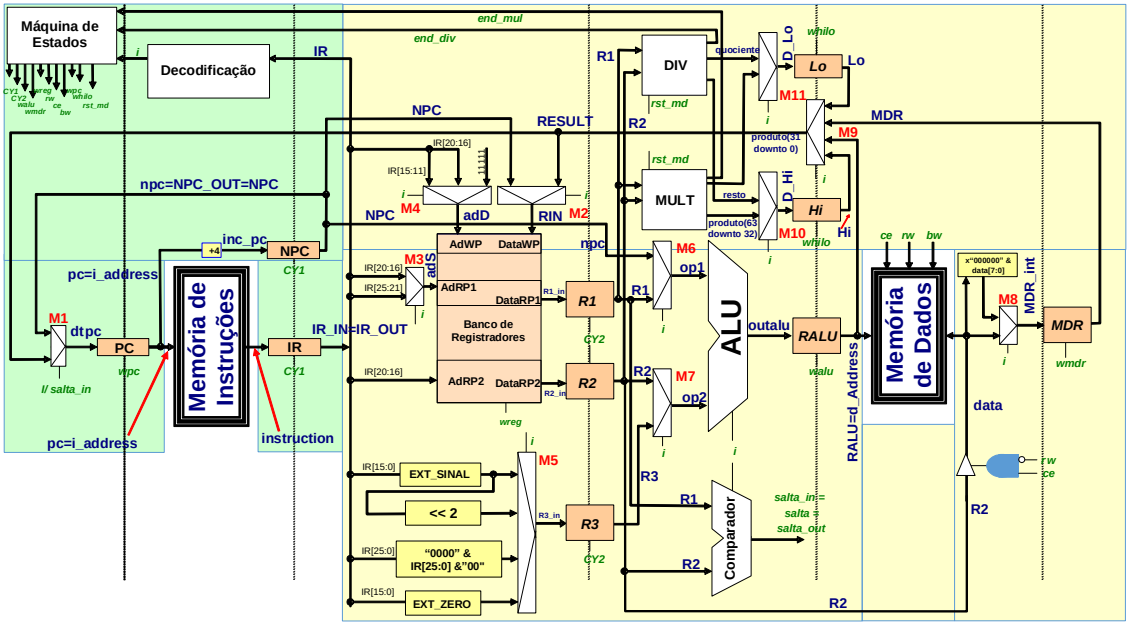
Prova P2 **Disciplina: Organização de Computadores**

Professor: Ney Laert Vilar Calazans

Aluno: 22/junho/2022

1. [2,0 pontos] Considere o processador MIPS multiciclo a seguir (visto em aula) e marque sobre ele todos os caminhos de dados e de controle usados para executar a instrução **JALR**. Ressalte também todos os sinais de controle efetivamente usados para executar a instrução. Não esqueça de identificar qual ou quais dos 11 sinais de controle gerados pela máquina de estados de controle do processador são relevantes para a execução da **JALR**. Use, além do diagrama de blocos fornecido abaixo, a especificação da instrução **JALR** que consta no Apêndice A do livro texto, a descrição VHDL da MIPS multiciclo, bem como a especificação textual da MIPS multiciclo.



2. [4 pontos] Assuma uma frequência de relógio de 400 MHz para uma organização **MIPS_S** (vista em aula) e com base neste fato, responda às questões abaixo sobre esta organização.

   (a) Qual o número de ciclos de relógio consumidos para a execução do programa abaixo nesta organização (Considere a área de dados fornecida, assumindo que a pseudo-instrução **la** leva 8 ciclos de relógio para executar sendo equivalente a duas instruções, um **lui** seguido de um **ori**, e que a instrução **syscall** leva 4 ciclos de relógio para executar).

   (b) Qual o tempo de execução do programa, em **segundos**.

   (c) O que faz este programa? Diga em uma frase.

   (d) Diga se o programa contém alguma subrotina. Em caso afirmativo, identifique-a no código.

```
1.          .data
2.   ar:    .word       0x12 0xff 0x3 0x14 0x878
3.   s:     .word       5
4.
5.          .text
6.          .globl      main
7.   main:  la          $t0,ar
8.          la          $t1,s
9.          lw          $t1,0($t1)
10.  l:     beq         $t1,$zero,end
11.         lw          $t2,0($t0)
12.         multu       $t2,$t2
13.         mflo        $t2
14.         sw          $t2,0($t0)
15.         addiu       $t0,$t0,4
16.         addiu       $t1,$t1,-1
17.         j           l
18.  end:   ori         $v0,$zero,10
19.         syscall
```

3. [2,5 pontos] Seja dado o trecho de programa abaixo. Suponha que este trecho é executado no pipeline do MIPS_S visto em aula, supondo a máxima capacidade de resolução de conflitos de dados (inclusive com estrutura mestre-escravo para acesso ao banco de registradores), mas sem predição de saltos. Responda às questões abaixo.
   (a) Qual o número mínimo ideal de ciclos de clock para a execução das 14 primeiras instruções deste programa no pipeline do MIPS? (do `add $t1,$zero,$zero` até o `j Loop`, supondo que o laço vai ser executado completamente ao menos 1 vez). (0,5 pontos)
   (b) Determine o número real de ciclos de clock para executar este mesmo trecho completamente 1 vez. Detalhe a execução no diagrama pipeline abaixo, indique todos os adiantamentos de dados que ocorrerem (se ocorrerem) e mostre as bolhas na posição correta, caso estas existam. (2 pontos)

INSTRUÇÃO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
addu $t1, $zero, $zero																						
Loop: addu $t2, $t1, $t1																						
addu $t2, $t2, $t2																						
addu $t3, $a2, $t2																						
lw $t4, 0($t3)																						
addu $t5, $a1, $t2																						
lw $t6, 0($t5)																						
addu $t6, $t4, $t6																						
addu $t5, $a3, $t2																						
sw $t6, 0($t5)																						
slt $t7, $t1, $a0																						
beq $t7, $zero, Exit																						
addiu $t1, $t1, 4																						
j Loop																						

Convenções: **X** - bolha ***** - estágio em que um salto é executado (carga no PC)
- - estágio não usado ⟶ - adiantamento ou leitura após escrita no mesmo ciclo
Estágios do pipeline: **B** (Busca), **D** (Decodificação), **E** (Execução), **M** (Memória), **W** (Write-back)

4. [1,5 pontos] Converta o numeral 25,375 para o formato de precisão simples (SP) do padrão IEEE-754. Assuma que o numeral está expresso na base 10, usando o padrão brasileiro de representação, onde a vírgula separa a parte inteira da parte fracionária. Após converter, diga se a conversão resultou em uma representação exata ou não do numeral original. Em outras palavras, há um erro de representação resultante da conversão ou não?

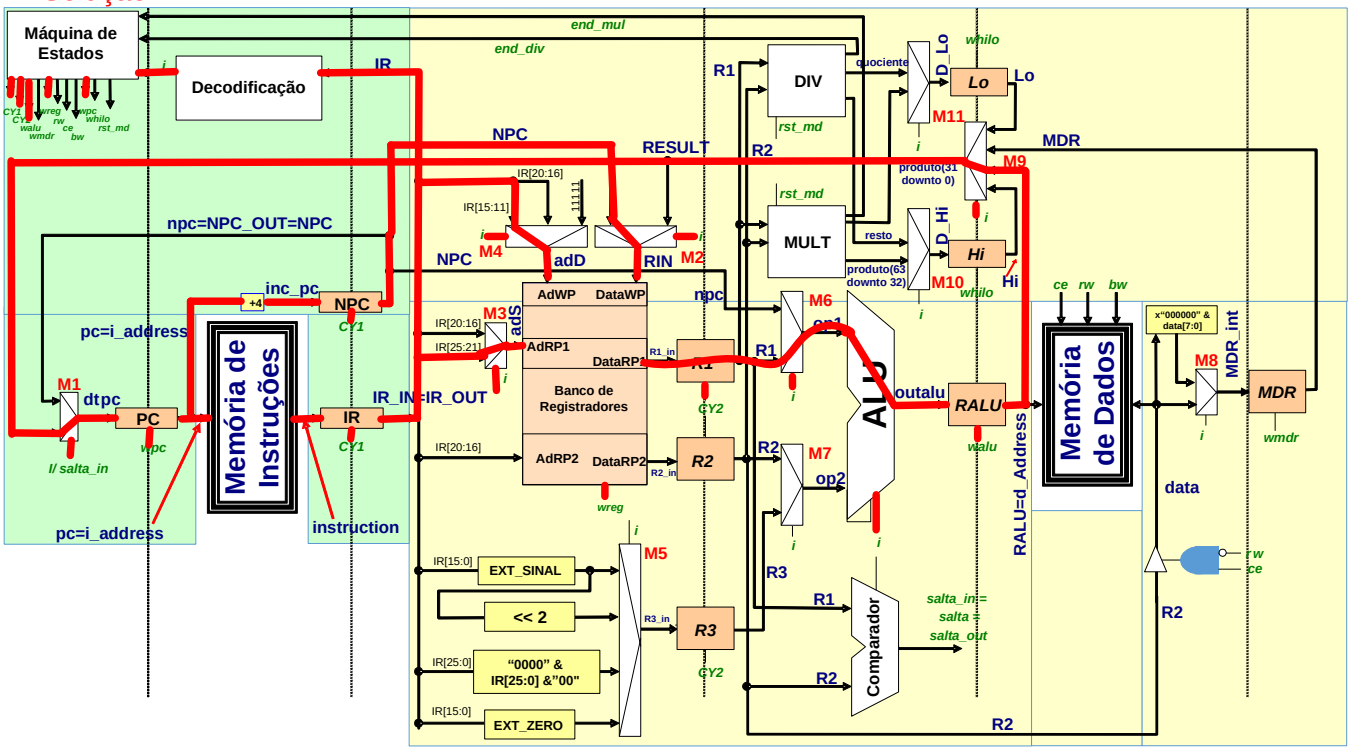
Prova P2 **Disciplina: Organização de Computadores**

Professor: Ney Laert Vilar Calazans

Aluno: **Gabarito** 22/junho/2022

1. [2,0 pontos] Considere o processador MIPS multiciclo a seguir (visto em aula) e marque sobre ele todos os caminhos de dados e de controle usados para executar a instrução JALR. Ressalte também todos os sinais de controle efetivamente usados para executar a instrução. Não esqueça de identificar qual ou quais dos 11 sinais de controle gerados pela máquina de estados de controle do processador são relevantes para a execução da JALR. Use, além do diagrama de blocos fornecido abaixo, a especificação da instrução JALR que consta no Apêndice A do livro texto, a descrição VHDL da MIPS multiciclo, bem como a especificação textual da MIPS multiciclo.

Solução:



2. [4 pontos] Assuma uma frequência de relógio de 400 MHz para uma organização MIPS_S (vista em aula) e com base neste fato, responda às questões abaixo sobre esta organização.
   (a) Qual o número de ciclos de relógio consumidos para a execução do programa abaixo nesta organização (Considere a área de dados fornecida, assumindo que a pseudo-instrução **la** leva 8 ciclos de relógio para executar sendo equivalente a duas instruções, um **lui** seguido de um **ori**, e que a instrução **syscall** leva 4 ciclos de relógio para executar).
   (b) Qual o tempo de execução do programa, em **segundos**.
   (c) O que faz este programa? Diga em uma frase.
   (d) Diga se o programa contém alguma subrotina. Em caso afirmativo, identifique-a no código.

Solução:

```
1.          .data
2.   ar:    .word       0x12 0xff 0x3 0x14 0x878
3.   s:     .word       5
4.
5.          .text
6.          .globl      main                # ciclos
7.   main:  la          $t0,ar              # 8 - $t0 recebe ponteiro para vetor ar
8.          la          $t1,s               # 8 -
9.          lw          $t1,0($t1)          # 5 - $t1 receve tamanho do vetor
10.  l:     beq         $t1,$zero,end       # 4 - Se tratamento do vetor acabou, fim
11.         lw          $t2,0($t0)          # 5 - Senão, lê elemento de ar para $t2
```

```
12.         multu       $t2,$t2          # 67 - Gera quadrado do elemento de ar
13.         mflo        $t2              # 4 - Busca resultado de lo para $t2
14.         sw          $t2,0($t0)       # 4 - Escreve resultado de volta em ar
15.         addiu       $t0,$t0,4        # 4 - Avança ponteiro para próx elemto
16.         addiu       $t1,$t1,-1       # 4 - Decrementa contador dos que restam
17.         j           l                # 4 - Volta a testar fim do processo
18. end:    ori         $v0,$zero,10     # 4 - Gera 10 (ident de syscall fim prog)
19.         syscall                      # 4 - Termina o programa
```

a) (1,5 pontos) O programa contém apenas um laço com três linhas preparatórias executadas antes dele (linhas 7 a 9) e duas instruções de fechamento do programa (linhas 18 e 19). As instruções fora do laço executam em 8+8+5 + 4+4=29 ciclos de clock. O laço ocupa as linhas 10 a 17 e é executado ou totalmente (para todo elemento do vetor **ar**) ou apenas a primeira linha (quando terminar o processamento). O laço executado totalmente gasta 4+5+67+4+4+4+4+4=96 ciclos e a última vez gasta apenas 4 ciclos para executar a instrução beq. Como a cadeia **ar** possui 5 valores, o laço é executado 6 vezes (5 de forma completa). Agora o número de ciclos para executar este programa pode ser facilmente determinado: num_ciclos= 29 + 5*96 + 4 = 513 ciclos.

b) (0,5 pontos) Como a frequência de execução dada é de 400MHz, um ciclo dura (1/(400 * 10^6))s ou 2,5ns ou 2,5x10^{-9} s. Logo, o tempo total de execução do programa, que gasta 513 ciclos de relógio (clock) para executar é 513*(2,5x10^{-9} s) ou seja, *tempo total de execução do programa=1,2825x10^{-6} s*.

c) (0,5 pontos) O programa eleva cada elemento (número inteiro) do vetor **ar** ao quadrado, substituindo os valores do vetor pelos quadrados de seus elementos.

d) (1,5 pontos) O programa não possui sub-rotinas, pois não existe nele uso de instruções jal/jalr (para entrar na subrotina) e jr $ra ou instrução equivalente para voltar da subrotina.

3. [2,5 pontos] Seja dado o trecho de programa abaixo. Suponha que este trecho é executado no pipeline do MIPS_S visto em aula, supondo a máxima capacidade de resolução de conflitos de dados (inclusive com estrutura mestre-escravo para acesso ao banco de registradores), mas sem predição de saltos. Responda às questões abaixo.

a) (0,5 pontos) Qual o número mínimo ideal de ciclos de clock para a execução das 14 primeiras instruções deste programa no pipeline do MIPS? (do add $t1,$zero,$zero até o j Loop, supondo que o laço vai ser executado completamente ao menos 1 vez).

Solução: Como são 14 instruções e o pipeline possui 5 estágios, o número mínimo ideal de clocks para executar o trecho é 5 ciclos para terminar a primeira instrução (preenchendo cada estágio com uma instrução) e 1 ciclo para terminar cada instrução subsequente. Ou seja, o número mínimo ideal de clocks para executar o trecho é 5+13=18 ciclos de clocks.

b) (2 pontos) Determine o número real de ciclos de clock para executar este mesmo trecho completamente 1 vez. Detalhe a execução no diagrama pipeline abaixo, indique todos os adiantamentos de dados que ocorrerem (se ocorrerem) e mostre as bolhas na posição correta, caso estas existam.

Solução: Com a capacidade mencionada de resolver conflitos, o número real de ciclos de clock para executar o trecho é 19, apenas 1 a mais que o número ideal. O detalhamento está no diagrama pipeline abaixo. Isto supõe que o PC é atualizado sempre no 5º. estágio, mas poderia ser no 4º. estágio.

INSTRUÇÃO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
addu $t1, $zero, $zero	B	D	E	-	W																	
addu $t2, $t1, $t1		B	D	E	-	W																
addu $t2, $t2, $t2			B	D	E	-	W															
addu $t3, $a2, $t2				B	D	E	-	W														
lw $t4, 0($t3)					B	D	E	M	W													
addu $t5, $a1, $t2						B	D	E	-	W												
lw $t6, 0($t5)							B	D	E	M	W											
addu $t6, $t4, $t6								B	D	X	E	-	W									
addu $t5, $a3, $t2									B	X	D	E	-	W								
sw $t6, 0($t5)											B	D	E	M	W							
slt $t7, $t1, $a0												B	D	E	-	W						
beq $t7, $zero, Exit													B	D	E	-	W					
addiu $t1, $t1, 4														B	D	E	-	W				
j Loop															B	D	E	-*	W*			

Convenções: **X** - bolha; ***** - estágio em que um salto é executado (carga no PC); **- -** estágio não usado; → - adiantamento ou leitura após escrita no mesmo ciclo

Estágios do pipeline: **B** (Busca), **D** (Decodificação), **E** (Execução), **M** (Memória), **W** (Write-back)

4. [1,5 pontos] Converta o numeral 25,375 para o formato de precisão simples (SP) do padrão IEEE-754. Assuma que o numeral está expresso na base 10, usando o padrão brasileiro de representação, onde a vírgula separa a parte inteira da parte fracionária. Após converter, diga se a conversão resultou em uma representação exata ou não do numeral original. Em outras palavras, há um erro de representação resultante da conversão ou não?

Solução: O formato SP usa 32 bits. O número 25,375 é positivo, logo o campo de sinal do significando é 0. O número, quando convertido para binário, ponto fixo, fornece $(25,375)_{10} = (11001,011)_2$. Ao normalizar o resultado, tem-se $(25,375)_{10} = (11001,011)_2 = (1,1001011 * 2^4)_2$. Desta forma normalizada é fácil obter o expoente: 4+127(a polarização do expoente)=131, que convertido para base 2 em 8 bits fornece 10000011. O significando em 23 bits é obtido omitindo-se 1, e completando com 0s os bits menos significativos, o que fornece o seguinte: 10010110000000000000000. O numeral completo no formato IEEE-754 SP é então:

$$(01000001110010110000000000000000)_2 = 0x41cb0000$$

Claramente, esta é a representação exata de $(23,375)_{10}$, ou seja, não há erro gerado pela conversão, pois toda a parte fracionária (depois de normalizada) cabe nos 23 bits do significando.