**Laboratório V – Conflitos no Pipeline**

Nome: Gabriel Gatti da Silva Matrícula: 2021100336

Nome: Thiago F. N. Lahass Matrícula: 2021100178

Data: 15/12/22

**1. OBJETIVOS**

Após completar as atividades deste laboratório, você irá:

• Conhecer como funciona o pipeline da CPU MIPS;

• Verificar o ganho de desempenho da CPU pipeline com relação à CPU monociclo;

• Verificar a influência dos conflitos no desempenho da CPU pipeline;

• Verificar com alguns conflitos podem ser evitados pela reorganização de instruções.

**2. ATIVIDADES**

Código utilizado para realizar a atividade ( fibonacci.asm ):

*.data*

*num: .word 10*

*res: .word 0*

*vout: .space 400*

*.text*

*#carregando registradores com valores aleatórios*

*la $s1, Vout*

*lw $a0, num*

*li $t0, 1*

*li $t1, 1*

*li $t2, 1*

*loop:*

*ble $a0, $t0, end*

*sw $t2, 8($s1)*

*addi $s1, $s1, 4*

*addi $t0, $t0, 1*

*addi $14, $0, -2*

*move $t3, $t2*

*add $t2, $t2, $t1*

*move $t1, $t3*

*b loop*

*end:*

*sw $t2, 8($s1)*

*sw $t2, res*

**Tarefa 1:**

Código corrigido:

*.data*

*num: .word 10*

*res: .word 0*

*vout: .space 400*

*.text*

*#carregando registradores com valores aleatórios*

*la $s1, vout*

*lw $a0, num*

*li $t0, 2*

*li $t1, 1*

*li $t2, 1*

*sw $t1, 0($s1)*

*loop:*

*bel $a0, $t0, end*

*sw $t2, 4($s1)*

*addi $s1, $s1, 4*

*addi $t0, $t0, 1*

*addi $14, $0, -2*

*move $t3, $t2*

*add $t2, $t2, $t1*

*move $t1, $t3*

*b loop*

*end:*

*sw $t2, 4($s1)*

*sw $t2, res*

Tabela de resultados:

| *num* | *res* | *#instruções* |
| --- | --- | --- |
| 10 | 55 | 90 |
| 20 | 6765 | 190 |
| 30 | 832040 | 290 |
| 40 | 102334155 | 390 |

Tabela 1.

**Tarefa 2:**

**Questões:**

a) Quais são estes possíveis conflitos e a sua causa (dependência dados, mudança de fluxo)?

Temos que, o *addi* da linha 4 feito pela pseudo-instrução *li* não terminou antes que o mesmo registrador fosse utilizado pela instrução *sw* da linha 6, logo há um erro.

A instrução *slt* que faz parte da pseudo-instrução *ble*( linha 7 do bloco de instruções) não gravou o resultado devido no registrador de destino ( feito na 5a etapa) antes que a próxima instrução (*beq*) já tenha feito a comparação ( feito na 3a etapa) com registrador que seria destino do resultado do *slt*, e portanto é feito uma comparação indevida. Além disso, são realizadas operações indevidas quando o *beq* não faz a comparação antes de ir para a próxima instrução.

O próximo problema está na instrução *move $t1, $t3*, pois duas instruções acima há *move $t3, $t2*, logo esta instrução ainda terminou de carregar o conteúdo do *reg $t2* para o *reg $t3* antes que a instrução *move $t1, $t3*, que se utiliza do conteúdo de *reg $t3* tenha alterado seu valor.

Por fim, o último conflito ocorre na instrução *b loop*, pois o PC vai para a instrução da label *end* antes que a comparação da instrução tenha sido feita.

b) Quais conflitos podem ser resolvidos por reorganização das instruções durante a compilação? Apresente o código reorganizado.

Todos os conflitos supracitados podem ser resolvidos utilizando-se bolhas para que dê tempo das instruções chegarem no estágio que devem antes das instruções posteriores utilizar/modificar os elementos manipulados por eles.

Código reorganizado:

*.data*

*num: .word 10*

*res: .word 0*

*vout: .space 400*

*.text*

*#carregando registradores com valores aleatórios*

*la $s1, vout*

*lw $a0, num*

*li $t0, 2*

*li $t1, 1*

*li $t2, 1*

*nop*

*nop*

*sw $t1, 0($s1)*

*loop:*

*slt $1, $t0, $a0*

*nop*

*nop*

*nop*

*beq $1, $0, end*

*nop*

*nop*

*nop*

*sw $t2, 4($s1)*

*addi $s1, $s1, 4*

*addi $t0, $t0, 1*

*addi $14, $0, -2*

*move $t3, $t2*

*add $t2, $t2, $t1*

*nop*

*nop*

*move $t1, $t3*

*b loop*

*nop*

*nop*

*nop*

*end:*

*sw $t2, 4($s1)*

*sw $t2, res*

Tabela de resultados:

| *num* | *res* | *#instruções* |
| --- | --- | --- |
| 10 | 55 | 186 |
| 20 | 6765 | 400 |
| 30 | 832040 | 610 |
| 40 | 102334155 | 820 |

Tabela 2.

**Tarefa 3:**

Tabela de resultados:

| *num* | *res* | *#instruções* | *#bolhas* |
| --- | --- | --- | --- |
| 10 | 55 | 121 | 27 |
| 20 | 6765 | 251 | 57 |
| 30 | 832040 | 381 | 87 |
| 40 | 102334155 | 511 | 117 |

Tabela 3.

**Questões:**

Calcule o ganho de desempenho da CPU pipeline em comparação com a CPU monociclo para as mesmas entradas. Justifique sua resposta.

Rodando o programa para cada um dos valores de num = { 10, 20, 30, 40 }, obtemos a seguinte tabela do tempo gasto em cada uma das CPU’s:

| *num* | *Tempo (tp) gasto na CPU pipeline (ps)* | *Tempo (tm) gasto na CPU monociclo (ps)* | *Razão (tp)/(tm)* |
| --- | --- | --- | --- |
| 10 | 83700 | 48400 | 0,578 |
| 20 | 176700 | 100400 | 0,568 |
| 30 | 269700 | 152400 | 0,565 |
| 40 | 362700 | 204400 | 0,564 |

Tabela 4.

Portanto, vemos que houve aproximadamente uma redução de 43% no tempo gasto para executar o mesmo código.