**Pequena Lista de Exercícios resolvidos sobre o assunto Pipelines - OAP**

**Professor: Ney Laert Vilar Calazans**

Última alteração em: 27/maio/2022

1. (P1-2001\_1) Suponha que existe uma máquina onde as instruções de divisão gastam 15 ciclos de relógio (*clocks*) para executar, e todas as demais gastam, em média, 3 ciclos. Seja dado um programa a acelerar, onde 20% do número total de instruções executadas são divisões. A equipe de engenheiros de hardware menciona ser possível reduzir para 8 ciclos a execução da divisão. Porém, esta mudança acarretará um aumento de 15% no período de *clock*, nada mais sendo afetado. Responda sobre este assunto:
2. Que percentagem do tempo total a máquina original gasta fazendo divisões, para o programa em questão?
3. A alteração sugerida pela equipe de engenheiros de hardware é recomendável? Em outras palavras, haverá ganho de desempenho para o programa a acelerar? Se sim, de quanto?

Resposta:

1. Como os percentuais do total de instruções de cada classe são dados, bem como o CPI médio das classes, basta uma regra de três simples para formular a solução na regra de três, t representa o tempo de execução total):

20%das instruções\*15 clocks (por divisão)+80% das instruções\*3clocks (por instrução) --100%\*t

20%das instruções\*15 clocks (por divisão) ------------------------------------------------------------- x%\*t

* Resolvendo a regra de três, tem-se:
* **x%\*t = (100% t \* 20%\*15 clocks) / (20%\*15 clocks + 80%\*3clocks) = 55,56%\*t**

1. A alteração sugerida pela equipe de engenheiros de hardware é recomendável? Em outras palavras, haverá ganho de desempenho para o programa a acelerar? Se sim, de quanto?

* Para calcular se haverá ganho, basta computar a razão dos tempos antes e depois das alterações, usando a fórmula geral seguinte:

t = no. de instruções \* CPImédio \* T, onde T é o período de relógio

* o número de instruções não muda, o CPImédio muda, pois a alteração faz o número de clocks da divisão passar de 15 para 8. Finalmente, o período T muda, aumentando 15%. Para calcular a razão que dá o *speed-up*, ou seja, t\_antes/t\_depois, deve-se começar pelo cálculo do CPImédio antes e depois, ou seja:

CPImédio\_antes = (20%\*15 + 80%\*3)/100% = 5,4 clocks/instrução

CPImédio\_depois = (20%\*8 + 80%\*3)/100% = 4,0 clocks/instrução

* A partir daí, basta computar a razão, dividindo as fórmulas de t\_antes e t\_depois, notando que:

no. instruções antes = no. instruções depois;

T\_depois = 1,15\*T\_antes.

A fórmula do *speed-up* fica então speed-up = t\_antes/t\_depois = (5,4\*T\_antes)/(4,0\*1,15\*Tantes) = 1,173...

* **Ou seja, a alteração é recomendável, pois há ganho de desempenho, que será de um pouco mais de 17%.**

1. (P1-2001\_1) O código abaixo é um trecho de um programa escrito para o processador MIPS, para mover um *string* de uma região de memória para outra, regiões estas cujos endereços iniciais estão armazenados nos registradores $t2 e $t6, respectivamente.

**add $t1, $0, $0 ; $t1 é deslocamento do início da cadeia destino, inicia com 0**

**Loop: add $t3, $t2, $t1 ; $t2 aponta início da cadeia fonte, $t3 é endereço do caractere**

**lbu $t4, 0($t3) ; lê caractere da cadeia fonte: $t4 🡸 PMEM($t3)**

**add $t5, $t6, $t1 ; $t5 é endereço onde escrever caractere na cadeia destino**

**sb $t4, 0($t5) ; escreve caractere na cadeia destino: PMEM($t5) 🡸 $4**

**addi $t1, $t1, 1 ; incrementa deslocamento do início da cadeia destino**

**bne $t4, $0, Loop ; se caractere copiado não é o último (‘\n’), continua**

Supondo que o programa esteja sendo executado no pipeline do MIPS de 5 estágios, pede-se:

1. Qual o número de ciclos para executar uma vez este trecho na máquina MIPS-S sem pipeline vista em aula? Qual o número mínimo **ideal** de ciclos de *clock* para a execução das sete primeiras instruções deste programa no pipeline do MIPS? (do **add $t1,$0,$0** até **bne $t4, $0, Loop**, apenas 1 vez)
2. Determine o número **real** de ciclos para executar este mesmo trecho, desenhando o diagrama de tempos e explique a causa das bolhas, caso estas existam. Suponha que **não existe** **nenhum suporte para adiantamento de dados** (*forwarding*), **nem hardware para resolver conflitos de controle.** Suponha ainda que a detecção de conflitos se dá quando uma instrução está no estágio de decodificação em, comparação com instruções anteriores em estágios seguintes do pipeline.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| **add $t1, $0, $0** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Loop: add $t3, $t2, $t1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **lbu $t4, 0($t3)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $t5, $t6, $t1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **sb $t4, 0($t5)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **addi $t1, $t1, 1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **bne $t4, $0, Loop** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Convenções: X - bolha F - flush do pipeline**

**- - estágio não usado - adiantamento ou leitura após escrita no mesmo ciclo**

**\* - indica uso do estágio para salto (escrita no PC)**

**B - Busca -- D - decodificação e busca de operandos -- E - execução de operação -- M - Memória -- W - *write-back***

1. Repita a questão proposta no item b) para o caso da técnica de **adiantamento** ser utilizada. Escolha (diga qual é sua escolha e seja coerente com ela) se a máquina em questão possui estrutura de registradores do tipo mestre-escravo no banco de registradores.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| **add $t1, $0, $0** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Loop: add $t3, $t2, $t1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **lbu $t4, 0($t3)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $t5, $t6, $t1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **sb $t4, 0($t5)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **addi $t1, $t1, 1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **bne $t4, $0, Loop** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Resposta:**

1. **Na máquina MIPS-S todas as instruções gastam 4 ciclos, exceto as instruções de load (LW e LBU), que gastam 5 ciclos e as instruções de multiplicação e divisão, que gastam 67 ciclos. Assim, na MIPS-S executar uma vez o trecho gastaria: (4+4+5+4+4+4+4=29) ciclos de relógio. Em um pipeline ideal o trecho executaria em 11 ciclos (5 para preencher o pipe e terminar a primeira instrução e mais 6 para terminar a execução das 6 instruções seguintes do trecho.**
2. **Ver diagrama abaixo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | **20** | 21 | 22 | 23 | 24 | 25 |
| **add $t1, $0, $0** | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Loop: add $t3, $t2, $t1** |  | **B** | **D** | **X** | **X** | **X** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **lbu $t4, 0($t3)** |  |  | **B** | **X** | **X** | **X** | **D** | **X** | **X** | **X** | **E** | **M** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $t5, $t6, $t1** |  |  |  |  |  |  | **B** | **X** | **X** | **X** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |
| **sb $t4, 0($t5)** |  |  |  |  |  |  |  |  |  |  | **B** | **X** | **X** | **X** | **D** | **E** | **M** | **-** |  |  |  |  |  |  |  |
| **addi $t1, $t1, 1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |
| **bne $t4, $0, Loop** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **B** | **D** | **E** | **-\*** | **-** |  |  |  |  |  |

1. **Escolhe-se o uso de um banco de registradores com estrutura mestre-escravo. Assim, ver a resposta completa deste item no diagrama abaixo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | **11** | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| **add $t1, $0, $0** | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Loop: add $t3, $t2, $t1** |  | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **lbu $t4, 0($t3)** |  |  | **B** | **D** | **E** | **M** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $t5, $t6, $t1** |  |  |  | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **sb $t4, 0($t5)** |  |  |  |  | **B** | **D** | **E** | **M** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **addi $t1, $t1, 1** |  |  |  |  |  | **B** | **D** | **E** | **-** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **bne $t4, $0, Loop** |  |  |  |  |  |  | **B** | **D** | **E** | **-\*** | **W** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Note-se um caso particular de uso de adiantamento entre os ciclos 6-8 acima. No final do ciclo 6, o dado lido pela instrução lbu chega ao processador no estágio 4. No entanto, a instrução sb precisaria ler este valor durante a sua decodificação (no ciclo 6). Assim, deve-se adiantar o dado do estágio M no ciclo 6 ou para o estágio M da instrução add no ciclo 7, ou para o estágio E da instrução sb no mesmo ciclo 7. Se a primeira opção for usada, outro adiantamento tem de ser feito no ciclo 8, do estágio M da instrução add para o estágio M da sb. Esta é a opção mostrada na tabela acima. A segunda opção não necessitaria deste adiantamento adicional.**

1. (P1-2001\_1) Seja dado o trecho de programa abaixo, a ser executado no processador MIPS. Suponha uma implementação do MIPS com máxima capacidade de solução de conflitos de dados (inclusive a estrutura mestre escravo do Banco de Registradores, que permite leitura e escrita do mesmo registrador em um único ciclo), mas sem capacidade de realizar predição de saltos (ou seja, saltos condicionais são realizados no quarto estágio do pipeline). Mostre o diagrama pipeline para a execução das instruções do trecho durante os primeiros 22 ciclos de relógio, supondo que $5 aponta para o início de um vetor de 2 posições com o seguinte conteúdo: 125 000

**Loop: lw $3, 0($5)**

**beq $0, $3, Xuxu**

**add $3, $7, $3**

**addi $5, $5, 4**

**beq $0, $0, Loop**

**Xuxu: add $2, $2,$2**

**add $4,$4, $4**

**add $6, $6, $6**

**...**Resto do programa

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| **Loop: lw $3, 0($5)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **beq $0, $3, Xuxu** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $3, $7, $3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **addi $5, $5, 4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **beq $0, $0, Loop** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Xuxu: add $2, $2,$2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $4,$4, $4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **add $6,$6, $6** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Convenções**: **X - bolha F - flush do pipeline**

- **- estágio não usado** **- adiantamento ou leitura após escrita no mesmo ciclo**

**\* - indica uso do estágio para salto (escrita no PC)**

**B - Busca -- D - decodificação e busca de operandos -- E - execução de operação -- M - Memória -- W - *write-back***

**Resposta:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrução/Ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| **Loop: lw $3, 0($5)** | **B** | **D** | **E** | **M** | **W** |  |  |  |  | **B** | **D** | **E** | **M** | **W** |  |  |  |  |  |  |  |  |
| **beq $0, $3, Xuxu** |  | **B** | **D** | **X** | **E** | **-** | **-** |  |  |  | **B** | **D** | **X** | **E** | **-\*** | **-** |  |  |  |  |  |  |
| **add $3, $7, $3** |  |  | **B** | **X** | **D** | **E** | **-** | **W** |  |  |  | **B** | **X** | **D** | **E** | **F** | **F** |  |  |  |  |  |
| **addi $5, $5, 4** |  |  |  |  | **B** | **D** | **E** | **-** | **W** |  |  |  |  | **B** | **D** | **F** | **F** | **F** |  |  |  |  |
| **beq $0, $0, Loop** |  |  |  |  |  | **B** | **D** | **E** | **-\*** | **-** |  |  |  |  | **B** | **F** | **F** | **F** | **F** |  |  |  |
| **Xuxu: add $2, $2,$2** |  |  |  |  |  |  | **B** | **D** | **E** | **F** | **F** |  |  |  |  | **B** | **D** | **E** | **-** | **W** |  |  |
| **add $4,$4, $4** |  |  |  |  |  |  |  | **B** | **D** | **F** | **F** | **F** |  |  |  |  | **B** | **D** | **E** | **-** | **W** |  |
| **add $6,$6, $6** |  |  |  |  |  |  |  |  | **B** | **F** | **F** | **F** | **F** |  |  |  |  | **B** | **D** | **E** | **-** | **W** |

**Convenções**: **X - bolha F - flush do pipeline**

**- - estágio não usado** **- adiantamento ou leitura após escrita no mesmo ciclo**

**\* - indica uso do estágio para salto (escrita no PC)**

**B - Busca -- D - decodificação e busca de operandos -- E - execução de operação -- M - Memória -- W - *write-back***