LISTA DE EXERCÍCIOS 3 – CAPÍTULO 4

Infraestrutura de Hardware
Aluno: João Marcos Alcântara Vanderley
E-mail/Login: jmav@cin.ufpe.br
Professora: Edna Natividade da Silva Barros

1. [0.5] Explique o conceito de pipeline e descreva quais as vantagens e desvantagens deste tipo de implementação?
R: O conceito de pipeline se resume a execução de várias instruções ao mesmo tempo, cada uma em um estágio diferente. Esse tipo de implementação trás consigo a vantagem de melhorar o desempenho aumentando a taxa de execução das instruções, ademais, uma melhor utilização dos estágios. A parte desvantajosa dessa implementação são os conflitos que podem ocorrer (estrutural, dados e controle) e principalmente o fato de que para termos um pipeline eficiente, nós dependemos muito de como o conjunto de instruções foi construído, logo, dependendo do conjunto de instruções, o nível de complexidade da implementação do pipeline pode se tornar muito alta.

2. [1.0] Considere que os estágios individuais do caminho de dados têm as latências descritas na tabela 1. Também assuma que todas as instruções de uma determinada aplicação executada pelo processador podem ser divididas nas classes conforme apresentado na tabela 2.

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Tabela 1- Tempo dos estágios do pipeline

ALU/logic	Jump/branch	LDUR	STUR
42%	23%	25%	10%

Tabela 2 – Frequência de classes de instruções

a. Qual é o tempo de ciclo do clock em um processador com pipeline e sem pipeline (mono-ciclo)?
R: O tempo de ciclo do clock em um processador com pipeline é determinado pelo estágio mais lento, nesse caso o ID, logo sabemos que será . Isso se deve ao fato de que no pipeline, todos os estágios duram um único ciclo de clock, logo o ciclo de clock deve ser longo o suficiente para acomodar a operação mais lenta. Em um processador sem pipeline, cada instrução passa por todos os estágios, então para sabermos o tempo de ciclo de clock, basta somarmos o tempo de todos os estágios, que fica: 250 + 350 + 150 + 300 + 200 = 1250ps

b. Qual é a latência total de uma instrução LDUR em um processador com e sem pipeline?
R: Como a instrução LDUR é do "tipo" Load, sabemos que ela utiliza os 5 estágios.
Dessa forma, se for com pipeline: Latência total = 5 ∗ 350ps = 1750ps
Se for sem pipeline: Latência total = Soma de todos os estágios = 250ps + 350ps + 150ps + 300ps + 200ps = 1250ps

c. Se pudermos dividir um estágio do caminho de dados em pipeline em dois novos estágios, cada um com metade da latência do estágio original, qual estágio você dividiria e qual é o novo tempo de ciclo do clock do processador?
R: Dividiria o estágio mais longo, que nesse caso é o ID. Dessa forma conseguiríamos reduzir o tempo de ciclo, e, após realizarmos a divisão do estágio, o novo tempo de ciclo do clock do processador seria equivalente a 300ps, que é a latência do novo estágio mais longo, o MEM.

d. Supondo que não haja stalls ou conflitos, qual é a utilização da memória de dados?

R: A memória de dados é utilizada apenas pelas instruções LDUR e STUR. Dessa forma, como sabemos a frequência de classes de instruções, então teremos:
Distribuição da instrução LDUR no processador = 25%
Distribuição da instrução STUR no processador = 10%
Logo, a utilização da memória de dados seria dada por: 20% + 15% = 35% dos ciclos de clock

e. Supondo que não haja stalls ou conflitos, qual é a utilização da porta de escrita do registrador da unidade de "Registradores"?
R: A porta de escrita do registrador da unidade de "Registradores" deve ser utilizada pelas instruções ALU e LDUR. Temos o seguinte:
Distribuição da instrução LDUR no processador = 25%
Distribuição da instrução ALU no processador = 42%
Portanto, a utilização da porta de escrita do registrador será dada pela soma dessas duas instruções, ficando: 25% + 42% = 67%

3. [1.0] Considere o programa descrito abaixo.

```
LOOP: ld x10, 0(x13)
ld x11, 8(x13)
add x12, x10, x11
slli, x12, x12,2
addi x13, x13, 16
bnez x12, LOOP
```

a. Considerando que os conflitos de dados e de controle foram resolvidos com a inserção de NOPs pelo compilador, qual o código será compilado e como fica o diagrama de execução de pipeline para as duas primeiras iterações deste loop.
R:

```
1 LOOP: ld x10, 0(x13)
2       ld x11, 8(x13)
3       nop
4       nop
5       add x12, x10, x11
6       nop
7       nop
8       slli, x12, x12,2
9       addi x13, x13, 16
10      nop // esse nop é referente ao conflito de dado do slli que tem com a instrução de desvio(linha 11)
11      bnez x12, LOOP
12      nop
13      nop
14      nop

15 LOOP: ld x10, 0(x13)
16       ld x11, 8(x13)
17       nop
18       nop
19       add x12, x10, x11
20       nop
21       nop
22       slli, x12, x12,2
23       addi x13, x13, 16
24       nop // esse nop é referente ao conflito de dado do slli que tem com a instrução de desvio(linha 25)
```

25 bnez x12, LOOP
26 nop
27 nop
28 nop

O diagrama está na planilha em anexo.

b. Considerando que o loop executa 4 iterações, qual o CPI do programa?
R: O CPI é a divisão do número de ciclos de clock pela quantidade de instruções que estão sendo executadas. Logo, como todas as instruções estão dentro de um loop que executa 4 vezes, teremos que o cálculo do CPI será dado por:
CPI = (14*4 + 4) / (6*4) = 2,5

c. Suponha que foi implementada no pipeline uma unidade de detecção de conflitos
e uma unidade de encaminhamento, como fica o código gerado pelo compilador?
Qual o novo CPI?
R: Para detecção de conflitos e unidade de adiantamento basta retirar 1 ou 2 NOP´s dos conflitos de dados
1 LOOP: ld x10, 0(x13)
2 ld x11, 8(x13)
3 nop
4 add x12, x10, x11
5 slli, x12, x12,2
6 addi x13, x13, 16
7 bnez x12, LOOP
8 nop
9 nop
10 nop

11 LOOP: ld x10, 0(x13)
12 ld x11, 8(x13)
13 nop
14 add x12, x10, x11
15 slli, x12, x12,2
16 addi x13, x13, 16
17 bnez x12, LOOP
18 nop
19 nop
20 nop

21 LOOP: ld x10, 0(x13)
22 ld x11, 8(x13)
23 nop
24 add x12, x10, x11
25 slli, x12, x12,2
26 addi x13, x13, 16
27 bnez x12, LOOP
28 nop
29 nop
30 nop

31 LOOP: ld x10, 0(x13)
32 ld x11, 8(x13)
33 nop
34 add x12, x10, x11
35 slli, x12, x12,2
36 addi x13, x13, 16

37 bnez x12, LOOP
38 nop
39 nop
40 nop

Logo, o CPI fica:
CPI = (10*4 + 4) / (6*4) = 1,83

d. Considerando o pipeline da letra c, suponha que a resolução do desvio foi adiantada para o estágio ID, qual o novo CPI?
R:
1 LOOP: ld x10, 0(x13)
2 ld x11, 8(x13)
3 nop
4 add x12, x10, x11
5 slli, x12, x12,2
6 addi x13, x13, 16
7 bnez x12, LOOP
8 nop

9 LOOP: ld x10, 0(x13)
10 ld x11, 8(x13)
11 nop
12 add x12, x10, x11
13 slli, x12, x12,2
14 addi x13, x13, 16
15 bnez x12, LOOP
16 nop

17 LOOP: ld x10, 0(x13)
18 ld x11, 8(x13)
19 nop
20 add x12, x10, x11
21 slli, x12, x12,2
22 addi x13, x13, 16
23 bnez x12, LOOP
24 nop

25 LOOP: ld x10, 0(x13)
26 ld x11, 8(x13)
27 nop
28 add x12, x10, x11
29 slli, x12, x12,2
30 addi x13, x13, 16
31 bnez x12, LOOP
32 nop

Logo, o CPI fica:
CPI = (8*4 + 4) / (6*4) = 1,5

4. [1.5] Considere a sequência de instruções abaixo que é executada em um pipeline de cinco estágios

sub x15, x12, x11
ld x13, 8 (x15)
ld x12, 0 (x15)
or x13, x15, x13

and x13, x13, x12
sd x13, 0 (x15)

a. Calcule o tempo de execução considerando NOPs foram inseridos para garantir a execução correta.
R:
sub x15, x12, x11
nop
nop
ld x13, 8 (x15)
ld x12, 0 (x15)
nop
or x13, x15, x13
nop
nop
and x13, x13, x12
nop
nop
sd x13, 0 (x15)
O tempo de execução será as instruções originais somadas com os NOP's inseridos, além de somar 4 dos estágios restantes da última instrução. Dessa forma fica:
Tempo de execução = (6+7) + 4 = 17

b. É possível reorganizar o código para minimizar o número de NOPs necessários?
R: Nessa questão não conseguimos reorganizar o código de uma maneira em que o número de NOPs necessários (7 NOP's foram utilizados) seja reduzido.

c. Se o processador tem uma unidade de adiantamento, mas a unidade de detecção de conflito não foi implementada, o que acontece quando o código original é executado?
R: O código continua executando corretamente. Só precisamos de uma unidade de detecção de conflito apenas para inserir uma bolha quando a instrução seguinte a uma instrução load usa o resultado do load, o que não acontece no código dessa questão.

d. Se o processador tem uma unidade de adiantamento, especifique quais sinais são setados em cada ciclo nas unidades de adiantamento (fowarding) e de detecção de conflitos na Figura abaixo
R:

Ciclo de clock	Foward A	Foward B	PCWrite
1	00	00	1
2	00	00	1
3	00	00	1
4	10	00	1
5	01	00	1
6	00	00	1
7	00	01	1
8	00	00	1
9	00	00	1

5. [0.5] Descreva a técnica de previsão de desvio baseada em previsor de 2 bits.
R: Na previsão de branch dinâmica o hardware mede o comportamento da branch atual registrando o histórico recente de cada branch em um buffer(que é um dispositivo que serve para fazer o armazenamento de memória temporária), e basicamente assume que o futuro comportamento vai continuar o mesmo e faz previsões. Se a previsão der errado, o pipeline vai parar durante a nova busca e o hardware também atualizará o histórico de acordo. O preditor de 2 bits altera a previsão de desvio somente após duas previsões incorretas sucessivas. Sabemos que

dois bits são mantidos no buffer de previsão e há um total de quatro estados diferentes, dois desses estados correspondentes a um estado 'taken' e dois correspondentes a um estado 'not-taken'.

6. [1.0] A importância de ter um bom preditor de desvio depende da frequência com que os desvios condicionais são executados. Junto com a precisão do preditor de desvio, este dado determinará durante quanto tempo o pipeline fica paralisado devido aos desvios incorretos. Neste exercício, suponha que a frequência de execução das classes de instruções seja a seguinte:

Tipo R	Beqz/bnez	jal	ld	sd
40%	30%	5%	22%	8%

Assuma que a precisão de alguns preditores de desvios para a aplicação é dada por:

Always Taken	Always not taken	2 Bits
45%	55%	85%

a. Os ciclos de paralisação devido aos desvios mal previstos aumentam o CPI. Qual é o CPI extra devido a desvios errados com a técnica "Always Taken"? Suponha que os resultados dos desvios sejam determinados no estágio de EX e aplicados no estágio MEM. Adicionalmente não há conflitos de dados e que nenhum slot de atraso é usado.
R: Sabemos que cada desvio mal previsto irá resultar em três ciclos de stall, logo como a precisão da técnica "Always Taken" é de 45%, o CPI extra será dado por:
$$\mathbf{CPI = 3 * (1 - 0.45) * 0.3 = 0.495}$$

b. Calcule o CPI para o preditor "Always not taken".
R: Da mesma forma que a questão anterior, sabemos que cada desvio mal previsto irá resultar em três ciclos de stall, logo como a precisão com a técnica "Always not taken" é 55%, o CPI extra será dado por:
$$\mathbf{CPI = 3 * (1 - 0.55) * 0.3 = 0.405}$$

c. Calcule o CPI para o preditor de 2 bits.
R: Para o preditor de 2 bits será da mesma forma, só que com 85% de precisão, logo o CPI fica:
$$\mathbf{CPI = 3 * (1 - 0.85) * 0.3 = 0.135}$$

d. Com o preditor de 2 bits, que speed-up seria alcançado se pudéssemos converter metade das instruções de desvio para alguma instrução ALU? Suponha que as instruções preditas correta e incorretamente tenham a mesma chance de serem substituídas.
R: Se queremos converter metade das instruções, então basta calcularmos o CPI com 0.15, visto que teríamos 1.5 ciclos de stall. Logo, o CPI fica:
$$\mathbf{CPI = 3 * (1 - 0.85) * 0.15 = 0.0675}$$
Então o speed-up será dado por 0.135 / 0.0675 = 2, ou seja, 100%

e. Algumas instruções de branch são muito mais previsíveis do que outras. Se sabemos que 70% de todas as instruções de desvio executadas são desvios de loop-back fáceis de prever e que são sempre previstos corretamente, qual é a precisão do preditor de 2 bits nos 30% restantes das instruções de desvio?
R: Se nós temos um total de X instruções, nossa taxa de acerto nas previsões iria ser de 0.3 * 0.85 * Y com o preditor de 2 bits, assim como fizemos acima. Se considerarmos que 70% de todas as instruções de desvio executadas são de loop-back fáceis de prever e que são sempre previstas corretamente, iriamos ter a seguinte situação:
$$\mathbf{0.7 * 0.3 * 1 * Y + 0.3 * 0.3 * P * Y = 0.3 * 0.85 * Y}$$
$$\mathbf{P = 0.5}$$
Logo, a precisão do preditor de 2 bits nos 30% restantes das instruções de desvio é de 50% (isso porque sabemos que no total a precisão é de 85%).

7. [1.0] Considere o padrão de ocorrências de instruções de desvio em um programa dada

por: T, NT, T, T, NT. Qual a precisão para preditores abaixo:

a. Preditor estático: Always-taken
R: **Always taken : 3/5 = 60%**

b. Preditor estático: Always-not-taken
R: **Always not − taken : 2/5 = 40%**

c. Preditor dinâmico de 2 bits (considere como estado inicial (0,0) para os primeiros 4 desvios
R:
Resultados
T, NT, T, T

Valor do preditor no momento da previsão
00, 01, 00, 01

Verdadeiro ou Falso
F, V, F, F

Precisão
1 / 4 = 25%

d. Preditor dinâmico de 2 bits (considere como estado inicial (0,0) considerando que os desvios são executados em loop infinito
R: Para determinar a precisão no estado inicial, devemos trabalhar através das previsões de branch até que o valor do preditor comece a repetir. Logo, teremos:
Resultados
T, NT, T, T

Valor do preditor no momento da previsão
Ocorrência 1: 00, 01, 00, 01, 10
Ocorrência 2: 01, 10, 01, 10, 11
Ocorrência 3: 10, 11, 10, 11, 11
Ocorrência 4: 10, 11, 10, 11, 11

Verdadeiro ou Falso no estado inicial
V, F, V, V, F

Precisão
3 / 5 = 60%

8. [2.5] Neste exercício, o desempenho de processadores de 1-issue e 2-issue serão comparados, levando em consideração as transformações do programa que podem ser feitas para otimizar a execução de um processador com 2-issue estático. As questões neste exercício referem-se ao código abaixo que calcula o loop dado por:

```
for (i = 0; i != j; i+ = 2) {
c[i] = (d[i] + d[i+1])*2;
}
addi x12, x0, 0
jal x0,EP
AGAIN: slli x5, x12, 3
add x6, x10, x5
ld x7, 0 (x6)
ld x25, 8 (x6)
add x26, x7, x25
```

slli x26, x26, 2
add x27, x11, x5
sd x26, 0 (x27)
addi x12, x12, 2
EP: bne x12, x13, AGAIN

Com o uso dos seguintes registradores para armazenar as variáveis

i	j	End. d	End. c	Valores temporários
x12	x13	x10	x11	x5-x7, x25-x27

Suponha que o processador com static 2-issue tenha as seguintes propriedades:
1. Uma instrução deve ser uma operação de acesso à memória; a outra pode ser uma instrução aritmética / lógica ou um desvio.
2. O processador tem unidades de adiantamento e detecção de conflitos e o desvio é resolvido no estágio ID.
3. O processador tem uma previsão de desvio perfeita.
4. Duas instruções podem não ser despachadas juntas em um pacote se uma depender da outra.
5. Se um stall for necessário, ambas as instruções no pacote de issue devem parar.
Considerando as propriedades descritas acima responda às questões a seguir.

a. Desenhe um diagrama de pipeline mostrando como o código fornecido acima é executado no processador com 2-issue estático. Suponha que o loop termine após quatro iterações.
R: O diagrama encontra-se na planilha.

b. Qual é o speedup obtido considerando um processador de 1-issue e um de 2-issue? (Suponha que o loop execute milhares de iterações.)
R: Não ocorrerá speed-up porque o código vai rodar uma quantidade de vezes nos ciclos iguais para os 2 processadores, o 1-issue e o 2-issue.

c. Desenrole o código de forma que cada iteração do loop desenrolado trate de duas iterações do loop original. Em seguida, reorganize / reescreva seu código desenrolado para obter melhor desempenho no processador de um issue. Você pode assumir que j é um múltiplo de 4.
R:
beqz x13, DONE
addi x12, x0, 0
jal EP
AGAIN: slli x5, x12, 3
add x6, x10, x5
ld x7, 0 (x6)
ld x25, 8 (x6)
addi x12, x12, 2
add x26, x7, x25
add x27, x11, x5
sd x27, 0 (x26)
EP: bne x12, x13, AGAIN

d. Qual é o speedup obtido considerando um processador de 1-issue e um de 2-issue considerando os códigos otimizados da letra (c)?
R: Bom, para calcularmos o speed-up nesse caso basta notarmos a quantidade de ciclos por iteração de cada um dos processadores acima em sua forma otimizada. O primeiro tem 9 enquanto que o segundo tem 7.5, logo o speed-up será:
9 / 7,5 = 1,2

e. Repita os exercícios das letras (c) e (d), mas desta vez suponha que o processador 2-issue pode executar duas instruções aritméticas / lógicas juntas. (Em outras palavras, a primeira instrução em um pacote pode ser qualquer tipo de instrução, mas a segunda deve ser uma instrução aritmética ou lógica. Duas operações de memória não podem ser programadas ao mesmo tempo.)
R:

```
beqz x13, DONE
addi x12, x0, 0
jal EP
AGAIN: slli x5, x12, 3
add x6, x10, x5
ld x7, 0 (x6)
ld x25, 8 (x6)
addi x12, x12, 2
add x26, x7, x25
add x27, x11, x5
sd x27, 0 (x26)
EP: bne x12, x13, AGAIN
```

O speed-up será calculado de forma semelhante, porém agora os ciclos por instrução de cada iteração para os processadores 1-issue e 2-issue ficam, respectivamente, 7 e 6. Logo, o valor do speed-up fica:
7 / 6 = 1,14

f. Na sua opinião qual as vantagens e desvantagens de static multi-issue?
R: As vantagens de se utilizar o static multi-issue é o fato de instruções independentes poderem ser executadas no mesmo ciclo, que aumentam o desempenho de certo modo. Contudo, uma desvantagem é o grande número de registradores utilizados, o que ocasiona numa maior dificuldade na hora de fazer a implementação.

9. [1.0] Descreva a técnica de execução especulativa dinâmica com preditor de 2 bits. Quais as vantagens e desvantagens de se executar o código da questão anterior em um processador com execução especulativa dinâmica?
R: Nesse tipo de implementação, a CPU tem a autonomia para escolher, de acordo com as dependências e desvios, quais serão as instruções executadas em cada ciclo. Isso trás a vantagem de evitar conflitos (estrutural e de dados) além de não introduzir erro em caso de uma previsão falha. Contudo, uma desvantagem é que, caso o mecanismo de previsão não seja eficiente, a técnica não será eficiente.