Aluna: Camila Xavier de Medeiros

Lista de Exercícios – Cap. 4

1. [0.5] Explique o conceito de pipeline e descreva quais as vantagens e desvantagens deste tipo de implementação?

Resposta: Pipeline é uma técnica de implementação na qual várias instruções são executadas ao mesmo tempo, cada uma em um estágio diferente. Isso gera uma melhor utilização dos estágios e melhor taxa de execução. Ele apresenta inúmeras vantagens e sua principal é a melhora no desempenho, pois, mesmo não diminuindo o tempo da execução de uma tarefa, ele melhora a taxa de execução de um conjunto de tarefas. ou seja, ocorre a execução de múltiplas tarefas simultâneas usando diferentes recursos e que não precisam mais esperar um a instrução anterior acabar seu estágio para entrar no pipeline. Porém, essa implementação também possui desvantagens, como a ocorrência de conflitos estruturais (um recurso não pode ser usado por dois estágios ao mesmo tempo), de dados (necessita esperar que a instrução anterior complete a escrita no registrador), de controle (decisão de qual instrução vai ser executada depende da instrução em execução - desvio). Além disso, se os estágios não estão balanceados (possuem a mesma latência), a eficiência será menor, pois podem precisar usar NOP'S, bolhas, entre outros recursos que diminuem o desempenho desse tipo de implementação.

2. [1.0] Considere que os estágios individuais do caminho de dados têm as latências descritas na tabela 1. Também assumam que todas as instruções de uma determinada aplicação executadas 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
45%	20%	20%	15%

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)?

Resposta: **Com pipeline:** 350ps (o tempo do maior estágio);

Sem pipeline: 1250ps = 250ps+350ps+150ps+300ps+200ps (soma de todos os estágios = tempo de uma instrução);

b. Qual é a latência total de uma instrução LDUR em um processador com e sem pipeline?

Resposta: **1250ps** para ambos os casos.

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?

Resposta: Eu dividiria o ID que tem o tempo de 350ps (maior tempo entre os estágios) ficando 2 estágios de 175ps cada. E o tempo de ciclo de clock do processador ficaria 300ps (o maior estágio agora, que é o da MEM).

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

Resposta: 35% de 300ps = **105ps;**

e. Supondo que não haja stalls ou conflitos, qual é a utilização da porta de escrita do registrador da unidade de "Registradores"?

Resposta: 65% de 200ps = **130ps;**

3. [1.0] Considere o programa descrito abaixo.

```
LOOP: ld x10, 0(x13)
      ld x11, 8(x13)
      add x12, x10, x11
      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.

Resposta:

LOOP:

1- ld x10, 0(x13)
2- ld x11, 8(x13)
NOP
NOP
3- add x12, x10, x11
4-addi x13, x13, 16
NOP
5- bnez x12, LOOP
NOP
NOP
NOP

obs: O diagrama de execução está no documento do excel anexado junto com esse docs.

b. Considerando que o loop executa 4 iterações, qual o CPI do programa?

Resposta: 11 interações com os NOPS vezes 4 ciclos, mas os estágios (5) menos um, dividido por 5 interações vezes quatro que dá 20 (sem os NOPS) = 2,4.

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?

Resposta:

1- ld x10, 0(x13)
2- ld x11, 8(x13)
NOP
3- add x12, x10, x11
4-addi x13, x13, 16
5- bnez x12, LOOP
NOP
NOP
NOP

Novo CPI: 9 interações vezes 4 ciclos mais 5 (dos 5 estágios) menos um, dividido por 5 vezes 4, sem os NOPS em baixo = 2.

d. Considerando o pipeline da letra c, suponha que a resolução do desvio foi adiantada para o estágio ID, qual o novo CPI?

Resposta:

1- ld x10, 0(x13)
2- ld x11, 8(x13)
NOP
3- add x12, x10, x11
4-addi x13, x13, 16
5- bnez x12, LOOP
NOP

Novo CPI: 7 interações vezes 4 ciclos mais 5 (dos 5 estágios) menos um, dividido por 5 vezes 4 (sem os NOPS em baixo) = 1,6.

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 (x2)
or x13, x15, x13
sd x13, 0 (x15)
```

a. Calcule o tempo de execução considerando que NOPs foram inseridos para garantir a execução correta.

Resposta:

1-sub x15, x12, x11
NOP
NOP
2-ld x13, 8 (x15)
NOP
3-ld x12, 0 (x2)
4-or x13, x15, x13
NOP
NOP
5-sd x13, 0 (x15)

Tempo de execução: 10 instruções com os NOPS mais 5 (estágios) menos 1 = 14 ciclos de clock.

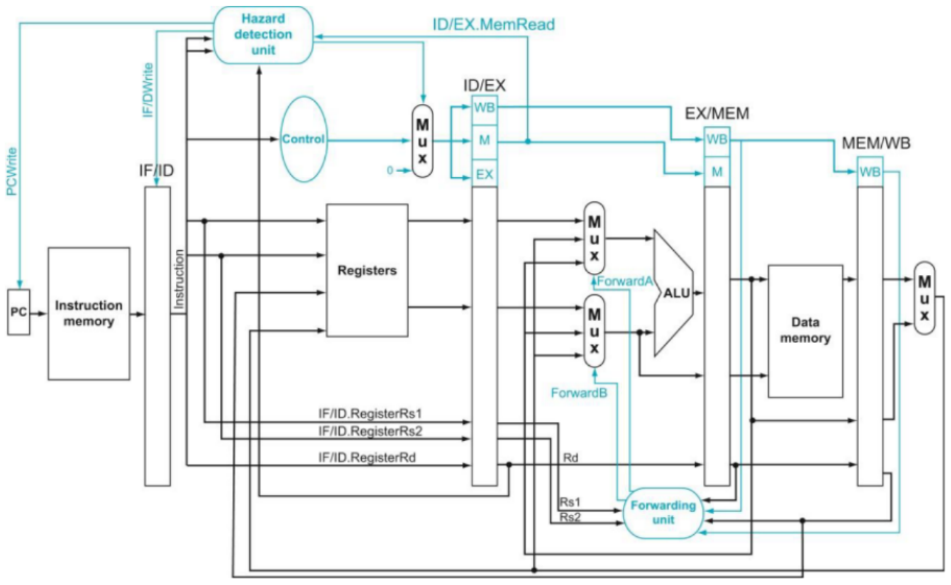
b. É possível reorganizar o código para minimizar o número de NOPs necessários?

Resposta: Infelizmente não por conta dos conflitos e das dependências existentes entre eles.

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?

Resposta: Então, se a unidade de detecção de conflito não for implementada, que é justamente a responsável por adicionar bolhas na CPU, quem irá resolver esse problema será o compilador, adicionando, assim, NOPs.

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



Pipeline com encaminhamento e detecção de conflito.

Resposta:

ciclo 1 = (fa = 00, fb = 00, pcw = 1);

ciclo 2 = (fa = 00, fb = 00, pcw = 1);

ciclo 3 = (fa = 00, fb = 00, pcw = 1);

ciclo 4 = (fa = 10, fb = 00, pcw = 1);

ciclo 5 = (fa = 01, fb = 00, pcw = 1);

ciclo 6 = (fa = 00, fb = 01, pcw = 1);

ciclo 7 = (fa = 00, fb = 00, pcw = 1);

ciclo 8 = (fa = 00, fb = 00, pcw = 1);

ciclo 9 = (fa = 00, fb = 00, pcw = 1).

5. [0.5] Descreva a técnica de previsão de desvio baseada em previsor de 2 bits.

Resposta: Como o próprio nome já diz, a técnica de previsão de desvio prevê o desvio e executa as instruções que viriam depois de acordo com a previsão, melhorando, assim, o desempenho do programa. já o preditor de 2 bit, é um dos mecanismos de previsão de desvio que apenas muda sua previsão (taken/not taken) quando ocorre dois erros consecutivos. Ela pode ser implementada com a tabela da história dos desvio (que guarda os bits atuais da previsão que podem ser 00, 01, 10 ou 11) e usa os bits menos significativos do PC para endereçar a tabela. além disso, ela pode pode ser implementada com o "branch targer buffer", que é um buffer com endereços de desvio e previsão que consegue buscar buscar a instrução imediatamente caso a previsão seja que o mesmo ocorra.

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.

Resposta: Se supormos que existem 1000 instruções o CPI original seria de 1,004. Além disso, a classe de instrução Beqz/bnez correspondem 300 instruções (30%), e com o erro de 55% no always taken, seriam 165 instruções. Adicionando 3 NOPs a cada instrução, e calculando o novo CPI que seria 1,499. O CPI extra seria de 0,495.

b. Calcule o CPI para o preditor "Always not taken".

Resposta: Se supormos que existem 1000 instruções o CPI original seria de 1,004. Além disso, a classe de instrução Beqz/bnez correspondem 300 instruções (30%), e com o erro de 45% no always not taken, seriam 135 instruções. Adicionando 3 NOPs a cada instrução, e calculando o novo CPI que seria 1,409. O CPI extra seria de 0,405.

c. Calcule o CPI para o preditor de 2 bits.

Resposta: Se supormos que existem 1000 instruções o CPI original seria de 1,004. Além disso, a classe de instrução Beqz/bnez correspondem 300 instruções, e com o erro de 15% no preditor de 2 bits, seriam 45 instruções. Adicionando 3 NOPs a cada instrução, e calculando o novo CPI que seria 1,139. O CPI extra seria de 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 corretas e incorretamente tenham a mesma chance de serem substituídas.

Resposta: O CPI anterior: 1,139. Transferindo metade das instruções de desvio para alguma de ALU, ficaríamos com 15% de instrução de desvio, logo, 15% de 300 é 45, adicionando 3 NOPs em cada erro, o novo CPI seria de 1,0715. Speed-up: 1,06299. O que significa que o novo é aproximadamente 6,3% mais rápido.

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?

Resposta: Se já temos certeza de 70% e a precisão do preditor de 2 bits é de 85%, então a precisão nova vezes os 30% dos desvios tem que dar 15%, ou seja a precisão nova é de 50%.

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

Resposta: Ele acerta 3 e erra duas das 5, logo = 60%.

b. Preditor estático: Always-not-taken .

Resposta: Ele acerta 3 e erra duas das 5, logo = 40%.

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

Resposta: considerando só os primeiros 4 desvios e começando a partir do 00 (NT) e só mudando a previsão a cada 2 erros = 25%.

d. Preditor dinâmico de 2 bits (considere como estado inicial (0,0) considerando que os desvios são executados em loop infinito).

Resposta: Quando os desvios entram em loop infinito, os 2 primeiros ciclos são irrelevantes, e logo depois ele fica acertando 3 (pois ele fica no estado taken sempre) e errando 2 das 5, logo 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];
}
```

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

Resposta: O diagrama de execução está no documento do excel anexa junto com esse docs.

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.)

Resposta:

```
for (i = 0; i != j; i+ = 2) {
        c[i] = d[i] + d[i+1];
}

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

CPI Com 1-issue: 13.1000 + 3 + 4 dividido por 9.1000 + 2. Considerei 1000 (pois na questão manda considerar milhares). São 13 dentro do loop, 3 fora e 4 dos estágios menos 1, dividido pelas instruções executáveis que são 9 dentro do loop e 2 fora = 1,4449.

Acesso a memória	Aritmética/lógica, desvio
NOP	ADDI x12, x0,0
NOP	SLLI x5,x12,3
NOP	JAL EP
NOP	ADD x6,x10,x5
LD x25,8 (x6)	ADD x27,x11,x5
LD x7, 0 (x6)	ADD x26,x7,x25
SD x27,0 (x26)	ADDI x12,x12,2
NOP	BNE x12,x13, AGAIN

CPI Com 2-issue: 1 dividido por 11/8 (o contrário de CPI) que é o número de instruções dividido por o número de duplas que dá aproximadamente 0,72.

Speedup: 0,72/1,4449 = 0,503. Ou seja, aproximadamente 50% mais rápido.

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.

Resposta: desenrolei 4 vezes o loop e reorganizando o código:

```
for (i = 0; i != j; i+ = 2) {
        c[i] = d[i] + d[i+1];
}

        addi x12, x0, 0
        jal EP
        NOP
AGAIN:slli x5, x12, 3
        add x6, x10, x5
        ld x7, 0 (x6)
        add x26, x7, x25
        ld x25, 8 (x6)
        add x27, x11, x5
        sd x27, 0 (x26)
        addi x12, x12, 2
        NOP
        NOP
AGAIN:slli x45, x12, 3
```

```
        add x14, x16, x45
        ld x15, 0 (x14)
        add x18, x7, x19
        ld x19, 8 (x14)
        add x20, x17, x45
        sd x20, 0 (x18)
        addi x12, x12, 2
        NOP
        NOP
AGAIN:slli x21, x12, 3
        add x23, x10, x21
        ld x7, 0 (x23)
        add x26, x7, x25
        ld x25, 8 (x23)
        add x27, x33, x21
        sd x27, 0 (x26)
        addi x12, x12, 2
        NOP
        NOP
AGAIN:slli x22, x12, 3
        add x24, x10, x22
        ld x28, 0 (x30)
        add x29, x7, x31
        ld x31, 8 (x30)
        add x32, x34, x22
        sd x32, 0 (x29)
        addi x12, x12, 2
        NOP
        NOP
EP: bne x12, x13, AGAIN
        NOP
```

d. Qual é o speedup obtido considerando um processador de 1-issue e um de 2-issue considerando os códigos otimizados da letra (c)?

Resposta:

CPI novo 1 issue: 12.4 + 4 + 3 dividido por 9.4 + 4 + 2 = 1,309;

CPI novo 2 issue: 1 dividido por 48/36= 0,72;

Speedup: 0,72/1,309 = 0,55 que é aproximadamente 55% mais rápido.

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.)

Resposta:

Aritmética/lógica	Qualquer uma
addi x12,x0,0	jal EP
add x27,x11,x5	ld x25, 8 (x6)
slli x5,x12,3	ld x7, 0(x6)
add x26,x7,x25	add x6,x10,x5
addi x12,x12,2	sd x27, 0 (x26)
bne x12,x13, AGAIN	NOP

x4

```
for (i = 0; i != j; i+ = 2) {
        c[i] = d[i] + d[i+1];
}

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

CPI novo 1 issue: 12.4 + 4 + 3 dividido por 9.4 + 4 + 2 = 1,309;

CPI novo 2 issue: 1 dividido por 11/7 = 0,63;

Speedup: 0,63/1,309 = 0,481 que é aproximadamente 48% mais rápido.

f. Na sua opinião qual as vantagens e desvantagens de static multi-issue?

Resposta: A principal vantagem do static multi-issue é que é possível executar instruções que não possuem dependências entre si, paralelamente, fazendo o CPI diminuir, diminuindo também o ciclo de clock o que por conseguinte também diminui o CPU time, melhorando o desempenho do código. porém, a grande desvantagem dele é que possui uma difícil utilização e manuseio, pois o processador aumenta o nível de sua complexidade pois terá que reorganizar o codigo, além de precisar separar as instruções em pacotes.

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?

Resposta: A execução especulativa dinâmica prevê, como o próprio nome da diz, o resultado das instruções para remover dependências e atrasos que possam existir, ou seja, ela define o fluxo das instruções, sendo importante, então, ter um bom mecanismo de previsão. Um deles é o preditor de 2 bit, é um mecanismo de previsão de desvio que apenas muda sua previsão (taken/not taken) quando ocorre dois erros consecutivos. Ela pode ser implementada com a tabela da história dos desvio (que guarda os bits atuais da previsão que podem ser 00, 01, 10 ou 11) e usa os bits menos significativos do PC para endereçar a tabela. além disso, ela pode pode ser implementada com o “branch targer buffer”, que é um buffer com endereços de desvio e previsão que consegue buscar buscar a instrução imediatamente caso a previsão seja que o mesmo ocorra. As vantagens de se executar o código da questão anterior com execução especulativa dinâmica são que nem sempre o loop unrolling é benéfico pois as vezes ele não consegue ser muito eficiente, então a especulação dinâmica consegue resolver esse problema. Por outro lado, desvantagens são que ocorre um aumento de complexidade devido ao escalonamento dinâmico, há um maior tempo para acessar a memória e dificuldade de resolver algumas dependências entre as instruções.