

Arquitetura e Organização de Computadores II

Técnicas básicas para explorar o Paralelismo em nível de instrução

Prof. Nilton Luiz Queiroz Jr.

Escalonamento de Pipeline

- Um modo de manter o pipeline totalmente ocupado é encontrar sequências de instruções não relacionadas;
- Para evitar stalls no pipeline é importante separar a execução de instruções dependentes das instruções que “causam dependência”
 - A distância ideal é a latência em ciclos da instrução que causa a dependência;
- Esse escalonamento é dependente da arquitetura e do programa;
 - Depende da quantidade de latência das unidades funcionais;
 - Da quantidade de de ILP do programa;



Escalonamento de Pipeline

- Suponha o seguinte código:

```
for (i=999;i>=0;i--)  
    x[i]=x[i]+s;
```

Esse trecho de código convertido para MIPS ficaria da seguinte maneira:

Loop:	L.D F0,0(R1)	;F0=Elemento do array
	ADD.D F4,F0,F2	;adiciona um valor ao elemento do array
	S.D F4,0(R1)	;armazena o resultado
	DADDUI R1,R1,#-8	;decrementa o ponteiro (double word)
	BNE R1,R2,Loop	;Volta ao loop se r1!=r2

- Assuma um pipeline de 5 estágios e as seguintes latências quando existe dependência de dados:

Instr. que produz o result.	Instr. que usa o result.	Latência em ciclos
FP ALU op	outra FP ALU op	3
FP ALU op	Store Double	2
Load Double	FP ALU op	1
Load Double	Store Double	0

Assumindo também uma latência de load de inteiros = 1

Como ficaria o preenchimento do pipeline na arquitetura MIPS, incluindo os stalls:

- Sem escalonamento? Com escalonamento?

Escalonamento de Pipeline

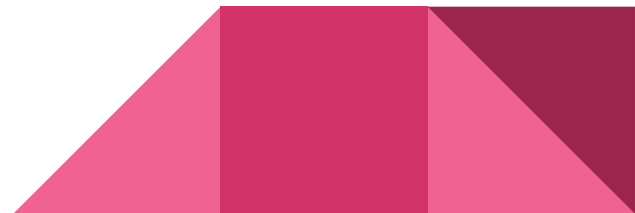
- Sem Escalonamento:

Ciclo

Loop:

1. L.D F0,0(R1)
2. stall
3. ADD.D F4,F0,F2
4. stall
5. stall
6. S.D F4,0(R1)
7. DADDUI R1,R1,#-8
8. stall
9. BNE R1,R2,Loop

- A instrução no ciclo 1 é um load, e sua próxima instrução é uma instrução ULA (1 ciclo de delay)
- A instrução no ciclo 3 é uma instrução ULA, sua próxima um store (2 ciclos de delay)
- A instrução no ciclo 6 é um store e a próxima uma instrução ULA (sem delay)
- A instrução no ciclo 7 é uma instrução ULA e em inteiros onde é necessário um ciclo para que o R1 possa ser lido



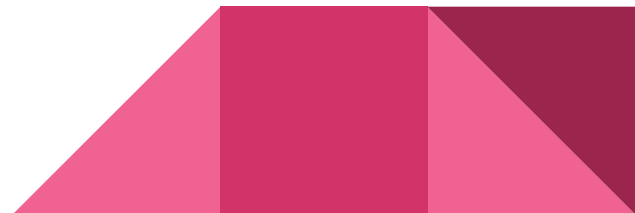
Escalonamento de Pipeline

- Com escalonamento:

Loop:

1. L.D F0,0(R1)
2. DADDUI R1,R1,#-8
3. ADD.D F4,F0,F2
4. stall
5. stall
6. S.D F4,8(R1)
7. BNE R1,R2,Loop

- Note que o escalonamento posiciona a instrução DADDUI longe da leitura de R1, e também aproveita o stall entre um Load e uma operação FP ALU, sendo assim são eliminados 2 stalls e a quantidade de ciclos cai para 7;
- Porém, perceba que ainda existe instruções que causam overhead;
 - As instruções vinculadas ao controle de loop
 - DADDUI;
 - BNE;



Escalonamento de Pipeline

- Existe alguma maneira de reduzir a quantidade de overhead causados pelas instruções de adição de inteiros e branches?



Escalonamento de Pipeline

- Existe alguma maneira de reduzir a quantidade de overhead causados pelas instruções de adição de inteiros e branches?
 - Loop unrolling (também chamado de desdobramento de loop);



Loop Unrolling

- Um método simples de aumentar o número de instruções relativas a um branch;
- Essa técnica simplesmente multiplica o corpo do loop diversas vezes;
 - Por fim faz com que o loop volte para o início, reajustando seu contador apenas uma vez;
- Ajuda no escalonamento;
- Para melhor desempenho aumenta o número de registradores necessários para o loop;
 - Apenas replicar o loop irá atrapalhar o escalonamento de maneira independente;
- Aumenta o tamanho do código final gerado;




Loop Unrolling

- Dado o seguinte trecho de código:

Loop:	L.D F0,0(R1)	;F0=Elemento do array
	ADD.D F4,F0,F2	;adiciona um valor ao elemento do array
	S.D F4,0(R1)	;armazena o resultado
	DADDUI R1,R1,#-8	;decrementa o ponteiro (double word)
	BNE R1,R2,Loop	;Volta ao loop se r1!=r2

Como ele ficaria após a aplicação de loop unrolling, sabendo que a diferença entre R1 e R2 é inicialmente um múltiplo de 32 (ou seja, a quantidade de iterações do loop é múltiplo de 4



Loop Unrolling

Loop:

1. L.D F0,0(R1)
2. ADD.D F4,F0,F2
3. S.D F4,0(R1) ;eliminou DADDUI & BNE
4. L.D F6,-8(R1)
5. ADD.D F8,F6,F2
6. S.D F8,-8(R1) ;eliminou DADDUI & BNE
7. L.D F10,-16(R1)
8. ADD.D F12,F10,F2
9. S.D F12,-16(R1) ;eliminou DADDUI & BNE
10. L.D F14,-24(R1)
11. ADD.D F16,F14,F2
12. S.D F16,-24(R1)
13. DADDUI R1,R1,#-32
14. BNE R1,R2,Loop



Loop Unrolling

- Considere novamente a tabela:

Instr. que produz o result.	Instr. que usa o result.	Latência em ciclos
FP ALU op	outra FP ALU op	3
FP ALU op	Store Double	2
Load Double	FP ALU op	1
Load Double	Store Double	0

Assumindo também uma latência de load de inteiros = 1

- Mostre o ciclo inicial para cada instrução do código após a aplicação do loop unrolling;

Loop Unrolling

	Ciclo
1. L.D F0,0(R1)	1
2. ADD.D F4,F0,F2	3
3. S.D F4,0(R1)	6
4. L.D F6,-8(R1)	7
5. ADD.D F8,F6,F2	9
6. S.D F8,-8(R1)	12
7. L.D F10,-16(R1)	13
8. ADD.D F12,F10,F2	15
9. S.D F12,-16(R1)	18
10. L.D F14,-24(R1)	19
11. ADD.D F16,F14,F2	21
12. S.D F16,-24(R1)	24
13. DADDUI R1,R1,#-32	25
14. BNE R1,R2,Loop	27



Loop Unrolling e escalonamento

- Mostre o código com as instruções escalonadas e os ciclos iniciais de cada instrução após o escalonamento;



Loop Unrolling e escalonamento

1.	L.D F0,0(R1)	1
2.	L.D F6,-8(R1)	2
3.	L.D F10,-16(R1)	3
4.	L.D F14,-24(R1)	4
5.	DADDUI R1,R1,#-32	5
6.	ADD.D F4,F0,F2	6
7.	ADD.D F8,F6,F2	7
8.	ADD.D F12,F10,F2	8
9.	ADD.D F16,F14,F2	9
10.	S.D F4,0(R1)	10
11.	S.D F8,-8(R1)	11
12.	S.D F12,-16(R1)	12
13.	S.D F16,-24(R1)	13
14.	BNE R1,R2,Loop	14



Resumo do Loop Unrolling

- Para fazer o loop unrolling foi necessário tomar as seguintes decisões:
 - Determinar que o loop unrolling seria útil;
 - Usar diferentes registradores
 - Para evitar conflitos que ocorreriam caso os mesmos registradores fossem usados;
 - Eliminar instruções extras;
 - Ajustar o código de término de iteração do loop;
- É necessário conhecer todas as dependências dentre as instruções;



Limitações do Loop unrolling

- Existem 3 limitações para os ganhos que podem ser alcançados pelo loop unrolling:
 - Diminuição na quantidade de overhead amortizado;
 - Tamanho de código;
 - Compilador;
 - Escalonamento dos registradores;



Exercícios

1. Considere a tabela abaixo e o seguinte trecho de código:

Instr. que produz o result.	Instr.que usa o result.	Latência em ciclos
FP ALU op	outra FP ALU op	3
FP ALU op	Store Double	2
Load Double	FP ALU op	1
Load Double	Store Double	0
Int ALU op	Int ALU op	0
Int ALU op	Branch	1

Loop:

1. L.D F4, 0 (R1)
2. MUL.D F8, F4, F0
3. L.D F6, 0 (R2)
4. ADD.D F10, F6, F2
5. ADD.D F12, F8, F10
6. S.D F12, 0 (R3)
7. DADDUI R1, R1, 8
8. DADDUI R2, R2, 8
9. DADDUI R3, R3, 8
10. DSUB R5, R4, R1
11. BNEZ R5, Loop

Faça o escalonamento de instruções para minimizar a quantidade de stalls

Exercícios

2. Dado o código do exercício 1 aplique a técnica de loop unrolling sabendo que a quantidade de iterações é múltipla de 2.

