Desvios com atraso e otimizações correlatas

Desvios com atraso: motivação

- **Desvios condicionais: beq $s1, $s2, L**
  - **Teste: $s1 == $s2 (V ou F)**
  - **Determinação do endereço da próxima instrução**
    - » **PC = PC + 4 (desvio não tomado)**
    - » **PC = PC + D (desvio tomado)**
- **Qual a próxima instrução a ser buscada?**
  - **Depende do resultado do teste**
    - » **Só se pode buscá-la quando resultado disponível**

Desvios com atraso: motivação

- **Micro-arquitetura mono-ciclo**
  - **Nenhuma incerteza na busca**
    - » **A próxima só é buscada depois que o desvio terminar**
- **Micro-arquitetura com pipeline**
  - **Busca simultânea à decodificação do desvio**
  - **Como buscar se não se sabe onde?**
- **Solução: pausa no pipeline**
  - **Até se conhecer o resultado do teste**
    - » **Degradação do desempenho devido aos desvios**

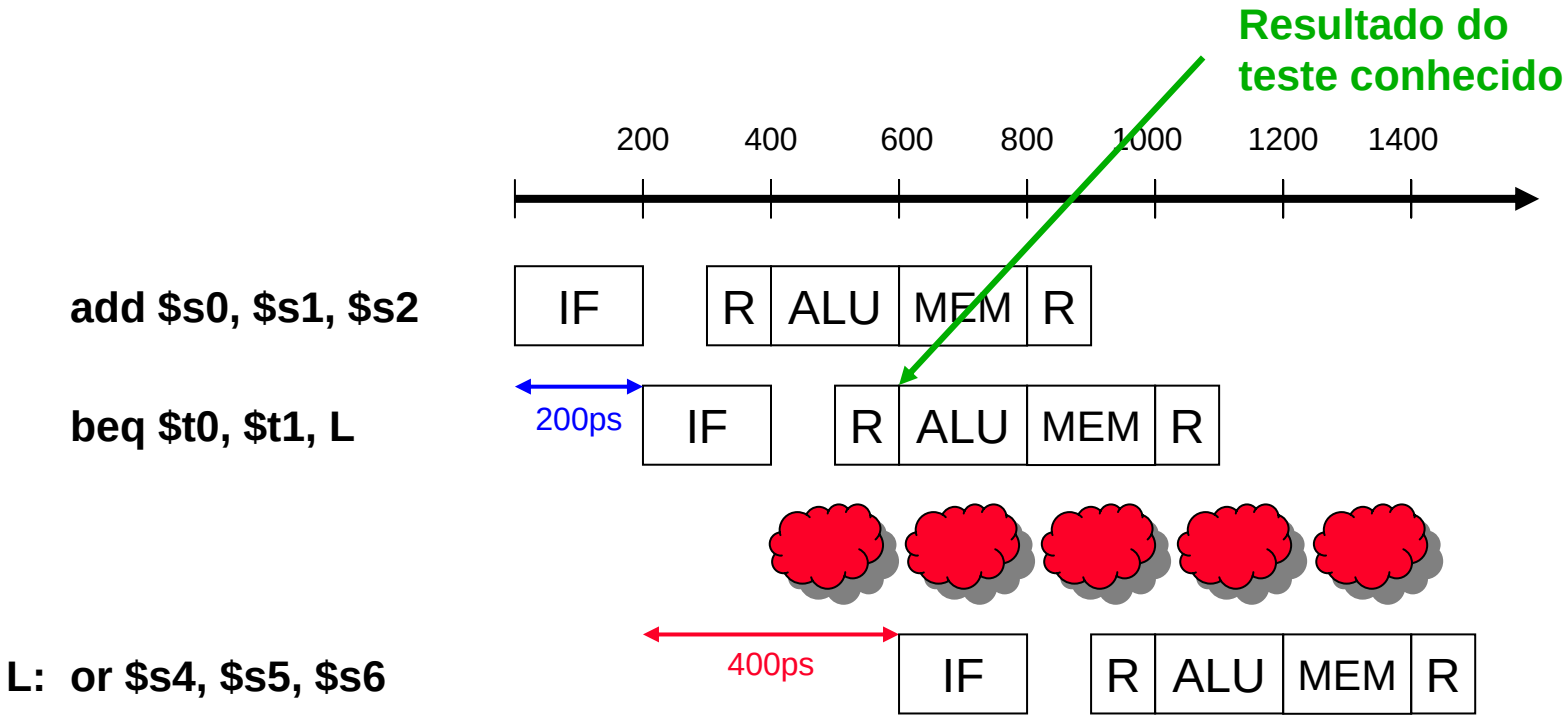
Pausa devida ao desvio

```
    add $s0, $s1, $s2

    beq $t0, $t1, L

    lw $t2, $t3, 300($s3)

    ...

L:  or $s4, $s5, $s6
```

Hipótese: teste resolvido no segundo estágio

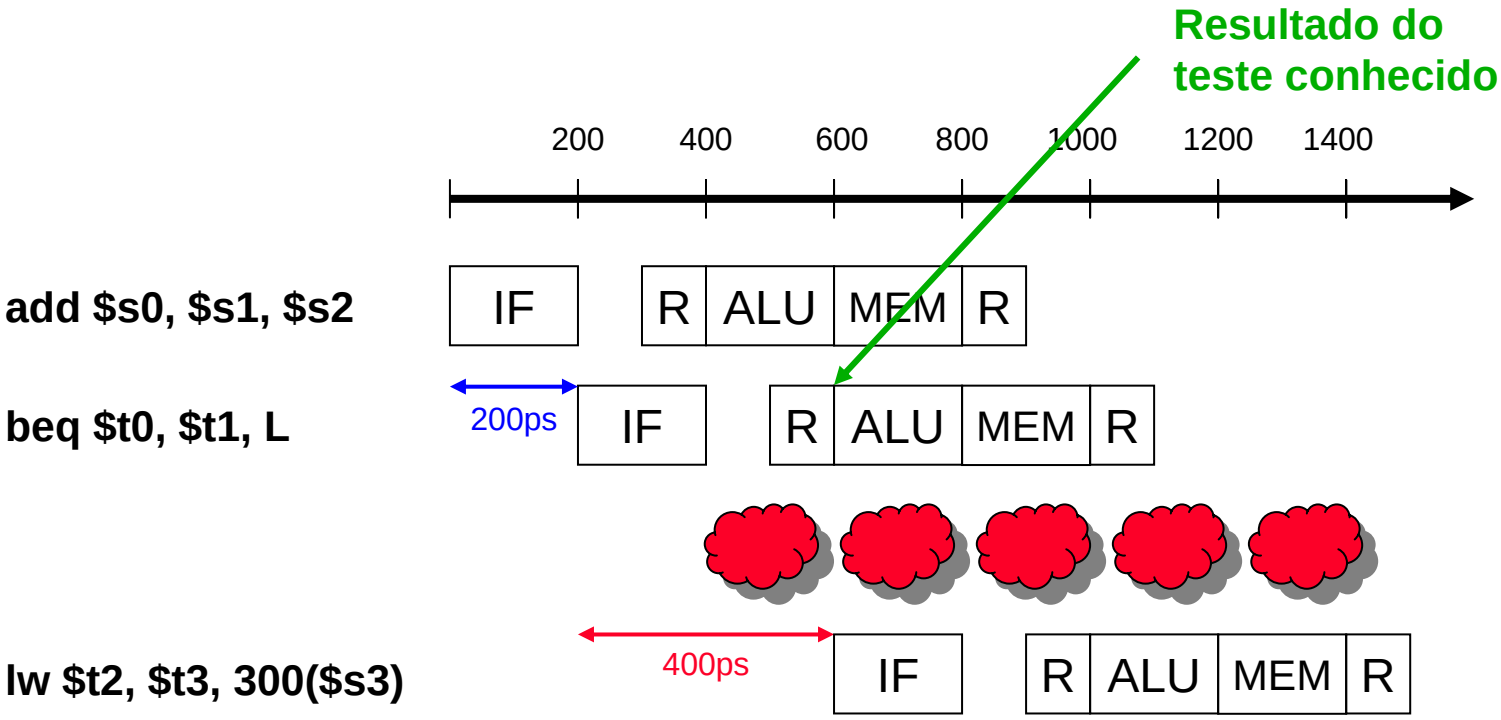
Cenários: desvio tomado ou não tomado

Pausa quando desvio tomado



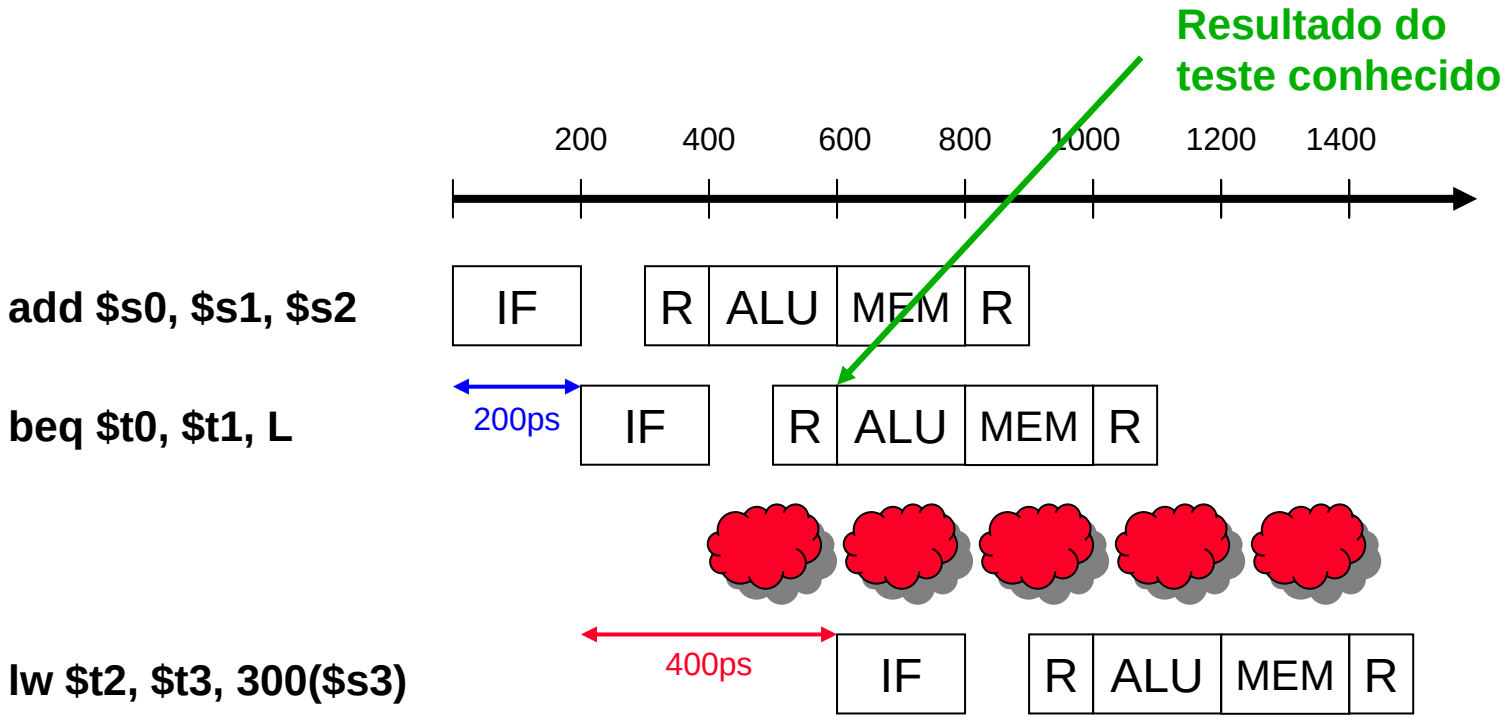
Penalidade de um ciclo

Pausa quando desvio não tomado



Penalidade de um ciclo

Impacto da pausa

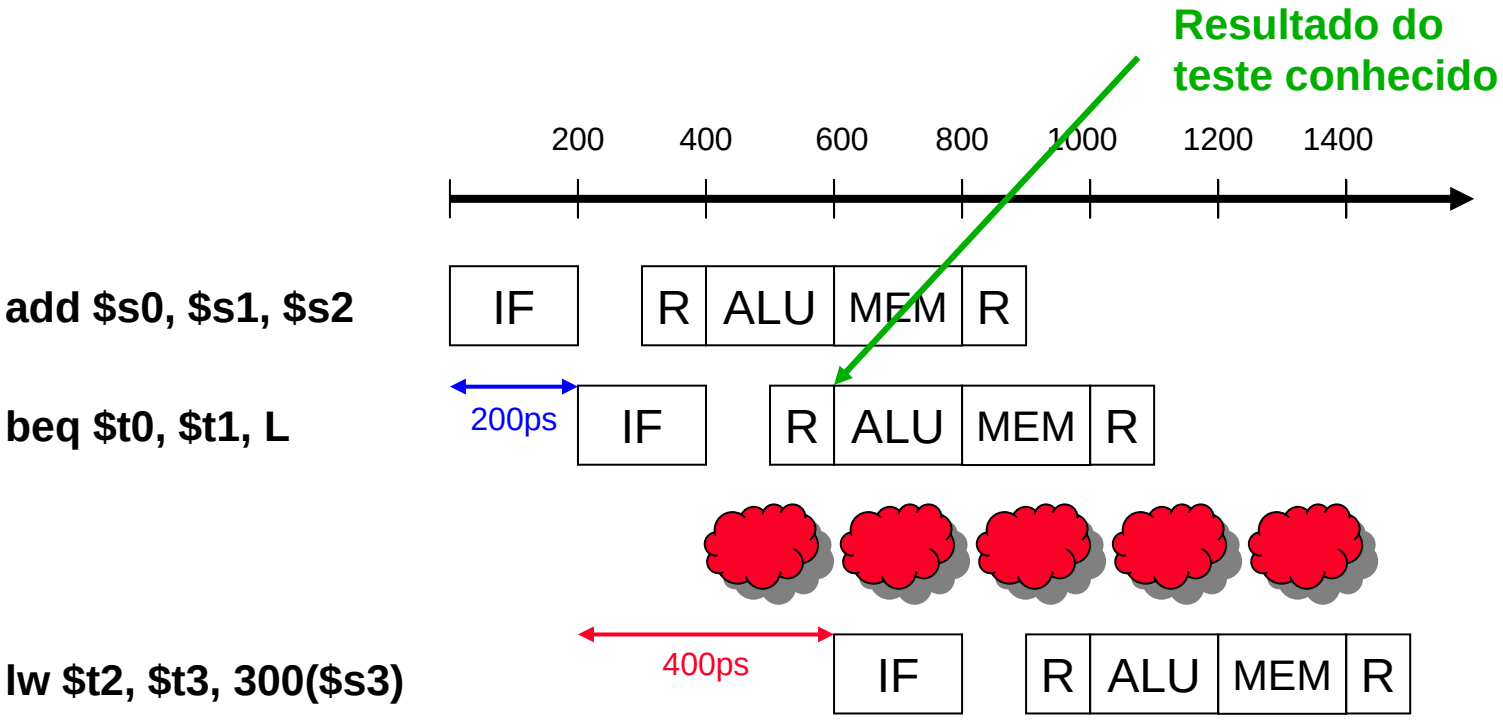


Conclusão: Penalidade de um ciclo sempre imposta à instrução que segue o desvio, qualquer que seja o resultado do teste.

Exemplo: se 13% de desvios (SPECint2000), então CPI = 1,13*

*** Se fosse possível distinguir (no início de ID) entre um branch e outras instruções**

Como contornar o problema?



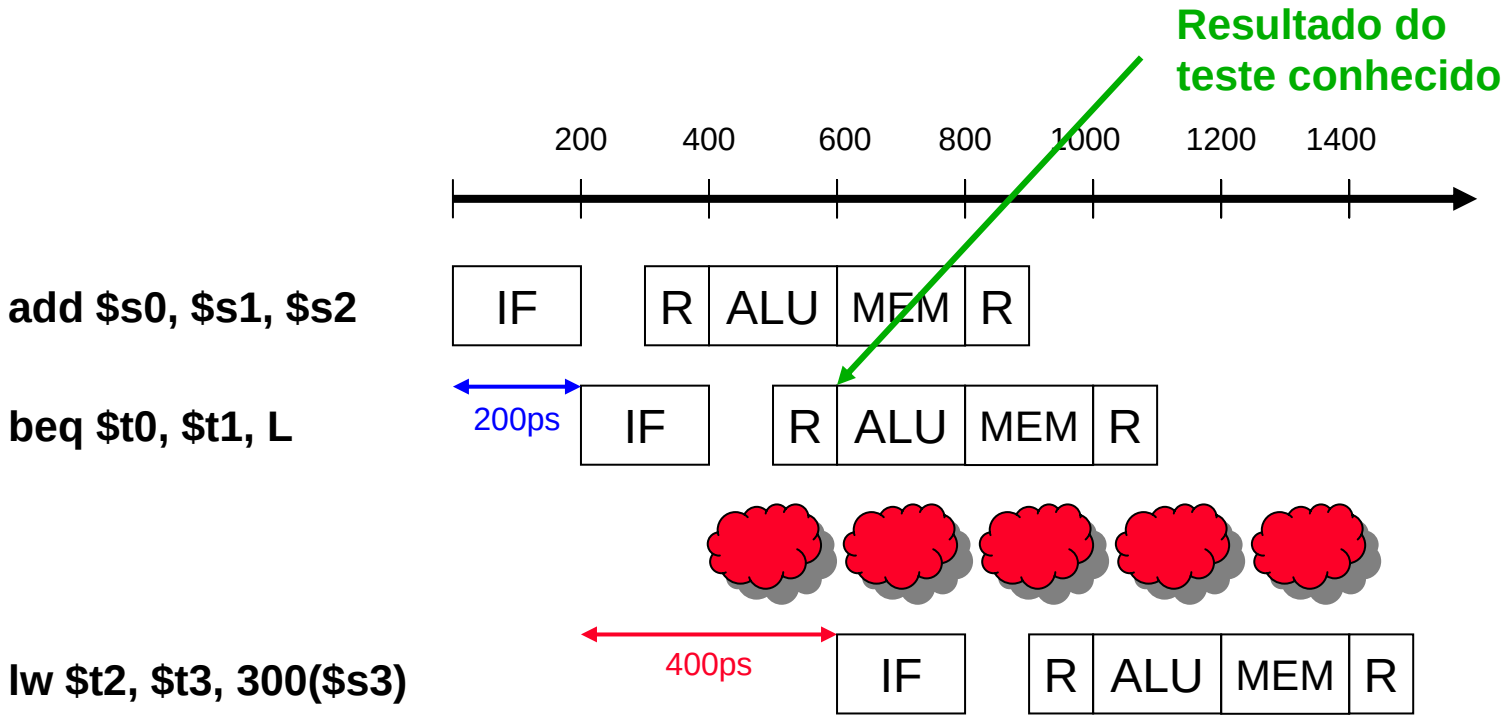
Por que não fazer algo de útil no lugar da bolha?

Por exemplo, por que não mover o add para o lugar dela?

— Porque o add nem sempre seria executado!

(violação da semântica do programa original)

Como contornar o problema?



Por que não fazer algo de útil no lugar da bolha?

Por exemplo, por que não mover o add para o lugar dela?

Por que não redefinir a semântica do desvio para que o add fosse sempre executado?

“Delayed branch”

- **Desvio com atraso**
  - **É tomado (se for o caso)**
  - **Depois de executada a instrução sucessora.**
- **Conseqüência:**
  - **A instrução sucessora será sempre executada**
    - » **Independentemente do resultado do teste**
- **Noção de “delay slot”**
  - **Instrução independente do desvio pode ser acomodada em um nicho**

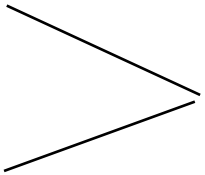
“Delayed branch”

Untaken branch	IF	ID	EX	ME	WB				
Instruction i+1		IF	ID	EX	ME	WB			
Instruction i+2			IF	ID	EX	ME	WB		
Instruction i+3				IF	ID	EX	ME	WB	
Instruction i+4					IF	ID	EX	ME	WB

Taken branch	IF	ID	EX	ME	WB				
Instruction i+1		IF	ID	EX	ME	WB			
Target			IF	ID	EX	ME	WB		
Target + 1				IF	ID	EX	ME	WB	
Target + 2					IF	ID	EX	ME	WB

“Delayed branch”

- **Generalização:**
  - **desvio condicional**
    sucessor seqüencial 1
    sucessor seqüencial 2
    .....
    sucessor seqüencial n
  - **instrução-alvo (se tomada)**



“Delay slots”

- **Uso prático:**
  - **1 único “delay slot” é suficiente**
    - » **Em pipelines de 5 estágios**
    - » **Acomoda teste e desvio (se feitos no estágio ID)**

“Delayed branch”

- **Para preencher o “delay slot”, ...**
- **Onde obter instruções?**
  - **Otimização 1: Procurar nos predecessores**
  - **Otimização 2: Procurar nos sucessores**
  - **Otimização 3: Procurar no endereço-alvo**

“Delayed branch”

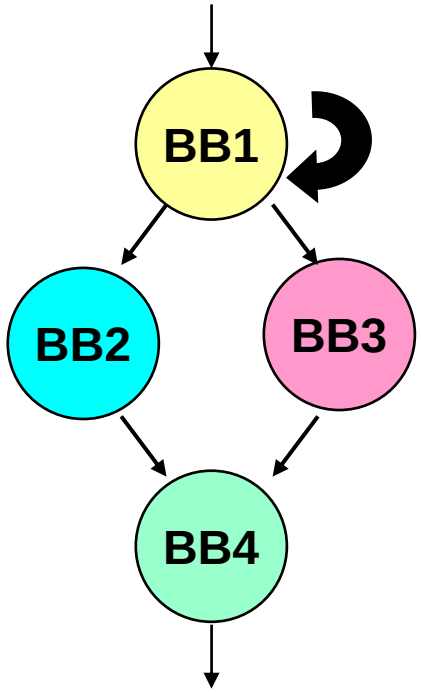
- **Para preencher o “delay slot”, ...**
- **Quando é legal mover uma instrução?**
  - **Otimização 1: Procurar nos predecessores**
    - » **Quando respeitadas dependências de dados**
  - **Otimização 2: Procurar nos sucessores**
    - » **Quando respeitadas dependências de dados**
  - **Otimização 3: Procurar no endereço-alvo**
    - » **Quando respeitadas dependências de dados**

“Delayed branch”

- **Para preencher o “delay slot”, ...**
- **Quando é eficiente mover uma instrução?**
  - **Otimização 1: Procurar nos predecessores**
    - » **Sempre vale a pena**
  - **Otimização 2: Procurar nos sucessores**
    - » **Vale a pena quando desvio não tomado**
  - **Otimização 3: Procurar no endereço-alvo**
    - » **Vale a pena quando desvio tomado**

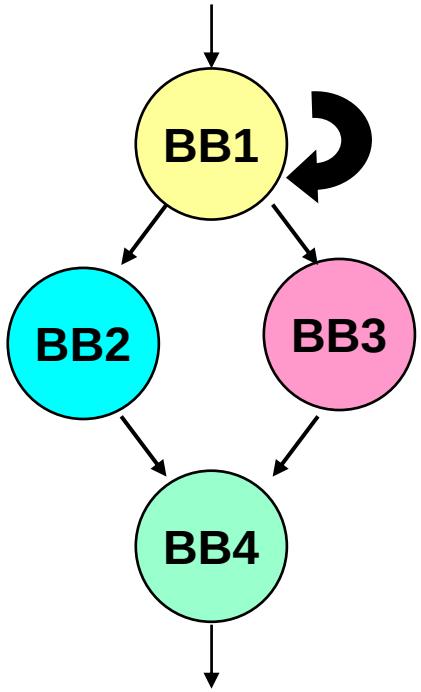
Otimização 1a: instrução anterior
(legal, sempre eficiente)

	...
	add $s1, $s2, $s3
	beqz $s2, else
	nop
then:	add $t1, $t2, $t3
	lw $t0, 0($t1)
	j exit
else:	add $t1, $t4, $t5
	lw $t0, 0($t1)
exit:	add $s1, $s1, $t0
	...



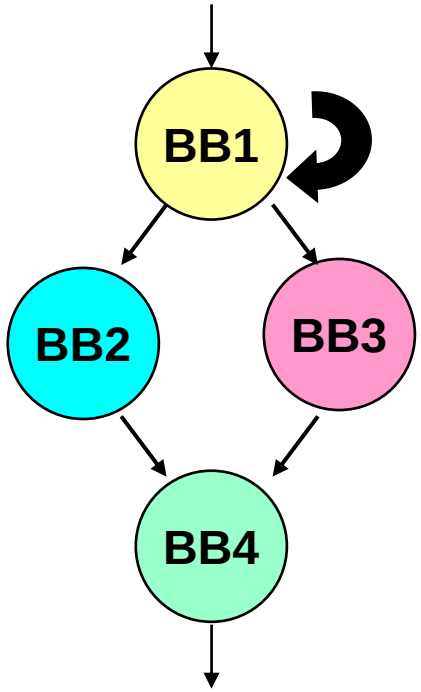
Otimização 1a: instrução anterior
(legal, sempre eficiente)

	...			...
	add $s1, $s2, $s3			beqz $s2, else
	beqz $s2, else			**add $s1, $s2, $s3**
	nop		then:	add $t1, $t2, $t3
then:	add $t1, $t2, $t3			lw $t0, 0($t1)
	lw $t0, 0($t1)			j exit
	j exit		else:	add $t1, $t4, $t5
else:	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...
	...			



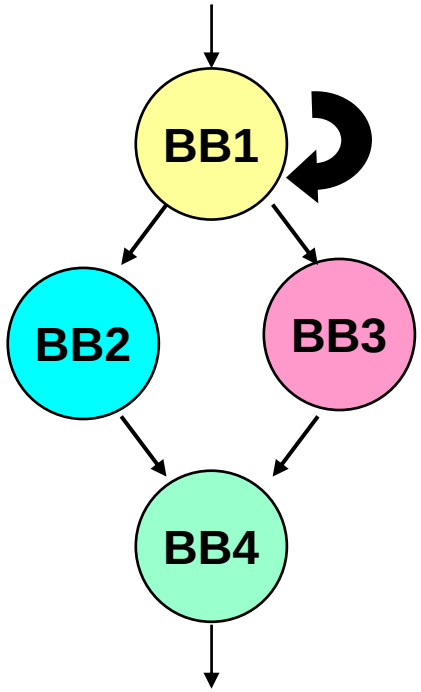
Otimização 1b: instrução anterior (ilegal)

	...
	add $s1, $s2, $s3
	beqz **$s1**, else
	nop
then:	add $t1, $t2, $t3
	lw $t0, 0($t1)
	j exit
else:	add $t1, $t4, $t5
	lw $t0, 0($t1)
exit:	add $s1, $s1, $t0
	...



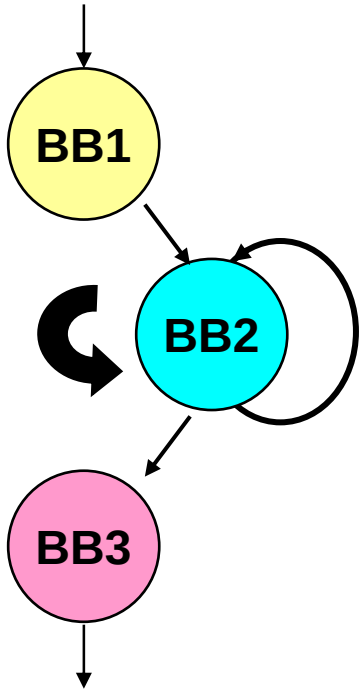
Otimização 1b: instrução anterior (ilegal)

	...			...
	add $s1, $s2, $s3			beqz $s1, else
	beqz $s1, else			**add $s1, $s2, $s3**
	nop		then:	add $t1, $t2, $t3
then:	add $t1, $t2, $t3			lw $t0, 0($t1)
	lw $t0, 0($t1)			j exit
	j exit		else:	add $t1, $t4, $t5
else:	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...
	...			



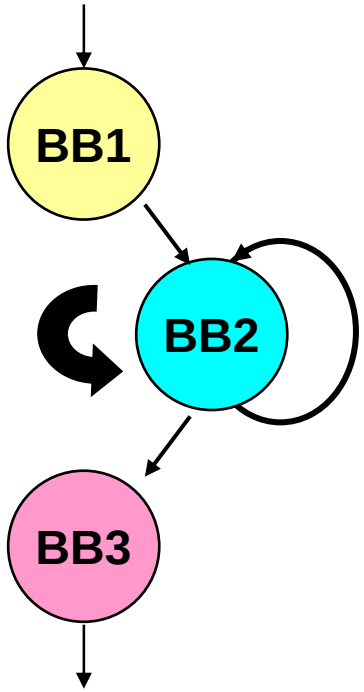
Otimização 1c: instrução anterior
(legal, sempre eficiente)

	...
	add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**
	add $s1, $s2, $s3
	beqz $s1, loop
	nop
	add $t1, $t4, $t5
	lw $t0, 0($t1)
	add $s4, $s1, $t0
	...



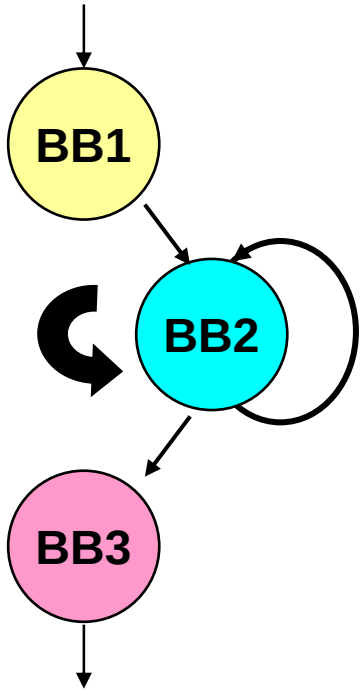
Otimização 1c: instrução anterior
(legal, sempre eficiente)

	...			...
	add $s6, $zero, $s0			add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**		loop:	add $s1, $s2, $s3
	add $s1, $s2, $s3			beqz $s1, loop
	beqz $s1, loop			**sub $s4, $s5, $s6**
	nop			add $t1, $t4, $t5
	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)			add $s4, $s1, $t0
	add $s4, $s1, $t0			...
	...			



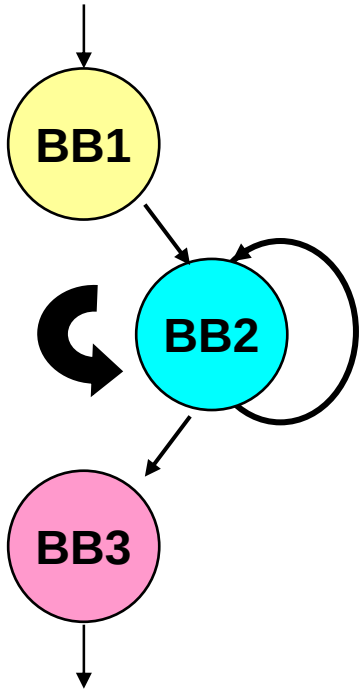
Otimização 1d: instrução anterior (ilegal)

	...
	add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**
	add $s1, $s2, $s3
	beqz $s4, loop
	nop
	add $t1, $t4, $t5
	lw $t0, 0($t1)
	add $s4, $s1, $t0
	...



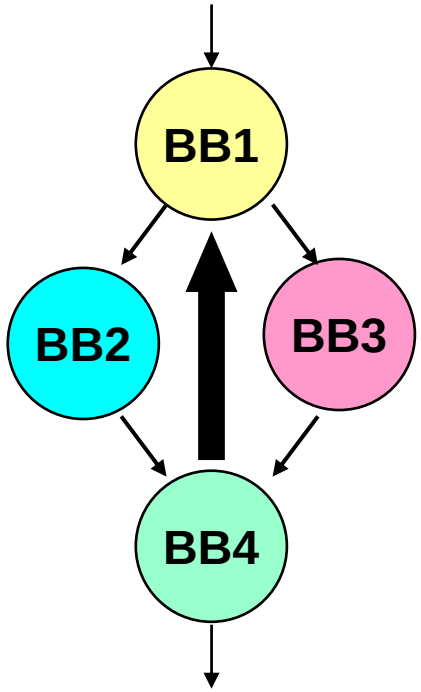
Otimização 1d: instrução anterior (ilegal)

	...			...
	add $s6, $zero, $s0			add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**		loop:	add $s1, $s2, $s3
	add $s1, $s2, $s3			beqz $s4, loop
	beqz $s4, loop			**sub $s4, $s5, $s6**
	nop			add $t1, $t4, $t5
	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)			add $s4, $s1, $t0
	add $s4, $s1, $t0			...
	...			



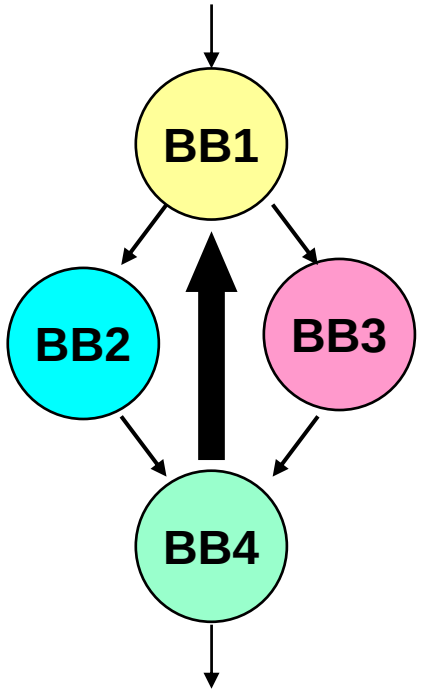
Otimização 2a: instrução posterior
(legal, sempre eficiente)

	...
	beqz $s2, else
	nop
then:	add $t1, $t2, $t3
	lw $t0, 0($t1)
	j exit
else:	add $t1, $t4, $t5
	lw $t0, 0($t1)
exit:	**add $s1, $s2, $s3**
	add $s1, $s1, $t0
	...



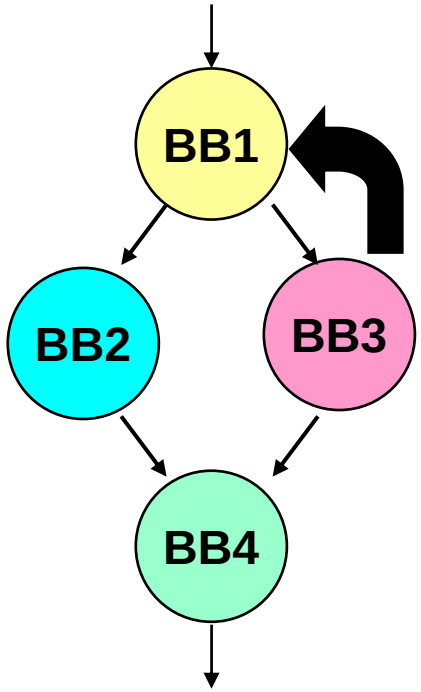
Otimização 2a: instrução posterior
(legal, sempre eficiente)

	...			...
	beqz $s2, else			beqz $s2, else
	nop			**add $s1, $s2, $s3**
then:	add $t1, $t2, $t3		then:	add $t1, $t2, $t3
	lw $t0, 0($t1)			lw $t0, 0($t1)
	j exit			j exit
else:	add $t1, $t4, $t5		else:	add $t1, $t4, $t5
	lw $t0, 0($t1)			lw $t0, 0($t1)
exit:	**add $s1, $s2, $s3**		exit:	add $s1, $s1, $t0
	add $s1, $s1, $t0			...
	...			



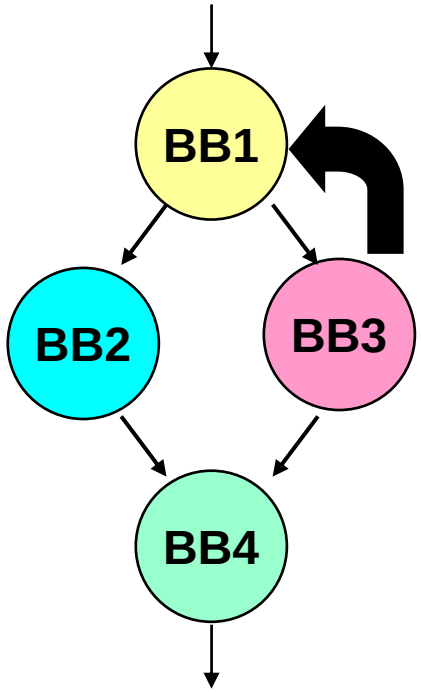
Otimização 2b: instrução default
(legal, eficiente quando desvio não tomado)

	...
	add $s1, $s2, $s3
	beqz $s1, else
	nop
then:	**sub $s4, $s5, $s6**
	add $t1, $t2, $s4
	lw $t0, 0($t1)
	j exit
else:	add $t1, $t4, $t5
	lw $t0, 0($t1)
exit:	add $s1, $s1, $t0
	...



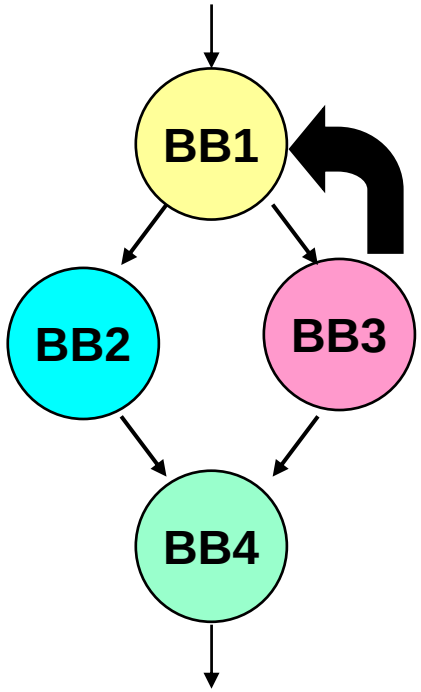
Otimização 2b: instrução default
(legal, eficiente quando desvio não tomado)

	...			...
	add $s1, $s2, $s3			add $s1, $s2, $s3
	beqz $s1, else			beqz $s1, else
	nop			**sub $s4, $s5, $s6**
then:	**sub $s4, $s5, $s6**		then:	add $t1, $t2, $s4
	add $t1, $t2, $s4			lw $t0, 0($t1)
	lw $t0, 0($t1)			j exit
	j exit		else:	add $t1, $t4, $t5
else:	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...
	...			



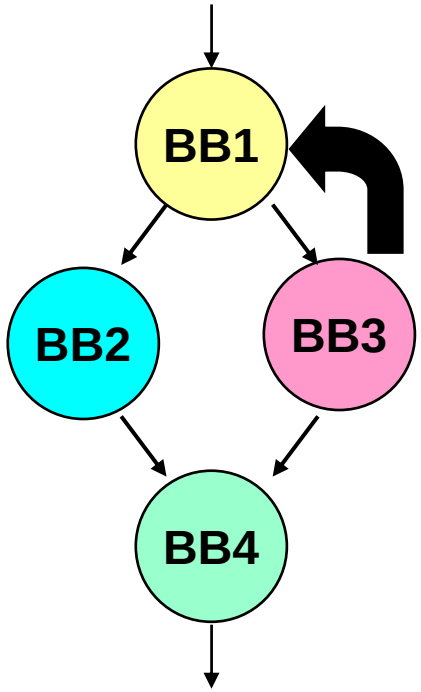
Otimização 2c: instrução default (ilegal)

	...
	add $s1, $s2, $s3
	beqz $s1, else
	nop
then:	**sub $s4, $s5, $s6**
	add $t1, $t2, $s4
	lw $t0, 0($t1)
	j exit
else:	add $t1, $t4, $s4
	lw $t0, 0($t1)
exit:	add $s1, $s1, $t0



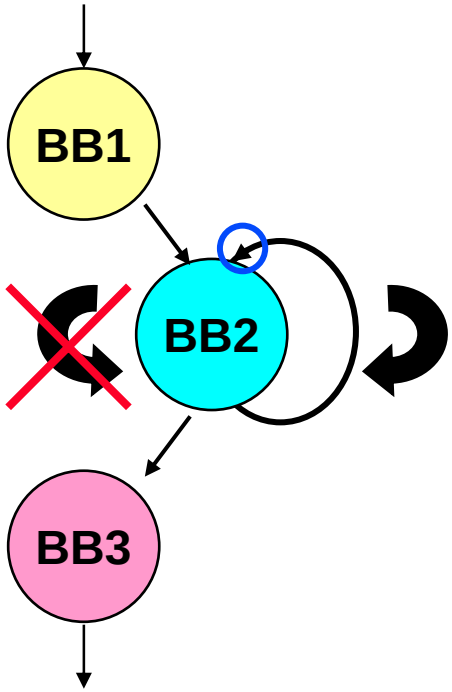
Otimização 2c: instrução default (ilegal)

	...			...
	add $s1, $s2, $s3			add $s1, $s2, $s3
	beqz $s1, else			beqz $s1, else
	nop			**sub $s4, $s5, $s6**
then:	**sub $s4, $s5, $s6**		then:	add $t1, $t2, $s4
	add $t1, $t2, $s4			lw $t0, 0($t1)
	lw $t0, 0($t1)			j exit
	j exit		else:	add $t1, $t4, $s4
else:	add $t1, $t4, $s4			lw $t0, 0($t1)
	lw $t0, 0($t1)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...



Otimização 1d: instrução anterior (ilegal)

	...			...
	add $s6, $zero, $s0			add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**		loop:	add $s1, $s2, $s3
	add $s1, $s2, $s3			beqz $s4, loop
	beqz $s4, loop			**sub $s4, $s5, $s6**
	nop			add $t1, $t4, $t5
	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)			add $s4, $s1, $t0
	add $s4, $s1, $t0			...
	...			

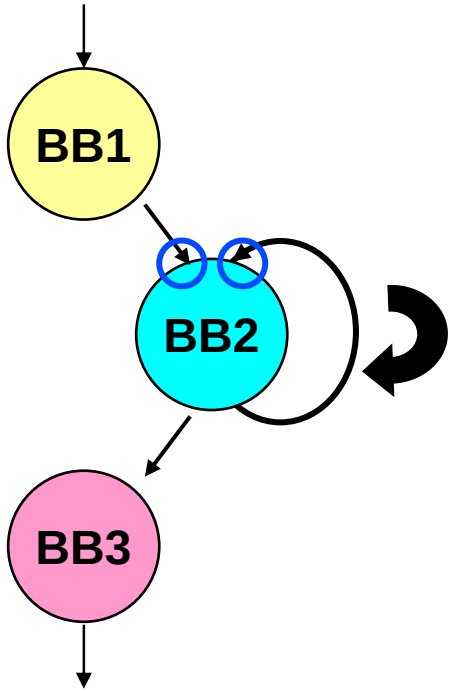


Antes de mostrar mais otimizações, vamos revisitar exemplo anterior...

Instrução para o slot pode ser buscada no endereço-alvo (próxima iteração do laço)

Otimização 3a: instrução-alvo (ilegal)

	...			...
	add $s6, $zero, $s0			add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**		loop:	add $s1, $s2, $s3
	add $s1, $s2, $s3			beqz $s4, loop
	beqz $s4, loop			**sub $s4, $s5, $s6**
	nop			add $t1, $t4, $t5
	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)			add $s4, $s1, $t0
	add $s4, $s1, $t0			...
	...			

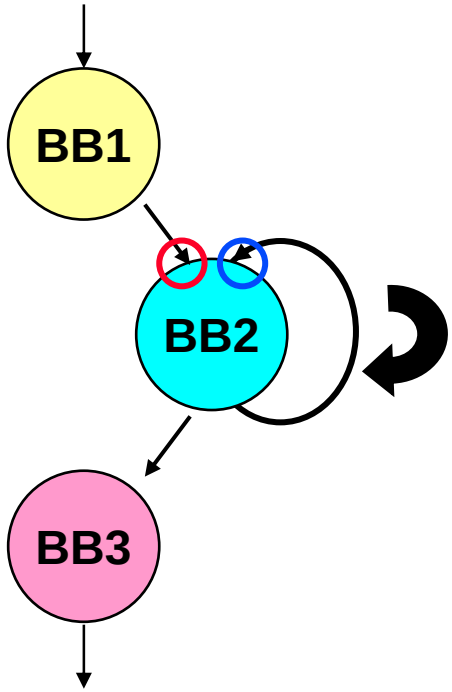


Como BB2 tem dois pontos de entrada, a execução de sub da primeira iteração seria removida.

Pode ser tornada legal inserindo código de compensação.

Otimização 3a: instrução-alvo
(legal, eficiente exceto à saída do laço)

	...			...
	add $s6, $zero, $s0			add $s6, $zero, $s0
loop:	**sub $s4, $s5, $s6**		loop:	add $s1, $s2, $s3
	add $s1, $s2, $s3			beqz $s4, loop
	beqz $s4, loop			**sub $s4, $s5, $s6**
	nop			add $t1, $t4, $t5
	add $t1, $t4, $t5			lw $t0, 0($t1)
	lw $t0, 0($t1)			add $s4, $s1, $t0
	add $s4, $s1, $t0			...
	...			

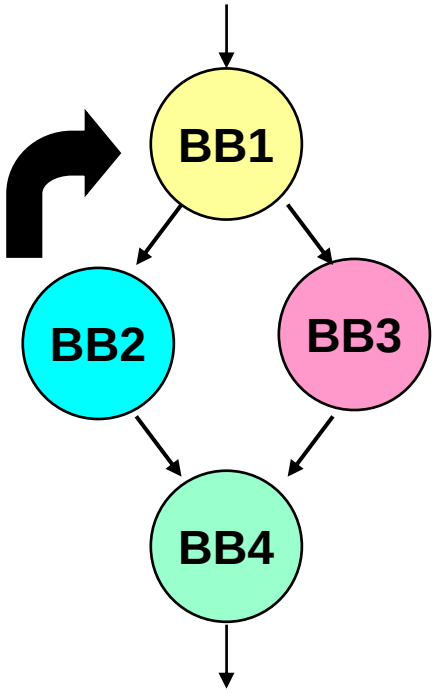


Instrução original é movida para o slot (em sentido contrário ao ciclo)

Uma cópia é inserida como código de compensação.

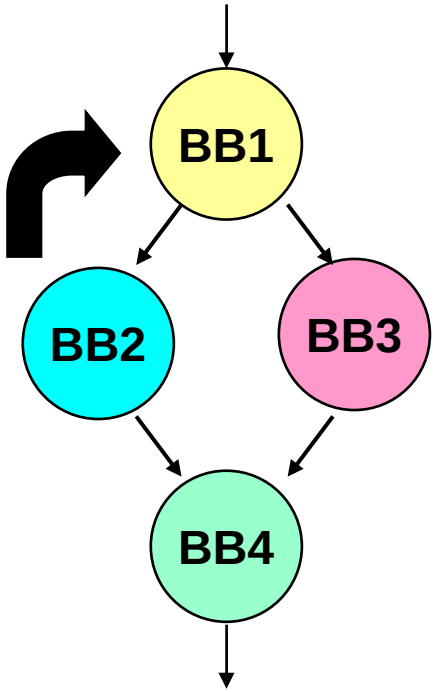
Otimização 3b: instrução-alvo
(legal, eficiente quando desvio tomado)

	...
	add $s1, $s2, $s3
	beqz $s1, else
	nop
then:	sub $s4, $s5, $s6
	add $t1, $t2, $t3
	lw $t0, 0($t1)
	j exit
else:	**add $t7, $t4, $t5**
	lw $t0, 0($t7)
exit:	add $s1, $s1, $t0
	...



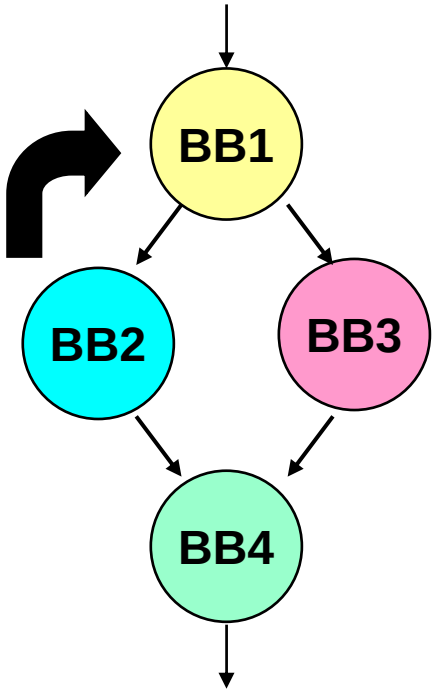
Otimização 3b: instrução-alvo
(legal, eficiente quando desvio tomado)

	...			...
	add $s1, $s2, $s3			add $s1, $s2, $s3
	beqz $s1, else			beqz $s1, else
	nop			**add $t7, $t4, $t5**
then:	sub $s4, $s5, $s6		then:	sub $s4, $s5, $s6
	add $t1, $t2, $t3			add $t1, $t2, $t3
	lw $t0, 0($t1)			lw $t0, 0($t1)
	j exit			j exit
else:	**add $t7, $t4, $t5**		else:	lw $t0, 0($t7)
	lw $t0, 0($t7)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...
	...			



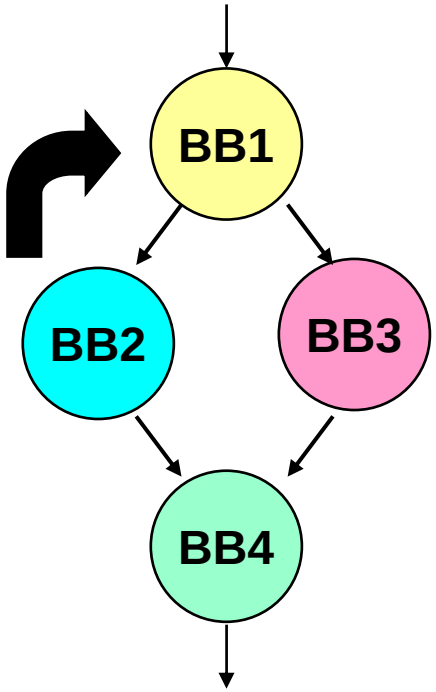
Otimização 3c: instrução-alvo (ilegal)

```
        ...
        add $s1, $s2, $s3
        beqz $s1, else
        nop
then:   sub $s4, $s5, $s6
        add $t1, $t2, $t3
        lw $t0, 0($t1)
        j exit
else:   add $t2, $t4, $t5
        lw $t0, 0($t2)
exit:   add $s1, $s1, $t0
        ...
```



Otimização 3c: instrução-alvo (ilegal)

	...			...
	add $s1, $s2, $s3			add $s1, $s2, $s3
	beqz $s1, else			beqz $s1, else
	nop			**add $t2, $t4, $t5**
then:	sub $s4, $s5, $s6		then:	sub $s4, $s5, $s6
	add $t1, **$t2**, $t3			add $t1, **$t2**, $t3
	lw $t0, 0($t1)			lw $t0, 0($t1)
	j exit			j exit
else:	**add $t2, $t4, $t5**		else:	lw $t0, 0($t1)
	lw $t0, 0($t2)		exit:	add $s1, $s1, $t0
exit:	add $s1, $s1, $t0			...
	...			



Impacto da otimização

- **Eficiência do compilador**
  - **Preenche cerca de 80% dos slots.**
  - **Cerca de 70% das instruções executadas úteis**
    - » **30% de perda**
    - » **Referência: Computer Architecture: A Quantitative Approach, 2nd edition, Figura 3.31, p. 171.**
- **E quando não se consegue preenchê-los?**
  - **O compilador inserte “nops”**
    - » **Ou o montador**

“Delayed branch”: limitações

- **Não vale a pena para pipelines longos**
  - **Pipelines com 7-8 estágios são comuns**
    - » **Para trabalhar em freqüência mais altas**
    - » **Difícil encontrar instruções para preencher slots.**
- **Aspecto da micro-arquitetura vísivel no ISA**
  - **Dificuldade de programação**
    - » **Em linguagem de montagem**
  - **Por isso montadores costumam escondê-lo**
    - » **Mas depois reorganizam o código “assembly”**
    - » **Antes de gerar o código objeto**

“Delayed branch”: uso

- **Usado em:**
  - **Alguns RISCs: MIPS, SPARC, PA-RISC, SH**
    - » **Delayed branch**
    - » **Annulling delayed branch**
  - **Alguns DSPs: TMS320C3x, SHARC**
- **Não usado em:**
  - **PowerPC, ARM**