Predição de Desvios e Processadores Superescalares Especulativos

CES-25 – Arquiteturas para Alto Desmpenho
Prof. Paulo André Castro
pauloac@ita.br
Sala 110 – Prédio da Computação
www.comp.ita.br/~pauloac
IEC - ITA

Tomasulo Especulativo

- Se os desvios tem dependência RAW com uma interação, ainda é possível a linearização de Loops sem execução especulativa ?
  - Não!!! É necessário a execução especulativa!
- Novas questões trazidas por execução superescalar
  - Mais Hardware (estações de reserva, unidades de execução, etc.)
  - Controle e detecção de dependência mais complexa
  - Memória adicional para garantir correção no caso de predição errônea.
  - **Necessidade de predição de desvios mais eficiente para pagar custo do novo hardware com desempenho!!!!**

Dependências de Controle

- A medida que aumenta o número de instruções executadas por clock (diminui CPI) aumenta o fluxo potencial de instruções
  - Logo, os atrasos causados por desvios podem se tornar também muito sérios

- Como minimizar o problema?
  - Simples! Não parar ou desacelerar em desvios

Prevendo o Futuro….

- Predição Estática
  - Prever sempre Seguir
  - Prever sempre Não Seguir
  - Qual a melhor opção ?
- Predição Dinâmica
  - Desvios condicionais podem ser executados vários vezes em programa
    - Ex.: Loops

Predição Dinâmica de Desvio

- Mais instruções em paralelo significam maior impacto por atrasos de desvios condicionais

- Abordagem de prever sempre não seguir do desvio condicional pode ser ineficiente

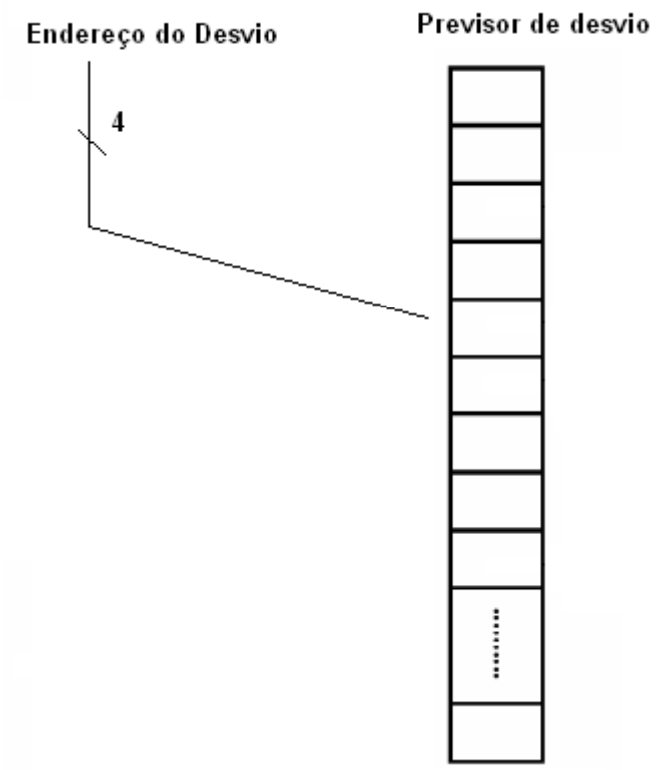
```
        addi R1,R1,0
        addi R4,R3,5000
loop:
        load R2,0(R3)
        addi R3,R3,1
        add R1,R1,R2
        bne R3,R4,#loop
```

Predição Dinâmica de Desvio

- Solução ?
- Associar um bit a cada desvio condicional
  - Se seguido na última passagem, prever Seguir
  - Se não seguido na última passagem prever não seguido
- Ainda é preciso calcular o endereço de destino do desvio condicional

Buffer de Previsão de Desvio

- Associado ao endereço (ou apenas bits mais baixos) de cada desvio há um previsor de desvio
- Ao encontrar uma entrada segue-se a previsão dada. Se erra atualiza-se a previsão
- Porque utilizar apenas bits mais baixos?
- Algum problema em utilizar apenas bits mais baixos?



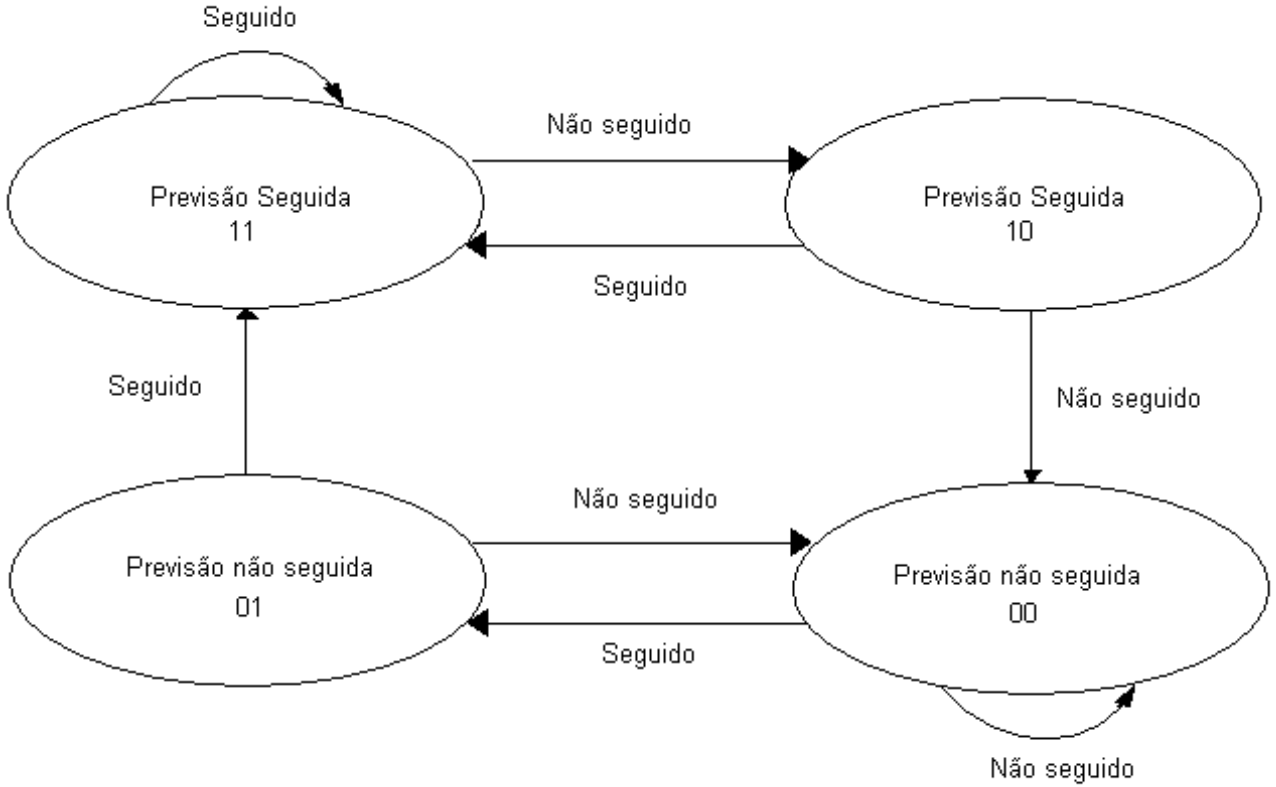
Predição Dinâmica de Desvio

- Possível Erro duplicado no laço interno em relação a predição sempre seguir se for utilizado a predição de 1 bit

```
do {
   do {
     sum+=v[i][j++];
   } while(j<500);//no inicio S, ao final NS
}while(i<500);
```

- Solução ?
- Poderia ser melhorado utilizando predição com 2 ou mais bits.

Predição dinâmica de Desvios
2 bits



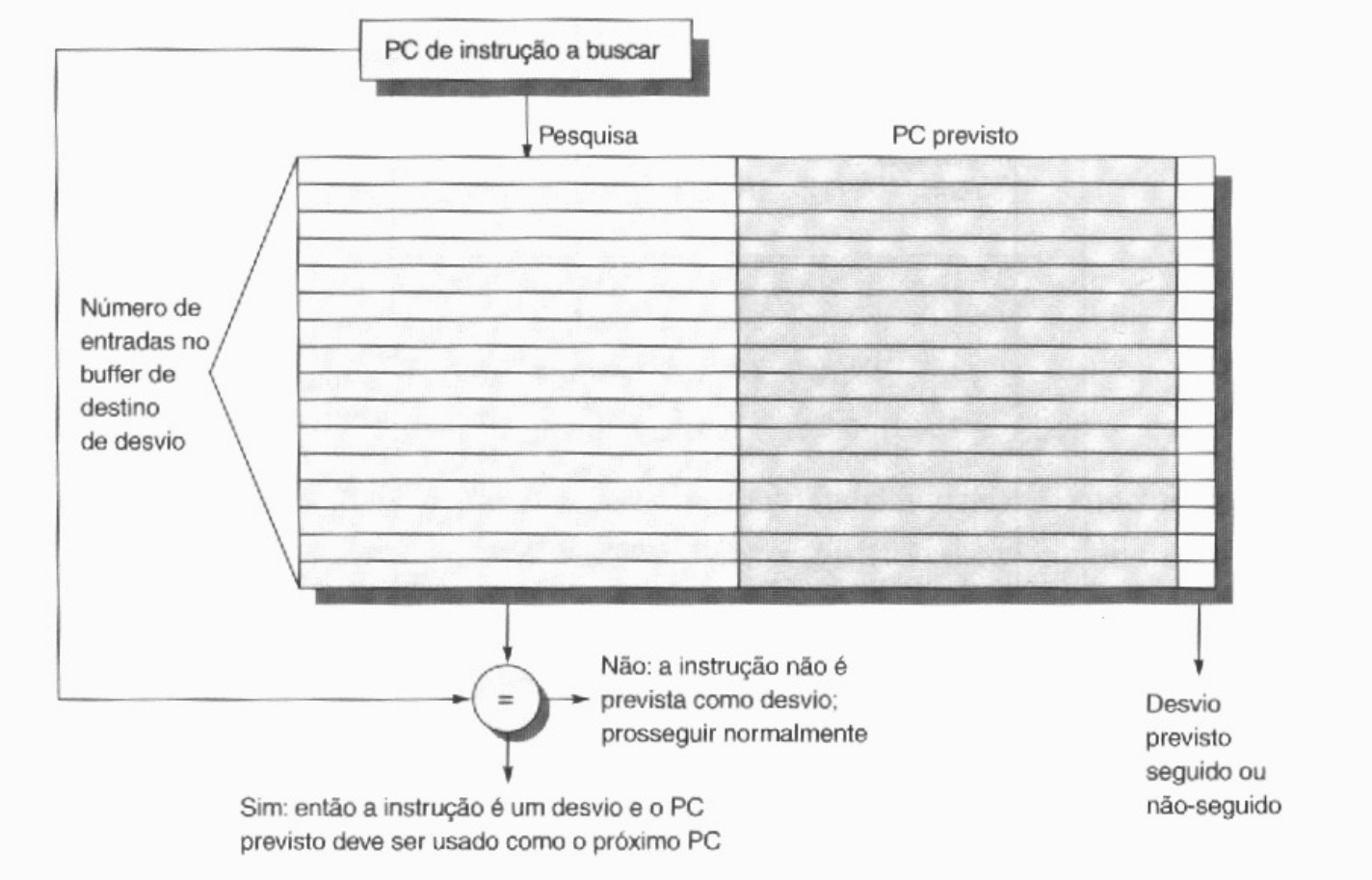
Predição de Desvio

- Permite determinar qual a próxima instrução antes de concluir a instrução, a qual pode ser atrasada por dependências
- Necessário decodificar a instrução para calcular o endereço de destino (fase ID), o que causa atraso, se o desvio for seguido.
- Seria possível determinar o destino (endereço) em IF (solução ideal) ?

Predição Dinâmica de Desvio com Buffer de Destino de Desvio

- Utiliza-se um buffer de destino de desvio, onde cada entrada tem
  - Bit de predição
  - PC de destino
  - PC do Desvio (ou parte deste para diminuir tamanho da tabela)
  - O buffer deve ser interno ao processador

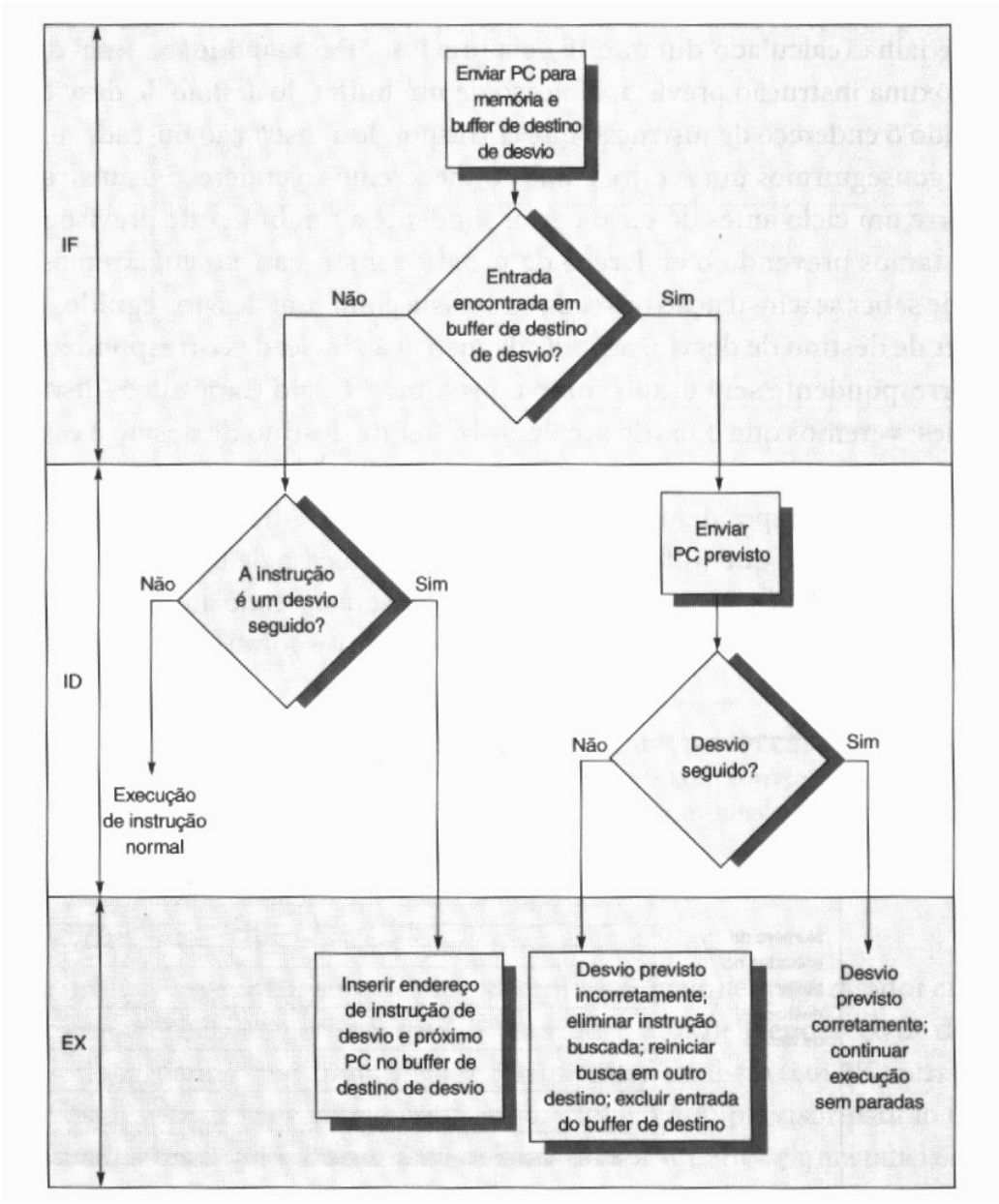
Buffer de Destino de Desvio



Buffer de Destino de Desvio

- Também chamado de Cache de Destino de Desvio
- O buffer pode determinar o próximo PC previsto ao final da fase de IF, pois utiliza o endereço da instrução de desvio como índice
- Sem o buffer seria possível determinar o próximo PC previsto apenas após a fase de ID
- Insere-se no buffer apenas desvios que são seguidos

Etapas envolvidas no tratamento de instrução com buffer de destino de desvio



Desvios Correlacionados

- Um desvio anterior pode determinar o comportamento do desvio seguinte

```
D1: if(d==0)
        d=1;
D2: if(d==1)
        .....
```

- Pode-se criar em D2 uma previsão para o caso de D1 ser seguido e outra previsão caso contrário

Desvios Correlacionados

- Quando se utiliza o resultado de um desvio para selecionar qual de dois previsores de 1 bit utilizar, dá-se a este previsor o nome de previsor (1,1)

- No caso geral (m,n), utiliza-se os m desvios anteriores para selecionar 2^m previsores, cada um com n bits

- Um previsor comum é (2,2). Utilizado por exemplo em alguns computadores Alpha

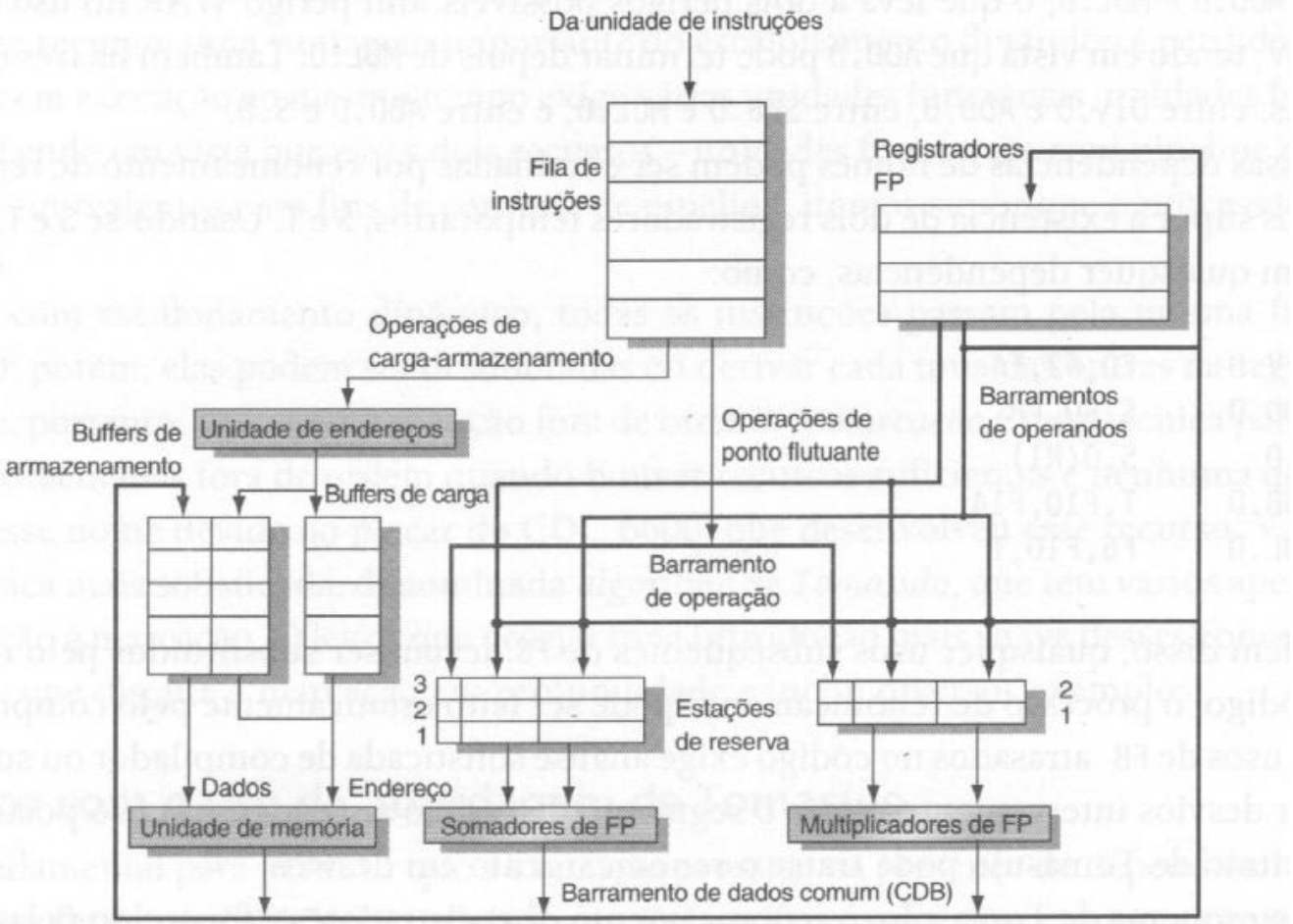
Sumário

- Predições eficientes de desvios são mais importantes a medida que aumenta o paralelismo de nível de instrução (ILP) (mais instruções executadas simultaneamente)
- Predição estática não fornece boas taxas de acerto
- Predição dinâmica é utilizada com base :
  - Nas passagens anteriores
  - No co-relacionamento com desvios anteriores
- Deve-se utilizar um buffer de destino de desvio para otimizar a determinação do endereço previsto para o desvio seguido.

Sumário 2

- Como garantir a execução corretas nos casos de falha de predição?
- Evitando que as instruções alterem de modo irreversível o comportamento até que se tenha certeza da sua execução. (execução especulativa)
- Especulação baseada em hardware é utilizada por vários processadores modernos como:
  - PowerPC 604, 604, G3,G4
  - MIPS R10000, R12000
  - Alpha 21264
  - Pentium II/ III/ 4
  - AMD K5/K6/Athlon

Tomasulo


Da unidade de instruções
Fila de instruções
Registradores FP
Operações de carga-armazenamento
Buffers de armazenamento
Unidade de endereços
Buffers de carga
Operações de ponto flutuante
Barramentos de operandos
Barramento de operação
3
2
1
Estações de reserva
2
1
Dados
Endereço
Unidade de memória
Somadores de FP
Multiplicadores de FP
Barramento de dados comum (CDB)

Incorporando a Execução Especulativa

- Por melhor que seja a predição de desvios, certamente irá falhar algumas vezes

- Em processadores superescalares, muitas instruções podem ser executadas e gravadas antes de um desvio condicional ser resolvido. Certo ou Errado? Por quê?
- Exemplo:
  - LW R1, 0 (R3) ; suponha que MEM[0+R3] não está em cache
  - BEQ R1,R2, P1
  - ADD R4,R6,R8

  P1:
  - SUB R3,R8,R4
  - .....

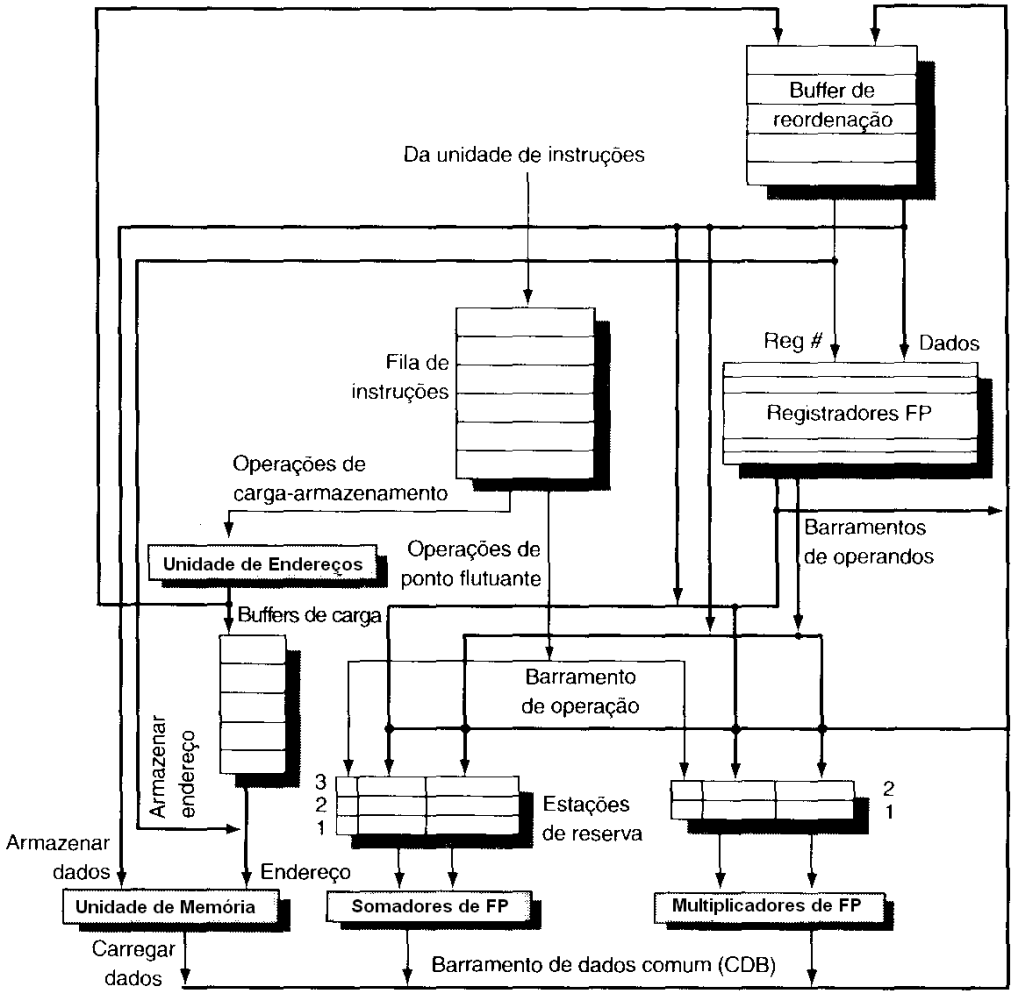
Alterações em Tomasulo

- Deve ser possível reverter as gravações feitas por todas as instruções realizadas após a emissão de um desvio condicional cuja predição falhou. Como fazer?

- Introdução de nova fase e memória adicional
  - Emissão, Execução, Gravação e **Consolidação**

- Memória Adicional
  - Buffer de Reordenação

Processadores Superescalares Especulativos
AlgoritmoTomasulo Especulativo



Exemplo 1 - Superescalar Especulativo

```
L.D F6,34(R2)
L.D F2,45(R3)
MUL.D F0,F2,F4
SUB.D F8,F6,F2
DIV.D F10,F0,F6
ADD.D F6,F8,F2
```

Qual a situação quando a operação MUL.D está pronta para ser consolidada?

Estações de reserva

Nome	Ocupado	Op	Vj	Vk	Qj	Qk	Dest	A
Load1	não							
Load2	não							
Add1	não							
Add2	não							
Add3	não							
Mult1	não	MUL.D	Mem[45 + Regs[R3]]	Regs[F4]			#3	
Mult2	sim	DIV.D		Mem[34 + Regs[R2]]	#3		#5	

Buffer de reordenação

Entrada	Ocupado	Instrução		Estado	Destino	Valor
1	não	L.D	F6,34(R2)	Consolidar	F6	Mem[34 + Regs[R2]]
2	não	L.D	F2,45(R3)	Consolidar	F2	Mem[45 + Regs[R3]]
3	sim	MUL.D	F0,F2,F4	Gravar resultado	F0	#2 × Regs[F4]
4	sim	SUB.D	F8,F6,F2	Gravar resultado	F8	#1 – #2
5	sim	DIV.D	F10,F0,F6	Executar	F10	
6	sim	ADD.D	F6,F8,F2	Gravar resultado	F6	#4 + #2

Status de registradores de FP

Campo	F0	F1	F2	F3	F4	F5	F6	F7	F8	F10
Reordenação #	3						6		4	5
Ocupado	sim	não	não	não	não	não	sim	...	sim	sim

Fonte: Hennessy & Patterson, pg. 168. Erro no Livro: Est. Res. Mult 2: Dest = #3 e A=#5

Exemplo 2 - Superescalar Especulativo

```
LOOP:
L.D F0,0(R1)
MUL.D F4,F0,F2
S.D F4,0(R1)
DADDIU R1,R1,#-8
BNE R1,R2,LOOP
```

Qual a situação após duas execuções completas do laço e as duas primeiras instruções consolidadas?

Exemplo 2

	Buffer de reordenação					
Entrada	**Ocupado**	**Instrução**		**Estado**	**Destino**	**Valor**
1	não	L.D	F0,0(R1)	Consolidar	F0	Mem[0 + Regs[R1]]
2	não	MUL.D	F4,F0,F2	Consolidar	F4	#1 × Regs[F2]
3	sim	S.D	F4,0(R1)	Gravar resultado	0 + Regs[R1]	#2
4	sim	DADDIU	R1,R1,#-8	Gravar resultado	R1	Regs[R1] – 8
5	sim	BNE	R1,R2,Loop	Gravar resultado		
6	sim	L.D	F0,0(R1)	Gravar resultado	F0	Mem[#4]
7	sim	MUL.D	F4,F0,F2	Gravar resultado	F4	#6 × Regs[F2]
8	sim	S.D	F4,0(R1)	Gravar resultado	0 + #4	#7
9	sim	DADDIU	R1,R1,#-8	Gravar resultado	R1	#4 – 8
10	sim	BNE	R1,R2,Loop	Gravar resultado		

	Status de registradores de FP								
Campo	**F0**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**
Reordenação #	6				7				
Ocupado	sim	não	não	não	sim	não	não	. . .	não

Status	Esperar até	Ação ou contabilidade
Emitir todas as instruções	Estação de reserva (r) e ROB (b) disponíveis	`if (RegisterStat[rs].Busy)/* instr. grava rs durante execução */ {h ← RegisterStat[rs].Reorder; if (ROB[h].Ready)/* Instr. já concluída */ {RS[r].Vj ← ROB[h].Value; RS[r].Qj ← 0;} else {RS[r].Qj ← h;} /* espera por instrução */ } else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0;}; RS[r].Busy ← yes; RS[r].Dest ← b; ROB[b].Instruction ← opcode; ROB[b].Dest ← rd;ROB[b].Ready ← no;`
Operações de FP e armazenamentos		`if (RegisterStat[rt].Busy) /* instr. grava rt durante execução */ {h ← RegisterStat[rt].Reorder; if (ROB[b].Ready) /* Instr. já concluída */ {RS[r].Vk ← ROB[h].Value; RS[r].Qk ← 0;} else {RS[r].Qk ← h;} /* Espera por instrução */ } else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0;};`
Operações de FP		`RegisterStat[rd].Qi=b; RegisterStat[rd].Busy ← yes; ROB[b].Dest ← rd;`
Cargas		`RS[r].A ← imm; RegisterStat[rt].Qi=b; RegisterStat[rt].Busy ← yes; ROB[b].Dest ← rt;`
Armazenamentos		`RS[r].A ← imm;`
Executar operação de FP	(Rs[r].Qj = 0) e (Rs[r].Qj = 0)	Calcular resultados – operandos estão em Vj e Vk
Load etapa 1	(Rs[r].Qj = 0) e não existe nenhum armazenamento anterior na fila	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load etapa 2	Etapa 1 de Load concluída e todos os armazenamentos anteriores em ROB têm endereço diferente	Ler de Mem[Rs[r].A]
Store	(Rs[r].Qj = 0) e armazenamento no início da fila	`ROB[h].Address ← RS[r].Vj + RS[r].A;`
Gravar todos os resultados exceto armazenamento	Execução feita em *r* e CDB disponível.	`b ← RS[r].Reorder; RS[r].Busy ← no; ∀x(if (RS[x].Qj=b) {RS[x].Vj ← result; RS[x].Qj ← 0}); ∀x(if (RS[x].Qk=b) {RS[x].Vk ← result; RS[x].Qk ← 0}); ROB[b].Value ← result; ROB[b].Ready ← yes;`

Status	Esperar até	Ação ou contabilidade

Emitir todas as intruções	Estação de reserva (r) e ROB (b) disponíveis	`if (RegisterStat[rs].Busy)/* instr. grava rs durante execução */`
		`{h ← RegisterStat[rs].Reorder;`
		`if (ROB[h].Ready)/* Instr. já concluída */`
		`{RS[r].Vj ← ROB[h].Value; RS[r].Qj ← 0;}`
		`else {RS[r].Qj ← h;} /* espera por instrução */`
		`} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0;};`
		`RS[r].Busy ← yes; RS[r].Dest ← b;`
		`ROB[b].Instruction ← opcode; ROB[b].Dest ← rd;ROB[b].Ready ← no;`

Operações de FP e armazenamentos		`if (RegisterStat[rt].Busy) /* instr. grava rt durante execução */`
		`{h ← RegisterStat[rt].Reorder;`
		`if (ROB[b].Ready) /* Instr. já concluída */`
		`{RS[r].Vk ← ROB[h].Value; RS[r].Qk ← 0;}`
		`else {RS[r].Qk ← h;} /* Espera por instrução */`
		`} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0;};`

Operações de FP		`RegisterStat[rd].Qi=b; RegisterStat[rd].Busy ← yes;` `ROB[b].Dest ← rd;`
Caregamentos		`RS[r].A ← imm; RegisterStat[rt].Qi=b;` `RegisterStat[rt].Busy ← yes; ROB[b].Dest ← rt;`
Armazenamentos		`RS[r].A ← imm;`
Executar Op. de FP	(Rs[r].Qj = 0) e (Rs[r].Qj = 0)	Calcular resultados – operandos estão em Vj e Vk
Load etapa 1	(Rs[r].Qj = 0) e não existe nenhum armazenamento anterior na fila	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load etapa 2	Etapa 1 de Load concluída e todos os armazenamentos anteriores em ROB têm endereço diferente	Ler de Mem[Rs[r].A]
Store	(Rs[r].Qj = 0) e armazenamento no início da fila	`ROB[h].Address ← RS[r].Vj + RS[r].A;`

Gravar, exceto store	Execução feita em *r* e CDB disponível.	`b ← RS[r].Reorder; RS[r].Busy ← no;`
		`∀x(if (RS[x].Qj=b) {RS[x].Vj ← result; RS[x].Qj ← 0});`
		`∀x(if (RS[x].Qk=b) {RS[x].Vk ← result; RS[x].Qk ← 0});`
		`ROB[b].Value ← result; ROB[b].Ready ← yes;`

Nova fase: Consolidação

Store	Execução feita em *r* e (Rs[r].Qk = 0)	`ROB[h].Value ← RS[r].Vk;`

Consolidação	A instrução está no início do ROB (entrada h) e ROB[h].ready = yes	`d = ROB[h].Dest; /* dest. de registrador, se existir */`
		`if (ROB[h].Instruction==Branch)`
		`{if (branch is mispredicted)`
		`{clear ROB[h], RegisterStat; fetch branch dest;};}`
		`else if (ROB[h].Instruction==Store)`
		`{Mem[ROB[h].Address] ← ROB[h].Value;}`
		`else /* insere o resultado no destino de registrador */`
		`{Regs[d]← ROB[h].Value;};`
		`ROB[h].Busy ← no; /* libera entrada de ROB *`
		`/* libera registrador de dest. se nenhuma outra instr. estiver gravando nele */`
		`if (RegisterStat[d].Qi==h) {RegisterStat[d].Busy ← no;};`

Nova fase: Consolidação

Status	Esperar até	Ação ou contabilidade
Store	Execução feita em *r* e (Rs[r].Qk = 0)	ROB[h].Value ← RS[r].Vk;

Consolidação	A instrução está no início do ROB (entrada h) e ROB[h].ready = yes	d = ROB[h].Dest; /* dest. de registrador, se existir */
		if (ROB[h].Instruction==Branch)
		{if (branch is mispredicted)
		{clear ROB[h], RegisterStat; fetch branch dest;};}
		else if (ROB[h].Instruction==Store)
		{Mem[ROB[h].Address] ← ROB[h].Value;}
		else /* insere o resultado no destino de registrador */
		{Regs[d]← ROB[h].Value;};
		ROB[h].Busy ← no; /* libera entrada de ROB *
		/* libera registrador de dest. se nenhuma outra instr. estiver gravando nele */
		if (RegisterStat[d].Qi==h) {RegisterStat[d].Busy ← no;};

Emissão Múltipla em Processadores Superescalares

- Execução fora de ordem e especulativa ajudam a fazer o CPI aproximar-se de 1, mas não a diminuir abaixo deste valor.
- Para isto, é necessário emitir várias instruções simultaneamente. Como?
- É possível fazer a emissão de várias instruções em um mesmo clock resolvendo os seguintes problemas :
  - Emissão em paralelo mas marcação das dependências entre instruções em sequência (decodificação)
    - É preciso duplicar (ou multiplicar) as unidades de carregamento e decodificação de instruções
  - Uso do CDB
    - É preciso duplicar (ou multiplicar) a capacidade do CDB

Exemplo

- Vejamos, o impacto da especulação em Processador Superescalar com Emissão Múltipla.
- Exemplo (Hennessy, 3e., p. 172-173, 2003):
- LOOP:
  - LD R2, 0 (R1)
  - DADDIU R2,R2, 1
  - SD R2, 0 (R1)
  - DADDIU R1,R1, 4
  - BNE R2, R3, LOOP
- Comentário: Não se preocupem com o que faz o código, apenas com o tempo necessário para executá-lo.

Emissão Múltipla sem Especulação

Número da iteração	Instruções		Emite em ciclo de clock número	Executa em ciclo de clock número	Acesso de memória em ciclo de clock número	Grava CDB em ciclo de clock número	Comentário
1	LD	R2,0(R1)	1	2	3	4	Primeira emissão
1	DADDIU	R2,R2,#1	1	5		6	Esperar por LW
1	SD	R2,0(R1)	2	3	7		Esperar por DADDIU
1	DADDIU	R1,R1,#4	2	3		4	Executar diretamente
1	BNE	R2,R3,LOOP	3	7			Esperar por DADDIU
2	LD	R2,0(R1)	4	8	9	10	Esperar por BNE
2	DADDIU	R2,R2,#1	4	11		12	Esperar por LW
2	SD	R2,0(R1)	5	9	13		Esperar por DADDIU
2	DADDIU	R1,R1,#4	5	8		9	Esperar por BNE
2	BNE	R2,R3,LOOP	6	13			Esperar por DADDIU
3	LD	R2,0(R1)	7	14	15	16	Esperar por BNE
3	DADDIU	R2,R2,#1	7	17		18	Esperar por LW
3	SD	R2,0(R1)	8	15	19		Esperar por DADDIU
3	DADDIU	R1,R1,#4	8	14		15	Esperar por BNE
3	BNE	R2,R3,LOOP	9	19			Esperar por DADDIU

Emissão Múltipla com Especulação

Número da iteração	Instruções		Emite em ciclo de clock número	Executa em ciclo de clock número	Acesso de leitura em ciclo de clock número	Grava CDB em ciclo de clock número	Consolida em ciclo de clock número	Comentário
1	LD	R2,0(R1)	1	2	3	4	5	Primeira emissão
1	DADDIU	R2,R2,#1	1	5		6	7	Esperar por LW
1	SD	R2,0(R1)	2	3			7	Esperar por DADDIU
1	DADDIU	R1,R1,#4	2	3		4	8	Consolidar em ordem
1	BNE	R2,R3,LOOP	3	7			8	Esperar por DADDIU
2	LD	R2,0(R1)	4	5	6	7	9	Sem retardo de execução
2	DADDIU	R2,R2,#1	4	8		9	10	Esperar por LW
2	SD	R2,0(R1)	5	6			10	Esperar por DADDIU
2	DADDIU	R1,R1,#4	5	6		7	11	Consolidar em ordem
2	BNE	R2,R3,LOOP	6	10			11	Esperar por DADDIU
3	LD	R2,0(R1)	7	8	9	10	12	O mais cedo possível
3	DADDIU	R2,R2,#1	7	11		12	13	Esperar por LW
3	SD	R2,0(R1)	8	9			13	Esperar por DADDIU
3	DADDIU	R1,R1,#4	8	9		10	14	Executada mais cedo
3	BNE	R2,R3,LOOP	9	13			14	Esperar por DADDIU

Opções ao algoritmo de Tomasulo

- **Técnicas de SW** para aumentar o paralelismo em nível de instrução
  - **Reorganização** de instruções para reduzir atrasos
  - **Linearização** de Laços (“loop unrolling”)
- Very Long Instruction Word
  - As instruções são “empacotadas” em grupos de instruções independentes
  - Intel: EPIC (Explicitly parallel instruction computing)

Exemplo sem reorganizar: O que acontece?

Instrução que produz resultado	Instrução que utiliza resultado	Latência em ciclos de clock
Op de FP da ULA	Outra op de FP da ULA	3
Op de FP da ULA	Armazenamento duplo	2
Carga dupla	Op de FP da ULA	1
Carga dupla	Armazenamento duplo	0

```
                                        Ciclo de clock de emissão
Loop:   L.D       F0,0(R1)                      1
        parada                                  2
        ADD.D     F4,F0,F2                      3
        parada                                  4
        parada                                  5
        S.D       F4,0(R1)                      6
        DADDUI    R1,R1,#-8                     7
        parada                                  8
        BNE       R1,R2,Loop                    9
        parada                                 10
```

Reorganizando as instruções

Esse código exige 10 ciclos de clock por iteração. Podemos fazer a alteração de ordem das instruções do loop para obter apenas uma parada:

```
Loop:   L.D       F0,0(R1)
        DADDUI    R1,R1,#-8
        ADD.D     F4,F0,F2
        parada
        BNE       R1,R2,Loop    ;desvio retardado
        S.D       F4,8(R1)      ;alterado e intercambiado com DADDUI
```

O tempo de execução foi reduzido de 10 ciclos dc clock para 6. A parada depois de ADD.D destina-se ao uso de F4 pela instrução S. D.

Linearização de Loops em Software (pelo compilador)

- Exemplo:
  - for(i=1000;i>0;i--)
    - x[i]=x[i]+s;

Em Assemby MIPS:

```
Loop:  ;// R1, R2 inicio e final do array, F2 = S
L.D  F0,0(R1)        ;//F0 elemento de array
ADD.D F4,F0,F2  ;// soma escalar em F2
S.D F4,0(R1)        ;// armazena resultado
DADDDUI R1, R1,#-8 ;// decrementa ponteiro
BNE R1,R2 LOOP
```

Original:

Loop:

L.D F0,0(R1)

ADD.D F4,F0,F2

S.D F4,0(R1)

DADDDUI R1, R1,#-8

BNE R1,R2 LOOP

Linearizando:

```
Loop:
        L.D         F0,0(R1)
        ADD.D       F4,F0,F2
        S.D         F4,0(R1)        ;descarta DADDUI & BNE
        L.D         F6,-8(R1)
        ADD.D       F8,F6,F2
        S.D         F8,-8(R1)       ;descarta DADDUI & BNE
        L.D         F10,-16(R1)
        ADD.D       F12,F10,F2
        S.D         F12,-16(R1)     ;descarta DADDUI & BNE
        L.D         F14,-24(R1)
        ADD.D       F16,F14,F2
        S.D         F16,-24(R1)
        DADDUI      R1,R1,#-32
        BNE         R1,R2,Loop
```

Resumo do Exemplo de Linearização de Laços

1. Eliminar perigos WAR e WAW pela renomeação explícita de registradores

2. Mudar ordens de execução de instruções para diminuir atrasos

3. Determinar que era válido mover S D para depois dc DADDUI e BNE, e então encontrar a quantidade para ajustar o deslocamento de S D.

4. Linearização de laços:
   1. Identificar paralelismo entre iterações do laço
   2. Eliminar as instruções de teste e de desvio extras, e ajustar o código de término e de iteração do loop.

Resumo: Paralelismo em Nível de Instrução (ILP)

- **Execução Fora de Ordem**: executar instruções fora da ordem de emissão pode aumentar a eficiência do processador e evitar ociosidade de unidades funcionais
  - Algoritmo de Tomasulo
  - Soluções Estáticas (Compilador: VLIW, EPIC)
  - Tratamento de Perigos WAW e WAR
- **Predição de Desvio**
  - Predição Estática
  - Predição Dinâmica (n bits)
  - Predição Dinâmica com correlacionamento com m desvios anteriores. Preditor (m,n)
  - Buffer de Destino de Desvio

Resumo: Paralelismo em Nível de Instrução (ILP)

- Execução Especulativa: executar instruções mesmo antes de determinar o resultado de um desvio pode melhorar o desempenho do processamento desde que tenha uma boa taxa de acerto da predição

- Emissão Múltipla: carregar e decodificar instruções (e gravar) em paralelo pode permitir que o CPI caia abaixo do limite teórico de 1 CPI.

Resumo

Tipo	Entrada Execução	Det. Perigos	Escalon.	Caract.	Exemplos
Superescalar	Dinâmica	HW	Dinâmica	Exec. Fora de Ordem	IBM PowerPC
Superescalar Especulativo	Dinâmica	HW	Dinâmica Especulativa	Exec. Fora de Ordem com Esp.	PIII/P4 MIPS R10000 Alpha 21264
VLIW	Estática	SW	Estática	Sem perigos intra pacote	Trimedia, i860, Itanium

Status	Esperar até	Ação ou contabilidade
Emitir todas as instruções	Estação de reserva (r) e ROB (b) disponíveis	`if (RegisterStat[rs].Busy)/* instr. grava rs durante execução */ {h ← RegisterStat[rs].Reorder; if (ROB[h].Ready)/* Instr. já concluída */ {RS[r].Vj ← ROB[h].Value; RS[r].Qj ← 0;} else {RS[r].Qj ← h;} /* espera por instrução */ } else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0;}; RS[r].Busy ← yes; RS[r].Dest ← b; ROB[b].Instruction ← opcode; ROB[b].Dest ← rd;ROB[b].Ready ← no;`
Operações de FP e armazenamentos		`if (RegisterStat[rt].Busy) /* instr. grava rt durante execução */ {h ← RegisterStat[rt].Reorder; if (ROB[b].Ready) /* Instr. já concluída */ {RS[r].Vk ← ROB[h].Value; RS[r].Qk ← 0;} else {RS[r].Qk ← h;} /* Espera por instrução */ } else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0;};`
Operações de FP		`RegisterStat[rd].Qi=b; RegisterStat[rd].Busy ← yes; ROB[b].Dest ← rd;`
Cargas		`RS[r].A ← imm; RegisterStat[rt].Qi=b; RegisterStat[rt].Busy ← yes; ROB[b].Dest ← rt;`
Armazenamentos		`RS[r].A ← imm;`
Executar operação de FP	(Rs[r].Qj = 0) e (Rs[r].Qj = 0)	Calcular resultados – operandos estão em Vj e Vk
Load etapa 1	(Rs[r].Qj = 0) e não existe nenhum armazenamento anterior na fila	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load etapa 2	Etapa 1 de Load concluída e todos os armazenamentos anteriores em ROB têm endereço diferente	Ler de Mem[Rs[r].A]
Store	(Rs[r].Qj = 0) e armazenamento no início da fila	`ROB[h].Address ← RS[r].Vj + RS[r].A;`
Gravar todos os resultados exceto armazenamento	Execução feita em *r* e CDB disponível.	`b ← RS[r].Reorder; RS[r].Busy ← no; ∀x(if (RS[x].Qj=b) {RS[x].Vj ← result; RS[x].Qj ← 0}); ∀x(if (RS[x].Qk=b) {RS[x].Vk ← result; RS[x].Qk ← 0}); ROB[b].Value ← result; ROB[b].Ready ← yes;`
Store	Execução feita em *r* e (Rs[r].Qk = 0)	`ROB[h].Value ← RS[r].Vk;`
Consolidação	A instrução está no início do ROB (entrada h) e ROB[h].ready = yes	`d = ROB[h].Dest; /* dest. de registrador, se existir */ if (ROB[h].Instruction==Branch) {if (branch is mispredicted) {clear ROB[h], RegisterStat; fetch branch dest;};} else if (ROB[h].Instruction==Store) {Mem[ROB[h].Address] ← ROB[h].Value;} else /* insere o resultado no destino ce registrador */ {Regs[d]← ROB[h].Value;}; ROB[h].Busy ← no; /* libera entrada de ROB * /* libera registrador de dest. se nenhuma outra instr. estiver gravando nele */ if (RegisterStat[d].Qi==h) {RegisterStat[d].Busy ← no;};`

Número da iteração	Instruções		Emite em ciclo de clock número	Executa em ciclo de clock número	Acesso de memória em ciclo de clock número	Grava CDB em ciclo de clock número	Comentário
1	LD	R2,0(R1)	1	2	3	4	Primeira emissão
1	DADDIU	R2,R2,#1	1	5		6	Esperar por LW
1	SD	R2,0(R1)	2	3	7		Esperar por DADDIU
1	DADDIU	R1,R1,#4	2	3		4	Executar diretamente
1	BNE	R2,R3,LOOP	3	7			Esperar por DADDIU
2	LD	R2,0(R1)	4	8	9	10	Esperar por BNE
2	DADDIU	R2,R2,#1	4	11		12	Esperar por LW
2	SD	R2,0(R1)	5	9	13		Esperar por DADDIU
2	DADDIU	R1,R1,#4	5	8		9	Esperar por BNE
2	BNE	R2,R3,LOOP	6	13			Esperar por DADDIU
3	LD	R2,0(R1)	7	14	15	16	Esperar por BNE
3	DADDIU	R2,R2,#1	7	17		18	Esperar por LW
3	SD	R2,0(R1)	8	15	19		Esperar por DADDIU
3	DADDIU	R1,R1,#4	8	14		15	Esperar por BNE
3	BNZ	R2,R3,LOOP	9	19			Esperar por DADDIU

Figura 3.33 **Os tempos de emissão, execução e gravação de resultado para uma versão de emissão dual de nosso pipeline *sem* especulação**. Observe que a instrução L.D que segue a instrução BNE não pode iniciar sua execução mais cedo, porque tem de esperar até o resultado do desvio ser determinado. Esse tipo de programa, com desvios dependentes de dados que não podem ser resolvidos mais cedo, mostra a força da especulação. Unidades funcionais separadas para cálculo de endereços, operações da ULA e avaliação de condição de desvio permitem que várias instruções sejam executadas no mesmo ciclo.

Número da iteração	Instruções		Emite em ciclo de clock número	Executa em ciclo de clock número	Acesso de leitura em ciclo de clock número	Grava CDB em ciclo de clock número	Consolida em ciclo de clock número	Comentário
1	LD	R2,0(R1)	1	2	3	4	5	Primeira emissão
1	DADDIU	R2,R2,#1	1	5		6	7	Esperar por LW
1	SD	R2,0(R1)	2	3			7	Esperar por DADDIU
1	DADDIU	R1,R1,#4	2	3		4	8	Consolidar em ordem
1	BNE	R2,R3,LOOP	3	7			8	Esperar por DADDIU
2	LD	R2,0(R1)	4	5	6	7	9	Sem retardo de execução
2	DADDIU	R2,R2,#1	4	8		9	10	Esperar por LW
2	SD	R2,0(R1)	5	6			10	Esperar por DADDIU
2	DADDIU	R1,R1,#4	5	6		7	11	Consolidar em ordem
2	BNE	R2,R3,LOOP	6	10			11	Esperar por DADDIU
3	LD	R2,0(R1)	7	8	9	10	12	O mais cedo possível
3	DADDIU	R2,R2,#1	7	11		12	13	Esperar por LW
3	SD	R2,0(R1)	8	9			13	Esperar por DADDIU
3	DADDIU	R1,R1,#4	8	9		10	14	Executada mais cedo
3	BNE	R2,R3,LOOP	9	13			14	Esperar por DADDIU