

**MO401**

**Arquitetura de Computadores I**

2006

Prof. Paulo Cesar Centoducatte  
[ducatte@ic.unicamp.br](mailto:ducatte@ic.unicamp.br)  
[www.ic.unicamp.br/~ducatte](http://www.ic.unicamp.br/~ducatte)

MO401 - 2007 Revisado MO401 5.1

**MO401**

**Arquitetura de Computadores I**

**Paralelismo em Nível de Instruções**  
 Exploração Dinâmica: Branch Prediction e Múltiplo Issue

"Computer Architecture: A Quantitative Approach" - (Capítulo 3)

MO401 - 2007 Revisado MO401 5.2

**Paralelismo em Nível de Instruções**  
 Exploração Dinâmica

- **Scheduling Dinâmico**
  - Algoritmo de Tomasulo - continuação
- Algoritmo Tomasulo e Branch Prediction
- Múltiplas Instruções Issuing/Cycle
- Esquemas de Branch Prediction
  1. 1-bit Branch-Prediction Buffer
  2. 2-bit Branch-Prediction Buffer
  3. Correlating Branch Prediction Buffer
  4. Tournament Branch Predictor
  5. Branch Target Buffer
  6. Return Address Predictors

MO401 - 2007 Revisado MO401 5.3

**Técnicas para redução de stalls**

Technique	Reduces
Dynamic scheduling	Data hazard stalls
Dynamic branch prediction	Control stalls
Issuing multiple instructions per cycle	Ideal CPI
Speculation	Data and control stalls
Dynamic memory disambiguation	Data hazard stalls involving memory
Loop unrolling	Control hazard stalls
Basic compiler pipeline scheduling	Data hazard stalls
Compiler dependence analysis	Ideal CPI and data hazard stalls
Software pipelining and trace scheduling	Ideal CPI and data hazard stalls
Compiler speculation	Ideal CPI, data and control stalls

Capítulo 3 (red arrow) / Capítulo 4 (blue arrow)

MO401 - 2007 Revisado MO401 5.4

**Algoritmo de Tomasulo**

- IBM 360/91 (1967 - não havia caches; tempo de acesso à memória grande e instruções de FP com grandes latências (delay))
- Idéia: Alto desempenho sem compilador especial
- Um pequeno número de registradores floating point (4 no 360) evita um bom scheduling das operações pelo compilador.
  - Tomasulo: Como ter efetivamente mais registradores? Como resolver os hazards RAW, WAW e WAR?
  - seguir quando os operandos estiverem prontos e renaming implementado no hardware!
- Descendentes:
  - Alpha 21264, HP 8000, MIPS 10000, Pentium III, PowerPC 604, ...

MO401 - 2007 Revisado MO401 5.5

**Algoritmo de Tomasulo**

- Controle & buffers **distribuído** na Function Units (FU)
  - FU buffers chamado de "**Reservation Stations**"; mantem operandos pendentes
- Substituição dos **Registradores** nas instruções por **valores ou apontadores** para a **Reservation Stations (RS)**: denominado **register renaming**;
  - Evita os hazards WAR e WAW
  - Se existe mais reservation stations que registradores, então pode-se fazer otimizações não realizadas pelos compiladores
- Resultados da **RS** para a **FU**, (**sem usar os registradores**), **broadcasts** dos resultados para todas as FUs usando o **Common Data Bus**

MO401 - 2007 Revisado **Load e Stores tratados como FUs com RSs** MO401 5.6

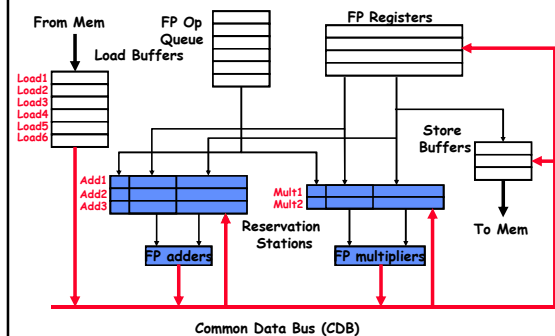
### Algoritmo de Tomasulo Exemplo

- Foco: Unidades de ponto-flutuante e load-store
- Cada estágio pode ter um número arbitrário de ciclos
- Múltiplas unidades funcionais
- Diferentes instruções possuem tempos diferentes no estágio EX
- Unidades disponíveis: load-store; mult e adder

MC401-2007  
Revisado

MC401  
5.7

### Estrutura Básica de uma Implementação do Algoritmo de Tomasulo (para o MIPS)



MC401-2007  
Revisado

MC401  
5.8

### Reservation Station

- Op:** Operação a ser executada na unidade (e.g., + or -)
- Vj, Vk:** Valores dos operandos Fontes
- Store buffers tem campos V, resultados devem ser armazenados
- Qj, Qk:** Reservation stations produzirá os operandos correspondentes (valores a serem escritos)
- Qj, Qk = 0 => ready
  - Store buffers tem somente Qi para RS producing result
- Busy:** Indica que a reservation station e sua FU estão busy
- A:** Mantém informação sobre o end. de memória calculado para load ou store
- Register result status (campo Qi no register file)** — Indica para cada registrador a unidade funcional (reservation station) que irá escreve-lo. Em branco se não há instruções pendentes que escreve no registrador.

MC401-2007  
Revisado

MC401  
5.9

### Exemplo 2: Tomasulo & Loop

```

Loop: LD      F0, 0(R1)
      MULTD   F4, F0, F2
      SD      F4, 0(R1)
      SUBI    R1, R1, #8
      BNEZ   R1, Loop
    
```

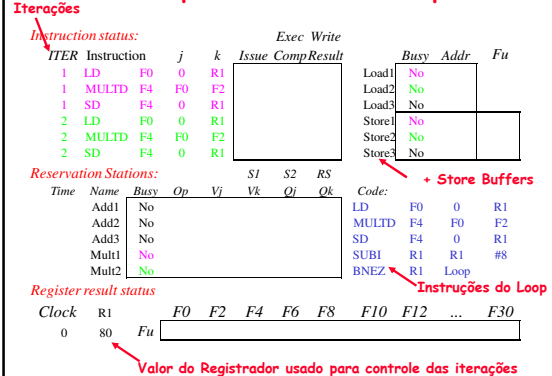
Branch Prediction ~ Taken

- Assuma que Mult gasta 4 clocks
- Assuma que o 1º load gasta 8 clocks (L1 cache miss), 2º load gasta 1 clock (hit)
- Para maior compreensão mostraremos os clocks para SUBI e BNEZ
  - Real: instruções inteiras são tratadas pela unidade de inteiros
  - 2 iterações

MC401-2007  
Revisado

MC401  
5.10

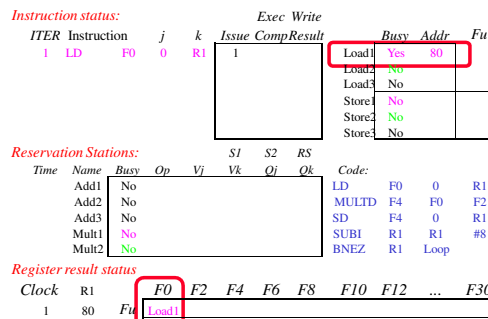
### Exemplo 2: Tomasulo & Loop



MC401-2007  
Revisado

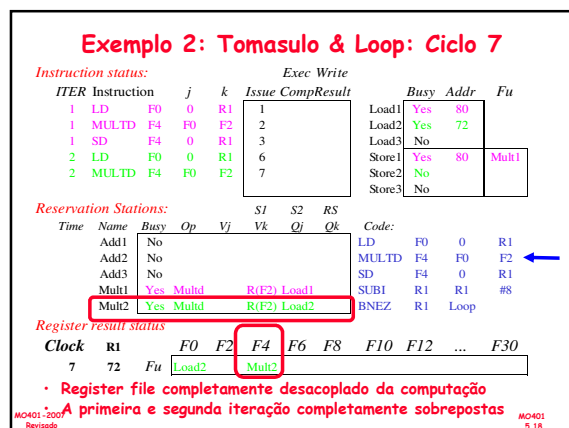
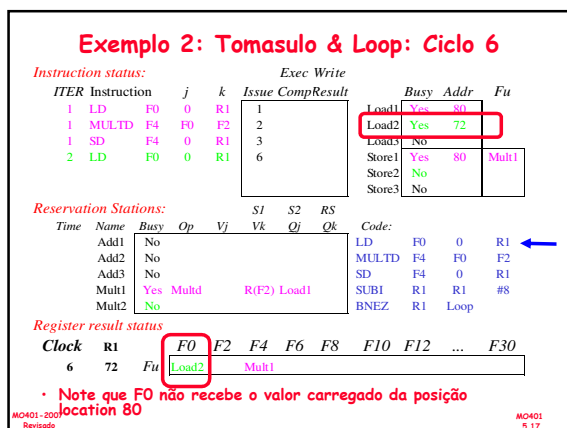
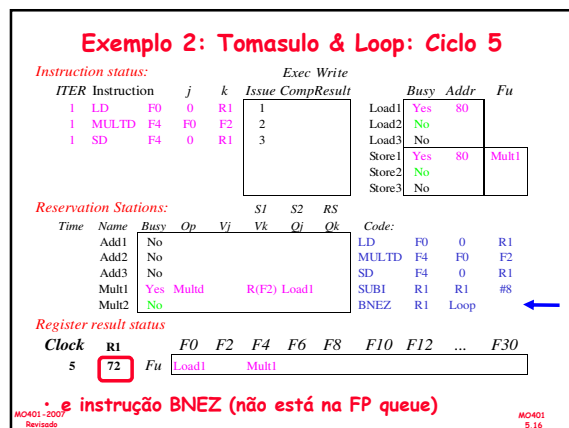
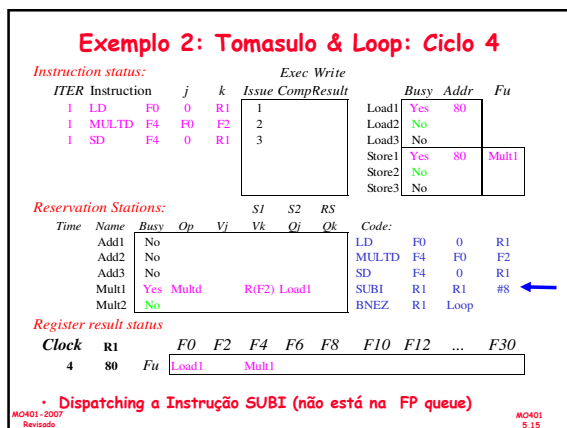
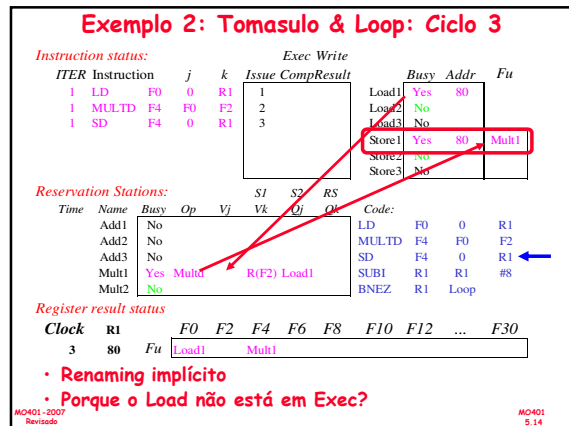
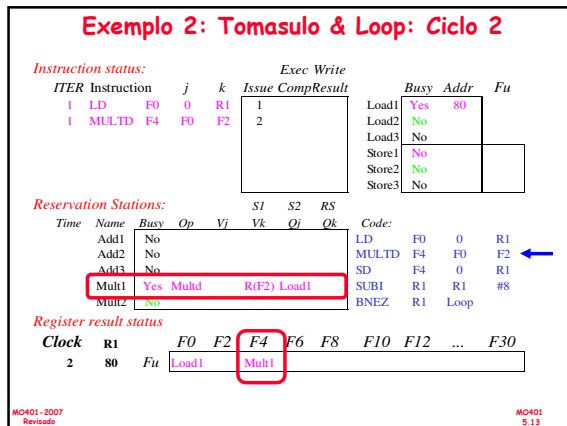
MC401  
5.11

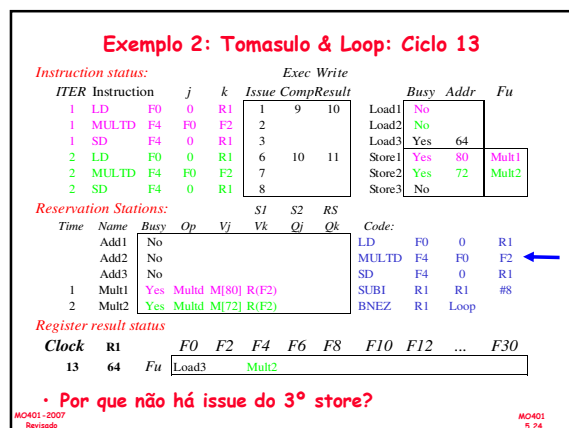
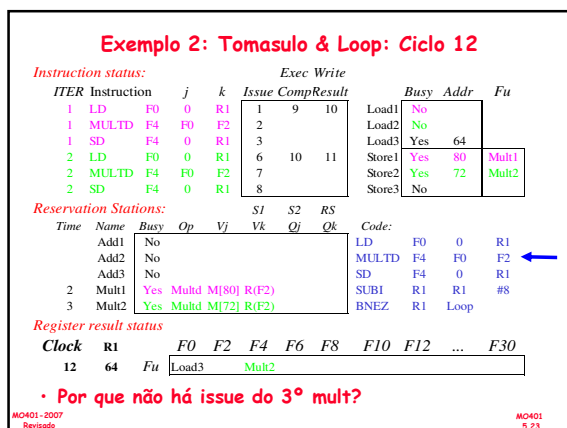
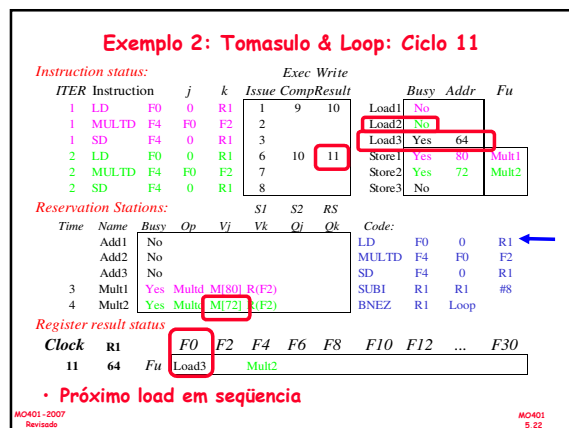
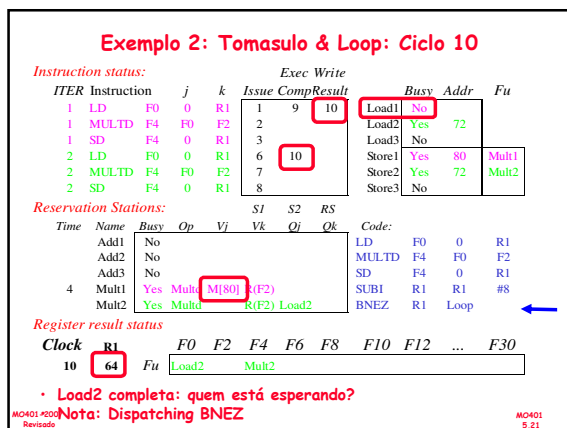
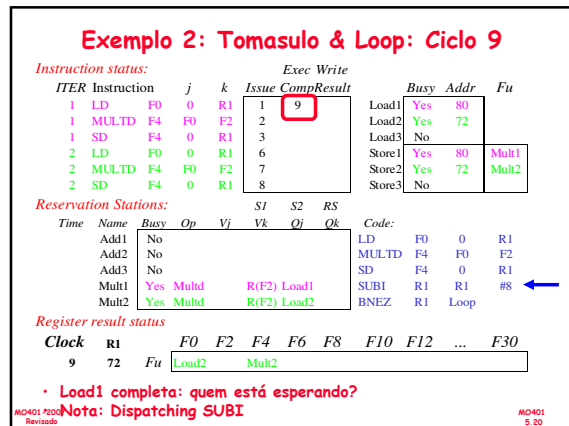
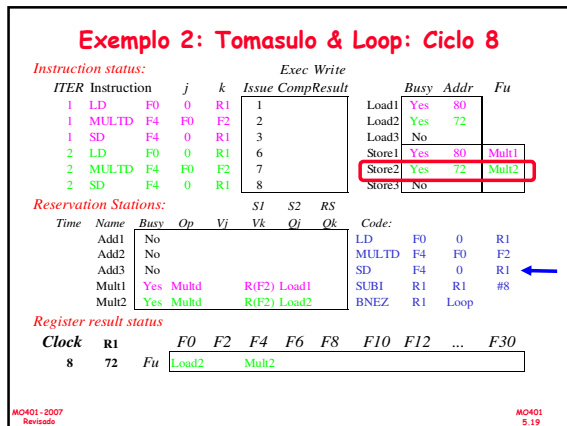
### Exemplo 2: Tomasulo & Loop: Ciclo 1

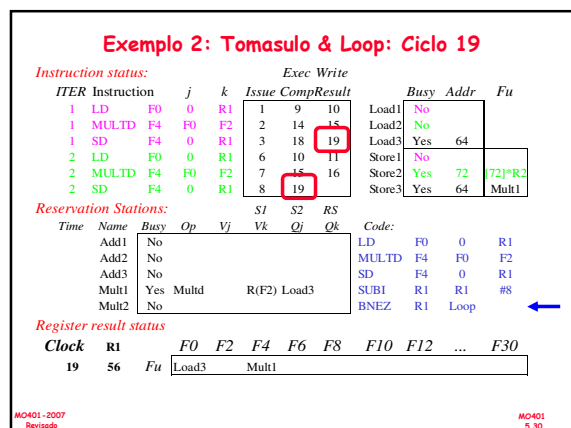
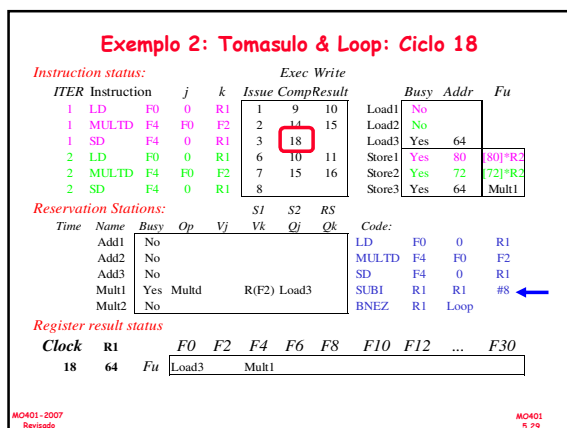
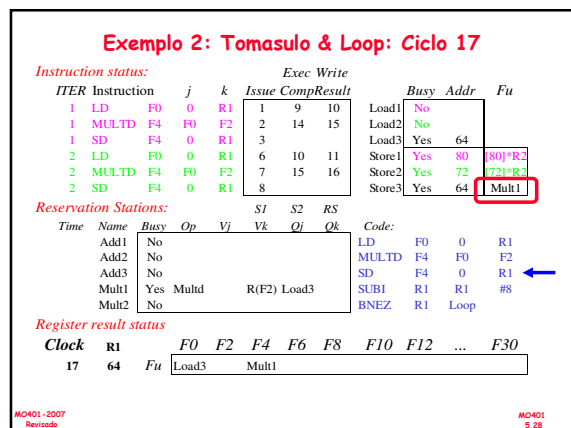
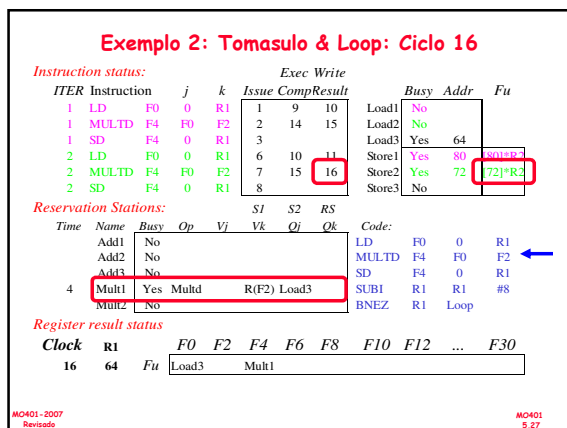
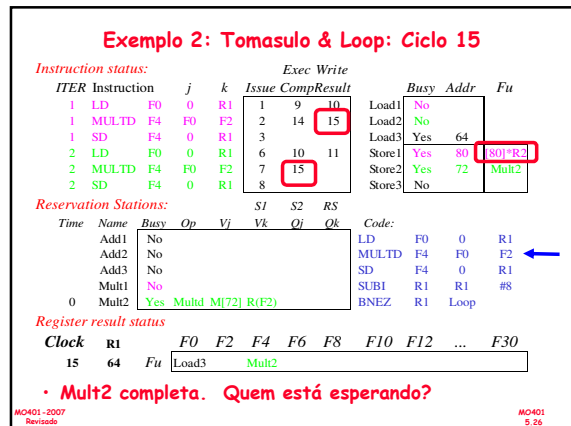
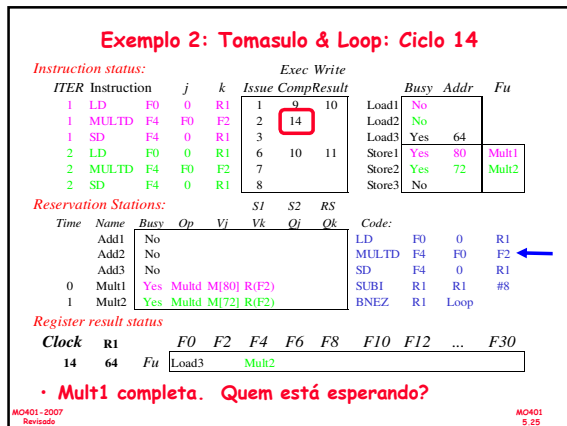


MC401-2007  
Revisado

MC401  
5.12







### Exemplo 2: Tomasulo & Loop: Ciclo 20

**Instruction status:**

ITER	Instruction	j	k	Issue	Comp	Result	Busy	Addr	Fu	
1	LD	F0	0	R1	1	9	10	Load1	Yes	56
1	MULTD	F4	F0	F2	2	14	15	Load2	No	
1	SD	F4	0	R1	3	18	19	Load3	Yes	64
2	LD	F0	0	R1	6	10	11	Store1	No	
2	MULTD	F4	F0	F2	7	15	16	Store2	No	
2	SD	F4	0	R1	8	19	20	Store3	Yes	64

**Reservation Stations:**

Time	Name	Busy	Op	Vj	Vk	Oj	Ok	Code:
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd			R(F2)	Load3	SUBI R1 R1 #8
	Mult2	No						BNEZ R1 Loop

**Register result status**

Clock	R1	F0	F2	F4	F6	F8	F10	F12	...	F30
20	56	Fu	Load1	Mult1						

**In-order issue, out-of-order execution and out-of-order completion.**

MC401-2007 Revisado 5.31

### Por que Tomasulo sobrepõe iterações de loops?

- Register renaming**
  - Múltiplas iterações usam diferentes destinos físicos para os registradores (loop unrolling dinâmico).
- Reservation stations**
  - Permite que instruções adiantem em relação as operações inteiras de controle
  - Valores antigos dos regs estão no buffer - evita WAR stall
- Tomasulo monta o data flow dependency graph on the fly.**

MC401-2007 Revisado 5.32

### O esquema de Tomasulo tem duas grandes vantagens

- Lógica de detecção de hazard distribuída
  - reservation stations e CDB distribuído
  - Se múltiplas instruções esperam por um único resultado elas podem recebe-lo simultaneamente (broadcast no CDB)
  - Se é usado um register file centralizado, as unidades só poderão ler seus resultados quando o barramento estiver disponível.
- Eliminação dos hazards WAW e WAR stalls

MC401-2007 Revisado 5.33

### Interrupções Precisas?

- Tomasulo:**
  - In-order issue,
  - out-of-order execution,
  - out-of-order completion
- É necessário cuidados com aspectos do **out-of-order completion** para que se tenha breakpoints precisos no stream de instruções.

MC401-2007 Revisado 5.34

### Relação entre Interrupções Precisas e Especulação

- Especulação é uma forma de adivinhação.
- Importante para **branch prediction**:
  - É necessário um bom sistema de adivinhação para prevê a direção do branch.
- Se especularmos e estivermos errado, é necessário voltar e reiniciar a execução a partir do ponto em que foi feita a previsão incorreta:
  - Isto é semelhante ao que ocorre nas exceções precisas!
- Técnica para interrupções/exceções precisas e especulação: **in-order completion** ou **commit**

MC401-2007 Revisado 5.35

### Suporte de HW para Interrupções Precisas

- Buffer para os resultados de instruções que não terminaram (uncommitted instructions): **reorder buffer**
  - 3 campos: instr, destino, valor
  - Usar número do **reorder buffer** no lugar da reservation station quando a execução completa
  - Suprir operandos entre **execution complete & commit**
  - (Reorder buffer pode ser **operando source** => mais regs como RS)
  - Instruções **commit**
  - Uma vez que a instrução **commits**, o resultado é colocado no registrador
  - Como resultado é mais fácil desfazer instruções especuladas devido a um "mispredicted branches" ou uma **exceção**

MC401-2007 Revisado 5.36

### Algoritmo de Tomasulo Especulativo 4 passos

- 1. Issue** — pega a instrução da **FP Op Queue**  
Se há **reservation station** e **reorder buffer slot** livres: **issue instr** & envia operandos & **reorder buffer no.** para o destino (este estágio é comumente chamado de "dispatch")
- 2. Execution** — opera sobre os operandos (EX)  
Quando ambos os operandos estão prontos executa; se não monitora o CDB a espera do resultado; quando ambos estiverem na reservation station, executa; verifica se há RAW
- 3. Write result** — termina a execução (WB)  
Escreve, usando o Common Data Bus, em todas FUs que estão esperando por esse valor & **reorder buffer**; marca a reservation station como disponível.
- 4. Commit** — atualiza o registrador com o resultado em **reorder**  
Quando a instr. é a primeira no **reorder buffer** & o resultado está presente: atualiza o registrador com o resultado (ou store na memória) e remove a instr. do **reorder buffer**. Se **Mispredicted branch** então **flushes reorder buffer** (normalmente chamado de "graduation")

MC401-2007 Revisado MC401 5.37

### Qual a complexidade do hardware com Reorder Buffer (ROB)?

- Como encontrar a última versão do registrador?
  - (como definido no artigo Smith) é necessário uma rede de comparação associativa
  - Uso de **register result status buffer** para descobrir qual **reorder buffer** recebeu o valor
- É necessário que o ROB tenha várias portas (como o banco de registradores)

MC401-2007 Revisado MC401 5.38

### Tomasulo: Resumo

- Reservations stations: **register renaming** implícito aumentando o conjunto de regs + **buffering source operands**
  - Evita que os registradores sejam o gargalo
  - Evita **WAR** e **WAW** hazards
  - Implementa **loop unrolling** em HW
- Não se limita a blocos básicos
- Hoje, ajuda nos **cache misses**
  - Não stall para L1 Data cache miss
- Contribuições
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- "Descendentes" do 360/91: Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

MC401-2007 Revisado MC401 5.39

### Algoritmo de Tomasulo e Branch Prediction

- 360/91 prevê branches mas não especula: o pipeline é parado até que o branch seja resolvido
  - Não há especulação; somente instruções podem ser completadas
- Especulação com Reorder Buffer permite que a execução ultrapasse o branch, e que haja descarte se o branch falha
  - Apenas é necessário manter as instruções no buffer até que haja o **commit** do branch

MC401-2007 Revisado MC401 5.40

### Branch Prediction com N instruções Issue por ciclo de clock

1. Branches irão chegar **n** vezes mais rápidos em um processador **n**-issue
2. Lei de Amdahl => o impacto relativo do controle de stalls será maior que a potencial diminuição do CPI em um processador **n**-issue

MC401-2007 Revisado MC401 5.41

### 7 Esquemas de Branch Prediction

1. 1-bit Branch-Prediction Buffer
2. 2-bit Branch-Prediction Buffer
3. Correlating Branch Prediction Buffer
4. Tournament Branch Predictor
5. Branch Target Buffer
6. Return Address Predictors

MC401-2007 Revisado MC401 5.42

### Dynamic Branch Prediction

- Desempenho =  $f(\text{precisão}, \text{custo do misprediction})$
- Branch History Table:** Bits menos significativos do PC usados como índice de uma tabela de valores de 1 bit
  - Informa se o branch foi tomado ou não na última vez
  - Não há comparação do endereço (menos HW, mas pode não ser o branch correto)

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Branch History Table
0	1	1	0	0	1	1	1	0	0	0	0	0	1	0	1	

0xaaa0002 Addi r2, r0, 7  
 Bne r0, r2, 0xffff0002  
 Add r3, r5, r7

MC401-2007 Revisado 5.43

### Dynamic Branch Prediction

- Quando descobre que errou, atualiza a entrada correta, elimina as instruções erradas do pipeline e recomeça o fetch de **0xffff0002**
- Problema:** em um loop, 1-bit BHT irá causar **2 mispredictions** (em média nos loops - na entrada e na saída):
  - No fim do loop quando ele termina
  - Na entrada do loop quando ele prevê *exit* no lugar de looping
  - Em um loop com 10 iterações
    - somente 80% de precisão
    - mesmo que os **Taken** sejam 90% do tempo

MC401-2007 Revisado 5.44

### Dynamic Branch Prediction (Jim Smith, 1981)

- Solução: esquema com 2-bit onde só há troca na previsão se houver duas **misprediction**:

- Vermelho:** not taken
- verde:** taken
- Adicionado uma **Histerese (inércia)** para tomar a decisão

MC401-2007 Revisado 5.45

### Correlating Branches

**Idéia:** **taken/not taken** dos branches recentes são relacionados com o comportamento dos próximos branches (como um histórico do comportamento dos branches)

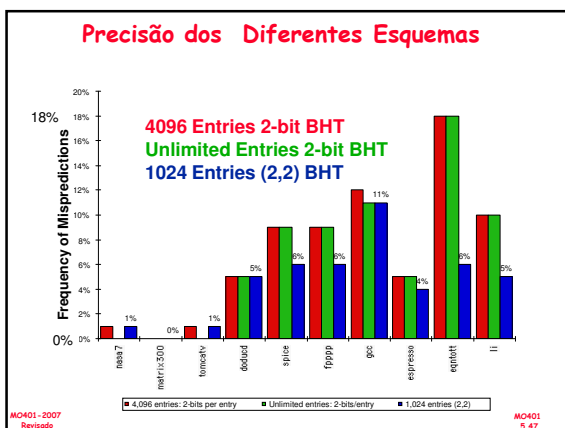
- O comportamento recente dos branches seleciona entre 4 previsões para o próximo branch, atualizando a previsão

**(2,2) predictor:** 2-bit globais e 2-bit locais

```

If (d==0) d=1;
...
If (d=1) ...;
  
```

MC401-2007 Revisado 5.46



### Execução Predicada

- Evita-se branch prediction "colocando-se" os branches em instruções com execução condicional:

**if (x) then A = B op C else NOP**

- Se falso, então nem há **store result** nem causa exceção
- ISA expandido do Alpha, MIPS, PowerPC, SPARC possui **move condicional**; PA-RISC pode anular (**annul**) qualquer instrução seguinte.
- IA-64: **64 1-bit condition fields** seleciona a execução condicional de qq instrução
- Esta transformação é chamada de "**if-conversion**"

**Desvantagens de conditional instructions**

- Gasta um clock se "**annulled**"
- Se a condição é avaliada atrasada -> Stall
- Condições complexas reduzem a eficácia; a condição é conhecida tardiamente no pipeline

MC401-2007 Revisado 5.48



### Precisão BHT

- Mispredict motivado por:
  - Adivinhação errada para o branch
  - Acesso à história do branch errado (índice)
- Uma tabela com 4096 entradas varia de 1% (nasa7, tomcatv) a 18% (eqntott) de misprediction, com 9% para o spice e 12% para o gcc
- Para o SPEC92, 4096 entradas é tão bom quanto infinitas entradas

MC401-2007  
Revisado

MC401  
5.49

### Tournament Predictors

- Motivação: correlating branch predictors falha na previsão de tipos de branches importantes; pode-se melhorar o desempenho provendo informações globais e não somente locais
- Tournament predictors: usa 2 sistemas de predição, 1 baseado em informações globais e 1 baseado em informações locais, combinando-os com o uso de um seletor
- Ajuda a selecionar a previsão correta para o branch certo

MC401-2007  
Revisado

MC401  
5.50

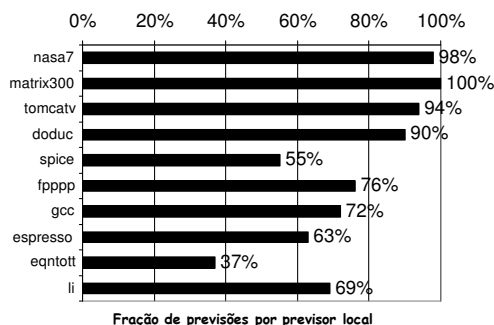
### Tournament Predictor no Alpha 21264

- 4K 2-bit counters para escolha a partir da predição global ou da predição local
- Global predictor: também tem 4K entradas e é indexado por uma história dos últimos 12 branches; cada entrada no global predictor é uma predição standard de 2-bit
  - 12-bit pattern: bit i 0 => i-ésimo branch (anterior) not taken; bit i 1 => i-ésimo branch (anterior) taken;
- Local predictor: consiste de uma predição de 2 níveis:
  - Top level: local history table consistindo de 1024 entradas de 10-bit; cada entrada de 10-bit corresponde aos 10 branches mais recentes para a entrada. História de 10-bit permite casar 10 branches.
  - Next level: A entrada selecionada a partir da local history table é usada como índice da tabela de 1K entradas formando um contador de 3-bit, que fornece a predição local
- Tamanho total:  $4K*2 + 4K*2 + 1K*10 + 1K*3 = 29K$  bits! (~180,000 transistores)

MC401-2007  
Revisado

MC401  
5.51

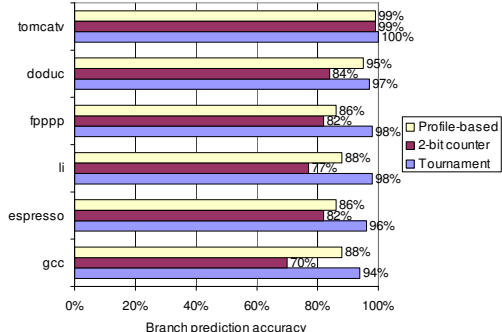
### % de predições a partir da predição local no esquema Tournament Prediction



MC401-2007  
Revisado

MC401  
5.52

### Precisão de Branch Prediction

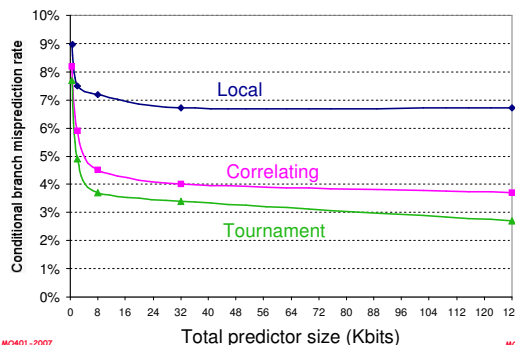


Profile: branch profile feito na última execução

MC401-2007  
Revisado

MC401  
5.53

### Precisão v. Tamanho (SPEC89)



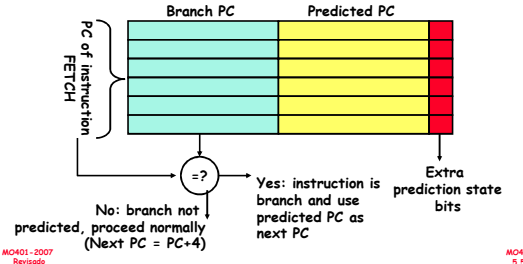
MC401-2007  
Revisado

MC401  
5.54

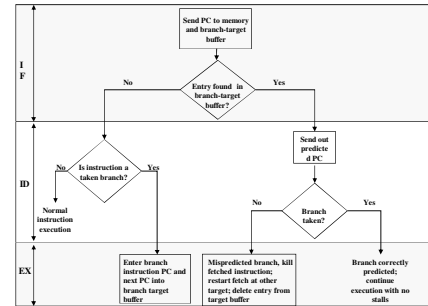
## O Endereço é necessário junto a Predição

- Branch Target Buffer (BTB): Endereço do branch é índice para acesso à predição e ao endereço alvo do branch (se taken)

(Figure 3.19)



## Algoritmo para branch-target buffer



## Caso Especial: Return Addresses

- Register Indirect Branch:** difícil de ser predito o endereço
- SPEC89: 85% dos **register indirect branches** são retornos de procedimentos
- Já que os procedimentos usam uma pilha para seu controle, salvar os **return address** em pequenos buffers do tipo pilha : 8 a 16 entradas tem baixa taxa de miss

MO401-2007 Revisado

MO401 5.57

## Quanto maior (sofisticado) melhor?

- 21264 usa **tournament predictor** (29 Kbits)
- 21164 usa um **simple 2-bit predictor** com 2K entradas (ou um total de 4 Kbits)
- SPEC95 benchmarks: 21264 é melhor
  - 21264 média de 11.5 mispredictions por 1000 instruções
  - 21164 média de 16.5 mispredictions por 1000 instruções
- Transaction processing (TP) !
  - 21264 média de 17 mispredictions por 1000 instruções
  - 21164 média de 15 mispredictions por 1000 instruções

MO401-2007 Revisado

MO401 5.58

## Resumo: Dynamic Branch Prediction

- Predição é um recurso importante na execução escalar
- Branch History Table:** 2 bits para melhorar a precisão em loops
- Correlation:** Explora a correlação entre loops executados recentemente e o próximo loop.
  - Mesmo que sejam branches diferentes
  - Ou execuções do mesmo branch
- Tournament Predictor:** mais recursos para soluções que competem entre si, escolha de uma solução
- Branch Target Buffer:** inclui o **branch address** & a predição
- Predicated Execution:** pode reduzir o número de branches e o número de **mispredicted branches**
- Return address stack:** usado em predição de jumps indiretos

MO401-2007 Revisado

MO401 5.59