

MO401
Arquitetura de Computadores I

2006
 Prof. Paulo Cesar Centoducatte
ducatte@ic.unicamp.br
www.ic.unicamp.br/~ducatte

MO401-2007 Revisado MO401 4.1

MO401
Arquitetura de Computadores I

Paralelismo em Nível de Instruções
Exploração Dinâmica

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

MO401-2007 Revisado MO401 4.2

Paralelismo em Nível de Instruções
Exploração Dinâmica

- **ILP - Conceitos Básicos**
- **Revisão Pipelining**
- **Instruction-Level Parallelism - ILP**
 - Formas de exploração de ILP
 - » Direta
 - » Vetorial
 - »
- **Hazards e ILP**
 - Limites
- **Scheduling Dinâmico**
- **Algoritmo de Tomasulo**

MO401-2007 Revisado MO401 4.3

Conceitos Básicos

- **Instruction Level Parallelism - ILP**
 - Sobreposição na execução de instruções
 - » Parcial - Pipeline
 - » Total - execução paralela de instruções
- **Pipeline**
 - Técnica básica utilizada para exploração de ILP
- **Diversas técnicas estendem as idéias de pipeline aumentando a exploração de ILP**
 - Dinâmica (uso intensivo de Hardware)
 - Estática (uso intensivo de Software - compiladores)
- **Dinâmica** - desktop e servidores
 - Pentium III e 4; Athlon; MIPS R1000 e R12000, PowerPC 603; UltraSparc III; Alpha 21264; ...
- **Estática**
 - Sistemas Embarcados
 - IA-64; Itanium; VLIW; ...

MO401-2007 Revisado MO401 4.4

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

MO401-2007 Revisado MO401 4.5

Revisão Pipelining

Pipeline CPI = Ideal pipeline CPI + Structural Stalls + Data Hazard Stalls + Control Stalls

- **Ideal pipeline CPI:** desempenho máximo atribuído a uma implementação
- **Structural hazards:** O HW não pode suportar uma dada combinação de instruções
- **Data hazards:** Uma Instrução depende do resultado de uma instrução anterior que ainda está no pipeline
- **Control hazards:** Causado pelo "delay" entre o "fetching" da instrução e a decisão se o fluxo de controle vai ser alterado ou não (branches and jumps)

MO401-2007 Revisado MO401 4.6

Instruction-Level Parallelism (ILP)

- Block Básico (BB) – reduz a disponibilidade de ILP
 - BB: conjunto de instruções executadas seqüencialmente sem desvios exceto na sua entrada e na sua saída
 - **Frequência dinâmica média dos branches: 15% a 25%**
- ⇒ 4 a 7 instruções executadas entre um par de branches
- Há dependências entre instruções em um BB
- Para obter um bom desempenho deve-se explorar ILP através dos blocos básicos.

MO401-2007
Revisado

MO401
4.7

Instruction-Level Parallelism (ILP)

- Forma mais simples: **loop-level parallelism**
 - Explora paralelismo através das iterações de um loop

```
For (i=1; i<=1000; i=i+1)
    x[i] = x[i] + y[i];
```

- **Computação Vetorial**
- **Dynamic (static) branch prediction com loop unrolling (compilador)**

MO401-2007
Revisado

MO401
4.8

Dependência de Dados e Hazards

- Instr_J é **data dependent** da Instr_I
Instr_J lê operando antes de Instr_I escreve-lo

```

I: add r1, r2, r3
  → J: sub r4, r1, r3
```

- ou Instr_J é **data dependent** de Instr_K que é dependente de Instr_I
- Causada por uma "True Dependence" (termo de compiladores)
- Se uma dependência verdadeira causa um hazard no pipeline, o chamamos de **Read After Write (RAW) hazard**

MO401-2007
Revisado

MO401
4.9

Dependência de Dados e Hazards

Loop: L.D	F0,0(R1)	↻	
ADD.D	F4,F0,F2	↻	
S.D	F4,0(R1)		Quais as Dependências?
ADD.D	F1,F3,F4		
DADDUI	R1,R1,#-8	↻	
BNE	R1,R2,Loop	↻	

Dados de Ponto-Flutuante

Dados Inteiros

MO401-2007
Revisado

MO401
4.10

Dependência de Dados e Hazards

- Dependências são próprias dos **programas**
- A presença de dependências indica um **potencial** para a ocorrência de um **hazard**. Hazard e sua penalidade (**em stalls**) são próprios do **pipeline**
- Importância das dependências de dados
 - 1) indicam a possibilidade de um hazard
 - 2) determina a ordem na qual os resultados devem ser calculados
 - 3) determinam um **upper bound** para o paralelismo que pode ser explorado.
- **Normalmente se usa esquemas em HW para evitar hazards**

MO401-2007
Revisado

MO401
4.11

Dependência de Dados e Hazards

- Identificação das Dependências
 - Em registradores - mais fácil
 - Em posições de memória - mais difícil
 - » 100(R4) e 20(R6) - podem ser iguais
 - » 20(R4) e 20(R4) - podem ser diferentes

Como supera-las?

- Mantendo-se as dependências porém evitando-se os hazards
 - Escalonamento
 - » Dinâmico - Hardware
 - » Estático - Software (compilador)
- Eliminando-se as dependências por alguma transformação do código

MO401-2007
Revisado

MO401
4.12

Dependência de Nome

• Name Dependence:

Quando 2 instruções usam o mesmo registrador ou posição de memória e não há fluxo de dados entre as instruções associadas com o nome:

Há 2 versões de **Dependência de Nome**

- Anti-dependence
- Output Dependence

MO401-2007
Revisado

MO401
4.13

Anti-Dependência

- Instr_J escreve operando **antes** que Instr_I a leia

```
I: sub r4, r1, r3
J: add r1, r2, r3
K: mul r6, r1, r7
```

- Chamada de "anti-dependence" pelos projetistas de compiladores.
Resulta do reuso do nome "r1"
- Se a anti-dependência causa um hazard no pipeline, ele é chamado de **Write After Read (WAR)** hazard

MO401-2007
Revisado

MO401
4.14

Dependência de Saída

- Instr_J escreve o operando **antes** que a Instr_I o escreva.

```
I: sub r1, r4, r3
J: add r1, r2, r3
K: mul r6, r1, r7
```

- Chamada de "output dependence" pelos projetistas de compiladores.
Resulta do reuso do nome "r1"
- Se a dependência de saída causa um hazard no pipeline, ele é chamado de **Write After Write (WAW)** hazard

MO401-2007
Revisado

MO401
4.15

ILP e Hazards de Dados

- HW/SW devem preservar a "ordem do programa": a "ordem" de execução das instruções deve ser a mesma da execução seqüencial (1 por vez) como definida no código fonte original do programa.
- Estratégia do HW/SW: deve explorar paralelismo preservando a ordem do program **somente onde ela afeta a saída do programa**
- Instruções envolvidas em uma dependência de nome podem ser executadas simultaneamente **se o nome usado nas instruções for trocado** de forma a eliminar o conflito
 - Register renaming - resolve a dependência de nome para regs
 - Pelo compilador ou por HW

MO401-2007
Revisado

MO401
4.16

Dependências de Controle

- Todas as instruções, de alguma forma, são dependentes de controle devido a um conjunto de branches e, em geral, essa dependência de controle deve ser preservada para se preservar a ordem do programa

```
if p1 {
    S1;
};
if p2 {
    S2;
}
```

- S1 é dependente de controle em p1 e S2 é dependente de controle em p2 porém não o é em p1.

MO401-2007
Revisado

MO401
4.17

Ignorando Dependências de Controle

- Dependências de controle não precisam ser preservadas quando a execução da instrução que não deveria ser executada (violando a dependência de controle) **não afeta** a corretude do programa
- 2 propriedades críticas para corretude de um programa:
 - Fluxo dos dados
 - Comportamento na Presença de Exceções

MO401-2007
Revisado

MO401
4.18

Comportamento em Exceções

- Preservando o comportamento sob exceções => qualquer alteração na ordem de execução das instruções **não deve alterar como as exceções são geradas** no programa (=> **não deve haver novas exceções**)

Exemplo:

```
DADDU    R2, R3, R4
BEQZ    R2, L1
LW      R1, 0(R2)
L1:
```

- Qual o problema se movermos o LW para antes do BEQZ (não há dependência de dados entre BEQZ e LW)?
 - Pode ser alterado o resultado do programa!
 - Se o lw gerar uma exceção devido a proteção de memória haverá uma exceção que não deveria existir.

MC401-2007
Revisado

MC401
4.19

Fluxo de Dados

- Data flow:** fluxo de dados através das instruções que produzem resultados e que os utilizam (consomem)
 - branches fazem o fluxo ser dinâmico, determina qual instrução dinamicamente é fornecedora do dado

Exemplo:

```
DADDU    R1, R2, R3
BEQZ    R4, L
DSUBU    R1, R5, R6
L: ...
OR      R7, R1, R8
```

- OR depende de DADDU ou de DSUBU?

- O fluxo de dados deve ser preservado na execução.

MC401-2007
Revisado

MC401
4.20

Fluxo de Dados

Exemplo:

```
DADDU    R1, R2, R3
BEQZ    R12, L
DSUBU    R4, R5, R6
DADDU    R5, R4, R9
L:
OR      R7, R8, R9
```

- Suponha que (R4) não é usado após o rótulo L e que DSUBU não gera exceção neste caso.

- O fluxo de dados não é afetado se trocarmos a ordem das instruções BEQZ e DSUBU

MC401-2007
Revisado

MC401
4.21

Vantagens de Scheduling Dinâmico

- Trata de casos que não são conhecidos em tempo de compilação
 - Casos que envolvem referências à memória

- Simplifica o compilador

- Permite que um código compilado para um pipeline execute de forma eficiente em um pipeline diferente

- Hardware speculation - técnica com bom desempenho que usa scheduling dinâmico como base

MC401-2007
Revisado

MC401
4.22

HW : Paralelismo de Instruções

- Idéia Principal: permitir que instruções após a que está em "stall" prosigam

```
DIVD    F0, F2, F4
ADD     F10, F0, F8
SUBD   F12, F8, F14
```

- Habilitar **out-of-order execution** e permitir **out-of-order completion**

- Diferenciar quando uma instrução **inicia a execução** e quando ela **completa a execução** em 2 tempos, em ambos ela está **em execução**

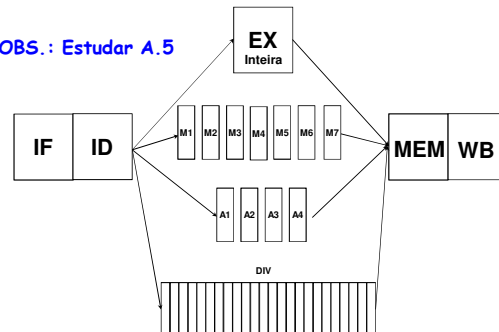
- Em um pipeline com schedule dinâmico todas as instruções passam pelo estágio **issue** (decodificação, **hazard estrutural?**) em ordem (**in-order issue**)

MC401-2007
Revisado

MC401
4.23

Scheduling Dinâmico Implementação - MIPS

OBS.: Estudar A.5



MC401-2007
Revisado

MC401
4.24

HW : Paralelismo de Instruções

- **out-of-order execution:** possibilita a ocorrência de hazards WAR e WAW

```
div.d    f0,f2,f4
add.d    f6,f0,f8
sub.d    f8,f10,f14
mul.d    f6,f10,f8
```

add.d e sub.d => (f8) WAR
 - add.d espera por div.d
 add.d e mul.d => (f6) WAW

MC401-2007
Revisado

MC401
4.25

HW : Paralelismo de Instruções

- **out-of-order completion:** problemas com execuções
- **out-of-order completion** deve preservar o comportamento sob execuções como se fosse executado em uma máquina **in-order**
- **Processadores com scheduling dinâmico** preservam o comportamento sob execuções garantindo que as instruções não possam gerar exceções até que o processador saiba que a instrução que gerou a exceção está sendo completada.

MC401-2007
Revisado

MC401
4.26

HW : Paralelismo de Instruções

- **Processadores com scheduling dinâmico** podem gerar exceções imprecisas: uma exceção é dita imprecisa se o estado do processador quando ela foi gerada não corresponde exatamente ao estado que ela ocorreria se a instrução fosse executada sequencialmente.
Exemplo:

- Uma instrução fora de ordem já foi completada e uma instrução anterior a ela gera a exceção
- Uma instrução fora de ordem ainda não foi completada e uma instrução posterior a ela gera a exceção

MC401-2007
Revisado

MC401
4.27

Scheduling Dinâmico Implementação - MIPS

- Pipeline simples tem 1 estágio que verifica se há hazard estrutural e de dados: **Instruction Decode (ID)**, também chamado de **Instruction Issue**
- Quebrar o estágio **ID** do pipeline de 5 estágios em dois estágios:
 - **Issue** — Decodificação das instruções, verificação de hazards estruturais
 - **Read operands** — Espera até não haver data hazards, então lê os operandos

MC401-2007
Revisado

MC401
4.28

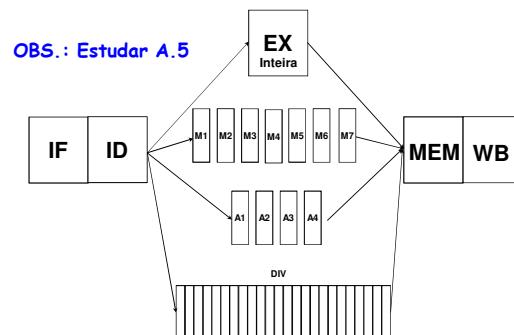
Scheduling Dinâmico Implementação - MIPS

- Estágio **EX** segue o de leitura de operandos como no pipeline simples.
- **OBS.:**
 - A execução pode levar múltiplos ciclos, dependendo da instrução
 - O pipeline permite múltiplas instruções em EX, tem múltiplas unidades funcionais (FUs)

MC401-2007
Revisado

MC401
4.29

Scheduling Dinâmico Implementação - MIPS



MC401-2007
Revisado

MC401
4.30

Algoritmo Dinâmico : 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 RAW?
 - seguir quando os operandos estiverem prontos e renaming implementado no hardware!
- Descendentes:
 - Alpha 21264, HP 8000, MIPS 10000, Pentium III, PowerPC 604, ...

MC401-2007
Revisado

MC401
4.31

Algoritmo de Tomasulo

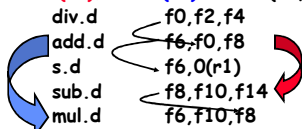
- Controle & buffers **distribuídos** na Function Units (FU)
 - FU buffers chamado de "**reservation stations**"; mantém operandos pendentes
- Substituição dos **Registradores** nas instruções por **valores** ou **apontadores** para a "**reservation stations (RS)**"; denominado **register renaming** :
 - 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**
- Load e Stores tratados como FUs com RSs

MC401-2007
Revisado

MC401
4.32

Algoritmo de Tomasulo Register Rename

- **WAR(f8): WAW(f6) e RAW(f0, f6 e f8)**



- **Eliminando WAR e WAW - register rename**

- Suponha dois registradores temporários S e T

```

div.d   f0, f2, f4
add.d   S, f0, f8
s.d     S, 0(r1)
sub.d   T, f10, f14
mul.d   f6, f10, T
    
```

- 1) F8 deve ser substituído por T no resto do código - requer análise mais sofisticada (branches, ...)
- 2) Neste exemplo o **register rename** pode ser realizado pelo compilador (análise estática)

MC401-2007
Revisado

MC401
4.33

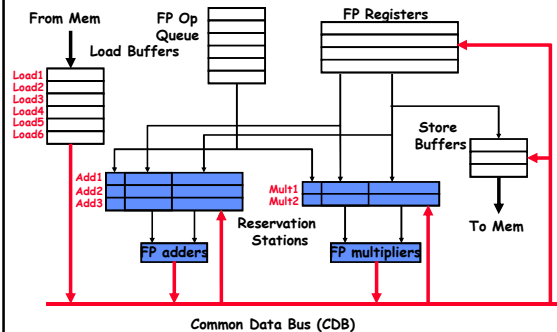
Algoritmo de Tomasulo Exemplo

- 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
4.34

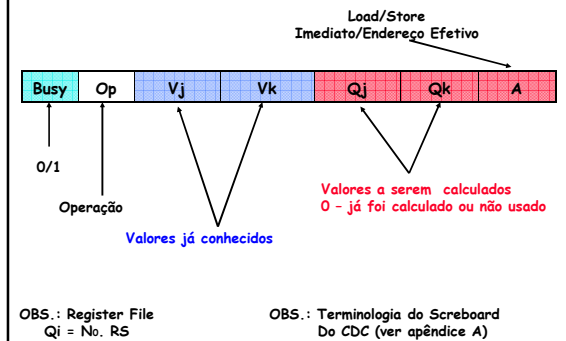
Estrutura Básica de uma Implementação do Algoritmo de Tomasulo (para o MIPS)



MC401-2007
Revisado

MC401
4.35

Reservation Station



OBS.: Register File
Qj = No. RS

OBS.: Terminologia do Screenshot
Do CDC (ver apêndice A)

MC401-2007
Revisado

MC401
4.36

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 ocupadas
- 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
4.37

3 estágios do algoritmo de Tomasulo

- 1.Issue**— pega a instrução na "FP Op Queue"
Se a reservation station está livre (não há hazard estrutural), issues instr & envia operandos (**renames registers**)
- 2.Execute**—executa a operação sobre os operandos (EX)
Se os dois operandos estão prontos executa a operação;
Se não, monitora o **Common Data Bus** (espera pelo cálculo do operando, essa espera resolve RAW)
(quando um operando está pronto -> reservation table)
- 3.Write result** — termina a execução (WB)
Broadcast via **Common Data Bus** o resultados para todas unidades; marca a reservation station como disponível

MC401-2007
Revisado

MC401
4.38

3 estágios do algoritmo de Tomasulo

- data bus normal: dado + destino ("go to" bus)
- **Common data bus:** dado + **source** ("come from" bus)
 - 64 bits de dados + 4 bits para endereço da Funcional Unit
 - Escreve se há casamento com a Funcional Unit (produz resultado)
 - broadcast

MC401-2007
Revisado

MC401
4.39

Exemplo do Alg. Tomasulo

- Trecho de programa a ser executado:

```

1 L.D   F6,34(R2)
2 L.D   F2,45(R3)
3 MUL.D F0,F2,F4
4 SUB.D F8,F2,F6
5 DIV.D F10,F0,F6
6 ADD.D F6,F8,F2
    
```

RAW?: (1-4); (1-5); (2-3); (2-4); (2-6);
 WAW?: (1-6)
 WAR?: (5-6)

MC401-2007
Revisado

MC401
4.40

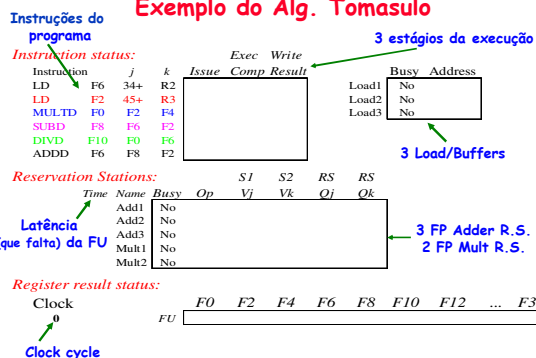
Exemplo do Alg. Tomasulo

- **Assumir as seguintes latências:**
 - Load: 1 ciclo
 - Add: 2 ciclos
 - Multiplicação: 10 ciclos
 - Divisão: 40 ciclos
- **Load-Store:**
 - Calcula o endereço efetivo (FU)
 - Load ou Store buffers
 - Acesso à memória (somente load)
 - Write Result
 - » Load: envia o valor para o registrador e/ou reservation stations
 - » Store: escreve o valor na memória
 - » (escritas somente no estágio "WB" - simplifica o algoritmo de Tomasulo)

MC401-2007
Revisado

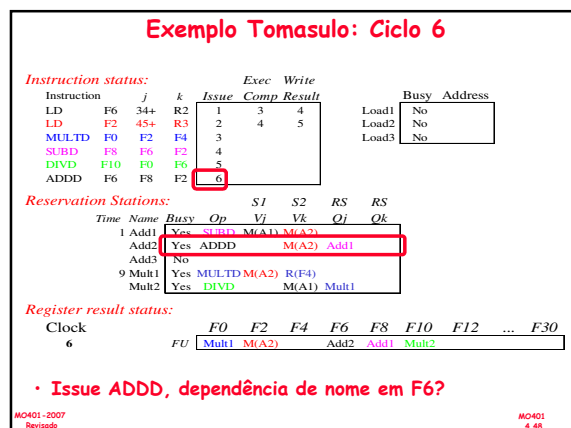
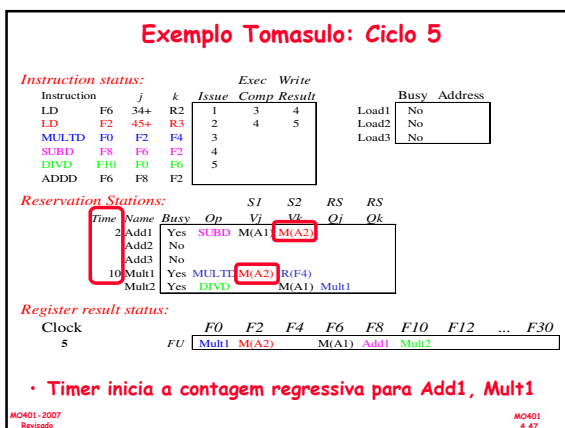
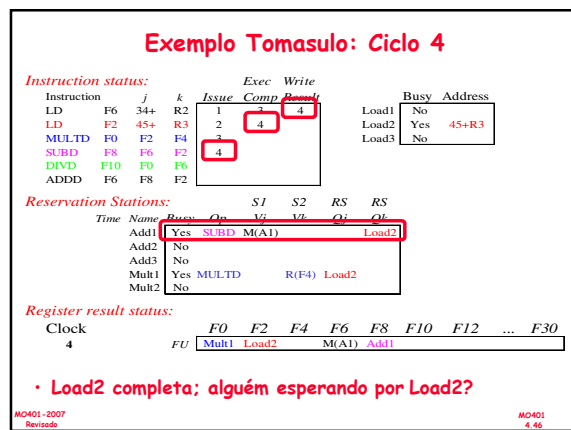
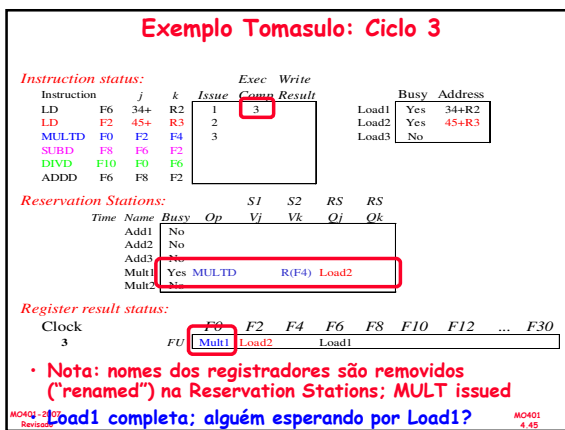
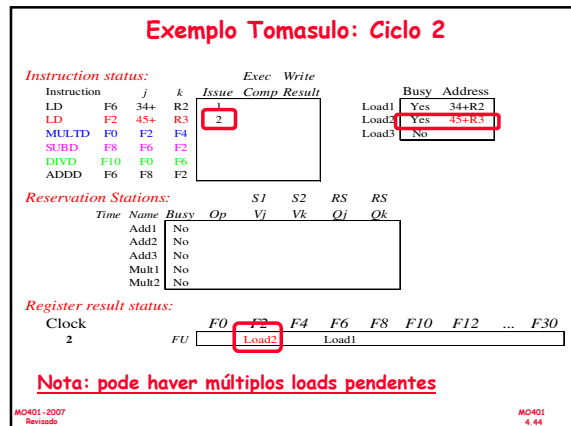
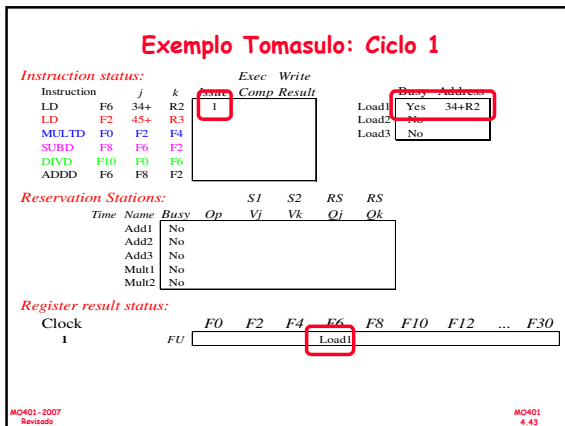
MC401
4.41

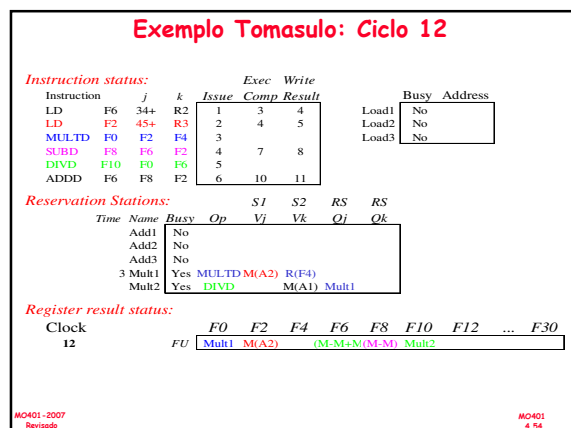
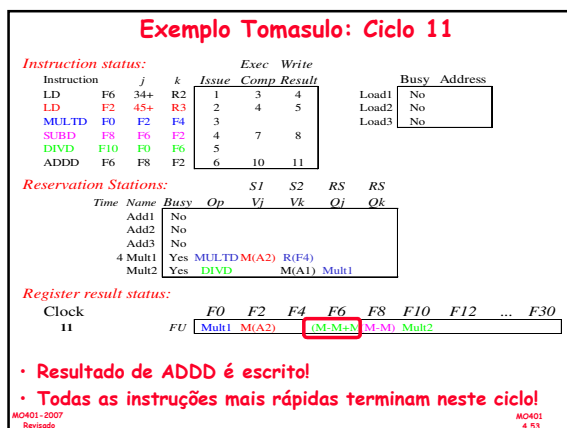
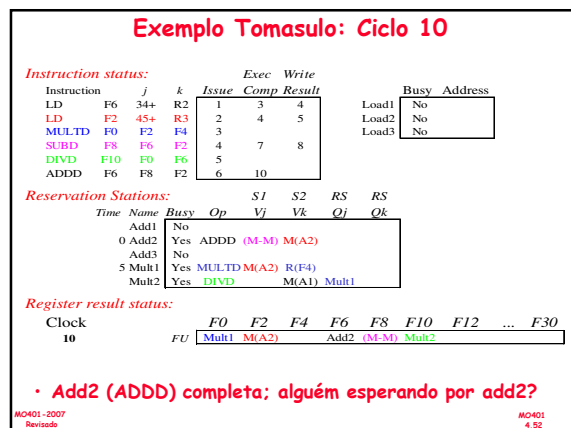
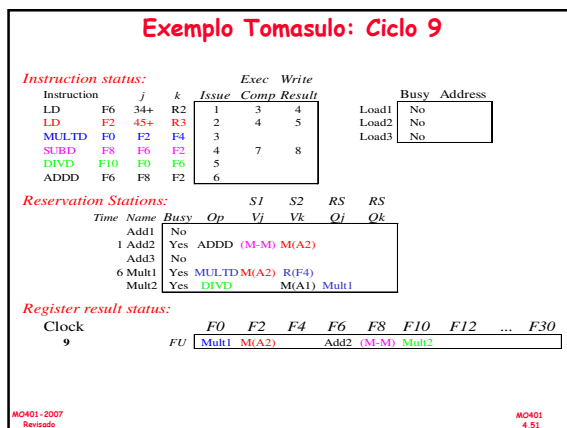
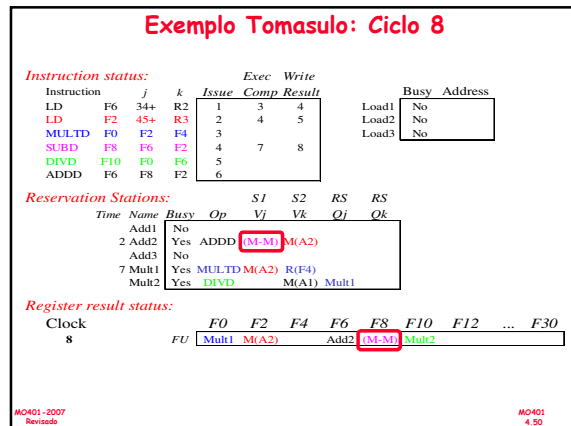
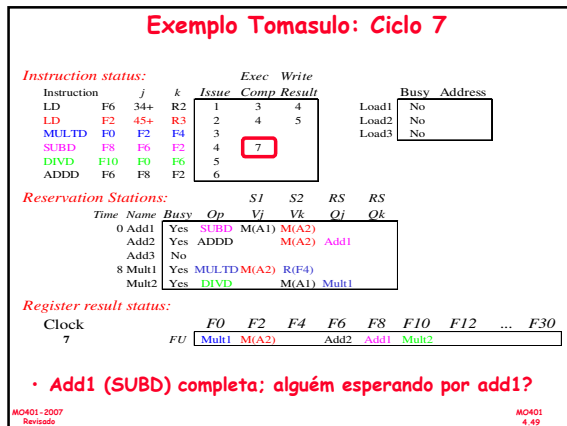
Exemplo do Alg. Tomasulo



MC401-2007
Revisado

MC401
4.42





Exemplo Tomasulo: Ciclo 13

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3						
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qi	Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1	M(A2)	(M-M)+V(M-M)	Mult2				

M0401-2007 Revisado M0401 4.55

Exemplo Tomasulo: Ciclo 14

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3						
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qi	Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU	Mult1	M(A2)	(M-M)+V(M-M)	Mult2				

M0401-2007 Revisado M0401 4.56

Exemplo Tomasulo: Ciclo 15

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15					
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qi	Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD	M(A1)	Mult1		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(A2)	(M-M)+V(M-M)	Mult2				

• Mult1 (MULTD) completa; alguém esperando por mult1?

M0401-2007 Revisado M0401 4.57

Exemplo Tomasulo: Ciclo 16

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15	16				
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qi	Qk
	Add1	No					
	Add2	No					
	Add3	No					
40	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(A2)	(M-M)+V(M-M)	Mult2				

• Agora é só esperar que Mult2 (DIVD) complete

M0401-2007 Revisado M0401 4.58

**Plutando alguns ciclos
(façam como exercício os ciclos faltantes?)**

M0401-2007 Revisado M0401 4.59

Exemplo Tomasulo: Ciclo 55

Instruction status:

Instruction	j	k	Exec Write			Load1	Load2	Load3	Busy	Address
			Issue	Comp	Result					
LD	F6	34+	R2	1	3	4	No	No	No	
LD	F2	45+	R3	2	4	5	No	No	No	
MULTD	F0	F2	F4	3	15	16				
SUBD	F8	F6	F2	4	7	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qi	Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(A2)	(M-M)+V(M-M)	Mult2				

M0401-2007 Revisado M0401 4.60

